POLITECNICO
MILANO 1863

Design Document

Students & Companies

January 7, 2025

Authors:

- Dimić Nikola
- Tatalović Veljko
- Žiga Edin

Contents

1 Introduction

1.1 Purpose

The purpose of the Students & Companies (S&C) platform is to provide a centralized, user-friendly environment where university students can search for internship opportunities and where companies can advertise their offers. By consolidating internship advertisements in one place, the platform allows students to easily compare opportunities based on criteria such as required skills, offered benefits, and other terms. It ensures a safe and transparent process, minimizing the risk of exploitation by providing mechanisms for university supervision and feedback. Companies benefit from the platform's reach, enabling them to connect with a broad pool of talented students efficiently. In addition, the platform fosters trust by ensuring university oversight, making it a reliable space for students and companies to interact.

1.2 Scope

The S&C platform is designed to streamline the internship process by addressing the needs of three main user groups: Companies, Students, and Universities. Each group is provided with custom functionalities to support their respective roles in the internship ecosystem.

The platform enables companies to efficiently create and manage internship advertisements. These advertisements include detailed descriptions of requirements, projects, and relevant terms, ensuring clarity for prospective applicants. Companies can accept internship applications from students and handle the entire selection process, supported by tools for conducting interviews and distributing structured questionnaires. While the platform is conducting all the necessary steps in order to complete the selection process, interviews are supposed to be held on the external service (Microsoft Teams, Google Meet...) Once an internship is filled, companies are encouraged to provide feedback on their experience with the platform and its functionalities.

Students can utilize the platform to upload their data and CVs, creating comprehensive profiles that highlight their skills and experiences. They can search for internship opportunities that align with their goals and proactively initiate the application process. The platform enables easy and smooth participation in the selection process by allowing students to respond to invitations to interviews and questionnaires. In addition, students receive timely notifications regarding their selection status and are encouraged to provide feedback on their experience. For any challenges encountered during an internship, the platform provides a mechanism for students to submit complaints.

Universities play a crucial role in ensuring the integrity of the internship process by handling complaints submitted by students. They monitor and resolve issues, ensuring that internships provide a safe and productive environment. This oversight contributes to maintaining the reputation of the platform as a trusted intermediary between students and companies. Handling of the complaints should be done between the university, student, and company directly and not through the platform.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Complaint**: A formal request filed by a student to address issues or disputes related to an internship. Complaints are handled and resolved by universities.
- **Feedback**: Insights provided by students or companies about the internship experience. Feedback contributes to platform analytics and recommendations.
- **Status**: The state of an internship, which can be one of the following:
  - Active: Open for applications and participation.
  - Closed: Completed or no longer accepting applications.
  - Draft: In preparation and not visible to students.
- **Recommendation System**: A feature of the platform that suggests internships to students based on their uploaded CVs, skills, and preferences.
- **Complaint Resolution**: The process carried out by universities to address and resolve issues raised by students regarding internships.

1.3.2 Acronyms

- AI: Artificial Intelligence
- API: Application Protocol Interface
- AWS: Amazon Web Service
- HR: Human Resources
- MVC: Model View Controller
- NFRs: Nonfunctional Requirements

- S&C: Students & Companies
- UI: User Interface

1.3.3 Abbreviations

- FR*: Functional Requirement
- UC*: Use Case

1.4 Revision history

v1.0 - 25/12/2024 - Initial release
v1.1 - 28/12/2024 - Finished section 1
v1.2 - 02/01/2025 - Draft Section 2, Section 3
v1.3 - 4/01/2024 - Finalized diagrams for section section 2 & 3, section 4 and 5 draft
v1.4 - 7/01/2024 - Finalized all sections, spell check

1.5 Reference documents

The document is based on the following materials:

- The specification of the *RASD* and *DD* assignment of the *Software Engineering II* course a.a. 2024/25
- Slides of the course on WeBeep
- Larman, C. (2002). Applying UML and patterns: An introduction to object-oriented analysis and design and the Unified Process. Prentice Hall PTR.

1.6 Document structure

This DD document consists of the following parts:

1. **Introduction**: A brief description of the project that is focused on the purpose, the goals that are aimed to be achieved, and the scope that is covered with its development.
2. **Architectural Design**: Shows how Students & Companies is built on a three-tier setup hosted on AWS, using containerization for easy scaling. It highlights key back-end managers for core functionalities and illustrates how the system can run multiple instances across different servers as needed.

3. **User Interface Design** : Presents possible layouts of the user interface for the mobile and desktop applications.

4. **Requirements Traceability**: Explains how the requirements defined in the RASD map the design components explained in this document.

5. **Implementation, Integration and Test Plan**: Detailed explanation of how the platform described in the previous sections will be developed and tested.

6. **Effort Spent**: The references to any documents and software used to create this document.

2 Architectural Design

2.1 Overview

As the S&C platform will be entirely hosted using Amazon Web Services (AWS), it will utilize a three-layer architecture by design. This architecture consists of the presentation layer for the user interface and interaction, the application layer for data processing, and finally, the database layer where all application data will be stored. The choice of AWS and this architecture came from the ease of deployment, as AWS provides a ready-made solution for this type of application. As this section as a whole includes many UML diagrams, it must be noted that they were produced by the standards proposed by C. Larman[1].

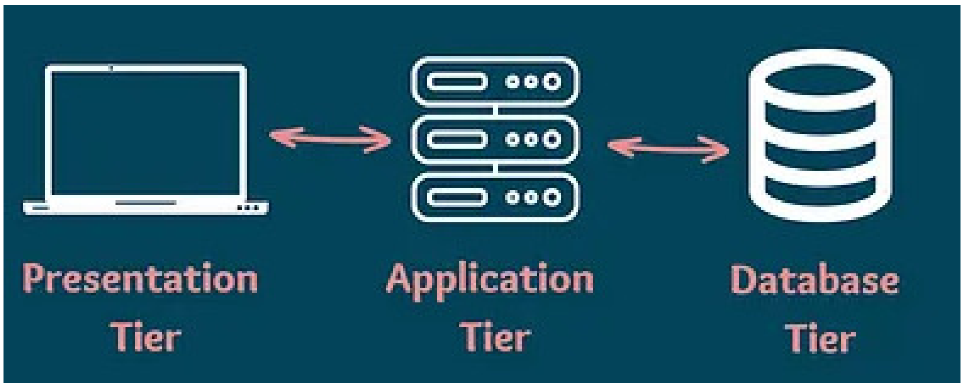


Figure 1: Three tier architecture [2]

2.1.1 Distributed view

Figure 2 presents the distributed view of the system, showcasing its function and interaction with other elements. As noted, the system will be hosted on AWS, which utilizes containerization and Kubernetes, allowing the system to scale as needed. Figure 2 depicts a hypothetical example of 27 instances of the service running across three servers and their interactions.

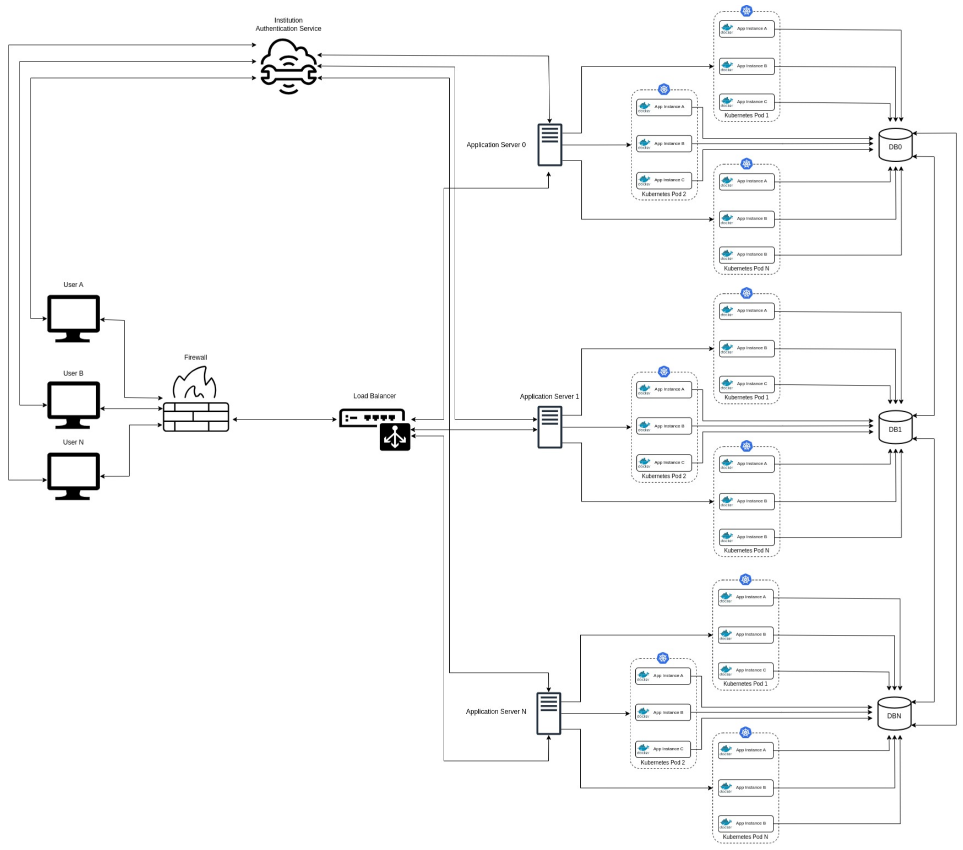


Figure 2: Distributed view

2.2 Component view

As AWS is widely regarded as the ”golden application” of the three-tier architecture[2], the presented component view depicts the components running on the Application Tier, or the second layer, in other words, the system's back end. The Database Tier and Presentation Tier components were segregated for better clarity. While most components are self-explanatory, some possess complex responsibilities. This section clarifies their roles within the system.

- AccountManager: Provides functionalities for managing user profiles.
- AdvertisementManager: Manages internship advertisements on the platform.
- Application Protocol Interface (API): Serves as an endpoint for receiving, routing, and transmitting data between the active Application instances and the back-end system.

