HUMANLAB

Bring ideas to life
VIA University College

TRIC! - The Right Choice

Project Report

Students:

Bogdan-Alexandru Mezei (293137)

Daria-Maria Popa (293087)

Natali Munk-Jakobsen (293132)

Supervisor:

Henrik Kronborg Pedersen

Number of characters: 105103

Software Engineering

7th Semester

Date: 14/12/2022

Table of contents

List of figures and tables

Abstract

Daria-Maria Popa

This paper proposes the idea of a voting tool to be used in collaboration with a theatre play, putting forward the concept of augmented democracy and how individual votes can affect the lives of many.

The introductory chapter delves into the concept of augmented democracy and digital twinning, describing the main company stakeholder, their wishes and delimitations for 'TRIC - The Right Choice', and thoroughly underlining the reasoning behind the presented concept - making the audience aware of the possible future of democracy in the western world and its consequences through the use of a new tool capable of mimicking the user's behavior and acting as its own entity.

The Analysis investigates the problem domain and concludes the list of domain objects and contracts presenting their intended interactions in the system, as referenced by the actors of the system.

Design takes over the analyzed problem domain and present the main technical choices of the system, defined as a client-server voting system using single-page front-end architecture and a three-layer back-end, and the reasoning behind them in relation to the stakeholders' requirements.

The paper succinctly shows the implementation process and result of applying the resulted system design, while highlighting important elements of the resulted system, their scope and role in reaching the aim of the project, with the testing chapter displaying, through the use of several testing methods, how successful the implementation of the system was.

Lastly, the paper concludes the results of the project and presents its status as accepted by the main stakeholder, the company, managing to achieve the set requirements and be successfully used in a real-world scenario.

1 Introduction

Daria-Maria Popa

Democracy has developed across the world since medieval times and exists as the foundation of the Western world today (Roser M. and Herre B., 2013, Solijonov A., 2016). Although access to democratic rights and participation in voting has become easier compared to the past, statistics show that the rate of participation in democratic processes is gradually decreasing (Roser M. and Herre B., 2013, Solijonov A., 2016).

The theatre has been one of the main means for holding democratic debates since ancient Greek and Roman times (Morris, 2014), with its effect on culture and education being a constant and acknowledged factor of artistic plays even in modern times. Numerous people have investigated the relationship between theatre and democracy, with an important note being that "Democracy and theatre evolved at the same time, and the two were intertwined." (Morris, 2014).

In modern displays, the relation between theatre and democracy is most commonly shown through the use of interactive theatre (Bucher, 2018), either implicating the audience as characters or making use of modern technology to provide them with a decisional role in certain scenarios, for example when solving a crime (Mystery of Edwin Drood | Gonzaga University, 2019).

The advance in technology brought innovations not only in theatre but in all aspects of life, including providing solutions for the lack of participation and awareness in the democratic process. Electronic voting trials have been implemented in several countries all over the world (IDEA, 2016), and multinational organizations, states, and technology companies have started collaborating in their use of technology for protecting and improving human rights and democracy (The Tech for Democracy initiative, 2021).

However, digital technologies still have the potential for providing new possible implementations of democracy in the real world. Augmented democracy is the idea of using virtual representations of people to allow them to participate in democratic decision-making, where personalized AI representations, also referred to as "digital twins", can be used to expand the ability of people to engage in democratic debates (Hidalgo C., 2018).

HumanLab is a professional theatre company that showed interest in the relationship between interactive theatre and, specifically, augmented democracy. Based in Horsens, Denmark, HumanLab involves both performers and professionals in the fields of performing arts, cognitive science, engineering, physiotherapy, and anthropology, and its poetic visual storytelling is an expression of the constant dialogue between tradition and innovation (About HUMANLAB, 2022).

HumanLab proposes TRIC ("The Right Choice"), a project that aims to dive deeper into the concept of augmented democracy through a theatre play whose objective is to raise the audiences' awareness about the crisis that democracy is going through and the civic responsibility everyone has regarding the democratic process and the impact of technology in it (About HUMANLAB, 2022).

The project's purpose is to ask "What is in store for the future of democracy?" (About HUMANLAB, 2022) while highlighting the existing socio-political status quo, a highly relevant topic in the current political climate which is facing a worldwide crisis concerning the participation of the public within the democratic process (Solijonov A., 2016).

Through the use of the theatre play and the 'digital twin' concept, TRIC's objective is to allow the audience to interact directly with the artistic performance and raise awareness regarding the risks of using technology within the democratic process.

To retrieve relevant information from the audience and maintain the conversation on the topic at hand (Augmented democracy), HumanLab proposes through TRIC a predetermined set of topics with a fixed set of answers each representing a tuple of political philosophies presentable on the audience's personal mobile devices, ensuring voter anonymity in the retrieved and persisted data after the play, and limiting the demographic of the play to the company's general theatre audience (About HUMANLAB, 2022).

Overall, the need to raise public awareness on the matter of democratic processes is of invaluable need in current times (Solijonov A., 2016) and HumanLab's idea of creating an augmented space for democratic debates through the use of theatre play will be thoroughly analyzed in the following chapters of this report.

2 Analysis

Daria-Maria Popa

What project TRIC offers is creating a voting tool that would support the creative presentation of the influence of an augmented democracy and its possible effects in the actual world, as a way to incentivize and inspire the audience to recognize their civic duty as members of a democratic system.

The problem domain focuses on the relationship between the population and the elective system, specifically the democratic elective system, in a novel electoral environment represented by a theater play with augmented democracy at its core.

The augmented democracy system in the theater play makes use of the digital twin concept to relieve the democratic process participant, represented by an audience member, of the need to have an active presence in this process, opting instead to have their “ digital twin” take their place.

To achieve its purpose as described by the stakeholders, the TRIC voting tool must present the set conversation topics and their answers to the audience and have a way for the audience to interact with the said topic and voice their personal opinions in a way that simulates vote results being released to the public in the real world, all while integrating technology in the play as a means for their digital twin to coexist and become an active participant in the discussion.

The above statements represent the basis for defining the requirements that a project must adhere to in order to fulfill the stakeholders’ vision for TRIC.

2.1 Requirements

In order for the requirements to be created, the people and entities that interact with the system and have their goals fulfilled by it were identified. For TRIC, these primary actors (Larman, 2004) of the system were identified as the audience members, fulfilling the role of users of the system, and the admins, represented by the personnel in charge of managing the flow of actions during the play.

For the users, their goals with the system are described by their want to interact with the play and influence its outcome, as well as the want to see their direct impact on the play.

Similarly, for the admin, their goals of the system are enabling and encouraging the users to join and interact with the play and providing ways for them to do so through their use of the system.

Besides the primary actors, the secondary actors of the system are defined as the Company (HumanLab) and the Theatre Crew, represented by the people on stage creating the theatre play. These secondary actors (Larman, 2004) do not interact directly with the system, but provide and use information to and from the system.

In order to ensure complete and qualitative requirements, the ISO/IEC 25010:2011 standard was used (Britton, 2021) by following the eight quality characteristics when constructing the requirements. These characteristics are Functional Suitability, Usability, Reliability, Performance Efficiency, Security, Compatibility, Maintainability, and Portability (abbreviated from now on as FURPSCMD).

To represent the functional features of the system and the Functional Suitability (abbreviated as 'F') characteristic from the FURPSCMD standard, a list of functional requirements was created. Each individual functional requirement was written following the Connextra user stories format 'As a…, I want… so that…' (Cohn, 2019), for its readability and focus on the 'Who', 'What', and 'Why' of each functionality. The non-functional requirements focus on covering the rest of the quality characteristics, the URPSCMD.

Both the functional and non-functional requirements lists follow the SMART principles (Faus, 2021) to ensure good future testability and are ordered by priority, the first element in each list representing the most important requirement for the system.

The prioritized list of functional requirements is as follows:

1. As a user, I want to be able to cast my vote so that I can influence the outcome of the play

15. As an admin, I want to be able to start the play so that I can allow the users to join and be a part of the play.

2. As a user, I want to be able to select a profile picture and add a username so that I can better personalize my profile for the play

3. As an admin, I want to be able to start voting periods for questions so that I can ensure a good flow of the play

4. As a user, I want to be able to see the voting results after each voting period so that I know which story branch was chosen by the majority

6. As an admin, I want to be able to send the last question along with the predicted answer for each user so that I can demonstrate the impact of a digital twin

7. As an admin, I want to be able to manually stop voting periods for questions so that I can ensure no downtime in the play

8. As an admin, I want to be able to manage the list of questions so that the performers can keep the storyline up to date

9. As an admin, I want to be able to show the current question and its results to the actors so that I can inform them of the next decision in the play

10. As a user, I want to be able to see my final voting profile so that I can have an overview of my choices at the end of the play

5. As an admin, I want to be able to set a timer for vote periods so that the voting length can be customized to fit the play dynamic

11. As a user, I want to be able to download my voting results so that I can have a digital souvenir of the play which I can share

12. As an admin, I want to be able to end the play and save the voting information so that only I have access to the voting system and voting results at the end of a play

13. As a user, I want to be able to see the list of contributors of the app so that I can be better informed about the play

14. As an admin, I want to be able to edit the list of contributors of the app so that it can be kept up to date

16. As an admin, I want to start a countdown so that I can control the timing of the play for the actors

17. As a user, I want to be able to give my consent regarding the processing of my data so that I can make an informed choice about joining the play

18. As an admin, I want to hide the results of the previous question so that I can ensure the audience's attention will return to the play

The prioritized list of non-functional requirements is as follows:

1. The theatre play must be held in locations with good cellular reception

2. The vote results must be saved at the end of the play

3. Persisted vote results must not contain any user data

4. The user data must be cleaned after each play

5. The users' future answers must be able to be predicted based on previous answers

6. The users' final vote results must be downloadable in an image format

7. Timers must be measured in seconds

8. Answer categories must be chosen from the Pragmatic - Idealist and Conservative - Progressive dichotomies

9. Same question answers must have opposite categories

10. Any Admin system interactions must be recorded locally and recoverable in case of system refresh or crash

11. User data must be recoverable in case of system refresh or crash

12. Play management options must require authentication

13. The system must be accessible on all ranges of mobile devices and their operating systems

14. The system must be able to handle the whole theatre audience as simultaneous users

2.2 Functional Requirements

As aforementioned the main actors of the system are the user and the admin. These actors are the driving factor for the functional requirements, also referred to as user stories (Cohn, 2019), but also for the use cases of the system, which focus on describing the behavior of the system (Informal Semantics for UML Use Case Diagrams, 2019).

A use case provides a detailed explanation of how the primary actors will use the system, focusing on how the system behaves in response to their requests (Larman, 2004).

For this project, a total of four use cases have been identified based on the 18 functional requirements. These use cases can be represented using a use case diagram (Figure 2.1) which depicts the contract for how the system should operate, in a visual manner (Cockburn, 2000).

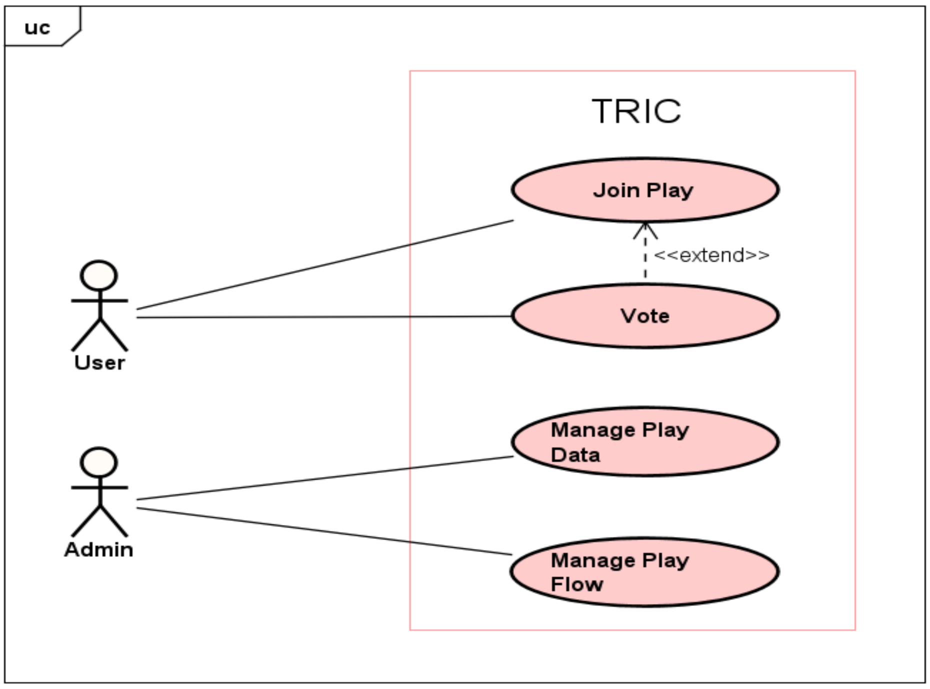


Figure 2.1. Use Case Diagram

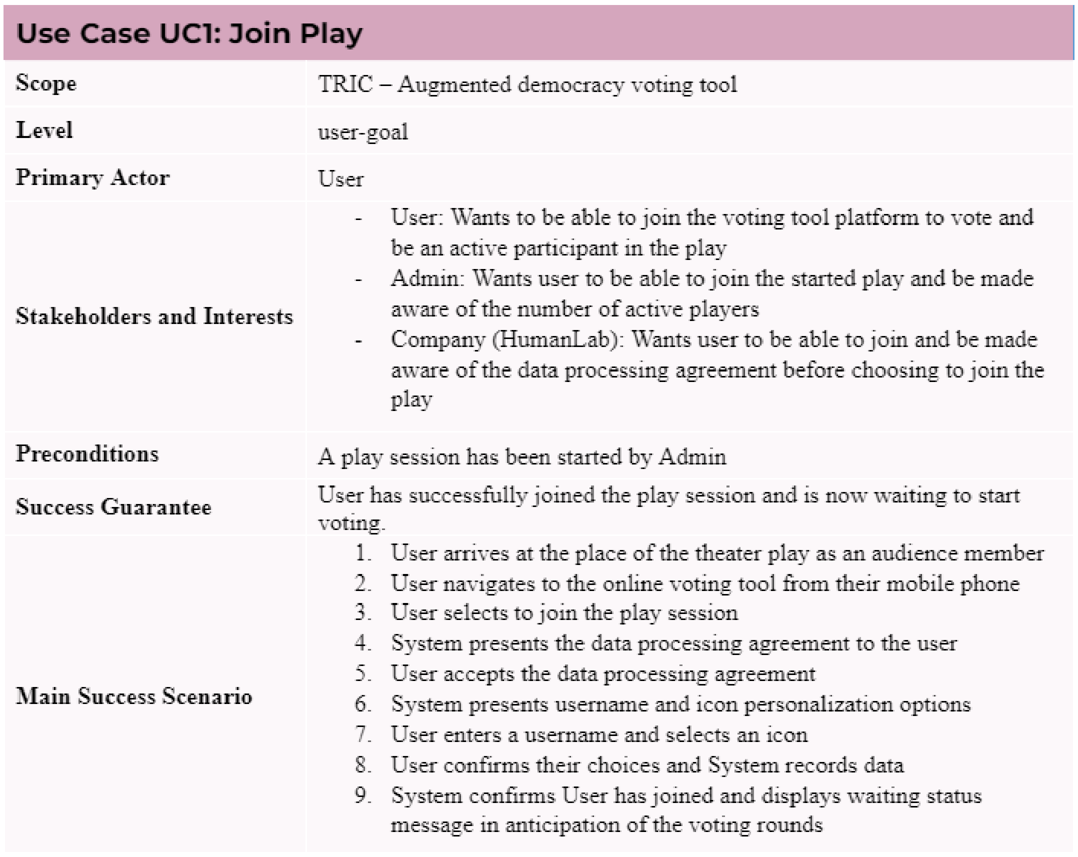
The above image depicts the use case structure of this system, with the two primary actors, the user and the admin, and their two use cases each, 'Join Play' and 'Vote', respectively 'Manage Play Data' and 'Manage Play Flow'.

The use cases associated with the User present an 'extend' relationship from the 'Vote' to the 'Join Play' due to the nature of the 'Vote' use case being fully dependent on the User actor having previously joined the play.

The 'Manage' use cases associated with the Admin actor, highlight again the role of the Admin as the actor controlling both the data shown and the overall integration of the voting tool with the artistic display.

To further illustrate the expected system behavior (Larman, 2004), each use case was further defined with the help of fully dressed use case descriptions (Appendix C - Use Case Descriptions). A partial example of this can be seen in Figure 2.2, for the 'Join Play' use case. The scope delimits the system whose use is being described, while the 'user-goal' level classified the use case as one related to the goals of a primary actor, mentioned in the 'Primary Actor' section (Larman, 2004).

The following sections describe, in order, what are the wanted system behaviors as told by the stakeholders, what should be true upon starting the use case and then upon successfully finishing the use case, and what that successful flow is.



Use Case UC1: Join Play	
Scope	TRIC – Augmented democracy voting tool
Level	user-goal
Primary Actor	User
Stakeholders and Interests	- User: Wants to be able to join the voting tool platform to vote and be an active participant in the play - Admin: Wants user to be able to join the started play and be made aware of the number of active players - Company (HumanLab): Wants user to be able to join and be made aware of the data processing agreement before choosing to join the play
Preconditions	A play session has been started by Admin
Success Guarantee	User has successfully joined the play session and is now waiting to start voting.
Main Success Scenario	1. User arrives at the place of the theater play as an audience member 2. User navigates to the online voting tool from their mobile phone 3. User selects to join the play session 4. System presents the data processing agreement to the user 5. User accepts the data processing agreement 6. System presents username and icon personalization options 7. User enters a username and selects an icon 8. User confirms their choices and System records data 9. System confirms User has joined and displays waiting status message in anticipation of the voting rounds

Figure 2.2. Join Play Use Case Description

The created fully dressed use case descriptions also include all the possible branching from the main scenario, the related non-functional requirements, any variations on how to perform the steps, how often the use case is expected to be triggered in the real-life system, and any other miscellaneous notes (Appendix C - Use Case Descriptions), adhering to the format set by A. Cockburn (Larman, 2004).

In order to visualize the behavior detailed by the use case descriptions as a progression of actions in the system, the use of activity diagrams was employed following the UML 2.5 standard (Fakhroutdinov, 2022). For the ‘Join Play’ use case described above, the sequence of actions follows as shown in Figure 2.3.

The box surrounding the main flow of the activity diagram displays the “At any time, the User exits and rejoins the play session” extension flow which acts as a parent activity (super-activity) for the rest of the flow (Fakhroutdinov, 2022). The various decision nodes help show how the alternative flows of the use case interact with the main flow and where in the timeline they can happen.

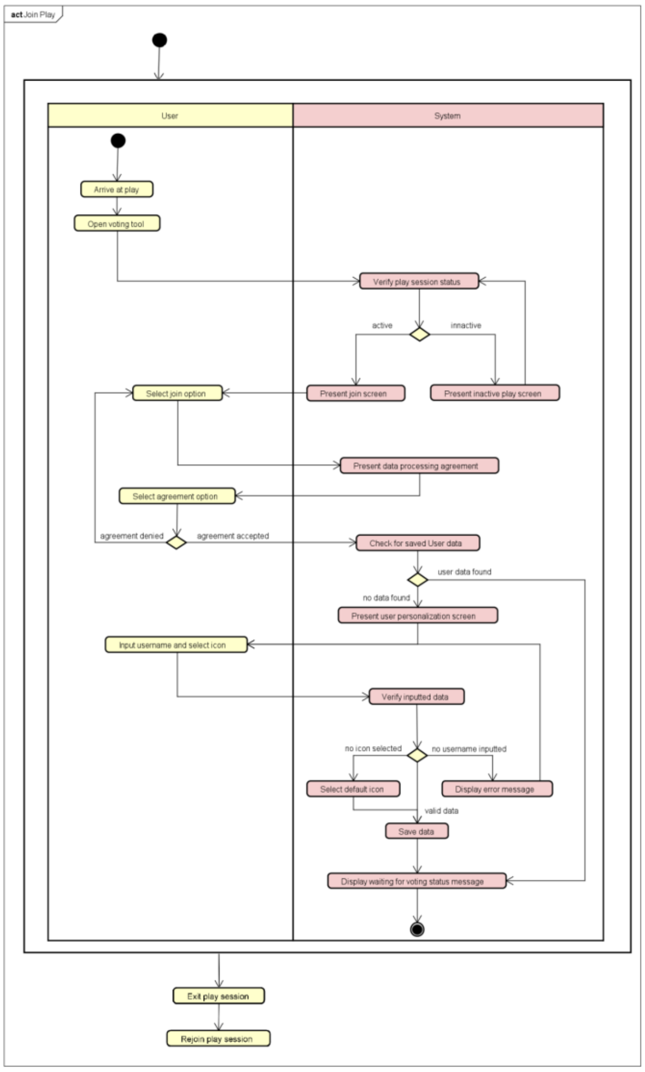


Figure 2.3. Join Play Activity Diagram

To visualize the particular scenario in the flow of the use case when the User tries to join the app, a sequence diagram was made Figure 2.4 using the same UML 2.5 standard (Fakhroutdinov, 2022). The sequence diagram helps model the constraints and loops that are parts of this scenario, and therefore better understand the intended system behavior.

