Bilkent University
Department of Computer Engineering

CS319 Term Project

Section 3
Group 3D
Erasmus++

Final Report

Group Members

-Yunus Eren Türkeri (22001842)
-Ender Utlu (22001983)
-Ege Ayan (22002478)
-Alp Ertan (22003912)
-Mehmet Feyyaz Küçük (22003550)

Table of Contents

1.0 Introduction

Speaking in terms of implementation, although we have a functioning web-application for Erasmus Mobility Program, we are disappointed to say that our application does not satisfy some of the features promised earlier in the semester. Due to the false decisions and actions taken throughout the semester (see the bullet points in **2.0 Lessons Learned**), we couldn't achieve some important features in time. The implementation includes coordinator as a general actor (including course coordinator, departmental coordinator) and lacks the actors for Faculty Administration Board, Bilkent Instructors and International Students Office.

In addition, the system lacks Excel form processing features that are needed for candidate placements and nomination process. So, the system does not handle placements and nomination process internally, rather, the former format of e-mailing of candidates and coordinators is required (apart from the application, the candidates and students should conduct placement process through mailing).

Also, the system is not functional for the course transfer process (which occurs after the mobility program ends).

Though the system lacks such important features, the system is functional for the following use cases (bold cases are the actual buttons and navigation in the system):

- Applying and creating an application (for candidate).
- Uploading PDF files for the required documents (for candidates).
- **Roadmap** for candidates to follow up their application progress.
- Announcements system between candidates and coordinators to reduce mail exchange.
- **To Do List** for coordinators to view candidates' application stages (stages of progress).
- Approval or disapproval of the candidate submissions (i.e., application form) through **To Do List**.
- **Profile** page with editable information; profile photo, personal email, password, mobile phone number and about me (candidate bio).
- Register candidates through **Register Candidates** (for admins).
- **Student List** to view the students registered into the system (for admins and coordinators).
- Docx files for **Learning Agreement** template and **Pre-approval Form** for candidates to fill in.

2.0 Lessons Learned

Overall, we are not so satisfied with our end product and its functionality. However, we are happy with what we have learned during the development process.

- We could have adopted all the details and features of the application domain in a better and more efficient way, so that we could understand the domain better. This would result in a better requirements elicitation stage that would enhance the analysis and design reports in terms of content. The lesson learned here is that as engineers, we need to ask questions to clients over and over again, so that we can avoid making hidden assumptions during the design stage.
- We needed to do better research and work on the implementation decisions, for example, frameworks or the languages to be used should be decided earlier than we did so that we could utilize mandatory tools, programming languages, frameworks and technologies better. As we didn't decide on "what and how to use" earlier, we had struggles about time allocation for the implementation due to other lectures' workload. This resulted in an unwanted way so that we couldn't implement most of the functionalities which we aimed to put into the system.
- We learned that work distribution is not that simple and straightforward for an extensive project. We divided our team into two groups, 3 people for frontend and 2 people for backend. The hardest part of this division was connecting the backend to frontend. The main issue was the dynamics of the design, such that there was a flow of design change regularly. As we noticed that the design reports had missing features in them, we needed to add new features into both backend and frontend. However, the inadequate and asynchronous communication between backend and frontend slowed us down. For instance, the frontend team needed to add new properties and attributes to the java classes, but these new features should have correspondence in the backend too. So this made it harder for us to work independently and we needed to have strong coordination. This is of course not a bad symptom, but if we did better on the analysis and design report, both backend and frontend teams could build their parts compatible with each other. The lesson here is that those UML diagrams are actually very crucial for every subsystem in the implementation, that would enhance the quality of team-work.

Such experiences taught us that we need to put most of our efforts into these few but important points in order to have a more efficient and qualified product.

3.0 User's Guide

3.1 Admin

Admins are responsible for system maintenance and student registration into the system. Note that the coordinator accounts will not be created through the system, rather, they will be created separately at the database, with each of them having their specific ID's. The admins will then register each coordinator manually with their bilkent ID's, bilkent email addresses, and generated passwords.

