Students&Companies Design Document

Alessandro Salvatore, Erdal Yalçın, Leonardo Ratti

Academic Year: 2024-25

Contents

1 Introduction

1.1 Purpose

This document contains the design description of the Students&Company system. It includes the architectural design, the user interface design and the descrpition of the operations that the system will perform. It also show how the requirements and use cases detailed in the RASD document are satisfied by the design of the system.

This document is intended to be read by the developers of the system, the testers and the project managers. It is also intended to be used as a reference for the future maintenance of the system.

1.2 Scope

The Students&Company system is a web application that allows companies to advertise internship opportunities for university students. A recommendation system is used for enhancing the matching possibilities between the two parties. The system provides also suggestions for students and companies to increase their chances to get matched by the recommendation system.

A more detailed description of the system can be found in the RASD document, whilist in this document is provided a detailed description of the design of the system to implement the requirements and use cases described in the RASD document.

1.2.1 Definitions

- **User**: with User we refer to an active individual that can be either a Student, a Comapny or a University.
- **Advertise**: An Internship is advertised only if it has been posted and its application deadline is not yet expired.
- **Recommendation**: It's a platform feature that starts a matching between a company and a student. If both parties accept the recommendation, the student is taken for the selection process
- **Applying**: a student applies for an internship if he/she manually searches for it and applies for it and the internship was not recommended to him/her.
- **Accepting**: The act of students or companies, who got recommended to each other, to confirm their interest.
- **Contact**: The mutual acceptance of recommendation between student and company on the same internship.
- **Selection**: It's the process that starts after the expiration date of applying for the

internship. The company interviews every accepted student and picks the best one(s) for their needs.

- **Selected student**: He is the student who has been chosen for the internship by the company.
- **Candidate**: he is a Student that has passed to the selection phase of an internship.
- **Feedback**: Consists of two separate moments. The first round of feedback is when the users likes/dislikes the recommendation given by the system. The second type of feedback is submitted after the end of the internship, where students and companies rate the experience through a 5 star review form.
- **Comment**: Is anything written in the dedicated Comments section. Serves the student or the company currently engaged, to highlight something about the experience with each other. If that's a complaint from either, the University of the student will manage the situation.
- **Complaint**: It's a specific type of Observation, where the University of the student can manage the situation between the parties.
- **Observation**: It's a type of comment that is not a complaint. It could just report some good or neutral facts about the experience.

1.2.2 Acronyms

- S&C: "Students&Companies", the name of the platform

1.2.3 Abbreviations

- **Wn**: n-th World Phenomena
- **Sn**: n-th Shared Phenomena
- **Gn**: n-th Goal
- **Dn**: n-th Domain Assumption
- **Rn**: n-th Requirement
- **Cn**: n-th Component

1.3 Revision History

Revised on	Version	Description
07-Jan-2025	1.0	Initial Release of the document

1.4 Reference Documents

- The specification document of the project: **Assignment RDD AY 2024-2025**
- The **RASD** document of the project

1.5 Document Structure

The document is divided into 6 sections:

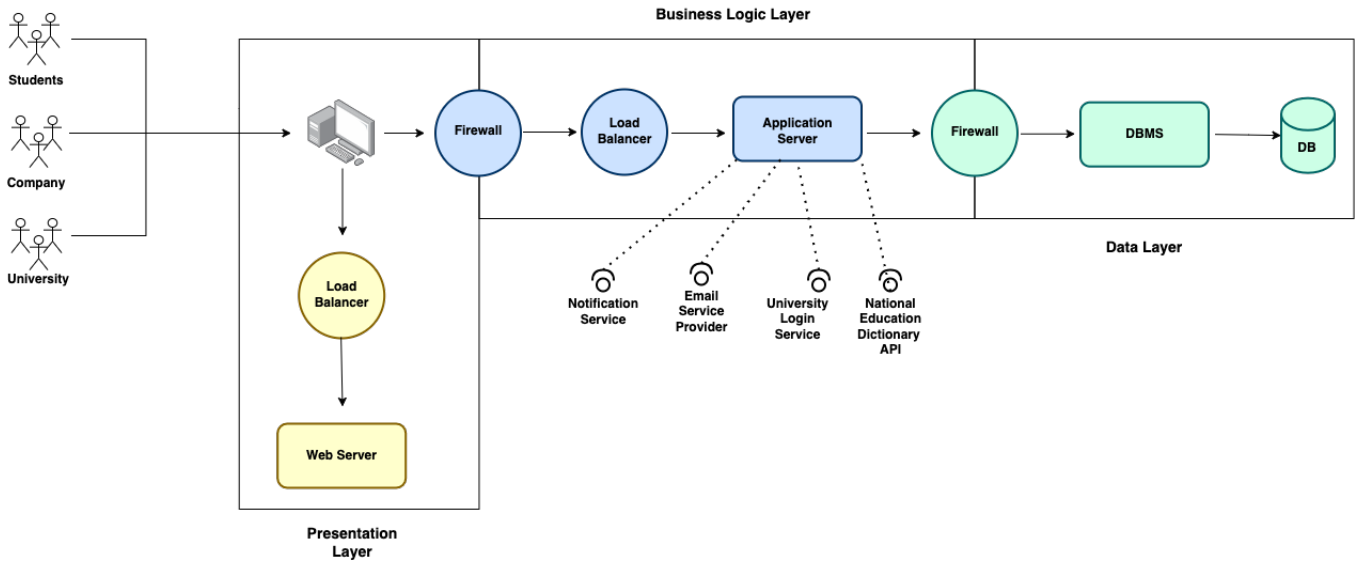
- **Introduction**: provides a brief description of the purpose and the scope of the system. Moreover, it contains the definitions, acronyms, and abbreviations used in the document and the reference documents.
- **Architectural Design**: this section provides a description of the architecture of the system, including the components and the interfaces between them. It also includes the runtime view of the most important operations of the system, the deployment view, and the architectural styles and patterns used in the system.
- **User Interface Design**: includes the mockups of the user interface of the system.
- **Requirements Traceability**: this section shows how the requirements described in the RASD document are satisfied by the design of the system.
- **Implementation, Integration and Test Plan**: this includes the step-by-step plan for the implementation and testing of the system.
- **Effort Spent**: this section highlights the effort spent by each member of the group to redact this document.

2 Overall Description

2.1 Overview: High-level Components and Interaction

The following diagram is designed to clearly demonstrate the high-level components required by the system and the relationships between them.

The Presentation Layer interacts directly with users and is responsible for managing the front-end side. This layer provides the user interface and collaborates with the Load Balancer and Web Server to optimize system performance and functionality. User interactions within the interface are forwarded from the Presentation Layer to the back-end, enabling additional processing and functionality. Furthermore, to enhance the system's security and mitigate potential threats from external networks, a Firewall component is incorporated into the architecture. The Business Logic Layer processes incoming requests from the Presentation Layer and facilitates access to the core system functionalities, such as internship applications and recommendation mechanisms, through the use of APIs and Services. In summary, the Business Logic Layer, which houses the Application Server designed in accordance with RESTful standards, ensures the seamless transfer of operations to the Data Layer, thus ensuring the system operates in an efficient, secure, and effective manner.



2.2 Component View

In this section we show the components and their relationships. The following sections will explain the interaction between interfaces and details on each method of interfaces.

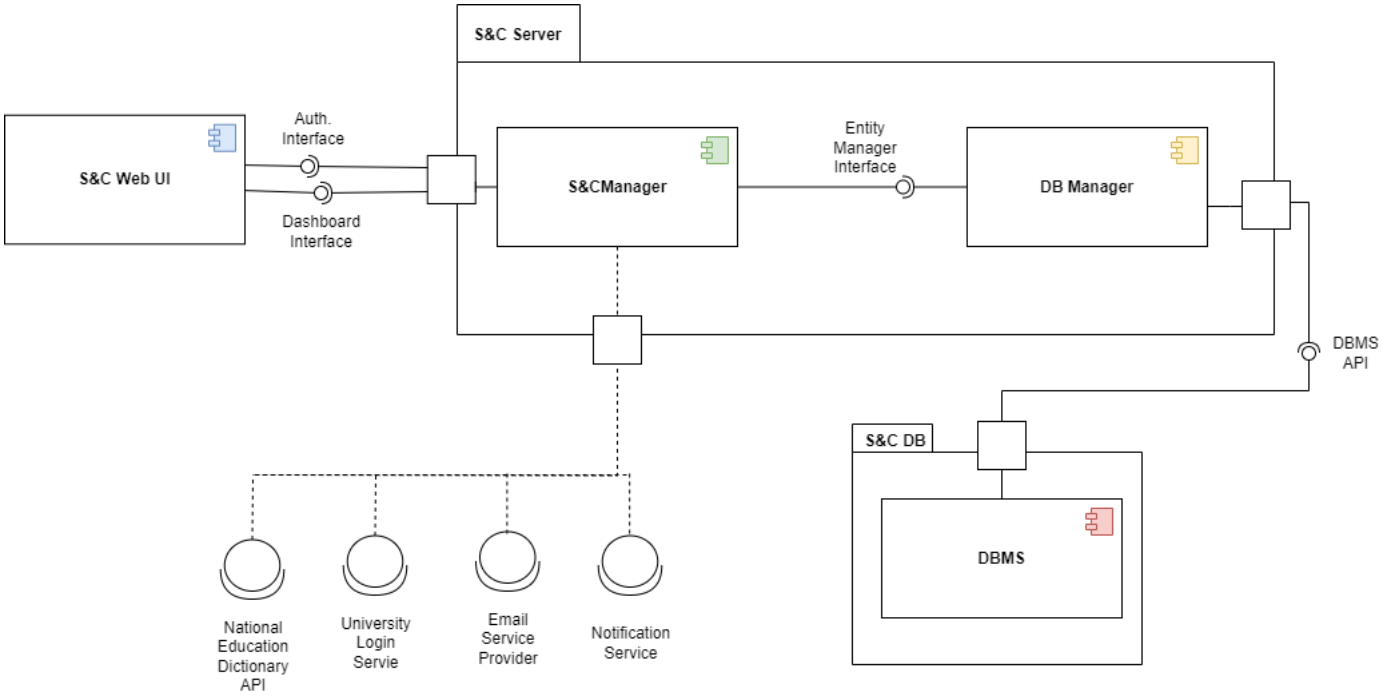


Figure 1: Component Diagram of the Student&Companies System

2.2.1 DB Manager

This component is responsible for communication with a Database Management System (DBMS). It follows the Adapter design pattern, allowing other components to interact with the DBMS without needing to write any SQL code themselves.

2.2.2 S&C-SP

The S&C-SP subsystem implements S&C's logic. It provides all the system's features, including the recommendation system.

1. C1 - Dashboard Manager :
   It's used as an intermediary between the WebUI and the other components of the server to provide the web application only the functionality strictly necessary for it to work properly.
2. C2 - Account Manager :
   This component handles the operations needed to manage each User's account, and to view the Users' profiles. It allows Users to login, using the Authentication Manager's interface.

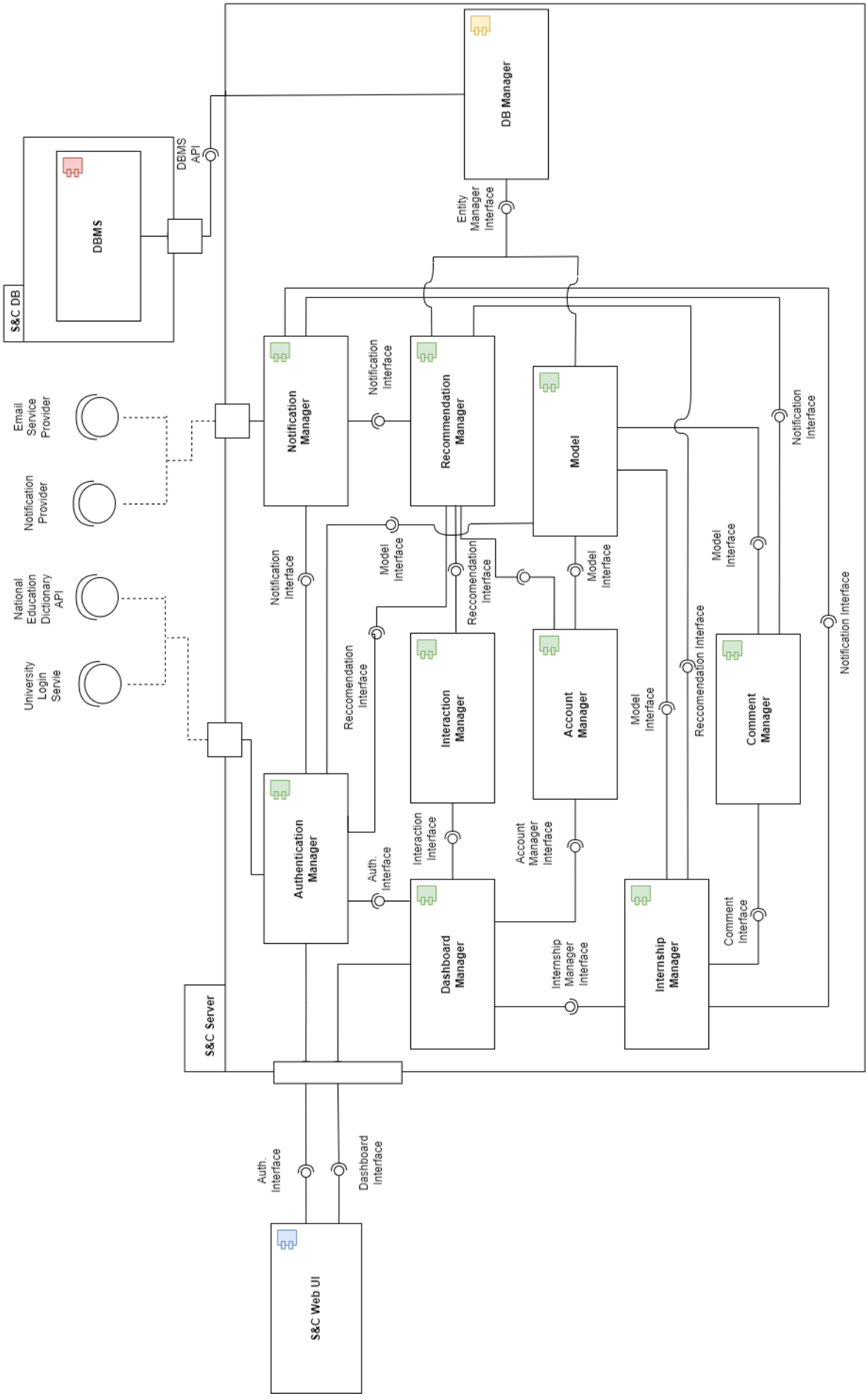


Figure 2: Component Diagram of the S&C-SP subsystem

3. C3 -Authentication Manager :
   This component handles signup, login and logout operations.
4. C4 -Internship Manager :
   This component handles the operations needed to manage an Internship, to view any information about it and to accept it.
5. C5 - Comment Manager :
   This component handles the operations needed to view and write comments (complaints and observations).
6. C6 -Recommendation Manager :
   This component is responsible for the recommendation mechanism for students and internships. It takes the input data from the user interactions with the system and reloads the associated recommendations.
7. C7 -Interaction Manager :
   This component is responsible for registering feedbacks and interactions of the user with the web application, transforming this information into parameters, and passing it to the recommendation manager for elaboration.
8. C8 -Notification Manager :
   This component is responsible for the dispatch of notifications.
9. C9 - Model :
   This component facilitates the interaction with and representation of S&C-SP data.

2.3 Deployment View

We adopted a 3-tier architecture hosted on cloud infrastructure, designed to optimize performance and scalability while reducing costs.

Let's explore more in detail:

1. **Scalability and Flexibility** - the ability to add or remove resources such as virtual machines, performance cores, or memory as needed, in conjunction with the use of load balancing services, allows the servers to adapt to changes in traffic or workload.
2. **Security** - Firewalls help to protect the application server against data breaches and other security threats.
3. **Cost-efficiency** - the choice of using a cloud provider allows us to only pay for the resources that we are actually using, which can help to lower the overall costs, and in general reduces the waste of resources.

The cloud provider is an ideal choice for hosting large, high-traffic applications. The chosen cloud provider will need to offer all of these features in order to meet our needs.

The web server resides in the De Militarized Zone and is the primary interface for user interac-

tion. The application server executes the core application logic, processing user requests and handling all the operations. Load balancers are implemented to evenly distribute incoming traffic across multiple instances of Web and Application Servers.

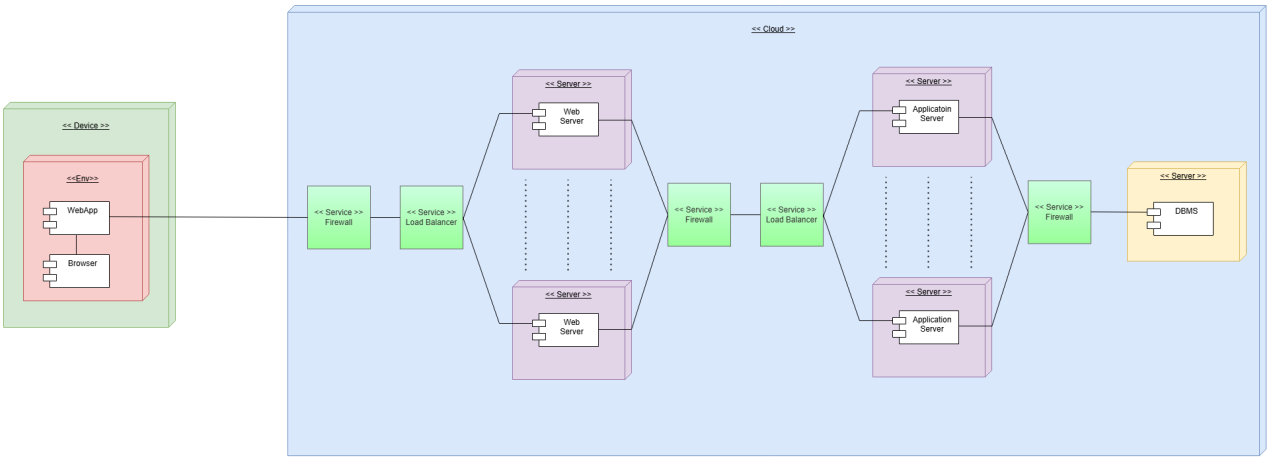


Figure 3: Component Diagram of the S&C server subsystem

2.4 Runtime View

This section contains the sequence diagrams of the most important operations of the system. The diagrams include the components that we have already described in the previous section and the external ones that are involved in the operations.

Moreover, the authentication mechanism for each user at the beginning of each operation is not explicitly included. It's assumed that the authentication and authorization will be implemented using a token-based process. These tokens are sent to the users each time they log in. They must be included with each request that requires authentication or authorization.

Student Registers

The diagram represented below shows the process of a Student creating an account to the S&C website. First, the student compiles the sign-up form with all the requested information. After that, the form information is checked and if the parameter result is valid, it is stored in the DB and a new account is generated. From there, the University login service, related to the university that the Student inserted in the form, is shown. If the Student logs successfully in its university account the AuthenticationManager triggers the verification email through the EmailServiceProvider.
Eventually, all the companies that advertise an internship that the system recognizes valid to recommend are notified of the new student subscription to the platform.

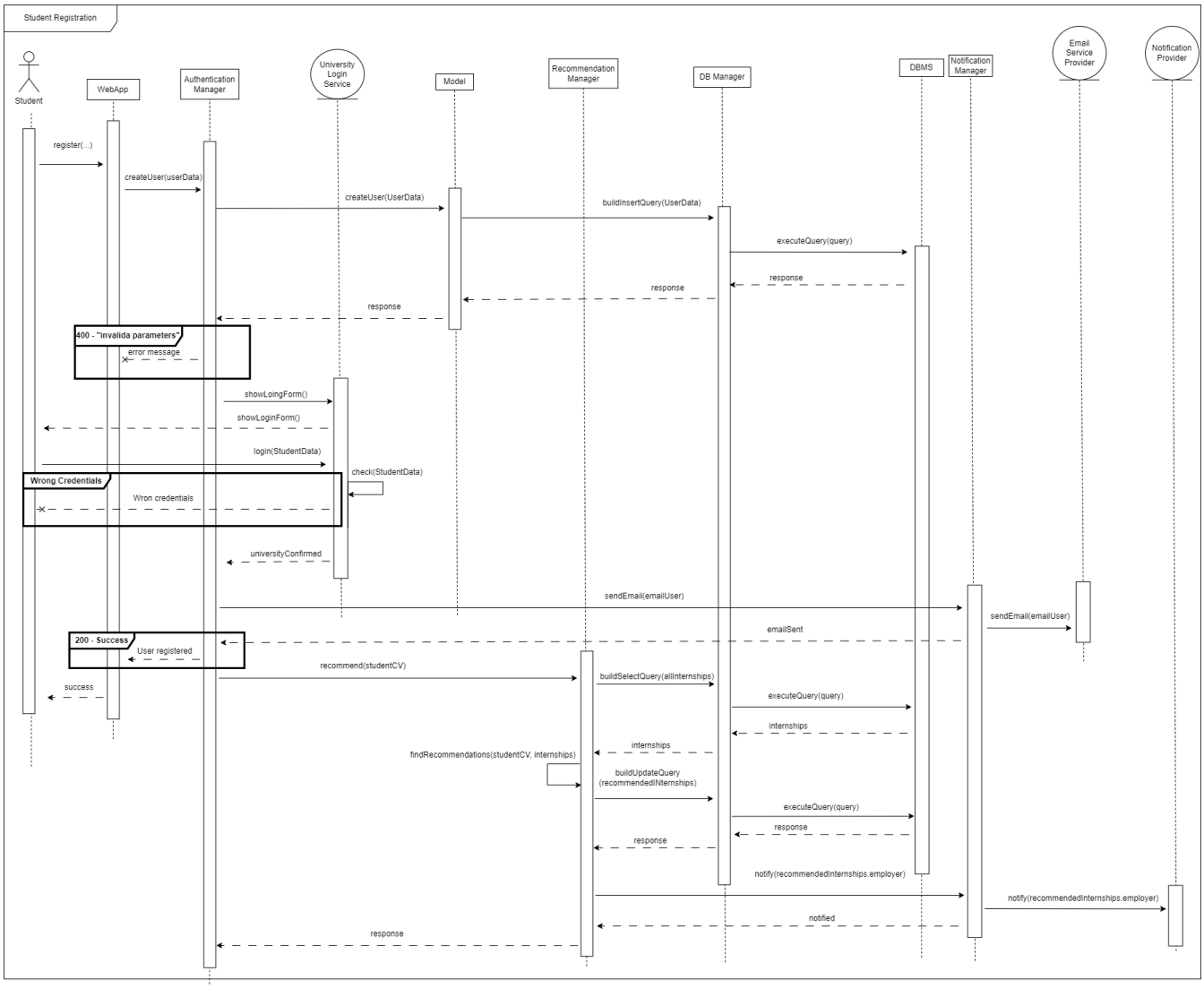


Figure 4: Student Registers to S&C

Company Registers

The diagram represented down below shows the process of a Company creating an account to the S&C website. First, the Company compiles the sign up form with all the requested information, then, as the request is submitted through the proper API call, the AuthenticationManager component handles the request, and, if the parameters of the request were valid, the AuthenticationManager triggers the verification email through the EmailServiceProvider.

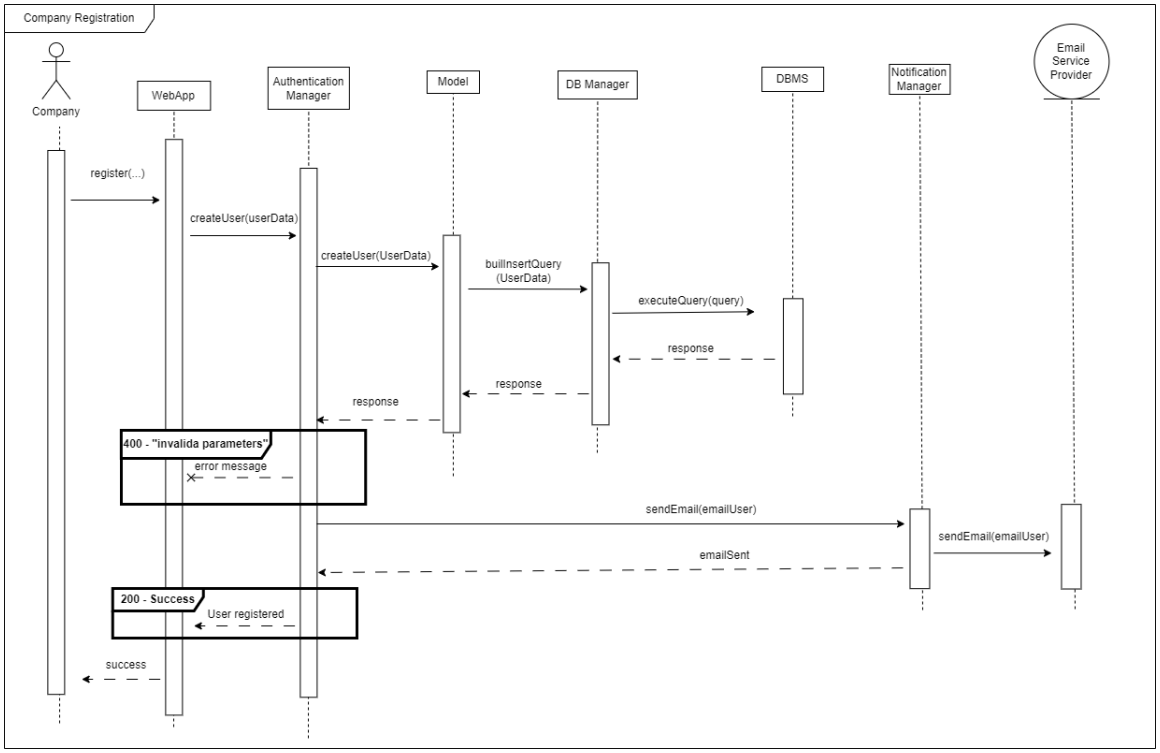


Figure 5: Company Registers to S&C

University Registers

The diagram represented down below shows the process of a University creating an account to the S&C website. First, the University compiles the sign up form with all the requested information. From there, the University information is checked using the National Education Dictionary. If the information results correct, then the information is stored in the DB and a new University account is generated. At last, the AuthenticationManager triggers the verification email through the EmailServiceProvider.

