Bilkent University

CS-319

Object-Oriented Software Engineering

Spring 2022/2023

DESIGN REPORT ITERATION 1

Group “BOSGII”

Instructor: Eray Tüzün

Teaching Assistant(s): Muhammad Umair Ahmed, Yahya Elnouby, Tolga Özgün

05/05/2023

Member Name	Student ID
Göktuğ Yılmaz	21903048
Berkay İnceişçi	21802088
Oğuz Kuyucu	21902683
Safa Eren Kuday	21902416
İlayda Zehra Yılmaz	22001769
İsmail Emre Deniz	21901913

1. Introduction

Our project aims to develop a web application dedicated to the evaluation process of internships, specifically for students and internship report graders affiliated with the Bilkent Engineering Faculty. The primary objective of the project is to automate the manual processes that currently exist in the system. This website allows students and graders to expect a smoother experience, saving time and reducing errors.

1.1 Purpose of the system

The purpose of the system is to enhance the overall experience of all parties involved in the process of handing in and grading the internship reports, including the department secretary, instructors/graders, and students. By automating and simplifying the tedious work typically handled by the department secretary, the system aims to streamline the entire process, saving time and reducing errors. Additionally, the platform will provide grader and student-friendly interfaces, allowing them to easily manage their tasks and stay informed throughout the process. Ultimately, the goal is to create a more efficient and user-friendly academic system.

1.2 Design goals

1.2.1 Security

The system has an admin who has complete control over all users, including the secretary, student, and grader. The system also ensures security for students taking CS299/399 courses. The secretary initializes students in the system, and the system sends them randomly generated private passwords for access.

1.2.2 Usability

To ensure a positive user experience, our web application should prioritize usability. This means designing the web page with a clear and intuitive layout, consistent design elements, and minimal as possible while being a functional interface. To enhance user experience, we will ensure that all functionalities in our web application can be accessed with a maximum of three clicks. Moreover, only the actions that are specific to the user type will be displayed, further reducing any unnecessary elements on the interface. This approach will simplify the user experience, enabling users to easily find and access the functionalities they need without encountering any difficulties.

1.2.3 Maintainability

Our web application should be maintainable in order to function effectively. Like many web applications, ours will be a complex system that can face several changes throughout the time of its lifecycle, such as new features, bug fixes, functionality updates, and performance enhancements. We can ensure 4 maintainabilities with these attributes: well-organized code, clear documentation, and standardized development practices.

