POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Students&Companies

Design Document

Andrea Carrara
and Federica Currò Dossi

January 7, 2025

Contents

1 | Introduction

This chapter provides essential background information about the design document, including its purpose, scope and structure. It also introduces key terminology and references needed for understanding the technical content that follows.

1.1. Purpose

The purpose of the following document is to present a detailed design description of Students&Companies. It provides developers with implementation guidelines while serving as a technical agreement between customers and contractors. The document transforms the requirements and specifications from the requirements analysis and specification document [1] into concrete architectural decisions, describing the system's components and their interactions. It outlines the design patterns, technical interfaces and deployment strategies that will guide the development team in building a robust platform that fulfills stakeholders' needs. Additionally, it establishes a clear roadmap for implementation and testing phases, ensuring that the final system aligns with both technical requirements and business objectives.

1.2. Scope

Students&Companies is designed to bridge the gap between academic education and practical workplace experience. The platform facilitates the internship matching process by enabling students to create detailed profiles with CVs and preferences, while companies can post comprehensive internship positions including project details, required skills and terms. S&C features personalized recommendations through statistical analysis, manages the entire selection process from interviews to final outcome and provides tools for monitoring ongoing internships. Additionally, universities can oversee the progress of their students, ensuring quality and addressing concerns that arise during internships. For a comprehensive overview of the platform's functionalities and requirements, refer to the requirements analysis and specification document [1].

1.3. Glossary

This table contains the key definitions, acronyms and abbreviations used in the document.

ID	Description
S&C	Students&Companies
US	University student
IN	Internship job
PO	Internship position
CO	Company
UN	University
Match	A US and a CO declare interest in each other
Contact	Selection process and internship progress
Rn	Requirement number `n`

Table 1.1: Glossary

1.4. Reference Documents

The document follows Professor Di Nitto's project assignment structure [2].

1.5. Document Structure

This document describes the design specification for Students&Companies, following an architecture-centric approach that progresses from system-wide design decisions to specific implementation details. It is organized into six chapters, each providing increasingly detailed technical information needed for implementation.

Introduction

The first chapter establishes the foundation of the design document through purpose and scope definitions, provides a comprehensive glossary, lists reference documents and explains the document's organization.

Architectural Design

The second chapter presents the system's architecture through multiple views: an overview of components and their interactions, detailed component descriptions, deployment specifications, runtime behavior illustrations, component interfaces and the rationale behind architectural styles and patterns.

User Interface Design

The third chapter details the user interface through mockups and interaction flows, covering each user category's specific interface requirements and presenting key screens and navigation patterns.

Requirements Traceability

The fourth chapter establishes clear links between the requirements outlined in the requirements analysis and specification document [1] and their corresponding design elements, demonstrating how architectural decisions satisfy functional and non-functional requirements.

Implementation, Integration and Test Plan

The fifth chapter provides a comprehensive roadmap for system realization, detailing the implementation order of components, their integration strategy and the testing methodology to ensure system quality.

Workload

The sixth chapter quantifies each author's time investment in the design document.

2 | Architectural Design

This chapter provides a detailed architectural design of Students&Companies, starting with a high-level overview of the system's design choices and their rationale. It then explores the low-level component view, illustrating major system modules and their interactions using various diagrams. A deployment view follows, showing how software components distribute across hardware nodes. Runtime views are then presented through sequence diagrams depicting key system operations. The chapter concludes with a discussion of chosen architectural styles, patterns and other significant design decisions.

2.1. Overview

Students&Companies requires an architecture that can efficiently handle the complex interactions between students, companies and universities while ensuring system scalability and data security. The platform adopts a three-tier client-server architecture, separating presentation, application logic and data management into distinct layers. This section provides an overview of the application of this architecture to the problem at hand, while a more general discussion of the architectural style belongs in later sections.

Presentation Tier

At the presentation tier, the web app serves as the user interface, allowing S&C to be accessed by users through their browsers. The app is responsive, ensuring accessibility across various devices while maintaining a consistent user experience.

Application Tier

The application tier consists of three server components.

The web server handles incoming client requests, managing user sessions and providing load balancing capabilities to distribute traffic effectively across multiple instances of the application server. The application server contains the core business logic, processing user requests to coordinate the entire internship lifecycle from application to completion.

The mail server manages the email workflow, determining when to send notifications for signup confirmations, selection outcomes or internship comments. For actual email delivery, the mail server integrates with an external email provider that handles the delivery infrastructure, ensuring reliable communication.

Data Tier

The data tier employs a DBMS server to store and manage all system data. This includes user profiles, internship positions, ongoing matches, selection processes and internship records. The DBMS server provides structured data storage with mechanisms for maintaining integrity through transaction management and constraint enforcement, optimizing query performance through indexing and caching, and implementing access controls and encryption for sensitive user data.

Overall, this architecture enables efficient data flow while maintaining scalability and security. When a user interacts with the web app, their request flows through the web server to one of the application server instances, which processes it leveraging the data from the DBMS server and triggers notifications through the mail server when necessary.

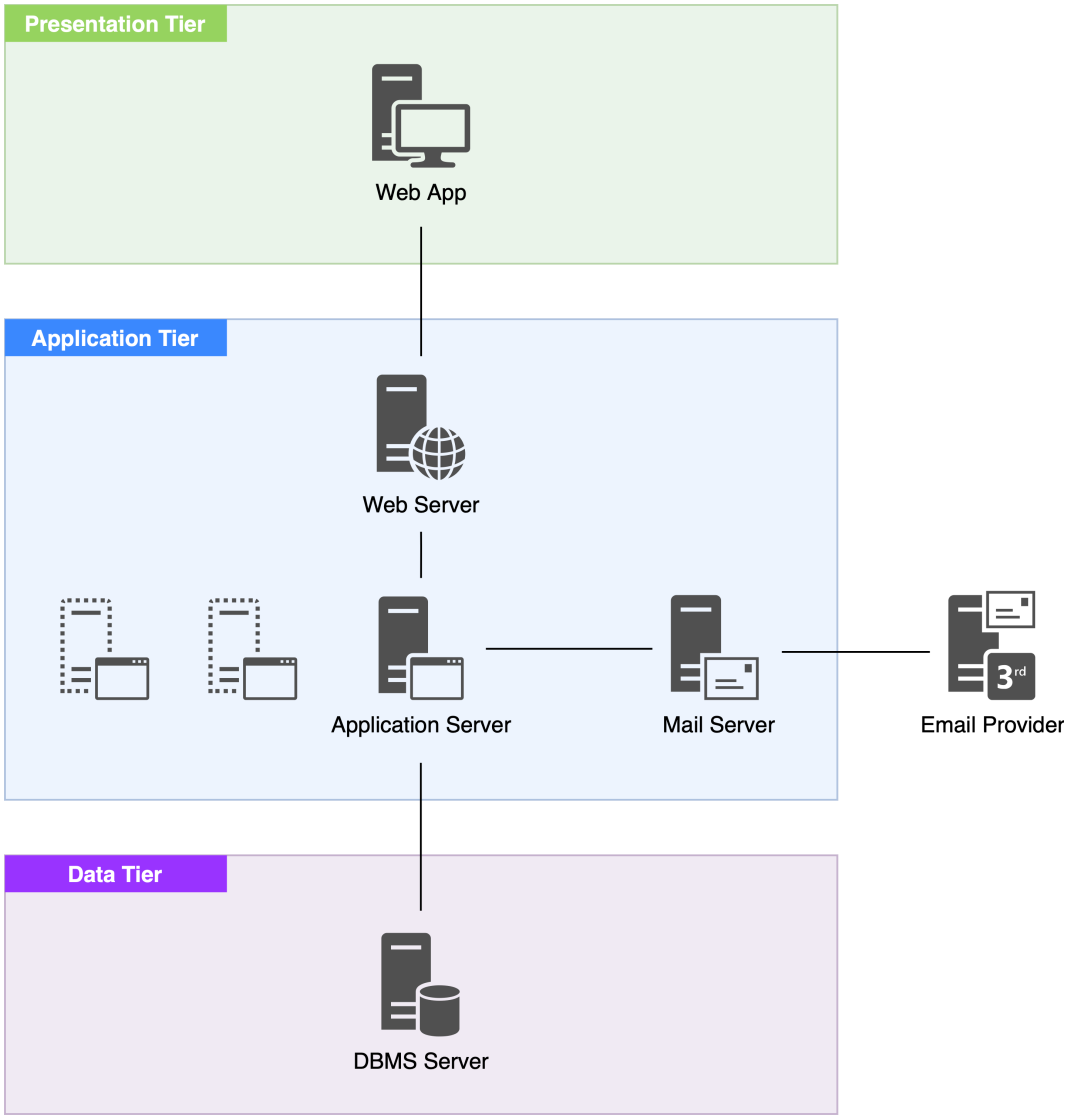

Figure 2.1: Architecture

2.2. Component View

This section illustrates the logical organization of Students&Companies by breaking down the system into its major software modules and mapping their relationships. Using a series of increasingly detailed diagrams, the component view first presents a high-level view showing the system elements and their interactions. It then dives into individual components, examining their low-level structures, responsibilities and interfaces, with particular attention to how they collaborate to implement the platform's functionality.

2.2.1. High-Level View

The high-level diagram below shows the components of Students&Companies and their interactions. The application server exposes five interfaces to the web app, each corresponding to a domain of functionality that will be further discussed in the next section.

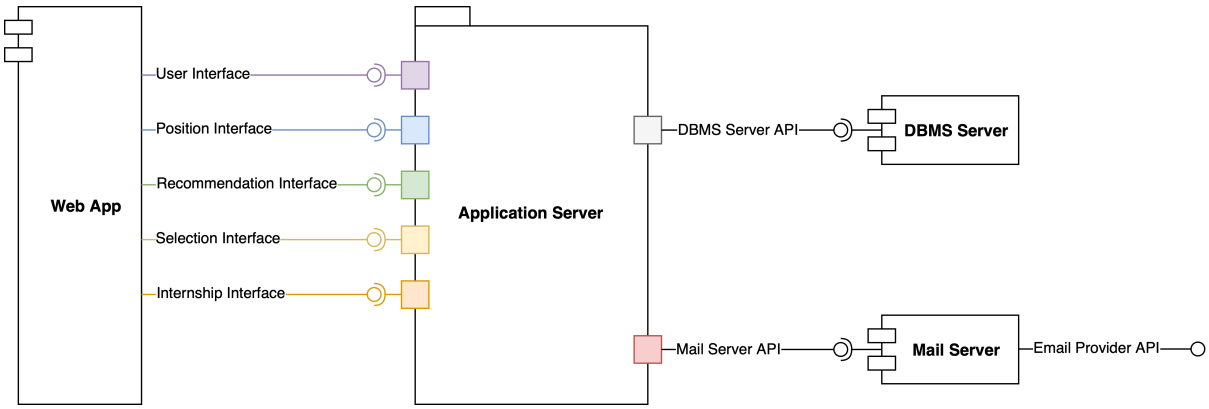

Figure 2.2: High-level component view

2.2.2. Low-Level View

The low-level view below builds upon the high-level diagram above by providing a detailed representation of the internal architecture of the application server. The codebase is structured according to the model-view-controller pattern. In this architecture, the model acts as the core data structure of the system, representing and managing the content of the database. In turn, managers act as controllers, bridging the model and the external interfaces, which serve as the entry points for users, ensuring a clean separation of concerns that makes the system modular, scalable and maintainable.

Each manager is organized into multiple submanagers, which encapsulate specific functionalities within its domain. These submanagers allow the managers to remain focused on their broader responsibilities while delegating finer tasks. The following subsections delve into each manager, describing its role and breaking down its submanagers.

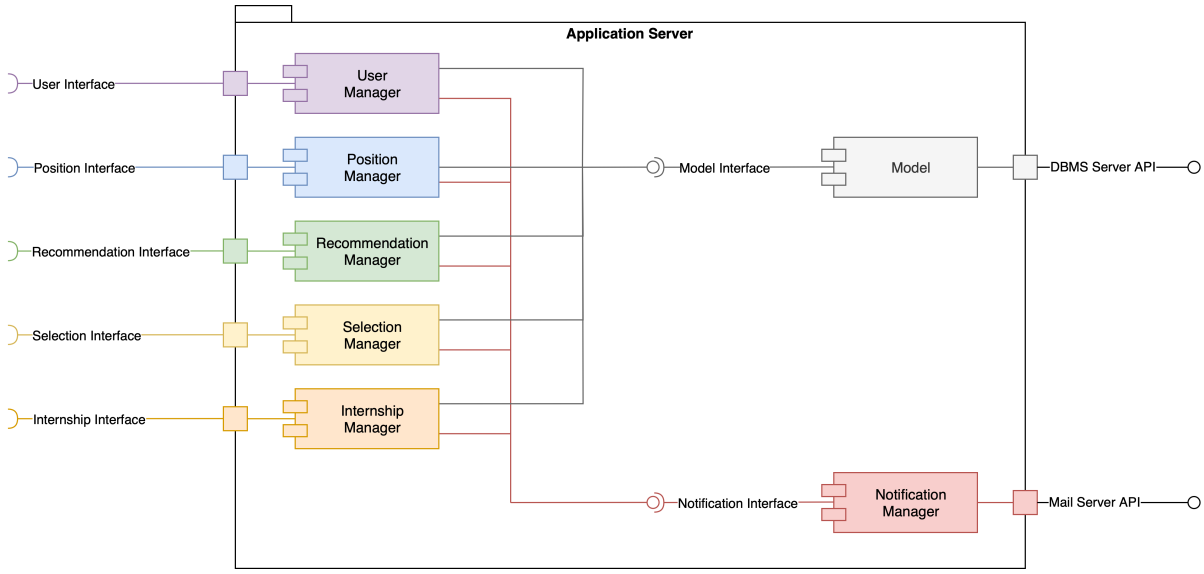

Figure 2.3: Low-level component view

User Manager

The user interface serves as the bridge between users and the user manager, channeling requests related to account creation, authentication and management.

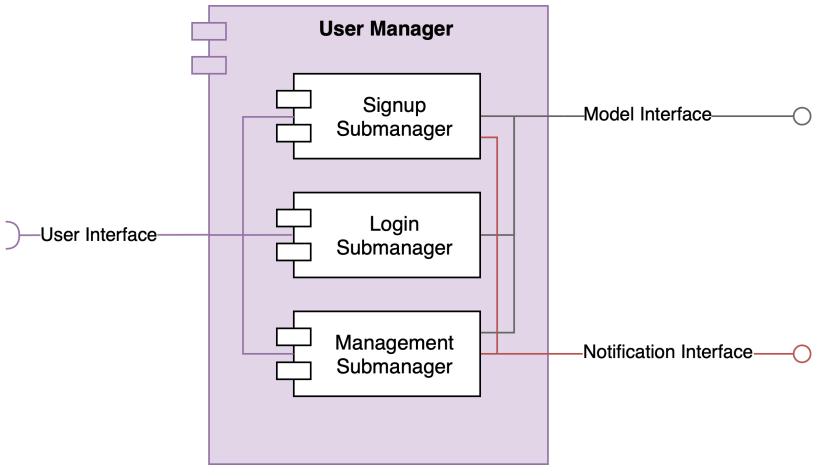

Figure 2.4: User manager

The user signup submanager handles user onboarding by validating fields and enforcing rules like password length. It securely integrates new users into the system while delegating confirmation emails to the notification manager.

The user login submanager supports user logins, validating credentials and issuing tokens for secure session management. It also ensures that expired tokens are revoked.

The user management submanager supports users in updating their profiles, including editing preferences and uploading CVs, validating and storing changes. Additionally, it provides an option for account deletion.

Position Manager

The position interface acts as the portal for managing internship postings, funneling requests from both companies and students to the position manager. This manager ensures that opportunities are validated, maintained and accessible to all relevant users.

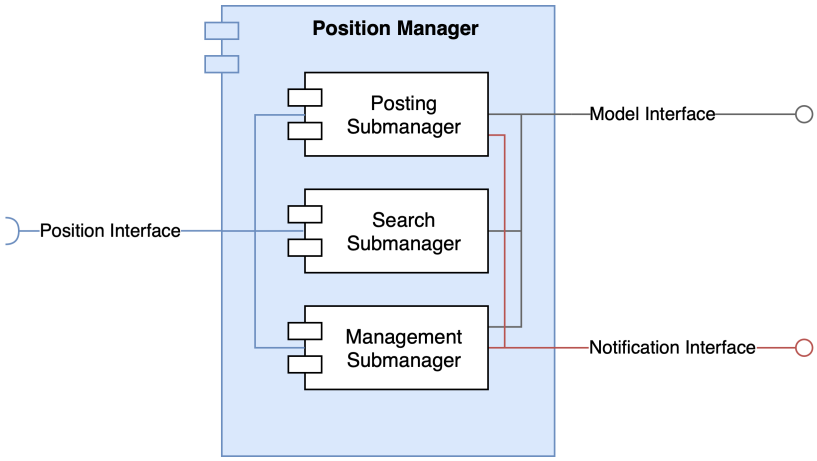

Figure 2.5: Position manager

The position posting submanager allows companies to post new positions, validating details such as required skills and terms.

