Testing Documentation

Formula Fast

Version History

Version	Date	Author(s)	Summary of Changes
0.1	24 Mar 2024	Gabriel Azhari	Created design documentation pages.
0.2	27 Mar 2024	Gabriel Azhari	Finished the introduction page.
0.3	27 Mar 2024	Gabriel Azhari	Began the term definition list on the summary page.
0.4	27 Mar 2024	Ashton David Ryan Franklin	Wrote the unit test documentation page explaining what is to be tested by the unit tests, and why we are testing these components.
0.5	30 Mar 2024	Que Hung Dang	Added test case in integration test page
0.6	30 Mar 2024	Gabriel Azhari	Finished the conclusion paragraph on the summary page.
0.7	30 Mar 2024	Mahmoud Sherif Mohamed Radwan	Added the completed system testing documentation.
0.8	01 Apr 2024	Matthew Marbina	Added validation tests for functional requirements from requirements modeling section

Testing Documentation Introduction

Overview:

Education is crucial in understanding the complex world today, and many people may find it hard to have any enthusiasm when it comes to learning. However, educational games successfully blend the excitement of gaming with the benefits of learning. This is proven by studies that show that gamification in education can lead to higher levels of motivation, increased attention spans, and stronger retention of the learned material compared to conventional teaching methods.

Taking this into account, we aim to create an educational game that manages to enhance the learning experience for all. The software we create will be called “Formula Fast”. Formula Fast will be a racing game where the user races against other cars that are computer-controlled. The user can only move their car forward by answering math equations and the user will have to cross the finish line before the timer countdown reaches 0. Also, the equations will become progressively more and more difficult as the user completes various levels. To achieve this we will be using Java as our coding language and VS Code as our IDE. We will also be using JavaFX and Scene Builder for the GUI.

For the design portion of the project we will be using a class diagram to illustrate the initial design concept for our project, paired with a textual description of the diagram. Furthermore, we will be creating an interactive mockup of the user interface for our software using Balsamiq, paired with a textual description of each writeframe. We will also cover how we will be storing data involved in the project (CSV files), the aforementioned development environment we will be using, and what patterns we will be using in regards to our software.

Specifically, for this testing portion of the project we will be discussing our detailed test plan for our software. Our plan will specify the test cases we are using to assess the quality of our software and the justification for their inclusion in our testing. Moreover, this testing documentation will demonstrate why our plan is sufficient and complete. This documentation will cover four parts of our test plan: unit testing (testing focused on each unit of the software as implemented in the source code), integration testing (testing on the connecting of the units in the software, reflected in the software's overall design and architecture), validation testing (testing that the requirements detailed in the requirements modelling are validated in our software), and system testing (testing the software and other systems as a whole). To test our software we will be using a combination of white box/structural testing and black box /functional testing. Also, for the unit testing we will be using JUnit and for the integration, validation, and system testing we will be creating a testing document to detail each test we will perform on the software.

Objectives:

The objectives of the project and software are to:

- apply the principles of software engineering towards a real-world problem
- work with, interpret, and follow a detailed specification
- create models of requirements and design from the given specification
- implement our design in Java and deal with decisions made earlier in the design process
- create graphical, user-facing content and applications
- write robust and efficient code
- write good, clean, well-documented Java code that adheres to best practices
- reflect on good/bad design decisions made over the course of the project

With regards to educational objectives, the main objective of Formula Fast is to create a game that helps teach arithmetic to young children in elementary school while giving them an enjoyable and memorable experience. Specifically, Formula Fast will help the player learn addition, subtraction, multiplication, and division by starting with easy equations and progressively making the given equations more and more difficult as the user completes levels and progresses through the game.

References:

CS2212 Group Project Specification. UWO OWL. Version 1.0. https://owl.uwo.ca/access/content/group/aa2311c9-4cef-497f-8d9c-b502023be21c/project/CS2212%20Group%20Project%20Specification.pdf. Accessed 15 Jan. 2024.

Formula Fast Requirements Documentation. Version 2.2. https://wiki.csd.uwo.ca/display/COMPSCI2212W2024GROUP65/Formula+Fast?src=contextnavpagetreemode. Accessed 29 Feb. 2024.

Formula Fast Design Documentation. Version 1.7. https://wiki.csd.uwo.ca/display/COMPSCI2212W2024GROUP65/Design+Documentation+Introduction. Accessed 27 Mar. 2024.

Unit Testing Plan

Unit test classes will be written for all core classes, in this case those classes are AccountHandler, DataHandler, DebugHandler, LevelHandler, QuestionHandler, RacerHandler, and SceneHandler. Testing these core classes that handle the primary functionality of our software and contain the methods being called by other UI controller classes will allow us to test a variety of cases that could break the software as a whole.

