Bilkent University

Department of Computer Engineering

2022-2023 Fall Semester

CS 319: Object-Oriented Software Engineering

ERASMUS PROJECT

DESIGN REPORT ITERATION 1

Group Members	Student Numbers
Bora Yılmaz	22003359
Cengizhan Terzioğlu	22003014
Emirkan Derköken	22002693
Göktuğ Kuşcu	22002867
Murat Güney Kemal	22002692
Onur Asım İlhan	21903375

Instructor: Eray Tüzün

Teaching Assistants: Metehan Saçakçı, Emre Sülün, Muhammad Umair Ahmed, İdil Hanhan, Mert Kara

TABLE OF CONTENTS

1. Introduction

1.1. Purpose of the System

The application is a web-based erasmus and exchange application system for Bilkent University. The primary goal of the application is to provide an easy application process for students and to provide a more organized system for the instructors and coordinators.

The application also aims to minimize the usage of paper and emails.

1.2. Design Goals

The design goals of the application are outlined by the non functional requirements of the project. The application aims to provide an easy to use and reliable solution for students and instructors.

1.2.1 Usability

Since the application will be used by a wide range of students and instructors, the website will be designed with simplicity in mind to provide an easily navigable interface for all users. The calendar and to-do list features will assist the user with their work. The website will allow students to prepare their pre-approval forms with minimal paper and email usage.

1.2.2 Reliability

The website will prevent any loss of data in the case of a system crash. Any occurring error will appropriately indicate the error’s reason and consequences.

1.2.3 Performance

The website will process any operation and show its result in under 3 seconds. The loading times of the website will be in under 2 seconds.

1.2.4 Supportability

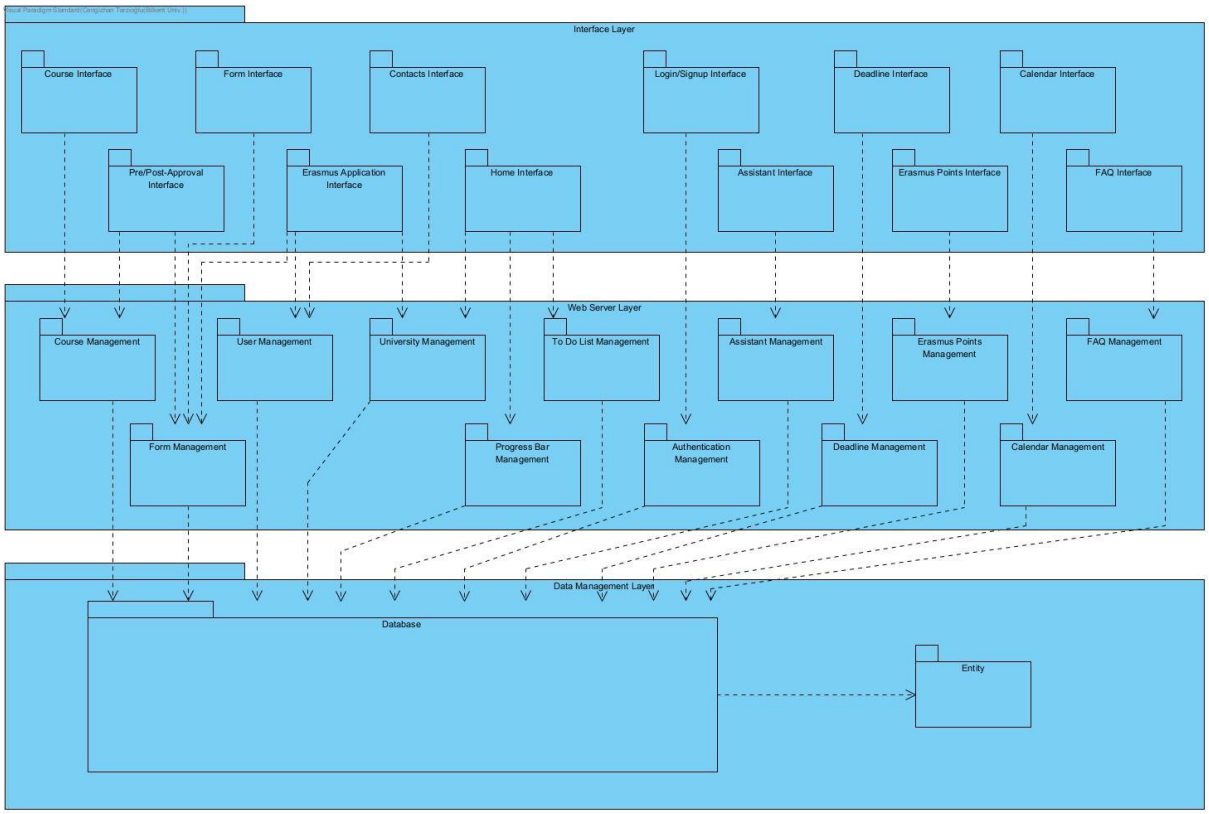
The application will support Google Chrome version 107 and above, and Firefox version 102 and above. The website design will be consistent across different devices and browsers.

1.2.5 Security

The website contains many sensitive information like the ID's of the students. Therefore the website will only provide the required data for each operation and will not allow any users to access sensitive information. The website will be protected against any cyberattacks and will try to prevent any data breaches.

