Software Engineering 2 project Abate Singe

Design Document

Deliverable:	DD
Title:	Design Document
Authors:	Kevin Pio Abate, Gabriele Lorenzo Singe
Version:	1.0
Date:	16-December-2024
Download page:	https://github.com/Keevin16/AbateSinge

Contents

Table of Contents . . . 3

List of Figures . . . 5

List of Tables . . . 5

1 Introduction . . . 6
- 1.1 Purpose . . . 6
- 1.2 Scope . . . 6
- 1.3 Definitions, acronyms . . . 6
  - 1.3.1 Definitions . . . 6
  - 1.3.2 Acronyms . . . 6
- 1.4 Document Structure . . . 6

2 Architectural Design . . . 7
- 2.1 Overview: high-level components and interactions . . . 7
- 2.2 Component view . . . 7
  - 2.2.1 Entity Manager . . . 8
  - 2.2.2 S&C Task Manager . . . 8
  - 2.2.3 Logical Description of Data . . . 9
- 2.3 Deployment view . . . 10
- 2.4 Runtime view . . . 11
  - 2.4.1 User sign up . . . 11
  - 2.4.2 User log in . . . 12
  - 2.4.3 User search for other user . . . 13
  - 2.4.4 User view other user's profile . . . 14
  - 2.4.5 User modify personal informal . . . 15
  - 2.4.6 Student views internship list . . . 16
  - 2.4.7 User views internship's details . . . 17
  - 2.4.8 User searches an internship . . . 18
  - 2.4.9 Company creates a internship . . . 19
  - 2.4.10 Student apply to an internship . . . 20
  - 2.4.11 Company invites a student to be a candidate . . . 21
  - 2.4.12 Company views list of students applications . . . 22
  - 2.4.13 Company decides to accept students for the evaluation . . . 23
  - 2.4.14 Student decides to accept a company invitation . . . 24
  - 2.4.15 Company manually closes internship open phase . . . 25
  - 2.4.16 Company views students responses to questionnaire . . . 26
  - 2.4.17 Company views list of evaluated students . . . 27
  - 2.4.18 Company selects a student for the internship . . . 28
  - 2.4.19 Company ends evaluation phase without selecting a student . . . 29
  - 2.4.20 User submits updates or complaints for ongoing internship . . . 29
  - 2.4.21 User submits feedback for completed internship . . . 30
  - 2.4.22 Company accesses its internships . . . 31
  - 2.4.23 Student views enrolled internships . . . 32
  - 2.4.24 Student views evaluation questionnaire . . . 33
  - 2.4.25 Student submits evaluation questionnaire responses . . . 34
- 2.5 Component interfaces . . . 35
  - 2.5.1 API endpoints . . . 35

2.6 Selected architectural styles and patterns . . . 38
2.7 Other design decisions . . . 38

3 User Interface Design . . . 39

4 Requirements Traceability . . . 43

5 Implementation, Integration and Test Plan . . . 47
5.1 Overview . . . 47
5.2 Implementation and Integration Plan . . . 47
5.2.1 Server Side . . . 47
5.2.2 Client Side . . . 47
5.2.3 Full testing . . . 47
5.3 System testing . . . 48

6 Effort Spent . . . 49

References . . . 50

List of Figures

1 High level description diagram . . . 7
2 Component diagram of S&C system . . . 7
3 Component diagram of S&C system . . . 8
4 Component diagram of S&C system . . . 9
5 Component diagram of S&C system . . . 10
6 Sign Up . . . 11
7 Log in . . . 12
8 Sign Up . . . 13
9 User view other user's profile . . . 14
10 User modify personal informal . . . 15
11 Student views internship list . . . 16
12 User views internship's details . . . 17
13 User searches an internship . . . 18
14 Company creates a internship . . . 19
15 Student apply to an internship . . . 20
16 Company invites a student to be a candidate . . . 21
17 Company views list of students applications . . . 22
18 Company decides to accept students for the evaluation . . . 23
19 Student decides to accept a company invitation . . . 24
20 Company manually closes internship open phase . . . 25
21 Company views students responses to questionnaire . . . 26
22 Company views list of evaluated students . . . 27
23 Company selects a student for the internship . . . 28
24 User submits updates or complaints for ongoing internship . . . 29
25 User submits feedback for completed internship . . . 30
26 Company accesses its internships . . . 31
27 Student views enrolled internships . . . 32
28 Student views evaluation questionnaire . . . 33
29 Student submits evaluation questionnaire responses . . . 34
30 Home page . . . 39
31 Sign Up . . . 39
32 Log In . . . 40
33 Modify your profile . . . 40
34 Home page . . . 41
35 Search profiles . . . 41
36 Create internship . . . 42
37 Search internships . . . 42

List of Tables

3 Time spent on each section by contributors. . . . 49

1 Introduction

1.1 Purpose

1.2 Scope

Students&Companies (S&C) is a platform designed to facilitate the matching of university students seeking internships with companies offering such opportunities. The platform aligns students and companies based on the students' resumes, completed projects, and the terms provided by the companies. Additionally, S&C advertises internships offered by companies, allowing students to manually search for positions that align with their interests. To further enhance the user experience, the platform employs a recommendation system, which proactively suggests new internship opportunities tailored to the interests and qualifications of each student,also suggesting to companies new students with matching qualification for the internship.
The platform is implemented through a three-tier architecture.

1.3 Definitions, acronyms

1.3.1 Definitions

User: It is used as a generalization to represent both company and student accounts

1.3.2 Acronyms

- **S&C** - Students&Companies
- **API** - Application Programming Interface
- **DBMS** - DataBase Management System
- **RASD** - Requirement Analysis and Specifications Document
- **UML** - Unified Modeling Language

1.4 Document Structure

- **Overview**:
  Provides a general description of the system's main aspects, focusing on the key components and their interactions without delving into the system's internal workings.
- **Component View**:
  Offers a detailed breakdown of the elements comprising the components introduced in the Overview, with particular emphasis on the server components.
- **Component Interfaces**:
  Explains how various components within the system interact with each other and outlines the interfaces used to facilitate interaction with external components.
- **Runtime View**: Uses UML diagrams to illustrate how components interact during execution, in relation to specific use cases.

2 Architectural Design

This section aimed to present and analyze the architecture of the S&C system using a top-down approach. We began by introducing the overall architecture, followed by a diagram detailing the system's components, with a focus on the ckbSP subcomponent. An ER diagram was then used to illustrate the system's logical data structure, and the deployment view was presented, highlighting the involved layers and tiers. Additionally, sequence diagrams were employed to depict key runtime interactions, while class diagrams provided an analysis of the component interfaces. Finally, we discussed the architectural design decisions and the rationale behind them.

2.1 Overview: high-level components and interactions

The figure below provides an overview of the components that comprise the system. In this document, the user interface and the client (e.g., the browser) will be referred to as the Frontend, while the application layer and the data layer will be referred to as the Backend.

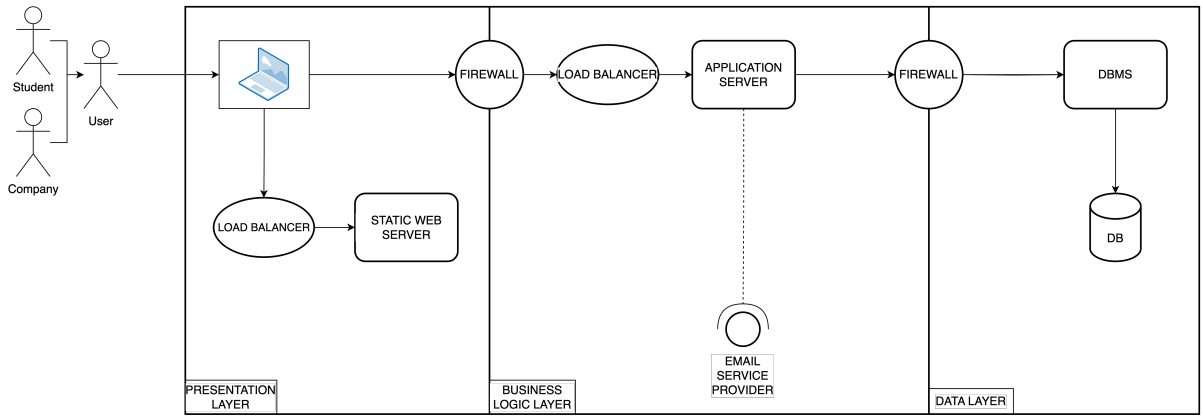

Figure 1: High level description diagram

The application servers are designed to be stateless according to REST standards, and the system includes firewalls to enhance security.

2.2 Component view

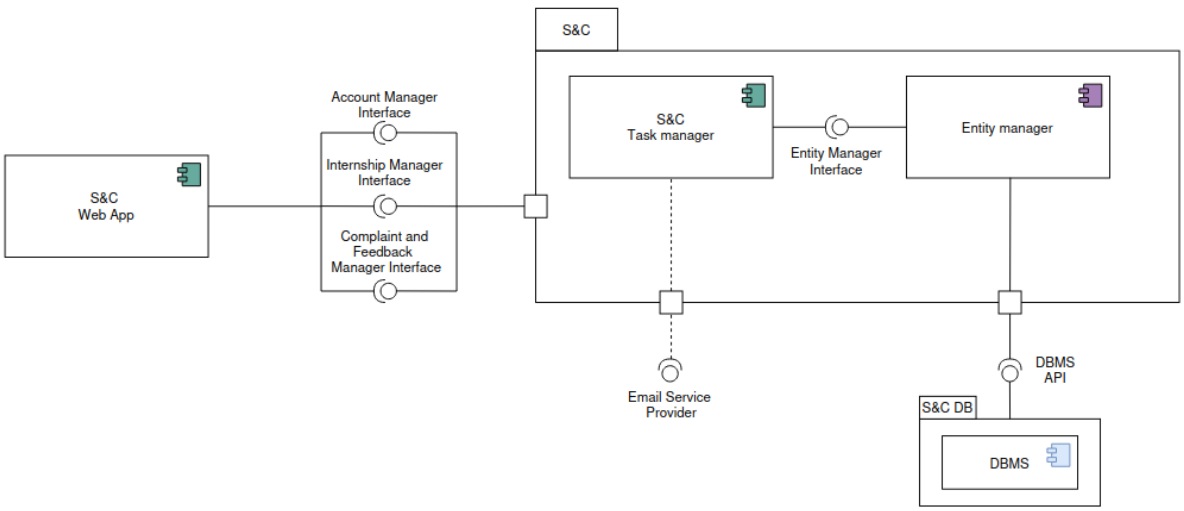

Figure 2: Component diagram of S&C system

2.2.1 Entity Manager

This component handles communication with the Database Management System (DBMS). It implements the adapter design pattern, enabling other components to interact with the DBMS without requiring them to write SQL code directly.

2.2.2 S&C Task Manager

The S&C Task Manager subsystem is responsible for executing S&C's logic. It facilitates all the system's functionalities and coordinates interactions with other components.

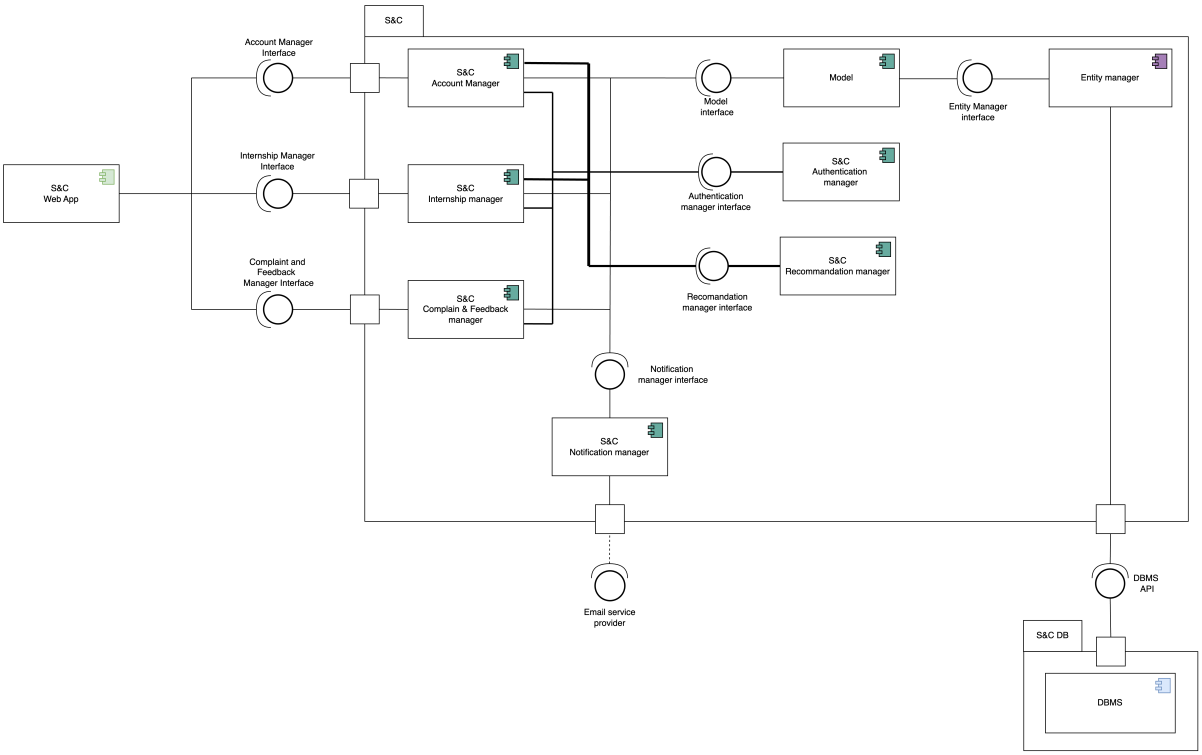

Figure 3: Component diagram of S&C system

1. **Account Manager:** This component manages the operations required to handle user accounts, such as updating personal information or modifying a CV in the case of students. It also enables viewing other users' profiles. Additionally, it allows user login and sign up by interfacing with the Authentication Manager.
2. **Internship Manager:** This component is responsible for handling the operations required to create an internship, view its details, manage student evaluations, and perform the final selection process.
3. **Complaint and Feedback Manager:** This components handles the complains, updates and final feedback from the student and the company during the active internship.
4. **Authentication Manager:** This component manage sign up, login and logout tasks.
5. **Notification Manager:** This component handles the notifications delivery.
6. **Recommendation Manager:** This component sends recommendations whenever a CV is updated, notifying the companies of interested students, or when a new internship is published, sending notifications to matching students.
7. **Model:** This component facilitates the interaction with the entity manager.

2.2.3 Logical Description of Data

In this section, the ER diagram of S&C's data is presented. An ER diagram is a graphical representation of the database structure, illustrating the relationships between entities and their attributes.

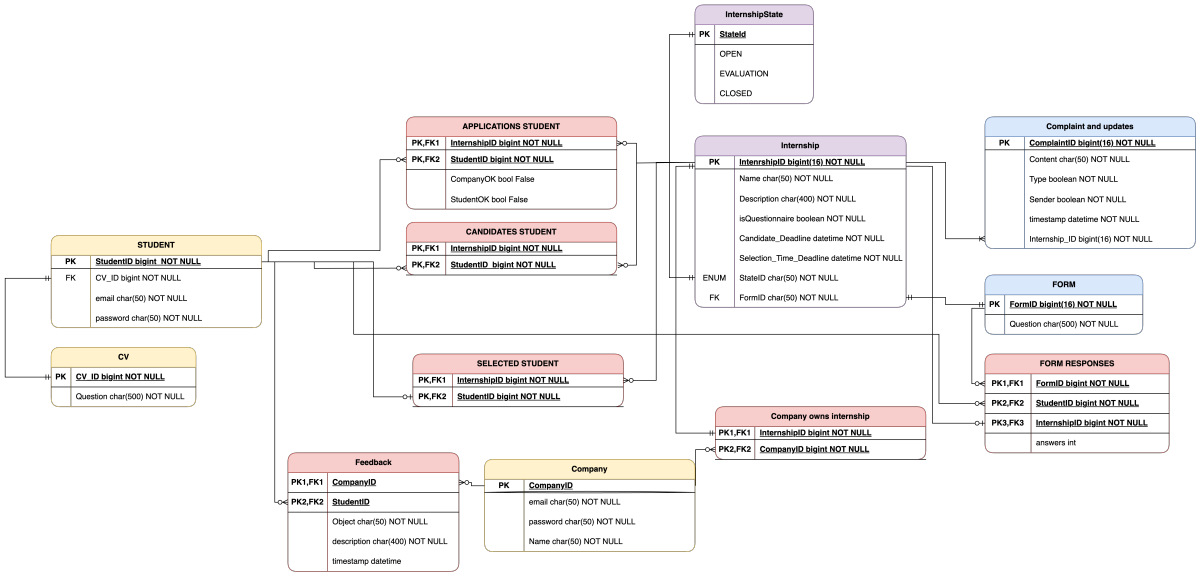

Figure 4: Component diagram of S&C system

Additional constraints must be established in the database management system to ensure data coherence and consistency.
When the candidate's deadline has finished (Open status finished), it is necessary to check that the list of candidates no longer changes.
The selected student must be in the list of candidates when the status transitions from Evaluation to Close.
The list of evaluations student can't be empty when passing to the evaluation phase.
An internship cannot be owned by two different companies, and a CV must be owned by only one student.