For each class we aim to write a test case for each and every public method, testing the most common use of that method, as well as edge cases that may appear at run time to the best of our ability.

These unit tests allow us to ensure data will be handled and saved appropriately as a loss of data would result in the inability of the instructor to accurately track students. We are also testing the software's ability to accurately generate randomized questions suited to each level to ensure an increasing difficulty and a randomness to each level to avoid students memorizing question answers.

Integration Testing Plan

Test Case Name:	**Main Game Flow Initialization Test**
Test Case Description:	Test the initial game flow from countdown to the first question display after the game starts.
Test Steps:	1. Instantiate 'GameplayController' and trigger 'initialize()' method. 2. Verify that 'username_label' is set correctly. 3. Verify that 'level_label' is set to "Level_1". 4. Confirm the countdown starts and ends correctly. 5. Check that 'racer_handler' starts racers after the countdown. 6. Ensure that the first question is generated and displayed on "question_label".
Pre-Requisites:	AccountHandler must be initialized with a current user
Expected Results:	1. 'username_label' reflects the current user's name. 2. 'level_label' reads "Level 1". 3. 'countdown_timer_label' counts down from 5 to 0 and then is not visible. 4. Racers are initiated and moving after countdown. 5. A valid first question is displayed in"question_Label".
Test Category:	Integration Test
Requirement:	The game must initialize and display all elements properly before starting gameplay.
Automation:	Manually run by human
Date Run:	March 27th, 2023
Pass /Fail:	Pass
Test Results:	All tests steps completed successfully. The 'username_label' matched the current user's username, 'level_label' correctly displayed "Level 1", the countdown functioned correctly, 'racer_handler' successfully started the racers, and the first question was properly generated and displayed on "question_label".
Remarks:	None

Test Case Name:	**Level Completion Flow Test**
Test Case Description:	Test the transition from the successful completion of a level to the level passed screen, including score calculations and high score determination.
Test Steps:	1. Trigger the completed_level() method. 2. Verify that the playing flag is set to false. 3. Verify that the racer_handler stops all racers. 4. Check the calculated score and verify it's stored correctly. 5. Verify that the SceneHandler switches to the "LevelPassed" scene. 6. In the LevelPassedController, confirm that the score and high score display correctly.

Pre-Requisites:	The GameplayController must be in a state where the player is about to complete the last question of the level.
Expected Results:	1. The game stops the level and all racers come to a halt. 2. The score is calculated based on remaining time. 3. If a new high score is achieved, it's updated in LevelHandler. 4. The scene transitions to the "LevelPassed" scene. 5. The LevelPassedController displays the correct score and high score status.
Test Category:	Integration Test
Requirement:	The game must correctly handle level completion, score calculation, and transition to the level passed screen.
Automation:	Manually run by human
Date Run:	March 27th, 2023
Pass/Fail:	Pass
Test Results:	The game correctly transitioned from the final question of a level to the "LevelPassed" scene. The score was calculated as expected and matched the time remaining. The high score was correctly identified and updated if applicable. The "LevelPassed" screen displayed the accurate score and high score status.
Remarks:	None

Test Case Name:	**Leaderboard Display Test**
Test Case Description:	Test the leaderboard loading functionality, which includes retrieving top scores and displaying them on the leaderboard screen.
Test Steps:	1. Transition to the leaderboard scene from the main menu. 2. Trigger the load_leaderboard_content() method in LeaderboardController. 3. Verify that the DataHandler retrieves the sorted data for top scores. 4. Confirm that the leaderboard UI is updated with the top scores and corresponding usernames. 5. Test the return functionality to go back to the main menu from the leaderboard screen.
Pre-Requisites:	The application is in a state where the leaderboard can be accessed, and the DataHandler has a list of top scores available.
Expected Results:	1. Leaderboard scene is accessed from the main menu. 2. Top scores with usernames are displayed correctly in the leaderboard UI. 3. The user can return to the main menu using the provided UI control.
Test Category:	Integration Test
Requirement:	The leaderboard must accurately retrieve and display the top five scores from the database and allow users to return to the main menu.
Automation:	Manually run by human
Date Run:	March 28th, 2023
Pass/Fail:	Pass
Test Results:	The leaderboard content was loaded successfully, with the top five user scores and usernames displayed correctly. The return to the main menu functioned as expected, demonstrating the leaderboard's integration with the rest of the application.
Remarks:	None

