ITD Ricci, Paoli, Grisoni

POLITECNICO
MILANO 1863

ITD
Implementation and Test Deliverable

Deliverable:	ITD
Title:	Implementation and Test Deliverable
Authors:	Lorenzo Ricci, Matteo Giovanni Paoli, Samuele Grisoni
Version:	1.1
Date:	09-02-2025
Download page:	https://github.com/Slaitroc/RicciPaoliGrisoni/
Copyright:	Copyright © 2025, Ricci, Paoli, Grisoni – All rights reserved

0 Contents

Contents . . . 3

1 Document Information . . . 7

2 Introduction . . . 8
2.1 Purpose . . . 8
2.2 Scope . . . 8
2.3 Definitions, Acronyms, Abbreviations . . . 8
2.3.1 Definition . . . 8
2.3.2 Acronyms . . . 9
2.3.3 Abbreviations . . . 10
2.4 Revision History . . . 10

3 Implemented Functionalities and Requirements . . . 11
3.1 Product Functions . . . 11
3.1.1 Requirements . . . 11

4 Adopted Development Software . . . 14
4.1 Framework . . . 14
4.1.1 Frontend . . . 14
4.1.1.1 React . . . 14
4.1.2 Backend . . . 14
4.1.2.1 Spring Boot . . . 14
4.1.2.2 Lucene . . . 14
4.1.2.3 Firebase . . . 15
4.1.3 Data layer . . . 15
4.1.3.1 JPA . . . 15
4.1.3.2 Hibernate . . . 15
4.2 Tools . . . 15
4.2.0.1 MariaDB . . . 15
4.2.0.2 Traefik . . . 15
4.3 Languages . . . 16

5 Source Code Structure . . . 17
5.1 Code Organization . . . 17
5.2 Implemented Api Call . . . 18

6 Testing Strategy . . . 19
6.1 Unit Testing . . . 19
6.2 Integration Testing . . . 19

7 Installation Guide . . . 20
7.1 Prerequisites . . . 20
7.2 Backend Setup . . . 20
7.3 Frontend Setup . . . 21
7.4 Final Steps . . . 21

8 Effort Spent **22**

9 References **23**
9.1 Used Tools 23
9.2 Reference Tools 23

A Assignement RDD AY 2024-2025 . . . . . **24**

0 List of Figures

1 Code Source 17
2 Example of API Call visualization in Swagger 18
3 Positive response 18
4 Negative responses 18
5 Coverage result 19

0 List of Tables

4 ITD Acronyms 10
5 RASD Abbreviations 10
6 Document Revision History 10
7 Effort - Lorenzo Ricci 22
8 Effort - Matteo Giovanni Paoli 22
9 Effort - Samuele Grisoni 22

1 Document Information

Version	Modification Details
v1.0: 02-02-2025	First version.
v1.1: 09-02-2025	Modifications: • Updated section 4.2 by writing a new paragraph to explain Traefik • Added 5.2 illustrating the swagger documentation

2 Introduction

2.1 Purpose

The purpose of the Student&Company (S&C) platform is to enable students to enroll into internships that will enhance their education and strengthen their CVs, while letting companies publish internship offers and select the best candidates through interviews. More over, S&C allow students' universities to monitor each of their students' progress and intervene if needed. The platform support and aid the users throughout the entire process by provide suggestion to the uploaded CVs and internship offers, automatically matches students and companies thanks to a proprietary algorithm, manage the distribution and collection of interviews and provides a space for filing and resolving complaints. The reader can find more information about the platform in the RASD document. In the remaining part of this chapter we will present a summary of the technical choices made for the creation of the platform and different bullet point lists and table including the Goals that we are trying to accomplish with this software and the Definition, Acronyms, Abbreviations used in this document.

2.2 Scope

This document, Implementation and Test Document (ITD), provides a comprehensive description of the implementation and testing phases of the S&C platform. Specifically, it focuses on the functionalities developed, the adopted frameworks, and the structure of the source code. Additionally, it includes a detailed testing strategy, covering the procedures, tools, and methodologies used during the development process. This document also serves as a guide for installing and running the platform, offering installation instructions and addressing any prerequisites or potential issues. The effort spent by the team members is also summarized to provide insight into the workload distribution.

