POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering 2
Design Document

Author(s): **Pica Mirko - 10811404**
Pianalto Riccardo - 10835980
Prendin Christian - 10827556

Academic Year: 2024-2025
Version: 1.0
Release date: 07/01/2025

Contents

1 | Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of Students&Company. It is addressed to the developers who have to implement the requirements and could be used as an agreement between the customer and the contractors.
The document is also intended to provide the customer with a clear and unambiguous description of the system's functionalities and constraints, allowing the customer to validate the requirements and to verify if the system meets the expectations.

1.2. Scope

Students&Companies (S&C) is a platform designed to simplify and optimize the matching process between university students looking for internship opportunities and companies offering them. The system analyzes students' profiles, CVs, and indicated preferences, matching them with internship offers posted by companies, which include details on required skills, technologies used, and proposed conditions. Using advanced recommendation algorithms, S&C suggests suitable opportunities to students and notifies companies of candidates who best meet their needs. The platform also supports the entire selection cycle, from application and interview management to feedback collection, ensuring a structured and transparent experience for all users involved, including universities that monitor the progress of internships and address any issues.

1.3. Definition, Acronyms, Abbreviations

Acronyms	Definition
DD	Design Document
RASD	Requirements Analysis & Specification Document
ST	Student
CP	Company
S&C	Students&Company
User	All STs and CPs
API	Application Programming Interface
RX	Requirement X
CMP	Component

Table 1.1: Acronyms used in the document.

1.4. Revision History

Version 1.0 - 07/01/2025

1.5. Reference Documents

- Specification Document Assignment

1.6. Document Structure

The document is structured in seven sections, as described below.

Introduction. In the first section, the chapter elucidates the significance of the Design Document, providing comprehensive definitions and explanations of acronyms and abbreviations. Additionally, it recalled the scope of the Students&Company system.

Architectural Design. The second section shows the main components of the system and their relationships. This section also focuses on design choices and architectural styles, patterns and paradigms.

User Interface Design. The next section, the third, describes the user interface of the system, providing mockups and explanations of the main pages.

Requirements Traceability. The fourth section describes the requirements of the system, showing how they are satisfied by the design choices.

Implementation, Integration and test Plan. This fifth part provides an overview of the implementation of the various components of the system, it also shows how they are integrated and it gives a plan for testing them all.

Effort Spent. In the sixth section are included information about the number of hours each group member has worked for this document.

References. The last section contains the list of the documents used to redact this Design Document.

2 | Architectural Design

2.1. Overview

Here we represent an overview of how the entire S&C architecture is composed of:

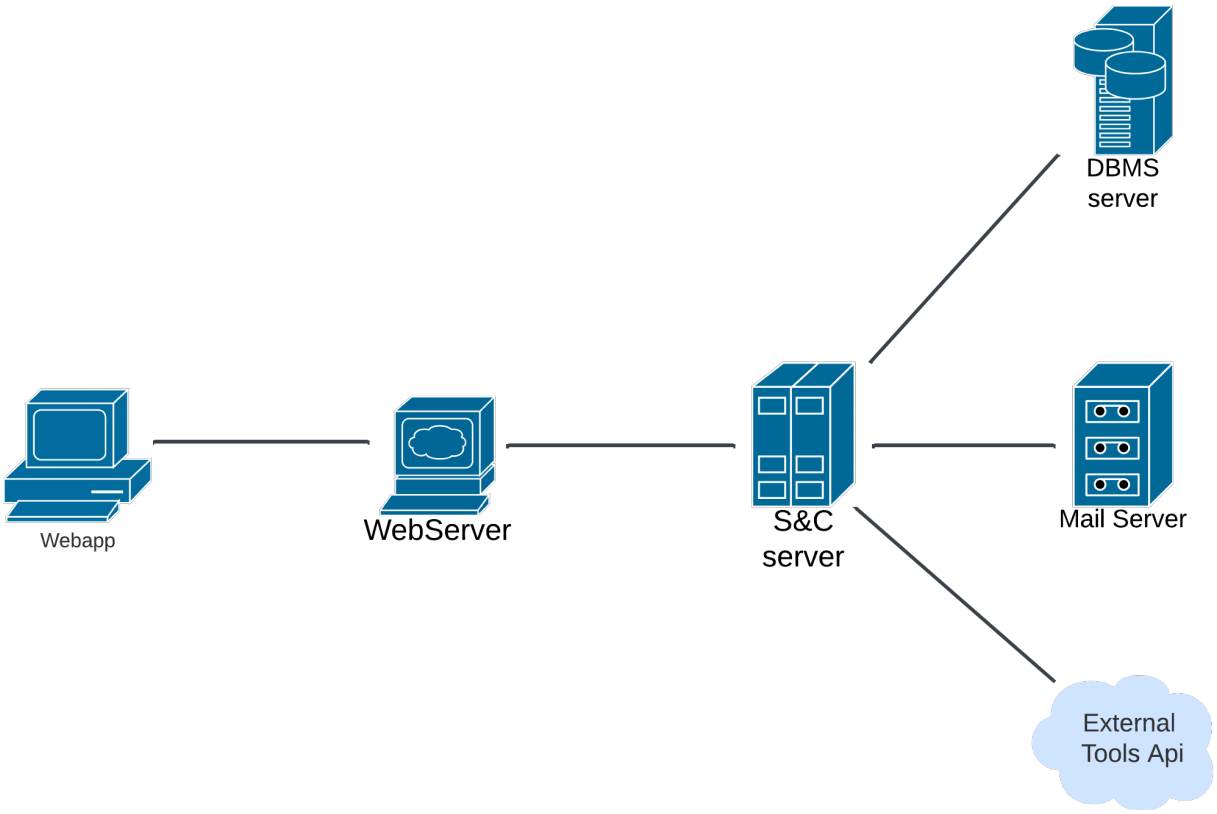


Figure 2.1: S&C Overview.

Client side:

- **WebApp:** Represents the User Interface of the system, providing access through a web application to the offered services such as registration, login, updating profile overview, creating internships, searching and applying for internships, creating tasks for candidates, submitting complaints or providing feedback. It's responsible for all

the User interactions, and it communicates with the main server through the Web Server, using secure protocols like HTTPS.

Server side:

- **Web Server:** handles communication with Users, receiving and processing their inputs. Additionally, it provides load balancing for requests, distributing them among various replicas of the S&C Server. It also manages the User sessions.
- **S&C Server:** the core of the system, contains most of the logic of the software and handles interactions between different components. It also coordinates the communication with the DBMS, triggers the recommendation algorithm through the External Tools API and sends notifications to the users. It serves as the primary server for the entire website and is replicated across multiple machines to handle a high volume of requests.
- **DBMS Server:** stores data related to Users, Internships, Resumes, Recommendations, Complaints and Feedback. It acts as the repository for essential information.
- **Mail Server:** is responsible for sending confirmation email when a new User registers on S&C, enhancing the User registration process.
- **External Tools API:** used to run the recommendation algorithm. It also takes feedback in inputs to reinforce the algorithm over time, ensuring better accuracy in the matching process.

2.2. Component View

2.2.1. High Level Diagram

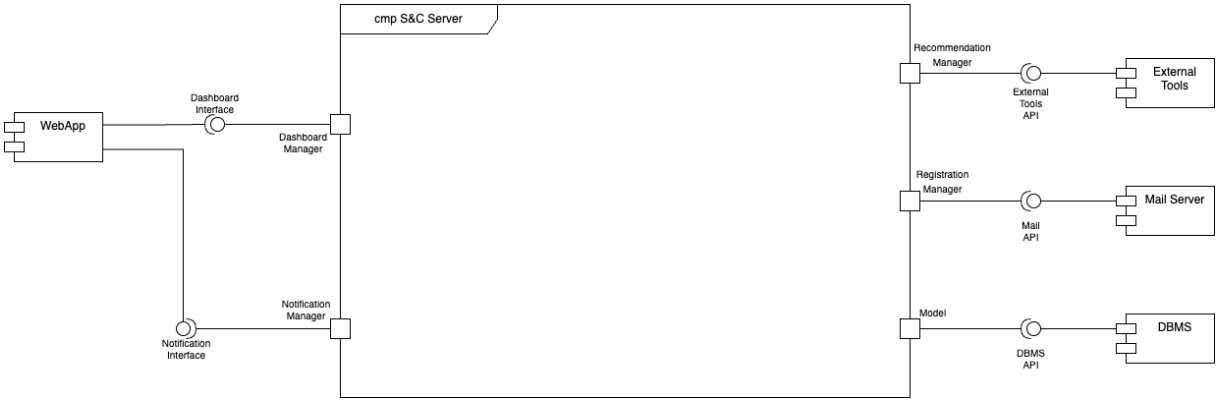


Figure 2.2: High Level Diagram.

In the figure above is the high level component diagram of S&C where it's represented the external components of S&C and how they communicate with the S&C server, in particular:

- **WebApp**: serves as the external access point for Users, allowing communication with the S&C Server through the Dashboard Interface—the sole means for Client-Server interaction from the User side. The S&C Server can relay notifications, such as Student uploading overviews or Internship creation, to Users through the Notification Interface.
- **DBMS:** is the storage repository for all User data, Internships, Feedback and Complaints. It communicates with the S&C Server via the DBMS API, which is connected to the Model component.
- **Mail Server:** responsible for sending registration confirmation emails, the Mail Server communicates with the S&C Server using the Mail API interface. This interface is linked to the Registration Manager component, which oversees the User registration process..
- **External Tools:** external application used for running and reinforcing a sophisticated recommendation algorithm. It communicates with the S&C Server through the External Tools API, connecting to the Recommendation Manager component. The Recommendation Manager handles the notification process for the matching phase of the system.

2.2.2. Low Level Diagram

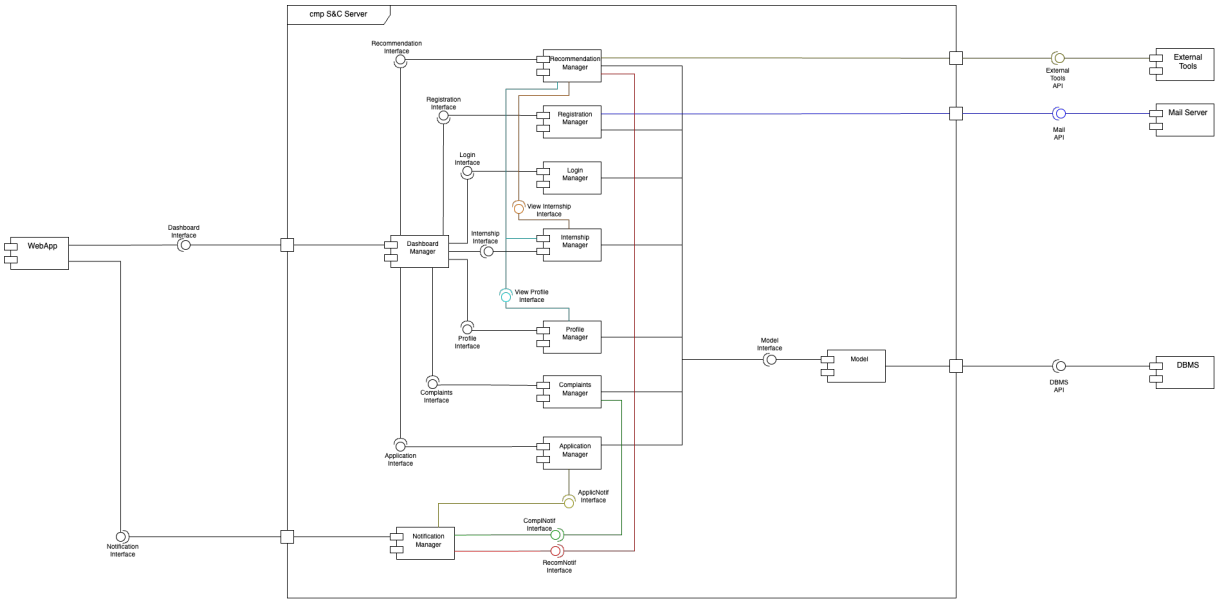


Figure 2.3: Low Level Diagram.

The figure above represents the complete architecture of S&C website with each components inside the S&C Server:

- **Dashboard Manager:** This fundamental component orchestrates all the communication between User and the S&C website. Users communicate with the system through the Dashboard Interface, so the Dashboard Manager address all the requests to the right component, depending on the User's needs.
- **Model:** This component represents the dataset of the server, and it act as a mask to the database server, so every component have to interface with it to access data from DBMS, which will access it directly through the DBMS API.
- **Recommendation Manager:** The component that handles everything about the recommendation phase of the system. It periodically triggers a new computation for finding new matches between students and internships, and it does that by using the External Tools API, which communicates directly with the system in which the computation resides. It also communicate with the model for retrieving all the information necessary to the user to visualize the correct recommendation list, each time there is a new request incoming from the Dashboard Manager. This component additionally communicates with the Profile Manager through the View Profile Interface, every time that a company, from it's recommendation list wants

to check the profile overview of a Student.

- **Registration Manager:** This components handles the registration of new Users on the system. The User when trying to create a new account interacts with the Dashboard Manager, which contacts the Registration Manager through the Registration interface. Then the Registration Manager handle the request and manages to contact the Mail Server through the Mail API, to send a confirmation email to the new User. Once this is done it uses the Model through the Model Interface for saving user's information on the DBMS.
- **Login Manager:** This component handle the Login process for registered Users. When a User attempts to Login, the Dashboard Manager address the request to this component, which contacts the Model through the Model Interface to retrieve data from the DBMS and so authenticate the User.
- **Profile Manager:** Component that allows Students to create their profile overview. When a Student starts the process to complete it's profile, the Dashboard Manager forwards the requests to the Profile manager using the Profile Interface. The Profile Manager communicates with the Model component through the Model Interface to store new information in the DBMS. It also manage to retrieve Students information when a company wants to view the overview of a Student within the recommendation list.
- **Internship Manager:** This component allow the Companies to create, modify and visualize internships. When a company wants to publish a new internship, interacts with the dashboard manager, which forwards the requests to the Internship Manager. This component manage to save new published internship, modify the status or retrieve information for internship search from the Model component, interacting with it via the Model Interface.
- **Application Manager:** This is the main component of the selection phase of an internship. When a company wants to start preparing tasks or submit them to candidates, interacts with the Dashboard Manager, who forwards the requests to the Application Manager, who manage to interact with the Model through the Model Interface for saving information. It can also notify the candidates when the tasks are ready to be carried out, or results are ready, contacting the Notification Manager module through the ApplicNotif Interface.
- **Complaints Manager:** This component handles the complaints related to an internship. When a User wants to file a complaint about an internship, interact

with the Dashboard Manager, who forwards the requests to the Complaint Manager through the Complaint Interface, which needs to Notify the University of the Student related complaint, contacting the Notification Manager module through the InterviewNotif Interface. It can also store the complaints on the model for the purpose of reinforcing the algorithm.

- **Notification Manager:** Component that handles each notification that has to be sent to the Users, in particular when a new match has been found from the recommendation algorithm, it sends a notification both to the Student and the Company related with that match; when the interview phase is completed, it sends the notification to all the candidates informing them wether they've been selected or not; when a new complaint is made from a company or a student, it notify the university that has to handle it. All the communication from other components to this are made through the InterviewNotif Interface, ComplNotif Interface or the Recommendation Interface, and it communicates with the web app through the Notification Interface..

2.2.3. Recommendation Manager

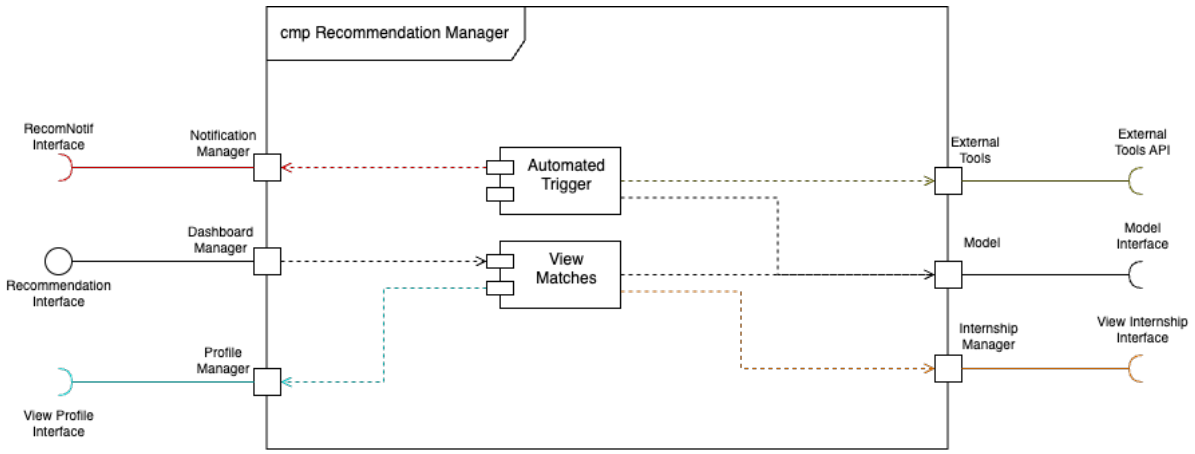


Figure 2.4: Recommendation Manager.

The Recommendation Manager is composed by two other sub-components:

- **Automated Trigger component:** This component is essential for ensuring the continuous and autonomous operation of the recommendation system. Its primary function is to periodically initiate the recommendation algorithm, which scans the database for new or updated profiles and internship postings. By doing so, the component guarantees that students and companies receive the most relevant and recent

matches based on evolving criteria such as updated CVs, new internship opportunities, and feedback collected from previous experiences. This component interfaces directly with the External Tools API, responsible for executing the complex matching logic that considers various parameters, including skills, interests, and project domains.

- **View Matches component:** The View Matches component retrieves the latest matches identified by the Automated Trigger and displays them through the user's dashboard. For students, a list of internship opportunities that align with their skills and interests. For companies, potential candidates whose profiles closely match the requirements of their internship postings. Users can also filter and sort recommendations. Additionally, the View Matches component provides direct links to student profiles or internship details.

2.2.4. Internship Manager

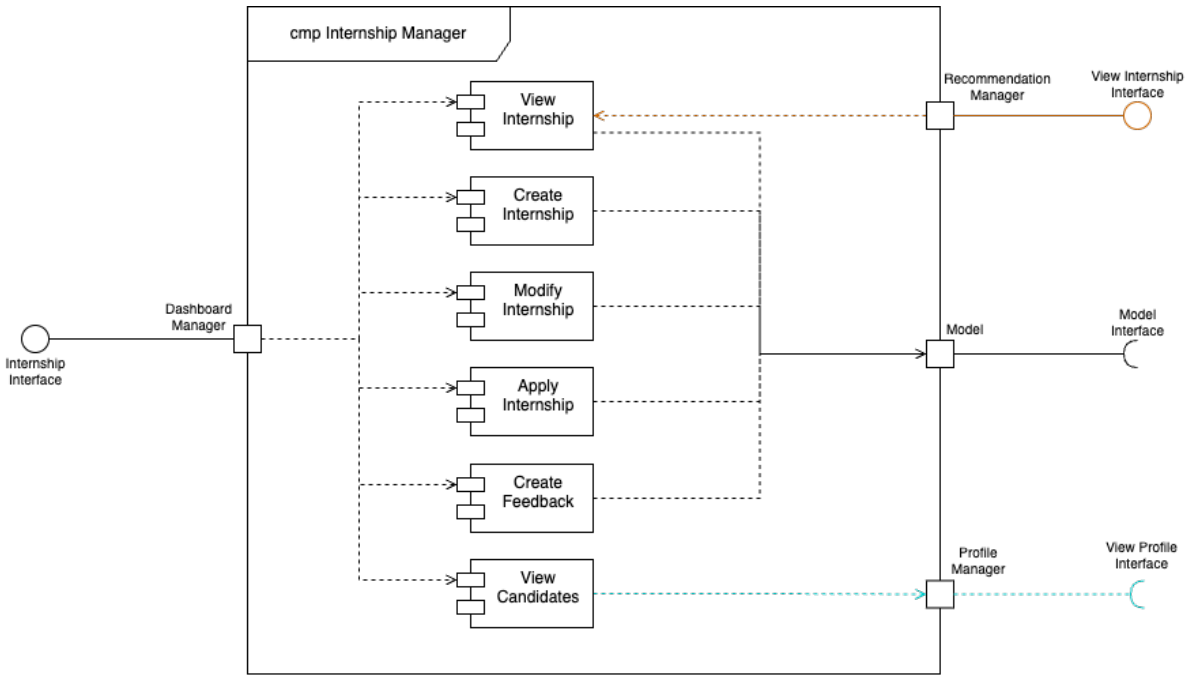


Figure 2.5: Internship Manager.

The Internship Manager is made by the following components:

- **View internship component:** The View Internship component provides a comprehensive interface for accessing and managing internship postings within the system. For students, the component acts as a searchable catalogue of available intern-

ships, with filtering and sorting options. This component pulls data directly from the DBMS through the Model Interface, ensuring real-time synchronization with the latest postings. It allows companies to view the number of applicants, monitor engagement, and update internship details if necessary. For students, each internship listing includes detailed descriptions, project requirements, and any associated benefits. Additionally, the View Internship component integrates with the Apply Internship subcomponent.

