Software Requirements Specification (SRS)
FlashCars

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

Table of Contents

1 Introduction

This is a Software Requirements Specification document for our educational game FlashCars. FlashCars is a game that puts a spin on the traditional study method of flash cards by introducing the incentive of winning a race to make studying feel more rewarding. In this document FlashCars will be the main subject divided down into separate sections starting off in this section the purpose, scope, keywords/definitions, and overall organization of the document can be found here. In section two information on our perspective on our product, the major functions, the characteristics of our users, constraints, assumptions, dependencies, and apportioning of requirements is displayed. The third section hosts the specified requirements for FlashCars. Modeling requirements, multiple diagrams of the function of FlashCars, and a data dictionary can all be found in section four. In section five the results of the prototype build of FlashCars is displayed and discussed. Section six is the references and section seven is the point of contact.

1.1 Purpose

The purpose of this Software Requirements Specification document is to minimize ambiguity and miscommunication between all involved in FlashCars. The development team can use this as a blueprint for implementing the game's features and functionality. It is also used by 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. The main function of this game is in the flash card format style for studying, where questions are displayed to the user and answering correctly then rewards the player with moving their car forward. 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 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.

FlashCars - Name of the product

Car - a representation of the player that is manipulated by the answer the user provides.

Card - the displayed question that must be answered by the user to progress the game.

Question - A prewritten question based on either math, language arts, history, or science the user will need to answer to progress the game.

Subject - mentioned above this is the topic of the question that will be displayed to the user.

1.4 Organization

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

Section 2: An overall description of the project and its intended behavior including its 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

The game FlashCars will serve as a multi purpose study tool that 4th and 5th grade students can use to practice and reinforce what they are learning in the classroom. The educational content of the game comes from questions based on the general curriculum of 4th and 5th graders in Massachusetts. The game on top of reinforcing what the student has already learned also builds good study habits for the student that can benefit them moving forward in their academic career. The gameplay is simple and straightforward, after choosing their difficulty and subject the user will be prompted a random question from a premade set and will have to answer by typing in their response. Then based on a correct or incorrect response the car will move forward or not at all.

2.2 Product Functions

The 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, 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.

When first opening the game the first function of the game is creating 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.

The Diagram below is the high level goal diagram for FlashCars. Represented is the goal that we are ultimately trying to achieve and how we are trying to achieve this. It is derived from three components being the choice of subject, choice of difficulty, and the ncp car the player plays against to introduce a competitive component for engagement with the game.

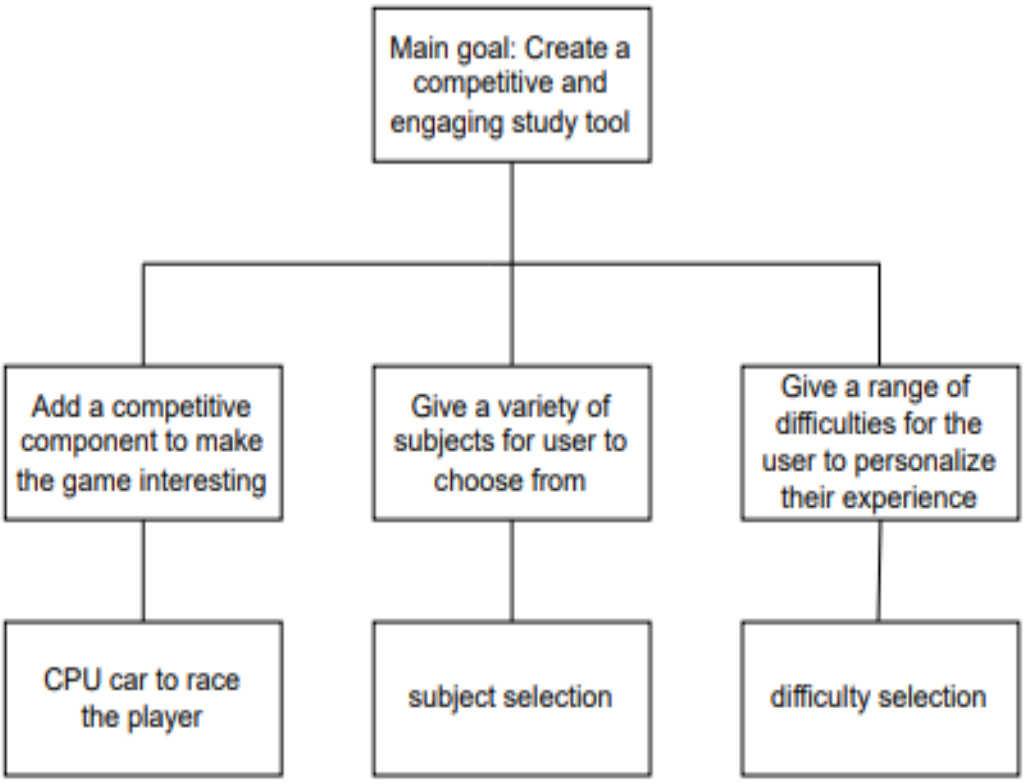

Figure 1. High level goal diagram of our objective with FlashCars

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 of math, history, and science at the 4th and 5th grade level listed in the Massachusetts Department of Education curriculum

2.4 Constraints

- 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.

2.5 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

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 two difficulties from which the player can select.

4. The game will display:

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

    4.2 A question.

    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.

6. The game will track player's time and display upon winning the race.

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

    7.1 This will be shown by the player reaching the finish line.

    7.1 This will bring the player to another screen with a congratulatory message.

8. The game will store player data; including name, username, and password

    8.1 The password will be stored as a sha256 hash.

4 Modeling Requirements

4.1 Use case Diagram

The use case diagram below depicts the general functions a user will be able to use when playing the game.Mainly there are two direct functions presented to the user, a login that is used to personalize the game to the user, and the game itself. Extended from the login is the ability to create an account which will be done the first time the game is launched, this sets up the use of the login feature for every use after the first. Included in the play game use are three major uses that are answering a question, choosing a difficulty, and choosing a subject. The choosing of a difficulty and subject is fairly straightforward but necessary for being able to start the game. Once in the game the main function of progressing is answering the questions. The extensions of answering a question are if the answer given is correct or incorrect. For an incorrect answer the progression of the game is halted until the correct answer is given at that point a new question is displayed and the player's car is moved along the track a set distance.

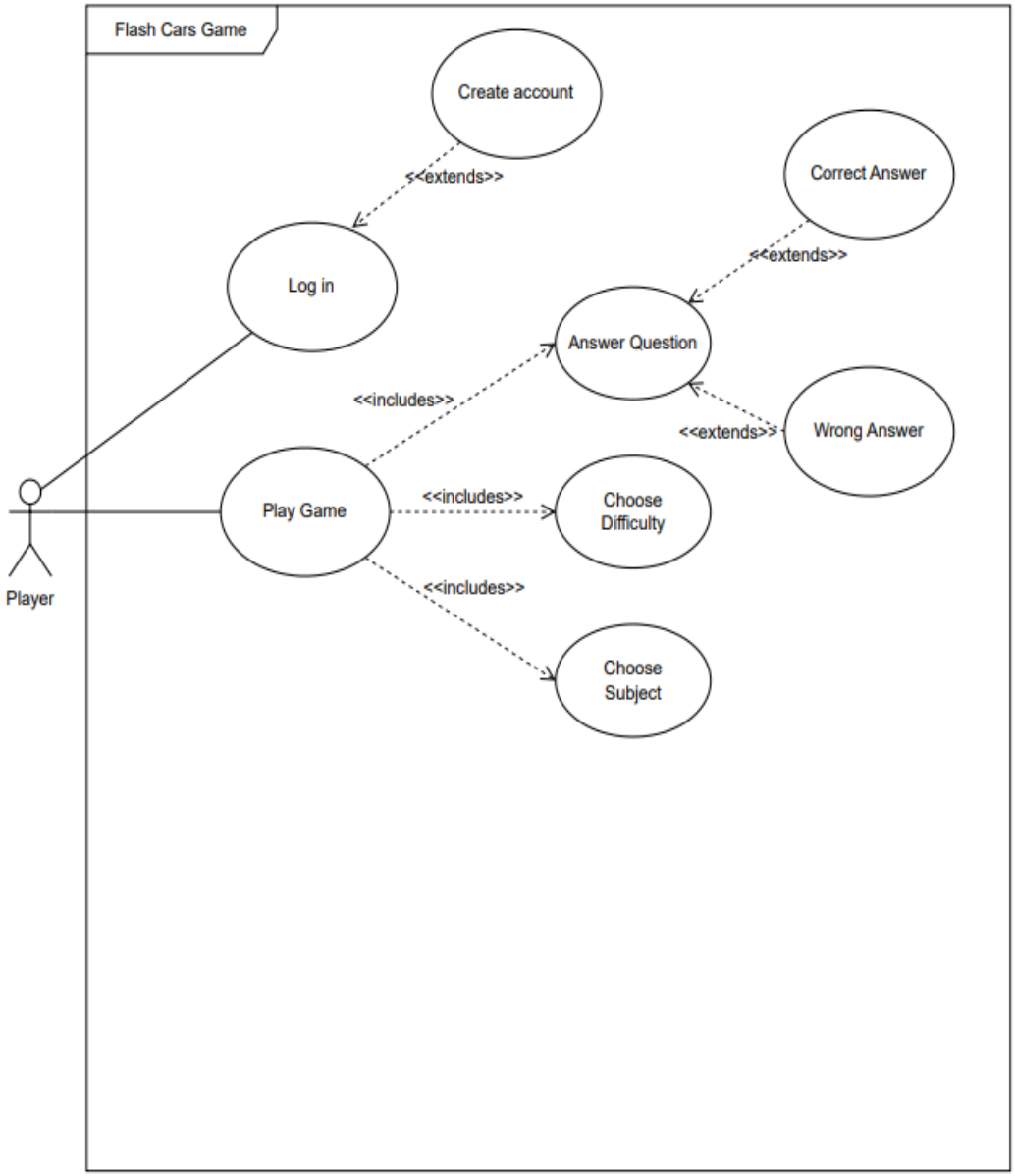

Figure 2. Use case diagram of majors uses of FlashCars.

Use Case Name:	Login
Actors:	Player
Description:	After the creation of an account the user will need to enter a username and password to play the game
Type:	Primary
Includes:	none
Extends:	none
Cross-refs:	none
Uses cases:	User will use this to have a more personalized experience granting them access to the game and also displaying it in game as well

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

Use Case Name:	Create Account
Actors:	Player
Description:	When first launching the game the player must .
Type:	Secondary and essential
Includes:	none
Extends:	Login
Cross-refs:	none
Uses cases:	Create a new subject of cards.

Use Case Name:	Incorrect Answer
Actors:	Player
Description:	The answer given by the player was incorrect.
Type:	Secondary and essential
Includes:	none
Extends:	Answer Question
Cross-refs:	none
Uses cases:	halts the progression of the game until the user gives the correct answer.

Use Case Name:	Correct Answer
Actors:	Player
Description:	The answer given by the player was correct.
Type:	Secondary and essential
Includes:	none
Extends:	Answer Question
Cross-refs:	none
Uses cases:	Moves the car forward and prompts the user with the next question.

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 seven 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, and deals with creating an account and logging into the game. The question class handles the managing of all the questions that will be pulled by FlashCars to then be displayed to the user. NPCCAr is the class representation of the npc car the user will race against. The car class is the user's car and will be manipulated according to the inputs by the user. Finally the SubjectSelection and DifficultySelection handle the saving of the choice the user makes before starting the game and then relaying the choice to the FlashCars class.