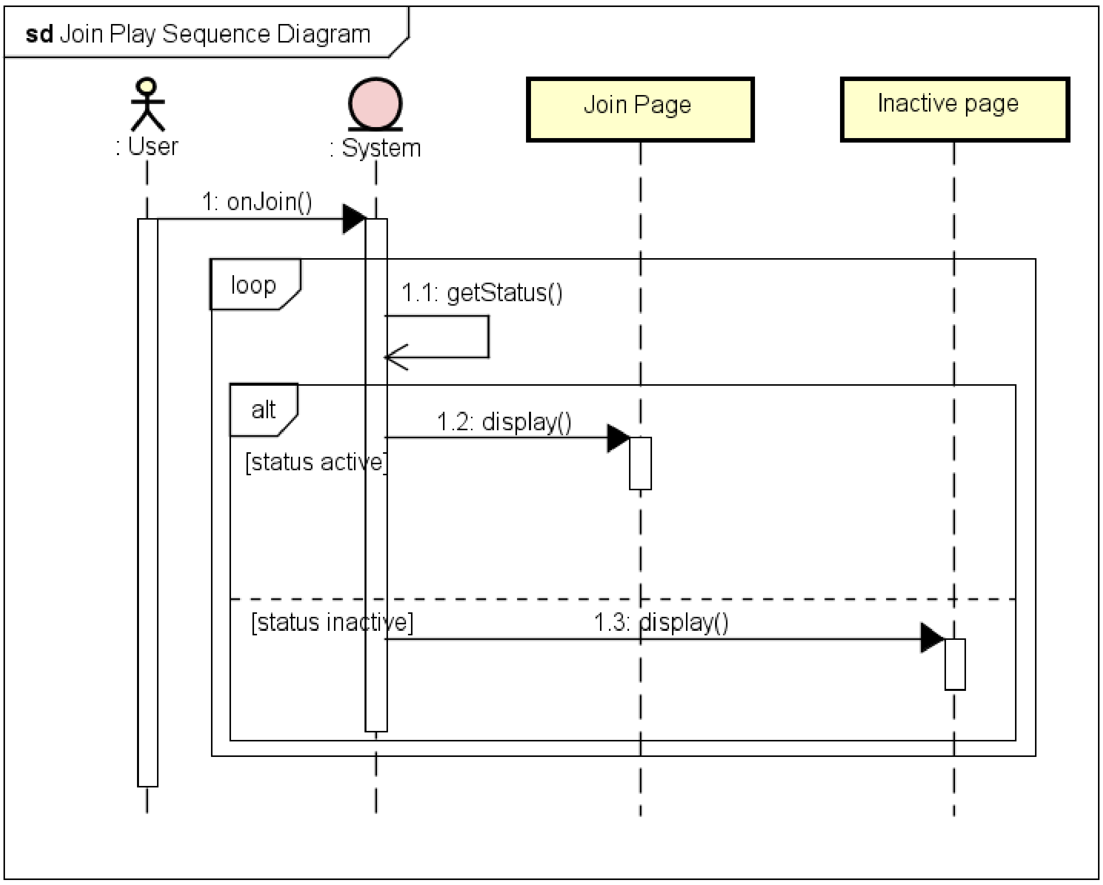


Figure 2.4. Join Page Sequence Diagram

Altogether, the above diagrams are useful tools for outlining and comprehending the intended behavior of the system, as described through the perspective of the main actors of the system and defined by the user stories.

2.3 Non-Functional Requirements

Compared to the functional requirements, the non-functional requirements do not describe the system's behavior, but set constraints through different quality attributes and specifications. These constraints refer, as aforesaid, to the Usability, Reliability, Performance Efficiency, Security, Compatibility, Maintainability, and Portability of the system and software, according to the ISO/IEC 25010:2011 standard (Britton, 2021).

Regarding the reliability of this system, the 'The theatre play must be held in locations with good cellular reception', 'Any Admin system interactions must be recorded locally and recoverable in case of system refresh or crash', and 'User data must be recoverable in case of system refresh or crash' non-functional requirements were created. These represent some of the key non-functional requirements of the system due to the importance of the voting tool having good availability and fault tolerance (Britton, 2021) in order for it to aid the artistic display, not distract from it, as mentioned by the stakeholders.

The security of the system is referenced through the 'Persisted vote results must not contain any user data', 'The user data must be cleaned after each play', and 'Play management options must require authentication' requirements. The first two requirements are related to the GDPR rules for processing personal information (Intersoft Consulting, 2016), while the last non-functional requirement aims to maintain the integrity of the system and underlines the difference between the User actor and Admin actor in terms of authentication and authorization rules.

The 'The users' final vote results must be downloadable in an image format', 'Timers must be measured in seconds', 'Answer categories must be chosen from the Pragmatic - Idealist and Conservative - Progressive dichotomies' and 'Same question answers must have opposite categories' non-functional requirements ensure the usability of the system, by focusing on providing a shallow learning curve, good accessibility, and user error protection, again relating to the stakeholders' intention of having an easy-to-use tool.

For good system maintainability, the 'Vote results must be saved at the end of the play' and 'The users' future answers must be able to be predicted based on previous answers' non-functional requirements were created. These enforce the stakeholder need for gathering and processing data from the plays, as well as having answers be predictable and using that to create 'digital twins'.

For the portability of the system, it was emphasized by the company stakeholder that 'The system must be accessible on all range of mobile devices and their operating systems', ensuring that any member of the audience could choose to join, and 'The system must be able to handle the whole theatre audience as simultaneous users' requirement noting the capacity performance efficiency that the system must maintain in that case.

As the voting tool is described by the stakeholders as its own standalone system, no related compatibility non-functional requirements were created.

Overall, the non-functional requirements of the system ensure through their use of the ISO/IEC 25010:2011 standard a qualitative and testable system that follows the stakeholders' interest.

2.4 Domain model

The functional and non-functional requirements outline the domain objects that exist and interact in the problem domain.

One of the main objects is the user, represented by the audience of the theater play. The user is the one whose decisions and biases are observed for their digital twin to be created and eventually replace the original user in the voting process. The user wants to have an active presence and interacts with the play by joining it and casting their votes, therefore making it one of the actors of the domain model.

Another domain object is the admin, whose main role is that of a moderator for the play. As a moderator, the admin also directly interacts with the play, making it the second actor of the domain model. The need for a moderator stems from the need for timing in the way the topics, voting times, and results are being displayed to the user, a critical step in making the audience feel immersed in the play.

The play is the third domain object, being the representation of the environment described by the problem domain. The play is one moderated by the admin and joined by the user and serves as the channel through which the democratic debate can take place.

Another important part of the domain is the questions. The question domain object is the visual representation of the topics to be discussed during the play by the user. For the users to express their opinions on certain questions, the answer object is needed. The answer object represents the possible choices any user may make, similar to the choice of a candidate on a ballot.

As expressed in the domain limitations, a question may have only two answers and the user may choose only one of them, the answers being a separate domain object of their own. For the answers to help delimitate and build the digital twins of users, each answer was delimited by the stakeholders as holding two categories, a primary and a secondary one, each representing a different political philosophy (i.e. Pragmatism is a political philosophy).

Finally, to represent a user's choice on a certain question that has a set number of possible answers, the vote domain object was identified as the last object which helps define the problem domain and the core object in making TRIC a voting tool.

A domain model was created to visually depict how these real-world objects interact with one another (Larman, 2004). The domain model that follows provides a "visual dictionary" (Larman, 2004) of the concepts and ideas presented in the background description.

As depicted in the domain model, most of the domain objects are described by certain attributes which originate from the problem domain analysis as well.

The user and admin are both represented in the problem domain of a voting tool by their username, with the normal users also having a profile image as their visual representation in the play, while the admin lacks the need for such a representation in the play and has instead a password attribute due to expressed security reasons by the stakeholders.

The play domain object is represented in the domain model by its status, general play information, and contributor information which defines the people responsible for creating the play.

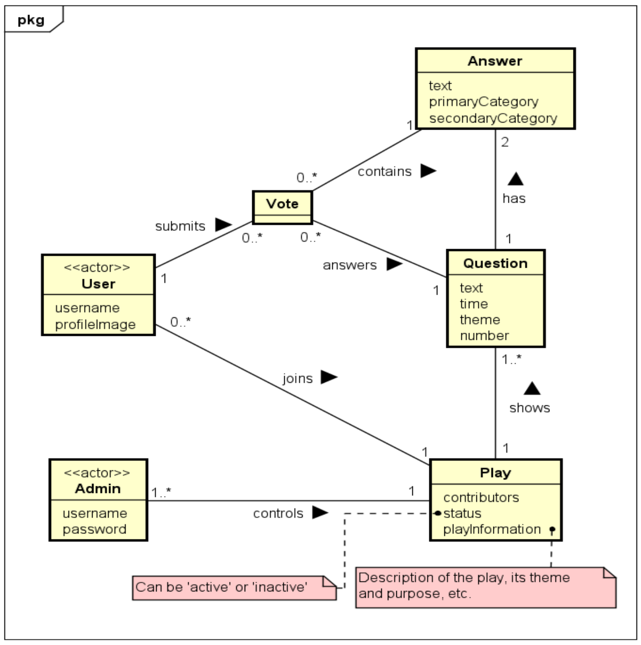


Figure 2.5. Domain Model

The question and answer objects are both represented by their text, the one being shown to the voters (the users). The question has the time, theme, and number as the other attributes, due to the need for timing in the play, an ordering of questions, and overall different themes that could be depicted throughout the play.

Overall, studying the overall problem domain resulted in the construction of the domain model depicted and described above, which stands as the basis for the further design of the project at hand and which will be expanded upon in the following chapter of the current report.

3 Design

3.1. UI Design

Bogdan Mezei

Since the TRIC web application will be utilized by members of the audience while taking part in an artistic performance, there are some important considerations that have been taken into account while designing the user interface.

Firstly, as the application will be used within a theater hall, it is essential for the user interface to use a variety of darker colors and shades so as not to be straining to the eyes of the audience in a dimly lit environment while still preserving a high contrast for easily readable text. (Figure 3.1)

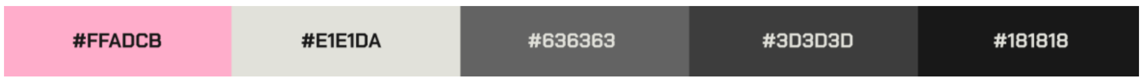


Figure 3.1.Color Palette

While designing the UI, the three principles of emotional design have been prioritized as a part of the design process (Norman, 2004). For the visceral layer, a lot of attention has been put into the color choices. The chosen palette is limited to 5 different colors which allow for a simplistic interface with plenty of contrast ranging from a slightly cool off-white to a neutral off-black. The accent color is a bold, warm pink which has been chosen in order to draw attention to specific UI elements and provide a reference to HumanLab’s brand identity.

Secondly, as a part of the behavioral layer, the functionality of the app has been analyzed and as the users will have to split their attention between using the application and watching the performance, there is a flow between voting periods and performance periods that must be taken into account and which leads to a limited time interval where the user has the possibility to view and interact with the application.

As such, the design is heavily focused on minimalist design choices which increase readability and allow the application to instantly communicate to the audience the functionality of each interface element even within the very short, limited time they are using the UI.

For this reason, all the UI elements are presented within a square, bordered boxes effectively segmenting each individual component in a logical and concise manner. This is also in relation to the cognitive design principles (Weller, 2004)

The most important elements are color coded with the bright pink color in order to draw attention and allow the user to easily figure out how to interact with the system even under the pressure of a timer. The results screen has been designed to be less distracting, allowing the user to quickly glance over what they have chosen in the previous voting period and then immediately direct their attention toward the performance.

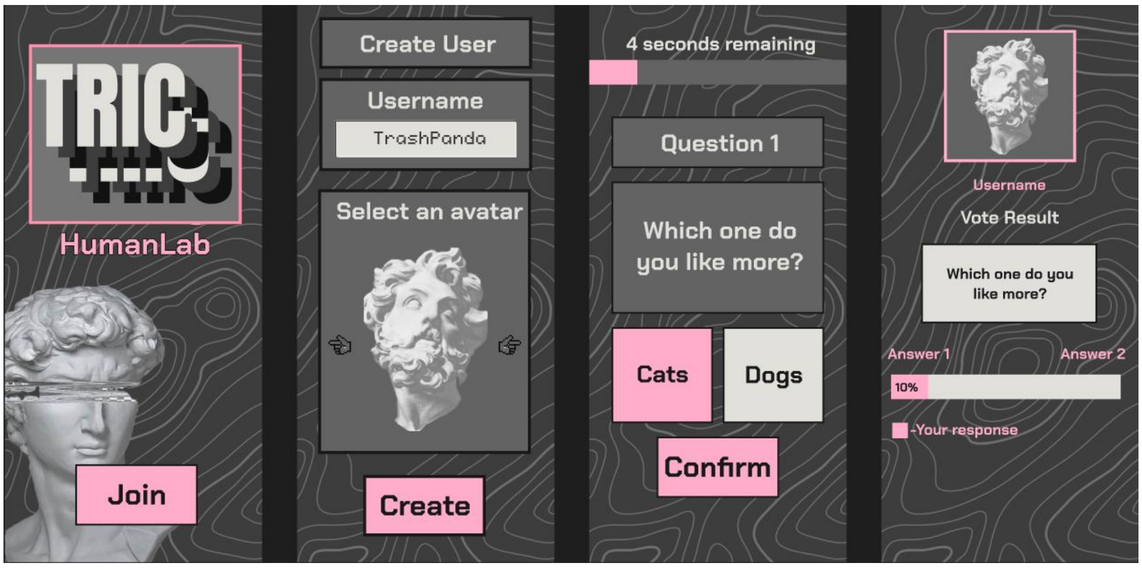
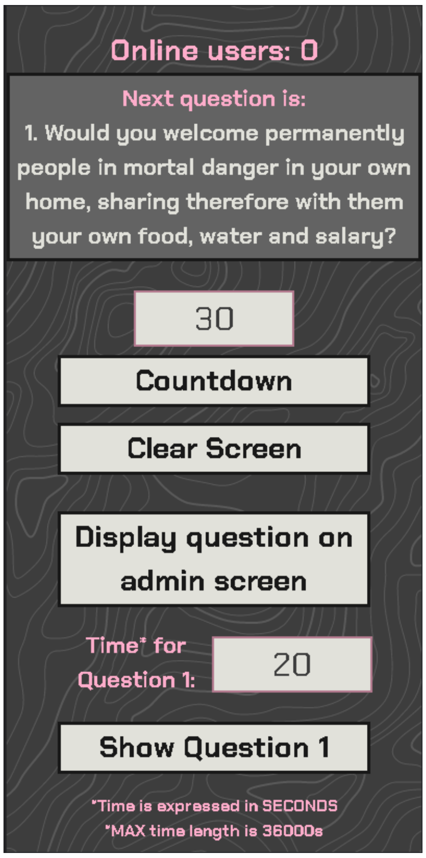


Figure 3.2. UI Design

The graphics present throughout the application depicting ancient marble statues and “glitched” text and images have been chosen together with the artistic director of the TRIC project as means of communicating a fusion between the antiquated art of theater and the novel digital graphics of today (Appendix F). In general, the graphics of the application are also part of the visceral layer, aiming to attract the attention of the user.

The user interface on the admin side of the application is designed while keeping the same principles in mind but focusing more on showing useful information and less on minimalist design choices (Figure 3.3). This is due to the fact that the administrator of the play has the need for more control on the play and will not be limited in how long they are interacting with the application. This part of the user interface is also designed with a more specialized user in mind. (Appendix F)


Online users: 0

Next question is:
1. Would you welcome permanently people in mortal danger in your own home, sharing therefore with them your own food, water and salary?

*Time is expressed in SECONDS
*MAX time length is 36000s

Figure 3.3. Admin UI Design

The reflective layer is taken into account throughout the whole application. The aim is to allow the user to form an emotional connection/a personal identity within the application and instill an emotional response. This becomes apparent from the very start when the user is asked to create a user that they can identify with by choosing their own username and profile picture.

This layer is also a big part of the voting session, as the results screen allows the user to reflect upon their choice and see how the other users have voted. In the final results page, the design aims to convey as much information as possible in an easy-to-read format, akin to an infographic and even allows for the downloading of a digital souvenir with the purpose of sharing and reflection.

3.2. Social Interaction Design

Bogdan Mezei

The application has been designed to allow audience members to vote between two political parties. This helps to create a more interactive and engaging experience for the audience and encourages them to participate in the decision-making process and have their voices heard.

By incorporating features which allow the users to see how they have voted compared to everyone else in the audience, the application helps to foster a sense of community and collaboration among audience members and helps to make the theater-going experience more immersive.

Another important part of the social interaction design of the system is how the users can download a digital souvenir after the play. This souvenir contains their username and chosen picture along with all their vote results showcasing the footprint they have left upon the show and it represents something they can identify with. This also represents a social interaction in the sense that it is meant to be shared among their peers which facilitates the relationships and awareness of augmented democracy principles within all parties.

3.3. Front-end Design

Daria-Maria Popa

The front-end architecture was structured as a single-page web application in React with functional-style TypeScript. The single-page architecture allows for improved rendering performance due to its dynamic rendering of data, a desired behavior for this system due to its user-side complexity.

React was preferred as the library with which to create the front-end part of the system due to its single-page application capabilities such as being able to display web pages more quickly and efficiently than pure JavaScript or TypeScript usage (Herbert, 2022) and have a more efficient approach of manipulating the DOM through its use of a Virtual DOM (Documentation - DOM Manipulation, 2021).

React also encourages the creation of reusable components, an useful tool when structuring complex user interfaces (UIs) and user interactions with the systems, such as the ones presented by the actors of the system in the analysis part of this report.

TypeScript was preferred over JavaScript for the readability and debugging improvements that having types brings (typescriptlang, 2022). The functional-style refers to the programming style intended to be employed when implementing the solution. Functional was chosen over Object Oriented Programming since it allows for modular code and better parallel programming (Fathima, 2021).

Since the presentation layer responsibility is part of the front-end for single-page applications, there is a system need for state and lifecycle management in order to save, update, and retrieve the relevant information used throughout all the application components.

For this project, the state management was done using Redux, a state container (Redux, 2022) that makes use of a store in order to record the system variables. The Redux state diagram design for this project can be seen in Figure 3.4, following the Redux pattern (Redux, 2022), with each state being immutable and only one store for the application broken up into individual slices each handling a certain functionality of the system, following the Redux slice pattern (Taranto, 2022).

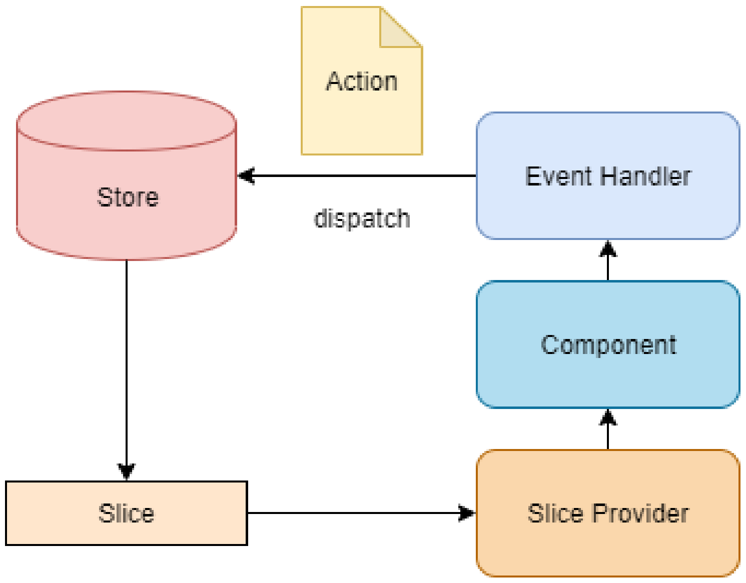


Figure 3.4. Redux State Diagram

React hooks design pattern was used for managing the lifecycle behavior of the application components (Hooks Pattern, 2022), providing direct access to the state, managed by the Redux store.

The overarching architectural structure of the frontend system can be better described through the use of a package diagram. As represented in Figure 3.5 there are seven main packages: API, app, components, style, models, reducers, and services.

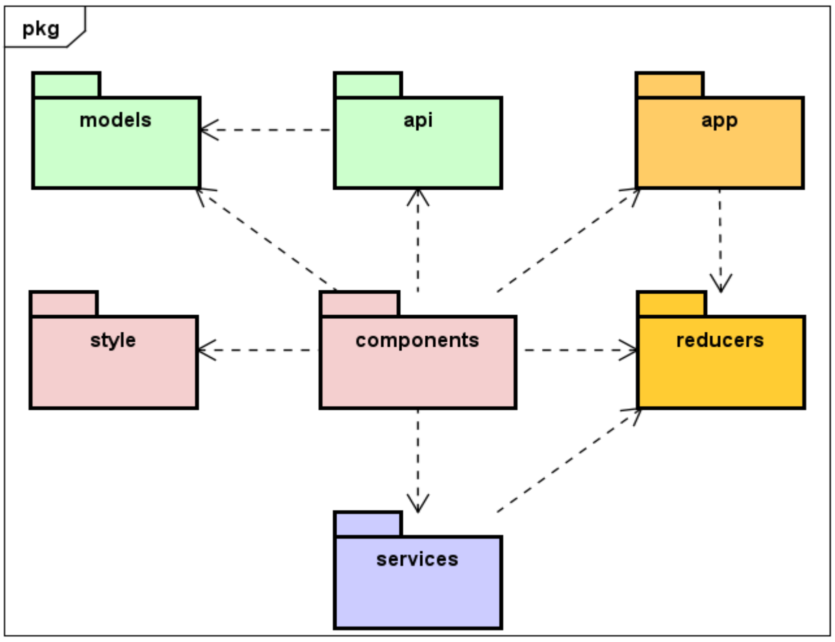


Figure 3.5. Frontend Package Diagram

The component package handles the UI logic, split into multiple smaller reusable elements called components. These handle the presentation layer of the system, complete with the style package handling the visual side of the components.

