TU WIEN Informatics

Analysis, Design and Prototypical Implementation of a Performant and Secure IoT System for Novel Martial Arts Equipment

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Martin Mattäus Śmiech, BSc
Matrikelnummer 01426853

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr. Thomas Grechenig
Mitwirkung: Dipl.-Ing. Mag. Roland Breiteneder
Dr. Dominik Hölbling, MSc., Bakk.

Wien, 25. September 2022

Martin Mattäus Śmiech | Thomas Grechenig

TU WIEN Informatics

Analysis, Design and Prototypical Implementation of a Performant and Secure IoT System for Novel Martial Arts Equipment

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Martin Mattäus Śmiech, BSc
Registration Number 01426853

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr. Thomas Grechenig
Assistance: Dipl.-Ing. Mag. Roland Breiteneder
Dr. Dominik Hölbling, MSc., Bakk.

Vienna, 25th September, 2022

Martin Mattäus Śmiech | Thomas Grechenig

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Martin Mattäus Śmiech, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. September 2022

Martin Mattäus Śmiech

Danksagung

Zuallererst danke ich Roland Breiteneder und Dominik Hölbling für die Betreuung und Unterstützung bei der Umsetzung dieser Diplomarbeit. Ebenfalls danke ich Thomas Grechenig ohne den die Rahmenbedingungen zur Umsetzung der Diplomarbeit nicht geschaffen worden wären.

Außerdem bedanke ich mich bei Dea Čizmić für die regelmäßige Unterstützung während der Umsetzung der Diplomarbeit und auch für die Zusammenarbeit über die gesamte Studienzeit hinweg. Für das gründliche Korrekturlesen bedanke ich mich bei Fabian Guschlbauer. Weiters bedanke ich mich bei René Baranyi, Matej Kosco, Philipp Habinger und Christoph Lichtemannegger für die Beratung und Feedback zur technischen Umsetzung sowie zur Festlegung des State-of-the-Art.

Besonderer Dank geht auch an meine Lehrer, welche mich im richtigen Maß gefördert und gefordert haben im schulischen Werdegang und ohne die ich es nicht so weit gebracht hätte - insbesondere Robert Grüneis, Andrea Meyrhuber, Wolfgang Zachl, Irene Dunzinger und Gertraud Niedergesäß.

Abschließend danke ich auch meiner Mutter, Anna Śmiech, für die Unterstützung über die gesamte Ausbildungszeit und darüber hinaus. Ich danke und gedenke meinen Großeltern, Janusz Szumlakowski und Marta Szumlakowska, sowie meinem Onkel, Piotr Kozak, die den Abschluss meines Studiums nicht mehr erleben durften.

Acknowledgements

First of all, I would like to thank Roland Breiteneder and Dominik Hölbling for advising and supporting me during the course of this thesis. I would also like to thank Thomas Grechenig without whom the circumstances for creating this thesis would not have been met.

Furthermore, I would like to thank Dea Čizmić for regular support during the work at this thesis as well as the collaboration throughout the entirety of my studies. For thoroughly proofreading the thesis I would like to thank Fabian Guschlbauer. Moreover, I would also like to thank René Baranyi, Matej Kosco, Philipp Habinger and Christoph Lichtemannegger for the advice and consultation on the implementation of the prototype and establishing the state-of-the-art.

Special thanks to my teachers who profoundly shaped my career path and without whom I would not have made it this far - in particular Robert Grüneis, Andrea Meyrhuber, Wolfgang Zachl, Irene Dunzinger and Gertraud Niedergesäß.

Last but not least I would also like to thank my mother, Anna Śmiech, who supported me throughout my entire education and beyond. I thank my grandparents, Janusz Szumlakowski and Marta Szumlakowska, as well as my uncle, Piotr Kozak, who are no longer with us and can not see me graduate.

Kurzfassung

Das schnelle und sichere Auswerten von Sportdaten ist eine komplexe technische Herausforderung, besonders dann, wenn Datenaufzeichnung und -verarbeitung räumlich verteilt stattfinden. Der technische Fortschritt über verschiedene Domänen hinweg ermöglicht nun die präzise und effiziente Erfassung sowie Übertragung bewegungsspezifischer Daten. Im Kampfsportbereich ist dies jedoch noch ein verhältnismäßig neues Konzept und es gibt nur wenige etablierte Lösungen und technische Standards, die Sportler:innen, deren Trainer:innen und eventuelle weitere Sportveranstaltungsbeteiligte wie Schiedsrichter:innen unterstützen.

In dieser Arbeit wurden mehrere Exemplare neuartiger sensorbesetzter IoT-Boxhandschuh-Prototypen eingesetzt. Diese emittieren Daten über Bewegungen und Schlagkraft der Träger:innen. Daraus ergibt sich die Forschungsfrage, ob ein zeitgemäßes System, das derartige Daten in einer definierten Zeitspanne unter Einhaltung von Sicherheitsmaßnahmen übertragen und anzeigen kann, im Kampfsportbereich realisierbar ist.

Zur Beantwortung dieser Frage wurde ein wissenschaftlicher methodischer Vorgang bestehend aus Expert:innenbefragung, Modellierung, prototypischer Implementierung, Laborexperimenten sowie Messung davon (mit Hilfe von State-of-the-Art Werkzeugen) und quantitative Analyse eines Software-Systems durchgeführt. Die Anforderungen dafür berücksichtigen Aspekte die notwendig sind zur zeitkritischen und sicheren Übertragung von sensiblen biometrischen Messdaten. Diese wurden auf Basis von in wissenschaftlicher Literatur vorgegebenen Referenzwerten überprüft. Für die Latenz wurde ein Zielwert von weniger als durchschnittlich 300 ms festgelegt. Die Sicherheit der Implementierung und die dafür notwendige Zusammenstellung der Software-Komponenten darf keine kritischen Sicherheitslücken aufweisen.

Das resultierende System bewältigt die Übertragung von einem Edge-Device bis hin zur Live-Anzeige in einem Webbrowser in durchschnittlich 207,57 ms bei einer WLAN-Verbindung mit einer Standardabweichung von 80,95 ms respektive 262,19 ms bei einer mobilen Datenverbindung mit einer Standardabweichung von 110,18 ms. Im Hinblick auf Sicherheit wurden alle, durch statische Analyse gefundenen, kritischen Sicherheitslücken beseitigt. Die Ergebnisse zeigen, dass die Anwendungsfälle unter diesen Bedingungen möglich sind. Abschließend werden weiterführende Einsatzmöglichkeiten und Verbesserungspotenziale vorgestellt.

Abstract

Evaluating sports data in a timely and secure manner is challenging and technically complex, even more so in a spatially distributed system. Recent advancements in both science and technology enable efficient and precise body tracking which lead to a surge in adoption across various domains ranging from health to entertainment. Such developments also create opportunities in the martial arts domain where computer-based systems for expert support are still a novel concept with few established technological standards. This includes support for athletes, coaches and people involved in sports events such as referees to facilitate quick insight.

Multiple sets of prototypes of novel IoT-boxing-gloves have been provided for use as data sources within this work. These generate continuous real-time data in a spatially distributed fashion. This data has to be collected, evaluated and presented. Any delay in evaluating such data would lead to degradation of the quality of the results which is crucial to use cases such as competitions, live transmissions and medical implications. Given the circumstances, the research question about the feasibility of a system which enables time-critical and secure transmission, processing and live presentation of data in the martial arts domain emerges.

To answer these questions, a methodological process consisting of domain expert interviews, modeling, a prototypical implementation, laboratory experiments, measurements (using state-of-the-art tools) and quantitative analysis of a software system has been conducted. The requirements for this include considerations of an adequate level of technical performance, less than 300 ms on average, and security of the implementation, requiring zero critical vulnerabilities, as well as secure transmission of the biomedical and personal data.

Hence, a state-of-the-art software system is presented which is capable of transmitting data from an edge device all the way to the live presentation in a web browser in an average time of 207.57 ms with a standard deviation of 80.95 ms using a Wi-Fi connection or 262.19 ms with a standard deviation of 110.18 ms using a cellular connection. In regards to security, all critical vulnerabilities found through static analysis have been mitigated. The presented results prove the feasibility of the use cases. Furthermore, prospects on future directions are provided including the exploration of additional possible use cases as well as areas of potential improvements.

Contents

CHAPTER 1

Introduction

Recent advances in both science and technology become increasingly relevant in sports. This helps with the support of athletes as well as event organization such as competitions [LKH+02]. Popular examples of current research and development are already advanced in sports such as tennis [PYHZ14] and soccer [Boj14] where practical applications have already been used for years, even at large scale events like the soccer World Cup 2014. Another example is Peloton's online spinning classes which enable multiple participants to workout together and compete with each other remotely [Pel].

In contrast, such integration and utilization of technology in training and competition settings is still a novel concept in the martial arts domain. The term "martial arts" refers to and overlaps with many sport disciplines, particularly combat sports such as boxing, kickboxing and mixed-martial arts (MMA), which can be practiced competitively or for self-defense [Wet15]. In this domain, the knowledge and experience on practical applications of smart equipment and accompanying systems are more limited compared to other domains. Current works are typically centered around basic movement measurement, predominantly using inertial measurement units (IMUs) encompassing accelerometers and gyroscopes, without proper punch detection and little to no domain-specific analysis [WEST19].

To aid with this lack of technical applications in the martial arts domain, a comprehensive software system is explored and designed for training and recreation alike. A set of novel, sensor-equipped and wirelessly connected boxing glove prototypes has been provided for the purposes of this work. These Smart Gloves [1] use a patented [ZHF+18] approach to quantify physical metrics of punches not only through an embedded inertial sensor but also by utilizing force measurement units. These prototypes have been used to capture and subsequently evaluate data throughout, acting as sensors in the context of Internet-of-Things (IoT). Basic functionality such as the acquisition and display of basic

[1] The term "Smart Gloves" is used as a working title.

sensor data have already existed but are not sufficient for state-of-the-art scenarios and use cases that require aggregation and processing in a distributed system.

Within this work, a software system is explored by analysing the requirements and deriving expert/user-centered use cases desired from such a system. Designs around those connected boxing gloves which enable multiple users to interact with each other live are evaluated. A prototype of such a system will be implemented and evaluated. This concerns practices and approaches from several technical fields, particularly Internet-of-Things (IoT), cloud computing, distributed systems in general and web engineering.

A wide variety of use cases is possible for such a system and could potentially be expanded upon. For example, certain use cases could gravitate towards a mix of online spinning classes and live workouts. A simplified overview of a concept is presented in Figure 1.1 which shows multiple participants using the Smart Gloves to interact with an underlying software system. This could be extended to competitions, live broadcasts or transmissions. Such features could also provide feedback and insight for users, which can be athletes and coaches alike. For example, a number of participants could be boxing on site or at home, and see their peers' performances on a screen while an instructor could use her/his expertise to analyse the results. This could be further extended by incorporating expertise from experts from other domains, such as medical aspects, to, for example, interrupt workouts if certain thresholds are reached and could result in the participants harm.

1.1 Rationale

The scope of this work inherently carries complexity due to the wide variety of topics and domains it stretches across. One major challenge is posed by using the novel prototype Smart Gloves along with its custom software and limitations. Another challenge is integrating the prototype hardware and its software into a distributed system to eventually collect and evaluate the data emitted. Subsequently, a user has to be presented with a visual representation of the results of the processed data. It is crucial that the system provides adequate performance adherent to the state-of-the-art in software engineering and comparable systems. For this, it is necessary to compose a set of software components using software engineering practices and activities in order to provide the desired functionality while maintaining such adequate performance. To achieve and verify this, a process of evaluation is applied which involves several different approaches, a key part of which is measuring and comparing performance metrics of potential individual software components as well as the entire software architecture via lab experiments, observation/measurements as well as quantitative analyses. Moreover, since personal data will be collected, an appropriate level of security needs to be established to adhere to best practices and legal requirements.

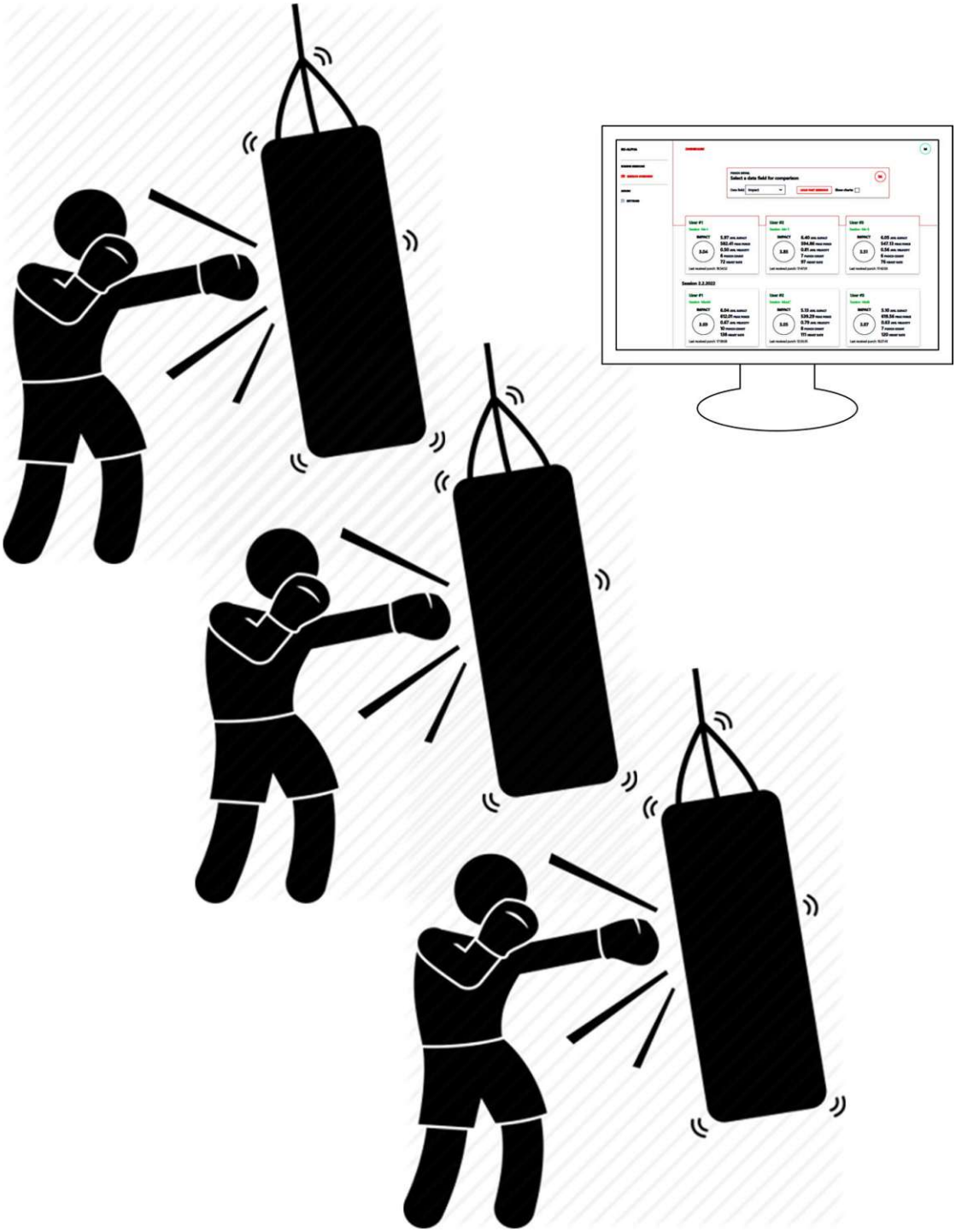


Figure 1.1: A simplified exemplary illustration of a live presentation of a boxing session and comparison of participants. This figure uses the royalty free black-and-white punching icon [Lib].

This work's contributions can be described as follows:

- Answering the question whether such a system is possible under given performance targets by providing and evaluating a reference implementation of the proposed system.
- Can a sufficient level of security be achieved for that system.
- Domain-specific live data presentation in the web.
- A system design expanding on novel boxing glove prototype technology using an unprecedented combination of sensors.
- A comparison of various software components on multiple layers/tiers.

1.2 Theoretical and Technical Foundations

As this work stretches across multiple different domains and concepts, the theoretical and technical foundations, as well as contextual and domain-specific backgrounds including a small glimpse into history, are introduced and detailed in this section. These foundations will be used throughout the work and provide background for possible system designs as well as the results and evaluations thereof, especially in Chapter 3 and Chapter 4.

1.2.1 Domain Background

To provide extended background information on the context and domain of this work, explanations and illustration of boxing as well as martial arts are detailed. The term martial arts is defined as combat based on codified systems and traditions for purposes of sports, self-defense, personal development and competitions [Spr10]. Martial arts goes back to ancient history with first competitive applications believed to be around 3000 BCE in ancient India, Babylonia and Egypt [GS10, Brib]. Over time, the sub-categories of martial arts, fist fighting and wrestling, have been included in the Greek Olympic Games from 776 BCE [Brib]. From fist-fighting on, boxing first formally appeared Olympic games in 688 BCE [Bria]. Earliest evidence of protective gear such as gloves or hand coverings to avoid injuries dates back to 1500 BCE [Bria].

Martial arts can be classified into several groups such as striking, grappling, throwing, weapons-based, low-impact/meditative-style and MMA/hybrid-sports-style [BxR]. Boxing falls into the category of striking techniques [BxR]. As previously mentioned, over time protective gear has been developed and applied to improve health and safety of the athletes. The main protective gear for (amateur) boxing, an official Olympic sport since 1904, are boxing gloves which are nowadays produced by various manufactures using pads consisting of different types of foam [CAAO18, HHK+10]. They significantly absorb and dissipate the impact force on both of the boxers' hands and protect them

from injuries to a certain extent. However, there is still a significant number of injuries reported [CAAO18].

Boxing as a sport is nowadays performed involving two participants/contestants (i.e. boxers) with padded gloves carrying out offensive (i.e. punches) and defensive (blocks, dodges, ...) actions [Bria]. For practice and training sessions, one boxer can practice punches against a punching bag [McN]. Along with techniques and equipment, also the competitive settings have evolved. During boxing matches, detected hits from punches constitute to a score, varying based on the location of the hit, to determine a contestants performance in both amateur and professional boxing [HHK+10]. This detection is error-prone as it relies on the perception of the referees which often lead to controversies [HHK+10]. Such tedious manual point detection and score keeping by referees, judges and event managers lead to the exploration of automated approaches. Efforts to support the organization and execution of boxing events and competitions have been made, for example, by creating an automated scoring system that aids referees by analyzing the scores of participants in a live fashion [HBC21].

1.2.2 Sensor Technology and Equipment

As previously outlined in the domain background, equipment and protective gear for boxing is getting more and more sophisticated up to the point of using micro-electro-mechanical sensors (MEMS) to measure movements. Such sensors for human-bio mechanical use-cases have to be wearable, i.e. integrable into clothes, gear or devices worn on the human body [FC10]. Three types of sensors are of particular interest to this work:

- Gyroscopes which measure angular orientation along one or several defined axes making use of Earth's magnetic fields [Dad14].
- Accelerometers which measure the linear acceleration along one or several defined axes in m/s^2 [Dad14].
- Force measurement sensor deriving the force from pressure changes in a sensor pad during compression, due to punches with target contact [ZHF+18].

The gyroscope and accelerometer can be integrated into an inertial measurement unit (IMU), which estimate relative orientation to an inertial frame [FC10]. As these sensors are subject to issues such as drift and distortion errors individually, they can be used to complement each other to compensate those issues. Recorded over time, the measured signals of these sensors provide time-series data that, if it is triaxial, can be used to evaluate movement through three-dimensional space [WEST19].

Sensors are also a typical part of an IoT system where they produce data that can be consumed and processed by other connected devices [SCZ+16]. The connection to other devices usually happens via a wireless Bluetooth or Wi-Fi connection which makes them part of a computer network or distributed system [SCZ+16].

The Smart Gloves subject to this work make use of such sensors, in particular an IMU integrating a triaxial gyroscope and triaxial accelerometer as well as an unprecedented force measurement sensor, to measure physical parameters and convert them to electrical signals and, in turn, to digital time-series data. Based on which performance features describing the quality of the punches can be calculated and extracted [WEST19]. They also act as IoT sensors in this setting.

At the time of writing the provided gloves are in a near-final prototyping stage [ZHF+18]. They have been developed by a group of researches and stakeholders. The designs are not completely final and still subject to change. An illustration of a single prototype glove can be seen in Figure 1.2. The Smart Glove prototypes contain an integrated system including a rechargeable battery, a Bluetooth [SIG] modem, an IMU and a force sensor pad [ZHF+18].

Figure 1.2: An example photograph of a single prototype boxing glove provided for this thesis using patented technology [ZHF+18].

The units of measurements are meters-per-second (m/s) for the velocity/speed, meters-per-second-squared (m/s^2) for the acceleration and newton (N) for the force, as standardized by the International System of Units (SI) [Laba].

1.2.3 Software Engineering Concepts

Based on the novel sensor technology integrated into the Smart Glove prototypes, there are many possibilities for applications and expansions around it, such as virtual workouts, group classes for training, competitions, events as well as broadcasting or streaming. To properly utilize the hardware capabilities and make use of the sensor data, in particular the previously described IMU data as well as the force data which is unique to this work, software is necessary. To develop software, the approach of software engineering is applied [RGI75]. Therefore, concepts and fundamentals of software engineering in the fields of software architecture, requirements and activities such as testing are laid out in this section to later build upon [BCK03, CNYM12]. These fundamentals will become relevant in the chapters Results 3 and Evaluation 4.

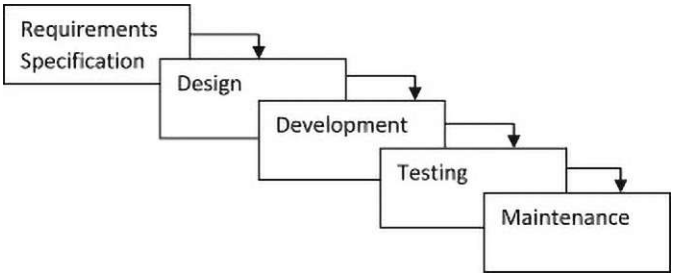


Figure 1.3: An illustration of the classic software engineering Waterfall model [GPC20].

A general overview of classic software engineering methodology and software development processes is presented in Figure 1.3 as the Waterfall model [GPC20]. It is a sequential-linear life cycle model incorporating a structured path across five stages, from requirements to maintenance, particularly suitable for insular projects where the requirements are fully established in the beginning [GPC20]. Throughout such a software development process, several artifacts are created, especially at the design stage, such as a software architecture along with its models of structure and persistence, documentation as well as prototypes including the source code [TvdH07]. The final step of this process, maintenance, is considered negligible for the scope of this work as it concerns operation and adjustments of the software after it has been delivered to a customer.

Requirements in Software Engineering

In software engineering, requirements provide a description of what functionality a system should provide, the quality of this functionality as well as the context in which it will be used [CNMR18]. These requirements are divided into functional (FRs) and non-functional requirements (NFRs) [CNYM12]. They are elicited at the beginning of the software project by inquiring them from stakeholders or domain-experts as part of requirements engineering [CNMR18]. Typically, during the process of software engineering, both functional and non-functional requirements are implemented. However, the presence of implemented requirements alone does not guarantee that they are useful or usable without the presence of NFRs such as performance, security and interoperability. FRs are considered "hard" characteristics, providing definitions of the functions which a software system has to perform. NFRs on the other hand are considered "soft" characteristics which constitute to quality and may be difficult or complex to achieve. Hence, when developing a software system both FRs and NFRs have to be taken into account.

Software Architecture

A software architecture is means to structure software and its functionality in a meaningful way [BCK03]. The definition by Bass et al. [BCK03] of architecture typical in software engineering is:

> The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them [BCK03].

The nature of this work involving data production and consumption/processing across multiple different devices, from sensors over edge-devices and servers to end-user-devices, makes architectural approaches from the domains distributed systems and IoT particularly relevant [Kam10, SCZ+16, WEST20]. The functionality necessary to fulfil requirements is typically structured and embedded within a software architecture.

Microservice Architecture A state-of-the-art software architecture approach to such systems is considered the microservice architecture, the fundamentals of which are laid out in this section [Fow, Thö15, AAE16]. A microservice software architecture is an architectural style which involves dividing a software application into independently deployable parts where each has defined functionality (i.e. services) accessible via interfaces [Fow]. A major perk of this architectural approach is heterogeneity in terms of technology choice, i.e. individual services can use different software components if they share a common interface [AAE16].

Persistence Concepts As this work deals with distributed and partitioned data, persistence concepts for data storage are also relevant. In database systems, there are three major theorems to consider: ACID (Atomicity, Consistency, Isolation and Durability), CAP (Consistency, Availability and Partition tolerance) and BASE (basically available, soft state, eventually consistent) [Pri08]. Traditional relational-database-management-systems (RDBMS) or structured query language (SQL) systems provide ACID properties for transactions which brings reliability to the data storage [Mas15]. More modern, not-only-SQL (NoSQL) systems on the other hand, provide faster propagation and more distributed performance at the expense of initial reliability [Mas15]. In this work, a combination of both approaches is explored.

Software Testing Levels and Scopes

An integral activity to validate a software implementation is software testing [Bin00]. Since the focus of this work lies on the performance and security of a software system and, by extension, the prototype thereof, the fundamentals of software testing are also explained. Software testing distinguishes tests into four levels of three different scopes [Bin00].

Unit Testing The granularity of this test is comprised of the smallest executable unit, i.e. a method of a class in object-oriented settings. In this case, the unit scope is actually of method scope.

Integration Testing An integration tests scope is a system or subsystem comprised of a set of units. This test demonstrates the collective (inter)operability among units. Therefore, integration tests are of subsystem scope.

System Testing By extension, a system test's scope is the entire system. System tests can be categorized into: functional (IO), performance (response time, resource utilization), stress/load testing.

Acceptance Testing This is a system scope test to verify the customer's criteria based on prior negotiation and an established test plan.

1.2.4 Software Performance

Performance is a NFR in software engineering which describes performance constraints [CNYM12]. These constraints are can be defined through time/space bounds (e.g. workloads), throughput, response times and/or available system storage [CNYM12]. This NFR will be particularly relevant later on in the chapters Results 3 and Evaluation 4, as it is the centerpiece of performance benchmarks and measurements.

Performance constraints concerning response times or latencies typically measure delays in milliseconds [KSG04, SS16]. Such delays can range up to several seconds, for example less than two seconds, in systems like sports event broadcasting from a live event to the final display [KSG04, CHWB11, Gue02]. In other live applications such as video conferencing and Voice-over-IP (VoIP), delays of up to 300ms are considered acceptable [SS16, WLS95].

Performance Testing

Performance testing is of paramount importance within this work and therefore the fundamentals are examined. Performance testing is an umbrella term for different types of performance tests such as load testing, stress testing, scalability testing among others [GB18].

In performance testing the principal testing types are static and dynamic testing according to International Software Testing Qualifications Board (ISTQB) [GB18]. Static testing concerns itself with the architecture and design. Its activities revolve around high-level activities such as technical reviews of performance requirements, algorithm designs, queries, system and network architecture as well as critical segments of the system. Dynamic testing on the other hand relies on the execution of different automated test implementations across all four testing levels defined in 1.2.3.

Scalability

Technical scalability is necessary to handle the demand or load on a system and provide performance and efficiency. In terms of distributed systems, there are two types of

technical scalability: horizontal and vertical [CVRM+10]. Vertical scalability is concerned with hardware resources of individual machines, i.e. components such as CPU, RAM, etc. whereas horizontal scalability focuses on the amount of machines in parallel such that they can perform tasks in parallel. Since this work is based around distributed systems, horizontal scalability is the focus. A state-of-the-art approach in software engineering to facilitate horizontal scalability is achieved through the previously described microservice software architecture 1.2.3. This can be aided by using containerized services of which multiple instances can be run in parallel while also scaling according to demand [Fow].

Software Performance Engineering (SPE)

Software Performance Engineering (SPE) covers a wide range of software engineering activities designated to ensure the NFR of software performance (in the sense of capacity and timeliness) [WFP07]. SPE concerns the architecture and design of software systems and has to be integrated into the development process of such systems [WFP07]. SPE activities include: identification of concerns (i.e. system operations and resources such as network connectivity, processes/threads and buffers), definition and analysis of requirements as well as performance testing. In general, there are two SPE approaches which are distinguished into measurement-based and model-based approaches [WFP07]. The former relies on testing and measuring and is particularly well-suited for the herein relevant architecture since end-to-end performance is a common issue in Wireless Sensor Networks (WSNs) such as in this work [DMM+13]. Additional comparative methods such as benchmarking also fall into this category by extension.

1.2.5 Software Security

Security is a very broad term in the domain of software engineering. It is a non-functional characteristic of a software system and hence an NFR. The focus shifted from protecting systems from a hardware and software perspective to protecting information within such systems which also concerns socio-technical interactions within organizations using such systems [Fen08]. Nevertheless, security aspects and considerations are subject to the underlying technology and typically embedded therein but also have to be respected in the process and interaction design, planning and execution. Even popular and widely used software frameworks, libraries and tools like Apache HTTP Server and Apache Tomcat [PSO19] as well as modern operating systems [Sur18] struggle with security threats and vulnerabilities. Ideally, known security vulnerabilities should be identified and eliminated early on and considered throughout the software design process using assistance provided by tools and security databases [QPL18].

CIA Triad

A fundamental approach for evaluating security requirements is the CIA triad, describing Confidentiality, Integrity and Availability [Fen08]. Over time, original triad has frequently been enhanced by several additional tenets, such as Authenticity (Au) and

non-Repudiation (nR) which are in a relation to integrity and classified therein. As reviewed by Spyridon Samonas and David Coss [Fen08] the original triad and its enhancements often lack awareness of socio-technical issues. This lead them to re-conceptualize the triad in a broader way such that the intersecting areas, especially of availability and confidentiality, are better represented and more adjusted to modern environments, particularly on emerging mobile technologies and privacy concerns. The enhanced triad is visualized as Venn diagram in Figure 1.4.

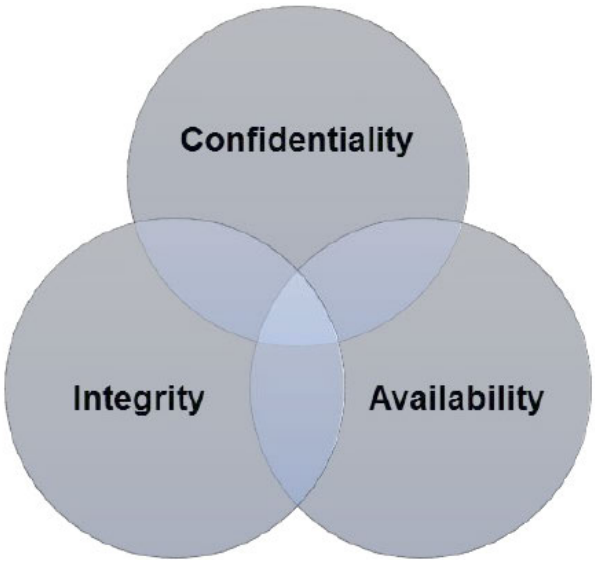


Figure 1.4: The revised CIA triad as Venn diagram by Spyridon Samonas and David Coss [Fen08].

One of the primary tenets of information security (infosec) is confidentiality which restricts certain information to specific authorizations to allow access with a major corner-stone being perceived trust. Another primary tenet is integrity, which conceptualizes the professional practices, standards and responsibilities for achieving adherence to commonly accepted principles and values. The last of the primary tenets is availability, which describes timely and reliable access to a system but also in a wider understanding entails the usability of that system.

Network Security

When data is exchanged between different devices, typically network communication is involved. According to the OSI model, network communication as relevant in this work is organized into 7-layers [Ala14]. An exemplary illustration of the OSI model with data flow as well as en- and decapsulation of data is presented in Figure 1.5.

An early fundamental layer, where security measures can be applied is the transport layer. This also requires security measures, otherwise this concern will be delegated to

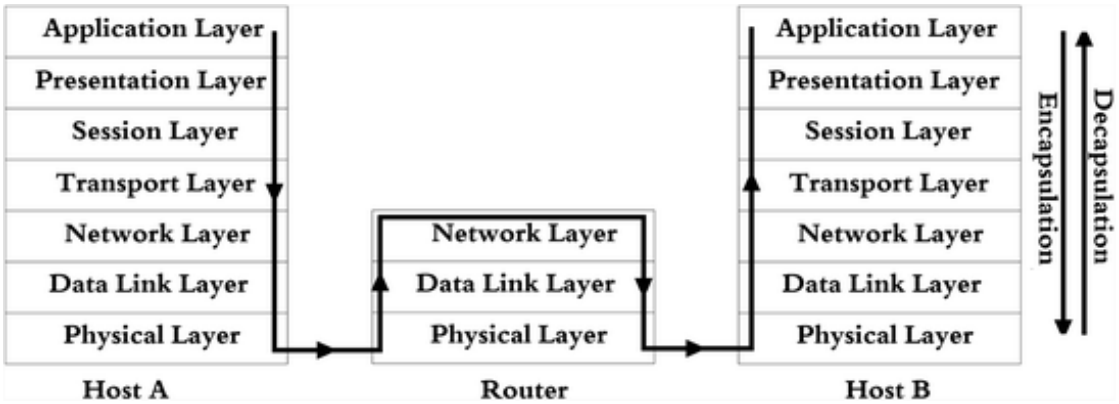


Figure 1.5: The OSI model in a typical scenario with data flow and encapsulation and decapsulation [Ala14].

other network layers which increases security risks. Security on this layer is standardized as Transport Layer Security (TLS) [DR08].

