eA picture containing text, sign

Description automatically generated

College of Computing

Computer Science Department

CS3141 Team Software Project

Spring 2022

**Project Knievel**

Section: \_R02\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team #: \_11\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| Roll # | Name | Role |
| --- | --- | --- |
| 10 | Ben Fosdick | Developer |
| 9 | Logan Eerdmans | Scrum Master |
| 11 | Preston Foster | Developer |
| 12 | Kelby Gingerich | Developer |
| 14 | Ymailda Hernandez | Developer |

Instructor:

Serein AL-Ratrout

# **Abstract**

(***Abstract*** *is one-paragraph summarizes your project, describes the content and scope of the project objective, methodology, findings, and conclusion. So, you need to write one-paragraph that gives an abstract idea about the entire project, the aim of the project, the process model you used, the tools, what you have done, the results, and your conclusion. If you think the project is worth extending to a Final Year Project (FYP) by you or other students or can be adopted and extended by industry/market, then mention that here and add it also as future work.)*

Example of abstract

In this project a student registration web application for university students and staff was developed, the aim of this application is to provide a simple set-up of programs for student enrolment, improve efficiencies and eliminate unnecessary paperwork. The system mainly has two modules: students and staffs. Students can create account then sign in/out, add, update, delete, and modify their data and schedule. Staff can also create account and then sign in, add, update, delete, and modify their data. Waterfall process model followed during project development and the implementation was realized by use of object-oriented PHP, HTML, MYSQL and Dreamweaver technologies.

It has been found that the final system was simple and user friendly with easy user interface, hence the end-users do not need to undergo extensive training or require any special skills. It was also secure, and reliable.

In this project a trick generation application for skateboarders, bikers, and scooterers was developed, the aim of this application is to provide a simple and approachable means of learning new tricks, improving the user’s skills, and providing a new fun way to ride. The application essentially has two key components, the trick lists for each respective vehicle and the trick generator. Users can pick a vehicle, pick the level of difficulty they wish to take on, then generate a trick for that specific vehicle. Once a given trick has been generated, the user can then click or tap on the trick to view more information about it. The agile process model followed during project development and the implementation was realised by use of Flutter and Dart.

In testing, our final application was found to be simple and user friendly. The interfaces are clean and easily navigable and the minimalistic design significantly simplifies the work any given user has to do in order to use the application. Our design minimizes the amount of clicks a user needs to do between steps in order to save time, reduce frustrations, and increase ease of use. It was also secure and consistently found to be reliable.

# **Table of Contents**

[Abstract](#_heading=h.pjgzsug152eo)

[Table of Contents](#_heading=h.8e420o9x1mn2)

[Table of Figures](#_heading=h.2g6o4livdfk8)

[1.1 Introduction](#_heading=h.30j0zll)

[1.2 Problem Statement](#_heading=h.nrmbr1w467mk)

[1.3 Aim and Objectives](#_heading=h.3znysh7)

[1.4 Stakeholders  
 The primary audience are Scooter riders, Skateboarders, and BMX riders. As well as the development team and managers.](#_heading=h.62yx7sku97cy)

[1.5 Methodology](#_heading=h.8msuvglsfzzh)

[1.6 Tools](#_heading=h.jx4b8j5l64ml)

[1.7 High-Level Business Requirements](#_heading=h.6vyreuy8panq)

[1.7.1 Functional Requirements](#_heading=h.clft5ekj4v6u)

[1.7.2 Non-functional requirements](#_heading=h.un65sqtxwugx)

[1.8 Product backlog](#_heading=h.6xvbp2gml5xn)

[2.1 Use Case Diagram  
 A diagram was constructed to help visualise as well as organise the basic standard setup used for the application. This was made using PlantUML, a free, online tool made for the design creation of use case diagrams. The diagram also shows fundamental connections between the user and related actions to help represent the structured flow that the stakeholders can expect to follow. There used to be arrows connecting the actor to the use cases, but they have been switched to lines. 1](#_heading=h.s9vwomvytogx)

[2.2 Class Diagram](#_heading=h.c7g2yzsx72kb)

[Similar to the use case diagram, the class diagram has been made using Plant UML. The Class diagram shows how the classes are involved with each other. Each class shows the variables used in the class and what type of variable each of them are. The classes also show the methods and the return types for each class. 1](#_heading=h.s2ukrwc5hidj)

[2.3 Sequence Diagram](#_heading=h.9trzpeonwiov)

[Plant UML has also been used to create the sequence diagram. The sequence diagram shows how the objects in the application interact with each other. The Rider is the actor and also the user. Maps on the right side is google maps on the phone. 1](#_heading=h.mk07tij4lqug)

[3.1 Introduction](#_heading=h.e2rqido9gqhf)

[3.2 Graphic User Interface](#_heading=h.5gh3c3qj1o9z)

[3.3 Class Implementation](#_heading=h.8zroji4n7hcs)

[3.3.1 Trick Class Implementation](#_heading=h.f16ls2rtvke1)

[Validation](#_heading=h.tyjcwt)

[Limitations and Future Work](#_heading=h.35z0rgsgkho6)

[References](#_heading=h.nco6v06p84nj)

# **Table of Figures**

Chapter 1

Specification

# **1.1 Introduction**

Project Knievel is a mobile application that generates a trick or series of tricks for scooter, bike, and skateboard riders. It will have a collection of tricks that the user can add tricks to that are not already in the collection, remove tricks, and edit the name and/or description of the tricks. This application can be used by riders at parks or in street locations to generate a trick in order to help the rider improve their skillset and learn tricks that are out of their current trick vocabulary.