2.3 Definitions, Acronyms, Abbreviations

This section provides definitions and explanations of the terms, acronyms, and abbreviations used throughout the document, making it easier for readers to understand and reference them.

2.3.1 Definition

The definition shared between this document and the RASD document are reported in the following list:

- **University**: A university that is registered on the S&C platform.
- **Company**: A company that is registered on the S&C platform.
- **Student**: A person who is currently enrolled in a University and is registered on the S&C platform.
- **User**: Any registered entity on the S&C platform.
- **Internship Offer**: The offer of an opportunity to enroll in an internship provided by a Company. The offer remains active on the platform indefinitely until the publishing Company removes it
- **Participant**:A Participant is an entity that interacts with the platform for the purpose of find or offering an Internship Position Offer, like Students and Companies
- **Recommendation Process**: The process of matching a Student with an Internship offered by a Company based on the Student's CV and the Internship's requirements made by the S&C platform.

- **Recommendation/Match**: The result of the Recommendation Process. It is the match between a Student and an Internship.
- **Spontaneous Application**: The process of a Student spontaneously applying for an Internship that was not matched through the Recommendation Process.
- **Interview**: The process of evaluating a Student's application for an Internship done by a Company through the S&C platform.
- **Feedback**: Information provided by Participant to the S&C platform to improve the Recommendation Process.
- **Internship Position Offer**: The formal offer of an internship position presented to a student who has successfully passed the Interview, who can decide to accept or reject it.
- **Suggestion**: Information provided by the S&C platform to Participant to improve their CVs and Internship descriptions.
- **Confirmed Internship**: An Internship that has been accepted by the Student and the offering Company.
- **Ongoing Internship**: A internship that is currently in progress. All Ongoing Internships are Confirmed Internships, but the vice versa is not always true.
- **Complaint**: A report of a problem or issue that a Student or Company has with an Ongoing Internship. It can be published on the platform and handled by the University.
- **Confirmed Match**: A match that has been accepted by both a Student and a Company.
- **Rejected Match**: A match that has been refused by either a Student or a Company.
- **Pending Match**: A match that has been accepted only by a Student or a Company, waiting for a response from the other party.
- **Unaccepted Match**: A match that has been refused by either a Student or a Company.

2.3.2 Acronyms

The following acronyms are used throughout the document:

Acronym	Definition
ITD	Implementation and Test Document
CV	Curriculum Vitae
UI	User Interface
UX	User Experience
DB	Database
API	Application Programming Interface
ORM	Object-Relational Mapping
DBMS	Database Management System
SPA	Single Page Application
DMZ	Demilitarized Zone
JPA	Java Persistence API
JS	JavaScript
JWT	JSON Web Token
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
SQL	Structured Query Language
CRUD	Create, Read, Update, Delete

Table 4: ITD Acronyms

2.3.3 Abbreviations

The following abbreviations are used throughout the document:

Abbreviations	Definition
S&C	Students&Companies

Table 5: RASD Abbreviations

2.4 Revision History

Revised on	Version	Description
2-2-2025	1.0	Initial release of the document
09-2-2025	1.1	Insert paragraphs about API endpoint and Traefik

Table 6: Document Revision History

3 Implemented Functionalities and Requirements

3.1 Product Functions

This section outlines the essential functionalities and detailed requirements of the platform, structured to support the key objectives defined in the scope of the product.

1. **User Management**: The platform allows Students, Companies, and Universities to register and login. It also provides Students with the ability to upload and modify their CVs, and Companies with the ability to view and manage their Internships Offer.
2. **Internship Creation and Management**: Companies can create, publish, and manage Internship Offers on the platform. They define details such as job description, requirements, location, and benefits.
3. **Student Application Process**: Students can browse available Internships and apply to Internships either through automatic matching or by submitting Spontaneous Applications. They can also track the status of their Applications throughout the process.
4. **Automated Recommendations**: The platform matches Students with suitable Internships based on their CVs and the specific requirements set by Companies. Once a match is found, both Students and Companies are notified, and they can accept or reject the Recommendation.
5. **Interview Management**: Companies can create and assign Interviews to Students, which include closed and open questions to assess their suitability for an Internship. Both Students and Companies can track the Interview progress, and Companies can evaluate Student responses. Companies can also select among students who have passed the interview those to whom they will propose an Internship Position Offer.
6. **Interview Reassignment**: Companies can reassign Interviews question to other Students, allowing them to shorten the time needed to find the right candidate.
7. **Feedback for Improvement**: The platform collects Feedback from Students and Companies to improve the Recommendation Process by dynamically change the Recommendation Process
8. **Complaint Management**: Students and Companies can publish Complaints about Ongoing Internships, notifying Universities.
9. **Notification System**: Notifications are sent to Students, Companies, and Universities when relevant events occur, such as new Internships, founded matches, Interview assignments, Internship Position Offers, Sign-up confirmation or Communications.