Test Case Name:	**Login To Main Menu Flow Test**
Test Case Description:	Test the login process, including user authentication and transition from the login screen to the main game menu.
Test Steps:	1. Start the application to load the LoginController. 2. Input valid username and password credentials into the login interface. 3. Trigger the login event to authenticate the user.Verify that AccountHandler authenticates the credentials and sets the current user. 4. Confirm that upon successful authentication, SceneHandler transitions to the main menu scene.
Pre-Requisites:	The LoginController must be loaded, and the test environment should have a mock user account for authentication.
Expected Results:	1. The provided credentials are accepted, and the user is logged in. 2. The current user is set in the AccountHandler. 3. The scene changes to the main menu after successful login.
Test Category:	Integration Test
Requirement:	The game must authenticate the user and transition to the main menu upon successful login.
Automation:	Manually run by human
Date Run:	March 27th, 2023
Pass/Fail:	Pass
Test Results:	The login process completed successfully with the mock user account. The AccountHandler correctly set the current user session, and the SceneHandler transitioned the application to the main menu scene, confirming the application handles the login process correctly.
Remarks:	None

Test Case Name:	**Level Failure Flow Test**
Test Case Description:	Test the flow from failing a level in the game due to running out of time to displaying the level failed screen.
Test Steps:	1. Set up the GameplayController with a level in progress and the timer close to running out. 2. Force the timer to expire and trigger the fail_level() method. 3. Verify that the playing flag is set to false.Verify that the racer_handler stops all racers. 4. Check that SceneHandler switches to the "LevelFailed" scene. 5. In the LevelFailedController, confirm that any necessary cleanup or setup for retrying the level is handled.
Pre-Requisites:	The GameplayController must be in a state where the level is active and the time is about to run out.

Expected Results:	1. The game should detect the failure condition and halt gameplay. 2. All racers are stopped to indicate the end of the level. 3. The game transitions to the "LevelFailed" scene. 4. The LevelFailedController is ready to guide the player to retry the level or return to the main menu.
Test Category:	Integration Test
Requirement:	The game must correctly handle a level failure due to time running out and transition to the level failed screen.
Automation:	Manually run by human
Date Run:	March 26th, 2023
Pass /Fail:	Pass
Test Results:	The gameplay was halted as expected when the timer ran out, and the racers stopped moving. The screen transitioned to the "LevelFailed" scene without issues. No unexpected behavior was observed, and the player was given the option to retry or exit to the menu, indicating the game's robust handling of failure scenarios.
Remarks:	None

Test Case Name:	**Instructor Mode Access Test**
Test Case Description:	Test the access to instructor mode from the login page and the functionality of displaying a player's progress within instructor mode.
Test Steps:	1. Log in as an instructor from the login page. 2. Transition to the instructor mode UI. 3. Enter a username into the search field and trigger the search event. 4. Verify that the DataHandler retrieves the correct data for the input username. 5. Confirm that the InstructorModeController updates the UI with the player's data, including level scores and total score. 6. Test the exit functionality to return to the login page from instructor mode.
Pre-Requisites:	The login functionality must be working, and there should be an instructor account and at least one player account with progress data.
Expected Results:	1. Instructor logs in successfully and accesses instructor mode. 2. Upon searching for a username, their progress data is displayed correctly. 3. The total score and individual level scores are accurately reflected in the UI. 4. The instructor can exit back to the login page successfully.
Test Category:	Integration Test
Requirement:	Instructor mode must retrieve and display user progress correctly and allow the instructor to exit back to the login page.
Automation:	Manually run by human
Date Run:	March 27th, 2023
Pass /Fail:	Pass

Test Results:	The login as an instructor was successful, and the transition to the instructor mode UI was seamless. A player's username was entered, and their game progress, including scores for each level and the total score, was displayed accurately. Exiting instructor mode returned the application to the login page as expected.
Remarks:	None

Test Case Name:	**Tutorial Sequence Test**
Test Case Description:	Test the sequential display of tutorial parts and the ability to exit to the main menu after completion.
Test Steps:	1. Load the tutorial scene. 2. Simulate the button clicks for each part of the tutorial. 3. Verify that each part becomes visible only after the preceding part's button is clicked. 4. Upon clicking the final part, check that the finish button appears. 5. Simulate the finish button click and verify the transition to the main menu.
Pre-Requisites:	AccountHandler must be initialized with a current user
Expected Results:	1. Each part of the tutorial is shown in sequence as the user clicks through. 2. The finish button becomes visible after the last part is displayed. 3. The main menu scene is loaded upon clicking the finish button.
Test Category:	Integration Test
Requirement:	The tutorial must guide the user through each step and then return to the main menu upon completion.
Automation:	Manually run by human
Date Run:	March 28th, 2023
Pass/Fail:	Pass
Test Results:	The tutorial operated correctly, with each part appearing after the corresponding button click. The finish button was displayed at the end of the tutorial, and the click event correctly transitioned to the main menu.
Remarks:	None

