Bilkent University

Department of Computer Science

CS 319: Object-Oriented Software Engineering

INTERNSHIP MANAGEMENT SYSTEM

Design Report

5 May 2023

RIGEL

Aytekin İsmail, 22003988

Ece Ateş, 22002908

İzgi Nur Tamcı, 22002682

Ömer Asım Doğan, 21903042

Zeynep Begüm Kara, 22003880

1. Introduction

This report introduces Rigel software for Bilkent Engineering Department internship management by explaining its purpose and design goals, providing its high-level software architecture and low-level design components. The low-level design will provide packages, class interfaces, and design patterns. High-level software architecture consists of subsystem decomposition, hardware/software mapping, persistent data management, access control, and boundary conditions. This report concludes with a glossary and references.

1.1 Purpose of the System

Rigel software aims to create an online platform for internship report management for Bilkent Engineering Department. Previously, this process was done mainly by email. The instructors had to download student reports and give feedback about them from their computers. Rigel offers a web application to handle reports without downloading, enabling actors to complete tasks and track their progress through a single application. The expected users are students who take x299 and x399 courses, instructors, department chair, department secretary, and admin for technical maintenance purposes. The system will also provide announcements, instructor statistics, and student-instructor assignments aside from its report management functionalities.

1.2 Design Goals

Non-functional requirements for this application are narrowed down due to limited development time and budget. According to the application domain and the preliminary research, the main design goals for this application are usability and functionality.

1.2.1 Usability

Rigel software is designed with minimal web pages to handle all desired functionalities. The application aims to guide users to their desired task webpage within 4 clicks. UI (user interface) is intuitive to its actors as the basis for the design is the STARS system. Each actor will have access to all of their functionalities from their main page either from the navigation bar or drop-down menu and they can familiarize themselves with the system within 10 minutes.

1.2.2 Functionality

Rigel software aims to offer report management from the first report submitted to the instructor’s approval of the final report. Therefore, multiple actors will take part during the different stages of this process. For instance, the department secretary will be able to enter company grades into the system so that instructors will be able to start Part B of the report evaluation form. The organization of these multi-actor tasks will be handled by the system. It will also provide automatically generated forms for the instructors and they can save their progress in these forms. There are a number of novel features such as displaying the statistics of the instructors, requesting and granting extensions, and giving feedback without downloading reports within the web application.

