ATD Ricci, Paoli, Grisoni

POLITECNICO MILANO 1863

ATD
Acceptance Testing Deliverable

Deliverable:	ATD
Title:	Acceptance Testing Deliverable
Authors:	Lorenzo Ricci, Matteo Giovanni Paoli, Samuele Grisoni
Version:	1.0
Date:	09-02-2025
Download page:	https://github.com/Slaitroc/RicciPaoliGrisoni/
Copyright:	Copyright © 2025, Ricci, Paoli, Grisoni – All rights reserved

0 Contents

0 List of Figures

0 List of Tables

1 Introduction

1.1 Tested Project

- **Project Name:**
  - ◇ RojasJinLe ⟶ https://github.com/felixsrp/RojasJinLe
- **Authors:**
  - ◇ Felix Rojas
  - ◇ Tai Le
  - ◇ Wenjie Jin
- **Reference Documents:**
  - ◇ RASD
  - ◇ DD
  - ◇ ITD

1.2 Delivered Prototype

The delivered prototype consists of an Android portable device application that connects to a set of containers responsible for running:

- ◇ the Postgres Database;
- ◇ an SMTP local server for sending email;
- ◇ Adminer: a service to visualize and handle DB data;
- ◇ the backend serving the api endpoints.

1.3 Acronyms, Abbreviations, and Definitions

Here, we present a table that contains some acronyms and abbreviations along with their definitions, which will be used throughout this document or are defined within the context of the application.

ACRONYM	Definition
RASD	Requirements Analysis and Specification Document
ITD	Implementation and Testing Document
ATD	Acceptance Test Document
ECT	Edge Case Test
Anomaly	Unexpected behavior of the application
Note	Peculiarities of the functionality and/or design choices
Skill/Tag	A keyword that describes an ability of a student or a requirement of a job offer, used in the matching algorithm
Assigment	The document that contains the requirements of the project described in a non-formal way

Table 3: Acronyms

2 Installation

2.1 Prerequisites Installation

The ITD provided by the group under analysis properly covers the installation guide for the software prerequisites. The software in question is:

- Windows Subsystem for Linux 2 (WSL2):
  a compatibility layer that enables running a full Linux kernel on Windows. It is needed to be able to install and use the next prerequisite;
- Docker Desktop:
  an application that enables running and managing Docker containers on Windows, integrating WSL 2 for improved performance and native Linux compatibility.
- Node.js:
  a JavaScript runtime built on Chrome's V8 engine that allows executing JavaScript code outside the browser, enabling backend development and server-side applications.

Since these are very common tools and pieces of software, widely used by developers, their installation was not necessary, as our systems already had them installed and set up.

2.2 Backend Environment Set Up

In the `backend` directory, there are all the needed files to properly configure the backend services. The "`docker-compose.yaml`" file defines four services:

- Postgres service:
  Database;
- Adminer service:
  Database Manager with UI;
- Mail service:
  SMTP server for sending email in a dev environment;
- API service:
  the service responsible to run the node server derving the business logic api.

As specified by the ITD, the only services that shall run in separated containers are the first three as the api one shall be run locally, ignoring the docker compose setup defined for it.

Installation Steps

The steps we executed once in the main project directory are:

1. Navigate to the `backend` directory:

```
cd backend
```

2. Install the Node.js project dependencies:

```
npm install
```

3. Run the three docker containers:

   ```
   docker compose up -d postgres adminer maildev
   ```

   The `-d` argument lauch the containers in detached mode meaning the terminal will still be available to run other commands (while by default it keeps showing the docker logs)

4. Install api server dependencies:

   ```
   npm install
   ```

5. Execute the npm script to construct the database:

   ```
   npm run migration:run
   ```

6. Seed the database:

   ```
   npm run seed:run:relational
   ```

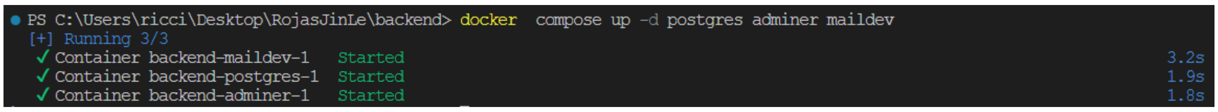
7. Start the server locally on our device:

   ```
   npm run start:dev
   ```

Result

After executing all the previous commands, we have three containers running the following services:

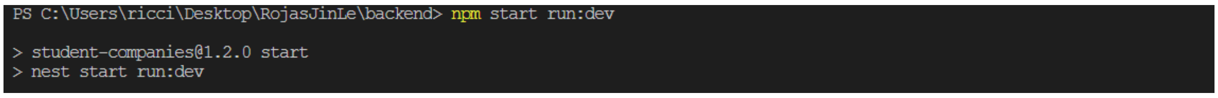
- Postgres service:
  Database;
- Adminer service:
  Database Manager with UI;
- Mail service:
  SMTP server for sending email in a dev environment;



```
● PS C:\Users\ricci\Desktop\RojasJinLe\backend> docker  compose up -d postgres adminer maildev
  [+] Running 3/3
   ✓ Container backend-maildev-1   Started                                                                    3.2s
   ✓ Container backend-postgres-1  Started                                                                    1.9s
   ✓ Container backend-adminer-1   Started                                                                    1.8s
```

Figure 1: Running Containers

And the api server running on directly on our device OS ready to serve on `localhost:3000`:



```
PS C:\Users\ricci\Desktop\RojasJinLe\backend> npm start run:dev

> student-companies@1.2.0 start
> nest start run:dev
```

Figure 2: Running API server

2.3 Frontend Environment Set Up

As the prototype client is an Android application we need an emulator to be able to test it. The ITD specify to use the Android Studio IDE emulator to run the application; so we installed it on our devices. The ITD also provided us with a link to a cloud storage containing the build of the client app that we downloaded and copied into the emulator.

Installation Steps

1. Install Android Studio IDE from its website;
2. Create a new android device emulator and start it;
3. Download the client build file from the cloud storage link;
4. Copy the build file into the running emulator
5. Open the client application;

Result

The result is a running portable android device emulator containing the client application ready for testing.

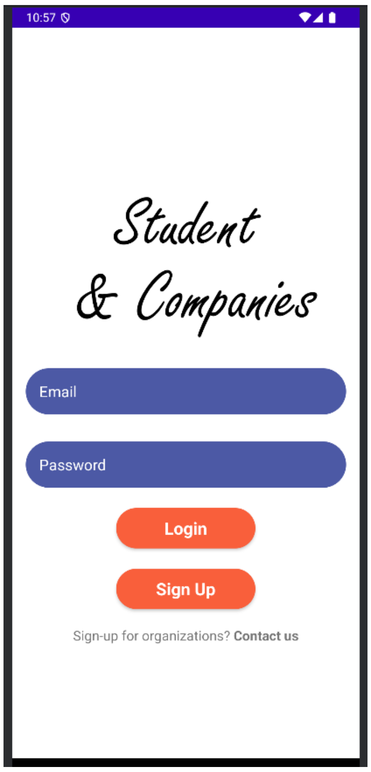


Figure 3: Android Emulator running the client App

3 Tests

In this chapter, we present all the tests performed on the application to ensure its quality, correctness, and compliance with the requirements identified in the RASD and ITD documents.
The first batch of subsections presents, in a structured way, the results of tests conducted on one or more functionalities of the application. Each subsection describes the expected behavior of the functionality before testing it, the general steps the user must follow to interact with it, and the results, explaining how the functionality behaved during typical use. It also includes an evaluation of the functionality, a section on *anomalies*, which we define as unexpected behaviors (marked with an "X" as if they were bullet points), and *notes* which highlighting peculiarities of the functionality and/or design choices that are not necessarily incorrect but worth mentioning (marked with a "☐").
Additionally, each subsection contains one or more Edge Case Tests (ECT), where the functionality is tested in less common but still possible scenarios, such as signing up with a password that is too short or attempting to log in with an incorrect password. Each edge case test follows the same structure as described above.
The second batch of subsections presents all the missing features of the application. Each missing feature is accompanied by a description, an explanation of why it is critical for the application's correctness, and an analysis of which requirements are partially or entirely unsatisfied due to its absence.

