POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Design Document

Students&Companies

Author:
Riccardo Bonfanti
Jie Chen

Student ID: 273115 276324
Advisor: Prof.Elisabetta Di Nitto
Academic Year: 2024–2025

Deliverable Information

Deliverable:	DD
Title:	Design Document
Authors:	Riccardo Bonfanti, Jie Chen
Version:	1.0
Date:	22-December-2024
Download page:	https://github.com/JieCver1/BonfantiChen
Copyright:	

Contents

Deliverable Information **ii**

Contents **iii**

1 Introduction **1**

A Purpose . . . 1
B Scope . . . 1
C Definition,acronyms,abbreviations . . . 1
C.1 Definitions . . . 1
C.2 Acronyms . . . 2
C.3 Abbreviations . . . 3
D Revision History . . . 3
E Reference Documents . . . 3
F Document Structure . . . 3

2 Architectural Design **5**

A Overview . . . 5
B Components view . . . 6
B.1 High-level components and interactions . . . 7
B.2 Low-level components and interactions . . . 8
C Deployment view . . . 14
D Component interfaces . . . 16
E Runtime view . . . 22
F Selected architectural styles and patterns . . . 33
F.1 Three-Tiered Architecture . . . 33
F.2 RESTful API . . . 34
F.3 Model-View-Controller (MVC) Pattern . . . 34
G Other design decisions . . . 35

G.1 Observer Pattern 35
G.2 State Pattern 35
G.3 Availability 35
G.4 Scalability 35
G.5 Security 36

3 User Interface Design **37**
A User Interface Design 37
A.1 Welcome Page 37
A.2 Register Page 37
A.3 Header bar 39
A.4 Student's view 40
A.5 Company's view 47
A.6 Student and Company's view 53
A.7 University's view 54

4 Requirements Traceability **57**

5 Implementation, Integration and Test Plan **61**
A Implementation Plan 61
B Integration Plan 62
C System Testing 67

6 Effort Spent **69**

7 References **71**

1 | Introduction

A. Purpose

The purpose of this document is to provide a detailed description of Student&Companies. It will help developers to implement the required system features and it should provide the customer with a clear description of the system, allowing him to verify that it meets the specified requirements.

B. Scope

Student&Companies is a platform that connects students, companies, and universities to facilitate the internship research, announcement and selection process. The platform provides services such as internship announcements, profile management, a recommendation system, interview management, and performance feedback. The platform is designed to be user-friendly and easy to use for students, companies, and universities. It is a web-based application that can be accessed from any device with an internet connection.

C. Definition,acronyms,abbreviations

C.1. Definitions

- **Student:** A person who is looking for internships.
- **Company:** An organization which wants to announce internship opportunities to students.
- **University:** An educational institution that is related to students and their internships.
- **User:** A generic term for students, companies, and universities who use the platform.
- **Candidate:** A term for students whose applications are selected and that will take

part in the interview process.

- **Internship:** A opportunity offered by companies to students to gain practical experience in a real job environment.
- **CV:** Curriculum Vitae, a document that contains all necessary information about students to be able to apply for internships.
- **Recommendation:** A suggestion made by the platform to students and companies based on statistical analyses and keyword searches.
- **Interview:** A questionnaire form, that can be followed by an external meeting between students and companies, to evaluate the student preparation and make him understand what the company is looking for.
- **Feedback:** Helpful information written by students and companies about their internship experiences to improve a performance of the two parties.
- **Notification:** A message sent by the platform to inform students and companies about important events, such as new internship offers, matching CVs, interview results etc.
- **Interview:** A meeting between students and companies to decide an assignment of the internship offer.
- **Platform:** The Students&Companies (S&C) system that provides the services to students, companies, and universities about internships.
- **Keyword:** A significant word or tag used to describe content, such as the skills, experiences, and preferences of students and companies.
- **Comment:** The text that is written by students and companies to provide feedback or complaints about their internship experiences.
- **Complain:** A text that expresses dissatisfaction, issues, or annoyance about the internship experiences. It will be treated as a synonym of feedback in this document.

C.2. Acronyms

- **S&C:** Students&Companies
- **CV:** Curriculum Vitae
- **UI:** User Interface
- **UX:** User Experience

- **API:** Application Programming Interface
- **HTTPS:** Hypertext Transfer Protocol Secure
- **TLS:** Transport Layer Security
- **REST:** Representational State Transfer

C.3. Abbreviations

- **CO:** Company
- **ST:** Student
- **UNI:** University

D. Revision History

E. Reference Documents

- Assignment RDD AY 2024–2025.

F. Document Structure

This document is structured as follows:

- **Section 1: Introduction**
  It contains the summary of main architectural styles and choices that drive the design of the system. It also provides a brief description of the scope of the project as already mentioned in the RASD. In addition, it include the specification of definitions, acronyms, and abbreviations of the terms used in this document. At the end, it notes the revision history for updates to the document and the reference documents that were used during the development of this document.
- **Section 2: Architectural Design**
  It provides a high-level overview of the system architecture, including the main components and their interactions. It also describes in detail each component by specifying the interfaces using the UML component diagram and list of the methods that each component can perform Then the deployment view of the system is presented with the description of the hardware and software components and the behavior of the system in runtime view is presented with the sequence diagrams

that illustrate the interactions. Finally the architectural styles and patterns chosen for the system will be reported with a brief description of the reason for the choice.

- **Section 3: User Interface Design**
  In this section, the user interface design of the system is presented with a brief description of the interactions between each component and mockups that illustrate the main functionalities of the system. With respect to the RASD, this section provides a more detailed description of the user interface design and highlights the interactions from the perspectives of different types of users.

- **Section 4: Requirements Traceability**
  It connects the requirements of the system with the components of the architecture presented in the previous sections by providing a table that maps each requirement to the components that implement it.

- **Section 5: Implementation, Integration, and Test Plan**
  Describe the plan for the implementation, integration, and testing of the system. It includes the description of the development process and emphasizes the the order of the implementation of the components withing the specification of the reason for the choice. It also includes the description of the testing process and the aim of each test case.

- **Section 6: Effort Spent**
  The time spent by each group member on each task will be registered in this section. It will be used to present the effort dedicated by each member and to present the progress of the development of the project.

- **Section 7: References**
  The other references that not include in the reference documents will be added in this section.

2 | Architectural Design

A. Overview

S&C will be developed as a multi-tiered, client-server architecture, as shown in Figure 2.1. The system will be divided into three main layers: the presentation layer, the application layer, and the data layer. The presentation layer will be responsible for managing the user interface and the user interaction. The application layer will be responsible for managing the application logic. The data layer will be responsible for managing the data storage and the data access.

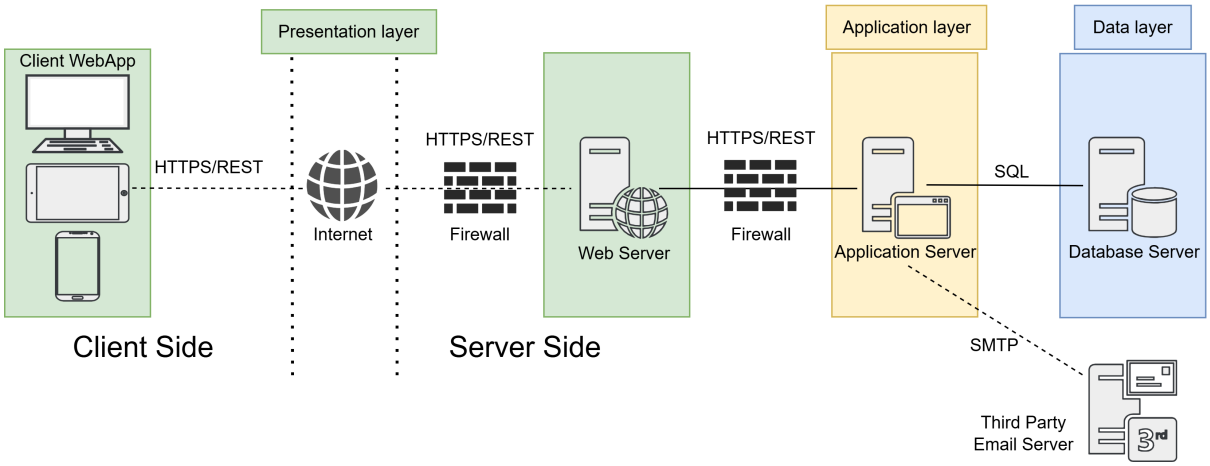


Figure 2.1: S&C architectural overview

In the following paragraph we describe each tier presented in Figure 2.1: and what layer it deploys:

Client Side

- **Web App:** It is the user interface. It will be responsible for managing the user interaction. This means that it will be responsible for hosting part of the presentation layer.

Server Side

- **Firewall:** It will be responsible for managing the security of the system, by filtering the incoming and outgoing traffic and restricting the access based on predefined rules. It will be placed between the Web Server and the Internet and between the Application Server and the Web Server. In this way, the web server will reside in a DMZ (Demilitarized Zone), while the application server will reside in a protected internal network.
- **Web Server:** It serves as a gateway between the client and the application server (backend). It will be responsible for hosting part of the presentation layer. For example, it will be responsible for serving the web pages to the client, handling requests routing to the application server, managing load balancing, and handling security.
- **Application Server:** It will be responsible for managing the application logic. This means that it will host the application layer. For example, it will be responsible for processing the client requests, execute the business logic, and coordinates with the database and email server.
- **Database Server:** It will be responsible for managing the data storage and the data access. This means that it will host the data layer. For example, it will be responsible for storing and retrieving the application data, and executing the database queries.
- **Mail Server:** It will be responsible for managing the email communication. This means that it will be responsible for sending emails to the users. It is triggered by the application server.

The Figure 2.1 also shows how the tiers interact with each other. The Web App interacts with the Web Server through HTTPS/REST requests; the Web Server interacts with the Application Server through HTTPS/REST calls; the Application Server interacts with the Database Server through SQL queries; and the Application Server interacts with the Mail Server through SMTP requests.

B. Components view

In this section we describe the components of the S&C platform, their interactions, and the interfaces they expose. For understandability we divide the components into two categories: high-level components and low-level components.

B.1. High-level components and interactions

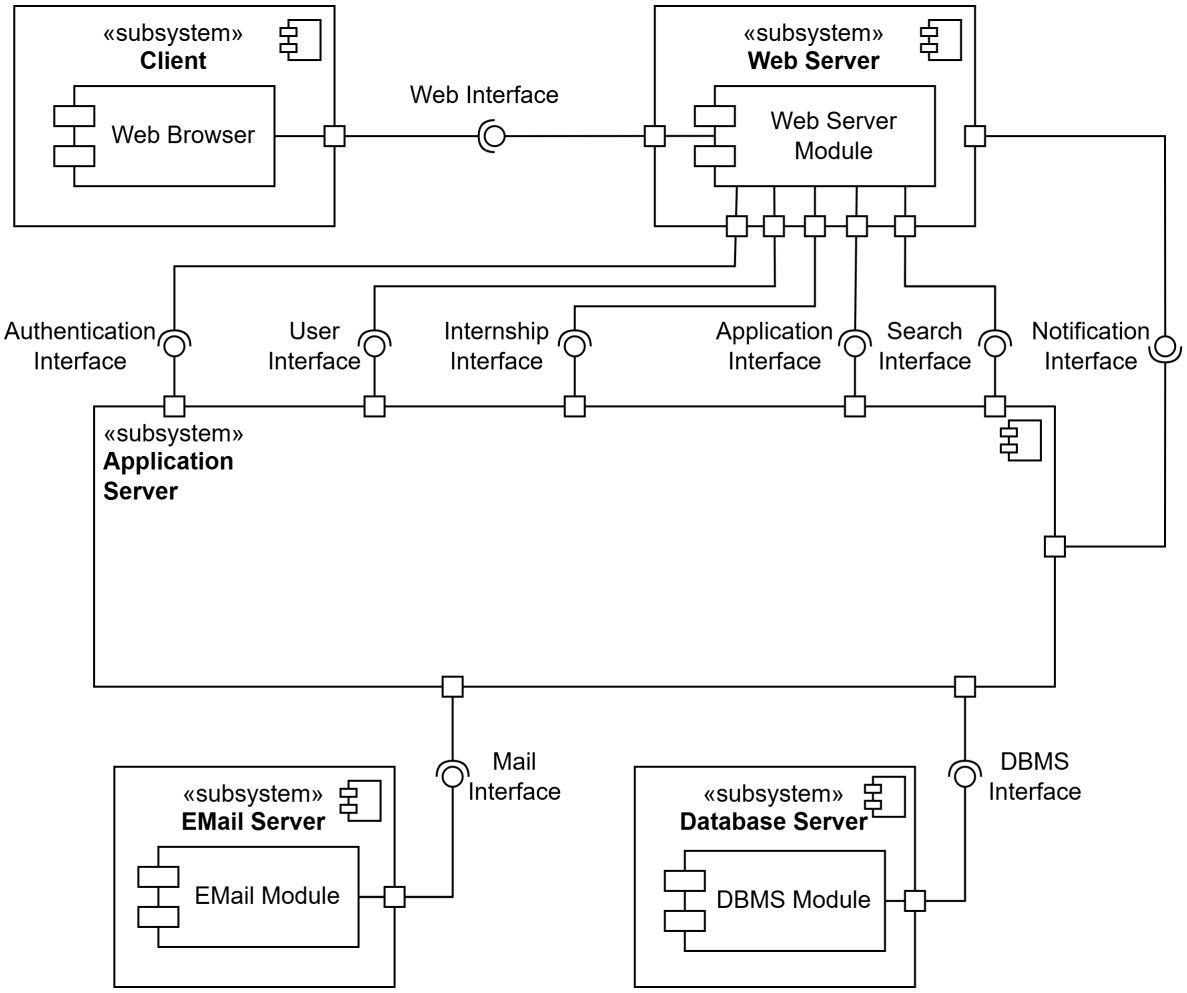


Figure 2.2: High Level Component Diagram

In the figure above we show the high-level components of the S&C platform. In particular we show which are the components that are run on each tier of the system.

- **Web Browser:** It serves as the external interface for the user to interact with the platform. It sends requests to the Web Server and receives responses from it. It is responsible for rendering the web pages and handling user input.
- **Web Server Module:** It serves as a gateway between the client and the application server. It handles load balancing and security, ensuring that requests are properly routed through the interfaces that are provided by the application server. It also provides the Notification Interface, which allows the Web Server Module to send notifications to users.

- **Application Server:** Hosts all the system's internal components and provides the following interfaces: Authentication Interface, User Interface, Internship Interface, Application Interface, and Search Interface.
- **DBMS:** It communicates with the application server through the DBMS Interface and is responsible for managing the data storage and retrieval operations. It ensures data consistency and integrity, and provides mechanisms for data backup and recovery.
- **Email Server:** It is responsible for the sending of emails for user registration confirmation. The Application Server interacts with the Email Server via the Email Interface to send these emails.

B.2. Low-level components and interactions

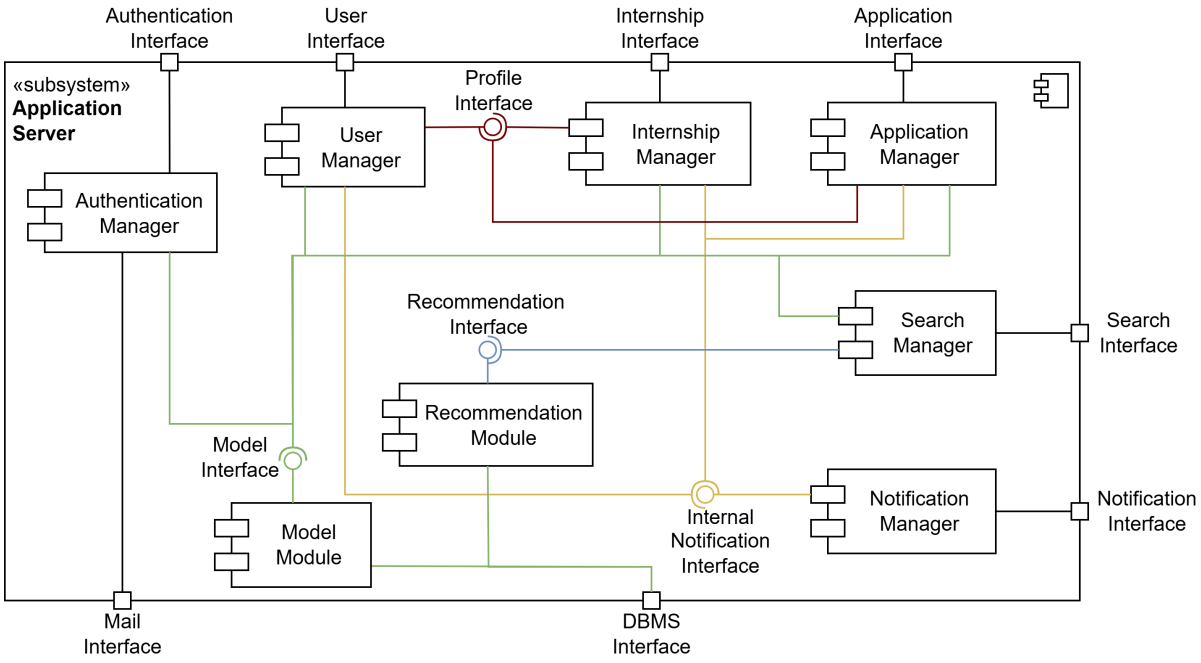


Figure 2.3: Low Level Component Diagram

In Figure 2.3, we show the low-level components of the platform. In particular, these components provide more detailed insights into the internal workings of the Application Server. The low-level components are divided as follows:

Authentication Manager

This component manages user authentication and authorization processes. It interacts with the Model Module through the provided interface in order to store and retrieve information from the database. It includes the following subcomponents:

- **Registration Manager:** It is responsible for managing the user registration process, including the creation of new user accounts and sending verification emails through the Mail Interface.
- **Login Manager:** It is responsible for managing the user login process, including verifying user credentials and maintaining user sessions.

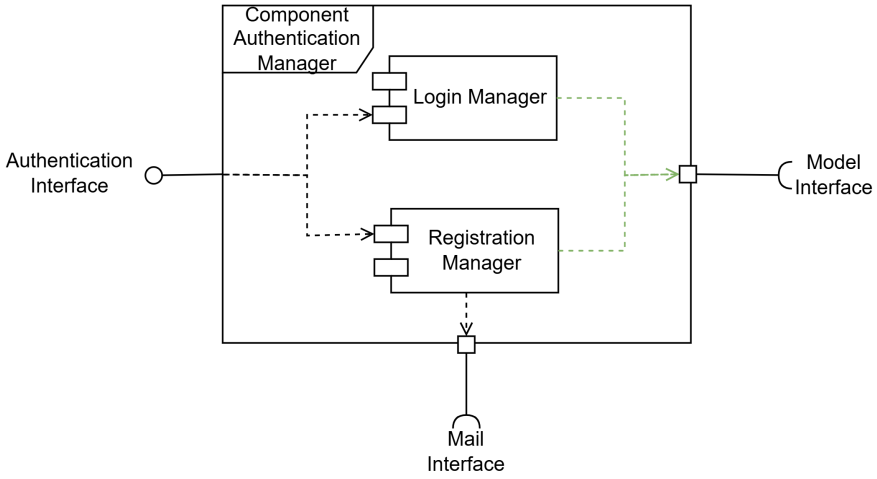


Figure 2.4: Authentication Manager

User Manager

This component is responsible for managing user-related operations, such as user profile management and visualization, and handling user-specific notifications. It interacts with the Model Module to store and retrieve user information from the database.

- **View Profile Manager:** Provides users with an overview of their activities and notifications. It provides a Profile Interface that allows other components to access user data.
- **Profile Modification Manager:** Allows users to view and edit their personal information.

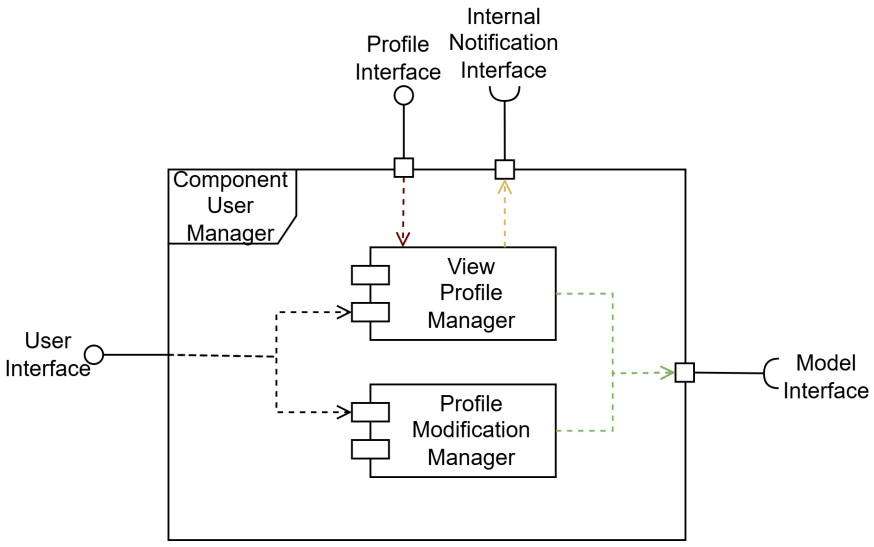


Figure 2.5: User Manager

Internship Manager

The Internship Manager is the component used by the S&C platform to manage internship-related operations. It interacts with the Model Module to store and retrieve internship information and related data (such as chat messages and feedback) from the database. It includes the following subcomponents:

- **Creation Manager:** Manages the creation of internship and their posting.
- **View Internship Information Manager:** Manages the display of internship information when users request them.
- **Selection Manager:** Manages the selection process for internships. It allows companies to go through the list of applications and select candidates for their internships. After the interview phase, it also allows them to decide whether to accept or reject a candidate for the internship. It interacts with the Internal Notification Interface to notify students in case they are selected for interviews and in case they are accepted or rejected for an internship.
- **Chat Manager:** Manages the chat functionality between student and company. It interacts with the Notification Manager through the provided interface to notify users when they receive a new message.
- **Feedback Manager:** Manages the feedback system for internships, allowing both companies and students to provide feedback on their internship experience. It interacts with the Notification Manager through the provided interface to notify users

when they receive new feedback.

