4.5 Report

Write a short two-page report describing your work. Include on the first page:

• Student names and ID numbers.

• The structure of your application and any design decisions you had to

make. We are particularly interested in communication between classes

and how interfaces and/or inheritance were used. You might want to

reference your UML class diagram.

• An explanation of unit test coverage and why you got a high/low percentage

coverage.

Include on the second page:

• Your thoughts and feedback on the project.

• A brief retrospective of what went well, what did not go well, and what

improvements you could make for your next project.

• The effort spent (in hours) on the project per student.

• A statement of agreed percentage contribution from both partners.

**Design decisions**

The structure of your application and any design decisions you had to

make. We are particularly interested in communication between classes

and how interfaces and/or inheritance were used. You might want to

reference your UML class diagram.

Math for round difficulty

**Test coverage**

An explanation of unit test coverage and why you got a high/low percentage

coverage.

For JavaFX we managed the testing in an excel spreadsheet manually to insure we wouldn’t have bugs and scene builder related issues which covered feature that we had implemented in gui. For unit test coverage we tested the core components for model classes and service classes

**Thoughts and feedback**

Your thoughts and feedback on the project.

A brief retrospective of what went well, what did not go well, and what

improvements you could make for your next project.

We setup a Trello board initially but didn’t really use our Trello board towards the end of the project we instead mostly worked in person and relied on communication heavily when working remotely to know who is working on what and using branches also made the process of avoiding merge conflicts.

We setup our Timeline for the project using a Gantt chart which provided a good outline for us to follow and we stuck to that plan for the first 4 weeks but in that final week the time we add set out to complete javafx classes and service classes we underestimated the amount of time these classes would take which forced our hand and made us have to push back some of our tasks and crunch to get these features implemented which limited our testing coverage to find bugs. It also didn’t help that we didn’t go back to our Gantt and adjust our goals and hours after we initially set it up which just contributed to the extreme increase in workload in the last week.

Our class diagrams changed significantly when we first created them in the planning requirements faze as well haven’t had significant experience in large multithreaded applications.

We should have had a higher static amount of hour put into the project each week rather than slowly ramping up our workload this would have left us with more time to deal with risks, issues, and bugs and left our last week to debugging, documentation and packaging up the project.

Improvements that we could make for the next project could include better planning implementation

NOTES:

