FACULTY
OF ELECTRICAL
ENGINEERING
CTU IN PRAGUE

Bachelor’s thesis

Web platform for donation

Aiya Rakhimova

Department of Software Engineering and Technologies
Supervisor: Ing. Kyrylo Bulat

May 23, 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Rakhimova** Jméno: **Aiya** Osobní číslo: **499327**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro darování

Název bakalářské práce anglicky:

Web platform for donation

Pokyny pro vypracování:

The project aims to create a platform for donations to sick people who need material assistance. As part of this work:
- analyze existing applications and platforms for donations available on the market,
- design a user interface for the application to motivate future users to use it,
- analyze technologies for the implementation of front-end, back-end, and database parts of the application,
- design application architecture with a focus on the back-end part of it,
- implement a proof of concept version of the platform,
- design test cases,
- perform appropriate testing of a proof of concept and automate testing of the core scenarios.

Seznam doporučené literatury:

Richards, Mark. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020.
Walls, Craig. Spring Boot In Action. Shelter Island, NY: Manning Publications, 2016.
Tidwell, Jenifer, Charlie Brewer, and Aynne Valencia. Designing Interfaces: Patterns for Effective Interaction Design. Beijing: O'Reilly, 2020.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Kyrylo Bulat katedra počítačů FEL

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

Datum zadání bakalářské práce: **14.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Kyrylo Bulat
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

.
Datum převzetí zadání

Podpis studentky

Acknowledgments

I would like to thank my supervisor Ing. Kyrylo Bulat for his help, useful advice and guidance while working on the project. In addition, I would also like to express my gratitude to my family for their support throughout the entire period of study at CTU in Prague.

Declaration

I declare, that I have done assigned bachelor thesis alone led by supervisor and cited all sources of information in accordance with the Guideline for adhering to ethical principles.

In Prague on May 23, 2023 .....………………

Abstract

This bachelor's thesis deals with designing and implementing a web application for donating to sick people. The created application is divided into three parts: Front-end, Back-end, and Database. The Front-end part is implemented using React technology. The Back-end is written in Java using the Spring Boot framework and the REST API. The Database uses PostgreSQL technology.

Keywords: donation, donor, indigent, fundraising, React, Java, REST API, PostgreSQL, Spring Boot, web application.

Abstract

Tato semestrální práce se zabývá navrhem a implementaci webové aplikaci pro poskytování darů nemocným lidem. Vytvořená aplikace je rozdělena do třech částí: Frontend, Backend, a Databáze. Frontendová část je implementována pomocí technologie React. Backend je napsán v jazyce Java s použitím frameworku Spring Boot a REST API. Databáze využíva PostgreSQL technologii.

Klíčová slova: dar, donor, nemocné lidi, React, Java, REST API, PostgreSQL, Spring Boot, webová aplikace.

Contents

Introduction **1**

1 State-of-the-art **3**

1.1 Existing solutions 3

1.1.1 Dobro 4

1.1.2 Planeta.ru 5

1.1.3 Crowdfunder 5

1.1.4 JustGiving 6

1.1.5 Darujme.cz 7

1.2 Summary 8

2 Analysis **9**

2.1 Requirements 9

2.1.1 Functional requirements 9

2.1.2 Non-functional requirements 11

3 Design **13**

3.1 Use cases 13

3.2 Architecture of the application 17

3.3 Front-end 18

3.3.1 Pages layout 18

3.3.2 Technologies 21

3.4 Back-end 23

3.4.1 Application tier 23

3.4.2 Technologies 23

3.4.3 Data tier 24

4 Implementation **27**

4.1 Back-end 27

4.1.1 Data layer 27

4.1.2 Business layer 30

4.1.3 REST API layer 31

4.2 Security 34
4.3 Front-end 37
4.3.1 Components of frontend 37
5 Testing 41
5.1 Unit testing 41
5.2 Scenario testing 42
5.2.1 Registration 42
5.2.2 Username and password changing 43
5.2.3 Donate 44
5.2.4 Create report with documents (with preview) 44
5.2.5 Delete document from report 45
5.3 Usability testing 45
5.3.1 Tester 1 46
5.3.2 Tester 2 46
5.3.3 Tester 3 47
5.4 Summary of testing 47
Conclusion 49
List of literature 51
Acronyms 55
Code fragments 57
Final GUI 61
Executing application 65

List of Figures

1.1 Dobro logo 4
1.2 Planeta.ru logo 5
1.3 Crowdfunder logo 6
1.4 JustGiving logo 6
1.5 Darujme.cz logo 7

3.1 Assigned use-cases to two types of user’s role 14
3.2 Assigned use-cases to donor and indigent 15
3.3 Three-tier architecture 17
3.4 The registration page of the needy 18
3.5 Form to fill out during registration №1 19
3.6 Form to fill out during registration №2 19
3.7 Patient profile page 20
3.8 The page for adding a new treatment report 20
3.9 Final structure of the application 25

4.1 The data layer structure 27
4.2 User entities group 28
4.3 The UML diagram 29
4.4 Example of using Lombok annotations 29
4.5 Code snippet of interface Document Repository 30
4.6 Donate function code fragment of UserService class 31
4.7 DocumentController endpoints using Swagger 32
4.8 Visual interpretation of UserController using Swagger 33
4.9 Backend structure 34
4.10 Sequence diagram of authorization and registration processe 35
4.11 Implementation of the registration process on the server side 36
4.12 Code fragment of auth.service.js 36
4.13 Code snippet of AuthorizationPage.js 37
4.14 Part of the code of DocumentUploader component 38
4.15 Part of the code of ReportCreatePage component 40

Introduction

Access to affordable health care is an essential aspect of a healthy and prosperous society. However, nowadays medicine is an expensive “pleasure” and not everyone can afford it. This creates a serious problem for people in need of medical treatment because it entails negative consequences for their health and well-being.

To solve this problem, various solutions have been created, such as government subsidies, insurance systems and charitable organizations that provide financial assistance to people in need. While these solutions are beneficial, they are not always effective in providing immediate financial assistance to those in dire need.

Therefore, it was decided to develop a web application that will provide people with a quick and easy way to donate to people in need of medical treatment. This platform will serve as a link between donors and indigents. Donors can contribute any amount they wish, and the donated money can be used to pay for treatment, hospital bills and related expenses.

The goal of this bachelor's thesis is to develop a web application that can act as a platform for donations to sick people. The thesis will begin with an analysis of existing market solutions. This analysis will help to identify the shortcomings of competitors and opportunities for developing a better platform.

After analyzing the problem, a prototype of the web application will be developed. The future platform will include several features such as authentication and registration, profile management, document upload and donation system. The web application will be tested for the correctness of processing user data and performing various operations.

Chapter **1**

State-of-the-art

This chapter analyzes several existing solutions on the market, their pros and cons.

1.1 Existing solutions

The market is saturated with various donation systems, but most of them are not aimed at healthcare purposes. Instead, they focus on raising funds for non-medical purposes, such as environmental protection, animal shelters, or restoration of historic buildings.

Platforms that allow sick people to post their financial aid ads often do not require any confirmation of a person's health status. The lack of verification often leads to potentially fraudulent activities and misuse of funds, thereby causing distrust of potential donors.

Moreover, web platforms often do not provide any information about the use of funds and the results of treatment, leaving donors in the dark about how and where their contributions were used and what impact they had.

Several donation platforms have been studied to get information about their functionality. It is important to note that the solutions considered may not represent the entire range of available products. The following systems were analyzed:

- Dobro
- Planeta.ru
- Crowdfunder
- JustGiving
- Darujme. cz

1.1.1 Dobro

The Dobro[1] project was developed in 2013 by a team of developers Mail.ru.The creators of the project were inspired by the problems of people who live in the Commonwealth of Independent States.

The main goal of the project is to help people who need it the most. The platform also provides an opportunity to donate to orphans, animal shelters and projects related to nature protection.

Figure 1.1: Dobro logo

A summary of the Dobro project’s pros and cons:

Pros:

- clear user interface
- list of success stories
- open list of people in need
- reliable information about patients

Cons:

- lack of medical evidence of a person's illness
- donations are made only through foundations
- partial treatment report

[1] https://dobro.mail.ru/projects/?recipient=kids

1.1.2 Planeta.ru

Planeta[2] is a social and service ecosystem that helps to monetize ideas, projects, creativity and author's content in various ways. In addition, the project also cooperates with charitable foundations.

Figure 1.2: Planeta.ru logo

Pros:

- user-friendly interface
- reliable information
- bonuses from the company for donors
- the opportunity to receive money prematurely if the amount exceeds half of the required amount

Cons:

- small number of fees for sick people
- lack of medical evidence of a person's illness
- donations are made only through foundations
- inactive community

1.1.3 Crowdfunder

Crowdfunder.co.uk[3] is a popular crowdfunding platform based in the United Kingdom. It provides individuals and organizations with the opportunity to create fundraising campaigns and establish contacts with potential supporters. This allows people to share their project ideas, set funding goals, and offer rewards or incentives to sponsors who contribute to their campaigns.

[2] https://planeta.ru/
[3] https://www.crowdfunder.co.uk/

Figure 1.3: Crowdfunder logo

Pros:

- the different types of crowdfunding
- stylish and easy-to-use interface
- list of popular projects
- list of people who have made a donation to a project
- the possibility of making a donation without registration

Cons:

- little information about the organization itself
- there is no history of expenses of donated funds

1.1.4 JustGiving

JustGiving[4] is the trusted platform for online giving. They help people raise money for the charities and people they care about the most.

Back in 2000, JustGiving.com began with one simple goal - to enable charities to receive donations online from anywhere in the world.

Figure 1.4: JustGiving logo

Pros:

- easy navigation
- the different types of crowdfunding
- list of people who have made a donation to a project
- the possibility of donating without registration

[4] https://www.justgiving.com/

Cons:

- in some places, a bad design
- there is no history of expenses of donated funds

1.1.5 Darujme.cz

Darujme.cz[5] is the largest online donation platform in Czech Republic.

They provide organizations, institutions, but also cities, towns, and countries with a secure tool to be seen and heard, to simply ask for a donation, and to maintain a relationship with their donors.

DARUJME.CZ

Figure 1.5: Darujme.cz logo

Pros and cons of the platform:

Pros:

- user-friendly interface
- list of current projects
- update of news about fees
- the ability to write comments on fees

Cons:

- the collection is made for foundations, not specific people
- the Society does not take an active part in donating funds to sick people
- insufficient information and confirmations

[5] https://www.darujme.cz/

1.2 Summary

Based on the analysis of existing competitors, it can be summed up that all these platforms do not support the possibility of making donations of funds directly to people in need, without intermediaries. Also, not all the above platforms have the necessary functionality for creating reports and high user activity.

For this reason, it was decided to create a platform in which the necessary, useful and convenient functionality will be present.

Chapter **2**

Analysis

This chapter provides an analysis of functional and non-functional requirements for the development of a web application.

2.1 Requirements

The requirements will describe the necessary functionality of the web platform. Usually, when developing any software, two types of requirements are distinguished:

- functional
- non-functional

2.1.1 Functional requirements

Functional requirements specify the behaviors the product will exhibit under specific conditions. They describe what the developers must implement to enable users to accomplish their tasks (user requirements) [1].

The web platform should contain functionality for the following groups:

- users
- patient’s form
- treatment report
- documents

2.1.1.1 Users

The platform serves two user groups: patients who require assistance and donors who offer support. All users can perform the following primary actions:

- register a new account
- modify their user information
- sign in to the system
- log out
- delete account

2.1.1.2 Patient’s form

When sick individuals register on the platform, they are required to fill in a form providing details about their health condition, current treatment and illness progression. The web application allows users to edit this information at any time. Donors, on the other hand, can access forms of patients on a separate page, and can add certain forms to their "Favorites" section for easier tracking of their treatment progress.

As a result, the platform must have capabilities to:

- create new forms
- edit existing forms
- add and remove forms to and from the "Favorites" section

2.1.1.3 Treatment report

To ensure the transparency and accountability, indigents are required to publish treatment reports and update them as needed to demonstrate the appropriate use of donated funds. The system offers the following features for this purpose:

- create new treatment report
- edit recent report
- delete recent report
- view treatment history

2.1.1.4 Documents

Medical certificates are a confirmation of the presence of a disease in a person, therefore, in addition to a textual description, the patient must attach all the necessary documents. To do this, the platform has the following functions:

- uploading scan paper or electronic document/s to system
- deleting uploaded documents

In addition, the platform also provides the ability to download uploaded documents. This can be useful for donors who want, for example, to get better acquainted with medical certificates.

2.1.2 Non-functional requirements

Non-functional requirements (NFRs) are requirements that do not relate to the functionality of the system but to its operational characteristics. NFRs deal with issues such as reliability, response time, throughput, maintainability, usability, and portability [2].

Based on the definition given above, the following basic requirements can be deduced:

- availability
- security
- scalability
- usability
- functional completeness

2.1.2.1 Availability

In addition to ensuring the constant availability of the platform, it is also important to allocate regular maintenance hours. This involves allocating approximately 2-3 hours every 2 months to update the system and eliminate any problems that may arise.

2.1.2.2 Security

To ensure the security of user data, it is extremely important to use a reliable password encryption method. The method involves converting user passwords into an encoded format that is difficult for third parties to decrypt. Thanks to the use of encryption, confidential information, such as a password, is protected from potential threats and unauthorized access.

2.1.2.3 Scalability

With the passage of time, as the user base grows, it becomes essential for the system to have the capability to scale horizontally. This means that the system should be designed and configured in a way that allows addition of more servers or resources to accommodate the increasing number of users and their demands.

2.1.2.4 Usability

The platform is designed to directly serve people who need financial assistance for their treatment, eliminating the need for intermediaries. It offers a user-friendly and intuitive interface with a clean and optimized design, ensuring ease of use for all users.

2.1.2.5 Functional completeness

In addition to the point above, the system contains the most necessary functionality to achieve the goal of the needy and the donor. The platform includes essential features such as enabling patients to create fundraising forms, allowing the creation of treatment reports, and facilitating donations from donors.

Chapter **3**

Design

This chapter describes the possible interactions of users with the system and with each other. It also describes the selected application architecture, used technologies and presents a prototype of the web platform.

3.1 Use cases

A use case can be defined as "a specific interaction between a system and its actors, representing a coherent sequence of transactions that fulfills a particular goal of the actors" [3]. It describes the functional requirements of the system from the perspective of its users or external entities.

The web application primarily focuses on one key role. The user is the main individual who utilizes the platform for their needs. While the project may potentially involve other roles, such as an administrator, for the purpose of the basic prototype, the user role is considered sufficient.

Further, users in the system can be divided into two types: authorized and unauthorized. These two types have different sets of functionality available to them. (Figure 3.1)

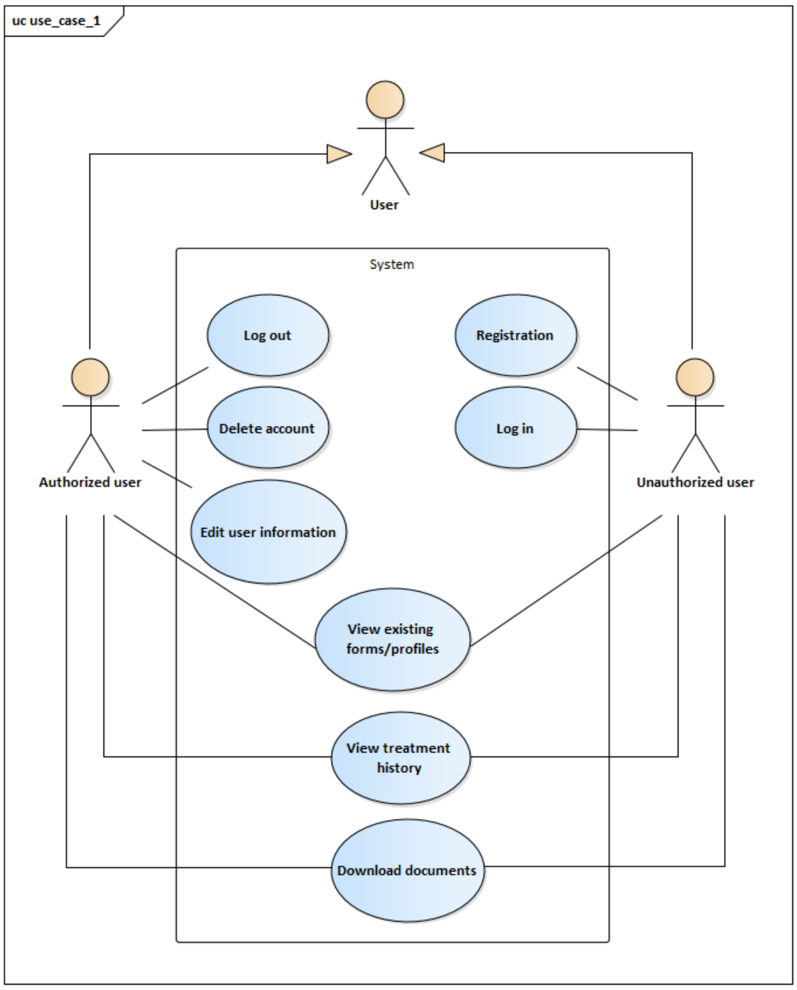


Figure 3.1: Assigned use-cases to two types of user's role

3.1.1 Registration

A basic operation for any user who is using the platform for the first time.

