POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering II
Computer Science and Engineering

Design Document
Students&Companies

Authors:
Russolillo Simone
Visani Valeria Benedetta Cecilia
Wang Toni

Students ID:
11100725
10730247
10817365

Academic Year:
2024-25

Contents

1 Introduction

1.1. Purpose

The *Students&Companies (S&C)* platform bridges the gap between university students seeking internships and companies offering them. It simplifies the process of matching students with internship opportunities based on their skills, experiences, and preferences, as well as companies' requirements and offered benefits.

The software involves three main actors: **students**, **companies**, and **universities**.

- **Students** use the platform to search and apply for internships, submit their CVs, and receive recommendations tailored to their profiles.
- **Companies** advertise internships, specify requirements, and manage the selection process for suitable candidates.
- **Universities** monitor the execution of internships and handle complaints or issues that may arise.

S&C features a **recommendation system** that matches students and internships using mechanisms ranging from keyword-based searches to advanced statistical analyses. The platform also facilitates communication, supports the selection process, and tracks internship progress to ensure transparency for all involved parties.

1.2. Scope

The Students&Companies (S&C) platform is a web application designed to facilitate communication and matchmaking between university students seeking internships and companies offering them. The platform simplifies and automates the process by enabling students to explore and apply for internships, while also allowing companies to advertise their openings and identify suitable candidates. Additionally, a sophisticated recommendation system enhances the user experience by automatically suggesting relevant matches to both students and companies based on their preferences and requirements.

This document aims to outline the key architectural decisions behind the design and implementation of the S&C platform. Given the diverse user base, which includes students, companies, and universities, and the need for simultaneous interaction among these parties, a web application was chosen as the foundation. Its accessibility and ease of use ensure a seamless experience for users across various locations and devices.

The complexity of the platform, along with the distinct functionalities it provides—such as recommendations, selection processes, and feedback collection—led to the choice of a microservices architecture. This architectural style was selected due to its ability to offer scalability, flexibility, resilience, and modularity. Each microservice operates independently, allowing for targeted scaling based on demand, individual updates and deployments, and clear separation of responsibilities. The result is a system that is both maintainable and adaptable to evolving requirements.

From a deployment perspective, the system adopts a three-tier architecture. The user client layer represents the web and mobile interfaces used by students, companies, and universities. The server layer hosts all microservices, which manage the business logic and application functionality. Finally, the shared database layer ensures consistency in data storage, maintaining information about users, internships, recommendations, and feedback.

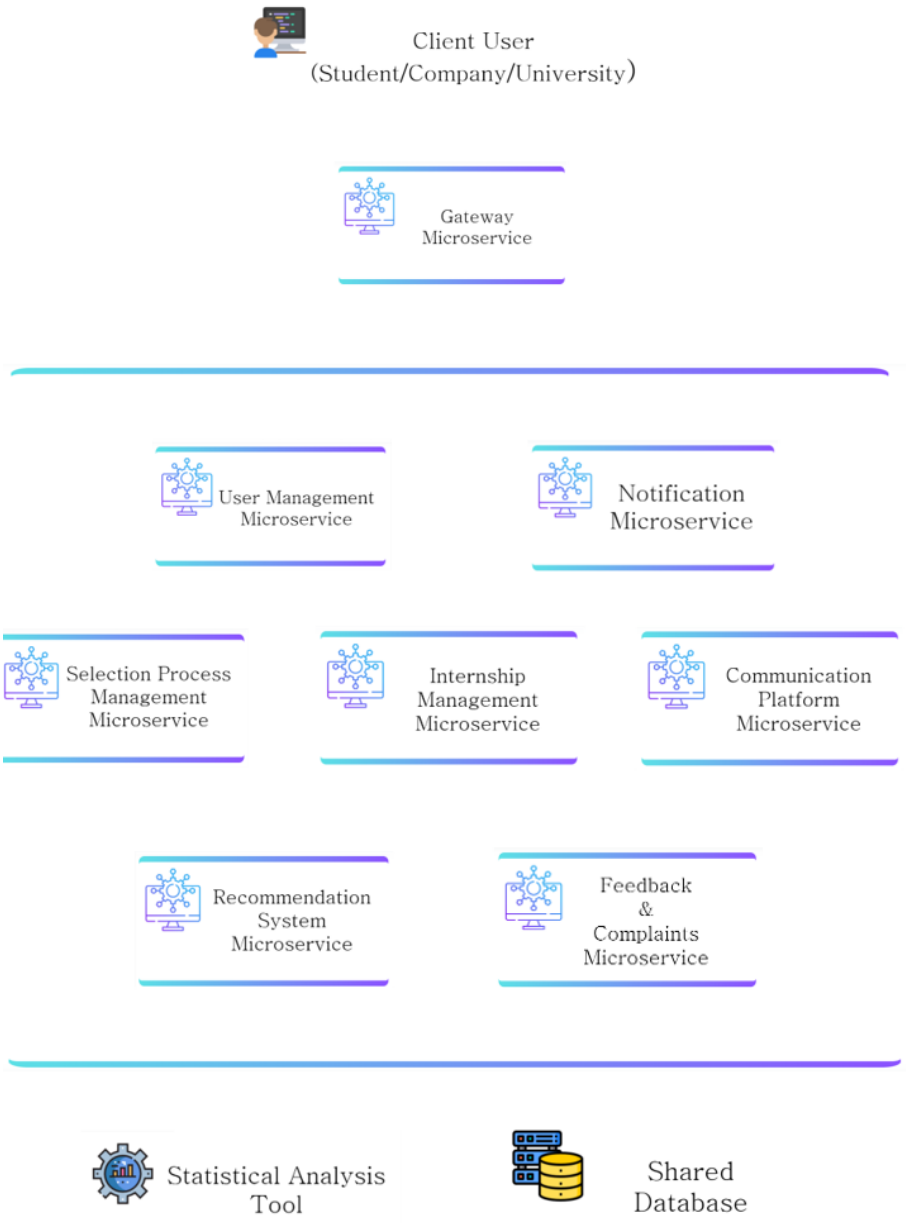
To manage interactions between microservices, the platform uses a combination of communication patterns based on specific functional needs. For asynchronous interactions, an event-driven communication model is employed. In this approach, some microservices act as event publishers while others function as consumers. For example, the Notification Microservice processes events related to complaints, messages, and new recommendations, while other services publish events to reflect changes in system states, such as the pubblication of a new message on the communication platform or the acceptance of a student for an interview.

For functionalities that require immediate interactions, synchronous communication mechanisms are used. This includes scenarios such as retrieving a list of internships or submitting CVs, where sequential processing and immediate feedback are necessary. The combination of event-driven and synchronous communication ensures that the system remains both responsive and straightforward to use, catering to the dynamic needs of its users.

Various architectural patterns are utilized in the design to ensure efficient communica-

tion between microservices. The Student&Company microservices expose RESTful APIs, enabling them to handle external requests and return data or perform computations as needed. Internally, an event-driven architecture is implemented to enhance performance and minimize coupling among microservices. Additionally, the Model-View-Controller (MVC) pattern is adopted to separate responsibilities, ensuring a clear distinction between managing incoming requests and rendering graphical user interfaces.

All these architectural choices are just mentioned here to provide an overview of the system; they will be better explained and unpacked down the line of this document. The following image shows the major components of the Students&Companies system.



1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

A brief list of the most meaningful and relevant terms and synonyms used in this document is reported here, in order to make reading process smoother and clearer:

Term	Definition
Internship, Placement, Work-Experience	A temporary work opportunity offered by a company, designed for students to gain practical experience in a professional environment while applying their academic knowledge.
CV, Resume	A document created by a student containing their personal information, skills, educational background, and work experience, used to apply for internships or jobs.
Recommendation System, Suggestion System	A feature of the platform that identifies and matches suitable internships for students or suitable candidates for companies based on their profiles, preferences, and requirements.
Student Profile	A digital representation of a student within the system, containing personal details, uploaded CVs, skills.
Company Profile	A digital representation of a company within the system, containing details about the company, uploaded projects or internships.
Recommendation Process	The sequence of steps executed by the system to align the skills and preferences of students with the requirements of available internships offered by companies.
Feedback, Suggestions	Information collected from students and companies during the selection process and the internship to refine the matching system and improve user satisfaction.

Term	Description
Communication Space, Chat Feature, Messaging System	A feature in the platform that allows students, companies, and universities to interact and share important updates or resolve concerns.
Selection Process	A phase in which companies evaluate student applications, conduct interviews, and finalize the selection of candidates for internships.
Interview Setup, Interview Management	The process supported by the system to schedule, conduct, and manage interviews between companies and students.
Monitoring by University	The process where the university oversees the activities and outcomes of student internships and intervenes if necessary.
Complaint Resolution	The process of identifying and addressing issues raised by students or companies during or after the internship period.
Submission Deadline, Application Deadline	The last date for students to submit applications for an internship or for companies to post available projects on the platform.
Notification System, Alert System	A functionality in the platform that keeps users informed about new opportunities, deadlines, or important events.
Platform, System, Application	All synonyms for the software platform being developed to manage the interactions and processes related to internships.
Statistical Analysis	The process by which the system evaluates collected feedback and interactions to improve its recommendation algorithms and user experience.
Architectural Style	A way of designing a system that defines general principles and patterns for how different parts should interact and be organized.

Term	Description
Structure/View	A representation of a system that highlights specific aspects, such as how components are connected, how they interact, or how they are distributed across different locations.
Component	An independent part of a system that performs a specific function and can often be reused in different systems or contexts.
API	A set of rules and tools that allow different software programs to communicate and share information or functionality with each other.
RESTful API	A type of API that uses standard web protocols like HTTP to let systems access and manipulate resources, following specific guidelines for simplicity and scalability.
Web Application	A program accessed through a web browser that combines client-side and server-side resources to provide interactive features and services.
Microservice	A design approach where a system is divided into small, self-contained services that handle specific tasks and can be developed, deployed, and updated independently.
Three-tier architecture	A system divided into three parts: one for displaying information to users (presentation), one for processing logic (application), and one for managing and storing data (database).
Event-Driven Architecture	A system where actions are triggered by events, such as a notification being sent when a user performs a specific action.

Term	Description
Synchronous communication among microservices	A communication method where one service sends a request to another and waits for the response before continuing.
Asynchronous communication among microservices	A communication method where one service sends a request and does not wait for a response, allowing the system to handle tasks more flexibly.

1.3.2. Acronyms

A list of acronyms used throughout the document for simplicity and readability:

- RASD - Requirements Analysis and Specification Document
- DD - Design Document
- S&C - Students & Companies
- API - Application Programming Interface
- UI - User Interface
- UML - Unified Modeling Language
- DB - Database
- DBMS - Database Management System

1.4. Reference Documents

Here's a list of reference documents that have been used in order to shape the Design Document of the *Students&Companies* system. In the following, all external sources of information that have contributed to the design of this document are mentioned.

1. Stakeholders' specification provided by the R&DD assignment for the Software Engineering II course at Politecnico Di Milano for the year 2024/2025.
2. "29148-2018, ISO/IEC/IEEE International Standard, Systems and software engineering, Life cycle processes, Requirements engineering", by IEEE, 2018.
   Link: `https://ieeexplore.ieee.org/document/8559686`
3. UML specifications, version 2.5.1.
   Link: `https://www.omg.org/spec/UML/2.5.1/About-UML`

1.5. Document Structure

The Design Document for the Student&Company project are organized into five primary parts: the first is the introduction, while the remaining four each focus on a specific aspect of the system's overall design, which will help facilitate the development and final implementation of the product.

The **Introduction** serves to provide a concise overview of the project, detailing the objectives and goals that are to be accomplished through its development, as outlined earlier in the RASD.

Moving on, Section 2, titled **Architectural Design**, is the most critical design-related section and aims to present the software architecture of Student&Company through various views and structural representations. This section is divided into multiple parts:

The first part, Overview, delivers a high-level summary of the core components of the system and how they interact with each other, explained in an informal notation that makes the structure more accessible.

The second part, Component View, presents the first of the architectural structures, the Component & Connector structure, which is crucial for demonstrating the system's components from a dynamic perspective and the way in which they collaborate to meet the final objectives; it largely uses UML component diagrams to convey these interactions.

The Deployment View, the third part, focuses on the deployment structure of Student&Company, illustrating how the software components correspond to the physical hardware that will run the application. The mapping between the software and hardware is illustrated with UML deployment diagrams, which are extremely helpful in visualizing this relationship.

Then, the Runtime View follows, employing sequence diagrams to describe the flow of events and interactions within the system's components, ensuring consistency with the previously discussed sections.

The Component Interfaces section comes next, where a detailed specification of the important methods and functions exposed by each interface of the system's components is provided, making sure to cover all relevant aspects.

The final part of Section 2 discusses the Selected Architectural Styles and Patterns, offering a review of the primary architectural styles and patterns, followed by a detailed explanation of why they were selected for this particular project.

Section 3, on **User Interface Design**, shifts focus to the design of user interfaces (UI), offering guidelines for UI designers on how the final application should appear, including color schemes, the placement of key UI elements, and also the logical role that these interfaces play in the development process, clarifying what functionalities they provide to the user.

Following this, Section 4, which covers **Requirement Traceability**, provides a matrix that clearly shows how the requirements for Student&Company, which were previously

drawn up, map onto the components discussed in earlier sections of the document, ensuring that all requirements are adequately addressed by the system.

Finally, Section 5, **Implementation, Integration, and Test Plan**, explains the strategy for implementing the system, detailing the order in which the components will be developed, the approach for integrating new sub-components into the application as it progresses, and the testing strategy to ensure that all components work seamlessly together within the system.

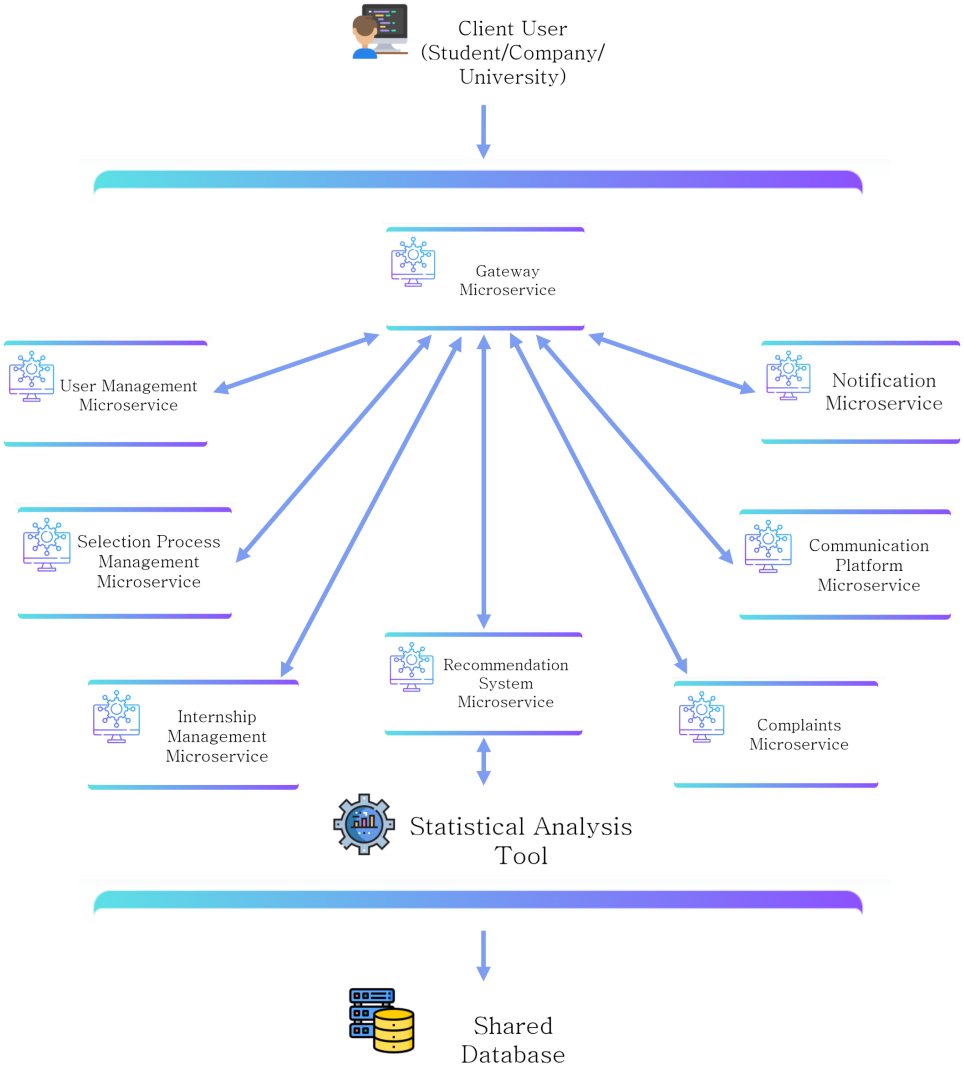
2 | Architectural Design

2.1. Overview

2.1.1. High Level View

This section provides an overview of the architectural components that make up the Students&Companies system and highlights their key interactions.

The Students&Companies system adopts a microservices architectural approach. This style of architecture involves designing the application as a collection of independent services, each responsible for a specific functionality. These services collaborate with one another to fulfill the system's overall objectives. To enhance clarity, a high-level representation of the system, including its microservices, is shown in the following image:



From a high-level viewpoint, the elements mentioned earlier are illustrated in the figure below. At the top, the client is represented by the first icon, while the Students&Companies system is depicted as a collection of microservices. At the bottom of the figure, the data layer is represented with a database icon.

The Students&Companies system comprises several microservices, each fulfilling specific responsibilities. Their details are as follows:

- **Gateway Microservice:** This microservice manages two key functionalities. Firstly, it handles the authentication process for users, such as students, companies, and universities. All incoming user requests are processed by the Gateway, which dispatches them to the relevant microservice for handling. Similarly, all system responses are routed back through the Gateway to reach the users. To ensure scalability and performance, this microservice can be replicated on the server side.
- **User Management Microservice:** Responsible for managing user data and profiles, this microservice oversees the profiles of students, companies, and universities.

It provides user profile pages and ensures seamless access to personal data whenever required.

- **Recommendation System Microservice:** This microservice is tasked with analyzing student CVs, company project details, and internship data to recommend suitable matches between students and companies. It uses algorithms and data from the shared database to optimize the pairing process.
- **Internship Management Microservice:** This microservice handles all aspects of internships. It supports the creation and updating of internship listings, manages student applications, and provides detailed views of ongoing and completed internships.
- **Selection Process Management Microservice:** Dedicated to assisting in the selection and recruitment process, this microservice provides features such as managing interview schedules, handling offer confirmations, and finalizing internship agreements.
- **Communication Platform Microservice:** This microservice facilitates communication between students, companies, and universities. It provides a dedicated platform for exchanging messages, addressing inquiries, and resolving internship-related issues.
- **Feedback&Complaint Microservice:** Focused on gathering and managing feedback, this microservice enables users to submit complaints or reviews about internships. It provides tools for universities to monitor and address reported issues effectively.
- **Notification Microservice:** Responsible for managing notifications throughout the system, this microservice listens to events (event-driven architecture) and sends timely alerts to users, such as internship deadlines, application updates, or system announcements.
- **Statistical Analysis Tool:** This tool aids in analyzing historical data and user feedback to refine the recommendation algorithms and improve the overall user experience.

As for the interactions between these microservices, they will be elaborated upon in greater detail in the subsequent sections of this document. For now, it is important to highlight that all communications between the client and the Students&Companies platform are routed through the Gateway Microservice. Additionally, the various user interfaces provided by the platform are supplied by different microservices, each handling the data and functionalities specific to their domain. However, the delivery of these interfaces to the client is always mediated by the gateway.

All microservices in the system expose **RESTful APIs**, which serve as the primary access points to their data and functionalities. When necessary, microservices interact with each other through these APIs, and the specifics of these interactions will be outlined in subsequent views.

Moreover, every microservice has access to a **shared database** via the interfaces provided by the DBMS. While all microservices share this common data space, each one is responsible for distinct operations and computations based on the specific functionalities it offers.

The system's internal interactions are also built upon an **event-driven architecture**. In this pattern, microservices can asynchronously publish events to an Event Bus or Event Queue. Other components within the system can consume these events and perform actions in response. Further details regarding this architecture will be provided in the following sections of the Design Document.