2. High-level software architecture

2.1 Subsystem decomposition

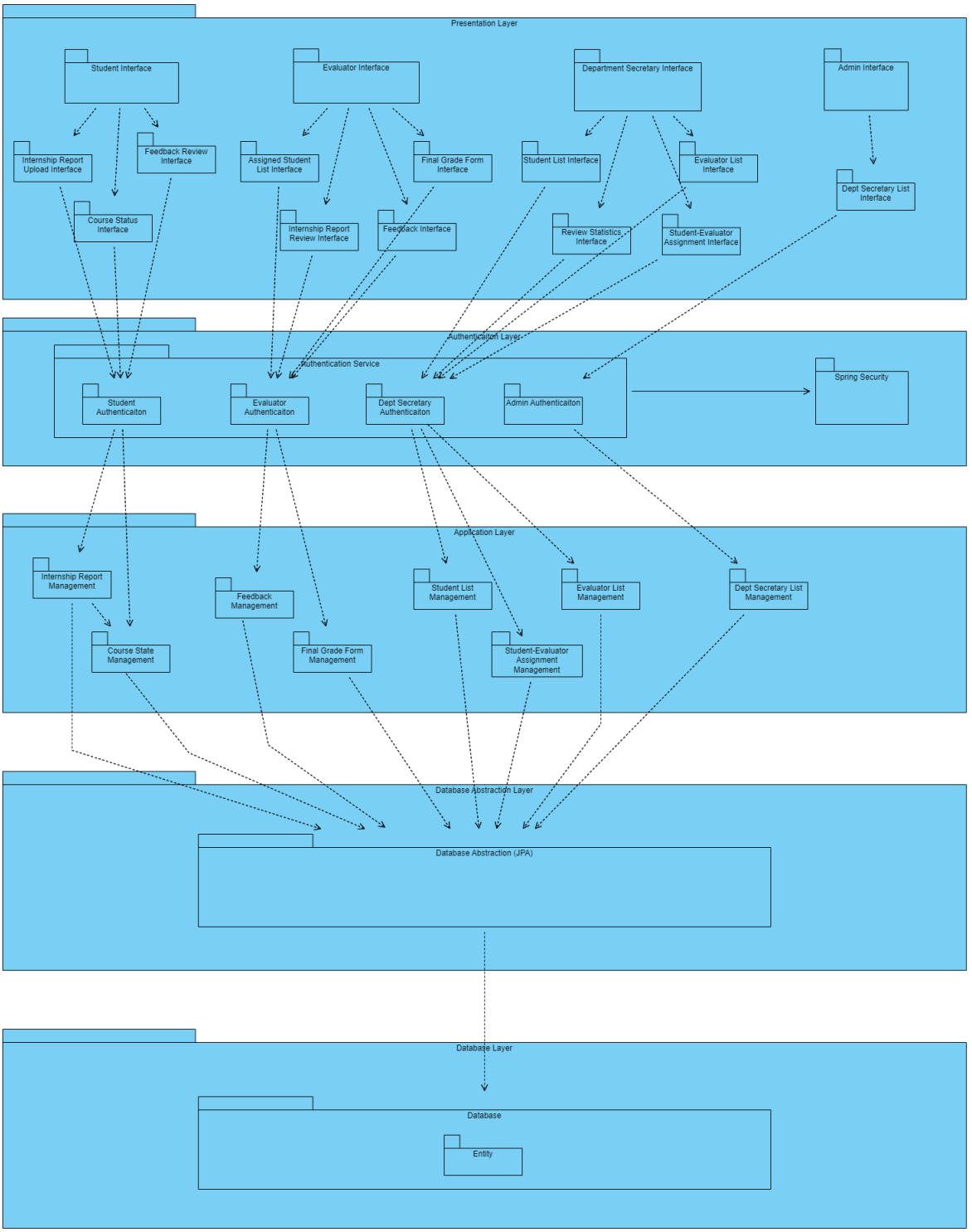

Figure 1: Subsystem Decomposition Diagram

Our internship report management system is divided into five different layers. The first layer is the Presentation layer, which is responsible for providing an interface for each type of user in the system - students, evaluators, department secretaries, and admin. Each user will have their own specific interface that they can use to access the system's features. Moving down the diagram, we have the Authentication layer, which is responsible for ensuring that only authorized users can access the system. This layer includes an authentication service and Spring Security, which work together to provide a secure login process. The third layer is the Application layer, which contains the core logic of the system. This includes classes which are responsible for handling the business logic of the system. The fourth layer is the Database Abstraction Layer, which provides an abstraction over the underlying database. This layer allows the system to communicate with the database without affecting the application code. Finally, we have the Database Layer, which is responsible for storing and managing the data required by the system. By dividing our system into these different layers, we can ensure that each component is modular and can be managed and maintained independently.

2.2. Hardware Software Mapping

For the hardware configuration, the database server and web server are going to provide the necessary hardware resources, such as CPU, memory and disk space for the database. Web server uses them to run the application logic of the web application and the database uses them to store and retrieve data for the web application. Amazon Web Serves (AWS) is chosen for an appropriate web server and it includes virtual machines (EC2 instances). For the database server Amazon RDS is going to be used, as well.

Now, let’s look at node responsibilities and communication between nodes. Amazon Web Serves is responsible for handling incoming requests from users and serving the necessary web pages and information to the users. It is also responsible for the application logic of the web application, so on the user side, it is going to process user input, interpret them and generate responses accordingly, and on the database side it is going to retrieve necessary data and store new or updated information. Amazon RDS is responsible for storing and retrieving data for the web application. It is going to listen to requests from the web server and send data asked accordingly. Also, it is responsible for storing data sent by AWS. Since we use MySQL, this communication is done using MySQL.

Frontend, the client-side of the application, is built using React, a JavaScript library for building user interfaces. The React application runs on the client-side, in the web browser, and communicates with the server-side using HTTP protocol. The server-side of the application is built using Java and the Spring Boot framework, which provides a set of tools for building web applications. The Spring Boot runs on EC2 instances and listens for incoming HTTP requests from the client-side of the application. The Spring Boot application communicates with the database server hosted on Amazon RDS using MySQL. The database server stores application data and responds to requests from the Spring Boot application.

