Software Requirements Specification (SRS)
Project X

Team:	Group 2
Authors:	Gavin Ippolito, Salwan Sabil, James Walsh, Isaiah Andrade
Customer:	4-5th graders
Instructor:	Professor James Daly

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification document is to minimize ambiguity and miscommunication between all involved in the project. The development team can use this as a blueprint for implementing the game's features and functionality. It is also used by the Stakeholders and customers to ensure their needs and expectations are met. This document acts as (and is part of) a contract between the development team and stakeholders and customers.

1.2 Scope

This prototype will include a very basic version of the FlashCars game; a competitive educational game designed for multiple players. It will include a start menu, a game screen, and a win/loss screen. The application domain is educational and entertaining games. It targets students from 4th to 5th grade in an attempt to provide a more engaging learning experience that will allow students to better retain the learning material.

1.3 Definitions, acronyms, and abbreviations

SRS - Software Requirements Specification

Core - A single processing unit in a in a computer processor that can execute instructions.

GPU - Graphics Processing Unit

RAM - Random Access Memory

SSD - Solid State Drive

LTS - Long Term Support

Unity - A real-time 2D and 3D development engine.

1.4 Organization

The structure of the rest of the document will be as follows:

Section 2: An overall descriptions of the project and it's intended behavior including it's different interfaces, constraints, diagrams and explanations of its intended functionality, user interaction, and its assumptions and dependencies.

Section 3: A list of the functional requirements of the game.

Section 4: Diagrams to present the game's behavior and structure in for form of Use Case, Class, Sequence, and State diagrams.

Section 5: A description of what the prototype does along with instructions on how to run the prototype and example scenarios.

Section 6: A list of references and sources used for the project. There is also a link to the game website.

Section 7: Point of Contact.

2 Overall Description

This section will provide an overview of the product. This includes the context of the product and who the target audience will be, constraints of the system, and highlights the major functionalities of the product. Along with the constraints of the system, this section will provide constraints of the user by listing the assumptions of the user's knowledge. Lastly, this section will state the requirements that were determined beyond the scope of this version/release and will briefly discuss features that could be seen in later versions/releases.

2.1 Product Perspective

This product will provide an educational game for 4th to 5th grade students. It gives a student an alternate way to learn material outside of the classroom. The game's interactive nature will engage a student differently than traditional learning.

The constraints of the game are as follows:

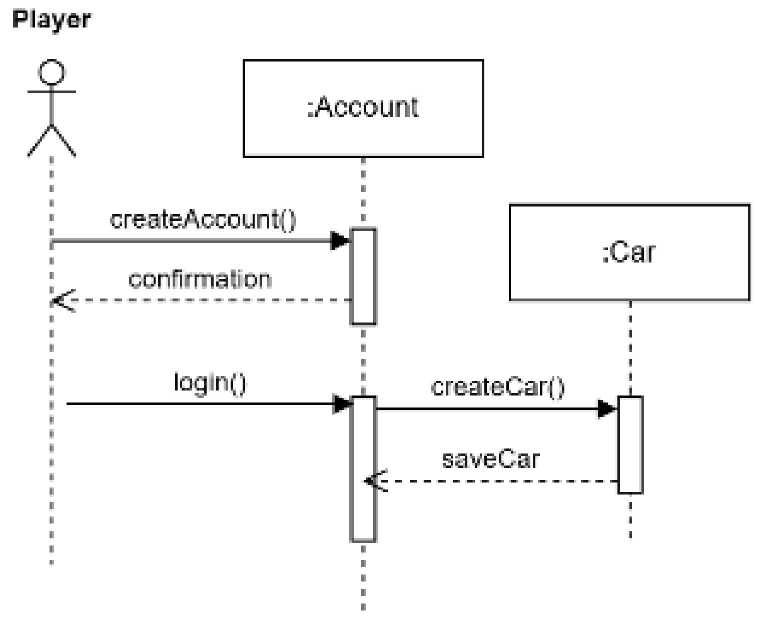
> System and User Interface: The target system is a Windows, Mac, or Linux laptop or desktop computer. The user is expected to use a keyboard and mouse to play.
>
> Hardware Interfaces: Because the game engine is Unity, the hardware constraints are as follows, minimum of 4 cores, 16 GB of RAM, 100 GB SSD, 1 GB GPU memory, DX10 or newer GPU.
>
> Software Interface: In the early stages of development, the player must have Unity 2022 LTS downloaded to be able to compile and play the game.
>
> Communication Interfaces: To download the proper packages, the system must have an internet connection.
>
> Memory Constraints: The target device should have X RAM to be able to compile and play the game smoothly.
>
> Operational Constraints: The system must follow the system/user interface and hardware constraints and allocate the correct resources to allow it to run as intended.
>
> Site Adaptation Operations: Adapting to user-specific sites is out of the scope of this product.

2.2 Product Functions

The first major function of the game is creating an account. Before the user can play the game they will have to make an account. To create an account, the user will have to provide a username, password, and email. Once their account is created, they can log in.

Once the user logs in, they will be prompted to select a car. This will be their avatar when playing. After selecting a car, they can begin the game.