2.2. Component View

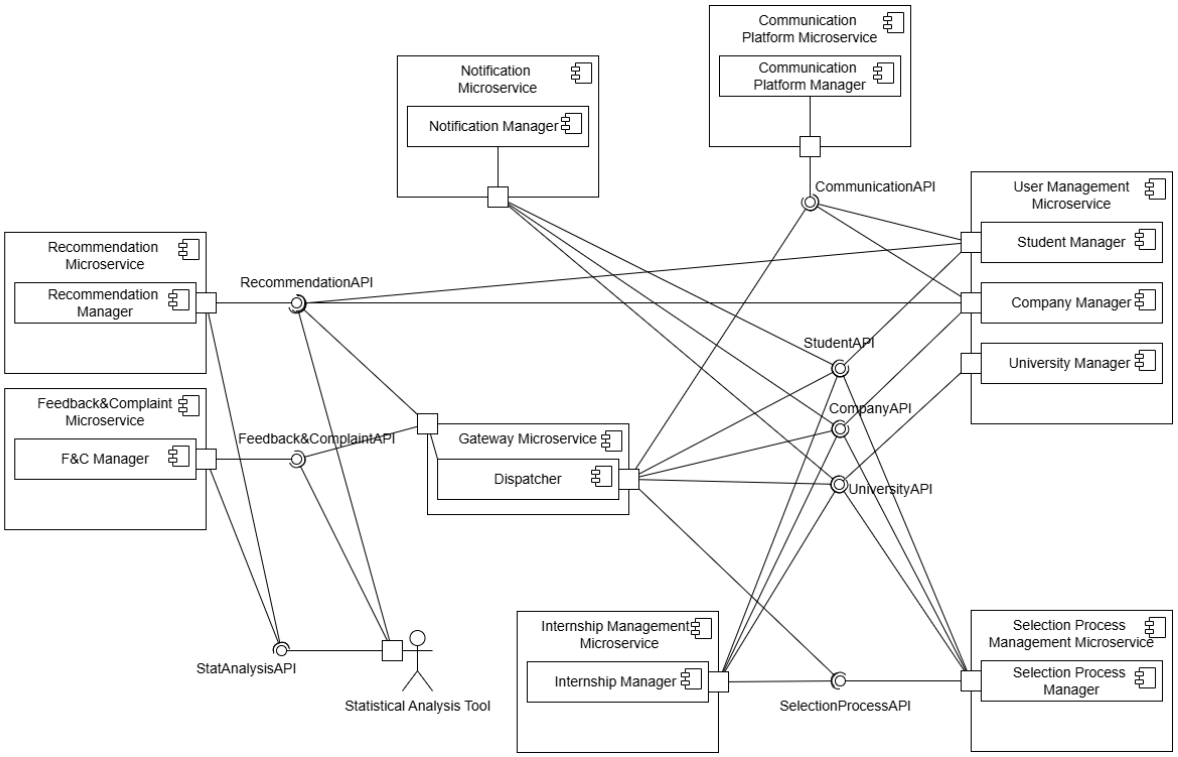
This section aims to provide an overview of the various components that constitute the Students&Companies system. UML Component diagrams are utilized to represent the logical software elements that work collaboratively to achieve the objectives defined for the development of the system.

Given that the structural components of the Students&Companies platform cannot be effectively represented in a single diagram, multiple diagrams are presented. This approach is adopted to ensure clarity and avoid overcrowding the visualizations. The diagrams have been organized based on the principle of grouping components that frequently interact with one another.

It is worth noting that, despite the division into separate diagrams for explanation purposes, the system functions as a cohesive whole. The components depicted in these diagrams belong to the same system and collectively contribute to the overall operation and functionality of the platform.

2.2.1. RESTful APIs Component Diagram

One of the key features highlighted in the introductory section of the Students&Companies project is that microservices provide RESTful APIs to handle incoming commands, share data, and interact with each other. This view focuses on outlining, from a broad perspective, how these microservices leverage their APIs to operate efficiently.



Let's analyze the key concepts conveyed by the diagram. This UML component diagram is used to illustrate the RESTful APIs provided by each microservice within the Students&Companies system. These APIs perform specific computations on data related to the functionality offered by the respective microservice. For example, the Internship Management Microservice exposes an API that processes internship data within the shared database and returns relevant internship information.

The Gateway Microservice serves as the entry point for all user requests. The Dispatcher component inside the Gateway is responsible for routing user requests to the appropriate microservices that are tasked with handling them. This explains why the Dispatcher "uses" all the APIs exposed by the other microservices.

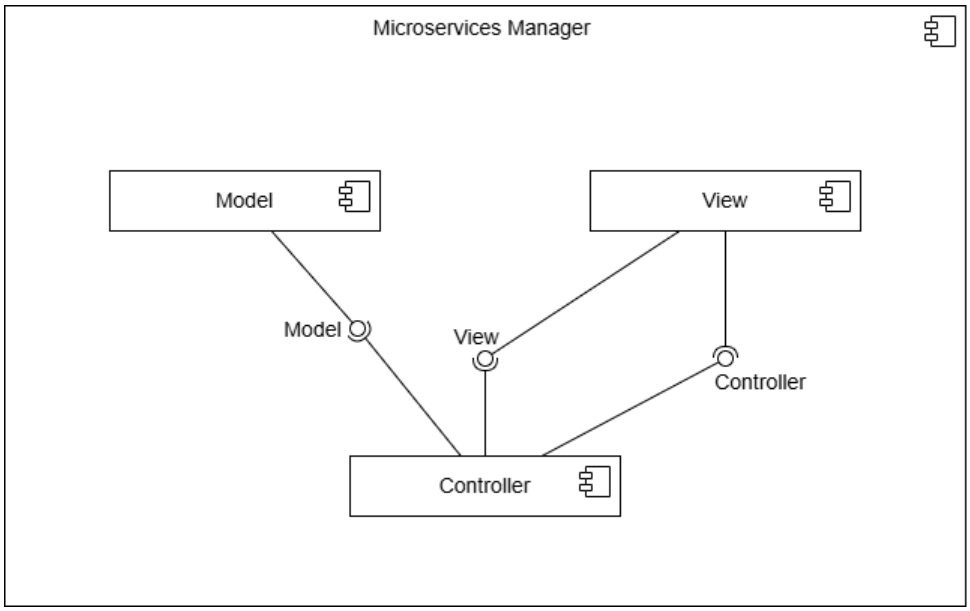
In certain cases, the microservices in the Students&Companies system may also interact with one another through their RESTful APIs to perform joint computations In this diagram, these interactions are shown. For instance:

- The Notification Microservice needs to query the Student Manager, University Manager, Company Manager inside the User Management Microservice whenever it needs to deliver the notification to a specific set of users.
- The Notification Microservice must query the User Management Microservice when the list of all students in the Students&Companies system is required (for example, to notify all students about a new internship opportunity available on the platform).
- The Communication Platform Microservice uses the Notification API to deliver alerts such as internship status updates, interview invitations, or feedback to the appropriate users.

- The Recommendation System Microservice must leverage the Statistical Analysis Tool to refine its matching algorithms and provide better internship recommendations based on student profiles and internship data.
- The User Management Microservice uses the Internship Management API to update student profiles once they have been matched with an internship, ensuring that the students' records reflect their current opportunities.
- The Internship Management Microservice must interact with the Selection Process Management Microservice to facilitate the selection process, including interview scheduling and the finalization of student placements.
- The Feedback&Complaint Microservice interacts with the Statistical Analysis Tool to provide useful information for recommendations.

This diagram must be interpreted considering the event-driven paradigm adopted in the Students&Companies project. Some interactions and event flows within the system occur asynchronously, following an event-driven approach that is further explained in subsequent views. This is why certain interactions, which may appear absent here, are actually managed asynchronously to improve the overall system performance.

Another crucial aspect to highlight regarding this diagram is the "Manager" components within the various microservices. These components internally implement a Model-View-Controller pattern to deliver their services. The following representation illustrates a component diagram that focuses solely on the internals of the "Manager" component for a generic microservice:



The Model component is responsible for organizing and managing data. Each microservice in the Students&Companies system is designed to handle a specific portion of the data domain. For example, the Internship Management Microservice focuses solely on managing and providing information about internships. Consequently, each Model

component within the Manager components ensures data consistency with the DBMS, accessing it when necessary, and performing related operations.

The View component offers the user interface. Since different microservices handle various types of information within the Students&Companies ecosystem, multiple user interfaces are distributed across the microservices and managed by their respective View components.

The Controller acts as the "bridge" between the Model and the View. This component is responsible for:

- Receiving requests from the View component when the user interacts with the user interface.
- Performing the appropriate computation based on those requests.
- Manipulating data through the Model, passing user requests to the data layer.
- Updating the View if the computation results in changes to the user interface.

Throughout this document, the term "Manager" component will be used in a more general sense to encompass the behavior and functionality of this internal architecture.

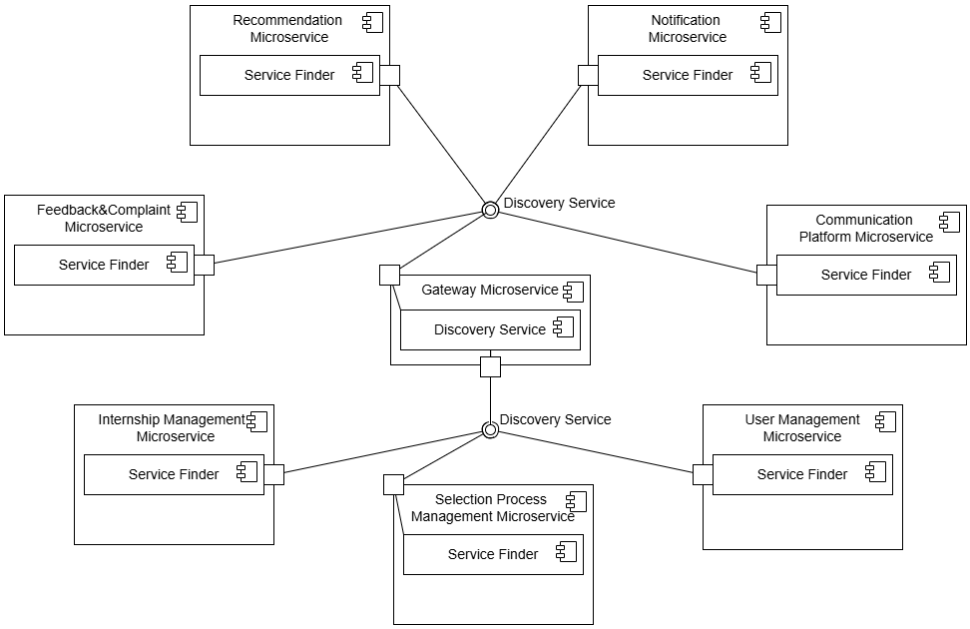
2.2.2. Service Discovery Component Diagram

This simple component diagram illustrates an important service provided by the Gateway Microservice to allow all microservices to locate and collaborate with each other. This service is known as "Service Discovery" and involves maintaining a register (inside the Gateway Microservice) of active microservices.

In this way:

- A newly launched microservice can register itself to be discovered by other microservices.
- All microservices can contact other active microservices in the system to send requests.

Here's the diagram:

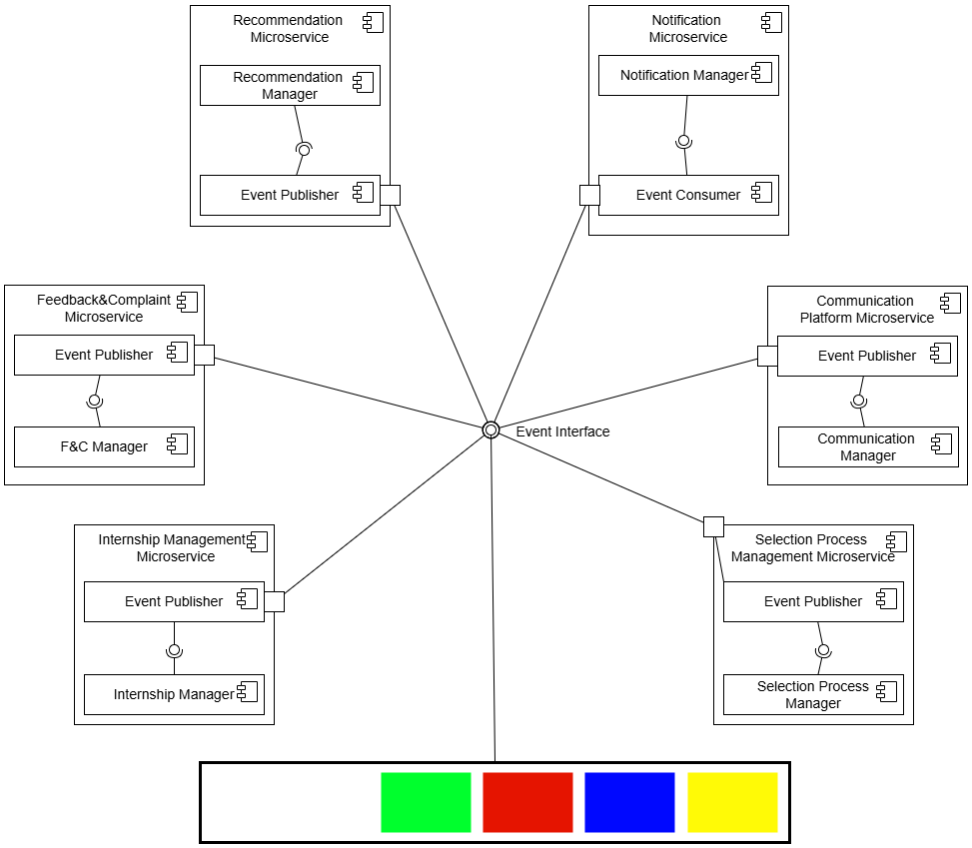


The interpretation is straightforward. The Gateway Microservice contains a component called the Discovery Service, which implements the logic to provide the discovery functionality. The Discovery Service component exposes an API (`DiscoveryServiceAPI`), which can be used by all other microservices to locate each other.

2.2.3. Event-Driven Pattern Components

This section illustrates all the components required to implement asynchronous, event-driven communication between microservices in the system. This design decision has been made to optimize the system, particularly for interactions between microservices that would otherwise require computationally expensive synchronous communication.

The following diagram shows the major components that enable the event-driven architecture:



The most important element is the Bus component, which can be thought of as a queue of messages or events. Microservices in the Students&Companies system can either produce events (i.e., push events onto the queue) or consume events (i.e., read events from the queue) and take actions accordingly.

It is important to note that this representation is intentionally general and not tied to any specific software product, framework, or implementation. The concrete implementation details of this design are left to the product's implementation.

From the diagram, it is clear that only certain microservices are shown, as they are the ones utilizing this mechanism to communicate and interact. Within these microservices, there is an Event Publisher component if the microservice needs to publish events to the Bus, and an Event Consumer component if the microservice needs to read events from the Bus.

For example, the Notification Microservice listens for events to generate notifications for users based on the events in the system.

The Feedback&Complaint Microservice publishes events whenever a new feedback or complaint is submitted.

The Communication Platform Microservice publishes events when new messages or communications are exchanged.

The Recommendation Microservice publishes events to notify about new recommenda-

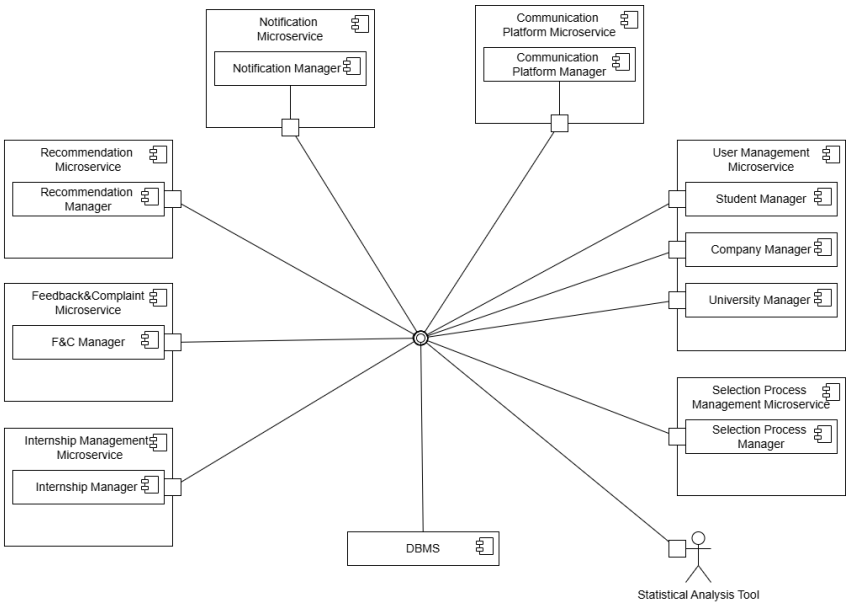
tions made between students and internships.

The Selection Process Microservice publishes events related to the progression or completion of the selection process.

The Internship Management Microservice publishes events whenever an internship is created, updated, or closed.

2.2.4. Data Layer Access Component Diagram

This diagram provides a convenient view of how access is performed by the various microservices on the shared data layer (shared DBMS).



The DBMS exposes an interface that is used by all microservices to interact with the persistent data stored within it.

Looking at each microservice individually, it can be observed that they all contain a "Manager" component, which is responsible for manipulating and accessing the data.

Each microservice in the Students&Companies system is responsible for a specific section of the data domain. For example, the Internship Management Microservice only works with the data related to internships stored in the shared DBMS and provides other microservices with the relevant information about internships whenever needed.

2.2.5. User Interfaces Component Diagram

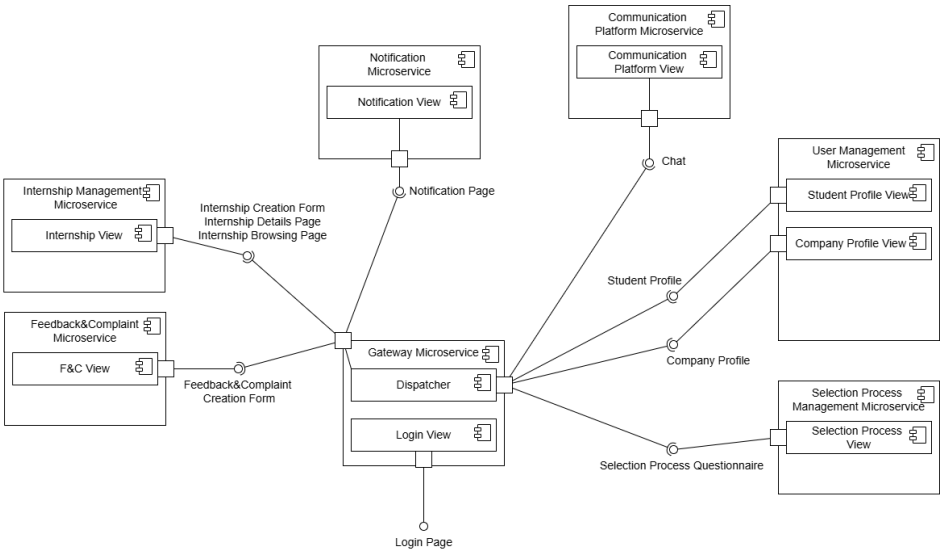
This diagram provides a convenient view of the system in terms of user interfaces. The design choices made regarding the user interfaces in the Students&Companies system can be summarized as follows:

- The different user interfaces are not gathered in a single component; instead, they

are distributed across various microservices. This decision arises from the fact that different UIs present different views of the application's data. Therefore, some microservices (handling specific sections of the data domain) may be better suited to provide a particular user interface than others.

- Every time a user interface is presented to the user, the Gateway Microservice is responsible for transmitting the data, acting as an intermediary between the user and the internal microservices. As a result, all UI content passes through the Gateway before reaching the client user.

This diagram attempts to illustrate both points by representing a View component inside each microservice that is responsible for building and offering specific user interfaces when required. The Gateway Microservice intercepts the user's requests, and when a new UI is needed, it contacts the appropriate microservice to supply it. It should be noted that this UML diagram is non-standard, in the sense that the interfaces exposed by the components in this view are UIs rather than sets of methods. However, it is intended as a helpful illustration to show part of the MVC pattern (with View components inside the microservices) and the distribution of UIs across the system.



2.3. Deployment View

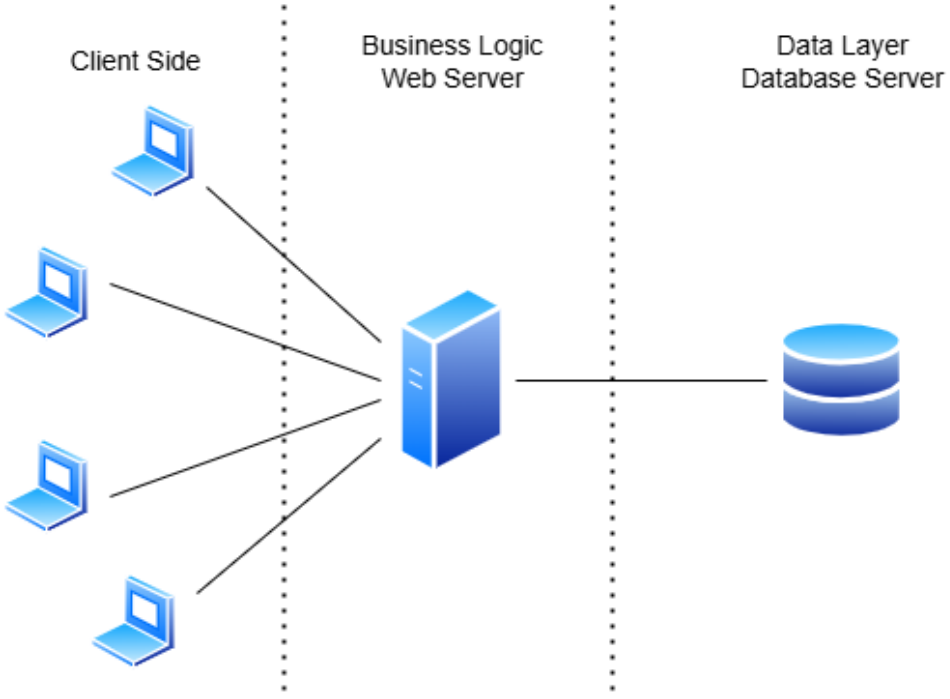
The deployment view focuses on the execution aspects of the Students&Companies system, detailing how and where the system components are run. It looks at how services are distributed, how they are orchestrated, and the infrastructure supporting the application. Essentially, this section provides a connection between the software components that drive the system's functionality and the physical hardware needed for their operation.