Test Case Name:	**Debug Mode Toggle Test**
Test Case Description:	Test the toggling of debug mode from the main menu and its effects on level button labels and the debug text label.
Test Steps:	1. Load the main menu scene. 2. Simulate the key press event for the debug mode toggle (Key 'P'). 3. Verify that the debug mode is enabled in the DebugHandler. 4. Check that the text label for debug information becomes visible. 5. Verify that all level buttons show as unlocked regardless of actual game progress. 6. Simulate the key press event to toggle debug mode off. 7. Confirm that the debug mode is disabled. 8. Ensure the debug text label is not visible and level buttons reflect the actual unlock status.
Pre-Requisites:	AccountHandler must be initialized with a current user

Expected Results:	1. The debug mode toggles on and off with the key press. 2. When debug mode is on, all levels appear unlocked, and debug text is visible. 3. When debug mode is off, the actual unlock status of levels is shown, and debug text is hidden.
Test Category:	Integration Test
Requirement:	Debug mode should correctly affect the visibility of debug-related UI elements and level access.
Automation:	Manually run by human
Date Run:	March 28th, 2023
Pass/Fail:	Pass
Test Results:	The debug mode toggled on and off correctly, influencing the visibility of the debug label and the lock status of level buttons as expected. Key presses were captured appropriately, and the UI updated in accordance with the debug mode status.
Remarks:	None

Test Case Name:	**Ending Scene Navigation Test**
Test Case Description:	Test the functionality of the ending scene buttons, including transitions to the leaderboard, main menu, and the game exit process.
Test Steps:	1. Load the ending scene. 2. Simulate the click event on the "Leaderboard" button and verify the scene transition. 3. Return to the ending scene and simulate the click event on the "Main Menu" button, then verify the scene transition. 4. From the ending scene, simulate the click event on the "Save & Exit" button and verify the application's exit behavior, including any data saving process.
Pre-Requisites:	The ending scene FXML is correctly set up with EndingController, and the game data is in a state that triggers the ending scene upon completion.
Expected Results:	1. Clicking the "Leaderboard" button transitions to the leaderboard scene. 2. Clicking the "Main Menu" button transitions back to the main menu. 3. Clicking the "Save & Exit" button saves the current game data and exits the application.
Test Category:	Integration Test
Requirement:	The ending scene must offer navigational functionality to the leaderboard, main menu, and facilitate game data saving upon exit.
Automation:	Manually run by human
Date Run:	March 28th, 2023
Pass/Fail:	Pass
Test Results:	Each button on the ending scene functioned as expected. The transitions to the leaderboard and main menu were successful, and the "Save & Exit" button's functionality was confirmed through the application's closure and verification of saved data.
Remarks:	None

All of these integration tests are top-down.

Validation Testing Plan

Test Case Name	Functional Requirement: Graphical User Interface (GUI)
Test Case Description	Ensure the software has a graphical user interface that allows the users to interact with the system
Test Steps	1. Launch the software 2. Navigate through all screens and interact with all buttons, text fields, etc.
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	All screens are displayed properly and all interactive elements properly respond to user input
Test Category	Validation Test
Requirement	The software must include a graphical user interface that allows users to interact with the system and responds to all user inputs
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	All screens were displayed properly and all interactive elements properly responded to user input
Remarks	None

Test Case Name	Functional Requirement: Multiple Screens
Test Case Description	Ensure the software has five distinct pages (main menu/login, gameplay, instructions/tutorial, high score, progress)
Test Steps	1. Launch the software 2. Progress through/play the game 3. Count the number of different screens the game displays 4. Interact with each screen to verify they function properly
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	All five screens, along with their elements, are clearly displayed and function properly
Test Category	Validation Test
Requirement	The software must properly display the five desired screens, with each screen and its elements functioning as intended
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	All five desired screens were properly displayed, with each screen and its elements functioning as intended
Remarks	None

Test Case Name	Functional Requirement: Mouse-Based Interaction
Test Case Description	Ensure the software supports mouse-based interactions allowing the user to navigate through the games menu, make selections, and control some game elements with the mouse
Test Steps	1. Launch the software 2. Perform multiple mouse-based interactions (navigating through menus, making selections, controlling game elements) 3. Verify proper functionality and responses
Pre-Requisites	The software runs/opens properly, is ready for testing, and a mouse is connected
Expected Results	The software recognizes all mouse-based interactions and properly responds to each click
Test Category	Validation Test

