SEPI2 2X – S16
Internship 101

Project Report

Group 10: Andrei-Mihai Surueanu student number: 240370
Robert Iulian Zainea student number: 240000

Supervisors: Henrik Kronborg Pedersen (HEKP)
Stephan Erbs Korsholm (SEK**)**

Table of Contents

1. Abstract 2
2. Introduction 3
3. Analyze 3
3.1. Requirements 3
3.1.1. Functional Requirements 3
3.1.2. Non-Functional Requirements 4
3.2. Use Case Diagram 4
3.3. Use Case Descriptions 6
3.4. Class diagram (model) 8
4. Design 8
4.1. Class diagram 9
5. Implementation 9
6. Test 11
7. List of References 11
8. Appendices 12
8.1. Appendix 3 – Analysis documentation 12
8.1.1. Use Case diagram 12
8.1.2. Use Case descriptions 13
8.2. Appendix 4 – Design documentation 21
8.2.1. Design class diagram 21

1. Abstract

This project shows a problem, which is creating a centralized system of internships that companies and students use in congruence. The problems of this system included creating a database to support the functionality of the program, use RMI to connect multiple computers and a method to store real files into a database.

(Shaw, 2005)

2. Introduction

This is a complete extraction from the Background Description inside the Project Description that is present in the uploaded documents.

"At the moment when a student at university needs to find an internship, he has several few options. Either a teacher can suggest companies with internships to the student that the teacher knows and then recommend the student to the company, and guide the student to apply for that company and for that internship. Or either the student must rely on personal network to find an internship in the near proximity (that is, the same town or country) or sometimes even abroad in less cases. This means that the student must talk to older colleagues that have already been through the practical placement period and receive from them feedback (that is, impressions, comments, suggestions).

The lack of a centralized system of internships has led to a problem that almost all students face when the time comes to choose an internship.

The student's reliance on teachers or their own network to find an internship is a major issue in our society as we know it today. An alternative to this approach is to search the World Wide Web with the hope of finding the right internship or the most suitable one. And having at their disposal hundreds or even thousands of web sites to look at and to choose from is not a valid solution either.

The need of a centralized system in this scenario is obvious."

3. Analyze

3.1.Requirements

The system Internship 101 has two types of actors, two types of users. This means that the system has one user which is the Company, as in general terms referring to an abstract entity called company. The second type of user that the system has is the Student.

3.1.1. Functional Requirements

- The company must be able to create an internship that represents an open position that a student can apply for.
- The system creates an internship using information collected from the company.
- The information necessary to create an internship consists of: the title, the date when the internship starts, the specialization choosing from Embedded, Cross Media and Business Information Systems, the duration in months, a description.
- The student must be able to view all the available internships.
- The student should be able to search for an internship using various filters.
- The filters for searching an internships could be location (town, country), company, specialization, duration.
- The student must be able to apply for an internship.