3.1.1 Requirements

The following requirements have been implemented in the S&C platform:

[R1] The platform shall allow any unregistered students to register by providing personal information and selecting their University.

[R2] The platform shall allow any companies to register by providing company information.

[R3] The platform shall allow any universities to register by providing university information.

[R4] The platform shall allow Users to log in using their email and password.

[R5] The platform shall send notifications to Users when relevant events occur.

[R6] The platform shall allow Companies to create and publish Internship offers specifying details.

[R8] The platform shall provide Students with Matches automatically obtained by the Recommendation Process.

[R9] The platform shall allow Students to view and navigate all available Internships.

[R10] The platform shall enable Students to submit Spontaneous Applications to Internships they choose.

[R11] The platform shall allow Students to submit their CV.

[R12] The platform shall allow Students to modify their CV.

[R13] The platform shall allow Students to monitor the status of their Spontaneous Applications.

[R14] The platform shall allow Students to monitor the status of their Recommendation.

[R15] The platform shall display to Students all the Internships found by the Recommendation Process.

[R16] The platform shall display to Companies all the CVs of Matched Students obtained by the Recommendation Process.

[R17] The platform shall allow Students and Companies to accept a Recommendation.

[R18] The platform shall allow Companies to accept a Spontaneous Application.

[R19] The platform shall start a Selection Process only if both the Company and the Student have accepted the Recommendation.

[R20] The platform shall start a Selection Process only if the Company has accepted the Spontaneous Application.

[R21] The platform shall allow Companies to create Interviews.

[R22] The platform shall allow Companies to submit Interviews to Students they have initiated a Selection Process with.

[R23] The platform shall allow Students to answer Interview questions and submit them.

[R24] The platform shall allow Companies to manually evaluate Interview submissions.

[R25] The platform shall allow Students and Companies to monitor the status of their Interviews.

[R26] The platform shall enable Companies to complete the Interview process by submitting the final outcome to each candidate.

[R27] The platform shall enable Companies to send an Internship Position Offer to a Student only if he previously passed the relative Interview.

[R28] The platform shall enable Students to accept or reject an Internship Position Offer sent by a Company only if he previously passed the relative Interview.

[R29] The platform shall collect Feedback from both Students and Companies regarding the Recommendation Process.

[R32] The platform shall allow registered Universities to access and monitor Internship Communications related to their Students.

[R33] The platform shall provide a dedicated space for Students and Companies to exchange Communications about the current status of an Ongoing Internship.

[R35] The platform shall enable companies to assign a set of interview questions to students, allowing the same set of questions to be reassigned to multiple one.

The following requirements have **not** been implemented in the S&C platform:

[R7] The platform shall allow Companies to terminate their Internship offers at their own discretion

[R30] The platform shall provide Suggestions to Students on improving their CVs

[R31] The platform shall provide Suggestions to Companies on improving Internship descriptions.

[R35] The platform shall allow registered Universities to handle Complaints and to interrupt an Internship at their own discretion.

4 Adopted Development Software

Different frameworks, technologies, and languages have been used to implement the S&C platform. The following sections provide an overview of the main frameworks used for the backend, frontend, and database layers. Each section includes a brief description of the framework, its main features, and the reasons for choosing it.

4.1 Framework

4.1.1 Frontend

React React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. It is a declarative, efficient, and flexible library that allows developers to compose complex UIs from small and isolated pieces of code called "components"
React was chosen for the frontend layer of the S&C platform because of its simplicity, flexibility, and performance. It allows for the creation of reusable UI components, which is essential for building a complex web application like ours. React also provides a virtual DOM, which improves performance by updating only the necessary parts of the UI when the application state changes. This feature is particularly useful for the S&C platform, where real-time updates are required to display different aspect of the application like newly founded matches or recently received notifications. React also has a large and active community, which ensures good support, documentation, and a wide range of third-party libraries and tools that can be used to enhance the development process. In particular, libraries like Framer Motion and Material-UI have been used to create smooth animations, fluent transitions and consistent and responsive UI design.

