IT Deliverable

Edoardo. S. Gribaldo, Federico Rosa

Corso di Software Engineering 2, Politecnico di Milano

S&C web app

(code available at https://github.com/edogriba/GribaldoRosa)

Table of Contents

1. Introduction

The goal of this document is to point out and explain the processes and the choices we made while implementing our web application. It is just a summary of our ITD, without entering too much in the details since all the important information has already been written on the RASD and on the DD.

Keywords: [Tap here to add keywords.]

2. Requirements and functions

We implemented all the functions needed to fulfil the requirements written of ITD done by group of 2 people. In particular, we developed the following functions:

User registration

User Login

User Update

User Profile Visualization

Student Application Scrolling and Visualization

Student Internship Scrolling and Visualization

Student Application Confirmation/Refusal

Student Search

Company Internship Position Scrolling

Company Internship Scrolling and Visualization

Company Create Position

Company Application Scrolling and Visualization

Company Acceptance/Rejection/Assessment of Application

Company/Student Complaint Creation

University Internship Scrolling

Since they were not requested because we were 2 people, we choose not to develop the following functions:

Collection of statistics

Recommendation Algorithm

Collection of feedbacks

These part are however documented on the RASD and the DD since group of 2 students were requested to analyze them.

3. Adopted Development Framework

We adopted the following stack Python Flask + JWT Authentication + React + TailwindCSS + Flowbite + SQLite.

For the backend, we chose Python because it's a flexible language and has many libraries that help to create the backend. In this case we choose Flask to manage the users and JWT Tokens to manage the authentication and the session. Indeed since the goal was to deliver a working prototype we thought that Flask was powerful enough for our needs. Plus we both had previous experience with Flask.

For the frontend, we chose React because we believe it was the best match to our needs. React is really powerful and usually it is easy to integrate in a project. Again we both had a little bit of previous experience on React. Finally we also chose to use Flowbite which is a library for frontend components in order not to lose too much time in the design of the components and therefore focus more on their logic. Flowbite is built on top of TailwindCSS and their components can be customized by using TailwindCSS notation, so we though that it was a good trade off between accelerating the UI design and the possibility of having customized components.

For the database we used SQLite, because we thought that it was enough to have a traditional SQL database since the point of the project was not to deploy a huge database and try to manage a big number of transactions. Plus any SQLite database is a file, therefore it is easy to submit as a part of the code.

The API are extensively commented on the api.md inside the project.