Requirement	The software must support mouse-based interactions by allowing the user to perform multiple actions with their mouse properly
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software properly responded to all mouse-based interactions
Remarks	None

Test Case Name	**Functional Requirement: Feedback System**
Test Case Description	Ensure the software provides visual or auditory feedback in response to user actions
Test Steps	1. Launch the software 2. Perform multiple actions in the game that should provide the user with visual/auditory feedback 3. Verify the software provides visual or auditory feedback in response to the user's actions
Pre-Requisites	The software runs/opens properly, is ready for testing, and includes multiple actions that could provide feedback
Expected Results	The software provides the user with visual/auditory feedback given correct/incorrect inputs in the gameplay, or any other aspects of the software
Test Category	Validation Test
Requirement	The software must provide the user with visual/auditory feedback in response to their actions in certain parts of the game
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software provides users with visual feedback in response to correct/incorrect answers in the gameplay, any incorrect passwords /login input, or invalidly creating a new account
Remarks	None

Test Case Name	**Functional Requirement: Main Menu**
Test Case Description	Ensure the software has a main menu/login screen that includes options to start a new game, load a save file, view instructions, view high scores, and exit the game
Test Steps	1. Launch the software 2. Verify the game has a main menu/login screen 3. Verify the screen has options to start a new game, load a save file, view instructions, view high scores, and exit the game 4. Interact with each option to verify their functionality
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	The main menu/login screen is properly displayed with all options displayed and functioning properly
Test Category	Validation Test
Requirement	The software must have a main menu/login screen that includes options to start a new game, load a save file, view instructions, view high scores, and exit the game
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software has 2 pages, a login screen and a menu screen, which gives the user the option to create a new account (start a new game) that leads to the tutorial/instructions, login (load save file), view the leaderboard, and logout

Remarks	None

Test Case Name	**Functional Requirement: Instructions/Tutorial**
Test Case Description	Ensure the software includes an instructional/tutorial screen with detailed instructions on playing the game
Test Steps	1. Launch the software 2. Create a new account 3. Verify the tutorial provides useful information to a first-time user 4. Verify the user can progress through and exit the tutorial screens
Pre-Requisites	The software runs/opens properly, is ready for testing, and a new account has been created
Expected Results	The tutorial has multiple screens the user can progress through at their own pace that provide valuable information about the software's gameplay
Test Category	Validation Test
Requirement	The software must include an instructional/tutorial screen with detailed instructions on playing the game
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software has a short tutorial that is shown after the user creates a new account, and provides the new player with information about the gameplay
Remarks	None

Test Case Name	**Functional Requirement: Scoring System**
Test Case Description	Ensure the software tracks the score for each level (score calculated from time taken to complete a level)
Test Steps	1. Launch the software 2. Create a new account/login to your account 3. Complete a level 4. Verify a score is calculated after completion of a level
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	The software calculates a score upon level completion based on the time it took to complete the level
Test Category	Validation Test
Requirement	The software must track the score for each level (score calculated from time taken to complete a level)
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software displays the user's score upon completion of each level, with the score being based on how fast they complete the level
Remarks	None

Test Case Name	**Functional Requirement: Progression System**
Test Case Description	Ensure the software includes at least 3 levels of progressing difficulty for the player to progress through
Test Steps	1. Launch the software 2. Create a new account & complete the tutorial 3. Ensure the game has at least 3 levels, and they get progressively more difficult

Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	The software includes at least 3 levels, and each level is more difficult than the previous level
Test Category	Validation Test
Requirement	The software must include at least 3 levels of progressing difficulty for the player to progress through
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software has 6 levels the player can progress through, with each level being more difficult than the last
Remarks	None

Test Case Name	**Functional Requirement: Save/Load Game State**
Test Case Description	Ensure the software allows players to save their progress locally to their system and reload that progress at a later point, progress will automatically save after a level is complete
Test Steps	1. Launch the software 2. Create a new account, complete the tutorial, and complete the first-level 3. Logout of the game and then log back in 4. Verify that the user's progress has been saved
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	The software automatically saves progress upon level completion and allows players to continue where they left off
Test Category	Validation Test
Requirement	The software must allow players to either save their progress locally to their system or have the system automatically save upon level completion, so they can reload that progress at a later point
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software automatically saves the user's progress/data upon level completion, and lets them continue where they left off once they log back in
Remarks	None