The app package handles the custom hooks used to access the store and Redux store, while the reducers package holds the Redux slices, conforming to the Redux best practices of structuring the system (Redux, 2022). The hooks and the actions presented by the slices are related to the components in order to allow the system to dynamically change the displayed information using Redux's Event Handler (Figure 3.4).

Api package holds the connection layer information to the server side (back-end) of the system. It makes use of certain model classes, structured in the model package, in order to send and receive information to and from the server. Certain events on the UI should trigger changes to be made in the back-end in order to persist or retrieve data, therefore the connection between the components and the api and model classes.

Lastly, the services package contains the authorization and authentication configurations employed on the system, resulted from security-related non-funtional requirements analyzed in the previous chapter. They help ensure the safety of both main system actors and the system's adherence to the GDPR regulations, these services are used by the components, and relevant information is saved in the Redux store.

The components that are part of the 'component' package, as described above, have been structured in three groups based on their target audience: user components, admin components, and general use (both by the user and by the admin) components, as visualized by the dependency diagram in Figure 3.6.

For a more complete overview of the front-end architecture, the dependency diagram can be generated across all the packages listed above (Appendix D - Frontend Dependency Diagram). This diagram brings an added level of specificity into the design of the packages, how they are connected, and how they work together for creating the user-side of the system.

The Redux state diagram together with the package diagram and its description and the component dependency diagrams help outline the overall front-end design, whose implementation will be detailed in the following chapter.

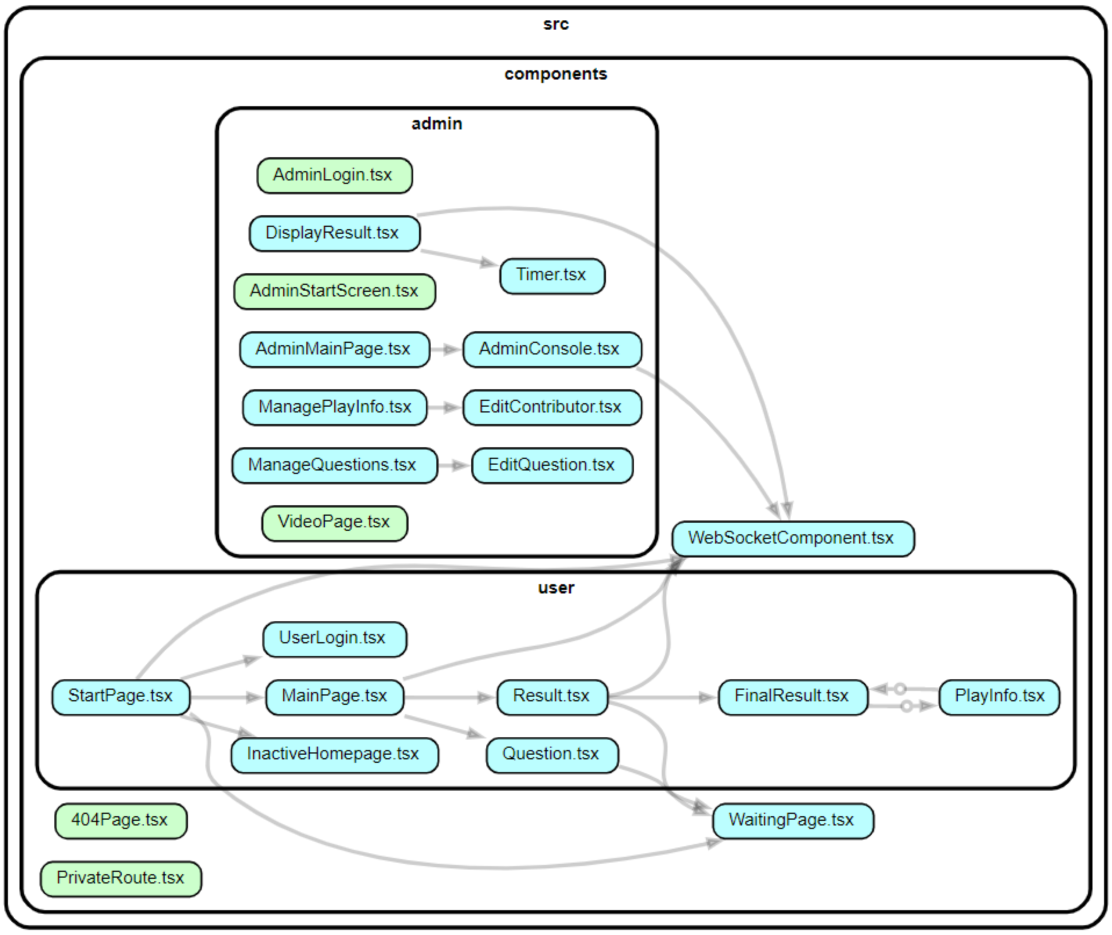


Figure 3.6. Component Dependency Diagram

3.4. Networking Design

Natali Munk-Jakobsen

The general structure of the system is based on **Client - Server architecture** and the communication between the client and server is established with HTTP connection and WebSocket.

3.4.1. HTTP Connection

The main communication between the client and server is handled by HTTP requests and responses using REST API in the server part and Axios in the client part (Figure 3.7).

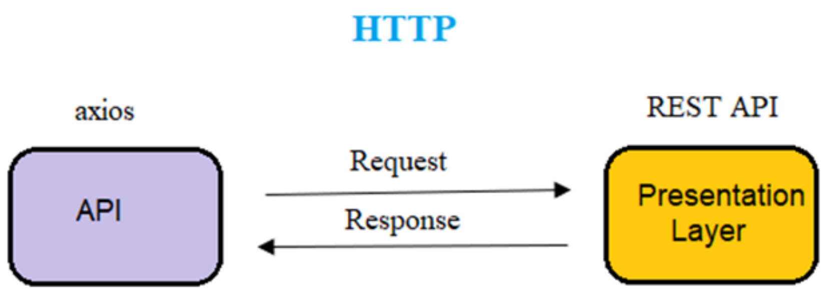


Figure 3.7. HTTP Connection

Representational State Transfer (REST) is an architectural style defining a set of rules for creating web services. REST API conforms to the REST design principles when using HTTP requests from the client to create, read, update and delete data. REST API was chosen because of its simplicity, flexibility, independence, scalability, and ability to handle all data types.

Axios is an HTTP client Javascript library based on XMLHttpRequests service used to send asynchronous HTTP requests to REST endpoints. In this project, Axios was preferred over Fetch API, another promise-based HTTP client, for the following reasons:

1) Axios uses XMLHttpRequest which is widely supported by most browsers including older browsers.

2) Axios automatically signifies the data when sending requests to the server and transforms the data returned from the server.

3)Axios has better error handling and can throw 400 and 500-range errors.

3.4.2. Websocket Connection

According to the project requirements, the admin needs to send real-time updates, thus a protocol for real-time messaging was needed. The WebSocket protocol was used for handling real-time messages in the system (Figure 3.8). Websockets allow full-duplex bidirectional communication between the server and the client after the initial HTTP handshake, therefore both the server and the client can send messages at any time without any delay. However, for this project, the WebSocket connection was used only for sending messages from the server to the client in the project.

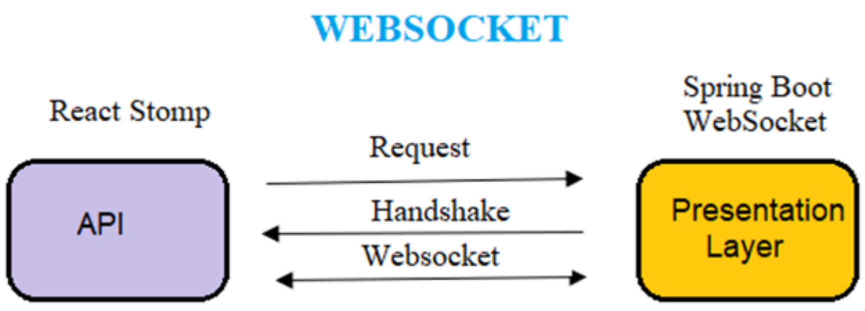


Figure 3.8. WebSocket Connection

STOMP messaging was used with Spring Boot WebSocket to enable the server to send real-time messages asynchronously, without requiring the client to send a request each time. STOMP is a simple text-orientated messaging protocol operating on top of the lower-level WebSocket.

Spring Boot Framework has been preferred because it provides an easy configuration for web socket connection allowing Web socket message broker to create STOMP endpoints for the WebSocket handshake and to send and receive messages on. React-stomp, a JavaScript library to create React WebSocket components with STOMP, was used in the front-end part for receiving WebSocket messages from the server.

3.5. Back-end Server Design

Natali Munk-Jakobsen

The architecture of the back-end server is **“Three-layer Spring application architecture”** and consists of Presentation, Application, and Data Access Layers (Figure 3.9). The most important feature of the layered architecture is separation of concerns among components. Components with similar functionalities are organized into layers and each layer performs a specific role in the system.

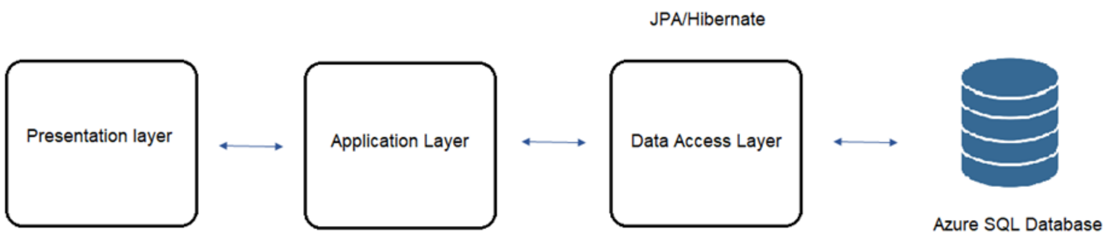


Figure 3.9. The architecture of the back-end server

The presentation layer consists of three controller classes providing APIs for creating, retrieving, adding, updating, and deleting data. The UserController manages requests from the primary users of the application and does not require authorization.

The AdminController responds to requests from the admin console for managing the application. The methods in this class are accessible with role-based validation and require an access token. The AuthController handles requests for the login process using the JWT (auth0.com, 2021) Authentication flow.

Spring Boot is an open-source Java-based framework used to build a RESTful Web service in the Presentation Layer. Spring Boot was chosen because it provides an easy approach to creating Service and Controller classes using auto-configuration, annotations, and easy dependency management, therefore reducing development time and increasing productivity (spring.io, 2022).

The Application Layer (Business/Service Layer) mediates communication between Presentation and Data Access layers and contains business logic. The Application Layer consists of service interfaces and their implementations.

The service interfaces have been added between the presentation layer and business logic to be compatible with SOLID principles. According to The Dependency Inversion Principle of SOLID, high-level modules should not depend on low-level modules, they both should depend on abstractions. Interfaces enable the change of higher-level and lower-level components without affecting any other layers and provide maintainable and extendable code (Martin, Robert C., 2003).

Data Access Layer consists of repository components that encapsulate the logic needed to access data sources. In order to separate persistence operations from high-level business service classes, the Repository pattern was used in the data access layer.

The Repository Pattern consists of an interface definition that declares the persistence methods and implementation class that provides data store-specific implementations of each interface method. The pattern abstracts the data store, centralizes data access functionality, and improves the reusability of the persistence code. Furthermore, it provides better maintainability by decoupling the infrastructure used to access databases from the business logic (Evans, E., 2004).

In this project, the repository pattern was used together with two technologies: Hibernate and Spring Data JPA. **The Java Persistence API (JPA)** provides a specification for mapping Java domain model objects to the relational database tables using annotations. **Hibernate** is an object-relational mapping (ORM) solution for Java environments that provides a reference implementation of JPA. Hibernate was chosen for the implementation of JPA because it has fast performance due to its caching mechanism, generates database-independent queries, and is highly scalable (hibernate.org, 2022).

Spring Data JPA is one of the Spring Data modules and is used for integrating Spring applications with JPA using different APIs to perform CRUD operations. It provides a set of interfaces for creating data access repositories for JPA-based data access layers. This abstraction makes it easier to work with Hibernate or other JPA providers and reduces the amount of boilerplate code required to execute simple queries (spring.io, 2022).

Microsoft Azure SQL Database is the relational database management system used for the persistence of the system. It is a fully managed platform as a service (PaaS) database engine built for the cloud (azure.microsoft.com, 2022). Microsoft Azure SQL

Database was chosen for database management since it automates updates, provisioning, and backups and takes care of scalability and availability. Furthermore, it is easy to integrate with Azure App Service, a fully managed PaaS for building web applications, that is used for hosting back-end servers.

The use of **Azure App Service** was preferred primarily because of its easy deployment on a scalable, secure and reliable cloud infrastructure. It is also useful for providing built-in HTTPS support and CORS support, besides offering DevOps integration by streamlining CI/CD with Git.

To give another overview of the system and its design, the organization of the system is shown in the package diagram in Figure 3.10. Controller, Service, and Repository packages contain interfaces or classes for the layers mentioned in the general architecture of the back-end server, respectively Presentation, Application, and Data Access Layers.

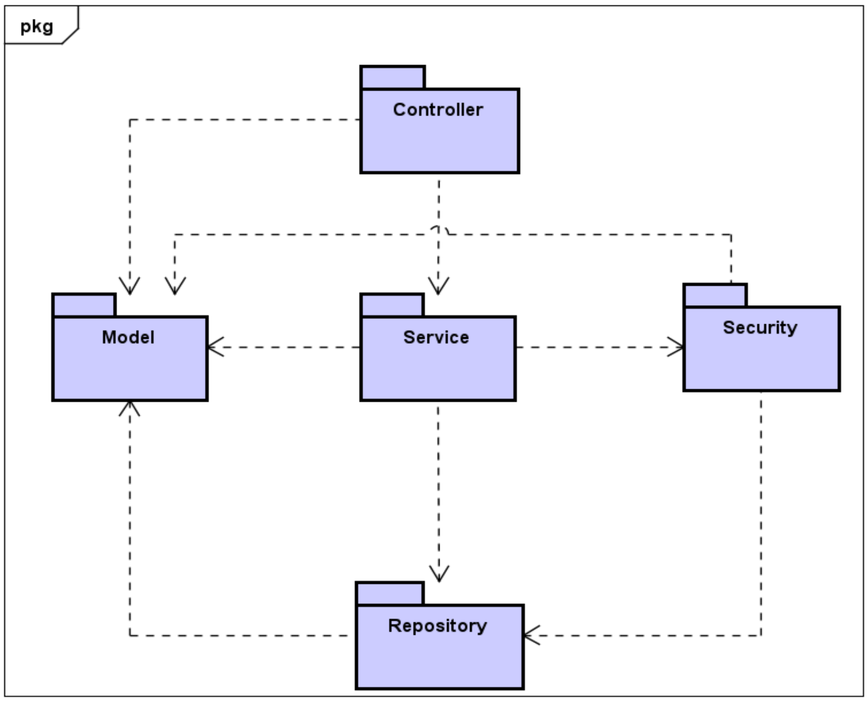


Figure 3.10. Backend Server Package DIagram

The Model Package consists of Domain Entities and Data Transfer Objects and has a connection with Controller, Service, and Repository packages since all layers make use of relevant classes from this package for different purposes.

Security Package contains classes and configuration needed for authentication and authorization. It is connected to Service, Model, and Repository packages in order to manage login requests from the client and persistence for the admin user data. The service implementation class related to user authentication is not directly connected to the repository package. The security package provides an additional helper tier between service and repository layers for managing the logic for authorization-authentication processes.

A more detailed overview of the system architecture was demonstrated with a UML class diagram showing all classes and interfaces and relations between them (Appendix D - Backend Class Diagram).

3.6. Security Design

Bogdan Mezei

Since the admin is responsible for managing all parts of the play and ensuring a good flow within the system, there is a need to authenticate them beforehand in order to ensure unauthorized users cannot interfere with the system.

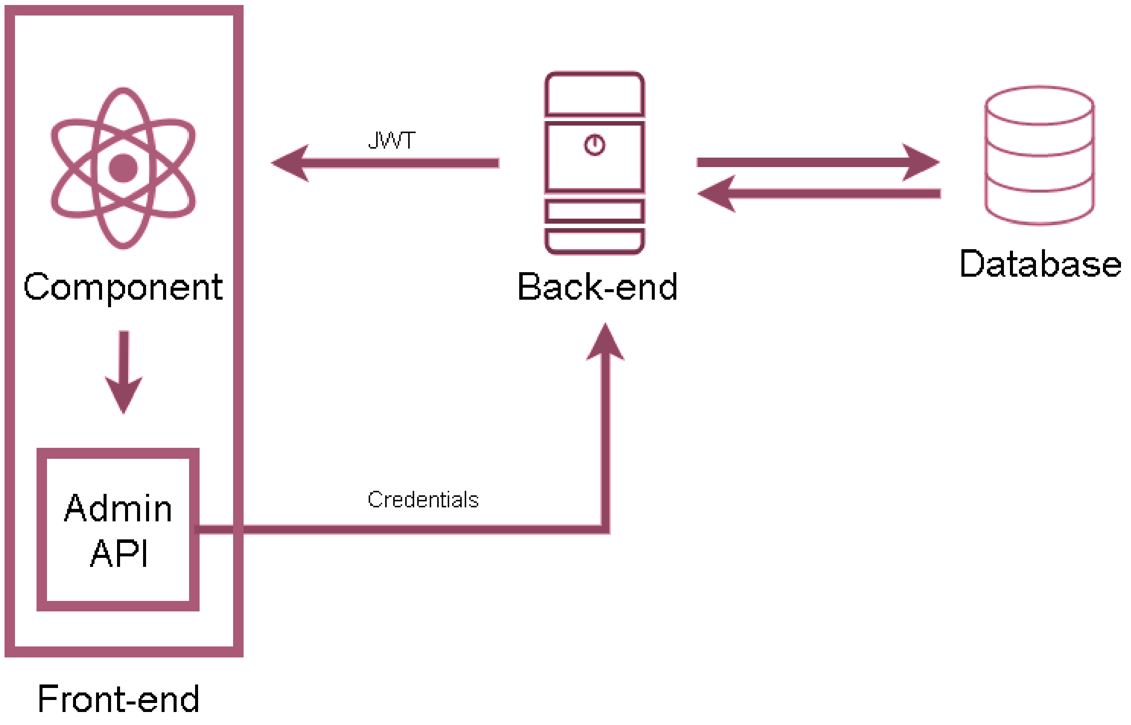


Figure 3.11. Receiving JWT

The authentication for the administrator is done using JSON Web Tokens (JWTs) (auth0.com, 2021) and takes the following steps which will be described in more detail in the implementation part of the report:

1. The login react component sends the admin credentials to the back-end
2. The back-end checks the database for the provided credentials
3. The back-end creates the JWT if the credentials are correct
4. The back-end returns the JWT
5. The JWT is used on the client side within the Admin API to make authorized requests to the backend (such as login, manage questions, etc.)

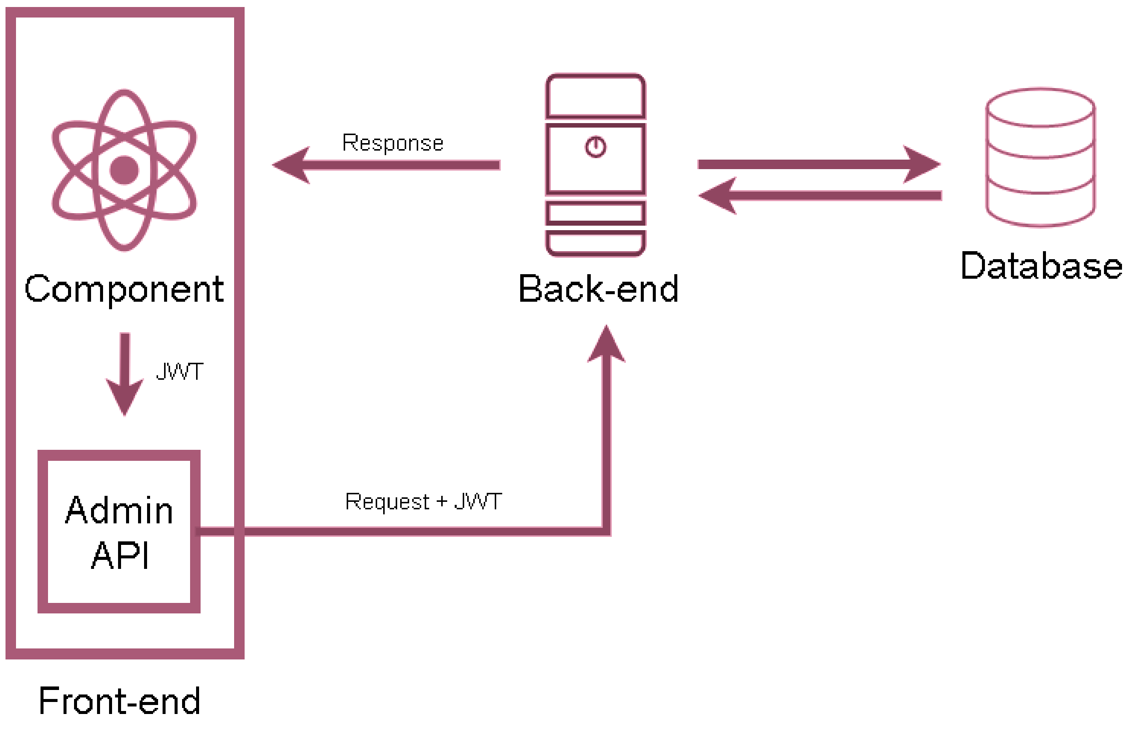


Figure 3.12. Sending Authenticate Request

3.7. Machine Learning Design

Daria-Maria Popa

In order for a user's answers to be determined based on previous voting choices inside of the play, the system design needs to encapsulate a way to generate these predictions. For this project, this prediction system was designed through the use of an unsupervised machine learning model as part of the back-end system.

