POLITECNICO
MILANO 1863

Software Engineering 2
Acceptance Testing Documentation

Author(s): **Shreesh Kumar Jha - 11022306**
Samarth Bhatia - 11059097
Satvik Sharma - 11054680

Academic Year: 2024-2025
Version: 2.0
Release date: 6/02/2025

Contents

Contents **i**

1 Introduction **1**
- 1.1 Purpose . . . 1
- 1.2 Scope . . . 1
- 1.3 Revision History . . . 2
- 1.4 Reference Documents . . . 2
- 1.5 Document Structure . . . 2

2 Implemented Requirements **5**
- 2.1 Overview . . . 5
- 2.2 Dependency Installation . . . 5
  - 2.2.1 Backend Dependencies: . . . 5
  - 2.2.2 Frontend Dependencies . . . 6
  - 2.2.3 Backend Dependencies . . . 6
  - 2.2.4 Frontend Dependencies . . . 7
- 2.3 Backend Installation . . . 7
  - 2.3.1 Cloned the Repository . . . 7
  - 2.3.2 Started the Backend Server . . . 7
  - 2.3.3 Database Initialization . . . 7
- 2.4 Frontend Installation . . . 7
  - 2.4.1 Navigated to the Frontend Directory . . . 8
  - 2.4.2 Installed Frontend Dependencies . . . 8
  - 2.4.3 Ran the Frontend Application . . . 8
  - 2.4.4 Tested API Endpoints using Postman . . . 8
  - 2.4.5 Internship Search & Applications . . . 8
  - 2.4.6 Messaging & Complaints System . . . 8
- 2.5 Problems Encountered & Documentation Incoherences . . . 8
- 2.6 Observations . . . 9

3 Acceptance Test Case **11**

3.1 Acceptance Test Cases 11

3.1.1 Core Requirements (Users) 12

3.1.2 Student Requirements 13

3.1.3 Application Requirements 14

3.1.4 Company Requirements 15

3.1.5 Feedback and Suggestions 16

3.1.6 Database Tests 16

3.1.7 Model Tests 23

3.1.8 Authentication Service Tests 26

3.2 Additional Notes 28

3.3 Areas for Improvement 28

3.3.1 Meta-Testing (Tests for Tests) 28

4 Effort Spent **29**

5 References **31**

5.1 References 31

List of Tables **33**

1 | Introduction

1.1. Purpose

The Students & Companies (S&C) platform was created by a different team as a component of the Software Engineering 2 course, and the Acceptance Test Document (ATD) attempts to verify and methodically test its implementation. This document guarantees that the project satisfies the requirements listed in the Design Document (DD) and the Requirement Analysis and Specification Document (RASD). Verifying essential features, identifying discrepancies, and assessing how well the system satisfies user requirements and expected behaviors are the main goals of the testing process.

1.2. Scope

The Students & Companies (S&C) platform is designed to facilitate internship opportunities for students by connecting them with potential recruiters and enabling universities to monitor student internships. The platform includes the following key functionalities:

- **Internship Lookup for Students:** Depending on their preferences, education, and skill set, students might look for and apply for internships.
- **Visibility for Companies:** Recruiters can post internship openings and match qualified student profiles with them.
- **Selection Process Management:** Employers are able to manage selections, set up interviews, and evaluate student applications.
- **Recommendation System:** Makes individualized internship recommendations using user data and gathered statistics.
- **Communication & Feedback:** Allows for message sharing, problem reporting, and post-internship feedback between students and businesses.
- **University Monitoring:** To guarantee academic compliance, universities might supervise their students' internships.

The S&C platform was developed by:

- Edoardo S. Gribaldo
- Federico Rosa

The GitHub repository for the project can be found here: S&C Repository

1.3. Revision History

Version	Date	Description	Authors
1.0	06 February 2025	Initial Release	Shreesh Kumar Jha, Samarth Bhatia

Table 1.1: Revision History

1.4. Reference Documents

- Reference to Previous Year Student Projects for Structuring the Document
- **RASDv1**
- **ITDv1**
- **DDv1**

1.5. Document Structure

This document is structured as follows:

Installation Setup

- Details the steps followed to install and test the S&C platform.
- Highlights any encountered issues, inconsistencies, or missing documentation.