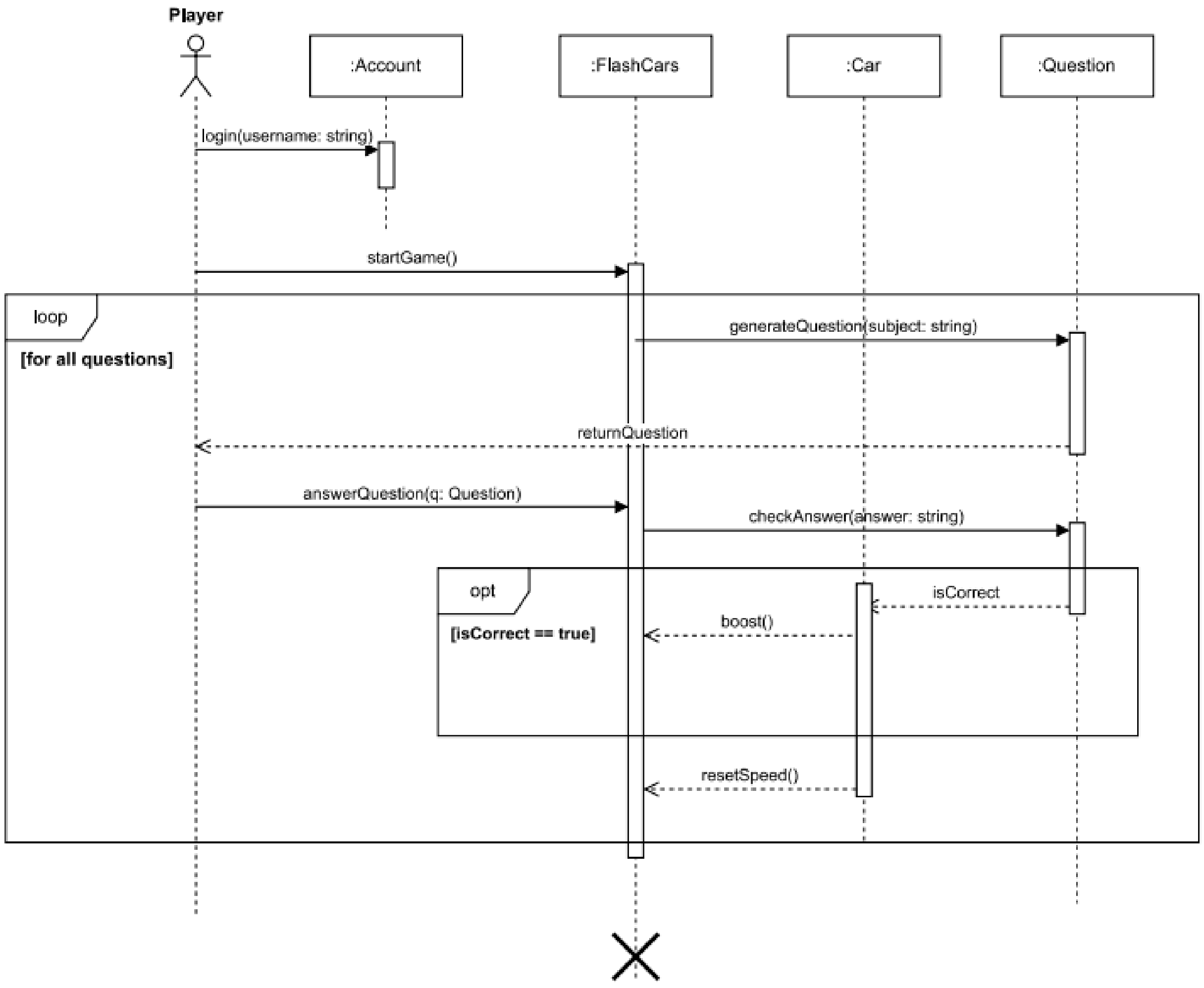
These two actions are represented in the sequence diagram below.



The next major function is playing the game. To play the game, the user will select a difficulty and subject. Buttons will be displayed to make the selection. They can select easy or hard for the difficulty and choose from Math, Science, English, or History for their subject of choice. After doing so, they will be prompted to start the game, this will be done by clicking a button.

Once the game is started, there will be a question displayed at the top of the screen and possible answers below with the player's car on the track below that. As the user answers questions, the car will advance if correct or stay in place if wrong. The game will exhaust all questions and end the game.

This functionality is displayed in the sequence diagram below.



2.3 User Characteristics

The user is expected to have basic knowledge of a computer and know how to use the keyboard and mouse to interact with it. The target audience is 4th to 5th graders. Therefore, they are expected to have learned or are learning the topics listed in the Massachusetts Department of Education curriculum.

2.4 Constraints

Safety-critical properties are characteristics of a system that are essential to avoiding failure that will lead to the system not performing properly. Safety-critical properties must be met. In the context of this system, safety-critical properties are user email and passwords not being secure. This could lead to people other than the user having access to their account. Another property that, if violated, the system will not perform properly is the question information not being factual. Because the product is used for educational purposes, the system will not perform properly if the questions asked do not have factual answers.

2.5 Assumptions and Dependencies

It is assumed that the user will play the game from a Windows, Mac, or Linux system, and the system supports the newest version of Unity. It is also assumed that the system will have a keyboard and mouse that provide input. Lastly, the system must have a display screen that provides output to the user.

2.6 Apportioning of Requirements

This release of the game will not be published and will not be available to outside users through the Internet. Players being able to play against each other from multiple different devices is also beyond the scope of the current release.

Users being able to connect a gaming controller is beyond the scope of the current release.

3 Specific Requirements

Requirements:

1. The game will feature a car racing theme where players answer trivia questions to progress along a race track.
2. There will be various subjects for players to choose from, including Math, Science, and History at a 4-5th grade level.
3. Each subject will contain multiple levels from which the player can select.

    3.1 Additionally, the game will include a customizable level.

    3.2 The player will input questions and answers. This allows the player to have the option to study specific material.

4. The game will display:

    4.1 A race track background with a car avatar representing the player.

    4.2 A question will be displayed.

    4.3 Multiple-choice answer options below the question.

5. Players will answer questions to move their car forward on the track:

    5.1 A correct answer will move the car a set distance.

    5.2 For an incorrect answer, the car will remain stationary.

    5.3 After two incorrect answers, a hint will be displayed to help the player learn the material.

7. The game will track players' scoring and will produce feedback accordingly: Points will be awarded for each correct answer.