The View Internship Information Manager, Chat Manager, and Feedback Manager also interact with the View Profile Manager through the provided Profile Interface, since they need to show to both students and companies other users profile information.

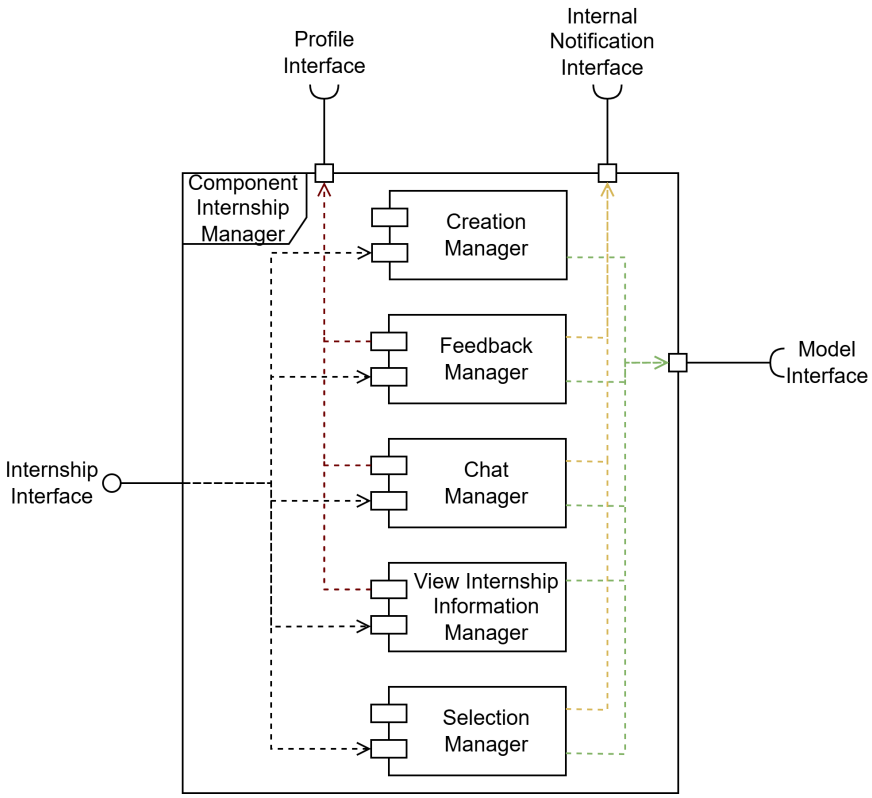


Figure 2.6: Internship Manager

Application Manager

The Application Manager is the component used by the S&C platform to handle every aspect of the application process for internships. It interacts with the Model Module to store and retrieve application information and related data from the database. It includes the following subcomponents:

- **Submission Manager:** Handles the submission of applications for internships. It allows students to apply for internships and submit their applications. It also allows candidates who have been accepted for the internship to accept or reject the offer from the company. It interacts with the Internal Notification Interface to notify students when their application status changes.
- **View Application Information Manager:** Manages the display of application

information to users who request them and have the rights to access them.

- **Interview Manager:** Manages the interview process for applications; it allows companies to create and modify interview forms and record the results of interviews. It interacts with the Internal Notification Interface to notify students when they are selected for interviews, and notify companies when the results of the interviews are available.
- **Questionnaire Manager:** Manages the questionnaire system for applications, allowing students to respond to interview questions and both students and companies to view the responses. It interacts with the Internal Notification Interface to notify companies when a student has completed the interview form.

The View Application Information Manager, Interview Manager, and Submission Manager also interact with the View Profile Manager through the provided Profile Interface, since they need to show to both students and companies other users profile information.

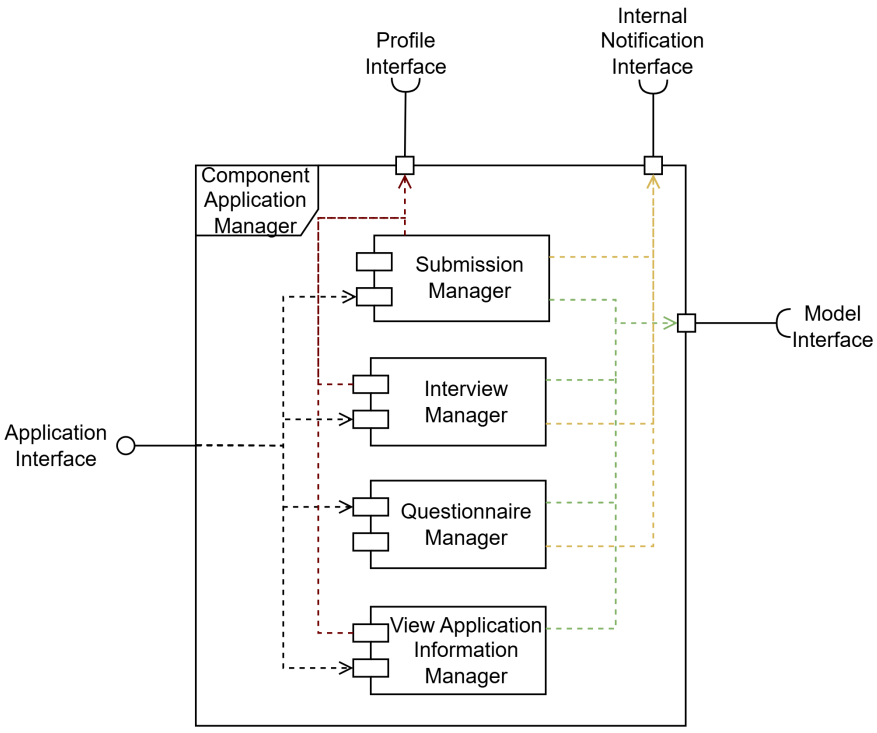


Figure 2.7: Application Manager

Search Manager

The Search Manager is responsible for managing the search functionality of the platform, allowing users to search using filters or keywords. It interacts with the Model Module

to retrieve search results from the database. It also interacts with the Recommendation Module to provide personalized, real-time recommendations to users.

Notification Manager

The Notification Manager is responsible for managing the notification system of the platform, allowing users to receive real-time updates on their activities. It provides an Internal Notification Interface that allows other components to send notifications to users.

Recommentation Module

The Recommendation Module is responsible for generating personalized recommendations for users based on their interests, skills, and activities. It interacts directly with the DBMS Server to retrieve the necessary information from the database in order to provide recommendations to users. It also interacts with the Search Manager to provide real-time recommendations based on search results.

Model Module

The Model Module is responsible for managing the data storage and retrieval operations of the platform. It interacts with the DBMS to store and retrieve information from the database. It provides the Module Interface that allows other components to access the data stored in the database.

C. Deployment view

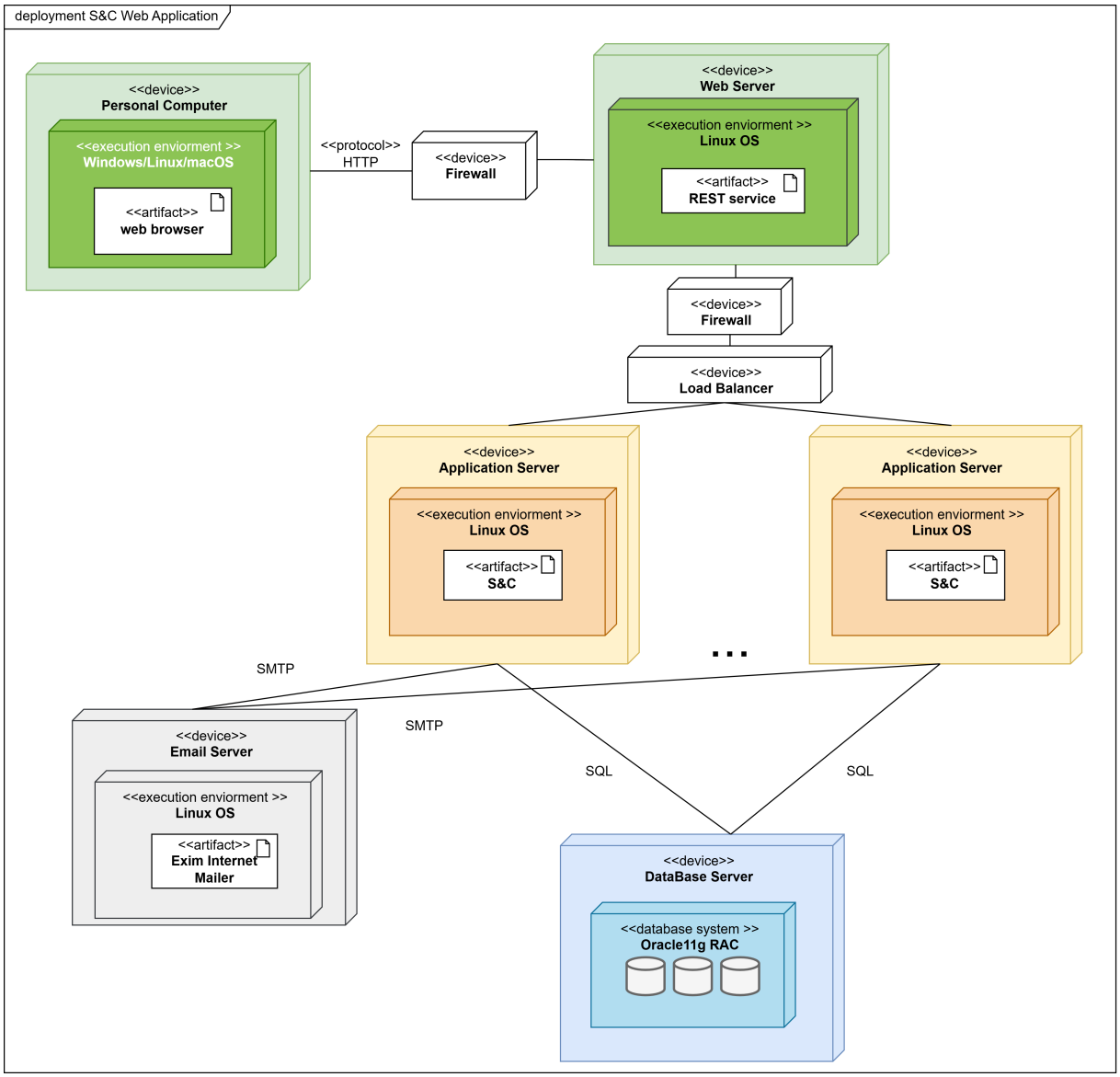


Figure 2.8: S&C deployment diagram

The infrastructure of the S&C platform is described in below using a deployment diagram and a description of the components and their interactions. As described at the beginning of this document, the system is divided into three layers: the presentation layer, the application layer, and the data layer. The deployment diagram in Figure 2.8 shows the physical distribution of the components across different servers and the communication between them.

- **Personal Computer:** Anyone interested in using the platform can access it through

any type of personal computer. Users can also access the platform using any device capable of running a web browser. This component communicates and interacts with the Web Server via HTTPS protocols and RESTful API services.

- **Web Server:** The Web Server hosts the web application and serves web pages to users. It handles HTTP requests from clients and forwards them to the Application Server. Together with the Firewall and Load Balancer, it ensures the security, scalability, and availability of the system. It acts as a gateway between the client and the Application Server.
- **Application Server:** This component contains the core application logic of the platform, managing the operations necessary to provide services and functionalities to users. Thanks to RESTful API services and the Load Balancer, it can handle multiple requests simultaneously without the risk of overloading or crashing. It interacts with the Database Server to access or store data, such as recording a student's new internship. Additionally, it communicates with the Mail Server, particularly during user registration, to authenticate email addresses and verify user identities.
- **Database Server:** Most primary operations of the platform require frequent interaction with the Database Server. This component manages personal data, tracks the progress of applications and internships, stores chat histories between users, and more. The Application Server retrieves or stores data using SQL queries.
- **Mail Server:** During the registration process, the platform requires users to verify their email addresses to confirm their identity and complete account creation. The Application Server interacts with the Mail Server via SMTP requests to send verification emails.
- **Firewall:** The Firewall protects the core components of the system by filtering incoming and outgoing traffic and restricting access based on predefined rules. For example, if an unauthorized individual attempts to access the Application Server and modify data in the Database Server without proper credentials, the Firewall blocks the attempt, preventing unauthorized access.
- **Load Balancer:** The Load Balancer distributes incoming client requests to ensure that no one application server is overwhelmed to operate at peak efficiency. It is placed between the Web Server and the Application Server, in order to optimize the performance and availability of the system.

D. Component interfaces

In the following paragraphs, we describe the interfaces of the main components of the platform specifying the methods that each component can perform. Note:

- *userId* are unique identifiers generated by the system for each user once they register on the platform and are used to access the data related to that user.

- *userTypes* enumeration is used to distinguish between the different types of users that can register on the platform: Student, Company, University.

- *SearchType* enumeration used to distinguish between the different types of searches that can be performed on the platform: Internship and User Profile.

- *NotificationType* enumeration used to distinguish between the different types of notifications that can be sent on the platform in order to handle them in a different way.

- *InternshipID* is a unique identifier generated by the system for each internship published successfully and is used to access the data related to that internship.

- *ChatID* is a unique identifier generated by the system for each chat created between two users and is helpful to access the data related to that chat more easily.

- *ApplicationID* is a unique identifier generated by the system for each application submitted by a student for an internship and is used to access the data related to that application and take a track of its status.

Registration Manager

- registration (String email, String password, String name, String surname, University university, List[Field] interests, List[String] skills): Boolean
- registration (String email, String password, String legalName, String EIN, String department, List[Field] fields): Boolean
- registration (String email, String password, String name, String legalName): Boolean
- generateUserId (): int
- confirmRegistration (String email): Boolean

Login Manager

- login (String email, String password): Boolean

User Manager

Profile Modification Manager

- uploadCV (int userId, File cv): Boolean
- updatePersonalInfo (int userId, String name, String surname): Boolean
- updateUniversityInfo (int userId, University university): Boolean
- updateInterestsAndSkills (int userId, List<Field> interests, List<String> skills): Boolean
- updatePassword (int userId, String oldPassword, String newPassword): Boolean
- updateLegalInfo (int userId, String legalName, String EIN, String department): Boolean
- updateProfilePicture (int userId, File profilePicture): Boolean
- addFieldOfInterest (int userId, Field field): Boolean
- removeFieldOfInterest (int userId, Field field): Boolean
- addSkill (int userId, String skill): Boolean
- removeSkill (int userId, String skill): Boolean
- checkFileFormat (File file): Boolean
- saveModifications (int userId): Boolean
- saveModifications (int userId, String email, String password, String name, String surname, University university, List[Field] interests, List[String] skills): Boolean
- saveModifications (int userId, String email, String password, String legalName, String EIN, String department, List[Field] fields): Boolean
- saveModifications (int userId, String email, String password, String legalName): Boolean

View Profile Manager

- getProfile (int userId): Profile
- getListOfEnrolledStudents (University university): List[Student]

Search Manager

- search (int userId, String keyword): List[Result] and List[Recommendation]
- searchFilterByType (int userId, SearchType type, List[Result]): List[Result]
- searchFilterByField (int userId, Field field, List[Result]): List[Result]
- searchFilterByLocation (int userId, String location, List[Result]): List[Result]
- searchFilterByTime (int userId, Date startDate, Date endDate, List[Result]): List[Result]

Notification Manager

- notify (int userId, String notification, NotificationType type): Boolean
- notify (int userId, String notification): Boolean
- notify (String internshipId, String notification): Boolean
- getNotifications (int userId): List[Notification]
- getNotificationDetails (int userId): Notification
- isRead (int userId, String notification): Boolean
- deleteNotification (int userId, String notification): Boolean
- deleteAllNotifications (int userId): Boolean

Recommendation Module

- getRecommendations (int userId): List[Recommendation]
- generateRecommendations (int userId): List[Recommendation]
- revaluateRecommendationsList (int userId, List[Recommendation] oldRecommendations): List[Recommendation]
- needRecommendationUpdate (int userId): Boolean

Internship Manager

Creation Manager

- createInternship (String title, String description, Field field, String location, Date startDate, Date endDate, int duration, String position, Date deadline): Boolean

- generateInternshipID (): String
- addInternshipInList (String internshipId, Company company): Boolean

View Internship Information Manager

- getInternshipInformation (String internshipId): Internship
- getInternshipList (Company company): List[Internship]
- getInternshipList (Student student): List[Internship]

Selection Manager

- selectStudentForInterview (String internshipId, List[Student]): Boolean
- selectStudentForInternship (String internshipId, List[Student]): Boolean
- getSelectedStudents (String internshipId): List[Student]
- updateStudentStatus (String internshipId, Student student, Status status): Boolean
- getCandidatesList (String internshipId): List[Student]
- acceptOffer (String internshipId, int userId): Boolean
- rejectOffer (String internshipId, int userId): Boolean

Feedback Manager

- writeFeedback (int userId, String internshipId, String feedback): Boolean
- getFeedback (String internshipId): List[String]

Chat Manager

- sendMessage (int receiverId, String ChatID, String message): Boolean
- getChatHistory (int userId, String ChatID): List[Message]
- createChat (int userId1, int userId2): String
- getChatID (int userId1, int userId2): String
- openChat (String ChatID): Boolean
- closeChat (String ChatID): Boolean
- getChatList (int userId): List[Chat]

Application Manager

Submission Manager

- applyForInternship (String internshipId): Boolean
- submitApplication (int userId, String internshipId): Boolean
- generateApplicationID (): String
- acceptOffer (String internshipId, int userId): Boolean
- rejectOffer (String internshipId, int userId): Boolean
- getApplicationStatus (String internshipId, int userId): Status
- updateApplicationStatus (String internshipId, int userId, ApplicationStatus status): Boolean
- getApplicationList (int userId): List[Application]

View Application Information Manager

- getApplicationList (String internshipId): List[Application]
- getApplicationStatus (String applicationID, int userId): Status

Interview Manager

- setUpInterview (int companyId, Form questionnaire, String info, List[Student] students): Boolean
- submitInterviewForm (String descriptionLetter, List[String] question): Boolean
- recordInterviewResults (String formID, int userId, ApplicationStatus InterviewResult): Boolean
- getInterviewResults (String applicationID): List[ApplicationStatus]
- addQuestion (String question): List[String]
- removeQuestion (String question): List[String]

Questionnaire Manager

- getInterviewForm (int userId, String formID): Form
- respondForm (int userId, String formID, List[String] answers): Boolean

- getFormResponses (String internshipId): List[Form]
- checkAnswerValidity (List[String] answers): Boolean

Model Module

- checkValidity (String email, String password, String name, String surname, University university): Boolean
- checkValidity (String email, String password, String legalName, String EIN): Boolean
- checkValidity (String email, String password, String legalName): Boolean
- confirmRegistration (int userId, String email, String password, String name, String surname, University university, List[Field] interests, List[String] skills): Boolean
- confirmRegistration (int userId, String email, String password, String legalName, String EIN, String department, List[Field] fields): Boolean
- confirmRegistration (int userId, String email, String password, String name, String legalName): Boolean
- checkCredentials (String email, String password): Boolean
- getUserId (String email): int
- getMatchingTuple (String keyword): List[Result]
- getRecommendations (int userId): List[Recommendation]
- retrieveProfileInfo (int userId): Profile
- retrieveInternshipInfo (String internshipId): Internship
- retrieveProfileInfo (int userId): Profile
- storeNewCV (int userId, File cv): Boolean
- checkSubmissionValidity (int userId, String internshipId): Boolean
- submitApplication (int userId, String internshipId): Boolean
- retrieveApplicationDetails (int userId, String internshipId): Application
- checkInterviewValidity (int companyId, Form questionnaire, String info, List[Student] students): Boolean
- createInterview (int companyId, Form questionnaire, String info, List[Student] students): Boolean

- retrieveInterviewForm (int formId): Form
- storeResponse (int userId, int formId, List[String] answers): Boolean
- checkSelection (String internshipId, List[Student] students): Boolean
- storeSelection (String internshipId, List[Student] students): Boolean
- checkFeedbackValidity (int userId, String internshipId, String feedback): Boolean
- storeFeedback (int userId, String internshipId, String feedback): Boolean
- retrieveChatList (int userId): List[Chat]
- retrieveChatHistory (int userId, int chatId): List[Message]
- checkValidity (int userId, int chatId, String message): Boolean
- storeMessage (int userId, int chatId, String message): Boolean
- storeInternship (String title, String description, List[Field] fields, String location, Date startDate, Date endDate, int duration, String position, Date deadline): Boolean
- checkSelectionValidity (String internshipId, List[Student] students): Boolean
- retrieveEnrolledList (int userId): List[Student]

E. Runtime view

In the following section we describe the runtime view of the S&C platform, focusing on the interactions between the components and the sequence of operations that occur during the execution of the system. We provide a sequence diagram for each of the main use cases of the platform, showing the interactions between the components and the flow of data between them. This is still a high-level description, so function names, results, errors, and other details will be modified or added during the development process.

