POLITECNICO
MILANO 1863

SOFTWARE ENGINEERING II

Students&Companies - Implementation Document

Authors
Priuli Elia
Viceconti Veronica
Zuccoli Marco

29 January 2025
Version 1

Contents

1 What you need

1.1 To be installed

- latest JDK
- Tomcat 9
- Eclipse
- MySql Server and WorkBanch

For a more detailed guide concerning installation, please refer to the following file: Guide file.

1.2 DataBase

- Sql create schema script
- Sql create triggers script
- Sql populate schema script

Note: If you feel stuck please write as an email (marco.zuccoli@mail.polimi.it / veronica.viceconti@mail.polimi.it / elia.priuli@mail.polimi.it). We'll figure it out together.

2 Introduction

2.1 Scope

This document presents all the information regarding the implementation of a first prototype of the S&C software; the used technologies, programming languages, frameworks, APIs and developed functionalities. It also offers insights into the code structure and provides installation instructions.

3 Implemented functionalities

The functionalities that we have implemented are:

1. Allow unregistered users to create an account on S&C,
2. Allow registered users to log in to their accounts,
3. Allow the companies to publish their internships,
4. Allow the students to publish their CVs and their internships' preferences,
5. Allow the students to view all the matches found,

6. Allow the companies to view all the matches found,
7. Allow the companies and the students to accept or decline the proposed matches,
8. Allow the companies to create the interview form
9. Allow the companies to add responses of the students during the interview, to the form,
10. Allow the students to browse through the available internships,
11. Allow the students to directly make an application to the available internships,
12. Allow the students to send complaints,
13. Allow the companies to send complaints,
14. Allow the students to give the final feedback,
15. Allow the companies to give the final feedback,
16. Notify the companies whenever a new match is found (when a student apply from tab 'available internships'),
17. Create matches using keyword-search.

The few functionalities that we have not done are:

1. Allow the companies to modify their internships,
2. Allow the companies to delete their internships,
3. Allow the students to modify their CVs and their internships' preferences,
4. Allow the students to delete their CVs and their internships' preferences,
5. Notify the students whenever a new match is found,
6. Notify the user whenever match is finished and need a feedback,
7. Allow the universities to see their own students and their ongoing internships.

The first four functionalities (modification and deletion) have not been implemented as our web application is a prototype. The user can perform all the main functions, although these four add value to the user experience. However, as they are not essential for a prototype, they can be implemented at a later stage.

The last functionality (related to the university) has not been implemented because, as a group of three students, we were not meant to managing students' complaints by the university. Therefore, the "University" user would only be able to view the complaints, without being able to do anything else. For this reason, we decided not to implement even the visualization, but this can be added in the future.

3.1 Design Choices

Our prototype doesn't include all the design choices made in DD document, and here we will specify our motivations:

- The search engine component was not implemented because we did not implement all possible search filters for internships by the student, but only a filter by company name. However, in the future we plan to use the component to process all types of filters that may be useful to the user;
- In the various sequence diagrams within the Design Document (DD), all the interactions are depicted, from the most generic component down to the most specific one. However, during the implementation phase, while developing a web application using servlets, we directly connected the user's request to the component closest to the DBMS. This was done to avoid a mere series of data transfers between components, which would have otherwise occurred;
- The verifyCV function, which is responsible for checking the correctness of the CV, has not been implemented yet but could be added in the future;
- In the prototype, the interview form has been implemented statically with 3 questions. In the future, customization is planned for both the number and the questions themselves.

4 Adopted programming languages

Since the software is intended to be a webapp, the chosen programming languages are the following:

- Javascript: to manage dynamic pages (creation and user actions)
- Java: to implement business logic and connection with the Database

- PHP: used just to access the Google Firebase Cloud Notification (FCM) API, in order to implement the notification service
- SQL: to run operations on a relational DB (MySQL)

For the server side, Apache Tomcat has been used as a webserver.