2. High-level Software Architecture

2.1 Subsystem Decomposition



A three layer architecture is chosen in order to ensure the maintainability goal is satisfied.

Interface Layer consists of packages that will keep the boundary objects, which are the website pages. Every page depends on at least one of the controller packages from the Web Server Layer which allows user interaction through buttons that are implemented on the page. Each package includes related pages and functions those pages are able to perform. This layer is designed to ensure user-friendliness and simplicity goals are satisfied.

Web Server Layer consists of packages that will keep the controller objects. These controller objects contain functions that are performed on the server side. Each package in this layer is created to perform a different core functionality. For example, the Course Management package contains all of the functionalities that are related with courses such as course proposal, course approval, getting a list of already approved courses etc. Each package depends on the database.

Data Management Layer contains Database and Entity packages. Database package is created to communicate with the database so that data addition, retrieval and deletion operations can be performed. Entity package keeps persistent objects that are created during the usage of this application.

2.2 Hardware/Software Mapping

Additional hardware components aren’t needed to run our erasmus application project. It is a web based project so web browsers are needed and hardware systems must be strong enough to run a web browser. Web browser usage is compulsory to use our application, but it can be used on any electronic device as long as they have web browser support. For personal computers some I/O devices are necessary such as keyboard, mouse and monitor. The application runs in web browsers such as Google Chrome, Mozilla Firefox, Safari and Opera. Though it might be incompatible for old web browsers such as Internet Explorer 8 due to API and external library usage.

2.3 Persistent data management

We selected SQLite to use in our project. We chose SQLite because it is free and object relational which makes OOP principles to be easily applied. Also our team mostly had experience with SQLite database which was another reason to choose it. Objects will be kept as tables and each table will consist of key-value pairs. Data kept in the objects will be edited using the services that are implemented in the Web Server Layer and requests for editing will be done through the Database package of the Data Management Layer. The data can change at any time so it is important to send HTTP requests such as GET, POST and DELETE to the database dynamically. Also cookies will be used to transfer continuously used data from page to page.

2.4 Access control and security

Access control and security are very essential concepts for our erasmus application project. Each user will have different permissions that are determined according to their roles. When the user logs in to the system, our system checks all the roles that user has and then opens the home pages related with those roles. These homepages have access to other role related pages that the user has permission to enter. Any page that user does not have permission to will not be shown. Only the required information is pulled from the database to our front-end for each page to make the system more secure. Each password will be hashed to give extra protection.

2.5 Boundary Conditions

2.5.1 Initialization

Users don’t need to install any program to their devices since our project runs on a web server at all times. This makes the use of the application easier. Users just need a device that has web browser support and access to Internet connection. Before log in only log in, sign up, FAQ and partner universities pages can be seen. User needs an account that is kept in the database to be able to log in to the system. The users can perform their actions from different web browsers or different tabs of the same web browser at the same time. For each web browser, they will have to log in separately. Though they will not have to log in again to perform an action in a different tab of the same web browser if the user logged in to the system from

another tab. All data will be initialized from the database at the time of the creation of the page.

2.5.2 Termination

The Erasmus Application makes use of a four layered architecture. Hence the whole system does not have to be shut down when one of the subsystems needs to be terminated for maintenance. Furthermore, the debug mode of the Flask framework makes it possible to perform maintenance while the website is up. Users will be notified at least one day prior to the termination if they are planned terminations. Before the termination of the system, each data is saved to the database to prevent any data loss.

2.5.3 Failure

When an error occurs, a message will be displayed to the users, detailing what went wrong (e.g “The course you have chosen cannot be added as it does not have a Bilkent equivalent.”). For debugging purposes, any issues regarding the developer side will be logged in detail to a file.

