POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering 2
Design Document

Author(s): **De Matteis Alessandro - 10845281**

Marino Margherita - 10795242

Monaco Giorgio - 10775329

Academic Year: 2024-2025
Version: 1.0
Release date: 06/01/2024

Contents

Contents **i**

1 Introduction **1**
1.1 Scope . . . 1
1.2 Definitions, Acronyms, Abbreviations . . . 1
1.2.1 Definitions . . . 1
1.2.2 Acronyms . . . 2
1.2.3 Abbreviations . . . 2
1.3 Reference Documents . . . 2
1.4 Document Structure . . . 2

2 Architectural Design **5**
2.1 Overview . . . 5
2.2 Component View . . . 7
2.2.1 Presentation Layer . . . 7
2.2.2 Web Server . . . 7
2.2.3 Application Server (Business Logic Layer): . . . 8
2.2.4 Database (Data Layer) . . . 16
2.2.5 External Systems . . . 16
2.3 Deployment View . . . 17
2.3.1 Key Architectural Choices . . . 18
2.4 Component Interfaces . . . 19
2.4.1 Account Interface . . . 19
2.4.2 Internship Interface . . . 19
2.4.3 Enhance Interface . . . 20
2.4.4 Feedback Interface . . . 20
2.4.5 Chat Interface . . . 21
2.4.6 Report Interface . . . 21

2.4.7 Recommendation Interface . . . 22
2.4.8 Notification Interface . . . 22
2.4.9 Database Interface . . . 22
2.4.10 Calendar Interface . . . 23
2.4.11 Web Server Interface . . . 23
2.4.12 Controller Interface . . . 23
2.4.13 Display Interface . . . 24
2.4.14 Open Chat Interface . . . 24
2.4.15 AITools Interface . . . 24
2.4.16 DataAnalytics Interface . . . 24
2.4.17 Email Interface . . . 24
2.5 Runtime View . . . 25
2.6 Selected Architectural Styles and Patterns . . . 56
2.6.1 Model-View-Controller (MVC) . . . 56
2.6.2 Centralized Access . . . 57
2.6.3 Observer Pattern . . . 57
2.6.4 Repository Pattern . . . 57
2.7 Other design decisions . . . 58
2.7.1 Event-Driven Asynchronous Processing . . . 58
2.7.2 Security Enhancements . . . 58
2.7.3 Database Query Optimization . . . 58
2.7.4 Integration Flexibility . . . 58
2.7.5 Error Logging and Monitoring . . . 58

3 User Interface Design 59
3.1 Sign Up [UC1] . . . 59
3.2 Insert an enhanced Internship Insertion [UC3] . . . 61
3.3 Enhance CV [UC4] . . . 63
3.4 Submit a Feedback [UC5] . . . 65
3.5 Accept a Candidate [UC8] . . . 67
3.6 Schedule an Interview [UC14] and Start an Internship [UC15] . . . 71

4 Requirements Traceability 73

5 Implementation, Integration and Test Plan 77
5.1 Overview . . . 77
5.2 Implementation plan . . . 77
5.3 System Testing Plan . . . 85

6 Effort Spent **87**

7 References **89**

7.1 References 89

7.2 Used Tools 89

List of Figures **91**

List of Tables **93**

1 | Introduction

1.1. Scope

This document outlines the domain and objectives of the system, focusing on facilitating interactions and processes among users, including students, companies, and universities. It reviews the high-level architectural design, emphasizing the adoption of established architectural styles such as multi-tier architecture and modular patterns. The chosen design ensures scalability, security, and maintainability while addressing key functional and non-functional requirements of the system. Additionally, this document provides a framework for understanding how the components and subsystems integrate to deliver a cohesive and efficient solution.

1.2. Definitions, Acronyms, Abbreviations

1.2.1. Definitions

- **Software Architecture**: The set of structures needed to reason about the system. These structures comprise software elements, relations among them and properties of both.
- **Recommendation**: The process of identifying and suggesting internships to students and suitable candidates to companies using keyword searches, statistical analyses, or other matching algorithms.
- **Selection Process**: A structured process following a contact in which companies interview and assess students to determine fit.
- **Matching**: The automated or semi-automated process of pairing students with internships based on their profiles and internship requirements.
- **Feedback**: Information provided by students and companies on the quality of the matchmaking process or the internship experience, used to improve recommendations.

- **Internship Monitoring**: Ongoing observation on internship status.
- **Users:** referred to logged-in guests: Universities, Companies and Students.
- **Guest:** non logged visitors.
- **S&C Calendar:** a built-in calendar where all events (start and end date of internship, interviews) are visible and scheduled.

1.2.2. Acronyms

- **S&C**: Students & Companies.
- **UI**: User Interface.
- **API**: Application Programming Interface.

1.2.3. Abbreviations

- **[R*]:** Functional Requirement.
- **[UC*]:** Use Case.
- **[S]** Student.
- **[C]**: Company.
- **[U]**: University.

1.3. Reference Documents

The document is based on the following materials:

- The specification of the RASD and DD assignment of the Software Engineering 2 course a.a 2024/2025.
- Slides of the course on WeBeep.

1.4. Document Structure

Introduction: Provides an overview of the system's goals, scope, and purpose, setting the context for the following sections.

Architectural Design: Describes the architectural styles, patterns, and components of the system, including the detailed organization of layers and the interactions among the components.

User Interface Design: Presents the main user interface designs for meaningful use cases, showcasing how users interact with the system.

Requirements Traceability: Maps the functional and non-functional requirements from the RASD document to the specific components and subsystems described in the architectural design.

Implementation, Integration and Test Plan: Details the system implementation steps, component integration, and testing strategy, including unit, integration, and system tests to ensure requirements are met.

Effort Spent: Overview of the team's time allocation for each document section.

References: Bibliography listing all resources, documentation and software used in preparing this document.

2 | Architectural Design

2.1. Overview

The architectural design of the S&C system is based on a modular and scalable 4-tier architecture. This structure ensures a clear separation of responsibilities, improves maintainability, and supports efficient interactions between components. The application is designed to provide a seamless platform for students, companies, and universities to manage and interact with internships through an integrated system.

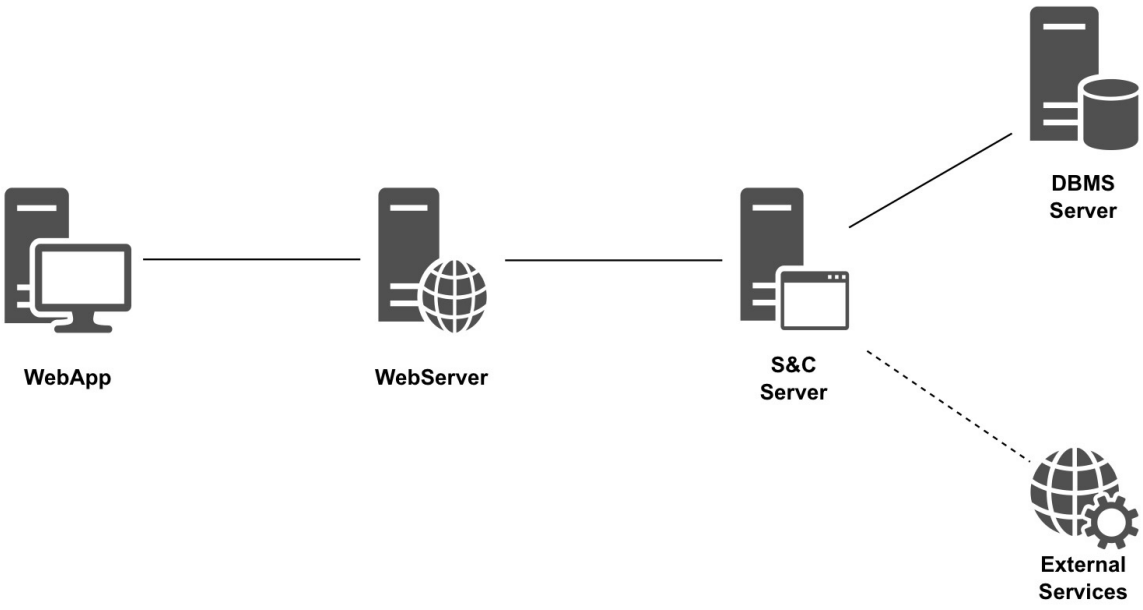

Figure 2.1: Four tier architecture.

The four tiers of the architecture are:

- **Web Application (Presentation Layer):** Acts as the external access point for users, providing an intuitive and responsive user interface.
- **Web Server:** Handles communication between the web application and the application server, managing incoming requests and responses.
- **Application Server (Business Logic Layer):** Hosts the core business logic of the system, processing data and coordinating operations.
- **Database (Data Layer):** Stores all persistent data, ensuring secure and efficient retrieval and updates.

This 4-tier architecture ensures that the system can handle complex operations while maintaining high performance and scalability.

2.2. Component View

The component diagram below illustrates the primary components of the system and their interactions across the four tiers. External Systems, such as AI Tools, Data Analytics Service and eMail Provider, are presented as black-boxes that expose only the interfaces used by the system.

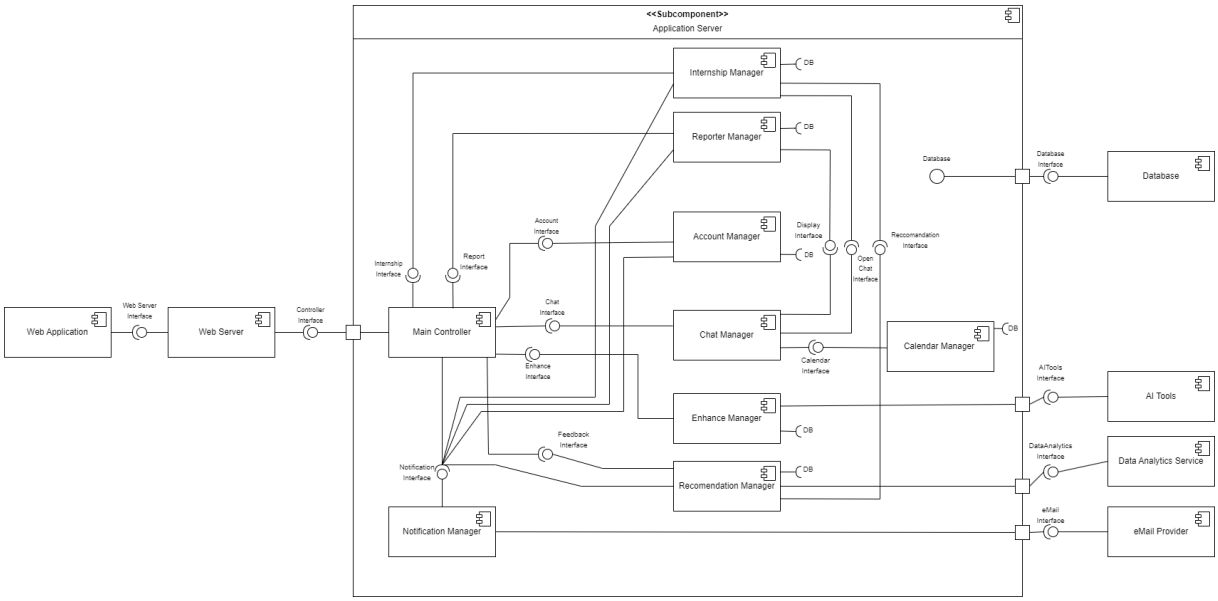

Figure 2.2: Component diagram.

2.2.1. Presentation Layer

The presentation layer serves as the external interface for users (students, companies, and universities). It represents the front-end of the system, including only the ability to perform very basic checks such as the recognitions of empty or incomplete forms, all other parts of the logic are entirely handled by the application server.

As shown before, this layer contains the Web Application that provides a responsive interface for accessing the system through different web browsers.

2.2.2. Web Server

Acts as the intermediary between the web application and the application server, it indeed processes HTTP/HTTPS requests from the web application, forwards requests to the application server and sends back responses. It ensures secure and reliable communication between the user-facing frontend and backend services.

2.2.3. Application Server (Business Logic Layer):

The application server hosts the core functionalities and business rules of the system, including all the logic, and coordinates the information between the presentation layer and the data layer.

As shown before, this layer contain several components:

Main Controller

The Main Controller serves as the central node for communication and coordination within the Application Server. It ensures the proper routing of data and requests between the Web Server and the various backend managers, acting as the primary mediator between the Web Application and the backend services. It simplifies system architecture by managing the complexity of data flow and interactions between components. This design enhances system efficiency and reduces errors, providing a streamlined mechanism for backend operations.

Account Manager

Manages user authentication, account information, including updates and role assignments. It communicates with the Database in order to verify, access, store and delete account information. Indeed, this component is responsible for account creation and for the authentication process.

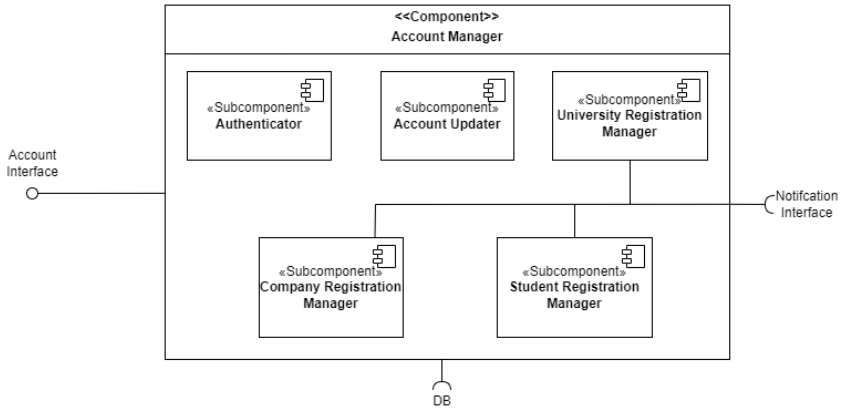

Figure 2.3: Subcomponent diagram Account Manager

It consists of the following subcomponents:

- **Authenticator**: this subcomponent handles the authentications of the users by checking the credentials.
- **AccountUpdater**: it is responsible for updating account data and deleting the account.
- **University Registration Manager**: it manages the registration and the assignment of the role to new universities.
- **Company Registration Manager**: it manages the registration and the assignment of the role to new companies.
- **Student Registration Manager**: it manages the registration and the assignment of the role to new students.

Notification Manager

Processes and sends notifications, in real time as required, to all users. Sends an in-app notification and an email using the interface provided by the email provider.Additionally, it utilizes its own interface to listen for all possible events that may occur.

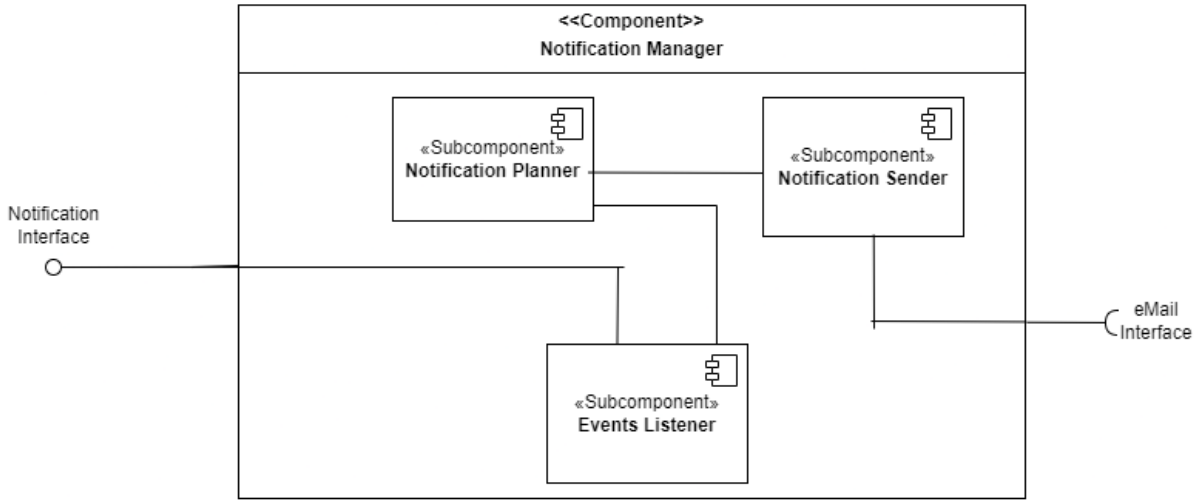

Figure 2.4: Subcomponent diagram Notification Manager

It consists of the following subcomponents:

- **Notification Planner**: It manages the registration of events that trigger the delivery of notifications.
- **Notification Sender**:It handles the actual delivery of notifications to the Web Application.
- **Events Listener**: It is responsible for detecting the occurrence of events that trigger the notification delivery.

Internship Manager

Manages internship listings, including modifications and status updates, while also overseeing the collection of all applications submitted by students. It communicates with the Database in order to verify, access, store and delete internship information.

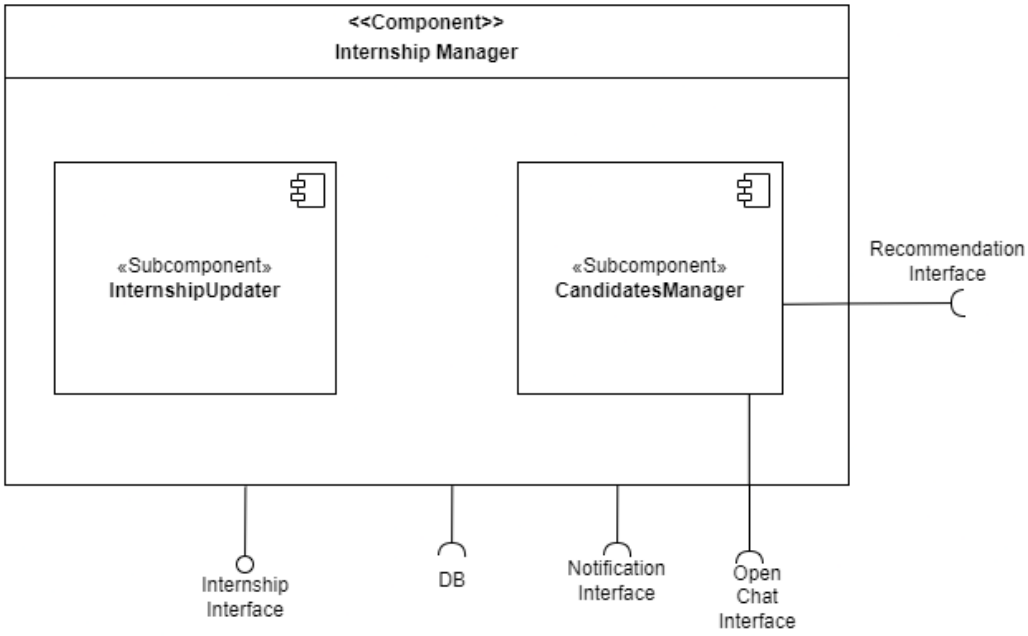

Figure 2.5: Subcomponent diagram Internship Manager.

It consists of the following subcomponents:

- **Internship Updater**: it manages every kind of update made to the internships.
- **Candidate Manager**: it collects every candidate that made application for each internship

Chat Manager

Facilitates real-time communication among students, companies, and universities by managing messages, interviews, and offers. Additionally, it interacts with the database to verify, access, store, and delete messages along with their associated timestamps. Finally, it is responsible for handling the submission of complaints and, if necessary, closing a chat.

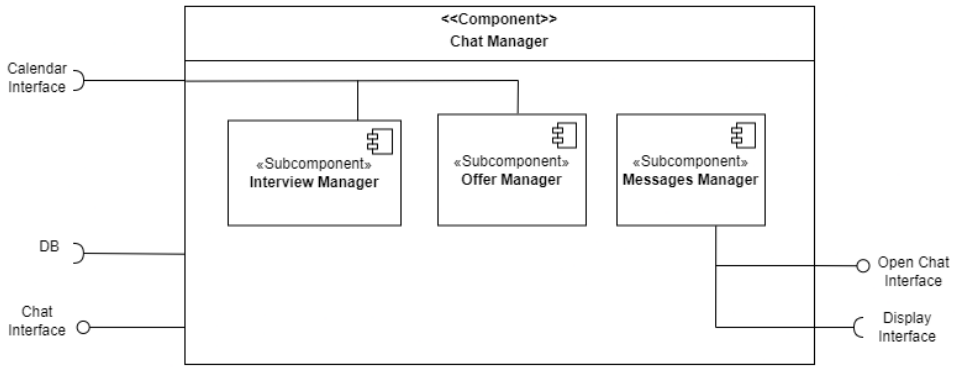

Figure 2.6: Subcomponent diagram Chat Manager.

It consists of the following subcomponents:

- **Interview Manager**: It is responsible for managing interview proposals made by companies and adding the event to the calendar.
- **Offer Manager**: It is responsible for managing contract proposals made by companies and, in the case of a positive outcome, adding the start and end dates of the internship to the calendar.
- **Messages Manager**: It is responsible for facilitating communication between the two parties, ensuring a real-time chat experience with zero delays and no loss of information. It also takes part in the potential process of terminating an internship by offering the interface for closing the chat.

Calendar Manager

Manages events and schedules for internships and deadlines, while providing an interface for other components to add potential events to the calendar.

Enhance Manager

It aims to provide services integrated with AI Tools to offer suggestions regarding CVs and internship postings.