The position search submanager empowers students to search for opportunities through keywords. It ranks and prioritizes results, highlighting the most relevant options.

The position management submanager supports companies in modifying existing position postings, ensuring all changes are logged, including updates in their status.

Recommendation Manager

The recommendation interface fuels the platform's personalized experience by connecting users with tailored matches. The recommendation manager transforms system data into actionable suggestions, enhancing engagement.

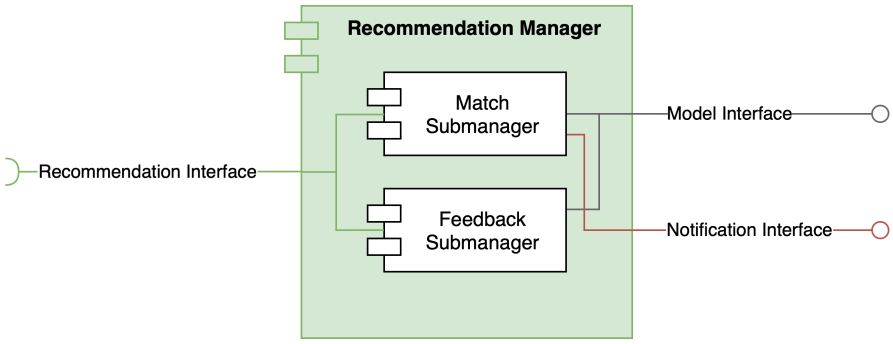

Figure 2.6: Recommendation manager

The recommendation match submanager applies statistical analysis to generate recom-

mendations by comparing user profiles and position descriptions. It also ensures users are informed about new recommendations by working with the notification manager.

The recommendation feedback submanager tracks feedback forms to refine future suggestions, improving relevance over time.

Selection Manager

The selection interface connects university students and companies through the selection process, channeling application and interview workflows to the selection manager.

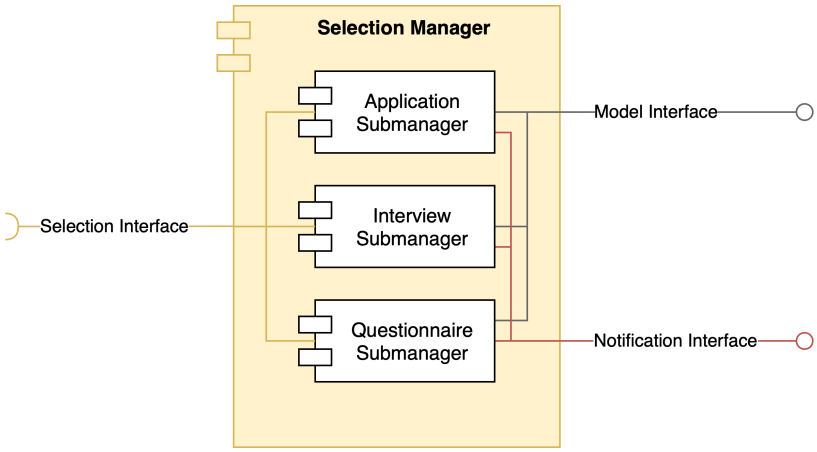

Figure 2.7: Selection manager

The selection application submanager oversees the lifecycle of applications, from initial submission to final outcome. It makes sure that both students and companies are informed throughout the workflow by integrating with the notification manager.

The selection interview submanager coordinates interviews by validating availability and resolving conflicts, ensuring schedules are efficiently managed.

The selection questionnaire submanager supports companies in evaluating candidates by managing the distribution and storage of questionnaires.

Internship Manager

The internship interface facilitates collaboration among students, companies and universities, channeling related workflows to the internship manager. This manager monitors ongoing internships, ensuring transparency.

The internship progress submanager records status updates, providing all stakeholders with insights into internship progress. Additionally, it collaborates with the notification manager to send update emails.

The internship comment submanager coordinates the posting of comments when progress is reported or a concern is raised.

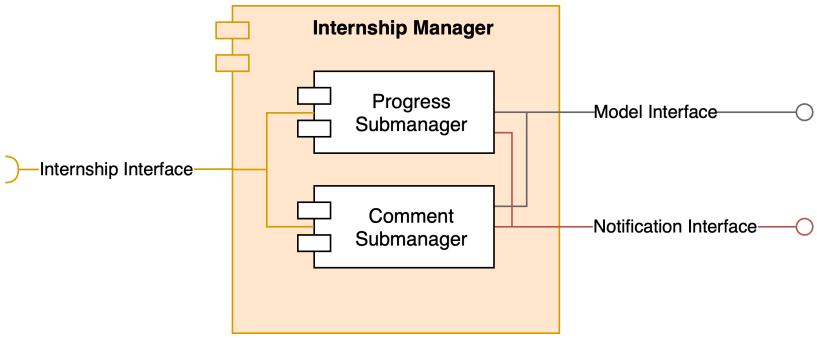

Figure 2.8: Internship manager

Notification Manager

The notification interface connects other managers to the notification manager, which acts as the centralized hub for preparing and sending all emails. By consolidating communication, it ensures consistency and reliability across the system.

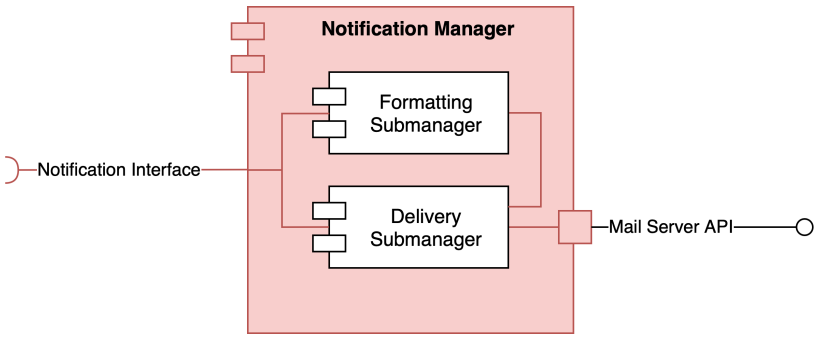

Figure 2.9: Notification manager

The notification formatting submanager prepares emails by structuring data provided by other managers, ensuring clarity for the recipient.

The notification delivery submanager communicates with the mail server API to send notifications, tracking delivery and retrying failed attempts to maintain reliability.

2.3. Deployment View

This section provides a detailed representation of how the software components are physically deployed on hardware nodes, together with their interactions. It outlines the infrastructure, focusing on the communication protocols, security measures and data management strategies employed in the system. The deployment view below is designed to align with the three-tier architecture defined in the overview.

A firewall is deployed between the browser and the web server to protect the internal infrastructure. It acts as the first line of defense by filtering incoming traffic and allowing only HTTPS requests on designated ports. This ensures that exclusively legitimate traffic reaches the web server, mitigating risks of unauthorized access or malicious attacks.

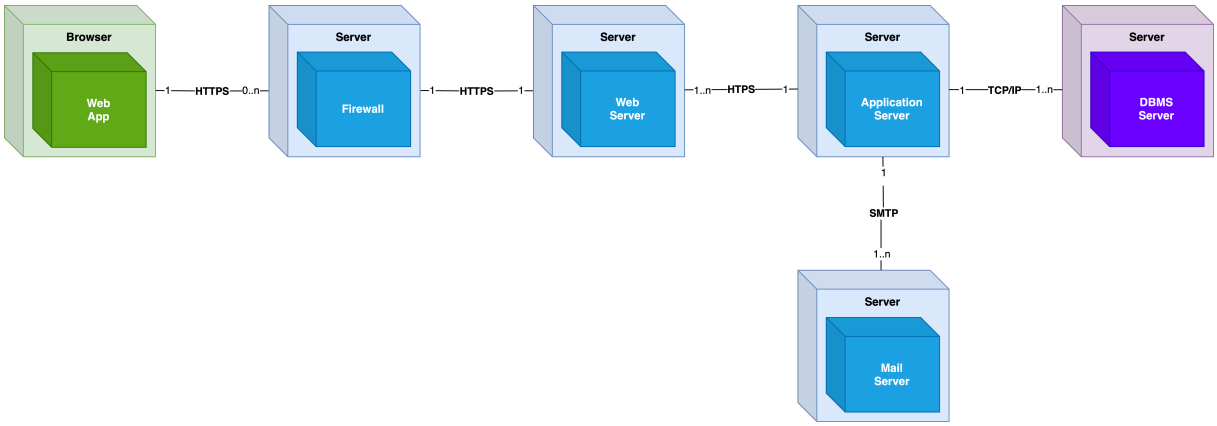

Figure 2.10: Deployment view

The connection between the firewall and the web server, and subsequently between the web server and the application server, uses HTTPS. This protocol encrypts all transmitted data, ensuring confidentiality and integrity for securing sensitive operations such as user authentication. The application server then communicates with the DBMS server using the TCP/IP protocol, providing reliable and ordered data exchange. This is suitable for database queries and responses, ensuring accurate transactions between the application and the database. Lastly, the application server interacts with the mail server to send emails using SMTP. This ensures efficient email delivery while separating email-related tasks from core ones.

A relational database management system like PostgreSQL is used for structured data storage, providing ACID compliance on a well-defined schema. As they represent unstructured data, files such as uploaded CVs are stored locally on the DBMS server's file system. Rather than storing the binary data in the database itself, the file paths are saved as references in it instead. This approach thus separates structured from unstructured data, optimizing database performance and simplifying file management.

2.4. Runtime View

The runtime view captures the dynamic behavior of the system by detailing how its components interact to perform specific operations. Using sequence diagrams, this section illustrates the communication pathways and timing relationships among components, offering insights into the system's execution flow. These runtime perspectives not only guide implementation but also serve as a reference for testing and troubleshooting.

2.4.1. User

User Logs Out

The user is logged in and on the home page.

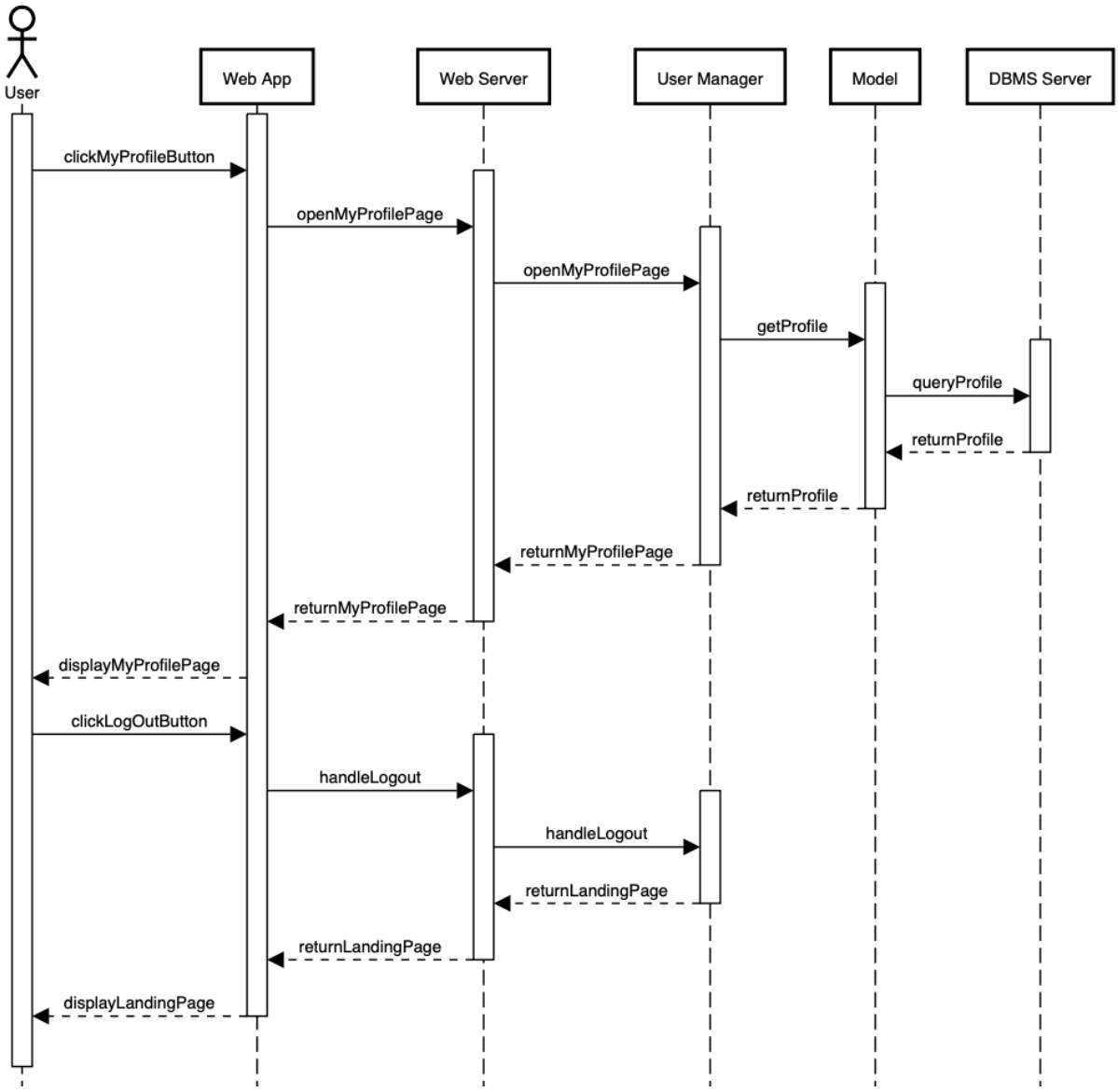

Figure 2.11: User logs out sequence diagram

2.4.2. Student

Student Signs Up

The student is not signed up.

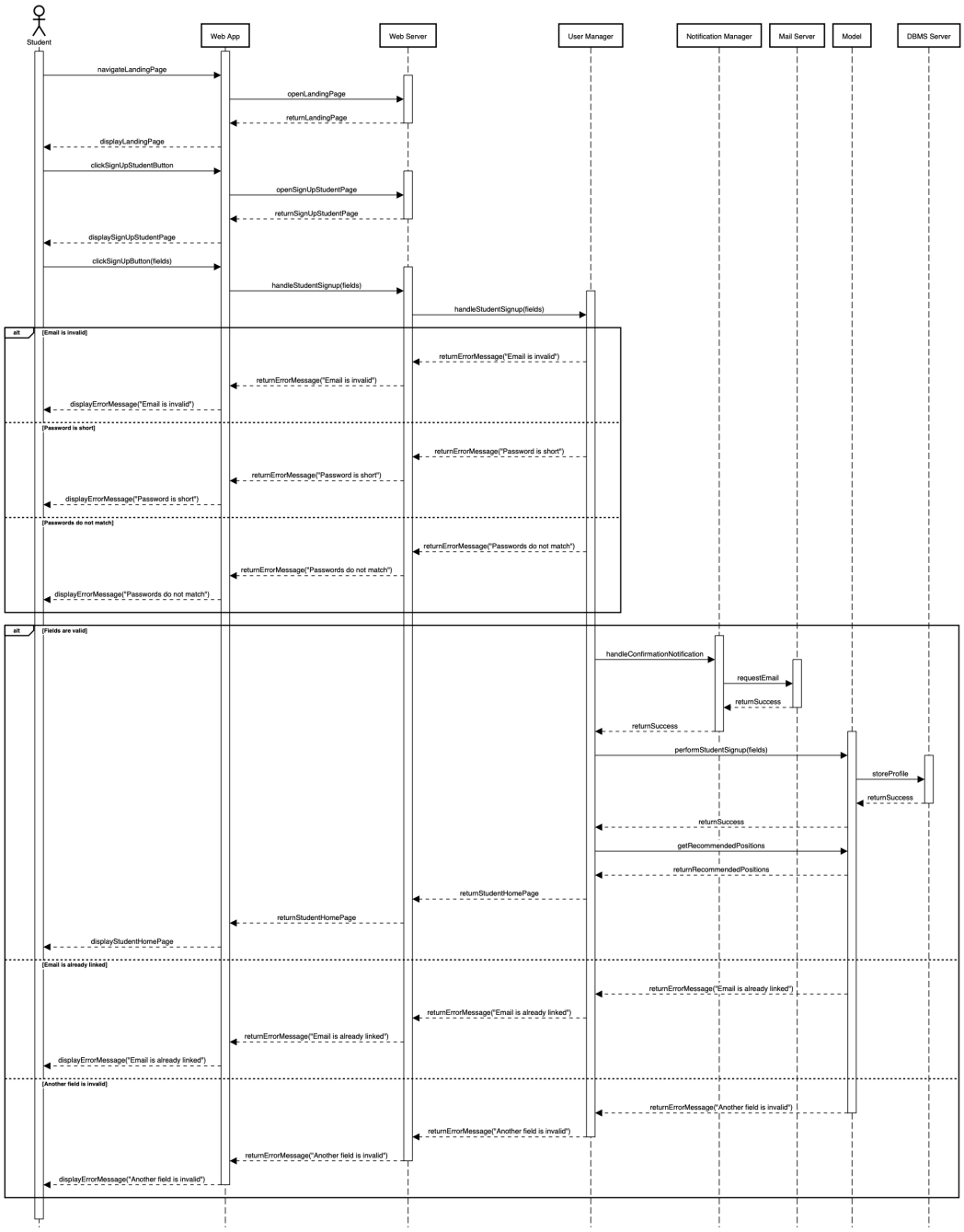

Figure 2.12: Student signs up sequence diagram

Student Logs In

The student is signed up.