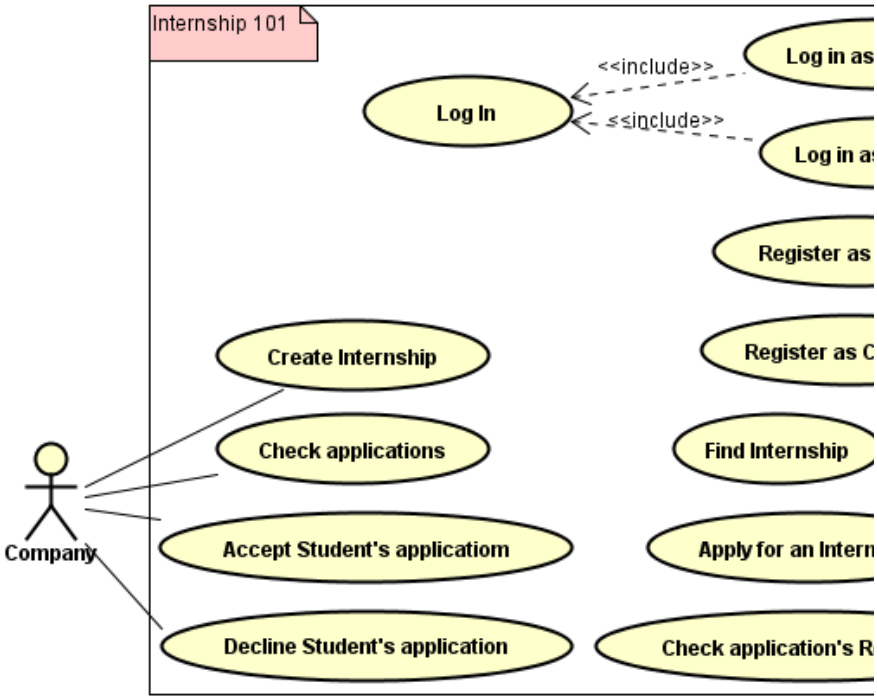
- The student must be able to upload a CV and a Cover Letter so that the file can be accessible by the company that has created that internship.
- The company must be able to select an internship that has been created and view the description of that internship and the students that have applied for that internship.
- The company must be able to select a student, which has applied for the previously selected internship, and see that application and download the CV and Cover Letter.
- The company must be able to accept or decline an application from a student.
- The system deletes the student's application from the Database when the company chooses decline.
- The student should be able to check the responses for the submitted applications.
- The student can accept or decline the internship that the company has provided an answer of acceptance.

3.1.2. Non-Functional Requirements

- The system must have a Graphical User Interface
- The system has to be implemented in Java.
- The system must have a connection with a PostgreSQL Database.
- The system should implement client/server architecture.
- The Database must be on the server side of the application.
- The system must have a Registration and Log-in procedures.
- The system should use Remote Method Invocation to store data into the Database and retrieve data from the Database to display the information to the users.
- The system must have a division option to differentiate between the Students and the Companies.

3.2. Use Case Diagram

The Use Case Diagram present below is a visual representation of the functionalities that the Internship 101 system should be able to perform.

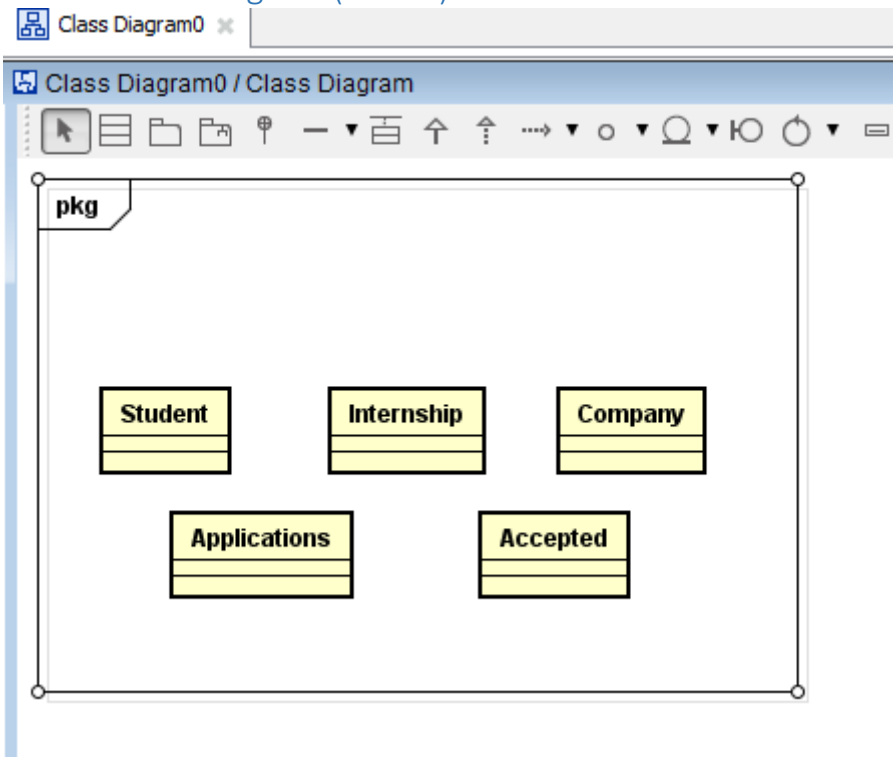

Internship 101
Log In
<<include>>
<<include>>
Log in as
Log in as
Register as
Register as C
Create Internship
Check applications
Accept Student's applicatiom
Decline Student's application
Find Internship
Apply for an Intern
Check application's R
Company

