Bilkent University

Department of Computer Engineering

CS 319 Term Project

ErasmusNET

Group 3F

Arda Baktır, Beyza Çağlar, Gülin Çetinus, Mehmet Kağan İlbak, Zeynep Selcen Öztunç

Instructor: Eray Tüzün
Teaching Assistant(s): Metehan Saçakçı, Emre Sülün, Muhammed Umair Ahmed, İdil Hanhan and Mert Kara

Design Report

November 29, 2022

Contents

1. Introduction

1.1 Purpose of the system

ErasmusNET is a web-based manager for the Erasmus program, aiming to aid outgoing students, incoming students, department coordinators, and international office coordinators with their responsibilities. The system's primary purpose is to make every part of the Erasmus procedure easier by uniting every operation the students and the coordinators have to perform so that every problem can be dealt with in one place. For instance, the students can upload multiple required files while making an application or make an appointment with the coordinators to discuss any issues they may have during the application process. The coordinators can directly resolve the student's problem or arrange a meeting. They can also see and approve all the outgoing students' pre-approval forms in one place without the risk of the pre-approval forms being scattered or lost in the hundreds of emails sent to the coordinators.

1.2 Design Goals

The design goals of ErasmusNET are inspired by the non-functional requirements specified in the Analysis report. These goals are essential specifications because they are aimed at enhancing the user's experience.

1.2.1. Performance

The performance of the system affects the quality of the user experience. Therefore, the performance of the software should be close to standard websites. However, it is not expected to be at a millisecond scale.

1.2.2. Usability

The UI of the program should be intuitive and easy to understand for a better user experience.

1.2.3. Reliability

An unreliable system may result in significant delays and other issues. Therefore, it is essential that the program works as expected on all platforms.

1.2.4. Security

ErasmusNET contains sensitive information on the student's identity, passport, visa, bank account, CV, and transcript. Hence, the program must be secure and well-protected.

1.2.5. Scalability

The program will have many users. Therefore, it must be able to accept extensive file submissions, such as application documents, especially close to application deadlines.

1.2.6. Maintainability

The program should not be maintained with difficulty; it should require minimal effort.

1.2.7. Deployability

ErasmusNET is a program that the user does not have to develop. Thus, it should be easy to deploy for the ease of the user.

1.2.8. Extensibility

The program should allow trivial changes without too much effort.

2. High-Level Software Architecture

2.1 Subsystem Decomposition

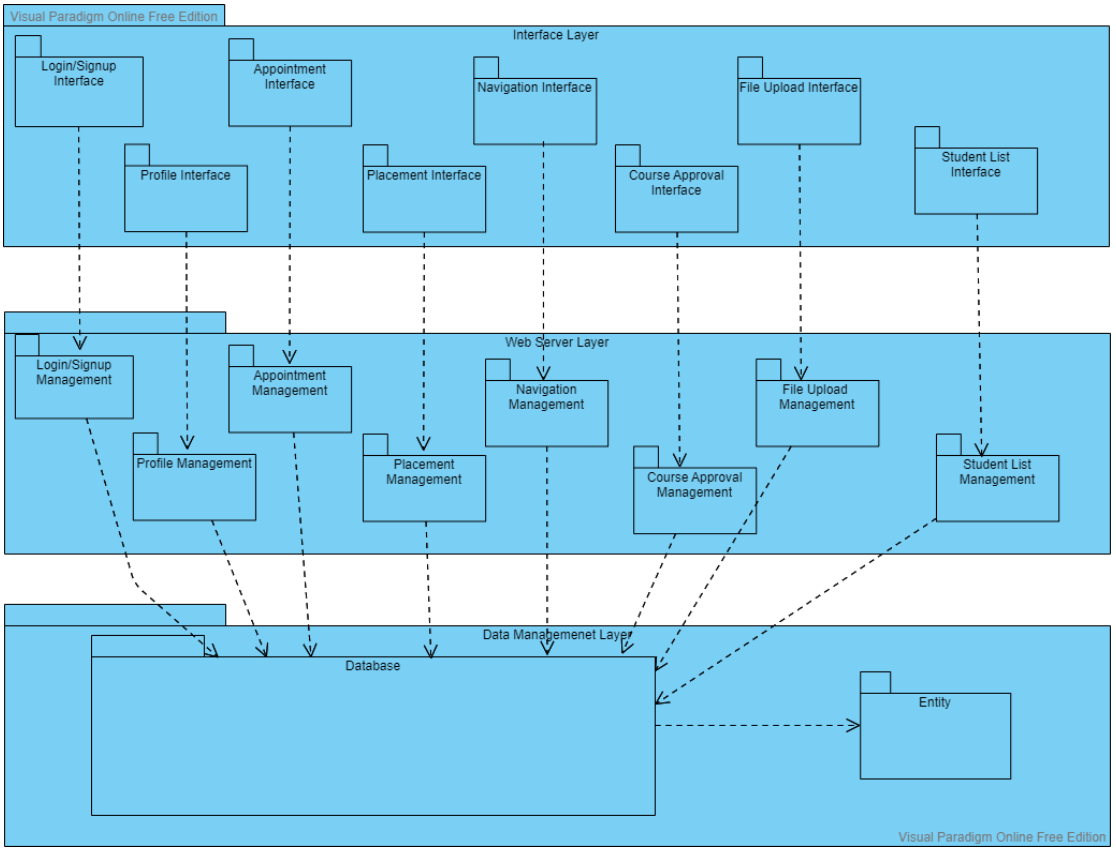

Figure 1: Subsystem Decomposition of ErasmusNET

Our project will consist of three layers: Interface Layer, Web Server Layer, and Data Management Layer. The Interface Layer has the pages for ErasmusNET. This layer corresponds to the boundary objects of the project. The user can interact with the pages to use the offered functionalities. Each UI element will be designed based on usability and extensibility design goals.

