POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Design Document

Project by:
David Gadiaga
Andrea Pesciotti
Simone Somazzi

Professor: **Matteo Camilli**
Academic Year: **2024-2025**

Contents

List of Figures

List of Tables

1 | Introduction

1.1. Purpose

Internships have become a cornerstone of academic and professional development, making it increasingly vital to provide students with streamlined access to opportunities. However, traditional internship management often relies on disjointed systems, where students struggle to discover suitable opportunities, companies face challenges in attracting qualified candidates, and universities have limited oversight of the process.

The Student&Company (S&C) Platform aims to address these challenges by offering a centralized system that bridges these gaps. Through this platform:

- **Students** can easily explore internship opportunities, enhance their CVs, and apply efficiently.
- **Companies** gain access to advanced tools for improving internship postings and managing applications effectively.
- **Universities** can monitor the entire process, mediate issues, and ensure fair practices across all parties involved.

This document provides a comprehensive overview of the platform's functionalities and constraints. It is intended for:

- **Developers:** To serve as a clear reference for the implemented requirements and as a basis for agreement between the customer and contractors.
- **The Customer:** To deliver an unambiguous description of the system's capabilities, enabling validation of the requirements and verification that the system meets expectations.

1.2. Scope

The Students&Companies (S&C) Platform is designed to streamline the process of connecting university students seeking internships with companies offering them. By providing a centralized and user-friendly interface, the platform fosters effortless interactions between students, companies, and universities, supporting the entire internship lifecycle from start to finish.

The platform caters to three primary user groups—Students, Companies, and Universities—each equipped with specific functionalities tailored to their roles:

Students can create and enhance their CVs, browse and apply for internships that align with their skills and interests, and receive personalized recommendations through a recommender system. They can also submit complaints to address any issues, track their applications and internships, and monitor progress throughout the process.

Companies can post detailed internship opportunities, receive suggestions to improve postings, and access profiles of recommended candidates. They can schedule and conduct interviews, select candidates, and manage active internships efficiently. In addition, companies can address performance or internship-related issues through the complaint submission and resolution system.

Universities play a supervisory role by monitoring the progress and feedback of affiliated students' internships. They can handle complaints, intervene when necessary, and ensure internships meet academic and ethical standards.

1.3. Definition, Acronyms, Abbreviations

Acronym	Definition
DD	Design Document
RASD	Requirements Analysis & Specification Document
ST	Student
UNI	University
STG	Student Group
S&C	Student&Company
UC	Use Case
UI	User Interface
User	All Students, Companies and Universities

API	Application Programming Interface
RX	Requirement X
CMP	Component
UML	Unified Modelling Language
DB	Database
DBMS	Database Management System
REST	Representational State Transfer
MCV	Model View Control

1.4. Revision History

The following are the revision steps made by the team during the DD development:

- **Version 1.0** - 07/01/2025

1.5. Reference Documents

- Specification of RASD and DD assignment
- Slides of the course of Software Engineering 2

1.6. Document Structure

The document is organized into seven sections, as outlined below:

- **Introduction:** This section highlights the importance of the Design Document and provides clear definitions and explanations of acronyms and abbreviations. It also revisits the scope of the Student&Company system, setting the stage for the subsequent sections.
- **Architectural Design:** This section presents the primary components of the system and their interrelationships. It emphasizes key design decisions, architectural styles, patterns, and paradigms adopted for the system.
- **User Interface Design:** The third section describes the system's user interface, including mockups and detailed explanations of the main pages, illustrating how users interact with the platform.
- **Requirements Traceability:** This section outlines the system requirements and demonstrates how the design decisions fulfill them, ensuring a clear traceability of

requirements throughout the design process.

- **Implementation, Integration, and Test Plan:** This part provides an overview of the implementation of the system's components, describes their integration, and outlines a comprehensive plan for testing them to ensure functionality and reliability.
- **Effort Spent:** The sixth section documents the number of hours each team member contributed to developing this Design Document, offering transparency regarding the effort distribution.
- **References:** The final section lists all the documents and resources used in drafting this Design Document, ensuring proper attribution and traceability of information sources.

2 | Architectural Design

2.1. Overview

The S&C platform employs a three-tier architecture, a widely adopted design pattern in software development that separates applications into three distinct layers: the presentation layer, the application layer, and the data layer. Each of these layers serves a specific function, contributing to the system's scalability, maintainability, and flexibility. Below are the details of each layer:

- **Presentation Layer:** This is the user interface (UI) of the application, responsible for how the system is presented to the end-user. It interacts directly with the user, capturing inputs and displaying results. The presentation layer ensures that users can easily access and interact with the system's functionalities, typically through web browsers, mobile apps, or desktop applications.
- **Application Layer:** Often referred to as the business logic layer, this is where all the processing of data and execution of business rules occurs. It receives requests from the presentation layer, processes the logic, and sends responses back. The application layer is responsible for handling complex operations such as data validation, computations, decision-making processes, and ensuring that data flows smoothly between the user interface and the database.
- **Data Layer:** This layer is responsible for storing, managing, and retrieving data. It typically includes databases, file systems, or other data storage solutions that support the application's needs. The data layer ensures that all data related to the application—such as user information, transactional records, or application settings—is organized, secured, and accessible when needed. The application layer communicates with this layer to retrieve and update information as required.

The primary benefit of this three-tier architecture is that each layer can be developed, maintained, and scaled independently by different development teams. This modular approach allows for better resource management and reduces the complexity of managing

the system as a whole. Changes made to one layer, such as updates to the presentation or application logic, can be implemented without affecting the other layers. Additionally, scalability is a significant advantage, as each tier can be scaled up or down independently depending on demand, ensuring optimal performance and resource utilization without impacting the entire system. This separation of concerns also improves maintainability, as each layer can be updated or replaced with minimal disruption to the others.

2.2. Component View

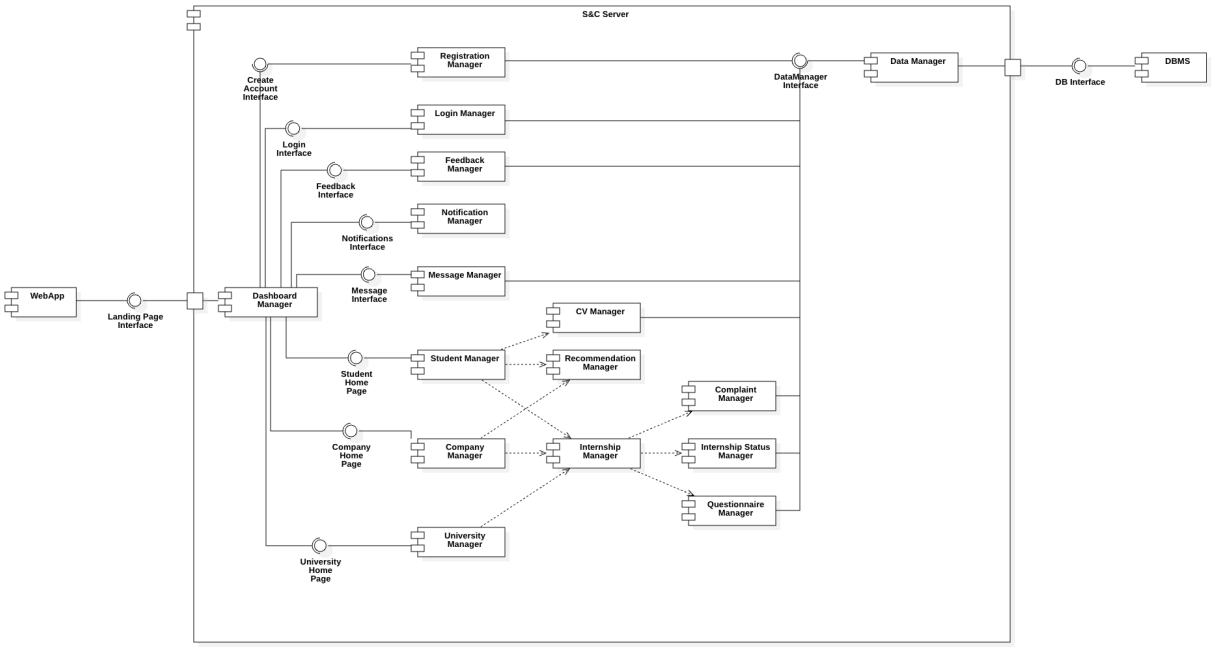


Figure 2.1: Component Diagram

- **Dashboard Manager**: It is the pivotal component that orchestrates all communication between users and the S&C platform. Acting as the central hub for user interactions, it manages all requests routed through the Dashboard Interface. Whether users are registering, applying for internships, or managing their profiles, the Dashboard Manager directs their requests to the appropriate system components. By integrating with other modules, it provides an easy user experience while ensuring efficient communication across the platform.
- **Registration Manager**: it is responsible for handling the registration process for all users on the platform, including students, companies, and universities. It ensures that user data is properly validated and securely stored. The manager also over-

sees the verification process for universities, ensuring that institutions are properly accredited. Additionally, it manages account recovery workflows and ensures that users activate their accounts correctly.

- **Login Manager**: it handles the login process for registered users. It is responsible for managing user authentication, session management, and ensuring that secure login mechanisms, such as two-factor authentication and CAPTCHA, are implemented. The Login Manager also manages account lockouts, password resets, and controls user access levels to maintain the system's security.
- **Student Manager**: it oversees all functionalities related to students on the platform. This component allows students to update their personal details and manage their profiles. It serves as a point of interaction between students and other modules, such as internships and recommendations.
- **Company Manager**: it is responsible for managing all company-related functionalities on the platform. It ensures that companies provide complete and accurate details on their profiles and that internship postings are up to date. The Company Manager also assists companies in improving the quality of their internship postings, ensuring they meet the platform's standards and attract the best candidates.
- **University Manager**: it manages the registration and verification of universities on the platform. It provides universities with tools to monitor and track the internships of their affiliated students. The University Manager also facilitates the resolution of complaints, ensuring that internships comply with academic standards and regulations.
- **CV Manager**: it allows students to upload, edit, and enhance their CVs. This module provides suggestions and best practices to help students create impactful CVs and integrates with the Recommendation Manager to improve the content of students' CVs based on feedback received.
- **Internship Manager**: it handles the entire lifecycle of internships, from posting opportunities to completion. It allows companies to post, update, and manage internships and enables students to apply and track their internship status.
- **Internship Status Manager**: it provides comprehensive tracking and management of internship statuses. Beyond simply updating statuses such as "pending," "active," or "completed," it facilitates deeper insights by generating progress reports for universities and companies. It allows stakeholders to review milestones, address any delays, and ensure that internships adhere to expected timelines.

- **Questionnaire Manager**: it distributes and manages feedback forms sent to students and companies. It not only collects and analyzes responses to evaluate internship quality but also uses the insights to suggest actionable improvements. The manager ensures all feedback is utilized to enhance the overall internship experience.
- **Message Manager**: it is responsible for facilitating direct communication among users. This module ensures direct messaging between students, companies, and universities. It integrates with the Notification Manager to notify users of unread messages and supports features such as file sharing and automated replies. The Message Manager also archives communications to ensure accountability and provide a reference for future interactions.
- **Complaint Manager**: it is responsible for handling the submission and resolution of complaints from students, companies, and universities. It ensures that all complaints are properly logged, categorized, and processed in an efficient manner. The Complaint Manager works closely with the University Manager to address any academic-related complaints and ensures that all complaints are handled according to platform policies.
- **Recommendation Manager**: it provides personalized internship recommendations for students based on their CVs and preferences. It also suggests potential candidates to companies based on their internship requirements. The Manager uses keyword-based and statistical analysis to improve the accuracy of recommendations, ensuring that students and companies are paired effectively.
- **Feedback Manager**: it collects and analyzes feedback from users about the platform and its features. It generates actionable insights from the feedback to continuously improve the user experience. This component is crucial for refining CV suggestions, internship postings, and recommendations, ensuring the platform evolves based on user input.
- **Notification Manager**: it handles all notifications and alerts sent to users, such as updates on internship statuses, application statuses, and reminders. It ensures that notifications are sent in a timely and relevant manner through email, SMS, or in-app messages. The Notification Manager also manages user preferences, ensuring that notifications are relevant and that users can customize how and when they receive alerts.
- **Data Manager**: it handles all data storage and retrieval operations, interfacing directly with the DBMS through the DB Interface. It provides a centralized ac-

cess point for components like the Registration and Internship Managers via the DataManager Interface, ensuring secure, consistent, and efficient data management across the system.

- **Email Manager**: it handles all email related operations, primarily focusing on user verification. It sends account activation emails during registration and manages password recovery emails. It ensures secure and reliable communication between the platform and its users.

2.3. Deployment View

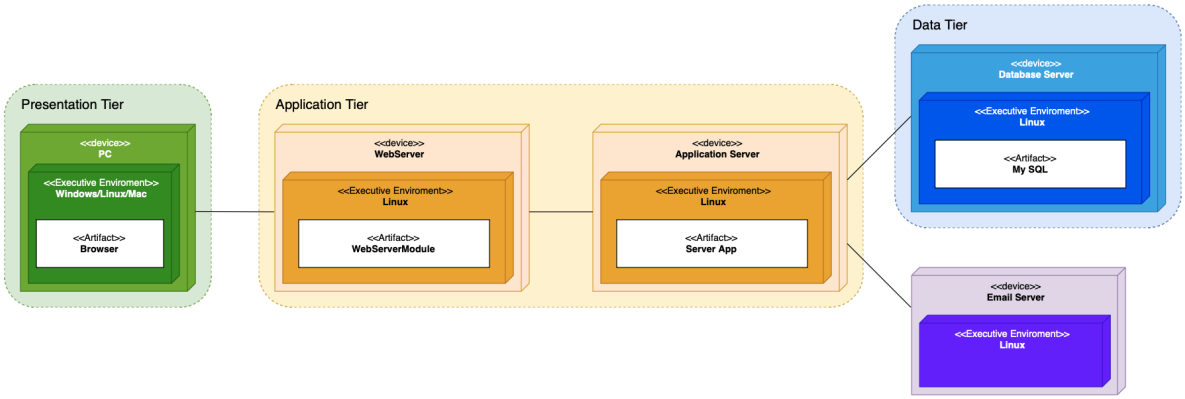


Figure 2.2: Deployment Diagram

The deployment view presented here is important as it describes the system's execution environment while also illustrating the topological distribution of the application.

The system's deployment architecture ensures efficient operation, scalability, and security across its components. Users can access the system using any personal computer or device with a web browser, such as mobile phones, tablets, or desktops. The browser communicates with the Web Server to send and receive requests.

The Web Server acts as a gateway, handling incoming client requests and performing load balancing to distribute them across multiple Application Servers. It does not execute business logic but instead ensures that traffic is managed efficiently, preventing server overload. The Web Server also provides the necessary HTML, JSON, JavaScript, and CSS files to clients.

A firewall sits between clients and the system to ensure security. It limits access by applying strict rules, protecting the system from potential malicious intrusions and attacks.

It also checks for any suspicious requests before they reach the server.

The Application Server is the heart of the system's business logic. It manages requests from the Web Server, routing them to the appropriate module through the Dashboard Manager. The Application Server communicates with the Database Server via a model gateway, ensuring that any changes to the database are processed and updated via TCP/IP. It is replicated to handle large user traffic and ensure system scalability.

The Database Server stores and manages critical system data using MySQL. The Application Server retrieves and updates this data through the model module and database driver to maintain data consistency.

After registration, users must confirm their accounts by clicking a verification link sent via email. The Application Server contacts the Email Server using the SMTP protocol to send the confirmation email immediately after the registration process, ensuring secure user verification.

2.4. Runtime View

2.4.1. User Registration

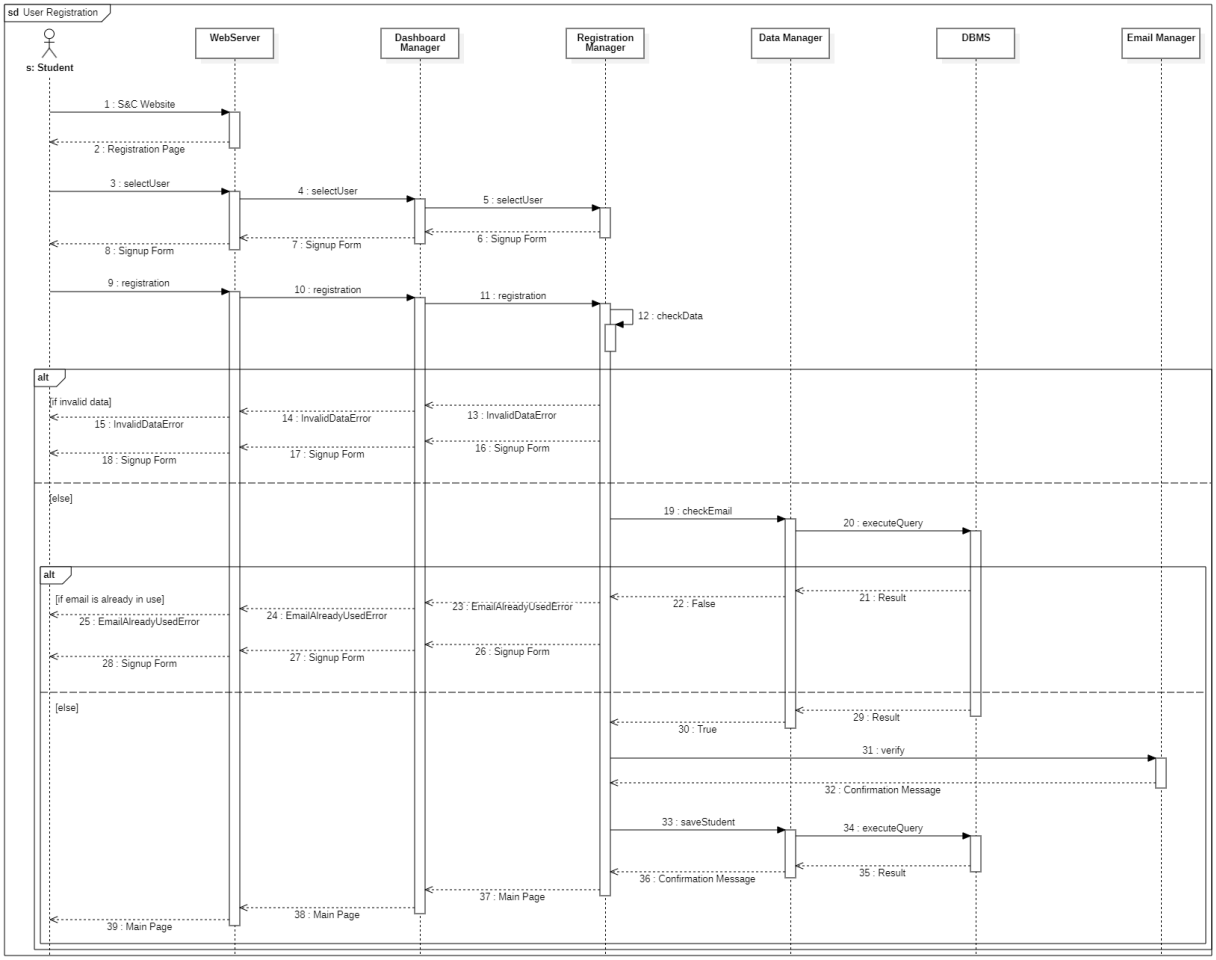


Figure 2.3: Diagram for [UC1]

The user registration process begins with the user selecting their role (student, company, or university) and accessing the signup form. The form requires personal details, such as name, email, and password, and additional information based on the role. Once the data is submitted, the system validates its completeness and verifies whether the email is already in use. Errors prompt the user to revise their input. If the data passes validation, the system saves the information, sends a verification email, and redirects the user to the main page. This process ensures secure and role-specific onboarding for all types of users.