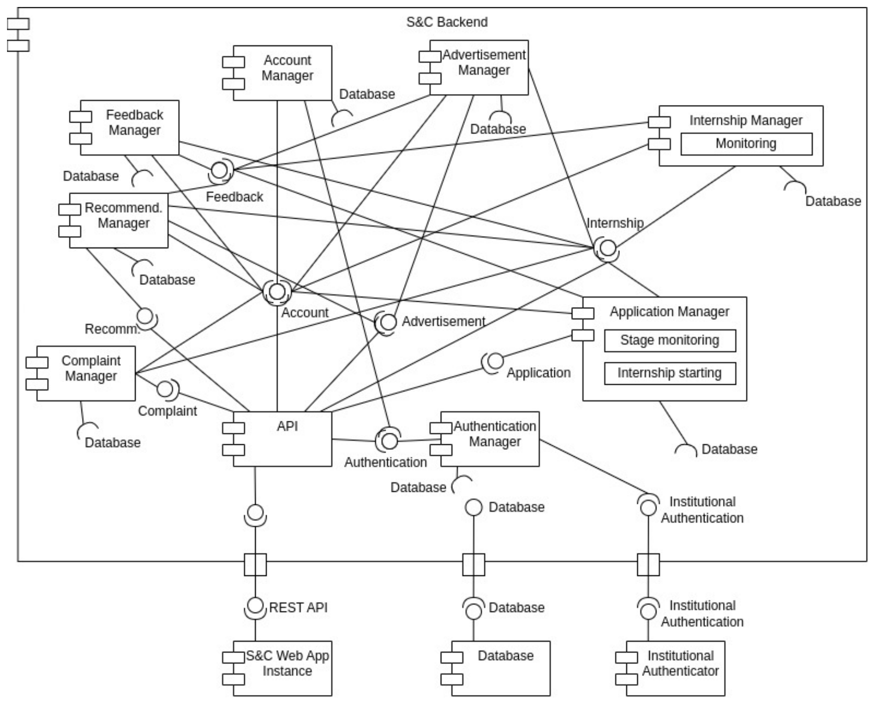


Figure 3: Component Diagram

- ApplicationManager: Processes student internship applications, including stage transitions, rejections, acceptances, and internship initialization.
- AuthenticationManager: Handles user authentication and routes authentication requests to institutional authenticators.
- ComplaintManager: Processes the submission of complaints during active internships.
- FeedbackManager: Processes feedback for users involved in internships.
- InternshipManager: Manages internships and the associated users. Handles user state transitions during internships and initiates feedback requests.
- RecommendationManager: Provides recommendations to students based on user data.

Presentation Layer and Database Layer components:

- Database: Stores platform data (Layer 3).

- Institutional Authenticator: An external service whose API enables institutional login for users.
- S&C Web App Instance: An instance of the web application (Layer 1).

2.3 Deployment view

The deployment view presented in Figure 4 depicts a simplified instance of the S&C running on AWS, similar to the diagram in Section 2.1. This view differs in that it takes additional hardware and software elements into account, but from a logical standpoint, it remains very similar to the diagram in Figure 2. Moreover, it should be noted that this diagram highlights a single instance of the system, and does not present the possibility of multiple pods with multiple containers on multiple servers, as shown in Figure 2.

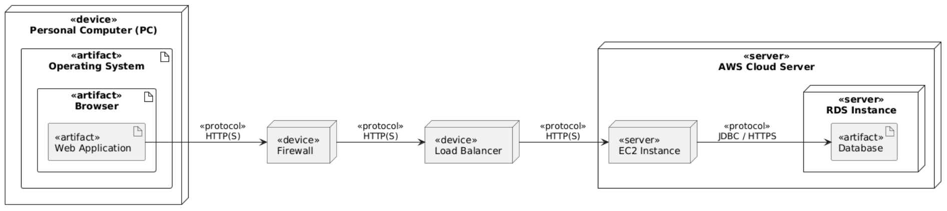


Figure 4: Deployment view

2.4 Runtime view

The following section presents the runtime view diagrams for each use case outlined in the RASD v1 document accompanying this report. It covers 17 use cases and illustrates the interactions between the system's components. It is important to note that the API interface artifact has been excluded from the diagrams to avoid unnecessary clutter, as its inclusion offers little to no benefit to the communication process between components. For the sake of simplicity, this section assumes that the API is integrated as part of the AWS Server component.

[UC1] Student and University User Registration

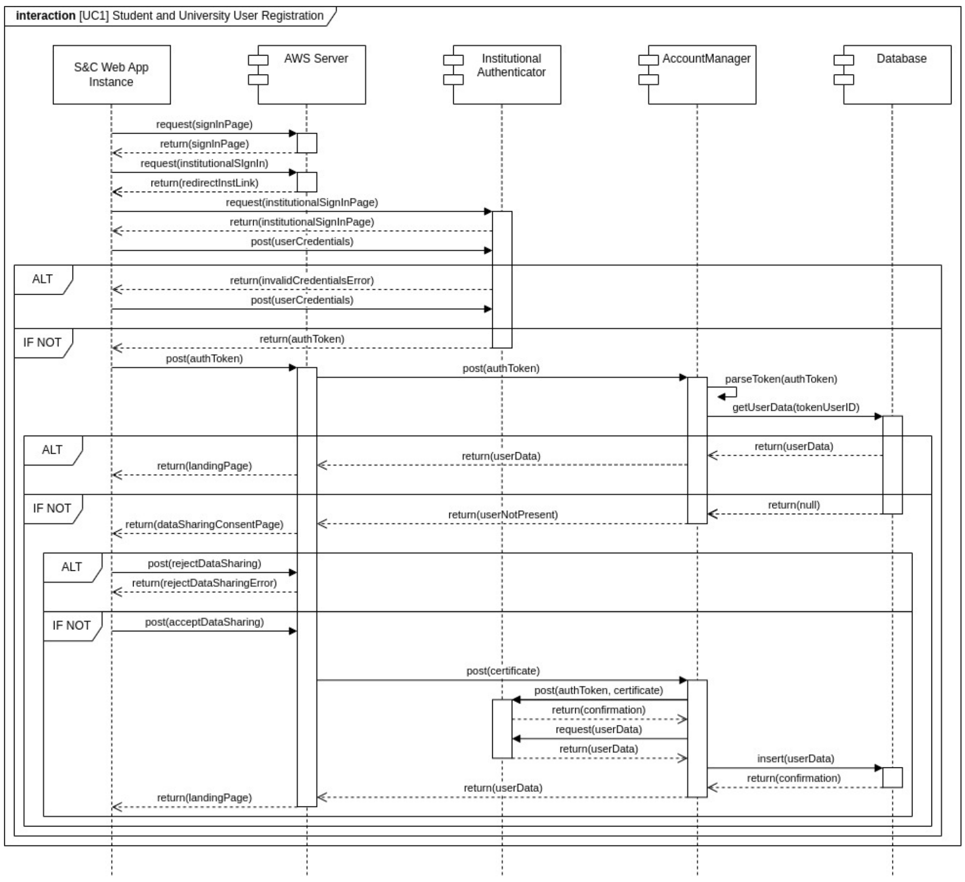


Figure 5: UC1 - Runtime View Diagram

The diagram in Figure 5 illustrates the interactions between system components during the Student and University user registration process. As noted, the system supports the use of an external authenticator (such as Microsoft 365), which provides the data required to set up a user's profile. The diagram shows the data transfer and storage in the platform's database at the final step, while the preceding elements depict the communication and exchange of the authentication token between various system components.

[UC2] Student and University User Login

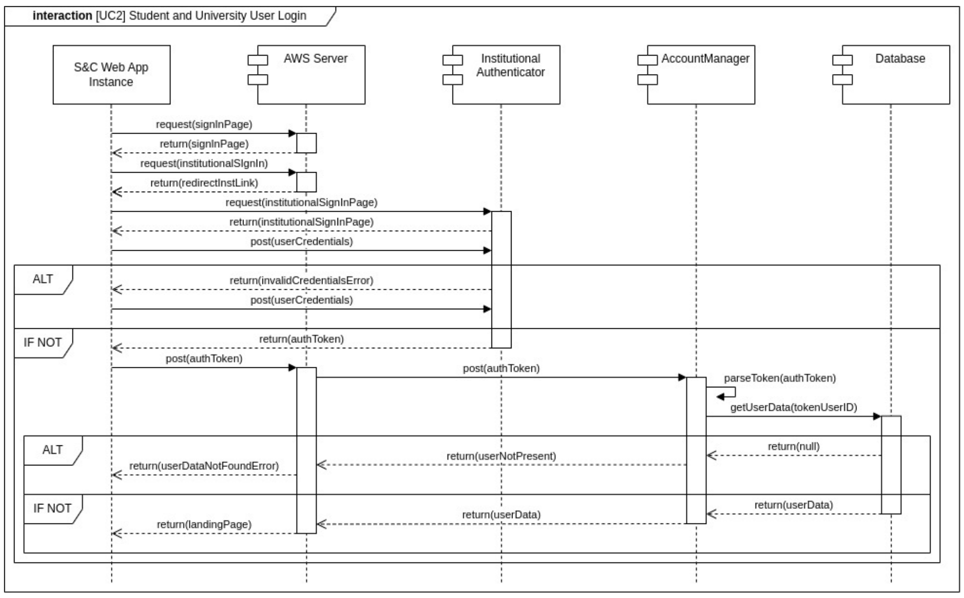


Figure 6: UC2 - Student and University User Registration

The second diagram can be considered a derivative of the first, as the communication process closely resembles that of the registration process. It should be noted that the final alternative flow depicted in this diagram actually marks the beginning of the registration process. If a user attempts to authenticate with an account that is not registered with the service, the system will automatically initiate the sign-in process.

