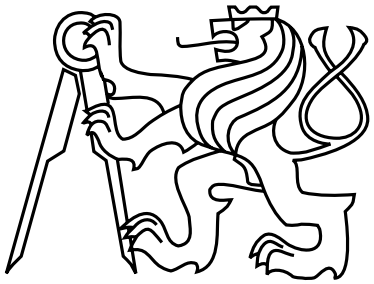
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science



Bachelor thesis

Anonymous Communication Between Students and Teachers

Samuel Klas

Supervisor: Ing. Božena Mannová, Ph.D.

Study Programme: Open informatics

Field of Study: Software

May 20, 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Klas** Jméno: **Samuel** Osobní číslo: **483772**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Anonymní komunikace mezi studenty a učiteli

Název bakalářské práce anglicky:

Anonymous Communication Between Students and Teachers

Pokyny pro vypracování:

Proveďte analýzu, návrh a implementaci aplikace pro podporu anonymní komunikace mezi učiteli a studenty. Cílem práce je zatraktivnit a zpříjemnit výuku studentům a vyučujícím tak, aby se studenti aktivněji zapojovali do výuky a zjednodušit komunikaci mezi studentem a učitelem. Proveďte analýzu požadavků a hodnocení již existujících nástrojů pro tuto komunikaci (např. TEAMS). Na základě této analýzy navrhněte základní funkcionality navrhované aplikace, jako jsou konferenční hovor, sdílení obrazovky, chat a možnost vytváření diskusních skupin. Zvolte architekturu aplikace a technologie pro implementaci. Navrhněte přívětivé uživatelské prostředí. Věnujte se i problematice bezpečné komunikace. Aplikaci otestujte. Zhodnoťte výsledky a navrhněte případné další funkcionality nebo jiná zlepšení. Využijte vhodných prostředků SE

Seznam doporučené literatury:

[1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach ,
ISBN-10: 9780078022128
[2] https://blog.genial.ly/en/techniques-online-communication-students-and-teachers/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Božena Mannová, Ph.D., kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.09.2021** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Božena Mannová, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Acknowledgements

First of all, I am extremely grateful to my supervisor, Ing. Božena Mannová, Ph.D. for her advice, patience and endless positivity she has granted me throughout the creation of this thesis. Writing this thesis would not have been possible without her support.
I would also like to extend my thanks to my family and friends for their emotional support, helping me keep my spirits high during my entire university journey.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Pavlice, Slovakia on 20. 5. 2022 .....

Abstract

This thesis concerns itself with supporting teaching in a university setting by creating an application that allows students to communicate anonymously with their teachers. By hiding their identity, it helps students alleviate their anxiety, encourages them to actively participate more in class and have a more positive experience. An analysis of currently used communication software served as basis for the design, implementation and testing of the application.

Keywords: classroom participation, anonymity, communication, web application, Java, Spring, React

Abstrakt

Táto práca sa zaoberá podporou výučby v univerzitnom prostredí vytvorením aplikácie, ktorá umožňuje študentom so svojimi učiteľmi komunikovať anonymne. Skrytím ich identity im pomáha zmierniť ich úzkosť, povzdudzuje ich sa viac aktívne zúčastňovať na hodinách a mať z nich pozitívnejší zážitok. Analýza sučasne používaných komunikačných softwarov slúžila ako základ pre dizajn, implementáciu a testovanie aplikácie.

Kľúčové slová: účasť v triede, anonymita, komunikácia, webová aplikácia, Java, Spring, React

Preklad názvu: Anonymná komunikácia medzi študentmi a učiteľmi

Contents

List of Figures

List of source code

Chapter 1

Introduction

Classroom participation refers to the methods students use to actively take part in classes. This can be done by way of raising one's hand and responding to the teacher's questions, asking one's own questions, partaking in discussions or sharing one's opinions. Participating helps students familiarize themselves with the subject matter, clearly present their arguments and shows the teacher they have made an effort to understand their teaching. Students who actively participate in class often take away more from the subject as they regularly engage with it on a deeper level compared to simple memorization.

1.1 Motivation

Despite all these advantages, most students rarely choose to take active part in classes. Whether because of poor communication skills, fear of looking foolish by asking for clarification on simple topics or struggling to speak before large crowds of people, the anxiety of speaking up is a real issue among students. As study material tends to build on top of previously explained and discussed topics, by sitting silently instead of asking for explanations, the student not only robs themselves of another chance to understand the topic, but makes it more difficult for them to keep up with the subject as they do not understand the core concepts.

It is easy to talk about these issues because I experienced them myself during my studies. Even though most teachers encouraged the asking of questions, no matter how trivial they seemed, I did not take their advice. I was anxious and feared being judged for my lack of knowledge, more often coming from my colleagues rather than the teacher, so I searched for explanations on the internet instead. Sometimes the answers I found were satisfactory, but more often they were either incomplete or straight up incorrect, so I spent more time catching up, understood less of the subject matter and ended up walking away with a worse grade than I could have, had I asked for help at the right time.

1.2 Objective

The objective of this bachelor thesis is to help these students alleviate some of this social anxiety and encourage them to participate more often in classes. We hope to achieve this by providing a way to communicate anonymously with their teachers and fellow students during and outside of lectures. By hiding their identity, any question or contribution will not be traceable back to the student, eliminating the irrational fear of prejudice or laughter at their expense, which should help them speak up.

The process of creating this platform starts out with the review and analysis of existing software used for communication between students and teachers. The output of this analysis is then used to decide on the key features and functions of our solution, followed by the design, choice of used technologies and implementation of the application.

1.3 Application overview

The communication platform was implemented as a web application that supports communication through forums and live chat. It also allows teachers to create simple quizzes for students to solve as preparation for lessons, and generate statistics regarding completion rate and the average success rate of each question to see which topics the students are struggling with the most.

1.4 Chapter overview

The thesis consists of five different chapters, each of which concerns itself with a phase of development mentioned in the Objective section. The chapters are as follows :

- **Review of existing software**, which reviews existing options in terms of popular communication platforms, highlights their benefits and drawbacks, and draws inspiration for which functionalities could be used or adapted for use in anonymous communication.
- **Business analysis** describes the key functionalities and requirements based on the previous review.
- **Design**, which talks about the design of the application, including the model class diagram, chosen technologies and frameworks and the reasons behind them.
- **Implementation** goes over the implementation details of various functionalities of the application, including snippets of code, screenshots of the application and the discussion of alternative solutions.
- **Testing** of the application prototype, including testing phases and issues revealed through it.
- **Conclusion**, which summarizes the work and talks about future development of the application.

Chapter 2

Review of existing software

In this chapter, we review some of the popular existing solutions while focusing on their key functionalities. While none of the reviewed applications were designed with full anonymity in mind, there are features that could be adapted for use in anonymous communication.

2.1 Microsoft Teams

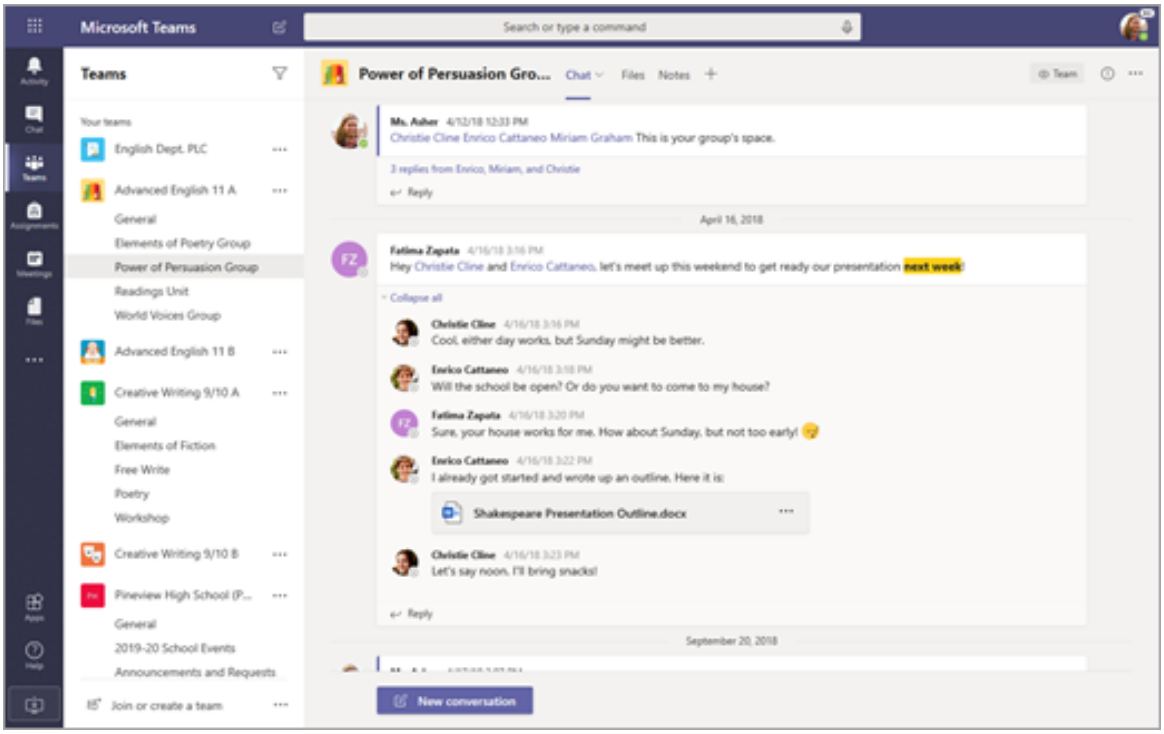


Figure 2.1: Example of the Microsoft Teams environment, with a live chat visible in the center and a list of channels on the left side[1]

Microsoft Teams is a communication platform developed by Microsoft. Though mainly used for business communication, it has been widely adopted by many high schools and universities across the world as their primary communication platform during the COVID-19 lockdown. Microsoft Teams allows its users to schedule meetings, voice calls with the option of adding video or screen sharing. It also allows them to create and join teams, where they can communicate using text messages, images and GIFs, share files and create meetings joinable by other team members. Users can also create private or group chats without the need to belong to the same team.[2]

Teachers can use the Assignments feature to assign homework to students. This is done by creating multi choice quizzes or by requiring students to upload files containing their work. They can assign them to individuals or groups of students, grade, add feedback or return their work in case they are not satisfied with the results. The advantages of Microsoft Teams include a great variety of additional functionalities in the form of addons, and the fact that it comes with no additional costs if the company already owns a Microsoft 365 license.

The drawbacks include the lack of a forum feature. Although this functionality can be simulated by using different channels and conversations within these channels, this solution is not nearly as intuitive as a regular forum. Another disadvantage comes in the form of security risks due to the sharing of resources within teams, as team members could potentially share malicious files or accidentally expose confidential information.

2.2 Moodle

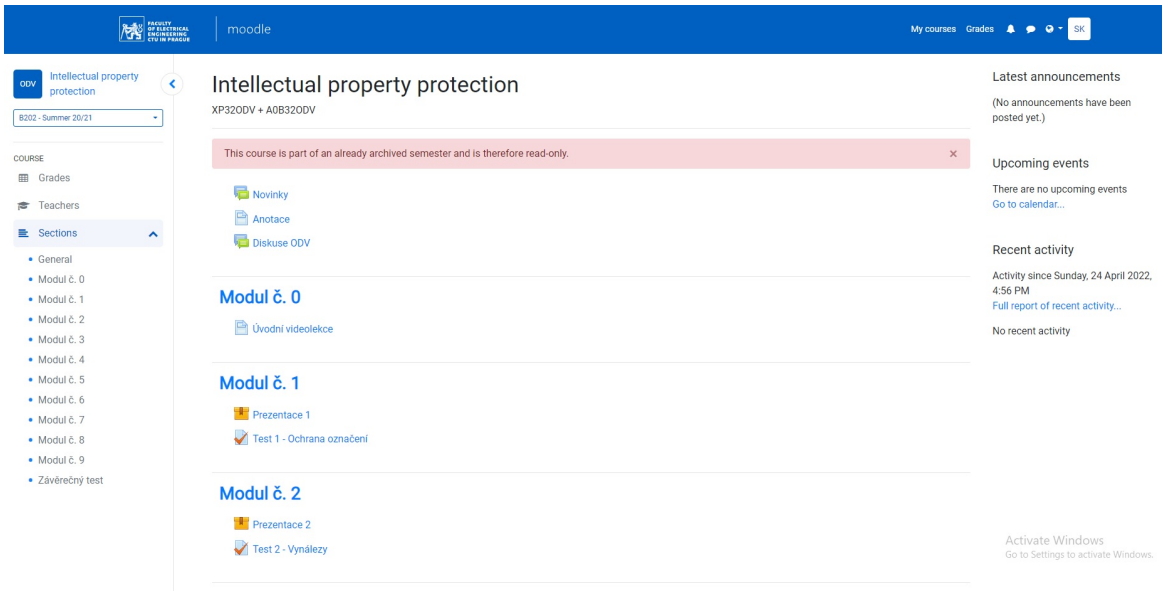


Figure 2.2: Example of the Moodle environment, specifically the subject Intelectual property protection from the Faculty of electrical engineering at the Czech technical university[3]

Moodle is a free learning management system (LMS for short) distributed under the GNU general public license as open source software. Moodle supports many of the functionalities as Teams, such as resource sharing, assigning and grading homework, communication between users including a course wide discussion forum.[4]

It also supports plugins to extend existing functionalities or add new ones, for example allowing video conferencing by integrating applications such as Zoom[1] or BigBlueButton[2], and themes to change the look and functionality of the whole site or specific courses.