7.1 The car will progress double the set amount if the player has answered correctly three consecutive times.

7.3 Feedback messages will appear after each answer, providing explanations for correct and incorrect answers.

7.4 Sound effects will play for correct/incorrect answers.

7.5 An appropriate car animation will occur depending if the player's answer is correct or incorrect.

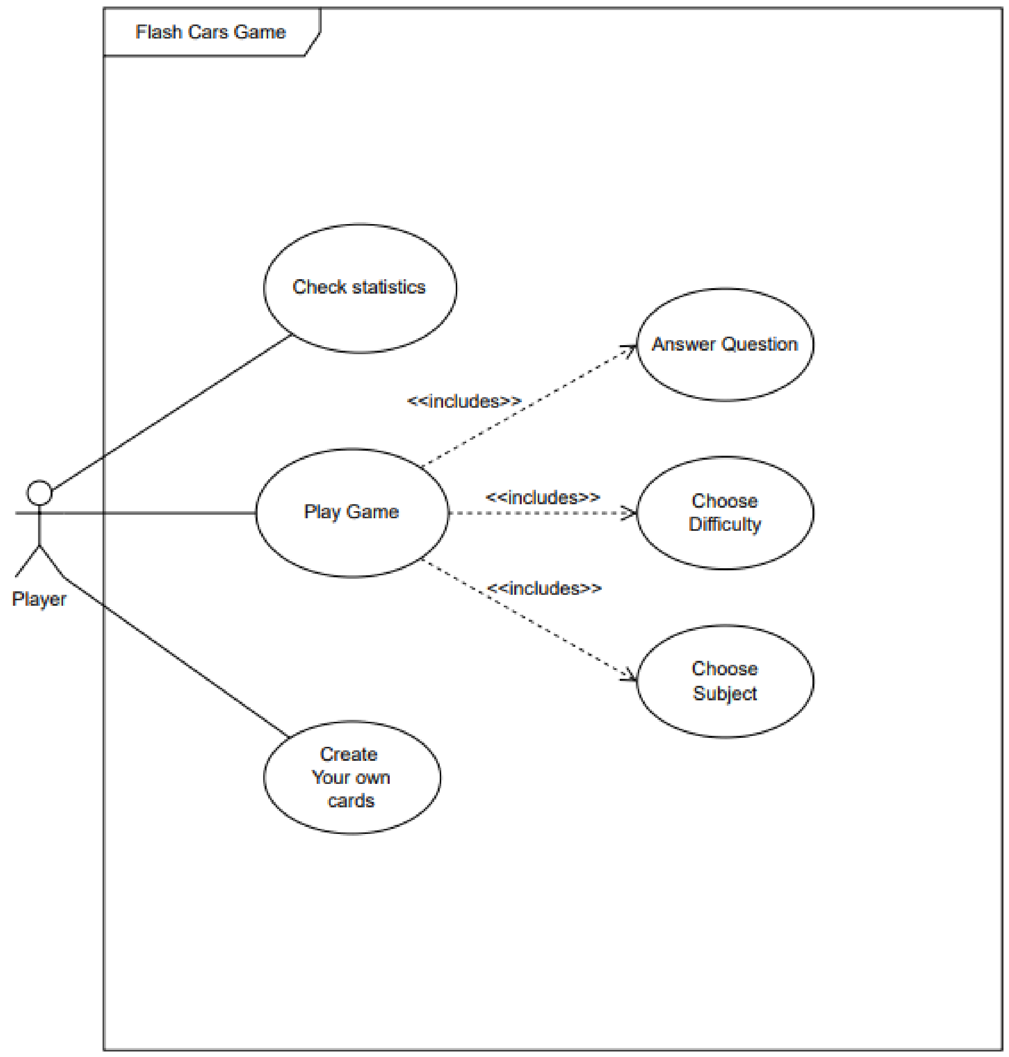
8. The game will end after a set amount of questions are answered correctly.

9. The game will store player data; including subject performance, level completion, and high scores.

4 Modeling Requirements

4.1 Use case Diagram

The use case diagram below depicts the general functions a user will be able to access when playing the game. Directly the player will be able to check their statistics to see how they are progressing, access the game, and create their own study sets to work on subjects that may be more personalized for them. The play game use includes sub uses that when prompted the user must complete before the game can be started/ completed, and creating your own cards is an extension of choosing the subject.


Flash Cars Game
Check statistics
Answer Question
<<includes>>
Play Game
<<includes>>
Choose Difficulty
<<includes>>
Choose Subject
Player
Create Your own cards

Use Case Name:	Check Statistics
Actors:	Player
Description:	The player will choose to look at their statistics to see how they perform
Type:	Primary
Includes:	none
Extends:	none
Cross-refs:	none
Uses cases:	Check performance

Use Case Name:	Play Game
Actors:	Player
Description:	The act of playing the game
Type:	Primary and essential
Includes:	Answer Question, Choose mode, Choose subject
Extends:	none
Cross-refs:	none
Uses cases:	The playing of the game

Use Case Name:	Answer question
Actors:	Player
Description:	Main functionality of the game player will do this to advance
Type:	Secondary and essential
Includes:	none
Extends:	none
Cross-refs:	none
Uses cases:	Progressing through the game

Use Case Name:	Choose Difficulty
Actors:	Player
Description:	Necessary to starting the game the player will choose which mode they want to play
Type:	Secondary and essential
Includes:	none
Extends:	none
Cross-refs:	none
Uses cases:	Selecting the mode of the game.