Acceptance Test Cases

- Describes the test cases executed against the system.

- Maps tests to system requirements and goals as outlined in the `RASDv1`, `DDv1`, and ITDv1.
- Documents observed results, including any failures and potential fixes.

2 | Implemented Requirements

2.1. Overview

We adhered to the installation instructions found in **IT Document (ITD)v1**. The installation procedure was simple because of the comprehensive and well-organized guide. We encountered no problems throughout the installation, in contrast to other intricate setups. Every stage went without a hitch, and the system was successfully and consistently deployed.

2.2. Dependency Installation

2.2.1. Backend Dependencies:

To install the backend, we followed these steps:

Installed Python and Virtual Environment

- Created a virtual environment and activated it using:

```
python3 -m venv env
source env/bin/activate  # For Linux/Mac
env\Scripts\activate  # For Windows
```

Installed Required Python Packages

- Using the `requirements.txt` file from the repository:

```
pip install -r requirements.txt
```

Installed Node.js and npm (For API Calls Testing)

- Installed Node.js (latest LTS version) and verified npm was installed properly:

```
node -v
```

```
npm -v
```

Installed MongoDB as the Database

- Since the backend uses MongoDB, we installed and ran it using:

```
mongod --dbpath=/data/db
```

2.2.2. Frontend Dependencies

The frontend is built using React. The following dependencies were installed:

Installed React and Required Libraries

```
npm install
```

Checked TailwindCSS and Other UI Libraries

- Verified TailwindCSS and Flowbite were installed correctly.

2.2.3. Backend Dependencies

To install the backend, we followed these steps:

Installed Python and Virtual Environment

- Created a virtual environment and activated it using:

```
python3 -m venv env
source env/bin/activate  # For Linux/Mac
env\Scripts\activate  # For Windows
```

Installed Required Python Packages

- Using the `requirements.txt` file from the repository:

```
pip install -r requirements.txt
```

Installed Node.js and npm (For API Calls Testing)

- Installed Node.js (latest LTS version) and verified npm was installed properly:

```
node -v
```

```
npm -v
```

Installed MongoDB as the Database

- Since the backend uses MongoDB, we installed and ran it using:

```
mongod --dbpath=/data/db
```

2.2.4. Frontend Dependencies

The frontend is built using React. The following dependencies were installed:

Installed React and Required Libraries

```
npm install
```

Checked TailwindCSS and Other UI Libraries

- Verified TailwindCSS and Flowbite were installed correctly.

2.3. Backend Installation

2.3.1. Cloned the Repository

```
git clone https://github.com/edogriba/GribaldoRosa.git
cd ITD/backend
```

2.3.2. Started the Backend Server

Activated the virtual environment and ran the server:

```
source env/bin/activate  # For Linux/Mac
python3 run.py
```

2.3.3. Database Initialization

Ran the `createDB.py` script to set up MongoDB collections.

2.4. Frontend Installation

2.4.1. Navigated to the Frontend Directory

```
cd ../frontend
```

2.4.2. Installed Frontend Dependencies

```
npm install
```

2.4.3. Ran the Frontend Application

```
npm start
```

Accessed the Application on Localhost

Opened `http://localhost:3000` in a browser.

Testing Environment Setup

2.4.4. Tested API Endpoints using Postman

- All API requests responded as expected.

Tested User Registration & Login

- Successfully created and logged in as a Student, Recruiter, and Admin.

2.4.5. Internship Search & Applications

- Verified that students could search and apply for internships.

2.4.6. Messaging & Complaints System

- Messages and complaints were successfully created and retrieved.