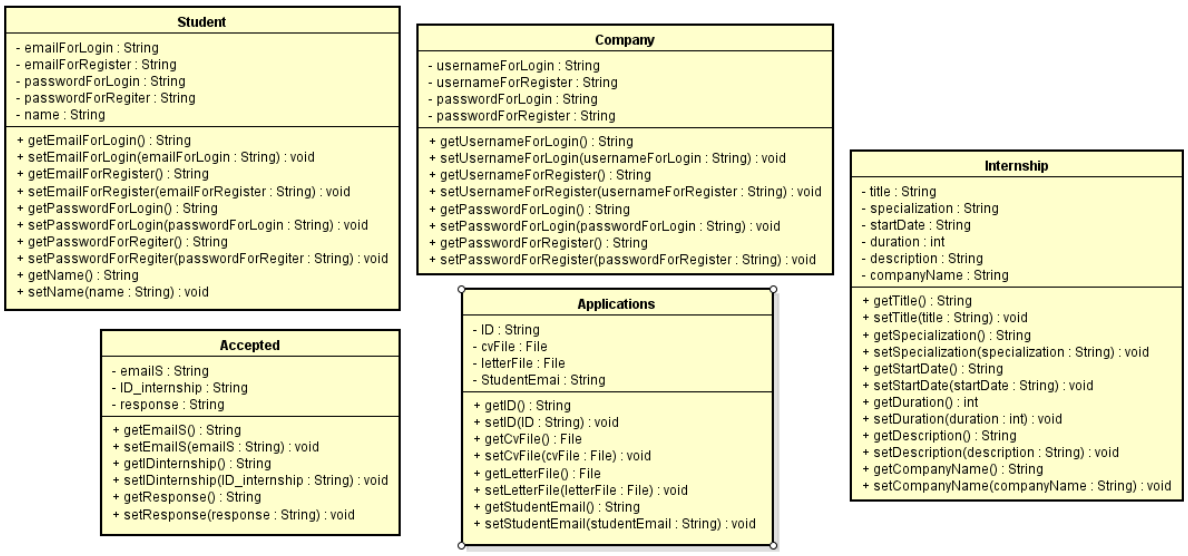
3.3.Use Case Descriptions

ITEM	VALUE
UseCase	Create Internship
Summary	User creates an internship position for students,
Actor	Company
Precondition	Company profile opened.
Postcondition	Program saves the internship in the database and informs the user of success.
Base Sequence	1. User selects a Create Internship option. 2. User fills up with information about the internship (that is, title, specialization, starting dat e, duration and description). 3. User confirms. 4. Inform user of successfully creating the Internship.
Branch Sequence	
Exception Sequence	Exit: 1-2 As base sequence. 3. Program does not save anything in the Database.
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Apply for an Internship
Summary	
Actor	Student
Precondition	Student profile opened.
Postcondition	
Base Sequence	1. User selects one of the displayed Internships. 1. User selects Apply option. 2. User loads a CV document. 3. User loads a Cover Letter. 4. User selects the upload option called Apply.
Branch Sequence	
Exception Sequence	Only one file loaded: 1-4 As base sequence. 5. Inform user to load both files. Application exists: 1-4 As base sequence. 5. Inform user that application exists.
Sub UseCase	
Note	