# **1.2 Problem Statement**

Riders often have a difficult time coming up with tricks to learn. This primarily occurs when they do not have people to ride with who are better than them or know different tricks. Other times, riders get stuck in a rut of only doing a few tricks that they already know and do not learn new tricks in part because they don’t know what to try. Alternatively, when riding in groups that have significantly different skill levels, it can be difficult to enjoy games like “S.C.O.O.T.” because the person(s) that are better, choose tricks that are well out of the less experienced player’s skill level.

# **1.3 Aim and Objectives**

The aim of this project is to develop a cross platform mobile application to help Scooter, Skateboard, and BMX riders expand their trick vocabulary, learn about skatepark features, and create an attainable yet competitive riding experience with fellow riders.   
**Objectives:**

* To allow riders to generate tricks to perform…
* To display information about skatepark features…
* To educate riders on what tricks can be performed and how to perform them…
* To measure trick vocabulary expanding progress…
* To encourage a healthy, competitive riding experience with friends…

# **1.4 Stakeholders** The primary audience are Scooter riders, Skateboarders, and BMX riders. As well as the development team and managers.

# **1.5 Methodology**

We used the Agile methodology called Scrum which follows the process below:

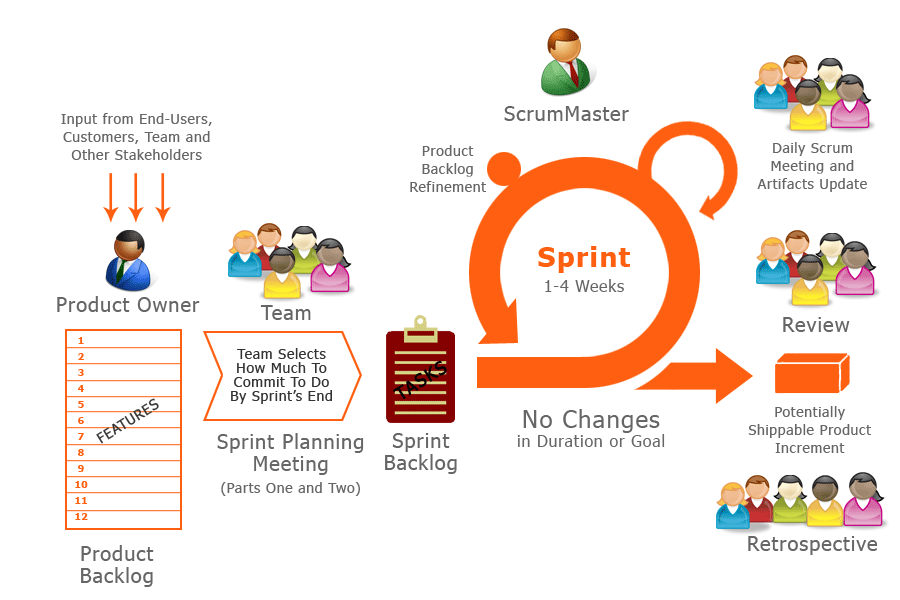


Fig 1.1 Diagram of Scrum process

**Product Backlog:** A list of prioritised feature requirements containing all the user stories for the product that the product owner assigns and drafts.

**Sprint Planning Meeting:** A planning meeting at the beginning of a sprint to revise the product backlog (Team effort) to select condensed goals of the user stories to put in the sprint backlog.

**Sprint Backlog:** The features that the team and product owner have agreed upon to implement during a given sprint.

**Sprint Daily Meetings:** Daily update meetings that should last no more than a few minutes to keep the team on track. It includes three main questions for each developer and asked by the scrum master: What have you gotten done since the last meeting? What am I working on next? What are my blocker/obstacles (if any)?

**Sprint:** This is an allocated time of 1-4 weeks of work and development of the tasks in the product backlog. Each sprint ends when the predetermined sprint timebox ends.

**Retrospective:** The team reflects on the work that was completed in the last sprint and what can/should be improved on for the next sprint. Solutions can be presented to be executed in the next sprint. There are two primary questions asked: What went well in the last sprint? What could be improved in how we worked in the last sprint?

**Three roles:**

**Product owner:** The person who defines the feature requirements for the product.

**Scrum master:** The developer who is responsible for facilitating the scrum process.

**Team:** The developers that are allocated full-time to a sprint.

# **1.6 Tools**

We will use Git and Github for version control and collaboration and Google Flutter SDK to develop the cross platform application. Dart is the primary programming language we will use.