Although technically free in case the institution installs it on their own server, they will need to take into account the cost of maintaining the server and the need to hire administrators to support it. Moodle offers several plans of cloud hosting on their own servers by way of MoodleCloud, with the most expensive plan allowing up to 1000 users and 5GB of storage space[5], so it is not a valid solution for schools whose needs exceed these values.

2.3 Discord

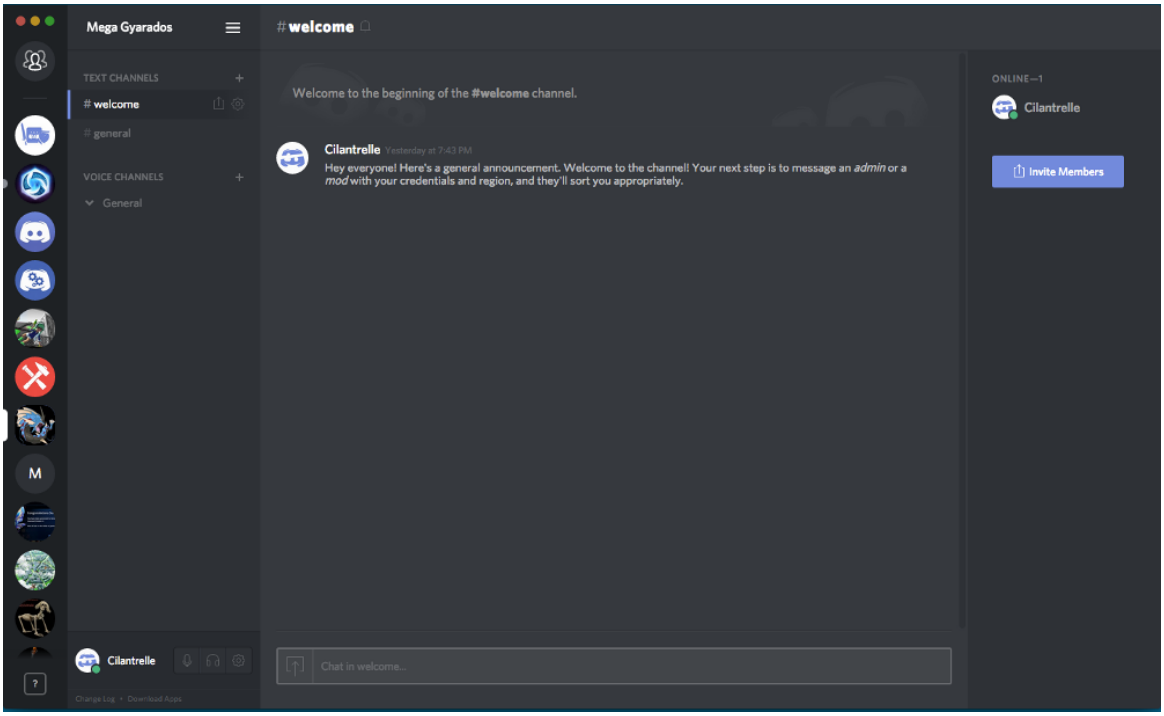


Figure 2.3: Example of the Discord environment[6]

Discord is a communication platform for online communities. It is available both as a desktop and a web application. The main building block of Discord are servers, which consist

[1] https://zoom.us/
[2] https://bigbluebutton.org/

of multiple text and voice channels. Users are invited to the servers exclusively via invite links and assume different roles in each server.

Text channels support sending text messages, pictures and emotes, allow users to tag[3] other users based on their identity or roles on the specific server. Voice channels support video and screen sharing of multiple users at the same time.[7]

A unique advantage of Discord is the ability for users to set a a different identity on different servers, and to change this identity whenever they wish. Another advantage is the option to create and add bots to servers or channels to enhance the user experience by reacting to different chat commands or events in the server.

2.4 Takeaways

The first and most obvious takeaway from these communication platforms is the need for a feature that allows users to communicate with each other. While live chat offers synchronous and more dynamic communication, forum threads tend to be focused on a single topic and active over a longed period of time. In both cases, adapting them for anonymity is quite simple, as it only requires us to hide the information about the student from the other users. We have decided to implement both of these methods with the mentioned modification.

Another useful functionality would be a way for teachers to assess their students' knowledge. If teachers were to issue assignments to students in our application, they would not be able to see individual students' results. This of course means that the assignments would not be gradable or compulsory, as there would be no way of knowing the grade an individual student achieved, or even if they completed the assignment. Still, we believe this feature to be worthwhile, as it would allow students to test their knowledge and for teachers to see which topics are clear and which the students struggle with the most by looking at the assignment statistics.

In case a student's identity gets revealed for some reason, it is important for them to have the ability to generate a new identity, as is the case for Discord users.

Although practically all modern communication software offers some form of voice communication, sometimes along with screen and video sharing functionality, including all three of the reviewed applications, we have decided not to include it in our application, as it would be very difficult to implement while retaining full student anonymity.

[3] slang - A person is tagged when they are identified in a post on social media.

Chapter 3

Business analysis

3.1 Functional and non functional requirements

This section concerns itself with the analysis of the functional and non functional requirements, which arise from the initial assignment specification and discussion with the thesis supervisor.

3.1.1 Functional requirements

1. Hidden student identity
    - The students' real identities will be hidden from teachers and other students. Each student will be assigned a display name at the time of registering their account. Each student will be able to generate a new display name.
2. Invite links
    - The application will support the adding of students to subject via invite links. Teachers will be able to generate an invite link and set its expiration time. They will also be able to deactivate these links prematurely.
3. Discussion forum
    - Each subject will contain a discussion forum. Users belonging to this subject will be able to create new threads. They will also be able to create new posts within these threads, reply to other users', edit and delete their own posts. The creator of the thread will be able to mark the thread as closed, after which no more changes to this thread will be possible.
4. Subject wide chat
    - The application will allow users to open a live chat to ask questions or discuss issues regarding the chosen subject. Any user belonging to this subject will be able to participate in this chat. Teachers' replies will be easily discernible from the students' replies.

5. Quizzes

    - The application will provide an efficient way for teachers to create quizzes and publish them in their respective subjects. Each quiz will contain multiple questions, each question will provide multiple possible answers with a single correct choice. The quiz results will be immediately visible to the student after sending in their solution, including the achieved point total and the correctness of each answer. The application will enable teachers to generate the statistics of each quiz to see the participation, average achieved overall point total and the average achieved points for each question in the quiz.

3.1.2 Non functional requirements

1. Security

    - The application will be resistant to common cyber attacks such as injection attacks or cross site scripting. User passwords will be stored in their hashed form using a strong encryption algorithm. Server resources will be protected and only accessible to users with the required authorization.

2. Performance

    - The application shall be able to load every page in 3 seconds or less when the number of active users is less than 1000.

3. Responsive design

    - The application frontend will be designed with responsiveness in mind, so that it renders well on devices of different screen sizes and viewports by scaling and rearranging components accordingly.

3.2 Process domain model

This section describes some of the essential processes that will take place while using the application. The diagrams contain two actors, a user and the system. The user, being the primary actor, initiates the process while the system interacts with the user. While these diagrams do not differentiate users based on their roles, in the application this will not hold true, e.g, a student will not be able to create a new subject.

3.2.1 Creating new thread

Any user belonging to the subject will be able to create a new thread in its forum. The user will be asked to provide a unique thread title and content. After submitting the form, the system will create a thread with the title and an initial post with the content inside the thread.

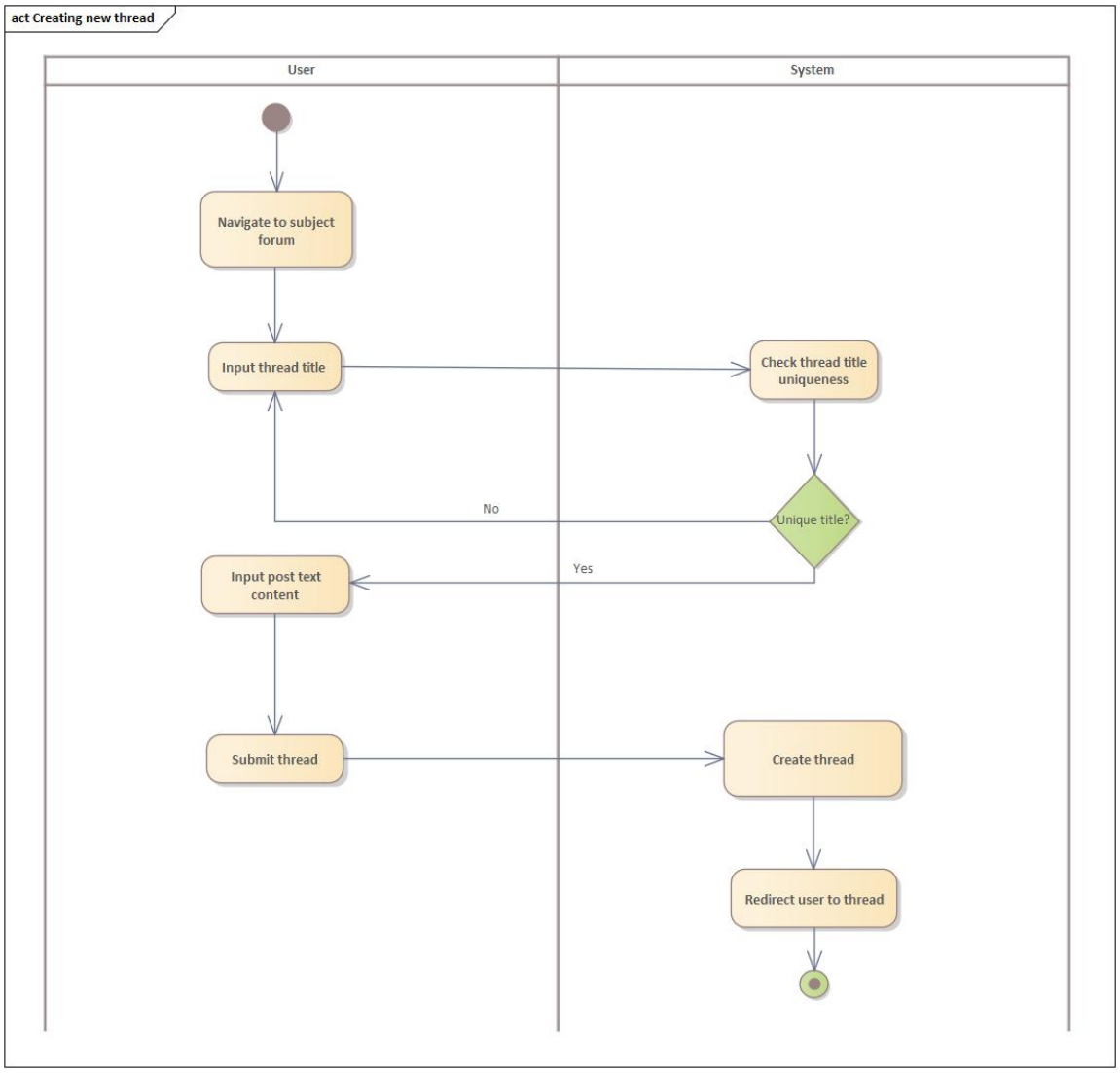


Figure 3.1: Creating new thread process diagram

3.2.2 Closing subject

A teacher belonging to a subject will be able to close it, choosing to either archive or delete the subject. If they choose to archive it, the subject will remain browsable by users belonging to it, but it will block all interaction within the subject and will not allow new users to join. By choosing to delete the subject, all data contained within the subject will be deleted and therefore no longer be accessible. This method should only be chosen after careful consideration, as the subject information may be useful to students even long after the closing of the subject.