The goal of this section is to offer a clear understanding of the architectural decisions that contribute to efficient use of resources, scalability to accommodate different levels of demand, and flexibility to adapt to changing operational conditions.

As in the previous sections, this view is represented through a series of diagrams that explore the deployment environment at various levels of detail. These diagrams are complementary and should be seen as a layered illustration of the same concept, starting from a broad overview and progressing to more intricate details.

2.3.1. High-Level Deployment View

This diagram illustrates the essential components required to present the deployment view of the Students&Companies system.



As depicted in the figure, the Students&Companies platform follows a three-tier architecture from a deployment standpoint. The illustration is divided into three distinct areas, representing the three layers: the client users (on the left), the server (in the center), and the data layer (on the right).

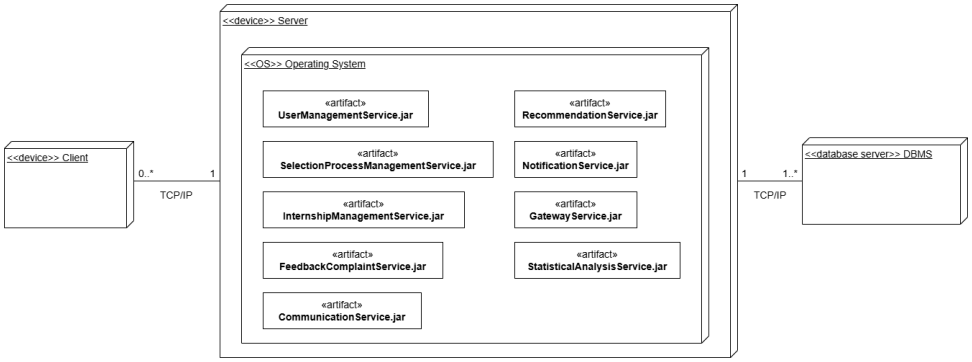
Users interact with the platform through devices with internet access, capable of sending and receiving requests to the server hosting Students&Companies. The entire system is deployed on a single remote server, where all microservices operate as separate programs and communicate with one another. The data layer is supported by a remote Database server that provides persistent data storage for the entire platform.

This deployment model offers several advantages. Primarily, it ensures a clear separation of concerns between the different layers, particularly between the application's business logic (hosted on the server) and the data layer (within the DBMS). Furthermore, it is a scalable architecture, which can be extended to handle varying workloads. For example, the data layer can be replicated across multiple servers to enhance performance

when accessing and manipulating data. Similarly, this approach can be applied to the business layer if necessary.

2.3.2. Detailed Deployment View

The following diagram provides additional details on how the software components that make up the Students&Companies application are mapped to hardware elements.



The primary components in this diagram remain consistent with the previous description. On the left side, we have the client-side, where users are equipped with devices that have internet connectivity to communicate with the server via the TCP/IP protocol. The multiplicity annotations on the connection between the client and the server indicate that multiple clients can interact with the server concurrently.

The server is represented as the central component in the diagram. The UML deployment diagram allows for a detailed view of the execution environment of the Students&Companies system by nesting various boxes. In this case, all the microservice artifacts are mapped to a single server instance. These microservices run as separate processes within the server.

On the right-hand side, we see the data layer, which consists of a database server that is connected to the central server via the internet (using TCP/IP).

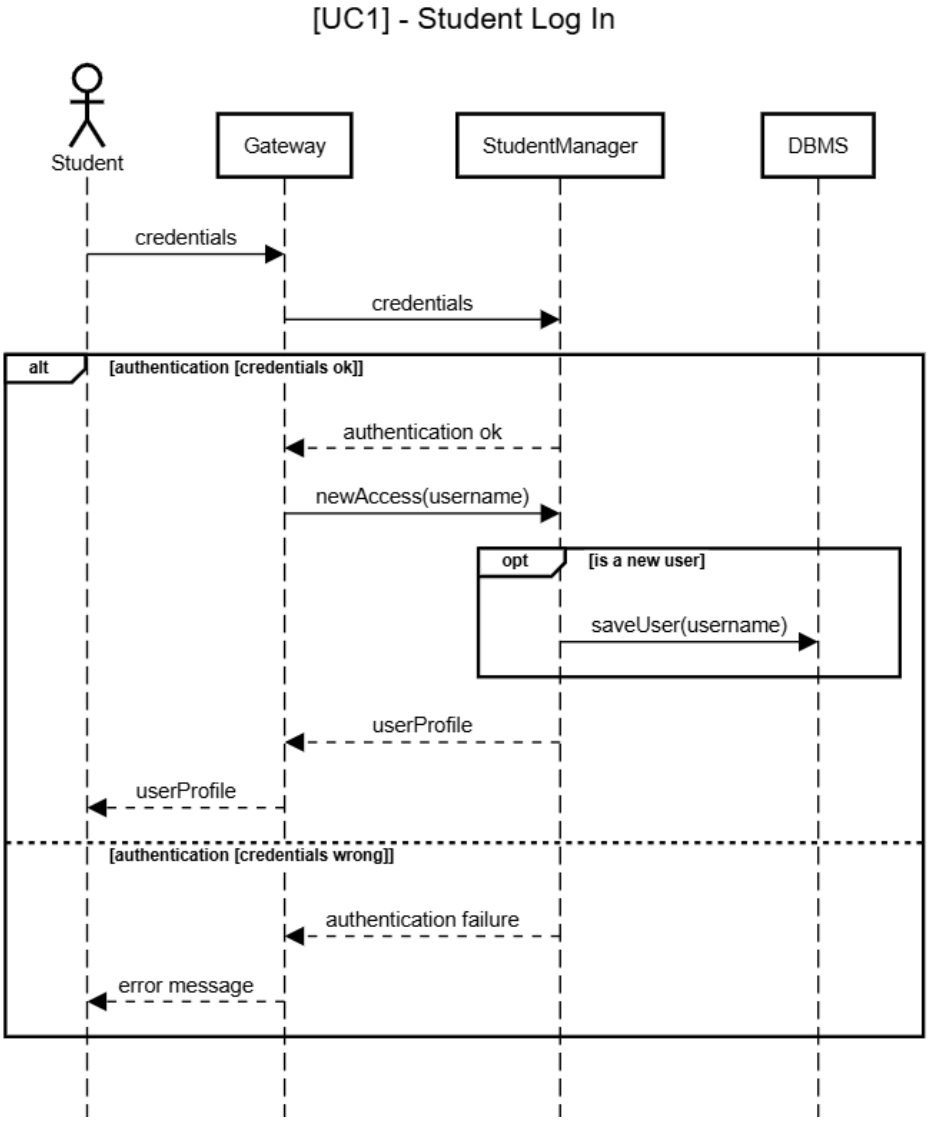
This design has been selected because it supports seamless scaling of the system. For instance, the database server can be replicated to handle multiple incoming requests from the server in parallel. If required, a more sophisticated distributed architecture can be implemented by distributing the microservice components across different network nodes, ensuring the system continues to operate smoothly with appropriate configuration. Furthermore, microservices can be replicated within the server to further enhance performance.

2.4. Run Time View

This section offers an in-depth look at the internal workings of the system and the interactions between its components during runtime. It focuses on the dynamic behavior

of the software, highlighting how control, data, and communication flow between various modules or components.

Student Log In



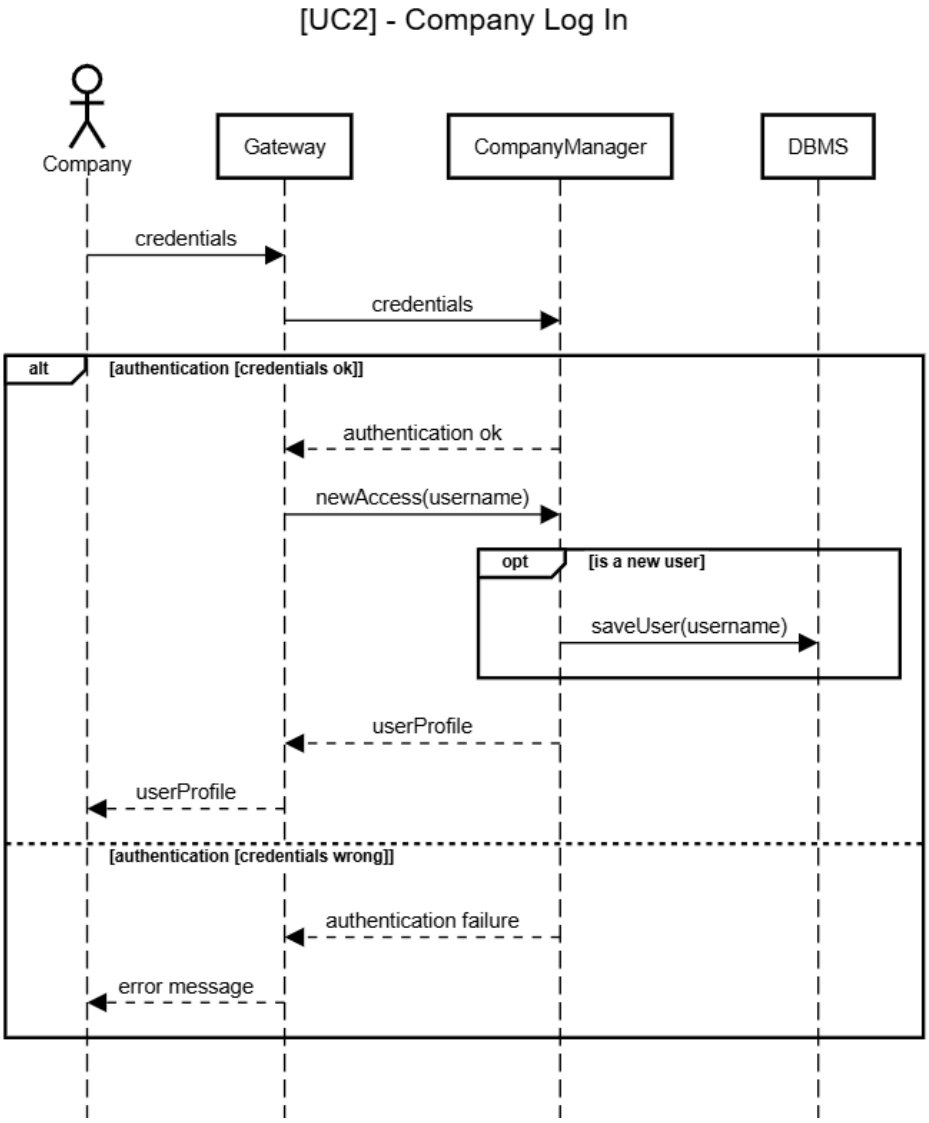
The Student Log In process begins when the Student submits their credentials (username and password) to the Gateway Microservice. The Gateway forwards these credentials to the Student Manager for authentication. If the credentials are valid, the Student Manager confirms the successful authentication by sending an "authentication ok" message to the Gateway Microservice. The Gateway then notifies the Student Manager of the user's access by transmitting the username.

If the Student is identified as a new user, the Student Manager requests the DBMS to store the student's information. Subsequently, the Student Manager retrieves the user's profile and sends it back to the Gateway Microservice, which in turn delivers the profile to the Student, finalizing the login process.

In cases where the credentials are invalid, the Student Manager notifies the Gateway Microservice of the authentication failure. The Gateway then generates and sends an error message to the Student, informing them of the unsuccessful login attempt.

This sequence ensures that authenticated students gain seamless access to their profiles, while unsuccessful attempts are met with clear feedback to guide corrective action.

Company Log In



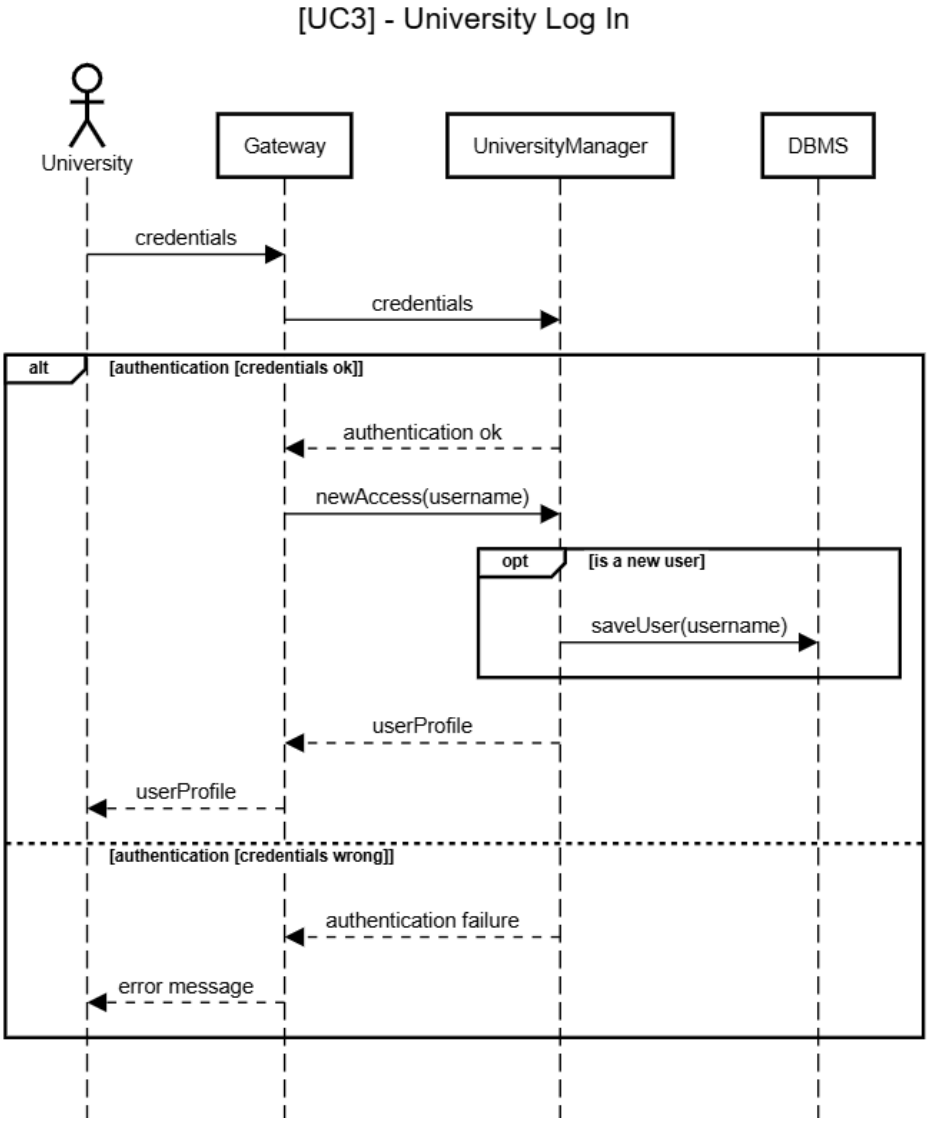
The Company Log In process begins when the Company provides their credentials (username and password) to the Gateway Microservice. The Gateway then forwards these credentials to the Company Manager for authentication. If the credentials are correct, the Company Manager confirms the successful authentication by sending an "authentication ok" message to the Gateway Microservice. In turn, the Gateway forwards the username to the Company Manager to check if the company is a new user.

If the Company is a new user, the Company Manager requests the DBMS to store the company's details. Once the company's information is stored, the Company Manager retrieves the user profile and sends it back to the Gateway. The Gateway then returns the user profile to the Company, completing the login process.

On the other hand, if the credentials are incorrect, the Company Manager sends an authentication failure message to the Gateway. The Gateway then generates and delivers an error message to the Company, indicating that the login attempt was unsuccessful.

This flow ensures that the Company can either access their profile on successful authentication or receive an error message if authentication fails.

University Log In



The University Log In process begins when the University provides their credentials (username and password) to the Gateway Microservice. The Gateway forwards these

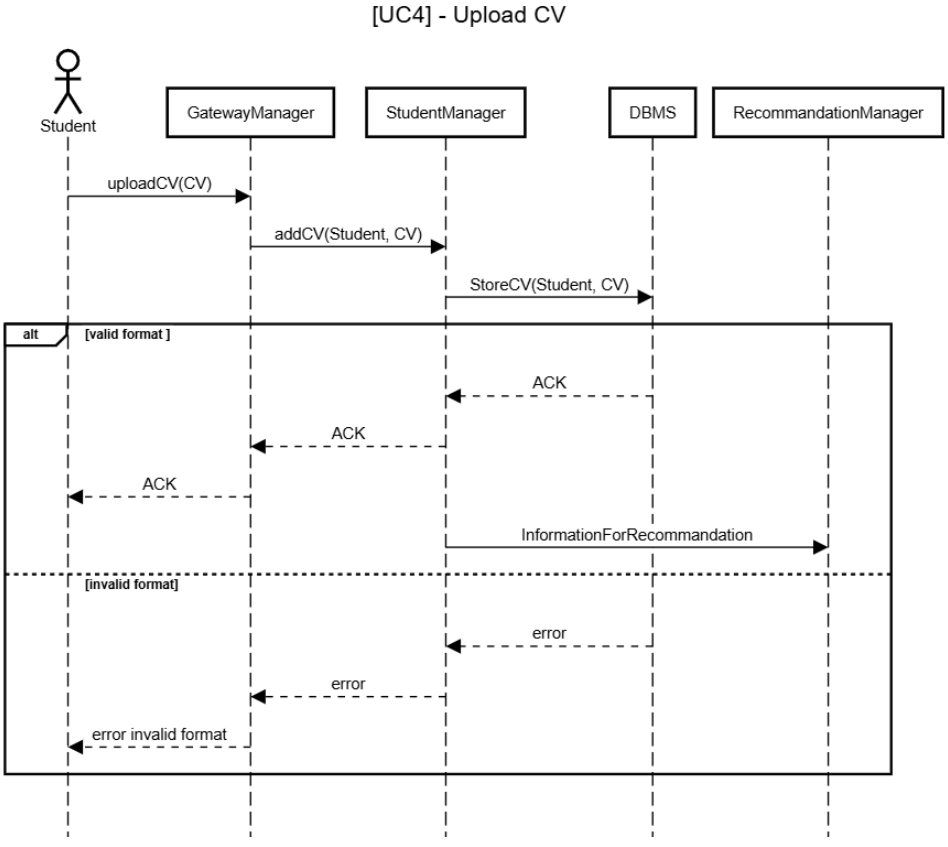
credentials to the University Manager for authentication. If the credentials are correct, the University Manager sends an "authentication ok" message back to the Gateway Microservice. The Gateway then sends the username to the University Manager to verify if the University is a new user.

If the University is a new user, the University Manager requests the DBMS to store the university's information. After storing the details, the University Manager retrieves the user profile and sends it to the Gateway Microservice. The Gateway then forwards the user profile to the University, completing the login process.

If the credentials are incorrect, the University Manager sends an authentication failure message to the Gateway. In turn, the Gateway generates and sends an error message to the University, notifying them of the failed login attempt.

This flow ensures that the University can either log in successfully and access their profile or receive an error message if authentication fails.

Upload CV



The Upload CV process begins when the Student sends their CV to the Gateway Manager through the uploadCV(CV) request. The Gateway Manager forwards this request, along with the CV and Student information, to the Student Manager. The Student Manager then sends a StoreCV(Student, CV) command to the DBMS to store the CV in the

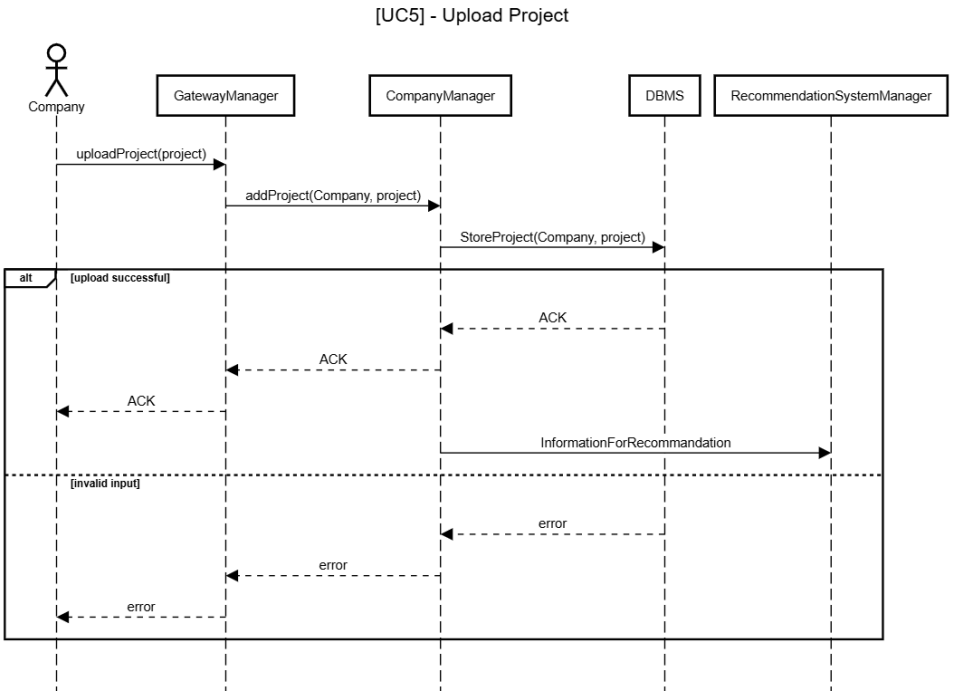
database. Once the DBMS successfully stores the CV, it sends an "ACK" message to the Student Manager to confirm the storage operation.