3.1.2 Log in

Functions for previously registered users.

3.1.3 Edit user information

An important use case is to edit the user data.

3.1.4 View existing forms/profiles

The user can view all existing forms/profiles of indigents

3.1.5 View treatment history

In addition to viewing the profile, the user can view the entire history of the patient's treatment.

3.1.6 Log out

Functions for authorized users.

3.1.7 Delete account

If necessary, the account can be permanently deleted.

Authorized users are also divided into two main groups: donors and indigents. Each of them has its own set of necessary functionalities. (Figure 3.2)

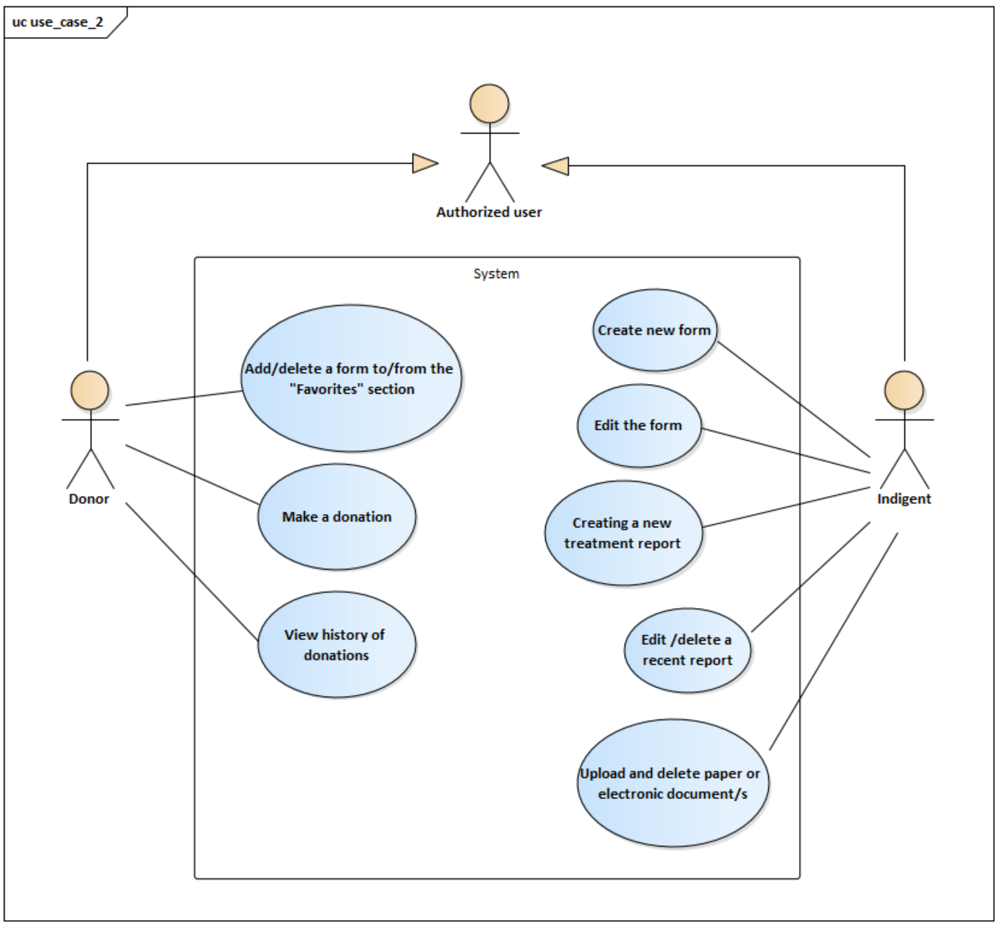

uc use_case_2
Authorized user
System
Donor
Indigent
Add/delete a form to/from the "Favorites" section
Make a donation
View history of donations
Create new form
Edit the form
Creating a new treatment report
Edit /delete a recent report
Upload and delete paper or electronic document/s

Figure 3.2: Assigned use-cases to donor and indigent

3.1.8 Add/delete a form to/from the “Favorites” section

For donors, the system supports the function of adding and deleting forms in the "Favorites" category, so that it is easier and more convenient to monitor the treatment of certain people.

3.1.9 Make a donation

The most important function for donors.

3.1.10 View history of donations

The system supports the function of viewing the donations history.

3.1.11 Creating new form

After registration, every sick person is required to fill out a certain form.

3.1.12 Editing new form

If necessary, the form can be adjusted.

3.1.13 Creating a new treatment report

The system allows patients to create treatment reports.

3.1.14 Edit/delete a recent treatment report

If necessary, recent reports can be changed or deleted.

3.1.15 Upload/delete paper or electronic document/s

A patient can upload document/s to the application or delete them.

3.2 Architecture of the application

For this project, a three-tier architecture is the most suitable, which consists of the:

- presentation tier or user interface (UI),
- application tier, where data is processed,
- data tier, where the data associated with the application is stored and managed [4].

The three-tier architecture offers several advantages in software development. Here are a few of its key benefits:

- maintainability and reliability: with clear separation of concerns, the three-tier architecture promotes modular design and code reusability. Changes or updates made in one layer do not affect the others, simplifying maintenance and reducing the likelihood of introducing bugs or errors,
- scalability: any tier can be scaled independently of the others as needed,
- security: because the presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall, preventing SQL injections and other malicious exploits [4].

Figure 3.3 is a visual demonstration. More information about each layer is described in the following sections.

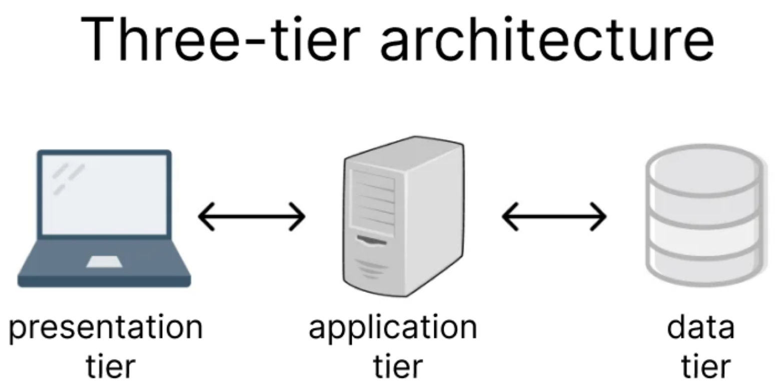


Figure 3.3: Three-tier architecture

3.3 Front-end

The presentation tier is the user interface and communication layer of the application, where the user interacts with the application. Its main purpose is to display information to and collect information from the user [4].

3.3.1 Pages layout

Before developing any user interface, there is a need to create a prototype. The following most important pages can be distinguished from the entire system:

- the registration page of the needy - Figures 3.4 - 3.6
- the profile page - Figure 3.7
- the page for adding a treatment report - Figure 3.8

Below are the prototypes of these pages.

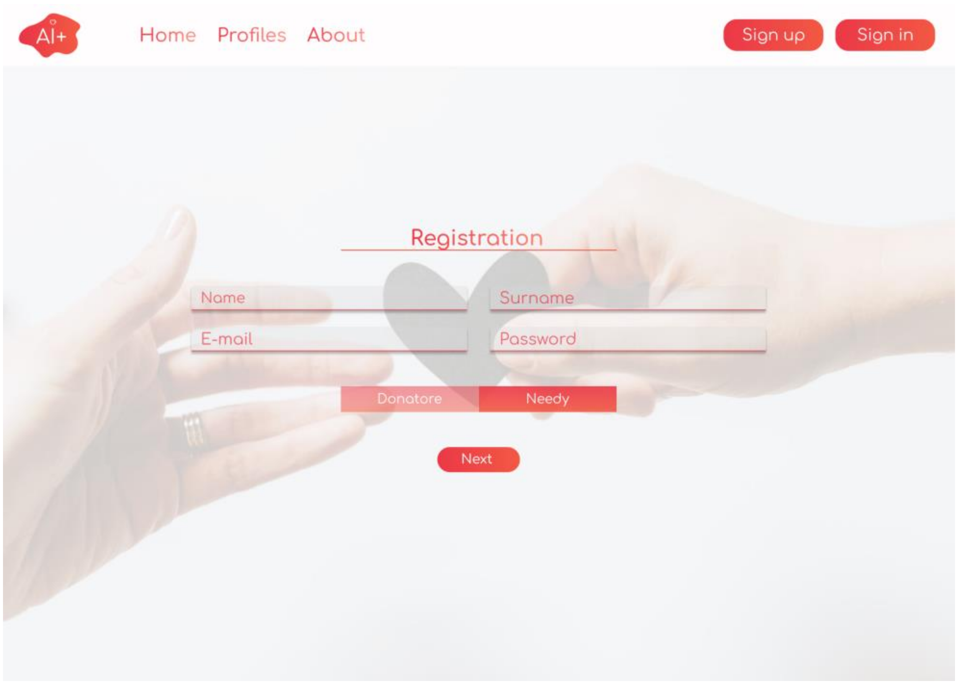


Figure 3.4: The registration page of the needy

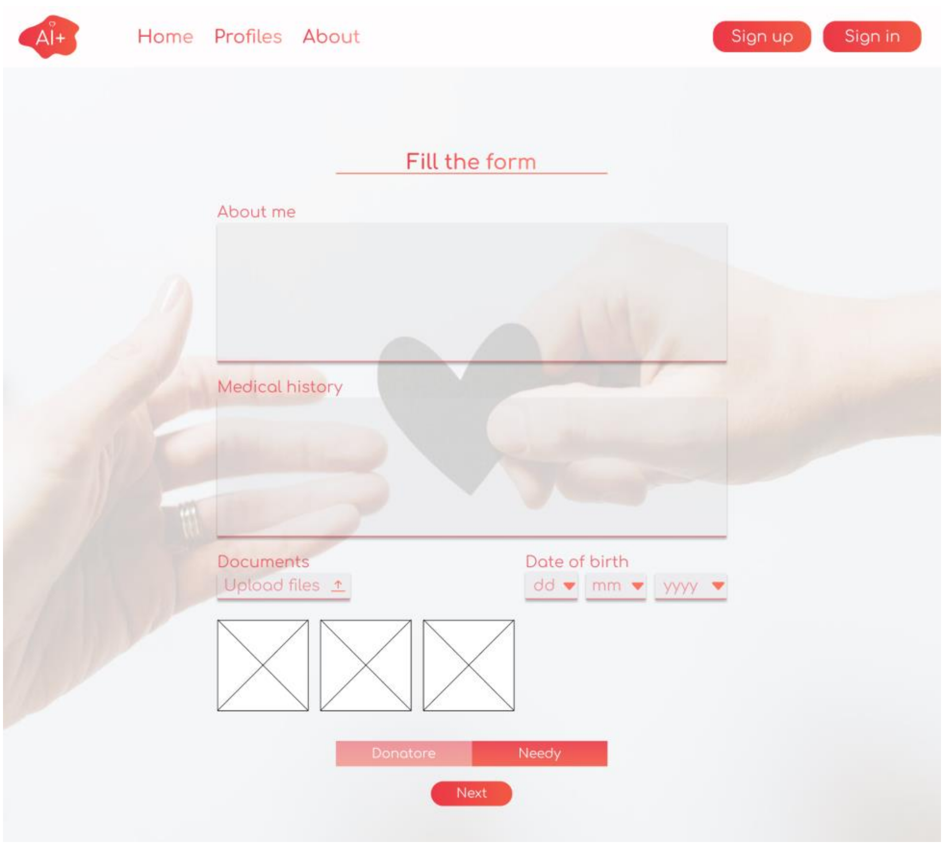


Figure 3.5: Form to fill out during registration №1

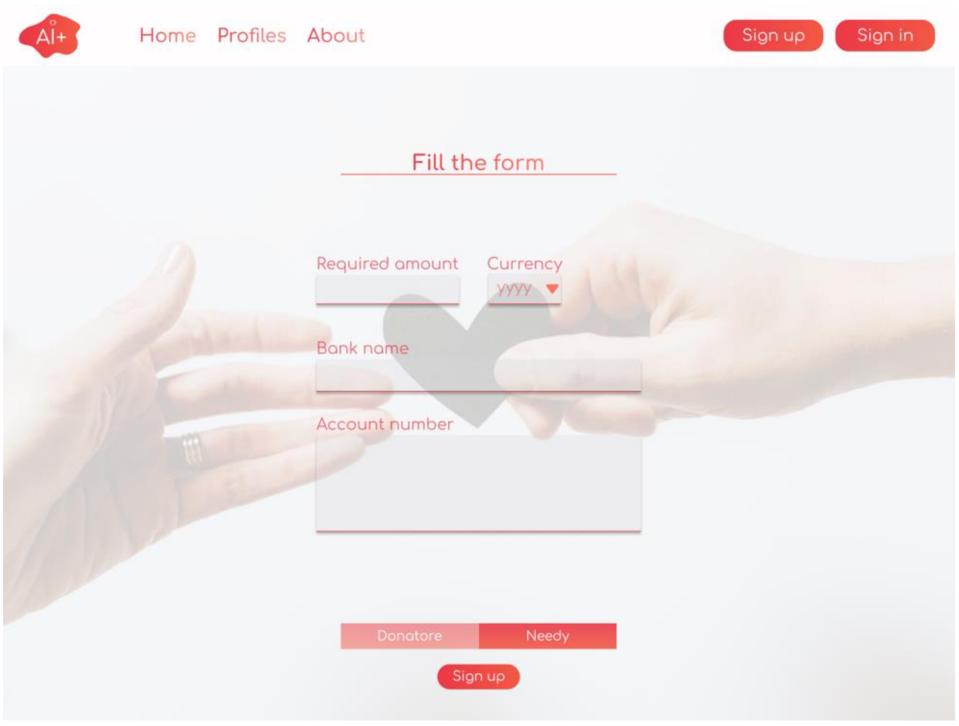
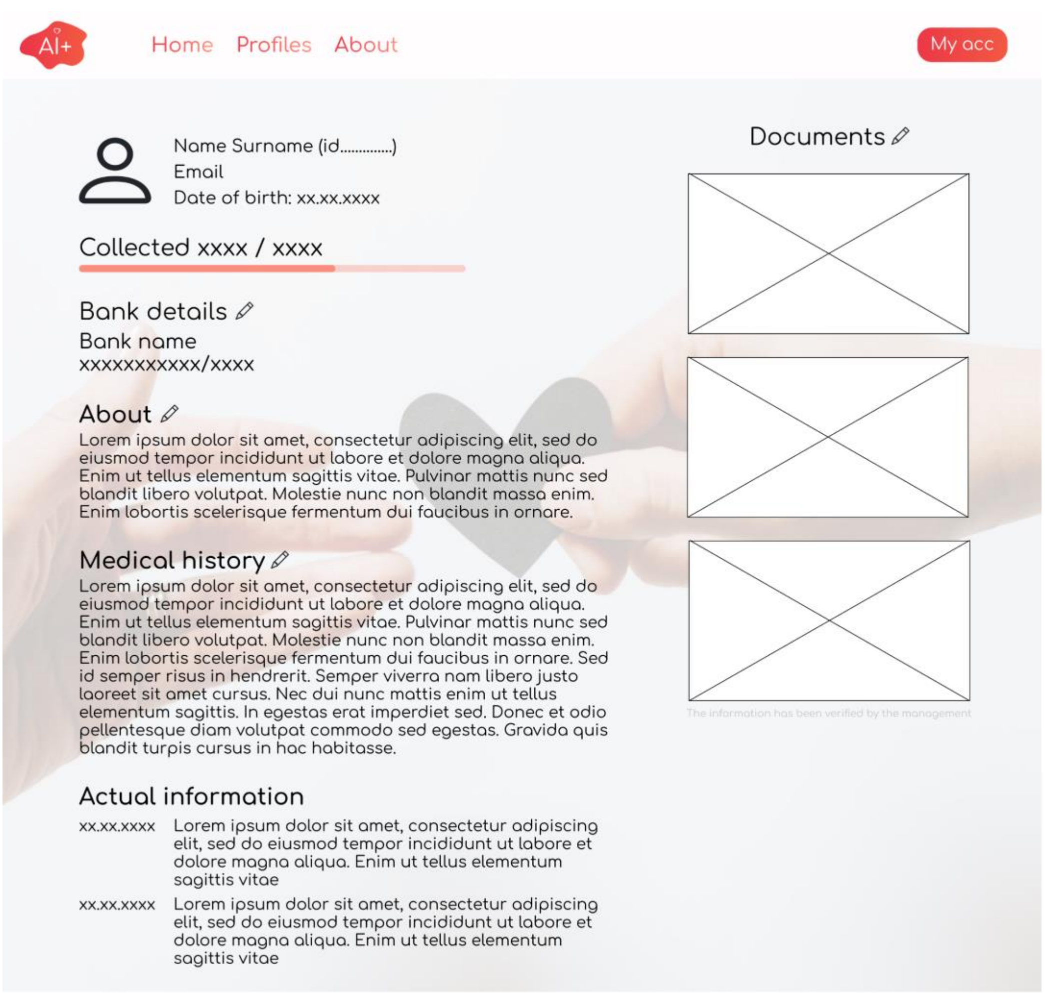


Figure 3.6: Form to fill out during registration №2



Name Surname (id.............)
Email
Date of birth: xx.xx.xxxx

Collected xxxx / xxxx

Bank details
Bank name
xxxxxxxxxxx/xxxx

About
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Enim ut tellus elementum sagittis vitae. Pulvinar mattis nunc sed blandit libero volutpat. Molestie nunc non blandit massa enim. Enim lobortis scelerisque fermentum dui faucibus in ornare.