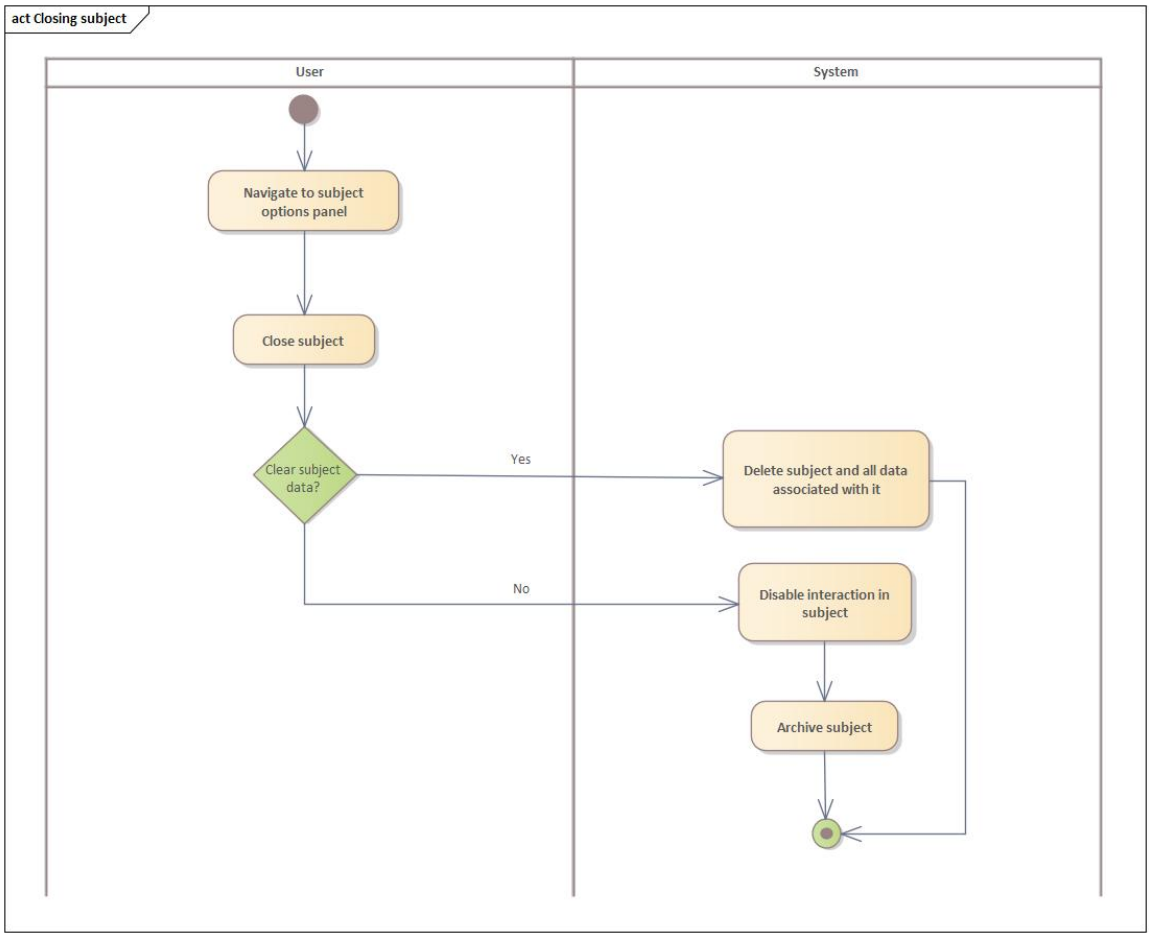


Figure 3.2: Closing subject process diagram

3.2.3 Creating quiz

When creating a quiz, the teacher will need to input a title first, then add as many questions as they want. Within each question, they will need to assign the points achievable by correctly answering the question and add a number of choices, one of which they will mark as correct. After submitting the quiz, it will be added to the subject quiz list with all subject students being able to solve it.

3.3 Business domain model

Based on the analysis of the functional requirements and the process model, we have come up with a concept of the business domain model, which describes the hierarchy of the entities in the system. It can be roughly divided into four parts:

- User on the top left
- Live discussion chat on the top right

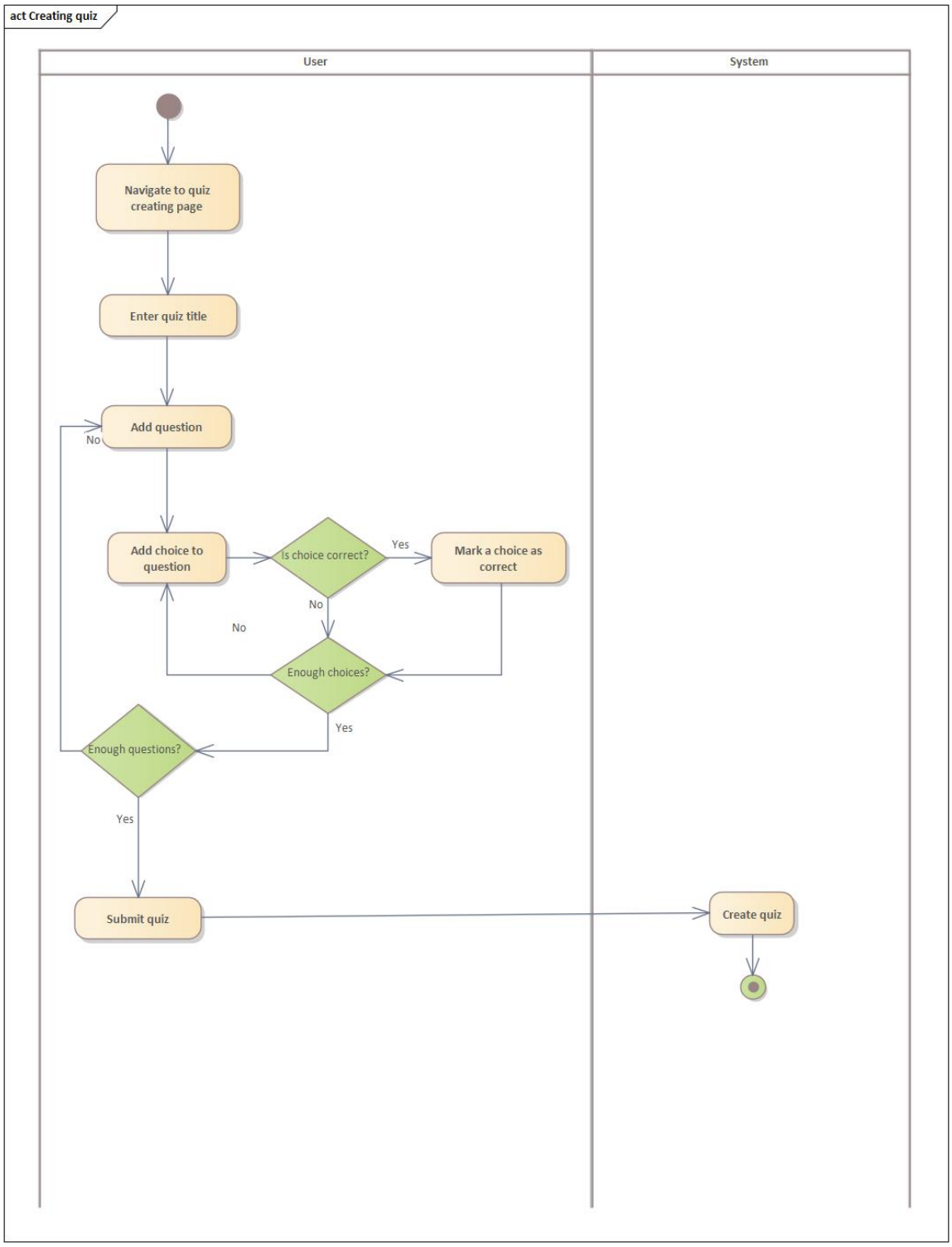


Figure 3.3: Creating quiz process diagram

- Forum in the bottom left
- Quiz on the bottom right

The reason why users are not directly connected to the chat entity is because it describes a subject wide chat, with all users in a subject being able to participate by default.

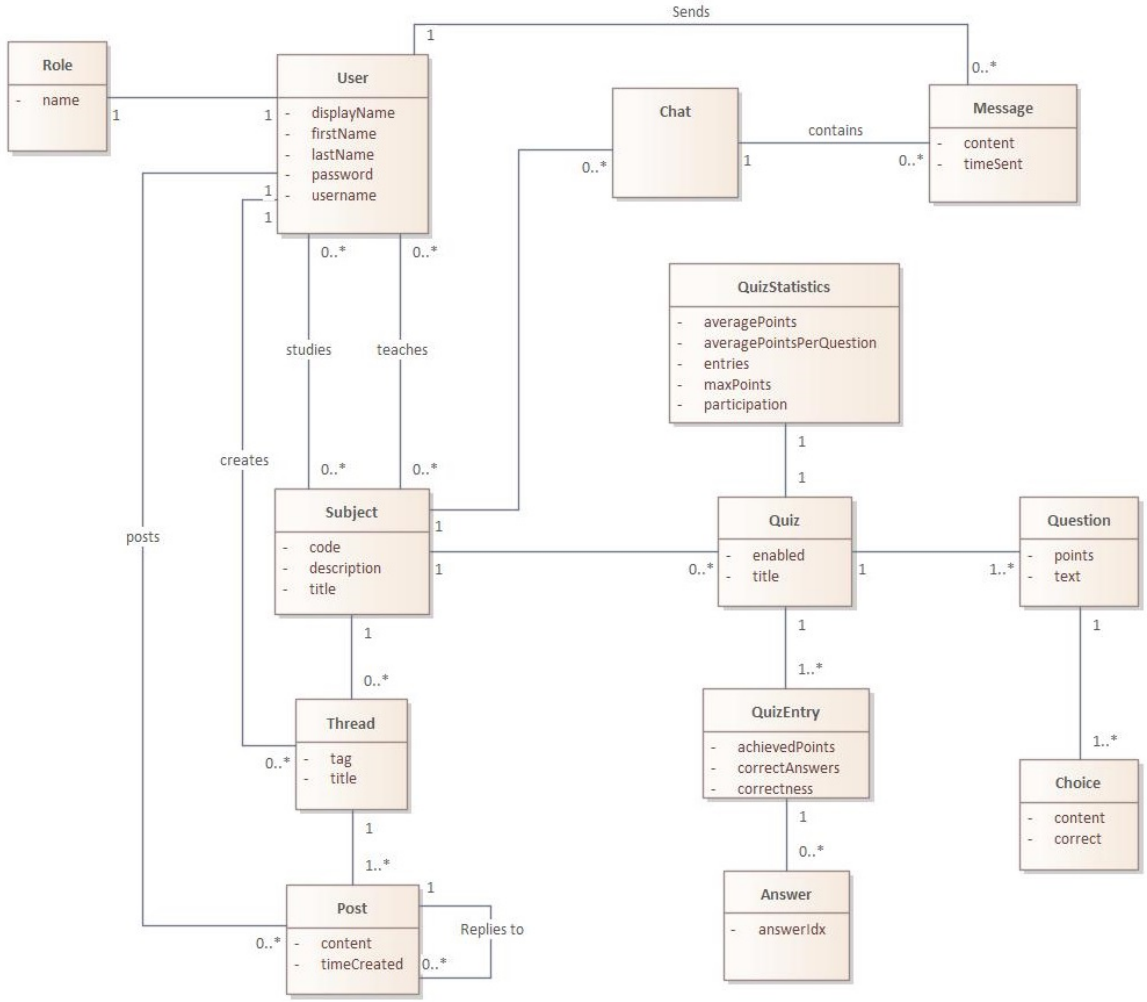


Figure 3.4: Business domain model

Chapter 4

Design

Based on the analysis of the functional and non functional requirements done in the previous chapter, as well as the main instructions of the thesis assignment, it is necessary to choose suitable technologies to implement the application. As modern technologies overwhelmingly support the creation of web applications, we have decided to implement our project as such. There are two main design pattern of developing web applications : single page and multi page applications. The next section briefly describes both of these approaches, their benefits and drawbacks, and the decision which one of them to use to implement the application.