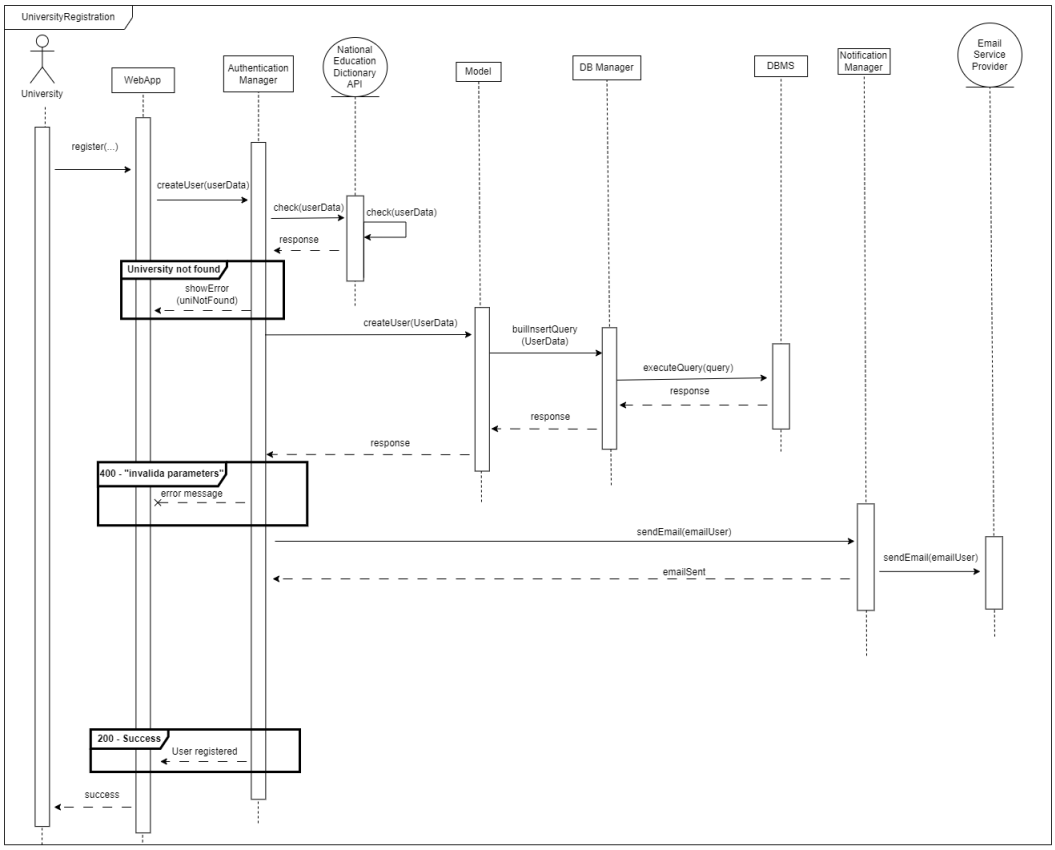


Figure 6: University Registers to S&C

User Views Template

The diagram below shows the process of a User (either student or company) visualizing the suggested template by the platform.

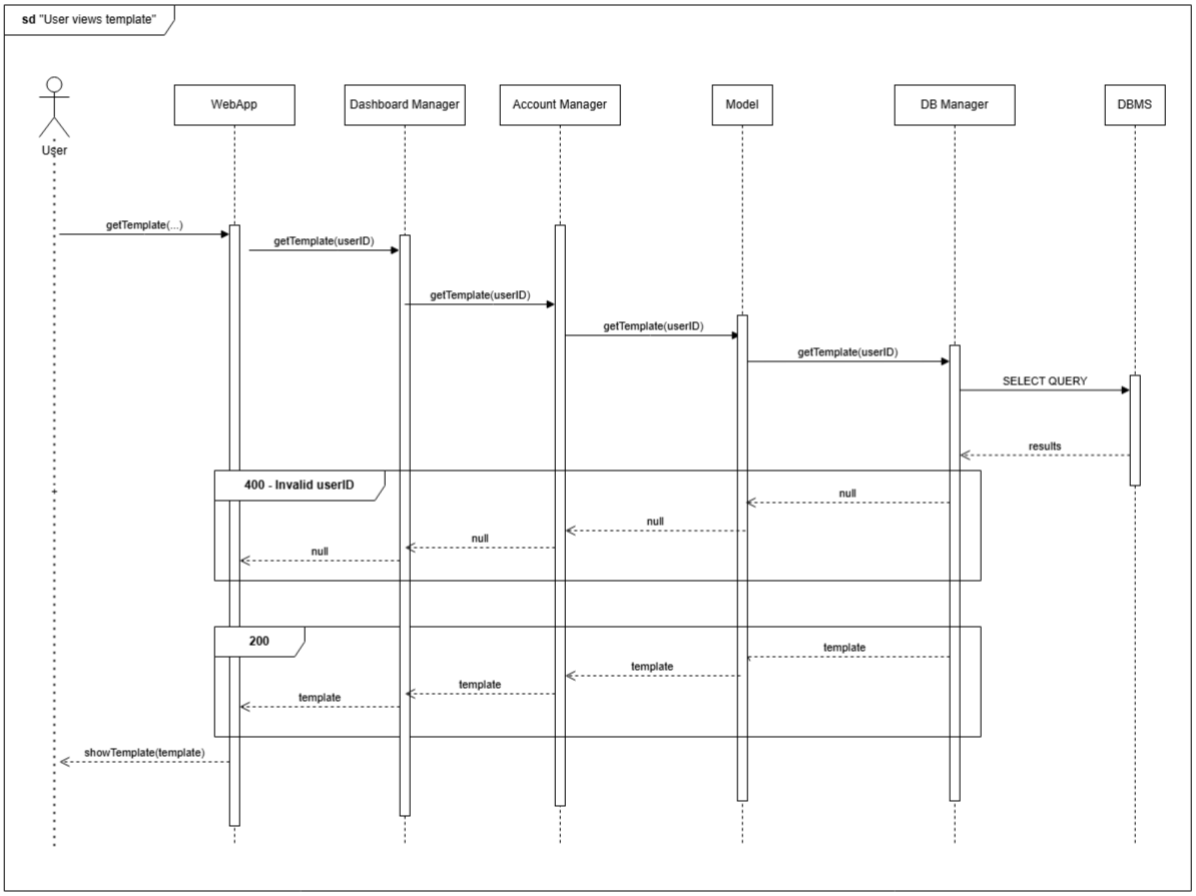


Figure 7: User Views Template

User Login

This diagram illustrates the login process within the system, regardless of the user type. We assumed that the user identifies their own type through interfaces and navigates the event flow using specific tabs and buttons to ultimately reach the login page.

The system verifies the user's credentials by querying with the database through the Authentication Manager. If the provided credentials (email and password) are incorrect, the user encounters an error message with error handling mechanisms as shown in the diagram. However, if the credentials are valid, the authorized managers retrieve the user's Id from the database and redirect to the user's home page by using that.

In addition, the diagram includes a loop representation to prevent that the potential database errors do not affect the system's overall functionality.

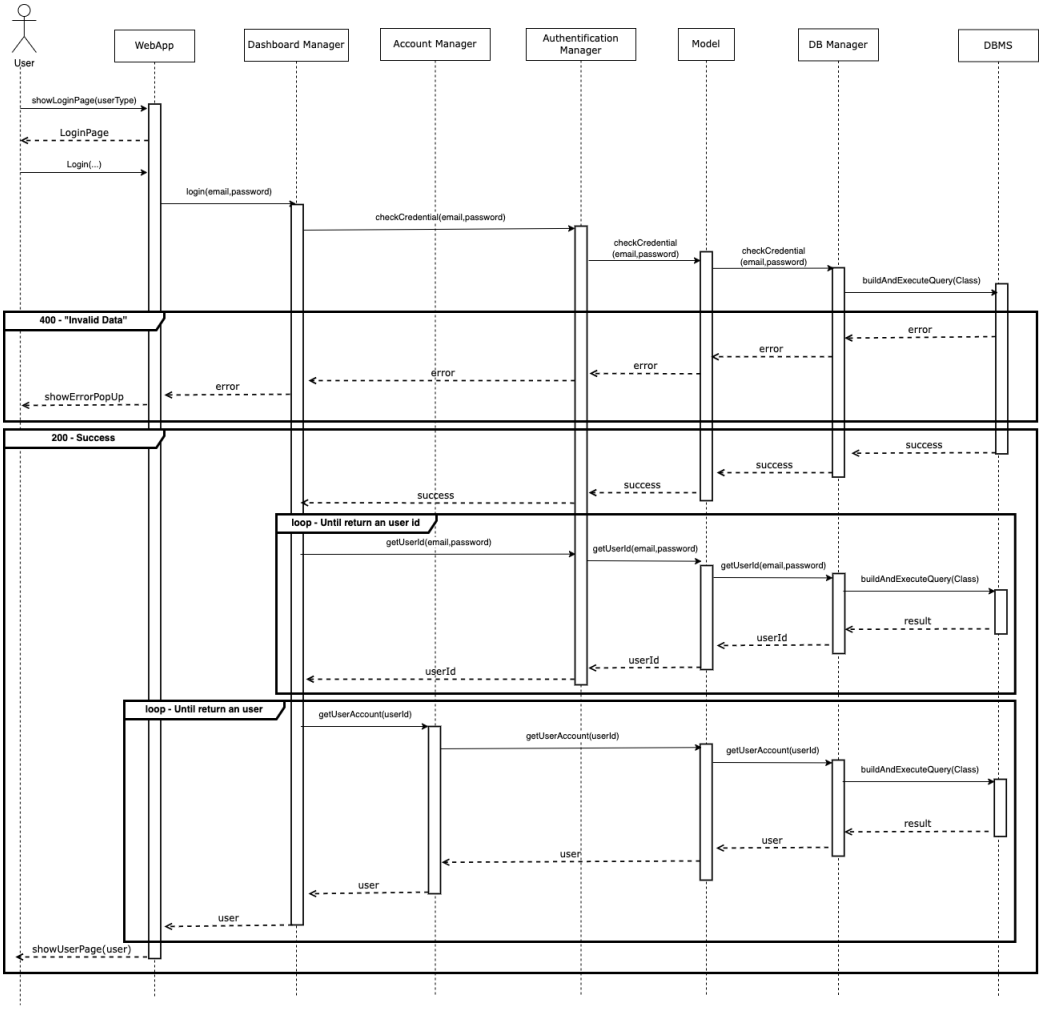


Figure 8: User Login

Student Updates CV

The student initiates the process by uploading their CV, which is handled by the WebApp and subsequently passed to the Dashboard Manager, Account Manager, and Model components for validation and processing. If the data are valid, the CV is updated in the database using a query managed by the DB Manager and the DBMS. Upon successful update, the Recommendation Manager generates internship recommendations based on the updated CV, retrieving relevant internships and notifying associated employers. In case of an error (e.g., incorrect data), the system responds with an appropriate error message. Notifications are sent to employers through the Notification Provider, ensuring the flow of updates and recommendations.

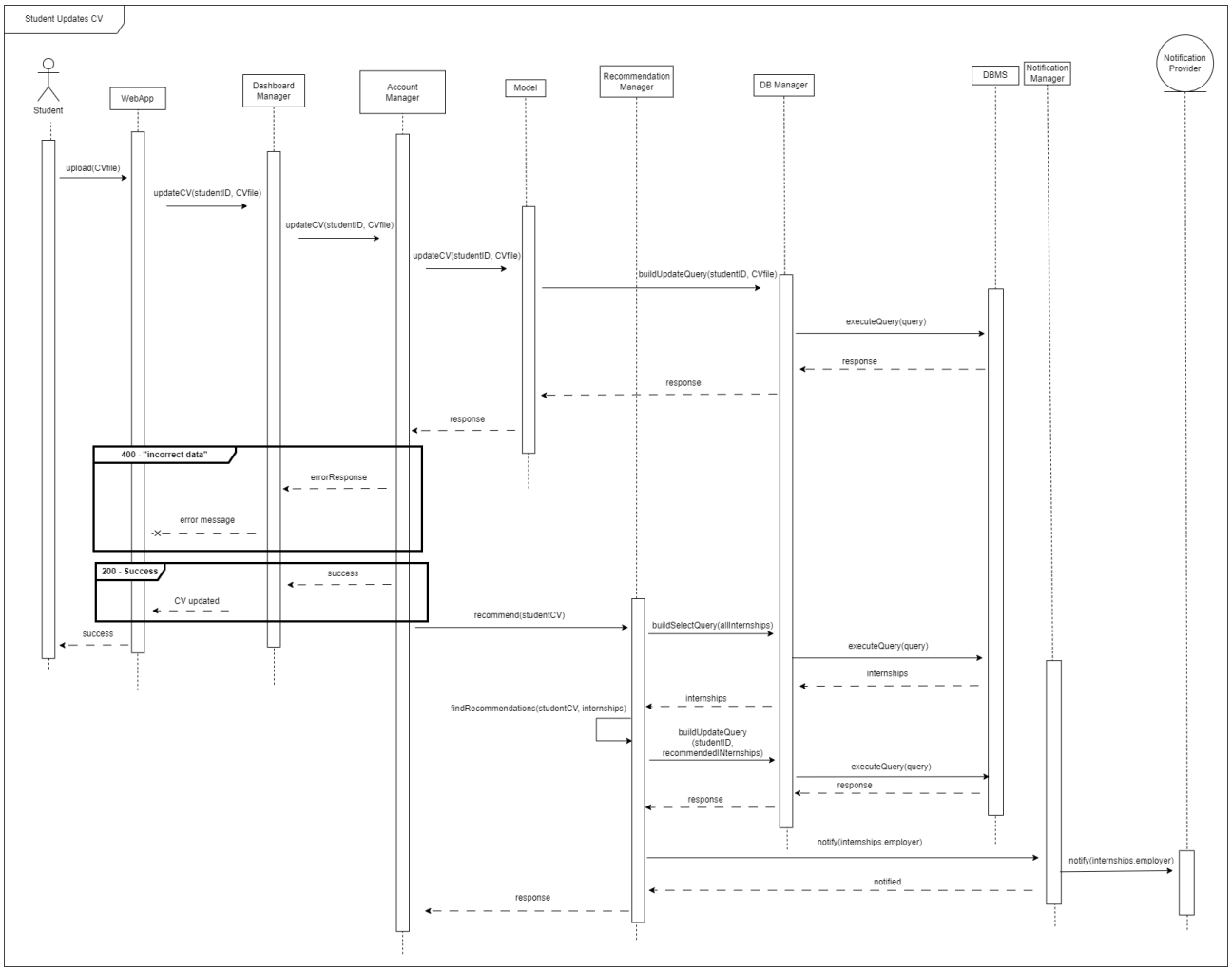


Figure 9: Student Updates CV

Student Changes University

The student begins the process by submitting a form with their new email associated to their new university to the WebApp. The form data are forwarded to the Dashboard Manager and Account Manager for validation and processing. If the parameters are valid, the system interacts with the University Login Service to verify the student's credentials and confirm the university enrollment. Once confirmed, the Account Manager updates the student's university information in the database by sending a query to the DB Manager, which is executed by the DBMS. Upon successful update, the system sends a success response to the student. In case of errors, such as invalid parameters or incorrect credentials, the system returns an appropriate error message through the WebApp. This ensures clear communication with the student at every stage of the process.

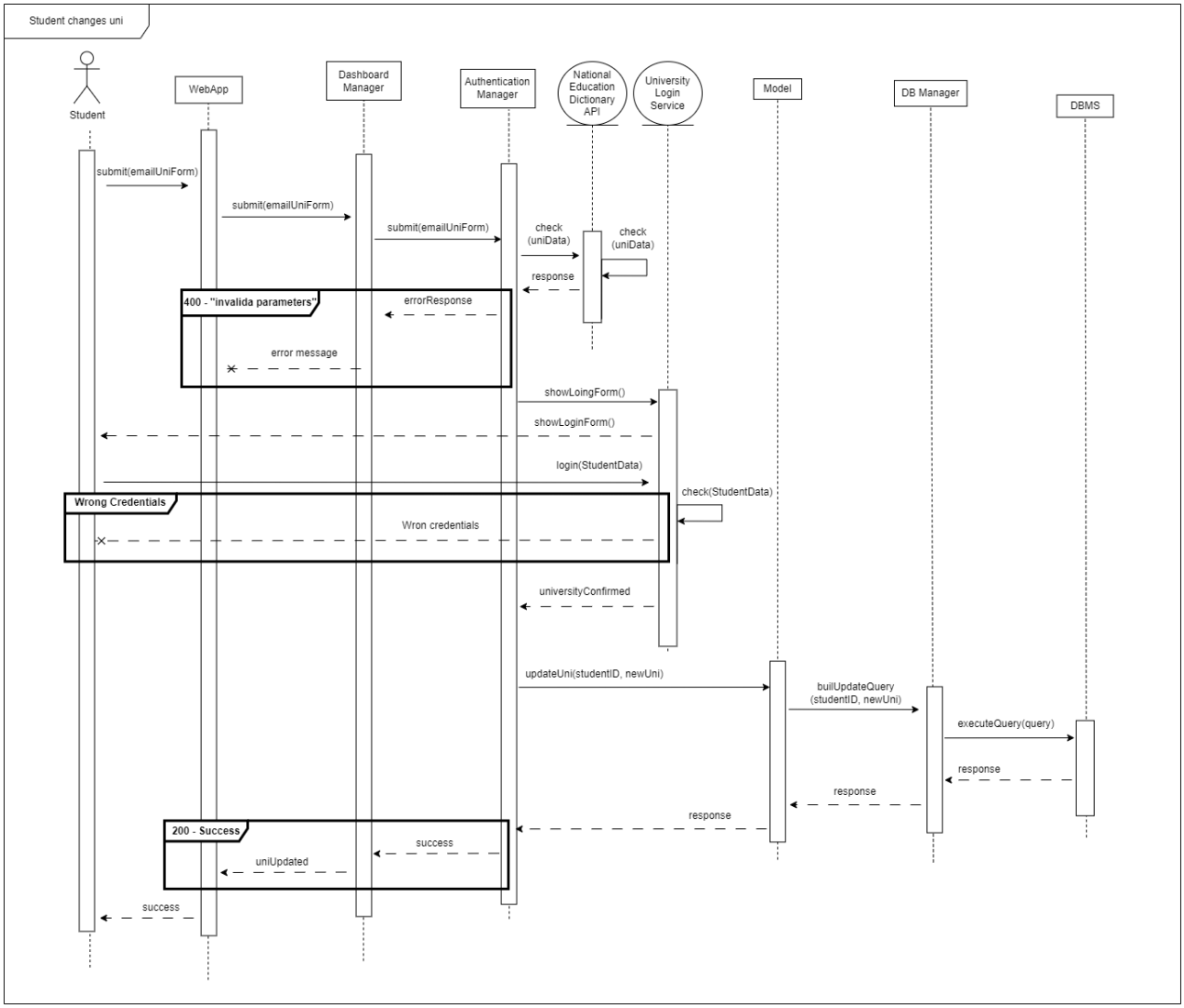


Figure 10: Student Changes University

Company Posts An Internship Advertisement

This diagram illustrates the scenario of posting an internship advertisement through the system according to the company's needs. The process begins with the company navigating to the appropriate page via the system interfaces to initiate advertisement creation. We assumed that all required fields within the interface are completed. The system first receives the information and advertisement details filled out by the company user and save them into the DBMS through the relevant managers using the parameters adInfo(advertisementInfo) and companyId. Upon successful creation of the advertisement within the system, relevant student profiles are identified and recommended through the recommendation manager for the advertisement, and these suggested students are saved to the database. Furthermore, the system notifies the identified students of the newly created advertisement, which might align with their interests, through the notification manager and provider. The diagram also incorporates error-handling mechanisms to improve system security and efficiency.

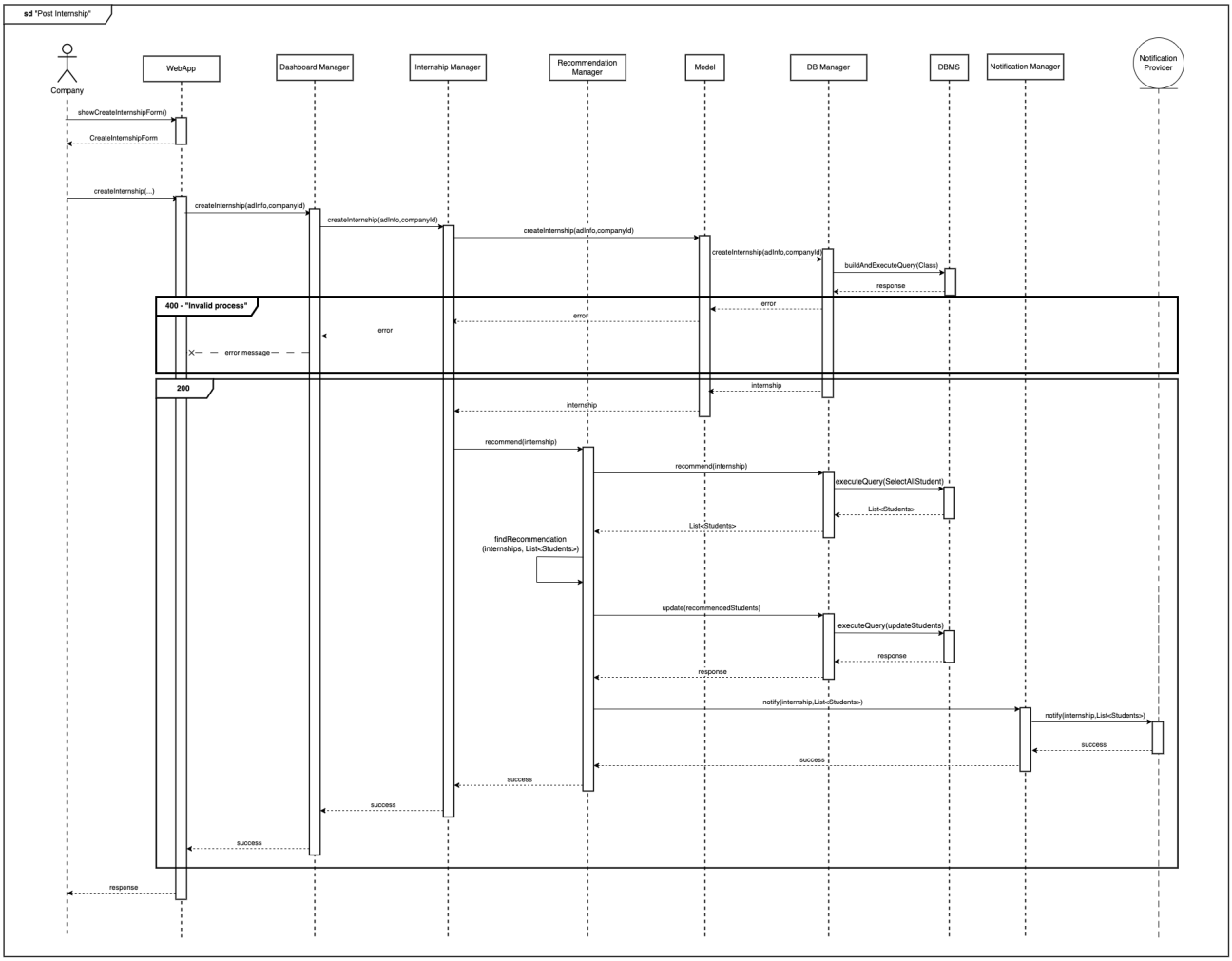


Figure 11: Company Create Internship

User Views Relevant Internships

The diagram below shows the process of a user visualizing the list of internships he's interested in.
In the case of a University, it retrieves all the internships the students from that university were selected for

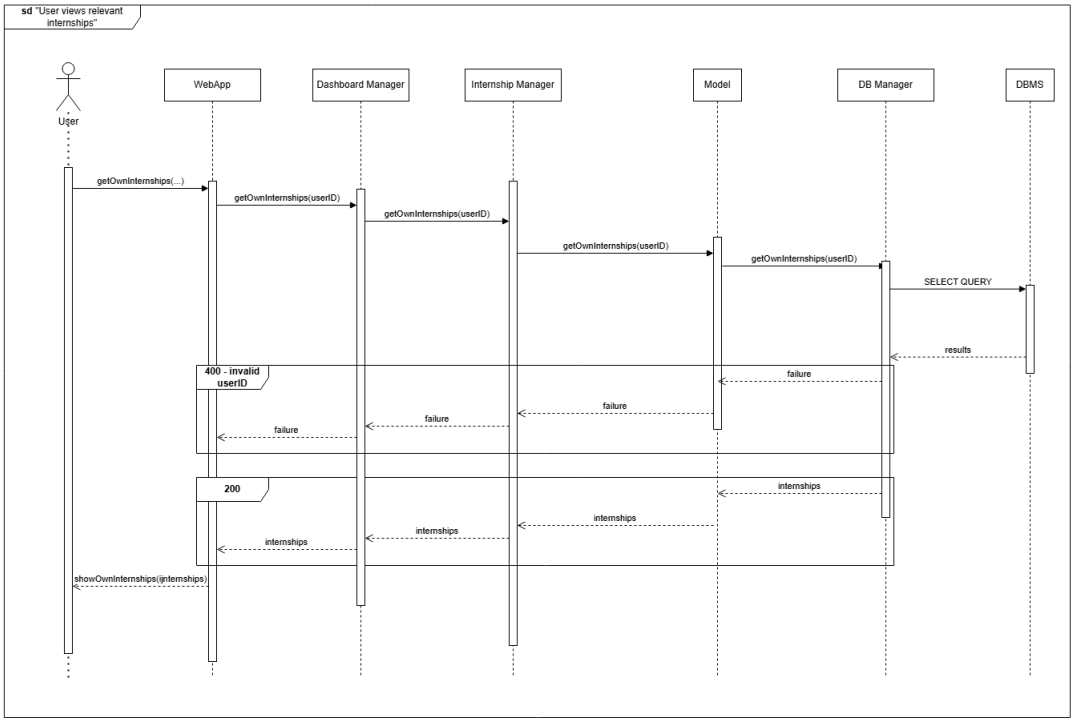


Figure 12: User Views Relevant Internships

Student Views and Apply For The Advertisement

This diagram illustrates the scenario in which students utilize the system's homepage interface to list general internship advertisements and view their details. The process begins with the student accessing the system's homepage, where the system retrieves a list of previously shared advertisements from the Database Management System (DBMS) and presents it to the user. If the list contains advertisements, we assumed that the student will proceed to view the details of a selected advertisement as part of the scenario.