The Student Manager, upon receiving the confirmation from the DBMS, sends an "ACK" message back to the Gateway Manager. The Gateway Manager then relays the "ACK" confirmation to the Student, indicating the successful upload of the CV.

Additionally, the Student Manager sends the InformationForRecommendation data to the Recommendation Manager, so that the uploaded CV can be considered for any relevant recommendations for internships or other opportunities.

This process ensures that the CV is uploaded and stored securely while triggering the subsequent recommendation process for the student.

Upload Project



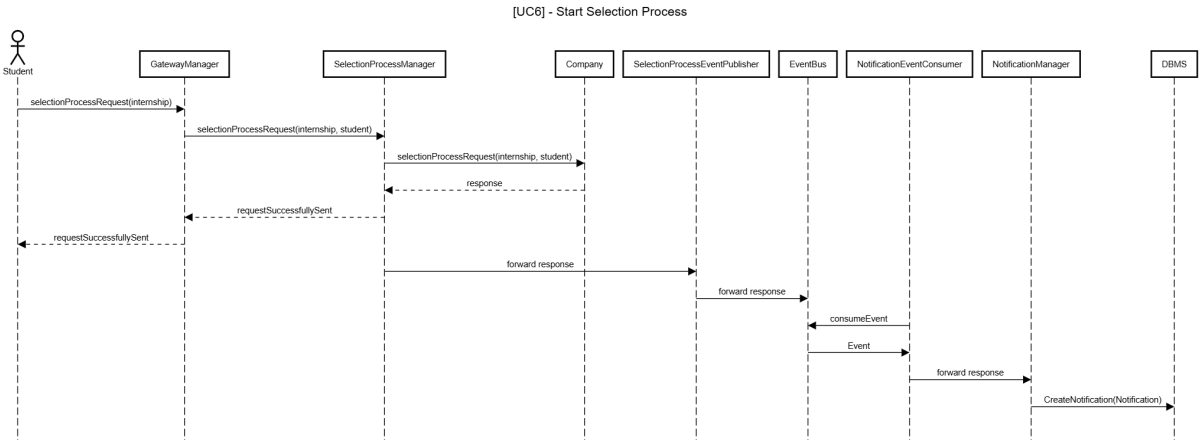
The Upload Project process begins when the Company sends the project details to the Gateway Manager through the uploadProject(project) request. The Gateway Manager forwards this request, along with the project and Company information, to the Company Manager. The Company Manager then sends a StoreProject(Company, project) command to the DBMS to store the project in the database. Once the DBMS successfully stores the project, it sends an "ACK" message back to the Company Manager to confirm the storage operation.

Upon receiving the confirmation from the DBMS, the Company Manager sends an "ACK" message to the Gateway Manager. The Gateway Manager then relays this "ACK" confirmation to the Company, indicating the successful upload of the project.

Additionally, the Company Manager sends the InformationForRecommendation data to the Recommendation Manager, so that the uploaded project can be considered for any relevant recommendations.

This process ensures that the project is uploaded and stored properly while initiating the recommendation process for the company.

Start Selection Process



The Start Selection Process initiates when the Student sends a selectionProcessRequest(internship) to the Gateway Manager, signaling their intent to begin the selection process for a particular internship. The Gateway Manager then forwards this request, including both the internship and student details, to the Selection Process Manager.

Upon receiving the request, the Selection Process Manager sends the selectionProcessRequest(internship, student) to the relevant Company offering the internship. The Company reviews the request and sends a response back to the Selection Process Manager.

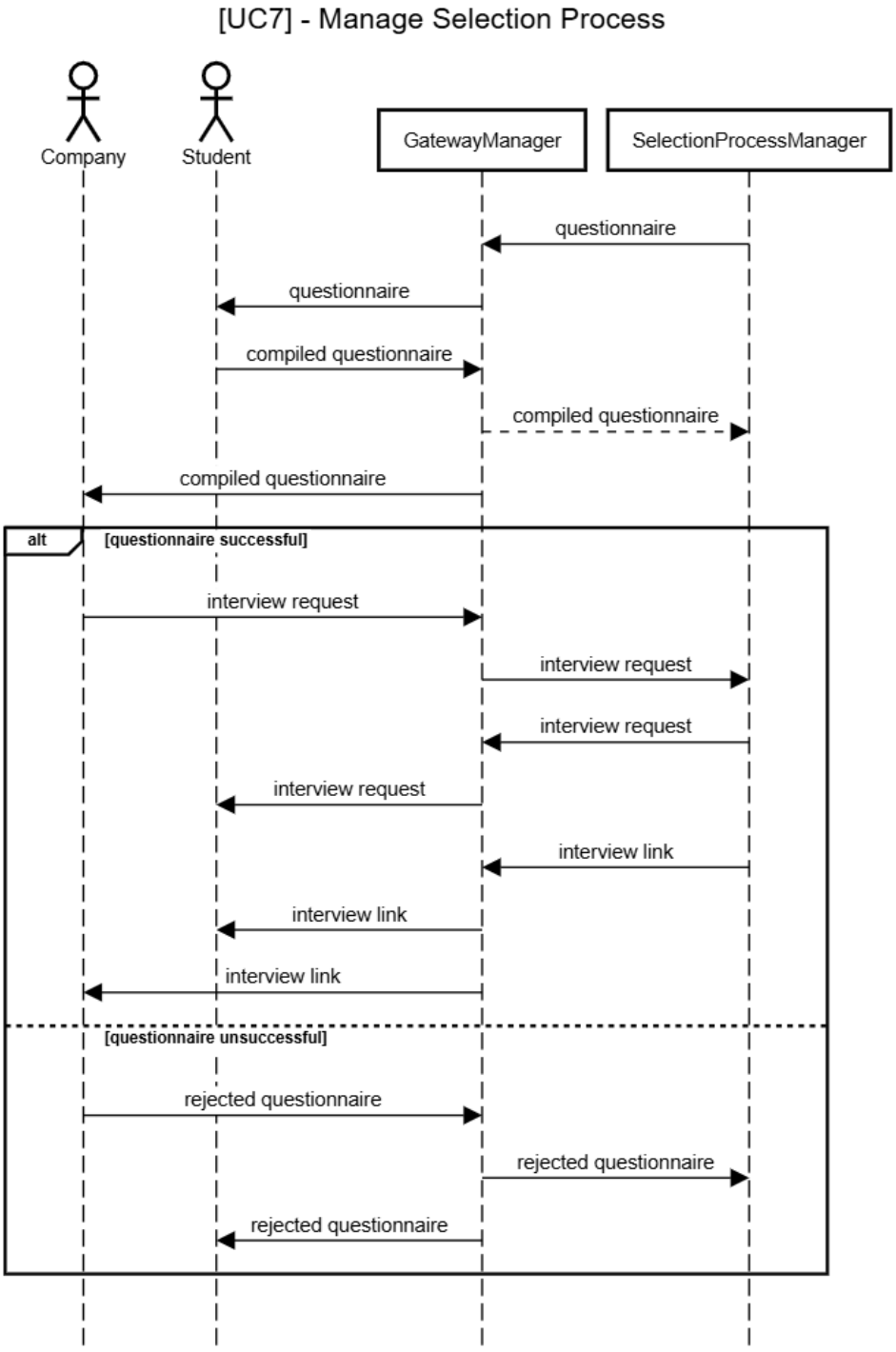
Once the response from the Company is received, the Selection Process Manager communicates to the Gateway Manager, sending the requestSuccessfullySent message to indicate that the selection process request has been successfully initiated. The Gateway Manager then forwards this confirmation to the Student.

Simultaneously, the Selection Process Manager publishes the response to the Selection Process Event Publisher, which then sends the response to the Event Bus. The Notification Event Consumer, which is monitoring the Event Bus for relevant events, consumes the event and processes it accordingly.

Following this, the Notification Event Consumer sends the response to the Notification Manager, which subsequently creates a notification for the Student or Company. The Notification Manager creates a new notification entry and stores it in the DBMS to keep a record of the event.

This sequence ensures that both the Selection Process and corresponding notifications are properly managed and communicated to the involved parties.

Manage Selection Process



The Manage Selection Process begins when the Selection Process Manager initiates the process. It sends a request for a questionnaire to the Gateway Manager, which is then responsible for forwarding the questionnaire to the Student.

The Student, upon receiving the questionnaire, fills it out and sends the compiled responses back to the Gateway Manager. The Gateway Manager then sends the completed

questionnaire to the Selection Process Manager for evaluation.

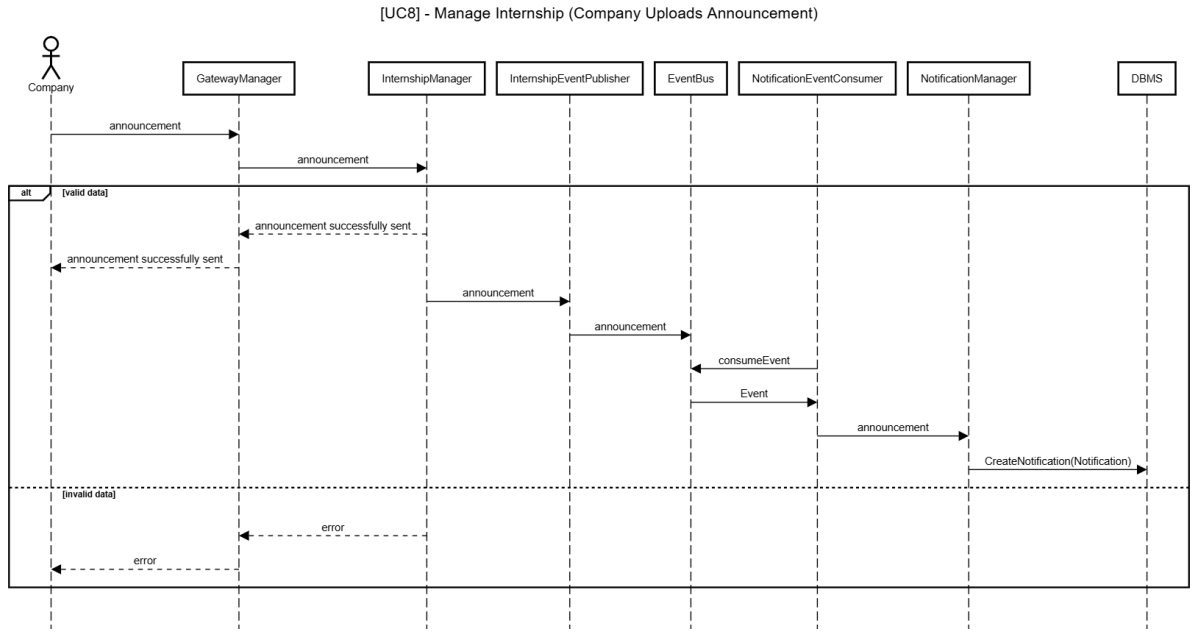
Following the evaluation, if the questionnaire is deemed successful, the Company sends an interview request to the Gateway Manager. The Gateway Manager forwards this interview request to the Selection Process Manager, which, in turn, sends the request to the Gateway Manager. The Gateway Manager then sends the interview request to the Student.

To proceed with the interview, the Selection Process Manager generates an interview link and sends it to the Gateway Manager. The Gateway Manager then communicates the interview link to both the Student and the Company to facilitate the interview scheduling.

In the case where the questionnaire is deemed unsuccessful, the Company sends the rejected questionnaire to the Gateway Manager. The Gateway Manager forwards the rejection to the Selection Process Manager, and subsequently, notifies the Student about the rejection.

This sequence ensures that the selection process is properly managed, with effective communication between all involved parties, and provides necessary updates regarding interview scheduling or rejections.

Company Uploads Announcement



The process begins when the Company sends an announcement to the Gateway Manager. Upon receiving the announcement, the Gateway Manager forwards it to the Internship Manager for further processing. The Internship Manager confirms that the announcement has been successfully sent and notifies the Gateway Manager, which, in turn, informs the Company that the announcement has been successfully processed.

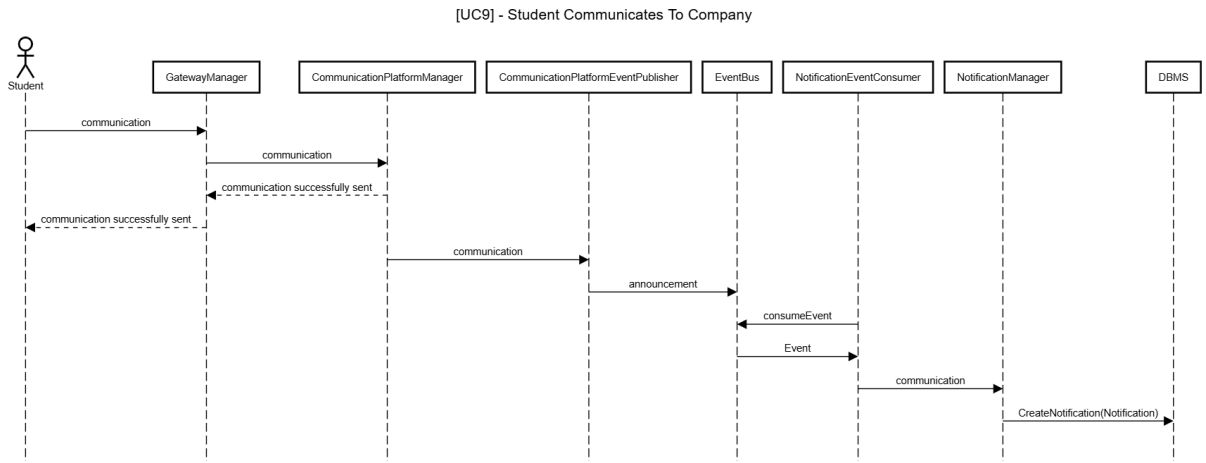
Next, the Internship Manager sends the announcement to the Internship Event Pub-

lisher, which then publishes the announcement to the Event Bus. The Notification Event Consumer is subscribed to the Event Bus and consumes the event when it is published. The Event Bus forwards the event to the Notification Event Consumer, which then passes the announcement to the Notification Manager.

The Notification Manager processes the announcement and creates a notification in the system by sending a request to the DBMS to store the notification. This ensures that the announcement is communicated to the relevant users, and a notification is generated and stored for future reference.

This sequence ensures that the announcement is successfully transmitted, processed, and stored within the system, while also notifying the appropriate users through the Notification Manager.

Student Communicates To Company



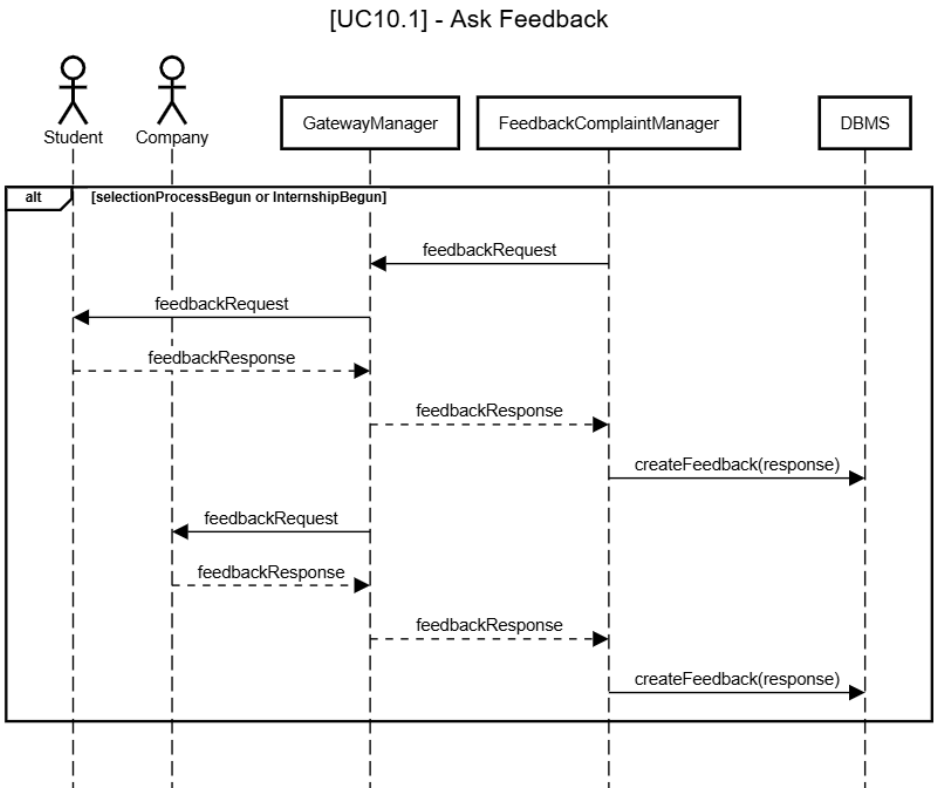
The process begins when the Student sends a communication request to the Gateway Manager. Upon receiving the request, the Gateway Manager forwards it to the Communication Platform Manager for processing. The Communication Platform Manager confirms the successful transmission of the communication and sends a success response back to the Gateway Manager, which then notifies the Student that the communication has been successfully sent.

Subsequently, the Communication Platform Manager forwards the communication to the Communication Platform Event Publisher, which publishes the event to the Event Bus. The Notification Event Consumer, which is subscribed to the Event Bus, consumes the event as soon as it is published. The Event Bus sends the event to the Notification Event Consumer, which processes the communication and sends it to the Notification Manager.

The Notification Manager then creates a notification in response to the communication and stores it in the DBMS by sending a request to create the notification. This ensures that the communication is logged and notifications are made available to relevant users within the system.

This sequence of events allows for seamless communication between the Student and the Company, along with the appropriate creation and storage of notifications for tracking and future reference.

Ask Feedback

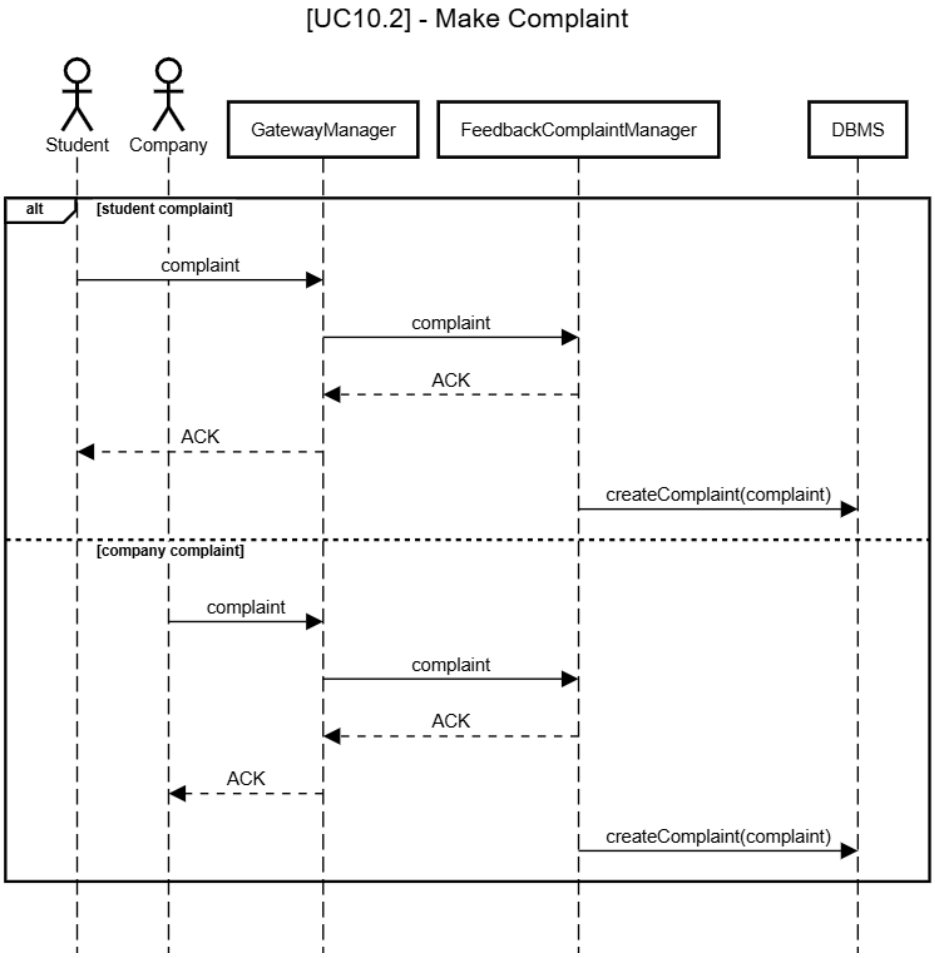


The process begins with the Feedback&Complaint Manager initiating a feedback request, which is sent to the Gateway Manager. Upon receiving this request, the Gateway Manager forwards it to the Student, prompting them to provide feedback. The Student then responds by sending their feedback to the Gateway Manager, which subsequently forwards the response to the Feedback&Complaint Manager. The Feedback&Complaint Manager stores the feedback in the DBMS by sending a request to create a new feedback entry.

Simultaneously, the Gateway Manager also sends a feedback request to the Company. The Company responds by providing their feedback, which is sent back to the Gateway Manager. The Gateway Manager then forwards this feedback to the Feedback&Complaint Manager, which once again stores the feedback in the DBMS by creating a new feedback entry.

