CUMREP
Design Report

Yaşar Tatlıcıoğlu- 22003856
Sabri Eren Dağdelen- 22001764
Anıl Altuncu- 21901880
Ömer Fırat Bekiroğlu- 22002239
Yamaç Yiğit Ozan- 22003595
Emirhan Ay- 22203902

1.Introduction

1.1 Purpose of the System

CUMREP is a web application that aims to automate and coordinate the process of summer training evaluation system. Currently all the system proceeds through three different platforms(Moodle/Google Drive/Email). During the process, actors use various platforms and contact other actors through these platforms; it makes the process unnecessarily long and tiring. CUMREP offers to remove this tedious and longlasting system and enable actors to communicate and share documents on a single platform. CUMREP will provide direct convenience to students and evaluators and enable other actors to follow the process.Each actor can easily do their task in the process with CUMREP, possible confusions and mistakes will be minimized.

1.2 Design Goals

1.2.1 Usability

CUMREP's main purpose is to provide more straightforward and quicker access to files and communication between actors. CUMREP, offers an easily accessible and understandable user interface (UI) to actors. Removing this tiring cross-platform system, CUMREP will enable actors to perform the desired action with a minimum number of clicks (maximum 3), with a menu system that can be easily used and reached at the top corner.

1.2.2 Maintainability

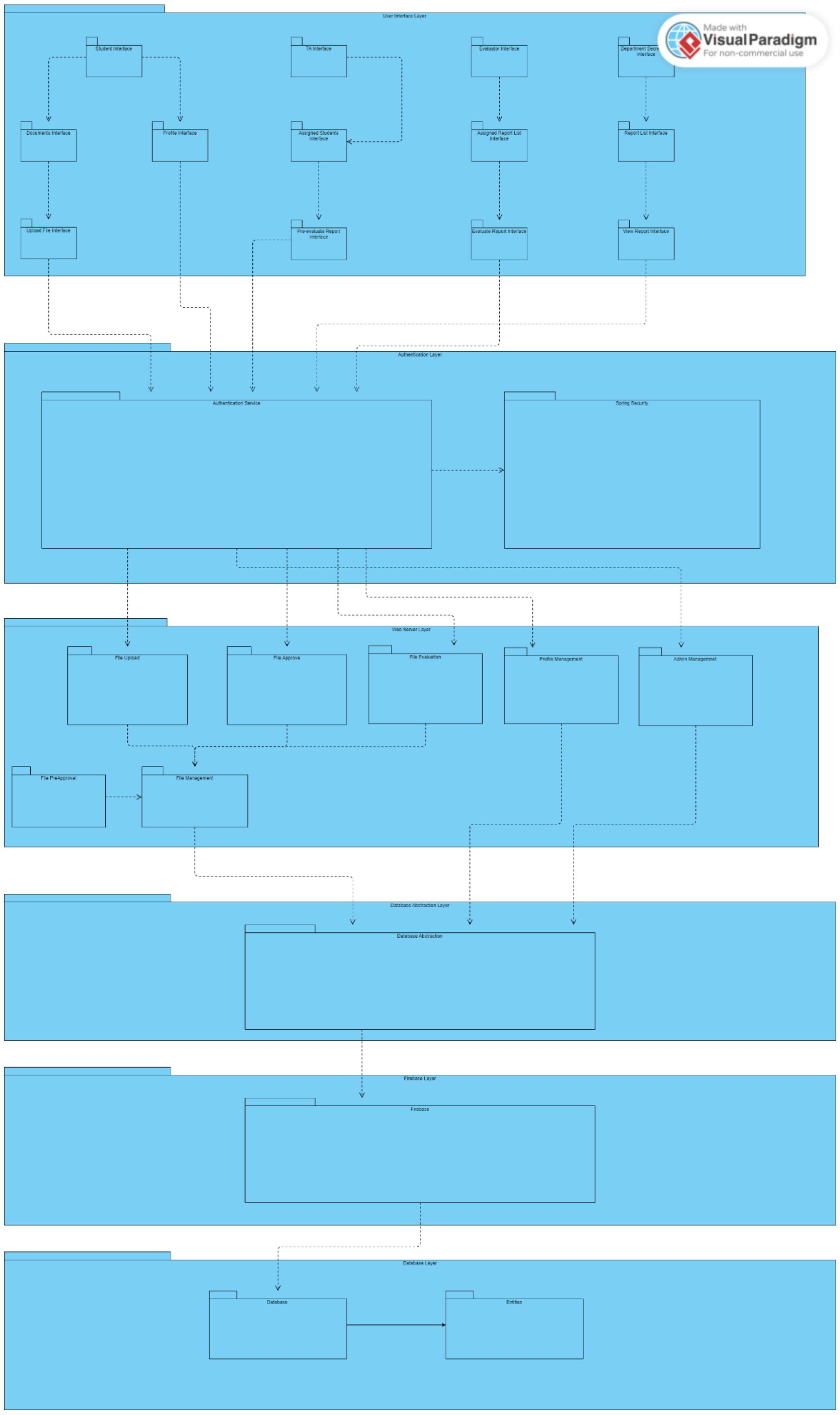
In Bilkent University like every course, CS 299/399 may alter over the years. How students report is evaluated, who will evaluate it, how many times it will be evaluated, how reports will be rated?.. may change over time and CUMREP is ready for this. As a result, the design of the software is created to ensure that properties can be modified quickly according to these criteria changes.