- In the **Home** page, through the **Register Candidate** button in the side navigation bar, admins can register each candidate into the system.
- Admins are responsible for maintenance of questions asked in the **FAQ** section in the top navigation bar. Any user can ask their questions through the link provided for the **FAQ** section in the top navigation bar. Then, admins with the email address erasmuspp49@gmail.com can answer the user questions and update the **FAQ** page accordingly.

3.2 Coordinator

Coordinator is the user that can approve or disapprove candidate application stages (the forms which the students submit into the system), and they can easily find/view students in the form of a table. Each row of the table contains the information for a student, i.e., student email, so that the coordinator can easily find their specific bilkent e-mail address. They can also post announcements in the main menu, which can be seen from all any user.

- Coordinators can view the students list in the **Home** page>**Students List**. They can search for any student name via CTRL+F. This removes the struggle for the coordinators to find the bilkent e-mail addresses of students by scrolling for many lines. Rather, they can easily reach their information in a table.
- Coordinators can approve or disapprove the application stages (application forms, course registration & equivalence forms, learning agreement forms, and pre-approval forms). Also, they can view the applications in a bigger pop-up. There is also a filtering feature for the listed applications, so that a coordinator can decide which types of forms to be listed in their **To Do List** screen, i.e. they can filter the course equivalency request forms so that they view only them in a list.
- Coordinators can post new announcements through the **Add New Announcement+** button in the **Home** page. When they post a new announcement, the section **Announcements** is updated dynamically in the main screen view. Users can also expand the announcements to view them in a bigger pop up.
- Coordinators can ask questions about the system through the **FAQ** section in the top navigation bar.

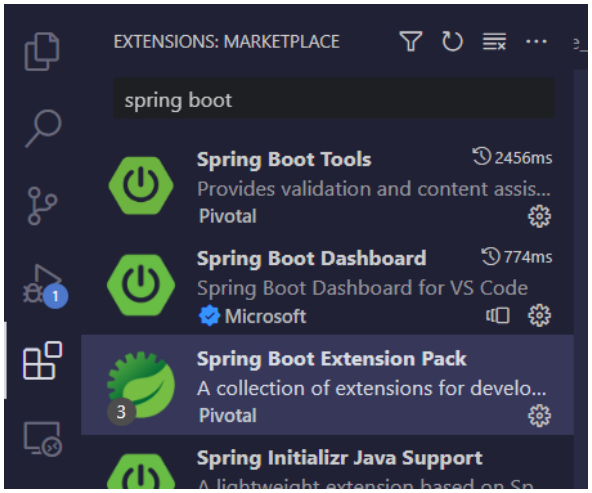
3.3 Students (Candidates)

Students can view and update their **Profile** information, can apply for the Erasmus mobility program through **My Applications**, and they can view announcements posted in the **Home** page.