# **1.7 High-Level Business Requirements**

# **1.7.1 Functional Requirements**

* The user should be able to indicate whether they ride Scooter, Skateboard, or BMX
* The user should be able to generate a random trick to perform on what they ride
* The user should be able to add, delete, and edit tricks names and descriptions on the list of tricks
* The user should be able to check whether they know a trick or not on the trick list
* The user should be able to check whether only tricks they know are generated
* The user should be able to press a link to YouTube on how to perform a given trick
* The user should be able to see a stream of images or a GIF of the trick being performed
* The user should be able to see pictures and descriptions of different skatepark features. (Down rail, quarter pipe, mini ramp).
* The user should be able to select a rideable feature(s) and only have tricks for that/those feature(s) generated(Ie. “doing a rail sesh.” or “mini ramp sesh.” or to check only features that are present at a given skatepark).
* The user should be able to see their progress as they learn more tricks and move up in the difficulty levels.
* The trick list should be organised/displayed into different difficulty levels.

## **1.7.2 Non-functional requirements**

* Portability: Must work on mobile devices. Must not exceed a reasonable amount of space for a mobile application.
* Compatibility: Must work cross platform on IOS and Android OS.
* Performance: Trick generation should take less than 0.25 seconds. Pages must load within 2 seconds.
* Core functions should be available offline
* Scalability
* Usability: Users should be able to use the application with minimal training

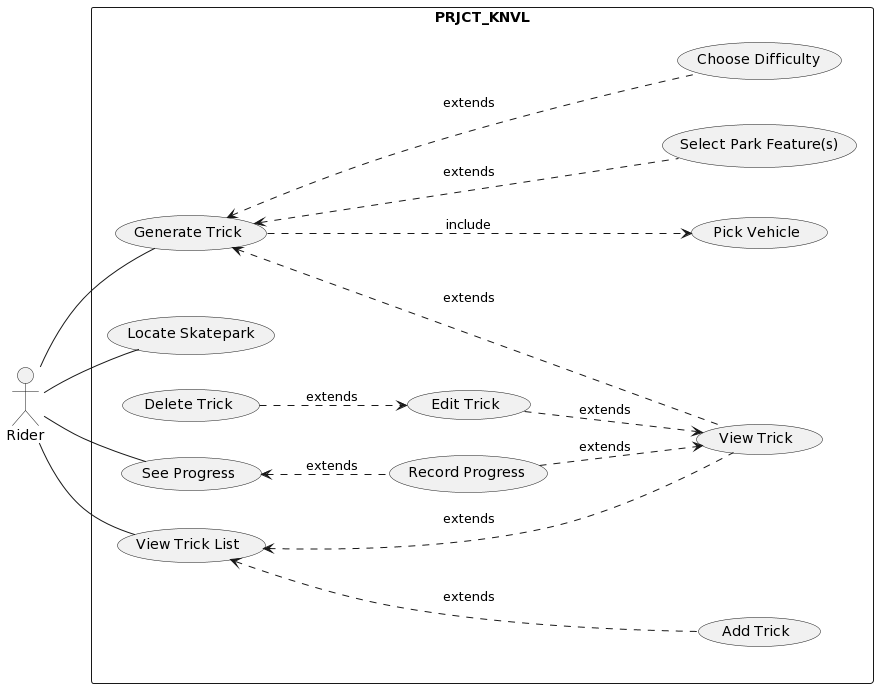
# **1.8 Product backlog**

| **Priority** | **User Story** | **Tasks** | **Estimated Effort** | **Sprint** |
| --- | --- | --- | --- | --- |
| **1** | **As a Rider, I want to select what vehicle I ride, so that I can generate tricks that can be done on my vehicle.** | **Design and implement the interface** | **2 H** | **1** |
| **Organise the trick lists for the different vehicles** | **1 H** |
| **1** | **As a Rider, I want to generate random tricks depending on what I ride.** | **Design and implement interface** | **2 H** | **1** |
| **Implement a trick list** | **2 H** |
| **Implement trick generation from the list** | **1 H** |
| **2** | **As a Rider, I want to add, delete, and edit trick names and descriptions on the trick list.** | **Design and implement interface to display tricks** | **2H** | **2** |
| **Implement add functionality** | **1H** |
| **Implement delete functionality** | **1H** |
| **Implement edit functionality** | **1H** |
| **2** | **As a Rider, I want to check if I know/understand a trick on the trick list.** | **Design and implement interface to display a trick description** | **2H** | **2** |
| **2** | **As a Rider, I want to only generate tricks that I know/understand.** | **Implement a toggle associated with each trick that indicates whether it will be generated** | **2H** | **2** |
| **2** | **As a Rider, I want to be able to watch a YouTube video on how to perform a trick on the trick list.** | **Design and implement an interface to display a link to a matching video.** | **1H** | **3** |
| **2** | **As a Rider, I want to visualize the performance of a trick via a set of images or GIF.** | **Design and implement an interface to display the unique images of each trick** | **2H** | **3** |
| **2** | **As a rider, I want to input that I have learned or completed a trick and am ready to get generated a new one** | **Design and implement functionality to mark tricks as complete and generate a new one.** | **2H** | **3** |
| **3** | **As a Rider, I want to see pictures and descriptions of different features that a skatepark provides and only generate tricks based on what features the park provides.** | **Design and implement an interface to display info about a given skatepark.** | **4H** | **4** |
| **Implement functionality to make notes about a certain skatepark.** | **2h** |
| **Implement functionality to display a helpful chart about a given skatepark’s features.** | **4h** |
| **Design and implement an interface to generate tricks based on skate park features.** | **3H** |
|
|
| **3** | **As a Rider, I want to record my progress and increase the difficulty level of performing tricks and only generate tricks based on difficulty level.** | **Design and implement an interface to record user progress.** | **2H** | **4** |
| **Implement a difficulty based trick generation system.** |
|

