POLITECNICO
MILANO 1863

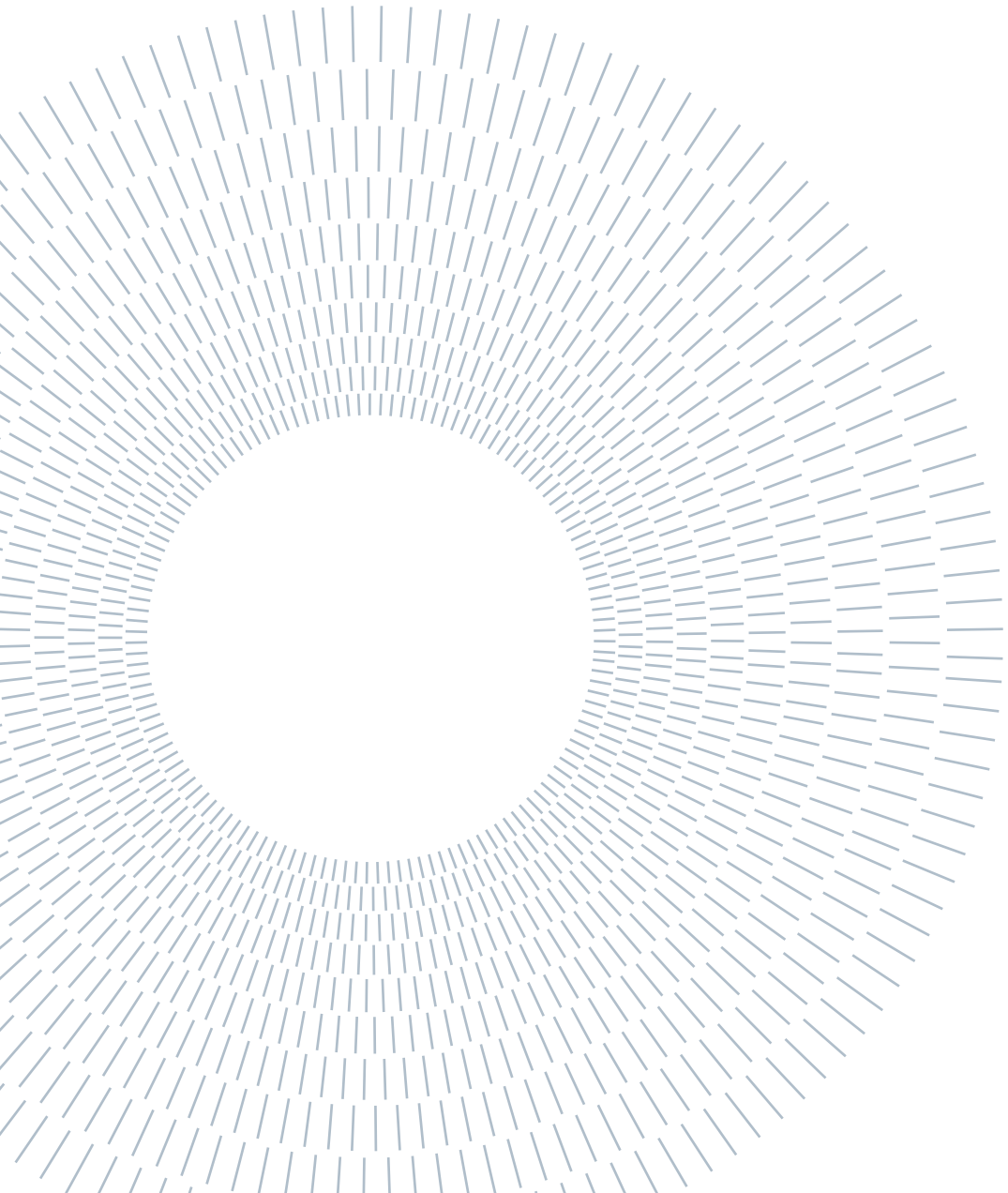
SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Design Document

Students&Companies

Author:
Riccardo Bonfanti
Jie Chen

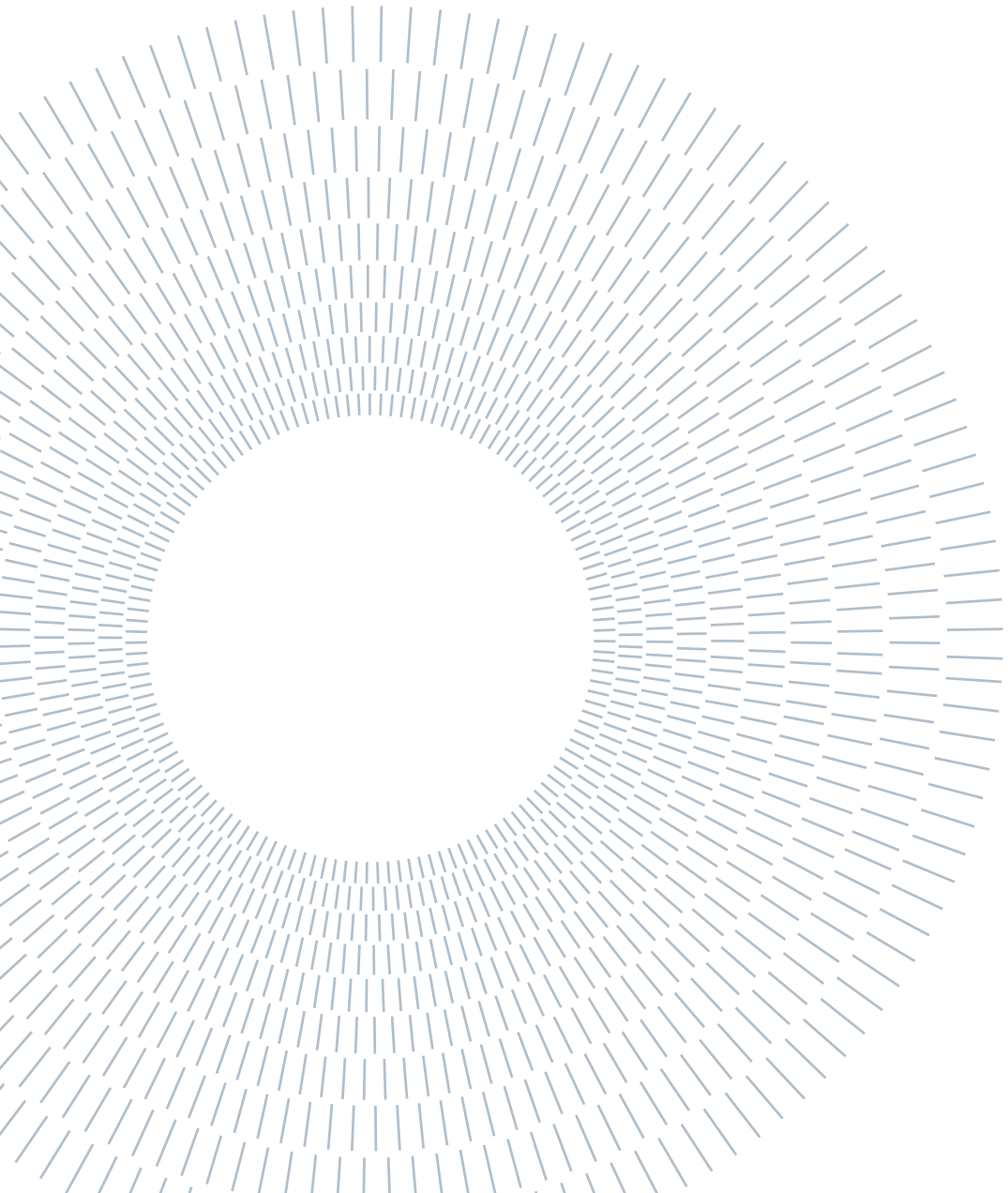
Student ID: 273115 276324
Advisor: Prof.Elisabetta Di Nitto
Academic Year: 2024–2025



Deliverable Information

Deliverable:	DD
Title:	Design Document
Authors:	Riccardo Bonfanti, Jie Chen
Version:	1.0
Date:	22-December-2024
Download page:	https://github.com/JieCver1/BonfantiChen
Copyright:	

Contents



1 | Introduction

A. Purpose

The purpose of this document is to provide a detailed description of Student&Companies. It will help developers to implement the required system features and it should provide the customer with a clear description of the system, allowing him to verify that it meets the specified requirements.

B. Scope

Student&Companies is a platform that connects students, companies, and universities to facilitate the internship research, announcement and selection process. The platform provides services such as internship announcements, profile management, a recommendation system, interview management, and performance feedback. The platform is designed to be user-friendly and easy to use for students, companies, and universities. It is a web-based application that can be accessed from any device with an internet connection.

C. Definition,acronyms,abbreviations

C.1. Definitions

- **Student:** A person who is looking for internships.
- **Company:** An organization which wants to announce internship opportunities to students.
- **University:** An educational institution that is related to students and their internships.
- **User:** A generic term for students, companies, and universities who use the platform.
- **Candidate:** A term for students whose applications are selected and that will take

part in the interview process.

- **Internship:** A opportunity offered by companies to students to gain practical experience in a real job environment.
- **CV:** Curriculum Vitae, a document that contains all necessary information about students to be able to apply for internships.
- **Recommendation:** A suggestion made by the platform to students and companies based on statistical analyses and keyword searches.
- **Interview:** A questionnaire form, that can be followed by an external meeting between students and companies, to evaluate the student preparation and make him understand what the company is looking for.
- **Feedback:** Helpful information written by students and companies about their internship experiences to improve a performance of the two parties.
- **Notification:** A message sent by the platform to inform students and companies about important events, such as new internship offers, matching CVs, interview results etc.
- **Interview:** A meeting between students and companies to decide an assignment of the internship offer.
- **Platform:** The Students&Companies (S&C) system that provides the services to students, companies, and universities about internships.
- **Keyword:** A significant word or tag used to describe content, such as the skills, experiences, and preferences of students and companies.
- **Comment:** The text that is written by students and companies to provide feedback or complaints about their internship experiences.
- **Complain:** A text that expresses dissatisfaction, issues, or annoyance about the internship experiences. It will be treated as a synonym of feedback in this document.

C.2. Acronyms

- **S&C:** Students&Companies
- **CV:** Curriculum Vitae
- **UI:** User Interface
- **UX:** User Experience

- **API:** Application Programming Interface
- **HTTPS:** Hypertext Transfer Protocol Secure
- **TLS:** Transport Layer Security
- **REST:** Representational State Transfer

C.3. Abbreviations

- **CO:** Company
- **ST:** Student
- **UNI:** University

D. Revision History

E. Reference Documents

- Assignment RDD AY 2024–2025.

F. Document Structure

This document is structured as follows:

- **Section 1: Introduction**
- **Section 2: Architectural Design**
- **Section 3: User Interface Design**
- **Section 4: Requirements Traceability**
- **Section 5: Implementation, Integration, and Test Plan**
- **Section 6: Effort Spent**
  The time spent by each group member on each task will be registered in this section.

It will be used to present the effort dedicated by each member and to present the progress of the development of the project.

- **Section 7: References**
  The other references that not include in the reference documents will be added in this section.

2 | Architectural Design

A. Overview

S&C will be developed as a multi-tiered, client-server architecture, as shown in Figure ??. The system will be divided into three main layers: the presentation layer, the application layer, and the data layer. The presentation layer will be responsible for managing the user interface and the user interaction. The application layer will be responsible for managing the application logic. The data layer will be responsible for managing the data storage and the data access.