4.1.2 Backend

Spring Boot Spring is a popular open-source framework for building enterprise Java applications. It provides comprehensive infrastructure support for developing Java applications, including configuration management, dependency injection, and aspect-oriented programming. Spring was chosen mainly for its modularity that allow developers to use only the parts of the framework they need, as well as for its ease of use and large community support. It also integrates well with other Java technologies, such as Hibernate and JPA. Spring Boot also simplifies the development process by providing a set of defaults and conventions, which reduces the amount of boilerplate code that developers need to write. This allows the team to focus on the business logic of the application, rather than dealing with low-level configuration details. The Spring Data JPA module was used to interact with the database, providing a high-level abstraction over the underlying SQL queries. This allows developers to write database queries using Java objects and annotations, rather than raw SQL statements. Spring Boot web was used to create RESTful APIs that can be consumed by the frontend layer of the application. These APIs are used to perform CRUD operations on the database such as creating, reading, updating, and deleting data.

Lucene Lucene is a high-performance, full-featured text search engine library written in Java. It is widely used in information retrieval and text mining applications, such as web search engines, document management systems, and e-commerce platforms. Lucene provides a rich set of features for indexing, searching, and analyzing text data, including support for full-text search, faceted search, and fuzzy matching. These capabilities were used to implement the matching algorithm that is used to match students with internships. The algorithm is based on the similarity between the student's CV and the internship description requirements. Thanks to the fuzzy matching feature of Lucene, as well as it's speed

and accuracy, the algorithm is able to provide a list of the most suitable internships for each student even if the CV and the internship description are not an exact word-by-word match.

Firebase Firebase is a platform developed by Google for creating mobile and web applications. It provides a variety of services, including a real-time database, authentication, cloud storage, and hosting. It allows developers to build high-quality applications quickly and efficiently. Firebase was chosen for the backend layer of the S&C platform because of its authentication capability, handling the users account confirmation, sign-up and login and the ability to send them notifications with the Firebase Cloud Messaging service when relevant events occur.

4.1.3 Data layer

JPA Java Persistence API (JPA) is a Java specification for managing relational data in Java applications. It provides a set of standard interfaces and annotations for mapping Java objects to database tables and vice versa. JPA is part of the Java EE platform and is implemented by various ORM (Object-Relational Mapping) frameworks, such as Hibernate, EclipseLink, and OpenJPA. JPA was chosen for the database layer of the S&C platform because of its ease of use, flexibility, and compatibility with other Java technologies and framework, in particular Spring. Moreover JPA allows developers to write database queries using Java objects and annotations, rather than raw SQL statements. This makes the code more readable, maintainable, and less error-prone while providing a high-level abstraction over the underlying SQL queries, which simplifies the development process and reduces the amount of boilerplate code that developers need to write.

Hibernate Hibernate is a high-performance, object-relational mapping (ORM) framework for Java that handles the mapping between Java classes and database tables, as well as the generation of SQL queries and the management of database connections. Hibernate was chosen for the database layer of the S&C platform because of its ease of use, flexibility, and compatibility with other Java technologies as well as the rich set of features such as caching, transaction management, and query optimization, which improve the performance and scalability of the application.
Hibernate is widely used in enterprise Java applications and has a large and active community, which ensures good support, documentation, and a wide range of third-party libraries and tools that can be used to enhance the development process.

4.2 Tools

MariaDB MariaDB is an open-source relational database management system (DBMS) that is compatible with MySQL. It is widely used in web applications, e-commerce platforms, and content management systems. MariaDB was mainly chosen for its compatibility with Hibernate and JPA, which simplifies the integration with the backend layer of the S&C platform as well as for its open-source nature, which allows developers to use it freely without any licensing costs.