[UC3] Register Company User

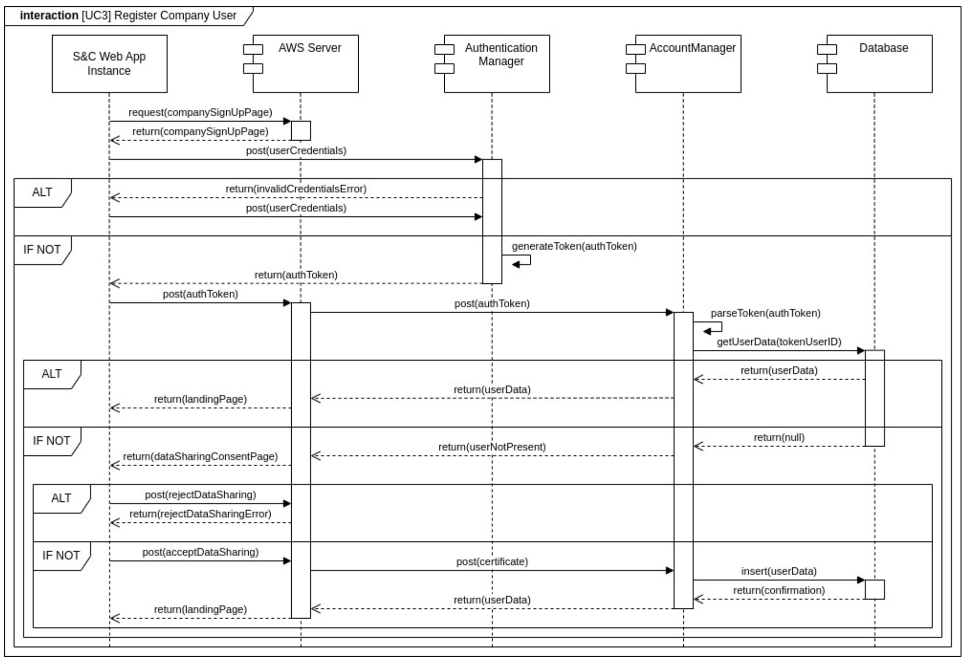


Figure 7: UC3 - Register Company User

The company user registration process is very similar to the student and university user registration, except that companies do not have access to external authentication services, unlike students and universities. The communication process remains largely the same throughout.

[UC4] Login Company User

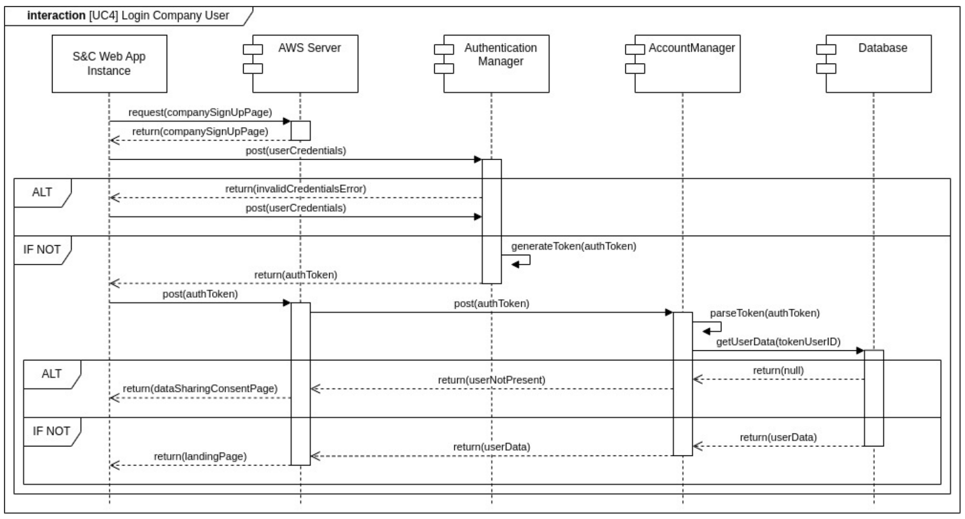


Figure 8: UC4 - Login Company User

As before, the login diagram is essentially a variation of the sign-up diagram, as the communication remains largely similar between the two processes. It should be noted that the final alternative flow presented leads to the option of signing up a new company user.

[UC5] Edit profile

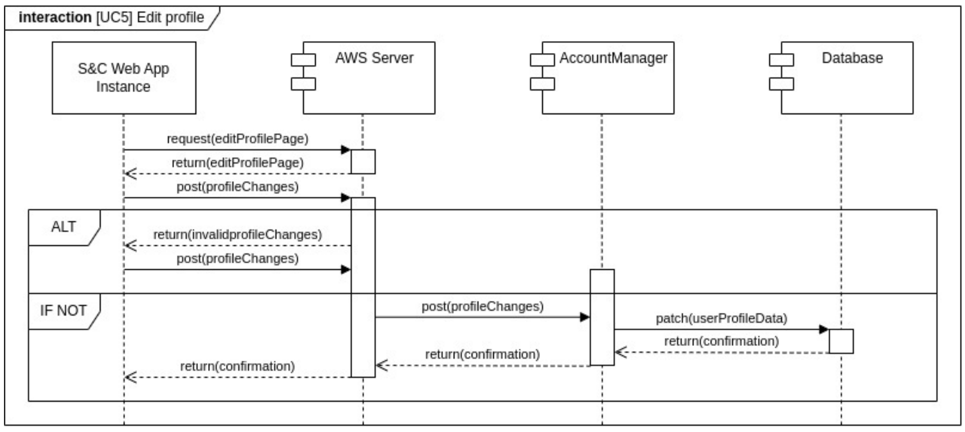


Figure 9: UC5 - Edit profile

The edit profile interaction assumes that the user is fully authenticated and has accessed the platform's landing page, as outlined in the original use case in the provided RASD v1 document. This explains the absence of the Authentication Manager component. Furthermore, the Account Manager component handles most of the required edits during this interaction.

[UC6] Create and Publish Internship Advertisement

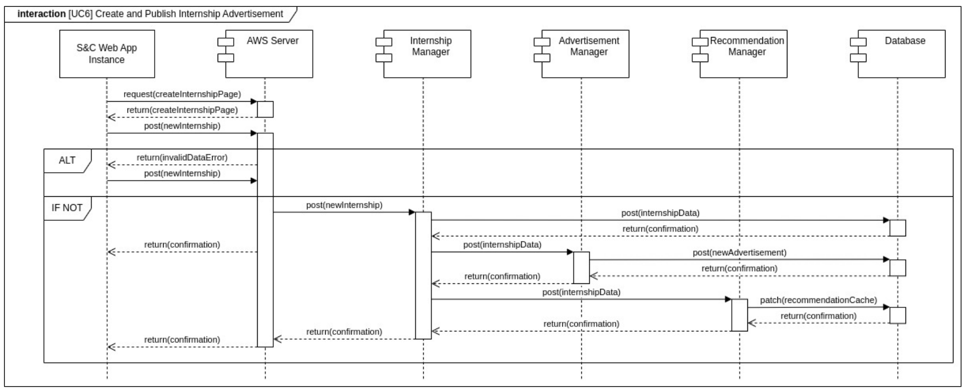


Figure 10: UC6 - Create and Publish Internship Advertisement

The sixth use case, which involves creating and publishing a new internship advertisement, includes some complex interactions to achieve the desired functionality. These interactions primarily occur between the Internship Manager, the Advertisement Manager, and the Recommendation Manager, as the latter two depend on the Internship Manager's functionality. The Internship Manager creates the internship in the database and then passes the relevant internship data to enable the creation and publication of the advertisement, as well as to update the recommendations cache.

[UC7] Edit Internship Advertisement

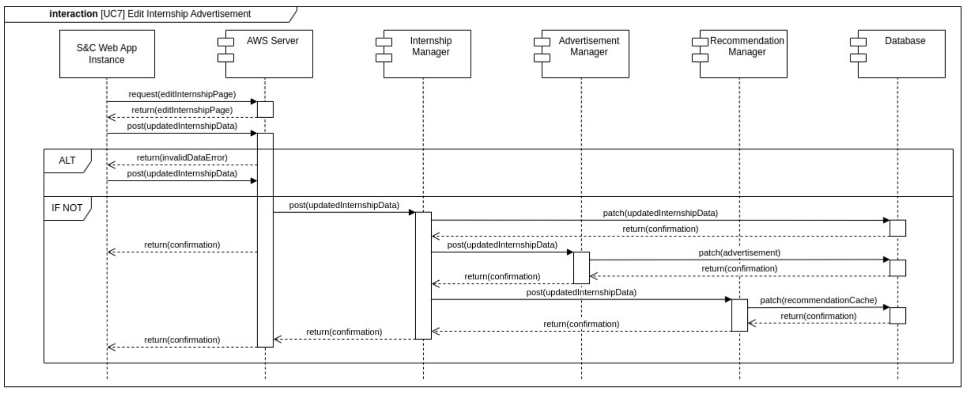


Figure 11: UC7 - Edit Internship Advertisement

The interactions required for editing an active internship advertisement are largely similar, if not identical, to those in UC6, with the main difference being the type of operations conducted between certain components (primarily, inserts being replaced with patches).

[UC8] Delete Internship Advertisement

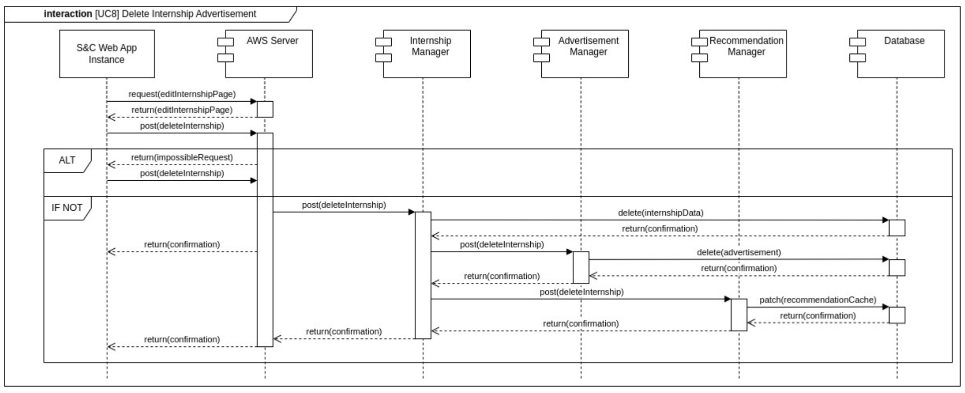


Figure 12: UC8 - Delete Internship Advertisement

As with the previous two use cases, the interactions between the components remain largely the same, with minor changes to the types of operations (specifically, inserts/patches being replaced by delete operations).