- **Create internship component:** This component facilitates the creation and publication of new internship offers by companies. It provides a structured form that guides users through defining essential parameters, such as internship title, required skills, project description, duration, and compensation details. Once the form is completed, the Create Internship component validates the input to ensure completeness and consistency. It then interfaces with the DBMS through the Model component to store the new internship in the database. If errors are detected during the validation process, the component provides immediate feedback, allowing the user to correct any issues before submission.
- **Modify internship component:** The Modify Internship component enables companies to update or delete existing internship postings. Users can access this component through the View Internship interface, selecting the internship they wish to edit. Changes made through the Modify Internship component are instantly propagated to the DBMS, ensuring all users view the most up-to-date information.
- **Apply internship component:** The Apply Internship component is central to the student experience on the platform. It allows students to submit applications for internships directly through the system. Upon clicking the "Apply" button, the component gathers the student's profile data, CV, and any additional required documents before forwarding the application to the target company. Companies are notified of new applications via the Notification Manager, ensuring timely review and response. The Apply Internship component also includes mechanisms for students to withdraw applications if necessary, with relevant notifications sent to the company and the system updated accordingly.
- **Create feedback component:** After an internship is completed, the Create Feedback component enables students and companies to provide feedbacks about their experience. Companies evaluate student performance, highlighting strengths and areas for improvement. Similarly, students can rate their experience with the company, providing insights into the quality of mentorship, project alignment, and over-

all satisfaction. This component compiles feedback into structured reports, which are stored in the DBMS. The feedback serves as valuable input for refining the recommendation algorithm, improving the accuracy of future matches by incorporating performance data and user satisfaction.

- **View Candidates component:** This component allows companies to access and manage the list of students who have applied for their internships. Through an interactive dashboard, companies can view applicant profiles, download CVs, and review additional submitted documents. The component includes sorting and filtering tools. Additionally, it provides features for marking applicants as shortlisted, rejected, or accepted. Once candidates are selected, the component forwards their details to the Interview Manager, initiating the next phase of the recruitment process. Notifications are sent to the selected students, informing them of their application status.

2.2.5. Profile Manager

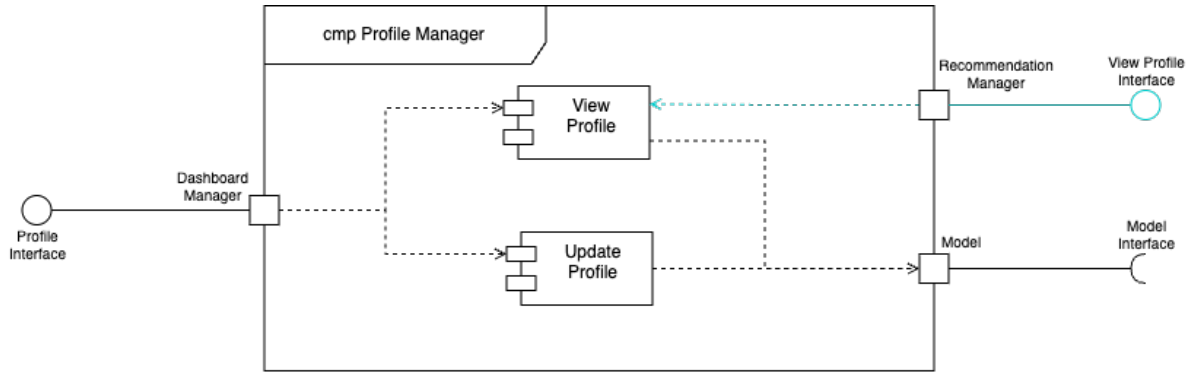


Figure 2.6: Profile Manager.

The Profile Manager is composed by the following components:

- **View Profile component:** The View Profile component allows users to access and review profiles stored within the system. For students, this component displays their profile overview, including personal information, educational background, skills, and uploaded CVs. For companies, it provides access to student profiles that match internship postings or appear on recommendation lists. The component supports viewing profiles in detail and exporting them for offline review. The View Profile component integrates directly with the DBMS to ensure that profile data is current and reflective of the latest updates made by students.

- **Update Profile component:** This component is designed to enable students to manage and update their profiles. Students can edit sections related to personal details, work experience, skills, and education. The Update Profile component performs real-time validation to ensure that all data entered adheres to format and completeness standards. Once submitted, the updated data is written to the DBMS via the Model component. The system also supports partial updates, allowing students to save progress and complete their profiles incrementally.

2.2.6. Complaints Manager

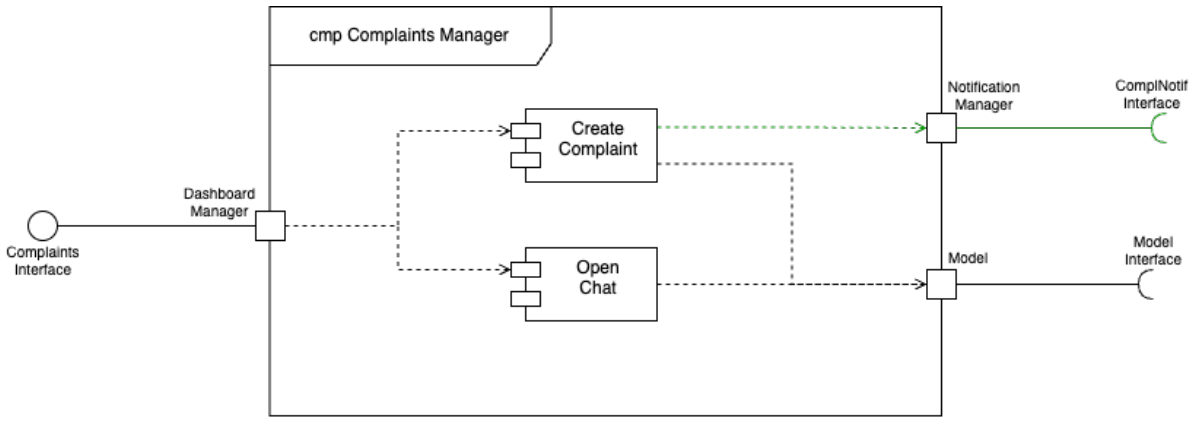


Figure 2.7: Complaints Manager.

The Complaints Manager is made by the following components:

- **Create Complaint component:** The Create Complaint component allows users to file formal complaints about internships. This feature is available to both students and companies, providing a structured form to capture detailed descriptions of issues encountered during the internship process. The component sends complaints to the relevant university, which is notified through the Notification Manager. Complaints are logged in the DBMS, ensuring traceability and enabling further investigation if needed.
- **Manage Complaints component:** This component facilitates direct communication between universities and users regarding complaints. It can manage a secure chat channel, allowing real-time interaction to resolve issues efficiently. And gives universities opportunity to resolve complaints.

2.2.7. Application Manager

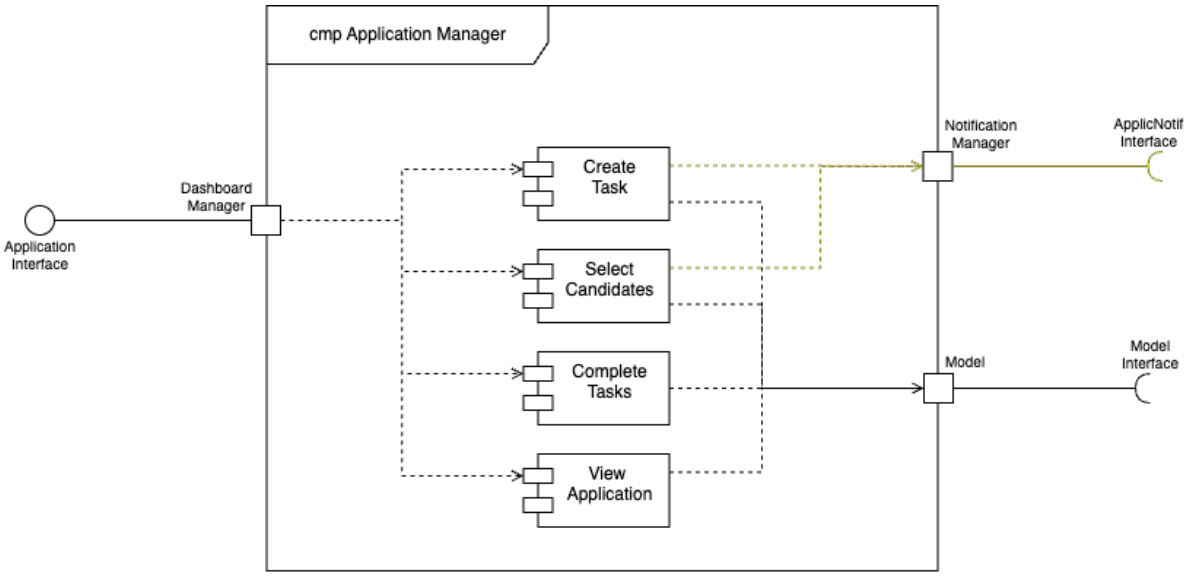


Figure 2.8: Application Manager.

The Application Manager is composed by the components:

- **View Application:** The View Application component allows companies and students to access and monitor the status of ongoing applications. Companies can track candidates' progress through various selection stages, while students can check the status of their applications and receive updates on any required actions. This component ensures transparency and easy access to information regarding the selection processes.
- **Create Task component:** The Create Task component is a key element of the selection process, enabling companies to design and assign practical exercises or technical assessments to candidates during the evaluation phase. This component assess students' technical skills and problem-solving abilities by simulating real-world scenarios or creating knowledge-based tests.
- **Select Candidates component:** The Select Candidates component is crucial for the final phase of the selection process. This module allows companies to review task results, rank candidates based on performance, and finalize selections for internship offers or further interviews. It ensures that companies efficiently manage and evaluate large pools of applicants, streamlining decision-making and communication.

- **Complete Tasks:** The Complete Tasks component gives candidates the ability to complete assigned tasks during the selection process. These tasks may include technical tests, projects, or practical exercises that must be finalized within a specified deadline. Companies can track task completion and evaluate the results, using them as criteria to select the most suitable candidates.

2.3. Deployment View

In this section, the Deployment diagram of the S&C system will be shown, followed by a description of the components and their interactions:

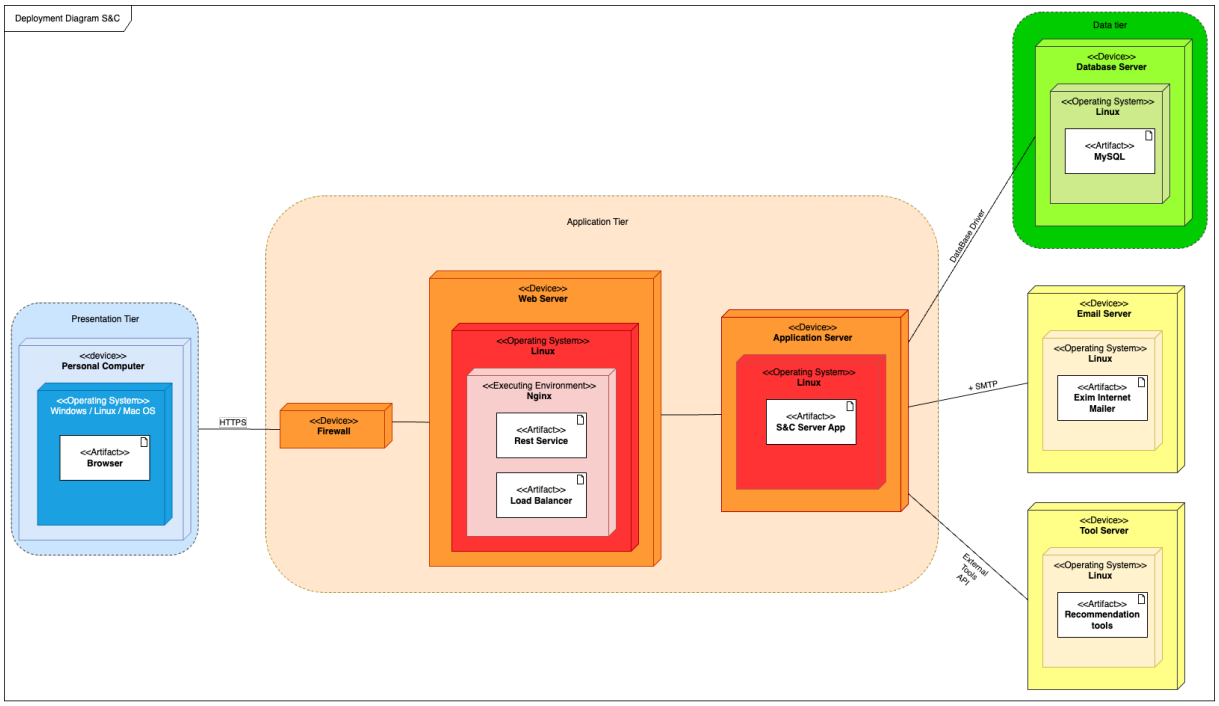


Figure 2.9: Deployment Diagram.

Personal Computer: STs and CPs can access to the S&C system through their preferred browser. The browser will communicate with the Web Server. Users can also use every device that allows them to search on a web browser, such as mobile phone, tablet, ecc.

Web Server: The Web Server provides access to the Application Server to everyone that access to the S&C system through a web browser. It doesn't handle any logic for the system, but it has the very important task to balance the load request incoming from the users. It also provide all the necessary file to have a right render of the page, such as

HTML, CSS, JSON and JavaScript.

Firewall: It provides a way to limit the attack surface of any potential intruder by providing strict access rules.

Application Server: The application server contains all the logic at the base of the S&C system. It communicates with the clients through HTTPS protocol managed by the Web Server. It also communicates with the Database Server through the DataBase Driver, with the Mail Server via SMTP and with the Tool Server by using a dedicated API

Database Server: All the data about Users, internships and matching are stored in the Database Server and managed by MySQL. The Application Servers can retrieve information by contacting the model module, which contacts the physical Database via the Database driver.

Email Server: Once completed the registration process, Users have to confirm their profile by clicking on the link sent via Email by the Email Server. The application Server, after the registration process, contacts the Email Server through SMTP protocol to send the confirmation Email to the Users.

Tool Server: With the External Tool APIs, the Application Server can send useful information to the Tool Server, enhancing the precision of the algorithm. Furthermore the Application Server can decide to run the algorithm to retrieve other matches and send the notification to the related Users.

2.4. Runtime View

2.4.1. Log in to the system

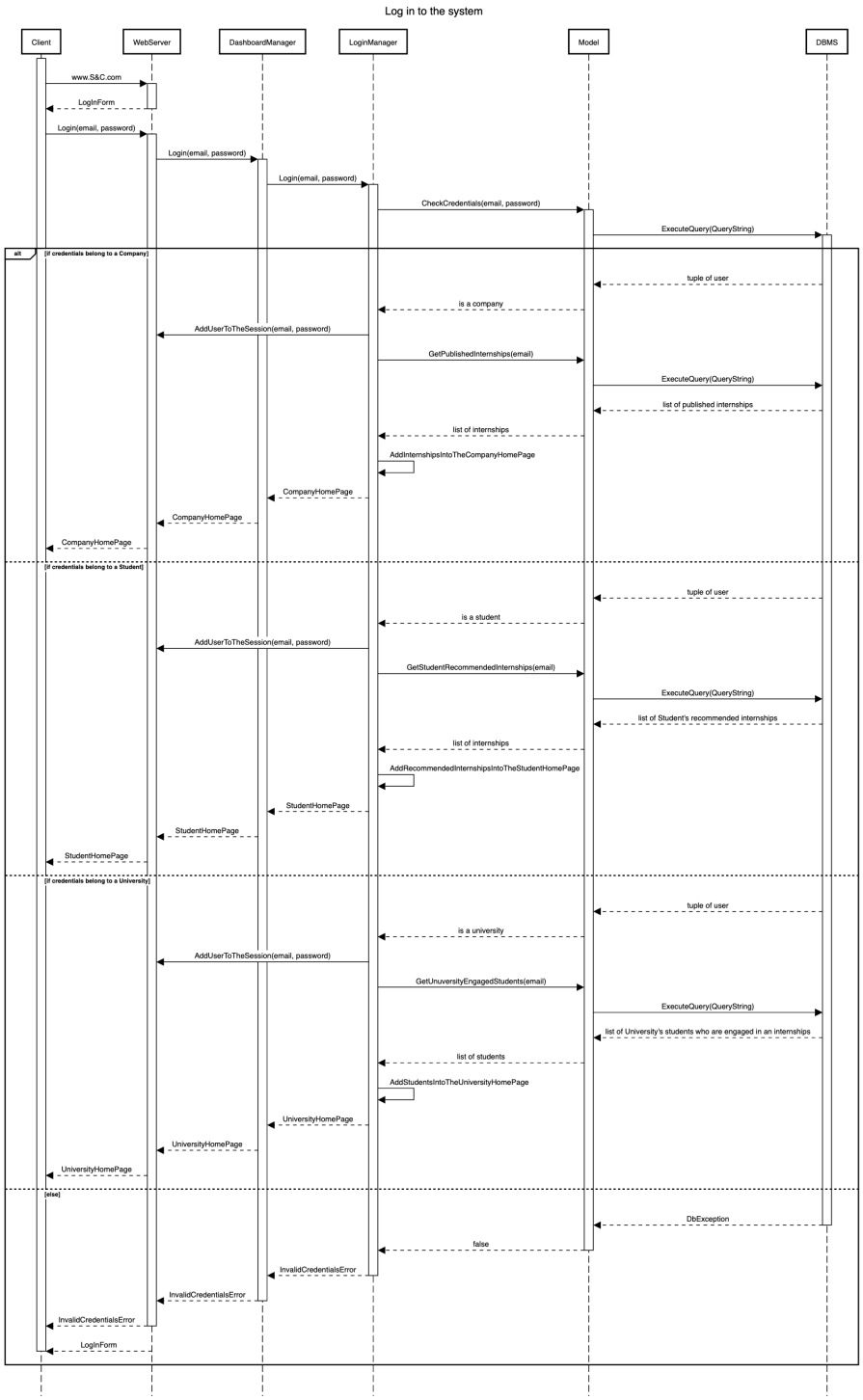


Figure 2.10: Runtime view for 'Log in to the system'.

This sequence diagram illustrates the login process for different types of users. The Client accesses the S&C website, initiating the login by submitting credentials (email and password). The WebServer forwards the request to the DashboardManager, which interacts with the LoginManager to validate the provided credentials. The LoginManager queries the Model to verify the user's information on the database. Depending on the user's type (Company, Student, or University), the system retrieves relevant data, such as published internships or recommended offers, and displays the appropriate homepage. In case of invalid credentials, an error message is shown, and the login form is presented again to the Client.

2.4.2. Log out from the system

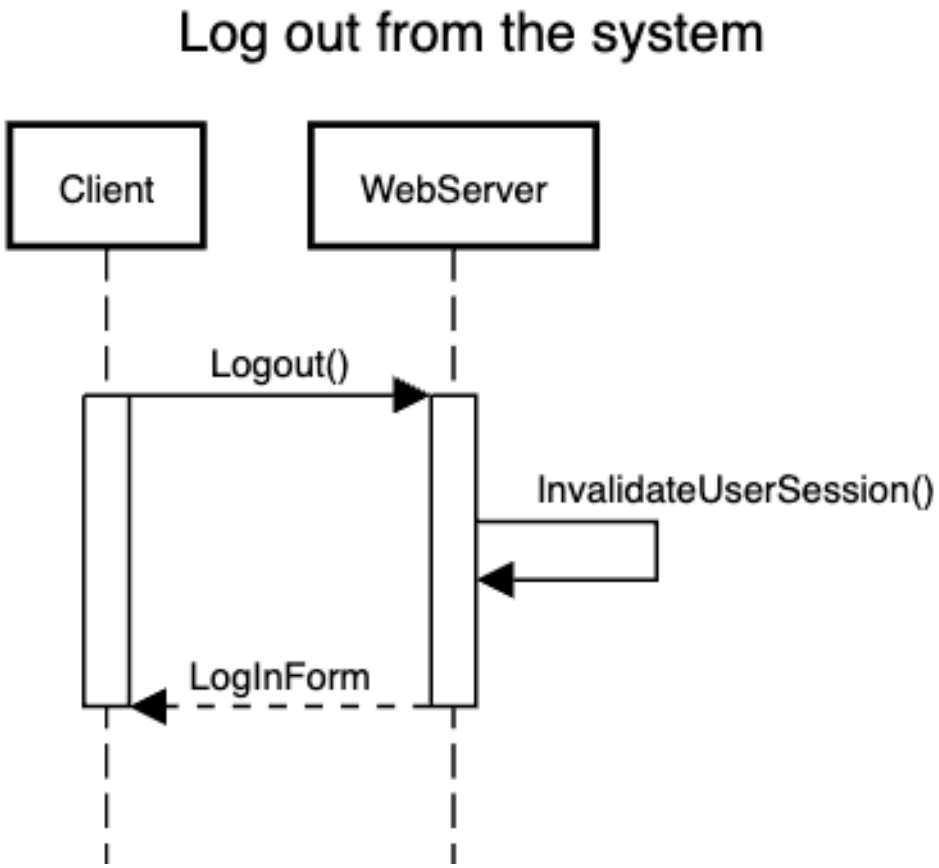


Figure 2.11: Runtime view for 'Log out from the system'.

This sequence diagram represents the logout process. The Client initiates the logout by sending a request to the WebServer. Upon receiving the request, the WebServer invalidates the user session, ensuring the client is logged out. The WebServer then redirects the Client to the login form, completing the process.