Security Testing

Since this project deals with personal data in a distributed system accessible via a web interface, the access as well as the storage and transmission need to be secured adequately. This issue is addressed at multiple layers, supported through automation during the development process. The Open Web Application Security Project (OWASP) provides a top ten of the most critical security risks for web applications [Fouc]. The top ten list of the years 2021 and 2017 compared to each other is illustrated in Figure 1.6. This serves as a reference on what to look out for when designing such a system.

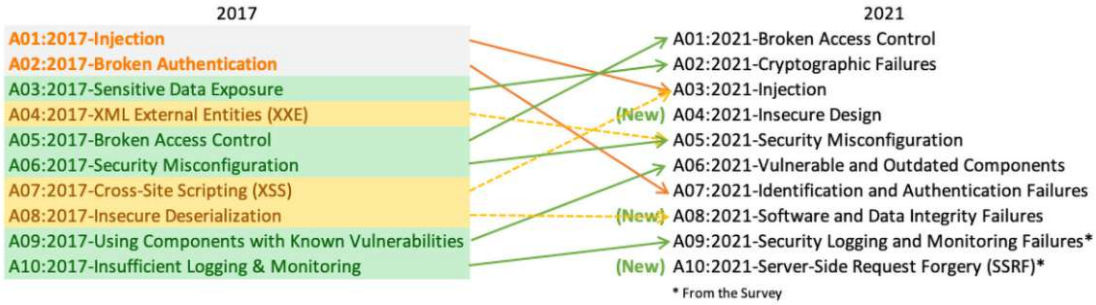


Figure 1.6: The top ten most critical web application security risks as presented by the OWASP Foundation [Fouc]. This image shows the survey results for 2021 compared against the results from 2017.

Software Composition Analysis (SCA) Software composition analysis (SCA) is an application security testing method which focuses on finding vulnerabilities in software

that stems from its dependencies [ITW21].

Static Application Security Testing (SAST) As complexity and interconnectivity of software solutions rises, so do the potential vulnerabilities and breaches. To counteract this development, tools for static analysis, especially for static application security testing (SAST) are maturing and becoming integrated into software development processes [OMGSC18]. This constitutes to white box testing, as source code and configurations have to be available and transparent [Sca21].

1.2.6 Visualization

Raw data can be overwhelming and not useful for understanding [VW05]. As this work also presents data to the user interacting with the system, visualization is used as a method to derive insight from data. It can further enable visual comparison and inference of meaning. Such visualizations can be applied using charts, which there are three basic kinds of - line, bar and pie charts [PS17]. These charts can be integrated into the user interface of a software application to assist the user in understanding large amounts of data [PS17].

1.3 State-of-the-Art

In order to build a performant software platform based on the Smart Glove prototypes by applying software engineering methods, the state-of-the-art needs to be established as well. This is done through research, particularly literature review, and environment analysis of the concerning fields such as Internet-of-Things (IoT), distributed systems, cloud computing and web development/engineering. In particular, technology and solutions regarding software architecture (i.e. software components), performance and security, as previously described in Section 1.2.3, and related current works are evaluated. Hence, building on top of the previously described fundamentals in 1.2, this chapter presents state-of-the-art technologies used in those fields and provides initial analysis and comparison as preparation for later benchmarks detailed in Chapter 3. Furthermore, it presents and evaluates related current works in the aforementioned fields.

1.3.1 Technologies

What all the previously referred to fields have in common is a varying level of distribution to at least a client and a server portion of a software application as well as means of communication between them [CN19]. According to the paradigm of separation of concerns, parts of the software architecture will be approached separately [HL95]. One part of the distribution is presentation and/or partial computation on the client's device whereas the other is operated and validated on a backend server. The backend can be further distributed into services that can be run on different machines. For the sake of simplification, the software components are approached and categorized according to the

classic 3-Tier Architecture [Kam10]. The presentation tier will be referred to as frontend, the application tier will be referred to as backend and the data tier will be referred to as persistence and transmission/communication. Additionally, in respect to modern network architectures and IoT terminology, edge devices will also be considered. Beyond that, testing and evaluation approaches and tools are discussed as well.

The herein considered options also take into account popularity or usage metrics as well as number of contributions in the respective domains and areas. The goal is to find suitable tools, components and approaches for a system design and prototype. The scope is limited to open-source technology, therefore no proprietary software components are considered.

Backend Technologies

The research of state-of-the-art technologies for the backend takes into account various aspects, from different runtime environments and compatible languages to individual frameworks and libraries. These are explored and compared with each other. There are different ways of executing server side code. These include native binaries (i.e. platform specific code), intermediary languages/code and also virtual machines/runtime environments, such as Java and its relatives [TAC+06].

Java-based technologies The Java Virtual Machine (JVM) offers an intermediary layer to execute a number of languages, including Java and Kotlin, to run on a wide range of supported operating systems on a variety of hardware [PG03]. There is an open-source reference implementation, namely OpenJDK [Cora]. As JVM-based languages such as previously mentioned Java and Kotlin are high-level languages and as such offer sophisticated layers of abstraction, they inherently facilitate means for security and code safety as well, making it easier to write secure code [Mar97]. Also, state-of-the-art static code analysis tools tend to miss fewer security vulnerabilities in Java code than in comparable C/C++ code and therefore supporting the process of writing secure software better [GPP15].

The Spring Framework and its lighter variant Spring Boot are well-known and proven industry standard frameworks for server-side software development in the Java ecosystem [Ver]. It comprises a range of libraries utilizing Java technology, thus providing a mature and statically type-safe environment with many security aspects inherent by design. The functionality built on top of Spring can be (sub)divided and orchestrated as services in a cloud.

Server-side JavaScript Engines Server-side JavaScript is increasing in popularity and using popular libraries such as Node.JS and express they can be used to create full-fledged web backend solutions [Alb]. On the one hand, two major downsides to this choice would be the single-threaded execution of code and lack of static type safety which poses risks for errors and potential security violations during development, but on the

other hand it's inherent event driven approach as well as non-blocking I/O improves performance and dynamic typing generally tends to increase productivity [Pea].

Node.JS and express are younger and less established technologies compared to the Spring ecosystem [Pea]. Node.JS and express rely on server-side JavaScript engines. It is also possible to use Node.JS to build well-performing and scalable server applications [BGDI16]. It facilitates developer convenience and fewer restrictions like a dynamic type system.

Native Native solutions, i.e. solutions which compile to a system-specific binary ahead-of-time (AOT) such as GraalVM are also an emerging option for server-side applications [WSH+19]. Performance gains of a native version of an equivalent non-native program lies mostly in the application startup [WSH+19]. Native is also the option with the lowest memory footprint compared to the other aforementioned approaches [Bou].

Ruby Ruby is a high level programming language which is interpreted, making it comparable to Python rather than native binaries [Poi]. An early but established backend framework that emerged with the Web 2.0 is Ruby on Rails (RoR) [BK07]. Just like Node.JS and Spring, it provides create-retrieve-update-delete (CRUD) functionality but with additional developer convenience features like data model based scaffolding. It also supports asynchronous JavaScript and XML (AJAX) requests and can be used to create a REpresentational State Transfer (REST) Application Programming Interface (API). Rails is a framework running on Ruby [Poi].

Frontend Technologies

For the frontend, web based technologies are considered. This way the solution can be platform independent and accessible via a common up-to-date browser. Modern web application are typically single page applications (SPA) and their performance is measured in 4 stages: loading, scripting, rendering, painting [CN19]. There is an increasing number of open-source web frontend frameworks and libraries, commonly utilizing JavaScript, with their respective popularity varying over time [Moza]. In this work, the top four frontend solutions by usage have been examined - those being Angular, React, Vue and Svelte [oJb]. An important aspect is rendering performance, which primarily comes down to the way those frameworks/libraries interact with the document-object-model (DOM). Security, as well as loading performance, is predominantly relevant at transmission and respective communication libraries as well as the underlying network. Approaches targeting security considerations can be integrated into all of the aforementioned frameworks and libraries.

JavaScript Modern browsers support the execution of JavaScript code through their respective JavaScript engines and the functionality thereof is standardized through the ECMAScript (ES) specifications [Mozb]. The minimum common version of JavaScript is ES5.1 (2012), the most recent being ES2020 [Moza]. Every year, Ecma International publishes a new specification of JavaScript functionality, and expanding it further on

an annual basis [Mozb]. This includes support for features such as Web Components, a browser native standard to organize websites to encapsulate elements as reusable components [Net]. However, the herein described frameworks and libraries make use of or at least support the underlying standard functionality such as Web Components, utilizing and expanding them by state-of-the-art means of web development, with resources and scalability in mind. Therefore, it is still viable to use frameworks and libraries to not re-implement common functionality from scratch.

In addition to the ECMAScript specifications, there are languages expanding on the ECMAScript specification and adding features not (yet) present to such extent or widely available. TypeScript is one such example developed by Microsoft which is usually compiled/transpiled to regular JavaScript (ahead-of-time) to be executed in a respective JavaScript engine [Che19]. It can be used for backend development as well if the backend is using a JavaScript environment [Pea]. The language was designed to mitigate shortcomings of JavaScript/ES5 which is known to be error prone due to dynamic typing and therefore less developer guidance. Due to the typical build steps as ahead-of-time compilation to JavaScript, minification and compression, TypeScript has no performance impact at runtime.

Angular Angular is a full-fledged frontend framework containing more than just a rendering library [Dai]. Angular is known for manipulating the real DOM rather than using a VDOM and has a larger and more complex code base compared to React and Vue (depending on configuration and bundle size) [Dai]. Another common complaint is its steeper learning curve [Wad]. Since version 9, it makes use of a new rendering engine called Ivy by default which uses an Incremental DOM to improve performance and keep a smaller memory footprint [Dul]. It also makes use of shadow DOM, the new encapsulation method brought forward through web components. Angular uses TypeScript by default.

React At the time of writing React is the most popular rendering library for the web [Dai, oJa]. React is compelling because it is efficient in terms of size and rendering performance. It achieves its performance by utilizing a virtual DOM (VDOM) instead of interacting directly with the real DOM [CN19]. Manipulating the real DOM of a browser is slower than most JavaScript operations and it is done unnecessarily often by most frameworks [Cod]. In contrast to Angular and Vue it is not a framework and also lacks features like routing and state management. There are various popular additions to mitigate those shortcomings of React, for example, Redux for state management and React Router for managing routes [Zam20]. Another important addition to React are Hooks which allow to reuse stateful logic across individual components without having to go through their hierarchies, thus further facilitating performance gains [Soua].

Vue The third considered web frontend solution is Vue.js. In contrast to React, Vue is a framework, however a much lighter one than Angular [Dai]. Vue also utilizes a virtual DOM and thus offering comparable performance to React. Specifically, with a

small amount of elements Vue performs worse while with a large number of elements Vue appears to outperform React [BP20].

Svelte The last considered solution is Svelte [Sveb]. Like Angular and Vue, Svelte is component based. It achieves high performance through AOT compilation to JavaScript. Therefore, at runtime, no virtual DOM is used and it removes the need to operate the framework at runtime which strips it of typical framework overhead [Lib22]. Therefore, Svelte is considered a compiler instead of a framework like Angular and Vue or a rendering library like React. SvelteKit is an official framework built around Svelte providing features like routing, service worker support and modern server-side rendering, bringing it into the ballpark of other frameworks such as Angular and Vue in terms of features [Soc].

Transmission and Protocols

In this section the various state-of-the-art means and methods of transmitting the data will be compared. This section also focuses on transport layer security. Furthermore, the following examined means of communication are not mutually exclusive.

Representational State Transfer (REST) State-of-the-art best practice for distributed web application is a Representational State Transfer REST API exposed on a backend server such that a frontend can communicate with it using standard Hypertext Transfer Protocol (HTTP) methods for create, read, update and delete (CRUD) calls [FSF09, Fau19]. A RESTful API typically communicates using either JavaScript Object Notation (JSON) or Extensible Markup Language (XML) data formats. Traditionally, web services used to rely on SOAP [W3C] or RPC [Groc] communication, but those are now considered too overhead-heavy or too complex, respectively, and this is where RESTful web services excel by providing a universal and standardized HTTP-based API.

Constrained Application Protocol (CoAP) In the IoT domain, there is another protocol of particular relevance - the constrained application protocol (CoAP) [BCS12]. This protocol is especially useful in scenarios with devices of constrained resources or capacity (i.e. energy, network, processing capabilities). CoAP provides a REST architectural style to make information available to wider networks such as the Internet. It uses the user datagram protocol (UDP) as underlying protocol instead of the heavier TCP [Ala14].

Lightweight Machine-to-Machine (LwM2M) Lightweight machine-to-machine (LwM2M) is a protocol for IoT device management and related service enablement [Spea]. It also provides a REST architectural style and builds on top of CoAP. In contrast to MQTT, LwM2M offers device management features such as monitoring and over-the-air (OTA) firmware updates [Tur]. The newer version 1.2 of LwM2M introduces support for additional protocols such as HTTP and even MQTT itself [Speb].

Message-Oriented Middleware (MOM) A popular communication pattern across message-oriented middleware (MOM) is publish and subscribe (pub/sub) [LYK+11]. Through that mechanism, a client can subscribe to a topic and receive messages for the subscribed topic and be automatically notified in that event. MOMs are asynchronous mechanisms, where the sender can send messages despite the recipient being unavailable at the moment [YQC+19]. This is especially suitable for cloud architectures with many heterogeneous devices such as IoT sensors and edge devices.

In the IoT domain, Message Queuing Telemetry Transport (MQTT) is a protocol which had a surge in popularity because it is lightweight and based on TCP/IP [YQC+19, PST18]. It provides pub/sub functionality on low-power devices and enables reliable real-time messaging with limited bandwidth. It supports Transport Layer Security (TLS) (further detailed in 1.2.5) and SASL for security. It is unsuitable for large payloads and persistence, however. There are popular and established message broker solutions implementing MQTT such as Eclipse Mosquitto [Lig17] and RabbitMQ [YQC+19]. In contrast to LwM2M, MQTT is focused on message delivery and transmission [Tur]. See Figure 1.7 for an illustration of a typical IoT platform using MQTT.

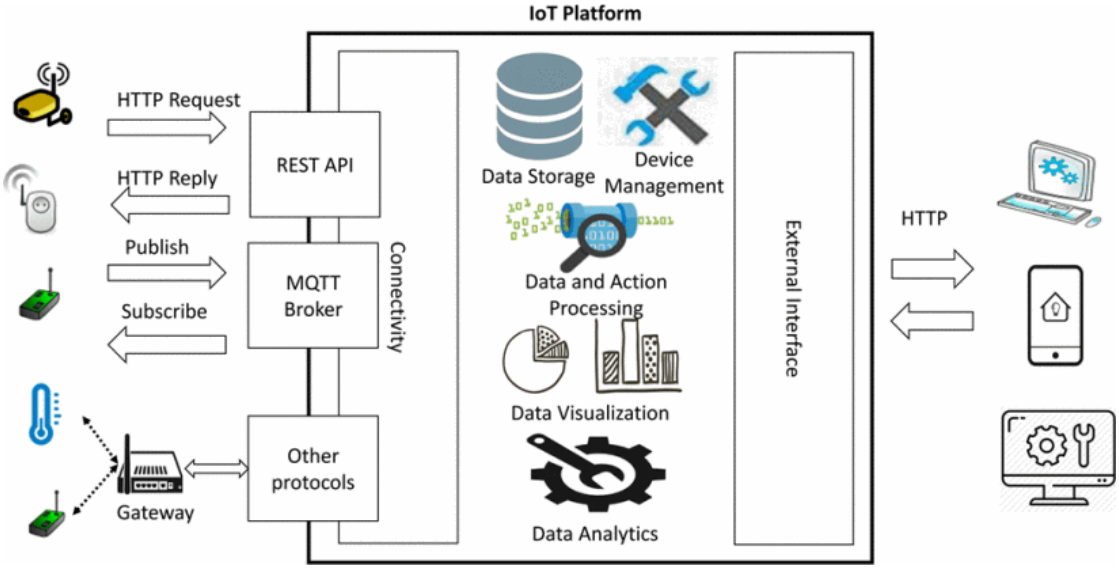


Figure 1.7: An exemplary reference IoT platform using MQTT and HTTP [IHK18].

WebSockets The WebSocket API is a W3C standard and provides fast, TCP connection based communication between web clients and servers [Kan]. WebSockets provide full-duplex/bi-directional communication but has no default communication protocol [Cho15]. A communication protocol that can be used with the WebSocket API is publish and subscribe through, for example, the Streaming Text Oriented Messaging Protocol (STOMP), making it especially suitable for realtime data streaming [WSM13].

Remote Procedure Calls (RPC) Remote Procedure Calls (RPCs) allow for a more programming-oriented call of specific functions or procedures on a remote machine [KS18]. A fast, popular, cross-platform and open-source implementation is gRPC by Google [Gooc]. gRPC can be used for CRUD via function calls and other custom function calls beyond that. Data can be passed as arguments but this data is predefined in size. gRPC also supports data streams. However, RPC is designed for a one-to-one communication model and is not designed to stream undefined amounts of data continuously [Mur]. Beyond that, gRPC requires protobuf as format for communication. Data is transmitted in binary form since it uses HTTP/2 as its underlying transfer protocol [Tan]. In contrast, WebSockets and message queues can be text based due to HTTP/1.x and typically JSON as format. According to Nathan Murthy [Mur], gRPC is ideal for communication between microservices in a cloud, but not between a web client and a web server (i,e. hypertext media).

Web Server Technologies

To provide the aforementioned backend and frontend services a suitable web server is necessary. Nginx is a popular and state-of-the-art web server with high concurrent processing capabilities and functionality such as reverse proxy and load balancing [CLNW12, LJD+14, Zam20].

Persistence Strategies

Distributed systems typically feature capabilities for persisting data in various capacities and across different nodes [CVRM+10]. These capabilities range from simplistic in-memory caches to traditional relational databases. The options are not exclusive and can be combined or used complementary.

Relational Databases Contextual data, for example, the height and weight of a user which is relevant to this work, is not subject to frequent changes and can be stored over longer periods of time. For such use cases, relational databases like PostgreSQL are considered state-of-the-art as they provide configurable security mechanisms like access control and encryption [MTSA19].

NoSQL Databases In contrast/addition to traditional relational databases, NoSQL databases offer more flexibility so it can be tailored towards specific persistence requirements, which favor performance, for example [Zam20]. For time critical data, a fast data storage is necessary, such as realtime or in-memory databases.

Firebase is a publicly available cloud service by Google which offers a wide range of features through APIs and libraries [Good]. It is a popular choice for projects such as the herein described one. The service is partially open-source and offers many open-source tools hosted on Google's servers [Soub]. It acts as a backend service, making an architecture built around that a serverless application and, within that service, it offers

an unstructured (NoSQL) real-time data base and user management/authentication [LYL+18]. A major feature of Firebase is its Realtime Database. Similar to message queues, client libraries provide listeners for specific data nodes (which can be compared to topics) and receive updates on that data. In contrast to message queues, Firebase offers more than just temporary persistence. No further realtime databases will be considered due to the project constraints as described in chapter 1.4.

Content Delivery Networks Content Delivery Networks (CDNs) provide a means to aggregate repetitive requests and reply to them using caches [VP03]. They are suitable for static content to optimize loading performance of web pages. In this case, it can be used for scripts, images and stylesheets, all of which can be compressed additionally. Cloudflare is a popular public cloud service provider with a robust CDN [LJD+14]. It can be used for static and dynamic content. Furthermore, it supports transport layer security mechanisms such as HTTPS and SSL. Cloudflare is also compatible with certificate validation for nginx [LJD+14]. An alternative to Cloudflare is Myra, which is a Germany based provider of global content delivery networks advertising General Data Protection Regulation (GDPR) [God17] compliance [Gmbc].

Edge Devices

From an IoT perspective, the smart boxing gloves merely serve as sensors, so they need to be connected to some sort of edge device to receive, (pre)process and transfer the data to a server [SCZ+16]. Any programmable and network connected (i.e. smart) device between the local sensor and a remote server can be considered an edge device. The focus in this work is on smartphones or tablets running Android, see chapter 1.4 for reference. On these devices, software is necessary to collect and process data from the sensor and send it to a server. In this case, a Bluetooth connection to the Smart Gloves to collect the data and Wi-Fi or mobile networks to connect to a remote server. The processing capabilities of the device can take off computational load from the servers. In the herein described project, that is collecting, preprocessing and aggregation of the boxing glove sensor data, see Chapter 3.4.

Security Tools

As described in Section 1.2.5, there are different approaches to verifying application security such as SAST and SCA. These can be assisted using tools.

SonarQube SonarQube, a popular SAST solution, provides an open-source platform with support for analysis of software projects for security vulnerabilities, hotspots and bugs [S.Ab]. Generally, SonarQube provides a set of metric definitions, including a security rating, ranging from A from E where A is considered the best and E the worst [Sona].

- A = 0 Vulnerabilities

- B = at least 1 Minor Vulnerability
- C = at least 1 Major Vulnerability
- D = at least 1 Critical Vulnerability
- E = at least 1 Blocker Vulnerability

Furthermore, it provides means to configure quality gates that can be configured as requirements which need to be passed during the development process to maintain a code quality standard. Additionally, there is a range of plugins and integrations available and an active community maintaining these. It also integrates Git version control and supports additional metrics from other tools such as Open Web Application Security Project (OWASP) dependency-check. SonarSource [S.Aa], the creators of SonarQube, also provide a public cloud service, SonarCloud [S.Ac], where public analysis results are available of different software solutions, including popular ones, like popular and widely used Apache software, see Figure 1.8. Unfortunately, the public SonarCloud results of the Apache projects in question do not publicly provide details on the types and severities of the vulnerabilities at the time of writing.

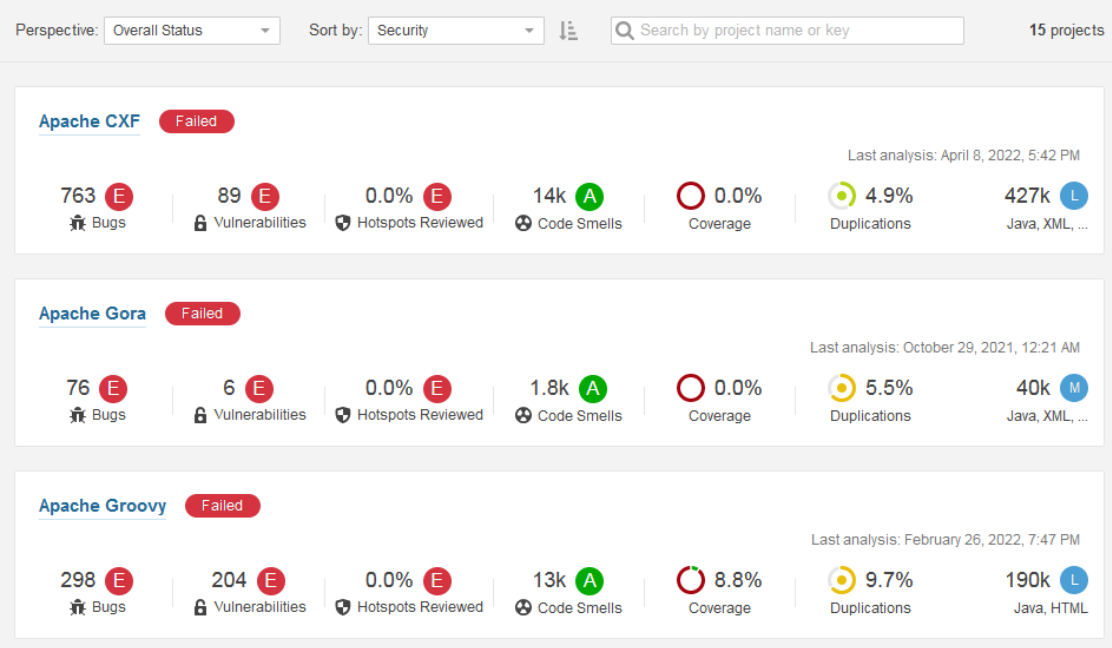


Figure 1.8: Even widely used and popular software solutions struggle with security and quality, such as tools from the Apache Software Foundation like CXF, Gora and Groovy [S.Ac, Foue].

OWASP Dependency-Check A well regarded tool to perform SCA is OWASP dependency-check [Foub, ITW21]. It evaluates software solutions with its dependencies and transitive dependencies to find Common Vulnerability and Exposure (CVE) entries. It makes use of the National Vulnerability Database (NVD) of the National Institute of Standards and Technology (NIST) [oSN]. Furthermore, it can be integrated with development and build environments such as Apache Maven [Foud] to support the development process and improve security.

Snyk Snyk is another tool with a bespoke database of security vulnerabilities and threat detection capabilities [Limb]. It acts both as a SAST as well as a SCA tool to identify vulnerabilities in dependencies as well as transitive dependencies. More than that, Snyk advertises itself as a platform which helps developers by automatically reporting newly discovered issues and provides hints on how to fix them.

npm audit Another SCA tool is npm audit centered around Node.JS package auditing [Zam20, npm]. It is particularly useful for JavaScript based software solutions and projects. It is bundled with the package manager of Node (npm).

Network Analysis Tools Modern applications are increasingly distributed and rely on network connections, which also pose potential attack surfaces. Wireshark is a tool enabling network analysis that provides capture and inspection network traffic [San17]. This enables packet inspection and verification of transport layer security to verify confidentiality and integrity of network-based communication. There are also other tools which might be integrated into a development environment such as the Network Inspector in Android Studio which can be used to analyze traffic of mobile Android apps [LLC].

Visualization Tools

There are various visualization libraries available supporting types of visualizations and charts, as laid out in 1.2.6, for the web, including D3.js, Chart.js and historically Flot [PS17].

D3.js D3 is an open-source visualization framework designed to work with a modern browser's document object model (DOM) [Bos]. It is capable of bespoke and interactive visualizations as well as data-driven transformations.

Chart.js Chart.js is an open-source state-of-the-art visualization library for the web capable of various different visualization types supporting quantitative, nominal and ordinal data values [DR19]. In contrast to D3.js, Chart.js is limited to eight types of standard charts but is customizable and has great performance in those [PS17].

Flot According to Pokorný et al. [PS17], Flot is a popular web visualization solution. However, the last update to this library was in 2014 and is not described as compatible with modern browser and will therefore not be considered [IL].

1.3.2 Current Scientific Works

This work stretches across multiple fields, including software engineering, internet-of-things, internet security, and data visualization. This section focuses on current notable works from those fields, ideally, which combine many, if not all, of those. Some notable works with similar aspects related to IoT platforms as well as sports and body tracking are examined in the following.

IoTSport

One exemplary work combining IoT and sports is IoTSport, created by Partha Pratim Ray [Ray15] describes a framework for a distributed architecture to support sports and recreational activities. This particular framework describes all components necessary for technology-supported interaction of multiple athletes. It defines the ways how users can interact with each other using the framework and lays out protocols and implementation details. It partially touches upon the same aspects as relevant in this work: security/privacy, open-source, internet-of-things, in an otherwise technology agnostic fashion. It details the kinds of sports equipment and wearable technology that can be used for it. A subset of the ideas brought forward is applicable in the herein described work, for example, if the Android phone with the corresponding app is considered as the Data Processing Layer (DPL) and the web app is considered as Visualization and Service Layer (VSL). Another similarity is the acknowledgement and distinction between several types of mobile connectivity such as cellular and Wi-Fi. There is no reference implementation provided by the research group yet, at the time of writing [Ray15]. The multi-layer architectural framework of IoTSport is presented in Figure 1.9.

Design of real-time monitoring platform for internet of things based on cloud platform

This more general work is concerned with creating a platform for monitoring data through distributed devices [WY20]. There, an IoT system is proposed using MQTT to collect real-time data from sensors and process it in a cloud environment to eventually visualize it in a web browser. On the server, a Redis Pub/Sub message queue is used. The web browser communicates with the cloud backend using HTTP and WebSockets. Using those industry standard software components, it achieves a push speed of 270 messages of average 1000 bytes length with 10 terminals connected simultaneously.

BodyCloud

The BodyCloud framework supports Body Sensor Networks (BSN) as a specialized form of Wireless Sensor Networks (WSN) with cloud integration to enable cross-disciplinary

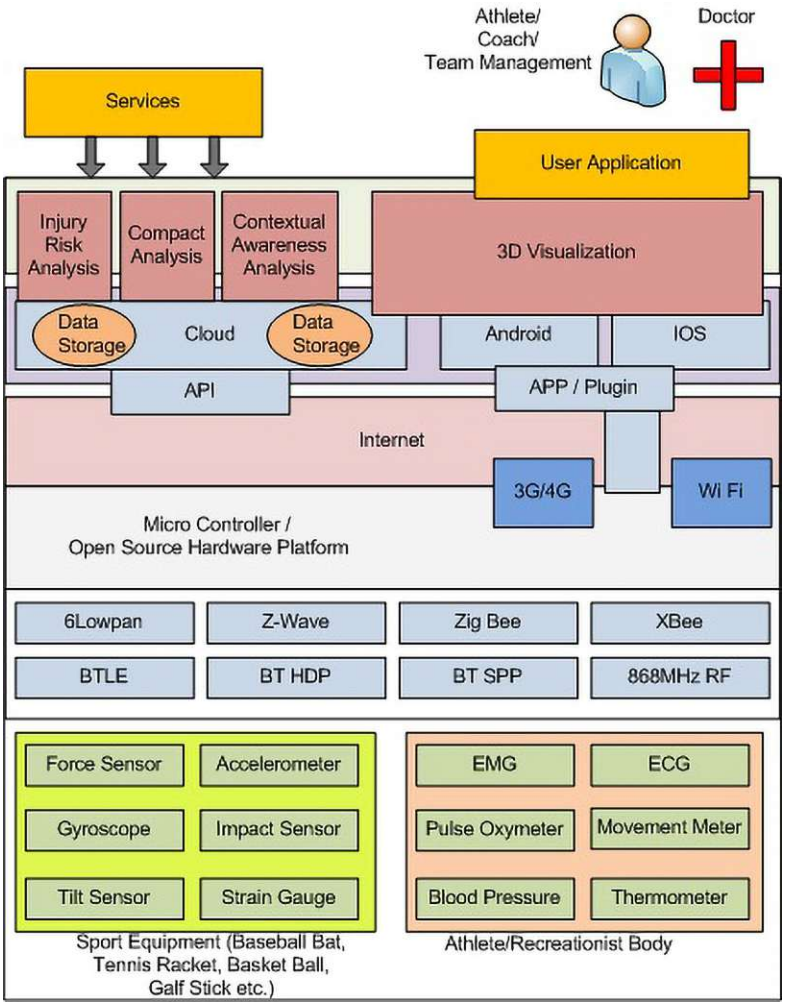


Figure 1.9: IoTSport describes a multi-layer architectural framework for different kinds of sports to be embedded into an IoTSystem using sensors [Ray15].

applications [FPD12]. Such applications are based on user interactions and collaborations. There are several application domains: health, sports and games, similar to the herein described work. Through the aforementioned BSNs sports performance, the athlete's movements in particular, can be monitored and analyzed. By extension, these detected movements can be used for games. The BodyCloud project also deals with a range of technical challenges, including secure transmissions and protection of user privacy. It uses an underlying architecture which processes data in realtime and presents results. During the course of that work, a prototype with a web interface has been developed using Google App Engine.

1.3.3 Related Commercial Systems

In addition to current scientific related works, there is a variety of commercially available or announced products on the market already tapping into parts of the herein explored concepts. An overview of related commercial work can be seen in Figure 1.10. The range includes boxing technique (also referred to as tech-boxing) workouts, home fitness, online classes, personal training (PT) software and various fitness apps and systems.

Figure 1.10: Overview of both directly and indirectly related work present on the market [Pel, Incb, Corb, dM, EW, TS, Gmbd, Nas, It, Incd, Ltdc, Jac, Gmba, Vir, gF, O¨, Tra, Inca, Gmbe, GbR, Lima, Per, Fit, MFHI, Groa, Gmbb, rG, Grob, Ltde, UA].