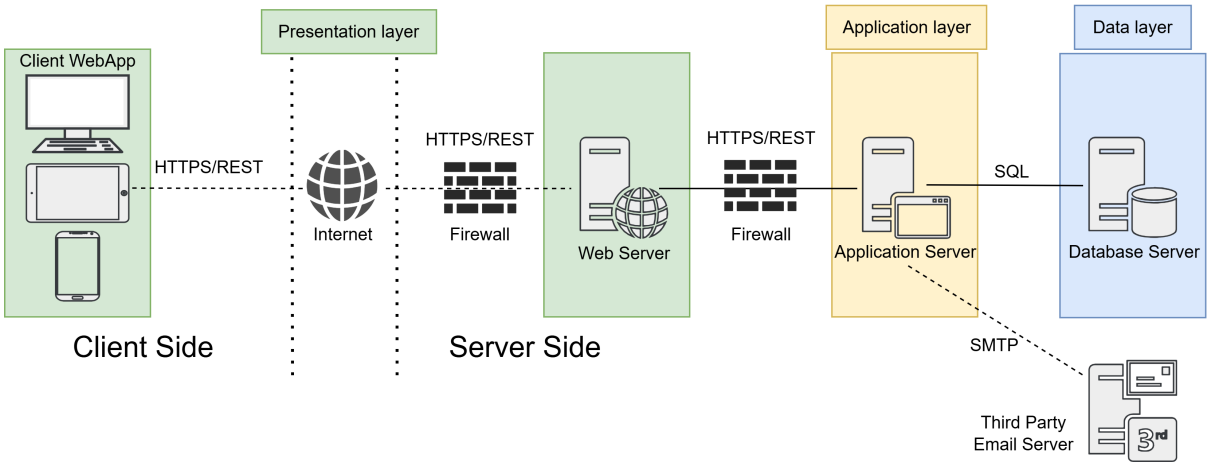


Figure 2.1: S&C architectural overview

In the following paragraph we describe each tier presented in Figure ??: and what layer it deploys:

Client Side

- **Web App:** It is the user interface. It will be responsible for managing the user interaction. This means that it will be responsible for hosting part of the presentation layer.

Server Side

- **Firewall:** It will be responsible for managing the security of the system, by filtering the incoming and outgoing traffic and restricting the access based on predefined rules. It will be placed between the Web Server and the Internet and between the Application Server and the Web Server. In this way, the web server will reside in a DMZ (Demilitarized Zone), while the application server will resides in a protected internal network.
- **Web Server:** It serves as a gateway between the client and the application server (backend). It will be responsible for hosting part of the presentation layer. For example, it will be responsible for serving the web pages to the client, handling requests routing to the application server, managing load balancing, and handling security.
- **Application Server:** It will be responsible for managing the application logic. This means that it will host the application layer. For example, it will be responsible for processing the client requests, execute the business logic, and coordinates with the database and email server.
- **Database Server:** It will be responsible for managing the data storage and the data access. This means that it will host the data layer. For example, it will be responsible for storing and retrieving the application data, and executing the database queries.
- **Mail Server:** It will be responsible for managing the email communication. This means that it will be responsible for sending emails to the users. It is triggered by the application server.

The Figure?? also shows how the tiers interact with each other. The Web App interacts with the Web Server through HTTPS/REST requests; the Web Server interacts with the Application Server through HTTPS/REST calls; the Application Server interacts with the Database Server through SQL queries; and the Application Server interacts with the Mail Server through SMTP requests.

B. Components view

B.1. High-level components and interactions

B.2. Low-level components and interactions

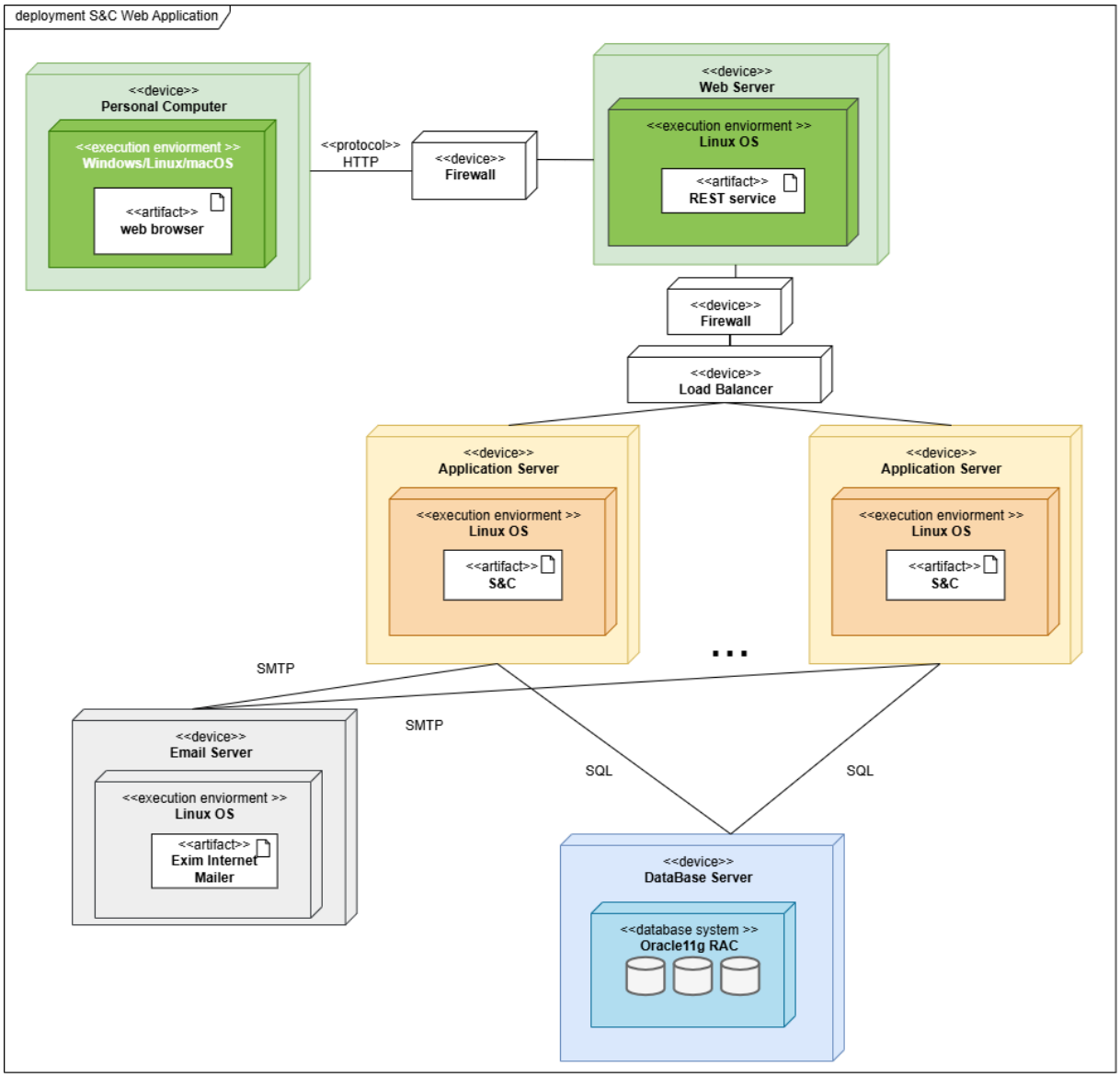


Figure 2.2: S&C deployment diagram

C. Deployment view

The infrastructure of the S&C platform is described in below using a deployment diagram and a description of the components and their interactions. As described at the beginning of this document, the system is divided into three tiers: the presentation layer, the application layer, and the data layer. The deployment diagram in Figure **??** shows the physical distribution of the components across different servers and the communication between them.

- **Personal Computer:** Anyone interested in using the platform can access it through any type of personal computer. Users can also access the platform using any device

capable of running a web browser. This component communicates and interacts with the Web Server via HTTPS protocols and RESTful API services.

- **Web Server:** The Web Server hosts the web application and serves web pages to users. It handles HTTP requests from clients and forwards them to the Application Server. Together with the Firewall and Load Balancer, it ensures the security, scalability, and availability of the system. It acts as a gateway between the client and the Application Server.
- **Application Server:** This component contains the core application logic of the platform, managing the operations necessary to provide services and functionalities to users. Thanks to RESTful API services and the Load Balancer, it can handle multiple requests simultaneously without the risk of overloading or crashing. It interacts with the Database Server to access or store data, such as recording a student's new internship. Additionally, it communicates with the Mail Server, particularly during user registration, to authenticate email addresses and verify user identities.
- **Database Server:** Most primary operations of the platform require frequent interaction with the Database Server. This component manages personal data, tracks the progress of applications and internships, stores chat histories between users, and more. The Application Server retrieves or stores data using SQL queries.
- **Mail Server:** During the registration process, the platform requires users to verify their email addresses to confirm their identity and complete account creation. The Application Server interacts with the Mail Server via SMTP requests to send verification emails.
- **Firewall:** The Firewall protects the core components of the system by filtering incoming and outgoing traffic and restricting access based on predefined rules. For example, if an unauthorized individual attempts to access the Application Server and modify data in the Database Server without proper credentials, the Firewall blocks the attempt, preventing unauthorized access.
- **Load Balancer:** The Load Balancer distributes incoming client requests to ensure that no one application server is overwhelmed to operate at peak efficiency. It is placed between the Web Server and the Application Server, in order to optimize the performance and availability of the system.

D. Component interfaces

In the following paragraphs, we describe the interfaces of the main components of the platform specifying the methods that each component can perform. Note:

- *userNames* are unique identifiers generated by the system for each user once they register on the platform and are used to access the data related to that user.

- *userTypes* enumeration is used to distinguish between the different types of users that can register on the platform: Student, Company, University.

- *SearchType* enumeration used to distinguish between the different types of searches that can be performed on the platform: Internship and User Profile.

- *NotificationType* enumeration used to distinguish between the different types of notifications that can be sent on the platform in order to handle them in a different way.

- *InternshipID* is a unique identifier generated by the system for each internship published successfully and is used to access the data related to that internship.

- *ChatID* is a unique identifier generated by the system for each chat created between two users and is helpful to access the data related to that chat more easily.

- *ApplicationID* is a unique identifier generated by the system for each application submitted by a student for an internship and is used to access the data related to that application and take a track of its status.