Test Case Name	**Functional Requirement: Leaderboard**
Test Case Description	Ensure the software includes a leaderboard that displays the users with the top 5 highest scores
Test Steps	1. Launch the software 2. Login/create a new account 3. Navigate to menu 4. Click leaderboard button 5. Verify the top 5 highest total scores, along with each respective username, are properly displayed
Pre-Requisites	The software runs/opens properly, is ready for testing, and the user is in the menu
Expected Results	The leaderboard displays the users with the top 5 highest scores, along with their usernames. It also properly updates the leaderboard when a new user gets a total score that is higher than any of the current scores on the leaderboard, and properly sorts the leaderboard each time

Test Category	Validation Test
Requirement	The software must include a leaderboard that displays the users with the top 5 highest scores
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software has a properly functioning leaderboard that displays the users with the top 5 highest scores, which properly updates with a new top score
Remarks	None

Test Case Name	**Functional Requirement: Multiple Players**
Test Case Description	Ensure the software supports multiple players, creating new user accounts when a new game is started
Test Steps	1. Launch the software 2. Create a new account 3. Logout of the current account 4. Create another new account 5. Verify that you can log into both accounts
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	The software can support multiple players/new accounts being made and properly saves each account's respective data
Test Category	Validation Test
Requirement	The software must support multiple players, creating new user accounts when a new game is started
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software properly handles multiple players and saves the data of each new account that is created
Remarks	None

Test Case Name	**Functional Requirement: Instructor Mode**
Test Case Description	Ensure the software includes an “instructor mode” that shows a view of the level progression and level scores for every player registered to the system
Test Steps	1. Launch the software 2. Enter the correct instructor mode password 3. Verify the software properly handles all possible outcomes after searching for a name
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	If the instructor enters the wrong instructor mode password, a message is displayed saying so. If they enter the correct password, they are sent to instructor mode. When they search for a valid username, that user's current level, score for each individual level, and total score are displayed.
Test Category	Validation Test
Requirement	The software must include an “instructor mode” that shows a view of the level progression and level scores for every player registered to the system

Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software includes an “instructor mode” that shows a view of the level progression and level scores for every player registered to the system
Remarks	None

Test Case Name	**Functional Requirement: Debug Mode**
Test Case Description	Ensure the software includes a debug mode for development and testing allowing developers to skip to specific levels by unlocking all the levels
Test Steps	1. Launch the software 2. Login/create a new account 3. Execute the correct input to enter debug mode 4. Verify all levels of the game have been unlocked
Pre-Requisites	The software runs/opens properly, is ready for testing, and the user is in the menu
Expected Results	The user clicks the correct key on their keyboard to enter instructor mode, which then displays a message verifying that debug mode is active and unlocks all 6 levels of the game
Test Category	Validation Test
Requirement	The software must include a debug mode for development and testing allowing developers to skip to specific levels by unlocking all the levels
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software includes a debug mode for development and testing allowing developers to skip to specific levels by unlocking all the levels
Remarks	None

Test Case Name	**Functional Requirement: Housekeeping & Error Handling**
Test Case Description	Ensure the software handles data properly when exiting, and all errors must be handled appropriately and displayed to the user
Test Steps	1. Launch the software 2. Provide multiple operations that involve data handling 3. Intentionally include errors (wrong password/username, including black fields, creating an account with a username that already exists, etc.) 4. Verify all errors display proper messages to the user
Pre-Requisites	The software runs/opens properly & is ready for testing
Expected Results	The software properly handles all input data from the user, and all errors are handled appropriately with corresponding messages being displayed given the error
Test Category	Validation Test
Requirement	The software must handle data properly when exiting, and all errors must be handled appropriately and displayed to the user
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software handles data properly when exiting, with any errors being handled appropriately and displayed to the user
Remarks	None

Test Case Name	**Functional Requirement: Display Mathematical Equations and Validate Input For Structure & Correctness**
Test Case Description	Ensure the software displays mathematical equations during gameplay, takes and validates responses from the player during gameplay, and then clearly displays whether an answer is correct or incorrect
Test Steps	1. Launch the software 2. Login/create a new account 3. Navigate to any level available 4. Once the level starts, verify a mathematical equation is displayed above the text field 5. Verify that there is a clear display of whether the input answer was correct/incorrect
Pre-Requisites	The software runs/opens properly, is ready for testing, and the user is at a gameplay level
Expected Results	The software displays a mathematical equation at the start of the game, with a new question appearing after every correct answer, with the software also validating and displaying whether the answer is correct/incorrect
Test Category	Validation Test
Requirement	The software must display mathematical equations during gameplay, take and validate responses from the player during gameplay, and then clearly display whether an answer is correct or incorrect
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software properly displays a mathematical equation at the start of the game, with a new question being displayed after each question is correctly answered. The current question counter to the right of the equation also lights up green if the answer is correct, and red if incorrect
Remarks	None