A selection of four individual products are discussed in the following:

FightCamp

FightCamp is a home workout solution specifically for boxing using a combination of a proprietary set of wearable sensors with boxing gloves as well as an iOS application [Incb]. The system offers workout programs and rewards users with points for their performances in real time. Beyond that, it is also geared to train the users' technique. At the time of writing, there is no real time interaction with other users, however. There is an online leaderboard where users can compete with each other for top scores in a non-real-time fashion. The user interface of the most recent version, 3.0. (at the time of writing), is presented in Figure 1.11 [Incc].

Peloton

Peloton is a popular home workout and fitness class platform utilizing proprietary hard- and software for a wide range of possible workouts. [Pel] It has a similar presentation to FightCamp but is not centered around one type of sports but multiple ones like cycling, running and yoga, for example. Peloton offers a subscription-based service via smartphone apps with features such as workout programs, fitness tracking with metrics, achievements and schedules. Despite there being cycling classes with possible multiple simultaneous

Figure 1.11: Overview of the user interface of the FightCamp app in version 3.0, only displaying a user's performance using either punch count, speed, duration or aggregates thereof as metrics [Incc].

participants, there are no live comparisons of individual performances displayed at a glance.

Move It Swift

Move It Swift is a new product comprised of motion sensor-equipped boxing gloves along with a software platform [It]. It provides features such as workout programs and virtual classes assisted through video content as suggested by the promotional material put out by the manufacturer [Asi]. Competitive use is also supported through leaderboards [Ltda]. The boxing gloves perform motion sensing with sensors placed in the wrist area but no dedicated force sensing, however, using algorithms a force is estimated [Asi]. In total, it offers three metrics to track a users progress: punch count, average speed and average force (which is estimated using algorithms) [Ltda]. At first, the project was only available through an IndieGogo campaign where it achieved its funding goals and has moved on to its own website where the product can now be ordered [It, Asi]. Also, the Make It software which makes use of the Move It Swift gloves is available only as iOS and Android app [Ltda, Ltdb].

Myzone

Myzone offers a fitness tracking platform encompassing software and a set of wearable hardware products targeted at the health and fitness market supporting individual and group workouts [Ltdd]. Myzone also supports spinning classes and a visual presentation

of multiple participants simultaneously on a single screen. There are no public details on how this single screen multi user solution is realized. There are also smartphone apps provided to make use of the Myzone software platform. In contrast to this work, Myzone focuses on wearable sensors providing data such as heart-rate measurements coming from proprietary hardware devices rather than motion or impact data and also lacks boxing specific features [Ltdd]. Figure 1.12 shows the user interface of the Myzone software displaying multiple users and their fitness metrics.

Figure 1.12: Overview of the user interface of the Myzone software, displaying fitness metrics of multiple users in a gym [Mac].

1.3.4 Summary

As already noted at the individual related works, there are several overlaps among them as well as differences to this work which will be summed up in this section. In the research-oriented related works 1.3.2, all three presented works overlap by incorporating distributed systems or IoT (which is also based on distributed systems) and data processing within that system. Although they have architectural similarity in that regard, none of those provide a system tailored for boxing, incorporating the herein available sensor data and providing performance and security implementations and evaluations thereof.

The commercial related works 1.3.3 have a focus on sports in common. All of them feature sports activities or equipment as well as software solutions around them, albeit different kinds of sports such as cycling, general fitness and boxing. The main difference between them and this work is a software system specifically designed around the provided Smart Gloves. Their unique combination of sensors and data generated through them is not present in the other works, even those with a focus on boxing such as FightCamp and Move It Swift. Furthermore, the Smart Gloves are extended by a software system through

this work, as sensors without accompanying logic/software do not provide usable features, adds boxing specific features not available in other systems such as Peloton or Myzone. For better illustration of the differences and commonalities see Table 1.1.

Feature/Related Work	This work	FightCamp	Peloton	Move It Swift	Myzone
Smart Glove support	Yes	No	No	No	No
Boxing specific feature set	Yes	Yes	No	Yes	No
Force measurement	Yes	No	No	No	No
Speed calculation	Yes	Yes	Yes	Yes	No
Multiple simultaneous users	Yes	No	Yes	No	Yes
Web interface	Yes	No	No	No	Yes

Table 1.1: Tabular comparison of the features of commercial related work and this work [Incb, Pel, Asi, Ltdd].

1.4 Environment and Constraints

This work has a set of constraints it needs to respect. These include compatibility to the data generated by the smart boxing gloves specifically as well as the software accompanying it. Furthermore, the data needs to be collected via predetermined edge devices and cloud infrastructure.

1.4.1 Data Characteristics

This section explains the data characteristics which will be used throughout the thesis, building on top of the sensor technology and the prototype Smart Gloves presented in 1.2.2. The Smart Glove prototypes provide a range of Bluetooth services through a Bluetooth 5 system-on-a-chip (SoC). The main service considered herein sends data about the force (deduced from the measured pressure within the pad) and the IMU which utilizes a gyroscope and accelerometer. It also sends metadata including about the data characteristics itself and the model information which will not be further discussed here. The data is organized in events. The events relevant in this thesis are punch events, starting from before a punch till after a punch has been performed by the user.

During a punch event, pressure data is transmitted every millisecond, and IMU data every 3 milliseconds. Such a Bluetooth packet contains 20 bytes of data, with 10 fields of 2 bytes each. See Table 1.2 for a representation of such a packet including respective data types.

1.4.2 Transmission

The herein used Smart Glove prototypes emit the sensor data via Bluetooth. The emitted data is collected via an Android phone using a dedicated app. There, it is stored as ASCII-encoded comma-separated-value (CSV) text file. It is then sent to a network

Field	Data Type
IMU Gyro X	int16
IMU Gyro Y	int16
IMU Gyro Z	int16
IMU Acc X	int16
IMU Acc Y	int16
IMU Acc Z	int16
Force 1	uint16
Force 2	uint16
Force 3	uint16
Counter	uint16

Table 1.2: The fields and their corresponding data types of such an event packet.

endpoint for the purposes of this thesis. The endpoint in question is the Firebase Realtime Database. These data packets which arrive processed at that endpoint are transferred in JSON format. Table 1.3 represents the structure of the JSON payload.

Data	Description
Equipment	MAC-address, name and left or right side information
Session	Session identifier, type, duration and user identifier
User Performance	Peak and average acceleration, force, velocity associated with a user identifier and a session identifier

Table 1.3: Data computed and transmitted to the realtime database.

1.4.3 Architectural Constraints

The provided functionality of the Smart Gloves is tied to an Android application which makes use of a Firebase Realtime Database. These constraints limit the software architecture and are detailed in this section.

Android Edge Device

The Smart Gloves provide data via a Bluetooth connection. The connection is managed by an Android device acting as edge device running a custom mobile application which has been provided for the purposes of this work. This custom application handles Bluetooth device detection, pairing and subsequently reception as well as initial partial processing of the data.

Firebase Realtime Database

From that Android edge device, the data is transferred to a Firebase Realtime Database [Good] instance using the mobile application and an underlying connection which can

be either Wi-Fi or cellular based. See Figure 1.13 which illustrates the connections and data flow between the devices and endpoints respectively.

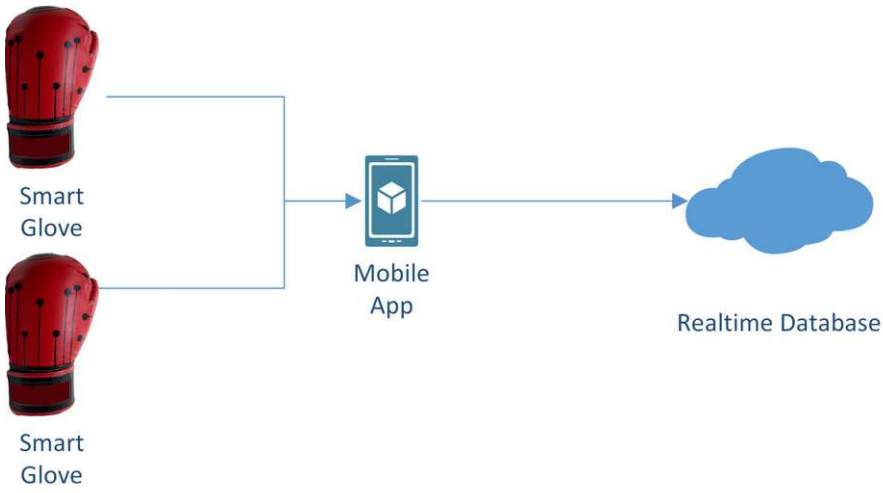


Figure 1.13: The provided architecture representing a pair of Smart Gloves and the data flow originating from these. The data flow originates in the Smart Gloves acting as IoT-sensors, going through the edge device where it is handled by a custom mobile application and later transmitted into the realtime database endpoint using an underlying internet connection which is either Wi-Fi or cellular based.

1.5 Aims

Based on the coarse conceptualization outlined in the general introduction, the finer aims of this work are established. These include the analysis of requirements, design of a state-of-the-art software system utilizing the Smart Glove prototypes that can provide user-centered functionality, a prototypical implementation of that design and the evaluation of the prototypical implementation. The implementation should provide desired functionality with multiple Smart Glove prototypes acting as distributed data sources/sensors while reaching performance and security goals as time critical transmission of contextual data is performed. That concerns approaches, practices and open-source software components to transmit, process and present the live data in a web user interface. The different individual state-of-the-art software components need to be evaluated to satisfy defined performance and security targets and consequently composed together into a coherent stack. The entire assembled stack needs to be tested and evaluated. By the course of the evaluation, scientific hypotheses are to be proven and research questions answered. Ultimately, the result should be a proof of feasibility of such a system.

1.6 Research Questions and Hypotheses

Derived from the literature review, theoretical and technical foundations as well as the state-of-the-art, this work concerns itself with the following research questions.

RQ1: What are the functional requirements for a software system designed around the Smart Gloves?

RQ2: Is it feasible to design a software system based on the Smart Gloves acting as IoT-sensors using state-of-the-art software components in order to visualize the performance of multiple participants live?

RQ3: Is it feasible to achieve the defined performance targets of a latency less than 300 milliseconds on average from edge device to web frontend using the proposed system and its reference prototype implementation?

RQ4: How can the system performance be optimized and can a significant difference be achieved?

RQ5: Can the given security target of zero critical vulnerabilities detected using four established tools be achieved for that system?

RQ6: How can the level of security be enhanced?

Subsequently, based on the research questions, the state-of-the-art and theoretical and technical foundations, two hypotheses are formulated as the main subject of this thesis.

The first hypothesis is concerned with performance. Is it possible to create a system fast enough to transmit, process and visualize the data emitted by the Smart Gloves. As previously found and laid out in the foundations 1.2.4, acceptable delays range from 300 milliseconds to several seconds. Therefore, this work will use the lower bound, i.e. a mean latency of lower than 300 milliseconds from the edge-device to the user frontend as a target and scientific hypothesis.

H1: With a selected set of state-of-the-art tools and performance measurement data samples collected during multiple experiments, a system design can be implemented which achieves an average latency of less than 300 milliseconds from edge device to web frontend.

The second hypothesis is concerned with whether it is possible to reach adequate levels of security. This is done by CVE and static code analysis in regards to security ratings. The prototype will be tested against four state-of-the-art tools. As described during the fundamentals in- SonarQube, OWASP Dependency-Check, Snyk and npm audit - to determine the number of found CVEs and shortcomings in the code. Based on the security rating definitions as described in 1.2.5 and 1.3.1, security vulnerabilities are

ideally eliminated, i.e. brought down to a number of zero. Therefore, the hypothesis for the security aspect is zero detected known security vulnerabilities that have critical severity within the prototype in each of those tools.

H2: With a selected set of state-of-the-art tools and security analysis metrics collected during evaluation against these tools, a system design can be implemented which contains no critical security vulnerabilities.

CHAPTER 2

Methods

This chapter details the scientific methods used for answering the research questions and proving the hypotheses as well as methods of requirement extraction. The individual methods are structured and laid out in chronological order. The herein examined and applied scientific methods are based on a set of best practice approaches typical for the software engineering domain [WH06]. The methodological process is illustrated in Figure 2.1. The applied scientific methods include:

- Interviews
- Modeling
- Development and test of a prototype
- Laboratory experiments
- Observations and measurements
- Quantitative analysis based on reference values found through scientific literature review

2.1 Interviews

The initial steps are comprised of research, literature review and domain expert interviews to analyze and establish the requirements for the system design. In terms of software engineering, both functional (FRs) and non-functional requirements (NFRs) are distinguished as described in 1.2.3.

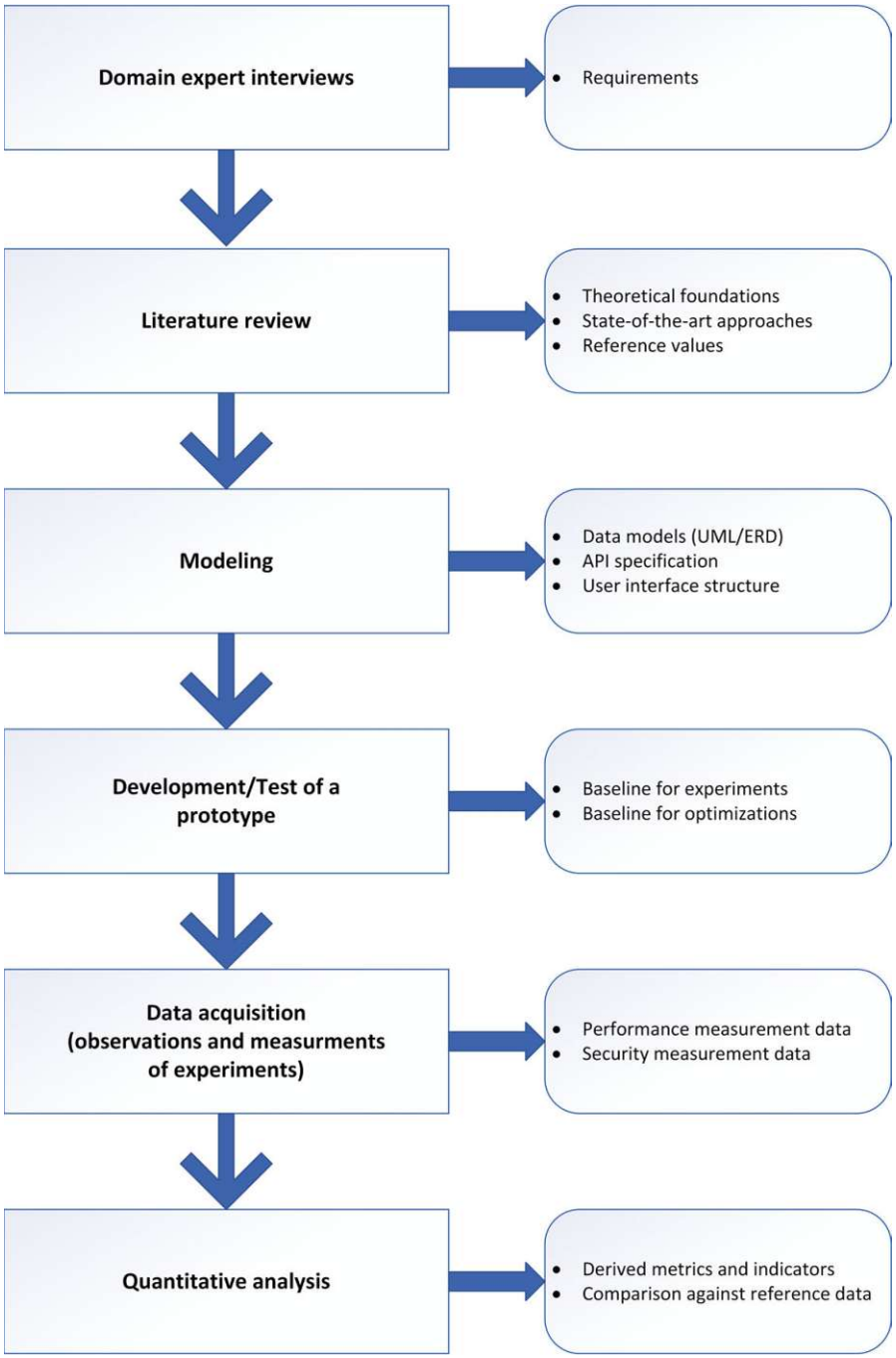


Figure 2.1: The methodological process illustrated with the steps on the left and the respective results on the right.

The interview questions are designed as open questions to allow for argumentation and interpretation of the answers, as this is part of quality criteria suggested by Mayring

Identifier	Interview Question
IQ1	What are the experiences, if any, with computer systems in Martial Arts so far?
IQ2	What features would you expect from a computer system involving the Smart Glove prototypes?
IQ3	Are there any existing systems known or in use?
IQ4	If there are known existing systems, what are their limitations?

Table 2.1: Tabular enumeration of the interview questions.

[May16]. The interview questions are listed in Table 2.1.

The guideline-based interviews with domain experts are to be qualitatively evaluated as described by Mayring et al. [Bor06] to establish the FRs relevant for such a system. After the FRs are established, NFRs are extracted and evaluated by analyzing state-of-the-art related works and industry standards.

2.2 Modeling

From the requirements found through the interviews, state-of-the-art evaluation and environment analysis, a software system design has to be created. This part of the methodological process is accomplished through modeling. In particular, diagrams such as unified modeling language (UML) [Bin00] diagrams, entity relationship diagrams (ERD) [CTKT13] and OpenAPI [KCS20] specifications are used to model the system. Additionally, the network architecture is modelled as well.

2.3 Development and Test of a Prototype

Before a software prototype can be developed according to the previously described models, a comparative assessment of state-of-the-art and open-source software technologies is conducted using both primary and secondary statistical analysis and scientific findings. These are based on metrics such as performance and security measurements as well as surveys regarding popularity and number of contributions. Subsequently, a selection of the best options for technologies and approaches has to be performed. This is part of developing a prototype is crucial as it constrains the achievable performance. Therefore, the selection process includes a series of benchmarks and comparisons of these technologies which involves practices from static performance testing as described in 1.2.4.

The selection is then composed into a software stack utilizing certain architectural styles and approaches like the microservice architecture as described in the fundamentals in 1.2.3. After the components have been picked in the previous step, they need to be assembled into a coherent design which adheres to the requirements established in this work. For this, compatibility of the software components with each other needs to be

assured. In this part, the functionality as well as data flow and persistence concerns are assigned to components as previously modelled. This design is then implemented into an actual prototype that can be run.

2.4 Laboratory Experiments

The actual test environment and amount of provided equipment is quantitatively, geographically and temporarily limited. There is only limited amount of testing equipment, computing resources as well as time and space available for the course of this work. Therefore, laboratory experiments constituting to four performance test runs consisting of 100 punches each are to be conducted to acquire representative real-world data. Two test runs of a initial feature-complete build of the prototype and two of an optimized build. The test runs are further divided into the underlying network connection, Wi-Fi and cellular.

2.5 Observations and Measurements

During the four laboratory experiments, observations and measurements are conducted and documented. These include performance measurements which lead to a total of four independent data samples which enable comparison with each other. The first two data sets collected serve as baseline for optimizations. The performance optimizations are performed using the measurement based approach in the sense of Software Performance Engineering (SPE) [WFP07].

In addition to performance, security of the prototypical implementation of the functionality is also a core concern of this work. Based on the fundamentals described in 1.2.5, this process is supported by state-of-the-art approaches and tools. The prototype will be tested against four state-of-the-art tools which provide security measurements as described in 1.3.1. These tools are SonarQube, OWASP Dependency-Check, Snyk and npm audit. The output of the tools includes measurements/observations on CVEs and shortcomings in the code as well as recommendations on how to improve them which form the basis for security optimizations.

After both performance and security optimizations have been performed, another two data sets are collected and evaluated.

2.6 Quantitative Analysis

Lastly, a thorough quantitative analysis and evaluation of the data gathered through the observations and measurements is performed to prove the hypotheses. The evaluations follows quantitative methodology using statistical measurements and tests which include mean, median, standard deviation, variance, range, significance tests [Liu11]. Using an unpaired t-test as a parametric statistic, a significant statistical difference can be

determined between the samples [Kim15]. In such cases with the assumption of unequal variances, the Welch variant of the t-test is considered more reliable [ZZ93]. To establish statistical difference in question, a null hypothesis is formed suggesting that there is no difference between the means of the two different samples. This hypothesis can be rejected with a probability value (p) and a significance level alpha (α) to conclude whether the differences in the results are statistically significant or not. The reference formulas used in the unpaired t-test assuming unequal variances are denoted in the equations 2.1, 2.2 and 2.3.

n_{s_1} and n_{s_2} represent the sample sizes of the two samples, respectively.

μ_{s_1} and μ_{s_2} represent the mean values of the two samples, respectively.

Null hypothesis:

$$H_0 : \mu_{s_1} - \mu_{s_2} = 0 \tag{2.1}$$

$S^2_{s_1}$ and $S^2_{s_2}$ represent the variances of the two samples, respectively.

df represents the degrees of freedom (df):

$$df = \frac{[\frac{S^2_{s_1}}{n_{s_1}} + \frac{S^2_{s_2}}{n_{s_2}}]^2}{\frac{(S^2_{s_1}/n_{s_1})^2}{n_{s_1}-1} + \frac{(S^2_{s_2}/n_{s_2})^2}{n_{s_2}-1}} \tag{2.2}$$

$$t_{df} = \frac{\mu_{s_1} - \mu_{s_2}}{\sqrt{\frac{S^2_{s_1}}{n_{s_1}} + \frac{S^2_{s_2}}{n_{s_2}}}} \tag{2.3}$$

CHAPTER 3

Results

This chapter details the findings and results of the application of the methods previously described in Chapter 2 and of the necessary activities involved. This is crucial to answering the research questions and prove the hypotheses outlined in 1.6. Furthermore, this chapter provides the baseline for further evaluations in Chapter 4.

3.1 Domain Expert Interviews

To find the requirements and specifications of a system like this, as well as to pave the way for answering research question RQ1 1.6, domain experts have been interviewed using a guideline-based interview as described in 2.1.

In total, three experts from the martial arts and boxing domain have been interviewed. Prior to the interview, it has been established that they are in fact involved in both events and competitions, as well as coaching and have a personal history with training and participating in martial arts events themselves. They have also been told about the Smart Glove prototypes and the sensorical capabilities, i.e. movement analysis and punch measurement.

The domain experts have then been asked a series of open questions which are defined in 2. The answers have been evaluated in a summative manner using qualitative analysis as described by Mayring et al. [Bor06]. When conducting the interviews, it has been found that the domain experts put an emphasis and agree on a particular set of arguments. During the interviews, some answers were given in anticipation of questions not yet asked. The answers of the interviewees have been inductively categorized, analyzed, summed up and abstractly paraphrased as follows.

The findings regarding the first interview question, IQ1 - “What are the experiences, if any, with computer systems in Martial Arts so far?”, are reduced to the following:

- There is little to no tool support using actual equipment at events/training sites.
- Computer systems are used at competitions but only with manual inputs for scores/performances.

In regards to the second interview question, IQ2 - “What features would you expect from a computer system involving the Smart Glove prototypes?”, the following inputs have been compiled:

- Ability to compare multiple athletes/contestants against each other.
- Live display of the performances of contestants/participants.
- Measurements or metrics in respect to the punches and quality thereof - visualized as text or charts.

The answers to the third interview question, IQ3 - “Are there any existing systems known or in use?”, come down to:

- Only computer systems that support manual input of scores at competitions.
- Occasionally, video footage is used in both training and competitive settings for analysis but not supported by automation.

The last question, IQ4 - “If there are known existing systems, what are their limitations?”, is reflected by:

- Lack of martial arts/boxing specific features in existing systems for software assisted training.

Many of the findings went beyond the scope of this work. In addition to the the findings of the first, third and fourth question, one of the domain experts expressed the desire of scoring support or point detection support for referees to improve their accuracy. Another domain expert was interested whether the performance metrics/graphs could be embedded into video broadcasts, either live or recorded. These findings are considered potential future work.

3.2 Requirements

As explained in 1.2.3, a software system is created by implementing both functional and non-functional requirements in the process of software engineering.

3.2.1 Functional Requirements

The FRs are derived from guideline-based interviews with domain experts, as presented in Section 3.1. Based on the findings of the qualitative content analysis of the aforementioned interviews, a system utilizing the specific smart boxing glove prototypes is required to do provide the following functionality:

- The system must display punch data sets (live as well as historic) using the available metrics
  - Most recent value of a given selected metric (impact, mean/peak velocity, peak acceleration, peak force)
  - Average impact of a user in a session
  - Average peak velocity
  - Maximum peak force
  - Number of punches
- The system must allow users to log in into their account using username and password

This list is also the answer to the first research question RQ1 defined in 1.6 which has been inferred from the interview and qualitative content analysis thereof in 3.1. In addition to the functional requirements found through the structured qualitative analysis of the domain expert interviews, there are two additional functional requirements derived from the environmental constraints as well as literature review of the state-of-the-art. These two are the following:

- The system must perform differential transmission of punch data between the realtime database (Firebase), edge devices (Android) and backend/cloud services
- The system must perform transmission of punch data between the service layer (backend) and presentation layer (web)
  - Using a WebSocket interface for live data
  - Using an HTTP-based REST API for past data

The differentiation between the two protocols/connection types, HTTP and WebSocket, is done to avoid long-polling behavior typical for HTTP communication and leverage a low latency connection and modern browser support as described by Chopra [Cho15]. The usage of a defined interface type constitutes a functional requirement.

3.2.2 Performance Requirements

In this section, the NFR of performance is analyzed. Building on the foundation laid out in 1.2.4, several reference systems from different domains and their respective constraints/recommendations are evaluated.

The duration of end-to-end transmission (starting from the edge device collecting the sensor data) including processing time on the individual node is compared against reference systems including live broadcasting of sports events and video conferencing solutions. A crucial factor here is that the data is coming from multiple different sources simultaneously and needs to be aligned as closely as possible to increase accuracy of live comparisons and avoid distortions. Also, the communication between nodes happens over different connection types, such as wireless to and from the edge device, involving either Wi-Fi or cellular connections such as 4G while backend/server connections are wired.

In sports event broadcasting, typical latency from the live event to the final display is in a range of several seconds [KSG04]. Sports event broadcasters like ESPN allow for delays of up to two seconds to process additional graphical effects into live broadcasts [Gue02]. According to Cavallero et al. [CHWB11] the latency requirements for embedding graphical effects live network broadcasting are less than a second. Video conferencing is an even more suiting application domain with similar constraints due to the distributed data sources. In this application domain, the data sources concurrently send data to their peers either via a central server or broadcast it directly [WLS95].

For internet based applications, Suznjevic et al. [SS16] offer different recommendations of one-way-delays (OWD) in what they consider "real-time network services" based on the type of application, specifically Voice over IP (VoIP), instant messaging, different kinds of online video games and remote desktop services. There is no category matching the broadcasting/streaming application type used in this work, however, an OWD of up to 400ms is considered acceptable for several real-time network services such as VoIP [SS16]. Unfortunately, Suznjevic et al. [SS16] do not differentiate based on connection type such as wireless or wired, and only take into account network communication between two hosts/devices, thus limiting the comparability to this work as a combination of wireless and wired communication across more than two hosts/devices is used herein.

Since any network delay degrades the Quality of Experience (QoE) of network services and can lead to loss of information (e.g. context), misinterpretation or render the information completely incomprehensible, a delay of 300ms is considered acceptable [WLS95, SS16]. For the sake of summary and structured comparison, the found reference values are presented in Table 3.1.

3.2.3 Security Requirements

As explained in 1.2.5 and 1.3.1, there are metrics and categorizations for security levels. According to Qian et al. [QPL18], known security vulnerabilities should be eliminated early on and considered in the design process using tools and security databases. Hence,

Source	Reference value
Kalman et al. [KSG04]	up to "several seconds"
Gueziec [Gue02]	< 2 s
Cavallero et al. [CHWB11]	< 1 s
Suznjevic et al. [SS16]	< 400 ms
Willebeek-LeMair et al. [WLS95]	< 300 ms

Table 3.1: Tabular representation and comparison of latencies/performance constraints according to different sources in live systems in descending order [KSG04, Gue02, CHWB11, SS16, WLS95].

the goal of zero known critical security vulnerabilities detected through static analysis (SAST and SCA) is set as requirement. These map differently to the individual tools, for reference, the A to E classification as used by SonarQube will be applied from which A will be used as a requirement there. In Snyk, simply zero detected known CVEs will be used, as will be for npm audit and OWASP dependency-check.

Furthermore, adequate security for handling biometric and personal data, also in respect to legal requirements (prospectively the GDPR, for example), is required. Even though the proposed system design does not involve third party system integration, the amount of user data to be transferred is desired to be kept as low as necessary to achieve the use cases. To prevent unintended recipients to be able to access data used by the system, transport-layer security between the network devices is required. Beyond that, security measures are important to prevent tampering/manipulation at possible applications/use cases such as in event settings, competitions and championships.

3.3 Use Cases

For further conceptualization, a formalization of use cases is performed based on the requirements and, in turn, domain experts as well as state-of-the-art, as described in 3.2. The herein proposed software system has various use cases. These are as follows:

- View session performances
  - Live stream data
  - Load historic data
- View chart visualizations (of live and historic data)
- Comparison of different participants using different metrics such as
  - Impact
  - Mean velocity
  - Peak acceleration

- Peak force
- Peak velocity

A use case diagram (UCD) of a combination of multiple use cases is modelled as described in the methods 2.2 and shown in Figure 3.1. These use cases include multiple participants using a pair of Smart Gloves each and a visual display of the current performances on a screen. There can be multiple site users and multiple participants. A site user can also be a participant but they can also be different persons. The login is represented in the UCD as it is required for the site user's use cases but is not explicitly treated as a use case.

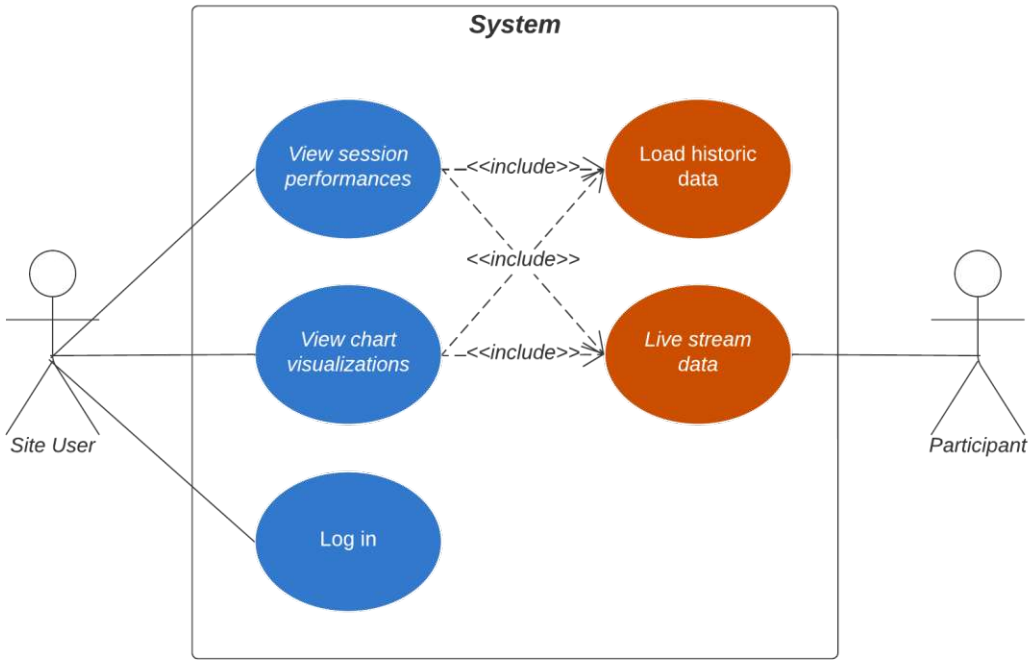


Figure 3.1: A use case diagram with the two actors site user and participant as well as the four use cases including a representation of the login.

3.4 System Specification

This section specifies the software system in regards to the requirements, constraints and environment in order to provide the functionality needed to enable the use cases. Since multiple options and choices for the selection of software components were present, it had to be decided on a selection. The decision process is detailed in the following.

3.4.1 Software Component Comparison and Composition

As outlined in the methods in 2.3, a software stack needs to be composed. For this, the most suitable software components need to be chosen from the previously established state-of-the-art 1.3. Since a wide variety of options has been laid out, a selection of software components has to be decided upon. To narrow down the choice a series of benchmarks and comparisons is performed which also factor in static performance testing considerations in regards to software architecture as described in 1.2.4. The benchmark results are evaluated and detailed in the following. The environment and constraints described in 1.4 as well as the previously outlined performance and security requirements lead to architectural constraints which are taken into account herein. Compatibility to the Firebase cloud services along with its SDK is therefore necessary. The benchmark process is split into frontend and backend technologies, as these layers communicate over standardized interfaces with each other can can therefore be cherry-picked independently.