This flow ensures that both the Student and Company are able to submit their feedback, which is then securely stored in the system for future reference or analysis.

Make Complaint

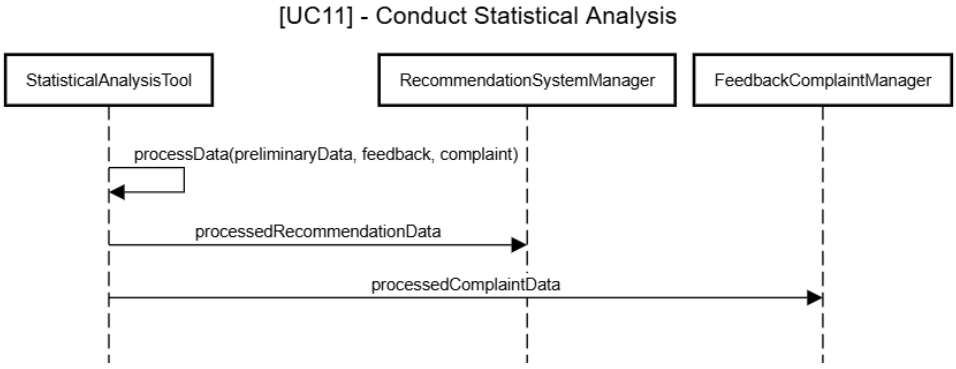


The process begins when the Student submits a complaint to the Gateway Manager. Upon receiving the complaint, the Gateway Manager forwards it to the Feedback&Complaint Manager for further processing. The Feedback&Complaint Manager acknowledges receipt of the complaint by sending an "ACK" response to the Gateway Manager, which then sends a confirmation back to the Student. The Feedback&Complaint Manager also stores the complaint in the DBMS by creating a new complaint entry.

Similarly, the Company can also file a complaint by sending it to the Gateway Manager. The Gateway Manager passes this complaint along to the Feedback&Complaint Manager. After receiving the complaint, the Feedback&Complaint Manager sends an "ACK" response to the Gateway Manager, which in turn confirms the action with the Company. Finally, the Feedback&Complaint Manager stores the Company's complaint in the DBMS by creating a new complaint entry.

This process ensures that complaints from both the Student and Company are properly received, acknowledged, and stored in the system.

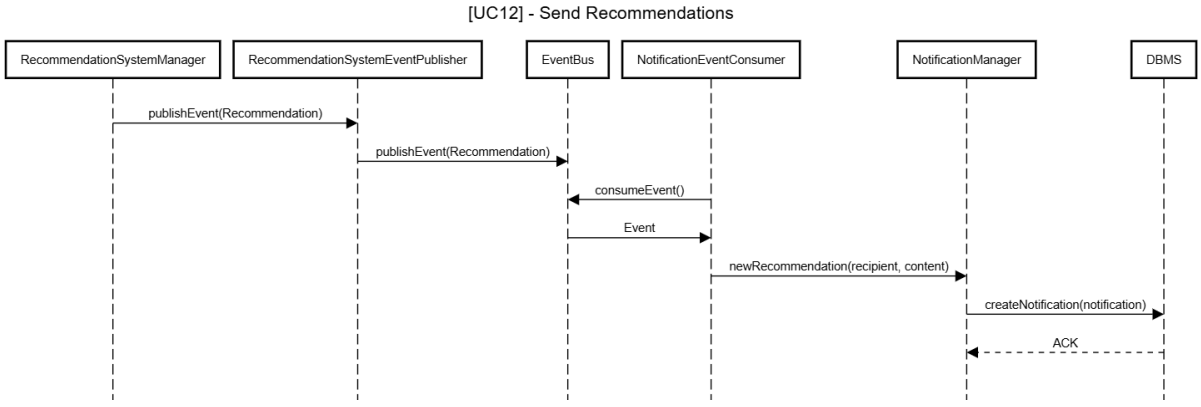
Conduct Statistical Analysis



The process begins when the Statistical Analysis Tool sends a request to process the data, which includes preliminary data, feedback, and complaints, to itself. After processing the data, the Statistical Analysis Tool sends the processed recommendation data to the Recommendation System Manager for further use. Simultaneously, the tool sends the processed complaint data to the Feedback&Complaint Manager for analysis and management.

This sequence of events ensures that both recommendation data and complaint data are appropriately processed and made available to their respective managers for further action.

Send Recommendations



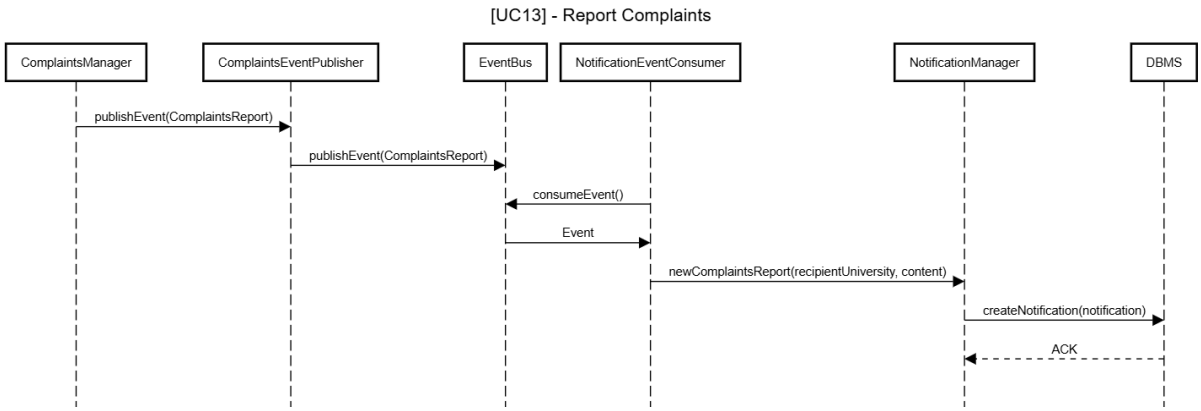
The process begins when the Recommendation System Manager sends a request to the Recommendation System Event Publisher to publish a new recommendation event. The event publisher then forwards this publish request to the Event Bus, where the Notification Event Consumer is actively listening for such events. Upon receiving the event, the Notification Event Consumer processes it and generates a new recommendation with the recipient and content details.

Subsequently, the Notification Manager is tasked with creating the corresponding notification. The notification data is sent to the DBMS for storage, and the DBMS confirms

the successful storage of the notification by sending an "ACK" response to the Notification Manager.

This sequence ensures that the recommendation is successfully disseminated and stored for future access.

Report Complaints

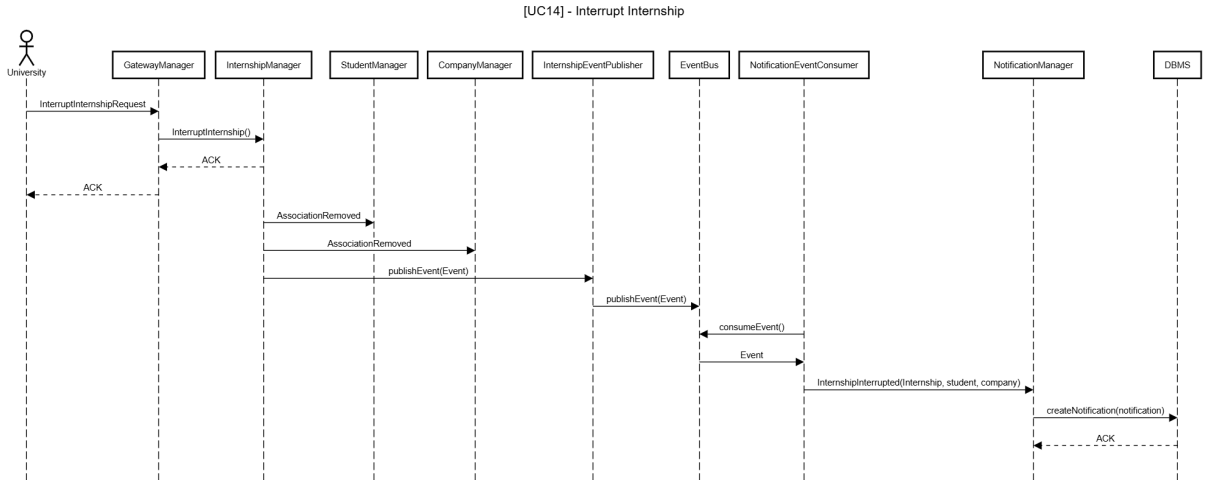


The process begins when the Complaints Manager sends a request to the Complaints Event Publisher to publish a complaints report event. The Complaints Event Publisher then forwards this event to the Event Bus, where the Notification Event Consumer is monitoring for incoming events. Upon detecting the event, the Notification Event Consumer processes the complaint and generates a new complaints report, which includes the recipient university and the content of the complaint.

Following this, the Notification Manager is responsible for creating a notification based on the generated complaints report. The notification details are sent to the DBMS for storage, and the DBMS responds with an "ACK" message to confirm the successful storage of the notification.

This sequence ensures that complaints are reported, notifications are generated, and the system maintains an up-to-date record of all complaints.

Interrupt Internship



The process begins when the University sends an InterruptInternshipRequest to the Gateway Manager. Upon receiving this request, the Gateway Manager forwards the InterruptInternship command to the Internship Manager. The Internship Manager processes the interruption request and sends an "ACK" confirmation to the Gateway Manager, which in turn sends a corresponding "ACK" message to the University.

Subsequently, the Internship Manager notifies the Student Manager and Company Manager about the removal of the internship association by sending an AssociationRemoved message to both. The Internship Manager then triggers the publishing of an event by sending a publishEvent request to the Internship Event Publisher, which subsequently forwards the event to the Event Bus.

The Notification Event Consumer, which is actively listening for events on the Event Bus, consumes the event and processes it. The Event Bus transmits the event to the Notification Event Consumer, which generates a new notification with the details of the interrupted internship, including the student and company involved. This information is sent to the Notification Manager, which is responsible for creating the notification in the system.

Finally, the Notification Manager stores the notification in the DBMS by sending a createNotification request, and the DBMS responds with an "ACK" to confirm the successful storage of the notification.

2.5. Component Interfaces

This section provides a detailed enumeration of the interfaces available across the various components of the Student&Companies system, accompanied by specifications for their internal methods. Each method is described to clarify its purpose and functionality.

The tables below are organized by microservice for better clarity and navigation.

1 - Gateway Microservice

DiscoveryServiceAPI

Return Type	Signature	Description
URL	findService(String serviceName)	This method is offered to allow all microservices to locate each other and therefore collaborate.
void	registerService(String serviceName, URL location)	This method allows a newly-created service to register itself on the list of available services, in order to be located by the other microservices if needed.
List<URL>	allServices()	This method returns a list of all the available services at the moment in the sys- tem.

GatewayControllerAPI

Return Type	Signature	Description
String	/selectRole	Allows to select the role of a user (either student, company or university).
NotificationView	/notification	Gets the list of notifications for the current user.
PersonalProfile	/profile	Returns the personal profile of the current user.

Return Type	Signature	Description
Create InternshipView	/create-internship	Returns the view to create a new internship for an authenticated company.
Create CVView	/create-cv	Returns the view to create a new CV for an authenticated student.
Boolean	/isInternshipAvailable	Returns a boolean value representing whether the internship is available or not. The internship name can be passed as parameter to the API call.
StudentProfileView	/sendApplication	Sends the student's application to an internship and returns to the student prodile view. The internship name and the username of the user can be passed as parameters to the API call.
List<String>	/all-internships-abs	Returns all the internships on S&C.
List<String>	/all-internships-company	Returns all the intership from a specific company. The username of the company can be passed as a parameter to the API call.
String	/internship-detail	Returns the details of a specific internship. The internship name can be passed as a parameter to the API call.
String	/CV-detail	Returns the details of a specific student CV. The student username can be passed as a parameter to the API call.

2 - User Management Microservice

UserAPI

Return Type	Signature	Description
Integer	/getUserRole	Returns the role of the specified user, the username can be passed to the API call as a parameter.
Boolean	/userExists	Checks whether the user accessing the system is a new user or not. The username can be passed as a parameter to the API call.
void	/saveUser	Adds a new user on S&C, the username and the role can be passed as parameters to the API call.

3 - Notification Microservice

NotificationAPI

Return Type	Signature	Description
List<Notification>	/getNotifications(String userId)	Retrieves all notifications for a specific user.
Boolean	/markAsRead(String notificationId)	Marks a specific notification as read.
Boolean	/sendNotification(Notification notification)	Sends a notification to a user or group of users.

Notification - EventConsumer

Return Type	Signature	Description
Event	consumeEvent()	Reads an event from the event bus.
List<Event>	consumeAllEvents()	Reads all the events published on the event bus.

4 - Recommendation System Microservice

RecommendationAPI

Return Type	Signature	Description
List<Internship>	/getRecommendations(String studentId)	Provides a list of internship recommendations for a student.
List<CV>	/getRecommendations(String companyId)	Provides a list of CV recommendations for a company.
Boolean	/submitPreferences(String userId, Preferences preferences)	Updates the user's preferences for recommendation refinement.
FeedbackReport	/analyzeFeedback()	Analyzes feedback data to improve recommendation algorithms.

Recommendation - EventPublisher

Return Type	Signature	Description
void	publishEvent(Event)	Publishes an event on the event bus.

5 - Communication Platform Microservice

CommunicationAPI

Return Type	Signature	Description
Boolean	/sendMessage(String senderId, String receiverId, String message)	Sends a message between a student and a company or viceversa.
List<Message>	/getMessages(String userId, String conversationId)	Retrieves all messages for a specific conversation.
Boolean	/createGroupChat(List<String> participantIds)	Creates a group chat for multiple participants.

Communication - EventPublisher

Return Type	Signature	Description
void	publishEvent(Event)	Publishes an event on the event bus.

6 - Selection Process Management Microservice

SelectionProcessAPI

Return Type	Signature	Description
Boolean	/scheduleInterview(String studentId, String companyId, Date date)	Schedules an interview between a student and a company.
List<Questionnaire>	/getQuestionnaire(String processId)	Retrieves the questionnaire for a specific selection process.
Boolean	/finalizeSelection(String processId, String studentId)	Finalizes the selection and assigns a student to the internship.

SelectionProcess - EventPublisher

Return Type	Signature	Description
void	publishEvent(Event)	Publishes an event on the event bus.

7 - Internship Management Microservice

InternshipManagementAPI

Return Type	Signature	Description
Boolean	/createInternship(Internship internship)	Allows companies to create a new internship.

Return Type	Signature	Description
Boolean	/updateInternship(String internshipId, Internship internship)	Updates an existing internship's details.
List<Internship>	/getAllInternships(String companyId)	Retrieves all internships created by a specific company.
Boolean	/deleteInternship(String internshipId)	Deletes a specific internship.

InternshipManagement - EventPublisher

Return Type	Signature	Description
void	publishEvent(Event)	Publishes an event on the event bus.

8 - Feedback&Complaint Microservice

Feedback&ComplaintAPI

Return Type	Signature	Description
Boolean	/submitFeedback(String userId, Feedback feedback)	Allows users to submit feedback about internships or processes.

Return Type	Signature	Description
Boolean	/submitComplaint(String userId, Complaint complaint)	Allows users to submit a complaint regarding internships.
List<Feedback>	/getFeedback(String internshipId)	Retrieves all feedback associated with a specific internship.
List<Complaint>	/getComplaint(String internshipId)	Retrieves all complaints associated with a specific internship.
ComplaintStatus	/getComplaintStatus(String complaintId)	Retrieves the status of a submitted complaint.

Feedback&Complaint - EventPublisher

Return Type	Signature	Description
void	publishEvent(Event)	Publishes an event on the event bus.

2.6. Selected Architectural Styles and Patterns

This section outlines the key architectural decisions made during the design of the system, providing insight into the reasoning behind each choice. The objective is to develop a solution that not only addresses the current needs but also offers room for future growth and adaptation. By offering a detailed perspective, this section ensures that both developers and stakeholders gain a clear understanding of how these decisions support the system's ongoing evolution and overall success.

Microservices Architecture

The decision to implement a microservices architecture stems from various factors that enhance the flexibility and scalability of the application:

- **Scalability:** Microservices enable the independent scaling of individual components, allowing for targeted resource allocation without the need to scale the entire system, resulting in an agile and adaptable application.
- **Fault Tolerance:** Microservices architecture reduces the risk of system-wide failures. By isolating services, a failure in one module has minimal impact on others, ensuring the system remains operational.
- **Deployment and Productivity:** Microservices promote continuous integration and delivery processes. The separation of concerns across services increases team efficiency, with each member focusing on a specific service, thus fostering better collaboration.
- **Maintainability:** Microservices simplify the management and maintenance of the system by decoupling services, making it easier to work on individual components without having to understand the entire system, ultimately improving overall system maintainability.

RESTful APIs

The decision to utilize the REST architectural style for developing our APIs enables a scalable, efficient, and interoperable communication framework. This approach brings several significant benefits:

- **Simplicity and Intuitiveness:** RESTful APIs are designed around standard HTTP methods, ensuring an easy-to-understand and intuitive communication model.
- **Scalability:** By being stateless, REST facilitates horizontal scalability, allowing the system to handle increased load by adding more instances without complex coordination between them.
- **Flexibility and Modularity:** REST APIs promote a clean, modular architecture, with each microservice managing a specific data domain. This separation of concerns enhances maintainability and clarity in the system.
- **Interoperability:** Since REST relies on universal HTTP protocols, it ensures cross-platform compatibility, enabling seamless integration with various technologies and devices.
- **Statelessness:** The stateless nature of REST simplifies server-side logic and error recovery, enhancing system reliability and making the application more resilient to failures.

Model-View-Controller (MVC)

The implementation of the Model-View-Controller (MVC) architecture provides a structured, maintainable, and scalable foundation for the application. This design choice offers several advantages:

- **Modular Organization:** MVC splits the Manager components within the microservices into three distinct sections: Model, View, and Controller. This organi-

zation improves code readability, simplifies debugging, and allows each component to be developed independently.

- **Separation of Concerns:** By enforcing a clear distinction between the different parts of the application, MVC simplifies development and enables easier maintenance, extension, and modification of individual components without unintended consequences for others.
- **Reusability and Extensibility:** The modular nature of MVC fosters code reusability, allowing developers to reuse components across different areas of the application or even in future projects.
- **Testability:** MVC enhances the ability to test the application, allowing for isolated testing of each component. This ensures that each part functions correctly and contributes to a more stable system.
- **Improved Collaboration:** The division of responsibilities in MVC facilitates team collaboration, as different developers can work on the Model, View, or Controller independently, streamlining the development process.

Event-Driven Architecture (EDA)

The adoption of Event-Driven Architecture (EDA) is pivotal in enhancing real-time responsiveness and scalable communication in the Student&Companies system. Specifically, this architecture supports asynchronous notifications, enabling seamless interaction between the various components, including the Notification Consumer, Selection Process Publisher, and Internship Publisher. These components generate events when specific actions occur, such as the start or end of a selection process or internship. The Notification Consumer can then asynchronously process these events, crafting the necessary notifications for the involved parties without interrupting the core functionalities of the Selection Process and Internship microservices.

In addition to enabling asynchronous processes, EDA brings several key advantages to the system:

- **Decoupled Components:** EDA promotes a decoupled architecture, where components communicate through events. This flexibility allows the independent development, testing, and deployment of each component, ensuring minimal dependencies between them.
- **Scalability:** EDA supports scalability by distributing services and components across the system. This asynchronous nature ensures the system can handle varying loads efficiently, adapting to increasing demands without compromising performance.

2.6.1. Database Management

The data layer of the Student&Companies application consists of a centralized database that ensures persistent data storage and access for all microservices within the system.

This design is motivated by the fact that, while each microservice in the system focuses on different functionalities, they often need to interact with overlapping sets of data (such as student profiles, company internships, and feedback). By using a shared database, the system facilitates consistent data management and seamless data sharing across microservices.

In terms of database management, internal partitions are implemented to segregate data for different microservices. To ensure reliability and high availability, each partition is replicated.

3 | User Interface Design

This chapter displays a more in depth analysis of the "3.1.1" section called "User Interface" of the Requirements Analysis and Specification Document (RASD) related to the S&C project. Within the scope of detail permitted in a design document, it is possible to explore the various interactions the platform provides to external users, whether they are students or companies' authorized employees. The screenshots displayed in this section serve as mockups of the user interfaces provided by the system, therefore they have to be interpreted as general guidelines for the designers and developers to grasp the application overall look at the end of the implementation process.