2. High-Level Software Architecture

2.1 Subsystem Decomposition

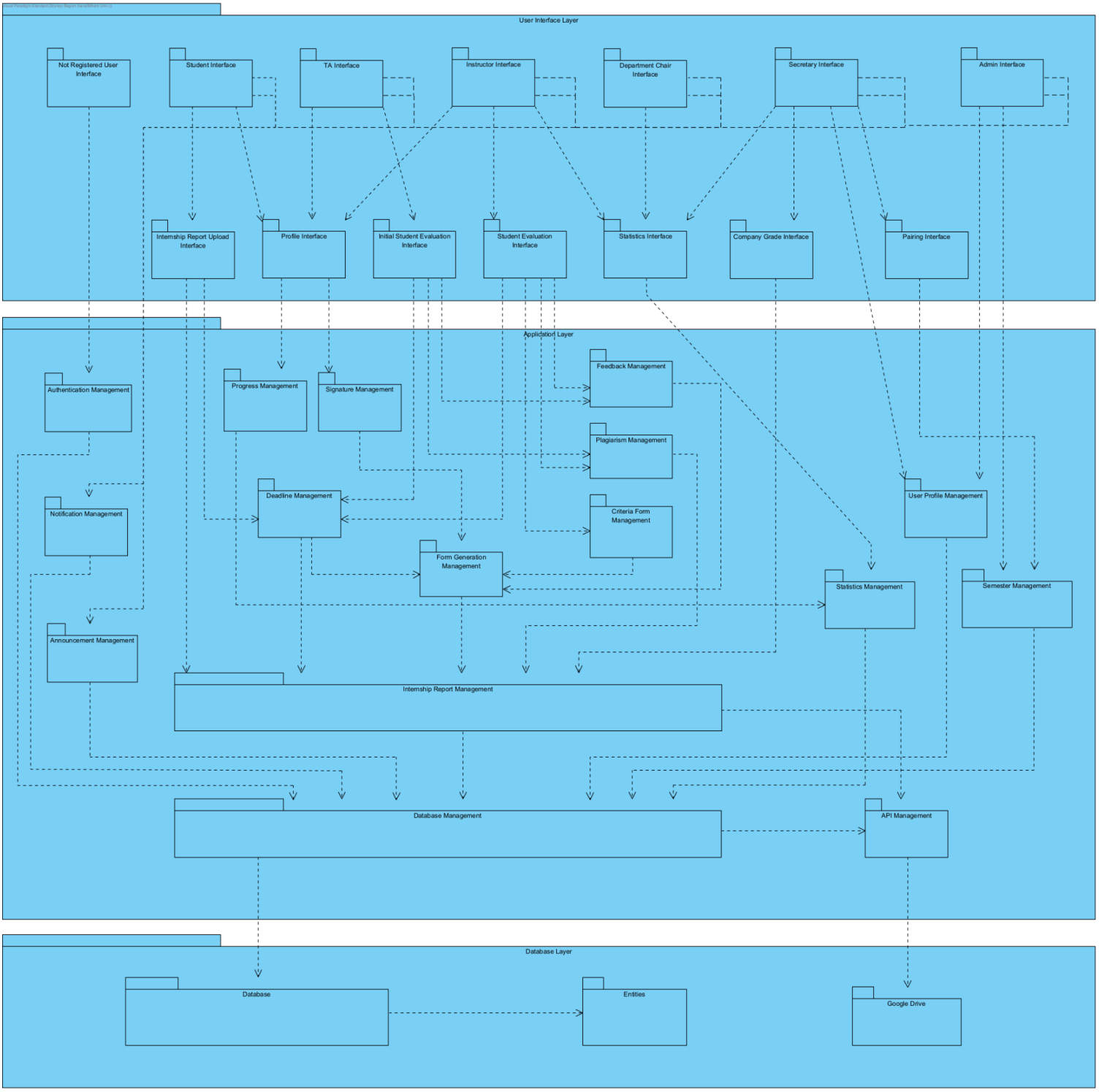


Diagram link: https://imgur.com/T3QoMov

2.2 Hardware / Software Mapping

We will use React.js with Javascript for the front end of our project. We will also use HTML5 and CSS3. Considering that modern browsers support all of the mentioned software, it can be said that our project will only need a stable modern browser, which is minimal demand for a website application.

We expect about 1.000 students to use our system in one semester. Considering everyone else included in this process, it is safe to assume less than 2.000 people will use our website in total, but this number decreases to about 100 users if we consider the number of users that will use the website concurrently [1]. For these reasons, in our backend, we will use Java, Spring Boot, MongoDB, and Google Drive API. Google Drive can sustain up to 50.000 requests per project per day and 10 queries per second per IP address, which is quite enough for the requirements of our program, where most users won't use it daily (except maybe Instructors who will review and grade student reports) [2]. Considering our program's simple requirements, we've decided that the Intel Core i5, or AMD Ryzen processor, which are some of the most common processors currently, would be enough and well-equipped to run our application. Since these processors are already highly advanced, they can easily handle the website even if our application's requirements increase in the future. Otherwise, our website does not require any additional hardware. We believe that it can be run on both computers and mobile phones (that can run a modern browser) without any problems.

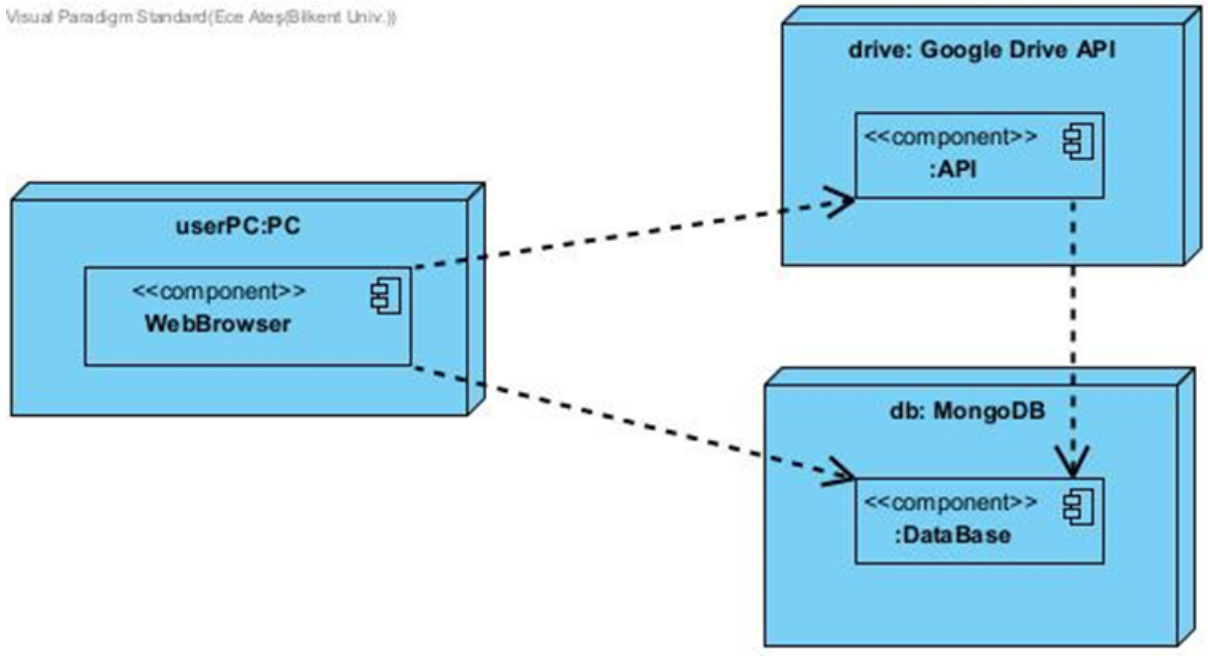


Figure: A diagram showing the relations between the program's hardware and software components.

As can be seen in the above diagram, our software interacts with both Google Drive API services and the MongoDB database. Also, Google Drive API will use MongoDB to reach information about the users.

2.3 Persistent Data Management

Since our program is mostly about editing, uploading, and downloading different kinds of reports and reviews, we made file reaching a priority. We've decided to use Google Drive to store the relevant files. The free version of Google Drive allows us to store 400.000 items, including files, folders, and shortcuts, and it gives a total of 30GB of storage area [2] [3]. Considering that on average a 35-page report pdf (which is an acceptable amount of pages for a student's internship report) is about 1.5 MB = 0.0015 GB, we can store about 20.000 such files. This number isn't enough to hold all the reports for all the semesters, but it is enough to store the files for one semester. When the semester ends, the files can be transferred into another, larger database for achieving purposes, thus clearing the space for the next semester.

We've decided to keep the user information and the basic information of the reports on MongoDB due to its high performance, fault tolerance, and ease of use. Since MongoDB can run on several servers, it also has high reliability. Since it replicates the data across those servers, it is relatively easy to recover data in the case of a crash. The data we will store on MongoDB will be the essential information of the users, such as email, password, etc. Also, we will store a basic representation of the reports as objects. They will have data about the report deadline, current status, instructor, etc.

