**Highway Racers**

**SPROJ Report**



**Abdullah Ahmad 23100018**

**Abeeha Ishfaq 23100336**

**Aman Tariq 23100167**

**Bakhtiar Rasheed 22100267**

**Muhammad Hamza 23100247**

**Advisor: Waqar Ahmed  
School of Science and Engineering  
Lahore University of Management Sciences  
21 April 2023**

**Acknowledgment and Dedication**

1. We are grateful to Dr Waqar Ahmed and the Computer Science department at LUMS for introducing Game development as a Senior project for the first time and in collaboration with Mindstorm Studios.
2. This project has helped us build new leading skills and prepared us for a possible career in the game industry.

**Certificate**

I certify that the senior project titled “**Add project title here**” was completed under my supervision by the following students:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

and the project deliverables meet the requirements of the program.

------------------------------------- Date:

**Advisor (Signature)**

------------------------------------- Date:

**Co-advisor (if any)**

# Table of Contents

[Table of Contents 4](#_Toc134520981)

[List of Figures 6](#_Toc134520982)

[1. Introduction 7](#_Toc134520983)

[a. Introduction 7](#_Toc134520984)

[b. Objective and Scope 7](#_Toc134520985)

[c. Development Methodology 7](#_Toc134520986)

[d. Contributions 8](#_Toc134520987)

[2. System Requirements 9](#_Toc134520988)

[a. System Actors 9](#_Toc134520989)

[b. Functional Requirements 9](#_Toc134520990)

[c. Non-functional Requirements 10](#_Toc134520991)

[3. System Architecture 11](#_Toc134520992)

[a. Architecture Diagram 11](#_Toc134520993)

[b. Architecture Description 11](#_Toc134520994)

[c. Justification of the Architecture 12](#_Toc134520995)

[d. Tools and Technologies 14](#_Toc134520996)

[4. Game Narrative 16](#_Toc134520997)

[5. Literature Study 17](#_Toc134520998)

[5.1. Mindstorm Studios Summer Program’s Playlists 17](#_Toc134520999)

[5.2. The Art of Game Design: A Book of Lenses 18](#_Toc134521000)

[6. Software Development Methodology and Plan 20](#_Toc134521004)

[a. Software Process Selection 20](#_Toc134521005)

[7. System User Interface 21](#_Toc134521006)

[8. Testing and Evaluation 26](#_Toc134521007)

[9. Deployment Guidelines 27](#_Toc134521008)

[10. Lessons Learnt 28](#_Toc134521009)

[11. Conclusion 29](#_Toc134521010)

[a. Summary 29](#_Toc134521011)

[b. Challenges 29](#_Toc134521012)

[c. Future 30](#_Toc134521013)

[12. Review checklist 31](#_Toc134521014)

[13. References 32](#_Toc134521015)

# List of Figures

|  |  |  |
| --- | --- | --- |
| Image Number | Description of Image | Section of Image |
| 1 | Diagram of system architecture | 3a |
| 2 | Start screen of Highway Racers | 7.1 |
| 3 | Settings Screen of Highway Racers | 7.2 |
| 4 | Cinematics Screen of Highway Racers | 7.3 |
| 5 | 1st image Gameplay Screen of Highway Racers | 7.4 |
| 6 | 2nd image Gameplay Screen of Highway Racers | 7.4 |
| 7 | Pause Screen of Highway Racers | 7.5 |
| 8 | Ending Screen of Highway Racers | 7.6 |

# Introduction

## Introduction

Highway Racers is an endless runner game designed for mobile devices that offers a thrilling and exciting experience to players. The game features a coin collection and a refueling tank system that adds an extra layer of complexity and strategy to the gameplay.

The game falls under the category of casual arcade games and is designed for mobile devices such as smartphones and tablets. It is developed to provide an engaging and entertaining experience for users who enjoy playing arcade-style games.

The target audience for Highway Racers is primarily individuals who enjoy playing games on their mobile devices. The game is suitable for people of all ages, especially those who love arcade-style games. The game is designed to be easy to play, yet challenging enough to keep players engaged and entertained for extended periods.