The machine learning algorithms have been designed through the use of Weka, a Java-based machine learning software (Frank et al., 2016), in order to facilitate easy integration with the Java back-end of this system. Weka was also chosen for its well-documented built-in machine learning tools and direct access to Python-based tools, the more used programming language for machine learning problems (Frank et al., 2016).

The 'unsupervised' model design refers to models that decipher themselves the provided data to reveal hidden patterns and insights (Unsupervised Machine learning -

Javatpoint, 2022). The goal of such models aligns with the intended behavior of the systems in terms of vote prediction, that being 'finding underlying patterns and structures in the given data' (Unsupervised Machine learning - Javatpoint, 2022).

Another consideration when choosing the type of model as unsupervised was the nature of the data. For this system, the data represents real-world voting data, with no known 'labels' or expected outcomes, since the prediction is supposed to happen before the user has the chance to vote. Therefore, the system must be able to predict what the user might have chosen with no added supervision, simply based on the given data, consisting in this case of previous answers, as specified by the non-functional requirements in the previous chapter.

Lastly, the number of expected users each play was a deciding factor for the type and possible complexity of the machine learning models to be used in the system. Since the expected number of players is that of a normal HumanLab play audience (approximated together with the company at under 50), this put a significant limitation on the number of models that could find data patterns using such limited data points.

The chosen model based on all the previously stated considerations and delimitations and the result of the machine learning design was the agglomerative hierarchical clustering model (Hierarchical Clustering in Machine Learning - Javatpoint, 2022), whose implementation will be detailed in the following chapter.

3.8. Database Design

Natali Munk-Jakobsen

The database modeling is done in three steps. Firstly, a conceptual model was made to present an overall picture of the system by using the information gathered from business requirements. Entities and relationships were defined considering the stakeholder's needs and modeled in the Conceptual ER diagram (Figure 3.13). The database model consists of three independent schemas for handling the admin login process, participating in the voting, and managing play info, respectively: admin_login, play, and voting.

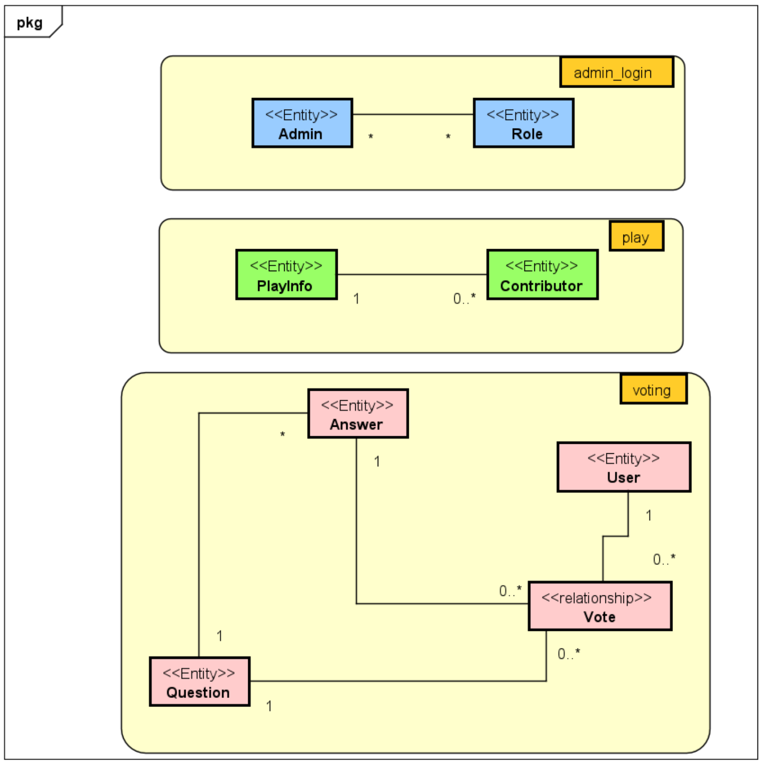


Figure 3.13. Conceptual ER Diagram

Secondly, a logical model was designed to enrich the conceptual model by defining the columns in each entity (Figure 3.14).

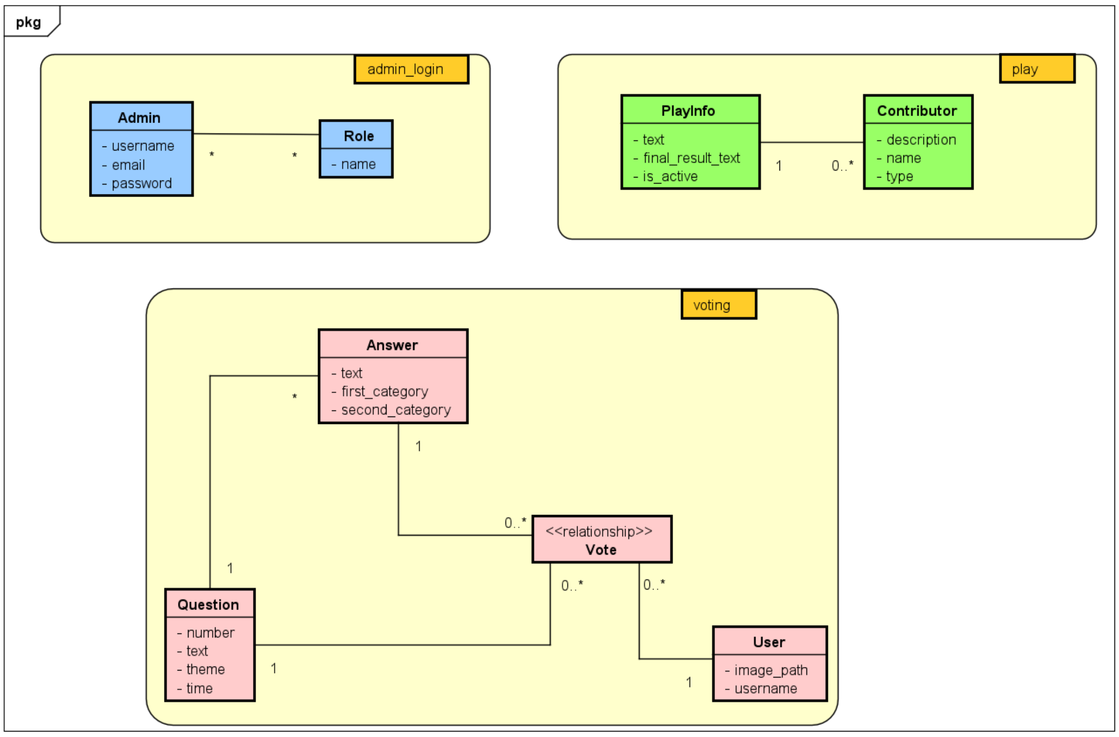


Figure 3.14. Logical ER Diagram

Lastly, the Physical Model was developed as a representation of the actual design of the database (Figure 3.15). Since the Physical ER diagram is designed to be instantiated as a database, it shows how data should be structured and contains an accurate use of data type for entity columns, including primary keys, foreign keys, and constraints.

The main focus of the system (Requirement 1) is casting user votes. User, Question, Answer, and Vote tables are held in the voting schema and are responsible for casting user votes. A user can vote by choosing one answer for each question. In order to manage participation in the play and voting process, user information was needed. The user entity holds personalized user info. Question and Answer entities contain other required data for voting. The relationship between the User, Question, and Answer entities is a ternary relationship.

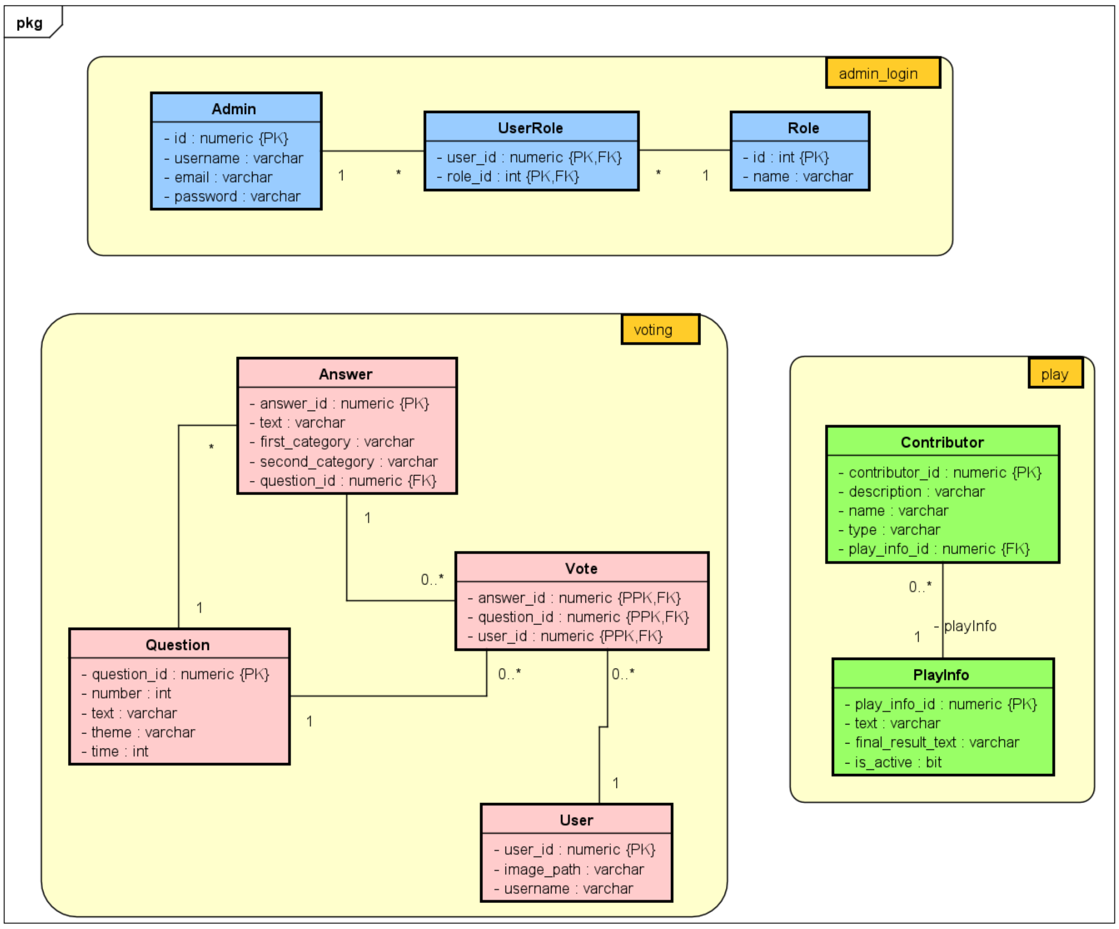


Figure 3.15. Physical ER Diagram

Even though the relation between the question and answer was delimited by stakeholders as " A question has only two answers" and this delimitation affected the design of the system in other parts, there is no restriction in the database design for this delimitation. The relationship between the question and answer was modeled as a one-to-many relationship using a foreign key in the answer entity.

Admin and Role tables are in the admin_role schema and hold the data for authorization/authentication processes. The relationship between Admin and Role is a many-to-many relationship. A junction table, UserRole, was used in the database to bridge the tables together by referencing the primary keys of each table.

Play schema holds PlayInfo and Contributor tables. In addition to general information and the status of the play, a list of contributors is a part of Play Info. The relationship between PlayInfo and Contributor entities was represented with a one-to-many relationship and handled by adding a foreign key in the Contributor table.

3.9. CI/CD and DevOps

Bogdan Mezei

DevOps has been used as an approach to development in order to ensure the code is easily testable and maintainable. For this reason, a workflow containing continuous integration and continuous deployment pipelines has been developed.

3.9.1 Continuous Deployment (CD)

Within the workflow there are two separate deployment jobs. The first one is run every time new code is pushed to a pull request. The job automatically deploys the new version to a testing environment. This step is an important prerequisite for the continuous integration pipeline.

The second job is run only when a pull request is merged into the master branch, and after the continuous integration job is fully successful. This job deploys the newly merged version into the production environment.

The testing environment is experimental and meant as a sandbox for testing purposes as opposed to the production environment which is deployed only after the code has been fully tested.

3.9.2 Continuous Integration (CI)

The continuous integration job within the workflow automatically runs end-to-end tests on the testing environment after the CD pipeline has finished deploying there. Each test suite is run sequentially, and a report is generated at the end. In the case that any tests fail, the new code version is not allowed to be merged into the master branch.

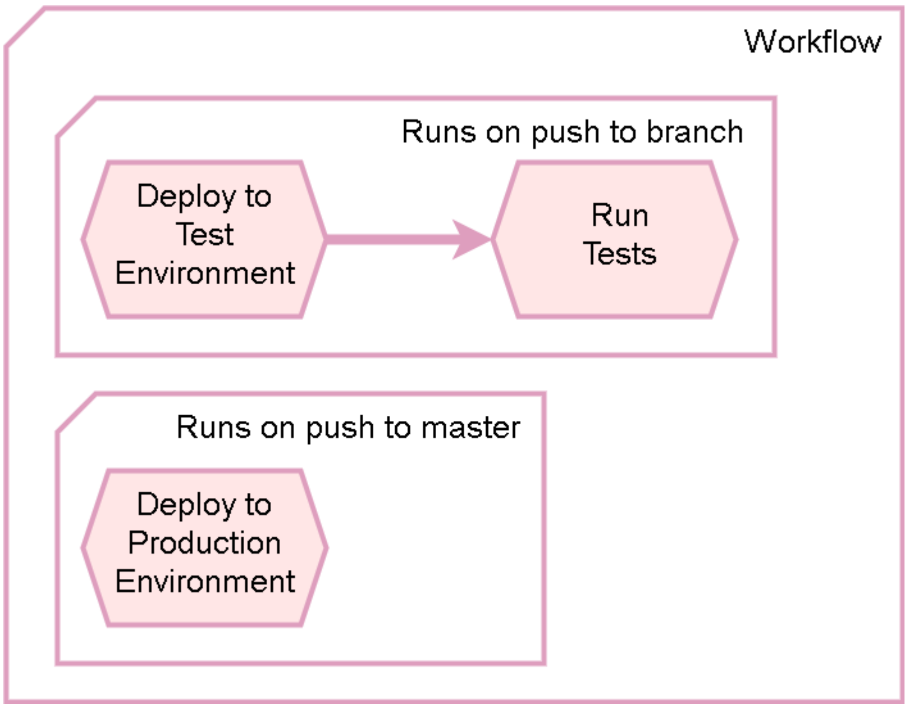


Figure 3.16. CI/CD workflow

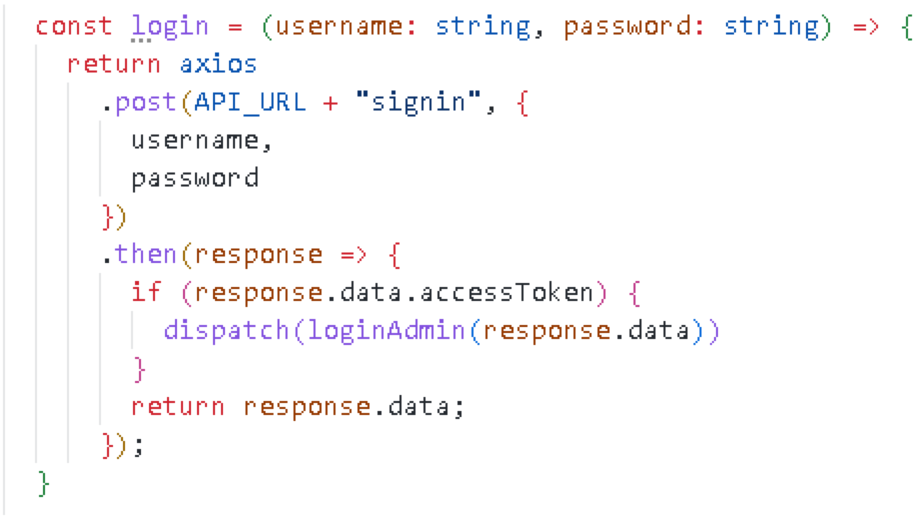
4 Implementation

4.1 Security

Bogdan Mezei

The following steps are the in-depth version of the authentication explained in the design part of the report:

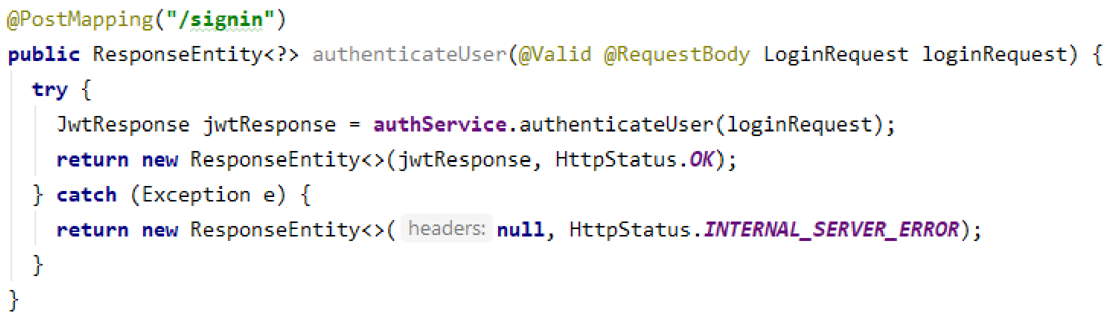
1. The admin can input their credentials in the AdminLogin component which calls the login function within AdminApi. The credentials are sent to the backend using Axios.



```
const login = (username: string, password: string) => {
  return axios
    .post(API_URL + "signin", {
      username,
      password
    })
    .then(response => {
      if (response.data.accessToken) {
        dispatch(loginAdmin(response.data))
      }
      return response.data;
    });
}
```

Figure 4.1. Login Function

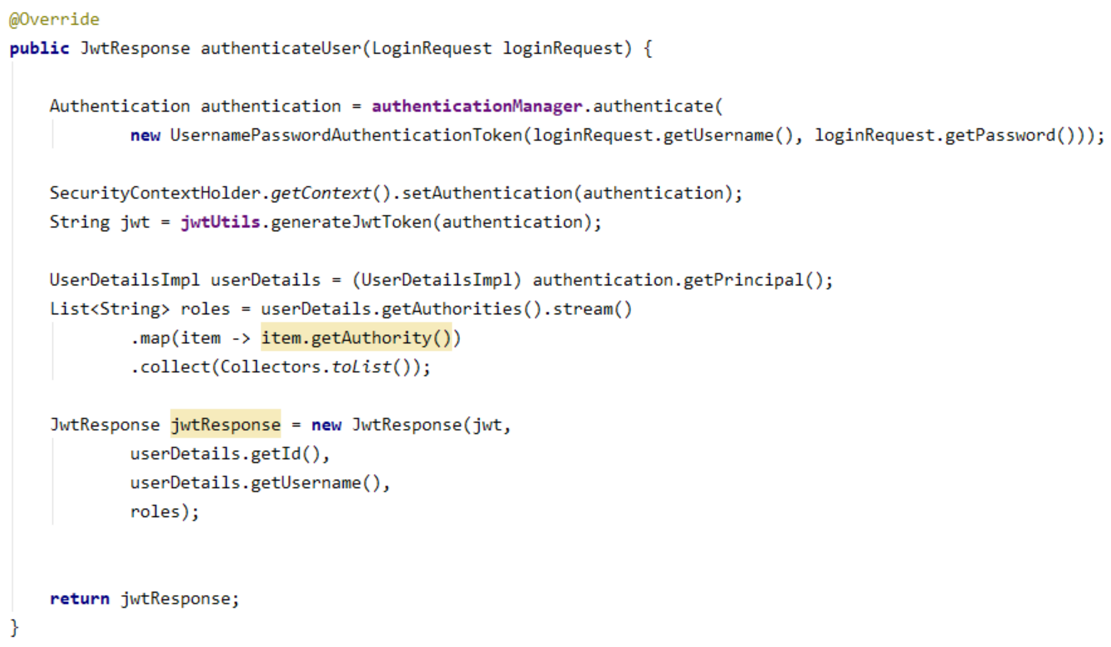
2. The back-end receives the credentials through the "/signin" POST mapping and authenticates the administrator. The password is hashed using BCrypt and checked against the database along with the username.



```
@PostMapping("/signin")
public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {
  try {
    JwtResponse jwtResponse = authService.authenticateUser(loginRequest);
    return new ResponseEntity<>(jwtResponse, HttpStatus.OK);
  } catch (Exception e) {
    return new ResponseEntity<>( headers: null, HttpStatus.INTERNAL_SERVER_ERROR);
  }
}
```

Figure 4.2. AuthenticateUser method

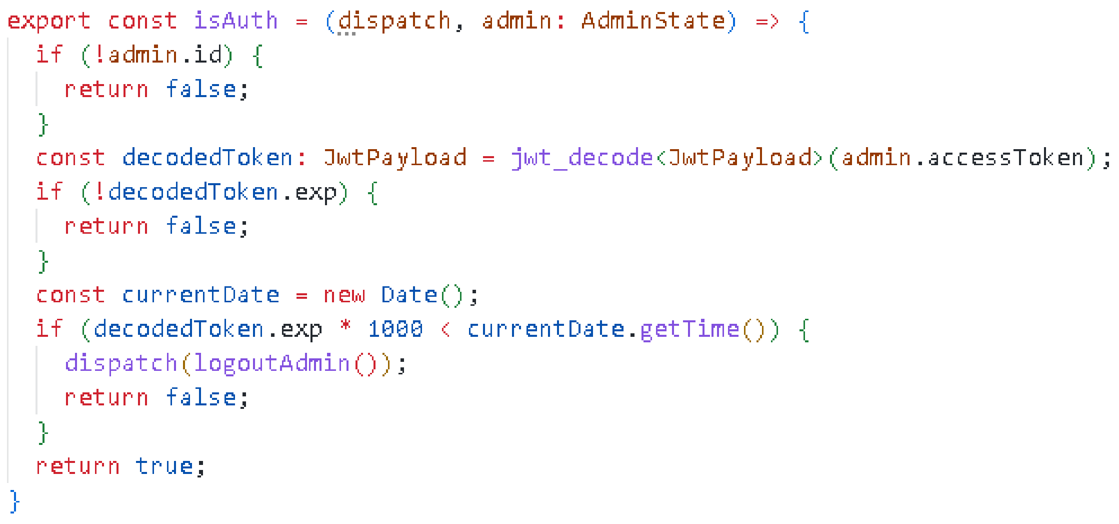
3. If the password is correct a JSON Web Token is created and sent back to the front-end.



```java
@Override
public JwtResponse authenticateUser(LoginRequest loginRequest) {

    Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword()));

    SecurityContextHolder.getContext().setAuthentication(authentication);
    String jwt = jwtUtils.generateJwtToken(authentication);

    UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
    List<String> roles = userDetails.getAuthorities().stream()
            .map(item -> item.getAuthority())
            .collect(Collectors.toList());

    JwtResponse jwtResponse = new JwtResponse(jwt,
            userDetails.getId(),
            userDetails.getUsername(),
            roles);


    return jwtResponse;
}
```

Figure 4.3. Creating JSON Web Token

4. The created JWT is sent as a response for the same request made with the POST mapping "/signin".



```typescript
export const isAuth = (dispatch, admin: AdminState) => {
  if (!admin.id) {
    return false;
  }
  const decodedToken: JwtPayload = jwt_decode<JwtPayload>(admin.accessToken);
  if (!decodedToken.exp) {
    return false;
  }
  const currentDate = new Date();
  if (decodedToken.exp * 1000 < currentDate.getTime()) {
    dispatch(logoutAdmin());
    return false;
  }
  return true;
}
```

Figure 4.4. Admin Authentication

5. On the client-side, the JWT is saved within AdminState in LocalStorage through the Redux Store. This is done in order to keep the session active for as long as the token is valid. If the token is expired when checked, the admin will be logged out and LocalStorage is cleaned. This token is subsequently used in every request made by the AdminApi in order to authorize the user.