2.5. Problems Encountered & Documentation Incoherences

- **No Issues Faced During Installation.**

- The provided IT Document (ITD) was clear and accurate, ensuring a smooth setup.
- There were no missing dependencies or undocumented configurations.

2.6. Observations

- Well-structured documentation made installation easy.
- Database initialization was straightforward without requiring manual data imports.
- No additional configurations were needed beyond those in the ITD.

3 | Acceptance Test Case

3.1. Acceptance Test Cases

The **Students & Companies (S&C) platform** requirements are divided into two main categories:

- **Core Requirements**: Fundamental system features that must be implemented and functional.
- **Goal Reaching Requirements**: Additional functionalities mapped to user goals, improving usability and efficiency.

The following sections outline some important test cases, linking each requirement (R1, R2, etc.) from the *RASD* document to its corresponding test. **Overall: 23 Backend Tests.**

3.1.1. Core Requirements (Users)

Requirement	Test Description	Expected Outcome	Actual Outcome	Status
R1	Allow users to sign up.	Users can register successfully.	✓ Works as expected.	✓ Pass
R2	Allow users to fill in profile information when signing up.	Profile data saved successfully.	✓ Works as expected.	✓ Pass
R3	Allow users to log in.	Users can access their accounts.	✓ Works as expected.	✓ Pass
R4	Allow users to log out.	Users are logged out securely.	✓ Works as expected.	✓ Pass
R5	Allow users to update their profile information.	Profile updates are saved.	✓ Works as expected.	✓ Pass
R6	Allow users to examine their own internships.	Users can see the internships they applied for.	✓ Works as expected.	✓ Pass

Table 3.1: Core Requirements Test Cases for Users

3.1.2. Student Requirements

Requirement	**Test Description**	**Expected Outcome**	**Actual Outcome**	**Status**
R7	Allow students to examine open internship positions.	Available internships are listed.	✓ Works as expected.	✓ Pass
R8	Allow students to examine their own applications.	Applications and statuses are visible.	✓ Works as expected.	✓ Pass
R9	Allow students to search for a specific internship position.	Search results match the criteria.	✓ Works as expected.	✓ Pass
R10	Allow students to apply for an internship position.	Application is submitted successfully.	✓ Works as expected.	✓ Pass
R11	Notify students when a suitable internship is opened.	Students receive notifications for relevant internships.	✓ Works as expected.	✓ Pass
R12	List different internship positions aligned with student profiles.	Internships displayed match student skills.	✓ Works as expected.	✓ Pass

Table 3.2: Student Requirements Test Cases

3.1.3. Application Requirements

Requirement	**Test Description**	**Expected Outcome**	**Actual Outcome**	**Status**
R13	Allow students to confirm or refuse an accepted internship offer.	Students can accept/reject offers.	✓ Works as expected.	✓ Pass
R14	Allow companies to request skill assessments and schedule interviews.	Recruiters can request interviews.	✓ Works as expected.	✓ Pass
R15	Allow students to access interview details and links.	Interview details are available in the dashboard.	✓ Works as expected.	✓ Pass
R16	Allow students to see the status of their applications.	Application status updates correctly.	✓ Works as expected.	✓ Pass

Table 3.3: Application Requirements Test Cases

3.1.4. Company Requirements

Requirement	**Test Description**	**Expected Outcome**	**Actual Outcome**	**Status**
R17	Allow companies to open and examine internship positions.	Companies can create and view internships.	✓ Works as expected.	✓ Pass
R18	Allow companies to accept or reject applications.	Recruiters can update application statuses.	✓ Works as expected.	✓ Pass
R19	Allow companies to close internship positions.	Internship status updates to "Closed."	✓ Works as expected.	✓ Pass
R20	Notify companies when a new relevant student profile is available.	Recruiters receive notifications about matching students.	✓ Works as expected.	✓ Pass