2.4.2. User Login

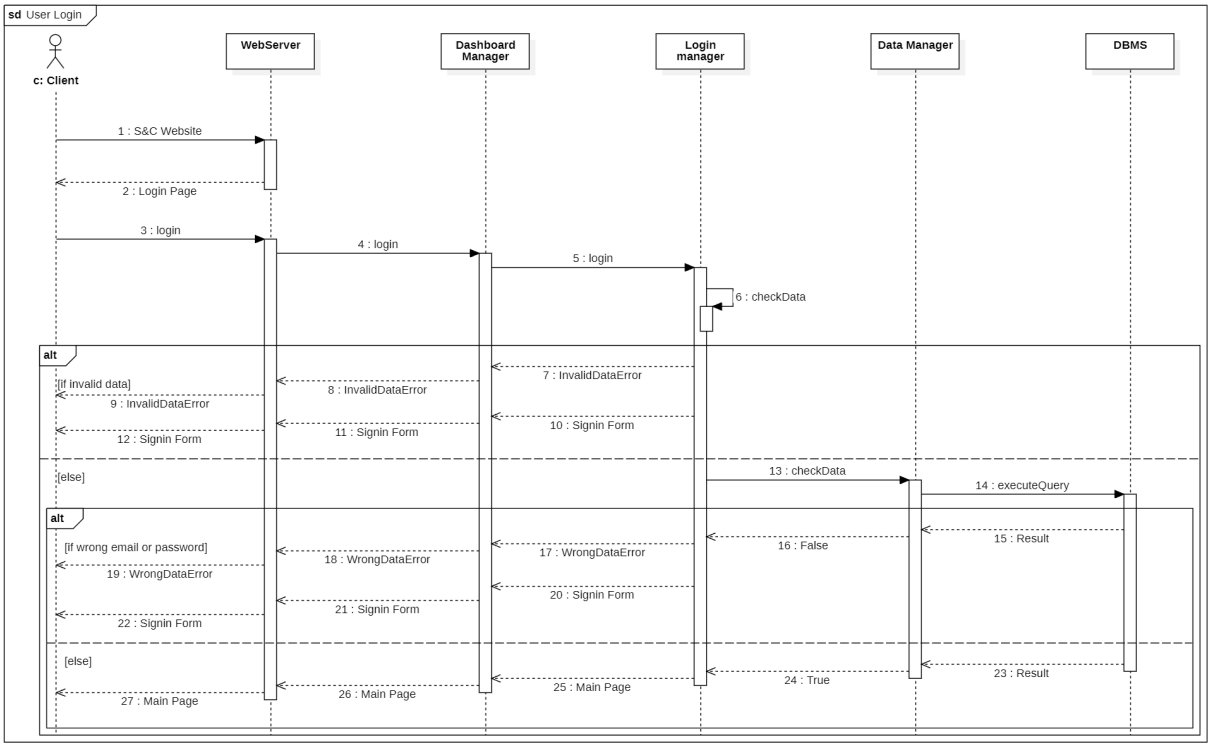


Figure 2.4: Diagram for [UC2]

This diagram outlines the user login process on the platform. The user accesses the login page and submits their email and password. The system validates the input, checking for completeness and correctness. If any issues are found, such as missing fields or invalid credentials, an error message is displayed, guiding the user to resolve the problem. If the credentials are valid, the system authenticates the user and grants access to the main dashboard. For invalid email-password combinations, specific error feedback is provided to avoid confusion and streamline the login experience.

2.4.3. Insert CV Information

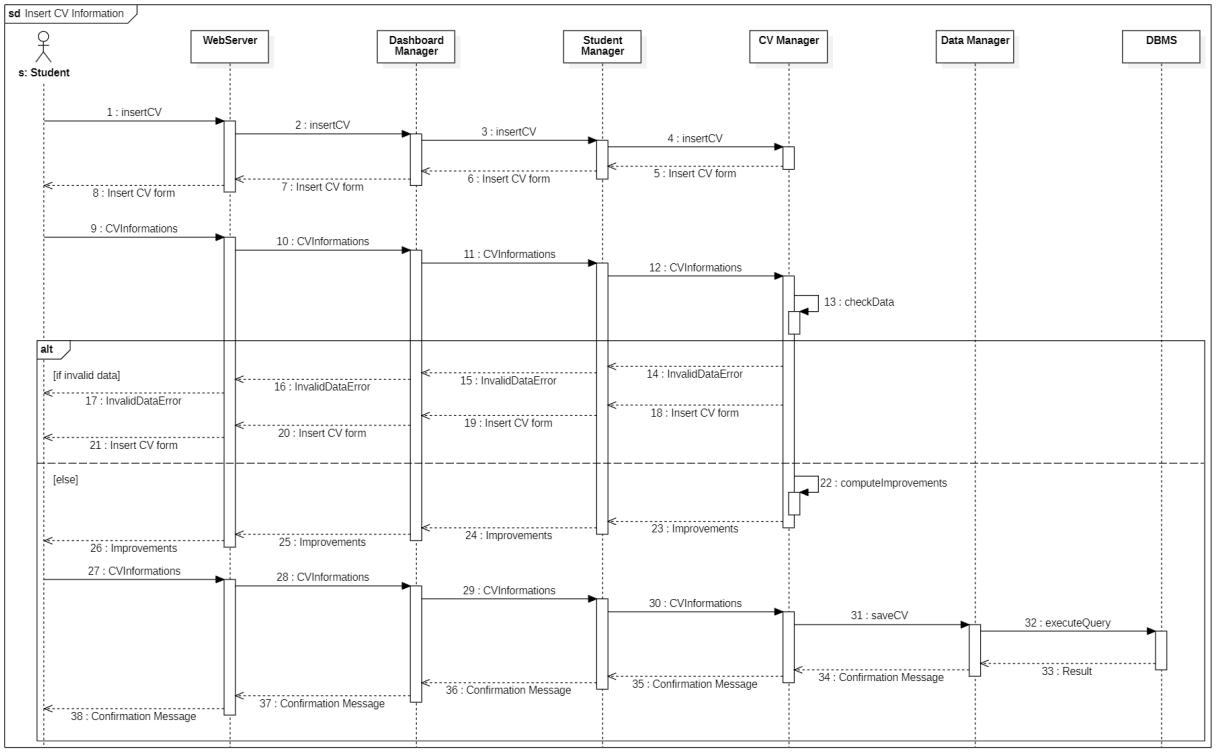


Figure 2.5: Diagram for [UC3]

This diagram describes how a student adds their CV information to the platform. The student accesses the CV insertion form, where they input details such as education, skills, and work experience. The system validates the data for completeness and accuracy. If errors are found, an error message prompts corrections. For valid inputs, the system provides improvement suggestions to optimize the CV's presentation. Once finalized, the CV is saved in the database, and the student receives a confirmation message. This ensures students can maintain professional, high-quality CVs.

2.4.4. Update CV Information

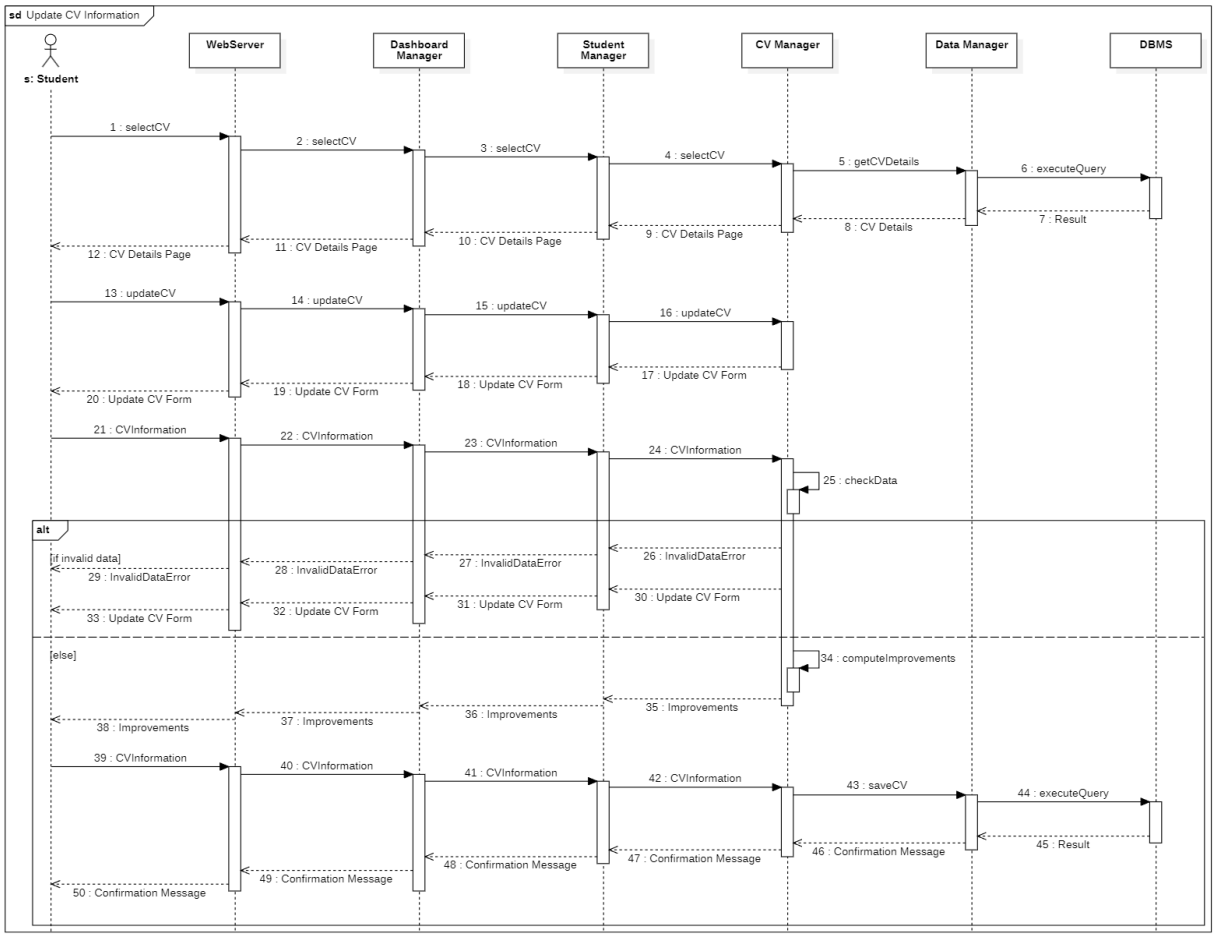


Figure 2.6: Diagram for [UC4]

This diagram describes how a student updates their CV on the platform. The student selects their CV, views its current details, and accesses the update form to modify fields like education, skills, or work experience. Once submitted, the system validates the input for completeness and correctness, providing feedback on any errors. If the data is valid, the system may offer suggestions to further improve the CV's structure or content. After finalization, the updated CV is saved to the database, and a confirmation message is sent to the student. This process ensures students maintain accurate and competitive CVs on the platform.

2.4.5. Create Internship

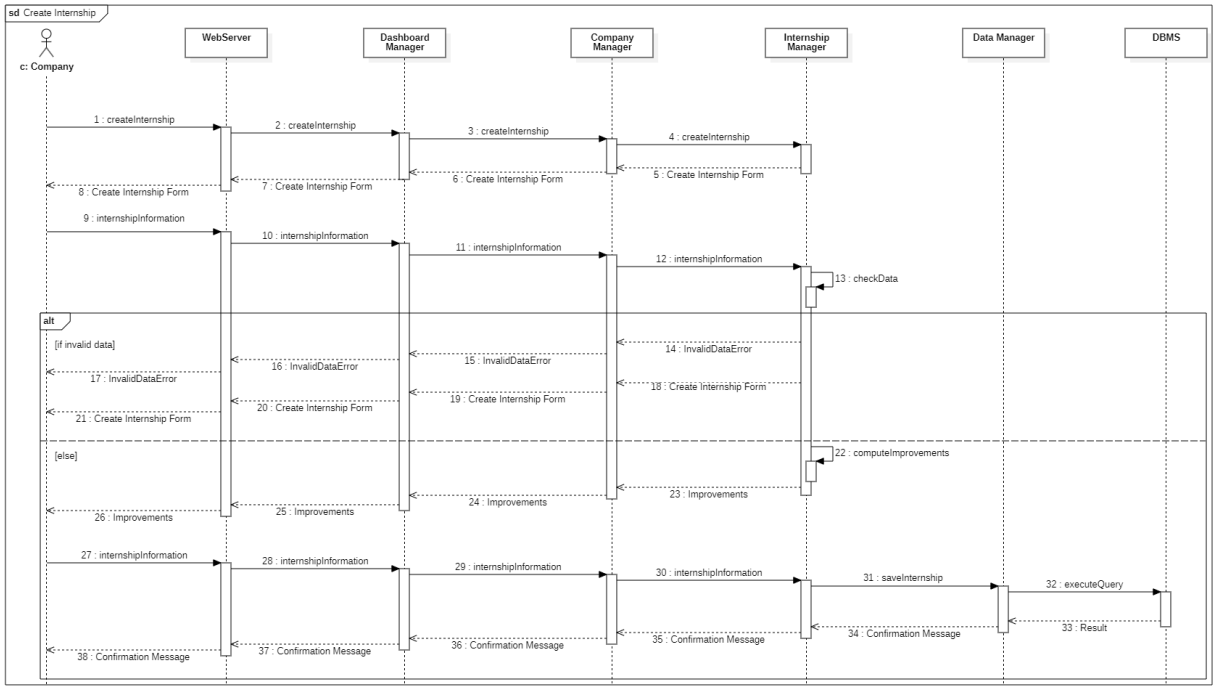


Figure 2.7: Diagram for [UC5]

This diagram explains how a company creates a new internship listing. The company begins by accessing the internship creation form, where they input details like the internship name, description, deadline, and requirements. After submitting the form, the system validates the input. If errors are detected, the system provides feedback for corrections. For valid data, improvement suggestions may be offered to optimize the internship posting. Once finalized, the internship is saved in the database, and a confirmation message is sent to the company. The process ensures that internships are detailed, accurate, and aligned with platform standards.

2.4.6. Update Internship

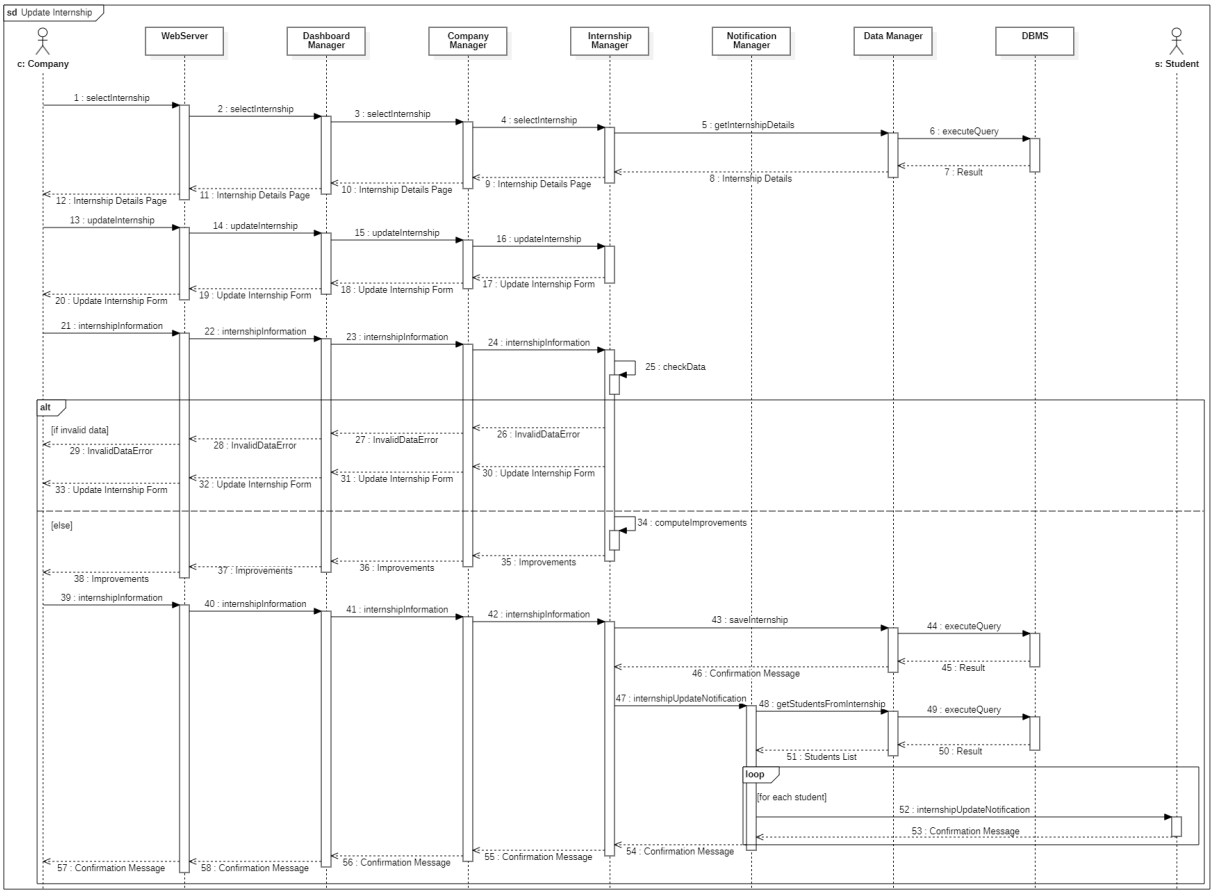


Figure 2.8: Diagram for [UC6]

This diagram illustrates the process for a company to update an internship listing. The company begins by selecting the internship to edit, and the system retrieves its details. After making changes, the company submits the updated information, which the system validates for completeness and accuracy. If errors are detected, the system displays a message prompting corrections. If valid, the system may suggest improvements to refine the listing, such as clarifying descriptions or adjusting deadlines. Once the updates are finalized, the information is saved in the database. Notifications are sent to all students associated with the internship, informing them of the changes and maintaining transparency.