Moreover to test some of the functionalities we had to create not only a student account but also a company one. Because of this, and the impossibility of creating a company account in the application we had to manually create the company account with POST request directly sent to the server via Postman. The call was made to the endpoint */api/v1/auth/email/register* with the following body:

```
"email": "company@mail.com",
"password": "password",
"fullName": "CompanyName",
"phoneNumber": "00000000",
"address": "Street 123, Milan, Italy",
"image": "",
"role": 3 // Role is 1 for admin, 2 for student, 3 for company, 4 for university
```

We are aware that the ITD described a procedure to populate the database with the necessary data for the tests. However, we were unable to login with the already create user in the database so we had to create new users manually. As mentioned before the lack of a sign-up functionality for companies forced us to create the company account manually.

3.1 Implemented Feature: Signup & Login

This test focuses on the registration process. As described in the ITD, the frontend application should only allow users of type student to register their accounts.

3.1.1 Expected Behavior

Students can use the frontend application to create an account and then log in using the previously provided information.

3.1.2 Steps

1. Open client application;

2. Click the "`Sign Up`" button;
3. Insert requested information (Name, School Email, University, Mobile, Password, City);
4. Upload CV file through device's file system integration;
5. Add skills keywords;
6. Click "`Signup`" button;
7. Insert the newly created user's email and password;
8. Click the "`Login` button.

3.1.3 Result

`SUCCESS`

- **Description**
  At the end of the process, the student is successfully registered on the platform and can log in to access his/her personal pages.
- **Anomalies & Notes**
  No anomalies were observed at the end of this process.
  - ☐ The implementation documentation states that the app does not validate user input, assuming it is correct, but it does not specify what constitutes valid input.

3.1.4 Edge Case Tests

◇ **Email Duplication**

- **ECT - Description**
  We attempted to register a new user on a different device using an email that was already in use. The client application did not provide any error message and appeared to allow the registration.

  For further testing, we directly executed the API call to `/api/v1/auth/email/register` using the same email. The response body returned:

  ```
  "status": 422,
  "errors": {
          "email": "emailAlreadyExists"
          }
  ```

  This behavior highlights a flawed frontend-backend integration. Checking the database using the Adminer service confirmed that no new user was registered, nor was the existing user with the same email updated.
- **ECT - Anomalies & Notes**
  - X The frontend appears to allow duplicate email registrations without displaying an error message.
  - ☐ The server correctly prevents the registration of a new user with an email that is already in use.

◇ **Empty registration input fields**

- **ECT - Description**
  We attempted to register a new user while leaving all fields empty except for the email and password. The client application displayed an error message requesting the CV file. After providing the file and attempting registration again, the client application appeared to allow the process to proceed.

  Checking the database no new user were created. For further testing we execute the call to the API endpoint `/api/v1/auth/email/register` that providing only email, password, cv and user type in the body. The response was:

  ```
  "status": 422,
  "errors": {
          "fullName": "fullName should not be empty"
          }
  ```

- **ECT - Anomalies & Notes**
  - X The frontend appears to allow the registrations with most all the fields empty except for email, passowrd and CV without displaying an error message.
  - ☐ The server correctly prevents the registration of a new user with some required field recognized as empty.

3.2 Implemented Feature: Authentication System

This test verifies the correctness of the authentication mechanism, which should generate an authenticated user session token. This token is required to successfully make API calls that require authentication and authorization to access restricted resources. To test this feature, we used direct API calls.

3.2.1 Expected Behavior

Upon successful login, the system should generate and provide an authenticated session token (access token and refresh token). API calls requiring authentication should return a valid response only if a valid session token is included in the Authorization header. If an API request is made without an authentication token or with an invalid/expired token, the system should return an appropriate authentication error response. Logging out should invalidate the session token, preventing further authenticated API calls without re-authentication.