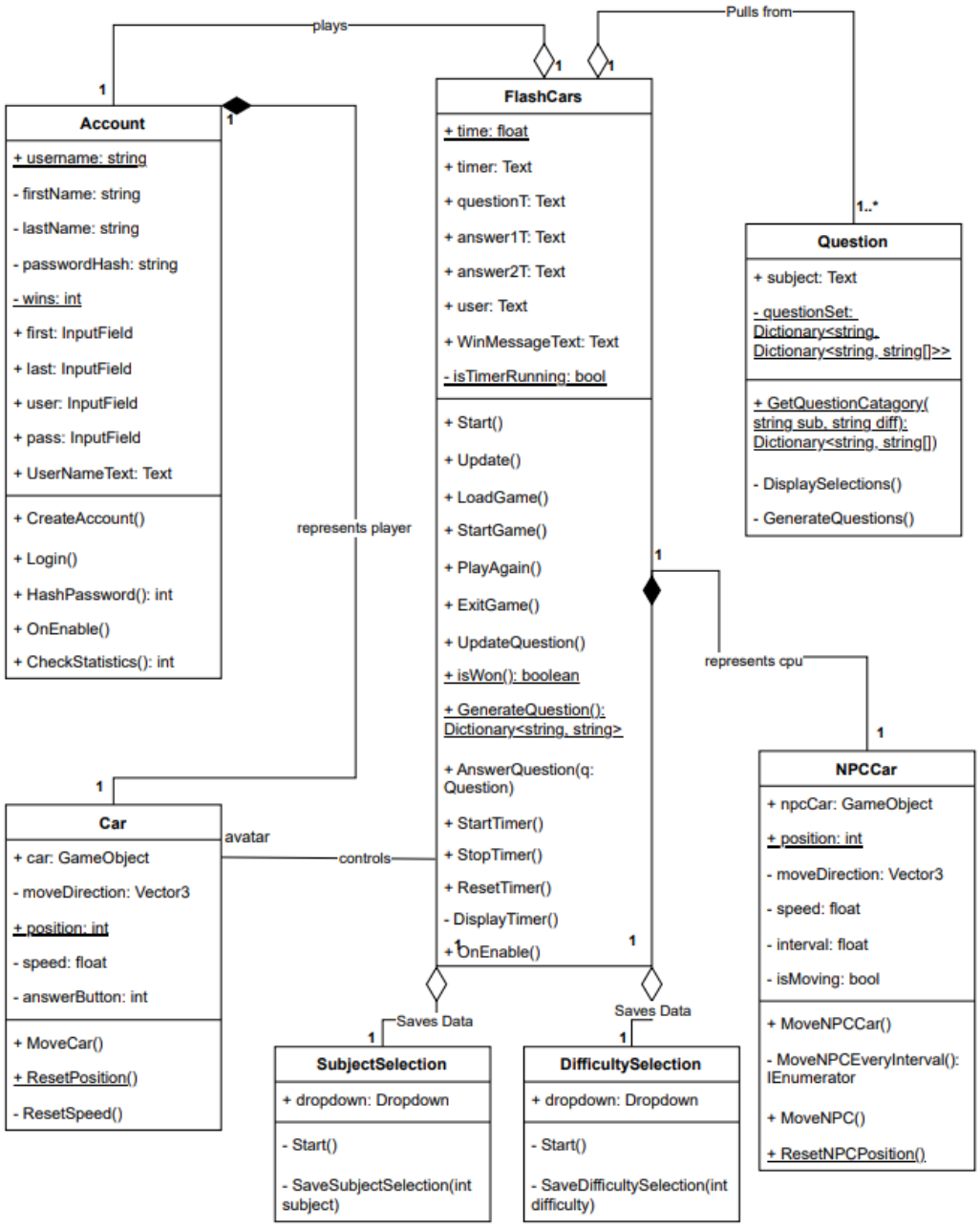

Figure 3. Class diagram of FlashCars showing the major classes and how they communicate with each other.

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
	-firstname: string	inputted first name of user
	-lastname: string	inputted last name of user
	-wins: int	number of wins by user
	-passwordHash: string	password entered by user
	+first: InputField	where user will input first name
	+last: InputField	where user will input last name
	+user: InputField	where user will input username
	+pass: InputField	where user will input password
	+UserNaneText: Text	displays user name
Methods		
	+CreateAccount()	allows user to create a personal account
	+Login()	User enters username and

		password to enter the game
	+HashPassword(): int	converts imputed password to hash bytes, then to a string and returns the string
	+OnEnable():	allows the user to play the game after logging in successfully
Relationships	FlashCards - Creates an account to use and represent themselves as in the game 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		
	+time: float	how long the session has lasted
	+timer: Text	display of the time to the user
	+questionT: Text	question displayed to the user
	+answer1T: Text	1st answer displayed to the user
	+answer2T: Text	2nd answer displayed to the user
	+user: Text	displays the username of the user
	+WinMessageText: Text	text displayed when user wins
	-isTimerRunning: bool	checks if timer is running
Methods		
	+Start()	starts the program

	+Update()	updates the state of the game
	+LoadGame()	will load the selections of subject and difficulty the user makes into the game
	+StartGame()	starts the game presenting the first question and moving the npc
	+PlayAgain()	replays the game
	+ExitGame()	exits the game
	+UpdateQuestion()	after a correct answer this changes the question
	+isWon(): boolean	checks if the game has been won
	+GenerateQuestion(): Dictionary<string, string>	holds all the questions will be called to display the question