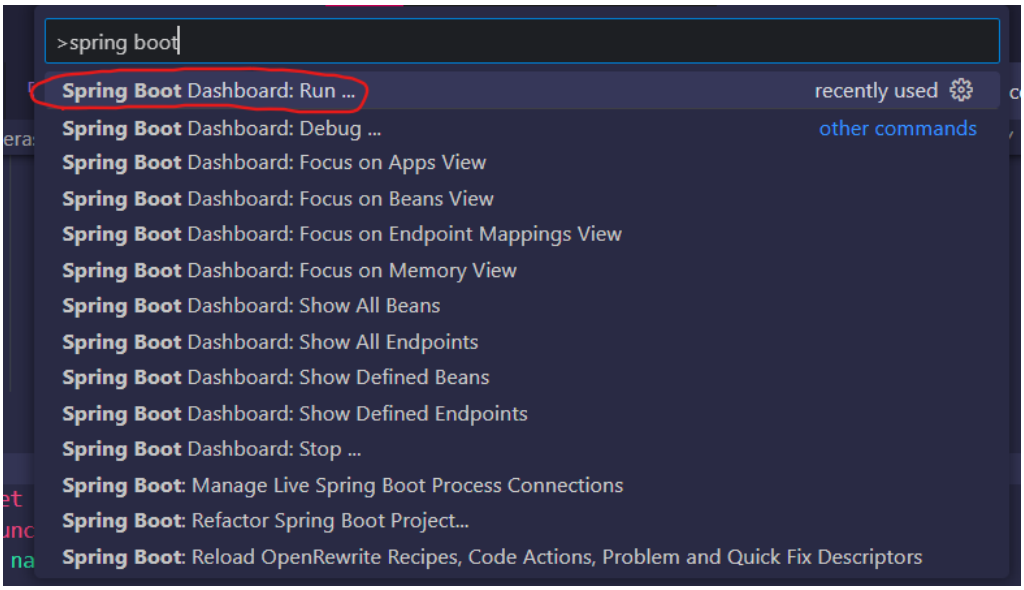
- Students can change their personal email, mobile phone number and about me section in the **Profile** screen. They can also change their passwords through this screen (**Profile**), via the button **Change Password.** This button navigates the users (students) to the **Change Password** screen. From that screen, they can change their passwords.
- Students can create a new application from **Home>My Applications**. When they press the **Apply** button, they can add a new application into their list of **My Applications**. By clicking **Go to Application**, they can navigate to their current stage of application (i.e., the very first stage where the student needs to upload a Statement of Purpose, a CV, and an Application Form). From each application stage, they can navigate to **Roadmap** to view the progress of their applications.
- Students can also expand the cards in the **Announcements** to view them in a bigger pop up.
- Students can ask questions about the system through the **FAQ** section in the top navigation bar.

4.0 Build Instructions

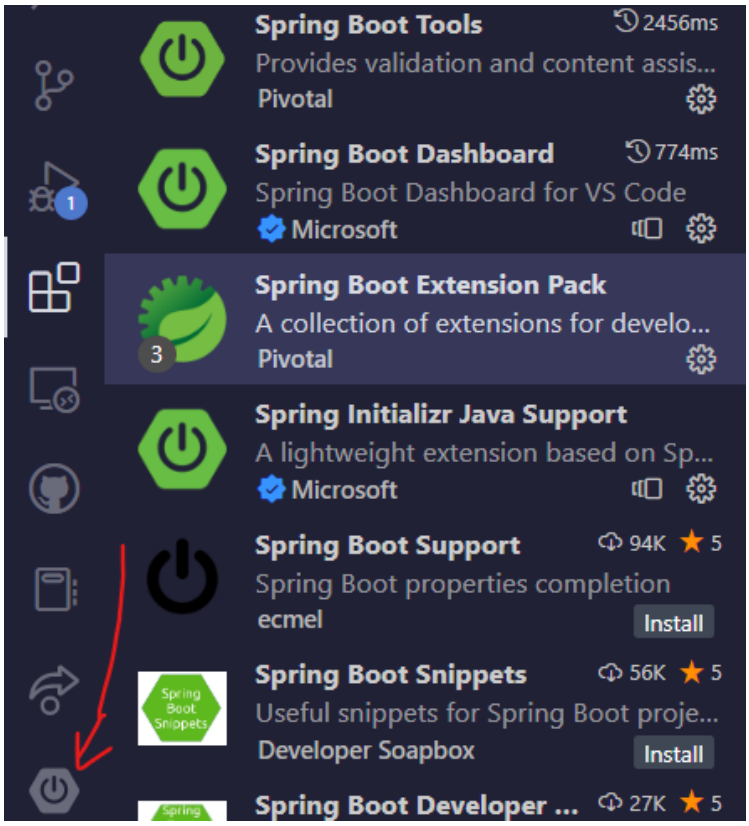
1. Download Node.js to your machine (if it's not downloaded already) from https://nodejs.org/en/download/.
2. Clone our repository from https://github.com/mfkucuk/CS-319.
3. Open the project in VSCode (we downloaded the extensions for Spring Boot in VSCode, so this instruction manual explains building Spring Boot in VSCode specifically).
4. Download "Spring Boot Extension Pack" from Extensions in VSCode.