2.4.7. Search Internship

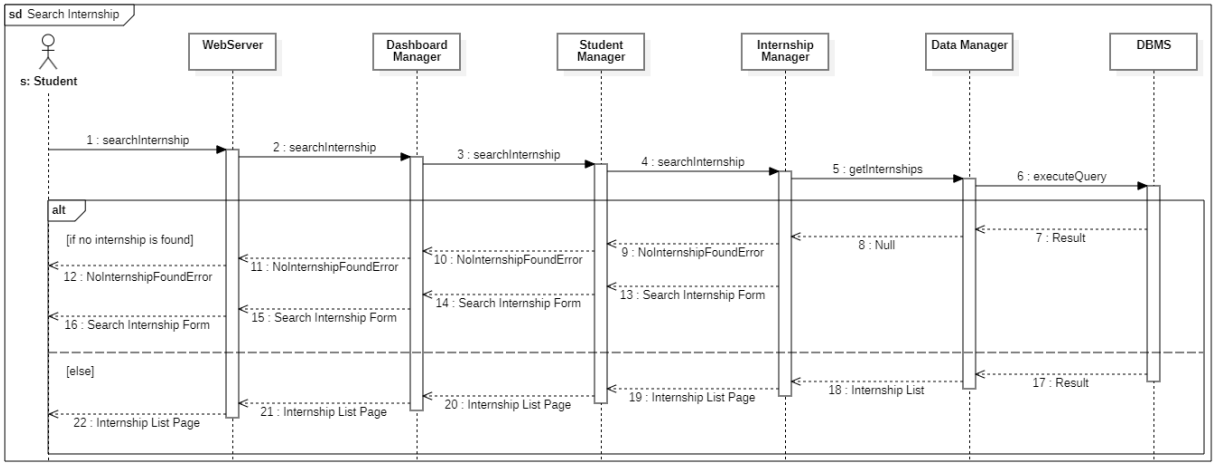


Figure 2.9: Diagram for [UC7]

This sequence diagram depicts how a student searches for internships on the platform. The student provides search criteria, such as keywords, deadlines, or required skills, which the system uses to query the database. If no matches are found, the system displays an error and suggests refining the search. If matches are found, a list of internships is returned, allowing the student to browse through opportunities. This process ensures that students can efficiently find internships that align with their preferences and qualifications, facilitating their journey toward securing practical experience.

2.4.8. Submit Application

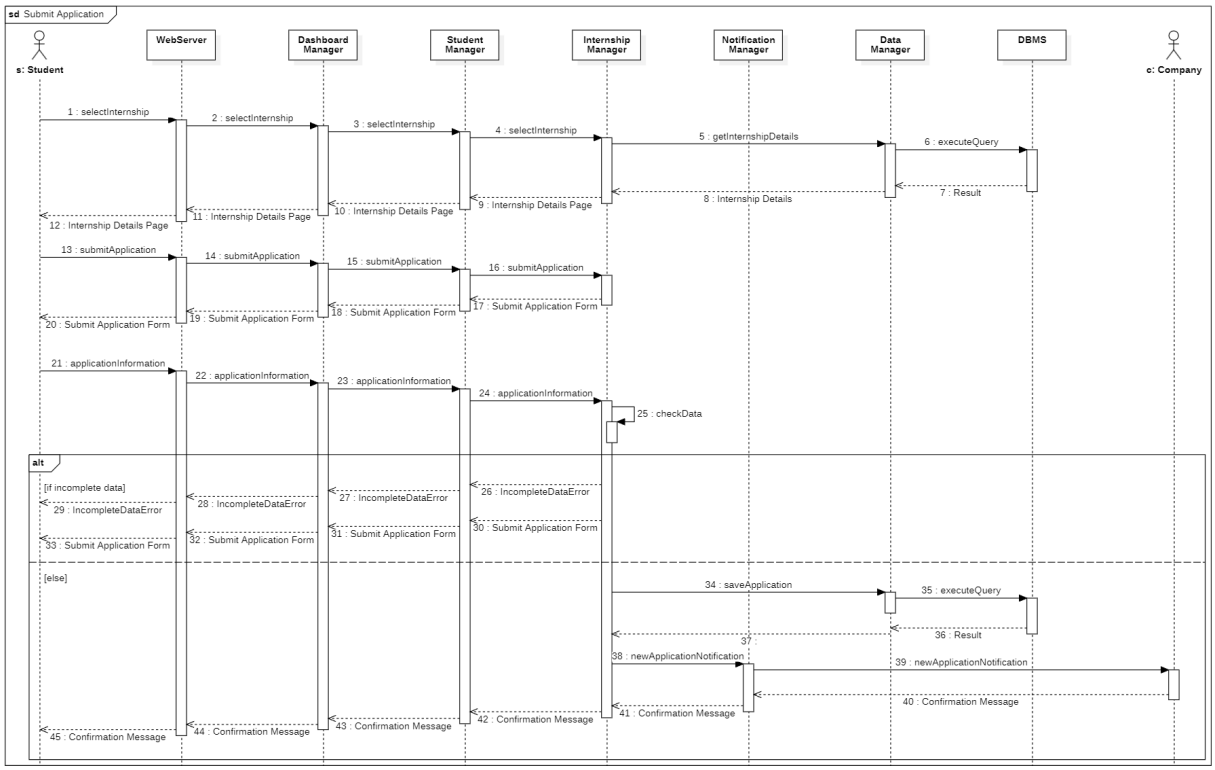


Figure 2.10: Diagram for [UC8]

The diagram explains how a student submits an application for an internship. After selecting an internship, the student views its details and accesses the application form. They fill out the required fields, such as a cover letter or additional details, and submit the form. The system validates the input, ensuring all fields are complete and free of errors. If valid, the application is saved in the database, and the company offering the internship is notified. The student receives a confirmation message, ensuring transparency and completion of the process. This facilitates an easy application experience for both students and companies.

2.4.9. Insert Questionnaire

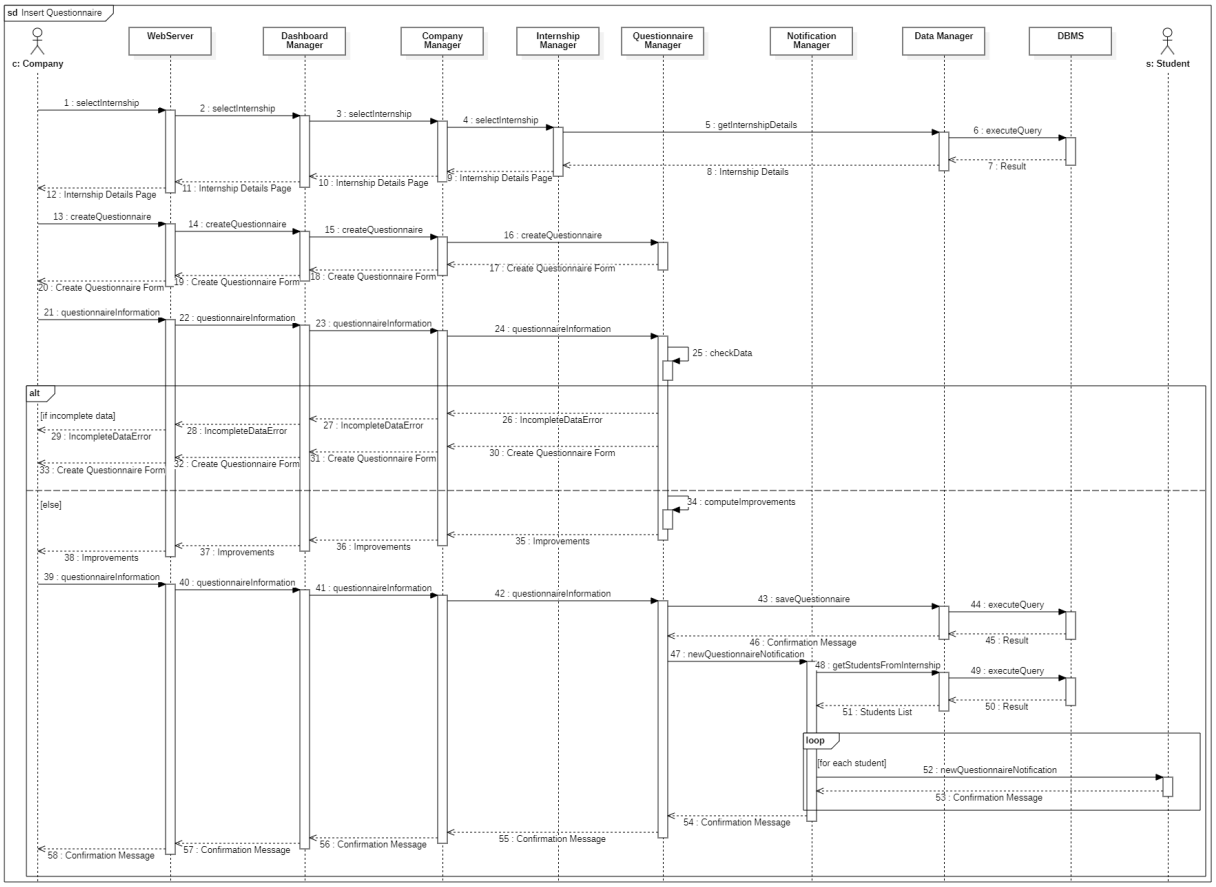


Figure 2.11: Diagram for [UC9]

This diagram shows the process for a company to create a questionnaire for an internship. The company begins by selecting the internship and accessing the questionnaire creation form. They input the questions and structure of the questionnaire and submit it for validation. The system checks the data for completeness and correctness, displaying errors if necessary. For valid inputs, improvement suggestions are provided to enhance the questionnaire. Once finalized, the questionnaire is saved to the database. Notifications are sent to students associated with the internship, informing them of the new questionnaire and enabling them to respond promptly.

2.4.10. Update Questionnaire

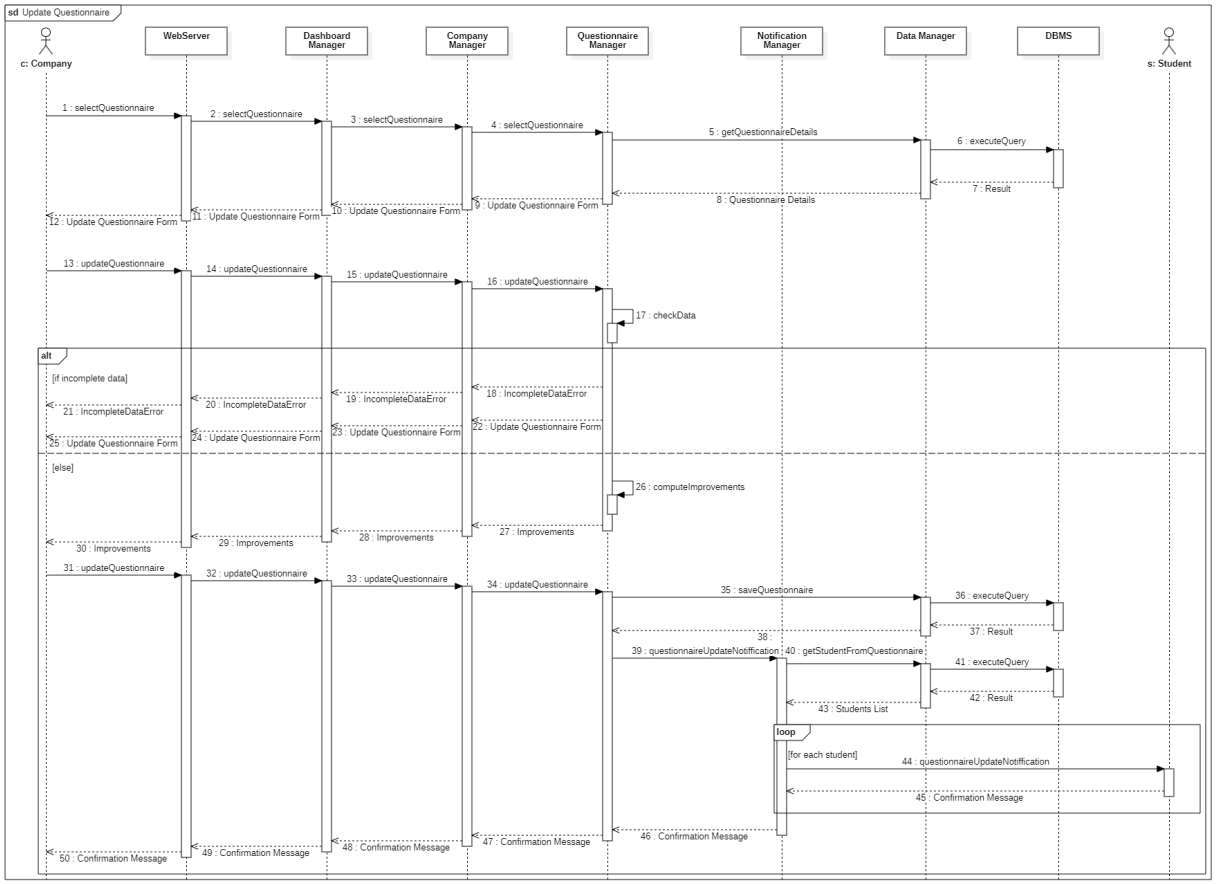


Figure 2.12: Diagram for [UC10]

This sequence diagram explains how a company updates an existing questionnaire linked to an internship. The process starts with the company selecting the questionnaire to edit, and the system retrieves its current details. The company updates the questions or structure and submits the changes. The system validates the new data, displaying errors for incomplete or incorrect inputs. If valid, suggestions for improvement may be offered, such as enhancing clarity or relevance. Once finalized, the updated questionnaire is saved to the database. Notifications are sent to the students associated with the internship, informing them of the updated content to ensure they are aware of the changes.

2.4.11. Answer Questionnaire

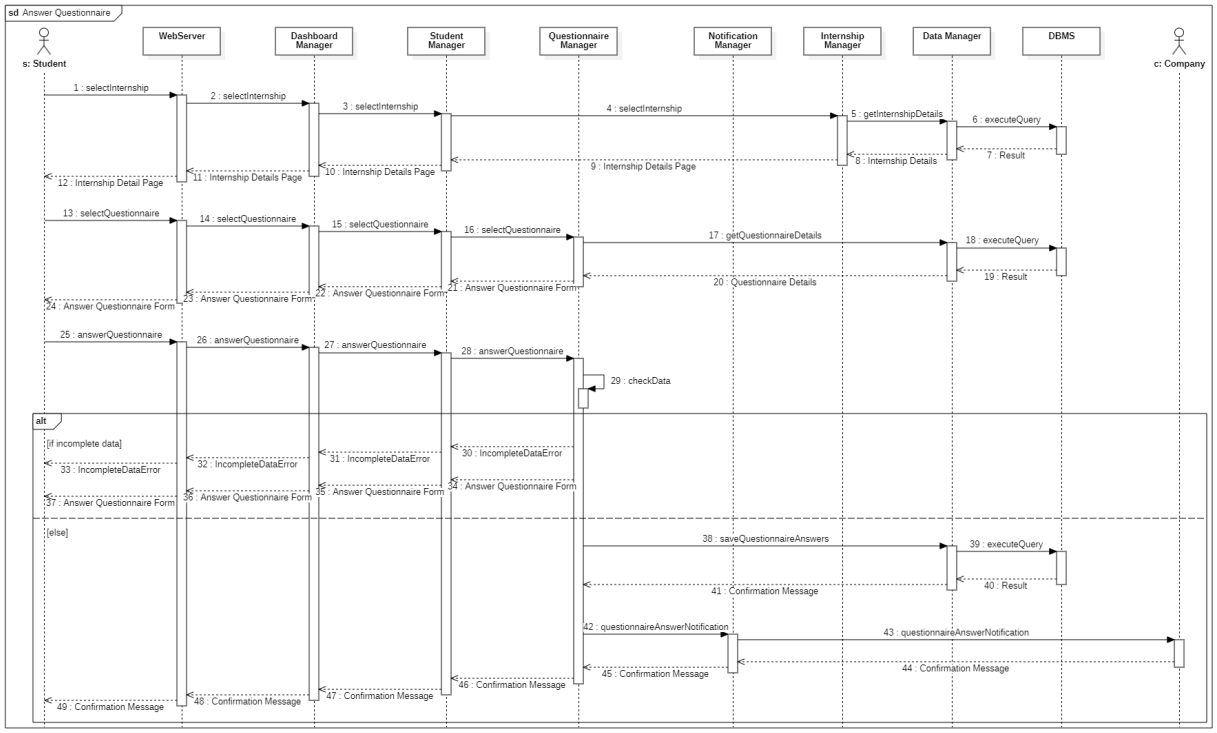


Figure 2.13: Diagram for [UC11]

This sequence diagram illustrates the process of a student answering a questionnaire linked to an internship. The student selects an internship, and the system retrieves its details for display. They then select the associated questionnaire and access its form. After completing the questionnaire, the system validates the responses. If there are incomplete or invalid answers, the system prompts the student to correct the input. Once the answers are complete, the system saves the responses to the database. A notification is sent to the company, and the student receives a confirmation message, ensuring the process is completed transparently.