2.4.3. Register an account to S&C

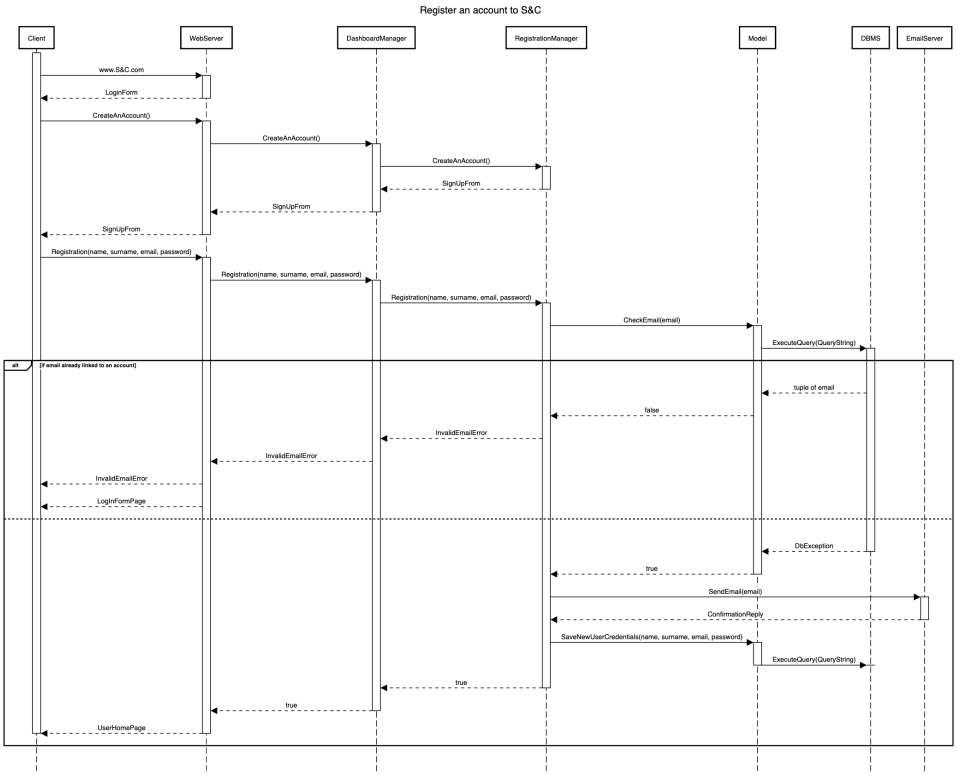


Figure 2.12: Runtime view for 'Register an account to S&C'.

This sequence diagram represents the process of user registration on the platform. The Client accesses the S&C website and is presented with the login form. Upon selecting 'Create an Account,' the request is processed through the WebServer, DashboardManager, and RegistrationManager, which returns a signup form to the Client. The Client submits personal information, including name, surname, email, and password. The WebServer forwards the data through the DashboardManager to the RegistrationManager, which verifies the email's uniqueness by querying the database. If the email is already linked to an account, an error message is sent back to the Client, and the login form is shown again. Otherwise, the system proceeds to send a confirmation email. Upon receiving a reply from the EmailServer, the new user credentials are saved in the database, and the Client is redirected to their homepage, completing the registration process.

2.4.4. View and update their profile

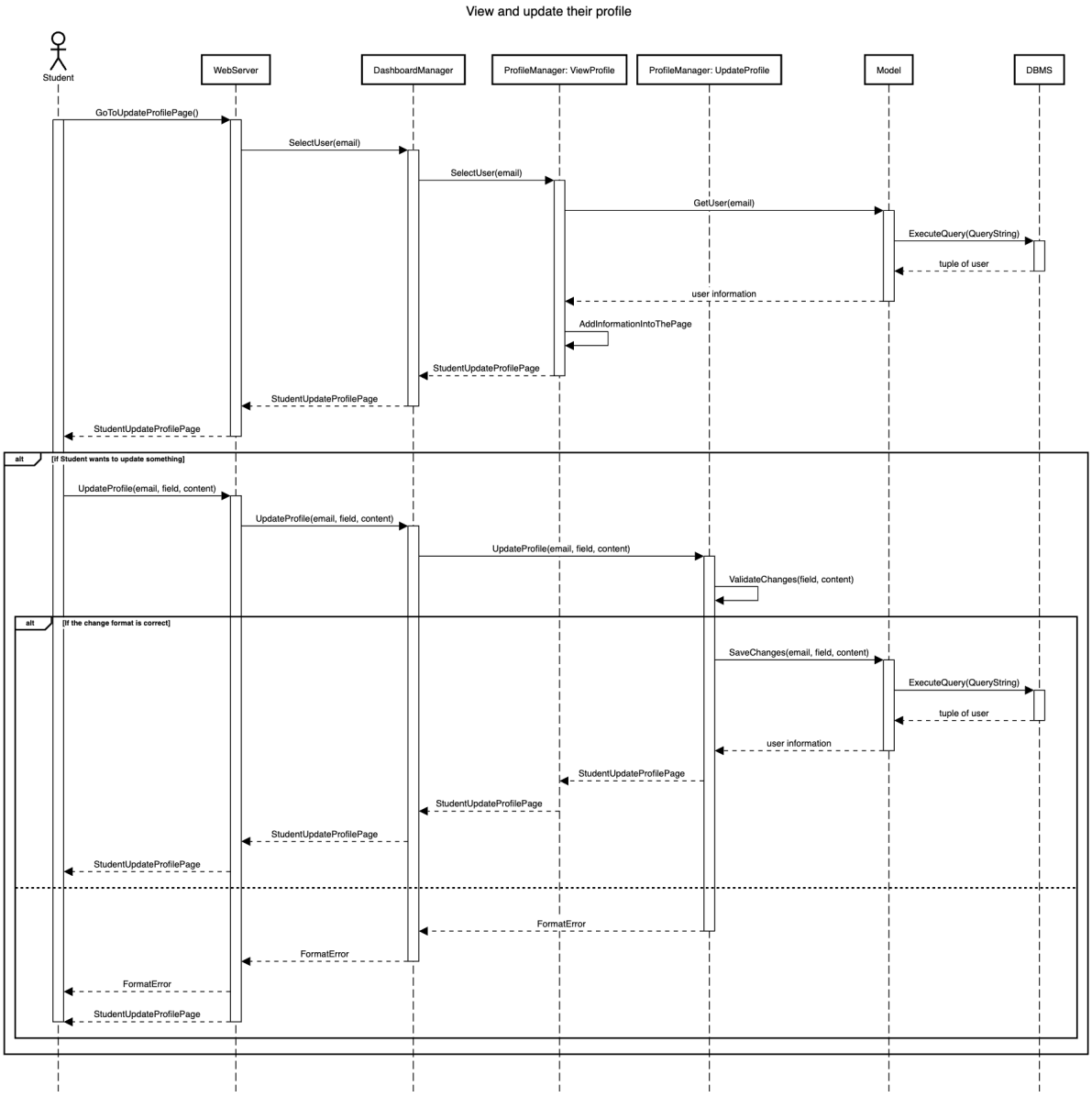


Figure 2.13: Runtime view for 'View and update their profile'.

This sequence diagram illustrates the process through which a student can view and update their profile on the S&C platform. Initially, the student accesses the update profile page via the WebServer. The WebServer forwards the request to the DashboardManager, which interacts with the ProfileManager (ViewProfile) component to retrieve user data from the Model by querying the database. The retrieved information is then displayed on the profile update page. If the student decides to update their profile, the changes are submitted to the WebServer, which processes the request through the DashboardManager.

The update is handled by the ProfileManager (UpdateProfile) component, which validates the changes. If the format is correct, the new information is saved to the database, and the updated profile page is displayed. In case of an error in the format, the student receives an error message while remaining on the update profile page.

2.4.5. Search for internships

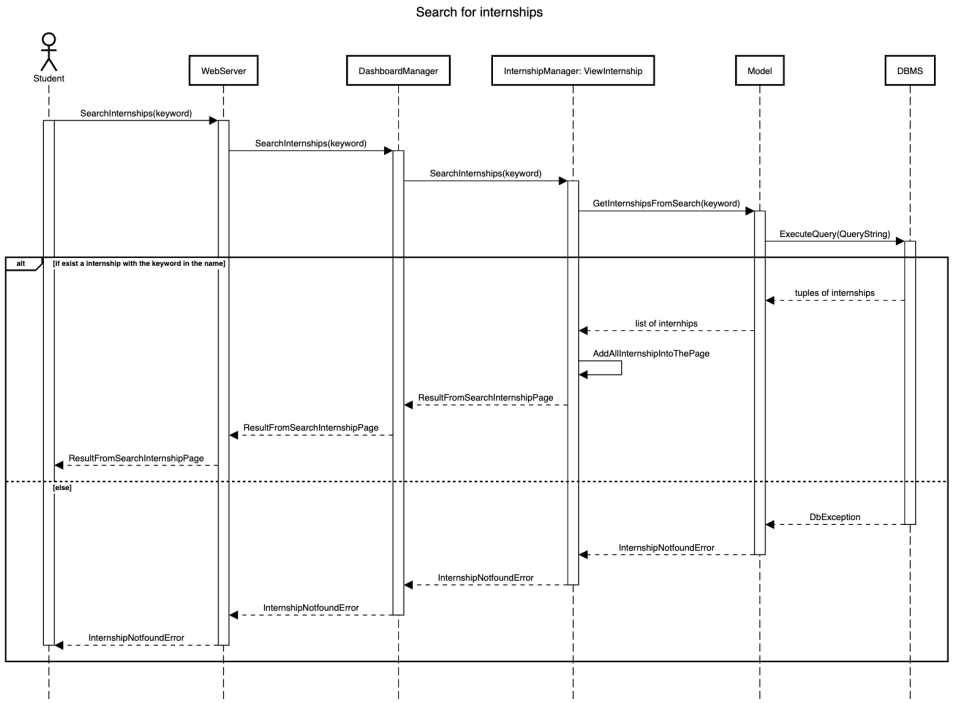


Figure 2.14: Runtime view for 'Search for internships'.

This sequence diagram represents the process through which a student searches for internships on the S&C platform. The student initiates the search by entering a keyword, which is sent to the WebServer. The WebServer forwards the request to the DashboardManager, which interacts with the InternshipManager (ViewInternship) to retrieve relevant internships from the database via the Model by querying the database. If internships matching the keyword are found, they are displayed to the student in the search results. If no matching internships exist, an error message is returned indicating that no results were found.

2.4.6. Apply for an internship

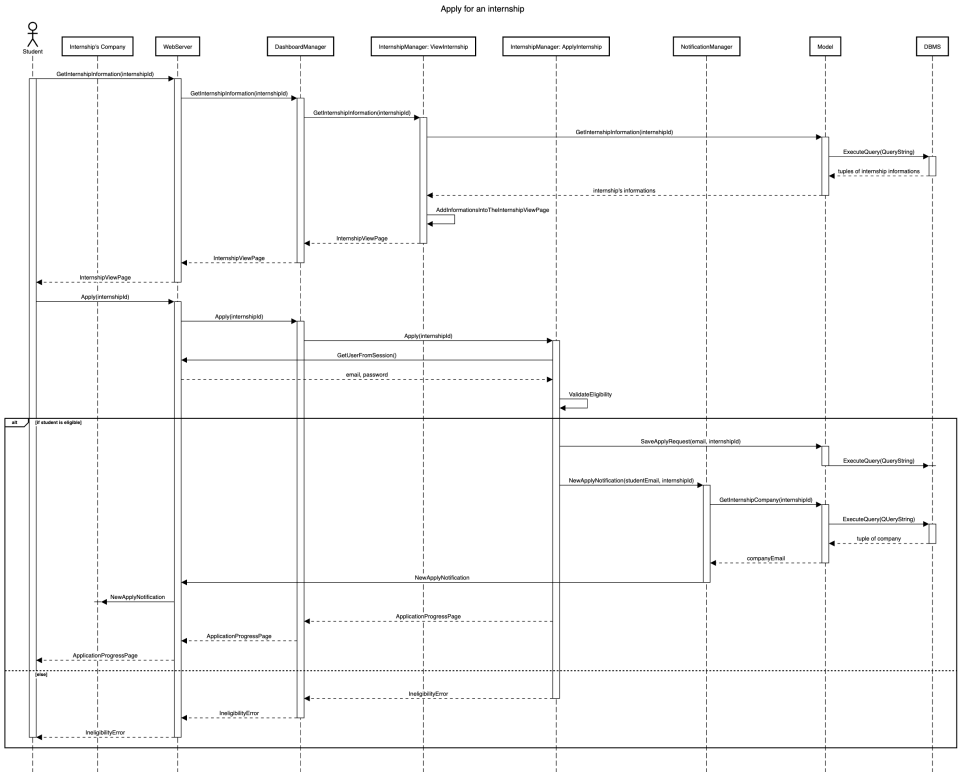


Figure 2.15: Runtime view for 'Apply for an internship'.

This sequence diagram illustrates the process of applying for an internship on the S&C platform. The student begins by requesting detailed information about a specific internship. This request is handled by the WebServer, which retrieves the relevant data from the DashboardManager and InternshipManager by querying the database through the Model. Once the internship details are displayed, the student proceeds to apply. The WebServer forwards the application request to the DashboardManager, which delegates it to the ApplyInternship component. The system validates the student's eligibility and, if successful, saves the application in the database. A notification is then sent to the company offering the internship via the NotificationManager. If the student is not eligible to apply, an error message is returned, informing them of the issue.

2.4.7. Monitor the status of their applications

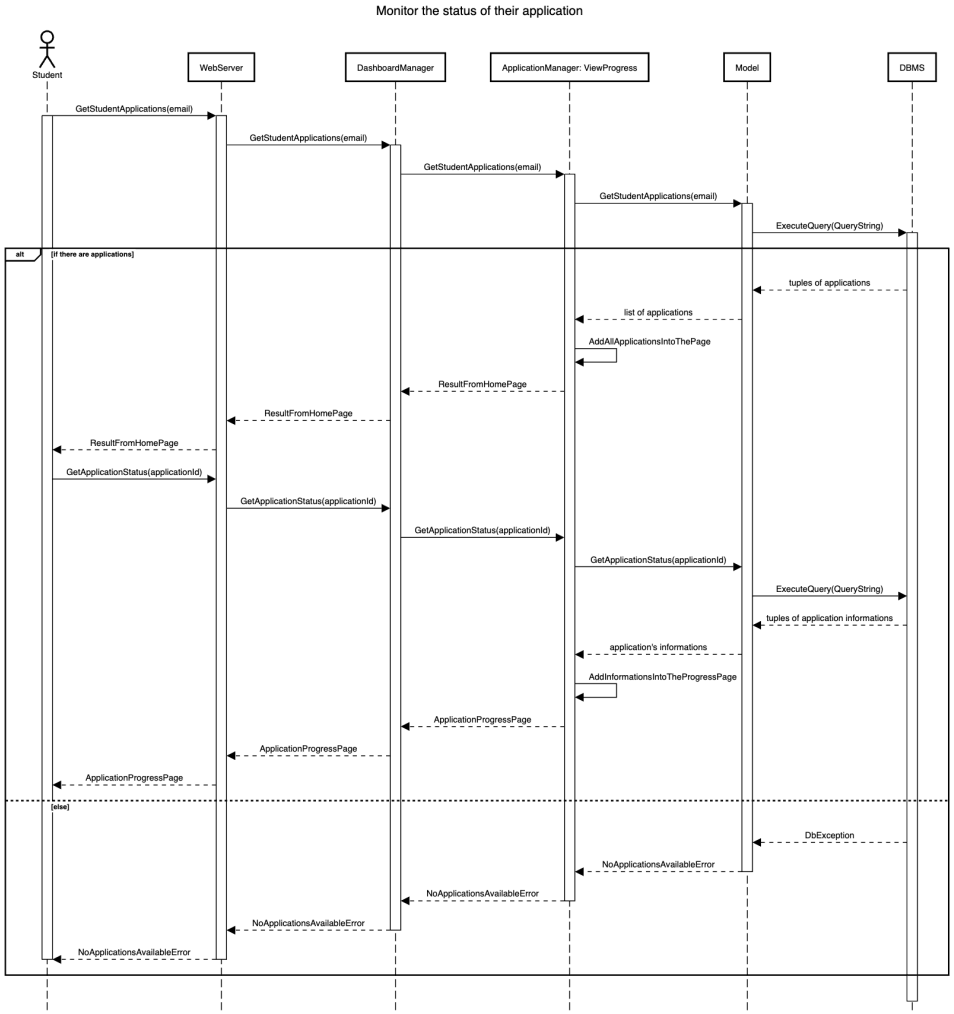


Figure 2.16: Runtime view for 'Monitor the status of their applications'.

This sequence diagram represents the process through which a student monitors the status of their internship applications. The student begins by requesting a list of their applications from the WebServer. The request is passed through the DashboardManager to the ApplicationManager (ViewProgress), which retrieves the relevant data by querying the database via the Model. If applications are found, they are displayed to the student on the homepage. The student can then select a specific application to view its current status. This triggers another query to fetch detailed information about that application, which is than presented on the application progress page. If no applications exist, an error message is returned, informing the student that no records are available.

2.4.8. Accept or reject internship offers

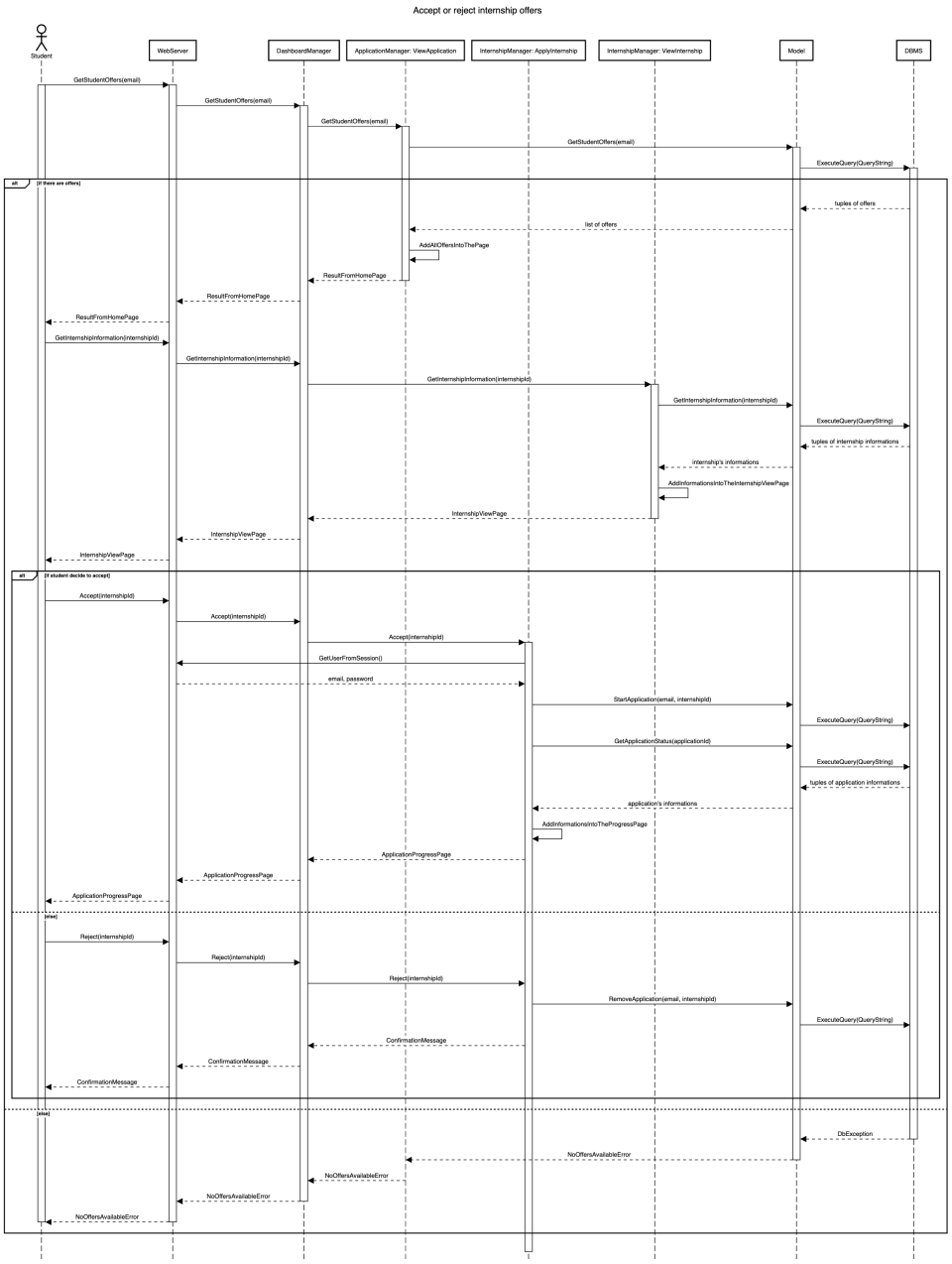


Figure 2.17: Runtime view for 'Accept or reject internship offers'.

This sequence diagram illustrates the process by which a student accepts, or rejects internship offers on the S&C platform. The student initiates the process by requesting a list of available offers. This request is forwarded by the WebServer through the DashboardManager to the ApplicationManager, which queries the database for matching offers via the model. If offers are available, they are displayed to the student. The student can then select a specific internship to view more details. This triggers a request to the Intern-

shipManager to retrieve detailed information about the selected internship. If the student decides to accept the offer, the WebServer forwards the acceptance to the ApplyInternship component, which initiates the application process and updates the status in the database. Conversely, if the student rejects the offer, the system removes the application entry from the database, and a confirmation message is shown to the student. If no offers are found, the system returns an error message indicating the absence of available offers.