2.4 Access Control and Security

In our Rigel Internship Report Management System, we emphasize the security of the student and administration's information and thus set an access control system. The access control system is based on the roles of the users which are given either by the admin or department secretary. Among these roles, there are unregistered users, students, instructors, TA, department secretary, department chair, and admin. As mentioned above, the admin is responsible for creating the user and assigning the role of department secretary for each engineering department. Afterward, each department's secretary creates users with the assigned student, instructor, department chair, and TA roles.

On the client side, our plan is to allow access to different UI to their users with roles by protected routing. When a user logs in, they are led through their own pacific routes according to their access permission based on their role. For example, a student would not have access to UI where the instructor sees all of the submitted reports.

On the backend side, the system logs in the user based on their roles that were given to them in the sign-up. Therefore, this role attribute is sent to the system's logic which ensures that certain features are functional only in their appropriate users' interfaces. For example, while similar student lists are shown to both instructors and department chairs, only the latter would have the functionality of withdrawing the student.

When it comes to access control to our database and backend code, since we plan to implement our system with STARS, we opted to not use any firewall in our security architecture. We aim to run the project on the school's local servers, thus providing a certain

level of protection to our backend code automatically. Also, MongoDB automatically encrypts the data before storing it. MongoDB ensures security by strong authentication with a username/password for access to the database which better the security overall.

	Student	Not Registered User	Admin	Department Chair	Secretary	Instructor	TA
Login		+					
Announcements	+	+	+	+	+	+	+
Main Page	+		+	+	+	+	+
Save/Submit Internship Report	+						
Ask for extension	+						
View Report	+					+	+
View Progress	+		+	+	+	+	+
Evaluate Report						+	
Give Extension						+	
View Calendar	+					+	+
View Profile	+				+	+	
Make Announcement				+	+	+	
Create / Delete Department Secretary, Chair Accounts			+				
Initialize Semester			+				
Create / Delete Student Accounts			+		+		
Delete Accounts			+		+		
Start Semester					+		
View List of Users			+	+	+		
Pair Instructor with Students					+		
Change Pairings					+		

Enter Company Evaluation Scores					+		
View Summer Training Grade Form					+	+	
View Criteria Report						+	
View Statistics				+		+	
Initial check of reports							+
Change withdrawal status						+	
Notifications	+					+	

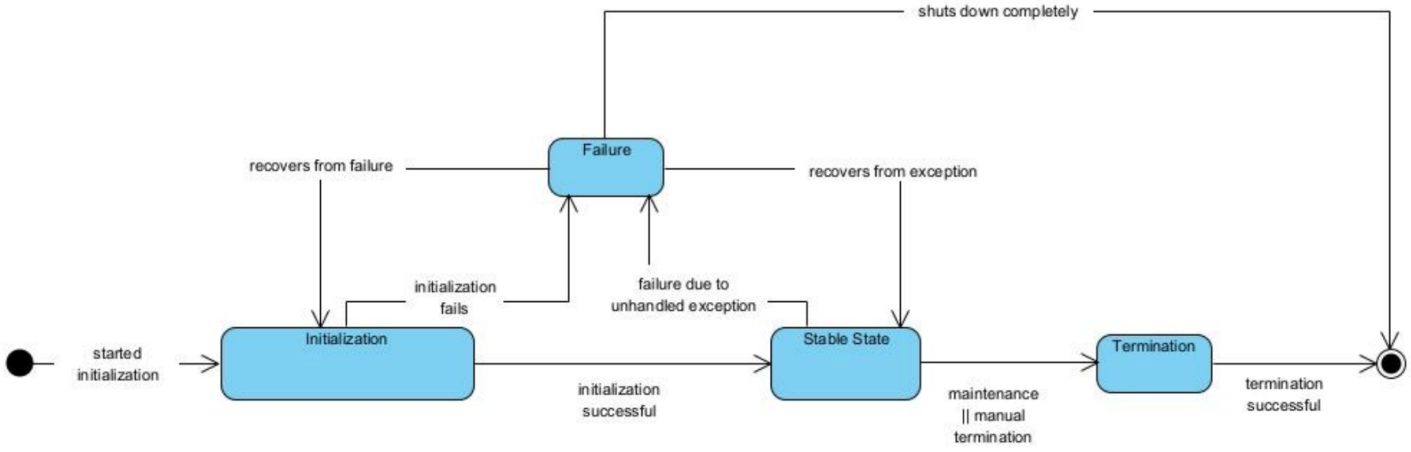
2.5 Boundary Conditions

Boundary conditions are a set of constraints that determine our web application's behavior at the edges of its usage. There are multiple types of conditions that push our system to these boundaries:

1) **Network Connectivity Issues:** Our application's requirements related to minimum network bandwidth and maximum network latency should be specified so that users can have a smooth experience. Our application aims to let users transfer and edit files while the rest of the information exchange will happen in forms in the application's interfaces. Therefore, 0.5 Mbps should be enough to exchange maximum-sized data of 4-5 pages which is 10-15 KB, between the user and the servers. Another important requirement is network latency as it affects the page load-time, as well as making the uploading process faster or slower. Therefore, considering instructors would need to edit the file in real-time within the application and students would need to upload their reports fast and easily, network latency should be less than 200 milliseconds. Conditions falling below these requirements would not result in failures but poor performance and user experience.
2) **Timeouts and Failures:** Related to network issues, timeouts, and failures can happen due to network bandwidth and latency among other causes when performing certain operations. One of these operations may be HTTP requests where the server may take longer than 10 seconds to respond and the system aborts the request, displaying a related error message. Apart from timeouts, failures can also happen due to scalability issues where servers cannot handle heavy loads with more than expected 100 concurrent users and their requests. For instance, these timeouts or direct failures can occur if 200 students try to upload their reports right before the deadline time or they go over the expected word count in their reports leading to too large of file

size for the minimum network bandwidth of the application. Similarly, issues related to databases and APIs can cause timeouts and failures. For instance, if there are some hardware issues in database servers like disk failure, the data can become unavailable or even lost. But hopefully, since MongoDB is a reliable database, this boundary condition is very unlikely to happen.