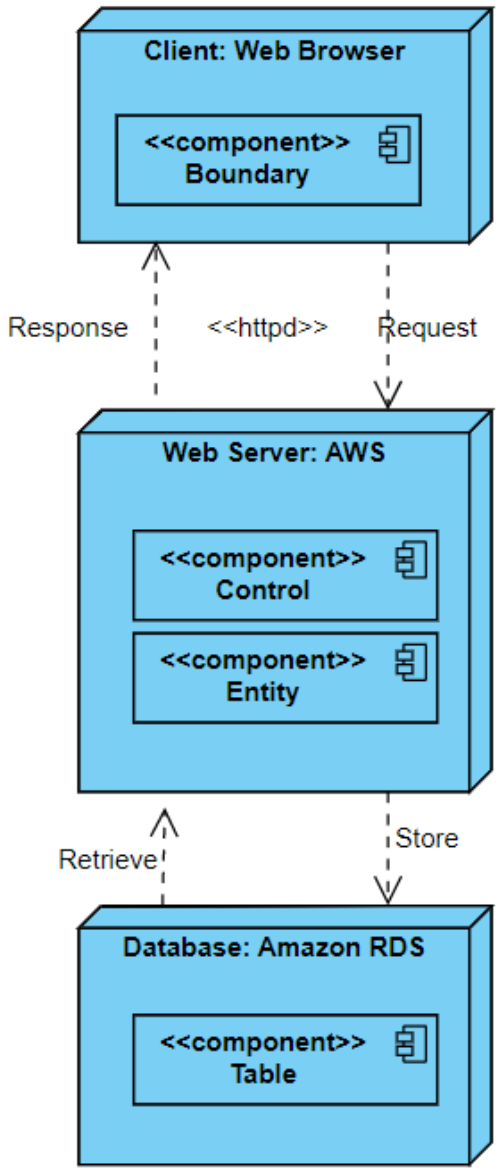

Figure 2: Deployment Diagram

2.3. Persistent Data Management

Persistent data management is the process of storing, retrieving, and managing data in a way that ensures its long-term availability and persistency. Our system has to be able to be restored in case it crashes we shut it down in a controlled manner. Therefore, we need persistent data management for our long-living objects. Above all, we have to maintain the data that is required for throughout all system executions rather than a single execution, such as entity objects and required info to execute the system.

We are using MySQL as a database language for our web application, and our database design involves storing the data related to entity objects. This ensures that the necessary information such as students, internship reports, grades, instructors etc., is stored in the database. This enables our system to be consistent because we keep the information in the database that is necessary for persistency of the system. That's why our system is able to bring this data every time required and to restore itself in case of a shutdown.

Additionally, MySQL provides mechanisms for data backup, replication, and recovery. These mechanisms can help ensure that the data is not lost due to hardware failure, software bugs, or other unexpected events. Additionally, MySQL provides security features to ensure that the data is protected from unauthorized access and malicious attacks. Also, since it is a relational database, it is even more reliable.

2.4. Access Control and Security

Our Internship Manager web-based application, BIM, contains important information such as mail, ID, passwords, and reports of the students. So securing the application and data has great significance. Therefore, each user has different authorization and verification levels, which will be handled by Spring Security. Authorization achieved by attaining roles for users. Each of the users has certain roles assigned to them. These roles are student, instructor, teaching assistant, secretary, and admin.

All users will be able to see just their user interfaces. For example, users whose role is a student cannot see the secretary home page. Same for the secretary. It will be handled on the client side. In addition, external users will not be able to access the system as all users

will be created and granted access by the system, which increases the security of the system and provides admin to control access of users.

On the server-side, we will implement a system that assigns a user's role upon registration to the system. Later, when a user attempts to log in, their user type information is accessed from the database and sent to the client-side. Furthermore, each user can only make specific requests to the server-side, as we restrict certain functionalities to certain user types. For instance, a student cannot grade the internship report, a secretary cannot provide feedback, and the admin cannot view or change the passwords of other users, which are hashed to keep user information safe.

Access Control Matrix

	Student	Instructor	TA	Secretary	Admin
Login	x	x	x	x	x
Change password	x	x	x	x	x
Upload report	x				
See grade	x				
View feedback report	x				
View status of feedback report	x				
Generate final PDF		x			
View assigned student list		x	x		
Download student internship report		x	x	x	
Drop Selected Student		x			
Give feedback		x	x		
Grade report		x			

View statistic				x	
Upload company evaluation form				x	
View student list				x	
View submissions of student		x		x	
Assign student to grader				x	
Initialize evaluators				x	
Initialize students				x	
View final PDF				x	
Reassign student the grader				x	
Initialize secretaries					x
seize secretaries					x
view secretaries					x

Figure 3:Access Control Matrix

2.5 Boundary Conditions

Here is an overview of our system's boundary conditions. Our goal is to have few failures overall. We want to create a recovery mechanism for system failures that would return us to the state in which failure occurred.