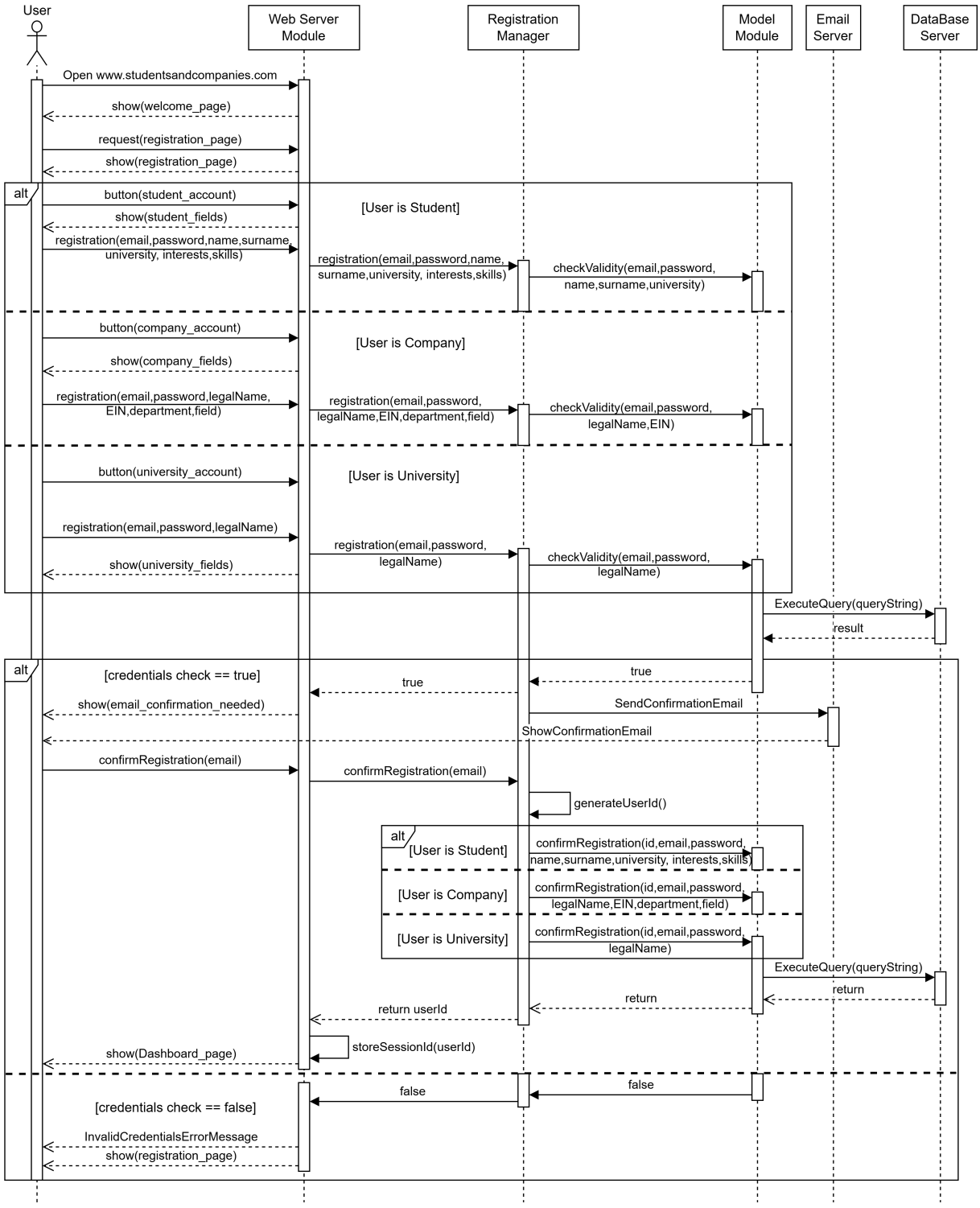


Figure 2.9: Registration to S&C Sequence Diagram

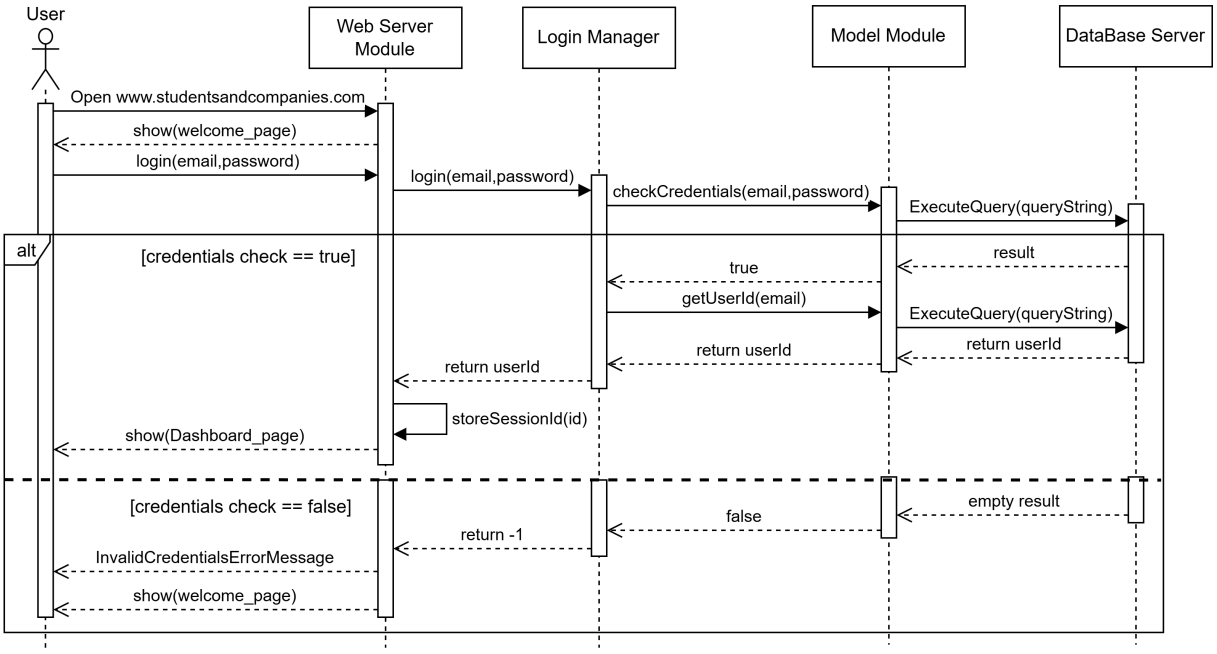


Figure 2.10: Login to S&C Sequence Diagram

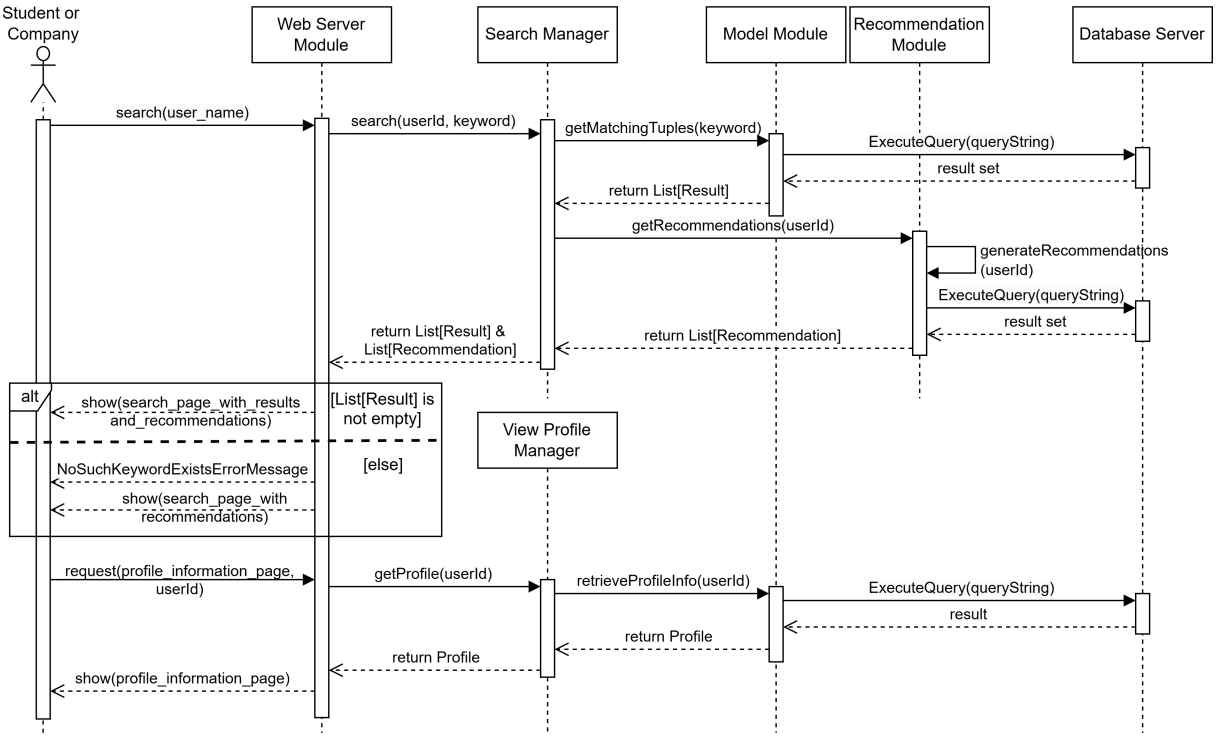


Figure 2.11: User sees profile information Sequence Diagram

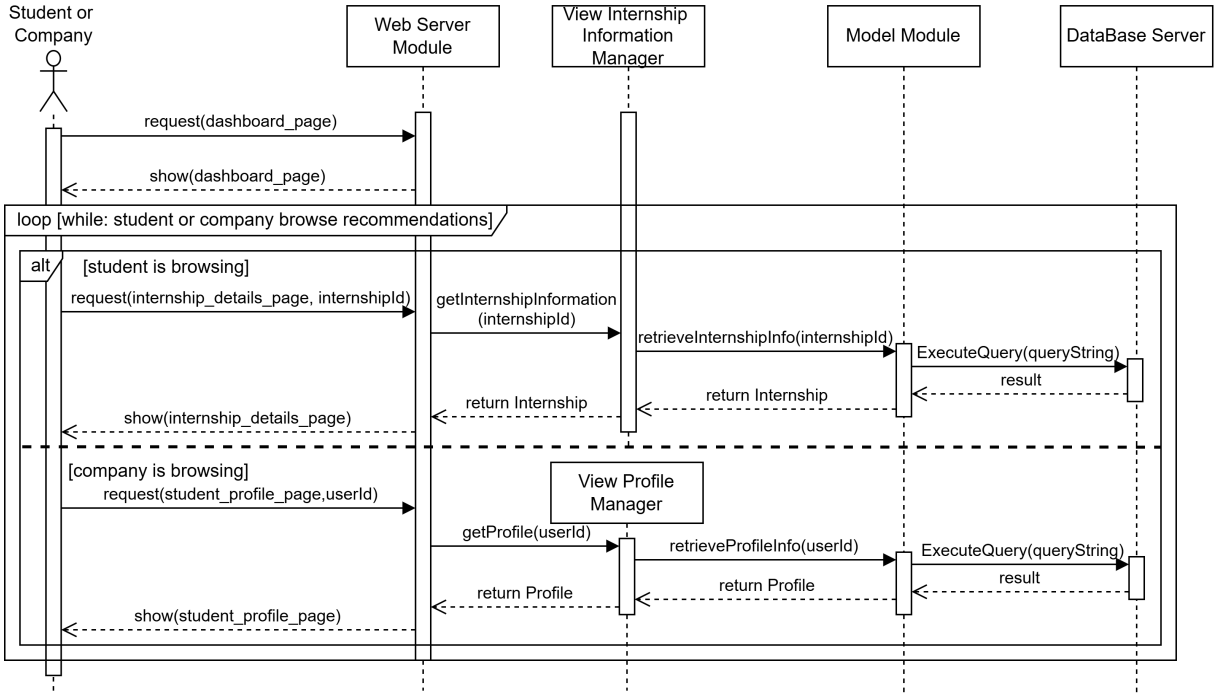


Figure 2.12: Student or Company views recommendation Sequence Diagram

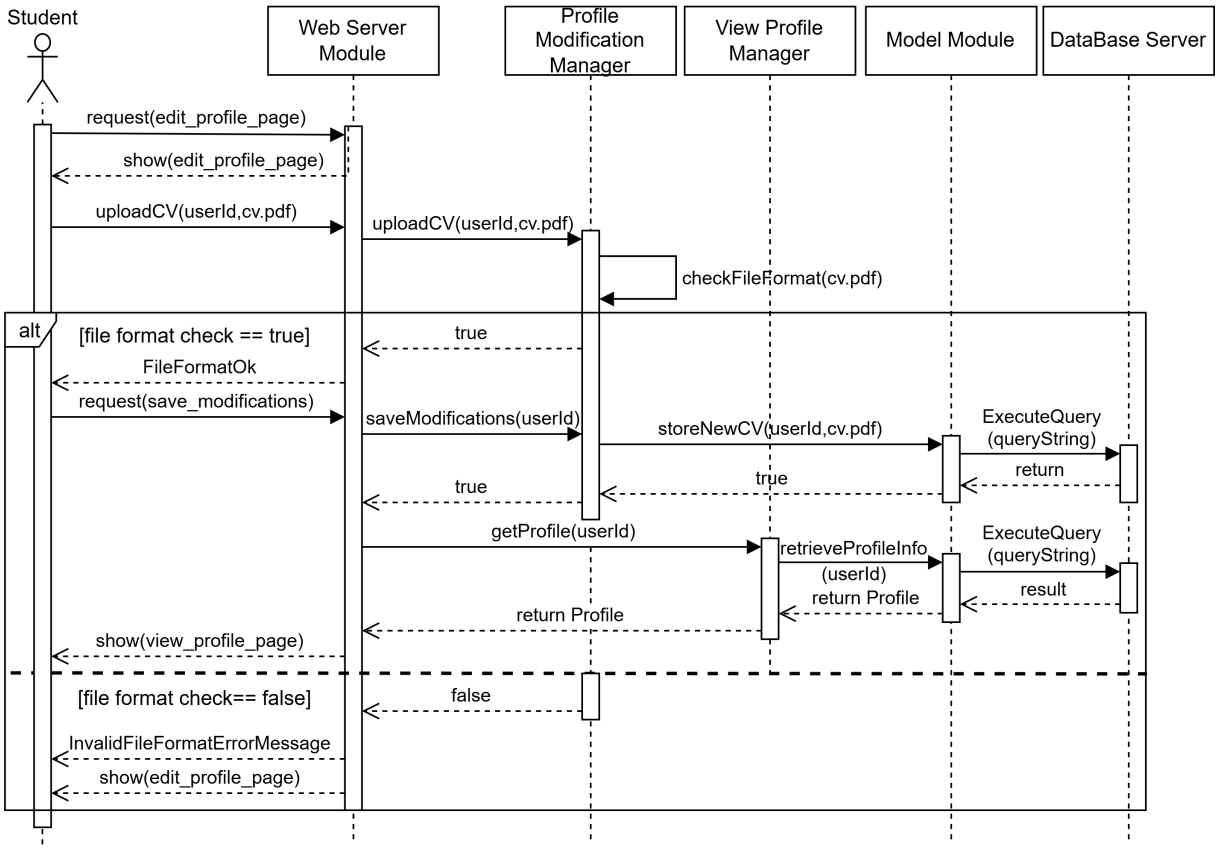


Figure 2.13: Student uploads CV to his profile Sequence Diagram

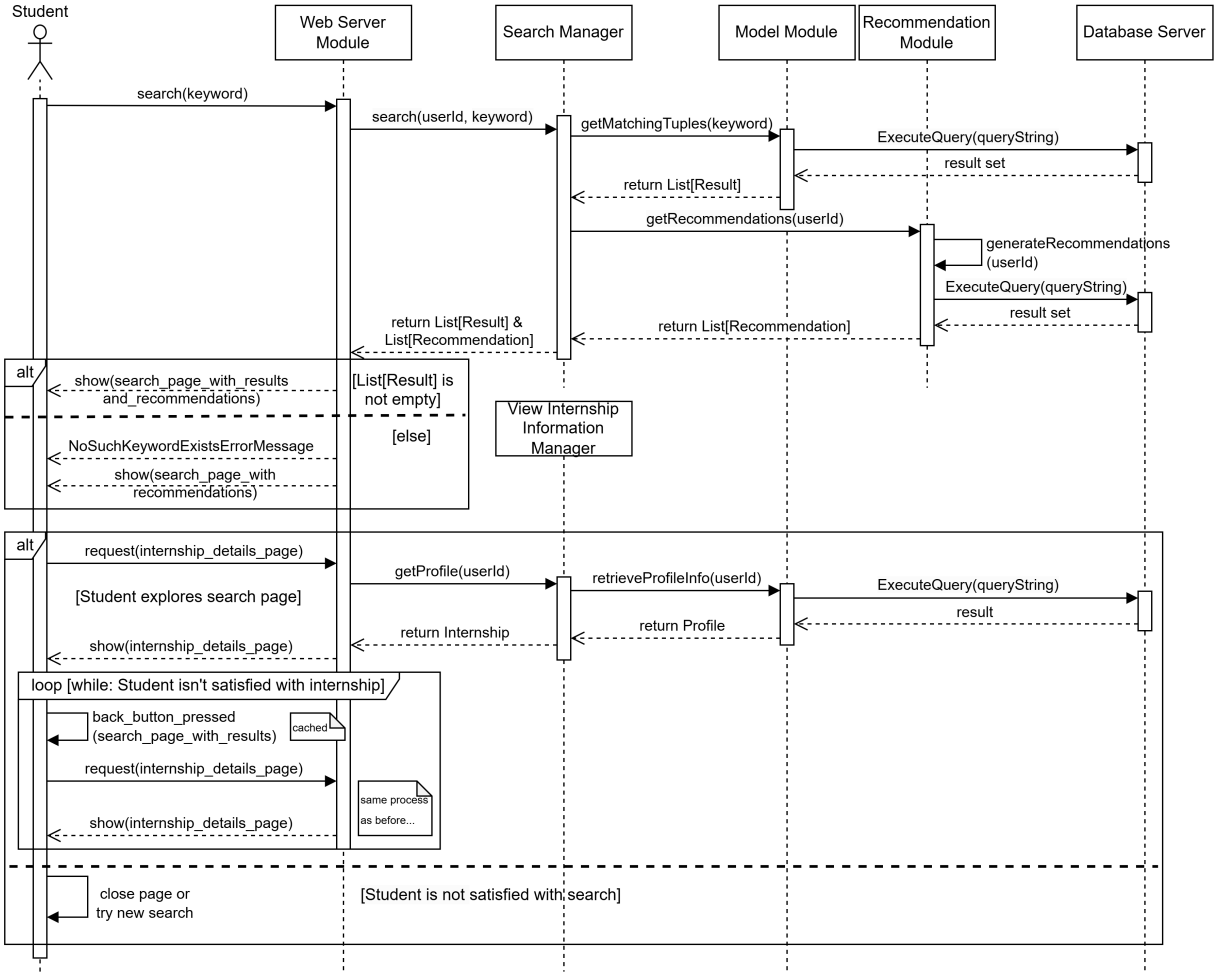


Figure 2.14: Student searches for an internship Sequence Diagram

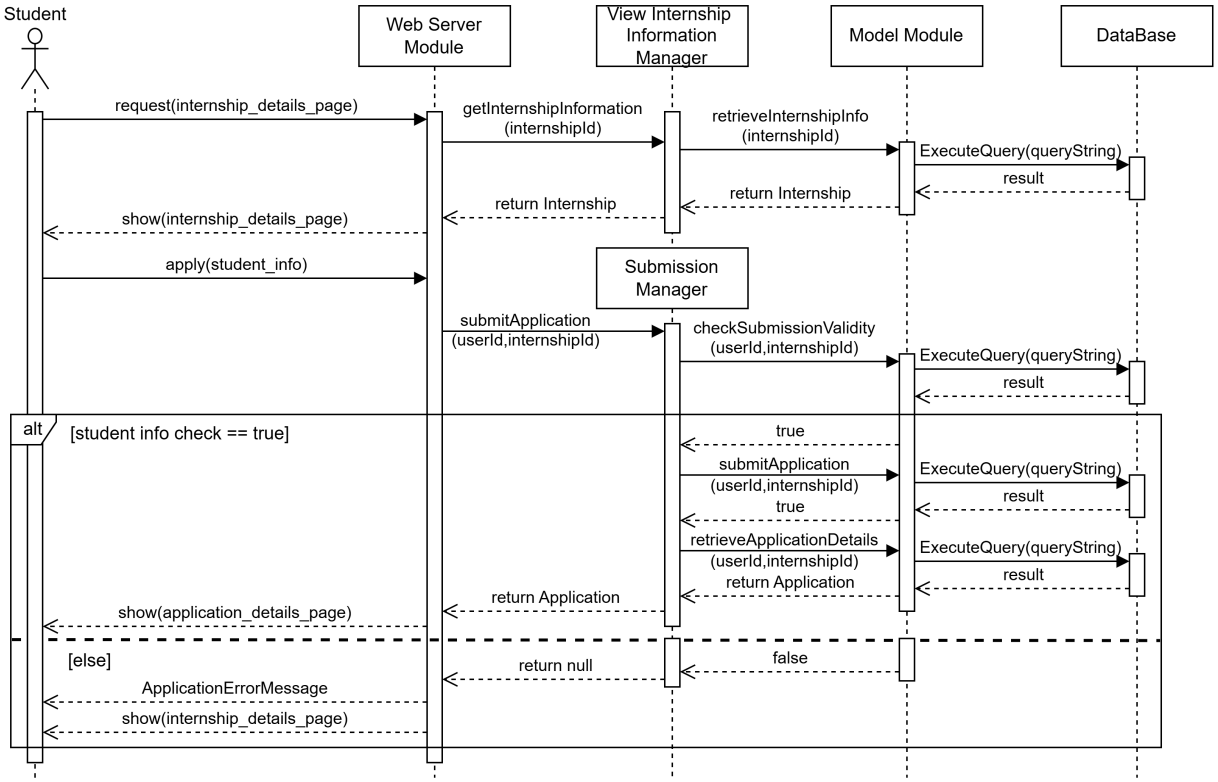


Figure 2.15: Student applies for an internship Sequence Diagram

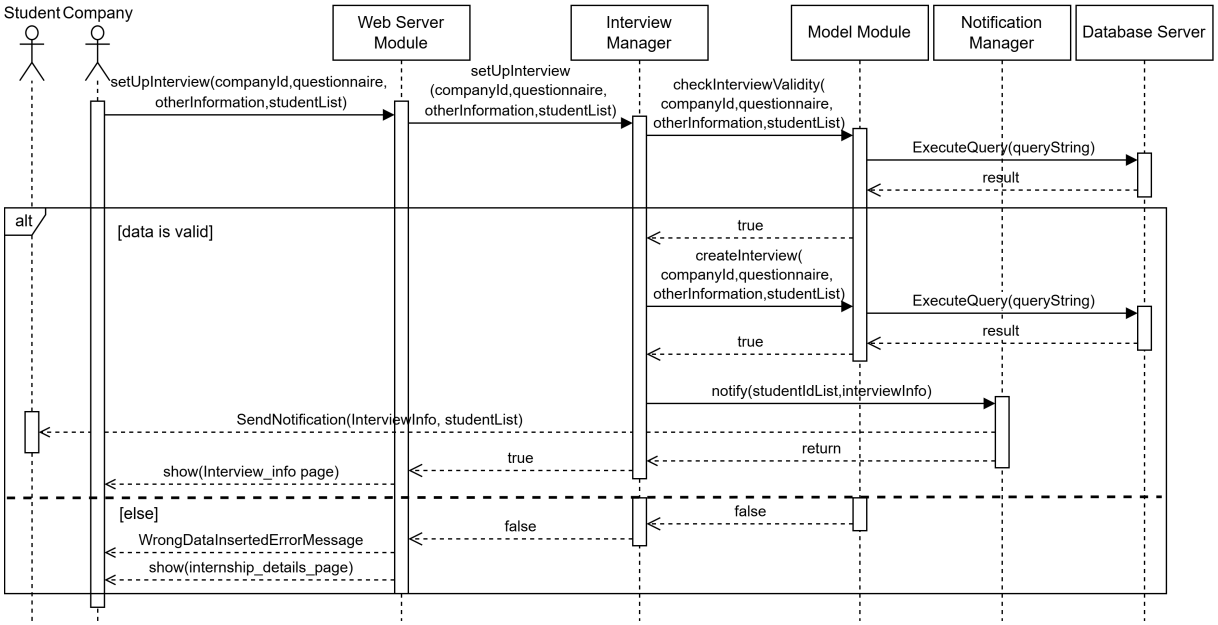


Figure 2.16: Company creates an interview Sequence Diagram

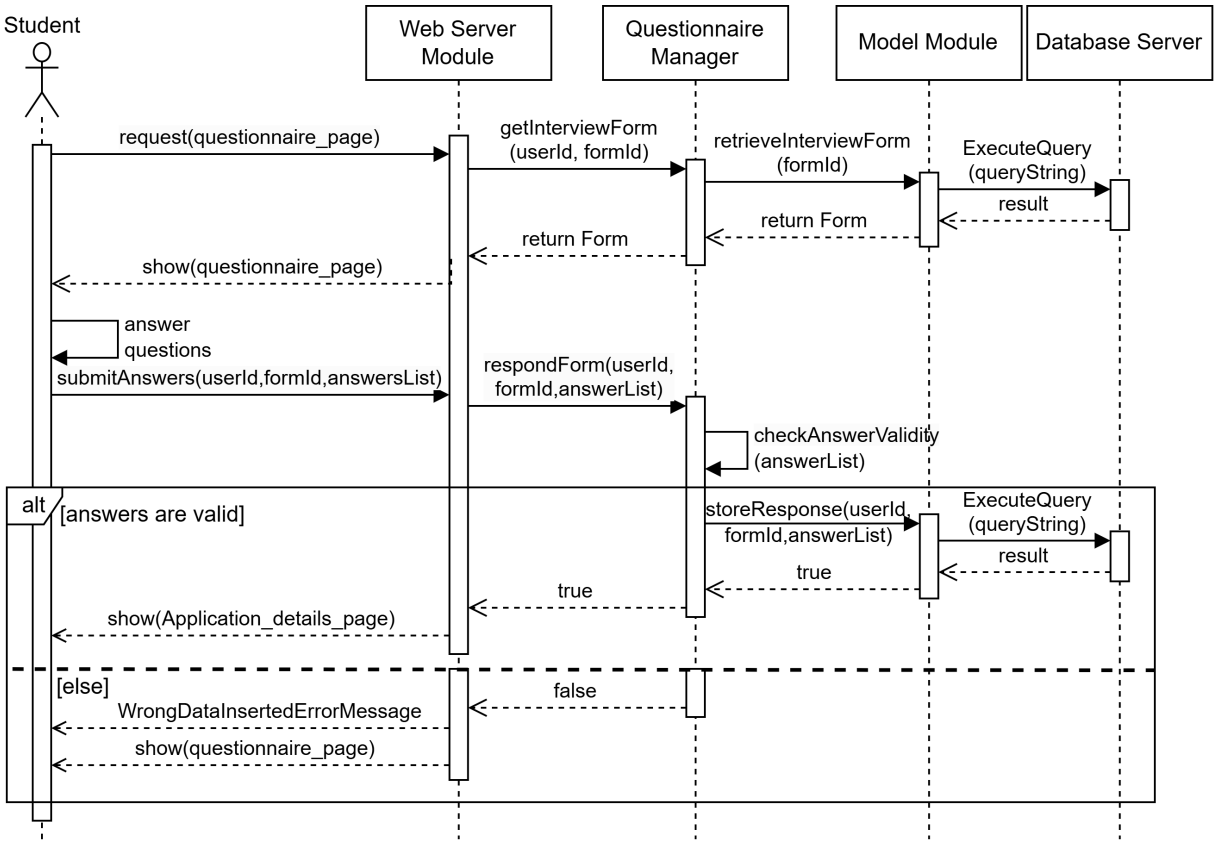


Figure 2.17: Student responds to an interview questionnaire Sequence Diagram

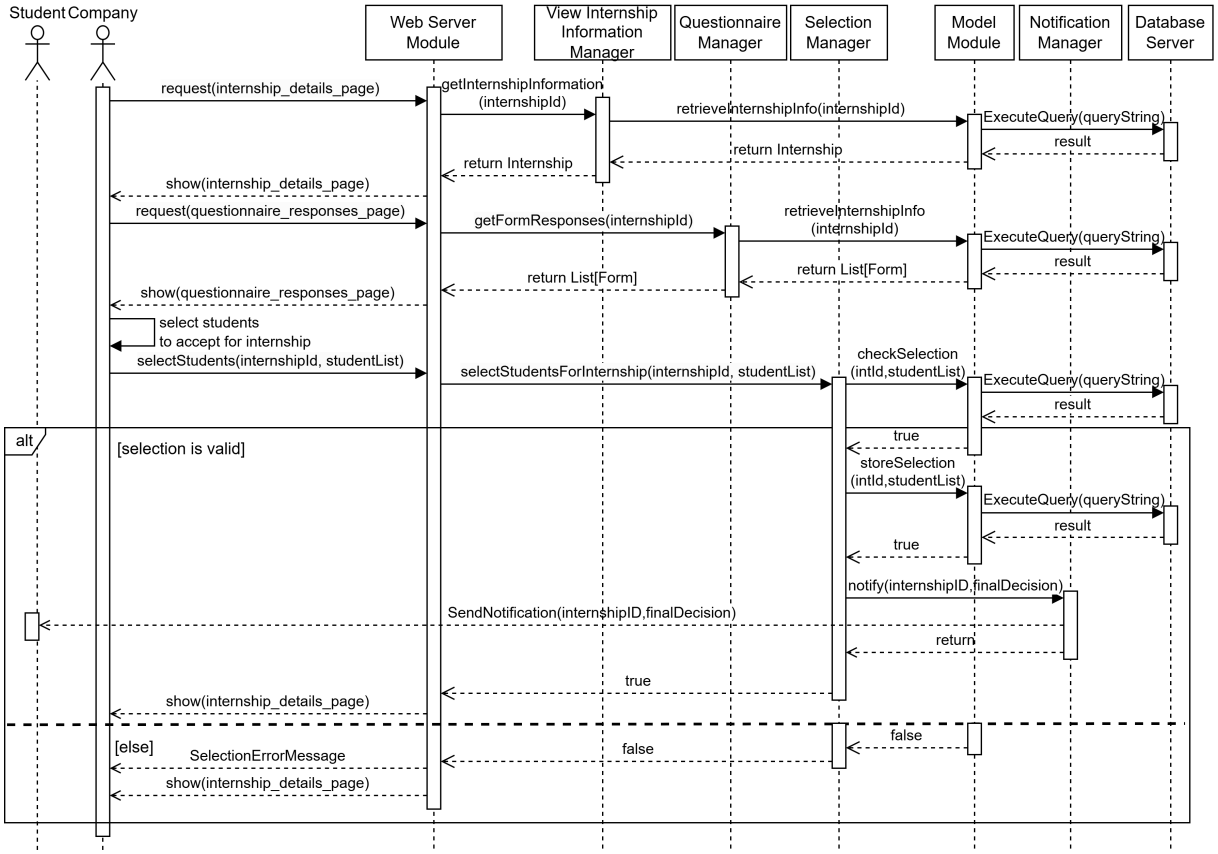


Figure 2.18: Company sends interview results Sequence Diagram

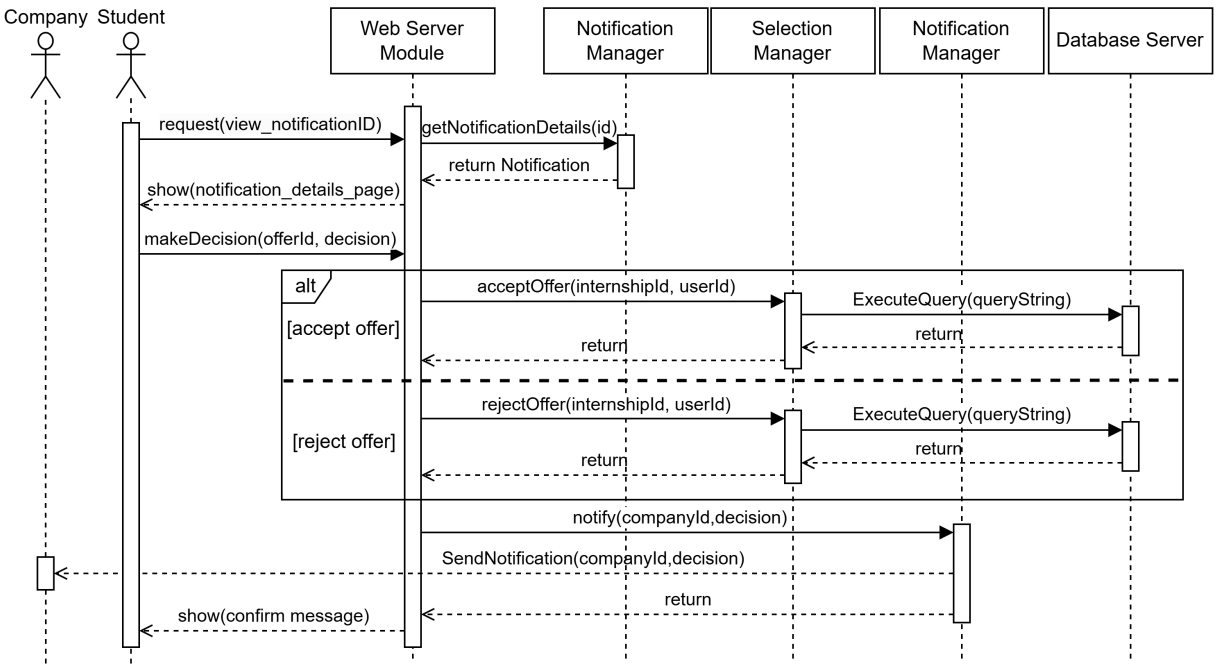


Figure 2.19: Student accepts/rejects an offer Sequence Diagram

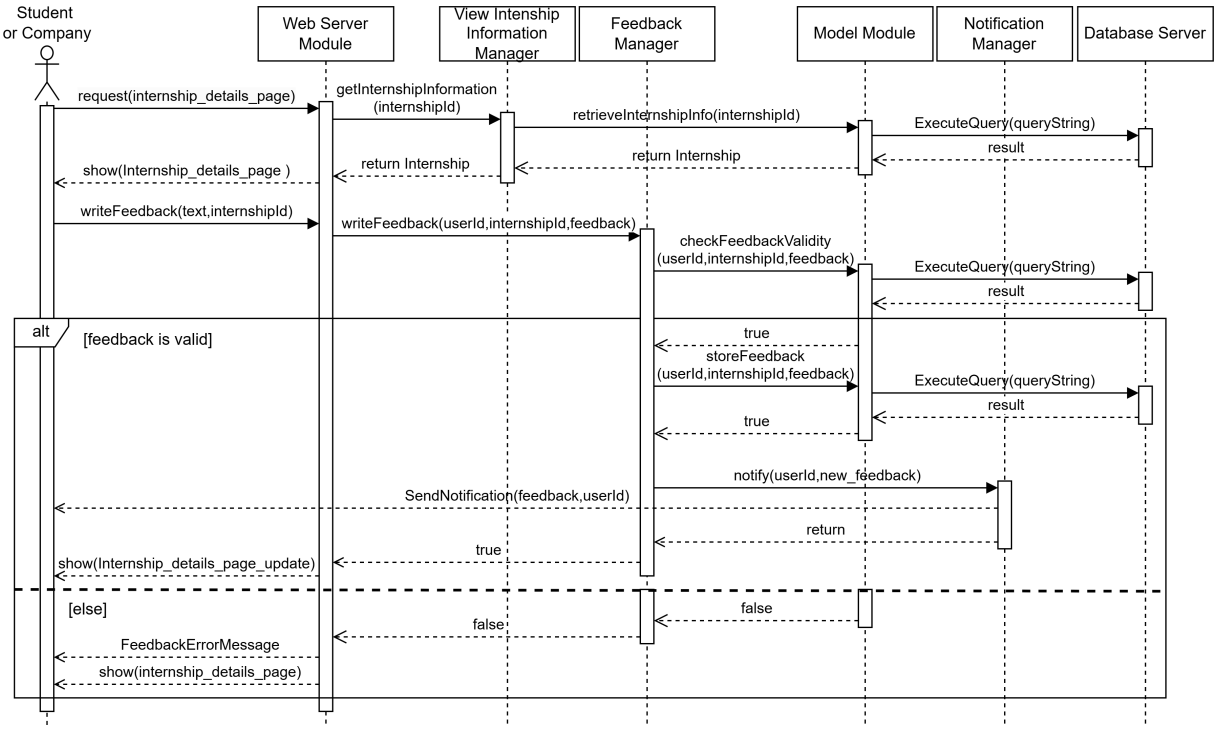


Figure 2.20: Student or Company writes feedback Sequence Diagram

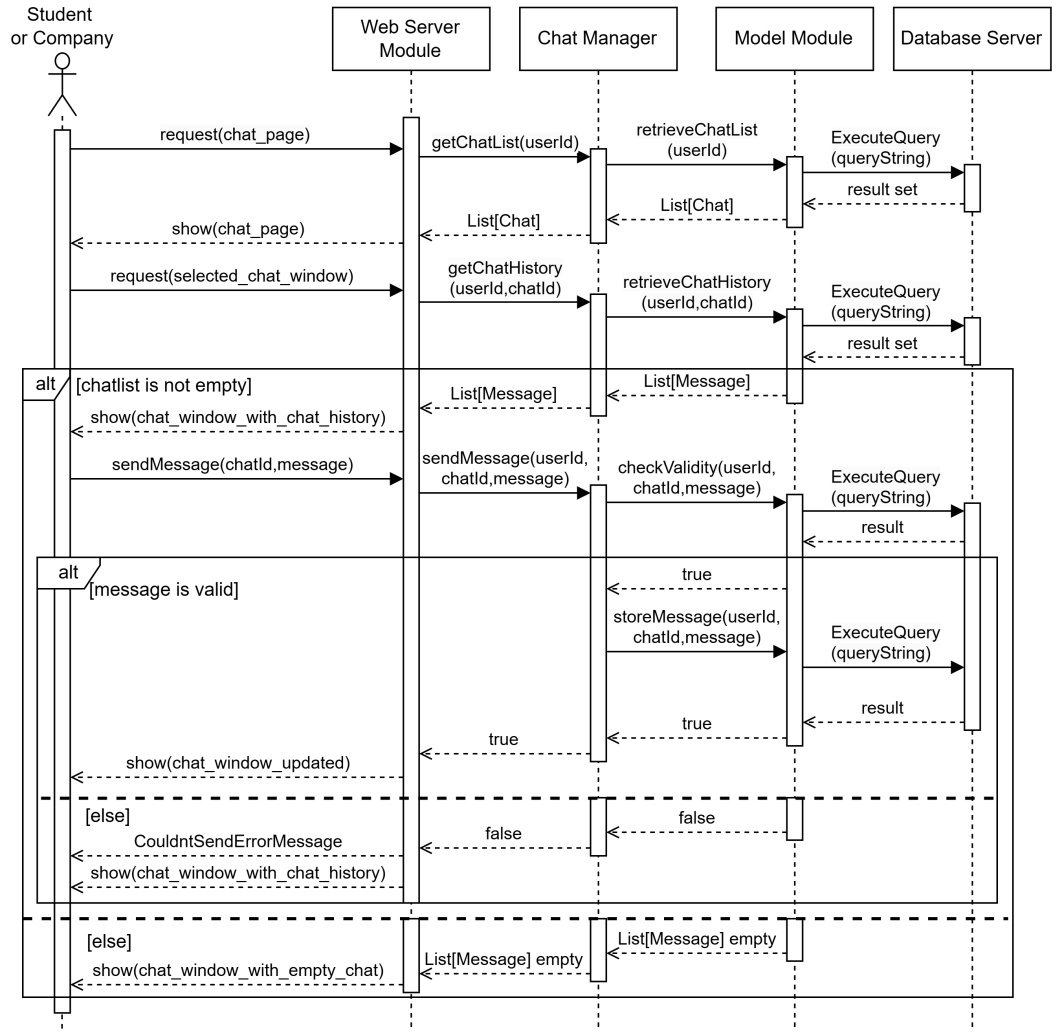


Figure 2.21: Student or Company chats with each other Sequence Diagram

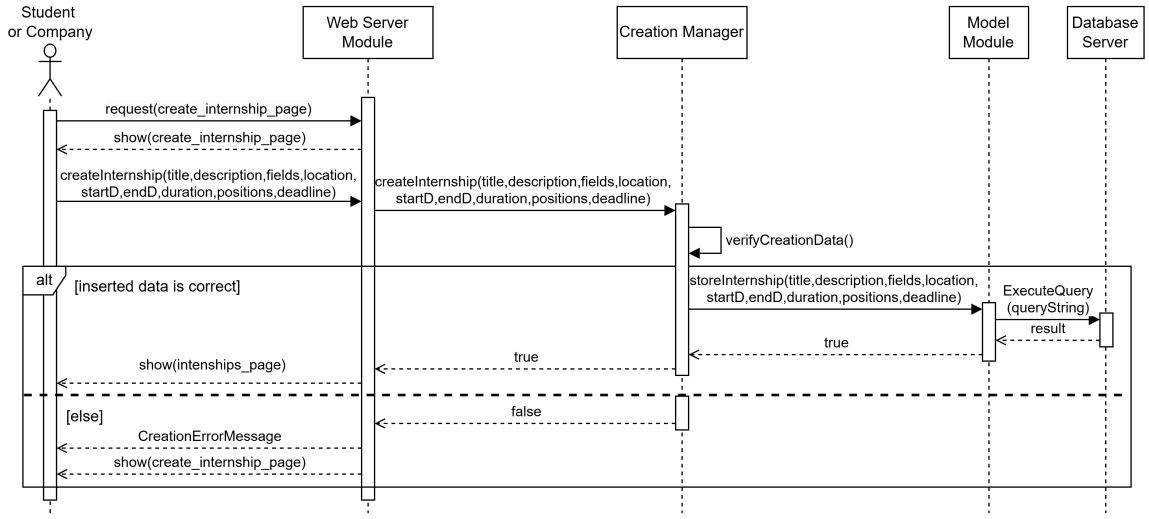


Figure 2.22: Company creates and publishes an internship Sequence Diagram

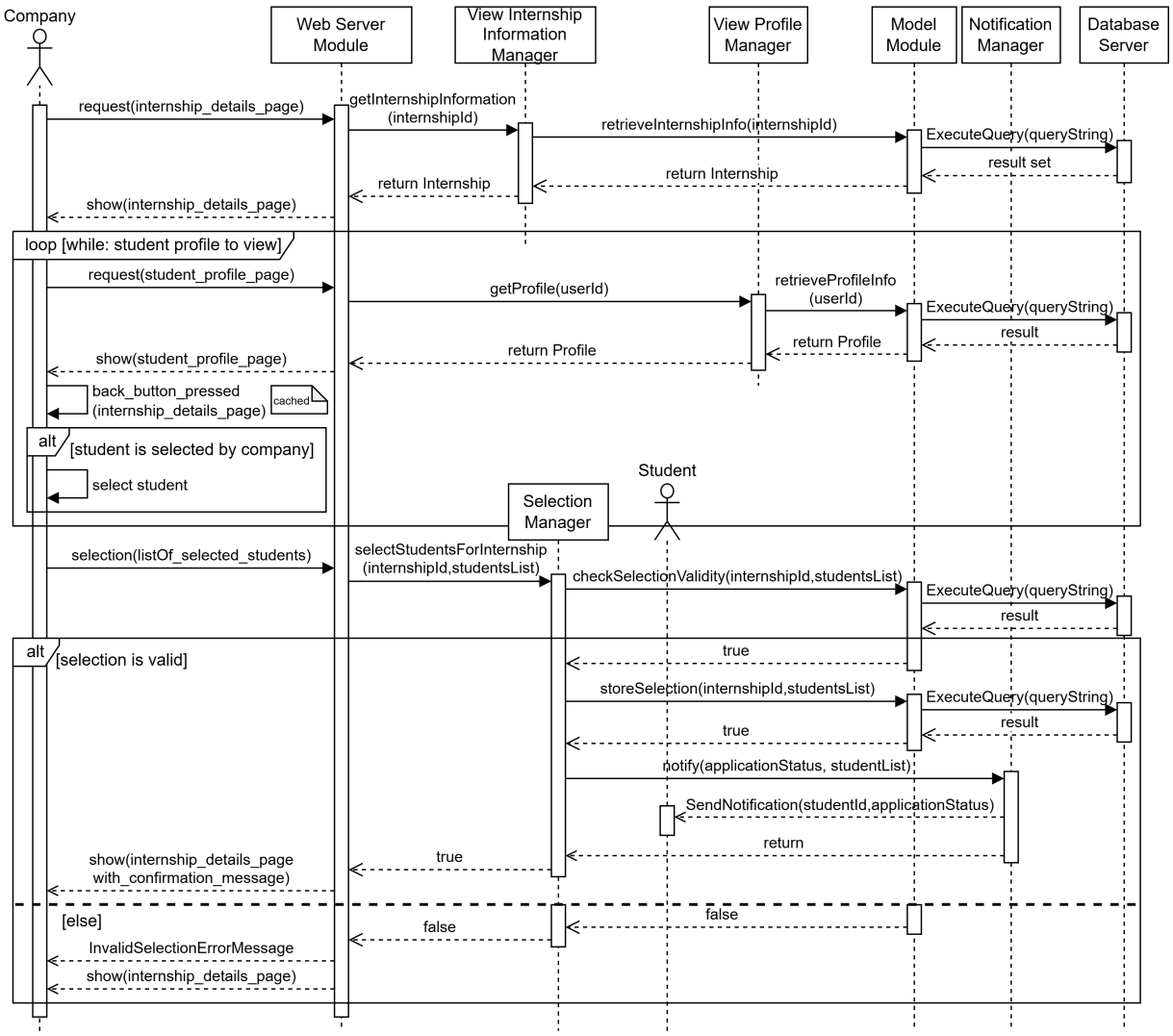


Figure 2.23: Company selects candidates Sequence Diagram

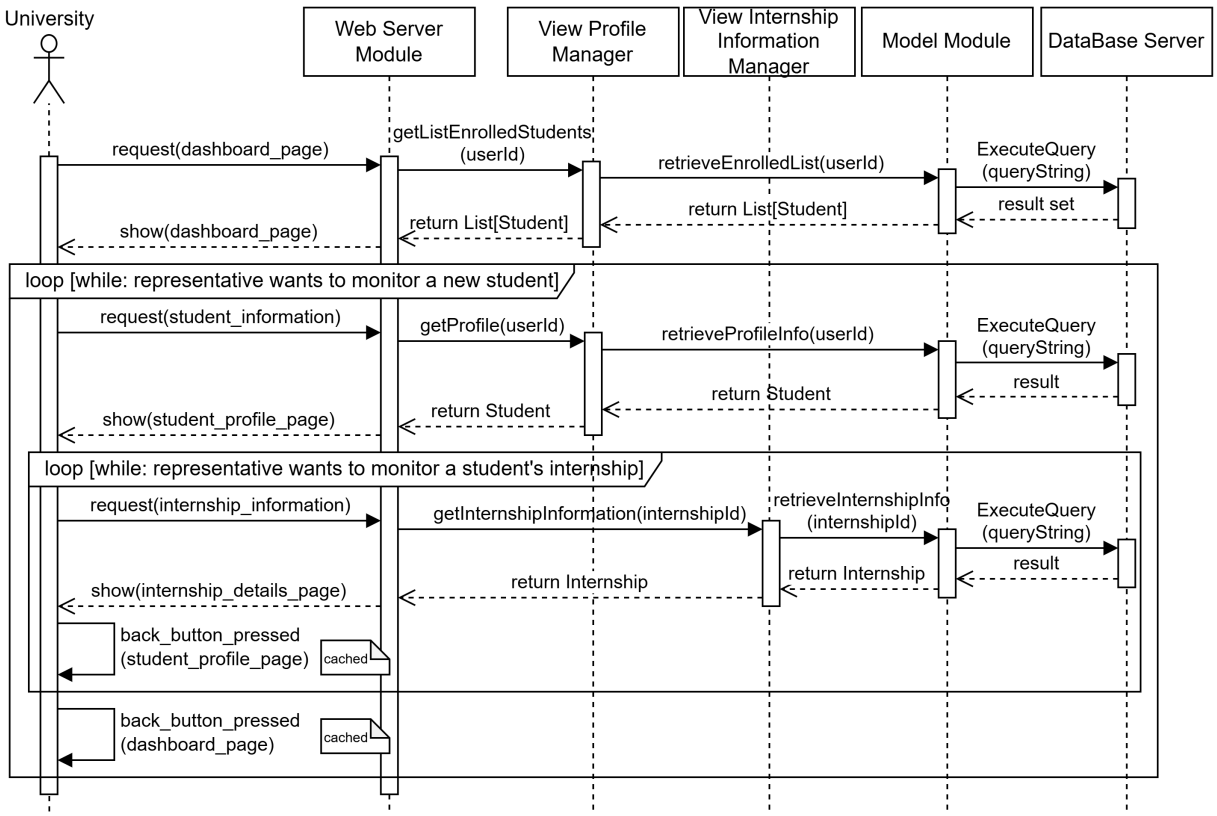


Figure 2.24: University monitors the internship processes of students Sequence Diagram

F. Selected architectural styles and patterns

F.1. Three-Tiered Architecture

As described in Section Overview, the Students&Companies (S&C) platform is build using a multi-tier architecture. This decision was made with the aim of providing a more scalable and flexible system. The system is divided into three main layers: the presentation layer, the application layer, and the data layer. Each layer has its own responsibilities and plays a specific role in the system: the presentation layer serves as the front end, accessible through the GUI, while the application layer and data layer together form the back end of the system, accessible via API-based methods.

- **Presentation Layer:** The presentation layer is implemented as a generic web application accessible through a web browser. It is responsible for managing the presentation logic, including user interaction, the user interface, and rendering information. This layer serves as the front end of the system, the only part that the user can access directly.

- **Application Layer:** The application layer is implemented as a set of RESTful web services. It is responsible for managing the functional logic of the system, controlling communication between the presentation layer and the data layer. This layer allows the system to react to user input and generate appropriate responses accordingly. It includes the Application Server and is also used to interact with third-party services.
