



National College of Ireland

BSc in Computing

Software Developer

2020/2021

Antonio Penna

X01417383

X01417383@student.ncirl.ie

Bike-Ability

Technical Report



BIKE-ABILITY

ABSTRACT

With an application such as Bike-Ability, that guides bike renters to the correct station, so that no time is lost in reaching an unavailable bike station, there would be one less reason for lateness to any destination, the job probably the most important of them all. Such an application would save many commuters much time, money and misery in the long run; After all the adage that time is money that cannot buy lost time back rings especially true when it becomes the motto of the application under development that this report is about.

Antonio Penna

BSc in Computing

GLOSSARY OF ACRONYMS ABBREVIATIONS

Abbreviations	Augmentation
App	Application
BRS	Bike Rental Service
CRUD	Create, read, update and delete
FB	Firebase
FR/S	Functional Requirements/Specifications
GUI	Graphic User Interface
NFR/S	Non-Functional Requirements/Specifications
OS	Operating System
OSCVL	Open-Source Computer Vision Library
PDA	Problem/Domain Analysis
POI	Point of Interest
ROI	Region of Interest
RPB-Pi	Raspberry Pi
SLA	Service Level Agreement
TPU	Tensor Processing Unit
TDD	Test Driven development
UI	User Interface
XML	XML extensible Mark-up Language

CONTENTS

Glossary of Acronyms Abbreviations	2
EXECUTIVE SUMMARY	1
INTRODUCTION.....	2
Background	2
The App.....	3
Aims.....	4
TECHNOLOGY.....	5
HARDWARE.....	5
Camera.....	5
Raspberry PI (RPB-Pi)	5
Android OS Mobile Phones	5
Memory/ Ram on the device	5
Google USB Accelerator.....	6
SOFTWARE.....	7
Android OS.....	7
Android Studio Application	7
Raspberry PI (RPB-Pi) OS (Formerly Rasparian)	7
Firebase Database (FB)	7
Open CV.....	8
TensorFlow Lite	8
Balena Etcher software.....	9
Rational Rose for use case Diagram.....	9
Balsamiq Mockups 3.....	9
J-unit.....	9
PROGRAMMING LANGUAGES	10
Java for Android.....	10
Python.....	10
STRUCTURE	11
SYSTEM	12
System Use Case Diagram	13

USE-CASE DIAGRAMS	14
UML Reserve Bike.....	14
Flow Description	14
UML Open Map	16
Flow Description	16
UML Open Chat.....	17
Flow Description	18
UML Open Dashboard	19
Flow Description	20
UML Manage Inventory.....	22
Flow Description	22
UML Change Language	24
Flow Description	24
FUNCTIONAL REQUIREMENTS SPECIFICATION (frs).....	26
Functional Requirements	26
Login and registration Requirement (Priority: High).....	27
Available Bike Detection (Priority: High)	27
Reserve Available Bike (Priority: High).....	27
Cancel reservation (Priority: High)	27
Availability Notification (Priority: High).....	28
Reserve a Bike (Priority: High).....	28
Rental Statement (Priority: High).....	28
NON-FUNCTIONAL REQUIREMENTS	29
Usability Requirements	29
Data Requirements	29
User Requirements	29
Reliability Requirement (Priority: High)	30
Performance/Response time requirement (Priority: High).....	30

Security requirement (Priority: High)	31
Protection (Priority: High).....	31
Recovery requirement (Priority: High)	31
Availability Requirement (Priority: High)	32
Recovery Requirement (Priority: Medium)	32
Maintainability Requirement (Priority: Medium)	32
DESIGN & ARCHITECTURE	33
Graphic User Interface.....	34
UX Wireframes	34
Bike-Ability Screen-Shots.....	37
IMPLEMENTATION	43
TESTING	51
Test Driven Development, Test Cases.....	51
Create Account.....	52
Login	53
Administration Inventory Management	54
Unit Testing.....	55
Integration Testing.....	58
UI Testing with Espresso	59
EVALUATION	63
FUTURE WORK	66
CONCLUSION	66
APPENDICES.....	67
National College of Ireland	77
Declaration Of Ethics Consideration.....	77
School of Computing.....	77
Please circle (or highlight) as appropriate	77
Introduction	78
Sources of Data:.....	78
Checklist.....	79
Ethics Clearance Guidelines When Human Participants Are Involved.....	80
Considerations in data collection.....	80
Informed consent	80

EXECUTIVE SUMMARY

Reaching a bike station without available bikes is a waste of a bike-user's time, as it necessitates having to go to another station that may be a minimum of 300 meters away in order to get a bike. An app was therefore needed that could save people time; Especially when time is scarce, such as in the morning when the goal of getting to work on time can be somewhat thwarted by the extra 300-meter walk to the next bike station. Aptly named, "Bike-Ability" that needed app is developed for the exact purpose of avoiding that type of time-wasting scenario. The app implements a Camera to keep track of the bike availability that is in turn relayed to the bike renting stakeholder.

Having the application installed will mean that the user's interaction with the bike station is more manageable in that the user is now notified of available bikes at bike stations. A camera that overlooks the parking bay/station is positioned and equipped to detect how many bikes are available through OpenCV and TensorFlow in Real-time. Information on the number of bikes on hand that are ready for use is then sent to the application end-user who will be online with the application installed on their Android handset.

INTRODUCTION

BACKGROUND

A TripAdvisor Dublin bike reviewer recently lamented: *“What use is a Dublin Bikes subscription when the stations are empty most times you need a bike?... I would not recommend Dublin Bikes.”* (TripAdvisor.com, 2022)

Its not only tourists with the same complaint though; nor is also not only empty bike stations reached for no good reason when trying to rent a bike, that is frustrating renters; the same applies to full stands upon returns of bikes. This is another common type of complaint when it comes to bike rental schemes, as seen in this trip advisor review: *‘I got a bike from Kevin treet9[sic] cycled to Fallon & Byrne Wicklow street stand was full. Ended up cycling to far side of Stephens Green to put back bike! Met later 3 Italian tourists trying to return bikes to Wicklow Street. More bike stands in Wicklow street please’* (TripAdvisor.com, 2022)

At its genesis, the bike Renting industry opened its doors during the boom of smartphones in tandem with the development of the IOT for which services started appearing on markets in support of it with commensurate speed. The first bike sharing service (BSS) was introduced to Ireland around Sept 2009 with the advertising funded Dublin Bike scheme following shortly after other major European cities such as Paris, and Barcelona in 2007. (DeMaio, 2020)

As supplementary and alternative form of public transport therefore, Dublin Bikes needed to increase its reliability. Close friends and relatives became an echo chamber for that need. The idea for the app was born a few weeks after hearing about that need one too many times. In light of both the current environmental crisis and looming economic crisis, it also makes sense that people would seek out more reliable alternative transportation such as bicycles or reaching their destination on foot. But when the walk would take an inordinate amount of time, that would impede daily productivity, then bicycles look like the most viable alternative.

The other stated reason for the rise in alternative transport forms to that of fossil fuel consuming transport, is the environment. European Union plans to reduce the number of fossil fuel driven vehicles by 2030, a plan that will benefit from increased reliability of city bike rental schemes. (Irish Times, 2020) The pandemic so it seems only sped up such plans as more people can be seen on roads with electrically propelled vehicles. And, albeit a new trend it has taken modern cities by storm in recent years, and has already carved out for itself a niche space in the eco-friendly urban transport arena. Civitas, a research thinktank advising the European Union on raising the efficiency of bike rental schemes recognizes that more bicycles on the road means less cars and therefore less fuel consumption and less parking space is needed. Another reason for increasing the attractiveness of bicycle schemes and cycling in general as alternative mode of transport is the mobility and travel independence provided by such schemes for people who do not have access to cars. Civitas (page 4) Even the United Nations weighed in on the push for increased use of bicycles. (United Nations, 2011)

Whatever the reasons, the need for reliable forms of alternative urban mobility increased over the past couple of decades, as traffic on urban and city roads became increasingly congested. Incrementally then the need for such a transport network grew in tandem with the responses offered, to the point now where people using such services want access to them at fingertips.

THE APP

With the application installed, user’s interactions with bike stations should be much more manageable with the current raised incidence of nuisance and time wasting experienced due to lack of facility; the notification that the user will receive in advance of reaching a bike station through the app, will fix that problem. Once bike rental schemes gain a reputation of being reliable in that way, more people would possibly opt to commute in that way. A second user-class is the Administration/ user. This user is able to perform CRUD administrative duties on the app.

Overlooking the bike station, a camera is positioned, equipped and trained through machine learning using TensorFlow to detect the number of bikes that are available. A real-time number of bikes ready and available for use, is then sent to the online handset of the application end-user. In that way, App users will then be able to reserve one of the available bikes.

As part of their services many and BRS Providers include apps that offer to the public an array of functions and information such as real-time information about station locations in relation to the current position of the user. Figure 1 exhibits a comprehensive list of basic functions on offer by such apps according to European Union funded research on bike rental services across the EU. (Pan European MasterPlan for Cycling Promotion, May 2021) The feature that this project aims to bring to that market is missing from that list, which is to improve bike rental facilities by providing real-time information on bike availability. In other words, the main feature of the Bike-Ability App, is therefore a completely new concept that is being introduced to that market.

Back-end	Front-end
Station Monitoring	Registration
Redistribution Planning	Rental
Defect Management	Information
Customer Data Management	Customer Data Management
Billing	Payment

Fig. 1. Back- and Front-end functionalities of most Bike Rental Apps across Europe as researched by Pan European Master Plan for Cycling promotion

AIMS

In summary, the functional objectives of the project, i.e to develop the app called Bike Ability that is scalable and that will support 6 use cases, namely:

- a) A Check Bike availability function
- b) A bike Reservation tool
- c) Notifications of future availabilities
- c) Bike Administration
- d) Contact us option and
- e) A Chatbot that user can information from about the app
- f) Registration and Login
- g) Language Changer

The application is aptly named 'Bike-Ability' with its name capturing its main functions being to inform or notify its end-user of the availability of bikes at a particular location in real-time. Additionally, app users will be able to reserve bikes through use of the app. With the app, its end-users who are also bike rental customers will therefore mainly save themselves from reaching bike stations without any bikes available.

The guaranteed time-saving aim of the application would not be possible without the user being able to reserve a bike within a set time of reaching a particular bike station. Without the reservation function, the application would serve less of a purpose, especially in situations where for example another user scoops up the available bike after a notification issued of its availability to the app user.

In a case such as that where another customer takes for example the last available bike, it would mean that the application served not much of a purpose to its users, because in that case the app user would be stuck in the same position as one who did not use the application at all. Notifications, are extra UX performance features implemented via the Dashboard activity. Through that, the app acquires a more personalized feel with timely messages sent to users that notifies them when bikes become available and they are able to get further information streamlined for their needs via the Chat function.

TECHNOLOGY

Technology wise the project uses a set of specifications that comprises both hardware and software components employed to function in tandem for the provision of the deliverables of Bike-Ability. Machine Learning is employed for TensorFlow supervised learning.

HARDWARE

Camera

An RPB-Pi micro-controlled camera is implemented as main back-end technological feature in order to achieve the application deliverables on offer, i.e. mainly, the detection of available city rental bikes at bike docking stations.

In turn, the Raspberry Pi as microcontroller incorporates Linux Distro and Open-Source Computer Vision Library (Open CV) that is an “open-source computer vision and machine learning software library”. (www.Open CV.org)

Raspberry PI (RPB-Pi)

RPB-Pi is a single board computer with a microcontroller. Its use in this project centers around the detection of bicycles in their respective parking spaces. When a bike is detected, Raspberry Pi sends the detection to Firebase.

Android OS Mobile Phones

Mobile Phones with different Android versions, ranging from Kit-Kat version 4 to Q version 11 will be used for testing purposes.

Memory/ Ram on the device

The device used will require minimum 2Gig of Ram in other words a mobile phone device of 5 or 6 years old should be sufficient. The higher the memory on the device used, the better-quality output from the application can be expected.

Google USB Accelerator

With Raspberry Pi capturing images at a mere rate of 0.5 frames per second, the goal of producing real-time information would be thwarted. This necessitated the implementation of an accelerator. Following research and comparisons of market offerings, the final choice of accelerator was made between competing Intel Movidius Neural Compute Stick and Google Coral TPU Edge Accelerator (CTA). The latter, according to the experts at TowardsScience.com, “contains an Edge Tensor Processing Unit (TPU), which provides fast inference for deep learning models at comparably low power consumption.” (Google Coral USB Accelerator Introduction, 2019)

The Coral also outperformed the “Movidius” in terms of frame rate per second, which informed the final decision to invest in it.



Fig. 2. Google Coral Tensor Process Unit (TPU) Source: Google Coral.com

SOFTWARE

Android OS

The application will be developed on Android Studio. The user will interact with the storage space – keep track of the bike there for example.

Android Studio Application

Android Studio is used alongside Java programming language to develop the application, while Python programming language will be used for the Raspberry PI integration of the Camera, with Firebase acting as an intermediary between Python and Android.

Raspberry PI (RPB-Pi) OS (Formerly Rasparian)

This is a Back-End Functionality used for the implementation of the camera module.

RP is the main component implemented for the detection of bike availability. It is a single board computer housing a microcontroller and the operating system was installed on an SD card for that purpose.

Firestore Database (FB)

FB is used for storing user data and bike availability data. It is a real-time database, this is exactly the function that I need to display information in real time. FB will work in conjunction with the Raspberry Pi that will save the information on FB. In this case, Android will act as a reader that will read from FB. For the login and registration stored on FB, android functions as both reader and writer to FB. FB therefore serves as intermediary between Raspberry PI and the Android application. While the Raspberry Pi camera captures bike availability information to be saved on FB, the Android application will access and retrieve the information and display it in a List called “Recyclerview” at the end-users request. All User app interaction and user information is stored on Firestore database.

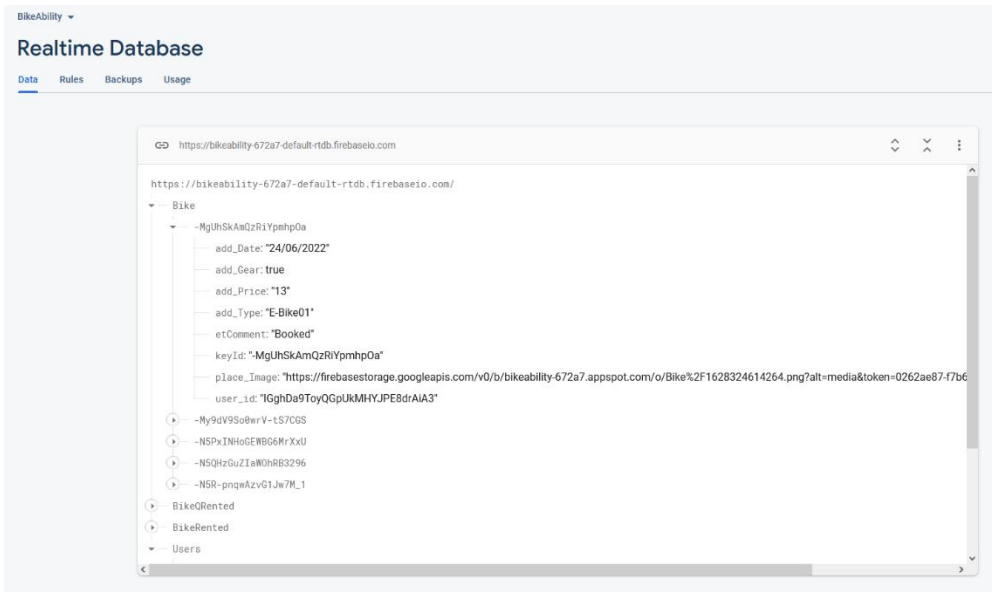


Fig. 3. Firebase database Structure

Open CV

According to the Open CV website, Open CV was built to ‘provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products’. (www.Open CV.org)

It is a BSD-licensed product, meaning that it belongs to a group of permissive free software licenses, that imposes the least number of restrictions on its use and distribution. As such its use is unencumbered by intellectual property laws as well as financial constraints. When this package was researched it was additionally found that its users found it easy to work with and its code is easily modified. (OpenCV team, 2020) That is one of the reasons that this library was chosen to draw rectangular shaped regions of interest or bounty boxes for object detection. OpenCV works in conjunction with TensorFlow that is set to detect bikes.

Even though its accessibility and ease of use informed the decision to use it for the purposes of this project, being open-source it may not match dedicated real-time object detection tools such as YOLO V3. Nonetheless it suffices in more way than one for the purposes of this project. Besides, by comparison to YOLO, Open CV out-performed the resource hungry YOLO when it came to using it on the tiny hardware components of Raspberry PI. Economically then, OPEN CV needed less resources to complete the same task as its counterpart YOLO in achieving the same standard of results.

TensorFlow Lite

The core open-source library of Tensor Flow, which is an end-to-end open source supervised machine learning platform is chosen to develop and train the ML framework of this project. It has a

“comprehensive, flexible ecosystem of tools, libraries and community resources”. For the purpose of this project Support Framework TensorFlow Lite, an optimized version that is less heavy on resources was used, as it is especially designed for mobile devices, such as Raspberry Pi that is limited in its scope as hardware. TensorFlow comes with a pre-trained object detection model. Machine learning is employed to distinguish between the two when detecting bicycles. The pre-trained contains 90 objects for recognition, however the model will be trained to recognize additional custom images of bicycles so as to increase detection ratio.