	+AnswerQuestion(q: Question)	judges if the correct answer has been chosen of a specific question
	+StartTimer()	starts the timer
	+StopTimer()	stops timer
	+ReseetTimer()	resets timer
	-DisplayTimer()	displays timer on screen for user
	+OnEnable()	allows the user to play the game after logging in successfully
Relationships	Account - Is directly accessed from the account and played through	

	that. Car - Receives commands through FlashCars based on gameplay from the user. Question - Pulls from this class the question that will be displayed to the user. NPC Car - Controls the cpu car. SubjectSelection - Receives what subject has been chosen. DifficultySelection - Receives what difficulty has been chosen.

Element Name		**Description**
Car		the representation of the player inside the game
Attributes		
	+car: GameObject	the car object
	-moveDirection: Vector3	direction of the car
	+position: int	where the car is
	-speed: float	how fast the car moves
	-answerButton: int	button user clicks to answer a question
Methods		
	+MoveCar()	moves the car

	+ResetPosition()	after the game is finished the position of the car is reset
	-ResetSpeed()	resets car speed
Relationships	Account - Is a representation of the user by way of the account. FlashCars - receives commands through FlashCars based on gameplay from the user.	

Element Name		**Description**
Question		Will handle presenting questions and answers to user
Attributes		
	+subject: Text	topic of questions that will be asked
	-questionSet: Dictionary<string, Dictionary<string, string[]>>	set of questions for each subject and difficulty
Methods		
	+GetQuestionCatagory(string sub, string diff): Dictionary<string, string[])	retrieves correct question and answer set
	-DisplaySelections()	displays the selection of subject and difficulty
	-GenerateQuestions()	creates a map of all the questions

Relationships	FlashCars - Provides questions and answers to FlashCars.