- **Data Layer:** The data layer is responsible for managing the data storage and access within the system. All operations that require data manipulation must be performed through interactions with the data layer. The platform uses a Relational Database Management System (RDBMS), making the data accessible through SQL queries.

F.2. RESTful API

The Representational State Transfer (REST) style is designed to be stateless, enabling more efficient and seamless communication between the client and the server. It uses standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. The decision to incorporate RESTful APIs into the architecture provides advantages in terms of performance, modifiability, and simplicity by defining conventions for interacting with resources in resource-oriented manner.

F.3. Model-View-Controller (MVC) Pattern

One of the most recommended design patterns for the Three-Tier Architecture is the Model-View-Controller (MVC) pattern. It separates the application into three components: the model, the view, and the controller, minimizing interdependencies between the components and improving the maintainability, manageability, and scalability of the system. Each component can be developed, tested, and maintained independently.

- **Model:** Contains the state and application logic and is independent of the other components.
- **View:** Represents the visual presentation logic of the Model and is responsible for displaying data to the user.
- **Controller:** Acts as an intermediary between the Model and the View. It receives user input forwarded by the View, then processes operations and updates the Model and the View accordingly.

G. Other design decisions

- **Design patterns related to the behavioral aspects:** Observer Pattern and State Pattern.
- **Design decisions related to the system's requirements:** Some design decisions are already described in the RASD, such as reliability, availability, scalability, security, maintainability, and portability. The following sections revisit availability, scalability, and security to emphasize their importance in the system.

G.1. Observer Pattern

The Observer pattern is particularly useful when multiple objects need to be notified about a change in the state of another object. In the context of the S&C platform, a large number of functionalities require the participation of multiple objects, such as notifying users about the results of an interview or updates on the status of an application.

G.2. State Pattern

The State pattern is recommended to efficiently manage operations across different states and handle transitions between them, as it allows objects to change their behavior when their internal state changes. In the context of the S&C platform, the State pattern can be used to manage the lifecycle of an application for a internship position.

G.3. Availability