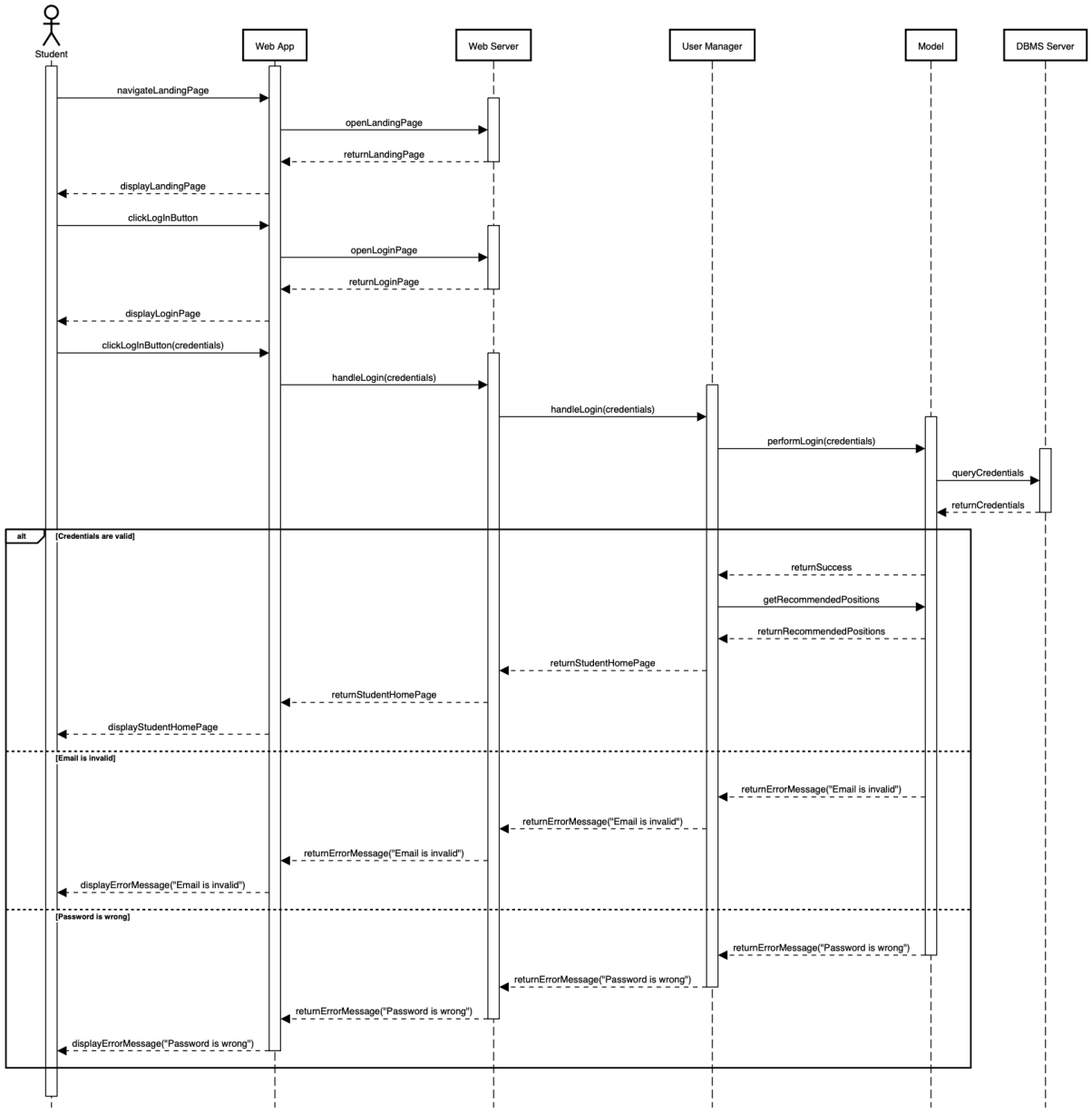

Figure 2.13: Student logs in sequence diagram

Student Accepts Recommendation

The student is logged in, has uploaded a CV, entered preferences and ticked the "Keep Me Updated" field.

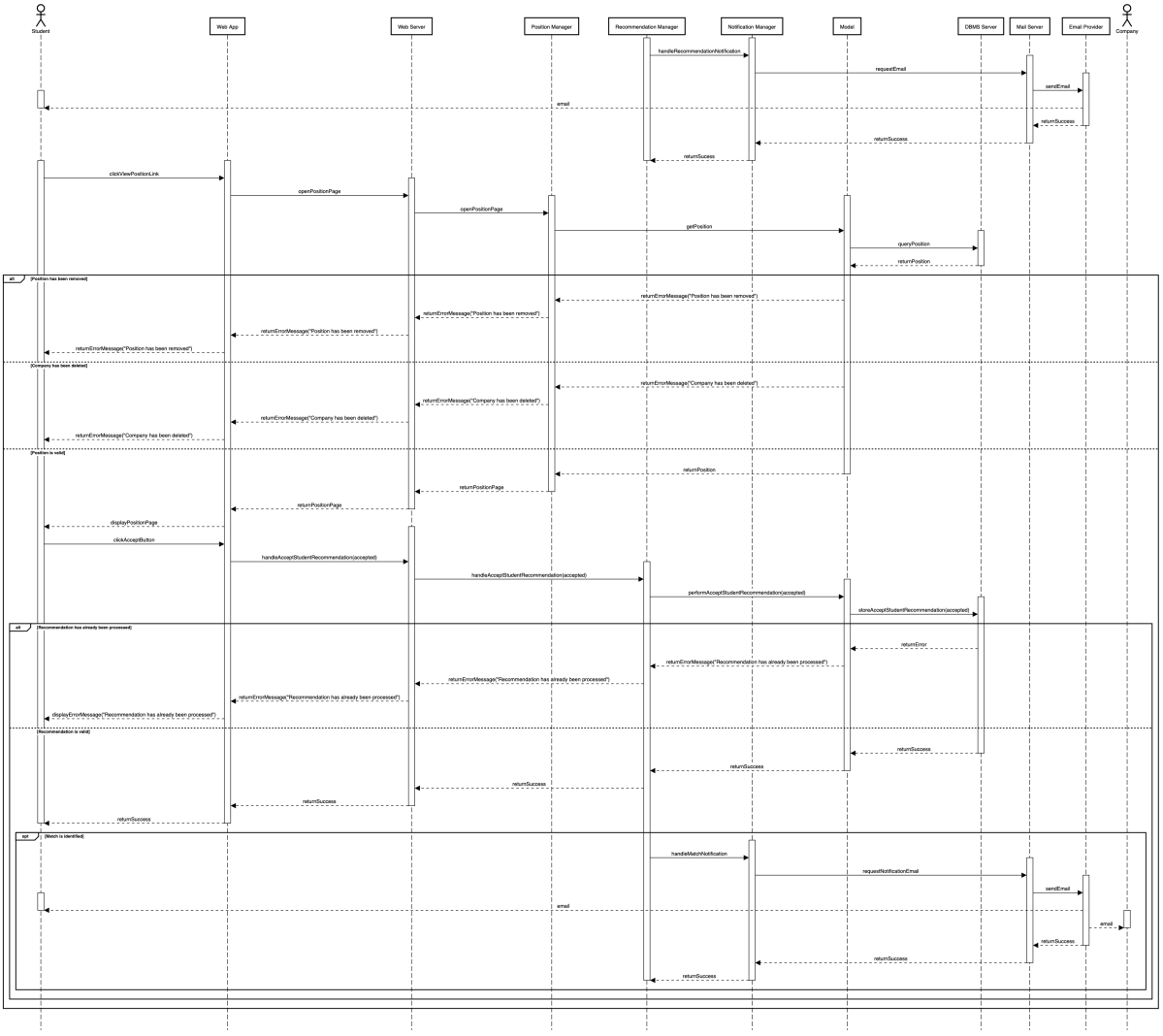

Figure 2.14: Student accepts recommendation sequence diagram

Student Accepts Interview

The student is logged in, has received a notification on a scheduled interview and he is on the home page.

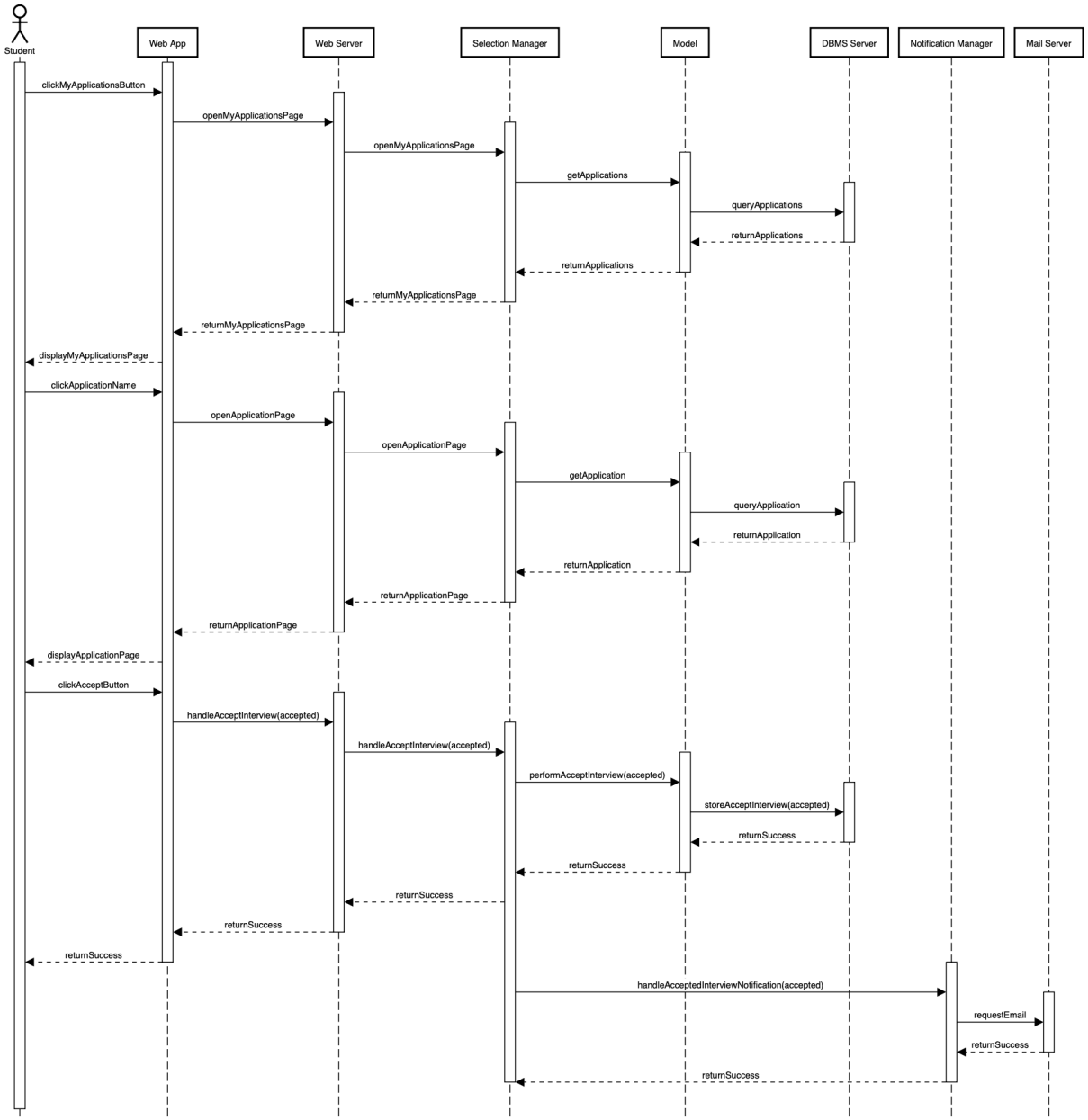

Figure 2.15: Student accepts interview sequence diagram

Student Comments Internship

The student is logged in, doing an internship and he is on the home page.

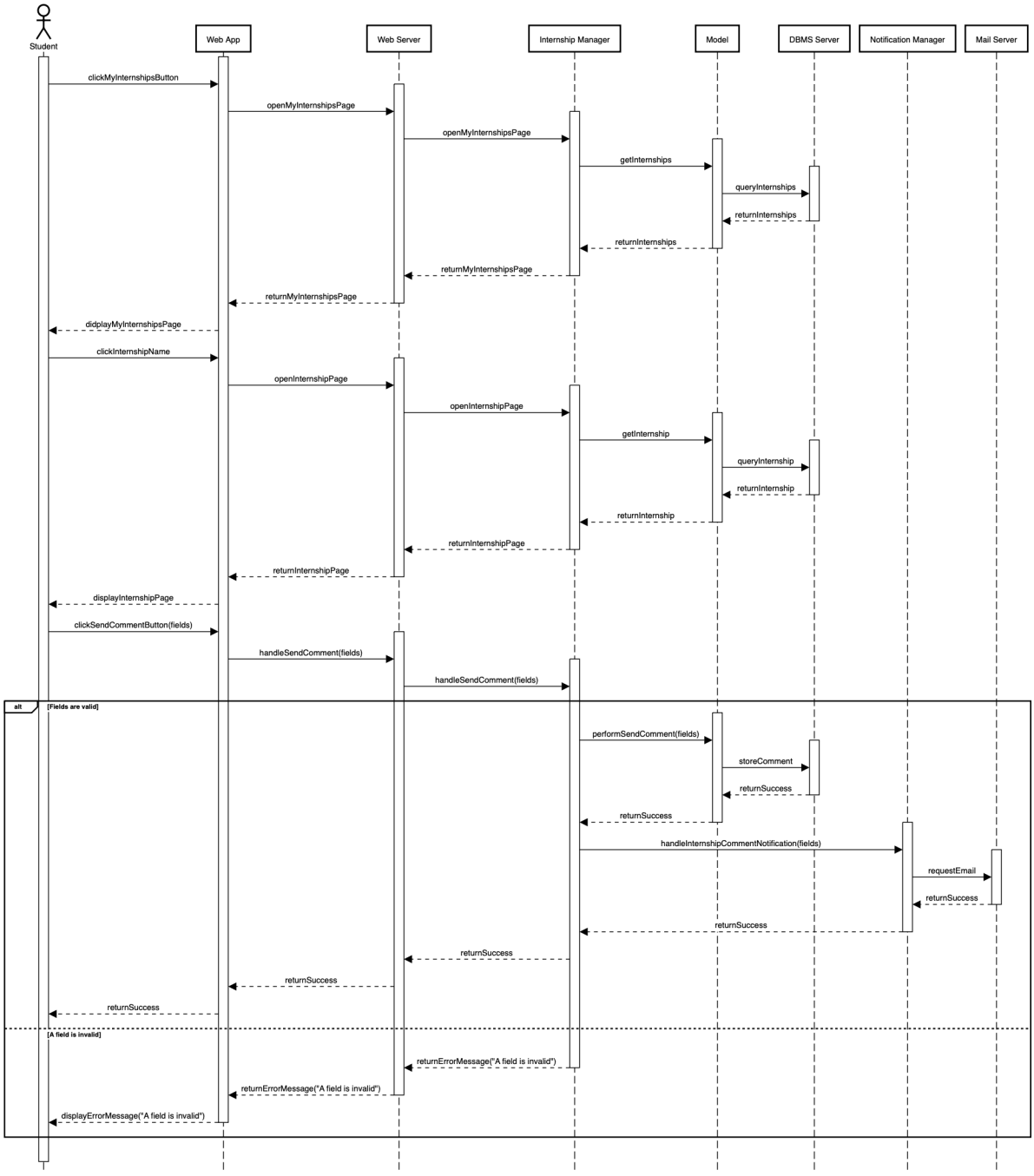

Figure 2.16: Student comments internship sequence diagram

2.4.3. Company

Company Signs Up

The company is not signed up.