2.4.9. Complete questionnaires from companies

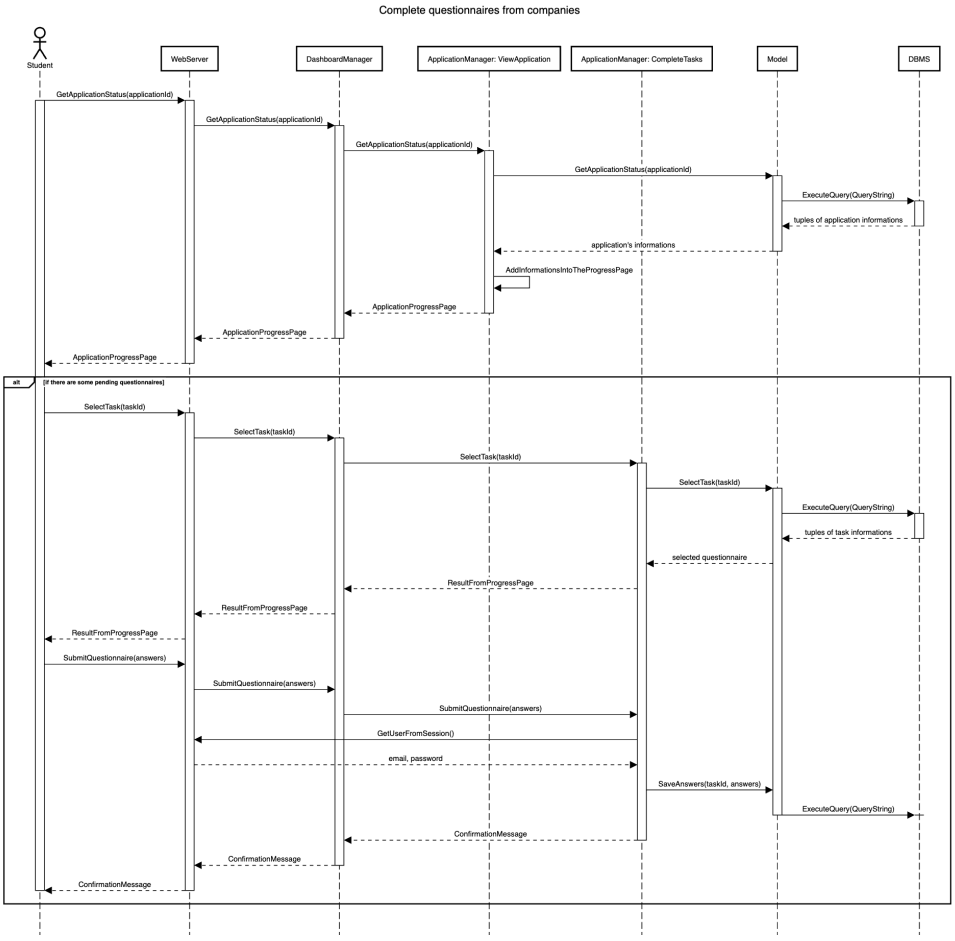


Figure 2.18: Runtime view for 'Complete questionnaires from companies'.

This sequence diagram illustrates the process by which a student completes and submits questionnaires assigned by companies as part of an internship application. The student begins by checking the status of their applications through the WebServer. This request is forwarded through the DashboardManager to the ApplicationManager, which queries the database for relevant application information. If pending questionnaires are available, the student selects a specific task to complete. The WebServer processes this request by

retrieving the task details from the database via the CompleteTasks component of the ApplicationManager. After filling out the questionnaire, the student submits the answers through the WebServer, triggering a process to save the responses in the database. A confirmation message is returned to the student, indicating successful submission. If no questionnaires are available, the student simply views the application progress page without further actions.

2.4.10. Provide feedback on completed internships

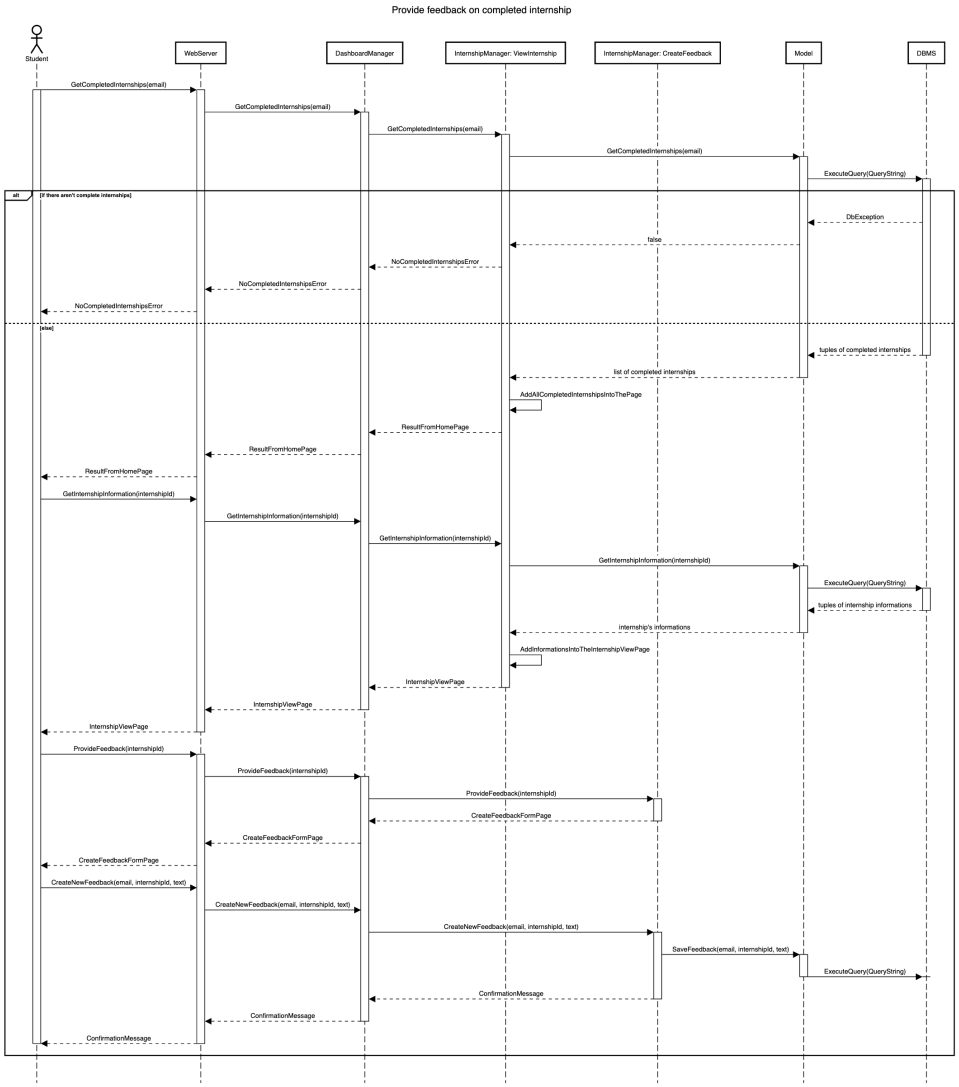


Figure 2.19: Runtime view for 'Provide feedback on completed internships'.

The sequence diagram shows how a student provides feedback on a completed internship. First, the student requests a list of completed internships from the WebServer that

contacts the DashboardManager and the InternshipManager, which queries the database via the model. If no completed internships are found, an error message is shown to the student. If internships exist, they are displayed on the student's homepage. The student selects an internship and views its details, which are retrieved from the Model by querying the database. The student then chooses to provide feedback and a form to submit a feedback is prompted to him. After the student submits the feedback, the WebServer sends the data to the DashboardManager, which saves it in the database. Finally, a confirmation message is sent back to the student.

2.4.11. File complaints about internships

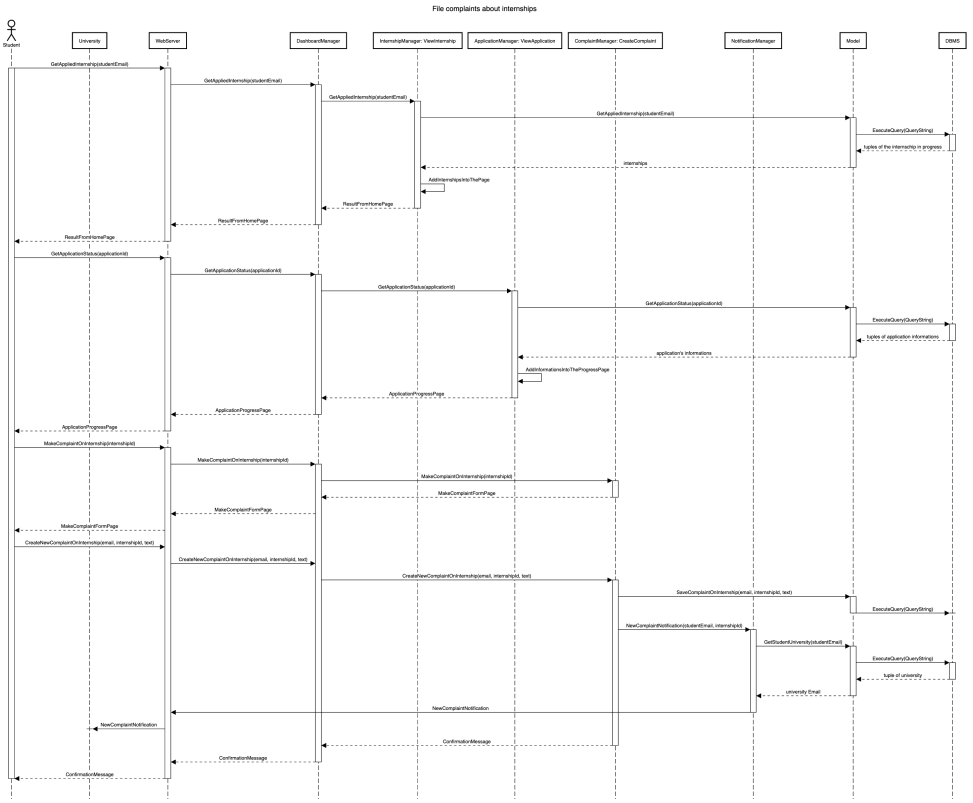


Figure 2.20: Runtime view for 'File complaints about internships'.

The sequence diagram illustrates the process for a student to file complaints about internships. Initially, the student requests a list of applied internships by sending a request to the WebServer, which queries the DashboardManager and the InternshipManager. The internships in progress are retrieved from the Model and database and displayed on the student's homepage. Than the student clicks on a specific application and details are retrieved and displayed on the student's progress page. The student then chooses to make a

complaint about the internship. The WebServer forwards the request to the DashboardManager, which invokes the ComplaintManager to create a complaint form. The form is returned to the student via the WebServer and DashboardManager. The student submits the complaint with relevant details, which are stored in the database. After saving the complaint, the NotificationManager is triggered to send a notification to the student's university. The NotificationManager retrieves the university's email from the Model and sends the notification. Finally, the DashboardManager sends a confirmation message to the student, signaling the completion of the complaint process.

2.4.12. Create and publish internships

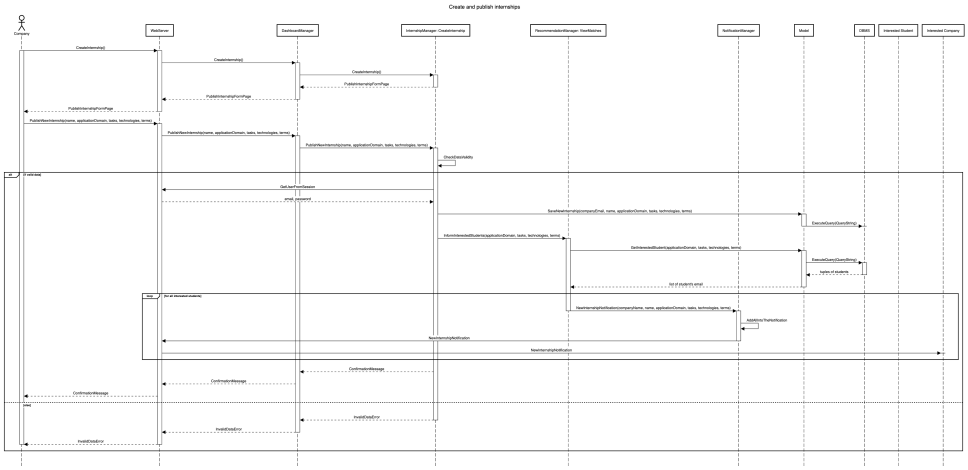


Figure 2.21: Runtime view for 'Create and publish internships'.

The sequence diagram shows how a company creates and publishes an internship. First, the company requests to create an internship, and the WebServer forwards the request to the DashboardManager, which triggers the InternshipManager to present a form for the internship details. The company submits the internship information, and the system checks if the data is valid. If the data is valid, the internship is saved in the Model and database. The InternshipManager then informs interested students by notifying the RecommendationManager, which retrieves the list of interested students from the Model and sends notifications through the NotificationManager. Each student receives a notification about the new internship. Finally, a confirmation message is sent to the company, signaling the successful creation and publication of the internship. If the data is invalid, an error message is sent back to the company.

2.4.13. View and manage published internships

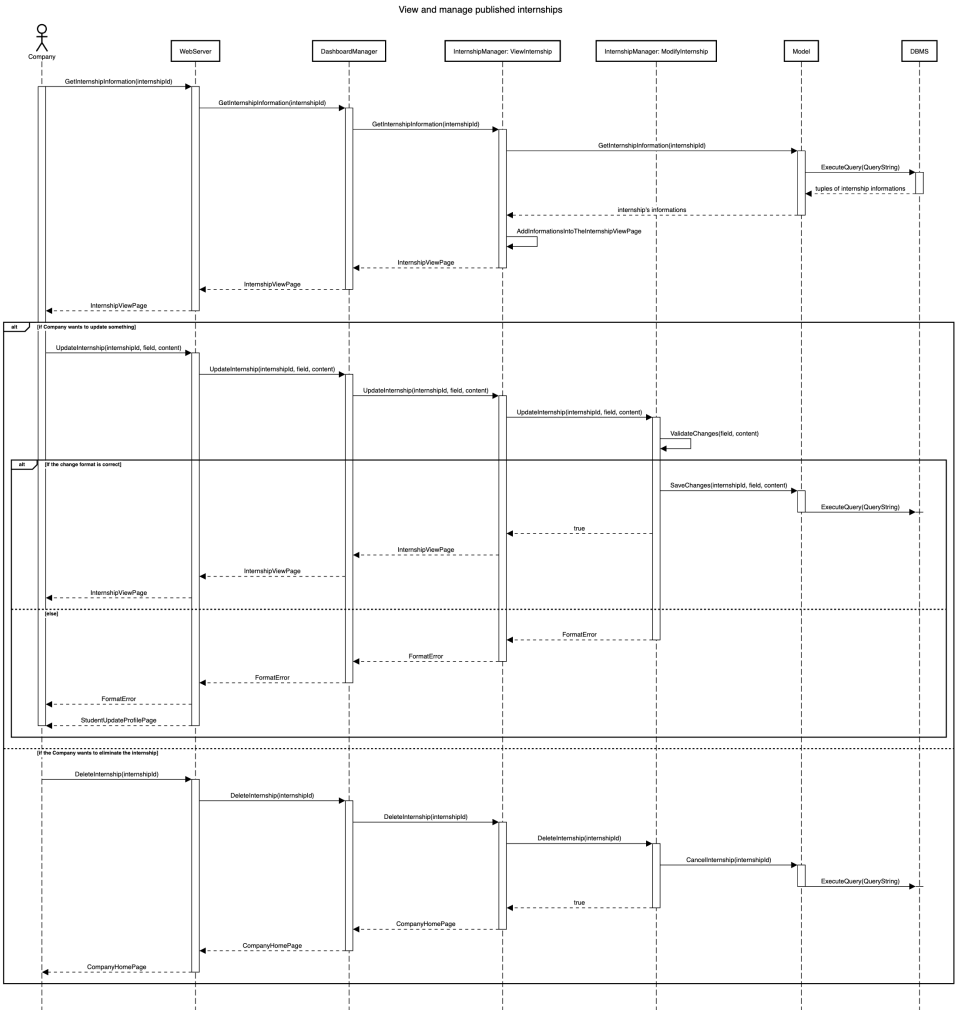


Figure 2.22: Runtime view for 'View and manage published internships'.

The sequence diagram illustrates how a company can view and manage its published internships. First, the company retrieves internship details by providing the internship ID, which is processed by the WebServer and DashboardManager, eventually fetching the information from the Model and DBMS, and displaying it in the Internship View Page. If the company wants to update an internship, they submit the update request to the WebServer, which forwards it to the DashboardManager and the InternshipManager. The InternshipManager validates the changes. If the changes are valid, the new data is saved in the DBMS. If the company wants to delete the internship, they initiate the deletion process, which is handled by the DashboardManager and InternshipManager. The internship is removed from the database, and the company is redirected to the Company Home Page with confirmation of the deletion. If any updates or changes fail due to format

issues, an error message is shown to the company.

2.4.14. View recommended students for internships

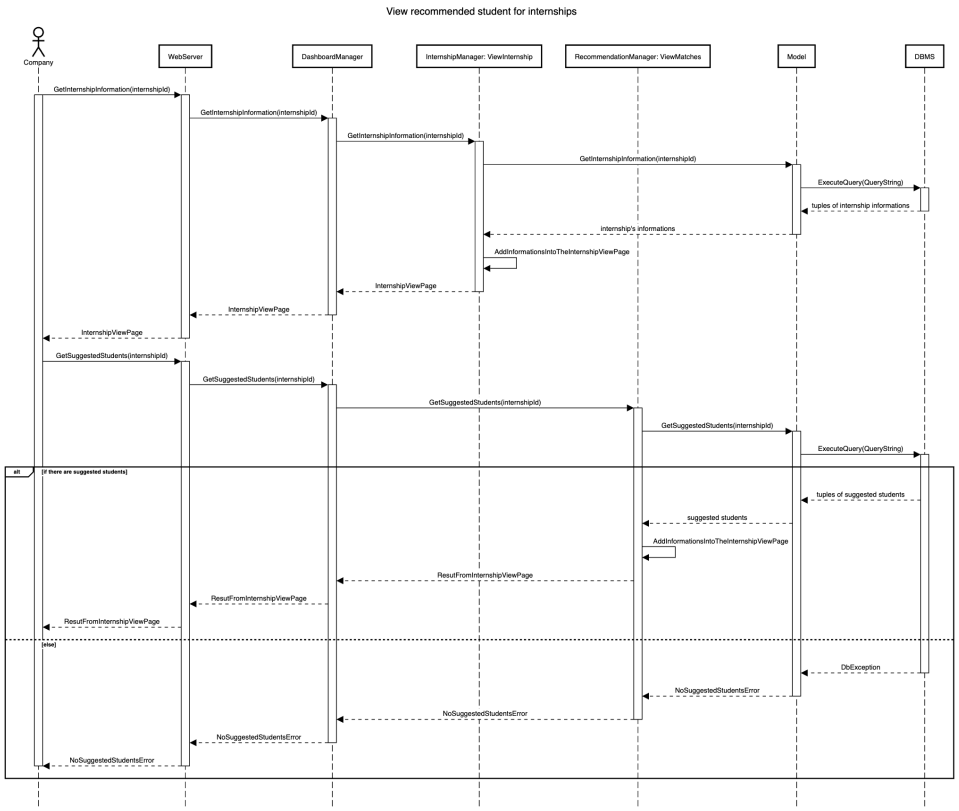


Figure 2.23: Runtime view for 'View recommended students for internships'.

The sequence diagram shows the process where a company views recommended students for internships. Initially, the company retrieves internship details by providing the internship ID. The WebServer and DashboardManager process this request and fetch the relevant internship information from the Model and DBMS, displaying it on the Internship View Page. Next, the company requests a list of suggested students for the internship. This request is forwarded to the RecommendationManager, which retrieves the list of suggested students from the Model and DBMS. If there are suggested students, their details are displayed in the Internship View Page. If there are no suggested students, an error message is shown indicating that there are no suggested students for the internship.

2.4.15. Contact students for internships

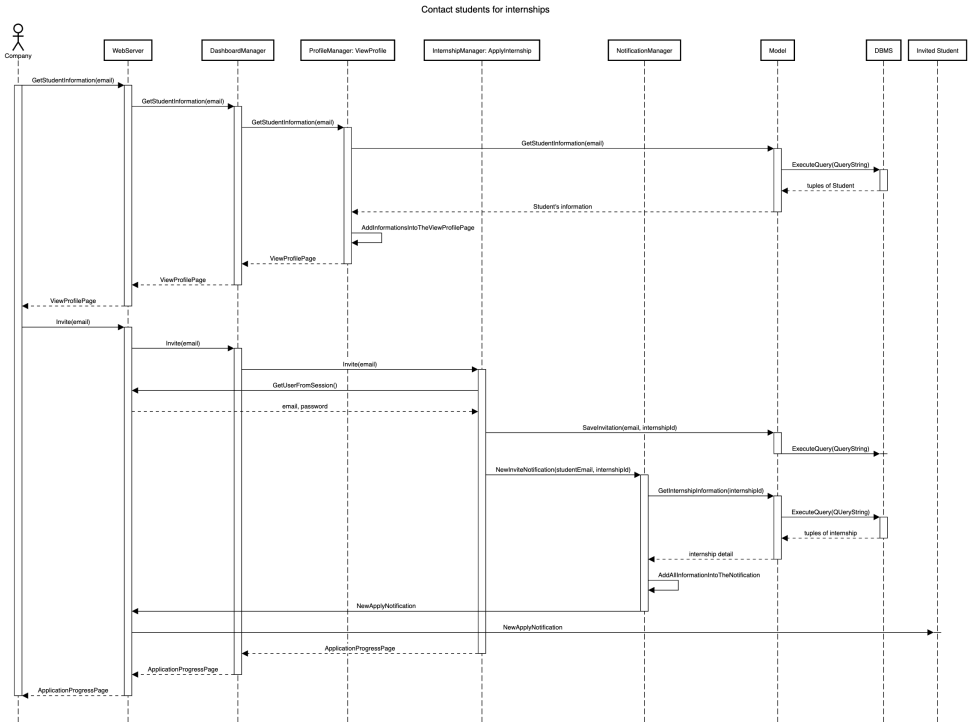


Figure 2.24: Runtime view for 'Contact students for internships'.

