Team Contract

Meetings:

1. Meetings will be held via video call via Discord or Zoom every Monday at 3:00 p.m. EST.
2. The length of the meeting will be roughly an hour.
3. The discussion will be about what members accomplished in the previous week, and tasks will be divided among team members for the coming week.
4. Meeting minutes for group meetings and TA meetings will rotate between each member on a weekly basis.

Work Norms:

1. Team members are expected to work roughly 5-7 hours a week, or however long is required to complete the assigned tasks, to make sure the team does not fall behind schedule.
2. Team deadlines will be weekly; tasks being assigned in the Monday meeting will be due in the following Monday meeting.
3. All work will be reviewed by another team member for code cleanliness, correctness, as well as organization.
4. In cases where there are different opinions, decisions will be decided through voting, with the majority winning.
5. The IDE that will be used for the project will be in VScode with the main language of Java.

Work Division:

1. Work will be divided by case classes or use cases, depending on which is easier for the current task set.
2. When assigning weekly tasks, who takes on the responsibility of a certain issue will be up to the discretion of the group.
3. Each team member is responsible for notifying the team of any delays or issues as soon as they arise.
4. As the project progresses, team members agree to remain flexible and open to renegotiating task assignments and deadlines when necessary.
5. If a team member is unable to perform their duties due to an emergency, their tasks will be temporarily reassigned to other team members to ensure project progress.

Formula Fast

Requirements Documentation

Version History

Version	Date	Author(s)	Summary of Changes
0.1	29 Jan 2024	Ashton David Ryan Franklin Que Hung Dang Matthew Marbina Gabriel Azhari Mahmoud Sherif Mohamed Radwan	Created initial pages for team contract, requirements documentation, and meeting minutes.
0.2	30 Jan 2024	Ashton David Ryan Franklin	Added a list of functional requirements to the Functional Requirements page.
0.3	01 Feb 2024	Ashton David Ryan Franklin	Modifications to the functional requirements list. Created the instructor actor. Modifications to the student actor.
0.4	04 Feb 2024	Mahmoud Sherif Mohamed Radwan	Added the Domain Analysis page for the Requirements Documentation
0.5	05 Feb 2024	Matthew Marbina	Added all 20 use case templates to the Functional Requirements page
0.6	05 Feb 2024	Gabriel Azhari	Added the overview, objectives, and references information to the Introduction page.
0.7	05 Feb 2024	Mahmoud Sherif Mohamed Radwan	Modified the Domain Analysis page: improved wordiness/sentence flow, added new content, made it more coherent and complete, overall.
0.8	06 Feb 2024	Ashton David Ryan Franklin	Added the administrator actor template to the functional requirements section.
0.9	06 Feb 2024	Ashton David Ryan Franklin	Merged use cases 15 and 16. Uploaded activity diagrams for use cases 13-15.
1.0	06 Feb 2024	Ashton David Ryan Franklin	Modified use case 16. Uploaded activity diagram for use case 16.
1.1	07 Feb 2024	Ashton David Ryan Franklin	Uploaded the non-functional requirements list.
1.2	08 Feb 2024	Gabriel Azhari	Modified use case 8 and uploaded the activity diagrams for use cases 5-8.
1.3	08 Feb 2024	Ashton David Ryan Franklin	Uploaded summary document draft. Added section for use case diagram to functional requirements page.
1.4	09 Feb 2024	Mahmoud Sherif Mohamed Radwan	Fixed small typo on functional requirements use case #11
1.5	09 Feb 2024	Matthew Marbina	Removed functional requirement 16 from the functional requirement list, as it was combined with functional requirement 15, and fixed small typos in use case models #13 & #15.
1.6	09 Feb 2024	Matthew Marbina	Added the use case diagram to functional requirements page
1.7	09 Feb 2024	Mahmoud Sherif Mohamed Radwan	Uploaded the activity diagrams for use cases 9-12 in the functional requirements page.
1.8	09 Feb 2024	Matthew Marbina	Uploaded activity diagrams for use cases 17-19 in the functional requirements page, also slightly modified and updated the use case diagram in the functional requirements page
1.9	10 Feb 2024	Que Hung Dang	Uploaded the activity diagram for use cases 1-4 in the functional requirements page. Also finalized the team contract one last time before submission.
2.0	12 Feb 2024	Mahmoud Sherif Mohamed Radwan	Updated the activity diagrams for 9-12 (fixed error with the decision nodes).
2.1	12 Feb 2024	Gabriel Azhari	Modified the introduction and added some term definitions to the summary.
2.2	12 Feb 2024	Matthew Marbina	Modified error with final nodes in activity diagrams 17-19 and re-uploaded them to the functional requirements page.

Requirements 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 be timed to see how long it takes them to cross the finish line. 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.

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.

Domain Analysis

What is the software domain for this project?

The software domain for this project is educational games, with a specific focus on interactive learning games. The project combines elements of game design with educational content, with the goal of creating an engaging learning environment for our users.

Who is the target audience for the game?

The target audience for this game is elementary school students as our game will be focusing on improving basic mathematical arithmetic skills (addition, subtraction, multiplication, division) for our users.

Does this project fall into a specific subdomain or category within this software domain?

This project falls into the subdomain of mathematical educational games as the main goal of this project is to provide a fun and interactive learning environment for our users to learn basic mathematical arithmetic. It also falls into this subdomain as its content is math-oriented.

What do we know about the domain?

- Engagement is very crucial
    - Our users are more likely to learn and gain the benefits of our projects if the learning process is engaging.
    - We will lose a majority of our user base if our game is not engaging as there will be no incentive for our users to keep playing.
- Immediate feedback is a must
    - Providing immediate feedback to our users helps them correct their mistakes and do better and thus, continuing to play our game
- “Gamification” in Education
    - Gamification is the act of incorporating elements of game design, such as levels, leaderboards, points, etc. into non-gaming environments (educational content in this case). This phenomenon has shown to increase engagement for users.

What are the common issues encountered in this domain (list at least 3)?

- Maintaining engagement
    - Keeping our target audience (elementary school students) engaged with an educational game can be challenging as this demographic can lose interest quickly, especially as the novelty of the game wears off.
- Balancing Difficulty
    - Not every student is at the same level, so it will be difficult to create a game that is of equal difficulty -or at least playable- by all users in our target demographic.
- Accessibility
    - While accessibility is a must-have with all applications, an educational game can prove to be very difficult to make accessible. This is because it relies heavily on engagement which can be hard to provide equally for users with accessibility issues.

What are the common solutions to the above issues in this domain?

- Increasing difficulty with each level
    - Increasing the difficulty with each level will engage the user to keep playing as they will be challenged more and more and will want to keep improving.
- Keeping track of high scores
    - Keeping track of high scores (i.e. a leaderboard) gives the user an incentive to keep playing and try to beat their high score; thus, increasing engagement.
- Feedback, Encouragement, and Positive Reinforcement.
    - Providing feedback for users when they answer incorrectly, coupled with encouragement and recognizing their efforts can help our users learn from their mistakes and continue playing our game.
    - Putting the user in an environment where they are competing with their own high scores, and trying to improve based on their own previous attempts instead of comparing them with other players can keep our users focused on their own goals instead of comparing themselves with others and getting discouraged.
- Providing Accessibility while keeping engagement
    - By making the game playable for all users whilst also keeping the same level of engagement for everybody, we must design the game in a way in which it plays similarly to everyone. For example, some players can use text to answer each question, whilst others can use their mouse. We must also consider colour-blindness, ensuring UI elements are in a logical tab order, as well as avoiding flashing colours and keeping our overall game-design accessible for everybody

How can we use this domain understanding to improve or accelerate development of this project?