Test Case Name	**Functional Requirement: Return To Main Menu / Level Navigation**
Test Case Description	Ensure the software allows users to navigate between levels by returning to the menu upon completion
Test Steps	1. Launch the software 2. Login/create a new account 3. Navigate to an available gameplay level 4. Complete the level 5. Verify the "Continue" returns the user to the menu for each level
Pre-Requisites	The software runs/opens properly, is ready for testing, and the user is at a gameplay level
Expected Results	Upon level completion for each level, clicking the Continue button navigates the user back to the menu where all the other levels are located
Test Category	Validation Test
Requirement	The software must allow users to navigate between levels by returning to the menu upon completion
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software allows users to navigate between levels by returning to the menu upon completion after clicking the Continue button
Remarks	None

Test Case Name	**Functional Requirement: High Score Notification**

Test Case Description	Ensure the software includes a high score message that will display the user's high scores for each level individually
Test Steps	1. Launch the software 2. Login/create a new account 3. Navigate to an available level 4. Complete the level 5. Verify the software displays you've gotten a new high score after the first completion 6. Re-run the same level, get a higher score, and verify the high score message is displayed again
Pre-Requisites	The software runs/opens properly, is ready for testing, and the user has an account they can complete a level on
Expected Results	For each level of the game, a high score message is displayed and the user's high score is updated if the score they just recorded is higher than their current high score
Test Category	Validation Test
Requirement	The software must include a high score message that will display the user's high scores for each level individually
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software properly calculates whether the user has gotten a new high score for either of the levels, and displays a message upon level completion indicating so
Remarks	None

Test Case Name	**Functional Requirement: Computer-Controlled Racers**
Test Case Description	Ensure the software has computer-controlled racers to complete with the player during the level, the level is only considered completed if the user beats the computer racers
Test Steps	1. Launch the software 2. Login/create a new account 3. Navigate to an available level 4. Verify that there are 3 other cars, which are controlled by the computer, and function as intended
Pre-Requisites	The software runs/opens properly, is ready for testing, and the user is at a gameplay level
Expected Results	Each level in the game has 3 computer-controlled cars that the user needs to beat to the finish line to complete the level. Each car moves forward randomly, and the user will fail the level if one of the 3 cars reaches the finish line before them
Test Category	Validation Test
Requirement	The software must have computer-controlled racers to complete with the player during the level, the level is only considered completed if the user beats the computer racers
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software has 3 computer-controlled cars on each level, with each car moving forward at a random pace, with the user falling the level if one of the 3 cars reaches the finish line before them
Remarks	None

Test Case Name	**Functional Requirement: Display Level Completion and Unlocking**
Test Case Description	Ensure the software clearly displays when a level has been completed, and when a new level has been unlocked

Test Steps	1. Launch the software 2. Login/create a new account 3. Navigate to an available level 4. Complete the level 5. Verify the software displays whether the level has been completed and if a new level has been unlocked
Pre-Requisites	The software runs/opens properly, is ready for testing, and the user is at a gameplay level
Expected Results	The software displays whether the current level has been passed/failed, with the next level clearly being unlocked upon return to the main menu when a level is passed
Test Category	Validation Test
Requirement	The software must clearly display when a level has been completed, and when a new level has been unlocked
Automation	Manually run by human
Data Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The software clearly displays when a level has been passed/failed, with the "LOCKED" text over the next level's button being removed
Remarks	None

System Testing Plan

Test Case Name:	Windows 10
Test Case Description:	Ensure the software runs on Windows 10
Test Steps:	1. Go on a device that uses Windows 10 as its operating system 2. Run the program 3. Ensure it runs without any errors/bugs/exceptions
Pre-Requisites:	A device that uses Windows 10 as its operation system
Expected Results:	Software will run free of errors/bugs/exceptions.
Test Category:	System Test
Requirement:	Software must run on Windows 10
Automation:	No
Date Run:	29 Mar 2024
Pass/Fail:	Pass
Test Results:	The program ran on Windows 10 perfectly without any bugs, errors, exceptions, etc.
Remarks:	None

Test Case Name:	Pass Level
Test Case Description:	Verify that successfully completing a level unlocks the next level.
Test Steps:	1. Create a new player account 2. Start Level 1 3. Pass Level 1 4. Verify that Level 2 is unlocked
Pre-Requisites:	The account to be created does not already exist.
Expected Results:	The next level will be unlocked.
Test Category:	System Test
Requirement:	Players must progress to the next level upon passing the current level.
Automation:	No
Date Run:	29 Mar 2024
Pass/Fail:	Pass
Test Results:	After passing the first level, the next level was unlocked successfully.
Remarks:	None

Test Case Name:	Fail Level
Test Case Description:	Verify that failing a level does **not** unlock the next level and keeps it locked
Test Steps:	1. Create a new player account 2. Start Level 1 3. Fail Level 1 4. Verify that Level 2 is locked
Pre-Requisites:	The account to be created does not already exist.