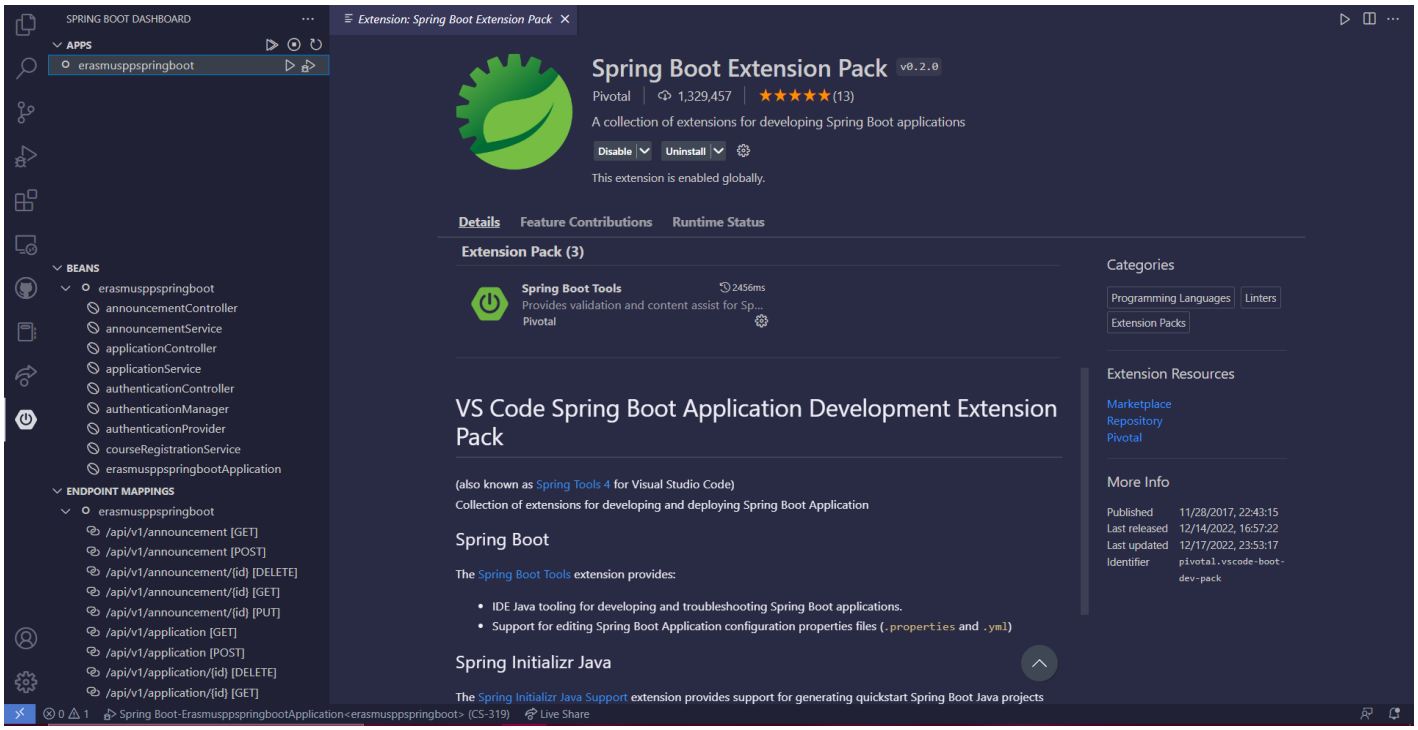
5. When Spring Boot Extension Pack is downloaded, use the command ctrl+shift+p (or directly click on the top search bar and type ">") and type ">Spring Boot Dashboard: Run…"