3) **API Connection Issues:** Similarly, we plan to use Google Drive API both within our application UI as a pdf editor and database for storing internship reports. Therefore, it is crucial to make sure that Google Drive API connection is made and running before initializing the system so that the users can access the reports. During the stable state of the system, the issue might be throttling and rate of Google Drive API limiting which limits the number of requests to 100 per hour per project [2]. Additionally, the API may limit the amount of data that can be retrieved or modified in a single request or over a given period of time. Therefore, in cases of overload of users, a “Quota Exceeded” error may be received leading to system failures. This failure would lead to an important data loss due to the ungraceful termination of the system. Therefore, periodic data saving to the API will be implemented so that the data of instructors are protected against crashes.
4) **User Authentication and Authorization Issues:** Wrong login credentials would be the first and most obvious boundary condition where permission to access the application would not be granted.
5) **Issues related to Invalid Input from Users:** Across all of the different interfaces of the application, there is a number of forms that the user is asked to fill in. Instructors, department secretaries, and students are the main users who enter their input. One of the invalid inputs would empty fields in instructors’ forms in submission attempts. In cases where Internship Form and Report Evaluation Form are not fully filled out, the system should not allow instructors to submit their work. In cases where their e-signature is not uploaded, the system should similarly give a warning stopping them from sending the official form without it. Likewise, for revision requests, there should be at least a comment explaining the feedback. For the department secretary responsible for creating the accounts, boundary conditions would be not fully filled with user info or the role, and the Bilkent mail address type not matching. Lastly, for students, their upload file should be of a supported type and within the maximum file size. Their submission also should not be completed without an explanation of the revision.



2.5.1 Initialization

Considering that Rigel System will run on a web browser, registered users would need to log in to their accounts with their correct credentials which corresponds to user authentication and authorization issues explained above. Non-registered users would only see the opening page of the application with a button to log in and to FAQ, and the app logo. When users are logged into their accounts, their respective interfaces are accessed and necessary data like their own or their students' reports. Since these fetchings are made from Google Drive APIs, their connection and running will be ensured. On the server side, a good internet connection will be ensured in initialization so that there will be no failures related to HTTP requests, file upload, or manipulation. Lastly, before initialization, we will make sure that the database is connected and running by implementing a database connection test before this state.

2.5.2 Termination

There are three ways to terminate the Rigel Internship Management System, the first one is when the user logs out on their will. In this case, the last periodic auto-save to API will be applied to store the latest version of their work. Thus, database and API connections will be safely ended before the termination. Moreover, in cases where there is a pending operation like the upload of a file, the system will not allow the termination so that users would not lose their data.

Secondly, the admin user may terminate the system. They may terminate the system for maintenance or when the semester is over, ensuring that all of the students' reports, instructors' evaluations, and statistics are backed up to the university's database.

Thirdly, in cases where recovery strategies did not work and termination happened with a complete shutdown, the data will have been saved by periodic auto-saving and alerts will be given to the admin user for quick action and maintenance.

2.5.3 Failure

For failures that happen in initialization and stable state, the recovery strategies will be applied. One of them is rollback procedures where the application will go back to its previous version. For instance, as seen in the diagram, initialization will occur again in case of failures. Or if it happens in the stable state, the operation will be reversed so that the system recovers back to the stable state. For instance, if a file cannot be uploaded, the application will roll back to the version without any upload attempt. Moreover, alerts will be sent to the admin user related to the failure so that rapid development is possible.

3. Low-Level Design

3.1 Object Design Trade-offs

Due to limited development time and budget, Rigel software only focuses on implementing a few design goals.

3.1.1 Usability vs. Security

The system will authenticate users with a password and email address for logging in. There will be no two-step authentication in Rigel software similar to the STARS system. This provides easier access to the system and all actors can get into their desired functionality within 3 clicks. Furthermore, the system will be functional as long as the user is logged in and it does not automatically log off after a specific period. This design choice enhances usability but makes Rigel software less secure than the SRS system where automatic termination is implemented.

3.1.2 Functionality vs. Usability

Although the design of the system is intuitive and similar to the STARS system which all users are familiar with, there are many new features provided in Rigel software. For instance, instructors have multiple modes when viewing student reports. Feedback mode allows them to comment on the student reports and criteria report mode allows them to view student reports as they fill in the criteria report. Users can explore these functionalities from the navigation bar and drop-down menu provided on their main page.

3.1.2 Security vs. Automation

Due to security reasons, some steps are not automated in the system. For instance, the department secretary has to enter company grades manually because company reports are confidential documents. Only authorized people in Bilkent like the department secretary can view them. In addition, all the new users have to be signed up by either the department secretary or admin to the system. We assume that this manual sign-up will ensure that users will be from Bilkent Engineering Department.