The system is designed to be highly available, ensuring that users can access the platform at any time, as described in the RASD, with at least 99.8 percent uptime. To achieve this, critical components should be replicated across multiple servers to provide redundancy and fault tolerance in case of failure. Load balancing is correctly configured to distribute incoming traffic, preventing overload on any single server. Continuous monitoring of the system's performance allows for the detection and resolution of any issues that may arise in real-time.

G.4. Scalability

The platform is designed to handle increased user loads in the future by scaling individual layers independently, thanks to the architectural styles and patterns mentioned earlier. As described in the RASD, the system can be scaled horizontally by adding more servers

or vertically by increasing the resources of existing servers, without affecting performance. This scalability is essential, especially as the number of users grows, leading to a higher volume of application requests over time.

G.5. Security

The system is designed to ensure the privacy and security of user data both during transmission over the network and while stored in the database. This includes the use of authentication and authorization mechanisms to ensure that only authorized users can access the system, reducing the risk of unauthorized access. Additionally, a firewall and Intrusion Detection System (IDS) are set up in the network to protect the system from external threats and attacks. Protocols like HTTPS are used to encrypt communication between the client and server, and other encryption algorithms are employed to protect sensitive data stored in the database, such as user passwords.

3 | User Interface Design

A. User Interface Design

In this section, the user interface design will be presented using mockups within the short description. The design focuses on optimizing the user experience and ensuring that the user can easily navigate on the website to perform the desired actions. There will be separate in subsection to describe more clearly the main pages needed to satisfy the user requirements.

A.1. Welcome Page

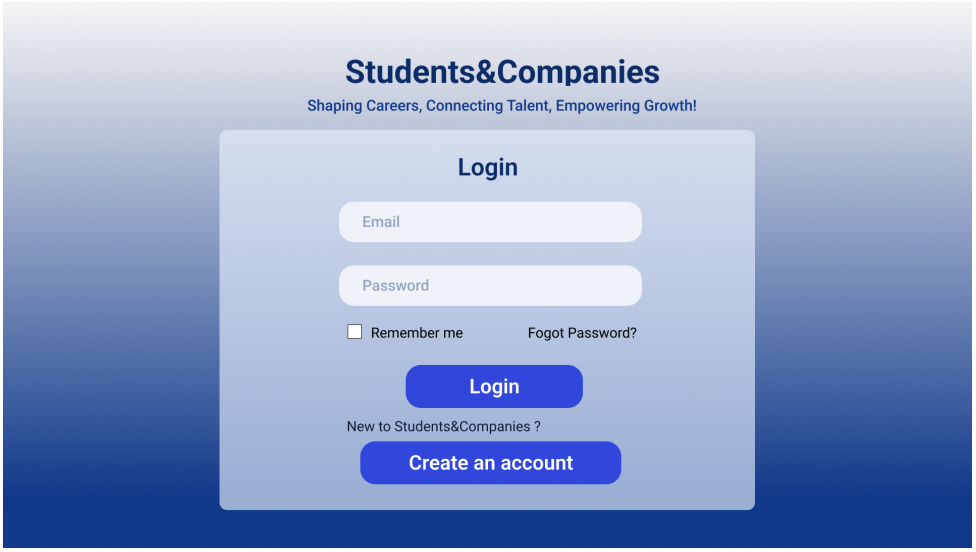


Figure 3.1: Welcome Page

A.2. Register Page

If the User is not registered and wants to create an account, they will be asked to choose the type of account they want to create. Clicking on the type listed will redirect to the corresponding Register Page where the user can fill in the required information.

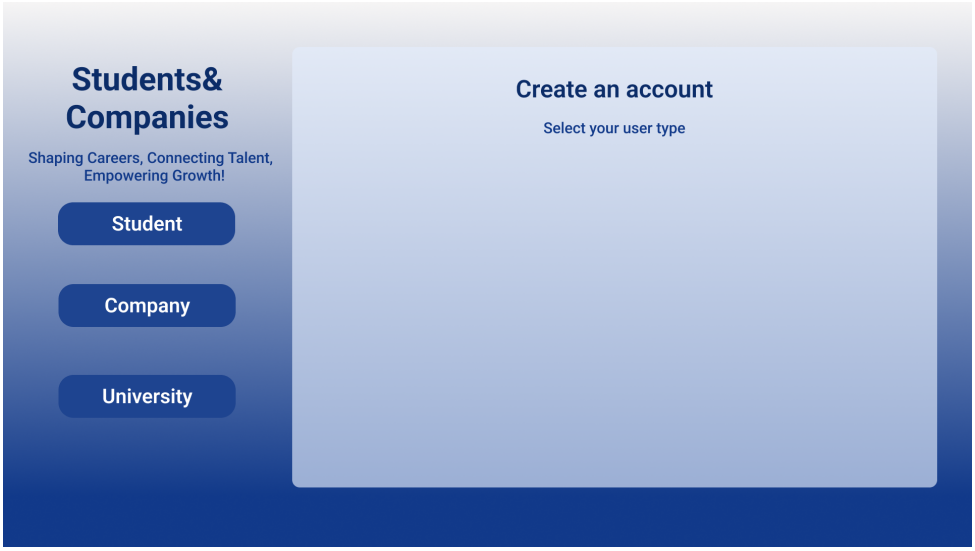


Figure 3.2: Register Page

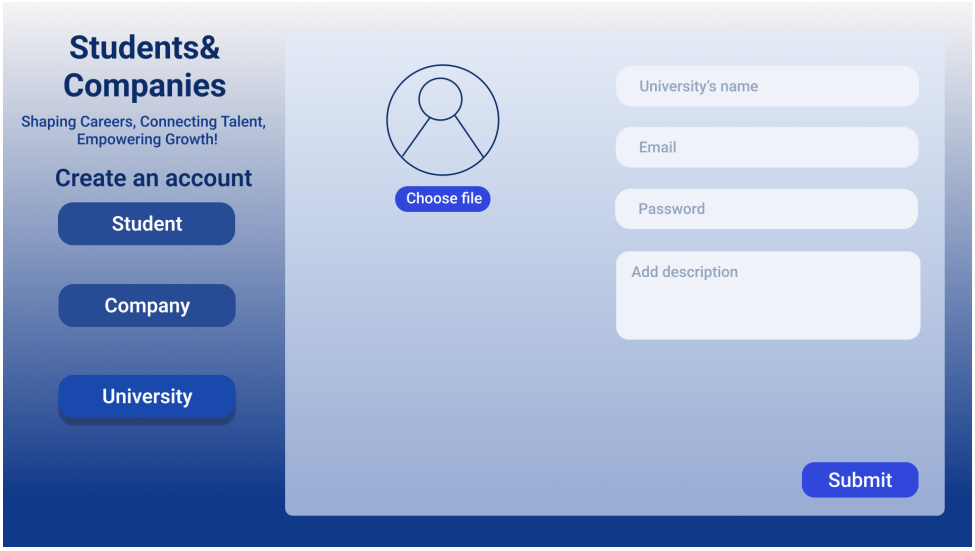


Figure 3.3: University create account

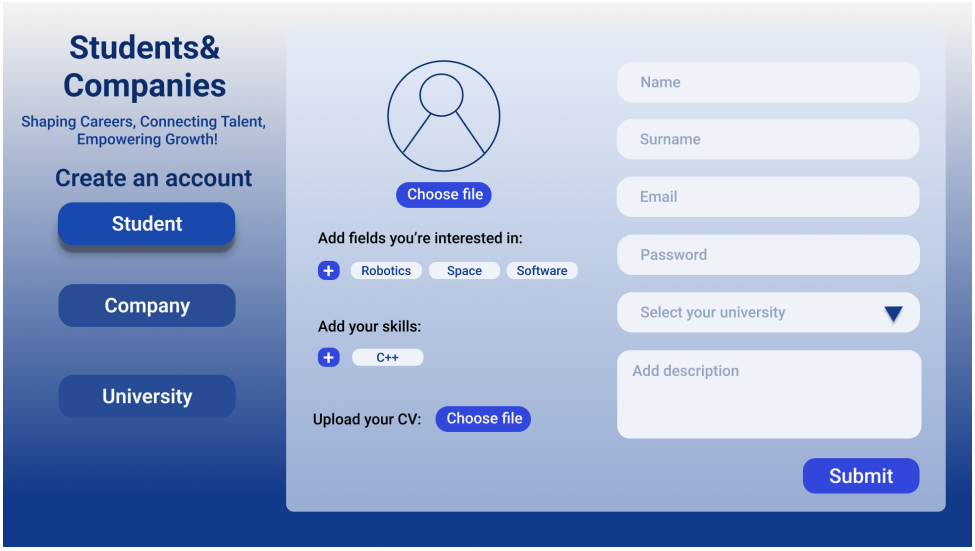


Figure 3.4: Student create account

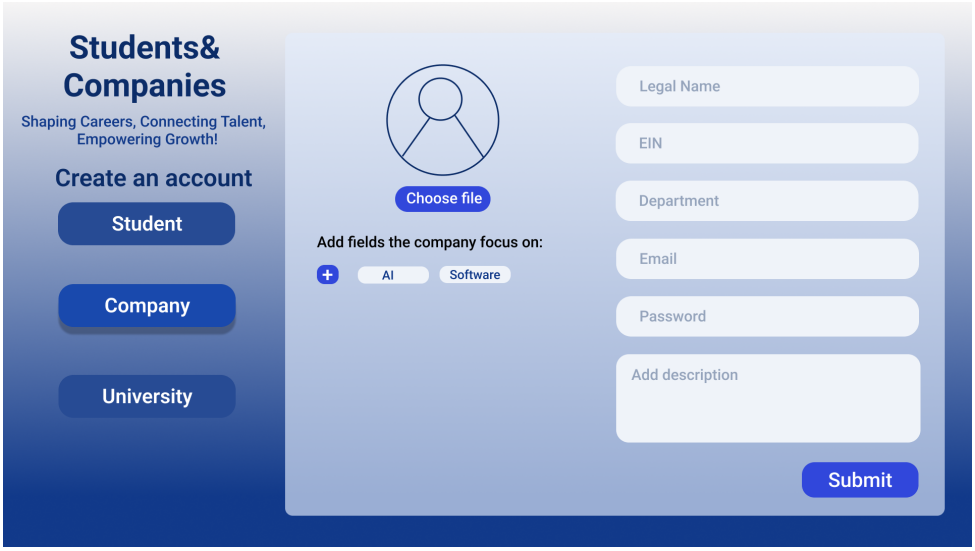


Figure 3.5: Company create account

A.3. Header bar

The header bar is displayed on every page of the platform and contains the logo of the platform, the side menu button, the notification bell, and the user profile button. In particular, there is also the chat icon that allows the Student and Company to access available chat lists.