3.1. NavBar

The navbar is the cardinal navigational component of the S&C platform, designed to provide quick and intuitive access to the platform's core features. It features a minimalist, user-friendly design with the following icons:

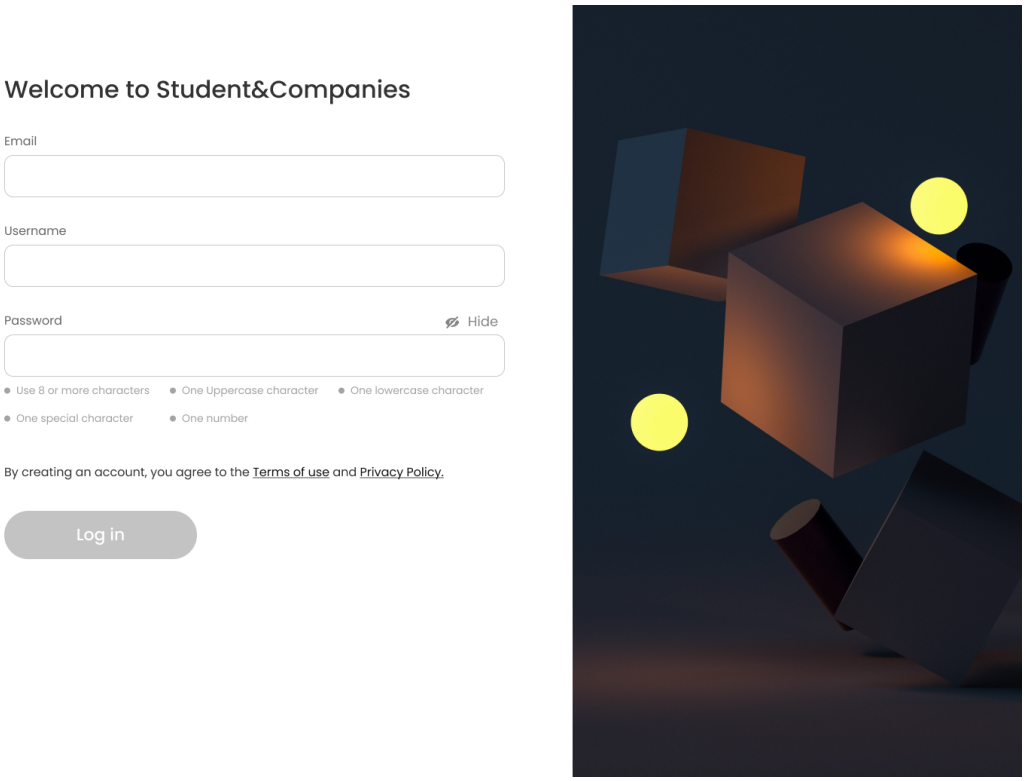
- Messages: Provides access to the messaging interface, allowing real-time communication between students and companies.
- Inbox: Displays notifications, updates, and invitations from companies or students in a streamlined format.
- Search: Redirects to the Internship Browsing Page, enabling users to find internships, through either a keyword search or recommended hot topics.
- Profile: Takes users to their profile page, where they can view and edit their information, CV, or internships offers (depending on the user).

The navbar ensures a seamless user experience, remaining accessible across all pages of the platform.

3.2. Login Page

Login Page Description: The login page serves as the primary gateway for users to access the S&C platform. It features a clean and intuitive design to ensure a seamless user experience. The page includes the following key elements:

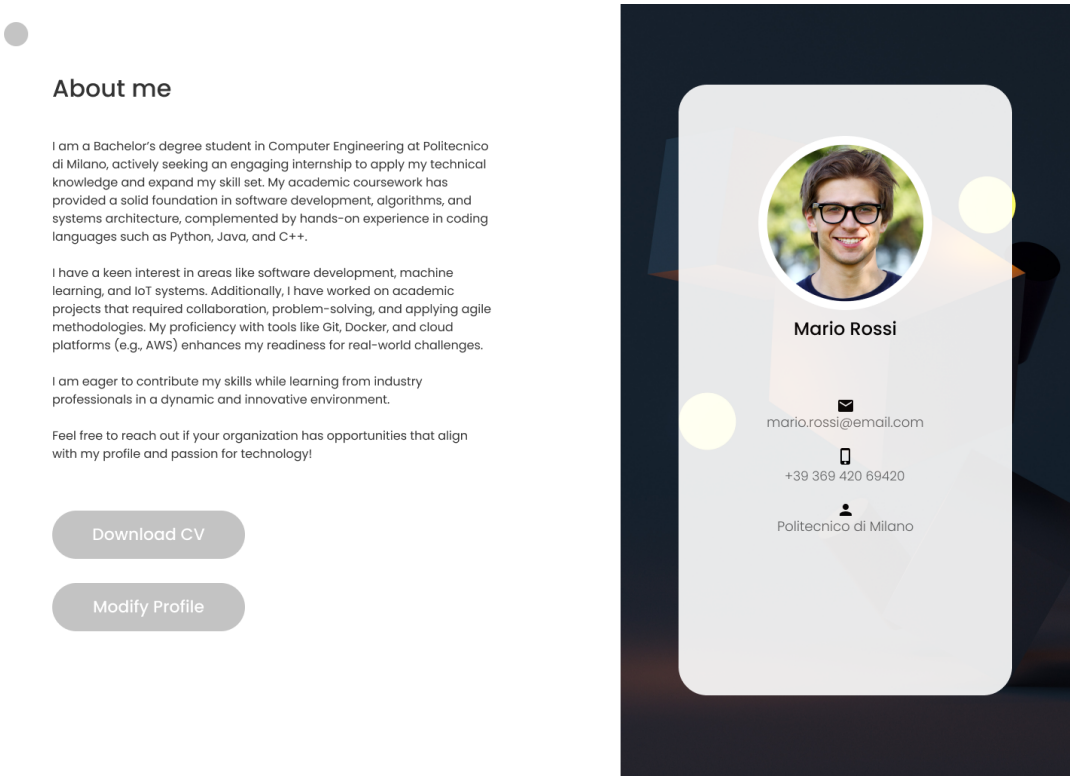
- Username Field: a text input where users enter their username, checks the validity of the data entry.
- Email Field: A text input field where users enter their registered email address. It validates the format to ensure correct data entry.
- Password Field: A secure input field for users to enter their password, with masking for privacy. An optional "Show Password" "Hide" toggle enhances usability.
- Login Button: A button that, when clicked, authenticates the provided credentials and, if correct, grants access to the platform.



3.3. Student Profile

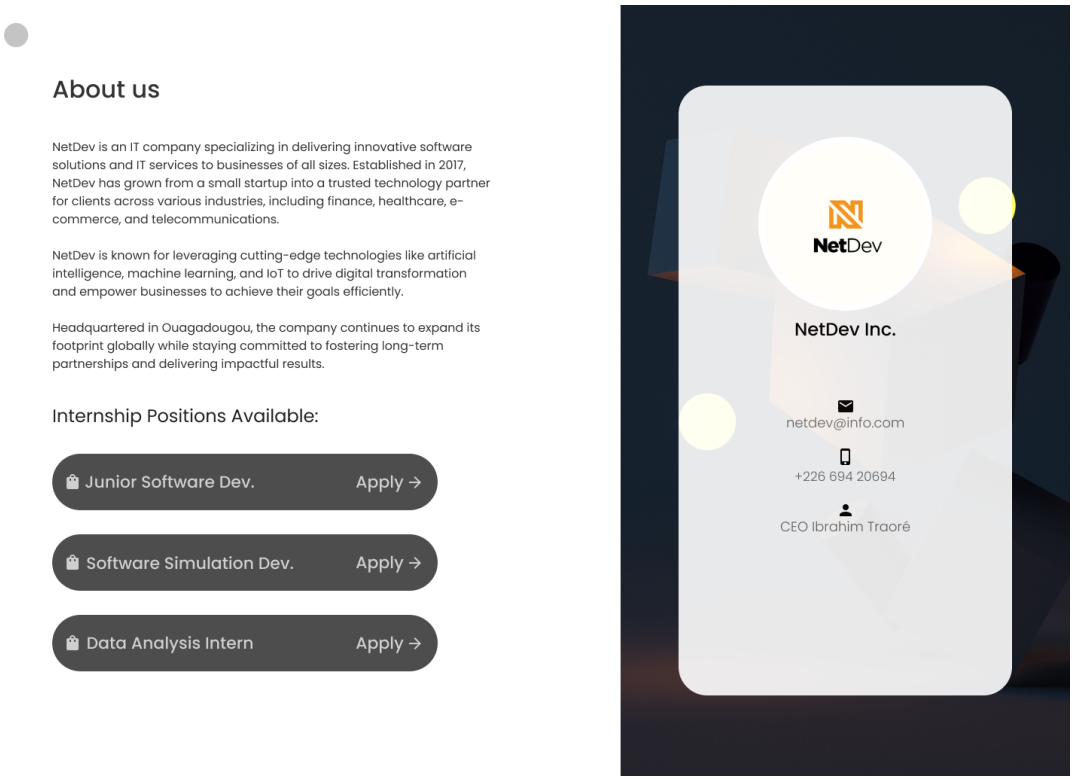
Student Profile: The student profile page is designed to showcase a student's essential information in a structured and user-friendly manner. Each profile includes a customizable profile picture, along with the student's email and phone number as primary contact details. A unique feature of the student profile is the inclusion of the university name at the bottom-right section, highlighting the institution the student is currently affiliated with. Additionally, along side a brief "about me" description, the page provides an option

on the left side for users to download the student's CV directly, facilitating easy access to their qualifications and achievements.



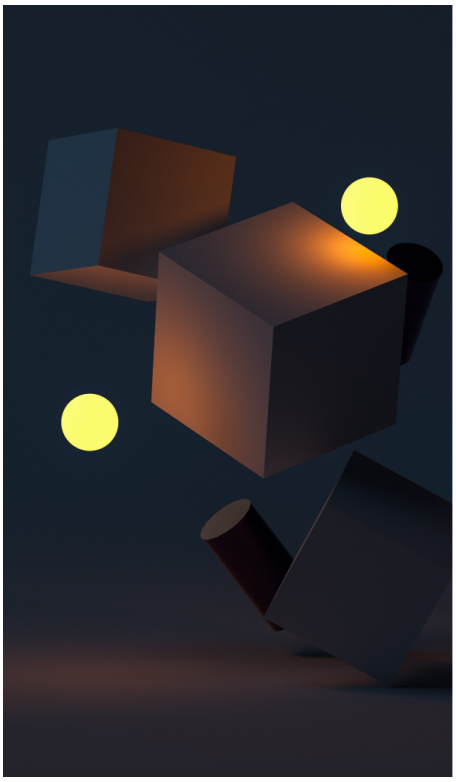
3.4. Company Profile

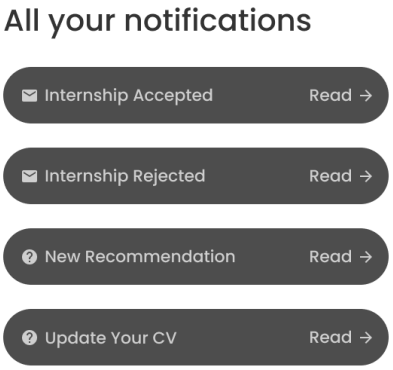
The company profile page offers a professional overview of an organization, displaying its identity and opportunities. Similar to the student profile, it features a customizable profile picture, along with the company's email and phone number for direct communication. Distinguishing itself from the student profile, the company page displays the name of its current CEO at the bottom-right section. On the left side, following an "about us" paragraph, the page provides a dedicated section to showcase the internships the company is currently offering, enabling students to explore potential opportunities with ease.



3.5. Notification Inbox

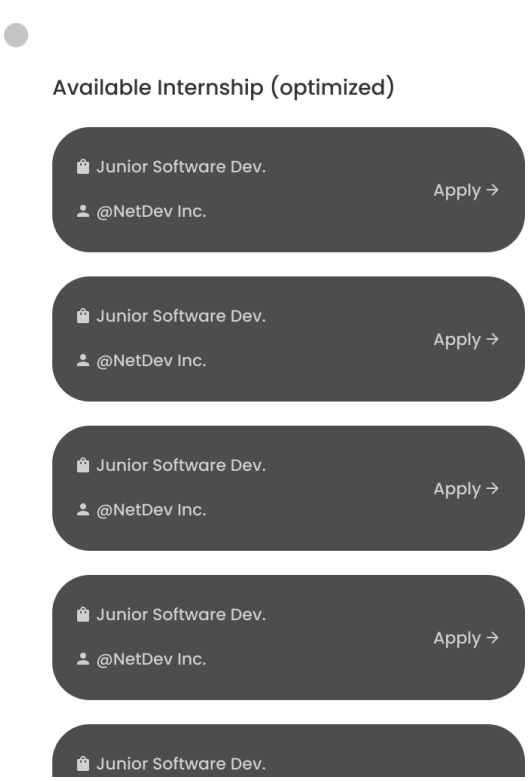
This section displays the list of notifications received by the system and potentially other users, the order in which are displayed is from the most recent one at the top to the least recent at the bottom.





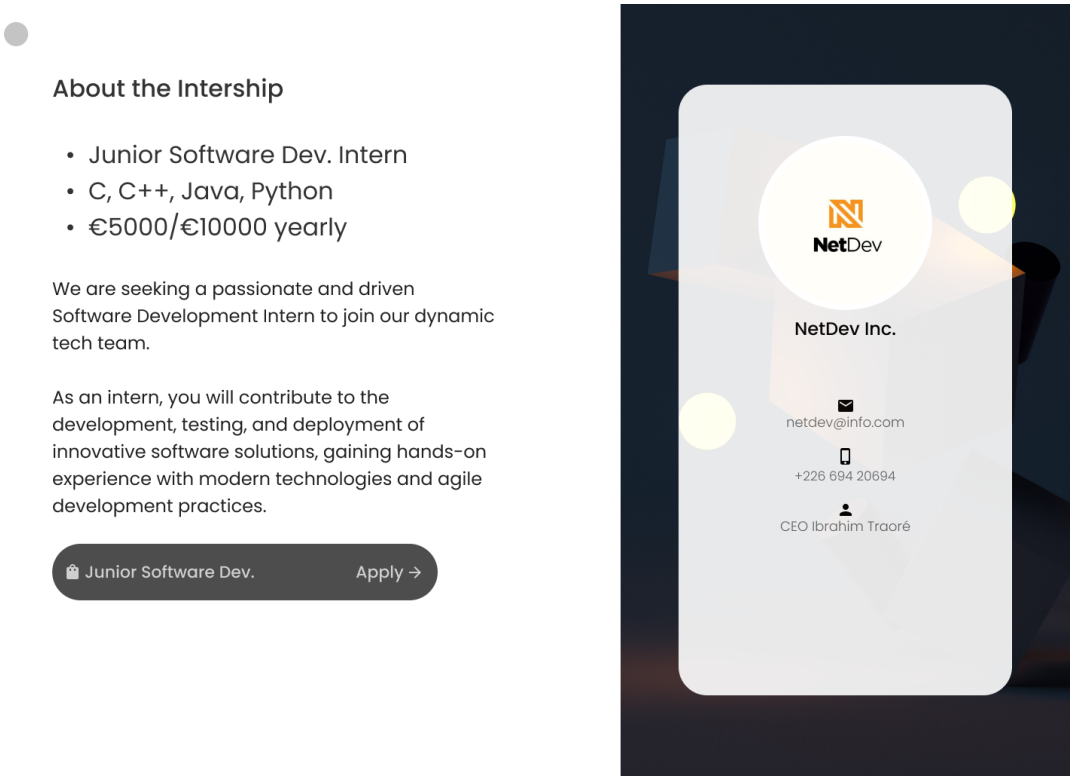
3.6. Internship Browsing Page

The interface is divided in two main parts, on the right there is the search bar followed by the list of hot topics (internship related) of the current period, on the left there are the recommended internships available, sorted by the Recommendation Engine.



3.7. Internship Information Details

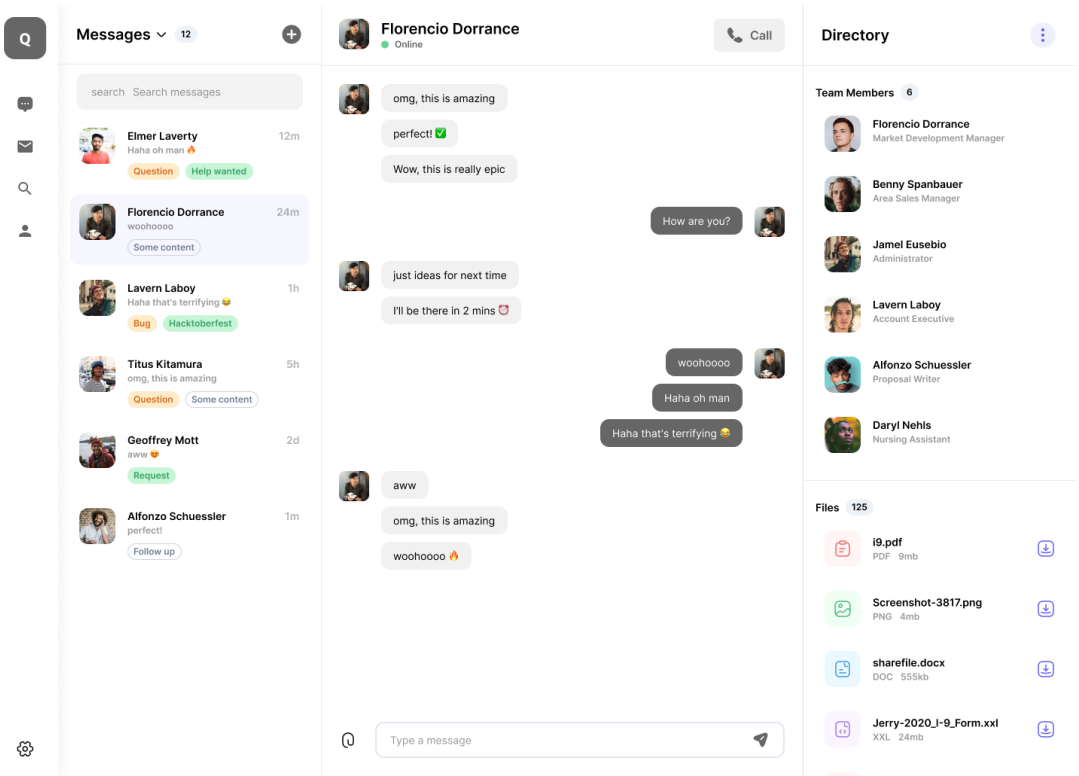
In this page are displayed all the information regarding the selected internship, followed by the information about the company that is offering the position. The button "Apply", once clicked, starts the application process.



3.8. Chat

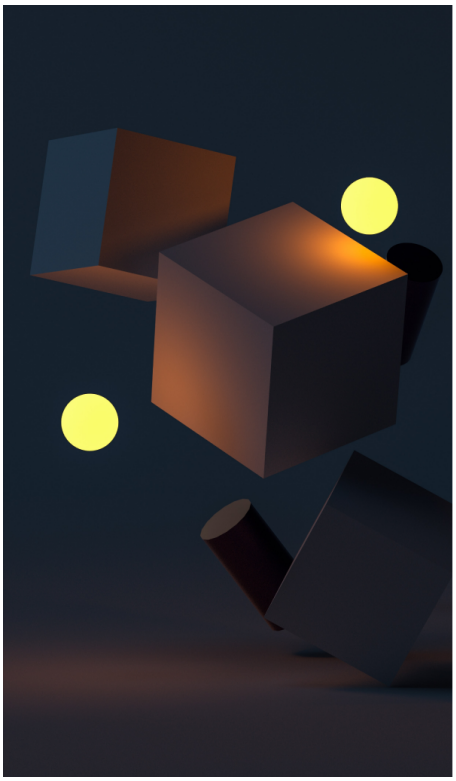
The chat page is a streamlined interface designed for real-time communication between users. Key elements include:

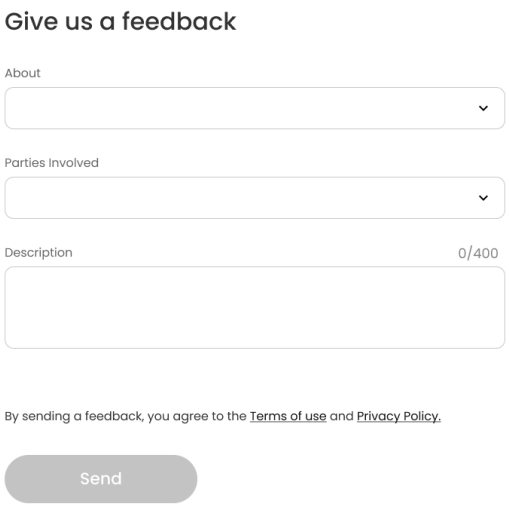
- Chat List: Displayed on the left side of the page, it shows a list of recent conversations, including the name and profile picture of the other party.
- Chat Window: Occupying the main area of the page, the chat window displays the active conversation in a chronological message thread.
- Input Field: Located at the bottom of the chat window, the input field allows users to type messages. Additional controls, send button and file attachment option, are present along side the text field.
- User Details: A small section or header that shows information about the chat participant, such as their name, profile picture, and availability status.
- Team Members Section: Positioned to the side, this section lists all participants in the current conversation. It displays their names, profile pictures, and role in the company, supporting easy identification and collaboration.