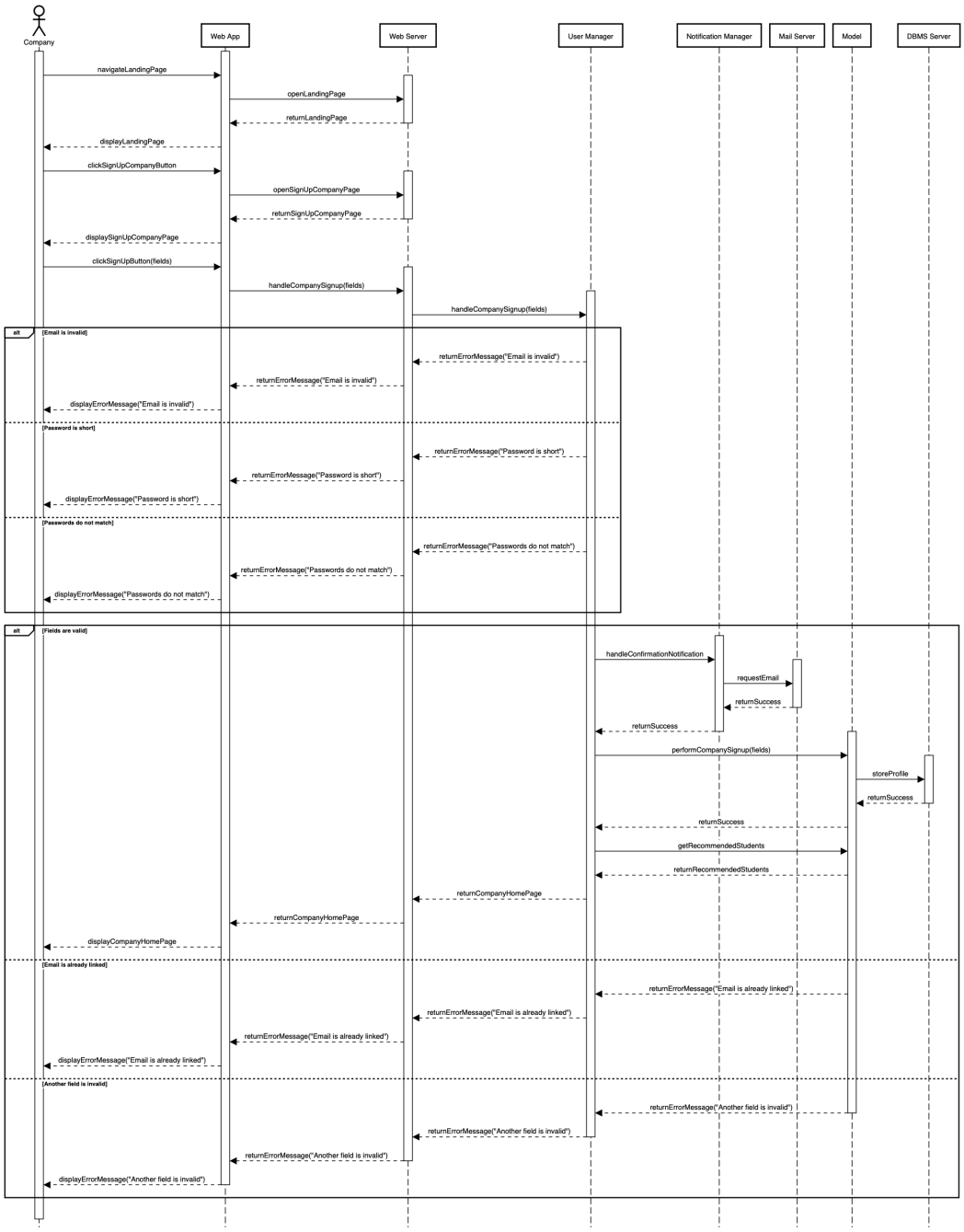

Figure 2.17: Company signs up sequence diagram

Company Logs In

The company is signed up.

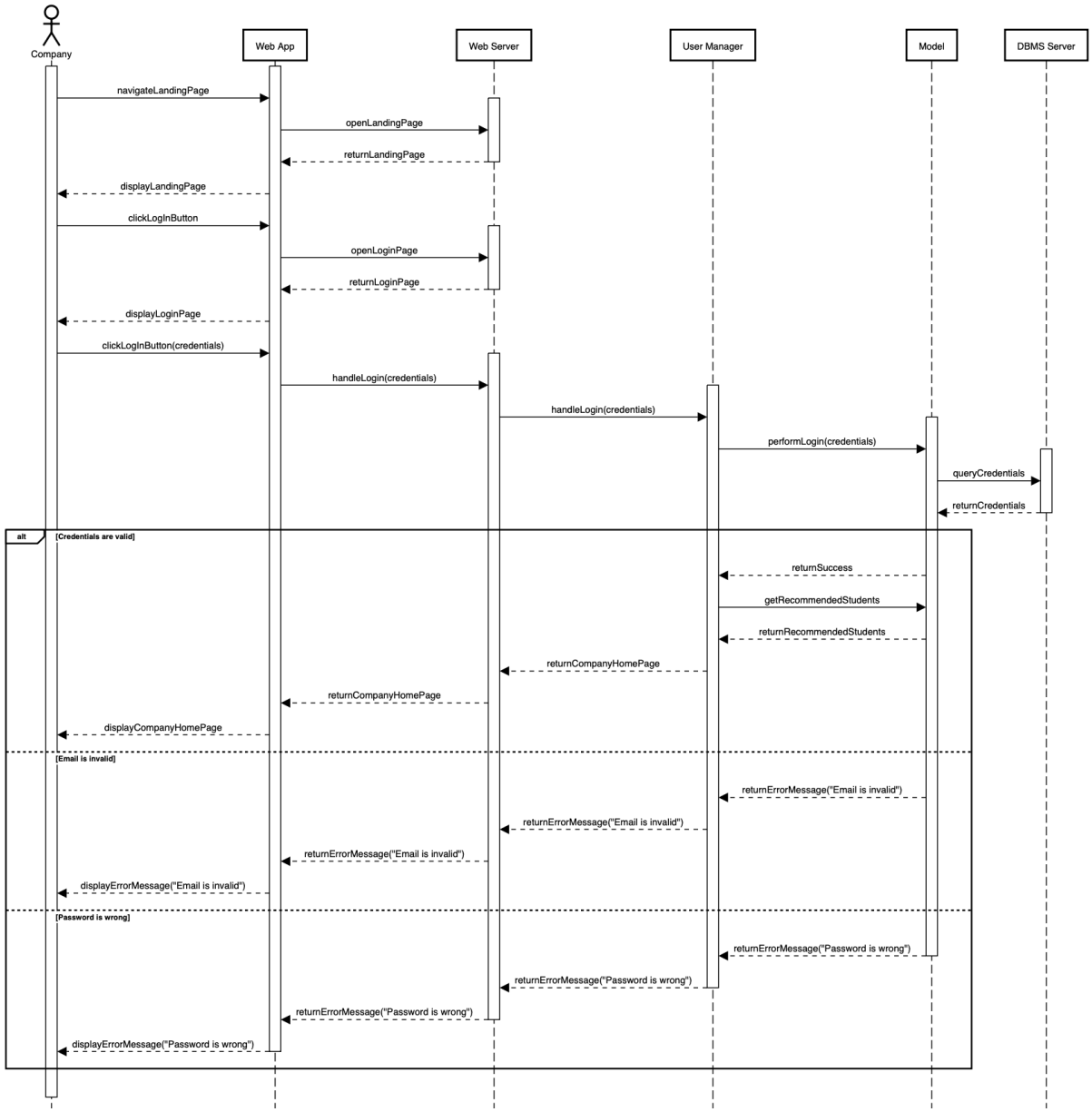

Figure 2.18: Company logs in sequence diagram

Company Accepts Recommendation

The company is logged in, has posted a position and has ticked the "Keep Me Updated" field.

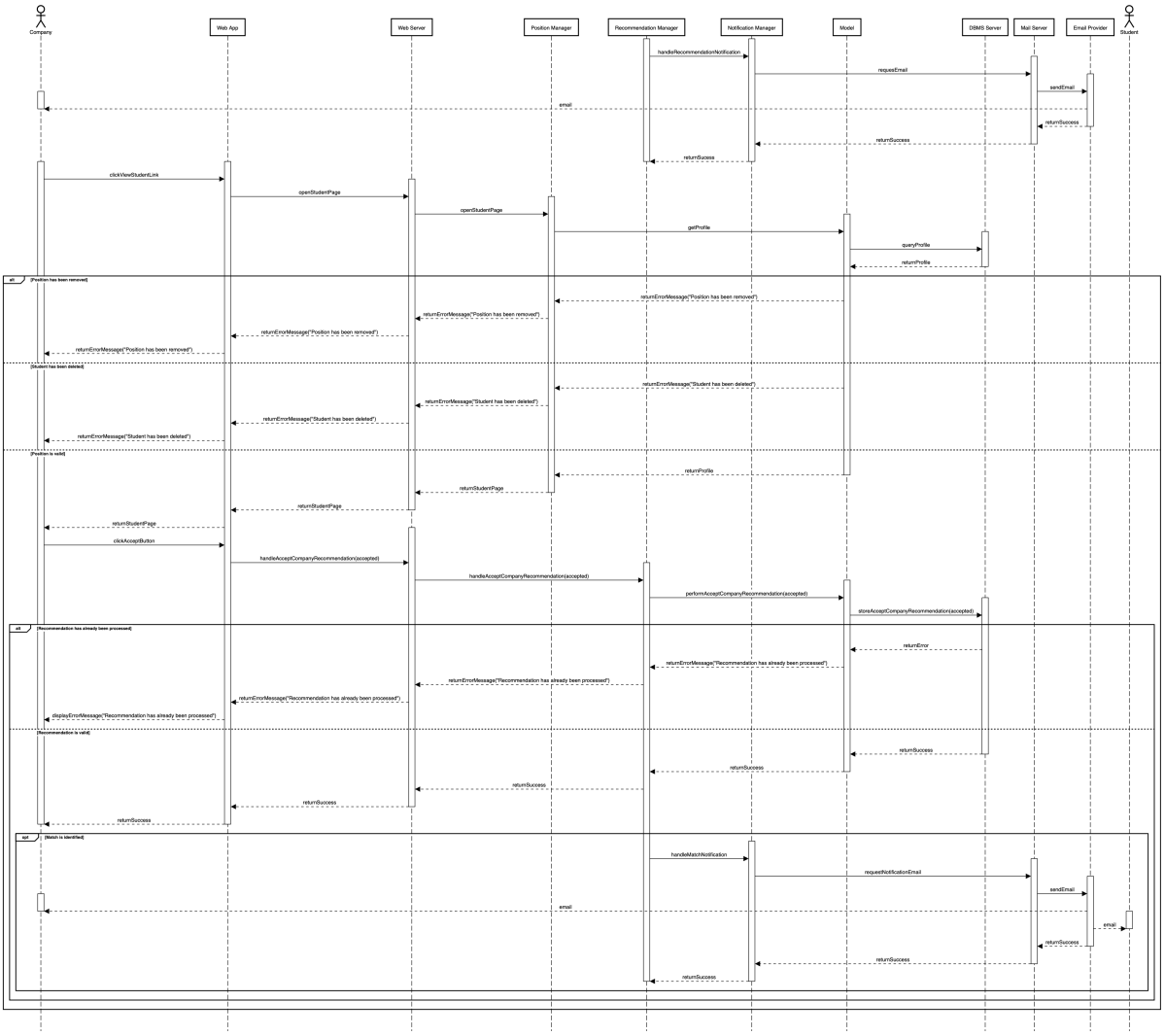

Figure 2.19: Company accepts recommendation sequence diagram

Company Schedules Interview

The company is logged in, has posted a position and it is on the home page.

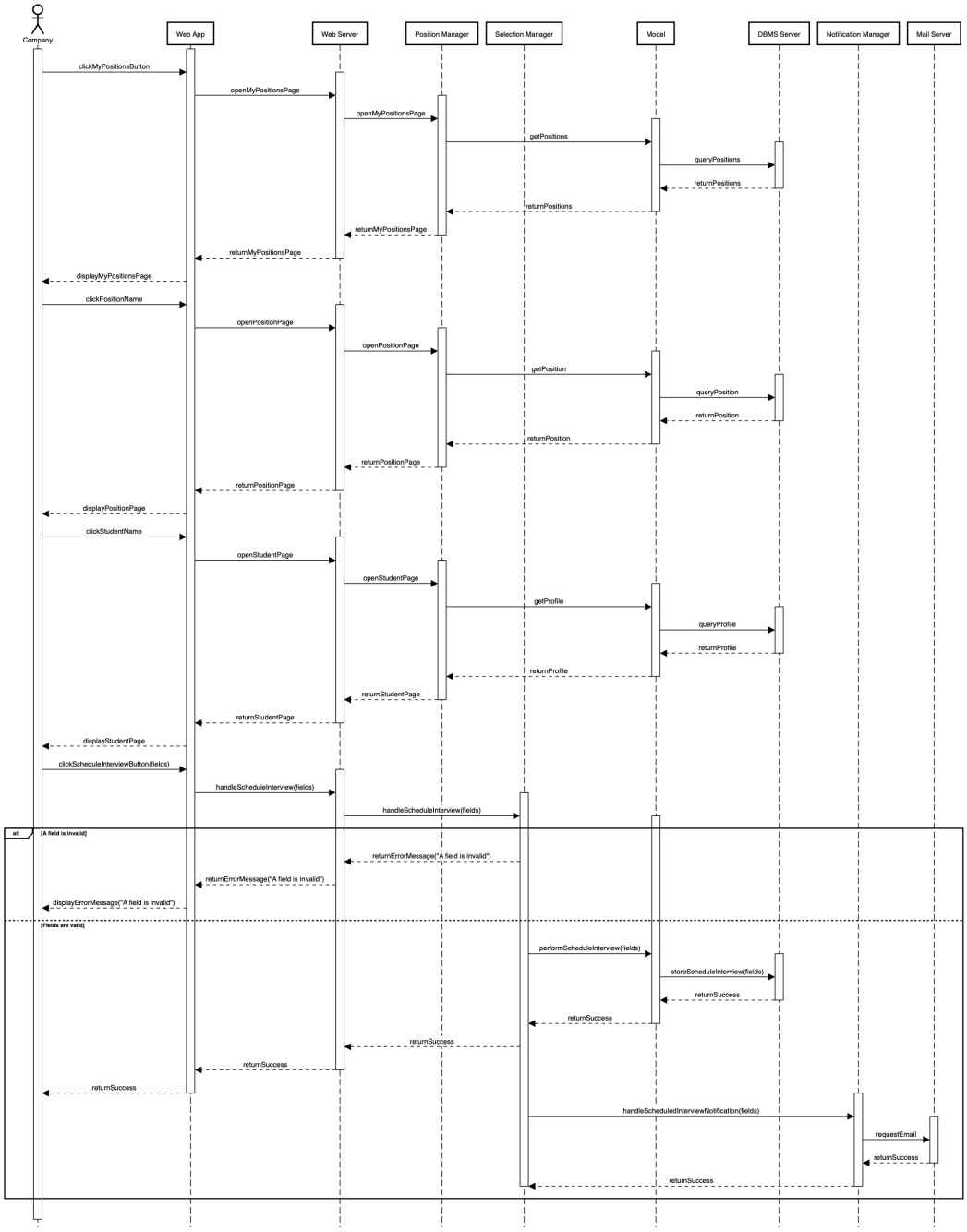

Figure 2.20: Company schedules interview sequence diagram

Company Comments Internship

The company is logged in, hosting an internship and it is on the home page.