A.4. Student's view

Once access to the platform, to optimizing the user experience, the student will be able to use the side menu to navigate to the desired page: *Search, Dashboard, My applications and My internships.*

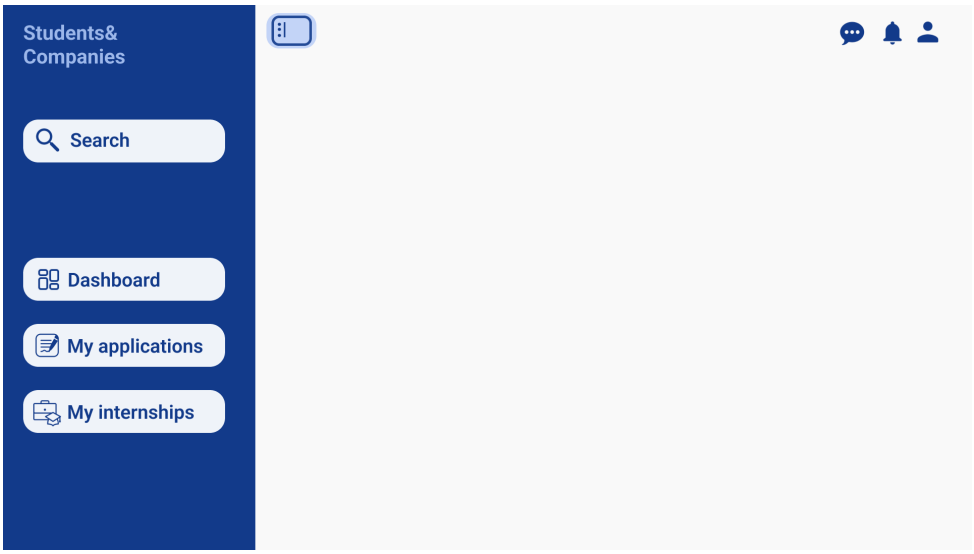


Figure 3.6: Student's Side Menu

Dashboard page

By default, students are directed to the Dashboard page upon logging in.
The Dashboard provides an overview that allows students to quickly assess the current status of their applications and the performance of internships they are participating in or have completed. Clicking on a specific application or internship within the overview list redirects students to a detailed page for that application or internship.
Next to the historical internship section, there is a window displaying the latest comments on the current internship with quick access to the Feedback and Complaint pages for the ongoing internship, click on *Open*.
Additionally, students can use the search bar to find internships or user profiles by entering relevant keywords. Below the search bar, there is a *Suggested Job Searches* section, where students can click on a keyword to quickly search for internships that match their interests. Below this section, a list of recommended internship announcements tailored to the student is displayed. Each announcement includes a brief description, and students can take one of three actions:

- Click the *Apply Now* button to apply for the internship.

- Click the block to view more details about the internship announcement.
- Click the X to remove the internship announcement from the display.

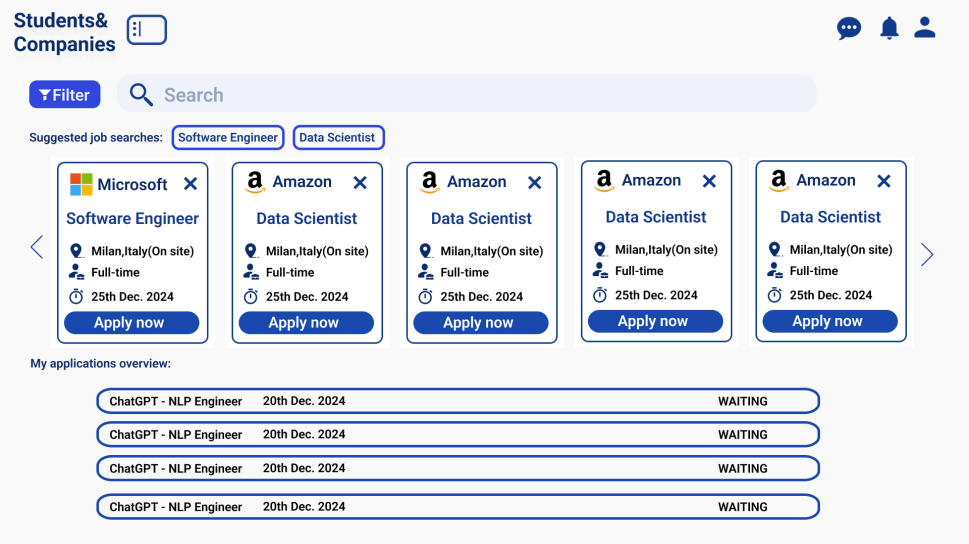


Figure 3.7: Student's Dashboard 1

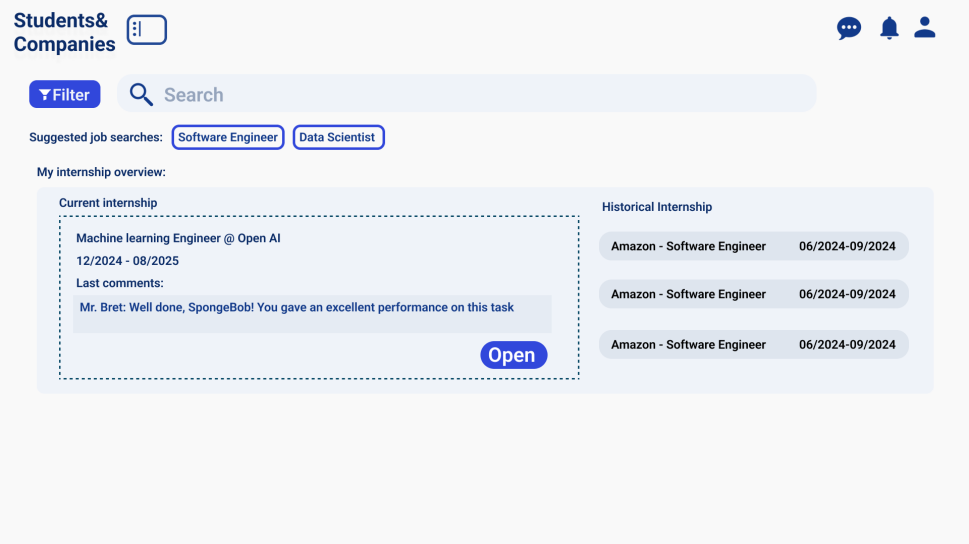


Figure 3.8: Student's Dashboard 2

My application page

Here in figure3.9, the student can view the status of their applications and take action based on the status of the application:

- Click on the *View details* button to view the page with details of the internship announcement, figure3.15.

- Click on the *Invited to interview* button to view the details of the interview and questionnaire requested by the company to fill in, figure3.10.
- Click on the *Accept or Reject Offer* button to view the details offer letter and to take the decision to accept or reject the offer, figure3.13.

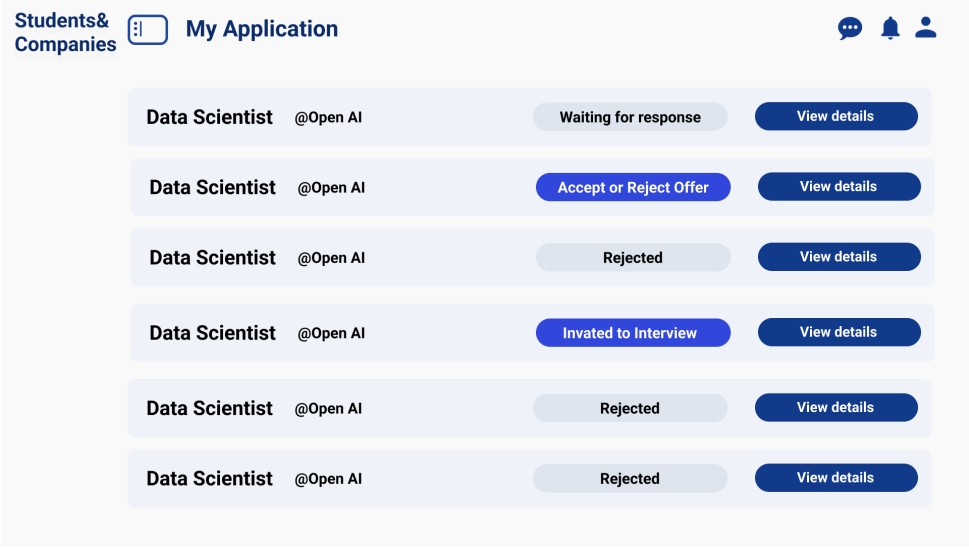


Figure 3.9: My application

Interview details page

Show the detail information related to the interview and the pre-interview questionnaire asked by the company to candidate to fill in.
Once completed all questions with the provided answers, the student can click on the *Submit* button to send the answers to the company and save the answers in the platform for future reference.

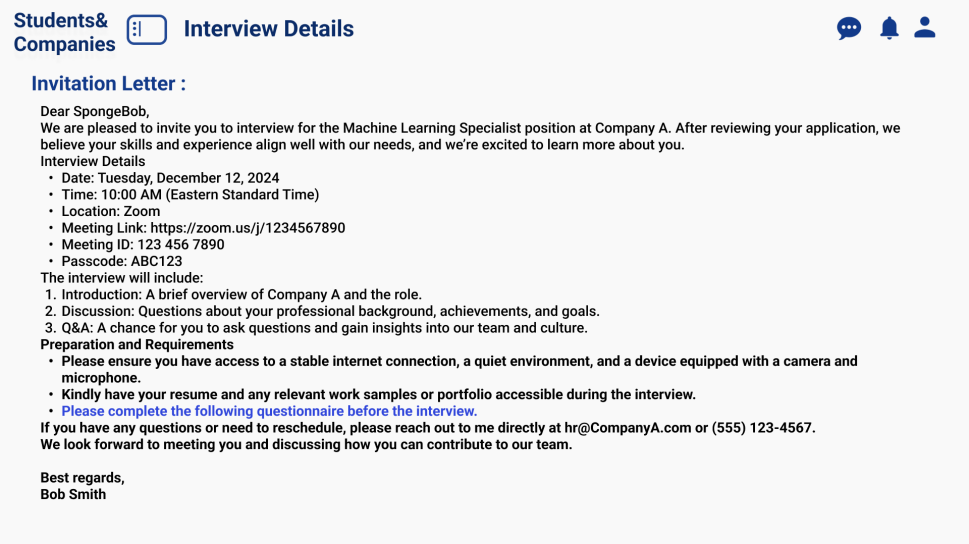


Figure 3.10: Interview details 1

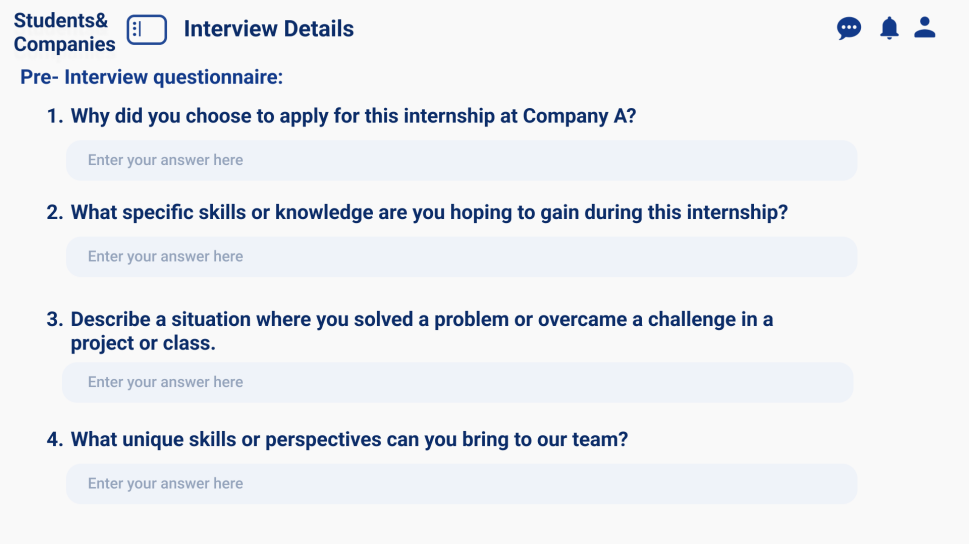


Figure 3.11: Interview details 2

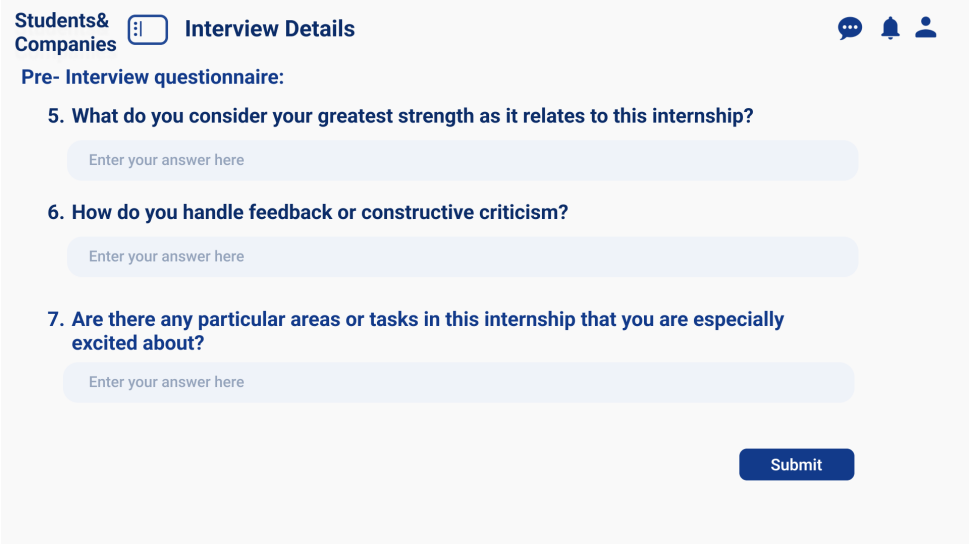


Figure 3.12: Interview details 3

Offer page

Provide the offer letter sent by the company to the student. The student can take the decision to accept or reject the offer by clicking on the corresponding button. Then the student decision will be sent to the company and recorded in the database, the application status will be updated accordingly.

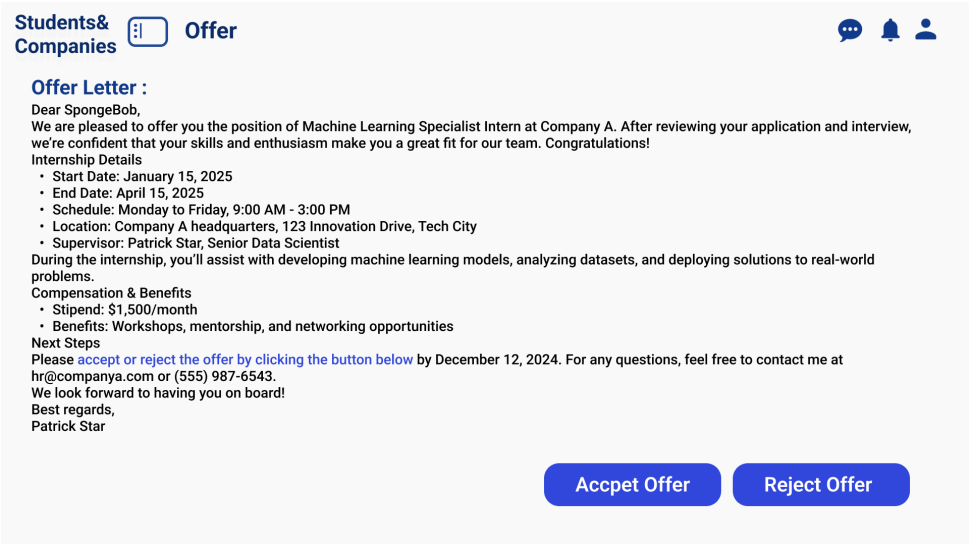


Figure 3.13: Accept or Reject Offer

My internship page

Here in figure3.14, the student can view the list of internships they are participating in or have completed.
For the current internship, the student can access the feedback and complaint page by clicking on the *Add comments* button.
For the historical internship, the student can view the feedback and complaints recorded by the company and the student during the internship clicking on the *View details* button.

Figure 3.14: My internship

Internship announcement page

Represents the internship announcement details by the student's view side.

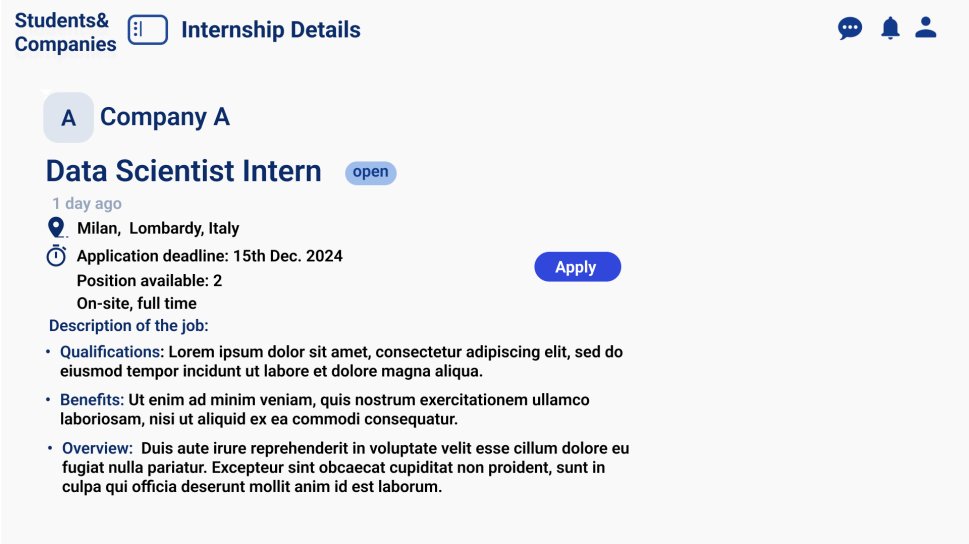


Figure 3.15: Internship announcement details

Student profile page

It shows a student's profile from the prospective of other's user once looking for his/her profile. The personal information such as profile photo, name, email and major are displayed in this page as well as the list of internships the student has participated.

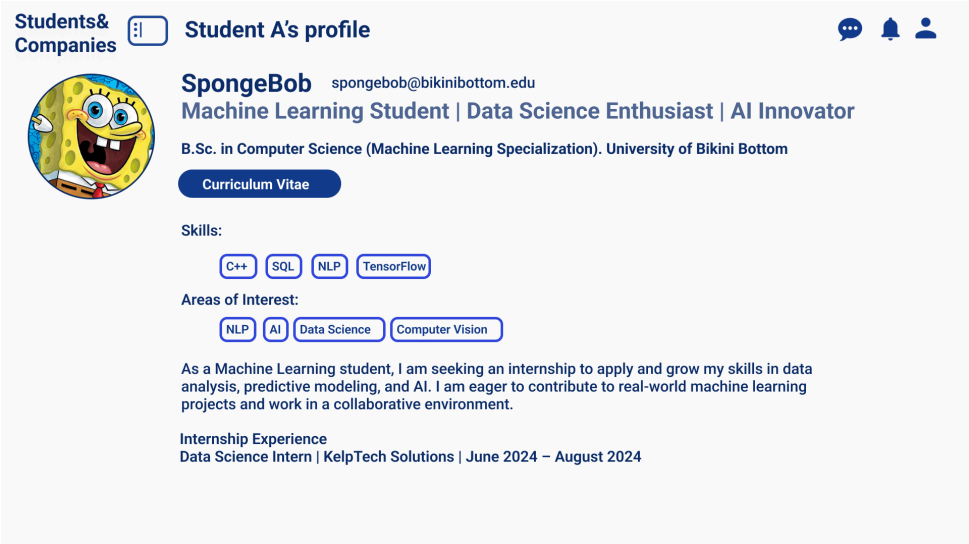


Figure 3.16: Student's profile from other's view

A.5. Company's view

As the student, the company can use the side menu to navigate to the desired page: *Search*,*Dashboard*,*Publish Internship*,*Internship Management*.