The trained model enjoys optimization through size reduction, because this has the benefit of storage space being occupied, that additionally requires less time to download and less bandwidth, making the model resource efficient. Another benefit that optimization brings is that smaller models requires less RAM to run and this provides for optimal stability and performance. In exchange for an affordable amount of accuracy, quantization shrunk the files fourfold, that further increases the efficiency of detection. (Real-Time Object Detection Using TensorFlow, 2021)

Balena Etcher software

Balena is used to write the Raspberry Pi OS on the SD card. Raspberry Pi does not come with a pre-installed OS and thus requires an operating system. It has an SD card slot where the OS will be written on and manually installed.

Rational Rose for use case Diagram

This software allows for the plotting of USE Case Diagrams that are incidentally also herewith included.

Balsamiq Mockups 3

A wireframe will be mocked up with this software in order to showcase the application to prospective customers in advance of installation or engagement.

J-unit

The unit testing framework, for Java programming in Android, J-Unit is used for the testing of Bike-Ability. J-Unit is instrumental in the unit testing of small snippets of Code.

PROGRAMMING LANGUAGES

Java for Android

Java for android is the main programming language used to develop the application because of knowledge, experience and familiarity with the language. Kotlin was considered but is not fully stable.

Python

Python is used for the detection of Bikes with Raspberry Pi. It is the main language used for work with Raspberry Pi. The array of languages that could be used for Raspberry Pi is limited to C++, Python and Scratch. Python was prime candidate for reasons of it being a familiar language.

STRUCTURE

Thus far, an overview of the project was introduced by the background, aims of the project and the technologies that are employed in its development. The rest of this document is divided into sections that details System- and Use Case Diagrams including Functional and non-functional requirements under the System heading. Following the latter are sections that detail Design and Architecture, Implementation details and a summary of Tests, including Validation, Integration and Unit Testing and an Evaluation section that provides an overview of the evaluation process.

Whereas the Implementation section details technical aspects of code implementation, the testing section provides examples of some of the tests performed with screenshot examples of the code used to implement the concepts in this project. The final three sections conclude the report with details on Future Work, a Conclusion and a list of all appendices attached, such as the project proposal for this report and reflective journals that reflects on the different stages of the project on a monthly basis over the time period that it was in progress.

SYSTEM

The RpB-Pi camera module captures bike availability information to be saved on Firebase, that can then be accessed by the Android application that will in turn display it in a List called Recycle View for the end-user's perusal. TensorFlow contains a list of objects in this case bicycles that will be recognized during training and OpenCv is instructed by tensor flow to encase or place the detected bicycle from the object list inside the detection frame. Firebase will essentially be acting as an intermediary between Raspberry Pi and Android.

System workflow starts where users will be required to create new user accounts with account details that they register, otherwise users will be disallowed from accessing the features of the application. The Account creation process therefore comprises starts with the creation of a unique login ID and password. The latter will be required to have specific characteristics; the username must contain for example a capital letter and numerical values and the password must be of a certain length containing certain types of characters for the once off registration process. Another test that both username and password should withstand is that there should not pre-exist another user with the same user name.

Once user details are successfully registered the user details are ready for use at 'login'. After the user enters the registered login details, Firebase initiates the authentication process to validate user credentials. User login credentials serves as user interface that provide users with an access path to the main menu of the application. The account creation process is a once off procedure and certain information of the user will be collected and stored securely in online Firebase database. An added layer of protection to recover lost or forgotten user details such as username and passwords will also be coded into the application.

Data Logistics are handled by Firebase. Upon the verification of the user's new account details and credentials, the application connects to Firebase Authentication that will be employed to facilitate the account verification and login experience of the user. It is at this stage that the application will be able to verify that there are not any duplicate accounts with the same account user name or passwords. Happenstance that duplicates account details are entered by the new user, then the user will be informed be prompted by the application to enter new details as the initial registration attempt failed.

SYSTEM USE CASE DIAGRAM

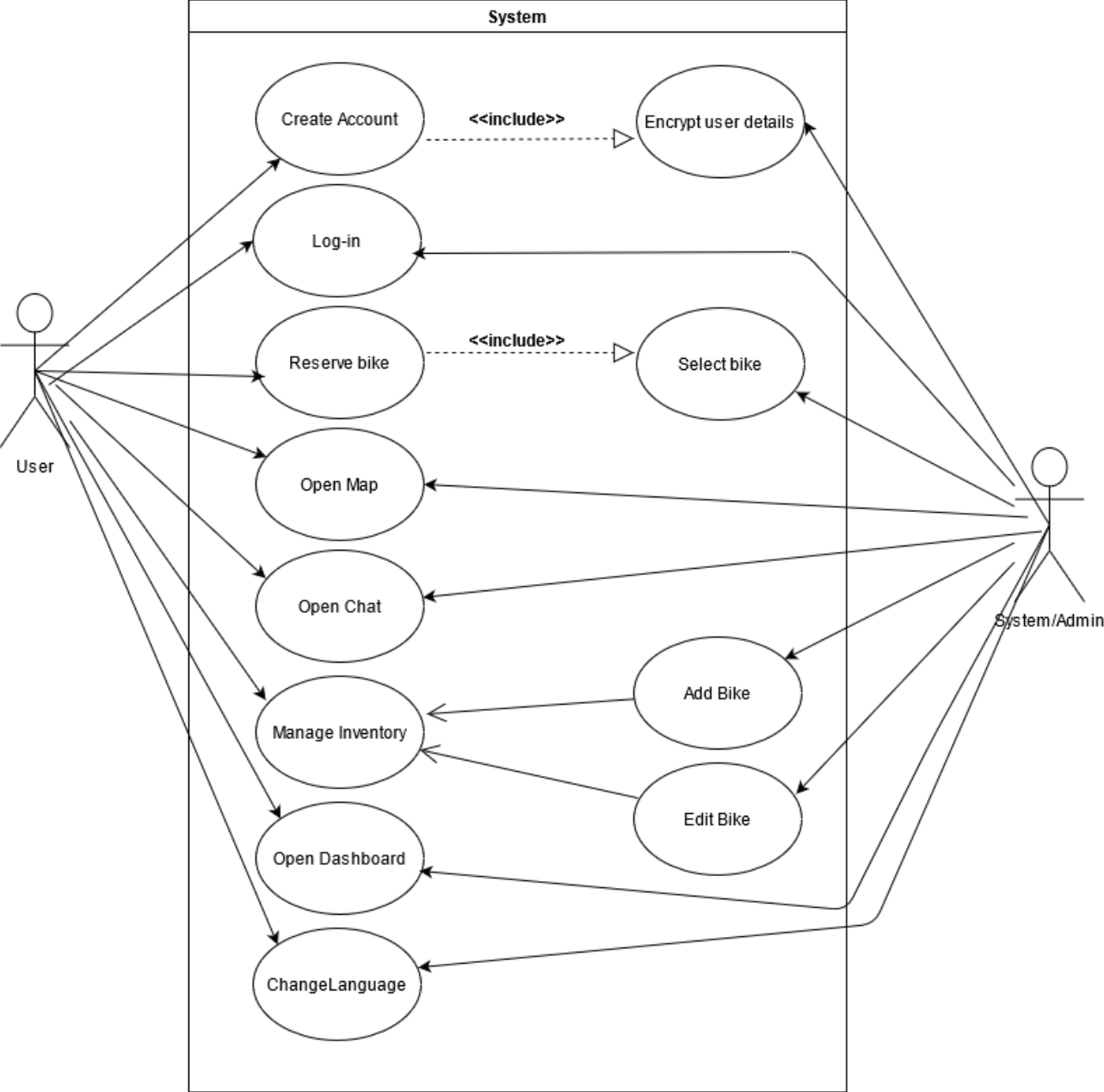
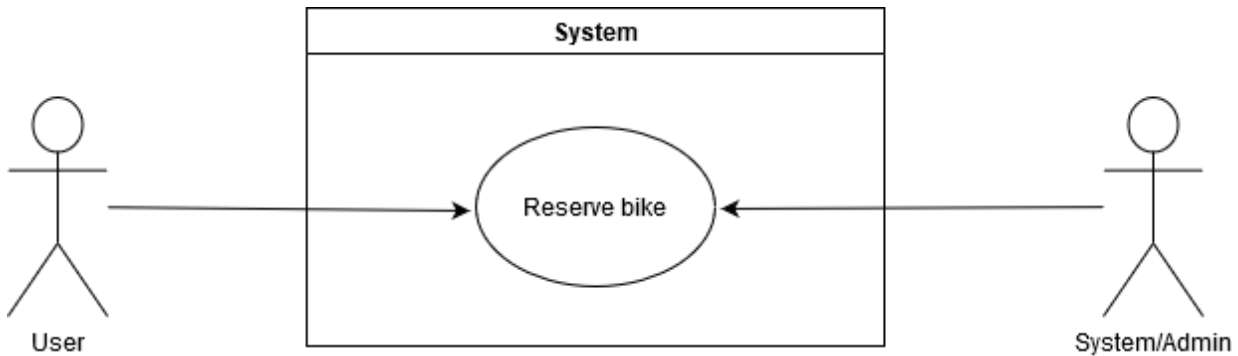


Fig. 4. Use case diagram

USE-CASE DIAGRAMS

UML Reserve Bike



Requirement 1 < Reserve Bike >

The requirement name is Reserve a bike

Use Case

Reserve Bicycle

Description & Priority

The reserve a bike requirement allows the Actor to reserve an available bike from the bike station. This requirement carries high priority because without a reservation there is no point in informing the actor when there is a bike available, since someone else can take the available bike, meaning that the actor will get to an empty bike station, i.e. the exact thing that this app is trying to avoid. It is therefore imperative to reserve a bike for the functionality of the overall system.

Scope

The scope of this use case is to allow the user to reserve a Bike

Description

This use case describes the process of reserve a bike

Use Case Diagram

The actors are the User and System Admin

Flow Description

Precondition

The application is online, the user must be logged in. The phone must be powered on

Activation

This use case starts when a request to rent a bike is send

Main flow

The System identify that the user has press the rent a bike button

1. The System display the list of bikes available from the List
2. The User select the bike from the bike list
3. The User reserve the bike
4. The system displays a successful message of bike reserved

Alternate flow

A1 : Error Message

The system displays an error message; Bike unavailable for reservation

The System ask the user to click on the message to continue

The use case continues at position 3 of the main flow

Exceptional flow

N/A

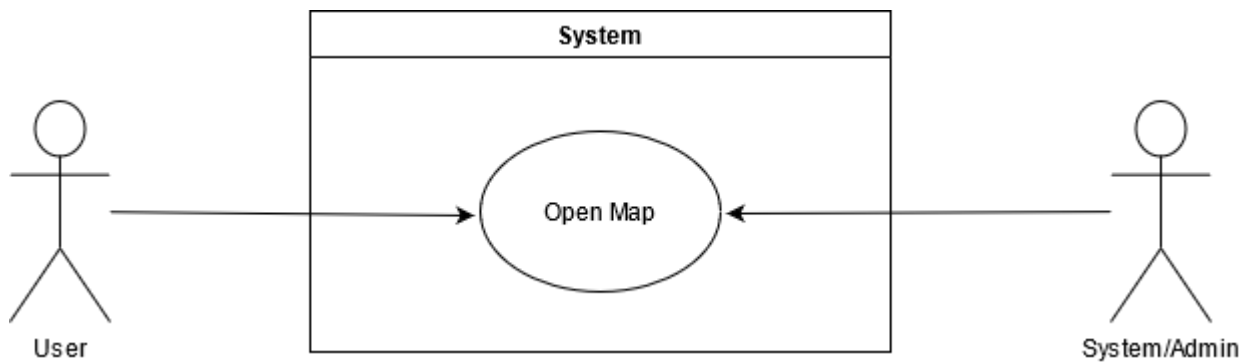
Termination

The system presents the next main menu with Welcome Message

Post condition

The system goes into a wait state

UML Open Map



Requirement 2 <Open Map >

The requirement name is Open Map

Use Case

Open Map

Description & Priority

The rent a Bike requirement is required that allow the Actor user to rent a bike from the bike station. The priority for this requirement is high. This is the main activity of the application. The user is able to get notified of the availability of the bike through the image feed from the camera.

Scope

The scope of this use case is to allow the user to rent a Bike

Description

This use case describes the process of renting a bike

Use Case Diagram

The actors are the User and System Admin

Flow Description

Precondition

The application is online, the user must be logged in. The phone must be powered on

Activation

This use case starts when a request to rent a bike is send

Main flow

1. The System identify that the user has press the rent a bike button
2. The System display the list of bike available from the List
3. The User select the bike from the bike list
4. The User rent the bike
5. The system display a successful message of bike rented

Alternate flow

A1 : Error Message

1. The system display an error message; Bike unavailable
2. The System ask the user to click on the message to continue
3. The use case continues at position 3 of the main flow

Exceptional flow

N/A

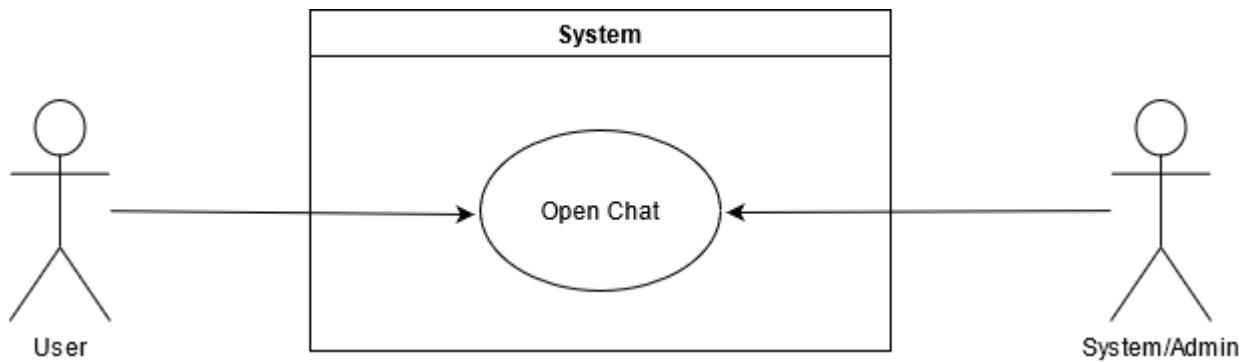
Termination

The system presents the next main menu

Post condition

The system goes into a wait state

UML Open Chat



Requirement 3 < Open Chat >

Use Case

Open Chat

The requirement name is “Open Chat”

Description & Priority

The Open Chat requirement is required allow the Actor user to share a bike from the menu list and include his or her contact details. Priority of this requirement is medium, because this is not one of the main features of the application, but is an extra to facilitate the user interaction

Scope

The scope of this use case is to allow the user to Open Chat

Description

This use case describes the process of Open Chat

Use Case Diagram

The actors are the User and System Admin

Flow Description

Precondition

The application is online, the user must be logged in. The phone must be powered on

Activation

This use case starts when a request to open chat is send

Main flow

1. The System display a list of topics that user can select
2. The User type the selected topic and submit it
3. The System confirm the user of the selected topic
4. The User is presented with information about the selected topic
5. The User select the topic(A1)

Alternate flow

A1 : Error Message

1. The system displays an error message;
2. The System display a message and ask the user to choose different topic
3. The User change the initial request
4. The use case continues at position 2 of the main flow

Exceptional flow

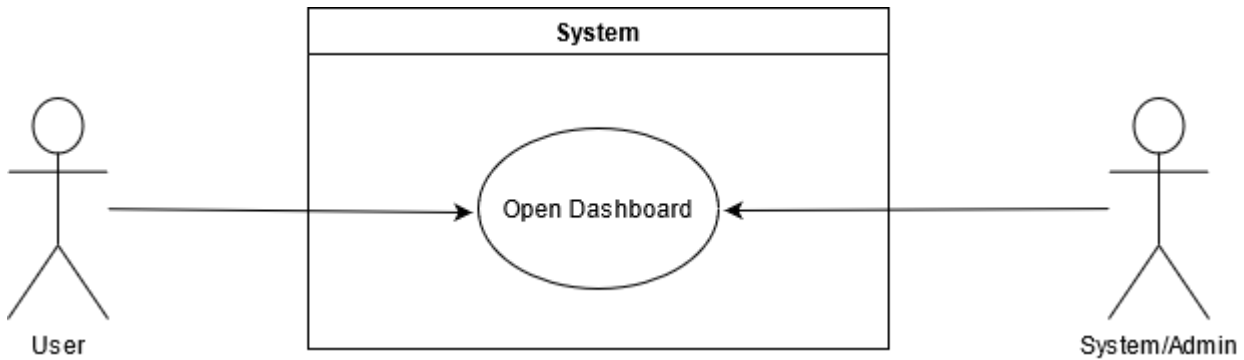
N/A

Termination

The system presents the next main menu with Welcome Message

Post condition

The system goes into a wait state



Requirement 4 <Open Dashboard >

The requirement name is Open Dashboard

Description & Priority

Open Dashboard requirement gives the Actor access to the dashboard that allows for the activation or deactivation of notifications that notify the user soonest a bicycle that is currently unavailable becomes available. Priority on this requirement is high because it is one of the main functions of the application that allows the user to choose whether to activate notifications or not

