Computer Science

Final Documentation - 2022 Fall Software Engineering

Easy Run

Tang Sheng
Penghao Weng
Yuqian Sun

Supervised by Lihua, Xu

Table of Contents

1. Description

2. The process

3. Requirements & Specifications

4. Architecture & Design

5. Reflections and Lessons Learned

1. Description

EasyRun is a web application built on Java SpringBoot and React. It aims to provide American graduate school application services and intermediate institution rating services for Chinese students. It allows users to view intermediaries and graduate program information. There are two types of logging-in users, intermediate institutions and students. Instructors can change profiles, change institution information, upload applications, and send contracts to students. Students can change profiles, star institutions/programs, accept/reject contracts, and rate/comment on institutions.

2. The process

We followed an XP process and met with each other in two-week's iteration.

<u>Iteration1</u>

Together:

1. API design; Brainstorm on class diagram, use case diagram.

Kyrie:

1. Finish Back-end file structure design.
2. Finish implementation of packages {demo.student.account, demo.instructor.account, demo.utils, demo.objects}

Jay:

1. Finish database design and React frame work for front-end.
2. Finish most part of class diagram design.
3. Design one sequence diagram.

Sienna:

1. Learning React and Selenium;Design one sequence diagram.

Future Tasks

1. Revise and design a better back-end file structure
2. Do the front end with REACT
3. Revise all the APIs to make them standard and robust
4. Start writing the test files

<u>Iteration2</u>

Jay:

1. Finish Sequence Diagram Design.
2. Finish the Login page and Register page front-end build-up. Successfully connect with the APIs and retrieve data from the database. Basic data flow test cases for these two pages pass.
3. Finish the Home page design for page redirects.
4. Help with API error fixes during front-end interactions (i.e. new API which returns the list of institutions for instructor user type register to choose from, and other error code cases.)

Kyrie:

1. Finished ALL back-end service design and implementation.
2. Passed all postman API tests.
3. Coordinated with Jay to fix some bugs in back-end service.
4. Optimized backend file structures for a bit.
5. Fixed some complex relationships between entities.
6. Add features for instructor entity and profile service.

Sienna

1. Cowork the register page
2. Learning selenium to write UI test
3. Write bots for login and register page that tests on all the buttons and input field(on Chrome currently)

Future Tasks

1. Make possible changes to the current testing file structure to make higher code usage.
2. With the current front-end,write bots for newly created pages.
3. Complete the Programs page, Institutions page, Schools page, and Profile page.

Iteration 3

Jay:

Complete Front-end Development Version 1:

1. Finish the User Profile Page for two different types of users.
2. Finish all user actions including Edit Profile, Reset Password, Edit Institution Information, Upload Application, Send Contract, Process Contract, Rate Institution, Star Programs.
3. Finish the basic Institutions, Page and Programs Page.
4. Finish specific pages for Institutions, Programs, and Universities.
5. Finish basic window redirects of the App bar (Header).

Kyrie:

Complete Back-end Development Version 1:

1. Add some implementations/functions of the back-end in response to front-end needs.
2. Rewrite some codes in order for better logic and pattern.

Sienna:

1. Complete the section on the specific institution page
2. Finish the user comment actions on specific institution pages.

Future Tasks

EasyRun Version 2:

1. Better decorate the App bar and design a Footer bar.
2. Construct a user image system if time permits
3. Construct a user chatting system if time permits
4. Re-design and re-implement some backend files according to useful design patterns.

Iteration 4

Jay:

1. Add “rated information” feature and “Rating Radar” graph.
2. Update “sign contract” action with the back-end changes.
3. Update “comment” action according to the back-end changes.

Kyrie:

1. Add feature: (1) When students get contracts, show overall rating if rated. (2) Change the contract's status field from String type to Integer type (3) Provide more parent comment information when getting a list of comments;
2. Complete Proxy Pattern when the client gets all programs and all universities.
3. Complete Observer Pattern on the action of the instructor preparing contract. Will simulate an external email sending system (once a new contract is initiated, notify the student)
4. Fixed minute bugs.