Element Name		**Description**
NPCCar		the opposing car in the game the user will race against
Attributes		
	+npcCar: GameObject	the npc car object
	-moveDirection: Vector3	direction of the car
	+position: int	where the car is
	-speed: float	how fast the car moves
	-interval: float	time interval until npc car moves
	-isMoving: bool	checks if npc car is moving
Methods		
	+MoveNPCCar()	moves the npc car
	-MoveNPCEveryInterval(): IEnumerator	determines the interval that the npc car moves at
	+MoveNPC()	debugger that tells if the npc car is moving or not
	-ResetNPCPosition()	resets npc car position
Relationships	FlashCars - Gives representation of the npc opponent that the player races against.	

Element Name		**Description**
SubjectSelection		handles the subject that the user wants to play
Attributes		
	+dropdown: Dropdown	dropdown menu on the main menu for the subject
Methods		
	-Start()	creates the dropdown
	-SaveSubjectSelection(int subject)	saves selected subject to determine what subject will be played
Relationships	FlashCars - Allows the selection of subjects before playing the game.	

Element Name		**Description**
DifficultySelection		handles the difficulty the user wants to play
Attributes		
	+dropdown: Dropdown	dropdown menu on the main menu for the subject
Methods		
	-Start()	creates the dropdown
	-SaveSubjectSelection(int subject)	saves selected difficulty to determine what difficulty will be played
Relationships	FlashCars - Allows selection of difficulty before playing the game.	

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 logging into the game.

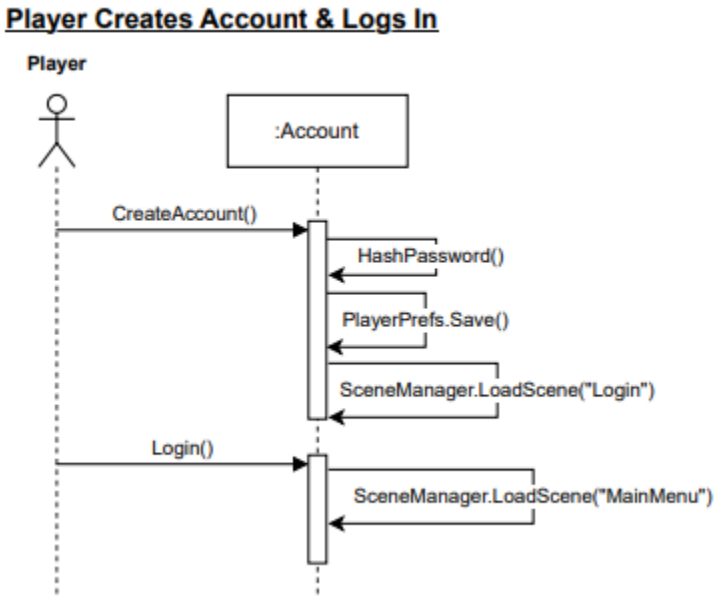

Figure 4. Sequence diagram of making an account and logging in

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. The user initially logs in,selects both a subject (math, history, or science), and then starts the game. Once the game has started the main game loop starts and a question will then be retrieved and displayed to the player. The player then inputs their response which will then be checked for correctness. If it is correct the car will move forward and if not then the car will stay still. This loop will be repeated until all questions have been exhausted.