Traefik Traefik is a modern HTTP reverse proxy and load balancer that is widely used in microservices architectures. Traefik was chosen for the deployment layer of the S&C platform because of its simplicity, flexibility, and compatibility with Docker. It allows developers to easily route traffic to different services, manage SSL certificates, and scale the application horizontally by adding or removing instances of the backend layer while also providing a web-based dashboard that allows developers to monitor the status of the application, view logs, and configure settings. We mainly used Traefik to manage the routing of the different services that compose the S&C platform, each of them running in a separate Docker container, by blocking private call without a valid JWT token, and redirecting authentication requests from the backend to the authentication provider.

4.3 Languages

The following languages have been used to implement the S&C platform:

- **Java**: Java is a general-purpose programming language that is widely used in enterprise applications, web development, and mobile applications. It is mainly known for its portability, which allows developers to write code once and run it on any platform that supports Java and its virtual machine. Java was chosen for the backend layer of the S&C platform because of its performance, scalability, and compatibility with other Java technologies and frameworks, such as Spring and Hibernate, as well as the team members' familiarity with the language.
- **JavaScript**: JavaScript is a high-level, interpreted programming language that is widely used in web development. It is mainly known for its ability to create interactive and dynamic web pages. JavaScript was chosen for the frontend layer of the S&C platform because of its flexibility, performance, and compatibility with modern web browsers, as well as the large and active community that provides good support, documentation, and a wide range of third-party frameworks and libraries like React.

5 Source Code Structure

5.1 Code Organization

The image illustrates the structured organization of the source code for the S&C platform. It highlights the distribution of functionalities across different packages, ensuring modularity and maintainability.