1.2.3 Reliability

Since internship files and reports are crucial for both students and the school, there shouldn't be any data loss, file upload issues, server problems etc. CUMREP, with the utilized servers and databases, such problems will not happen and it will be a reliable platform for actors. For web-servers uploading, downloading and forwarding files will be carried out meticulously. Also the application is traceable by actors so they can rely on it.

2. High Level Software Architecture

2.1 Subsystem Decomposition



2.2 Hardware/Software Mapping

We will use next.js which is a react framework for frontend purposes and for tailwind CSS for design. The advantages of these tools are Server-side rendering: Next.js offers server-side rendering (SSR) out of the box, which means the initial HTML is generated on the server and sent to the browser, leading to better performance and SEO optimization. Automatic code splitting: Next.js automatically splits your JavaScript code into smaller chunks, which are loaded only when needed, leading to faster page loading times. Built-in routing: Next.js has built-in routing, which makes it easy to create a multi-page application with a clean URL structure. Easy deployment: Next.js can be easily deployed to various platforms such as Vercel, AWS, and Google Cloud, and offers automatic deployment and scaling. Utility-first approach: Tailwind CSS is a utility-first CSS framework, which means that it provides a set of pre-defined classes that can be used to style HTML elements. This approach enables developers to create complex layouts and designs quickly and easily. Customizable: Tailwind CSS is highly customizable, with the ability to configure the framework to fit the specific needs of a project. Developers can add or remove classes, modify colors, and adjust other settings as needed. Low specificity: Tailwind CSS uses low specificity classes, which means that the framework does not override existing CSS styles easily. This makes it easy to integrate with existing projects and stylesheets. Mobile-first approach: Tailwind CSS is designed with a mobile-first approach, which means that it is optimized for small screens and mobile devices, leading to a better user experience on mobile devices.
We will use Firebase which includes Firestore, storage, and authentication services. The advantages of Firebase are,
Real-time database: Firebase provides a real-time database that enables developers to build real-time applications without worrying about managing the infrastructure. The database is hosted in the cloud and can be accessed from anywhere.
Authentication: Firebase provides authentication services that enable developers to easily add user authentication to their applications. This includes support for social logins like Google, Facebook, and Twitter.
Cloud storage: Firebase provides cloud storage that enables developers to store and serve user-generated content such as images, videos, and other files.
Analytics: Firebase provides powerful analytics tools that enable developers to track user behavior, monitor performance, and measure the effectiveness of their applications.
Crash reporting: Firebase provides crash reporting that helps developers identify and diagnose application crashes and errors.
Hosting: Firebase provides hosting that enables developers to easily deploy and host their applications on a global content delivery network (CDN).
Easy integration: Firebase can be easily integrated with other Google services such as Google Cloud Platform, Google Analytics, and Google Ads.
Scalability: Firebase is designed to scale automatically based on the needs of the application, making it easy to handle large volumes of traffic and data.

2.3 Persistent Data Management

CUMREP utilizes Firebase as its backend to effectively implement Persistent Data Management. Firebase's real-time database and cloud storage services are used to store and retrieve data, and to store user-generated content such as files. The data is structured into collections and documents, and accessed through Firebase's API. Firebase's security measures ensure that only authorized users have access to the data, while its scalability features allow the website to handle high traffic and data volumes. Overall, Firebase's real-time database and cloud storage services enable the website to implement a robust and reliable Persistent Data Management system.

2.4 Access Control And Security

CUMREP implements robust security and access control mechanisms to ensure the confidentiality, integrity, and availability of its data. Firebase's built-in security rules are utilized to restrict access to the data based on user roles and permissions. Access control is enforced at the database level, and Firebase's authentication service is used to verify the identity of users. Additionally, the website utilizes HTTPS to encrypt data in transit, preventing unauthorized access. The website also implements various security best practices such as password hashing, input validation, and protection against common web vulnerabilities These measures ensure that the website remains secure and protected against unauthorized access and attacks.

2.5 Global Control Flow

The user has a unique profile and role managed by Firebase's authentication service, enforcing access control at the database level. The website's design and layout are implemented with Tailwind CSS and React, enabling a responsive and intuitive user experience. Client-side routing enables smooth navigation between pages without requiring a server request. Overall, the global control flow ensures that each user can interact with the website efficiently and securely without interfering with other users' experiences.