Use Case

Open Dashboard

Scope

The scope of this use case is to allow the User to access the dashboard where they can activate notifications

Description

This use case describes the process of disable or enable notifications

Use Case Diagram

The actors are the User and System Admin

Flow Description

Precondition

The application is online, the user must be logged in. The phone must be powered on

Activation

This use case starts when a request to access Dashboard is send

Main flow

1. The System display the Dashboard
2. The User is presented with activate or deactivate notification

3. The System notify the User that the notification was enabled or disabled

Alternate flow

N/A

Exceptional flow

N/A

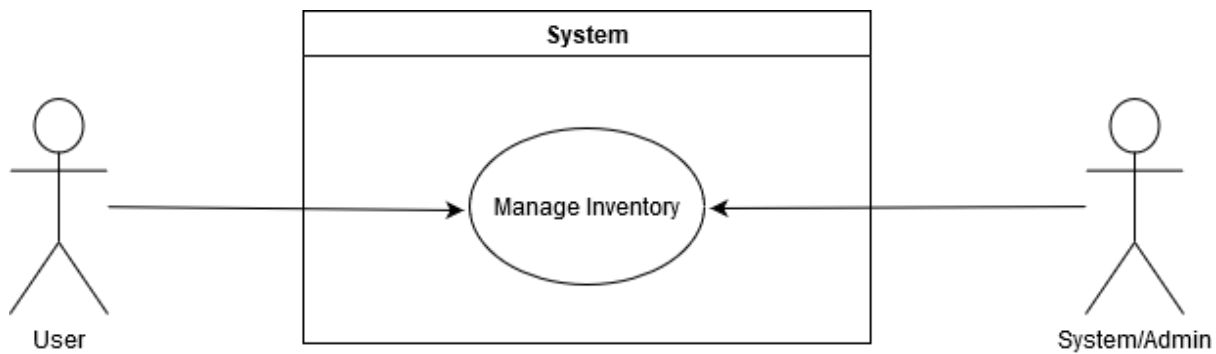
Termination

The system presents the next main menu with Welcome Message

Post condition

The system goes into a wait state

UML Manage Inventory



Requirement 5 <Manage Inventory >

The requirement name is Manage Inventory

Description & Priority

Manage inventory is a back-end functionality that allow the administrator/Actor to for example remove, add or edit the number of bikes that are being repaired. Priority on this requirement is high because the administration function is highly prioritized.

Use Case

Manage Inventory

Scope

The scope of this use case is to allow the System Admin to update and edit Bike menus

Description

This use case describes the process of update and edit Bike

Use Case Diagram

The actor for this use case is System Admin

Flow Description

Precondition

The application is online, the System Admin must be logged in

Activation

This use case starts when a request to access inventory is send

Main flow

1. The System display the inventory
2. The System Admin add the bike details to the inventory
3. The System notify the Admin that the bike was added successful
4. The System Admin remove the bike from the inventory
5. The System notify the Admin that the bike was added successful

Alternate flow

A1 : Error Message

1. The system displays an error message
2. The System display a message and ask the user to choose different topic
3. The User change the initial request
4. The use case continues at position 2 of the main flow

Exceptional flow

N/A

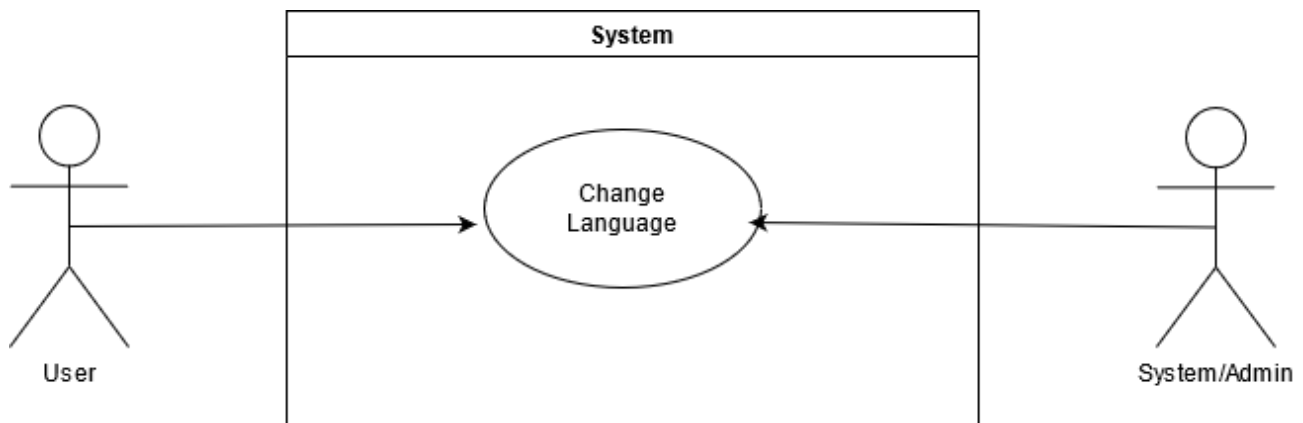
Termination

The system presents the next main menu with Welcome Message

Post condition

The system goes into a wait state

UML Change Language



Requirement 6 <Manage Inventory >

The requirement name is Change Language

Description & Priority

Change Language is a back-end functionality that allow the User/Actor to change language from English to Italian end vice versa, Priority on this requirement is low because the application is fully functional with the default English language, in other words, this is an optional usability feature.

Use Case

Change Language

Scope

The scope of this use case is to allow the User to change Language

Description

This use case describes the process of Change Language

Use Case Diagram

The actor for this use case is the User

Flow Description

Precondition

The application is online, the User must be logged in

Activation

This use case starts when a request to change language is send

Main flow

1. The System display the language setting

2. The User select the desired language
3. The System notify the User that the language was changed successful

Alternate flow

A1 : Error Message

1. The system displays an error message
2. The System display a message and ask the user to choose different selection
3. The User change the initial request
4. The use case continues at position 2 of the main flow

Exceptional flow

N/A

Termination

The system presents the next main menu with Welcome Message

Post condition

The system goes into a wait state

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

The intended outputs of the system can be summarized by the following table:

FR No.	FR Name	Description	Priority
1	Login and Registration Page	Customer Details Registered and Login Credentials validation	High
2	Available Bike Detection	“Computer vision model trained on a bicycle picture, using machine learning”. Open CV used.	High
3	Reserve Available Bike	Customer upon finding available bike may reserve an available bike which automatically locks the bike with the reservation code generated. If unused a penalty fee levied because of the inconvenience unused reserved bikes would pose to other users who may not have cancelled. If reservation cancelled within window period (for example 5 minutes) then no fee levied.	High
4	Cancel reservation	The customer is allowed to cancel a reservation within a set amount of time	High
5	Availability Notification	A notification will be sent to inquiring customers who will be allowed to choose to be notified Notifications is a front-end functionality that allow the User/Actor to get notified when bikes that they are interested in that has already been booked, becomes available. Priority on this requirement is Medium because the application is fully functional without it, but it does enhance and is connected to the core feature of the app, which is to prevent users from getting to empty bike docking stations	Medium
6	Reserve/Rent a Bike	Bike Rental is available through the Reserve button on the home-screen navigation bar of the App	High
7	Rental Statement	Keeps track of number of rentals, reservations and cancellations	Medium
8	Administration	The Admin user have his/her own access and CRUDE functions	High

Functional Requirements

Login and registration Requirement (Priority: High)

Users will be required to create a unique login ID and password with specific characteristics such as that the username must contain for example a capital letter and numerical values and the password must be of a certain length containing certain types of characters for the once off registration process. A that both username and password should withstand is that there should not pre-exist another user with the same username. This process comprises the registration and creation of a new user account without which the user is disallowed the use of the application. When using the registered login details, Firebase will initiate the authentication process that will validate user credentials before being allowed access to the website functions and content.

Available Bike Detection (Priority: High)

This is made possible through OpenCV that a camera module is trained with to detect available bikes. The camera is affixed to a spot, typically a light post or a building or even trees in range of the bike station. “Computer vision model trained on a bicycle picture, using machine learning”. Open CV used.

Reserve Available Bike (Priority: High)

Customer upon finding available bike may reserve an available bike which automatically locks the bike with the reservation QRCode generated. If unused a penalty fee levied because of the inconvenience unused reserved bikes would pose to other users who may not have cancelled. If reservation cancelled within window period (for example 5 minutes) then no fee levied.

Cancel reservation (Priority: High)

A cancellation button will be visible and available for the user after a reservation has been made. This will be implemented on the Navigation Drawer menu and on a dropdown menu under the customer user profile.

Availability Notification (Priority: High)

A notification of available bikes can be scheduled by user that will provide real time information of bikes available at a certain station chosen on the map. This is one of the main perks of having the app installed.

Reserve a Bike (Priority: High)

The Rent a bike feature is implemented via the Navigation Drawer menu and The Home activity that will give users the option to reserve a bike.

Rental Statement (Priority: High)

These statements are track records of user's rental activities that are also available to the user under the user profile dropdown menu.

Administration / User (High)

This function serves one of the two user classes, namely the Administrative User, that will have Admin CRUDE function.

NON-FUNCTIONAL REQUIREMENTS

Non-Functional requirements are requirements that are needed by the app from its users and are largely impacted by the usability of the App. These requirements were elicited through Usability Testing techniques.

Usability Requirements

The application design sport features that users are generally familiar with, such as a Login Page, with buttons leading to menus in ways that most standard application UI's look. The reason for this is so that the application would not require any training per se and novice Android users should be able to use the application easily. Nielsen's 10 Heuristics is complied with in the design and architecture of the App.

The usability of the application forms the body of Non-functional requirements of the app and depends on a number of factors that are prioritized, such as the use of screen real estate i.e., where buttons are placed and importantly, the ease of use of the application in terms of its Graphic User Interface (GUI).

Data Requirements

Due to the FB database being a real-time database, it offers the facility needed to display information in real time. It will be deployed in conjunction with the RPB-Pi when information needs to be saved on FB. Android is used as both reader and writer to the FB database to both write and read in the retrieval process, the user's login details that was stored on FB. FB therefore acts as intermediary communicator between the Android application and RPB-Pi.

User Requirements

The first user requirement for the system to work is an Android OS mobile device. Mobile phones or tablets that that are capable of running Android Kit-Kat 4.4 right up until Android v.11. are need to run the application. Developing the application for this range of Android operated devices ensures that a larger number of mobile phone users will be able to download and use the application. An expansive user base for the application is important because the service or industry i.e., bike rental schemes that are public or private, that the application is useful for have maximally the working population of a city to serve.

Reliability Requirement (Priority: High)

The operating time taken for the application to perform its function indicates the reliability of the application. Typically, the amount of downtime of the application is a measure of its reliability and this is the main yardstick for Bike-Ability. Take the example of FB, that is owned by Google to illustrate the importance of up-time/ down-time functioning of the application; Google guarantees a 99.95% SLA for FB, meaning that FB is available to its users at a rate of 99.95%. The method for calculating "Monthly Uptime Percentage" is done by establishing the number of continuous 5-minute periods of downtime experienced by users of the application over a one month period and then subtracting that from 100%.

For the purpose of Bike-Ability, "Downtime" means more than five percent Error Rate due to complete lack of connectivity to the service. Downtime is measured based on server-side Error Rate., [which in turn refers to] "the number of Valid Requests that result in a response with HTTP Status 500 and Code "Internal Error" divided by the total number of Valid Requests during that period. Repeated identical requests do not count towards the Error Rate unless they conform to the Back-off Requirements" (Realtime Database, Firebase, 2020)

This is the same method that will be used to test the down-time ratio of Bike-Ability. (See the testing section under section 9 below)

The application's reliability is also dependent on the accuracy of input/output response ratio, in other words whether the information provided(outputs) accurately correlates to what was requested (inputs). Bike-Ability is expected to both accurately capture, record and notify of available bikes. Once these functions can be achieved smoothly and without bugs 99.8% of the time, then the application can be said to be reliable. The maximum acceptable time (or average time) for a service request will be 1 second.

Performance/Response time requirement (Priority: High)

Performance and Response time requirements refers to how fast the system is required and also expected to respond to user inputs and requests. Due to the nature of the main purpose of the app having to do with providing real-time information, the priority of this requirement is High. Here a number of factors inter alia, “Big o notation”, hardware, the ram on the device and software and the particular code that will be used needs to be considered as impactful on the overall system response time and general performance of its main functions.

It is further expected that the system will function without lagging that increases downtime. The system must complete updating the databases, with newly registered user information, bike sharing information and rental information as required by users. All such functions must be in a state of readiness to perform soonest the application is logged in to.

Security requirement (Priority: High)

This aspect of apps functionality will be verified at testing stage (See 9.1 below), and that will determine whether for example the user is securely logged in without third party access to user information and without duplications. Firebase will securely store user passwords in FB, the NoSQL database managed by Google Inc.

Protection (Priority: High)

Implementations that will protect the user from malicious or accidental access, modification, disclosure, destruction, or misuse are:

- Historical data sets will include activity logs that are only available to logged in users.
- Data Encryption – Application data will be encrypted.
- Data integrity will be maintained by frequent data integrity checks.

Recovery requirement (Priority: High)

Back-ups will be stored using NoSQL Firebase so that there will not a loss of data as Firebase has boasts reliability of 99.99%.

Availability Requirement (Priority: High)

The application is expected to have a 99% availability rate due to the nature of its main functions being to relay real-time information to customers.

Recovery Requirement (Priority: Medium)

In case of failure, user information is retained and not lost where the system shut down in an unexpected or unforeseen way. The use of Cloud based storage and Firebase offer protection against loss of information. This is useful for the recovery of passwords and user information also and will be used in that manner.

Maintainability Requirement (Priority: Medium)

Maintainability as requirement is an indication of how easy or difficult it is to modify the application. Bug-Fix reports are scheduled to periodically issue that will provide details of changes to the application. A support feature in the form of a 'contact developers' link will enable users to get in contact with developers for bug fixes. A complaint section will be implemented so that problems with the application can be reported and fixed within a set turnaround time.

DESIGN & ARCHITECTURE

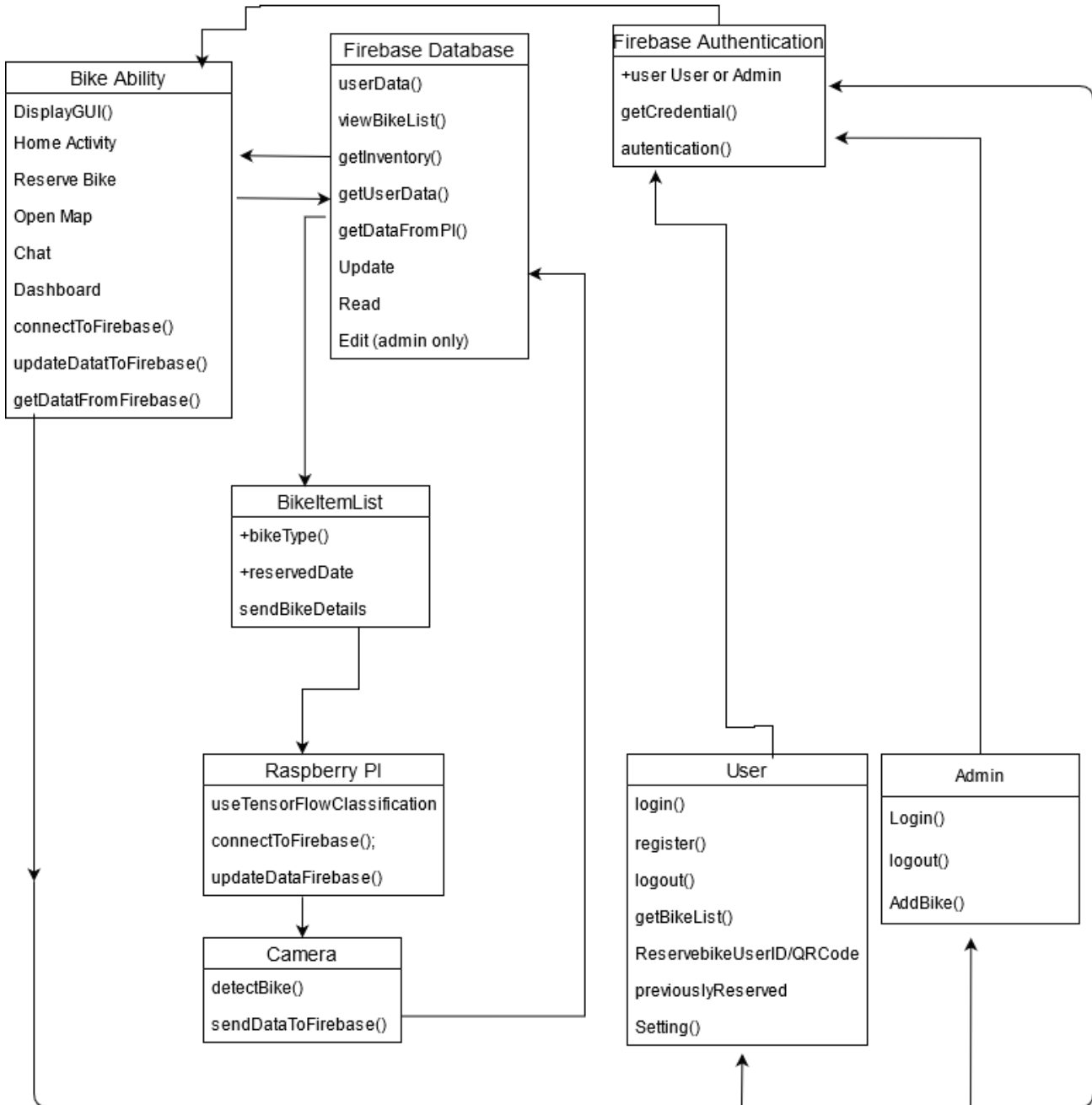


Fig. 5. System Architecture Schema that shows the components of the system

The app is designed and built with two classes of users in mind namely Bike Rental Customers and Bike Docking station administrative staff members that are referred to as ‘Users’ and ‘Admin’ respectively. The first component of the app is the ‘Login/-Out’ Activity, where the system

communicates with Firebase to create and authenticate user credentials. The app accesses Firebase when the user logs in to the app. Available bikes are stored in Firebase, that is updated, read and edited after Raspberry Pi detects available bikes. In other words, Firebase functions as intermediary between the app and the Raspberry Pi Camera.