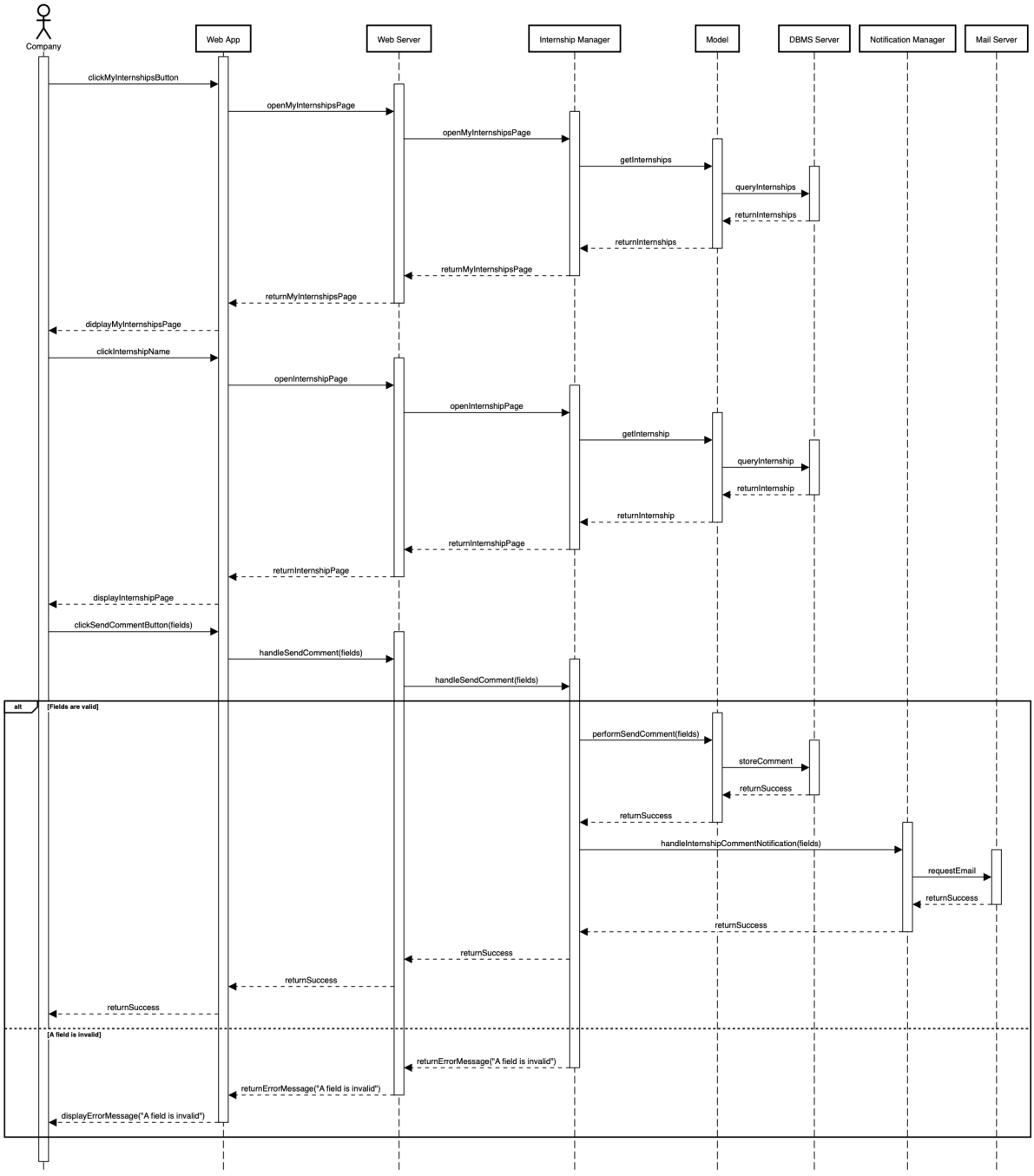

Figure 2.21: Company comments internship sequence diagram

2.4.4. University

University Signs Up

The university is not signed up.

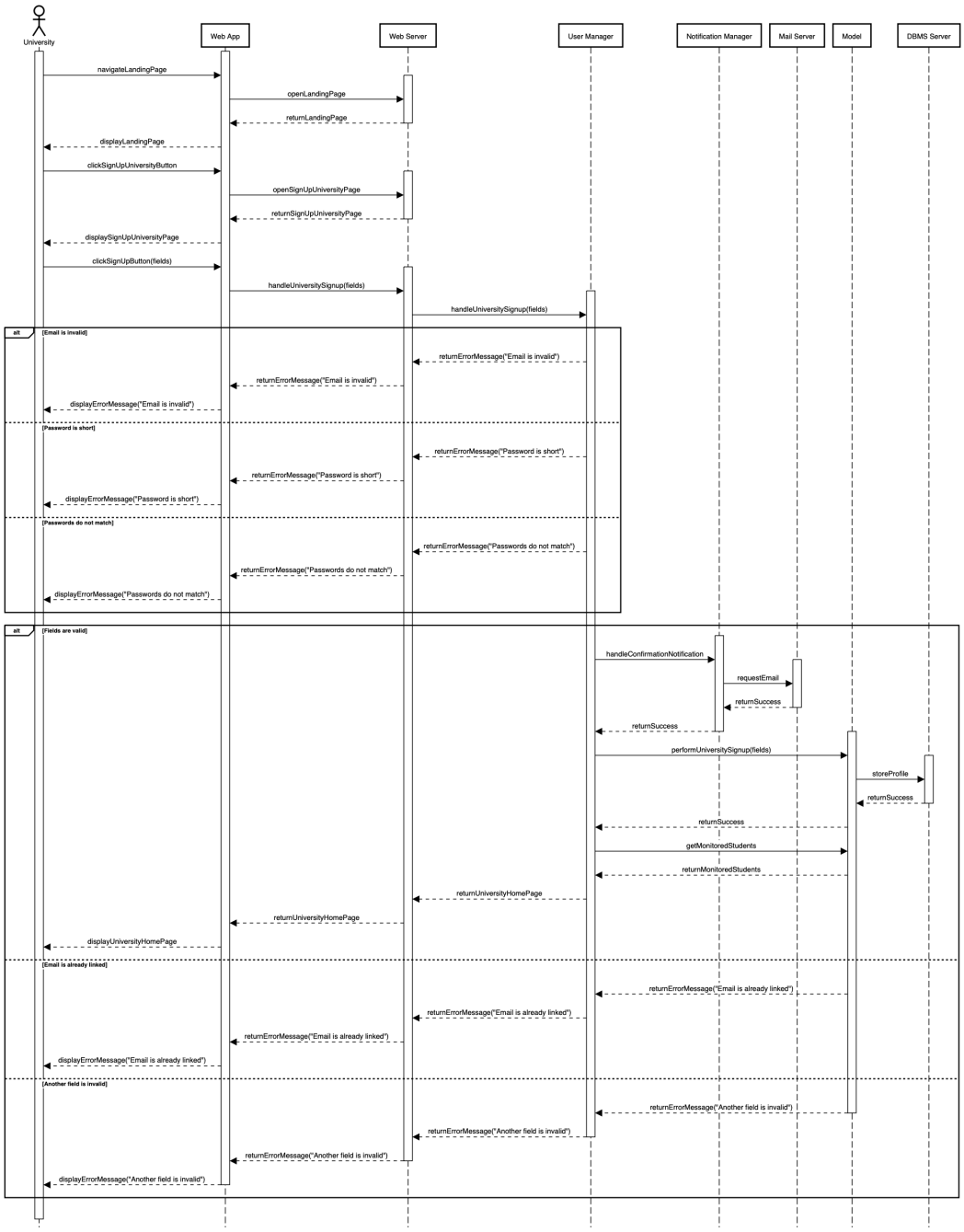

Figure 2.22: University signs up sequence diagram

University Logs In

The university is signed up.

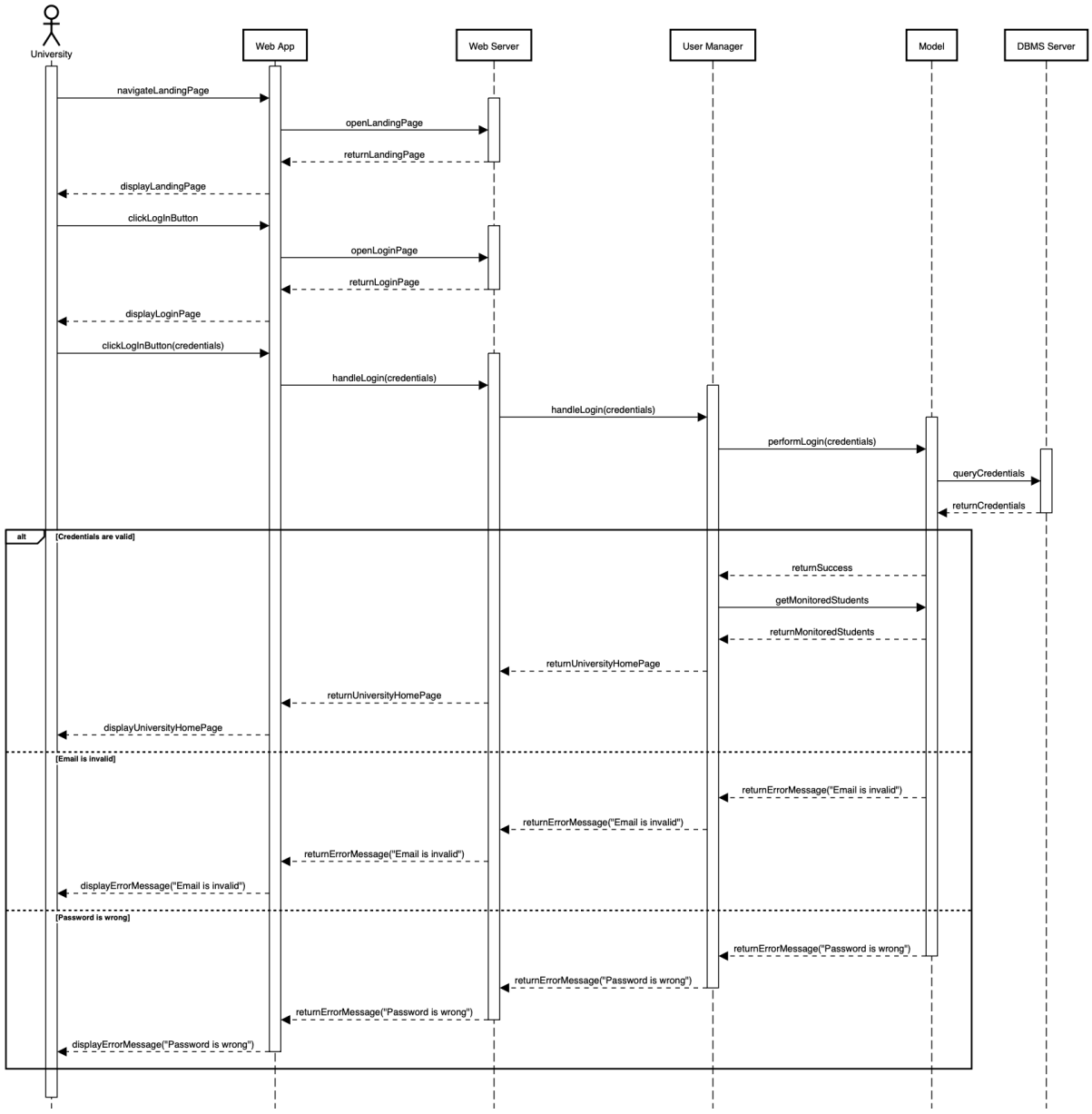

Figure 2.23: University logs in sequence diagram

University Comments Internship

The university is logged in and on the home page.

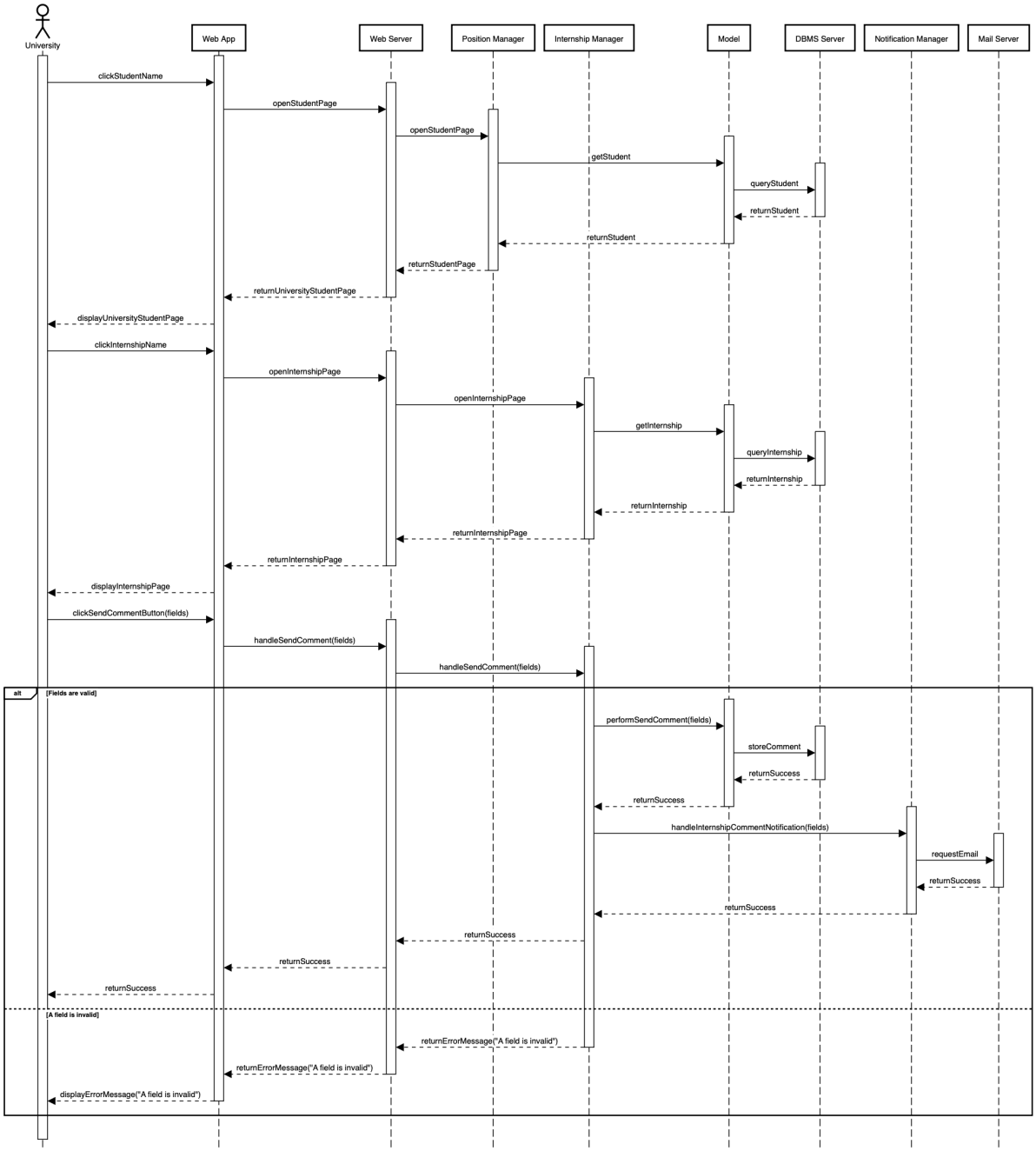

Figure 2.24: University comments internship sequence diagram

2.5. Component Interfaces

This section lists the component interfaces that define the methods provided by managers within the system to interact with other components. Each manager encapsulates a specific set of functionalities and communicates through standardized interfaces to ensure modularity. For methods requiring database access, there is a corresponding method in the model interface to handle related operations. For instance, if the user manager includes the method `handleUserSignup` to manage user signup, then the model will provide the method `performUserSignup` to validate and store the data in the database. This separation ensures a clean architecture by dividing user-facing logic and data-handling responsibilities. Note that once a user logs into Students&Companies, their email is stored in the session.