Expected Results:	The next level will be locked.
Test Category:	System Test
Requirement:	Players must retry the current level after failing it as the next level is still locked.
Automation:	No
Date Run:	29 Mar 2024
Pass/Fail:	Pass
Test Results:	After failing the first level, the next level was still locked.
Remarks:	None

Test Case Name:	**Leaderboard Update**
Test Case Description:	Verify that the leaderboard accurately and promptly updates its info after the user completes a level
Test Steps:	1. Create a new player account 2. Complete Level 1 with a high score 3. Check leaderboard for an update
Pre-Requisites:	▪ The account to be created does not already exist. ▪ All users on the leaderboard have only completed and passed one level
Expected Results:	Leaderboard accurately reflects the new user and their high score.
Test Category:	System Test
Requirement:	Leaderboard must update accurately and promptly.
Automation:	No
Date Run:	29 Mar 2024
Pass/Fail:	Pass
Test Results:	After completing the first level with a high score, the leaderboard updated accurately and promptly with the new user and their high score.
Remarks:	None

Test Case Name:	**Instructor Mode Functionality**
Test Case Description:	Verify that instructor(s) can access and view the progress of any player
Test Steps:	1. Log in as an instructor 2. Search for a specific player 3. View the player's progress and ensure the data is accurate
Pre-Requisites:	Player accounts with progress exist.
Expected Results:	Instructor successfully logs in and views player progress.
Test Category:	System Test
Requirement:	Instructor(s) must be able to view player progress.
Automation:	No
Date Run:	29 Mar 2024
Pass/Fail:	Pass
Test Results:	Instructor was able to login, search for an existing player, and view their progress.

Remarks:	None

Test Case Name:	**Create Account Functionality**
Test Case Description:	Verify that the user can create an account and login successfully.
Test Steps:	1. Create account 2. Verify that the account is created successfully, user is navigated to tutorial, and the user is logged in
Pre-Requisites:	The account to be created does not already exist.
Expected Results:	Account is successfully created and player is navigated to the tutorial and is successfully logged in.
Test Category:	System Test
Requirement:	Create Account must successfully create an account, navigate user to tutorial, and login.
Automation:	No
Date Run:	29 Mar 2024
Pass/Fail:	Pass
Test Results:	Account was successfully created and the user was navigated to the tutorial page and was successfully logged in.
Remarks:	None

Test Case Name:	**Login/Logout Functionality**
Test Case Description:	Verify that the user can successfully login and logout of their account
Test Steps:	1. Log in with existing account 2. Verify that the user is logged in 3. Logout of the account 4. Verify that the user is logged out
Pre-Requisites:	The account to be logged into must already exist.
Expected Results:	User successfully logs in and out of the account.
Test Category:	System Test
Requirement:	Login must correctly log the user into their session and Logout must correctly log out the user and terminate their session.
Automation:	No
Date Run:	29 Mar 2024
Pass/Fail:	Pass
Test Results:	User successfully logged into their account, and then successfully logged out of their account.
Remarks:	None

Testing Documentation Summary

In conclusion, this testing documentation outlines a comprehensive test plan for our software system, Formula Fast. The test plan focuses on unit testing, integration testing, validation testing, and system testing. This plan prioritises the use of white box/structural testing and black box/functional testing in order to ensure that our software meets all of the functional and non-functional requirements. Also, for the unit testing portion we are using JUnit to write unit tests for our Java code, and making sure the unit tests cover all the major functions in the code. Through this test plan we aim to provide a sufficient and complete testing strategy to assess the quality of our software.

Gamified Learning: The gamification of learning is an educational approach that seeks to motivate students by using video game design and game elements in learning environments

IDE: An Integrated Development Environment is a software application that helps programmers develop software code efficiently

GUI: A Graphical User Interface is a digital interface in which a user interacts with graphical components (Ex. buttons, menus, etc.)

White Box/Structural Testing: Testing focused on control flow, decision paths, and boundary cases

Black Box/Functional Testing: Testing focused on assessing whether the software meets all of its functional and non-functional requirements