2.4.12. Evaluate Application

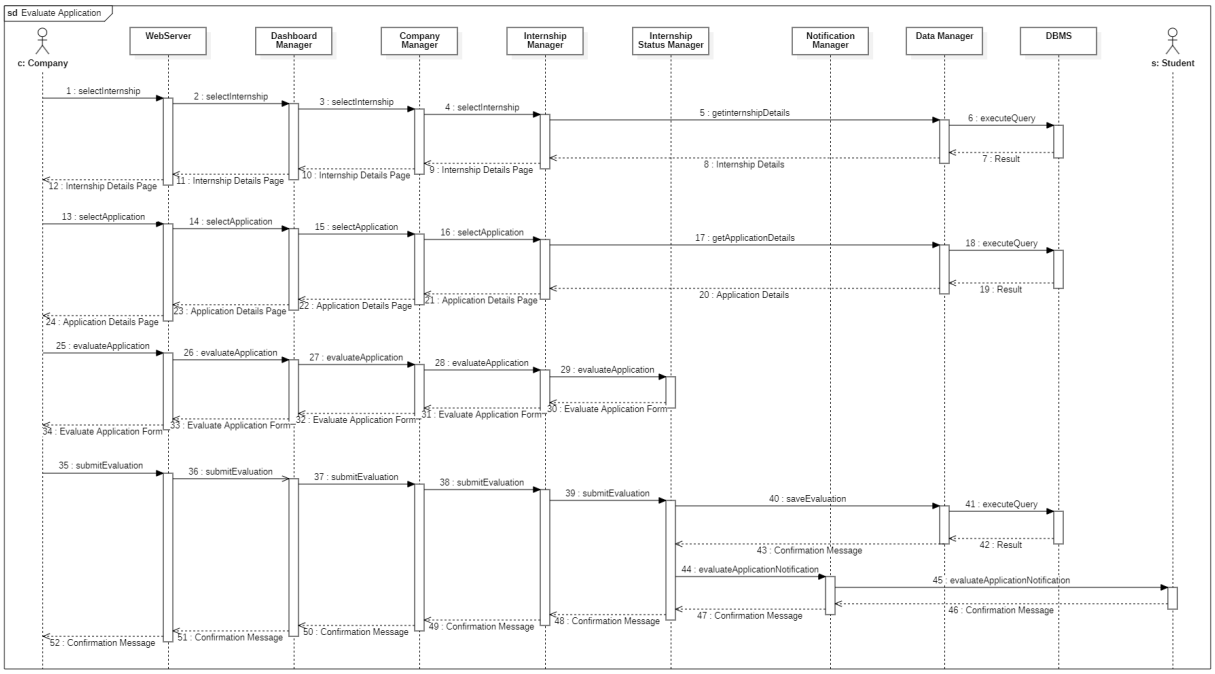


Figure 2.14: Diagram for [UC12]

This sequence diagram illustrates the process by which a company evaluates a student's application for an internship. The process begins when the company selects the internship for which they want to evaluate applications, and the system retrieves the relevant details. The company then selects a specific application from the list, and the system fetches its details for review. After reviewing the application, the company accesses the evaluation form, where they provide feedback or a decision (e.g., accept or reject). The system validates the evaluation and saves it to the database. A notification is sent to the student regarding the outcome, and the company receives a confirmation message, ensuring transparency and effective communication.

2.4.13. Monitor Internship

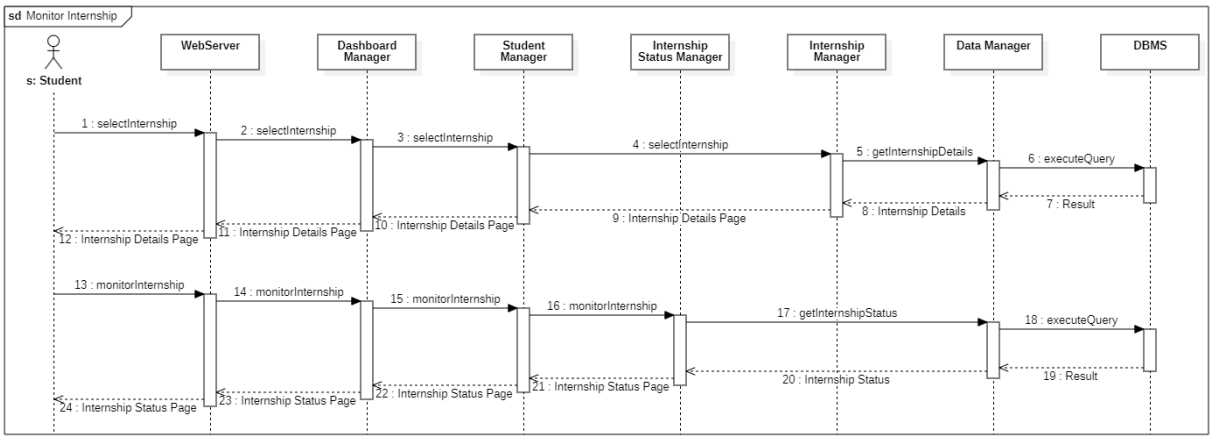


Figure 2.15: Diagram for [UC13]

This diagram illustrates how a student monitors their internship's status. The student selects their internship from a list and accesses its details. The system retrieves the current status from the database, such as ongoing, completed, or interrupted, and displays it on the monitoring page. This functionality provides students with up-to-date information on their internship progress and any related changes. By making this information accessible, the system ensures students remain informed and can take action if issues arise.

2.4.14. Write Feedback

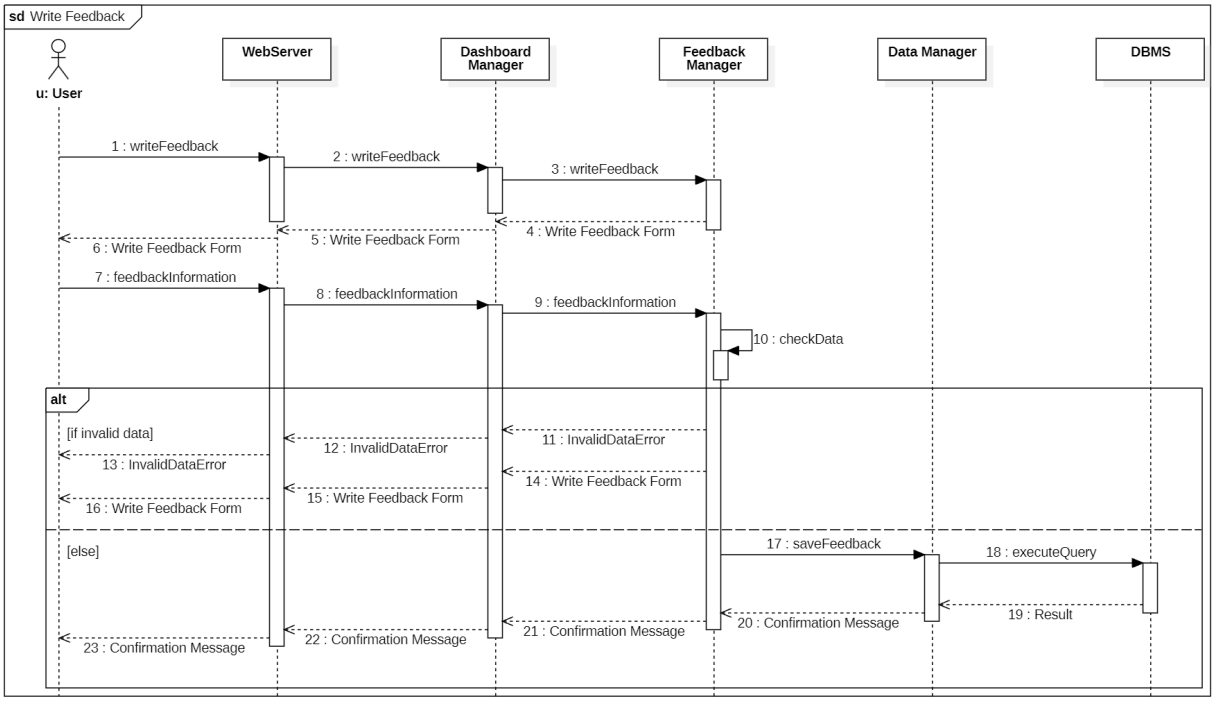


Figure 2.16: Diagram for [UC14]

This sequence diagram shows how a user provides feedback on the platform. The user accesses the feedback form and enters details about their experience or suggestions for improvement. The system validates the input for completeness. If the data is incomplete or incorrect, the system prompts the user to revise their input. Once validated, the feedback is saved in the database, and a confirmation message is sent to the user. This feature ensures the platform can collect valuable insights to enhance its functionality and user experience.

2.4.15. Write Complaint

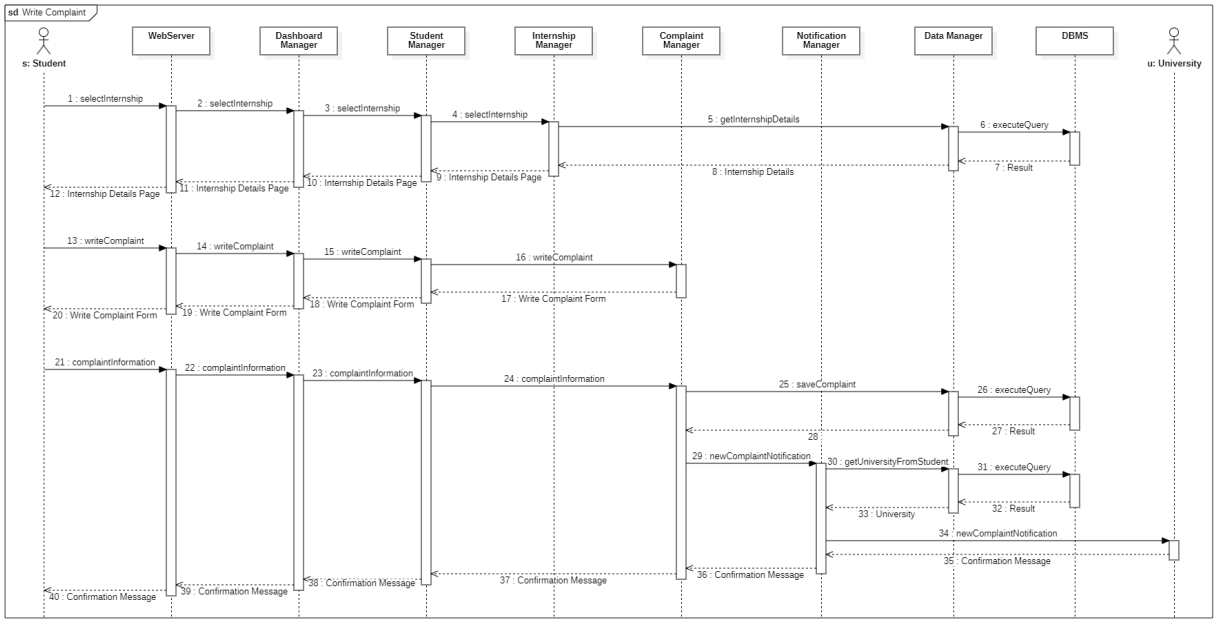


Figure 2.17: Diagram for [UC15]

This diagram explains how a student can file a complaint about an internship. The student begins by selecting the relevant internship and viewing its details. They then access the complaint form and input the required information, such as the nature of the complaint and any relevant details. Once submitted, the system validates the data for completeness and accuracy. The complaint is saved in the database and forwarded to the university responsible for the student. Notifications are sent to the student and the university to acknowledge the submission and initiate follow-up actions. This ensures transparency and accountability for resolving issues.

2.4.16. Handle Complaint

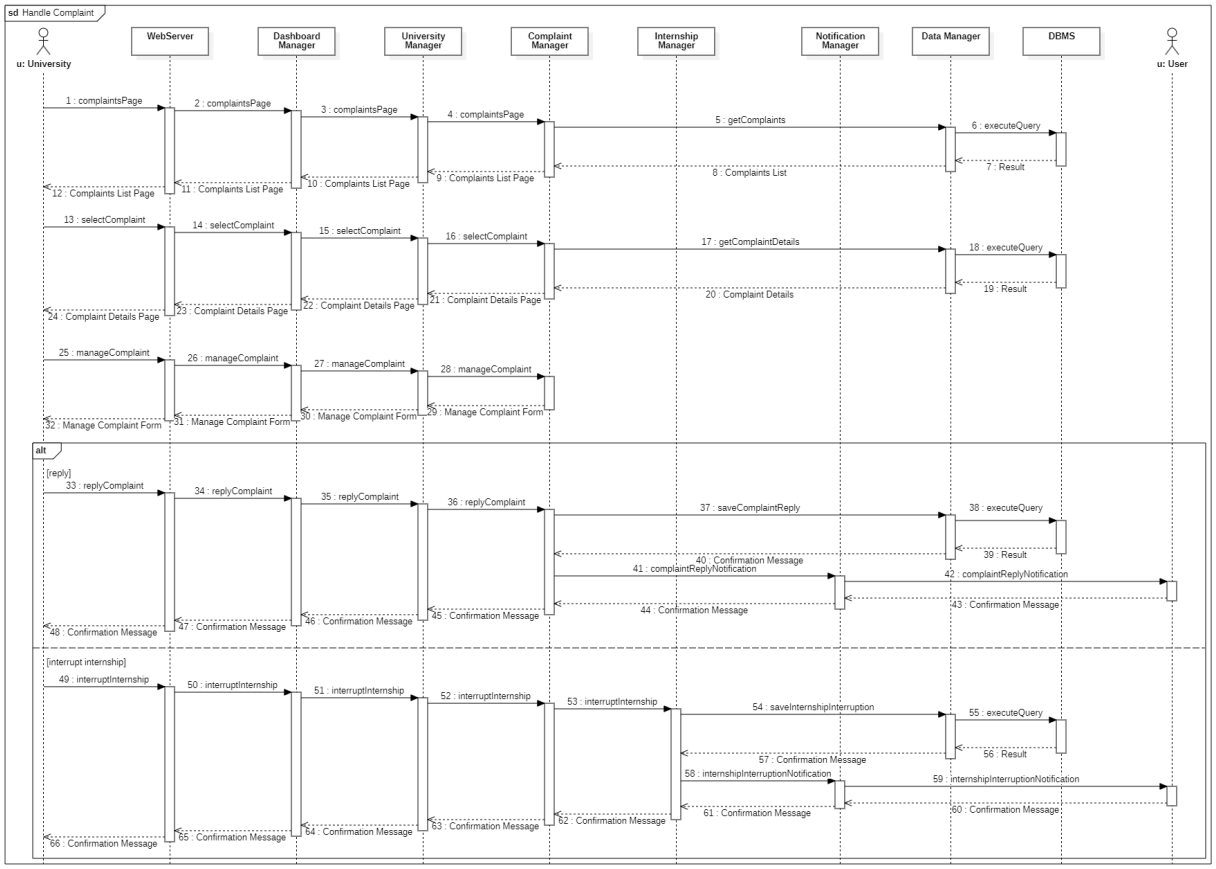


Figure 2.18: Diagram for [UC16]

This sequence diagram details how a university manages a complaint filed by a student. The university accesses the list of complaints and selects one to view its details. After reviewing the information, they can take action by responding to the complaint or deciding to interrupt the internship. The system validates and saves any actions taken. Notifications are sent to the relevant parties, including the student, company, or internship manager, depending on the resolution. This process ensures complaints are addressed efficiently and transparently, maintaining accountability among all stakeholders.

2.5. Component Interface

Registration Manager

- **selectUser**(*Enum userType*)
  Selects the type of user (Student, Company or University).
- **registration**(*String name, String surname, String email, String password, int universityId, String universityName*)
  Registers a new student with the given details.
- **registration**(*String name, String surname, String email, String password, String companyName*)
  Registers a new company with the given details.
- **registration**(*String name, String surname, String email, String password, String universityName*)
  Registers a new university with the given details.
- **checkData**(*String email, String password*)
  Verifies that the provided email and password meet the required validation rules.

Login Manager

- **login**(*String email, String password*)
  Logs in a user with the given email and password.
- **checkData**(*String email, String password*)
  Verifies that the provided email and password meet the required validation rules.

Email Manager

- **verify**(*String email*)
  Verifies the provided email address (e.g., sends a confirmation or validation link).

Internship Manager

- **createInternship**()
  Opens a form used to input details for a new internship.
- **internshipInformation**(*String name, Date deadline, List<Requirement> requirements, String description, String location*) : Internship

Provides comprehensive information about the internship. Returns an **Internship** object containing all relevant details.

- **computeImprovements**(*Internship internship*) : List<String>
  Computes a list of improvements or suggestions to make an internship more appealing. Returns a list of suggested improvements (**List<String>**).
- **updateInternship**()
  Opens a form to update an existing internship.
- **selectInternship**(*String internshipName*) : Internship
  Selects a specific internship by its name. Returns the selected **Internship**.
- **searchInternship**(*Date deadline, List<Requirement> requirements*) : List<Internship>
  Searches for internships that match the given criteria. Returns a list of matching **Internship** objects.
- **interruptInternship**(*String internshipName*)
  Interrupts or terminates an ongoing internship given by its name.
- **checkData**(*String name, Date deadline, List<Requirement> requirements, String description, String location*) : boolean
  Checks the validity or completeness of the internship data. Returns **true** if the data is valid, **false** otherwise.
- **submitApplication**()
  Opens a form to submit an application.
- **applicationInformation**(*String text*) : Application
  Provides comprehensive information about the application. Returns an **Application** containing the application details.
- **selectApplication**(*int applicationId*) : Application
  Selects a particular application for viewing or processing. Returns the selected **Application**.
- **evaluateApplication**()
  Opens a form to evaluate an existing application.
- **submitEvaluation**(*int applicationId, String decision*)
  Submits a final evaluation or decision on an application.