Medical history
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Enim ut tellus elementum sagittis vitae. Pulvinar mattis nunc sed blandit libero volutpat. Molestie nunc non blandit massa enim. Enim lobortis scelerisque fermentum dui faucibus in ornare. Sed id semper risus in hendrerit. Semper viverra nam libero justo laoreet sit amet cursus. Nec dui nunc mattis enim ut tellus elementum sagittis. In egestas erat imperdiet sed. Donec et odio pellentesque diam volutpat commodo sed egestas. Gravida quis blandit turpis cursus in hac habitasse.

Actual information

xx.xx.xxxx	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Enim ut tellus elementum sagittis vitae
xx.xx.xxxx	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Enim ut tellus elementum sagittis vitae

Documents

The information has been verified by the management

Figure 3.7: Patient profile page

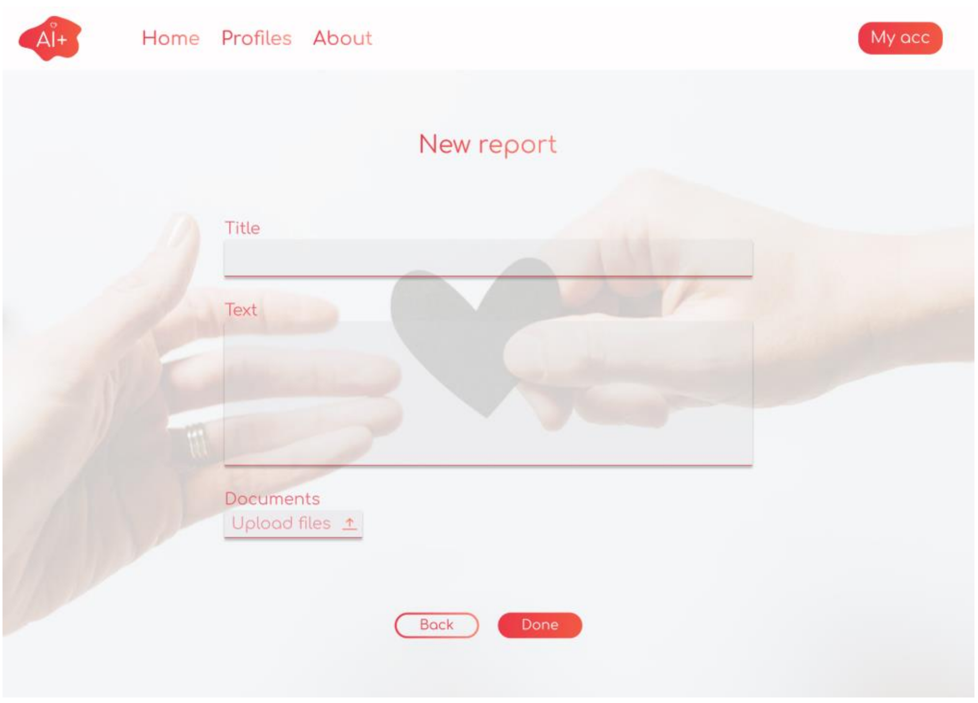


Figure 3.8: The page for adding a new treatment report

3.3.2 Technologies

There is a wide variety of frameworks and libraries available for front-end development. Among them, three of the most popular and widely used ones are:

- Vue.js
- Angular
- React.js.

3.3.2.1 Vue.js

Vue is a progressive JavaScript framework for building user interfaces. It is based on standard HTML, CSS and JavaScript and provides a declarative and component-based programming model that helps to effectively develop both simple and complex user interfaces [5].

3.3.2.2 React

React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications [6].

3.3.2.3 Angular

Angular is a TypeScript-based, free and open-source web application framework led by the Angular Team at Google. It is designed to create single-page client applications using HTML and TypeScript. Angular follows the component-based architecture and promotes a structured approach to development [7].

3.3.2.4 Chosen technology

React offers several advantages that contribute to its popularity. Some of the key advantages of React technology are:

- component-based architecture: this architecture allows developers to break complex user interfaces into smaller, reusable components. This promotes code reuse, modularity, and easier maintenance.
- virtual DOM (Document Object Model): React utilizes a virtual DOM, which is a lightweight representation of the actual DOM. This approach improves performance by reducing direct manipulation. React efficiently updates only the necessary parts of the UI, resulting in faster rendering and improved overall performance.

- JSX (JavaScript XML): React technology uses a syntax extension that allows developers to write HTML-like code within JavaScript. It simplifies the process of building and maintaining the UI structure.

In addition to the above, React has a very extensive and active developer community. This ensures continuous improvement, frequent updates and a rich ecosystem of tools.

3.3.2.5 Axios

Axios was used to communicate with the server side of the application. Axios is a library that is used to make API requests, return data from the API, and then perform actions with this data in the React application.

3.3.2.6 SCSS and Material-Ui

With the help of Sass and the Material UI library, a simple and concise design of the platform was created.

Sass is a stylesheet language that’s compiled to CSS. It allows to use variables, nested rules, functions, and more, all with a fully CSS-compatible syntax. Sass helps keep large style sheets well-organized and makes it easy to share design within and across projects [8].

Material UI is an open-source React component library that implements Google's Material Design. It includes an extensive collection of ready-made components that are ready to be used in production right out of the box.
Material UI has a set of customization options that make it easy to implement own custom design system on top of the basic components [9].

3.4 Back-end

Back-end development, also known as server-side development, covers the activities related to the server and database components of a web application. It involves handling the logic and data processing that occurs behind the scenes when users interact with a website or application. As Martin Fowler described, the development of the backend focuses on "the logic that determines how data is stored and accessed" [10].

The back-end part of the web platform can be divided into two tiers: application and data.

3.4.1 Application tier

In a three-tier architecture, the application tier, also known as the middle or the business logic tier, plays a crucial role in processing and managing the business logic and rules of an application. It acts as an intermediary layer between the presentation layer (UI) and the data storage layer. This tier is responsible for executing and coordinating the application's core functionalities, handling user requests, and providing the necessary processing and manipulation of data [4].

3.4.2 Technologies

Technologies used for this part of the application are described below.

3.4.2.1 Java

Java is a very popular and widely used programming language created by Sun Microsystems (currently owned by Oracle). It is known for its strict typing and object-oriented nature. According to James Gosling, the creator of Java, it was designed to be "simple, object-oriented and familiar" [11]. Java has numerous advantages such as: simplicity, security, performance, reliability and much more.

One of the notable features of Java is its versatility and platform independence. As stated in official document on Java technology, it follows the "Write Once, Run Anywhere" (WORA) principle, which allows applications to run on any system equipped with the Java Runtime Environment (JRE). The Java source code is compiled into bytecode, which can be interpreted and executed by the Java Virtual Machine (JVM) regardless of the underlying operating system [12].

3.4.2.2 Spring

Spring Boot is a widely used Java-based framework that provides a simplified approach to creating reliable and scalable web applications. It aims to simplify the development process by reducing boilerplate code and providing a convention-based approach rather than configuration.

The Spring framework includes various modules that solve different tasks of enterprise application development, such as data access, web development, security, and much more. Thanks to its modular architecture, Spring allows to expand the capabilities of the application in accordance with the requirements of a specific application [13].

3.4.2.3 Lombok

Lombok is a Java library that aims to reduce boilerplate code and enhance developer productivity by providing automated code generation for common Java tasks. It offers a set of annotations that can be added to Java classes, allowing the library to generate getter and setter methods, constructors, equals and hashCode methods, and more [14]. For example:

- @NoArgsConstructor - generates no arguments constructor for an annotated class
- @Getter/@Settrer - generates getters and setters for attributes of an annotated class

3.4.2.4 Swagger

Swagger is an open-source software framework that allows developers to design, build, document, and consume RESTful web services. It provides a set of tools and specifications for creating and documenting APIs. With Swagger, developers can define API endpoints, request/response structures, data models, and authentication mechanisms in a standardized and machine-readable format [15].

3.4.3 Data tier

The data tier, sometimes called data access, is where the information processed by the application is stored and managed. This can be a relational database management system such as PostgreSQL, MySQL, Oracle and others [4].

PostgreSQL technology was chosen for this web application. It is a powerful, open-source object-oriented relational database management system (ORDBMS) that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads [16].

It provides flexible access to databases, efficient data organization and storage, and comprehensive management of records, including versioning and manipulation.

One notable advantage of PostgreSQL is its broad functionality and a rich set of data types that enable developers to handle diverse data requirements [16]. This flexibility is beneficial in the context of the mentioned web application.

PostgreSQL's compliance with the ACID principles is another significant feature. As stated by Bruce Momjian in the book "PostgreSQL: Introduction and Concepts", PostgreSQL ensures "atomicity, consistency, isolation, and durability", making it reliable and safeguarding data against failures and losses [17].

The database was the last layer of the architecture that was not described. Now the final structure of the project can be seen in the Figure 3.9, where all the basic technologies that will be used in the development of this web application are indicated.

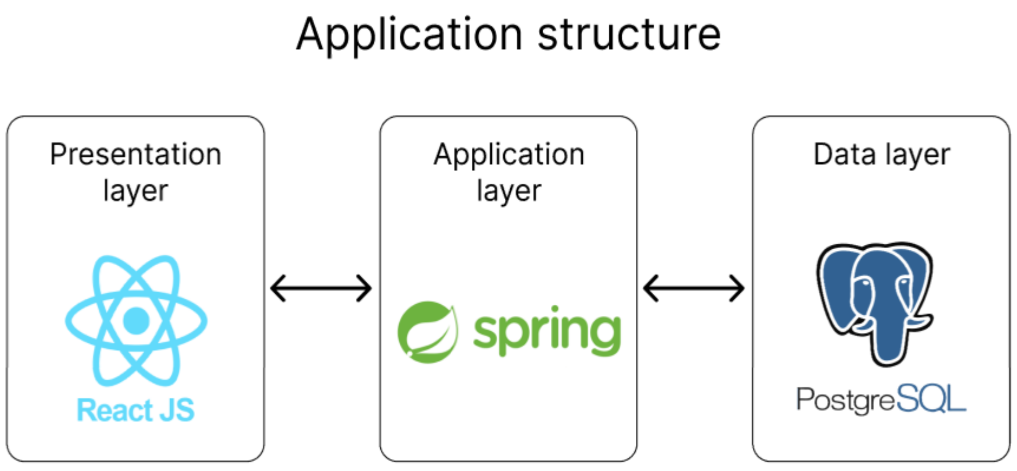


Figure 3.9: Final structure of application

Chapter **4**

Implementation

The web application was implemented according to the design described in the "Design" chapter. As mentioned earlier, the frontend is developed using React, the backend part is written in Java and PostgreSQL is used as a database.

4.1 Back-end

The back-end of the web application is organized into three layers:

- data layer
- business layer
- REST API layer

By dividing the back-end into these three layers, the application achieves separation of concerns, modularity, and scalability. Each layer has its specific responsibilities, allowing for easier maintenance, code reuse, and flexibility in making changes or enhancements to the application. More details about each level are described in subsections below.

4.1.1 Data layer

The data layer consists of two main blocks, as shown in the Figure 4.1:

- entities,
- repositories.

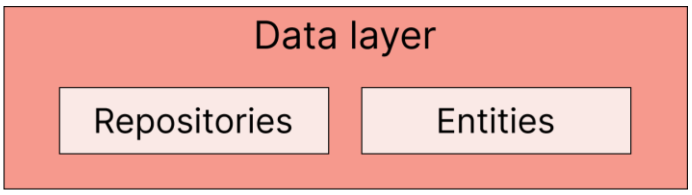


Figure 4.1: The data layer structure

4.1.1.1 Entities

In a database, an entity is represented as a set of attributes and is typically mapped to a table [18]. In other words, entities in JPA are nothing but POJOs representing data that can be persisted to the database.

The project contains a group of user entities, which consists of an abstract *User* class, from which two other classes *Donor* and *Indigent* are inherited, and enum *UserType*. Visually, these entities are shown in Figure 4.2.

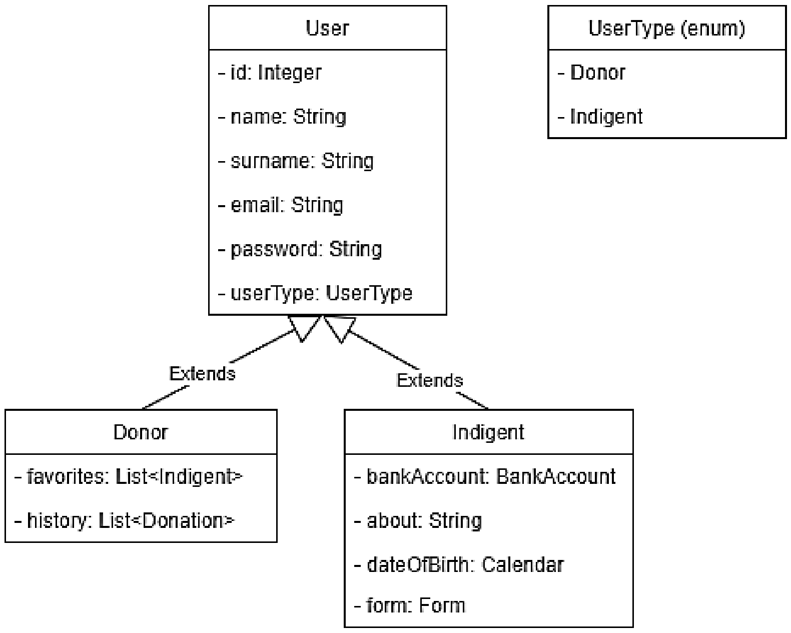


Figure 4.2: User entities group

In addition to the entities mentioned above, the project consists of:

- *BankAccount entity*- represents the bank account object
- *Donation* - describes the donation object
- *Document* - the object that stores the data of the uploaded document
- *Form* - represents the indigents form object
- *Report* - describes the treatment report object

The final representation of all entities and their relationships is depicted in the UML diagram in Figure 4.3

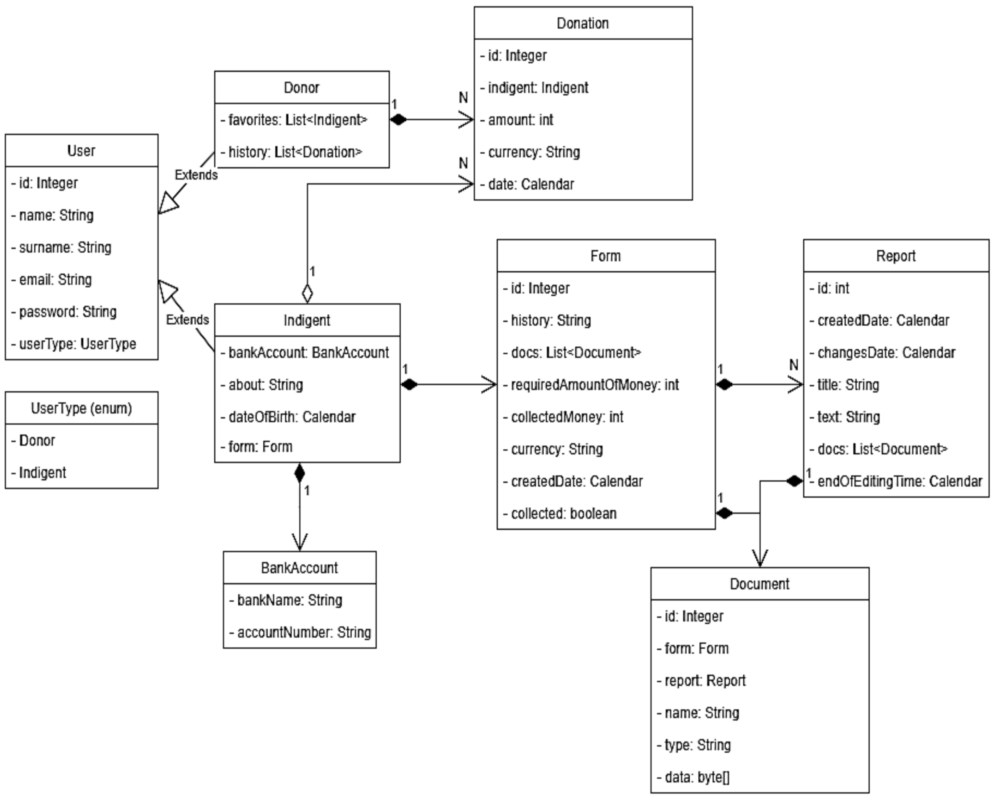
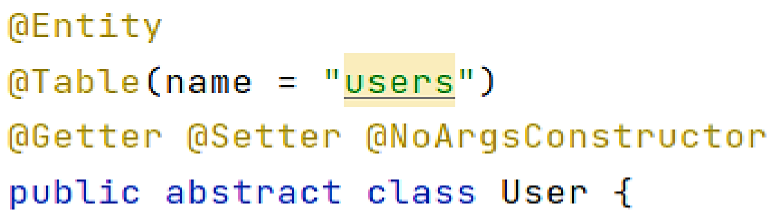