The system is designed to allow a single admin user to manage the play at one time. While theoretically allowed to have multiple admin accounts and multiple users logged in at the same time as admin, in practice only one single account is present in the data. This does not constitute an issue as the system is intended by the stakeholders to be used with a single admin account at a time. The system is also closed for registration for this same reason, to keep out unauthorized users from managing the play. Every admin account is added manually to the database by inserting a row with the username and hashed password.

4.2 Front-end

Daria-Maria Popa

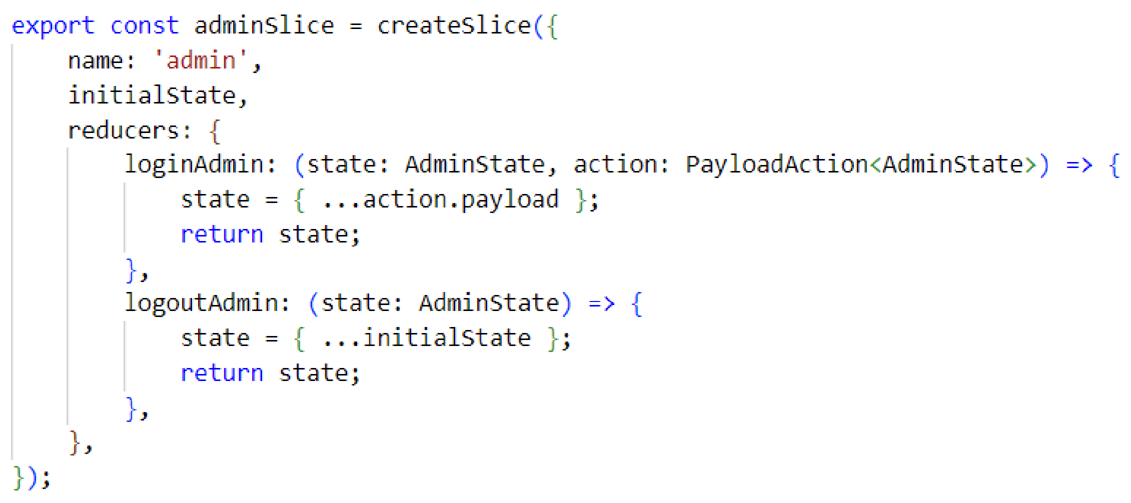
The front-end implementation uses, as mentioned in the Design chapter, Redux pattern in order to manage the different states of the system resulting from the different user interactions and data flows. The system global state refers to the condition of the voting application based on the inputs that have been saved in it (Redux, 2022).

By default, the Redux state uses the web browser session storage. However, the implementation of this system made use of a persisted Redux store, in order to save the application state in the local storage (redux-persist, 2022), ensuring that events such as refreshing or exiting the application would not have an effect on the persisted information and that the user or admin state could be retrieved in its latest version upon reopening the application.

The implementation of the overall Redux state management in the system can be split into three parts: the slices, the store, and the custom hooks.

Firstly, for the Redux slices, there is a total of seven and they are based on the different features and data types present in the system. These slices contain their own reducers, functions that when called upon with a certain action and the current state as parameters, alter the current state and return its updated version.

An example of one of these slices can be seen in Figure 4.5 which portrays the ‘adminSlice’. It’s purpose, as the naming convention suggests (Redux, 2022), is managing the Admin data throughout the global context of the application.



```
export const adminSlice = createSlice({
    name: 'admin',
    initialState,
    reducers: {
        loginAdmin: (state: AdminState, action: PayloadAction<AdminState>) => {
            state = { ...action.payload };
            return state;
        },
        logoutAdmin: (state: AdminState) => {
            state = { ...initialState };
            return state;
        },
    },
});
```

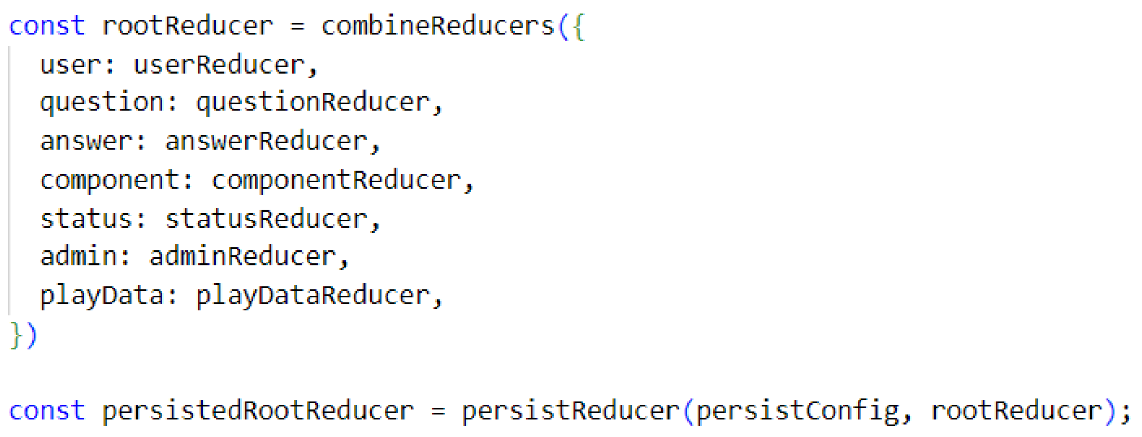
Figure 4.5. Admin Slice

Every slice is created using the ‘createSlice’ function, and takes a name, used to refer the reducer by and differentiate it from the other reducers in the store, an initial state, and the reducers of the slice.

The ‘initialState’ is used to set the state of the slice when the reducers are called with undefined state values or the local storage is empty (createSlice | Redux Toolkit, 2022). Because the front-end application uses TypeScript, the method header of the reducers can define the expected types of the state and action, which ensures the relevant type of data is being sent to each slice and, further on, persisted into the local storage of the user’s web browser.

Conforming to the Redux regulations of correctly using the reducers (Redux, 2022), each reducer modified the state and has no other side-effect, then returns the updated state, and application lifecycle updated accordingly.

Secondly, the Redux store is an object with several methods that help configure it. The store holds all the states of the application and combines them into one ‘rootReducer’ (Figure 4.6). This reducer is the one persisted into the local storage, as mentioned in the design of the front-end application.

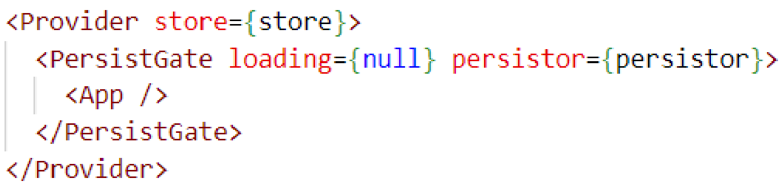


```
const rootReducer = combineReducers({
  user: userReducer,
  question: questionReducer,
  answer: answerReducer,
  component: componentReducer,
  status: statusReducer,
  admin: adminReducer,
  playData: playDataReducer,
})

const persistedRootReducer = persistReducer(persistConfig, rootReducer);
```

Figure 4.6. rootReducer

In order to make the root reducer persistable, the ‘persistReducer’ Redux function is used (Figure 4.6), with a specific configuration that holds the information about the key (the name of the reducer when written into storage) and type of storage to be used when persisting the state information. The default storage type for persisted reducers in web applications and the one used for this application is the local storage (redux-persist, 2022).

As the final step of setting the store, specifically a persisted store, to be accessible throughout the application, the Provider and PersistedGate components are wrapped around the main App component of the single page web application (Figure 4.7), as required setup by Redux (redux-persist, 2022).

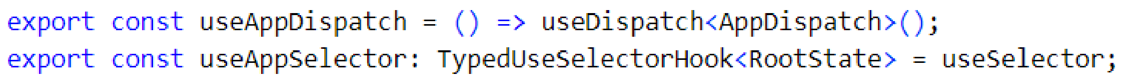


```
<Provider store={store}>
  <PersistGate loading={null} persistor={persistor}>
    <App />
  </PersistGate>
</Provider>
```

Figure 4.7. Provider and PersistGate

Lastly, in order for the state accessed from the store, certain Redux custom hooks were used throughout the application (Hooks at a Glance – React, 2022). These have been added as part of the 'app' package together with the store, and represent custom implementations of the 'useSelector' and 'useDispatch' functions (Figure 4.8).

The custom implementation makes use of the current application dispatch and root state exported from the store, and simplifies the further code usage of the hooks by not having to specify the type of dispatch and root state every time one of the hooks is called in a component.



```
export const useAppDispatch = () => useDispatch<AppDispatch>();
export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;
```

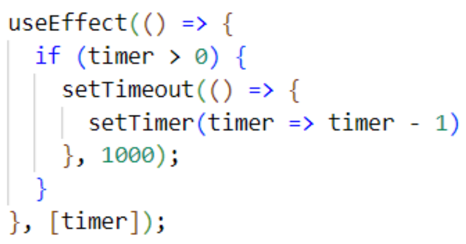
Figure 4.8. useAppDispatch and useAppSelector

The dispatch function has been used to dispatch actions to the store in order to modify the state, while the selector has been used to retrieve certain information from the store and subscribe to the store for any changes.

Following the Design chapter of this report, the React components have been split into the three groups: admin, user, and general components. The components use the React hook pattern in order to manage the application lifecycle. Besides the Redux custom hooks described above, the application makes use of base React hooks such as 'useState' and 'useEffect'.

The 'useState' hook has been used throughout the components for for local state management - states that were only relevant for that particular component, and where using Redux global state management would have been redundant. Examples of these states are local timers, counters, and boolean values used to control the look of the UI, since the 'useState' hook rerenders the view on state changes.

‘useEffect’ was used to implement side-effects after lifecycle events such as component ‘mount’ and ‘unmount’ or state changes for certain useState or useAppSelector variables (Figure 4.9). The image bellow depicts one of these examples, where whenever the timer value changes, the useEffect hook triggers and, on a certain condition, a side-effect happens, in this case the timer being decreased by 1 after waiting for a second.

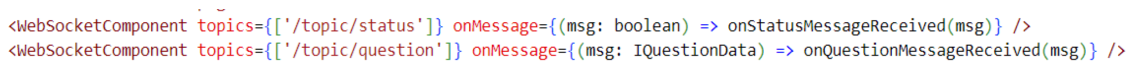


```
useEffect(() => {
  if (timer > 0) {
    setTimeout(() => {
      setTimer(timer => timer - 1)
    }, 1000);
  }
}, [timer]);
```

Figure 4.9. UseEffect Hook

For passing information directly from a parent component to a child component, when that information was not relevant for the rest of the application or important to be persisted as part of the root state, the use of Props was employed. The props allowed for context-based information to be persisted, such as specific waiting messages and animations for the user, based on where the waiting component was being shown from.

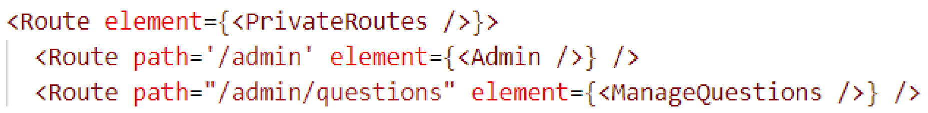
For the front-end implementation of the WebSocket connection between the client and server side of the system, a SockJSClient component was used inside of a custom WebSocket component. The custom WebSocket component made use of the props structure explained above in order to receive the topic name the socket was meant to listen to, and the ‘onMessage’ function reference called when a new message on the topic was heard from the server. This WebSocket component was used as part of the user components and some admin components such as the DisplayResult, in order to listen for certain topic messages, such as a new question being sent out by the admin (Figure 4.10).



```
<WebSocketComponent topics={['/topic/status']} onMessage={(msg: boolean) => onStatusMessageReceived(msg)} />
<WebSocketComponent topics={['/topic/question']} onMessage={(msg: IQuestionData) => onQuestionMessageReceived(msg)} />
```

Figure 4.10. WebSocket Component

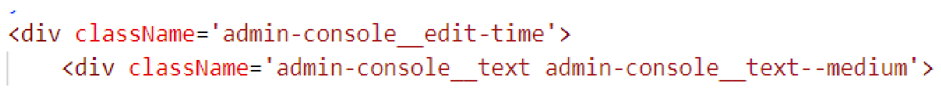
React Router has been used in order to separate the user from the admin routes, and to secure the admin routes using the 'isAuth' function described in the Security implementation. The 'PrivateRoute' component was added as a new Route element to encapsulate all secure application paths, check for a valid authentication token, and re-route to the intended path or the login component based on the result (Figure 4.11).



```
<Route element={<PrivateRoutes />}>
  <Route path='/admin' element={<Admin />} />
  <Route path="/admin/questions" element={<ManageQuestions />} />
```

Figure 4.11. React Router

For structuring the class attributes of the HTML elements present throughout the components and the overall CSS styling of the application, the BEM methodology was used (Strukchinsky, 2022), splitting the class names into block, element, and modifier classes (Figure 4.12).



```
<div className='admin-console__edit-time'>
    <div className='admin-console__text admin-console__text--medium'>
```

Figure 4.12. CSS styling and BEM methodology

The icons, constants, and common functions used throughout the application, such as the Http-common file which exports a function that creates the basis of an axios request to be sent to the back-end, have been organized inside of the util package.

The constants in the util package refer to the URLs used when creating requests, and all the text shown in the application. The text being saved as constants helps the code structure by simplifying the component look, allowing for reusable constants, and setting a good foundation for future prospects of implementing localization in the application.

The model TypeScript files from the model package of the frontend application have been implemented as TypeScript interfaces that contain the same data structure as the one used in the back-end of the system. The interface implementation compared to classes comes from the usage of functional style programming throughout the front-end application.

These models have been used in the components, reducers, and primarily in the api functions, when creating http requests and retrieving response data from the back-end to ensure data was being sent and retrieved in the correct format, concluding the front-end implementation and leading into the one of the back-end server and database.

4.3 Back-end Server and Database

Natali Munk-Jakobsen

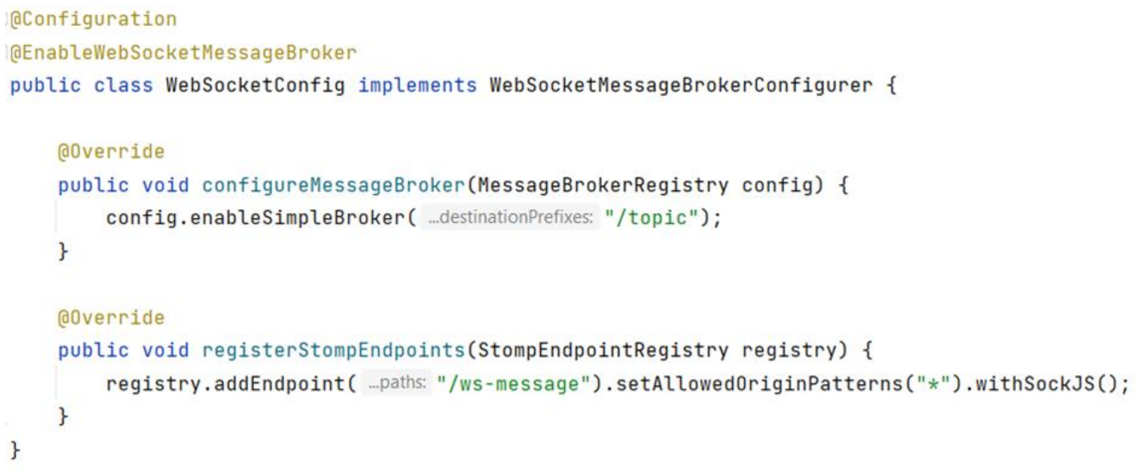
The presentation layer of the back-end is responsible for establishing the communication between the client and server. Three controller classes in the presentation layer contain methods to manage HTTP requests and WebSocket connections.

Controller classes are annotated with @RestController annotation which is used for making RESTful web services and allows the class to respond to requests made by the client. @RequestMapping is used with the class definition in controllers to create the base URI for the HTTPs requests.

AuthController responds to login requests from the admin. UserController and AdminController contain HTTP methods (GET, POST, PATCH, DELETE) to handle requests from the users and the admin. @GetMapping, @PutMapping, @PatchMapping, and @DeleteMapping request mapping annotations are used to map HTTPs requests with controller methods. Further information about the API can be found in the appendices (Appendix G - API Documentation).

All request-handling methods of the controller classes return a ResponseEntity object as a response. ResponseEntity represents the entire HTTP response including the status code, headers, and body. ResponseEntity is a generic type, therefore any type can be used as the response body.

Web Socket configuration is made in WebSocketConfig class implementing WebSocketMessageBrokerConfigurer interface from Spring Framework (Figure 4.13). This interface defines methods for configuring message handling with simple messaging protocols from WebSocket clients.



```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker( ...destinationPrefixes: "/topic");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/ws-message").setAllowedOriginPatterns("*").withSockJS();
    }
}
```

Figure 4.13. WebSocket Configuration

@EnableWebSocketMessageBroker annotation is used to configure the Web socket message broker to create STOMP endpoints for the WebSocket handshake and to send and receive messages. This annotation needs to be used in conjunction with the @Configuration annotation. A simple message broker is set to carry the messages to the client on destinations prefixed with “/topic” in the configureMessageBroker method. A STOMP over WebSocket endpoint at “/ws-message” is registered in the registerStompEndpoints method. In addition, SockJS fallback options are enabled for this endpoint, therefore alternative transports can be used in case the client does not support WebSockets natively.

An instance of SimpMessagingTemplate class from Spring Boot is created in controllers. This class provides methods for sending messages to a user through the WebSocket connection. ConvertAndSend method with destination and payload parameters is used for sending the actual intended message to the target destination.

Another annotation used in Controller classes is @Autowired. Each controller class is autowired to the related Service interface. In Spring Boot, @Autowired annotation is used for dependency injection and helps to auto-wire the collaborative beans. The principle of dependency injection is creating dependent objects outside of a class and injecting those objects into the class in different ways. Therefore, the creation and binding of the dependent objects are outside of the class that depends on them.

The Application Layer serves as an intermediary for data exchange between the presentation and the data access layers and consists of Service interfaces and their implementation classes. Service interfaces contain methods to store, retrieve, update and delete data. Service implementation classes are annotated with @Service annotation and autowired to the related repository interface.

The Data Access Layer consists of 7 repository interfaces extending the JpaRepository interface provided by Spring Data JPA. Spring Data Commons, a part of the Spring Data project, uses a set of repository interfaces to provide shared core infrastructure across the Spring Data modules. These interfaces, namely Repository, CrudRepository, and PagingAndSortingRepository, are technology-neutral and can be used with both relational and non-relational databases.

The Repository interface is a marker interface that captures the type of the entity and the entity’s id. The CrudRepository interface defines a repository that contains standard CRUD operations. The PagingAndSortingRepository extends the CrudRepository and adds findAll methods to sort and paginate the result. These two repositories are supported by other Spring Data projects, thus the methods in the interfaces can be used for different database systems. JpaRepository is an extension for Repository that also contains JPA-specific methods.