3.9. Feedback/Complaint

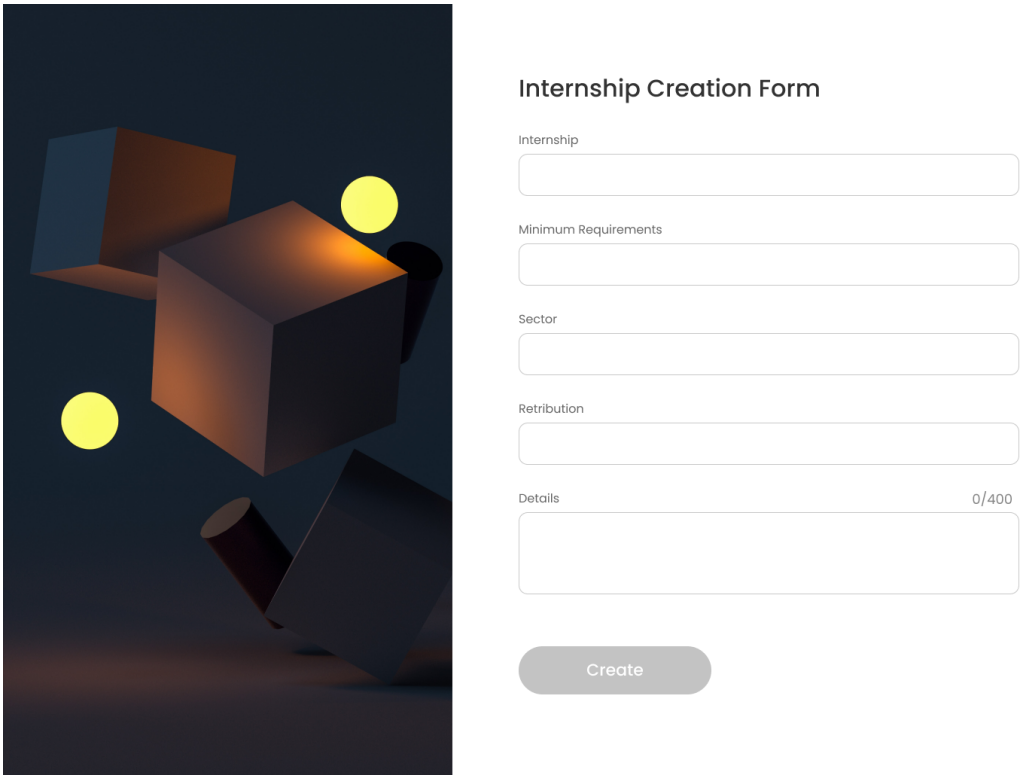
This page contains a form used to send a formal complaint or give feedback directly to the S&C platform itself. It is divided in "About" drop list, "Parties involved" drop list, "Description" text field and at the bottom the "Send" button.





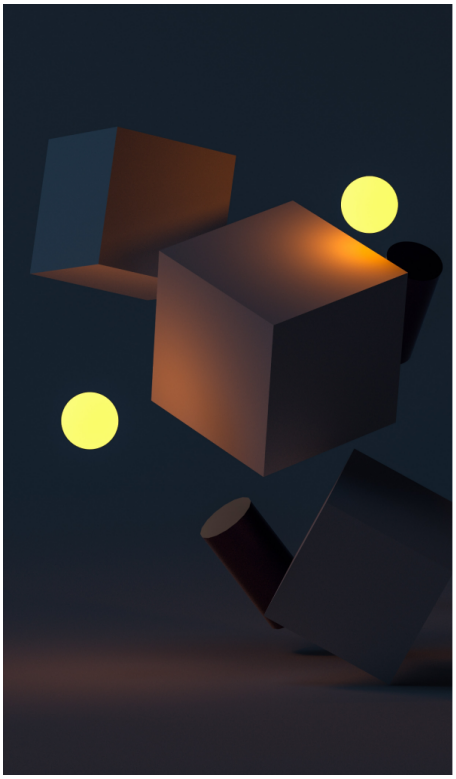
3.10. Internship Creation Form

This page contains a form used by authorized companies to create an intership position in the S&C platform. It is divided in the following text fields: "Intership", "Minimum Requirements", "Sector", "Retribution", "Description" and at the bottom the "Create" button.



3.11. Preliminary Questionnaire

This page contains a form required as a preliminary questionnaire at the beginning of the selection process. It is divided in "Select your References" drop list, "Areas of expertise" drop list, "List of course" text field, "Technical skill set" text filed, "Description" text field and at the bottom the "Send" button.



Preliminary Internship Questionnaire

Select your references

Areas of expertise

List of courses

Technical skill set

Description (something about yourself) 0/400

Send

4 | Requirements Traceability

The traceability matrix provided below aims to establish a clear connection between the requirements outlined in the Requirements Analysis and Specification Document (RASD) and the components of the Student&Companies system responsible for addressing them.

The referenced components are discussed in greater detail in Section 2.2, Component View, of this Design Document, where their roles and responsibilities within the system are thoroughly explained.

ReqID	Requirement Description	Components Involved
R1.1	The system allows students to create their profile.	Gateway Microservice (Dispatcher), User Management Microservice (Student Manager).
R1.2	The system allows students to upload their CV.	Gateway Microservice (Dispatcher), User Management Microservice (Student Manager), Recommendation System Microservice (Recommendation System Manager).
R1.3	The system provides suggestions to students on how to submit their CV.	Gateway Microservice (Dispatcher), User Management Microservice (Student Manager).
R1.4	The system allows students to browse internships.	Gateway Microservice (Dispatcher), Selection Process Microservice (Selection Process Manager, Selection Process Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)

ReqID	Requirement Description	Components Involved
R2.1	The system allows companies to create their profile.	Gateway Microservice (Dispatcher), User Management Microservice (Company Manager).
R2.2	The system allows companies to upload their internships.	Gateway Microservice (Dispatcher), User Management Microservice (Company Manager), Recommendation System Microservice (Recommendation System Manager).
R2.3	The system provides suggestions to companies on how to submit their internship descriptions.	Gateway Microservice (Dispatcher), User Management Microservice (Company Manager)
R3.1	The system notifies students about suitable internships.	Recommendation System Microservice (Recommendation System Manager, Recommendation System Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)
R3.2	The system notifies companies about suitable student CVs.	Recommendation System Microservice (Recommendation System Manager, Recommendation System Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)

ReqID	Requirement Description	Components Involved
R3.3	The system recommends based on keyword searching.	Recommendation System Microservice (Recommendation System Manager, Recommendation System Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)
R3.4	The system recommends based on the characteristics of students and internships.	Recommendation System Microservice (Recommendation System Manager, Recommendation System Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)
R3.5	The system recommends based on statistical analysis.	Statistical Analysis Tool, Recommendation System Microservice (Recommendation System Manager, Recommendation System Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)
R3.6	The system collects information regarding the selection process.	Selection Process Microservice (Selection Process Manager), Gateway Microservice (Dispatcher)
R3.7	The system collects information regarding the internship.	Internship Microservice (Internship Manager, Internship Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer), Gateway Microservice (Dispatcher)

ReqID	Requirement Description	Components Involved
R3.8	The system asks students and companies to provide feedback and suggestions.	Feedback&Complaint Microservice (Feedback&Complaint Manager), Gateway Microservice (Dispatcher)
R4.1	The system allows students to apply for an internship.	Gateway Microservice (Dispatcher), Selection Process Microservice (Selection Process Manager, Selection Process Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)
R4.2	The system allows companies to accept student applications.	Gateway Microservice (Dispatcher), Selection Process Microservice (Selection Process Manager, Selection Process Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)
R4.3	The system allows students to sign the contract for the internship.	Selection Process Microservice (Selection Process Manager), Gateway Microservice (Dispatcher), Feedback&Complaint Microservice (Feedback&Complaint Manager)
R4.4	The system allows companies to sign the contract for the internship.	Selection Process Microservice (Selection Process Manager), Gateway Microservice (Dispatcher), Feedback&Complaint Microservice (Feedback&Complaint Manager)

ReqID	Requirement Description	Components Involved
R4.5	The system supports the selection process by helping manage interviews and also finalise the selections.	Selection Process Microservice (Selection Process Manager), Gateway Microservice (Dispatcher), Feedback&Complaint Microservice (Feedback&Complaint Manager)
R5.1	The system provides spaces for the official announcements.	Internship Microservice (Internship Manager, Internship Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)
R5.2	The system provides spaces where interested parties can complain, communicate problems, and provide information about the current status of the ongoing internship.	Communication Platform Microservice (Communication Platform Manager, Communication Platform Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer), Gateway Microservice (Dispatcher)
R6.1	The system allows universities to create their profile.	Gateway Microservice (Dispatcher), User Management Microservice (University Manager).
R6.2	The system allows universities to monitor the situation of internships and handle complaints.	Feedback&Complaint Microservice (Feedback&Complaint Manager, Feedback&Complaint Event Publisher), Notification Microservice (Notification Manager, Notification Event Consumer)

ReqID	Requirement Description	Components Involved
R6.3	The system allows universities to interrupt an internship.	Gateway Microservice (Dispatcher), Internship Microservice (Internship Manager, Internship Event Publisher), User Management Microservice (Student Manager, Company Manager), Notification Microservice (Notification Manager, Notification Event Consumer)

5 | Implementation, Integration and Test plan

5.1. Overview

This section outlines the processes of implementation, integration, and testing for the Students&Companies (S&C) platform, describing how its key functionalities were developed and validated to ensure reliability and effectiveness.

The chapter is divided into three main parts:

1. Feature Identification: This part focuses on the platform's key functionalities, derived from primary use cases, and highlights the microservices that support them - Section 5.2.1
2. Component Integration and Testing: This section describes the integration of microservices to test features, using a thread-based strategy that mirrors real-world scenarios - Section 5.2.2
3. System Testing: Here, the focus shifts to testing the platform as a whole, ensuring that all components work together seamlessly to deliver a complete and robust system - Section 5.3

The platform was designed with a microservices-based architecture, enabling the division of functionalities into modular components. During implementation, each microservice was developed to address a specific aspect of the system, such as recommendation, selection process management, or feedback handling. However, the testing strategy was designed to go beyond validating individual components, ensuring that the overall system functionalities meet the required specifications.

The testing process was executed in three phases:

1. Unit Testing: In the first phase, unit tests were performed on individual microservices to ensure that each component functions correctly in isolation. This phase focused on verifying the internal logic and functionality of each microservice, ensuring a solid foundation for the system.
2. Feature Integration Testing: The next phase involved integration testing based on the platform's key features. Microservices were integrated to test specific use cases, such as the matching process between students and companies, ensuring that they worked together as intended. A thread-based strategy was used to test these

features incrementally, starting from simple interactions and gradually integrating more complex functionalities.

3. System Testing: In the final phase, comprehensive system tests were performed to validate the entire platform. This stage involved testing the system as a whole, ensuring that all microservices and integrated features functioned cohesively and met the overall functional requirement

By combining unit tests, feature-based integration tests, and comprehensive system testing, this approach provides a robust framework to identify and resolve issues progressively. This ensures that the final system is both scalable and reliable, meeting the diverse needs of its users.

5.2. Implementation Plan

5.2.1. Features Identification

For the development of the Students&Companies (S&C) platform, several key features have been identified, each representing a core functionality that directly supports user interactions and business processes. The order in which these features are listed reflects the sequence in which they will be implemented and tested. Each feature is a logical component of the system, providing specific, user-visible functionalities that contribute to the overall experience and effectiveness of the platform.

Profile Creation Features

The profile creation feature is available for students, companies, and universities. For students and companies, the feature is more detailed compared to universities.

For students, the functionality includes the ability to initially register on the platform, add information to their profile, and upload photographs and documents. For companies, they must be able to upload information and photographs to their profiles and create pages for internships, where they can also upload relevant information, photos, and documents. For universities, the profile functionality is simpler; they only need to upload basic information.

Furthermore, in the meantime, the Statistical Analysis Tool collects data from students and companies, both from their profiles and their actions.

Selection Process Management Features

The selection process management feature consists of several stages.

Initially, the student browses company profiles and views their internship opportunities. When the student finds an interesting internship, they apply to start the selection process. At this point, the company reviews the student's profile and decides whether to accept them into the selection process. If the decision is positive, a notification is sent to the student, informing them of the acceptance. This new association is then saved.

The platform also supports managing the selection process by enabling questionnaires through the system and providing the option for online interviews.

Communication Features

The communication feature is designed to facilitate interaction between students and companies throughout the internship. This feature includes two main components.

The first is the official internship page, a dedicated space where both parties can interact with distinct permissions. Companies have the ability to post announcements, while students can engage by commenting on these posts.

The second component is a direct chat, which functions as a private messaging system between the student and the company. This allows for streamlined discussions and the

exchange of internship-related information.

Notifications are sent to users to highlight important events, such as new official announcements, incoming chat messages, or the conclusion of the internship. All data related to these communications and the internship itself is securely stored in the database.

Overview of the features: feedback and complaints collection, recommendation, and complaint report

For clarity, here is an overview of the features: feedback and complaints collection, recommendation, and complaint report. These features will be analyzed in detail later on.

The static analysis tool collects preliminary data from student profiles, company profiles, as well as data during the selection process and the internship (see UC 4, 5, 6, and 7). The feedback and complaints microservice periodically requests feedback from both students and companies, both during the selection process and throughout the internship, and records any complaints (see UC 10).

The static analysis tool then receives and processes this data, sending the relevant information to the recommendation microservice for recommendations and to the feedback and complaints microservice for complaint reports (see UC 11). The recommendation microservice produces the actual recommendations (see UC 12), while the feedback and complaints microservice generates the complaint report (see UC 13).

Feedback and Complaints Collection Features

The feedback and complaints collection feature is divided into two functionalities: gathering feedback and collecting complaints.

The feedback collection process is initiated by the system. Periodically, the system sends requests to both students and companies, prompting them to provide feedback related to the ongoing selection process or internship. This ensures timely and relevant input from users regarding their experiences. The feedback provided is then securely stored in the database for further processing and reference.

On the other hand, the complaint collection process is user-initiated. Both students and companies have the ability to submit complaints at any time. They can access a dedicated section within the management areas of either the selection process or the internship and express their concerns or issues. These complaints are also saved in the database, ensuring they are properly documented and accessible for resolution or analysis.

Providing Recommendations Features

The purpose of the recommendation system feature is to provide targeted recommendations through notifications to both students and companies. To achieve this, several stages are required. Initially, information must be collected. This begins with gathering preliminary data from user profiles and actions within the system. Additionally, more detailed information is provided directly by users, including feedback and complaints.

These large amounts of data are then processed by the system, organized, and categorized based on their intended destination. Positive and useful information relevant to the recommendations is ultimately sent to the dedicated recommendation section, where it is further processed. The final stage involves transforming this data into notifications for the users.

Complaints Report Generation Features

The complaint report generation feature aims to produce a report of complaints, categorized by student and internship, enriched with additional useful information automatically gathered from the system. This report is then sent to universities via a notification, as they are responsible for monitoring the situation.

The system collects both preliminary data and data provided by users, similar to the process outlined in the previous section. Preliminary data includes user actions and profile information, while more detailed data is gathered from user feedback and complaints. The Statistical Analysis Tool collects the preliminary data, while the Feedback and Complaints Microservice gathers and processes the feedback and complaints submitted by users. It's important to note that user feedback, in addition to formal complaints, can also serve as a valuable source of information for the complaint report.

Once the data is collected, it is processed by the system. Relevant data for the complaint report is then forwarded to the appropriate section, where it undergoes further processing to create the finalized complaint report. This report is enriched with the necessary information, making it suitable for sending to the universities. Finally, the report is transformed into a notification and sent to the universities for monitoring purposes.

5.2.2. Components Integration and Testing

In the integration testing phase of the Students&Companies (S&C) platform, a hread-based strategy was chosen to ensure a systematic and realistic evaluation of the system. This approach focuses on testing complete end-to-end flows, or "threads," that represent specific use cases, such as matching students with internships or managing the selection process.

The thread-based strategy was selected because it allows the system to be tested in a way that closely mirrors real-world usage. By focusing on functional threads derived from key features, we can ensure that the microservices interact seamlessly to deliver the desired functionalities. This incremental approach also simplifies the detection and resolution of issues, as it starts with the core features and progressively integrates additional functionalities.

In practice, the strategy involves first testing simpler flows that require fewer components, such as user profile creation, and then gradually introducing more complex scenarios, like recommendation and communication. This ensures that each functionality is validated individually and in combination with others, providing confidence in the system's robustness and reliability.

Profile Creation Features

The profile creation feature supports students, companies, and universities, with students and companies having more detailed functionalities than universities.

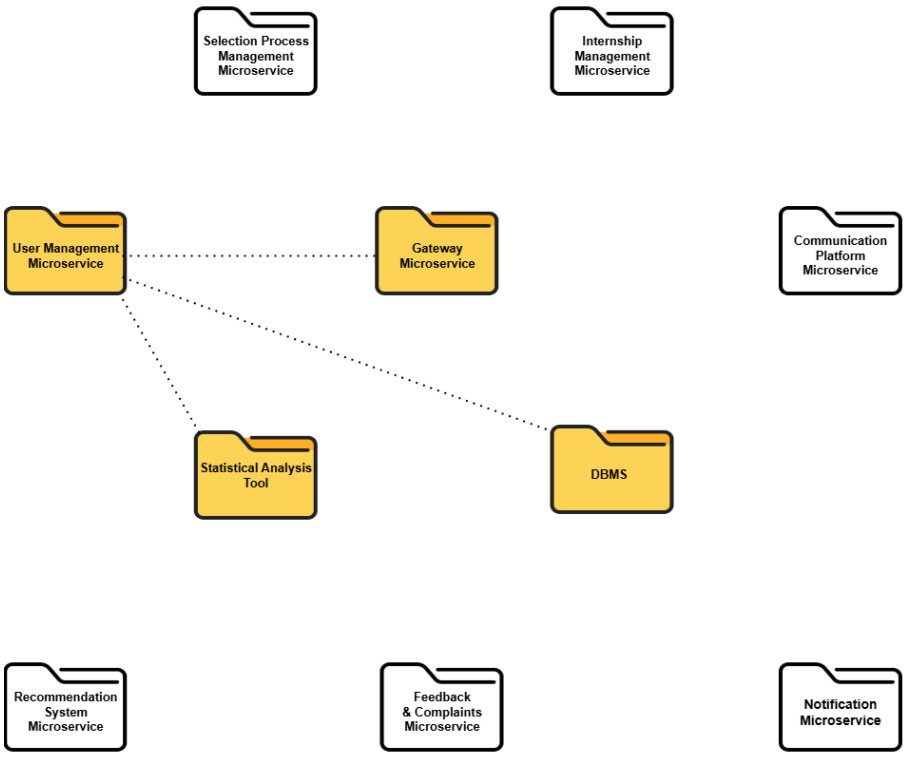
The process starts with user registration via the gateway microservice, which validates the user's information, such as email and password. For students, the registration is managed by the student user management microservice, which processes and stores the data in the database. Similarly, companies and universities have their data handled by their respective microservices.

After registration, users log in through the gateway microservice. Students can add detailed information to their profiles, including skills, experiences, and documents, managed by the student user management microservice and stored in the database. Companies can add company descriptions, upload documents and photos, and create internship pages, with the company user management microservice handling these operations. Universities have simpler profiles, where only basic details are uploaded through the university user management microservice.

Additionally, the Statistical Analysis Tool participates by collecting data from the profiles and actions of both students and companies. This includes information from student and company profiles as well as their search activities, such as internships viewed or applied for. This data is used to provide valuable insights into the matching process and recommendations.

Throughout this process, the gateway microservice handles connections and secure data transfer, while the user management microservices ensure proper profile creation, updates, and storage. The Statistical Analysis Tool contributes by analyzing user data to enhance the platform's recommendation and matching functionalities.

Here is a schematic view of the development and testing of this feature:


Selection Process Management Microservice
Internship Management Microservice
User Management Microservice
Gateway Microservice
Communication Platform Microservice
Statistical Analysis Tool
DBMS
Recommendation System Microservice
Feedback & Complaints Microservice
Notification Microservice

Selection Process Management Features

The selection process management feature is designed to guide both students and companies through a series of steps ultimately culminating in the formal acceptance of the student for an internship.

The process begins when a student browses the available internship opportunities posted by companies. This step is enabled by the gateway microservice, which provides the platform interface and allows students to easily navigate through different company profiles and their internship offers. This step is made possible also thanks to the usser management microservice that provides the substance for this browsing process. As the student explores these opportunities, their profile information, preferences, and actions are managed by the student user management microservice, which ensures their data is updated and stored in the database.

Once the student finds an internship that interests them, they can apply to start the selection proess. This application triggers the selection process microservice, which records the application and initiates the workflow for the company to review. The company can then access the student's profile and decide whether to accept them into the selection process. This decision-making is facilitated by the company user management microservice, which retrieves the student's details stored in the database, allowing the company to make an informed decision based on the student's qualifications and other relevant data.

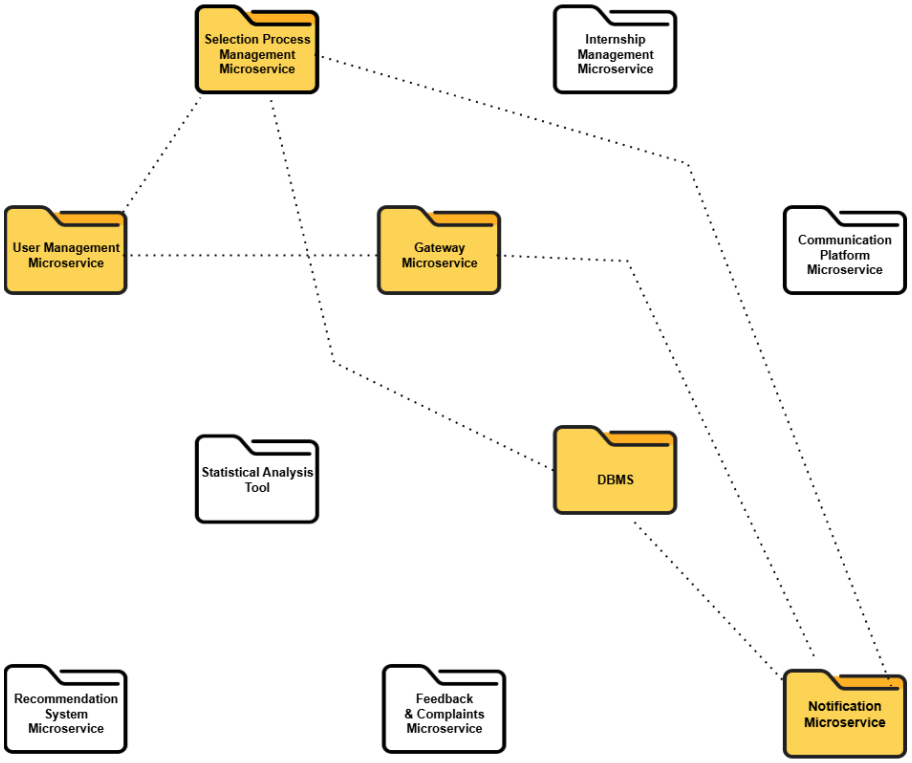
If the company accepts the student, the notification microservice sends an automatic notification to the student, informing them of the company's decision. This notification marks the official start of the selection process, and the new association between the student and the company is recorded in the system by the selection process microservice.