Registration Manager

- createAccount (): None
- selectUserType (UserType userType): Boolean
- registration (String email, String password, String name, String surname, University university, List[Field] interests, List[String] skills): Boolean
- registration (String email, String password, String legalName, String EIN, String department, List[Field] fields): Boolean
- registration (String email, String password, String name, String surname, String legalName): Boolean
- generateUserName (): String

Login Manager

- login (String email, String password): Boolean

User Manager

- uploadCV (Student userName, File cv): Boolean
- updatePersonalInfo (String userName, String name, String surname): Boolean
- updateUniversityInfo (String userName, University university): Boolean
- updateInterestsAndSkills (String userName, List<Field> interests, List<String> skills): Boolean
- updatePassword (String userName, String oldPassword, String newPassword): Boolean
- updateLegalInfo (String userName, String legalName, String EIN, String department): Boolean
- updateProfilePicture (String userName, File profilePicture): Boolean
- addFieldOfInterest (String userName, Field field): Boolean
- removeFieldOfInterest (String userName, Field field): Boolean
- addSkill (String userName, String skill): Boolean
- removeSkill (String userName, String skill): Boolean

View Profile Manager

- getProfile (String userName): Profile
- getListOfEnrolledStudents (University university): List[Student]

Search Manager

- search (String keyword): List[Result]
- searchFilterByType (SearchType type, List[Result]): List[Result]
- searchFilterByField (Field field, List[Result]): List[Result]
- searchFilterByLocation (String location, List[Result]): List[Result]
- searchFilterByTime (Date startDate, Date endDate, List[Result]): List[Result]

Notification Manager

- notify (String userName, String notification, NotificationType type): Boolean
- getNotifications (String userName): List[Notification]
- getNotificationDetails (String userName, String notification): Notification
- isRead (String userName, String notification): Boolean
- deleteNotification (String userName, String notification): Boolean
- deleteAllNotifications (String userName): Boolean

Recommendation Module

- getRecommendations (String userName): List[Recommendation]
- generateRecommendations (String userName): List[Recommendation]
- revaluateRecommendationsList (List[Recommendation] oldRecommendations): List[Recommer
- needRecommendationUpdate (String userName): Boolean

Internship Manager

Creation Manager

- createInternship (): None
- publishInternship (String title, String description, Field field, String location, Date startDate, Date endDate, int duration, int totalPositions, Date deadline): Boolean
- generateInternshipID (): String
- addInternshipInList (String internshipID, Company company): Boolean

View Internship Information Manager

- getInternshipInformation (String internshipID): Internship
- getInternshipList (Company company): List[Internship]
- getInternshipList (Student student): List[Internship]

Selection Manager

- selectStudentForInterview (String internshipID, List[Student]): List[Student]

- selectStudentForInternship (String internshipID, List[Student]): List[Student]
- getSelectedStudents (String internshipID): List[Student]
- updateStudentStatus (String internshipID, Student student, Status status): Boolean
- getCandidatesList (String internshipID): List[Student]

Feedback Manager

- writeFeedback (String internshipID, String feedback, String userName): Boolean
- getFeedback (String internshipID): List[String]

Chat Manager

- sendMessage (String ChatID, String message, String sender, String receiver): Boolean
- getChatHistory (String ChatID): List[Message]
- createChat (String userName1, String userName2): String
- getChatID (String userName1, String userName2): String
- openChat (String ChatID): Boolean
- closeChat (String ChatID): Boolean
- getChatList (String userName): List[Chat]

Application Manager

Submission Manager

- applyForInternship (String internshipID): Boolean
- submitApplication (String internshipID, String userName): Boolean
- generateApplicationID (): String
- acceptOffer (String internshipID, String userName): Boolean
- rejectOffer (String internshipID, String userName): Boolean
- getApplicationStatus (String internshipID, String userName): Status
- updateApplicationStatus (String internshipID, String userName, ApplicationStatus status): Boolean

- getApplicationList (String userName): List[Application]

View Application Information Manager

- getApplicationList (String internshipID): List[Application]
- getApplicationStatus (String applicationID, String userName): Status

Interview Manager

- createInterviewForm (): None
- submitInterviewForm (String descriptionLetter, List[String] question): Boolean
- recordInterviewResults (String formID, String userName, ApplicationStatus InterviewResult): Boolean
- getInterviewResults (String applicationID): List[ApplicationStatus]
- addQuestion (String question): List[String]
- removeQuestion (String question): List[String]

Questionnaire Manager

- getInterviewForm (String formID): InterviewForm
- respondForm (String formID, List[String] answers): Boolean
- getFormResponses (String internshipID): List[Questionnaire]
- getSpecificFormResponse (String formID): Questionnaire

E. Runtime view

F. Selected architectural styles and patterns

F.1. Three-Tiered Architecture

As described in Section Overview, the Students&Companies (S&C) platform is build using a multi-tier architecture. This decision was made with the aim of providing a more scalable and flexible system. The system is divided into three main layers: the presentation layer, the application layer, and the data layer. Each layer has its own responsibilities and plays a specific role in the system: the presentation layer serves as the front end, accessible

through the GUI, while the application layer and data layer together form the back end of the system, accessible via API-based methods.

- **Presentation Layer:** The presentation layer is implemented as a generic web application accessible through a web browser. It is responsible for managing the presentation logic, including user interaction, the user interface, and rendering information. This layer serves as the front end of the system, the only part that the user can access directly.
- **Application Layer:** The application layer is implemented as a set of RESTful web services. It is responsible for managing the functional logic of the system, controlling communication between the presentation layer and the data layer. This layer allows the system to react to user input and generate appropriate responses accordingly. It includes the Application Server and is also used to interact with third-party services.
- **Data Layer:** The data layer is responsible for managing the data storage and access within the system. All operations that require data manipulation must be performed through interactions with the data layer. The platform uses a Relational Database Management System (RDBMS), making the data accessible through SQL queries.

F.2. RESTful API

The Representational State Transfer (REST) style is designed to be stateless, enabling more efficient and seamless communication between the client and the server. It uses standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. The decision to incorporate RESTful APIs into the architecture provides advantages in terms of performance, modifiability, and simplicity by defining conventions for interacting with resources in resource-oriented manner.

F.3. Model-View-Controller (MVC) Pattern

One of the most recommended design patterns for the Three-Tier Architecture is the Model-View-Controller (MVC) pattern. It separates the application into three components: the model, the view, and the controller, minimizing interdependencies between the components and improving the maintainability, manageability, and scalability of the system. Each component can be developed, tested, and maintained independently.

- **Model:** Contains the state and application logic and is independent of the other components.

- **View:** Represents the visual presentation logic of the Model and is responsible for displaying data to the user.
- **Controller:** Acts as an intermediary between the Model and the View. It receives user input forwarded by the View, then processes operations and updates the Model and the View accordingly.

G. Other design decisions

- **Design patterns related to the behavioral aspects:** Observer Pattern and State Pattern.
- **Design decisions related to the system's requirements:** Some design decisions are already described in the RASD, such as reliability, availability, scalability, security, maintainability, and portability. The following sections revisit availability, scalability, and security to emphasize their importance in the system.

G.1. Observer Pattern

The Observer pattern is particularly useful when multiple objects need to be notified about a change in the state of another object. In the context of the S&C platform, a large number of functionalities require the participation of multiple objects, such as notifying users about the results of an interview or updates on the status of an application.

G.2. State Pattern

The State pattern is recommended to efficiently manage operations across different states and handle transitions between them, as it allows objects to change their behavior when their internal state changes. In the context of the S&C platform, the State pattern can be used to manage the lifecycle of an application for a internship position.

G.3. Availability

The system is designed to be highly available, ensuring that users can access the platform at any time, as described in the RASD, with at least 99.8 percent uptime. To achieve this, critical components should be replicated across multiple servers to provide redundancy and fault tolerance in case of failure. Load balancing is correctly configured to distribute incoming traffic, preventing overload on any single server. Continuous monitoring of the system's performance allows for the detection and resolution of any issues that may arise

in real-time.

G.4. Scalability

The platform is designed to handle increased user loads in the future by scaling individual layers independently, thanks to the architectural styles and patterns mentioned earlier. As described in the RASD, the system can be scaled horizontally by adding more servers or vertically by increasing the resources of existing servers, without affecting performance. This scalability is essential, especially as the number of users grows, leading to a higher volume of application requests over time.

G.5. Security

The system is designed to ensure the privacy and security of user data both during transmission over the network and while stored in the database. This includes the use of authentication and authorization mechanisms to ensure that only authorized users can access the system, reducing the risk of unauthorized access. Additionally, a firewall and Intrusion Detection System (IDS) are set up in the network to protect the system from external threats and attacks. Protocols like HTTPS are used to encrypt communication between the client and server, and other encryption algorithms are employed to protect sensitive data stored in the database, such as user passwords.

3 | User Interface Design

A. User Interface Design

In this section, the user interface design will be presented using mockups within the short description. The design focuses on optimizing the user experience and ensuring that the user can easily navigate on the website to perform the desired actions. There will be separate in subsection to describe more clearly the main pages needed to satisfy the user requirements.

A.1. Welcome Page

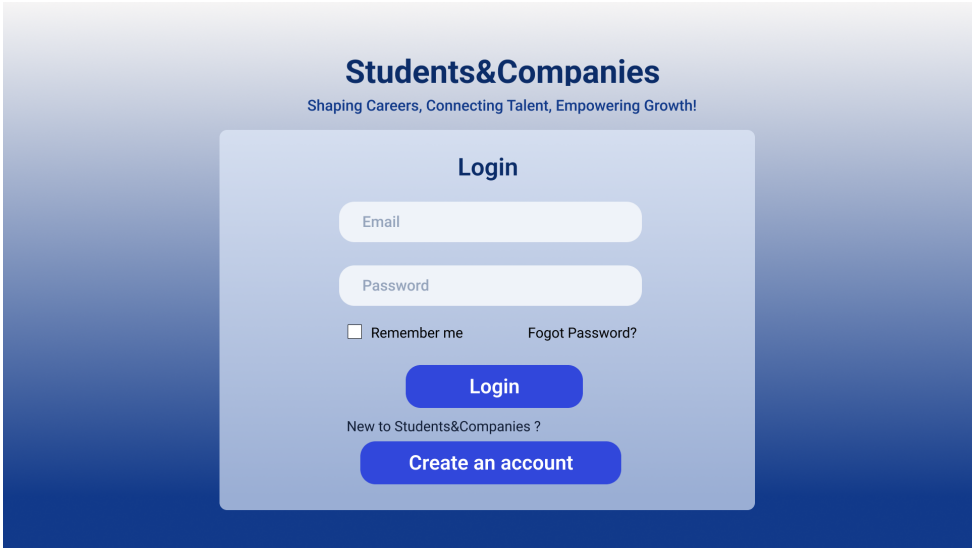


Figure 3.1: Welcome Page

A.2. Register Page

If the User is not registered and wants to create an account, they will be asked to choose the type of account they want to create. Clicking on the type listed will redirect to the corresponding Register Page where the user can fill in the required information.