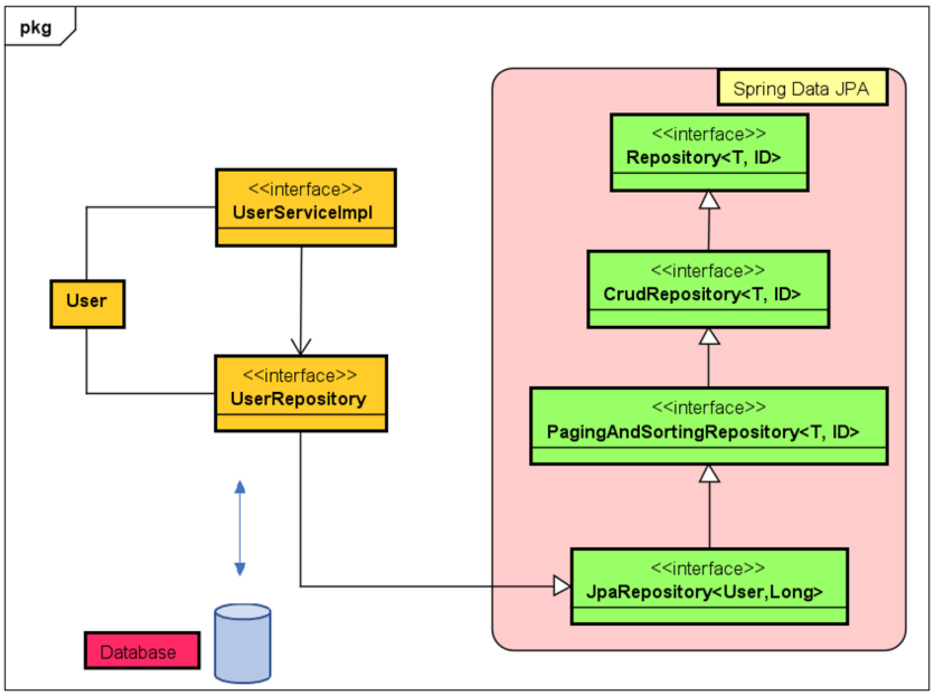
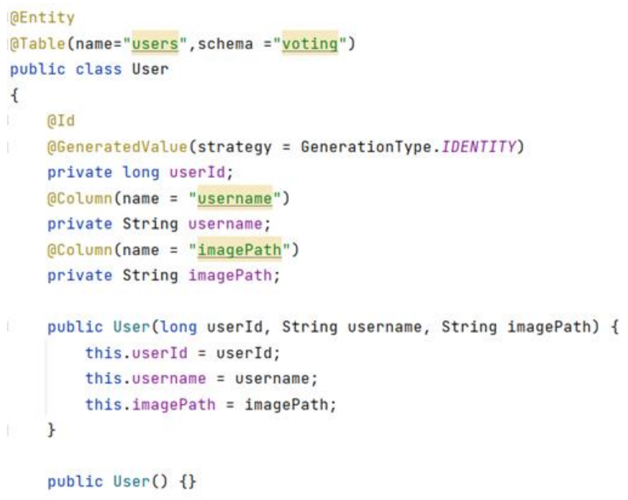


Figure 4.14. Implementation of Spring Data JPA for User

Figure 4.14 shows the implementation of Spring Data JPA for the User entity. User is a JPA entity mapped to the “users” table in the “voting” schema using @Entity and @Table annotations. In addition to an auto-incremented unique id, it has two more columns annotated with @Column. Moreover, a default no-args constructor was added to the class to fulfill the JPA specifications (Figure 4.15).



```
@Entity
@Table(name="users",schema ="voting")
public class User
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long userId;
    @Column(name = "username")
    private String username;
    @Column(name = "imagePath")
    private String imagePath;

    public User(long userId, String username, String imagePath) {
        this.userId = userId;
        this.username = username;
        this.imagePath = imagePath;
    }

    public User() {}
```

Figure 4.15. User JPA Entity

When the client sends a request to retrieve data, for example, a list of Users, the related controller method calls getAllUsers() method in the UserService interface. The implementation class (UserServiceImpl) is autowired to the UserRepository interface and calls findAll() from this interface. FindAll() is a method from CrudRepository retrieving all instances of the User type from the database.

The model package contains JPA entities and Data Transfer Objects. Therefore all three layers make use of this package for different purposes. CategoryRate, FinalCategory, FinalResult, PlayResult, Result, and Status classes are DTOs that are used only for transferring the data from the server to the client. These objects are created in service methods and define how the data will be sent over the network and to be presented to the client. They are useful when the data needed by the client is composed of many different entities and the presentation model needs all the data at once because they help to reduce the number of remote calls.

Besides Data transfer objects, the Model package has 8 JPA entities, each one is mapped to a database table. The Answer, Question, User, Vote, Contributor, and PlayInfo entity objects were also used to transfer the data between processes, thus additional DTO classes were not created for these objects.

4.4 Machine Learning

Daria-Maria Popa

For the implementation of the machine learning algorithm a separate service was made as part of the back-end server. The 'PredictionService' interface and its implementation class 'PredictionServiceImpl', contain four methods, showing the prediction lifecycle throughout any particular play: 'generatePredictions', 'werePredictionsGenerated', 'getPredictionForUser', and 'clearPredictions'.

The main method and the first one to be called by the system during a play is the 'generatePredictions'. The method header takes the number of questions as a parameter, which is used in order to prepare the prediction data matrix.

The implementation class of the service autowires the vote, user, and answer repository, which are used to get all the users, gather their vote data, and for each vote get the answer text in order to determine if the user has chosen the first or the second option, and represent it as a 0 or 1 in the matrix.

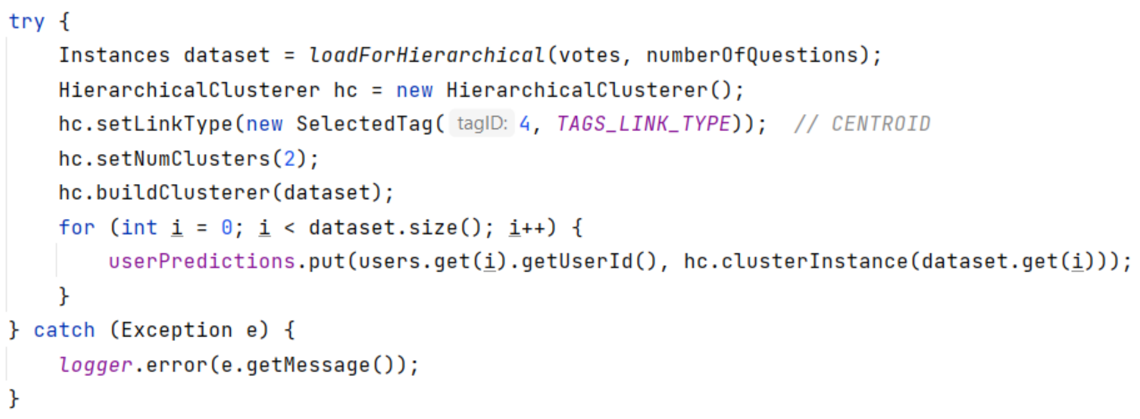
The resulting votes matrix represents an Integer matrix with a row of length equal to the number of questions in the play for each player, with only 0's and 1's saying if the user has voted the first or the second option for the respective question. For questions where the user has not voted, the 0 option was set as default.

This data is then formatted in a Weka-specific ARFF dataset format (Attribute-Relation File Format (ARFF), 2002), containing the dataset name, a list of attributes matching the data features to be analyzed by the model, and the data value represented in this case by the user's vote choices.

The hierarchical clusterer is set to have two final clusters, related to the two possible final answer choices. An important part when calculating the clusters is determining which data points are "close" and can be grouped together. This was set using a linkage of type 'Centroid' (weka-dev-3.9.6 API, 2022). The centroid linkage calculates the distance between the clusters based on their geometric center and decides what values are part of which cluster based on their relative distance to these centroid points (Figure 4.16). To calculate the distance between two data points, the default Euclidean distance function (weka-dev-3.9.6 API, 2022) was used on the model (Figure 4.16).

The reason for choosing this linkage type is closely related to the type of data. Since the data represents votes on political subjects, the centroid linkage is a good method of calculating and displaying how these political views influence each other and how, although there can be political 'factions' represented in this dataset by slight differences in voting patterns, they all join into bigger coalitions (i.e. 'left-leaning and 'right-leaning' political parties) (Choosing the right linkage method for hierarchical clustering, 2016).

After the model is built on the ARFF dataset, the resulted labels are saved in a HashMap together with the id of the user (Figure 4.16). The labels represent the predictions for the final answer for all the users as either a 0 or a 1. The hierarchical clustering groups all the users into either group 0 or group 1 based on similarities from their previous vote choices calculated using the set linkage-type, where in the end group 0 represents that the user is more likely to choose the first option based on previous choices, and group 1 is the opposite. The 'agglomerative' part of the hierarchical clustering model refers exactly to this many-to-x-clusters grouping.



```
try {
    Instances dataset = loadForHierarchical(votes, numberOfQuestions);
    HierarchicalClusterer hc = new HierarchicalClusterer();
    hc.setLinkType(new SelectedTag( tagID: 4, TAGS_LINK_TYPE));  // CENTROID
    hc.setNumClusters(2);
    hc.buildClusterer(dataset);
    for (int i = 0; i < dataset.size(); i++) {
        userPredictions.put(users.get(i).getUserId(), hc.clusterInstance(dataset.get(i)));
    }
} catch (Exception e) {
    logger.error(e.getMessage());
}
```

Figure 4.16. Creating hierarchical clustering model

As the generatePredictions method is called when the Admin requests the last question from the repository, creating and running the model is wrapped in a try-catch block, and the error is logged using a standard Java logger, so as not to affect the retrieval of the question data for the admin.

The 'werePredeictionsGenerated' is used to check for the case when the prediction model failed to run, and the 'getPredictionForUser' uses the 'getOrDefault' method to check for a prediction label linked to a certain user id or return the default value 0.

Lastly, when the Admin requests to end the play, the controller method calls upon the 'clearPredictions' (among other service calls) in order to clear the map of any user data and have it ready for future plays.

Since the data for a play is low dimensional data with four-five questions (features) for each play, and the dataset for each play is really small with approximately

30-50 players, the benefits and efficiency of running a machine learning model compared to implementing an algorithm to calculate previously chosen answers and their underlying categories and make the predictions based on that are minimal to almost non-existent (High dimension and low dimension data science, 2017).

Therefore, for the cases when the machine learning model would not work (i.e. a single user in the system) and as an overall second option to generating the predictions for the final vote, a 'getPredictedAnswer' method was implemented, to manually calculate and return a predicted answer for the user based on previously chosen answer categories.

The method was added as a part of the PredictionService and it gathers the user's vote data based on the provided user id in the method header, and uses it to create two arrays 'primaryCategories' and 'secondaryCategories', based on what answers the user has previously chosen and what primary and secondary categories these had.

A hash map is used to store key-value pairs of the category name and its weighted frequency sum in the user's votes. The weighted category value is calculated as the number of times the category was voted on as a primary category summed with the frequency of the category as a secondary category divided by two (Figure 4.17).

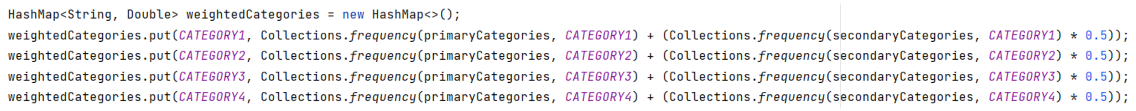

```
HashMap<String, Double> weightedCategories = new HashMap<>();
weightedCategories.put(CATEGORY1, Collections.frequency(primaryCategories, CATEGORY1) + (Collections.frequency(secondaryCategories, CATEGORY1) * 0.5));
weightedCategories.put(CATEGORY2, Collections.frequency(primaryCategories, CATEGORY2) + (Collections.frequency(secondaryCategories, CATEGORY2) * 0.5));
weightedCategories.put(CATEGORY3, Collections.frequency(primaryCategories, CATEGORY3) + (Collections.frequency(secondaryCategories, CATEGORY3) * 0.5));
weightedCategories.put(CATEGORY4, Collections.frequency(primaryCategories, CATEGORY4) + (Collections.frequency(secondaryCategories, CATEGORY4) * 0.5));
```

Figure 4.17. Manual calculation of user prediction

This is based on the concept of primary and secondary categories for each answer, as developed together with the company stakeholder, where a primary category would value 1 and a secondary category would value 0.5 when calculating the prediction. This weighted sum was not implemented for the machine learning model since machine learning models work with categorical data and the weight would prove redundant in the final prediction (Lakshmanan, 2019).

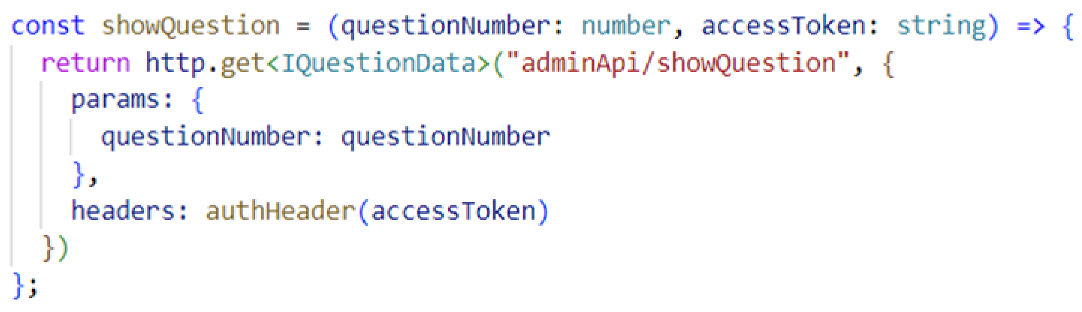
After populating the ‘weightedCategories’ hash map, the predicted answer for the last question was chosen based on the category with the biggest weighted sum, or in the case of all sums being equal based on the primary category of the last question in order to mimic the user’s latest behavior.

Overall, both prediction methods offer a reliable solution to generating the users’ last vote prediction, with similar metrics in terms of performance, and whose overall results and accuracy will be analyzed further in the following chapter of the report.

4.5 Implementation Overview

Natali Munk-Jakobsen

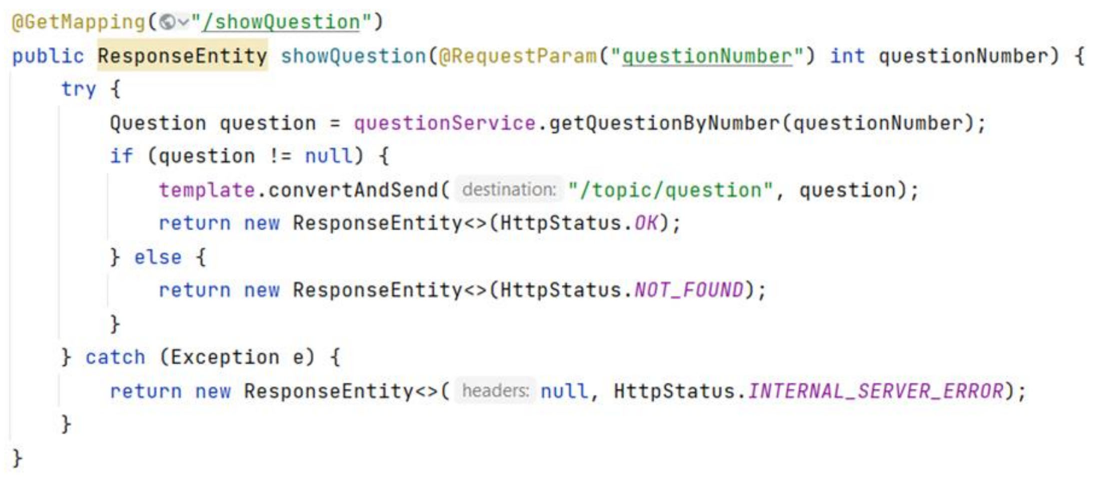
The complete path through the system can be better illustrated through the example: “The admin sends a question to the user screen”. The admin logs in to the system using valid credentials and clicks “Start” to start the voting period. In order to show a new question the admin clicks the “Show Question” button. The onClick event of the button executes a method to call the relevant function in the AdminApi. ShowQuestion function (Figure 4.18) has two parameters, question number, and access token, and it sends a GET request to the server using the question number as a parameter. Since AdminApi in the server requires JWT authentication, the token is provided in the HTTP header.



```
const showQuestion = (questionNumber: number, accessToken: string) => {
  return http.get<IQuestionData>("adminApi/showQuestion", {
    params: {
      questionNumber: questionNumber
    },
    headers: authHeader(accessToken)
  })
};
```

Figure 4.18. ShowQuestion function in AdminApi

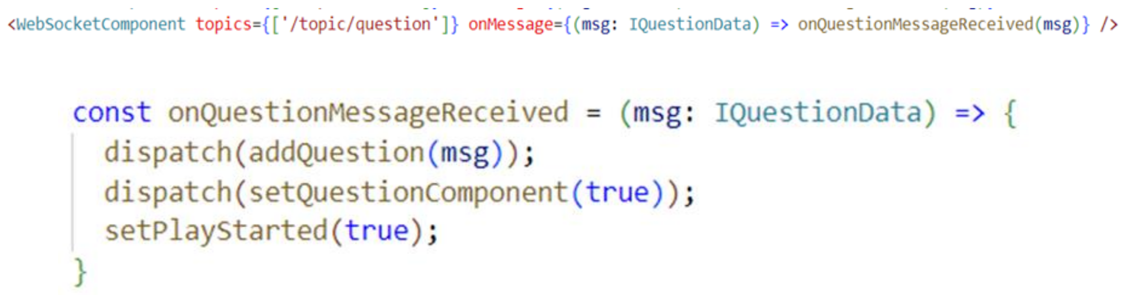
ShowQuestion method in AdminController (Figure 4.19) responds to GET request made by the client and retrieves the Question by request parameter, and question number, calling relevant methods in QuestionService and QuestionRepository, respectively. findByQuestionNumber method in QuestionRepository calls a JPA finder method for the QuestionNumber field and the data is retrieved from the database.



```
@GetMapping(©v"/showQuestion")
public ResponseEntity showQuestion(@RequestParam("questionNumber") int questionNumber) {
    try {
        Question question = questionService.getQuestionByNumber(questionNumber);
        if (question != null) {
            template.convertAndSend( destination: "/topic/question", question);
            return new ResponseEntity<>(HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    } catch (Exception e) {
        return new ResponseEntity<>( headers: null, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Figure 4.19. ShowQuestion method in AdminController

ConvertAndSend method from SimpMessagingTemplate class is used to send the Question object to “topic/question” destination. All clients listening to the topic receive the message and the onQuestionMessageReceived function is called. This function modifies Question and Component states in the store and the Question component in the MainPage component is shown to display the question and answers on the screen (Figure 4.20).