User Interface
`handleStudentSignup(String name, String surname, String email, String password, String confirmPassword, boolean keepMeUpdated, String preferences, String cv)`
`handleCompanySignup(String name, String email, String field, String password, String confirmPassword, boolean keepMeUpdated)`
`handleUniversitySignup(String name, String email, String password, String confirmPassword, boolean keepMeUpdated)`
`handleDeleteProfile()`
`handleLogin(String email, String password)`
`handleLogout()`
`handleUpdateStudentProfile(String name, String lastname, String email, String password, String confirmPassword, String preferences, String cv)`
`handleUpdateCompanyProfile(String name, String email, String field, String password, String confirmPassword)`
`handleUpdateUniversityProfile(String name, String email, String password, String confirmPassword)`
`openStudentHomePage()`
`openCompanyHomePage()`
`openUniversityHomePage()`
`openMyProfilePage()`

Table 2.1: User interface

Position Interface
`handlePostPosition(String name, String domain, String project, String tasks, String terms)`
`handleDeletePosition(int positionId)`
`handleUpdatePosition(int positionId, String domain, String project, String tasks, String terms)`
`handleSearchPosition(String keyword)`
`openMyPositionsPage()`
`openPositionPage(int positionId)`
`openStudentPage(String studentEmail)`
`openCompanyPage(String companyEmail)`

Table 2.2: Position interface

Recommendation Interface
`handleAcceptStudentRecommendation(int recommendationId, boolean accepted)`
`handleAcceptCompanyRecommendation(int recommendationId, boolean accepted)`
`handleFillOutFeedbackForm(List<String> fields)`

Table 2.3: Recommendation interface

Selection Interface
`handleApplication(int positionId)`
`handleWithdrawApplication(int applicationId)`
`handleUpdateSelectionStatus(int selectionId, String status)`
`handleScheduleInterview(int selectionId, Date date, String mode)`
`handleAcceptInterview(int interviewId, boolean accepted)`
`handleAddQuestionnaire(int selectionId, List<String> fields)`
`handleFillOutQuestionnaire(int questionnaireId, List<String> fields)`

`openMyApplicationsPage()`
`openApplicationPage(int appplicationId)`

Table 2.4: Selection interface

Internship Interface
`handleUpdateInternshipStatus(int internshipId, String status)`
`handleSendComment(int internshipId, String comment)`
`openMyInternshipsPage()`
`openInternshipPage(int internshipId)`

Table 2.5: Internship interface

Notification Interface
`handleConfirmationNotification(String email)`
`handleStudentApplicationNotification(String studentEmail, int applicationId)`
`handleCompanyApplicationNotification(String companyEmail, int applicationId)`
`handleRecommendationNotification(String studentEmail, String companyEmail, int recommendationId)`
`handleMatchNotification(String studentEmail, String companyEmail, int recommendationId)`
`handleSelectionStatusNotification(String studentEmail, String companyEmail, int selectionId)`
`handleScheduledInterviewNotification(String studentEmail, int interviewId)`
`handleAcceptedInterviewNotification(String companyEmail, int interviewId, boolean accepted)`
`handleAddedQuestionnaireNotification(String studentEmail, int questionnaireId)`
`handleFilledOutQuestionnaireNotification(String companyEmail, int questionnaireId)`

handleInternshipStatusNotification(String studentEmail, String companyEmail, String universityEmail, int internshipId, String status)
handleInternshipCommentNotification(String studentEmail, String companyEmail, String universityEmail, int internshipId, String authorEmail, String comment)

Table 2.6: Notification interface

Model Interface
performStudentSignup(String name, String surname, String email, String password, boolean keepMeUpdated, String preferences, String cv)
performCompanySignup(String name, String email, String field, String password, boolean keepMeUpdated)
performUniversitySignup(String name, String email, String password, boolean keepMeUpdated)
performDeleteProfile(String email)
performLogin(String email, String password)
performUpdateStudentProfile(String name, String lastname, String email, String password, String preference, String cv)
performUpdateCompanyProfile(String name, String email, String field, String password)
performUpdateUniversityProfile(String name, String email, String password)
getRecommendedPositions(String studentEmail)
getRecommendedStudents(String companyEmail)
getMonitoredStudents(String universityEmail)
getProfile(String email)
performPostPosition(String companyEmail, String name, String domain, String project, String tasks, String terms)
performDeletePosition(int positionId)
performUpdatePosition(int positionId, String name, String domain, String project, String tasks, String terms)

performSearchPosition(String keyword)
getPositions(String companyEmail)
getPosition(positionId)
performAcceptStudentRecommendation(int recommendationId, boolean accepted)
performAcceptCompanyRecommendation(int recommendationId, boolean accepted)
performFillOutFeedbackForm(String email, List<String> fields)
performApplication(String studentEmail, int positionId)
performWithdrawApplication(int applicationId)
performUpdateSelectionStatus(String email, int selectionId, String status)
performScheduleInterview(int selectionId, String description, Date date, String mode)
performAcceptInterview(int interviewId, boolean accepted)
performAddQuestionnaire(int selectionId, List<String> fields)
performFillOutQuestionnaire(int questionnaireId, List<String> fields)
getApplications(String studentEmail)
getApplication(int applicationId)
performUpdateInternshipStatus(String email, int internshipId, String status)
performSendComment(String email, int internshipId, String comment)
getInternships(String email)
getInternship(int internshipId)

Table 2.7: Model interface

2.6. Architectural Styles and Patterns

This section examines the architectural styles adopted in Students&Companies, explaining their general principles and how they guide implementation decisions. Understanding these foundational patterns is crucial for developers working on the system, as they influence everything from component interaction to code organization.

Client-Server Architecture

The system's foundation rests on a client-server architecture, which separates components into service requesters, called clients, and service providers, called servers. This separation enables the user interface to evolve independently from data processing and storage, while centralizing resource management on the server. Developers can thus focus on their domain, either crafting responsive user experiences or implementing robust business logic.

Modern applications refine this pattern through tiers, which are logical groupings of related functionality. A three-tier architecture divides the application into three distinct layers: a presentation tier that handles the user interface, an application tier that processes business logic and a data tier that manages storage. This separation gives developers clear boundaries for their work and lets teams works independently, reducing development bottlenecks and simplifying maintenance.

REST Architecture

REST (Representational State Transfer) guides how these tiers communicate over the web. Originally conceived for document transfer, REST has evolved into a comprehensive style for data exchange. Its core principle of statelessness means that servers maintain no information about past requests. Instead, each client request must contain all necessary context. This architectural choice significantly impacts development: backend developers do not need to manage session states, system administrators benefit from straightforward load balancing since any server can handle any request and cache developers can optimize performance by focusing on request parameters rather than server state.

MVC Pattern

The Model-View-Controller (MVC) pattern organizes software into three interconnected components, each with distinct responsibilities. The model represents the core data structure, encapsulating the database, while the view handles user interaction and presentation. Controllers act as intermediaries, processing user requests to run business logic and coordinate workflows. This separation of concerns enhances scalability, modularity and maintainability, allowing developers to focus on specific aspects of the system without impacting others, ultimately fostering efficient collaboration among teams.

2.7. Other Design Decisions

While architectural styles and patterns provide the system's foundation, many other critical design decisions shape its implementation. This section outlines these choices, focusing on practical aspects that developers must consider when implementing the system.

Deployment Approach

The deployment strategy embraces horizontal scaling, running multiple application server instances rather than scaling up individual servers. This approach requires developers to write stateless code that works across instances, but provides significant benefits: the

system continues functioning even if some instances fail, resource allocation becomes more flexible as administrators can add or remove instances based on demand and load balancing becomes straightforward since any instance can handle any request.

Security Implementation

Security permeates every aspect of the design through multiple complementary layers: all client-server communication uses HTTPS encryption to protect data in transit, sensitive information receives additional encryption in the database and session management relies on time-limited tokens rather than storing session data on the server.

Data Management

Data management emphasizes both integrity and performance: the database implements transaction management to maintain consistency even during concurrent operations, constraint enforcement catches validity issues early and performance optimization through strategic indexing and caching improves response times.

3 | User Interface Design

This chapter presents the user interface design of Students&Companies, focusing on its structure and functionality to cater to students, companies and universities. The objective of the design is to create a seamless, intuitive and efficient user experience. To achieve this, it adheres to core principles that enhance simplicity, accessibility and consistency throughout the platform.

Ensuring consistency across the system, the design employs uniform fonts, colors and layouts, allowing users to navigate with ease and confidence. Accessibility is a key tenet, with the interface accommodating users with disabilities and ensuring responsiveness across devices by adhering to Web Content Accessibility Guidelines (WCAG). Finally, simplicity drives the design process, prioritizing logical workflows that minimize user effort and facilitate task completion.

3.1. Flow Diagram

The flow diagram below serves as a map of the platform's pages, illustrating their interconnections. It provides a unified view to highlight overlaps and parallelisms across the functionalities for students, companies and universities. This approach underscores the distinct aspects of the navigation while emphasizing its coherence.

All primary pages of the platform are represented in this diagram, providing an overview of the core workflows. To maintain clarity, some finer details, such as the profile and questionnaire editors, interview page, feedback and comment forms, have been omitted but are mentioned in the requirements analysis and specification document [1]. The pages highlighted in the diagram will be discussed in detail in the next section as they have been chosen as representative examples that showcase both the functionalities and the needs of each user category.

3.2. Selected Pages

The following section focuses on a selection of key pages within the platform.

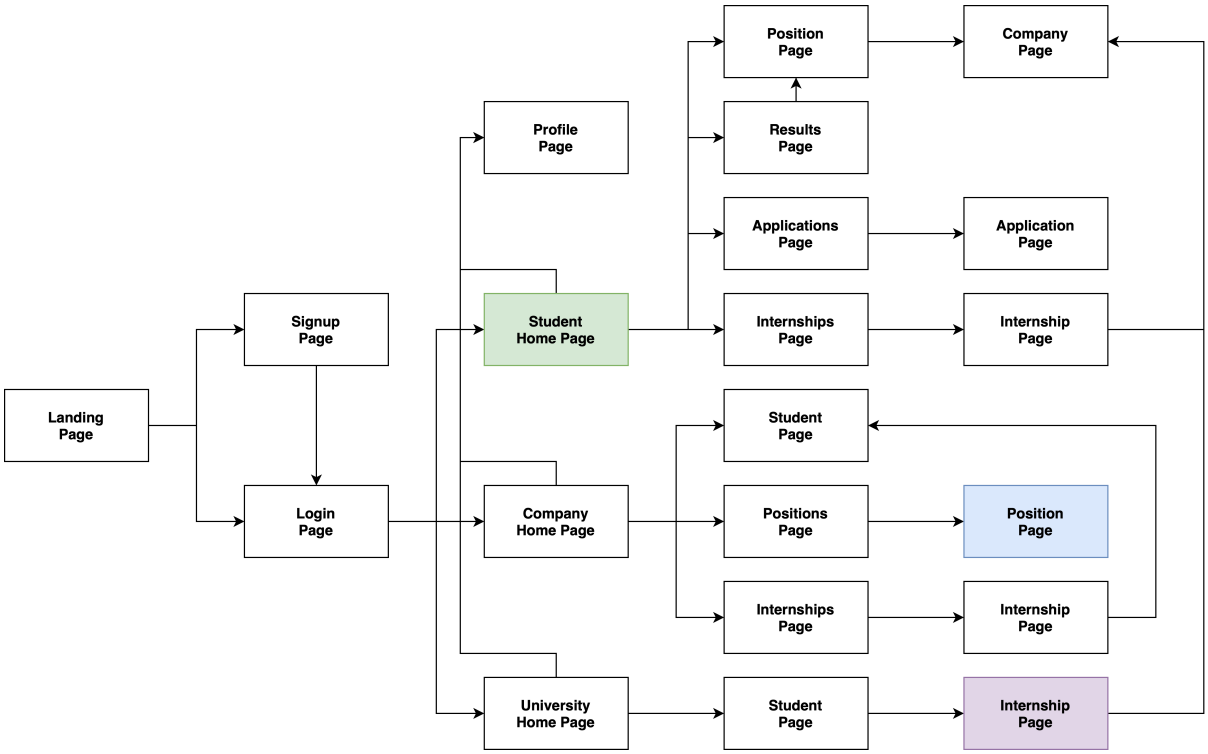

Figure 3.1: Flow diagram

3.2.1. Student Home Page

The student home page serves as the central hub for university students, designed to help them discover and apply for internship positions. It features a header that includes links to the pages outlined in the flow diagram: the home, applications, internships and profile pages. Note that this header is consistent across all of the pages, ensuring seamless navigation. Similarly, the footer contains standard links such as privacy policy, terms of service and contact information, maintaining uniformity throughout the platform.

The main body of the home page is focused on enabling students to search and explore positions. It includes a search bar with filters for location, job type and field, allowing users to refine their results. Below it, a list of recommendations is displayed. Each position includes essential details such as location, job type and field, with options to expand for more information. Students can view an expanded description of the position, along with clear buttons to either accept or decline the recommendation.

3.2.2. Company Position Page

The company position page maintains consistency with the platform's design through a familiar header and footer. The body contains buttons for adding a questionnaire and scheduling an interview, with the date and time of the upcoming interview prominently displayed. Below, it lists the questionnaires filled out by the student, which can be expanded to review answers. Based on his performance, the company can either select or reject the candidate using the respective buttons, streamlining the selection process.

