POLITECNICO DI MILANO

SOFTWARE ENGINEERING II

S&C Platform

Design Document

Version 1.0

Giovanni Ni 10831328
Xinyue Gu 10840236

Indice

1. Introduction

1.1 Purpose

The main goal of this document is to support the development team in bringing the envisioned system to life. It provides a detailed overview of the chosen system architecture, including an in-depth analysis of individual components. Additionally, it explains the complex interactions between these components, offering a clear understanding of the system's internal dynamics.

The Design Document (DD) also outlines comprehensive plans for implementation, integration, and testing. These plans are thoughtfully developed, prioritizing tasks, estimating the required effort, and evaluating the impact of each component on stakeholders. In summary, this document serves as a guiding framework, aligning the development process with strategic objectives to ensure a cohesive and effective realization of the system.

1.2 Scope

The scope of the Students & Companies (S&C) platform focuses on facilitating efficient and effective matching between university students seeking internships and companies offering them. It enables companies to provide detailed information about internships, including project specifics and terms, while allowing students to upload and manage CVs highlighting their skills, experiences, and attitudes. The platform supports the entire selection process by enabling students to manually search and apply for internships, providing tools for companies to review applications, set up interviews, and finalize selections. Additionally, it monitors the progress of internships by collecting updates and logging any complaints reported during their execution, although resolving these complaints falls outside the platform's responsibilities. To improve the overall experience, the system gathers feedback and statistical data from students and companies regarding internships and platform functionality. Furthermore, it incorporates recommendation mechanisms, from keyword-based searches to sophisticated statistical analyses, to suggest relevant matches and notify users about suitable opportunities. This comprehensive approach ensures a streamlined process for managing internships and enhances the platform's ability to connect students with appropriate opportunities effectively.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Student**: An individual engaged in learning or academic activities, typically enrolled in an educational institution or a course. In a model, a Student might have attributes like name, email, titles.

- **Company**: A company is a legal entity formed by a group of individuals to engage in and operate a business enterprise, with the purpose of generating profit, providing goods or services, or pursuing other specific objectives. In a model, a company might have attributes like name, contact email, internship positions.

- **Internship**: An internship is a structured, often temporary work experience offered by an organization to individuals, typically students or recent graduates, to provide practical exposure to a specific field or industry.

1.3.2 Acronyms

- **[S&C]**: Students&Companies Platform
- **[UI]**: User Interface
- **[UML]**: Unified Modelling Language
- **[RASD]**: Requirement Analysis and Specification Document
- **[API]**: Application Programming Interface
- **[DD]**: Design Document
- **[DB]**: Database

1.4 Revision History

- Version1.0 (January 05th 2025 : first version);

1.5 Reference Documents

- The specification of the RASD and DD assignment of the Software Engineering II course, held by Professor Matteo Rossi, Elisabetta Di Nitto and Matteo Camilli at Politecnico di Milano, A.Y 2024/2025
- Slides of Software Engineering 2 course on WeBeep platform.

1.6 Document Structure

1. Introduction:
This section discusses the importance of the Design Document, provides definitions and explanations for acronyms and abbreviations, and outlines the scope of the Student & Company system.

2. Architectural Design:
Describes the system's main components, their relationships, and the design decisions, including architectural styles, patterns, and paradigms.

3. User Interface Design:
Explains the user interface of the system, with mockups and detailed descriptions of the primary pages.

4Requirements Traceability:
Outlines the system requirements and explains how the design choices satisfy these requirements.

5.Implementation, Integration, and Test Plan
Provides an overview of the system's implementation, the integration of components, and the testing plan.

6.Effort Spent: Summarizes the hours contributed by each team member in preparing the document.

7.References: Lists the documents and resources used in creating the Design Document.

2. Architectural Design

2.1 Overview

In this section we are going to describe the architecture of our system, starting from a high-level view going to detailed functions, justifying all the choice for adopted patterns.

2.1.1 High-level components and tier interaction

The S&C system is organized into a three-tier architecture, which ensures modularity and efficient operation. The tiers are defined as follows:

1. **Presentation Tier**:
Also referred to as the user interface layer, this tier is primarily designed to interact with users. It displays information in a clear and understandable format and manages user input. By focusing on providing a responsive and intuitive interface, it facilitates seamless interaction with the system. This tier encompasses all components users directly interact with to perform actions and view results.

2. **Application Tier**:
This tier houses the system's core logic and processes. It is responsible for handling user requests, executing application functionality, and serving as the intermediary between the user interface and the data storage layer. This is where the main processing and decision-making occur, ensuring effective data handling and system operations.

3.**Data Tier**:
Dedicated to managing and storing data, the data tier is responsible for organizing, retrieving, and manipulating information. It utilizes database systems to maintain data integrity, security, and availability. This tier responds to the application tier's requests, ensuring accurate and efficient data handling.

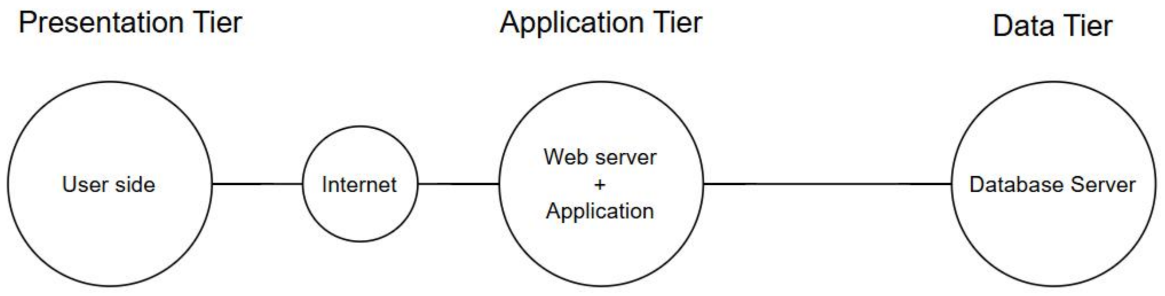


Figure 2.1 - S&C Three tier architecture

The clear separation of these tiers improves modularity, maintainability, and scalability of the system. Each layer communicates exclusively with its adjacent tier, requiring the application tier to mediate interactions between the presentation tier and the data tier.

The system is a distributed application based on the widely used client-server paradigm. The S&C platform is designed to be accessed through a web browser by registered users (Students or Companies). Users initiate HTTPS requests to a Web Server, which processes the request and returns the appropriate web page via the Application Server.
Furthermore, the application server plays a key role by retrieving the required data from:

-**Shared Databases**: these store general information such as user profiles, internship position information.

-**Private Databases**: these contain sensitive or specific information, including user account passwords, interview results.

This architecture ensures secure and efficient data handling while supporting the platform's functionality for a seamless user experience.
At server side, we have also an email provider who is responsible for sending registration confirmation emails.

To enhance security, a **front-end firewall** has been installed on the router connecting the internal S&C network to the internet. For even greater protection, **back-end firewalls** can also be implemented between the presentation layer and the application layer, as well as between the application layer and the data layer.

Additionally, the system uses **Load Balancing** to evenly distribute incoming requests across multiple servers, improving performance and avoiding overloading any single server. Specifically, to route requests to the **Web Server** and **Application Server**, all requests from **web browsers** are received by the load balancer, which directs them to the Web Server for initial handling. The **Web Server** processes the request and then forwards it to the Application Server for further processing.

Finally, **Clustered Database Management Systems** (DBMS) are utilized to enhance scalability, fault tolerance, and data availability by distributing and replicating data across multiple database nodes within the cluster.

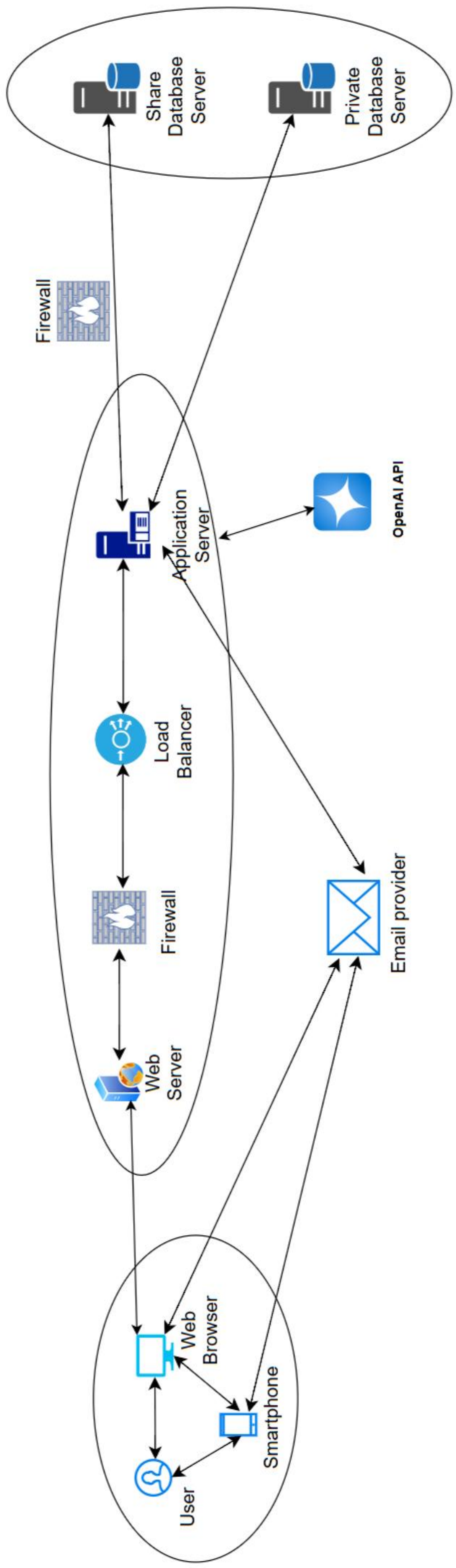


Figure 2.2 - S&C Distributed view

2.2 Component View

In the diagram below, the components of the S&C platform are categorized by their respective tiers:

- **Presentation Tier**: Represented by the WebApp, this tier focuses on the user interface, providing a seamless and interactive web experience for users.
- **Data Tier**: Represented by the Database Management System (DBMS), this tier handles the management of shared and private data, offering structured storage solutions to support the platform's functionalities.
- **Application Layer**: Highlighted in purple, this layer forms the core of the S&C platform, incorporating key business logic and management functionalities.
- **External Services**: The email provider and OpenAI provider are depicted in a different color, as it represents third-party services integrated into the platform.

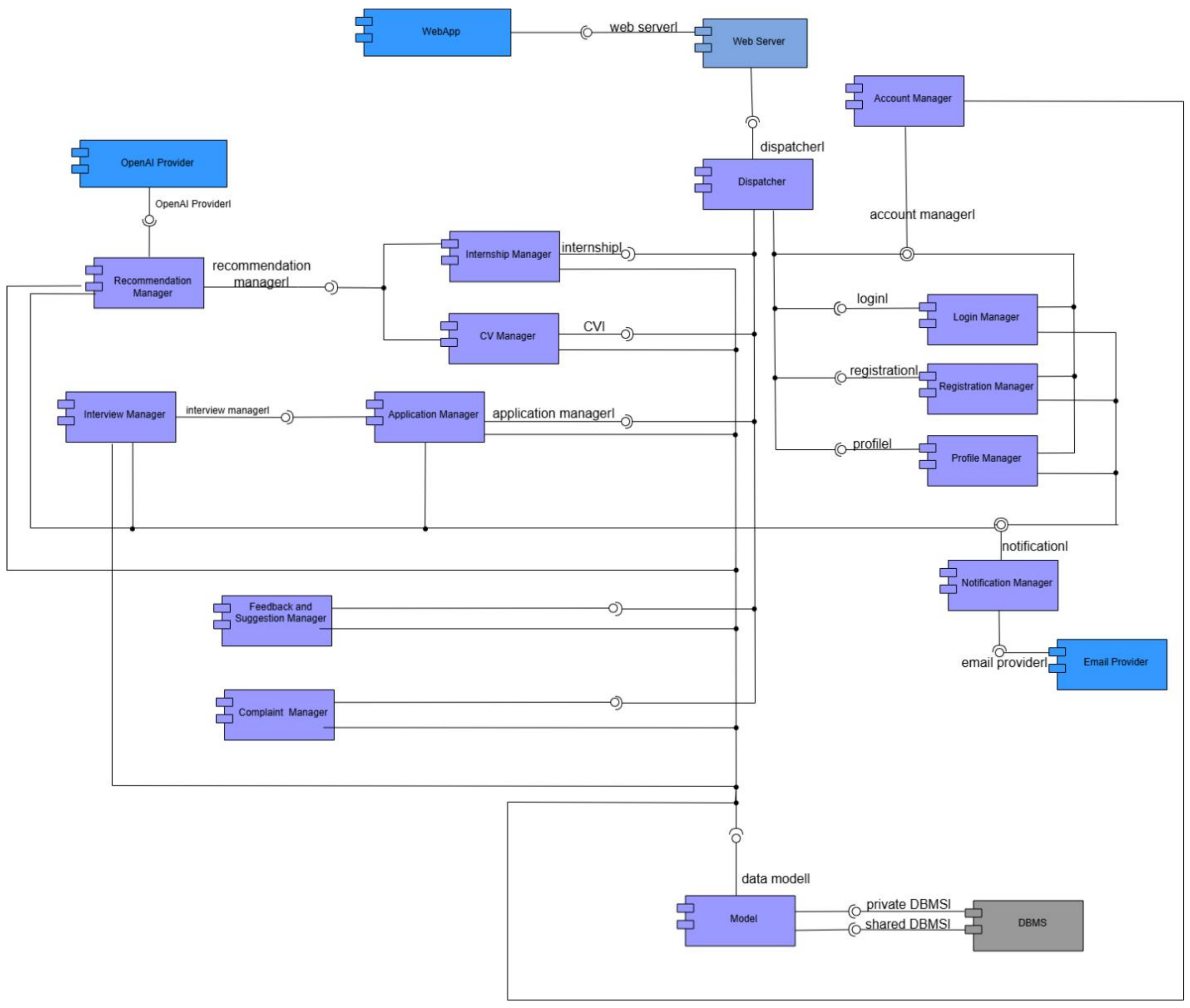


Figure 2.3 - S&C Component view

Each component plays a critical role in ensuring the platform's functionality and efficiency.

- The **WebApp** functions as the interface for students and companies to engage with the S&C platform. It facilitates communication with the system via HTTP requests directed to the Webserver.

- **Webserver** acts as the interface through which users (students and companies) interact with the various managers and functionalities of the S&C platform.

- The **Dispatcher** oversees communication between different managers, ensuring a smooth workflow and efficient data exchange across the system.

- **Internships Manager**: This component manages the available internship positions from companies. It interacts with the Application Manager to handle any applications submitted for internship positions.

- **Interview Manager**: This component manages the organization of interviews when a company accepts a student's application for an internship position. It coordinates the scheduling and logistics to facilitate the interview process.

- **CV Manager**: This component works in collaboration with the Student Profile Manager. It is responsible for uploading and managing the student's CV, ensuring it is properly stored and accessible for relevant processes.

- **Recommendation manager**: This component generates personalized recommendations for both students and companies. It may include algorithms designed to identify the most suitable matches between students and internship opportunities, enhancing the pairing process.

- **Model**: This high-level component represents the data on the server and serves as an interface to the database server. It acts as an "abstraction layer", masking the database server and ensuring that all other components interact with it to access data from the DBMS.

- **Applications Manager**: this component handles applications from students for an internship position through Internship Manager.