We can use this domain understanding to improve/accelerate the development of this project by utilizing studies and findings in the educational games field to design game elements that are proven to enhance learning and engagement whilst also maintaining accessibility. We can also use this domain understanding as a major influence in our game-designing stage, as the game’s design and UI plays a big factor into how it is perceived by our target audience. We can also reference the best practices used in this domain and subdomain to see what has proven to work in the future and utilize these findings in our game design and/or development. Finally, by leveraging the information acquired, existing solutions, and best practice, we can reduce the time and effort taken in the development of this project; thus, improving our efficiency as a software development team.

Functional Requirements

Overview

Table of Contents

Functional Requirements List

1. The software must have a graphical user interface that allows the users to interact with the system.
2. The software must have at least five distinct pages, a main menu, a gameplay screen, an instructions screen, a high score list, and a progress screen
3. The software must support mouse-based interactions allowing the user to navigate through the games menu, make selections, and control some game elements with the mouse.
4. The UI must provide visual or auditory feedback in response to user actions.
5. The software must have a main menu screen that includes options to start a new game, load a save file, view instructions, view high scores, and exit the game.
6. The software must include an instructional screen with detailed instructions on playing the game.
7. Software must track the score for each level (score calculated from time taken to complete a level)
8. Software must include at least 3 levels of progressing difficulty for the player to progress through.
9. The software must allow 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. Confirmation messages must be displayed when progress Is saved or loaded.
10. The software must include a high score section that will display the high scores for each level individually.
11. The software must support multiple players, creating new user accounts when a new game is started.
12. 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.
13. The software must include a debug mode for development and testing allowing developers to skip to specific levels or unlock levels during run time.
14. Software must handle data properly when exiting, and all errors must be handled appropriately and displayed to the user.
15. Software must display mathematical equations during gameplay, take and validate responses from the player during gameplay, and then clearly display whether an answer was correct, incorrect, or invalid.
16. The software must allow users to navigate between levels by returning to the menu upon completion.
17. The software must clearly state when a new personal or overall high score is met on a level.
18. The software must have computer-controlled racers to complete with the player during the level, level is only considered completed if the user beats the computer racers.
19. The software must clearly display when a level has been completed, and when a new level has been unlocked.

Actor Templates

Actor:	Player
Description:	The user who is currently playing the game, a student in the class the instructor is monitoring.
Aliases:	Student
Inherits:	N/A
Actor Type:	Person
Active / Passive:	Active

Actor:	Instructor
Description:	The instructor monitoring the progression of students using the game.
Aliases:	Teacher
Inherits:	Player

Actor Type:	Person
Active / Passive:	Active

Actor:	Administrator
Description:	Someone involved in the development of the software that needs to use debugging mode.
Aliases:	Developer, Tester
Inherits:	Teacher
Actor Type:	Person
Active / Passive:	Active

Use Case Templates

1.

Use Case Name:	Graphical User Interface (GUI)
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	Allow the player to interact with and navigate the system easily
Preconditions:	The GUI is properly displayed on the computer screen
Trigger:	Player launches the game and attempts to click a button/key
Scenario:	1. Player launches game 2. GUI is properly displayed on screen 3. Player clicks mouse or keyboard key 4. GUI properly responds to player’s input
Alternatives:	None
Exceptions:	1. Game does not properly launch 2. GUI not fully displayed on screen 3. GUI does not properly respond to player’s input
Priority:	Highest

2.

Use Case Name:	Multiple Screens
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To display & allow the user to navigate between five different game screens
Preconditions:	The game is properly launched, and the GUI properly works
Trigger:	Player attempts to navigate to a game screen
Scenario:	1. Player launches game 2. Player selects one of the five screens to navigate to 3. The corresponding game screen is properly displayed

Alternatives:	1. Player selects wrong game screen and goes back 2. Player selects new game screen while already in the game 3. Player refreshes browser
Exceptions:	A game screen is not properly displayed after player navigates to it
Priority:	High

3.

Use Case Name:	Mouse-Based Interactions
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	Allow players to use their mouse/keyboard to interact with the games GUI to perform multiple operations
Preconditions:	The game us properly launched and the player has a working mouse/keyboard
Trigger:	Player tries to navigate game screens, make selections, or control game elements with their mouse
Scenario:	1. Player launches game 2. Player navigates to one of the game screens with their mouse 3. Player makes a selection or controls a game element with their mouse 4. Game correctly responds to the player's mouse-based interactions
Alternatives:	1. Game requires the player to use a keyboard-based interaction
Exceptions:	The game does not react properly to the mouse/keyboard interactions or cannot properly read the players input
Priority:	High

4.

Use Case Name:	Feedback Systems
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To properly provide visual/auditory feedback given user actions
Preconditions:	The game is properly launched, GUI is properly displayed, and the mouse-based interactions work properly
Trigger:	The player clicks a button/enters a command
Scenario:	1. Player launches game 2. Player attempts an action, such as a button click, or command enter 3. The UI provides visual/auditory feedback in response to the action, to confirm the desired action has been taken
Alternatives:	None
Exceptions:	The UI does not properly provide the feedback or if the players' desired action is not read properly
Priority:	High

5.

Use Case Name:	Main Menu
Primary Actors:	Player

Secondary Actors:	None
Goal in context:	To display the title of our game, provide five options for the user to select from, a visual representative of our game, and information about the game’s development
Preconditions:	The game is properly launched, and the GUI properly displays the Main Menu screen
Trigger:	Player loads game and selects an option from the Main Menu screen
Scenario:	1. Player launches game 2. Player sees Main Menu along with all the information 3. Player selects one of the five options from the Main Menu (start new game, load previously saved game, instructions, high scores, exit) 4. Game properly loads the new screen from the option the player selects
Alternatives:	1. Player decides to return to Main Menu while playing the game
Exceptions:	Main Menu screen does not properly load, or one of the five options does not load when selected
Priority:	High

6.

Use Case Name:	Instructions/Tutorials
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	Provide the player with detailed instructions on how to play the game and use the application
Preconditions:	The game is properly launched, and the player selects to read the instructions/view the tutorial
Trigger:	The player selects the instructions or tutorial button in the Main Menu screen
Scenario:	1. Player launches game 2. Players selects instructions/tutorial button in the Main Menu screen 3. A text-based, picture-based, or interactive tutorial with detailed instructions on how to play the game and use the application is displayed on the screen 4. The player reads the instructions or completes the tutorial and learns all major features
Alternatives:	None
Exceptions:	Instruction screen/interactive tutorial does not load properly
Priority:	High

7.

Use Case Name:	Scoring System
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To award points or score based on player performance, that can also be displayed to the player
Preconditions:	The game is properly launched, and the player is playing the game as intended
Trigger:	The player completes an action/level that would earn them points/score

Scenario:	1. Player launches game 2. Player is playing game as intended 3. Player completes action/level to earn points/scores (time taken to complete level) 4. The scoring system properly awards the correct amount of points/score 5. The player's new score is displayed
Alternatives:	None
Exceptions:	The scoring system does not correctly award the player their points/score, or does not display their score
Priority:	High

8.

Use Case Name:	Progression Software
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To progress players through at least three levels, stages, or modules
Preconditions:	The game is properly launched, and the player is playing the game as intended
Trigger:	The player completes a level by achieving predefined educational objectives or reaching a certain score
Scenario:	1. Player launches game 2. Player is playing game as intended 3. Player completes level by achieving predefined educational objectives or reaching a certain score 4. The progression machine allows the player to move on to the next level 5. The new level/stage is properly displayed on the player's screen
Alternatives:	1. Player does not complete and has to retry current level/game
Exceptions:	Progression machine does not properly progress player to next level/stage, or does not properly reset the level/game if they've failed
Priority:	High

9.