## Objective and Scope

The main objective of Highway Racers is to achieve the highest score possible while avoiding obstacles and collecting coins. The refueling of the tank system adds an additional layer of gameplay, making it more challenging for players to achieve their objective. The game is designed to be addictive and challenging, keeping players engaged and striving to improve their scores with each playthrough.

The scope of Highway Racers is limited to the gameplay itself. The game does not include any social features, such as multiplayer or online leaderboards. However, players can compare their scores with friends by sharing screenshots or playing together in person. The game is designed to be a standalone experience that provides players with an entertaining and engaging gameplay experience that they can enjoy on their mobile devices.

## Development Methodology

As game developers working on Highway Racers, we utilized an Agile development methodology, specifically Scrum, to guide the development process.

The Scrum development process involves dividing the development cycle into short iterations or sprints, typically lasting between one to four weeks. At the beginning of each sprint, we would hold a sprint planning meeting to identify the game's features and prioritize them based on their importance.

During the sprint, the team would hold daily stand-up meetings to discuss the progress of the project and identify any issues or roadblocks that need to be addressed. At the end of each sprint, we would hold a sprint review to showcase the completed work and gather feedback from stakeholders. This feedback would then inform the next sprint.

After each sprint review, the team would hold a sprint retrospective meeting to discuss the successes and challenges of the sprint and identify areas for improvement. Throughout the development cycle, we would utilize continuous integration to regularly integrate code and assets into the game to ensure that it functions properly.

To ensure the game is of high quality, we would conduct testing throughout the development process to identify any issues or bugs that need to be addressed. Once the game is complete, we would plan the release and develop a plan for ongoing maintenance and support

## Contributions

One of the innovative aspects of Highway Racers is its unique combination of features. The endless runner gameplay, coin collection, and refueling of the tank system provide players with a challenging and entertaining experience. These features add an extra layer of complexity and strategy to the game, setting it apart from other endless runner games.

Furthermore, the game's development on the Unity engine allows for high-quality graphics and smooth gameplay, which enhances the overall player experience. Unity has become one of the most popular game engines for developing mobile games due to its cross-platform capabilities and its ability to handle complex game mechanics.

In terms of how Highway Racers is better than other similar solutions, its unique combination of gameplay features and accessibility sets it apart from other endless runner games. Many endless runner games rely solely on simple gameplay mechanics such as jumping or sliding, without adding any additional layers of complexity.

In conclusion, the innovative combination of features, accessibility, and development on the Unity engine make Highway Racers a unique and entertaining game that sets itself apart from other similar solutions in the endless runner game genre.

# System Requirements

## System Actors

|  |  |
| --- | --- |
| **Actor Name** | **Description** |
| Player | The user controls the virtual bike on the racing track, trying to avoid obstacles and traffic and collect coins. |

## Functional Requirements

* **Bike Control**

The game allows the player to control the bike's sideways movement using buttons on a touchscreen.

* **Track Design**

The game includes multiple track designs with varying obstacles and scenery.

* **Obstacles**

The game includes different types of obstacles that can affect the player's speed or cause the player to crash and the game to end.

* **Score System**

The game includes a scoring system that reflects the player's performance, based on factors such as distance traveled, and coins collected.

* **Sound Effects and Music**

The game should include appropriate sound effects for bike movements, collisions, and power-ups, as well as background music that enhances the game's atmosphere.

* **User Interface**

The game should have a user-friendly interface that allows the player to navigate through menus, select race modes, and view game statistics.

## Non-functional Requirements

* **Performance:**

The game is responsive and runs smoothly on different devices, without lag or stuttering.

* **Compatibility:**

The game is compatible with different platforms, such as Windows, iOS, and Android.

* **Usability:**

The game is easy to learn and use, with clear instructions and intuitive controls.

* **Accessibility:**

The game is accessible to players with disabilities, with features such as keyboard-only controls, high-contrast mode, and closed captioning.

* **Reliability:**

The game is stable and dependable, with minimal crashes or errors, and quick recovery from failures.

* **Scalability:**

The game can handle a large number of downloads without compromising performance or usability.