Sienna:

1. Complete Selenium test on Selenium IDLE on Chrome
2. Fix the header problems and some redirect problems

Future Tasks

EasyRun Version 3:

1. More test cases
2. Profile photo & logo of universities.

3. Requirements & Specifications

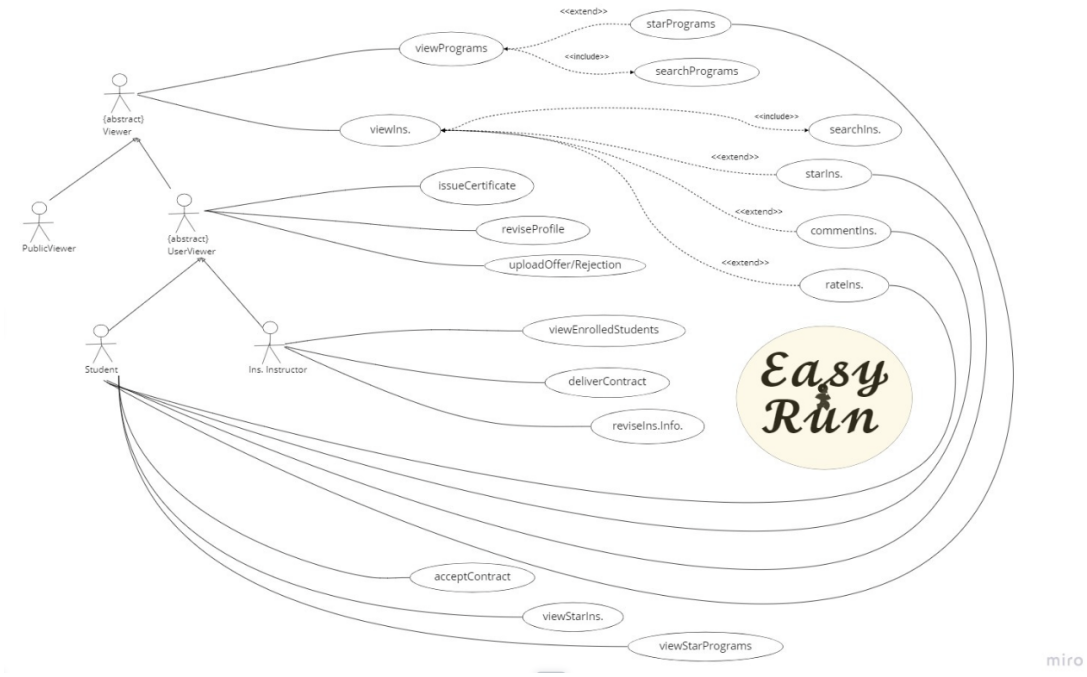

Figure: Use Case Diagram

UC1 Issue Certificate
UC2 View Programs
UC3 Star Programs
UC4 Search Programs
UC5 View Institutions
UC6 Search Institutions
UC7 Star Institutions
UC8 Comment Institutions
UC9 Rate Institutions
UC10 Revise Profile
UC11 Upload Offer
UC12 deliver Contract
UC13 Revise Institution Info
UC14 Accept Contract

UC3 Flow of Events for the [starPrograms] Use Case

1.1 Preconditions

1. The actor has to be Student.
2. The actor must log in.
3. The actor needs to do this action in the program detail page.

1.2 Main Flow

1. The actor clicks the star button in the program detail page and starts the program.

1.3 Subflows

1. The detail page displays the new star status and we update the database.

1.4 Alternative Flows

1. When the actor clicks the button, if the program is already starred by the actor, un-star the program instead.

4. Architecture & Design

4.1 System Architecture

Front-end:

The system basically consists of 6 pages: home page, login page, register page, profile page, institutions page, and programs page. Most of the user actions are performed by opening a dialog component instead of redirecting to other new pages. Former 3 pages are the basic pages which are constructed in directory: ./frontend/src/basic pages/. Other three complex pages are

constructed in their individual directories and would be illustrated below.

./frontend/src/basic pages/:

Home page, Login page and Register page.

./frontend/components/:

Reusable components i.e., Alerting, specific page information card, Rate box, Table, etc.

./frontend/components/DialogController.js/:

The controller of opening a dialog component to execute user actions.

./frontend/forms/:

Some reusable fill-out forms.

./frontend/institution page/:

List out institutions and specific institution page.

./frontend/layout/:

Specific page (institutions, universities, programs) skeleton and Header.

./profile page/:

List out programs and specific university/program page.

./rating forum/:

Forum of specific institutions.

./user actions/:

All log-in users' actions

./services/:

API calls of different action services.

Back-end:

The system contains six packages and 2 java files at the root at *./backend/src/main/java/com/easyrun/demo*.

Run *DemoApplication.java* to start the server.

Package *Instructor* contains controller, service, and data-access layer related to API's concerning instructors.