Chapter 2

Analysis and Design

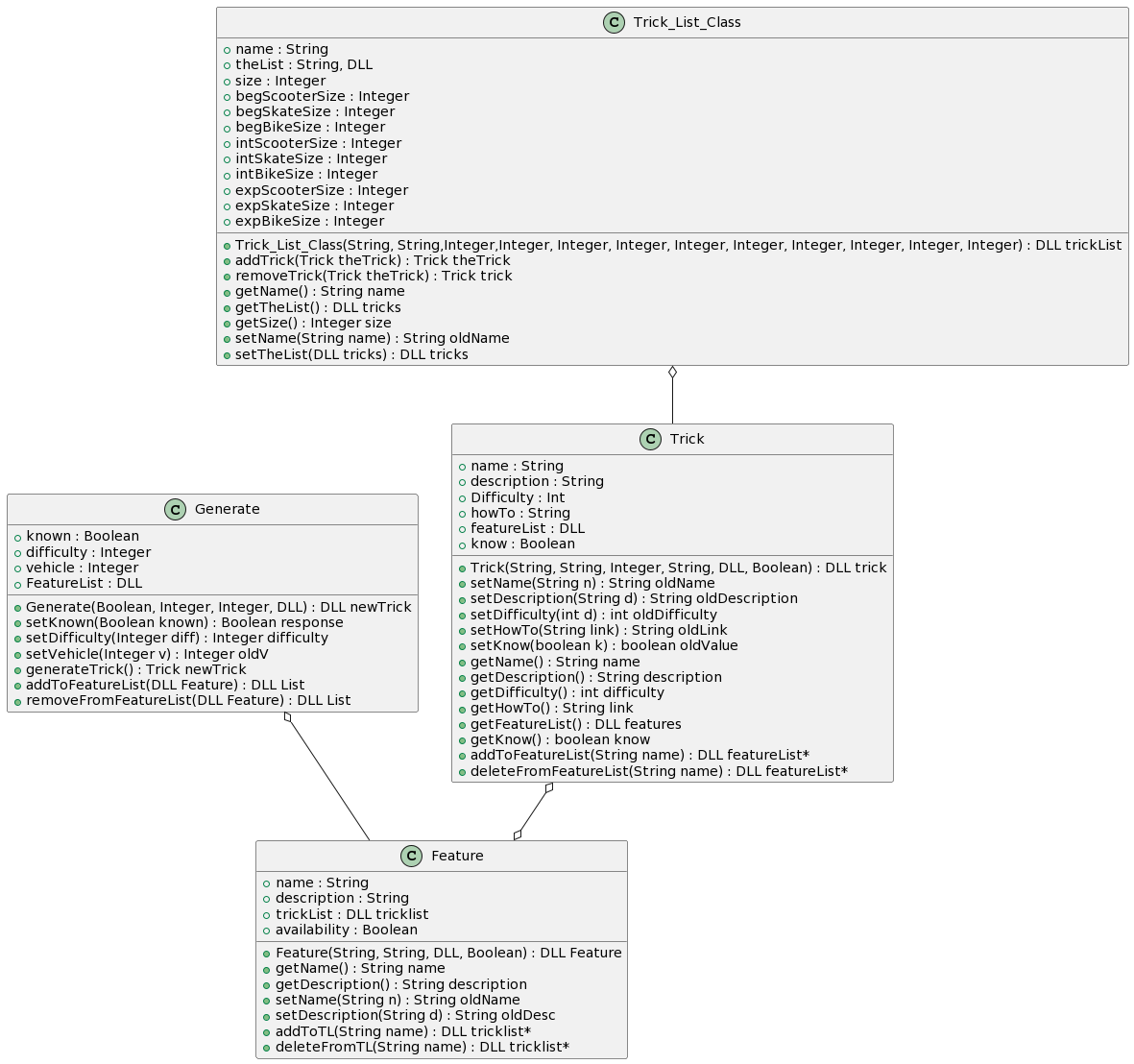
# **2.1 Use Case Diagram** A diagram was constructed to help visualise as well as organise the basic standard setup used for the application. This was made using PlantUML, a free, online tool made for the design creation of use case diagrams. The diagram also shows fundamental connections between the user and related actions to help represent the structured flow that the stakeholders can expect to follow. There used to be arrows connecting the actor to the use cases, but they have been switched to lines. 1

****Fig 2. Use Case Diagram of Project Knievel

# **2.2 Class Diagram**

# Similar to the use case diagram, the class diagram has been made using Plant UML. The Class diagram shows how the classes are involved with each other. Each class shows the variables used in the class and what type of variable each of them are. The classes also show the methods and the return types for each class. 1

A change that has been made to the diagram this sprint includes removing the vehicle field and all of the corresponding methods in the trick class.



# **2.3 Sequence Diagram**

# Plant UML has also been used to create the sequence diagram. The sequence diagram shows how the objects in the application interact with each other. The Rider is the actor and also the user. Maps on the right side is google maps on the phone. 1

# 

Chapter 3

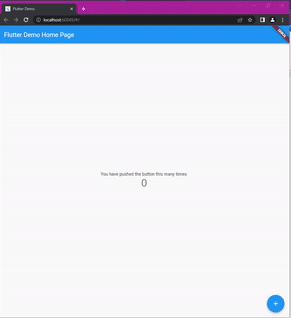
Implementation

The following [report](http://people.uncw.edu/simmondsd/documents/450_Implementation%20and%20Testing%20Report.pdf) is a good example that you can follow for implementation please refer to pages 25 - 30, and here is another [example](http://api.uofk.edu:8080/api/core/bitstreams/13308397-e07c-47ef-83e5-3bbb2e9f0a81/content) for your reference.

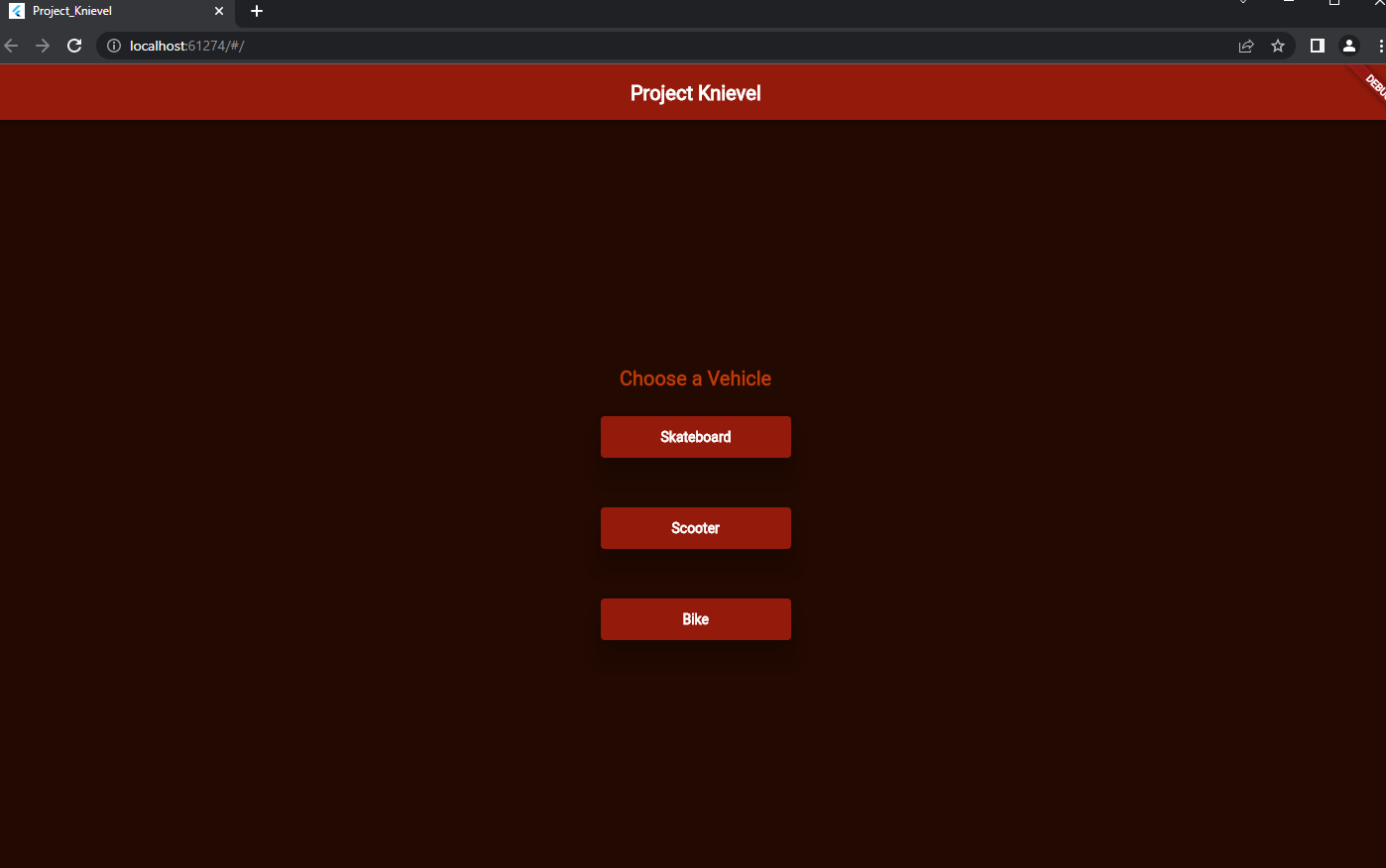
# **3.1 Introduction**

For There are two main sections to this implementation chapter. The first is the implementation of the graphic user interface and the second is the implementation of the backend classes.