- **Complaint Manager**: This component manages the ongoing internships for both students and companies, overseeing the progress and status of active internship placements. It ensures that both parties are supported throughout the internship duration.

- **Feedback and Suggestion Manager**: This component collects ratings, feedback, and suggestions from students and companies. It assists the Recommendation Manager by providing valuable insights that help refine and improve the matching process for better and more suitable results.

- **Account Manager**: This component manages user accounts, including creating and updating personal information, such as updating CV details for students. It interacts with the private account database to store and retrieve user data securely.

- **Registration Manager**: This component is responsible for managing the student registration process on the S&C platform. It provides a specific registration form based on the user type, which must be filled with the appropriate information. For instance, an student would need to type the university name as part of their registration.

- **Login Manager**: This component handles the login process for registered users, ensuring secure access to the platform by verifying credentials and managing authentication.

- **Profile Manager**: This component manages user profiles, displaying personal details and relevant information. It utilizes shared profile data to ensure consistency and accuracy across the platform.

- **Notification Manager**: This component manages the notifications that need to be sent to users, such as when companies receive new applications. It ensures that users are promptly alerted about relevant events or updates on the platform. It uses email providers to send notifications to students and companies, tailoring the messages based on their individual preferences.

- **DBMS** (Database Management System):

  - The SharedDBMS manages shared data, such as profiles, internship positions. It utilizes a clustered database system to ensure scalability, reliability, and efficient data handling across multiple nodes.
  - The PrivateDBMS stores sensitive information such as user passwords.

- **Email Provider**: This component is responsible for sending emails and communicates with the S&C Server through the mail API interface to manage email notifications and communications.

- **OpenAI provider**: The recommendation system for students and companies may integrate external services like OpenAI. This allows the system to suggest the most suitable internship opportunities to students based on their research history and match companies with the ideal candidates for specific roles.

2.3 Deployment View

In this section, the Deployment Diagram of the S&C system will be presented, followed by a detailed description of the components and their interactions. The deployment diagram will illustrate how the system's components are distributed across various servers and how they communicate with each other to ensure efficient operation of the platform. Each component will be explained in terms of its role, how it interacts with other components, and how the overall system architecture supports scalability, security, and performance.

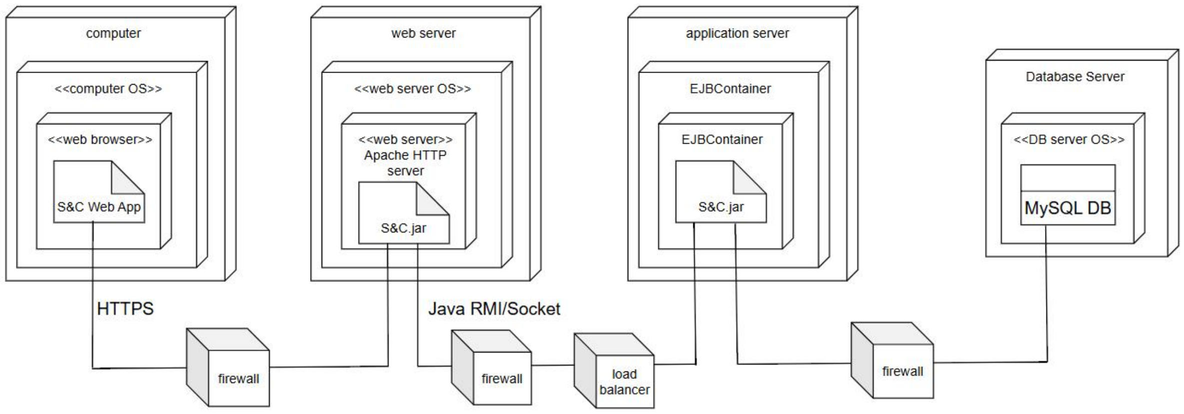


Figure 2.4 - S&C deployment diagram

- **Computer**: Students and companies can access the system using any type of computer through their preferred web browser. The browser communicates with the Web Server, acting as the interface for both students and companies to navigate the platform, ensuring a seamless and user-friendly experience. Users can also access the system from any device with web browsing capabilities, such as mobile phones, tablets, and other devices, allowing for flexible and convenient access to the platform.

- **Firewall**: is a critical security component in the S&C platform, designed to protect against unauthorized access and cyber threats. It enforces rules and policies to control network traffic, defending the platform from web vulnerabilities such as Cross-Site Scripting (XSS) and SQL injection attacks. The Firewall works alongside the Web Server to secure data transmission between users' devices (such as computers) and the S&C platform, using protocols like HTTPS to ensure encrypted communication. By defining and enforcing access rules, the Firewall blocks malicious traffic while allowing legitimate requests, thereby safeguarding the platform's integrity and preventing potential security breaches.

- **Web Server**: The Web Server acts as an intermediary, providing users with access to the services of the Application Server when they reach the system through a web browser. While the Web Server does not handle business logic, it performs load balancing by distributing incoming requests from clients across multiple Application Servers to manage high user traffic effectively. After receiving a request, the Application Server processes the necessary business logic and returns the results to the Web Server. The Web Server then generates dynamic web pages, incorporating the relevant data (such as updated scores) and delivers these pages to the users' browsers, ensuring they see the most up-to-date information. Anyway, sometimes for static content requests, the Web Server efficiently handles these locally, ensuring a responsive user experience without involving the Application Server. Additionally, it serves HTML, JSON, JavaScript, and CSS files to the client's browser, ensuring proper page rendering and a smooth user experience.

- **Load Balancer**: a crucial component in the S&C platform's infrastructure, playing a key role in optimizing performance, ensuring fault tolerance, and effectively managing incoming web traffic. Its primary function is to evenly distribute user requests across multiple instances of the Web Server. By monitoring the health of individual Web Server instances, the Load Balancer intelligently routes traffic only to healthy servers, preventing service disruptions and improving overall system reliability.In situations of increased demand or potential server failures, the Load Balancer dynamically adjusts the distribution of incoming requests, optimizing resource utilization and maintaining a smooth user experience. This ensures that no single server becomes overwhelmed, contributing to improved scalability and responsiveness. By spreading the load across multiple servers, it ensures high availability and scalability, allowing the S&C platform to handle many concurrent users without performance degradation.

- **Application Server**: contains the business logic of the entire system, processing the core functions and handling requests. It communicates with the client through the HTTPS protocol, managed by the Web Server. Additionally, the Application Server communicates with the Database Server through the model gateway, enabling seamless access to the data stored in the database and ensuring that the system can retrieve and manipulate data as needed.

- **Database Server**: serves as a central repository within the S&C platform, responsible for storing and managing persistent data that is critical to the platform's functionality. It holds and organizes essential data such as user profiles, internship details, interviews, and other important information. The Application Server actively interacts with the Database Server, storing and retrieving data as needed during various operations.By maintaining a structured and organized database, the platform ensures the persistence and reliability of its data. The Database Server enhances the robustness of the S&C platform by providing a secure, efficient, and scalable storage solution for the wide range of data generated and managed within the system.

2.4 Runtime View

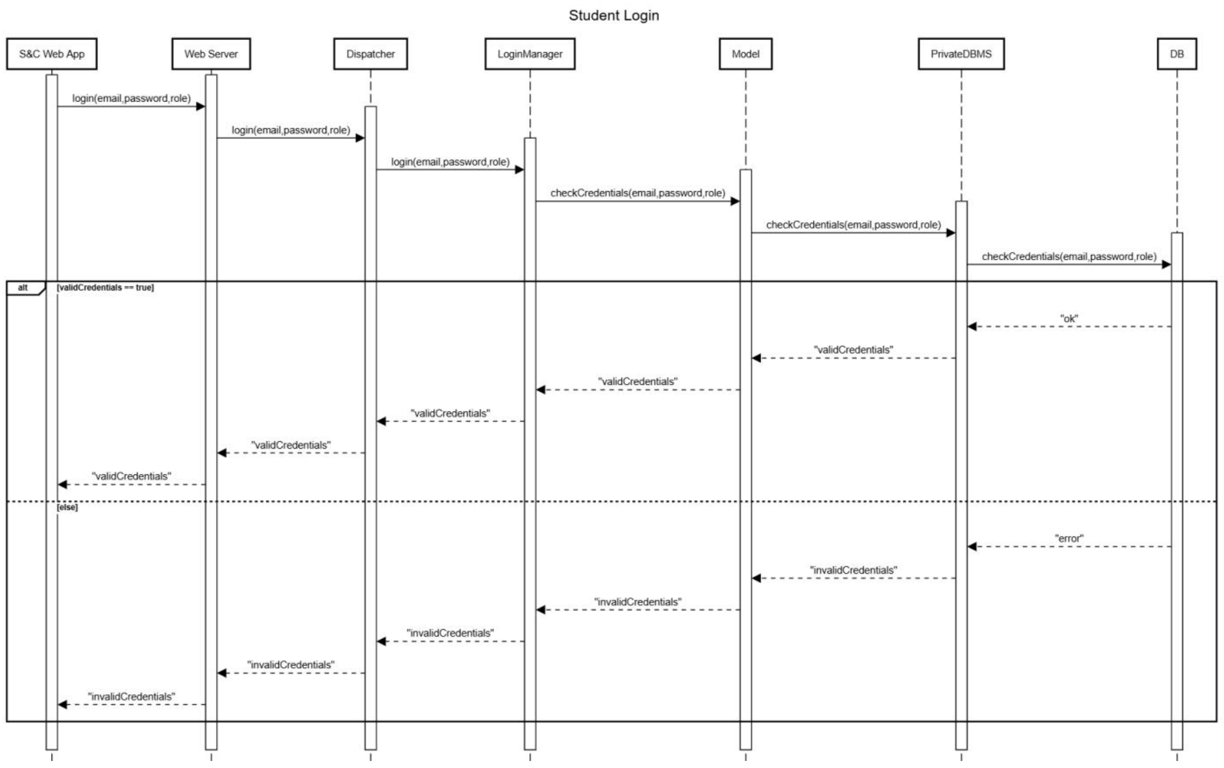


Figure 2.5 - Student Login Runtime View

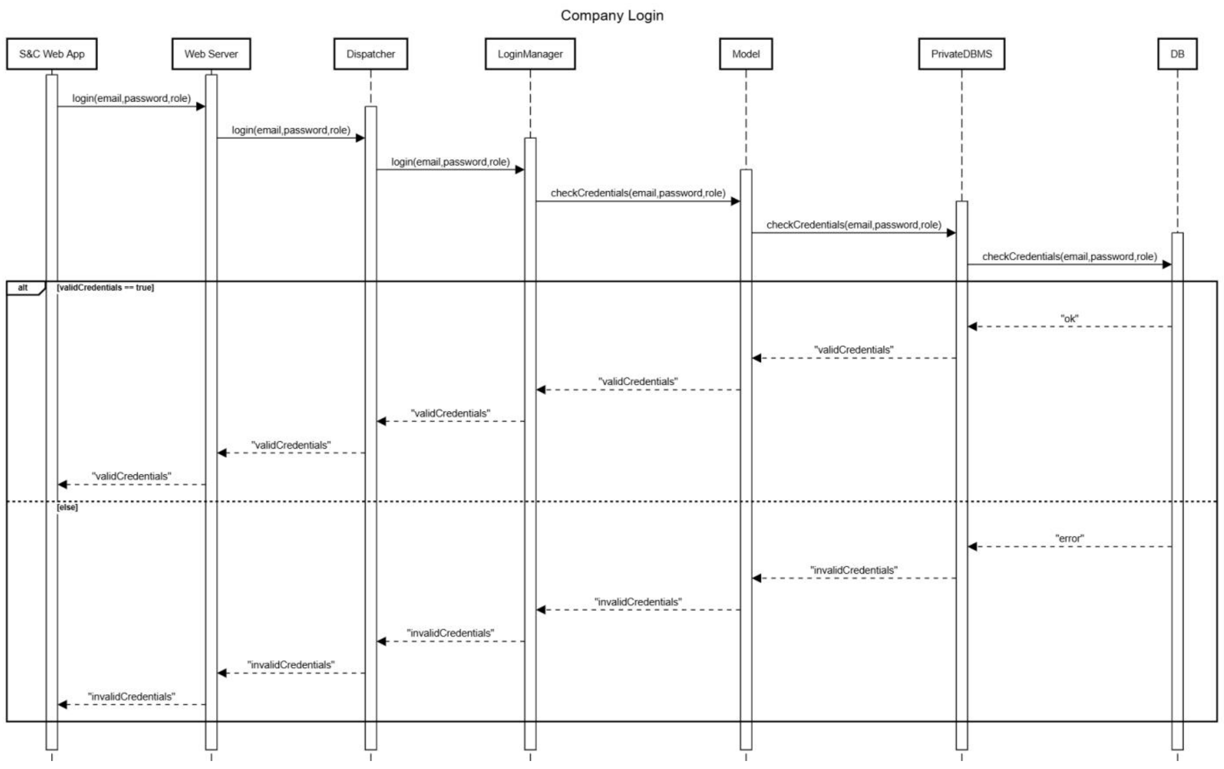


Figure 2.6 - Company Login Runtime View

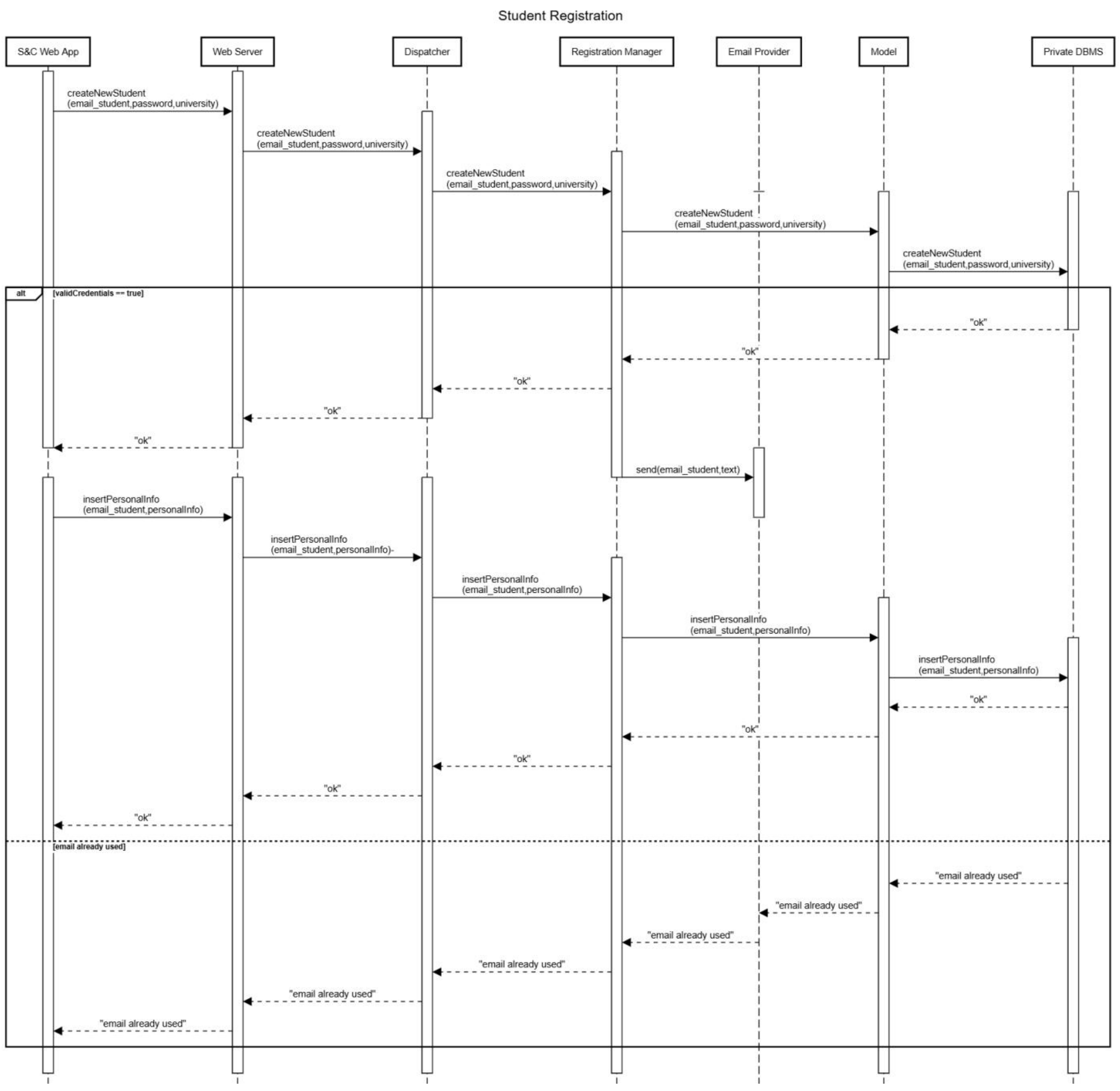

Student Registration
S&C Web App
Web Server
Dispatcher
Registration Manager
Email Provider
Model
Private DBMS
createNewStudent (email_student,password,university)
createNewStudent (email_student,password,university)
createNewStudent (email_student,password,university)
createNewStudent (email_student,password,university)
createNewStudent (email_student,password,university)
alt
[validCredentials == true]
"ok"
"ok"
"ok"
"ok"
"ok"
send(email_student,text)
insertPersonalInfo (email_student,personalInfo)
insertPersonalInfo (email_student,personalInfo)-
insertPersonalInfo (email_student,personalInfo)
insertPersonalInfo (email_student,personalInfo)
insertPersonalInfo (email_student,personalInfo)
"ok"
"ok"
"ok"
"ok"
"ok"
[email already used]
"email already used"
"email already used"
"email already used"
"email already used"
"email already used"
"email already used"