Questionnaire Manager

- **createQuestionnaire**()
  Opens a form to create a new questionnaire.
- **questionnaireInformation**(*List<String> questions*) : Questionnaire
  Provides comprehensive information about the questionnaire. Returns a **Questionnaire** containing the questions details.
- **computeImprovements**(*Questionnaire questionnaire*) : List<String>
  Computes a list of improvements or suggestions to make the questionnaire more appealing. Returns a list of suggested improvements (**List<String>**).
- **updateQuestionnaire**()
  Opens a form to update an existing questionnaire.
- **selectQuestionnaire**(*int questionnaireId*) : Questionnaire
  Selects a particular questionnaire by its ID. Returns the selected **Questionnaire**.
- **answerQuestionnaire**(*List<String> answers*)
  Provides answers to a questionnaire.

Internship Status Manager

- **monitorInternship**(*String internshipName*) : Status
  Provides the status of an internship. Returns the (**Status**) of the selected internship.

Complaint Manager

- **writeComplaint**()
  Opens a form to write a new complaint.
- **complaintInformation**(*String text*) : Complaint
  Provides details of the complaint. Returns a **Complaint** containing its text.
- **complaintsPage**()
  Opens a page listing existing complaints.
- **selectComplaint**(*int id*)
  Selects a particular complaint by its ID.
- **manageComplaint**()
  Opens a form to manage or respond to a complaint.

- **replyComplaint**(*String text*)
  Submits a reply to the selected complaint.
- **interruptInternship**(String internshipName)
  Interrupts an internship based on a complaint or issue.

CV Manager

- **insertCV**()
  Opens a form to insert a new CV.
- **selectCV**() : CV
  Returns the selected **CV**.
- **updateCV**()
  Opens a form to update the CV.
- **CVInformation**(*List<String> cvDetails*) : CV
  Provides comprehensive information about a CV. Returns a **CV** containing all relevant details.
- **checkData**(*CV cv*) : Boolean
  Checks the validity or completeness of the CV data. Returns **true** if the data is valid, **false** otherwise.
- **computeImprovements**(*CV cv*) : List<String>
  Computes a list of improvements or suggestions to make the CV more appealing. Returns a list of suggested improvements (**List<String>**).

Feedback Manager

- **checkData**(*Feedback feedback*) : Boolean
  Checks the validity or completeness of the feedback data. Returns **true** if the data is valid, **false** otherwise.
- **writeFeedback**()
  Opens a form to write a new feedback.
- **feedbackInformation**(*String text*) : Feedback
  Provides details of a particular feedback. Returns a **Feedback** object containing feedback details.

Notification Manager

- **internshipUpdateNotification**(*String text*)
  Sends a notification about an internship update.
- **newQuestionnaireNotification**(*String text*)
  Sends a notification about a new questionnaire.
- **questionnaireUpdateNotification**(*String text*)
  Sends a notification about a questionnaire update.
- **questionnaireAnswerNotification**(*String text*)
  Sends a notification when a questionnaire is answered.
- **newComplaintNotification**(*String text*)
  Sends a notification when a new complaint is submitted.
- **complaintReplyNotification**(*String text*)
  Sends a notification about a complaint reply.
- **internshipInterruptionNotification**(*String text*)
  Sends a notification that an internship has been interrupted.
- **newApplicationNotification**(*String text*)
  Sends a notification about a new application.
- **evaluateApplicationNotification**(*String text*)
  Sends a notification about the evaluation of an application.

Data Manager

- **checkEmail**(*String email*) : Boolean
  Checks whether the given email exists or meets specific criteria.
- **saveStudent**(*Student student*)
  Saves a new student record.
- **checkData**(*String email, String password*) : Boolean
  Checks the validity of the email and password.
- **saveInternship**(*Internship internship*)
  Saves a new internship record.
- **getInternshipDetails**(*String internshipName*)
  Retrieves details of an internship by its name.

- **getStudentsFromInternship**(*String internshipName*) : List<Student>
  Retrieves a list of students associated with a given internship.
- **getInternships**(*Date deadline, List<Requirement> requirements*) : List<Internship>
  Retrieves internships matching certain criteria.
- **saveQuestionnaire**(*Questionnaire questionnaire*)
  Saves a new questionnaire.
- **getQuestionnaireDetails**(*int questionnaireId*)
  Retrieves details of a questionnaire by its ID.
- **getStudentFromQuestionnaire**(*int questionnaireId*) : List<Student>
  Retrieves a list of students linked to a given questionnaire.
- **saveQuestionnaireAnswers**(*String studentName, List<String> answers*)
  Saves the answers to a questionnaire for a specific student.
- **getInternshipStatus**(*String internshipName*) : Status
  Retrieves the status of an internship.
- **saveComplaint**(*Complaint complaint*)
  Saves a new complaint record.
- **getUniversityFromStudent**(*String studentName*) : University
  Retrieves the university linked to a given student.
- **getComplaints**(*String universityName*) : List<Complaint>
  Retrieves a list of complaints associated with a specific university.
- **getComplaintDetails**(*int complaintId*) : Complaint
  Retrieves details of a complaint by its ID.
- **saveComplaintReply**(*String text, String recipient, String author*)
  Saves a reply to a complaint, including the recipient and author.
- **saveInternshipInterruption**(*String internshipName, String reason*)
  Logs an internship interruption event.
- **saveCV**()
  Saves a CV record.
- **saveFeedback**()
  Saves a feedback entry.

- **saveApplication**()
  Saves a new application record.
- **getApplicationDetails**(*int applicationId*)
  Retrieves information about an application.
- **saveEvaluation**()
  Saves an evaluation or decision about an application.

2.6. Selected architectural styles and patterns

This application will be developed using a 3-Tier architecture, which organizes the software into three logical and physical layers. Each tier operates on its own infrastructure, reducing system complexity, and increasing both flexibility and scalability. Moreover, each tier can be developed in parallel by separate teams and can be updated or scaled independently without impacting the other tiers.

3-Tier Architecture

- **Presentation Tier**:
  This top-level tier directly interacts with the Users, collecting input and displaying outputs generated by the lower tiers.
- **Application Tier**:
  This middle-tier processes requests arriving from the Presentation Tier, performing any required computation. It may also query or update data from the Data Tier, then combine and process the returned data to complete the tasks requested by Users. As it may need to handle multiple concurrent requests, this tier must guarantee data security and integrity.
- **Data Tier**:
  The lowest tier, which manages, retrieves, and stores data. It provides interfaces so that the Application Tier can effectively work with the underlying data.

Client-Server Interactions

The system largely follows a client-server architecture:

- The **client** is the front-end user interface, serving as the link between the end users and the system.
- The **server** is the back-end platform, receiving user requests, performing necessary computations, and returning the results.

In a pure client-server model, the client typically initiates requests, and the server—acting as a passive element—processes and returns the requested data or responses. However, some features require the server to be proactive, such as sending notifications or interacting with external tools without a direct client request. In these cases, the server adopts a more event-driven or active role.

Model-View-Controller (MVC) Pattern

The application's internal structure follows the MVC pattern, a design principle that divides the software into three interconnected components:

1. **Model**:
   Contains methods for handling data—saving, retrieving, and manipulating information from the database.

2. **View**:
   Manages how data is visually represented to the end user. It specifies the layout and user interface elements displayed.

3. **Controller**:
   Serves as the link between the View and the Model. It intercepts user interactions (e.g., button clicks) from the View, executes the necessary operations, and ensures the appropriate data and responses flow between Model and View.

By combining a 3-Tier architecture, a client-server approach, and the MVC pattern, the system ensures modularity, maintainability, and scalability, all while providing a coherent separation of concerns and clear communication between components.

2.7. Other Design Decisions

2.7.1. Availability

The introduction of load balancing and replication mechanisms significantly enhances the reliability and availability of the S&C system. Load balancing optimizes resource utilization by distributing incoming requests evenly across servers, thereby preventing performance bottlenecks. Concurrently, replication ensures fault tolerance by duplicating essential data and services. This redundancy reduces the impact of potential failures and enables the system to maintain consistent data management and service availability even under challenging conditions.

2.7.2. Scalability

A microservices architecture is intentionally designed to be both independently deployable and inherently scalable. Each microservice can be deployed and updated autonomously, thereby preventing disruptions to the entire system when modifications are made. Furthermore, microservices empower the S&C platform to handle increasing demands by efficiently scaling specific services as needed.

2.7.3. Notification Handling

Notification management is crucial in the S&C system, affecting every stage of the internship life cycle, from creation to ongoing tracking and completion. Notifications are orchestrated to reach users upon login, or, if a user is already logged in, immediately as they become available. This approach ensures that students, companies, and universities receive timely updates on key events, such as new internship postings, interview schedulings, or feedback requests.

2.7.4. Ease of Deployment

Adopting a microservices strategy also provides significant advantages in terms of deployment. Individual services can be updated independently, allowing direct rollouts without impacting the entire system. In case any issues arise, the microservices approach simplifies diagnosing, isolating, and correcting problems—no need to halt the full platform. This deployment flexibility not only accelerates the release of updates but also boosts resilience and agility, enabling efficient troubleshooting and maintenance.

2.7.5. Data Storage

To streamline operations and simplify data management, the S&C system employs a unified DBMS that contains information about students, companies, universities, internships, and any additional data relevant to matching or recommendation processes. Centralizing this data reduces the time and complexity associated with retrieval and updates, leveraging the interrelated nature of these components.

3 | User Interface Design

This section describes the user interface of the S&C system, providing an overview of the various pages that make up the website. The interfaces are designed to ensure ease of use and a comfortable experience for all users. Color schemes and design choices have been carefully selected to provide a clear, intuitive, and visually appealing platform, avoiding unnecessary complexity and enhancing user satisfaction.

The desktop browser version is emphasized, reflecting the system's primary goal of facilitating internship discovery. However, equivalent pages will be adapted for the mobile version to ensure an effortless user experience through appropriate rescaling and interface adjustments.

As outlined in the RASD, the design mockups presented here serve as a foundational representation. They are subject to refinement and optimization as the system evolves based on testing and user feedback.

3.1. Overview

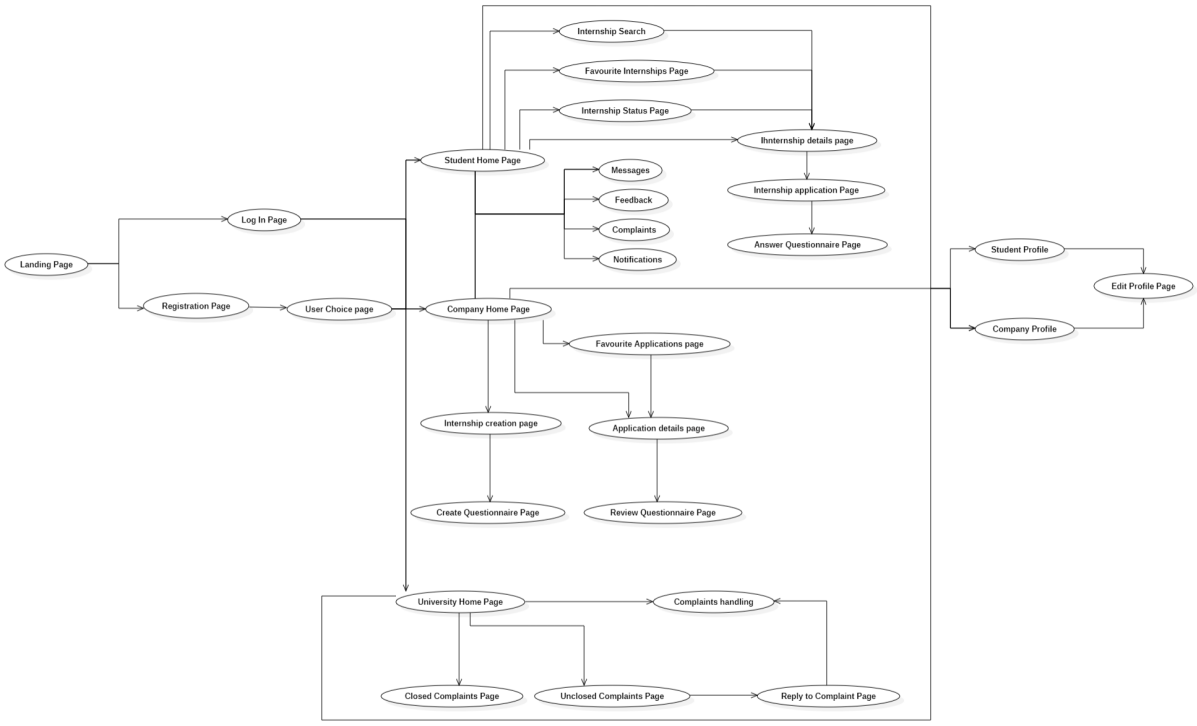


Figure 3.1: Enter Caption

The provided graph offers a comprehensive overview of the S&C system's pages, illustrating their connections and navigation pathways for users. In the graph, interfaces that serve the same function for multiple users are shown only once, avoiding repetition and preventing confusion about their shared use. Communication between entities is represented separately, with arrows indicating their interactions. This approach is designed to enhance the clarity and ease of comprehension of the graph. Each page is described in the section below, providing further details on its functions and user interactions.

3.2. Interfaces

This initial illustration depicts the platform's homepage, where users can choose to log in or sign up. [Figure 3.2]

Figure 3.2: Landing Page

After a user decides to create an account, they will be asked their role as either a student, company or a university. [Figure 3.3]

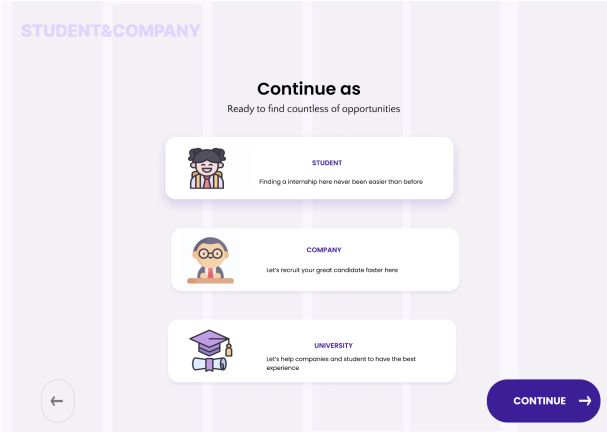


Figure 3.3: Choice page

Depending on the role decided by the user, they will be prompted to input various information that can vary based on the specific role. [Figures 3.5, 3.4, 3.6]

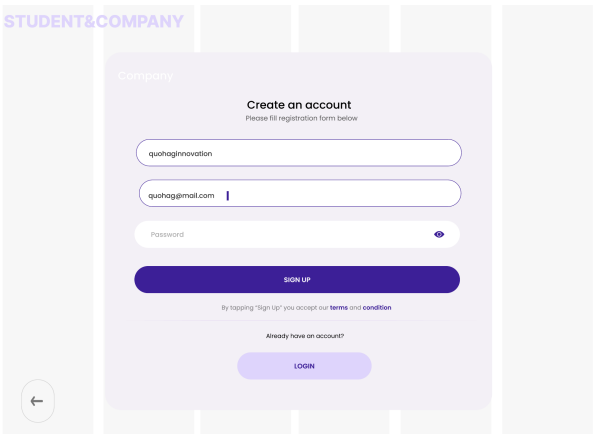


Figure 3.4: Create account for Companies

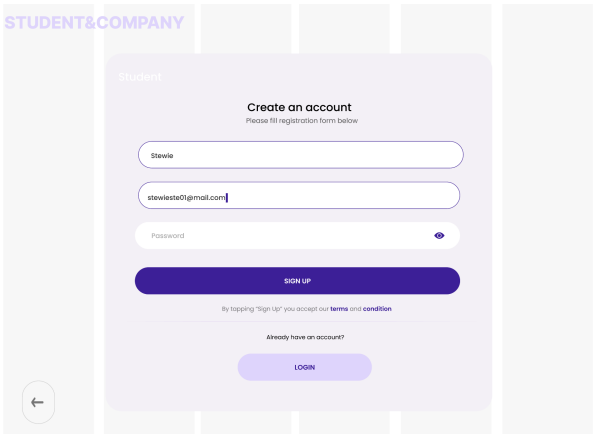


Figure 3.5: Create account for Students

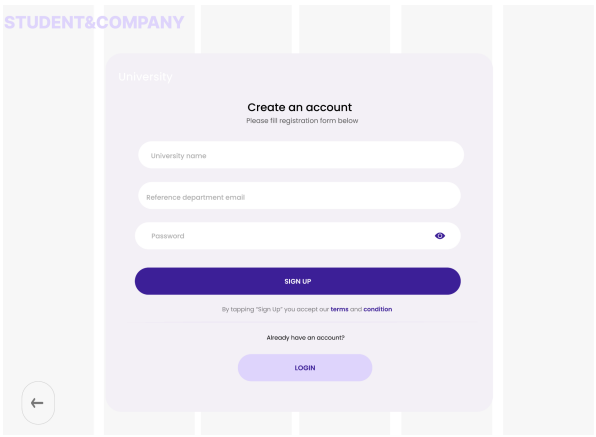


Figure 3.6: Create account for Universities

Users with an existing account can access the platform by clicking the appropriate button on the homepage, which directs them to the login page. From there, they can log in using

their username and password or by signing in with their Google or Facebook account. If a user mistakenly selects the login option without having an account, they can simply click the "Create Account" button to navigate to the registration page. [Figures 3.7, 3.8, 3.9]

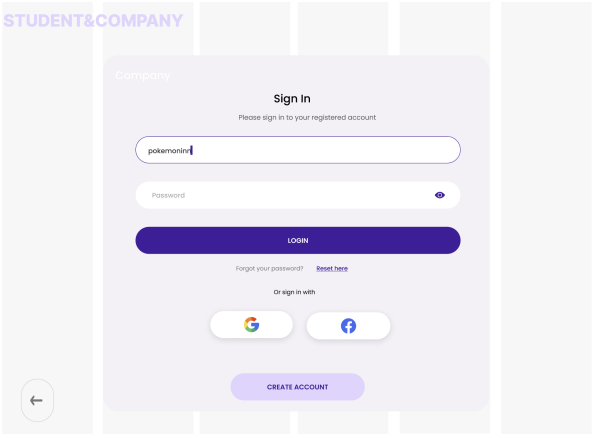


Figure 3.7: Company Sign in

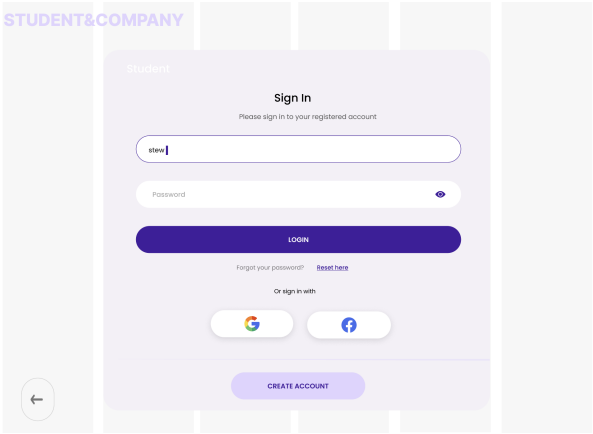


Figure 3.8: Student Sign in

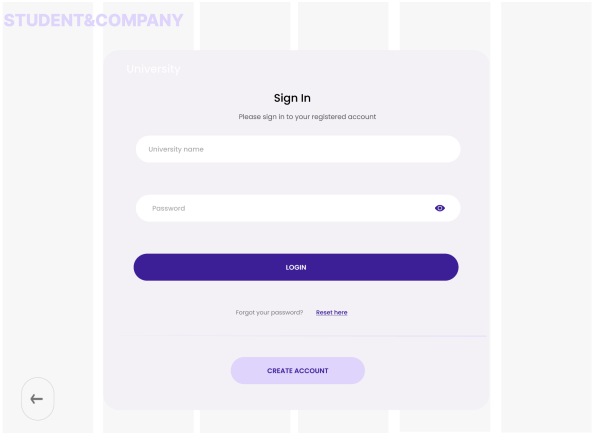


Figure 3.9: University Sign in

3.2.1. Company Interfaces

The company's Dashboard [Figure 3.10] serves as the central hub for all platform operations. From this main page, the company can easily access essential information, including its profile, notifications, proposed internships, and applications submitted by students. Additionally, it allows the company to communicate with students after the internship application process, as well as provide feedback or submit complaints.