The Web Server Layer contains all the subsystems to manage the Interface Layer in the backend. For example, Student List Management will fetch the necessary student information from the database to be displayed in the Interface Layer. It has classes and interfaces for viewing and managing documents and personal information of students. Like the Student List Management, other subsystems in this layer manage different parts of the ErasmusNET, and all of these subsystems depend on the database for fetching data. The separation of these subsystems provides a maintainable and efficient system.

The Data Management Layer has the Database and Entity subsystems. The Database subsystem communicates with the database, whereas the Entity has Entity classes and communicates with the Entity objects.

2.2 Hardware/Software Mapping

ErasmusNET is a web-based software meaning it runs on a web server. There is no need for special-purpose hardware. The software should be available as long as the users have a device connected to the Internet and can access a web browser. Google Chrome, Safari, Opera, and Mozilla Firefox support the website.

2.3 Persistent Data Management

In the implementation of our project, we decided to use Firebase since our backend team had the most knowledge about Firebase. The database will have tables for students, courses, universities, instructors, exchange coordinators, admins, placement list, and so on. The primary keys of these tables will be selected so that it uniquely identifies each table record, such as the IDs of users. Cookies are used to store user-related information between pages.

2.4 Access Control and Security

Our project provides security and access control. Only the required amount of information is sent to the front end. When any of the users log in to the system, the system matches the credentials of the users to one of the user types. It determines the information that will be available to the user according to that

type. All of the users can log in and have their profile page. They also can access the ‘About’ and ‘Contact’ pages, in which information about the exchange programs is displayed, and contact information on exchange-related people or institutions is given. The students have access to the student interface, where they can apply to the exchange program, make appointments with coordinators, and upload syllabi of the courses they want to take. They can also view many pages such as Knowledge Library, Student Stories, and Document Templates.

The coordinators have access to the coordinator interface, where they can approve/ reject applications and appointment requests and view information about students. The information may include the student’s uploaded files, Erasmus/Exchange grades, preferred universities, and contact information about the student and the student’s first-degree relatives.

Admins have access to the admin interface. They can create, modify and delete deadlines, view every user’s info, and modify information about the program, such as the contact numbers and description of the program.

2.5 Boundary Conditions

2.5.1 Initialization

ErasmusNET runs on the web; therefore, there is no need to install the software. As mentioned above, the only requirement to access the website is a device with Internet access and web browsers.

2.5.2 Termination

The website subsystems work as a whole, which means that when a single subsystem is terminated, the whole system terminates. If the admin terminates a subsystem, all of the data created by users in that subsystem are saved to the database.

2.5.3 Failure

The failures caused by the developers will return different HTTP codes for different operations. A pop-up will be displayed notifying the users about why the error has occurred (e.g., “You cannot change your preferred universities anymore, the deadline has passed”). The project will be run on AWS. Thus AWS will do the error handling.

3. Low-Level Design

3.1 Object Design Trade-offs

- **Maintainability versus Performance:** Our project divides each object into its own entity, implementation and controller class. This can cause a performance reduction, but it is ideal for data abstraction and its maintainability as it can be changed controllably.
- **Memory versus Performance:** Since we are using objects for each datum this can cause a limitation in memory, but with this objects for each datum approach, everything would be more efficient.
- **Security versus Usability:** Since the login procedure consists of ID and password login, this decreases our security, but would increase the usability by being easy to access to the services.
- **Functionality versus Usability:** The main purpose of this project is to make the Erasmus application process more easy to use for all the students and the coordinators. Because of this, we need both functionality and usability. This causes the project to have decreased usability with all the functions needed for the project.

