



SUMMER TRAINING REPORT EVALUATION SYSTEM

Design Report

Instructor: Eray Tüzün

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

Group Name: Quokka

Berkay AKKUŞ - 21903321

İdil ATMACA - 22002491

Emre Melih GÜVEN - 22003944

Mehmet Emre KANTAŞ - 22003693

Yusuf ÖZYER - 21903339

Deniz SUN - 22003981

May 5, 2023

1. Introduction.....	4
1.1 Purpose of the System.....	4
1.2 Design Goals.....	4
1.2.1 Usability.....	4
1.2.2 Automation.....	4
2. High-Level Software Architecture.....	6
2.1. Subsystem Decomposition.....	6
2.2. Hardware/Software Mapping.....	7
2.3. Persistent Data Management.....	8
2.4. Access Control and Security.....	9
2.5. Boundary Conditions.....	12
2.5.1. Initialization.....	12
2.5.2. Stable State.....	12
2.5.3. Termination.....	13
2.5.4. Failure.....	14
3. Low-Level Design.....	14
3.1. Object Design Trade-offs.....	14
3.1.1. Usability vs Functionality.....	14
3.1.2. Automation vs Rapid Development.....	15
3.2 Final Object Design.....	16
3.3. Layers.....	17
3.3.1 User Interface Management Layer.....	17
3.3.2 Web Server Layer.....	18
3.3.3 Data Management Layer.....	19
3.3.3.1 Entity.....	19
3.3.3.2 Database.....	20
3.4. Class Interfaces.....	21
3.4.1. User Interface Layer Class Interfaces.....	21
3.4.1.1. LoginPage.....	21
3.4.1.2. FAQpage.....	21
3.4.1.3. RecoverPasswordPage.....	21
3.4.1.4. UserTypePage.....	21
3.4.1.5. ProfilePage.....	21
3.4.1.6. OthersProfilePage.....	21
3.4.1.7. OwnProfilePage.....	21
3.4.1.8. FileSubmissionPage.....	21
3.4.1.9. CompanyFormsPage.....	22
3.4.1.10. NotificationsPage.....	22
3.4.1.11. MakeAnnouncementPage.....	22
3.4.1.12. Sidebar.....	22
3.4.1.13. ReportsHistoryPage.....	22
3.4.1.14. ReportsEvaluationPage.....	23
3.4.1.15. GiveFeedbackPage.....	23
3.4.1.16. StatisticsPage.....	23

3.4.1.17. DisplayList.....	23
3.4.1.18. ManageUsersPage.....	23
3.4.1.19. StudentListPage.....	23
3.4.1.20. InstructorProgressPage.....	24
3.4.1.21. AddUserPage.....	24
3.4.1.22. EditUserPage.....	24
3.4.1.23. RemoveUserPage.....	24
3.4.1.24. ReassignInstructorPage.....	24
3.5 Design Patterns.....	24
3.5.1 Decorator Pattern.....	24
3.5.2 Strategy Pattern.....	25
3.5.3 Façade Pattern.....	25
4. References.....	27

1. Introduction

1.1 Purpose of the System

Summer Training Report Evaluation System is a web application with the aim of automating and easing the summer training report evaluation process of Bilkent University Faculty of Engineering. The current system relies both on paperwork and digital platforms such as Google Drive, e-mail, Microsoft Excel, and Moodle. This application will help the academic staff complete the evaluation process faster and more reliably while improving time efficiency. The administrative assistants will upload the company evaluation forms to the system. Students will upload their summer training reports and instructors can fetch them from the system to grade them. Students can track their own submission progress and the instructors can track the submissions of their assigned students. Additionally, administrative assistants, instructors, and summer training coordinators will be able to see the statistical data for all submission stages and their completion status.

1.2 Design Goals

1.2.1 Usability

Users can reach the functionalities of the system easily without needing to navigate through many pages. The main functionalities are added to a sidebar that is specific to user types with their related actions. Users will be able to reach the page they want in maximum 1 click due to having them in the sidebar. By gathering the functionalities on the sidebar, the content organization and navigation options are enhanced, which improves the overall user experience.

1.2.2 Automation

The current system is very time-consuming as there are a lot of platforms and paperwork involved. This system will reduce the workload of instructors, administrative assistants and summer training coordinators. Administrative assistants can upload students' company evaluation forms to the system and instructors can download them when entering a student's grades. Instructors can check these evaluation forms and enter the assessment scores easily on the grade form. It also

reduces possible human errors related to paperwork and grading. As the system will keep track of each student's progress, it will also improve the trackability of the summer training reports' approval states.

2. High-Level Software Architecture

2.1. Subsystem Decomposition

<https://online.visual-paradigm.com/share.jsp?id=323434323430362d31#diagram:workspace=wnzurcod&proj=0&id=1>