3.2.2 Steps

First phase:

1. API call to `/api/v1/auth/email/login` to obtain token and refresh token
2. API calls, providing the token as the Authorization header, to endpoints that require it to receive a positive response.
3. API calls, not providing the token as the Authorization header, to endpoints that require it, expecting a negative response.

Second Phase:

1. Login with a student profile;
2. Logout.

3.2.3 Result

`FAILED`

- **Description**
  The user login request correctly returns the access token and refresh token in the response body, along with other relevant user information. When executing a set of API requests that require authentication and providing the token, the expected results are successfully obtained. However, executing the same set of calls without providing the token still allows them to execute without errors, indicating that authentication enforcement is not properly implemented.
  List of tested API calls:
  - `/api/v1/feedbacks` - create a new feedback
  - `/api/v1/internships` - create a new internship
  - `/api/v1/applications` - apply to an internship
  - `/api/v1/applications/:companyID` - update application status
  - `/api/v1/interviews` - send interview to student
  - `/api/v1/questionnaires` - send questionnaire to student
  - `/api/v1/complaints` - create a new complaint
- **Anomalies & Notes**
  First Phase:
  - X API endpoints that require authentication process requests even without the Authorization token, allowing unauthorized access to restricted functionalities.
  - X The ability to create, update, or retrieve sensitive data without authentication exposes the system to potential security vulnerabilities, including data breaches and unauthorized modifications.
  - ☐ The issue affects multiple high-impact endpoints, including internship applications, interview management, feedback submission, and complaints, which should only be accessible to authenticated users

  Second Phase
  - X After the user logs out, swiping to the right on an Android device allows returning to the previous pages, indicating that the session has not been invalidated.

3.3 Implemented Feature: CV Upload & Profile Management

This test concerns the CV upload phase and the profile management section within the frontend application. The ITD specifies that the profile management functionality is not fully working, but since it is still considered an implemented feature, it will be tested.

3.3.1 Expected Behavior

The student should be able to upload a CV to the platform and view personal data in the corresponding section of the client application.

3.3.2 Steps

1. Register as a student;
2. Upload the CV using the designated input field during registration;
3. No additional steps available for profile management.

3.3.3 Result

`PARTIAL SUCCESS`

- **Description**
  The client application does not include a profile management section, making it impossible to test this functionality. A student can upload a CV during registration, but there is no option to modify or update it later, as no dedicated frontend section exists for this purpose.

- **Anomalies & Notes**

  - ✗ The client application lacks a profile management section, preventing users from viewing or editing their personal data.
  - ✗ Once uploaded during registration, the CV cannot be modified or updated, as no frontend functionality exists for this.
  - ☐ The CV upload feature functions correctly at the time of registration.

3.3.4 Edge Case Tests

◇ **CV File Format Handling**

- **ECT - Description**
  The CV column in the user table of the database by default contains a link to a template PDF. However, even when a user uploads a CV file, the database does not seem to handle it correctly, as the field remains empty.
- **ECT - Anomalies & Notes**

  - ✗ The uploaded CV does not appear in the user table, suggesting improper handling.
  - ☐ Further investigation is needed to determine if the CV file is uploaded but not linked, or if the upload process itself is failing.

3.4 Implemented Feature: Uploading Internship

This test concerns the internship creation phase, which a company user executes to publish an internship offer on the platform.

3.4.1 Expected Behavior

The company should be able to access the frontend section that allows internship creation, enter the required data, and submit it to the server. Once submitted, the internship should become available for all students on the dashboard page.

3.4.2 Steps

1. Company clicks the `Add Internship` button;
2. Company inserts the required data;
3. Company clicks the `Submit`button to send the internship data to the server;
4. The student views the newly created internship on the dashboard page.