2.3 Deployment view

The system utilizes a 3-tier architecture hosted on cloud infrastructure.
The entire architecture is hosted with a cloud provider, offering several benefits over traditional in-house hosting, such as:

1. **Scalability and Flexibility** - the ability to scale resources like virtual machines, performance cores, or memory as needed, coupled with load balancing services that help the servers adjust to changes in traffic or workload.

2. **Cost-efficiency** - the pay-as-you-go pricing model of cloud providers allows payment only for the resources actually used, thus reducing overall costs.

3. **Security** - services like real-time monitoring and firewalls ensure the protection of the application server from data breaches and other security risks.

These features make cloud providers an ideal choice for hosting large, high-traffic applications. The selected cloud provider must offer all these capabilities to meet our requirements.
The web server is located in the DMZ and serves as the primary interface for user interactions, handling HTTP requests and serving web pages. The application server runs the core application logic, processes user requests, and manages business operations. Load balancers are deployed to distribute incoming traffic evenly across multiple Web and Application Servers.

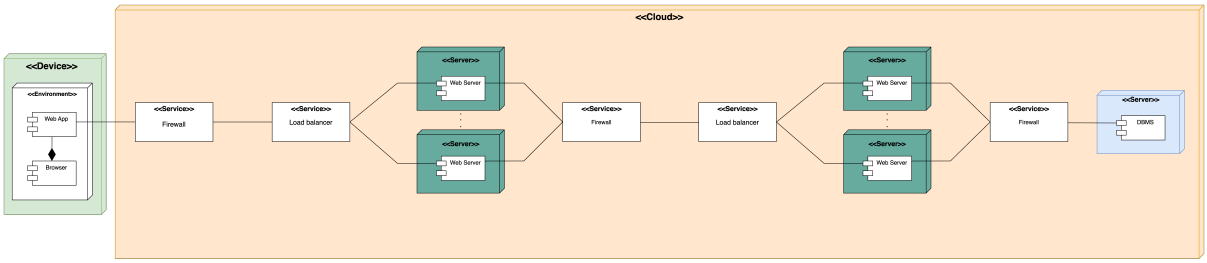

Figure 5: Component diagram of S&C system

2.4 Runtime view

2.4.1 User sign up

The diagram presented below illustrates the sign-up process of a user. First, the user fills out the sign-up form with the email, password, and account type. Then, once the request is submitted through the appropriate API call, the Account Manager component processes the request.
If the request parameters are valid, the Account Manager triggers the verification email via the Email Service Provider.

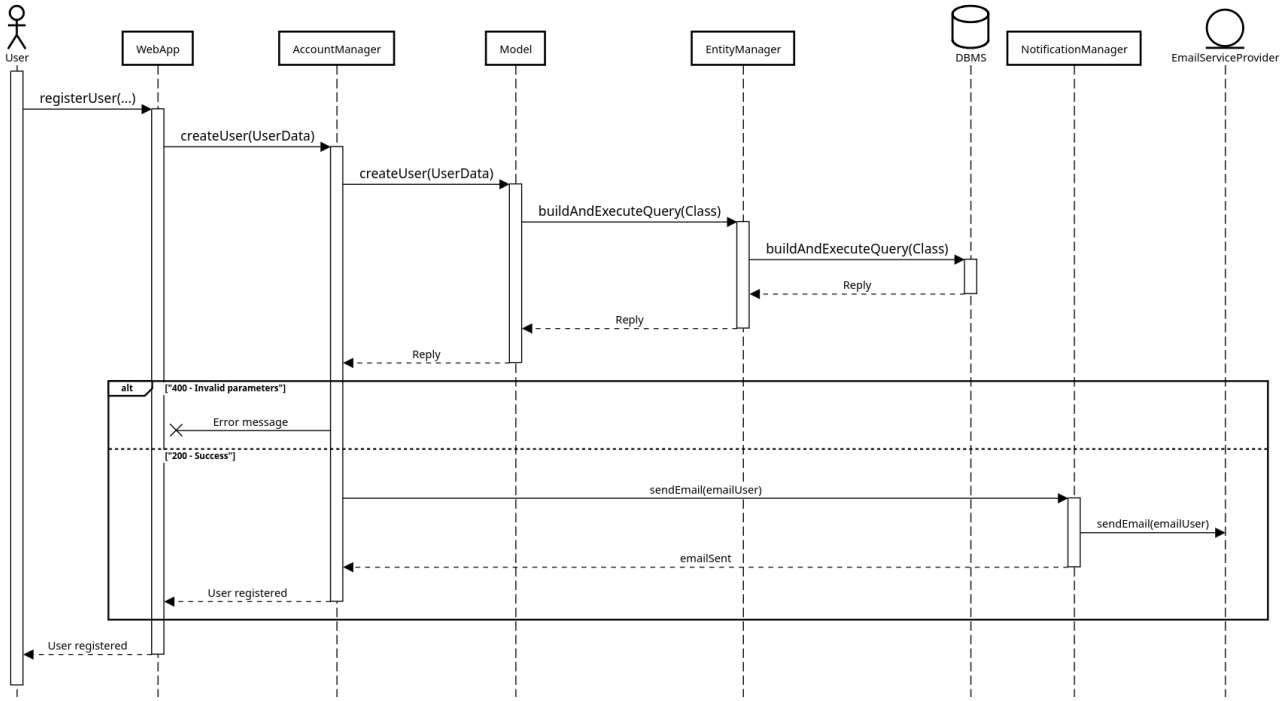

Figure 6: Sign Up

2.4.2 User log in

The diagram presented below shows the process of a user logging into their account. First, the user fills out the login form with their credentials (email and password). The user must have verified his account before trying to login. Then, once the request is submitted through the appropriate API call, the Account Manager component processes the request. If the credentials are valid, the user is provided with an authentication token generated by the Authentication Manager through the Account Manager. Otherwise, the process must be repeated.

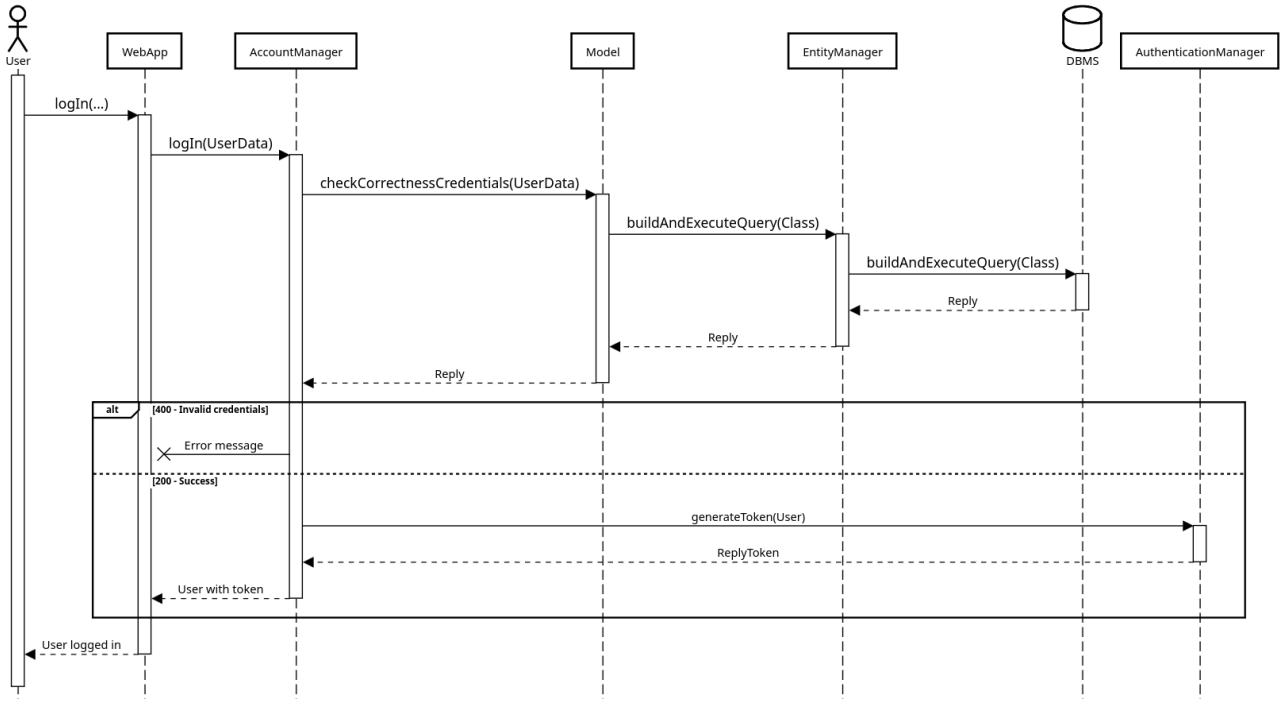

Figure 7: Log in

2.4.3 User search for other user

The diagram below shows the process of a user searching for an user using a keyword. After successful authentication, the Account Manager interacts with the Model to search for user based on the provided keyword. This triggers the Entity Manager to execute a query on the DBMS. The user list data flows back through the components, starting from the Entity Manager to the Account Manager, and ends with the Account Manager sending the list of internships to the WebApp.

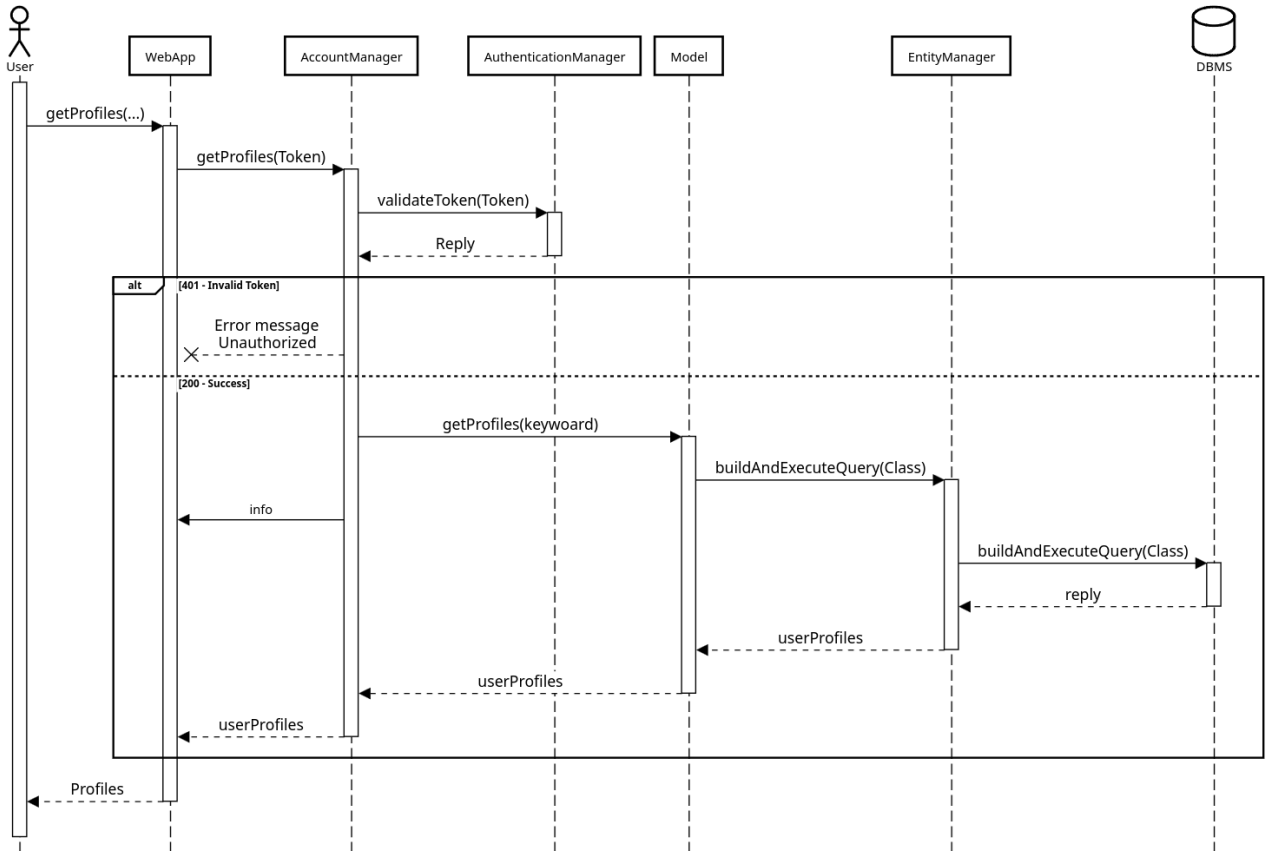

Figure 8: Sign Up

2.4.4 User view other user's profile

The diagram below illustrates the process of a user viewing the details of an user. After successfully logging in, the Account Manager interacts with the Model to retrieve the account details using the provided user ID. This process involves the Entity Manager querying the DBMS for the relevant information. If the Account ID is invalid, the Entity Manager returns a null response. Then the response is passed back through the Account Manager to the Web App.

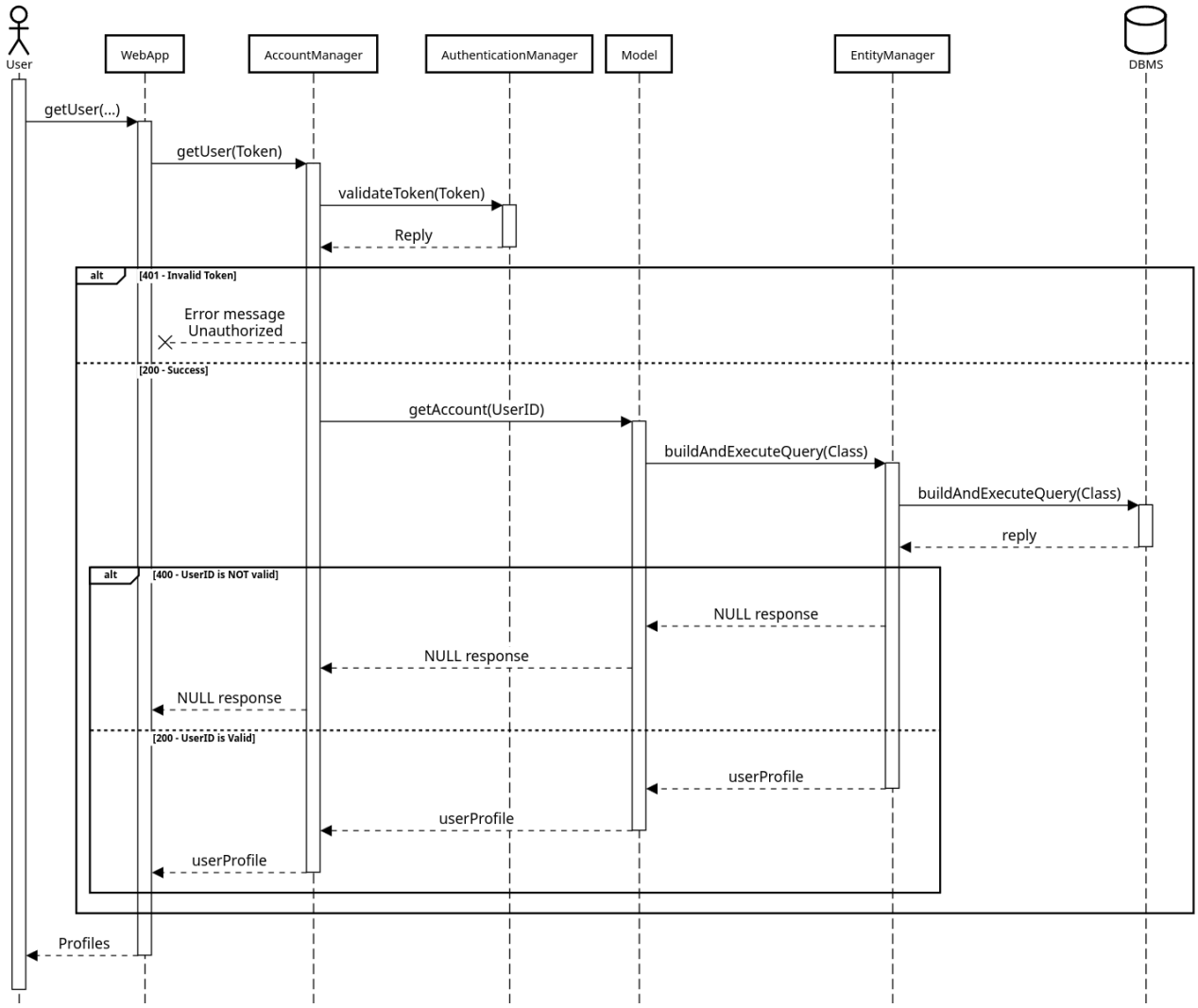

Figure 9: User view other user's profile

2.4.5 User modify personal informal

The diagram below illustrates the process of a user modifying their personal information.
The user navigate through the modify profile button and modify his personal data. Then, as the request is submitted through the appropriate API call, the Account Manager processes the request. Upon successful token validation and role authorization, the Account Manager communicates with the Model to modify the account using the provided data. Error handling mechanisms are implemented to manage cases where invalid data is provided. This interaction triggers the Entity Manager to execute a query on the DBMS and send also the information to the recommendation system. If the profile data is invalid, the Entity Manager returns a null response, which is passed back through the Account Manager to the WebApp. In the case of successful profile modification, the success signal flows back through the components, ending with the Account Manager sending the success notification to the WebApp.