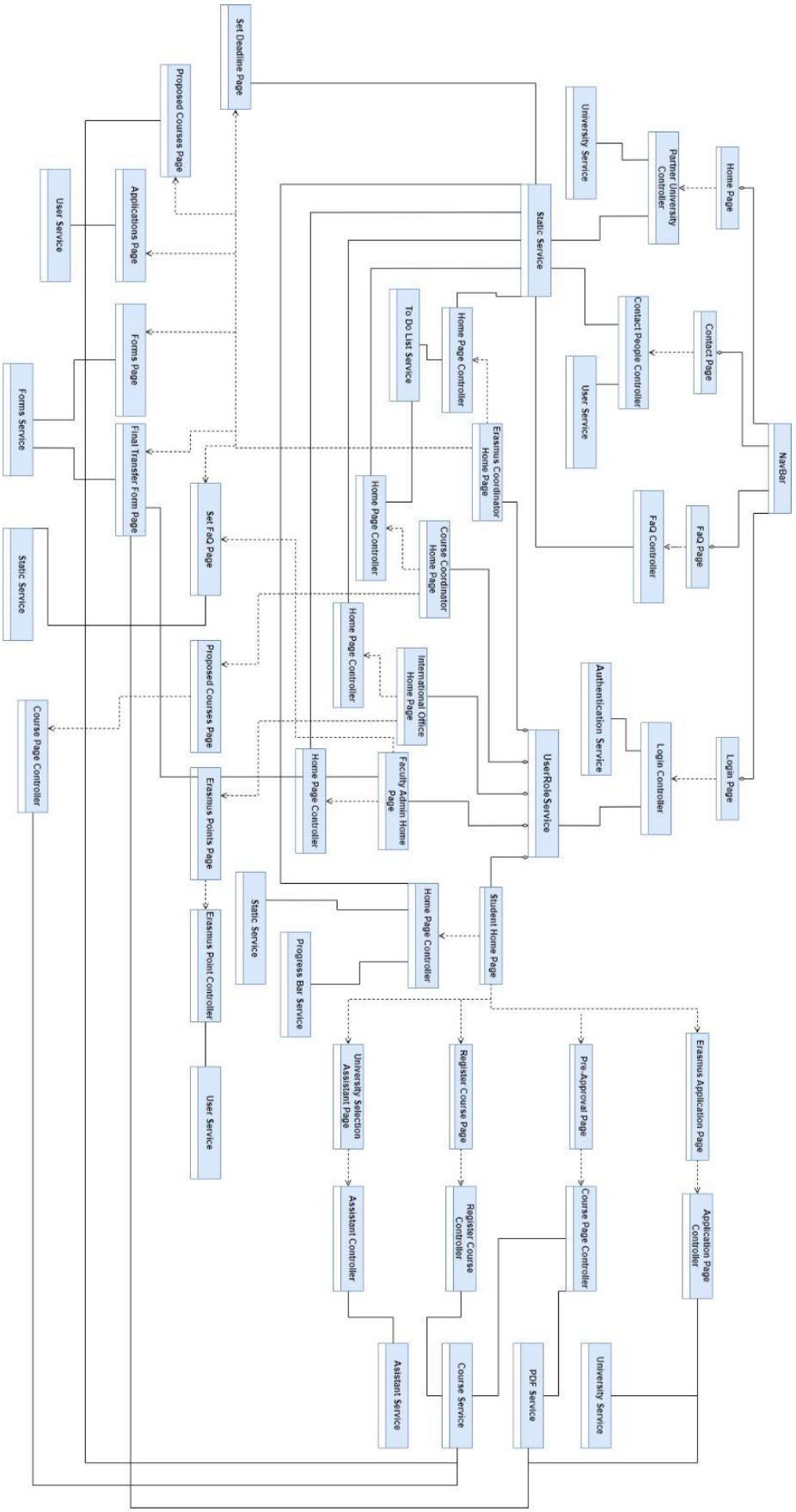
3.Low Level Design

3.1 Object Design Trade-offs

Scalability versus Production Speed: This application makes use of a templating engine in order to build and render HTML files in the backend using Python language instead of using JavaScript in the frontend. This technique greatly increases the production speed and production cost since it requires less developers and less effort. However; it also requires a lot more computation to be made on the server-side since a request has to be sent even for making slight changes on the boundaries and little to no computation is done in the users’ machine. Consequently; as the website traffic gets hotter, the performance drops more significantly compared to an API approach. However; since this application is meant for public use and there is a challenging time constraint, it is decided that production speed is more important than scalability

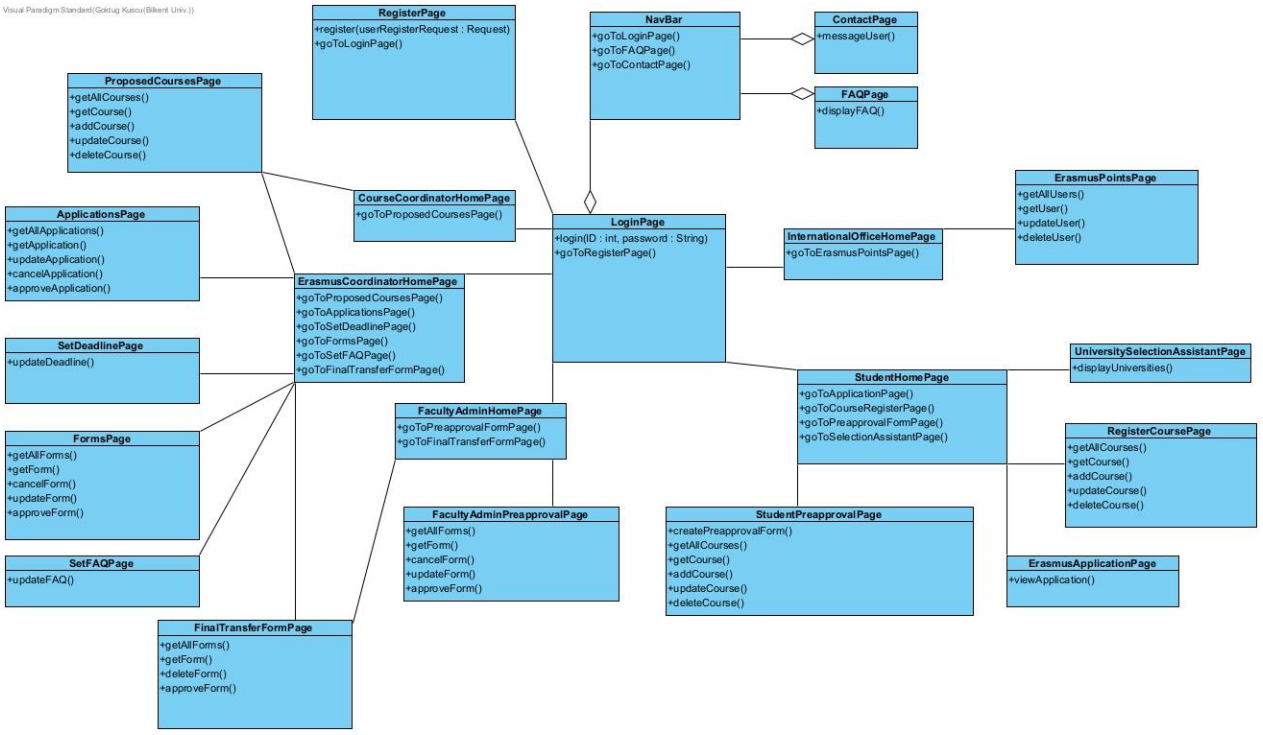
Maintainability vs Flexibility: In this application; the Model View Template architecture is used instead of the classic Model View Controller architecture. Since the views are coupled with the models in Model View Template architecture, making changes on the application is easier. However; this situation also requires the developers to be more in touch with each other's code pieces which makes it harder to resolve conflicts and failure, hence it is a disadvantage when it comes to maintainability. It is decided that flexibility is more important than maintainability since this application is a small-scale application which is being developed by a small team and the likelihood of having misinterpretations is quite high.,