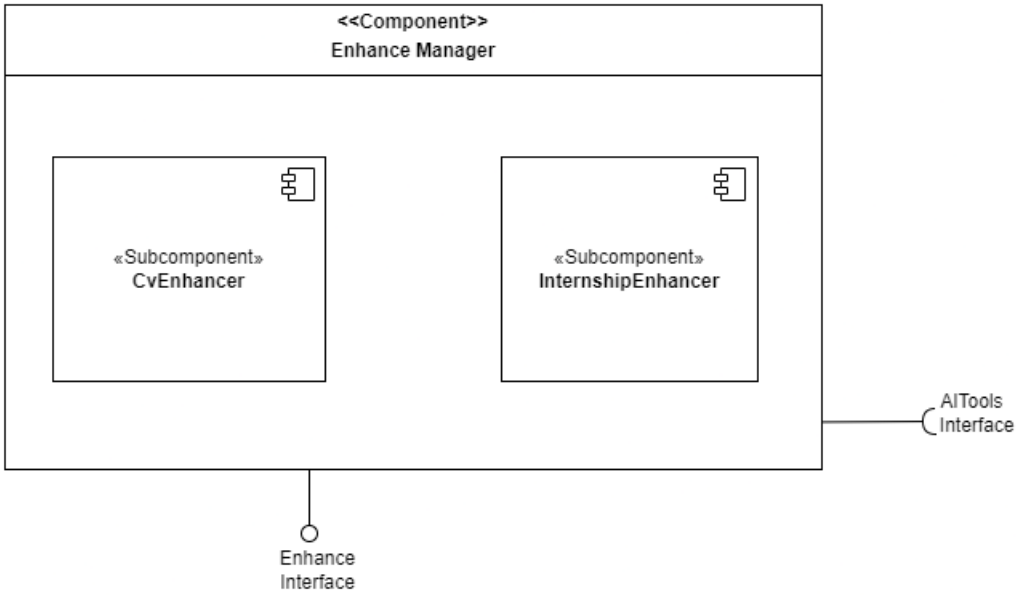

Figure 2.7: Subcomponent diagram Enhance Manager.

It consists of the following subcomponents:

- **Cv Enhancer**: It assists students in enhancing their CVs by analyzing them and providing improvement suggestions through AI tools.
- **Internship Enhancer**: It assists companies in improving their postings by analyzing them and providing suggestions for enhancements through AI tools.

Recommendation Manager

It aims to provide personalized suggestions by leveraging external data analytics services and all data stored in the database, in addition to deliver real-time suggestions, it utilizes the notification interface. Finally it also handles feedback's submission.

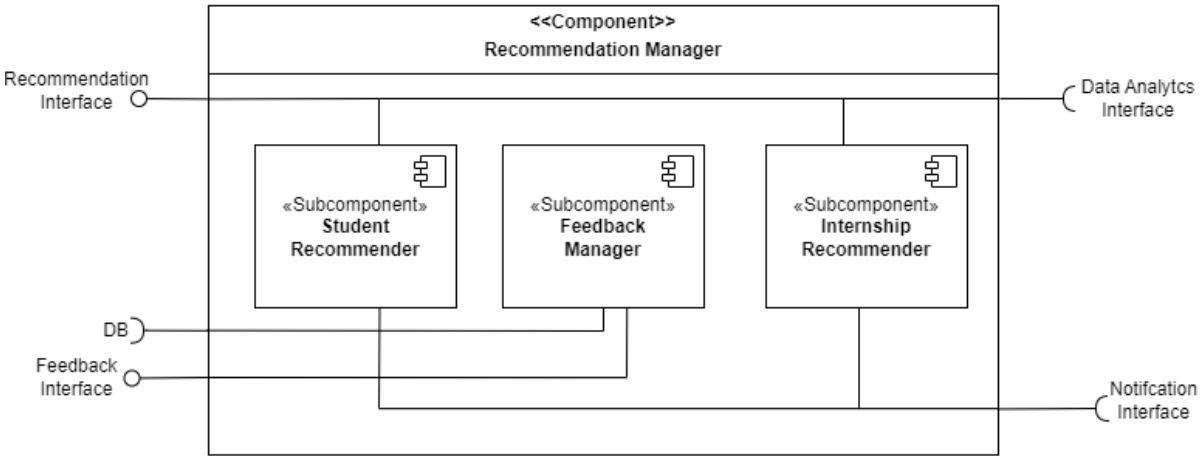

Figure 2.8: Subcomponent diagram Recomendation Manager.

It consists of the following subcomponents:

- **Student Recommender**: It is responsible for managing the recommendations generated for students.
- **Internship Recommender**: It is responsible for managing the recommendations generated for companies.
- **Feedback Manager**:It is responsible for collecting feedback and saving it in the database, ensuring it can be utilized for generating recommendations.

Reporter Manager

It is responsible for providing interfaces to the chat for uploading information or submitting complaints during an internship. It is also responsible for managing and storing them, and, if deemed necessary by the university, requesting the closure of the chat through the interface provided by the chat manager.

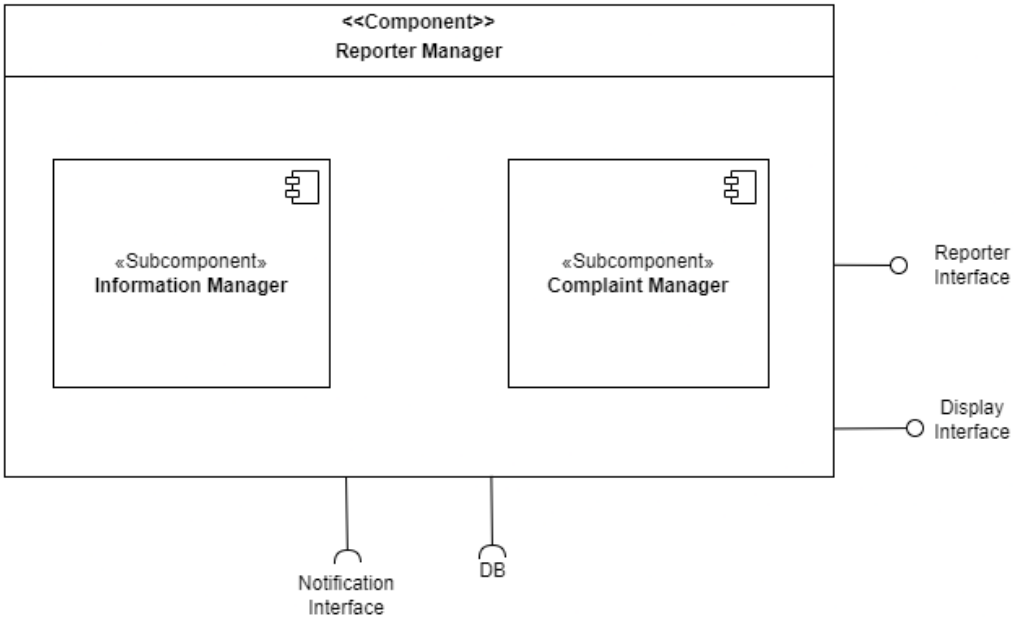

Figure 2.9: Subcomponent diagram Reporter Manager.

It consists of the following subcomponents:

- **Information Manager**: it handles every information added through the chat.
- **Complaint Manager**: It stores and handles all complaints submitted via the chat and, if necessary, is responsible for closing the chat through the corresponding interface.

2.2.4. Database (Data Layer)

The Database serves as centralized storage for all application data, directly connected to the Application Server for all read and write operations. It ensures data consistency and supports efficient querying through a well-structured relational model, optimized with indexing and normalization.

Designed with strict access controls, it guarantees secure interactions, with only authorized components managing data. The Database supports seamless integration with the Application Server via defined interfaces, ensuring a clear separation of concerns. Backup and recovery mechanisms further enhance reliability, making it a robust and efficient foundation for the system's data management needs.

2.2.5. External Systems

These external system communicates with the *Application Server* via APIs.

- **AI Tools:** Enhance user CVs and internship descriptions with intelligent processing.
- **Data Analytics Services:** Provide insights and recommendations for internships based on user preferences and behavior.
- **Email Provider:** Sends notification emails to users through the Notification Manager.

2.3. Deployment View

The deployment view represents the architecture that separates the system into the Presentation Tier, Application Tier (divided in Web Server and Application Server) and Data Tier, ensuring modularity, scalability, and maintainability.

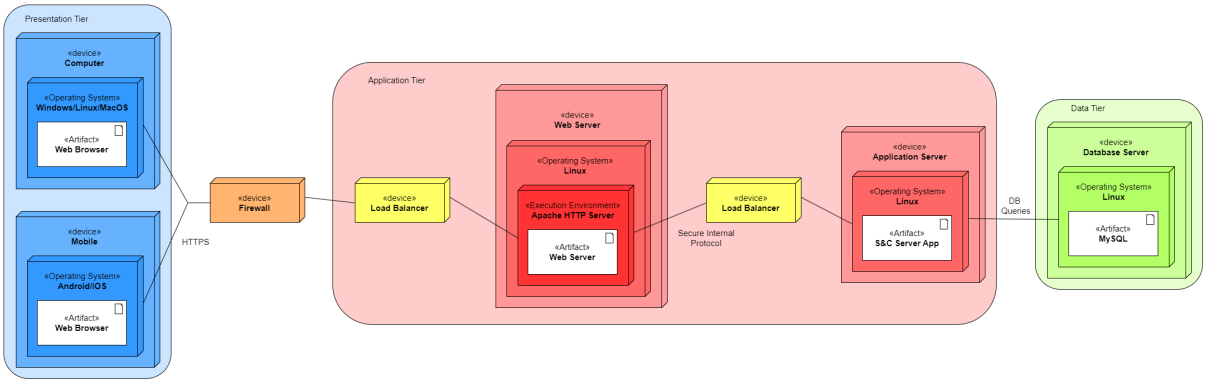

Figure 2.10: Deployment view.

- **Presentation Tier:** Includes computers (Windows/Linux/macOS) and mobile devices (Android/iOS). These devices host web browsers that serve as the interface for users (students, companies, and universities). The communication between the devices and the system passes through a Firewall to ensure secure external connections. All communication is secured via HTTPS.
- **Application Tier:** The Firewall acts as the first layer of defense for incoming connections. Load Balancers are used at both the Web Server and Application Server levels to distribute requests evenly, prevent overload, and ensure high availability.
  - **Web Server:** Hosted on a Linux environment, the Web Server uses an Apache HTTP Server as its execution environment. The Web Server handles incoming HTTP/HTTPS requests from the Presentation Tier and forwards them to the Application Server.
  - **Application Server:** Also hosted on a Linux environment, the Application Server contains the S&C Server App, which implements the business logic of the system. Communicates with both the Web Server and the Database Server.
- **Data Tier:** It's the Database Server, a centralized storage system running on Linux. The MySQL artifact manages all application data, ensuring consistency and supporting efficient querying. Directly connected to the Application Server for all read/write operations, enabling real-time updates and reliability.

2.3.1. Key Architectural Choices

- The architecture follows a clear separation of concerns by isolating the user interface, application logic, and data management into distinct tiers. This modular design enhances scalability and simplifies system maintenance.
- The database employs a centralized storage strategy, serving as a single source of truth to eliminate data duplication and ensure consistency across the system.
- Security is prioritized through the use of HTTPS for all external communications and secure internal protocols between system components, safeguarding data and preventing unauthorized access.
- Load balancers are strategically integrated to manage increased traffic, distribute requests evenly, and maintain high system availability under varying loads.

2.4. Component Interfaces

2.4.1. Account Interface

The **Account Interface**, offered by the Account Manager, helps the Main Controller to manage user account operationns. Its key methods and corresponding functionalities are as follows:

- `createAccount()`: It allows the callback of a new form to be filled in for account creation via the *Account Manager*.
- `SignUp(userdata UserData)`: It allows managing the user registration process through the *Account Manager*.
- `selectRole(string Role)`: It allows role selection during registration or login by interacting with the *Authenticator*.
- `login(String email, String password)`: It allows authenticating the user through the *Authenticator*.
- `openProfile(profile Profile)`: It allows opening the personal profile page using the *AccountUpdater*.
- `search(string CompanyName)`: It allows searching for company profiles using the *AccountUpdater*.
- `ClickOnCompany(profile Company)`: It allows accessing a company's profile during a search via the *AccountUpdater*.
- `clickOnProfile(profile Profile)`: It allows viewing another user's profile using the *AccountUpdater*.
- `clickOnNotification()`: It allows managing notification interactions via the *AccountUpdater*.

2.4.2. Internship Interface

The **Internship Interface**, offered by Internship Manager, helps the Main Controller in managing internships and candidates. Its methods include:

- `postInternship()`: It allows the callback of a new internship form to be filled in through the *InternshipUpdater*.

- `uploadInternship(internshipdata InternshipData)`: It allows uploading internship details via the *InternshipUpdater*.
- `InternshipInfo(internshipdata InternshipData)`: It allows sending a fully compiled form to the *InternshipEnhancer*.
- `clickOnInsertion(string Insertion)`: It allows viewing an internship insertion using the *InternshipUpdater*.
- `getInsertion()`: It allows fetching personal internship insertions from the *InternshipUpdater*.
- `apply(internship Internship)`: It allows a student to apply for an internship through the *CandidatesManager*.
- `getCandidates(internship Internship)`: It allows retrieving a list of candidates using the *CandidatesManager*.
- `acceptStudent()`: It allows accepting a candidate for an internship via the *CandidatesManager*.
- `clickOnCandidate()`: It allows viewing a candidate's profile using the *CandidatesManager*.
- `contactStudent(student Student)`: It allows contacting a specific student through the *InternshipUpdater*.
- `accept(internship Internship)`: It allows accepting an internship application using the *CandidatesManager*.

2.4.3. Enhance Interface

The **Enhance Interface**, offered by Enhancer Manager, helps the Main Controller in managing the enhancement of CVs. The method is:

- `enhance(cv CV)`: It allows sending a request to enhance a CV to the *InternshipEnhancer*.

2.4.4. Feedback Interface

The **Feedback Interface**, offered by Feedback Manager (Recommendation Manager), helps the Main Controller in handling periodic feedback requests and submission after closing an internship's insertion. Its methods include:

- `closeInsertion(string Insertion)`: It allows closing an internship insertion window and obtaining a periodic feedback request via the *Feedback Manager*.
- `submitFeedback(feedback Feedback)`: It allows submitting feedback using the *Feedback Manager*.

2.4.5. Chat Interface

The **Chat Interface**, offered by Chat Manager, helps the Main Controller in facilitating user communication, interviews, and contract offers. It includes:

- `clickOnChat()`: It allows opening the personal chat page via the *Messages Manager*.
- `selectChat(profile Company/Student)`: It allows opening a chat related to the parameter inserted using the *Messages Manager*.
- `clickOnInterview()`: It allows the callback of a new interview form to be filled in via the *Interview Manager*.
- `scheduleInterview(date Date,time Time,string InterviewData)`: It allows sending a request to schedule an interview to the *Interview Manager*.
- `acceptInterview()`: It allows accepting a scheduled interview via the *Interview Manager*.
- `declineInterview()`: It allows declining an interview through the *Interview Manager*.
- `clickOnOffer()`: It allows viewing an internship offer using the *Offer Manager*.
- `proposeInternship(date sDate,date eDate,string InternshipDetails)`: It allows offering an internship contract using the *Offer Manager*.
- `acceptOffer()`: It allows accepting an internship offer via the *Offer Manager*.
- `declineOffer()`: It allows declining an offer through the *Offer Manager*.

2.4.6. Report Interface

The **Report Interface**, offered by Report Manager,helps the Main Controller in managing complaints and information updates. It includes:

- `clickOnComplaint()`: It allows the callback of a new complaint form to be filled in through the *Complaint Manager*.

- `submitComplaint(complaint Complaint)`: It allows submitting a complaint via the *Complaint Manager*.
- `clickOnInformation()`: It allows the callback of a new information form to be filled in using the *Information Manager*.
- `submitInformation(string Information)`: It allows submitting information through the *Information Manager*.
- `InterruptInternship(student Student,internship Internship)`: It allows sending-ing a request to interrupt an internship via the *Reporter Manager*.

2.4.7. Recommendation Interface

The **Recommendation Interface**, offered by Reccomendation Manager, helps Internship Manager in integrating analytics to recommend students for internships. Its method is:

- `clickOnCandidates()`: It allows to trigger the generation of the recommended students for a specific internship via *Student Recommender and Data Analitycs Service*.

2.4.8. Notification Interface

The **Notification Interface** handles event-driven notifications. It's offered by the *Notification Listener* to all other components to provide real-time notifications. It includes:

- `sendEvent()`: It enables other components to dispatch events that subsequently trigger notifications via the *Event Listener*.

2.4.9. Database Interface

The **Database Interface** provides storage and retrieval operations to all other system's components. Its methods include:

- `query()`: It allows retrieving data from the database.
- `createNewAcc(userdata UserData)`: It allows the storage of new account details.
- `addInternship(internship Internship)`: It allows the storage of internship details to the database.
- `addFeedback(feedback Feedback)`: It allows the storage of feedback in the database.

- `applyInternship(internship Internship)`: It enables the storage of a student's application for an internship.
- `addInterview(interview Interview)`: It allows storing interview's details in the database.
- `acceptStudent(student Student)`: It facilitates the recording of a student's acceptance status and enables the initiation of a chat.
- `addAcceptedStudent(student Student,internship Internship)`: It facilitates the documentation of a student's initiation of work for the specified internship..
- `addEmploymentCOntract(string Student,internship Internship,date sDate,date eDate,String internshipDetails)`: It allows storing contract's details.
- `deleteEmployment(student Student,internship Internship)`: It allows deleting contract's details, in case of interruption.
- `addComplaint(complaint Complaint)`: It allows storing every submitted complaint.
- `addInfromation(string Information)`: It enables storing every submitted information in the database.

2.4.10. Calendar Interface

The **Calendar Interface** integrates the *Calendar Manager* with *Chat Manager* for scheduling events on the calendar from the chat page. Methods include:

- `addCalendar(interview Interview,date Date,time Time)`: It enables the addition of interview schedules to the calendar.
- `addCalendar(string InternshipDetails,date sDate,date eDate)`: It allows scheduling internship's start and end date on the calendar.

2.4.11. Web Server Interface

The **Web Server Interface** provides access to the web application via the web browser.

2.4.12. Controller Interface

The **Controller Interface** helps the Web Server to interact wiht the Main Controller.

2.4.13. Display Interface

The **Display Interface**, offered by the Report Manager, allows Chat Manager to retrieve all the information, regarding a chat, managed by the report Manager.

- `getsReport(chat Chat)`: It allows retrieving all the information and complaint submitted in tha chat.

2.4.14. Open Chat Interface

The **Open Chat Interface**, offered by Chat Manager, helps Internship Manager in creating new chat when a candidate is accepted.

- `openChat(student Student)`: It allows the creation of a new chat between company and an accepted candidate.

2.4.15. AITools Interface

The **AITools Interface** provided to Enhance Manager, contains the essential methods for leveraging AI algorithms on external system to generate suggestions for improving CVs and enhancing internship postings.

2.4.16. DataAnalytics Interface

The **DataAnalytics Interface** provided to the Recommendation Manager, contains the essential methods for supplying recommendation algorithms on an external system, enabling the generation of tailored recommendations for companies and students.

2.4.17. Email Interface

The **Email Interface** handles external communication through the *Email Provider*, ensuring smooth email notifications.

2.5. Runtime View

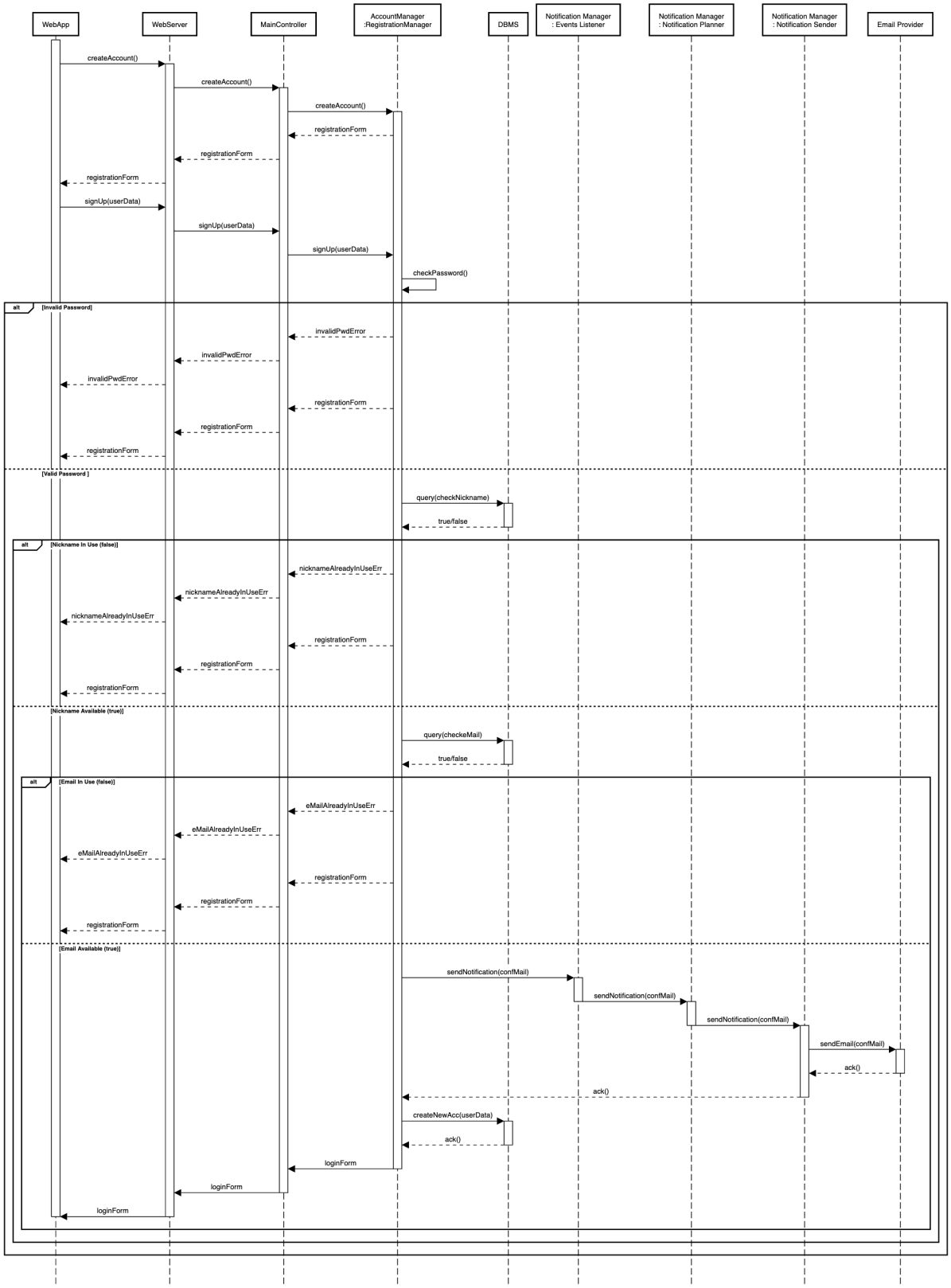

Figure 2.11: [UC1] Sign Up.