* **Localization:**

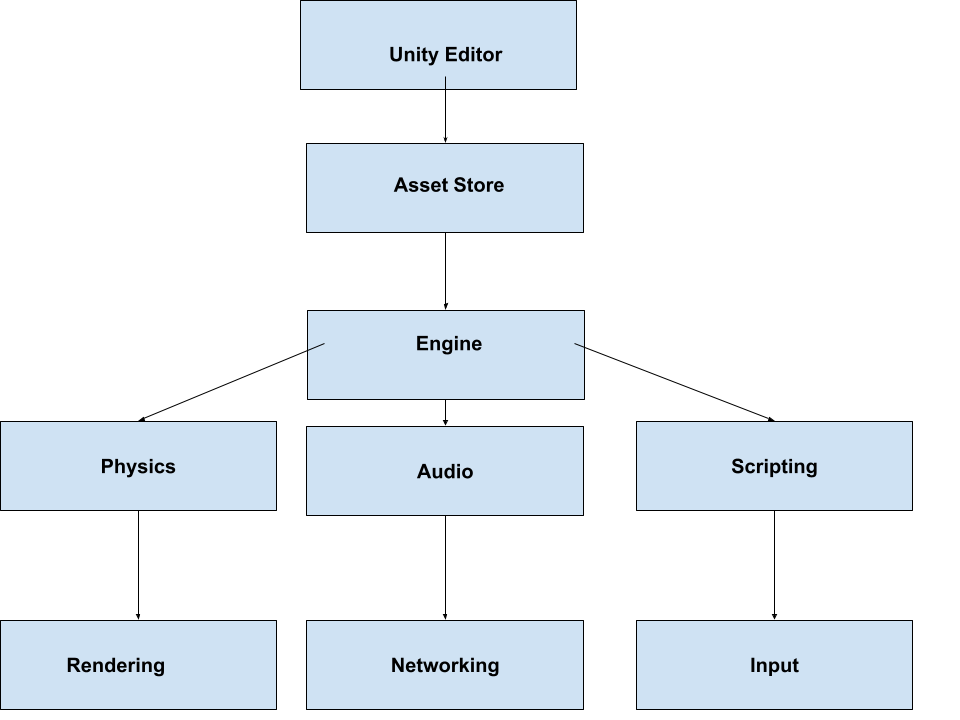
The game is localized to the Pakistani environment with appropriate graphics and audio.

* **Aesthetics:**

The game has appealing graphics, animations, and visual effects that enhance the game's atmosphere and immersion.

# System Architecture

## Architecture Diagram



## Architecture Description

The Unity Editor is the main interface for creating and managing Unity projects. The Asset Store is a marketplace where developers can find and purchase assets, plugins, and other resources for their projects.

The Engine is the core component of the Unity game engine, responsible for managing game objects, scenes, physics, rendering, scripting, and other features.

The Engine is divided into several subsystems, including Physics, Audio, Scripting, Rendering, Networking, and Input, each of which is responsible for specific functionality.

## Justification of the Architecture

**Pros:**

* Unity Editor provides a user-friendly UI for developers to create and manage projects and design game scenes interactively. Using the unity editor, we created different prefabs(models) of game objects for each new scene and we adjusted the scene to our liking with a predefined set of controls such as movement, rotation, and scaling.
* The Asset Store has a vast library of resources for different assets in the game such as scripts, UI, game objects for different game scenarios, and multiple levels of customization. We used pre-built assets such as bikes, environment, and UI from the asset store as it saved time and effort for asset creation and also got new a inflow of ideas by discovering the works of other people on the asset store.
* Unity Engine’s modular subsystems allow for efficient management and optimization of specific functionalities thus improving performance and scalability. Each game object in Unity can have different components such as an animation controller, audio source, and colliders so when we were making our game and wanted to add physics and audio to our bike, we just added two components, rigid body (to provide physics functionality for the bike) and audio source( to control audio playback for the bike) and were able to accomplish the task of integrating multiple characteristics for the bike with the help of Unity game engine’s simplified developer API for creating functional game objects with ease.