Table 3.4: Company Requirements Test Cases

3.1.5. Feedback and Suggestions

Requirement	Test Description	Expected Outcome	Actual Outcome	Status
R24	Allow students and companies to rate the internship experience.	Ratings are recorded successfully.	✓ Works as expected.	✓ Pass
R25	Allow students to provide feedback on the internship experience.	Students can submit feedback.	✓ Works as expected.	✓ Pass
R26	Allow companies to send feedback/news to students.	Students receive feedback notifications.	✓ Works as expected.	✓ Pass
R27	Allow both parties to file complaints about the internship.	Complaints are submitted and reviewed.	✓ Works as expected.	✓ Pass

Table 3.5: Feedback and Suggestions Test Cases

3.1.6. Database Tests

The following tests verify the database operations for different modules in the **Students & Companies (S&C) platform**. These tests check insertion, retrieval, updates, and constraints for key entities.

Assessment Database Tests

Test Case	Description	Expected Outcome	Status
Insert assessment	Add a new assessment record	✓ Successfully inserted	✓ Pass
Invalid insert	Insert invalid assessment data	✗ Raises exception	✓ Pass
Retrieve last assessment by application ID	Fetch last added assessment for an application	✓ Retrieves latest assessment	✓ Pass
Retrieve with invalid application ID	Query non-existent application ID	✗ Returns None	✓ Pass

Table 3.6: Assessment Database Test Cases

Application Database Tests

Test Case	**Description**	**Expected Outcome**	**Status**
Insert application	Add a new application entry	✓ Successfully inserted	✓ Pass
Invalid insert	Insert application with missing fields	✗ Raises exception	✓ Pass
Retrieve application by ID	Fetch an application record	✓ Retrieves correct data	✓ Pass
Retrieve application by student ID	Get applications for a student	✓ Retrieves correct data	✓ Pass
Retrieve application by internship ID	Get applications for an internship	✓ Retrieves correct data	✓ Pass
Update application status	Modify an application status	✓ Status updates successfully	✓ Pass

Table 3.7: Application Database Test Cases

Company Database Tests

Test Case	**Description**	**Expected Outcome**	**Status**
Insert company	Add a new company entry	✓ Successfully inserted	✓ Pass
Duplicate company insert	Try inserting a duplicate company	✗ Raises exception	✓ Pass
Retrieve company by ID	Fetch a company record	✓ Retrieves correct data	✓ Pass
Retrieve company by email	Fetch company details using email	✓ Retrieves correct data	✓ Pass
Update company details	Modify company profile data	✓ Updates successfully	✓ Pass

Table 3.8: Company Database Test Cases

Complaint Database Tests

Test Case	Description	Expected Outcome	Status
Insert complaint	Add a new complaint entry	✓ Successfully inserted	✓ Pass
Invalid insert	Insert complaint with missing fields	✗ Raises exception	✓ Pass
Retrieve complaints by internship ID	Fetch complaints for an internship	✓ Retrieves correct data	✓ Pass
Retrieve complaints for non-existent internship	Query complaints for a non-existent ID	✗ Returns empty list	✓ Pass

Table 3.9: Complaint Database Test Cases

Internship Database Tests

Test Case	Description	Expected Outcome	Status
Insert internship	Add a new internship entry	✓ Successfully inserted	✓ Pass
Invalid insert	Insert internship with missing fields	✗ Raises exception	✓ Pass
Retrieve internship by ID	Fetch internship details	✓ Retrieves correct data	✓ Pass
Retrieve internship by application ID	Fetch internship for a given application	✓ Retrieves correct data	✓ Pass
Retrieve internships by company ID	Fetch all internships posted by a company	✓ Retrieves correct data	✓ Pass
Update internship status	Modify internship state (e.g., ongoing, finished)	✓ Updates successfully	✓ Pass