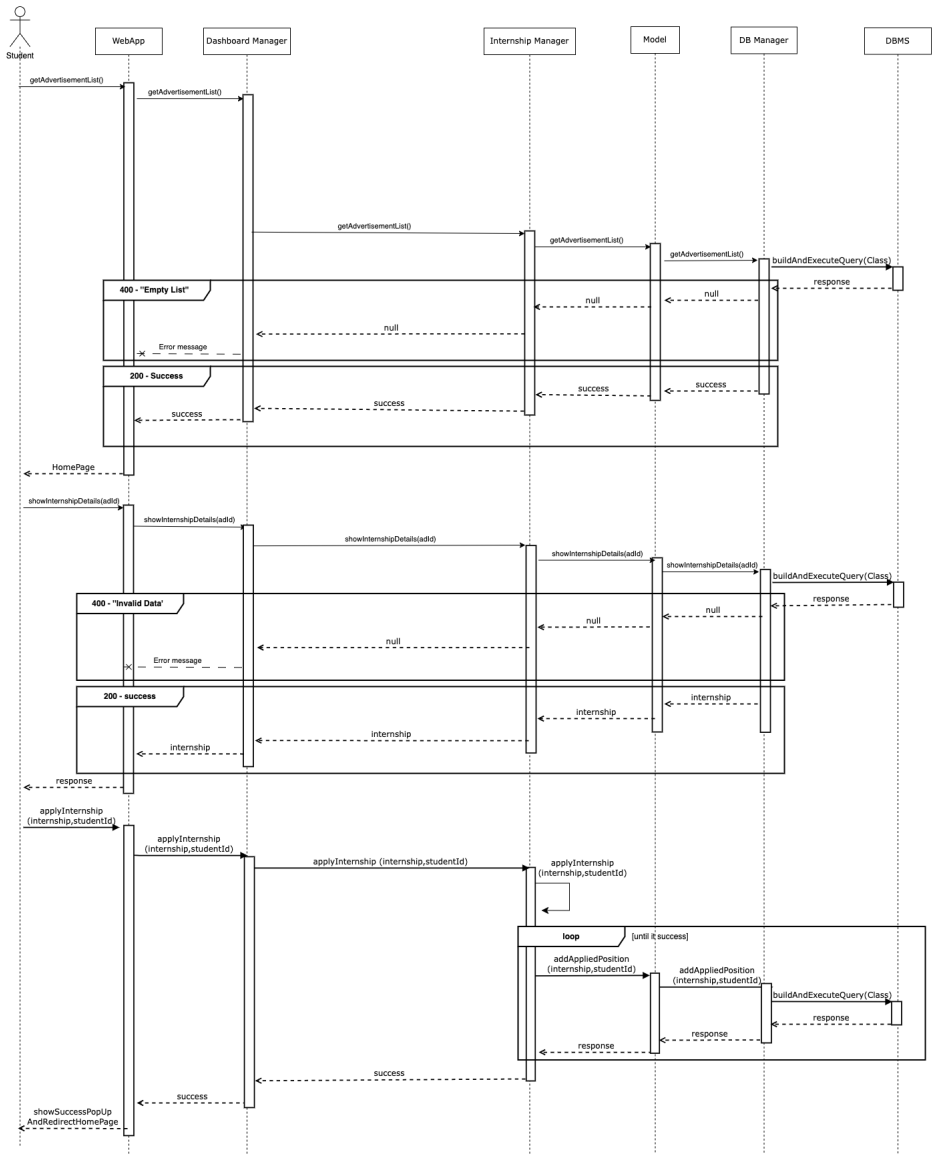


Figure 13: Student Views Advertisements

Company Views Recommended and Applying Students

This diagram represents the scenario in which companies view a list of advertisements previously shared through their own profiles. The company navigates to the "Manage Internship" page, and the system sends a request to the DBMS through the relevant managers using the companyId parameter. The system retrieves the list of advertisements and displays it on the interface. The diagram also incorporates a condition to check whether any advertisements exist. If the list is available, the scenario proceeds with the company selecting one of the advertisements.

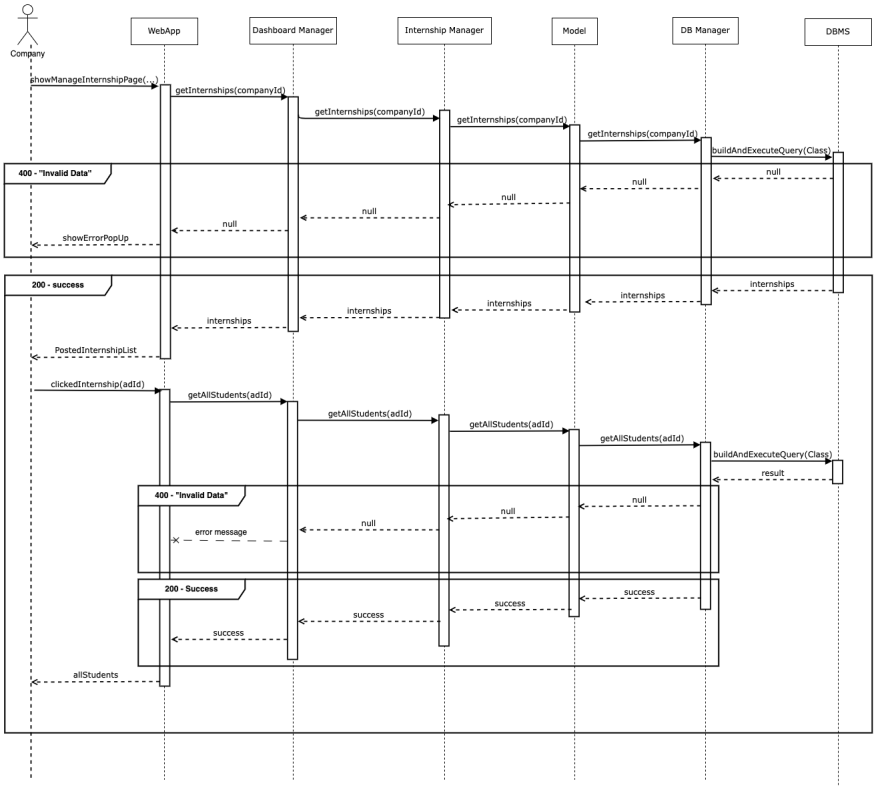


Figure 14: Company Views Recommended and Applying Students

Company Views The Student Details and Accepts

This diagram is presented as a continuation of the scenario shown in Figure 19. In summary, the company has accessed the advertisements they previously shared and navigated to the desired advertisement cell. In this scenario, we assumed that the company selects a student's profile from the "All Candidates" list via the interface.

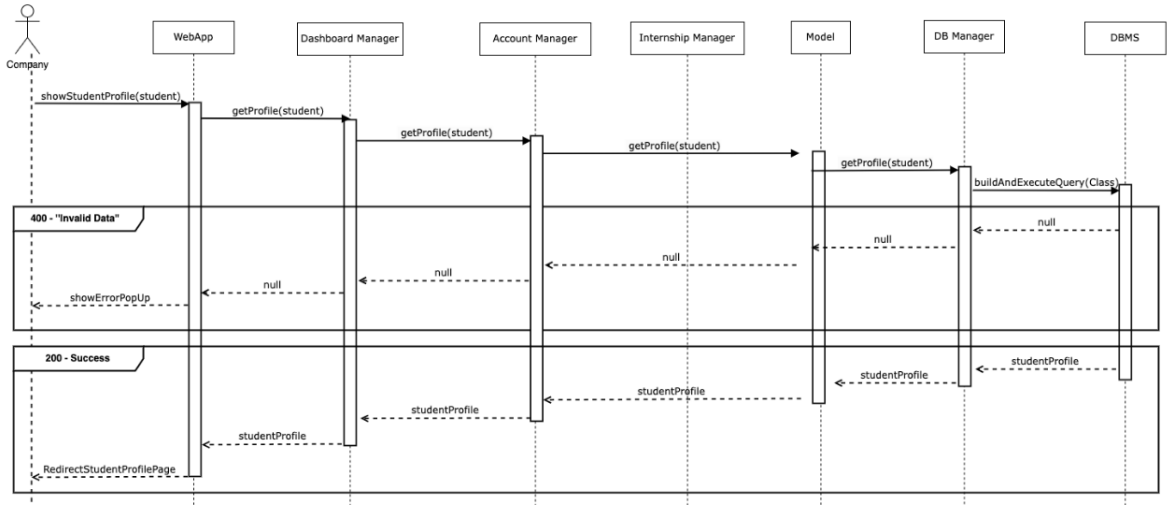


Figure 15: Company Views The List of Student

User Accepts

The diagram below shows the process of a User (either student or company) accepting the other. Both studentID and internshipID are needed for the operation.

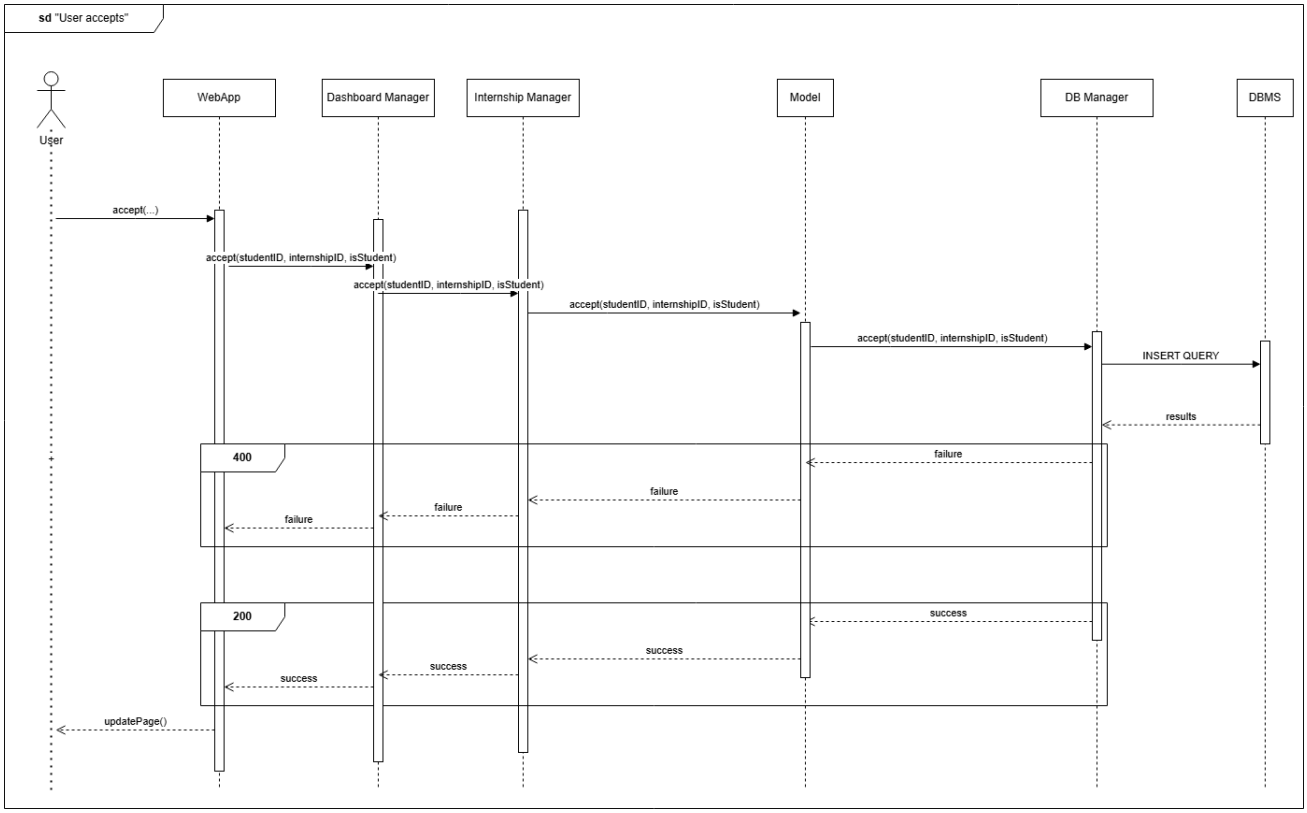


Figure 16: User Accepts

User Gives Feedback About Recommendations

The diagram demonstrates the process of feedback on recommendations, regardless of their user type. This attribute can be use only for student and company user types, so the diagram shows some conditions to handle error for unauthorized user types.

The process begins with functions including the internship class (containing attributes such as companyId, advertisementId, or studentId) , userId and a boolean response data type. The updateRecommendationStatus function allows each user to provide feedback only once, and it records that in the database. In addition, the system analyzes the feedback collected to optimize the recommendation mechanism, allowing the system to offer more accurate and personalized suggestions over time. We also assumed that the recommendation manager owns the analyzed internship list and then system updates the recommendation list for an user.

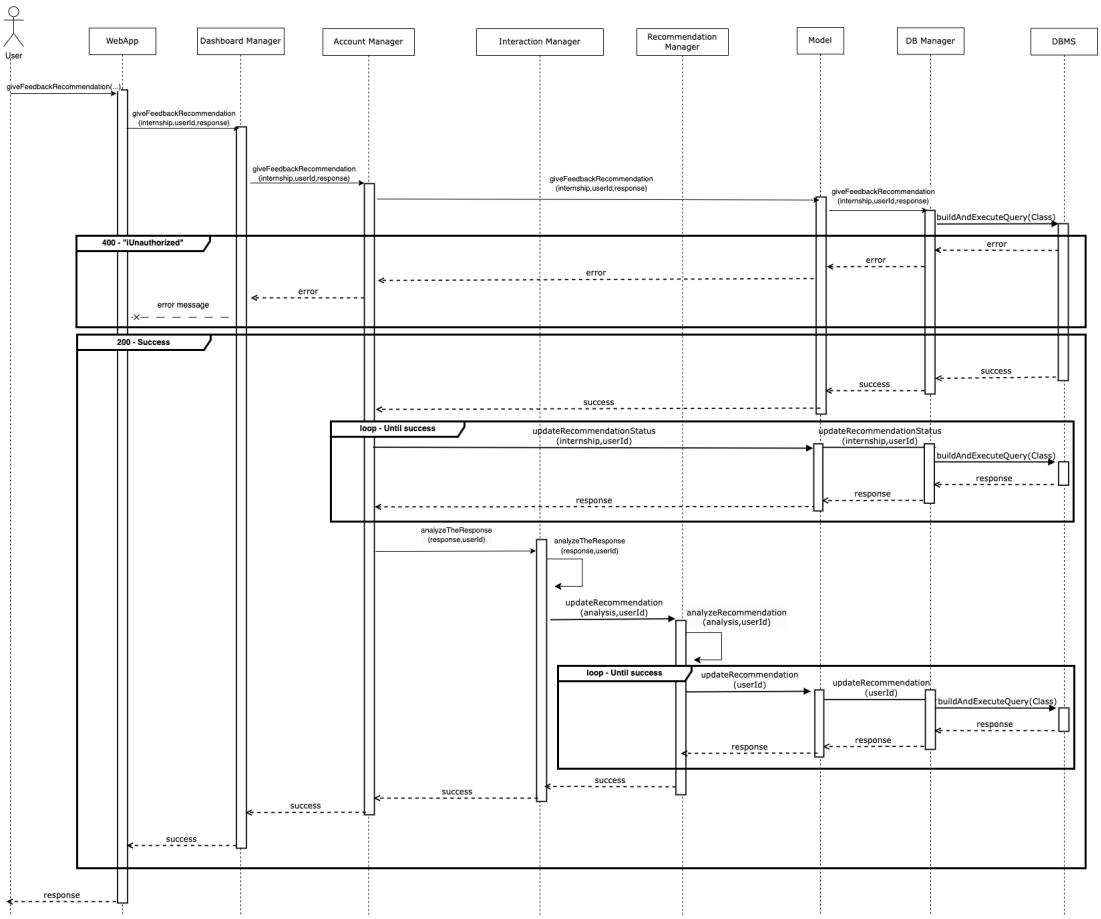


Figure 17: User Recommendation Feedback

Company Sends Interview Form

The company initiates the process by submitting an interview form to the WebApp. The form data is forwarded to the Dashboard Manager, Internship Manager, and Model components for validation and processing. If the parameters are valid, the system builds an insert query and sends it to the DB Manager, which is then executed by the DBMS to save the interview time slots. Upon successful saving, the system retrieves the list of candidates for the interview from the database and sends them notifications about the available time slots. In case of errors, such as invalid parameters, the system returns an appropriate error message through the WebApp. This ensures clear communication and efficient execution of the interview scheduling process.

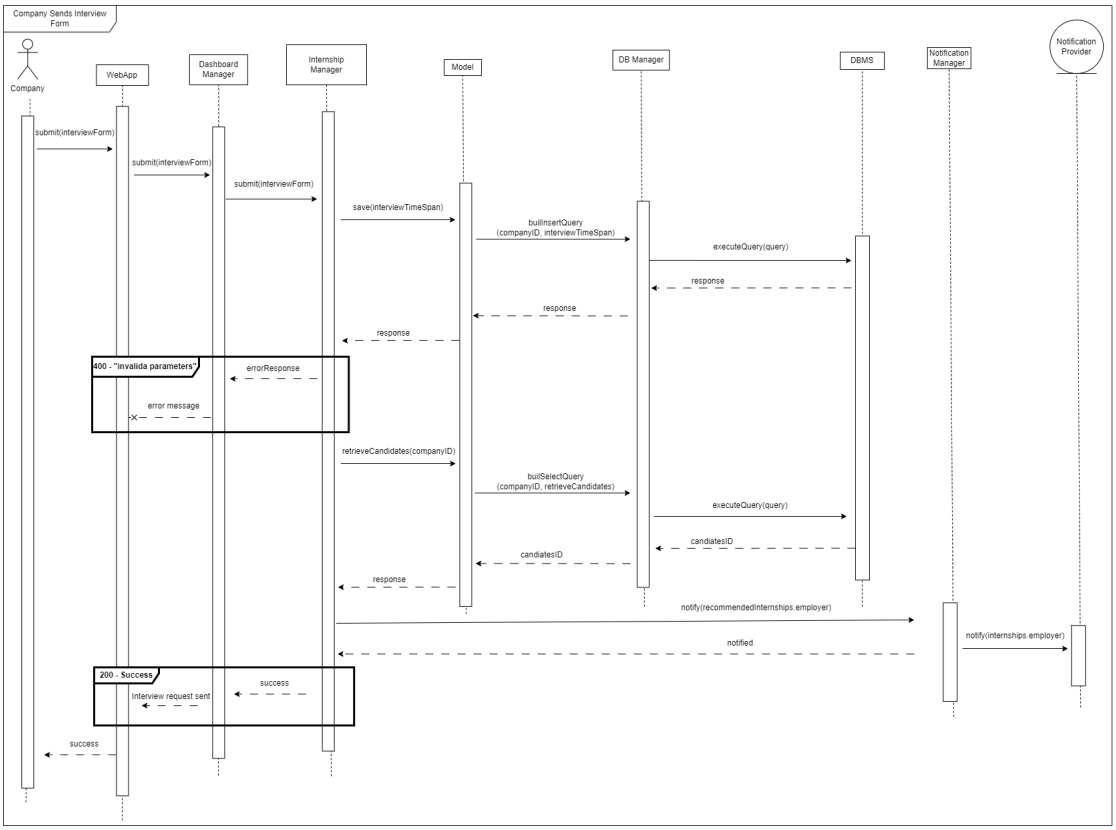


Figure 18: Company Sends Interview Form

Student Selects Interview Date

The student submits a time slot form via the WebApp, which is then handled by the Internship Manager. This manager interacts with a Model component to save the student's ID, internship ID, and the chosen time slot. A database insert query is built and executed by the DB Manager, interacting with the DBMS Manager. The process can result in either a 400 error (invalid parameters) or a 200 success message. Upon success, the Internship Manager retrieves the company information associated with the internship from the database. This information is used by the Notification Manager to notify the company through a notification provider.

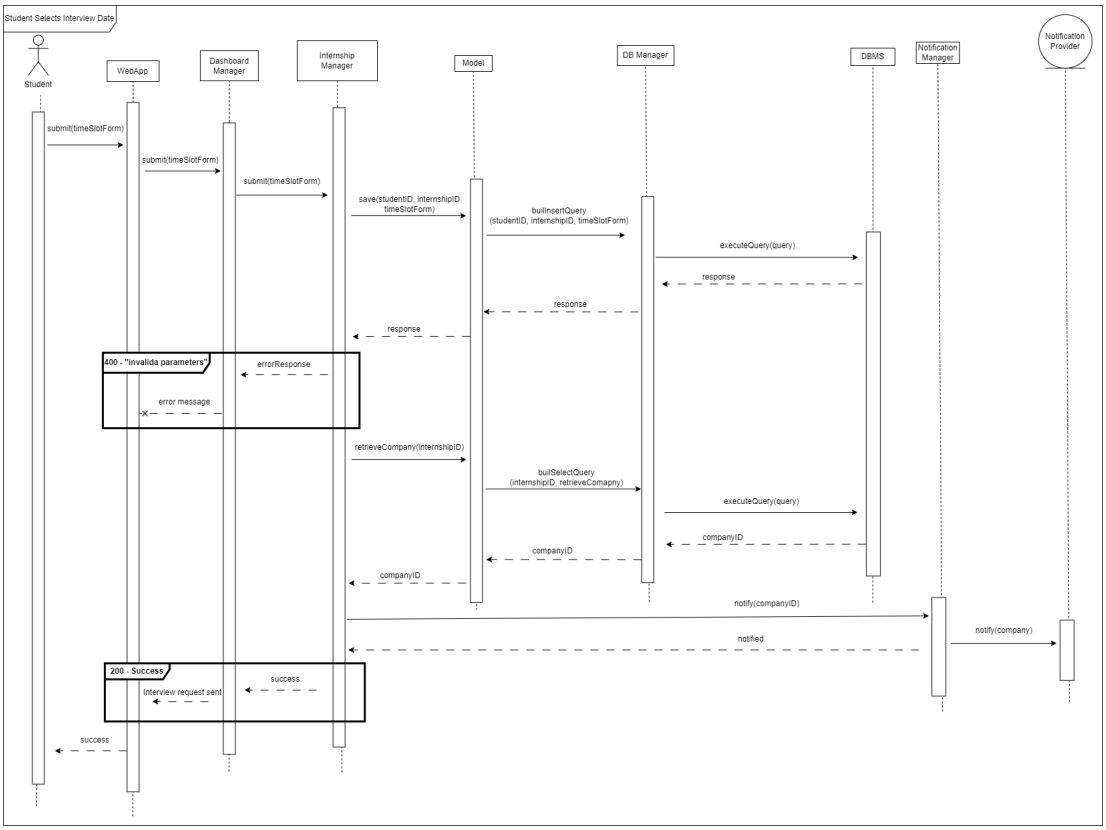


Figure 19: Student Selects Interview Date

Company Selects Candidate

The company sends a request containing the candidate ID and internship ID via the WebApp, which is received by the Internship Manager. This manager then uses a Model component to handle the data. A database insertion query is constructed and executed by the DB Manager, interacting with the DBMS. The process can result in a 400 error (invalid parameters) or a 200 success message. Upon successful candidate selection, the Internship Manager triggers a notification process. The Notification Manager then notifies the selected candidate through a notification provider.

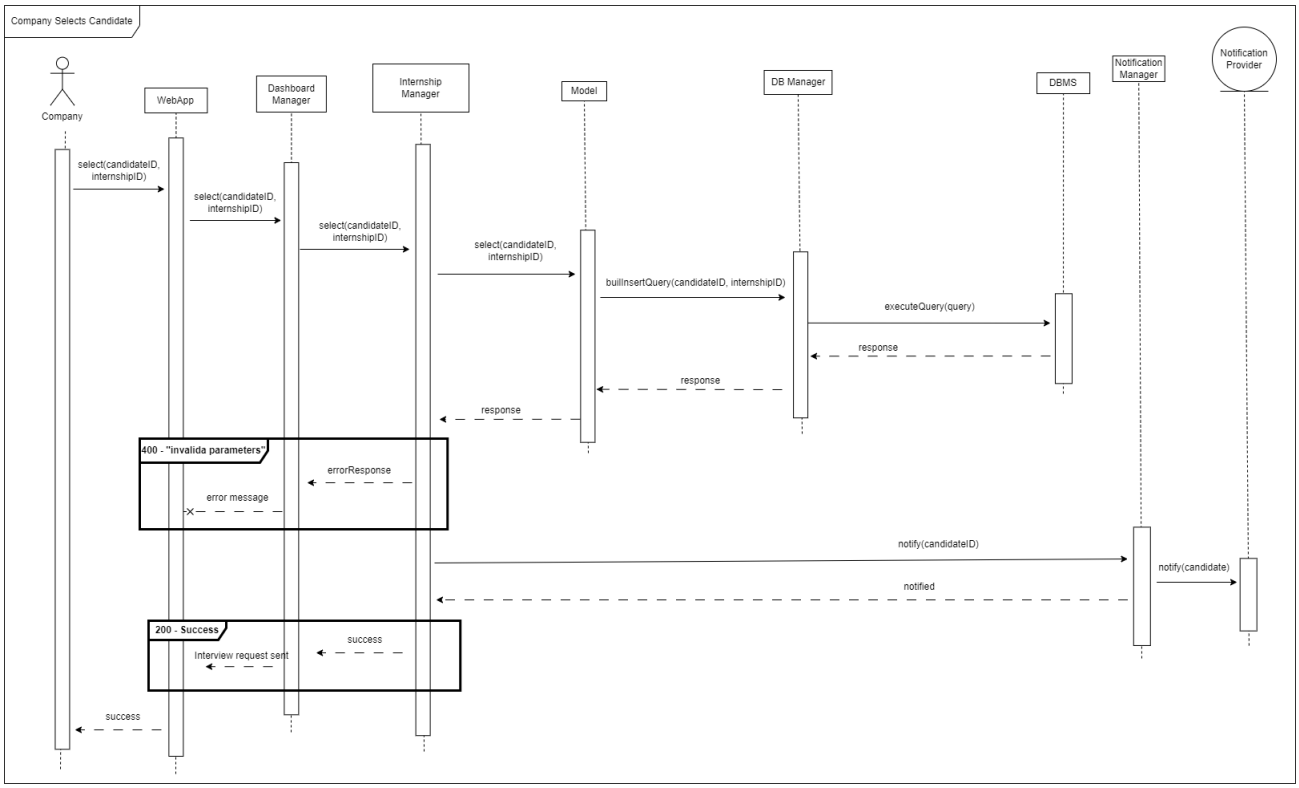


Figure 20: Company Selects Candidate

Student Decides Offer

This diagram illustrates the process by which a student responds to a company's internship offer within the system. The sequence begins when the student navigates to the "Matched Internship" page from the "My Internship" page. The system retrieves the list of matched internships using the studentId through the relevant managers through the database and displays the results on the interface. Error-handling mechanisms ensure that only valid matches are presented, and if no matches are found, an error message is displayed to the user. Only and only if an internship offer requires a response which shows in the alt section in the diagram, the system records the student's decision in the database within a controlled loop to prevent potential errors. Upon acceptance, the advertisement status is updated, the changes are saved, and a notification is sent to the relevant company using the companyId and advertisementId parameters from the internship class. In the event of rejection, the advertisement is removed from the student's "Applied Positions" page. After processing the response, the system updates the interface, allowing the student to view the revised information while also preparing the database for the other related pages.

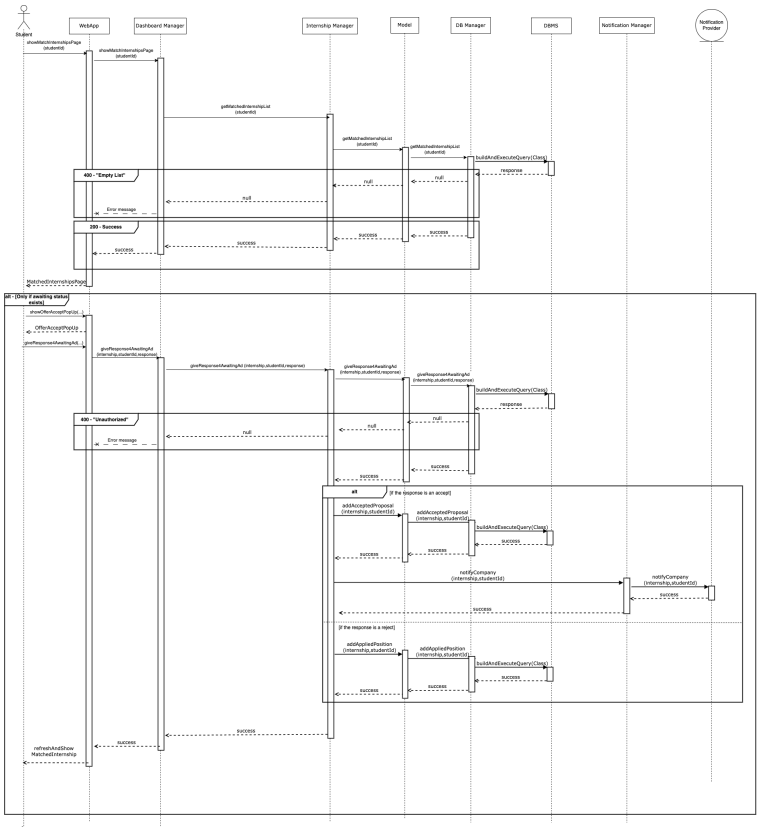


Figure 21: Student Accepts Or Rejects Internship

User Views Comments

The diagram below shows the process of a User visualizing comments of an internship. If the internshipID is not valid, the process is interrupted.

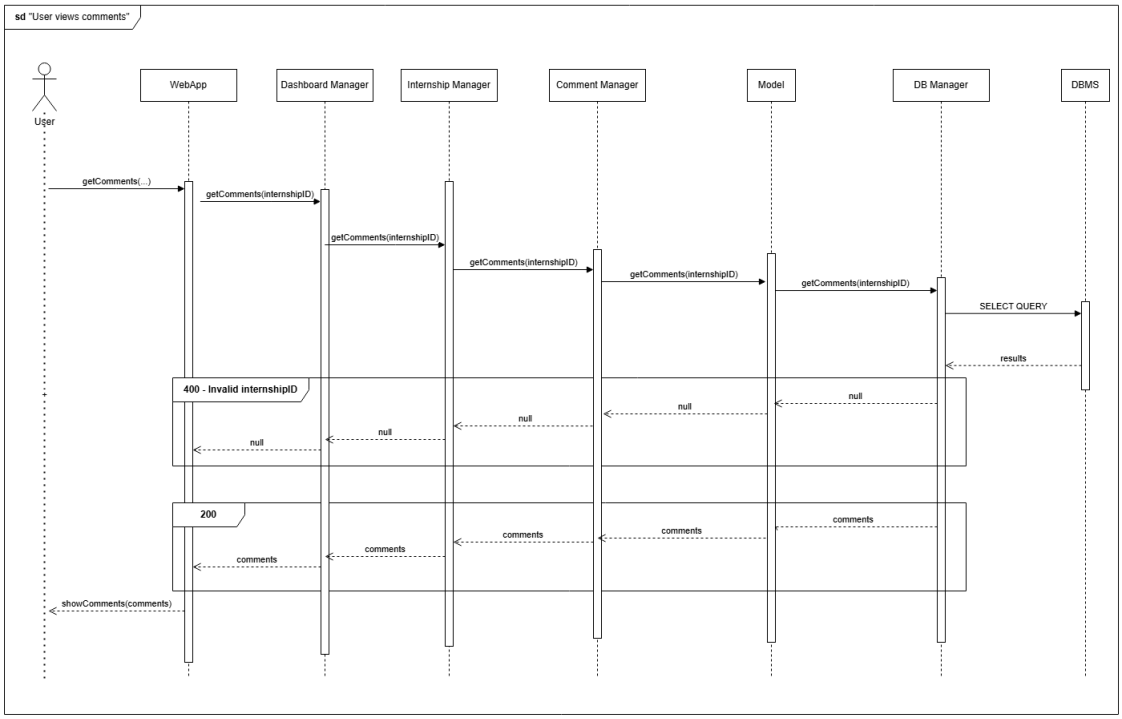


Figure 22: User Views Comments

User Writes Observation

The diagram below shows the process of a User (either student or company) writing an observation about an on-going internship. In this case no notifications are sent, which component will be important for the complaint part.

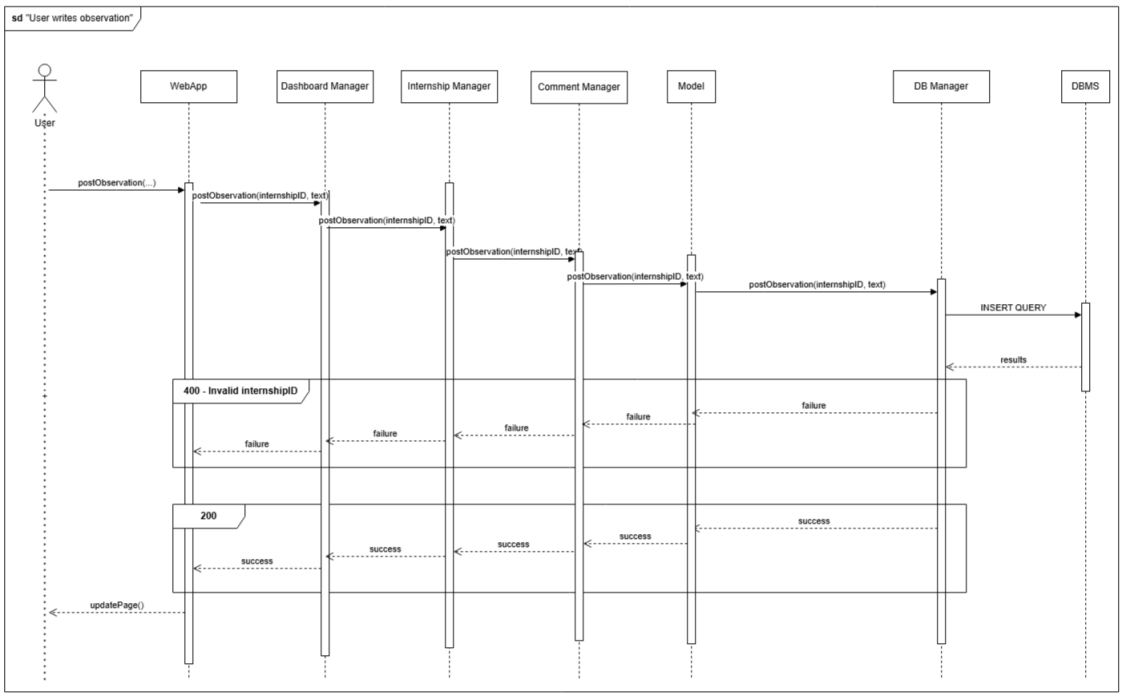


Figure 23: User Writes Observation

User Writes Complaint

The diagram below shows the process of a User (either student or company) writing a complaint about an on-going internship. After the complaint has been posted successfully, the system will send a notification to the university of the student that is doing the internship, through the notification provider.

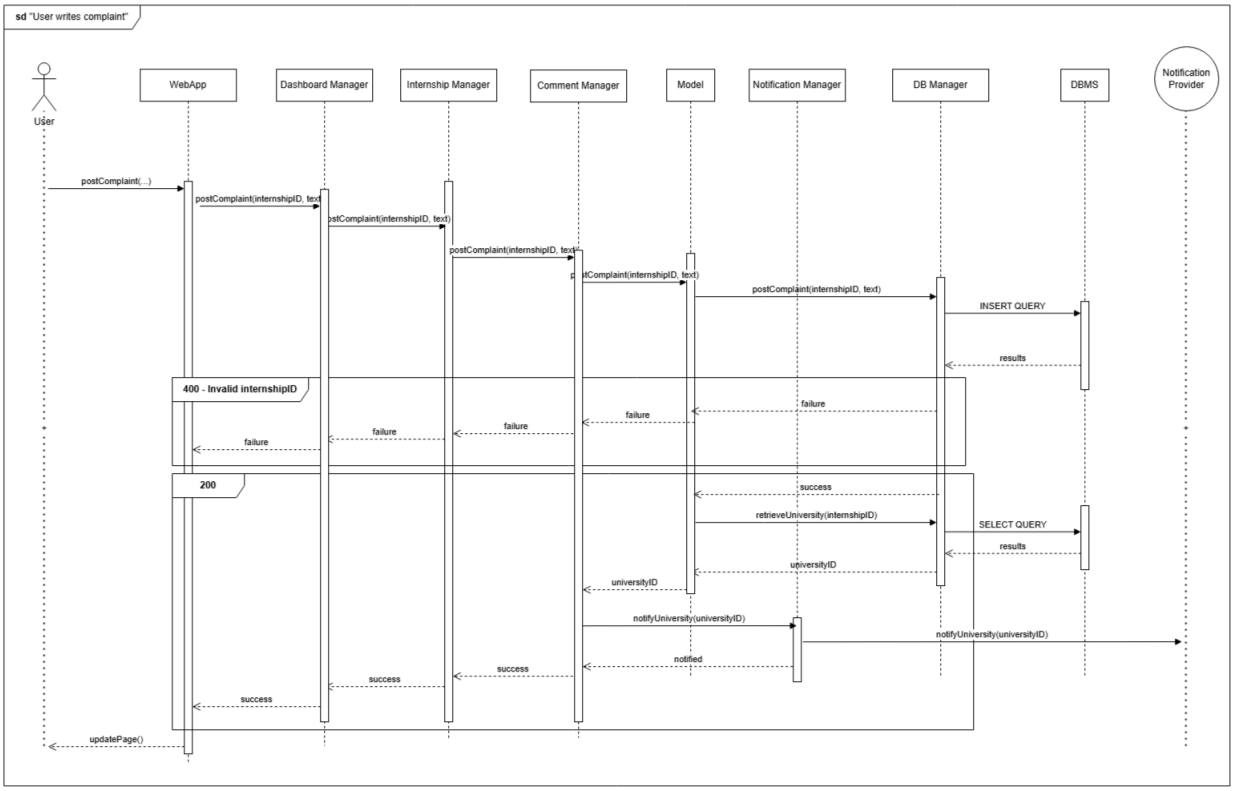


Figure 24: User Writes Complaint

University Interrupts Internship

The diagram below shows the process of a University interrupting an on-going internship. If the internshipID is not valid, the process interrupts. The platform will send notifications to the company and the student relative to that internship.

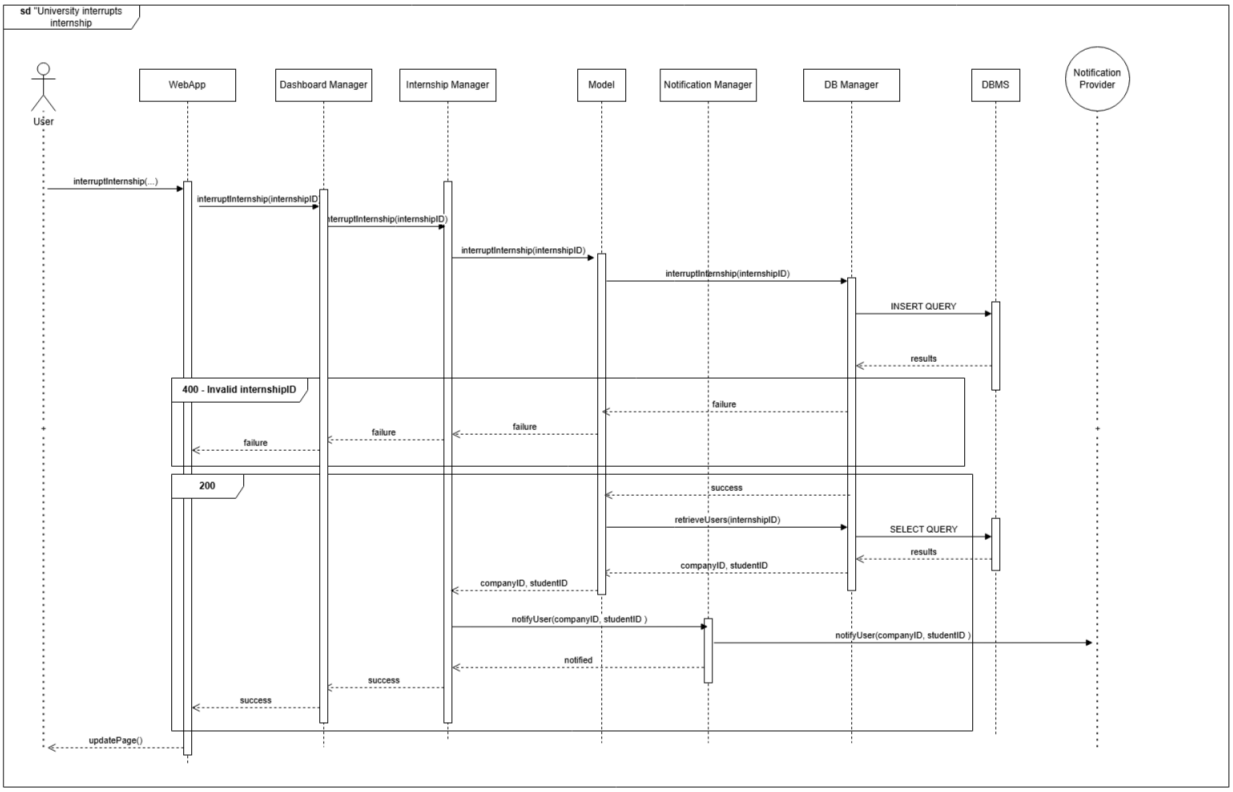


Figure 25: University Interrupts Internship

User Gives Feedback About The Experience

The diagram shows that users must navigate to the "Accepted Proposal" page to provide feedback on their internship experience. Error-handling mechanisms, as in previous diagrams, ensure system security by addressing scenarios such as empty lists or unauthorized access. Functions utilizing the UserId parameter connect to the database through authorized managers, retrieving lists of accepted proposals or associated companies for display on the user interface. The mechanism within ”alt” is designed to operate exclusively for cases that require feedback.Submitted responses are stored in the database, lists are updated via authorized managers, and the Interaction Manager processes the feedback for analysis. This interaction improves recommendation mechanism and helps to make it more correctly to the users, and finally the user interface is refreshed to reflect the updated status.

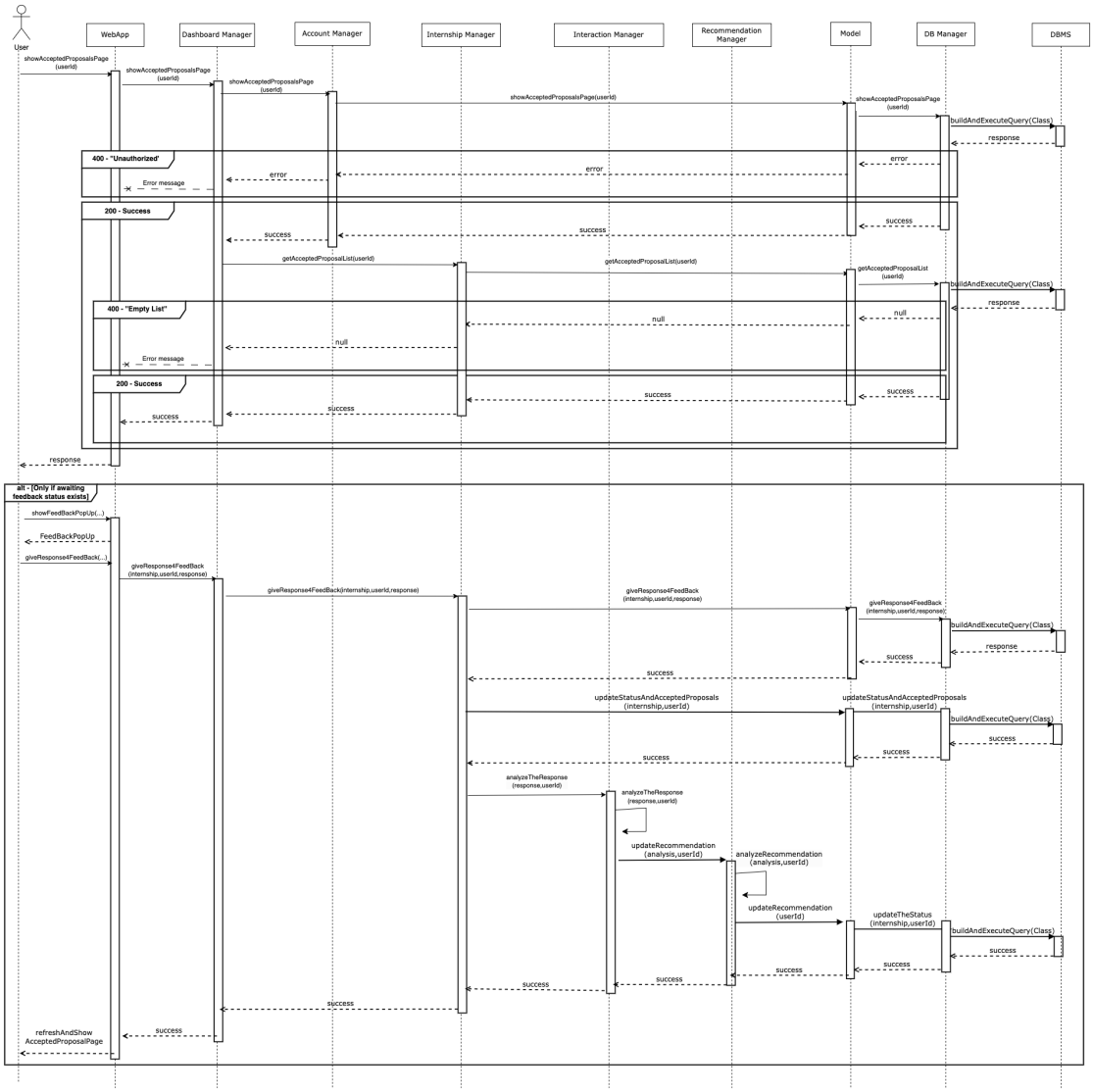


Figure 26: User Feedback

2.5 Component Interfaces

In this section, we use a class diagram to illustrate the component interfaces. The methods are presented as clearly as possible, and we expect their functionality to be easily understandable.

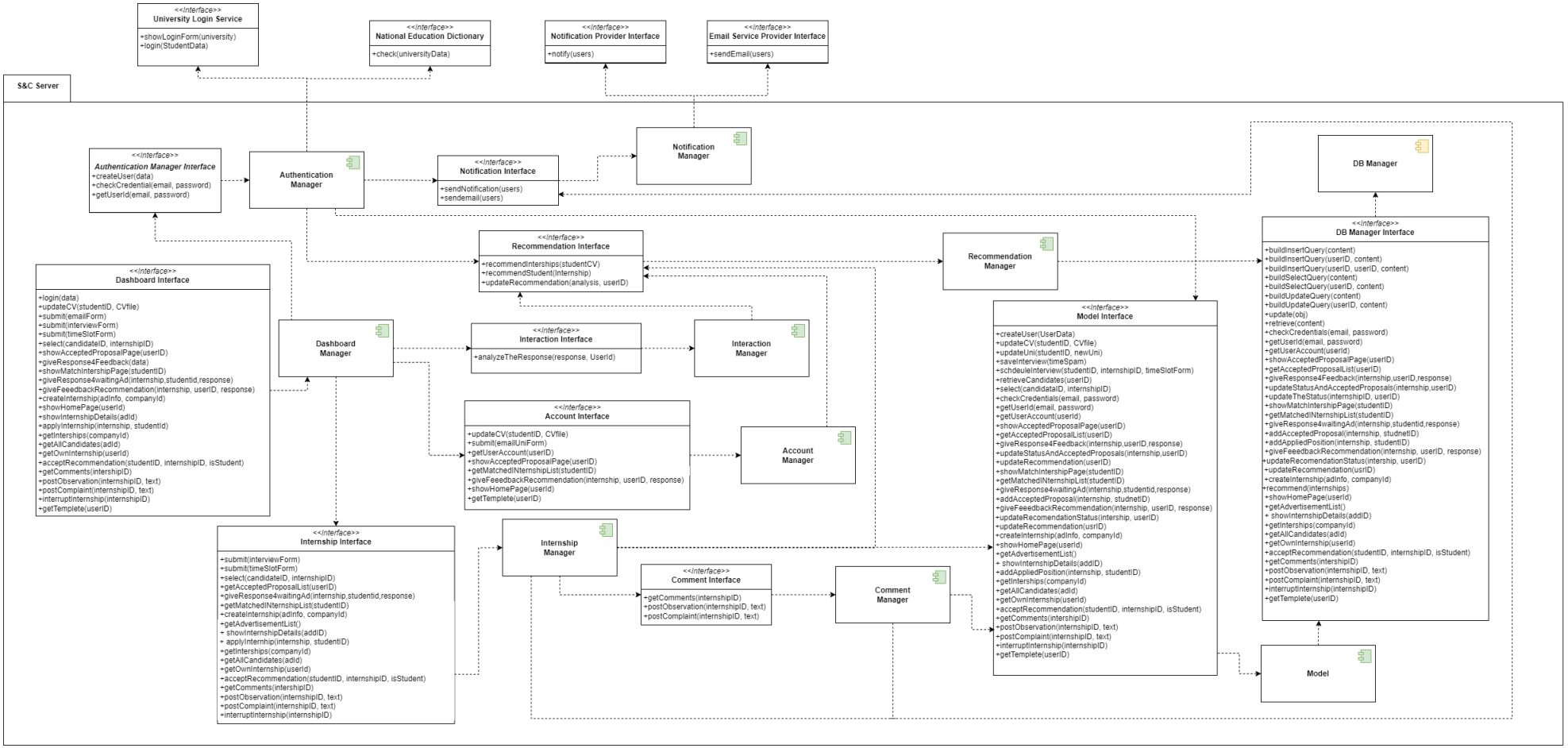


Figure 27: Class diagram with interfaces of the S&C System

2.5.1 API Endpoints

In this section the API endpoints are presented. The focus is on the method used, on the parameters required and on the response.

POST api/users/register

Request Body: User: Object

200 Response: message: String("User successfully registered")

400 Response: error : String("Invalid request data")

POST api/users/login

Request Body: User: Object

200 Response: message: String("User successfully logged in")

400 Response: error : String("Invalid request data")

PUT api/student/{studentId}/updateCV

Request Body: CVfile: Object, studentId: int

200 Response: message: String("CV successfully uploaded")

400 Response: error : String("Invalid request data")

PUT api/student/{studentId}/changeUni

Request Body: changeUni: emailUniForm, studentId: int

200 Response: message: String("Student successfully changed university")

400 Response: error : String("Invalid parameters")

POST api/internship/create

Request Body: adInfo: Object, companyId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid process")

GET api/internship

Request Body: userId: int

200 Response: internships: Array of Object

400 Response: error : String("Empty List")

GET api/internship/{internshipId}

200 Response: internship: Object

400 Response: error : String("Invalid data")

POST api/internship/{internshipId}/apply

Request Body: studentId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid data")

PUT api/student/{studentId}/appliedPosition

Request Body: internship: Object

200 Response: message: String("Success")

GET api/internship/created

Request Body: companyId: int

200 Response: internships: Array of Object

400 Response: error : String("Invalid data")

GET api/internship/allCandidates

Request Body: companyId: int, adId: int

200 Response: allCandidates: Array of Object

400 Response: error : String("Invalid data")

PUT api/internship/allCandidates

Request Body: internship: Object, student: Object

200 Response: message: String("Success")

GET api/internship/studentProfile

Request Body: student: Object, internship: Object

200 Response: studentProfile: Object

400 Response: error : String("Invalid data")

DELETE api/internship/removeFromAllCandidates

Request Body: internship: Object, student: Object

200 Response: message: String("Success")

DELETE api/internship/removeFromMatchedInternships

Request Body: internship: Object, student: Object

200 Response: message: String("Success")

DELETE api/internship/removeFromApprovedProfiles

Request Body: internship: Object, student: Object

200 Response: message: String("Success")

PUT api/company/approvedProfile

Request Body: internship: Object, student: Object

200 Response: message: String("Success")

POST api/user/acceptRecommendation

Request Body: userId: int, internshipId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid internshipId")

POST api/user/recommendationFeedback

Request Body: feedback: Object, userId: int, internshipId: int

200 Response: message: String("Success")

400 Response: error : String("Unauthorized")

PUT api/user/recommendations

Request Body: userId: int, internshipId: int

200 Response: message: String("Success")

POST api/internship/{internshipId}/interviews

Request Body: interview: Object, studentId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid parameters")

POST api/internship/{internshipId}/timeSlot

Request Body: timeSlot: time, studentId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid parameters")

POST api/internship/{internshipId}/selectsStudent

Request Body: studentId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid parameters")

GET api/user/{userId}/acceptedProposals

200 Response: acceptedProposals: Array of Objects

400 Response: error : String("Unauthorized")

GET api/user/{userId}/matchedInternships

200 Response: matchedInternships: Array of Objects

400 Response: error : String("Empty List")

POST api/student/{studentId}/response

Request Body: internship: Object, response: boolean

200 Response: message: String("Success")

500 Response: error : String("Server side error")

PUT api/student/{studentId}/acceptedProposals

Request Body: internship: Object

200 Response: message: String("Success")

POST api/internship/{internshipId}/feedback

Request Body: userId: int, response: boolean

200 Response: message: String("Success")

500 Response: error : String("Server side error")

GET api/internship/{internshipId}/comments

Request Body: userId: int

200 Response: comments: Array of Object

400 Response: error : String("Invalid internshipId")

POST api/internship/{internshipId}/comments

Request Body: userId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid internshipId")

POST api/internship/{internshipId}/observation

Request Body: observation: Object, userId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid internshipId")

GET api/internship/{internshipId}/observations

Request Body: userId: int

200 Response: observations: Array of Object

400 Response: error : String("Invalid internshipId")

POST api/internship/{internshipId}/complaint

Request Body: complaint: Object, internshipId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid internshipId")

GET api/internship/{internshipId}/complaints

Request Body: userId: int

200 Response: complaints: Array of Object

400 Response: error : String("Invalid internshipId")

PUT api/internships/{internshipId}/interrupt

Request Body: companyId: int, studentId: int

200 Response: message: String("Success")

400 Response: error : String("Invalid internshipId")

GET api/user/{userId}/template

200 Response: template: Object

400 Response: error : String("Invalid userId")

2.6 Selected Architectural Styles and Patterns

- **Three-Tier** The 3-tier architecture divides the system into three logical layers: presentation (Web Server), business logic (Application Server), and data storage (Database). It allows scalability, simplifies maintenance, and enhances control and security.
- **RESTful APIs** RESTful APIs (Representational State Transfer) follow a design paradigm based on specific principles for creating web services. They use standard HTTP methods (GET, POST, PUT, DELETE) and are stateless, allowing easy data exchange between systems. It provides seamless integration with various platforms and systems, it is simple to implement and reduces server load.
- **On-Cloud** The system is hosted on a cloud infrastructure, guaranteeing scalability and cost efficiency (all resources that are asked to the cloud provider are effectively used).

2.7 Other Design Decisions

2.7.1 Database

We selected a relational database for our system design because it is effective at storing structured data, enforcing data integrity, and providing fast query performance. In particular, we decided to use PostgreSQL, because it is open source and it is one of the most used relational databases.

2.7.2 Distributed MVC Pattern

This pattern divides the application into three core components: Model, View, and Controller. The Model manages the data and business logic, the View presents the data to users, and the Controller processes user input and system responses. This division allows simultaneous development of distinct parts of the system. The distributed aspect of this pattern particularly aids in handling complex, scalable applications by facilitating efficient data processing and user interface management across different systems or networks.

2.7.3 Interaction Manager

We decided to implement a separate manager to take care of each interaction of the user with the system. In this first version the interaction are limited to feedback given by the users. However, if the recommendation mechanism needs to be modified in order to consider also other interactions of the users (e.g. logging viewing time and mouse movements), this is going to be possible through adaptation of the Interaction Manager.

3 User Interface Design

This section presents the interface mockups, illustrating the operational flow of the system and the functionalities that will interact with the user. The system, which consists of three distinct primary interfaces designed based on user types (university, student, and company), is intended to be accessible on any desktop or mobile browser capable of facilitating user interaction.

Common Interfaces

The system interface includes entrance, sign up, and sign in pages, which are commonly accessible regardless of user type. The mechanism within the system is structured according to user roles, such as universities, companies, or students, directing each type to the pages relevant to their specific functions.

Figure 28: Entrance

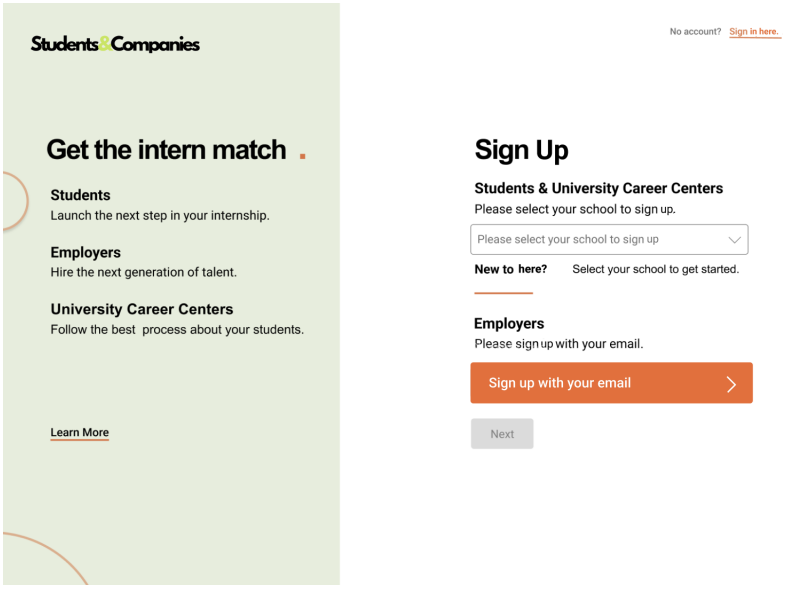


Figure 29: Home Sign Up Page

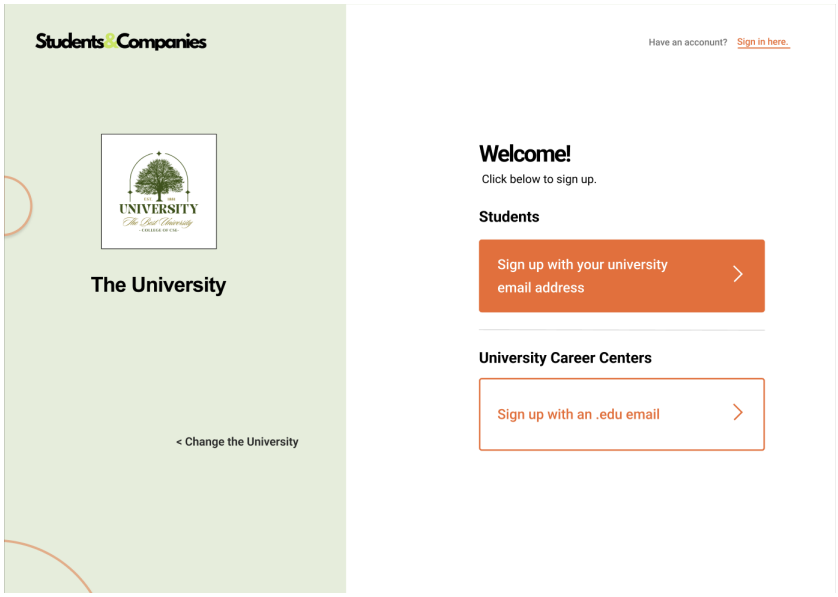


Figure 30: Student & University Sign Up

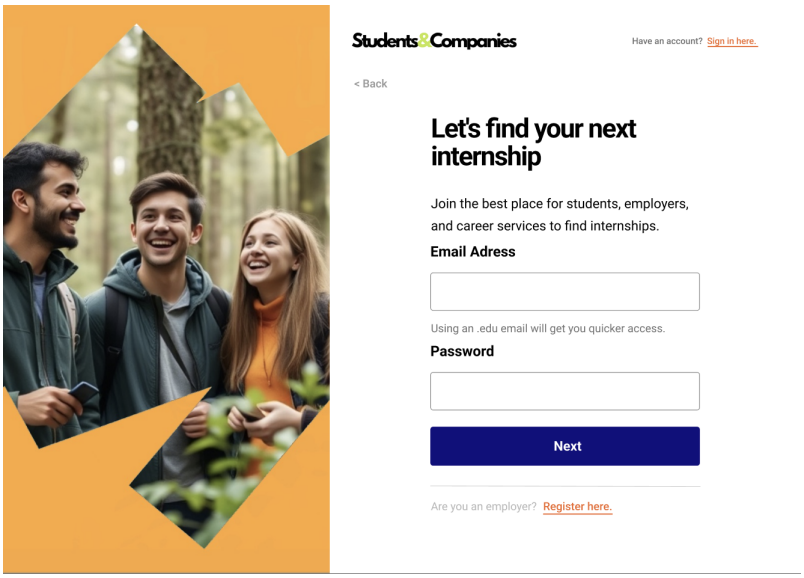


Figure 31: Student Sign Up Page

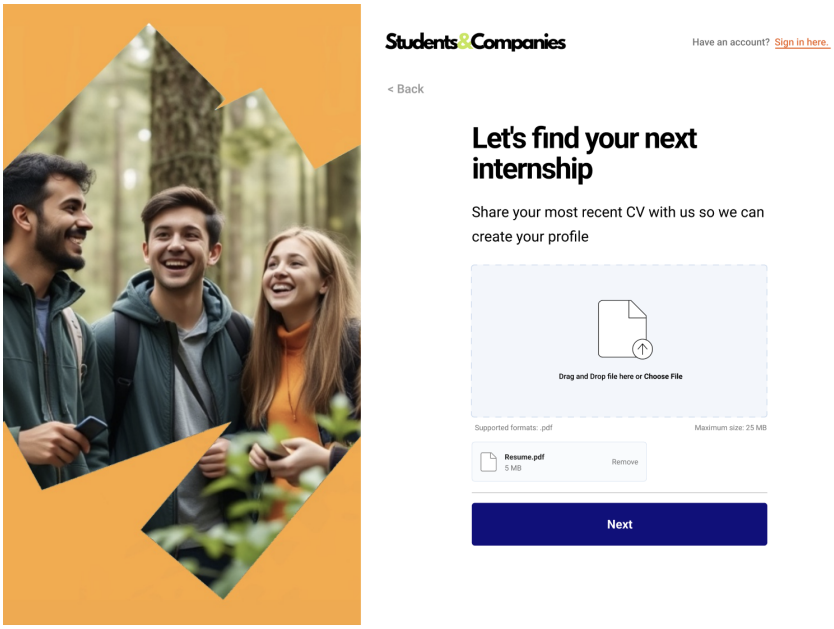


Figure 32: Student Upload CV

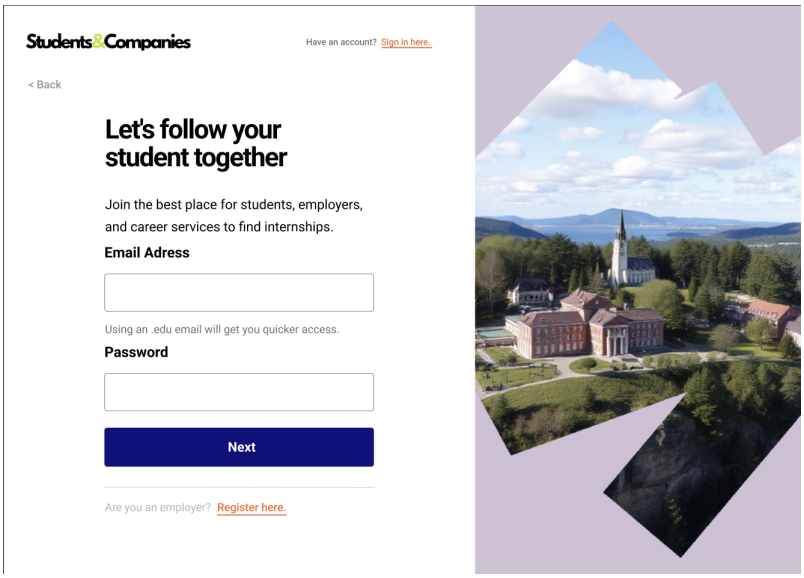


Figure 33: University Sign Up Page

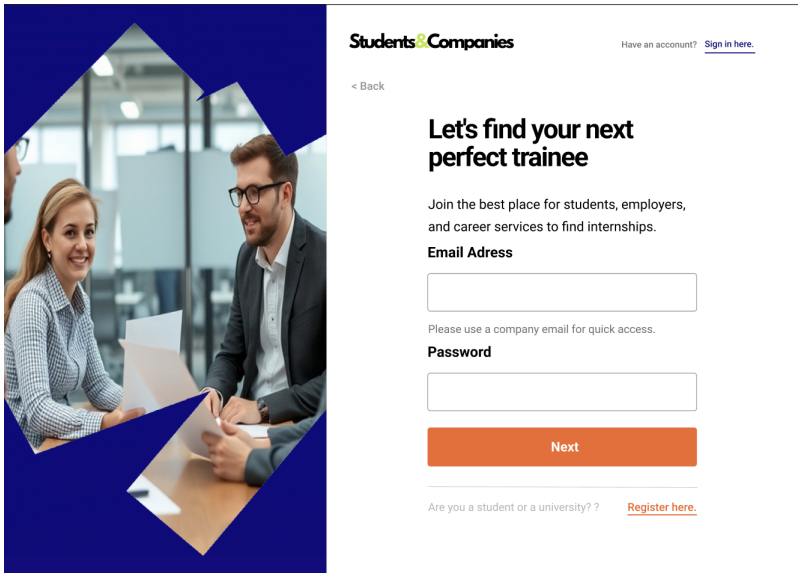


Figure 34: Employer Sign Up Page

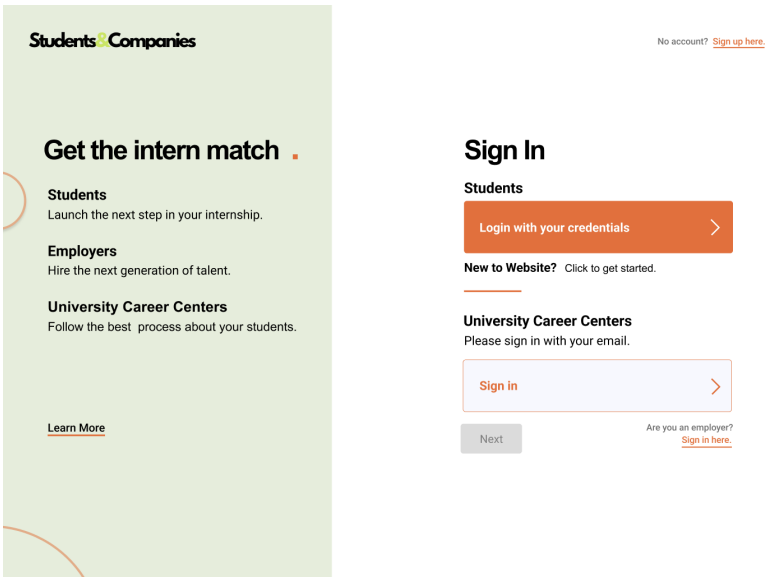


Figure 35: Home Sign In Page

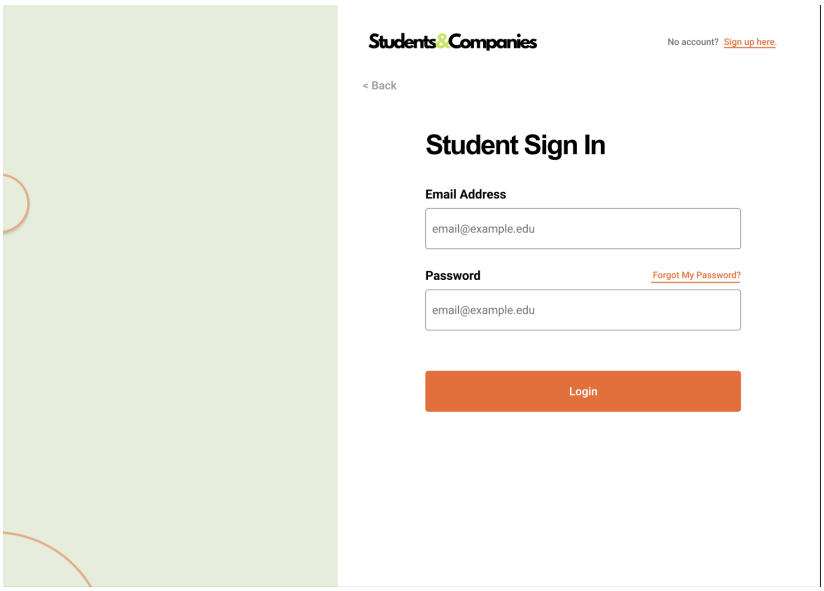


Figure 36: Student Sign In

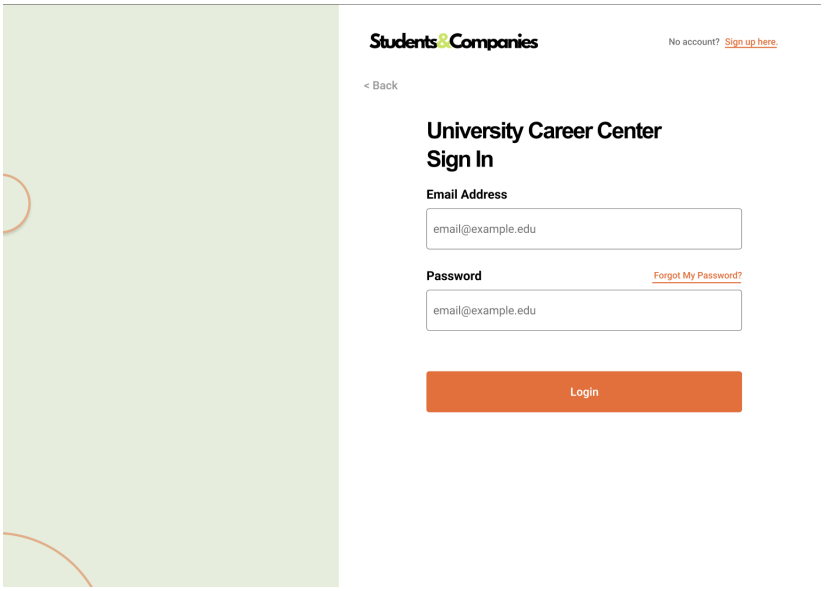


Figure 37: University Sign In

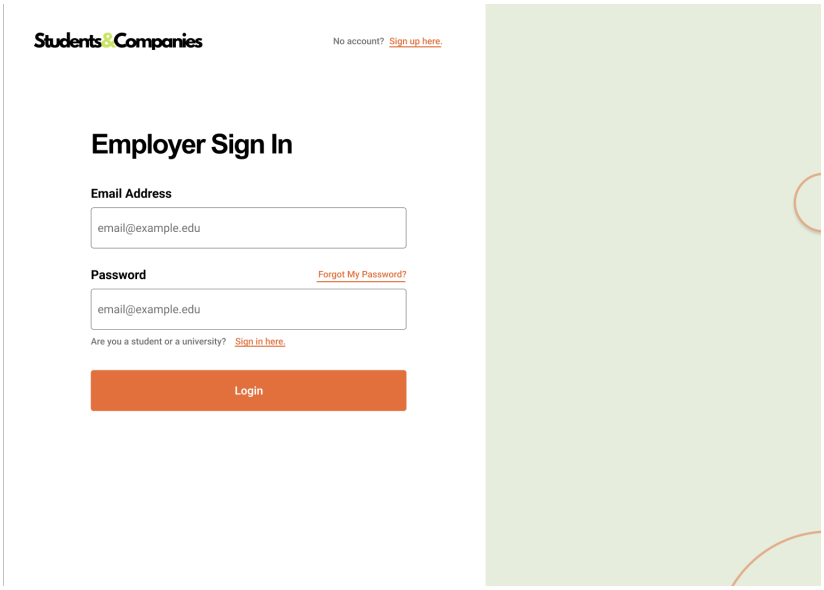


Figure 38: Employer Sign In

Student Pages

This section represents the pages that will be displayed to the student users, who have already registered in the system and provided the necessary credentials. The system offers a variety of functionalities, such as the ability for students to apply for internship postings, update their profiles, and receive recommendations and suggestions generated by the system. These functionalities and more are detailed in the design presented in this section.