Package *Student* contains controller, service, and data-access layer related to API's concerning students.

Package *User* contains controller, service, and data-access layer related to functions that don't require users to login.

Package *Objects* contains a micro-service, and data-access layer related to all the non-actor objects.

Package *Observers* contains event listeners and handlers for the observer pattern implementation.

Package *Utils* contains util classes such as response definitions.

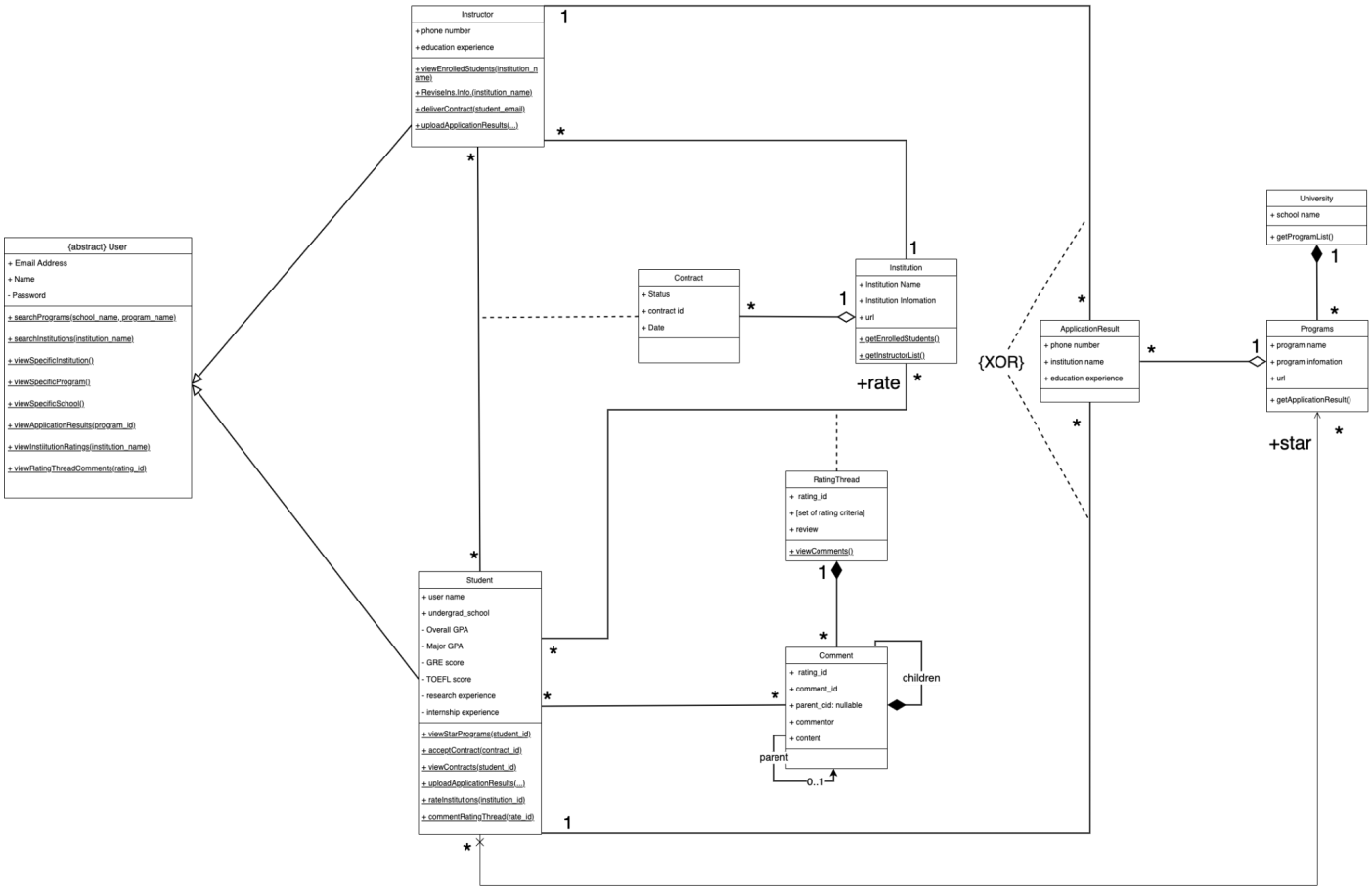

Figure: Class Diagram

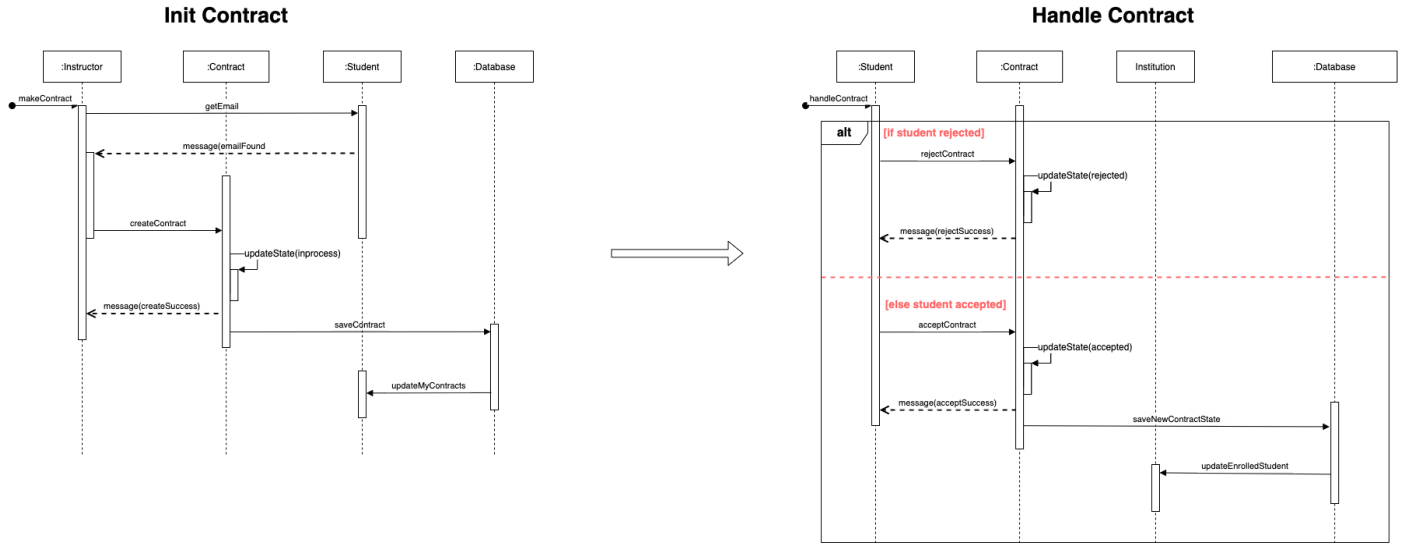

Figure: Sequential Diagram

4.2 Choice of Framework

Frontend: React

React allows high component reuse and helps the code to be more organized. The JSX grammar combines Javascript, HTML, and CSS, which makes the writing code logically clearer. The hook and use state function can keep track of state changes that introduce easier control over user interactions. Based on the feature, we design the classes and file structures to be reusable component-based, behavior-based, and page based.

Backend: Spring Boot + Maven

Spring Boot reduces lots of development time and increases productivity, avoids writing lots of boilerplate Code, Annotations, and XML Configuration, while it is still very easy to integrate Spring Boot Applications with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security.
Maven helps manage all the processes, such as building, documentation, releasing, and distribution in project management. It speeds up the project construction process significantly.

Test: part Selenium Python, Selenium IDE

Selenium is a web bot tool that allows acceptance testing. The Selenium Python tool allows developers to have more control over the process of testing. While the Selenium IDE is a tool that can record user behavior and replay it. So the test is designed by feature.

5. Reflection and lessons learned

Together

- Frequently updating the changes that happened in the developing process is important. Taking the API documentation as an example, we have made some changes to the API documentation in the process but didn’t record them. So confusion happened in communication between the backend and frontend in passing information.

Kyrie

- Learned Java, Maven, Spring Boot, and PostgreSql.
- Learned to apply design patterns and design principles in practice.

- Refactoring according to design patterns is important. This often happens when new requirements come in and new evolutions are needed while the current structure is not efficient enough.

Jay

- Learned React and Github
- Don’t consider too much when starting the project. In the beginning, we tried to come up with an excellent plan and reusable structure for frontend components. A better solution is to refactor bit by bit during the process that is closer to the best choice.

Sienna

- Learned React, Selenium and Github
- How to build a project from scratch and use UML to help organize
- How to collaborate with others with code since people have slightly different coding styles that people need to be adjusting.
- The CSS in react should be dealt with more carefully, since some later on testing operations have things to do with the CSS selector, each UI component ought to have their own unique identifier to avoid confusion.