Figure 2.7 - Student Registration Runtime View

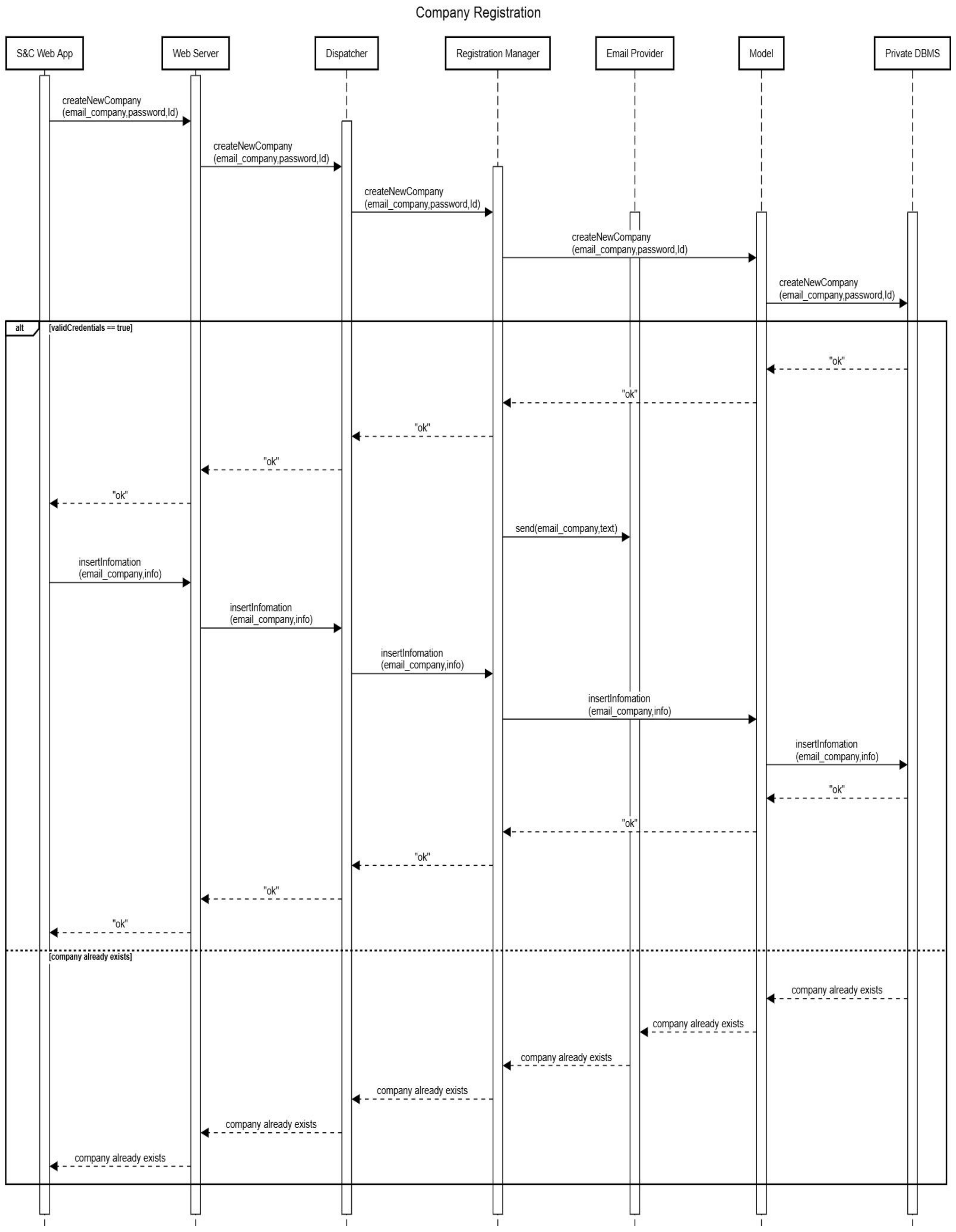


Figure 2.8 - Company Registration Runtime View

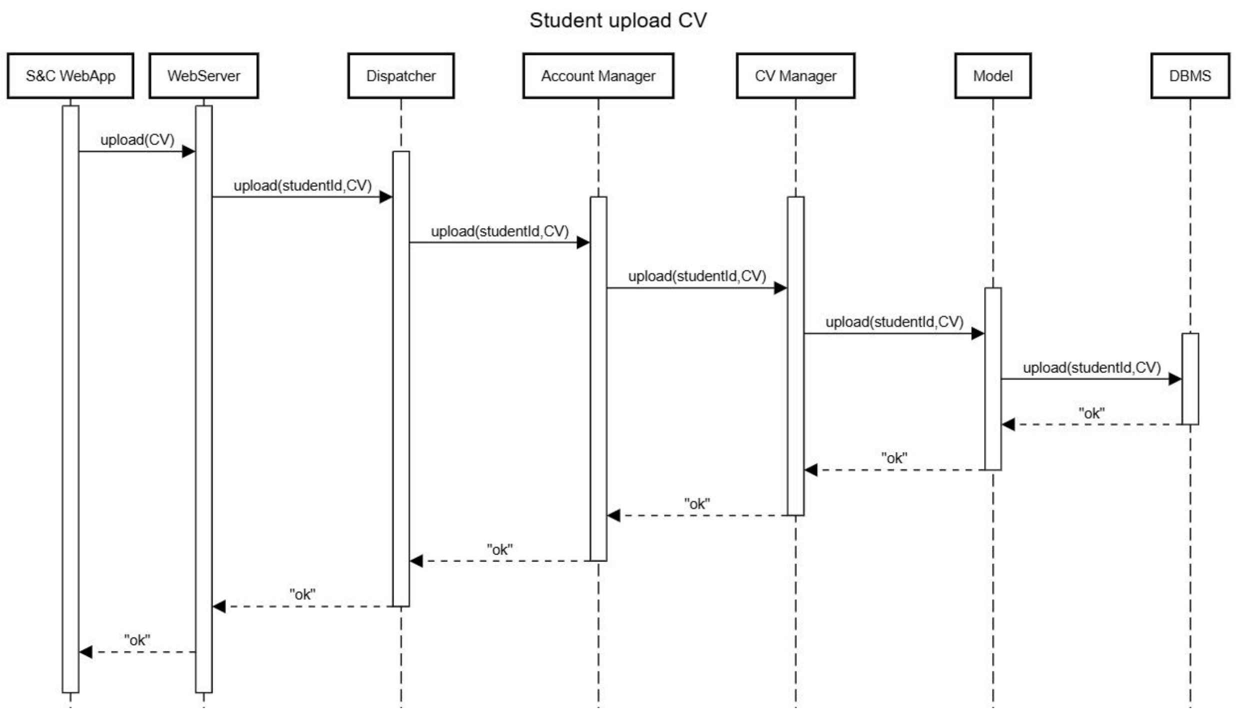


Figure 2.9 - Student uploads CV Runtime View

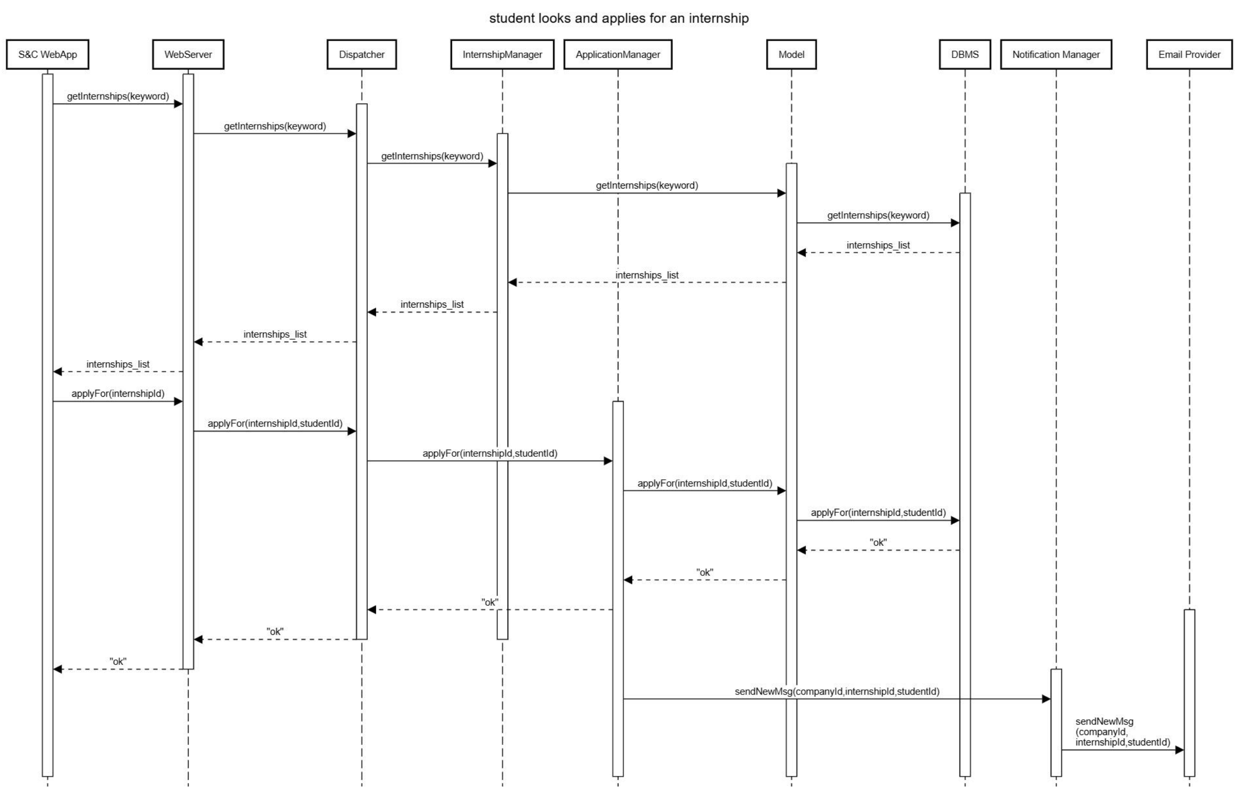


Figure 2.10 - Student looks and applies for an internship Runtime View

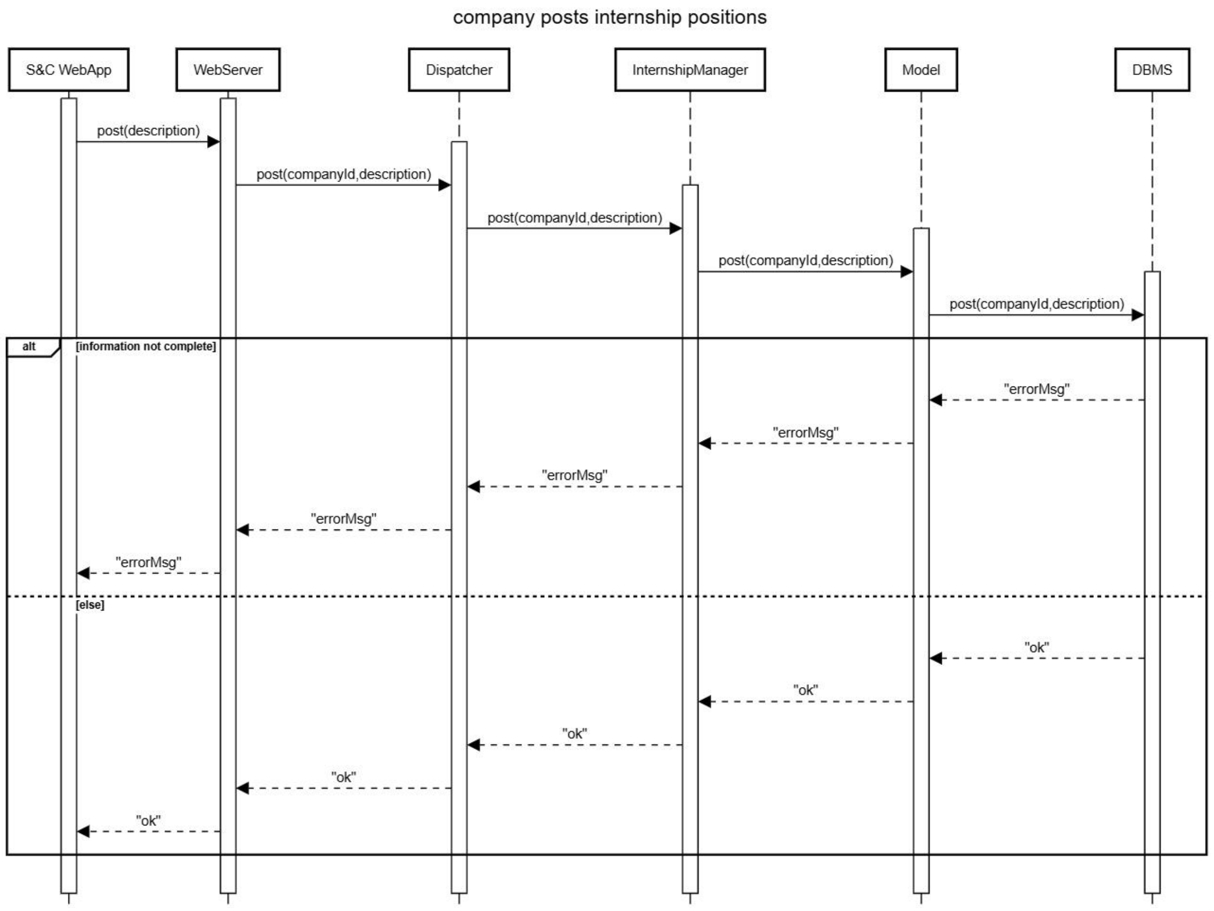

company posts internship positions
S&C WebApp
WebServer
Dispatcher
InternshipManager
Model
DBMS
post(description)
post(companyId,description)
post(companyId,description)
post(companyId,description)
post(companyId,description)
alt
[information not complete]
"errorMsg"
"errorMsg"
"errorMsg"
"errorMsg"
"errorMsg"
[else]
"ok"
"ok"
"ok"
"ok"
"ok"