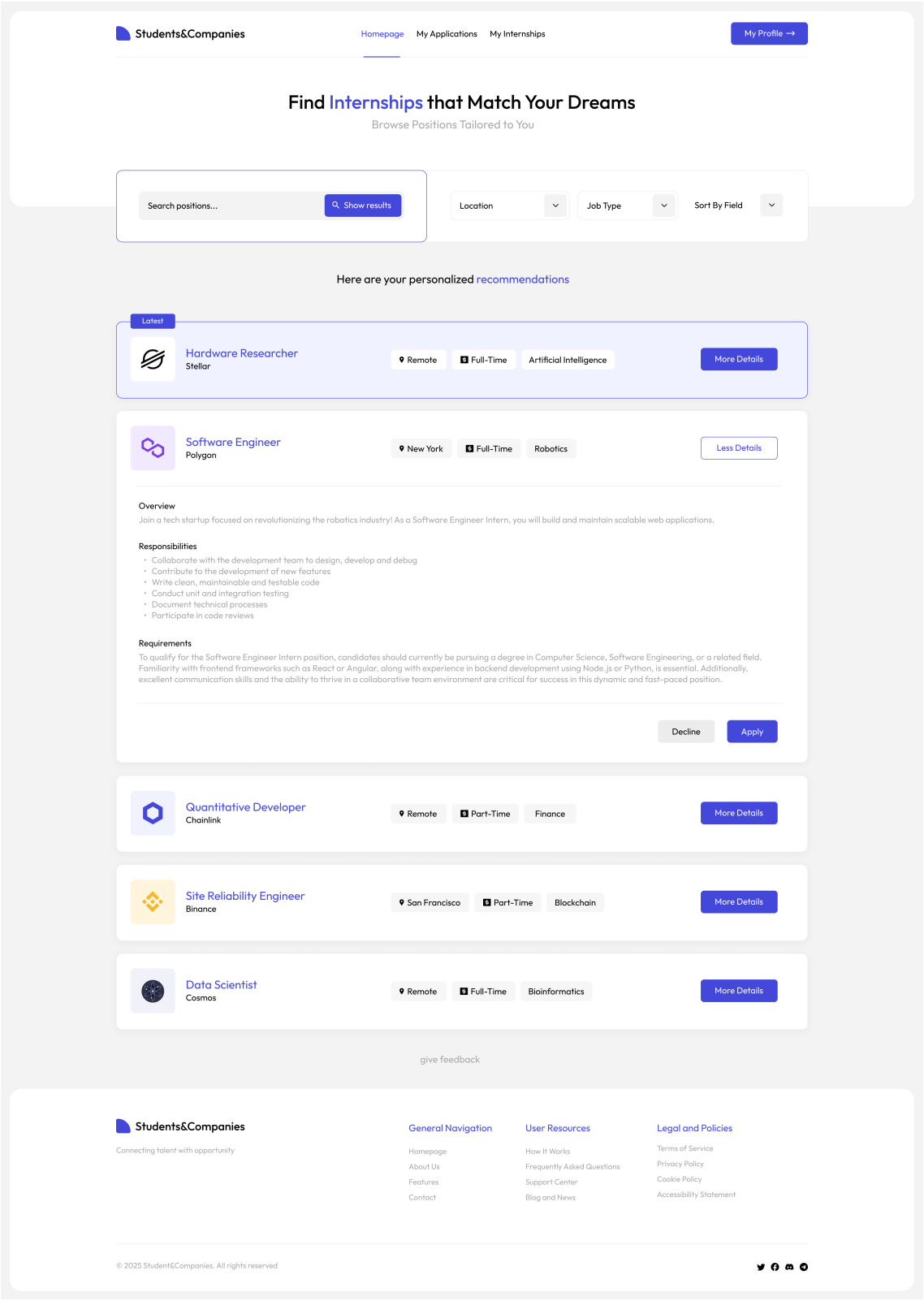

Figure 3.2: Student home page

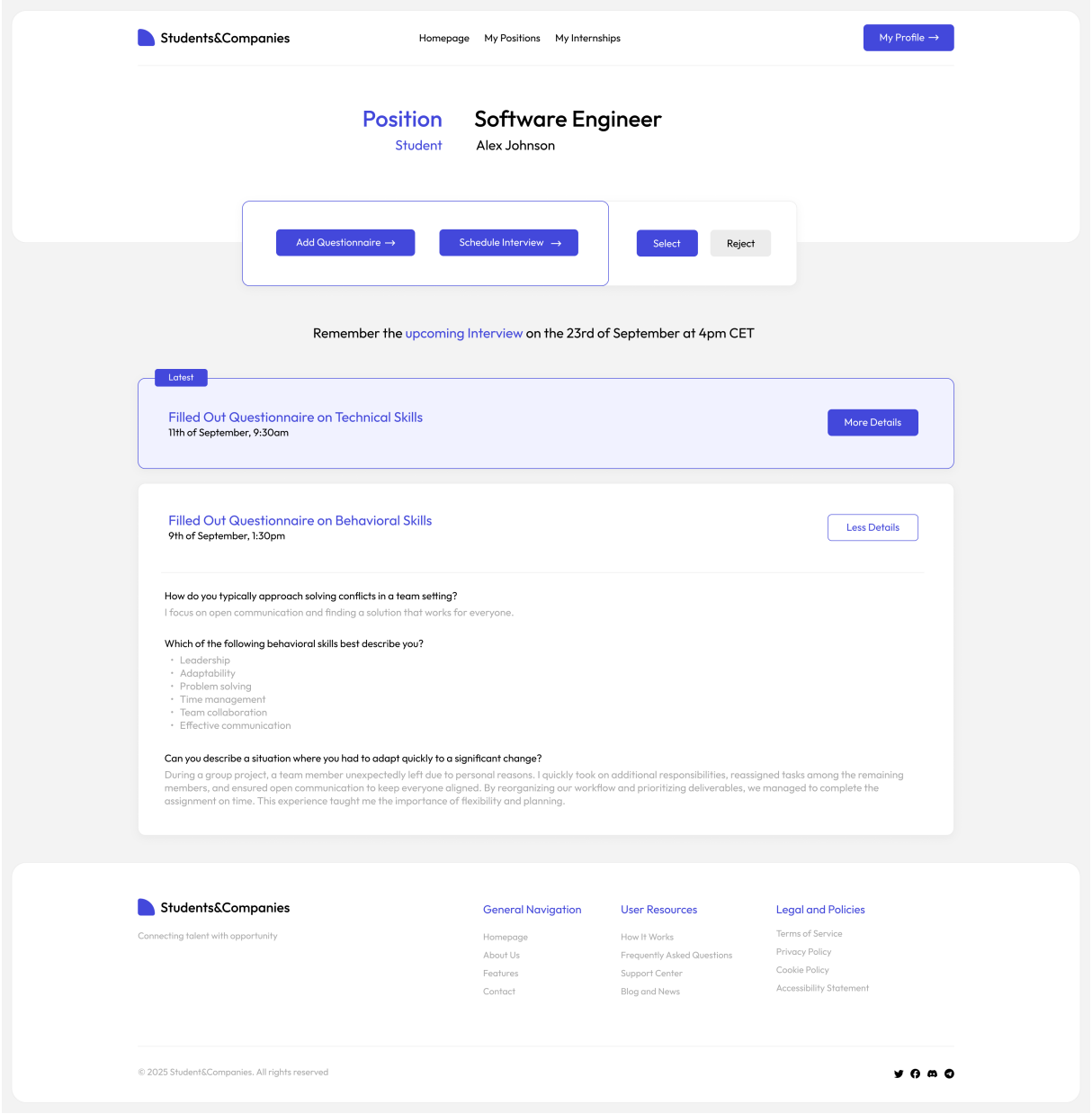

Figure 3.3: Company position page

3.2.3. University Internship Page

The university internship page ensures consistency with the platform through a standardized header and footer design. The body allows universities to view comments on the internship or write them to report progress or address concerns. These are displayed in an organized list, where each entry can be expanded to review the full content. This interface facilitates communication between students, companies and universities, ensuring internships remain productive and beneficial for all parties involved.

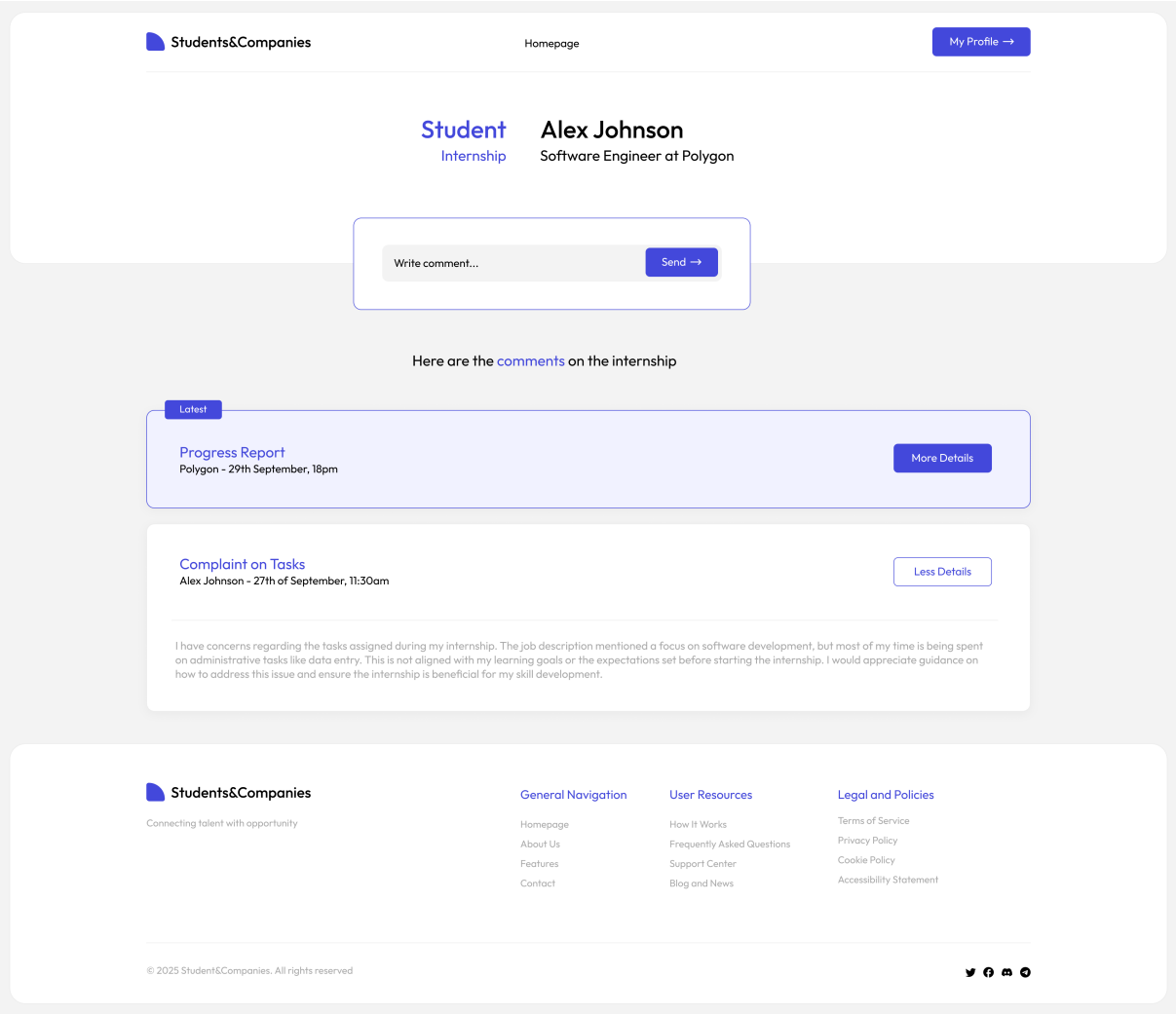

Figure 3.4: University internship page

4 | Requirements Traceability

This chapter illustrates how the requirements specified in the requirements analysis and specification document [1] are assigned to the submanagers. The mapping employs three-letter acronyms for clarity and consistency; for example, User Signup Submanager is abbreviated as USS. The table below presents the correspondence between the requirements and their respective submanagers.

ID	Submanagers
R1	USS, NFS, NDS
R2	UMS, NFS, NDS
R3	ULS
R4	ULS
R5	UMS
R6	UMS
R7	UMS
R8	UMS
R9	PSS
R10	PSS
R11	PSS
R12	SAS, NFS, NDS
R13	SAS
R14	RMS
R15	RMS
R16	RFS
R17	SAS
R18	SAS
R19	SQS

R20	SIS
R21	SIS
R22	SAS
R23	IPS
R24	IPS
R25	ICS
R26	ICS
R27	USS, NFS, NDS
R28	UMS, NFS, NDS
R29	ULS
R30	ULS
R31	UMS
R32	PPS
R33	PMS
R34	PMS
R35	PMS
R36	PSS
R37	RMS
R38	RMS
R39	RFS
R40	SAS
R41	SAS
R42	SQS
R43	SQS
R44	SIS
R45	SAS
R46	IPS
R47	IPS
R48	ICS

R49	ICS
R50	USS, NFS, NDS
R51	UMS, NFS, NDS
R52	ULS
R53	ULS
R54	PSS
R55	IPS
R56	PSS
R57	IPS
R58	IPS
R59	ICS
R60	ICS
R61	RMS, NFS, NDS
R62	RMS, NFS, NDS
R63	RMS, NFS, NDS
R64	RMS, NFS, NDS
R65	SAS, NFS, NDS
R66	SAS, NFS, NDS
R67	SQS, NFS, NDS
R68	SQS, NFS, NDS
R69	SIS, NFS, NDS
R70	SIS, NFS, NDS
R71	SIS, NFS, NDS
R72	SAS, NFS, NDS
R73	IPS, NFS, NDS
R74	IPS, NFS, NDS
R75	IPS, NFS, NDS
R76	ICS, NFS, NDS
R77	ICS, NFS, NDS

R78	ICS, NFS, NDS

Table 4.1: Requirements traceability

5 | Implementation, Integration and Test Plan

This chapter dives deep into how Students&Companies is implemented, integrated and tested. The chosen approach follows a bottom-up strategy. It begins with developing and testing foundational elements before integrating the rest of the system. This gradual process focuses on the independent development of basic modules, which are tested in isolation before being integrated into the larger system. This independence eliminates the need to consider the entire system upfront, making it easier and faster to address issues and apply changes. A key factor of this method is the use of drivers, which are tests that simulate the inputs essential for validating individual modules.

5.1. Test Plan

Following the bottom-up strategy, a layered testing plan is employed to ensure system reliability. Unit tests use drivers to simulate inputs for individual submanagers, verifying their functionality in isolation. As modules are progressively integrated, integration tests verify the interactions between components, such as database queries, ensuring that combined functionalities work as expected. Once a manager is fully integrated, functional tests are performed to guarantee the alignment with the functionalities, goals and constraints outlined in the requirements analysis and specification document [1]. To assess the system's ability to handle real-world demands, load testing evaluates the platform's performance under normal and peak traffic conditions, while performance testing measures responsiveness, ensuring the system meets the required standards for speed.

5.2. Implementation and Integration

The plan begins with the development and testing of base components, specifically the foundational submanagers within each manager. Once these base submanagers are implemented and tested, the remaining ones for the same manager are integrated. To ensure clarity, when all submanagers of a manager are integrated, they are represented in the diagrams through their respective manager component.

The first step is to implement and test the model.

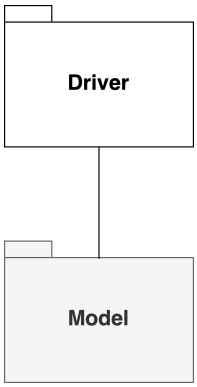

Figure 5.1: Implementation step 1

The development proceeds to step two with the user signup and login submanagers.

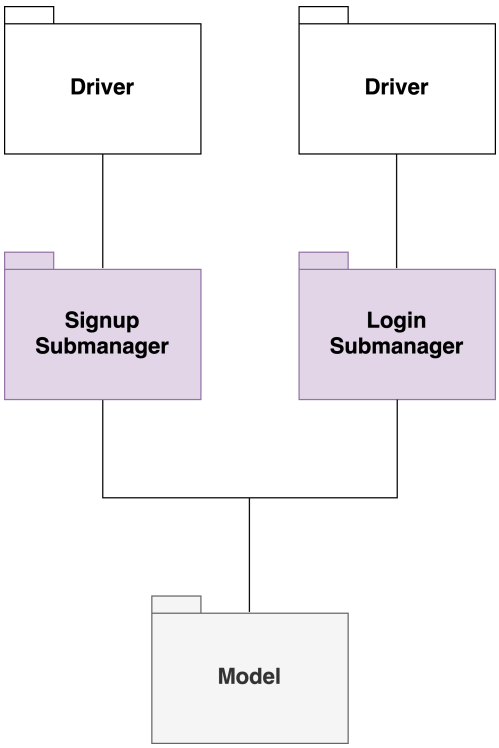

Figure 5.2: Implementation step 2

The user management submanager is integrated in the third step.

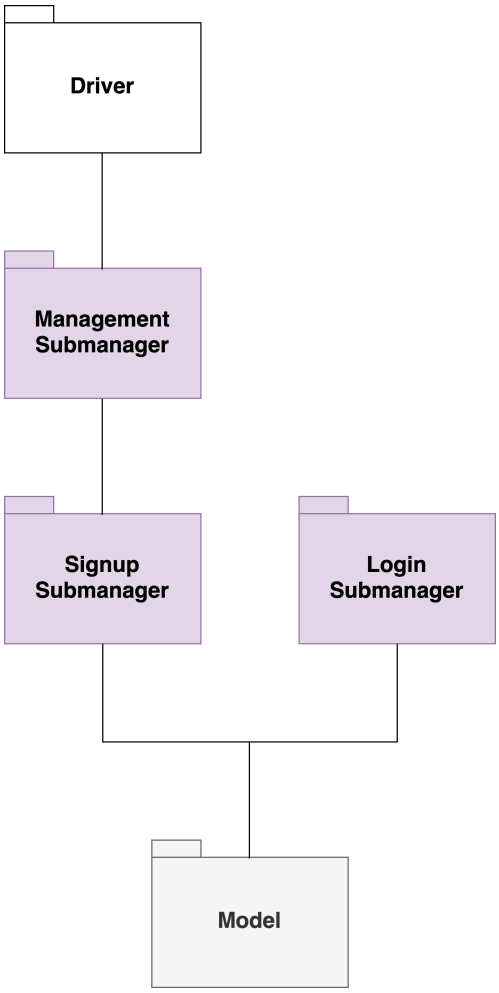

Figure 5.3: Implementation step 3

Step four applies the same pattern to the position posting and search submanagers.

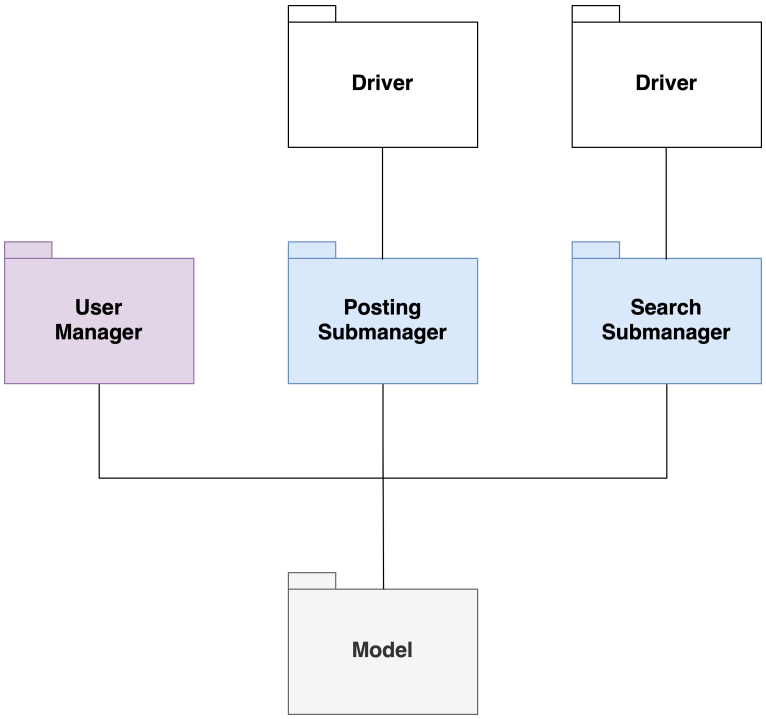

Figure 5.4: Implementation step 4

The position management submanager is integrated in the fifth step.

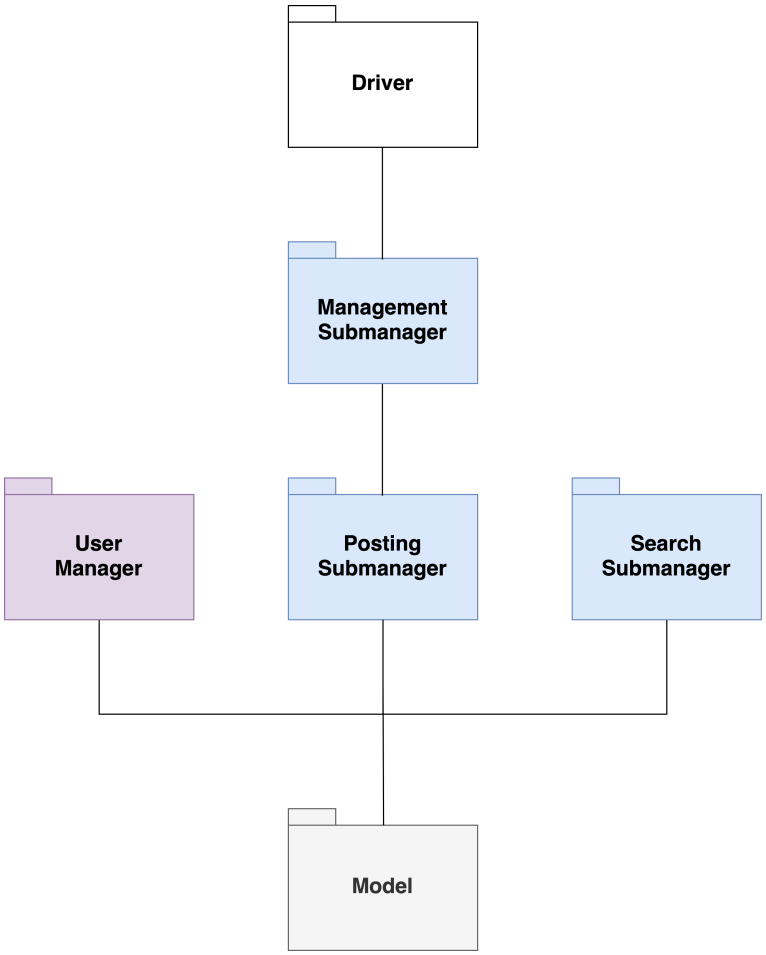

Figure 5.5: Implementation step 5

The development proceeds to step six with the recommendation submanagers.