3.4.3 Result

PASSED

- **Description** The internship creation process works as expected, and students can successfully view newly created internships in the dashboard.
- **Anomalies & Notes**
  No anomalies were noticed during the process.
  - ☐ The internship includes a location field, which is displayed in the client application. However, the company cannot modify this field during the creation phase, as it is inherited from the company's location set during registration.
  - Since the ITD specifies that there is no validation on the input provided, we will not include edge case tests for this feature.

3.5 Implemented Feature: Search Internship

3.5.1 Expected Behavior

The internship search functionality should allow users to filter available internships based on specific criteria. The filtering options should include:

- The ability to search for internships in a specific location.
- The ability to filter by title, required skills, optional skills, company name, and description and other relevant text information.

The results should only display internships matching the selected filters, and filtering should exclude non-matching entries.

3.5.2 Steps

1. Perform a search using location-based filtering within the frontend application.
2. Perform a search using internship data-based filtering, entering different values such as title, skills, company name, and description.
3. Verify whether the returned results match the expected filtered entries.

3.5.3 Result

PARTIAL SUCCESS

- **Description**
  The internship search functionality partially works, as filtering by title, required skills, and optional skills functions correctly. However, filtering by location does not yield meaningful results, and searching using company name or description fails entirely.
- **Anomalies & Notes**
  - X Since internships inherit their location from the company that created them, searching by location does not produce relevant results.
  - X Filtering by company name or internship description does not work, meaning users cannot refine searches based on these fields.
  - ☐ Filtering by internship title and skills is functional.
  - ☐ Since there are no explicit API calls for internship search, the filtering mechanism seems to be implemented directly in the frontend application.

3.5.4 Edge Case Tests

◇ **Filtering by Non-Existent Location**

- **ECT - Description**
  We attempted to filter internships by entering a location that does not exist in the dataset. The search returned the same results as no filtering, indicating that location filtering is ineffective.
- **ECT - Anomalies & Notes**

  X The same results appear regardless of the selected location.
  ☐ The same result is obtained using actual internship location, meaning the filtering is not working.

◇ **Filtering by Partial Internship Description**

- **ECT - Description**
  We tested searching using a keyword contained within an internship description. The filtering failed, returning no results, even when internships matching the keyword were present in the dataset.
- **ECT - Anomalies & Notes**
  X Filtering by description does not work.

◇ **Filtering by Company Name**

- **ECT - Description**
  We searched for internships using the exact name of a company that had published internships. The filtering failed, returning the same results as no filtering.
- **ECT - Anomalies & Notes**
  X Filtering by company name does not work.
  ☐ nota

3.6 Implemented Feature: Pending Application

The test concerns the ability of companies to monitor their list of pending applications.

3.6.1 Expected Behavior

Companies should be able to review the list of candidates for their internship, along with their respective CVs, and select students who meet the requirements.

3.6.2 Steps

1. Register a new Company
2. Login with new Company
3. Create 2 Internships by pressing on `Add Internships` and pressing on submit after providing the required informations
4. Logout
5. Register a new Student
6. Login with new Student

7. Open Internship1 and press `Apply`
8. Navigate to `Recommendation`, open Internship2 and press `Apply`
9. Logout and login with Company
10. Press on the relative Internship to see the list of applications
11. Press on the `CV` button

3.6.3 Result

`PARTIAL SUCCESS`

- **Description**
  The Company successfully retrieves the list of applications for each of his internships but is not able to view the candidate CV, as a generic sample PDF is shown instead
- **Anomalies & Notes**
  - ☐ There is no difference on the company side between a spontaneous application and a recommendation.
  - ☐ The motivation text application is not shown to the Company at any point in the execution.
  - X In order to test this feature, it has been necessary to use postman to create a company account
  - X When a company tries to visualize the applicant's CV, it is forwarded to a link pointing to a sample PDF, even if the student correctly upload it during registration.

3.7 Implemented Feature: Recommendation Process

3.7.1 Expected Behavior

The platform should use the recommendation algorithm to suggest internships that students might be interested in and to recommend students who match internship requirements to companies.