The sequence diagram illustrates the process where a company contacts students for internships. First, the company retrieves the student's profile by providing their email. The WebServer, DashboardManager, and ProfileManager fetch the student's information from the Model and DBMS, displaying it on the View Profile Page. After reviewing the profile, the company decides to invite the student for the internship. The WebServer forwards the invitation request to the DashboardManager, which calls the InternshipManager to handle the invitation process. The system validates the session, stores the invitation in the Model, and sends a notification to the student about the internship invitation. The student receives the notification, which includes details of the internship. The company is redirected to the Application Progress Page.

2.4.16. Accept student applications

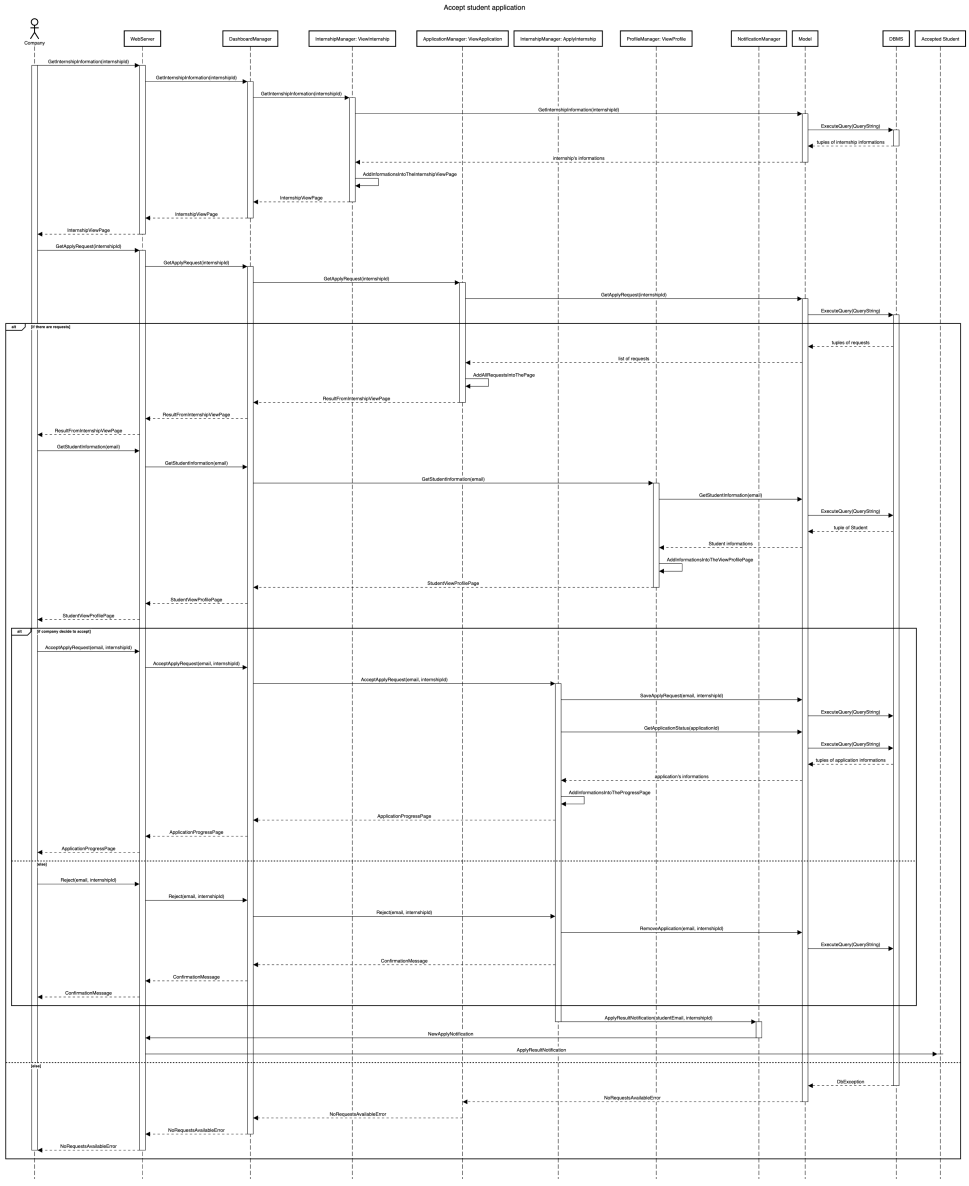


Figure 2.25: Runtime view for 'Accept student applications'.

The sequence diagram illustrates the process of a company managing internship applications. The company retrieves internship details through the WebServer, DashboardManager, InternshipManager, and Model, which interacts with the DBMS. The DashboardManager then displays the information. When reviewing applications, the company requests the list of applicants. This is fetched by the ApplicationManager, which queries the Model and DBMS to retrieve the data. The company can then review the student's profile, with the ProfileManager retrieving student information from the DBMS. If the

company accepts or rejects an application, the decision is passed through the WebServer, DashboardManager, and ApplicationManager. The Model updates the application status, and the NotificationManager notifies the student of the outcome. If no applications are available, the company receives an error .

2.4.17. Submit questionnaires for students

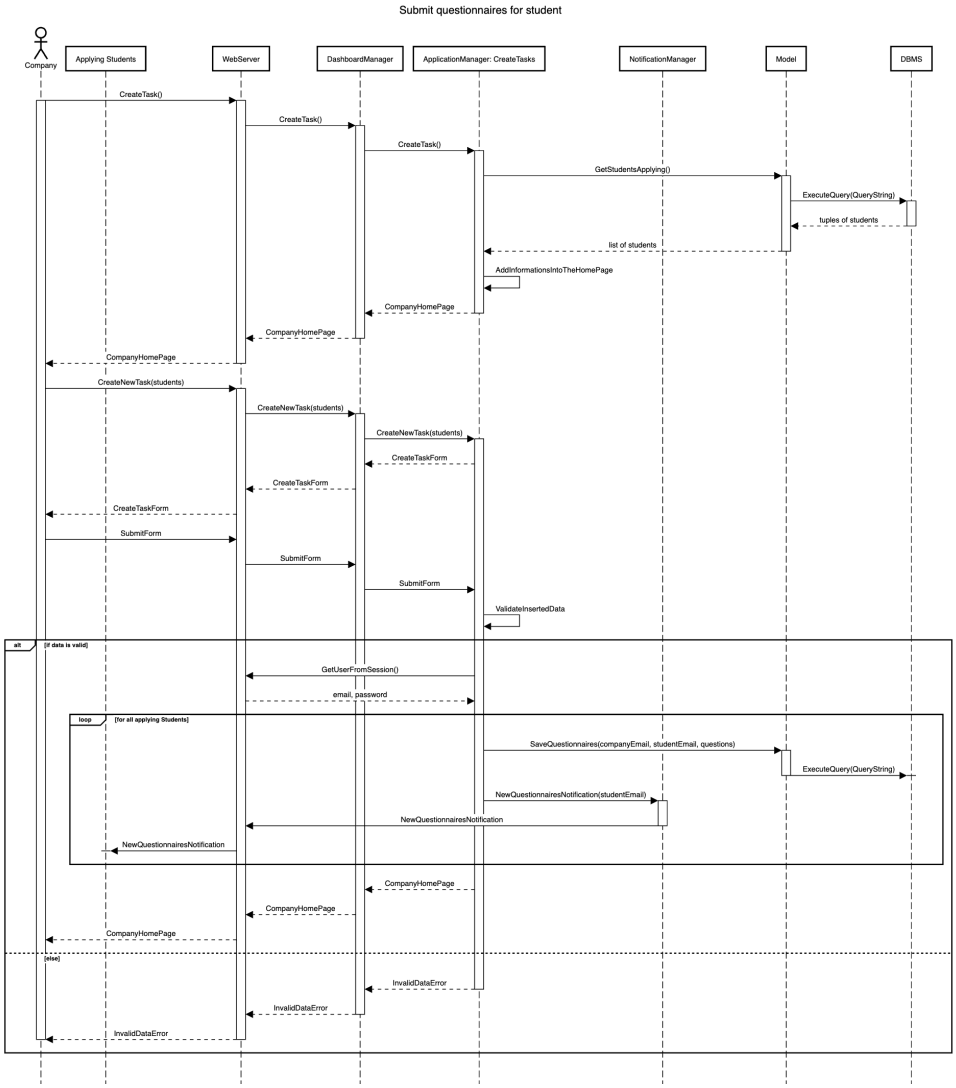


Figure 2.26: Runtime view for 'Submit questionnaires for students'.

The sequence diagram illustrates the process of a company submitting questionnaires to students. The company begins by initiating a task creation request through the WebServer, which forwards it to the DashboardManager and then to the ApplicationManager. The ApplicationManager queries the Model to retrieve a list of students applying for in-

ternships from the DBMS, and this data is displayed on the company's homepage. Next, the company creates a new task and submits a form through the WebServer, which sends the submission to the DashboardManager and ApplicationManager. The form data is validated, and if valid, the questionnaires are saved in the DBMS. The NotificationManager then sends notifications to each student about the new questionnaires. If the data is invalid, the company is informed of the error.

2.4.18. Finalize the selection process

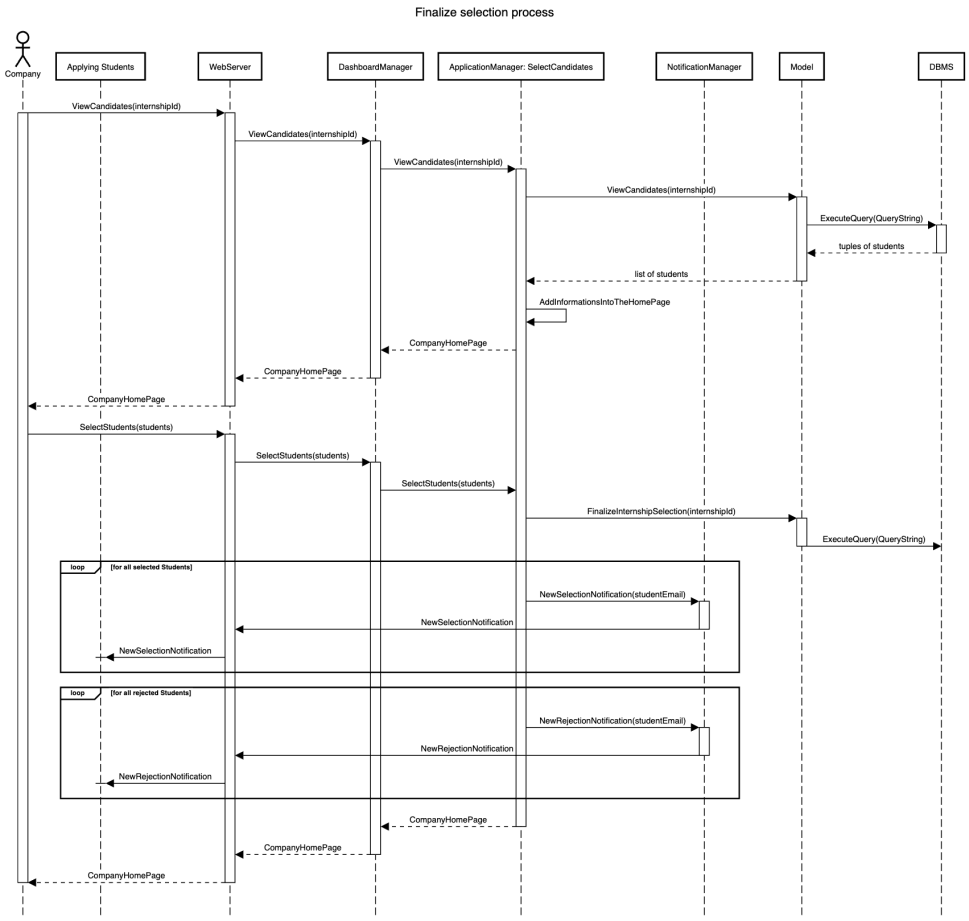


Figure 2.27: Runtime view for 'Finalize the selection process'.

This sequence diagram illustrates the process of finalizing the selection of candidates for an internship. The company starts by viewing the list of candidates through the WebServer, which communicates with the DashboardManager and ApplicationManager to retrieve the list of students from the Model. The Model queries the DBMS for the list of applicants and sends the results back to be displayed on the company's homepage. Once the company selects the students, this information is forwarded to the ApplicationManager, which

finalizes the selection in the DBMS. Notifications are then sent to both selected and rejected students via the NotificationManager. After the notifications are dispatched, the company's homepage is updated and displayed back to the company, completing the selection process.

2.4.19. Provide feedback on interns

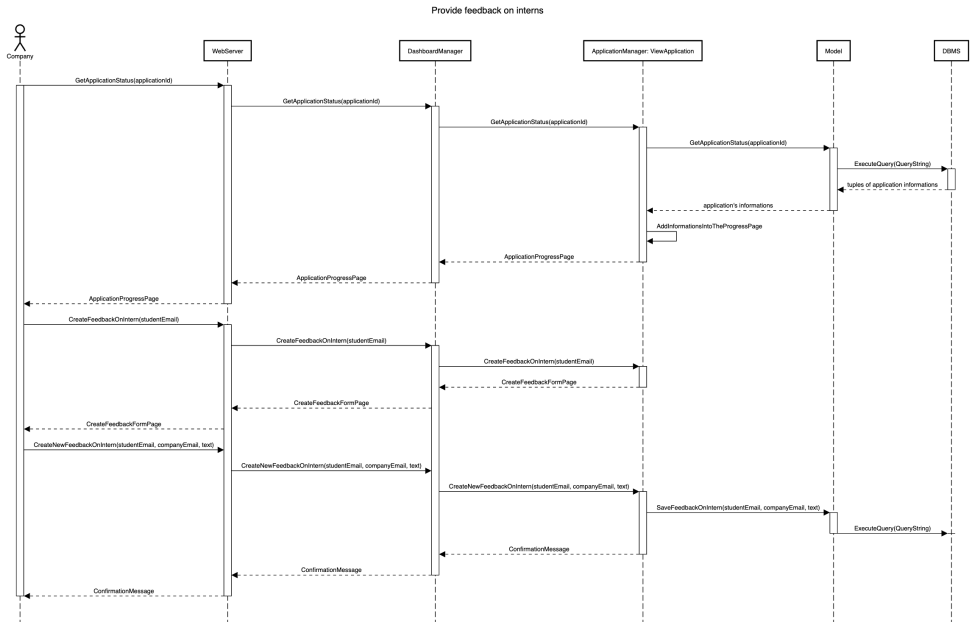


Figure 2.28: Runtime view for 'Provide feedback on interns'.

This sequence diagram illustrate the process for providing feedback on interns. The company begins by checking the application status through the WebServer, which forwards the request to the DashboardManager and the ApplicationManager to retrieve application details from the Model. The Model queries the DBMS for the application status and returns the relevant information, which is displayed on the company's progress page. The company initiates the feedback creation and is presented with a form. Next, the company submits the feedback for a selected intern by submitting the intern's email, company email, and feedback text. This request flows from the WebServer to the DashboardManager and the ApplicationManager, where the feedback is saved in the Model. The feedback is then stored in the DBMS. Once the feedback is saved, a confirmation message is displayed back to the company, completing the feedback submission process.

2.4.20. File complaints about students

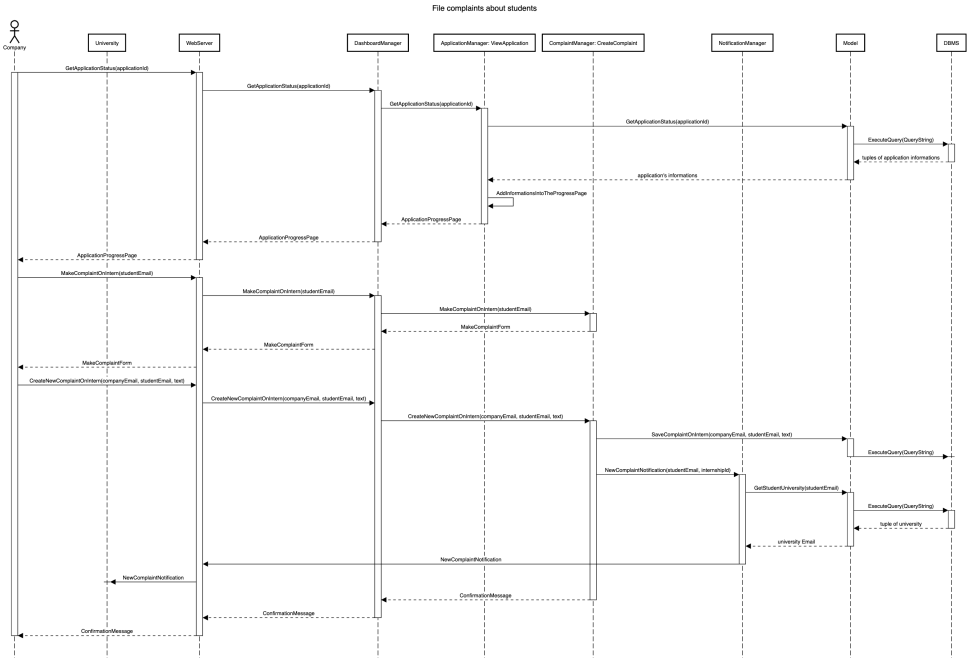


Figure 2.29: Runtime view for 'File complaints about students'.

This sequence diagram represents the process of filing a complaint, from a CP to a ST. The CP navigates to the application progress page of the student, so it can click on the button "create complaint", which creates a requests that is sent to the DashboardManager and redirected to the complaint manager for retrieving the pop-up complaint form. The CP can now fill the form with all the necessary information, such as a brief description of what is not going as expected. The form is sento to the DashboardManager with all the necessary parameters (company email, student email and the text description), that redirects the request to the ComplaintManager module, which manage to save the complaint in the DBMS by making a request to the Model module, and also to create a new Complaint Notification that will be sent to the ST university, with a confirmation receipt for the CP.

2.4.21. Monitor the status of all internships

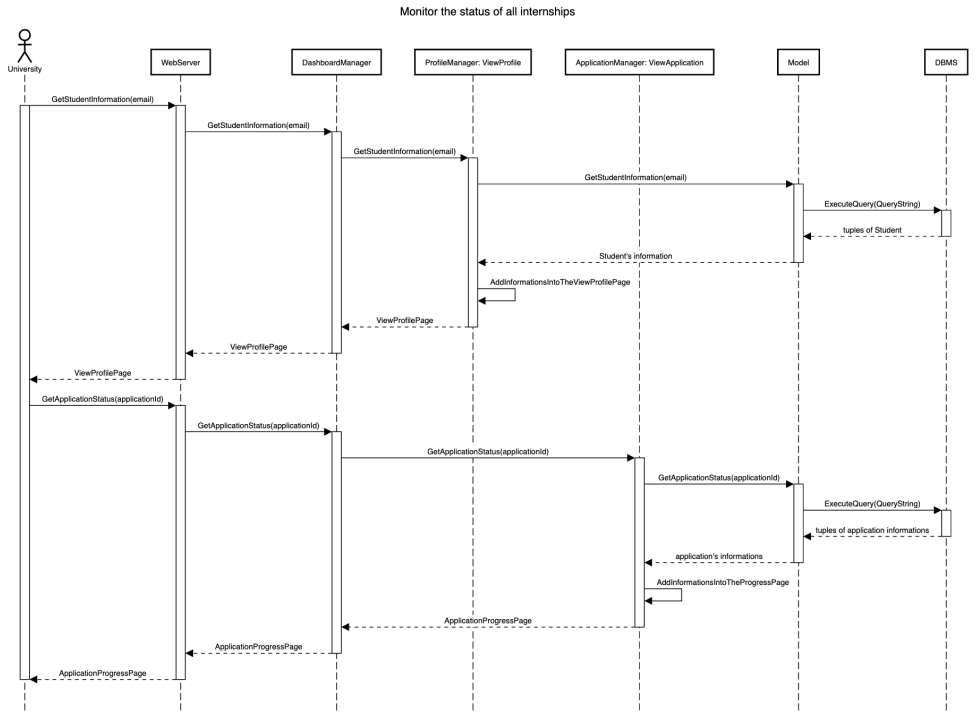


Figure 2.30: Runtime view for 'Monitor the status of all internships'.

This sequence diagram describes how a UV can monitor the status of all of their student involved in internships. The UV on it's home page, click on a view profile button related to a student, so a request of a ST profile is created and handled by the DashboardManager, that forwards it to the ProfileManager, which manages to retrieve the useful information on the DBMS by making a request to the Model module, who is going to create a query for the Database and send back to the ProfileManager all the needed information, adding them to the ViewProfilePage that will be sent directly to the University. Landed on that page, the University can click on the Application Progress Button for viewing the Application Progress Page. After the click, a request is sent to the DashboardManager, which forwards the request to the ApplicationManager, that through the model retrieve the data of the Student about his progression on the internship, adds them to the Application Progress Page and sends back the page to the university.