Figure 4.3: The UML diagram

In these classes the Lombok technology, which was mentioned earlier, was very useful. An example can be seen in the Figure 4.4



```
@Entity
@Table(name = "users")
@Getter @Setter @NoArgsConstructor
public abstract class User {
```

Figure 4.4: Example of using Lombok annotations

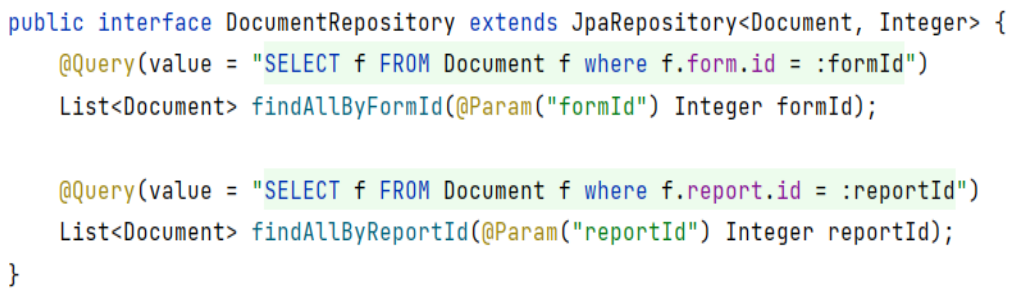
4.1.1.2 Repositories

In the Java and Spring ecosystems, repositories are key components that ensure uninterrupted communication with databases. Developers can use the capabilities of Spring Data JPA to simplify the implementation of the repository, as it offers built-in support for fundamental CRUD operations: create, read, update and delete. Moreover, Spring Data JPA gives developers the flexibility to define their own custom repository implementations, allowing them to create customized solutions if necessary [19].

A great example of a custom JpaRepository can be the interface *Document Repository*. It implements the following two methods:

- *findAllByFormId* - this method allows to find all the documents by the ID of the indigents form,
- *findAllByReportId* - a method that returns documents attached to a specific report.

Figure 4.5 represents a code snippet of the interface described above.



```
public interface DocumentRepository extends JpaRepository<Document, Integer> {
    @Query(value = "SELECT f FROM Document f where f.form.id = :formId")
    List<Document> findAllByFormId(@Param("formId") Integer formId);

    @Query(value = "SELECT f FROM Document f where f.report.id = :reportId")
    List<Document> findAllByReportId(@Param("reportId") Integer reportId);
}
```

Figure 4.5: Code snippet of interface Document Repository

The backend part of the project also contains repositories such as:

- *BankAccountRepostory*
- *DonationRepository*
- *FormRepository*
- *ReportRepository*
- *UserRepository*

It is also worth noting that some of them also contain custom functions for processing and searching data in the database.

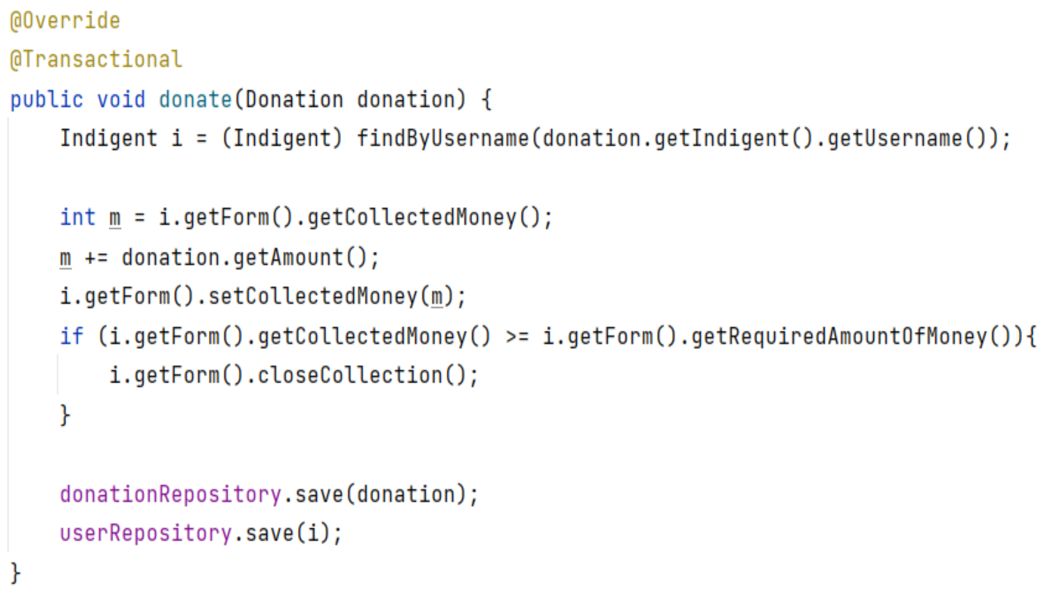
4.1.2 Business layer

The business or service layer contains "the logic that drives an application by coordinating the application's response to user input and its persistence and retrieval of domain objects from a database" [10]. It encapsulates complex business processes, handles validations, performs calculations, and organizes transformations according to the specific requirements and rules.

The business logic layer of the project consists of several service classes:

- *BankAccountService*
- *DocumentService*
- *ReportService*
- *UserService*

The most important service class is *UserService*. It describes the process of registration, updating user data, donation operation (code fragment in the Figure 4.6) and other processes. But it is worth noting that the authorization process has its own specifics in this case, which is described in more detail below in the Security section.



```java
@Override
@Transactional
public void donate(Donation donation) {
    Indigent i = (Indigent) findByUsername(donation.getIndigent().getUsername());

    int m = i.getForm().getCollectedMoney();
    m += donation.getAmount();
    i.getForm().setCollectedMoney(m);
    if (i.getForm().getCollectedMoney() >= i.getForm().getRequiredAmountOfMoney()){
        i.getForm().closeCollection();
    }

    donationRepository.save(donation);
    userRepository.save(i);
}
```

Figure 4.6: Donate function code fragment of User Service class

4.1.3 REST API layer

The REST API layer consists of two main blocks:

- controllers
- data transfer objects (DTOs)

4.1.3.1 Controllers

Controllers are components that handle incoming requests and provide responses back to the client. They act as an intermediary between the user interface (front-end) and the back-end logic. Controllers receive and interpret HTTP requests, extract relevant data, invoke appropriate actions or services, and generate an HTTP response to be sent.

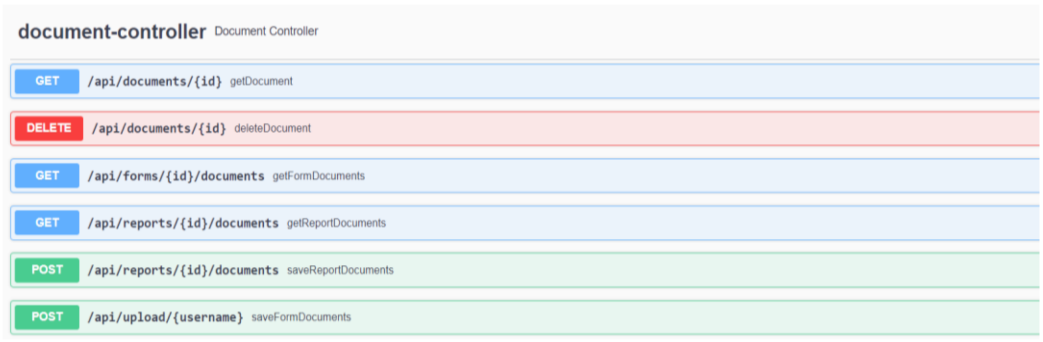
All API endpoints are described in four controllers:

- *DocumentController*
- *DonationController*
- *ReportController*
- *UserController*

For example, as the name implies, *DocumentController* is designed for user operations. Here are some of them:

- create new document for form/report
- receive documents full information
- get all indigent forms documents
- get reports documents
- delete document

The mentioned operations are shown in Figure 4.7.



document-controller Document Controller

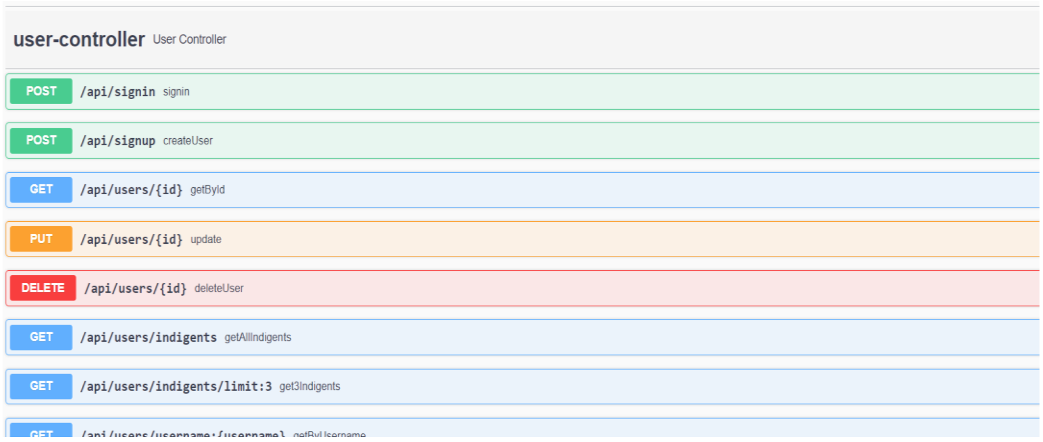
GET /api/documents/{id} getDocument
DELETE /api/documents/{id} deleteDocument
GET /api/forms/{id}/documents getFormDocuments
GET /api/reports/{id}/documents getReportDocuments
POST /api/reports/{id}/documents saveReportDocuments
POST /api/upload/{username} saveFormDocuments

Figure 4.7: DocumentController endpoints using Swagger

The *UserController* is responsible for operations related to user management, such as:

- registration
- login
- profile update
- delete account

Moreover, this class contains GET methods that are used to get data from the database and then use them on the frontend side. (Figure 4.8)


user-controller User Controller

```
POST    /api/signin  signin
POST    /api/signup  createUser
GET     /api/users/{id}  getById
PUT     /api/users/{id}  update
DELETE  /api/users/{id}  deleteUser
GET     /api/users/indigents  getAllIndigents
GET     /api/users/indigents/limit:3  get3Indigents
GET     /api/users/username:{username}  getByUsername
```

Figure 4.8: Visual interpretation of UserController using Swagger

4.1.3.2 DTOs

A DTO is an object that carries data between processes in order to reduce the number of methods calls between client and server [10]. The DOT pattern helps to minimize the overhead associated with multiple method calls by bundling related data into a single object, reducing network traffic and improving performance.

The project includes the following DTOs:

- *DocumentDTO*
- *DonationDTO*
- *ReportDTO*
- *UserDTO*

In addition to the above list, the project also has a *CredentialsDTO*, which contains the *username* and *password* attributes. This DTO is used when authorizing users. How this process goes is described in the "Security" section.

Having described all the levels of the backend structure, the final diagram can be depicted as shown in the Figure 4.9

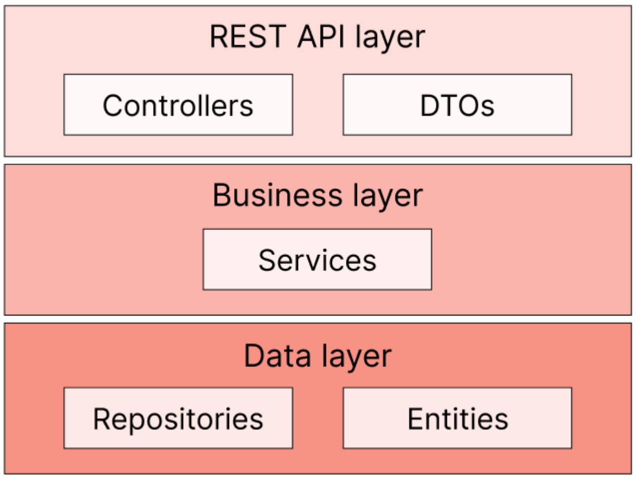


Figure 4.9: Backend structure

4.2 Security

As mentioned earlier, the system has two types of users: unauthorized and authorized, which in turn is divided into Donor and Needy. Each of these types of users has its own restrictions that should not be violated. Moreover, the platform must also securely store user data from various hacks and attacks.

To achieve these goals, the project implemented a security system using JWT. JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA [20].

Below is a diagram describing the principle of creating and receiving a JWT during authorization. (Figure 4.10)

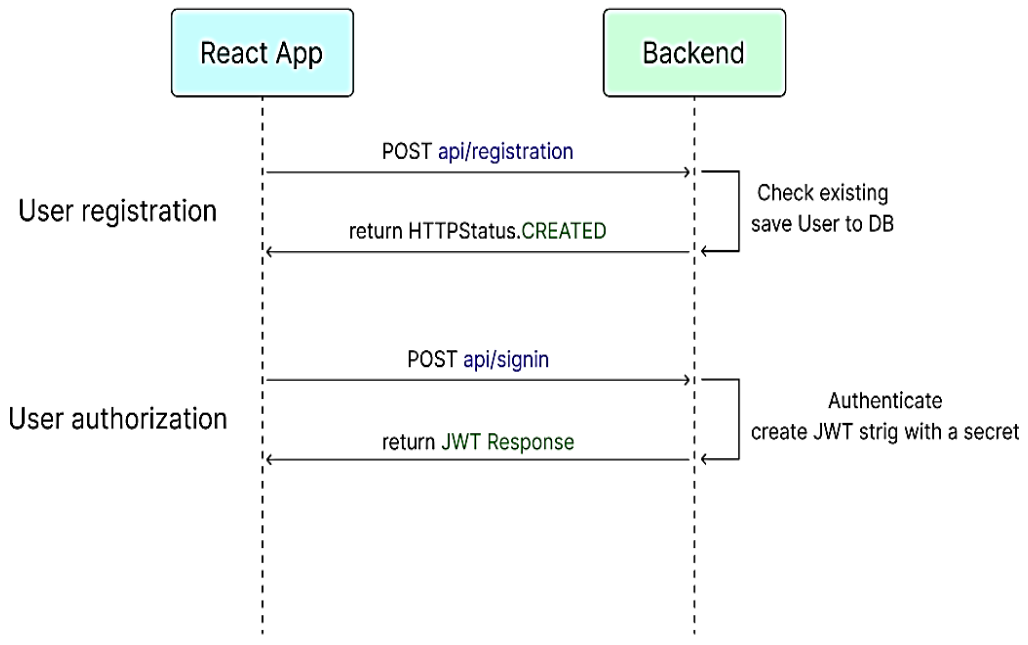
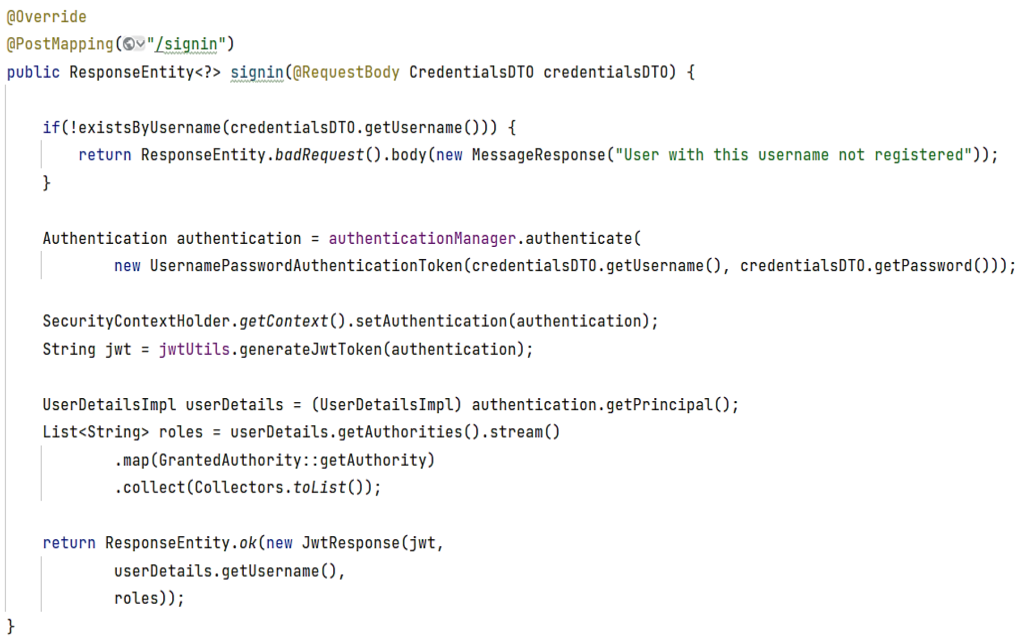


Figure 4.10: Sequence diagram of authorization and registration processes

During the registration process, users provide all the required information based on their selected user type. This data is sent to the server, where it is processed, hashed and stored securely in the database. (Figure 4.11). Upon successful registration, a *HTTP.CREATED* status is returned to the frontend, indicating that the registration was completed successfully. Next, the authorization process is carried out.