GRAPHIC USER INTERFACE

UX Wireframes

A wireframe is a layout of the application that has the purpose of demonstrating where buttons are placed, and to give a rough idea of how and where functions will be placed on the UI of the app. Buttons that support the functionality of the application such as ‘Reserve Button’, or ‘Check Availability’ button is placed intuitively and ergonomically, meaning that such buttons will be visible and placed where for example they can be reached with the use of the thumb in cases of one-hand phone use. The application will host a navigation drawer that is accessible through Hamburger menu. **Fig 6 to 8** below shows the wireframes that represents the planning, pre-coding phase of the design of the app. Through it, a rough idea of buttons and menu placements were represented by wireframes.

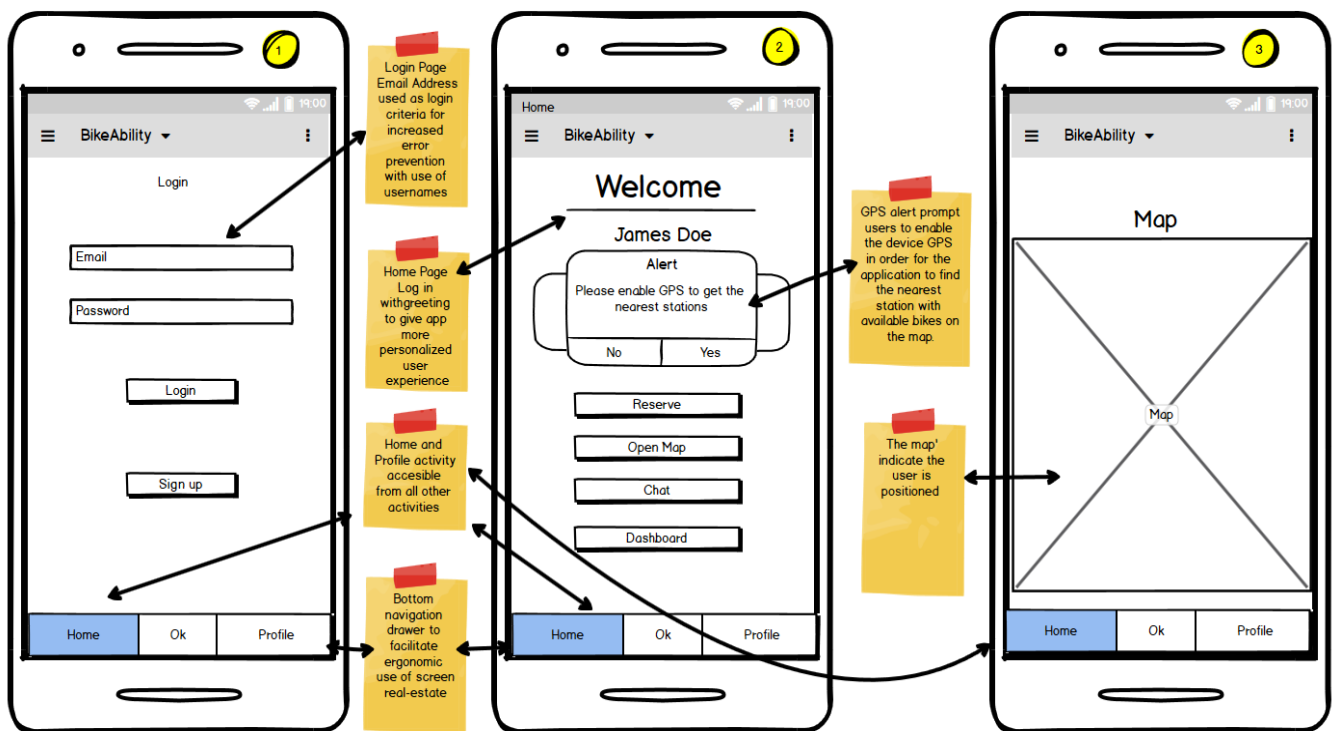


Fig. 6. Wireframe of Login, Welcome Screen and Map activity

From the Home page, shown in **Fig 6** above, all other activities are visible and accessible, with the planned for hamburger menu at the top right of the screen. At this stage, the placement of a three-dot settings button is also included.

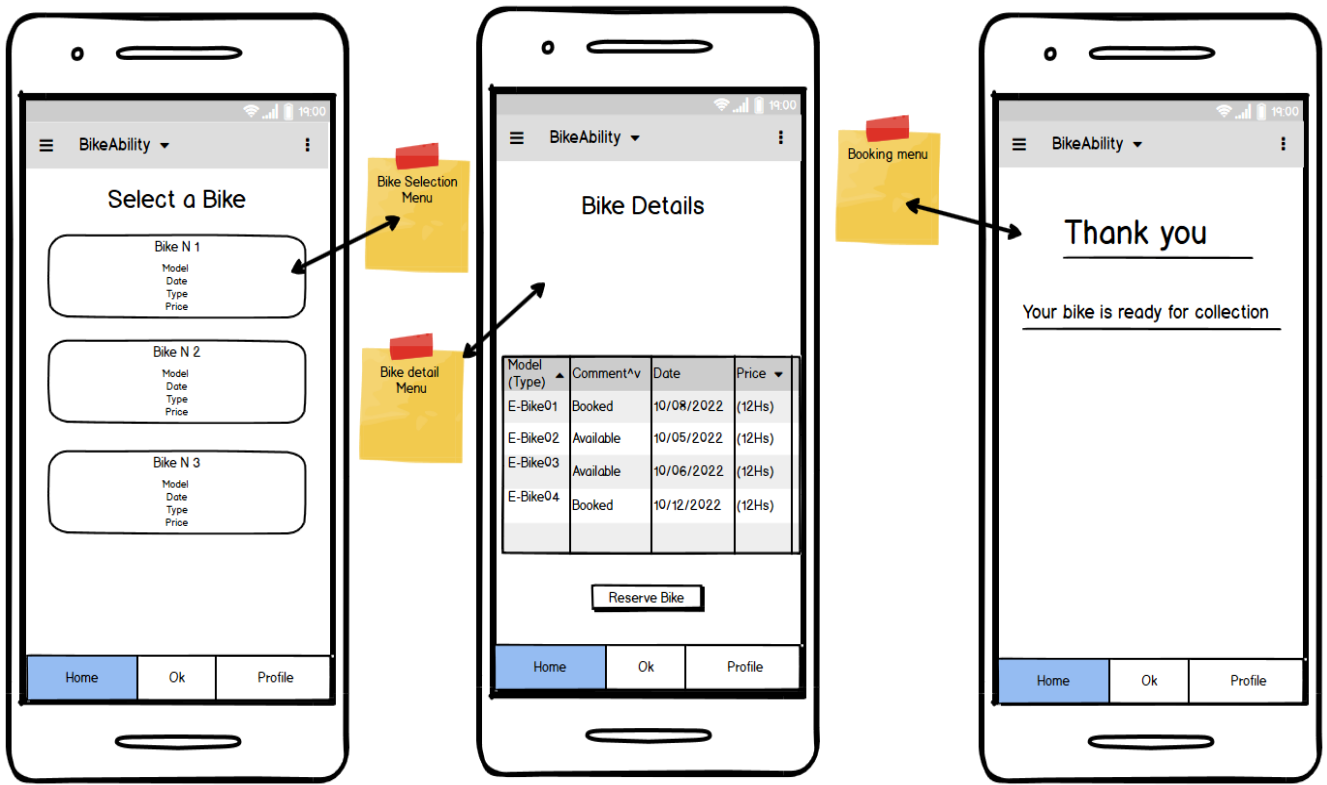


Fig. 7. Wireframes of Selection, Summary and After-booking page

Wireframe of **Fig 7** above details the 'Reserve a bike' activity, where bike is selected, selected bike details are then displayed to customer and finally the customer is notified that the bike is 'Ready for Collection'.



Fig. 8. Two Wireframes for Admin Activity

At planning stage, the wireframe at **Fig 8** demonstrates the process of Admin activities who has the power to CRUD bike inventory.

Bike-Ability Screen-Shots

Some UX design principles were incorporated in the design of the User Interface such as Nielsen's 10 Interactive UX Design Principles. One of the principles among others that was employed in the design of this app, was to consistently use what users already know. For example using the colour **green** that is universally used to indicate 'go' in traffic and 'red' to stop, for bikes being 'available' or 'booked' respectively.

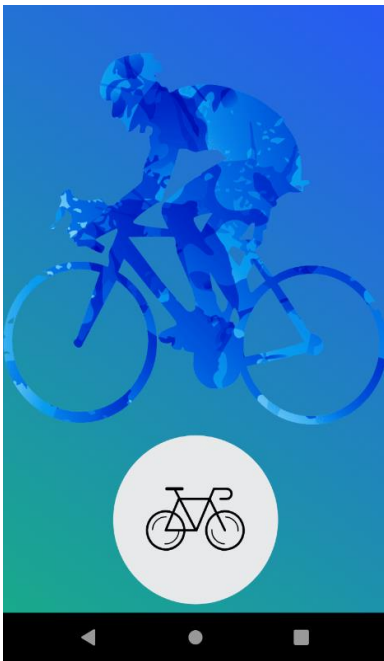


Fig. 9.

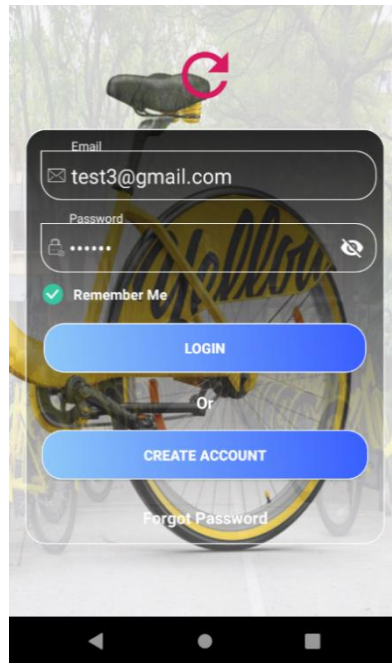


Fig 9.1

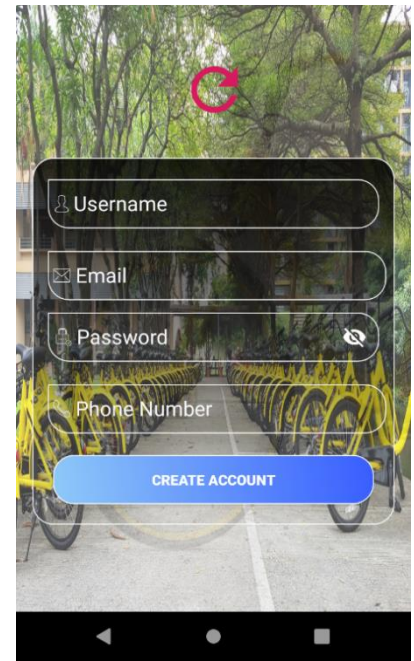


Fig 9.2

Fig 9 Above shows splash screens of the app. The app opens with the first image on the right, an animation with the bicycle wheels spinning for effect lasting for about 3 seconds before moving to the second page at **Fig 9.1**. At **Fig. 9.1**, the login page, the Login Button and Create Account Button is highlighted and enlarged so that the user knows exactly what the page is about at first glance.

When choosing the 'Create Account' button, the app opens the screen at **Fig 9.2** where user details can be filled in that are stored on firebase as the account gets created. The progress icon in red at the top center position serves as feedback mechanism to let the customer know that his or request is loading in align with the interactive design principles laid down under the Nielsen's 10. Whether the user is an authorised as Admin or ordinary user depends on the credentials entered and validated via Firebase.

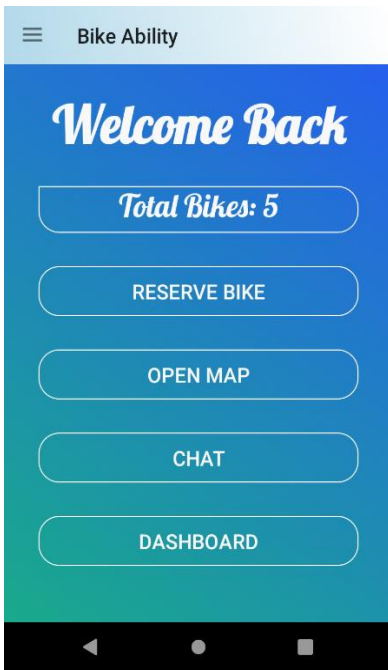


Fig. 10.

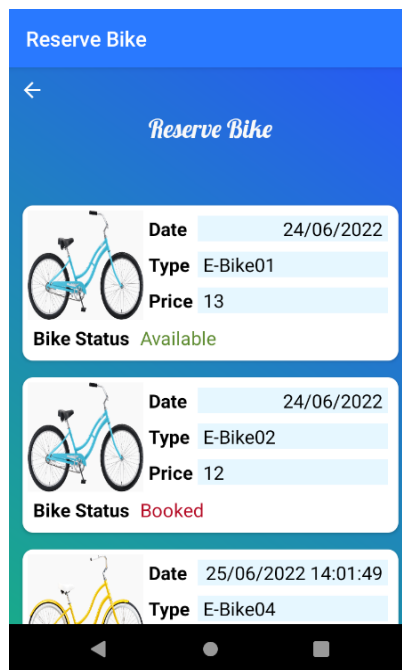


Fig 10.1

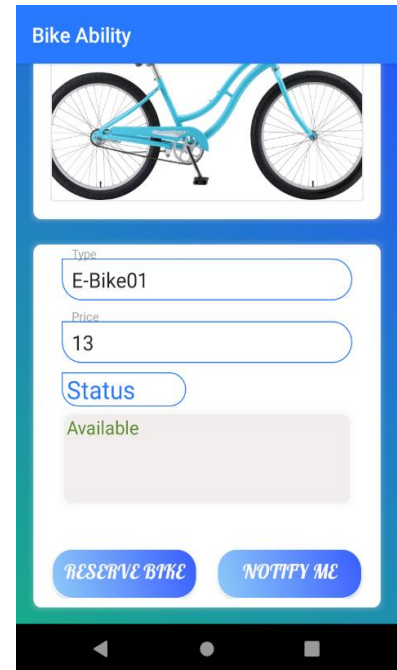


Fig 10.2

Fig 10 exhibits the main activity that is displayed to the user after successfully logging in. The user cannot access this page without being registered and without entering the authenticated credentials. Various activities are displayed in succession of each other whereby the user can access the activities by selecting them. The design at this point is kept to a minimal and the same color scheme is chosen as the opening animation. Blue and green colors were chosen for a calming effect as there are 5 different pieces of information on the page.

Fig 10.1 If the user selected for example the 'Reserve Bike' activity, the screen at Fig 10.1 opens, where the first thing at the top of the screen informs the user that s/he/they has now landed on the 'Reserve Bike' activity, where s/he/they are presented with a list of bikes. The 'Bike Status' field shows that the bike is **Available** in green and in **Red** if the bike is already booked. The choice of these colors Green and Red are based on these colors being used universally as colors for 'go' and 'stop' in traffic signs and thus users are already familiar with these colors that stands out to catch their attention and inform them. It is worth noting here that also the booked bikes can be clicked on, which directs the user to the activity at Fig 10.2

Fig 10.2 Once the user chooses an Available bike from the list in Fig 10.1, the activity at Fig 10.2 opens up that allows the user to 'reserve' the available bike or get notified when there is a status change for the bike that the user is interested in. 'Reserve Bike' and 'Notify Me' buttons appear larger than the rest of the text on the page so that they stand out to the user who then knows exactly what the page is about. If the selected bike is still 'available', a backend validation runs that result in the user being informed that the bike is already reserved if so. If the selected bike is a 'booked' bike, then the user can use the 'Reserve Bike' button to reserve the selected bike.