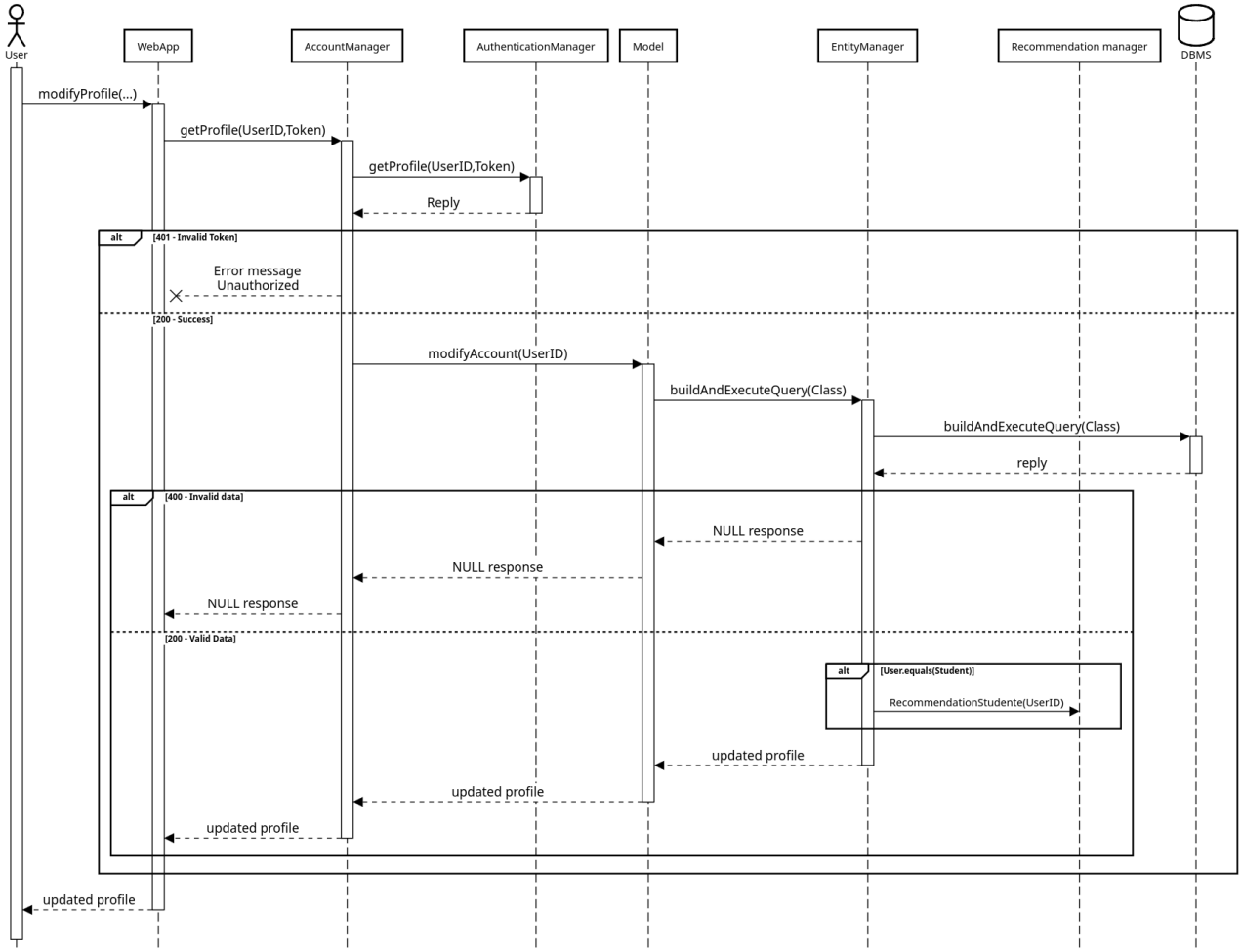

Figure 10: User modify personal informal

2.4.6 Student views internship list

The diagram presented below shows the process of a student viewing the list of all internships ever created. Upon successful validation, the Internship Manager interacts with the Model to retrieve internship data using the Entity Manager, which queries the DBMS for the information.

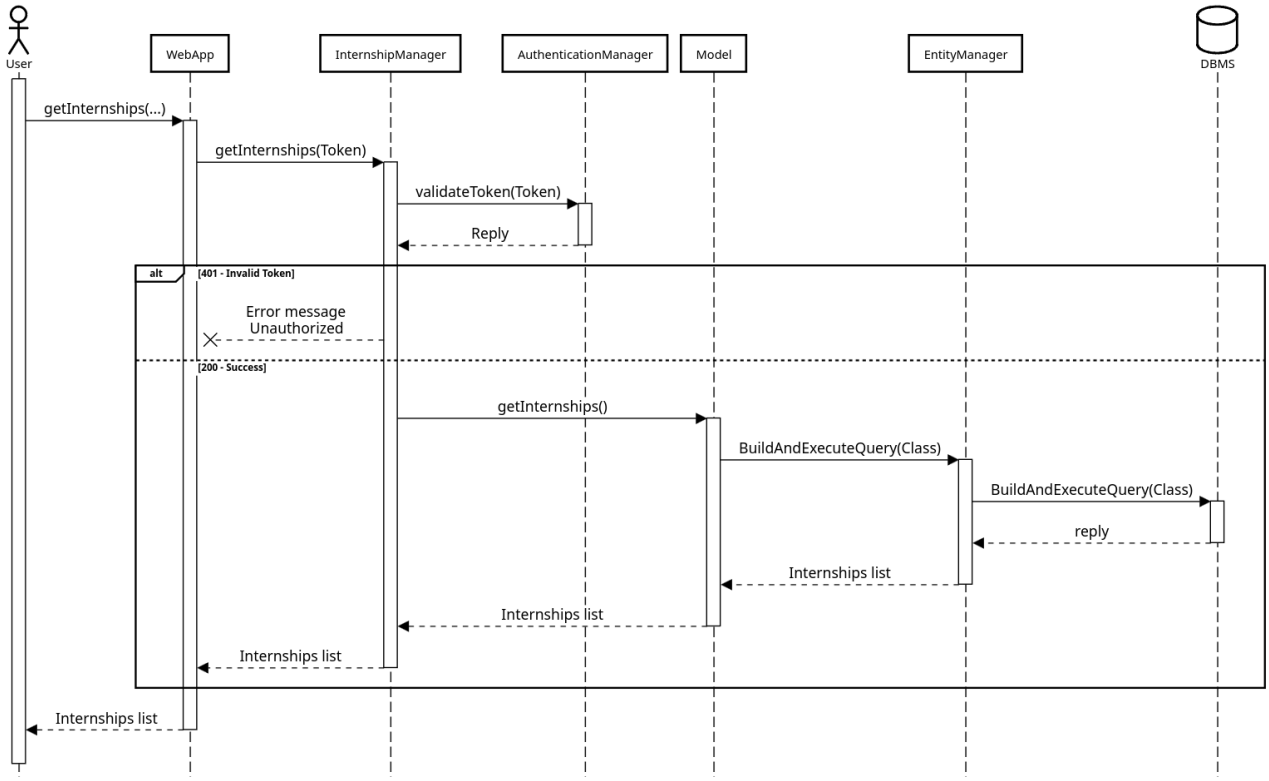

Figure 11: Student views internship list

2.4.7 User views internship's details

The diagram below illustrates the process of a user viewing the details of an internship. After successfully logging in, the Internship Manager interacts with the Model to retrieve the internship details using the provided Internship ID. This process involves the Entity Manager querying the DBMS for the relevant information. If the Internship ID is invalid, the Entity Manager returns a null response. Then the response is passed back through the Internship Manager to the Web App.

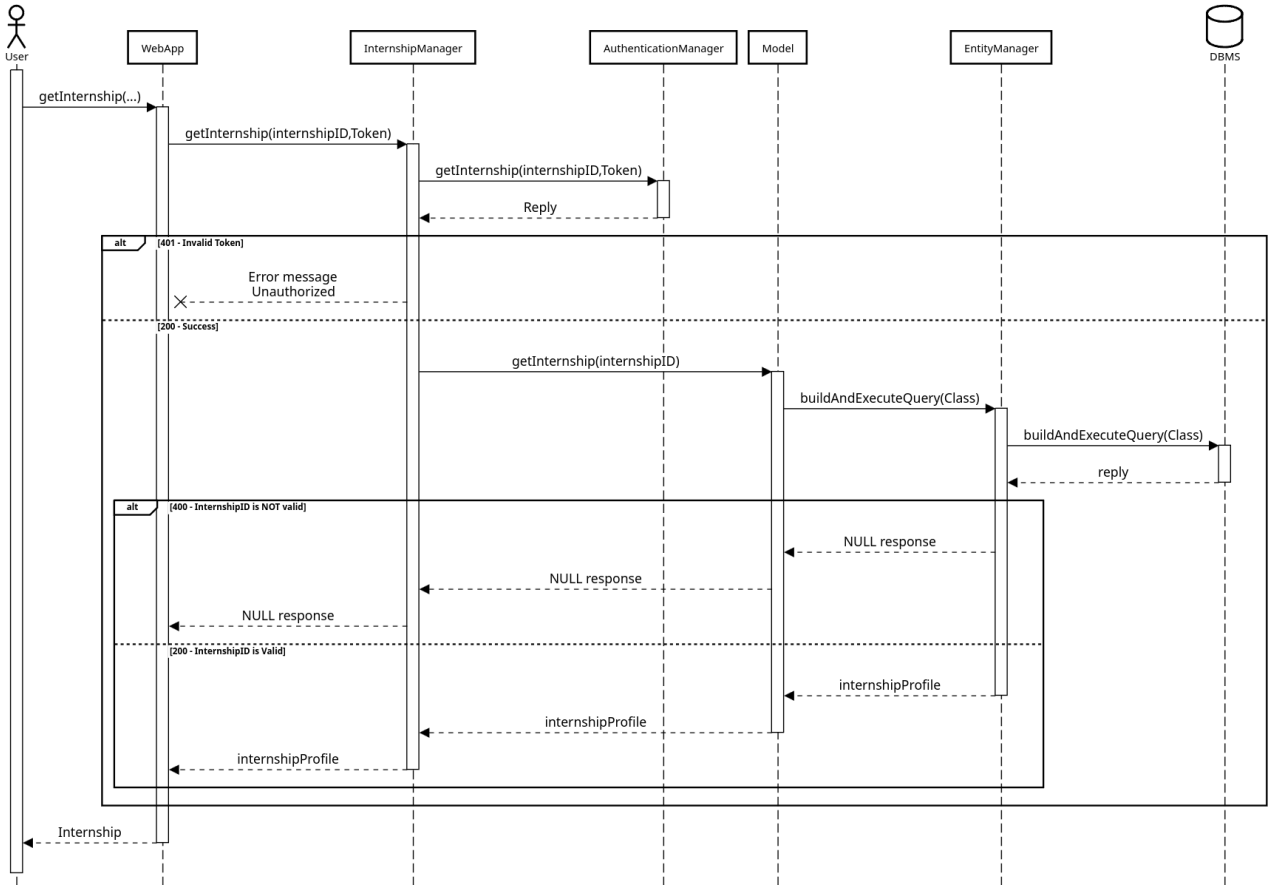

Figure 12: User views internship's details

2.4.8 User searches an internship

The diagram below shows the process of a user searching for an internship using a keyword. After successful authentication, the Internship Manager interacts with the Model to search for internships based on the provided keyword. This triggers the Entity Manager to execute a query on the DBMS. The internship data flows back through the components, starting from the Entity Manager to the Internship Manager, and ends with the Internship Manager sending the list of internships to the WebApp.

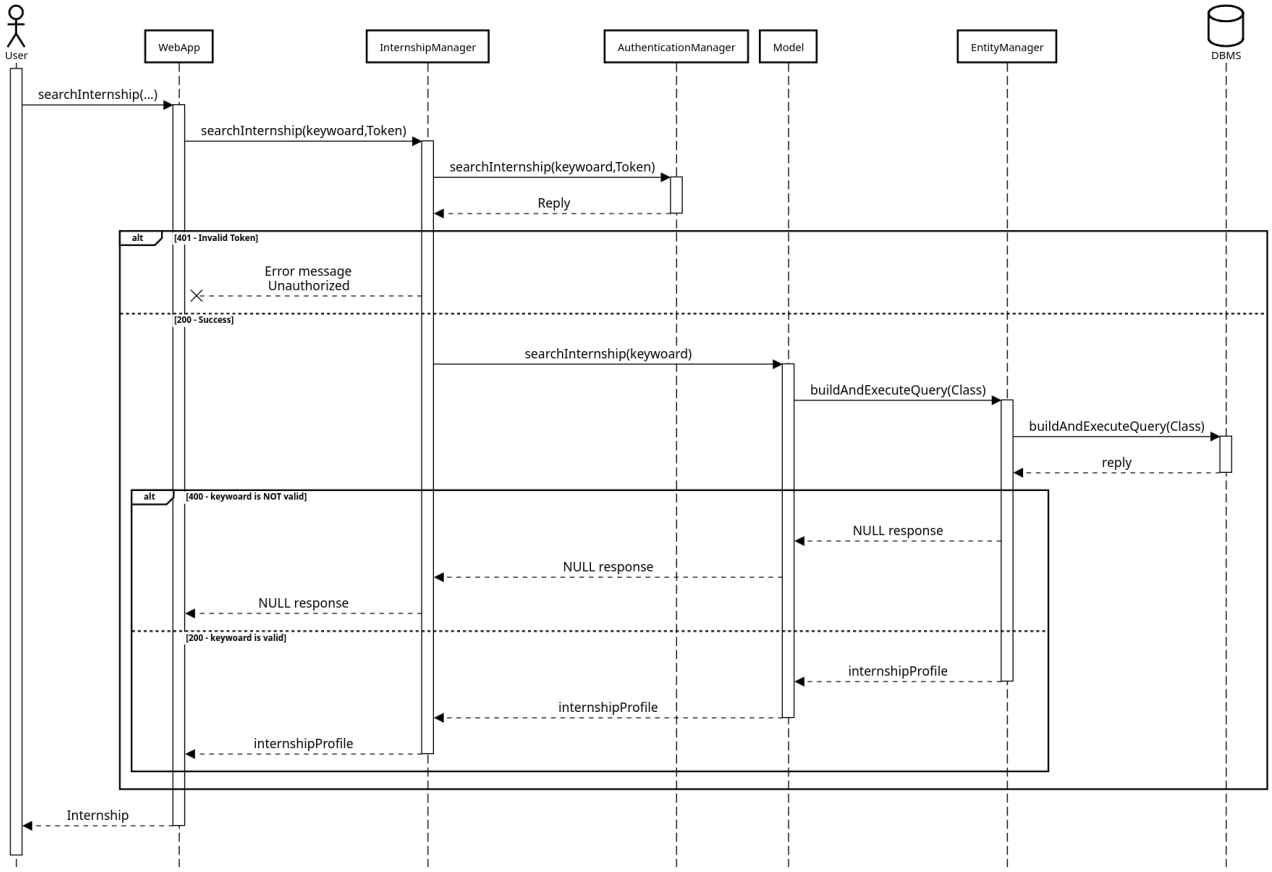

Figure 13: User searches an internship

2.4.9 Company creates a internship

The diagram below illustrates the process of a company creating an internship.
First, the company fills out the form with all the required information, such as the term offer, the description of the internship, the required skills, type of evaluation (eventually selection of questionnaire) and time constraints. Then, as the request is submitted through the appropriate API call, the Internship Manager processes the request. Upon successful token validation and role authorization, the Internship Manager communicates with the Model to create the internship using the provided data and the company's ID. This interaction triggers the Entity Manager to execute a query on the DBMS. If the internship data is invalid, the Entity Manager returns a null response, which is passed back through the Internship Manager to the WebApp. In the case of successful internship creation, the success signal flows back through the components, ending with the Internship Manager sending the success notification to the WebApp.