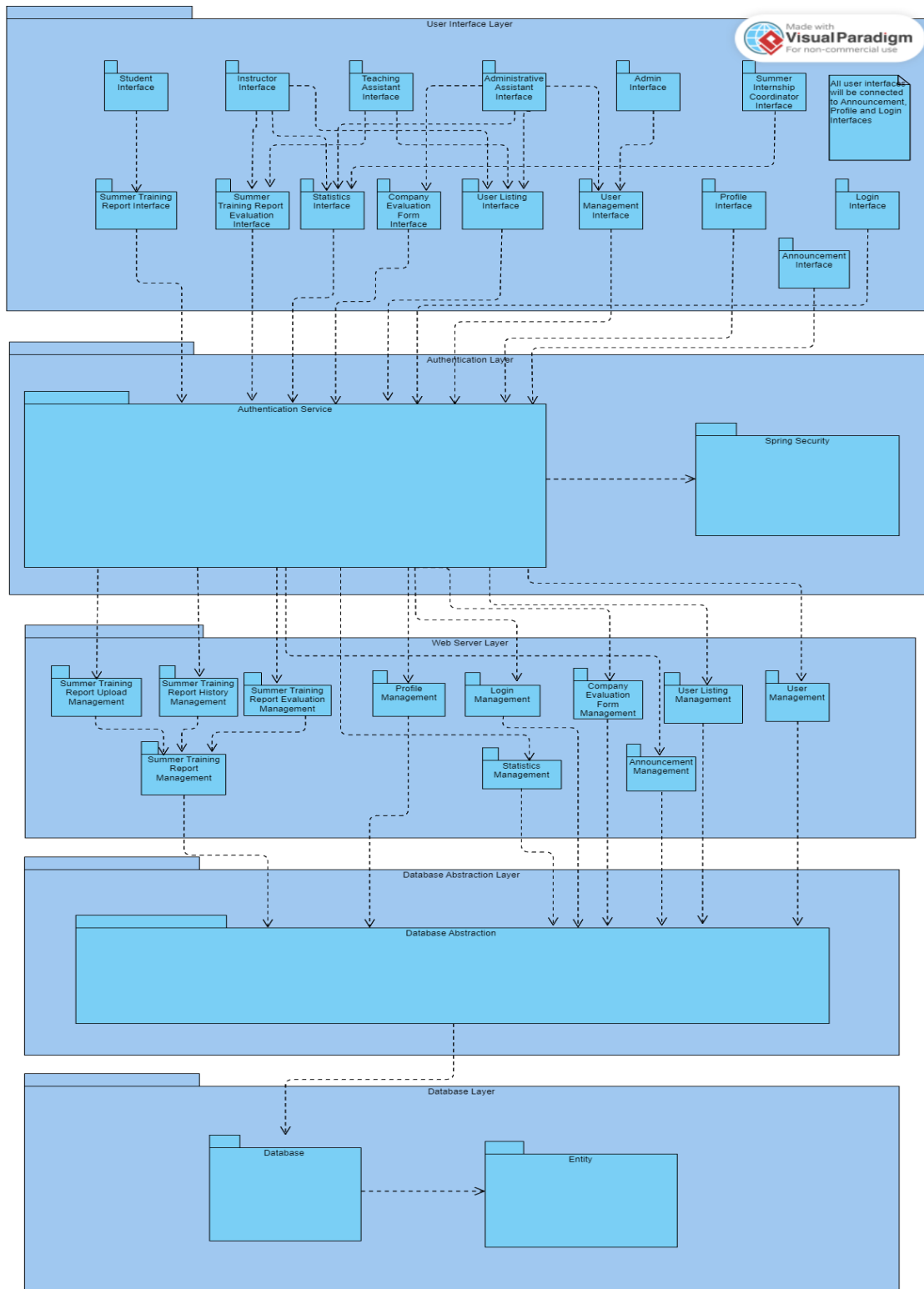


Figure 1: Subsystem Decomposition Diagram

2.2. Hardware/Software Mapping

We chose to use React in the front end for the Summer Training Report Evaluation System. Along with React, we will be using HTML5 and CSS3, that is built inside the React code. React offers high performance and speed. It is easier to maintain because of its modular structure and ease of learning. The variety of packages inside it makes it easier to set up the system and add new functionalities.

According to 2022-2023 academic year data, the total of second and third-year students is 469 for CS. The courses for summer training, CS299 and CS399, are taken at the end of the second and third years. The number of students taking their department's summer training course is 375, 406, and 198 for IE, EEE, and ME, respectively. The sum of engineering faculty students taking these classes is approximately 1448. Still, adding up the exceptions and other users like instructors, administrative assistants, and summer training coordinators can lead to 1600 users. Moreover, the system should be scalable enough to involve all engineering faculty users and work great with scalability. So, we have considered the requirements for such a system to have at least 8 GB of RAM, an SSD with at least 500 GB of storage, with room for growth, a 64-bit operating system, and a CPU with a clock speed of at least 2.5 GHz.

We will be using Java, Spring Boot, and MySQL for the backend. Java was the natural choice since our group is familiar with it. Spring Boot is an ideal framework for web applications built on Java, making it a great fit for our project. We selected MySQL as our database because it's widely used with projects using React and Spring Boot, letting the team utilize useful packages in Spring Boot for MySQL.

Moreover, we will be using local service for our MySQL database since it will provide us complete control over the security of our data, and performance is faster compared to a cloud-based server. To enhance usability, we will deploy the system from one machine only. The user will connect through the HTTPS link from their computer and run the system on the web server. In this way, the user would only need a device and a stable internet connection to run the system.

Deployment diagram:

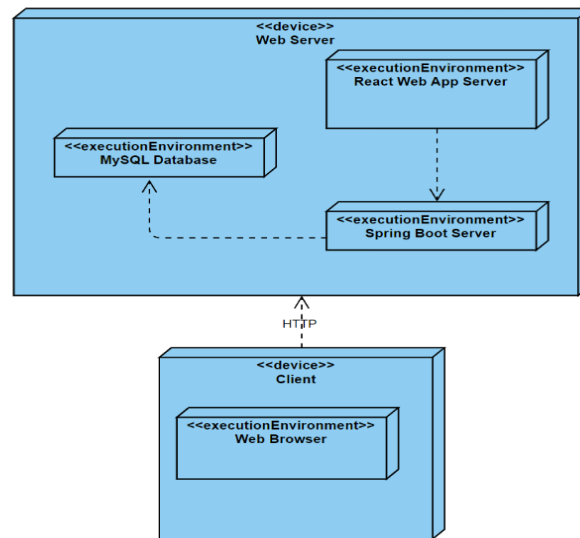


Figure 2: Deployment Diagram

2.3. Persistent Data Management

For our internship management system's database, we carefully considered both MongoDB and MySQL due to their widespread use and abundance of documentation. While MongoDB allows for a more flexible schema design, MySQL is better suited for fixed schema designs and structures. After careful consideration, we decided to proceed with MySQL for our internship management system because we chose React for the front end, and connecting MySQL with React is relatively easier and has more resources available for it.

We will implement a fixed data schema for our database that will include tables for students, instructors, and other necessary entity attributes such as name, surname, account id, and user role. Students will also have a class code associated with them. We will create relationships between these tables, such as assigning students to teaching assistants and instructors. A fixed and structured data schema will allow us to manage and maintain the system more easily than a flexible schema design.