3.2 Final Object Design

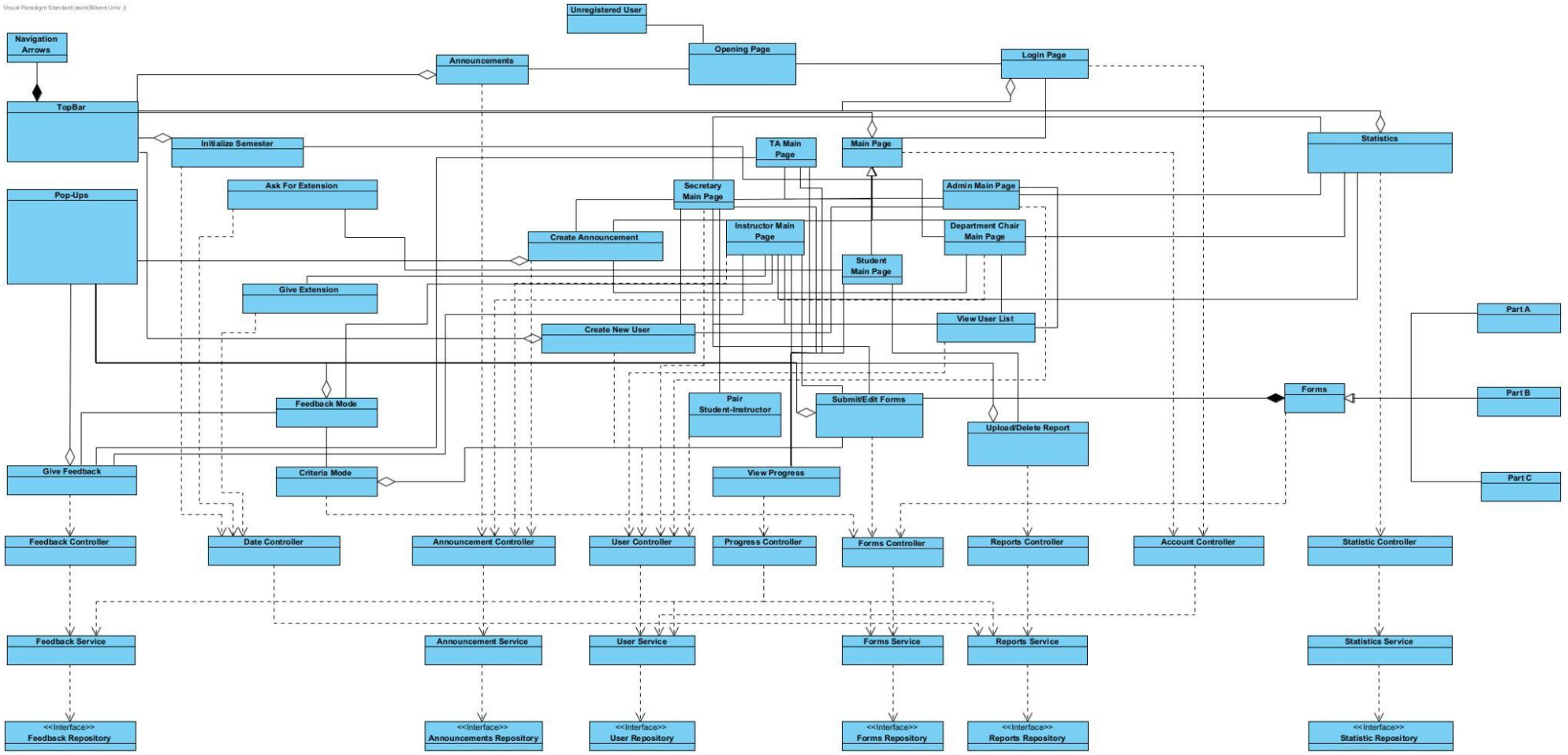


Diagram link: https://imgur.com/SpDqowl

3.3 Layers

3.3.1. User Interface Management Layer

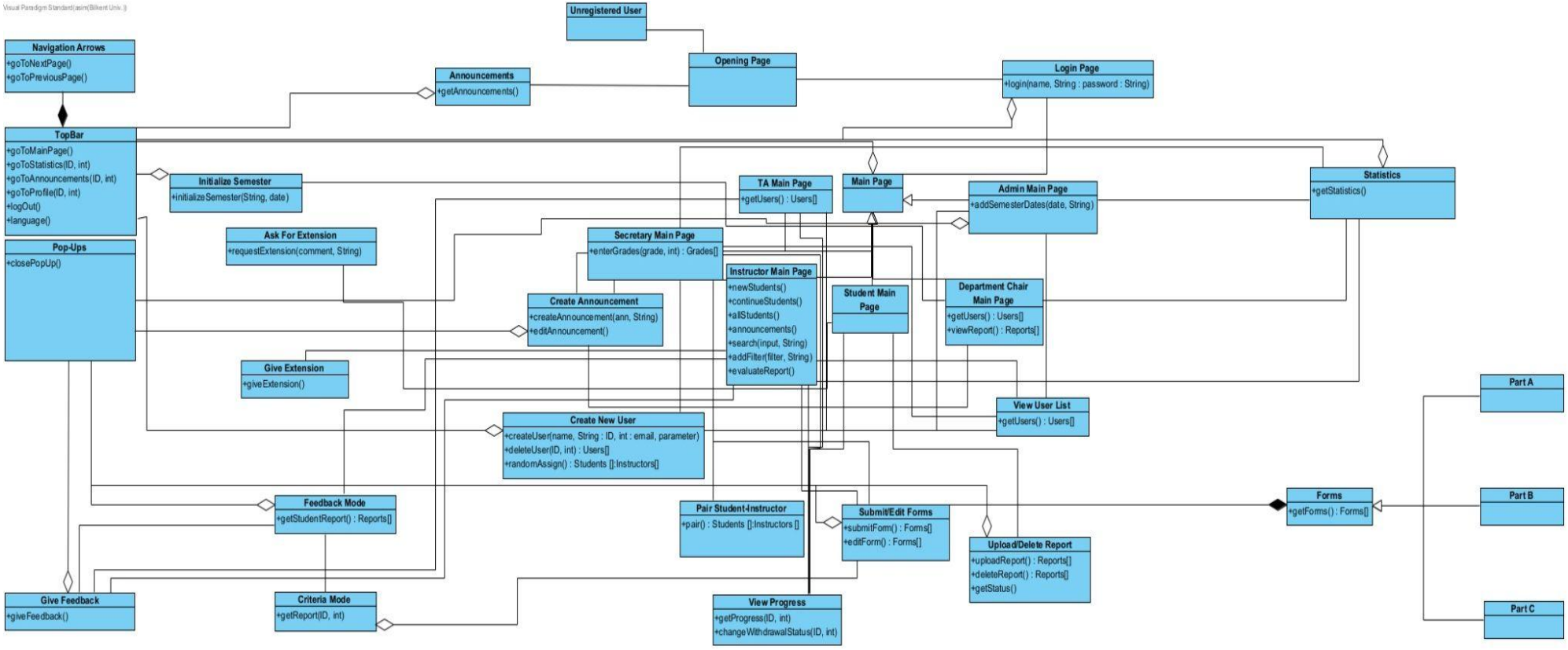


Diagram link: https://imgur.com/x12Np0p

3.3.2. Server Management Layer

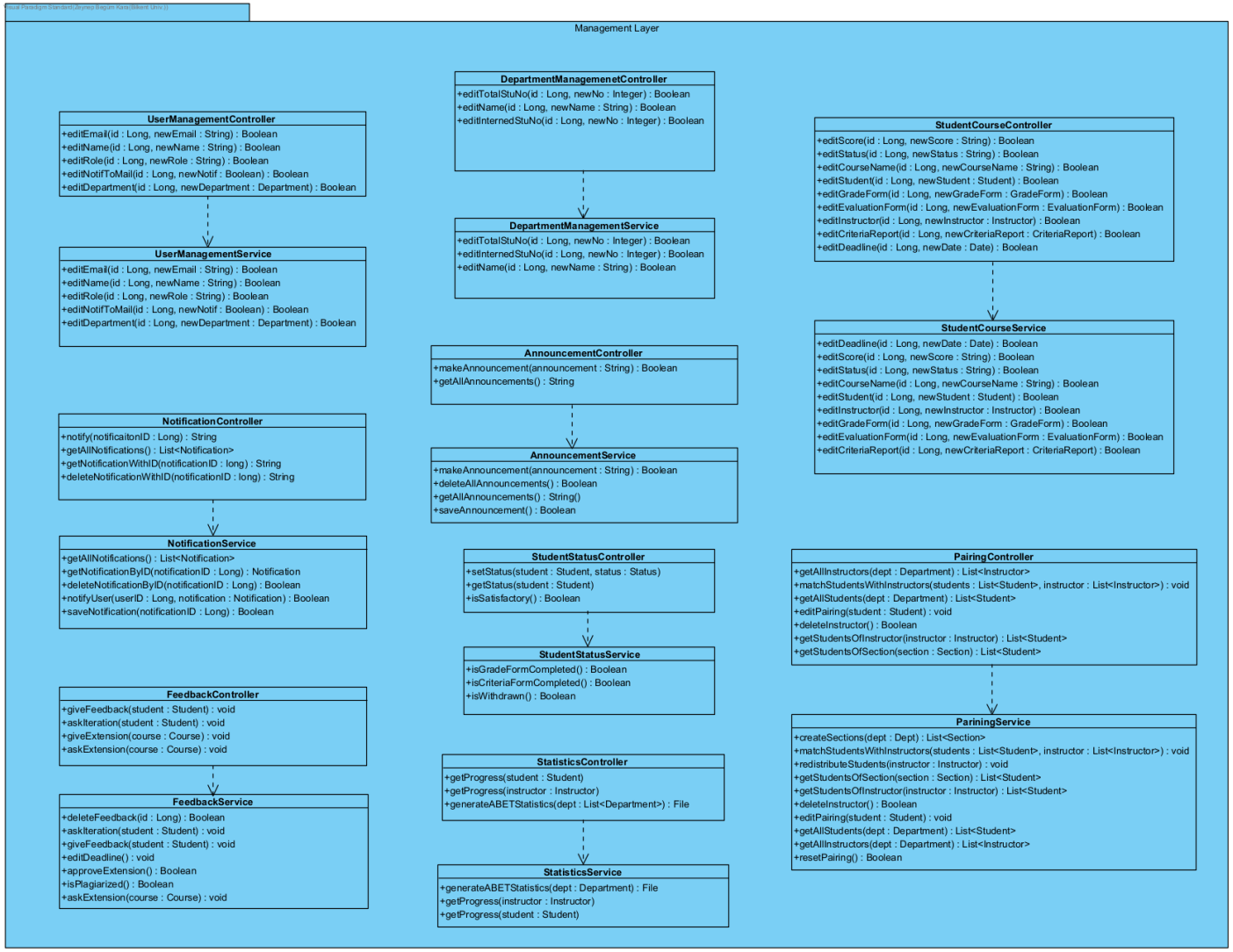


Diagram link: https://imgur.com/a/xvd6bf4

3.3.3. Data Management Layer

3.3.3.1 Data Management Layer: Entity Diagram

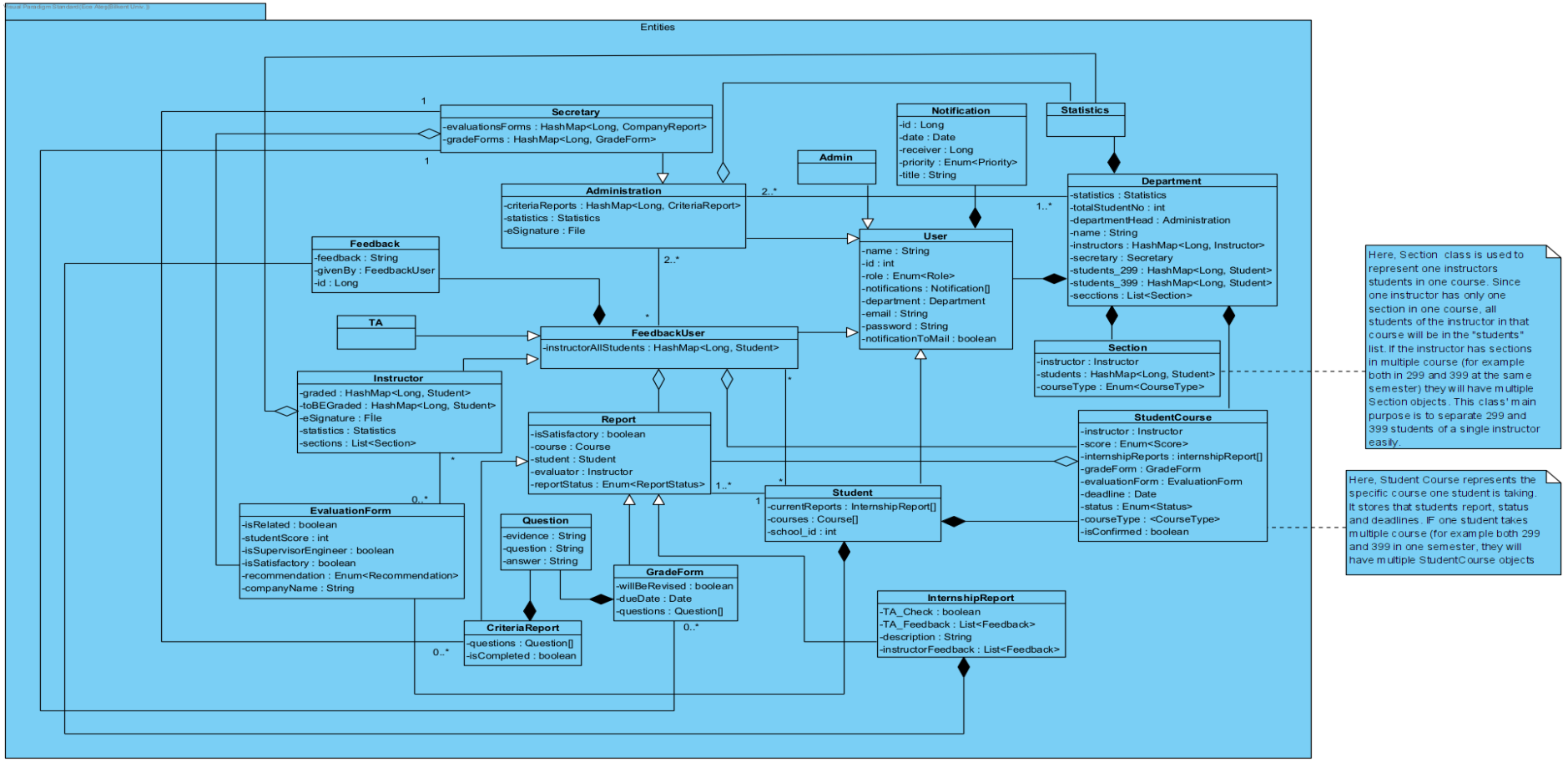


Diagram link: https://imgur.com/a/Y1ufHSN

3.3.3.2 Data Management Layer: Repository Diagram

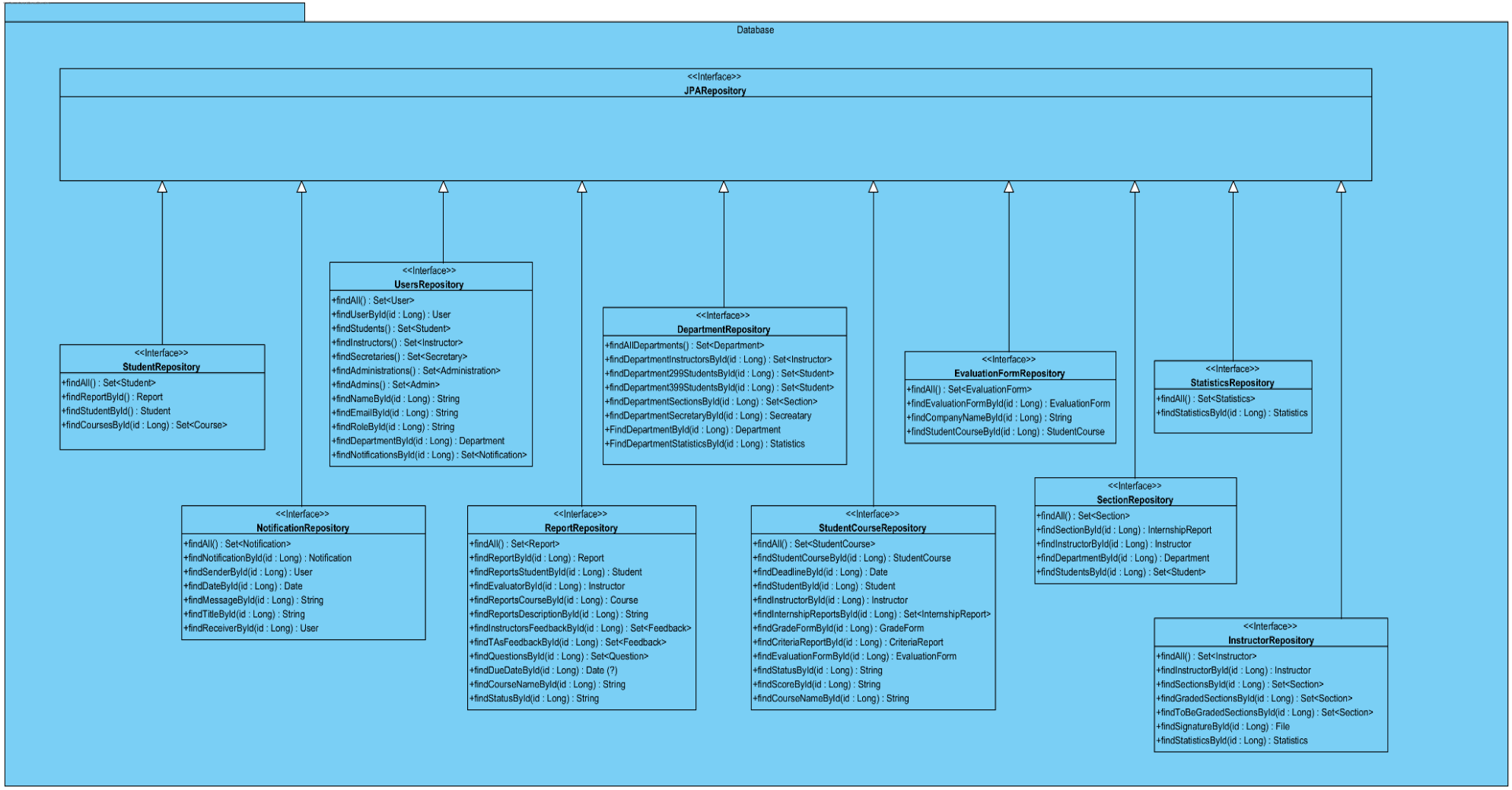


Diagram link: https://imgur.com/a/GuxjAMD

3.4 Packages

- **org.springframework.boot:** This package is a part of the Spring Framework and provides utilities for creating standalone, production-grade Spring-based applications.

- **com.google.api.services.calendar:** This package provides functionality for managing Google Calendar events, like creating, modifying, or deleting events.

- **com.google.api.services.drive**: This package allows developers to manage files stored in Google Drive programmatically, such as uploading, downloading, updating, and deleting files.

- **java.util:** This package contains various utility classes that provide functionality like data structures, date and time handling, input-output operations, and more.

- **React DOM:** It is a package that provides DOM-specific (document object model) methods for rendering React components on the webpage.

