Farm Game SENG Project Team 115

Caitlin Tam cta108   88675910

Sophie Moller smo227   68087975

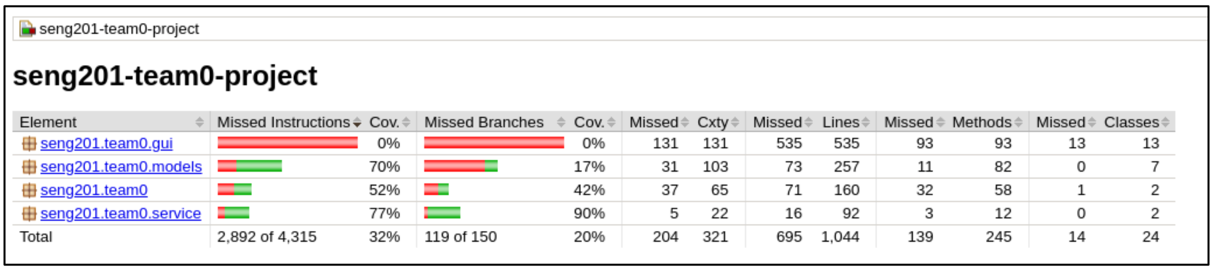
Farm Game

Our game application is structured into several key components: the Player Manager, Controllers, Models, and Service Classes. The Player Manager serves as the central game environment, managing the player state, launching screens, and handling round operations, while Controllers set up FXML pages and handle user interactions. Models encapsulate game entities such as Player, Round, Cart, Tower, Upgrade, RandomEvent, and Shop, providing and storing relevant information and attributes. Service Classes like RoundService and CartService manage the core game logic, such as round execution and random cart generation. We split up the code into these smaller components in order to ensure our code is easy to understand, test and reuse.

Communication between classes occurs through method calls, with the Player Manager interacting with both the Controllers and Service Classes to update the game state and UI. We employed use of modular code and high decoupling, limiting interactions between model classes allowed for flexible, maintainable, and testable code. This design ensures efficient data handling, encapsulation, which we developed and targeted throughout the development of our game, depicted in our UML class diagram.

We continuously strived to ensure our project was accessible. To achieve this, we included home button links on every page, and a give up option after each round. We designed a consolidated inventory table to avoid overwhelming users with too many screens, yet we separated the upgrade and tower tables to prevent confusion. Additionally, we incorporated sliders and buttons to enhance interactivity within the game, along with images and a progress bar to provide a clear and engaging user experience.

Unit Test Coverage



seng201-team0-project

seng201-team0-project

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
seng201.team0.gui		0%		0%	131	131	535	535	93	93	13	13
seng201.team0.models		70%		17%	31	103	73	257	11	82	0	7
seng201.team0		52%		42%	37	65	71	160	32	58	1	2
seng201.team0.service		77%		90%	5	22	16	92	3	12	0	2
Total	2,892 of 4,315	32%	119 of 150	20%	204	321	695	1,044	139	245	14	24

Our unit tests cover around 50% of the classes in the game, reflecting the structure of the application. We focused testing on the service and model classes, as these handle the core game logic, and did not test the controller and GUI classes. This decision was made because the game's logic is encapsulated within the service and model classes. Consequently, these classes have a high percentage of tested methods and lines. However, our testing coverage of service and model classes is not complete, as we did not include tests for trivial getters and setters. Given that the GUI comprises a substantial portion of the project's methods and lines, the overall testing coverage is around 50%.

Project Overview

The project was an excellent learning experience, demonstrating our significant improvement from the start, largely due to our consistent work determination and time we dedicated to the project. We are proud of the outcome, and worked well together, sharing similar work ethics and schedules. However, our initial planning and structure could have been better, and setting stricter deadlines would have allowed more time for improvements and additional features. If we were to do a similar project in the future we would invest more time into detailed upfront planning, set clearer deadlines, and increase our testing coverage to enhance code robustness for future projects.

Overall, we are happy with the outcome and the progress we made.

Hours spent on the Project:

Caitlin – 115

Sophie – 115

Contribution:

Caitlin Tam – I contributed 60% to the overall project.

Sophie Moller – I contributed 40% to the overall project