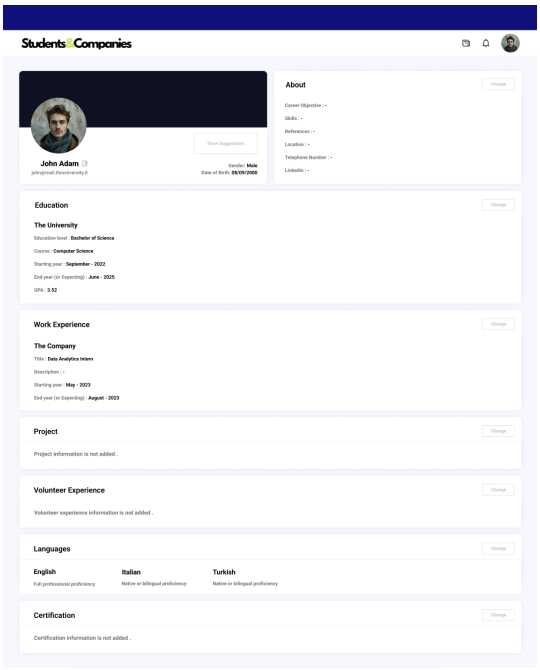


Figure 39: Student Profile Details

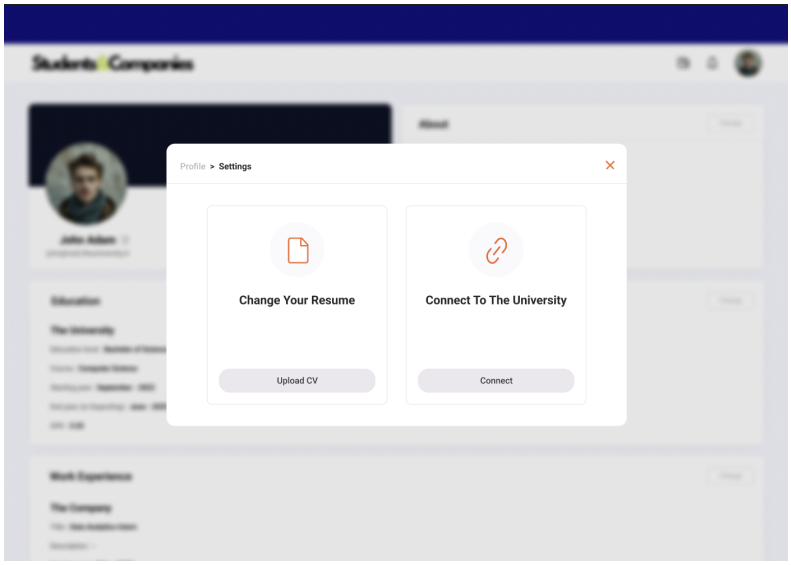


Figure 40: Student Settings

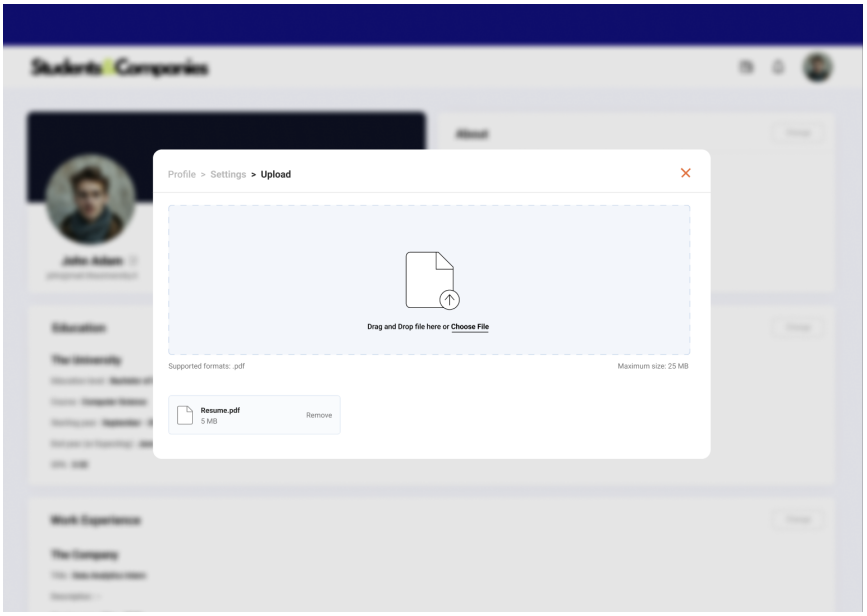


Figure 41: Student Update CV

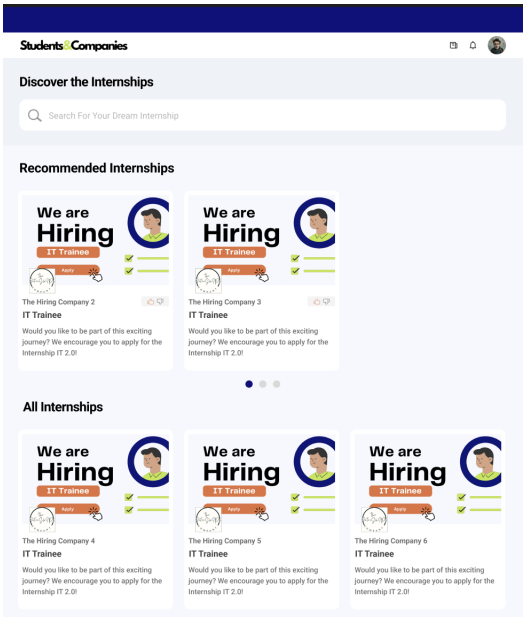


Figure 42: Student Home Page 1

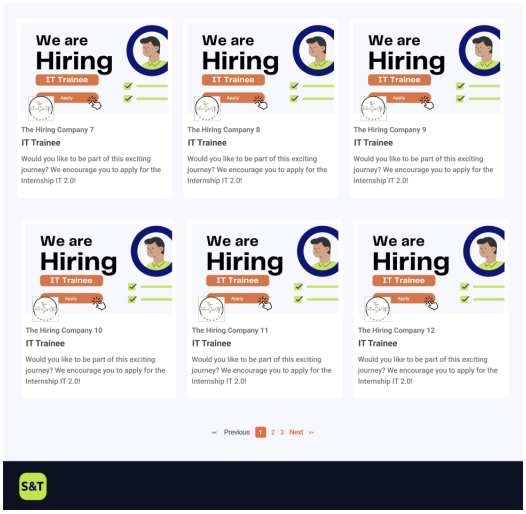


Figure 43: Student Home Page 2

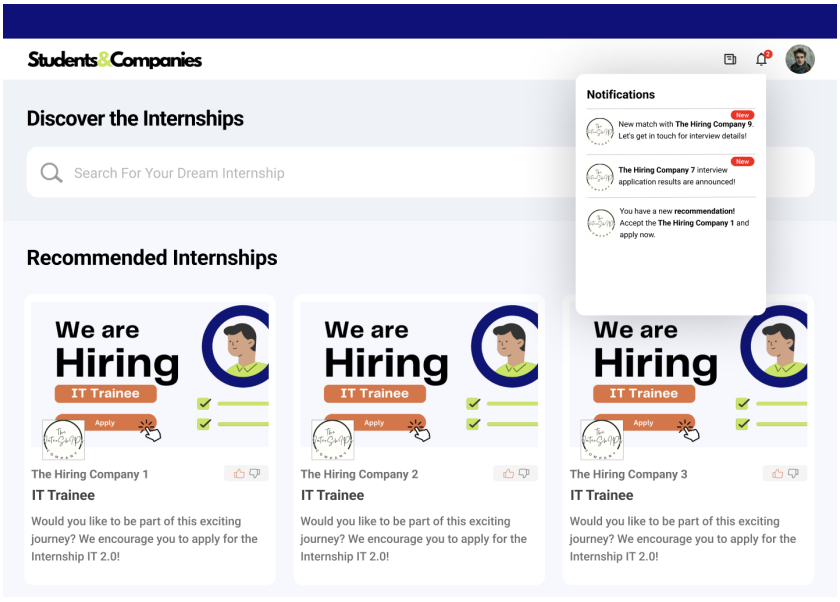


Figure 44: Student Notifications

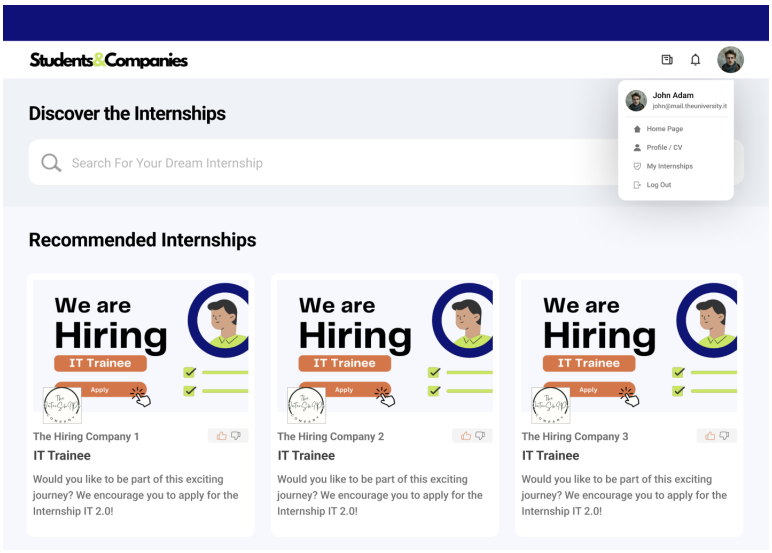


Figure 45: Student Profile Menu

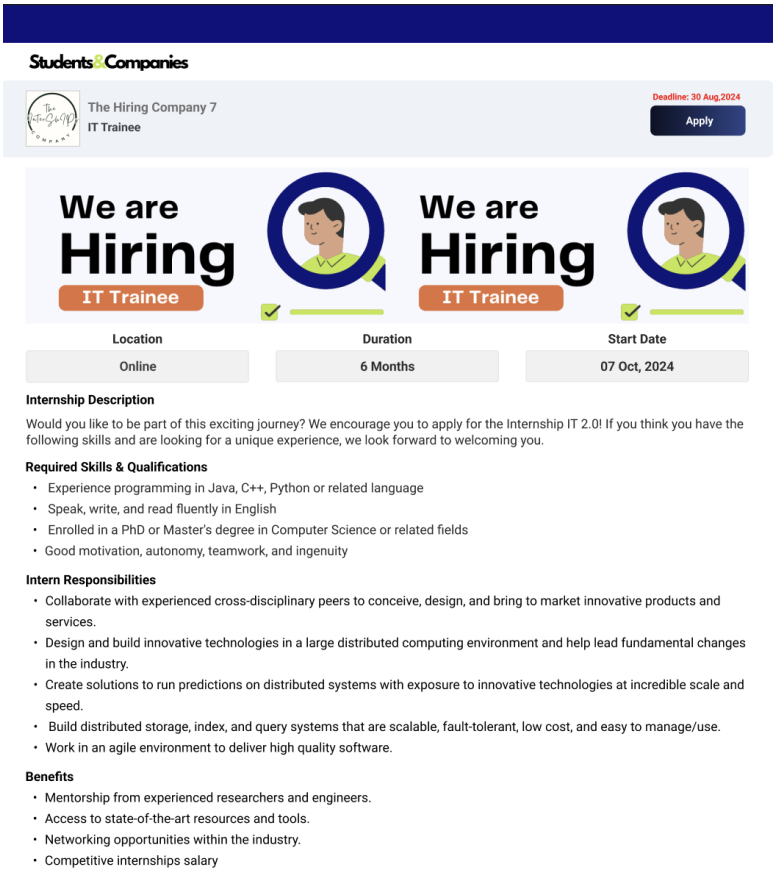


Figure 46: Advertisement Details Page

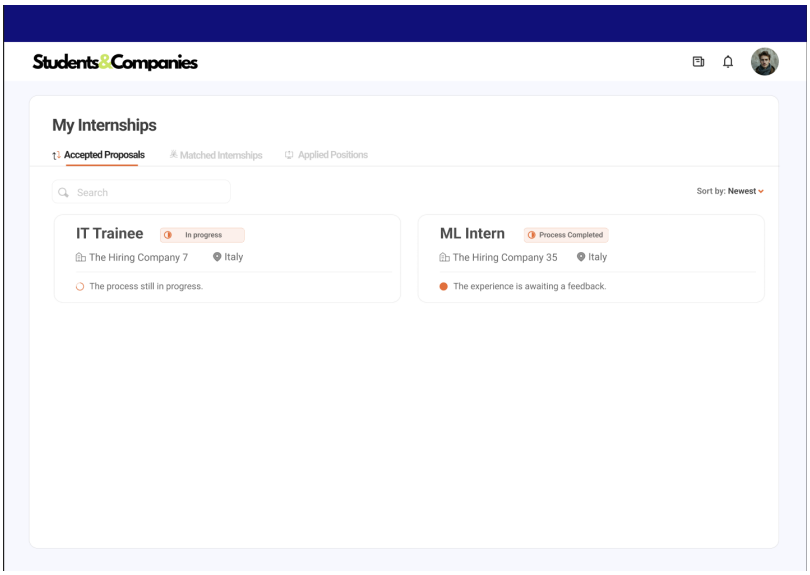


Figure 47: Student Accepted Internships

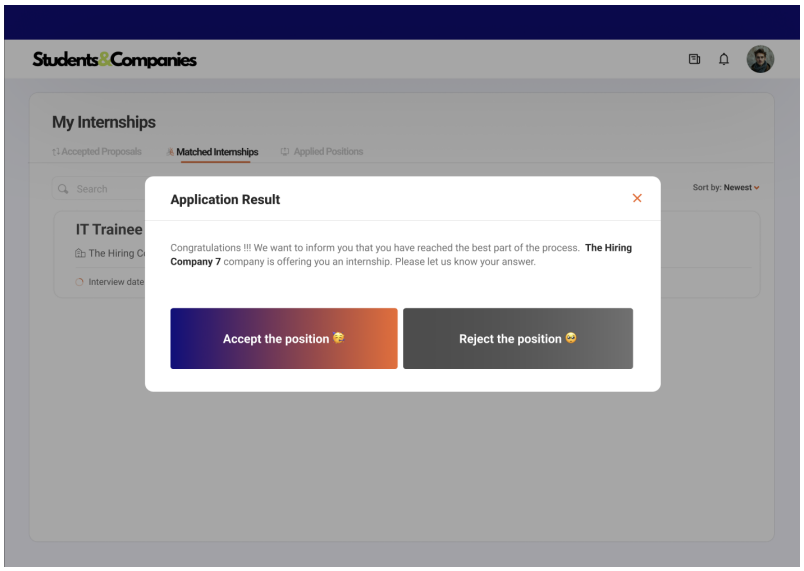


Figure 48: Accepting the position

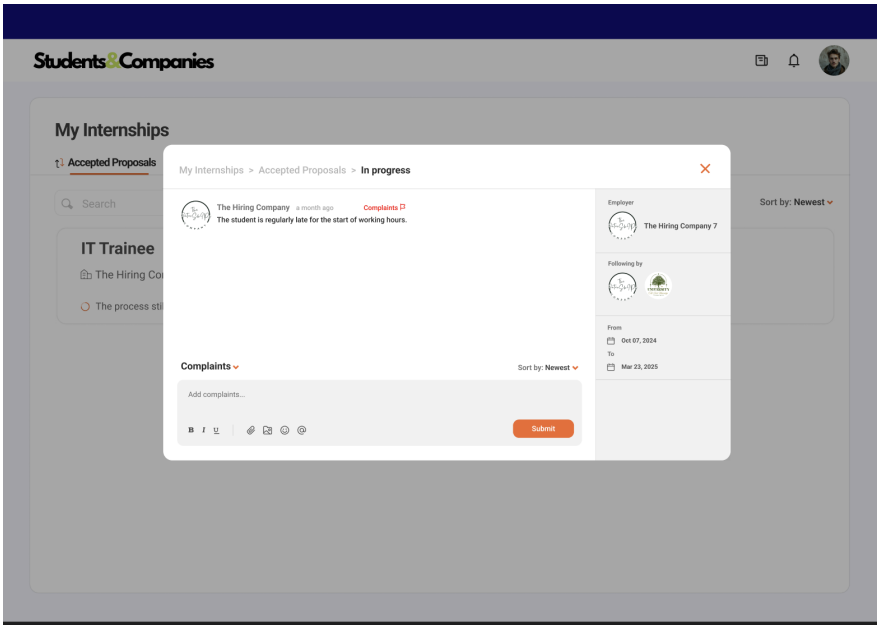


Figure 49: Student Ongoing Internship Complaints

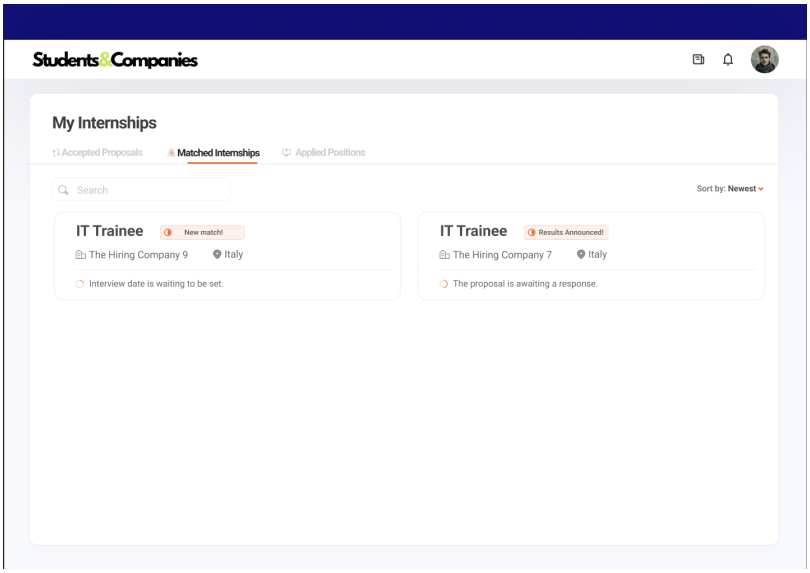


Figure 50: Student Matched Internships

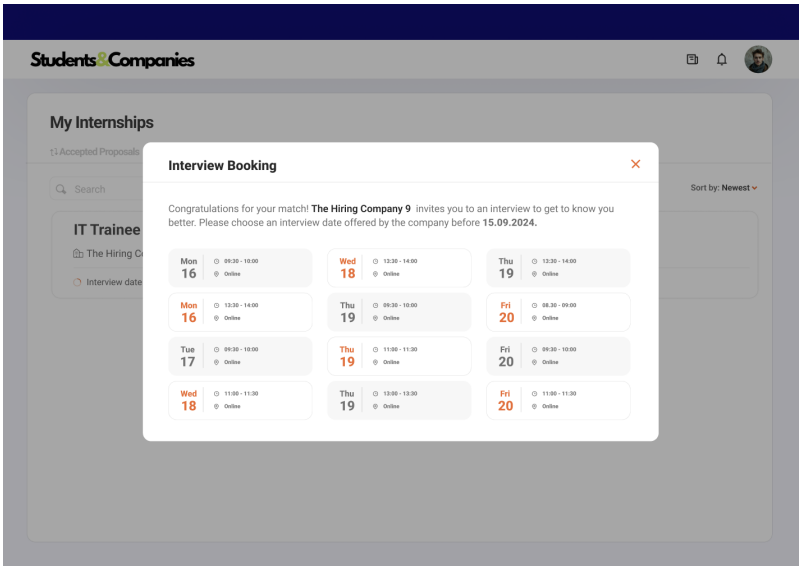


Figure 51: Student Interview Schedule

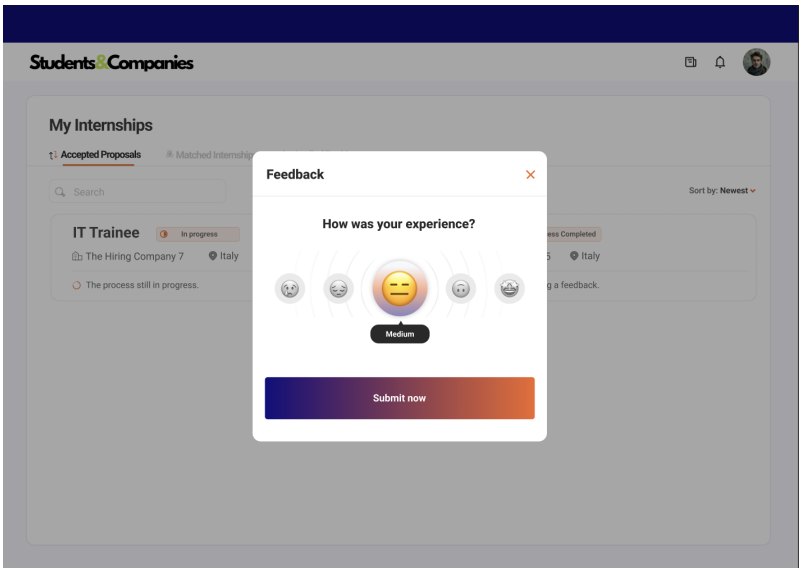


Figure 52: Student Feedback

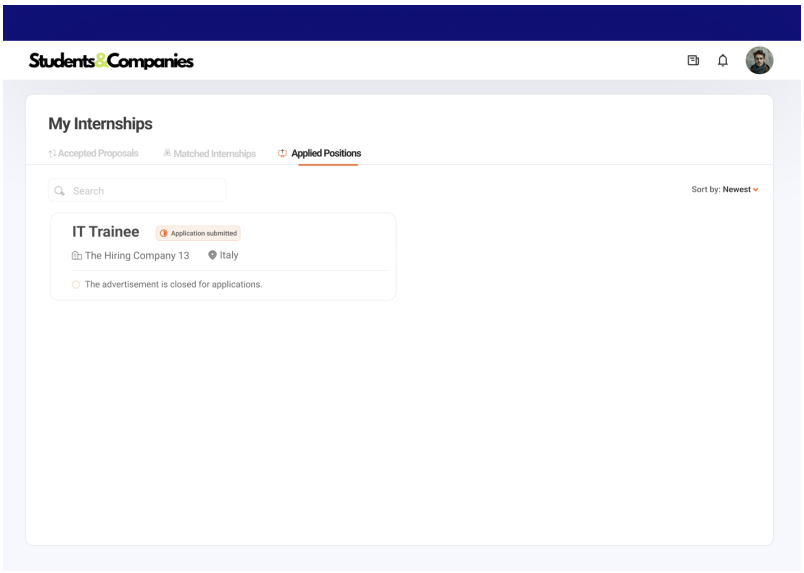


Figure 53: Student Applied Internships

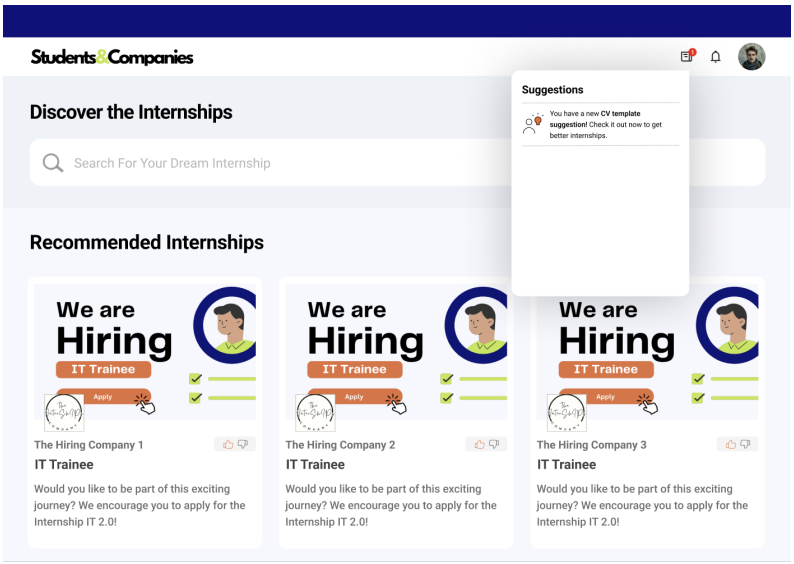


Figure 54: Student Suggestions Notification

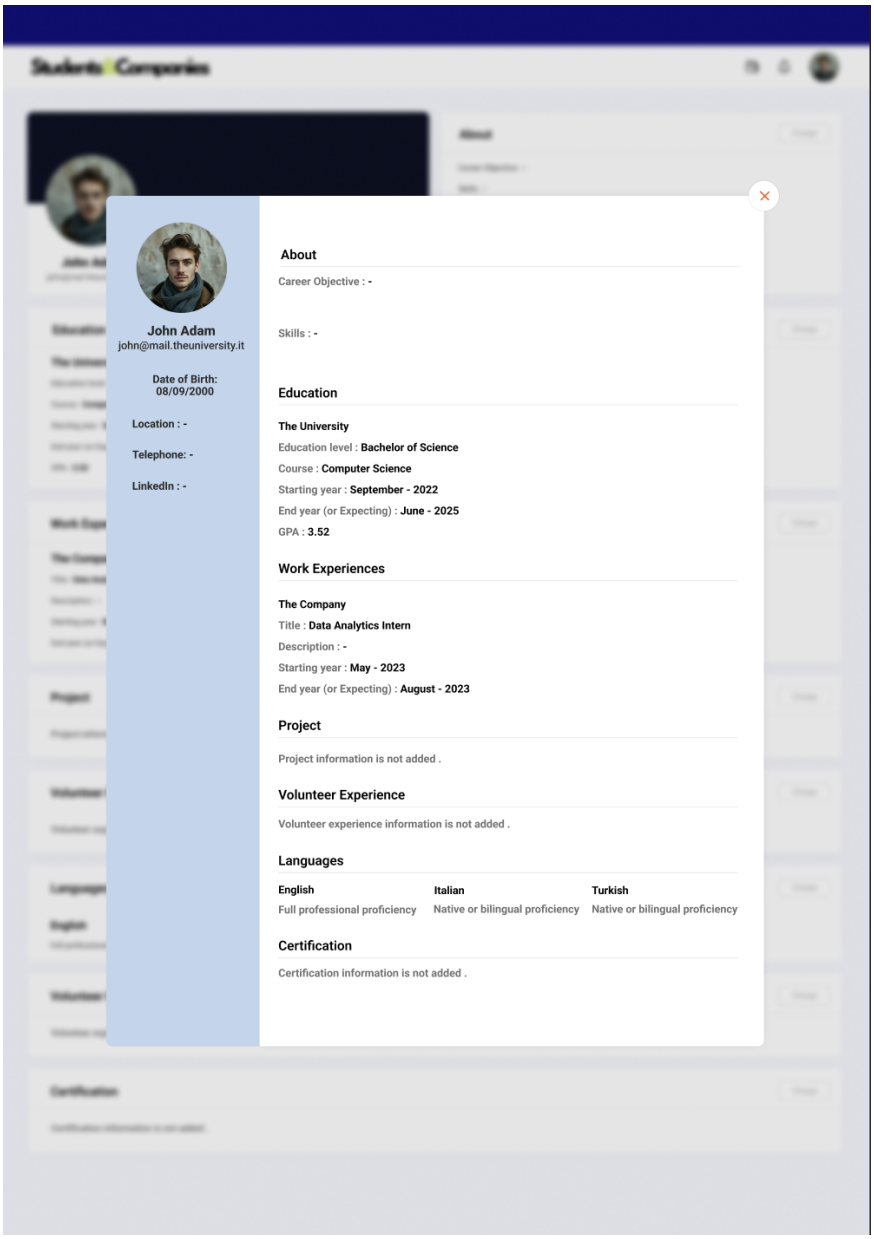


Figure 55: Student CV Suggestion Template

Company Pages

This section contains the pages design that will be displayed to company users after they provide the necessary credentials. Also, in this section presents the system designs that allow company users to create internship postings according to their needs, accept students recommended by the system or those who apply normally, and perform various functionalities such as interview scheduling, offering and more internship positions to students.