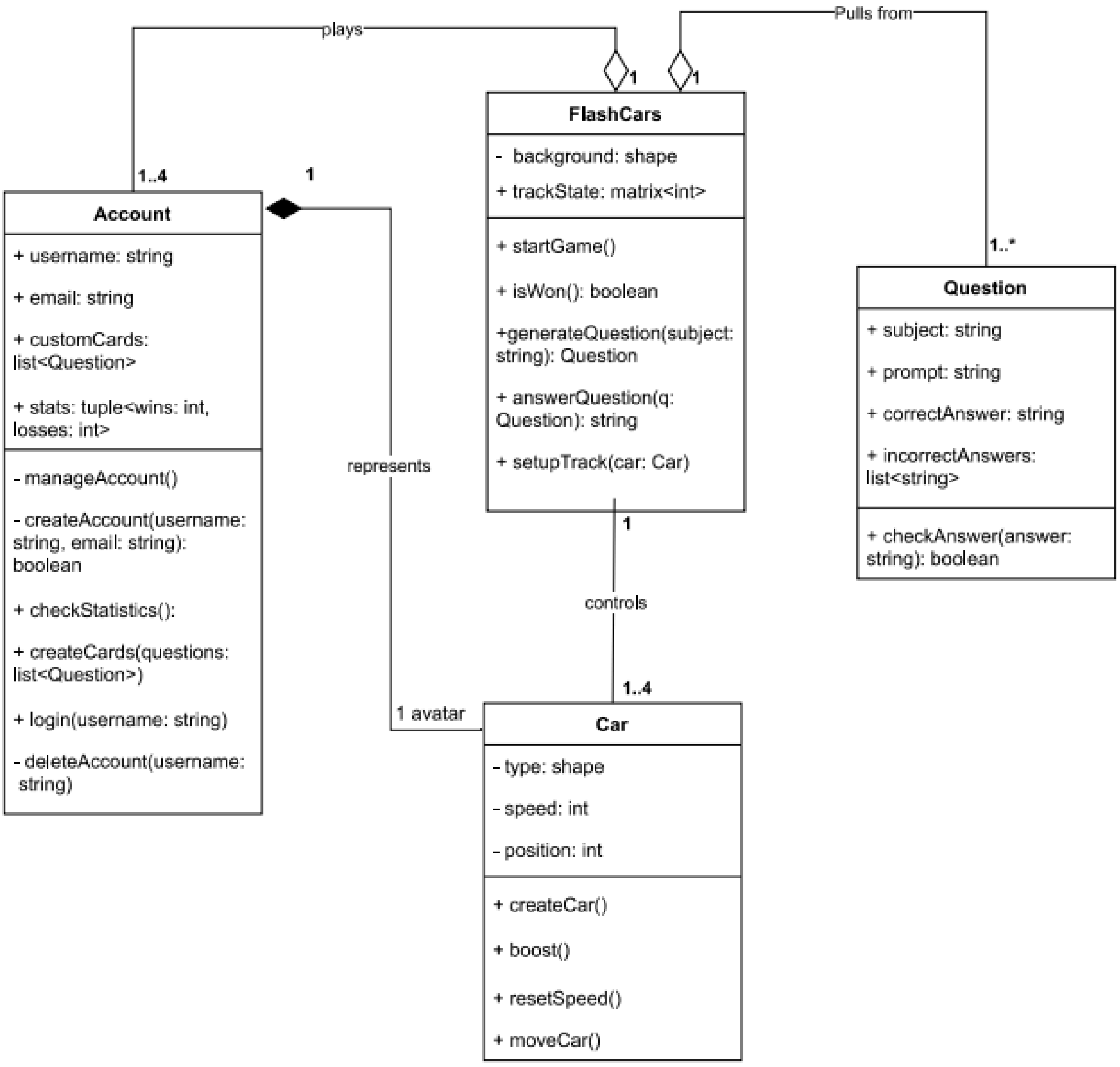
Use Case Name:	Choose Subject
Actors:	Player
Description:	Before starting the game the player must select what subject they would like to learn
Type:	Secondary and essential
Includes:	none
Extends:	none
Cross-refs:	none
Uses cases:	Determining the subject of game.

Use Case Name:	Create Your own Cards
Actors:	Player
Description:	This option will be presented to the user to allow them to study more personalized subjects
Type:	Primary
Includes:	none
Extends:	none
Cross-refs:	none
Uses cases:	Create new subject of cards

4.2 Class Diagram

Class diagrams are used as a way to model the general methods and attributes of the classes within a given project and their relationships. A class will be represented as a box with two main components: the attributes and the methods. The attributes of a class are the variables that define the objects of a class and the methods are the functions associated with the class.

The class diagram below has four major classes that all play specific roles when it comes to the complete product of our game. FlashCars will be the main and most important class as it houses the actual game and its functionality. The Account class holds everything to do with the player's interactions with the game.


plays
Pulls from
1
1
1..4
1
1..*
FlashCars
- background: shape
+ trackState: matrix<int>
+ startGame()
+ isWon(): boolean
+generateQuestion(subject: string): Question
+ answerQuestion(q: Question): string
+ setupTrack(car: Car)
Account
+ username: string
+ email: string
+ customCards: list<Question>
+ stats: tuple<wins: int, losses: int>
- manageAccount()
- createAccount(username: string, email: string): boolean
+ checkStatistics():
+ createCards(questions: list<Question>)
+ login(username: string)
- deleteAccount(username: string)
represents
1 avatar
Question
+ subject: string
+ prompt: string
+ correctAnswer: string
+ incorrectAnswers: list<string>
+ checkAnswer(answer: string): boolean
1
controls
1..4
Car
- type: shape
- speed: int
- position: int
+ createCar()
+ boost()
+ resetSpeed()
+ moveCar()

4.3 Data Dictionary

Element Name		**Description**
Account		This class will be what represents the player and will handle everything based on what the user wishes to do
Attributes		
	+username: string	the username player will use
	+email: string	email entered by user
	-password: string	password entered by user
	+customCards: enum	when a user wants to make their own flash cards
	+stats: tuple<wins:int, losses: int>	way for user to see how much they've won/loss
Methods		
	-manageAccount()	allows user to manage things about their account
	-createAccount(username: string, email: string, password: string): boolean	Used to create the users account
	+checkStatistics()	User can access their statistics
	+createCards(questions: dict)	how the user will create new cards
	+login()	grants access to the game
	-deleteAccount()	user wants to get rid of their account they can delete it

Relationships	FlashCards - give the player a character to play the game that is held in FlashCards Car - Is the representation of the user in the game manipulated by the user.