4. Structure of the Source Code.

The code is divided in two: backend and frontend and they interact through API calls. Moreover the backend is mainly in the app folder and divided in db which contains the actual db, managers, models, services and utils. The frontend's most important folder is the src. Inside it there are the following folders: api, assets, auth, components, context, pages. The pages folder is the one containing most of the components code. To be precise, the structure of the source code (obtained with the command tree and deleting the imported modules or virtual environment files) is the following:

```
backend
├── app
    ├── __init__.py
    ├── api.md
    ├── app.txt
    ├── db
    │   ├── SC.db
    │   ├── __init__.py
    │   ├── __pycache__
    │   │   └── __init__.cpython-313.pyc
    │   └─── dbModels
    │       ├─── __init__.py
    │       ├─── application_db.py
    │       ├─── assessment_db.py
    │       ├─── company_db.py
    │       ├─── complaint_db.py
    │       ├─── internshipPosition_db.py
    │       ├─── internship_db.py
    │       ├─── student_db.py
    │       ├─── university_db.py
    │       └─── user_db.py
    ├─── managers
    │   ├─── __init__.py
    │   ├─── application_manager.py
    │   ├─── internship_manager.py
    │   ├─── login_manager.py
    │   ├─── profile_manager.py
    │   ├─── registration_manager.py
    │   └─── search_manager.py
    ├─── models
    │   ├─── __init__.py
    │   ├─── application.py
    │   ├── assessment.py
    │   ├── company.py
    │   ├── complaint.py
    │   ├── internship.py
    │   ├── internshipPosition.py
    │   ├── student.py
    │   ├── university.py
    │   └── user.py
    ├── routes.py
    ├── services
    │   └── auth_service.py
    └── utils
      ├── auth.py
      ├── error_handler.py
      ├── json_return.py
      ├── request_validation.py
      └── utils.py
    ├── test
    ├── __init__.py
    ├── test_db
    │   ├── __init__.py
    │   ├── test_accessment_db.py
```

```
|    ├─── test_application_db.py
|    ├─── test_company_db.py
|    ├─── test_complaint_db.py
|    ├─── test_internshipPosition_db.py
|    ├─── test_internship_db.py
|    ├─── test_student_db.py
|    ├─── test_university_db.py
|    └─── test_user_db.py
├─── test_managers
|    ├─── __init__.py
|    ├─── test_login_manager.py
|    └─── test_registration_manager.py
├─── test_models
|    ├─── __init__.py
|    ├─── test_accessment.py
|    ├─── test_application.py
|    ├─── test_company.py
|    ├─── test_complaint.py
|    ├─── test_internship.py
|    ├─── test_internshipPosition.py
|    ├─── test_student.py
|    ├─── test_university.py
```

```
        │   └─── test_user.py
        ├─── test_runner.py
        ├─── test_services
        │   ├─── __init__.py
        │   └─── test_auth_service.py
        └─── test_utils
          ├─── __init__.py
          ├─── test_auth.py
          ├─── test_error_handler.py
          └─── test_utils.py
 ├─── createDB.py
 ├─── requirements.txt
 ├─── run_tests.ps1
 ├─── run.py
frontend
 ├─── public
        ├─── index.html
        ├─── logo.png
        ├─── robots.txt
        ├─── uploads
        │   ├─── 35
        │   │   └─── img4_1706265022.612039.jpg
        │   ├─── 40
        │    │    ├──── HomeworksScores.pdf
        │    │    └──── img1_1707384225.528203.jpg
        │    ├──── 43
        │    │    └──── img2_1706265220.077955.jpg
        │    └──── 44
        │         ├──── IMG_6502.jpeg
        │         └──── img4_1706265022.612039.jpg
        └──── user.jpg
├──── src
        ├──── App.js
        ├──── App.test.js
        ├──── api
        │    ├──── api.js
        │    ├──── application.js
        │    ├──── company.js
        │    ├──── example.js
        │    ├──── internship.js
        │    ├──── request.js
        │    ├──── student.js
        │    ├──── university.js
        │    └──── user.js
        ├──── assets
```

```
│   └──── index.css
├──── auth
│   └──── ProtectedRoute.js
├──── components
│   ├──── Footer.jsx
│   ├──── GoBack.jsx
│   ├──── Navbar.jsx
│   └──── Status.jsx
├──── context
│   └──── UserContext.js
├──── hooks
│   ├──── useAuth.js
│   └──── useNotification.js
├──── index.js
├──── pages
│   ├──── About
│   │   └──── About.js
│   ├──── Complaints
│   │   └──── ComplaintList.jsx
│   ├──── Create
│   │   └──── Company
│   │       ├──── CompanyCreatePosition.jsx
│   │       └─── CompanyCreatePositionView.js
│   ├─── Dashboard
│   │   ├─── Company
│   │   │   ├─── CompanyApplication.jsx
│   │   │   ├─── CompanyApplicationView.js
│   │   │   ├─── CompanyInternship.jsx
│   │   │   ├─── CompanyInternshipList.jsx
│   │   │   ├─── CompanyInternships.js
│   │   │   ├─── CompanyPosition.jsx
│   │   │   ├─── CompanyPositionApplications.jsx
│   │   │   ├─── CompanyPositionList.jsx
│   │   │   ├─── CompanyPositionView.js
│   │   │   ├─── CompanyPositions.js
│   │   │   ├─── CompanyProfile.js
│   │   │   ├─── CompanyProfileCard.jsx
│   │   │   ├─── CompanyProfileTable.jsx
│   │   │   └─── CompanyTab.jsx
│   │   ├─── Student
│   │   │   ├─── StudentApplication.jsx
│   │   │   ├─── StudentApplicationList.jsx
│   │   │   ├─── StudentApplicationView.js
│   │   │   ├─── StudentApplications.js
│   │   │   ├── StudentInternship.jsx
│   │   │   ├── StudentInternshipList.jsx
│   │   │   ├── StudentInternships.js
│   │   │   ├── StudentProfile.js
│   │   │   ├── StudentProfileCard.jsx
│   │   │   ├── StudentProfileTable.jsx
│   │   │   └── StudentTab.jsx
│   │   └── University
│   │       ├── UniversityInternship.jsx
│   │       ├── UniversityInternshipList.jsx
│   │       ├── UniversityInternships.js
│   │       ├── UniversityProfile.js
│   │       ├── UniversityProfileCard.jsx
│   │       ├── UniversityProfileTable.jsx
│   │       └── UniversityTab.jsx
│   ├── Home
│   │   ├── Company
│   │   │   └── HomeCompany.js
│   │   ├── Student
│   │   │   └── HomeStudent.js
│   │   └── University
│   │       └── HomeUniversity.js
│   ├── Login
│   │   └── Login.js
│   ├── NotFound.js
│   ├── Register
│   │   ├── Company
│   │   │   └── RegisterCompany.js
│   │   ├── Register.js
│   │   ├── Student
│   │   │   └── RegisterStudent.js
│   │   └── University
│   │       └── RegisterUniversity.js
│   ├── Search
│   │   └── Student
│   │       ├── SearchFilter.jsx
│   │       ├── SearchResults.jsx
│   │       ├── StudentPosition.jsx
│   │       ├── StudentPositionView.js
│   │       └── StudentSearch.js
│   ├── Update
│   │   ├── Company
│   │   │   └── CompanyUpdate.js
│   │   ├── Student
│   │   │   └── StudentUpdate.js
│   │   └── University
│   │       └── UniversityUpdate.js
│   └── Welcome
│       ├── JumboTron.jsx
│       ├── SocialProof.jsx
│       ├── Welcome.js
│       └── WhoTrustUs.jsx
├── reportWebVitals.js
└── setupTests.js
```