Use Case Name:	Save/Load Game State
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To save a player's progress locally and reload that progress at a later point
Preconditions:	The game is properly launched, and the player is playing the game as intended
Trigger:	Player reaches set checkpoint or manually saves game from the Main Menu

Scenario:	1. Player launches game 2. Player is playing game as intended 3. Player reaches set checkpoint or manually saves game from the Main Menu 4. Save game state properly captures the player’s last significant progression point 5. Game displays a clear confirmation message when the game is saved 6. Player starts game again and loads their previously saved state from Main Menu 7. Loading process brings player back to logical point in game without backtracking content
Alternatives:	1. Player does not reach nearest checkpoint or manually save game, and their progress is not saved 2. Player does not select to load previous game state and has to restart
Exceptions:	Save game state does not properly save at a checkpoint or when player manually saves, or loading process does not bring the player back to where they were when they saved
Priority:	High

10.

Use Case Name:	High Score Table
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	Maintain a list of the top players scores achieved for each level
Preconditions:	The game is properly launched, and the player is playing the game as intended or the player loads in the Main Menu
Trigger:	Player selects to view high score table from Main Menu or player completes gameplay level
Scenario:	1. Player launches game 2. Player views high score table in Main Menu screen or completes gameplay level 3. The high score table is properly displayed with the five best scores, with each player’s name/initials alongside their score 4. If the player’s score beats one of the top five scores, it is added to the correct spot of the high score table, with the lowest score being removed 5. Player views updated high table score
Alternatives:	1. Player completes gameplay level, but their score is not in the top five, so the high score table is displayed but their score is not added
Exceptions:	High score table is not properly displayed when player selects it from Main Menu or completes gameplay level, or a player’s score is not added to the table if it surpasses one of the current five high scores
Priority:	High

11.

Use Case Name:	Multiple Players
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	For players to create a new identity when starting a new game
Preconditions:	The game is properly launched, and the game has a start new game option in the Main Menu
Trigger:	Player selects to start a new game and is prompted to enter their identity

Scenario:	1. Player launches game 2. Player selects to start new game from Main Menu 3. Player enters their name, initials, username, or email 4. The game locally saves the player’s identifier 5. The game uses the identifier in the high score table, includes it in the save state, and uses it for tracking metrics
Alternatives:	1. Player launches game but chooses not to start a new game and instead load a previously saved game
Exceptions:	Player enters a character/name that is not accepted, or the game does not save the player’s identifier
Priority:	High

12.

Use Case Name:	Instructor Mode
Primary Actors:	Instructor
Secondary Actors:	Player
Goal in context:	Shows a view of the level progression and level scores for every player registered to the system
Preconditions:	The game is launched properly, and the instructor selects to enter instructor mode
Trigger:	The instructor selects instructor mode and enters the correct password
Scenario:	1. Instructor launches game 2. Instructor selects instructor mode from Main Menu 3. Instructor enters the correct password to enter instructor mode 4. Instructor views level progression and level scores for every player registered to the system
Alternatives:	1. Instructor enters the wrong password to enter instructor mode
Exceptions:	The game can not properly process the correct password, or the game does not properly display the information
Priority:	High

13.

Use Case Name:	Debug Mode
Primary Actors:	Administrator
Secondary Actors:	None
Goal in context:	Skip specific levels or unlock levels during run time for debugging purposes
Preconditions:	The game is launched, and the Administrator selects debug mode from Main Menu
Trigger:	Administrator enters secret password, key combination, cheat code, or secret button
Scenario:	1. Administrator launches game 2. Administrator selects debug mode 3. Administrator enters correct password, key combination, completes cheat code, or clicks secret button 4. Administrator enters debug mode and jumps to specific level without completing previous progression points
Alternatives:	1. Administrator enters wrong input/fails interaction with cheat code/button to enter debug mode
Exceptions:	The game can not properly process the correct input/interaction, or the developer can not properly jump to certain levels of the game
Priority:	High

14.

Use Case Name:	Housekeeping & Error Handling
Primary Actors:	Player, Instructor, Developer
Secondary Actors:	None
Goal in context:	To handle any errors efficiently and make sure the game/application runs smoothly
Preconditions:	The game is launched/is currently running
Trigger:	Any possible action the user can perform
Scenario:	1. User launches game 2. The user interacts with any possible action in the game 3. Any errors raised should be handled with clear messages shown to the user 4. If the user decides to exit the application, it is done cleanly with all data being saved correctly such that it is available the next the application is started 5. If the user decides to minimize/maximize the window, UI elements scale correctly to the window size 6. Any user input that could potentially be incorrect/cause faults is validated & checked for errors
Alternatives:	1. If the game/application crashes at any given point, a clear error message is shown to the user
Exceptions:	If the user crashes/runs into any error in the game/application and it is not handled correctly
Priority:	High

15.

Use Case Name:	Display Mathematical Equations and Validate Input For Structure & Correctness
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To display mathematical equations to the users during certain parts of the game, and to validate their answers
Preconditions:	The game is properly launched, and the player is playing the game as intended
Trigger:	The player reaches a point in the level/game where they have to answer a mathematical equation
Scenario:	1. Player launches game 2. Player reaches point in game where mathematical equation is displayed on their screen 3. Player enters their answer to the equation in the solution area 4. Game validates the player's answer in the system 5. Game checks if the answer provided by the user was correct, incorrect, or invalid. 6. Game displays a message stating if answer was correct, incorrect, or invalid 7. If answer was correct, player progression is increased.
Alternatives:	None
Exceptions:	Mathematical equation is not properly displayed on the player's screen, or the game does not correctly validate the answer
Priority:	High

16.

Use Case Name:	Return To Main Menu / Level Navigation
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To return back to the Main Menu upon completing a level
Preconditions:	The game is properly launched, and the player is playing the game as intended
Trigger:	The player completes any level in the game
Scenario:	1. Player launches game 2. Player starts new game/loads saved game 3. Player completes a level in the game 4. If player has completed every level, game notifies the player, otherwise, the game gives the player an option to return back to the Main Menu or Proceed to next level 5. Player selects the option of where they would like to proceed. 6. Player is sent to their specified destination
Alternatives:	1. Player does not choose to return to Main Menu upon level completion and navigates to next level/high score table
Exceptions:	Game does not display option to return to Main Menu, or player does not navigate to Main Menu upon level completion if they select to return
Priority:	Medium

17.

Use Case Name:	New Score Notification
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To clearly state when a new personal or overall high score is met on a level
Preconditions:	The game is properly launched, the player is playing the game as intended, and the high score table works properly
Trigger:	Player achieves new personal/overall high score on a level
Scenario:	1. Player launches game 2. Player completes a level in the game 3. Game computes the time it took for the player to complete the level 4. If a new personal/overall high score is met on the level, the player is clearly notified
Alternatives:	1. Player completes a level but does not achieve a new personal/overall high score
Exceptions:	Game does not correctly compute new personal/high score, or does not notify player of the new score
Priority:	Medium

18.

