UNIVERSITY OF YORK
DEPARTMENT OF COMPUTER SCIENCE

ENG 1
GROUP 9 - Kitchen Tossups

Report - Testing

Team Members:

Liam Burnand
Cameron Fox
Jack Cameron
Adam Penny
Eoin O'Connor
Ben Brown

PART A

Due to the simplicity in the project that we took over, it seemed counter-productive to implement tests prior to starting the second part of the assessment. However, for Assessment 2, we have chosen to follow a test-driven development approach, developing tests for relevant features prior to development whether these are automated or manual. This ensured that the development team were aware of the requirements for each feature, to ensure that all functionality is met and saves time as they do not have to rigorously test code, instead leaving this up to the testing team. If a test is missed and noticed by the development team, they could ask us to implement the relevant test, thus facilitating future development. As the development team is following a calendar to develop certain features by certain dates, we were able to spread the writing of tests throughout development, ensuring that as many edge cases as possible could be considered.

As we are using libGDX as our graphics framework, automating testing has proven challenging and requires classes to be structured in a way where logic and textures are completely separated. This causes errors as the game's graphics are unable to be instantiated for testing purposes meaning that live values cannot be accessed without requiring a complete refactor. Having done this would have been time consuming and would have significantly complicated the code base.

Instead we have put a major focus on manual testing, this provides both advantages and disadvantages. A disadvantage of this choice is that manual testing doesn't highlight the specific methods or classes where bugs may be present, possibly leading to development delays on specific features. On the other hand, implementing manual testing is much more suitable to testing graphical components such as the sizing or positioning of sprites on the screen which is much easier to test by observing whether the feature is implemented correctly and as planned.

Our manual tests are created from a combination of requirements both old (from assessment 1) and newly added in assessment 2. This ensured that all targets we required our game to meet could be accounted for. Each test features a name, requirement to pass, whether the test has been passed and an explanation for the reason it failed as shown in part B. Then in Part C we have described how we performed our tests by giving the criteria and the state of the system when we performed the test. Tests that have failed or only partially passed should be made a priority for the future development of the game.

PART B

Test Name	Requirements To Pass Test	Pass	Explanation (If Failed)
Intuitiveness Test	The game features a control screen on load up.		
↑	Pause menu can be opened using ESC key		
ControlOne CookTest	Only one cook can be controlled at one time by the player (FR_CONTROL_ONE_COOK)		
SwitchCooks Test	Should be able to change movable cook using the tab key. (UR_SWITCH_COOKS)		
ControlCook Test	Chef moves using either WASD or arrow keys. (Both should be implemented)		
↑	Stations can be interacted with using the E key when a valid item is held.		
CarryTest	Only one item can be carried by a chef at a time. (FR_CARRY)		
↑	Item can be picked up and placed using the E key		
ServeCusto merTest	Customers demand to be served by showing a list of orders on the left side of the screen. (UR_SERVE_CUSTOMER)		
ServeDishTe st	Be able to serve the finished dish at the counter to the customer using the E key. (UR_SERVE_DISH)		
↑	Is the order removed from the screen when it is served to the customer.		
↑	Is the item removed from the chef's possession when served to the customer.		
LoseReputat ionPointTest	Check whether a reputation point is lost after the time limit for the set difficulty. (FR_LOSE_REP_POINT)		
FunGameTe st	The game is enjoyable to play. (UR_FUN_GAME)		The level of enjoyment is subjective to the individual playing the game.

RestaurantDesignTest	Do graphics resemble the Piazza Building on Campus East? (UR_RESTAURANT_DESIGN)		Subjective. Resembles a restaurant kitchen in general.
DifficultyTest	There exists an option for easy, medium and hard mode within the menu screen (UR_DIFFICULTY)		
LimitedTimeTest	In easy mode customers leave the restaurant in 240 seconds (FR_LIMITED_TIME)		
↑	In medium mode customers leave the restaurant in 180 seconds		
↑	In hard mode customers leave the restaurant in 120 seconds		
CollisionDetectionTest	Cooks are unable to travel out of the bounds of the gameplay area, or walk through counters or other chefs		
ScaleTest	All components are clear and visible on display resolutions 1920 x 1080, 1366 x 768 (NFR_SCALE)		The games ideal resolution is 1280 x 720
↑	The game window can be scaled and behaves the same at different sizes.		This works occasionally and thus requires future improvement
OSTest	The game runs as expected on Windows (NFR_OS)		
↑	The game runs as expected on Mac OS		
MemoryUsageTest	The game doesn't require excessive amounts of memory and therefore runs on compatible systems with > 4 GB RAM (NFR_MEMORY_USAGE)		
SmoothGameTest	The game runs at a visually smooth frame rate with minimal stutter. (UR_SMOOTH_GAME)		
FastInteractTest	Test interactions take < 2 secs (NFR_FAST_INTERACT)		All interactions occur instantaneously. Ingredients in stations take longer by design.
SaveStateTest	The game state is saved correctly (UR_SAVE_GAME)		

ReadSaveTest	The game save is read correctly into the game to allow continuation.		The game saves but does not read back the save file.
StartReputationPointsTest	The player starts the game with 3 reputation points		
PattyFlipTest	Test a patty can only be flipped once the first side is cooked (FR_PATTY_FLIP)		
BurgerTest	Test cannot serve an improperly-constructed burger (FR_BURGER)		
SaladTest	Test cannot serve an improperly-constructed salad (FR_SALAD)		
PizzaTest	Test cannot serve an improperly-constructed pizza (FR_PIZZA)		
JacketPotatoTest	Test cannot serve an improperly-constructed jacket potato (FR_POTATO)		
PowerUpTest	Power Up - extra life (UR_POWER_UPS)		
↑	Power Up - increase chef speed		
↑	Power Up - clear next order		
↑	Power Up - speed up chopping		
↑	Power Up - speed up cooking		
CustomerArrivalsTest	Test that customers arrive one by one, or in groups of 2 or 3 (FR_CUSTOMER_ARRIVALS)		
GameOverTest	Game is over when the number of reputation points reaches zero (FR_GAME_OVER)		
GameWinTest	Game ends when all the customers have been served and the player has reputation points remaining (FR_GAME_WIN)		

Pass Rate = 39/44 = 89%

PART C

As automated testing was not used in this project, no testing or coverage report has been generated. We can, however, provide details on the test cases used for each manual test, which can be found at https://kitchentossups.github.io/assessment2#testdata.