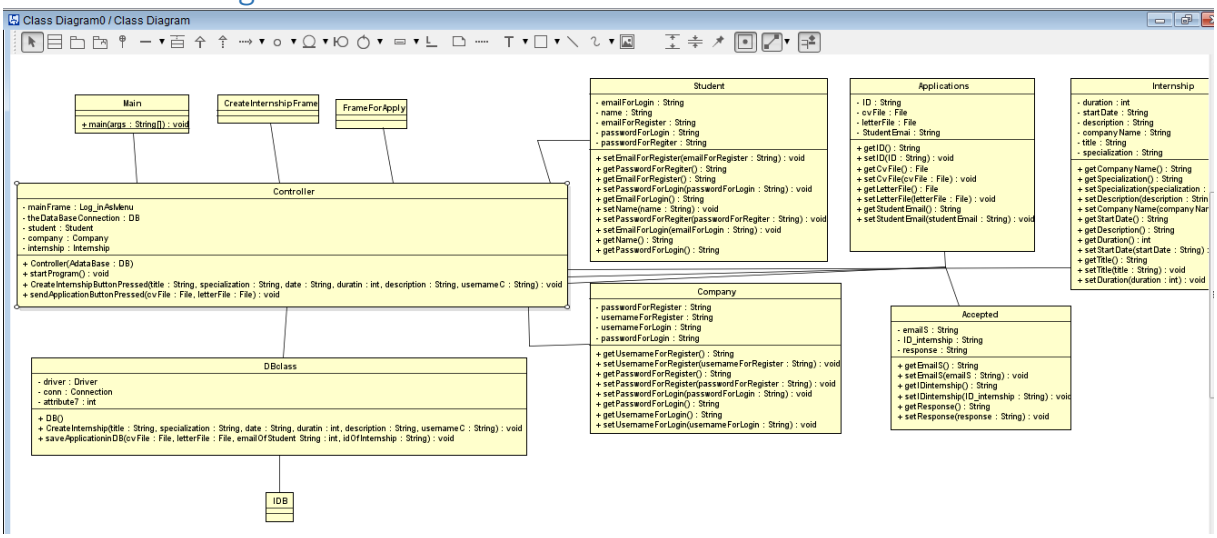
3.4. Class diagram (model)





4. Design

4.1.Class diagram



5. Implementation

The majority of the design choices based on the Analysis section are implemented with success in the final system.

The below Figure presents the “CreateInternship” method, a method that represents one of the key, core functionality of the system. This method implements the “Create Internship” use case present in the Use Case Diagram for the Internship 101 system.
The mentioned method can be located in the DB class, in the “storage” package. The method has as parameters the data needed to store in the PostgreSQL Database in order to create, but more important, in order to perserve the data in the internship for later usage. In the current method there are also Java

Database Connection related methods, for example:
“prepareStatement();”and “executeUpdate();”.

```
// CREATE INTERNSHIP
public void CreateInternship(String title, String specialization,
        String date, int duration, String description, String usernameC) {

    try {

        String Update = "INSERT INTO Internship "
                + "(title,companyName,specialization,start_date,duration,description)
                + "VALUES(?,?,?,?,?,?)";
        PreparedStatement preparedSt = conn.prepareStatement(Update);
        preparedSt.setString(1, title);
        preparedSt.setString(2, usernameC);
        preparedSt.setString(3, specialization);
        preparedSt.setObject(4, date);
        preparedSt.setInt(5, duration);
        preparedSt.setString(6, description);

        preparedSt.executeUpdate();

        preparedSt.close();

        JOptionPane.showMessageDialog(null, "Internship created");
        CreateInternshipFrame.CREATE_internship.dispose();
        CompanyProfile.companyProfileFrame.dispose(); // close the window

    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, e);
    }

}
```

Down below is a very important method, present as well in the DB class. This method presents a way to store in the Database the Application of a Student for the desired internship. This method is also a solid pillar for a proper functioning of the “Apply for an Internship” Use Case present in the Use Case Diagram. For the implementation of this method there were used the following elements: a “FileInputStream”, the method “setBinaryStream();” (to convert the File data type into an array of bytes data type, for a proper storage in the PostgreSQL Database), and for the creation of the actual “Applications” table in the