The sequence diagram outlines the sign-up process for students, companies, or universities. It starts with the user requesting an account through the WebApp, which retrieves a registration form via the WebServer and MainController. After the user submits their data, the RegistrationManager validates the password, username, and email by querying the database.

If any issues arise, such as an invalid password, duplicate username, or email, appropriate error messages are displayed, and the user is redirected to the registration form. Upon successful validation, the Notification Manager sends a confirmation email using the Email Provider. Finally, the RegistrationManager creates the account in the database and redirects the user to the login form, completing the process.

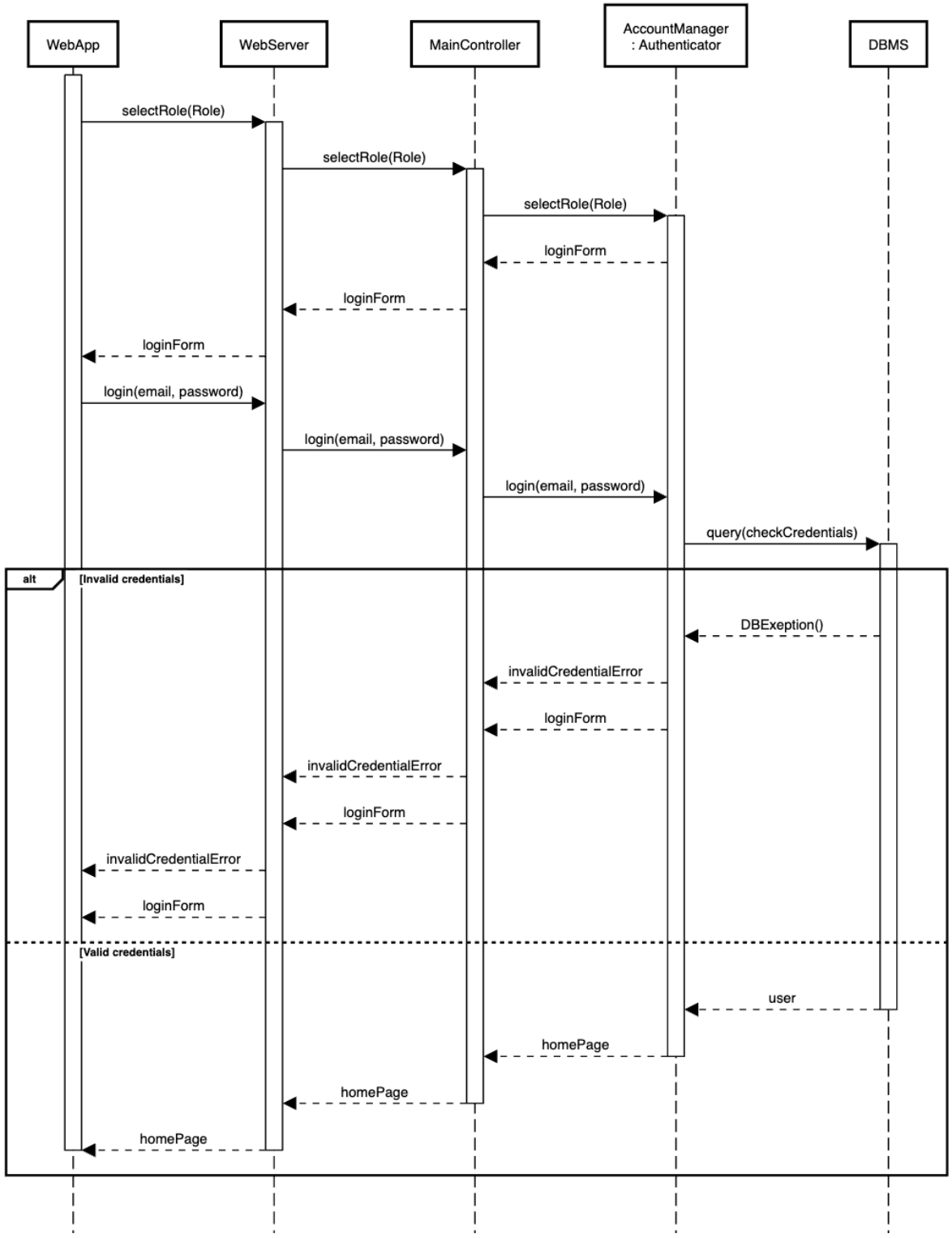

Figure 2.12: [UC2] Login.

The sequence diagram outlines the login process: the user selects their role (student, company, or university) on the WebApp, triggering a request to the WebServer, which passes it to the MainController and then the Authenticator. The Authenticator retrieves the login form from the AccountManager and sends it back to the WebApp.

When the user submits their email and password, the WebApp forwards the credentials to the Authenticator, which verifies them with the database. If invalid, the Authenticator returns an error message via the WebServer and WebApp. If valid, the database sends the user's information, and the Authenticator provides the home page. This ensures secure login validation and proper feedback.

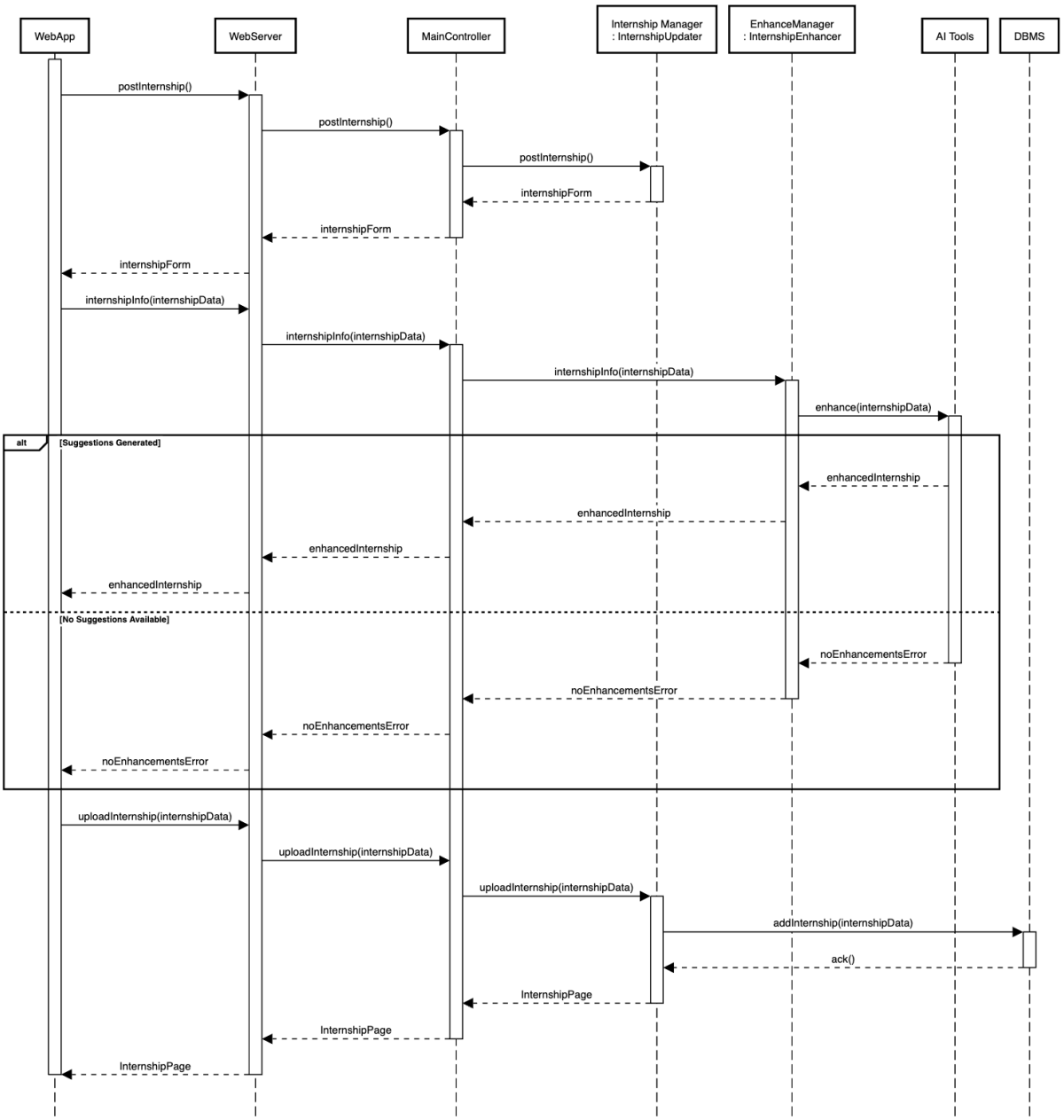

Figure 2.13: [UC3] Insert Enhanced Internship Insertion.

The sequence diagram illustrates the process of a Company, through the WebApp, inserting an enhanced internship. The process starts with the Company initiating a postInternship() request via the WebApp, which retrieves the internship form through the WebServer, MainController, and Internship Manager. The Company submits the internship data, which is enhanced by the EnhanceManager using AI Tools. If enhancements are available, they are returned to the WebApp; otherwise, an error message is sent.

After reviewing the enhancements, the Company uploads the internship data. This data is processed through the system and stored in the Database Management System. Once stored, an acknowledgment is sent back, and the Company, through the WebApp, receives the updated internship page as confirmation.

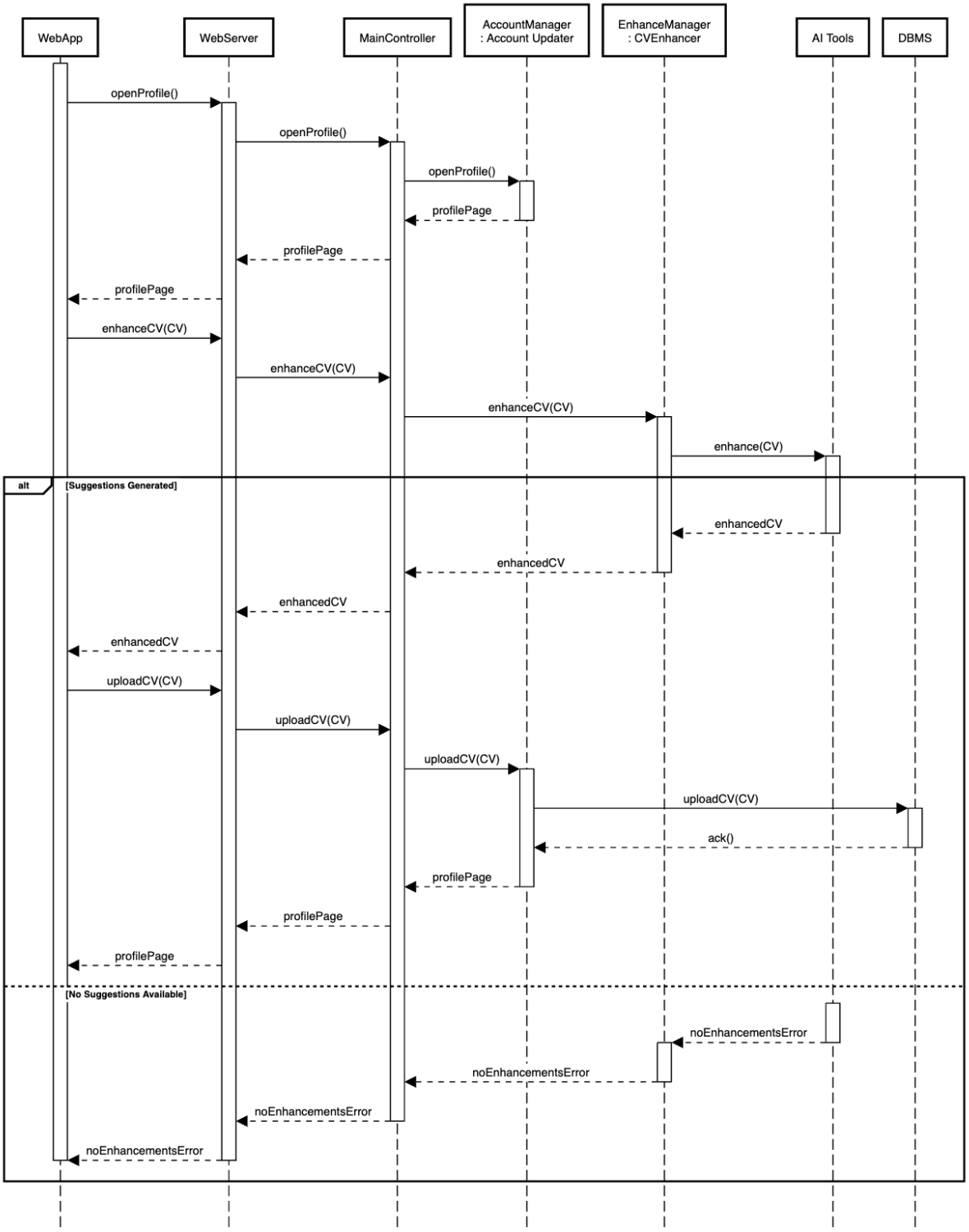

Figure 2.14: [UC4] Enhance CV.

The sequence diagram illustrates the process of a Student enhancing their CV via the WebApp. The process starts with the Student sending an openProfile() request, which retrieves the profile page through the WebServer, MainController, and AccountManager. Once the profile page is loaded, the Student submits their CV for enhancement.

The CV is processed by the EnhanceManager using AI Tools. If enhancements are available, the enhanced CV is returned to the Student; otherwise, an error message is sent. After reviewing the enhancements, the Student uploads the updated CV. This CV is processed through the system and stored in the Database Management System. Finally, an acknowledgment is sent back, and the Student receives the updated profile page as confirmation.

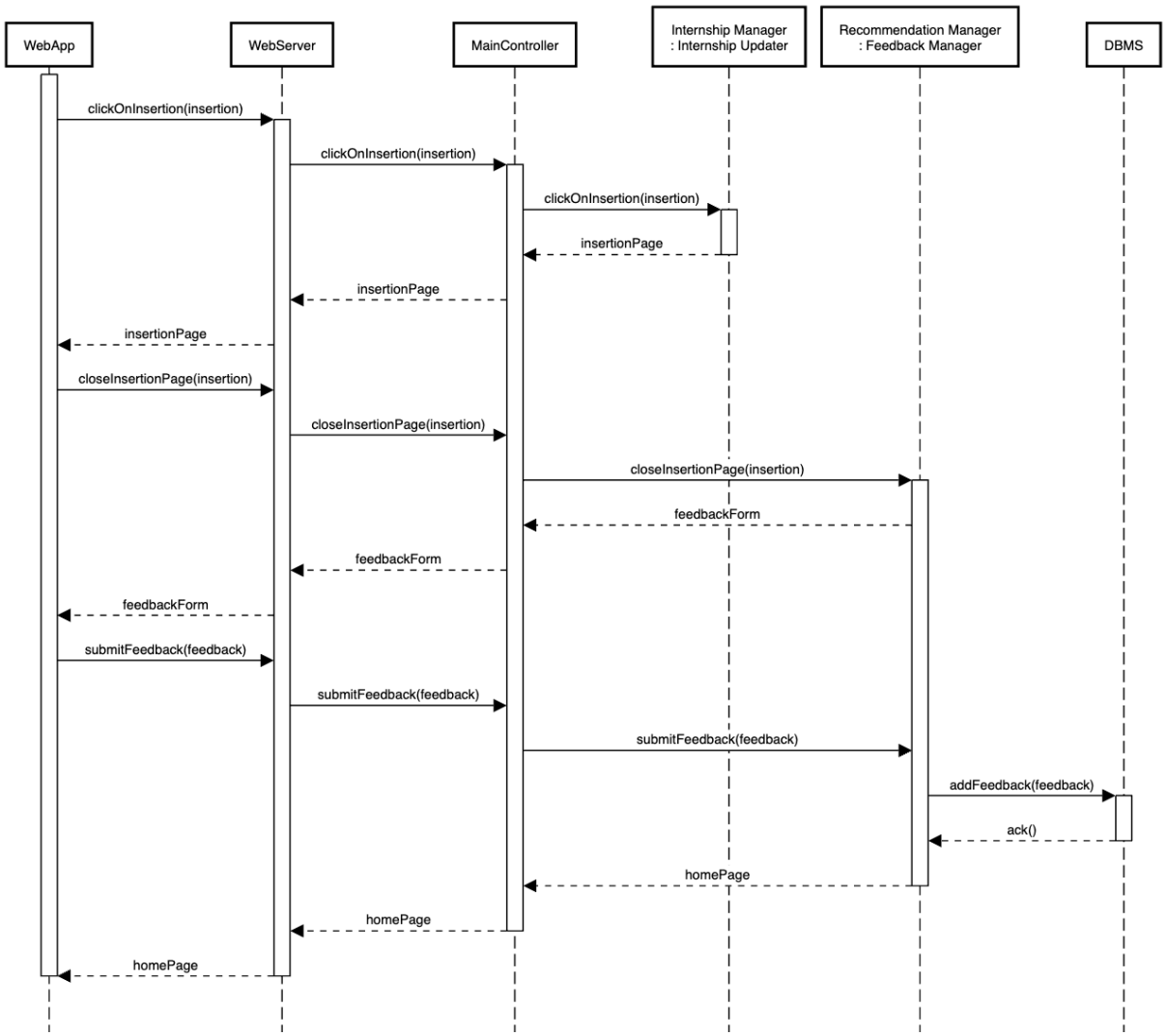

Figure 2.15: [UC5] Submit a Feedback.

The sequence diagram illustrates the process of submitting feedback in the system. It begins with the user clicking on an insertion via the WebApp, which retrieves the insertion page through the WebServer, MainController, and Account Manager. After viewing the insertion, the user closes the insertion page, triggering the feedback process.

The WebApp requests a feedback form, which is generated by the Recommendation Manager and passed back through the system. The user submits their feedback via the WebApp, and it is processed through the WebServer, MainController, and Recommendation Manager. The feedback is stored in the Database Management System, where it will be used by the Data Analytics service to improve the recommendation engine in the future. Once stored, an acknowledgment is sent back, and the user is redirected to the home page as confirmation of successful feedback submission.

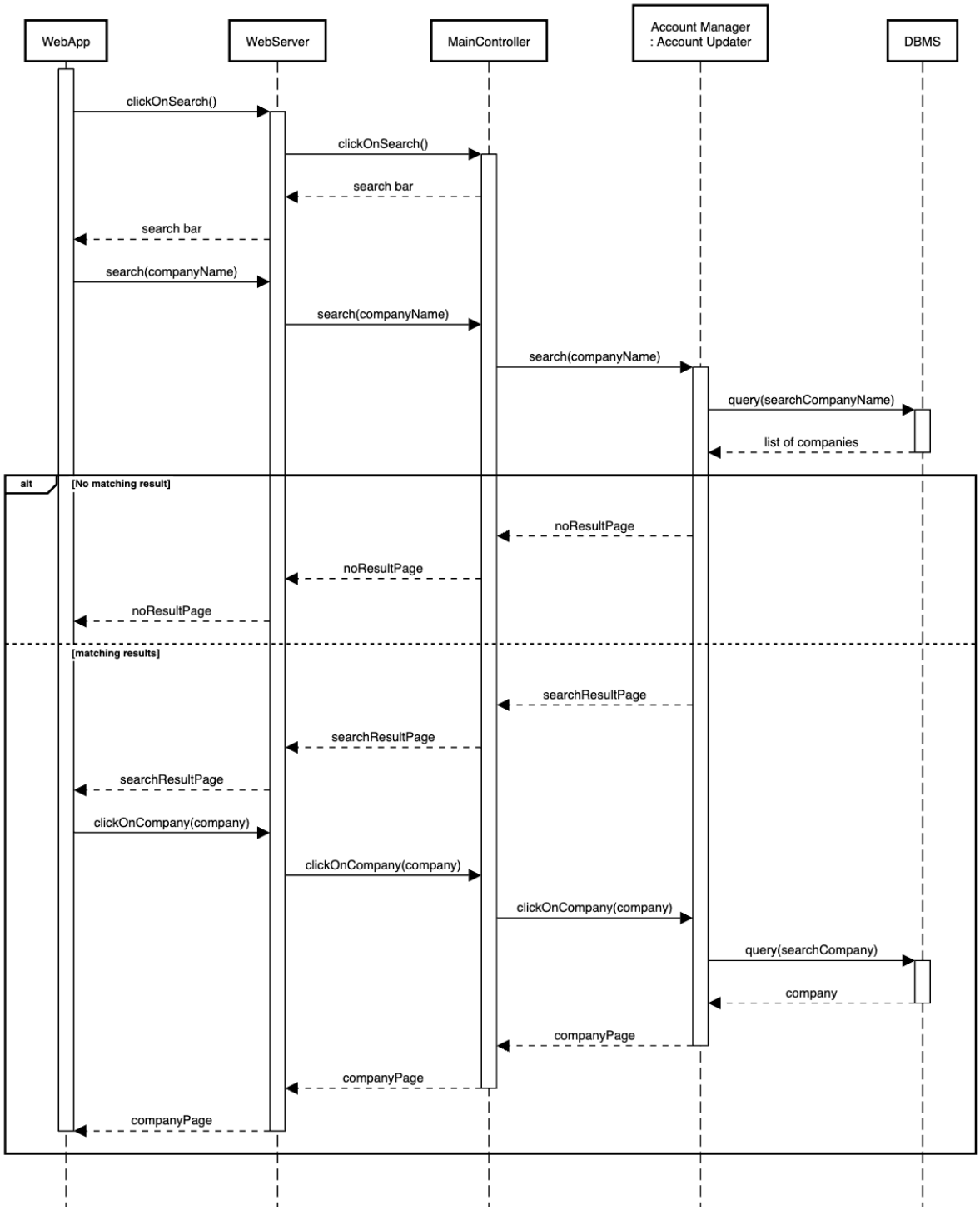

Figure 2.16: [UC6] Search for a Company.