Moreover, we will be using the Spring Data JPA package that Spring Boot provides. This package enables an abstraction layer to connect MySQL and Spring Boot with minimal query usage.

2.4. Access Control and Security

Our group prioritizes access control systems and security. We will be using the data sheets provided by the school and adding them to the system. Adding and removing a user from the system can only be done by Admins or Administrative assistants. We do not provide a signup option for our users regarding security. Instead, after the admin or administrative assistant adds the user to the system, they will receive their encrypted password via their Bilkent email. When a user is added to the system, the person who is adding them will be responsible for assigning their role, which will then be stored in the database layer. In this way, we will be able to differentiate various kinds of users. We are aware that a user can have two roles. This is common for summer training coordinators being an instructor as well. So, we will be providing an extra page after the login to these users that gives them a chance to switch between roles.

For the front end, users will only be shown pages that are related to them via the sidebar. We will be applying restrictions to pages to block a user from accessing a page that is not meant for them via changing the links in the browser. For example, a student cannot see the report evaluation page, whereas the instructor cannot see the upload reports page. Furthermore, whenever a user logs in to the system, we will get their role from the database and render the sidebar accordingly.

For security's sake, users have to log in to perform any of the use cases. We have assigned crucial acts to the actors that have high reliability. For example, only an administrative assistant would be able to send a reminder to an instructor that is falling behind in the report evaluation process. We have thought about giving this functionality to students as well, but we have decided that it could be a feature that could easily be abused. Moreover, the statistics page can only be seen by administrative assistants, summer training coordinators, and instructors. However, instructors can only see their own students, while administrative assistants and summer training coordinators have a more widespread view. Furthermore, since company forms are confidential, only administrative assistants will have full access to them. We will be using encryption to protect these forms. This ensures that only the students' assigned instructors will be able to access the form to enter their grades.

	Student	Instructor	Teaching Assistant	Summer Training Coordinator	Administrative Assistant	Admin
Login	X	X	X	X	X	X
Logout	X	X	X	X	X	X
View notifications	X	X	X	X	X	X
Make announcements to all users				X	X	
Make announcements to assigned users		X	X			
Change password	X	X	X	X	X	X
View reports' revision request	X					
View current status of a report	X	X				
View reports	X	X	X			
Edit/remove report	X					
Upload report with description	X					
Send revision request		X	X			
Assign due dates		X	X			
Fill/edit summer training grade form		X				
Upload e-signature		X				
Upload feedback		X	X			
View a student's current status		X		X	X	
View evaluation phases of grade forms		X		X	X	

Upload company evaluation form					X	
Enter grades for the company evaluation form		X				
View lists for all users				X	X	
View lists for assigned users		X	X			
View statistics for all students				X	X	
View statistics for assigned students		X				
Send reminders to instructors					X	
Assign/edit instructors and TAs to students					X	
Add/remove/edit all users					X	X
Add a data sheet for new users					X	X

Table 1: Access Control and Security

2.5. Boundary Conditions

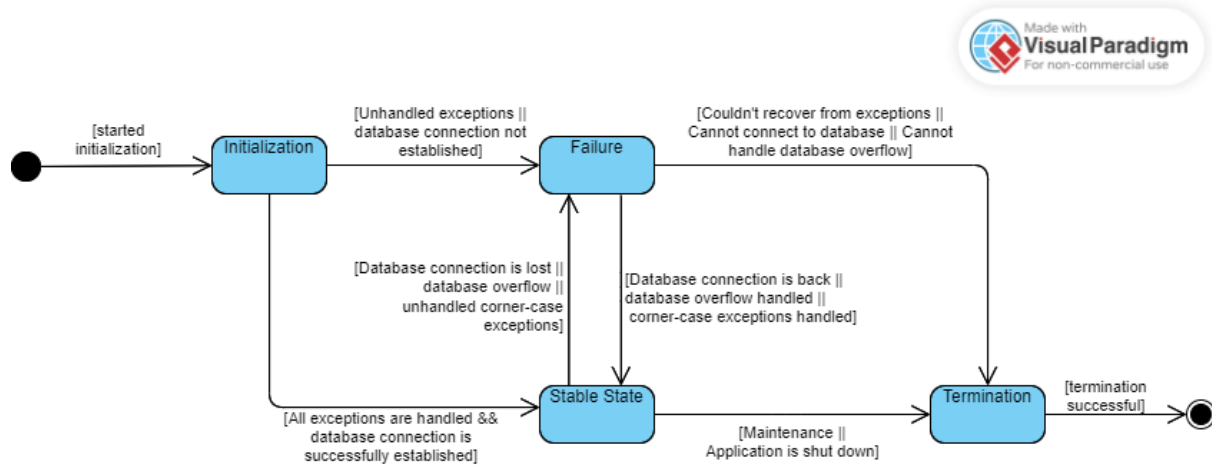


Figure 3: Boundary Conditions

2.5.1. Initialization

- The system source files need to execute without any uncaught errors.
- The backend server needs to be running on the designated port without any uncaught errors.
- All necessary packages and dependencies for both the frontend and backend need to be installed properly.
- The database needs to be ready and connected to the system for the system to be initialized.
- If there are unhandled exceptions such as syntax errors or missing dependencies, the system will shift to the failure state.
- If the database connection cannot be established or there are complications with the connection, the system will shift to the failure state.
- Else, if all exceptions are handled successfully and the database connection is established without issues, the system will transition to the stable state.

2.5.2. Stable State

- Once the system is successfully initialized with a functioning database connection and no uncaught errors, it enters the stable state.

- While the system is in the stable state, it will handle user requests and perform operations smoothly.
- If problems occur with the database, such as a lost connection or an overflow of data, or if there are unhandled exceptions from corner cases, (e.g., unexpected user inputs) the system will shift to the failure state.
- If the system transitions to the failure state due to the aforementioned issues, and those problems are resolved, the system can shift back to the stable state.
- Additionally, if there is maintenance scheduled or the application is being shut down intentionally, the system will shift to the termination state.