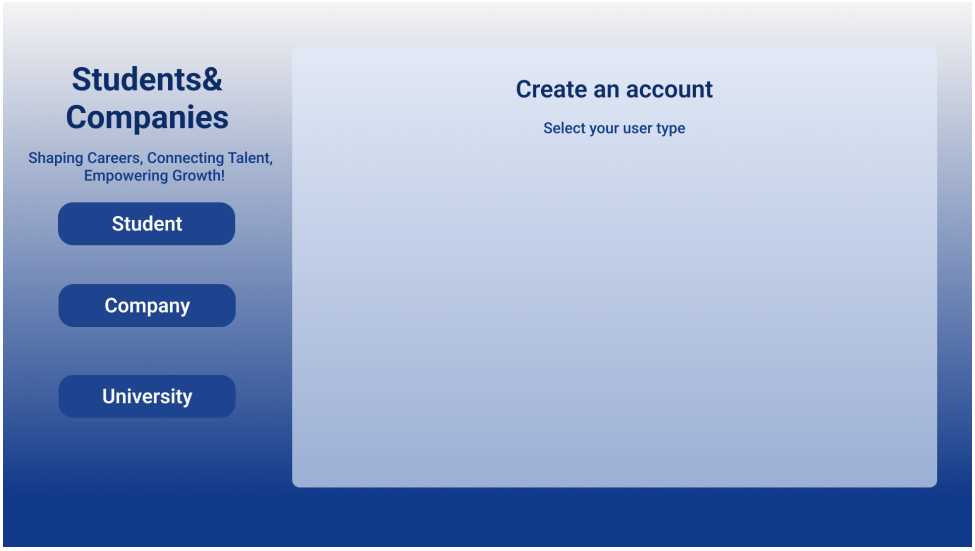


Figure 3.2: Register Page

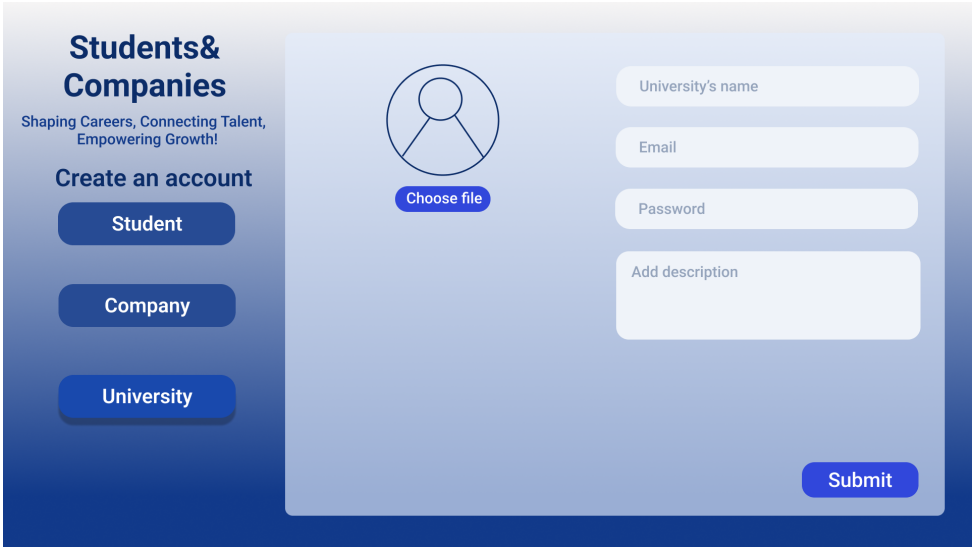


Figure 3.3: University create account

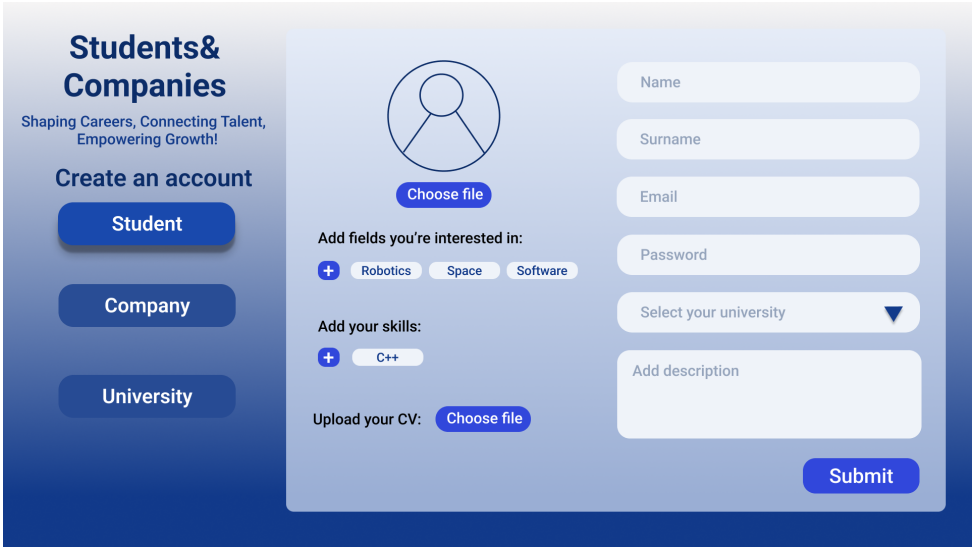


Figure 3.4: Student create account

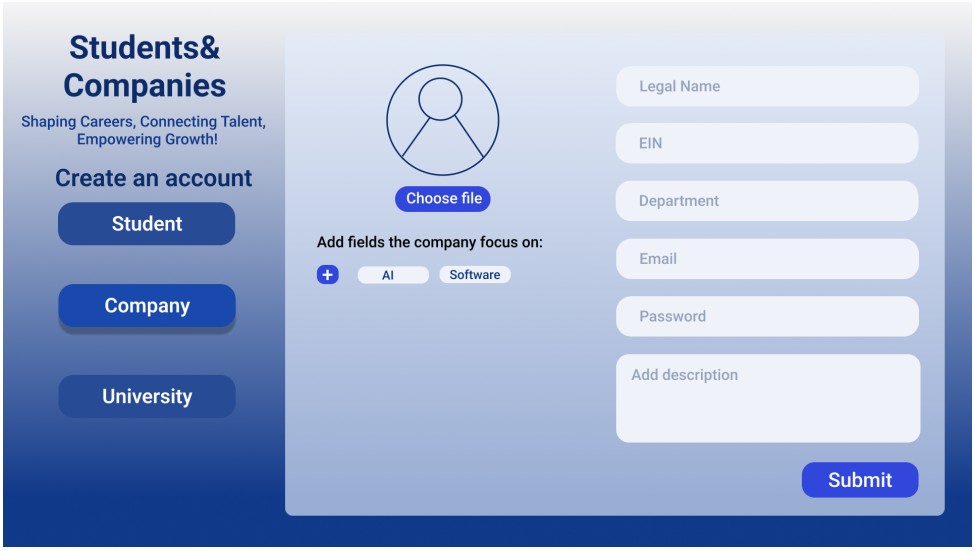


Figure 3.5: Company create account

A.3. Student's view

Once access to the platform, to optimizing the user experience, the student will be able to use the side menu to navigate the main functionalities provided by the platform to satisfy their needs.

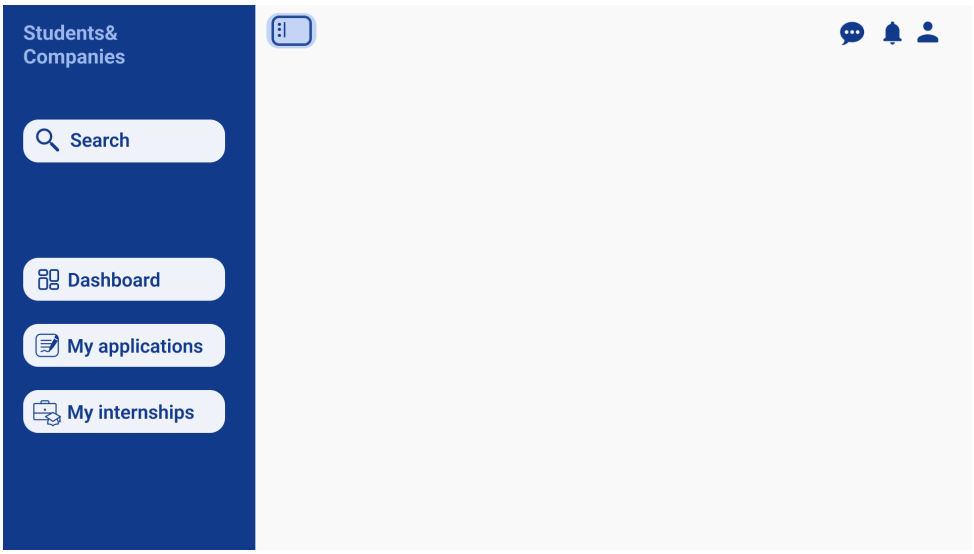


Figure 3.6: Student's Side Menu

By default, the student will be directed to the Dashboard page once logged in.