4.1 Single page and multi page application comparison

4.1.1 Multi page application

"A Multi-page Application is a web application consisting of a large number of pages completely refreshed every time when data changes on them. Any data change or data transfer to the server leads to a new page displayed in the browser"[8].

Also called traditional applications, they perform most of the application logic on the server, responding to each client request by creating and serving a new HTML page, which places a relatively high workload on the server, while requiring very little of the client side. Applications created using this method usually retain either full or partial functionality even in browsers without JavaScript support.

The main advantage of multi page applications is search engine optimization. Search engine crawlers are optimized for reading websites created in this manner, as they are fully rendered by the time they access them, as well as not being reliant on JavaScript to display the information contained within them, which the crawlers have historically had trouble interpreting[9].
As multi page applications are quicker and easier to crawl, this leads to search engines having a higher chance of connecting a user's search query keywords to the content displayed on the

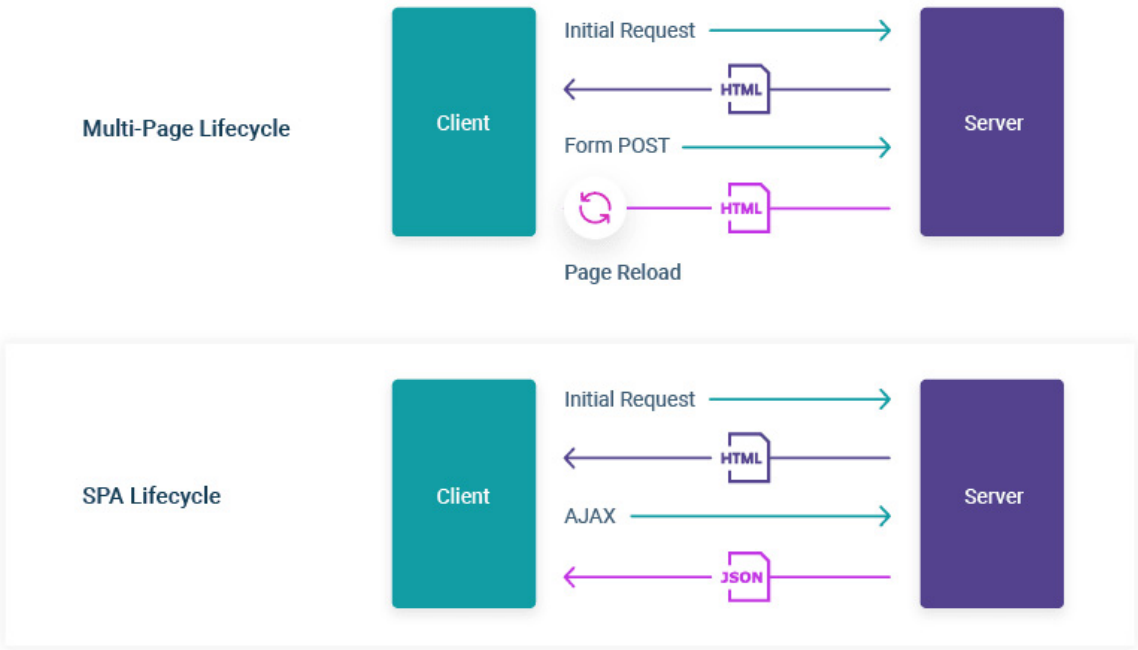


Figure 4.1: Multi page and single page application lifecycle comparison[8]

website, generating more traffic and attracting potential customers.

Popular examples of multi page applications are e-shop websites such as eBay[1] or Amazon[2]

4.1.2 Single page application

"A Single-page Application is a type of web application loaded from one page, and all user interaction with this service is carried out using one screen (page)"[8].

The way a typical single page application works is that on the initial load of the application, a mostly empty HTML page is loaded into the browser, along with a bundle of JavaScript code that is used to run the application. After this initial load, whenever the client makes a request, instead of returning a full HTML page, the server provides the data in the JSON[3] format using AJAX[4], which the application uses alongside JavaScript to render the appropriate view. As the browser no longer needs to wait for server responses every time they navigate to a new view, the transitions are faster and more seamless, which in turn improves user experience.

[1] https://www.ebay.com/
[2] https://www.amazon.com/
[3] JavaScript Object Notation
[4] Asynchronous JavaScript and XML

Another advantage that comes with the single page application implementation is the lessening of the load on the server. As the server only responds with raw data when necessary instead of creating and serving new HTML pages on every navigation change, instead leaving that responsibility to the browser, it frees up server resources to be used elsewhere.

Because of single page applications' heavy reliance on JavaScript to function, when accessing them from browsers with it disabled, they will either lose core parts of their functionality or stop functioning altogether. Although statistics on JavaScript usage are hard to find, based on Yahoo's analysis of traffic on their website in 2010, around 1.3 percent of users had JavaScript disabled[10]. We can only imagine that number has gone down throughout the following years due to the development of web technologies.

4.1.3 Conclusion

To complete one of the main tasks in the assignment, to create a convenient and user-friendly interface, and to best fulfill the non functional requirement of fast response times, as well as not having to consider search engine optimization as a priority for this project, we have chosen to implement the application as a single page application.

4.2 Used Technologies

4.2.1 Spring

Spring is an open source Java framework distributed under the Apache License 2.0 that provides infrastructure support for developing web applications[11]. It is one of the most popular backend frameworks and the most popular java backend framework according figure 4.2.

The main feature of Spring as a framework is the use of dependency injection. Dependency injection is a design pattern in which objects declare their dependencies, meaning the objects they require to function, which are provided for them by external code, in Spring called the IoC container. These objects are created, configured and managed by the container instead of their owning objects[12]. It is a form of inversion of control.
The main advantages of dependency injection is the loose coupling between application components, which increases their reusability and makes unit testing easier.

Spring consists of several modules which provide built in support for core functionalities of modern web applications, such as the Core Container module, which is responsible for the dependency injection feature described previously, data access, web, security or testing modules.

4.2.1.1 Spring Boot

One of the main disadvantages of Spring as a framework is its complexity and steep learning curve, which stems from the large amount of configuration needed to be written for each part of the application to function. It also requires developers to manually define all the

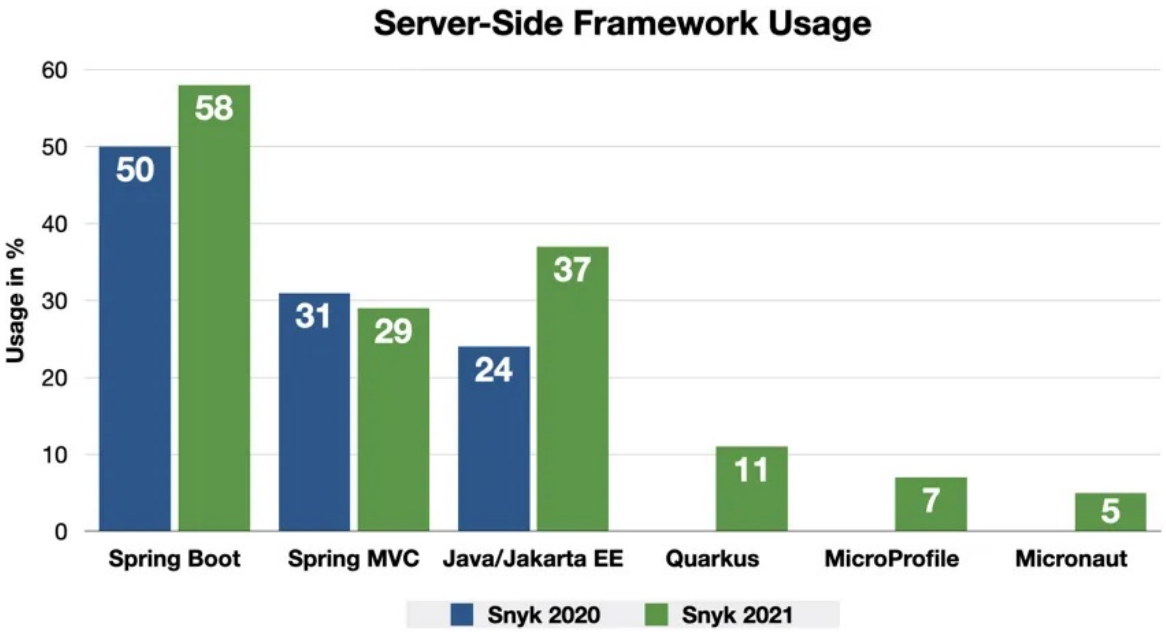


Figure 4.2: Java server-side framework usage in 2020 to 2021[13]

dependencies to be used in the application. This introduces needless complexity and time costs, as developers need to write a lot of boilerplate code before handling actual application logic.

Spring Boot is an extension of Spring which aims to solve these issues. It helps the configuration issue by way of auto-configuration, which automatically configures the application based on its specified dependencies, with the option of overriding this configuration if required[14].

It handles the issue of managing dependencies by introducing starters, which are sets of dependencies bundled together under a single name. For example, the spring-boot-starter-test starter includes the JUnit[5], Hamcrest[6] and Mockito[7] library dependencies, all of which are widely used for testing java applications.

Among other features, it also provides an embedded server that assists in testing the application functionality while in development, as instead of needing to set up a web server and deploying the application, we just need to run the application.

4.2.1.2 JPA

Java Persistence API is a specification that concerns itself with the persistence of data in Java applications[15]. It allows us to convert data from relational databases to java objects

[5] https://junit.org/junit5/
[6] http://hamcrest.org/JavaHamcrest/tutorial
[7] https://site.mockito.org/

and vice versa as a way of achieving Object Relational Mapping, which allows us to work with objects rather than having to write SQL statements. The mapping of objects and their relationships into database tables is configured by using annotations in Java classes that the JPA implementation uses to execute SQL statements.
As JPA is only a specification, it requires an implementation to perform the previously mentioned functionalities. For this role, Hibernate[8] was chosen.

While JPA is used to model the domain classes, the data access layer implementation is simplified by making use of the `Repository` interface from the Spring Data JPA module[16]. Each repository takes in the domain class and the data type of its ID as arguments and provides baseline CRUD[9] functionality for the class it manages. It supports creating simple queries by declaring their method in the repository class, for which Spring automatically generates the implementation based on the keywords and parameters in the method signature, as well as writing custom SQL queries and binding them to repository methods by using the `@Query` annotation.

```
public interface UserRepository extends JpaRepository<User, Long> {
    List<User> findByFirstName(String firstName);
    User findByUserName(String userName);
    User findByDisplayName(String displayName);
    List<User> findAll();
}
```

Listing 4.1: Repository class with custom queries

4.2.1.3 Security

Security in web application consists mainly of the following concepts: authentication and authorization.[17]

Authentication is the process of verifying a user's identity by requesting identifying credentials, comparing them to those stored in the application's database or an authentication server, and granting access to the application in case they match. One of the most simple ways of providing credentials is through the use of a password.

Authorization is the process of granting users access to application resources and features based on their configured permissions. A common way of determining this is by assigning users different roles. For example, a customer of an e-shop website are able to browse and order goods, while an admin is able to see incoming orders and purchases of all customers.

Spring supports both authentication and authorization, including protecting server resources by restricting user access based on their role and the configuration of custom servlet filters to intercept, manipulate, or block requests before accessing these resources[18]. It also offers implementations of various hashing algorithms to securely store passwords.