2.6 Boundaries

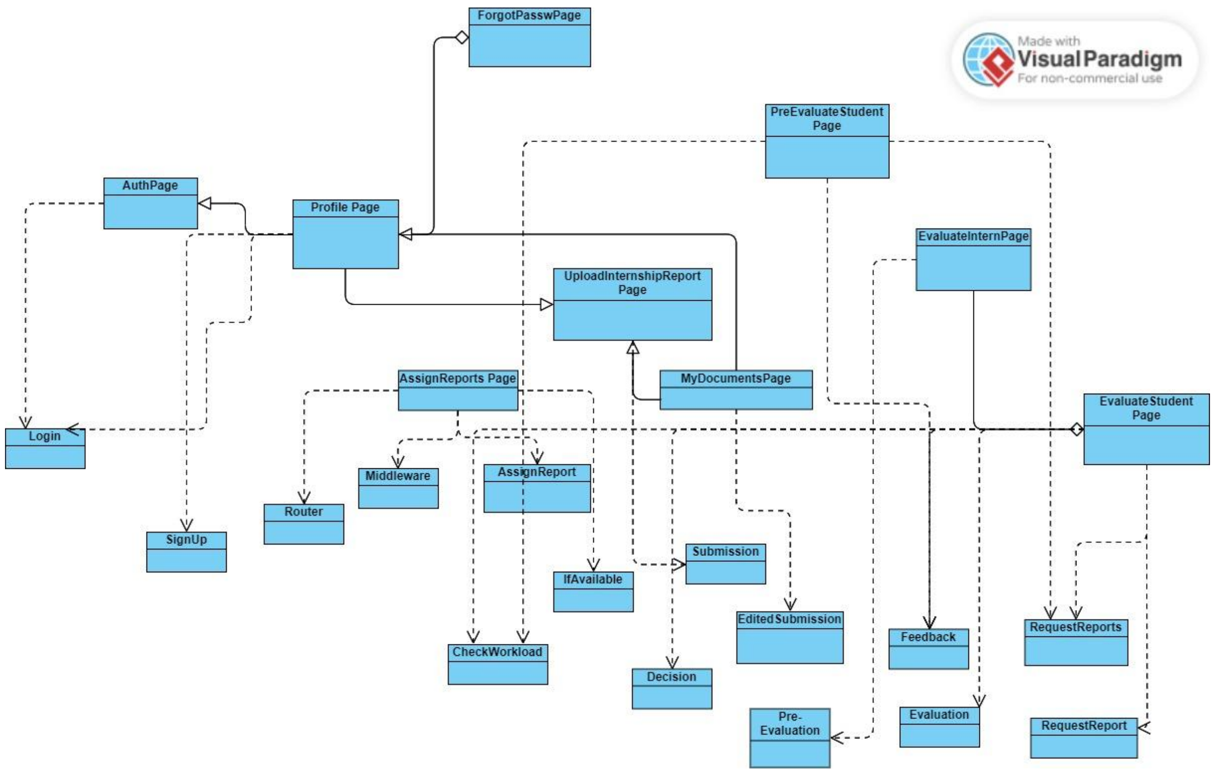
Boundaries are important in ensuring that CUMREP functions reliably and securely, both during initialization and termination, as well as in the event of failure. During initialization, the website is configured to ensure that all dependencies, such as the Firebase backend and Tailwind CSS and React libraries, are properly loaded and integrated. Proper initialization ensures that the website is fully functional and secure from the start. Similarly, termination is carefully managed to ensure that all data is saved, all sessions are closed, and all resources are freed up to avoid any data loss or security vulnerabilities. In the event of a failure, the website is designed to respond gracefully and securely. Firebase's real-time database and cloud storage services, as well as its automatic scaling capabilities, ensure that the website can handle high volumes of traffic and data without crashing or losing data. Overall, CUMREP implements robust boundaries to ensure reliable and secure operation, even in the face of challenges such as initialization, termination, and failure.