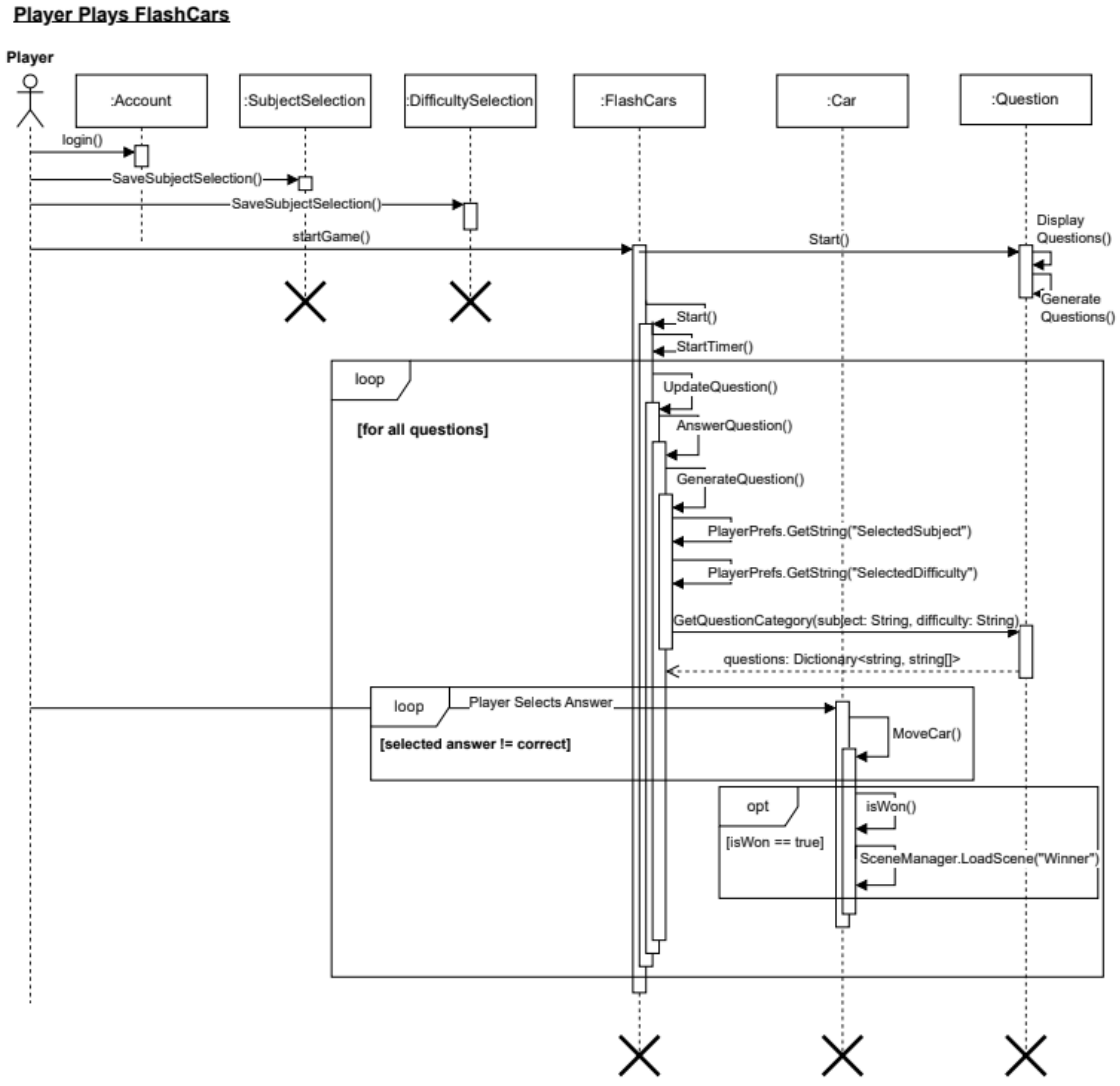

Figure 5. Sequence diagram of a play through of FlashCars.

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 create account menu where they can make their account. After that they will be asked to log in on the next screen. Once in the main menu the user then can choose their subject they would like to study and what difficulty they would like to play on. 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 move a set distance forward or not move at all until the question is answered correctly, once either of these conditions are met then a new question will be prompted to the user. Once all questions have been answered the player will then have the choice of restarting or quitting back to the main menu.