[8] https://hibernate.org/orm/
[9] create, read, update, delete

4.2.2 Authentication schemes

The following types of authentication were considered:

- **Basic authentication** is a simple mechanism in which the user sends their credentials in the form "username:password", which is then encoded using base64 encoding[19]. The resulting string is then included in the Authorization header in each request the user sends to the server. The server can then decode the content of the header and verify the user's identity. Because this information is sent with every request, along with the fact that it is easily decoded by a potential attacker if intercepted, it should only be used when communicating through secure channels.
- **Session based authentication**, in which the server creates a session for the user once they log in and send back the session ID. The session id is then stored in a cookie on the user's machine and sent along with every request while the user is logged in[20]. The server checks the session id against the session stored in the database. As the sessions are created and stored on the server, it may lead to performance issues when a large number of users are active at the same time. On the other hand, because of the same reason, this method allows admins of the system to terminate the session in case they suspect the user account was compromised.
- **Token authentication** works by providing the user a special access token once they have logged in[21]. The token is stored on the client side and is, similar to basic authentication, included in the Authorization header with each request. This allows the server to skip authenticating the user on every request, as the presence of a valid access token in the request acts as proof of the user's identity. For this approach to work, the server must be able to generate tokens that cannot be easily guessed by attackers, as well as having a way to check the validity of received tokens. This method is efficient and scalable, as tokens are stored on the client side, and checking the validity of tokens is much faster than querying the database.

For this project, token authentication was chosen because of the previously mentioned reasons and the relatively simple implementation thanks to available options such as the JSON Web Token.[10]

4.2.3 JWT

"JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed[22]."
The JWT in its encoded form is made up of three parts separated by dots:

- Header, which contains the type of the token, in this case the JWT, and the type of the signing algorithm used.
- Payload, made up of information about the user, such as the username and their roles, and additional information such as the expiration time.

[10] https://jwt.io/

- Signature created by taking the encoded first two parts and using the algorithm specified in the header along with a secret to cryptographically sign it.

JWT tokens usually aren't encrypted, meaning anyone can decode them and read their contents. The main purpose of the token is confirming that the token was not modified by the client or somewhere along the way to the client. For example, if an attacker wanted to increase his access rights by changing his user role to an admin role in the payload, they could modify the token and send it to the server. However, the signature computed by the server after receiving the modified token would no longer match the original signature contained in the third part of the token, and the server would reject the attacker access. Since the attacker does not have access to the secret used to create the original signature, modifying it is also not an option.

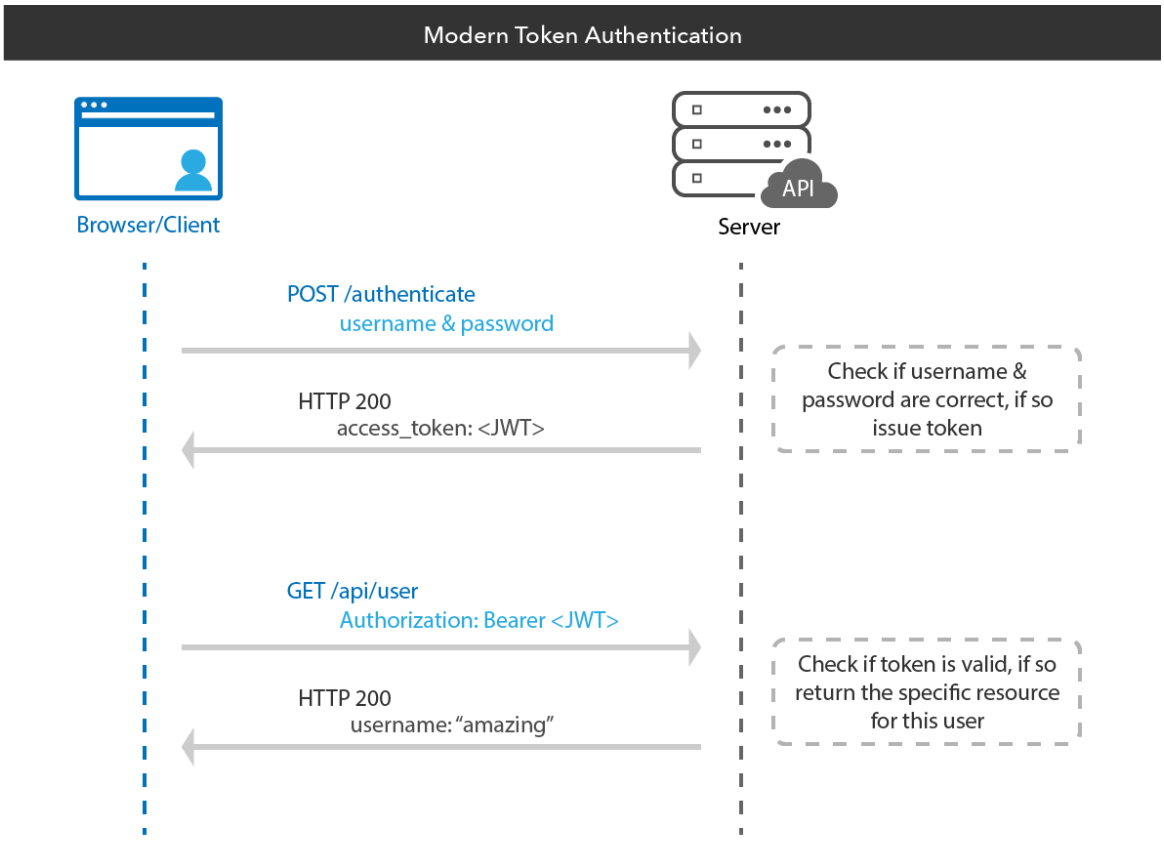


Figure 4.3: Token based authentication process[23]

4.3 React

Over the last decade, as the single page application approach became widespread, many frameworks and libraries used for their development began popping up. Some of the most popular frameworks include AngularJS, ReactJS, EmberJS, or Vue. All of these frameworks are maintained and updated regularly, and work to achieve the goal of facilitating the development of SPAs, so the choice came down to preference. ReactJS was the chosen framework, because of the author's previous experience with it as well as the desire to improve their knowledge of it.

ReactJS, more commonly referred to as React, is a JavaScript open-source library developed and maintained by Meta (formerly known as Facebook, rebranded in 2021).

React applications are made up of building blocks called components, which are reusable pieces of code rendered onto the screen[24]. While they may be compared to classes in other languages, they are in fact JavaScript functions that return HTML code comprised of HTML elements and/or other components. They can track and change their state using state variables, as well as pass data to child components by way of props[25]. Components re-render themselves on the screen whenever their state or props are updated.

A unique feature of React is its use of JSX, an extension of JavaScript syntax that allows us to combine HTML and JavaScript to write components. It really is just syntactic sugar which gets compiled into regular JavaScript on compilation.

4.3.1 React Router

As explained previously, navigation in single page applications does not involve loading different pages, instead using a single page and dynamically rewriting it with the necessary content. Because the user is always on a single page, there is no history of moving through pages and no changing of the URL, which introduces issues that negatively impact user experience. Because of the unchanging URL, it is not possible to link or bookmark specific parts of the application, and more importantly, makes navigation within the application using the back and forward browser buttons impossible. By pressing the back button, the average user expects to move back to the previous visited "page" in the application, but instead exits the whole application. Hitting the refresh button would also redirect them to the initial view of the application, regardless of their current position within the app.

React Router is the standard library used in React applications to emulate the desired behavior of navigation[26]. Its main feature is the use of the Route component, which takes a path string and a react component as inputs, and renders the component in case the specified path matches the current application URL.
It also allows us to manipulate the browser history and change the active URL by using links, or to do so manually in event handling functions. By changing the URL, the application triggers a re-render to the view that matches it and pushes the active URL to the browser history, which makes the back and forward buttons work as expected.

4.3.2 Responsive Design

Nowadays, websites and web application are accessed from a wide variety of devices, including desktops, tablets and most importantly, mobile phones. According to a study carried out in 2020[27], up to 68.1 percent of worldwide website visits came from mobile devices, which tells us it is no longer sufficient to design only for desktop use, websites need to adapt to screens of different sizes or risk losing customers.

Responsive design is a way of designing website and applications with this goal in mind[28]. It achieves this by employing CSS[11] techniques such as using relative size units, fluid or flexible containers, and media queries, which allow us to conditionally change the styling of different parts of the application, with the most commonly used condition being based on the device screen width. There are several available CSS frameworks that focus on responsive web design.

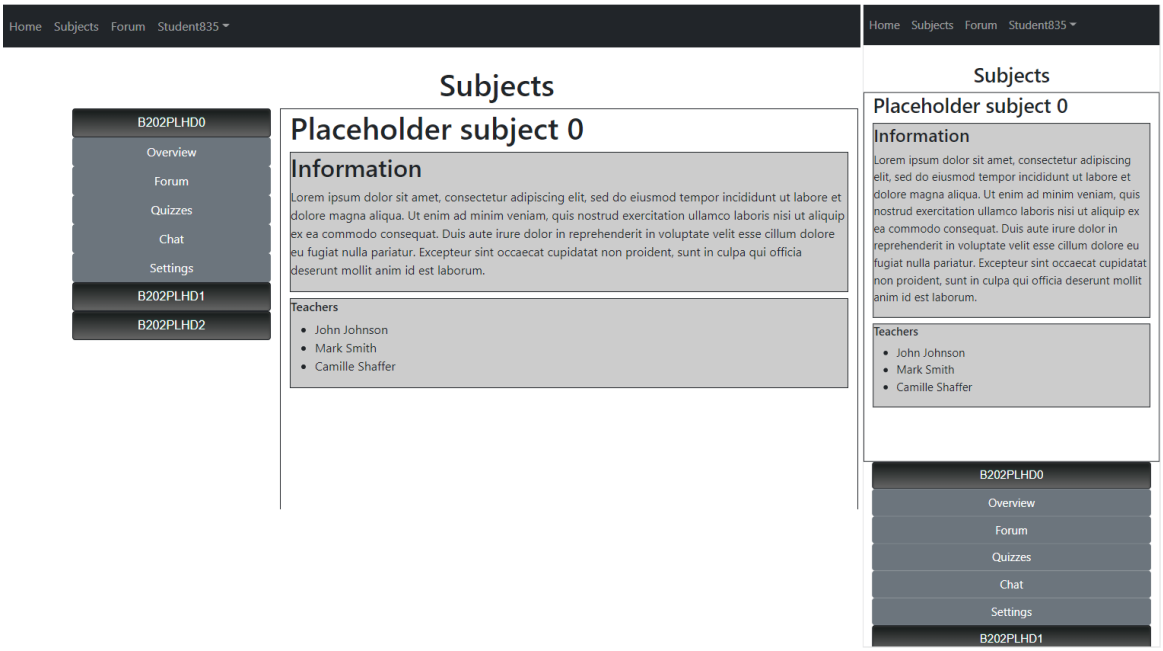


Figure 4.4: Comparison between the Subjects component rendered on a computer on the left, and on a phone on the right

4.3.2.1 Bootstrap

Bootstrap is an immensely popular open-source CSS framework that offers a vast amount of styles and components such as buttons, forms and tooltips[29]. Its main feature used for creating responsive websites is its grid system, which lets developers define their website's layout to make its content stretch, shrink or reposition itself to make the best use of the available space.

[11] Cascading Style Sheets

4.3.2.2 React-Bootstrap

React-Bootstrap is a relatively modern (released in 2019) React framework that incorporates Bootstrap into React applications[30]. While it offers the same features as Bootstrap (as it uses Bootstrap under the hood), its main benefit comes from the fact that it offers ready to use React components. This saves us from having to create these components from scratch by using regular HTML elements and applying Bootstrap styles to them, which leads to shorter and more readable component code.

This project mainly uses React-Bootstrap because of the readability and conciseness it provides, which can clearly be seen when creating complex components such as forms, but falls back to regular Bootstrap when applying simple styles is sufficient.

4.4 Security risks

As described in one of the non-functional requirements, the application must be able to resist common types of attacks against the application. The listed attacks were all part of the OWASP Top Ten over the last 10 years[31].

4.4.1 Injection attacks

Injection attacks consist of an attacker providing malicious input which, if not properly filtered by the application can expose critical data or damage the application[32]. SQL injections are a subset of injection attacks which focus on exposing vulnerabilities in the database access layer of the application. This is usually done through supplying unexpected inputs in the application frontend, which changes the meaning of the original database query if not properly validated. Queries provided by the `JpaRepository` interface from Spring repositories are properly parametrized, and therefore safe against SQL injection.