Table 3.10: Internship Database Test Cases

Internship Position Database Tests

Test Case	Description	Expected Outcome	Status
Insert internship position	Add a new internship listing	✓ Successfully inserted	✓ Pass
Invalid insert	Insert internship position with missing fields	✗ Raises exception	✓ Pass
Retrieve by ID	Fetch internship position details	✓ Retrieves correct data	✓ Pass
Retrieve by company ID	Fetch all internship positions for a company	✓ Retrieves correct data	✓ Pass
Retrieve by program name	Fetch internship positions by program name	✓ Retrieves correct data	✓ Pass
Search with filters	Filter internships by role, location, stipend	✓ Retrieves filtered results	✓ Pass
Update internship position status	Modify internship position state (e.g., Open, Closed)	✓ Updates successfully	✓ Pass

Table 3.11: Internship Position Database Test Cases

Student Database Tests

Test Case	**Description**	**Expected Outcome**	**Status**
Insert student	Add a new student record	✓ Successfully inserted	✓ Pass
Retrieve by ID	Fetch student details	✓ Retrieves correct data	✓ Pass
Retrieve by email	Fetch student using email	✓ Retrieves correct data	✓ Pass
Update student profile	Modify student information	✓ Updates successfully	✓ Pass

Table 3.12: Student Database Test Cases

University Database Tests

Test Case	**Description**	**Expected Outcome**	**Status**
Insert university	Add a new university record	✓ Successfully inserted	✓ Pass
Retrieve by ID	Fetch university details	✓ Retrieves correct data	✓ Pass
Retrieve by email	Fetch university using email	✓ Retrieves correct data	✓ Pass
Update university details	Modify university profile information	✓ Updates successfully	✓ Pass

Table 3.13: University Database Test Cases

User Database Tests

Test Case	Description	Expected Outcome	Status
Insert user	Add a new user record	✓ Successfully inserted	✓ Pass
Duplicate user insert	Attempt to insert duplicate email	✗ Raises exception	✓ Pass
Check email uniqueness	Verify if an email is unique before registration	✓ Returns correct boolean	✓ Pass
Retrieve user type by email	Fetch account type of a user	✓ Retrieves correct type	✓ Pass
Retrieve user type by ID	Fetch account type using user ID	✓ Retrieves correct type	✓ Pass

Table 3.14: User Database Test Cases

3.1.7. Model Tests

The following tests verify the model functionality within the **Students & Companies (S&C) platform**. These tests check class attributes, getters, setters, conversions to dictionary format, and integration with the database layer.

Assessment Model Tests

Test Case	Description	Expected Outcome	Status
Get assessment ID	Retrieve the assessment ID	✓ Returns correct ID	✓ Pass
Get application ID	Retrieve associated application ID	✓ Returns correct application ID	✓ Pass
Get date	Retrieve assessment date	✓ Returns correct date	✓ Pass
Get link	Retrieve assessment link	✓ Returns correct link	✓ Pass
Convert to dictionary	Convert assessment object to dictionary format	✓ Returns correct dict	✓ Pass
Add assessment (valid)	Insert a new assessment	✓ Successfully inserted	✓ Pass
Add assessment (invalid)	Insert an assessment with error	✗ Returns None	✓ Pass
Get last assessment by application ID (valid)	Retrieve last assessment for a valid application	✓ Returns correct assessment	✓ Pass
Get last assessment by application ID (invalid)	Retrieve last assessment for a non-existent application	✗ Returns None	✓ Pass
Handle exception in get last assessment	Simulate an exception scenario	✗ Raises exception	✓ Pass

Table 3.15: Assessment Model Test Cases

Application Model Tests

Test Case	Description	Expected Outcome	Status
Get application ID	Retrieve application ID	✓ Returns correct ID	✓ Pass
Get student ID	Retrieve associated student ID	✓ Returns correct ID	✓ Pass
Get internship position ID	Retrieve associated internship position ID	✓ Returns correct ID	✓ Pass
Get status	Retrieve application status	✓ Returns correct status	✓ Pass
Convert to dictionary	Convert application object to dictionary format	✓ Returns correct dict	✓ Pass
Add application (valid)	Insert a new application	✓ Successfully inserted	✓ Pass
Add application (invalid)	Insert application with missing fields	✗ Returns None	✓ Pass
Retrieve by ID (valid)	Fetch application by valid ID	✓ Returns correct data	✓ Pass
Retrieve by ID (invalid)	Fetch application by non-existent ID	✗ Returns None	✓ Pass
Handle exception in get by ID	Simulate an exception in retrieval	✗ Raises exception	✓ Pass