2.5.3. Termination

- Admins have the capability to schedule maintenance that will lead to the termination of the system.
- Prior to the scheduled maintenance, a notification will be sent to all users, indicating that the system will be under maintenance. The notification will include a countdown displayed within the application, notifying users of the time remaining until the maintenance begins.
- If any users are still logged in when the countdown ends, their data will be saved to the database, and they will be automatically logged out of the system.
- During the maintenance period, new user logins will not be allowed.
- Additionally, admins have the option to shut down the system completely, which will result in termination.
- Apart from scheduled maintenance or intentional shutdown by administrators, the system may also terminate due to unforeseen circumstances or errors.
- In the event of critical errors or system failures that cannot be resolved automatically, the system may terminate to prevent further damage or data corruption.
- Hardware or network failures, power outages, or other infrastructure issues may lead to an abrupt termination of the system.
- In these cases, the system will terminate immediately without prior notification to the users.
- If either the scheduled maintenance or the complete shutdown occurs, the system will terminate successfully.

- System data will be preserved in the database, allowing for later initializations after the termination.

2.5.4. Failure

- The failure state is reached if there are problems or unhandled exceptions from corner cases.

- Frontend side

Incompatibility between browsers and devices that may result in a non-functional interface. Failure to validate user inputs may result in invalid data to be submitted which leads to errors in data processing.

- Backend side

Inability to handle large inputs may result in performance degradation and system failures. If the user inputs are not validated properly it can lead to security breaches such as SQL injection and XSS attacks.

- Database side

Inconsistent data flow or the database reaching resource limits can prevent data from being stored properly.

- If failures occur, an informative message will be displayed to the users to notify them.
- Shortly after notifying users, the system will terminate.
- Users will see a pop-up message explaining the reason for the system crash and providing information on when the system is expected to be operational again.
- After the system becomes operational again, the last saved state of the users will be restored from the database, allowing them to resume their activities from where they left off.

3. Low-Level Design

3.1. Object Design Trade-offs

3.1.1. Usability vs Functionality

This application is made for the use of all academic staff that deals with summer training and students that are taking summer training courses. All users need to easily

learn how the application works. Crowding the application with extra features would make this learning process complicated, which damages the overall user experience. To prevent this, the usability goal was prioritized over functionality.

3.1.2. Automation vs Rapid Development

This application decreases the workload of users by reducing the need for repetitive tasks between academic staff and students. To make the procedures easier and faster, the “automation” design goal is prioritized over rapid development. As this goal is prioritized, the implementation of the related functionalities took more time than other functionalities. The main related functionality is: uploading reports to the system and downloading them from other user roles. For example, students upload summer training reports and instructors download these reports; instructors upload annotated feedback and students download them. As these file exchanges are the main reason for automation, they took longer time to implement than other functionalities in the system.

3.2 Final Object Design

<https://online.visual-paradigm.com/share.jsp?id=323434323338352d31#diagram:workspace=hylibbej&proj=1&id=1>

