BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS 299

SUMMER TRAINING REPORT

Elif Kurtay

21803373

Performed at

TÜBİTAK BİLGEM Software Technologies Research Institute (YTE)

06.07.2020 – 07.08.2020

Table of Contents

1 Introduction ... 3
2 Company Information ... 4
2.1 About The Company ... 4
2.2 About Your Department ... 5
2.3 About The Hardware And Software Systems ... 5
2.4 About Your Supervisor ... 6
3 Work Done ... 6
3.1 General Information On The Internship Project ... 6
3.2 User Interface and Frontend Development ... 9
3.2.1 Project Organization and Structure ... 9
3.2.2 Programming ... 11
3.2.3 User Interface ... 12
3.3 Backend Development ... 14
3.3.1 Initialization ... 15
3.3.2 Entity, DTO, and Mapper ... 15
3.3.3 Repository, Service, and Controller ... 17
3.3.4 Security and Authentication ... 19
3.4 Database Development ... 20
4 Performance and Outcomes ... 21
4.1 Solving Complex Engineering Problems ... 21
4.2 Ethical and Professional Responsibilities ... 22
4.3 Making Informed Judgments ... 23
4.4 Acquiring New Knowledge Using Appropriate Learning Strategies ... 23
4.5 Applying New Knowledge When Needed ... 24
5 Conclusions ... 25
References ... 26
Appendices ... 27

1 Introduction

This report details a summer internship at TÜBİTAK BİLGEM Software Technologies Research Institute (YTE), Ankara. The internship was performed online. I worked in the software development department of YTE. The company aims conducting research and development (R&D), developing technology and products, providing test and analysis services, and finding original solutions in national strategic and critical areas [1]. The department I worked at mainly focuses on developing software and researching related new technologies. The reason I chose this company is its broad range of software technologies and support for research and development. I enjoy learning and researching new software technologies. Therefore, I thought this company would be the right place for me to experience working.

I had an overall project of developing a web application. The project included developing database, backend, and frontend of this web application individually. The goal was to be involved in each process and aspect of building a software product. I studied and practiced creating a database. I learned a lot from backend development and spent many hours debugging there. I got familiar with frontend technologies for a web application thanks to the overall project. In addition, I have worked on learning and researching new technologies during the development of the project which was my goal for the internship.

The company provided interns with online classes to teach different technologies that we would use in our overall project. They also had online conferences on recent technologies and institutional information which created a familiarity in both conducting R&D and corporate life. All interns had the chance to communicate with all supervisors. Therefore, I get to gain information from supervisors in different branches as well.

The report will have detailed sections for introducing the company, the work I have done during internship, my performance in the company and my outcomes from the internship. The following section titled Company Information will provide YTE's brief history, the department I worked with and their projects. In the same section, my supervisor's information will also be given. Following that in the Work Done section, a detailed information of my overall internship project can be found. I divided this section into subsection to ease understand of my multi-layered project. Performance and Outcomes section has my personal thoughts on my general experience in working in YTE and what I learnt from there. I will then end the report with a brief conclusion.

2 Company Information

2.1 About The Company

TÜBİTAK
BİLGEM
YTE
YAZILIM TEKNOLOJİLERİ ARAŞTIRMA ENSTİTÜSÜ
SOFTWARE TECHNOLOGIES RESEARCH INSTITUTE

After operating as Software and Data Engineering Department under TÜBİTAK Department of National Research Institute of Electronics and Cryptology Design and Development Engineering (UEKAE), that department has been transformed into this institute in 2012. Software Technologies Research Institute (YTE) is affiliated with Informatics and Information Security Research Center (TÜBİTAK-BİLGEM). [1] Hence, the full name of the institute: TÜBİTAK BİLGEM YTE.

CMMIDEV/5 THE Open GROUP [2]

TÜBİTAK BİLGEM YTE is currently the one and only government institution owning a High Maturity Level (level 5) accreditation in Capability Maturity Model Integration (CMMI) modal, in Turkey. [1] Providing support for their mature corporate environment and organization. In addition, they are a member of the consortium of the Open Group. The Open Group is a global consortium with over 700 member organizations that achieves business objectives through technology standards by ensuring openness, interoperability and consensus between customers and suppliers. [2] This membership shows YTE's level of professionality and universality in technology business.

Vision and goals of YTE focuses on contributing to the development of informatics ecosystem to apply successful digital transformation policies and thus becoming a pioneer research institute. They are working towards their vision by conducting R&D software development projects in digital national policies, providing guidance, developing reference models, building capacities in new technological areas, and defining solutions with an innovative approach for digital transformation. [1] As a result of their multiple aims, YTE has four major departments: Digital Policy Development, Digital Transformation Planning, Software Development, and Capacity Building.

2.2 About Your Department

I did my internship in Software Development department in TÜBİTAK BİLGEM YTE. Software Development department priorities developing strategic, critical solutions by doing R&D for realizing key actions in the digital government ecosystem. The department follows High Maturity Level and Open Source Coded Technology Usage policy while providing its services. [1] Projects of the department can be listed as follows:

- Government Accounting Information System: enables a centralized management of the accounting transactions of all public institutes and agencies through a digital document-based information system. Completed in November 2019.

- Product Tracking System (ÜTS): a national product tracking and monitoring model for tracking medical devices and cosmetic products. Completed in December 2018.

- Integrated Social Assistance Service: an information system for centralized management for allocation of resources and activation of social assistance decisions, in assistance within the body of Minister of Labor, Social Services and Family.

- Open Source Coded Database Development: aimed at developing open-source database system (PostgreSQL), tools and expertise in YTE to create a strong alternative for commercial databases.

- Development Agencies Management System (KAYS): An integrated information system on financial management of the Development Agencies within the body of Minister of Industry and Technology.

- Energy Market Regulatory Management System (EBIS): A system ensuring centralized management of the energy market. [1]

2.3 About The Hardware And Software Systems

TÜBİTAK BİLGEM YTE uses a variety of software systems whereas its hardware is limited to computers since it is a department focused on software development.

Teams in the department, have liberty in choosing development environments and related software technologies as well as operating systems. However, they prefer open-source systems such as PostgreSQL and Linux. During my internship I have come across PostgreSQL for database, IntelliJ IDE, Android Studio and Visual Studio for development environments, Spring used in back-end development, and React and Flutter used in front-end and mobile development. I observed that Git and Github is used for team collaboration. Unfortunately, I am not aware of rest of the software systems that are utilized.

2.4 About Your Supervisor

During the internship, interns had a chance to contact any supervisor for help in their respected fields. However, we also had personal supervisors. Information of my personal supervisor can be found below.

Name:	Ahmet Emre Kılınç
Job Title:	Software Architect / Senior Software Developer
Education:	B.S. in Computer Engineering, Bilkent University, 2008
	M.S. in Information Systems, Middle East Technical University, 2014
	M.S. in Law of Information Technology, Hacettepe University, 2018
Email:	emre.kilinc@tubitak.gov.tr

Mr. Kılınç is currently working at TÜBİTAK BİLGEM YTE Software Development department on web-based software projects and leading front-end infrastructure team that makes architectural decisions. He has supervised my process during the internship and assisted me when needed. Luckily for him, I had more trouble in backend development and had to disturb other supervisors more than him. All other supervisors also had education in computer engineering.

3 Work Done

3.1 General Information On The Internship Project

The project of the internship was developing a web application. The goal of the project was for interns to learn the stages of web application development and practicing with different software tools in these stages. Another motivation behind the project was to get familiar with open-source technologies such as PostgreSQL. We also had many workshops and lectures on software technologies and how we should follow and learn from open-sources environments such as Linux. Self-learning had a big significance during the internship and development of the project.

The project consisted of three major stages: database (PostgreSQL), backend (Spring), and frontend (React). Detailed information about each stage can be found in their respective subsections (3.2, 3.3, 3.4).

There was no teamwork or division of labor in the project so that each intern could acquire knowledge on each stage. Supervisors gave advice on how to solve significant problems and monitored our project advancement. They also provided learning material and workshops to speed up the learning process. However, they did not assist or commit to the work done. Therefore, the project is solely my work.

The project was about making a social web application where people can organize and attend activities. The website was targeted to be used in company event organization. There was a guideline for the project which we had to follow. We got evaluated by the work we produced by the end of the internship according to the guidelines. However, we had some freedom to add and/or remove some features. The software tools we were taught and used were mandatory and are given in Table 1.

Stages	Programming Language Used	Software Development Environment Used	Software Tools Used
Frontend	JavaScript, CSS	Visual Studio	React, Node.js
Backend	Java	IntelliJ IDE	Spring Boot
Database	SQL	PostgreSQL	Postman (also related with backend) and pgAdmin

Table 1: Programming languages, software development environments and software tools used in respective development stages

The features to be added to the “Activity Planner” (the name given to the web application project by myself) had three phases. The first phase included features to ensure activity management and administration by a corporate user. A “corporate user” is a user who is organizing an activity (or event). Those users have more administrative actions (edit, delete, see who registered, add questions to be asked to the attendees, see attendees answers to said questions…) on activities whereas an “external user” can only see limited information about the activities and can register an application by submitting a form.

All the features given in these explained phases have been completed and implemented. However, implementations of each feature will not be shared in this report since it would be too much insignificant programming detail.

Phase 1 – Event Administration
1. Corporate user enters new event. (event name, event starting and ending dates) 2. Corporate user lists every event. (event name, event starting and ending dates) 3. Corporate user updates current event. (event name, event starting and ending dates) 4. Corporate user deletes current event. 5. Quotas are assigned to events and participants cannot register when quotas are met.
Rules and additional features: 1. The start date cannot be after the end date. 2. After the start date, the event cannot be updated or deleted. 3. After a successful operation, a success message is displayed. 4. An error message is displayed when an error is received in an operation. 5. The event organizer can put additional questions and fields to be entered on the registration screen, specific to the event.

Table 2: Phase 1 features implemented in the application.

The second phase focuses on activity applications by external users. The third and last phase is about organizing and reporting event information to the corporate user.