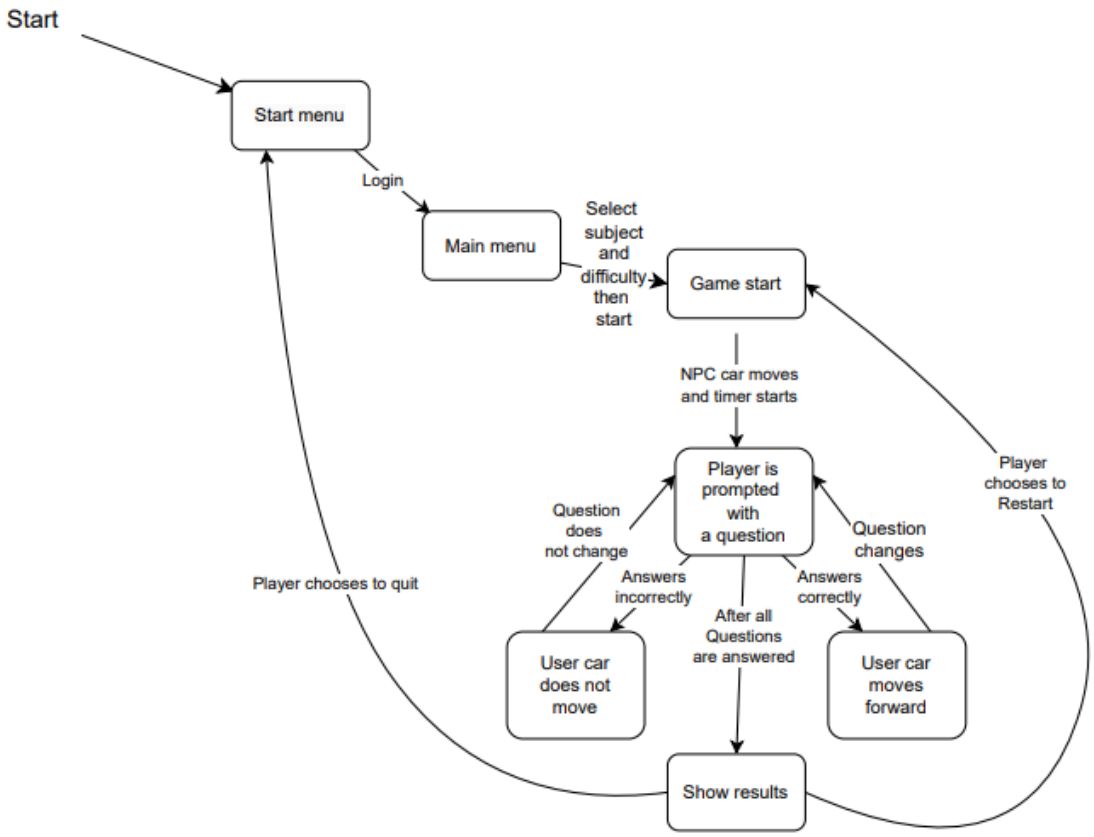

Figure 6. State Diagram of the different states FlashCars can progress through and one state flows to another.

5 Prototype

The Prototype v2 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-left or center-right of the screen and are labeled “SUBJECT” and “DIFFICULTY”. Clicking on the down arrow on the right side of the menus will reveal the options for each menu. This allows the user to select a subject or difficulty for the game. Pressing “LOAD GAME” will show the screen where the user will play the game.

At first, some text to show where the question will be displayed along with the track with the user’s car and the CPU car the user will be competing against, the two answer buttons, the selected subject and difficulty, the user’s username, a button labeled “START GAME”, and a timer that will start once the button is pressed. Starting the race is done by clicking the “START GAME” button at the bottom of the screen.

Once the “START GAME” button is pressed, the questions and answers are displayed, the timer starts, and the CPU car moves forward. The car moves if the user answers a question correctly. The user can click on the answer they want to give and their car will move forward if they answered correctly or nothing if they answered incorrectly. Once the user’s car moves forward, a new question will be displayed. After exhausting the entire list of questions to reach the finish line, the game will be over and the screen to show the user won will be displayed. If the CPU car gets to the finish line before the user, the screen to show the user lost will be displayed. In each of these screens, the winner's time will be displayed along with the option to play again or exit to the main screen.

5.1 How to Run Prototype

To run the prototype v2, 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 in Unity Hub, download the required files including the Unity Editor. Once these downloads are finished, click the “Add” dropdown, select “Add project from disk,” and select “FlashCarsPrototype” inside of the FlashCars repository. Then you can select “FlashCarsPrototype” from the list of projects and the Unity Editor will open the selected project. Now that you have the game open in the Unity Editor, in the bottom section of the screen under the “Project” tab navigate to the “Scenes” folder found in the “Assets” folder. Then, select the “CreateAccount” scene to create a username that will be displayed on screen or the “MainMenu” scene to select the desired subject and difficulty. Then, simply press the play button at the top of the screen to run the prototype.

If the “LOAD 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 you 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 “SUBJECT” menu and “Easy” from the “DIFFICULTY” menu. Once making their selections, the student will select the “LOAD GAME” button.

Figure 7. Displays the ability to choose the subject and difficulty.

This will bring them to the game screen where they can get an understanding of the layout before starting the game.