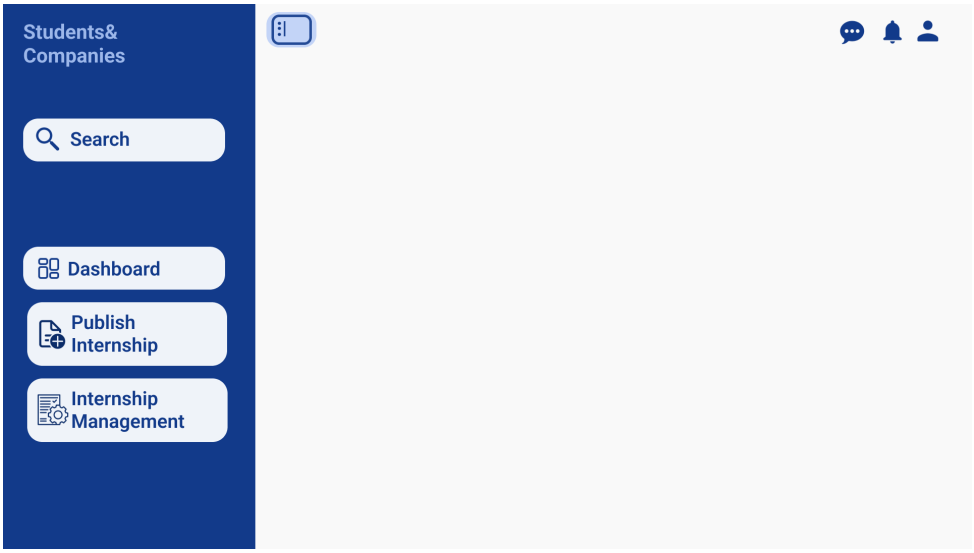


Figure 3.17: Company's Side Menu

Dashboard page

Similar to the student's view, the company is directed to the Dashboard page upon logging in. At the top of the page, the company can use the search bar as well as the search bar presented in the student's view.
The Dashboard provides an overview that allows companies to quickly assess the announcements they have published and allows them to access directly to the publish internship page by clicking on the *Publish new internship* button.
Below that, there is a section In progress internship overview that allows companies to monitor the status of the internships in progress and clicking on the open button will redirect the company to the feedback and complaint page for the that internship.

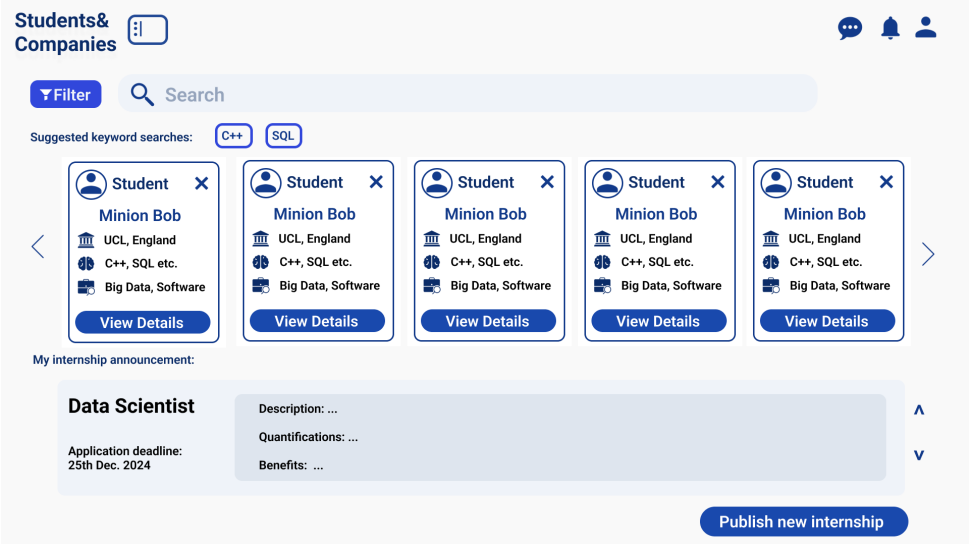


Figure 3.18: Company's Dashboard 1

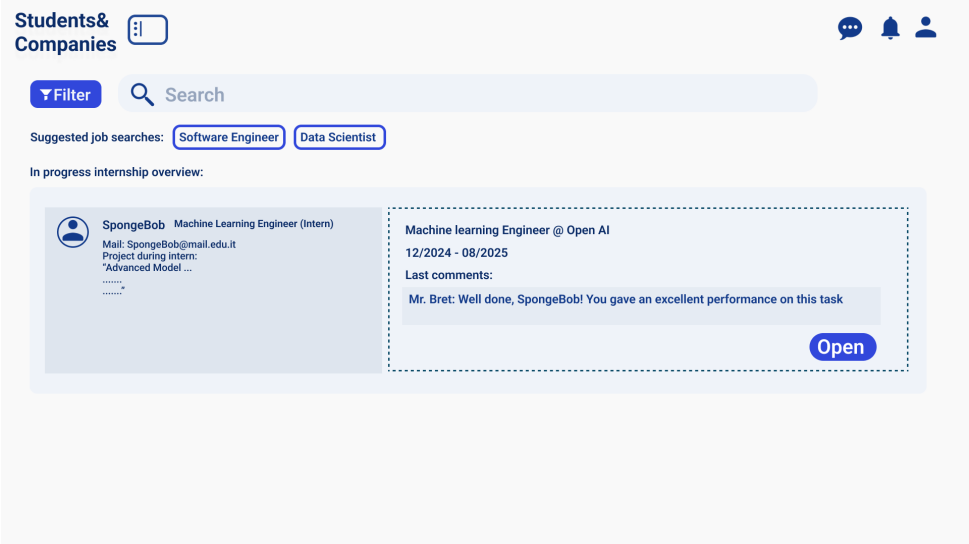


Figure 3.19: Company's Dashboard 2

Publish Internship page

In this page, the company can fill in the required information to publish a new internship announcement on the platform.

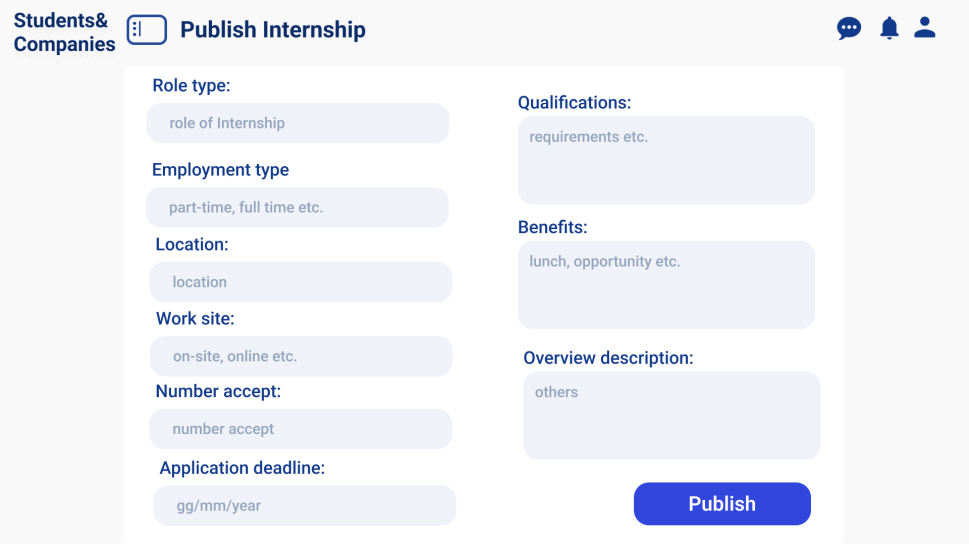


Figure 3.20: Publish Internship

Internship Management page

Here in figure3.21, the company can view the list of internships they have published and take action based on the status of the internship:

- In publishing, means the internship announcement is still open for students to apply, the company can click on the *View details* button to view the page with details of the internship announcement, figure3.23.
- In selection, means the deadline is reached and the internship enter in selection phase, the company can click on the *Select candidates* button to view the list of candidates who have applied for the internship and take action to select the candidates to process the interview, figure3.24.
- In progress, means that internship is in progress, the company can click on the *Add comments* button to access the feedback and complaints page for the internship, figure3.27.
- Completed, means that internship is finished, the company can click on the *View details* button to view the announcement details and feedback and complaints recorded by the company and the student during the internship.

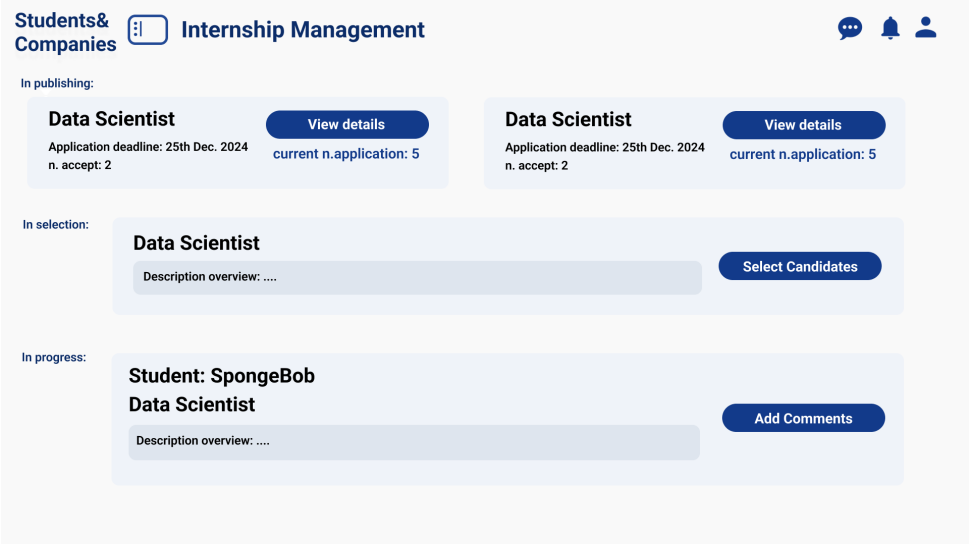


Figure 3.21: Internship Management 1

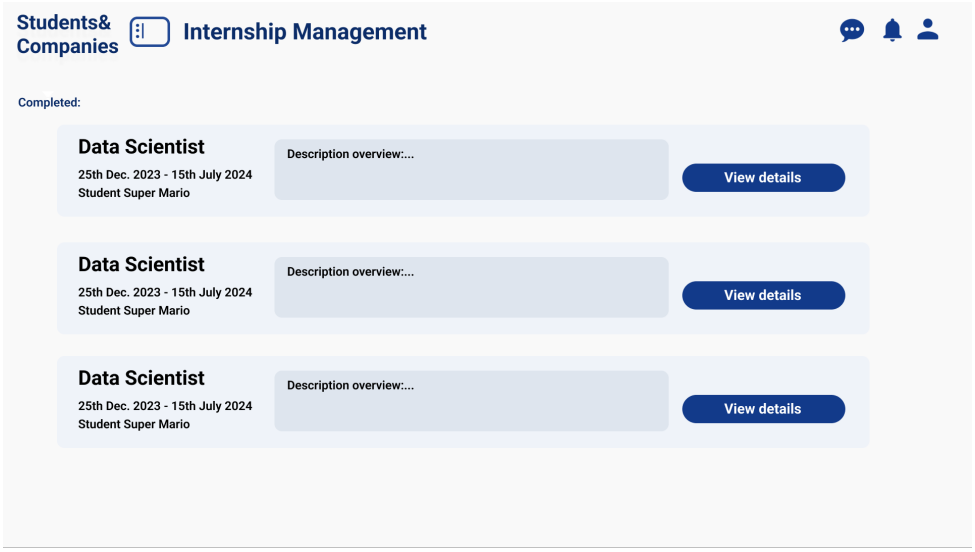


Figure 3.22: Internship Management 2

Internship details page

The internship details page in selection phase, the company can view the list of candidates who have applied for the internship and select the candidates to process the interview by clicking on the button *process interview*.

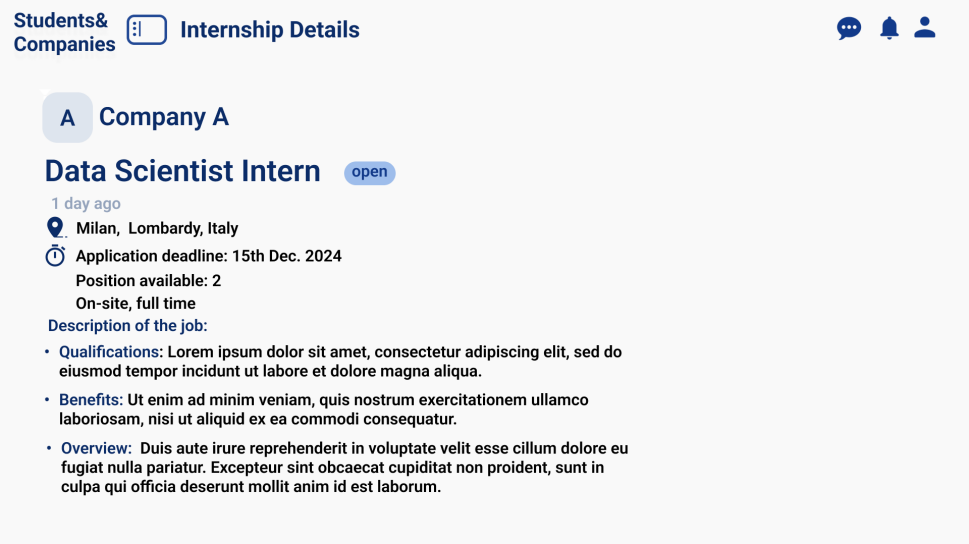


Figure 3.23: Internship details in publishing phase

Figure 3.24: Select candidates

Interview set up page

In this page, the company can write the invitation letter with the necessary information for the interview and prepare the questionnaire to ask the candidate to fill in before the interview. Clicking on the *Submit* button will send the invitation letter and the questionnaire to the student selected for the interview.

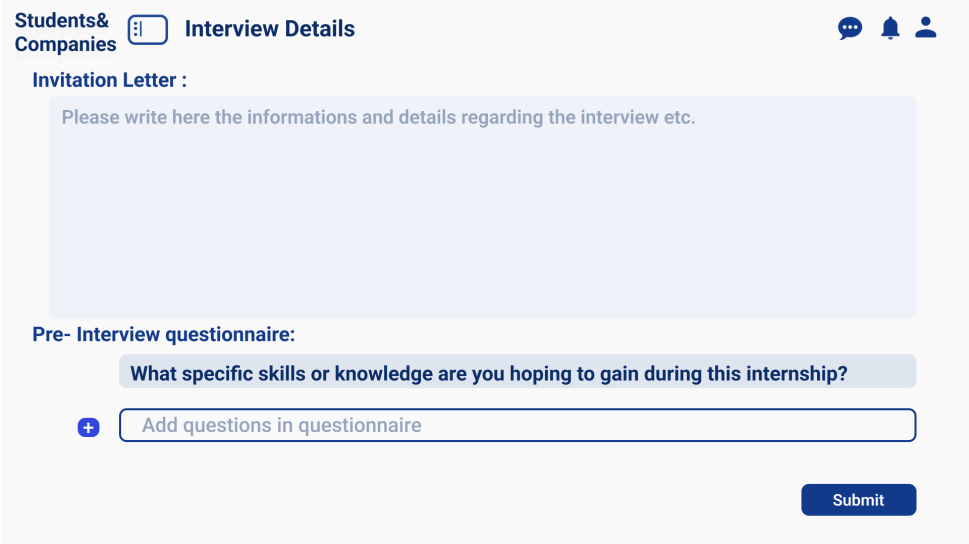


Figure 3.25: Set up interview

Company profile page

Figure3.26 shows a company's profile from the prospective of other's user once looking for its profile. Including the company brief description, contact email, specific fields of the company that the company is focusing on and the list of internships the company has published.

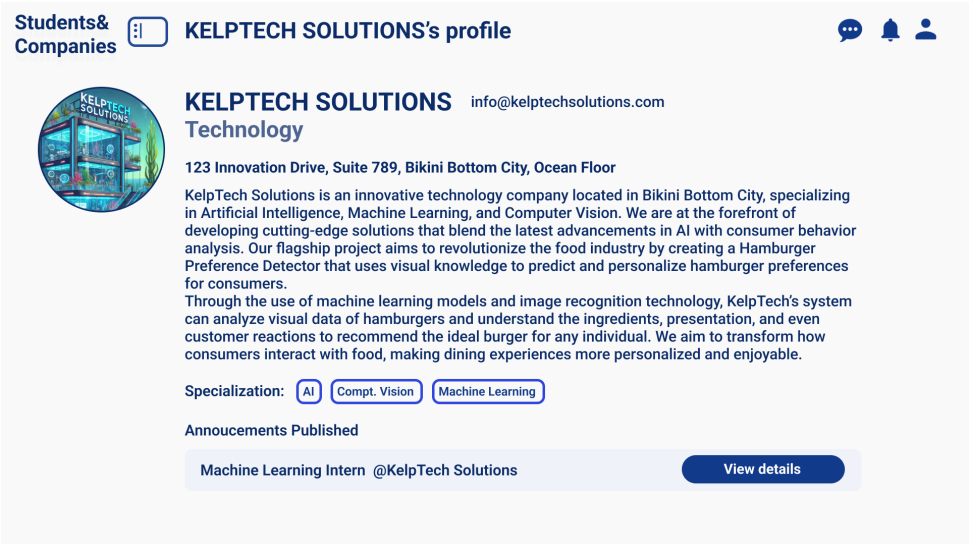


Figure 3.26: Company's profile from other's view

A.6. Student and Company's view

FeedBack and Compalint page

Figure3.27 shows the page where the involved parties can leave feedback and complaints about the internship. There are two blocks, one shows the feedback and complaints history and one is the box where the user can write the feedback or complaint.

Figure 3.27: Feedback and Complaint

A.7. University's view

For the university, the GUI is more simple than the student and company's view, because the university has only the role of monitoring the activities of its students. As other users, the university can use the side menu to navigate to the desired page: *Search*,*Dashboard*.

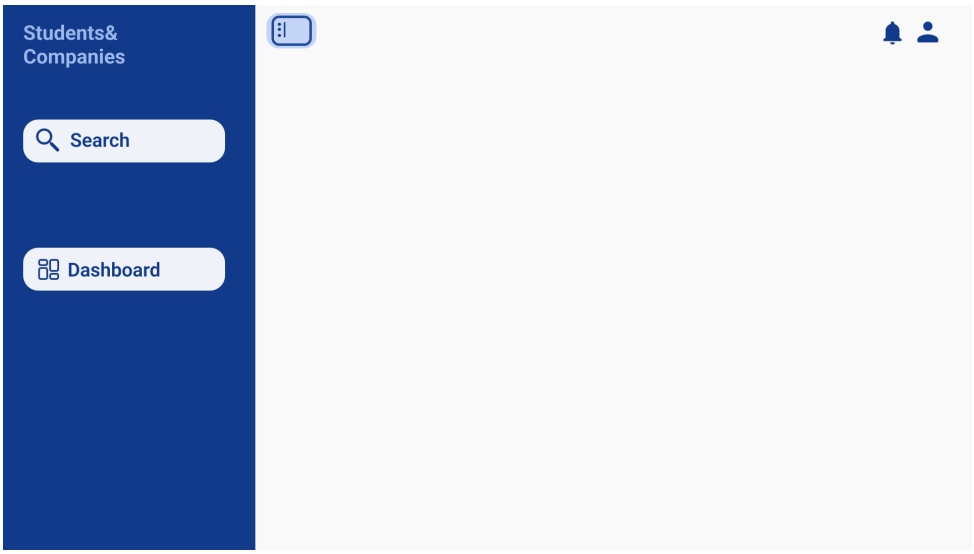


Figure 3.28: University's Side Menu

Dashboard page

The university is directed to the Dashboard page upon logging in. The search bar, the Student list and a specific student's activities overview will be displayed in this page. To change the student, click on the box of certain student in the Student list, the student's activities overview will be updated and showed respectively. If the university wants to view the Student's profile, click on the *Profile details* button.