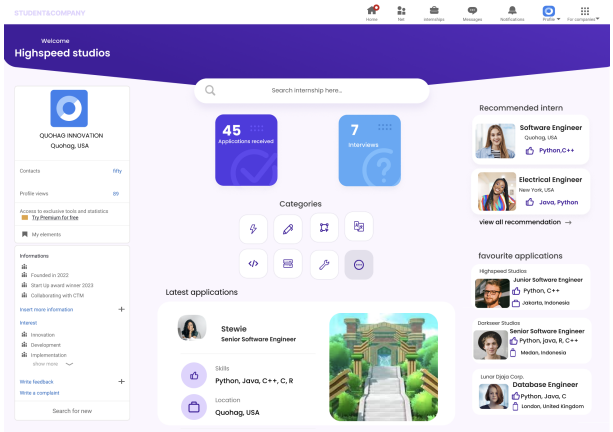


Figure 3.10: Company Home page

Below, we can see the structure of a company's personal profile. It includes key details such as location, field of activity, objectives, and a dedicated section for managing internships. To edit the profile, the company simply clicks on the desired section and follows the platform's guidance. Additionally, the platform offers suggestions to enhance the provided information, making it more appealing to potential viewers[Figures 3.11, 3.12].

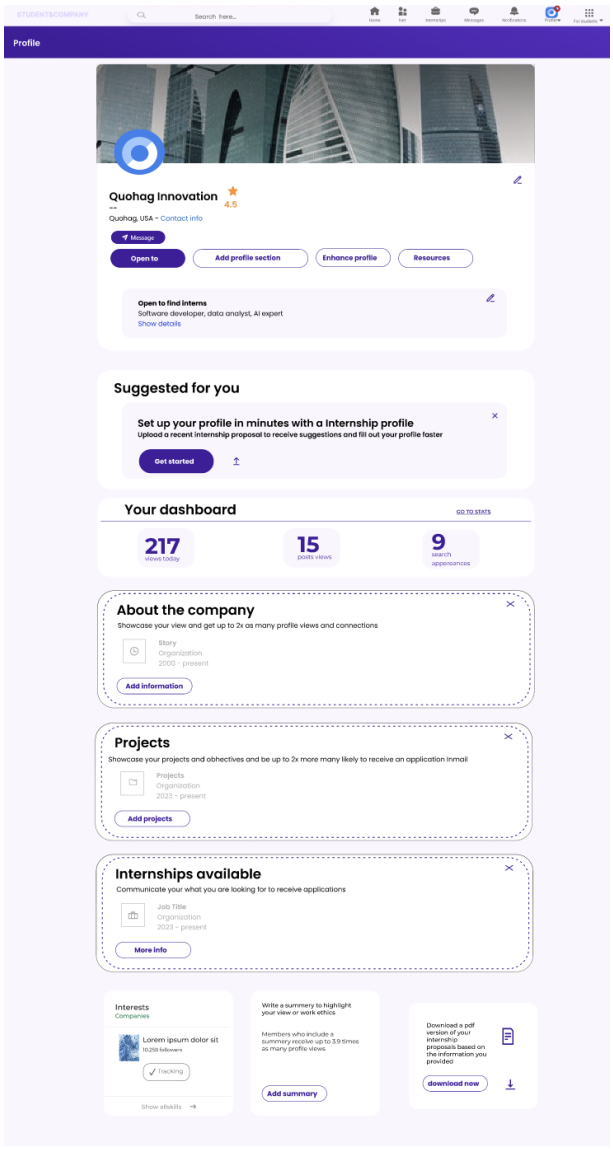


Figure 3.11: Company Personal profile

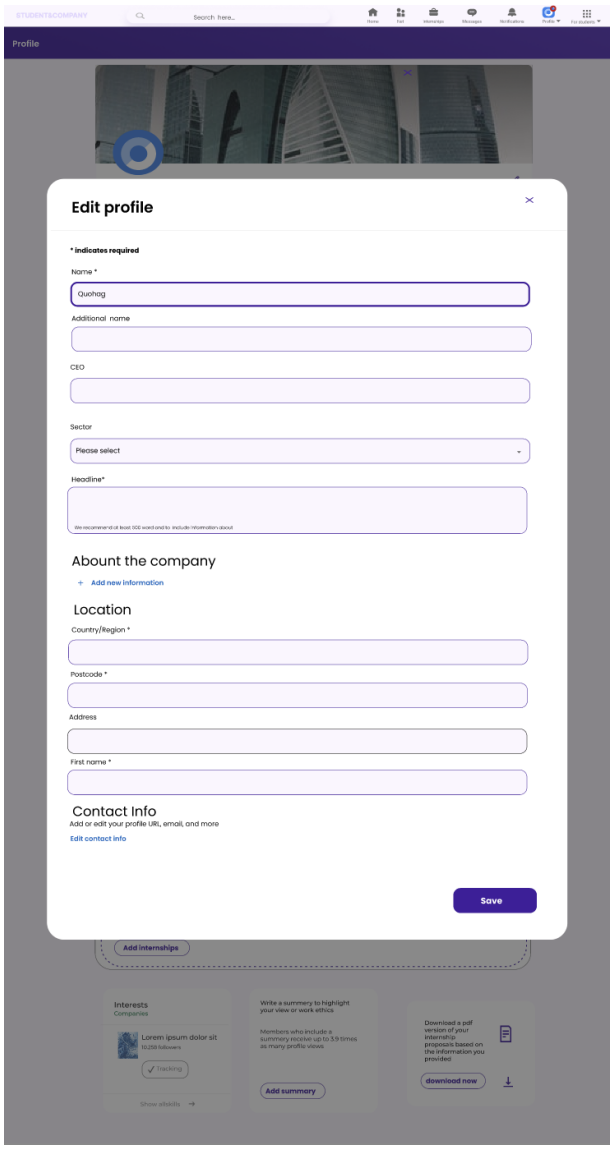


Figure 3.12: Company - Managing account

To edit internship information, the company can click on the designated button within its personal profile. From there, it can manage the internship details while also benefiting from suggestion.s provided by the platform [Figure 3.13].

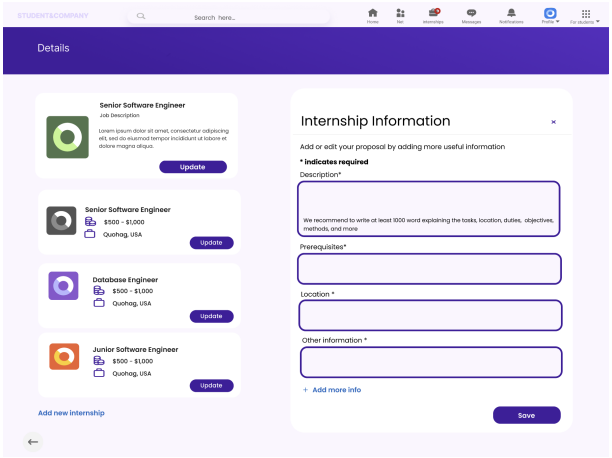


Figure 3.13: Internships management

On the homepage, a company can access the applications section to view all submitted applications. By selecting a specific application, the company can review its details in full, evaluate it positively or negatively, or save it for easier access later [Figure 3.14]

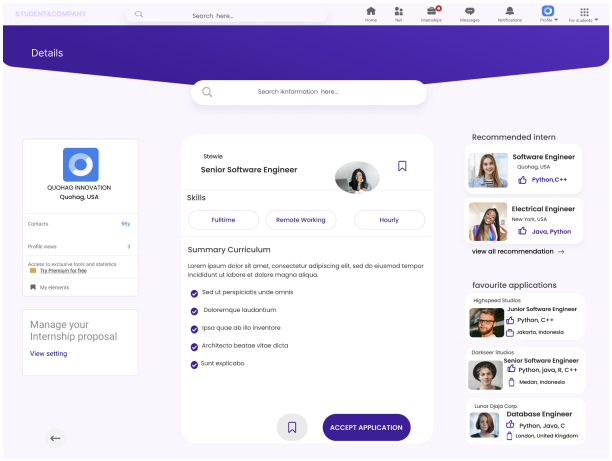


Figure 3.14: Applications review

After positively evaluating an application, the company can interview the applicant using a structured questionnaire tailored for each internship proposal. This questionnaire is created by the company and can be enhanced with suggestions provided by the platform [Figure 3.15].

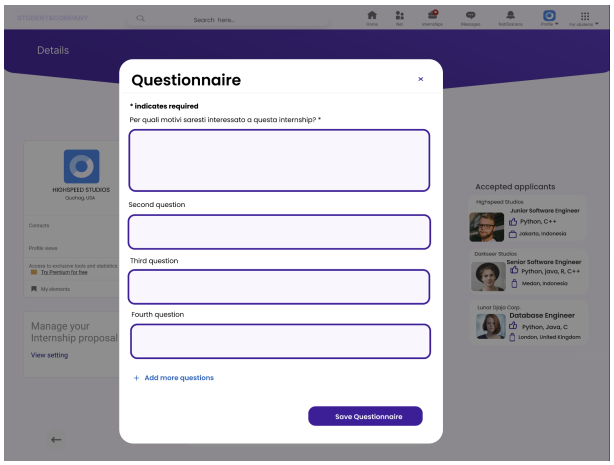


Figure 3.15: Interview and questionnaire creation

Once a student has completed the questionnaire, the company reviews the submitted answers and evaluates the application. The company can either decline it by providing a negative evaluation or accept it, officially initiating the internship [Figure 3.16].

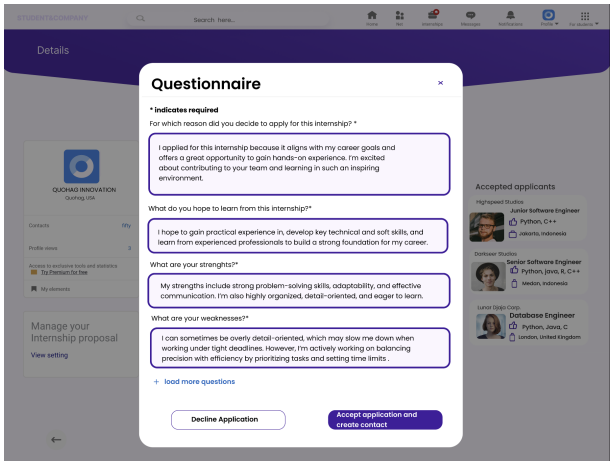


Figure 3.16: Interview evaluation

3.2.2. Student Interfaces

The student dashboard [Figure 3.17] is the centre of all possible operations on the platform. From this main page, the student can have access to all essential information, including their personal profile, notifications, available internships, and applications submitted to companies. In addition, it allows the student to communicate with the company after the internship application process is over, as well as provide feedback or submit complaints.

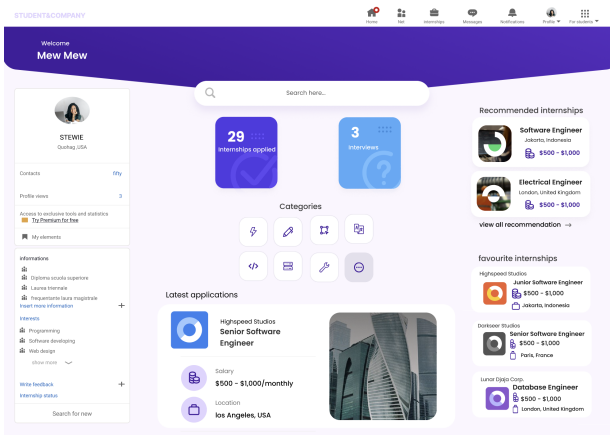


Figure 3.17: Student Home page

In the personal profile section, students can input their personal details as well as information relevant to creating a CV for internship applications. Students have the option to enter this information independently or follow platform suggestions and tips to enhance their CV, making it more appealing to potential employers. The information is organized into sections and will be visible to companies later on.

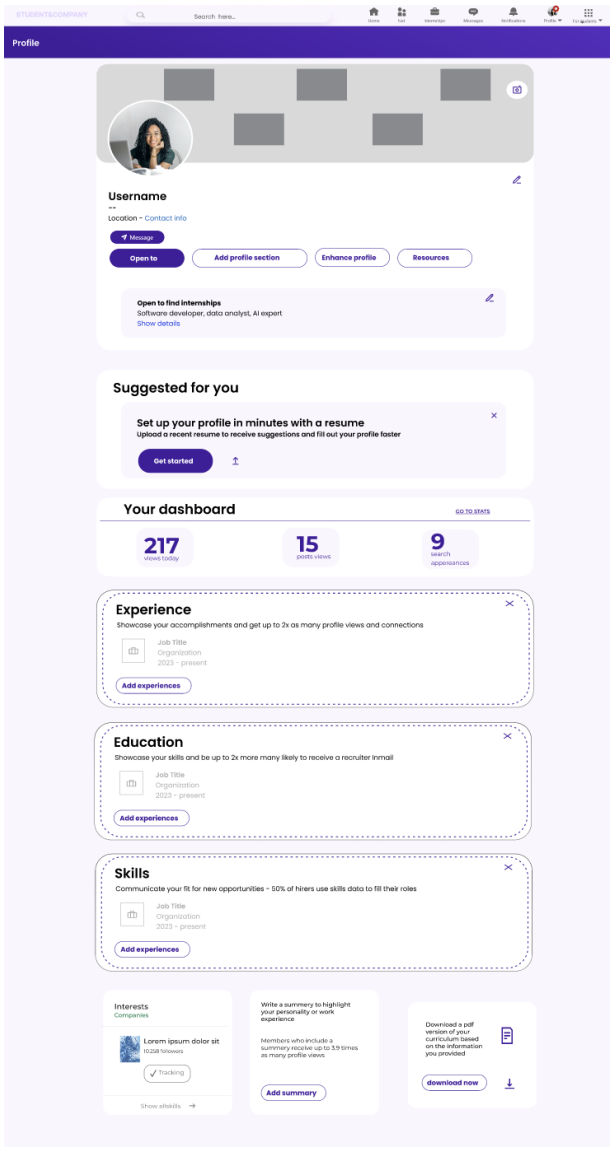


Figure 3.18: Student Personal profile

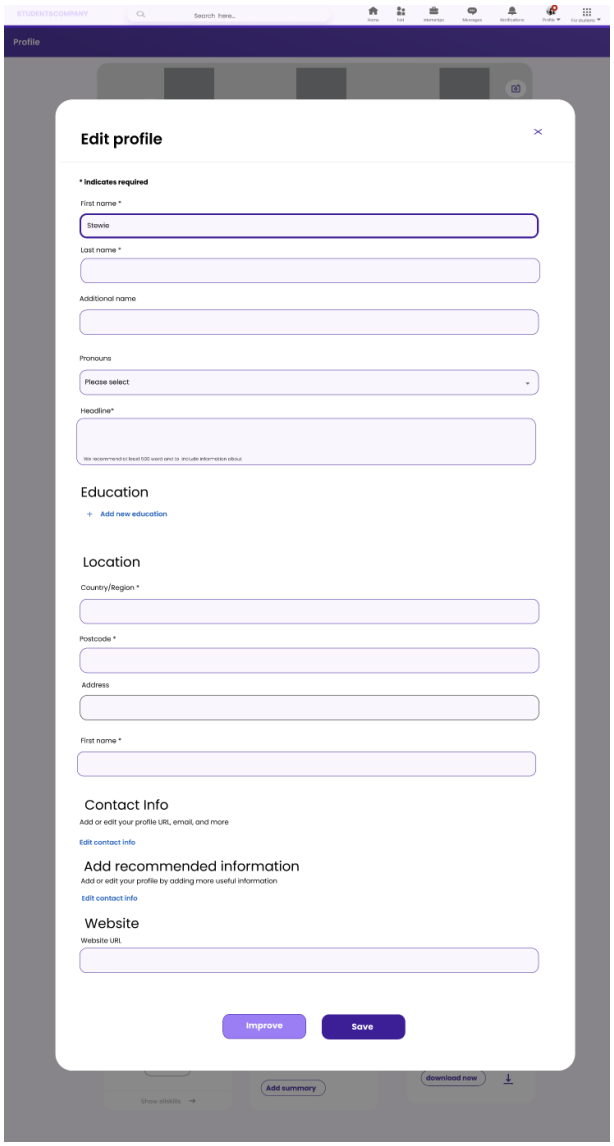


Figure 3.19: Student Profile management

In the search section, students can explore all available internships on the platform. To make the search more relevant and tailored to their personal interests, several filters are provided, allowing students to select internships based on criteria they specify. The internships will be displayed in a list, showing only key details. By clicking on a specific internship, students can view all the information and find the option to apply if they're interested in that particular opportunity.