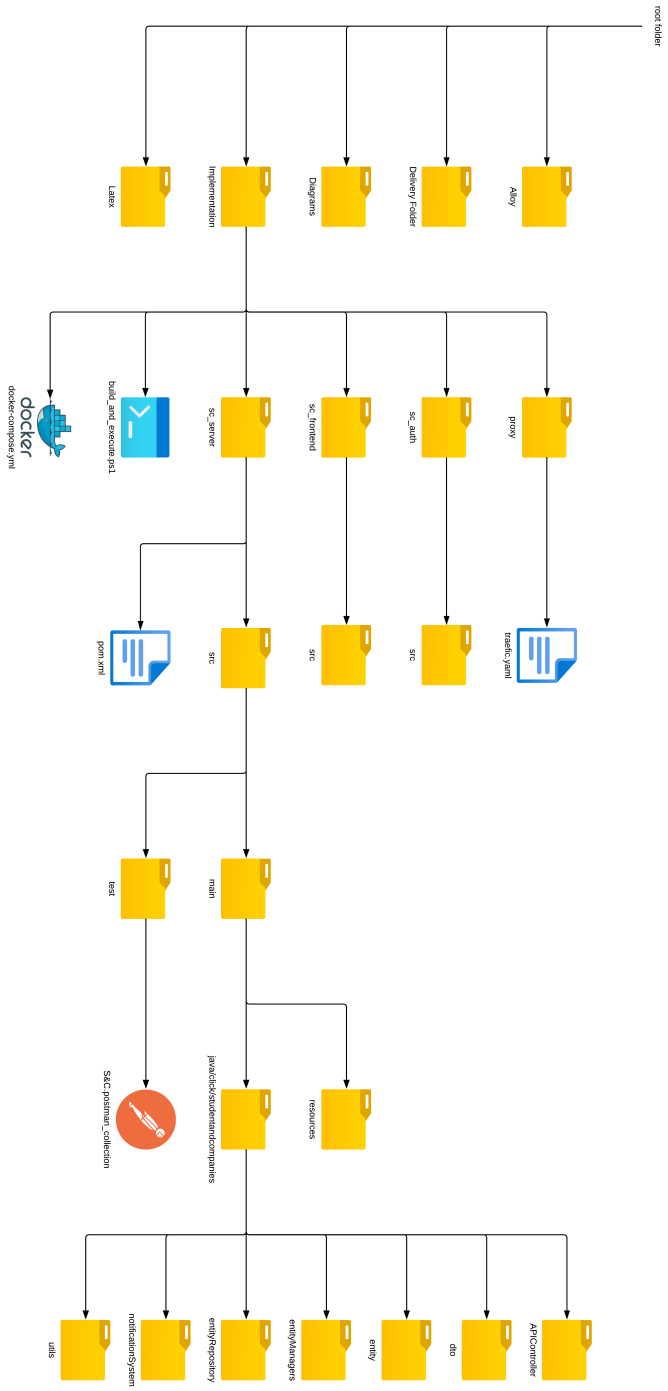


Figure 1: Code Source

5.2 Implemented Api Call

The prototype of the S&C platform includes a set of RESTful APIs used to interact with the frontend layer of the application. Most of the API endpoints were already discussed in the DD document, but some were changed, added, or removed during the implementation phase.
A Swagger documentation page was created to provide a detailed description of each API endpoint, including request and response parameters, request body, HTTP methods, and authentication requirements. Users can access the Swagger documentation by visiting the following URL: http://localhost:8443/swagger-ui/index.html or by using the YAML file located in the Implementation folder of the repository in a swagger editor. Note that the YAML file contains additional comments and descriptions that are not visible in the Swagger documentation available via the URL because they were added by hand.
Below are some images showcasing the main API endpoints implemented during the development phase and how they appear in the Swagger documentation.

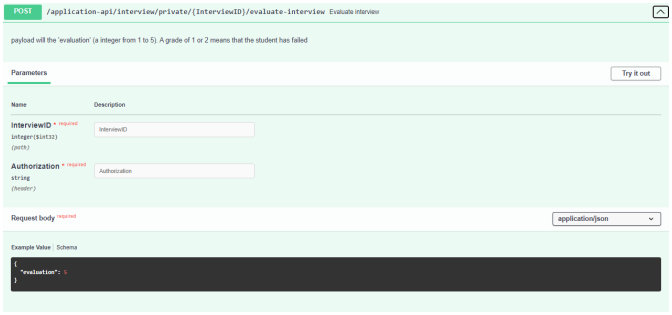


Figure 2: Example of API Call visualization in Swagger

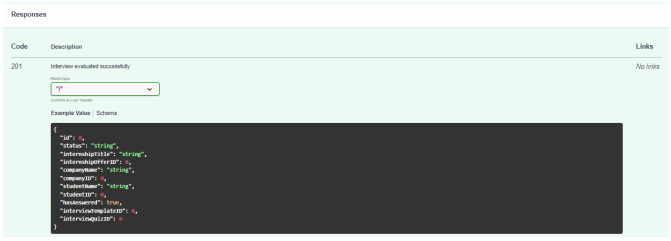


Figure 3: Positive response

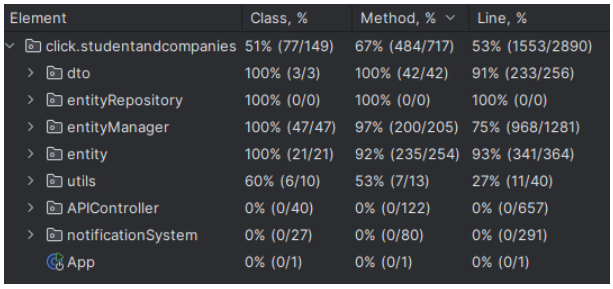
Figure 4: Negative responses

6 Testing Strategy

The testing strategy for the S&C platform is based on a combination of manual and automated testing techniques. Manual testing is performed by the development team to verify the correctness of the application's functionality, user interface, and user experience. Automated testing is performed using testing frameworks and tools to verify the correctness of the application's business logic, data integrity, and performance. The following sections will provide an overview of the testing strategy using during the development and the integration phase.

6.1 Unit Testing

For the development testing phase we mainly used JUnit and Mockito to test the backend layer of the S&C platform. JUnit is a popular testing framework for Java that is widely used in enterprise applications, web development, and mobile applications. Mockito is a mocking framework for Java that allows developers to create mock objects for testing purposes. It is mainly used to isolate the code under test from its dependencies, such as external services, databases, and APIs.
Following what we already describe in the DD document, during this phase we mainly focused on testing the business logic of the application and the data integrity. This is visible in the test classes that we created for each of the main components of the S&C platform which mainly test the CRUD operations of the different entities while postponing the test of the API end-points and the DTO classes to the integration testing phase.