3.2 Final object design

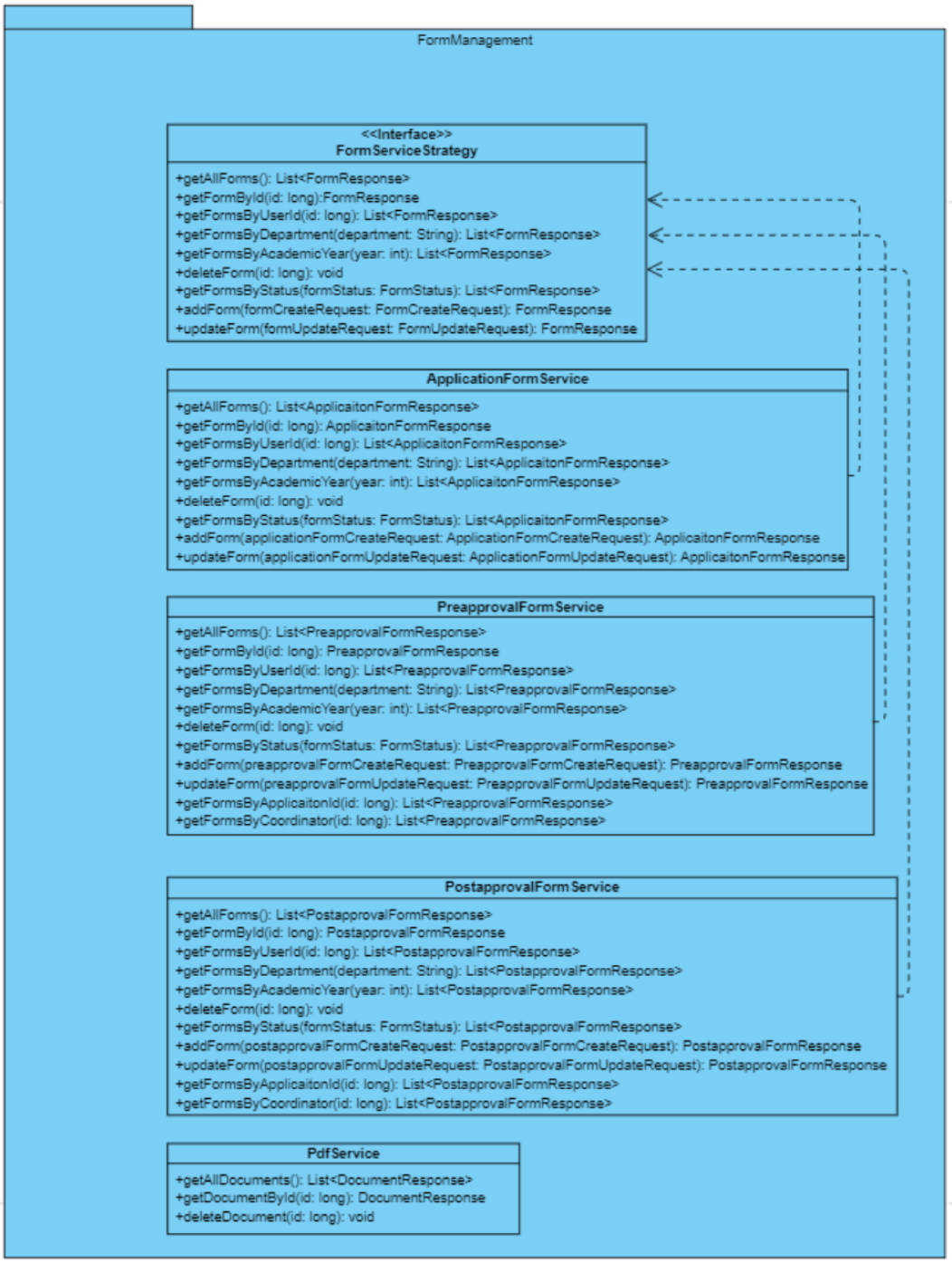


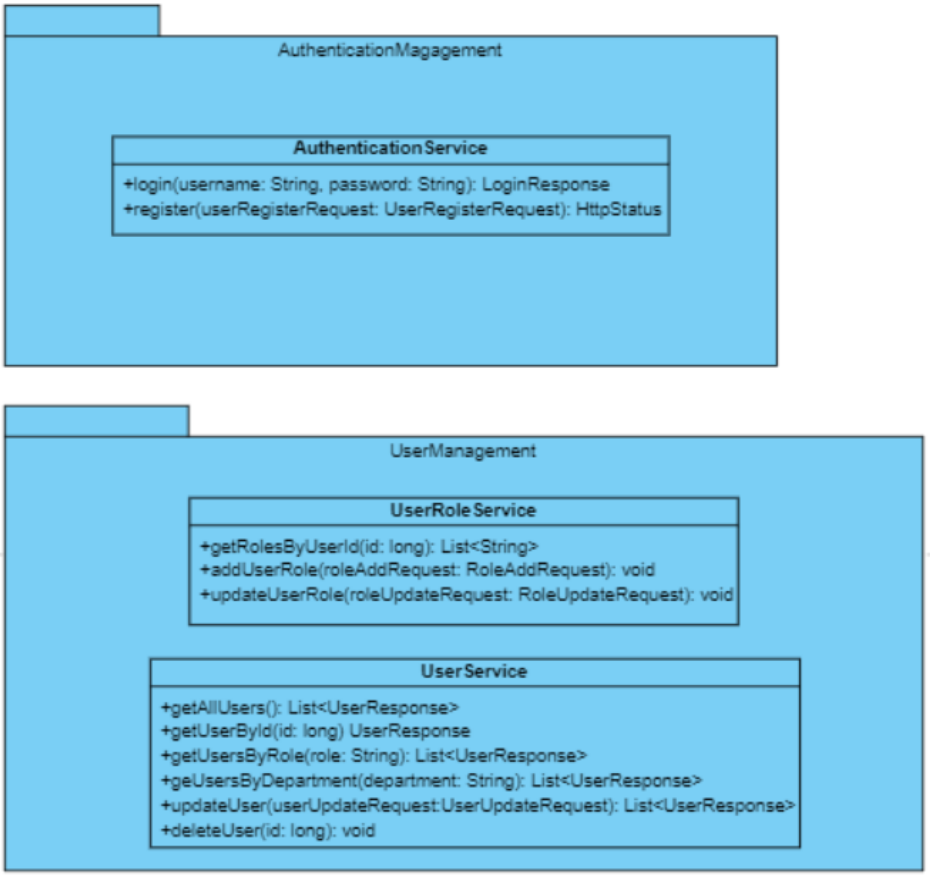
3.3. Layers

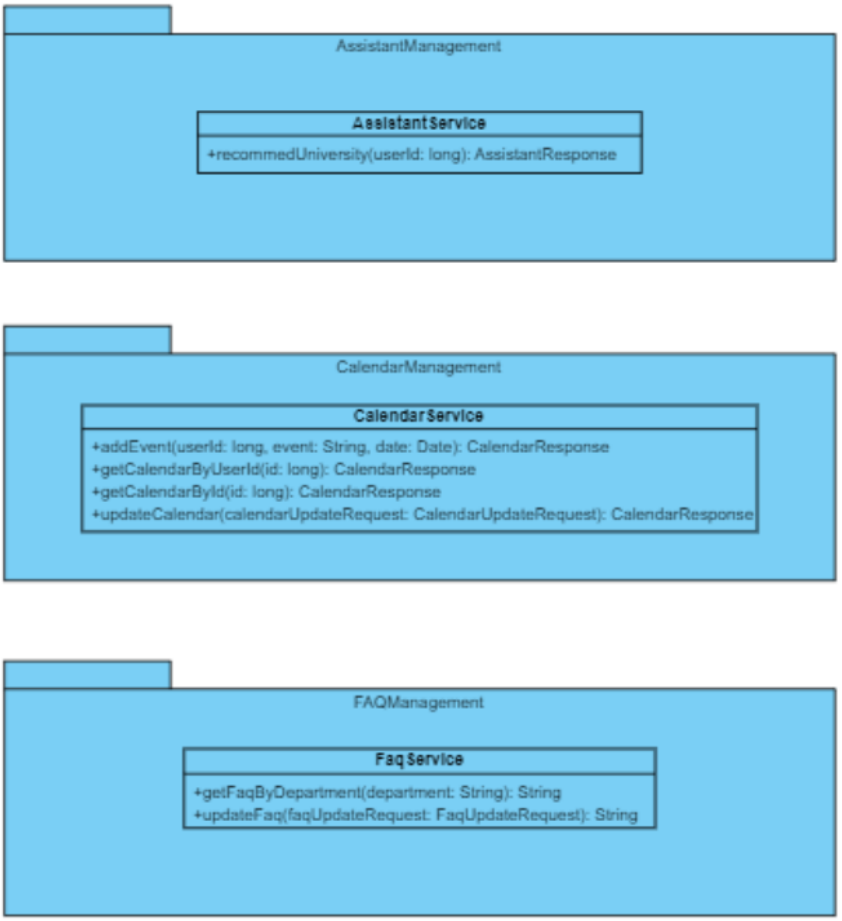
3.3.1 User Interface Management Layer

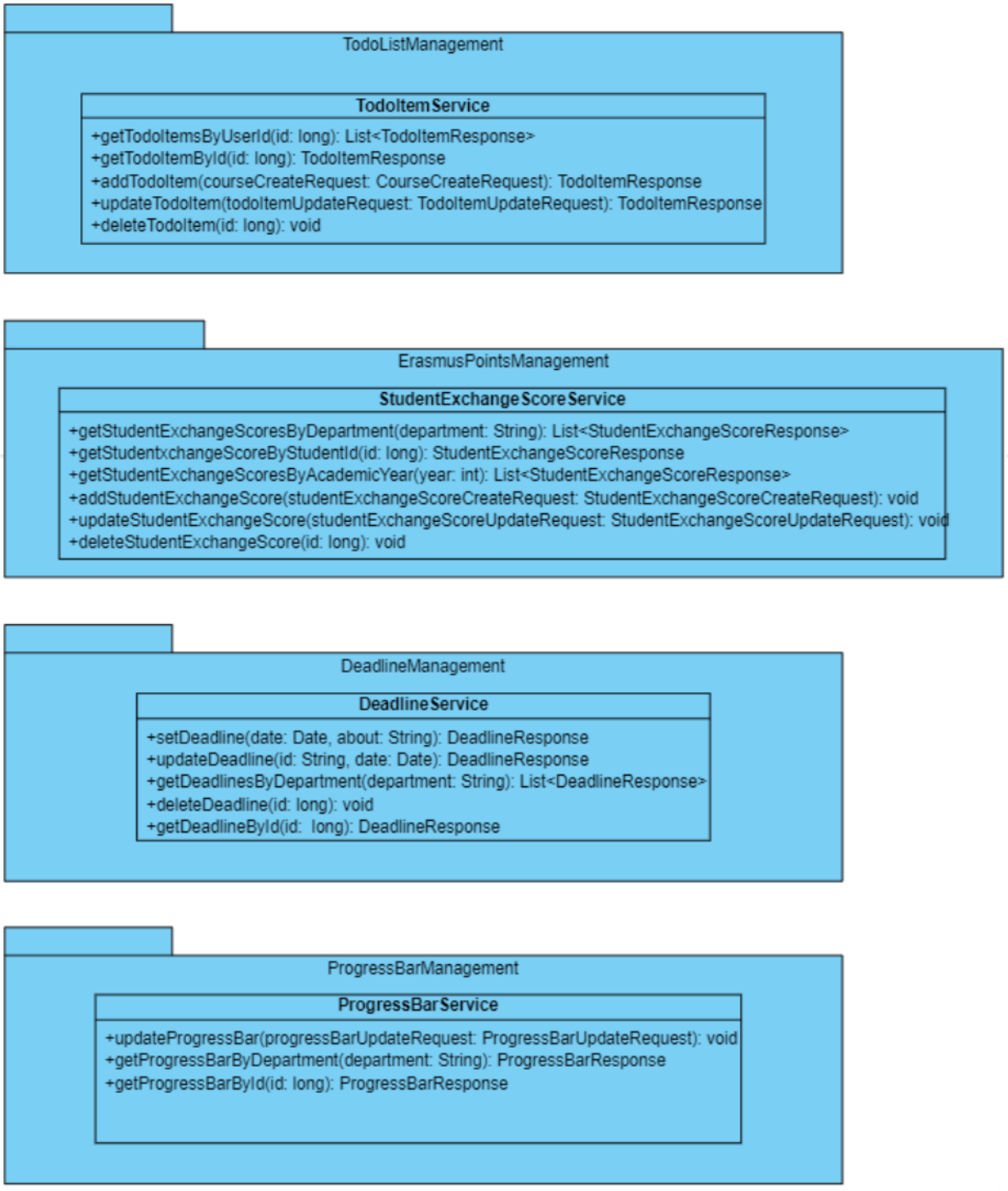


3.3.2 Web Server Layer

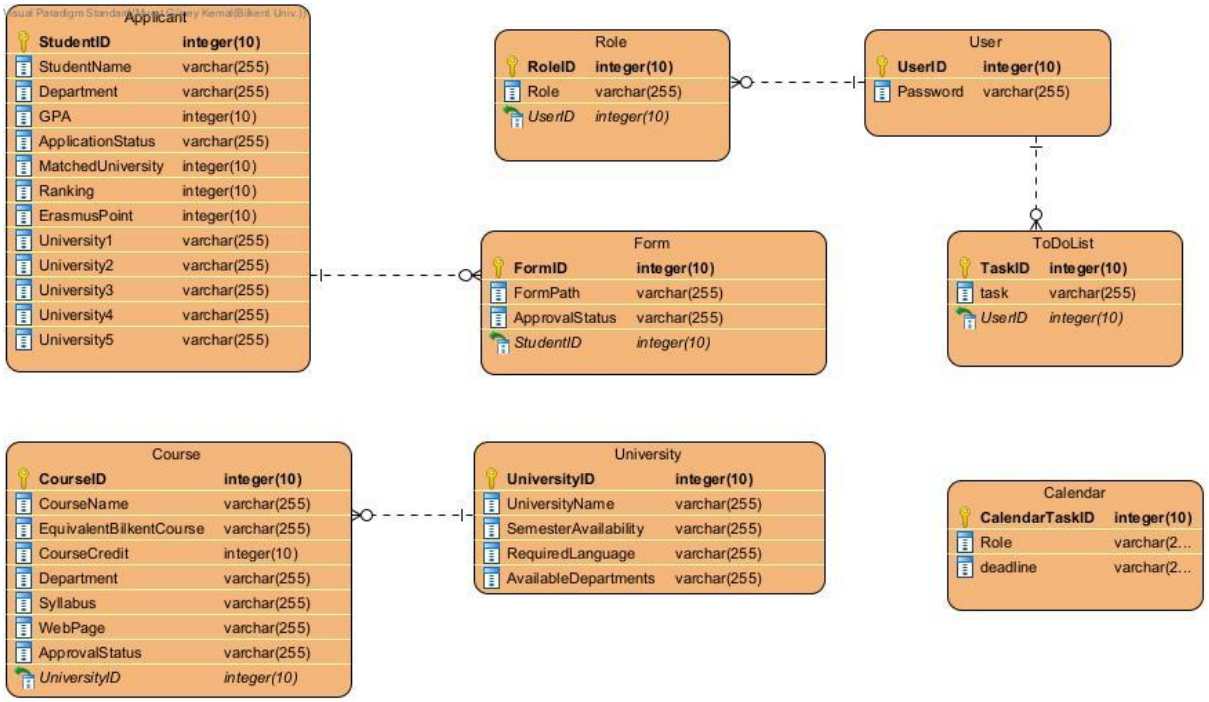


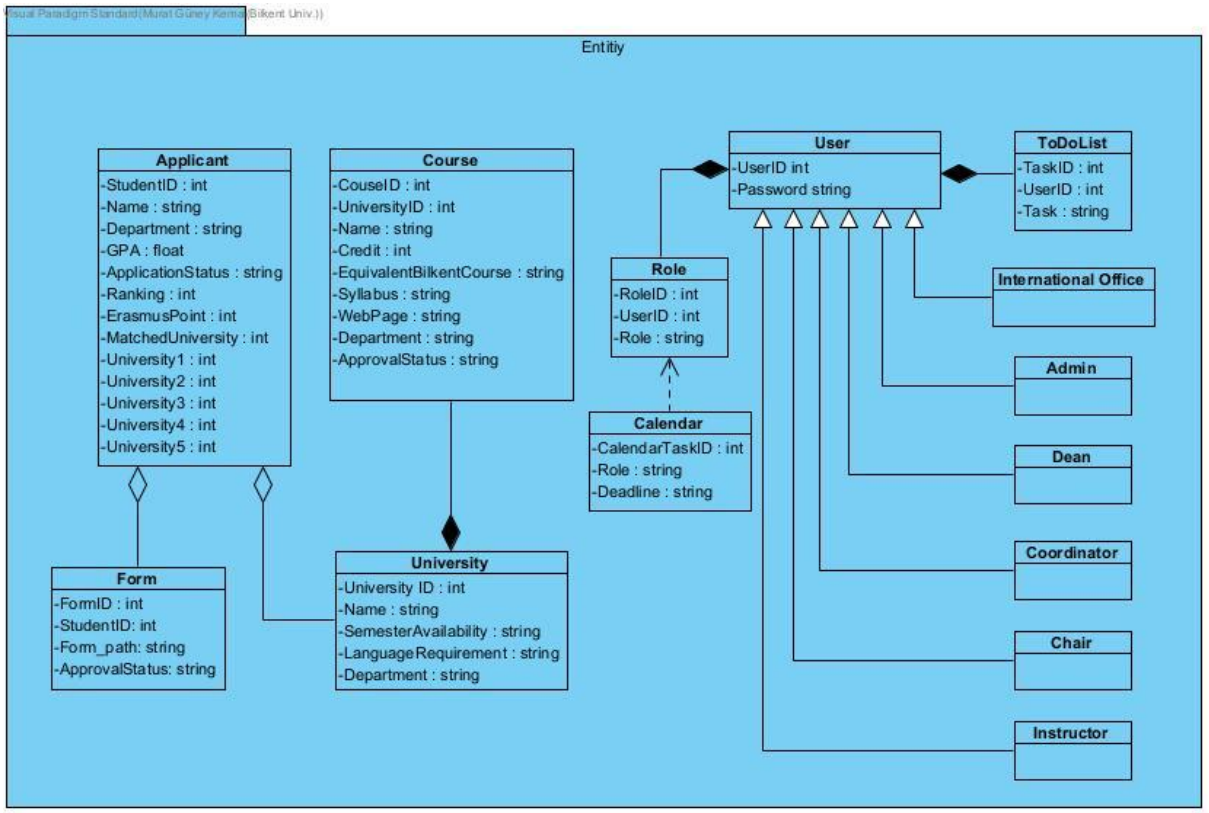

AuthenticationMagagement
AuthenticationService
+login(username: String, password: String): LoginResponse
+register(userRegisterRequest: UserRegisterRequest): HttpStatus
UserManagement
UserRoleService
+getRolesByUserId(id: long): List<String>
+addUserRole(roleAddRequest: RoleAddRequest): void
+updateUserRole(roleUpdateRequest: RoleUpdateRequest): void
UserService
+getAllUsers(): List<UserResponse>
+getUserById(id: long) UserResponse
+getUsersByRole(role: String): List<UserResponse>
+geUsersByDepartment(department: String): List<UserResponse>
+updateUser(userUpdateRequest:UserUpdateRequest): List<UserResponse>
+deleteUser(id: long): void
UniversityMagagement
UniversityService