The sequence diagram illustrates the process of searching for a company. It begins with the user initiating a search via the WebApp by clicking on the search option. This request is passed through the WebServer and MainController, which returns a search bar to the WebApp. The user enters the company name in the search bar, and the query is sent through the WebServer and MainController to the Account Manager.

The Account Manager queries the Database Management System for matching companies. If no results are found, a "no result" page is returned to the user. If matching results are found, a search result page is sent back.

The user can then click on a specific company from the results, triggering a request that is passed through the WebServer, MainController, and Account Manager. The Account Manager queries the database for the selected company's details, which are retrieved and sent back to the WebApp as a company page, completing the process.

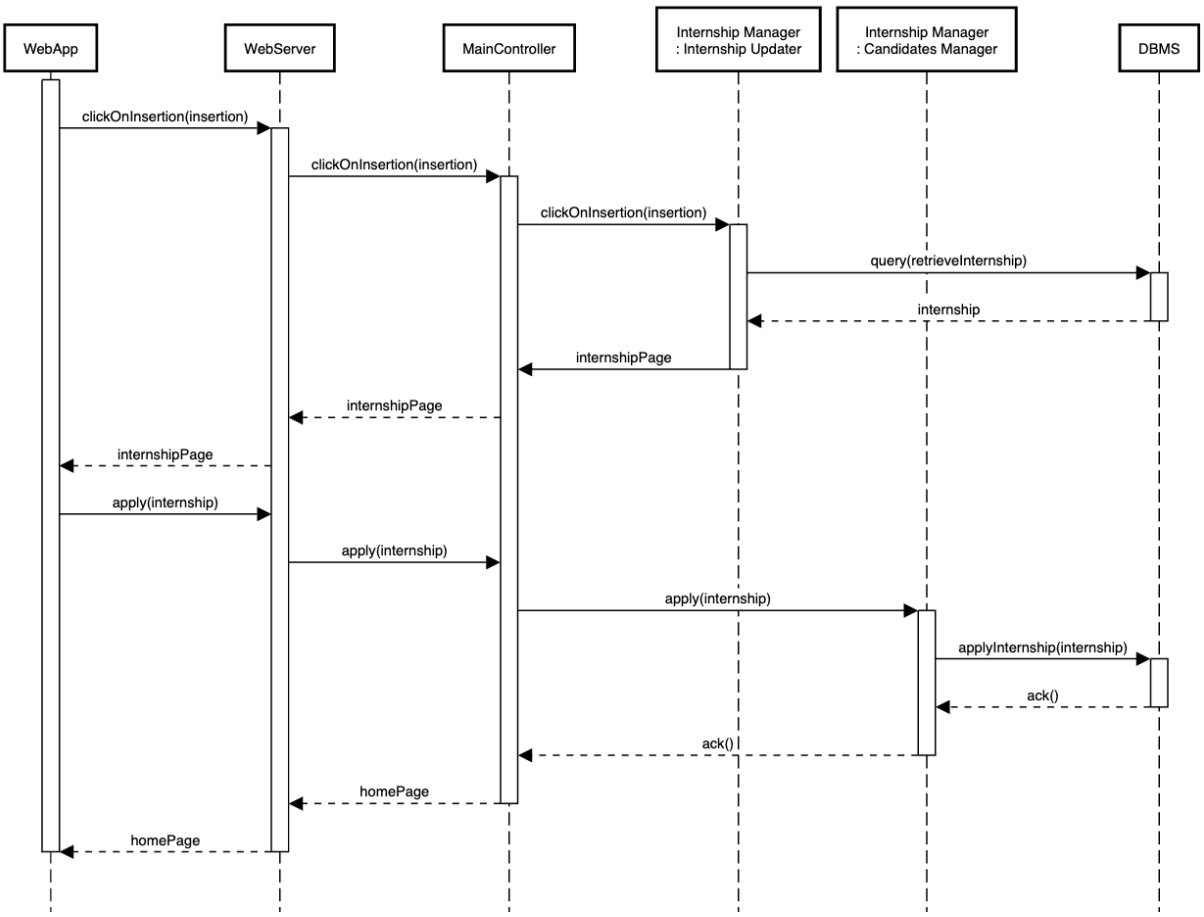

Figure 2.17: [UC7] Apply for an Internship.

The sequence diagram shows the process of a student applying for an internship through the S&C application. The student clicks on an internship insertion in the WebApp, which sends a request to the WebServer and MainController to retrieve the internship details from the database.

After reviewing the internship, the student clicks "apply," which triggers the MainController to forward the application to the Candidates Manager. The Candidates Manager updates the database, and once the application is acknowledged, the MainController sends the student back to the home page, which is displayed on the WebApp.

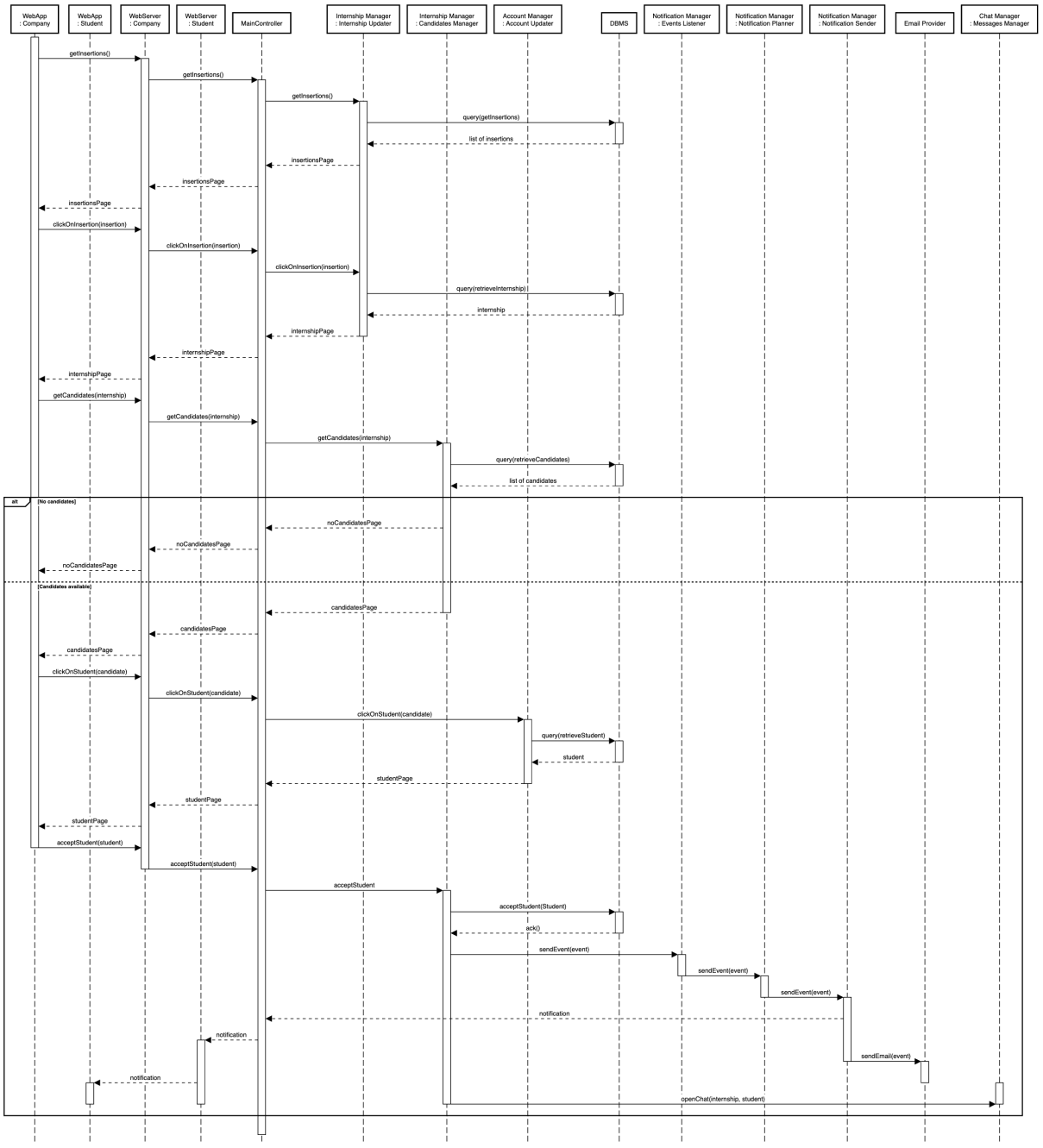

Figure 2.18: [UC8] Accept a Candidate.

The sequence diagram depicts the process of a company accepting a candidate for an internship. The company first retrieves a list of internship insertions, which are the internships they are offering. After selecting an internship, the company requests the list of candidates who have applied for it. If candidates are available, the company scrolls through the list and clicks on a candidate's profile to view their details. The system retrieves the student's information and displays it.

Once the company decides to accept the candidate, the action is processed by the Main-Controller, which updates the database and triggers notifications. The student is notified through the WebApp and via email. Additionally, a chat is opened between the company and the student for further communication.

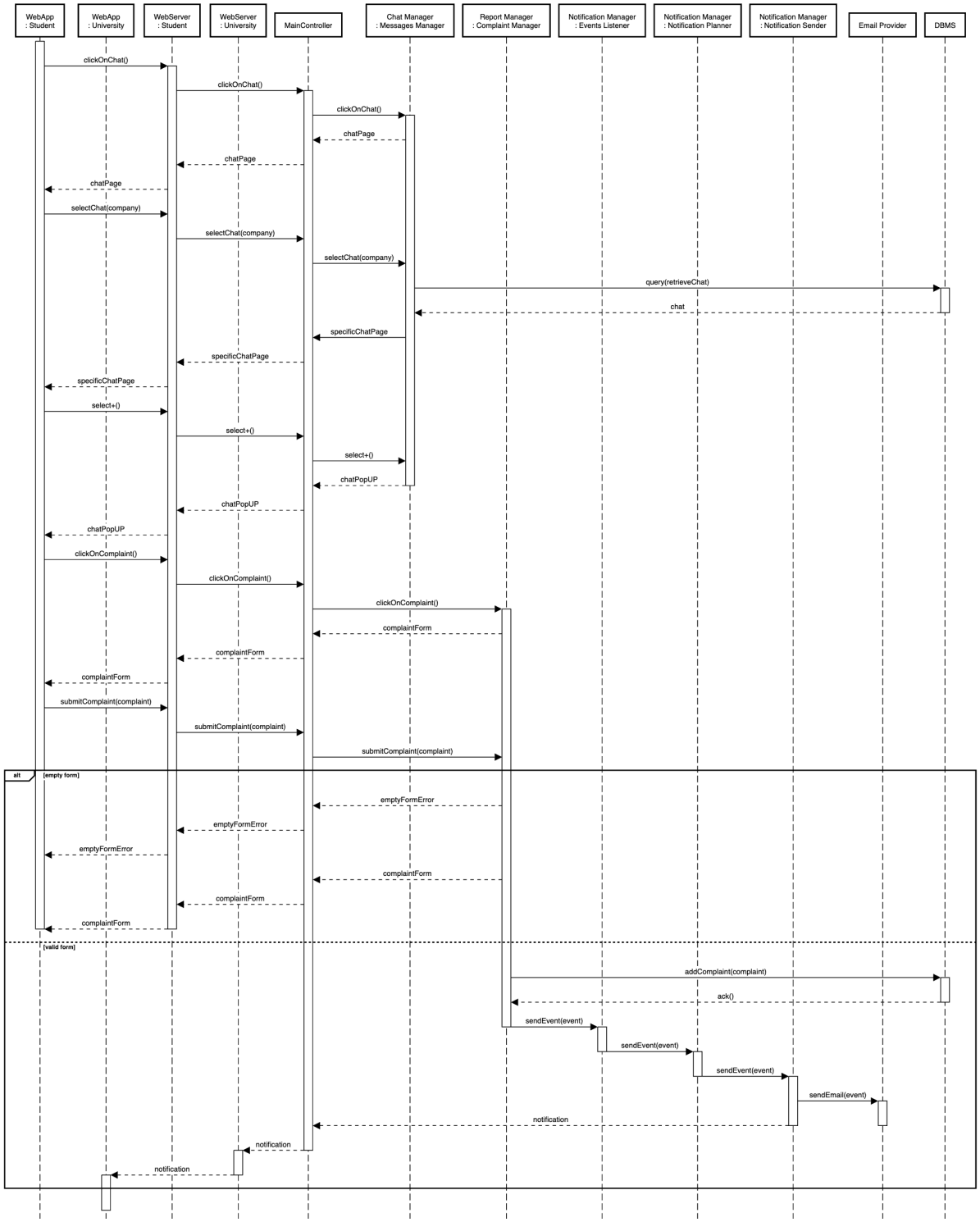

Figure 2.19: [UC9] Submit a Complaint.

The sequence diagram illustrates the process of a student submitting a complaint through the S&C application. The student first clicks on the chat logo in the WebApp, which opens the chat page. The WebServer retrieves and displays the chat page. The student then selects the specific chat with the company, and the system displays the chat details.

Next, the student clicks on the "+" logo, which opens a popup with various options, one of which is the "complaint" option. After selecting "complaint," the system presents the complaint form. The student fills it out and submits it. If the form is empty, an error is shown, prompting the student to complete it. Once the student submits a valid complaint, the Complaint Manager adds it to the database, and the system acknowledges the action.

The Complaint Manager triggers an event, which is processed by the Notification Manager. The event is passed through the Notification Planner and Notification Sender, which sends an email to the relevant parties, including the student's university. The university is notified about the complaint, completing the process for the student.

This same process can also be performed by the company. The company can submit a complaint in the same way, by selecting the chat and submitting a problem, which is processed similarly.

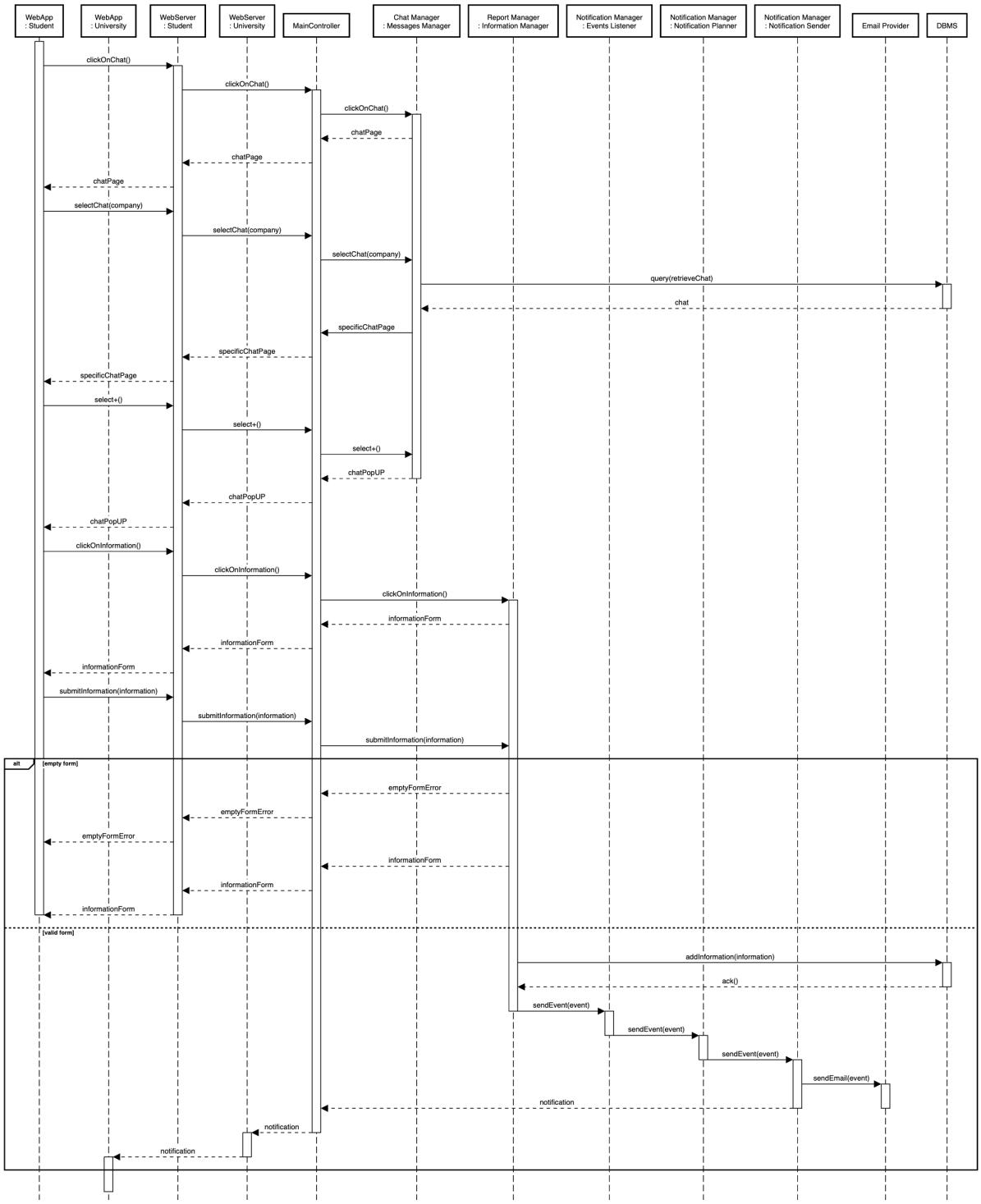

Figure 2.20: [UC10] Submit an Information.

The sequence diagram illustrates the process of a student submitting an information report through the S&C application. The student first clicks on the chat logo in the WebApp, which opens the chat page. The WebServer retrieves and displays the chat page. The student then selects the specific chat with the company, and the system displays the chat details.

Next, the student clicks on the "+" logo, which opens a popup with various options, one of which is the "information" option. After selecting "information," the system presents the information form. The student fills out the form and submits it. If the form is empty, an error is shown, prompting the student to complete it. Once the student submits a valid report, the Information Manager adds it to the database, and the system acknowledges the action.

The Information Manager triggers an event, which is processed by the Notification Manager. The event is passed through the Notification Planner and Notification Sender, which sends an email to the relevant parties, including the student's university. The university is notified about the information report, completing the process for the student.

This same process can also be performed by the company. The company can submit an information report in the same way, by selecting the chat and submitting the information, which is processed similarly.

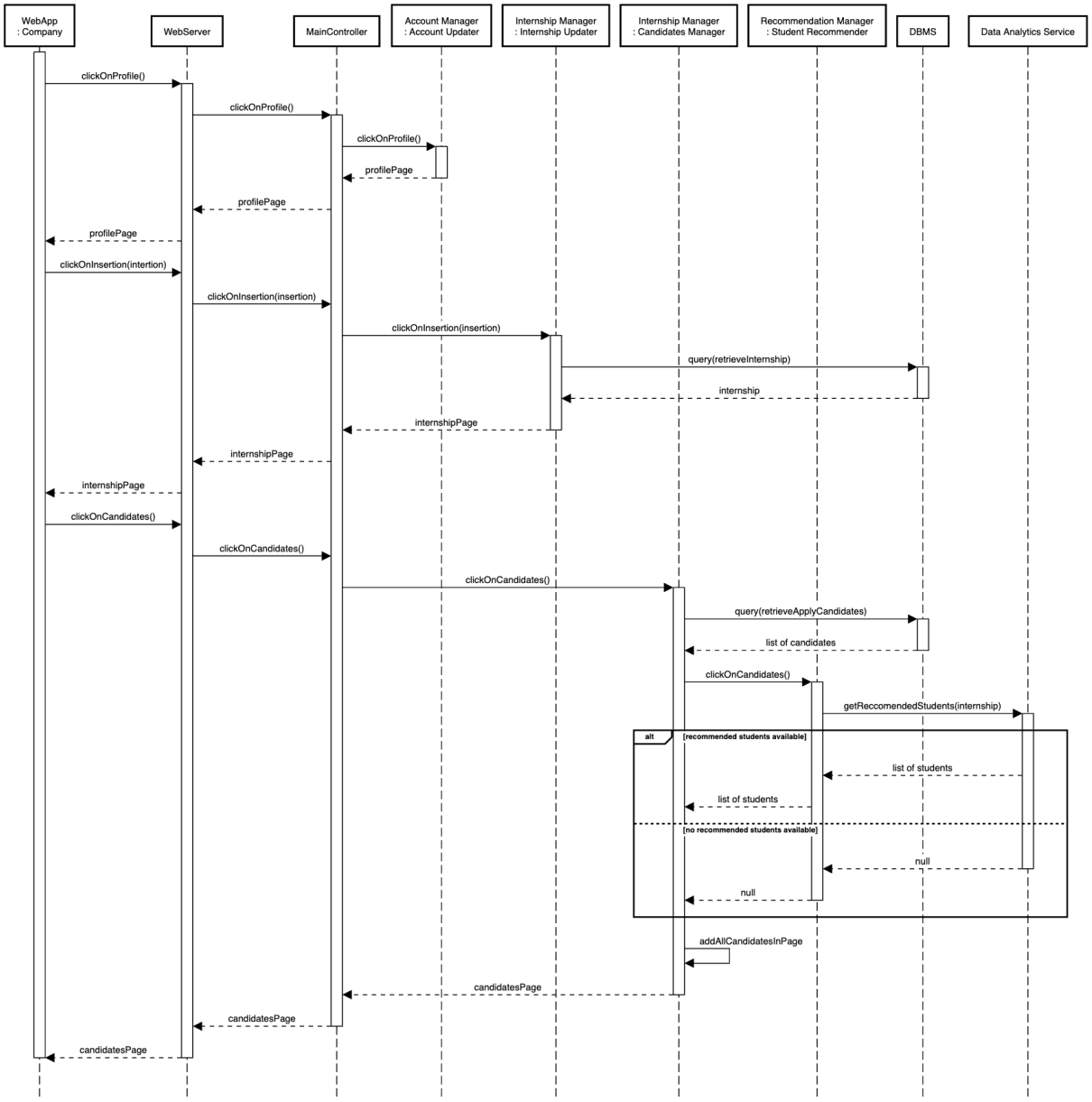

Figure 2.21: [UC11] Look for a student through recommended ones.