[UC9] View Recommendations & Offerings, Apply for Internship

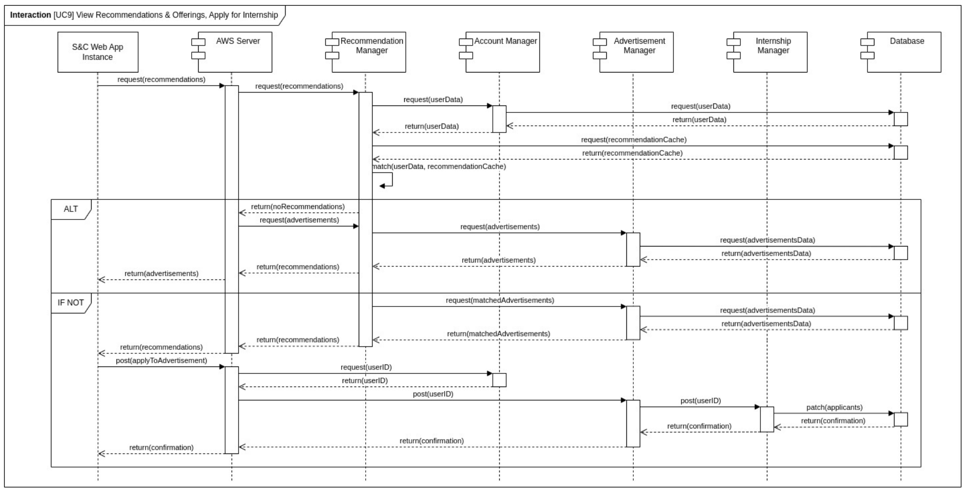


Figure 13: UC9 - View Recommendations & Offerings, Apply for Internship

The UC9 Runtime View diagram represents one of the more complex diagrams in this section. For simplification, it can be divided into two stages: the first stage involves fetching or generating relevant recommendations for a single user, while the second stage covers the application process for a single internship advertisement.

In the first stage, the Recommendations Manager gathers data based on the logged-in user's profile and retrieves its stored cache of internship advertisements from the database. Once the required data is retrieved, a matching algorithm runs to provide the user with advertisements relevant to their profile. An alternative flow exists where, if no match is found, the system returns a randomized set of advertisements.

The second stage handles the application process, which involves much simpler inter-component communication compared to the first stage, as shown in the diagram.

[UC10] Change candidate status (increment or reject)

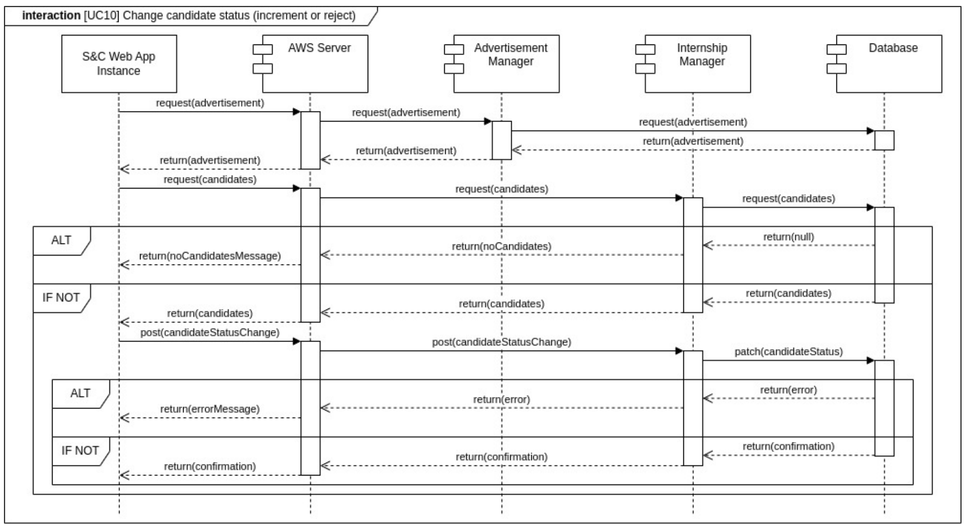


Figure 14: UC10 - Change candidate status (increment or reject)

UC10 illustrates the communication required to change a candidate's status. It begins by fetching all advertisements relevant to the user, and once one is selected, it retrieves all the candidates for that internship. An alternative flow exists for the scenario where no candidates have applied for the specific internship. Otherwise, the sequence continues by issuing a patch to the database for the selected candidate.

[UC11] Send Prompt to Candidate (questionnaire or interview invitation)

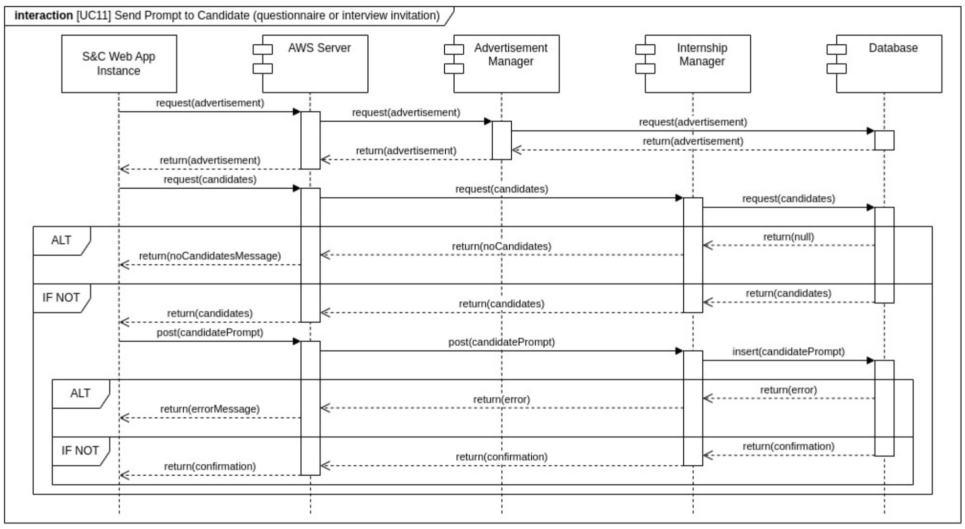


Figure 15: UC11 - Send Prompt to Candidate (questionnaire or interview invitation)

UC11 follows many of the same sequences presented in UC10, with the main differences lying in the types of communications between the components, such as patches being substituted for inserts.

[UC12] Reply to invitation

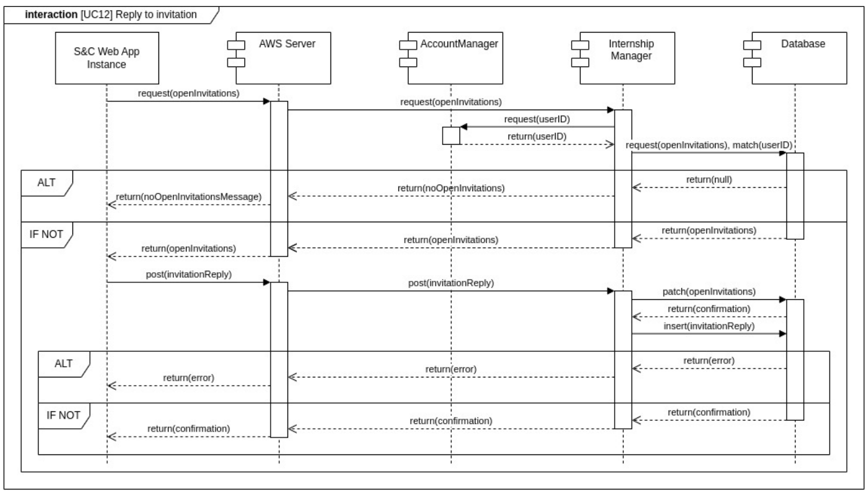


Figure 16: UC12 - Reply to invitation

Like UC11, the required interaction for UC12 is largely similar, but it contains a subtle difference, as two distinct database edits are required to complete it. First, a patch operation is performed to close the open invitation, followed by an insert operation to add the reply to the database.

[UC13] Initiate Internship

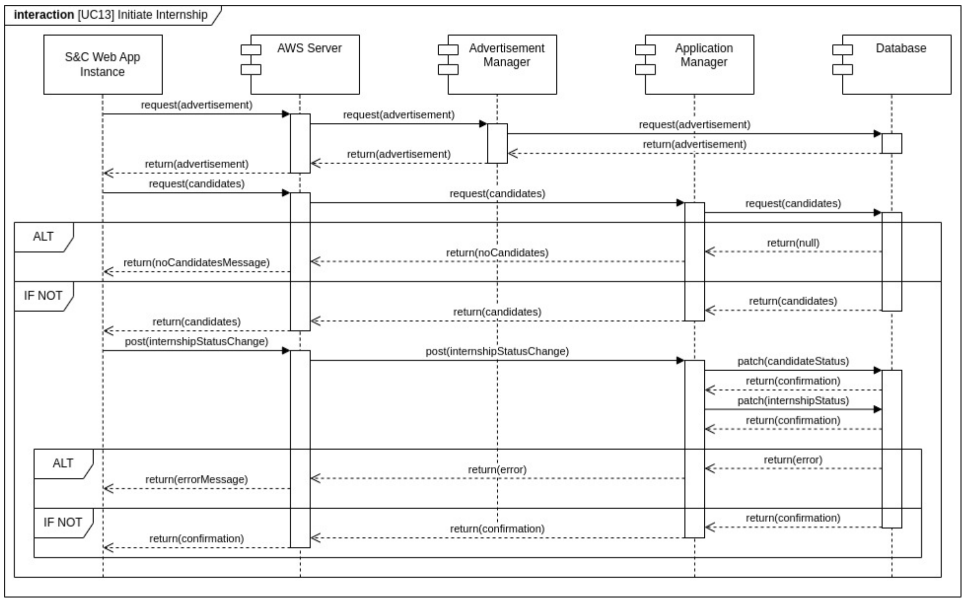


Figure 17: UC13 - Initiate Internship

UC13's communication process can be viewed as a mix of several presented use cases, with a degree of overlap between them. Ultimately, it comes down to having candidates who are eligible for the start of an internship, as well as two database edits required to initiate it.