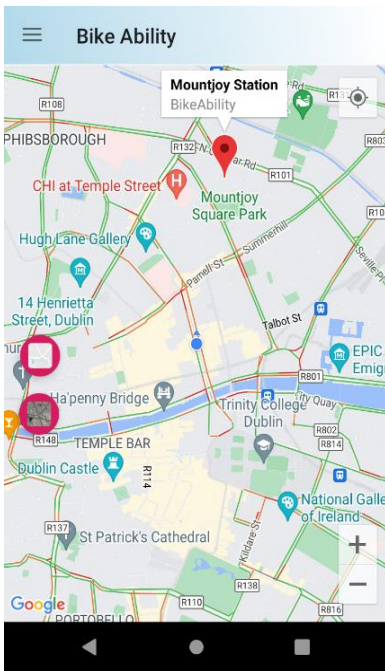


Fig. 11.

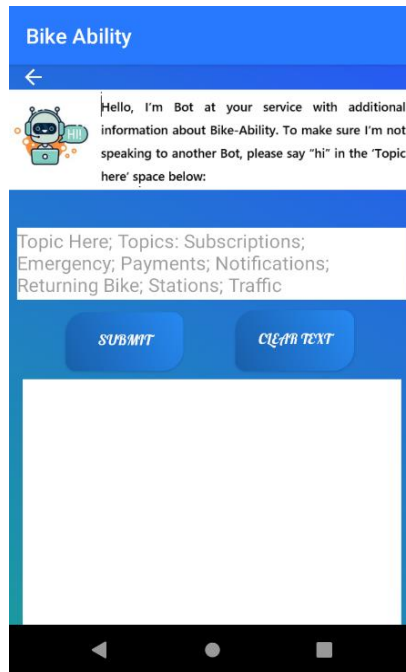


Fig 11.1

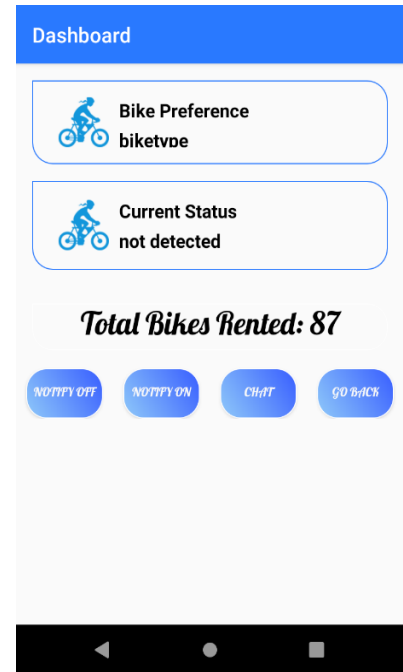


Fig 11.2

Fig 11. If the user selected the ‘Open Map’ activity from the main activity at **Fig 10**, the GPS coordinates automatically opens on the map at **Fig 11** and zooms in on the user’s position using a clickable POI. The red [POI](#) on the Map stands out to indicate the station where bikes are stationed. As a shortcut, and in accordance with Nielsen’s 10 Principles, the POI is clickable and directs user to the reserve bike activity. Through the Google Maps API, a red line is visible on the main roads that indicates high traffic density and by contrast green lines indicate low traffic density.

Fig 11.1 The screenshot demonstrates how the Chatbot works, opening at the top of the screen with the greeting and introduction to the user. The user is also given a list of topics in grey font in the background of the field where the selected topic can be typed. A ‘Submit’ and ‘Clear’ button is situated immediately under the topic field because it is the action that follows immediately after the selected topic is typed into its field and is therefore placed where it can be expected to be found for the sake of ease of use.

Fig 11.2. Screenshots the Dashboard activity following its selection from the main activity at **Fig 10**. On the Dashboard, the user’s bike preference by bike-type is displayed atop the page along with the ‘current status’ of his or her preferred bike. The Total Bikes rented is shown in a different, larger font so that it catches the user’s attention. ‘Notify On’ and ‘-Off’ buttons also feature on this page next to the Chat button that provides a short cut to the Chat-bot in case the user has questions about a certain feature of the app that information is not provided for on the Dashboard.

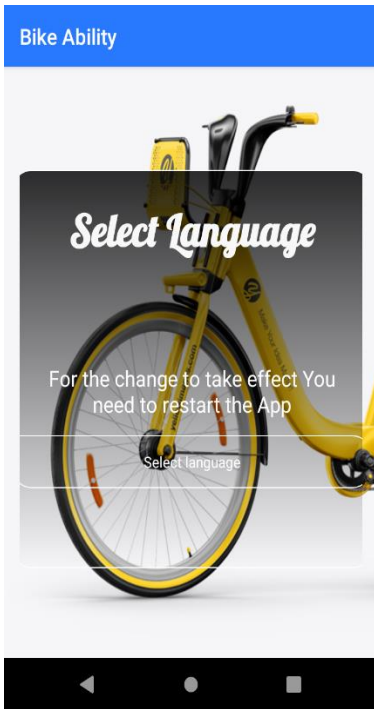


Fig. 12.

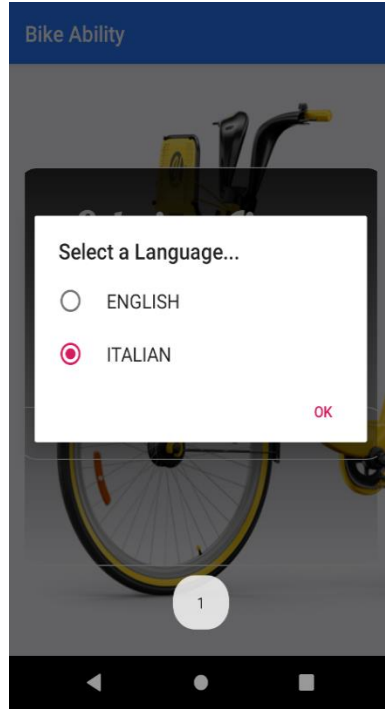


Fig 12.1

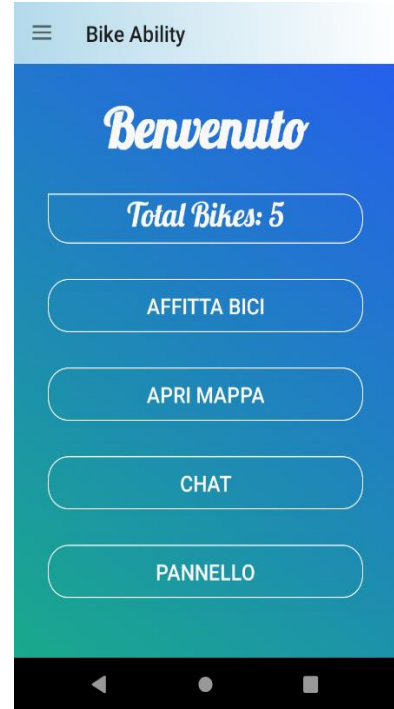


Fig 12.2

Fig 12 Screenshots the select language function that can be chosen from the hamburger menu on the home activity. The user is informed to restart the App after initializing another language preference. The title of the page is written in white at the top of the page for navigation purposes, to let the user know exactly what the page is about. After opting to select a language on the page at **Fig 12**, the user is directed to the page at **Fig. 12.1**. **Fig 12.2** is a screenshot of the welcome page after a language selection has been made and the app has been restarted, whereafter logging into the app will take place in the selected language. This also gives the app a more customized and personal feel.

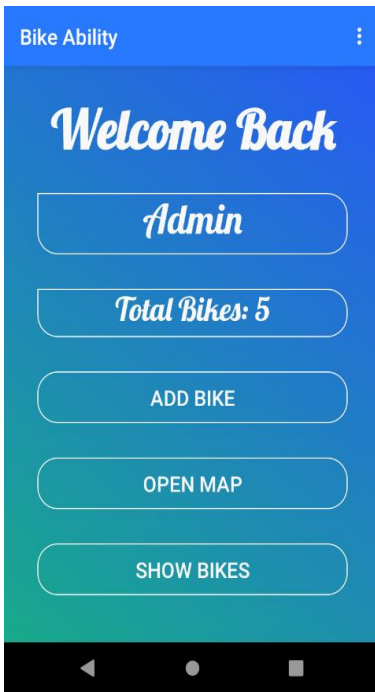


Fig. 13. Admin User

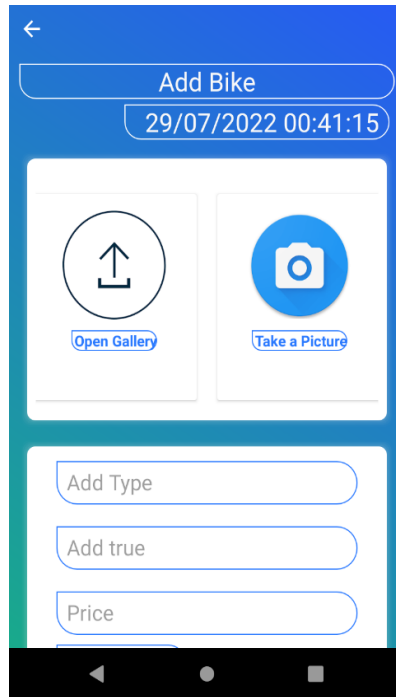


Fig 13.1 Add Bike Function

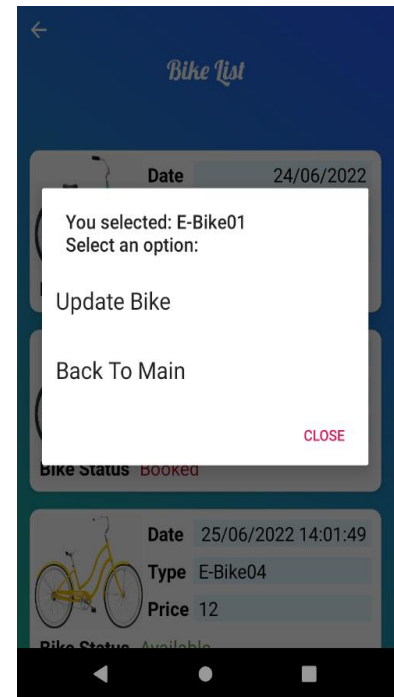


Fig 13.2 Pop Up Menu

Fig 13 is a screenshot of the Admin user welcome page that welcomes the Administrator back after successfully logging in with his or her credentials. Administrators can access more or less the same list as the user, when clicking on the ‘Add bike’ button that will allow him to choose a bike that he needs to modify the information of, such as changing the bike picture.

Fig 13.1 Screenshots the page that the user is directed to after selecting the ‘Add bike’ activity on the main page. The Admin User has the option to upload bike images via the gallery of the device that is displayed with a universally recognized upload icon. Alternatively, the Admin may choose to take a photo of a new bike whereby the device camera is directly accessible through the ‘take a photo’ button provided for on this page.

Fig 13.2 The selected bike is confirmed in the form of a feedback pop up message that confirms to the Admin which bike s/he selected. A shortcut is provided to go back to the main page at.

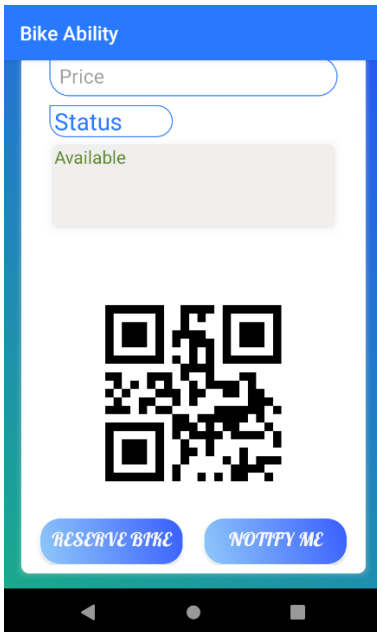


Fig. 14. QR Code for Bike Reservation

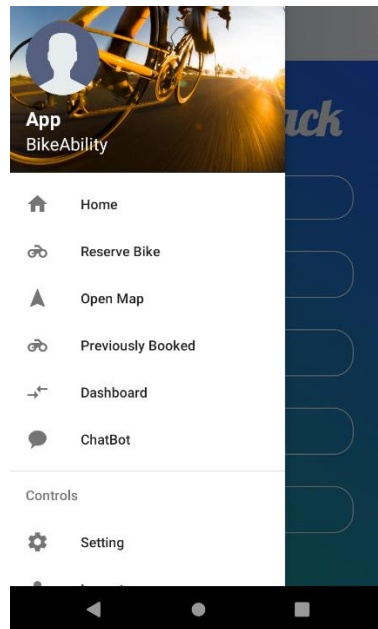


Fig 14.1 Navigation drawer

Fig 14 is a screenshot of a QR code that is generated after the successful reservation of a bike where this code saves in the gallery of the device automatically. The QR code can then be used at the bike story to unlock the bike. This ensures that no other user will be able to unlock the bike and feeds into the main utility of the app.

Fig 14.1 A photo can be uploaded by the user and the user's avatar will appear on the left-hand top corner of the navigation drawer. The font, size and choice of characters on the navigation drawer were kept simple against a white back ground, so as not to overwhelm the user as there are a number of activites, including 'settings' on that menu. The navigation drawer is scrollable.

IMPLEMENTATION

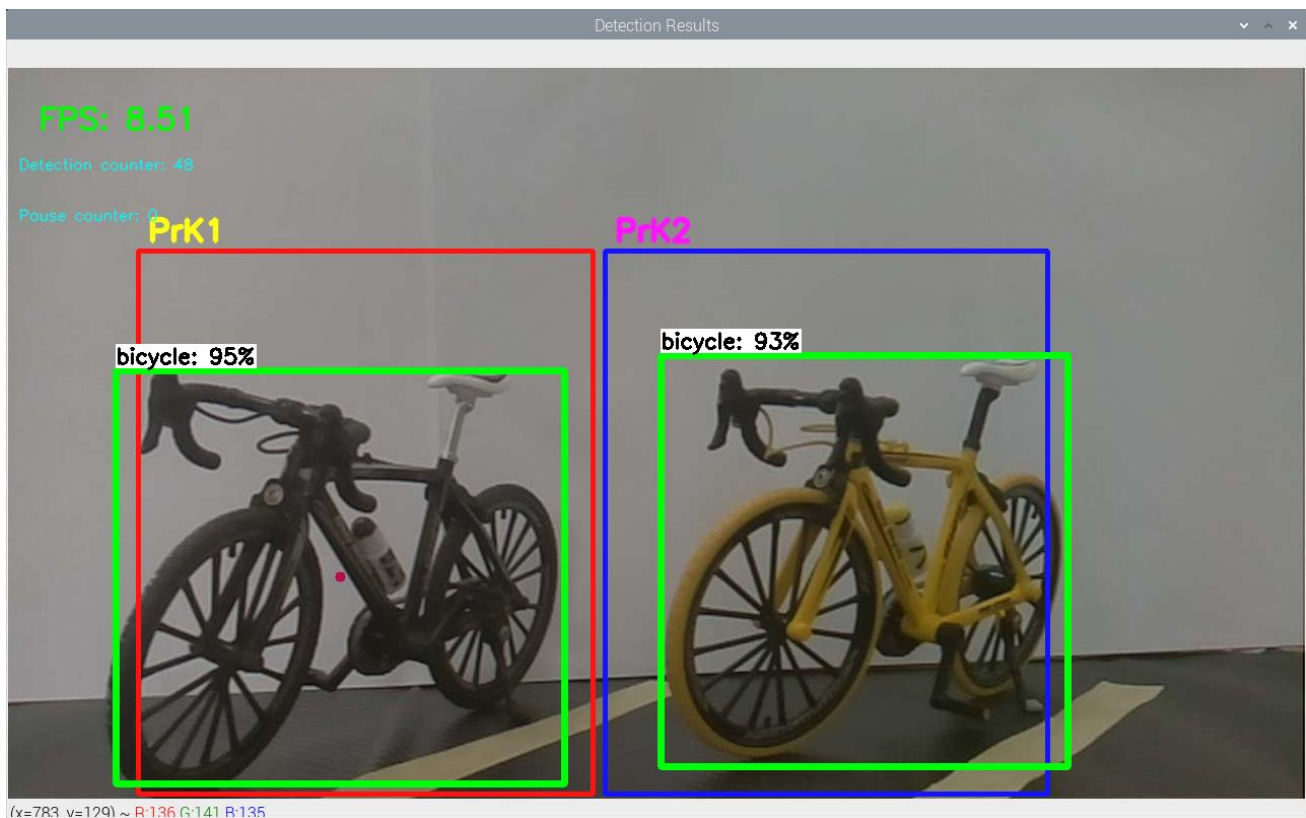


Fig. 15. Raspberry Pi Detection

Fig 15 details what is captured by the camera. The parking slots ROI are visible with bicycles in each frame outlined by different colors with OpenCV working in conjunction with TensorFlow.