The sequence diagram illustrates the process of a company looking for students through recommended profiles in the application. The company begins by clicking on their profile in the WebApp, which retrieves and displays the profile page via the WebServer and MainController. The company then selects an internship insertion they are offering, and the system retrieves and displays the internship details.

Next, the company clicks on the "candidates" option for the selected internship. The system retrieves the list of students who have applied for the internship from the database through the Candidates Manager. Simultaneously, the Recommendation Manager queries the Data Analytics Service for recommended students based on the internship details. If recommended students are available, their profiles are returned and added to the candidates page. If no recommended students are found, the system proceeds with only the applied candidates.

The candidates page is divided into two sections: one section lists the students who have applied for the internship, and the other section displays the recommended students whose profiles match the internship requirements. This layout allows the company to review both groups of students conveniently. The completed candidates page is then displayed to the company, concluding the process.

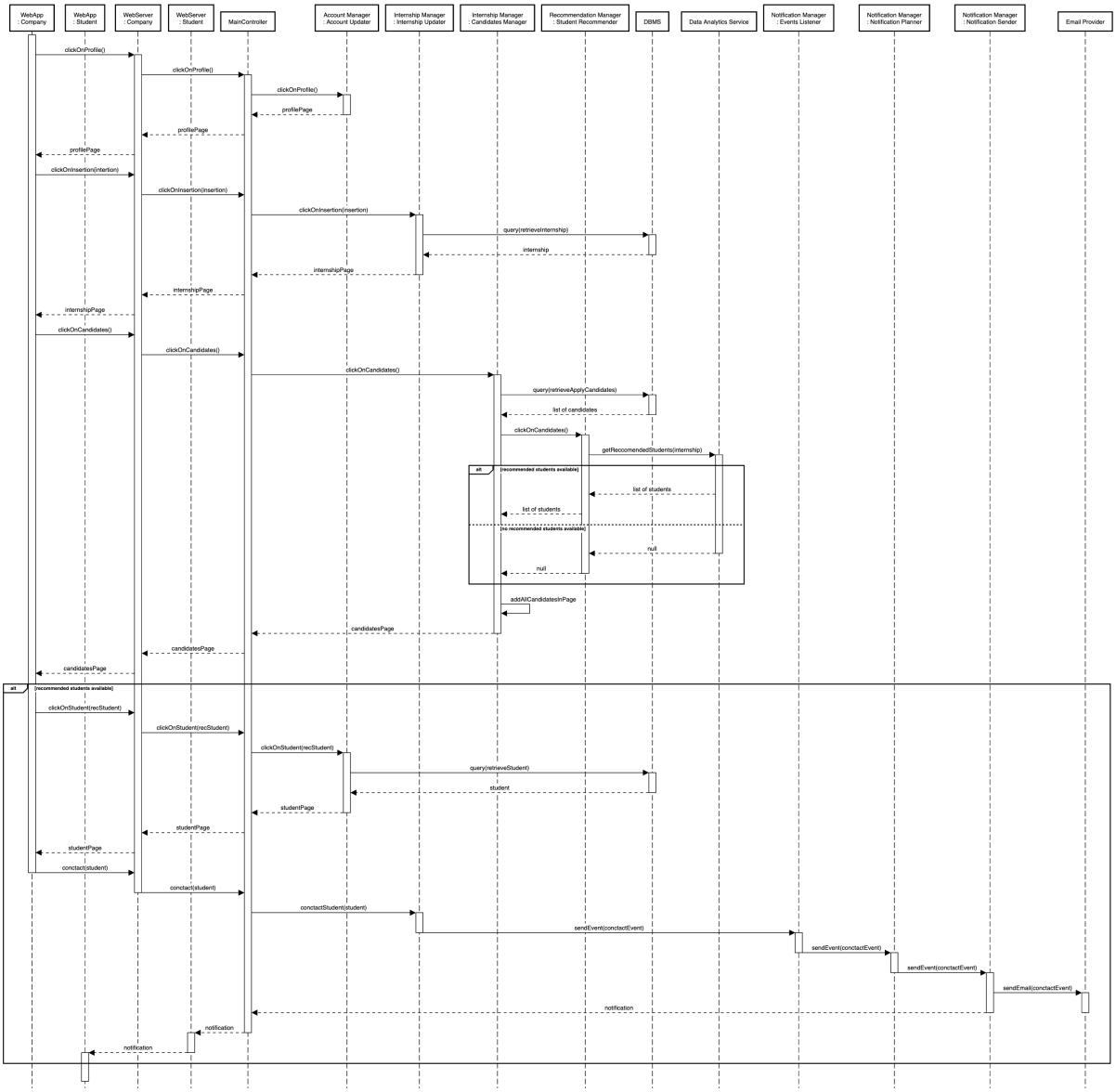

Figure 2.22: [UC12] Contact a Recommended Student.

The sequence diagram illustrates the process of a company contacting a recommended student through the S&C application. The company begins by clicking on their profile in the WebApp, which retrieves and displays the profile page via the WebServer and MainController. The company then selects an internship insertion they are offering, and the system retrieves and displays the internship details.

Next, the company clicks on the "candidates" option for the selected internship. The system retrieves the list of applied candidates from the database and queries the Data Analytics Service for recommended students through the Recommendation Manager. If recommended students are available, their profiles are added to the candidates page, which is divided into two sections: one for applied students and one for recommended students.

The company selects a recommended student, and the system retrieves the student's profile from the database. The profile is displayed to the company, who then initiates contact with the student. The MainController processes the contact request, triggering the Internship Updater to send a contact event to the Notification Manager. Notifications are sent via the Notification Planner and Notification Sender, and an email is sent to the student to inform them of the contact request. The WebApp also notifies the student, completing the process.

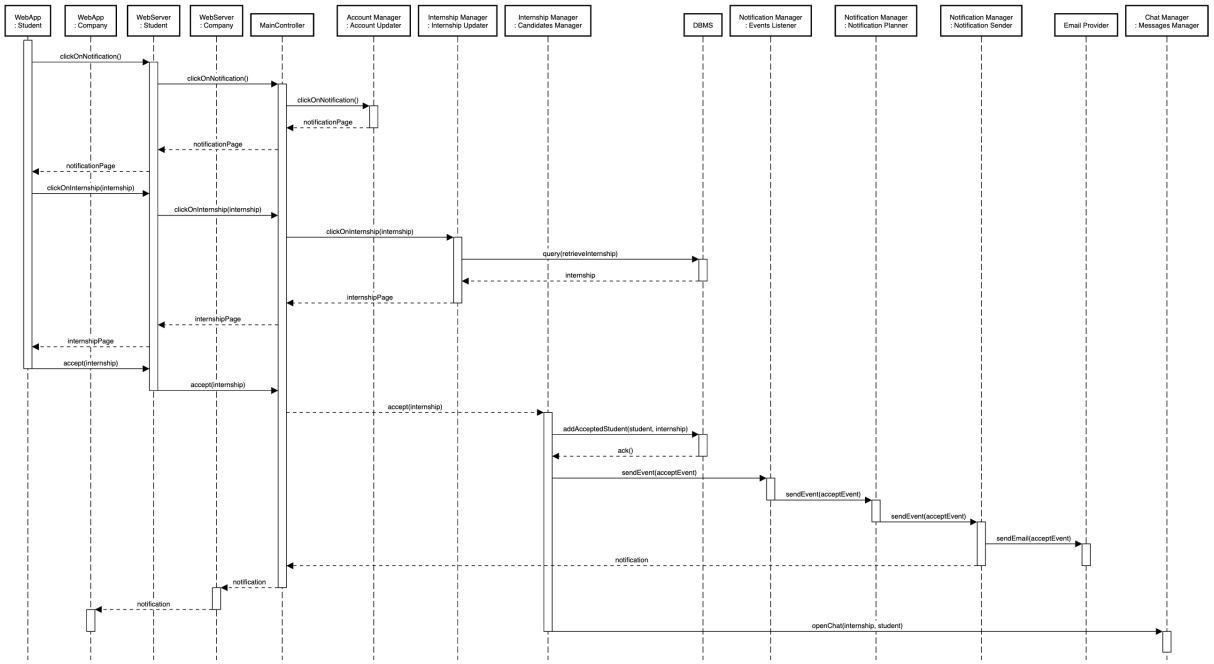

Figure 2.23: [UC13] Accept a Company.

The sequence diagram illustrates the process of a student accepting a company for an internship through the S&C application. The student starts by clicking on a notification in the WebApp, which retrieves and displays the notification page via the WebServer and MainController. The student selects the relevant internship from the notification page, and the system retrieves and displays the internship details.

The student then clicks "accept" for the internship. The WebServer sends the request to the MainController, which processes it through the Candidates Manager. The Candidates Manager updates the database to mark the student as accepted for the internship and triggers an event through the Notification Manager. Notifications are sent via the Notification Planner and Notification Sender, and an email is sent to the company to inform them of the student's acceptance. Additionally, a chat is opened between the student and the company for further communication, completing the process.

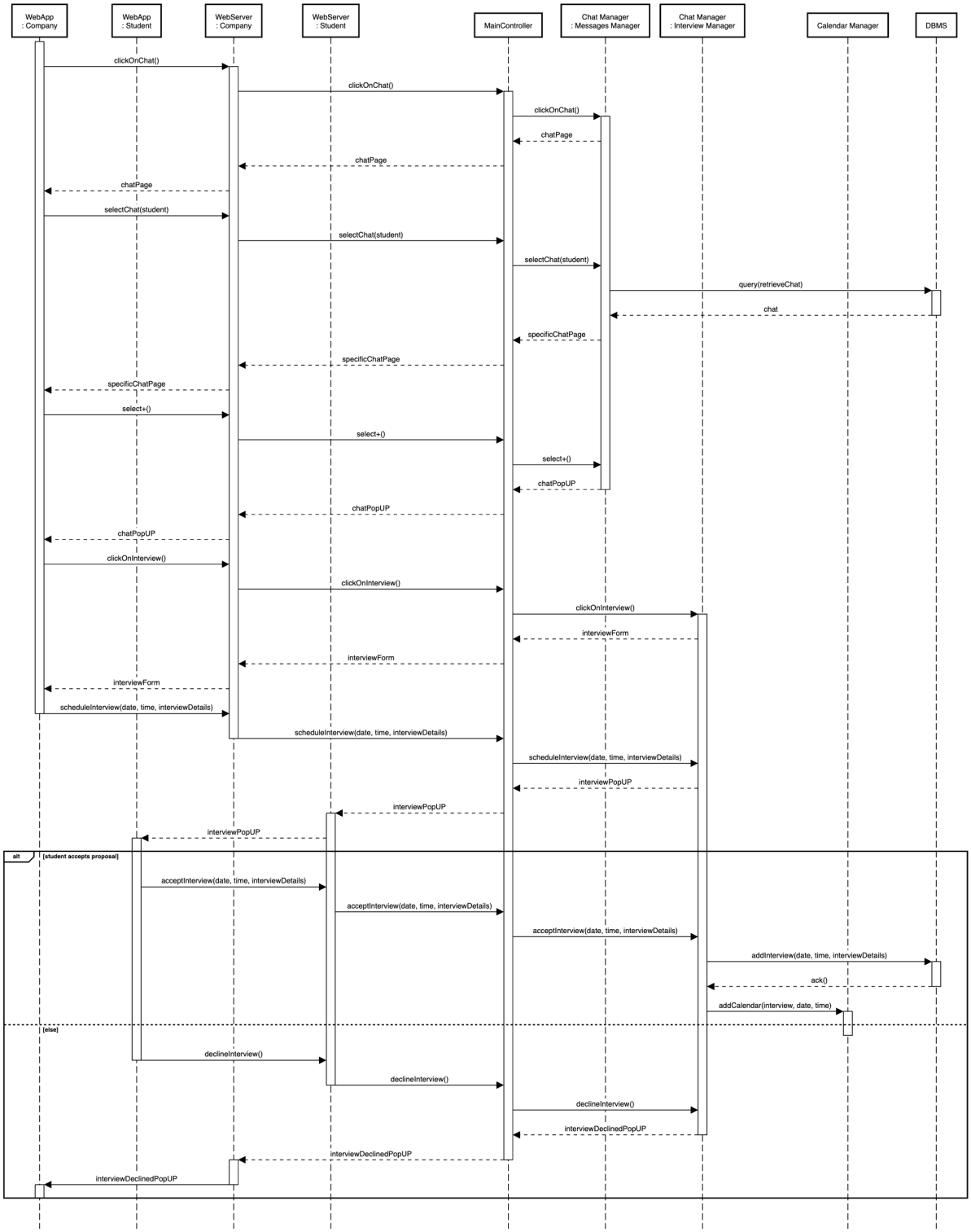

Figure 2.24: [UC14] Schedule an Interview.

a student through the S&C application. The company begins by clicking on the chat logo in the WebApp, which opens the chat page. The WebServer retrieves and displays the chat page. The company selects a specific chat with a student, and the system retrieves and displays the chat details.

Next, the company clicks on the "+" logo to open a popup with options and selects "interview." The system presents an interview scheduling form, where the company specifies the date, time, and details of the interview. The scheduling request is processed by the MainController and the Interview Manager, which prepares a notification for the student.

The student receives a popup notification about the interview proposal. If the student accepts the proposal, the Interview Manager adds the interview details to the database and updates the calendar through the Calendar Manager, adding the event. If the student declines, the system informs the company with a "declined interview" notification, completing the process.

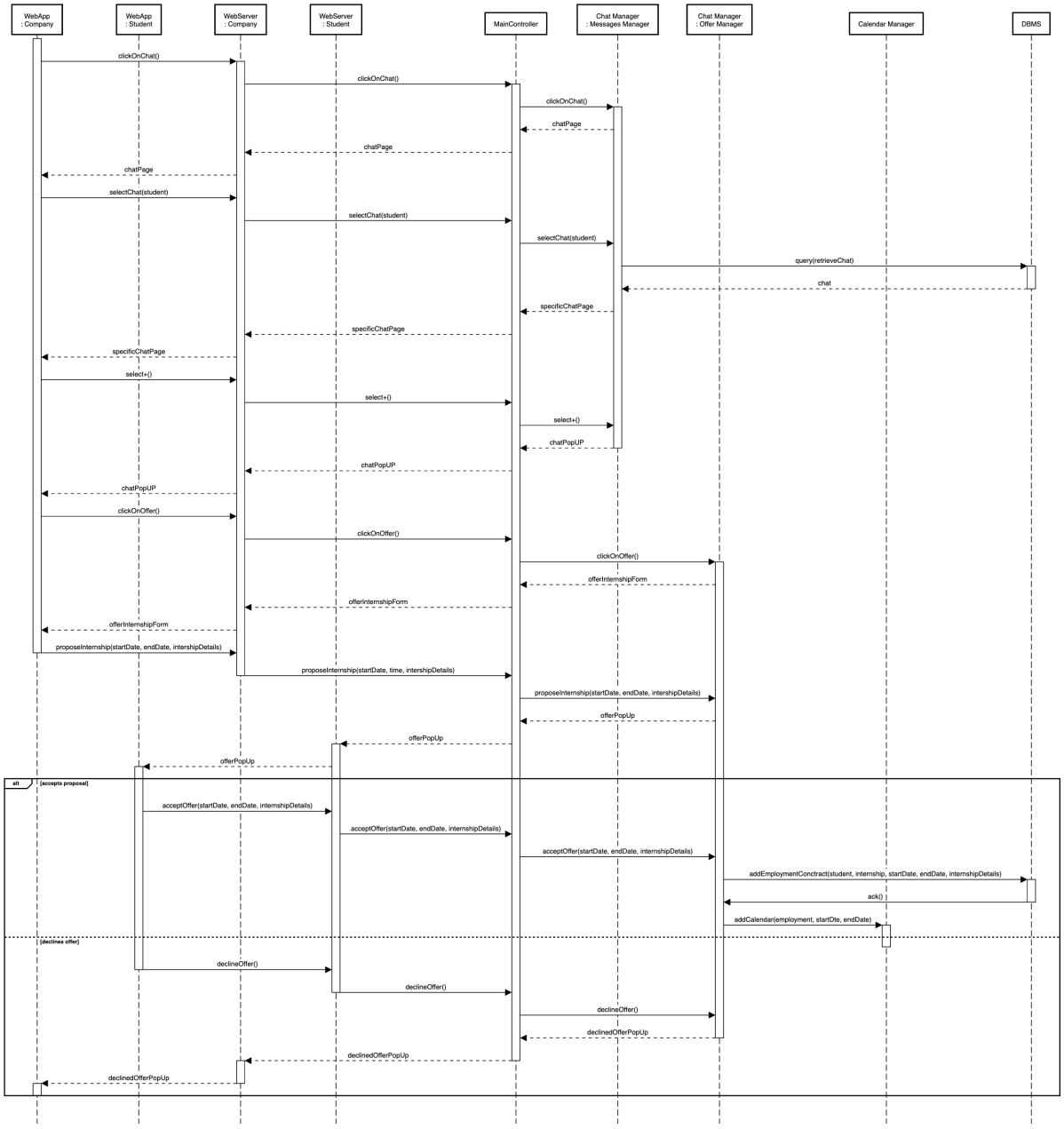

Figure 2.25: [UC15] Start an Internship.

The sequence diagram illustrates the process of a company proposing an internship offer to a student through the S&C application. The company starts by clicking on the chat logo in the WebApp, which opens the chat page. The WebServer retrieves and displays the chat page. The company selects a specific chat with a student, and the system retrieves and displays the chat details.

Next, the company clicks on the "+" logo to open a popup with options and selects "offer." The system presents an internship offer form, where the company specifies the start date, end date, and other internship details such as the workplace, remuneration, tasks, and other relevant information. Once the company submits the offer, the MainController processes the request through the Offer Manager, which sends a notification to the student.

The student receives a popup notification about the internship offer. If the student accepts, the Offer Manager updates the database by adding an employment contract with the provided details and updates the calendar with the internship dates. The event is added to both the student's and the company's calendars. If the student declines the offer, the system notifies the company with a "declined offer" popup, completing the process.

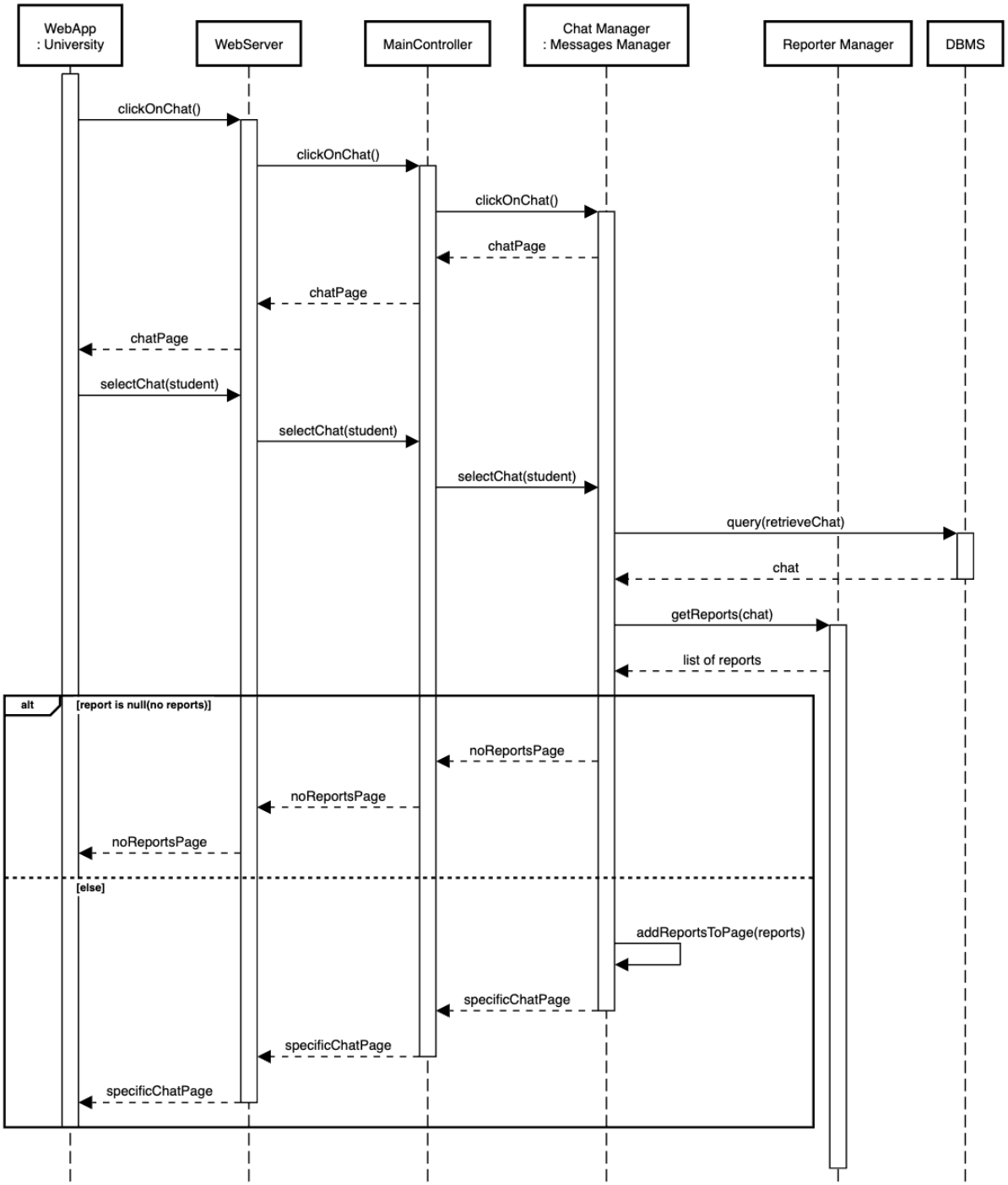

Figure 2.26: [UC16] Monitor internship.