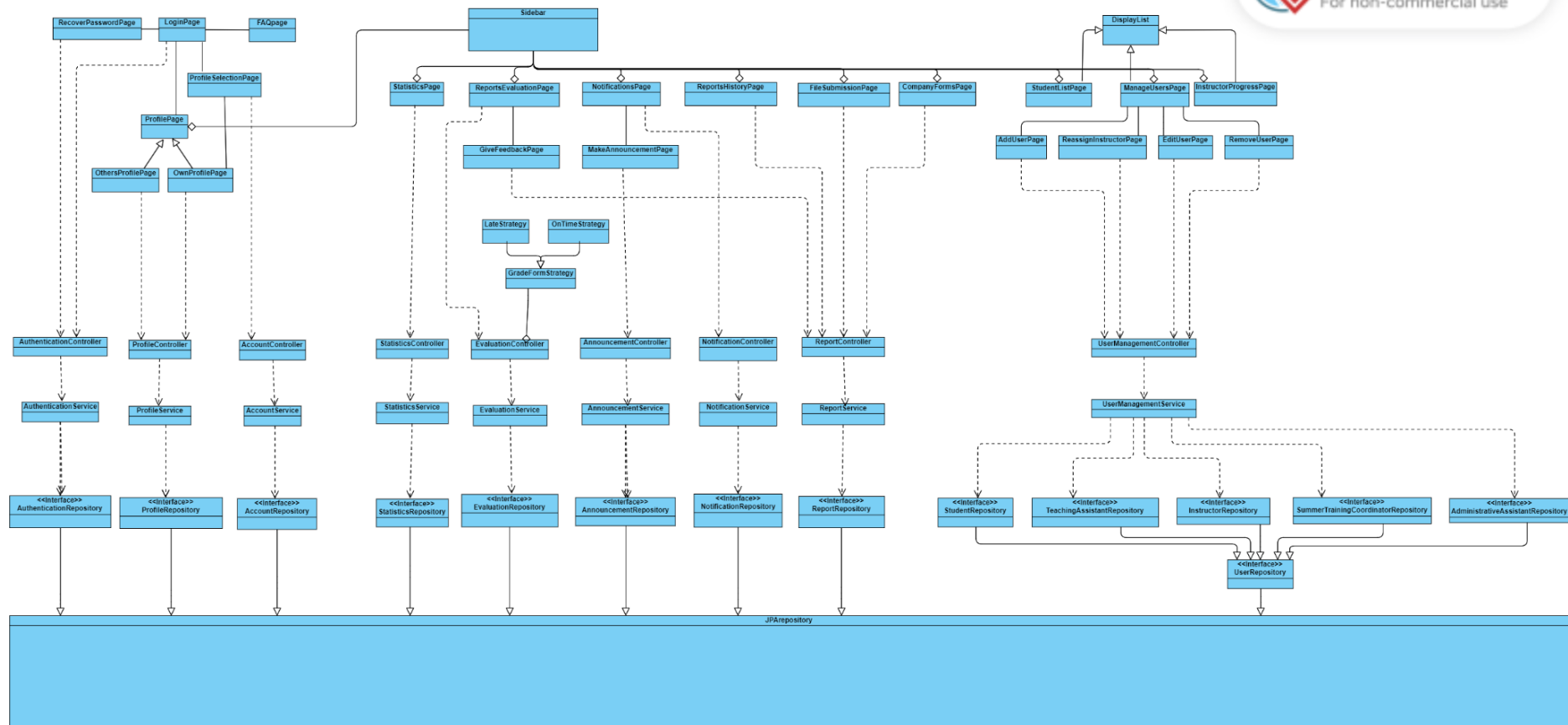


Figure 4: Final Object Class Diagram

3.3. Layers

3.3.1 User Interface Management Layer

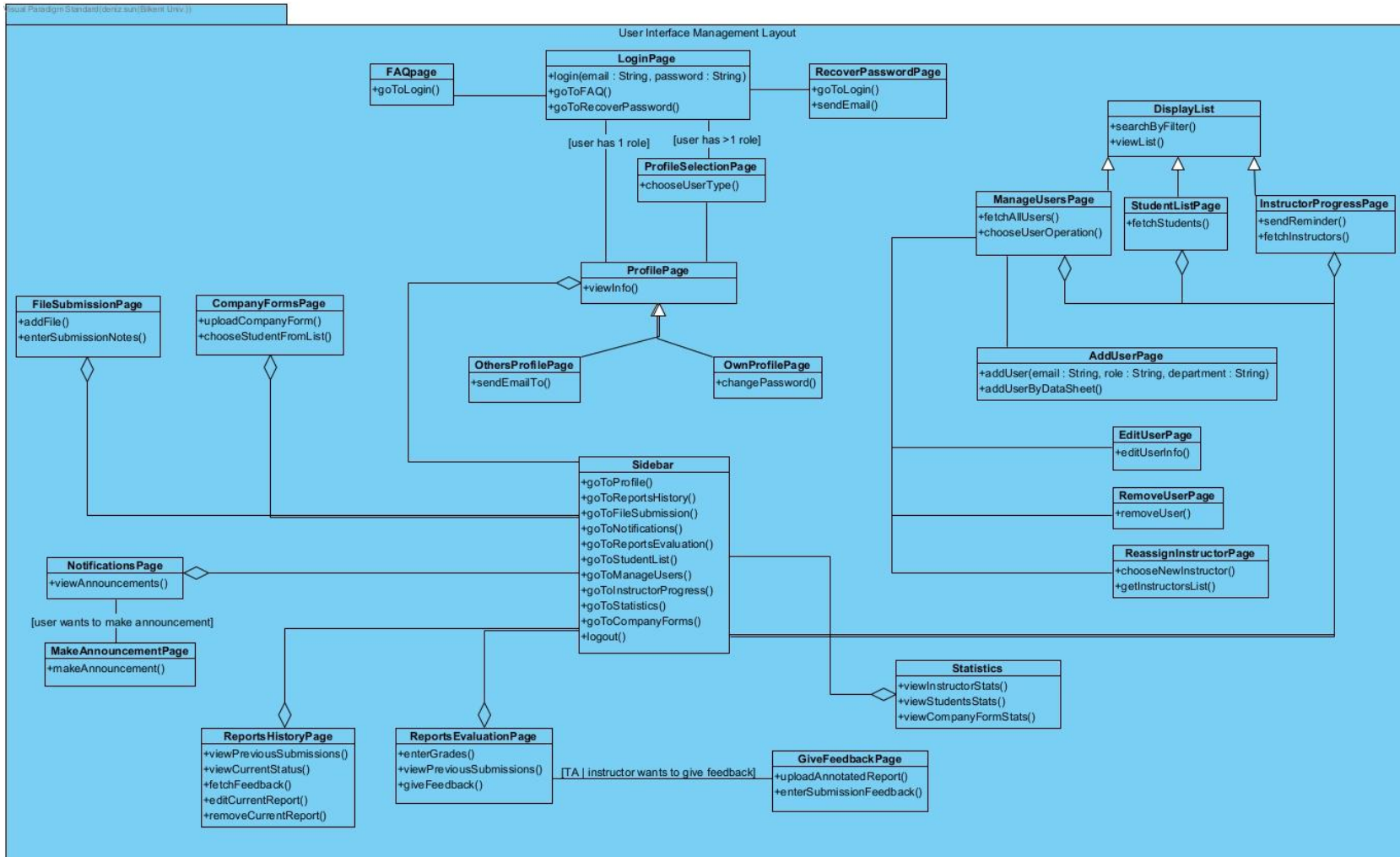


Figure 5: User Interface Management Layer Diagram

3.3.2 Web Server Layer

<https://online.visual-paradigm.com/share.jsp?id=323234343632302d3131#diagram:workspace=enuekdrh&proj=2&id=11>