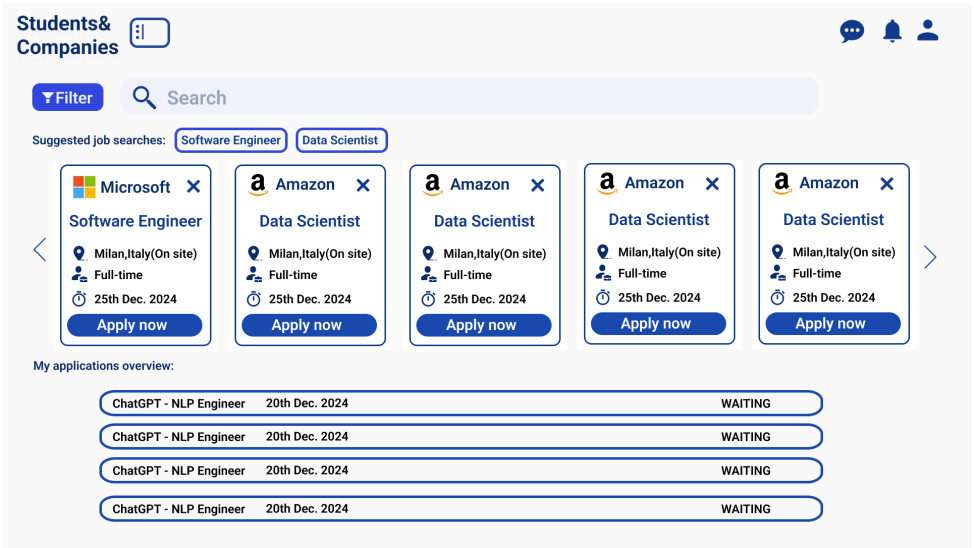


Figure 3.7: Student's Dashboard 1

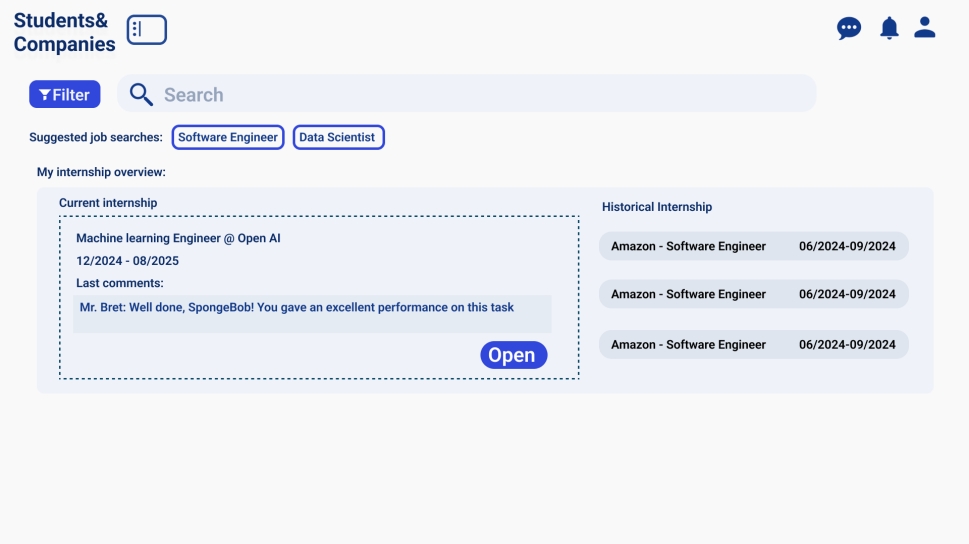


Figure 3.8: Student's Dashboard 2

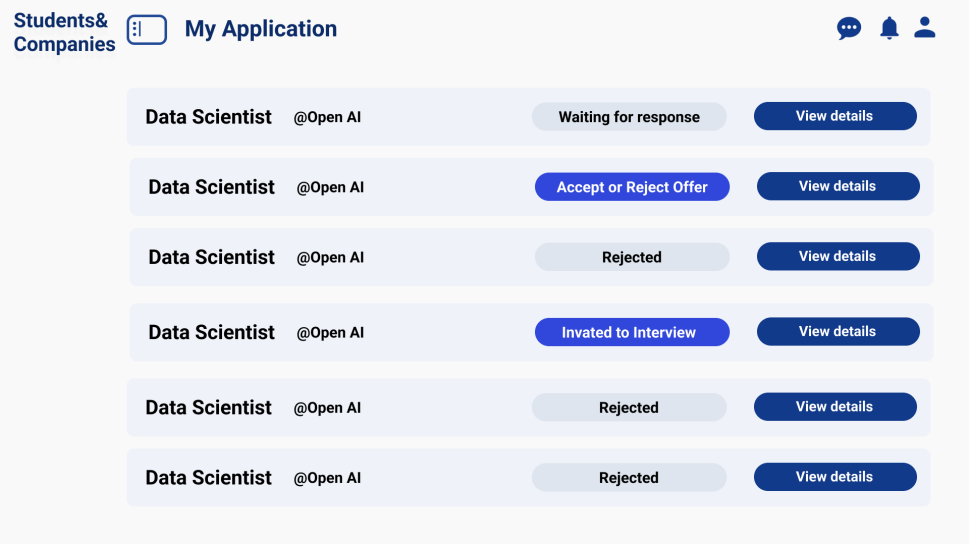


Figure 3.9: My application

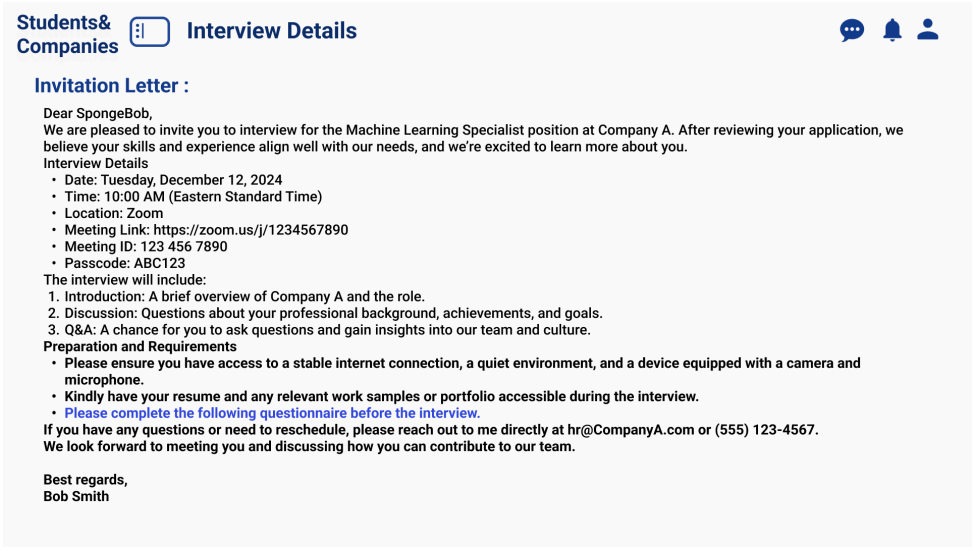


Figure 3.10: Interview details 1

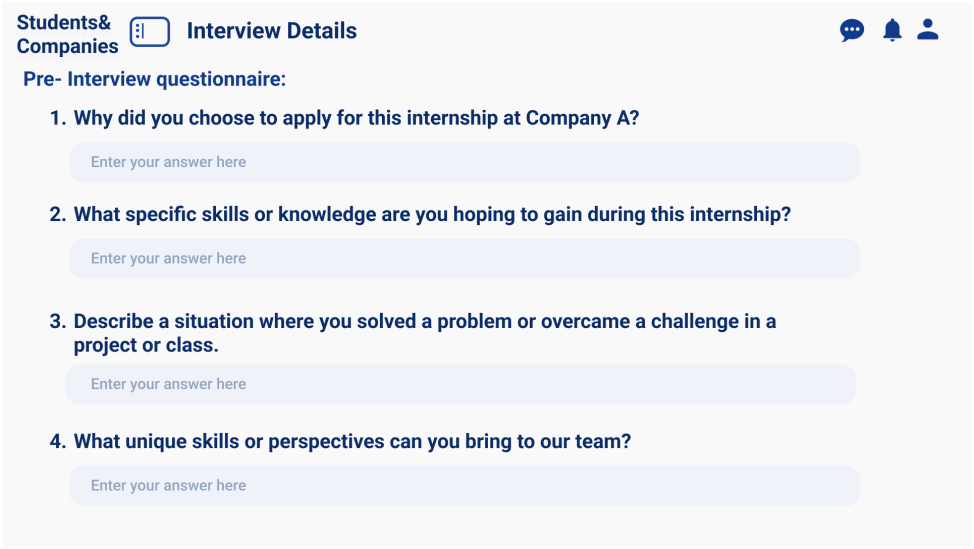


Figure 3.11: Interview details 2

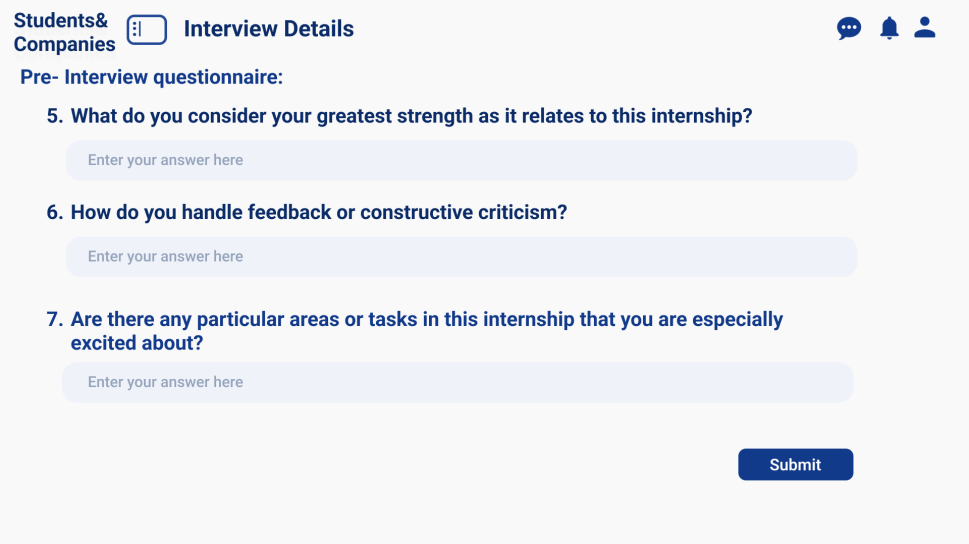


Figure 3.12: Interview details 3

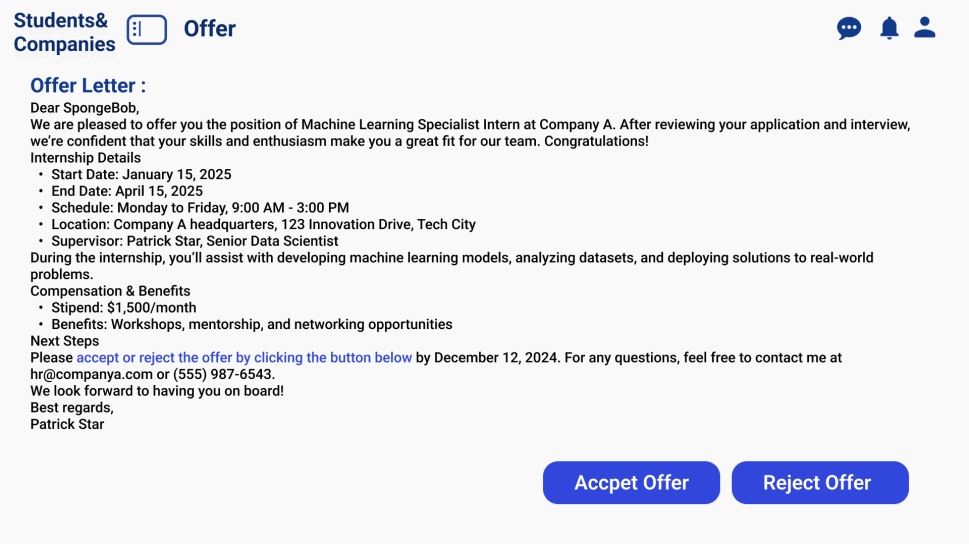


Figure 3.13: Accept or Reject Offer

Figure 3.14: My internship

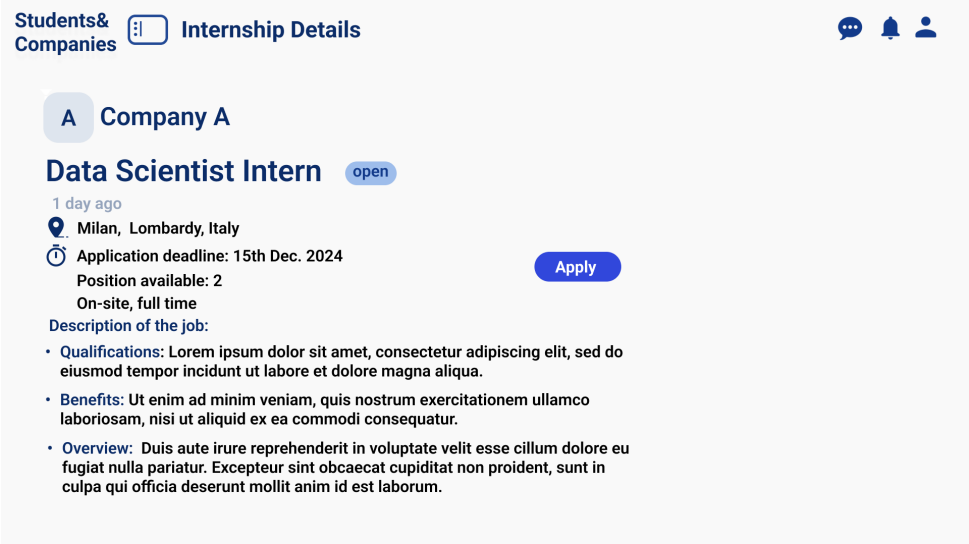


Figure 3.15: Internship announcement details

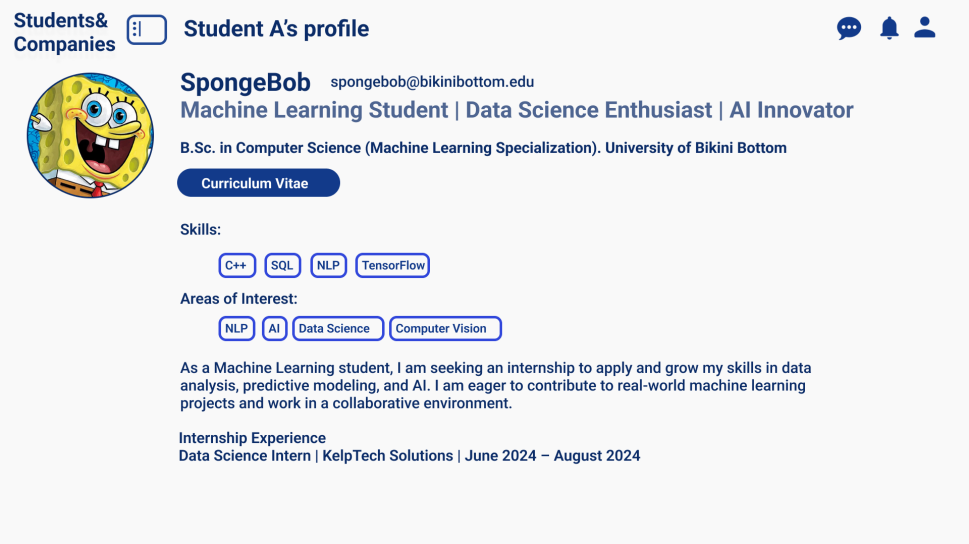


Figure 3.16: Student's profile from other's view

A.4. Company's view

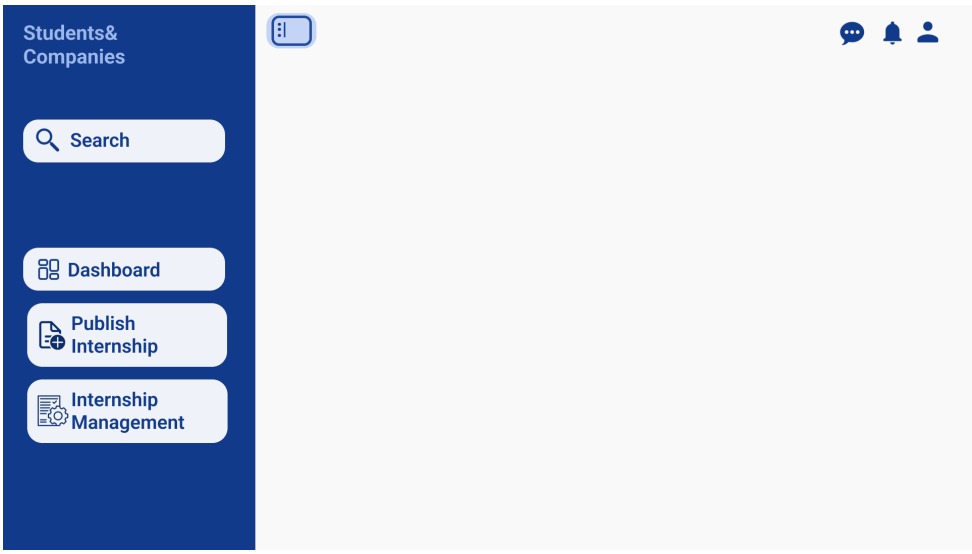


Figure 3.17: Company's Side Menu

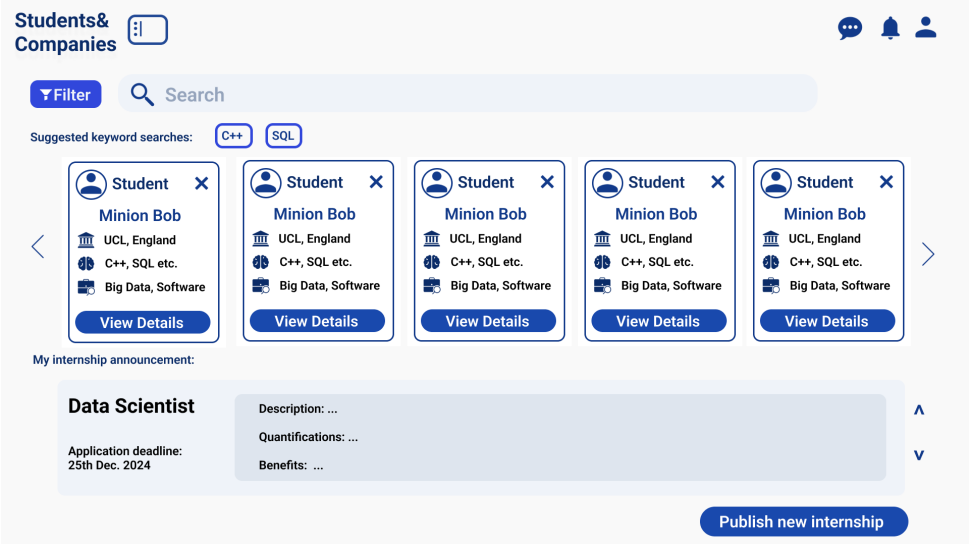


Figure 3.18: Company's Dashboard 1

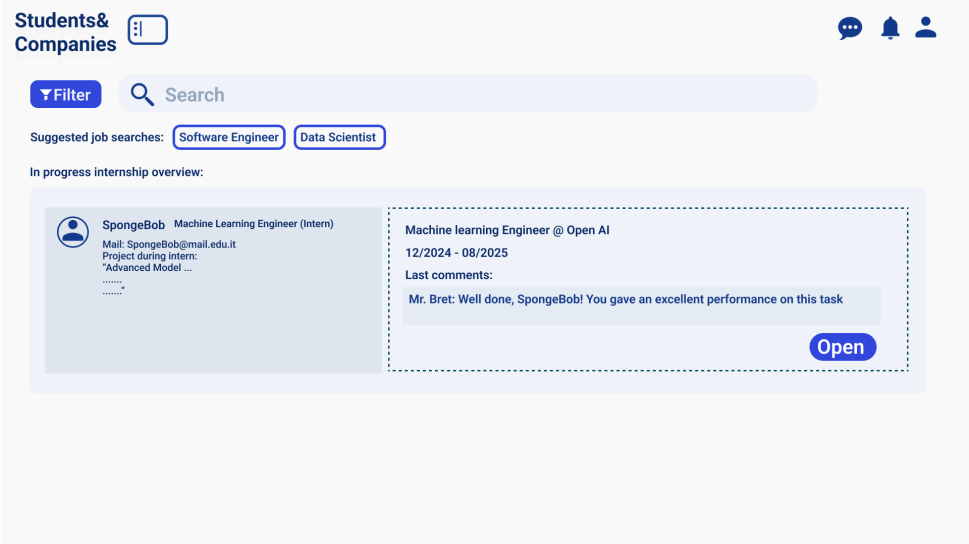


Figure 3.19: Company's Dashboard 2

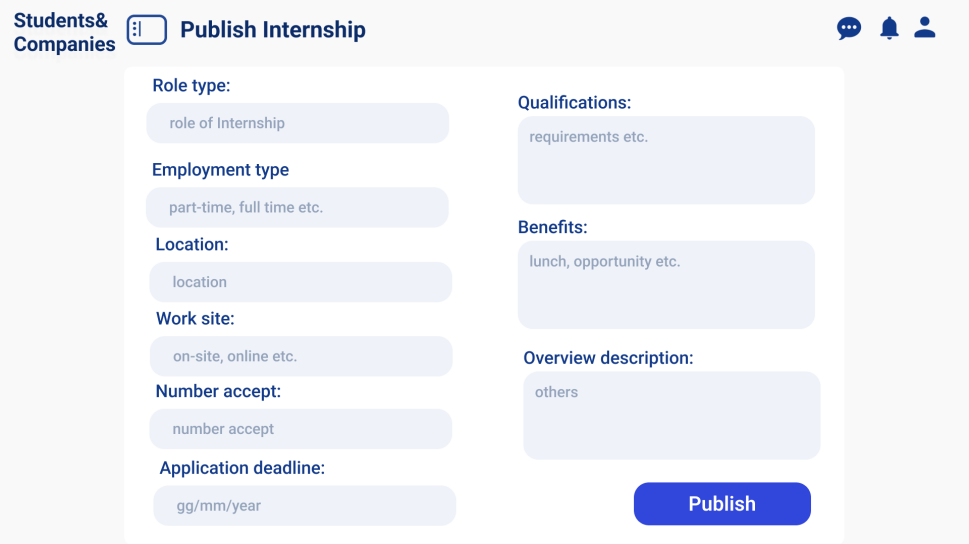


Figure 3.20: Publish Internship

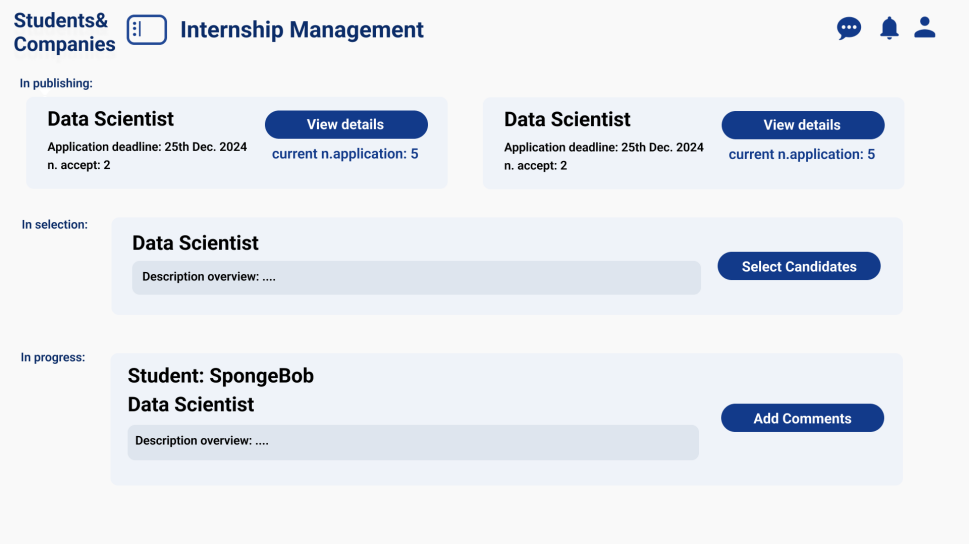


Figure 3.21: Internship Management 1

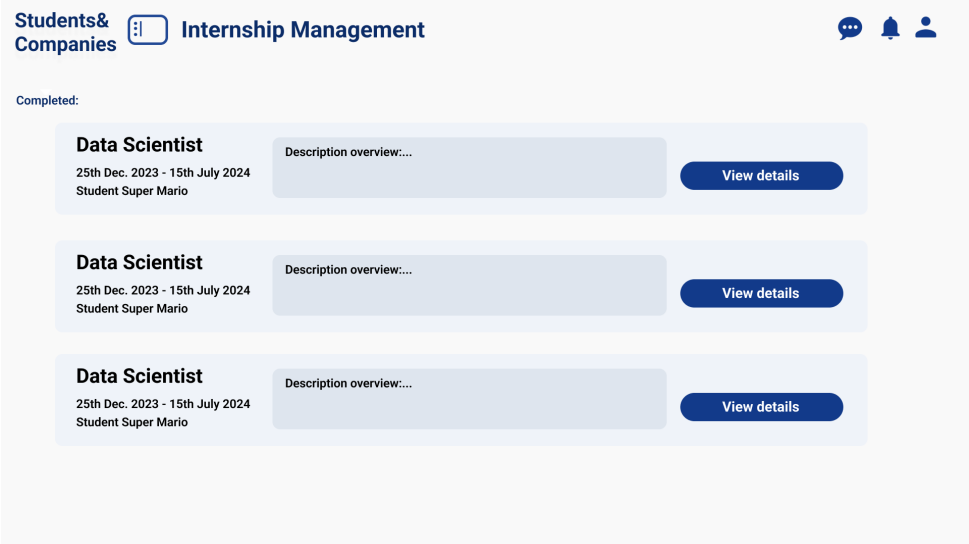


Figure 3.22: Internship Management 2

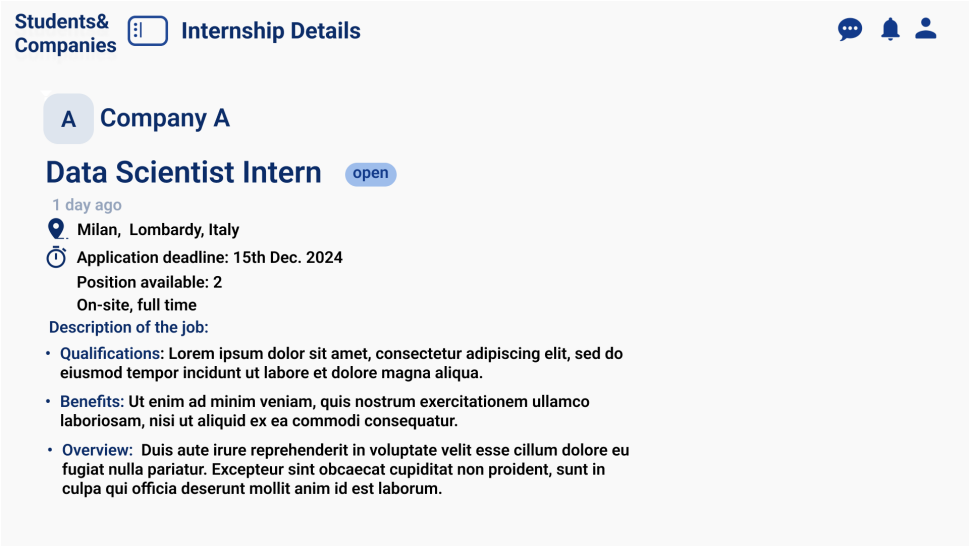


Figure 3.23: Internship details in publishing phase

Figure 3.24: Select candidates

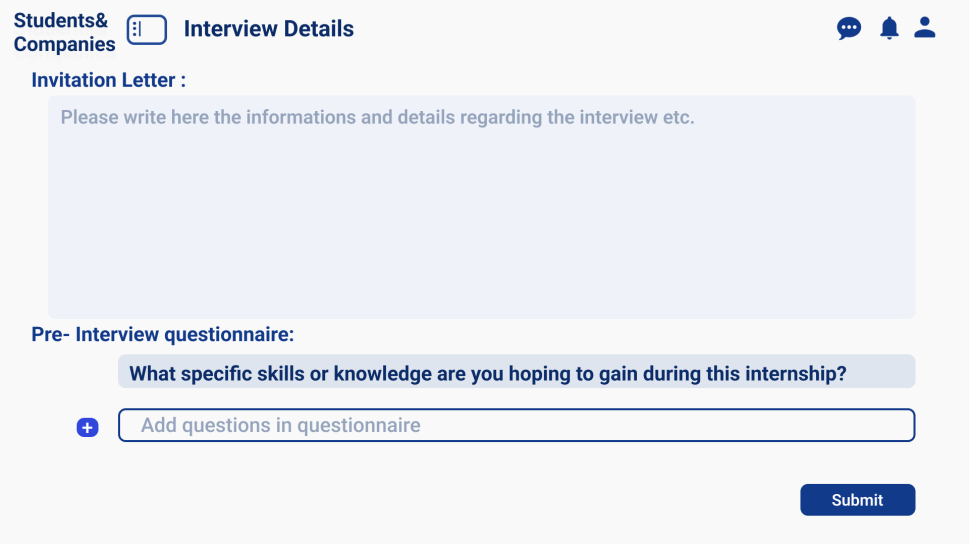


Figure 3.25: Set up interview

A.5. University's view

Figure 3.26: Feedback and Complaint

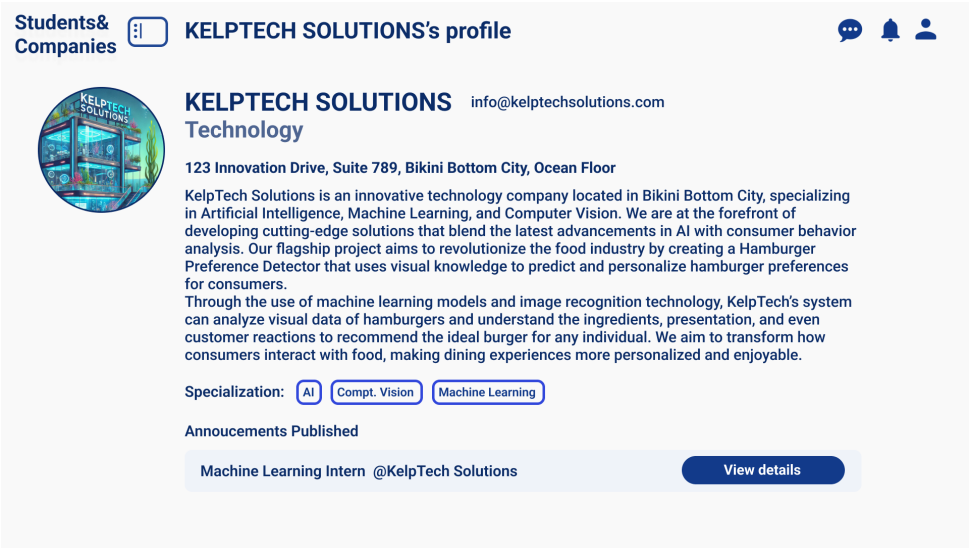