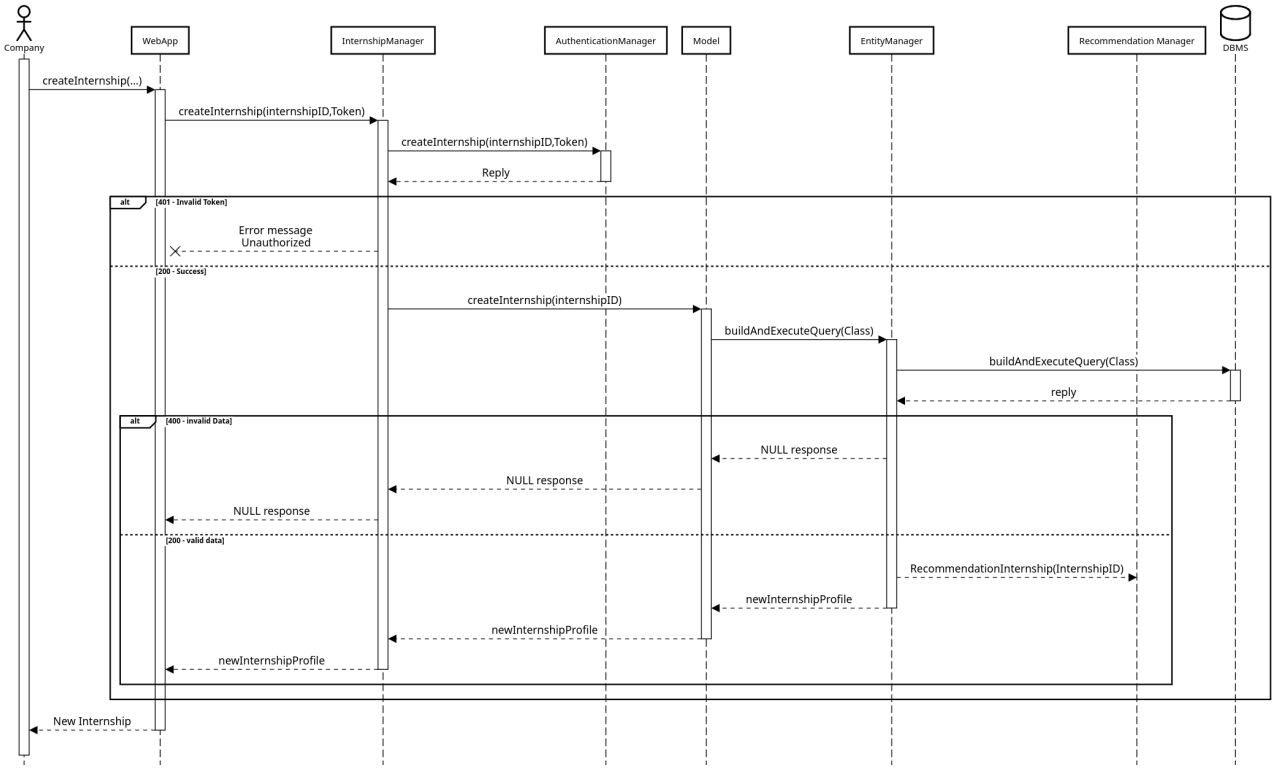

Figure 14: Company creates a internship

2.4.10 Student apply to an internship

The diagram below shows the process of a student applying to an internship.
The student search for an internship, view its details and decide to apply through the apply button. Then, as the request is submitted through the appropriate API call, the Internship Manager processes the request. Upon successful token validation and role authorization, the Internship Manager communicates with the Model to create the application of the student with the ok of the student. This interaction triggers the Entity Manager to execute a query on the DBMS. If the application is invalid, the Entity Manager returns a null response, which is passed back through the Internship Manager to the WebApp. In the case of a successful action, the success notification flows back through the components, ending with the Internship Manager sending the success signal to the WebApp.

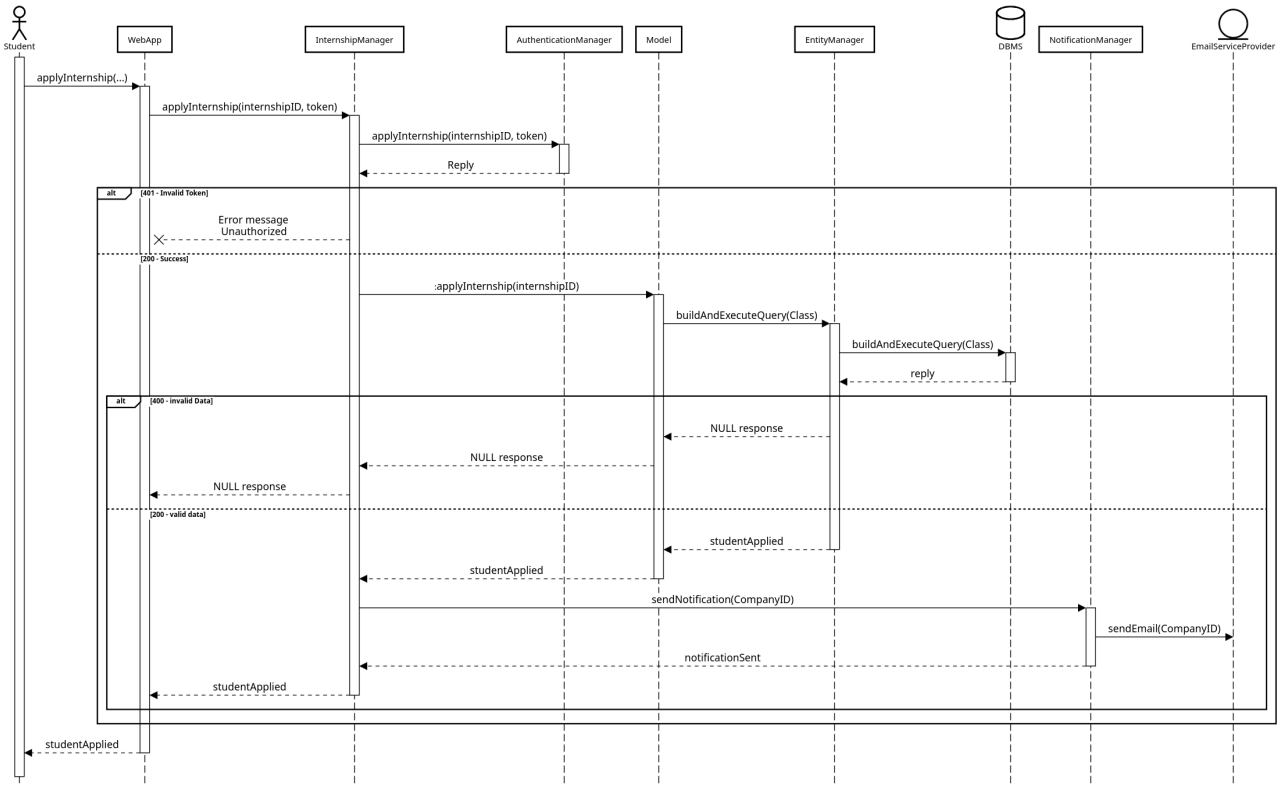

Figure 15: Student apply to an internship

2.4.11 Company invites a student to be a candidate

The diagram below illustrates the process of a company inviting a student to become a candidate.
The company searches for a student, views their details, and decides to invite them as a candidate by clicking the "Invite" button. Once the request is submitted through the appropriate API call, the Internship Manager processes the request. After successful token validation and role authorization, the Internship Manager communicates with the Model to create the student's application with the company's approval. This interaction triggers the Entity Manager to execute a query on the DBMS. If the application is invalid, the Entity Manager returns a null response, which is relayed back through the Internship Manager to the WebApp. If the action is successful, a success notification flows through the components, culminating in the Internship Manager sending a success signal to the WebApp.

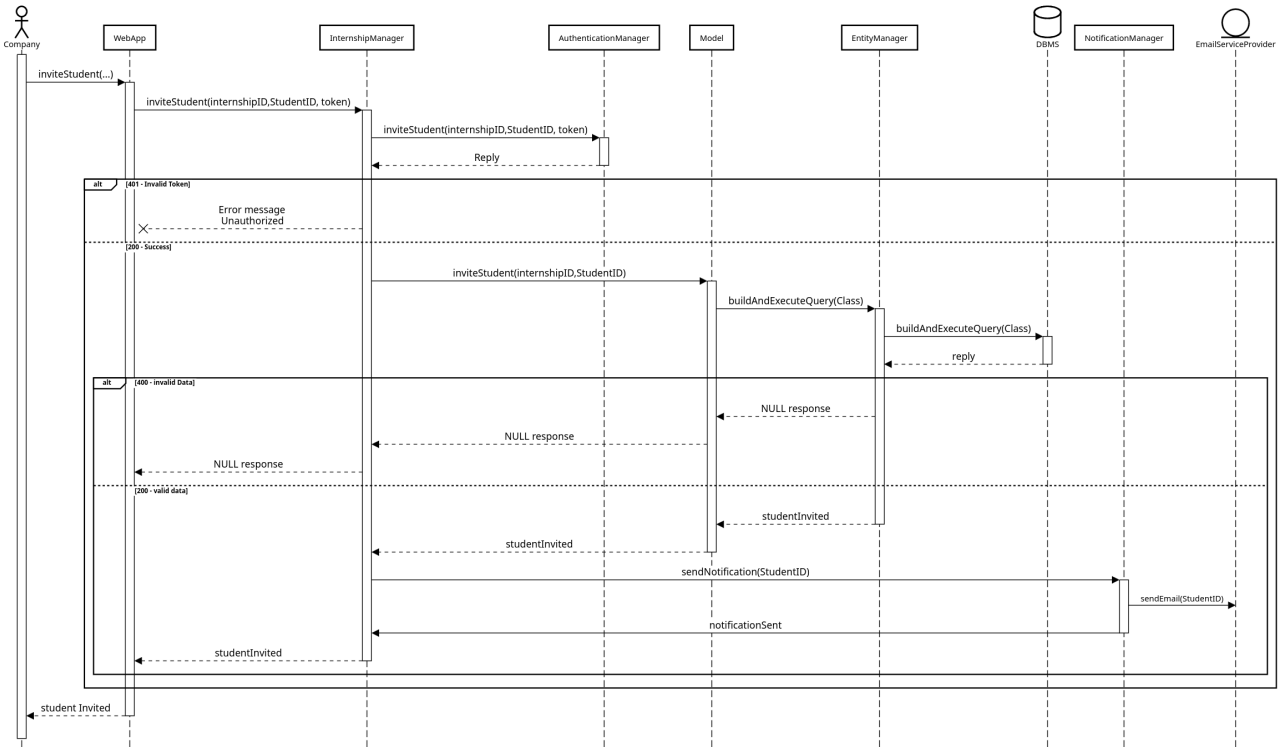

Figure 16: Company invites a student to be a candidate

2.4.12 Company views list of students applications

The diagram below illustrates the process of a company viewing the list of applications for one of their internships.
The company navigates to the active internships and selects the one for which they want to review the applications. After successful authentication, the Internship Manager communicates with the Model to retrieve the internship matching the ID and fetch the list of applications. This triggers the Entity Manager to execute a query on the DBMS. The internship data flows back through the components, starting from the Entity Manager to the Internship Manager, and ultimately ending with the Internship Manager sending the list of applications to the WebApp.

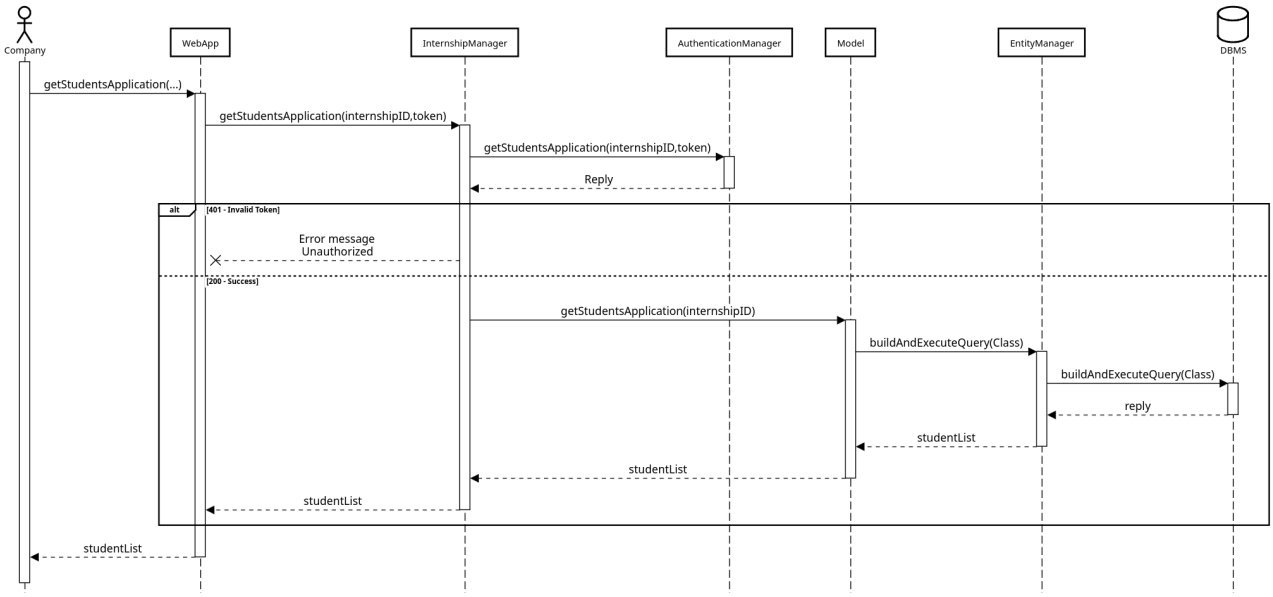

Figure 17: Company views list of students applications

2.4.13 Company decides to accept students for the evaluation

The diagram below illustrates the process of a company accepting a student for the evaluation phase. The company retrieves the list of student applications, with "Confirm" and "Reject" buttons next to each student's name. After successful authentication, the Internship Manager interacts with the Model to remove the student from the application list and, in the case of acceptance, add them to the evaluation list. This triggers the Entity Manager to execute a query on the DBMS. The internship data flows back through the components, starting with the Entity Manager, passing through the Internship Manager, and ultimately resulting in the Internship Manager sending a success notification to the WebApp. Additionally, an email is sent to the student through the email service provider.

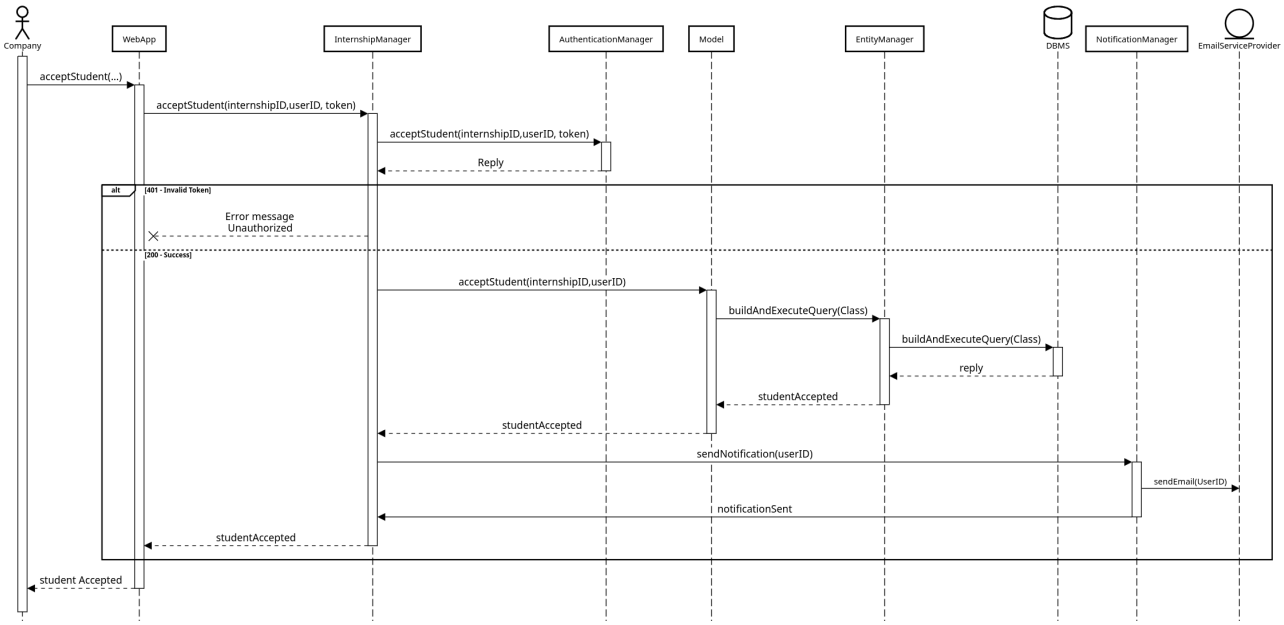

Figure 18: Company decides to accept students for the evaluation

2.4.14 Student decides to accept a company invitation

The diagram below illustrates the process of a student accepting an invitation for an internship.
The student views the internship and is presented with "Confirm" and "Reject" buttons. The student clicks on one button. After successful authentication, the Internship Manager communicates with the Model to either remove the student from the application list or add them to the evaluation list. This triggers the Entity Manager to execute a query on the DBMS. The internship data flows back through the components, starting with the Entity Manager, passing through the Internship Manager, and ultimately resulting in the Internship Manager sending a success notification to the WebApp. Additionally, an email is sent to the student via the email service provider.