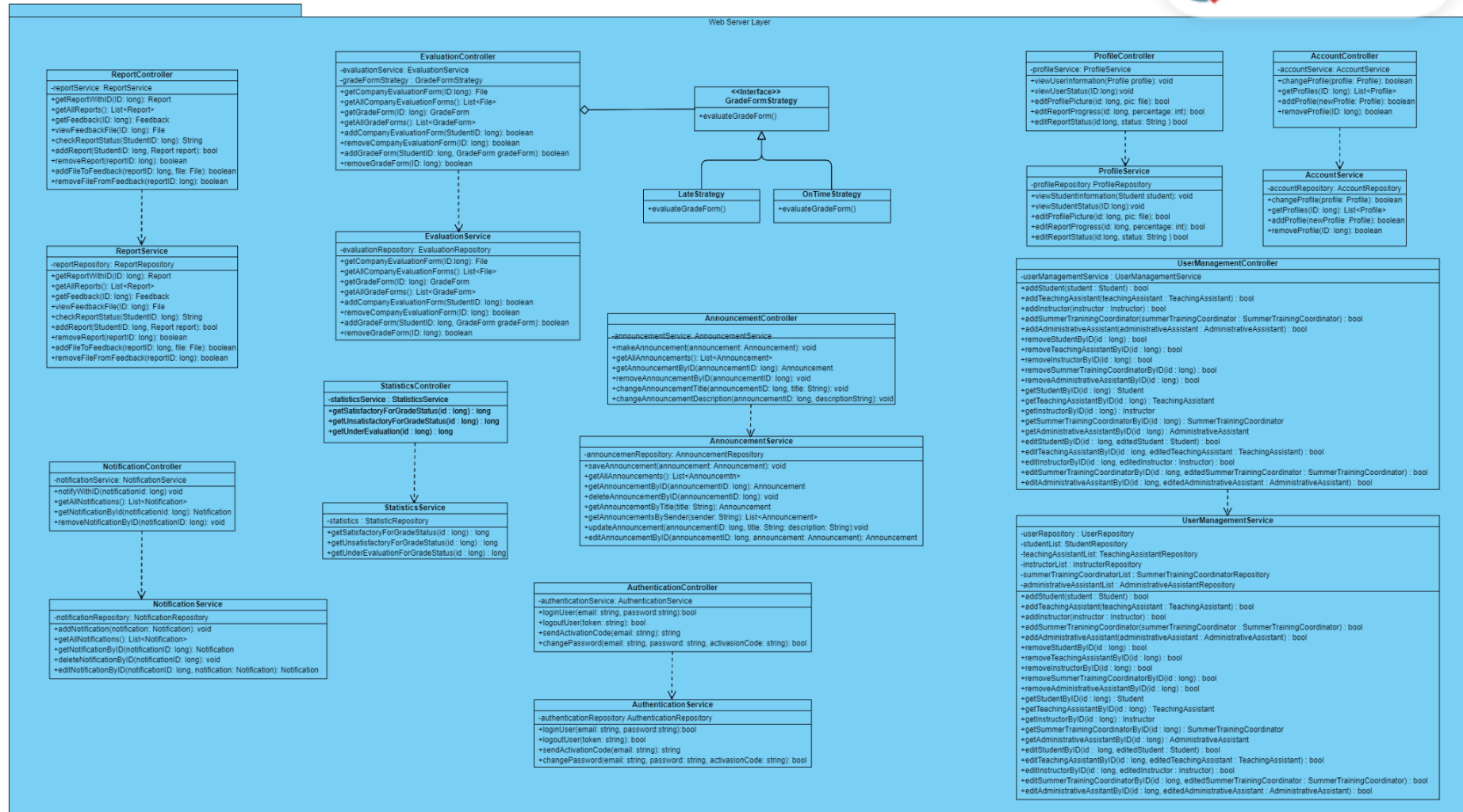


Figure 6: Web Server Layer Diagram

3.3.3 Data Management Layer

3.3.3.1 Entity

<https://online.visual-paradigm.com/share.jsp?id=323234343632302d39>

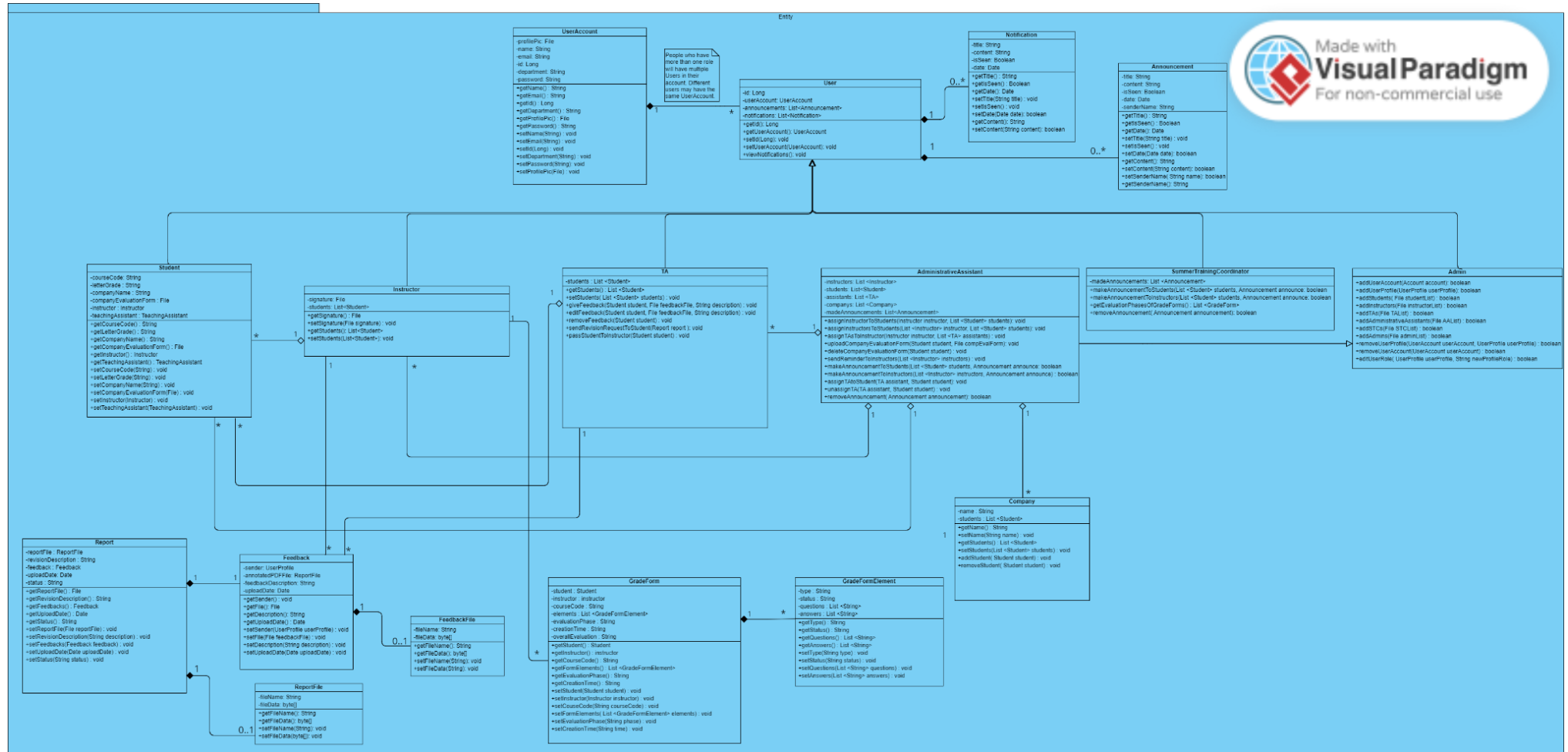


Figure 7: Entity Diagram

3.3.3.2 Database

<https://online.visual-paradigm.com/share.jsp?id=3234343234313>

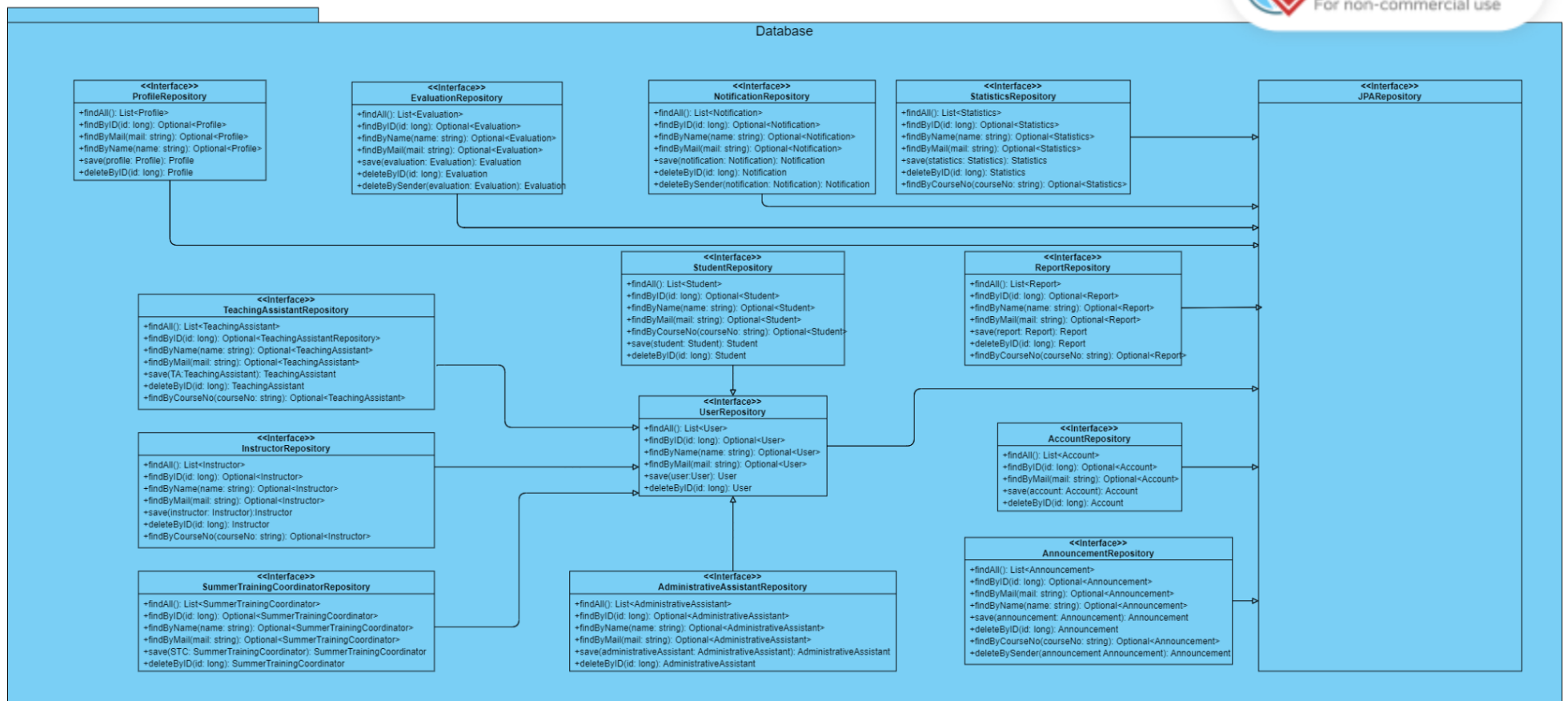


Figure 8: Database Diagram

3.4. Class Interfaces

3.4.1. User Interface Layer Class Interfaces

3.4.1.1. LoginPage

+login(email:String, password: String): On click, a request is sent to the web server in order to authenticate the user's credentials.

+goToFAQ(): Goes to the FAQ page. (Frequently Asked Questions)

+goToRecoverPassword(): On click, goes to the password recovery page.

3.4.1.2. FAQpage

+goToLogin(): On click, goes to the login page.

3.4.1.3. RecoverPasswordPage

+goToLogin(): On click, goes to the Login page.

+sendEmail(): On click, sends a verification email to the user's email.

3.4.1.4. UserTypePage

+chooseUserType(): Users with more than one role have to choose which account (which role) they are logging in to.

3.4.1.5. ProfilePage

+viewInfo(): Shows user information such as name, email, courses. Additionally, students can see their assigned instructor's name and email.

3.4.1.6. OthersProfilePage

+sendEmailTo(): Allows users to send an email to the person they are viewing the profile of.

3.4.1.7. OwnProfilePage

+changePassword(): Allows users to change their account password.

3.4.1.8. FileSubmissionPage

+addFile(): Allows users to upload summer training reports as a file to the system.

+enterSubmissionNotes(): Allows users to enter comments along with their file submission.

3.4.1.9. CompanyFormsPage

+uploadCompanyForm(): Allows administrative assistants to upload a student's company evaluation form to the system.

+chooseStudentFromList(): Allows administrative assistants to choose which student they are uploading the company evaluation form of.

3.4.1.10. NotificationsPage

+viewAnnouncements(): Allows users to view announcements.

3.4.1.11. MakeAnnouncementPage

+makeAnnouncement(): Allows administrative assistants, summer training coordinators and instructors to make announcements.

3.4.1.12. Sidebar

+goToProfile(): On click, goes to the user's Profile Page.

+goToReportsHistory(): On click, goes to the Reports History Page.

+goToFileSubmission(): On click, goes to the File Submission Page.

+goToNotifications(): On click, goes to the Notifications Page.

+goToReportsEvaluation(): On click, goes to the Reports Evaluation Page.