+getAllUniversities(): List<UniversityResponse>
+getUniversityById(id: long): UniversityResponse
+getUniversitiesByDepartment(department: String): List<UniversityResponse>
+getUniversitiesByCoordinator(id: long): List<UniversityResponse>
+getUniversitiesByExchangeSemesterAvailability(semseter: String): List<UniversityResponse>
+addUniversity(universityCreateRequest: UniversityCreateRequest): UniversityResponse
+updateUniversity(universityUpdateRequest: UniversityUpdateRequest): UniversityResponse
+deleteUniversity(id: long): void
CourseManagement
CourseService
+getAllCourses(): List<CourseResponse>
+getCourseById(id: long): CourseResponse
+getCoursesByUniversity(university: String): List<CourseResponse>
+getCoursesByDepartment(department: String): List<CourseResponse>
+getCoursesByApproval(isApproved: boolean): List<CourseResponse>
+addCourse(courseCreateRequest: CourseCreateRequest): CourseResponse
+updateCourse(courseUpdateRequest: CourseUpdateRequest): CourseResponse
+deleteCourse(id: long):void


AssistantManagement
AssistantService
+recommedUniversity(userId: long): AssistantResponse
CalendarManagement
CalendarService
+addEvent(userId: long, event: String, date: Date): CalendarResponse
+getCalendarByUserId(id: long): CalendarResponse
+getCalendarById(id: long): CalendarResponse
+updateCalendar(calendarUpdateRequest: CalendarUpdateRequest): CalendarResponse
FAQManagement
FaqService
+getFaqByDepartment(department: String): String
+updateFaq(faqUpdateRequest: FaqUpdateRequest): String


TodoListManagement
TodoItemService
+getTodoItemsByUserId(id: long): List<TodoItemResponse>
+getTodoItemById(id: long): TodoItemResponse
+addTodoItem(courseCreateRequest: CourseCreateRequest): TodoItemResponse
+updateTodoItem(todoItemUpdateRequest: TodoItemUpdateRequest): TodoItemResponse
+deleteTodoItem(id: long): void
ErasmusPointsManagement
StudentExchangeScoreService
+getStudentExchangeScoresByDepartment(department: String): List<StudentExchangeScoreResponse>
+getStudentxchangeScoreByStudentId(id: long): StudentExchangeScoreResponse
+getStudentExchangeScoresByAcademicYear(year: int): List<StudentExchangeScoreResponse>