5. Information on testing

This project makes extensive use of unit tests. Unit tests are written using the unittest framework and often use the unittest.mock module to simulate dependencies and isolate functionality in the testing phase. In all, more than 400 tests have been written covering all the backend functionalities regarding database calls, models, services and utils. The API calls were tested using Postman simulating the calls made to the frontend. Examples of the types of tests used:

- *Functionality Tests*: These tests check the correctness of individual functions or methods. They ensure that the functions return the expected results for the given input.
- *Mocking External Dependencies*: These tests use unittest.mock to mock external dependencies and verify interactions. This helps in isolating the unit of code being tested and ensures that the tests do not depend on external systems.

- *Exception Handling Tests*: These tests ensure that functions handle exceptions correctly. They verify that the appropriate exceptions are raised and handled as expected.
- *Database Interaction Tests*: These tests mock database interactions to verify that the application correctly interacts with the database. They ensure that the correct queries are made, and the expected results are returned.
- *Validation Tests*: These tests validate input data to ensure it meets the required criteria. They check that the validation functions correctly identify valid and invalid inputs.
- *Integration Tests*: Although less common, some tests may check the integration of multiple components. These tests ensure that different parts of the system work together as expected.

6. Installation instructions

Detailed instructions are available on the README.md of the project.

To install the project it is sufficient to clone the project from our GitHub repository (https://github.com/edogriba/GribaldoRosa). Then first go into the ITD folder with the command:

cd ITD/backend

Then create a virtual environment with venv (instructions online) and activate it.

Run this command:

pip install -r requirements.txt

Then start the server by running:

python3 run.py

Then go into the frontend:

cd ../frontend

And run this command:

npm install

Now let’s run the frontend

npm start

Now you can see the app on your localhost

To run the tests you can run ./run_tests.ps1

(if you’re on Mac you should first install powershell brew install –cask powershell)

7. Effort spent

Gribaldo: 120

Rosa: 140