Once the student is accepted, the selection process continues with the student completing questionnaires designed to assess their fit for the internship. These questionnaires are managed and delivered by the selection process microservice, which ensures that they are assigned to the appropriate students and tracked throughout the process. All responses are stored in the database for later review by the company or other relevant stakeholders.

Additionally, the platform provides the option for online interviews, which can be scheduled by the company and the student. The gateway microservice facilitates the scheduling and conducting of these interviews, while the selection process microservice ensures that the details of the interview, such as timing and participant information, are properly coordinated. The user management microservices continue to handle the specific user data for both the student and the company during this phase. At the end if the selection process is successfully completed, the student is officially accepted for the internship.

Finally, throughout the entire selection process, all information—including applications, questionnaires, responses, and interview details—is stored in the database. This centralized storage ensures that all data is easily accessible for future reference, follow-ups, or reporting.

Here is a schematic view of the development and testing of this feature:


Selection Process Management Microservice
Internship Management Microservice
User Management Microservice
Gateway Microservice
Communication Platform Microservice
Statistical Analysis Tool
DBMS
Recommendation System Microservice
Feedback & Complaints Microservice
Notification Microservice

Communication Features

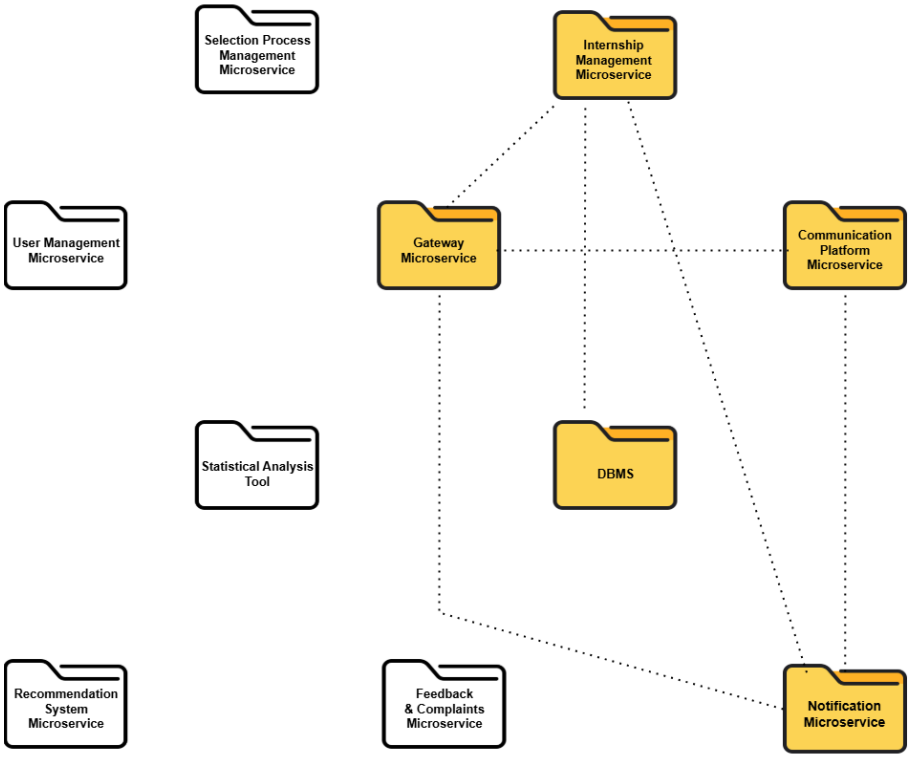
The communication feature is designed to enable effective interaction between students and companies throughout the internship. The process begins when a user accesses the platform through the gateway microservice, which handles user authentication and ensures secure connections. After logging in, users can navigate to the relevant sections of the platform where the communication tools are available.

One key component of this feature is the official internship page. This serves as a dedicated space for interaction between students and companies, with distinct permissions assigned to each party. Companies can use this page to post announcements related to the internship, and students can engage with these posts by adding comments. The internship management microservice ensures that announcements are correctly linked to their respective internship pages, while the communication microservice facilitates the commenting functionality. To keep users informed, the notification microservice sends alerts whenever new announcements are made.

Another important component is the direct chat system, which allows private communication between students and companies. Messages exchanged through this chat are processed by the communication microservice, ensuring secure and real-time delivery. Notifications for new chat messages are generated by the notification microservice, ensuring that users are promptly alerted. All interactions, including announcements, comments, and chat messages, are securely stored in the database for future reference.

Notifications play a crucial role in this feature, ensuring that users are kept updated about important events. Whether it's a new announcement, a received message, or the conclusion of an internship, the notification microservice manages the delivery of alerts. These notifications are seamlessly integrated into the user experience via the gateway microservice, providing visibility to users both during active sessions and upon logging in. Throughout the process, the internship management microservice ensures that all communications and interactions are accurately associated with their respective internships, with the database acting as the central repository for all related data.

Here is a schematic view of the development and testing of this feature:


Selection Process Management Microservice
Internship Management Microservice
User Management Microservice
Gateway Microservice
Communication Platform Microservice
Statistical Analysis Tool
DBMS
Recommendation System Microservice
Feedback & Complaints Microservice
Notification Microservice

Feedback and Complaints Collection Features

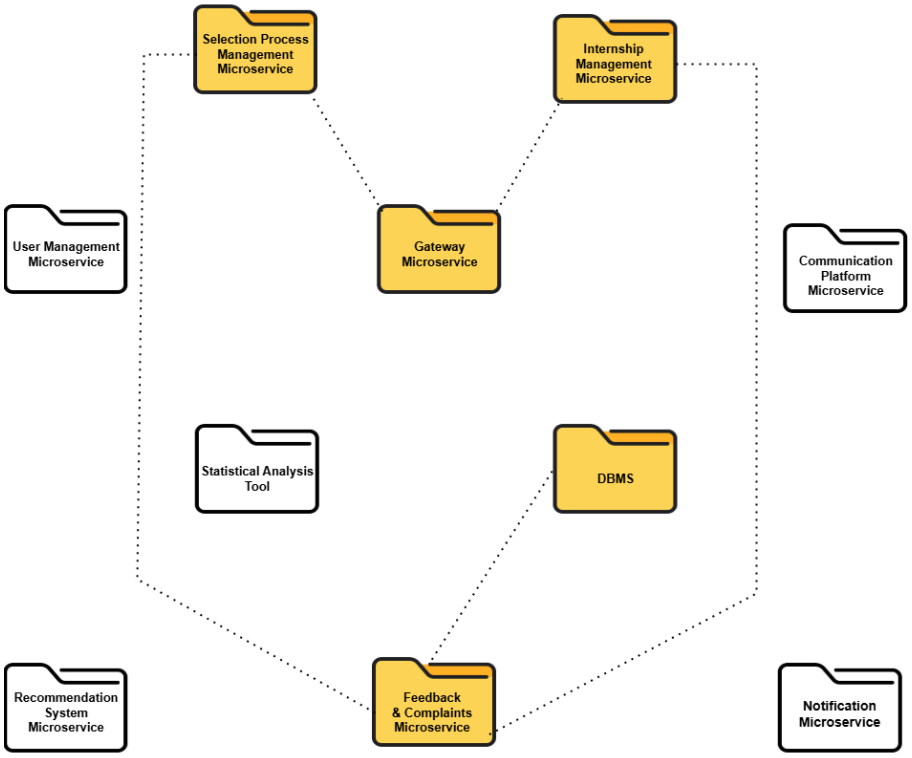
The feedback and complaints collection feature operates through two main functionalities: feedback collection and complaint submission. Each process is carefully structured to ensure seamless user interaction and efficient backend operations, with several microservices working together to manage data flow and processing.

The feedback collection process is automated and initiated by the feedback and complaints microservice. At scheduled intervals, this microservice directly interacts with the selection process management microservice or the internship management microservice to identify ongoing processes requiring feedback. It then sends requests to students and companies, prompting them to provide feedback related to these specific contexts. Once users submit their feedback through the platform's interface, the data is processed by the feedback and complaints microservice and associated with the relevant stage or event. The feedback is then securely stored in the database, creating a structured record for future analysis and decision-making.

The complaint submission process, on the other hand, is user-driven. Students and companies can access a dedicated section within the platform to submit complaints at any time. This section is integrated into the management interfaces for both the selection process and internships, allowing users to express their concerns seamlessly. When a complaint is submitted, it is received directly by the feedback and complaints microservice, which validates the input and queries the selection process management or internship management microservices to associate the complaint with the appropriate context. The validated complaint is then saved in the database, ensuring it is properly documented and available for resolution or review.

The microservices involved in these functionalities include the feedback and complaints microservice, which orchestrates the operations of collecting, organizing, and processing user input. It collaborates directly with the selection process management microservice and the internship management microservice to retrieve contextual information, ensuring that feedback and complaints are accurately linked to relevant events or stages. The database serves as the central repository for storing all collected data, providing secure and reliable storage for future use.

Here is a schematic view of the development and testing of this feature:


Selection Process Management Microservice
Internship Management Microservice
User Management Microservice
Gateway Microservice
Communication Platform Microservice
Statistical Analysis Tool
DBMS
Recommendation System Microservice
Feedback & Complaints Microservice
Notification Microservice

Providing Recommendations Features

The recommendation system is designed to deliver targeted notifications to both students and companies, involving several key stages, each supported by different microservices.

The process begins with data collection, which is split into two categories: preliminary data and detailed user data. Preliminary data is automatically gathered from user profiles and their actions on the platform, such as internship views, applications, and company activities. The Statistical Analysis Tool microservice is responsible for collecting this data, tracking user interactions. In addition, more detailed data comes directly from users, such as feedback and complaints. This information helps the system refine recommendations. The Feedback & Complaints Service collects and stores this user-generated input.

Once data is collected, it needs to be processed and organized. The Statistical Analysis Service categorizes the data based on relevance and user type, ensuring that it is ready for recommendation generation. This step ensures that only actionable data is passed on to the next phase.

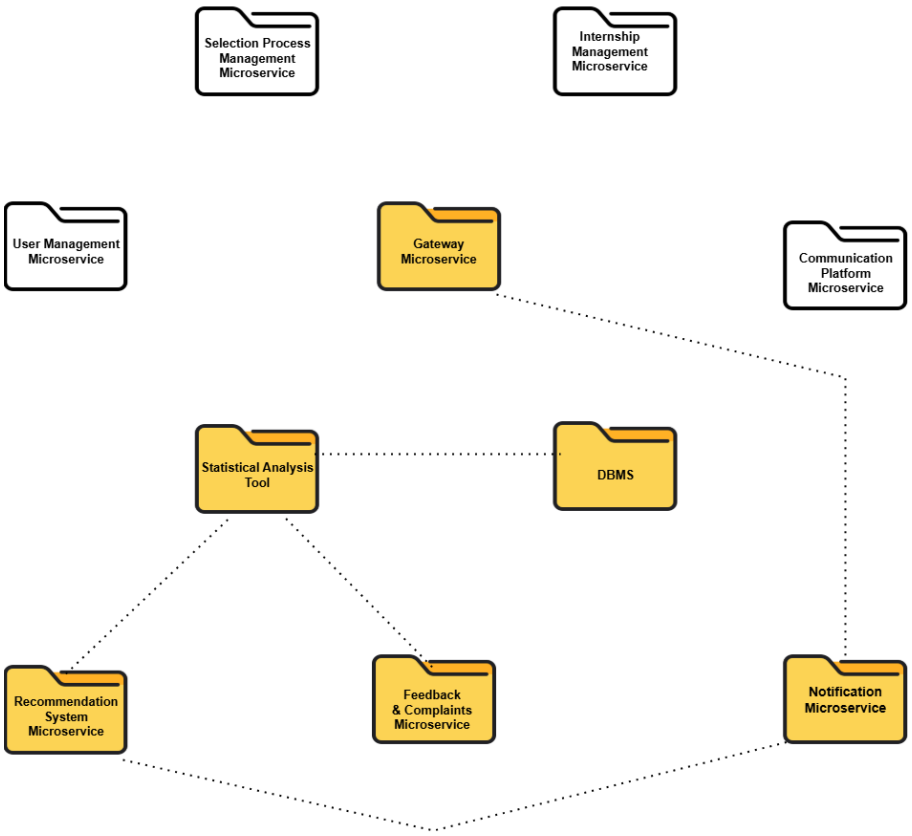
Next, the system generates personalized recommendations by analyzing the processed data. For example, a student who has applied for several software engineering internships may receive similar suggestions, or a company with multiple marketing internship listings may be connected with students in that field. The Recommendation Microservice handles this stage by applying algorithms to identify patterns and generate tailored suggestions.

After recommendations are generated, they are formatted into notifications. The Notification Microservice transforms these recommendations into user-friendly formats suitable for delivery across various channels, such as email or in-app alerts.

Finally, the Gateway Microservice ensures that the notifications are delivered to the appropriate users. It routes them according to user preferences, ensuring that each notification reaches its target in the most effective way.

These microservices work together seamlessly to ensure that the recommendation system provides relevant, timely notifications to students and companies, enhancing their experience on the platform.

Here is a schematic view of the development and testing of this feature:


Selection Process Management Microservice
Internship Management Microservice
User Management Microservice
Gateway Microservice
Communication Platform Microservice
Statistical Analysis Tool
DBMS
Recommendation System Microservice
Feedback & Complaints Microservice
Notification Microservice

Complaints Report Generation Features

The complaint report generation feature is designed to produce a detailed report of complaints, categorized by student and internship, and enriched with additional useful information gathered automatically from the system. The purpose of this report is to allow universities to monitor and address any issues raised by students or companies. The process begins with the collection of both preliminary data and user-provided data, which includes feedback and complaints submitted by students and companies.

The first step is the collection of preliminary data, which is automatically gathered from the system. This includes user actions, such as a student's interaction with internship listings—like viewing, applying for internships, or saving posts—and company actions, such as posting or managing internship listings. Additionally, user profile data, such as registration details, is collected. The Statistical Analysis Tool is responsible for gathering this preliminary data, tracking how users engage with the platform. In parallel, more detailed data comes directly from users in the form of feedback and complaints. Feedback could include general comments or ratings, and complaints refer to dissatisfaction with specific internships or platform features. Notably, feedback, in addition to formal complaints, can serve as a valuable source of information for the complaint report. The Feedback and Complaints Microservice is responsible for gathering and processing both feedback and complaints, ensuring that all user-generated input is captured.

Once the data is collected, the system processes it to organize and categorize the information for the complaint report. This stage involves distinguishing between different types of data, s uch as separating formal complaints from general feedback. Relevant details are extracted from both feedback and complaints to ensure only useful and actionable data is included in the report. The Feedback and Complaints Microservice plays a critical role in this stage, as it processes and organizes the data into a structured format, making it easier to generate the final complaint report.

After the data has been processed, the system moves on to generate the actual complaint report. The report is organized by internship and student, clearly indicating which complaints or feedback are tied to specific internships or individuals. This report is enriched with additional information automatically generated by the system, such as trends in user behavior, context surrounding the complaints, and any other relevant data that can provide further insight into the issues raised. The Feedback and Complaints Microservice generates this report, ensuring it contains all necessary details and is presented in an actionable format for the universities.

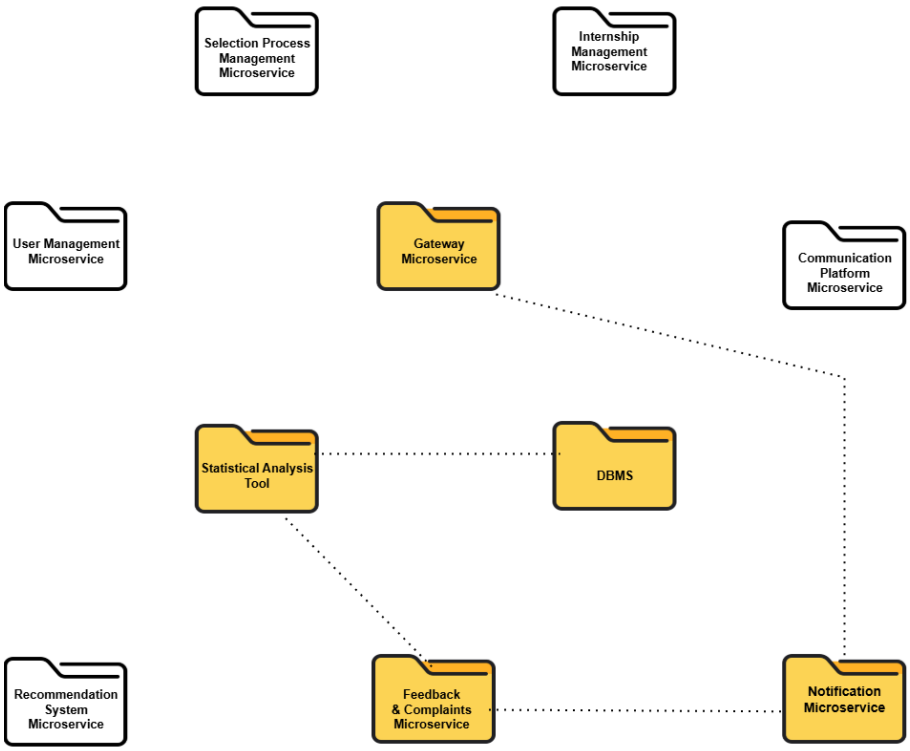
Once the complaint report has been finalized, it is converted into a notification format that can be sent to universities. This involves transforming the raw data into a user-friendly notification that universities can easily review and act upon. The Notification Microservice handles this step, ensuring that the report is clear, concise, and suitable for delivery across various channels, such as email or in-app notifications.

Finally, the notification containing the complaint report is sent to the universities. The Gateway Microservice is responsible for routing the notification to the appropriate recipients at the university, ensuring it reaches the relevant departments or staff members responsible for monitoring and addressing student and company concerns. The notifi-

cation is delivered through the university's preferred channels, ensuring timely access to the information.

Through this detailed process, the complaint report generation feature ensures that universities are promptly informed about any issues that need attention. By combining user feedback, complaints, and system-generated data, the feature helps universities stay informed, monitor the situation, and take appropriate action to improve the platform experience for all users.

Here is a schematic view of the development and testing of this feature:



5.3. System Testing

System testing is a critical phase of the development lifecycle where the fully integrated Students & Companies (S&C) platform is rigorously evaluated to ensure that it meets both functional and non-functional requirements. The testing environment is carefully designed to closely replicate the actual production setup, enabling a comprehensive analysis of the platform's behavior under realistic conditions.

Functional testing is focused on verifying that the platform meets the functional specifications outlined in the requirements documentation, such as use cases. Key functionalities are examined, including profile creation and management for both students and companies, the recommendation system's ability to match students with internships based on CV data and company requirements, and support for the selection process, such as interview scheduling and structured questionnaires. Communication features, including notifications and messaging, are tested to ensure seamless interaction between users, while feedback and complaint management functionalities are validated to confirm that users can effectively submit and track concerns.

Non-functional testing evaluates the system's performance, scalability, and reliability under a variety of conditions. This includes:

- Performance Testing: Measuring response times, throughput, and resource utilization to ensure that the system operates efficiently under typical conditions.
- Load Testing: Gradually increasing the number of concurrent users or sustaining a steady workload to verify that the platform can handle expected user volumes without performance degradation.
- Stress Testing: Simulating extreme conditions, such as sudden spikes in user activity or system failures, to test the platform's ability to recover and maintain availability in challenging scenarios.

To ensure thorough coverage, the testing methodology combines manual and automated approaches. Manual testing is employed to validate specific scenarios, such as edge cases in the recommendation system or workflows related to complaint resolution. Automated testing leverages techniques such as fuzz testing, concolic execution, and search-based strategies, enabling repeated evaluations of the system under varying conditions. These methods ensure the platform's robustness, reliability, and consistency across diverse environments.

Through this structured and comprehensive system testing approach, the S&C platform is validated to perform reliably and effectively in real-world scenarios, meeting the needs of students, companies, and universities alike.

6 Effort Spent

Time spent (measured in hours) on every section of the DD document by team member.

Student	Introduction	Architectural Design	User Interface Design	Requirements Traceability	Implementation, Integration, and Test Plan
Simone	10	30	10	5	5
Toni	5	25	25	1	5
Valeria	20	10	5	1	25

7 | References

- Sequence Diagrams made with sequencediagram.org
- Chapter 5 Diagrams and Component Diagrams of Chapter 2 made with draw.io
- UI and other diagrams made with Figma