Table 3.16: Application Model Test Cases

University Model Tests

Test Case	**Description**	**Expected Outcome**	**Status**
Get university ID	Retrieve the university ID	✓ Returns correct ID	✓ Pass
Get email	Retrieve email of the university	✓ Returns correct email	✓ Pass
Get university name	Retrieve the name of the university	✓ Returns correct name	✓ Pass
Convert to dictionary	Convert university object to dictionary format	✓ Returns correct dict	✓ Pass
Add university (valid)	Insert a new university	✓ Successfully inserted	✓ Pass
Add university (invalid)	Insert a university with missing fields	✗ Returns None	✓ Pass

Table 3.17: University Model Test Cases

User Model Tests

Test Case	**Description**	**Expected Outcome**	**Status**
Get user ID	Retrieve user ID	✓ Returns correct ID	✓ Pass
Get email	Retrieve user email	✓ Returns correct email	✓ Pass
Get user type	Retrieve user type (e.g., student, company)	✓ Returns correct type	✓ Pass
Validate correct password	Check correct password validation	✓ Returns True	✓ Pass
Validate incorrect password	Check incorrect password validation	✗ Returns False	✓ Pass
Get user type by email (valid)	Retrieve user type using email	✓ Returns correct type	✓ Pass

Table 3.18: User Model Test Cases

3.1.8. Authentication Service Tests

These tests validate authentication-related functionalities, including validation of different input types, email uniqueness, and authentication attributes.

Test Case	**Description**	**Expected Outcome**	**Status**
String validation (valid)	Check if a valid string is detected correctly	✓ Returns True	✓ Pass
String validation (invalid)	Check if an integer is incorrectly treated as a string	✗ Returns False	✓ Pass
Integer validation (valid)	Validate correct integer inputs	✓ Returns True	✓ Pass
Integer validation (invalid)	Validate incorrect non-integer inputs	✗ Returns False	✓ Pass
Float validation (valid)	Validate correct float inputs	✓ Returns True	✓ Pass
Float validation (invalid)	Validate incorrect non-float inputs	✗ Returns False	✓ Pass
Email uniqueness check (valid)	Check if an unused email is detected correctly	✓ Returns True	✓ Pass

Continued on next page

Test Case	**Description**	**Expected Outcome**	**Status**
Email uniqueness check (invalid)	Check if an existing email is detected correctly	✗ Returns False	✓ Pass
Email uniqueness check (exception)	Handle exceptions during uniqueness check	✗ Raises exception	✓ Pass
Email format validation (valid)	Validate correct email formats	✓ Returns True	✓ Pass
Email format validation (invalid)	Validate incorrect email formats	✗ Returns False	✓ Pass
Phone number validation (valid)	Validate correct phone number formats	✓ Returns True	✓ Pass
Phone number validation (invalid)	Validate incorrect phone number formats	✗ Returns False	✓ Pass
Password validation (valid)	Validate strong passwords	✓ Returns True	✓ Pass
Password validation (invalid)	Validate weak passwords	✗ Returns False	✓ Pass
URL validation (valid)	Validate correct URLs	✓ Returns True	✓ Pass
URL validation (invalid)	Validate incorrect URLs	✗ Returns False	✓ Pass
Location validation (valid)	Validate correct locations	✓ Returns True	✓ Pass
Location validation (invalid)	Validate incorrect locations	✗ Returns False	✓ Pass
Name validation (valid)	Validate correct names	✓ Returns True	✓ Pass
Name validation (invalid)	Validate incorrect names	✗ Returns False	✓ Pass
Degree program validation (valid)	Validate correct degree programs	✓ Returns True	✓ Pass
Degree program validation (invalid)	Validate incorrect degree programs	✗ Returns False	✓ Pass
GPA validation (valid)	Validate correct GPA values	✓ Returns True	✓ Pass
GPA validation (invalid)	Validate incorrect GPA values	✗ Returns False	✓ Pass
Graduation year validation (valid)	Validate correct graduation years	✓ Returns True	✓ Pass
Graduation year validation (invalid)	Validate incorrect graduation years	✗ Returns False	✓ Pass
Path validation (valid)	Validate correct paths	✓ Returns True	✓ Pass