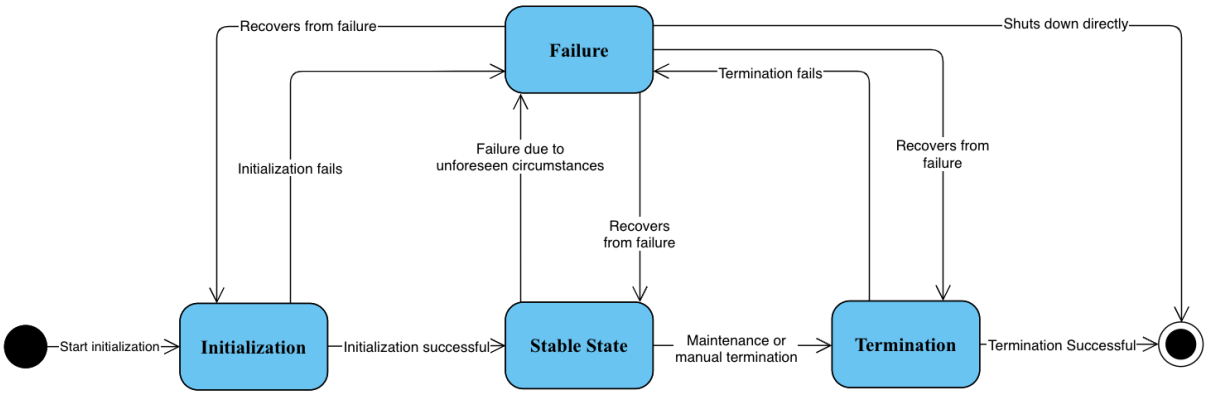

Figure 4: Boundary Conditions

2.5.1 Initialization

For development purposes, the services provided by the development tools in the local host will be used. The initialization process of the Bilkent Internship Management, BIM, application is going to start with running the MySQL Workbench program to start the MySQL database. Then, the com.bosgii.internshipmanagement file will be executed in the source file to make the backend accessible, enabling the REST API to function. The command "npm start" will be used to launch the React project for the front-end web application. Check the followings to start the application.

- ☐ Opening database
- ☐ Running the Spring Boot back-end
- ☐ Database connection
- ☐ Running React (front-end)
- ☐ Checking the communication between the back-end and front-end (REST API)

2.5.2. Termination

Termination of the project can be initiated by the admin because of maintenance or may occur due to unforeseen circumstances. In both cases, data loss is an undesirable situation for the application, and it must be addressed during the design phase. To ensure proper termination and avoid any failure, a specific order of steps should be followed.

If the admin decides to shut down the system, the reverse order of initialization should be used to correctly terminate the application. First, terminate the React application by typing "ctrl + c" in the terminal. Next, stop the BIM InternshipManagementApplication.java program. Finally, stop the database by using MySQL Workbench.

Following these steps will ensure that the project is correctly terminated, minimizing the risk of data loss or other issues that may arise during the termination process. Check the followings to terminate.

- ☐ Terminating React Application
- ☐ Stopping running InternshipManagementApplication.java file
- ☐ Closing the database

2.5.3 Failure

If a software problem or unexpected event occurs, the user will be directed to the login screen to prevent the system's database from becoming corrupted by invalid data or other issues. Implementing this strategy can significantly enhance the system's security and reliability. The project's objective is to minimize the occurrence of unexpected events that could potentially degrade usability and user experience.

Thanks to the use of a three-layered architecture, failures are less likely to occur in multiple layers at the same time. In most scenarios, the application can be recovered directly to a stable state. This architecture makes it difficult for the system to go down completely until the issue is in at least two layers, enabling quick recovery from failure.

3. Low-Level Design

3.1. Object Design Trade-Offs

Rapid Development vs. Functionality

In our internship management system, there is the necessity to develop our system in a limited time but also there is the necessity to add a lot of functionalities for different users (admin, students, instructors, etc.) who need some functionalities for internship reports (submitting, commenting, replying, etc.). In this case, developing these many functionalities

in a limited time is hard so we have to choose if we should spend more time and add more functionalities or spend less time and add not all of the functionalities. In this case, the functionality of the internship system is much more important than spending less time to create rapid development because adding all of the functionalities is necessary for the internship management system to work correctly. Thus, we should create a slower development and spend more time writing the program to add more functionality.