Phase 2 – Event Application
1. The external user lists all activities which have not exceeded the start date. (event name, start date, end date) 2. External user applies to the activity.
Rules and additional features: 1. The same user cannot apply multiple times to an event. 2. After an operation has been successfully completed, a success message is displayed. 3. An error message is displayed when an error is received in an operation.

Table 3: Phase 2 features implemented in the application.

Phase 3 – Event Reporting
1. The corporate user lists the applicants for the event. (name surname…) 2. The corporate user displays the activities in bar type charts according to the number of applicants. 3. The corporate user displays the applications for the event in bar type charts by day.

Table 4: Phase 3 features implemented in the application.

Even though it is not indicated in the phases and rules, naturally the application needed to have a login and registration features since there are multiple type of users and users have user-dependent information.

3.2 User Interface and Frontend Development

The frontend is developed in Visual Studio using JavaScript. Frontend stage of this project includes user interface (UI), allocation states of the UI components, and managing data in the database by making HTTP requests. The frontend runs on local server (from localhost:3000 port) in this project. A library named React is used in frontend.

React is a JavaScript library for building user interface. It is rendered on the server using Node.js. It is component based and declarative meaning each component has their own states and when a change occurs React will update only the relevant components rather than updating the whole page. Therefore, it is very effective. In addition, React provides another library named "Axios" which simplifies making HTTP requests from the browser which are crucial to access and manipulate data. Because of its effectiveness and simpleness, React is chosen in this project in frontend development.

To install React in a Visual Studio project Node.js and "npm" (Node.js package manager) needs to be installed. After these installations are complete npm commands to include React should be entered. However, there is another package manager called "yarn". Both can be used to install React. Yarn is created to solve some existing problems of npm. Yarn operates more securely, faster, and more reliable than npm. [4] Therefore, yarn is used in this project. After entered required commands to initialize your React project. The project will start listening to localhost:3000 and rendering the default program.

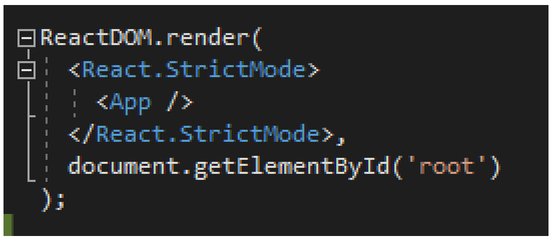
3.2.1 Project Organization and Structure

After React is installed into the project, the project folder has a default organization and includes default files such as ".gitignore", "README.md" and more. By default, JavaScript and CSS files are included in a folder named "src" (source). CSS files (.css) has style information of the components put on the page. JavaScript files (.js) has the code and functions of the components. Therefore, all work is done in this src folder with a small exception of modification in the "package.json" file which will be explained in the subsection 3.4 named "Database" in this report.

The folder hierarchy can be found in Appendix 5.1. Lowercase letters and the box color orange indicates a package or folder. Camel case naming (except for index.js), blue color, and ending ".js" indicates a JavaScript file. Lowercase letters, green color, and ending ".scss" indicates a CSS file. SCSS is an upgraded version of

CSS which has more functionality. Therefore, SCSS files are being used in this project rather than CSS. However, SCSS files also use CSS language which is why they are addressed as CSS files in this report.

Source file includes App.js, index.js and their respective CSS files with a components package. Source file also has additional JavaScript files for testing purposes, but those files are omitted in this report since no change or work is done on these files.



```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Figure 1: the implementation of ReactDOM.render function

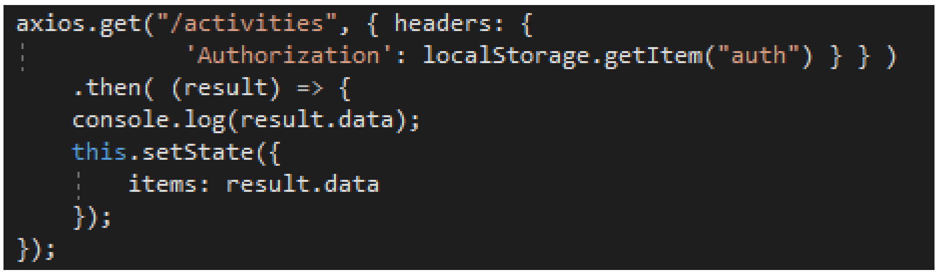
By default, React renders the file named "index.js" because it has the ReactDOM's render function. DOM stands for Document Object Model which is a programming interface for HTML and XML documents. DOM represents the page where programs can alter the document structure, style, and content. [5] "ReactDOM.render" function enables programs to create a website for a user interface. The "StrictMode" of React helps the programmers by giving more warnings and additional checks, it does not affect the build. Currently a component named "App" is rendered. React is used as a single page application, meaning different components inside App will be rendered separately when needed but the whole page is only the "App". Therefore, providing a CSS file for App.js file applies the style rules to all components rendered inside.

App.js has 5 main routes inside: Login, Profile, Homepage, Enrollment, and Activity Details. Most of these pages uses common components like "AppTopBar.js", whereas Login page has completely different components and style. Therefore, log in and registration pages are the only who have a different CSS file. Each route has a defined function inside App.js. These functions include the components used in the relative pages. Through the actions on the components, user can change their routes. For example, by clicking "Profile" button on the Homepage, user can change their route from "/homepage" to "/profile" and thus the Profile page (or components) are rendered in App.js.

To have an understandable design, all components used in App.js are put into a package named "components" (can be found in Appendix 1). Inside the package, components are one again divided according to the page they are used specifically except for common components like "AppTopBar.js". Detailed hierarchy of the packages can be found in Appendix 2 to 4.

3.2.2 Programming

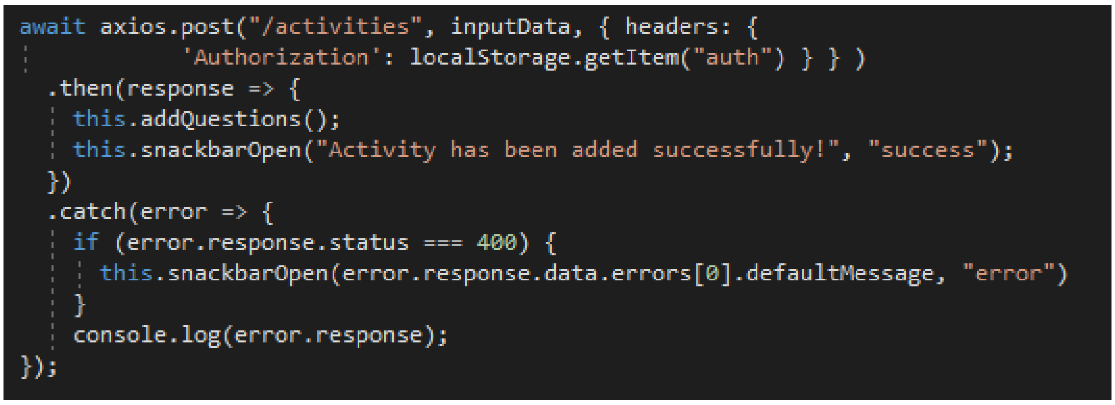
Components have a similar organization. Each component extends "Component" from "react" library. They have their "state" which is an object, a "componentDidMount" function, and a "render" function. State has data and variables that are used in the class. The "componentDidMount" function is called autonomically after component is created and it usually makes an HTTP request to gather required data from database and put into designated fields on the component. These HTTP requests are made by using the "axios" library. An example GET request is provided in Figure 2.



```
axios.get("/activities", { headers: {
              'Authorization': localStorage.getItem("auth") } } )
    .then( (result) => {
    console.log(result.data);
    this.setState({
        items: result.data
    });
});
```

Figure 2: implementation of an axios get function. This code snipet is taken from the JavaScript file where the Data Table in Homepage is filled with activity information.

In the first parameter, the route given is independent of the route of frontend, it is the address for backend. The second parameter sends a header named "Authorization" to the request. This is required because requests are private and secured for the user. After the user logs in, an authentication number is stored in the cache, through local storage, and user's personal information is not used or kept. This way security and privacy is provided to the user. Details of the authentication process is explained in 3.3 subsection. Requests that require input (POST and PUT), have the input data in the second parameter and headers in the third parameter. An example of a POST request is given in Figure 3.