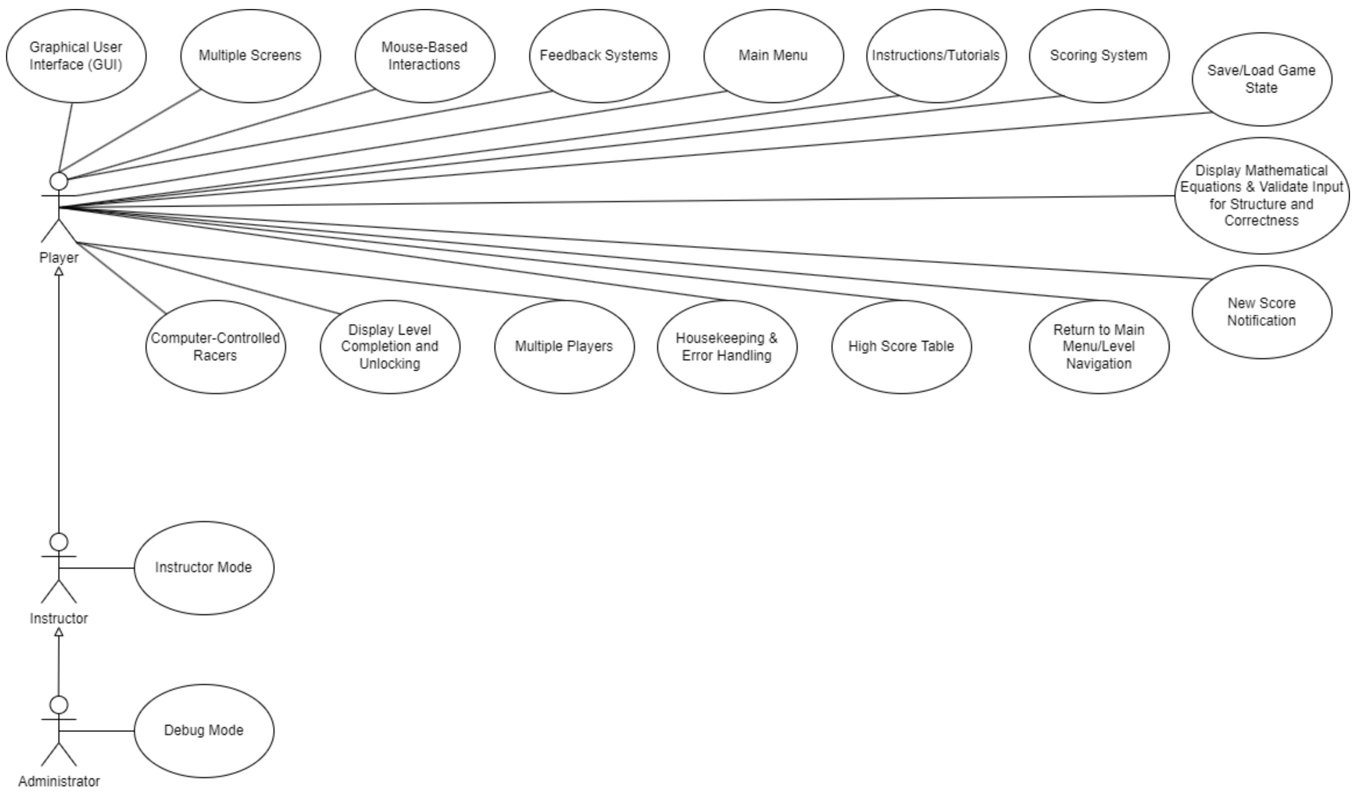
Use Case Name:	Computer-Controlled Racers
Primary Actors:	Player

Secondary Actors:	Computer-Controlled Racers
Goal in context:	To have the player compete with computer-controlled racers that they must beat to complete the level
Preconditions:	The game is properly launched, the player is playing the game as intended, and the answer validations work correctly
Trigger:	Player starts a new level of the game
Scenario:	1. Player launches game 2. Player starts new level 3. The computer-controlled racers also start when the player starts 4. The player correctly answers the mathematical equations to move forward, while the computer-controlled racers move forward at a timely pace given the level 5. If the player reaches the finish line before any of the computer-controlled racers, the level is complete
Alternatives:	1. The player loses to one of the computer-controlled racers
Exceptions:	Computer-Controlled Racers do not move at the correct pace of the level (or move at all), or the level does not stop when one of them reaches the finish line
Priority:	High

19.

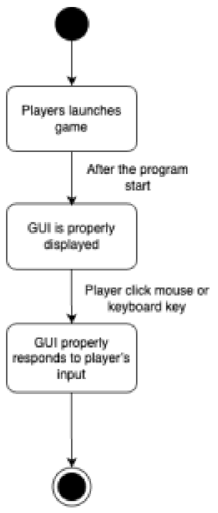
Use Case Name:	Display Level Completion and Unlocking
Primary Actors:	Player
Secondary Actors:	None
Goal in context:	To clearly display when a level has been completed and when a new level has been unlocked
Preconditions:	The game is properly launched, the player is playing the game as intended, and the progression machine works correctly
Trigger:	The player completes a level in the game
Scenario:	1. Player launches game 2. Player starts new level 3. Player completes new level 4. Game clearly displays the current level has been successfully completed 5. If there is another level to play, the game clearly displays a new level has been unlocked 6. If there are no more levels to play, the game clearly displays the player has fully completed the game
Alternatives:	1. Player starts new level, but they do not successfully complete it and unlock the next level, and no messages are displayed
Exceptions:	Game does not properly display message upon level completion/level unlocking/game completion
Priority:	Medium

Use Case Diagram

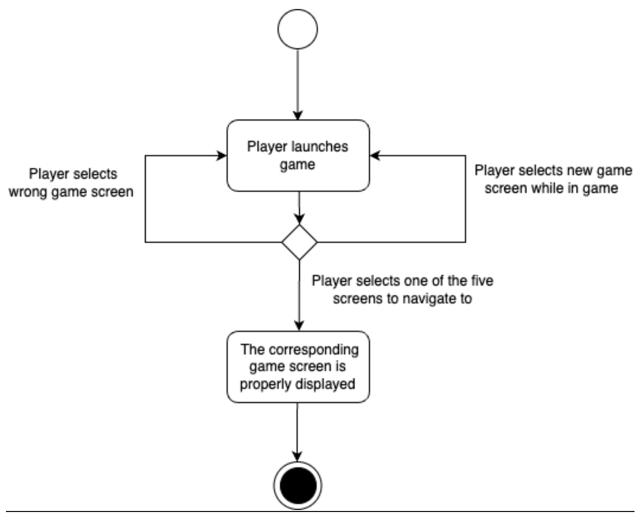

Graphical User Interface (GUI)
Multiple Screens
Mouse-Based Interactions
Feedback Systems
Main Menu
Instructions/Tutorials
Scoring System
Save/Load Game State
Display Mathematical Equations & Validate Input for Structure and Correctness
New Score Notification
Player
Computer-Controlled Racers
Display Level Completion and Unlocking
Multiple Players
Housekeeping & Error Handling
High Score Table
Return to Main Menu/Level Navigation
Instructor
Instructor Mode
Administrator
Debug Mode

Activity Diagrams

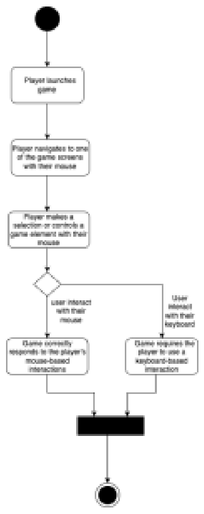
Graphical User Interface (Use Case #1)


Players launches game
After the program start
GUI is properly displayed
Player click mouse or keyboard key
GUI properly responds to player's input

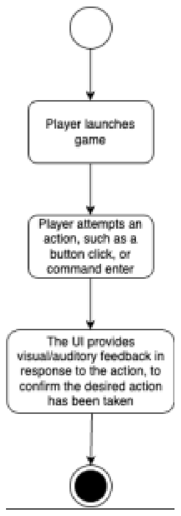
Multiple Screens(Use Case #2)


Player launches game
Player selects wrong game screen
Player selects new game screen while in game
Player selects one of the five screens to navigate to
The corresponding game screen is properly displayed