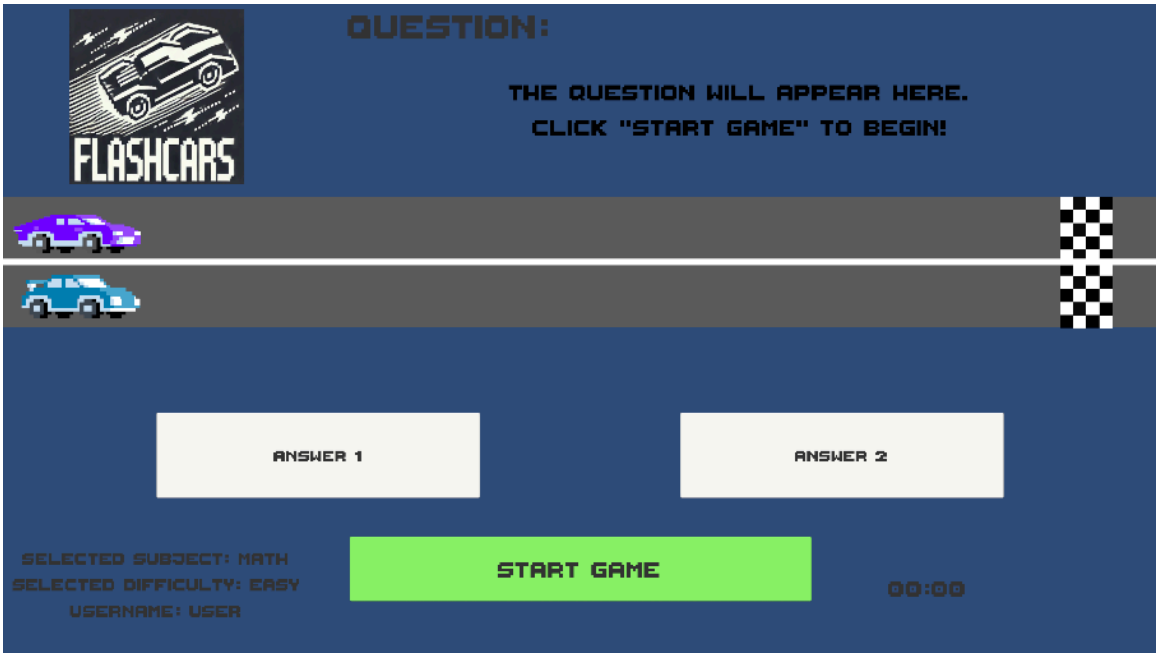

Figure 8. Displays an example of what the user will see before playing FlashCars.

Once the game is started, the CPU car will start moving periodically and the student will answer questions in an attempt to reach the finish line before the CPU car.

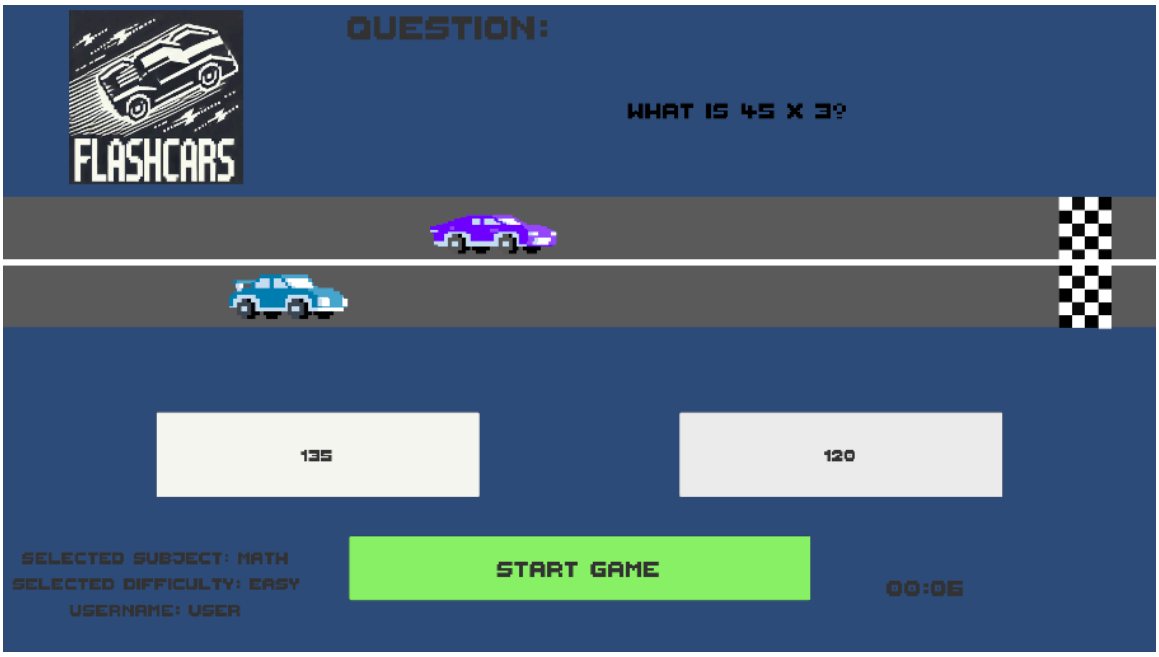

Figure 9. Displays an example of what the user will see while playing FlashCars.

After answering all questions and reaching the finish line before the CPU car, the game displays a congratulatory message, “You Win!” with the player’s performance statistics.

Figure 10. The screen displays when the user achieves victory.

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] Unity Technologies, *Unity Game Engine*, Version Unity 6. [Online]. Available: https://unity.com/products/unity-engine

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

[4] 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.