Element	Class, %	Method, %	Line, %
click.studentandcompanies	51% (77/149)	67% (484/717)	53% (1553/2890)
dto	100% (3/3)	100% (42/42)	91% (233/256)
entityRepository	100% (0/0)	100% (0/0)	100% (0/0)
entityManager	100% (47/47)	97% (200/205)	75% (968/1281)
entity	100% (21/21)	92% (235/254)	93% (341/364)
utils	60% (6/10)	53% (7/13)	27% (11/40)
APIController	0% (0/40)	0% (0/122)	0% (0/657)
notificationSystem	0% (0/27)	0% (0/80)	0% (0/291)
App	0% (0/1)	0% (0/1)	0% (0/1)

Figure 5: Coverage result

6.2 Integration Testing

For the integration testing phase we mainly used Postman as a "Proxy Driver" to test the RESTful APIs created with Spring Boot following the same strategy describe in the DD document at the 3° testing step. The main goal of this phase is to verify that the different components of the S&C platform work together correctly and that the data flow between them is consistent and reliable. The integration testing phase is performed by the development team to identify and fix any issues related to the interaction between the call made by the frontend and the computation made by the backend and the database. We created more than 30 different API call that were later used in Postman unit suite, that allowed us to test all the different functionalities of the platform, from the user registration to the internship creation, from the recommendation process to the interview management.
Postman allowed us to randomize request parameters and body content to test the system's robustness and evaluate different edge cases that could arise during normal platform usage. Examples include creating an internship with a missing field, accepting a recommendation that has already been approved, or attempting to fetch a non-existent internship and much more. The integration testing phase is essential to ensure that the S&C platform works as expected, all the functionalities are implemented correctly, and the right exceptions are thrown when the user tries to perform an invalid operation.

7 Installation Guide

This section provides the necessary steps to install and run the S&C platform.
For a more detailed installation guide, please refer to the README file available in the project's GitHub repository.

7.1 Prerequisites

Before proceeding with the installation, ensure you have the following dependencies installed on your system:

- **Docker**: required to run the backend services with the `docker-compose.yml` file.
- **PowerShell** (Windows users only): optional, but recommended to automate the backend startup process.
- **Node.js and npm**: Required to install and run the frontend application.

7.2 Backend Setup

Docker

1. Ensure Docker is installed and running on your system.
2. Open a terminal and navigate to the project root directory.

Spring .jar Files & Containerized Environment

A PowerShell script is provided to create the `.jar` files that will be needed to properly execute the backend services within the docker containers.

1. Open a PowerShell terminal.
2. Navigate to the "Implementation" directory.
3. Execute the script:

```
.\build-and-execute.ps1
```

4. When prompted, press **p** for production mode.
5. Confirm container rebuild by pressing **yes** when asked.

The script is useful as a shorthand to create the java build files, set up the environment variables and build and execute the containers.

Naturally, the same behavior could be obtained by manually compiling the Maven Java project inside the folders `sc_server` and `sc_auth` with the command `mvn clean package` and then building the containers with `docker compose up --build`

7.3 Frontend Setup

1. Ensure that **Node.js** and **npm** are installed on your system.
2. Open a terminal and navigate to the frontend directory.
3. Install dependencies by running:

```
npm install
```

4. Start the frontend with API support:

```
npm run dev:api
```

7.4 Final Steps

Once both the backend and frontend are running, the application should be accessible via the specified local development URL. Make sure both services are properly started before testing the platform.

8 Effort Spent

Lorenzo Ricci

Unit	Hours
Frontend	99
Backend	20
Testing	5
Report	1

Table 7: Effort - Lorenzo Ricci

Matteo Giovanni Paoli

Unit	Hours
Frontend	45
Backend	49
Testing	12
Report	3

Table 8: Effort - Matteo Giovanni Paoli

Samuele Grisoni

Unit	Hours
Frontend	39
Backend	61
Testing	5
Report	2

Table 9: Effort - Samuele Grisoni

9 References

9.1 Used Tools

[1] Lucid Software Inc. *Lucidchart - Online Diagram Software*. Purpose: used to realize all the diagrams along the documents. 2024. URL: https://www.lucidchart.com.

[2] MermaidJS. *Mermaid - Markdownish Syntax for Generating Diagrams and Flowcharts*. Purpose: Used for generating diagrams and flowcharts directly from text or markdown-like syntax. 2024. URL: https://mermaid-js.github.io/.