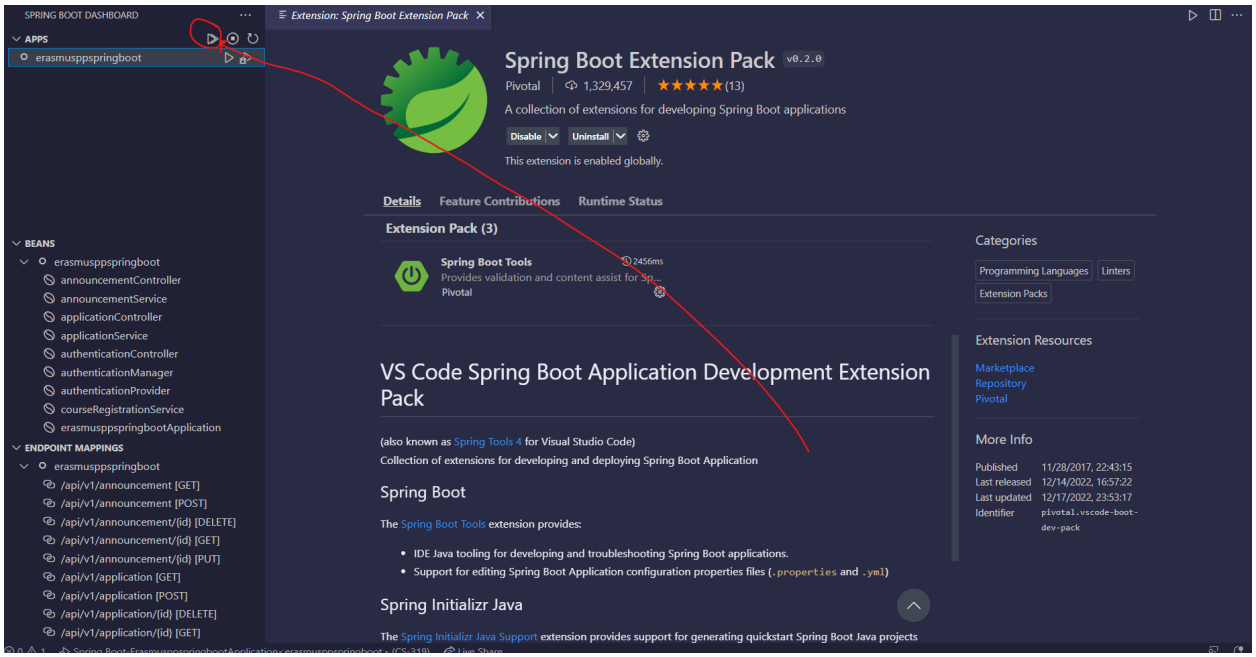
6. When Spring Boot Dashboard: Run… is called, an "on/off" button will appear on the left dashboard.



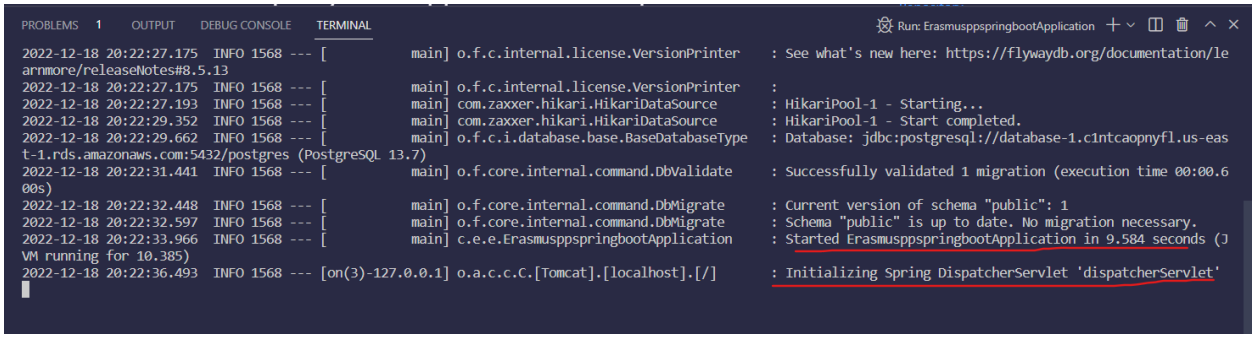
7. Click that on and off button and the screen will turn into the one below.