Figure 3.27: Company's profile from other's view

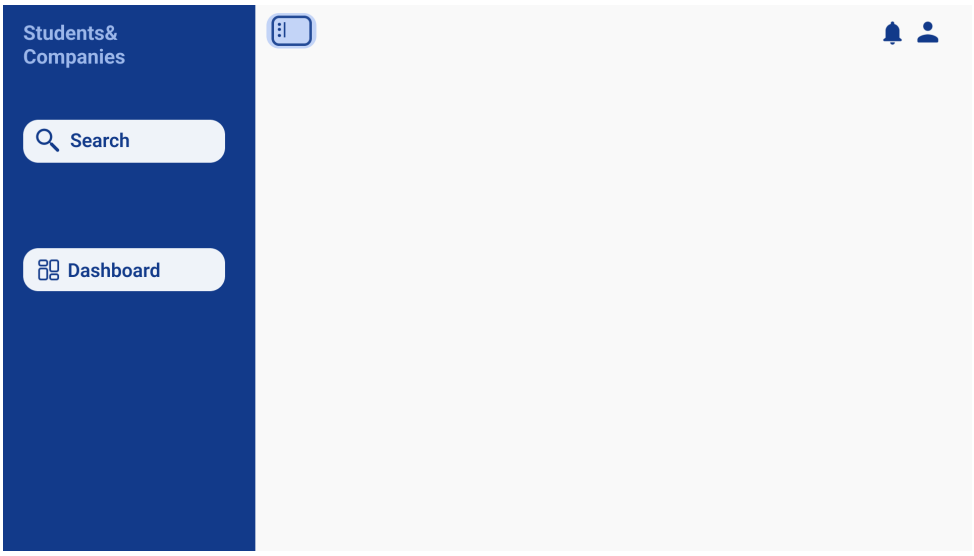


Figure 3.28: University's Side Menu

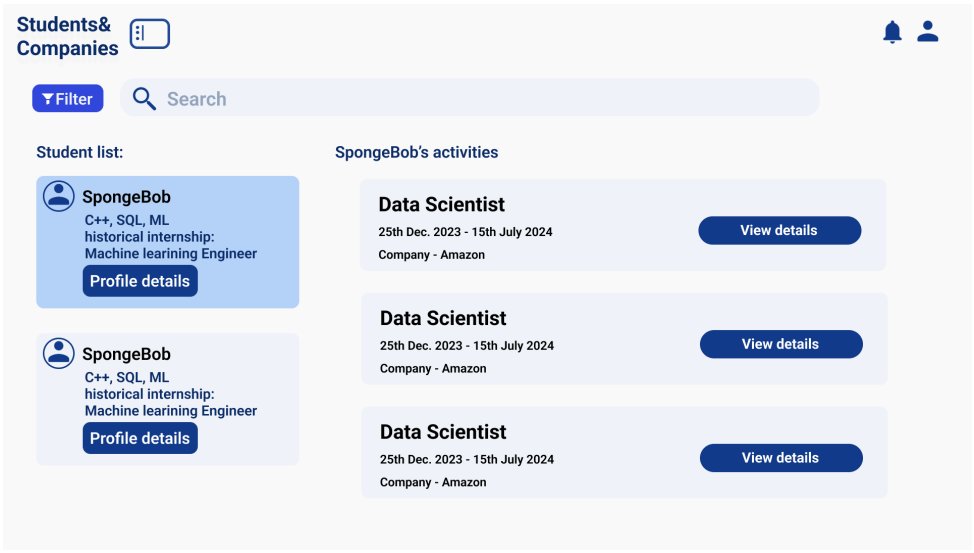


Figure 3.29: University's Dashboard 1

Figure 3.30: Feedback and Complaint

4 | Requirements Traceability

Authorization Manager

Registration Manager

- **R1**: S&C allows unregistered Users to sign up

Login Manager

- **R2**: S&C allows registered Users to login

User Manager

Profile Modification Manager

- **R3**: S&C allows STs to upload their CV in their profile section
- **R36**: S&C allows Users to modify their own profile data

View Profile Manager

- **R34**: S&C allows UNIs to access the list of all the enrolled STs that are registered on the platform
- **R35**: S&C allows Users to visualize the own and other users' profiles

Search Manager

- **R4**: S&C allows STs to search for internships

Notification Manager

- **R8**: S&C should notify STs when they are selected by the CO for the interview process
- **R9**: S&C should notify STs when a CO is interested in their profile

- **R10**: S&C should notify STs when he is successfully selected for a position
- **R11**: S&C should notify STs when a CO rejects their application
- **R12**: S&C should notify UNIs when their students start an internship
- **R13**: S&C should notify STs when an internship available matches their interest
- **R14**: S&C should notify COs when the deadline for a published internship has expired
- **R15**: S&C should notify COs when the candidate accepts the position
- **R16**: S&C should notify COs when the candidate refuses the position
- **R17**: S&C should notify COs and STs when a new chat message is available
- **R18**: S&C should notify COs when a ST with a CV that corresponds to their needs is available
- **R37**: S&C should notify UNIs when their students register on the platform

Recommendation Module

- **R19**: S&C should be able to analyze the User's data to provide the recommendations to both STs and COs

Internship Manager

Creation Manager

- **R5**: S&C allows COs to create internships by compiling all the information
- **R6**: S&C allows COs to set a deadline for submitting the application to an internship
- **R7**: S&C allows COs to publish internships

View Internship Information Manager

- **R21**: S&C allows STs to visualize information about a published internship
- **R33**: S&C allows UNIs to check the status of the internship records of their students

Selection Manager

- **R23**: S&C allows COs to record STs selection outcomes

Feedback Manager

- **R30**: S&C allows STs and COs to write feedback and complaints relating to the internship experience
- **R31**: S&C allows Users to view feedback and complaints relating to the internship experience

Chat Manager

- **R32**: S&C allows STs and COs to exchange information using the chat, only if the ST is participating or has participated in an internship offered by the company

Application Manager

Submission Manager

- **R20**: S&C allows STs to submit their application for an internship
- **R29**: S&C allows STs to accept or reject the offer after receiving the interview results