Database there was used the “bytea” data type for the “cv” and “letter” collumns.

```
// Save application to the Database
public void saveApplicationinDB(File cvFile, File letterFile,
        String emailOfStudent, String idOfInternship) {
    try {

        FileInputStream cvStream = new FileInputStream(cvFile);
        FileInputStream letterStream = new FileInputStream(letterFile);

        int IDinternship = Integer.valueOf(idOfInternship);

        String apply = "INSERT INTO Applications(cv,letter,email,ID_internship) "
                + "VALUES(?,?,?,?)";
        PreparedStatement preparedSt = conn.prepareStatement(apply);

        preparedSt.setBinaryStream(1, cvStream, cvFile.length());
        preparedSt.setBinaryStream(2, letterStream, letterFile.length());
        preparedSt.setString(3, emailOfStudent);
        preparedSt.setInt(4, IDinternship);

        preparedSt.executeUpdate();

        cvStream.close();
        letterStream.close();
        preparedSt.close();

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,e);
    }

}
```

6. Test

For the Internship 101 System there were performed numerous test to verify the adequate functioning of the implemented Use Cases. Still, there have been observed several malfunctions, bugs, misbehaviors for these Use Cases.

One of the misbehaviors is taking place when a user, both Company and Student, performs a Register operation. After the insertion of the information requested from the user, and the confirmation of the user, the program displays the message “You have registered successfully!” in a JOptionPane and offers the option to the user to go back to the Main Menu through the “Go to Login” button, this happens when the user tries to register with an email or with a username that already exists in the Database. The good behavior in this scenario is that the program does not store neither the Student nor the Company in the Database, for the reason that it violates the Primary Key constraint in the Database for the “email” and “namec” columns.

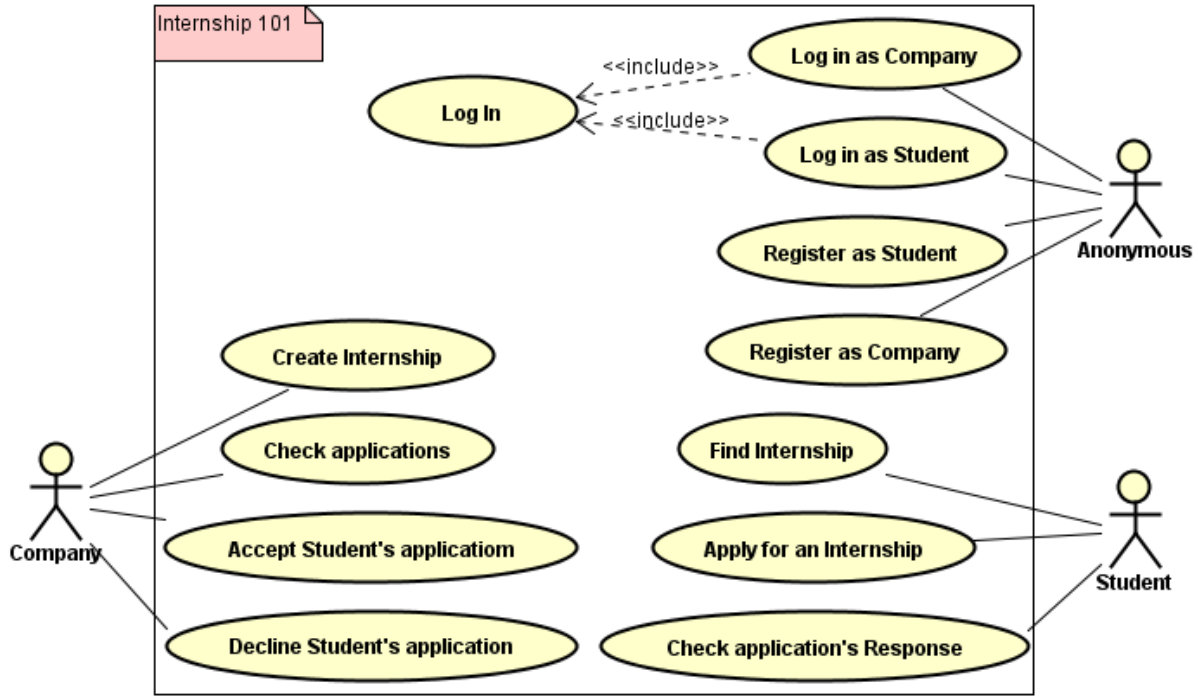
7. List of References

Shaw, M. (2005, October 18). *Writing Good Software Engineering Research Papers.* Retrieved from Carnegie Mellon University: https://www.cs.cmu.edu/~Compose/shaw-icse03.pdf