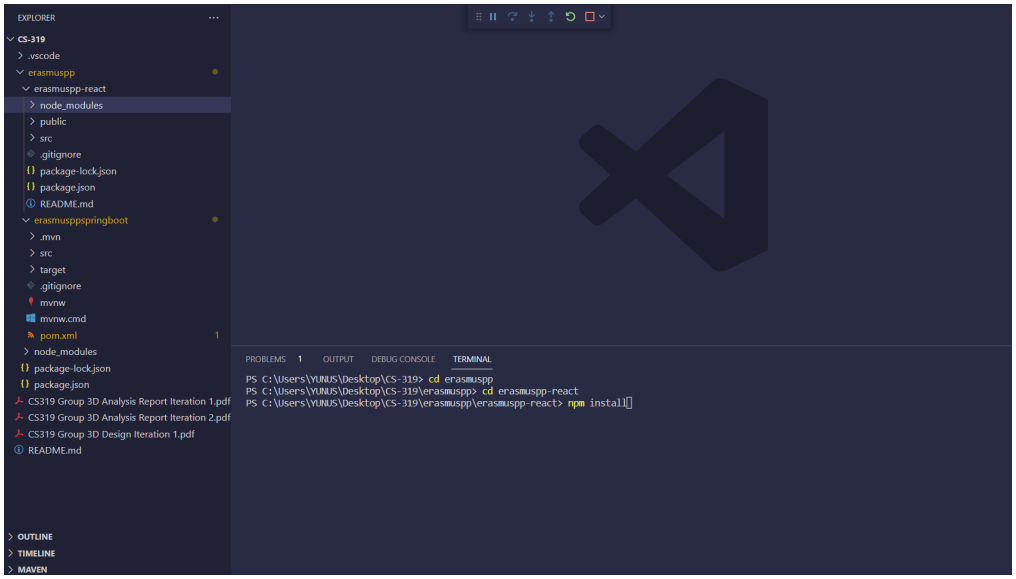
8. After having the dashboard in step 7, click the run button on the top left.



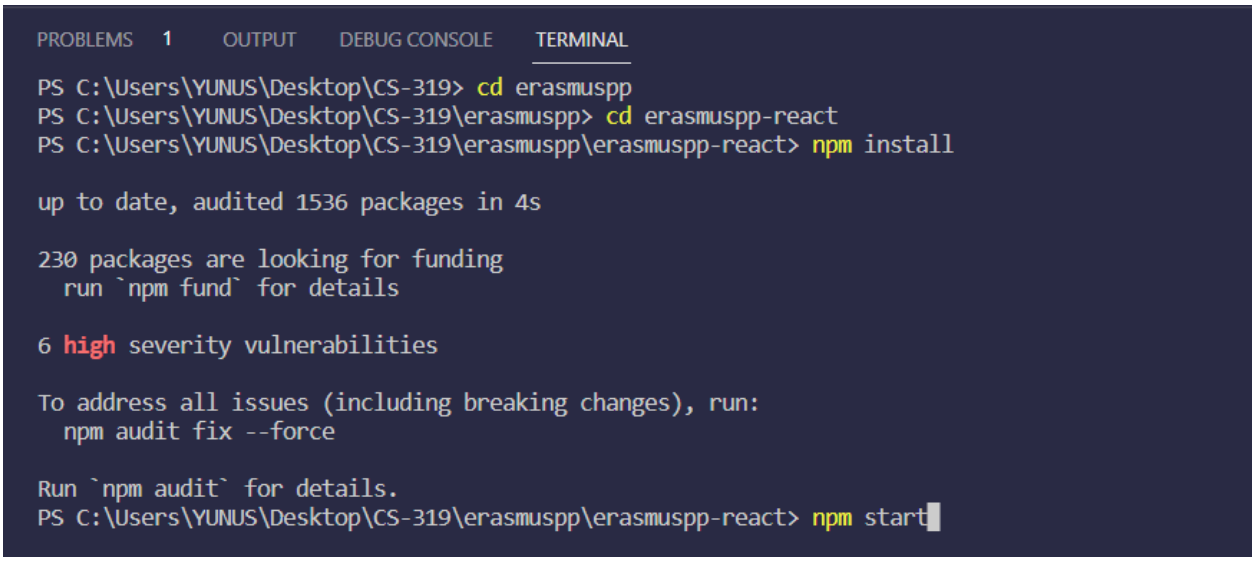
9. When the run button is clicked, if you see the following terminal, then it means that the backend is successfully connected to the project.