OpenCV is responsible for drawing the bounty box and ROI. At first, the ROI is an unfiltered area without having been assigned a classification object. Once classified, with an object detected in it, it becomes a 'bounty box'. TensorFlow is pre-trained and loaded image classification that contain the classified images list.

For this project, the ROIs are static parking lots PrK1 and PrK2 and the bounty boxes are the green boxes drawn by OpenCV around the object detected in this case, bicycles. When a bicycle is detected inside the ROI, the detection counter is initialized, and when it reaches 50fps, Firebase is update with the new bike detected. In turn this process triggers the notification that are activated by the user through the "notifyMe" button. In order to avoid overloading firebase, there is a pause counter that works on the same principle as the frame counter and firebase is only updated when new bicycles are detected.

```

# Detect bike inside parking spot, |
if object_name == ("bicycle") and (pause == 0): #
    x = int(((boxes[0][1] + boxes[0][3]) / 2) * IM_WIDTH)
    y = int(((boxes[0][0] + boxes[0][2]) / 2) * IM_HEIGHT)
    # use Open Cv to Draw a circle at center of objects
    cv2.circle(frame, (x, y), 5, (75, 13, 180), -1)

    ###Parking_num_1
    # If object is in inside box, increment PrK1 counter variable
    if ((x > TL_parking1[0]) and (x < BR_parking1[0]) and (y > TL_parking1[1]) and (y < BR_parking1[1])):
        prk_spot01 = prk_spot01 + 1

    ###Parking_num_2
    # If object is inside box, increment PrK1 counter variable
    if ((x > TL_parking2[0]) and (x < BR_parking2[0]) and (y > TL_parking2[1]) and (y < BR_parking2[1])):
        prk_spot02 = prk_spot02 + 1

```

Fig. 16. Code for Detection

```

# If bike has been detected inside parking space for more than 50 frames, send a notification to the user.
# and send a notification to the phone.
if prk_spot02 > 50:
    detected_prk_spot02 = True
    # Initialize counters
    prk_spot02 = 0
    #Update Firebase
    firebase.put(userId, 'detectedStatus', 'Available')
    firebase_parkSpace02.put(parkingSpace02, 'add_Date', date)
    firebase_parkSpace02.put(parkingSpace02, 'add_Type', "E-Bike02")
    firebase_parkSpace02.put(parkingSpace02, 'add_Gear', True)
    firebase_parkSpace02.put(parkingSpace02, 'add_Price', "12")
    firebase_parkSpace02.put(parkingSpace02, 'place_Image',
        "https://firebasestorage.googleapis.com/v0/b/bikeability-672a7.appspot.com/o/Bike%2F1628324614264.png?alt=media&token=8262ae87-f7b6-404c-a16a-178b1afdb9ba")
    firebase_parkSpace02.put(parkingSpace02, 'keyId', "--My9dV9So0wV-tS7C6S")
    # Pause bike detection
    pause = 1

```

Fig. 17. Detection and send notification

The two code snippets, **Fig 16** and **-17** above demonstrate how the Tensor Flow Lite file that contains bicycle image classifications is accessed in order to check that a bike is available in the parking slot. If the bike is found inside the parking slot (ROI) the system starts to count the number of frames. This is to ensure that passing bike or bicycles crossing the camera on the move are not included in the detection frame. If the bicycle remains after 50 frames counted, then the code in Fig...is triggered. This is how the bike is counted as detected and this in turn triggers user notifications if user activated notifications.

OpenCV

OpenCV set the regions of interest for TensorFlow as rectangular boxes around the bicycles only.

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem menuItem) {

    int id = menuItem.getItemId();

    if (id == R.id.home) {
        HomeFragment fragment = new HomeFragment();
        FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.frame_layout, fragment, "Home");
        fragmentTransaction.commit();
    } else if (id == R.id.drw_re_Bike) {
        Intent intent_custm_lis = new Intent(getApplicationContext(), MainActivity.this, CustomerRentBikeList.class);
        startActivity(intent_custm_lis);
    } else if (id == R.id.drw_openMap) {
        MapFragment fragment = new MapFragment();
        FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction()
            .replace(R.id.frame_layout, fragment, "Open Map").addToBackStack(null);
        fragmentTransaction.commit();
    } else if (id == R.id.drw_Prev_Booked) {
        Intent intent_shared_loc = new Intent(getApplicationContext(), MainActivity.this, CustomerPreviousBooking.class);
        startActivity(intent_shared_loc);
    } else if (id == R.id.drw_dashboard) {
        Intent intentDashboard = new Intent(getApplicationContext(), MainActivity.this, BikeInfoActivity.class);
        startActivity(intentDashboard);
    } else if (id == R.id.setting) {
        Intent intent_change_lang = new Intent(getApplicationContext(), MainActivity.this, SettingsActivity.class);
        startActivity(intent_change_lang);
        //Open ChatBot
    } else if (id == R.id.drw_chatBot) {
        Intent intent_chatBot = new Intent(getApplicationContext(), MainActivity.this, ChatBotActivity.class);
        startActivity(intent_chatBot);
    } else if (id == R.id.logout) {
        logoutUser();
        return true;
    }
    mDrawerLayout.closeDrawer(GravityCompat.START);
    return true;
}
```

Fig. 18. Navigation Drawer

The navigation drawer in the screenshot allows the user access to the navigate the activities on the navigation drawer upon clicking them.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splash);
    //Hide bar
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
    //calling the splashScreen method
    splashScreen();
    //Logo Animation
    imageLogoView = (ImageView) findViewById(R.id.imageLogoView);
    fromtop = AnimationUtils.loadAnimation(this, R.anim.fromtop);
    imageLogoView.setAnimation(fromtop);
}

public void splashScreen() {
    Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            startActivity(new Intent(this, LoginActivity.class));
            finish();
        }
    }, DELAY_MILLISECONDS);
}
}

```

Fig. 19. Splash Screen – an animation that loads soonest the application is launched.

The Constructor called ‘handler ()’ creates a new object of data type that provides a 3 second window before moving to the Login Activity. This is done for the purpose of actually seeing the introductory animation to run for enough time for the user to see it, which is purely an aesthetic or UX, feature.

```

private void loginEmailPasswordUser (String mEmail, String pwd) {
    progressBar.setVisibility(View.VISIBLE);

    if (mEmail.equals("") || pwd.equals("")) {
        Toast.makeText(this, "Please fill in required fields", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.INVISIBLE);
    } else if (!Patterns.EMAIL_ADDRESS.matcher(mEmail).matches()) {
        Toast.makeText(this, "Please enter a valid Email", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.INVISIBLE);
    } else if (pwd.length() < 6) {
        Toast.makeText(this, "Password is too short. Minimum of 6 Characters Long.", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.INVISIBLE);
    }
}

```

Fig. 20. Validation of Login activity

Here login credentials are fulfilled. At this stage it is checked that the email address, password with certain characteristics are not empty fields and that these details are in valid format, for example that the email address contains an @ character and the password is more than 6 characters long.

```

//Create account and check if field are empty
createAcctButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
    username = Objects.requireNonNull(userNameEditText.getText()).toString().trim();
    //Alternative Validation whit TextUtil
    if (TextUtils.isEmpty(username)) {
        userNameEditText.setError("Username cannot be Empty");
        progressBar.setVisibility(View.INVISIBLE);
    }
    //Note EditText require: "getApplicationConteXt" and Not "this,"
    else if (!Patterns.EMAIL_ADDRESS.matcher(emailEditText.getText().toString()).matches()) {
        //else if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        Toast.makeText(getApplicationContext(), "Please enter a valid Email", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.INVISIBLE);
    } else if (passwordEditText.length() < 6) {
        Toast.makeText(getApplicationContext(), "Password is too short. Minimum of 6 Characters Long.", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.INVISIBLE);
    } else if (myTrackingNumEditText.length() == 0) {
        Toast.makeText(getApplicationContext(), "Phone Number cannot be Empty.", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.INVISIBLE);
    } else {
        //upload data to Firebase
        final String username = userNameEditText.getText().toString().trim();
        final String email = emailEditText.getText().toString().trim(); //Final make the variable immutable example: final String s = "s"; s = "a"; // illegal
        final String password = Objects.requireNonNull(passwordEditText.getText().toString().trim());
        final String myTrackingNum = Objects.requireNonNull(myTrackingNumEditText.getText().toString().trim());

        //Show ProgressBar
        progressBar.setVisibility(View.VISIBLE);
        FirebaseAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
@Override
public void onComplete(@NonNull Task<AuthResult> task) {
    if (task.isSuccessful()) {
        //Clear input fields
        emailEditText.setText("");
        passwordEditText.setText("");
        myTrackingNumEditText.setText("");
        userNameEditText.setText("");

        //Add Fields to Firebase
        Users user = new Users(
            username,
            email,
            myTrackingNum
        );

        FirebaseDatabase.getInstance().getReference().child(Objects.requireNonNull(FirebaseAuth.getInstance().getUid()))
            .setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
@Override

```

Fig. 21. Create Account

When an account is created it is necessary for all fields to be fulfilled for example that a valid email is and the password has specified characteristics that will avoid sending null values to Firebase that would otherwise the user be found in Firebase.

```

firebaseDatabase.addValueEventListener(new ValueEventListener() {
@Override
public void onDataChange(DataSnapshot dataSnapshot) {
    BikeInfoObj userProfile = dataSnapshot.getValue(BikeInfoObj.class);
    assert userProfile != null;
    selectedBike = userProfile.getSelectedBike();
    detectedStatus = userProfile.getDetectedStatus();
    currentParked = userProfile.getCurrentParked();
    if (selectedBike != null && selectedBike.equals("E-Bike Adult") && currentParked.equals("Parked") && detectedStatus.equals("Available")) {
        Notification notification = new NotificationCompat.Builder(getApplicationContext(), CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_bike_adult)
            .setContentTitle("E-Bike Adult")
            .setContentText("Your Reserved Bike is Available")
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setCategory(NotificationCompat.CATEGORY_MESSAGE)
            .build();
        notificationManager.notify(1, notification);
    }
}
}

```

Fig. 22. Set notification

Fig 22 demonstrates the method that trigger the notifications that requires three conditions to be satisfied, namely that there has to be a bike available, it has to be detected in the parking space and it as to be an adult bike. When Raspberry Pi detects the bike inside the box, it sends these three conditions to Firebase, triggering the notification to be sent to the Renting customer.

```

//Save Data on Click button
reservBtn = (Button) findViewById(R.id.reservBtn);
reservBtn.setOnClickListener(v -> {

    final String add_Date = textViewDate.getText().toString().trim();
    final String bike_Type = editTextType.getText().toString().trim();
    final String bike_Price = editTextPrice.getText().toString().trim();
    final String etComment = editTextComment.getText().toString().trim();

    if (etComment.equalsIgnoreCase("Booked")) {
        editTextComment.setError("QR-Code has Assigned to another user.");
        Toast.makeText(getApplicationContext(), "The Bike has already been reserved, Please choose another.", Toast.LENGTH_LONG).show();
    } else {
        // Else call bar code generator
        MultiFormatWriter multiFormatWriter = new MultiFormatWriter();
        try {
            BitMatrix bitMatrix = multiFormatWriter.encode(editTextType.getText().toString(), BarcodeFormat.QR_CODE, 500, 500); //checking editText
            BarcodeEncoder barcodeEncoder = new BarcodeEncoder();
            Bitmap bitmap = barcodeEncoder.createBitmap(bitMatrix);
            imageView.setImageBitmap(bitmap);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

Fig. 23. Reserve bike if not booked already, the generate a QR-Code

The reserve bike button initiates the reservation tool that initially validates whether the bike was previously booked. If the bike was previously booked the customer gets notified accordingly and another bike has to be chosen; if not, then a QR code is generated that is automatically saved in the customer's Gallery on his/her device.

```

private void updateAvailability() {
    updAvailabilityDatabaseRef = FirebaseDatabase.getInstance().getReference().child("Bike");
    storageReference = FirebaseStorage.getInstance().getReference().child("Bike");

    final String etComment = editTextComment.getText().toString().trim();
    //Compress image to avoid size limitation error in Firebase, change data format from Bitmap to Byte Array then compress to jpg and upload to Firebase.
    uploadImage.setDrawingCacheEnabled(true);
    uploadImage.buildDrawingCache();
    Bitmap bitmap = uploadImage.getDrawingCache();
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos);
    byte[] data = baos.toByteArray();
    uploadTask = storageReference.putBytes(data)
        .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                storageReference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        Query query = updAvailabilityDatabaseRef.orderByChild("keyId").equalTo(uri.toString());
                        query.addListenerForSingleValueEvent(new ValueEventListener() {
                            @Override
                            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                                for (DataSnapshot ds : dataSnapshot.getChildren()) {
                                    FirebaseUser user_db = firebaseAuth.getCurrentUser();
                                    final Users user = ds.getValue(Users.class);
                                    assert user_db != null; //Check for not null similar to if statement
                                    user_id = user_db.getId();
                                    ds.getRef().child("etComment").setValue("Booked");
                                    ds.getRef().child("user_id").setValue(user_id);
                                }
                            }
                        });
                    }
                });
            }
        });
}
}

```

Fig. 24. Update Availability

In the code snippet at Fig 24 the booked bike is identified and assigned to the particular user-ID who booked it. In that way over-booking is avoided.

```

private void customerPrevBookingList() {
    ///Access Firebase Database Parent Bike
    databaseReference = FirebaseDatabase.getInstance().getReference().child("Bike");
    databaseReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            bikeUploadsList = new ArrayList<>();
            for (DataSnapshot uploadSnapshot : dataSnapshot.getChildren()) {
                /// This method load the bike list previously booked bike the current (userID)
                FirebaseAuth user_db = firebaseAuth.getCurrentUser();
                BikeObj userBikeObject = uploadSnapshot.getValue(BikeObj.class);
                assert userBikeObject != null;
                assert user_db != null;
                userID = user_db.getUid();

                ///Try and catch to avoid nullpoint exception
                try {
                    if (userBikeObject.getUser_id().equals(userID)) {
                        userBikeObject.setKeyId(uploadSnapshot.getKey());
                        bikeUploadsList.add(userBikeObject);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            ///Invoke the adapter
            CustomerPreviousBookingAdapter = new CustomerPreviousBookingAdapter( bikeUploadsList, CustomerPreviousBooking.this, bikeUploadsList);
            recyclerView.setAdapter(CustomerPreviousBookingAdapter);
            CustomerPreviousBookingAdapter.setOnItemClickListener(CustomerPreviousBooking.this);
        }
    });
}

```

Fig. 25. Customer Previous list

In the code screenshot at **Fig 25** the main list of bikes is read, looped and filtered by user-ID. If a bike is found against the user’s ID, the booked bike appears in the user’s previously booked bike list.

```

requestBtn.setOnClickListener(new View.OnClickListener() {
    @SuppressWarnings("setTextI18n")
    @Override
    public void onClick(View view) {
        if (userRequest.getText().toString().equalsIgnoreCase("hi")) {
            botAnswer.setText("Hello, I am the BikeAbility-Bot. A Pleasure to meet you. I can assist you with extra information about BikeAbility.\n\n" +
                "Please choose and type one of the following topics that I can assist you with.\n\n" +
                "Topics: Subscriptions; Emergency; Payments; Notifications; Returning Bike; Stations; Traffic");
        }
        // Subscriptions info
        else if (userRequest.getText().toString().equalsIgnoreCase("Subscriptions")) {
            botAnswer.setText("Oh so you want information about " + userRequest.getText().toString() + ". Here are a number of key information point about Subscription:\n\n" +
                "Bike ability currently has two types of subscriptions. \n" +
                "\tYearly Subscription\n" +
                "\tThree day ticket\n" +
                "\tIf you still have another type of subscription problem other than those listed here, then you can call customer service on 0871212315 and choose 3 from the phone menu");
        }
        // Emergency
        else if (userRequest.getText().toString().equalsIgnoreCase("Emergency")) {
            botAnswer.setText("Oh so you want information about " + userRequest.getText().toString() + ". Here are a number of key information point about Emergency:\n\n" +
                "When you encounter an emergency such as a flat tire, in case of a bicycle accident or any physical fault with the bicycle, please contact 0871212315\n\n" +
                "Is there is another topic I can give you information about? If so, Please type your additional topic in the 'topic here' field above");
        }
    }
});

```

Fig. 26. Chat bot

The code snippet at Fig... details the sequence of ‘if’/ ‘else’ statements used in the creation of the chat bot. Here, a user can choose different chosen topics and type them in the comment space available to which the chat bot algorithm then returns information on the user’s chosen topic.


```

private void checkInternetConnection() {
    // Invoke Connectivity Manager object to check connection
    ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
    boolean isConnected = activeNetwork != null &&
        activeNetwork.isConnectedOrConnecting();
    // Check for network connections
    if (isConnected) {
    } else {
        Toast.makeText(this, "No Internet Connection available", Toast.LENGTH_LONG).show();
    }
}
}

```

Fig. 27. Check Internet Connection

Internet connectivity is verified on the login page, the code for which process is displayed in the image at Fig 27. If there is no internet connection, a message notifies that customer accordingly, as the functioning of the app needs internet connectivity. Fig 28 shows GPS and location settings are checked in the same manner.