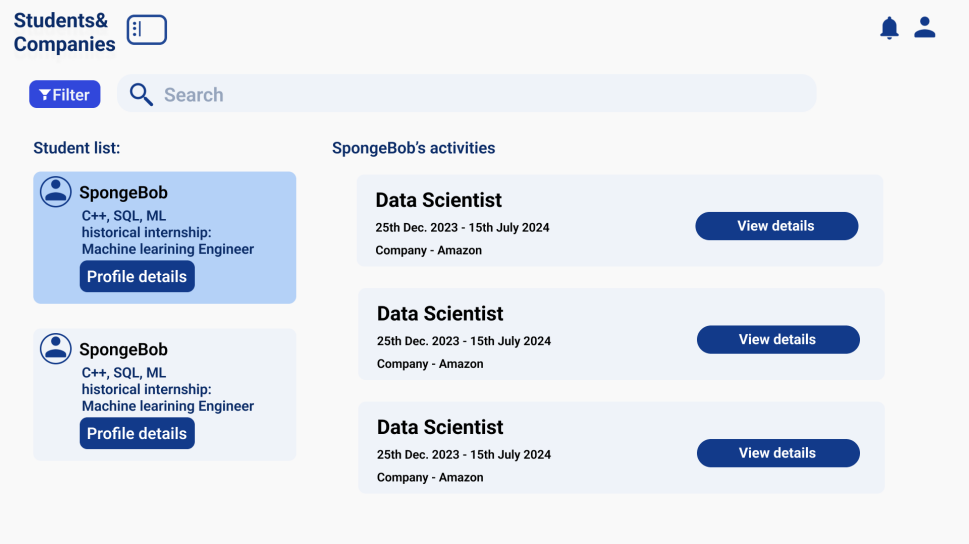


Figure 3.29: University's Dashboard 1

FeedBack and Compalint page

This page will be displayed when the university click on the *View details* button of one of the student's activities. The university can view the feedback and complaints history written by the student and the company during the internship and at the end of the internship.

Figure 3.30: Feedback and Complaint

4 | Requirements Traceability

Authorization Manager

Registration Manager

- **R1**: S&C allows unregistered Users to sign up

Login Manager

- **R2**: S&C allows registered Users to login

User Manager

Profile Modification Manager

- **R3**: S&C allows STs to upload their CV in their profile section
- **R36**: S&C allows Users to modify their own profile data

View Profile Manager

- **R34**: S&C allows UNIs to access the list of all the enrolled STs that are registered on the platform
- **R35**: S&C allows Users to visualize the own and other users' profiles

Search Manager

- **R4**: S&C allows STs to search for internships

Notification Manager

- **R8**: S&C should notify STs when they are selected by the CO for the interview process
- **R9**: S&C should notify STs when a CO is interested in their profile

- **R10**: S&C should notify STs when he is successfully selected for a position
- **R11**: S&C should notify STs when a CO rejects their application
- **R12**: S&C should notify UNIs when their students start an internship
- **R13**: S&C should notify STs when an internship available matches their interest
- **R14**: S&C should notify COs when the deadline for a published internship has expired
- **R15**: S&C should notify COs when the candidate accepts the position
- **R16**: S&C should notify COs when the candidate refuses the position
- **R17**: S&C should notify COs and STs when a new chat message is available
- **R18**: S&C should notify COs when a ST with a CV that corresponds to their needs is available
- **R37**: S&C should notify UNIs when their students register on the platform

Recommendation Module

- **R19**: S&C should be able to analyze the User's data to provide the recommendations to both STs and COs

Internship Manager

Creation Manager

- **R5**: S&C allows COs to create internships by compiling all the information
- **R6**: S&C allows COs to set a deadline for submitting the application to an internship
- **R7**: S&C allows COs to publish internships

View Internship Information Manager

- **R21**: S&C allows STs to visualize information about a published internship
- **R33**: S&C allows UNIs to check the status of the internship records of their students

Selection Manager

- **R23**: S&C allows COs to record STs selection outcomes

Feedback Manager

- **R30**: S&C allows STs and COs to write feedback and complaints relating to the internship experience
- **R31**: S&C allows Users to view feedback and complaints relating to the internship experience

Chat Manager

- **R32**: S&C allows STs and COs to exchange information using the chat, only if the ST is participating or has participated in an internship offered by the company

Application Manager

Submission Manager

- **R20**: S&C allows STs to submit their application for an internship
- **R29**: S&C allows STs to accept or reject the offer after receiving the interview results

View Application Information Manager

- **R22**: S&C allows COs to view the list of all applications that were submitted for a specific internship
- **R28**: S&C allows STs to check the status of their applications

Interview Manager

- **R24**: S&C allows COs to create forms to submit to candidates for the interview process
- **R27**: S&C allows COs to record the results of the interview

Questionnaire Manager

- **R25**: S&C allows candidates to respond to the received forms
- **R26**: S&C allows COs to visualize the responses of the candidates who have replied to the forms

5 | Implementation, Integration and Test Plan

To implement the system more effectively and efficiently, it is crucial to stabilize the development process. This provides a clear understanding of the tasks to be completed, their order of execution, and a reliable reference framework for teams working collaboratively on the same project. Therefore a proper planning for implementation and testing is essential. In following sections will be presented the strategy in this phase of the project: the implementation plan, the integration plan, and the testing plan.

As the platform consists of several components and features, each with varying levels of complexity and dependencies on other components, the Bottom-Up is the suitable approach to optimize the development process. This approach allows team members to work independently on different parts of the system simultaneously. The process begins with the leaves of the "use" hierarchy and progresses upward toward the root. This typically requires the creation of multiple drivers, one for each module. By following this approach, several subsystems will be developed and with the continuous integration of these subsystems, the final system will be created.

A. Implementation Plan

The priority of components to be implemented is determined by their functionality and the dependencies between them. Since the system is divided into a set of components, as described in Section 2, the following list outlines the importance of each component at the development stage, allowing for a smooth and efficient build process.

Component	Priority
Model Module	Very High
Notification Manager	Low
Authentication Manager	Medium
User Manager	High
Internship Manager	High
Application Manager	High
Recommendation Module	Medium
Search Module	Low

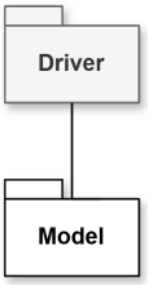
Table 5.1: Implementation Plan

The Model Module serves as the foundation for all other components, meaning it is responsible for managing the platform's data. It facilitates communication between other components and the DBMS, providing an interface for accessing the data. Therefore, it should be implemented in the first stage. Next, there is the Notification Manager supports communication between other components, such as the Authentication Manager, Internship Manager, and Application Manager. After that, the Authentication Manager should be implemented, as it is an essential first step for accessing the platform and controlling access to other components. Then following the dependency tree, the User Manager should then be implemented which is responsible for managing user data and providing an interface for user-related operations. Once the basic components are are implemented, the core functions of the platform—the Internship Manager and Application Manager—can be developed. Afterward, the Search Module and Recommendation Module can be implemented at the last stage, as they are secondary functions that are closely connected to the platform's core functionalities.

B. Integration Plan

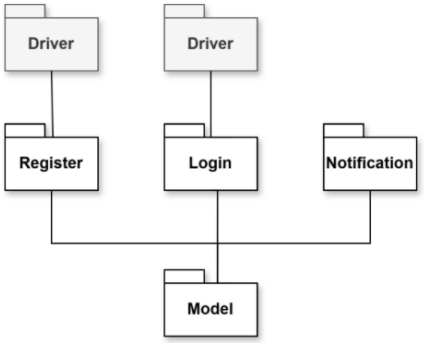
In this section will specify the order in which the components will be integrated and the components of each manager described in the previous paragraph. The integration process will be divided into two main phases: the first phase will focus on integrating the components of each manager, while the second phase will focus on integrating the managers themselves. Also during the integration process, the unit testing and the integration testing will be performed to ensure that the components are working correctly by adding the functionalities to the system.

1. **Model Module:** The connection between the Model Module and the DBMS will be established first, as it is responsible for managing the platform's data and providing an interface for other components to access the data.

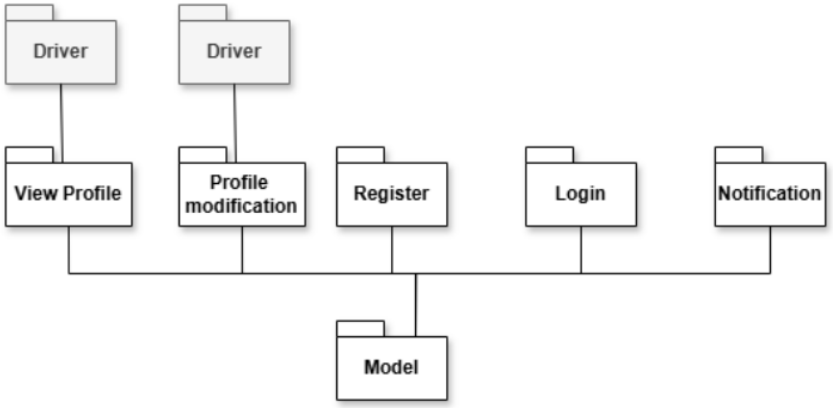


2. **Notification Manager:** The Notification Manager will be integrated next with the Model Module, at this stage, the connection between the Notification Manager and the Model Module will be established and tested.

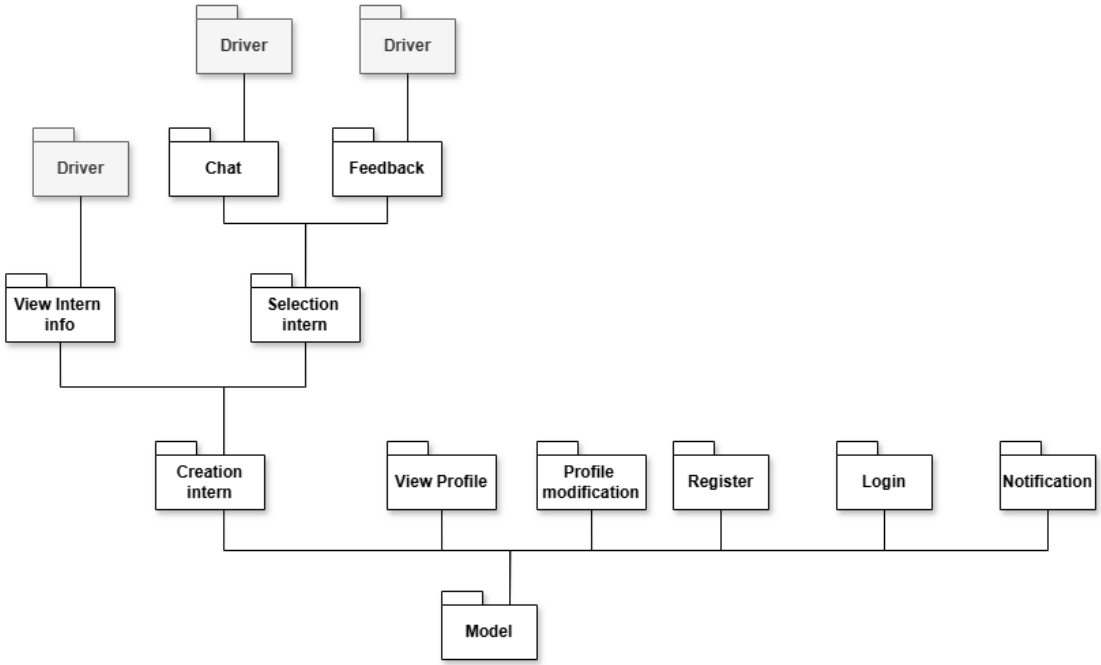
3. **Authentication Manager:** The Authentication Manager, including the Registration and Login Manager, will be integrated next, as these are the basic functionalities for accessing the platform. After interaction with the Model Module and the Notification Manager, their functionalities will be tested to ensure that reading and writing operations in the database work correctly and that notifications are sent properly during the registration and login processes. Additionally, the communication with the external email service will be verified.



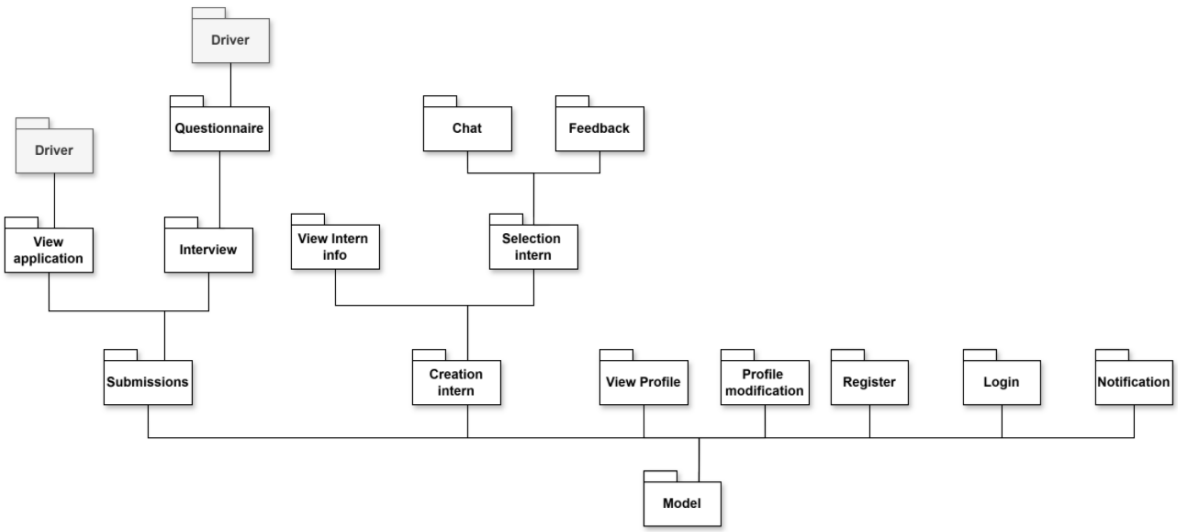
4. **User Manager:** As the last basic component, the View Profile Manager and Profile Modification Manager will be integrated. Since they depend only on components that have already been integrated, unit testing and integration testing can be conducted to ensure that all basic functionalities work correctly before moving to the integration of the core functionalities.



5. **Internship Manager:** The Internship Manager is the first core functionality to be integrated, as it is responsible for managing internship data and providing an interface for internship-related operations. The order of integration of its components is as follows: Creation Manager, View Internship Information Manager, Selection Manager, Chat Manager, and Feedback Manager, reflecting the order of processing internship evolution. Unit testing is crucial at this stage, and extreme cases should be evaluated. Mock data will be used to simulate real internship events. Integration testing of all components integrated so far will also be conducted to ensure that everything works without conflicts or crashes.

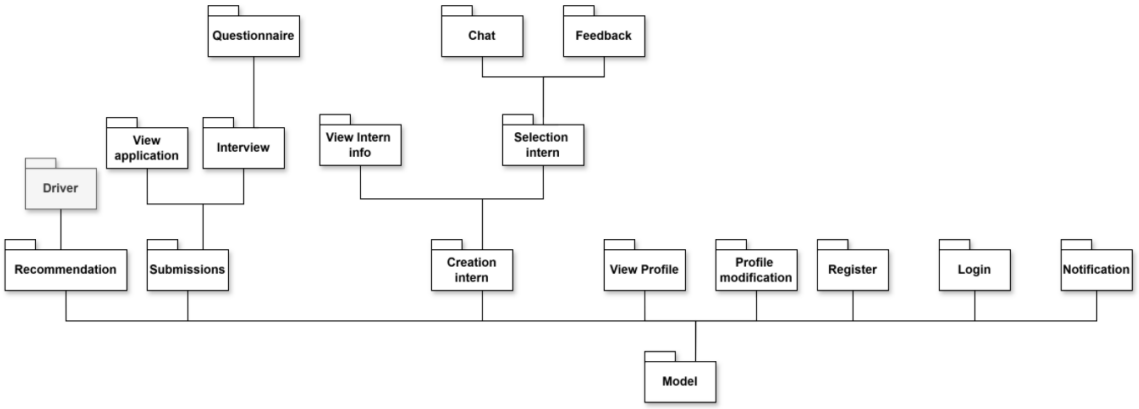


6. **Application Manager:** Next, the Application Manager, which manages applications and controls the selection process, will be integrated. The order of integration of its components is as follows: Submissions Manager, View Application Information Manager, Interview Manager, and Questionnaire Manager. It must collaborate seamlessly with the Internship Manager and the User Manager. Integration testing is critical during this phase since the platform's main functionalities are involved.

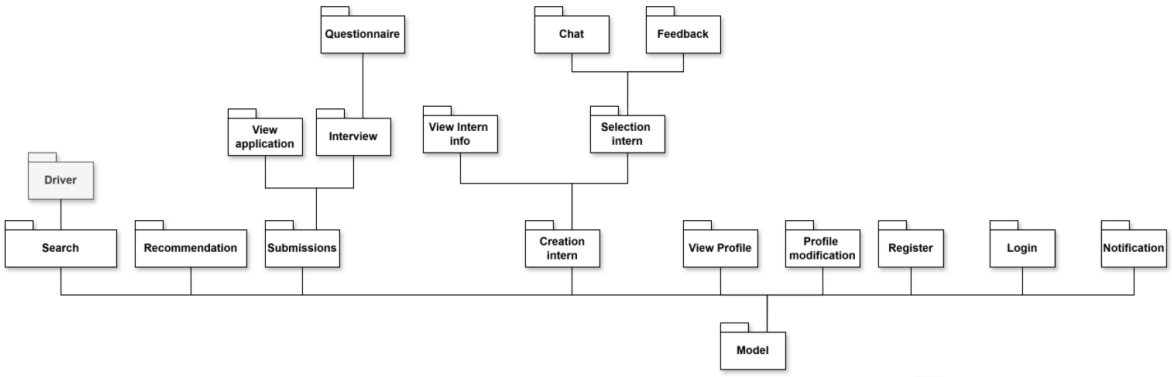


7. **Recommendation Module:** After integrating and testing the core and basic func-

tionalities, the Recommendation Module will be integrated. This module depends on all previously developed components. Testing of the recommendation algorithm will be performed to verify that it provides suitable recommendations based on the user's profile and internship experiences.



8. **Search Module:** Even though the Search Module is not closely related to all components developed so far, it will be integrated at the end of the integration process. This is because it is a secondary function closely connected to the platform's core functionalities. Once all other components have been integrated and tested, the Search Module can be integrated and tested. This approach ensures that any issues with the search functionalities are not caused by other components.



9. After completing the server-side integration, the client-side integration will be performed. The integration of server-side components with client-side components will be processed, and testing of the interaction between server-side and client-side components will be conducted.

C. System Testing

After the iterative testing process during the integration of the components, once the system is fully integrated, it needs to be tested as a whole using other testing techniques. In this section, the testing will focus on identifying any issues with the system's functionalities and determining whether all the requirements are satisfied. During this phase, all different roles of project members will be involved in the testing process, including developers, users, and black-box testers.

- **Functional Testing:** Functional testing is used to check if all functional requirements indicated in the RASD are fulfilled by the software. Communication between different stakeholders and users is important to understand if the functionalities are truly as expected.
- **Load Testing:** Load testing will check if the platform can handle the expected load, and if there are any bugs, such as memory leaks, mismanagement of memory, or buffer overflows. It will also identify the upper limits of the components and help evaluate the optimal architectural options. The system will be tested with increasing workloads until it reaches its capacity.
- **Performance Testing:** Performance testing aims to detect bottlenecks, inefficient algorithms, hardware or network issues, and to identify more efficient solutions to achieve specific goals. It should take into consideration response time, utilization, and throughput of the system.
- **Stress testing:** Stress testing ensures high maintainability and availability. It verifies that the system recovers gracefully after unexpected failures or crashes. The system should be tested by overwhelming its resources or reducing resources, for example, by randomly shutting down and restarting ports on a network switch to observe the system's behavior.
- **User Interface Testing:** User interface testing is essential to ensure that the connection between the client and the server is established smoothly. It will also verify that the user interface is user-friendly and that the user can easily navigate from the user's perspective.

6 | Effort Spent

In following table we provide the tracking of the effort spent by each group member in the development of this document.

Section	Jie Chen	Riccardo Bonfanti
1 – Introduction	0.5 hours	1.5 hours
2 – Architectural Design	6.5 hours	27 hours
3 – User Interface Design	19.5 hours	0 hours
4 – Requirements Traceability	0 hours	2 hours
5 – Implementation, Integration, and Test Plan	4.5 hours	0 hours
Total	31 hours	31.5 hours

Table 6.1: Effort spent for each section

7 References

- The names of *SpongeBob SquarePants* characters referenced in this document are the intellectual property of *Nickelodeon and Viacom International Inc.* We do not claim any ownership of the copyrighted material. The use of these names is intended solely for purposes such as commentary, criticism, analysis, or education, and falls under the "fair use" provisions outlined in Section 107 of the Copyright Act of 1976 (Articolo 70 della Legge sul Diritto d'Autore italiana (Legge n. 633/1941)). This use is non-commercial and transformative in nature, with no intention of infringing upon the copyright holders' rights.
- Lecture Slides of the course "Software Engineering 2", AA 2024/2025, by professor E. Di Nitto (Politecnico di Milano).
- We used Draw.io for the creation of the UML diagrams - `https://www.draw.io/`.
- We used GitHub for version control - `https://github.com/`.
- We used Visual Studio Code IDE for development of the LaTeX document - `https://www.visualstudio.com/`.
- We followed the Politecnico di Milano thesis template for the structure and style of the document - `https://www.overleaf.com/latex/templates/classical-format-thesis-scuola-di-ingegneria-industriale-e-dellinformazione-politecnico-di-milano/dkmvtndqkyxg`.