[3] Meta (Facebook Inc.) *React - A JavaScript Library for Building User Interfaces*. Purpose: used to create the UI prototype. 2024. URL: https://reactjs.org.

[4] MUI Team. *Material-UI: React Components for Faster and Easier Web Development*. Purpose: used to create the UI prototype. 2024. URL: https://mui.com.

[5] PlantUML Team. *PlantUML - Create UML Diagrams from Plain Text*. Purpose: used to create UML diagrams throughout the documentation. 2024. URL: https://plantuml.com/.

9.2 Reference Tools

[6] React Testing Library Team. *React Testing Library - Simple and Complete Testing Utilities for React*. Purpose: used for testing React components. 2024. URL: https://testing-library.com/docs/react-testing-library/intro/.

[7] Framer Team. *Framer Motion - Production-Ready Motion Library for React*. Purpose: used for adding animations to the UI prototype. 2024. URL: https://www.framer.com/motion/.

[8] Spring Team. *Spring Boot - Framework for Building Java Applications*. Purpose: used for back-end development. 2024. URL: https://spring.io/projects/spring-boot.

[9] MariaDB Foundation. *MariaDB - Open Source Database*. Purpose: used as the database for the application. 2024. URL: https://mariadb.org/.

[10] Hibernate Team. *Hibernate - Java Framework for ORM and Persistence*. Purpose: used for ORM and database interaction in the back-end. 2024. URL: https://hibernate.org/.

[11] Firebase Team. *Firebase Authentication - Authenticate and Manage Users*. Purpose: used for user authentication. 2024. URL: https://firebase.google.com/products/auth.

[12] Firebase Team. *Firebase Cloud Messaging - Cross-Platform Messaging Solution*. Purpose: used for sending notifications to devices. 2024. URL: https://firebase.google.com/products/cloud-messaging.

[13] JUnit Team. *JUnit - Testing Framework for Java*. Purpose: used for unit testing Java components. 2024. URL: https://junit.org/.

[14] Mockito Team. *Mockito - Java Mocking Framework*. Purpose: used for creating mock objects in tests. 2024. URL: https://site.mockito.org/.

[15] Mock Service Worker Team. *Mock Service Worker - API Mocking Library*. Purpose: used to mock API calls during testing. 2024. URL: https://mswjs.io/.

A Assignement RDD AY 2024-2025

Students&Companies (S&C) is a platform that helps match university students looking for internships and companies offering them. The platform should ease the matching between students and companies based on:

- the experiences, skills and attitudes of students, as listed in their CVs;
- the projects (application domain, tasks to be performed, relevant adopted technologies-if any etc.) and terms offered by companies (for example, some company might offer paid internships and/or provide both tangible and intangible benefits, such as training, mentorships, etc.).

The platform is used by companies to advertise the internships that they offer, and by students to look for internships. Students can be proactive when they look for internships (i.e., they initiate the process, go through the available internships, etc.). Moreover, the system also has mechanisms to inform students when an internship that might interest them becomes available and can inform companies about the availability of student CVs corresponding to their needs. We refer to this process as "recommendation". Recommendation in S&C can employ mechanisms of various levels of sophistication to match students with internships, from simple keyword searching, to statistical analyses based on the characteristics of students and internships. When suitable recommendations are identified and accepted by the two parties, a contact is established. After a contact is established, a selection process starts. During this process, companies interview students (and collect answers from them, possibly through structured questionnaires) to gauge their fit with the company and the internship. S&C supports this selection process by helping manage (set up, conduct, etc.) interviews and also finalize the selections. To feed statistical analysis applied during recommendation, S&C collects various kinds of information regarding the internships, for example by asking students and companies to provide feedback and suggestions. Moreover, S&C should be able to provide suggestions both to companies and to students regarding how to make their submissions (project descriptions for companies and CVs for students) more appealing for their counterparts. In general, S&C provides interested parties with mechanisms to keep track and monitor the execution and the outcomes of the matchmaking process and of the subsequent internships from the point of view of all interested parties. For example, it provides spaces where interested parties can complain, communicate problems, and provide information about the current status of the ongoing internship. The platform is used by students at different universities. Universities also need to monitor the situation of internships; in particular, they are responsible for handling complaints, especially ones that might require the interruption of the internship.