3. Low Level Design

3.1 Object Design Trade-offs

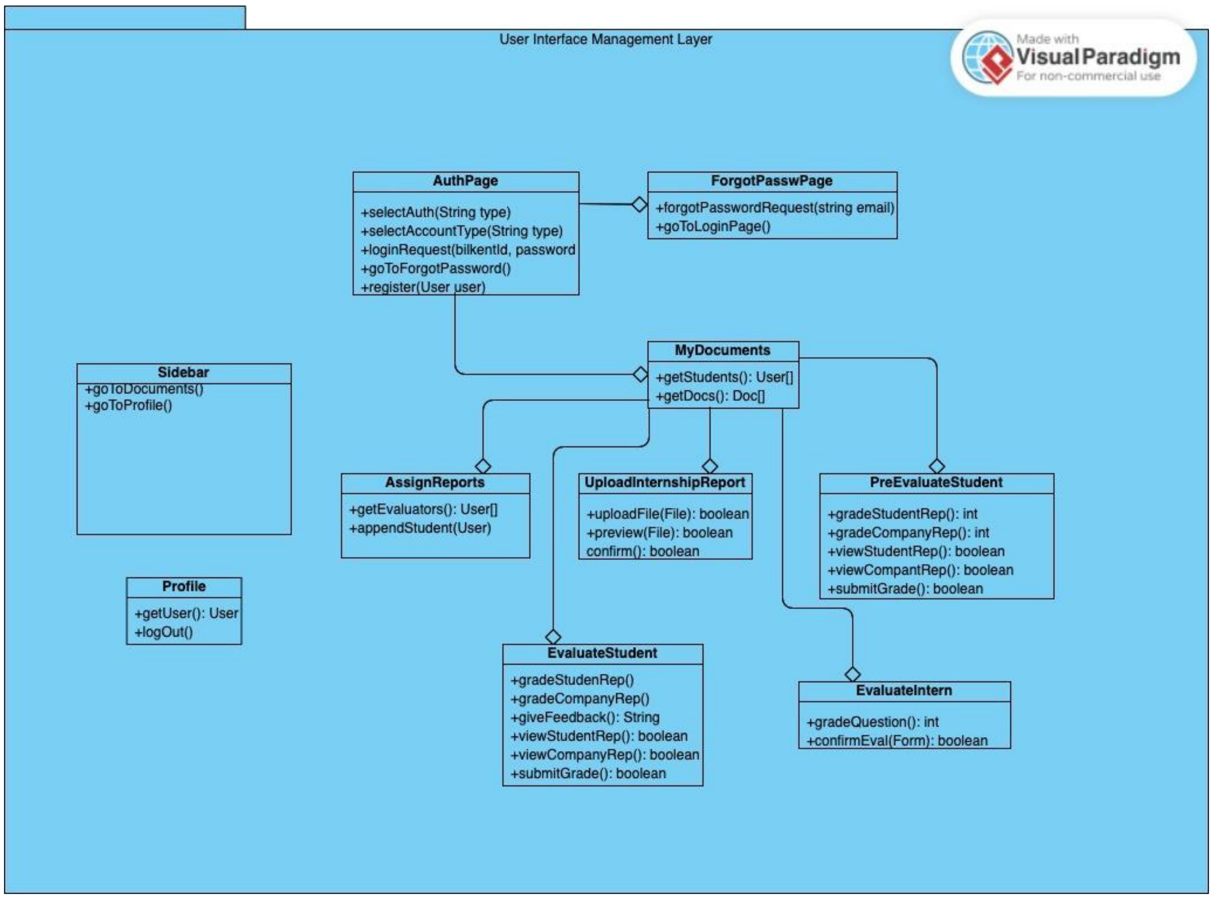
- Usability versus maintainability: When prioritizing usability, you may be tempted to implement features quickly and in a way that is easy for users to understand. However, this may make the code harder to maintain over time. On the other hand, when prioritizing maintainability, you may need to spend more time organizing and documenting your code, which can make the implementation slower and less user-friendly.
- Maintainability versus reliability: When prioritizing maintainability, you may be tempted to break up complex code into smaller, more modular pieces. However, this can make it harder to ensure that the code is reliable, since there may be more points of failure. Conversely, when prioritizing reliability, you may need to write more complex code that is harder to modify or maintain over time.
- Usability versus reliability: When prioritizing usability, you may be tempted to build a system that is easy for users to understand and use. However, this may make the system less reliable, since users may not always use the system in the way it was intended. Conversely, when prioritizing reliability, you may need to limit user options or add error handling mechanisms, which can make the system harder to use.