Figure 2.11 - Company posts internship positions Runtime View

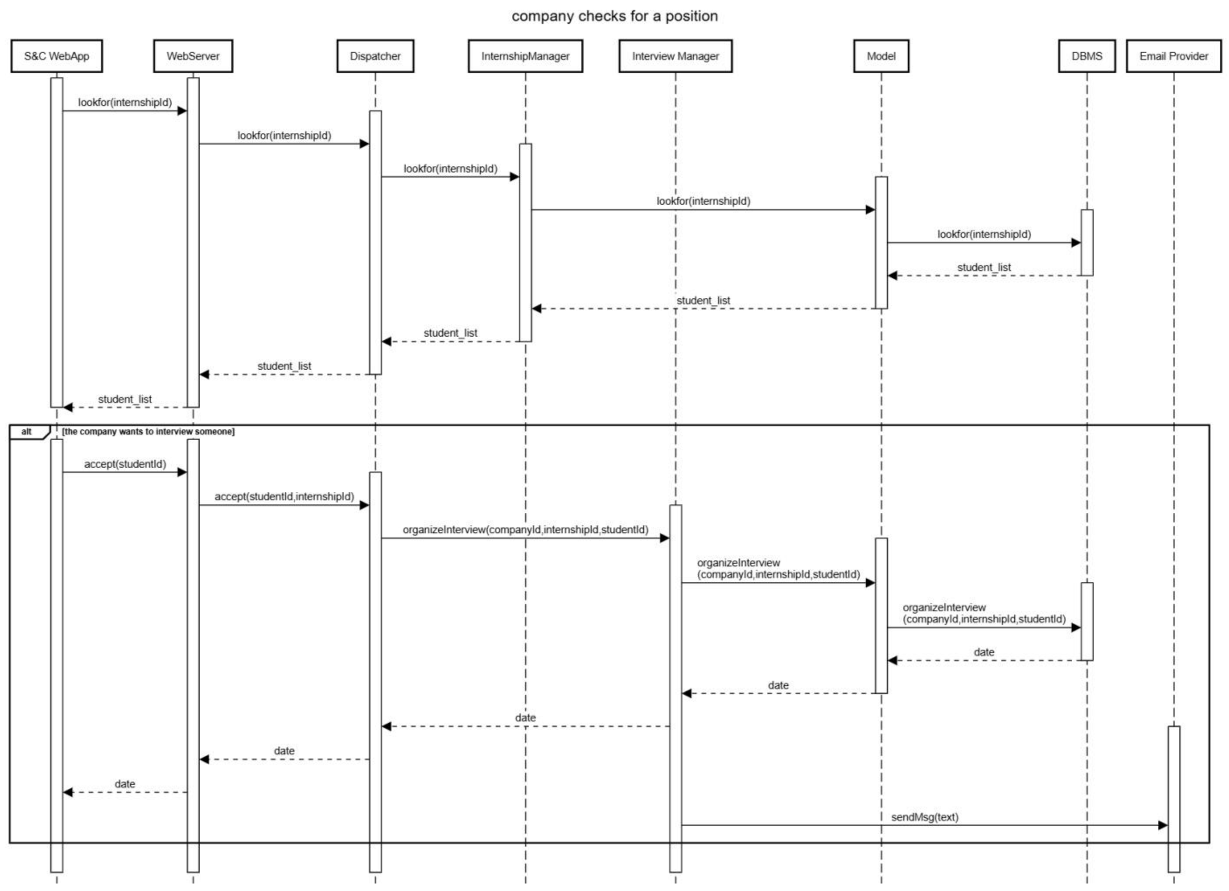


Figure 2.12 - Company checks for a position Runtime View

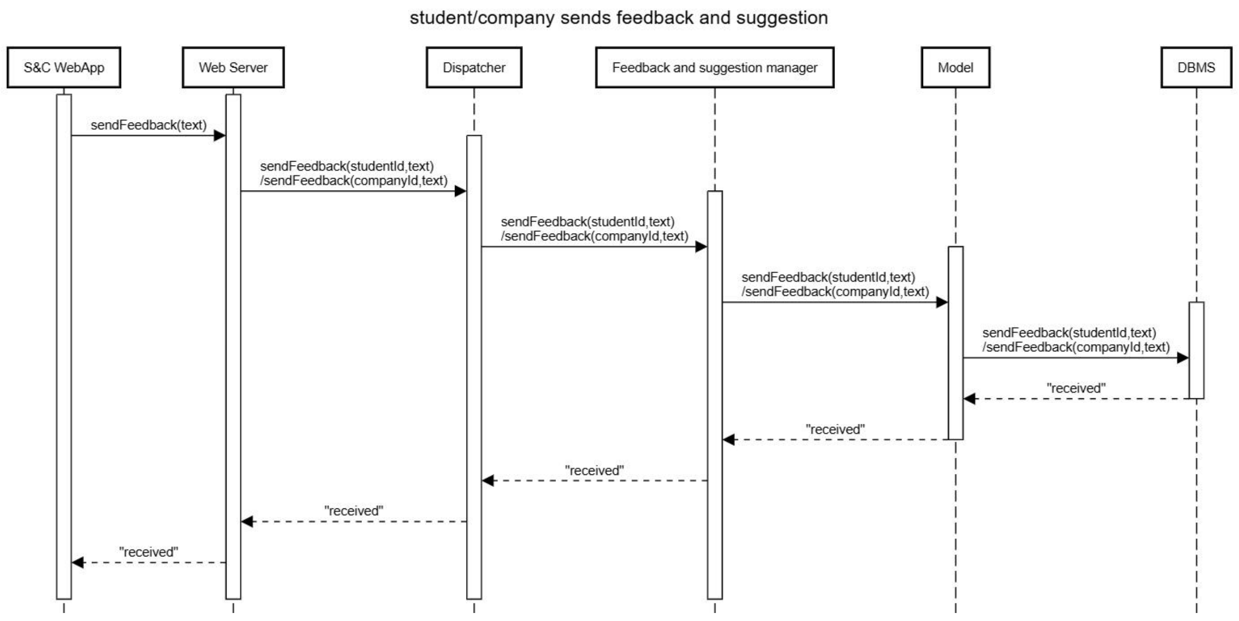

student/company sends feedback and suggestion
S&C WebApp
Web Server
Dispatcher
Feedback and suggestion manager
Model
DBMS
sendFeedback(text)
sendFeedback(studentId,text)
/sendFeedback(companyId,text)
sendFeedback(studentId,text)
/sendFeedback(companyId,text)
sendFeedback(studentId,text)
/sendFeedback(companyId,text)
sendFeedback(studentId,text)
/sendFeedback(companyId,text)
"received"
"received"
"received"
"received"
"received"

Figure 2.13 - Student/Company send feedback and suggestion Runtime View

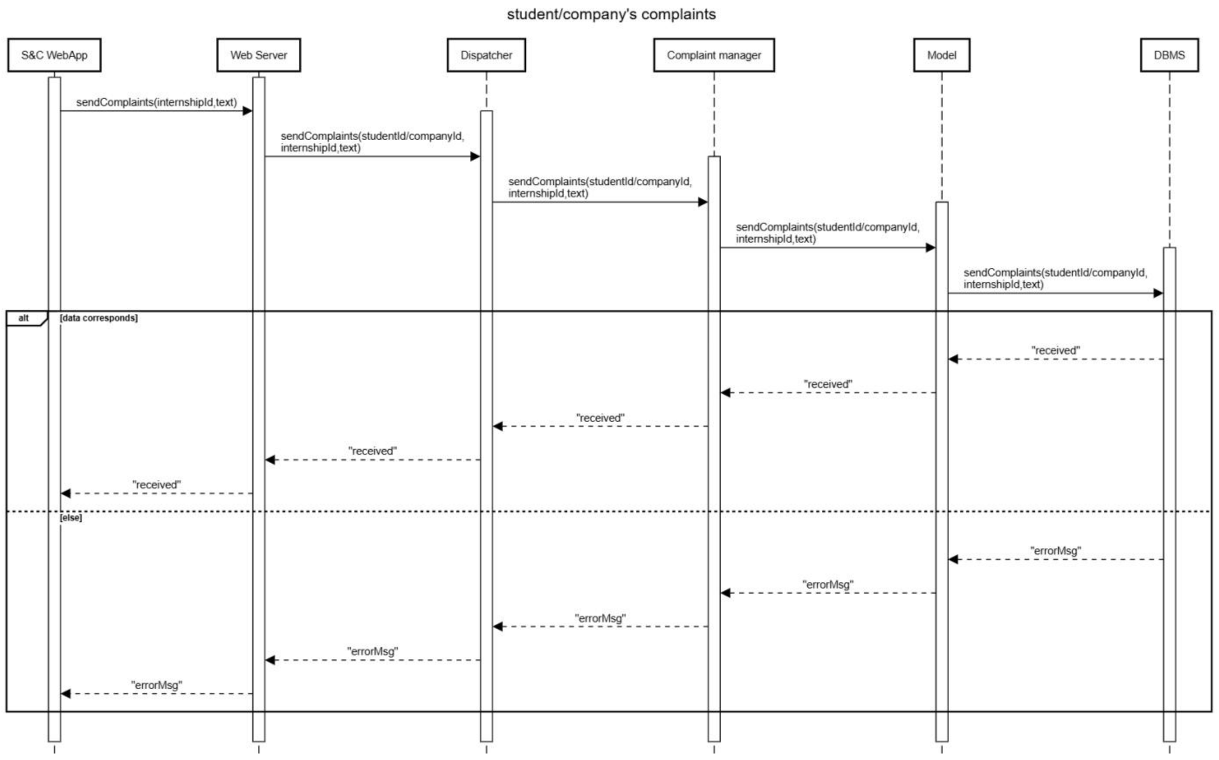

student/company's complaints
S&C WebApp
Web Server
Dispatcher
Complaint manager
Model
DBMS
sendComplaints(internshipId,text)
sendComplaints(studentId/companyId, internshipId,text)
sendComplaints(studentId/companyId, internshipId,text)
sendComplaints(studentId/companyId, internshipId,text)
sendComplaints(studentId/companyId, internshipId,text)
alt
[data corresponds]
"received"
"received"
"received"
"received"
"received"
[else]
"errorMsg"
"errorMsg"
"errorMsg"
"errorMsg"
"errorMsg"

Figure 2.14 - Student/Company's Complaints Runtime View

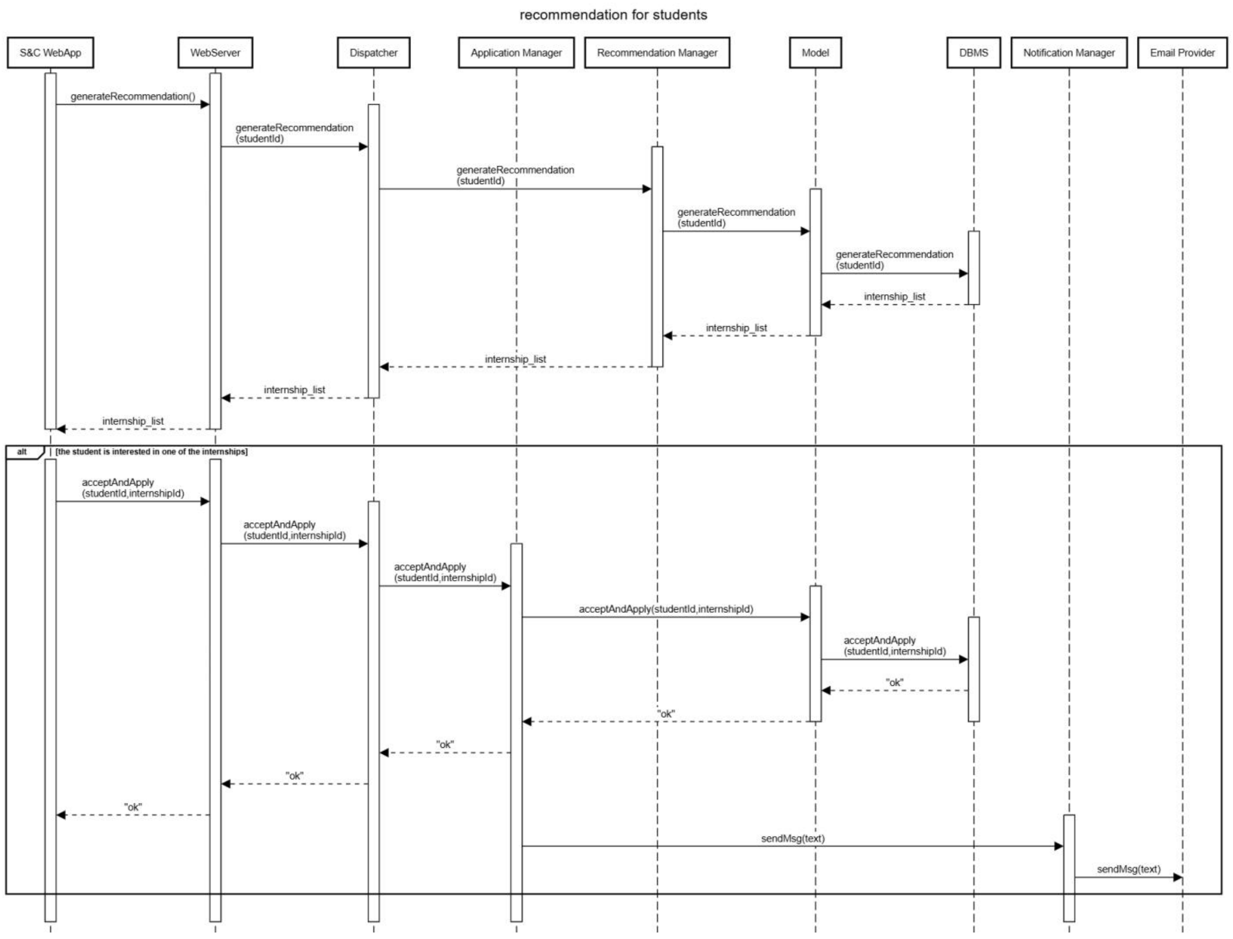

recommendation for students
S&C WebApp
WebServer
Dispatcher
Application Manager
Recommendation Manager
Model
DBMS
Notification Manager
Email Provider
generateRecommendation()
generateRecommendation (studentId)
generateRecommendation (studentId)
generateRecommendation (studentId)
generateRecommendation (studentId)
internship_list
internship_list
internship_list
internship_list
internship_list
alt
[the student is interested in one of the internships]
acceptAndApply (studentId,internshipId)
acceptAndApply (studentId,internshipId)
acceptAndApply (studentId,internshipId)
acceptAndApply(studentId,internshipId)
acceptAndApply (studentId,internshipId)
"ok"
"ok"
"ok"
"ok"
"ok"
sendMsg(text)
sendMsg(text)