3.7.2 Steps

1. Register a new Company
2. Login with new Company
3. Create an Internship providing skills to be checkd
4. Logout
5. Register a new Student, including skills
6. Login with Student
7. Navigate to `Recommendation` page and apply to Internship providing motivation message
8. Logout and login with Company
9. Press on the relative Internship to see the list of applications

3.7.3 Result

`PARTIAL SUCCESS`

- **Description** The student successfully applies for the internship and sees the corresponding application status as `in pending` while the company has received the application for his Internship. However, in some cases, a critical application crash occurs, preventing the user from accessing their recommendations.

- **Anomalies & Notes**

  ☐ The company is completely passive during this process.

  ☐ The difference between the two recommendation mechanism algorithms is not clear.

  X If the internship has no required skills, it will not be suggested by either of the two recommendation mechanisms.

  X The app may crash when the student opens the recommendations page, and a particular combination of skills has been chosen for an internship.

  X In order to test this feature, it has been necessary to use postman to create a company account

3.7.4 Edge Case Tests

◇ **Internship with various skill matching scenarios**

- **ECT - Description**
  Testing various combinations of required and optional skills in an internship to observe behaviour in the recommendation process.
- **ECT - Anomalies & Notes**

  X The app crashes when an internship has two compatible required skills and one incompatible required skill. Follow this steps to recreate the error:

  X If an internship has no required skills, it does not appear in any recommendations.

  1. Signup as a Student and add the following skills: skill1, skill2, skill3
  2. Logout and login with a company
  3. Create an internship with the following required skills: skill1, skill2, skill4
  4. Logout, login with student and open the recommendations page

3.8 Implemented Feature: Selection Process

The Selection Management feature allows companies to conduct interviews and questionnaires to systematically assess students. For this test, the steps involving the creation of the student and company, as well as the submission of the internship and the corresponding application, were omitted.

3.8.1 Expected Behavior

Companies should be able to initiate the process by providing students with a link to a pre-designed website containing the necessary instructions. Students then should be able to complete the interview or questionnaire as required. Upon submission, the company should be able to review the responses and updates the application status accordingly, determining whether the student has passed or failed.

3.8.2 Steps

1. Login as Company
2. Press Add Questionnaire
3. Logout and Login with Student
4. Navigate to `Applications` page
5. Press on `See Questionnaire` and press on link
6. Logout and Login with Company
7. Press on `Internship`
8. Press on `Answer` and press on `accept`

3.8.3 Result

`PARTIAL SUCCESS`

- **Description** The company can successfully send a questionnaire, including its title and link. The student is able to access the information provided by the company and navigate to the designated link. At any stage during this process, the company retains the ability to accept or reject the application. However, the student cannot see correctly the link for his interview.
- **Anomalies & Notes**
  - X The progression of the selection process differs from the state chart representation in the RASD document. In the implemented version, the company has the ability to reject the student before any submission is made and may also choose not to wait for the student's interview response before making a decision.
  - X In order to test this feature, it has been necessary to use postman to create a company account
  - ☐ Questionnaires or Interviews cannot be opened if a student has already been accepted for an internship

3.8.4 Edge Case Tests

- ◇ **Interview sent only**
  - **ECT - Description**
    The company sends an interview to a candidate student, providing the internship title and the link
  - **ECT - Anomalies & Notes**
    - X The application crashes when a student tries to open an interview sent alone.
- ◇ **Interview and Questionnaire sent**
  - **ECT - Description**
    The company sends both an interview and a questionnaire to the candidate student, including the necessary information, regardless of the order in which they are sent.
  - **ECT - Anomalies & Notes**
    - X If both an interview and a questionnaire are sent, the questionnaire overrides the interview, making the interview inaccessible. This is believed to be a frontend error, as the internship results to be correctly stored in the database

3.9 Implemented Feature: Send Feedback

The Send Feedback feature allows students to provide feedback to the platform with the aim to improve the statistical analysis used during the matching of companies and students. For this test, the steps involving the creation of the student and company were omitted.