Functionality vs. Usability

The internship management system has many attributes that might lower the usability of the system when they all are implemented. For example, the system needs to enable students to generate internship reports, reply to instructor comments and add revised reports as well as letting the students see their old revisions. If they are taking two internship classes, they should be able to take action in both of them. This is just some functionalities for student users, and it is already very complex. Adding these functionalities and adding many others to the other user roles increases the complexity of the system which lowers the usability of the system. However, we chose that the users should be able to perform complex functionalities even if it reduces usability.

3.2 Final Object Design

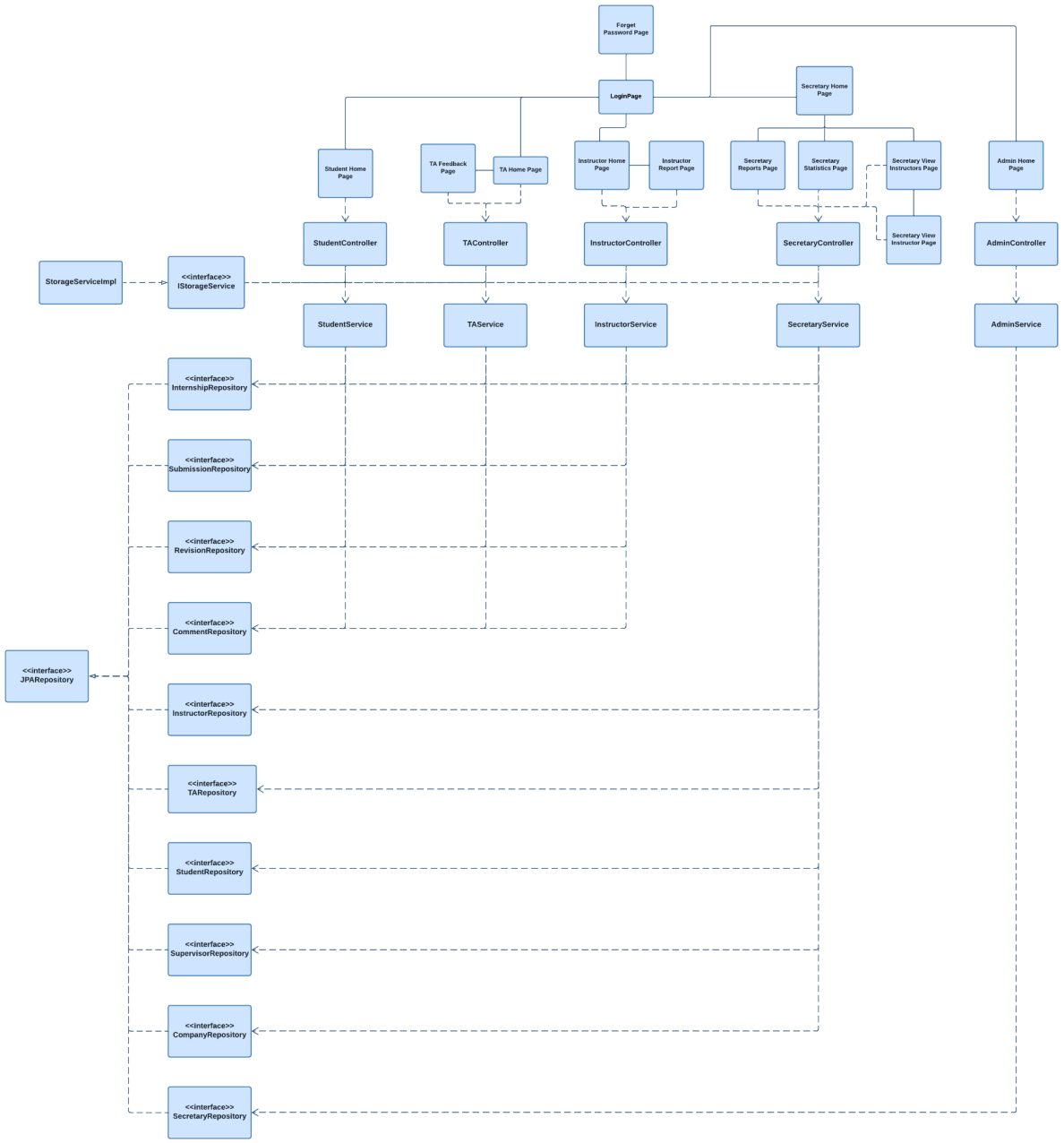

Figure 5: Final object design

3.3. Packages

3.3.1 User Interface Diagram

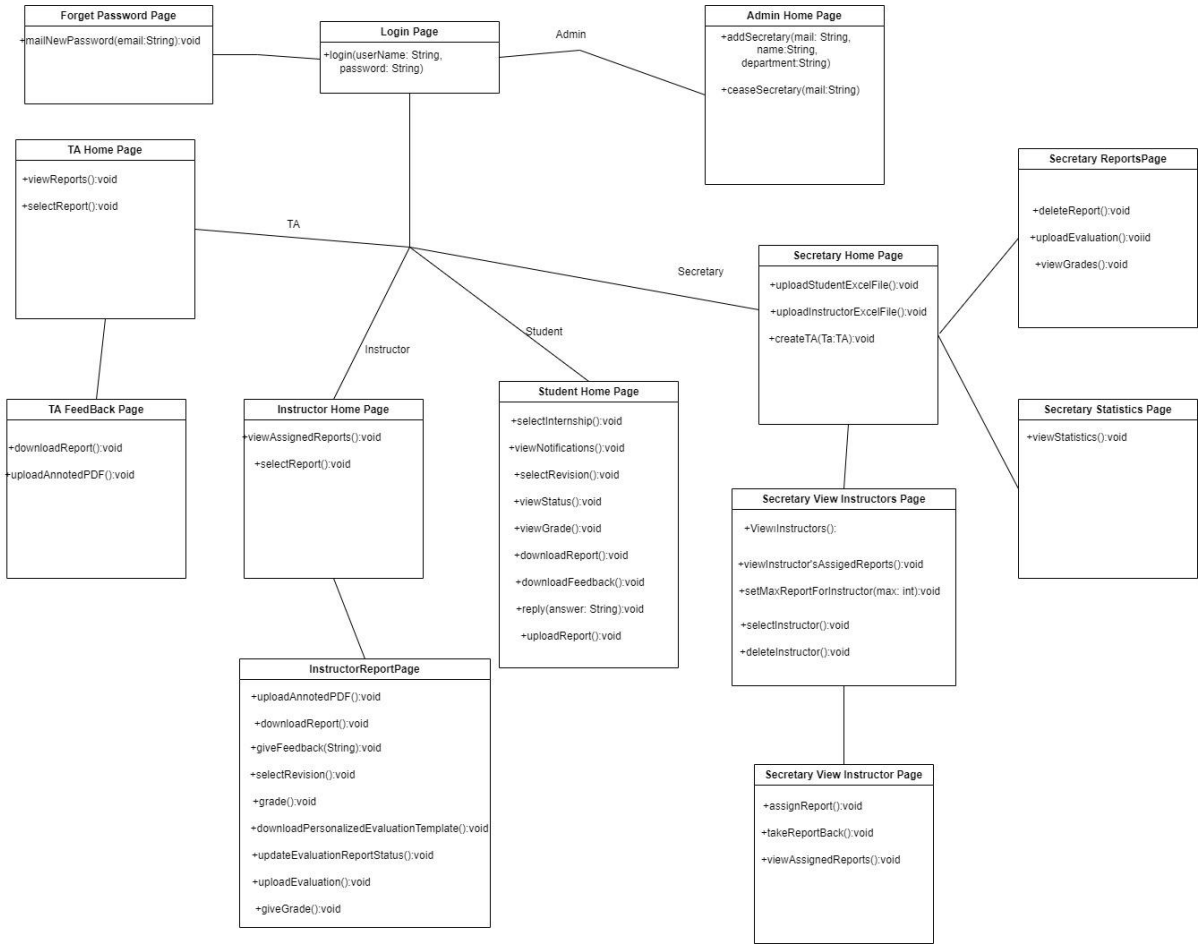