- **React Router:** This package allows client-side routing in a React application. It provides several components to help with the navigation and rendering of components based on the URL.

3.5 Design Patterns

There are three design patterns used in the implementation of Rigel software: Strategy, observer, and decorator patterns.

3.5.1 Strategy Design Pattern

A strategy design pattern provides the flexibility of choosing a way of implementing a feature without duplicating codes. In Rigel software, criteria mode, and feedback mode for instructors are essentially the same process, instructors can view and write on the document, save and continue afterward or submit the report. These actions are defined as interfaces, instead of methods in the student report class and criteria report class. In addition, strategy patterns will be used in student-instructor auto-matching by the department secretary. The matching can be done by listing students alphabetically, according to their course codes or their matched status, and then assigning a certain number of students to an instructor. The department secretary can choose to list students in one of these ways at any time.

3.5.2 Observer Design Pattern

Students' progress in the system is dependent on a valid company grade, student reports acceptance, and criteria report submission. All of these conditions are handled by other actors, therefore, the state of a student changes according to instructor or department

secretary events and the system has to coordinate these actions accordingly to update student progress. Similarly, student actors need to be notified about the initial report deadline when the department chair assigns a date. Due to the event-driven nature of Rigel software, an observer design pattern will be used in such operations where its results impact the state of other users.

3.5.3 Decorator Design Pattern

File upload and fetching operations will be the most commonly used features of Rigel software. Therefore, to handle the complexity and volume of the file operations, users can change the features of the file objects dynamically and file objects will be composed at runtime. Each file object can be uniquely defined by the combination of its file type, the file owner's user type, its creation date, and its name. The decorator design pattern is going to be used in order to identify file objects and is essential to handle files as we upload them to Google Drive API.

4. Glossary

Grade Form: Grade Form is the main form where each student's XX299 or XX399 grade is determined. This report is filled once when the student's grade is determined (as "satisfactory" or "unsatisfactory")
Internship Report: The internship report is the report students prepare about their internship experience. Students get feedback from TAs and their instructor for this report if it doesn't match the "satisfactory" grade requirements.