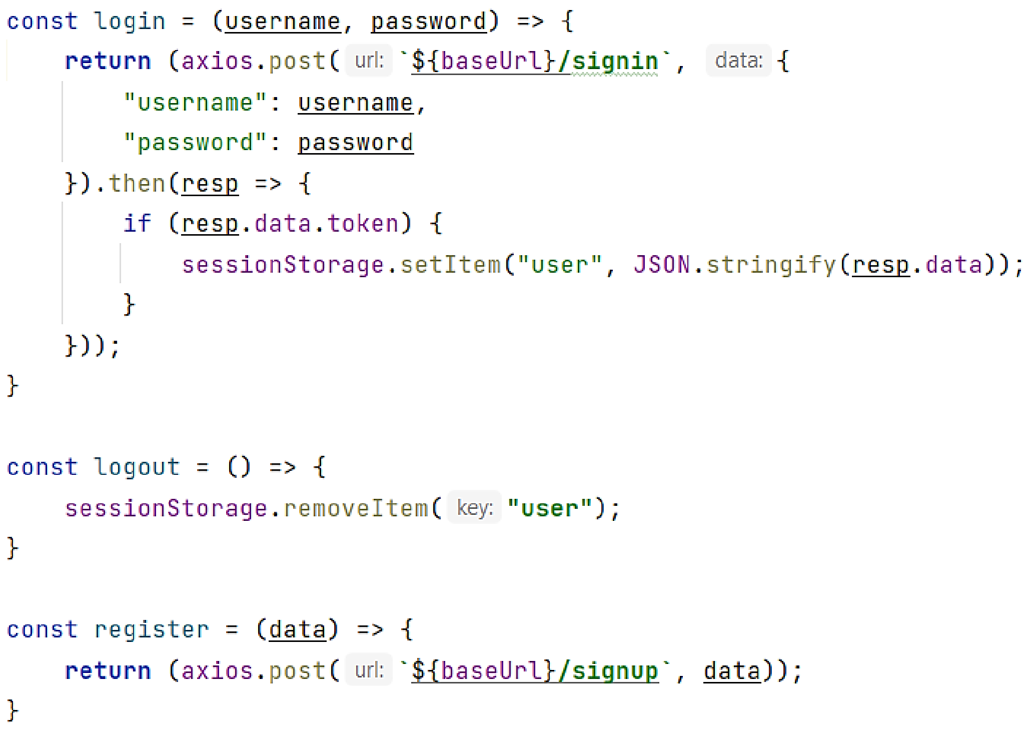
During authentication, the user enters their credentials for verification. If the data is correct, a new JWT is generated on the server and returned to the user as confirmation of successful authorization.

The received JWT Response contains a token, user information and his role. The *auth.service.js* class on frontend side is responsible for getting, saving and deleting JWT to/from sessionStorage. (Figure 4.12). Based on the existing or missing token, the React sets the visibility of pages for specific users.



```
@Override
@PostMapping("/signin")
public ResponseEntity<?> signin(@RequestBody CredentialsDTO credentialsDTO) {

    if(!existsByUsername(credentialsDTO.getUsername())) {
        return ResponseEntity.badRequest().body(new MessageResponse("User with this username not registered"));
    }

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

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

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

    return ResponseEntity.ok(new JwtResponse(jwt,
            userDetails.getUsername(),
            roles));
}
```

Figure 4.11: Implementation of the registration process on the server side



```
const login = (username, password) => {
    return (axios.post( url: `${baseUrl}/signin`,  data: {
        "username": username,
        "password": password
    }).then(resp => {
        if (resp.data.token) {
            sessionStorage.setItem("user", JSON.stringify(resp.data));
        }
    }));
}

const logout = () => {
    sessionStorage.removeItem( key: "user");
}

const register = (data) => {
    return (axios.post( url: `${baseUrl}/signup`, data));
}
```

Figure 4.12: Code fragment of auth.service.js

4.3 Front-end

The front-end part of the project is written using the React library. The key feature of React is composition of components, which can have its own internal state, properties (props), and methods [21].

In React, a component is a reusable, self-contained unit of UI that can be composed and nested to build complex UIs. They are organized in a hierarchical structure where parent components can have child components, forming a tree-like structure [22].

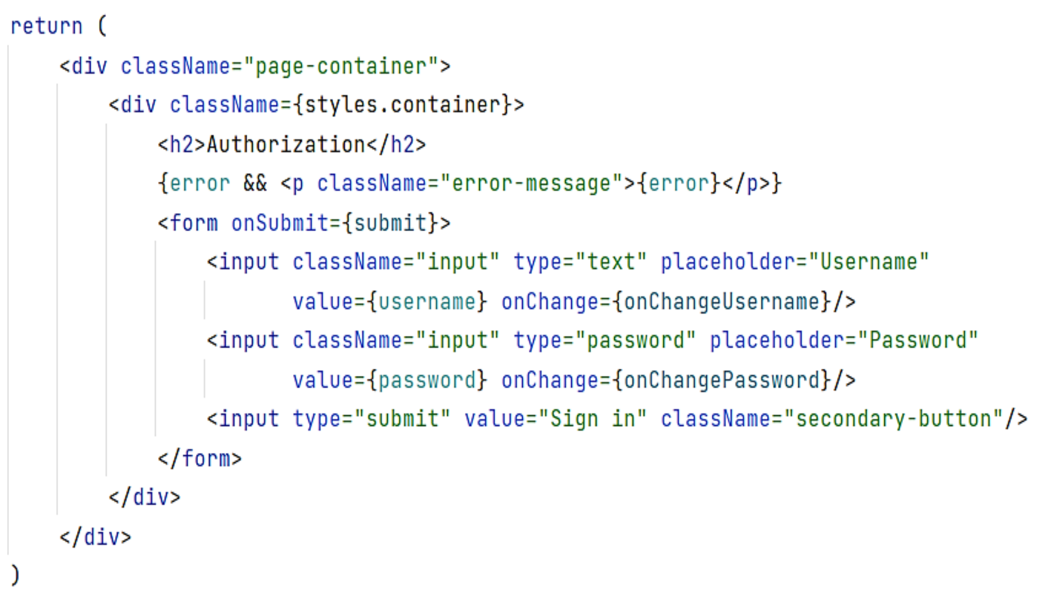
4.3.1 Components of frontend

Frontend components of a web application can be divided into 4 main groups:

- authorization components
- components of profile
- components of reports
- rest components

4.3.1.1 Authorization and registration pages

The authorization page contains a small form that must be filled in with credentials to log in. The code snippet is shown in the Figure 4.13

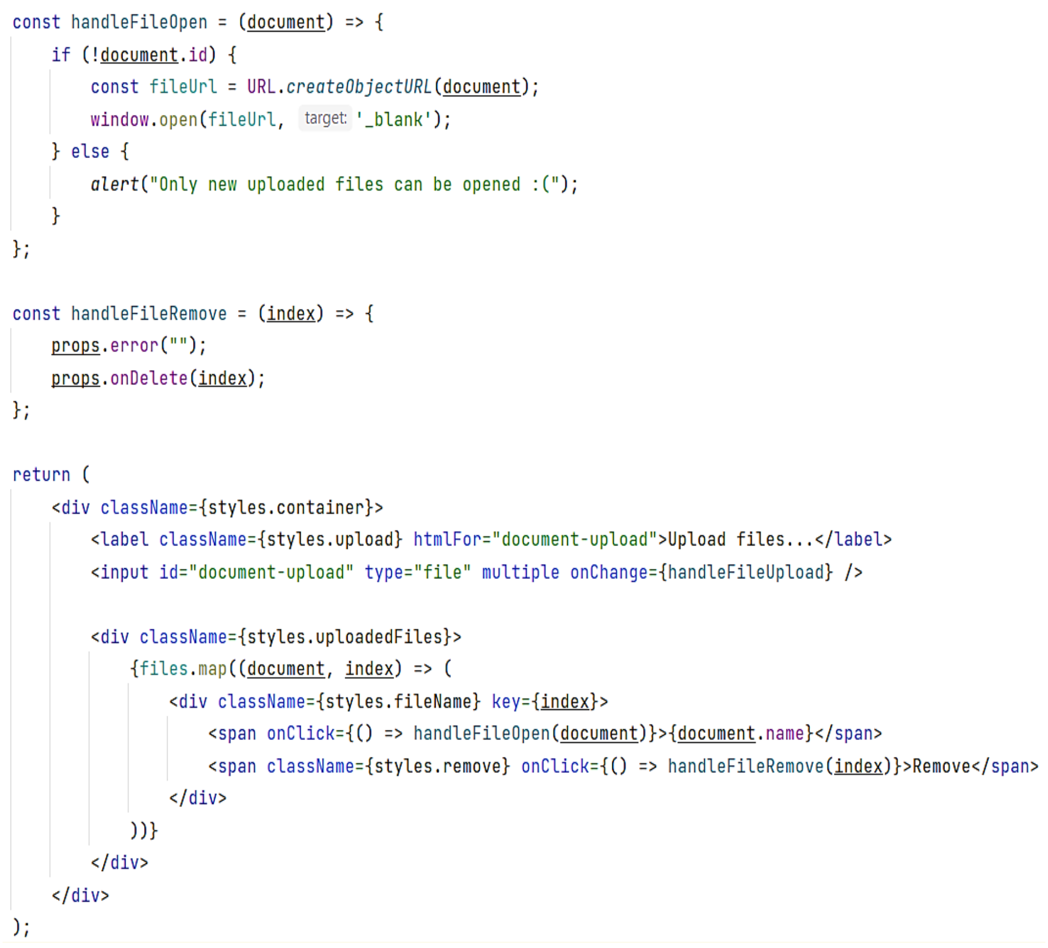


```
return (
    <div className="page-container">
        <div className={styles.container}>
            <h2>Authorization</h2>
            {error && <p className="error-message">{error}</p>}
            <form onSubmit={submit}>
                <input className="input" type="text" placeholder="Username"
                       value={username} onChange={onChangeUsername}/>
                <input className="input" type="password" placeholder="Password"
                       value={password} onChange={onChangePassword}/>
                <input type="submit" value="Sign in" className="secondary-button"/>
            </form>
        </div>
    </div>
)
```

Figure 4.13: Code snippet of AuthorizationPage.js

On the contrary, the registration page has a more complex structure, which is presented in Figures B.1-B.4, located in the appendix. Depending on the selected type of user (donor or indigent), the form to fill out changes dynamically.

In addition to the basic HTML elements, this page also contains custom elements, such as *CustomDatePicker*, created using the additional Material UI library, *CurrencySelector* and *DocumentUploader*. Each of these elements is designed as a separate component that performs certain operations. Part of the code from the *DocumentUploader* component is shown in Figure 4.14.



```
const handleFileOpen = (document) => {
    if (!document.id) {
        const fileUrl = URL.createObjectURL(document);
        window.open(fileUrl,  target: '_blank');
    } else {
        alert("Only new uploaded files can be opened :(");
    }
};

const handleFileRemove = (index) => {
    props.error("");
    props.onDelete(index);
};

return (
    <div className={styles.container}>
        <label className={styles.upload} htmlFor="document-upload">Upload files...</label>
        <input id="document-upload" type="file" multiple onChange={handleFileUpload} />

        <div className={styles.uploadedFiles}>
            {files.map((document, index) => (
                <div className={styles.fileName} key={index}>
                    <span onClick={() => handleFileOpen(document)}>{document.name}</span>
                    <span className={styles.remove} onClick={() => handleFileRemove(index)}>Remove</span>
                </div>
            ))}
        </div>
    </div>
);
```

Figure 4.14: Part of the code of DocumentUploader component

4.3.1.2 Components of profile

This group includes:

- *ProfilesPage* where all profiles of users who need financial assistance are displayed
- *ProfileDetailPage*
- *EditProfilePage* to edit user data
- *ProfilePage*

ProfilePage component responsible for displaying the personal profile of the user registered as a donor. While the *ProfileDetailPage* component is carried out the representation of indigents data.

This page slightly changes its content depending on whether the current user is authorized or not (using JWT). For an unauthorized user or
an authorized person in need, the donation function and the ability to add to favorites is not available. However, if the authorized user is the owner, then the page is transformed into a personal profile with all possible additional functions.

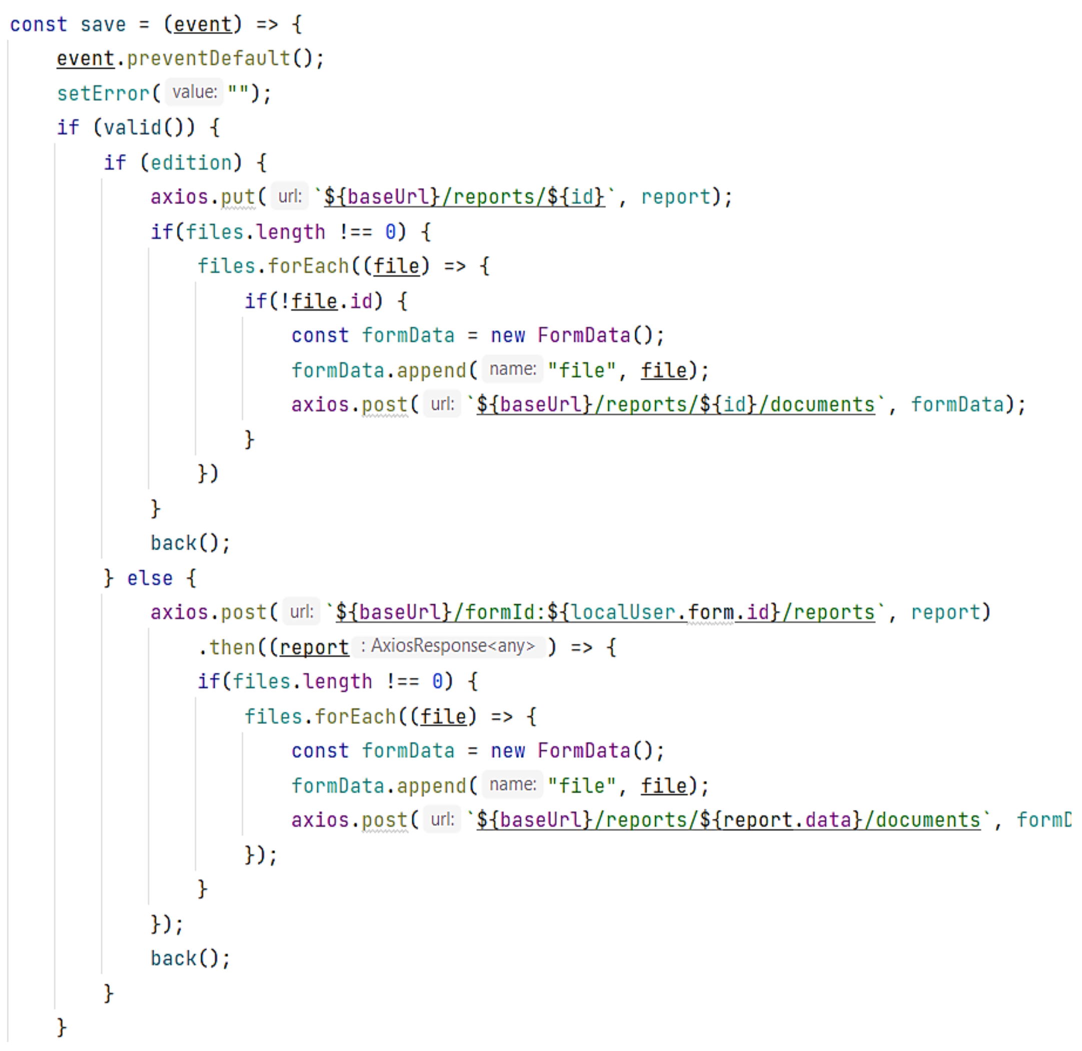
This *ProfileDetailPage* also contains embedded child components *ReportBlock*, which is responsible for displaying report information, and *MakeDonationWindow*, which opens after clicking on the "Donate" button and *AdditionalButtons* to change data, create new reports and delete an account.