3.2 Final Object Design

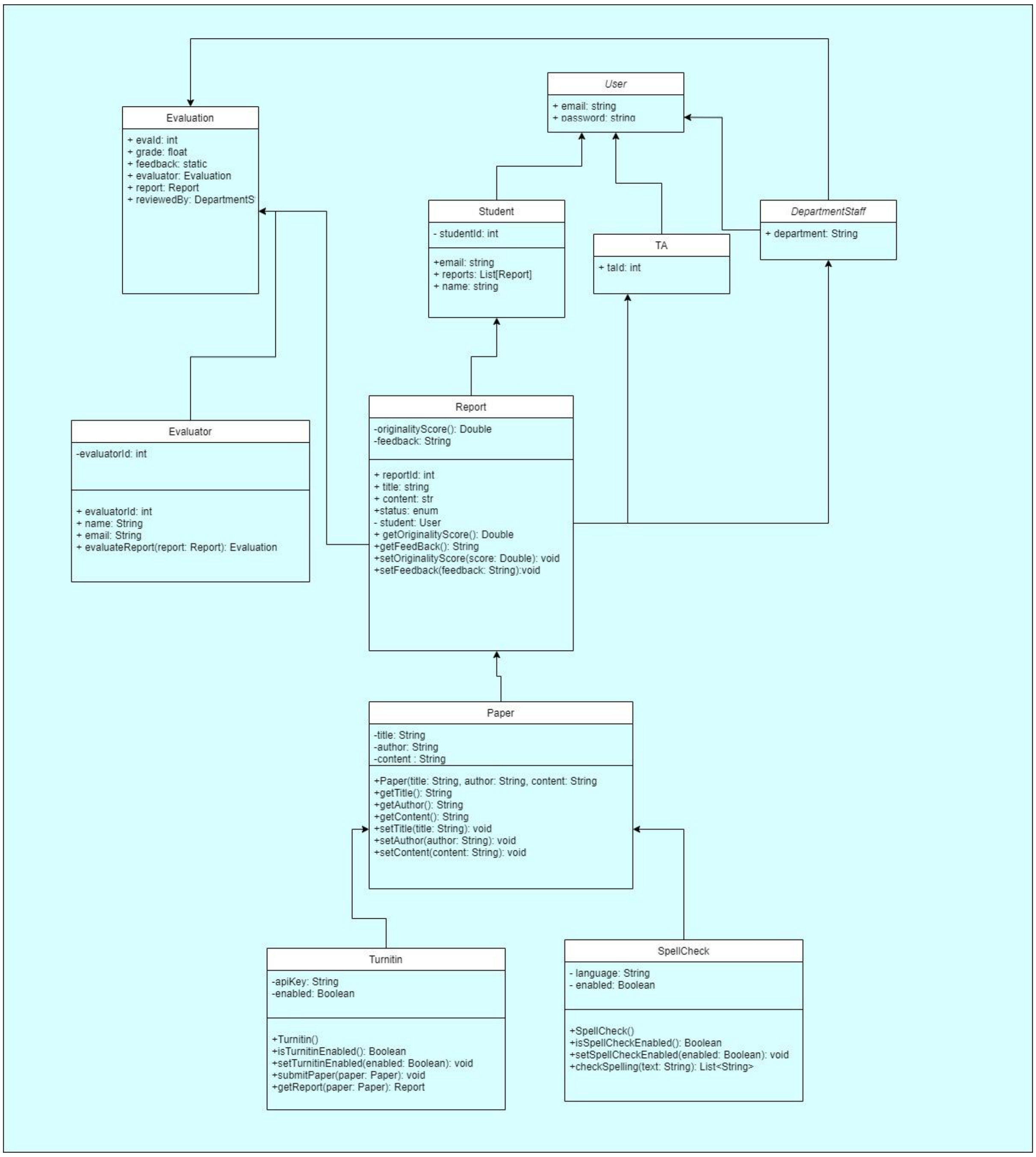


3.3. Layers

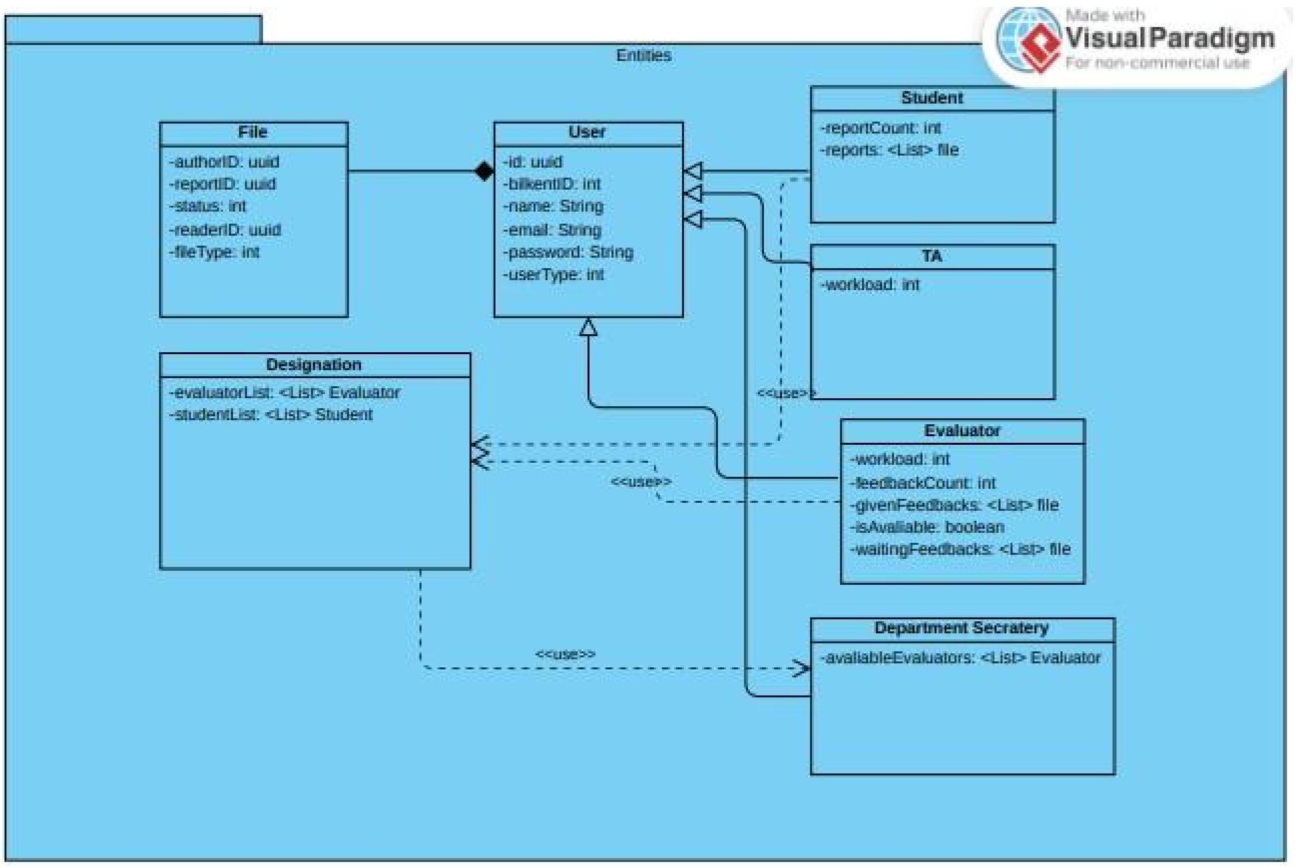
3.3.1 User Interface Management Layer

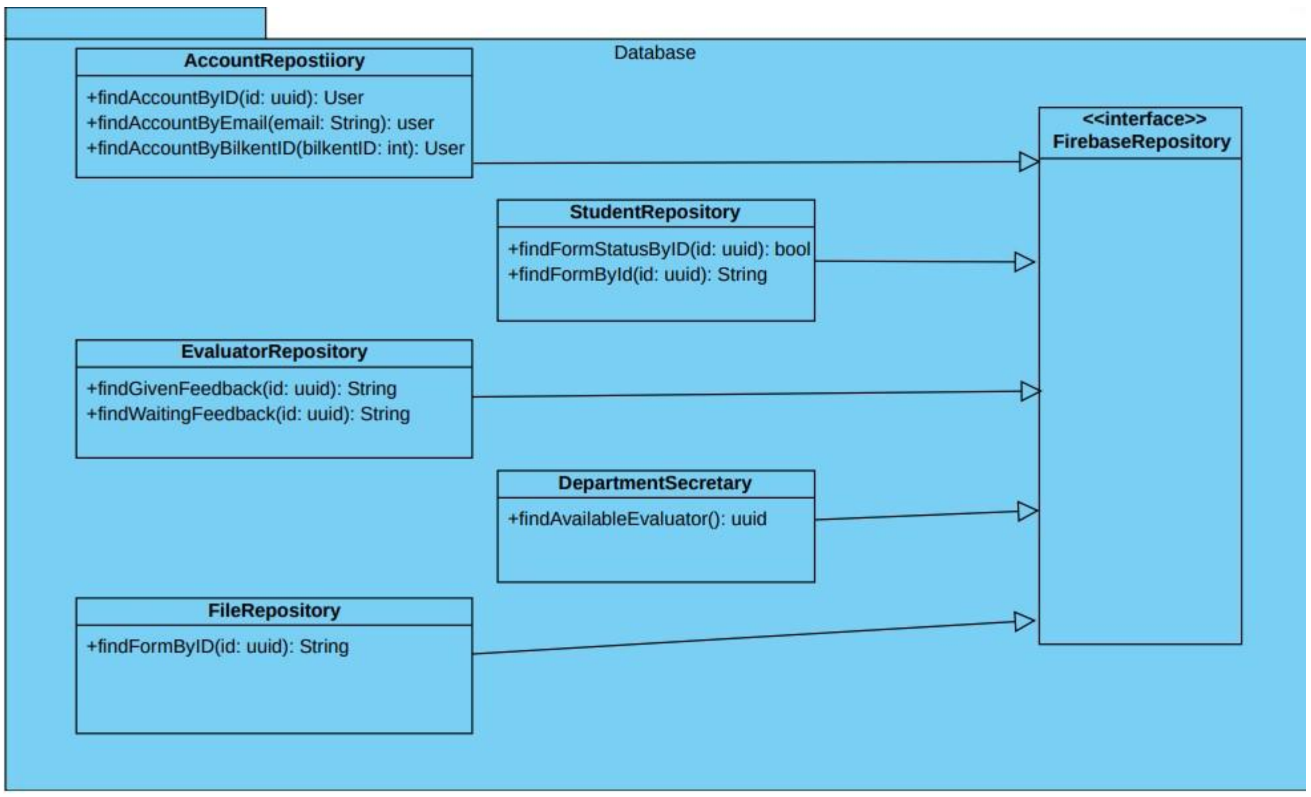


3.3.2 Web Server Layer



3.3.3 Data Management Layer





Database
AccountRepostiiory
+findAccountByID(id: uuid): User
+findAccountByEmail(email: String): user
+findAccountByBilkentID(bilkentID: int): User
<<interface>> FirebaseRepository
StudentRepository
+findFormStatusByID(id: uuid): bool
+findFormById(id: uuid): String
EvaluatorRepository
+findGivenFeedback(id: uuid): String
+findWaitingFeedback(id: uuid): String
DepartmentSecretary
+findAvailableEvaluator(): uuid
FileRepository
+findFormByID(id: uuid): String