8. Appendices

8.1.Appendix 3 – Analysis documentation

8.1.1. Use Case diagram



8.1.2. Use Case descriptions

ITEM	VALUE
UseCase	Log in as Company
Summary	
Actor	Anonymous
Precondition	User has selected the Log In as Company option from the Ma
Postcondition	
Base Sequence	1. User logs in (Use case: "Log In"). 2. Inform user of successful login. 3. Open Company profile.
Branch Sequence	
Exception Sequence	
Sub UseCase	Log In
Note	

ITEM	VALUE
UseCase	Log in as Student
Summary	
Actor	Anonymous
Precondition	User has selected the Log In as Student option from the Main menu.
Postcondition	
Base Sequence	1. User logs in (Use case: "Log In"). 2. Inform user of successful login. 3. Open Student profile.
Branch Sequence	
Exception Sequence	
Sub UseCase	Log In
Note	

ITEM	VALUE
UseCase	Log In
Summary	User logs in to enter the user's profile.
Actor	
Precondition	User has selected Log In as Student option or Log In as Company option from the Main menu .
Postcondition	The program has found the input in the Database.
Base Sequence	1. User enters email/username. 2. User enters password. 3. User confirms.
Branch Sequence	
Exception Sequence	Not registered or wrong input: 1-3 As base sequence. 4. Program asks user to try again and informs the user of wrong input.
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Register as Student
Summary	
Actor	Anonymous
Precondition	User selects the Register option from the Main menu and Register as Student option.
Postcondition	
Base Sequence	1. User enters name, email, password and retype password. 2. User confirms. 3. Program infroms user of successfully registration. 4. User selects the option to go to Log In.
Branch Sequence	
Exception Sequence	Email already exists: 1-2 As base sequence. 3. Inform user that the email exists. Password and retype password not equal: 1-2 As base sequence. 3. Inform user that the passwords do not match.
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Register as Company
Summary	
Actor	Anonymous
Precondition	User selects the Register option from the Main menu and Register as Company option.
Postcondition	
Base Sequence	1. User enters username, password and retype password. 2. User confirms. 3. Program infroms user of successfully registration. 4. User selects the option to go to Log In.
Branch Sequence	
Exception Sequence	Username already exists: 1-2 As base sequence. 3. Inform user that the username exists. Password and retype password not equal: 1-2 As base sequence. 3. Inform user that the passwords do not match.
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Find Internship
Summary	
Actor	Student
Precondition	Student profile opened.
Postcondition	
Base Sequence	1. User selects the desired filters. 2. User selects search.
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Apply for an Internship
Summary	
Actor	Student
Precondition	Student profile opened.
Postcondition	
Base Sequence	1. User selects one of the displayed Internships. 1. User selects Apply option. 2. User loads a CV document. 3. User loads a Cover Letter. 4. User selects the upload option called Apply.
Branch Sequence	
Exception Sequence	Only one file loaded: 1-4 As base sequence. 5. Inform user to load both files. Application exists: 1-4 As base sequence. 5. Inform user that application exists.
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Check application's Response
Summary	
Actor	Student
Precondition	Student profile opened.
Postcondition	
Base Sequence	1. User selects the Show Acceptance History option.
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Create Internship
Summary	User creates an internship position for students,
Actor	Company
Precondition	Company profile opened.
Postcondition	Program saves the internship in the database and informs th
Base Sequence	1. User selects a Create Internship option. 2. User fills up with information about the internship (that is, e, duration and description). 3. User confirms. 4. Inform user of successfully creating the Internship.
Branch Sequence	
Exception Sequence	Exit: 1-2 As base sequence. 3. Program does not save anything in the Database.
Sub UseCase	
Note	

8.2.Appendix 4 – Design documentation

8.2.1. Design class diagram