```
await axios.post("/activities", inputData, { headers: {
              'Authorization': localStorage.getItem("auth") } } )
  .then(response => {
    this.addQuestions();
    this.snackbarOpen("Activity has been added successfully!", "success");
  })
  .catch(error => {
    if (error.response.status === 400) {
      this.snackbarOpen(error.response.data.errors[0].defaultMessage, "error")
    }
    console.log(error.response);
});
```

Figure 3: implementation of an axios post function. This code snipet is taken from the JavaScript file where a question is added to an event by the owner in activity details section.

In this figure, snack bars are used to give the user information on whether their action has been successful or not. This provides a more user-friendly interface. Errors are also shared with the user for them to understand the problem. The "await" keyword can be used in asynchronous functions. This keyword makes the program wait for a response from the action performed, indicating that it finished its task, to continue. “Await” with “async” (in front of the function) is generally used in the program with axios functions.

In the render function, the imported components are given their parameters and are organized. Components are defined and used very similarly to HTML writing style. This organization is the return value of the render function. An example render function can be found in Appendix 5.

3.2.3 User Interface

The user interface of this project is aimed to be modern, minimalistic, and user-friendly. There is a general blue and white tone. The Login and Register pages have an animation which allow each to be transformed into the other. Other components only have their default animations such as the opening of the component “Accordion” and bar graphs’ default animation when being rendered.

Dialogs, which pop up, are used to secure user answer or choice of action. They are also used to get a response when there is no input area on the components.

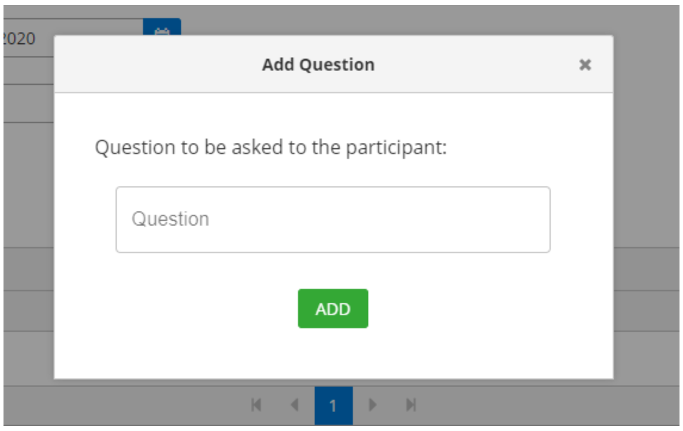
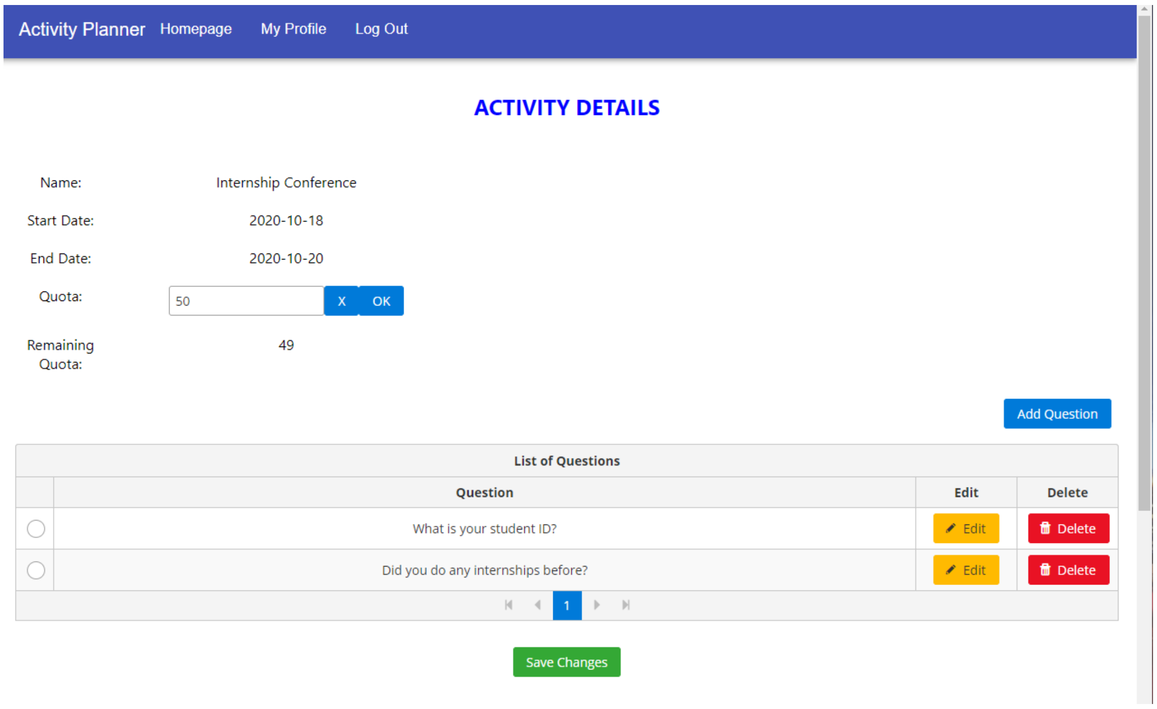


Figure 4: User interface of a “Dialog” compenent which pops up in front off the screen for the user to focus on.

The application pages always have the top bar for easy navigation within the application. Top bar has the name of the website “Activity Planner”. It includes links to the Homepage, My Profile, and Log Out. Homepage has available event that the user can join. My Profile has user-owned activities and their details. The Log Out link deletes the user’s authorization token from cache and sends the user to the Log In page. The following figures will each show a unique feature of the website without giving too much detail.


Activity Planner Homepage My Profile Log Out

ACTIVITY DETAILS

Name:	Internship Conference
Start Date:	2020-10-18
End Date:	2020-10-20
Quota:	50 X OK
Remaining Quota:	49

Add Question

List of Questions

	Question	Edit	Delete
○	What is your student ID?	Edit	Delete
○	Did you do any internships before?	Edit	Delete

Save Changes

Figure 5: User interface of an Activity Details page which can only be accessed by the owner of the activity from edit option in My Profile. All activity information can be modified except for remaining quota. Questions can be added and edited to the activity.

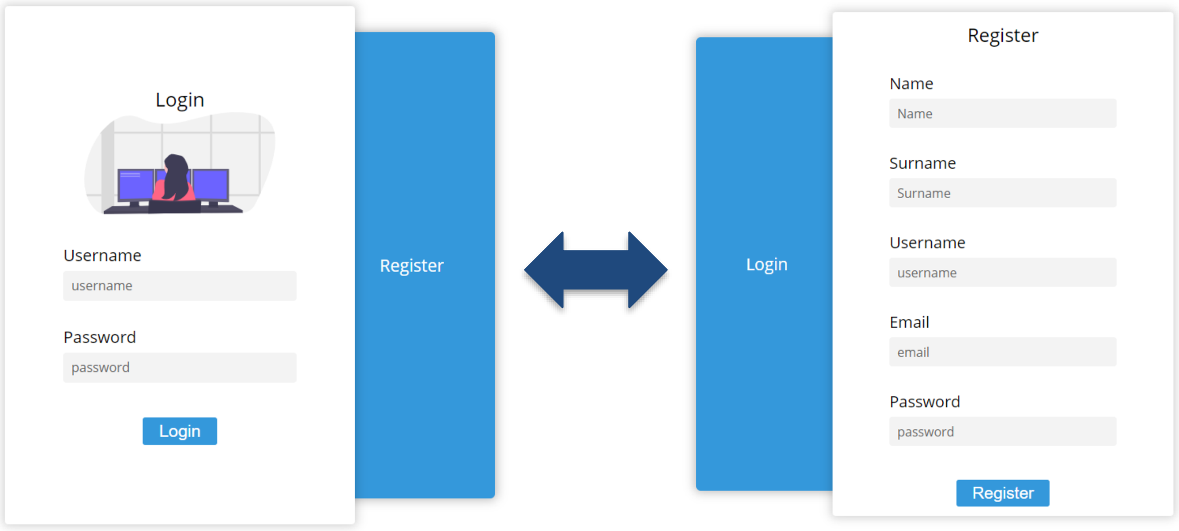
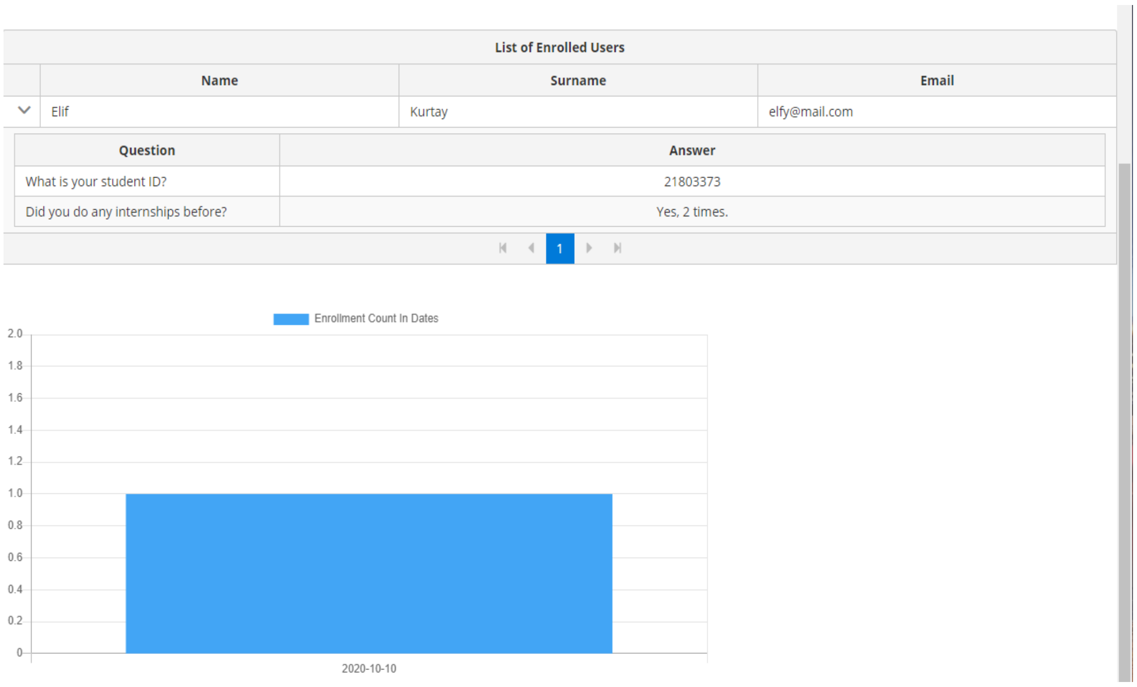


Figure 6: User interface of the Login and Register pages. Pressing the side bars of each page turns to the other one. It is a very modern, pleasant, and user-friendly frontend design. This animation is implemented through having left and right CSS designs which change according to the state of the main form component.



List of Enrolled Users			
	Name	Surname	Email
⌄	Elif	Kurtay	elfy@mail.com

Question	Answer
What is your student ID?	21803373
Did you do any internships before?	Yes, 2 times.

Figure 7: the user interface of a scrolled down version of an Activity Details page. The Figure 5 has the scrolled-up version of the same page. This part of the page has bar graphs and a data table of enrolled user's answers and information. The bar chart shows the number of user's registered to the activity on different days.

3.3 Backend Development

Backend development was the biggest and most challenging part of the project for me. Even though we learnt Java in Bilkent, the functionality and usages of Java required for this part was unknown to me. Therefore, not only did I needed to comprehend backend development, but I also had to study Java. Especially new tools in Java brought by Spring were new to me.

Backend development is where we will organize and manage data and requests in the project. This part is like a bridge between the user and the database. Development of user authentication, entity creation (objects that are created in database), repositories (database functions), and rest controller is completed in this part.

A new software technology called "Spring Boot" is used for the backend development of this project. This tool makes it very easy to create a stand-alone, production-grade Spring based applications. [6] It provides a complementary ecosystem to develop web applications. According to a survey on JetBrains, 61% of developers use Spring Boot for web frameworks. [7] Spring Boot is used with Java and for development environment IntelliJ IDE is chosen in this project. Another tool that is used in this part is the Postman application. Postman allows developers to send HTTP requests to test the rest controller of the application. This tool was frequently used to debug and test the program.

3.3.1 Initialization

Before creating the project in IntelliJ, Spring Boot is initialized with required dependences. Spring has a lot of useful dependences that are easy to add and implement to project. The link to the initializer is given below:

https://start.spring.io/

Through this link, I created a maven project in Java 14 with Spring Boot 2.3.4. Before generating the project, I added the dependences I would need and use. These dependences are: Spring Boot DevTools, Lombok, Spring Web, PostgreSQL Driver and Spring Security. After adding these to the project, I generated the project and opened it with IntelliJ IDE. These is only one additional thing to be added to the dependences and that is “mapstruct”. The dependency code of mapstruct in the “pom.xml” file can be found in Appendix 6. After updating pom.xml file, the initialization of the project is completed.

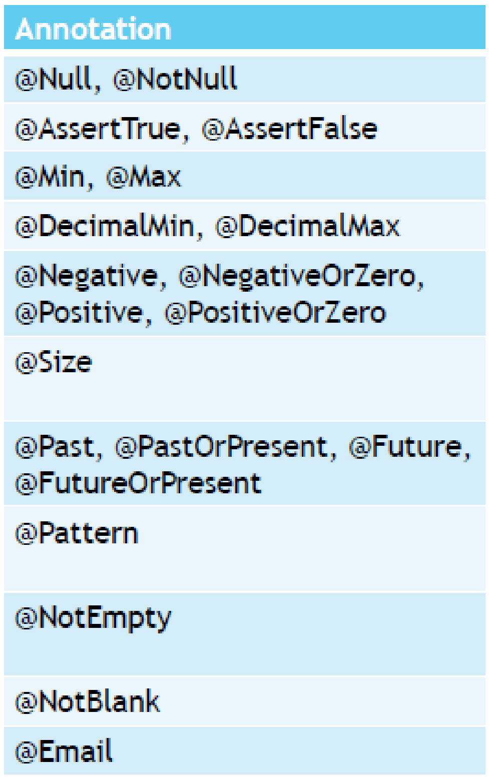
3.3.2 Entity, DTO, and Mapper

Entity is a class that can be saved to the database. These are usually object classes. When a class is annotated with “@Entity”, its variables are represented as columns in the database. In a database tables, as best practice, there are some specific columns that should always be included. These are an “id” column for the primary key of the table, a “version” column to keep track of modifications, creation date, and last modification date. Because each of the entities would need these values, I created a “BaseEntity” [Appendix 8] which included these columns. Other entities have all extended this base entity.

Entities created in the project and their variables:

- *Activity*: name, start date, end date, quota, remaining quota, owner, questions (List<Question>)
- *Authority*: id, authority (does not extend Base Entity but implements Granted Authority)
- *Enrolled User*: user id, name, surname, email, answers (List<Question>)
- *Question*: question, answer
- *Users*: name, surname, username, email, password

Even though all information in the entities are required at some point, there are times when all data is not required, or limited values of an entity is provided. For example, when user is logging in, user will only give us their username with password. When we need to get these values as an object from the JSON properties, we can use DTOs. DTO stands for Data Transfer Object. DTOs are not represented in the database and hence they are not entities. However, what makes them useful is a special constructor annotation named “@JsonCreator” with “@JsonProperty” used in its parameters. These annotations ease the transformation from JSON file to a Java object. In addition, we can put constraints on the variables which is again a very simple way to validate values. When the constructor is transforming JSON values into the parameters the annotations validate that the limits desired are met. If they are not it throws an error as a return to the HTTP request.



Annotation
@Null, @NotNull
@AssertTrue, @AssertFalse
@Min, @Max
@DecimalMin, @DecimalMax
@Negative, @NegativeOrZero, @Positive, @PositiveOrZero
@Size
@Past, @PastOrPresent, @Future, @FutureOrPresent
@Pattern
@NotEmpty
@NotBlank
@Email

Figure 8: Example constraint annotations in Spring. Most of these annotations are either used in the project or practiced during the research stage of the project.

Commonly used constraints are already available in the “javax.validation.constraints” library. But designing your own constraint annotation is also available. During the development of this project, I designed an annotation to validate Turkish ID numbers. It also worked but I realized I did not need it, so it is no longer part of the project. The annotation examples given above have their unique parameters. The use of annotations for size, future or present, and assert true can be observed in the Appendix 8.

DTOs in the project:

- User DTO
- Activity DTO [Appendix 8]

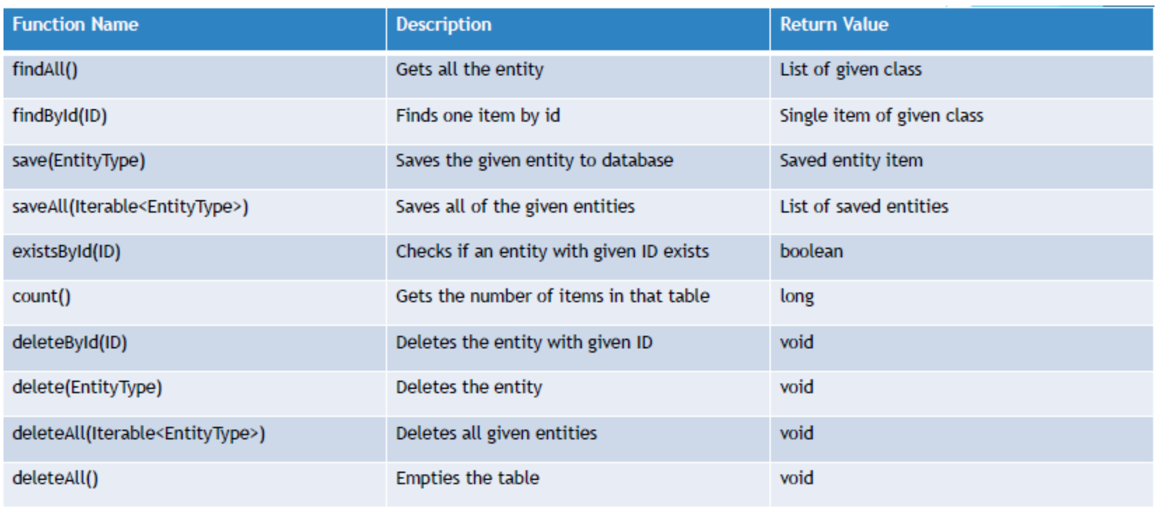
By this point we can accept and return a DTO object from the frontend and have entities in the database. What we need was a transition from DTO to entity. Mappers are doing exactly that. Mappers are an interface class, that have predefined methods names “mapToDto” and “mapToEntity”. When The return and parameter types are put in correctly in an abstract method, Spring automatically creates the method which maps one to the other. Meaning same variable names are matched with each other.

Mappers in the project:

- User Mapper
- Activity Mapper [Appendix 9]

3.3.3 Repository, Service, and Controller

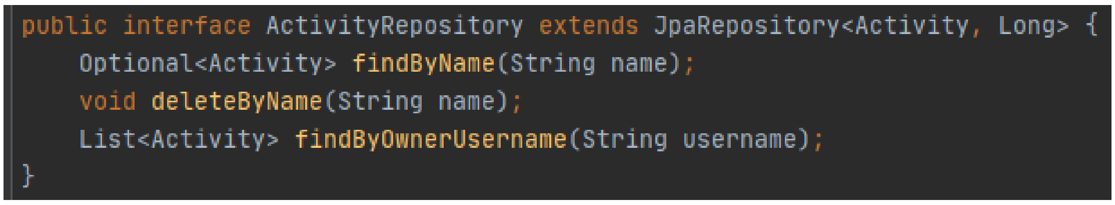
Repositories extend “JpaRepository” class from Spring Data JPA. Spring’s JPA framework enables an abstract database management by freeing you from session management, transaction and hibernate. There is only an interface and template functions which allow you to access database.



Function Name	Description	Return Value
findAll()	Gets all the entity	List of given class
findById(ID)	Finds one item by id	Single item of given class
save(EntityType)	Saves the given entity to database	Saved entity item
saveAll(Iterable<EntityType>)	Saves all of the given entities	List of saved entities
existsById(ID)	Checks if an entity with given ID exists	boolean
count()	Gets the number of items in that table	long
deleteById(ID)	Deletes the entity with given ID	void
delete(EntityType)	Deletes the entity	void
deleteAll(Iterable<EntityType>)	Deletes all given entities	void
deleteAll()	Empties the table	void

Figure 9: Current template functions in JpaRepository. Most of these functions are either used in the project or practiced during the research stage of the project.

In addition to the template functions given above, you can derive queries by writing the function in a specific format. For example, if the function name starts with “findBy” or “getBy” followed by a column name with logical keywords, the Jpa framework automatically creates the desired query in build. JpaRepository class takes two types: first is the Entity type and the second is the primary key’s type which is usually “Long”.