[UC14] Submit Complaint

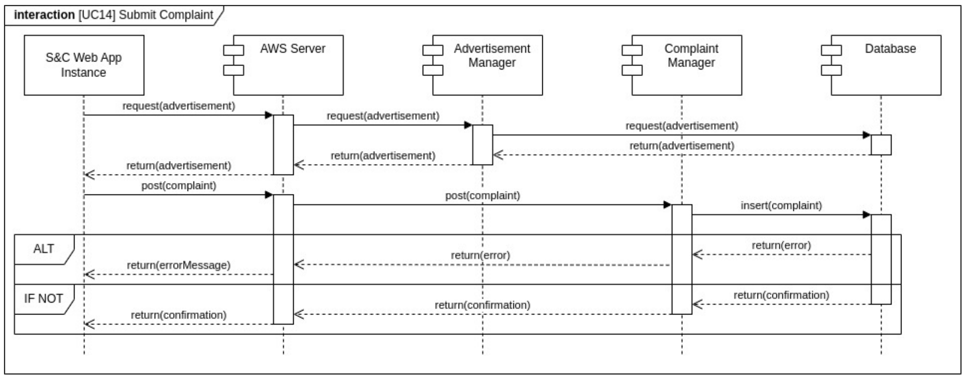


Figure 18: UC14 - Submit Complaint

The submit complaint runtime view is one of the simpler views in this document, as the interactions required among the components for its functionality are minimal. As shown in the diagram, the main actor is the complaint manager, who processes the complaint.

[UC15] Resolve complaint

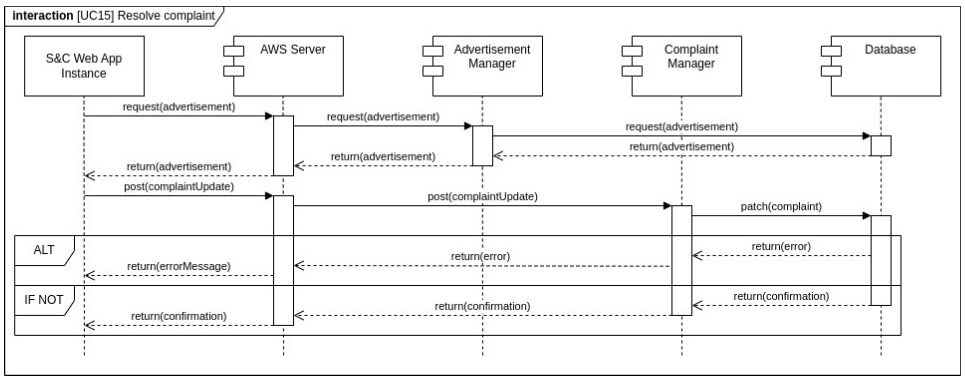


Figure 19: UC15 - Resolve complaint

As with many other use cases, UC15 is a simple variation of UC14, where the required operations are merely adjusted for the specific use case.

[UC16] Submit feedback

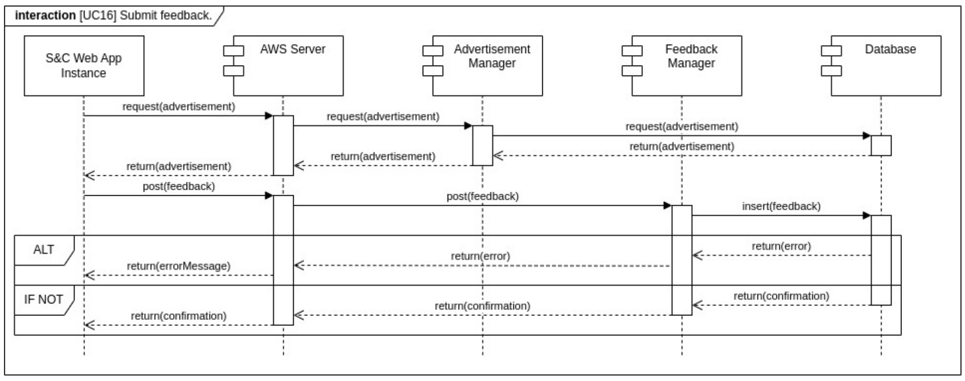


Figure 20: UC16 - Submit feedback

For the submission of feedback, there is a requirement that an internship be finished (either by its natural course or by early termination). Afterward, users may submit feedback, which is handled through the feedback manager, as shown in the diagram.

[UC17] View feedback

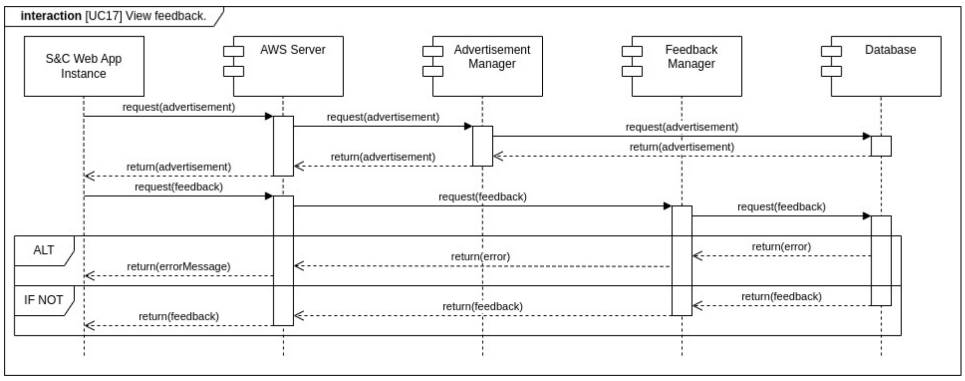


Figure 21: UC17 - View feedback

Lastly, the view feedback interaction is a simple variation of UC16, and like many of the previous ones, consists of minor changes in the types of interactions between the components.

2.5 Component interfaces

1. **AccountManager**
   (a) getUserData(tokenUserID)
   (b) insert(userData)
   (c) parseToken(authToken)
   (d) post(authToken, certificate)
   (e) request(userData)
   (f) return(userData)
   (g) return(userID)
   (h) return(userNotPresent)
2. **AdvertisementManager**
   (a) delete(advertisement)
   (b) patch(advertisement)
   (c) post(newAdvertisement)
   (d) post(userID)
   (e) request(advertisements)
   (f) return(confirmation)
3. **AuthenticationManager**

(a) generateToken(authToken)
(b) return(authToken)
(c) return(confirmation)
(d) return(invalidCredentialsError)
(e) return(institutionalSignInPage)
(f) return(userData)

4. **AWS Server**

   (a) post(authToken)
   (b) post(certificate)
   (c) post(deleteInternship)
   (d) post(feedback)
   (e) post(internshipStatusChange)
   (f) post(invitationReply)
   (g) post(newInternship)
   (h) post(updatedInternshipData)
   (i) request(advertisement)
   (j) request(candidates)
   (k) request(feedback)
   (l) request(openInvitations)
   (m) request(recommendations)
   (n) request(userID)
   (o) return(advertisement)
   (p) return(confirmation)
   (q) return(dataSharingConsentPage)
   (r) return(editProfilePage)
   (s) return(impossibleRequest)
   (t) return(invalidProfileChanges)
   (u) return(landingPage)
   (v) return(noOpenInvitations)
   (w) return(openInvitations)
   (x) return(rejectDataSharingError)
   (y) return(signInPage)
   (z) return(redirectLink)
   () return(userData)

5. **ComplaintManager**
   (a) insert(complaint)
   (b) patch(complaint)
   (c) return(confirmation)
   (d) return(error)
6. **Database**
   (a) return(advertisement)
   (b) return(confirmation)
   (c) return(error)
   (d) return(userData)
7. **FeedbackManager**
   (a) insert(feedback)
   (b) request(feedback)
   (c) return(confirmation)
   (d) return(error)
8. **InstitutionalAuthenticator**
   (a) return(authToken)
   (b) return(confirmation)
   (c) return(institutionalSignInPage)
   (d) return(invalidCredentialsError)
   (e) return(userData)
9. **InternshipManager**
   (a) delete(internshipData)
   (b) insert(application)
   (c) insert(candidatePrompt)
   (d) insert(invitationReply)
   (e) patch(internshipStatus)
   (f) patch(openInvitations)
   (g) patch(updatedInternshipData)
   (h) post(deleteInternship)
   (i) post(internshipData)
   (j) request(candidates)

(k) request(openInvitations)
(l) request(userID)
(m) return(error)
(n) return(noCandidates)
(o) return(noOpenInvitations)
(p) return(signInPage)

10. **RecommendationManager**
    (a) match(userData, recommendationCache)
    (b) patch(recommendationCache)
    (c) request(advertisements)
    (d) request(matchedAdvertisements)
    (e) request(recommendationCache)
    (f) request(userData)
    (g) return(confirmation)
    (h) return(recommendations)

11. **S&C Web App Instance**
    (a) post(acceptDataSharing)
    (b) post(applyToAdvertisement)
    (c) post(candidatePrompt)
    (d) post(complaint)
    (e) post(complaintUpdate)
    (f) post(feedback)
    (g) post(invitationReply)
    (h) post(internshipStatusChange)
    (i) post(profileChanges)
    (j) post(rejectDataSharing)
    (k) post(updatedInternshipData)
    (l) request(advertisement)
    (m) request(candidates)
    (n) request(editProfilePage)
    (o) request(feedback)
    (p) request(institutionalSignIn)
    (q) request(institutionalSignInPage)

(r) request(openInvitations)

(s) request(recommendations)

(t) request(signInPage)

(u) return(landingPage)

(v) return(noOpenInvitations)

(w) return(openInvitations)

2.6 Selected architectural styles and patterns

2.6.1 Three-Tier Architecture

As mentioned earlier, the platform will be fully implemented using AWS, which is regarded as one of the best examples of the three-tier architecture. More details can be found at the beginning of this section regarding the exact implementation for this specific platform, but in essence,

2.6.2 Model-View-Controller Pattern

For an application like this one, utilizing the Model-View-Controller (MVC) pattern is a logical choice. Given the chosen application hosting solution, using this pattern makes development much quicker and easier for all parties involved, while providing many benefits for long-term system evolution.