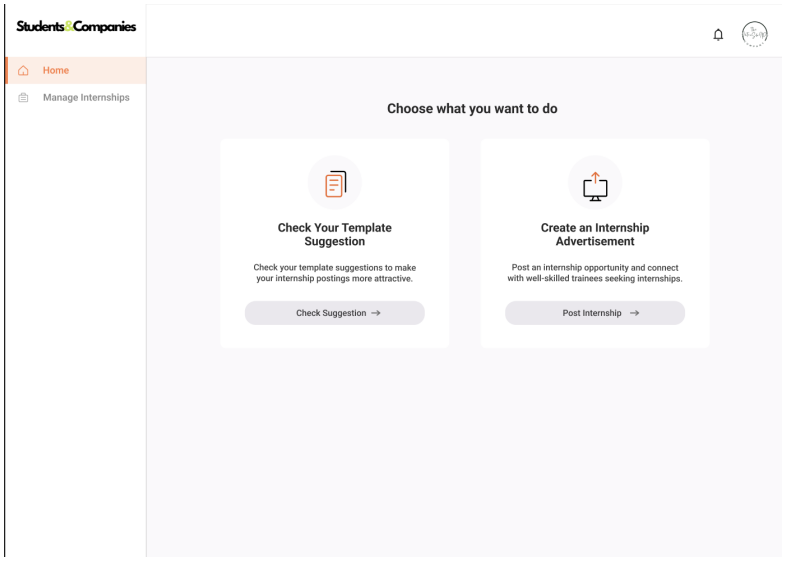


Figure 56: Company Home Page

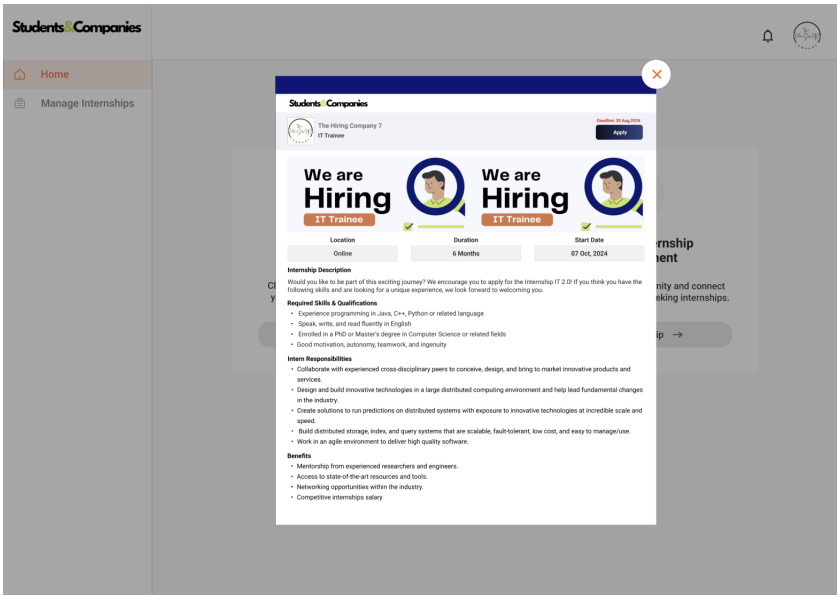


Figure 57: Company Advertisement Suggestion Template

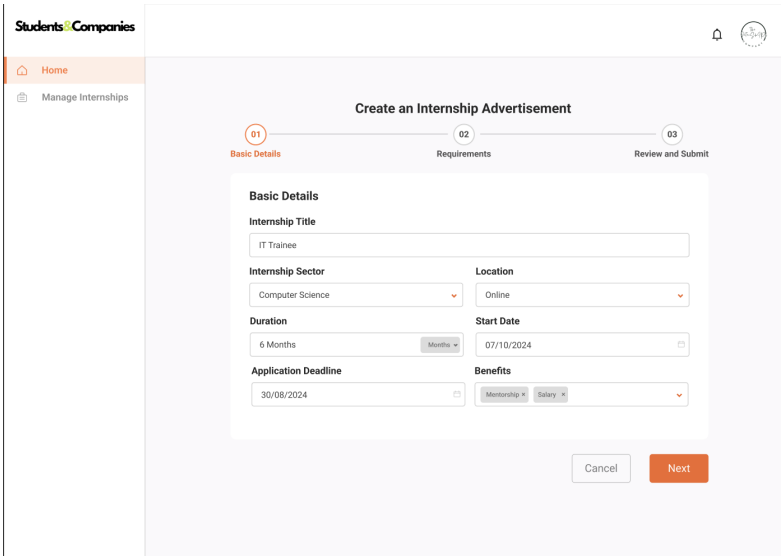


Figure 58: Create Internship Page 1

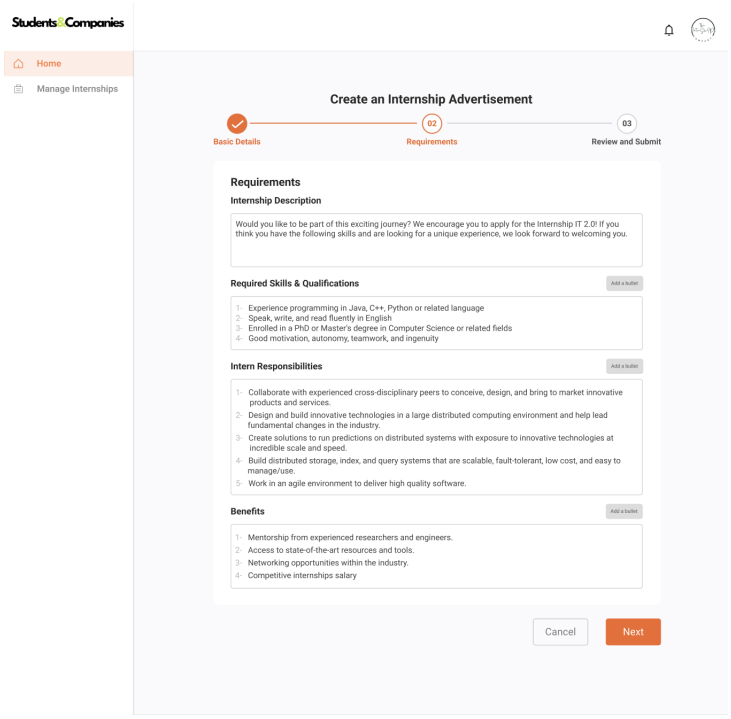


Figure 59: Create Internship Page 2

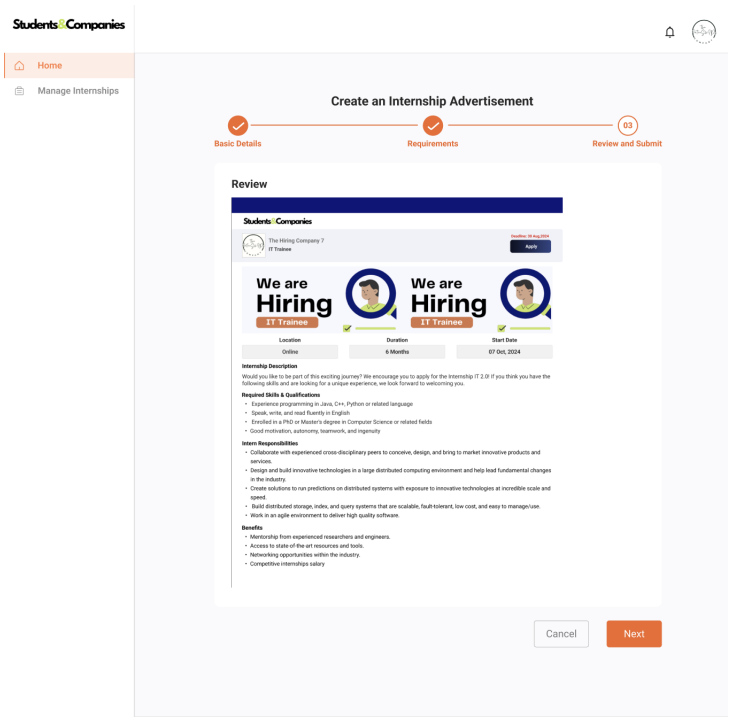


Figure 60: Create Internship Page 3

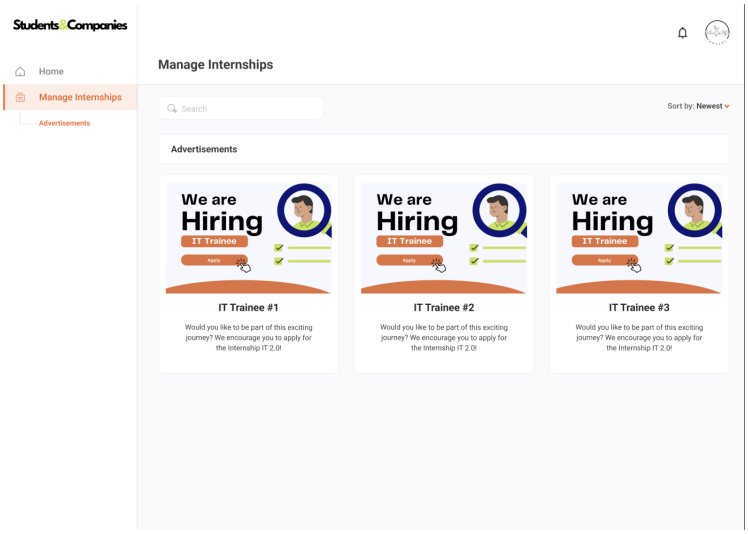


Figure 61: Manage Internships Page

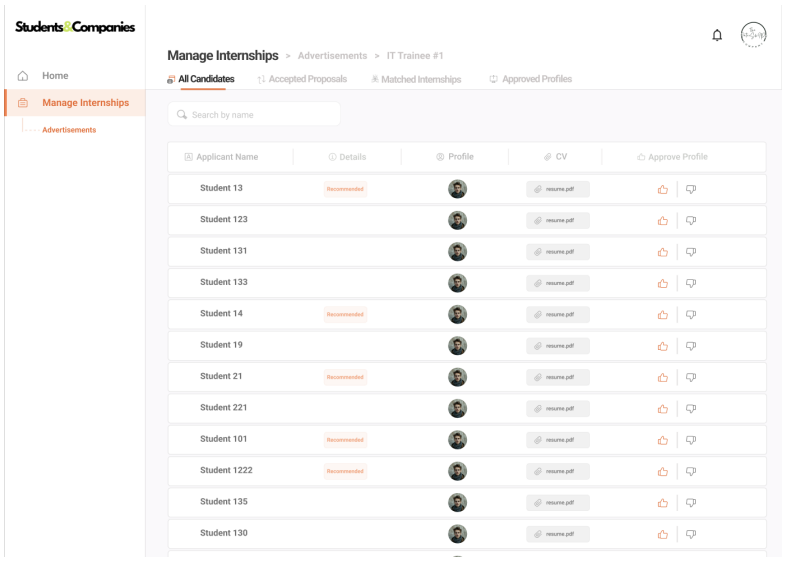


Figure 62: All Candidates Page

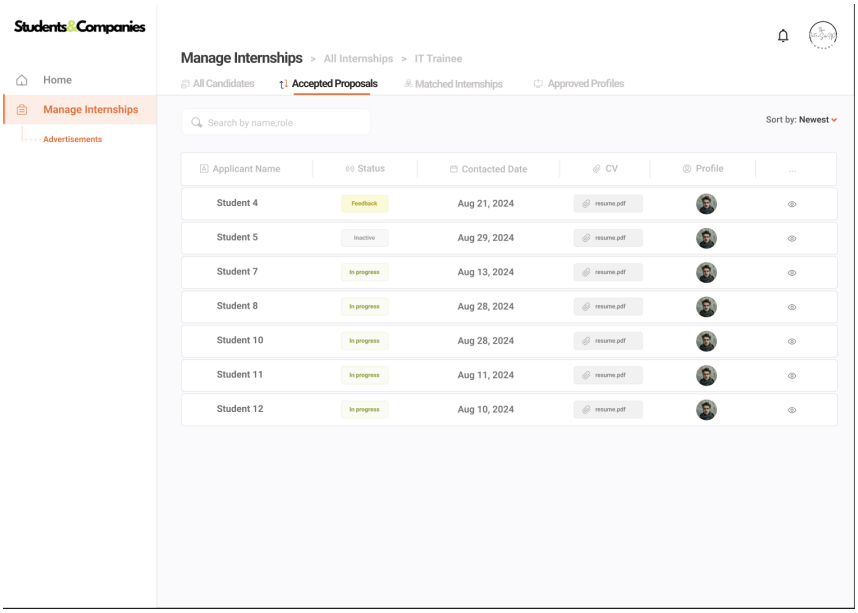


Figure 63: Company Accepted Proposals Page

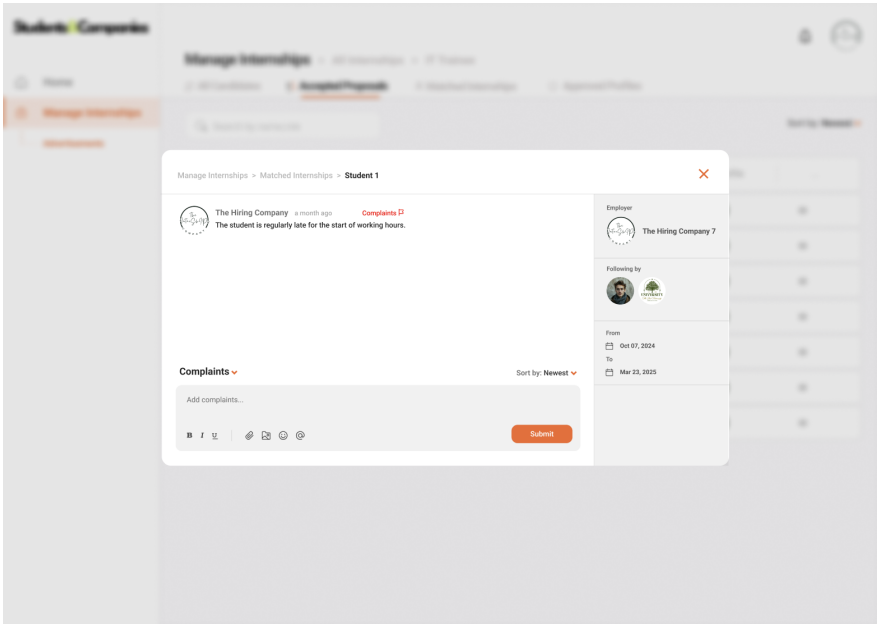


Figure 64: Company Ongoing Internship Complaints Page

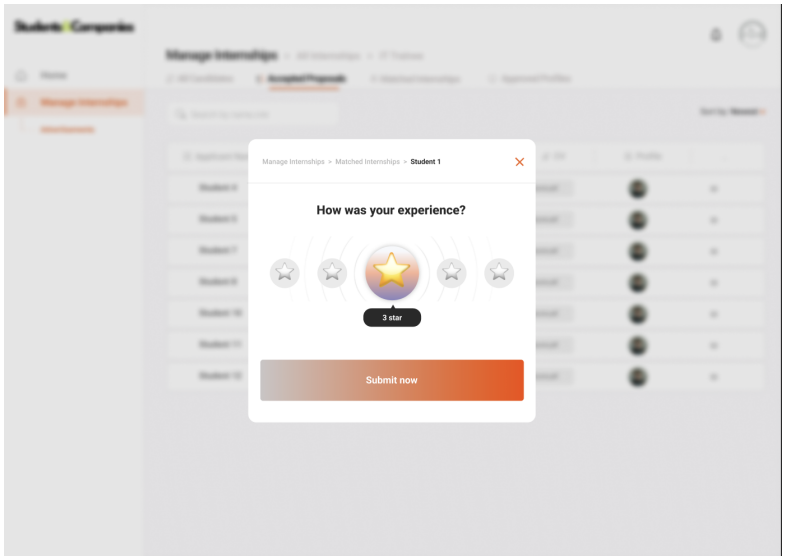


Figure 65: Feedback

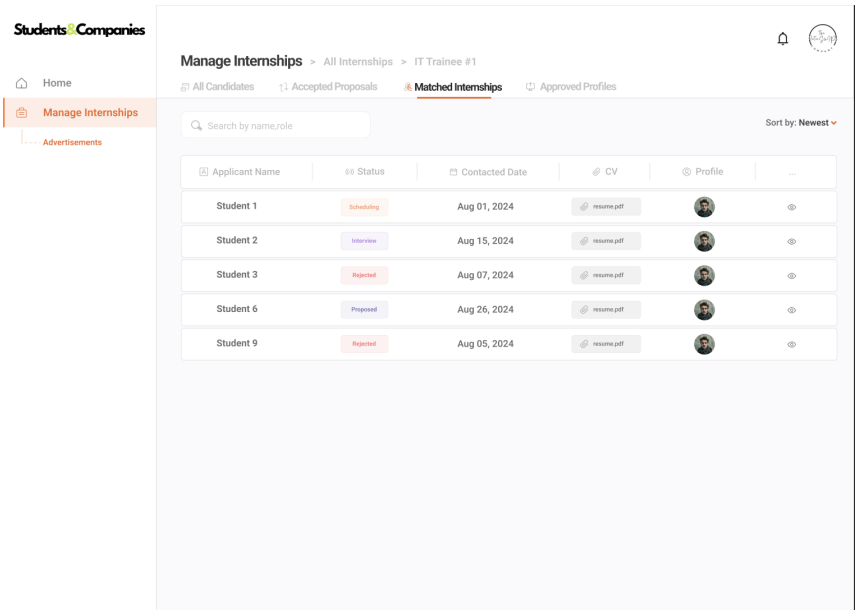


Figure 66: Company Matched Internships Page

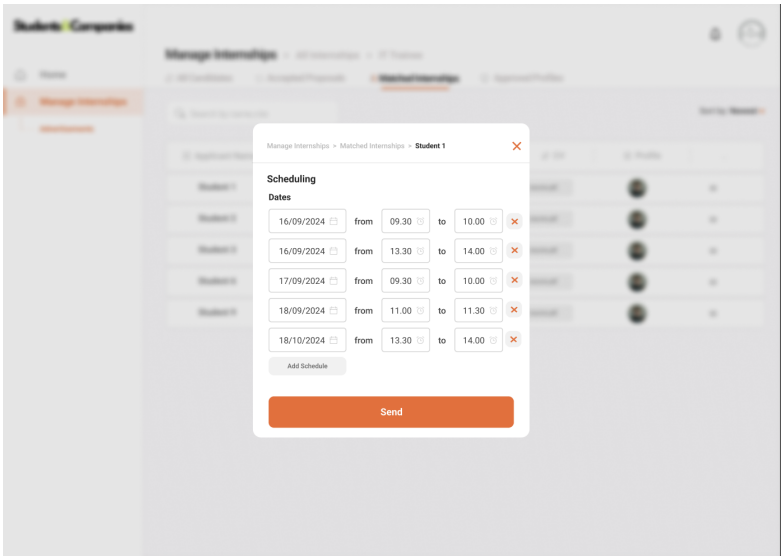


Figure 67: Company Interview Scheduling

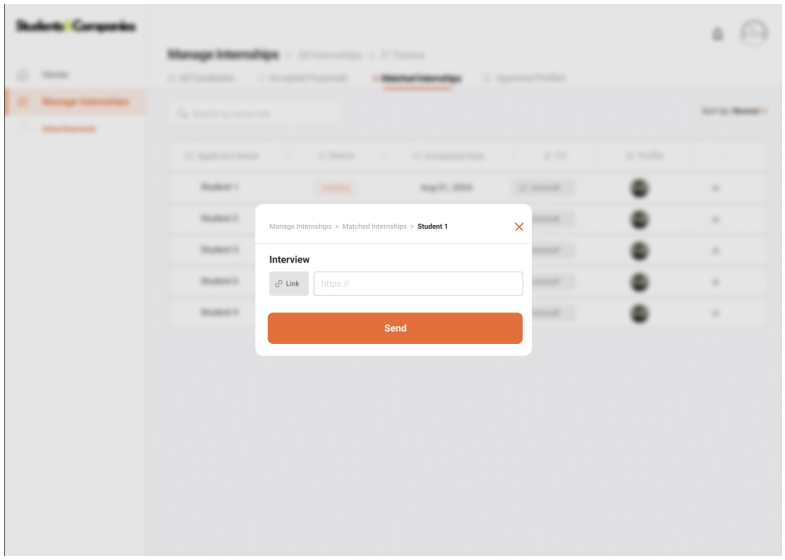


Figure 68: Interview Link

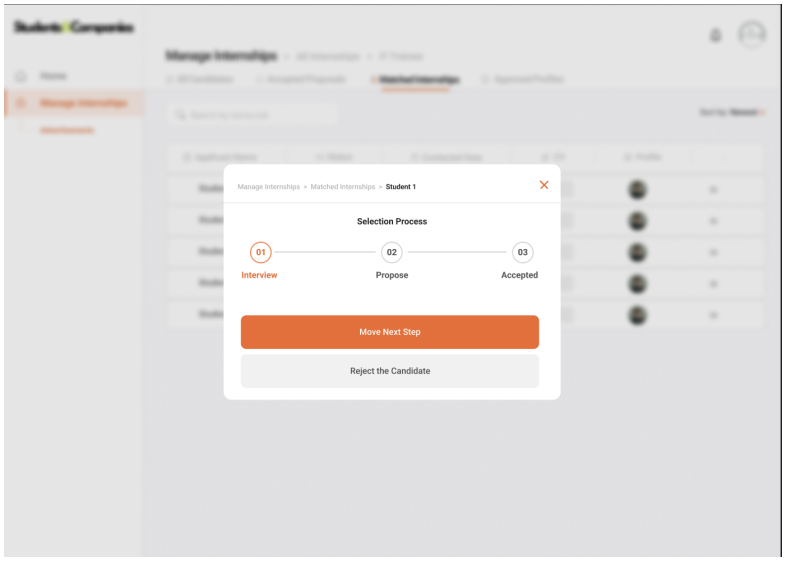


Figure 69: Student Selection Process Pop-Up

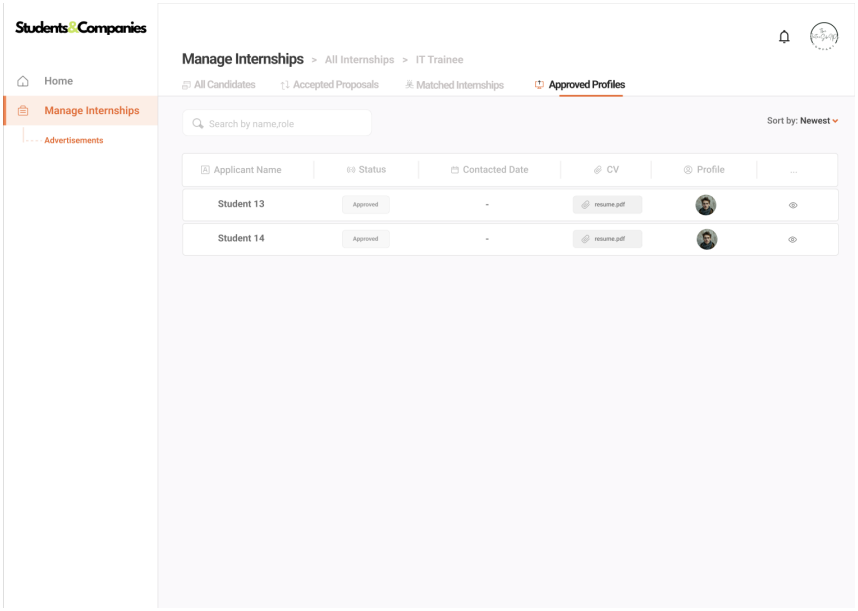


Figure 70: Company Approved Profiles Page

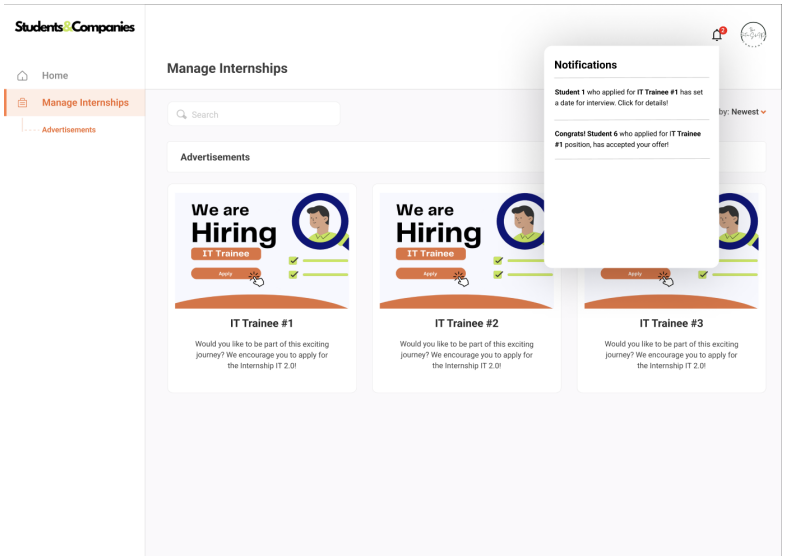


Figure 71: Company Notifications

University Pages

This section includes the page designs that university representatives can access after entering the necessary credentials into the system. The system provides interface that enable university representatives to monitor their students' internship processes, view any complaints that may arise during the internship, and, when necessary, interrupt the internship process.

