ENG1 - Assessment 2

Testing

Test2.pdf

Group 6

Freya Goodger	sg1967
Mikolaj Wyrzykowski	mw2179
Barnaby Matthews	bm1287
Cooper Love	cl2702
Oliver Cassey	oc854
Anna Hrynyshyn	ah2886
Oliver Thompson	ot699

Briefly summarise your testing method(s) and approach(es), explaining why these are appropriate for the project. (4 marks, ≤ 1 page)

For our testing, we have two main approaches: automated testing and manual testing. Automated testing is achieved using unit testing. We unit test logical sections of code where the output can be interpreted as correct or incorrect by a computer; for example, calculating the score at the end of the seven days. The project we took over was not set up for testing, so we chose our own approaches and retroactively applied them where we could to the original code we inherited. If necessary and not too challenging, we refactored code to be more testable. This usually went in hand with those parts being extended anyway, since modifications were being made it was easier to write it in a more testable way. A lot of the more testable parts of the project are saved for the second part, such as the score calculations, leaderboard and streaks. When writing those new sections, we wrote tests for them as well.
To view the status of testing over the whole project, we use JaCoCo to generate coverage reports, which integrate into IntelliJ (our IDE of choice). The results of this are discussed later in this document. This information is also available under the testing part of our website.

Manual testing is required for parts of the game which cannot be tested automatically (without significant processes that are beyond the scope of this project). Therefore, we have systematic manual test-cases to cover the rest of the game. These are available under the testing part of our website. They are laid out, as mentioned, in a systematic method, with specific step-by-step processes for the tester to follow to cover as many possible defects in the game as possible. This is not as bullet proof as automated testing, but is necessary for several parts of the game that a computer cannot quantify the same as a human, such as testing the way parts of the game look and “feel”.

Overall, we use several appropriate approaches for the testing of a game. These approaches were selected to cover as much of the game as possible. In particular, manual testing is very important for a game, due to subjectiveness and untestable aspects.

Give a brief report on the actual tests, including statistics of what tests were run and what results were achieved, with a clear statement of any tests that are failed by the current implementation. If some tests failed, explain why these do not or cannot be passed and comment on what is needed to enable all tests to be passed. If no tests failed, comment on the completeness and correctness of your tests instead (10 marks, ≤ 3 pages).

For traceability, we name the requirement that is being tested by the test. These are functional requirements, since we are testing the functionality of the components.
In the below table, “Test Name” is the name of the unittest in the source code of the game for automatic tests, and the name of the manual test available on the website.

Test Name	Requirement(s)	Test Type	Status
CharacterMovementTest	FR_INPUT_DETECTION,	manual	passed
GameDurationTest	FR_GAME_DURATION,	manual	passed
UITest	FR_UI	manual	passed
AssetMapTest	FR_MAP	manual	passed
CanInteractTest	FR_INPUT_DETECTION	manual	passed
AccessibilityTest	FR_SETTINGS	manual	passed
ChooseAvatarTest	FR_CHOOSE_AVATAR	automatic	(not made)
SleepTest	FR_SLEEP	automatic	(not made)
EnergyLevelTest	FR_ENERGY_ACTIVITY_COUNT	automatic	(not made)
HappinessLevelTest	FR_HAPPINESS_ACTIVITY_COUNT	automatic	(not made)
StudyLevelTest	FR_STUDY_ACTIVITY_COUNT	automatic	(not made)
FailureTest	FR_FAILURE	automatic	(not made)
StreakTest	FR_STREAK	automatic	(not made)
AchievementTest	FR_ACHIEVEMENTS	automatic	(not made)
ScoreboardTest	FR_SCOREBOARD	automatic	(not made)

The coverage report for the latest version of the game is available on our website here (link). Our automatic tests covered XX% of the game's classes(?). The other XX% is not testable automatically, so we use manual testing to cover the rest of it. These manual tests are available on our website here (link).

No tests were failed by the current implementation, and our tests have a high coverage over the entire project. All of our code passes all the automatic tests that exist for it. Furthermore, every functional requirement is covered by a test, whether it be manual or automated.