Equipment

To select the software components, a series of benchmarks is performed. For these benchmarks, as well as prototype evaluation using lab experiments later on, a set of equipment has been arranged and defined to create a baseline for benchmarks and comparisons. Therefore, the hardware and software environment used throughout this work is detailed in the following list:

- Lenovo ThinkPad P53 with an Intel Core i9 9880H and 32GB of DDR4 memory as web frontend machine and test server
- Windows 10 x86_64 as operating system
- Node.JS in version 12.18.4 along with npm 6.14.6
- Google Chrome as web browser
- Java (AdoptOpenJDK) 11 LTS
- Xiaomi POCO F3 6GB with Android 11 as edge device
- Firebase Spark instance
- Linux VM instance (Ubuntu 20.04 LTS, 1 x 2.0 GHz Intel Haswell, 2GB RAM)

In addition, a punching bag has been prepared and (re-)used throughout the laboratory experiments.

Frontend Benchmark

Before the user interface can be built, the most suitable frontend software components need to be selected. To put the various state-of-the-art frontend technologies found in 1.3.1 to the test and help with the selection process, a benchmark has been performed. For this, the "js-framework-benchmark" by Stefan Krause [Kra] has been used. The benchmark supports a wide range of web frameworks and libraries in various versions. It conducts measurements of duration for given tasks, startup metrics and memory allocation. In this work, the benchmark has been performed for the aforementioned frameworks and libraries and their respective most recent versions. Despite tool assistance, this was a time-consuming process as all of the frameworks and libraries in question had to be assembled in the desired versions and configured for the benchmark. The outcome and data gathered through conducting this benchmark serves as a primary statistical data source for the selection process of the frontend technology options.

The pure JavaScript functionality provided by modern browsers is referred to as Vanilla JavaScript or Vanilla JS. The execution has been performed in the most widely used browser in its most recent stable release at the time of of the benchmark, Google Chrome in version 88 [Goob, Kin]. The versions of the frameworks, libraries, compilers or tools were also chosen in the most recent version at that point in time - these were Svelte at version 3.29, Vue 3.0, react 17.0 and Angular 11.0 [Svea, You, Fac, Gooa].

The following figures 3.2, 3.3 and 3.4 display the detailed frontend benchmark results. The figures contain tables where the frameworks/libraries correspond to columns and the conducted tests to rows. The results are displayed as geometric mean and sorted from left to right. They are based on keyed implementations which manage associations between data and DOM elements through keys such that changes trigger DOM changes as well. For the sake of completeness, a comparison of startup metrics is included as well as it is part of the benchmark, however it is negligible for the purposes of this work as this is a one-time cost factor which is not relevant at runtime.

The most important benchmark result is the first one - duration for given tasks, due to the focus on temporal performance. Nevertheless, for the sake of completeness, all three benchmark results have been included. Ultimately, in all three of these benchmarks, the clear overall winner is Svelte.

Backend Benchmark

There has been a benchmark of three popular backend frameworks, namely Spring Boot 2, Node.JS and Ruby on Rails, each against a MongoDB database (except for static reads), conducted by Sergei Sizov [Siz]. For this benchmark, a REST API replying with a JSON response has been created in each of the frameworks mentioned. The findings are illustrated in Table 3.2. The major release of Spring Boot used in the benchmark is 2 which is also the latest available major release [VMw]. The same goes for Java and the corresponding JVM where the latest long-term-support release is 11 at the time of

Name Duration for...	vanillajs	svelte-v3.29.4	vue-v3.0.2	react-redux-hooks-v17.0.1 + 7.2.1	react-hooks-v17.0.1	angular-v11.0.2	react-v17.0.1	react-redux-v17.0.1 + 7.2.1
Implementation notes	772							
create rows creating 1,000 rows	101.6 ± 2.4 (1.00)	125.5 ± 2.0 (1.23)	126.6 ± 2.0 (1.25)	156.4 ± 3.9 (1.54)	150.9 ± 2.6 (1.48)	162.6 ± 2.4 (1.60)	169.3 ± 1.3 (1.67)	189.7 ± 1.2 (1.87)
replace all rows updating all 1,000 rows (5 warmup runs).	101.2 ± 1.0 (1.00)	127.7 ± 0.6 (1.26)	112.3 ± 0.9 (1.11)	122.6 ± 0.8 (1.21)	124.1 ± 1.1 (1.23)	130.6 ± 1.7 (1.29)	135.5 ± 1.4 (1.34)	148.7 ± 0.6 (1.47)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	279.5 ± 23.2 (1.00)	310.0 ± 9.5 (1.11)	322.1 ± 17.3 (1.15)	386.8 ± 12.9 (1.38)	343.3 ± 7.6 (1.23)	288.8 ± 18.3 (1.03)	444.2 ± 6.9 (1.59)	520.1 ± 23.4 (1.86)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	50.5 ± 10.7 (1.00)	61.5 ± 4.1 (1.22)	283.8 ± 10.1 (5.61)	96.3 ± 5.5 (1.91)	154.8 ± 10.8 (3.06)	128.4 ± 10.8 (2.54)	241.4 ± 8.8 (4.78)	112.0 ± 5.6 (2.22)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	55.3 ± 4.3 (1.04)	53.2 ± 0.4 (1.00)	59.2 ± 2.1 (1.11)	540.8 ± 7.5 (10.16)	535.2 ± 11.9 (10.06)	576.5 ± 13.5 (10.84)	539.9 ± 7.1 (10.15)	528.1 ± 7.7 (9.93)
remove row removing one row. (5 warmup runs).	19.8 ± 0.4 (1.00)	20.7 ± 0.3 (1.04)	21.6 ± 0.2 (1.09)	22.3 ± 0.3 (1.12)	20.9 ± 0.4 (1.05)	26.7 ± 2.3 (1.34)	22.9 ± 0.6 (1.15)	33.1 ± 0.4 (1.67)
create many rows creating 10,000 rows	974.6 ± 8.3 (1.00)	1,202.6 ± 26.3 (1.23)	1,109.2 ± 10.1 (1.14)	1,502.4 ± 16.4 (1.54)	1,604.8 ± 43.9 (1.65)	1,277.1 ± 18.1 (1.31)	1,620.6 ± 55.8 (1.66)	1,776.9 ± 22.1 (1.82)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown	245.0 ± 3.6 (1.00)	305.0 ± 9.9 (1.24)	288.0 ± 5.6 (1.18)	307.8 ± 4.6 (1.26)	340.0 ± 10.9 (1.39)	336.5 ± 2.9 (1.37)	372.1 ± 13.5 (1.52)	421.5 ± 4.0 (1.72)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown	154.3 ± 5.8 (1.00)	237.4 ± 4.8 (1.54)	206.7 ± 5.4 (1.34)	232.7 ± 4.1 (1.51)	241.3 ± 8.2 (1.56)	425.1 ± 9.1 (2.75)	245.3 ± 6.7 (1.59)	269.5 ± 17.5 (1.75)
geometric mean of all factors in the table	1.00	1.20	1.39	1.76	1.85	1.94	2.10	2.16

Figure 3.2: Duration for given tasks in milliseconds ±95% confidence interval. Green colored fields indicate significantly faster performance whereas red indicates significantly slower performance. Lower is better.

writing [Ado]. This benchmark result serves as secondary statistical data source for the selection process of the backend technologies.

Startup metrics (lighthouse with mobile simulation)

Name	vanillajs	svelte-v3.29.4	vue-v3.0.2	react-redux-hooks-v17.0.1 + 7.2.1	react-hooks-v17.0.1	angular-v11.0.2	react-v17.0.1	react-redux-v17.0.1 + 7.2.1
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	1,956.8 ± 0.3 (1.00)	1,955.8 ± 0.3 (1.00)	2,106.2 ± 0.5 (1.08)	2,585.8 ± 0.9 (1.32)	2,581.5 ± 1.3 (1.32)	2,449.5 ± 1.9 (1.25)	2,582.4 ± 0.8 (1.32)	2,692.9 ± 1.6 (1.38)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	16.0 ± 0.0 (1.00)	16.0 ± 0.0 (1.00)	16.0 ± 0.0 (1.00)	53.1 ± 1.2 (3.32)	16.0 ± 0.0 (1.00)	130.6 ± 2.7 (8.16)	16.0 ± 0.0 (1.00)	56.2 ± 1.6 (3.51)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	150.5 ± 0.0 (1.03)	146.0 ± 0.0 (1.00)	196.6 ± 0.0 (1.35)	281.0 ± 0.0 (1.93)	271.5 ± 0.0 (1.86)	294.7 ± 0.0 (2.02)	272.6 ± 0.0 (1.87)	289.7 ± 0.0 (1.98)
geometric mean of all factors in the table	1.01	1.00	1.13	2.04	1.35	2.74	1.35	2.12

Figure 3.3: Startup metrics of the selected frameworks. Green colored fields indicate significantly faster performance whereas red indicates significantly slower performance. Lower is better. This part of the benchmark has been included for the sake of completeness but is considered negligible for the purposes of this work as it concerns a one-time cost not relevant at runtime.

Framework	Static Reads (Req/Sec)	Writes (Req/Sec)	Reads (Req/Sec)
Spring Boot	19356	6426	5777
Express / Node.JS	11110	1310	2533
Ruby on Rails	2072	874	896

Table 3.2: Conclusion of Sergei Sizov's benchmark of web backend frameworks [Siz].

Transmission Benchmark

For the sake of comparison and subsequent selection of transmission protocols, a benchmark of three of the state-of-the-art transmission implementations, which are applicable of the aforementioned given the constraints, have been found during literature review which has been conducted by Meiyappan Kannappa [Kan]. The results are shown in Table 3.3. This benchmark result serves as secondary statistical data source for the selection process of the transmission technologies.

Memory allocation in MBs ± 95% confidence interval

Name	vanillajs	svelte-v3.29.4	vue-v3.0.2	react-redux-hooks-v17.0.1 + 7.2.1	react-hooks-v17.0.1	angular-v11.0.2	react-v17.0.1	react-redux-v17.0.1 + 7.2.1
ready memory Memory usage after page load.	1.1 (1.00)	1.1 (1.01)	1.2 (1.16)	1.4 (1.32)	1.3 (1.27)	1.8 (1.72)	1.3 (1.24)	1.4 (1.35)
run memory Memory usage after adding 1000 rows.	1.6 (1.00)	2.7 (1.69)	3.5 (2.22)	5.2 (3.29)	3.9 (2.44)	4.2 (2.66)	4.4 (2.76)	7.4 (4.62)
update eatch 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	1.9 (1.00)	3.0 (1.57)	3.7 (1.94)	6.0 (3.15)	4.7 (2.43)	4.6 (2.39)	5.2 (2.71)	8.3 (4.35)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	2.2 (1.00)	3.2 (1.46)	4.1 (1.84)	6.0 (2.69)	4.6 (2.09)	5.0 (2.24)	5.2 (2.34)	8.2 (3.69)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	2.3 (1.00)	2.4 (1.06)	2.6 (1.15)	3.0 (1.33)	3.0 (1.31)	3.5 (1.53)	3.0 (1.31)	3.1 (1.37)
geometric mean of all factors in the table	1.00	1.33	1.60	2.18	1.83	2.07	1.95	2.68

Figure 3.4: Memory allocation of the selected frameworks. Green colored fields indicate significantly lower memory allocation whereas red indicates significantly higher memory allocation. Lower is better.

Transmission Type	MQTT	WebSockets	gRPC
1000 messages (1.5 KB)	142 ms	16 ms	10 ms
10000 messages (1.5 KB)	1326 ms	127 ms	40 ms
100000 messages (1.5 KB)	2613 ms	911 ms	219 ms
1000 messages (4 KB)	250 ms	26 ms	10 ms
10000 messages (4 KB)	1696 ms	206 ms	38 ms
100000 messages (4 KB)	6880 ms	1767 ms	257 ms

Table 3.3: Results of Meiyappan Kannappa's benchmark of transmission protocols for reference implementations [Kan].

Overall, gRPC has the best results, in every category. Furthermore, WebSockets outperform the message broker MQTT consistently in every category, making MQTT the worst

performer of the three. However, some of these protocols are more suitable for certain requirements and use cases than others, as already laid out in the state-of-the-art section 1.3.1. For example, WebSockets are a standard that is natively supported by modern web browsers since they are HTTP/1.x compatible without the need for additional compatibility libraries/layers [Tan].

3.4.2 Software Stack

Based on the previous findings, through the benchmark results and comparisons, a stack has been composed as described in the following.

- Frontend
  - Svelte 3.35
  - Chart.js 2.9
- Backend
  - Spring Boot 2.5
  - OpenJDK 11
- Persistence
  - Firebase Realtime Database (required)
  - PostgreSQL 13
- Communication
  - HTTP(S)/REST API
  - WebSockets
  - Firebase Admin SDK

Additionally to the WebSocket and Firebase Admin SDK interfaces, an HTTP-based REST API is used for non-time-critical data.

As the contextual data which is required as well, such as user ID, needs to be stored securely to associate the performance data with the corresponding user and their individual programs, a persistence strategy is required. The solution of choice here is PostgreSQL, which provides sufficient functionality. For this data, performance is less relevant as it is not critical for the live presentations and can be loaded once at startup. Time critical data on the other hand, uses a realtime database, the Firebase Realtime Database which is accessed via the Firebase Admin SDK.

Since this work deals with time-critical data (partially) processed over various stages, visualization is an aid to support the facilitation of insight. For this, Chart.js has been chosen and used in version 2.9 which was the most recent at the time of composition.

Some of these components require additional accompanying libraries, compilation/building or runtime environments and also feature transitive dependencies which will be addressed in Section 3.5.

3.4.3 Network Architecture

As enumerated in the State-of-the-Art 1.3 and evaluated in the previous Section 3.4.2, the computer network architecture is designed around the constraints and found requirements using the composed software stack. This leads to a network topology model which can be seen in 3.5 where multiple edge devices with a pair of Smart Gloves each are connected to the MOM which multiple microservices are subscribed to and further process data for the connected web frontends. The backend layer consisting of distributed microservices can be orchestrated in a typical cloud fashion.

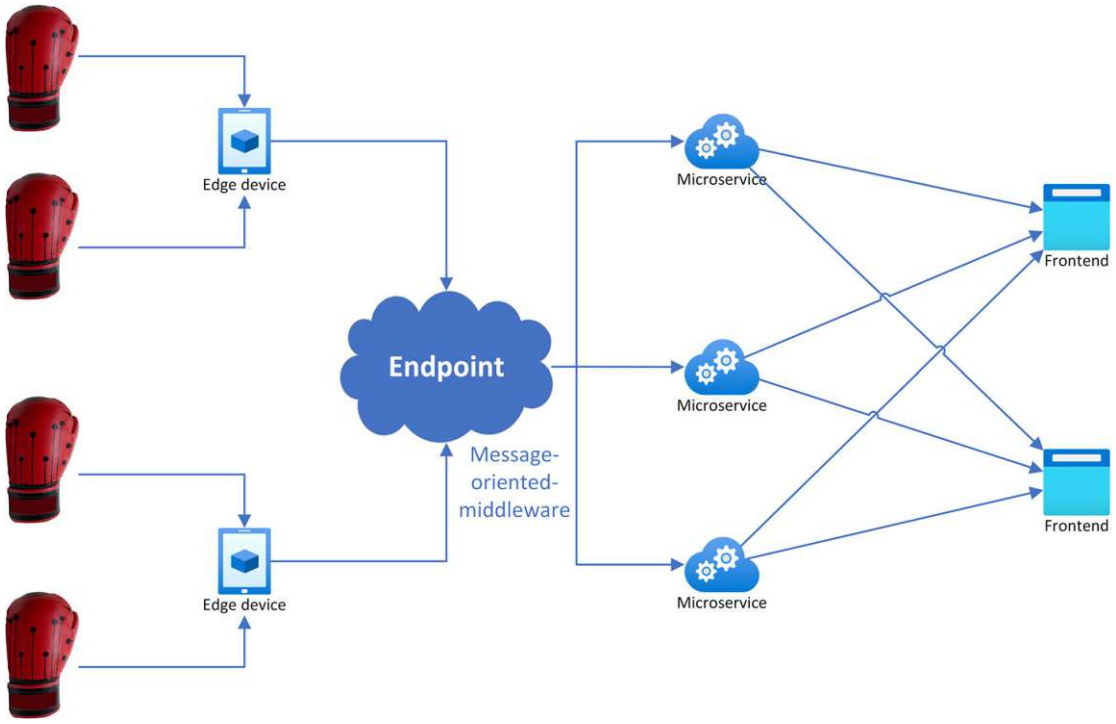


Figure 3.5: An illustration of the computer network architecture/topology. The individual microservices connected through the MOM are considered cloud infrastructure.

The practical operation of this setup required a considerable effort to configure properly and setup the desired ports, connections and security certificates across multiple devices. The Firebase Realtime Database had to be made accessible via the offered web interface where a token had to be issued that had to be integrated into the backend layer of the system where it had to be included in the configuration of the Firebase Admin SDK. On the end, the mobile edge device had to also be configured to work with the Firebase

Realtime Database instance using the Firebase Android SDK. The backend layer using Spring Boot had to be also configured to secure the HTTPS and WebSocket endpoints and make them available to the frontend as well as support proper authentication. The connection between the backend and frontend layer will be detailed in the following sections.

3.4.4 Data Transmission, Processing and Interfaces

As this is a distributed system at the core, the processing of the data can happen at multiple stages and nodes. In-between those, data has to be transmitted using interfaces. For example, a possibility would be, to transmit all of the data to the frontend in the browser to aggregate and process it there. This would have been inefficient, however, as this is an environment which is highly dependent on the capabilities of the device and there is more overhead in a browser. So, to leverage the distribution and inherent parallelization of the system, the data processing workload is spread across as many network nodes as possible. This means that early preprocessing, i.e. mapping to suitable data types, filtering of irrelevant data and compression already happens at the edge device. Then, aggregation takes place at the service layer within the individual microservices and evaluations (e.g. statistical analyses) are computed and cached. At the frontend, the prepared data from the backend is sorted and grouped by date, user and session for visualization.

There are different interfaces used for different kinds of data. Non-time-critical data will be exposed via a-based REST API. Time critical data is accessible via WebSockets.

REST API

Authentication and retrieval of past event data is carried out via an HTTPS-based REST API as defined by the requirements 3.2.1. The REST API is documented using OpenAPI and Swagger, see Appendix 5.4.

WebSocket Endpoint

The WebSocket endpoint uses STOMP messaging to transfer data organized as topics. Data is being transferred via the pub/sub messaging pattern. There, it is also serialized and deserialized from/to JSON respectively. This is used for the live transmission of punch event data. This also satisfies the functional requirement specified in 3.2.1.

3.4.5 Performance Test Design

Building on top of the methodology described in Chapter 2, the performance tests have to be designed specifically around the selected and composed technologies. Hence, to evaluate the system's performance, predominantly system (scope) tests are relevant that test end-to-end performance of the entire system, as described in 1.2.3, 1.2.4 and 1.2.4. As such, the response time from the edge device, which is the first point in the system

where the sensor data is received from the data source (i.e. Smart Gloves) and from there, all the way to the end user in the web browser/frontend is measured. Figure 3.6 visualized the performance measurement segment.

As there are many factors with influence on the measured time, four different test runs have been designed as part of laboratory experiments. It is distinguished by connection type, cellular and local Wi-Fi, as those are a known influence on network latency. Furthermore, a second set of test runs is set to be conducted after the findings and observations of the first/unoptimized test runs have been used as baseline for optimizations of the prototype implementation. The evaluations are detailed in Chapter 4 which uses standardized conditions with the equipment described in 3.4.1.

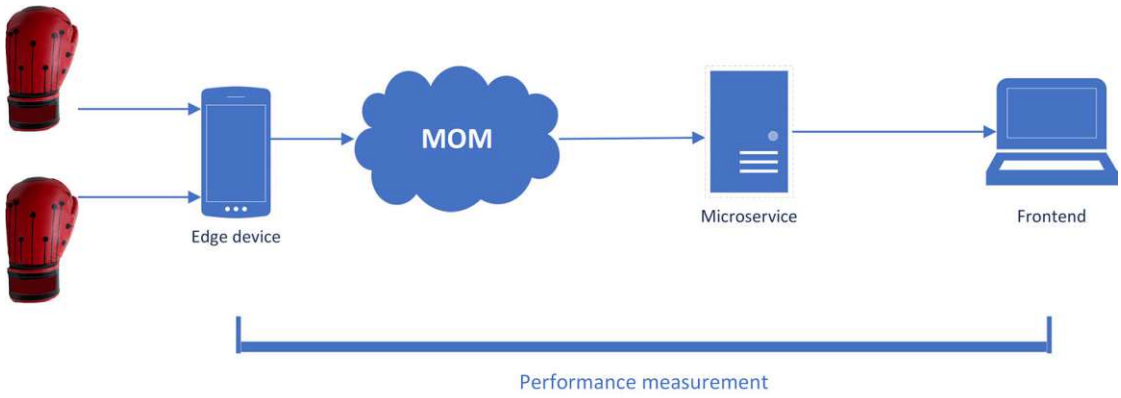


Figure 3.6: A diagram of the path within the system for which the performance is measured.

The time of each punch is be measured in milliseconds. Test runs are set to 100 punches each which is also the sample size. Each sample will be evaluated using the following quantitative analysis measures:

- Mean
- Median
- Standard deviation
- Variance
- Range
- Significance test
  - p values
  - t test

Each test run has to be conducted in the same location with the same devices, also the same glove prototypes used against the same punching bag, and similar distance between the device, e.g. Wi-Fi access points, edge device and the Smart Gloves. Furthermore, the clocks of the individual devices are synchronized before each test run. All of these measures are precautions to minimize interference and unintended deviations.

3.4.6 Security

As described in 1.3 security is a fundamental concern. Security is applied by using authentication to access endpoints secured by TLS which constitutes to HTTPS. For this, network packets are inspected and encryption is verified. The software design, implementation as well as configuration thereof are verified using static analysis approaches and state-of-the-art tools.

Authentication

State-of-the-art authentication can be achieved using JSON Web Token (JWT) [JBS]. That enables the backend server to operate in a sessionless and stateless fashion, thus further facilitating scalability. Furthermore, passwords are secrets which should not be stored as plain-text or message digest but hashed and slated according to NIST [GFN+17]. Therefore, passwords are hashed and salted using the novel state-of-the-art hashing algorithm Argon2 [BDK16] so they are not stored as plain text in the database. In addition, the authorization and authentication is also present on the edge device which is based on Firebase Authentication.

Spring Security

As previously found in Section 1.3.1 the JVM and the Spring Framework offer solid security concepts and considerations as a technical foundation at the backend. A major component in the Spring Framework for security is Spring Security. It includes configurable means for authorization, authentication (including password encoding, session management and request filters) as well as countermeasures for cross-site-request-forgery (CSRF), cross-site-scripting (XSS) and brute force attacks [NB19]. This functionality is applied and validated for role management and authorization, in combination with the previously described JWTs. It also supports TLS/HTTPS which is enabled via configuration. These considerations are in line with the security concepts, especially confidentiality and integrity, presented in 1.2.5.

Data Protection and Privacy

To comply with the law currently in effect, data access must be restricted depending on user privileges and logged [God17]. In some instances, data needs to be anonymized. This section is devoted to design decisions to satisfy these legal requirements. By default, user-related data will only be transmitted where necessary. The transmission is further

secured using TLS. User ID info will be represented as (pseudo)randomly generated numbers unless they choose a nickname which will be used publicly by the system.

Security Verification

The security of the prototype is evaluated and verified using a set of state-of-the-art tools as outlined in the research questions and hypotheses in 1.6. These tools are SonarQube, OWASP Dependency-Check, Snyk and npm audit. Using these tools, SAST for white box security analysis and code quality improvement as well as SCA for dependency analysis and CVEs known at that point in time are performed. Beyond that, SonarQube and Snyk also verify configurations for security issues. These checks include configuration of exposed ports, communication mechanisms, password encoding, error handling and logging [PDVH19]. Using this combined approach, an in-depth and comprehensive assessment of the security aspects is provided [Sca21, MTBHBH+20].

3.4.7 Persistence and Caching

The data model and entity relationships are designed and modelled as can be seen in Figure 3.7. As described in Section 3.4.6 a major concern was to collect minimal necessary user data. In total, the amount of database tables has been kept to a number of ten. As laid out in the use case description, the site user (represented via the user_account table and its associated roles is not the same as the participant in the actual model (but could be the same person in reality). Punch measurement data is associated with a punch which in turn is associated with a session which allows for evaluation and report generation.

Furthermore, critical data such as live punches are cached in memory for fast access. These are persisted at a later point asynchronously to minimize the impact on the live system and reduce latency.

3.5 Prototype Implementation

This section goes in-depth into the details of the prototypical implementation using the methodological approach described in 2.3 and composition of the previously established software stack. The goal of this section is to create an implementation of the requirements 3.2 while respecting the constraints 1.4.

3.5.1 Development Process and Environment

Each part of the software solution requires different build tools which are detailed in the respective sections. These tools are specified and combined in the build environment and pipelines. Development itself has been carried out in IntelliJ and Visual Studio Code as integrated development environments. Development progress and the source code has been organized and maintained using Git. A private git repository has been

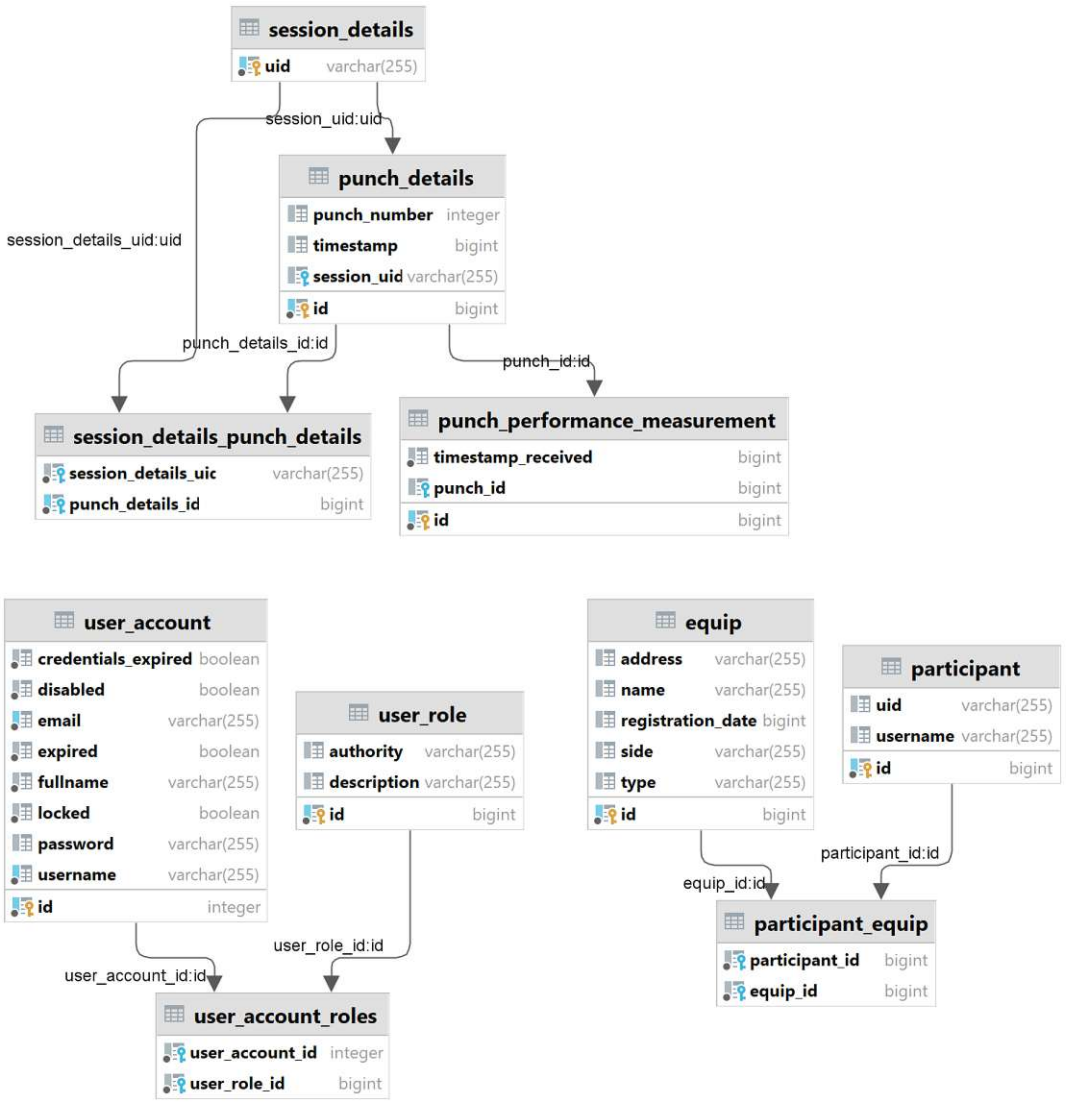


Figure 3.7: Database model as entity relationship diagram (ERD).

used at GitHub. For details about the hardware and operating system details see 3.4.1. Builds have been performed and deployed using Docker and made available on a private virtualised Linux server.

3.5.2 Backend

The backend implementation is done using Spring Boot on a Java 11 LTS basis, as previously established in 3.4.2.

Data Structure

To fulfill the requirements and implement the design laid out in Section 3.4, a structure had to be designed capable of doing so. In a typical object-oriented fashion, classes have been created to encapsulate structure, information and functionality to be subsequently used via objects which are instances thereof [PGK17]. They are loosely presented in Figure 3.8. These include service interfaces and implementations, their inheritance relation as well as data model entities and their corresponding data transfer object (DTO) classes.

Figure 3.8: Loose overview of backend classes as UML class diagram.

Reactivity

The backend implementation makes use of Project Reactor to transmit data to the connected clients asynchronously using Flux [PNELM21]. This reduces the need for looping on baseline Java data structures and (long) polling. Project Reactor has been used instead of the whole Spring WebFlux as intermediary data is stored for ease of access and evaluation in a relational database which is not supported by Spring WebFlux. The implementation avoided performance intensive operations as outlined by Ponge et al. [PNELM21].

Data Processing

At the backend, data is acquired by subscribing to the respective topics of the Firebase Realtime Database which can be compared to a traditional message queue PubSub pattern. The data is then aggregated and provisioned for the individual connected clients based on the roles of the authenticated user. As described in 1.4.

Persistence

Through the backend, historical data about past sessions is stored in the PostgreSQL database for statistical analyses. This is accessed securely via Spring Data JPA and its CRUD repositories [GDS+]. The actual database connection underneath is done using the JDBC driver for PostgreSQL [Grod].

APIs

There are two APIs designed on the backend layer that are exposed to web clients such as the frontend section of the prototype. These two APIs are an HTTP-based REST API and a WebSocket API.

REST API As described in section 3.4.4, the REST API has been implemented using Spring Boot and its REST capabilities. The REST API has been documented using OpenAPI V3 and springdoc-openapi maven plugin. It is exposed at the backend using a TLS-secured port. The access is restricted and configured using Spring Security.

WebSocket Endpoint In addition to the REST API, there is a connection based WebSocket endpoint using STOMP and SockJS as underlying technologies. This enables pub/sub functionality such that clients can subscribe to live events such as live incoming punches.

Build

The backend build is done using Maven. The output of which is deployed using docker. The underlying web servlet is provided via Apache Tomcat in both production and development environments.

3.5.3 Frontend

Suitable practices and approaches for the frontend development process are covered to a large extent by the works Modern Full-Stack Development by Frank Zammetti [Zam20] and Practical Svelte by Alex Libby [Lib22] which are used as references. This also applies to state management and reactivity considerations.

In contrast to the backend, functionality and information is encapsulated within components which may contain logic as well as UI-descriptions/views at the frontend. They also may reuse other components which constitutes the composite software design pattern [Rie97].

User Interface

The UI has been designed using Svelte and the notus-svelte template [Tim] which includes Tailwind CSS [Labb]. The flow of activities through the web application has been modelled using a UML activity diagram in Figure 3.9.

The dashboard-based UI is structured to accommodate these activities and is presented in Figure 3.10. It uses cards which create an overview of sessions. Each card represents a session of a user. The cards are grouped by date, the most recent ones being on top. Live sessions are therefore displayed at the very top. In each card, there is a circle with a value representing the most recent value for a selected field. The fields are based on the available metrics:

Beyond that, also means of visualization are applied to support comparisons and insight into the athletes performance. These are in the form of stylized line charts presenting the individual performances of each of the participants by using the set metric and its value over time. This poses a visual aid beyond textual representations in comparing different participants or boxing sessions of participants with each other. The dashboard with charts enabled is presented in Figure 3.11 and illustrates an example where two athletes' performances can be compared with each other in detail. Both the textual as well as the chart representations support the same metrics as described in Section 3.3.