**Cons**

* While the modular system provides a simple API for integrating multiple functionalities into our game object, it may result in increased complexity and potential dependencies between subsystems and become harder to debug. Such issues did occur in the development of our game, for example when we were adding different triggers for the bike to increase coin count when it collided with the coin, we faced multiple issues when trying to add audio feedback of coin collection within the trigger as well since we were working with multiple systems of unity game engine (colliders, event system, audio) at the same time and it took some time getting used to.
* The Asset Store may lead to over-reliance on pre-built assets and limit creativity and customization for game designers. While we did use pre-built assets from the asset store, we also designed some assets on our own after getting ideas from the assets built by the community in the asset store.
* Unity’s cross-platform support may lead to compatibility issues and additional testing and optimization efforts. We also faced such trouble when we were trying to test our game on mobile. Since we were making our game on PC, it was working and displaying UI correctly but when we tested it on mobile, the UI was not scaling and not showing properly. But afterward, we worked on the UI issues and fixed almost all of them.

The Unity system architecture is highly appropriate for our bike racing game, as it provides a powerful and efficient framework for developing a 3D game with complex physics and rendering requirements. The modular subsystems of the Engine, such as Physics and Rendering, are critical for implementing realistic bike physics and graphics in the game. Additionally, the Asset Store can be utilized to find pre-built assets such as bike models, road textures, and sound effects, saving time and effort in asset creation. Unity's cross-platform support is also beneficial for our game, as it allowed us to deploy our game on the Android platform without switching to a new language or tool.

To meet the performance requirement, Unity provides several tools and features that allow developers to optimize the game's performance, such as the Profiler, which helps identify performance bottlenecks and optimize the game's code and assets. Unity also offers features such as Asset Bundles, which allow developers to efficiently manage game assets and reduce the game's memory footprint.

To meet the compatibility requirement, Unity's cross-platform support allows developers to build games that can run on different platforms, such as Windows, iOS, and Android. Unity provides a single codebase that can be easily ported to different platforms, reducing the need for platform-specific code.

Unity's user-friendly interface and scripting language, C#, make it easy for developers to build games that are easy to learn and use, meeting the usability requirement. Unity also offers a wide range of pre-built UI components that can be customized to meet the specific needs of the game.

To meet the accessibility requirement, Unity offers features such as keyboard-only controls, high-contrast mode, and closed captioning, which can be easily implemented using Unity's scripting language and user interface tools.

Unity's robust testing and debugging tools, such as Unity Test Runner and Unity Remote, enable developers to build games that are reliable and stable, meeting the reliability requirement. Unity's ability to handle a large number of downloads, without compromising performance or usability, meets the scalability requirement.

## Tools and Technologies

List down the development stack, tools, technologies, etc. that you have used for development and deployment. Make sure that you mention the name and version of the tools.

We used Unity Game Engine and C# as the scripting language for building our game. The Unity version used 2021.3 LTS since it was a long-term support product and had all the latest features of Unity. We have deployed the Android Application Package (APK) of our game which can be installed on an Android smartphone and then can be played

# 

# Game Narrative

In Highway Racers, players take on the role of a daring bike driver who has stolen a high-performance bike, prompting a thrilling police chase. The game's objective is to outmaneuver the pursuing police cars and avoid capture while racing through the city streets.

The game features a range of challenging obstacles, including speeding cars and trucks, roadblocks, and barriers. To overcome these obstacles, strategically placed health pickups are available along the way, allowing the player to heal their bike after sustaining damage.

The game also includes a variety of different terrains, each with its unique visual design and exciting gameplay mechanics. From the bustling city center to the winding mountain roads, players must navigate stunning landscapes to progress through the game.

To add further excitement, coins are available to collect throughout the game. These coins help players achieve a high score, providing a sense of accomplishment and progression.

Players must remain vigilant, as the police will stop at nothing to catch them. The game requires sharp reflexes and exceptional driving skills to stay ahead of the pursuing law enforcement.

In conclusion, Highway Racers is an exciting and challenging game that offers players an immersive experience of racing through city streets while evading the police. The game's engaging gameplay mechanics, stunning visual design, and high-score mechanics provide players with hours of fun and entertainment.