3.2 Final Object Design

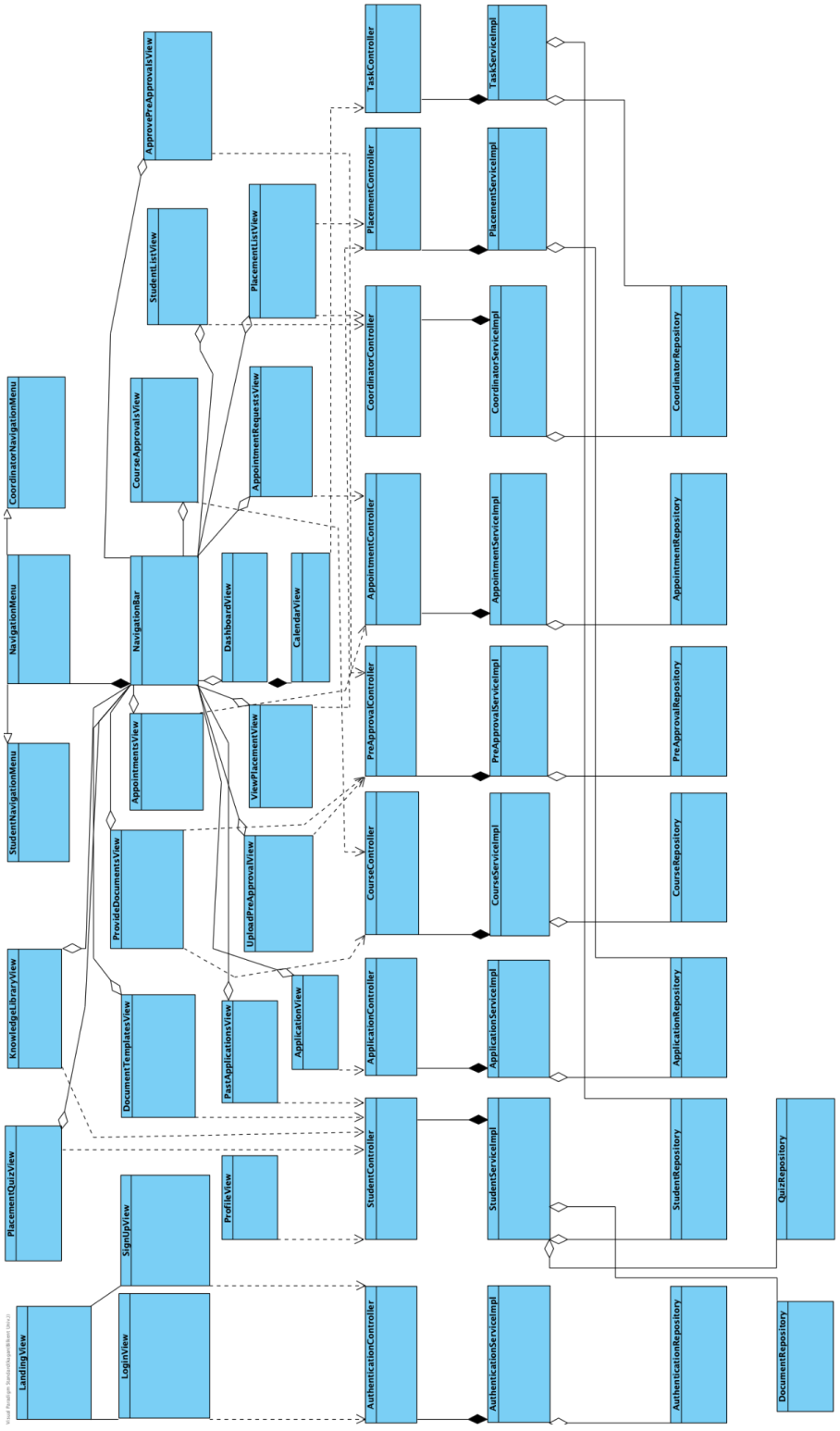

Figure 2: Diagram for Object Design

Left side of the diagram:

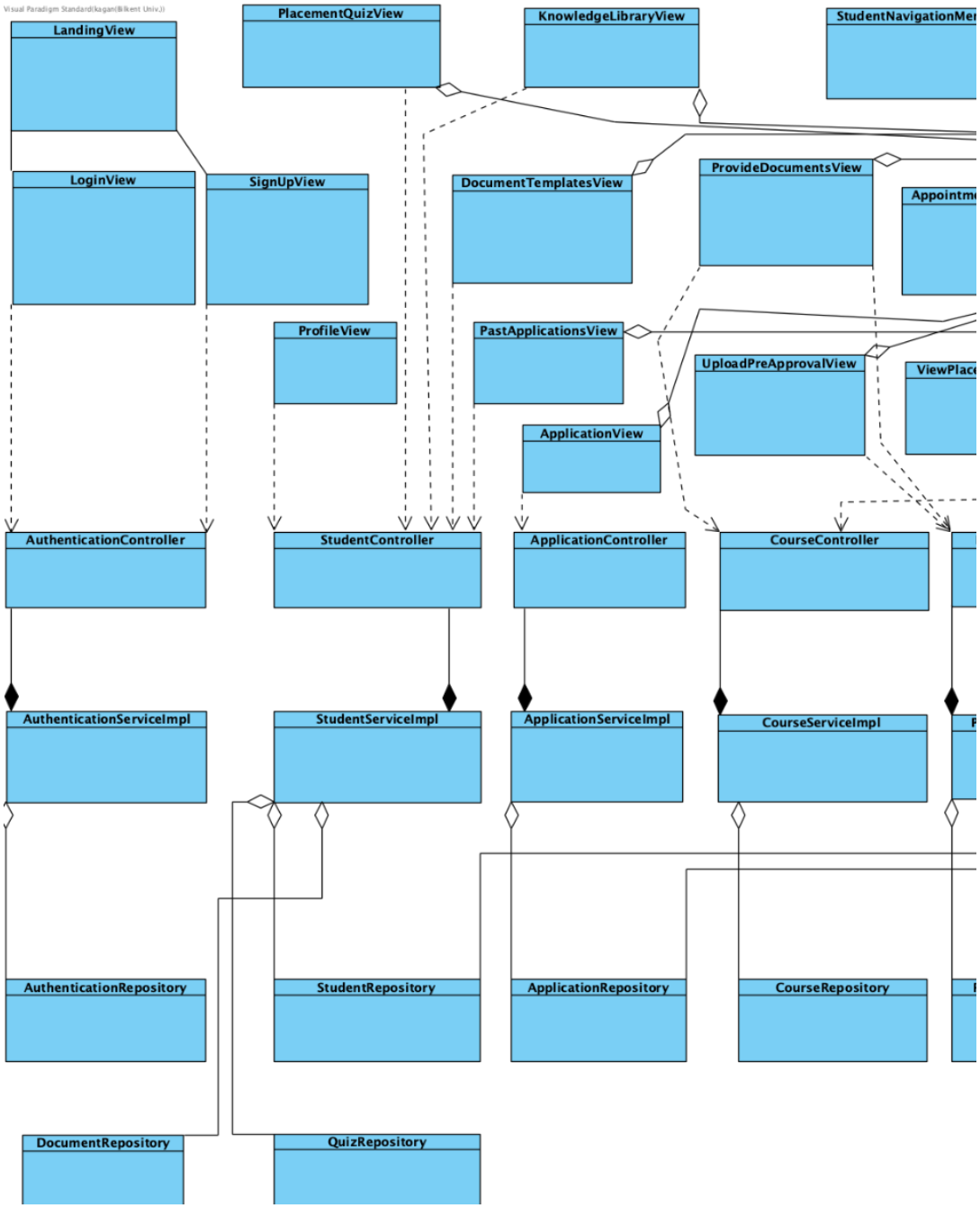

Figure 3: Left Side of the Object Diagram

Right side of the diagram:

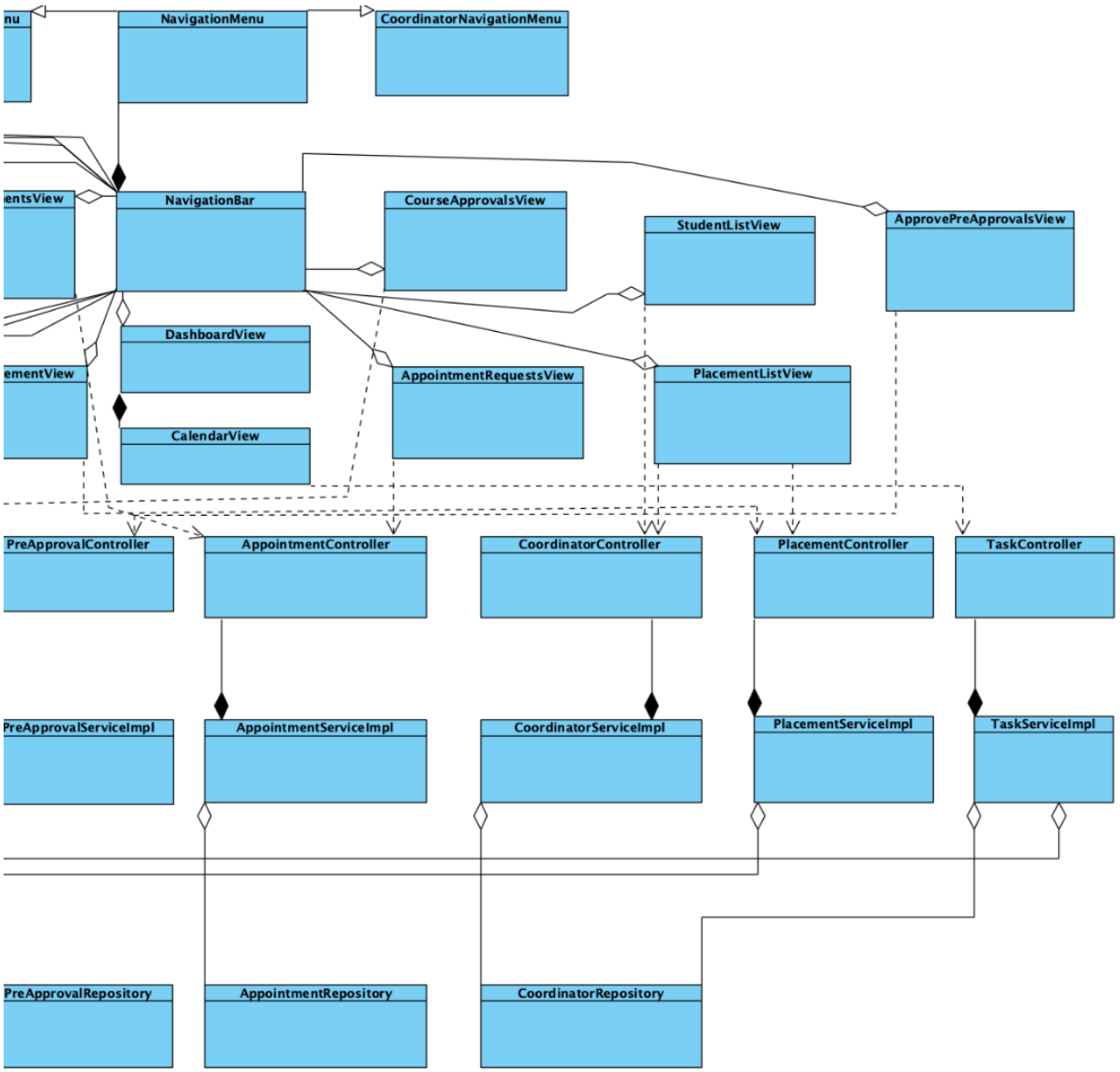

Figure 4: Right Side of the Object Diagram

3.3 Layers

3.3.1 User Interface Management Layer

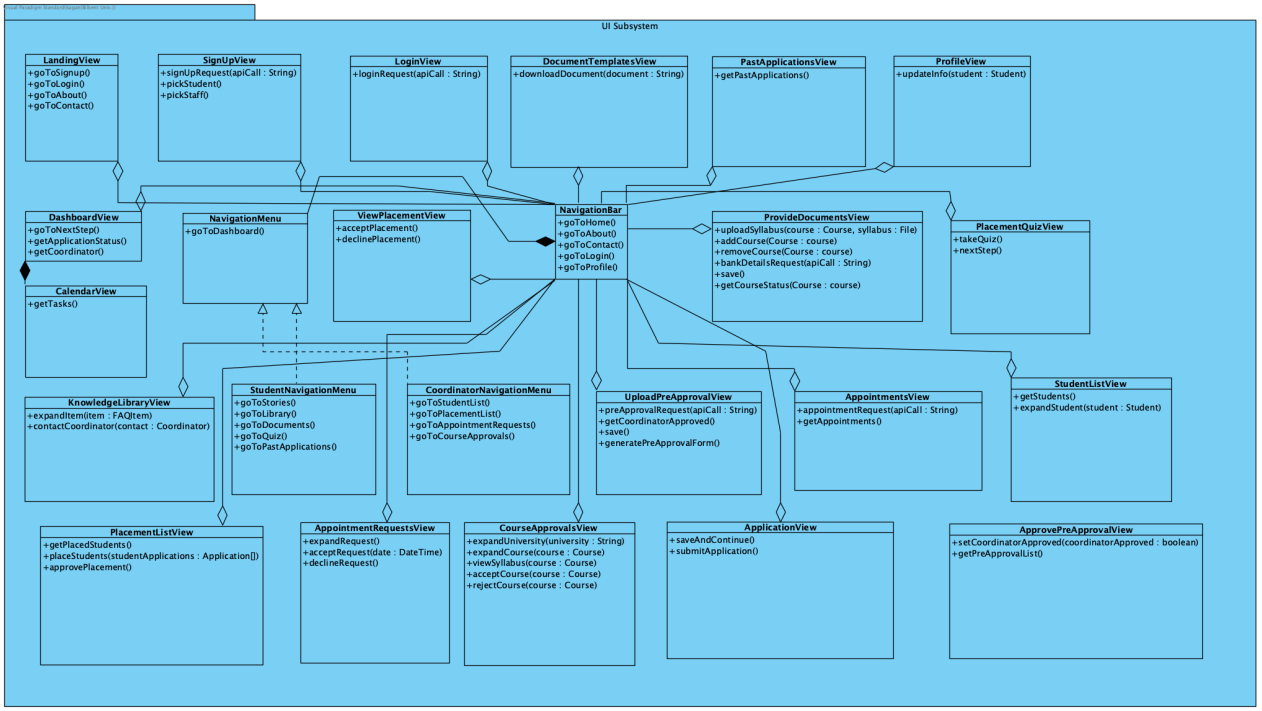

Figure 5: Class Diagram for the User Interface Management Layer

This is the boundary between the user and the controller. Classes in this layer are javascript objects and they communicate with the web server layer.