Figure 2.15 - Recommendation for students Runtime View

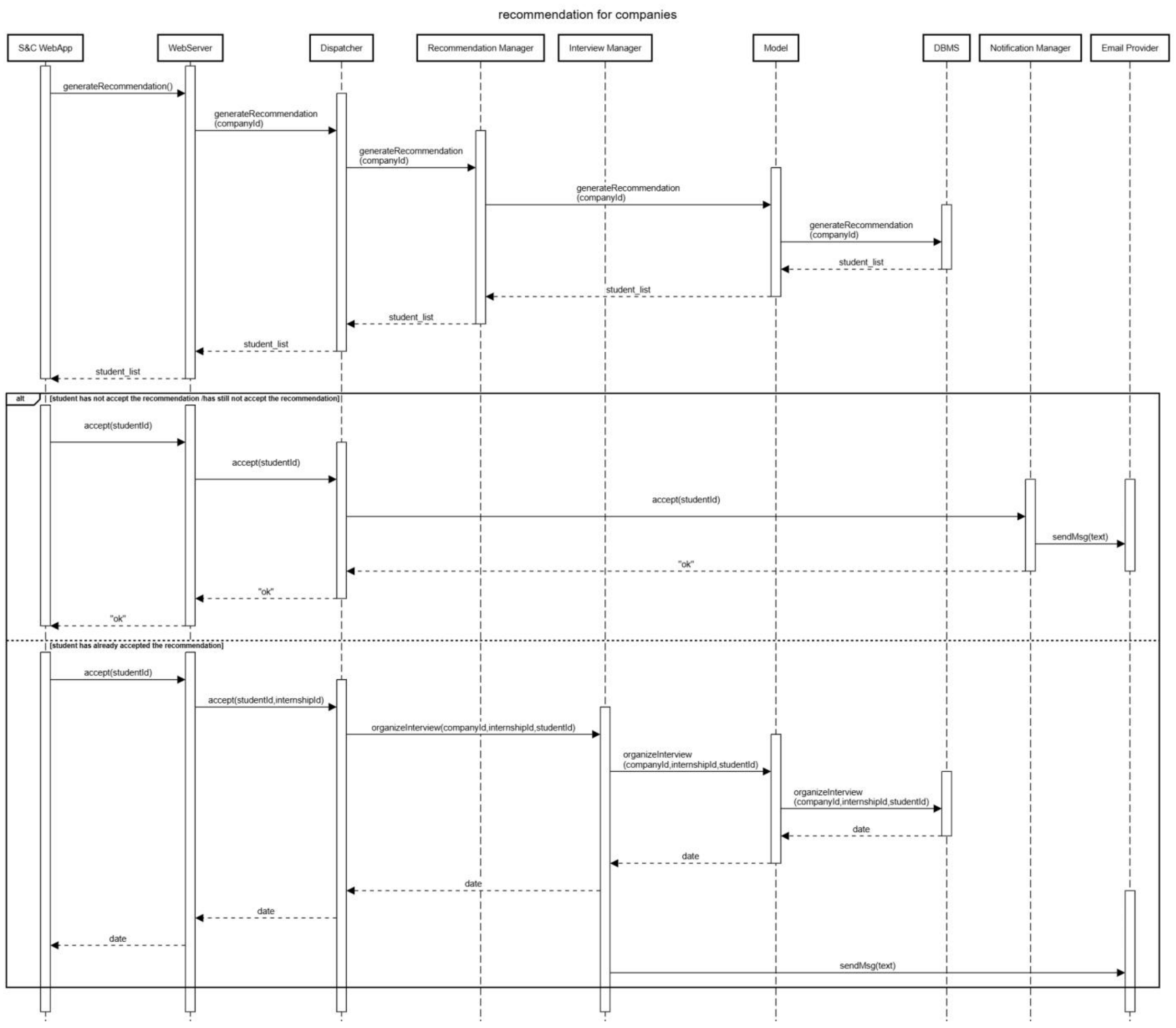

recommendation for companies
S&C WebApp
WebServer
Dispatcher
Recommendation Manager
Interview Manager
Model
DBMS
Notification Manager
Email Provider
generateRecommendation()
generateRecommendation (companyId)
generateRecommendation (companyId)
generateRecommendation (companyId)
generateRecommendation (companyId)
student_list
student_list
student_list
student_list
student_list
alt
[student has not accept the recommendation /has still not accept the recommendation]
accept(studentId)
accept(studentId)
accept(studentId)
sendMsg(text)
"ok"
"ok"
"ok"
[student has already accepted the recommendation]
accept(studentId)
accept(studentId,internshipId)
organizeInterview(companyId,internshipId,studentId)
organizeInterview (companyId,internshipId,studentId)
organizeInterview (companyId,internshipId,studentId)
date
date
date
date
date
sendMsg(text)

Figure 2.16 - Recommendation for companies Runtime View

2.5 Component Interfaces

Web ServerI

-login(email, password, role)
-createNewStudent(email_student, password, university)
-createNewCompany(email_company, password, company_id)
-insertPersonalInfo(id, personalInfo)
-updatePersonalInfo(id, peronalInfo)
-uploadStudentCV(student_id, CV)
-getInterships(keyword)
-applyFor(student_id,intershipId, documents..)
-sendFeedback(id, text)
-sendComplaint(id,intershipId,text)
-postInternship(company_id, internship)
-getSubmitedApplication(intership_id)
-organizeInterview(internship_id, student_id)
-generateRecommendation(id)
-acceptRecommendationAndApply(student_id, intership_id, documents..)

DispatcherI

-login(email, password, role)
-createNewStudent(email_student, password, university)
-createNewCompany(email_company, password, company_id)
-insertPersonalInfo(id, personalInfo)
-updatePersonalInfo(id, peronalInfo)
-uploadStudentCV(student_id, CV)
-getInterships(keyword)
-applyFor(student_id,intershipId, documents..)
-sendFeedback(id, text)
-sendComplaint(id,intershipId,text)
-postInternship(company_id, internship)
-getSubmitedApplication(intership_id)
-organizeInterview(internship_id, student_id)
-generateRecommendation(id)
-acceptRecommendationAndApply(student_id, intership_id, documents..)

Internship ManagerI

-getInterships(keyword)
-postInternship(company_id, internship)
-getSubmitedApplication(intership_id)

Interview ManagerI

-organizeInterview(internship_id, student_id, company_id)
-communicateInterview(internship)

CV ManagerI

-uploadStudentCV(student_id, CV)

Recommendation ManagerI
-generateRecommendation(id)

Application ManagerI
-applyFor(student_id,intershipId, documents..)
-sendApplicationMsg(company_id, text)
-acceptRecommendationAndApply(student_id, intership_id, documents..)

Complaint ManagerI
-sendComplaint(id,intershipId,text)

Feedback and Suggestion Manager
-sendFeedback(id, text)

ModelI
-checkCredentials(email, password, role)
-createNewStudent(email_student, password, university)
-createNewCompany(email_company, password, company_id)
-insertPersonalInfo(id, personalInfo)
-updatePersonalInfo(id, peronalInfo)
-uploadStudentCV(student_id, CV)
-getInterships(keyword)
-applyFor(student_id,intershipId, documents..)
-sendFeedback(id, text)
-sendComplaint(id,intershipId,text)
-postInternship(company_id, internship)
-getSubmitedApplication(intership_id)
-organizeInterview(internship_id, student_id, company_id)
-generateRecommendation(id)

Notification ManagerI
-sendApplicationMsg(company_id, text)
-sendInterviewMsg(id, internship)
-sendVerificationMsg(id)

Profile ManagerI
-uploadStudentCV(student_id, CV)
-insertPersonalInfo(id, personalInfo)
-updatePersonalInfo(id, peronalInfo)

Registration ManagerI
-createNewStudent(email_student, password, university)
-createNewCompany(email_company, password, company_id)

Login ManagerI
-checkCredentials(email, password, role)

DBMSI

-checkCredentials(email, password, role)
-createNewStudent(email_student, password, university)
-createNewCompany(email_company, password, company_id)
-insertPersonalInfo(id, personalInfo)
-updatePersonalInfo(id, peronalInfo)
-uploadStudentCV(student_id, CV)
-getInterships(keyword)
-saveApplication(student_id,intershipId, documents..)
-saveFeedback(id, text)
-saveComplaint(id,intershipId,text)
-addInternship(company_id, internship)
-getSubmitedApplication(intership_id)
-addInterview(internship_id, student_id, company_id)
-getRecommendation(id)

2.6 Selected Architectural Style and Patterns

2.6.1 Three-tiered architecture

The description is reported in the section 2.1.1

Here are the reasons for these chose:

- **Modularity** and **Maintainability**: This architecture separates the application into independent modules, such as the user interface and data management, each handling specific tasks. This modular design makes the system more understandable, easier to develop, and simpler to maintain, as individual components can be updated or modified without affecting the entire system. It enhances flexibility, allowing for efficient troubleshooting and easier scaling of the platform over time.

- **Scalability**: By dividing the system into three distinct layers, it becomes easier to scale and expand various functionalities as needed. For example, additional application servers can be added to handle increased traffic or user requests, while more database servers can be introduced to manage a larger volume of data. This layered approach allows for targeted scaling, ensuring that each part of the system can grow independently based on demand, leading to better performance and resource management.

- **Reusability**: This architecture promotes the reuse of code and components, making it easier to share functionalities across different user interfaces. For example, the data tier can be accessed and utilized by multiple applications, while common features and services can be reused across various parts of the system. This enhances efficiency by reducing the need for redundant development efforts, speeding up development cycles, and ensuring consistency across the platform.

- **Security**: By placing the data tier within an internal network and allowing access only through the application tier, the architecture strengthens access control and safeguards the database from unauthorized access. Additionally, using firewalls between the presentation tier and the application tier, as well as between the application tier and the data tier, further enhances the overall system security. These layered security measures help prevent potential vulnerabilities and ensure that sensitive data is well protected at each level of the system.

2.6.2 Model View Controller (MVC)

The **Model-View-Controller** (MVC) architectural pattern is a popular design approach in software development, offering a structured and organized method for building applications.

It separates an application into three interconnected components:

- **Model**: Represents the application's data and business logic. It is responsible for retrieving, processing, and storing data, as well as maintaining the state of the application. The Model is independent of the user interface (UI) and does not depend on how the data is presented.

- **View**: Represents the user interface (UI) components of the application. It displays the data from the Model to the user and listens for user input. The View is concerned only with the presentation of data, not with business logic or data management.

- **Controller**: Acts as an intermediary between the Model and the View. It listens for user input from the View, processes the input (using the Model), and then updates the View with the new data. The Controller handles the user's actions and updates the application state accordingly.

It is useful because:

- Separation of Concerns: MVC divides the application into distinct components with well-defined responsibilities, making the codebase easier to manage, understand, and maintain. Each component can be developed and modified independently without affecting others.
- Maintainability: Since the Model, View, and Controller are separate, it's easier to update or modify a specific part of the application without disrupting the rest of the system.
- Reusability: The separation allows components to be reused across different parts of the application.
- Testability: With clear separation between components, it's easier to write unit tests. You can test the Model independently from the View and Controller, ensuring that the business logic is working correctly without worrying about the UI.
- Parallel Development: Different teams or developers can work on the Model, View, and Controller independently, speeding up development. For example, designers can work on the View while developers focus on the Model and Controller.

2.7 Other Design Decisions

This section outlines some of the design decisions made for the system to ensure it functions as intended.

2.7.1 Avaliability

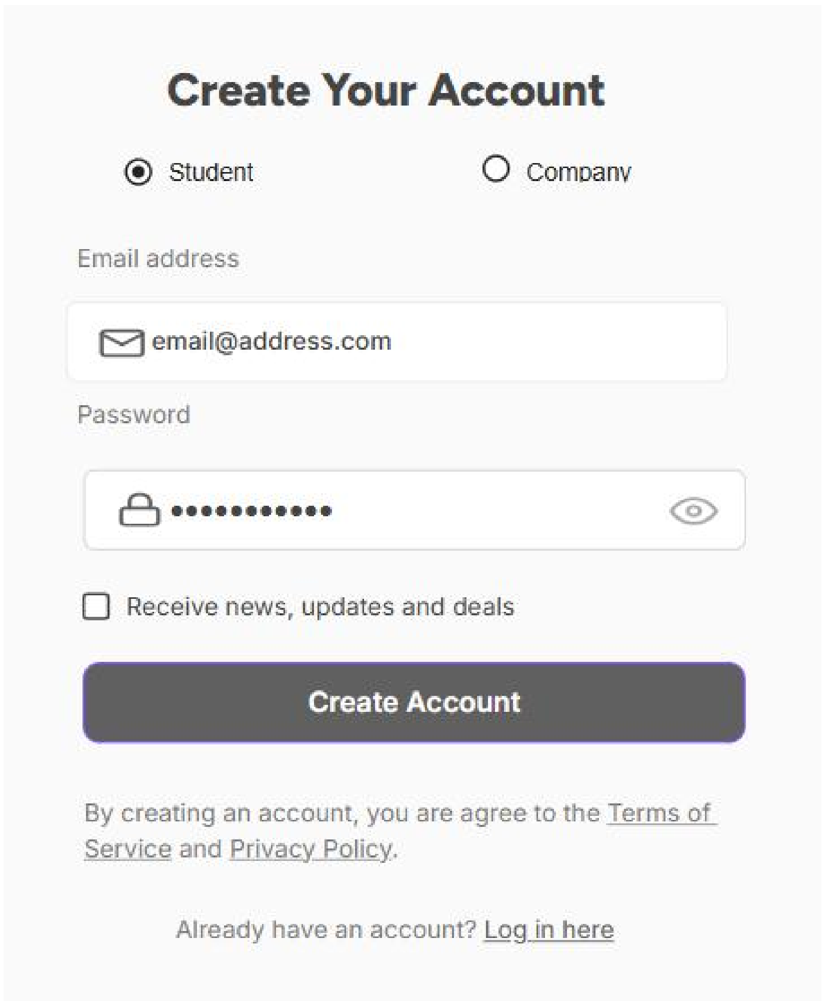
The implementation of load balancing and replication mechanisms greatly improves the reliability and availability of our system. Load balancing optimizes resource usage by distributing requests evenly across multiple servers, helping to avoid performance bottlenecks. At the same time, replication enhances fault tolerance by duplicating critical data and services. This redundancy reduces the impact of potential failures, ensuring that the system maintains consistent data management and service availability, even under adverse conditions.

2.7.2 Data Storage

To improve operational efficiency and simplify data management, we have selected a unified DBMS that contains information about both students and companies. This approach streamlines data retrieval and updates by utilizing the interconnected relationships between these components, reducing both time and complexity in managing the system.

3. User Interface Design

The majority of the key interfaces have already been introduced in the Requirements Analysis and Specification Document (RASD). In this Detailed Design (DD) document, in addition to the crucial interfaces, some additional ones have been included.