# Literature Study

# Mindstorm Studios Summer Program’s Playlists

Mindstorm Studios launched its Summer Program in 2022 for teaching different aspects of game development to aspiring developers. We were instructed to watch its playlists on Game Design, Game Programming, and Game Arts, make detailed notes on them, and upload them on GitHub in a single Word document.

* + 1. **Game Design**

Our key findings from the Game Design playlist were that game design has to be understood as a process with all its various steps, needed in designing the 6 most important elements: art, mechanics, progression, balanced economy, monetization, and UI design. We also learned about different types of games based on their mechanics, such as casual, hyper-casual, mid-core, and core, along with the distinctive features of each and their pros and cons.

* + 1. **Game Programming**

Our key findings from the Game Programming Playlist were the fundamentals of gameplay. We were also introduced to Unity and its different aspects.

* + 1. **Game Arts**

Our key findings from the Game Arts playlist were that art is created due to a process, including several steps such as preprocessing, concept art, modeling, texturing, and animation, and that at each step the overall UI had to be gauged on attractiveness. We also learned about different UI trends over time.

# The Art of Game Design: A Book of Lenses

# Mr. Waqar Ahmed assigned us the book “The Art of Game Design” by Jesse Schell to research from and regularly update our findings of each chapter in separate Word files, which we did so on GitHub. For each chapter, we would write half a page on its findings and half a page on how we could specifically implement its findings in our game. There were a total of 30+ chapters in this book.

# Its main idea was that video game design can be compared to the design of any other game in real life, and throughout the book, it regularly referenced real-life examples of games we play to show how the elements such as level of difficulty, are similarly designed. Most importantly, we learned different ‘lenses’ (ways of thinking or viewing the game elements) through which to look at the game design process, and so truly become game developers.

Chapter 1 taught that anyone can become a game designer, and the most important skills or gifts for becoming one do not include talent at all, but rather listening and passion. If the game designer can truly listen to his audience, client, and his team, he will be able to understand them better and make a better game. Moreover, an untalented but passionate game developer will always rise above the talented but bored developer, simply because the former will be willing to go the extra mile and make the game as best as possible.

# Chapter 2 taught us that game experience is derived from how all the elements complement each other. It is not about picking all individually best game elements and using them in the game, because they might not match each other. Thus, we learned to pay close attention to how all the imported assets, their aesthetics, sound effects, and music meshed together. Chapter 3 taught us that a good experience also rests on the game’s difficulty increasing appropriately, on it having a sufficient variety to keep the user intrigued, and on it being fun enough. Chapter 11 taught us different types of game balance to keep in mind.

Other chapters (17, 21, 22) broadened our vision for the game from a simple offline avoid-the-traffic game to a much more expansive multiplayer game with possibilities of guilds, teams, and playing with friends in the same game session. We have not implemented this, but this did give us a grander vision for the game and helped us understand how any game can be made much more interesting despite being complex.

# Software Development Methodology and Plan

## Software Process Selection

Agile processes focus on working deliverables, trying to implement new features in each iteration, whereas waterfall processes focus on the step-by-step implementation of one complete, final deliverable. While agile processes are more flexible than waterfall ones and get the product to market faster, they lack documentation. On the other hand, waterfall processes deliver a complete deliverable once, but are very intolerant of change.

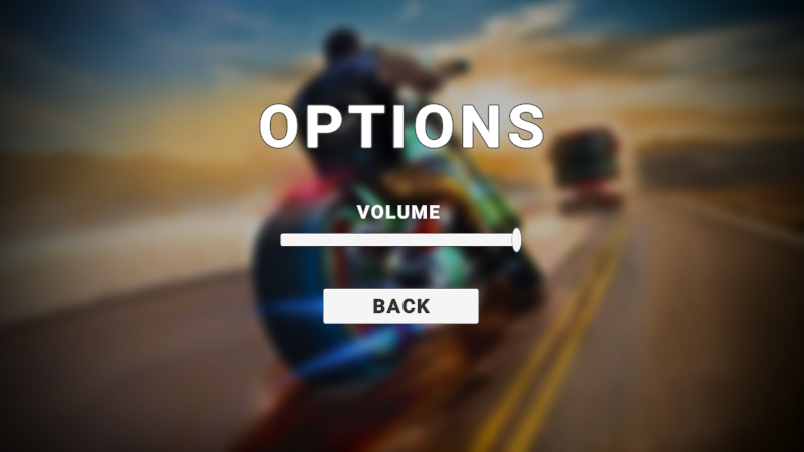
We knew that we would have to constantly change game features or their aesthetics based on the feedback we got from our mentors and our potential customers (videogame players whom we showed the game for feedback). Thus, we considered it better that we implement working iterations of the game to be able to test it and let others judge it as it is being developed. This would also help us discover bugs during the development process, instead of at the end when the whole game would have been made.