2.6.3 Facade Pattern

As the system will undoubtedly evolve over time with the addition of new features and many other adaptations, it will utilize the Facade pattern to simplify the interaction process as the number of subsystems increases. This makes system changes and evolution easier to manage.

2.7 Other design decisions

Unlike other implementations, many of the design decisions regarding availability, data storage, and security will be omitted simply because AWS provides a plug-and-play solution to many of these concerns.

2.7.1 Availability

The system is expected to be available 24/7 with minimal impact on the performance of the platform, regardless of the number of users interacting with it at any given time. As these design decisions are commonly referred to as Nonfunctional Requirements (NFRs), more details can be found within the RASD v1 document provided alongside this design document.

2.7.2 Data Storage

In terms of data storage, the platform will utilize a classic SQL-based database, with the possibility of an additional document-based database being implemented alongside the main database should the need arise. Hosting the database on AWS allows for easy expansion, regardless of the eventual size of the resulting database.

2.7.3 Security

As the system will support external authentication (many of which support and require multi-factor authentication to function), many security concerns regarding the protection of user accounts from brute-force attacks have already been addressed. Moreover, the platform will exclusively use secure communication protocols and multiple layers of encryption, in addition to being hosted on AWS, which should provide more than adequate security measures for the scope of this platform.

3 User Interface Design

In this chapter, possible layouts of the user interface for the mobile and desktop applications are presented.

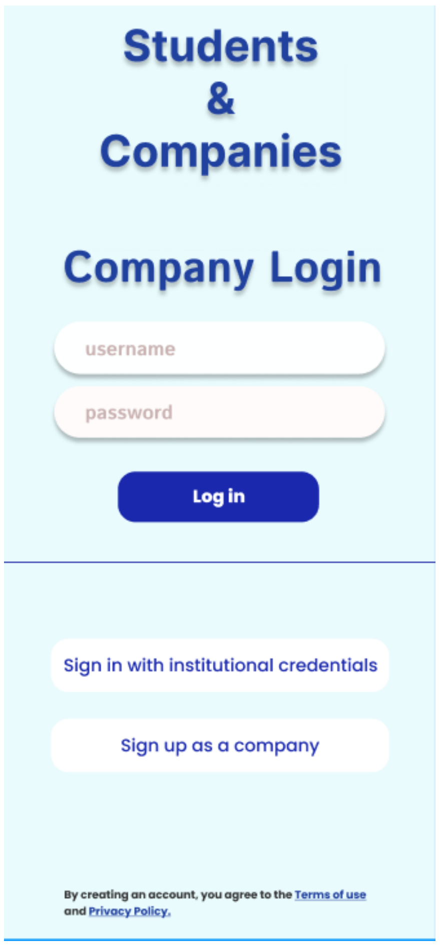

Students
&
Companies
Company Login
username
password
Log in
Sign in with institutional credentials
Sign up as a company
By creating an account, you agree to the Terms of use and Privacy Policy.

Figure 22: Log in form on mobile application

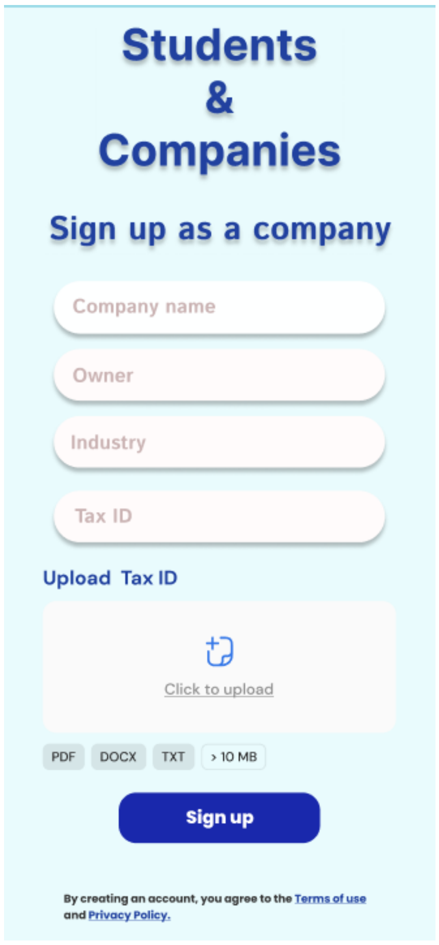
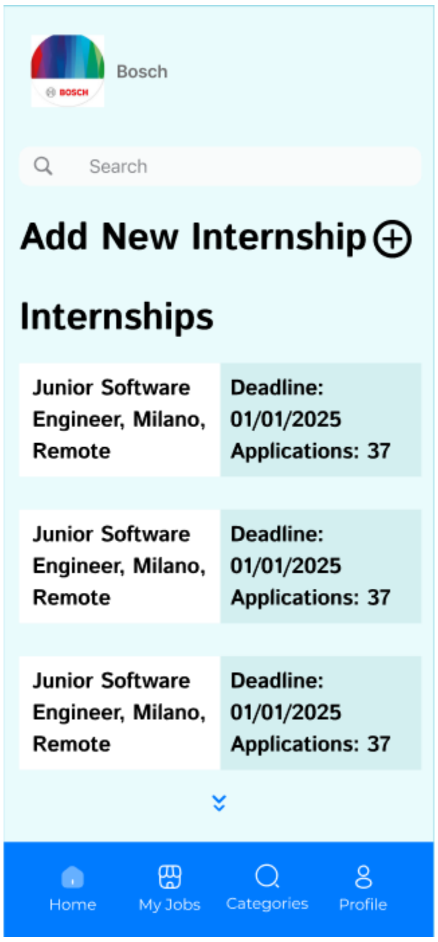


Figure 23: Sign up for companies on mobile application



Bosch

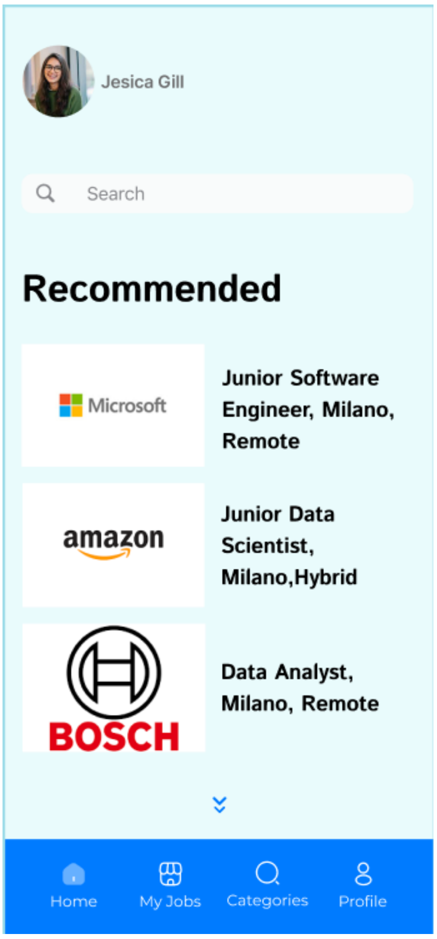
Search

Add New Internship ⊕

Internships

Junior Software Engineer, Milano, Remote	Deadline: 01/01/2025 Applications: 37
Junior Software Engineer, Milano, Remote	Deadline: 01/01/2025 Applications: 37
Junior Software Engineer, Milano, Remote	Deadline: 01/01/2025 Applications: 37

Figure 24: Main page for companies on mobile application



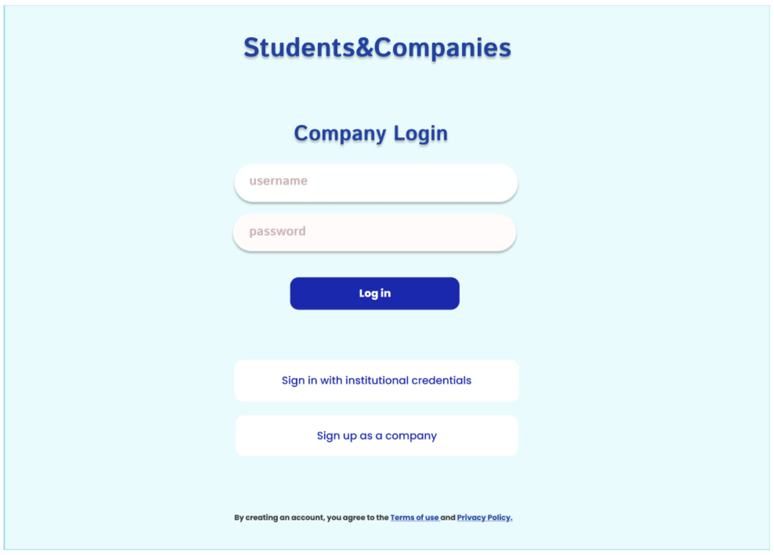
Jesica Gill

Search

Recommended

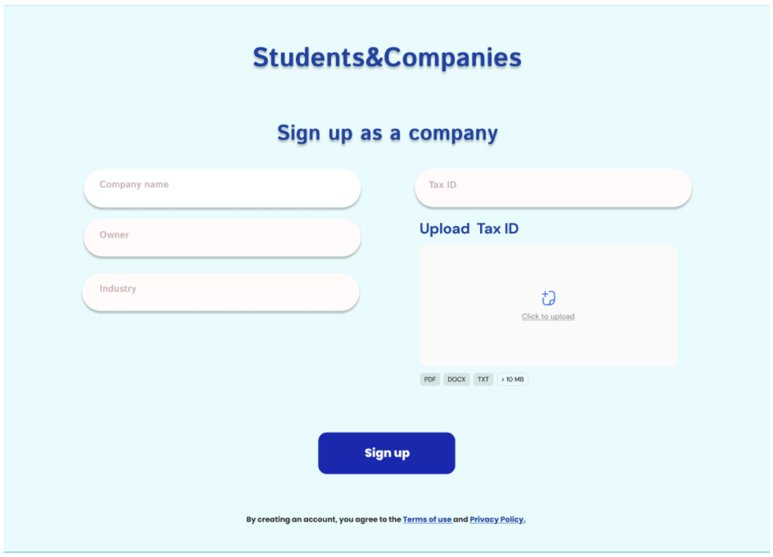
Microsoft	Junior Software Engineer, Milano, Remote
amazon	Junior Data Scientist, Milano,Hybrid
BOSCH	Data Analyst, Milano, Remote