The underlying components are composited as follows (elementary UI inputs such as buttons or text input fields are omitted for the sake of simplicity):

- Start page component (Index)
- Authentication layout component (Auth)
  - Login component
- Boxing dashboard layout (Boxing)
  - Boxing header component with settings (BoxingSessionsHeader)
  - Session components (Sessions)

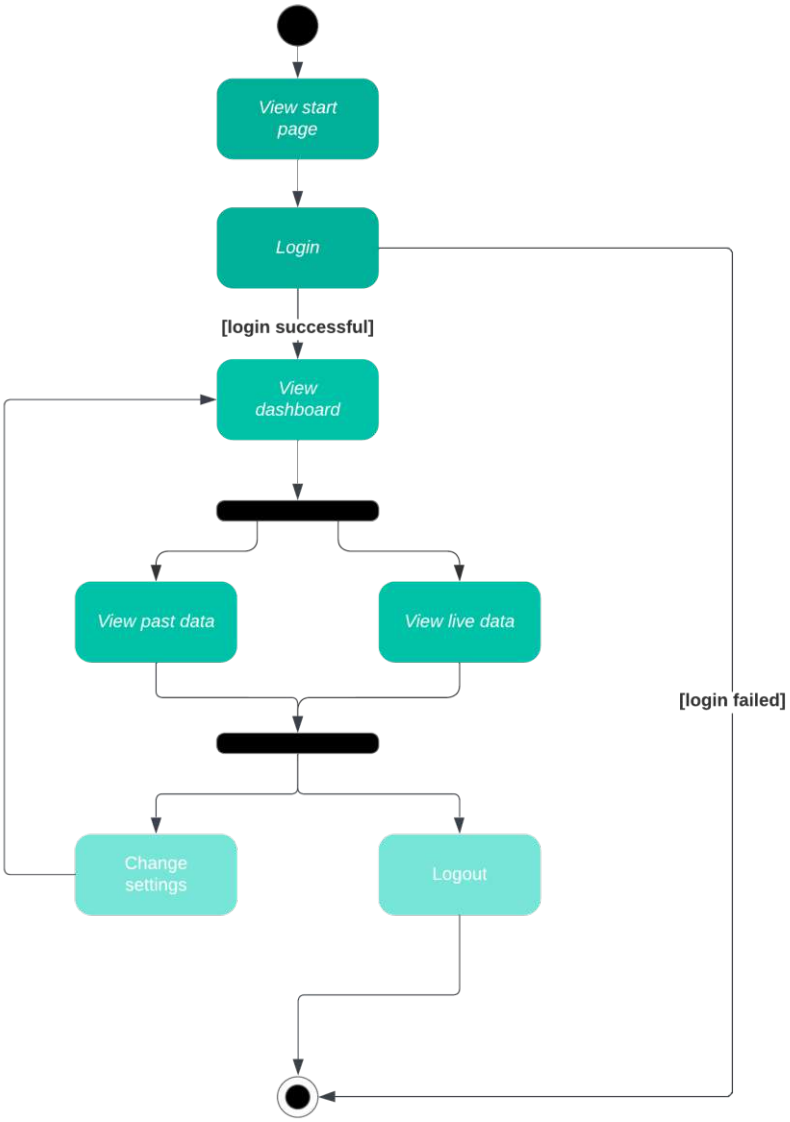


Figure 3.9: An overview of typical activities on the frontend modelled using a UML activity diagram.

* Card component displaying the performance of a session of a participant (CardParticipantPerformanceDetail)
* Card component displaying the visualization of the performance (CardParticipantPerformanceChart)

The pages of the boxing dashboard are protected by an auth-guard implementation which prevents users from opening the dashboard without having a valid authentication token.

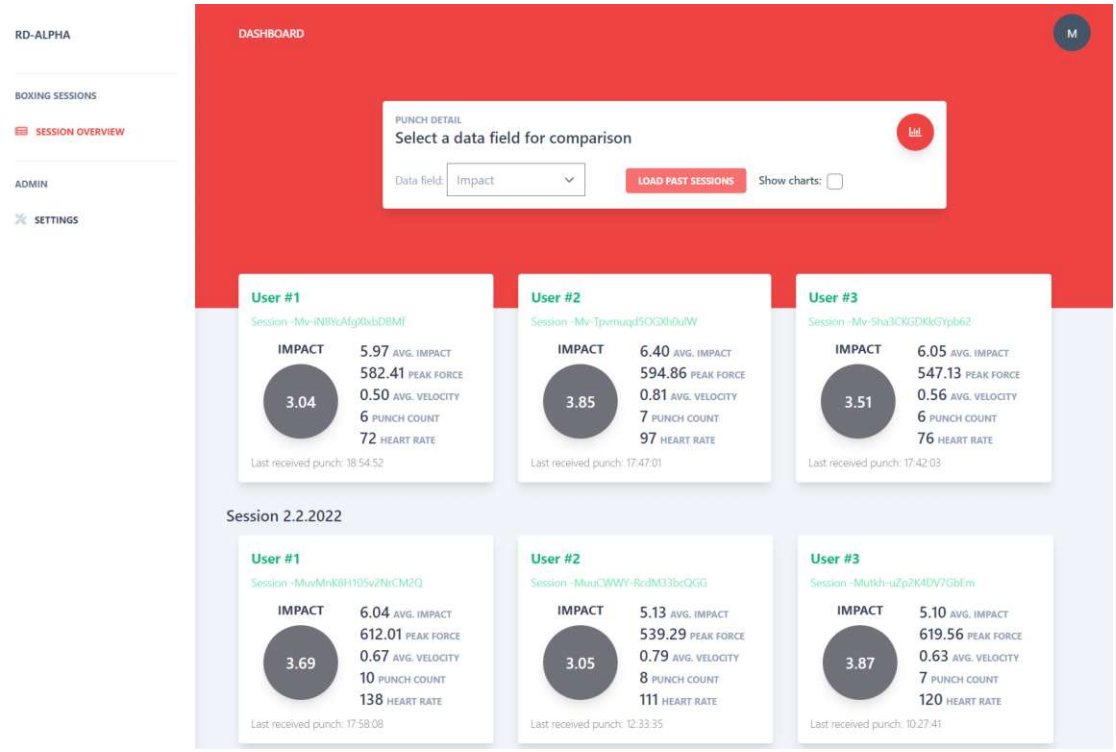


Figure 3.10: A screenshot of the dashboard with anonymized participants showing their individual performance of past and most recent activities.

This is another measure to incorporate an authentication process as described in 1.2.5.

As an example, the Svelte code of the session component is presented in Figure 3.12. The presented component triggers the loading of the required participants and establishes a session which is updated live as data is streamed in from the cloud backend services. This example makes use of reactive assignments using the JS-label syntax and underlying Svelte stores. Eventually, this is compiled to efficient and reactive JavaScript, CSS and HTML.

Logic

On the frontend, there is also a set of calculations performed as data is received incrementally on this end during live sessions. For this, complexity considerations have been taken into account, to calculate average and maximum values. Both, the maximum and the average values are computed using ES6 array reduce functions running in $O(n)$.

Furthermore, the punch data is processed based on its timestamps. Since the current ECMAScript capabilities regarding date and time are limited (and subject to change with the Temporal [DJPJP+] proposal), a library is used to assist with calculation, namely luxon [Lux].

Build

The build environment of the frontend is composed out of Node.JS (including NPM) and a docker container based off node on Alpine Linux.

3.5.4 Transmission

As described in the constraints section of the introduction 1.4.2, the Smart Gloves emit data via Bluetooth. This data is ASCII-encoded CSV data which is received and partially processed by the edge device. The flow can be seen in Figure 3.6 where the Smart Gloves act as IoT-sensors and the edge device pre-processes the received data and sends the results to a network endpoint.

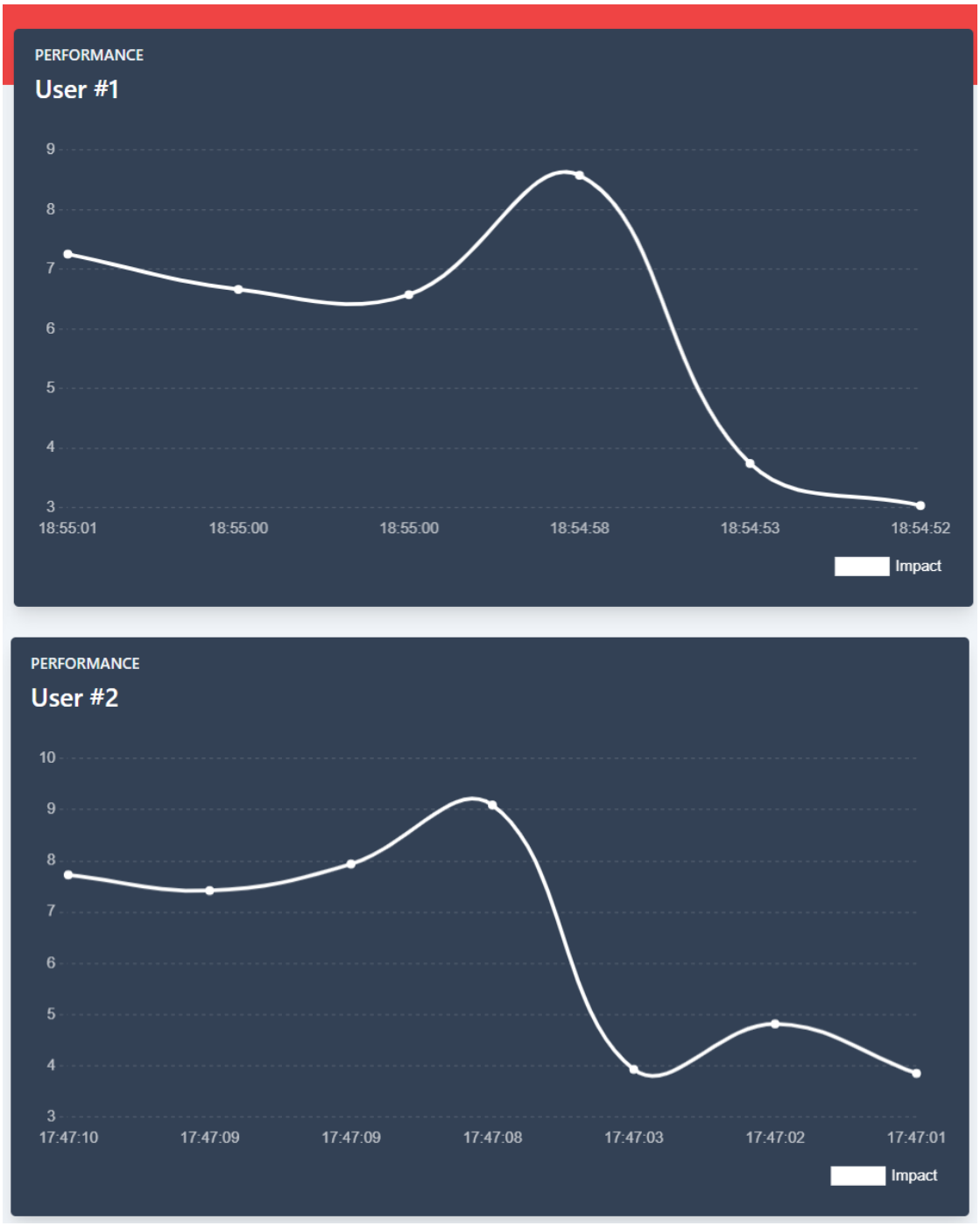


Figure 3.11: A screenshot of the dashboard with charts of the performances of anonymized participants, in this case their impact values over time. This supports the direct visual comparison of different participants or boxing sessions of a single participant.

```svelte
<script>
    import {chartsEnabled, punches, selectedField} from "../../stores/punch.store.js";
    import CardParticipantPerformanceDetail from "../../components/Cards/CardParticipantPerformanceDetail.svelte";
    import {participantService} from "../../services/participant.service";
    import {punchService} from "../../services/punch.service";
    import CardPunchDetailChart from "../../components/Cards/CardPunchDetailChart.svelte";

    participantService.retrieveParticipants();
    punchService.connectToSession();
</script>

<div>
    {#each [...$punches].reverse() as [sessionDate, session]}
        <h1 class="font-semibold text-xl py-3 text-blueGray-700">
            Session {sessionDate}
        </h1>
        <div class="flex flex-wrap">
            {#each [...session].reverse() as [sessionId, sessionPunches]}
                <CardParticipantPerformanceDetail
                        punches={sessionPunches}
                        selectedField={$selectedField}
                        participantId={sessionPunches[sessionPunches.length - 1].participantId}
                        sessionId="{sessionId}"
                />

                {#if $chartsEnabled}
                    <CardPunchDetailChart
                            punches={sessionPunches}
                            selectedField={$selectedField}
                            participantId={sessionPunches[sessionPunches.length - 1].participantId}
                            chartId={sessionId}
                    />
                {/if}
            {/each}
        </div>
    {/each}
</div>
```

Figure 3.12: The source code of the Svelte component used for the live presentation of individual training sessions. In typical Svelte fashion, it encapsulates logic and layout which is then compiled to efficient and reactive JavaScript, CSS and HTML.

CHAPTER 4

Evaluation

In this chapter the prototype is evaluated and tested in regards to the research questions and confirmation of hypotheses described in 1.6. This involves the methods of laboratory experiments, observations and measurements and quantitative analysis as described in 2. It focuses on both an unoptimized version of the prototype where the findings of its evaluation revealed the necessity of optimizations that lead to an optimized version of the prototype.

4.1 Performance

Performance is measured from the point where the sensor data is received by the mobile device to where the data arrives in the end user's browser. For reference of the performance test design, see Section 3.4.5. The mobile device was connected to the internet through either Wi-Fi or cellular networks with varying signal strengths over the courses of the performance tests. The measured latency varies based on interference, device characteristics (antennas of both access point and modem) and connection type. The test runs distinguish between the connection type: cellular and local Wi-Fi. Multiple performance test runs have been conducted. Before each test run, the devices' clocks have been synchronized using the same NTP server. The subsections 4.1.1 and 4.1.3 show the test run results pre and post optimizations. For both of these unoptimized results, there is an obviously large range and standard deviation.

4.1.1 Unoptimized Build

The initial feature-complete version of the the prototype implementation is referred to as the unoptimized version. This version has been evaluated using the two defined test runs of 100 punches each on both connection types. The nature of the measurement data collected during the test runs is visualized in the point charts in 4.1 and 4.2.

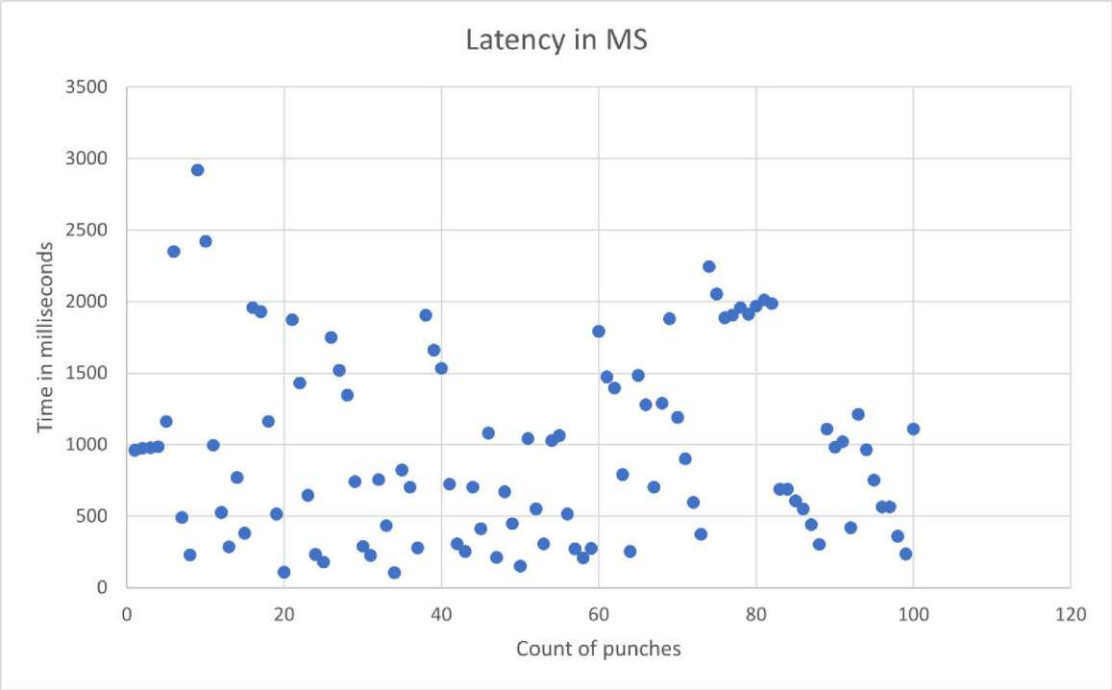


Figure 4.1: A point chart of the data measured using the unoptimized prototype build and a 4G cellular connection. On the vertical axis the time in milliseconds, on the horizontal axis is the number of the punch.

Measures	Cellular	Wi-Fi
Sample Size	100	100
Mean	966.76	721.12
Median	781	592.5
Standard deviation	645.64	449.73
Variance	421066.57	204302.83
Range	2813	2118

Table 4.1: Performance measurements of the unoptimized build in milliseconds using cellular 4G and Wi-Fi. Times are in milliseconds.

In addition to the visualization, a statistical summary is presented in Table 4.1. It can be observed that all statistical variables are lower and therefore better using a Wi-Fi connection.

However, neither of the unoptimized results reach the required average latency of less than 300ms average as constituted by H1. Despite that, this proves that a system is feasible, albeit with a non-desirable delay, and thus answering RQ2 of the research questions defined in 1.6 using the method of developing and testing a prototype.

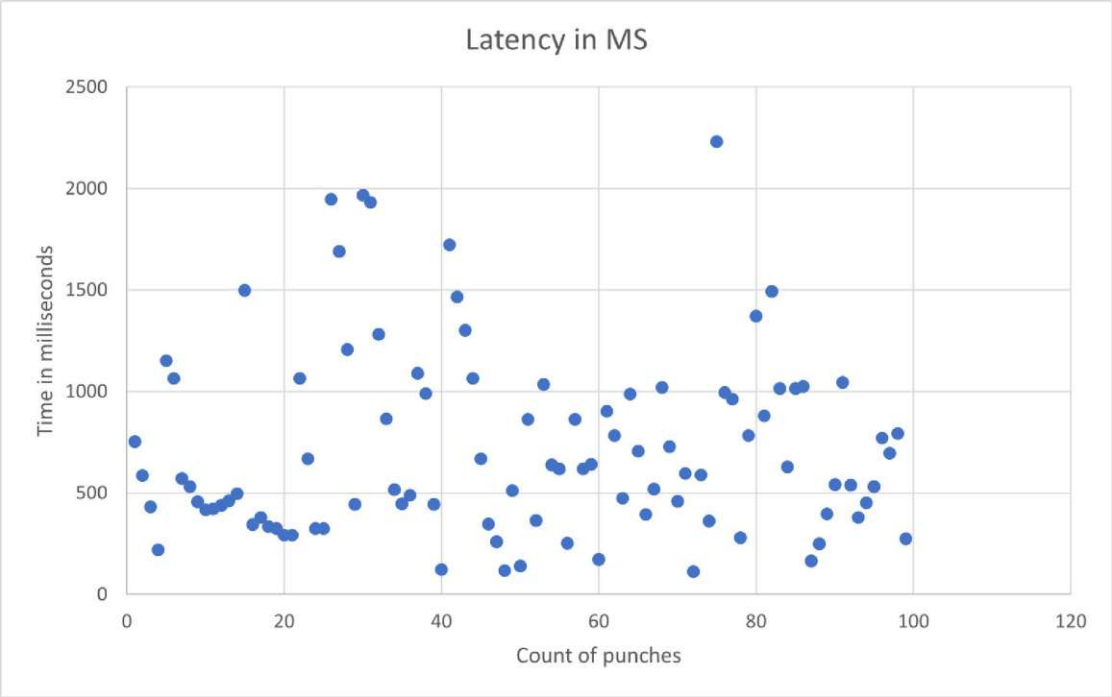


Figure 4.2: A point chart of the data measured using the unoptimized prototype build and a strong Wi-Fi connection. On the vertical axis the time in milliseconds, on the horizontal axis is the number of the punch.

4.1.2 Optimization

Based on the findings and first test run results (see 4.1.1) which did not meet the performance targets and hypothesis, the prototype has been analyzed for shortcomings and optimization options have been explored. These optimizations constitute the final part of the implementation process. The optimizations include:

- Tweaking the order and traversal of existing punch data to improve insertion of new punch data
  - Using session ID and punch number as unique identifiers and order data based on that in a map
  - This also improves iteration for presentation
- Asynchronous performance measurement persistence operations instead of synchronous or blocking persistence operations
- Remove redundant debug features such as unnecessary logging and introduction of adequate logging levels (such as DEBUG)

- Upgrade of dependencies to latest versions (at the time of development) to mitigate security concerns and benefit of their optimizations
  - Svelte from 3.35.0 to 3.46.4
  - Spring Boot from 2.5.6 to 2.6.3
  - Reactor from 3.4.11 to 3.4.15
- Mitigate security issues (see 4.2 found in configurations by tightening the configurations and applying suggestions provided by the tools

Using these, significant improvements of performance have been achieved and security issues resolved, all of which is explored in detail in the following. This poses the answer research question RQ4 (as defined in 1.6 as to how significant performance improvements can be achieved as later proven by the evaluation of the observations. Therefore, the methods used to answer this research question are observations and the subsequent optimizations as well as quantitive analysis thereof.

4.1.3 Optimized Build

This section is devoted to a prototype implementation which incorporates the optimizations detailed in 4.1.2, hence referred to as optimized build. Again, two test runs on both connection types have been carried out in the same environment as with the optimized build. The measurement data collected during the test runs is visualized in the point charts in 4.3 and 4.4.

A comparison has been performed of the unoptimized against the optimized build measurements, for both cellular and Wi-Fi connections. In Figure 4.5, all four performance measurements are illustrated next to each other using colored bins to illustrate the latency where lower is better and low is colored as green and high is colored as red.

The results for the cellular based comparison are presented in Table 4.2. The results for the Wi-Fi based comparison are presented in Table 4.3. It can be observed that a 72.88% times improvement in mean performance has been achieved with optimizations using a cellular connection. Using a Wi-Fi based connection, an improvement of 71.22% has been achieved. Additionally, t Tests for statistical significance are performed as well.

Measures	Unoptimized	Optimized
Sample Size	100	100
Mean	966.76	262.19
Median	781	239
Standard deviation	645.64	110.18
Variance	421066.57	12261.77
Range	2813	541

Table 4.2: Performance measurements of the unoptimized and optimized build using a cellular connection. Times are in milliseconds.

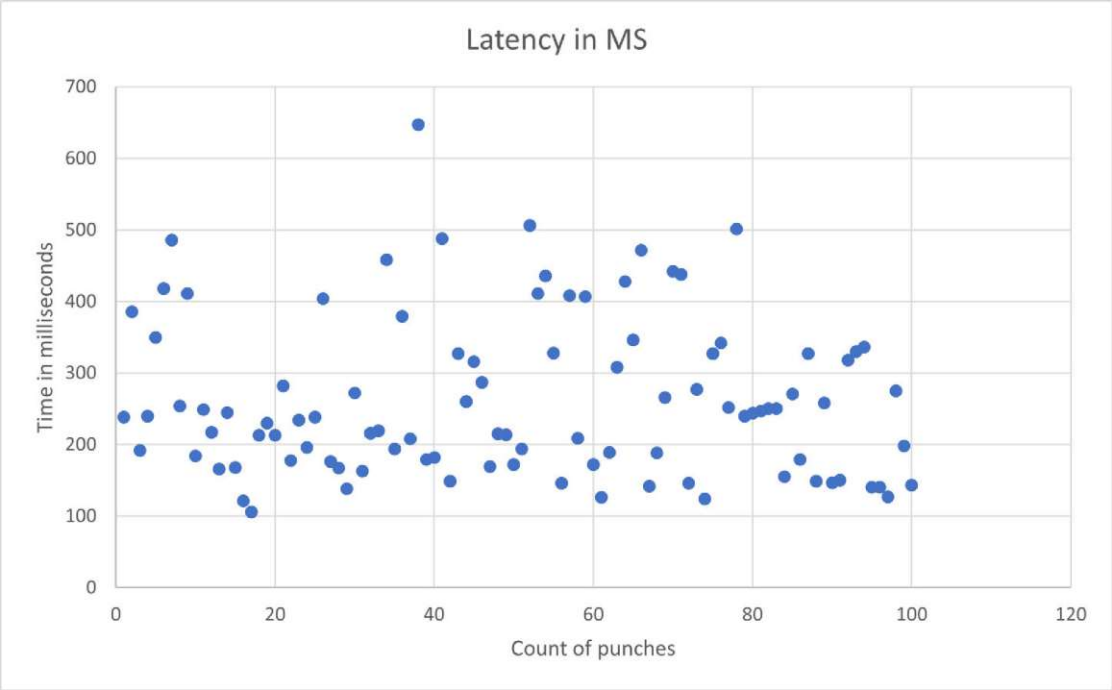


Figure 4.3: A point chart of the data measured using the optimized prototype build and a 4G cellular connection. On the vertical axis the time in milliseconds, on the horizontal axis is the number of the punch.

Measures	Unoptimized	Optimized
Sample Size	100	100
Mean	721.12	207.57
Median	592.5	189.5
Standard deviation	449.73	80.95
Variance	204302.83	6619.74
Range	2118	378

Table 4.3: Performance measurements of the unoptimized and optimized build using Wi-Fi. Times are in milliseconds.

A statistical summary of the test runs of the optimized build is presented in Table 4.4. This is as basis for comparison of the two connection types.

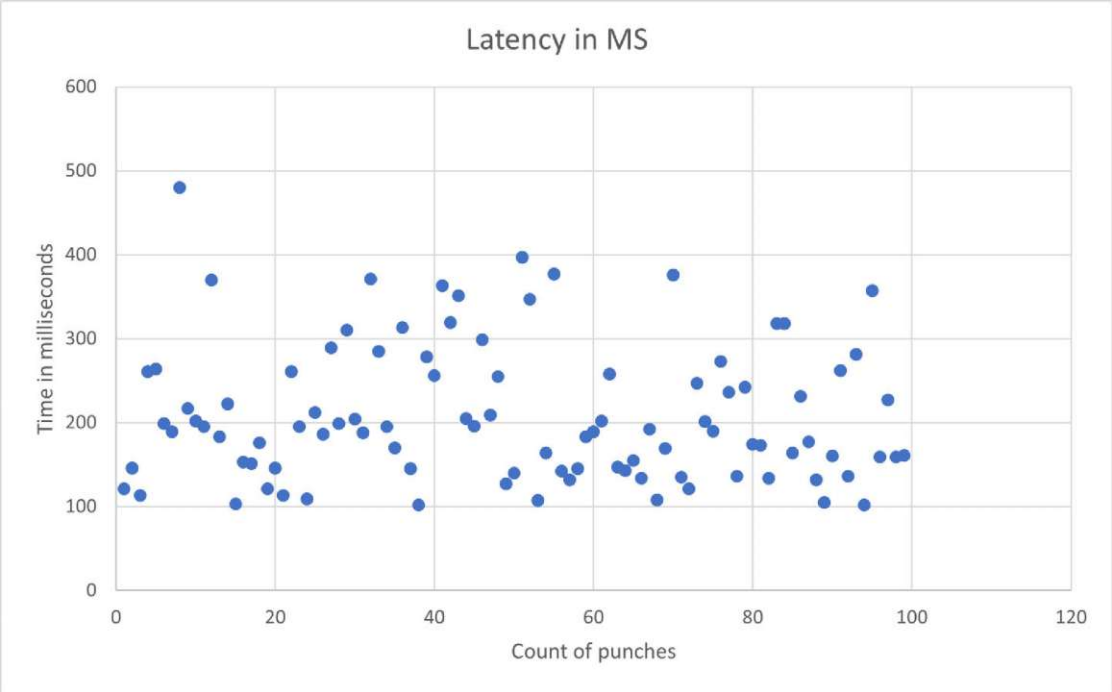


Figure 4.4: A point chart of the data measured using the optimized prototype build and a strong Wi-Fi connection. On the vertical axis the time in milliseconds, on the horizontal axis is the number of the punch.

Measures	Cellular	Wi-Fi
Sample Size	100	100
Mean	262.19	207.57
Median	239	189.5
Standard deviation	110.18	80.95
Variance	12261.77	6619.74
Range	541	378

Table 4.4: Performance measurements of the optimized build using cellular 4G and Wi-Fi. Times are in milliseconds.

For the significance testing from here on out, a two-tailed p-value is calculated as both data sets are known at the point of comparison and no prediction is necessary making the one-tailed values inappropriate. As the test samples are independent, two-sample t-tests are used. These have been performed in Microsoft Excel [Zai] using the unequal variances parameter as the variances are different with a factor of up to 34.33 like in the instance of the cellular-based unoptimized versus optimized samples. An alpha value of $\alpha = 0.05$ is used.

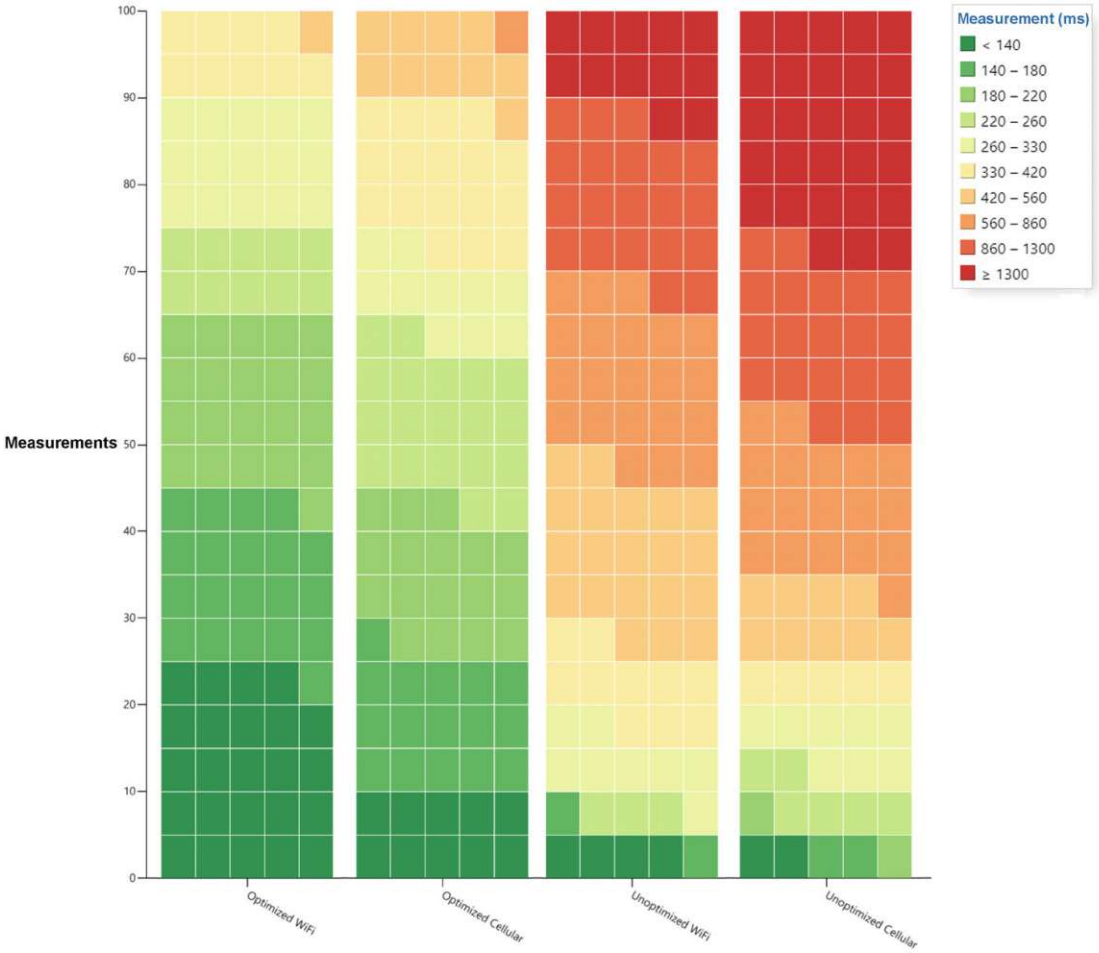


Figure 4.5: A bar chart visualization using heatmapped bins to illustrate the distributions of the measurement data. There are 10 bins representing ranges of latencies in milliseconds as depicted in the legend. The lower, the better. On the vertical axis is the group of data sets, on the horizontal axis the number of punches. This visualization has been created using SandDance by Microsoft Research [Res].