```
public interface ActivityRepository extends JpaRepository<Activity, Long> {
    Optional<Activity> findByName(String name);
    void deleteByName(String name);
    List<Activity> findByOwnerUsername(String username);
}
```

Figure 10: ActivityRepository interface used in the project which extends Spring’s JPA repository. Example for abstract functions are shown.

I did not need to use complex queries in this project. However, if needed writing an abstract function with name “findByNameOrderByDESC” would be enough the create the query. Furthermore, a Sort object can be added to the parameters and the function would sort the resulting value according to the given sort function. [8] I had to practice these queries while learning and researching Spring’s JPA repository. To sum up, any query required from the database is written as a function in repositories. Repositories created in this project:

- User Repository
- Activity Repository [Figure]
- Authority Repository

All features and functions in the program need to require data from the database or change some values in the database. Each function that has a unique requirement from the database is written in a “service” class. This class has a “@Service” annotation which is a variation of the “@Component” annotation. Service can be injected as bean as well and it is preferred in service layer classes because it specifies intent better.

The operations available in the program have their own functions. The function names are self-explanatory so I will only list the function signatures. Functions:

- List<Activity> listAllActivities()
- Activity getActivityByName(String name)
- List<Activity> getActivityByOwner(String username)
- Activity addActivity(Activity activity)
- Activity updateActivity(String name, Activity activity)
- boolean joinedActivity(String name, Users users, List<Question> answers)
- List<Question> getQuestions(String name)
- List<Question> addQuestions(String name, List<Question> questions)
- List<Question> addAnswers(String name, List<Question> questions)
- Activity updateActivityFromDB(Activity activity, Activity activityFromDB)
- void deleteActivity(String name)

These functions are calling the repositories to update the database and they are called from the “controller” class. More specifically a “rest controller” class. The Spring annotation “@RestController” allows the class to map REST requests (also mentioned as HTTP requests in this report) and request body is automatically accepted as the parameters of the mapping functions.

In the frontend development, REST requests were mentioned as requests made from the server that go to the backend to get information on database. In more detail, “axios” from the frontend send a request (get, post, put, delete) to the defined proxy server. In this project, the proxy is “http://localhost::8080” because Spring listens from port 8080. Then addresses are added to this proxy by axios. This request is captured and mapped by the rest controller. For example, if there was a get request sent to “/activities”, the function corresponding to that request type and address would be called in the rest controller. Then the function would perform what it supposed to, if there are no errors, it would return whatever is requested to the frontend to be displayed to the user.

Every function in the rest controller class has a request type mapping with one of the annotations “@GetMapping()”, “@PostMapping()”, “@PutMapping()”, and “@DeleteMapping()”. Parameters of the annotations may include the address or a variable which will be filled with the “@PathVariable” annotation inside the function’s parameters. Some data from the frontend is needed in PUT and POST requests. Putting a “@RequestBody” annotation in front of a parameter of the function would allow us to access to the data. An example controller function can be observed in Appendix 10.

After completing the repositories, service class, and rest controller class, I used Postman frequently to make sure each function is working correctly. Postman sped up the debugging process. It is a tool where you can send HTTP request to your program and you can give parameters, headers, body and many more. You also have a useful response section, that helps to understand what the problem is if there is one.

3.3.4 Security and Authentication

When entities, mappers, DTOs, repositories, service and rest controller classes are combined, the bridge between database and front is completed. Data man be modified and reached when needed. However, there is one important quality of an application that cannot be left out: security. The importance of protecting user’s personal information cannot be stressed enough. In order to have a secure environment, I had to create an authentication system. After trying multiple security systems, I landed on applying JSON Web Token (JWT) authentication.

JWT creates a token with three parts. First part is an encrypted version of the header. Second is an encrypted version of the payload. The last is the encrypted version of the signature. The encryption on first and second parts with a super-secret key should give the third part. That is basically how JWT validates requests.

In a util package, I created “JwtUtil” class which has helper functions for generating tokens, getting authorities, calculating expiration dates for tokens, and extracting username from tokens. This functions are created using the “Jwts” class in “io.jsonwebtoken” library.

A password encoder bean class is created by using “BCryptPasswordEncoder” class in “org.springframework.security.crypto.bcrypt” library. Another class to filter

Jwt requests has a function that filters the token into the three groups mentioned and tries to authenticate. This filter is given to the security configuration method of http. This configuration method is supplied by Spring. But it is overridden for this project. To access this function, I needed to create a class which extended "WebSecurityConfigurerAdapter" class in "org.springframework.security.config.annotation.web.configuration" library. The same class uses my password encoder when creating an authentication provider.

All objects related with login and their service and rest controller classes are programmed separately from the rest of the program. This was to have a more organized project structure.

3.4 Database Development

PostgreSQL and its development platform pgAdmin are used in the creation of the database. Database is where data of the application is stored in tables. Sequencing, indexing and associations between tables can be arranged in PostgreSQL. Changes in these tables are first requested from the frontend. Then these requests go to backend to get filtered and controlled. Backend decides which table to modify and then that table is updated in the database.

For privacy and security reasons, user passwords are stored in the database in encrypted forms. This encryption is also done in the backend. There is no programming done in the database part. All the database work is controlled from the backend and explained in the backend subsection 3.3.

Tables in this project can be divided into two groups. One group is tables based on entities (similar to objects in OOP) and the other tables are created from many-to-many relations of entity tables.

Entity based tables:

- *Activities*: is a table where each row has a unique and different activity. A new row is created when a new activity is created. And rows can be updated by the owner of that row's activity. Activities table has information about the activity (name, start date, end date, quota…) and a one-to-many relation with questions (to be asked to the enrolling user). Activities also creates another table through many-to-many relation.

- *Users*: has information of all users including their encrypted passwords. This table contributes to the enrollment table by providing "user_id". Users also has another many-to-many relation with authority table which creates the "user_authorities" table.

- *Authority:* has the types of authority users can have which are READ, WRITE and EXECUTE.

- *Enrolled User:* has limited information of the user and has a list of answers required to enroll in an activity. The answer list is a one-to-many relation with question entity.

- *Question:* has a single question with an answer. Answer field can be left empty.

Many-to-many relation-based tables:

- *Enrollment:* is automatically created when a user is enrolled in an activity. Multiple users can enroll the same activity and the same user can enroll to multiple activities. User info comes from the users table and activity information comes from "activities" table.

- *User Authorities:* is created with registration. Users usually have default authority (READ) unless specified otherwise.

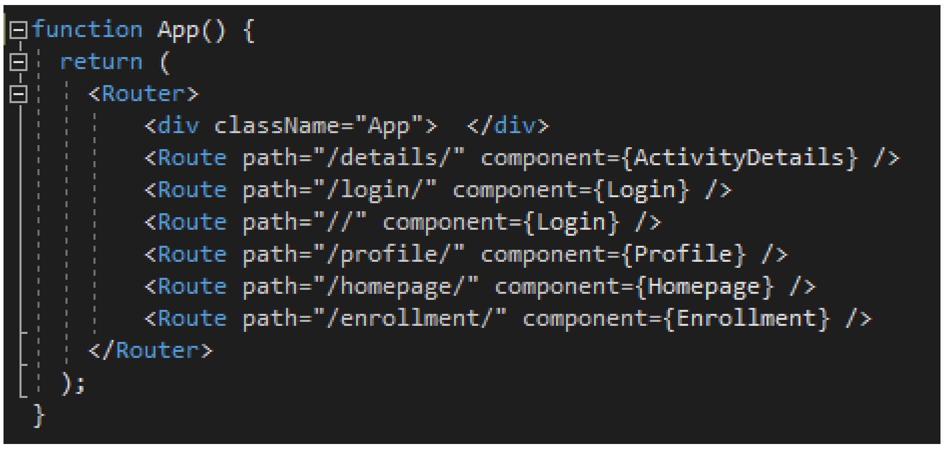
4 Performance and Outcomes

4.1 Solving Complex Engineering Problems

During the development of the project, I had multiple complex problems. Generally, the fact that I was learning everything for the first time made me use my researching skills to the maximum. I will briefly mention two of my biggest engineering challenges in this internship.

The first challenge for me was to design the project and more specifically the database relations. We were not given any designs or diagrams. In the workshops, we did not work on the project either. Therefore, designing and planning the project was all on us. Having no experience with web applications did not help either. So, I had to find many examples from Youtube or Github and investigated their project structures. However, deciding on the relations of the tables in the database was a different challenge. Because my problem was very specific, I could not use help from outside sources. So, I started trying different formations and observe the resulting tables. By using trial and error method, I realized how relations work in databases and then I figured out how I could represent my data in the most efficient way. For example, at first, I decided to do a different table for questions and a different table for question and answers. Then I realized that I could easily differentiate them by including user id and a many-to-many relation with enrolled user.

Another big problem for me was connecting the frontend with backend. Backhend had its default login page that I could not modify in frontend. And Frontend was listening on the localhost::4000 server whereas backend was listening from localhost:8080 server. After trying all the solutions I could find online and spending a lot of time on this problem. I decided to give up and reach my supervisor for help. He introduced me to “proxy” to change which port is React was sending requests to. For the login page, he also could not find a way to direct the default page to my desired page but managed to disable the default page. After managing to disable the login page, I thought about a practical way to fix my problem. I could not give a default route to the login page which has “/login” address but I thought I could link the default address which is empty to my login page. Sometimes complex problems need simple solutions.