4.3.1.3 Pages related to reports

This group consists of two components:

- *ReportsPage* - a page displaying all the reports that were created by selected indigent,
- *ReportCreatePage* in which there is a form for creating a new report or or editing an existing one.

Below is a code snippet that demonstrates the creation or editing operation in the *ReportCreatePage*. (Figure 4.15)



```
const save = (event) => {
    event.preventDefault();
    setError( value: "");
    if (valid()) {
        if (edition) {
            axios.put( url: `${baseUrl}/reports/${id}`, report);
            if(files.length !== 0) {
                files.forEach((file) => {
                    if(!file.id) {
                        const formData = new FormData();
                        formData.append( name: "file", file);
                        axios.post( url: `${baseUrl}/reports/${id}/documents`, formData);
                    }
                })
            }
            back();
        } else {
            axios.post( url: `${baseUrl}/formId:${localUser.form.id}/reports`, report)
                .then((report : AxiosResponse<any> ) => {
                if(files.length !== 0) {
                    files.forEach((file) => {
                        const formData = new FormData();
                        formData.append( name: "file", file);
                        axios.post( url: `${baseUrl}/reports/${report.data}/documents`, formD
                    });
                }
            });
            back();
        }
    }
```

Figure 4.15: Part of the code of ReportCreatePage component

4.3.1.4 Rest components

In addition to the components mentioned earlier, which acted as child components, the list includes:

- *DeleteWindow* - confirmation window for irrevocable deletion
- *DonationHistoryBlock* - a component that displays information about the made donation
- *IndigentProfileBlock* is displayed on the “Profiles” page and contains brief information about indigent
- *Header* with platform navigation and logo

Chapter **5**

Testing

Testing is an essential part of the development lifecycle, as it helps to identify and rectify defects, ensure software quality, and build confidence in the reliability and functionality of the system.

Three types of testing were carried out on the web platform:

- unit testing,
- scenario testing,
- usability testing.

5.1 Unit testing

“Unit testing is a practice that includes automated checks to verify the behavior of individual components of a program. The goal of unit testing is to detect defects in individual modules before their integration into the system as a whole.” [23].

In the unit project, two main groups are covered by tests:

- controllers,
- services.

An additional, software called Diffblue was used to write the tests. Diffblue Cover is designed to speed up the process of creating unit tests, which traditionally requires manual effort from developers. By automating this process, Diffblue aims to improve software quality, reduce the time and effort required for testing, and help developers catch bugs earlier in the development cycle [24].

Controller tests include:

- *DocumentControllerTest*
- *ReportControllerTest*
- *UserControllerTest*

While the service test group consists of:

- *DocumentServiceTest*
- *ReportServiceTest*
- *UserServiceTest*

5.2 Scenario testing

Scenario testing is a software testing activity that uses scenarios: hypothetical stories to help the tester work through a complex problem or test system [25].

In scenario-based testing, test cases are designed to simulate real-world scenarios that users may encounter when interacting with the software application. These scenarios typically represent specific sequences of user actions and system responses.

For this purpose, various scenarios have been created, here are some of them.

5.2.1 Registration

Description: The new user wants to register in web platform.

Steps:

1. Go to the registration page by clicking the "Sign up" button in the upper right corner.
2. Fill in the form with bad data and press the "Sign up" button.
3. Verify that an error message is displayed.
4. Fill in the all the necessary fields in accordance with the requirements and click the "Sign up" button.

Expected results:

- At the second point the system should check the input data for:
    a. absence of blank fields,
    b. absence of numbers and symbols in the “Name” and “Surname” fields,
    c. correctness of the format of the entered email,
    d. correspondence of the “Password” and “Repeat password” fields.

  After unsuccessful validation display an error message.
- By clicking on the button after filling out the form correctly, the platform should redirect the user to the profile page.

Conclusion: validation of the completed form and redirection to the desired page work correctly.

5.2.2 Username and password changing

Description: the authorized donor wants to change his username.

Steps:

1. Click on the button “Edit”.
2. Enter a new username.
3. Check box “Change password” and press button “save”.
4. Verify that a message is displayed indicating a successful saving.
5. Log out from system.
6. Sign in with new credentials (username and password).

Expected results:

- By clicking on button “Edit” system should redirect user to the Edit profile page.
- After checking the box inputs for entering a new password should be displayed.
- If the new username has not been registered yet, the system should display the message “Changes saved".
- After logging out, the user should be redirected to the main page.
- Upon successful authorization, the user must be on the profile page.

Conclusion: authorization with the new data was not successful because the new password was not saved in the database.

5.2.3 Donate

Description: donor wants to donate money.

Steps:

1. Click on “Donate” button.
2. Enter the desired amount in the dialog window that appears.
3. Select currency and click on the button.
4. Make sure that a message about a successful transaction displays.
5. Verify that the amount of funds collected has increased by the specified amount.
6. Check the section “History” in the profile page for the appearance of donation in it.

Expected results:

- By clicking on “Donate” button dialog window should be displayed.
- The system must check the field for emptiness and the entered value, which cannot be less than 1. In case of correctness, display a message about the successful completion of the donation.
- The number of donated funds should increase.
- The produced donation should be displayed in the section “History”.

Conclusion: the operation works correctly, but the donation was not recorded in the database.

5.2.4 Create report with documents (with preview)

Description: authorized indigent wants to create a new treatment report.

Steps:

1. Press the “New report”.
2. Fill out the form and upload documents.
3. Click on the name of uploaded document and make sure that the document is displayed in a new window.

4. Create document and check appearance of report in section “Actual information”

Expected results:

- After clicking on button “New report”, the page for creation should open.
- All names of the documents should be displayed under the upload button.
- Clicking on the name of the document should open a new window with the content of the file.
- New report should appear in “Actual information” section.

Conclusion: the document upload process was not allowed to select more than one file.

5.2.5 Delete document from report

Description: indigent wants to delete document attached to the report

Steps:

1. Press on button “Edit” near report
2. Remove document and save changes
3. Make sure that document was deleted successfully.

Expected results:

- The page for editing should be displayed by clicking on button “Edit”.
- Document should be removed.

Conclusion: delete operation works correctly.

5.3 Usability testing

Usability testing is a technique used in software development to evaluate the usability of a system, application, or website by observing real users as they interact with it. It aims to identify usability issues, gather user feedback, and assess the overall user experience [26].

This testing was conducted with the participation of three users of different ages and fields of activity. Each of them was given the opportunity to familiarize themselves with the application, and then share their impressions of using it.

The tests were conducted remotely using AnyDesk and TeamViewer applications, which allowed participants to interact with the application, and through video calls they shared their comments, recommendations and wishes.

5.3.1 Tester 1

The first participant, a student from a technical university with small testing experience, noticed a couple of issues while evaluating the system.

The first thing that caught their eye was the navigation buttons on the main page. They suggested that the navigation buttons on the main page seemed unnecessary and could be eliminated since they didn't serve any significant functional purpose.

Additionally, after making changes to user data in their profile, the participant felt that the system lacked a clear indication of successful data saving. To address this, they recommended implementing a pop-up browser window with a message instead of relying solely on the presence of highlighted green text above the form.

5.3.2 Tester 2

The second participant, a manager in an educational center over 40 years old, had no experience in such matters.

They identified a couple of defects during their evaluation of the system. Firstly, they observed that the input fields in the registration and authorization forms appeared faded, blending with the background image and lacking clarity. To address this issue, the participant suggested improving the visual expression of the input fields.

Furthermore, the participant noticed that the red field headers were challenging to differentiate from error messages, potentially causing confusion. They recommended using a different color for the field headers to enhance visual distinction.

Additionally, the participant proposed replacing the user's *id* displayed in the profile with their *username*. This change aimed to facilitate easy identification and memorization of other users' profiles.

5.3.3 Tester 3

The third participant, a high school student with no testing experience, found the design of the application to be concise. However, they encountered an inconvenience while attempting to change the indigents data. The process of editing several sections required constant switching between pages, which caused frustration. To address this issue, the participant suggested implementing a single page for editing that would consolidate all the sections in one place.

By proposing the creation of a unified editing page with all sections available simultaneously, the participant aimed to streamline the editing process and enhance user convenience.

5.4 Summary of testing

The tests carried out helped to identify errors in the functionality of the application, shortcomings of the user interface, as well as to form an opinion about the developed system as a whole.

Based on the tests conducted, the above-mentioned comments, suggestions and recommendations, the necessary corrections and bug fixes were made to the web application.

Conclusion

The purpose of the bachelor's thesis was to analyze existing platforms that allow making donations to people in need of medical treatment and to develop a prototype of a new improved platform.

Market analysis helped identify the functionality and shortcomings of existing platforms, which served as a foundation for designing the future web application. The prototype was created based on the design and selected technologies, incorporating essential features and providing a user-friendly and intuitive interface. Further, several types of testing were conducted to identify errors in the system, improve the user interface, as well as gather valuable feedback and ideas for further project development.

Despite the fact that the biggest and most important part of the project has been successfully implemented, there is always something to strive for and improve. In the future, the functionality of the project will only expand. First of all, it is necessary to implement the technology of verifying the authenticity of uploaded documents, as well as integrate the payment system. Moreover, thanks to the conducted testing, such ideas as the introduction of a filter for sorting questionnaires of those in need, a search engine, a bonus system for donors and others were identified.

The final version of the web application is available on GitLab[6].

[6] https://gitlab.com/aya.rakhimova2002/bachelor_project

List of literature

[1] Karl E. Wiegers. Software Requirements [online]. 2013, [Cited 2022-19-11]. ISBN-13: 978-0-7356-7966-5. Available from: https://www.booksfree.org/wp-content/uploads/2022/03/Software_Requirements_3rd_Edition_compressed.pdf

[2] Lauesen, Soren. Software Requirements: Styles and Techniques. 2002. Addison-Wesley Professional. ISBN-13: 978-0321122476.

[3] Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed [online]. 2004, [Cited 2022-20-11]. Available from: http://bsituos.weebly.com/uploads/2/5/2/5/25253721/applying-uml-and-patterns-3rd.pdf

[4] What is three-tier architecture? [online]. IBM. [Cited 2022-26-11]. Available from: https://www.ibm.com/topics/three-tier-architecture#:~:text=The%20chief%20benefit%20of%20three,without%20impacting%20the%20other%20tiers.

[5] What is Vue? [online]. Vue. [Cited 2022-26-11]. Available from: https://vuejs.org/guide/introduction.html#what-is-vue

[6] React (software) [online]. Wikipedia. [Cited 2022-26-11]. Available from: https://en.wikipedia.org/wiki/React_(JavaScript_library)

[7] Introduction to Angular concepts [online]. Google. [Cited 2022-26-11]. Available from:
https://angular.io/guide/architecture

[8] Sass documentation [online]. MacStadium. [Cited 2023-20-05]. Available from: https://sass-lang.com/documentation/

[9] Material UI - Overview [online]. Material UI. [Cited 2023-20-05]. Available from: https://mui.com/material-ui/getting-started/overview/

[10] Fowler, Martin. Patterns of Enterprise Application Architecture. 2002. Addison-Wesley Professional. ISBN-13: 978-0321127426.

[11] Gosling, James. The Java Language Specification [online]. 2002, [Cited 2022-27-11]. Oracle. Available from: https://www.oracle.com/java/technologies/introduction-to-java.html#318

[12] Gosling, James. The Java Virtual Machine Specification [online]. 2023 [Cited 2023-15-05]. Oracle. Available from: https://docs.oracle.com/javase/specs/jvms/se20/jvms20.pdf

[13] Walls, Craig. Spring in Action. 5th ed. 2018. Manning. ISBN-13: 978-1617294945.

[14] Borowiec, Rafal. Mastering Spring Boot 2.0. 2018. Packt Publishing. ISBN-13: 978-1787127562.

[15] OpenAPI Specification [online]. SmartBear. Cited [2023-15-05]. Available from: https://swagger.io/resources/open-api/

[16] What is PosgreSQL? [online]. The PostgreSQL Global Development Group. [Cited 2023-15-05]. Available from: https://www.postgresql.org/about/

[17] Momjian, Bruce. PostgreSQL: Introduction and Concepts [online]. 2001, [Cited 2023-15-05]. Addison-Wesley Professional. Available from: http://www.foo.be/docs-free/aw_pgsql_book.pdf

[18] Elmasri, R. & Navathe, S. B. Fundamentals of Database Systems. 2016. Pearson. ISBN-13: 978-0133970777.

[19] Spring Data JPA - Reference Documentation. Spring. [Cited 2023-16-05]. Available from:

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.custom-implementations

[20] Introduction to JSON Web Tokens [online]. Auth0. [Cited 2023-16-05]. Available from:
https://jwt.io/introduction

[21] React documentation [online]. Meta Open Source. [Cited 2023-16-05]. Available from:
https://react.dev/learn

[22] Wieruch, R. The Road to React [online]. 2018, [Cited 2023-16-05]. Available from:
https://sd.blackball.lv/library/the_road_to_react_(2020).pdf

[23] Martin, R. C. Clean Code: A Handbook of Agile Software Craftsmanship [online]. 2008, [Cited 2023-17-05]. Prentice Hall. Available from:
https://thixalongmy.haugiang.gov.vn/media/1175/clean_code.pdf

[24] Diffblue Cover [online]. Diffblue Ltd. [Cited 2023-17-05]. Available from:
https://www.diffblue.com/products

[25] Cem Kaner. An Introduction to Scenario Testing [online]. 2003, [Cited 2023-20-05]. Available from:
https://kaner.com/pdfs/ScenarioIntroVer4.pdf

[26] Dumas, J.S. and Redish, J.C. A Practical Guide to Usability Testing [online]. 1999, [Cited 2023-17-05]. Available from:
https://www.jedbrubaker.com/wp-content/uploads/2013/03/Dumas-99.pdf

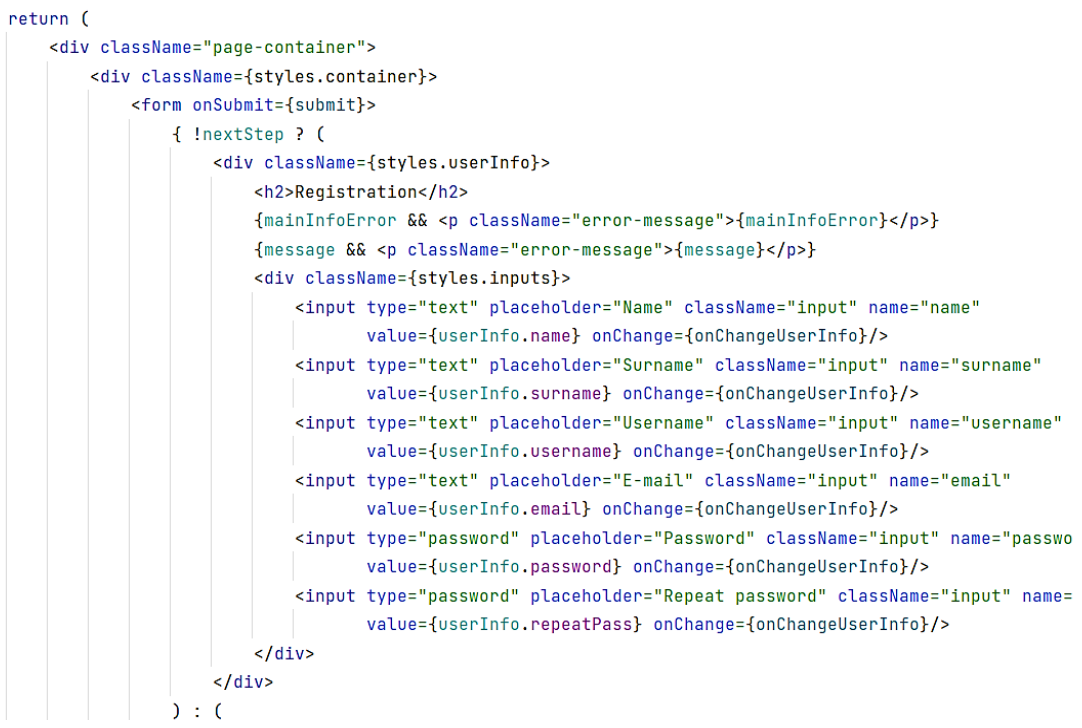
Appendix **A**

Acronyms

JS	JavaScript
REST	Representation state transfer
API	Application Programming Interface
SQL	Structured Query Language
NFR	Non-functional requirement
UI	User interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
JSX	JavaScript XML
XML	Extensible Markup Language
JRE	Java Runtime Environment
JVM	Java Virtual Machine
ORDBMS	Object Oriented Relational Database Management System
ACID	Four key properties of a transaction: atomicity, consistency, isolation, and durability
POJO	Plain Old Java Object
JPA	Java Persistence API
UML	Unified Modeling Language
DTO	Data Transfer Object
HTTP	HyperText Transfer Protocol
DB	Database
JSON	JavaScript Object Notation
JWT	JSON Web Token
RFC 7159	JSON Data Interchange Format
HMAC	Hash-based message authentication code
RSA	(Rivest–Shamir–Adleman) Public-key cryptosystem
ECDSA	Elliptic Curve Digital Signature Algorithm