Sample 1	Sample 2	p-value	t-statistics
Unoptimized cellular	Optimized cellular	$1.62 * 10^{-18}$	10.70
Unoptimized Wi-Fi	Optimized Wi-Fi	$1.37 * 10^{-19}$	11.18
Optimized cellular	Optimized Wi-Fi	0.0001014	3.97

Table 4.5: Significance testing of the unoptimized and optimized build samples using both connection types as well as a comparison between the two connection types on the optimized builds.

The results of the significance testing are presented in Table 4.5. The null hypothesis can be rejected on all of the t-tests as the p-values are less than the alpha value $\alpha = 0.05$. Therefore, all of these differences are significant.

There is an obviously significant difference between the unoptimized and optimized samples based on a cellular connection. Furthermore, there is an obviously significant difference as well between the unoptimized and optimized Wi-Fi samples. The difference between a cellular and Wi-Fi based connection using the optimized build version is also significant.

Overall, the best results during the performance tests have been achieved using the optimized build with a Wi-Fi connection, resulting in a 208.43ms mean latency. This proves hypothesis H1 and answers research question RQ3 as defined in 1.6 through the methods of developing and testing of a prototype, laboratory experiments and observations and optimizations thereof.

4.2 Security

During the evaluation of the prototype, a security analysis has been performed. As a part of that analysis, both SAST and SCA have been performed. Again, both on an unoptimized baseline version as well as an improved optimized version.

4.2.1 SAST

To improve the quality and security of the software prototype implementation and to strive towards clean code status, SonarQube has been used to perform SAST continuously. The SonarQube evaluation is for the Java-based backend only as there is no Svelte support yet in SonarQube at the time of writing [Sonb]. The security evaluation has been conducted on all 55 files, including code files and configuration files of the backend. The results of the defined unoptimized version are presented in Figure 4.6.

The findings in the unoptimized version concern solely configuration issues, these are:

- 1 occurrence of a security vulnerability in which a plain-text password encoder has been used in one instance for test/demo purposes
- 1 occurrence of Cross-Site Request Forgery (CSRF) not prevented
- 3 occurrences of e.printStackTrace() which is considered a debug feature and hence insecure

The evaluation results of the version after the optimizations are as can be seen in Figure 4.7. In this overview the A status is achieved which is regarded the highest within the SonarQube metrics.

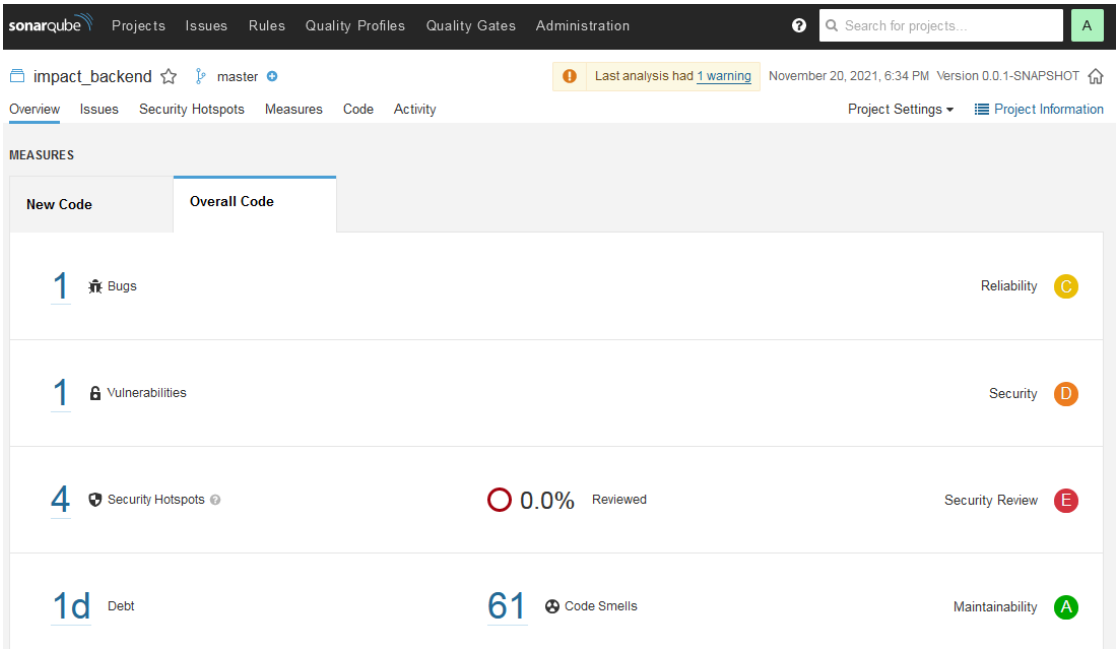


Figure 4.6: The overview of the unoptimized version with the ratings in different categories in SonarQube on 20.11.2021.

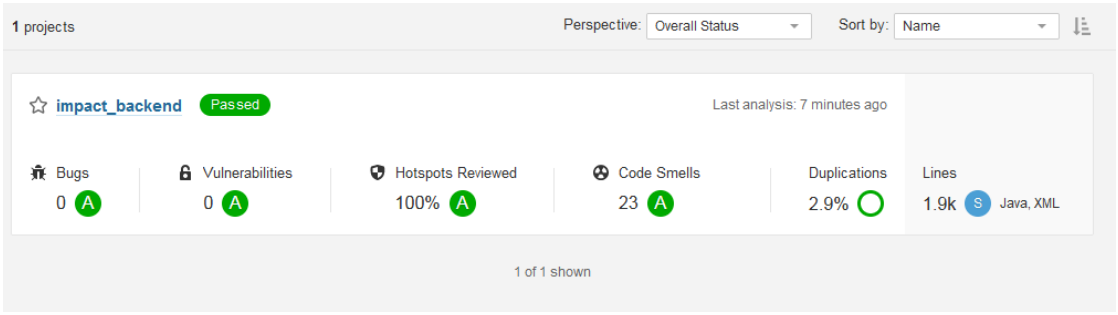


Figure 4.7: The overview of the optimized version with the ratings in different categories in SonarQube on 07.02.2022.

4.2.2 SCA

During the course of the software prototype implementation, SCA has been performed continuously as a method to evaluate and mitigate vulnerabilities or CVEs. Mitigation has been done by upgrading dependencies and configurations. This has been done using the following tools:

- OWASP Dependency-Check
- npm audit

- Snyk

Based on the findings, dependencies have been updated to mitigate found CVEs. Among these were prominent CVEs, such as the Log4j vulnerabilities CVE-2021-44228 and CVE-2021-45046 [WTA, Ana21].

OWASP Dependency-Check

The results of the OWASP dependency-check report contain the following detected CVEs for the unoptimized 4.6 and the optimized 4.7 build.

Dependency Name	Highest Severity	CVE Count
postgresql-42.3.0.jar	CRITICAL	2
spring-beans-5.3.12.jar	CRITICAL	2
jackson-databind-2.12.5.jar	HIGH	1
tomcat-embed-core-9.0.54.jar	HIGH	1
logback-core-1.2.6.jar	MEDIUM	1
netty-transport-4.1.69.Final.jar	MEDIUM	1
spring-core-5.3.12.jar	MEDIUM	1
spring-tx-5.3.12.jar	MEDIUM	1
spring-security-core-5.5.3.jar	LOW	1

Table 4.6: Unoptimized build OWASP dependency-check report results.

The unoptimized build analysis detected a total of four critical CVE instances in the dependencies postgresql and spring-beans, see Table 4.6.

Dependency Name	Highest Severity	CVE Count
tomcat-embed-core-9.0.56.jar	HIGH	1

Table 4.7: Optimized build OWASP dependency-check report results containing one instance of a high severity CVE.

After the mitigations of security issues and concerns, only one instance of a high severity CVE persisted in the OWASP dependency-check report for the optimized build. It concerns a transitive dependency of Spring Boot. This CVE is known under the ID CVE-2022-23181 and is registered in the NIST's NVD with the info "This issue is only exploitable when Tomcat is configured to persist sessions using the FileStore." [oSN]. Hence, this vulnerability is only exploitable under specific conditions and was published on 27.01.2022 which was less then a month old at the time of writing. Therefore, this poses no serious concern and can only be resolved with a future Spring Boot update.

npm audit

The frontend dependencies have been analyzed using npm audit [npm]. The findings of the npm audit are examined in the following. The initial vulnerability analysis results

are presented in Table 4.8. A full list of the vulnerabilities is detailed in the Table 4.9.

Severity	Vulnerabilities
info	0
low	0
moderate	6
high	1
critical	2
total	9

Table 4.8: Number of known vulnerabilities in the used dependencies of the unoptimized build according to npm audit.

Severity	Description
critical	Prototype Pollution in minimist
critical	Authorization Bypass Through User-Controlled Key in url-parse
high	Path Traversal: 'dir/../../filename' in moment.locale
moderate	Regular Expression Denial of Service in browserslist
moderate	Exposure of Sensitive Information to an Unauthorized Actor in nanoid
moderate	Regular Expression Denial of Service in path-parse
moderate	Regular Expression Denial of Service in postcss (3 different ones)

Table 4.9: Vulnerability details of the unoptimized build from the npm audit result.

The vast majority of the npm findings have been resolved using the suggested "npm audit fix" command which automates the mitigation process by finding compatible dependency updates where the security issues no longer exist [npm]. The remaining packages had to be updated manually which required minor adjustments to the code to comply with API changes. The optimized results are presented in Table 4.10.

Severity	Vulnerabilities
info	0
low	0
moderate	0
high	0
critical	0
total	0

Table 4.10: Number of known vulnerabilities in the used dependencies according to the npm audit result on the optimized build.

Ultimately, the list of frontend dependencies (including transitive dependencies) is represented in Table 4.11.

Type	Number of dependencies
prod	188
dev	38
optional	2
peer	2
peerOptional	0
total	227

Table 4.11: Number of dependencies including transitive dependencies categorized by type.

Snyk

The Snyk analyses results for the unoptimized build version are presented in Figure 4.8 and 4.9. These contain 1 security-critical vulnerability with a score of 7.5 out of 10 affecting the moment dependency in the frontend.

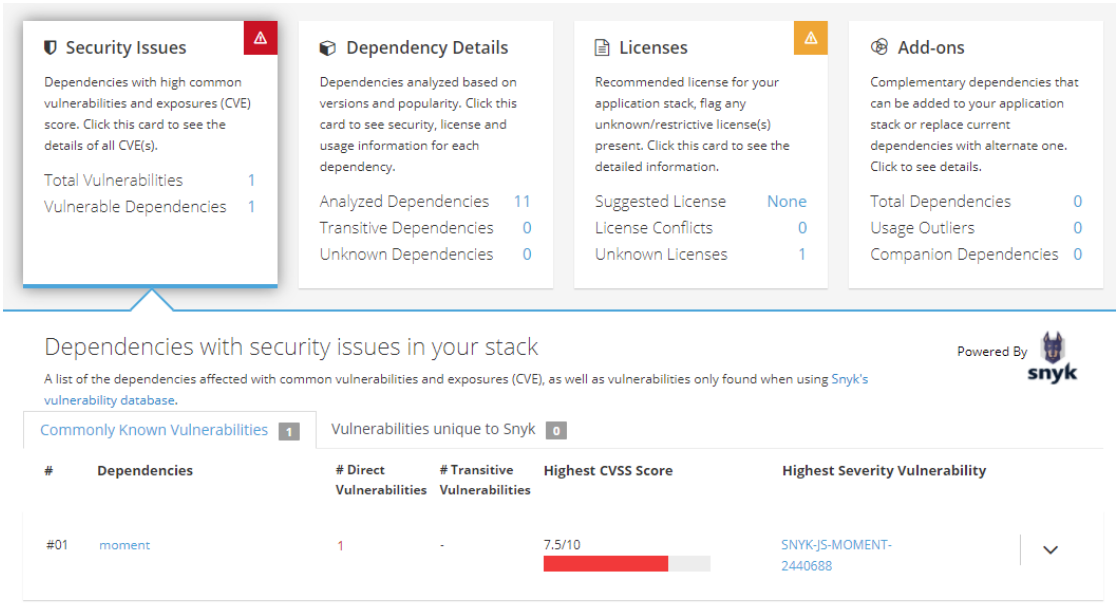


Figure 4.8: The overview of the unoptimized build version with the security findings in Snyk.

The Snyk analyses results from the optimized build versions are presented in Figure 4.10. The moment dependency issue has been resolved via a dependency update.

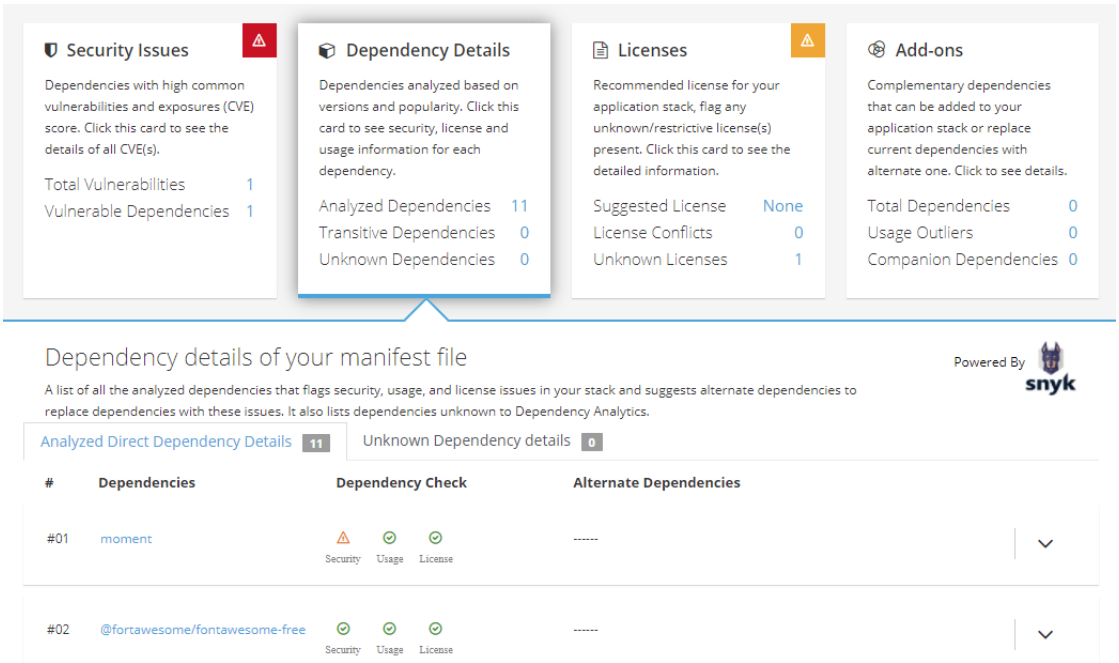


Figure 4.9: The dependency details of the unoptimized build version in Snyk. The top most dependency, moment, is the most critical one, all below satisfy security analyses.

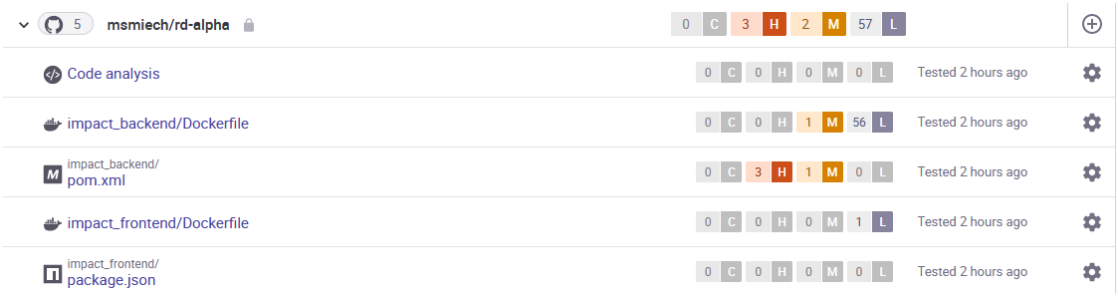


Figure 4.10: The Snyk analysis report overview of the optimized build. There are no critical vulnerabilities.

Combined

The following table illustrates the combined security evaluations and analyses results 4.12.

Tool	Unoptimized #	Optimized #
SonarQube	5	0
Snyk	1	0
OWASP dependency-check (backend)	4	0
npm audit (frontend)	2	0

Table 4.12: Aggregated critical security vulnerability and issue count across the 4 tools.

Overall, there are no critical security vulnerabilities in the optimized build. Thus, this answers research question RQ5 proving the feasibility of an implementation with zero critical security vulnerabilities, as defined in 1.6. The answering of this question involved the methods of developing and testing of a prototype and observations made using the four aforementioned tools to identify and eliminate the critical security vulnerabilities.

In addition, this answers the research question RQ6 in regards to how the level of security can be increased, as defined in 1.6. In short, it is sufficient to apply the suggestions of the respective tools, i.e. Snyk, SonarQube, npm audit and OWASP dependency-check, for example, by upgrading dependencies. The relevant parts of the methodological process to answer RQ6 have been the observations made using the four aforementioned tools as well as their recommendations to eliminate the critical security vulnerabilities.

4.2.3 Network packet analysis

To verify network and transport layer security, the packets from both ends of the data path have been analyzed using the tools:

- WireShark
- Android Studio Network Inspector

WireShark

The network traffic between the web frontend and the backend has been analyzed using WireShark. The inspected packets have been captured and evaluated to be properly TLS 1.2 encrypted. See Figure 4.11.

To verify TLS between the edge device and the Firebase Realtime database, the traffic has been intercepted by using a computer with Wireshark running as mobile Wi-Fi hotspot. The inspected packets are presented in Figure 4.12. The data has been captured during a boxing session and is indeed encrypted using TLSv1.2. As sanity-check, the IP address (34.107.226.223) that has been transmitted to has been looked up using IP WHOIS Lookup and it is indeed registered as one of Google's servers (i.e. Firebase) [DNS].

As an alternative, the Android Studio Network Inspector has been used but this has the limitation that it only supports HttpURLConnection and OkHttp libraries for network

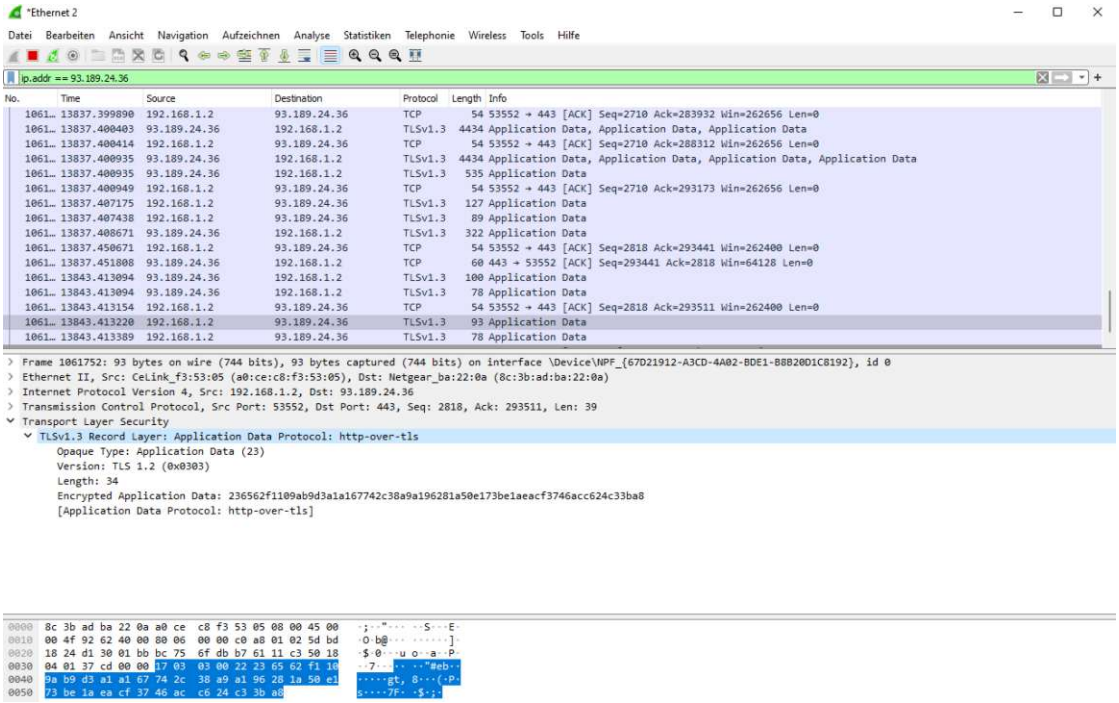


Figure 4.11: A screenshot of WireShark inspecting a TLS 1.2 encrypted packet to the web application.

connections and does not support the Firebase library's connections [LLC]. Therefore it only resulted in "Network inspector data unavailable" which rendered the feature unusable for this evaluation.

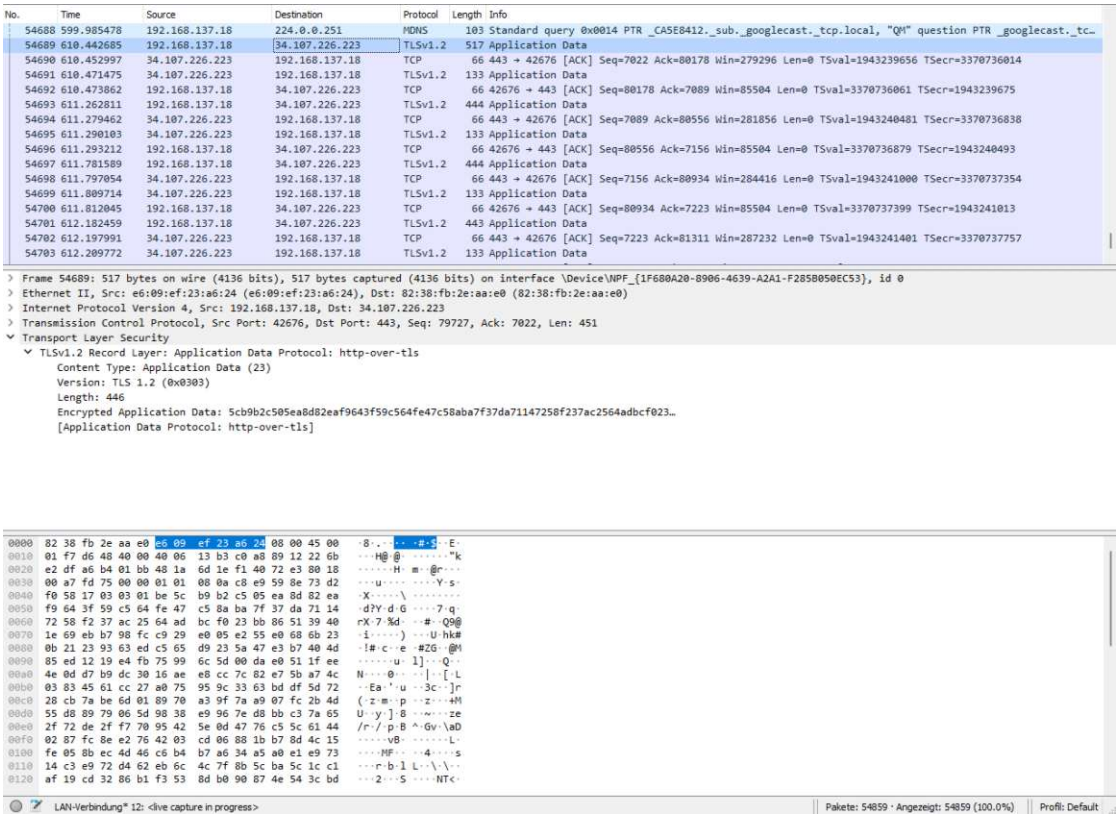


Figure 4.12: A screenshot of WireShark inspecting a TLS 1.2 encrypted packet from the edge device during a boxing session with an overview of other transmitted packets at the top.

CHAPTER 5

Discussion

In the final chapter, a general discussion of the findings and results, in particular the research questions and hypotheses, takes place. Furthermore, the limitations, a conclusion and, finally, an outlook as well as future work are laid out.

5.1 Résumé

Designing, implementing and evaluating a prototype using four laboratory experiments, two prior and two after optimizations, a series of findings has been made in chapters 3 and 4. These clearly demonstrated the feasibility of the proposed system. Furthermore, it also illustrated the significant effects that the optimizations had. How these findings apply to the research questions and hypotheses is discussed and summarized in the following.

Over the course of this work, all six research questions have been answered and both hypotheses have been proven. Each of the research question as well as the hypotheses including their results or proofs, respectively, will be discussed and summarized in this section in chronological order.

The first research question, RQ1, was about the functional requirements of the system and has been answered by conducting interviews with domain experts and performing a qualitative content analysis. The findings of that analysis have been used to formulate the functional requirements. These can be briefly summarized as the system's ability to display live or historic punch data sets using selected metrics and statistical features thereof as well as also providing an authentication mechanism. Subsequently, based on these requirements, use cases have been defined and modelled. Furthermore, this has been crucial to design and structure the application around these requirements, including its activities and user interfaces.

The second research question, RQ2, was concerned with the feasibility of a system fulfilling the requirements while utilizing the Smart Glove prototypes acting as IoT sensors and

using state-of-the-art components. The implementation presented in Chapter 3 which also served as baseline for Chapter 4 already proves that feasibility, albeit not reaching the desired performance target of an average latency of less than 300 milliseconds and security goal of having no known critical security vulnerabilities. Building on top of that, the optimized version reaches the performance and security targets.

The third research question, RQ3, focused on the feasibility of the performance target of less than 300 milliseconds average latency from edge device to web frontend in the system. It has been answered with the help of the optimizations implemented in Chapter 4, reaching a latency of 207.57 milliseconds on a Wi-Fi connection and 262.19 milliseconds on a cellular connection. Further possible optimization attempts are touched upon in the limitations 5.2 and outlook 5.4.

The fourth research question, RQ4, was about the possible performance optimizations and their significance. It has been answered in Chapter 4 using the described performance improvements such as removal of debug/log functionality, optimizing sorting/mapping algorithms and reducing the number of network requests through changes of data structures. The measured improvements have been successfully tested for statistical significance.

Based on the findings of the third and fourth research questions, RQ3 and RQ4, respectively, the results of the prototype's optimized builds using either cellular (262.19 milliseconds) or a Wi-Fi connection (207.57 milliseconds) confirm hypothesis H1, an average latency of less than 300 milliseconds. This has been achieved by applying the methods described in Chapter 2, in particular the evaluation of the performance measurements made based on the laboratory experiments and subsequent performance optimizations and quantitative analysis thereof including statistical significance tests.

For the sake of comparison of the H1 result, reference systems ranged between 300 milliseconds and several seconds, as analyzed in 3.2.2. Unfortunately, none of the works of the reference systems published detailed statistics on the performance metrics that have been achieved using these systems and only specify performance targets or recommendations which is summarized in the following. The pitch tracking system for television broadcasts of baseball games by Gueziec had a performance target of less than 2 seconds [Gue02]. The 3D tracking system for live sports broadcasts by Cavallero et al. [CHWB11] had a target latency of less than 1 second. The internet application delay recommendations specified by Suznjevic et al. [SS16] specify a set of target latencies depending on the application type, the lowest one regarding a broadcasting type of application was 400 milliseconds. Overall, the lowest performance reference value has been found in the work of Willebeek-LeMair et al. [WLS95] with a target latency of less than 300 milliseconds for their video conferencing application. The average performance measurements by this work have been well below 300 milliseconds and in the case of Wi-Fi connectivity closer to 200 milliseconds with an average value of 207.57 milliseconds.

The fifth research question, RQ5, dealt with the security target of zero known critical security vulnerabilities detected in the system using the four established tools. It has been answered through the security optimizations based on the recommendations issued

through the security tools, particularly dependency updates as recommended by the SCA tools and security configuration tightening as recommended by the SAST tools as detailed in Chapter 4. It has been found that these tools, SonarQube, OWASP Dependency-Check, Snyk and npm audit, provide helpful recommendations to mitigate the security concerns such as previously mentioned. These addressed OWASP top ten entries as shown in 1.2.5, such as security misconfiguration (A05), vulnerable and outdated components (A06) and security logging and monitoring failures (A09) from the 2021 list.

The sixth and final research question, RQ6, had a focus on security enhancements and optimizations. These have been achieved by applying the recommendations as described at the previous research question, RQ5, provided by the four used security tools which helped to reach the desired number of zero detected known critical vulnerabilities and thus answering the question.

Through the answering of the fifth and sixth research questions (RQ5 and RQ6), where implementing the recommendations issued by the security tools lead from an unoptimized baseline version to an optimized version reaching the security target of eliminating all critical security vulnerabilities, thus reducing them to zero. Hence, this confirms the second hypothesis, H2. The path to this result ranged through all the methods described in Chapter 2, particularly the security evaluation of the observations made using the four state-of-the-art security tools in Chapter 4 and subsequent mitigation of security concerns therein.

To put the H2 result in perspective, reference software projects frequently have critical security vulnerabilities as listed by the examples in 1.3.1 [Sur18]. Varying over time, popular software projects such as Apache HTTP Server had 206 CVE entries and Apache Tomcat had 131 CVE entries (not distinguished by severity) in 2019 based on the NIST and MITRE CVE [MIT] databases according to Piantadosi et al. [PSO19]. The number of CVE entries in this work summed up across the four tools is 63, where 62 originated from Snyk, 57 of which are low severity, and only one with high severity from the OWASP dependency-check (being transitive through Spring Boot's Apache Tomcat dependency) as can be seen in detail in 4.2. When comparing the SonarQube results, projects such as Apache CXF had 89 vulnerabilities, Apache Gora had 6 and Apache Groovy had 204, as mentioned in 1.3.1, whereas this work has 0 after optimizations.

In addition to the six research questions, both hypotheses, H1 and H2 have been therefore proven. The combination of the performance as well as security aspects provide a satisfying conclusion and integration of the three tenets of the CIA triad, confidentiality, integrity and availability, as introduced in 1.2.5, where availability described timely and efficient usage of the system while incorporating an adequate level of security.

5.2 Limitations

A limiting factor is the Firebase constraint in itself - which imposes a communication method between the edge device and the backend service, namely the Firebase Realtime

Database. Due to that limitation, the options for optimization of the message delivery between edge device and backend infrastructure are limited and cannot be tailored and tuned directly. It is subject to the vendor's implementation - the optimization of which has been addressed by updating the dependencies of that vendor. This extends to the applicability of tools as observed with the Android Studio Network Inspector which was unable to analyze the network packets between the edge device and the Firebase Realtime Database and WireShark had to be used as a workaround.

Another limiting aspect is the distributed and heterogeneous nature of the software architecture, network topology in conjunction with the environmental constraints. This had impact on the process of performance measurement as common tools such as Apache JMeter [Foua] could not be applied properly without the need for substantial adjustments. For example, the edge device is running a custom Android application for collecting, preprocessing and further transmitting the sensor data which could not have been easily integrated into a JMeter test plan to measure its performance. This would have required additional hooks/APIs on the edge device or configuring it to use JMeter as a proxy which also interferes with the way the Firebase library operates in the application on the device.

5.3 Conclusion

The software design utilizing the hardware shows the feasibility of the use cases. These include viewing and comparing of athletes' performances using live or historic data as well as visualizations of that data using a set of different metrics (impact, mean velocity, peak acceleration, peak force, peak velocity). Another conclusion is that the Smart Glove prototypes and Android-based edge devices are sufficient for the use cases. Moreover, the performance and security goals have been reached. Beyond the sixth research question, RQ6, and the second hypothesis, H2, the implementation of the prototype has satisfied the security requirements by providing TLS as well as using an authentication process and authorization mechanisms on the backend, frontend and the edge devices. These security measures help to prevent intrusions and unauthorized access, making it harder to cheat at events such as competitions and tamper with or manipulate the results and thus improving the integrity thereof.

A particular observation in regards to the performance aspect is that a Wi-Fi based connection has significantly lower latency than a cellular based connection. Hence, to get the overall best results with the prototype, a Wi-Fi connection with decent signal quality and as little wireless interference as possible as well as using the optimized build is recommended as the best environment for the system.

Conclusively, the system is suitable for comparison and live feedback on the athletes' performances using the Smart Glove prototypes. All six research questions have been answered and both of the hypotheses have been proven during the course of this work.

5.4 Outlook

For future work, this system could be integrated into a live scoring system used at official competitions such as championships like the Olympic games. It could be integrated in a referee support system and also support automated scoring during competitions as described in 1.2.1. The data gathered and processed could also be integrated as an overlay in a broadcast video stream or composited with a video.

Moreover, third party systems from the fitness or medical domain could be integrated. As explained in 1.2.1, there is still a significant number of injuries reported at professional boxing using gloves, hence this could not only improve well-being of the athletes but also reduce the risk of injury. This opens up possibilities for new use cases that can incorporate health considerations and can involve the athletes themselves, participating organizers or perhaps even authorities. For example, the movement and measured impact characteristics could be analyzed to improve the technique to avoid injuries. Certain thresholds could be defined in the system to issue warnings when reached so users can be warned of potential harm. Dedicated software and hardware (such as sensor-equipped chest straps) could be created to extend the functionality of the proposed system itself towards such use cases.