Figure 25: Main page for students on mobile application



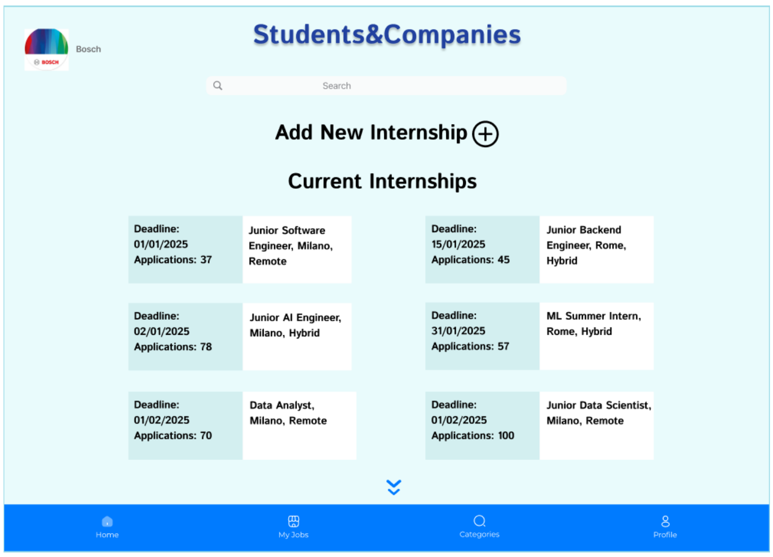
By creating an account, you agree to the Terms of use and Privacy Policy.

Figure 26: Log in form on desktop application



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

Figure 27: Sign up for companies on desktop application



Deadline: 01/01/2025 Applications: 37 — Junior Software Engineer, Milano, Remote

Deadline: 15/01/2025 Applications: 45 — Junior Backend Engineer, Rome, Hybrid

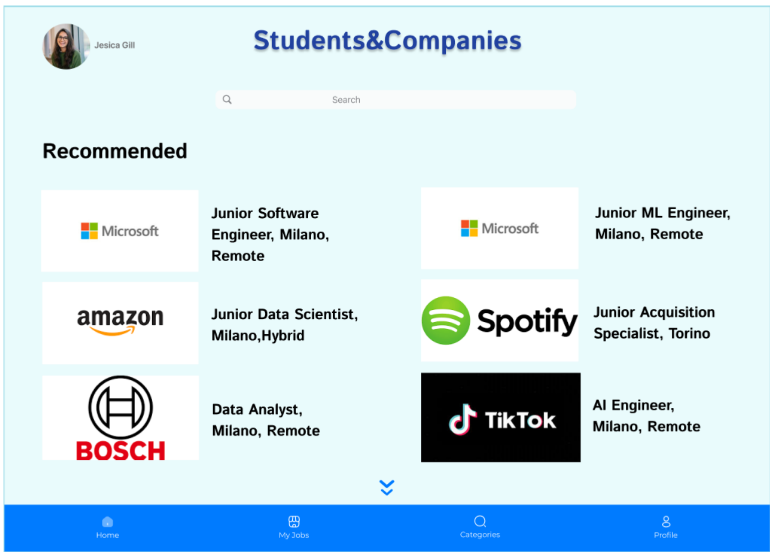
Deadline: 02/01/2025 Applications: 78 — Junior AI Engineer, Milano, Hybrid

Deadline: 31/01/2025 Applications: 57 — ML Summer Intern, Rome, Hybrid

Deadline: 01/02/2025 Applications: 70 — Data Analyst, Milano, Remote

Deadline: 01/02/2025 Applications: 100 — Junior Data Scientist, Milano, Remote

Figure 28: Main page for companies on desktop application



Jesica Gill

Students&Companies

Search

Recommended

Junior Software Engineer, Milano, Remote

Junior ML Engineer, Milano, Remote

Junior Data Scientist, Milano,Hybrid

Junior Acquisition Specialist, Torino

Data Analyst, Milano, Remote

AI Engineer, Milano, Remote

Figure 29: Main page for students on desktop application

4 Requirements Traceability

In this section it is explained how the requirements defined in the RASD, map the design components explained in this document.

Requirements	**Registration of Students** [FR1] The system shall allow new student users to register on the platform using their respective institutional accounts. [FR2] After an institutional account has been linked with the platform, the system shall allow student users to log in using their institutional credentials. [FR3] The system shall allow the registration of student users only if they are classified as students within the institutional account they provide.
Components	AccountManager Application Protocol Interface (API) AuthenticationManager Database InstitutionalAuthenticator S&C Web App Instance

Requirements	**Registration of Companies** [FR4] The system shall allow new company users to register on the platform using a valid email address. [FR5] After providing an email address, the system shall verify the provided address through a verification email. [FR6] After the new company user has verified their email, the system shall generate and deliver a password for the newly created account via the verified email. [FR7] The system shall allow the company user to log in using the validated email address and the provided password. [FR8] Upon the first login, the system shall require the new user to provide a certificate from the relevant tax authority, including the company name and unique identifier. The system shall also require the full company name and unique identifier to be entered manually in the relevant input fields alongside the certificate. [FR9] The system shall retain company user data only at two points: first, after successful email verification, where the provided email address and generated password will be retained as login credentials; and second, after submitting the initial verification information, where all submitted data up to that point shall be retained.
Components	AccountManager Application Protocol Interface (API) AuthenticationManager Database S&C Web App Instance

Requirements	**Registration of Universities** [FR10] The system shall allow new university users to register on the plat- form using pre-designated institutional accounts allocated for S&C. [FR11] After an institutional account has been linked with the platform, the system shall allow university users to log in using their institutional credentials. [FR12] If the institutional account has not been pre-designated for use on S&C, the system shall reject the registration attempt.
Components	AccountManager Application Protocol Interface (API) AuthenticationManager Database InstitutionalAuthenticator S&C Web App Instance

Requirements	**Profile Management** [FR13] The system shall, by default, provide a blank profile template for all users. [FR14] The system shall provide different collections of predetermined fields based on the user type. [FR15] The system shall allow all users to edit each field in their profile, with the exception of unique identifiers, such as tax numbers for companies and universities, and unique IDs for students, which cannot be altered. [FR16] The system shall allow each field to be configured as public or private, provided that data is present in the respective field. [FR17] The system shall automatically configure all fields with no data as private. [FR18] The system shall allow all users to provide public hyperlinks to other platforms which the system shall display in a format that clearly indicates the platform to which the hyperlink leads to. [FR19] The system shall integrate OpenAI's ChatGPT 4.0 API into each field to provide AI-generated suggestions for profile improvements.
Components	AccountManager Application Protocol Interface (API) AuthenticationManager Database InstitutionalAuthenticator S&C Web App Instance

Requirements	**Internship Advertisement Creation and Management** [FR20] The system shall only allow company users whose data has been verified to create, update, and delete public internship advertisements, where the system shall save unfinished advertisements as draft. [FR21] The system shall allow the creation of new internship advertise- ments only if they possess the following fields: name, internship duration, short description, location, internship type, and the num- ber of stages in the application process. [FR22] The system shall integrate OpenAI's ChatGPT 4.0 API into each field of the internship advertisement creation process to provide AI-generated suggestions for enhancing the advertisement. [FR23] The system shall allow for numerous types of stages to be created, including but not limited to questionnaires and video interviews, which may be scheduled on external platforms. [FR24] Should the internship creation process be interrupted at any stage, the system shall save the unfinished internship advertisement as a draft. [FR25] After creation, the system shall allow company users to make up- dates to all fields of each internship advertisement they have created, with the exception of that internship advertisement's unique ID and publication timestamp.
Components	AdvertisementManager AccountManager Application Protocol Interface (API) Database InstitutionalAuthenticator InternshipManager RecommendationManager S&C Web App Instance

Requirements	**Internship Search, Application, and Management of Applications** [FR26] The system shall analyze the profile of the student user upon login and recommend 3 internship advertisements based on matching qualifications and background, and display these recommendations prominently to the student user. [FR27] The system shall allow student and university users to view all internship advertisements posted by any company on the platform, not just those they are applying to. Company users, however, will only be allowed to view and manage the advertisements they have created. [FR28] The system shall allow all student users to apply to internship ad- vertisements, after which their profiles will be automatically in- serted into the list of applicants under the Stage 1 category. [FR29] Upon a successfully submitted application, the system shall notify the advertisement's creator. [FR30] The system shall only allow company users who are creators of the internship advertisement to move candidates further along the selection stages. [FR31] The system shall allow the company user, at any given stage, to reject any candidate without mandating a reason. [FR32] Upon a change in status, be it a rejection or a move up in the selection stages, the system shall always notify the candidate of the change. [FR33] Upon reaching the final stage as designated by the company, the system shall automatically classify the candidate as accepted for the internship. [FR34] The system shall allow the company user to request the commence- ment of an internship from the candidate's host university. After the internship is approved by the host university, it shall commence on a predetermined date.
Components	AdvertisementManager AccountManager Application Protocol Interface (API) Database InstitutionalAuthenticator InternshipManager RecommendationManager S&C Web App Instance

Requirements	**Internship Monitoring, Complaint Management & Feedback System** [FR35] During any stage of the internship, the system shall allow both the candidate and the company user to submit a complaint to the host university, after which the university user shall be automatically notified. [FR36] After a complaint is received, the university may choose to resolve the complaint or to terminate the internship. The system shall allow for both choices to be submitted. [FR37] The system shall not allow the termination of an internship by any party except the university user. The system shall allow a termination only after a complaint has been submitted. [FR38] Upon the successful completion of an internship, the system shall request feedback from both the company user and the intern regarding the internship experience. Moreover, the system shall re- quest feedback from all three users types involved in the internship regarding their satisfaction with S&C. [FR39] The system shall request the university to review the submitted internship feedback. After the review, the system shall allow the university to either include the ratings in the profiles of both the student user and the company user, or keep the feedback private.
Components	AccountManager Application Protocol Interface (API) ComplaintManager FeedbackManager Database InstitutionalAuthenticator InternshipManager S&C Web App Instance

5 Implementation, Integration and Test Plan

5.1 Overview

This chapter provides a detailed explanation of how the platform described in the previous sections will be developed and tested. The primary goal of the testing process is to identify and resolve the majority of bugs present in the code written by the development team. This ensures the platform functions as intended and meets its requirements. Furthermore, section 5.3 will offer a comprehensive explanation of how the various components of the code are integrated and interact with one another to create a cohesive system. Meanwhile, section 5.2 will focus on outlining the most important strategies and methods used during the implementation phase to successfully develop the project.