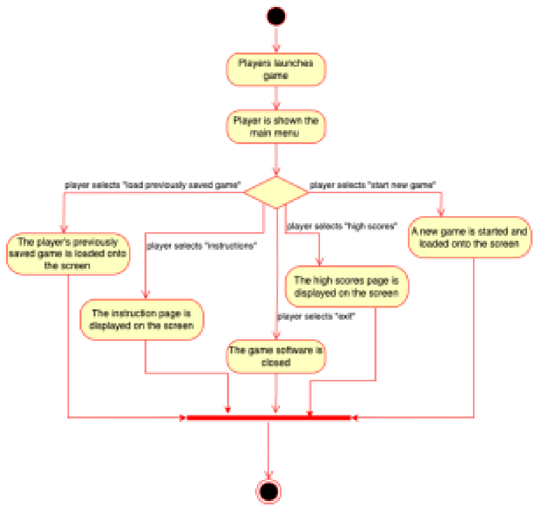
Mouse-Based Interactions(Use Case #3)


Player launches game
Player navigates to one of the game screens with their mouse
Player makes a selection or controls a game element with their mouse
user interact with their mouse
User interact with their keyboard
Game correctly responds to the player's mouse-based interactions
Game requires the player to use a keyboard-based interaction

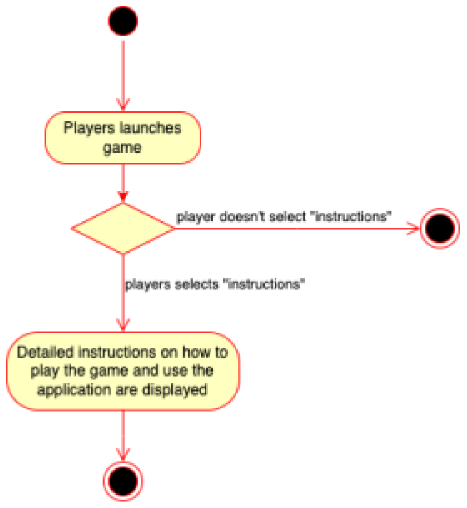
Feedback Systems(Use Case #4)


Player launches game
Player attempts an action, such as a button click, or command enter
The UI provides visual/auditory feedback in response to the action, to confirm the desired action has been taken

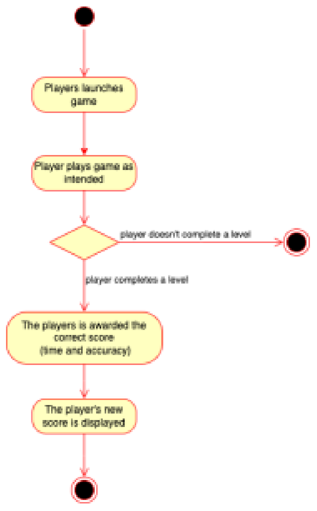
Main Menu (Use Case #5)


Players launches game
Player is shown the main menu
player selects "load previously saved game"
player selects "start new game"
player selects "high scores"
player selects "instructions"
The player's previously saved game is loaded onto the screen
A new game is started and loaded onto the screen
The high scores page is displayed on the screen
The instruction page is displayed on the screen
player selects "exit"
The game software is closed

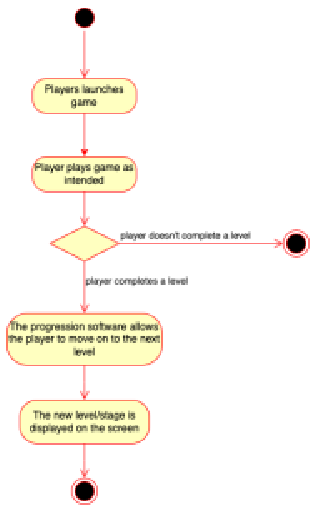
Instructions/Tutorial(Use Case #6)


Players launches game
player doesn't select "instructions"
players selects "instructions"
Detailed instructions on how to play the game and use the application are displayed

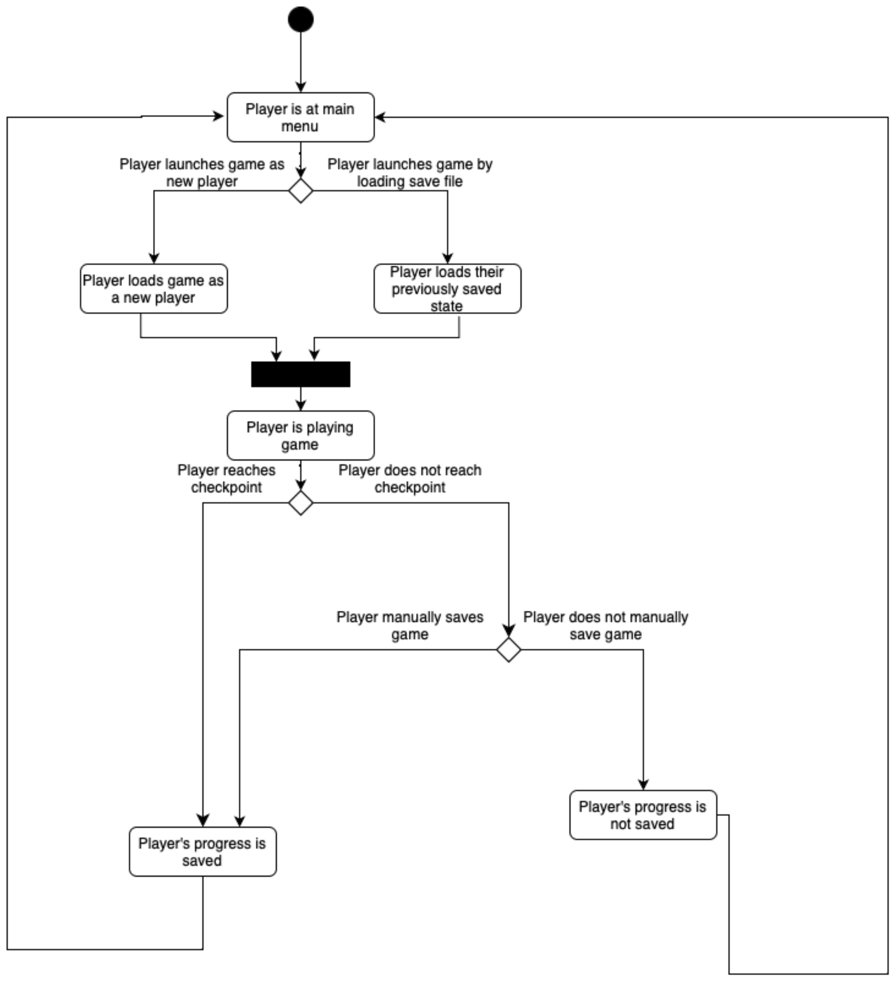
Scoring System Mode (Use Case #7)


Players launches game
Player plays game as intended
player doesn't complete a level
player completes a level
The players is awarded the correct score (time and accuracy)
The player's new score is displayed

Progression Software (Use Case #8)

