Software Engineering Practice

By : CHEKOU Nesrine,
McCONNELL Remy,
MOUNDI MAZOU Garira,
VANBALBERGHE Vincent

POLYTECH MONTPELLIER

UNIVERSITÉ DE MONTPELLIER

I. Introduction

The Fire Gang is proud to present its brand-new application : The Fire Simulator.

This application was developed as part of Software Engineering Practices, proposed and managed by Tiberiu STRATULAT.

This project was initially meant to be a forest fire simulator that would have allowed the user to create his/her own forest with various elements having their own properties and then launch a simulation to see how a fire would spread from a place he/she would have chosen. The purpose was also to provide a dynamic interface allowing the user to interact during the simulation and try to reduce the damages.

To be in line with the context of Software Engineering Practices, our original idea for the project quickly evolved into a game platform still based on the forest fire theme but with many more features. Thus, the application is now split into two main options : the solo part and the online one. The first of which contains the whole forests, called map, and element creation. The second is supposed to provide an online platform which would allow multiplayer and tournament games.

This subject piqued our interest for the following reasons : first of all, it is an original topic and none of us have worked on this type of subject before. Second of all, we are well too familiar with the typical event management application, and we figured most groups would pick a topic of this type (which is the case). We wanted to take advantage of the fact that we are students to create this type of project, rarely made in companies, but also at the same time to discover new development tools.

We called our team the Fire Gang, in other words, four Computer Science students at Polytech Montpellier, France. Nesrine CHEKOU, Garira MOUNDI MAZOU, Remy McCONNELL and Vincent VANBALBERGHE. Together, we have conceptualized, built, and realized this project with love.

II. Tools and Technologies

General

In this project, our mindset was to learn new technologies or the ones we lightly discussed in our courses. All the technologies we used were unknown to us (except collaboration tools such as Git, Google drive, …).

Database

For the database, we chose to work with MongoDB. From the outset, we opted to implement the database online with MongoDB Atlas, a free and documented resource.

MongoDB was introduced to us this semester in BDNG. We were intrigued by it because it brings flexibility, lighter and cleaner code, and moreover, MongoDB is a more modern technology than other options, such as PostgreSQL and MySQL.

IDE / text editor

The text editor is important to help in the development phase. We chose to work in IntelliJ IDEA instead of Eclipse that we commonly use in school. There were a few reasons for this decision : First, we found that the version control in IntelliJ was much more sophisticated and streamlined than in Eclipse, and with a project of this size, we needed to make sure we had easy access to our Git. Also, IntelliJ IDEA provides a lot of useful plugins which make the programming overall experience more enjoyable. Some of these plugins allow us to generate UML diagrams (Such as PlantUML and Sequence

Diagram plugin) with reverse engineering, making it easier for us to follow the agile method. This means that others could generate a great part of the javadoc instead of working on the diagrams and so on.

As part of the project, we used the Scene builder Application because it provides us with a user-friendly interface to quickly make views with JavaFX by dragging elements directly on the stage. Note that IntelliJ also have an integrated SceneBuilder Plugin but when the UI becomes more complex, it becomes very heavy to use.

Programming language

The required language is Java, thereupon, all of the project is coded in Java except for the views, which are coded in FXML.

Collaboration tools

We used Google Drive to store and share files amongst ourselves, like mockups pictures and use case reports. We also used DrawIO and LucidChart to create a majority of our UML diagrams. This website allows us to work on diagrams together simultaneously. Finally, the last tool we depended on was Git. During the development phase of the project, each task had its own branch. This allowed us to each work on different aspects of the application without interfering with each other. Once the work was functional and finished, we would merge the branch to Master.

III. Features

The user must have access to internet to logged in to Fire Simulator and use our features. Otherwise, he/she can only see the login and register page.

Legend : Implemented | not implemented | modified

Login

Initial features :

- log in to the game with a username and password.

We successfully implemented this use case since it was the first task we took on. We needed to make sure our connection to the database worked before implementing any other features, so we thoroughly worked on the login functionality.

Register

Initial features :

- create an account. This account is save in a distant Database.

We successfully implemented this use case simultaneously with Login.

User Profile

Initial features :

- consult all personal data. This included User information, gameplay rules, available level.
- consult perk tree
- modification on User information : change Username, password
- delete User account

We implemented some information about the game in this part that does not figure in the corresponding use case, this is due to an omission. This included consulting the gameplay Data : Characters, Items, xp, level, diamond and gold.

The perk tree is not implemented at this time but we replace it by a small panel that displays the level of the user.

Store

Initial features :

- consult coins gained in the tournament mode
- view the user's inventory of characters and their abilities
- view and purchase available store items

After some consideration, we decided to place the user's inventory in the user profile. Other than this change and a few naming differences, we followed our original use case very closely.

Element

Initial features :

- create element from basic element or user's Elements
- modify user's elements
- delete user's elements

We largely followed this UseCase, however we still need to implement some verification, for example, at this time, two elements can have the same color.

One user can only delete his own element. Basic elements cannot be deleted.

Map

Initial features :

- create a map
- save a map
- modify a map
- delete a map

A map can be created with basic elements, or with the user's element. A user can set an element on the map by clicking on the desired element and then clicking on the chosen box. We have not implemented the drag and drop functionality.

Simulation

Initial features :

- choose a map
- choose a character
- launch simulation

This use case is not finished. The main algorithm is available, but it needs to be continued to implement the 'scorched' state and to adjust the fire probability.