3.4 Class Interfaces

3.4.1 User Interface Layer Class Interfaces

3.4.1.1 AuthPage

public selectAuth(String type): On click select whether display register or login component.

public selectAccountType(String type): Select the account type

public loginRequest(String bilkentId, String password): On click, sends a request to the webserver to verify user credentials.
public goToForgotPassword(): Goes to the forgot password page on click

public register(User user): On click, registers the user with provided fields

3.4.1.2 ForgotPasswPage

public forgotPasswordRequest(String email): On click, sends a verification email to the user's email
public goToLoginPage(): On click, goes to the login page

3.4.1.3 MyDocumentsPage

public User[] getStudents(): retrieves the list of students to grade/feedback/assign evaluator

public Doc[] getDocs(): retrieves list of documents to grade or previously graded

3.4.1.4 UploadInternshipReport Page

public boolean uploadFile(PDF: File): Uploads the PDF document of internship report of student

public boolean preview(File): Shows the preview of uploaded document.

public boolean confirm(): Confirms the uploaded document

3.4.1.5 EvaluateInternPage:

public int gradeQuestion(): Grades the question about intern out of 5.

public boolean confirmEval(Form): Confirms the evaluation form and uploads it

3.4.1.6 PreEvaluateStudent Page:

public int gradeStudentRep(): Grades the student report out of 10.

public int gradeCompanyRep(): Grades the company report out of 10.

public boolean viewStudentRep(): Opens the student’s report.

public boolean viewCompanyRep(): Opens the companie’s report.

public boolean submitGrade(): Submits the overall grade to the system.

3.4.1.7 Profile Page

public User getUser(): retrieves the user information to display details.

public logOut(): Logs out from the account and forwards to auth page

3.4.1.8 AssignReports Page

public Users[] getEvaluators(): retrieves the list of evaluators the append the student

public boolean appendStudent(User): append the student to the selected evaluator.

3.4.1.9 EvaluateStudent Page

public int gradeStudentRep(): Grades the student report out of 10.

public int gradeCompanyRep(): Grades the company report out of 10.

public String giveFeedback(): Retrieves the feedback written for student

public boolean viewStudentRep(): Opens the student’s report.

public boolean viewCompanyRep(): Opens the companie’s report.

public boolean submitGrade(): Submits the overall grade to the system.

3.4.2. Web Server Layer Class Interfaces

Login

- handle_login_request(request) - This method would handle incoming login requests and receive email and password information. It would then pass this information to the generate_login_response for processing.
- generate_login_response(email, password) - This method would generate a response confirming the user's login and setting a session cookie for subsequent requests. If the username and password are invalid, it would generate a response with an appropriate error message and display a pop-up message on the client side.

Signup

- handle_signup_request(request) - This method would handle incoming signup requests and extract relevant information from the request headers and body, such as the new user's bilkentId, email, account type and password. It would then pass this information to the appropriate generate_signup_response for processing.
- generate_signup_response(bilkentId, email, password, accountType) - This method would generate a response confirming the successful creation of a new user account. If any of the inputs is invalid, An error message will be generated and a pop-up message will be displayed to the client. Also a verification mail will be sent to the user.

Submission

- handle_submission_request(request) - This method would handle incoming report submissions from student accounts. It would then pass this information to the appropriate generate_submission_response for processing.
- Method: generate_submission_response() - This method would generate a response confirming the successful submission of the report.

Pre-Evaluation

- handle_preEvaluation_request(request) - This method would handle the beginning of the pre-evaluation reports from the TA accounts. It would pass the acceptance situation to the generate_preEvaluation_response.
- generate_preEvaluation_response(reportStatus) - This method would generate a response passing the reports to the evaluation if reportStatus is “accepted”.

Edited Submission

- handle_editedSubmission_request(request) - This method would handle incoming edited submissions from student’s whose report receives a pre-evaluation or evaluation.
- generate_editedSubmission_response() - This method would generate a response confirming the successful edited submission of the report.

Evaluation

- handle_evaluation_request(request) - This method would handle incoming evaluations from evaluators. It would send the status of the report to the generate_evalution_response.
- generate_evaluation_response(reportStatus) - This method would generate a message and an informing interface to the student.