Element Name		**Description**
FlashCars		This class is a representation of the game
Attributes		
	-background: shape	the game being displayed on the screen
	+trackstate: matrix<int>	The path the car the user is controlling will follow
Methods		
	+StartGame()	starts the game
	+isWon(): boolean	returns is the game is won or not
	+generateQuestion(subject: string): Question	will choose the question to provide to the user
	+answerQuestion(Question): string	check users answer vs. correct answer
	+setupTrack	generates the track for gameplay
Relationships	Account - Is directly accessed from account and played through that Car - receives commands through FlashCars based on gameplay from user Question - Receives the question to be displayed	

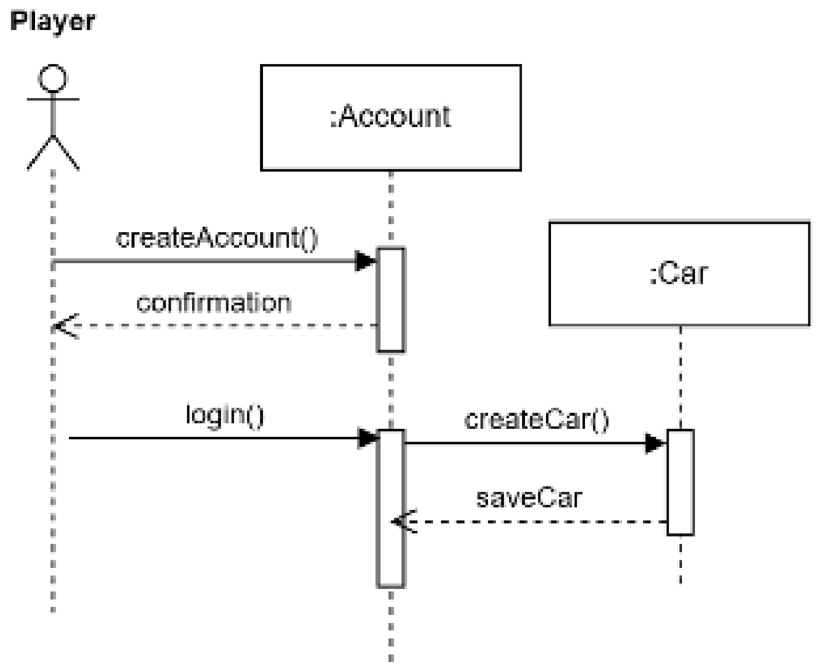
Element Name		**Description**
Car		the representation of the player inside the game
Attributes		
	-type: shape	the car sprite
	-speed: int	how fast the car is going
	-position: int	where the car is
Methods		
	+createCar()	generates a cara object for the car
	+boost()	if a player is on a streak of correct guesses boost will reward them with more speed
	+resetSpeed()	if answered incorrectly the speed will be reset
	+moveCar()	calculates speed of the car and moves it along the track accordingly
Relationships	Account - Is a representation of the user by way of the account FlashCars - receives commands through FlashCars based on gameplay from user	

Element Name		Description
Question		Will handle presenting questions and answers to user
Attributes		
	+Subject: String	topic of questions that will be asked
	+prompt: String	text displayed to user
	+correctAnswer: String	the correct answer
	+incorrectAnswers: list<String>	pool of incorrect answers to fill multiple choice questions
Methods		
	+checkAnswer(answer: String): boolean	check if user's answer matches correct answer
Relationships	FlashCars - Provides questions and answers to FlashCars	

4.4 Sequence Diagrams

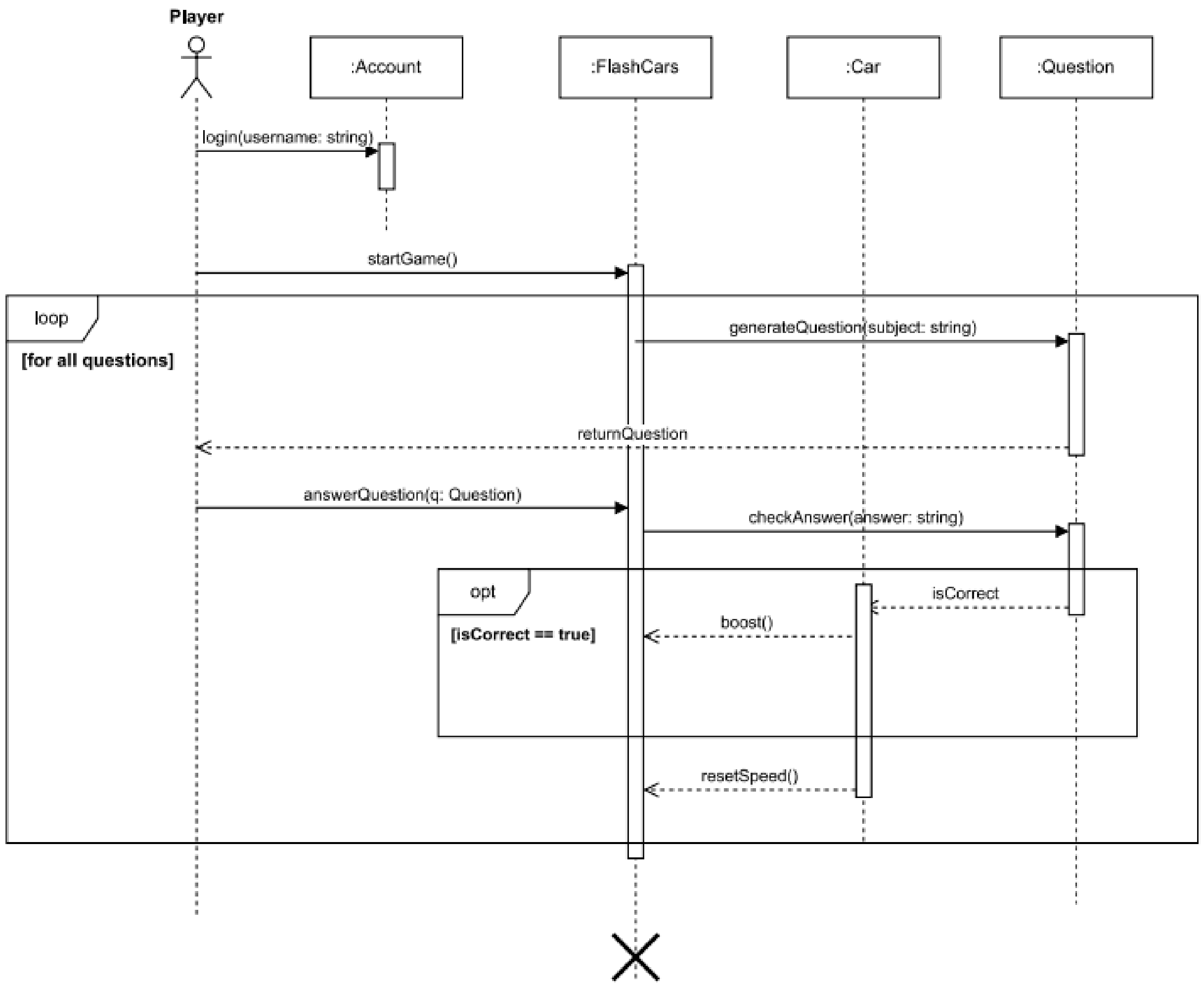
4.4.1 Sequence 1

A simple sequence diagram showing the process a user would go through to create their account and then their car to represent them in game.