# **3.2 Graphic User Interface**

For implementation thus far, we have a simple button which is linked to a counter variable. This very simple project is what we are aiming to build off of as we continue to further develop our project. Below is a gif of the very simple beginnings of our project and a higher resolution picture of it as well.

Previous GUI Design



Current GUI design

# **3.3 Class Implementation**

The implementation of classes follows the class diagram design of each class. Reference those for specific details on the class’s design.

# **3.3.1 Trick Class Implementation**

The Trick class has been mostly implemented with all the fields except the list of Features that the trick can be done on.6 It also has all its setter and getter methods fully implemented. The trick class is a non-static class that defines a Trick object. Each trick object holds the information necessary to tell if the user knows the trick or not, the difficulty of the trick, and a link to helpful resources to learn the trick, among other things. There are no additional libraries or classes currently used in the trick class.

**3.3.2 Trick List Class Implementation**

The Trick List class has been mostly implemented with all the fields except the list of Features that the trick can be done on. It also has all its setter and getter methods fully implemented. This list will hold all of the created tricks in the Trick class.

Chapter 4

## Validation

**4.1 Introduction**

There are two main sections in this chapter. In the first section the automated unit testing of the project’s classes is covered. In the second section, the manual GUI testing is covered.

**4.2 Unit Testing**

We implemented unit tests for all our classes using the Dart Test package.8 Each method of each class is tested at least once. These tests were each run after any changes were made to any of the classes.

Trick Class:

| **Method** | **Expected** | **Actual** | **completed by** |
| --- | --- | --- | --- |
| setName() | Tailwhip | Tailwhip | Ben Fosdick |
| setDescription() | The deck spins around 360 | The deck spins around 360 | Ben Fosdick |
| setDifficulty() | 2 | 2 | Ben Fosdick |
| setHowTo() | theLink.com | theLink.com | Ben Fosdick |
| setKnow() | false | false | Ben Fosdick |
| setVehicle() | 1 | 1 | Ben Fosdick |
| getName() | Tailwhip | Tailwhip | Ben Fosdick |
| getDescription() | The deck spins around 360 | The deck spins around 360 | Ben Fosdick |
| getDifficulty() | 2 | 2 | Ben Fosdick |
| getHowTo() | theLink.com | theLink.com | Ben Fosdick |
| getKnow() | false | false | Ben Fosdick |
| getVehicle() | 1 | 1 | Ben Fosdick |

Trick List Class

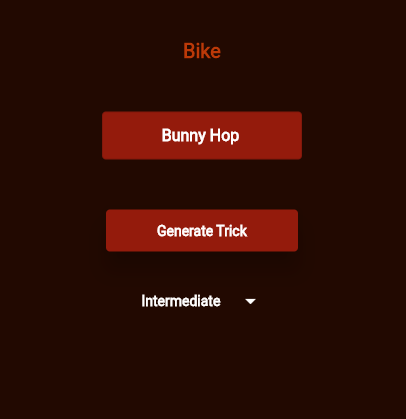
| **Method** | **Expected** | **Actual** | **Completed by** |
| --- | --- | --- | --- |
| TrickList.setName() | ‘Scooter List’ | ‘Scooter List’ | Mel Hernandez |
| TrickList.getName() | ‘Scooter List’ | ‘Scooter List’ | Mel Hernandez |
| TrickList.getSize() | 2 | 2 | Mel Hernandez |
| TrickList.addTrick() | c | c | Mel Hernandez |
| TrickList.removeTrick() | c | c | Mel Hernandez |

Other methods implemented

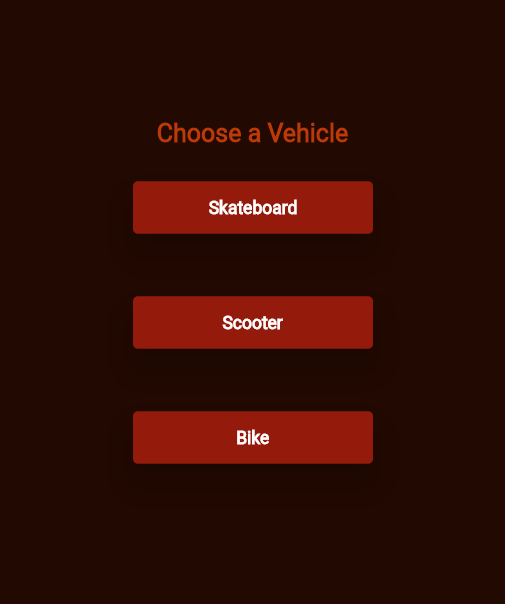
| **Method** | **Expected** | **Actual** |
| --- | --- | --- |
| Scooter Vehicle Button | Taken to Scooter Trick Generator | Taken to the Scooter Trick Generator |
| Scooter Trick generation | A scooter trick is generated | The scooter trick was generated |
| Skateboard Vehicle Button | Taken to Skateboard Trick Generator | Taken to the Skateboard Trick Generator |
| Skateboard Trick generation | A skateboard trick is generated | The skateboard trick was generated |
| Bike Vehicle Button | Taken to Bike Trick Generator | Taken to the Bike Trick Generator |
| Bike Trick generation | A bike trick is generated | The bike trick was generated |