10. Now cd into the “erasmuspp-react” directory and type in “npm install”. This will only work if the first step is already done (downloading Node.js).



11. Now the terminal should turn into the following. Type the command “npm start”.



12. After the “npm start” command is called, the http://localhost:3000/ will run in a browser automatically.
13. Now the website can be used.

5.0 Work Allocation of Team Throughout the Semester:

Yunus Eren Türkeri:

- Contributions to Analysis Report:
    - Non-functional Requirements
    - Use Case Diagram
    - Statechart diagrams
    - Mockups for the User Interface
- Contributions to Design Report:
    - System Decomposition
    - Final Object Design
    - User Interface Layer
    - Web Server Layer
    - Packages definitions
    - Class Interfaces definitions.
- Contributions to Implementation:
    - User interfaces using React.js
    - Inline css styling of components
    - Connecting FAQ section as an external actor (Notice)
    - Logo Design for Erasmus++Team
    - Api (http) calls via axios

Mehmet Feyyaz Küçük:

- Contributions to Analysis Report:
    - Introduction
    - Functional Requirements
    - Use Case Diagram
    - Class Diagram
    - Mockups for the User Interface
- Contributions to Design Report:
    - Introduction
    - Hardware/Software Mapping
    - Data Management Layer
    - Package Definitions
    - Class Interface Definitions
- Contributions to Implementation:
    - Spring Boot Deployment
    - Login using JWT
    - Connected a postgresql database to backend using amazon web services
    - Implementation of User.java class
    - Implementation of Announcement.java class
    - Implementation of Authentication.java class

    - Implementation of Application.java class

Ege Ayan:

- Contributions to Analysis Report:
    - Non-functional Requirements
    - Use Case Diagram
    - Sequence Diagram
    - Mockups for the User Interface
- Contributions to Design Report:
    - Hardware/Software Mapping (specifically Boundary Conditions)
    - Object Design Trade-offs
    - Final Object Design
    - User Interface Layer
    - Web Service Layer
    - Package Definitions
    - Class Interface Definitions
- Contributions to Implementation:
    - Installing Routing/Props infrastructure
    - Token Cookies
    - Api (http) calls via axios
    - File Base64 decode/encode
    - File download from local/remote source
    - Processing image files
    - Handling dynamic data using useEffect, useState

Ender Utlu:

- Contributions to Analysis Report:
    - Introduction
    - Use Case Diagram
    - Class Diagram
    - Mockups for the User Interface
- Contributions to Design Report:
    - Introduction
    - Hardware/Software Mapping
    - Data Management Layer
    - Package Definitions
    - Class Interface Definitions
- Contributions to Implementation:
    - User interfaces using React.js
    - Inline css styling of components
    - Api (http) calls via axios
    - Handling dynamic data using useEffect, useState

    - Defining route paths

Alp Ertan:

- Contributions to Analysis Report:
    - Introduction
    - Use Case Diagram
    - Activity Diagram
    - Mockups for the User Interface
- Contributions to Design Report:
    - Introduction
    - Design Criteria
    - Data Management Layer
    - Package Definitions
    - Class Interface Definitions
- Contributions to Implementation:
    - Spring Boot Deployment
    - Connected a postgresql database to backend using amazon web services
    - Implementation of File.java class
    - Implementation of Announcement.java class
    - Implementation of Application.java class