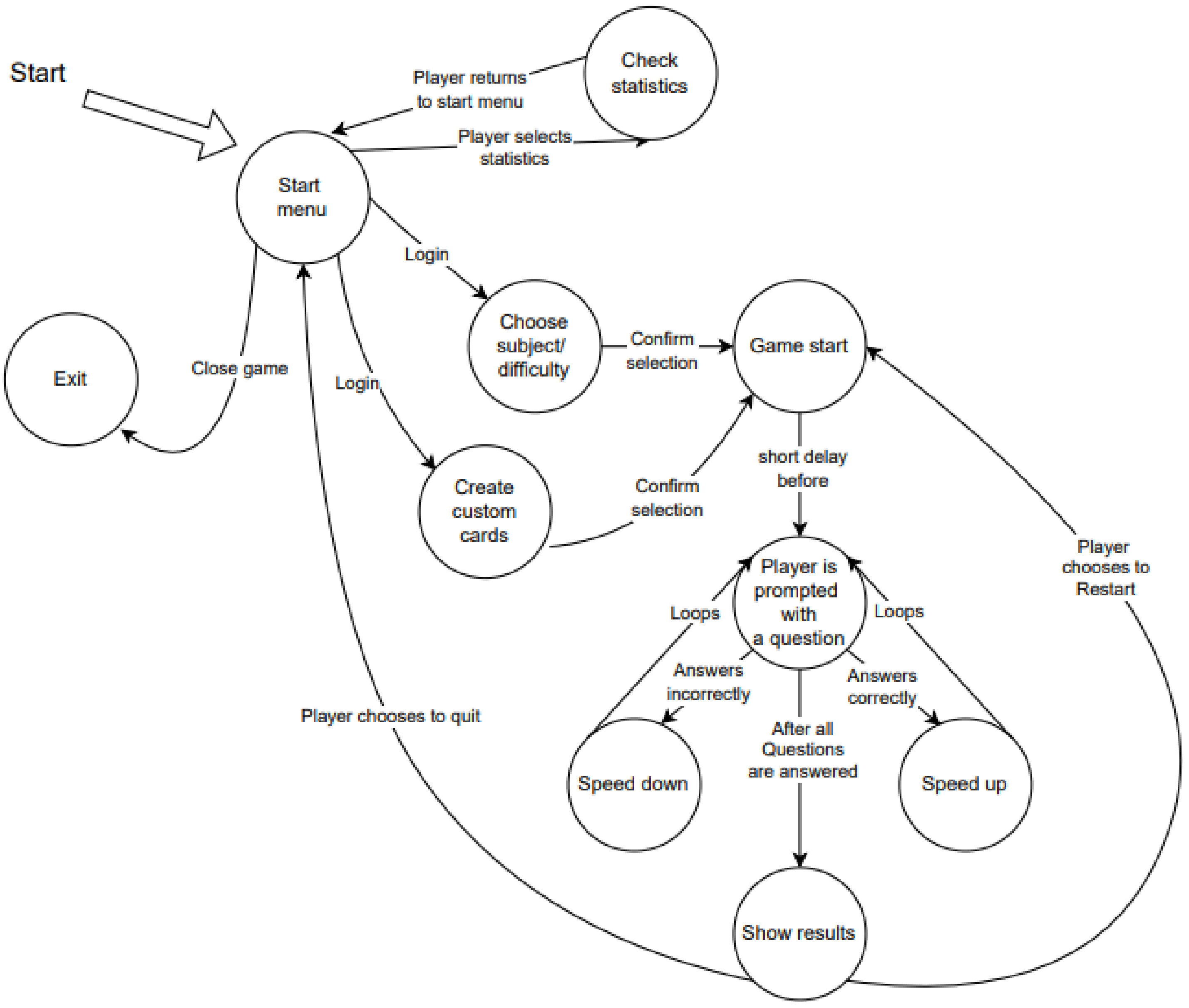
4.4.2 Sequence 2:

In the sequence the general game loop is shown as the user will log into the game, and start up a session. It also displays the game loop that the User will be engaged in once the game has started.



4.5 State Diagram

In the diagram below what is being displayed is how the state of the game will change as the user makes decisions and inputs. At the very beginning the user will be met with a start menu where they can choose to check their statistics, exit, or login to start the game. When logging in they are met with various options such as the subject they would like to study, the difficulty, or if they want to create their own custom set. Once they confirm their selection the game will start and the player will be prompted with their first question. Based on their answer the car will either speed up or slow down and then a new question will be asked. Once all questions have been answered the player will then have the choice of restarting or quitting back to the main menu.


Start
Start menu
Player returns to start menu
Check statistics
Player selects statistics
Login
Choose subject/ difficulty
Confirm selection
Game start
Exit
Close game
Login
Create custom cards
Confirm selection
short delay before
Player is prompted with a question
Player chooses to Restart
Loops
Loops
Answers incorrectly
Answers correctly
Player chooses to quit
After all Questions are answered
Speed down
Speed up
Show results

5 Prototype

The Prototype v1 will show the following functionality.

Selecting a subject and difficulty is done by interacting with dropdown menus displayed on the main menu screen. Both menus are displayed at the center-right or center-left of the screen and are labeled “Select Subject” and “Select Difficulty”. With your cursor, you can click on the down arrow on the right side of the menu. This allows you to select a subject or difficulty for the game.