Functionality could be extended by providing features for historic data, like search/filtering and also aggregates of multiple past sessions. Such aggregation results could again be visualized using appropriate or even interactive visualizations. The past data could also be paginated and lazy loaded dynamically when scrolling through it. Another feature that is possible is generating reports for sessions or users and making them exportable in a common format. Additionally, such previous records could be incorporated to ease the admission process for events.

The web frontend could completely incorporate the SvelteKit once there is a stable release. Also, the application of prerendering and server-side-rendering (SSR) could be explored for it [Lib22]. This could offload some work from the frontend to a backend server and enhance distribution.

To further improve results, more efficient and tailored edge devices could be applied. This would enable overall system design enhancements and reduce technical overhead. In particular, CoAP could be utilized to expose data and facilitate machine-to-machine communication which would be more efficient than HTTP based communication. Moreover, dedicated edge device hardware or common IoT hardware solutions such as Arduinos or Raspberry Pis could be considered [LYL+18].

With the proposed microservice-based cloud architecture, the problem of network congestion comes up. Where needed, the access to HTTP-based REST APIs can be profiled and subsequently optimized using solutions such as load balancing through a reverse proxy, for example [CLNW12].

Public cloud solutions can be used either complementary or as basis to operate and scale the system up or down according to demand, for example, in a Container-as-a-Service

(CaaS) capacity [CVRM+10]. As operating and managing such a scalable distributed system is challenging, tools that provide support and insight can be applied such as Prometheus for monitoring and logging as well as Grafana for visualization of the quality of the system performance [IHK18]. This concludes the extensive outlook and possible future work and marks the end of this thesis.

List of Figures

List of Tables

Bibliography

[AAE16] N. Alshuqayran, N. Ali, and R. Evans. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 44–51, 2016.

[Ado] AdoptOpenJDK. Latest release | adoptopenjdk - open source, prebuilt openjdk binaries. `https://adoptopenjdk.net/releases.html`. Accessed: 2021-02-17 11:05:02.

[Ala14] Mohammed M. Alani. *OSI Model*, pages 5–17. Springer International Publishing, Cham, 2014.

[Alb] Plebani Alberto. Web development comparison: Spring boot vs. express.js - dzone web dev. `https://dzone.com/articles/web-development-comparison-springboot-vs-expressjs`. Accessed: 2020-12-23 03:36:18.

[Ana21] Oxford Analytica. Apache software flaw could result in major breaches. *Emerald Expert Briefings*, 2021.

[Asi] Move It Asia. Move it - swift - smart boxing gloves (12oz). `https://move-it.store/products/move-it-swift-smart-boxing-gloves-12oz`. Accessed: 2022-09-17 10:55:22.

[BCK03] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003.

[BCS12] C. Bormann, A. P. Castellani, and Z. Shelby. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2):62–67, 2012.

[BDK16] A. Biryukov, D. Dinu, and D. Khovratovich. Argon2: New generation of memory-hard functions for password hashing and other applications. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 292–302, 2016.

[BGDI16] SL Bangare, S Gupta, M Dalal, and A Inamdar. Using node. js to build high speed and scalable backend database server. In *Proc. NCPCI. Conf*, page 19, 2016.

[Bin00] Robert Binder. *Testing object-oriented systems: models, patterns, and tools*. Addison-Wesley Professional, 2000.

[BK07] M. Bächle and P. Kirchberg. Ruby on rails. *IEEE Software*, 24(6):105–108, 2007.

[Boj14] Irena Bojanova. It enhances football at world cup 2014. *IT Professional*, 16(4):12–17, 2014.

[Bor06] Jürgen Bortz. *Statistik für Human-und Sozialwissenschaftler*. Springer Berlin, 09 2006.

[Bos] Mike Bostock. D3.js - data-driven documents. `https://d3js.org/`. Accessed: 2022-03-22 09:13:26.

[Bou] Petr Bouda. Pros and cons for using graalvm native-images - dzone java. `https://dzone.com/articles/profiling-native-images-in-java`. Accessed: 2020-12-23 11:34:34.

[BP20] Krzysztof Boczkowski and Beata Pańczyk. Comparison of the performance of tools for creating a spa application interface - react and vue.js. *Journal of Computer Sciences Institute*, 14:73–77, Mar. 2020.

[Bria] Britannica. Boxing. `https://www.britannica.com/sports/boxing`. Accessed: 2022-09-03 10:01:24.

[Brib] Britannica. Modern wrestling. `https://www.britannica.com/sports/wrestling/Modern-wrestling`. Accessed: 2022-09-03 06:19:05.

[BxR] BxRank.com. What is definition of martial arts | know everything about it. `https://bxrank.com/blog/article/definition-of-martial-arts-traditional-martial-art`. Accessed: 2022-09-01 03:19:25.

[CAAO18] Samir Chadli, Noureddine Ababou, Amina Ababou, and Nazim Ouadahi. Quantification of boxing gloves damping: Method and apparatus. *Measurement*, 129:504–517, 2018.

[Che19] Boris Cherny. *Programming TypeScript: Making Your JavaScript Applications Scale*. O'Reilly Media, 2019.

[Cho15] Varun Chopra. *WebSocket Essentials–Building Apps with HTML5 WebSockets*. Packt Publishing Ltd, 2015.

[CHWB11] Rick Cavallaro, Maria Hybinette, Marvin White, and Tucker Balch. Augmenting live broadcast sports with 3d tracking information. *IEEE MultiMedia*, 18(4):38, 2011.

[CLNW12] X. Chi, B. Liu, Q. Niu, and Q. Wu. Web load balance and cache optimization design based nginx under high-concurrency environment. In *2012 Third International Conference on Digital Manufacturing Automation*, pages 1029–1032, 2012.

[CN19] Dariusz Chęć and Ziemowit Nowak. The performance analysis of web applications based on virtual dom and reactive user interfaces. In Piotr Kosiuczenko and Zbigniew Zieliński, editors, *Engineering Software Systems: Research and Praxis*, pages 119–134, Cham, 2019. Springer International Publishing.

[CNMR18] Karina Curcio, Tiago Navarro, Andreia Malucelli, and Sheila Reinehr. Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, 139:32–50, 2018.

[CNYM12] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. International Series in Software Engineering. Springer US, 2012.

[Cod] Codecademy. React: The virtual dom. `https://www.codecademy.com/articles/react-virtual-dom`. Accessed: 2021-01-03 13:05:52.

[Cora] Oracle Corporation. Openjdk. `https://openjdk.java.net/`. Accessed: 2021-05-06 01:00:23.

[Corb] Striketec Corporation. Striketec. `https://striketec.com`. Accessed: 2021-01-08 10:29:49.

[CTKT13] Nergiz Ercil Cagiltay, Gul Tokdemir, Ozkan Kilic, and Damla Topalli. Performing and analyzing non-formal inspections of entity relationship diagram (erd). *Journal of Systems and Software*, 86(8):2184–2195, 2013.

[CVRM+10] Juan Cáceres, Luis M. Vaquero, Luis Rodero-Merino, Álvaro Polo, and Juan J. Hierro. *Service Scalability Over the Cloud*, pages 357–377. Springer US, Boston, MA, 2010.

[Dad14] Majid Dadafshar. Accelerometer and gyroscopes sensors: operation, sensing, and applications. *Maxim Integrated [online]*, 2014.

[Dai] Shaumik Daityari. Angular vs react vs vue: Which framework to choose in 2021. `https://www.codeinwp.com/blog/angular-vs-vue-vs-react/`. Accessed: 2020-12-24 01:14:07.

[DJPJP+] Philipp Dunkel, Maggie Johnson-Pint, Matt Johnson-Pint, Brian Terlson, Shane Carr, Ujjwal Sharma, Philip Chimento, Jason Williams, and Justin Grant. Temporal. https://tc39.es/proposal-temporal/docs/index.html. Accessed: 2022-02-22 08:44:58.

[dM] Curiouser Products Inc. (dba MIRROR). Mirror boxing workout classes. https://www.mirror.co/classes/boxing. Accessed: 2022-03-31 05:00:07.

[DMM+13] A. De Rubertis, L. Mainetti, V. Mighali, L. Patrono, I. Sergi, M. L. Stefanizzi, and S. Pascali. Performance evaluation of end-to-end security protocols in an internet of things. In *2013 21st International Conference on Software, Telecommunications and Computer Networks - (SoftCOM 2013)*, pages 1–6, 2013.

[DNS] DNSChecker.org. Ip whois lookup - lookup an ip address - dns checker. https://dnschecker.org/ip-whois-lookup.php?query=34.107.226.223. Accessed: 2022-05-07 12:29:21.

[DR08] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. *RFC*, 2008.

[DR19] Helder Da Rocha. *Learn Chart. js: Create interactive visualizations for the Web with Chart. js 2*. Packt Publishing Ltd, 2019.

[Dul] Chameera Dulanga. Incremental vs virtual dom. https://blog.bitsrc.io/incremental-vs-virtual-dom-eb7157e43dca. Accessed: 2021-01-04 04:55:24.

[EW] Inc. Everlast Worldwide. Everlast piq. https://www.everlastboxing.com.au/piq-x-everlast-boxing-tracker.html. Accessed: 2022-03-31 19:07:18.

[Fac] Facebook. React - versions. https://reactjs.org/versions/. Accessed: 2021-01-05 02:29:26.

[Fau19] Sebastian Faust. Rest - why it became the new norm on the web, 2019.

[FC10] Daniel Tik-Pui Fong and Yue-Yan Chan. The use of wearable inertial motion sensors in human lower limb biomechanics studies: A systematic review. *Sensors*, 10(12):11556–11565, 2010.

[Fen08] Kim Fenrich. Securing your control system: the" cia triad" is a widely used benchmark for evaluating information system security effectiveness. *Power Engineering*, 112(2):44–49, 2008.

[Fit] SLLR Enterprises LLC | Rockbox Fitness. Boxing classes | kickboxing workout | rockbox fitness class. https://www.rockboxfitness.com/. Accessed: 2022-04-01 16:53:55.

[Foua] Apache Foundation. Apache jmeter. https://jmeter.apache.org/. Accessed: 2021-01-23 05:37:30.

[Foub] OWASP Foundation. Owasp dependency-check. https://owasp.org/www-project-dependency-check/. Accessed: 2022-04-09 01:56:05.

[Fouc] OWASP Foundation. Owasp top ten. https://owasp.org/www-project-top-ten/. Accessed: 2022-08-28 03:32:27.

[Foud] The Apache Software Foundation. Apache maven. https://maven.apache.org/. Accessed: 2022-04-09 15:48:02.

[Foue] The Apache Software Foundation. The apache software foundation. https://www.apache.org/. Accessed: 2022-03-07 12:15:18.

[Fow] Martin Fowler. Microservices. https://martinfowler.com/articles/microservices.html. Accessed: 2020-12-23 11:47:48.

[FPD12] G. Fortino, M. Pathan, and G. Di Fatta. Bodycloud: Integration of cloud computing and body sensor networks. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pages 851–856, 2012.

[FSF09] Xinyang Feng, Jianjing Shen, and Ying Fan. Rest: An alternative to rpc for web services architecture. In *2009 First International Conference on Future Information Networks*, pages 7–10. IEEE, 2009.

[GB18] Bath Graham and Rex Black. Foundation level specialist syllabus performance testing, 2018.

[GbR] CxC GbR. Complexcoach. https://www.cxc.complexcoach.com. Accessed: 2022-04-01 01:52:03.

[GDS+] Oliver Gierke, Thomas Darimont, Christoph Strobl, Mark Paluch, and Jay Bryant. Spring data jpa - reference documentation. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/. Accessed: 2022-05-06 06:47:34.

[gF] gymGO Fitness. Personal trainer software | gymgo. https://gymgo.com/. Accessed: 2022-04-01 09:37:39.

[GFN+17] Paul A Grassi, James L Fenton, EM Newton, RA Perlner, AR Regenscheid, WE Burr, JP Richer, NB Lefkovitz, JM Danker, Yee-Yin Choong, et al. Nist special publication 800-63b: digital identity guidelines. *National Institute of Standards and Technology (NIST)*, 2017.

[Gmba] Fitogram GmbH. Fitogrampro. `https://www.fitogram.pro`. Accessed: 2022-04-01 11:32:51.

[Gmbb] Freeletics GmbH. Intensive workouts & individual training plans | freeletics. `https://www.freeletics.com`. Accessed: 2022-04-01 03:03:45.

[Gmbc] Myra Security GmbH. Global content delivery network – myra security. `https://www.myrasecurity.com/en/product-content-delivery-network/global-cdn/`. Accessed: 2021-04-14 11:24:49.

[Gmbd] ROOQ GmbH. Rooq boxing technology. `https://rooq.de`. Accessed: 2022-03-31 19:24:49.

[Gmbe] SimpliFlow International GmbH. Training software solutions - simpliflow. `https://www.simpliflow.com/homepage/de/`. Accessed: 2022-04-01 11:13:17.

[God17] Michelle Goddard. The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

[Gooa] Google. Angular. `https://angular.io/guide/releases`. Accessed: 2020-12-05 07:46:19.

[Goob] Google. Chrome releases: 2021. `https://chromereleases.googleblog.com/2021/`. Accessed: 2021-02-17 10:07:18.

[Gooc] Google. Core concepts, architecture and lifecycle | grpc. `https://grpc.io/docs/what-is-grpc/core-concepts`. Accessed: 2021-02-23 09:35:38.

[Good] Google. Firebase. `https://firebase.google.com/`. Accessed: 2021-01-18 11:58:05.

[GPC20] Paul S. Ganney, Sandhya Pisharody, and Edwin Claridge. Chapter 9 - software engineering. In Azzam Taktak, Paul S. Ganney, David Long, and Richard G. Axell, editors, *Clinical Engineering (Second Edition)*, pages 131–168. Academic Press, second edition edition, 2020.

[GPP15] Katerina Goseva-Popstojanova and Andrei Perhinschi. On the capability of static code analysis to detect security vulnerabilities. *Information and Software Technology*, 68:18 – 33, 2015.

[Groa] Equinox Group. Equinox luxury fitness club. https://www.equinox.com/. Accessed: 2022-04-01 03:00:01.

[Grob] Leap Fitness Group. Leap fitness-millions of users' choice. https://leap.app/. Accessed: 2022-04-01 03:15:18.

[Groc] The Open Group. Remote procedure call - introduction to the rpc specification. https://pubs.opengroup.org/onlinepubs/9629399/chap1.htm. Accessed: 2022-04-04 11:05:43.

[Grod] The PostgreSQL Global Development Group. Postgresql jdbc driver. https://jdbc.postgresql.org/. Accessed: 2022-05-06 20:53:47.

[GS10] T.A. Green and J.R. Svinth. *Martial Arts of the World: An Encyclopedia of History and Innovation*. Number Bd. 2 in Martial Arts of the World: An Encyclopedia of History and Innovation. ABC-CLIO, 2010.

[Gue02] A. Gueziec. Tracking pitches for broadcast television. *Computer*, 35(3):38–43, 2002.

[HBC21] D. Hölbling, R. Breiteneder, and L. Christoph. System zur automatisierten Wertungsvergabe bei Kampfsportarten, 2021. Registration ID: A 50619/2021.

[HHK+10] A.G. Hahn, R.J.N. Helmer, T. Kelly, K. Partridge, A. Krajewski, I. Blanchonette, J. Barker, H. Bruch, M. Brydon, N. Hooke, and B. Andreass. Development of an automated scoring system for amateur boxing. *Procedia Engineering*, 2(2):3095–3101, 2010. The Engineering of Sport 8 - Engineering Emotion.

[HL95] Walter L. Hürsch and Cristina Videira Lopes. Separation of concerns. Technical report, College of Computer Science, Northeastern University, 1995.

[IHK18] Ahmed A. Ismail, Haitham S. Hamza, and Amira M. Kotb. Performance evaluation of open source iot platforms. In *2018 IEEE Global Conference on Internet of Things (GCIoT)*, pages 1–5, 2018.

[IL] IOLA and Ole Laursen. Flot github release v0.8.3. https://github.com/flot/flot/releases/tag/v0.8.3. Accessed: 2022-03-22 09:09:39.

[Inca] Crunchbase Inc. web4trainer. https://www.crunchbase.com/organization/web4trainer. Accessed: 2022-04-01 12:13:37.

[Incb] Hykso Inc. Fightcamp | interactive at-home boxing workouts & equipment. https://joinfightcamp.com/. Accessed: 2021-01-08 10:26:12.

[Incc] Hykso Inc. Fightcamp | interactive at-home boxing workouts & equipment. https://blog.joinfightcamp.com/for-members/fightcamp-app-update-introducing-version-3-0/. Accessed: 2022-09-16 13:45:12.

[Incd] Tempo Interactive Inc. The award-winning ai-powered home gym | tempo. https://tempo.fit/. Accessed: 2022-03-31 19:41:08.

[It] Move It. Move it swift: Smart boxing gloves | indiegogo. https://www.indiegogo.com/projects/move-it-swift-smart-boxing-gloves. Accessed: 2021-01-05 12:14:04.

[ITW21] Nasif Imtiaz, Seaver Thorn, and Laurie Williams. A comparative study of vulnerability reporting by software composition analysis tools. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11, 2021.

[Jac] Robbie Jack. #1 personal trainer software - truecoach. https://truecoach.co/. Accessed: 2022-04-01 09:13:35.

[JBS] M. Jones, J. Bradley, and N. Sakimura. Rfc 7519 - json web token (jwt). https://tools.ietf.org/html/rfc7519. Accessed: 2021-03-01 09:53:25.

[Kam10] Channu Kambalyal. 3-tier architecture. *Retrieved On*, 2:34, 2010.

[Kan] Meiyappan Kannappa. Data streaming via grpc vs mqtt vs websockets | by meiyappan kannappa | medium. https://msmechatronics.medium.com/data-streaming-via-grpc-vs-mqtt-vs-5c30dd205193. Accessed: 2021-02-23 11:21:31.

[KCS20] Stefan Karlsson, Adnan Causevic, and Daniel Sundmark. Quickrest: Property-based test generation of openapi-described restful apis. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 131–141, 2020.

[Kim15] Tae Kyun Kim. T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540, 2015.

[Kin] Kinsta. Global desktop browser market share for 2021. https://kinsta.com/browser-market-share/. Accessed: 2021-02-17 10:05:41.

[Kra] Stefan Krause. Github - krausest/js-framework-benchmark: A comparison of the performance of a few popular javascript frameworks. https://github.com/krausest/js-framework-benchmark. Accessed: 2021-02-17 11:35:31.

[KS18] Sandor Kiraly and Szilveszter Szekely. Analysing rpc and testing the performance of solutions. *Informatica*, 42(4), 2018.

[KSG04] Mark Kalman, Eckehard Steinbach, and Bernd Girod. Adaptive media playout for low-delay video streaming over error-prone channels. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):841–851, 2004.

[Laba] National Physical Laboratory. What is the si unit of force? `https://www.npl.co.uk/resources/q-a/what-is-the-si-unit-of-force`. Accessed: 2022-09-19 05:58:17.

[Labb] Tailwind Labs. Github - tailwindlabs/tailwindcss: A utility-first css framework for rapid ui development. `https://github.com/tailwindlabs/tailwindcss`. Accessed: 2022-02-05 04:31:58.

[Lib] Free Icons Library. Punching icon #108256. `https://icon-library.com/icon/punching-icon-15.html`. Accessed: 2022-03-17 21:54:32.

[Lib22] Alex Libby. *Practical Svelte: Create Performant Applications with the Svelte Component Framework*. Apress, Berkeley, CA, 2022.

[Lig17] Roger A Light. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265, 2017.

[Lima] Les Mills International Limited. Taking fitness to the next level | les mills. `https://www.lesmills.com`. Accessed: 2022-04-01 02:34:00.

[Limb] Snyk Limited. Snyk open source. `https://snyk.io/product/open-source-security-management/`. Accessed: 2022-04-09 15:48:02.

[Liu11] Henry H Liu. *Software performance and scalability: a quantitative approach*. John Wiley & Sons, 2011.

[LJD+14] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu. When https meets cdn: A case of authentication in delegated service. In *2014 IEEE Symposium on Security and Privacy*, pages 67–82, 2014.

[LKH+02] Dario G. Liebermann, Larry Katz, Mike D. Hughes, Roger M. Bartlett, Jim McClements, and Ian M. Franks. Advances in the application of information technology to sport performance. *Journal of Sports Sciences*, 20(10):755–769, 2002. PMID: 12363293.

[LLC] Google LLC. Android studio network inspector. `https://developer.android.com/studio/debug/network-profiler`. Accessed: 2022-04-14 07:06:10.

[Ltda] Eggplant Technologies Ltd. Move it swift - android app. https://play.google.com/store/apps/details?id=com.spower.moveitswift. Accessed: 2022-09-17 11:08:21.

[Ltdb] Eggplant Technologies Ltd. Move it swift - ios app. https://apps.apple.com/us/app/move-it-swift/id1515225140. Accessed: 2022-09-17 09:08:40.

[Ltdc] Fitii Ltd. All-in-one personal training software | my pt hub. https://www.mypthub.net/. Accessed: 2022-04-01 10:49:54.

[Ltdd] Myzone Ltd. Group fitness tracking software | wearable fitness trackers. https://www.myzone.org/. Accessed: 2021-01-05 01:11:27.

[Ltde] Xiaomi Singapore Pte. Ltd. Mi band. https://www.mi.com/global/miband. Accessed: 2022-04-01 03:16:52.

[Lux] Luxon. luxon - immutable date wrapper. https://moment.github.io/luxon. Accessed: 2022-02-02 08:53:53.

[LYK+11] M. Li, F. Ye, M. Kim, H. Chen, and H. Lei. A scalable and elastic publish/subscribe service. In *2011 IEEE International Parallel Distributed Processing Symposium*, pages 1254–1265, 2011.

[LYL+18] W. Li, C. Yen, Y. Lin, S. Tung, and S. Huang. Justiot internet of things based on the firebase real-time database. In *2018 IEEE International Conference on Smart Manufacturing, Industrial Logistics Engineering (SMILE)*, pages 43–47, 2018.

[Mac] Gym Lead Machine. Technical motivation. https://cfpfit.com/technical-motivation/. Accessed: 2022-09-18 17:32:08.

[Mar97] Robert C Martin. Java and c++ a critical comparison. *Technical Note, Object Mentor*, 1997.

[Mas15] Robert T Mason. Nosql databases and data modeling techniques for a document-oriented nosql database. In *Proceedings of Informing Science & IT Education Conference (InSITE)*, volume 3, pages 259–268, 2015.

[May16] Philipp Mayring. *Einführung in die qualitative Sozialforschung*. Beltz, 2016.

[McN] Mary McNulty. Punching bag. https://www.encyclopedia.com/sports-and-everyday-life/games/games-and-hobbies/punching-bag. Accessed: 2022-09-03 10:09:26.

[MFHI] LLC MW Franchise Holdings International. Group fitness & boxing workouts | mayweather boxing + fitness. https://mayweather.fit/. Accessed: 2022-04-01 02:57:51.

[MIT] MITRE. Mitre cve. https://www.cve.org/. Accessed: 2022-09-24 08:29:43.

[Moza] Mozilla. Javascript | mdn. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed: 2020-12-16 03:58:37.

[Mozb] Mozilla. Javascript language resources - javascript | mdn. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources. Accessed: 2021-03-07 08:31:39.

[MTBHBH+20] Francesc Mateo Tudela, Juan-Ramón Bermejo Higuera, Javier Bermejo Higuera, Juan-Antonio Sicilia Montalvo, and Michael I Argyros. On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications. *Applied Sciences*, 10(24):9119, 2020.

[MTSA19] Antonios Makris, Konstantinos Tserpes, Giannis Spiliopoulos, and Dimosthenis Anagnostopoulos. Performance evaluation of mongodb and postgresql for spatio-temporal data. In *EDBT/ICDT Workshops*, 2019.

[Mur] Nathan Murthy. Rest, rpc, and brokered messaging | by nathan murthy | medium. https://medium.com/@natemurthy/rest-rpc-and-brokered-messaging-b775aeb0db3. Accessed: 2021-02-23 11:00:30.

[Nas] Condé Nast. Ergatta rower review: Turn your workouts into a game | wired. https://www.wired.com/review/ergatta-rower/. Accessed: 2022-03-31 05:26:35.

[NB19] Quy Nguyen and Oras F Baker. Applying spring security framework and oauth2 to protect microservice architecture api. *J. Softw.*, 14(6):257–264, 2019.

[Net] Mozilla Developer Network. Web components | mdn. https://developer.mozilla.org/en-US/docs/Web/Web_Components. Accessed: 2020-12-30 03:07:10.

[npm] npm. npm-audit docs. https://docs.npmjs.com/cli/v6/commands/npm-audit. Accessed: 2022-09-09 09:49:56.

[O¨] Sportlyzer OÜ. Player development and team management software | sportlyzer. https://www.sportlyzer.com/en/. Accessed: 2022-04-01 09:43:12.

[oJa] State of JavaScript. The state of javascript 2019: Front end frameworks. https://2019.stateofjs.com/front-end-frameworks/. Accessed: 2020-12-16 03:24:45.

[oJb] State of JavaScript. The state of javascript 2020: Front end frameworks. https://2020.stateofjs.com/front-end-frameworks/. Accessed: 2021-02-17 18:39:27.

[OMGSC18] Tosin Daniel Oyetoyan, Bisera Milosheska, Mari Grini, and Daniela Soares Cruzes. Myths and facts about static application security testing tools: an action research at telenor digital. In *International Conference on Agile Software Development*, pages 86–103. Springer, Cham, 2018.

[oSN] National Institute of Standards and Technology (NIST). Cve-2022-23181 - nvd - results. https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.56. Accessed: 2022-05-06 06:47:34.

[PDVH19] Christina Paule, Thomas F. Dullmann, and Andre Van Hoorn. Vulnerabilities in continuous delivery pipelines? a case study. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 102–108, 2019.

[Pea] Brad Peabody. Server-side i/o: Node vs. php vs. java vs. go | toptal. https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go. Accessed: 2020-12-30 12:49:22.

[Pel] Peloton®. Peloton® | exercise bike with indoor cycling classes streamed live & on-demand. https://www.onepeloton.com/. Accessed: 2021-01-15 08:54:14.

[Per] Ultimate Performance. World leading personal trainers | personal training london & uk. https://ultimateperformance.com/. Accessed: 2022-04-01 02:42:09.

[PG03] Ruben Pinilla and Marisa Gil. Jvm: Platform independent vs. performance dependent. *SIGOPS Oper. Syst. Rev.*, 37(2):44–56, April 2003.

[PGK17] Maxim Polenov, Sergey Gushanskiy, and Artem Kurmaleev. Synthesis of expert system for the distributed storage of models. In *Computer Science On-line Conference*, pages 220–228. Springer, 2017.

[PNELM21] Julien Ponge, Arthur Navarro, Clément Escoffier, and Frédéric Le Mouël. Analysing the performance and costs of reactive programming libraries in java. In *Proceedings of the 8th ACM SIGPLAN International Workshop on Reactive and Event-Based Languages and Systems*, REBLS 2021, page 51–60, New York, NY, USA, 2021. Association for Computing Machinery.

[Poi] Tutorials Point. Ruby on rails - introduction - tutorialspoint. https://www.tutorialspoint.com/ruby-on-rails/rails-introduction.htm. Accessed: 2021-02-17 09:53:06.

[Pri08] Dan Pritchett. Base: An acid alternative: In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability. *Queue*, 6(3):48–55, 2008.

[PS17] Pavel Pokorný and Kamil Stokláska. Chart visualization of large data amount. In Radek Silhavy, Petr Silhavy, Zdenka Prokopova, Roman Senkerik, and Zuzana Kominkova Oplatkova, editors, *Software Engineering Trends and Techniques in Intelligent Systems*, pages 460–468, Cham, 2017. Springer International Publishing.

[PSO19] Valentina Piantadosi, Simone Scalabrino, and Rocco Oliveto. Fixing of security vulnerabilities in open source projects: A case study of apache http server and apache tomcat. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, pages 68–78, 2019.

[PST18] Bartosz Pawłowicz, Mateusz Salach, and Bartosz Trybus. Smart city traffic monitoring system based on 5g cellular network, rfid and machine learning. In *KKIO Software Engineering Conference*, pages 151–165. Springer, 2018.

[PYHZ14] Tom Polk, Jing Yang, Yueqi Hu, and Ye Zhao. Tennivis: Visualization for tennis match analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2339–2348, 2014.

[QPL18] Kai Qian, Reza M. Parizi, and Dan Lo. Owasp risk analysis driven security requirements specification for secure android mobile software development. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–2, 2018.

[Ray15] Partha Pratim Ray. Internet of things for sports (iotsport): An architectural framework for sports and recreational activity. *Proceeding of IEEE EESCO, Vizag*, pages 79–83, 2015.

[Res] Microsoft Research. Github - microsoft/sanddance: Visually explore, understand, and present your data. https://github.com/microsoft/SandDance. Accessed: 2022-06-14 05:47:23.

[rG] runtastic GmbH. adidas runtastic: adidas running & adidas training apps. https://www.runtastic.com. Accessed: 2022-04-01 03:09:12.

[RGI75] D.T. Ross, J.B. Goodenough, and C.A. Irvine. Software engineering: Process, principles, and goals. *Computer*, 8(5):17–27, 1975.

[Rie97] Dirk Riehle. Composite design patterns. *SIGPLAN Not.*, 32(10):218–228, oct 1997.

[S.Aa] SonarSource S.A. Clean code | developer first | sonar. https://sonarsource.com/. Accessed: 2022-04-09 01:25:32.

[S.Ab] SonarSource S.A. Code quality and code security. https://www.sonarqube.org/. Accessed: 2021-01-23 06:55:56.

[S.Ac] SonarSource S.A. Sonarcloud. https://sonarcloud.io. Accessed: 2022-04-09 01:26:38.

[San17] Chris Sanders. *Practical packet analysis: Using Wireshark to solve real-world network problems*. No Starch Press, 2017.

[Sca21] Thomas P Scanlon. Fundamentals of application security testing tools. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA, 2021.

[SCZ+16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[SIG] Bluetooth SIG. Bluetooth technology website. https://www.bluetooth.com/. [Accessed 04-Sep-2022].

[Siz] Sergei Sizov. Benchmarking spring, rails and node.js against mongodb | by sergei sizov | medium. https://medium.com/@sergeisizov/benchmarking-spring-rails-and-express-against-mongodb-c05c0b1bda80. Accessed: 2021-02-17 07:30:29.

[Soc] Svelte Society. Sveltekit documentation. https://kit.svelte.dev/docs. Accessed: 2021-04-12 09:08:22.

[Sona] SonarQube. Metric definitions. https://docs.sonarqube.org/latest/user-guide/metric-definitions/. Accessed: 2022-03-07 12:15:18.

[Sonb] SonarQube. Support for "*.svelte" files. https://community.sonarsource.com/t/support-for-svelte-files/56394. Accessed: 2022-03-07 22:15:18.

[Soua] Facebook Open Source. Introducing hooks – react. https://reactjs.org/docs/hooks-intro.html. Accessed: 2021-02-18 12:08:30.

[Soub] Firebase Open Source. Firebase open source. https://firebaseopensource.com/. Accessed: 2021-01-19 12:03:26.

[Spea] OMA SpecWorks. Lightweight m2m (lwm2m). https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/. Accessed: 2021-04-26 08:18:55.

[Speb] OMA SpecWorks. Lwm2m v1.2 is now available. https://omaspecworks.org/lwm2m-v1-2-is-now-available/. Accessed: 2021-05-06 03:21:54.

[Spr10] Hanshi Chris Spruiell. *The Encyclopedia of Combative Flow: A Mixed Martial Arts Textbook and Dojo Training Manual*. The Encyclopedia of Combative Flow. Greyhound Books, 2010.

[SS16] Mirko Suznjevic and Jose Saldana. Delay limits for real-time services. *IETF draft*, 06 2016.

[Sur18] Pericherla S Suryateja. Threats and vulnerabilities of cloud computing: a review. *International Journal of Computer Sciences and Engineering*, 6(3):297–302, 2018.

[Svea] Svelte. Releases · sveltejs/svelte · github. https://github.com/sveltejs/svelte/releases. Accessed: 2021-02-18 12:40:19.

[Sveb] Svelte. Svelte 3: Rethinking reactivity. https://svelte.dev/blog/svelte-3-rethinking-reactivity. Accessed: 2021-02-17 11:42:02.

[TAC+06] Gang Tan, Andrew W Appel, Srimat Chakradhar, Anand Raghunathan, Srivaths Ravi, and Daniel Wang. Safe java native interface. In *Proceedings of IEEE International Symposium on Secure Software Engineering*, volume 97, page 106. Citeseer, 2006.