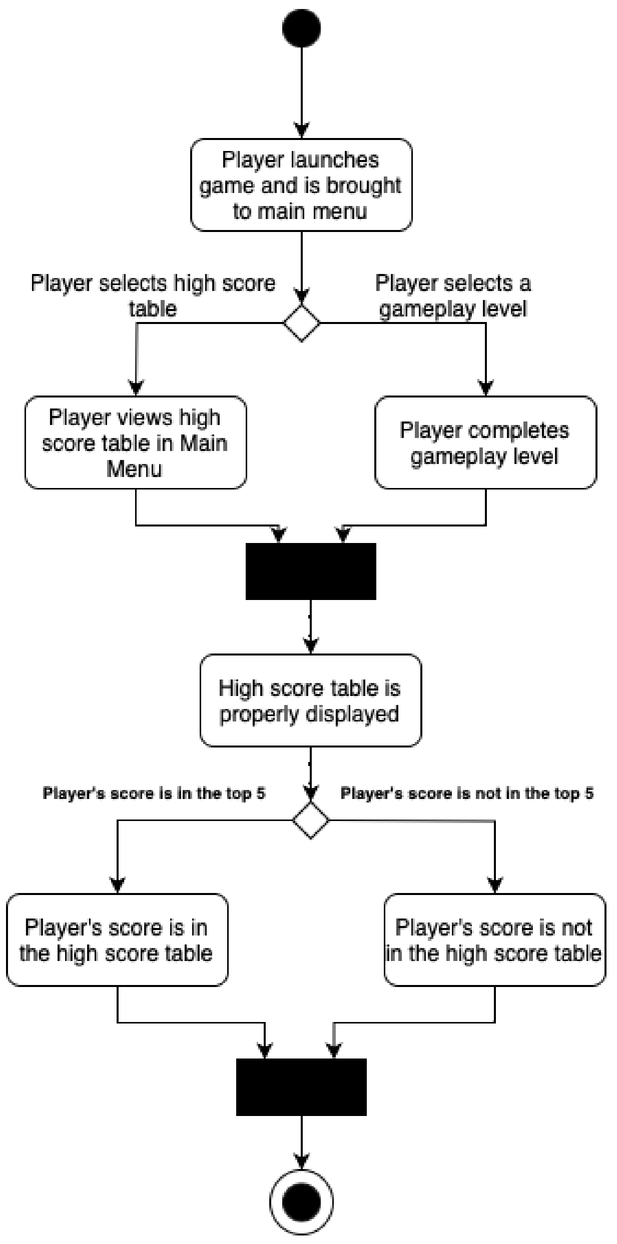

Players launches game
Player plays game as intended
player doesn't complete a level
player completes a level
The progression software allows the player to move on to the next level
The new level/stage is displayed on the screen

Save/Load Game State (Use Case #9)

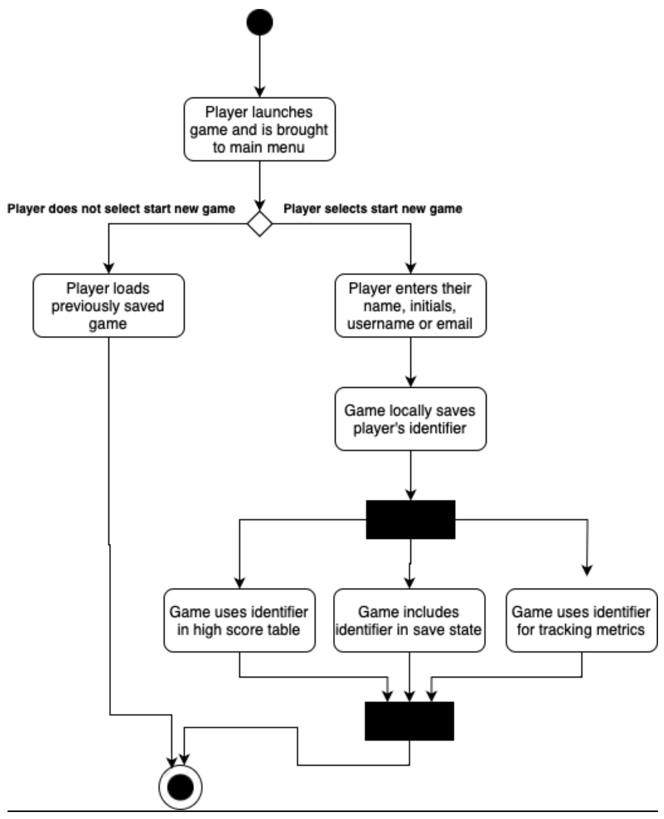

Player is at main menu
Player launches game as new player
Player launches game by loading save file
Player loads game as a new player
Player loads their previously saved state
Player is playing game
Player reaches checkpoint
Player does not reach checkpoint
Player manually saves game
Player does not manually save game
Player's progress is not saved
Player's progress is saved

\

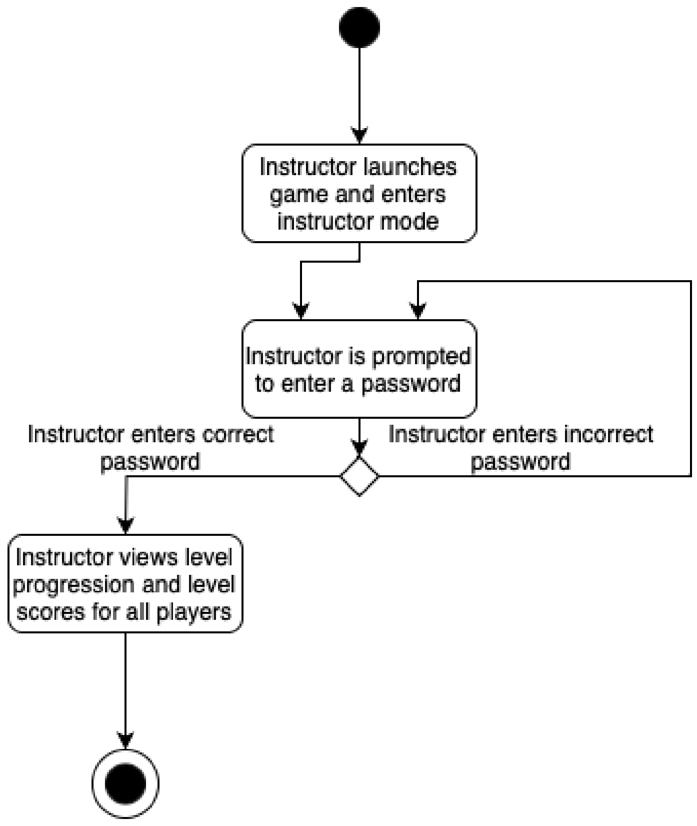
High Score Table (Use Case #10)


Player launches game and is brought to main menu
Player selects high score table
Player selects a gameplay level
Player views high score table in Main Menu
Player completes gameplay level
High score table is properly displayed
Player's score is in the top 5
Player's score is not in the top 5
Player's score is in the high score table
Player's score is not in the high score table

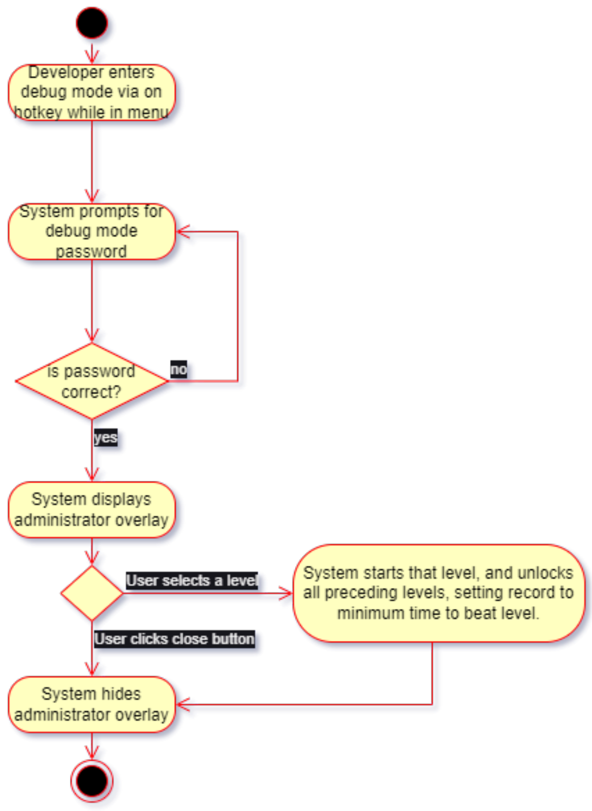
Multiple Players (Use Case #11)