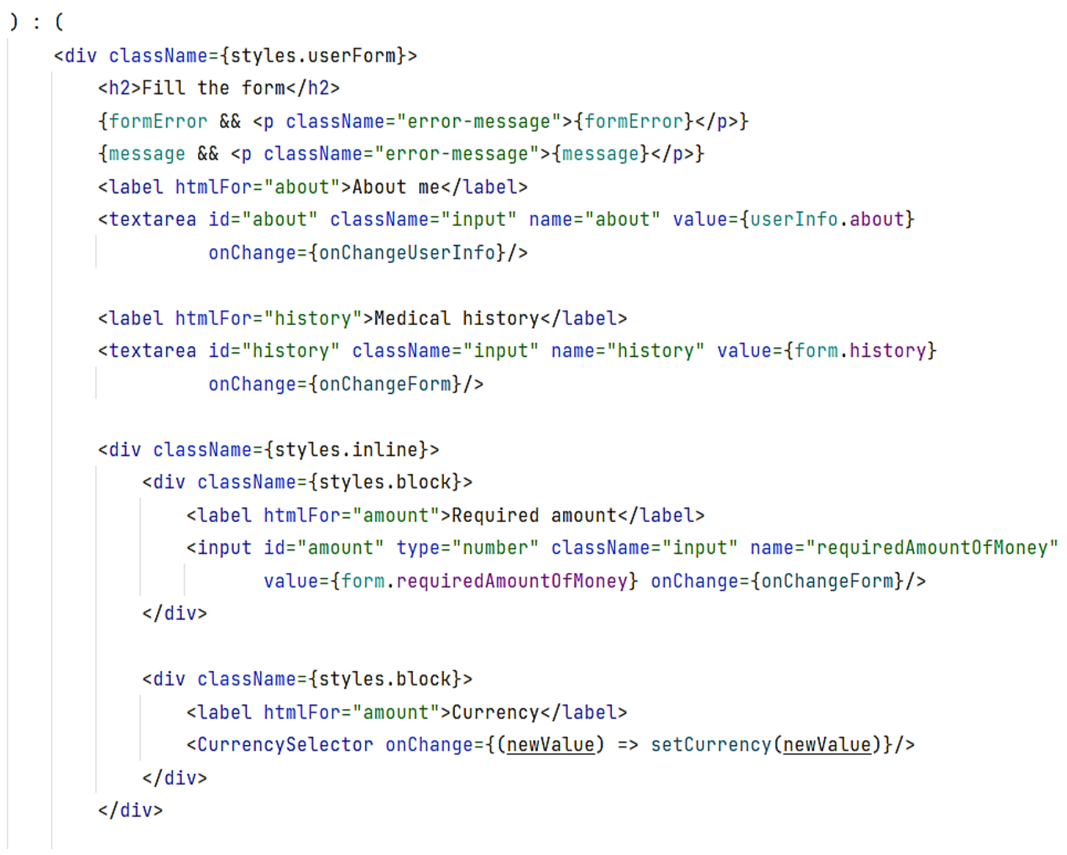
Appendix **B**

Code fragments



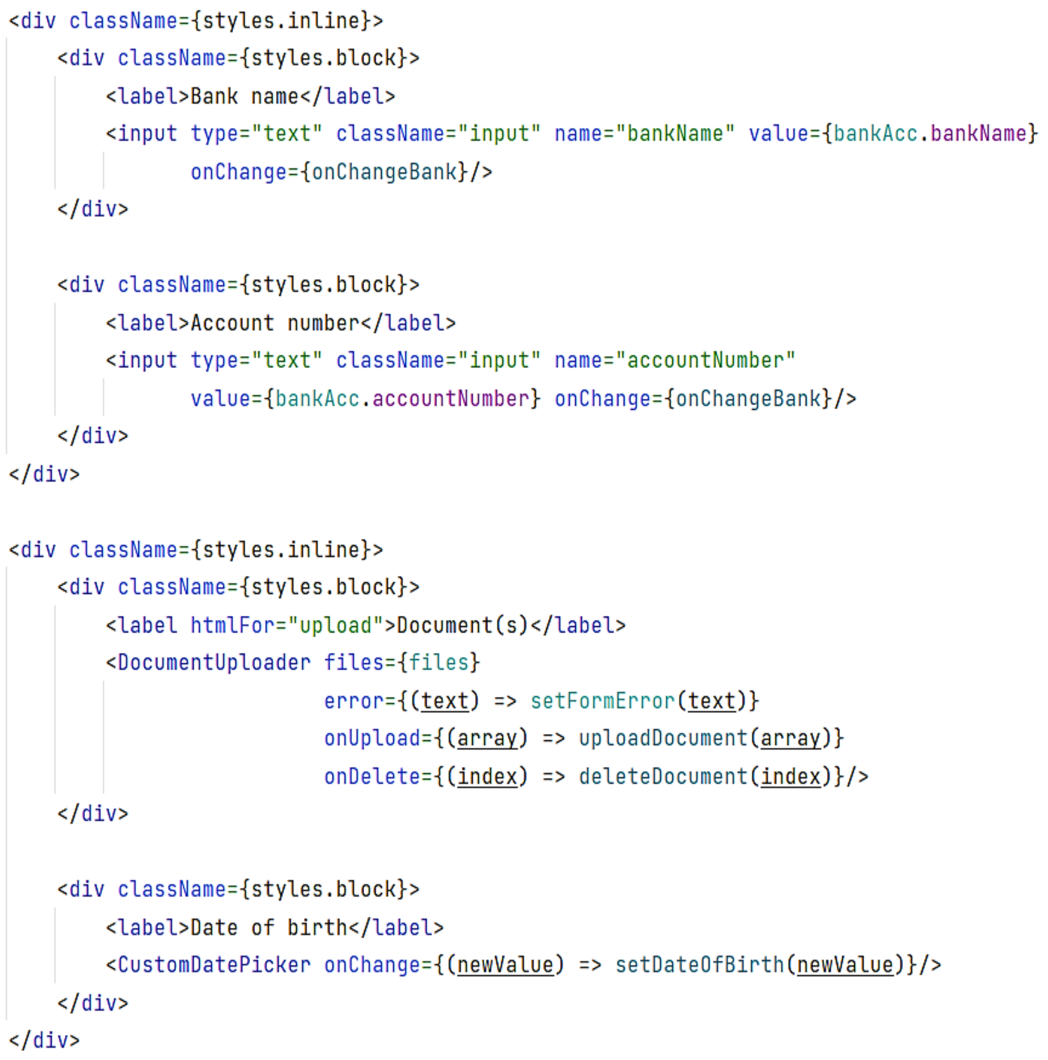
```
return (
    <div className="page-container">
        <div className={styles.container}>
            <form onSubmit={submit}>
                { !nextStep ? (
                    <div className={styles.userInfo}>
                        <h2>Registration</h2>
                        {mainInfoError && <p className="error-message">{mainInfoError}</p>}
                        {message && <p className="error-message">{message}</p>}
                        <div className={styles.inputs}>
                            <input type="text" placeholder="Name" className="input" name="name"
                                   value={userInfo.name} onChange={onChangeUserInfo}/>
                            <input type="text" placeholder="Surname" className="input" name="surname"
                                   value={userInfo.surname} onChange={onChangeUserInfo}/>
                            <input type="text" placeholder="Username" className="input" name="username"
                                   value={userInfo.username} onChange={onChangeUserInfo}/>
                            <input type="text" placeholder="E-mail" className="input" name="email"
                                   value={userInfo.email} onChange={onChangeUserInfo}/>
                            <input type="password" placeholder="Password" className="input" name="passwo
                                   value={userInfo.password} onChange={onChangeUserInfo}/>
                            <input type="password" placeholder="Repeat password" className="input" name=
                                   value={userInfo.repeatPass} onChange={onChangeUserInfo}/>
                        </div>
                    </div>
                ) : (
```

Figure B.1: RegistrationPage.js code fragment №1



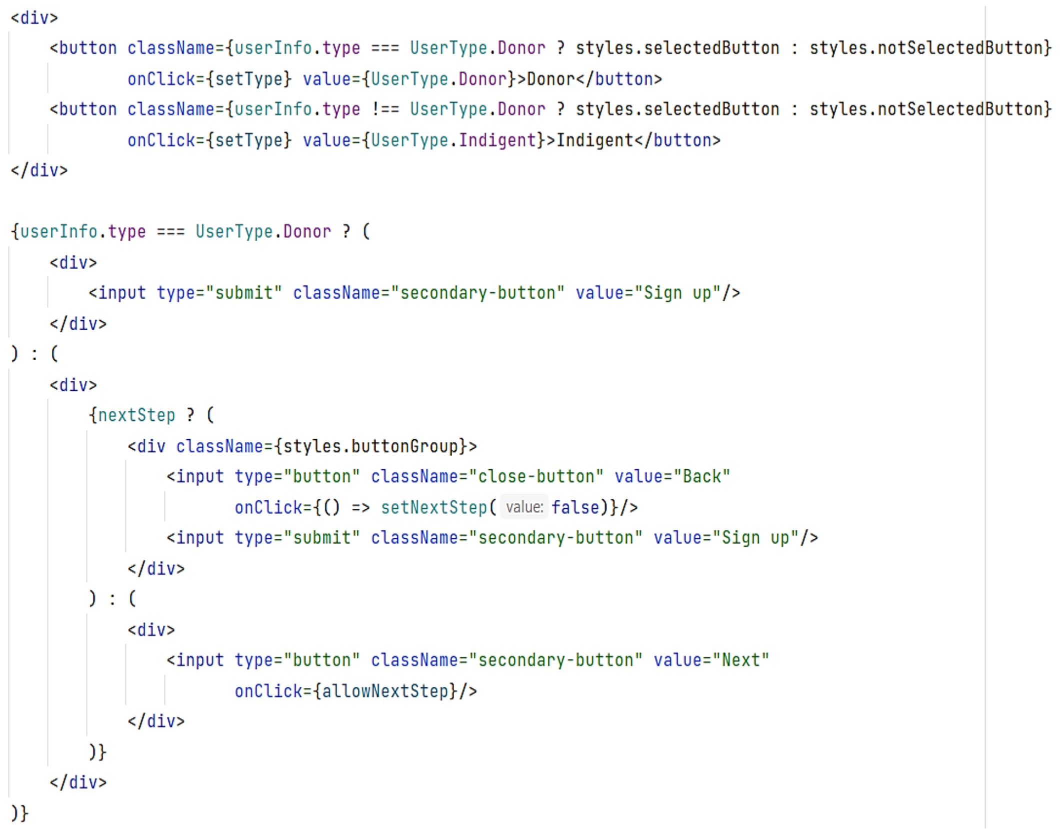
```
) : (
    <div className={styles.userForm}>
        <h2>Fill the form</h2>
        {formError && <p className="error-message">{formError}</p>}
        {message && <p className="error-message">{message}</p>}
        <label htmlFor="about">About me</label>
        <textarea id="about" className="input" name="about" value={userInfo.about}
                  onChange={onChangeUserInfo}/>

        <label htmlFor="history">Medical history</label>
        <textarea id="history" className="input" name="history" value={form.history}
                  onChange={onChangeForm}/>

        <div className={styles.inline}>
            <div className={styles.block}>
                <label htmlFor="amount">Required amount</label>
                <input id="amount" type="number" className="input" name="requiredAmountOfMoney"
                       value={form.requiredAmountOfMoney} onChange={onChangeForm}/>
            </div>

            <div className={styles.block}>
                <label htmlFor="amount">Currency</label>
                <CurrencySelector onChange={(newValue) => setCurrency(newValue)}/>
            </div>
        </div>
```

Figure B.2: RegistrationPage.js code fragment №2



```
<div className={styles.inline}>
    <div className={styles.block}>
        <label>Bank name</label>
        <input type="text" className="input" name="bankName" value={bankAcc.bankName}
               onChange={onChangeBank}/>
    </div>

    <div className={styles.block}>
        <label>Account number</label>
        <input type="text" className="input" name="accountNumber"
               value={bankAcc.accountNumber} onChange={onChangeBank}/>
    </div>
</div>

<div className={styles.inline}>
    <div className={styles.block}>
        <label htmlFor="upload">Document(s)</label>
        <DocumentUploader files={files}
                          error={(text) => setFormError(text)}
                          onUpload={(array) => uploadDocument(array)}
                          onDelete={(index) => deleteDocument(index)}/>
    </div>

    <div className={styles.block}>
        <label>Date of birth</label>
        <CustomDatePicker onChange={(newValue) => setDateOfBirth(newValue)}/>
    </div>
</div>
```

Figure B.3: RegistrationPage.js code fragment №3



```
<div>
    <button className={userInfo.type === UserType.Donor ? styles.selectedButton : styles.notSelectedButton}
            onClick={setType} value={UserType.Donor}>Donor</button>
    <button className={userInfo.type !== UserType.Donor ? styles.selectedButton : styles.notSelectedButton}
            onClick={setType} value={UserType.Indigent}>Indigent</button>
</div>

{userInfo.type === UserType.Donor ? (
    <div>
        <input type="submit" className="secondary-button" value="Sign up"/>
    </div>
) : (
    <div>
        {nextStep ? (
            <div className={styles.buttonGroup}>
                <input type="button" className="close-button" value="Back"
                       onClick={() => setNextStep( value: false)}/>
                <input type="submit" className="secondary-button" value="Sign up"/>
            </div>
        ) : (
            <div>
                <input type="button" className="secondary-button" value="Next"
                       onClick={allowNextStep}/>
            </div>
        )}
    </div>
)}
```

Figure B.4: RegistrationPage.js code fragment №4

Appendix **C**

Final GUI

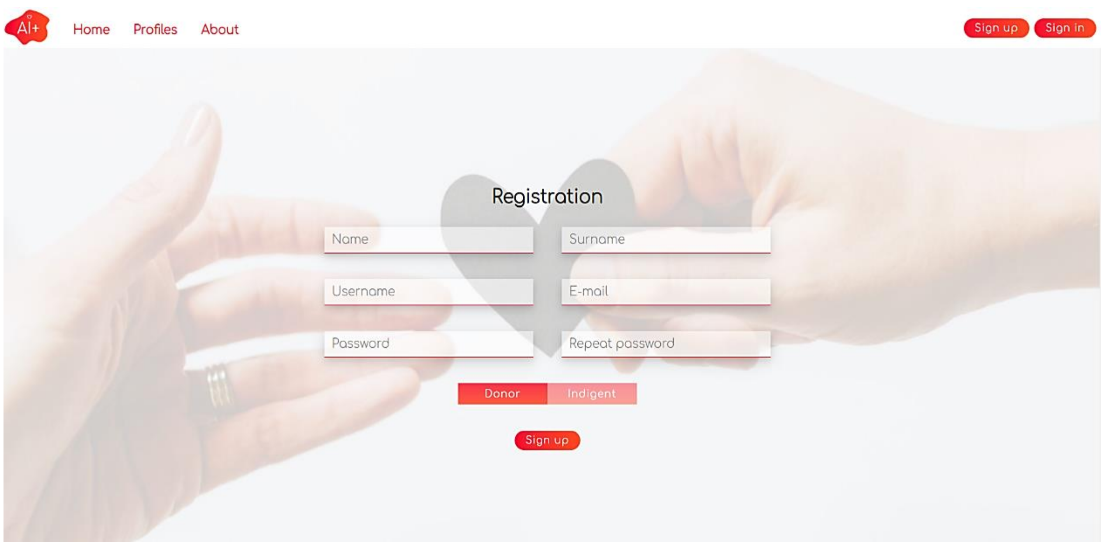
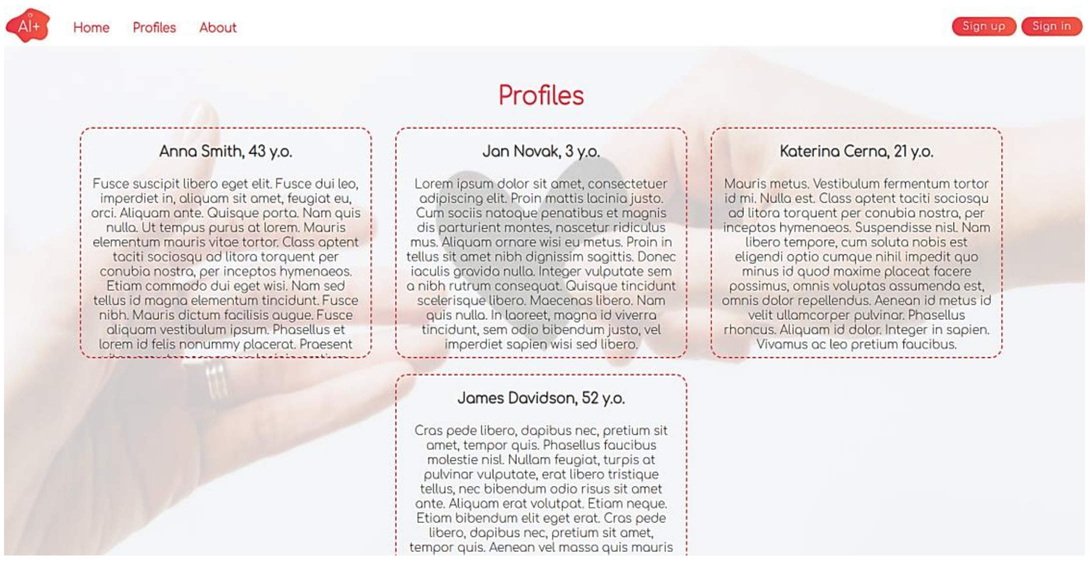


Figure C.1: Registration page



Profiles

Anna Smith, 43 y.o.