Create Your Account

[✓ Student] Company

Email address

email@address.com

Password

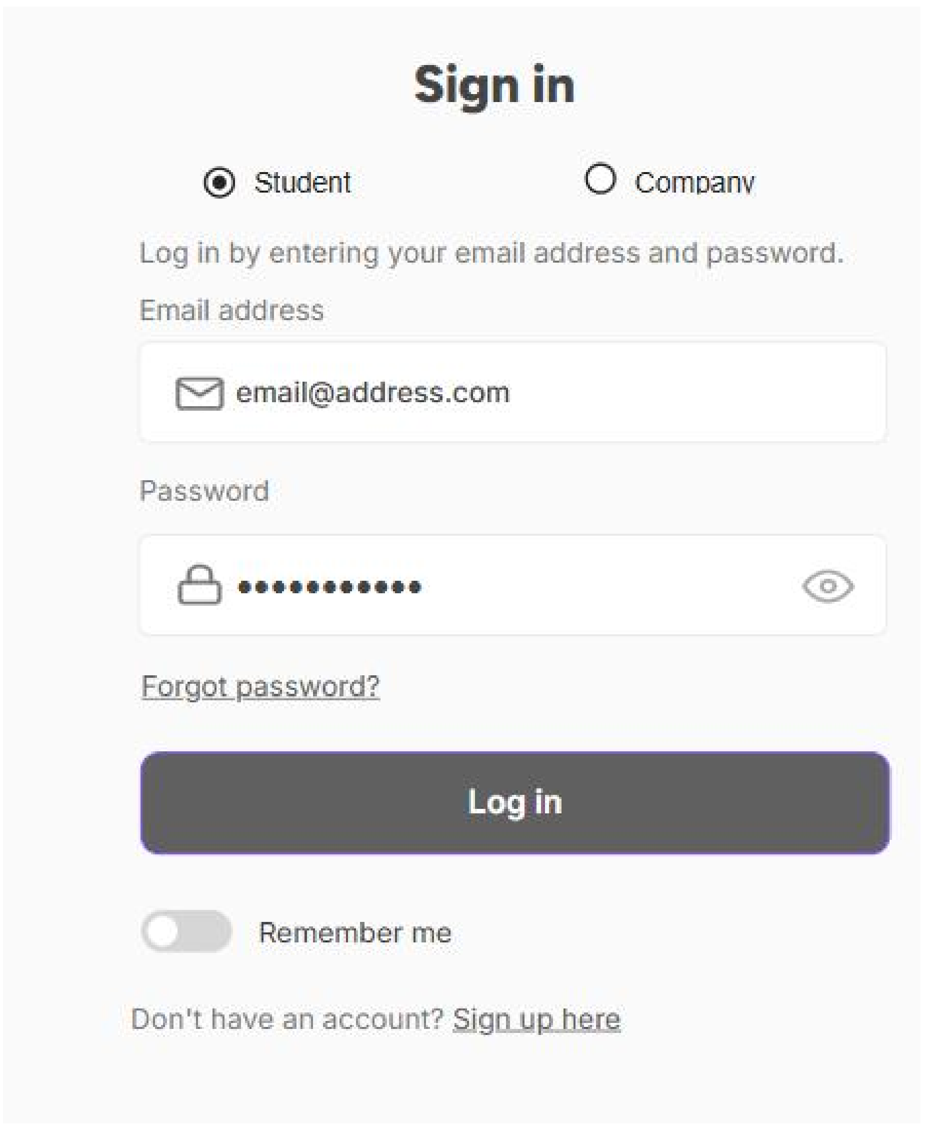
••••••••••••

☐ Receive news, updates and deals

By creating an account, you are agree to the Terms of Service and Privacy Policy.

Already have an account? Log in here

Figure 3.1 - S&C Sign up Interface


Sign in

[✓ Student] Company

Log in by entering your email address and password.

Email address: email@address.com

Password: ●●●●●●●●●●●●

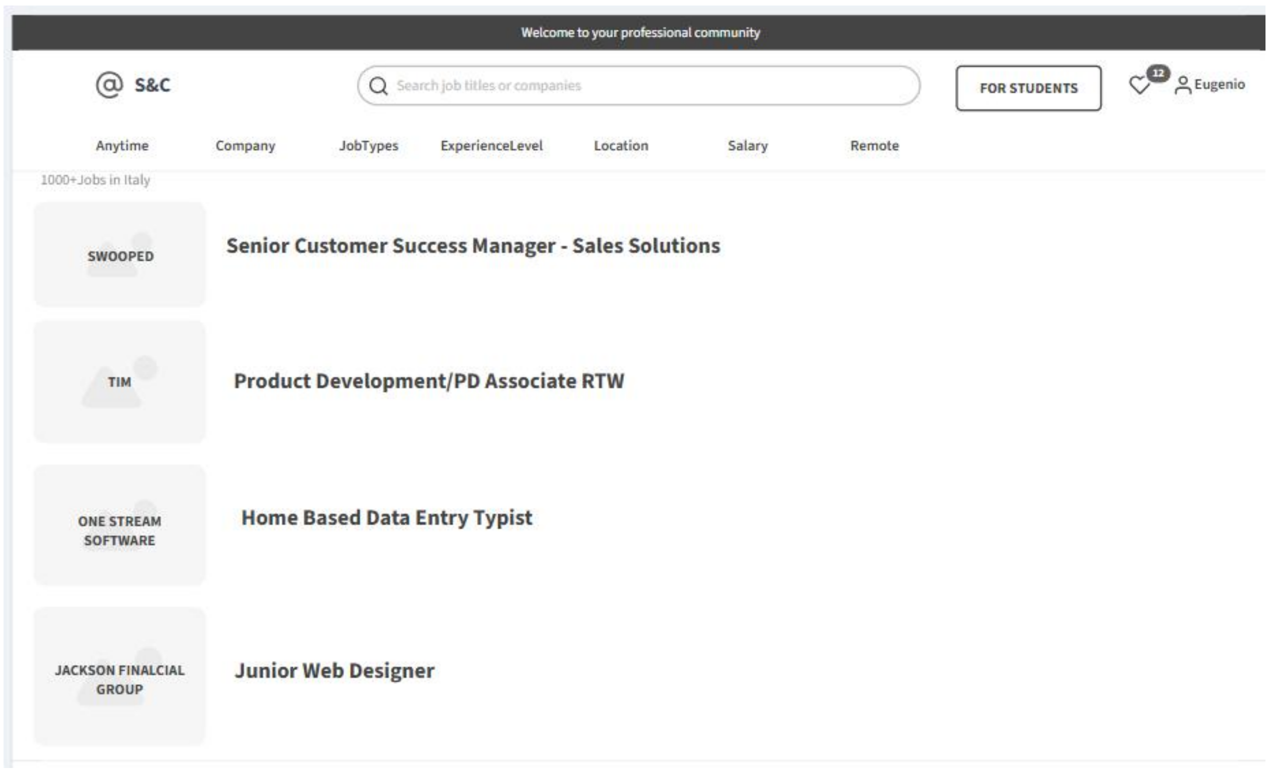
Forgot password?

Log in

Remember me

Don't have an account? Sign up here

Figure 3.2 - S&C Login Interface



1000+Jobs in Italy

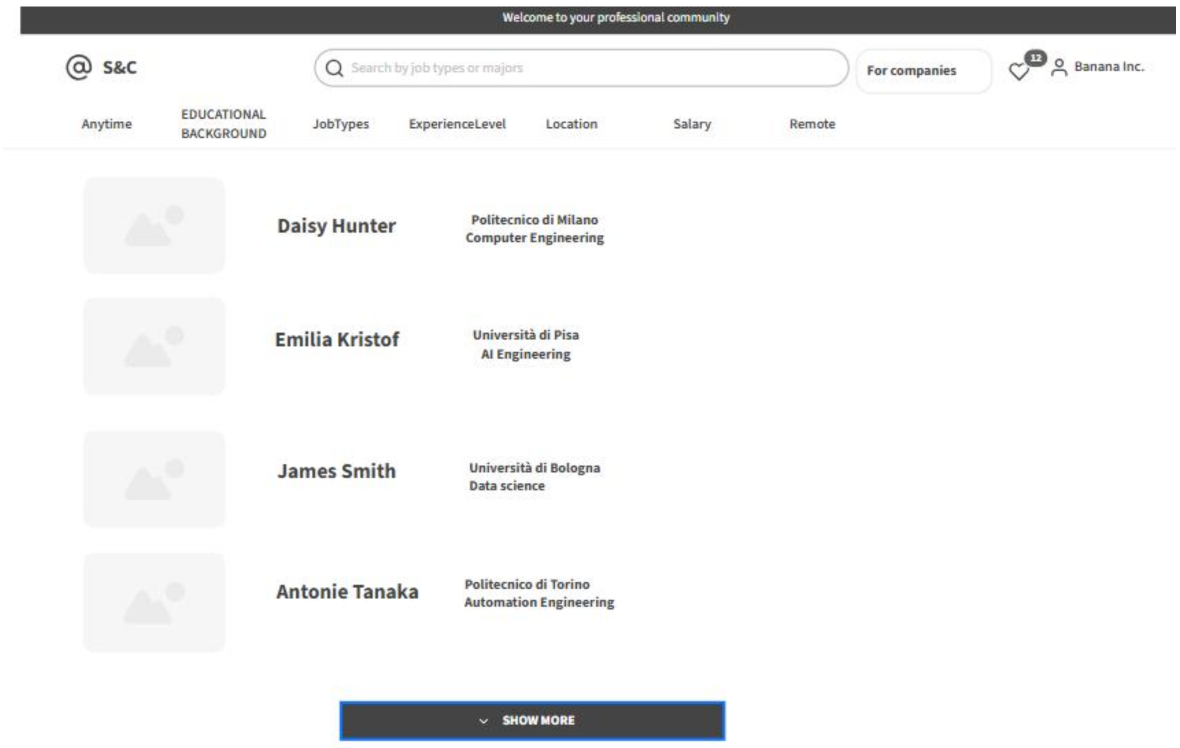
Senior Customer Success Manager - Sales Solutions

Product Development/PD Associate RTW

Home Based Data Entry Typist

Junior Web Designer

Figure 3.3 - S & C Student Recommendation Page



Welcome to your professional community

S&C

Search by job types or majors

For companies

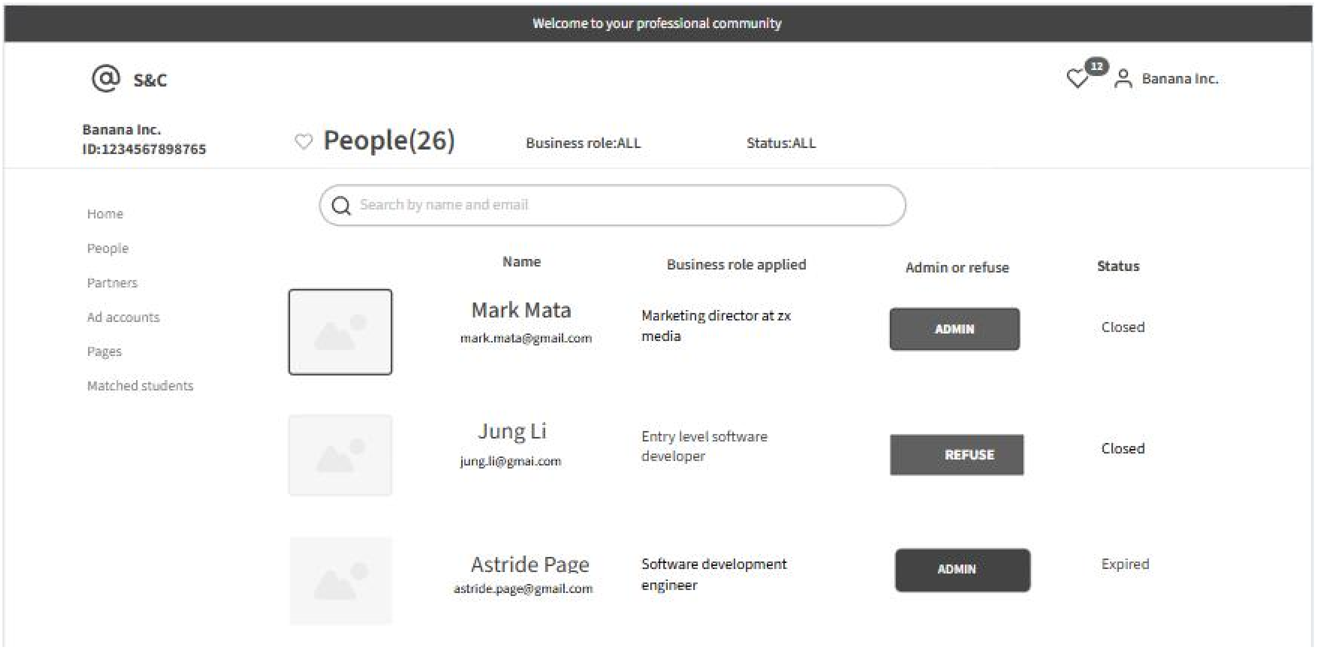
12 Banana Inc.

Anytime | EDUCATIONAL BACKGROUND | JobTypes | ExperienceLevel | Location | Salary | Remote

Daisy Hunter	Politecnico di Milano Computer Engineering
Emilia Kristof	Università di Pisa AI Engineering
James Smith	Università di Bologna Data science
Antonie Tanaka	Politecnico di Torino Automation Engineering

SHOW MORE

3.4 - S & C Company Recommendation Page



Welcome to your professional community

S&C

12 Banana Inc.

Banana Inc.
ID:1234567898765

People(26)

Business role:ALL

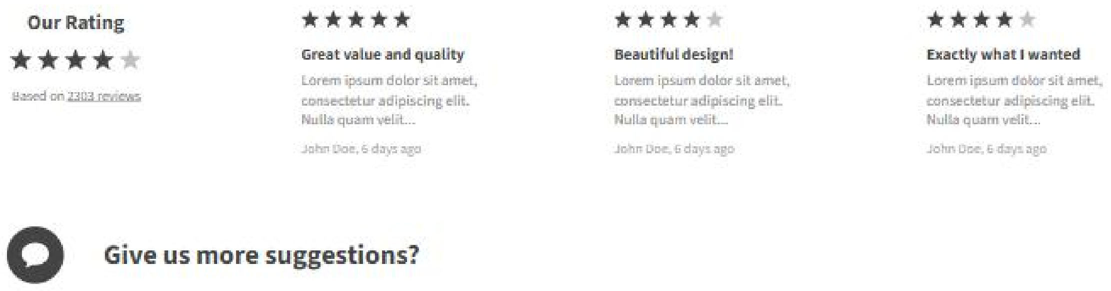
Status:ALL

Home
People
Partners
Ad accounts
Pages
Matched students

Search by name and email

Name	Business role applied	Admin or refuse	Status
Mark Mata mark.mata@gmail.com	Marketing director at zx media	ADMIN	Closed
Jung Li jung.li@gmai.com	Entry level software developer	REFUSE	Closed
Astride Page astride.page@gmail.com	Software development engineer	ADMIN	Expired

Figure 3.5 - S & C Company Home Page



Our Rating

★★★★☆

Based on 2303 reviews

★★★★★

Great value and quality

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit...

John Doe, 6 days ago

★★★★☆

Beautiful design!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit...

John Doe, 6 days ago

★★★★☆

Exactly what I wanted

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit...

John Doe, 6 days ago

Give us more suggestions?

Figure 3.6 -S&C Feedback and suggestion Pag

4.Requirements Traceability

This mapping illustrates how each functional requirement is addressed by specific design components within the S&C system, providing a clear linkage between the requirements and the system's architecture.

Requirements:	**[R1]** S&C allows Students to sign up. **[R2]** S&C allows Companies to sign up.
Components:	• WebApp • WebServer • Application Server: o Dispatcher o RegistrationManager o AccountManager o Model • DBMS: o PrivateDBMS

Requirements:	**[R3]** S&C allows Students to log in. **[R4]** S&C allows Companies to log in.
Components:	• WebApp • WebServer • Application Server: o Dispatcher o LoginManager o AccountManager o Model • DBMS: o PrivateDBMS

Requirements:	**[R5]** S&C allows Student to upload their CV. [R6] S&C allows Student to set university, skills, experience, preferences internship position.