4.4.2 Cross site request forgery

"CSRF is an attack that tricks the victim into submitting a malicious request. It inherits the identity and privileges of the victim to perform an undesired function on the victim's behalf."[33]. It takes advantage of the data stored in the browser that gets sent automatically along with every request, such as credentials stored in the cookies. In that case, the application would have no way of differentiating intentional requests made by the victim, and unintentional ones created by this attack.
In our case, the server authorizes requests based on the Authorization header, which does not get sent automatically. It gets populated and sent with the request in event handler methods, which the attacker using CSRF does not have access to.

4.4.3 Cross site scripting

In cross site scripting, an attacker attempts to insert malicious code into an otherwise legitimate website or application[34]. If successful, by the time a victim visits the application,

the code, usually a browser-side script is already part of the application and therefore has access to information stored in the browser. The code injection usually happens through user input fields whose input is not properly validated.
In React, all potential user inputs embedded in JSX are escaped by default and converted into strings, which eliminates possibilities of injection in these fields. If a user were to input JavaScript code inside script tags, React would ignore the validity of the HTML element they provided and render their whole input as a string. This allows for strong protection against cross site scripting attacks.

4.5 Class Diagram

The class diagram uses the business domain model as a basis and enhances it with data types of entity fields and concrete entity relationship types.

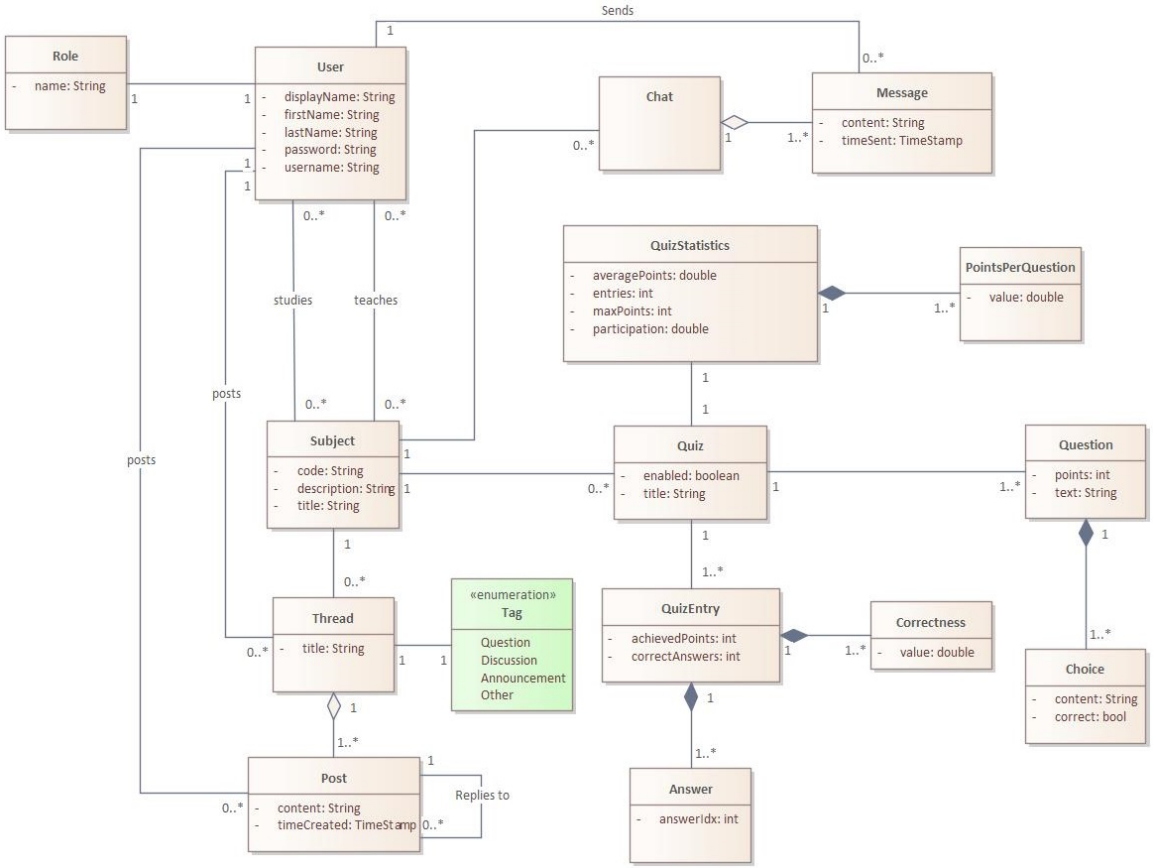


Figure 4.5: Class diagram

Chapter 5

Implementation

This chapter talks about the implementation details of the application, from its structure, to describing some of the key features with examples of code and application frontend views.

5.1 Structure

5.1.1 Backend

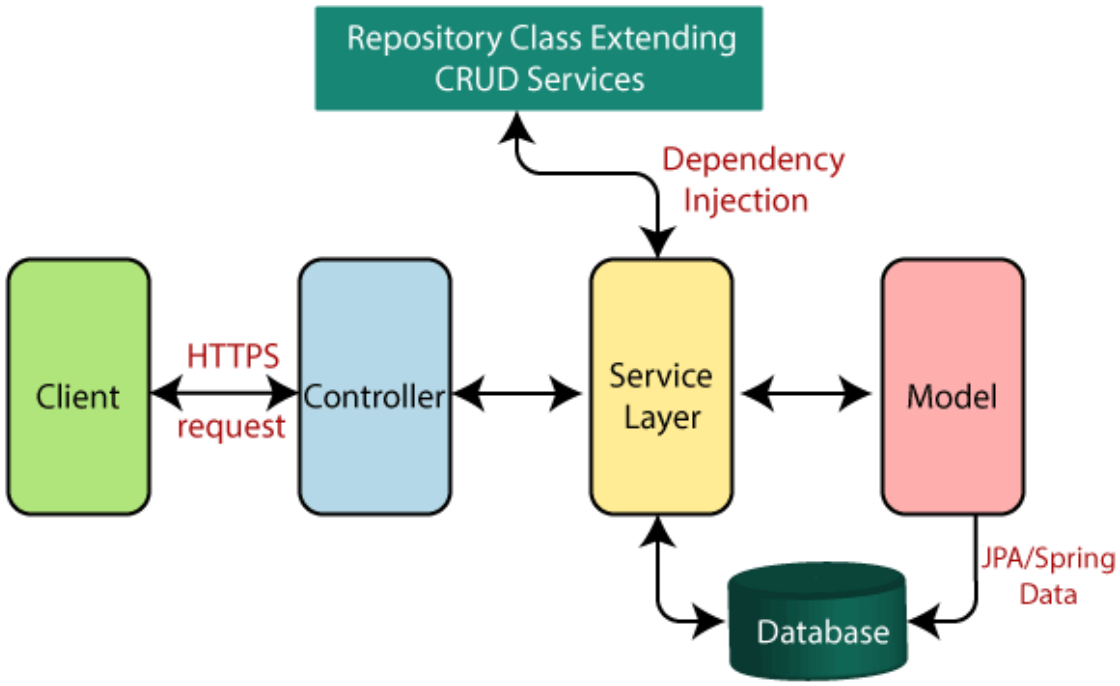


Figure 5.1: Backend application structure showcase[35]

The backend application is structured as a standard Spring multi-layered application, in which each layer makes use of the layer one level below it. The layers are as follows :

1. **Model**, which contains simple java classes, which are mapped into database tables through JPA. This layer contains no business logic, the classes are manipulated by the layers above it.

2. **Repository**, made up of classes implementing Spring's `JpaRepository`, which handle the persistence of objects contained in the model layer.

3. **Service** is a layer that contains the majority of business logic, its functions create, modify or delete the model objects and use the repository layer to persist these changes to the database.

4. **Controller**, which catches and handles HTTP requests by mapping them to specific functions, these functions extract the data contained within the request and pass it to the service layer.

There are other packages inside the backend structure that do not belong to the main layered architecture, but contain other vital code :

1. **Config**, which contain configuration classes for concerns such as security and web configuration.

2. **DTO** stands for data transfer object. These classes, like model classes, are simple java objects which only contain fields, getters and setters[36]. They are used to transfer information between the application and the web, while reducing the amount of data transferred, and more importantly exposing only the necessary information while doing so.

3. **Filter** which are used to intercept requests to our application and handle them if necessary. Currently there are only two filters which handle security in the form of authentication and authorization.

4. **Utils** classes contain utility function to be used in any part of the application. Right now there is only one utils class that is used to extract the currently logged in user from Spring's security context.

5.1.2 Frontend

The frontend structure is much more simple, as most of the files represent different react components and are separated into packages based on which application views they represent. For example, the forum and thread components belong in the Forum package.

5.2 Features

5.2.1 Login

As the majority of application features are only available to authenticated users, they need to log in if they wish to access them. On the frontend side, this is done by a simple login form, where the user inputs their email and password. On the backend side, the request

needs to get through the authentication filter that is attached to the `/login` URL, which attempts to authenticate the user. If successful, it returns the JWT access token in the HTTP response. This token is then stored in a cookie in the user's browser and attached to every subsequent request for authorization. Basic user information is also stored in the browser's local storage to save us from having to request it from the server every time we need to display it.

```
@Override
public Authentication attemptAuthentication(HttpServletRequest request,
        HttpServletResponse response) throws AuthenticationException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        log.info("Username is {}",username);
        log.info("Password is {}",password);
        UsernamePasswordAuthenticationToken authenticationToken =
        new UsernamePasswordAuthenticationToken(username,password);
        return authenticationManager.authenticate(authenticationToken);
        }

@Override
protected void successfulAuthentication(HttpServletRequest request,
    HttpServletResponse response, FilterChain chain, Authentication
    authentication) throws IOException, ServletException {
        User user = (User) authentication.getPrincipal();
        Algorithm algorithm = Algorithm.HMAC256("secret".getBytes());
        String accessToken = JWT.create().withSubject(user.getUsername()
            )
        .withExpiresAt(new Date(System.currentTimeMillis()+ longevity))
        .withIssuer(request.getRequestURL().toString())
        .withClaim("roles",user.getAuthorities().stream()
        .map(GrantedAuthority::getAuthority).collect(Collectors.toList()
            ))
        .sign(algorithm);

        Map<String,String> token = new HashMap<>();
        token.put("access_token",accessToken);
        response.setContentType(APPLICATION_JSON_VALUE);
        new ObjectMapper().writeValue(response.getOutputStream(),token);
        }
```

Listing 5.1: Functionality of the authentication filter, providing a JWT token on successful authentication

To be able to use Spring's `AuthenticationManager` class, we need to provide it an implementation of the `UserDetailsService` interface, which needs a `loadUserByUsername` function to work. This is easily done by having our `UserService` class implement this interface and use our repository layer to retrieve the user by email, which acts as the username in our application.

5.2.2 Student identity

Students' names are hidden to all other users while visiting the application. Instead, they are identified by a display name, which consists of a "Student" string followed by a randomly generated number from a range. This number can be changed by visiting their profile and generating a new identity. If a student chooses to generate a new identity, all of their previous contributions such as forum posts and chat messages will remain under the identity they had at the time of writing them.

To avoid the possibility of users abusing this function of generating new identities, they are limited to doing so once per day. This is achieved by having a function that resets their option to do so execute at midnight of every day, scheduled using a cron expression[1].

```
@Scheduled(cron = "0 0 0 * * *")
    public void enableChangingOfDisplayName() {
        List<User> users = userRepository.findAll();
        users.forEach(user -> {
            user.setCanResetDisplayName(true);
            userRepository.save(user);
        });
    }
```

Listing 5.2: Function scheduled with a cron expression that represents midnight

5.2.3 Joining subjects

An interesting issue that arises when we consider the required full anonymity of students is the way of them joining subjects while remaining anonymous. The standard way of a teacher inviting them by adding them one by one, or even providing a list of students is not a valid approach, as it would reveal the students' identities.

We decided to implement it by using invite links. A teacher can generate an invite link in a subject and share it with students either during a lecture or post it on the official learning portal of the university. The issue with this method is that any user of the application can join the subject if they get a hold of this link. This is solved by letting the teacher set a time of expiration for this link by choosing a value ranging from 1 minute to 1 week, after which the token expires and no longer gives access to the subject. They can also invalidate tokens prematurely. Expired links get removed automatically every day at midnight.