[Tan] Ross Tannenbaum. grpc anywhere. https://cloud.redhat.com/blog/grpc-anywhere. Accessed: 2022-04-01 10:22:05.

[Thö15] J. Thönes. Microservices. *IEEE Software*, 32(1):116–116, 2015.

[Tim] Creative Tim. Github - creativetimofficial/notus-svelte: Notus svelte: Free tailwind css ui kit and admin. https://github.com/creativetimofficial/notus-svelte. Accessed: 2022-02-05 04:30:37.

[Tra] Inc. Trayn. Trayn - design high-quality training programs for maximum results. https://trayn.com/. Accessed: 2022-04-01 09:51:04.

[TS] Inc. Tonal Systems. Tonal. https://www.tonal.com. Accessed: 2022-03-31 19:12:31.

[Tur] Martino Turcato. Lightweight m2m or mqtt: Choosing the right iot protocol for your needs. https://www.telit.com/blog/lwm2m-vs-mqtt/. Accessed: 2021-05-06 03:12:30.

[TvdH07] Richard N. Taylor and Andre van der Hoek. Software design and architecture the once and future focus of software engineering. In *Future of Software Engineering (FOSE '07)*, pages 226–243, 2007.

[UA] Inc. Under Armour. Mapmyfitness. https://www.mapmyfitness.com/app. Accessed: 2022-04-01 03:19:06.

[Ver] Brian Vermeer. Spring dominates the java ecosystem with 60% using it for their main applications. https://snyk.io/blog/spring-dominates-the-java-ecosystem-with-60-using-it-for-their-main-applications/. Accessed: 2020-12-30 02:45:47.

[Vir] Virtuagym. Virtuagym | the #1 fitness software for businesses and consumers. https://virtuagym.com/. Accessed: 2022-04-01 09:34:19.

[VMw] Inc. VMware. Spring boot releases - github. https://github.com/spring-projects/spring-boot/releases. Accessed: 2021-04-07 09:10:09.

[VP03] A. Vakali and G. Pallis. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6):68–74, 2003.

[VW05] Jarke J Van Wijk. The value of visualization. In *VIS 05. IEEE Visualization, 2005.*, pages 79–86. IEEE, 2005.

[W3C] W3C. Soap specifications. https://www.w3.org/TR/soap/. Accessed: 2022-04-04 11:04:17.

[Wad] Paridhi Wadhwani. React vs. angular vs. vue.js: Comparing the most popular front-end frameworks. https://www.bacancytechnology.com/blog/react-vs-angular-vs-vue-js. Accessed: 2020-12-23 08:12:44.

[WEST19] Matthew TO Worsey, Hugo G Espinosa, Jonathan B Shepherd, and David V Thiel. Inertial sensors for performance analysis in combat sports: A systematic review. *Sports*, 7(1):28, 2019.

[WEST20] Matthew TO Worsey, Hugo G Espinosa, Jonathan B Shepherd, and David V Thiel. An evaluation of wearable inertial sensor configuration and supervised machine learning models for automatic punch classification in boxing. *IoT*, 1(2):360–381, 2020.

[Wet15] Sixt Wetzler. Martial arts studies as kulturwissenschaft: A possible theoretical framework. *Martial Arts Studies*, 1:20–33, 11 2015.

[WFP07] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *Future of Software Engineering (FOSE '07)*, pages 171–187, 2007.

[WH06] Thomas Wilde and Thomas Hess. Methodenspektrum der wirtschaftsinformatik. Best practice methodologies for business informatics research, 2006.

[WLS95] M.H. Willebeek-LeMair and Z.-Y. Shae. Centralized versus distributed schemes for videoconferencing. In *Proceedings of the Fifth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pages 85–93, 1995.

[WSH+19] Christian Wimmer, Codrut Stancu, Peter Hofer, Vojin Jovanovic, Paul Wögerer, Peter B. Kessler, Oleg Pliss, and Thomas Würthinger. Initialize once, start fast: Application initialization at build time. *Proc. ACM Program. Lang.*, 3(OOPSLA), October 2019.

[WSM13] Vanessa Wang, Frank Salim, and Peter Moskovits. *Using Messaging over WebSocket with STOMP*, pages 85–108. Apress, Berkeley, CA, 2013.

[WTA] Free Wortley, Chris Thompson, and Forrest Allison. Log4shell update: Second log4j vulnerability published (cve-2021-44228 + cve-2021-45046) | lunasec. `https://www.lunasec.io/docs/blog/log4j-zero-day-update-on-cve-2021-45046/`. Accessed: 2022-01-09 11:20:58.

[WY20] Shengjie Wang and Hairong Yan. Design of real-time monitoring platform for internet of things based on cloud platform. In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pages 61–64. IEEE, 2020.

[You] Evan You. Releases · vuejs/vue-next · github. `https://github.com/vuejs/vue-next/releases`. Accessed: 2020-12-16 04:32:36.

[YQC+19] J. Yongguo, L. Qiang, Q. C oel0 hangshuai, S. Jian, and L. Qianqian. Message-oriented middleware: A review. In *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, pages 88–97, 2019.

[Zai] Charles Zaiontz. t test: unequal variances | real statistics using excel. `https://www.real-statistics.com/students-t-distribution/two-sample-t-test-uequal-variances/`. Accessed: 2022-05-06 03:12:55.

[Zam20] Frank Zammetti. *Modern Full-Stack Development: Using TypeScript, React, Node. js, Webpack, and Docker*. Springer, 2020.

[ZHF+18] Michael Zillich, Dominik Hölbling, Tobias Ferner, Andreas Baldinger, and Walter Wohlkinger. Device for detecting the punch quality in contact sports, 2018. Registration ID: WO2020041806A1.

[ZZ93] Donald W Zimmerman and Bruno D Zumbo. Rank transformations and the power of the student t test and welch t'test for non-normal populations with unequal variances. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 47(3):523, 1993.

Appendix

Measurement Data

This section includes the end-to-end performance measurements. Time measurements are in milliseconds.

Date	Punch #	End-to-end time in ms	Connection
19.01.2022	1	959	Cellular
19.01.2022	2	976	Cellular
19.01.2022	3	979	Cellular
19.01.2022	4	985	Cellular
19.01.2022	5	1164	Cellular
19.01.2022	6	2351	Cellular
19.01.2022	7	492	Cellular
19.01.2022	8	229	Cellular
19.01.2022	9	2919	Cellular
19.01.2022	10	2423	Cellular
19.01.2022	11	996	Cellular
19.01.2022	12	526	Cellular
19.01.2022	13	284	Cellular
19.01.2022	14	770	Cellular
19.01.2022	15	380	Cellular
19.01.2022	16	1958	Cellular
19.01.2022	17	1931	Cellular
19.01.2022	18	1161	Cellular
19.01.2022	19	516	Cellular
19.01.2022	20	110	Cellular
19.01.2022	21	1875	Cellular
19.01.2022	22	1430	Cellular
19.01.2022	23	647	Cellular
19.01.2022	24	233	Cellular
19.01.2022	25	178	Cellular
19.01.2022	26	1748	Cellular
19.01.2022	27	1521	Cellular

19.01.2022	28	1346	Cellular
19.01.2022	29	740	Cellular
19.01.2022	30	289	Cellular
19.01.2022	31	224	Cellular
19.01.2022	32	754	Cellular
19.01.2022	33	433	Cellular
19.01.2022	34	106	Cellular
19.01.2022	35	824	Cellular
19.01.2022	36	703	Cellular
19.01.2022	37	279	Cellular
19.01.2022	38	1904	Cellular
19.01.2022	39	1661	Cellular
19.01.2022	40	1533	Cellular
19.01.2022	41	724	Cellular
19.01.2022	42	306	Cellular
19.01.2022	43	252	Cellular
19.01.2022	44	703	Cellular
19.01.2022	45	414	Cellular
19.01.2022	46	1082	Cellular
19.01.2022	47	212	Cellular
19.01.2022	48	670	Cellular
19.01.2022	49	447	Cellular
19.01.2022	50	150	Cellular
19.01.2022	51	1043	Cellular
19.01.2022	52	549	Cellular
19.01.2022	53	308	Cellular
19.01.2022	54	1028	Cellular
19.01.2022	55	1064	Cellular
19.01.2022	56	515	Cellular
19.01.2022	57	271	Cellular
19.01.2022	58	209	Cellular
19.01.2022	59	276	Cellular
19.01.2022	60	1793	Cellular
19.01.2022	61	1474	Cellular
19.01.2022	62	1396	Cellular
19.01.2022	63	792	Cellular
19.01.2022	64	255	Cellular
19.01.2022	65	1486	Cellular
19.01.2022	66	1279	Cellular
19.01.2022	67	703	Cellular
19.01.2022	68	1290	Cellular
19.01.2022	69	1879	Cellular
19.01.2022	70	1192	Cellular

19.01.2022	71	900	Cellular
19.01.2022	72	595	Cellular
19.01.2022	73	372	Cellular
19.01.2022	74	2245	Cellular
19.01.2022	75	2055	Cellular
19.01.2022	76	1888	Cellular
19.01.2022	77	1906	Cellular
19.01.2022	78	1959	Cellular
19.01.2022	79	1913	Cellular
19.01.2022	80	1970	Cellular
19.01.2022	81	2013	Cellular
19.01.2022	82	1986	Cellular
19.01.2022	83	687	Cellular
19.01.2022	84	688	Cellular
19.01.2022	85	608	Cellular
19.01.2022	86	552	Cellular
19.01.2022	87	441	Cellular
19.01.2022	88	303	Cellular
19.01.2022	89	1111	Cellular
19.01.2022	90	983	Cellular
19.01.2022	91	1021	Cellular
19.01.2022	92	419	Cellular
19.01.2022	93	1212	Cellular
19.01.2022	94	963	Cellular
19.01.2022	95	753	Cellular
19.01.2022	96	565	Cellular
19.01.2022	97	564	Cellular
19.01.2022	98	361	Cellular
19.01.2022	99	236	Cellular
19.01.2022	100	1108	Cellular

Table 1: The performance measurement of the unoptimized prototype using a cellular connection.

Date	Punch #	End-to-end time in ms	Connection
19.01.2022	1	754	Wi-Fi
19.01.2022	2	587	Wi-Fi
19.01.2022	3	431	Wi-Fi
19.01.2022	4	221	Wi-Fi
19.01.2022	5	1153	Wi-Fi
19.01.2022	6	1066	Wi-Fi
19.01.2022	7	572	Wi-Fi
19.01.2022	8	531	Wi-Fi

19.01.2022	9	457	Wi-Fi
19.01.2022	10	418	Wi-Fi
19.01.2022	11	423	Wi-Fi
19.01.2022	12	440	Wi-Fi
19.01.2022	13	463	Wi-Fi
19.01.2022	14	497	Wi-Fi
19.01.2022	15	1498	Wi-Fi
19.01.2022	16	346	Wi-Fi
19.01.2022	17	379	Wi-Fi
19.01.2022	18	334	Wi-Fi
19.01.2022	19	325	Wi-Fi
19.01.2022	20	292	Wi-Fi
19.01.2022	21	292	Wi-Fi
19.01.2022	22	1066	Wi-Fi
19.01.2022	23	669	Wi-Fi
19.01.2022	24	325	Wi-Fi
19.01.2022	25	326	Wi-Fi
19.01.2022	26	1946	Wi-Fi
19.01.2022	27	1690	Wi-Fi
19.01.2022	28	1208	Wi-Fi
19.01.2022	29	444	Wi-Fi
19.01.2022	30	1968	Wi-Fi
19.01.2022	31	1931	Wi-Fi
19.01.2022	32	1283	Wi-Fi
19.01.2022	33	866	Wi-Fi
19.01.2022	34	516	Wi-Fi
19.01.2022	35	446	Wi-Fi
19.01.2022	36	489	Wi-Fi
19.01.2022	37	1090	Wi-Fi
19.01.2022	38	989	Wi-Fi
19.01.2022	39	445	Wi-Fi
19.01.2022	40	122	Wi-Fi
19.01.2022	41	1723	Wi-Fi
19.01.2022	42	1465	Wi-Fi
19.01.2022	43	1301	Wi-Fi
19.01.2022	44	1065	Wi-Fi
19.01.2022	45	670	Wi-Fi
19.01.2022	46	348	Wi-Fi
19.01.2022	47	261	Wi-Fi
19.01.2022	48	118	Wi-Fi
19.01.2022	49	513	Wi-Fi
19.01.2022	50	140	Wi-Fi
19.01.2022	51	862	Wi-Fi

19.01.2022	52	365	Wi-Fi
19.01.2022	53	1034	Wi-Fi
19.01.2022	54	640	Wi-Fi
19.01.2022	55	619	Wi-Fi
19.01.2022	56	252	Wi-Fi
19.01.2022	57	864	Wi-Fi
19.01.2022	58	619	Wi-Fi
19.01.2022	59	642	Wi-Fi
19.01.2022	60	174	Wi-Fi
19.01.2022	61	902	Wi-Fi
19.01.2022	62	784	Wi-Fi
19.01.2022	63	474	Wi-Fi
19.01.2022	64	988	Wi-Fi
19.01.2022	65	707	Wi-Fi
19.01.2022	66	394	Wi-Fi
19.01.2022	67	519	Wi-Fi
19.01.2022	68	1019	Wi-Fi
19.01.2022	69	728	Wi-Fi
19.01.2022	70	459	Wi-Fi
19.01.2022	71	596	Wi-Fi
19.01.2022	72	112	Wi-Fi
19.01.2022	73	589	Wi-Fi
19.01.2022	74	363	Wi-Fi
19.01.2022	75	2230	Wi-Fi
19.01.2022	76	994	Wi-Fi
19.01.2022	77	964	Wi-Fi
19.01.2022	78	280	Wi-Fi
19.01.2022	79	784	Wi-Fi
19.01.2022	80	1372	Wi-Fi
19.01.2022	81	881	Wi-Fi
19.01.2022	82	1494	Wi-Fi
19.01.2022	83	1014	Wi-Fi
19.01.2022	84	630	Wi-Fi
19.01.2022	85	1015	Wi-Fi
19.01.2022	86	1025	Wi-Fi
19.01.2022	87	166	Wi-Fi
19.01.2022	88	249	Wi-Fi
19.01.2022	89	398	Wi-Fi
19.01.2022	90	542	Wi-Fi
19.01.2022	91	1045	Wi-Fi
19.01.2022	92	540	Wi-Fi
19.01.2022	93	380	Wi-Fi
19.01.2022	94	452	Wi-Fi

19.01.2022	95	532	Wi-Fi
19.01.2022	96	770	Wi-Fi
19.01.2022	97	697	Wi-Fi
19.01.2022	98	793	Wi-Fi
19.01.2022	99	275	Wi-Fi
19.01.2022	100	983	Wi-Fi

Table 2: The performance measurement of the unoptimized prototype using a Wi-Fi connection.

Date	Punch #	End-to-end time in ms	Connection
20.02.2022	1	238	Cellular
20.02.2022	2	386	Cellular
20.02.2022	3	192	Cellular
20.02.2022	4	240	Cellular
20.02.2022	5	350	Cellular
20.02.2022	6	418	Cellular
20.02.2022	7	486	Cellular
20.02.2022	8	254	Cellular
20.02.2022	9	411	Cellular
20.02.2022	10	184	Cellular
20.02.2022	11	249	Cellular
20.02.2022	12	217	Cellular
20.02.2022	13	166	Cellular
20.02.2022	14	245	Cellular
20.02.2022	15	168	Cellular
20.02.2022	16	121	Cellular
20.02.2022	17	106	Cellular
20.02.2022	18	213	Cellular
20.02.2022	19	230	Cellular
20.02.2022	20	213	Cellular
20.02.2022	21	282	Cellular
20.02.2022	22	178	Cellular
20.02.2022	23	234	Cellular
20.02.2022	24	196	Cellular
20.02.2022	25	238	Cellular
20.02.2022	26	404	Cellular
20.02.2022	27	176	Cellular
20.02.2022	28	167	Cellular
20.02.2022	29	138	Cellular
20.02.2022	30	272	Cellular
20.02.2022	31	163	Cellular
20.02.2022	32	216	Cellular

20.02.2022	33	219	Cellular
20.02.2022	34	458	Cellular
20.02.2022	35	194	Cellular
20.02.2022	36	379	Cellular
20.02.2022	37	208	Cellular
20.02.2022	38	647	Cellular
20.02.2022	39	179	Cellular
20.02.2022	40	182	Cellular
20.02.2022	41	488	Cellular
20.02.2022	42	149	Cellular
20.02.2022	43	327	Cellular
20.02.2022	44	260	Cellular
20.02.2022	45	316	Cellular
20.02.2022	46	287	Cellular
20.02.2022	47	169	Cellular
20.02.2022	48	215	Cellular
20.02.2022	49	214	Cellular
20.02.2022	50	172	Cellular
20.02.2022	51	194	Cellular
20.02.2022	52	506	Cellular
20.02.2022	53	411	Cellular
20.02.2022	54	436	Cellular
20.02.2022	55	328	Cellular
20.02.2022	56	146	Cellular
20.02.2022	57	408	Cellular
20.02.2022	58	209	Cellular
20.02.2022	59	407	Cellular
20.02.2022	60	172	Cellular
20.02.2022	61	126	Cellular
20.02.2022	62	189	Cellular
20.02.2022	63	308	Cellular
20.02.2022	64	428	Cellular
20.02.2022	65	346	Cellular
20.02.2022	66	472	Cellular
20.02.2022	67	142	Cellular
20.02.2022	68	188	Cellular
20.02.2022	69	266	Cellular
20.02.2022	70	442	Cellular
20.02.2022	71	438	Cellular
20.02.2022	72	146	Cellular
20.02.2022	73	277	Cellular
20.02.2022	74	124	Cellular
20.02.2022	75	327	Cellular

20.02.2022	76	342	Cellular
20.02.2022	77	252	Cellular
20.02.2022	78	501	Cellular
20.02.2022	79	240	Cellular
20.02.2022	80	244	Cellular
20.02.2022	81	247	Cellular
20.02.2022	82	250	Cellular
20.02.2022	83	250	Cellular
20.02.2022	84	155	Cellular
20.02.2022	85	271	Cellular
20.02.2022	86	179	Cellular
20.02.2022	87	327	Cellular
20.02.2022	88	149	Cellular
20.02.2022	89	258	Cellular
20.02.2022	90	147	Cellular
20.02.2022	91	150	Cellular
20.02.2022	92	318	Cellular
20.02.2022	93	330	Cellular
20.02.2022	94	336	Cellular
20.02.2022	95	140	Cellular
20.02.2022	96	140	Cellular
20.02.2022	97	127	Cellular
20.02.2022	98	275	Cellular
20.02.2022	99	198	Cellular
20.02.2022	100	143	Cellular

Table 3: The performance measurement of the optimized prototype using a cellular connection.

Date	Punch #	End-to-end time in ms	Connection
20.02.2022	1	121	Wi-Fi
20.02.2022	2	146	Wi-Fi
20.02.2022	3	113	Wi-Fi
20.02.2022	4	261	Wi-Fi
20.02.2022	5	264	Wi-Fi
20.02.2022	6	199	Wi-Fi
20.02.2022	7	189	Wi-Fi
20.02.2022	8	480	Wi-Fi
20.02.2022	9	217	Wi-Fi
20.02.2022	10	202	Wi-Fi
20.02.2022	11	195	Wi-Fi
20.02.2022	12	370	Wi-Fi
20.02.2022	13	183	Wi-Fi

20.02.2022	14		222	Wi-Fi
20.02.2022	15		103	Wi-Fi
20.02.2022	16		153	Wi-Fi
20.02.2022	17		151	Wi-Fi
20.02.2022	18		176	Wi-Fi
20.02.2022	19		121	Wi-Fi
20.02.2022	20		146	Wi-Fi
20.02.2022	21		113	Wi-Fi
20.02.2022	22		261	Wi-Fi
20.02.2022	23		195	Wi-Fi
20.02.2022	24		109	Wi-Fi
20.02.2022	25		212	Wi-Fi
20.02.2022	26		186	Wi-Fi
20.02.2022	27		289	Wi-Fi
20.02.2022	28		199	Wi-Fi
20.02.2022	29		310	Wi-Fi
20.02.2022	30		204	Wi-Fi
20.02.2022	31		188	Wi-Fi
20.02.2022	32		371	Wi-Fi
20.02.2022	33		285	Wi-Fi
20.02.2022	34		195	Wi-Fi
20.02.2022	35		170	Wi-Fi
20.02.2022	36		313	Wi-Fi
20.02.2022	37		145	Wi-Fi
20.02.2022	38		102	Wi-Fi
20.02.2022	39		278	Wi-Fi
20.02.2022	40		256	Wi-Fi
20.02.2022	41		363	Wi-Fi
20.02.2022	42		319	Wi-Fi
20.02.2022	43		351	Wi-Fi
20.02.2022	44		205	Wi-Fi
20.02.2022	45		196	Wi-Fi
20.02.2022	46		299	Wi-Fi
20.02.2022	47		209	Wi-Fi
20.02.2022	48		255	Wi-Fi
20.02.2022	49		127	Wi-Fi
20.02.2022	50		140	Wi-Fi
20.02.2022	51		397	Wi-Fi
20.02.2022	52		347	Wi-Fi
20.02.2022	53		107	Wi-Fi
20.02.2022	54		164	Wi-Fi
20.02.2022	55		377	Wi-Fi
20.02.2022	56		142	Wi-Fi

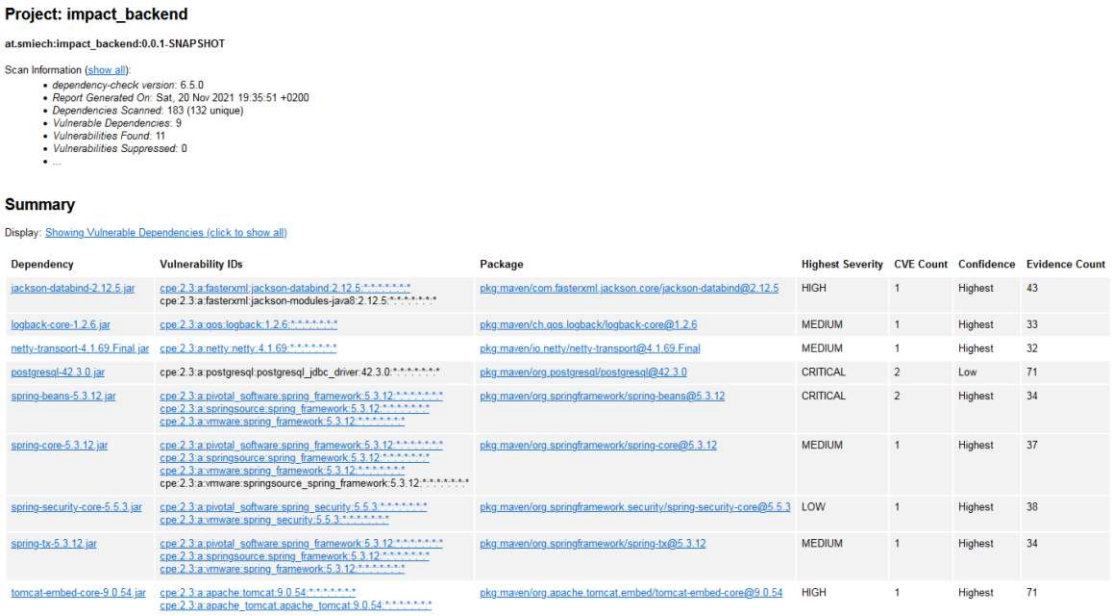
20.02.2022	57	132	Wi-Fi
20.02.2022	58	145	Wi-Fi
20.02.2022	59	183	Wi-Fi
20.02.2022	60	189	Wi-Fi
20.02.2022	61	202	Wi-Fi
20.02.2022	62	258	Wi-Fi
20.02.2022	63	147	Wi-Fi
20.02.2022	64	143	Wi-Fi
20.02.2022	65	155	Wi-Fi
20.02.2022	66	134	Wi-Fi
20.02.2022	67	192	Wi-Fi
20.02.2022	68	108	Wi-Fi
20.02.2022	69	169	Wi-Fi
20.02.2022	70	376	Wi-Fi
20.02.2022	71	135	Wi-Fi
20.02.2022	72	121	Wi-Fi
20.02.2022	73	247	Wi-Fi
20.02.2022	74	201	Wi-Fi
20.02.2022	75	190	Wi-Fi
20.02.2022	76	273	Wi-Fi
20.02.2022	77	236	Wi-Fi
20.02.2022	78	136	Wi-Fi
20.02.2022	79	242	Wi-Fi
20.02.2022	80	174	Wi-Fi
20.02.2022	81	173	Wi-Fi
20.02.2022	82	134	Wi-Fi
20.02.2022	83	318	Wi-Fi
20.02.2022	84	318	Wi-Fi
20.02.2022	85	164	Wi-Fi
20.02.2022	86	231	Wi-Fi
20.02.2022	87	177	Wi-Fi
20.02.2022	88	132	Wi-Fi
20.02.2022	89	105	Wi-Fi
20.02.2022	90	160	Wi-Fi
20.02.2022	91	262	Wi-Fi
20.02.2022	92	136	Wi-Fi
20.02.2022	93	281	Wi-Fi
20.02.2022	94	102	Wi-Fi
20.02.2022	95	357	Wi-Fi
20.02.2022	96	159	Wi-Fi
20.02.2022	97	227	Wi-Fi
20.02.2022	98	159	Wi-Fi
20.02.2022	99	161	Wi-Fi

20.02.2022	100	122	Wi-Fi

Table 4: The performance measurement of the optimized prototype using a Wi-Fi connection.

Security Details

OWASP Dependency-Check Report



Project: impact_backend

at.smiech:impact_backend:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 6.5.0
- *Report Generated On*: Sat, 20 Nov 2021 19:35:51 +0200
- *Dependencies Scanned*: 183 (132 unique)
- *Vulnerable Dependencies*: 9
- *Vulnerabilities Found*: 11
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
jackson-databind-2.12.5.jar	cpe:2.3:a:fasterxml:jackson-databind:2.12.5:*:*:*:*:*:*:* cpe:2.3:a:fasterxml.jackson-modules-java8:2.12.5:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.12.5	HIGH	1	Highest	43
logback-core-1.2.6.jar	cpe:2.3:a:qos.logback:1.2.6:*:*:*:*:*:*:*	pkg:maven/ch.qos.logback/logback-core@1.2.6	MEDIUM	1	Highest	33
netty-transport-4.1.69.Final.jar	cpe:2.3:a:netty:netty:4.1.69:*:*:*:*:*:*:*	pkg:maven/io.netty/netty-transport@4.1.69.Final	MEDIUM	1	Highest	32
postgresql-42.3.0.jar	cpe:2.3:a:postgresql:postgresql_jdbc_driver:42.3.0:*:*:*:*:*:*:*	pkg:maven/org.postgresql/postgresql@42.3.0	CRITICAL	2	Low	71
spring-beans-5.3.12.jar	cpe:2.3:a:pivotal_software:spring_framework:5.3.12:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.3.12:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.3.12:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-beans@5.3.12	CRITICAL	2	Highest	34
spring-core-5.3.12.jar	cpe:2.3:a:pivotal_software:spring_framework:5.3.12:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.3.12:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.3.12:*:*:*:*:*:*:* cpe:2.3:a:vmware:springsource_spring_framework:5.3.12:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-core@5.3.12	MEDIUM	1	Highest	37
spring-security-core-5.5.3.jar	cpe:2.3:a:pivotal_software:spring_security:5.5.3:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_security:5.5.3:*:*:*:*:*:*:*	pkg:maven/org.springframework.security/spring-security-core@5.5.3	LOW	1	Highest	38
spring-tx-5.3.12.jar	cpe:2.3:a:pivotal_software:spring_framework:5.3.12:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.3.12:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.3.12:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-tx@5.3.12	MEDIUM	1	Highest	34
tomcat-embed-core-9.0.54.jar	cpe:2.3:a:apache:tomcat:9.0.54:*:*:*:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.54:*:*:*:*:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.54	HIGH	1	Highest	71

Figure 1: OWASP dependency-check report result details of all found vulnerabilities in the unoptimized build on 20.11.2021.

Project: impact_backend

at.smiech:impact_backend:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 6.5.3
- *Report Generated On*: Mon, 07 Feb 2022 23:15:48 +0200
- *Dependencies Scanned*: 181 (124 unique)
- *Vulnerable Dependencies*: 1
- *Vulnerabilities Found*: 1
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
tomcat-embed-core-9.0.56.jar	cpe:2.3:a:apache:tomcat:9.0.56:*:*:*:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.56:*:*:*:*:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.56	HIGH	1	Highest	71

Figure 2: OWASP dependency-check report result details of the only found vulnerability in the optimized build on 07.02.2022.

SonarQube

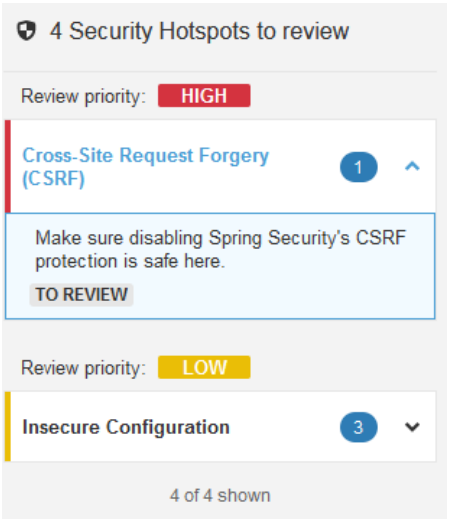


Figure 3: SonarQube details of the security hotspots in the unoptimized version on 20.11.2022.

Figure 4: SonarQube details of the security vulnerabilities in the unoptimized version on 20.11.2022.

REST API Specification

The the REST API which has been created using springdoc-openapi has the following method and path structure:

- POST /login
- GET /userdetails
- GET /participant
- GET /session
- GET /equip
- POST /measurement

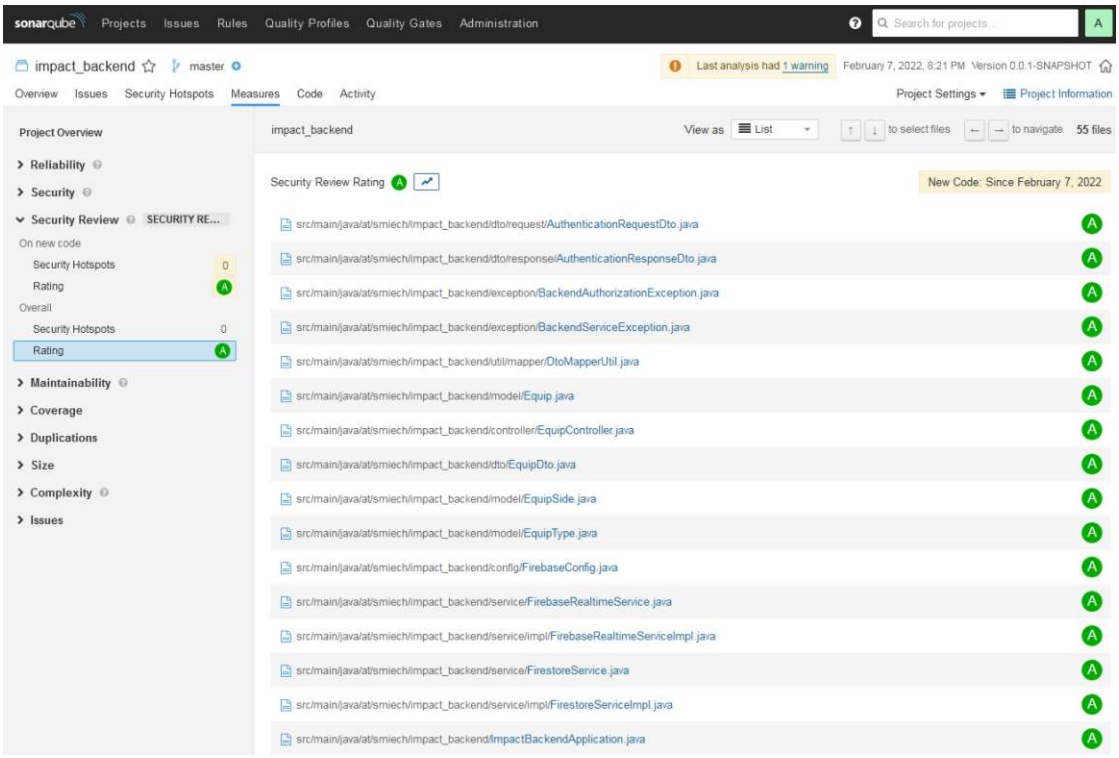


Figure 5: SonarQube security review details of all of the individual 55 backend files in the optimized build. This final SonarQube analysis and evaluation of the prototype implementation has been conducted on 07.02.2022.