Components:	• WebApp
	• WebServer
	• Application Server:
	  o Dispatcher
	  o Account Manager
	  o Profile Manager
	  O CV Manager
	  o Model
	• DBMS:
	  o SharedDBMS

Requirements:	**[R7]** S&C allows Companies to create internship positions on the platform. **[R8]** S&C allows Companies to set an application deadline for an internship position. **[R9]** S&C allows Companies to set a document list needed for an internship position. **[R16]** S&C allows Companies to manage their posts.

Components:	• WebApp
	• WebServer
	• Application Server:
	  o Dispatcher
	  o Internship Manager
	  o Model
	• DBMS:
	  o SharedDBMS

Requirements:	**[R11]** S&C allows Student to apply for internship positions.

Components:	• WebApp
	• WebServer
	• Application Server:
	  o Dispatcher
	  o Application Manager
	  o Model
	• DBMS:
	  o PrivateDBMS

Requirements:	**[R10]** S&C allows Student to search for internship positions.

Components:	• WebApp
	• WebServer
	• Application Server:
	  o Dispatcher
	  o Internship Manager
	  o Model
	• DBMS:
	  o SharedDBMS

Requirements:	**[R12]** S&C allows Student to send feedback and suggestions. **[R13]** S&C allows Companies to send feedback and suggestions.

Components:	• WebApp
	• WebServer
	• Application Server:
	  o Dispatcher
	  o Suggestion Manager
	  o Model
	• DBMS:
	  o PrivateDBMS

Requirements:	**[R14]** S&C allows Student to send complaints. **[R15]** S&C allows Companies to send complaints.

Components:	• WebApp
	• WebServer
	• Application Server:
	    o Dispatcher
	    o Complaint Manager
	    o Model
	• DBMS:
	    o PrivateDBMS

Requirements:	**[R17]** S&C allows Student and Companies to participate in interviews.

Components:	• WebApp
	• WebServer
	• Application Server:
	    o Dispatcher
	    o Interview Manager
	    o Model
	• DBMS:
	    o SharedDBMS

Requirements:	**[R18]** S&C notifies Students once there are new recommendations. **[R19]** S&C notifies Companies once there are new recommendations.

Components:	• WebApp
	• WebServer
	• Application Server:
	    o Dispatcher
	    o Recommendation Manager
	    o Notification Manager
	    o Model
	• DBMS:
	    o PrivateDBMS

Requirements:	**[R20]** S&C notifies Student about the result of an application.
Components:	• WebApp • WebServer • Application Server: o Dispatcher o Application Manager o NotificationManager o Model • DBMS: o PrivateDBMS

Requirements:	**[R21]** S&C notifies Companies about new applications request.
Components:	• WebApp • WebServer • Application Server: o Dispatcher o Internship Manager o Application Manager o NotificationManager o Model • DBMS: o PrivateDBMS

5.Implementation, Integration and Test Plan

5.1 Overview

In this final chapter, the system's implementation, integration, and testing strategy will be described. The approach chosen for this process is a Bottom-Up strategy.

With this strategy, the development begins with the lowest-level modules in the system's "uses" hierarchy, focusing on smaller functionalities that do not depend on other modules. Each of these modules will be tested using a driver, a temporary piece of code simulating other components. After each module is developed and tested, it is integrated into the system, replacing its corresponding driver. However, the newly integrated module will still require a new driver for further testing.

This Bottom-Up approach supports incremental integration, making it easier to identify and fix bugs, as testing starts with smaller, isolated parts of the system and expands as new modules are added. It also allows different development teams to work simultaneously on various functionalities, promoting parallel development.

5.2 Implementation plan

As mentioned in the overview, the implementation of this system will follow a bottom-up approach to avoid creating stub structures, which would be more challenging to implement and test. This method allows the development process to begin with the fundamental components, ensuring that each module is properly implemented and tested before moving on to the next, thereby facilitating a smoother and more manageable development cycle.

In particular:

- **Internal Database**: The internal database is designed to store all essential data, making its correct implementation crucial. The process should begin with designing and implementing the database schema based on the requirements of the data model. Once the schema is finalized, an appropriate database system, such as MySQL, should be selected to ensure efficient and reliable data management.
- **Application Server**: Implement the core logic that serves as the backbone of the application. This includes:
    - Choosing a Web Framework: Select a robust and scalable web framework such as Django to facilitate development.
    - User Authentication and Authorization: Develop secure mechanisms for user login, registration, and access control.

    - API Development: Create APIs to handle essential functionalities such as user registration, internships post, etc.
    - Database Integration: Ensure seamless integration with the internal database for efficient data retrieval and storage.
    - Real-Time Communication: Set up real-time communication channels with the web server for instant updates, such as scoring and notifications.

- **Web Server**: Responsible for managing user requests, serving static assets, and facilitating communication with the application server.
    - Choosing a Web Server Platform: Select an appropriate platform like Node.js to handle incoming traffic efficiently.
    - HTTP Request Handling: Configure the server to process incoming HTTP requests and route them to the application server for processing.
    - Static File Serving: Implement mechanisms to serve static assets, such as front-end files (HTML, CSS, JavaScript), efficiently.
    - Integration for Real-Time Updates: Establish seamless integration with the application server to enable real-time updates.

- **Front-end accessible through web browsers**:
    - Choosing a Frontend Framework: Select a suitable framework like Angular to build an interactive and responsive interface.
    - User Interface Design and Implementation

5.3 Feature Identification

[F1] Login and Registration Features

These fundamental features of S&C are essential for all users, including both companies and students. While their implementation will require relatively less time, they play a pivotal role in ensuring the proper functionality of the entire web application. As these features form the backbone for subsequent workflows, they will be prioritized and developed first to support the seamless integration of more advanced functionalities.

[F2] Post Features

This is one of the features used by company parts. This consent companies to post for internship positions, including setting an application deadline for an internship position and setting a document list needed for an internship position.

[F3] Profile Management

These features are reserved for all registered users. Allows users to create, update, and manage their profiles, including uploading CVs and personal information for students.

[F4] Search Features

For students these features include the use of the search bar on the website in order to retrieve the list of the ongoing internship positions that students can apply for its. Instead for companies these features include the possibility to check already submitted applications for a specific internship position. Employs algorithms to suggest internships to students based on their profiles and notify companies about students whose CVs match their requirements.

[F5] Suggestion Features

These features are reserved for all registered users. Allows users to give suggestions to S&C services.

[F6] Recommendation System

Employs algorithms to suggest internships to students based on their profiles and notify companies about students whose CVs match their requirements.

[F7] Complaint Features

These features are reserved for all users that are taking part in an ongoing internship period. This consent companies or students to complain about an internship experience.

[F8] User notification

feature that manages the email notifications on user's devices

5.4 Component Integration and Testing

This section provides an overview of the integration and communication between components, as well as the testing strategies employed to ensure their functionality.

The initial component to be implemented is the Model. It will be tested using a driver to simulate the behavior of other components that are yet to be developed. The Model is crucial as it manages all interactions with the database and serves as a foundation required by most of the other components.

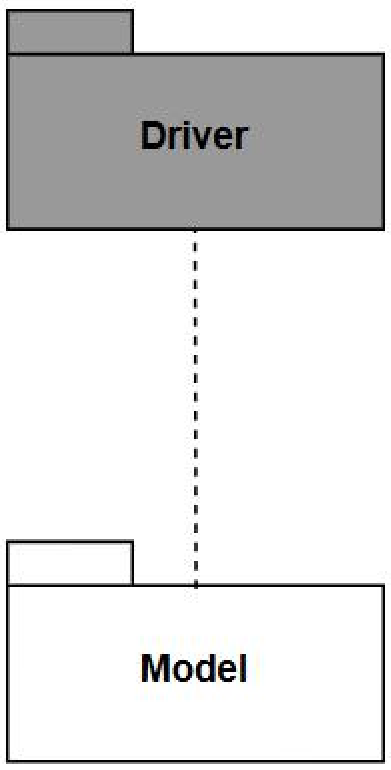


Figure 5.4.1

The integration of components will proceed with the Login and Registration features:

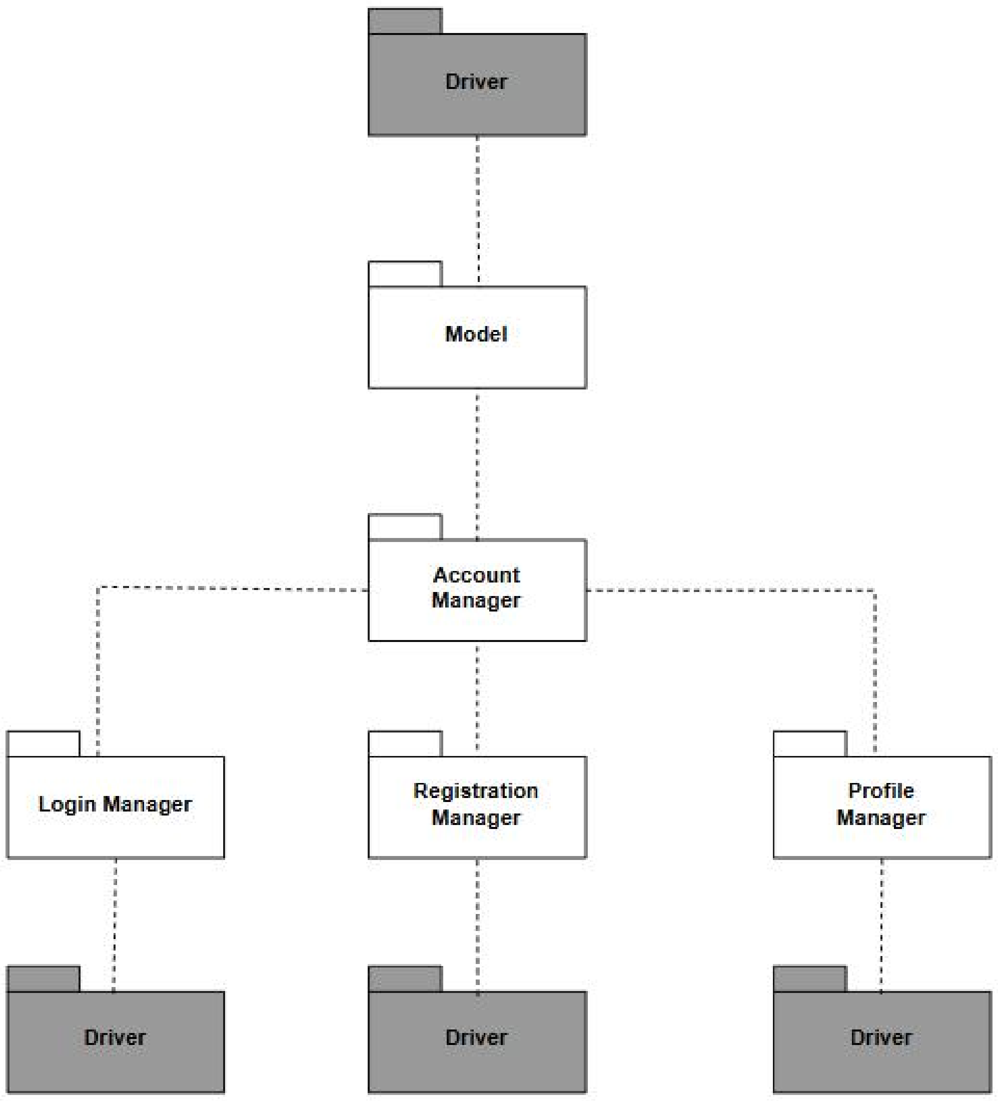


Figure 5.4.2

Then, we can implement two new features: Internship Manager and CV Manager

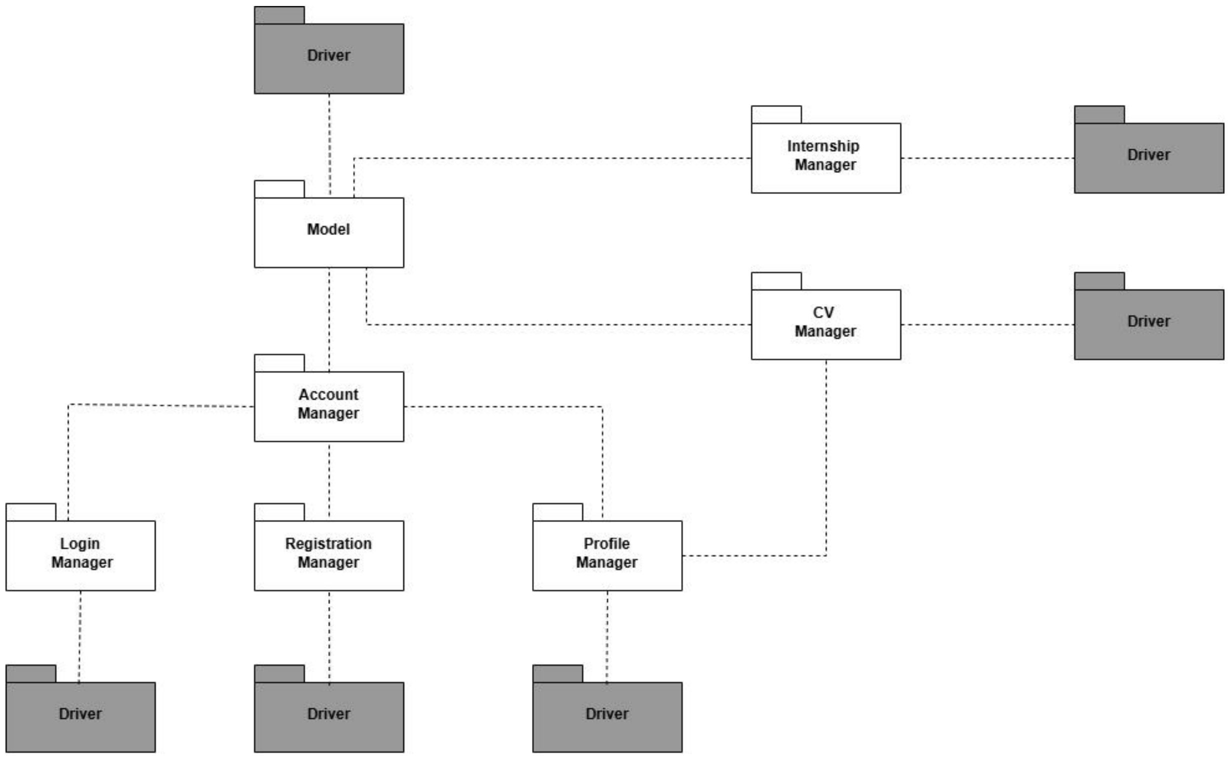


Figure 5.4.3

After that, we can immediately implement features about student side: Application manager.