Player launches game and is brought to main menu
Player does not select start new game
Player selects start new game
Player loads previously saved game
Player enters their name, initials, username or email
Game locally saves player's identifier
Game uses identifier in high score table
Game includes identifier in save state
Game uses identifier for tracking metrics

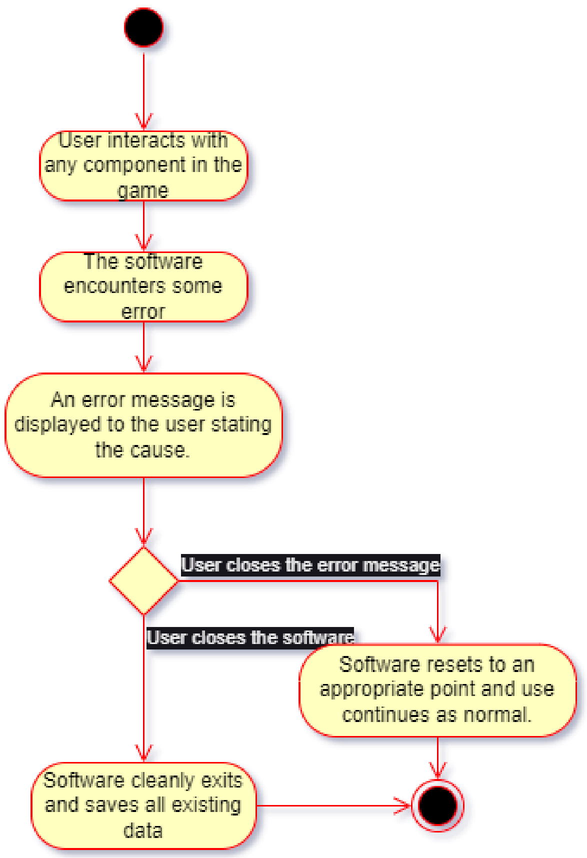
Instructor Mode (Use Case #12)


Instructor launches game and enters instructor mode
Instructor is prompted to enter a password
Instructor enters correct password
Instructor enters incorrect password
Instructor views level progression and level scores for all players

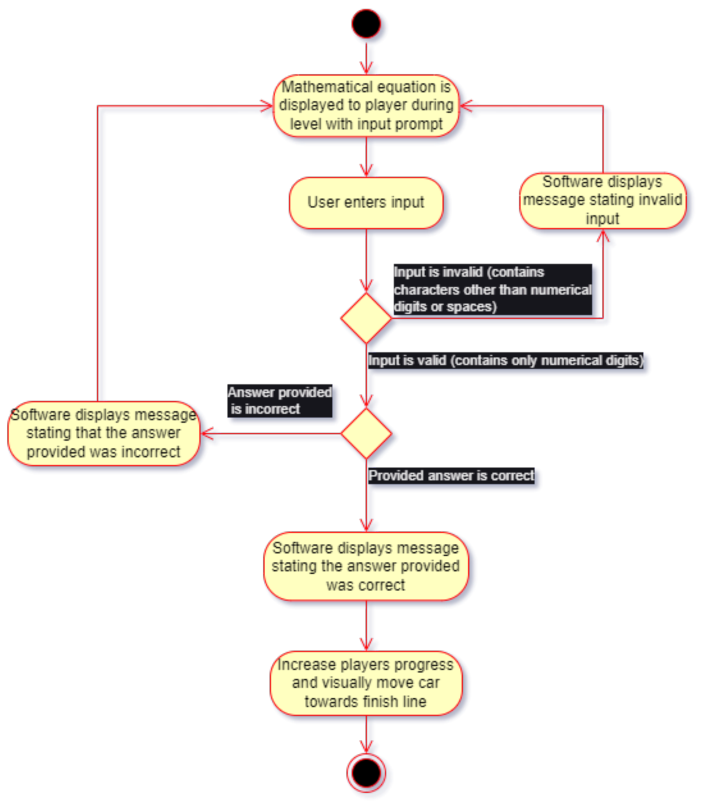
Debug Mode (Use Case #13)


Developer enters debug mode via on hotkey while in menu
System prompts for debug mode password
is password correct?
no
yes
System displays administrator overlay
User selects a level
System starts that level, and unlocks all preceding levels, setting record to minimum time to beat level.
User clicks close button
System hides administrator overlay

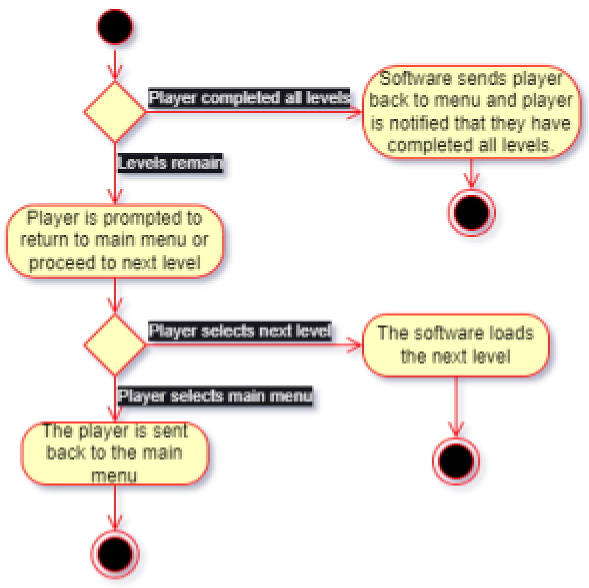
Housekeeping & Error Handling (Use Case #14)


User interacts with any component in the game
The software encounters some error
An error message is displayed to the user stating the cause.
User closes the error message
User closes the software
Software resets to an appropriate point and use continues as normal.
Software cleanly exits and saves all existing data

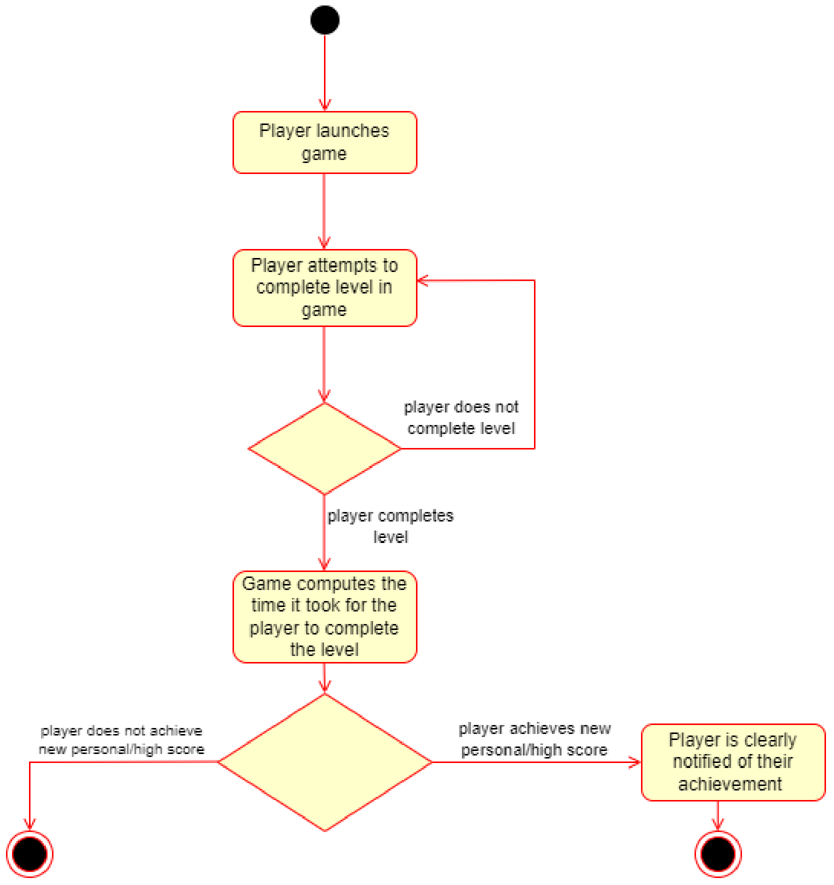
Display Mathematical Equations and Validate Input For Structure & Correctness (Use Case #15)