2.4.22. Handle complaints from students or companies

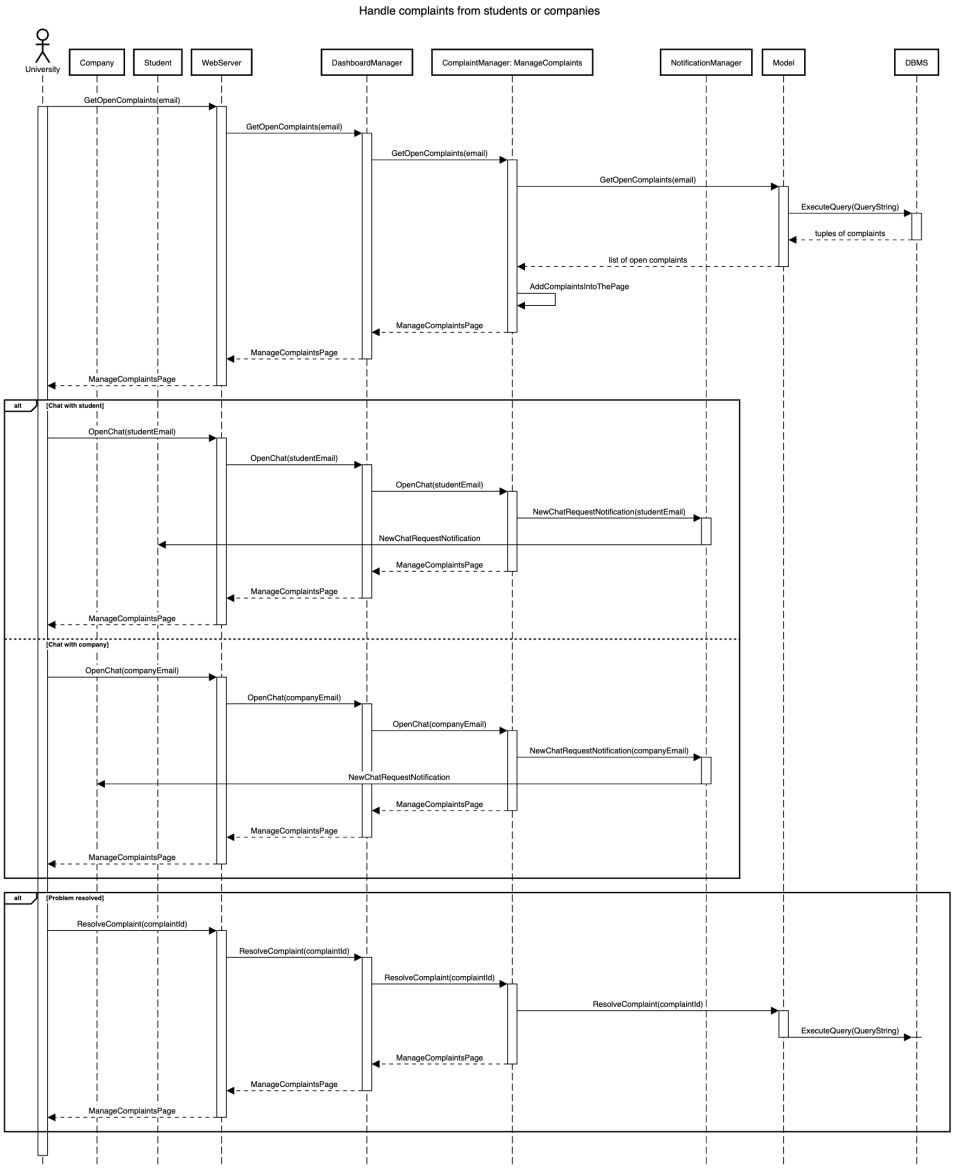


Figure 2.31: Runtime view for 'Handle complaints from students or companies'.

This sequence diagram represents the process of handling any complaint. The university by clicking on the "complaint section" button generates a requests with the university's email as parameter, to the DashboardManager, which forwards it to the ComplaintManager. After that, the ComplaintManager asks directly to the Model module to retrieve the data from the DBMS, and after receiving them, manages to dynamically create the page, that will be sent back to the university passing through the DashboardManager. At this point, the University can see the list of all complaints made from students or companies. In any case of complaints, if the University wants to resolve the problem, it

should open the chat with the complaining part, and does that by clicking on the "open chat" button, which generates a request with the user email as a parameter, that is sent to the DashboardManager and redirected to the ComplaintManager. This module now has the task to notify the complaining part about a request of opening chat by the University, and does that by sending a request to the NotificationManager with the user email as a parameter, so this component can directly send to that user a NewChatRequest notification. When the problem is solved, the University can click on the "Close Complaint" button, so this generates a request to the DashboardManager with the complaintId as a parameter, which forwards it directly to the ComplaintManager, that manage to save the status of the complaint on the DBMS by contacting the Model module.

2.4.23. Cancel problematic internships

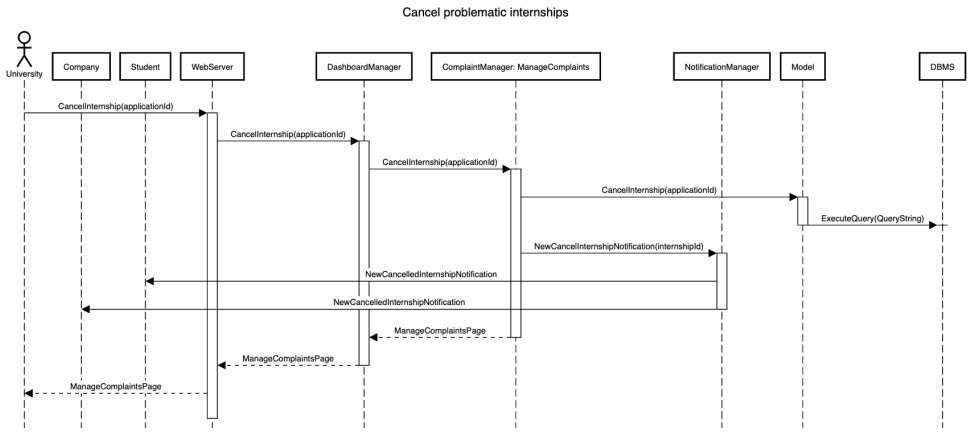


Figure 2.32: Runtime view for 'Cancel problematic internships'.

This sequence diagram represents the process of canceling an internship. When a university is not able to solve a complaint by chatting with the complained part, it can decide to cancel the internship. To do that, the University has to click on the "Cancel Internship" button present on the ManageComplaint page, so this generates a request with the applicationId that is sent to the DashboardManager, that forwards the request to the ComplaintManager, which has the tasks of deleting the relation between the student and the internship on the DBMS by contacting the Model module, and also to notify both the student and the company about the cancellation. For doing so, the ComplaintManager has to send a request to the NotificationManager, which manages to directly send the notification to both parts to inform them about the interruption.

2.4.24. Collect feedback from companies and students to improve matching

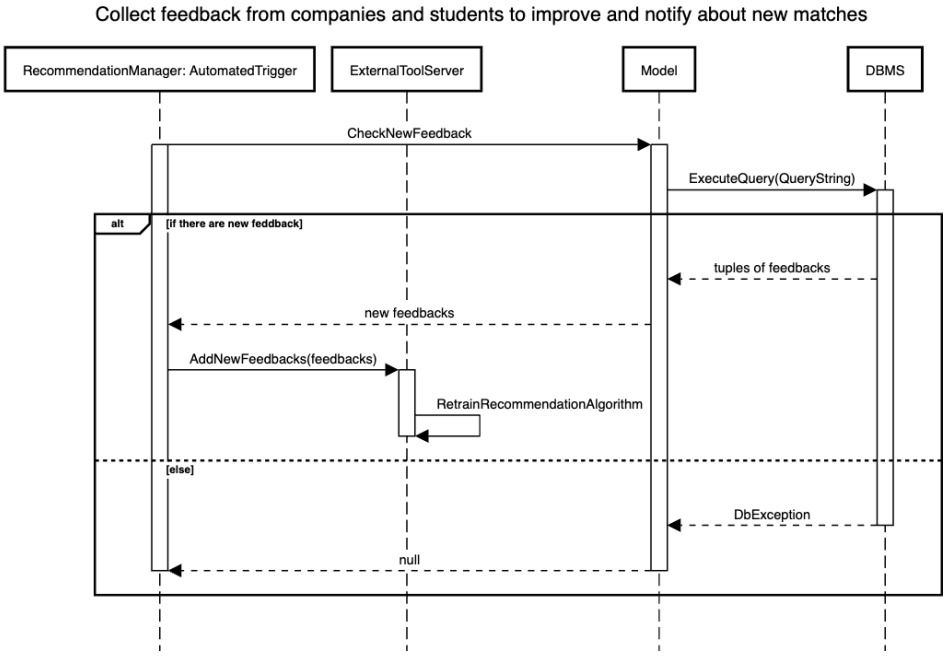


Figure 2.33: Runtime view for 'Collect feedback from companies and students to improve matching'.

This sequence diagram shows flow of collecting feedbacks from the users and improving the recommendation process. Periodically the subcomponent of the RecommendationManager, the AutomatedTrigger, checks for new feedback by creating a request that is sent to the Model. If new feedbacks are found, those are sent to the External Tool Server for retraining the algorithm.

2.4.25. Notify users about a new match

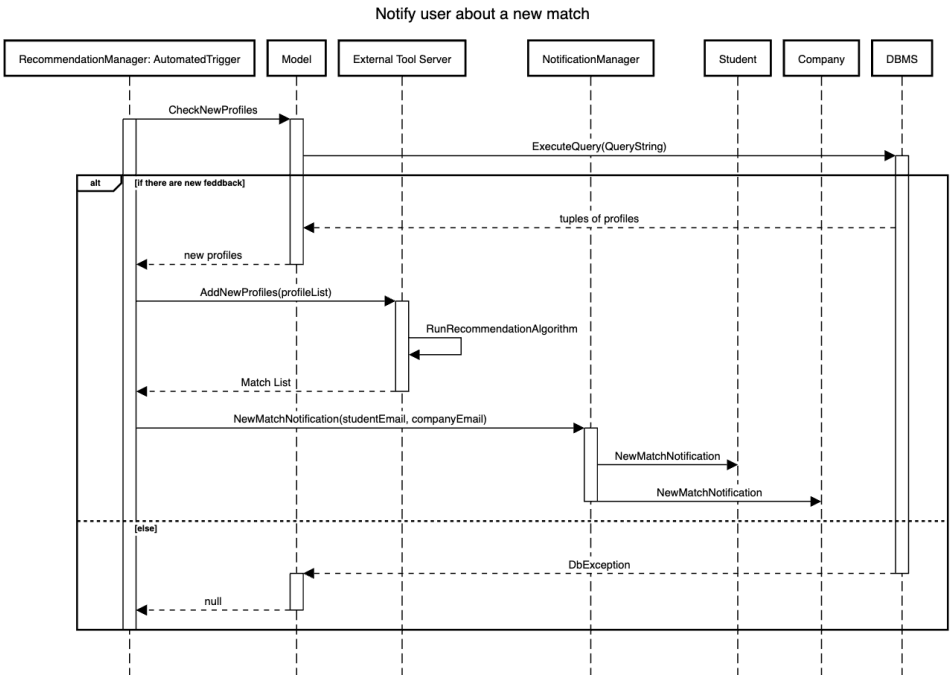


Figure 2.34: Runtime view for 'Notify users about new matches'.

This sequence diagram shows how the Users are notified about a new matching. Periodically, the subcomponent AutomatedTrigger of the RecommendationManager check for new profiles by creating a request to the Model module. If new profiles were inserted after the last check, the DBMS return to the model all the tuples of profiles that were inserted, resending them to the AutomatedTrigger. After that, the component can proceed to trigger the algorithm on the External Tool Server by sending to it the list of new users profiles, so the algorithm runs and the output is sent back to the Automated Trigger. Now the trigger is left with the only task to notify the users involved in the new matchings, and does that by creating a request to the NotificationManager, with both student and company email as parameters. If at the initial query of checking for new profiles returns null, that means that no new profiles were inserted from the last check, so there is no reason to run again the algorithm.

2.5. Component Interfaces

- **WebServer** (contains all the interfaces belonging to other managers, apart from NotificationManager and Model)
  - Request(login)
  - AddUserToTheSession(*String* email, *String* password)
  - Logout()
  - GoToUpdateProfilePage()
  - GetUserFromSession()
- **DashboardManager** (contains all the interfaces belonging to other managers, apart from NotificationManager and Model)
- **Model**
  - CheckCredentials(*String* email, *String* password)
  - GetPublishedInternships(*String* email)
  - GetStudentRecommendedInternships(*String* email)
  - GetUniversityEngagedStudents(*String* email)
  - CheckEmail(*String* email)
  - SaveNewUserCredentials(String name, *String* surname, *String* email, *String* password)
  - GetUser(*String* email)
  - SaveChanges(*String* email, *String* field, *String* content)
  - GetInternshipFromSearch(*String* keyword)
  - GetInternshipInformation(*String* internshipId)
  - SaveApplyRequest(*String* email, *String* password)
  - GetInternshipCompany(internshipId)
  - GetStudentApplicaitions(*String* email)
  - GetApplicationStatus(*String* applicationId)
  - GetStudentOffers(*String* email)

- StartApplication(*String* email, *String* internshipId)
- RemoveApplication(*String* email, *String* internshipId)
- SelectTask(*String* taskId)
- SaveAnswers(*String* taskId, *List<String>* answers)
- GetCompletedInternships(*String* email)
- SaveFeedback(*String* email, *String* internshipId, *String* text)
- GetAppliedInternship(*String* email)
- SaveComplaintOnInternship(*String* email, *String* internshipId, *String* text)
- SaveComplaint(*String* companyEmail, *String* studentEmail, *String* text)
- GetStudentUniversity(*String* email)
- SaveNewInternship(*String* companyEmail, *String* name, *String* applicationDomain, *String* tasks, *String* technologies, *String* terms)
- GetInterestedStudents(*String* applicationDomain, *String* tasks, *String* technologies, *String* terms)
- SaveChanges(*String* internshipId, field, content)
- RemoveInterneship(*String* internshipId)
- GetSuggestedStudents(*String* internshipId)
- SaveInvitation(*String* sutendEmail, *String* internshipId)
- GetApplyRequest(*String* internshipId)
- GetStudentInformation(*String* email)
- GetStudentsApplying()
- SaveQuestionnaires(companyEmail, studentEmail, questions)
- ViewCandidates(*String* internshipId)
- FinalizeInternshipSelection(*String* internshipId)
- SaveFeedbackOnIntern(*String* studentEmail, *String* companyEmail, *String* txt)
- SaveComplaintOnIntern(*String* companyEmail, *String* studentEmail, *String* text)

  - GetOpenComplaints(*String* universityEmail)
  - ResolveComplaint(*String* complaintId)
  - CancelInternship(*String* applicationId)
  - CheckNewFeedback()
  - CheckNewProfiles()
- **ViewProfile**
  - SelectUser(*String* email)
- **LoginManager**
  - Login(*String* email, *String* password)
- **RegistrationManager**
  - CreateAnAccount()
  - Registration(*String* name, *String* surname, *String* email, *String* password)
- **EmailServer**
  - SendEmail(*String* email)
- **ProfileManager**
  - SelectUser(*String* email)
  - UpdateProfile(*String* email, *String* field, *String* content)
  - ValidateChanges(*String* field, *String* content)
- **InternshipManager**
  - SearchInternship(*String* keyoword)
  - Apply(*String* internshipId)
  - Accept(*String* email, *String* internshipId)
  - Reject(*String* email, *String* internshipId)
  - ProvideFeedback(*String* internshipId)
  - CreateNewFeedback(*String* email, *String* internshipId, *String* text)
  - CreateInternship()

  - PublishNewInternship(*String* name, *String* applicationDomain, *String* tasks, *String* technologies, *String* terms)
  - UpdateInternship(*String* internshipId, *String* field, *String* content)
  - DeleIteInternship(*String* internshipId)
  - Invite(*String* email)

- **ApplicationManager**
  - SubmitQuestionnaire(*List<String>* answers)
  - CreateNewTask(*List<String>* studentEmails)
  - SelectStudents(*List<String>* studentEmails)
  - CreateFeedbackOnIntern(*String* studentEmail)
  - CreateNewFeedbackOnIntern(*String* studentEmail, *String* companyEmail, *String* txt)

- **ComplaintManager**
  - MakeComplaintOnInternship(*String* internshipId)
  - CreateNewComplaintOnInternship(*String* email, *String* internshipId, *String* text)
  - MakeComplaintOnIntern(*String* studentEmail)
  - CreateNewComplaintOnIntern(*String* companyEmail, *String* studentEmail, *String* text)
  - OpenChat(*String* studentEmail)

- **RecommendationManager**
  - InformInterestedStudents(*String* applicationDomain, *String* tasks, *String* technologies, *String* terms)
  - NewMatches()

- **ExternalToolServer**
  - AddNewFeedbacks(*List<String>* feedbacks)
  - AddNewProfiles(List<Student> profiles)

- **NotificationManager**

- NewApplyNotification(*String* email, *String* internshipId)
- NewComplaintNotification(*String* studentEmail, *String* internshipId)
- NewComplaintNotification(*String* companyEmail, *String* studentEmail)
- NewInternshipNotification(*String* companyName, *String* internshipName, *String* applicationDomain, *String* tasks, *String* technologies, *String* terms)
- NewInviteNotification(*String* studentEmail, *String* internshipId)
- NewQuestionnairesNotification(*String* studentEmail)
- NewSelectionNotification(*String* studentEmail)
- NewRejectionNotification(*String* studentEmail)
- NewChatRequestNotification(String email)
- NewCancelInternshipNotification(*String* internshipId)
- NewMatchNotification(*String* studentEmail, *String* companyEmail)

2.6. Selected Architectural Styles and Patterns

This section describes the key architectural styles and patterns chosen for the development of the system. The architecture of the application is structured to ensure scalability, maintainability, and ease of development. The three primary patterns implemented are the 3-tier architecture, Model-View-Controller (MVC) and the use of APIs to facilitate communication between components.

2.6.1. 3-tier Architecture

The system uses a 3-tier architecture so the application is divided into three layers:

- **Presentation Layer:** This layer handles the user interface (UI) and is responsible for interacting with the users. It receives user input, sends requests to the application layer, and displays the results.
- **Application Layer:** Also known as logic layer, this component processes the input from the presentation layer, applies the necessary logic, and interacts with the data layer.
- **Data Layer:** The data layer is responsible for managing the database and performing CRUD (Create, Read, Update, Delete) operations. It provides data to the

application layer as requested.

This architectural style enhances modularity and allows each layer to be developed and maintained independently, ensuring better scalability and fault isolation.

2.6.2. Model-View-Controller (MVC)

The application follows the Model-View-Controller (MVC) pattern to separate concerns and improve code organization. The application is divided into three logical components:

- **Model:** Represents the data and the business logic of the application. It responds to requests for information and updates the state of the application.
- **View:** Responsible for presenting data to the user. The view receives data from the model and displays it to the user in a suitable format.
- **Controller:** Manages user input and interacts with the model. It processes incoming requests, executes the appropriate logic, and returns the result through the view.

This pattern ensures separation of concerns, making the codebase easier to manage and extend.

2.6.3. APIs

The system relies on APIs (Application Programming Interfaces) to facilitate communication between different modules and external services. APIs are used to:

- Enable interaction between the frontend and backend systems.
- Integrate with third-party tools and external databases.

APIs provide a standardized way to access application logic, ensuring modularity, reusability, and flexibility in integrating new features.

2.7. Other Design Decisions

This section explains other design decisions that have been made to improve different aspects of the system.

2.7.1. Availability

Ensuring high availability is crucial for maintaining uninterrupted access to the system. Redundancy, load balancing, and failover mechanisms are implemented to minimize downtime and handle unexpected failures. The system is designed to operate across multiple servers and data centers, providing resilience against localized outages.

2.7.2. Scalability

The architecture is designed to scale horizontally by adding more servers or services as the user base grows. Microservices and containerization are employed to allow seamless scaling of individual components. This ensures the system can handle increasing workloads without significant redesign.

2.7.3. Data Storage

A distributed database system is used to manage large volumes of data while ensuring fault tolerance and data consistency. Replication strategies are implemented to enhance data durability and performance.

2.7.4. Security

Security measures are embedded at every level of the architecture. Data encryption and secure communication protocols (such as HTTPS) are part of the strategy to protect sensitive information. Role-based access control is used to restrict access to specific parts of the system, ensuring that only authorized users can perform certain actions.

3 | User Interface Design

This section outlines the User Interface of the S&C system, providing an overview of the various pages that form the core of the platform. The design mockups presented in this document focus on interaction dynamics and functionality rather than the final visual aesthetics, acknowledging that graphical elements may undergo adjustments during the testing and refinement phases. While the desktop browser version is emphasized due to its alignment with the platform's primary objective of connecting students and companies, equivalent pages will be designed and optimized for the mobile version to ensure a seamless and responsive user experience across devices.

As highlighted in the RASD, these design mockups serve as an initial representation of the interface and are subject to iterative enhancements based on testing outcomes and user feedback, ensuring that the system meets the expectations of its target audience while maintaining usability and efficiency.