Starting the game is done by clicking the “Start Game” button at the bottom of the screen. Doing so will load and display the gameplay screen.

Playing the game will begin once the gameplay screen is displayed. A car will appear on the left side of the screen and make its way along the track to the right side of the screen. The car moves if the player answers a question correctly. A question will be displayed at the top of the screen, and possible answers will be displayed below it. The player can click on the answer they want to give and the game will display if they got it correct or incorrect. A new question will then be displayed. After exhausting the entire list of questions, the game will be over, and you win.

5.1 How to Run Prototype

To run the prototype v1, visit the project's GitHub page. There you can copy the page URL and pull the repository to your remote file storage. Then you need to download Unity Hub 3.10. Once downloaded, you can create a new project by selecting the project's repository from your file storage. Unity will launch the game editor once you complete this process.

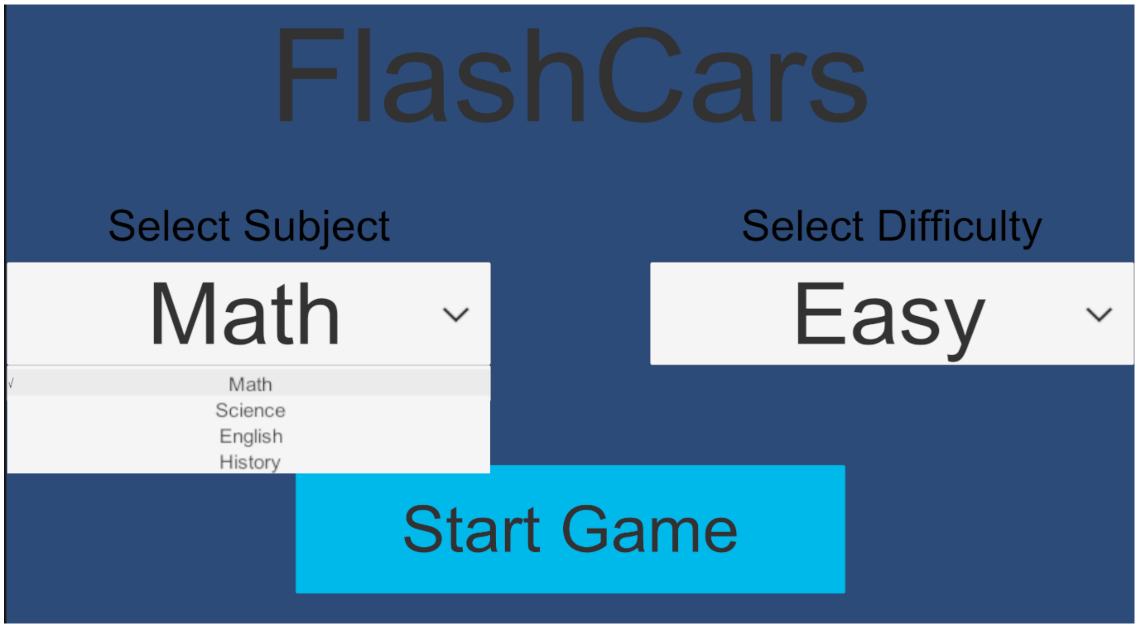
Now that you can open the game in Unity, simply press the play button at the top of the screen. This will execute the game file,s and you have successfully run the prototype.

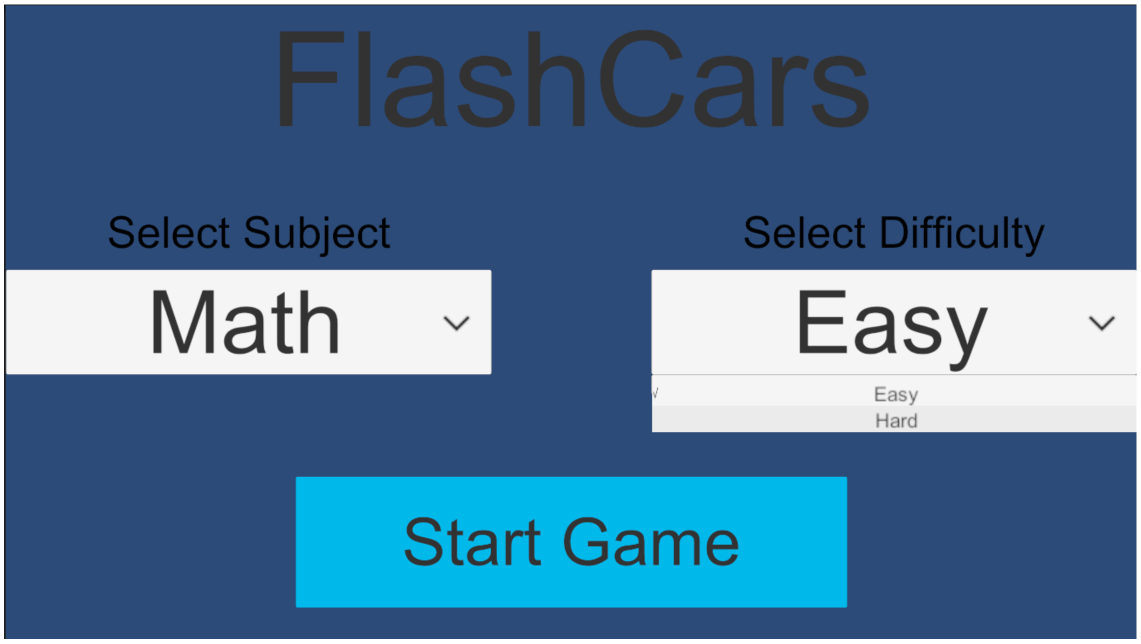
If the 'Start Game' button does not bring you to the next scene:

1. File > Build Profiles
2. Click on 'Scene List' under 'Platforms' on the left. If yo only see the MainMenu scene, this is the issue.
3. In the bottom window in Unity, open the 'Assets' folder, then 'Scenes'
4. Select the 'GamePlay' scene, go to File > Build Profiles, and under 'Scene List' select 'Add Open Scenes'
5. Repeat steps 3 and 4 for the 'Winner' scene to be able to navigate between all scenes

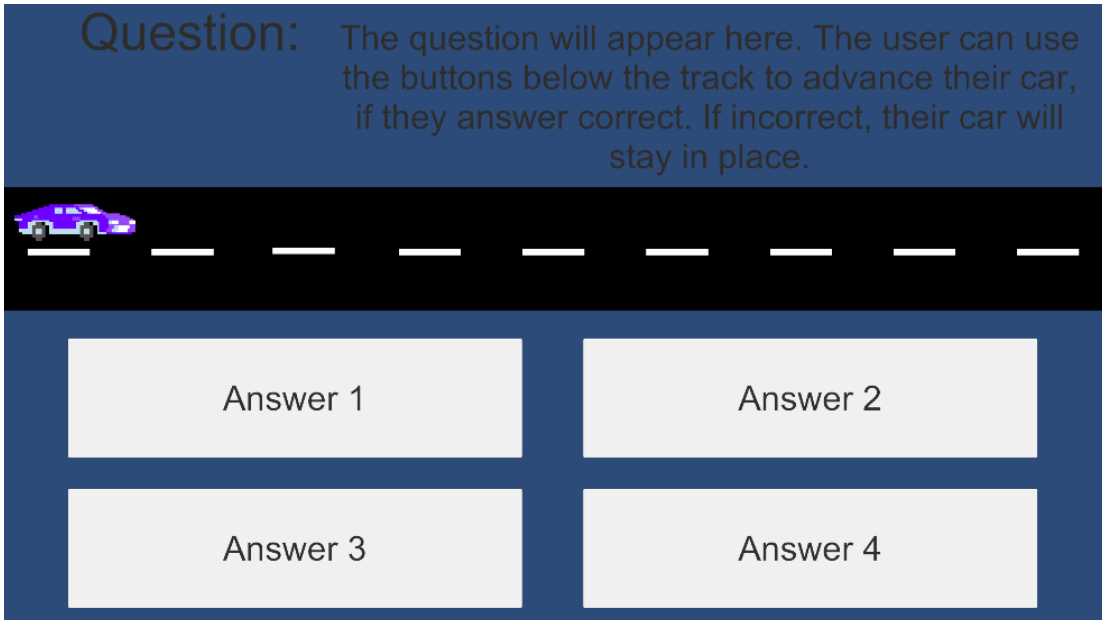
5.2 Sample Scenarios

A 4th-grade student wants to practice math using FlashCars. Once they successfully launch the game, the student is taken to the main menu screen, where dropdown menus will be displayed on the center-left and center-right. They select “Math” from the “Select Subject” menu and “Easy” from the “Select Difficulty” menu.



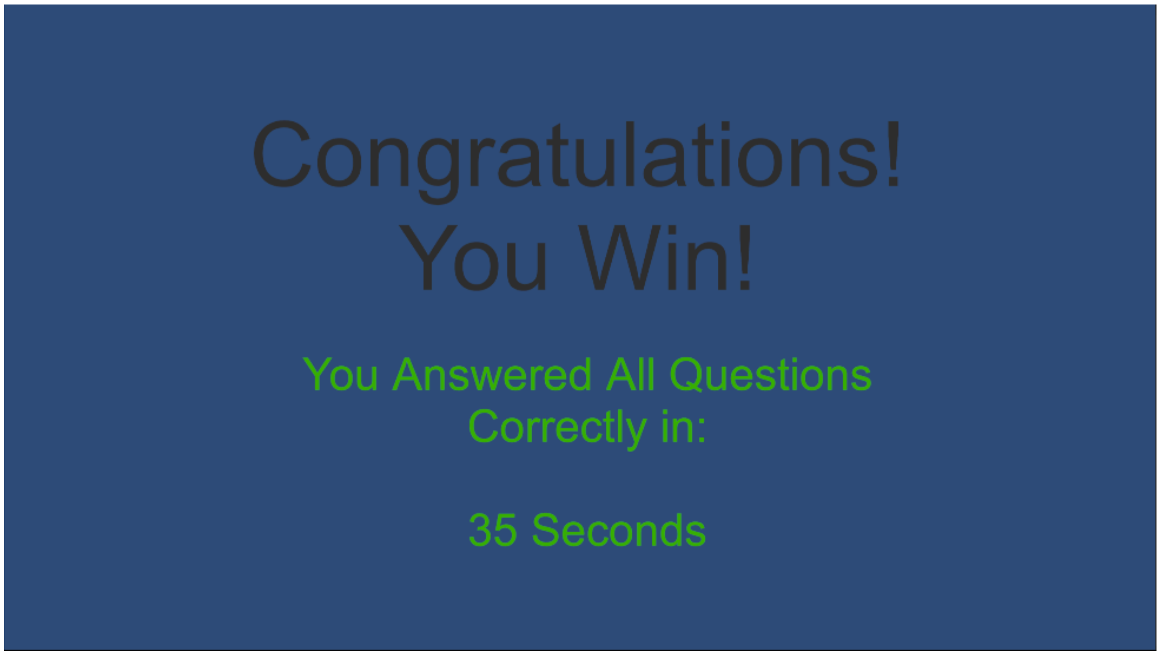


Once making their selections, the student will select the “Start Game” button on the bottom.



The question will appear here. The user can use the buttons below the track to advance their car, if they answer correct. If incorrect, their car will stay in place.

After answering the questions, the student's car will cross the finish line and the game displays a congratulatory message, "You Win!" with the player's performance statistics.



Congratulations! You Win!

You Answered All Questions Correctly in:

35 Seconds

6 References

[1] D. Thakore and S. Biswas, "Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks," Proceedings of IEEE Military Communication, Atlantic City, October 2005.

[2] Game Website: https://github.com/andrade01986219/FlashCars

[3] Car Image used in-game: https://assetstore.unity.com/packages/2d/pixel-cars-178447

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.