5 Code structure

Our code structure is composed of frontend and backend.
The frontend is composed of a set of folders, each folder contains different elements:

1. css, includes all the css files that we used to create the style of our pages, it's the style of the user interface,
2. img, includes all the images that we added into our webapp,
3. js, includes all the javascript files that we used to create dinamic pages and also for some frontend controls, such has right input. Almost all the name of the js files are quite explicative, but here we describe breafly the ones that have a tricky name:
   - accept_declineStudent_Company, this js file is used when the company open a match and has to decide whether accept or decline the student of that match,
   - acceptDeclineInterview_Company, this js file is used when the company open an interview already made of a student and has to decide whether accept or decline the student, and if possible, the internship becomes an ongoing internship,
   - internshipInfo_acceptDecline_Student, this js file is used for two reasons base on what the student is doing; the first one is when the student open an internship and visualize the information about that internship plus the possibility to accept/decline it, and the second one is when the student open a match (new,accepted, waiting for interview, ..) and visualize the information about it plus the possibility to accept/decline it if is a new one,
   - internshipViewCompany, this js file is used when the company want to see the information about the ongoing internship, all its proposed internships or the ones that are finished and need a feedback yet.
4. html, includes all the elements inside each page, it's the DOM (Document Object Model).

Instead, the backend is composed of five principal folders:

1. Bean, contains all the classes used to implement and realize our webapp,such ass Student,Company,Publication,Internship, .. ;
2. Controller, contains all the component of our webapp, each component receive a request from the frontend and create a response with all the things requested;
3. Dao, used to upload data on database and to download data from database, each DAO manages the queries and data that are requested for a particular class, i.e. StudentDAO contains all the methods that asks information about the student;
4. Filters, contains the servlet filters which are used to control if the user has already done the login before to access the server;
5. Test, contains the JUnit test case used to do the test cases.

To make things more clear, we want to explain the process that our webapp follows from frontend to backend and viceversa:

- The user connects to our webapp;
- The user make a request to the backend, i.e. clicking a button;
- The request is send to the backend specifing the component (controller) to call;
- The corresponding component analyse the request and, based on that, process the request, controlling everything the user is asking is reasonable;
- If the request is valid, the component pass the request to the corresponding DAO in order to question the database;
- The component send the response (good or bad) to the frontend;
- The frontend is updated and the user visualize it.

6 Testing

The tests planned within the design document are functional tests and load tests, but since the prototype is executed through localhost, load tests were not carried out as it is not yet within the final environment.
For the realization of functional tests, JUNIT 5 and the Mockito framework were used, which allowed for the creation of mockups of http requests to test the individual components.
The tests aim to verify every possible request to the server, verifying both

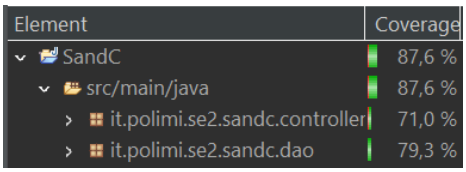

Figure 1: Reached test coverage

any positive paths and any negative paths; note that exceptions due to the db (SQLException) were not tested, as in our code they only occur in case of disconnection from it, since the data before being inserted are checked, and it was not possible to test the init() method of the classes as well as the CV upload functionality as the servlet context is needed which cannot be realized with the mockup. The iteration to test the various functionalities are mainly two:

1. Functionality that involves loading data into the database is tested by executing the request and then verifying that the data has been correctly inserted into the database;

2. Data retrieval functionality is tested by executing a request and verifying the consistency of the server's JSON response with the database contents.

In the case of functionalities that involve both loading and retrieving data, both tests are performed.
The database used for testing is emptied and refilled with SQL queries before each tested functionality, so as to know a priori what is contained in the database before the execution of the request and to know what should be contained after the execution. Furthermore, the schema used to test the functionalities on the db is different from the one used by the main application, "testsandc" is used instead of "sandc".