3.3.2 Web Server Layer

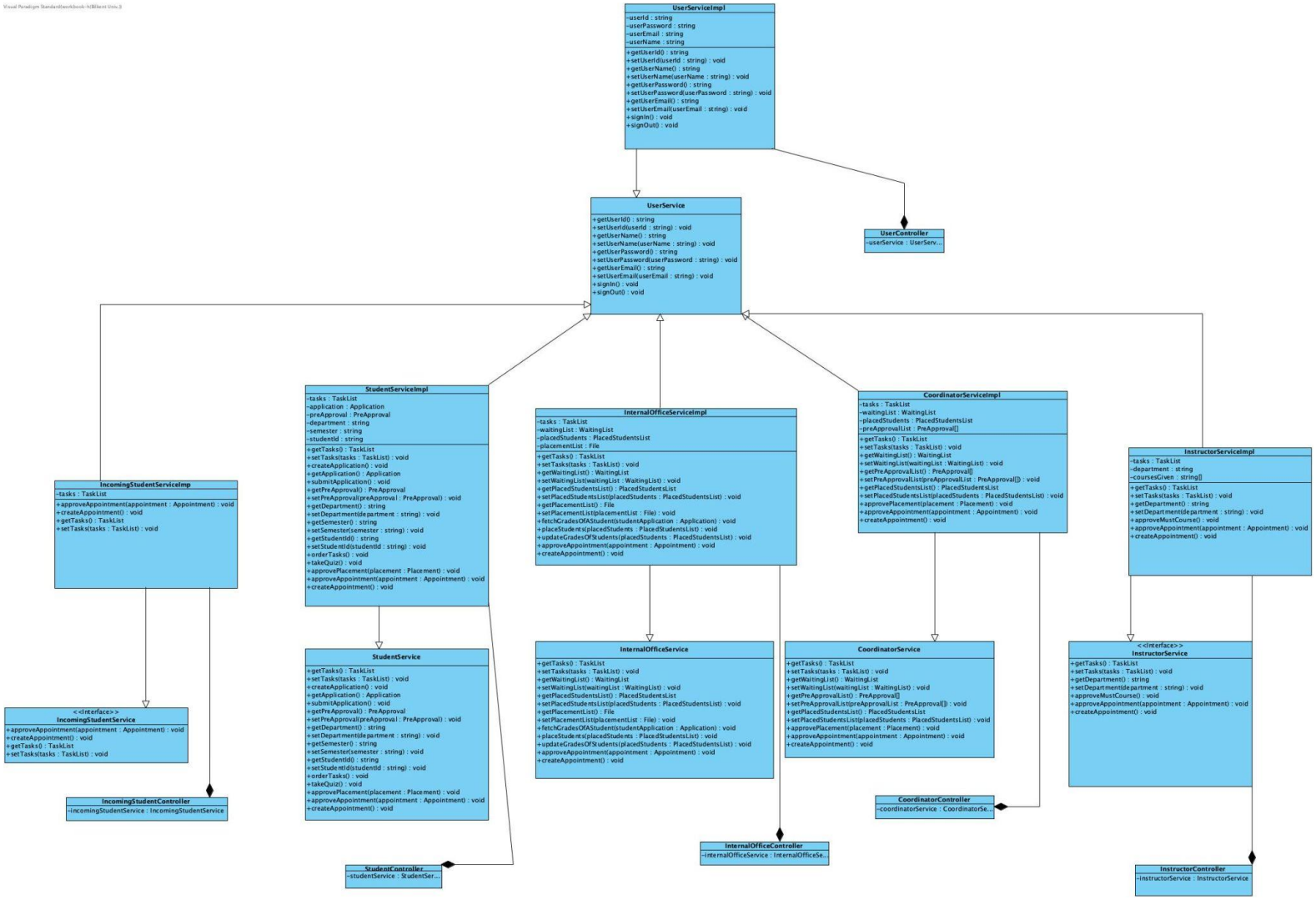

Figure 6: Web Server Layer's Student, Coordinator, Instructor, InternationalOffice and IncomingStudent Classes

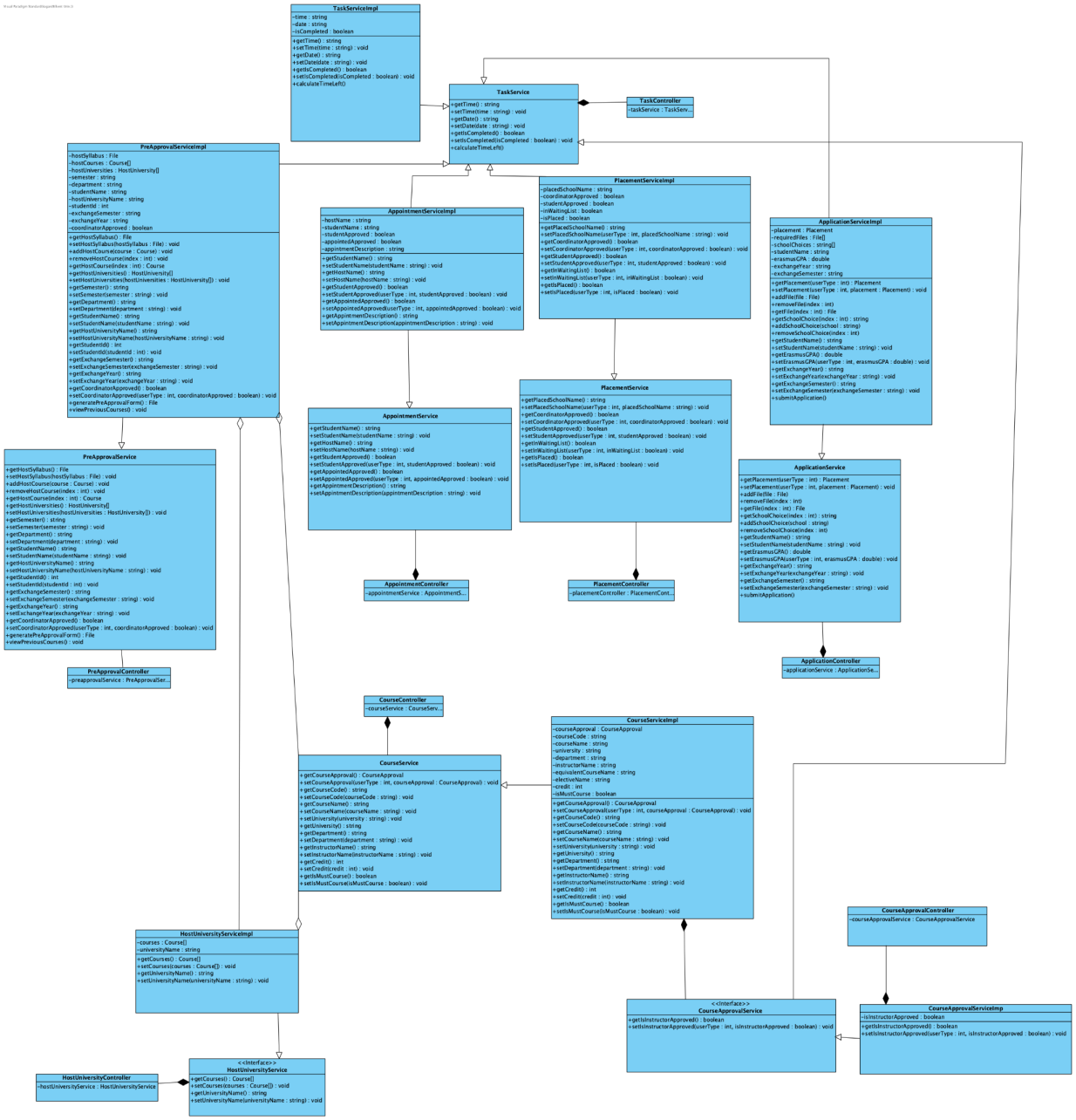

Figure 7: Web Server Layer's Appointment, Application, Course, Placement, PreApproval, Task, HostUniversity Classes

This layer acts as a controller between UI and data management layers. Primary responsibility of this layer connecting the UI to entities.