5.2.4 Forum

The design of the forum feature was heavily inspired by Reddit[2], which uses increasing indentation to signify replying in conversation chains. While posts being on the same horizontal level means that they are replying to the same post, increasing indentation shows the continuation of the reply chain. For easier traversal of the thread, the user is able to hide

[1] https://docs.oracle.com/cd/E12058$_0$1/*doc*/*doc*.1014/*e*12030/*cron*$_e$*xpressions.htm*
[2] https://www.reddit.com/

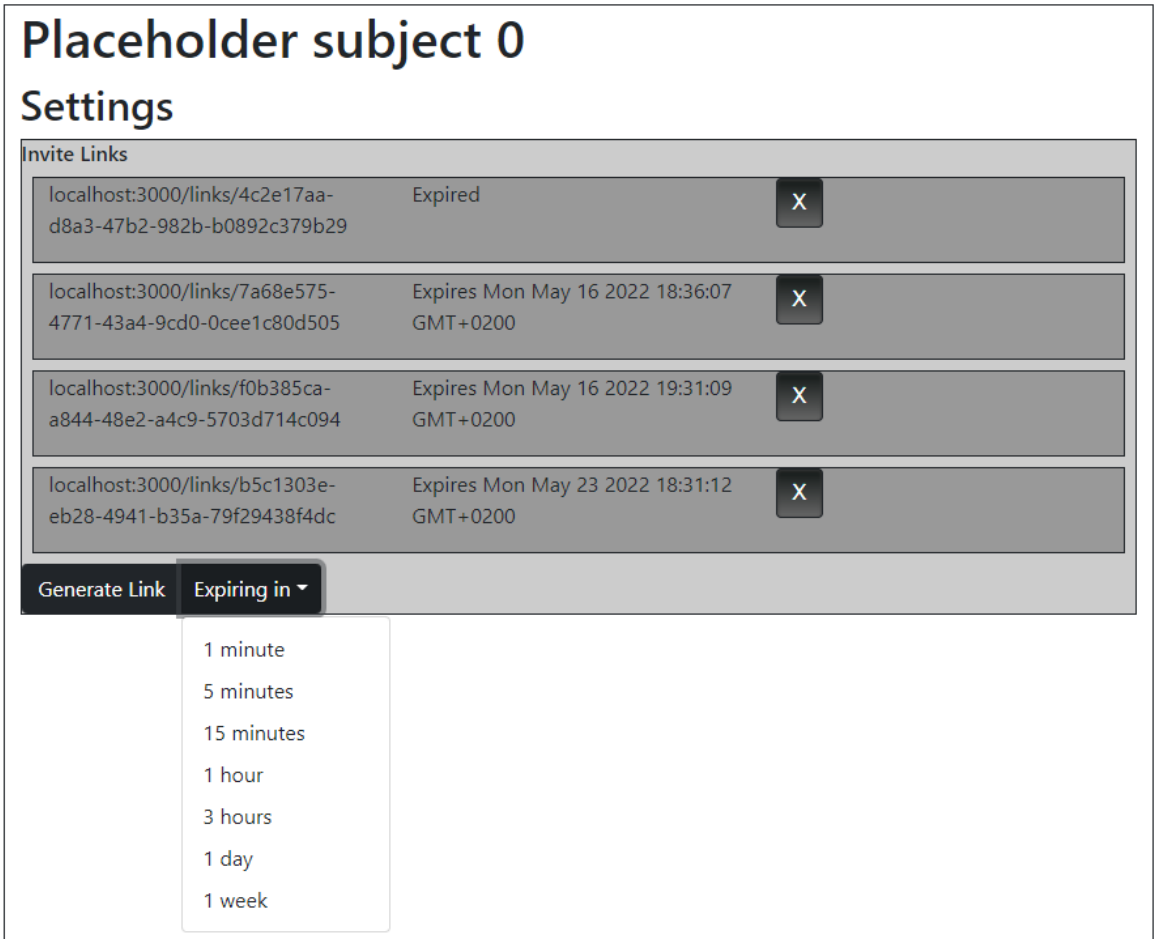


Figure 5.2: Showcase of a list of generated links for a given subject

all replies of a particular post by pressing the first button from the left as shown in figure 5.3, and load them back at a later point. The thread allows the user to edit and delete their own posts, and start new conversation chains.

5.2.5 Chat

The subject wide chat is implemented using websockets, which allows us to "open a two-way interactive communication session between the user's browser and a server"[37]. The main advantage of keeping an open connection is that the server is able to send information to the user without the user having to specifically request it first, as is the case communication over HTTP. When a user sends a message over websocket, the server first stores it in the database and relays it over to the other chat participants. When first loading up the chat screen, the browser sends a standard HTTP GET request to retrieve past messages, after which communication over websocket is used to receive new messages without having to reload the page.

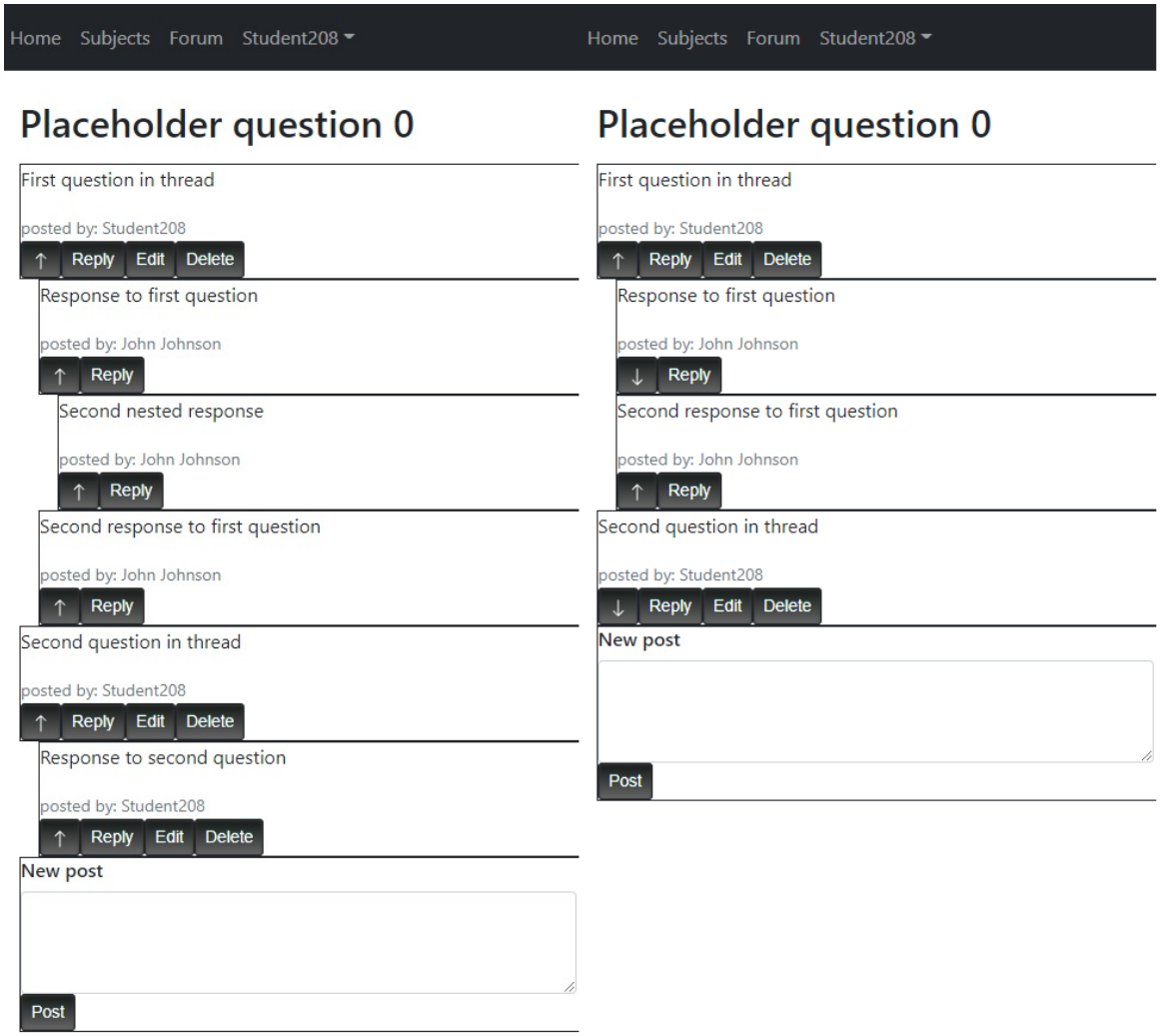


Figure 5.3: Example of the forum collapse replies feature. The picture on the left shows the original thread view, while the image on the right shows the same view with some of the replies hidden.

Any user belonging to the subject can open up a new chat conversation or participate in an ongoing one. The transmitted messages are simple object containing only the message content and the display name of its author. The only difference between messages between teachers and students is that the teachers' display names are highlighted in bold, as there is a higher chance of their contribution being valuable when answering questions, and therefore should be easily discernible from the students' messages.

5.2.6 Quizzes

The application provides a way for teachers to easily create multi choice quizzes for students to solve by way of the `QuizForm` component. As the questions and choices are part stored in React's state variables, every time a new question or choice gets added or deleted,

the state variable gets updated, making the component rerender so it always represents the current state of the quiz. After submitting the quiz, it becomes available to all students belonging to the subject.

Students can then solve the quiz and submit their entries to be evaluated. Although the evaluation of quiz entries and the generation of statistics for a given quiz are implemented in the backend, viewing them is not currently implemented in the application frontend.

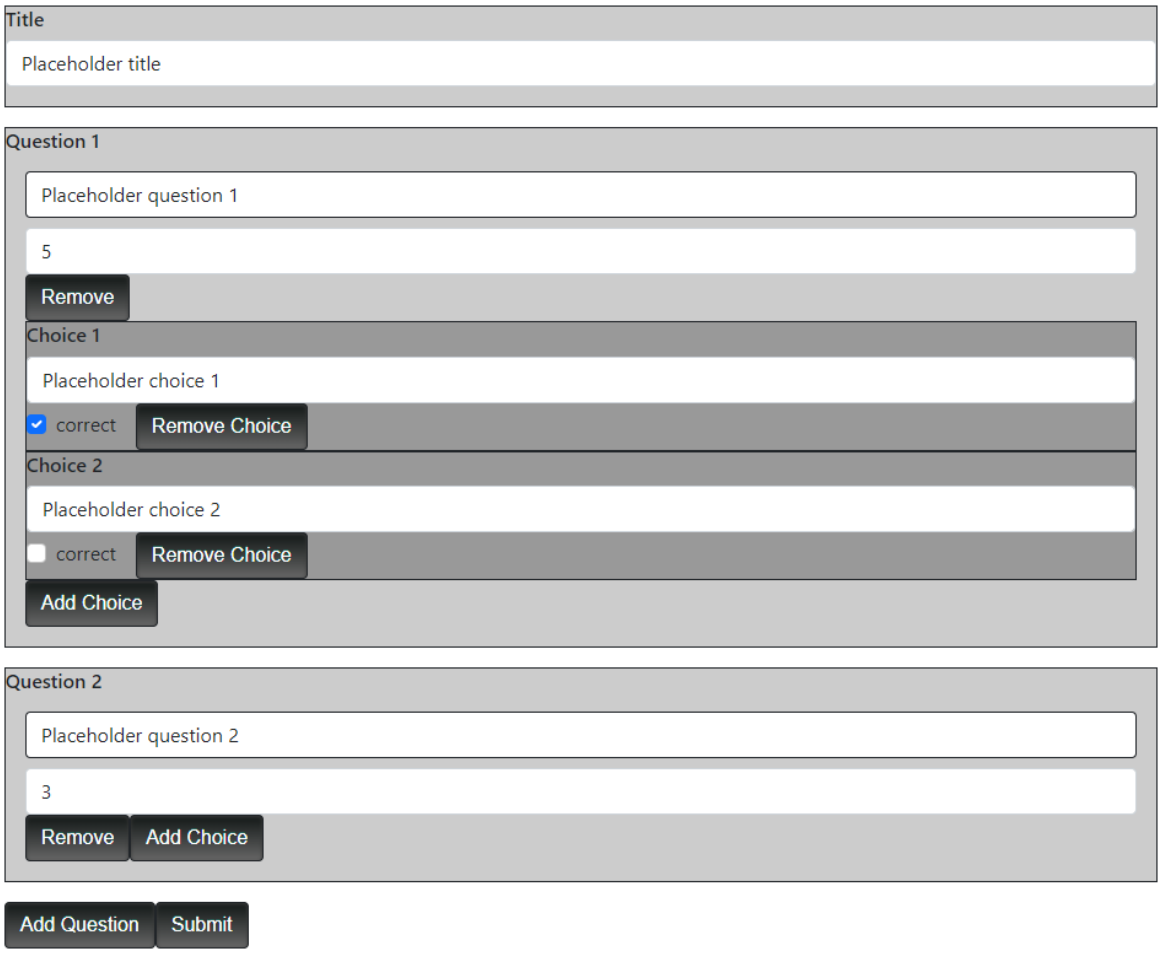


Figure 5.4: Showcase of the QuizForm component

Chapter 6

Testing

This chapter talks about the process of testing the application prototype, which consists of three parts:

- Testing of the service layer through unit and integration tests
- Testing of the application's API using Postman
- User testing

6.1 Service layer testing

Testing of the backend's functionality was focused on the service layer as it contains the majority of the application's business logic. By declaring the `spring-boot-starter-test` in our project, we gained access to previously mentioned frameworks JUnit, a framework used to write automated tests for java applications, and Mockito, which allows us to isolate the tested classes in unit tests by replacing their dependencies with mock objects with predefined behavior.