7 Installing the software

7.1 Prerequisite

- Tomcat 9
- MySql
- Eclipse
- A chomium-based browser (chrome, edge)(recommended)

To run correctly the upload and the saving of the cv it's necessary to change the web.xml file (contained in /src/main/WEB-INF/) to set a correct absolute path of where to save the files on your computer; the context param (see figure 2) "pathUploadCv" need as param the new absolute path where to upload

your cv, and need to be written in this form: C://path//to//upload// (or C:/path/to/upload/ for linux).

```
<context-param>
    <param-name>pathUploadCv</param-name>
    <param-value>C:\\Users\\epriu\\eclipse-workspace\\cv\\</param-value>
</context-param>
```

Figure 2: Parameter to upload the cv

In order to correctly execute all tests over the codebase, it's required to update the "setUp()" function in all .java files at src/main/java/it/polimi/se2/sandc/test, by inserting your DB user and password (see figure 3, row 74-75)

```
72          String driver = "com.mysql.cj.jdbc.Driver";
73          String url = "jdbc:mysql://localhost:3306/testsandc?serverTimezone=UTC";
74          String user = "<DB_USER>";
75          String password = "DB_PASSWORD";
76          Class.forName(driver);
77          connection = DriverManager.getConnection(url, user, password);
```

Figure 3: Set your DB user and password

To make the notification work the following steps need to be followed BEFORE the login procedure on the website:

1. clean the browser cache (recommended)
2. turn on browser notification from the operating system's settings (figure 4)
3. allow website notification from the browser (if necessary, set browser's anti-tracking system to the lowest setting) (figure 5)

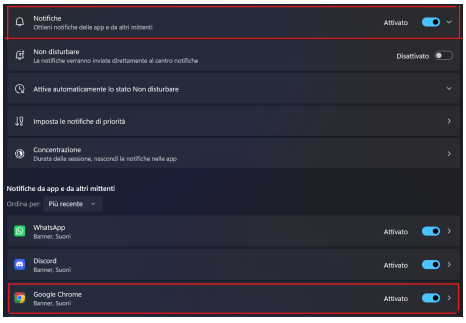

Figure 4: Turn on browser notification (Windows)

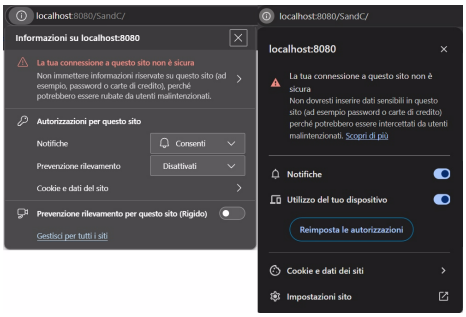

Figure 5: Allow browser notification (Edge - Chrome)

Note: since this is just a prototype, notification are only implemented to notify the company whenever a student apply for an available internship (spontaneous application).
If you want to test notification, you must open two users (student and company) with two different browsers due to session constraints.

7.2 Instructions

Here you can find a summary of all the phases you need to go through to run correctly the project:

- Install Tomcat 9,Eclipse and MySQL using the pdf guide written in 'what you need' section;
- Clone our project from git to your pc;
- Open the project with Eclipse;
- In Mysql create a new schema called "sandc", triggers and populate it using the sql files guide written in 'what you need - to be installed' section;
- Make notification work using the guide written above in 7.1 section;
- In Eclipse in web.xml (/src/main/WEB-INF/) put you our DBUser, DBPassword and your absolute path for cv (using the guide written above in 7.1 section);
- To run the project from Eclipse, go to DeliveryFolder/SandC and run it with Tomcat 9 server (right click - run as - run on server).

8 Effort spent

The table below show the number of hours that each member of the group worked for the implementation phase.

Member	Hours
Elia Priuli	112h
Veronica Viceconti	102h
Marco Zuccoli	102h

9 Reference

- previous RASD document
- previous DD document
- Assignment IT AY 2024-2025.pdf on the webeep page of the course