```
function App() {
  return (
    <Router>
        <div className="App">  </div>
        <Route path="/details/" component={ActivityDetails} />
        <Route path="/login/" component={Login} />
        <Route path="//" component={Login} />
        <Route path="/profile/" component={Profile} />
        <Route path="/homepage/" component={Homepage} />
        <Route path="/enrollment/" component={Enrollment} />
    </Router>
  );
}
```

Figure 11: App function in App.js with the page routes. The fix was having two routes for the Login page where one of the routes is the default loading page.

4.2 Ethical and Professional Responsibilities

Because this was an online internship, I did not have the chance to meet with my supervisors nor observe the company environment. I also did not see any work-related ethical or professional issues. However, about professional responsibilities like communication, I was respectful and professional with them and they were the same to me. We communicated through mail or messaging in work hours. They tried to help me with my problems as much as they could.

About ethical responsibilities, I worked during the working hours at home. I believe that was an important responsibility on me because I could easily not work since I was at home. However, I know that the aim of the internship was to improve myself and they were only there to assist me while I worked. Since they spent time to teach me when they have their work, I thought that I could at least be present at work in working hours even it was online.

4.3 Making Informed Judgments

I believe I made one informed judgment and I learned about another during the internship. My judgment was to protect user information and creating a secure application. What I learnt during my internship was to use open-source software.

I think customer privacy should be considered in every decision in programming. It has a great impact on society. Privacy and safety in digital life will reflect to real life. Valuing your customers shows respect and a safe environment for people. If we managed to get every software to be as secure as possible, I believe the society will have a safer environment where people respect each other's privacy. That is why I worked especially hard on learning about authentication and security systems.

After learning more about open-source software in the internship, I decided to use open-source software in my future projects as well. And I will try to contribute more to this society. Using open-source products are safer, freer, higher quality, cheaper and easier. It is reachable to everyone and everyone can benefit it. I believe this community creates a globally safe environment where you do not need to spend too much. It encourages everyone around to world to get to know technology. This community unites curious people around the world to create something even better.

4.4 Acquiring New Knowledge Using Appropriate Learning Strategies

One of the reasons I chose to work at TÜBİTAK was their extensive use of new software and technologies. I wanted to learn and research as much as I could with the best researchers in Turkey. I can proudly say that I learnt a new thing everyday during my internship. Even though it was hard to keep up, I acquired so much in a short period of time.

Appropriate learning strategies for me is to understand and apply. I need to understand the core mechanism and logic behind a new knowledge. If I manage to comprehend its logic, then I must practice on my own. What I did during the internship was follow these steps. I learned the topic by listening to the workshops and lessons very carefully. I took notes on parts I was not confident with. I found different sources, like Youtube, to go over the topics. Some educational Youtube playlists were adviced by the supervisors. After feeling confident about the topic, I tried to find examples. For finding examples, I had a wider range of sources. I used programming challenge sites like LeetCode and Hackerrank as well as Youtube courses. If I am going to exercise through a Youtube tutorial, then I always write the code too. If I only watch without writing, then I cannot write on my own. If there is a specific thing to research, I tend to search programming community sources such as Stack Overflow or Geeks by Geeks. By following these strategies, I acquired the following knowledges.

- *Database:* I understood the concept of database and especially studied SQL databases and their language.

- *PostgreSQL:* I specifically got familiar with PostgreSQL, its open-source community, its user interface pgAdmin. I started using sequencing and indexing in database, but I think I still have many more concepts to learn.

- *Back-end Development:* Through the content of my main project, I got to understand what is done in this stage. I learnt which software tools and technologies are involved in the backend development stage.

- *Spring and Java:* Spring's libraries on Java, forced me to learn more about Java as well. I studied annotations, beans, injections, repositories, entities, streams and many more widely used concepts in real life usage of Java.