3.1. Overview

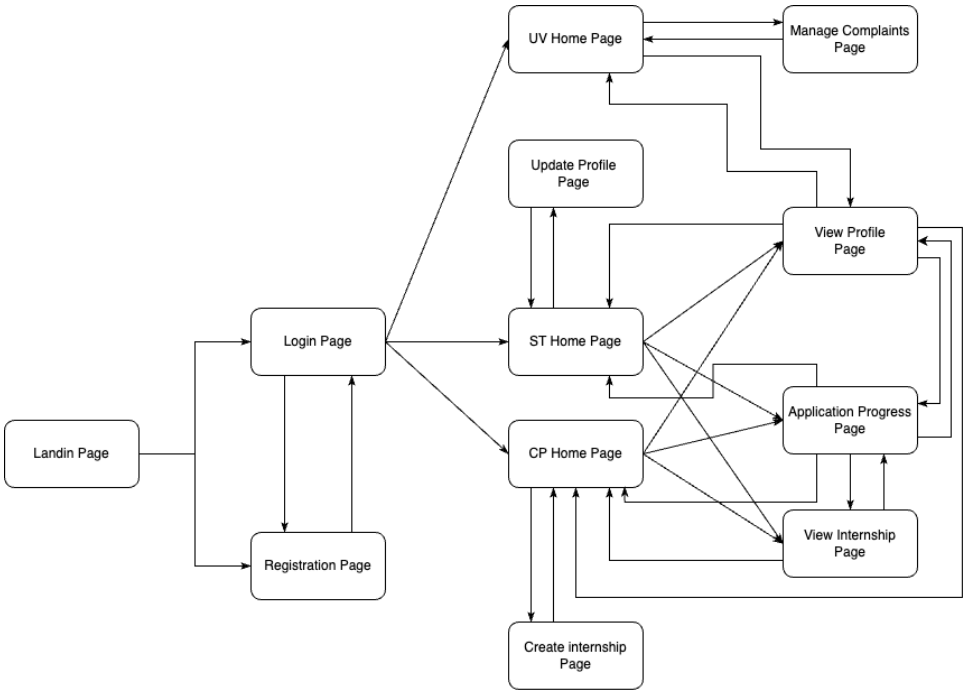


Figure 3.1: User Interface Overview.

The provided diagram presents a comprehensive overview of the S&C system's pages, illustrating their interconnections and the primary pathways for user navigation. Each page serves a specific purpose in enabling seamless interaction between students and companies. Detailed descriptions of each page are provided in the following sections, with the exception of the landing page, which is implicit as it primarily serves as an introduction to the platform and provides access to the login and registration pages.

3.2. Header

Figure 3.2: Header.

Across all pages in the S&C system, a uniform header ensures a consistent and user-friendly experience. The header prominently displays the platform name, a notification icon to keep users updated, a menu icon for easy navigation, and the user's name or

identifier. These elements provide quick access to essential features, enhancing usability across the platform.

In cases where the user is a company or a university, the header undergoes slight modifications to better cater to their role. For instance, instead of the user's name, the company or institution's name is displayed, reflecting their organizational identity. Additionally, the menu options adapt to include functionalities specific to these users, such as "Recommended Students" or "Create Tasks." Despite these changes, the cohesive design of the header remains consistent, ensuring familiarity and accessibility for all users.

3.3. User Interfaces

UI1. Login and Registration Pages

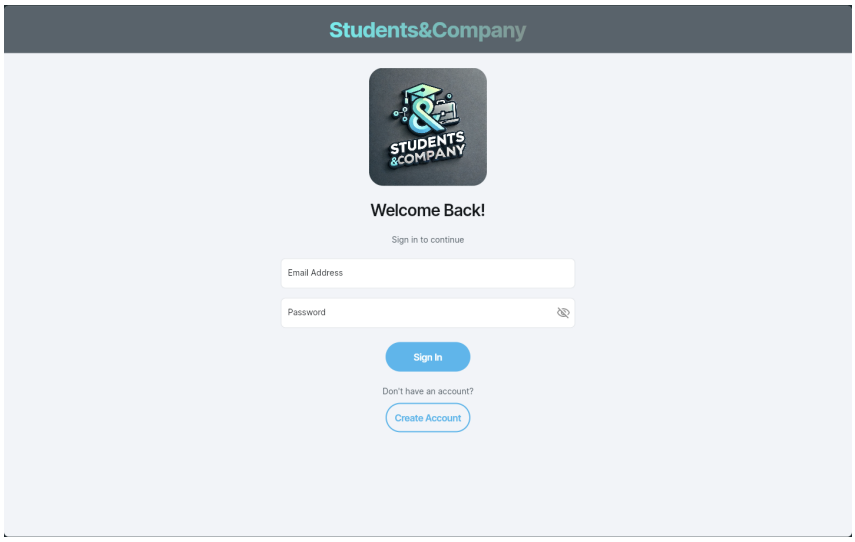


Figure 3.3: Login Page.

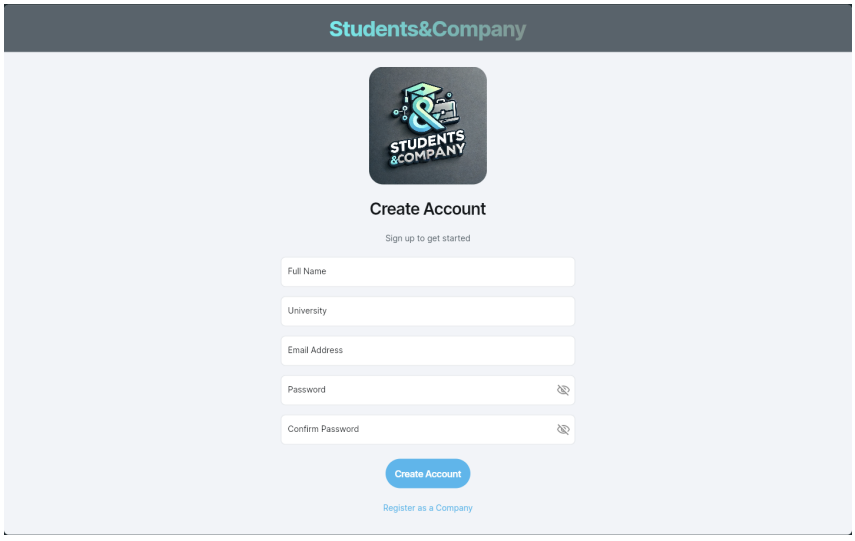


Figure 3.4: Registration Page.

The "Login" and "Registration" pages are simple form pages that allow user to register and sign in. For the Login page, either if the User is a ST or CP or UV, they just need to enter their credentials to login. The registration's steps are a bit different between them, for STs they need to specify their Full name, email, password and the the university they belong to; for both CPs and UVs, they need to specify some other aspects to make sure that not everyone can register as CP or UV.

UI2. ST Homepage

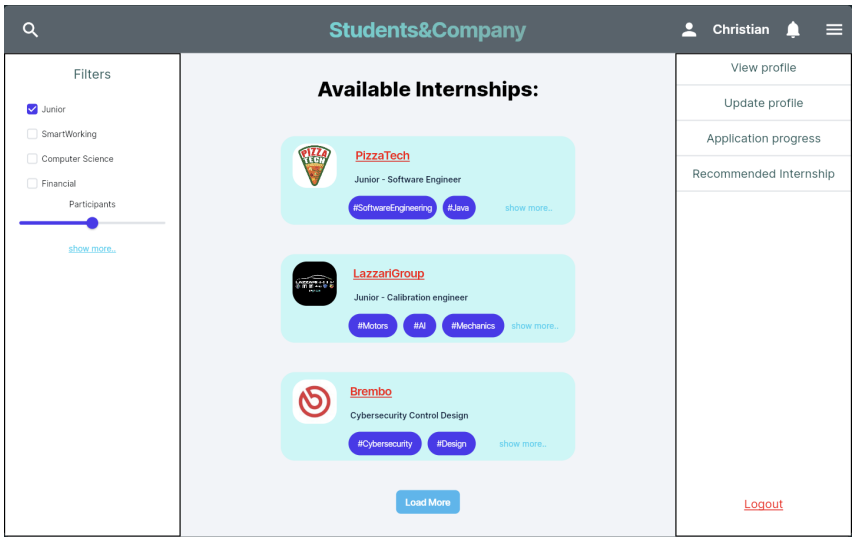


Figure 3.5: ST Homepage.

From the Login Page, the ST will be redirected to the Student Homepage, which is composed by a scrollable feed of internships in the center. On the left side ST can apply filters for a better search of internships, also with the possibility of a keyword search by writing on the lens at the top. On the right there is a drop-down menu that allows ST to view their profile, update it in the case they have not done it yet, the application progress page that will be explained later, and the possibility to Logout.

UI3. Update Profile Page

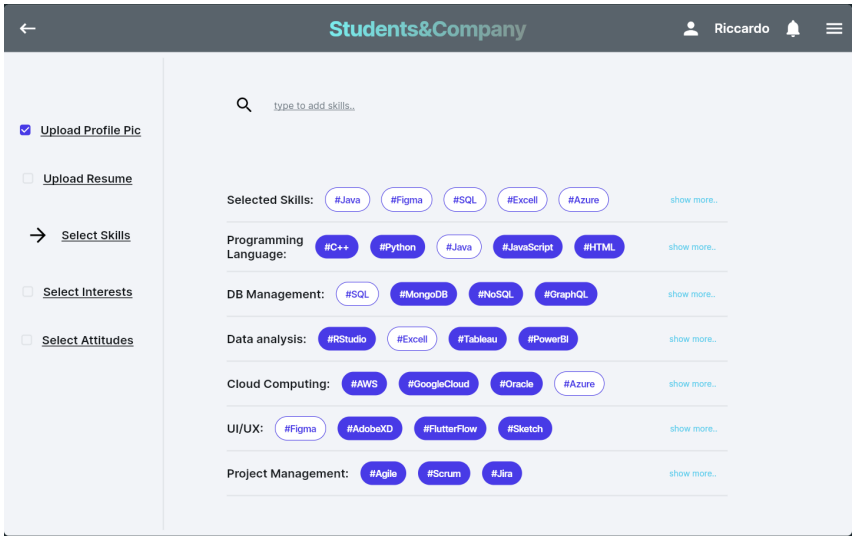


Figure 3.6: Update Profile Page.

The "Update Profile" form page is where the STs can enrich their profile overview by setting some pre-defined parameters that best describe them. This process starts with uploading a profile picture, a resume, select the individual skills, interests and attitude, but anyone can skip any of the steps, knowing that poor profiles are less likely to be matched with the right internship.

UI4. CP Homepage

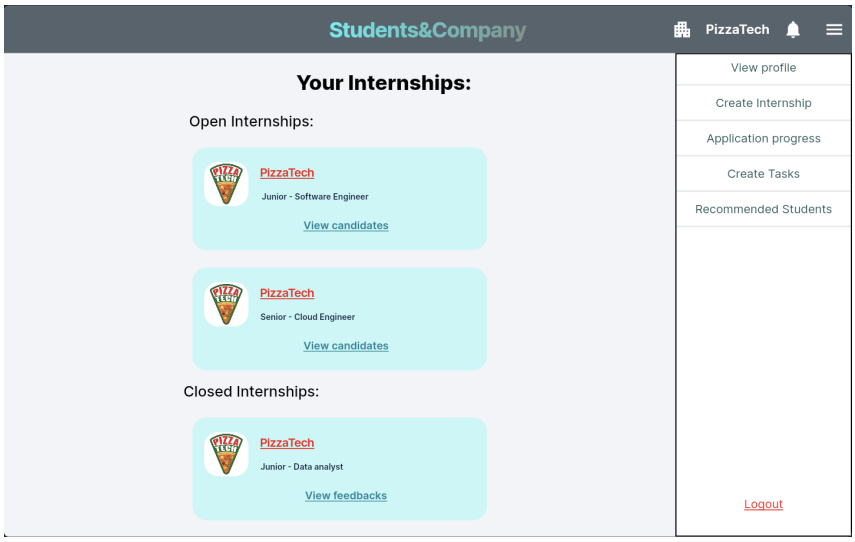


Figure 3.7: CP Homepage.

From the Login Page, the CP will be redirected to the Company Homepage, which is composed by the published internship by them. Clicking on top of an internship redirect to the View Internship page, which provides an overview of everything the CP needs to know.

UI5. Create Internship Page

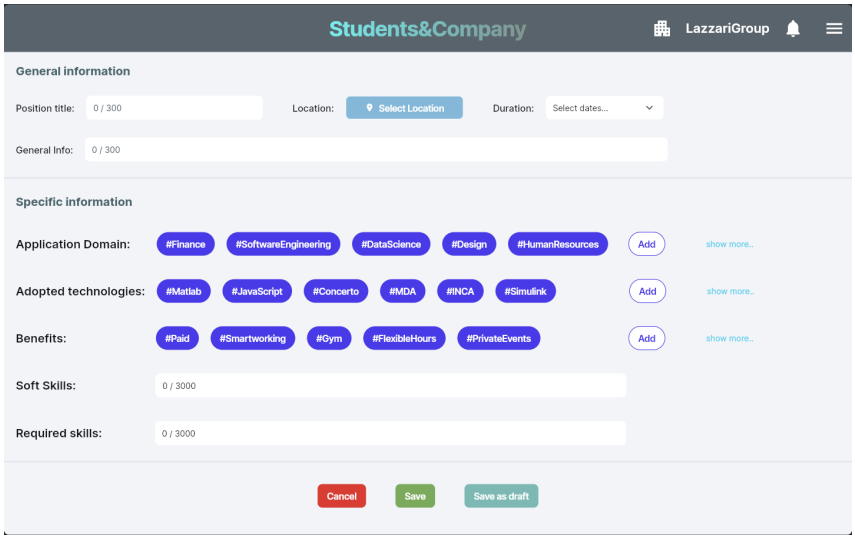


Figure 3.8: Create Internship Page.

In the "Create Internship" page, the CP can start to create a new announcement for an internship, starting by filling the General information forms, which include the position title, location, duration and some other general info. After that, the CP can start filling in the specific information forms by selecting some of the predefined tags that could be inherent to the internship they're proposing. At any moment the CP can cancel, pause or publish this process by clicking on the "Cancel", "Save as Draft" or "Publish" buttons at the bottom of the page.

UI6. View Profile Page

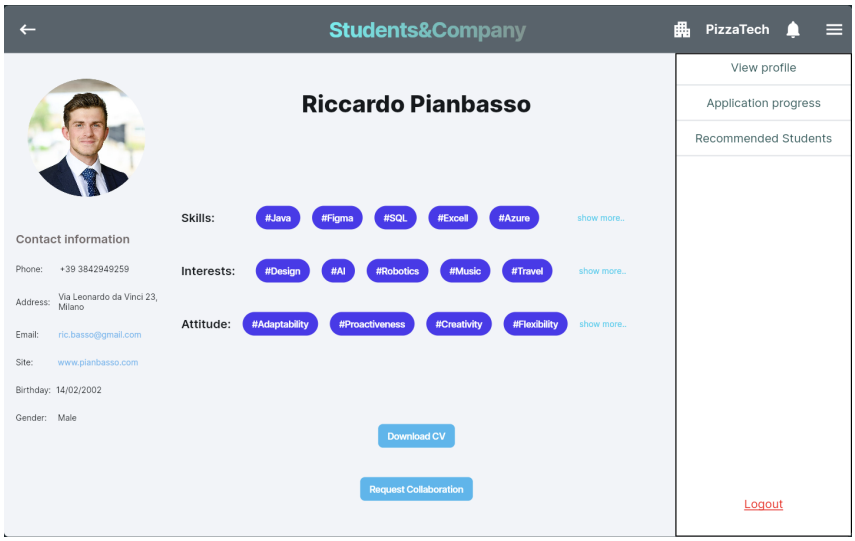


Figure 3.9: View Profile Page.

The "View Profile" page provides an overview of the student, and based on the User that is visualizing this page, some functionality may appear. In this case the company PizzaTech has received a matching notification for this student, and now has the possibility to contact him by clicking on the "Request Collaboration" button.

UI7. View Internship Page

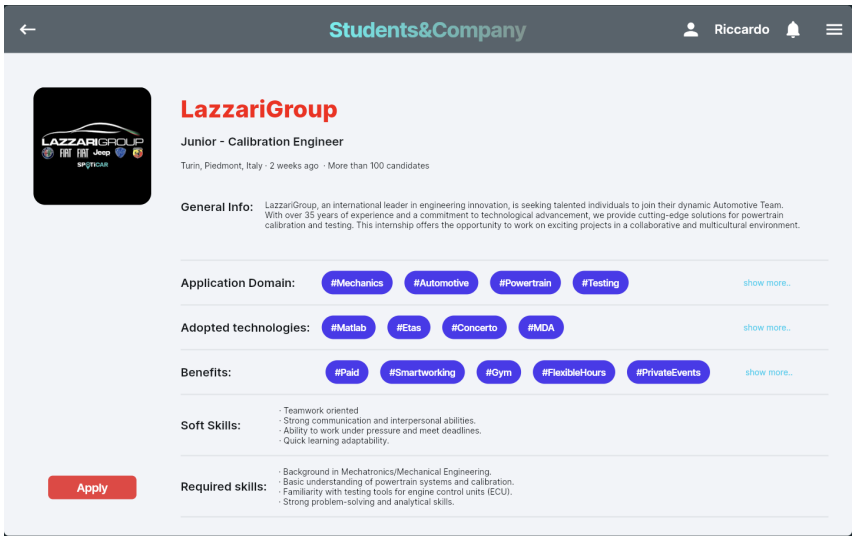


Figure 3.10: View Internship Page.

The "View Internship" page displays all the general and specific information of an internship. Depending on the User who visualizes the page, some changes may appear, as in this case is present the "Apply" button is present, which appears when a ST views this page. In case it is the publishing company that visualizes this page, it can also modify the parameters and the status of the internship.

UI8. Application Progress Page

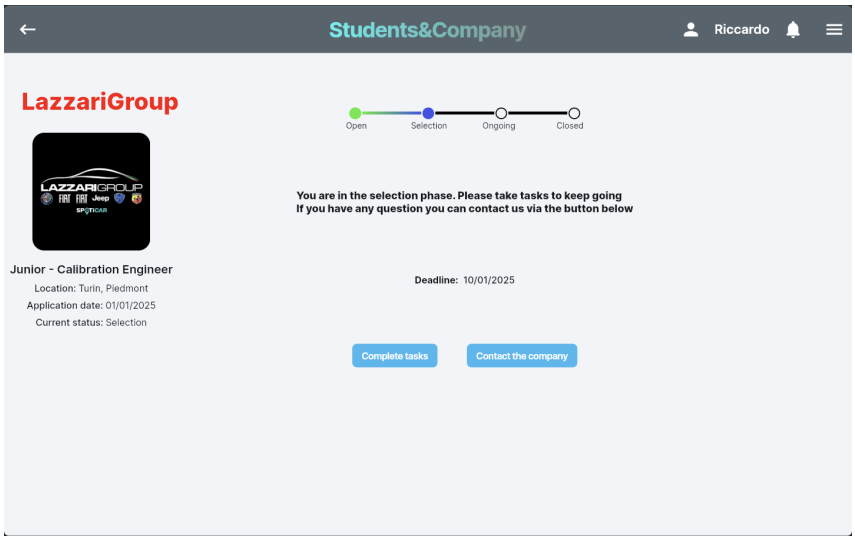


Figure 3.11: Application Progress Page.

The "Application Progress" page allows both Students and Companies to track the status of a specific internship for a student. For a Student, this page provides a clear visualization of the progress of their internship journey, categorized into four distinct phases:

- **Open**: This status represents internships that are still in the initial phase, where applications have been submitted but the selection process has not yet begun.
- **Selection**: In this phase, the company evaluates candidates by assigning tasks or conducting interviews to determine the most suitable applicants.
- **Ongoing**: This is the active phase of the internship, where the student is already working with the company.
- **Closed**: The final phase, marking the completion of the internship. At this stage, students can provide feedback regarding their experience.

This page provides an intuitive and structured way for users to understand the progress of each application at a glance.

UI9. University HomePage

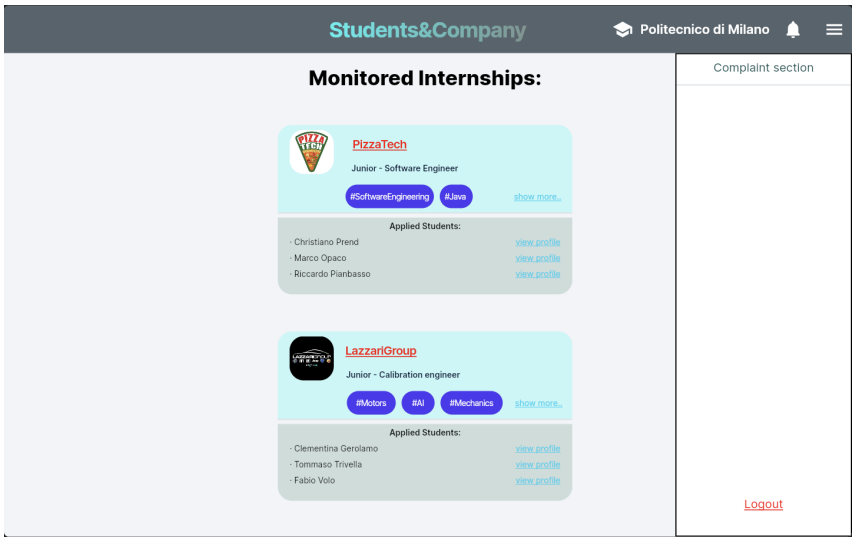


Figure 3.12: University Homepage.

From the Login Page, the UV will be redirected to the University Homepage, where the status of all student's internship can be monitored. After clicking on a view profile for a specific student, the UV can reach the Application Progress page through the View Profile Page. Here it can view the full journey between the Student and it's internship. Also, when a complaint is made, a notification will be received through the notification icon in the header, or the UV can check if there are any complaints on the Manage Complaint Page by clicking on the button placed in the right-side menu.