Continued on next page

Test Case	Description	Expected Outcome	Status
Path validation (invalid)	Validate incorrect paths	✗ Returns False	✓ Pass
Optional path validation (valid)	Validate correct optional paths	✓ Returns True	✓ Pass
Optional path validation (invalid)	Validate incorrect optional paths	✗ Returns False	✓ Pass

3.2. Additional Notes

The code is well-structured, with thorough validation tests covering various input scenarios. It ensures proper handling of valid and invalid data types across different authentication and validation functions.

The test suite effectively verifies different edge cases, exceptions, and expected behaviors, demonstrating a responsible and systematic approach to software quality assurance.

The immediate solution to any encountered issues, such as downloading individual data, highlights the team's problem-solving skills and commitment to collaboration.

The tested group was highly responsive, addressing queries with prompt and detailed explanations, showing a strong understanding of their system.

3.3. Areas for Improvement

3.3.1. Meta-Testing (Tests for Tests)

While the current test suite is comprehensive, additional meta-tests could be implemented to verify the effectiveness and completeness of the test cases themselves. This would help ensure that:

- Test functions actually cover all possible edge cases.
- Expected failures occur where intended.
- Mocks and patches behave correctly.

Implementing **meta-testing** would add an extra layer of validation to the test framework, reinforcing the reliability and robustness of the software testing process.

4 | Effort Spent

Team Member	**Task**	**Hours Spent**
Shreesh Kumar Jha	1. Introduction and Document Structure 2. Installation Setup and Backend Installation 3. Test Case Mapping to Requirements 4. Additional Notes and Areas for Improvement	12
Samarth Bhatia	1. Installation Setup and Frontend Installation 2. Testing Environment and API Testing 3. Backend and Database Testing 4. Authentication Service and Validation Tests	11
Satvik Sharma	1. Model Testing and Meta-Testing 2. Effort Spent Calculation and Formatting	6

Table 4.1: Effort spent by each member of the group.

5 | References

5.1. References

- Software Engineering 2 Course Materials, A.Y. 2024-2025.
- Daniel Jackson, *Software Abstractions: Logic, Language, and Analysis.*
- Assignment RDD AY 2024-2025.pdf.
- MongoDB, Inc. (n.d.). MongoDB Manual.
- Node.js Foundation. (n.d.). Node.js Documentation.

List of Tables

1.1 Revision History . . . 2

3.1 Core Requirements Test Cases for Users . . . 12
3.2 Student Requirements Test Cases . . . 13
3.3 Application Requirements Test Cases . . . 14
3.4 Company Requirements Test Cases . . . 15
3.5 Feedback and Suggestions Test Cases . . . 16
3.6 Assessment Database Test Cases . . . 16
3.7 Application Database Test Cases . . . 17
3.8 Company Database Test Cases . . . 17
3.9 Complaint Database Test Cases . . . 18
3.10 Internship Database Test Cases . . . 18
3.11 Internship Position Database Test Cases . . . 19
3.12 Student Database Test Cases . . . 20
3.13 University Database Test Cases . . . 21
3.14 User Database Test Cases . . . 22
3.15 Assessment Model Test Cases . . . 23
3.16 Application Model Test Cases . . . 24
3.17 University Model Test Cases . . . 25
3.18 User Model Test Cases . . . 26

4.1 Effort spent by each member of the group. . . . 29