View Application Information Manager

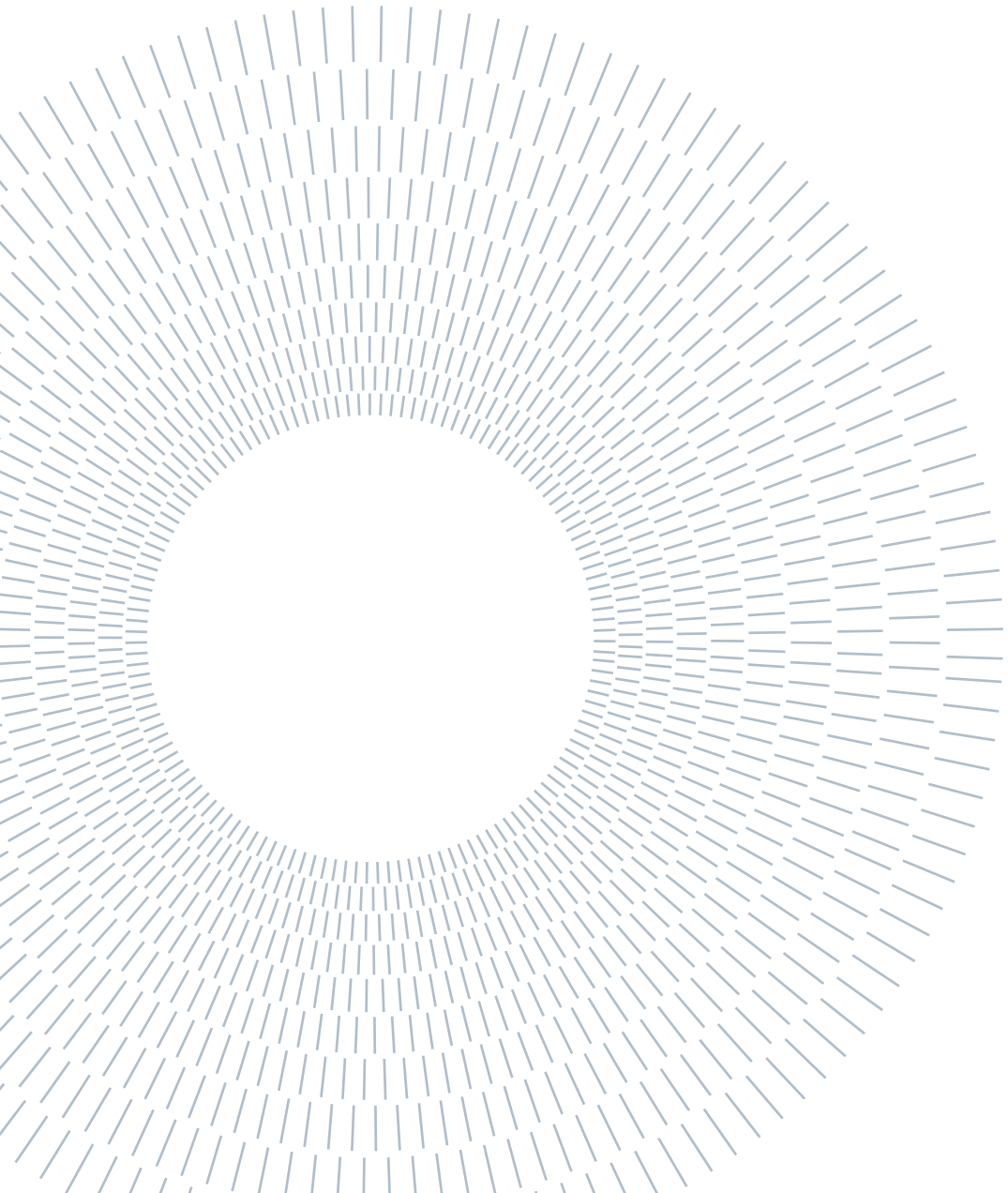
- **R22**: S&C allows COs to view the list of all applications that were submitted for a specific internship
- **R28**: S&C allows STs to check the status of their applications

Interview Manager

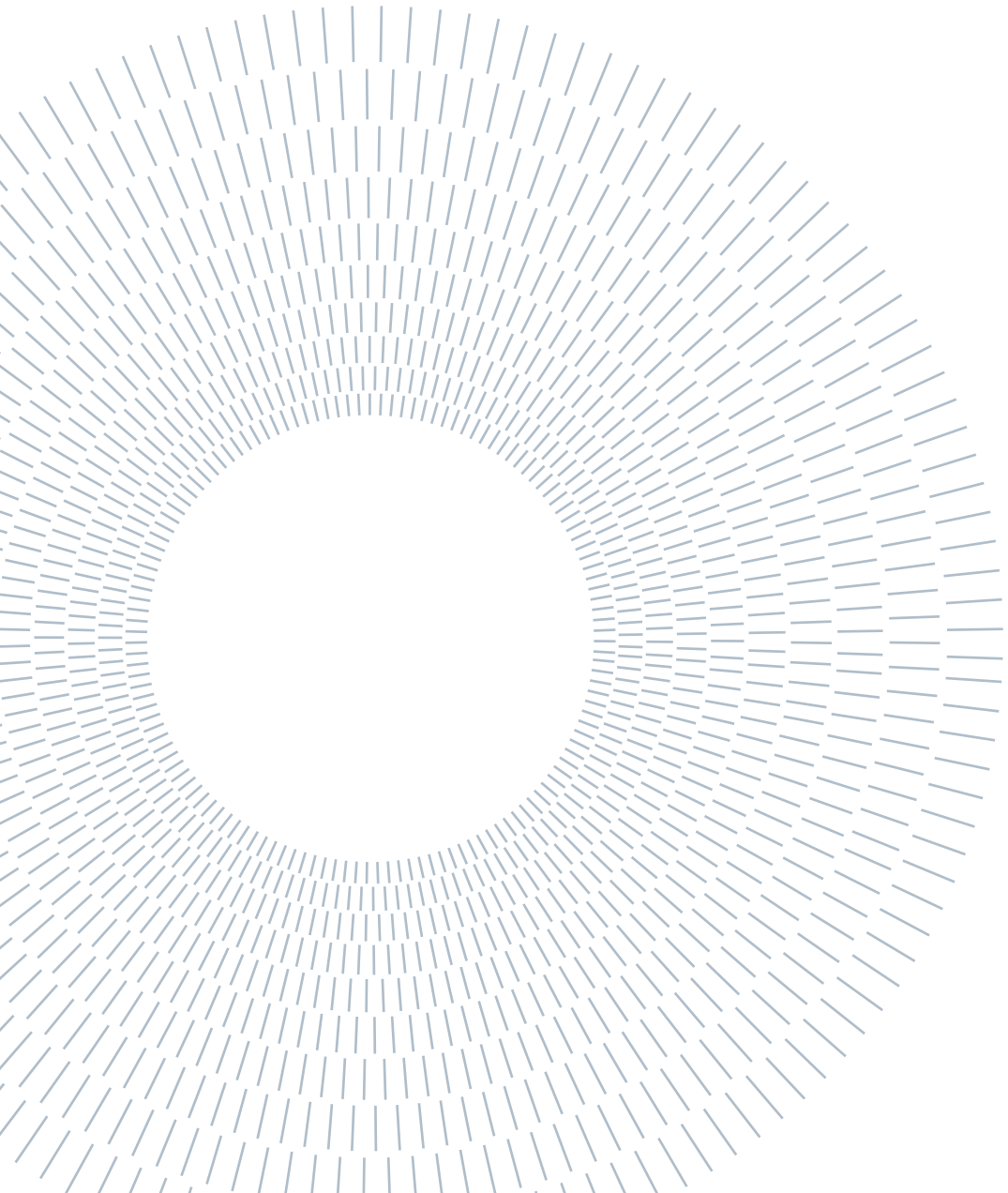
- **R24**: S&C allows COs to create forms to submit to candidates for the interview process
- **R27**: S&C allows COs to record the results of the interview

Questionnaire Manager

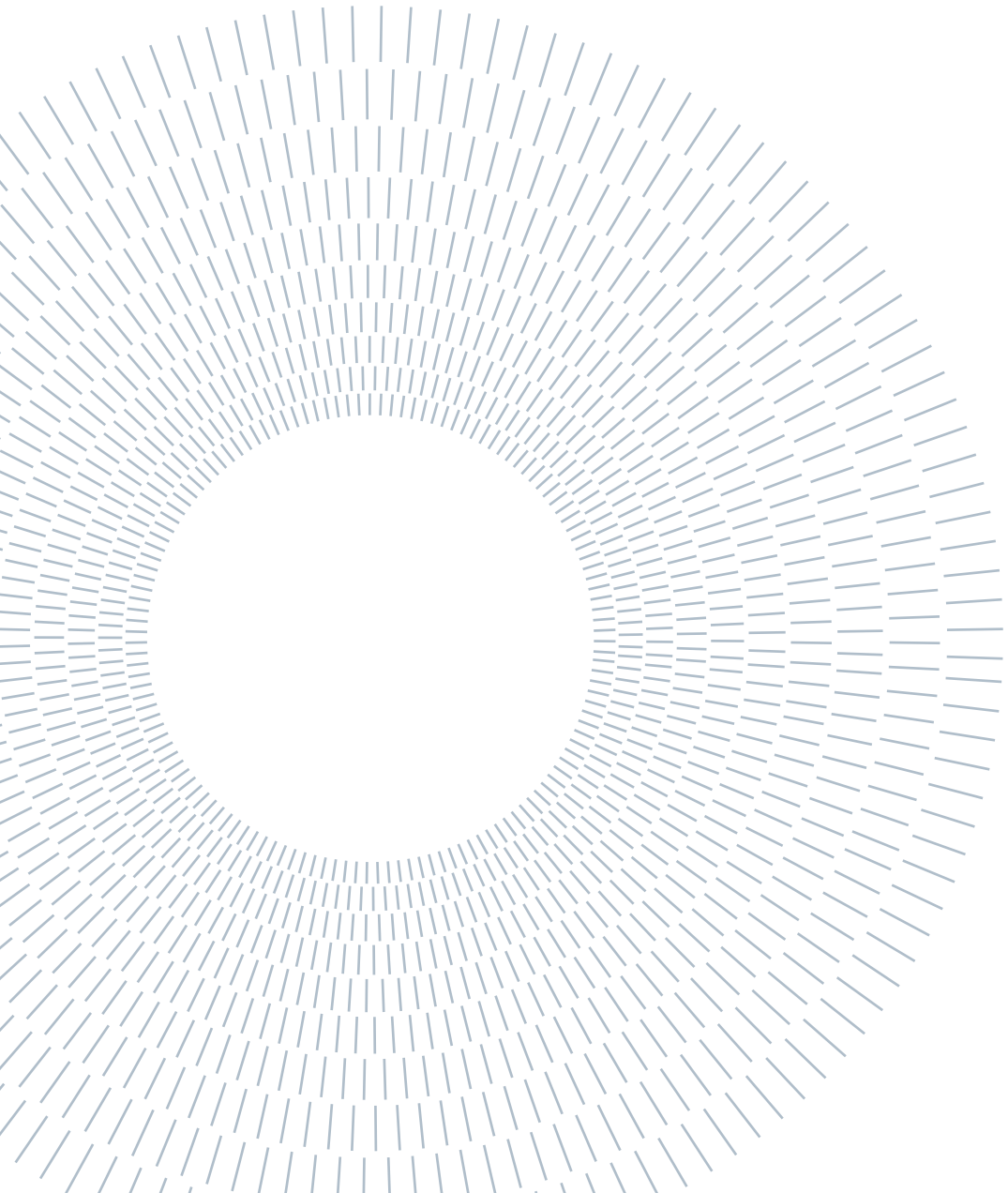
- **R25**: S&C allows candidates to respond to the received forms
- **R26**: S&C allows COs to visualize the responses of the candidates who have replied to the forms



5 | Implementation, Integration and Test Plan



6 | Effort Spent



7 References

- The names of *SpongeBob SquarePants* characters referenced in this document are the intellectual property of *Nickelodeon and Viacom International Inc.* We do not claim any ownership of the copyrighted material. The use of these names is intended solely for purposes such as commentary, criticism, analysis, or education, and falls under the "fair use" provisions outlined in Section 107 of the Copyright Act of 1976 (Articolo 70 della Legge sul Diritto d'Autore italiana (Legge n. 633/1941)). This use is non-commercial and transformative in nature, with no intention of infringing upon the copyright holders' rights.
- Lecture Slides of the course "Software Engineering 2", AA 2024/2025, by professor E. Di Nitto (Politecnico di Milano).
- We used Draw.io for the creation of the UML diagrams - `https://www.draw.io/`.
- We used GitHub for version control - `https://github.com/`.
- We used Visual Studio Code IDE for development of the LaTeX document - `https://www.visualstudio.com/`.
- We followed the Politecnico di Milano thesis template for the structure and style of the document - `https://www.overleaf.com/latex/templates/classical-format-thesis-scuola-di-ingegneria-industriale-e-dellinformazione-politecnico-di-milano/dkmvtndqkyxg`.