- *React and Axios:* React and Axios are both JavaScript libraries. These libraries introduced frontend development to me. I learned about components, states, render mechanisms, synchronous and asynchronous programming, REST requests, styling and many more.

- *JavaScript and CSS:* I had to learn these languages to code with React and Axios in frontend development. It was my first time using a scripting language and understanding style commands in CSS. These languages widened my perspective to programming.

- *Postman:* Postman is a program which can send HTTP requests and inspect their results. I used Postman in testing and debugging processes during my project development. Since the majority of time in any programming project is spent on debugging and testing, it was a very useful and helpful tool to learn.

4.5 Applying New Knowledge When Needed

The most important thing I used from my Bilkent education was Java and self-learning skills. Even though I had to study Java again, having ground information speed up my learning curve. Unfortunately, I did not have the same advantage at the rest of the parts of the internship. However, having practiced self-teaching and learning my comprehension method helped a lot more than knowing Java. I knew how to research and learn a topic the fastest way for myself. Applying my methods make me follow the heavy workload and intense education workshops.

On the other hand, about the project I can honestly say that there is not a solitary part which is not an application of new knowledge. The whole project is implemented thanks to brand new knowledge, tools, languages, or concepts. You can refer to the

Work Done section or 4.4 subsection to learn more about how much new knowledge I acquired and needed to use.

5 Conclusions

All in all, I have to say that the company and the internship was very fruitful and educational to me. YTE had many workshops and a detailed project plan that I did not have a second to waste during the internship period. I kept getting new knowledge and I got to practice on them right away. The supervisors were very helpful and attentive. I am truly grateful to have learned so much in so little time and to having the chance to learn from YTE's supervisors.

I learnt so much about real-life programming and communication skills as well. Even though we did not have a team project, I got to learn the language of different developers. I understood how each feature affects each stage in development and what is required to do a full-stack software development. To summarize again, I learned PostgreSQL for database, Postman tool, React and Axios libraries with JavaScript and CSS for frontend development, Spring Boot and new Java for backend development. I feel more like a computer engineer now. However, I believe the most important thing I learnt was how to research concepts that were new to me and the stages of software development. I have a clearer understand of what software development is and what I am studying. Furthermore, after the internship, I was reminded of how much I like researching and learning new software technologies. I realized that we have too much information waiting to be searched and technologies still have too much potential to grow. I wish follow new technologies better from now on. I am very grateful for such an enlightening internship in online conditions.

References

[1] “TÜBİTAK BİLGEM Software Technologies Research Institute”. https://yte.bilgem.tubitak.gov.tr/sites/images/Dosyalar/yte-katalog-eng.pdf. [Accessed: Oct 8, 2020]

[2] “The Open Group: Leading the development of open, vendor-neutral technology standards and certifications”. https://www.opengroup.org/about-us/what-we-do. [Accessed: Oct 8, 2020]

[3] Sağıroğlu, C. “TÜBİTAK BİLGEM Software Technologies Research Institute” https://yte.bilgem.tubitak.gov.tr/sites/images/Dosyalar/yte-presentation_september.pdf. [Accessed: Oct 8, 2020]

[4] Dumadag, J. (2018, May 17) “YARN vs. NPM” https://medium.com/@j.dumadag718/yarn-vs-npm-b2d58289fb9b. [Accessed: Oct 11, 2020]

[5] Chitalia, H. (2017, Oct 16) “Hey React, What is the Virtual DOM?” https://medium.com/coffee-and-codes/hey-react-what-is-the-virtual-dom-466ec333bf9a. [Accessed: Oct 11, 2020]

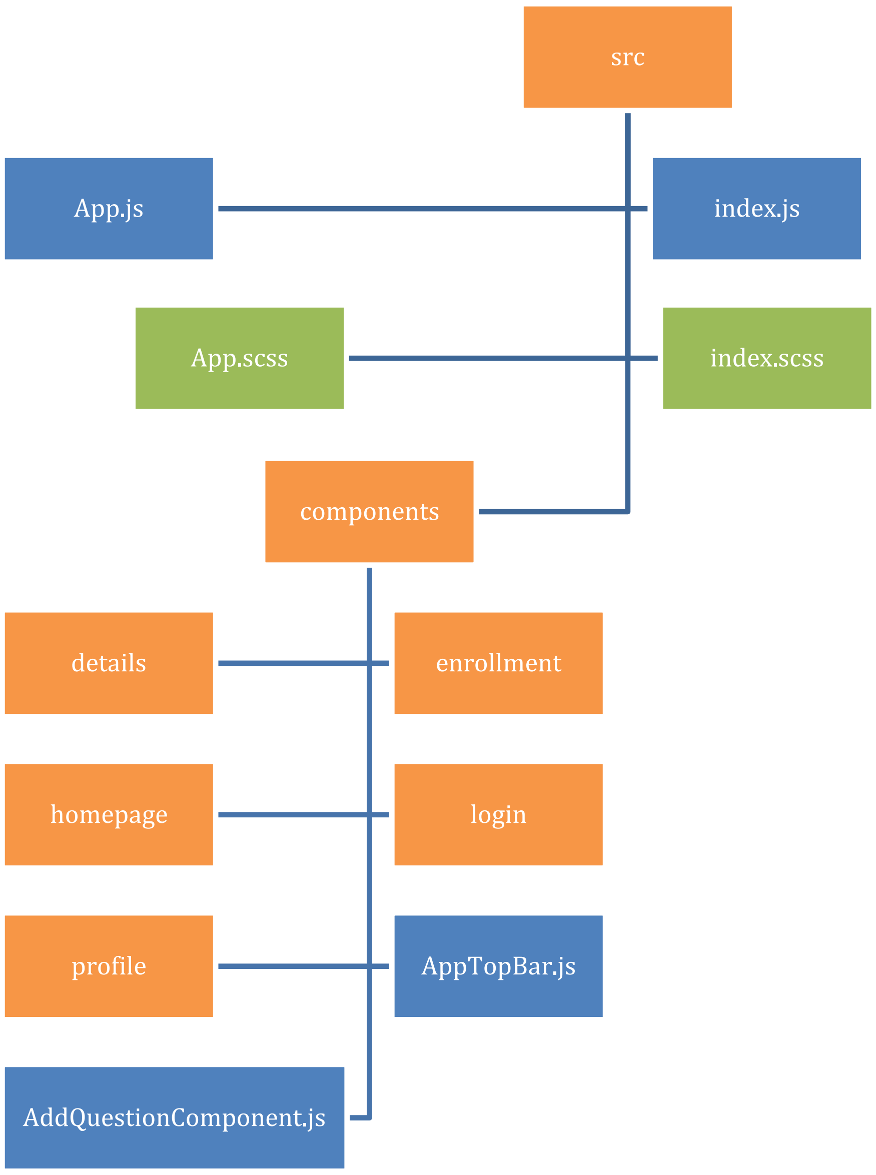
[6] “Spring Boot”. https://spring.io/projects/spring-boot. [Accessed: Oct 13, 2020]

[7] “Java: What frameworks do you use?” https://www.jetbrains.com/lp/devecosystem-2020/java/. [Accessed: Oct 13, 2020]

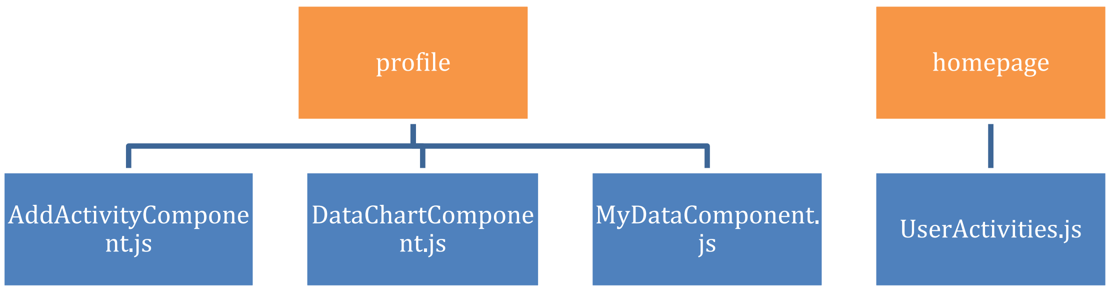
[8] Gierke, O., Darimont, T., Strobl, C., Paluch, M., Bryant, J. (2020, Sept 16) “Spring Data JPA - Reference Documentation” https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.repositories. [Accessed: Oct 13, 2020]

Appendices

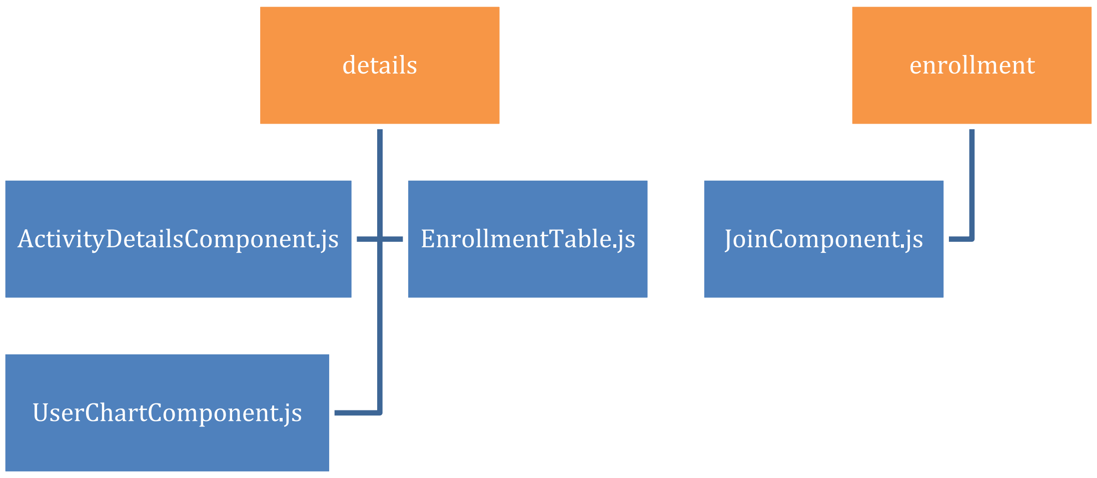
1.



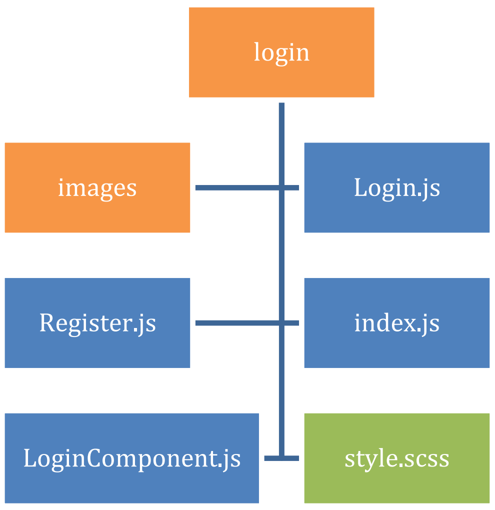
2.


profile
AddActivityComponent.js
DataChartComponent.js
MyDataComponent.js
homepage
UserActivities.js

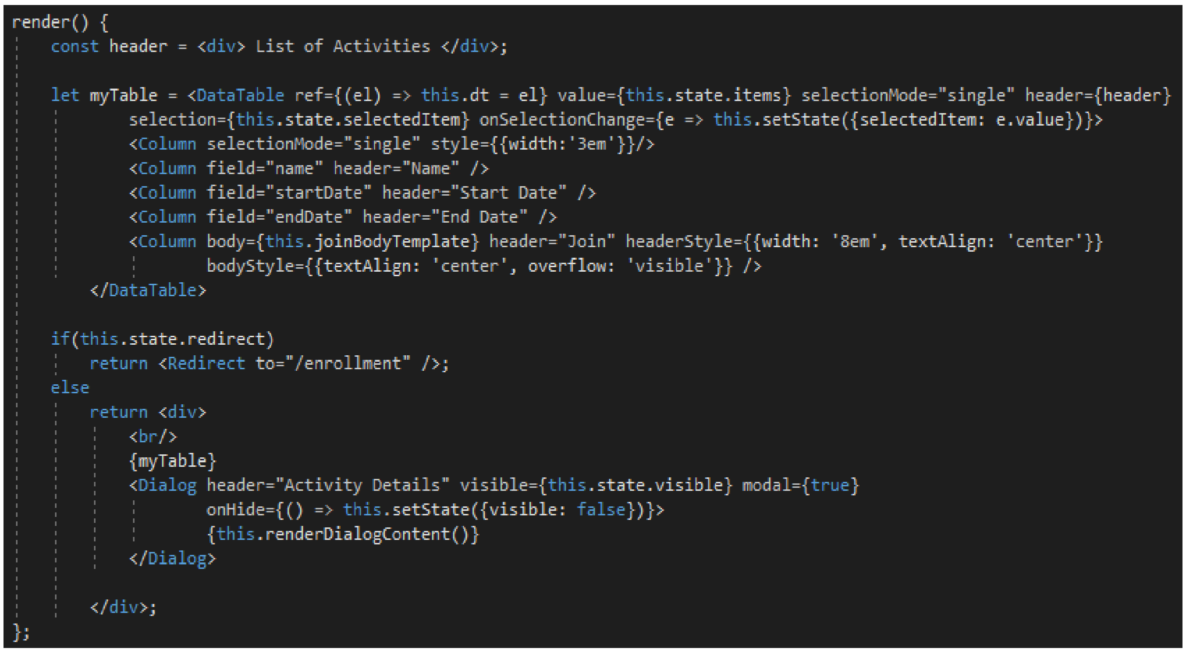
3.


details
ActivityDetailsComponent.js
EnrollmentTable.js
UserChartComponent.js
enrollment
JoinComponent.js

4.

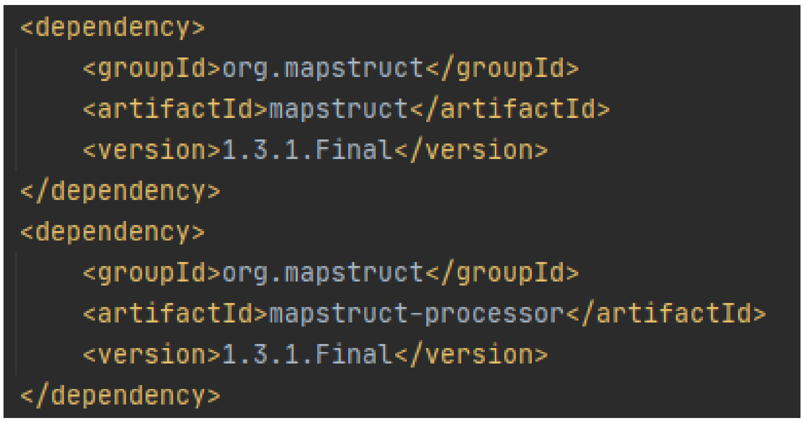

login
images
Login.js
Register.js
index.js
LoginComponent.js
style.scss

5.



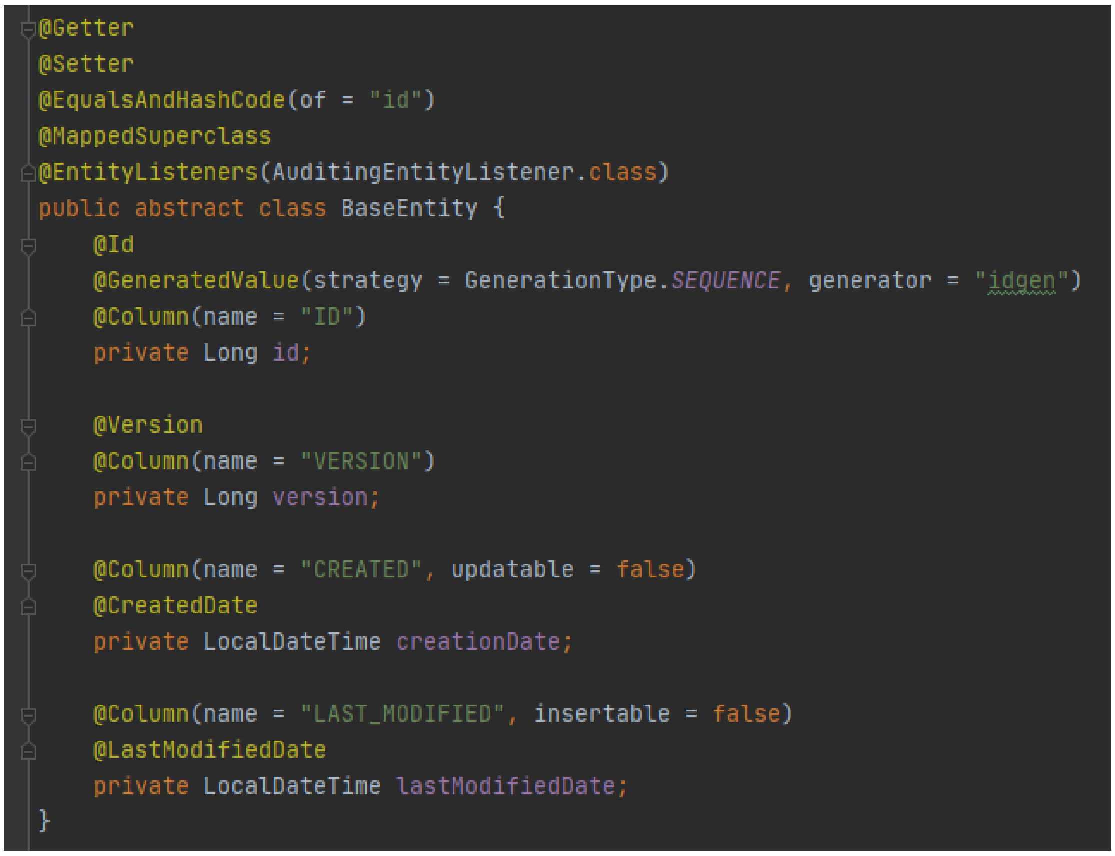
```
render() {
    const header = <div> List of Activities </div>;