Figure 6:User Interface Diagram

3.3.2 Web Server Layer

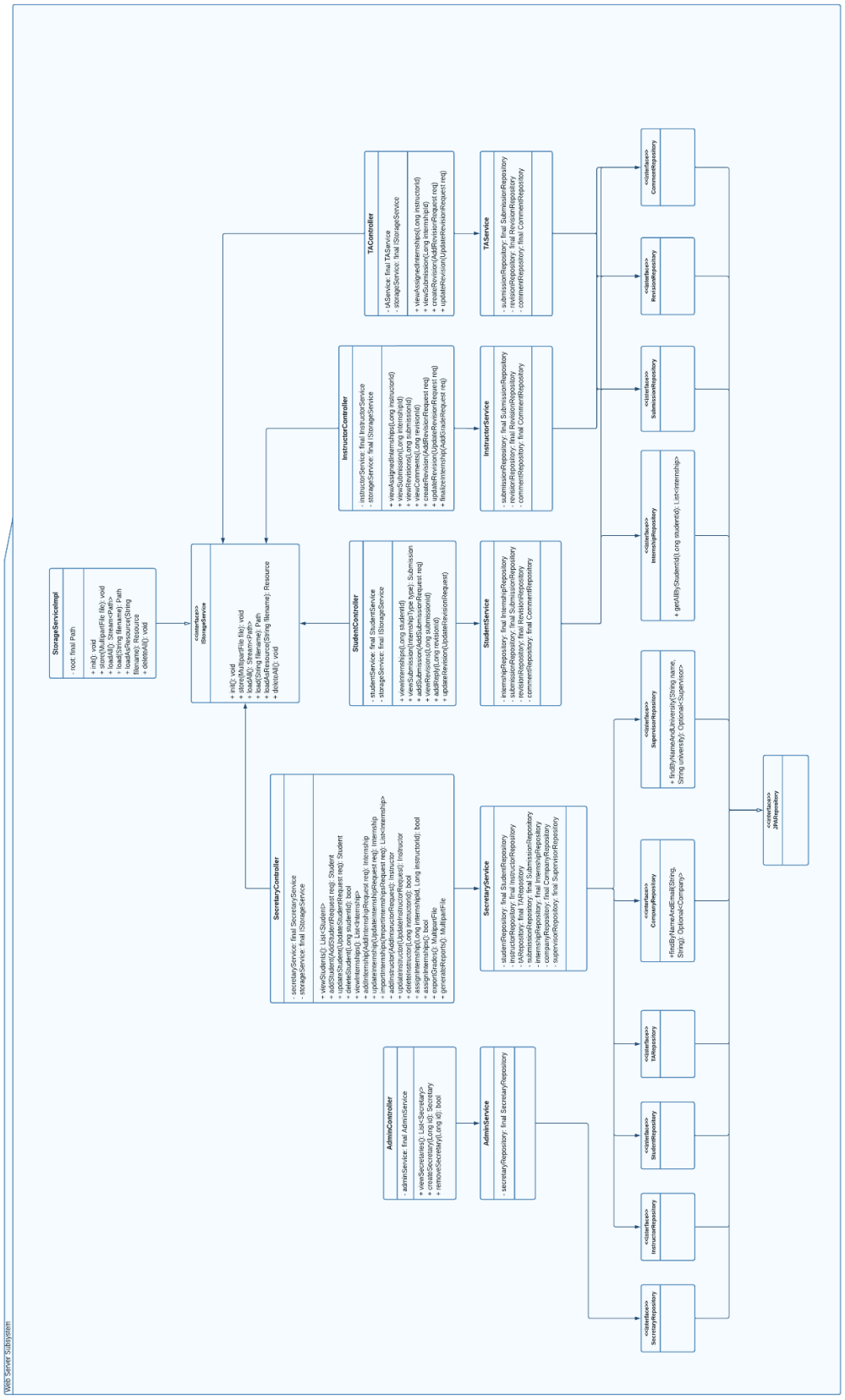

Figure 7: Web Server Layer Diagram

To avoid redundancy, methods for the Service classes are omitted. They are basically the same as the methods in the Controller classes, but they may vary in parameters and return types.

Request classes such as “AddRevisionRequest” or “UpdateInternshipRequests” are classes without methods, they are used to map the HTTP request messages to POJO’s.