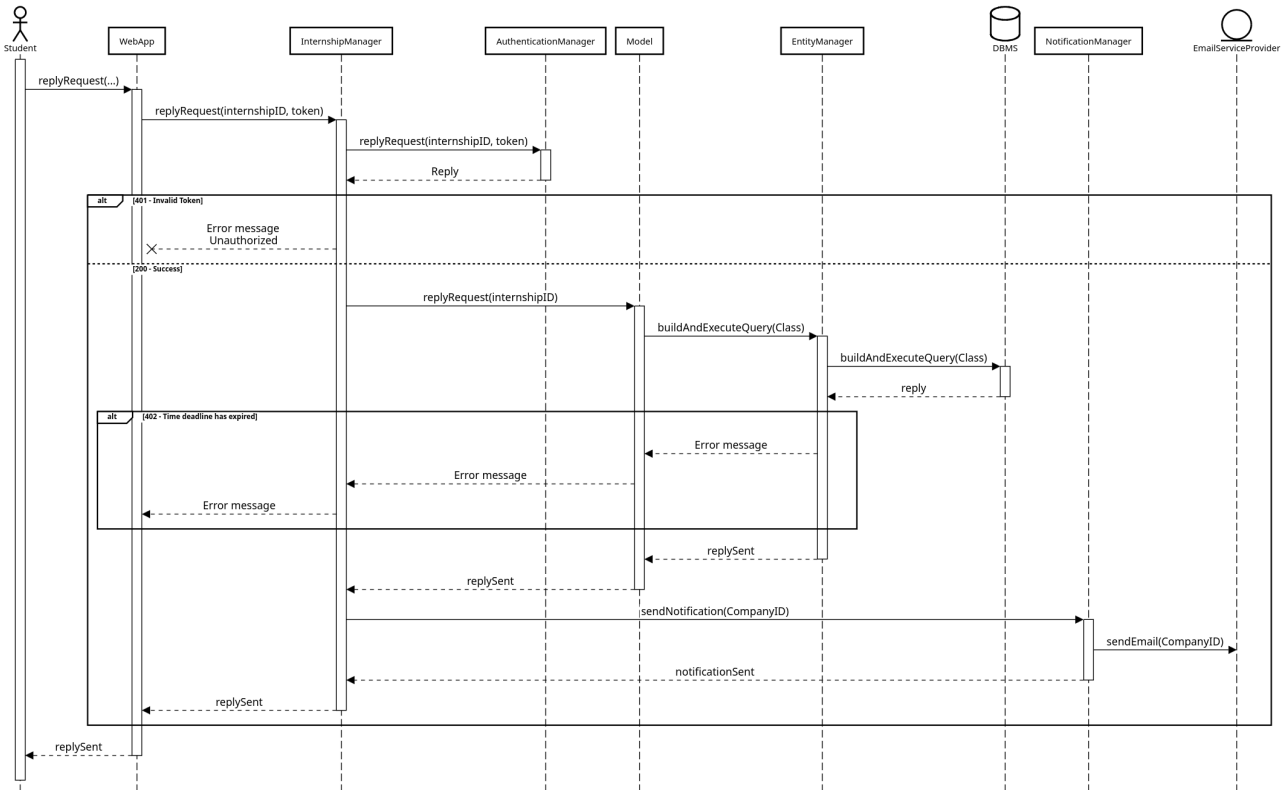

Figure 19: Student decides to accept a company invitation

2.4.15 Company manually closes internship open phase

The diagram below illustrates the process of a company manually closing the open phase of an internship.
The open phase is the period during which students can apply for the internship, or the company can invite students to join the list of candidates. This phase concludes either when the application deadline expires or when the company decides that the candidates meet their requirements. The company navigates to their active internship and confirms their decision to close the open phase. The request is then submitted through the appropriate API call, and the Internship Manager processes it. After successful token validation and role verification, the Internship Manager communicates with the Model to update the internship's status. If the modification is invalid, the Entity Manager returns a null response, which is relayed back through the Internship Manager to the WebApp. If the action is successful, a success notification flows through the components, culminating in the Internship Manager sending a success signal to the WebApp.

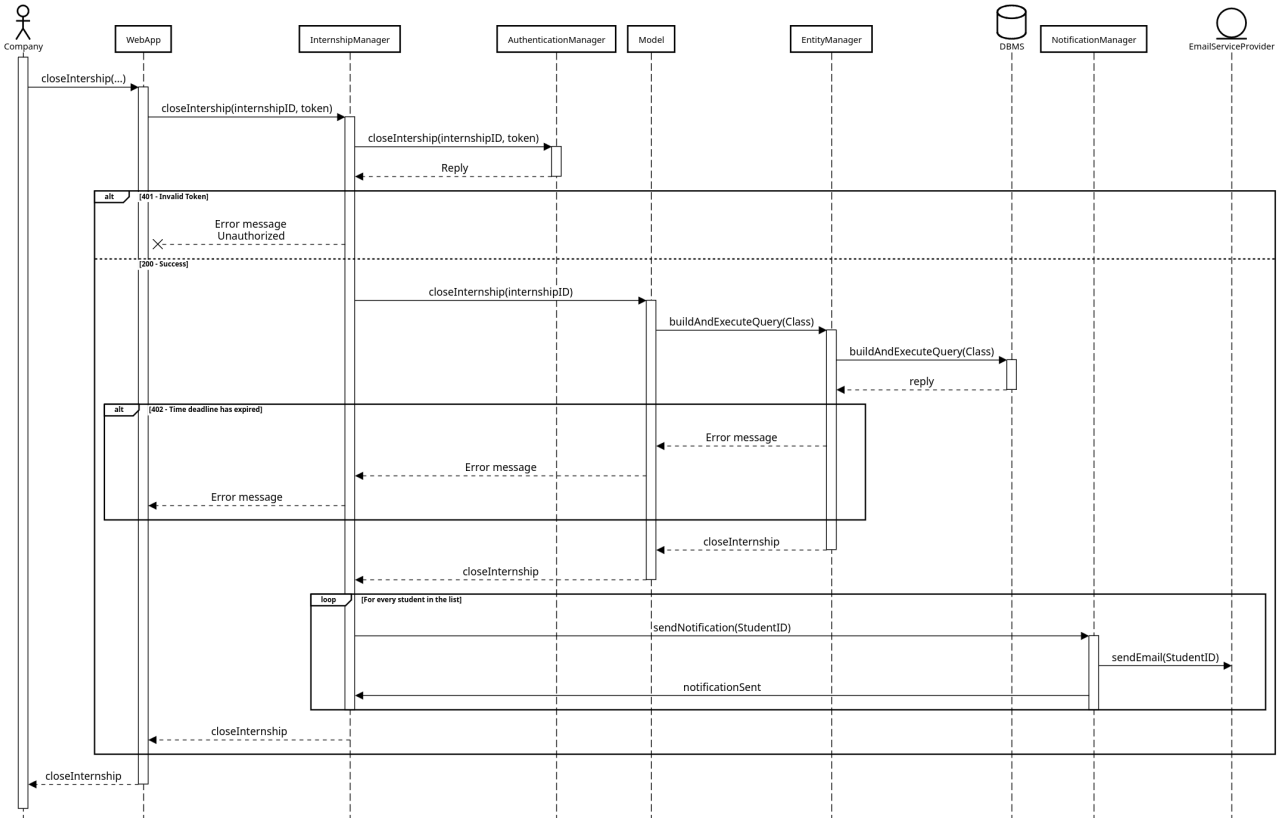

Figure 20: Company manually closes internship open phase

2.4.16 Company views students responses to questionnaire

The diagram below illustrates the process of a company viewing the questionnaire responses of students during the evaluation phase.
The company navigates to the internship in its evaluation phase and clicks on the "View Student Responses" button. After successful authentication, the Internship Manager communicates with the Model to retrieve the questionnaire responses. This triggers the Entity Manager to execute a query on the DBMS. If no responses are found, the Entity Manager returns a null response, which is relayed back through the Internship Manager to the WebApp. If responses are found, a success notification flows through the components, culminating with the Internship Manager displaying the responses on the WebApp.

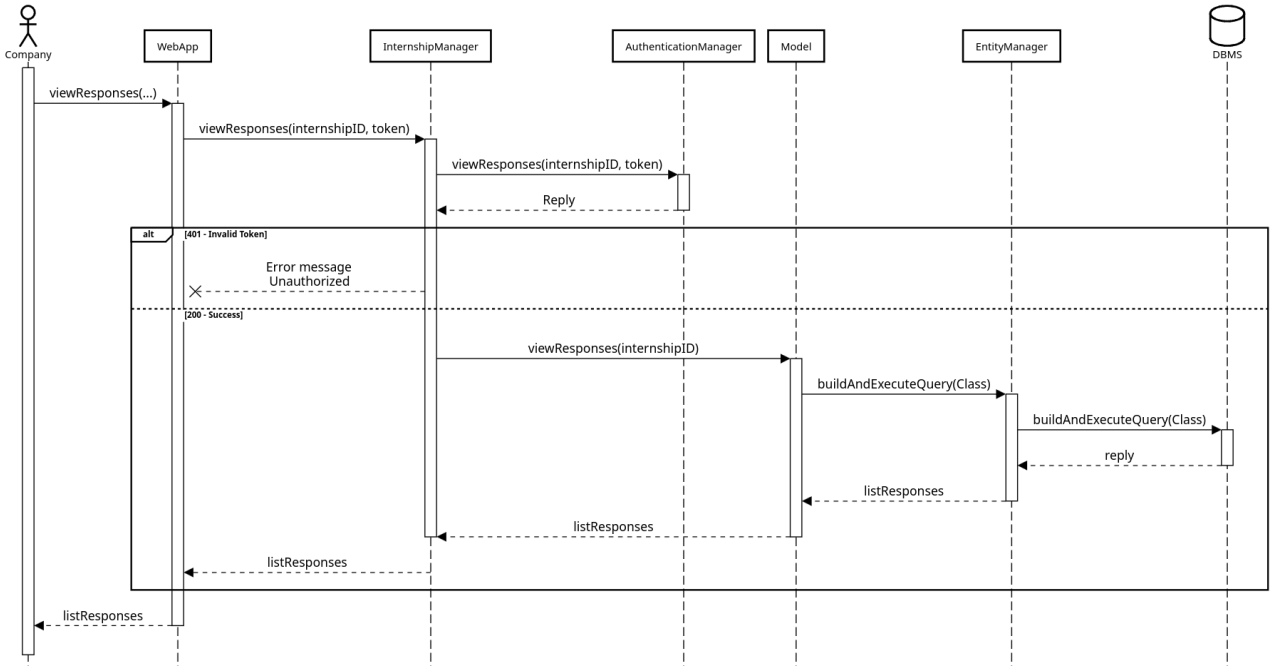

Figure 21: Company views students responses to questionnaire

2.4.17 Company views list of evaluated students

The diagram below illustrates the process of a company viewing the list of evaluated students for one of their internships.
The company navigates to the active internships and selects the one for which they want to view the evaluation list. After successful authentication, the Internship Manager communicates with the Model to retrieve the internship matching the ID and fetch the evaluation list. This triggers the Entity Manager to execute a query on the DBMS. The internship data flows back through the components, starting from the Entity Manager to the Internship Manager, and ultimately ends with the Internship Manager sending the list of evaluated students to the WebApp.

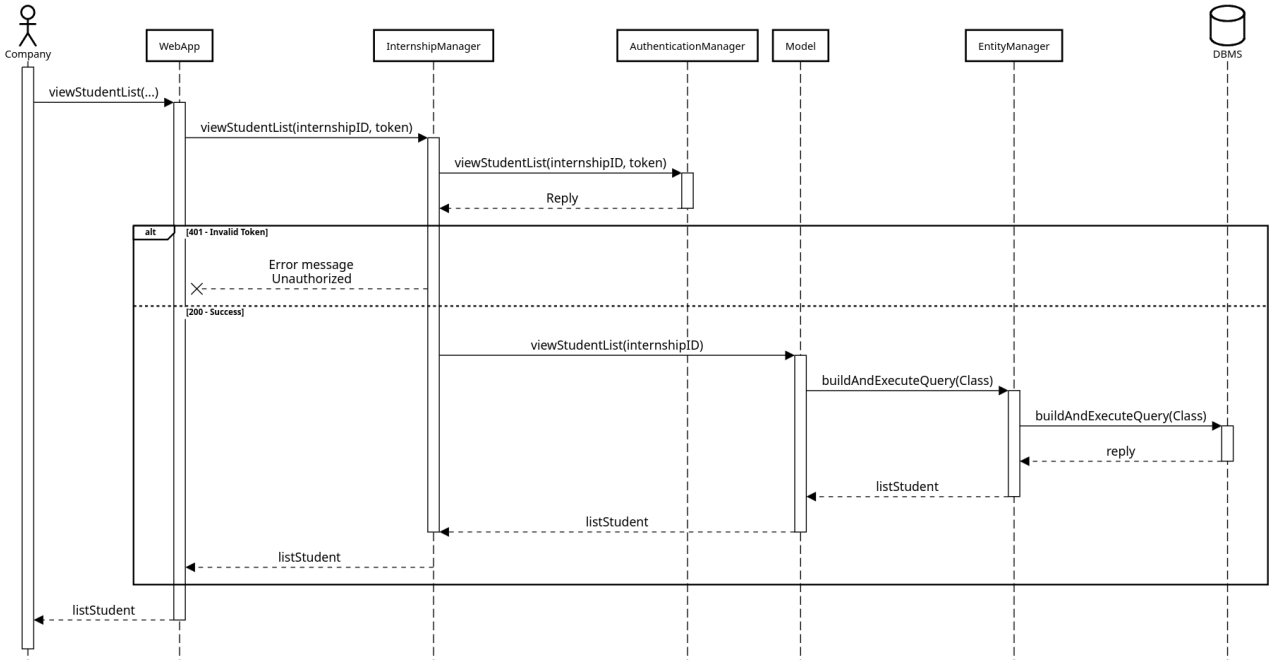

Figure 22: Company views list of evaluated students

2.4.18 Company selects a student for the internship

The diagram below illustrates the process of a company selecting the student from the evaluation candidates.
The company selects the internship and then views the list of evaluated students. Once a decision is made on which student to choose, the company confirms the selection by clicking the "Confirm" button. After successful authentication, the Internship Manager executes a query on the DBMS to assign the chosen student to the internship. If the modification is invalid, the Entity Manager returns a null response, which is relayed back through the Internship Manager to the WebApp. If the action is successful, ending with the Internship Manager sending a success signal to the WebApp. Additionally, an email with the evaluation results is sent to all evaluated students via the email service provider.

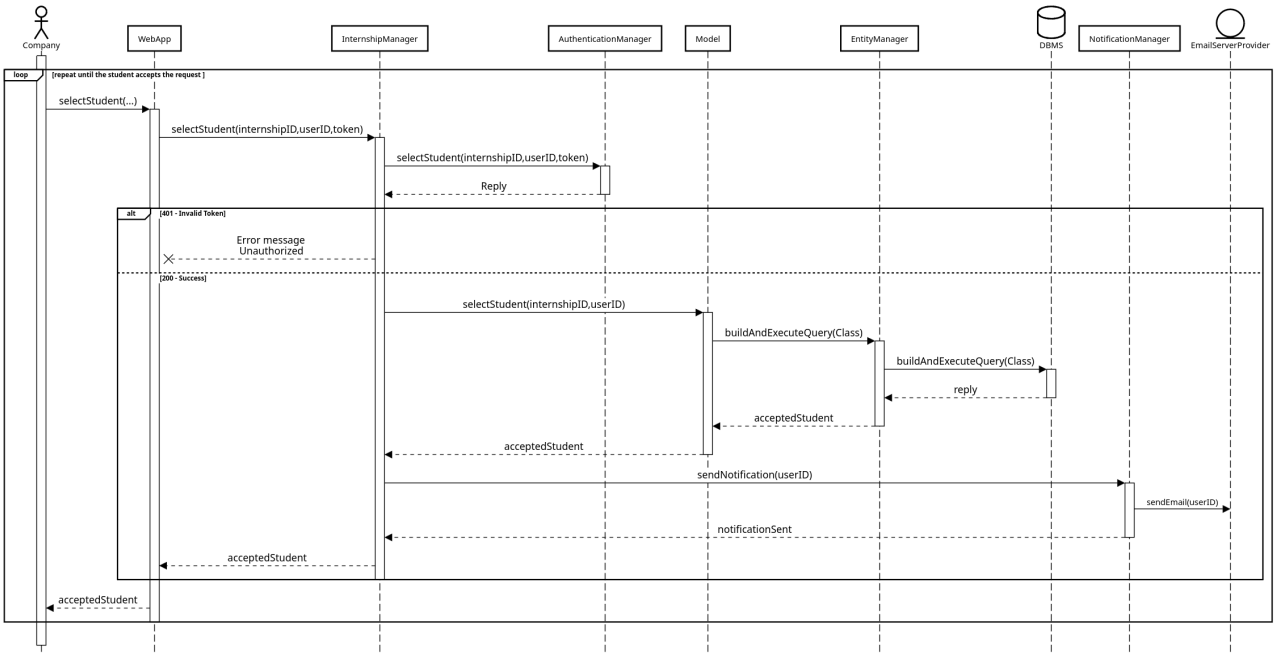

Figure 23: Company selects a student for the internship

2.4.19 Company ends evaluation phase without selecting a student

The diagram below illustrates the process of a company concluding the evaluation phase without choosing a student.
The company navigates to the internship in the evaluation phase and clicks on the "Delete Internship" button. The request is submitted through the appropriate API call, and the Internship Manager processes it. After successful token validation and role verification, the Internship Manager communicates with the Model to remove the internship and notify all enrolled students via email. If the deletion request is invalid, the Entity Manager returns a null response, which is relayed back through the Internship Manager to the WebApp. If the operation is successful, a confirmation notification flows through the components, culminating with the Internship Manager sending a success signal to the WebApp.