The Sequence diagram illustrates the process of a university monitoring the internship of one of its enrolled students through the S&C application. The university begins by clicking on the chat logo in the WebApp, which opens the chat page. The WebServer retrieves and displays the chat page.

The university selects a specific chat with a student, and the system retrieves the chat details from the database. Additionally, the system queries the Reporter Manager to fetch any associated reports (information or complaints) related to the selected chat.

If no reports are available, the system displays a "no reports" page to the university. If reports are found, they are added to the chat page, and the updated chat page with the reports is displayed. This ensures that the university can monitor both the interactions and any reports submitted by the student or the company. This process keeps the university informed about the internship's status and any potential issues requiring intervention.

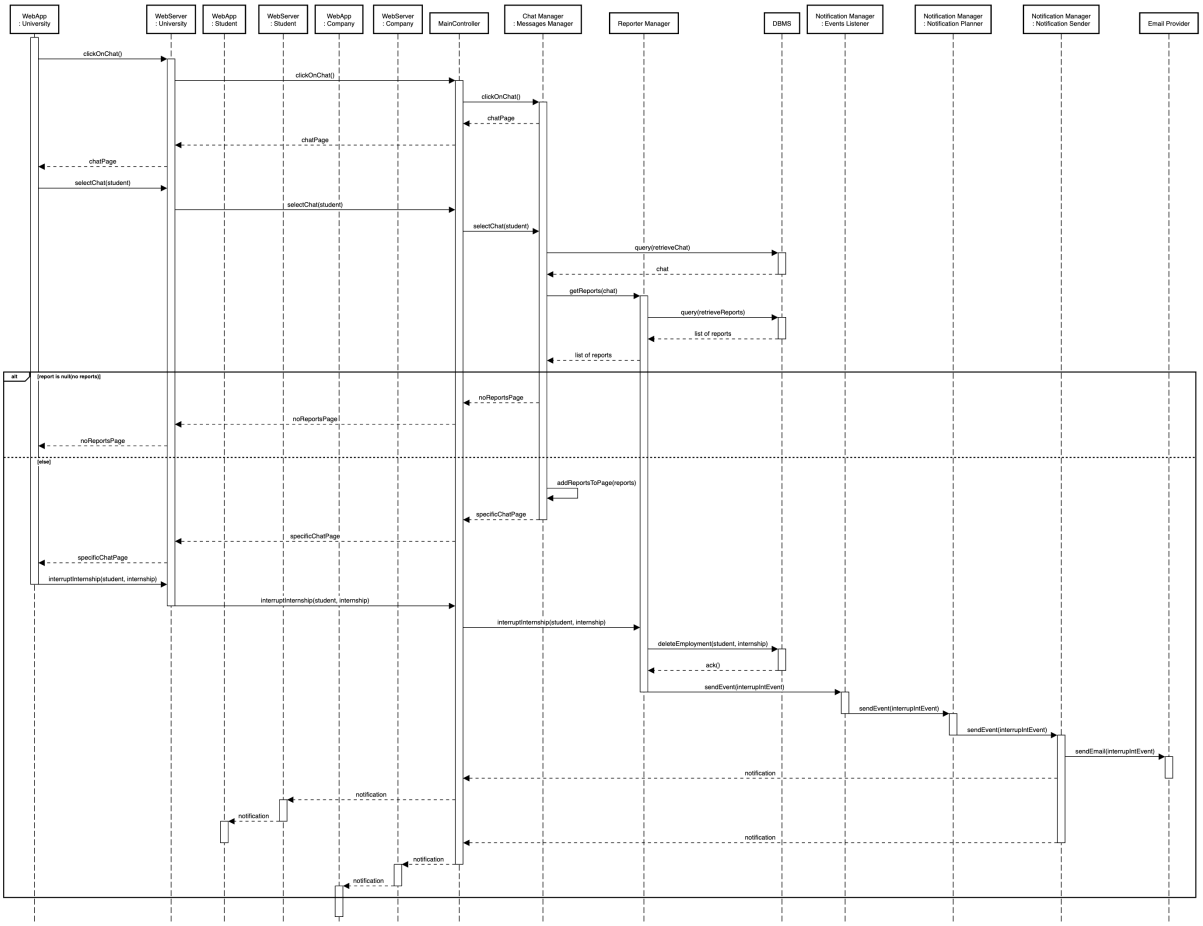

Figure 2.27: [UC17] Interrupt internship.

The sequence diagram illustrates the process of a university interrupting a student's internship through the S&C application when complaints are present and deemed necessary. The university begins by clicking on the chat logo in the WebApp, which opens the chat page. The WebServer retrieves and displays the chat page.

The university selects a specific chat with a student, and the system retrieves the chat details from the database. It also queries the Reporter Manager to fetch any associated reports (information or complaints) related to the internship. If no reports are found, the system displays a "no reports" page. If reports are available, they are added to the chat page, and the updated page is displayed to the university.

If the university determines that the complaints warrant intervention, it selects the "interrupt internship" option. The WebServer sends the request to the MainController, which forwards it to the Reporter Manager. The Reporter Manager deletes the employment record from the database, effectively terminating the internship.

The Reporter Manager then triggers an event through the Notification Manager. Notifications are processed and sent via the Notification Planner and Notification Sender. An email is sent to all relevant parties, and notifications are displayed to both the student and the company through their respective WebApps, informing them of the internship's termination. This ensures that all stakeholders are promptly informed of the decision.

2.6. Selected Architectural Styles and Patterns

The architectural design of the S&C system leverages a 4-tier architecture, combining well-established architectural styles and design patterns to ensure scalability, maintainability, and ease of integration.

The system separates concerns across distinct layers:

- **Presentation Layer:** Provides the user interface through a web application. This layer focuses on user interaction and performs basic checks, such as form validation, delegating all business logic to the application server.
- **Web Layer:** Acts as an intermediary, forwarding requests between the presentation layer and the application server while ensuring secure communication.
- **Application Layer:** Implements all the business logic of the system. This layer consists of modular components, each responsible for specific functionalities.
- **Data Layer:** Manages the storage and retrieval of data, ensuring consistency and persistence.

The separation of these layers ensures that the system is modular and can be scaled or modified without significant changes to other tiers. The separation between the Web Server and the Application Server was primarily motivated by the enhanced security this approach typically provides. Additionally, this decision positively impacts the system's development process by enabling parallel work across different tiers, allowing multiple teams to collaborate efficiently and expedite implementation.

2.6.1. Model-View-Controller (MVC)

The **MVC pattern** is applied to the presentation and application server layers, enabling a clear separation between the user interface (View), user interactions (Controller), and business logic (Model). This structure simplifies the management of the web application's user interface, allowing the frontend and backend to be developed independently. By isolating business logic from the presentation layer, the pattern ensures that updates to the user interface can be made without impacting the underlying logic, promoting maintainability and scalability.

2.6.2. Centralized Access

The Main Controller simplifies interactions by acting as a **facade** between the Web Server and backend components. This centralization reduces complexity, ensuring that the Web Server interacts with a unified interface rather than managing connections to multiple components. By consolidating these interactions, the system achieves greater modularity and ease of maintenance. Additionally, the Main Controller enables the implementation of shared logic, such as authentication, validation and error handling, in a single location. This approach avoids duplication across components and ensures consistency in enforcing security policies and logging activities.

This design also enhances scalability and debugging. With all requests passing through the Main Controller, load management can be centralized, and system behavior can be easily monitored. This approach reduces coupling between tiers, supporting the system's ability to adapt to future changes or expansions.

2.6.3. Observer Pattern

The **Notification Manager** leverages this pattern by subscribing to specific events, such as registration completion or new internship postings, and triggering notifications in response. By decoupling event handling from the core logic of other components, this approach ensures that components remain loosely coupled, reducing dependencies on the notification system. This design enhances modularity and extensibility, allowing new event types or notification methods to be added without impacting the rest of the system.

2.6.4. Repository Pattern

The **repository pattern** is used to abstract database operations, enabling the application server to interact with data through well-defined, high-level interfaces. This abstraction ensures consistency in how data is accessed and manipulated, while also reducing redundancy in database queries. By centralizing database interactions, this pattern improves maintainability, providing a single source of truth and making it easier to adapt to changes in the underlying data structures or queries.

2.7. Other design decisions

Beyond the core architectural styles and patterns, several additional design decisions were made to address specific challenges and enhance system functionality:

2.7.1. Event-Driven Asynchronous Processing

To maintain responsiveness and scalability, **asynchronous processing** was adopted for tasks like sending notifications. This ensures that user interactions remain smooth and uninterrupted, while heavy operations are handled independently.

2.7.2. Security Enhancements

The system enforces **HTTPS** for all communication between tiers to safeguard data in transit. Additionally, encryption is applied to sensitive data stored in the database, and Role-Based Access Control (RBAC) restricts user permissions based on their roles (e.g., student, company, or administrator).

2.7.3. Database Query Optimization

Frequent **queries**, such as internship searches or user profile retrievals, were optimized through indexing and caching mechanisms. These optimizations ensure high performance, particularly as the system scales to accommodate more users.

2.7.4. Integration Flexibility

External integrations, such as AI tools for CV enhancement and analytics for recommendations, were designed to be **modular**. This allows new services to be incorporated with minimal changes to the system, supporting adaptability to future requirements.

2.7.5. Error Logging and Monitoring

A unified **error-handling** and **logging framework** was implemented across all tiers. This approach facilitates debugging and ensures system administrators can monitor performance and address issues promptly.

3 | User Interface Design

This chapter presents the user interfaces of our application, showcasing key designs related to meaningful use cases. Each interface is tailored to specific functionalities and workflows, reflecting how users interact with the system to accomplish their goals. By illustrating these interfaces, we aim to provide a clear understanding of the application's usability, layout, and design principles.

In the RASD, interfaces for login and the main pages of the application were introduced. In this document, we extend this by presenting additional interfaces associated with other important use cases, further demonstrating how the system supports user needs in a seamless and intuitive manner.

3.1. Sign Up [UC1]

The first interface is tailored for students, requiring inputs for email, password, first name, last name, and username. The "Accept & Join" button finalizes the sign-up.

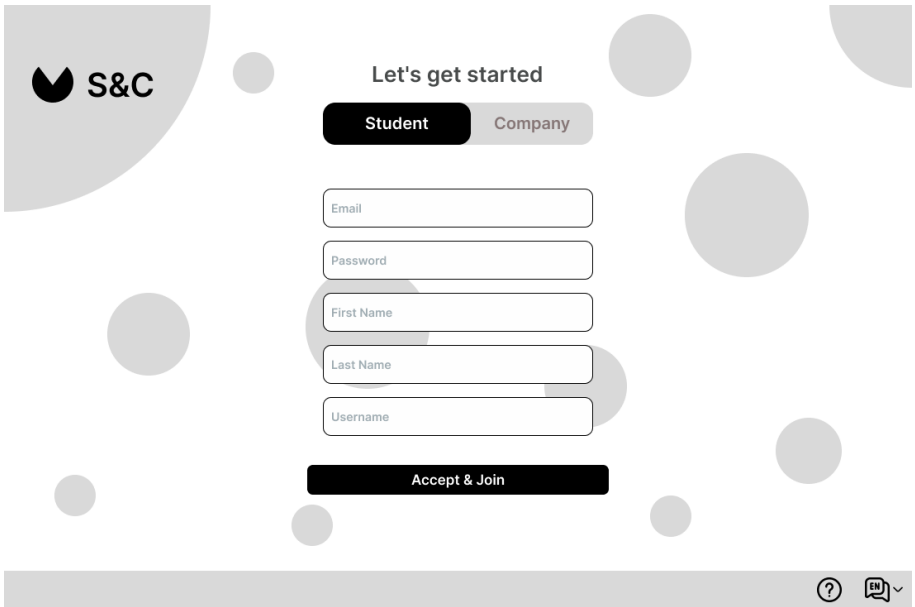

Figure 3.1: Sign Up Student.

The second interface allows company representatives to create accounts by filling out fields for email, password, name, VAT name, and username. The "Accept & Join" button completes the registration process.

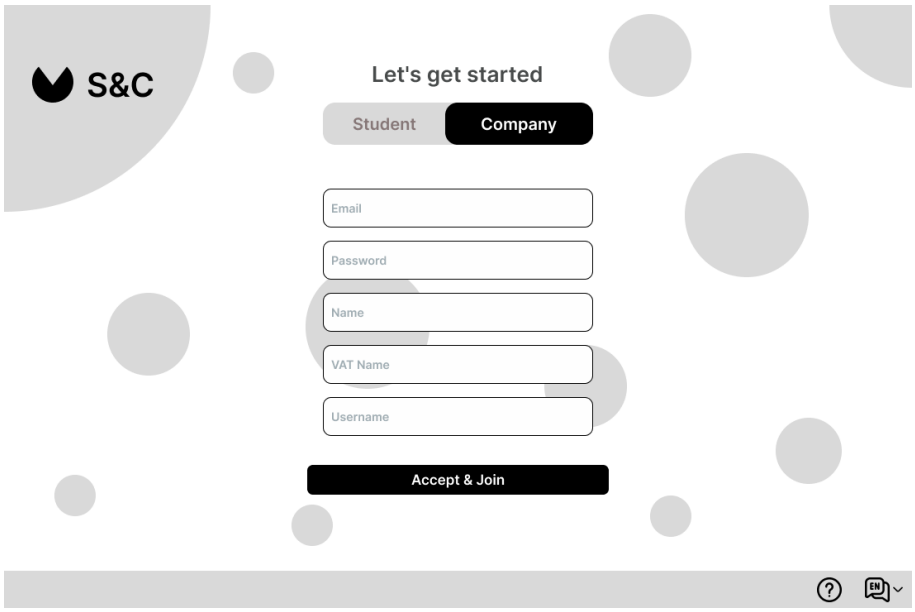

Figure 3.2: Sign Up Company.

A toggle at the top lets users easily switch between the Student and Company sign-up forms.

3.2. Insert an enhanced Internship Insertion [UC3]

Figure 3.3: Initial Postage

The first phase represents the initial "Post a New Internship Insertion" form. This interface is displayed when the user clicks the "Post" button on the homepage. The form contains fields for the internship's position name, location, duration, start date, and description, allowing the user to provide all necessary details.

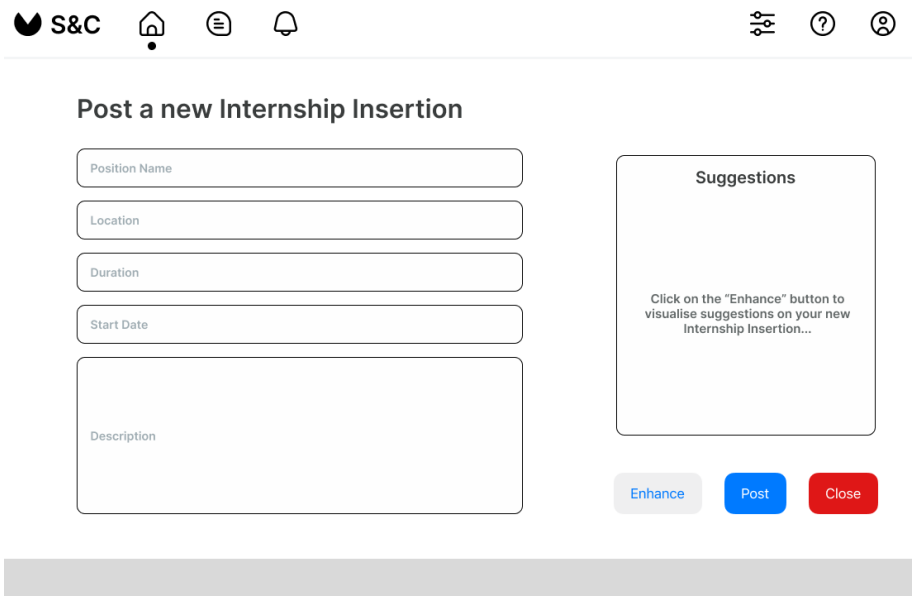

Figure 3.3: Post a New Internship.

Figure 3.4: Filled Form

In the second phase, the form is shown with all fields filled in by the user. The system is now ready to process or enhance the insertion based on the provided information.

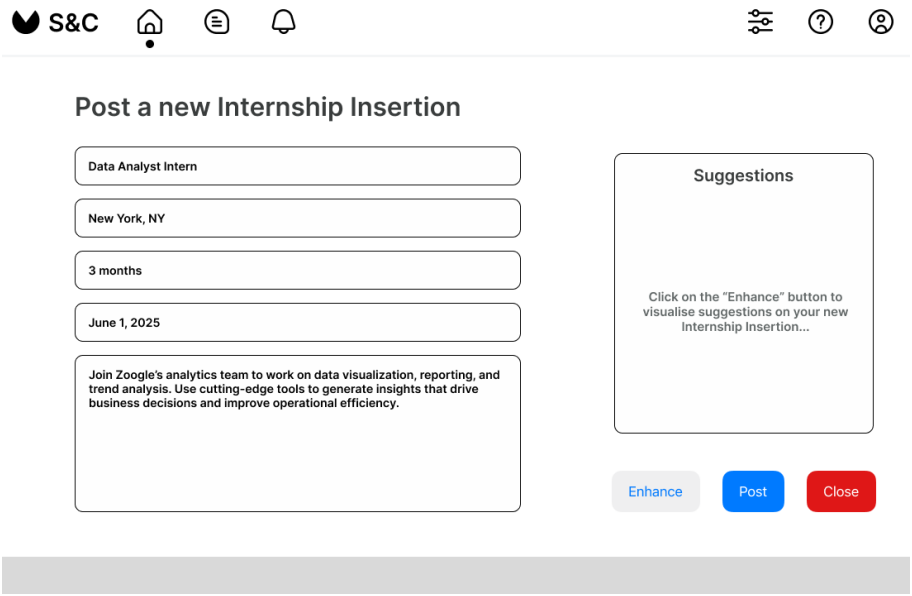

Figure 3.4: Post a New Internship (filled).

Figure 3.5: Enhanced Suggestions

The final phase is displayed after the user clicks the "Enhance" button. The system processes the input data and provides suggestions on the right panel to improve the internship insertion. These suggestions may add more specific details to the description to make the posting more appealing and clear.

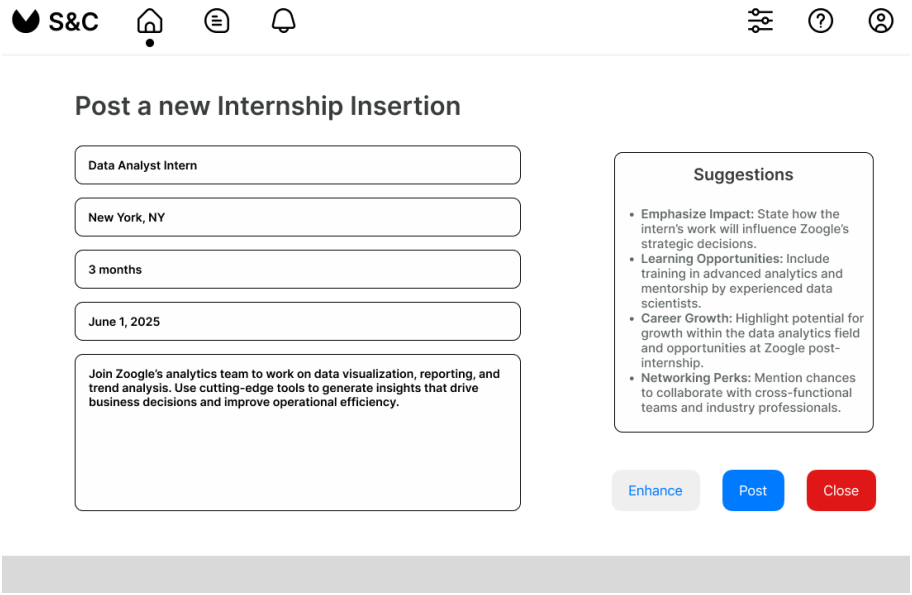

Figure 3.5: Suggestions on the Internship Post.