The view is not properly implemented and doesn't allow monitoring of the simulation.

We need this last property to implement Characters in the simulation.

Post

Initial features :

- write a post with privacy settings (public or friends only)
- write a comment on a post
- modify a post belonging to the user
- delete a post belonging to the user
- filter posts (all posts, only friends' posts, or the user's posts)

We implemented every feature except for the ability to add privacy settings to a post.

Friends

Initial features :

- consult the user's list of friends
- add new friends
- delete friends
- chat with friends

We followed this use case pretty closely, aside from the ability to chat with friends. We ultimately had to remove this feature due to time constraints.

The time available for the project coming to an end, we unfortunately gave up the following use cases : Solo Match (the game aspect of our application), Multiplayer and Tournaments (both depending on Solo Match and online aspects).

IV. Stages

Coming up with the idea :

In the early stages of this project we each came up with an idea and voted on which we liked the best. We ended up with two ideas we were happy with, a tournament management application and a forest fire simulator application, so we decided to merge the two. During the first week of the project, we exchanged ideas and conversed back and forth in order to fully grasp the project and plan out our vision of how we wanted it to be. Working on the use cases for the application helped us solidify our idea and prepared us for the work to come.

Splitting up the workload :

Once we understood what we wanted to create, the time had come to enumerate our use cases, draw our mockups and create our class diagram. We evenly split up the use cases between all of us and worked individually on them. However, for the class diagram we all worked together, still bouncing ideas off of one another and making small modifications to the project.

Preparing to work individually :

We started off the development phase with the login and register features. These we implemented together so we would all be on the same page with regards to our architecture and the connection to our database. Once this was completed, we each picked two to three features that we wanted to implement and complete them on our own during our winter break.

Coming together afterwards :

After working alone for a few weeks, we each demonstrated the work we had done and began the process of merging our git branches and unifying the UI. This was a relatively painless process because we started the development process in an organized manner. Although we did not finish everything we set out to do, we adopted the proper techniques and worked efficiently, and we believe that with a few more weeks of work we could deliver a completed and polished Fire Simulator.

Distribution of work

Use cases :

Nesrine C.	Map Management	User Profile	Post
Garira M.M.	Tournament	Match	Solo Match
Remy M.	Element	Account	Gameplay
Vincent V.	Friends	Multiplayer	Simulation

All the Use Cases were corrected and revised as a team.

Implementations :

Nesrine C.	Login	Element	Map Management (-)	Simulation
Garira M.M.	Login	Register	Friends	User Profile
Remy M.	Login	Register (-)	Store	Post
Vincent V.	Login	Element (-)	Map Management	Simulation

(-) means that the targeted developer has done less than 50% of this use-case and/or he was mostly a support.

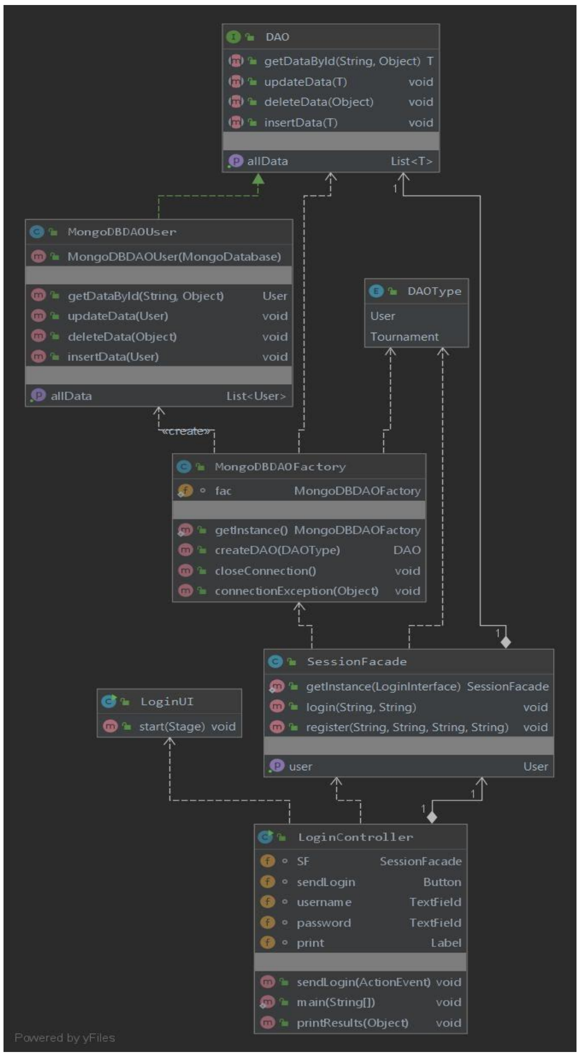
V. Project architecture

Design pattern

We mostly used the Singleton and Factory patterns. The former was used especially on our various facades between the UI and the DAOs due to the sheer amount of them and the need to maintain an instance of each throughout our program. The second pattern was used to create the DAOs depending on the type of data we needed it to handle.

We made an attempt to apply the Observer pattern for the Simulation use case but this one appeared to be superficial for what we meant to implement. Indeed, the JavaFX framework already provided us with the required environment.

We can see below the class diagram for the login UseCase. This is a good representation of the architecture we used for the other features of our application.



VI. Thanks

Special thanks to STRATULAT Tiberiu : our dear tutor, undeniably dedicated to helping us become better software architects. For his unequivocal advice, although sometimes radical, (“absolutely not”, “no, you’re wrong”), legitimate and justified, it ended up enriching our experience.