**4.3 Manual GUI Testing**

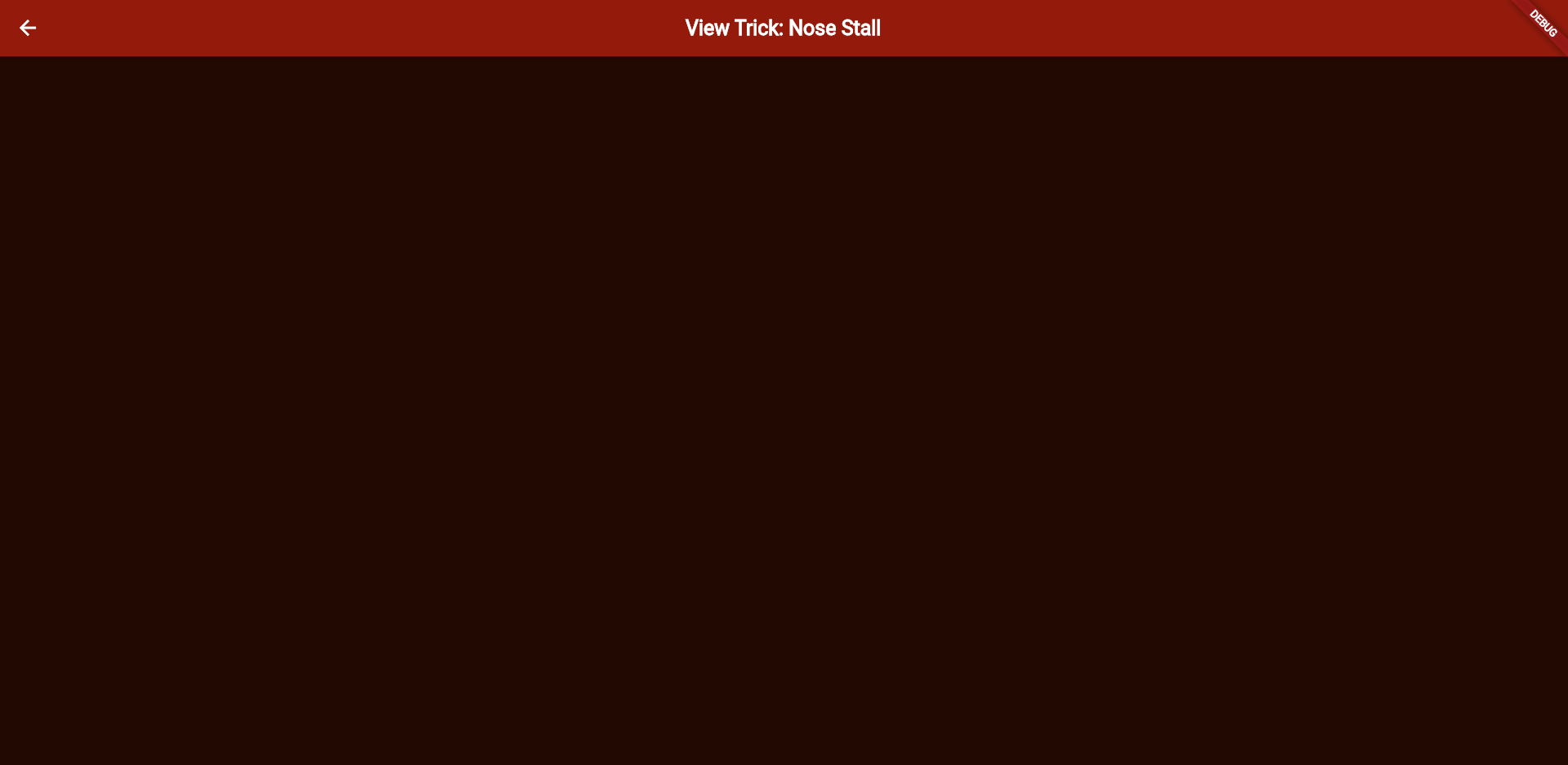
**Bike vehicle Button test**

****

**Buttons for 3 different vehicles**

****

**Trick Description Page Testing**

****

Chapter 5

## Limitations and Future Work

**For Chapter 5**

Limitations: address everything that the project left,  if some project backlog items/ features/ requirements have not been implemented then mention them in this part with an explanation/justification why you couldn't implement them (Time constraints the time was not enough, some developers were unavailable, because of COVID19, or  tool limitation ....etc.). Many students tend to feel that presenting the limits of their work makes work weaker. on the contrary, approaching this section shows maturity for the academic universe, and writing about them actually strengthens your work by identifying any problems before reviewers/readers find them.

Future work : if the limitations can be addressed in the future then add this in here in future work, moreover, if you believe this project can be extendable (add more features/ more parts) that the project is worth extending to a Final Year Project (FYP) by you or other students or can be adopted and extended by industry as a product so you can give directions for that in future work.

Chapter 6

Conclusion

**For  Chapter 6 (Conclusion),**: write what you have concluded.

Examples:

I solved many problems in the project…

This application/project/system was applied to improve the learning process.

The results of this project showed that system significantly facilitated the students' learning process.