3.3. Enhance CV [UC4]

Figure 3.6: Viewing the Profile

The first figure represents the profile page of a student within the application. This page displays the student's personal information, such as:

Name, Date of Birth, City, University, Mobile Number, Email and Password, and a section for the uploaded CV file. The student has the option to upload or enhance his CV using the respective buttons.

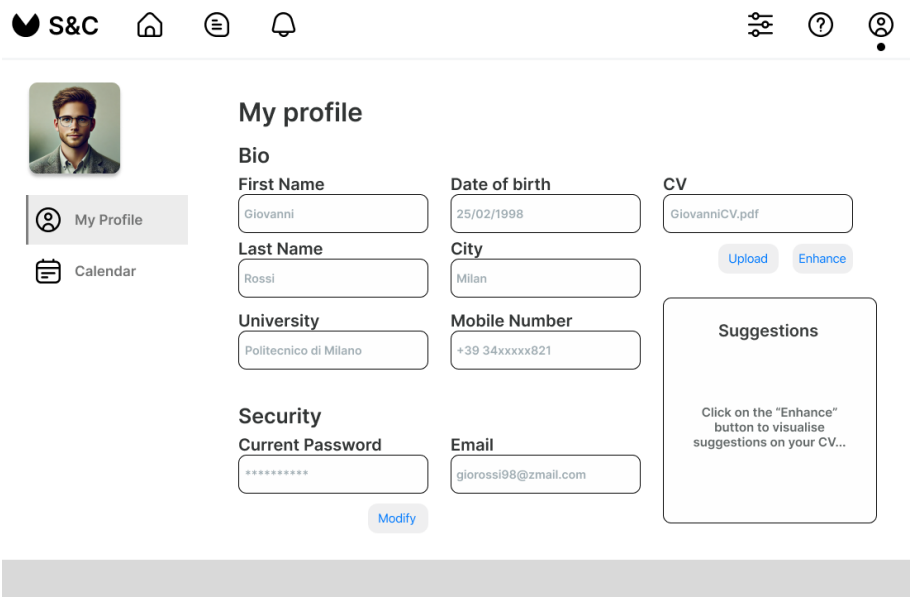

Figure 3.6: MyProfile page.

Figure 3.7: Enhancing the CV

The second figure is the result of clicking the Enhance button. The system processes the uploaded CV and provides tailored suggestions in the "Suggestions" section to improve the CV. Examples include adding specific skills, quantifying achievements, or including relevant certifications.

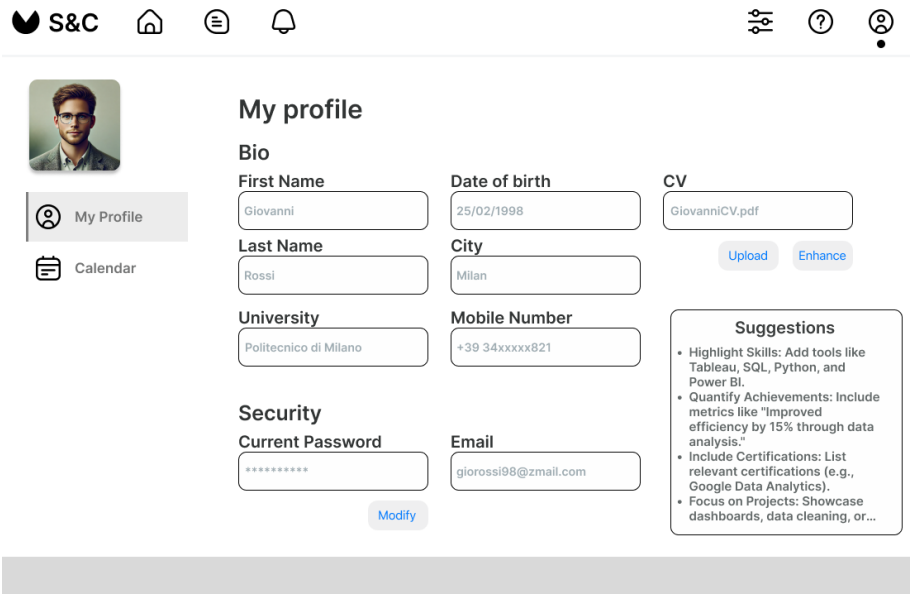

Figure 3.7: MyProfile page (enhance).

Modifying the CV

If the student decides to act on the suggestions, they need to modify the CV file externally using their preferred editing tools. Once updated, the student can re-upload the revised CV using the Upload button on the same profile page.

This flow ensures the CV remains user-controlled while benefiting from AI-driven enhancement suggestions provided by the system.

3.4. Submit a Feedback [UC5]

Figure 3.8: Viewing a Recommendation

The first image represents the system's recommendation for an internship insertion, tailored based on the analysis of the user. It includes key details: Position Title, Location, Duration, Start Date and a Description.

The user can click the "Apply" button to proceed (as explained on the [UC7]) or close the recommendation by clicking the exit button (X) in the top-right corner.

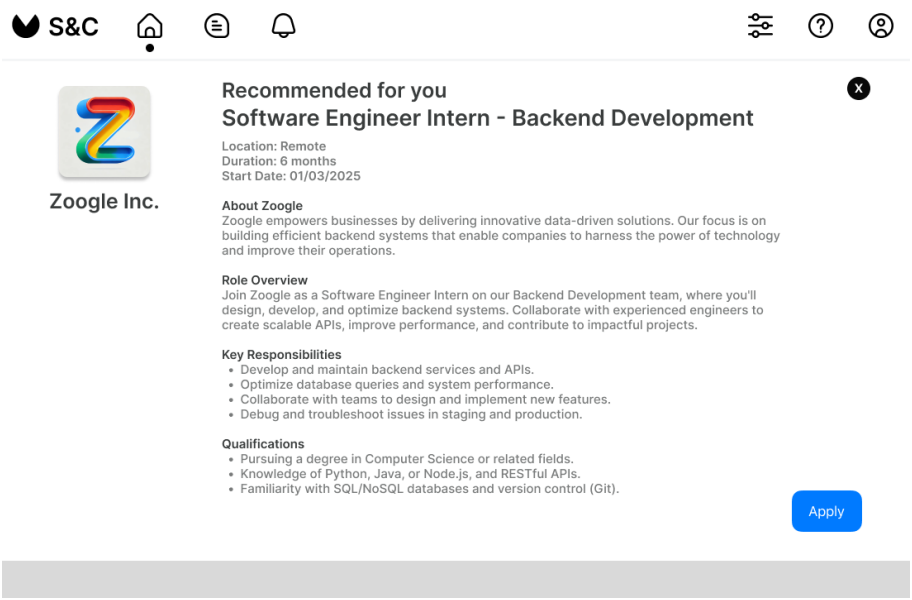

Figure 3.8: Recommended Insertion.

Figure 3.9: Providing Feedback

The second image is displayed when the user clicks the exit button (X) to close the recommendation. A pop-up asks the user to submit feedback on the recommendation, including:

- Rating: A 5-star rating system to evaluate the relevance or quality of the recommendation.
- Comments: A text box where the user can provide additional feedback, such as what they liked or what could be improved.

The "Submit" button sends the feedback to the system, helping refine future recommendations.

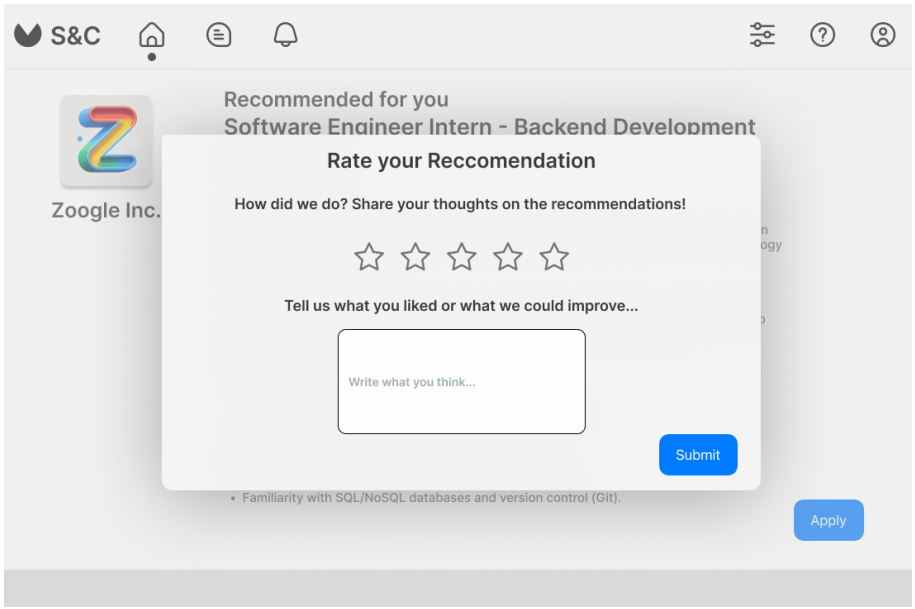

Figure 3.9: Ask for a Feedback.

3.5. Accept a Candidate [UC8]

Figure 3.10: View Internship Insertions

The first image represents the list of all internship insertions posted by the company. Each insertion is displayed with its title and location, allowing the company to select the relevant internship they wish to manage.

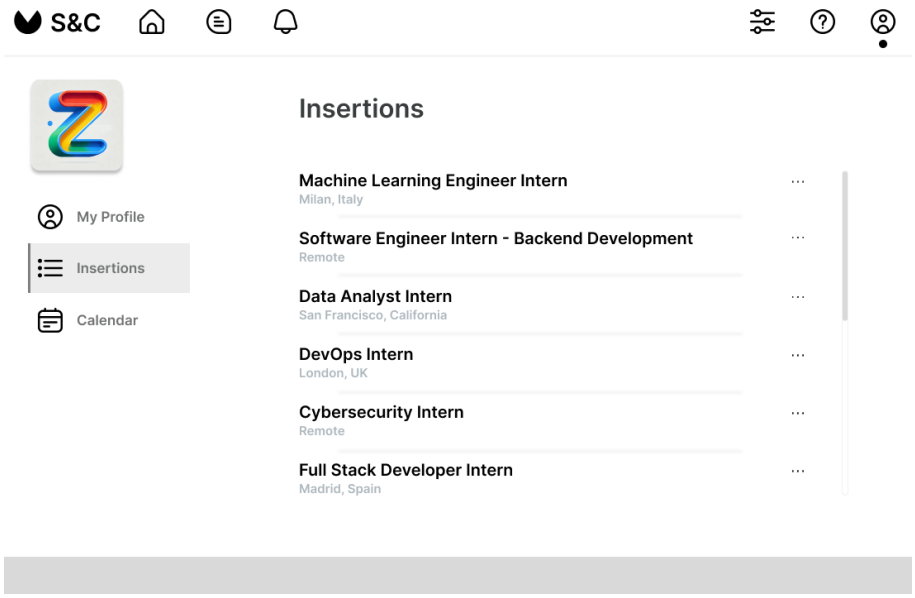

Figure 3.10: Insertions posted.

Figure 3.11: Open an Insertion

In the second image, a specific insertion is opened (e.g., "Software Engineer Intern - Backend Development"). The details of the insertion are displayed and three buttons are available:

- Candidates: To view the list of applicants and system-recommended candidates.
- Enhance: To optimize the insertion further.
- Delete: To remove the insertion.

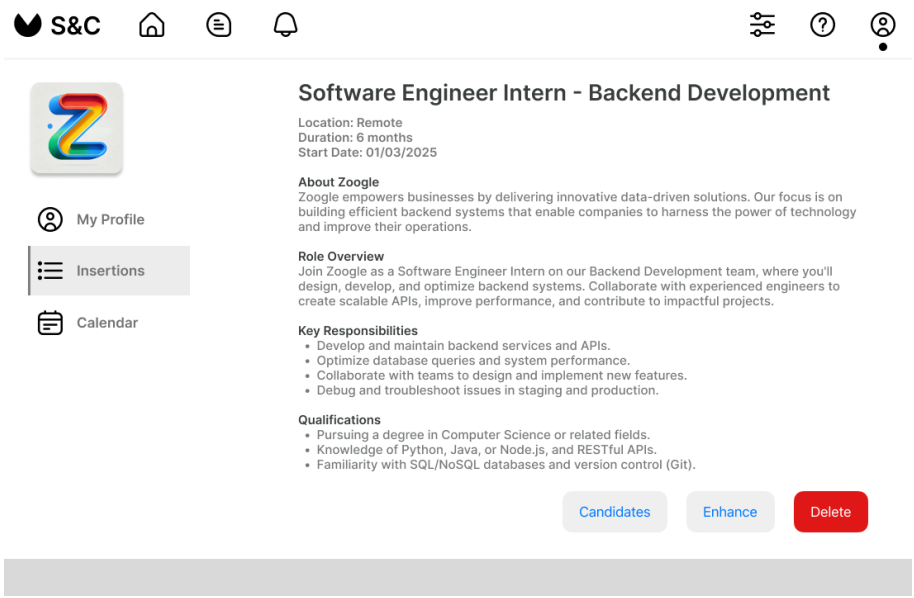

Figure 3.11: Insertion post example.

Figure 3.12: View Candidates

The third image shows the candidates associated with the selected insertion. The list includes:

- Applied Candidates: Users who directly applied for the position.
- Recommended Candidates: Profiles suggested by the system based on a match with the insertion requirements.

Users can search candidates by name, and tabs allow switching between "Applied" and "Recommended."

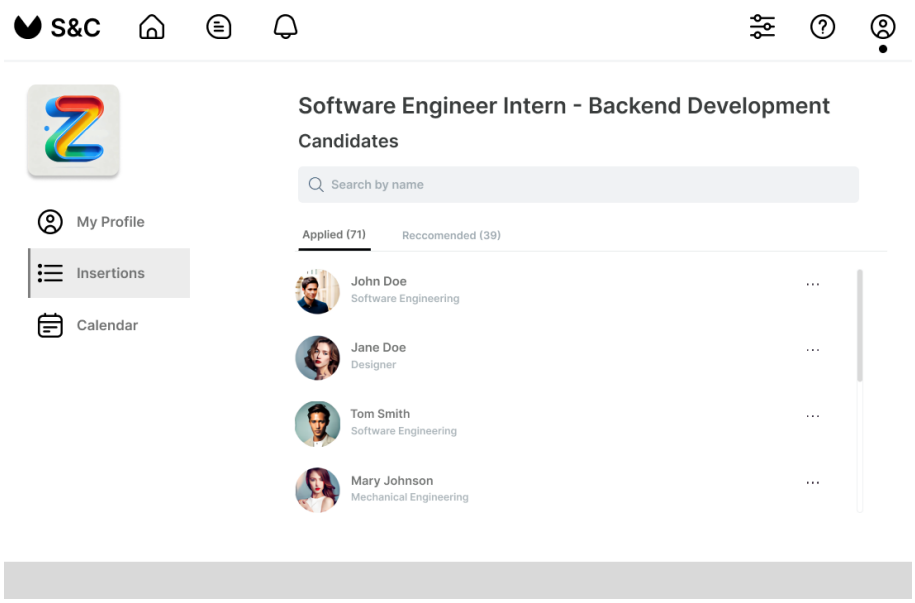

Figure 3.12: Insertion's candidates section.

Figure 3.13: Review and Accept a Candidate

In the last image, a specific candidate's profile is opened (e.g., "John Doe"). The pop-up provides:

- Personal Information: City, date of birth, phone number.
- Resume: A downloadable CV file.

And also two actions:

- Accept: Select the candidate and start a chat to proceed with further steps.
- Close: Exit the pop-up without taking action.

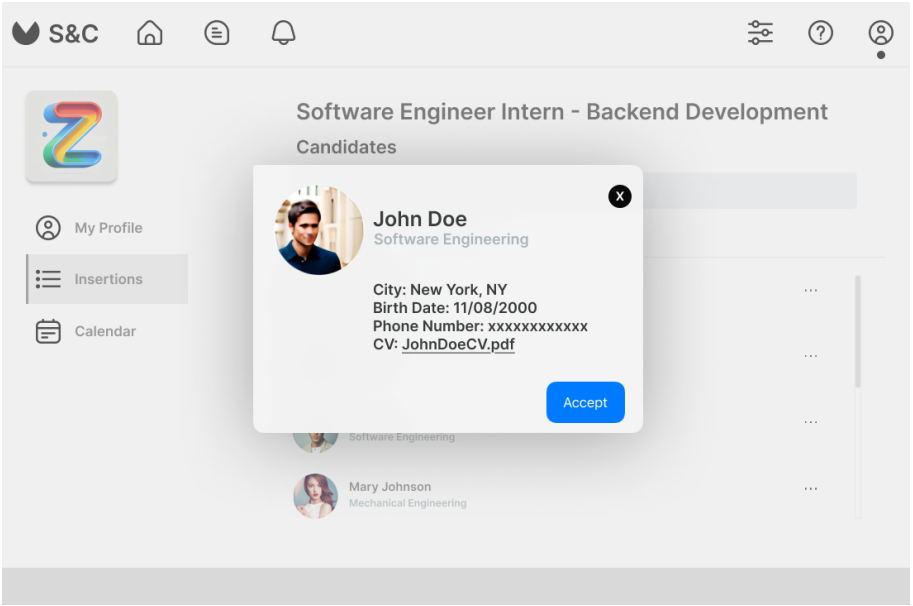

Figure 3.13: Review of a candidate.

3.6. Schedule an Interview [UC14] and Start an Internship [UC15]

Figure 3.14: Interview Proposal The first image represents a proposal for scheduling an interview. The company sends the interview details, including: position, location, time and date. The student can either accept or decline through two buttons.

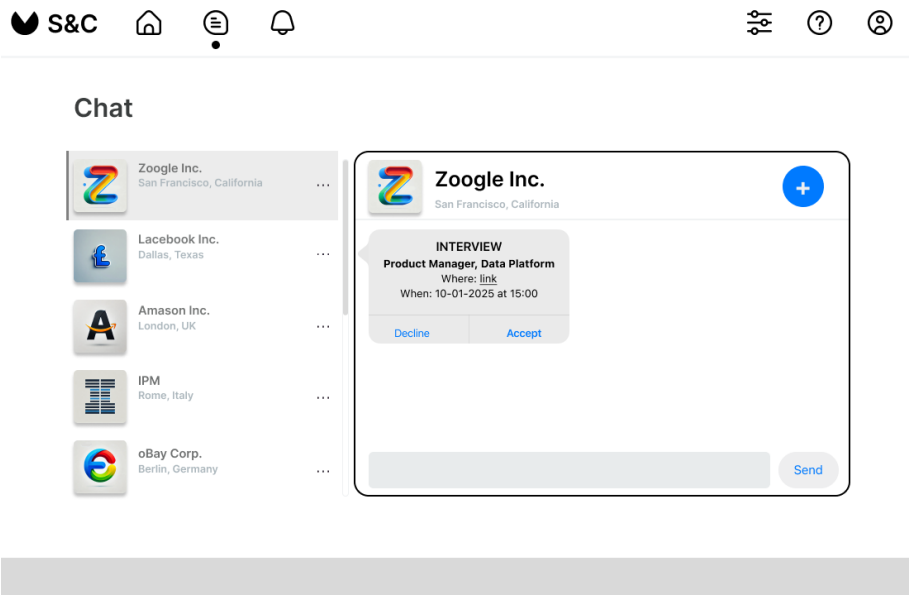

Figure 3.14: Interview proposal.

Figure 3.15: Internship Proposal The second image represents a proposal sent by a company (e.g., Zoogle Inc.) to a student in the chat. The proposal contains detailed information about the internship, such as: position, location, dates (start and end dates of the internship). The student can either accept or decline through two buttons.

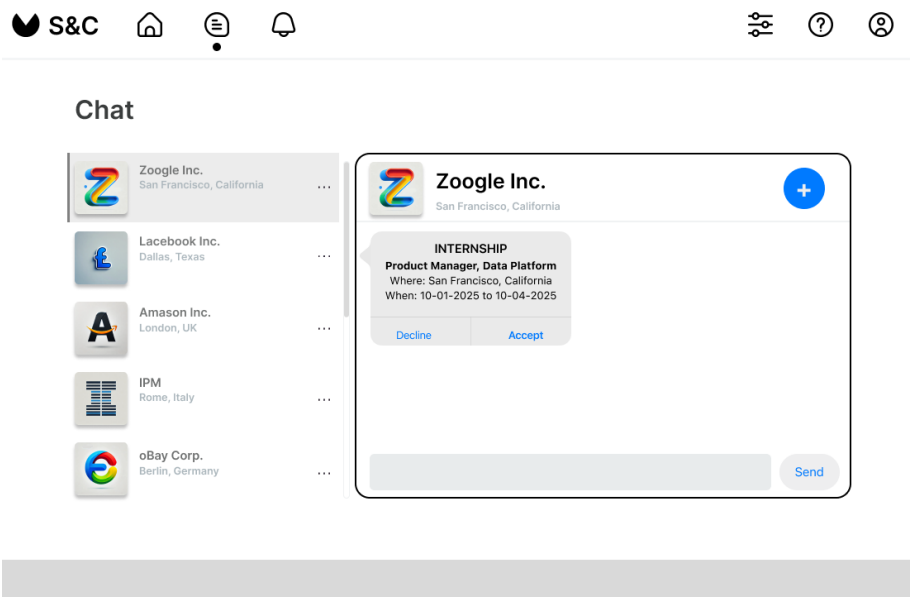

Figure 3.15: Internship proposal.

Interaction Flow

In both cases, the chat serves as the main communication tool between the company and the student. The proposals allow quick and clear decisions, fostering seamless scheduling and internship acceptance processes. Once accepted, further discussions or confirmations can be carried out in the same chat interface.