Mathematical equation is displayed to player during level with input prompt
User enters input
Software displays message stating invalid input
Input is invalid (contains characters other than numerical digits or spaces)
Input is valid (contains only numerical digits)
Answer provided is incorrect
Software displays message stating that the answer provided was incorrect
Provided answer is correct
Software displays message stating the answer provided was correct
Increase players progress and visually move car towards finish line

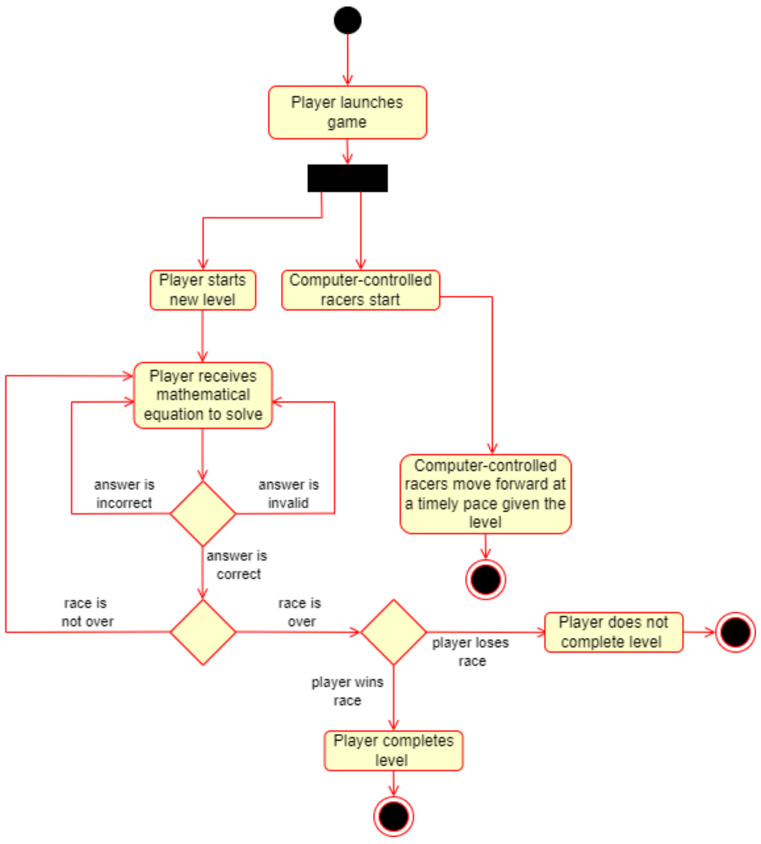
Return To Main Menu / Level Navigation (Use Case #16)


Player completed all levels
Software sends player back to menu and player is notified that they have completed all levels.
Levels remain
Player is prompted to return to main menu or proceed to next level
Player selects next level
The software loads the next level
Player selects main menu
The player is sent back to the main menu

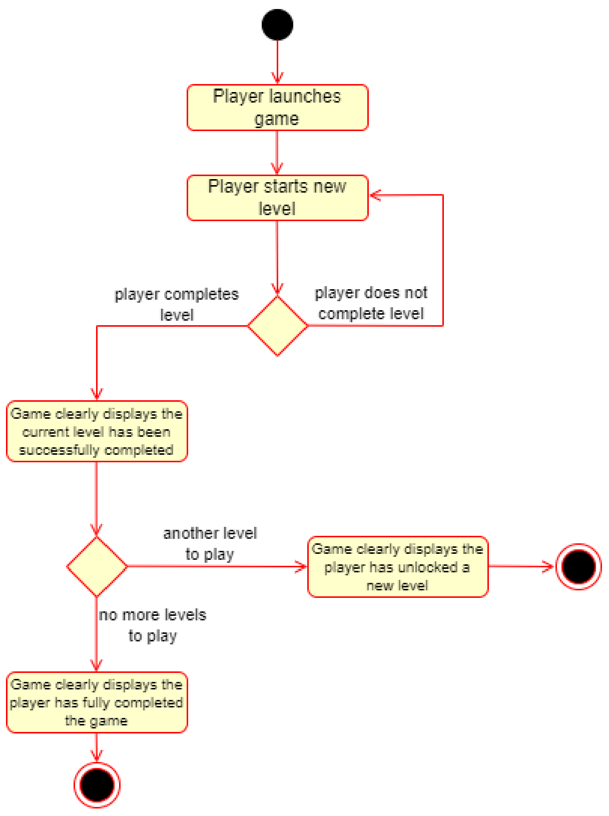
New Score Notification (Use Case #17)


Player launches game
Player attempts to complete level in game
player does not complete level
player completes level
Game computes the time it took for the player to complete the level
player does not achieve new personal/high score
player achieves new personal/high score
Player is clearly notified of their achievement

Computer-Controlled Racers (Use Case #18)


Player launches game
Player starts new level
Computer-controlled racers start
Player receives mathematical equation to solve
answer is incorrect
answer is invalid
answer is correct
Computer-controlled racers move forward at a timely pace given the level
race is not over
race is over
player loses race
Player does not complete level
player wins race
Player completes level

Display Level Completion and Unlocking (Use Case #19)


Player launches game
Player starts new level
player completes level
player does not complete level
Game clearly displays the current level has been successfully completed
another level to play
Game clearly displays the player has unlocked a new level
no more levels to play
Game clearly displays the player has fully completed the game

Non-Functional Requirements

1. The application must be developed using the Java programming language
2. Educational content of the game must be accurate, and appropriate for grade ranges two to four
3. User interface must be user friendly, intuitive, and align with UX best practices.
4. The save data must be stored locally on the machine and not require an internet connection.
5. All third party libraries used must be freely available.
6. All code must be commented using Javadoc, each file must have a comment at the top explaining author and purpose.
7. Code must be unit tested with JUnit 5 tests, excluding code that can only be tested via a GUIs actions.
8. The application must be executable on a windows 10 system with a standard Java installation.
9. The application must be self-contained, and not modify or create and files outside of its directory.
10. File size must be under 1 gigabyte
11. The application should run smoothly and not use excessive computer resources.
12. The application must take accessibility into consideration in the UI design and gameplay.
13. All code should be maintainable and reusable.
14. All the game content, as well as work products must be written in English.
15. The equations presented to the player must progress in difficulty to provide a challenge to the player.
16. The movement of sprites must be smoothly animated.
17. The software must have variety between the levels. (varying equation types)
18. The theme of racing must be evident throughout all pages of the game.
19. The game must have sound effects during gameplay. (engine sounds when car moves, notification sounds when getting a question correct or incorrect)

Requirements Documentation Summary

In conclusion, our goal is to create a “gamified learning” environment for young learners to expand their arithmetic knowledge is a fun and engaging way, that will leave them enjoying math opposed to dreading it. In order to create the most engaging experience we can we will have to provide immediate feedback to users and focus on making them feel like they’re really making progress to keep them on track.

In order to develop this software we will have to have a firm understanding of our software domain and keep in mind the three common issues encountered within this domain as listed earlier in the document. We will also have to adhere to the requirements and diagrams created during this phase of development as much as possible to ensure a smooth development process and a timely delivery of the final product.

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.

Software Domain ~ A software domain reflects a field of study that defines a set of common requirements, terminology, and functionality for any software program constructed to solve a problem in that field.

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.)