```
<WebSocketComponent topics={['/topic/question']} onMessage={(msg: IQuestionData) => onQuestionMessageReceived(msg)} />
```

```
const onQuestionMessageReceived = (msg: IQuestionData) => {
  dispatch(addQuestion(msg));
  dispatch(setQuestionComponent(true));
  setPlayStarted(true);
}
```

Figure 4.20. OnQuestionMessageReceived function

When the Question component is displayed, the timer countdown starts, and the user is allowed to vote. The user selects one of the answers and clicks the “Confirm” button, a vote object is created and a request to post the vote is sent to the server. A response is received from the server and WaitingPage Component is shown on the screen until the vote result is sent by the server.

Deployment of the system is done by using the Microsoft Azure SQL Database, Azure App Service, and Azure Static Web Apps. Further information about the deployment process can be found in the appendices (Appendix I - Technical User Guide).

5 Test

Bogdan Mezei

The purpose of this section is to describe the test plan, the different strategies involved in testing, and the requirement traceability matrix, describing if the functional requirements have been fulfilled. The testing has been planned and executed with integration, acceptance, and system testing in mind. Automated tests with unit and end-to-end tests as well as manual tests have been executed on the system. These different test specifications will be described in detail in the following chapter.

5.1 Test Specifications

5.1.1 Unit Testing

In order to ensure the quality and reliability of both the backend and the front-end, a suite of unit tests has been created. These tests are designed to exercise individual units of the code and check that they are working as expected. The tests are written in Java and use the JUnit testing framework to define and run the tests. The test suite has been designed to be comprehensive, covering a wide range of scenarios and edge cases to ensure that all aspects of the code are thoroughly tested.

The unit tests are run automatically using GitHub Actions. This allows for continuous integration and testing of the codebase, ensuring that any changes made to the code are automatically tested and any errors or bugs are quickly identified and fixed. The tests are run automatically whenever a new commit is pushed to the code repository, and the results are reported in the GitHub Actions dashboard. All the unit tests have been run at the end of the development period and all of them have passed ensuring the good quality of the system.

This allows developers to see the status of the tests and quickly identify any issues that need to be addressed. Overall, the use of GitHub Actions for automatically running the unit tests has been effective in helping to ensure the quality and reliability of the code.

5.1.2 Integration Testing

Integration testing was used in order to verify the functioning of the integrated components or systems. This technique was employed to identify any conflicts or issues that may arise when the various components of the system were combined, and to ensure that the different parts of the system worked together as a whole. The use of integration testing allowed for the simulation of real-world scenarios and the observation of the system's response in these scenarios.

This technique involved the use of automated end-to-end tests, to simulate the actions of a real user and to test the various components of the system. The use of this type of testing helped to reduce the time and effort required for manual testing, as well as providing more accurate and consistent test results. This led to the identification of potential issues more quickly and efficiently, allowing for a more effective and efficient integration testing process.

5.1.3 End-to-end Testing

End-to-end testing for the React frontend application has been performed using Puppeteer and runs automatically using GitHub Actions. This type of testing involves simulating real-world scenarios and user interactions with the application and checking that the application behaves as expected.

The tests are written in TypeScript and use Puppeteer to control a headless version of the Chrome web browser. The tests are run automatically using GitHub Actions, allowing for continuous integration and testing of the frontend code. The results of the end-to-end tests are reported in the GitHub Actions workflow.

End-to-end testing has been structured by creating different test suites encompassing all features of the application. Each testing suite starts by resetting the application to the same state and ends by disabling the application in order to ensure consistency.

These are the selected features for the system:

- ❖ Feature 1 - Questions database and display
- ❖ Feature 2 - Voting System
- ❖ Feature 3 - Voting Rounds
- ❖ Feature 4 - User Personalization
- ❖ Feature 5 - Contributions Page
- ❖ Feature 6 - Answer Prediction
- ❖ Feature 7 - Personal Voting Results
- ❖ Feature 8 - Downtime Management

Out of all of these features we have created the following test suites:

- ❖ Questions database and display
  - ➢ In this suite, the test logs in as admin, goes to the “Manage Questions” screen, checks existing questions, and performs operations for adding, deleting and editing questions.
- ❖ Voting System
  - ➢ In the second suite, the test logs in as admin in order to control the flow of the play while opening another tab as user. It checks the voting system fully, answering questions sent on the admin side and checking that the results screen is correct. This suite is also responsible for checking the answer prediction and downtime management for the admin. Within this test suite, features 2, 3, 6, 7 and 8 are encompassed.

- ❖ User Personalization
    - ➢ In the following suite, the test joins the system as an user and creates a profile by selecting an username and profile picture and checking them afterwards.
- ❖ Contributions Page
    - ➢ In this last suite, the test logs in as admin and goes to the “Manage Contributors” page and verifies the contributor management functionality.

5.1.4 System Testing

System testing has been done manually based on the use case descriptions and requirements of the application. In the following section, test suites have been created for each use case description, along with multiple test cases for each suite. This section concludes with the requirement traceability matrix which is used as proof that each requirement has been met. The steps of each test case are numbered according to the numbering of the steps within the use case so that they are easily traceable.

The web application was tested on both iPhone and Android devices using multiple tabs and clients. The testing was performed to ensure that the application was compatible with different devices and could handle multiple users accessing it simultaneously.

The following test suites have been created in accordance with the use case descriptions:

- ❖ Join Play
- ❖ Vote
- ❖ Manage Play Data
- ❖ Manage Play Flow

The following table is an example of a test suite, and it contains the first three test cases. For the complete test suites and test cases refer to the appendix (Appendix H - System Testing).

5.1.4.1 Test Suite 1: Join Play

Test Case ID	Description	Steps	Test Data	Expected Result	Actual Result	Pass/ Fail
TC101	User joins the play	1. User arrives at the place of the theater play 2. Open web application 3. Click “JOIN” 4. User is shown the agreement 5. Click “AGREE”	None	User gets taken to the account creation screen	As expected	Pass
TC102	User creates profile	6. User is on the profile creation screen 7. User inputs username and selects picture 8. User clicks “CREATE”	Username = “Alice”	System confirms User has joined and displays waiting status message in anticipation of the voting rounds	As expected	Pass
TC103	User joins the play and creates profile, admin ends the play at any point	*Steps 1-8* Admin ends the play at any point	Username = “Alice”	User state is cleared and inactive page is shown	As expected	Pass

Table 5.1. Test Suite 1: Join Play

5.1.4.2 Requirement Traceability Matrix

A requirement traceability matrix was used to ensure that all requirements were properly traced throughout the development process. This allows for easy identification of any missing or untraced requirements, as well as ensuring that all requirements are properly implemented and tested in the final product.

The following table is an example made for the first requirement. For the complete requirement traceability matrix refer to the appendix (Appendix H - System Testing).

Requirement #	Description	Test case ID	Status
1	As a user, I want to be able to cast my vote so that I can influence the outcome of the play	TC201 TC202 TC203 TC204 TC205 TC206 TC207 TC208	TC201 - Pass TC202 - Pass TC203 - Pass TC204 - Pass TC205 - Pass TC206 - Pass TC207 - Pass TC208 - Pass

Table 5.2. Requirement Traceability Matrix

5.1.5 Acceptance Testing

Acceptance testing was performed for our application in order to verify that it met the acceptance criteria defined by the customer. This testing was carried out by a representative of the theater company, who used the application during the premiere to ensure that it met their functional and non-functional requirements. The goal of this acceptance testing was to ensure that the application was fit for use by the theater company and met their needs.

The stakeholders used the application during rehearsals and provided feedback on its functionality and performance. This feedback was used to identify any issues or improvements that needed to be made before the application could be considered ready for use.

There have been a total of 3 rehearsals where the system has been fully tested with the actors in a practical scenario. Following these, the acceptance test was performed, which was a premiere in front of a live audience of around 40 people where the system has been utilized. This test has been confirmed and followed by a feedback session with HumanLab. The acceptance testing was successful, and the company gave their approval for the application to be potentially used in their upcoming performances (Appendix H - Acceptance Testing).

6 Results and Discussion

Natali Munk-Jakobsen

'TRIC - The Right Choice' project has been carried out following the demands of the stakeholders. The system has been designed in order to fulfill the 18 functional and 14 non-functional requirements. Four use cases have been structured based on the functional requirements and final results based on the conducted testing stated in the list below:

- The users can join the voting process after accepting the data processing agreement and personalizing their user profile by selecting username and icon.
- The admin can log in using valid administrator credentials and manage play information and questions.
- The admin can manage the play flow using the admin console.
- The admin can display the questions and results both on the admin result screen shown to the theater crew and the user screen.
- The users can cast vote for the question on the screen and see the vote result on their screens.
- The users can see their final voting profile and download the profile as a .png image.
- A .json file holding the final voting results for the play is created and downloaded when the admin ends the session.
- The predicted answer for the last question is created using the chosen prediction model (either machine learning or manual one).

The system has been implemented as a Client – Server application as planned and the connection between the client and server established with HTTP and WebSocket connections. Observed results of the acceptance test outlined connection issues when the system is run in areas with poor mobile service and not complying with the top priority non-functional requirement as listed in the Analysis chapter of the report.

Back-end proved stable during the different testing runs, with good performance when analyzing and responding to user requests. JWT authentication mechanism was used to allow the admin user to access the restricted part of the system, represented by the AdminAPI.

For the front-end part of the application, the response time is in an acceptable time range and the navigation between components works as intended. User-led acceptance testing results showed the UI design as intuitive.

The system is accessible from all mobile operating systems, with the need for improvement for a certain feature of the application that is not available on all web browsers on iOS devices: having permission to download the final profile image from a non-Safari browser.

The application and the database have been hosted in the cloud using Microsoft Azure Services. Azure hosting plan and pricing plan were chosen considering the current needs of the stakeholder and proved adequate throughout the implementation period, but this can be changed to be in line with future demands if needed.

7 Conclusions

Daria-Maria Popa

The ‘TRIC’ project has highlighted and analyzed the problem domain at hand, the resulting functional and non-functional requirements have been recorded and used to create the basis of the project design as represented by the domain model, the design drafted and its implementation documented and tested through multiple test methods, concluded with the acceptance test ran in a real theater play scenario.

The overall conclusion of the project can be detailed as follows:

❖ The user interface has been created considering the principles of emotional and cognitive design and it provides an intuitive and aesthetically pleasing user experience.

❖ The social aspect of the design has been followed and implemented successfully, with the acceptance testing proving so.

❖ The three-layered architecture of the back-end server corresponds to the analysis and design stages of the project and ensures flexibility, maintainability, and scalability.

❖ Interfaces connected layers and the structure complies with the Separation of Concerns and Dependency Inversion Principles and the back-end API has an acceptable response time.

Overall, the architectural structure, database management, networking, security, and user interface development have been implemented as described in the design section of this report.

The test results of the implemented system conclude that all the functionalities described in the functional requirements and use case descriptions in the analysis section have been successfully implemented and that the system performs as intended and specified by the non-functional requirements.

In conclusion, the project has been successfully completed, fulfilling all the set requirements, appealing to the intended audience, and, providing an augmented voting tool for the democratic space described by the artistic play.

8 Project future

Daria-Maria Popa

The final acceptance test of the system has been completed and marked as accepted by the company stakeholder, and the project, therefore, labeled as ready for production. However, several technical changes could be implemented in order to improve the product's efficiency, both in terms of performance and user social interaction and experience.

Firstly, a key takeaway from the acceptance test, as described in the Results and Discussion chapter, was the importance of adequate mobile service during the play. Although this non-functional requirement will still remain a priority during the set-up of future play, some system changes could take place in order to accommodate for the cases in which particular messages are not received by the users due to package loss.

These changes could include having a polling system in place for client-server communication instead of a simple topic listener, as it exists right now. This would mean the server could notify the connected web socket clients several times a second that a message was sent, such as a new question to be voted on, and the client would therefore have more chances to hear that something changed and update the state accordingly.

Secondly, for better scalability on iOS operating mobile devices, a future plan could include working with receiving the permissions for downloading images on non-Safari browsers, making the functionality of the 'digital souvenir' shown at the end of the play overall more accessible for the players, as it currently is on the other mobile operating systems.

Thirdly, the option for multiple admin logins during a play could be disabled from the system to ensure that, although the intended plan is to only have one admin in the system for now, if in the future multiple admin accounts are used they cannot affect the play flow for each other.

Similar to the possibility of multiple admin accounts in the future, the database system has also been designed to support more than 2 answers per question, and for the future of the project, this could be further expanded upon in the front-end and prediction model, to allow the company to ask more complex questions and receive in exchange a wider variety of answers.

Lastly, although the user icon has been implemented as a static representation of the user as a Greek statue, a possible future spin-off could focus on making the icon more dynamic, either by allowing for user-uploaded images or by modifying the icon features as the user progresses through answering the questions of the play, as a way to visualize how these answers shape their ‘digital twin’.

Overall, the are several improvement ideas for the future of the project, as presented above, which would enhance both the system’s management and behavior, and its response and interaction with the user.

9 Sources of information

- Hidalgo C., 2018, A bold idea to replace politicians. Available at: https://www.peopledemocracy.com/ [Accessed March 25, 2022].

- IDEA, 2016, ICTs In Elections Database, International Institute for Democracy and Electoral Assistance. Available at: https://www.idea.int/data-tools/question-view/742 [Accessed March 25, 2022].

- Roser M. and Herre B., 2013. Democracy. Published online at OurWorldInData.org. Available at: https://ourworldindata.org/democracy [Accessed March 25, 2022].

- Solijonov A., 2016. Voter Turnout Trends around the World, International Institute for Democracy and Electoral Assistance. Available at: https://www.idea.int/sites/default/files/publications/voter-turnout-trends-around-the-world.pdf [Accessed March 25, 2022].

- Tech for Democracy Conference, 2021, Available at: https://techfordemocracy.dk/ [Accessed March 25, 2022].

- Bucher, J., 2018. *Storytelling For Virtual Reality*. NY: Routledge.

- Morris, S., 2014. *Theatre, Democracy And Shakespeare | The Shakespeare Blog*. [online] Theshakespeareblog.com. Available at: <http://theshakespeareblog.com/2014/02/theatre-democracy-and-shakespeare/> [Accessed 26 March 2022].

- Gonzaga.edu. 2019. *Mystery Of Edwin Drood | Gonzaga University*. [online] Available at: <https://www.gonzaga.edu/news-events/stories/2021/11/1/mystery-of-edwin-drood> [Accessed 26 March 2022].

- Anon 2022. *ABOUT HUMANLAB.* [online] HUMANLAB. Available at: <https://www.humanlab.studio/about.html> [Accessed 20 Nov. 2022].

- Larman, C., 2004. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development

- Britton, J., 2021. *What Is ISO 25010?* [online] What Is ISO 25010? Available at: <https://www.perforce.com/blog/qac/what-is-iso-25010> [Accessed 27 Nov. 2022].

- ❖ Faus, A., 2021. *How to write SMART goals (with examples)*. [online] Work Life by Atlassian. Available at: <https://www.atlassian.com/blog/productivity/how-to-write-smart-goals> [Accessed 27 Nov. 2022].

- ❖ Cohn, M., 2019. *Why the Three-Part User Story Template Works So Well*. [online] Why the Three-Part User Story Template Works So Well. Available at: <https://www.mountaingoatsoftware.com/blog/why-the-three-part-user-story-template-works-so-well> [Accessed 27 Nov. 2022].

- ❖ Anon 2019. *Informal Semantics for UML Use Case Diagrams*. [online] Informal Semantics for UML Use Case Diagrams. Available at: <https://cs.uwlax.edu/~mzheng/CS743Fall19/UseCaseDiagrams.html> [Accessed 4 Dec. 2022].

- ❖ Cockburn, A., 2000. *Writing Effective Use Cases*. Agile Software Development Ser. https://doi.org/10.1604/9780201702255.

- ❖ Fakhroutdinov, K., 2022. *Unified Modeling Language (UML) description*. [online] Unified Modeling Language (UML) description. Available at: <https://www.uml-diagrams.org> [Accessed 5 Dec. 2022].

- ❖ Intersoft Consulting, 2016. *General Data Protection Regulation (GDPR) – Official Legal Text*. [online] General Data Protection Regulation (GDPR). Available at: <https://gdpr-info.eu/> [Accessed 5 Dec. 2022].

- ❖ Anon 2021. *Documentation - DOM Manipulation*. [online] TypeScript: Documentation - DOM Manipulation. Available at: <https://www.typescriptlang.org/docs/handbook/dom-manipulation.html> [Accessed 5 Dec. 2022].

- ❖ Herbert, D., 2022. *What is React.js? (Uses, Examples, & More)*. [online] What is React.js? (Uses, Examples, & More). Available at: <https://blog.hubspot.com/website/react-js> [Accessed 5 Dec. 2022].

- ❖ typescriptlang, 2022. *Why does TypeScript exist?* [online] TypeScript: Why does TypeScript exist? Available at: <https://www.typescriptlang.org/why-create-typescript> [Accessed 5 Dec. 2022].

- ❖ Fathima, S., 2021. *Functional Programming VS Object Oriented Programming (OOP) Which is better….?* [online] Medium. Available at:

<https://medium.com/@shaistha24/functional-programming-vs-object-oriented-programming-oop-which-is-better-82172e53a526> [Accessed 8 Dec. 2022].

- Anon 2022. *Redux*. [online] Redux. Available at: <https://redux.js.org/> [Accessed 6 Dec. 2022].

- Anon 2022. *Hooks Pattern*. [online] Hooks Pattern. Available at: <https://www.patterns.dev/posts/hooks-pattern/> [Accessed 6 Dec. 2022].

- Taranto, J., 2022. *A powerful React + Redux Toolkit pattern (reuseable state slices) | PreviousNext*. [online] A powerful React + Redux Toolkit pattern (reuseable state slices) | PreviousNext. Available at: <https://www.previousnext.com.au/blog/powerful-react-redux-toolkit-pattern-reuseable-state-slices> [Accessed 6 Dec. 2022].

- auth0.com, 2021. *JWT.IO - JSON Web Tokens Introduction*. [online] JSON Web Token Introduction - jwt.io. Available at: <http://jwt.io/> [Accessed 3 Dec. 2022].

- Strukchinsky, V., 2022. *BEM — Introduction*. [online] BEM — Introduction. Available at: <https://getbem.com/introduction/> [Accessed 8 Dec. 2022].

- spring.io, 2022. Spring Boot. [online] Spring Boot - Overview. Available at: <https://spring.io/projects/spring-boot> [Accessed 28 Nov. 2022].

- Martin, Robert C., 2003. Agile Software Development, Principles, Patterns, and Practices.

- Evans, E., 2004. Domain-driven Design: Tackling Complexity in the Heart of Software.

- hibernate.org, 2022. Hibernate ORM. [online] Idiomatic persistence for Java and relational databases. Available at: <https://hibernate.org/orm/> [Accessed 28 Nov. 2022].

- spring.io, 2022. Spring Data JPA. [online] Spring Data JPA - Overview. Available at: <https://spring.io/projects/spring-data-jpa> [Accessed 29 Nov. 2022].

- azure.microsoft.com, 2022. Azure SQL Database. [online] Build apps that scale with managed and intelligent SQL database in the cloud. Available at: <https://azure.microsoft.com/en-us/products/azure-sql/database/> [Accessed 29 Nov. 2022].

- Frank, E., Hall, M.A., Pal, C.J. and Witten, I.H., 2016. *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.
- Anon 2022. *Unsupervised Machine learning - Javatpoint*. [online] www.javatpoint.com. Available at: <https://www.javatpoint.com/unsupervised-machine-learning> [Accessed 6 Dec. 2022].
- Anon 2002. *Attribute-Relation File Format (ARFF)*. [online] Attribute-Relation File Format (ARFF). Available at: <https://www.cs.waikato.ac.nz/ml/weka/arff.html> [Accessed 8 Dec. 2022].
- Anon 2022. *Hierarchical Clustering in Machine Learning - Javatpoint*. [online] www.javatpoint.com. Available at: <https://www.javatpoint.com/hierarchical-clustering-in-machine-learning> [Accessed 6 Dec. 2022].
- Anon 2022. *redux-persist*. [online] redux-persist - npm. Available at: <https://www.npmjs.com/package/redux-persist> [Accessed 7 Dec. 2022].
- Anon 2022. *Hooks at a Glance – React*. [online] Hooks at a Glance – React. Available at: <https://reactjs.org/docs/hooks-overview.html> [Accessed 7 Dec. 2022].
- Lakshmanan, L., 2019. *Why feature weights in a machine learning model are meaningless*. [online] Medium. Available at: <https://towardsdatascience.com/why-feature-weights-in-a-machine-learning-model-are-meaningless-b0cd22a4c159> [Accessed 8 Dec. 2022].
- Anon 2016. *Choosing the right linkage method for hierarchical clustering*. [online] Cross Validated. Available at: <https://stats.stackexchange.com/questions/195446/choosing-the-right-linkage-method-for-hierarchical-clustering> [Accessed 8 Dec. 2022].
- Anon 2022. *weka-dev-3.9.6 API*. [online] weka-dev-3.9.6 API. Available at: <https://weka.sourceforge.io/doc.dev/index.html?weka/clusterers/HierarchicalClusterer.html> [Accessed 8 Dec. 2022].
- Anon 2017. *High dimension and low dimension data science*. [online] Pertinent Observations. Available at: <https://www.noenthuda.com/2017/04/07/high-dimension-and-low-dimension-data-science/> [Accessed 8 Dec. 2022].

- Norman, D.A., 2004. Emotional Design. Why We Love (or Hate) Everyday Things

- Donnelle Weller, 2004, The Effects of Contrast and Density on Visual Web Search, https://researchinuserexperience.wordpress.com/2004/07/12/the-effects-of-contrast-and-density-on-visual-web-search/

10 Appendices

Appendix A Project Description

Appendix B Requirements

Appendix C Use Case Descriptions

Appendix D Diagrams

Appendix E Code Base

Appendix F Figma Design

Appendix G API Documentation

Appendix H Testing Documentation

Appendix I User Guides

Appendix J Scrum Documentation

Appendix K Company Meetings

Appendix L Group Contract

Process Report

Students:

Bogdan-Alexandru Mezei (293137)

Daria-Maria Popa (293087)

Natali Munk-Jakobsen (293132)

Supervisor:

Henrik Kronborg Pedersen

Number of characters: 30967

Software Engineering

7th Semester

Date: 14/12/2022

Table of contents

1 Introduction

Daria-Maria Popa

This project has been conducted together with an external company, namely HumanLab, and aimed to achieve a live audience voting tool that could be used during HumanLab's latest show, 'TRIC - The Right Choice'.

The process of this project has been structured using SCRUM and Unified Process. This methodology was chosen both due to the team's experience using it and for its organized structure when working iteratively, being able to observe more clearly the current status of the work and allowing the team to make crucial choices early on during the planning.

The process has been split into two parts, referred to as BPR1 and BPR2, with the first part representing the initiation of the project and setting the ground understanding of the problem domain, and the second part handling everything else until delivery. The initial part has been split into 3-week sprints, totaling 4 sprints and focusing solely on the Inception phase of the project (Appendix A), with a reduced team capacity. This document will focus on the BPR2 process.

For the second part, the sprint length has been decided together with the team as 1 week, starting and ending on the Wednesday of each week, with the last sprint ending two days before the final project deadline.

The overall length of the second part of the project was described as 14 sprints (14 weeks), split into the 4 phases of the Unified Process (Inception, Elaboration, Construction, and Transition), with the first and last phase being the shortest at 2 sprints each, 3 sprints assigned for Elaboration, and the rest going for the longest and most complex phase - 7 sprints for Construction.

All SCRUM-related meetings such as the Sprint Planning, Sprint Review, and Daily Meetings (Appendix J) have greatly contributed to the progress and reaching of the readiness goal of the product. The Sprint Planning and Sprint Retrospective have been planned to be held on a weekly basis, each Wednesday, with an approximate duration of 1 hour each. The planning for each week has been done with consideration of the team's capacity (Appendix J – TeamCapacity). Daily meetings were planned in the form of 10-15 minutes of daily stand-ups, sharing the plan, blocking issues, and overall impression of the project progress throughout the team (Appendix J - DailyScrumPlan).