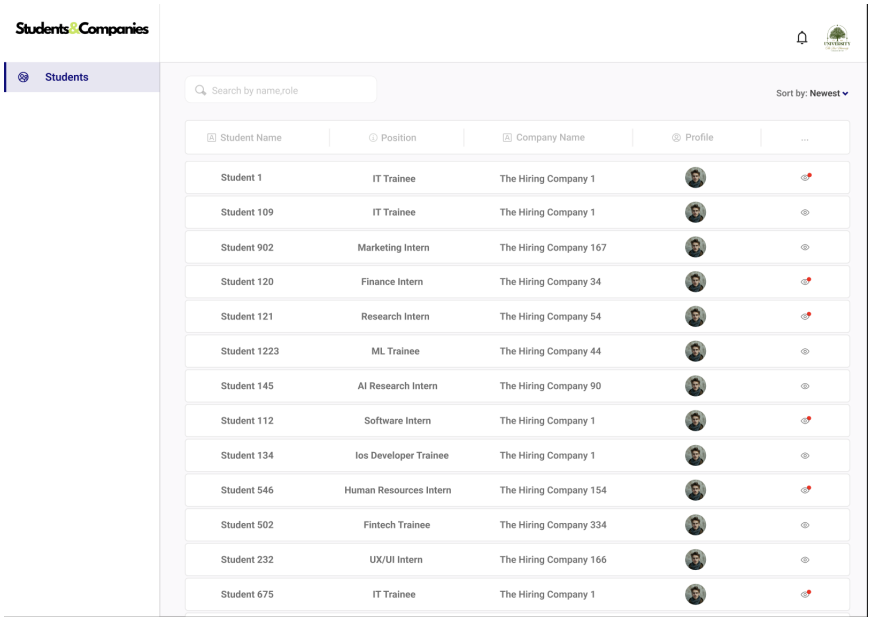


Figure 72: University Home Page

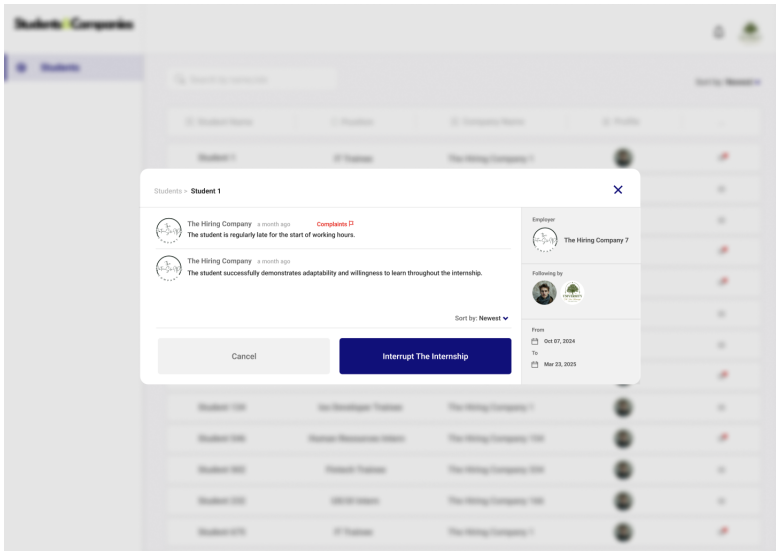


Figure 73: University Interrupt Page

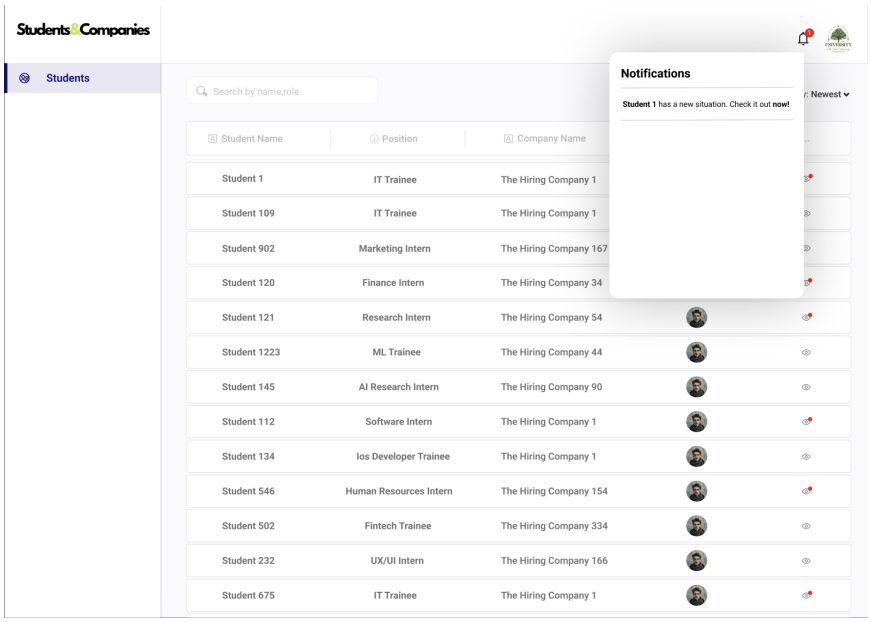


Figure 74: University Notifications

4 Requirements Traceability

R1: The system must allow a student who wants to register to sign up.	
C0	DB Manager
C3	Authentication Manager
C6	Recommendation Manager
C8	Notification Manager
C9	Model
C10	Email Service Provider
C11	Notification Provider
C13	University Login Service

R2: The system must allow a company who wants to register to sign up.	
C0	DB Manager
C3	Authentication Manager
C6	Recommendation Manager
C9	Model
C10	Email Service Provider

R3: The system must allow a university who wants to register to sign up.	
C0	DB Manager
C3	Authentication Manager
C8	Notification Manager
C9	Model
C10	Email Service Provider
C12	National Education Dictionary API

R4: The system must allow registered users to sign in using their credentials.	
C0	DB Manager
C1	Dashboard Manager
C2	Account Manager
C3	Authentication Manager
C9	Model

R6: The system must be able to send notifications to all users.	
C1	Dashboard Manager
C4	Internship Manager
C5	Comment Manager
C6	Recommendation Manager
C7	Interaction Manager
C8	Notification Manager
C10	Email Service Provider
C11	Notification Provider

R7: The system must allow registered students to upload their CVs on the platform.	
C0	DB Manager
C1	Dashboard Manager
C2	Account Manager
C6	Recommendation Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R8: The system must allow registered companies to post internship advertisements.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C6	Recommendation Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R9: The system must allow companies to review students' CVs and select candidates who meet their internship requirements.	
C0	DB Manager
C1	Dashboard Manager
C2	Account Manager
C9	Model

R10: The system must allow students to review internship advertisements and select them if they wish to apply.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C9	Model

R11: The system must allow students to manually search for internship opportunities and save them to their favorites.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C9	Model

R12: The system must notify students when there are updates regarding the internships they applied for or accepted.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C8	Notification Manager
C11	Notification Provider

R13: The system must notify a student and a company that accept each other.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R14: The system must allow companies to choose a suitable date for the interview, but only after a match is done.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R15: The system must schedule an interview at the date and time specified by the company and notify the selected students.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R16: The system must recommend a student and an internship to each other if the student's profile matches the needs of the company.	
C0	DB Manager
C1	Dashboard Manager
C2	Account Manager
C4	Internship Manager
C6	Recommendation Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R17: The system must allow companies to offer internship proposals to selected students after the interview process is completed.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R18: The system must allow companies and students to give feedback at the end of the internship in which they took part.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C7	Interaction Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R19: The system must allow the student and company who got recommended to each other to give feedback about the recommendation.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C6	Recommendation Manager
C7	Interaction Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R20: The system must provide a suggested template to users for improving their CV or project description.	
C0	DB Manager
C1	Dashboard Manager
C2	Account Manager
C9	Model

R21: The system must notify students when there are updates regarding the results of the internships they have applied for.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R22: The system must allow selected students to accept or decline internship proposal sent by companies.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R24: The system must allow companies to view and manage applications for the internships they have posted.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C6	Recommendation Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

R25: The system must allow students to view all details about the internships they have applied for, such as completion status, and deadlines.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C5	Comment Manager
C9	Model

R27: The system must allow universities to follow internship processes, handle complaints raised by students, and interrupt an internship if necessary.	
C0	DB Manager
C1	Dashboard Manager
C4	Internship Manager
C5	Comment Manager
C8	Notification Manager
C9	Model
C11	Notification Provider

5 Implementation, Integration and Testing Plan

5.1 Development Process and Approach

The system will be implemented, integrated, and tested using a bottom-up approach, taking into account the dependencies between components of the system. This strategy will be used for both the server and client sides, which will be developed and tested at the same time. Incremental integration testing will be applied to try to find and fix bugs as soon as possible during the development cycle. Since DB Manager and external services are assumed to be reliable, they don't need unit testing.

5.2 Implementation & Integration Plan

Since the application is mostly server-side, we will only describe the implementation of the server components. The client-side, which is actually a presentation layer, will be implemented and tested in parallel with the server-side.

5.2.1 Server Side

In the first step, the Model and the DB Manager will be implemented and unit tested using a Driver, which will substitute components that have not been implemented yet.

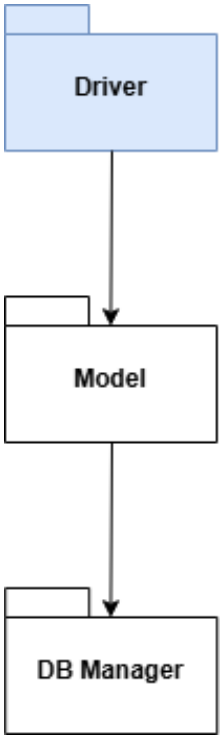


Figure 75: First step of Implementation

In the second step, the Notification Manager will be implemented and tested with a Driver, substituting all the components that uses it. The Notification Provider and the Email Service Provider will be implemented using Stubs.

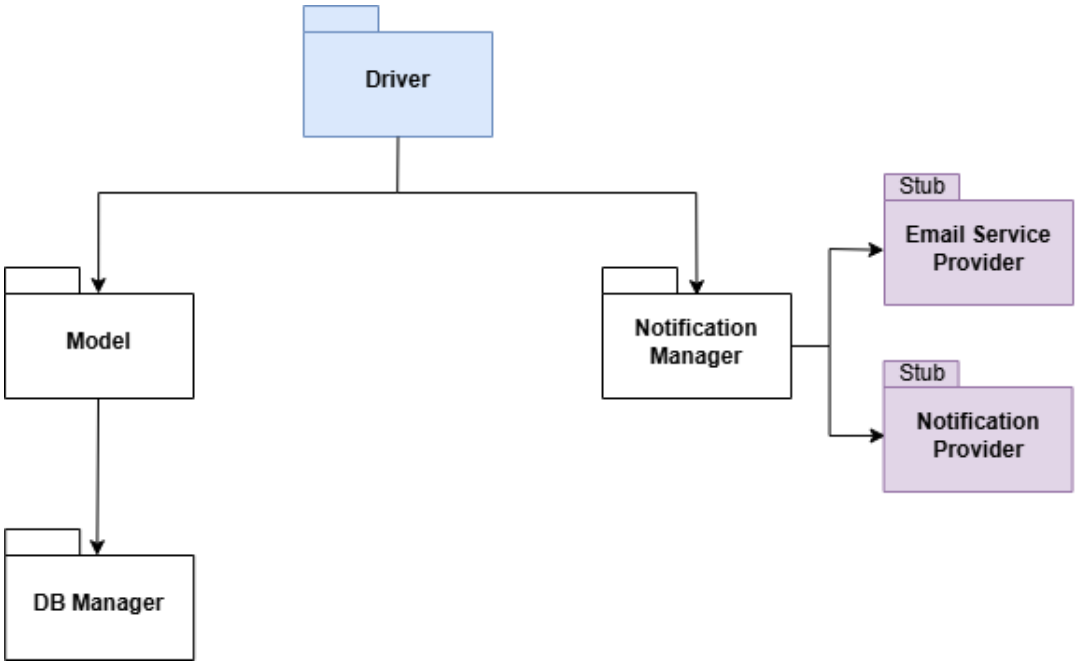


Figure 76: Second step of Implementation

In the third step, the Recommendation Manager and the Comment Manager will be implemented and and unit tested using a Driver.

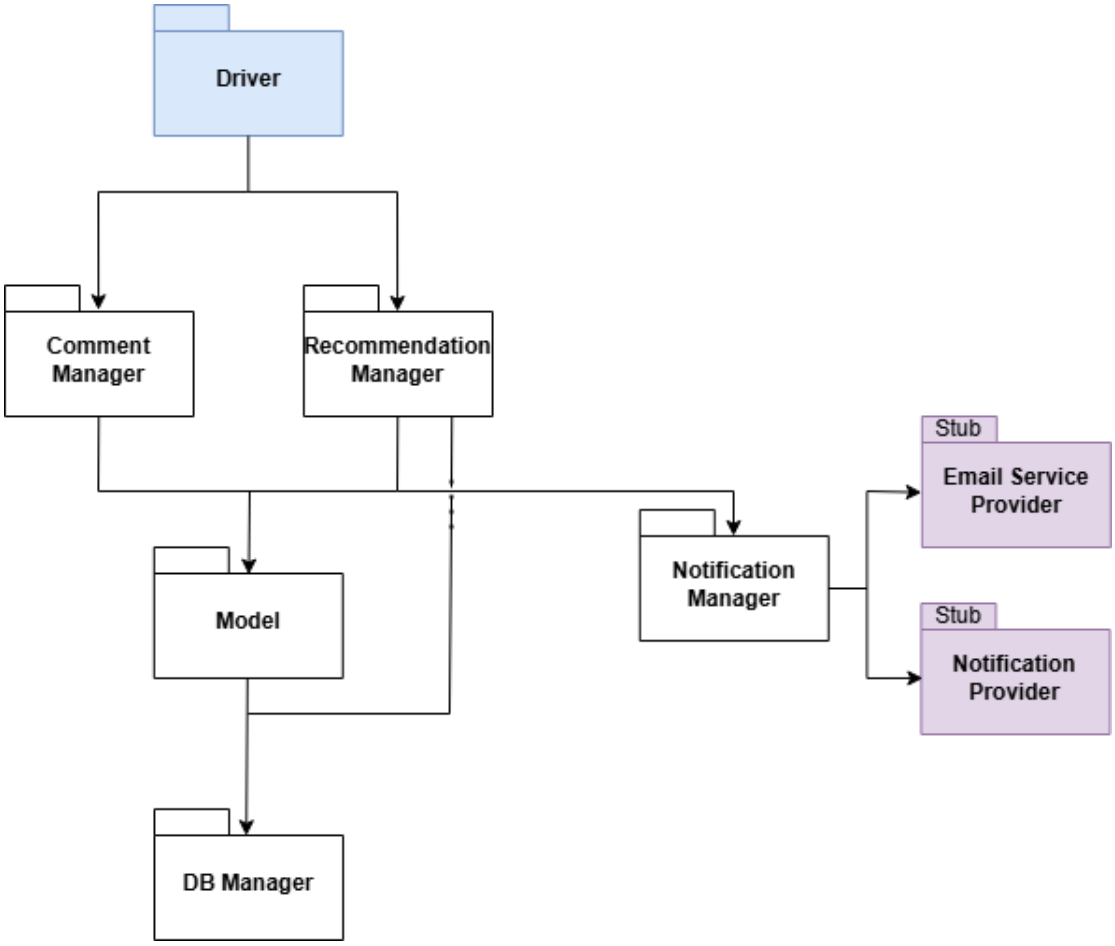


Figure 77: Third step of Implementation

In the fourth step, the Authentication Manager, Interaction Manager, Account Manager and Internship Manager will be implemented and tested in parallel, a Driver will substitute the Dashboard Manager. The implementation of the external services used by the Authentication Manager is done through stubs.

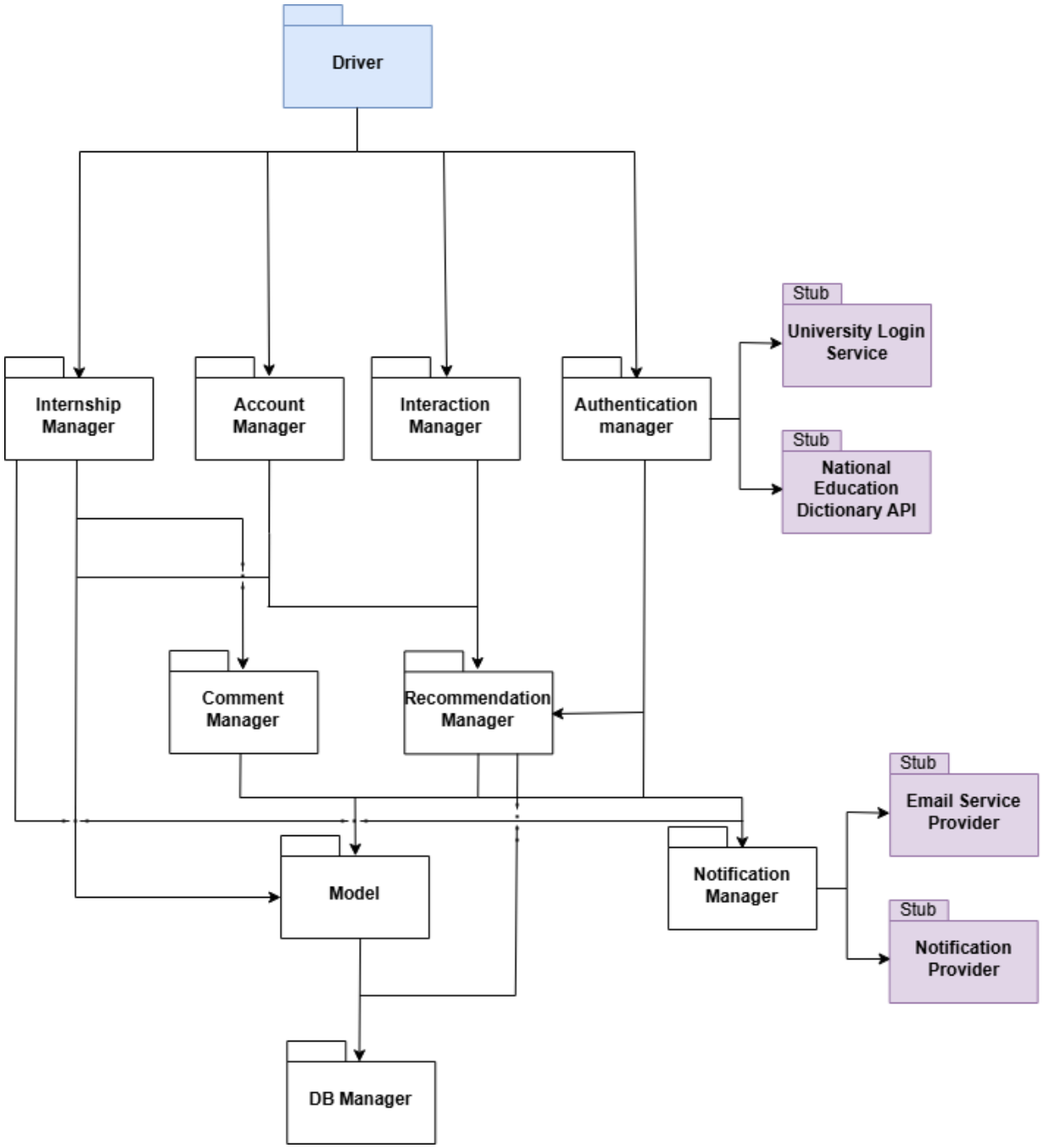


Figure 78: Fourth step of Implementation

In the fifth step, the implementation of the S&C server will be finished, with the completion of the Dashboard Manager. A driver will substitute the S&C Web UI for the unit test.

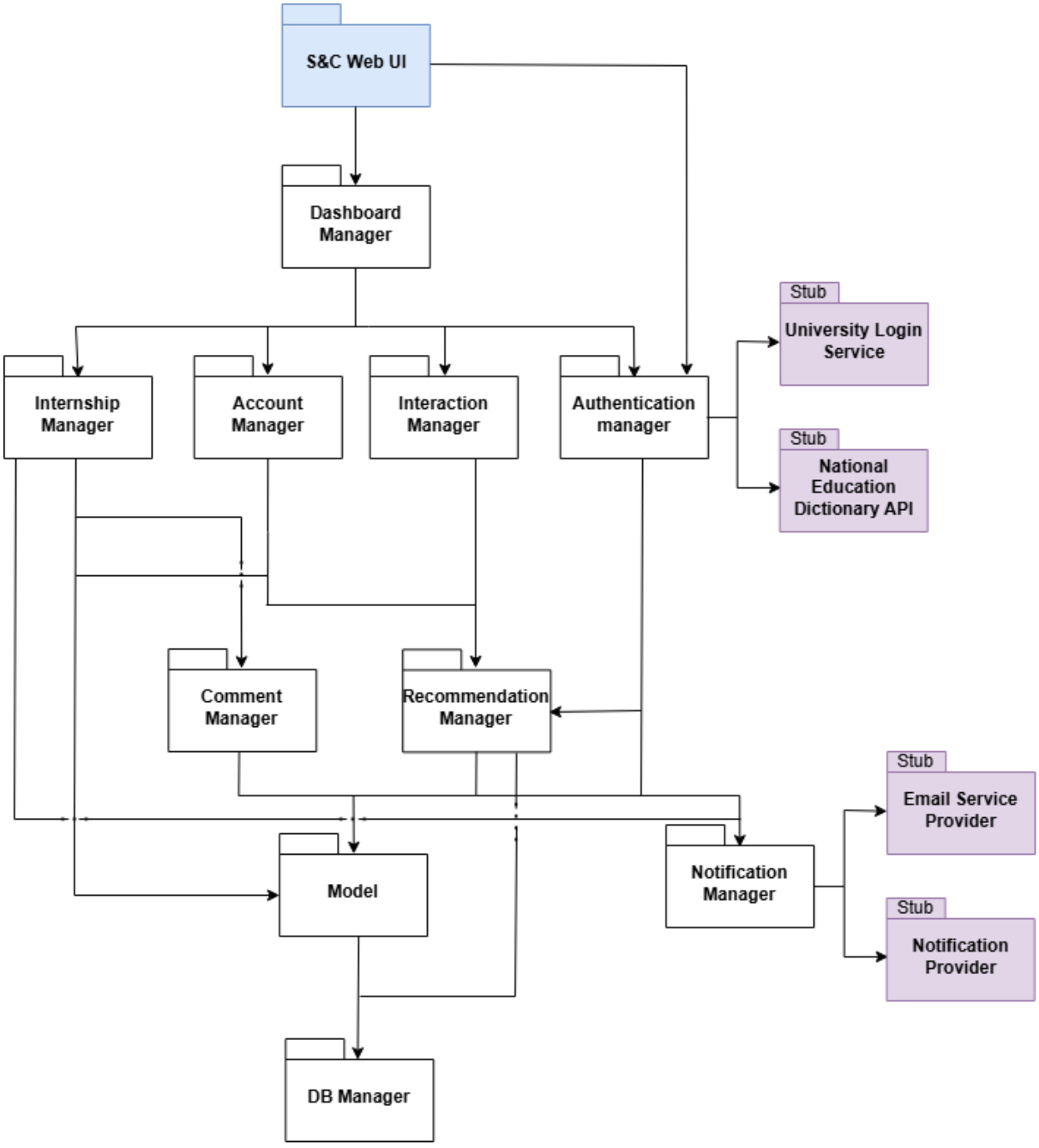


Figure 79: Fifth step of Implementation

Once all the components have been implemented and unit tested, full integration tests with Drivers and Stubs for external services are done. As the last step, integration tests with the real services take place.

6 Effort Spent

6.1 Effort Spent per Unit

This section shows the amount of time that each member has spent to produce the document.

UNIT	MEMBERS	HOURS
Setup	Ratti	4 hours
Chapter 1: Introduction, Chapter 7: References	Salvatore	1 hour
Overview: High-level Components and Interaction	Yalcin	3 hours
Component View	Ratti	5 hours
Deployment View	Salvatore	3 hours
Component Interfaces	Ratti, Yalcin	7 hours
Selected Architectural Styles and Patterns	Ratti, Salvatore	1 hour
Other Design Decisions	Ratti, Salvatore, Yalcin	1 hour
Chapter 3: User Interface Design	Yalcin	2 hours
Runtime View Diagrams	Ratti, Salvatore, Yalcin	27 hours
Chapter 4: Requirements Traceability	Salvatore	3 hours
Chapter 5: Implementation, Integration and Testing Plan	Salvatore	6 hours
Chapter 6: Effort Spent and final review	Ratti, Salvatore, Yalcin	8 hours

7 References

7.1 References and Tools

- **Overleaf** to compile and format this document.
- **draw.io** to do diagrams and to draw the component diagram and the deployment view.
- **GitHub** to share and collaborate on the project.
- **Google Docs** to write notes for writing this document.
- **Figma** to draw the mockups.