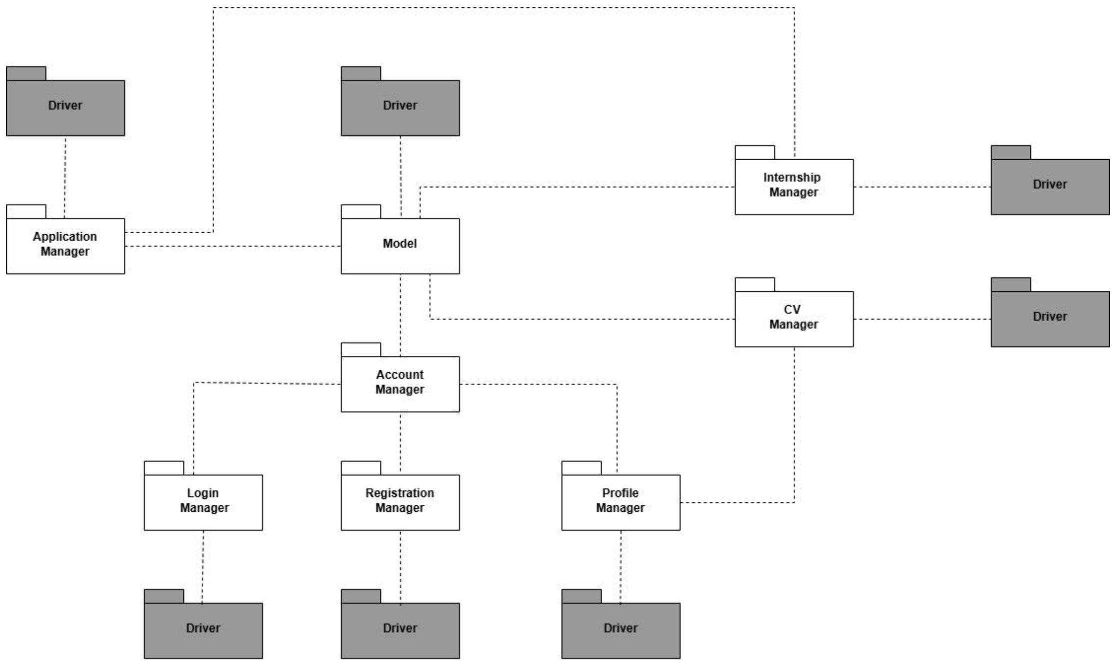


Figure 5.4.4

Then the implementation will continue the student side with the Interview Manager

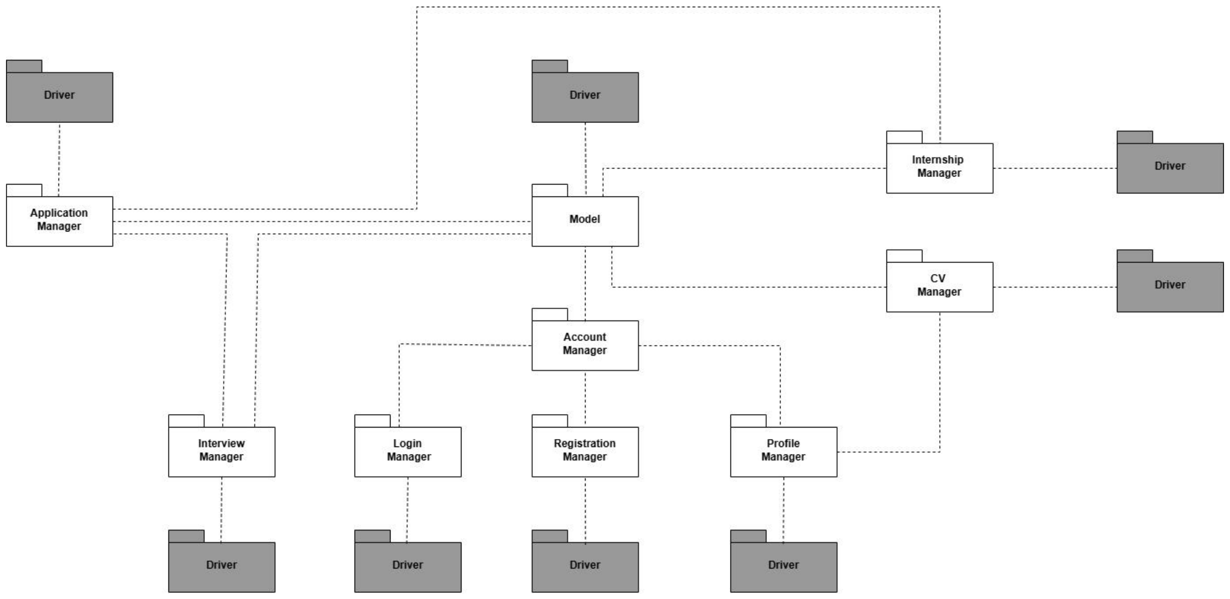


Figure 5.4.5

Now two features for both students and companies: Suggestion Manager and Complaint Manager

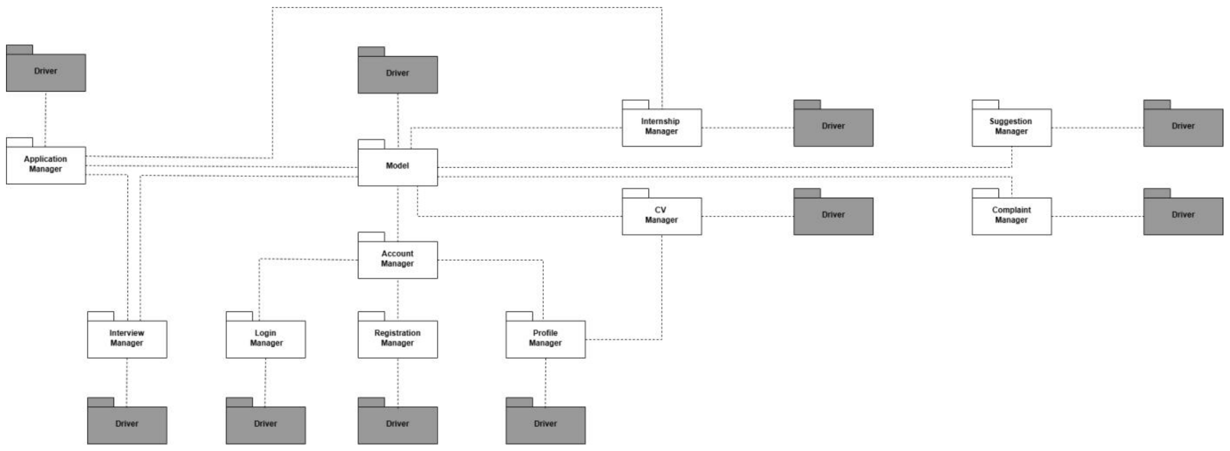


Figure 5.4.6

In sequence one of the main features of the platform: Recommendation manager

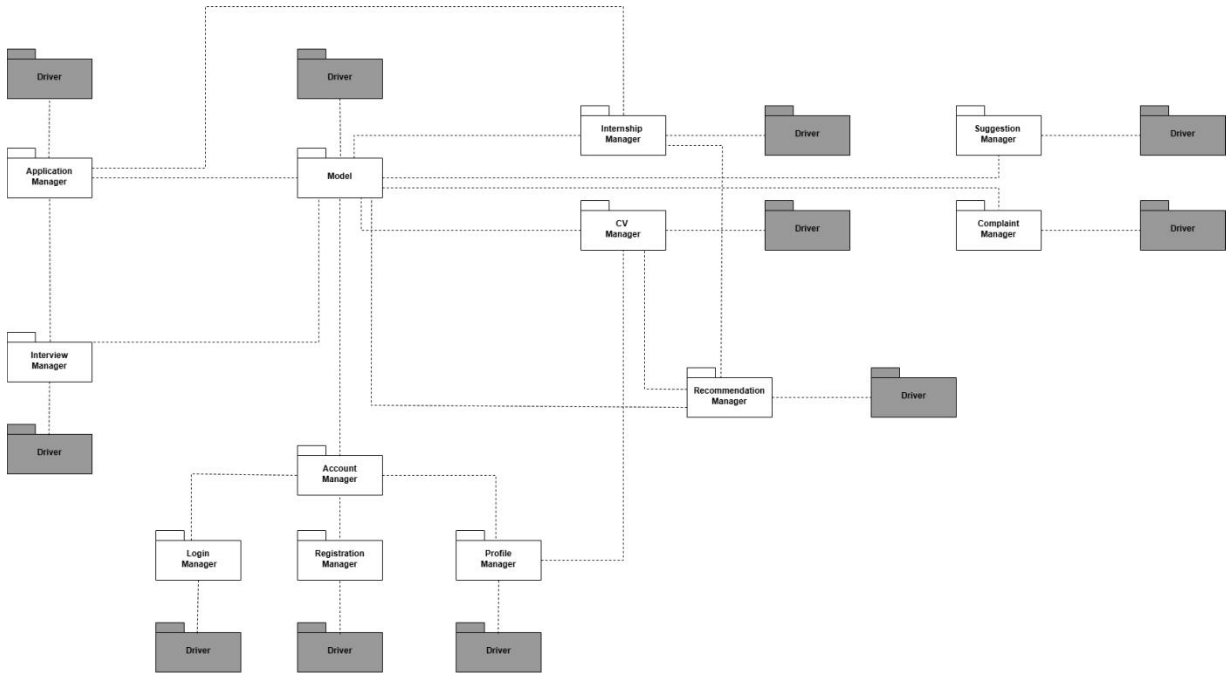


Figure 5.4.7

Finally, the NotificationManager

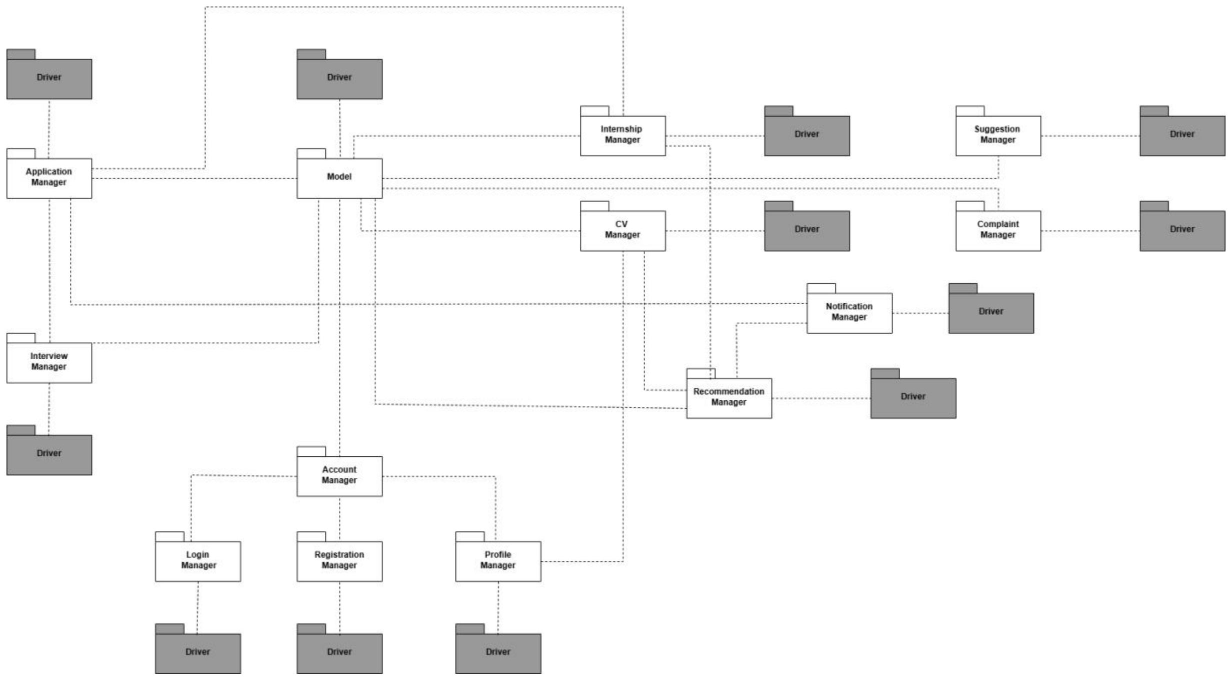


Figure 5.4.8

5.5 System testing

The S&C platform is subjected to a thorough system testing process to verify its reliability, performance, and usability. These testing efforts are carried out across various levels of detail and scope.

5.5.1 Component Testing

Throughout the development phase, individual components or modules are tested independently to confirm their functionality. To simulate the behavior of interconnected modules, Driver and Stub components may be used. These simulate both normal and abnormal behaviors, enabling the evaluation of each component's reliability and robustness.

5.5.2 Integration Testing

Once individual component testing is complete, the system proceeds to integration testing. The S&C platform utilizes a combination of bottom-up and thread-based strategies for integration, ensuring that all components function harmoniously together as a cohesive system.

5.5.3 System Testing

After the system has been implemented, unit tested, and integrated, it enters a thorough system testing phase. This phase ensures that all features are developed correctly and meet both the functional and nonfunctional requirements specified in the Requirements Analysis and Specification Document (RASD). System testing is not limited to the development team and may also involve stakeholders to ensure comprehensive evaluation.

Testing steps:

1.Functional Testing:

Functional testing will be conducted on the system to ensure that the workflow operates correctly and aligns with the functionalities outlined in the RASD document. This includes verifying that the goals, requirements, and use cases are met, as well as ensuring that the described scenarios can be accurately simulated.

2.Load Testing:

Load testing is essential for identifying potential issues such as memory leaks, buffer overflows, and inefficient memory management.

3.Performance Testing:

The system will undergo load testing to identify performance bottlenecks and assess its resilience under heavy workload conditions. This testing ensures that the system can support a large number of users concurrently while maintaining low response times. Additionally, it will help pinpoint areas where the software’s algorithms and overall performance can be optimized.

4.Stress Testing:

To ensure the system can recover after a failure, Stress Testing will be conducted by simulating a high volume of concurrent users or by reducing the system's computational resources. This testing will help assess the system's ability to handle extreme conditions and identify how it behaves under stress, ensuring that it can recover gracefully without data loss or significant disruption.

5.User Interface Testing:

It is crucial that the system functions properly across various devices and browsers, as outlined in the Requirements Analysis. This involves testing the usability and accessibility of the web app on different platforms to ensure it works seamlessly for both types of users, students and companies. The goal is to verify that the user experience remains consistent and efficient, regardless of the device or browser used.

5.6 Additional specifications on testing

5.6.1 Security Testing

- Authentication and Authorization:
  - Ensure that the user authentication process effectively validates user identities and prevents unauthorized access.
  - Test the authorization mechanisms to ensure that users can only access resources and perform actions appropriate to their roles (e.g., student or company).
  - Identify and address any vulnerabilities that may expose sensitive information or allow unauthorized access to the system, ensuring compliance with security standards.

5.6.2 Accessibility Testing

- Compliance with accessibility Standards:
  - Ensure that the S&C platform complies with relevant accessibility standards, such as WCAG (Web Content Accessibility Guidelines), to provide an inclusive experience for users with disabilities.

5.6.3 User Acceptance Testing

- End-User Validation:
  - Involve end-users and stakeholders in the testing process to ensure the platform's usability aligns with their needs and expectations. Collect their feedback to make iterative improvements and enhance the platform based on their user experience.

6. Effort Spent

The time tables written below represent just an approximation of the time spent for the writing and discussions the team had for each specific chapter of this document.

Giovanni Ni

Chapter	Effort (in hours)
1	3
2	25
3	11
4	18
5	15

Xinyue Gu

Chapter	Effort (in hours)
1	1
2	22
3	15
4	10
5	18

7 .References

7.1 Reference

The Requirement Engineering and Design Project specification document A.Y. 2024 – 2025

7.2 Used Tools

-User interface design: moqups.com

-Testing models and component architecture: draw.io

-Sequence diagram: sequencediagram.org

-High level architecures: draw.io