+addStudentExchangeScore(studentExchangeScoreCreateRequest: StudentExchangeScoreCreateRequest): void
+updateStudentExchangeScore(studentExchangeScoreUpdateRequest: StudentExchangeScoreUpdateRequest): void
+deleteStudentExchangeScore(id: long): void
DeadlineManagement
DeadlineService
+setDeadline(date: Date, about: String): DeadlineResponse
+updateDeadline(id: String, date: Date): DeadlineResponse
+getDeadlinesByDepartment(department: String): List<DeadlineResponse>
+deleteDeadline(id: long): void
+getDeadlineById(id: long): DeadlineResponse
ProgressBarManagement
ProgressBarService
+updateProgressBar(progressBarUpdateRequest: ProgressBarUpdateRequest): void
+getProgressBarByDepartment(department: String): ProgressBarResponse
+getProgressBarById(id: long): ProgressBarResponse

3.3.3. Data Management Layer





This layer represents the Data Management Layer of the application. This layer interacts with the Web Service Layer. The layer contains Database and its Entities. Database subsystems handle database communication with the help of related services. The Entities represent the tables of the database.

3.4 External Packages

Flask-login

This package provides user session management and helps with authorization and authentication. It handles the basic tasks of logging in and logging out.

Flask-SQLAlchemy

This package is an extension of Flask which adds SQLAlchemy support. It allows the use of Object Relational Mapping (ORM) for mapping database tables to the entity objects.

Flask-Uploads

This package provides a more flexible and efficient way of handling file uploads and file downloads

Flask-Admin

This package provides a built-in admin interface for a Flask application.

Scikit-learn

This package provides built-in machine learning models which are going to be used for training the university selection assistant

Jinja2

This package provides an extensible templating engine which makes the use of Python language in the backend replace the use of JavaScript in the frontend, hence increasing the production speed.