3.9.1 Expected Behavior

Students should be able to provide feedback on the platform's matching algorithm, including the ability to rate the accuracy of the matches and provide additional comments. The feedback should be stored in the database and used to improve the matching algorithm.

3.9.2 Steps

1. Login as Student
2. Navigate to the Account page
3. Press on Send Feedback button
4. Fill the form and press on Send

3.9.3 Result

`PARTIAL SUCCESS`

- **Description** While a student can successfully provide feedback, including a rating and comments, the feedback is not used to improve the matching algorithm, as described in the RASD document. The feedback is correctly stored in the database but it is never accessed by the platform while matching the students profile with the internships.
  Moreover, the UI does not explicitly state that the feedback will be used to improve the matching algorithm, as described in the RASD document, but appear as generic feedback form about the platform.
- **Anomalies & Notes**
  - X The feedback is not used to improve the matching algorithm, as described in the RASD document.
  - ☐ The feedback form is not explicitly stated to be used to improve the matching algorithm, as described in the RASD document.

3.9.4 Edge Case Tests

◇ **Feedback without filling all the fields**

- **ECT - Description**
  The student sends a feedback without filling all the fields
- **ECT - Anomalies & Notes**
  - ☐ The frontend correctly handles the case where the student does not fill all the fields but the feedback is also closed deleting the content of the form. This could be improved by displaying an error message to the student without closing the feedback form, avoiding the loss of the content that was already written.
  - X No check are made on the backend or in the database, making it possible to store a feedback without all the fields filled or with a rating that is not between 1 and 5.

3.10 Missing Feature: Notifications

3.10.1 Description

The application does not have a notification system to inform users about important events, such as when an application is accepted, or rejected, the finding of a new job offer or candidate or important deadline such as the expiration of a job offer or an interview. In the current state of the application, users have to manually check the application to see if there are any updates, which is not user-friendly and can lead to users missing important events. In the ITD the notification system was declared as a feature “out of the scope of the project”. However, we believe that this feature is critical for the application’s correctness and usability as it was correctly mentioned both in the RASD and Assigment document. We also notice the presence of“mail service” in the docker-compose file that was not actually used in the application but gives us a hint that the notification system was planned to be implemented with email notifications. This would have still been different that what was described in the RASD document, where the notification system was supposed to be implemented with in-app notifications but it would have been a good alternative to the current state of the application.

3.10.2 Unsatisfied Requirements

In the RASD document the notification system was described entirely in the **R26** requirement. This requirement states that: *the system shall send notifications to users regarding events such as new internship postings, new student applying, application status updates, questionnaire link and interview arrangements*. This requirement is **fully unsatisfied** due to the absence of the whole notification system, both in the front-end and back-end of the application.

3.11 Missing Feature: Complaint

3.11.1 Description

The application does not have a complaint system to allow users to report issue that may arise during an internship. While in the ITD it is correctly mentioned that the handling of the complaint by the University is out of scope for the project, the possibility to report an issue is still a critical feature that is missing in the application and should have been implemented. This feature is critical for the application’s correctness and usability, and more importantly it justified the necessity of a university account to be registered in the application and the mandatory rrequirement for a student to be associated with one of them.
By entirely omitting this feature including the ability to report an issue and forward it to the university, not just the university’s handling of the complaint, the application could be simplified to the point where student accounts could become generic user accounts. This would allow anyone searching for an internship offer to use the platform without needing to be enrolled in a university. However, this approach would differ from the one described in the RASD document, where the application was intended to be used exclusively by students, as detailed in various requirements and the *user characteristics* section.

3.11.2 Unsatisfied Requirements

In the RASD document the complaint system was described in the **R27** requirement, while the **R28** and **R29** requirements described the handling of the complaint by the University. The R27 requirement states that: *The system shall allow students and companies to submit complaints and feedback about internships through the platform*. This requirement is **fully unsatisfied** due to the absence of the whole complaint system, both in the front-end and back-end of the application.