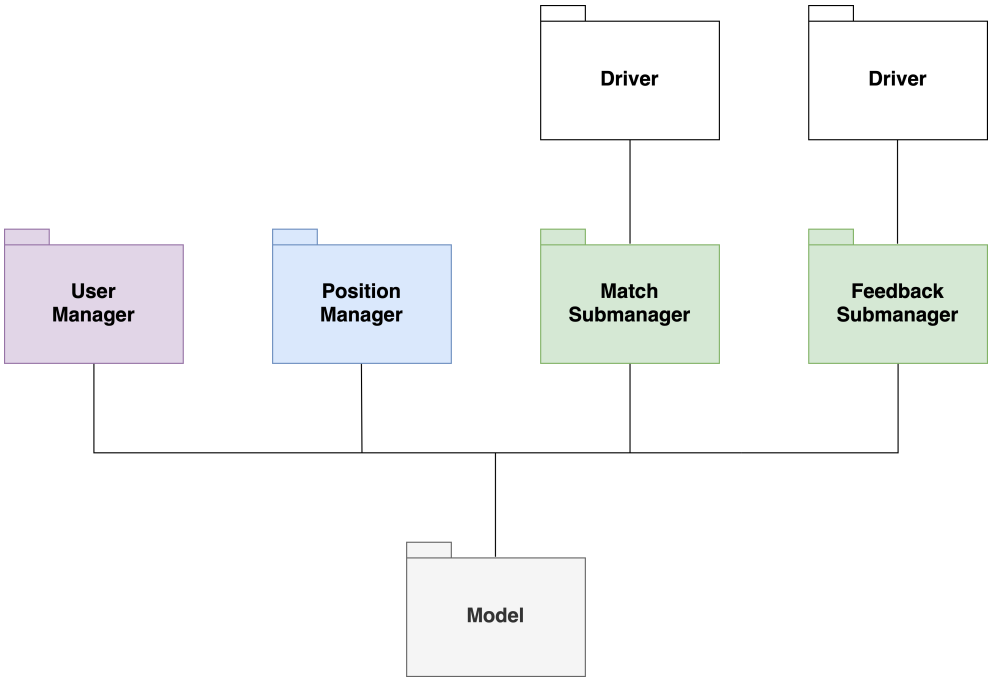

Figure 5.6: Implementation step 6

Step seven applies the same pattern to the selection application submanager.

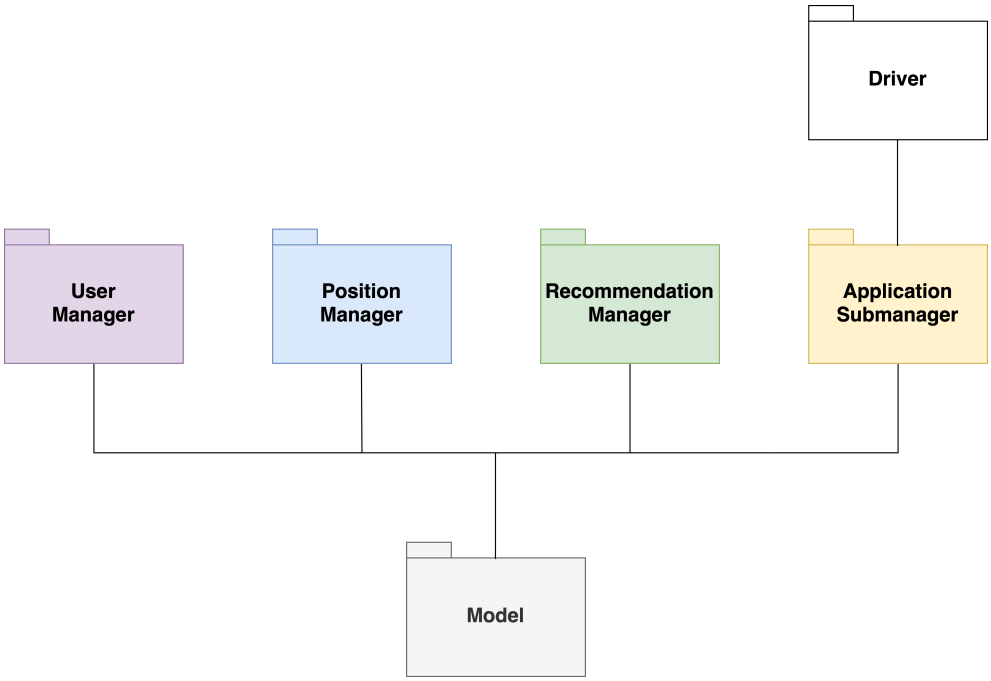

Figure 5.7: Implementation step 7

The selection interview and questionnaire submanagers are integrated in the eighth step.

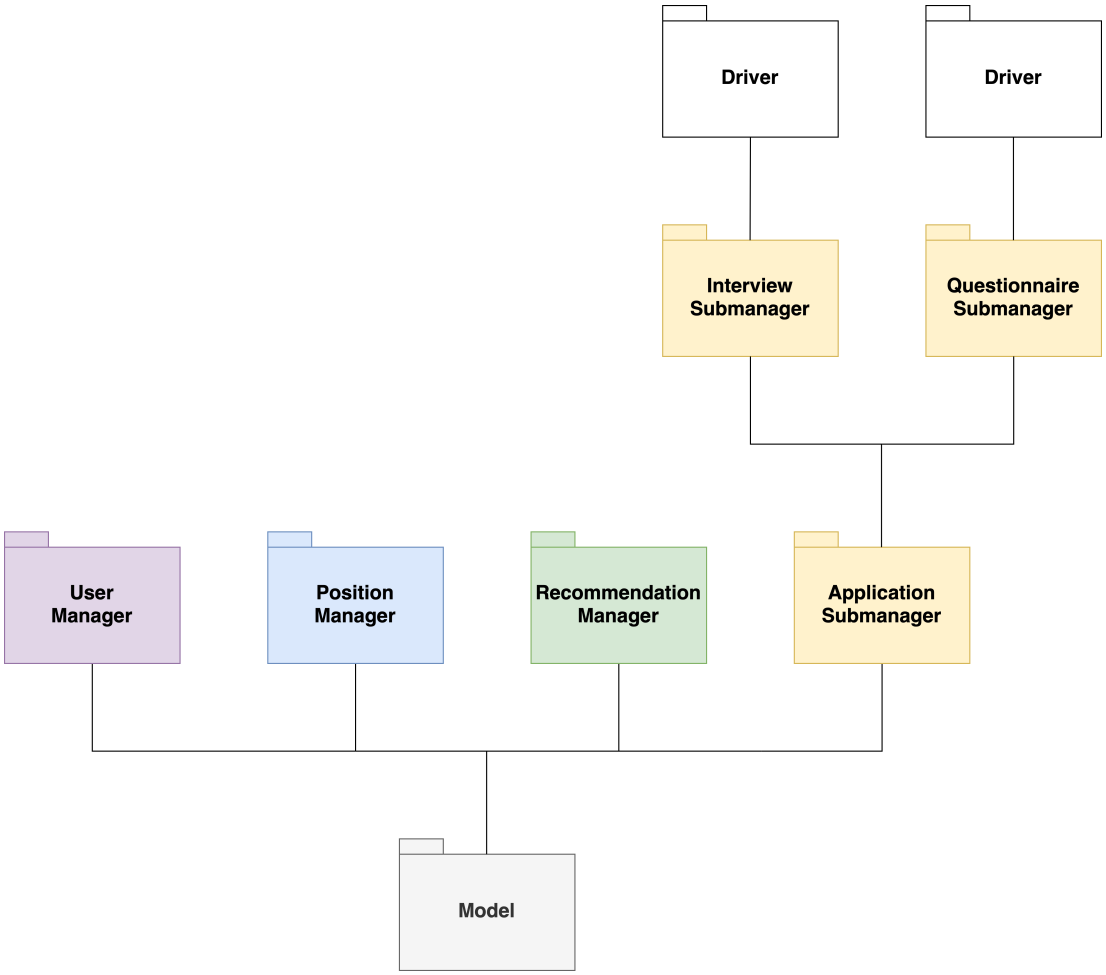

Figure 5.8: Implementation step 8

The development proceeds to step nine with the application submanagers.

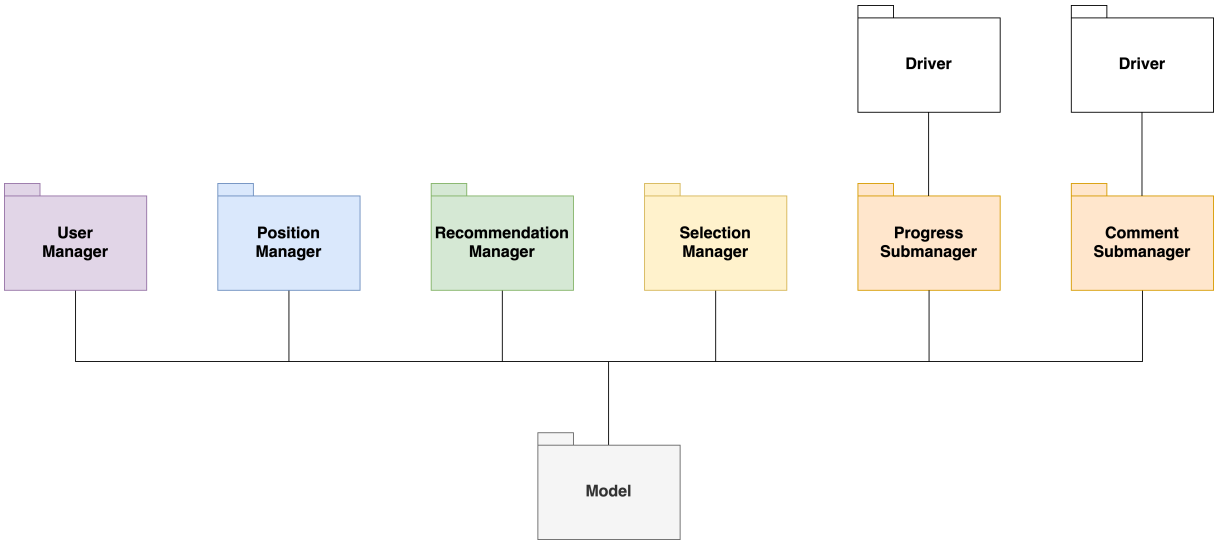

Figure 5.9: Implementation step 9

Step ten applies the same pattern to the notification submanagers.

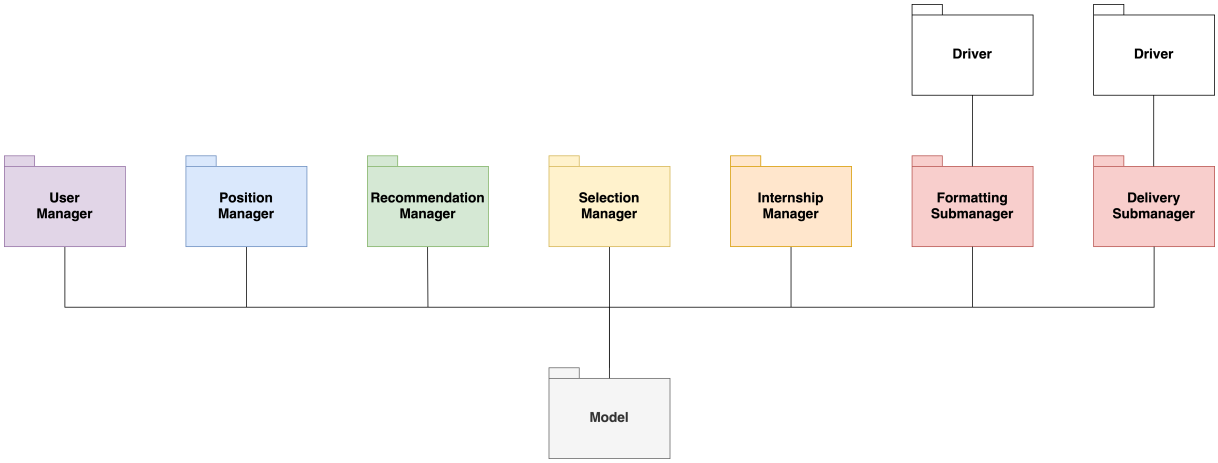

Figure 5.10: Implementation step 10

Note that the notification manager supports all implemented managers.

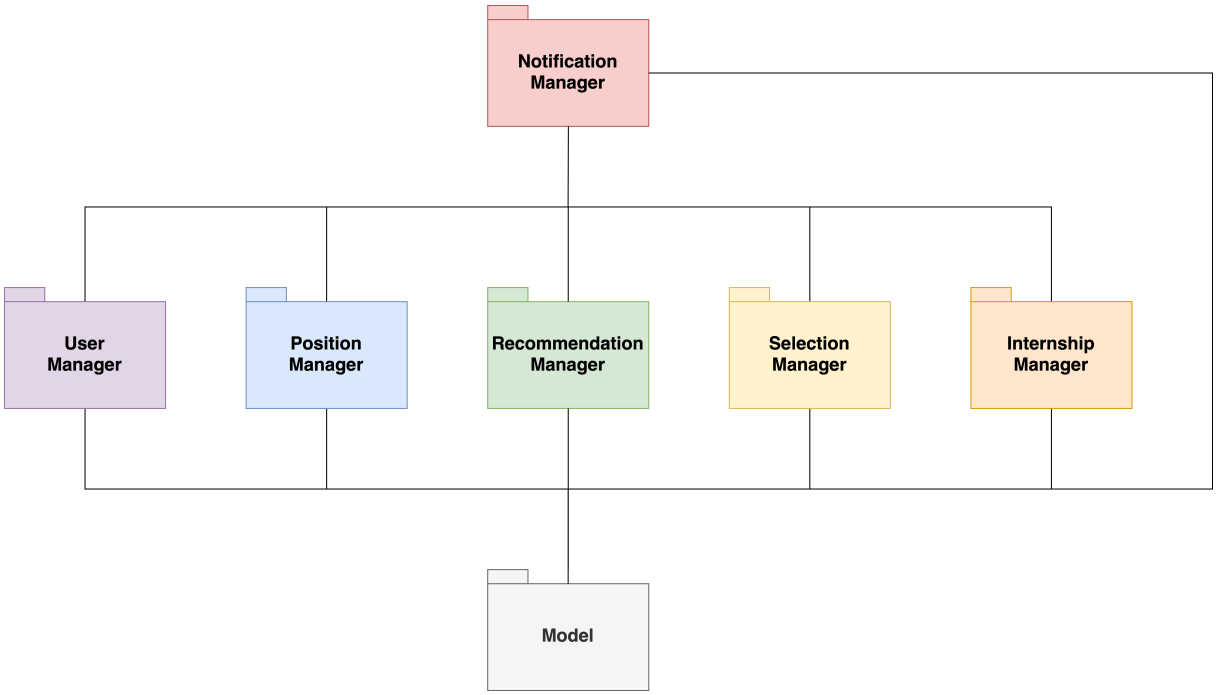

Figure 5.11: Implementation step 11

6 | Workload

This chapter quantifies the workload of the authors in writing the document.

Andrea Carrara

Task	Hours
Introduction	2
Overview	3
Component View	4
Deployment View	2
Runtime View	11
Architectural Styles and Patterns	2
Other Design Decisions	2
Peer Review	14
Total	**40**

Table 6.1: Workload of Andrea Carrara

Federica Currò Dossi

Task	Hours
Component Interfaces	8
User Interface Design	11
Requirements Traceability	2
Implementation, Integration and Test Plan	5
Peer Review	14
Total	**40**

Table 6.2: Workload of Federica Currò Dossi

Bibliography

[1] Andrea Carrara and Federica Currò Dossi. *Students&Companies: Requirements Analysis and Specification Document*. Software Engineering 2, 2024.

[2] Elisabetta Di Nitto. *Requirements Engineering and Design Project*. Software Engineering 2, 2024.

List of Figures

List of Tables