The system is useless, acceptable, usable, beneficial or maybe enjoyable and why do you believe that.

References

(Include any references to external documents or materials (for example, tutorials the team will be using, literature , web references or links to documentation of third-party tools you will use) here.

The references should be properly numbered and correctly used in the text.

The Reference section should be in the following fashion:

# References

[ ] Serein Alratrout," introduction to scrum" ,lecture notes, access date: September/2022

[1] PlantUML.com, “PlantUML in a nutshell”, documentation, access date: October 10, 2022.

[2] PlantUML.com, “Use case Diagram syntax and features”, documentation, access date: October 10, 2022.

[3] Ashraf Otagun, “How to Create a New Flutter project on GitHub in 5 Simple steps”, article, access date: October 11, 2022.

[4] Codes Easy, “Setup Flutter and Visual Studio Code In Windows (2022 Working)”, video, access date: October 13, 2022.

[5]: docs.flutter.dev, “Navigate to a new screen and back”, documentation, access date: October 28, 2022.

[6]: Academind, “Flutter Crash Course for Beginners 2021 - Build a Flutter App with Google's Flutter & Dart”, video, access date: October 28, 2022.

[7]: docs.flutter.dev, “An introduction to unit testing”, documentation, access date: October 28, 2022.

[8]: dart.dev, “test | Dart Package”, documentation, access date: October 28, 2022.

[9] Taio, “How to create a button bar for displaying a row of buttons?”, webforum, access date: October 28, 2022.

[10] Naresh Pradeep, “Flutter ButtonBar Example Tutorial”, article, access date: October 28, 2022.

[11] api.flutter.dev, “DropdownButton<T> class”, documentation, access date: October 31, 2022.

[12] jeroen-meijer, “How do I center text vertically and horizontally in Flutter?”, webforum, access date: October 31, 2022.

[13] Rajdeep, “How to center text in DropdownButton?”, webforum, access date: October 31, 2022.

[14]: api.flutter.dev, “List class - dart:core library - Dart API”, documentation, access date: November 3, 2022.

[15] Daniel, “How do you change the value inside of a textfield flutter?”, webforum, access date: November 11, 2022.

Journal, Magazine/ Newspaper Article

| [1] | Author, "Title," *Journal name,* p. pages, year.  Book |
| --- | --- |
| [2] | Author, Book Title, publisher, year.  Internet Web page: |
| [3] | Author, "Name of the Web Page," [Online]. Available: URL. [Accessed Date]. |

| [1] | PlantUML.com, “PlantUML in a nutshell”, [Online]. Available: <https://plantuml.com/> [October 10, 2022] |
| --- | --- |
| [2] | PlantUML.com, “Use case Diagram syntax and features”, [Online]. Available: <https://plantuml.com/use-case-diagram> [October 10, 2022] |
| [3] | Ashraf Otagun, “How to Create a New Flutter project on GitHub in 5 Simple steps” [Online]. Available: <http://bit.ly/3URBcu5> [October 11, 2022] |
| [4] | Codes Easy, “Setup Flutter and Visual Studio Code In Windows (2022 Working)”, [Online]. Available: <https://youtu.be/5izFFbdHnWY> [October 13, 2022] |
| [5] | docs.flutter.dev, “Navigate to a new screen and back”, [Online]. Available: <http://bit.ly/3hD2vtq> [October 28, 2022] |
| [6] | Academind, “Flutter Crash Course for Beginners 2021 - Build a Flutter App with Google's Flutter & Dart”, [Online]. Available: <https://youtu.be/x0uinJvhNxI> [October 28, 2022] |
| [7] | docs.flutter.dev, “An introduction to unit testing”, [Online]. Available: <https://docs.flutter.dev/cookbook/testing/unit/introduction> [October 28, 2022] |
| [8] | dart.dev, “test | Dart Package”, [Online]. Available: <https://pub.dev/packages/test#writing-tests> [October 28, 2022] |
| [9] | Taio, “How to create a button bar for displaying a row of buttons?”, [Online]. Available: <http://bit.ly/3g8WHYx> [October 28, 2022] |
| [10] | Naresh Pradeep, “Flutter ButtonBar Example Tutorial”, [Online]. Available: <https://codesinsider.com/flutter-buttonbar-example/> [October 28, 2022.] |
| [11] | api.flutter.dev, “DropdownButton<T> class”, [Online]. Available: <https://api.flutter.dev/flutter/material/DropdownButton-class.html> [October 31, 2022] |
| [12] | jeroen-meijer, “How do I center text vertically and horizontally in Flutter?”, [Online]. Available: <http://bit.ly/3TKM6AC> [October 31, 2022] |
| [13] | Rajdeep, “How to center text in DropdownButton?”, [Online]. Available: <http://bit.ly/3EaHKgx> [October 31, 2022.] |
| [14] | api.flutter.dev, “List class - dart:core library - Dart API”, [Online]. Available: <https://api.flutter.dev/flutter/dart-core/List-class.html> [November 3, 2022] |
| [15] | Daniel, “How do you change the value inside of a textfield flutter?”, [Online]. Available: <http://bit.ly/3hPiSU9> [November 11, 2022] |