4 | Requirements Traceability

In this chapter is highlighted the relation with the functional requirements, introduced in the RASD but re-described in this section, and the components of the application.

For a more meaningful and linear analysis, the relation highlighted is only between functional requirements and sub-components of the Application Server, except from the Main Controller that, as explained before, is introduced for simplifying interaction between Web Server and back-end, so it should have been involved in every action.

Only for some requirements related exclusively to visualize some data, is listed the *DB* as the component linked.

Req ID	**Description**	**Components**
[R1]	The system should allow an unregistered guest to sign up.	Account Manager (:User Registration Manager)
[R2]	The system should allow registered students to log in.	Account Manager (:Authenticator)
[R3]	The system should allow students to insert their CVs and manage their profile information.	Account Manager (:Account Updater)
[R4]	The system should provide personalized suggestions to students for enhancing their CVs to improve their chances of being selected.	Account Manager (:Account Updater), Enhance Manager (:CVEnhancer)
[R5]	The system should notify students when internships matching their skills, experiences, and interests become available.	Recommendation Manager (:Student Recommender), Notification Manager
[R6]	The system should allow students to browse available internship postings.	*DB*

[R7]	The system allows S to visualize the profile of other C.	*DB*
[R8]	The system should allow students to apply only for open internships.	Internship Manager (:Candidates Manager)
[R9]	The system should allow S to provide feedback, suggestions and rating to improve the recommendation process.	Recommendation Manager (:Feedback Manager)
[R10]	The system allows S to establish a chat with C when mutual interest is identified.	Chat Manager (:Messages Manager)
[R11]	The system should assist S in scheduling interviews.	Chat Manager (:Interview Manager, :Messages Manager)
[R12]	The system should allow S to track the status of their applications and final selections.	Chat Manager (:Messages Manager)
[R13]	The system should not allow a student already engaged in an internship to apply for other internships.	Internship Manager (:Candidates Manager)
[R14]	The system should provide S with a mechanism to report info or complaints related to the ongoing internship they are working on.	Reporter Manager
[R15]	The system allows a maximum of one unresolved complaint per S during an ongoing internship.	Reporter Manager (:Complaint Manager)
[R16]	The system should allow S to review the S&C Calendar.	Calendar Manager
[R17]	The system should insert the S's events on the S&C Calendar.	Calendar Manger
[R18]	The system should allow registered companies to log in.	Account Manager (:Authenticator)
[R19]	The system should allow C to manage their organization profile.	Account Manager (:Account Updater)
[R20]	The system should allow C to post new internship opportunities, specifying details such as required skills, tasks, and benefits.	Internship Manager (:Internship Updater)

[R21]	The system should provide suggestions to C to improve their internship postings, making them more appealing to S.	Enhance Manager (:Internship Enhancer)
[R22]	The system allows C to visualize the profile of other Users.	*DB*
[R23]	The system should inform C about the availability of S CVs corresponding to their needs.	Recommendation Manager (:Internship Recommender), Notification Manager
[R24]	The system should allow C to provide feedback and suggestions to improve the recommendation process.	Recommendation Manager (:Feedback Manager)
[R25]	The system allows C to establish a chat with S when mutual interest is identified.	Chat Manager (:Messages Manager)
[R26]	The system should allow only one worker per internship.	Internship Manager (:Candidates Manager)
[R27]	The system should enable C to track the progress of their internship selection processes, including applications and decisions.	Chat Manager (:Messages Manager)
[R28]	The system should allow C to submit complaints or information related to their ongoing internships.	Reporter Manager
[R29]	The system allows a maximum of one unresolved complaint per C during an ongoing internship.	Reporter Manager (:Complaint Manager)
[R30]	The system should not allow applications for C's internships that are currently ongoing	Internship Manager (:Candidates Manager)
[R31]	The system should allow C to review the S&C Calendar.	Calendar Manager
[R32]	The system should insert the C's events on the S&C Calendar.	Calendar Manager
[R33]	The system should allow U to log in.	Account Manager (:Authenticator)
[R34]	The system should allow U to monitor internships of their students.	Chat Manager (:Messages Manager), Reporter Manager

[R35]	The system should allow U to interrupt ongoing internships of their S when a complaint was submitted.	Chat Manager (:Messages Manager), Reporter Manager (:Complaint Manager), Notification Manager

Table 4.1: Traceability Matrix

5 | Implementation, Integration and Test Plan

5.1. Overview

In this chapter it will be described the Implementation, Integration, and Test Plan for the S&C application. The aim is to ensure that the system is built incrementally, with a focus on reliability, modularity, and ease of testing.

The implementation and integration of the S&C application will adopt a **Bottom-Up strategy**. This method begins by focusing on the foundational components located at the lower levels of the uses hierarchy. These modules are the smallest units of functionality that do not depend on other modules. Each module will be tested using dedicated drivers to ensure correctness and reliability before being integrated into the larger system.

As modules are validated, they will replace the corresponding drivers and be incrementally integrated into functional subsystems. This iterative process ensures that each subsystem is tested and functional before further integration. The Bottom-Up strategy facilitates parallel development, allowing different teams to work on separate modules simultaneously, and simplifies debugging by isolating issues within smaller parts of the system.

This strategy promotes incremental progress, enabling the early detection of bugs and providing stakeholders with visible milestones throughout the development process. The result is a robust and well-tested system that meets the application's requirements.

5.2. Implementation plan

The implementation plan follows a **Bottom-Up** approach, starting with foundational elements and progressing toward more complex components and user-facing services. Each stage builds on the previous one, ensuring a stable foundation for the subsequent layers and modules. This structured approach ensures that all dependencies are resolved before higher-level functionalities are developed.

1. **Database Management System (DBMS):**
   The first step is the implementation of the Database Management System (DBMS), which serves as the foundation of the system.

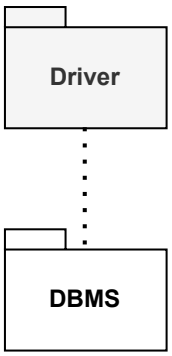

Figure 5.1: Integration 1: DBMS.

   Starting with the database ensures that all components relying on persistent data storage (e.g., Account Manager, Internship Manager) have a consistent and reliable backend.

2. **Authentication Services (Authentication Manager in Account Manager):**
   Once the DBMS is ready, the next step is implementing the Authentication Manager.

Figure 5.2: Integration 2: Authentication Manager.

The Authentication Manager is foundational for controlling access to the system. Its successful implementation ensures that subsequent components can rely on secure user identification and role management.

3. **Registration Services (Registration Managers in Account Manager):** The next step is developing the Registration Services, which handle the creation of new user accounts.

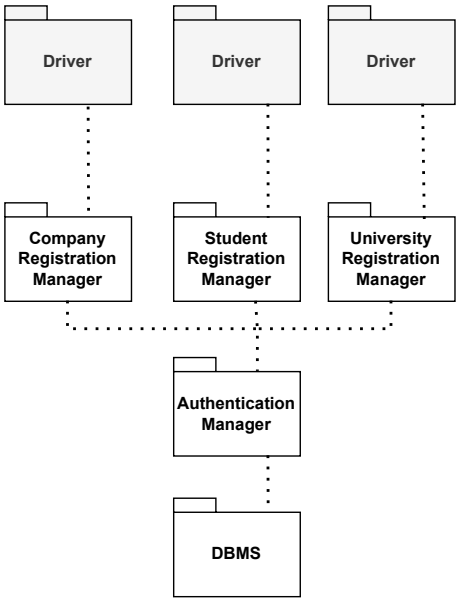

Figure 5.3: Integration 3: Registration Managers.

Registration Services depend on both the database and authentication systems. Completing these ensures that users can securely create accounts before managing profiles or accessing the system.

4. **Account Features (Account Updater in Account Manager):**
   The Account Updater handles profile management for all users.

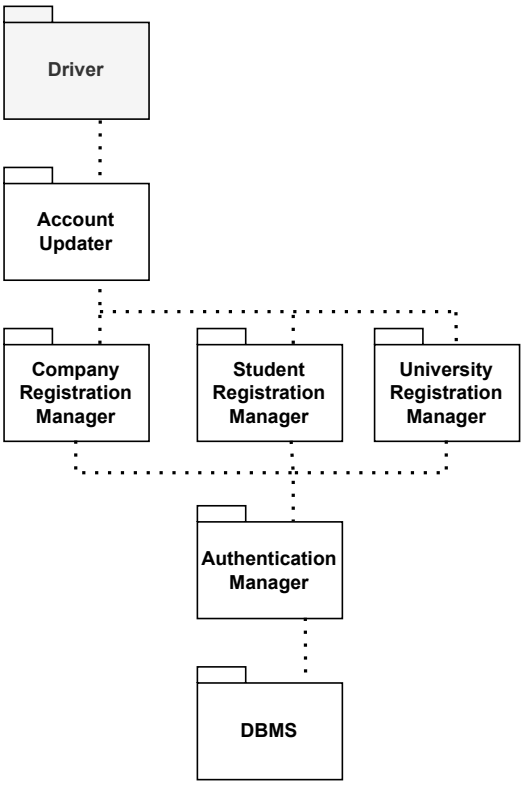

Figure 5.4: Integration 4: Account Updater.

Building the Account Updater after the Authentication and Registration Services ensures that all user actions are authenticated and roles are assigned, providing a secure context for profile management.

5. **Internship Manager:**
   The Internship Manager manages the creation, modification, and application processes for internships.

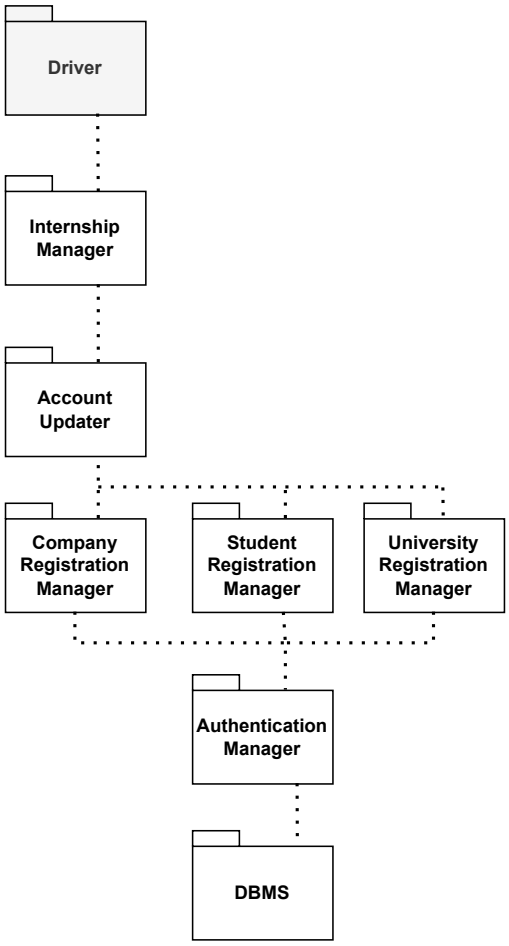

Figure 5.5: Integration 5: Internship Manager.

The Internship Manager builds upon user profiles created by the Account Manager. Its functionalities depend on a stable database and user registration system.

6. **Enhance and Recommendation Services:**
   The Enhance Manager and Recommendation Manager can be developed in parallel, as they operate independently but rely on the same underlying data structures.

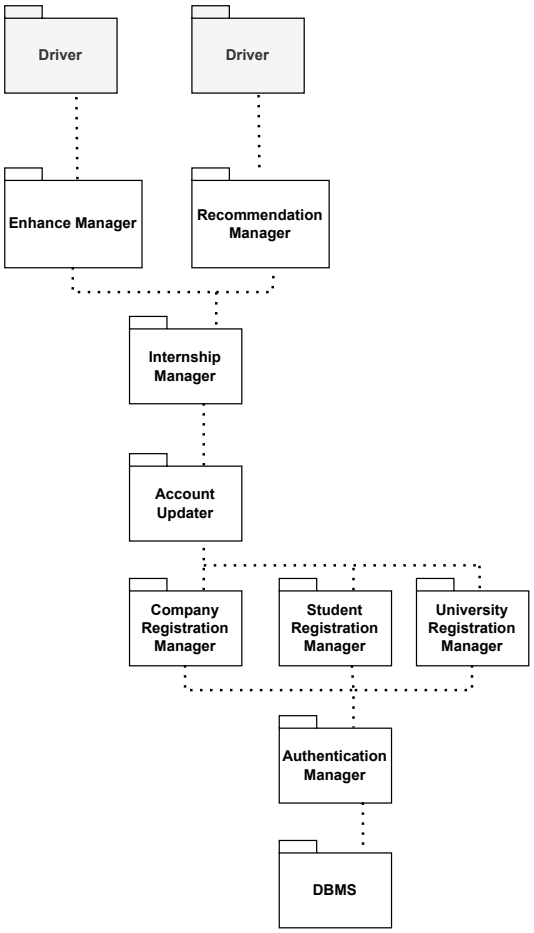

Figure 5.6: Integration 6: Enhance and Recommendation Services.

These services leverage data generated by the Internship Manager and Account Manager. Their modular nature allows simultaneous development.

7. **User Interactions (Chat and Report Manager):**
   The Chat Manager and Report Manager facilitate user interactions.

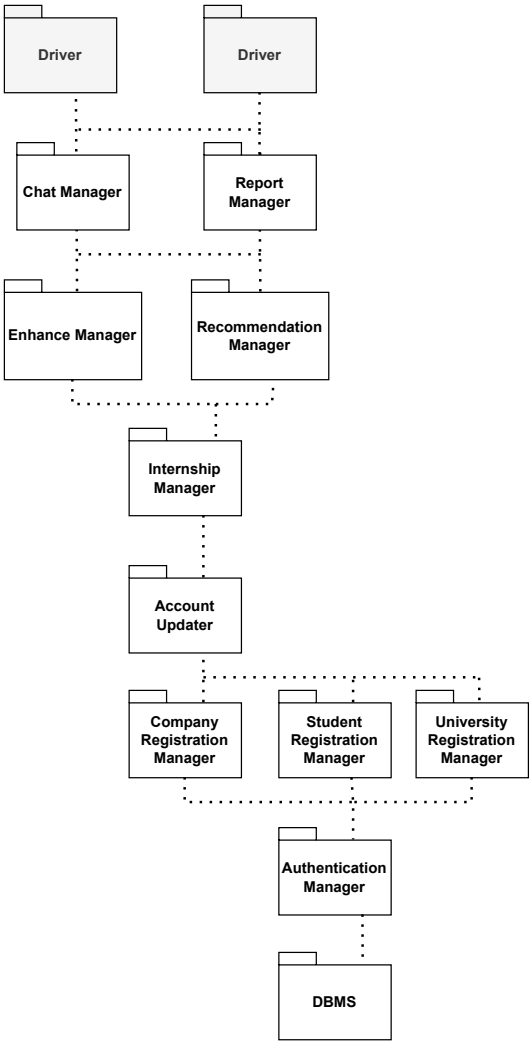

Figure 5.7: Integration 7: Chat and Report Manager.

These components rely on the existence of users, internships, and a communication infrastructure.

8. **Event Management Services (Notification and Calendar Manager):** The final step includes the Notification Manager and the Calendar Manager, as both depend on events generated by other components.

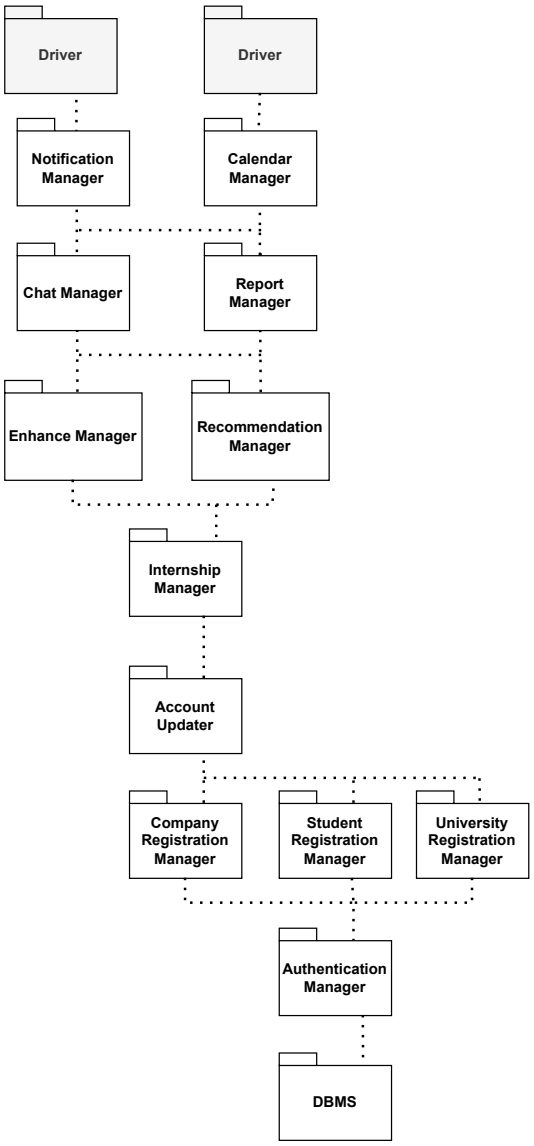

Figure 5.8: Integration 8: Notification and Calendar Manager.

Both the Notification Manager and Calendar Manager are crucial for managing user events and enhancing overall system usability. They are implemented last to ensure all triggering events are in place and functional.

5.3. System Testing Plan

Once all system components have been integrated, System Testing begins. System Testing ensures all modules function cohesively and meet requirements. The process includes:

- **Unit Testing**
  Each sub component undergoes isolated and specific testing using mock data.
- **Integration Testing**
  Test interactions between modules (e.g., data flow from the Enhance Manager to the Notification Manager).
- **Functional Testing**
  Verify that requirements, use cases and specifications previously described are properly satisfied.
- **Performance Testing**
  Evaluate the system's scalability under high traffic conditions to identify and address performance bottlenecks. The primary objective is to optimize the application's response time, resource utilization, and throughput, ensuring smooth operation and user satisfaction during peak usage periods.

6 | Effort Spent

Member of group	Effort spent	
De Matteis Alessandro	Introduction	3h
	Architectural Design	12h
	User Interface Design	3h
	Requirements Traceability	2h
	Implementation Integration and Test Plan	5h
	Logic Development	7h
Marino Margherita	Introduction	2h
	Architectural Design	17h
	User Interface Design	1h
	Requirements Traceability	1h
	Implementation Integration and Test Plan	5h
	Logic Development	6h
Monaco Giorgio	Introduction	2h
	Architectural Design	6h
	User Interface Design	13h
	Requirements Traceability	5h
	Implementation Integration and Test Plan	2h
	Logic Development	5h

Table 6.1: Effort spent by each member of the group.

7 References

7.1. References

- A.Y. 2024-2025 Software Engineering 2 Requirement Engineering and Design Project.

7.2. Used Tools

- *GitHub* for project versioning and sharing.
- LaTeX and *Overleaf* as editor for writing this document.
- *sequencediagram.org* for the sequence diagrams.
- *draw.io* for all the remaining diagrams' design.
- *figma* for the UI's design.
- *Google Documents* and *Notion* for collaborative writing, notes and reasoning.

List of Figures

2.1 Four tier architecture. 5
2.2 Component diagram. 7
2.3 Subcomponent diagram Account Manager 8
2.4 Subcomponent diagram Notification Manager 10
2.5 Subcomponent diagram Internship Manager. 11
2.6 Subcomponent diagram Chat Manager. 12
2.7 Subcomponent diagram Enhance Manager. 13
2.8 Subcomponent diagram Recomendation Manager. 14
2.9 Subcomponent diagram Reporter Manager. 15
2.10 Deployment view. 17
2.11 [UC1] Sign Up. 25
2.12 [UC2] Login. 27
2.13 [UC3] Insert Enhanced Internship Insertion. 29
2.14 [UC4] Enhance CV. 31
2.15 [UC5] Submit a Feedback. 33
2.16 [UC6] Search for a Company. 34
2.17 [UC7] Apply for an Internship. 36
2.18 [UC8] Accept a Candidate. 37
2.19 [UC9] Submit a Complaint. 39
2.20 [UC10] Submit an Information. 41
2.21 [UC11] Look for a student through recommended ones. 43
2.22 [UC12] Contact a Recommended Student. 45
2.23 [UC13] Accept a Company. 47
2.24 [UC14] Schedule an Interview. 48
2.25 [UC15] Start an Internship. 50
2.26 [UC16] Monitor internship. 52
2.27 [UC17] Interrupt internship. 54

3.1 Sign Up Student. 59
3.2 Sign Up Company. 60

3.3 Post a New Internship. 61
3.4 Post a New Internship (filled). 62
3.5 Suggestions on the Internship Post. 62
3.6 MyProfile page. 63
3.7 MyProfile page (enhance). 64
3.8 Recommended Insertion. 65
3.9 Ask for a Feedback. 66
3.10 Insertions posted. 67
3.11 Insertion post example. 68
3.12 Insertion's candidates section. 69
3.13 Review of a candidate. 70
3.14 Interview proposal. 71
3.15 Internship proposal. 72

5.1 Integration 1: DBMS. 78
5.2 Integration 2: Authentication Manager. 78
5.3 Integration 3: Registration Managers. 79
5.4 Integration 4: Account Updater. 80
5.5 Integration 5: Internship Manager. 81
5.6 Integration 6: Enhance and Recommendation Services. 82
5.7 Integration 7: Chat and Report Manager. 83
5.8 Integration 8: Notification and Calendar Manager. 84

List of Tables

4.1 Traceability Matrix 76

6.1 Effort spent by each member of the group. 87