5.2 Implementation Plan

The aim of this section is to describe the implementation strategies that will be used to develop, integrate, and test the various components of the Students & Companies application. The objective is to leverage the advantages of both the bottom-up and threads strategies

Using a threads strategy is effective because it makes progress visible to users and stakeholders, allowing for early feedback and engagement. This approach also minimizes the need for placeholder components, although it does make the integration process more complex.

A top-down methodology will be applied by first designing a basic structure or framework for the application. More advanced features, such as internship matching, application management, and feedback systems, will then be added incrementally as individual threads once they are validated.

This strategy enables different development teams to work concurrently on distinct tasks. Once a feature is fully developed and tested, it will be integrated into the overall software architecture. This approach ensures that validated components contribute to building a robust and functional application while maintaining steady progress and stakeholder visibility.

5.2.1 Features Identification

The features to implement are described starting from the requirements. Some requirements involve the development of new components, while others only require minor changes. Below is a summary of the most important features identified from the functional requirements.

[F1] Student and Company Registration

This feature enables students and companies to register on the platform. For students, the system verifies their institutional accounts, while for companies, email verification and additional validation steps, such as providing a tax certificate, are required. These operations are fundamental for ensuring user identity and data integrity.

[F2] Internship Advertisement Creation and Management

This core feature allows verified company users to create, edit, and manage internship advertisements. Key functionalities include specifying internship details, saving drafts, and integrating AI-generated suggestions to enhance the advertisements. This feature directly supports the primary goal of matching students with suitable internships and requires thorough testing to ensure reliability.

[F3] Internship Search and Application

This feature allows students to search for internships and apply directly through the platform. The system recommends internships based on the student's profile and qualifications. Upon application, the student's profile is added to the list of applicants, and notifications are sent to the relevant company users. This feature is critical for fostering engagement between students and companies.

[F4] Internship Monitoring and Complaint Management

This feature allows students, companies, and universities to monitor ongoing internships and submit complaints if needed. The system notifies the relevant parties and allows universities to resolve issues or terminate internships. Testing this functionality is essential to ensure smooth conflict resolution and maintain platform trustworthiness.

[F5] Feedback Collection and Profile Integration

After the completion of internships, feedback is collected from students, companies, and universities. The system allows universities to review and optionally integrate ratings into user profiles. This feature supports continuous improvement of the platform and enhances user profiles for future opportunities.

The majority of these features require seamless interaction between the client and server. Therefore, the dispatcher component must be developed at the very beginning to support effective communication and functionality across the platform.

5.3 Component Integration and Testing

In this section it is detailed which component is implemented in every stage of the development process and how the different components are implemented and tested.

[F1] Student and Company Registration

Allows students and companies to register on the platform.

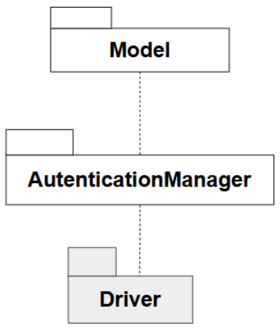


Figure 30: Student and Company Registration

[F2] Internship Advertisement Creation and Management

Verified company users can create, edit, and delete internship advertisements. Incorporates AI-generated suggestions to improve ad quality (if you choose to utilize RecommendationManager).

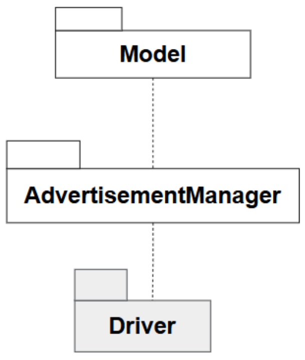


Figure 31: Internship Advertisement Creation and Management

[F3] Internship Search and Application

Students can search available internship ads and apply directly through the platform. The system recommends internships based on the student's profile and qualifications. Once a student applies, their application is recorded, and the company is notified.

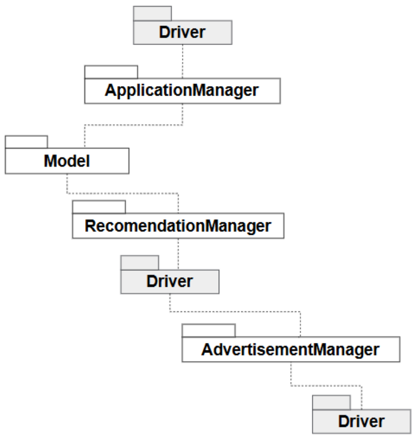


Figure 32: Internship Search and Application

[F4] Internship Monitoring and Complaint Management

Allows students, companies, and universities to monitor ongoing internships. Users can file a complaint if issues arise.

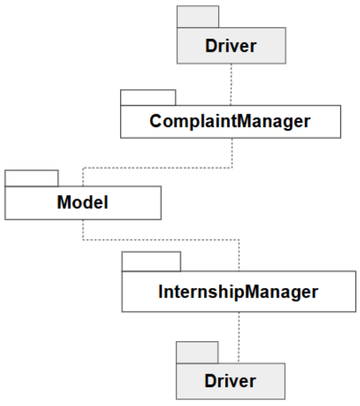


Figure 33: Internship Monitoring and Complaint Management

[F5] Feedback Collection and Profile Integration

Collects feedback from students, companies, and potentially universities after an internship ends.

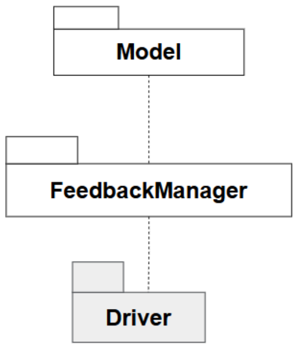


Figure 34: Feedback Collection and Profile Integration

5.4 System testing

S&C is a platform designed to match students with internships posted by various companies. To ensure that the platform delivers the intended functionalities, it must be tested thoroughly. During development, each individual component is tested in isolation; however, only when these components are integrated into the overall architecture can the entire system be tested as a whole. This holistic testing is critical for verifying both functional and non-functional requirements.

Once smaller modules have been validated, they are incrementally added to the platform's software architecture. After the entire architecture is built, system testing can begin in full. The primary objective of this phase is to confirm that S&C meets every requirement as documented in the RASD (Requirements Analysis and Specification Document). In this process, all stakeholders involved in software development, including, but not limited to, developers, project managers, student representatives, and company representatives, contribute to testing. Below is an outline of the main testing strategies we employ:

- **Functional Testing:** This is performed to verify that all functional requirements are satisfied. The most effective way to conduct this test is to run S&C according to the use cases described in the RASD and check whether each function (e.g., internship listing, student application, company account creation) is working as specified.

- **Performance Testing:** The main goal of this testing is to identify potential bottlenecks that might affect response time, scalability, and throughput. We also watch for inefficient algorithms or configurations in hardware and network usage. Performance tests include stress scenarios with large numbers of internship postings and concurrent student applications, helping us uncover optimization possibilities.

- **Usability Testing:** This type of testing observes real users, students and companies, as they navigate and use S&C. We gather feedback on how intuitive the interface is, whether the users can easily find and apply to internships, and if companies can smoothly post or edit internship advertisements. The focus here is on the human interaction aspect.

- **Load Testing:** In load testing, we deliberately place S&C under heavy usage scenarios to ensure that it can handle spikes of activities, such as large numbers of concurrent logins or applications submitted at the same time. This helps reveal any potential memory leaks or resource mismanagement issues. By simulating peak load conditions, we can assess how long the system can remain stable and responsive.

- **Stress Testing:** Stress tests push the system beyond its expected capacity to see how it recovers when resources become saturated. We may artificially inflate the number of users or reduce available system resources to see if S&C fails gracefully and recovers quickly. These extreme conditions help us confirm that worst-case scenarios are handled in a controlled manner.

5.5 Additional specifications on testing

Throughout development, it is essential to gather continuous feedback from both users and stakeholders. Each time a new feature (e.g., AI-driven internship suggestions, improved matching algorithms) is implemented, we incorporate feedback to refine the system further.

During an alpha test, a select group of users, often representatives from universities or pilot companies, try out the platform in a controlled environment. Their impressions help us measure the platform's ease of use and identify initial flaws. It can be particularly valuable to involve individuals who understand the internship process deeply, such as career services staff or HR personnel, as they can provide specialized feedback on whether S&C meets real-world needs.

The alpha test is also instrumental in identifying malfunctions or major design issues before moving on to the beta test, in which a broader set of real users, students from multiple institutions and additional companies, access the platform under more realistic conditions. This phase helps detect remaining bugs and usability issues in what approximates a production environment.

Finally, once S&C is deployed, continuous monitoring remains crucial. Certain logs and usage statistics (e.g., large spikes in the number of applications or error rates in authentication) should be automatically collected and made available to developers. These logs allow for ongoing debugging and performance tuning, ensuring that the platform remains reliable and efficient for all participants as usage grows and additional features are introduced.

6 Effort spent

The time tables written below represent an approximation of the effort spent for the creating each specific section of this document. These times for producing this document are based on the personal perception of the team members.

Student	**Section 1**	**Section 2**	**Section 3**	**Section 4**	**Section 5**	**Total Hours**
Veljko Tatalović	5	5	6	5	8	29
Edin Žiga	4	15	6	2	2	29
Nikola Dimić	2	2	6	10	10	30

Table 1: Effort Spent by Students

References

[1] C. Larman. *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and the Unified Process*. Safari electronic books. Prentice Hall PTR, 2002.

[2] MToC. Building a 3-tier architecture on aws, Nov 2023.