```

/////Show GPS alert
private void showSettingAlert() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("GPS setting!");
    builder.setMessage("GPS is not enabled, Please enable GPS in the settings menu ");
    builder.setPositiveButton("Setting", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialogInterface, int i) {
            com.bikeability.AdminPage.this.startActivity(new Intent("android.settings.LOCATION_SOURCE_SETTINGS"));
        }
    });
    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.cancel();
        }
    });
    try {
        builder.show();
    } catch (InflateException unused) {
    }
}
}
}

```

Fig. 28. GPS

TESTING

The importance of testing cannot be overstated in its function reduce the incidence of bugs in the application. (Android APP Testing Tutorial with Automation Framework, 2021)

As vital component of the project, the app was tested throughout the different phases of the project, at the end of each milestone reached. Performed on JUnit and Espresso an integrated framework for testing in Android, each test showed shortcomings of both process and product that were improved upon with each round of testing.

TEST DRIVEN DEVELOPMENT, TEST CASES

As a [TDD](#) strategy was pursued with the execution of this project, the following tests among others were drafted before the beginning of the writing of code. Doing it this way ensured smoother, bug free code. It involves writing out problem/domain analysis ([PDA](#)) where the problem is stated together with the conditions under which that problem is solved. In other words, those conditions are essentially then the ones under which the test passes. The problems in the PDA

Create Account

Test Case: Manual testing of the 'Create Account' activity
Title: Create Account Activity
Description: Registration of user account takes place through the Create Account activity, where user is required to furnish particulars that will be stored on Firebase for later authentication when logging in to the app.
Preconditions: The app is downloaded and installed. User registration details including e-mail address and password must be unique. No other user can be registered with the same details.
Assumption: The user device is online and Bike-Ability is installed on the device
Test Procedure: <ol style="list-style-type: none">1. Click on the app icon where it is appears on device to launch it2. The user enters personal details into fields prompted by the app3. The user inserts his unique e-mail address into the 'e-mail' field4. The user enters a unique password into the 'password' field5. The user clicks on the 'Create Account' button
Expected Result: The 'Create Account' activity contains and makes available to the user all fields that are needed to register the user's details on Firebase
Status: Test passed

Login

Test Case: Manual Testing of User Login Activity

Title: User Login Activity

Description: A registered user successfully logs in to Bike Ability

Precondition: The user has the application installed and launched on the testing device
The User's unique e-mail address and password that was registered during the account creation activity are now used again to login to the account.

Assumption: The user enters the correct details that matches the registered details.

Test Procedure:

1. Launch application by clicking on the app icon where it appears on the device
2. The user inserts the email address into the 'e-mail address' field that was used at the account creation stage
3. The user inserts the 'password' into the 'password' field
4. The clicks on the button 'Log In' button
5. User credentials are verified against the details saved for the account in Firebase

Expected Result: The 'LogIn' activity contains and makes available to the user all fields that need to be filled in and authenticated in order for user to access main activities of the app

Status: Test passed

Administration Inventory Management

Test Case: Admin add a new Bike
Title: Add a New Bike
Description: Admin adds a new bike image to the list of available bikes
Precondition: The App Administrator's must be online, have the app launched and be authenticated and logged in as Admin
Assumption: The user is logged in.
Test Steps: <ol style="list-style-type: none">1. Select the 'add Bike' from admin's home activity2. Choose and select 'Upload from Gallery' or 'Take Picture'3. A Toast message displays after successful upload or error
Expected Result: A new bike image should be successfully added to the list of available bikes
Status: Test passed.

Functional Testing

Once the Coding commenced to a level where all the components of the app, meaning all of the [FRS](#) were coded for, a series of test on each of the functions listed on the FRS, such as Login and Registration; Available Bike Detection; Reserve Available Bike; Cancel Reservation, etc were performed. These tests took place in the form of both Unit and finally Integration Testing.

Unit Testing

“Unit testing is a testing method where individual units of code are tested.” (Mike van Drongelen, Android Studio Cookbook)

The correct functioning of the apps and the performance of its components are verified through Unit Testing. And, for that purpose, J-Unit test library testing framework in Java that draws on direct support from Android Studio is used. Android Logcat helped with detecting errors and the troubleshooting.

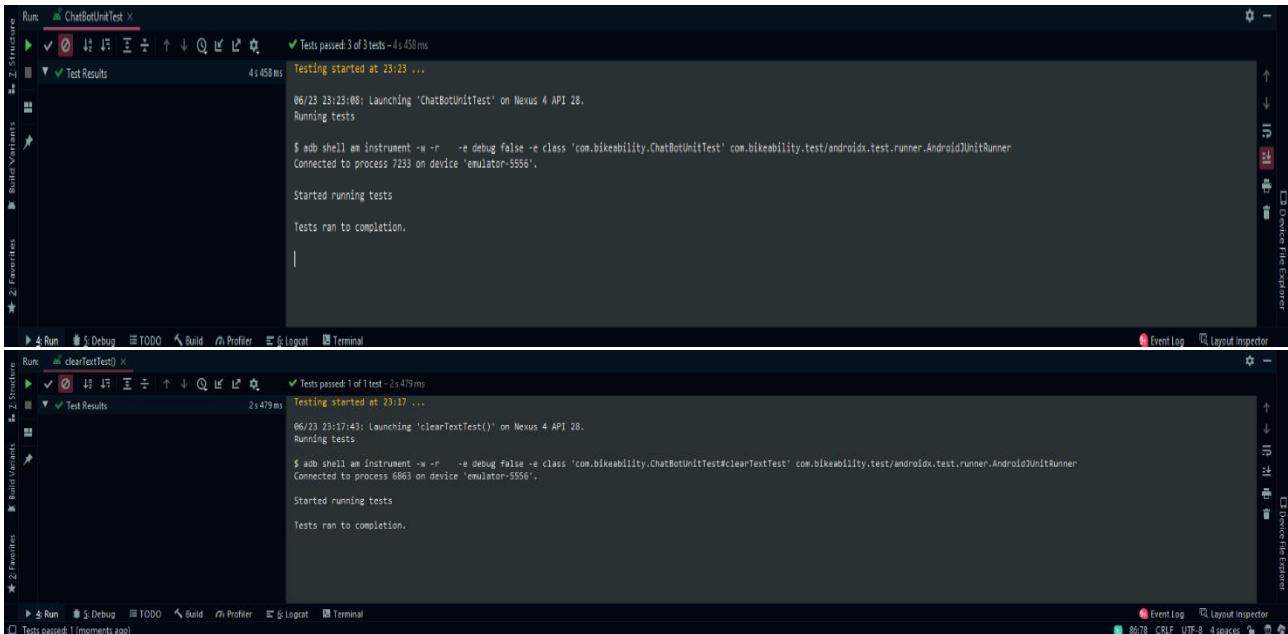


Fig. 29. Chatbot

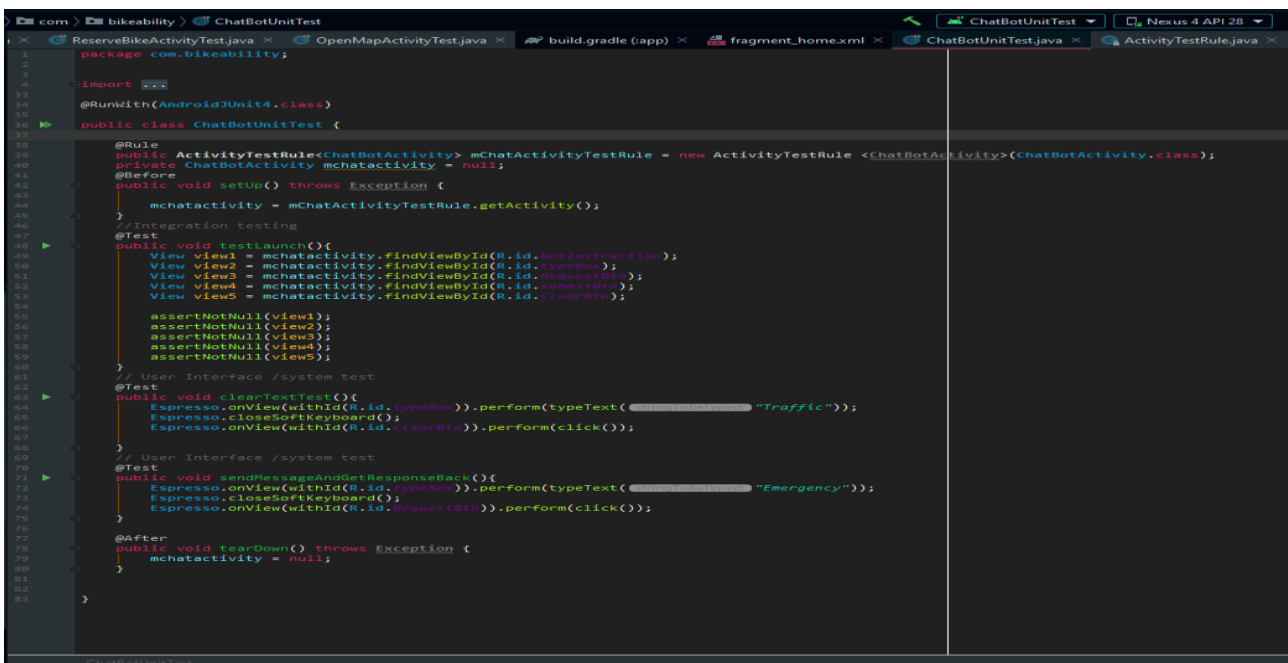


Fig. 30. ChatBot UnitTest

In the test imaged at **Fig 29** and **-30**, the Chatbot activity is launched and all buttons are loaded. Then on the clear test the word traffic is typed as a test word in the type box, and cleared via the ‘clear’ button again to test whether the ‘Clear’ function works to pass the test. These are actually two units of code that was tested. Another unit test involves the UI of the Chatbot, where it is checked whether a ‘request’ sent by the user gets the proper response displayed to the user; specifically, information on the topic of ‘emergency’ is requested by the user and the Chat Interface responds with the appropriate information on ‘emergencies’ that the user requested. The images at **Fig30** above demonstrates three such tests performed on the Chatbot interface.

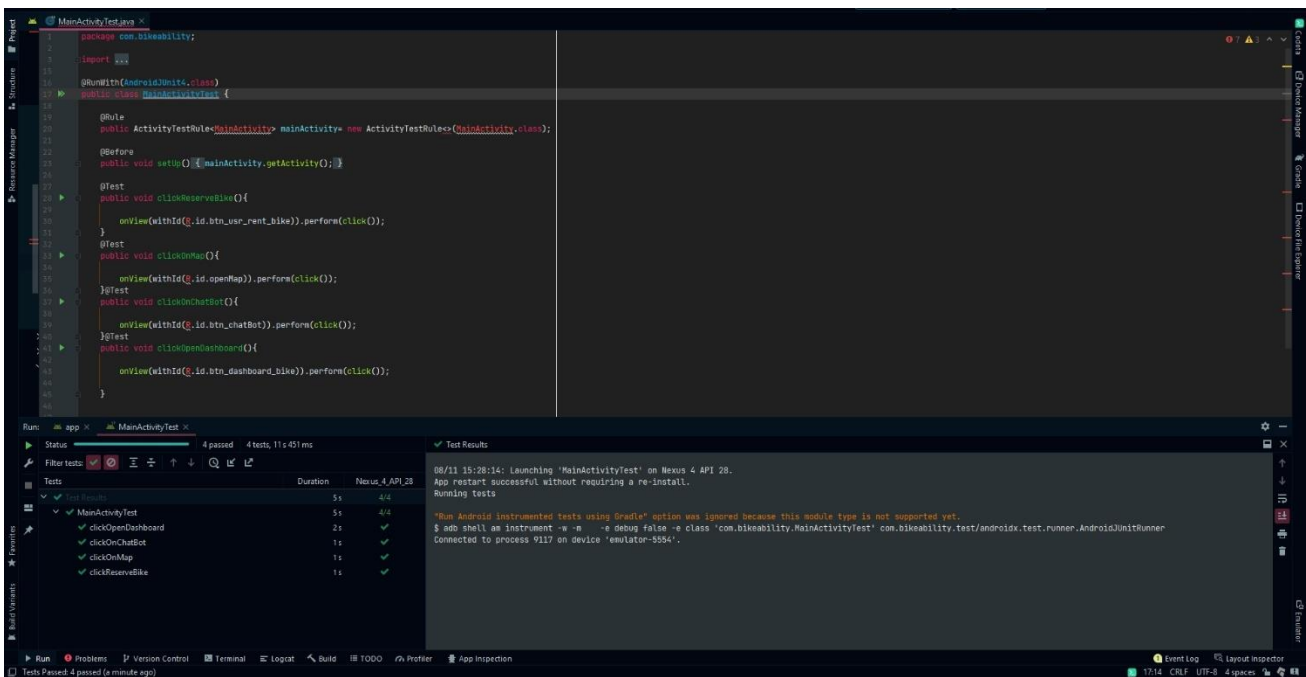


Fig. 31. Main Activity Test

Fig 31 above exhibits a screenshot of Main activity as Unit to be tested using J-Unit for the testing of the Main Activity. The activities button on the Main Activity was clicked on one by one and when they worked properly, the test passed.

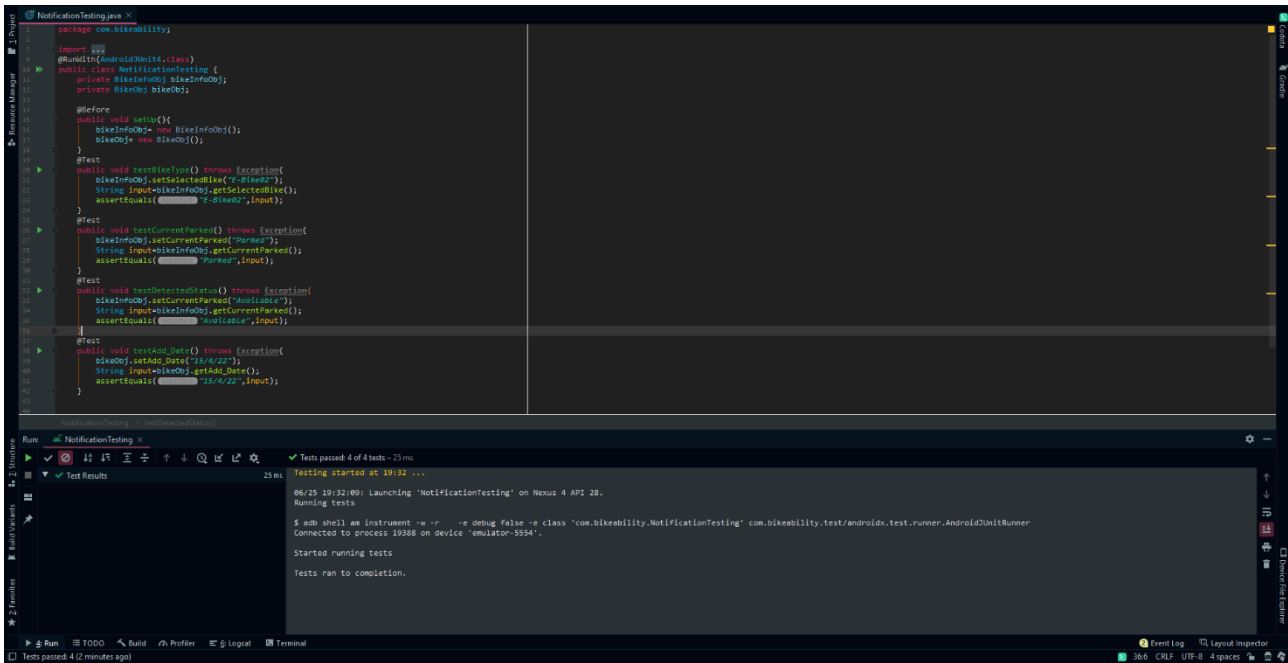


Fig. 32. Notification Test

Fig 32 above demonstrates the workings of the Notification button code, where if pressed, the user gets notified of the available bikes at his or her chosen bike station. Three conditions are satisfied before the notification is sent to the user regarding the type of bike, whether the bike is in the parking slot and whether it is available, since bikes can sometimes be in the parking slot, but not available. This test passes on meeting of the 3 conditions.

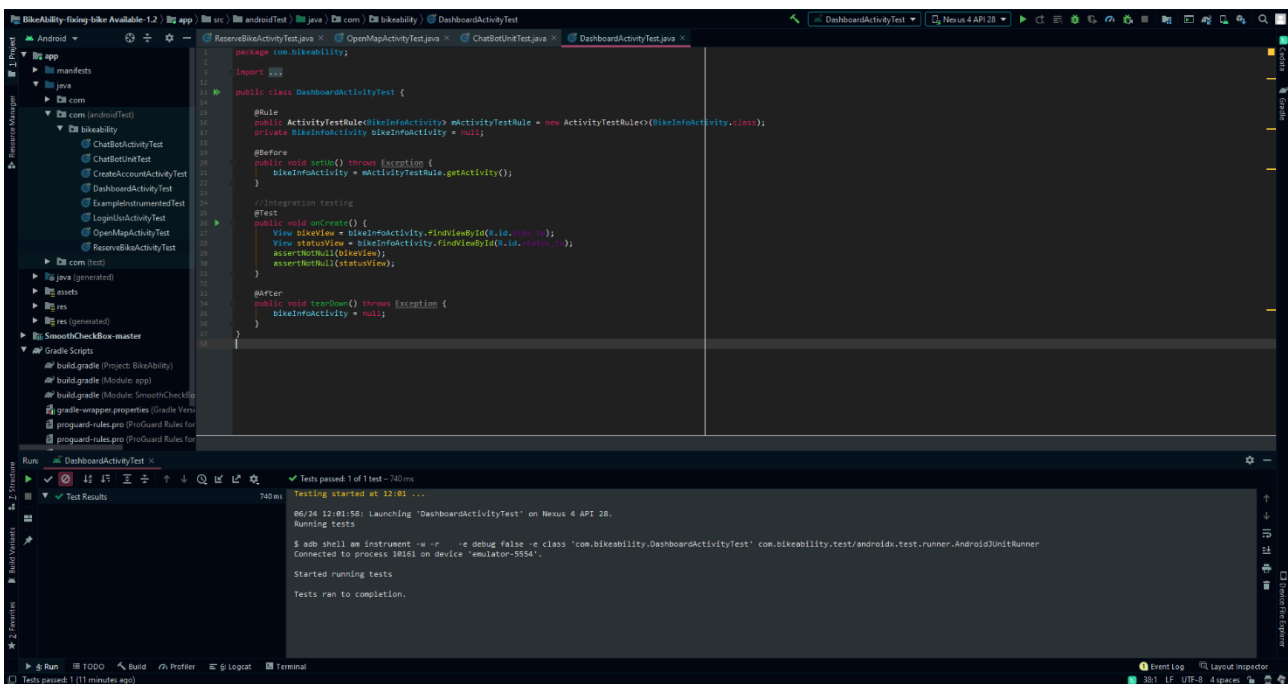


Fig. 33. A Unit test of the Dashboard activity that passes if the activity is accessed successfully.

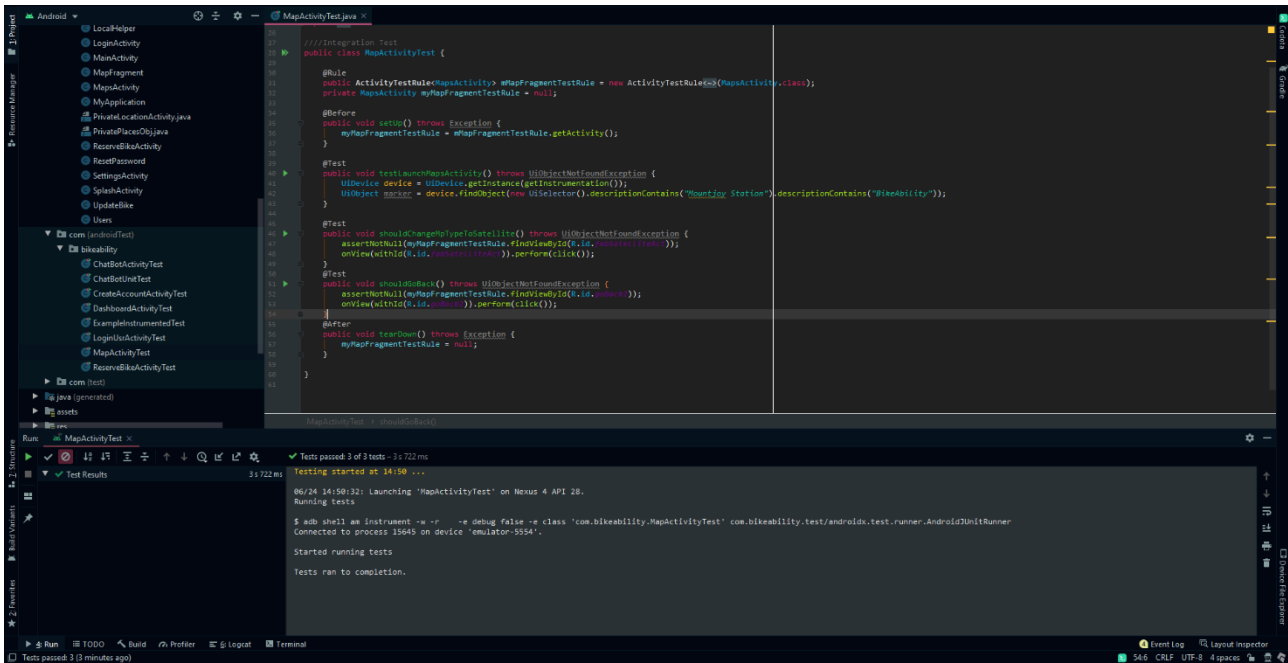


Fig. 35. Open Map Test

Cross-system functional testing was made possible through the UI testing framework Android UI Automator. (Android Developers, 2022)

This formed part of integration testing routine. And, for the particular test in the screenshot at **Fig 35** above, Switching between map views, i.e. Terrain and satellite views were tested on Google Maps that is loaded via an API, the external component that the system interacts with when the Map activity is loaded.

UI Testing with Espresso

The frontend User interface was tested using Espresso, the default testing suit that Android Studio includes. The ‘Record Test’ function was used to write the tests that emulate an end-users use of the application.

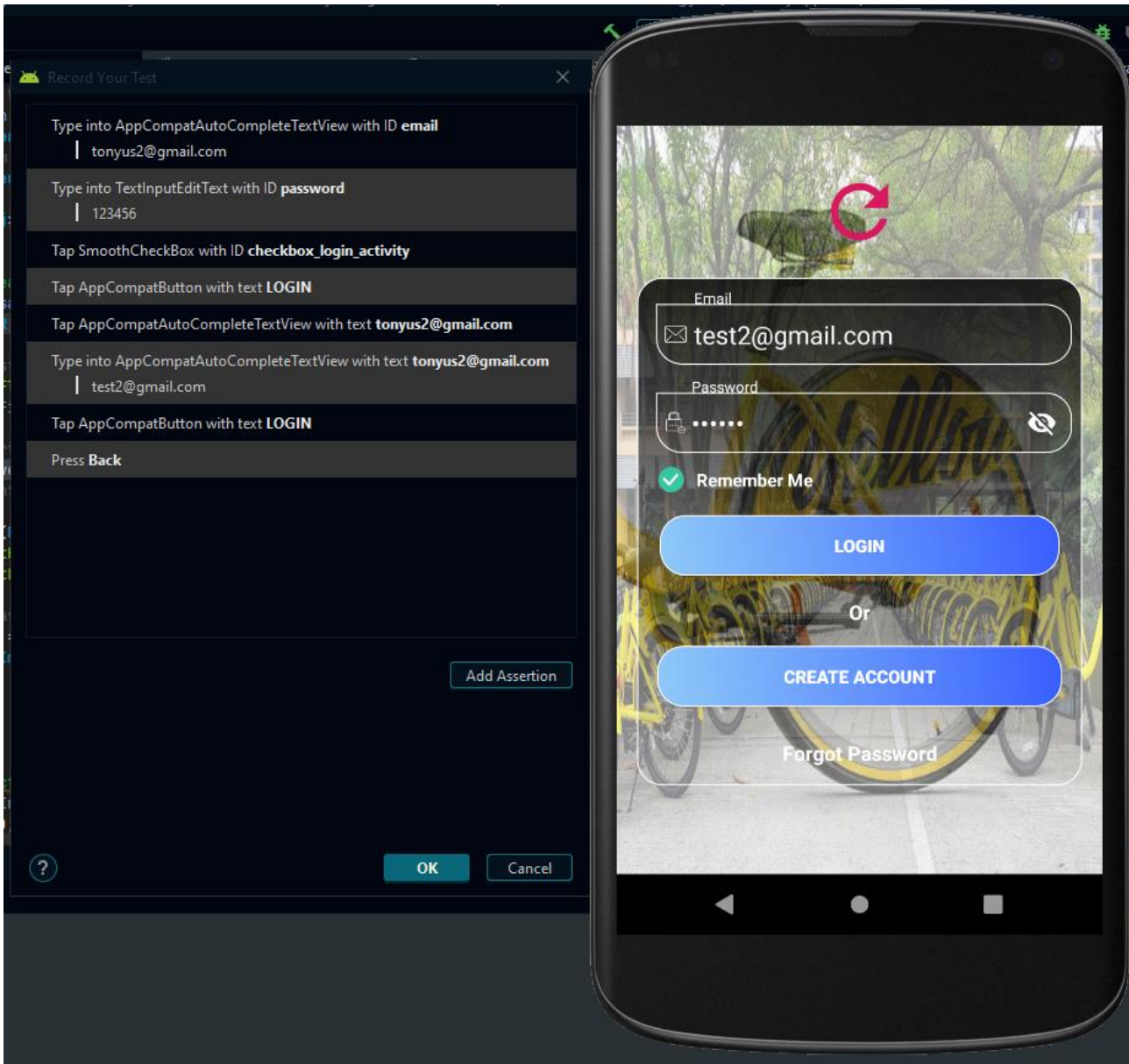


Fig. 36. Espresso Test set up

Fig 36 - After selecting the ‘record Espresso test option’ in Android Studio, the application launches on an emulator. The test is ready to be recorded via the ‘record your test’ function, that is used to update user interaction whether ‘Login’, or ‘Create an Account’ or any other use-case of the app.

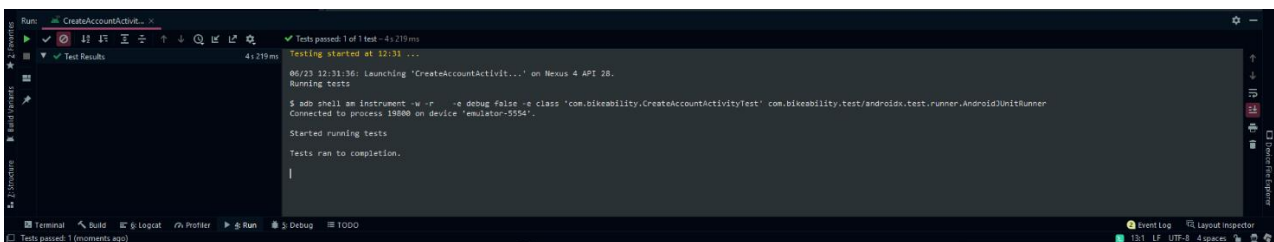


Fig. 37. Create Account