2.4.20 User submits updates or complaints for ongoing internship

The diagram below illustrates the process of a user submitting updates or complaints during an internship.
The request is initiated via the appropriate API call, and the Feedback and Complaint Manager handles the process. After successful token validation and role authorization, the Feedback and Complaint Manager interacts with the Model to record the update or complaint. If the provided data is invalid, the Entity Manager returns a null response, which is relayed back through the Feedback and Complaint Manager to the WebApp. In the case of a successful submission, a success notification is propagated through the components, culminating in the Feedback and Complaint Manager sending a confirmation signal to the WebApp.

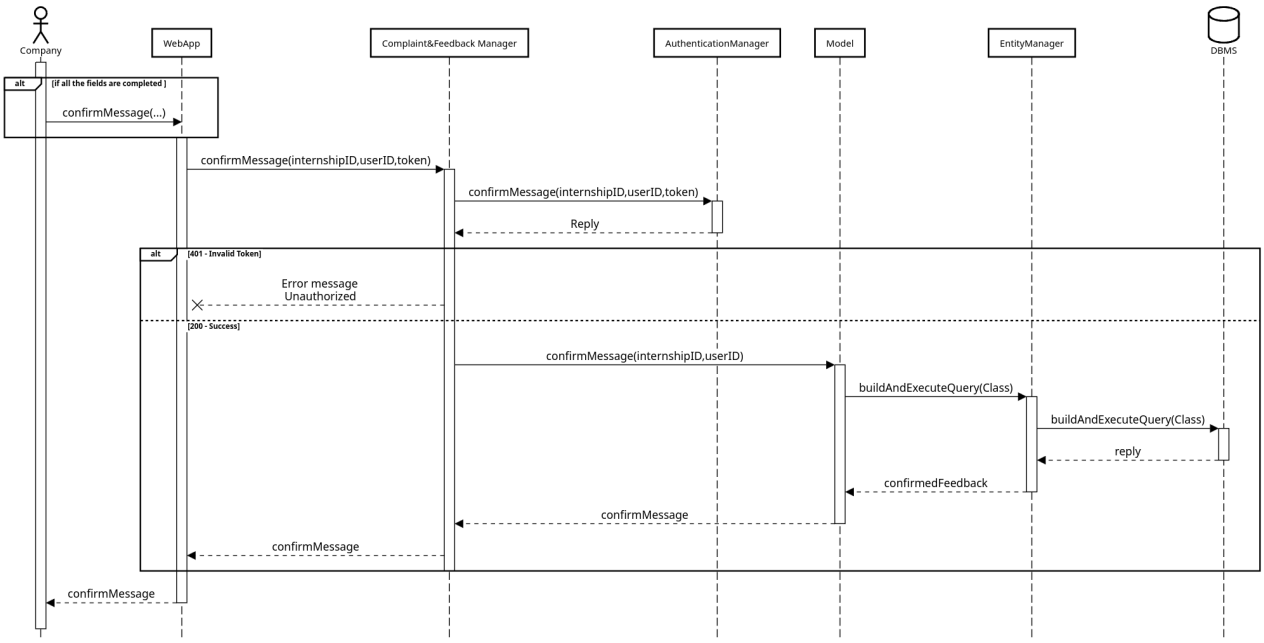

Figure 24: User submits updates or complaints for ongoing internship

2.4.21 User submits feedback for completed internship

The diagram below illustrates the process of a user submitting feedback at the conclusion of an internship. The request is initiated via the appropriate API call, and the Feedback and Complaint Manager handles the process. After successful token validation and role verification, the Feedback and Complaint Manager interacts with the Model to record the feedback for the internship. If the submitted data is invalid, the Entity Manager returns a null response, which is relayed back through the Feedback and Complaint Manager to the WebApp. In the case of a successful submission, a success notification is propagated through the components, culminating in the Feedback and Complaint Manager sending a confirmation signal to the WebApp.

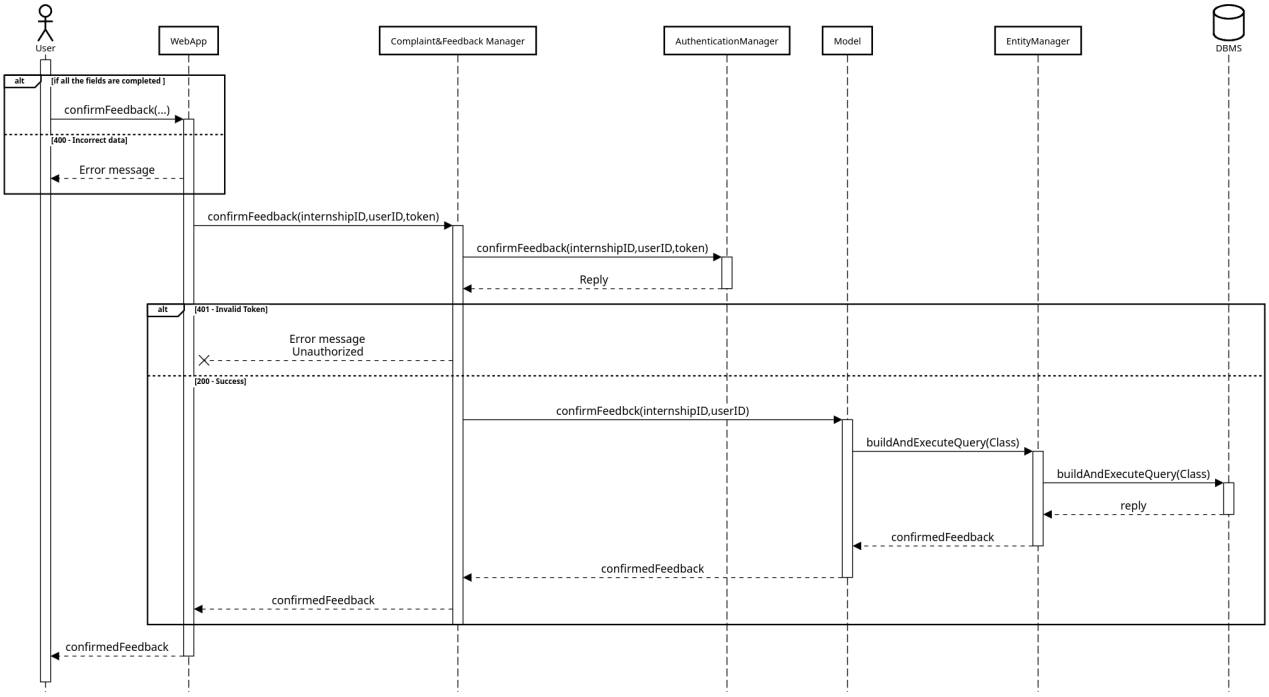

Figure 25: User submits feedback for completed internship

2.4.22 Company accesses its internships

The diagram below illustrates the process of a company accessing its internships.
After successful authentication, the Internship Manager engages with the Model to locate internships associated with the company. This prompts the Entity Manager to perform a query on the DBMS. The retrieved internship information is sent back through the system, starting with the Entity Manager, passing through the Internship Manager, and finally being delivered by the Internship Manager to the WebApp as a list of internships belonging to the company.

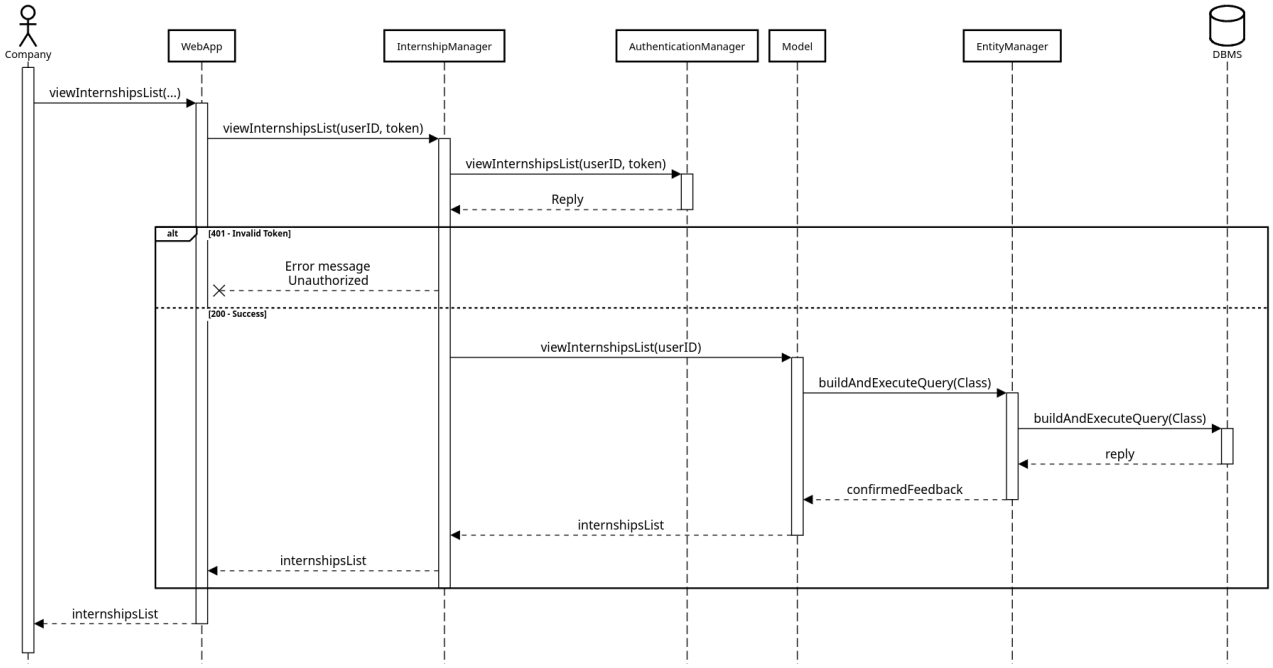

Figure 26: Company accesses its internships

2.4.23 Student views enrolled internships

The diagram below illustrates the process of a user viewing their active internships.
After successful authentication, the Internship Manager communicates with the Model to retrieve the active internships associated with the user. This triggers the Entity Manager to execute a query on the DBMS. The retrieved internship data flows back through the components, starting with the Entity Manager, passing through the Internship Manager, and finally being sent by the Internship Manager to the WebApp.

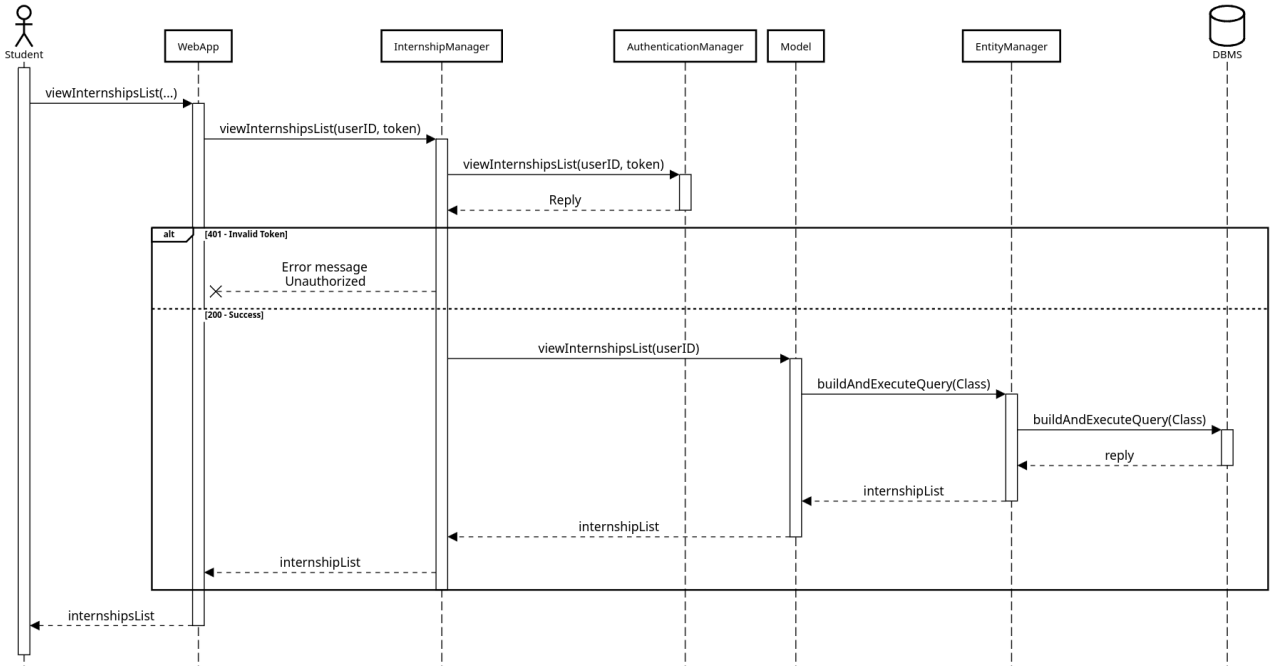

Figure 27: Student views enrolled internships

2.4.24 Student views evaluation questionnaire

The diagram below illustrates the process of a student viewing the evaluation questionnaire associated with an internship.
After successful authentication, the Internship Manager communicates with the Model to retrieve the questionnaire for the internship using the Internship ID. This interaction prompts the Entity Manager to execute a query on the DBMS. If no questionnaire is associated with the internship, the Entity Manager returns a null response, which is relayed back through the Internship Manager to the WebApp. In the case of a successful retrieval, the questionnaire data flows through the components, culminating with the Internship Manager sending the questionnaire to the WebApp.

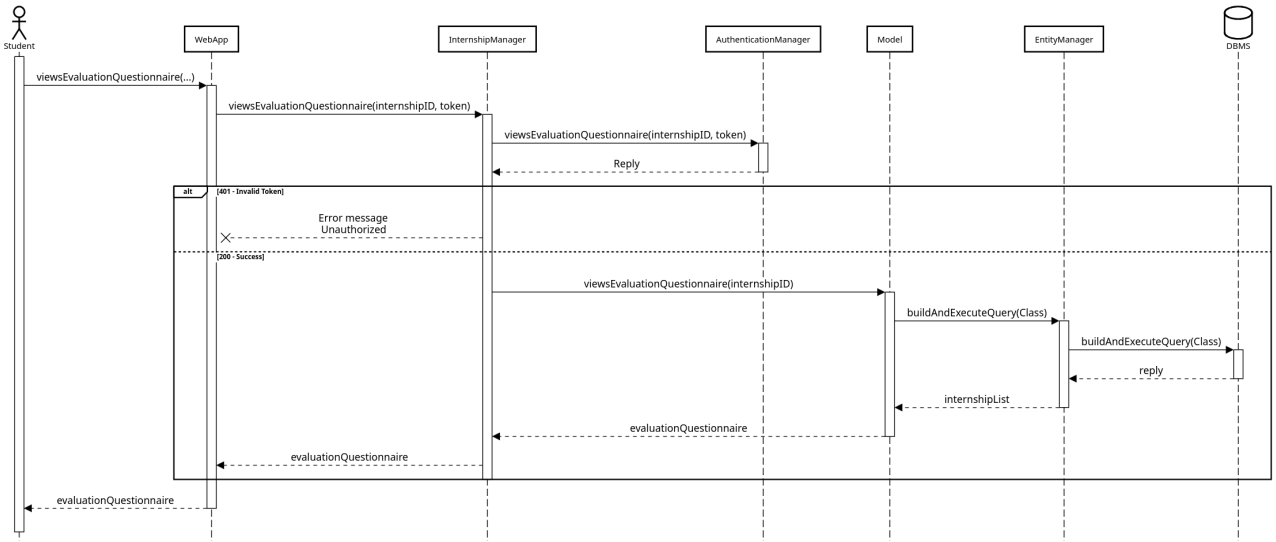

Figure 28: Student views evaluation questionnaire

2.4.25 Student submits evaluation questionnaire responses

The diagram below illustrates the process of a student submitting responses to the evaluation questionnaire.
After successful authentication, the Internship Manager communicates with the Model to process the student's responses. This action prompts the Entity Manager to execute a query on the DBMS. If the data is invalid, the Entity Manager returns a null response, which is relayed back through the Internship Manager to the WebApp. In the case of a successful submission, a success notification is propagated through the components, culminating with the Internship Manager sending a success signal to the WebApp.