3.12 Missing Feature: Chat Room

3.12.1 Description

The application does not have a chat room system to allow student, company and university to communicate with each other. This feature was described both in the RASD and DD document as a feature that was supposed to be implemented.
Unfortunately, this feature was not implemented in the application, nor was it mentioned in the ITD under the *Excluded Features* section. This would have been a nice-to-have feature that would have improved the usability of the application and even compensated for the lack of a complaint system.

3.12.2 Unsatisfied Requirements

In the RASD document the chat room system was described in the **R31** and **R32** requirements. Both requirements state that a chat room system should be implemented to allow students, companies, and universities to communicate with each other and explaining that the chat room should be used to discuss the complaints allowing the university to handle them better. These requirements are **fully unsatisfied** due to the absence of the whole chat room system, both in the front-end and back-end of the application.

4 Conclusions

In this document, we have presented the results of the tests conducted on the application to ensure its quality, correctness, and compliance with the requirements identified in the RASD and ITD documents. While the application successfully implements several features, critical issues were identified during testing. The most critical issues include the lack of authentication enforcement, which allows unauthorized access to sensitive functionalities, the absence of a complaint system, and application crashes when a student attempts to open an interview sent without a questionnaire. These issues significantly impact the application's usability, security, and reliability, and should be addressed promptly to ensure the application's overall quality.
We appreciate the opportunity to test the application and provide feedback on its functionality and performance. The idea of creating an application instead of a website is innovative and has the potential to improve the user experience with new features that would be impossible to implement on a more traditional platform, such as the use of the camera to take a picture of the students or the CV. We hope that the issues identified in this document will be addressed promptly to ensure the application's quality and reliability.

5 Effort Spent

Member	**Hours**
Lorenzo Ricci	10
Matteo Giovanni Paoli	10
Samuele Grisoni	10

Table 4: Effort

6 References

6.1 Used Tools

[1] Docker Inc. *Docker - Containerization Platform*. Purpose: used for containerization and application deployment. 2024. URL: https://www.docker.com.

[2] Docker Inc. *Docker Desktop - GUI for Docker Management*. Purpose: used for managing Docker containers on Windows and macOS. 2024. URL: https://www.docker.com/products/docker-desktop.

[3] Docker Inc. *Dockerfile - Build Automation for Containers*. Purpose: used to define containerized application environments. 2024. URL: https://docs.docker.com/engine/reference/builder/.

[4] Docker Inc. *Docker Image - Portable Application Packaging*. Purpose: used to create and distribute portable application environments. 2024. URL: https://docs.docker.com/get-started/images/.

[5] Docker Inc. *Docker Compose - Multi-Container Orchestration*. Purpose: used to manage multi-container Docker applications. 2024. URL: https://docs.docker.com/compose/.

[6] Microsoft Corporation. *WSL 2 - Windows Subsystem for Linux 2*. Purpose: used to run Linux distributions on Windows for development. 2024. URL: https://learn.microsoft.com/en-us/windows/wsl/.

[7] OpenJS Foundation. *Node.js - JavaScript Runtime*. Purpose: used for backend development and server-side scripting. 2024. URL: https://nodejs.org.

[8] OpenJS Foundation. *NPM - Node Package Manager*. Purpose: used for managing JavaScript packages and dependencies. 2024. URL: https://www.npmjs.com.

[9] Google LLC. *Android Studio - Official IDE for Android Development*. Purpose: used for developing and testing Android applications. 2024. URL: https://developer.android.com/studio.

[10] Google LLC. *Android - Mobile Operating System*. Purpose: used as the target platform for mobile application development. 2024. URL: https://www.android.com.

[11] Various Providers. *SMTP Server - Simple Mail Transfer Protocol*. Purpose: used for sending and relaying emails in applications. 2024. URL: https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol.

[12] Jakub Vrána. *Adminer - Database Management Tool*. Purpose: used for managing databases via a lightweight web interface. 2024. URL: https://www.adminer.org.

6.2 Reference Tools

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.