Fusce suscipit libero eget elit. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Aliquam ante. Quisque porta. Nam quis nulla. Ut tempus purus at lorem. Mauris elementum mauris vitae tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam commodo dui eget wisi. Nam sed tellus id magna elementum tincidunt. Fusce nibh. Mauris dictum facilisis augue. Fusce aliquam vestibulum ipsum. Phasellus et lorem id felis nonummy placerat. Praesent

Jan Novak, 3 y.o.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin mattis lacinia justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam ornare wisi eu metus. Proin in tellus sit amet nibh dignissim sagittis. Donec iaculis gravida nulla. Integer vulputate sem a nibh rutrum consequat. Quisque tincidunt scelerisque libero. Maecenas libero. Nam quis nulla. In laoreet, magna id viverra tincidunt, sem odio bibendum justo, vel imperdiet sapien wisi sed libero.

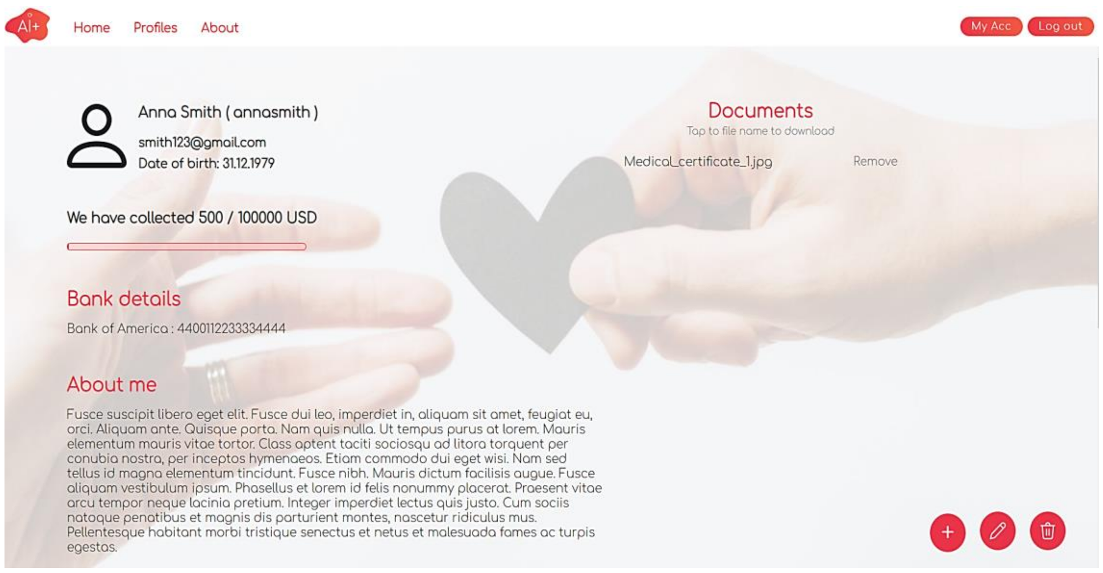
Katerina Cerna, 21 y.o.

Mauris metus. Vestibulum fermentum tortor id mi. Nulla est. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Suspendisse nisl. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Aenean id metus id velit ullamcorper pulvinar. Phasellus rhoncus. Aliquam id dolor. Integer in sapien. Vivamus ac leo pretium faucibus.

James Davidson, 52 y.o.

Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Phasellus faucibus molestie nisl. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Aliquam erat volutpat. Etiam neque. Etiam bibendum elit eget erat. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Aenean vel massa quis mauris

Figure C.2: Profiles page



Anna Smith (annasmith)

smith123@gmail.com

Date of birth: 31.12.1979

We have collected 500 / 100000 USD

Bank details

Bank of America : 4400112233334444

About me

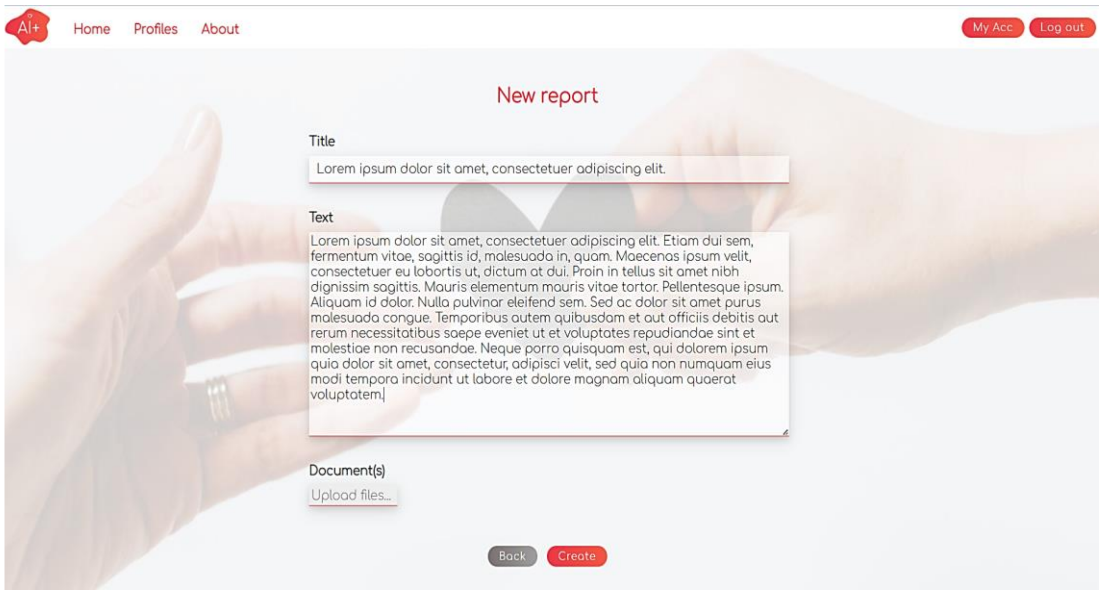
Fusce suscipit libero eget elit. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Aliquam ante. Quisque porta. Nam quis nulla. Ut tempus purus at lorem. Mauris elementum mauris vitae tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam commodo dui eget wisi. Nam sed tellus id magna elementum tincidunt. Fusce nibh. Mauris dictum facilisis augue. Fusce aliquam vestibulum ipsum. Phasellus et lorem id felis nonummy placerat. Praesent vitae arcu tempor neque lacinia pretium. Integer imperdiet lectus quis justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Documents

Tap to file name to download

Medical_certificate_1.jpg Remove

Figure C.3: Indigent’s profile page



New report

Title

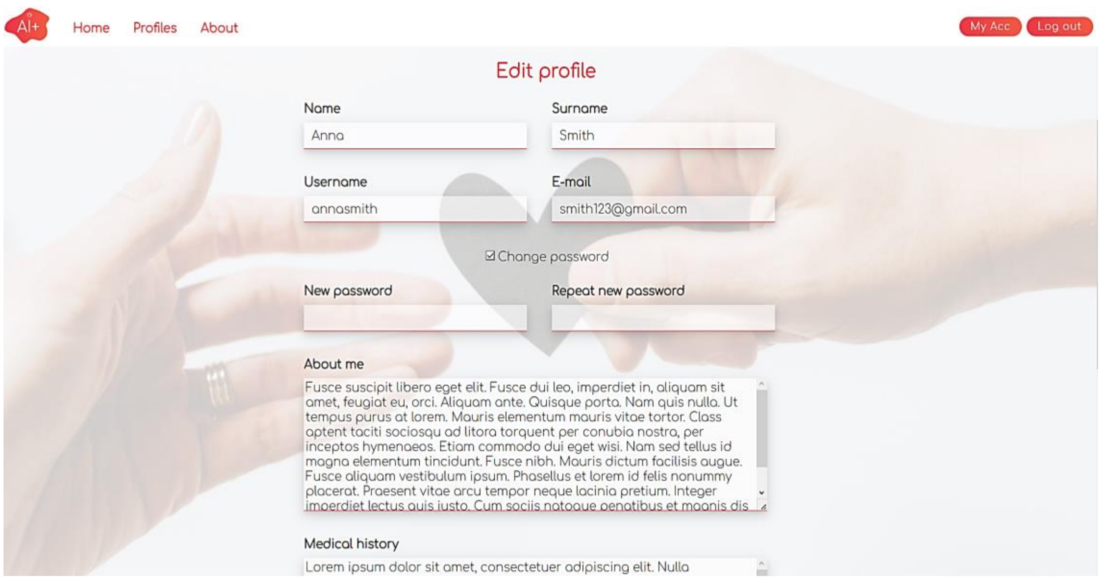
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Text

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Maecenas ipsum velit, consectetuer eu lobortis ut, dictum at dui. Proin in tellus sit amet nibh dignissim sagittis. Mauris elementum mauris vitae tortor. Pellentesque ipsum. Aliquam id dolor. Nulla pulvinar eleifend sem. Sed ac dolor sit amet purus malesuada congue. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.

Document(s)

Figure C.4: Page for creating a new report



Edit profile

Name: Anna

Surname: Smith

Username: annasmith

E-mail: smith123@gmail.com

☑ Change password

New password

Repeat new password

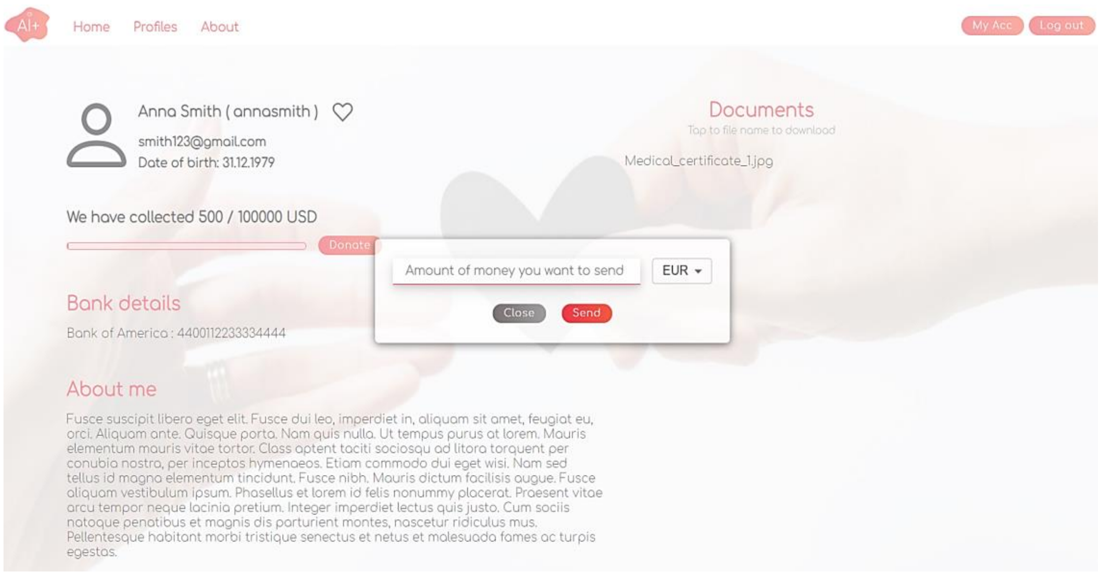
About me

Fusce suscipit libero eget elit. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Aliquam ante. Quisque porta. Nam quis nulla. Ut tempus purus at lorem. Mauris elementum mauris vitae tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam commodo dui eget wisi. Nam sed tellus id magna elementum tincidunt. Fusce nibh. Mauris dictum facilisis augue. Fusce aliquam vestibulum ipsum. Phasellus et lorem id felis nonummy placerat. Praesent vitae arcu tempor neque lacinia pretium. Integer imperdiet lectus quis iusto. Cum sociis natoque penatibus et magnis dis

Medical history

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla

Figure C.5: Page for editing user data



Anna Smith (annasmith)

smith123@gmail.com

Date of birth: 31.12.1979

Documents

Tap to file name to download

Medical_certificate_1.jpg

We have collected 500 / 100000 USD

Amount of money you want to send

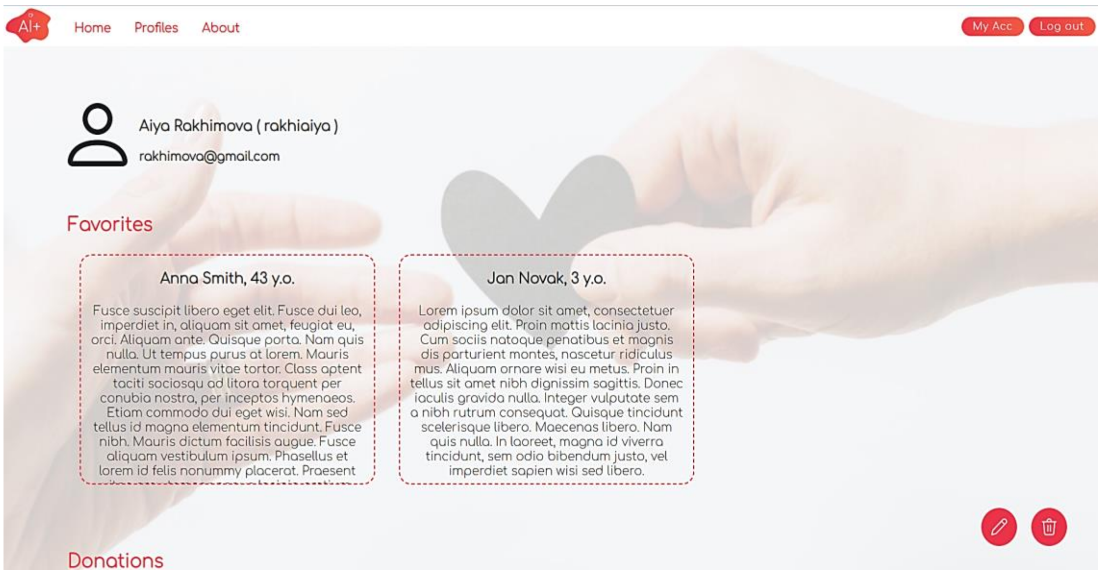
Bank details

Bank of America : 4400112233334444

About me

Fusce suscipit libero eget elit. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Aliquam ante. Quisque porta. Nam quis nulla. Ut tempus purus at lorem. Mauris elementum mauris vitae tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam commodo dui eget wisi. Nam sed tellus id magna elementum tincidunt. Fusce nibh. Mauris dictum facilisis augue. Fusce aliquam vestibulum ipsum. Phasellus et lorem id felis nonummy placerat. Praesent vitae arcu tempor neque lacinia pretium. Integer imperdiet lectus quis justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Figure C.6: Donate operation window



Aiya Rakhimova (rakhiaiya)

rakhimova@gmail.com

Favorites

Anna Smith, 43 y.o.

Fusce suscipit libero eget elit. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Aliquam ante. Quisque porta. Nam quis nulla. Ut tempus purus at lorem. Mauris elementum mauris vitae tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam commodo dui eget wisi. Nam sed tellus id magna elementum tincidunt. Fusce nibh. Mauris dictum facilisis augue. Fusce aliquam vestibulum ipsum. Phasellus et lorem id felis nonummy placerat. Praesent

Jan Novak, 3 y.o.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin mattis lacinia justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam ornare wisi eu metus. Proin in tellus sit amet nibh dignissim sagittis. Donec iaculis gravida nulla. Integer vulputate sem a nibh rutrum consequat. Quisque tincidunt scelerisque libero. Maecenas libero. Nam quis nulla. In laoreet, magna id viverra tincidunt, sem odio bibendum justo, vel imperdiet sapien wisi sed libero.

Donations

Figure C.7: Donor profile page

Appendix **D**

Executing application

Configure setting

Create local PostgreSQL database.

In *src/main/resources/application.properties* change setting to needed database.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/aiplus
spring.datasource.username=postgres
spring.datasource.password=admin
```

Run application

To start back-end run *Application.java* in IntelliJ IDEA

Location: *src/main/java/Application.java*

To start front-end open a terminal and enter following commands:

```
cd src/frontend
npm install
```

After the installation is complete enter

```
npm start
```

After starting up the application, components should be available:

- Front-end – htttp://localhost:3000
- Back-end – htttp://localhost:8081