3.3.3 Data Management Layer

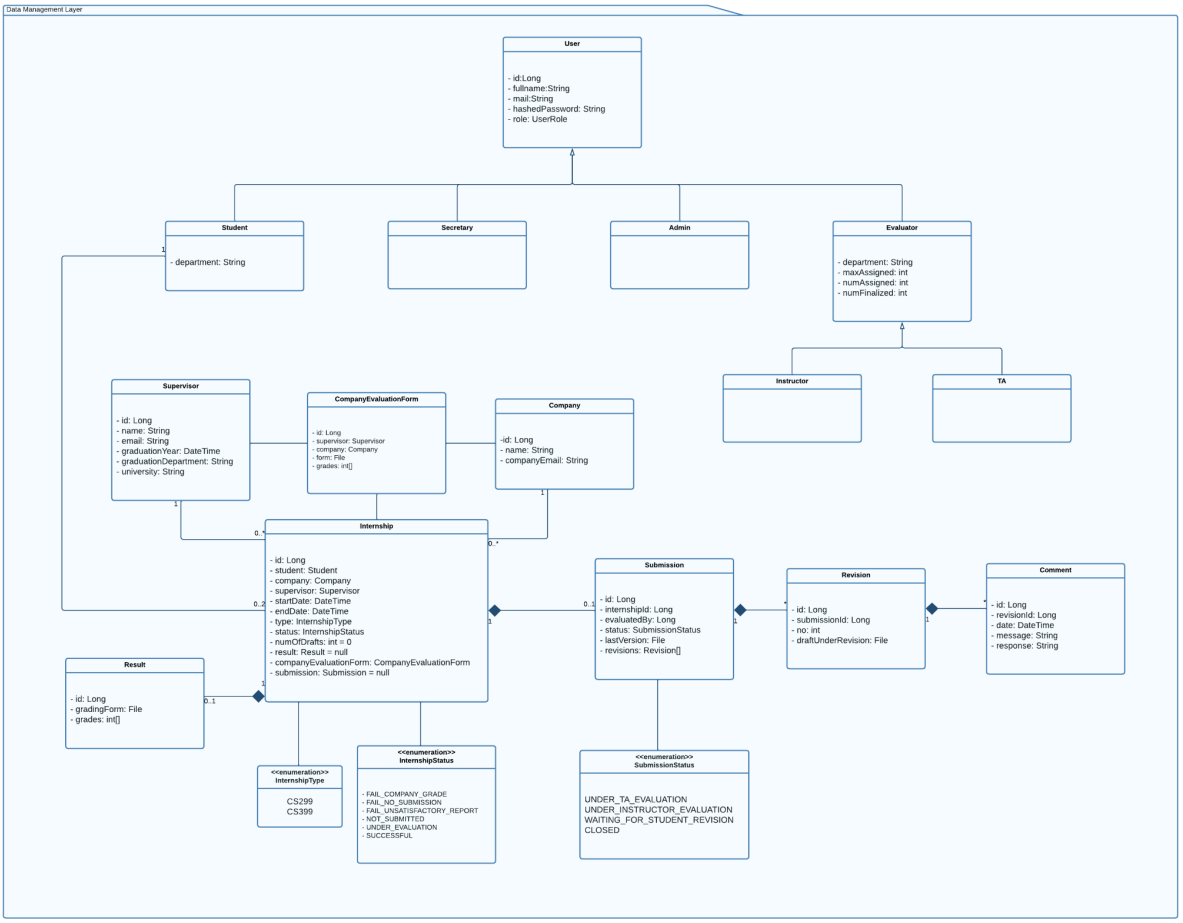

Figure 8: Data Management Layer Diagram

The associations between classes are achieved via foreign keys, which makes the size of JSON messages considerably smaller. For example, if the “Internship” class contained an instance of the “Submission” class, the JSON message would be nested and contain the submission file along with all the revisions, comments, etc. Instead, the id of the internship is added into the Submission class, as a foreign key. If a submission for a particular internship is desired, a search in the database according to the internship id needs to be performed.

3.4. Class Interfaces

3.4.1. User Interface Layer Class Interfaces

Secretary Report Page

deleteReport(): this function deletes report/internship completely. It is usable in case a student withdraws.

Instructor Report Page

downloadPersonalizedEvaluationTemplate(): this report downloads personalized(includes student's name, course etc) report evaluation template.

selectRevision(): this method selects revision 1, 2, 3 etc.

Student Home Page

selectRevision(): this method selects revision 1, 2, 3 etc.

3.5. Design Patterns

3.5.1. Facade

The first use of the Facade design pattern in the backend of the application is to provide an abstraction to the database access. For this purpose, "Repository" interfaces that extend the "JPARepository" interface provided by the Spring Boot are used.

Another appearance of the Facade design pattern is between controllers and the data access layer (repositories). Instead of performing boundary checks and applying business logic in the controller layer, a service layer for each controller is provided.

3.5.2. Strategy

In the application, a system for uploading/downloading files is needed. To achieve so, an interface called "IStorageService" is created, which is used by the controller classes. The interface provides only one concrete implementation for now, called "StorageServiceImpl", but another concrete implementation might be added in the future that uses different libraries.

3.5.3. Observer

To implement notification features in the application, Observer pattern will be used. The students will be notified when an instructor requests a revision or finalizes the submission, and the instructors will be notified when students make a submission or upload a revised version. In this case, both Student and Instructor are going to be Subject and Observer at the same time.

4. Glossary

5. References