    let myTable = <DataTable ref={(el) => this.dt = el} value={this.state.items} selectionMode="single" header={header}
            selection={this.state.selectedItem} onSelectionChange={e => this.setState({selectedItem: e.value})}>
            <Column selectionMode="single" style={{width:'3em'}}/>
            <Column field="name" header="Name" />
            <Column field="startDate" header="Start Date" />
            <Column field="endDate" header="End Date" />
            <Column body={this.joinBodyTemplate} header="Join" headerStyle={{width: '8em', textAlign: 'center'}}
                    bodyStyle={{textAlign: 'center', overflow: 'visible'}} />
        </DataTable>

    if(this.state.redirect)
        return <Redirect to="/enrollment" />;
    else
        return <div>
            <br/>
            {myTable}
            <Dialog header="Activity Details" visible={this.state.visible} modal={true}
                    onHide={() => this.setState({visible: false})}>
                    {this.renderDialogContent()}
            </Dialog>

        </div>;
};
```

6.



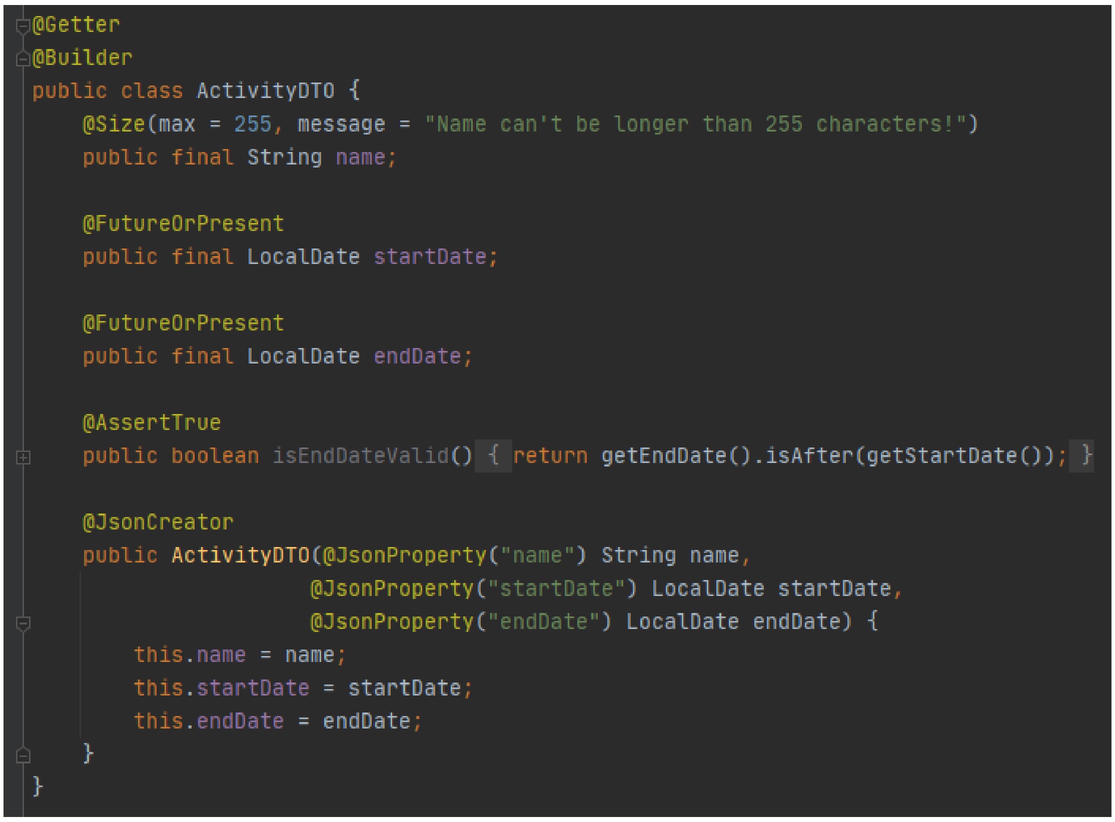
```
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
    <version>1.3.1.Final</version>
</dependency>
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct-processor</artifactId>
    <version>1.3.1.Final</version>
</dependency>
```

7.



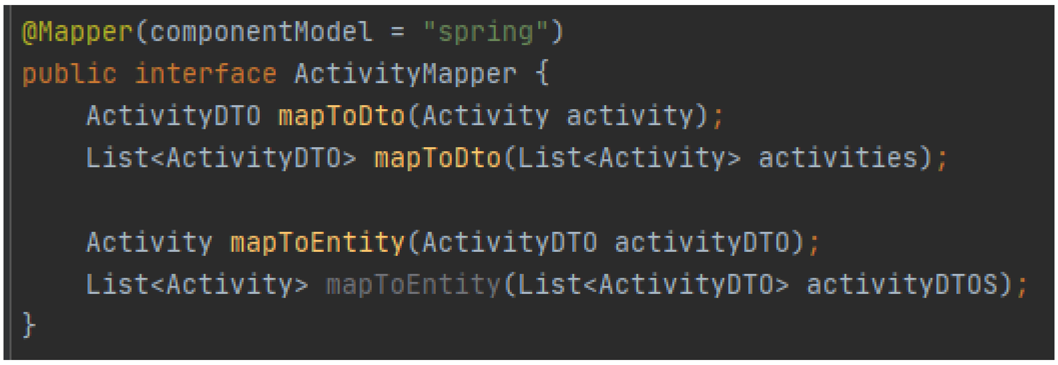
```java
@Getter
@Setter
@EqualsAndHashCode(of = "id")
@MappedSuperclass
@EntityListeners(AuditingEntityListener.class)
public abstract class BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "idgen")
    @Column(name = "ID")
    private Long id;

    @Version
    @Column(name = "VERSION")
    private Long version;

    @Column(name = "CREATED", updatable = false)
    @CreatedDate
    private LocalDateTime creationDate;

    @Column(name = "LAST_MODIFIED", insertable = false)
    @LastModifiedDate
    private LocalDateTime lastModifiedDate;
}
```

8.



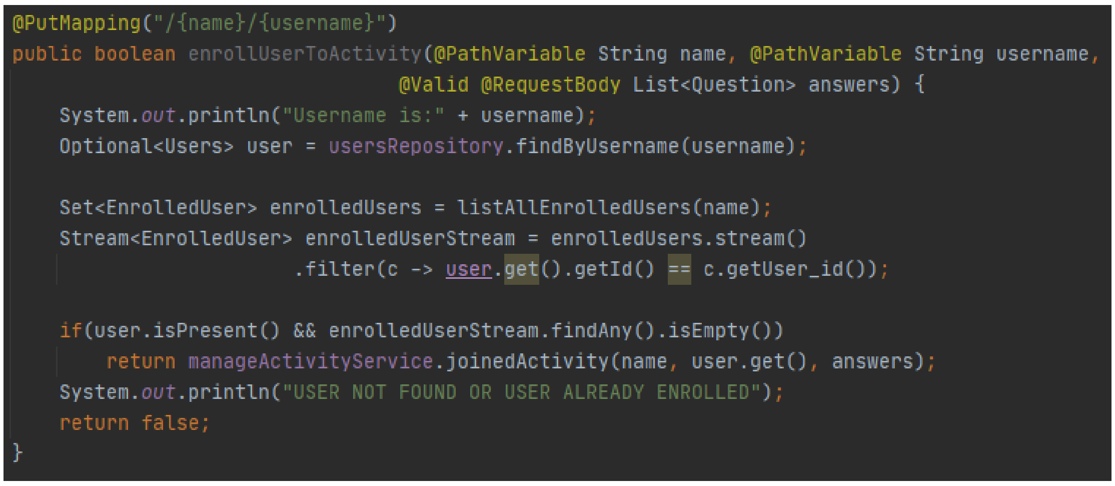
```java
@Getter
@Builder
public class ActivityDTO {
    @Size(max = 255, message = "Name can't be longer than 255 characters!")
    public final String name;

    @FutureOrPresent
    public final LocalDate startDate;

    @FutureOrPresent
    public final LocalDate endDate;

    @AssertTrue
    public boolean isEndDateValid() { return getEndDate().isAfter(getStartDate()); }

    @JsonCreator
    public ActivityDTO(@JsonProperty("name") String name,
                       @JsonProperty("startDate") LocalDate startDate,
                       @JsonProperty("endDate") LocalDate endDate) {
        this.name = name;
        this.startDate = startDate;
        this.endDate = endDate;
    }
}
```

9.



```java
@Mapper(componentModel = "spring")
public interface ActivityMapper {
    ActivityDTO mapToDto(Activity activity);
    List<ActivityDTO> mapToDto(List<Activity> activities);

    Activity mapToEntity(ActivityDTO activityDTO);
    List<Activity> mapToEntity(List<ActivityDTO> activityDTOS);
}
```

10.



```java
@PutMapping("/{name}/{username}")
public boolean enrollUserToActivity(@PathVariable String name, @PathVariable String username,
                                    @Valid @RequestBody List<Question> answers) {
    System.out.println("Username is:" + username);
    Optional<Users> user = usersRepository.findByUsername(username);

    Set<EnrolledUser> enrolledUsers = listAllEnrolledUsers(name);
    Stream<EnrolledUser> enrolledUserStream = enrolledUsers.stream()
                             .filter(c -> user.get().getId() == c.getUser_id());

    if(user.isPresent() && enrolledUserStream.findAny().isEmpty())
        return manageActivityService.joinedActivity(name, user.get(), answers);
    System.out.println("USER NOT FOUND OR USER ALREADY ENROLLED");
    return false;
}
```

Self-Checklist for Your Report

Please check the items here before submitting your report. This signed checklist should be the final page of your report.

☒ Did you provide detailed information about the work you did?

☒ Is supervisor information included?

☒ Did you use the Report Template to prepare your report, so that it has a cover page, the 8 major sections and 13 subsections specified in the Table of Contents, and uses the required section names?

☒ Did you follow the style guidelines?

☒ Does you report look professionally written?

☒ Does your report include all necessary References, and proper citations to them in the body?

☒ Did you remove all explanations from the Report Template, which are marked with yellow color? Did you modify all text marked with green according to your case?

Signature: ____Elif Kurtay_____________