6.2 API testing

Postman is an application that provides an interface to easily create, execute and save HTTP requests. The creation of a request consists of inputting the URL of the API endpoint, specifying the request method, and optionally the body of the request, which contains data to be sent to the server in case of POST or PUT requests. It also lets us add custom headers to the requests, which was needed as most of the application is only accessible by providing an authentication token in the Authorization header.

Apart from helping uncover bugs in the code, this type of testing was useful during development by showing us the serialization of java classes into JSON objects in the response body, as well as helping configure different response codes, as the frontend depends on them to differentiate between the types of errors that could arise, such as sending a 409 conflict code when a user tries to register under an email that is already in use, or a 403 forbidden code when accessing a resource they do not have access to.

6.3 User testing

6.3.1 Deployment

For the application to be testable by users, we needed to provide them a way to access it. We decided to do this by deploying the application on the web by making use of cloud hosting services, specifically Heroku[1] and Netlify[2]

6.3.1.1 Heroku

Heroku offers hosting services for applications written in many languages, including Spring applications with PostgreSQL as their accompanied database engine. It offers a free hosting plan that, while limiting access to most of its features is sufficient for our use case. The deployment process was simple, all it required us to do was download its command line interface plugin, package our backend application into a jar file and run a command to deploy it on the cloud.

6.3.1.2 Netlify

Much like Heroku, Netlify offers cloud hosting with a free plan, this time focusing on websites and web applications. The deployment of our React frontend was even simpler, only requiring us to build the application and upload the resulting `build/` folder.

6.3.2 Test scenario

The test scenario was designed to cover all of the implemented core functionalities. It is made up of the following steps:

1. Register in the application
2. Log in to their created account
3. Use a provided expired invite link to try and join a subject, confirm the subject was not added to their list of subjects
4. Use a provided active link to join a subject, confirm it has added a subject to their list of subjects
5. Navigate to the subject forum
6. Create a new thread with an initial post
7. Reply to their post
8. Edit the reply and delete it afterwards
9. Navigate to profile and reset their display name, confirm it has been changed

[1] https://www.heroku.com/free
[2] https://www.netlify.com/

10. Reset their display name again, confirm it has not been changed due to the once per day limit
11. Navigate back to the created forum thread, confirm they can still edit and delete their post even after they have changed their display name
12. Navigate to subject chats, create a new chat conversation and write a message
13. Navigate to quizzes, solve a quiz and submit their entry

6.3.3 Testing process

Testing was carried out by two different testers, both of whom have some experience with web application development, one of them being a student of the Open Informatics master's degree programme at CTU FEE, and the other working as a software developer. They were given the specified test scenario and asked to give their opinions on the various tested functionalities as they went through it.

6.3.4 Results

When asked about the overall design of the application, both of the testers expressed similar thoughts in that while it was functional, it was too basic. This was expected, as not much time was left for graphic design by the time the core functionalities were implemented.

Another comment was made on the lack of features meant for editing user profiles, as right now, the only supported feature is the generation of a new display name, other features such as editing of the email address or password is not possible at the moment.

There were also issues of insufficient validation of various inputs fields in the application.

One of the testers discovered an oversight in the implementation of the reply chains in forum threads. Because replies in continuing chains of posts have increasing left padding as mentioned in section 5.2.4, this means that every following reply has less space to render than the previous, eventually making them hard to impossible to read. While this would take a very long reply chain to achieve on computer screens, it is much more likely to happen on smaller devices. The reason for this oversight is that most of the development time was spent looking at the computer screen version of the application and not considering the mobile version enough.
This issue can and will be fixed by implementing a way to select a post in the middle of the chain of posts, moving it to the left and only showing reply chains starting with this post.

Chapter 7

Conclusion

The goal of the bachelor thesis was to create an application that supports teaching and increases class participation by providing a way for students to anonymously with their teachers. The argument for the use of anonymity being that one of the prevailing causes of low class participation is the students' anxiety which prevents them from asking questions or participating in discussions. Hiding their identity alleviates this anxiety, which helps them participate more often and have a more positive experience. Further research on the effects of anonymity in a school setting was part of the semestral project preceding this thesis.

The process started by taking a look at current popular communication platforms used worldwide and highlighting key features that could be adapted for use in anonymous communication. These features and other useful ideas were then used as basis for our business analysis, which specified the requirements and functionalities that the application would support. Based on the business analysis, fitting technologies and frameworks were chosen and the application prototype was implemented. The application prototype was finally user tested, which helped reveal some issues with the application frontend.

Along with fixing the issues that were revealed thanks to user testing, future releases of the application will focus on finishing features specified in the functional requirements, such as the visualisation of generated statistics for specific quizzes and better editing of user profiles. Emphasis will be placed on improving the graphic design and interactivity of the application by adding notifications, tooltips and animations. New features, such as private chat conversations independent of subjects or file sharing will also be considered in the future.

Bibliography

[1] Microsoft Teams. *Introducing the simplified Microsoft Teams for Education experience.* URL: <https://support.microsoft.com/en-gb/topic/introducing-the-simplified-microsoft-teams-for-education-experience-fd5b0668-4156-4ce1-a51a-e6f54827973d> [online] [Cit. 14.04.2022].

[2] Microsoft Teams. *Welcome to Microsoft Teams.* URL: <https://docs.microsoft.com/en-us/microsoftteams/teams-overview> [online] [Cit. 14.04.2022].

[3] *Intellectual property protection.* URL: <https://moodle.fel.cvut.cz/course/view.php?id=5840> [online] [Cit. 14.04.2022].

[4] Moodle. *About Moodle.* URL: <https://docs.moodle.org/400/en/About_Moodle> [online] [Cit. 14.04.2022].

[5] MoodleCloud. *The fastest way to start teaching and training online.* URL: <https://moodle.com/solutions/moodlecloud/> [online] [Cit. 14.04.2022].

[6] *Advanced Community Server Setup.* URL: <https://support.discord.com/hc/en-us/articles/213530048-Advanced-Community-Server-Setup> [online] [Cit. 15.04.2022].

[7] *What is Discord?* URL: <https://discord.com/safety/360044149331-What-is-Discord> [online] [Cit. 15.04.2022].

[8] *Single-Page Application vs Multi-Page Application: Pros, Cons, and Which is Better?* URL: <https://lvivity.com/single-page-app-vs-multi-page-app> [online] [Cit. 15.04.2022].

[9] Daniel Cartland. *Common Single Page Application Crawling Issues How To Fix Them.* URL: <https://www.deepcrawl.com/blog/best-practice/spa-seo/> [online] [Cit. 20.04.2022].

[10] Ruud Hein. *How Many Users Have JavaScript Disabled.* URL: <https://www.searchenginepeople.com/blog/stats-no-javascript.html> [online] [Cit. 20.04.2022].

[11] *Spring Framework.* URL: <https://spring.io/projects/spring-framework> [online] [Cit. 20.04.2022].

[12] Prasanna Dhananjay. *Dependency injection: design patterns using spring and guice.* Simon and Schuster, 2009.

[13] Karsten Silz. *Snyk JVM Ecosystem Report 2021 Finds Increased Usage of Java 11 in Production*. URL: <https://www.infoq.com/news/2021/07/snyk-jvm-2021/> [online] [Cit. 21.04.2022].

[14] *Spring Boot*. URL: <https://spring.io/projects/spring-boot> [online] [Cit. 25.04.2022].

[15] Matthew Tyson. *What is JPA? Introduction to the Java Persistence API*. URL: <https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html> [online] [Cit. 25.04.2022].

[16] *Working with Spring Data Repositories*. URL: <https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html> [online] [Cit. 25.04.2022].

[17] Okta. *Authentication vs. Authorization*. URL: <https://www.okta.com/identity-101/authentication-vs-authorization/> [online] [Cit. 25.04.2022].

[18] Spring. *The Security Filter Chain*. URL: <https://docs.spring.io/spring-security/site/docs/3.0.x/reference/security-filter-chain.html> [online] [Cit. 30.04.2022].

[19] mdn web docs. *HTTP authentication*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication> [online] [Cit. 30.04.2022].

[20] OWASP. *Session Management Cheat Sheet*. URL: <https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html> [online] [Cit. 30.04.2022].

[21] Fortinet. *Authentication Token*. URL: <https://www.fortinet.com/resources/cyberglossary/authentication-token> [online] [Cit. 30.04.2022].

[22] JWT. *Introduction to JSON Web Tokens*. URL: <https://jwt.io/introduction> [online] [Cit. 03.05.2022].

[23] Macy Ngan. *Modern Token Authentication in Node with Express*. URL: <https://developer.okta.com/blog/2019/02/14/modern-token-authentication-in-node-with-express> [online] [Cit. 03.05.2022].

[24] React. *Tutorial: Intro to React*. URL: <https://reactjs.org/tutorial/tutorial.html> [online] [Cit. 09.05.2022].

[25] React. *Components and Props*. URL: <https://reactjs.org/docs/components-and-props.html> [online] [Cit. 09.05.2022].

[26] React Router. *Tutorial*. URL: <https://reactrouter.com/docs/en/v6/getting-started/tutorial> [online] [Cit. 09.05.2022].

[27] Eric Enge. *Mobile vs. Desktop Usage in 2020*. URL: <https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage> [online] [Cit. 10.05.2022].

[28] *Responsive design*. URL: <https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design> [online] [Cit. 12.05.2022].

[29] Bootstrap. *Introduction*. URL: <https://getbootstrap.com/docs/4.1/getting-started/introduction/> [online] [Cit. 12.05.2022].

[30] React-Bootstrap. *Introduction*. URL: <https://react-bootstrap.github.io/getting-started/introduction> [online] [Cit. 12.05.2022].

[31] OWASP. *OWASP Top Ten*. URL: <https://owasp.org/www-project-top-ten/> [online] [Cit. 05.05.2022].

[32] René Milzarek. *Injection Attacks Types and How to Best Protect Your Web Apps?* URL: <https://crashtest-security.com/what-are-the-different-types-of-injection-attacks/> [online] [Cit. 05.05.2022].

[33] OWASP. *Cross Site Request Forgery (CSRF)*. URL: <https://owasp.org/www-community/attacks/csrf> [online] [Cit. 05.05.2022].

[34] OWASP. *Cross Site Scripting (XSS)*. URL: <https://owasp.org/www-community/attacks/xss/> [online] [Cit. 05.05.2022].

[35] *Spring Boot Architecture*. URL: <https://www.javatpoint.com/spring-boot-architecture> [online] [Cit. 14.05.2022].

[36] Okta. *Data Transfer Object DTO Definition and Usage*. URL: <https://www.okta.com/identity-101/dto/> [online] [Cit. 14.05.2022].

[37] *The WebSocket API (WebSockets)*. URL: <https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API> [online] [Cit. 16.05.2022].