Criteria Report: Criteria Report is filled as a part of ABET Accreditation. Instructors fill this report to show evidence of the "satisfactory" status of the student's internship report.
Evaluation Form: The evaluation Form is based on the company where the student had completed their internship. The company's executive evaluates the student, which influences the student's grade. The company is also evaluated on certain criteria, for example, whether the student's supervisor was an engineer.
Not Registered User: This user represents the guests that can't reach most of the functionality of the website. They can only reach the most common functionality on the index page of the website

5. References

[1] "Bilkent University." [Online]. Available: https://w3.bilkent.edu.tr/web/kalite_guvencesi/faaliyet_raporu_2016-2017.pdf. [Accessed: 29-Mar-2023].
[2] Google, "Shared Drive Limits in Google Drive," *Google Workspace Learning Center*. [Online].Available: https://support.google.com/a/users/answer/7338880?hl=en#:~:text=File%20and%20folder%20limits%20in%20shared%20drives,-Item%20cap&text=A%20shared%20drive%20can%20contain,well%20below%20the%20strict%20limit.[Accessed: 05-May-2023].
[3] Google, "Storage and upload limits for Google Workspace," *Google Workspace Admin Help*. [Online]. Available: https://support.google.com/a/answer/172541?hl=en.[Accessed: 05-May-2023].