UI10. Manage Complaints Page

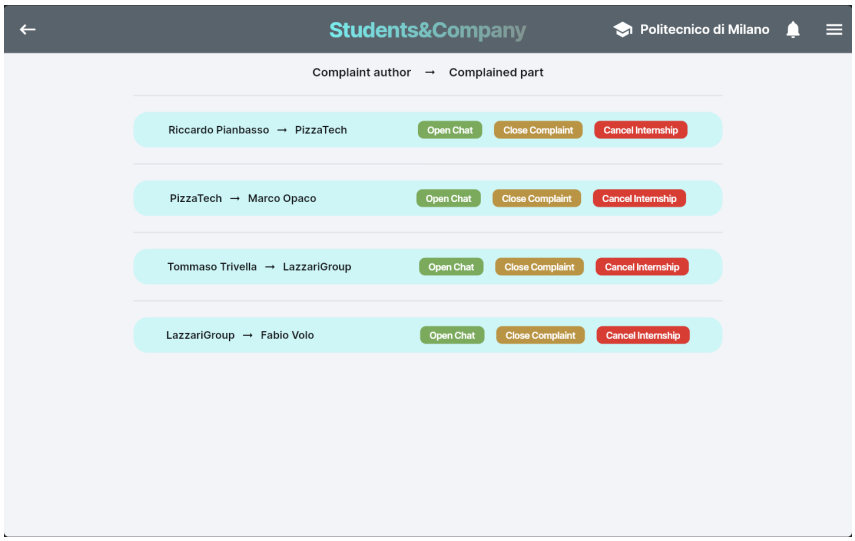


Figure 3.13: Manage Complaints Page.

From the University Homepage, the UV can reach this Manage Complaints Page, where all the filed complaints are listed there. When a new complaints is made, the UV can ask the complaint author to open a chat to have a better understanding on the situation. After that, the company can decide whether to Mark the complaint as solved or Close the internship.

4 | Requirements Traceability

In this section every component explained in this document is mapped to the correct requirements defined in the RASD.

Model:

- R1: The system allows STs to register by providing personal information, email, and a password.
- R2: The system shall allow Users to log in using their credentials.
- R3: The system shall allow STs to edit their profiles, including personal details and contact information.
- R5: The system shall allow CPs to post new internship opportunities, including position details, required skills, duration, and compensation.
- R7: The system shall notify CPs when STs apply to an internship.
- R16: The system shall allow STs to search for internships using keywords, filters, or location.
- R17: The system shall recommend internships to STs based on their skills, preferences, and past applications.
- R18: The system shall recommend STs to CPs for specific internships based on their skills, preferences, and past applications.
- R19: The system shall allow CPs to search for suitable STs based on their skills, academic background, and CVs.
- R20: The system shall maintain a database of all internships, applications, and evaluations.

Dashboard Manager:

- R1: The system allows STs to register by providing personal information, email, and a password.
- R2: The system shall allow Users to log in using their credentials.
- R3: The system shall allow STs to edit their profiles, including personal details and contact information.
- R5: The system shall allow CPs to post new internship opportunities, including position details, required skills, duration, and compensation.
- R6: The system shall allow STs to apply for internships by submitting a CV and optional cover letter.
- R8: The system shall allow CPs to review applications and schedule interviews with STs.
- R9: The system shall allow CPs to submit questionnaires to STs.
- R10: The system shall allow CPs to accept or reject applications and notify STs of the decision.
- R11: The system shall allow UVs to manage their students and monitor their activity.
- R13: The system shall allow CPs to track the progress of STs during the internship and provide regular feedback.
- R15: The system shall allow UVs to review the feedback and evaluation provided by both CPs and STs.
- R16: The system shall allow STs to search for internships using keywords, filters, or location.
- R19: The system shall maintain a database of all internships, applications, and evaluations.

Login Manager:

- R2: The system shall allow users to log in using their credentials.
- R4: The system shall enforce role-based access control to restrict access based on user roles.

Registration Manager:

- R1: The system shall allow STs to register by providing personal information, email, and a password.

Profile Manager:

- R3: The system shall allow STs to edit their profiles, including personal details and contact information.
- R19: The system shall allow CPs to search for suitable STs based on their skills, academic background, and CVs.

Internship Manager:

- R5: The system shall allow CPs to post new internship opportunities, including position details, required skills, duration, and compensation.
- R7: The system shall notify CPs when STs apply to an internship.
- R14: The system shall allow CPs to evaluate the performance of STs at the end of the internship.
- R15: The system shall allow UVs to review the feedback and evaluation provided by both CPs and STs.
- R20: The system shall maintain a database of all internships, applications, and evaluations.

Application Manager:

- R6: The system shall allow STs to apply for internships by submitting a CV and optional cover letter.
- R8: The system shall allow CPs to review applications and schedule interviews with STs.
- R9: The system shall allow CPs to submit questionnaires to STs.
- R10: The system shall allow CPs to accept or reject applications and notify STs of the decision.
- R13: The system shall allow CPs to track the progress of STs during the internship and provide regular feedback.

Recommendation Manager:

- R17: The system shall recommend internships to STs based on their skills, preferences, and past applications.
- R18: The system shall recommend STs to CPs for specific internships based on their skills, preferences, and past applications.
- R20: The system shall maintain a database of all internships, applications, and evaluations.

Complaints Manager:

- R11: The system shall allow UVs to manage their students and monitor their activity.
- R15: The system shall allow UVs to review the feedback and evaluation provided by both CPs and STs.
- R20: The system shall maintain a database of all internships, applications, and evaluations.

Notification Manager:

- R7: The system shall notify CPs when STs apply to an internship.
- R12: The system shall send notifications to users for events such as application submissions, approvals, or rejections.
- R17: The system shall recommend internships to STs based on their skills, preferences, and past applications.
- R18: The system shall recommend STs to CPs for specific internships based on their skills, preferences, and past applications.

5 | Implementation, Integration and Test Plan

5.1. Overview and Implementation Plan

This section explains how the Students&Companies (S&C) platform will be built, integrated, and tested. The development will happen in phases to ensure that each part of the system is created and tested individually before being connected to the rest. This way, problems can be caught early, and we can avoid larger issues later on.

The implementation will use a combination of two approaches:

- **Bottom-up** – We'll start by developing and testing the essential features of the systems that do not require other functionalities to work.
- **Thread-based** – Once the core features are in place, we'll add new features in parallel. This keeps development moving quickly and gives stakeholders something tangible to see along the way.

By using this hybrid method, different teams can work on separate features at the same time, integrating them as soon as they are ready. This not only speeds up development but also reduces the risk of big issues when the full system comes together.

Continuous feedback from stakeholders will be crucial throughout development.

Alpha Testing will be conducted internally by developers and select testers to identify major bugs early in the process. **Beta Testing** will involve releasing the platform to a limited audience to gather real-world feedback before full deployment.

During the testing phases, logs and reports will be collected to track issues, prioritize fixes, and improve overall system stability. This iterative approach ensures that the platform evolves based on real user interactions, allowing the team to fine-tune the system progressively.

5.2. Features Identification

The system's core features are extracted from the functional and non-functional requirements outlined in the RASD document. These features define how users will interact with the system and how different components will work together to provide a seamless experience. Below is a breakdown of the main features that will be developed, integrated, and tested to ensure the platform meets user expectations and operates smoothly.

[F1] User Authentication (Sign-up, Login, Logout). This is one of the fundamental features of the platform. It allows users to create an account, log in, and log out securely. Authentication is essential for accessing any part of the system, ensuring that users' data is protected and only authorized individuals can perform specific actions. The platform will support multiple user roles, including students, companies, and university staff, each with different levels of access and privileges. Testing this feature will focus not only on the correctness of login and registration flows but also on ensuring that users are assigned the appropriate role and that permissions are strictly enforced.

[F2] Profile Management. The profile management feature allows users to build and maintain detailed profiles, which serve as their digital resumes on the platform. Students can input information about their education, skills, and experiences, while companies can create organizational profiles showcasing their internship programs and job opportunities. Profile data is critical for the recommendation system, as it helps match students to internships that align with their qualifications and interests. Companies will also need access to student profiles to evaluate potential candidates during the recruitment process. Testing this feature will focus on ensuring that users can easily create and update their profiles, with real-time validation to prevent errors.

[F3] Internship Management. This feature allows companies to create, edit, and delete internship listings. Students can search for opportunities based on various filters (location, required skills, etc.) and apply directly through the platform. When an internship is created or updated, the system will automatically notify relevant students who match the internship criteria. This ensures that companies can quickly find suitable candidates and that students do not miss out on potential opportunities. Testing will ensure that internships are properly stored in the database, updates are reflected in real-time, and notifications are sent without delays.

[F4] Recommendation System. The recommendation system is designed to automate the matching process between students and internships. Using specialized algorithms, the system analyzes student profiles, resumes, and preferences to suggest relevant internship opportunities. This feature reduces the manual effort required from both students and companies, improving the efficiency and accuracy of the recruitment process. The recommendation engine will continuously learn from user interactions and feedback, refining its suggestions over time. Testing this feature will involve simulating different user scenarios to verify that the recommendations are accurate and relevant. Additionally, stress testing will be conducted to ensure the recommendation engine performs well under heavy loads.

[F5] Interview and Task Management. This feature allows companies to create tasks, assign them to candidates, and evaluate their performance as part of the interview and selection process. Tasks can range from coding challenges to written assignments, providing companies with deeper insights into candidates' abilities. Students receive notifications when tasks are assigned and can submit their work directly through the platform. Companies can then review submissions, score them, and make final selections based on performance. Testing will ensure that tasks are correctly assigned, deadlines are enforced, and submissions are stored securely. The system will also be tested for scalability, as companies may assign tasks to multiple candidates simultaneously.

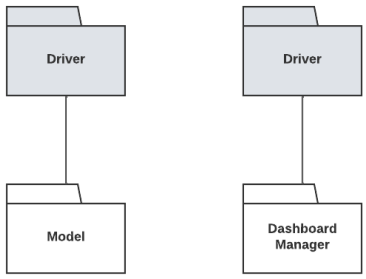
[F6] Feedback and Complaints. After internships are completed, students and companies can provide feedback on their experiences. This feedback loop helps improve the matching process by refining the recommendation algorithm and providing valuable insights to users. Additionally, the platform allows users to file complaints if issues arise during an internship. Complaints are forwarded to university staff, who can mediate and work towards resolving disputes. Testing this feature will involve validating the feedback forms, ensuring data integrity, and verifying that complaints are routed to the appropriate parties without delays.

[F7] Notification System. The notification system keeps users informed about important events, such as new internship postings, application updates, interview invitations, and complaint resolutions. Notifications will appear within the platform and can also be sent via email for critical actions. This feature ensures that users remain engaged and do not miss essential updates. Testing will focus on the timeliness and accuracy of notifications, ensuring that they trigger correctly based on user actions and system events.

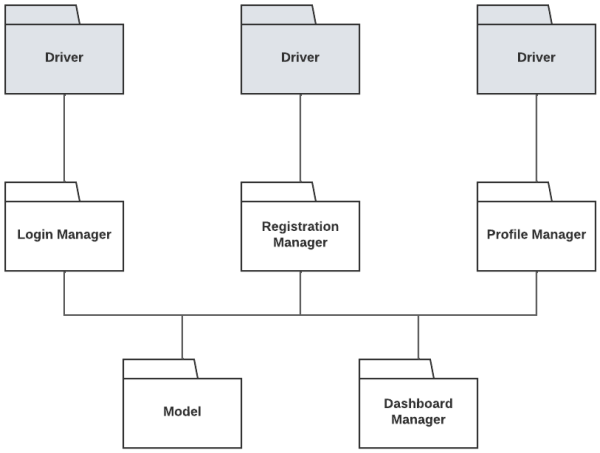
5.3. Integration Strategy

The integration process will follow a progressive bottom-up approach, starting with basic infrastructure and culminating in full system integration. Each integration phase will involve component validation to ensure smooth communication between modules. We assume that WebApp, Web Server and DBMS are already tested and working.

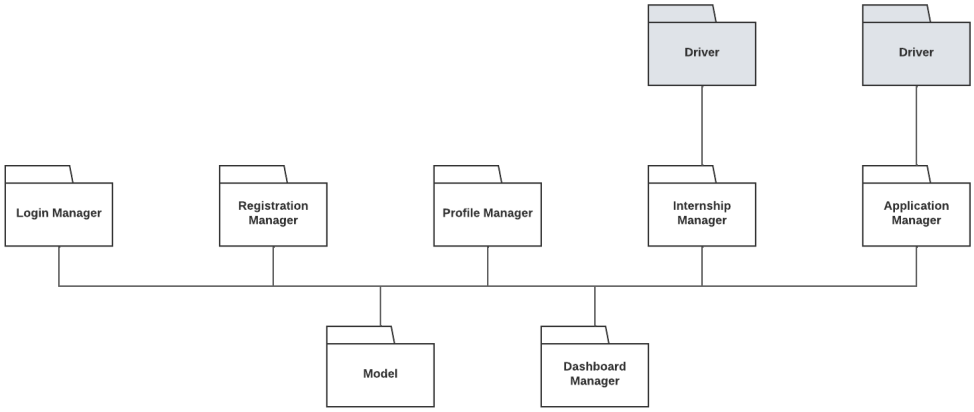
Phase 1 – Core Infrastructure: Integrate the WebApp, Web Server, and DBMS to establish basic user interaction and start testing the Model and the Dashboard Manager with proper drivers.



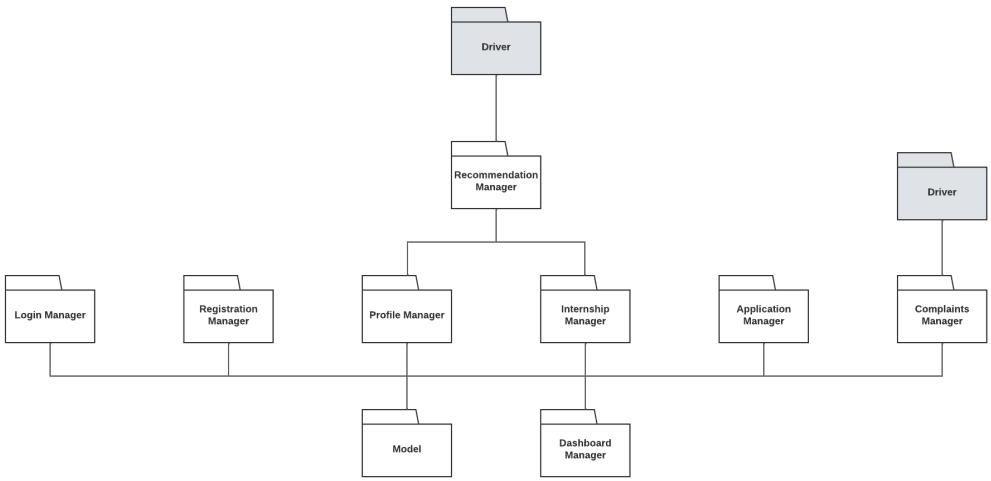
Phase 2 – Profile and User Management: Connect the Login Manager, Registration Manager and Profile Manager to the core system. Test operations on user profiles.



Phase 3 – Internship and Application modules: Incorporate the Internship Manager and Application Manager to enable internship creation and application. Test the end-to-end workflow, from internship posting to candidate application. Test the full interview cycle, from task creation to candidate evaluation.

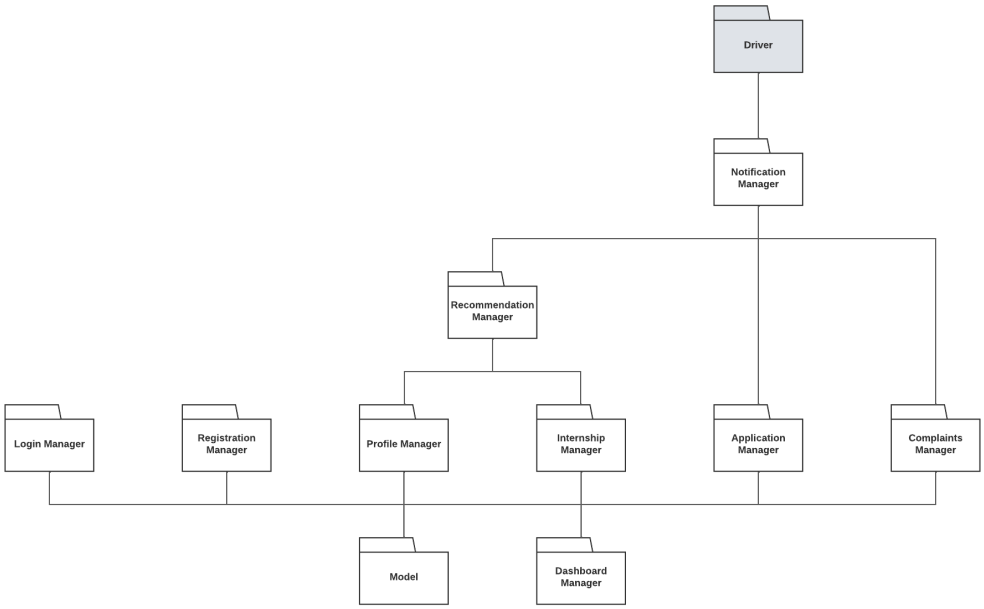


Phase 4 – Recommendation and Complaints Systems: Integrate the Recommendation Manager, linking it to the external tools for algorithm execution. Validate recommendation accuracy and matching efficiency. At the same time, another team can integrate the Complaints Manager, allowing users to submit complaints. Test communication channels between students, companies, and universities

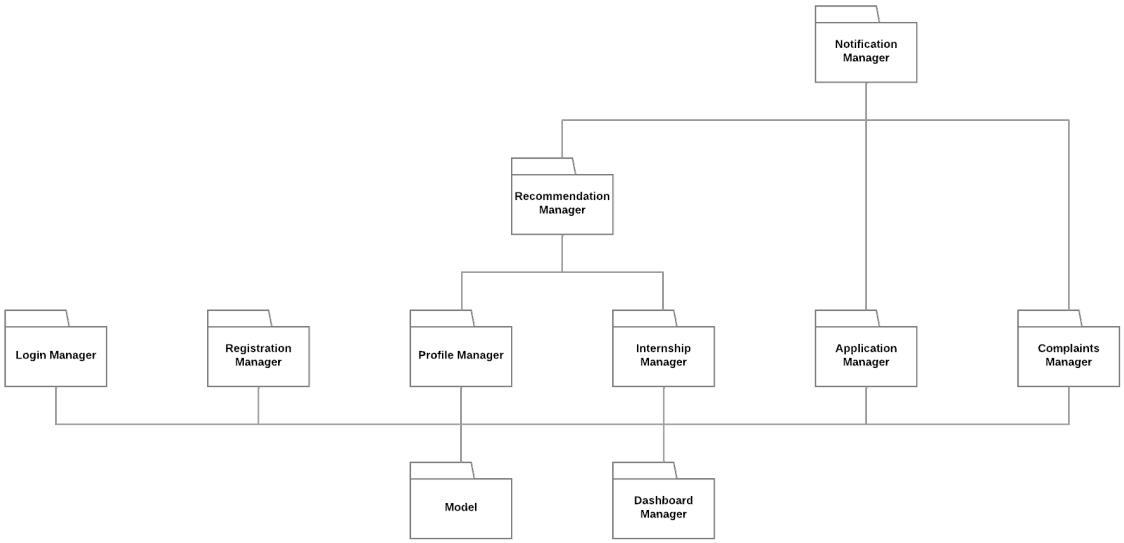


Phase 5 – Notification System: Integrate the Notification Manager, allowing the

system to notify users. Test different types of notification coming from the system.



Phase 6 – Full System Integration: Connect all components into a fully functional system. Perform load and stress tests to assess scalability and performance.



5.4. System Testing Strategy

After testing individual components to verify functionalities in isolation through the use of drivers that simulate the behaviour of missing components (unit testing) they will be progressively integrated with the rest of the system. After integration, a new driver will verify that the module not only operates as intended but also interacts correctly with the existing components. This incremental testing approach helps maintain system stability and ensures that the workflow remains consistent throughout development. After full integration of all components, system tests will be ran to verify the overall workflow, detect bugs, and confirm that the platform adheres to the functional and non-functional requirements outlined in the RASD document. The following types of tests will be applied to validate the platform:

- **Functional Testing:** Ensures that the platform behaves as described in the RASD by simulating real-world workflows. This includes validating user actions like authentication, internship applications, and task submissions, ensuring proper error handling and requirement fulfilment.
- **Load Testing:** Evaluates how the platform performs under increasing user load to identify memory leaks, slowdowns, and resource issues. This test ensures the system can handle high traffic without performance degradation.
- **Performance Testing:** Measures system speed and responsiveness under heavy workloads. Focuses on identifying bottlenecks in processes like internship matching and task management to ensure smooth operation even with multiple users.
- **Stress Testing:** Tests how the system reacts to extreme conditions, such as resource limitations or user surges, to verify its ability to recover and prevent data loss after failures.
- **Usability Testing:** Validates the platform's usability across different devices, browsers, and screen sizes. Ensures that the interface is intuitive, responsive, and accessible to all users, including those with disabilities.
- **Security Testing:** Checks for vulnerabilities in authentication, data protection, and system access. This includes penetration testing and encryption validation to safeguard user information.

6 | Effort Spent

Member of group	Effort spent	
Pianalto Riccardo	Introduction	1.5h
	Architectural Design	20h
	User Interface Design	5h
	Requirements Traceability	1h
	Implementation Integration Test Plan	1h
Pica Mirko	Introduction	1.5h
	Architectural Design	10h
	User Interface Design	3h
	Requirements Traceability	2h
	Implementation Integration Test Plan	10h
Prendin Christian	Introduction	2h
	Architectural Design	10h
	User Interface Design	20h
	Requirements Traceability	0.5h
	Implementation Integration Test Plan	1h

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

7 | References

7.1. References

- ISO/IEC/IEEE29148:2018 - Systems and software engineering - Life cycle processes - Requirements engineering.
- The Requirement Engineering and Design Project specification document A.Y. 2024–2025.

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' design.
- *Lucidchart* and *draw.io* for the other diagrams' design.
- *FlutterFlow* for the User Interface Design.
- *Google Docs* for collaborative writing, notes and reasoning.

List of Figures

List of Tables