3.3.3 Data Management Layer

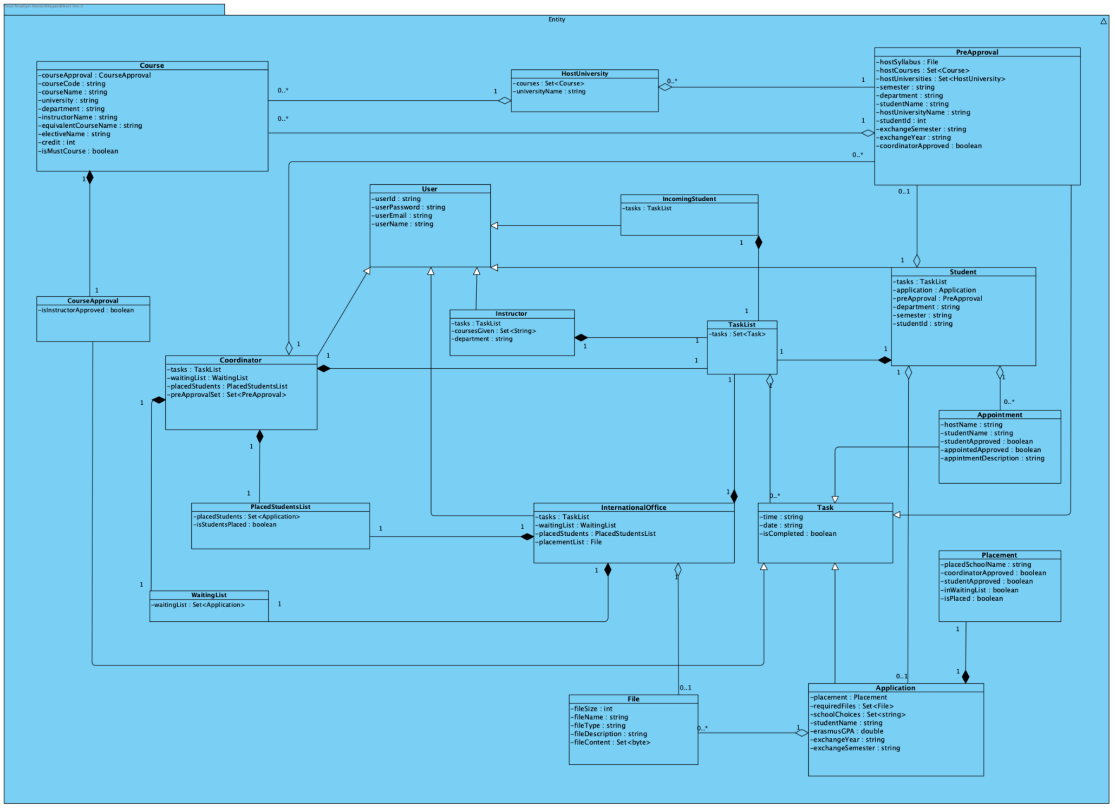

Figure 8: Entity Package for the Data Management Layer

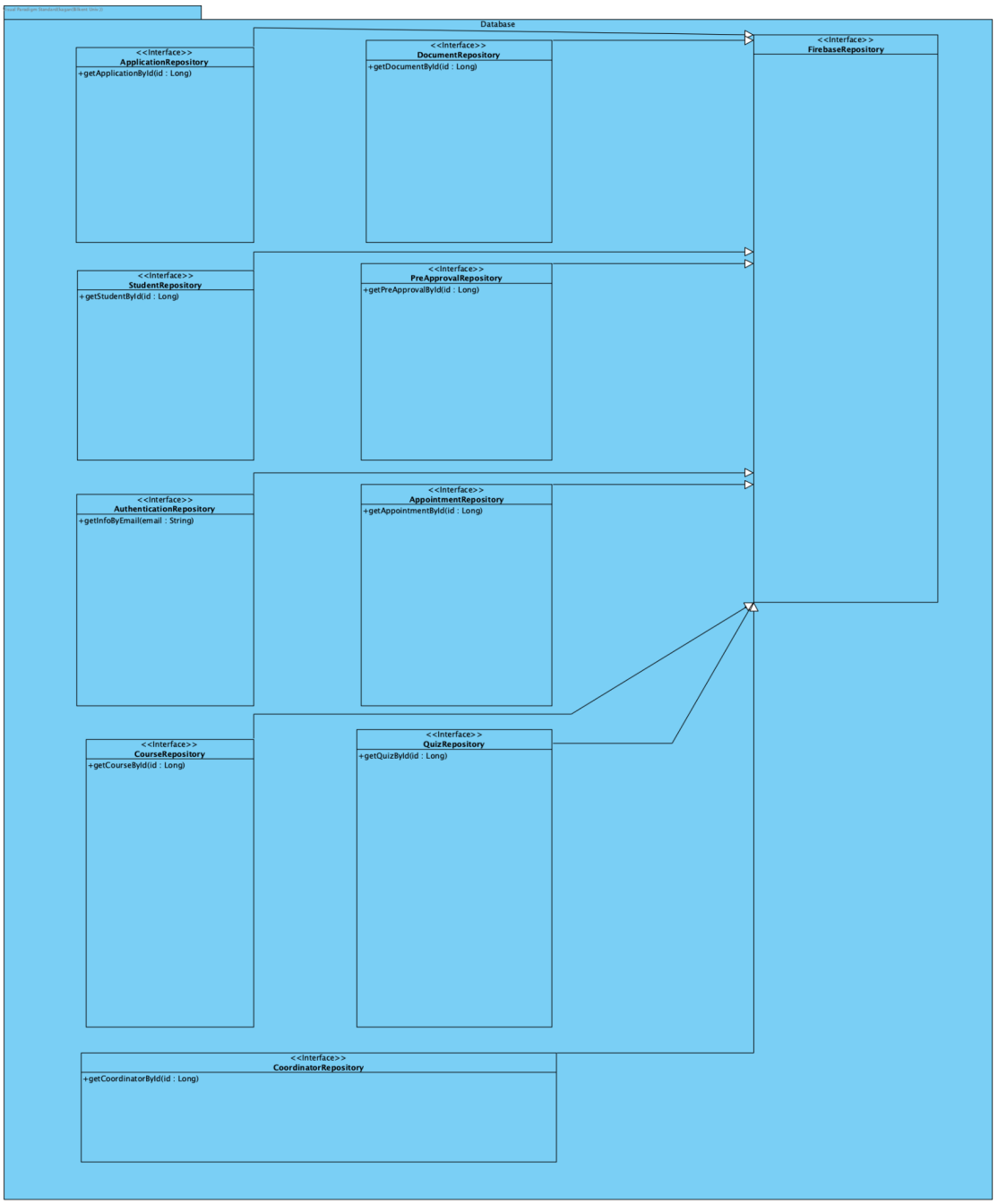

Figure 9: Database Package

This layer consists of Entity and Database subsystems. Primary responsibility of this layer is supporting the web server layer by managing and storing data.

3.4 Packages

3.4.1 Internal Packages

3.4.1.1 Entities

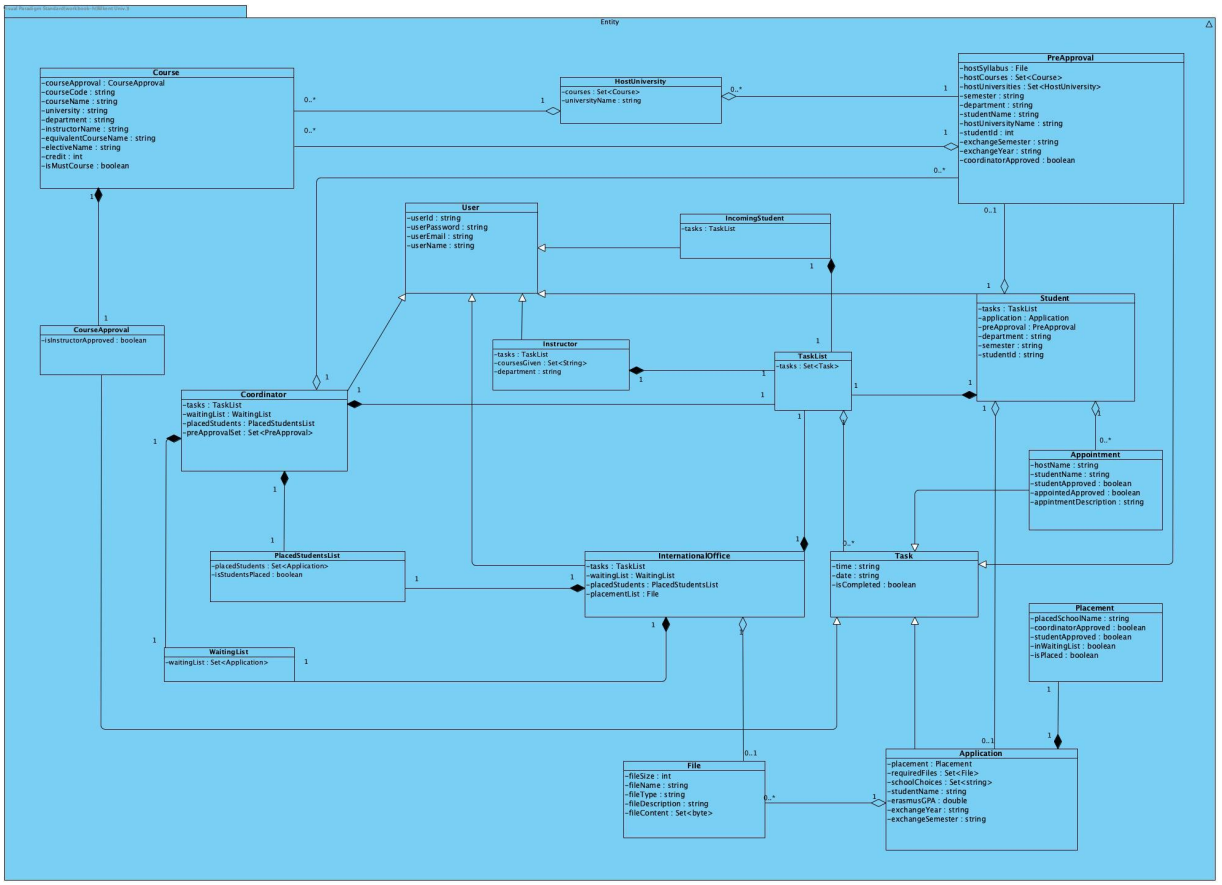

Figure 10: Entity Package

This package contains entity classes that contain methods and objects for managing the interactions and storing information.

For simplicity purposes we haven't included the getter and setter methods for each attribute.

3.4.2 External Packages

3.4.2.1 Firebase Authentication

This library is used to create a secure authentication service for the system.

3.4.2.2 Firebase Database

This library is used to create databases in our data management layer.

4. References

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd

Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0