# System User Interface

* 1. **Start Screen**



* Upon pressing the Play button, the user is taken to the Cinematic Screen.
* Upon pressing the Quit button, the game exits, and the user is taken back to his mobile home screen.
  1. **Settings Screen**



* Upon pressing the Settings button, the user is taken to the Settings Menu Screen.
  1. **Cinematics Screen**



* 1. **Gameplay Screen**

****

* By pressing the left and right arrow buttons, the user can maneuver the motorbike sideways to avoid obstacles.



* By pressing the pause button, the user is taken to the Pause Screen.
* The fuel left is displayed in the top-right corner.
* The user’s remaining lives are displayed in the form of a health bar on the top.
* In the center of the screen, the user can see the traffic, static obstacles, coin collectibles, fuel collectibles, and health collectibles and maneuver his bike to avoid or go towards these.
  1. **Pause Screen**

****

* 1. **Ending Screen**



* The user’s current score and coins collected are displayed in the top-left corner.

# Testing and Evaluation

As we developed our game, with each iteration we added new features and every member of the group played it to check the bugs and overall experience of the game. After every sprint, we tested our game with our friends who were university students of age 18-23.

The majority of the testers loved the game experience and story associated which was simple to understand and the game was fun to play. The obstacles added excitement to the game while the constant increase in speed added difficulty which was greatly appreciated by users. Many of the testers remarked on the game’s background music track being very aesthetic.

Below are some examples of how we manually tested our game and checked for bugs.

**Collision with Obstacles**

Let the bike run into an obstacle. On collision, that obstacle should disappear, a cloud of dirt should appear, and a collision sound should be played. The health bar should now display the amount of health to be decreased by 25% of the total amount.

**Colliding with Fuel Collectible**

Let the bike run into a fuel collectible. On collision, the fuel collectible should disappear and the fuel of the bike should increase by 15 units.

**Skipping Cinematic**

Press the Run Button on the Cinematic Screen. This should lead to the next screen loading immediately, without waiting for the cinematic sequence to finish.

**Stopping Bike when Runs out of Fuel**

When the fuel of the bike becomes 0 units, the bike should stop. The whole game should become paused, and a menu should pop up with buttons for buying fuel or quitting the game.

We did not use any automation tools.

# Deployment Guidelines

* Firstly, in the Final Deliverables folder of our group’s (P01-HighwayRacers) GitHub repo, go to the code subfolder and from there copy the google drive link from the text file for the APK.
* Next, paste the drive link in a browser and then download the APK file when the option for download appears after the page loads.
* To be able to install the game and play, the host device should be a smartphone and the operating system should be Android version 6 or higher. Now, click on the APK file once it has been downloaded and the device will ask you to install it. Click on Yes and install the APK.
* After the APK has finished installing, it will be available in your phone’s apps screen and you can click on the game icon to get started with the game.
* Alternatively, you can access the APK through this link:

<https://drive.google.com/file/d/1CVPYZa3zDR5ESobBEBV3aarUNJGAZ3ey/view?usp=sharing>

# Lessons Learnt

For our project, we had representatives of Mindstorm Studios as our mentors, who guided us throughout the game development process. We have learned several lessons from them.