Feedback

- handle_feedback_request(request) - This method would handle incoming feedback requests. It would then pass this information to the generate_feedback_response for processing.
- generate_feedback_response() - This method would generate a response confirming the successful submission of feedback.

ifAvailable

- handle_ifAvailable_request(request) - When a TA has no assessment, it sends a request to generate_ifAvailable_response in order to make TA's status available
- generate_ifAvailable_response(taAccount) - It receives the account of the Ta and sends this account to the list of available Teaching Assistants.

Assign Report

- handle_assignReport_request(request) - This method would handle the detection of the possible assignments of reports to the Ta's and send requests to the generate_assignReport_response.
- generate_assignReport_response(taAccount, report) - It receives the account of the TA and the report, then it assigns the report to the teacher. Necessary information about the report is sent to the TA.

Check Workload

- handle_checkWorkload_request(request) - This method handles the checking workload request by either the TA's and evaluators and sends the request to the generate_checkWorkload_response.
- generate_checkWorkload_response(taAccount/ evaluatorAccount) - This method generates an output showing all workloads of the TA's and evaluators in a sorted way. Appropriate information is sent back to the accounts making the request.

Request Reports

- handle_requestReport_request(request) - This method handles the incoming form requests made by the evaluators. It sends the information of the student to the generate_requestReport_response.
- generate_requestReport_response(studentAccount) - This method sends a notification/ message to the students who are asked to send a form by an evaluator. Necessary messages and an email will be sent.

Decision

- handle_decision_request(request) - This method will handle the incoming reports that are accepted by both a TA and an evaluator. It will send necessary information about the student, reports, forms and TA's and evaluators that accepted the reports to the generate_decision_response.
- generate_decision_response(studentAccount, taAccount, evaluatorAccount) - This method generates a response according to the decision of the department secretary. Appropriate feedback that indicates the acceptance of the internship report and form will be sent to the student.

Router

- map_request_to_handler(request) - This method would map incoming requests to the appropriate RequestHandler object based on the URL path and HTTP method.

Middleware

- authenticate_user(request) - This method would intercept incoming requests and check if the user is authenticated by checking for a valid session cookie. If the user is not authenticated, it would generate an error response and block the request.
- apply_csrf_protection(request) - This method would intercept incoming requests and apply CSRF protection by checking the authenticity of any POST or PUT requests that modify data. If the request fails the CSRF check, it would generate an error response and block the request.
- display_error_message(message) - This method would generate an error response with the specified message and display a pop-up message on the client side.

3.4.3. Data Management Layer Class Interfaces

3.4.3.1. ProfileRepository

Operations:

findAccountByID(id: uuid): Finds user by given uuid.

findAccountByEmail(email: String): Finds user by given email.

findAccountByBilkentID(bilkentID: int): Finds user by given bilkent id.

3.4.3.2. EvaluatorRepository

Operations:

findGivenFeedback(id: uuid): Finds uploaded feedback file by given uuid.

findWaitingFeedback(id: uuid): Finds uploaded file which is waiting to be evaluate by given uuid.

3.4.3.3 StudentRepository

Operations:

findFormStatusByID(id: uuid): Finds the status of the specific form by given uuid.

findFormById(id: uuid): Finds the specific form by given uuid.

3.4.3.4 DepartmentSecretaryRepository

Operations:

findAvailableEvaluator(): Finds the first available evaluator in the list.

3.4.3.5 FileRepository

Operations:

findFormByID(id: uuid): Finds specific form by given uuid.

3.5 Design Patterns

3.5.1 Decorator Pattern

CUMREP is a platform where a lot of files are collected and forwarded. Students upload their internship reports to CUMREP. Respectively, TA's, department secretary, evaluators will be able to download and view these reports. A report with feedback will be sent from the evaluators to the students through CUMREP. Students will upload a revised internship report again. The number of uploaded and downloaded reports may vary. We will use the decorator pattern to organize and name these reports accordingly. Each internship report uploaded in .pdf format will be renamed regardless of the name given by students. Names consisting of student number, iteration number (how many times the file got feedback) and date will be given respectively. Thanks to the decorator pattern, the name of the file will become "<Iteration Count>_<Student ID>_<Date>" after uploading. In addition, the files with feedback sent to the students will be arranged according to this format.

3.5.2 Strategy Pattern

Internship evaluation process depends on various evaluations and approvals or rejections. Basically evaluators evaluate internship reports and grade them accordingly. This has two options: approved and graded or rejected. When it is approved, the student part of the process finishes and the department secretary's task starts. On the other hand, students should upload revised reports and the same process goes on again. Before this process there is a precheck stage which is done by TA's. So TA's and evaluators have the ability to accept or decline the student reports at different stages. By using a strategy design pattern we will maintain these checking and grading stages easily. Also we can categorize these approval methods with a strategy design pattern in terms of just approving functionality(TA's) or both approving followed by grading (evaluators).

4. Glossary

Internship Report: Reports uploaded by students for evaluation for the first time.
Revised Report: Reports uploaded by students after one or more evaluations.

5. References

- Visual Paradigm
  https://www.visual-paradigm.com/