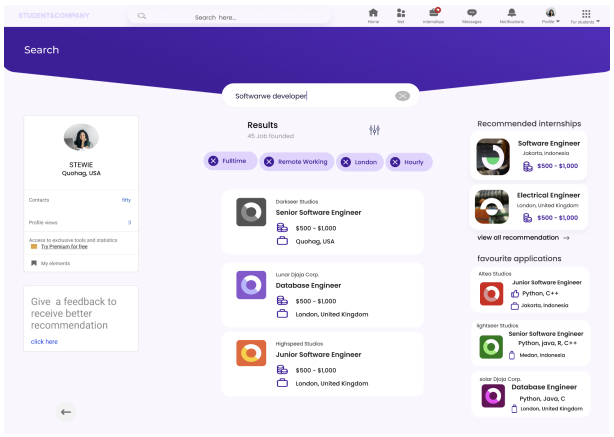


Figure 3.20: Internships lookup

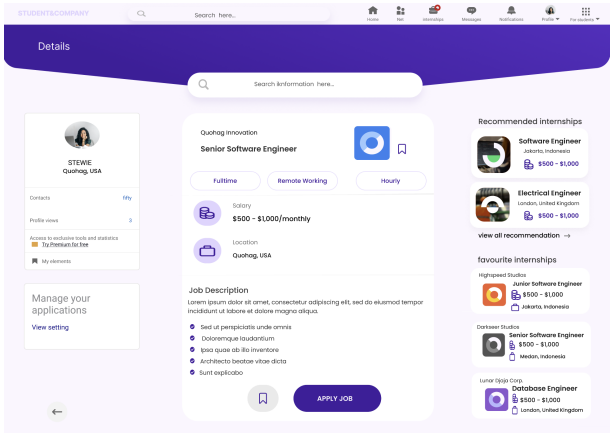


Figure 3.21: Internship details

After clicking the "Apply" button, the student can proceed with the internship application. To complete the application, the student must either upload their CV or consent to share their personal profile information with the company. Additionally, the student will need to provide other personal details, such as their name, surname, and email. Once all the required information is submitted, the student can send the application.

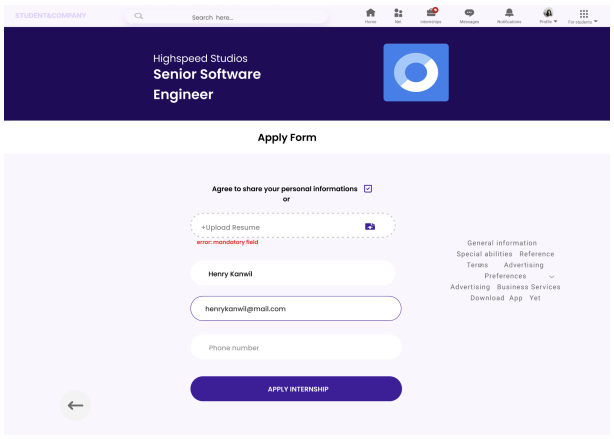


Figure 3.22: Application form

Once an application is submitted and a positive evaluation is received, the interview section will become available on the internship page. The interview is structured as a questionnaire, which is pre-prepared by the company for each specific internship. The student must answer all questions in the questionnaire before submitting it back to the company for final evaluation. Based on this evaluation, the company will make the final decision on whether to proceed with the internship.

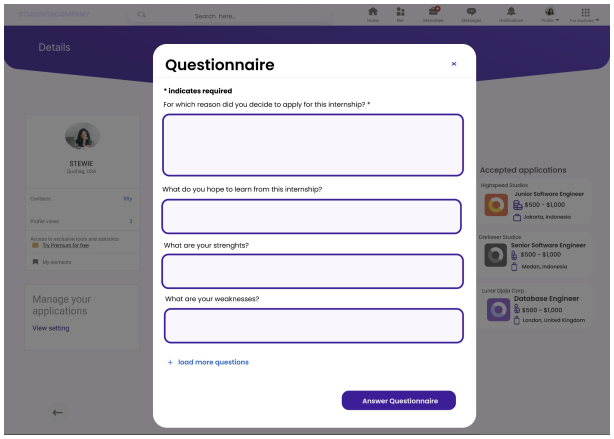


Figure 3.23: Interview reply

Throughout the application process, the student will be able to monitor the status of their application and receive notifications whenever there is a change. The status will update based on different stages, including: application sent, awaiting review, evaluated negatively, evaluated positively, interview ready, interview submitted, and interview evaluated.

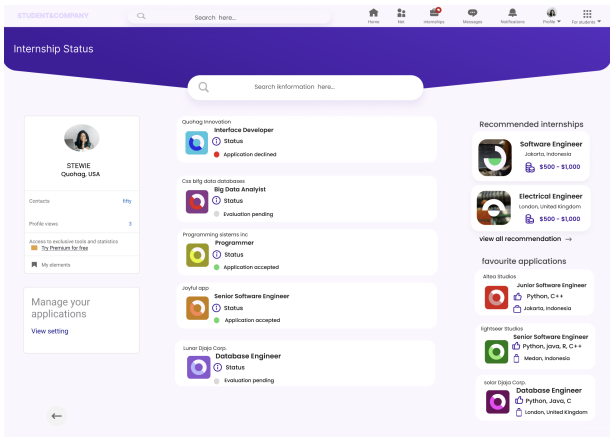


Figure 3.24: Application status

3.2.3. University Interfaces

The university's Dashboard [Figure 3.25] serves as the central hub for all platform operations. From this main page, the university can easily access essential information, including its notifications the personal profiles of the students enrolled, and all complaints sent by both students and companies. Additionally, it allows the university to communicate with students and companies after a complaint has been submitted, allowing for a better resolution of the problems.

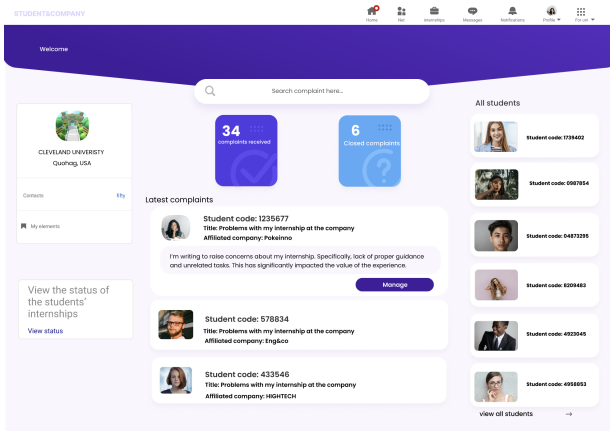


Figure 3.25: University Home page

By clicking on a complaint, the university can respond directly by sending a reply message to the user who submitted it [Figure 3.26].

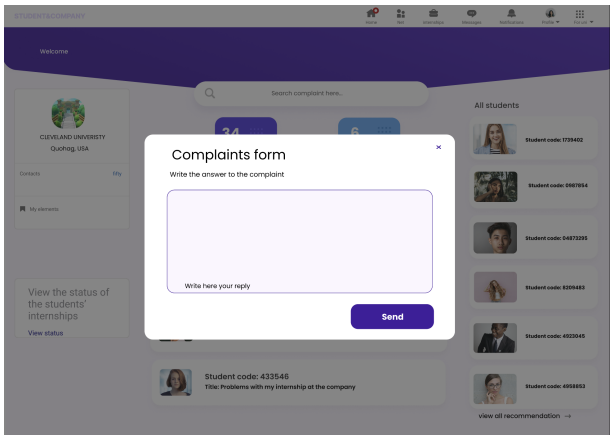


Figure 3.26: Complaints visualization

Complaints are managed through a chat between the university and the involved users, allowing the university to act as an intermediary. This enables the university to assist in resolving the issue or, if necessary, proceed with the conclusion of the internship [Figure 3.27]

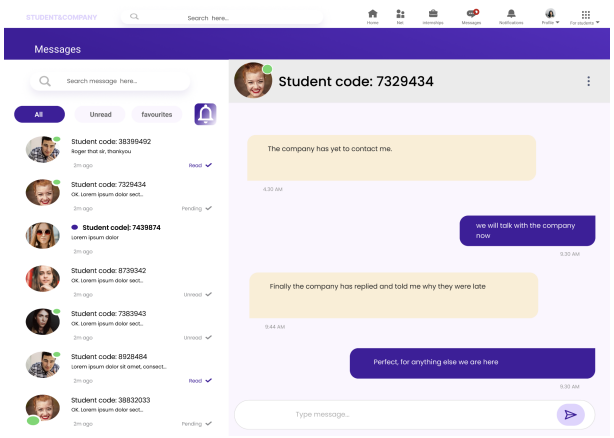


Figure 3.27: Complaints handling

3.2.4. Common Interfaces

By clicking on the designated button, users can view all notifications related to important updates, such as changes in internship details, application status, or new internship opportunities that may be of interest to students [Figure 3.28].

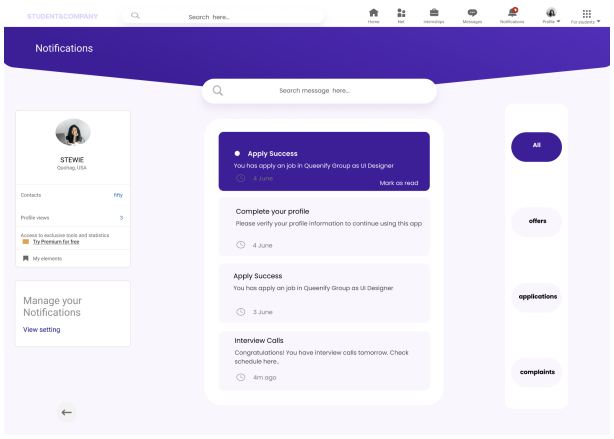


Figure 3.28: Notifications

Users can communicate with each other through a messaging system provided by the platform. Students and companies can interact to discuss internship details, while the university can also use the system to address any complaints or concerns [Figure 3.29].

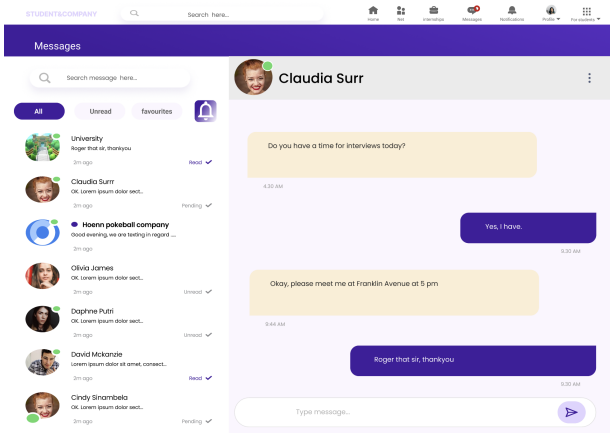


Figure 3.29: Messages

Students and companies can also receive feedback requests about their general experience with the platform, provide their own feedback, or submit complaints regarding the progress of the internship once it has started [Figure 3.30, 3.31]

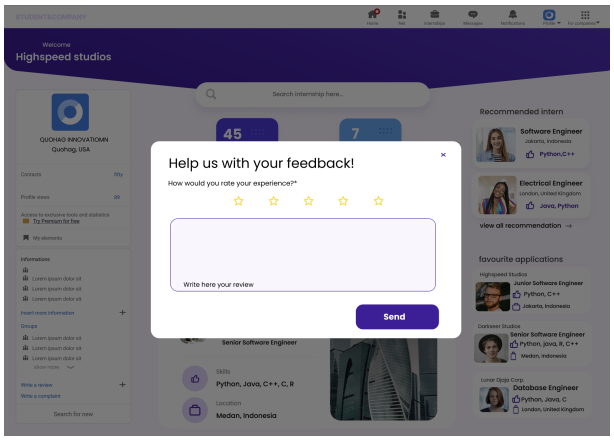


Figure 3.30: Feedback Request

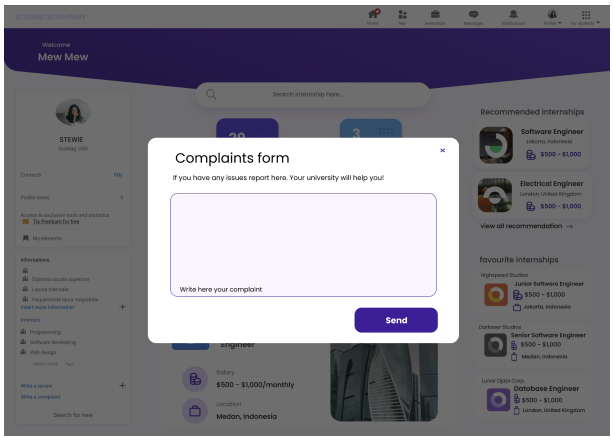


Figure 3.31: Complaints sending

4 | Requirements Traceability

Registration Manager

[R1] The system allows unregistered users to sign up

Login Manager

[R2] The system allows users to log in

Feedback Manager

[R30] The system allows students and companies to provide feedback

[R31] The system generated a feedback request to send to students and companies

Message Manager

[R25] The system asks users for permission to send messages

[R28] The system allows users to retrieve all messages sent

Dashboard Manager

[R6] The system allows companies to view a student's personal profile

[R9] The system allows a student to see a company's profile

Student Manager

[R5] The system allows users to update their personal profiles

[R7] The system allows users to retrieve all modified information

CV Manager

[R3] The system allows students to insert personal information for the creation of the CV

[R8] The system takes information from the student's personal profile

Company Manager

[R5] The system allows users to update their personal profiles

[R7] The system allows users to retrieve all modified information

Internship Manager

[R4] The system allows companies to insert information for the creation of the internship

[R11] The system allows students to accept the outcome of the evaluation

[R13] The system displays all the available internships

[R14] The system displays all the information of each specific internship

[R15] The system displays specific internships that better suit a student based on the information provided

[R16] The system allows students to visualize information about the internship

[R17] The system allows a registered student to select an internship to apply to

[R18] The system allows users to apply to an internship

[R19] The system allows companies to set up evaluation criteria

Internship Status Manager

[R10] The system changes the status of the evaluation after updates in the application process

[R12] The system allows companies to evaluate students'applications

[R19] The system allows companies to set up evaluation criteria

University Manager

[R5] The system allows users to update their personal profiles

[R7] The system allows users to retrieve all modified information

Complaint Manager

[R20] The system allows universities to set up criteria for internship's conclusion

[R21] The system allows users to name other parties involved in the complaint

[R22] The system displays the complaints sent by both students and companies

[R23] The system updates the status of a complaint

[R24] The system allows universities to close an internship

Notification Manager

[R26] The system generates a message of confirmation after an operation is done

[R27] The system generates a message once the status of the internship changes

Recommendation Manager

[R29] The system generates suggestions based on the information taken in the platform

5 | Integration, Implementation and Test Plan

5.1. Overview and Implementation

This final chapter outlines the system's implementation process, along with the strategies for integration and testing. A Bottom-Up approach will be used as the guiding methodology.

This strategy involves starting from the lower levels of the "uses" hierarchy, beginning with the development of smaller, independent modules that do not depend on other components to function. Each of these modules will be accompanied by a driver specifically created for testing purposes. Once a module has been implemented and verified, it will be integrated into the system, replacing its corresponding driver. However, a new driver will be developed for the next module to ensure continuous testing.

By following this approach, several functional subsystems will be progressively developed and tested before being combined into the final system. The Bottom-Up method supports incremental integration, allowing for early detection of bugs and errors since testing occurs in smaller, manageable segments as each module is completed. Additionally, this strategy enables parallel development, as independent teams can work simultaneously on different system components.

5.2. Features Identification

[F1] Login and Registration Features: This are the features needed by all of the user of the platform: Students, Companies and Universities. These features play a pivotal role like ensuring proper data validation and secure storage of registration details, managing the verification process for universities, implementing workflows for account recovery and activation, ensuring system security using secure login mechanisms and as such will be the first to be implemented.

[F2] Creation Feature: This set of features includes tools for creating and enhancing content, such as filling out questionnaires, submitting applications, and improving CVs.

[F3] View Features: This set of features includes all the action of visualization included in the platform such as looking at internships. They need the corresponding **F3**, **F4** features in order to work and are essential for other features

[F4] Search Features: These features cover all platform operations related to searching for content, such as finding internships or candidates.

[F5] Internship Features: These features provide access to internship related information, such as internship details and progress tracking. These also include joining and applying for an internship as well as replying to the questionnaires in case of acceptance

[F6] Evaluation Features: This category includes the operations of evaluating internship applications and interview performances.

[F7] Message Features: These features enable controlled messaging between all platform users.

[F8] Recommendation Features: This set involves personalized recommendations for students and companies, suggesting internships for students and suitable candidates for companies.

[F9] Complaints Features: These features allow users to submit complaints to the university once an internship has started, facilitating issue resolution.

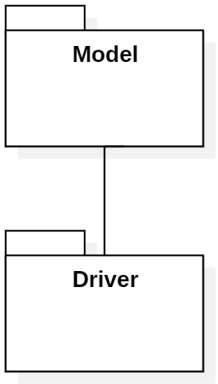
[F10] Feedback Features: This set covers feedback-related actions, including both submitting feedback and responding to feedback requests.