+goToStudentList(): On click, goes to the Student List Page.

+goToManageUsers(): On click, goes to the Manage Users Page.

+goToInstructorProgress(): On click, go to the Instructor Progress Page.

+logout(): On click, the user is logged out of the system.

+goToStatistics(): On click, goes to the Statistics Page.

+goToCompanyForms(): On click, goes to the Company Forms Page.

3.4.1.13. ReportsHistoryPage

+viewPreviousSubmissions(): Allows students to view their previous submissions/ revisions.

+viewCurrentStatus(): Allows students to view the status of their last submitted report, which is currently being graded.

+fetchFeedback(): Shows students feedback for their previous submissions/ revisions.

+editCurrentReport(): Allows students to edit their last submitted report.

+removeCurrentReport(): Allows students to remove their last submitted report.

3.4.1.14. ReportsEvaluationPage

+enterGrades(): Allows instructors to enter grades for each part of a student's summer training report.

+viewPreviousSubmissions(): Allows instructors to view a student's previous submission details.

+giveFeedback(): Allows instructors and TAs to give feedback on a student's current report by redirecting to the Give Feedback Page.

3.4.1.15. GiveFeedbackPage

+uploadAnnotatedReport(): Allows TAs or instructors to upload the annotated version (with feedback) of a student's report.

+enterSubmissionFeedback(): Allows TAs or instructors to add comments and notes about the feedback.

3.4.1.16. StatisticsPage

+viewInstructorStats(): Shows the grading progress of instructors and statistics.

+viewStudentsStats(): Shows the submission progress of students and statistics.

+viewCompanyFormStats(): Shows the statistics of uploaded company forms.

3.4.1.17. DisplayList

+searchByFilter(): Allows users to search for another user by filters such as name and department.

+viewList(): Shows users the list of users and the results of their search, if any.

3.4.1.18. ManageUsersPage

+fetchAllUsers(): Gets all the users registered in the system.

+chooseUserOperations(): On click, allows administrative assistants to choose an operation related to a user and redirects them to its specific page: Add User Page, Edit User Page, Remove User Page.

3.4.1.19. StudentListPage

+fetchStudents(): Gets all the students registered in the system.

3.4.1.20. InstructorProgressPage

+sendReminder(): Sends a notification to instructors to remind them to proceed with grading a student's report.

+fetchInstructors(): Gets all the instructors registered in the system.

3.4.1.21. AddUserPage

+addUser(email: String, role: String, department: String): Allows administrative assistants to add a user to the system by entering the user's email, role, and department.

+addUsersByDataSheet(): Allows administrative assistants to add multiple users by uploading a data sheet that stores the users.

3.4.1.22. EditUserPage

+editUserInfo(): Allows administrative assistants to edit a registered user's information.

3.4.1.23. RemoveUserPage

+removeUser(): Allows administrative assistants to remove a registered user from the system.

3.4.1.24. ReassignInstructorPage

+chooseNewInstructor(): Allows administrative assistants to change a student's assigned instructor by choosing a new one.

+getInstructorList(): Gets the instructor list for administrative assistants to choose from when reassigning a student's instructor.

3.5 Design Patterns

3.5.1 Decorator Pattern

This application will have a lot of file uploads and downloads. There will be three different documents corresponding to each student. First one is the company evaluation forms of students, the second one is the summer training reports and the third one is the final grade forms that instructors sign and submit. The company evaluation forms and student reports may have multiple versions (revisions). To ensure there is no confusion or overriding of documents, they need to be named according to a naming convention. This will be ensured by keeping the documents by student ID, document name and course code. Keeping track of the course code is crucial for students that are taking two summer training courses at once.

The administrative assistants choose which student they are uploading the company evaluation forms of. The system will rename the file to “StudentID_CEF_CourseCode_<iterNo>”. (CEF: Company Evaluation Form) When uploading the summer training reports, students can name them however they want. The system will rename it to “StudentID_Report_CourseCode_<iterNo>”. For both of these uploads, iterNo will keep track of the revisions. It will automatically increase if a new version of the file is uploaded for the same student. For the grade forms, the system will rename the file to “StudentID_GradeForm_CourseCode”.

3.5.2 Strategy Pattern

In Summer Training Management System, there will be a lot of approvals, rejections, and iteration requests for student reports. Also, the company evaluation form affects the process. Furthermore, there are cases where company evaluation forms are late. In order to manage all these situations well, we will use the Strategy Pattern. We will have 2 strategies. The first one is the algorithm for the early/on-time company evaluation forms, and the second one is for late company evaluation forms. These 2 strategies will help us break out from the evaluation process if a late company evaluation form is insufficient for the student to proceed with part B, which is the part where teaching assistants and instructors give feedback on summer training reports.

These 2 strategies will, because of the strategy design pattern’s nature, provide us with 2 different algorithms for the evaluation process of summer training grade forms of the students. Inner control points for each strategy are different. For example, in each stage of the strategy for late company evaluations, arrival and result of the company evaluation report will affect the flow of stages, But in early/on-time company evaluations, after part A, the system will no longer look for a company evaluation form.

3.5.3 Façade Pattern

To optimize the performance of our system, we implemented the Façade pattern as a data abstraction layer for the database. With frequent access to the database and the need for simultaneous regulation of user lists and report updates, accessing the entire database every time would be redundant. Thus, we simplified the interactions between the user and the system by encapsulating the database implementation behind a simple interface.

To maintain consistency and simplicity throughout the system, we also extended the Façade pattern to the web server layer, where we implemented structures called services. This allowed us to handle changes in system boundaries without affecting the functionality of the controllers.

4. References

1. Spring Boot Tutorials “Spring Boot Package Structure”,
<https://www.baeldung.com/spring-boot-package-structure>
2. Introduction to Project Lombok, <https://www.baeldung.com/intro-to-project-lombok>
3. Material UI, Overview, <https://mui.com/material-ui/getting-started/overview/>
4. Oracle Cloud Infrastructure Documentation, Managing a DB System
<https://docs.oracle.com/en-us/iaas/mysql-database/doc/managing-db-system.html>