Our first lesson was to appreciate game development truly as a process and realize the extent to which each non-development phase needed work. Previously, we considered game development to be mainly idea generation and development only, but our mentors had us go through an extensive ideation process in which we realized that ideation is half the work in development. The game will never be good enough if the ideas are not good enough. For this, we played several games of the genre we were interested in – racing, endless runners, traffic-avoiding, and stunt games. We brainstormed ourselves and also contacted several gamers to know what their thoughts about the ideal game were.

Secondly, we learned that nothing is the ‘right’ implementation or aesthetic. Some are better than others objectively, but in the end, it is all about the game producing the ‘right effect’ on the player, which depends on how the entire game’s elements combine. Thus, it is mainly trial-and-error until your game seems to be very aesthetic and pleasing to play.

Thirdly, we realized the significance of the game story at the end of our third sprint. Our mentors reviewed the game and were satisfied with the features implemented but they made us aware that the game was not telling its game story to the user. Each user needs to know why he is doing what the game is making him do in the game so that he can really live the game and experience it fully. We then made a cinematic that played before the game starts, so that users would know that our rider was speeding on a highway and is trying to evade a police car.

Lastly, we learned that GitHub Desktop is not suited at all for game development and that SourceTree is a much better option. The former could not facilitate the resolution of merge conflicts that were Unity-related (different game elements in-game scenes etc.) and it also did not allow us to visually view the work on different branches. With SourceTree, we could now merge conflicts more simply as it has a more user-friendly interface. It also displayed different branches’ work as differently-colored lines along a path, which would converge to show merging and diverge to show the creation of a new branch.

# Conclusion

## Summary

Although initially, all group members were new to the game development, we were focused on working on something unique and different as a final-year project. We wanted to develop a game that is engaging as well as enjoyable for the users at the same time.

Overall, we have learned a great deal about game design and development throughout the process. This project has taught us valuable skills that apply in future endeavors and lengthened our passion for game development.

## Challenges

During this project, we faced several technical and non-technical challenges that required us to be flexible, creative, and persistent to overcome them. One of the main technical challenges we encountered was the placement of the rider over the bike. We did not know how to use some software, such as Blender – a free 3D computer graphics software, that was needed in character animations and character’s arms bending. To address this, we relied on various tutorials on how to achieve this using Blender.

Another technical challenge we faced was merging our code on GitHub. We faced conflict issues multiple times while merging code and several other issues as well, such as “git LFS” and large-sized files, etc. We asked Mindstorm’s technical person regarding this issue and he suggested we use SourceTree for merging the code and that solved our various problems.

In addition to these technical challenges, we also encountered several non-technical issues, such as communication and scheduling conflicts among team members, and unexpected delays due to external factors such as changes in project requirements or unforeseen technical issues. To address these challenges, we set realistic deadlines and milestones and remained flexible and adaptable in the face of unexpected challenges.

## Future

The mobile game we have developed is designed to be easily extendable, allowing for future development and expansion by either ourselves or other developers. One potential area for extension is the addition of new levels, characters, and gameplay mechanics, which could be accomplished by building upon the existing codebase and expanding the game's asset library.

Another area for extension is the integration of additional features, such as multiplayer support. These features would require additional time and effort, but could significantly enhance the game's functionality and user experience.

# Review checklist

|  |  |
| --- | --- |
| **Chapter/Section Name** | **Reviewer** |
| 1 | Amaan Tariq, Abdullah Ahmad |
| 2 | Abeeha Ishfaq, Muhammad Hamza, Abdullah Ahmad |
| 3 | Bakhtiar Rasheed |
| 4 | Bakhtiar Rasheed, Abdullah Ahmad |
| 5 | Abdullah Ahmad, Bakhtiar Rasheed |
| 6 | Abdullah Ahmad, Bakhtiar Rasheed |
| 7 | Bakhtiar Rasheed, Abdullah Ahmad |
| 8 | Abdullah Ahmad |
| 9 | Bakhtiar Rasheed |
| 10 | Muhammad Hamza |
| 11 | Abdullah Ahmad, Bakhtiar Rasheed |
| 12 | Bakhtiar Rasheed |

# References

* Official Unity Documentation: <https://docs.unity3d.com/Manual/UnityOverview.html>
* Official Unity Website: <https://unity.com/>