[F11] Notification Features: These features handle the delivery of notifications regarding platform events and updates. They will be the last features implemented

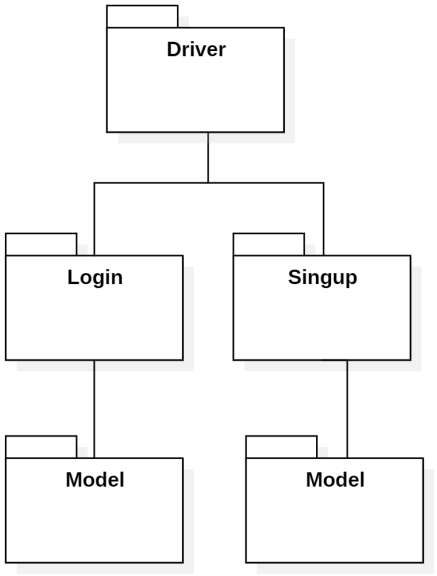
5.3. Overview and Implementation

Component integration and system testing should begin as soon as the DBMS and host server are prepared. Connections to the Mailing System and External Tools are not immediately required but will be essential once their respective features are incorporated. As previously mentioned, the integration process will follow a bottom-up approach.

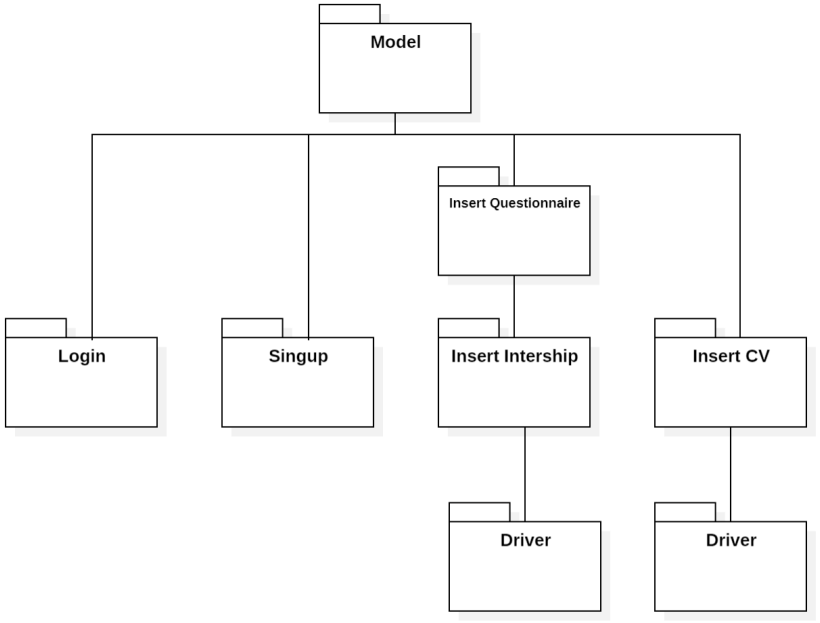
We will start our model that will be tested alongside a driver.



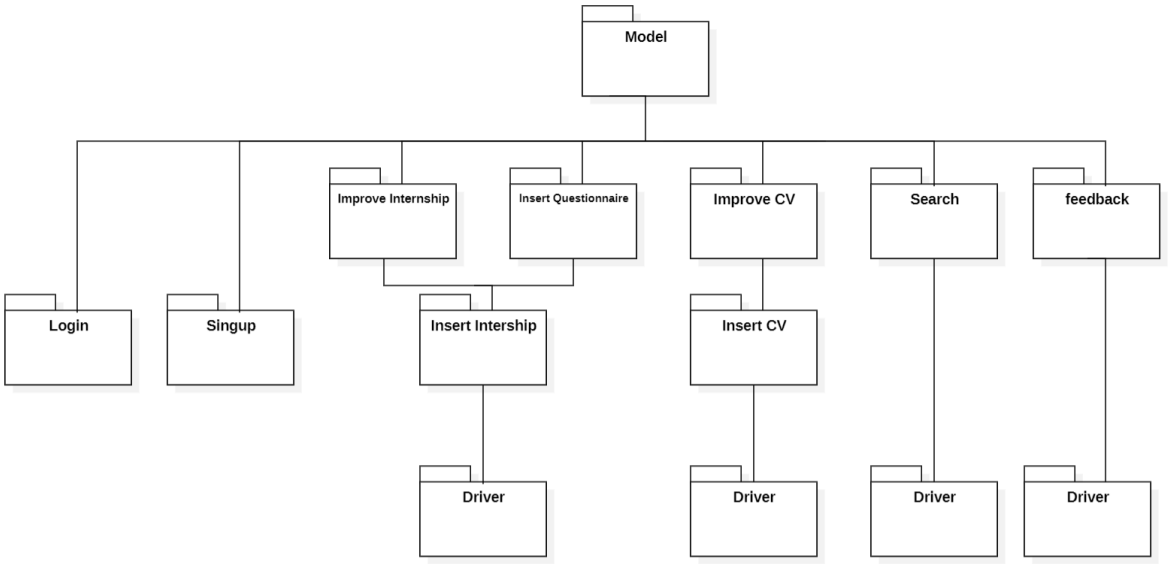
Once integration begins, the initial features to be implemented will be Log-in and Sign-up.



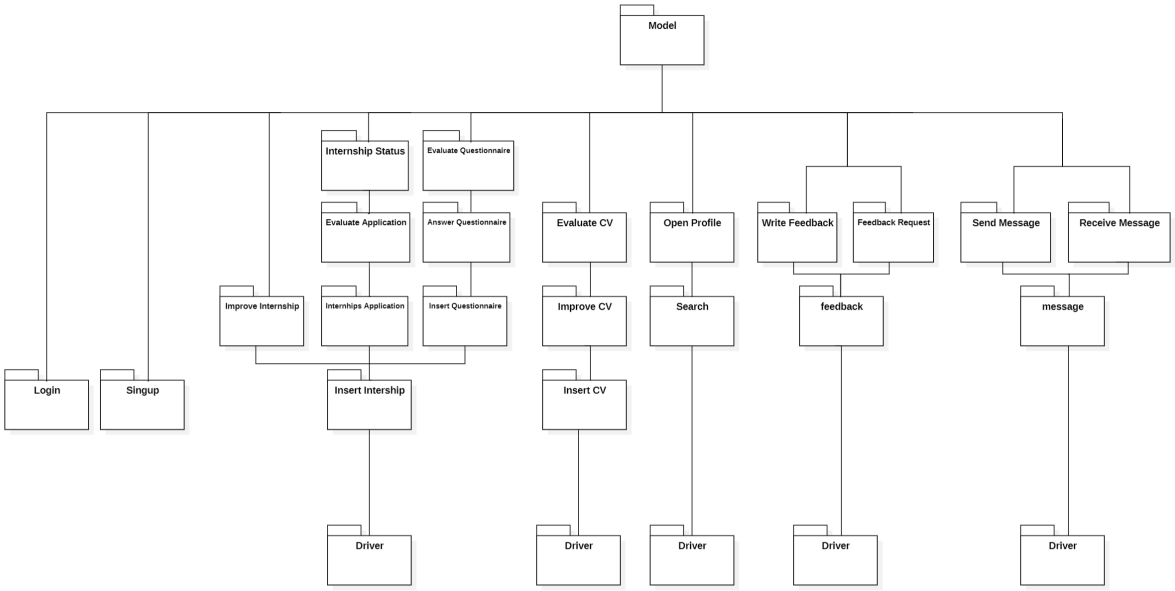
After testing the Log-in and Sign-up features, the remaining identified features will be added gradually, step by step.



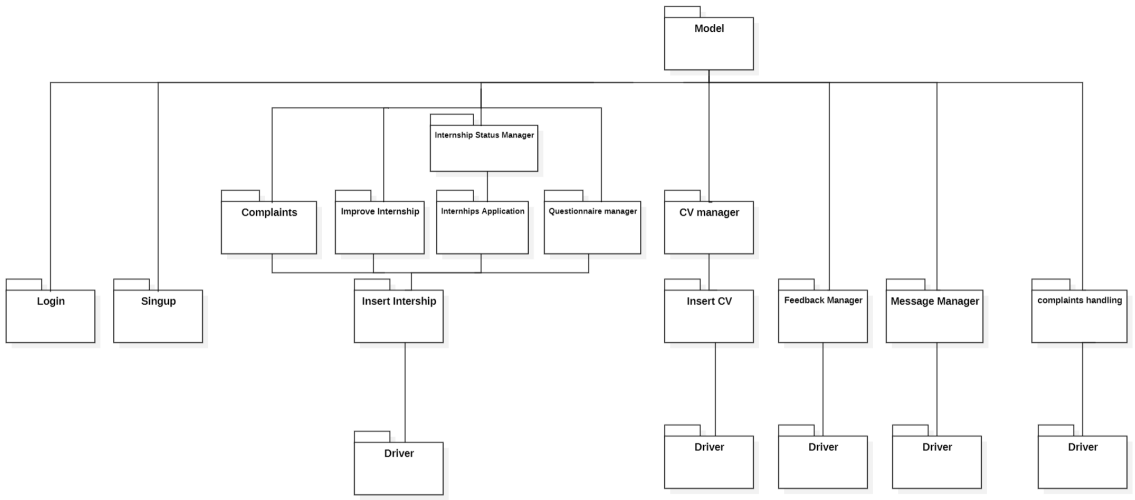
In the following, additional identified features will progressively be integrated, continuing the step-by-step approach.



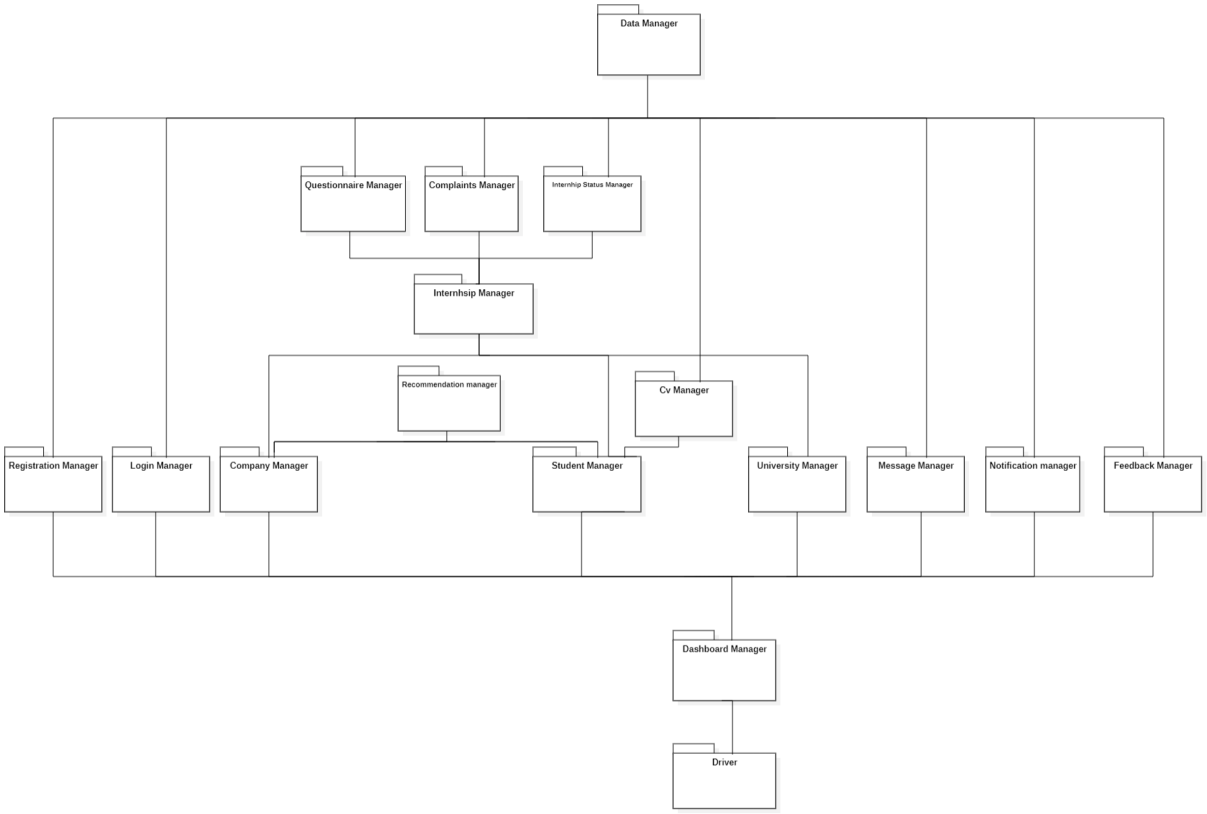
Finally, the last set of features will be added, completing the integration process.



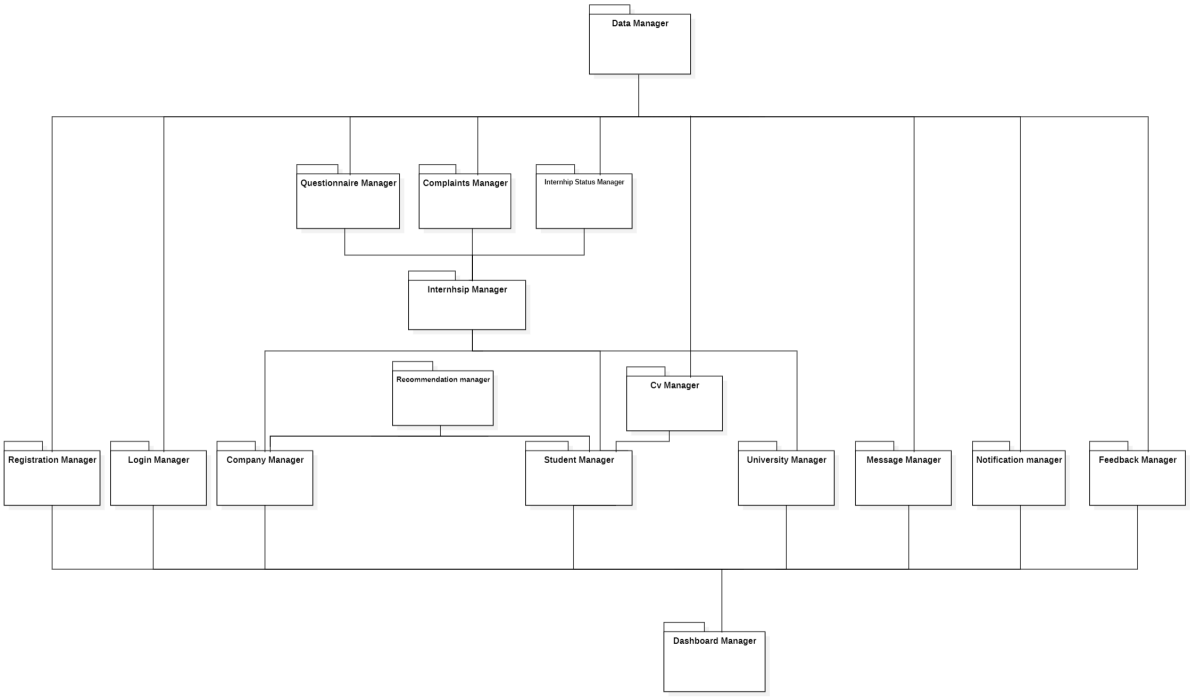
For simplicity, some of the previously integrated components will be consolidated within the Managers component. They will be grouped together to make the system's graph more comprehensible.



The final component to be integrated is the Dashboard Manager, which is essential for ensuring the proper workflow of the user interface.



After the removal of the Dashboard Manager's driver, the final system configuration is as follows:



5.4. Integration Strategy

Thorough testing must be conducted during and after the development phase to ensure that the S&C performs as expected. The testing aims to validate that the components and overall architecture fulfil the functional and non-functional requirements specified in the RASD document.

Component-Level Testing

During development, individual components must be tested to validate their functionality. Once a certain component is integrated into the system, it must be checked using stubs and drivers to ensure that the integrated system follows the correct workflow. Once a component passes its individual testing, it is integrated into the larger architecture. After integration, the system is re-tested to ensure functionality and adherence to the workflow.

System-Level Testing

Once all components are integrated, the system undergoes end-to-end testing. This step verifies the fulfilment of functional and nonfunctional requirements and ensures that the platform operates as intended. The following types of testing are employed:

- **Functional Testing:** This testing verifies that all functional requirements outlined in the RASD document are met. By simulating scenarios described in the use cases, testers ensure the system correctly executes tasks and achieves the expected outcomes.
- **Performance Testing:** Performance testing identifies bottlenecks that could impact response times, utilization, and throughput. It also uncovers inefficiencies in algorithms, hardware, or network configurations. By applying expected workloads and comparing performance metrics, optimization opportunities are identified.
- **Usability Testing:** Usability testing involves observing real users as they interact with the platform. This test ensures that the interface is intuitive, accessible, and user-friendly across various scenarios.
- **Load Testing:** Load testing evaluates the system under increasing workloads to identify its upper operational limits. It helps detect potential issues like memory leaks, buffer overflows, or memory mismanagement. By testing the system for prolonged periods under heavy load, testers ensure stability.
- **Stress Testing:** Stress testing examines the system's ability to recover from failures. This includes scenarios of resource reduction or overload. The goal is to ensure resilience and fault recovery under extreme conditions.
- **User Interface Testing:** This ensures the system works easily across different devices, browsers, and platforms.

Integration and Workflow Validation

For each newly developed component:

- A driver will verify that the component works correctly when integrated into the system.
- Subsequent integration tests ensure that existing workflows and module properties remain intact.

When all components are fully integrated:

- Functional testing will verify workflow consistency, alignment with the RASD document, and fulfillment of specified goals and scenarios.
- Load and stress testing will uncover and address memory management issues.

- Performance testing will ensure that the system handles simultaneous users efficiently, keeping response times within acceptable limits.
- User interface testing will validate usability across devices and platforms.

6 | Effort spent

	David Gadiaga	Andrea Pesciotti	Simone Somazzi
Section 1	4	2	2
Section 2	5	17	13
Section 3	11	4	5
Section 4	2	3	5
Section 5	6	3	2

Table 6.1: Hours spent per person