Company meetings have been scheduled together with the company representatives, with the rehearsals and final acceptance test dates being set early in the process, giving the team exact goals to aim for. Weekly 1-hour progress meetings have been planned towards the end of Construction and the beginning of the Transition phase as well, to ensure the alignment of the final product with the company goals (Appendix J).

Supervisor meetings have been structured as weekly 1-hour meetings as well, starting from the first sprint of the project. The latter sprints have been supplemented with extra meetings when needed.

Overall, the progress of the project has been closely monitored by all involved and aforementioned parties with the use of the specified process methods and methodologies, ensuring timely and qualitative delivery of the product.

2 Group Description

Bogdan Mezei

The group for this project is formed of three people, namely Bogdan, Daria, and Natali. We have chosen to work in this format not only because of prior experience working together but also due to our shared interests and common ideas for the project theme. Our team, which has been named “Team cake”, consists of two nationalities: Romanian and Turkish.

Because this is a multinational team, we can apply Hofstede’s model of cultural dimensions in order to analyze it, focusing specifically on our cultural differences and norms, understanding what they mean and how to use them to our advantage. This theory identifies 6 dimensions of cultures than can help distinguish multiple cultures from each other, each dimension level being scored on a scale from 0 to 100 (see ‘Figure 1: Hofstede cultural Dimensions’).

In our group’s case, the cultural dimension model shows a generally small cultural gap for most values, and only distinguishes some potential differences in power distance and indulgence levels. The dimension levels serve as an aid towards concluding what the team's strengths and weaknesses are, as well as deciding on some focus points and rules for team organization and communication.

For ‘Team Cake’ there is a good common understanding of the importance of maintaining group harmony (low individualism), achieving things through teamwork (low masculinity), settings clear goals and rules (high uncertainty avoidance), and valuing the balance between tradition and change (intermediate-term orientation), which means the group is well suited for working together in a well-organized group.

The power distance difference means that, although both values are high, one side of the team might be more inclined towards having a hierarchical distribution of power than the other, and therefore find it more difficult to initiate change or act without guidance. Related to this hierarchical culture, the Romanian indulgence level also indicates a more restrained behavior in professional settings.

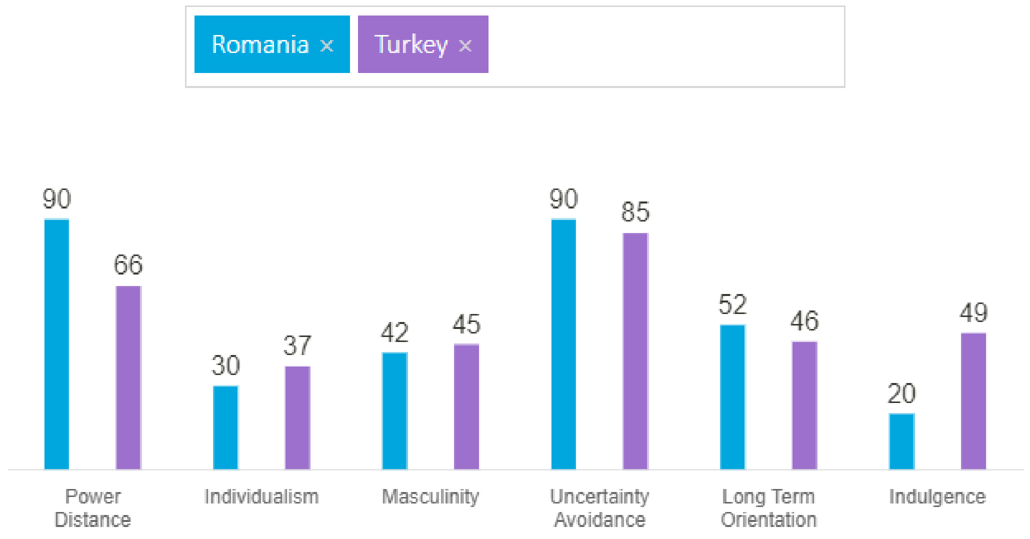


Figure 1: Hofstede Cultural Dimensions

This points towards a group need for an informal work setting, where distributing the power, discussing change, and setting task priorities is more productive due to everyone being involved in an equal manner.

Overall, applying Hofstede's theory enforced the need for our group to go for a flexible but well-delimited work approach, with a focus on balancing the power distribution, prioritizing tasks, and having a good balance of new and known technologies, which also led to our choice of an agile way of system development methodology.

While creating the team we have also taken into consideration the different E-stimate colors for group formation theory.

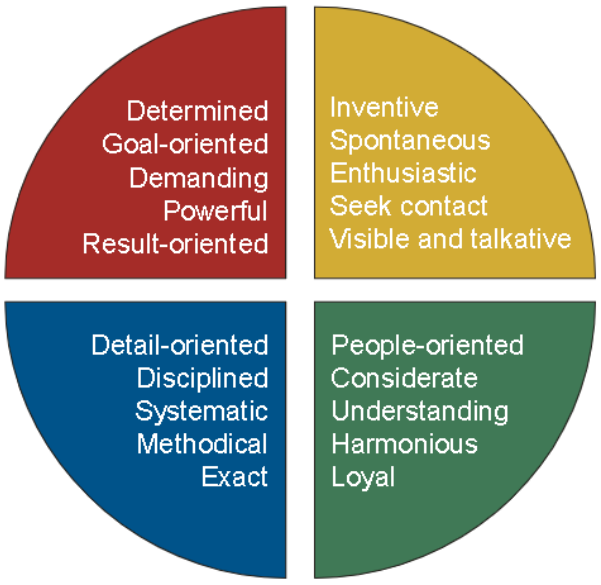


Figure 2: E-stimate personality profiles

Each member of the team has gotten the following scores in order:

- Daria – **Blue**, Green, Red, Yellow
- Natali – **Green**, Blue, Red, Yellow
- Bogdan – **Red**, Green, Blue, Yellow

According to the team profile, we have a very good balance having three different colors for each member. However, all team members scored very low in yellow so there might be a lack of communication and inventiveness so there will be some different measures in place such as more meetings where we brainstorm ideas together.

From past experiences, this team format has worked very well, as the combination of results, details, and people-oriented team members makes for a very robust group capable of managing its own time and getting good quality results all while maintaining a positive relationship between the group members.

3 Project Initiation

Natali Munk-Jakobsen

The first step of the project, forming a group, was completed quickly. Since we worked together in previous semesters and had positive experiences, we wanted to continue working together on our bachelor project as well. First of all, we prepared a group contract and stated our terms and rules of conduct. In this contract, we especially emphasized the importance of active communication, solving potential problems quickly before they grow, and complying with the determined time schedule (Appendix L).

At the beginning of the project, we talked about which technical fields and features we want to involve in our project, taking into account our interests and elective courses. We agreed not to include areas such as the Internet of Things or data analysis. Our common decision was to develop a mobile or web application.

At that stage, we did not know yet whether we would develop the project idea ourselves or work with a company. We started researching and brainstorming about possible project topics. We also attended the Company Presentation at VIA and the presentation made by the HumanLab theater group caught our attention. The Augmented Democracy, the concept they wanted to use in their theater plays, was very interesting and their demands were in line with our idea of developing a web/mobile application. Moreover, we could include machine learning by using predictive modeling to predict users' decisions.

Afterward, we contacted HumanLab and held an online meeting to talk about the details of the project. After reading the additional materials they provided us on the technical requirements and the concept of Augmented Democracy (Appendix K – TRIC concept, TRIC-Tech-Overview), we decided to work on the project with them.

Another issue we had to decide on was the type of application we were going to develop a web or a mobile application. After discussing the advantages and disadvantages of both application types, we came to the decision that web development is a more suitable option for this project, because the users will be using use the application only once while the theater play is being performed. Therefore, it would be easier for people to access the application via a web browser instead of downloading it.

The last part of the project initiation phase was to determine which project management framework and planning tools we would use in the project. This stage was also completed quickly because we were aware that the SCRUM method we used in the previous semesters fit well with our working style. We decided to use JIRA as the planning tool and GitHub for keeping track of the code. In addition, we agreed on taking small notes about important points while implementing the system, so we can use them when we start documenting the project.

After finishing the initiation phase fast and smoothly, we started to work on the project description.

4 Project Description

Natali Munk-Jakobsen

Before starting the actual project work, we needed to create a project description where we stated the main purpose and expected output of the project, time schedule, methods, and theories we plan to use in the project process.

The first step of the project description is the problem analysis phase, and the background description is based on the analysis and research we made in this stage. At the start of the process, we had a meeting with the HumanLab company and got general information about the TRIC project and its scope. Afterward, we researched relevant topics, such as democratic processes in the modern world, augmented democracy, technologies used in the democratic process, and the relationship between theater and democracy. This research helped us to gain a deeper insight into stakeholders' interests and needs and to define the problem domain more precisely.

The problem statement part of the project description represented the main problem which was formed based on the stakeholders' needs and demands. A group of sub-problems is included in the chapter to define all the problems that need to be dealt with to create a working solution. Similarly, the delimitation chapter was formed considering stakeholders' wishes and limitations to determine which problems will not be covered in the project.

We decided to use the SCRUM methodology in the project and the information about SCRUM meetings, sprint lengths, numbers of sprints, and the SCRUM roles are stated in the Methodology chapter. We divided the working period into 2 main parts: BPR1 and BPR2. We included BPR1 in the overall project period since we started the inception phase of the project in the 6th semester. However, different sprint lengths and the number of sprints were specified for the first and second parts.

In the Time Schedule chapter, we stated the total amount of expected work hours per student considering the number of ECTS for BPR1 and BPR2 courses. In addition, a time schedule diagram was added to the chapter to show the expected timeline for the four phases of the Unified Process.

In conclusion, the project description was a very important step of the project that gave us a clear understanding of the domain problem and the methodology we need to follow to achieve the project goals and guided us in the further stages of the project.

5 Project Execution

Daria-Maria Popa

Throughout the execution of the project, in order to ensure good communication between the team members and alignment with the SCRUM methodology several tools and methods were used.

The first one was regarding the use of a team capacity table before and during the Sprint planning session each sprint (Appendix J – Sprint Planning Meetings). These tables helped keep track of how many hours each team member was available to work each sprint, what the combined capacity was, and how this related to the intended weekly capacity, as calculated based on the ECTS score of the project.

This proved an important tool in planning the sprints because it allowed for a more precise and dynamic story point task assigning, based on personal capacity, and ensured not many tasks carried over from one sprint to the other.

Tracking the ‘to do’ tasks was done using Jira, an Atlassian product, with its integrated SCRUM board and tools such as burndown charts and cumulative flow diagrams. Choosing Jira came pretty naturally to the team since it was a tool we got accustomed to using from previous semesters and it offered all the functionality we needed to track our work.

A special task structure was created in order to ensure a good requirement traceability matrix and allow for a parent-child design between the SCRUM board’s representation of use cases.

The functional requirements were recorded under their own ‘functional requirement’ tasks and linked in a many-to-one way to the features (many functional requirements can be related to a feature), which represented the epics of the SCRUM board, and held multiple individual functionalities to be implemented, namely the ‘story’ tasks, as well as the ‘Qualification’ task, ensuring the proper testing of each newly introduced feature.

The use of Jira aided our progress throughout the project execution greatly, providing the visual representations needed to identify any bottlenecks or bad practices going on in our process early on, and quickly work on fixing them.

As we organized our sprints with the Unified Process phases in mind, multiple smaller goals were set during the project execution, matching the end of a phase. This helped split the overarching goal of the whole project into smaller, more easily manageable, and achievable goals, such as having a good foundational understanding and modeling of the problem domain and requirements for the Inception phase.

During the project execution, monthly releases have been used as a set goal for gradually achieving the intended outcome, but most importantly, as a tool for maintaining good communication with the collaborating company and ensuring early feedback.

Having testable prototypes, with varying degrees of fidelity, helped the quality of the feedback received, focusing the attention of the testing participant(s) earlier in the process on the functionality and flow of the system, rather than the visual design and appeal, helping us prioritize solving more critical priority technical and architectural problems.

An important part of this project's execution, realized together with the third and final prototype, was having the acceptance testing with a live audience and the company stakeholders. Through this, the project's quality was ensured as thoroughly tested and various helpful feedback was received before the delivery of the final product, setting the final step before concluding the project.

Overall, the project execution process went smoothly due to the thorough methods and tools used throughout, the results being deemed satisfactory by the team and the collaborating company, with various testing done to back this statement, especially the final acceptance test, whose result showed that the resulting system was able to be used in a real-life scenario and perform well while doing so.

6 Personal Reflections

Bogdan Mezei

Being part of "Team Cake" has been a very pleasant experience. We have worked together for a long time, ever since the second semester, so we know how everyone else works and how to cooperate with each other.

Even though the team members are the same as the previous semesters it has been the first time for all of us where we got to work on a semester project closely with a company and see our own product being used in a real-life scenario.

I believe all of us have very high-quality standards for the bachelor project and we have all tried our best to deliver a polished product. This is apparent in the way we have organized and prioritized all the tasks. I have enjoyed using Jira again as we have also used it last semester and I believe this time around we were even more efficient with it and managed to utilize the tool to its full potential.

For this project, we have decided to work using React which has been something entirely new for most of us and that meant extra time had to be put aside in order to learn how to use this framework. It has been a really enjoyable process for me to learn React as the first two first sprints have been populated with "research" tasks where we had the opportunity to learn the framework by reading, watching videos, and helping each other with small beginner projects.

Working with HumanLab has also been a really pleasant experience. Firstly, I think I speak for everyone when I say the project idea they have proposed has intrigued everyone. We also knew that we wanted to develop a web/mobile application and a voting system for a theater play dealing with the theme of augmented democracy has been really exciting for us.

In the early stages of development, we managed to schedule a lot of meetings together with HumanLab, receiving feedback on the UI design of the application and implementation details. The feedback has been mostly positive, and I really appreciated how they made it clear from the very start that we, as developers, have a lot of decision power over how the product should look and work. This has allowed our team to be creative and use a lot of the expertise acquired throughout our education while developing the system.

Later on, we also got to participate in rehearsals together with the actors which gave us the opportunity to make changes both in our system and in the play script itself in order to converge to a common flow of the play. These rehearsals have been really useful for our team as we had more chances to test the application in a more true-to-life scenario.

I have also had a very positive experience in regard to the supervision of the bachelor project. The supervisor has been really helpful in giving feedback on the progress of the project. We have scheduled regular meetings which ensured a good progression and quality.

In addition to improving my technical and organizational skills, this project has also given me the opportunity to work closely with my team members. I have enjoyed collaborating with them and learning from their expertise. We have built a strong working relationship and I believe that we have made each other better engineers as a result.

In conclusion, this project has been a fantastic learning experience for me and in my opinion, it is something we are all proud of. I am excited to continue learning and growing as a software engineer, and I am grateful for the opportunity to work on this project and to see it being used in the real world.

Natali Munk-Jakobsen

In our last semester, we were able to use all the knowledge and experience we had gained in previous semesters. For our bachelor project, we had the chance to work with a company. The project idea was interesting, but we were also excited about working on a real-life project and seeing our product being used by real users in the end. The HumanLab Theater Group was easy to communicate and work with. They stated their demands clearly and gave us quick feedback when we needed it. It made the analysis and design phases easier for us and we did not need to make big changes during the implementation of the project.

We were already familiar with most of the technologies we used in the project from our previous experiences. But React was a new technology for us. I had only a little experience from my internship, and it was completely new for the other team members. So, we had to spend some time researching and finding the best architecture for our system before starting the design and implementation of the front-end part. However, we managed to successfully use it in our project without having any problems and gained a lot of new knowledge.

We chose the SCRUM framework again as we used it several times and had good experiences. I think we applied the SCRUM methodology more professionally and effectively this time. We had better structured and organized SCRUM meetings compared to previous semesters. This resulted in an increase in our work quality and motivation. We used JIRA as a planning tool and personally I think it was a great decision. It really helped us with organizing our workload and managing our time efficiently.

As a team working together for a long time, we knew each other’s working style, interests, and expectations, so we did not have any problems with sharing tasks and responsibilities. Each group member was aware of the importance of performing group tasks on time and working cooperatively. Unlike the times we spent with online meetings, we had the opportunity to spend more time working together in person this semester and it helped us to improve our team cohesion. It was a very pleasant experience to work with Team Cake as always. We learned a lot from each other and made an important contribution to each other's technical and professional development through the years we worked together.

Communication with the supervisor was easy and beneficial. Having weekly meetings and receiving feedback frequently helped us to see if we were on the right track and what we needed to improve. Thanks to the supervision we received helpful information, fixed any issues early, did not waste time on the wrong path, and ensured good work quality.

In conclusion, this project was a great experience for me. The project idea was exciting, and I really had fun working on it. I am glad that we managed to fulfill all requirements and had a satisfying product being used in real life in the end. I am sure all the knowledge and experience I gained from this bachelor project will be very valuable and beneficial in my future career as a software engineer.

Daria-Maria Popa

This project marks our final one as ‘Team Cake’, a team that has been created 3 years ago and from which I have not only gained great knowledge but also great teammates and friends, so it means a lot to have this one last project as a culmination of our work put towards becoming software engineers throughout the years of our studies.

We were fortunate to work together with a company this semester on a really exacting and equally challenging project idea. This idea presented by HumanLab, of using a web application to let the audience vote on what happens during a live theater play, allowed us to collaborate with really creative people and see our final product be used by a real audience, something that I am really grateful for.

We put a lot of thought behind the organization and process for this project, something that I think could really be seen in the thorough schedules made ahead of time for the meeting and rehearsals, as well as our use of Jira and SCRUM.

After working together for so long and also having some experience working in a real company during and after our internships, we were really motivated and more prepared to make good use of the tools and knowledge we have gained, and the overall progress of the project went really smoothly because of this.

The work tasks were easily split between the team members based on our previous knowledge and a general understanding of how we work and what are our interests, and the use of Jira supported us a lot in ensuring that we are staying on schedule and signaling when we needed to prioritize or focus on certain tasks together.

A big part of this project was the new topics introduced such as React and Machine Learning, both things we have not worked on before and with which we had no previous experience, except some React knowledge. Because of this we planned for and spent a good part of the first few sprints researching the topics as well as we could, trying and testing smaller demo projects, and discussing more complex issues with some of our teachers. This helped build our confidence when tackling this project and, in the end, we faced no major issues and managed to implement a system that I am really proud of.

HumanLab as a collaborating company was really supportive and helpful in the feedback they gave and maintained an active communication channel with us, allowing for the quick exchange of ideas and clarification of questions.

The supervision of this project was also extremely important, and I am really happy about the communication and weekly meeting schedule we set together with our supervisor since it allowed us to receive really important feedback early on and have a professional second opinion of some of the project topics. Overall, the meeting with both the company and our supervisor helped polish the quality of our product and ensure the level of fidelity to the stakeholders' requirements that we managed to achieve.

All in all, this project was a really great experience, I learned a lot, not only technically but also relating to the process of working with a company and having your system be used in the real world. I am really proud and happy with what we managed to achieve together for this project, I think we grew a lot as software engineers during this last semester, and I am definitely looking forward to what we will continue to achieve in the future.

7 Supervision

Daria-Maria Popa

Supervision for this project was conducted together with Henrik Kronborg Pedersen and consisted of weekly status and feedback meetings throughout the length of the project.

The supervision meetings have been structured to focus on the current status, the overall progress of the last sprint, and the planning for the current sprint. The meetings were arranged to match as closely as possible the sprint review and new sprint planning day so that the supervision feedback could be taken into consideration and included in the sprint goals.

The help of the supervisor has been used by the whole team, with the meetings scheduled for all the team members to attend, and questions posed by all members based on the specific areas of work we were handling at the moment.

During this project, the supervision with Henrik had as the main scope ensuring the alignment of the company's requirements to the ones of the team and the quality standard set by the university, the supervision acting as a guide through the process of initiation until the delivery of the product.

The weekly meetings proved helpful in keeping track of the quality of the work done thus far, and the answers provided by the supervisors were up to the expected standard of the team, offering good insight into possible improvements, both structural and ones more related to the understanding of the problem domain at hand.

In order to answer some of the machine learning questions posed by this project, supervision was also asked from Frederik Thorning Bjørn. This was done during the design phase of the machine learning system and consisted of a knowledge-sharing session with the scope of identifying one or more adequate models that could be used in our project. Although it was a singular supervision meeting it contributed invaluable feedback toward the current design of the system.

All in all, the supervision for this project has been deemed really useful and proactive by all the team members, with the 1-hour weekly meetings proving to be a great practice for maintaining good communication with the supervisors and ensuring the alignment of the project with the goals set by all stakeholders and other parties involved.

8 Conclusions

Bogdan Mezei

This has been the first time we have developed a semester project for a company, so we quickly realized that having a robust plan beforehand is the key to success.

Communication between all team members, the supervisor, and the company stakeholders has never been more important. As we had to stick to monthly releases for the company, we had to put extra attention into managing our tasks and prioritizing what was most important at any given moment.

Additionally, we have attempted to communicate closely with each other every single day, even if it was just through the daily SCRUM meetings. This has benefited us greatly by knowing at all times what the other team members are currently working on and what is the status. This allowed us to closely coordinate with each other and finish the highest-priority tasks on time.

For this reason, we believe that using Unified Process and SCRUM as our process methodologies has been extremely rewarding. They offered the high degree of organization we needed while also being flexible enough to prioritize tasks based on company feedback.

One thing we tried applying that has proven to be very useful in the long run is related to documentation. We agreed as a team at the very start of the project work that while implementing the system we should add small key elements in the form of bullet points in the project report. Doing this allowed us to save a lot of time while documenting everything later on in the project development process as we could keep track of any implementation details, challenges, and important references.

To conclude, the group work recommendations from our team are as follows:

- ❖ Carefully plan what needs to be done ahead
- ❖ Prioritize tasks each sprint and remain flexible
- ❖ Always mark any questions or uncertainties and ask your teammates/supervisor
- ❖ Note down main ideas for the documentation from the very start
- ❖ Try communicating with the group members every day, even if it is just a short text detailing what you are working on at the moment

These group work recommendations have proven to be very effective to our team and allowed us to meet the expected results and every deadline. In the end, we are proud of the end result and the good cooperation we had with HumanLab and our supervisor.

9 Appendices

Appendix A Project Description

Appendix J Scrum Documentation

Appendix K Company Meetings

Appendix L Group Contract