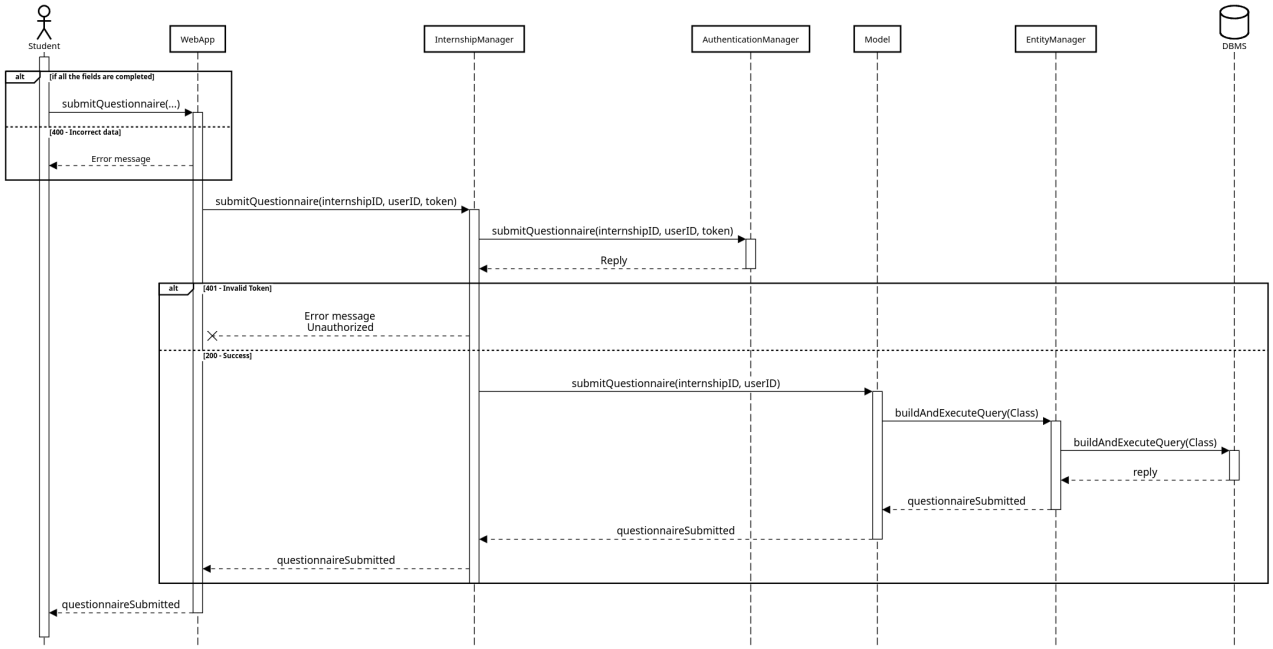

Figure 29: Student submits evaluation questionnaire responses

2.5 Component interfaces

2.5.1 API endpoints

POST api/user/login

- **Request Body:** User: Object
- **200 Response:** Email, token : Object
- **400 Response:** validationError : Object
- **401 Response:** loginError : Object

POST api/user/register

- **Request Body:** User: Object
- **201 Response:** Created
- **400 Response:** validationError : Object

GET api/internships

- **200 Response:** Internships : Array of objects
- **401 Response:** Unauthorized

GET api/internships/state/state

- **200 Response:** InternshipsState : Array of objects
- **400 Response:** validationError : Object
- **401 Response:** Unauthorized

GET api/internships/internhsipID

- **Request Body:** internshipID: int
- **200 Response:** InternshipsState : Array of objects
- **400 Response:** validationError : Object
- **401 Response:** Unauthorized

GET api/internships/owned

- **200 Response:** ownedInternship: Array of objects
- **401 Response:** Unauthorized

GET api/internships/applicated

- **200 Response:** applicatedInternship: Array of objects
- **401 Response:** Unauthorized

GET api/internships/underEvalutation

- **200 Response:** underEvalutationInternship: Array of objects
- **401 Response:** Unauthorized

POST api/internships/create

- **Request Body:** internship: Object
- **201 Response:** Created
- **400 Response:** validationError : Object
- **401 Response:** Unauthorized

POST api/internships/apply

- **Request Body:** internshipID: int
- **200 Response:** "Application successful." : String
- **400 Response:** error : Object
- **401 Response:** Unauthorized

POST api/internships/respondQuestionnaire

- **Request Body:** internshipID: int, response: Array of String
- **200 Response:** "Response saved." : String
- **400 Response:** error : Object
- **401 Response:** Unauthorized

POST api/internships/toEvaluationPhase

- **Request Body:** internshipID: int
- **200 Response:** internship: Object
- **400 Response:** error : Object
- **401 Response:** Unauthorized

GET api/account/accountID

- **Request Parameter:** accountId: int
- **200 Response:** account: object
- **400 Response:** validationError : Object
- **401 Response:** Unauthorized

GET api/prole/search

- **Request Parameter:** keyword: String
- **200 Response:** accounts: Array of Objects
- **401 Response:** Unauthorized

GET api/tournaments/search/keyword

- **Request Parameter:** keyword: String
- **200 Response:** internships: Array of Objects
- **401 Response:** Unauthorized

POST api/internship/acceptStudentAsCandidate

- **Request Parameter:** accountID: int, chosen: bool
- **200 Response:** "Saved choice." : String
- **401 Response:** Unauthorized

POST api/internship/selectStudentForInternship

- **Request Parameter:** accountID: int
- **200 Response:** "Student chosen." : String
- **401 Response:** Unauthorized

2.6 Selected architectural styles and patterns

1. **Three-Tier Architecture:** The 3-tier architecture organizes the system into three distinct layers:

   - **Presentation Layer:** Handled by the Web Server, responsible for the user interface.
   - **Business Logic Layer:** Managed by the Application Server, which processes the core functionality.
   - **Data Storage Layer:** Managed by the Database, ensuring data persistence.

   This structure enables a clear separation of responsibilities, simplifying maintenance, improving scalability, and strengthening security.

2. **RESTful APIs:** RESTful APIs (Representational State Transfer) are built on standard principles for designing web services. They use standard HTTP methods such as: `GET, POST, PUT, DELETE`. These APIs are stateless, facilitating seamless data exchange between systems. They provide:

   - Easy integration with various platforms.
   - Simplicity in implementation.
   - Reduced server load.

3. **Cloud Hosting:** The system is deployed on a cloud-based infrastructure, taking advantage of services provided by cloud providers. Key benefits include:

   - **On-Demand Scalability:** Quickly adjusts resources as needed.
   - **Cost Efficiency:** Pay for only the resources consumed.
   - **High Availability:** Ensures continuous system uptime and reliability.

2.7 Other design decisions

actor User participant Webapp participant Account Manager participant Model participant Entity Manager participant DBMS participant Notification Manager entity Email service provider

3 User Interface Design

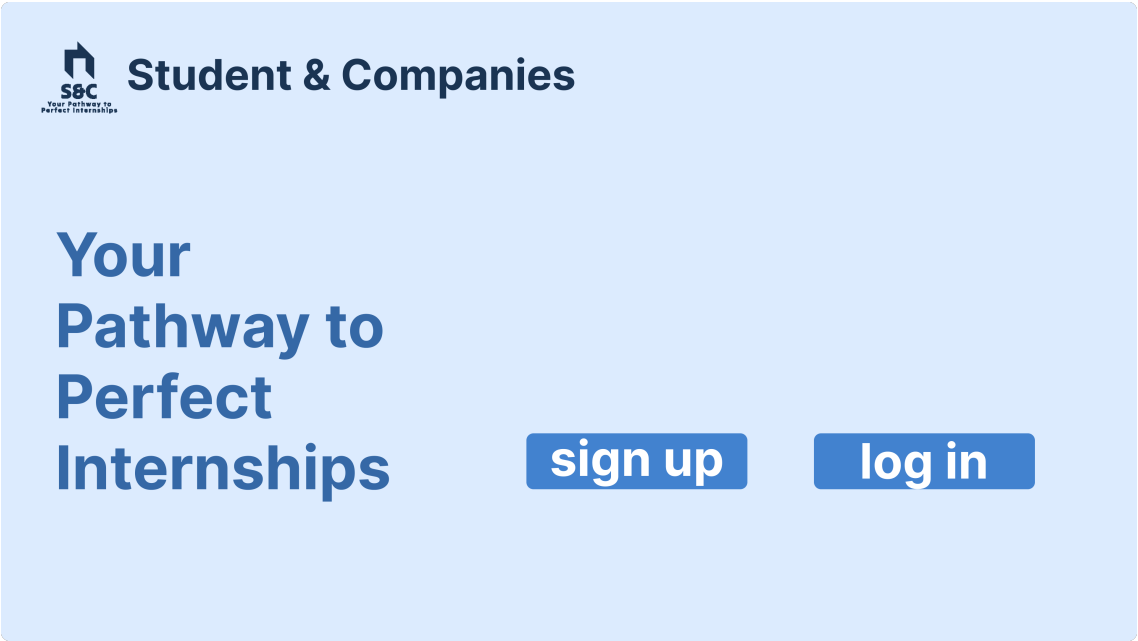

Figure 30: Home page

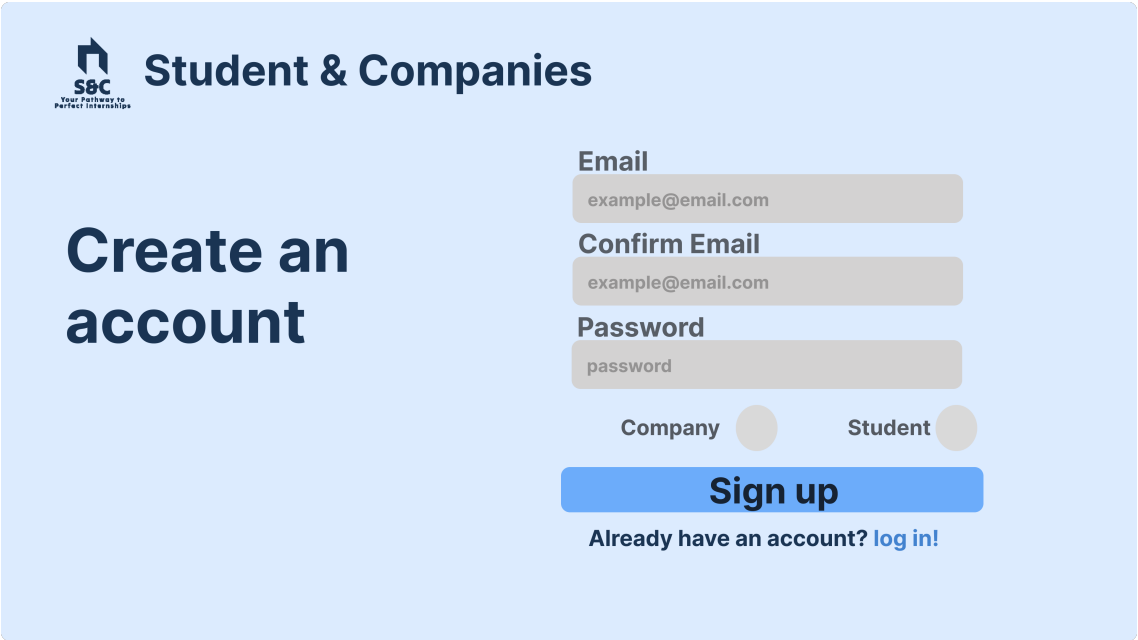

Figure 31: Sign Up

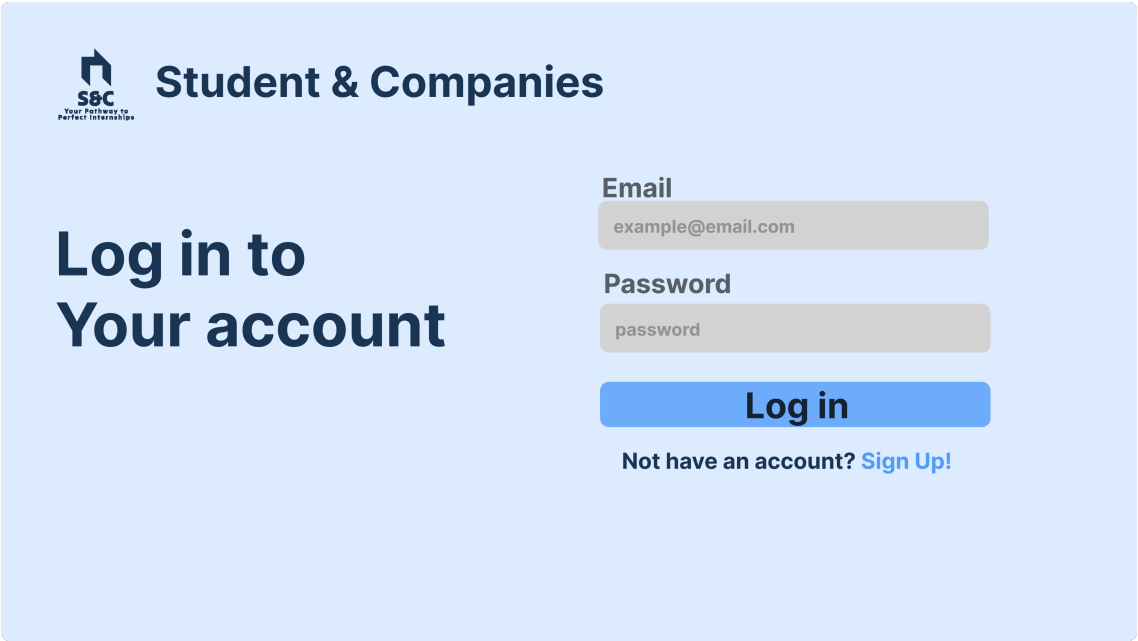

Figure 32: Log In

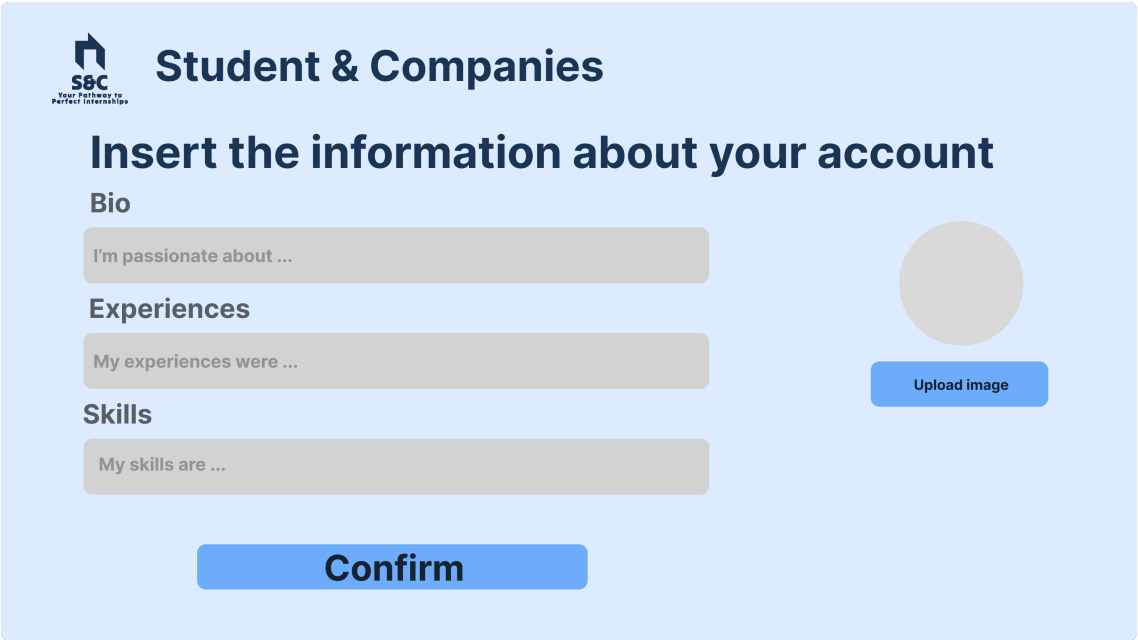

Figure 33: Modify your profile

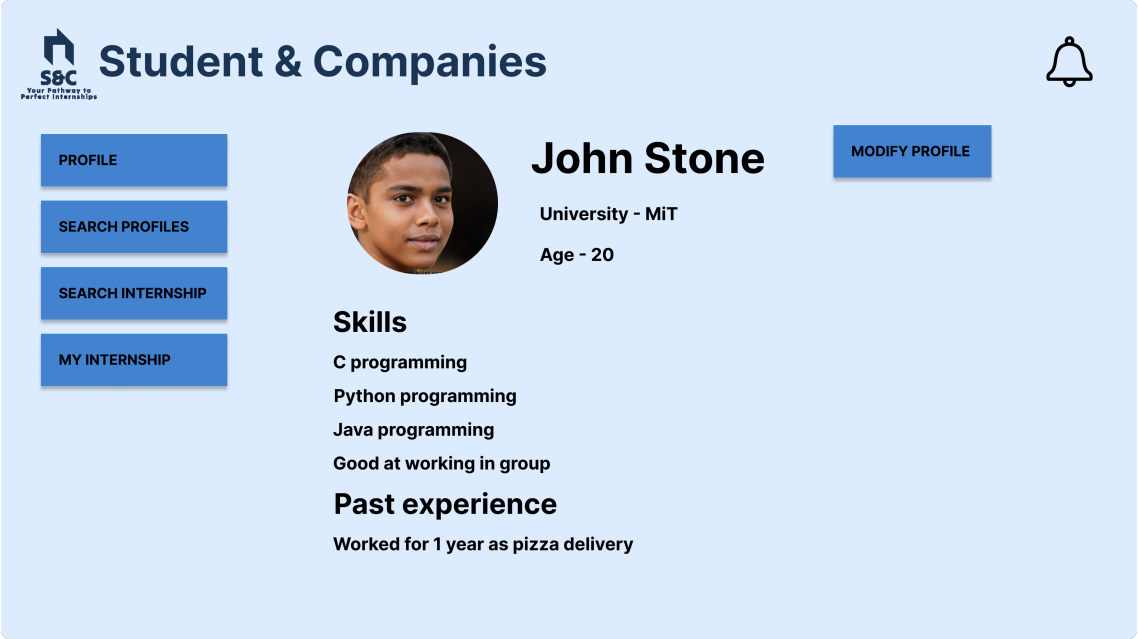

Figure 34: Home page

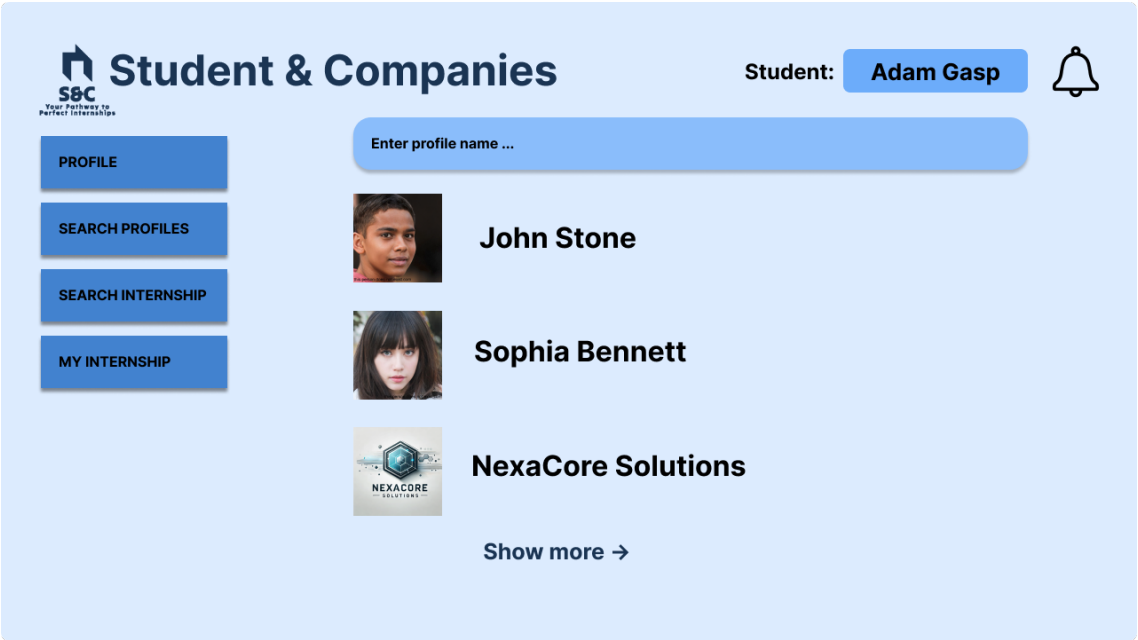

Figure 35: Search profiles

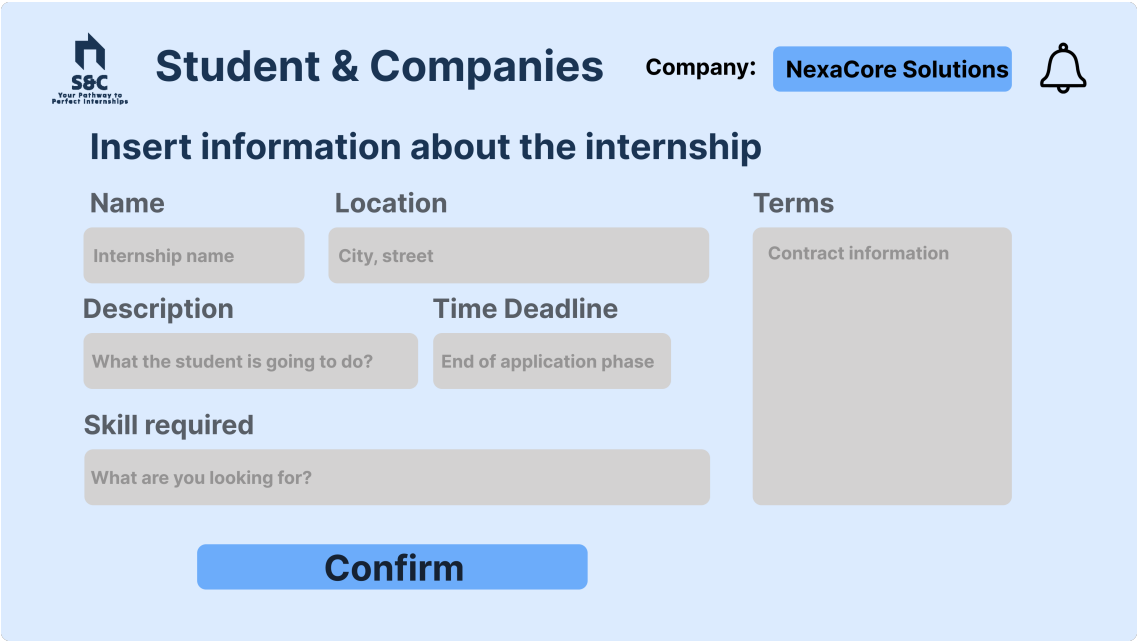

Figure 36: Create internship

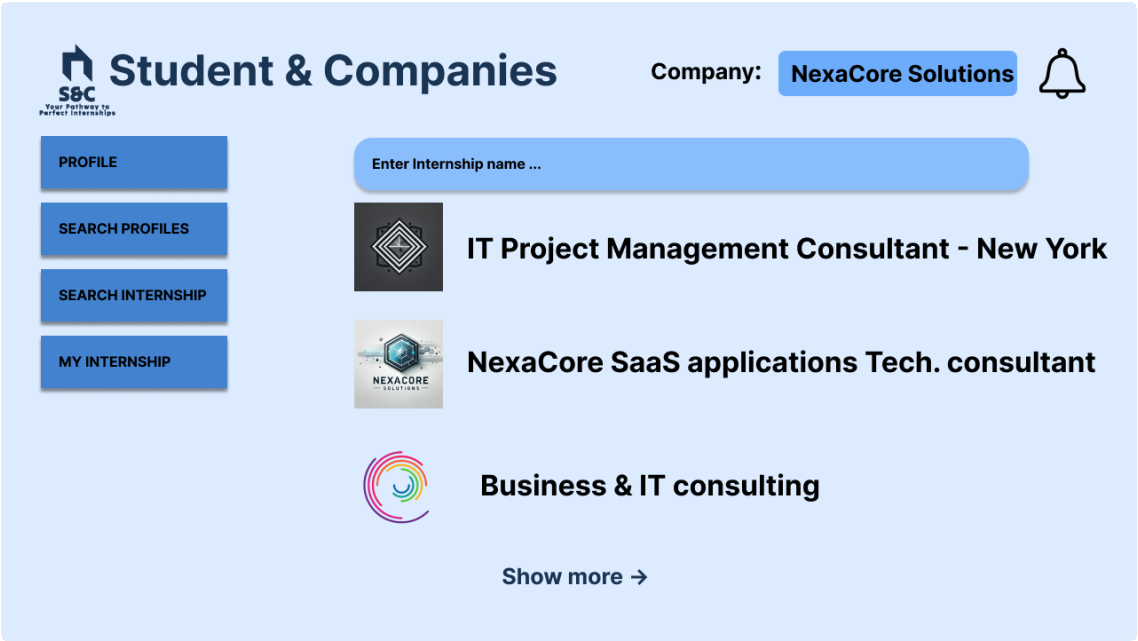

Figure 37: Search internships

4 Requirements Traceability

ID	Description	Components
R1	The system allows user to sign-up.	Account Manager Authentication Manager Model Entity Manager Email service provider
R2	The system allows user to log-in.	Account Manager Authentication Manager
R3	The system sends to users an email with a link to confirm their account registration.	Account Manager Email service provider
R4	The system allows users to modify their accounts at any moment.	Account Manager Authentication Manager Model Entity Manager
R5	The system allows users to browse other users' profiles.	Account Manager Authentication Manager Model Entity Manager
R6	The system allows users to browse the Internship opportunity available list.	Internship Manager Authentication Manager Model Entity Manager
R7	The system constrains users in an internship to periodically report ongoing status of the internship.	Complain Feedback Manager Authentication Manager Model Entity Manager
R8	The system enables users to receive system-generated notifications.	Notification Manager Email Service Provider
R9	The system allows users in an internship to make a complaint.	Complain Feedback Manager Authentication Manager Model Entity Manager
R10	The system allows students to fill out the information of their CV.	Account Manager Authentication Manager Model Entity Manager
R11	The system allows students to apply for an Internship at any moment.	Internship Manager Authentication Manager Model Entity Manager
R12	The system allows students to remove application in internship before the evaluation phase.	Internship Manager Authentication Manager Model Entity Manager
R13	The system allows students to work in only one internship at a time.	Internship Manager

ID	Description	Components
R14	The system prevents students from applying twice for an internship opportunity.	Internship Manager
R15	The system notifies the student when a new job opportunity of interest becomes available.	Recommendation Manager Notification Manager Email service provider
R16	The system allows students to choose whether to be evaluated by the company that notified them.	Internship Manager Authentication Manager Model Entity Manager
R17	The system allows students to fill out a questionnaire during the evaluation process.	Internship Manager Authentication Manager Model Entity Manager
R18	The system requires students to complete a feedback form provided by the platform at the end of the internship.	Complain Feedback Manager Authentication Manager Model Entity Manager
R19	The system allows companies to create an Internship at any moment.	Internship Manager Authentication Manager Model Entity Manager
R20	The system allows companies to define terms and projects in the Internship description.	Internship Manager Authentication Manager Model Entity Manager
R21	The system allows companies to hire one student for each internship opportunity.	Internship Manager Authentication Manager Model Entity Manager Notification Manager Email Service Provider
R22	The system provides companies with a default questionnaire for the evaluation of the Students.	Internship Manager
R23	The system allows companies to modify the default questionnaire given by the system to evaluate the student.	Internship Manager Authentication Manager Model Entity Manager
R24	The system allows companies to evaluate the candidate through a structured questionnaire.	Internship Manager
R25	The system allows companies to evaluate the candidate through an organized meeting.	Internship Manager
R26	The system helps companies to organize a meeting with the Students to evaluate.	Internship Manager Notification Manager Email Service Provider
R27	The system prevents companies from making any modifications once a job advertisement is published.	Internship Manager

ID	Description	Components
R28	The system allows companies to set a deadline for the start of the selection of the candidates.	Internship Manager Authentication Manager Model Entity Manager
R29	The system allows companies to decide to start manually the selection process.	Internship Manager Authentication Manager Model Entity Manager
R30	The system allows companies to select students who are not currently participating in another internship.	Internship Manager Authentication Manager Model Entity Manager
R31	The system notifies companies when a Student updates their CV with details related to the company's job description.	Recommendation Manager Notification Manager Email Service Provider
R32	The system allows companies to decide to evaluate or not a student that made the application for their internship opportunity.	Internship Manager Authentication Manager Model Entity Manager

For simplicity and clarity, the figure below illustrates the same relationships in a more compact format using a traceability matrix.

ID	Account Manager	Internship Manager	Complain & Feedback Manager	Authentication Manager	Model	Recommendation Manager	Entity Manager	Notification Manager	Email Service Provider
R1	x			x	x		x		x
R2	x			x					
R3	x								x
R4	x			x	x		x		
R5	x			x	x		x		
R6		x		x	x		x		
R7			x	x	x		x		
R8						x		x	x
R9			x	x	x		x		
R10	x			x	x		x		
R11		x		x	x		x		
R12		x		x	x		x		
R13		x							
R14		x							

ID	Account Manager	Internship Manager	Complain & Feedback Manager	Authentication Manager	Model	Recommendation Manager	Entity Manager	Notification Manager	Email Service Provider
R15						x		x	x
R16		x		x	x		x		
R17		x		x	x		x		
R18			x	x	x		x		
R19		x		x	x		x		
R20		x		x	x		x		
R21		x		x	x	x	x	x	x
R22		x							
R23		x		x	x		x		
R24		x							
R25		x							
R26		x				x		x	x
R27		x							
R28		x		x	x		x		
R29		x		x	x		x		
R30		x		x	x		x		
R31						x		x	x
R32		x		x	x		x		

5 Implementation, Integration and Test Plan

5.1 Overview

In this chapter, it is explained how the platform will be implemented and tested. The goal of testing is to identify the majority of the bugs in the code produced by the development team.
The system will be validated using a bottom-up approach, considering the dependencies between system components. This method will be applied to both the server and client sides, which will be developed and tested separately. Since the Entity Manager and external services are presumed to be dependable, unit testing for them is not required.

5.2 Implementation and Integration Plan

The aim of this section is to describe the implementation strategies that will be employed to develop, integrate, and test the various components.

5.2.1 Server Side

1. In the first step, the model is implemented and unit tested. The model, which utilizes the Entity-Manager, is responsible for all interactions with the database.

2. In the second step, the controllers are implemented and unit tested using drivers and stubs for modules that have not yet been implemented or are external. Integration tests with the model are also conducted.

3. In the third step, the Recommendation is implemented and unit tested using Drivers and Stubs for not yet implemented or external modules. Integration tests between the controllers are done at this stage.

4. Once all components have been implemented and unit tested, full integration tests with drivers and stubs for external services are performed. As the final step, integration tests with the actual services are conducted.

5.2.2 Client Side

Each component of the View is thoroughly unit tested using a stub of the REST API. A stub is a simplified version of the actual API that simulates its behavior. This allows the frontend (View) components to be tested independently of the backend services. By using the stub, developers can work on the frontend and backend in parallel, as the frontend can interact with the stubbed API instead of waiting for the backend to be fully implemented. This approach helps speed up development by enabling both teams to progress simultaneously without dependencies.

5.2.3 Full testing

After the successful implementation of both the backend and frontend components, the final phase of testing can be initiated. At this stage, the entire system, including both the server-side and client-side components, is thoroughly tested to ensure that all functionalities work as expected in an integrated environment. This includes verifying that the communication between the frontend and backend is seamless, and that all user interactions, data exchanges, and processing workflows function correctly. The objective of these final tests is to validate the system's overall performance, usability, and stability before it is distributed.

5.3 System testing

The platform must be tested to ensure that the functionalities developed by the different teams align with the expected outcomes. During the development phase, each created component needs to be verified; however, not all components can be tested in isolation from the overall architecture. As a result, stubs and drivers are required to replace the missing components effectively. The primary goal of this testing process is to determine whether the system meets the functional and non-functional requirements outlined in the RASD document. In this context, the involvement of all stakeholders in the software development process is essential. Therefore, testing is not solely the responsibility of developers, but other stakeholders also play a crucial role in validating the platform. The following steps will be undertaken during the testing phase:

- **Functional Testing:** Verify that all functional requirements are met by running the system according to the use cases specified in the RASD documents.
- **Performance Testing:** The main objective of this test is to identify latency or slow response times. It helps detect inefficient algorithms, hardware/network issues, or uncover optimization opportunities. To perform this test, it is necessary to load the system with the expected workload, measure and compare its performance, and identify potential optimization areas.
- **Usability Testing:** This method involves testing the features of a website, app, or other digital products by observing real users (for example, stakeholders can participate) as they attempt to complete tasks on the system.
- **Load Testing:** Expose the system to potential issues such as memory leaks, buffer overflows, or memory mismanagement. To conduct this test, it is essential to evaluate the system under increasing workloads until it can sustain the load for a predefined period.
- **Stress Testing:** This test verifies whether the system can recover gracefully after a failure. You can test this by gradually increasing the system's resource usage or reducing them to observe its recovery behavior.

6 Effort Spent

Section	Gabriele Lorenzo Singe	Kevin Pio Abate
1	2	1
2	20	40
3	1	0
4	6	10
5	4	1
tot	33	52

Table 3: Time spent on each section by contributors.

References

- Database and components diagrams made with: https://app.diagrams.net/
- Sequence diagrams made with: https://sequencediagram.org/
- User Interface mockups made with: https://www.figma.com/