```

43
44 @LargeTest
45 @RunWith(AndroidJUnit4.class)
46 public class LoginUserActivityTest {
47
48     @Rule
49     public ActivityScenarioRule<LoginActivity> mActivityScenarioRule = new ActivityScenarioRule<>(LoginActivity.class);
50
51
52     @Test
53     public void loginUserActivityTest() {
54         ViewInteraction appCompatAutoCompleteTextView = onView(
55             Matchers.allOf(ViewMatchers.withId(R.id.email),
56                 childAtPosition(
57                     childAtPosition(
58                         withId(R.id.txtInputLayoutEmailLogin),
59                         0),
60                     0),
61                 isDisplayed()));
62         appCompatAutoCompleteTextView.perform(replaceText("test2@gmail.com"), closeSoftKeyboard());
63
64         ViewInteraction textInputEditText = onView(
65             allOf(withId(R.id.password),
66                 childAtPosition(
67                     childAtPosition(
68                         withId(R.id.txtInputLayoutLoginPass),
69                         0),
70                     0),
71                 isDisplayed()));
72         textInputEditText.perform(replaceText("123456"), closeSoftKeyboard());
73
74         ViewInteraction smoothCheckBox = onView(
75             allOf(withId(R.id.checkBox_login_activity),
76                 childAtPosition(
77                     childAtPosition(
78                         withId(R.id.email_login_form),
79                         2),
80                     0)));
81         smoothCheckBox.perform(scrollTo(), click());
82
83         ViewInteraction appCompatButton = onView(
84             allOf(withId(R.id.sign_in_button), withText("LOGIN"),
85                 childAtPosition(
86                     allOf(withId(R.id.email_login_form),
87                         childAtPosition(
88                             withId(R.id.loginLayout),
89                             1)),
90                     3)));
91         appCompatButton.perform(scrollTo(), click());
92     }
93
94     @private static Matcher<View> childAtPosition(
95         final Matcher<View> parentMatcher, final int position) {
96
97         return new TypeSafeMatcher<View>() {
98             @Override
99             public void describeTo(Description description) {
100                 description.appendText("Child at position " + position + " in parent ");
101                 parentMatcher.describeTo(description);
102             }
103
104             @Override
105             public boolean matchesSafely(View view) {
106                 ViewParent parent = view.getParent();
107                 return parent instanceof ViewGroup && parentMatcher.matches(parent)
108                     && view.equals(((ViewGroup) parent).getChildAt(position));
109             }
110         };
111     }
112 }

```

Fig. 38. Automated test using Espresso Login User Test

```

Run: LoginUserActivityTest
Tests passed: 1 of 1 test - 3s 97ms
Testing started at 11:52:28 ...
Test Results 3s 97ms
06/23 11:52:28: Launching 'LoginUserActivityTest' on Nexus 4 API 28.
Running tests
$ adb shell am instrument -w -r -e debug false -e class 'com.bikeability.LoginUserActivityTest' com.bikeability.test/androidx.test.runner.AndroidJUnitRunner
Connected to process 17908 on device 'emulator-5554'.
Started running tests
Tests ran to completion.

```

Fig. 39. Login user Activity Test

In the case of Fig 37, Fig38 and Fig39, 'Create Account' and 'Login' tests are recorded respectively and the recordings are used for further testing.

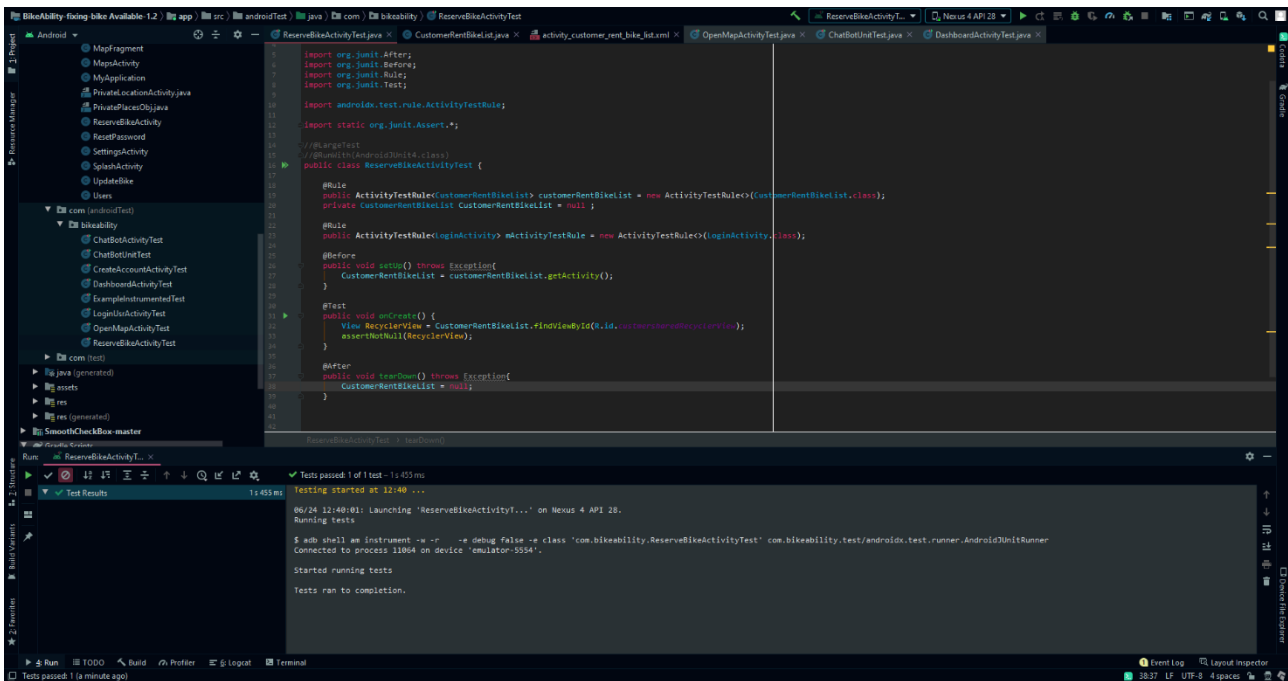


Fig. 40. Reserve Bike Test

Fig 40 - The Reserve Bike Activity is accessed after login. The test passes if the recycle view bike list is accessible. This is also a form of functional Testing as Reserve Bike Activity is on the list of Functional Requirements/Specifications.

EVALUATION

Mobile Phones with different Android versions ranging from Kit-Kat version 4 to Pie version 9 will be used to test the solution. Testing will be the most important phase of the project, because this will be the phase that determine whether the project is a success. Code will be tested step by step though, so that testing takes place throughout and intermittently to see if specific project goals that were set for specific times were meat.

The system will be evaluated by:

- Collecting the feedback from the users
- Make a survey and evaluate the result
- By monitoring the application periodically in order to find the weakness of the application
- By analysing and evaluating the application bugs and the frequency with which they appear

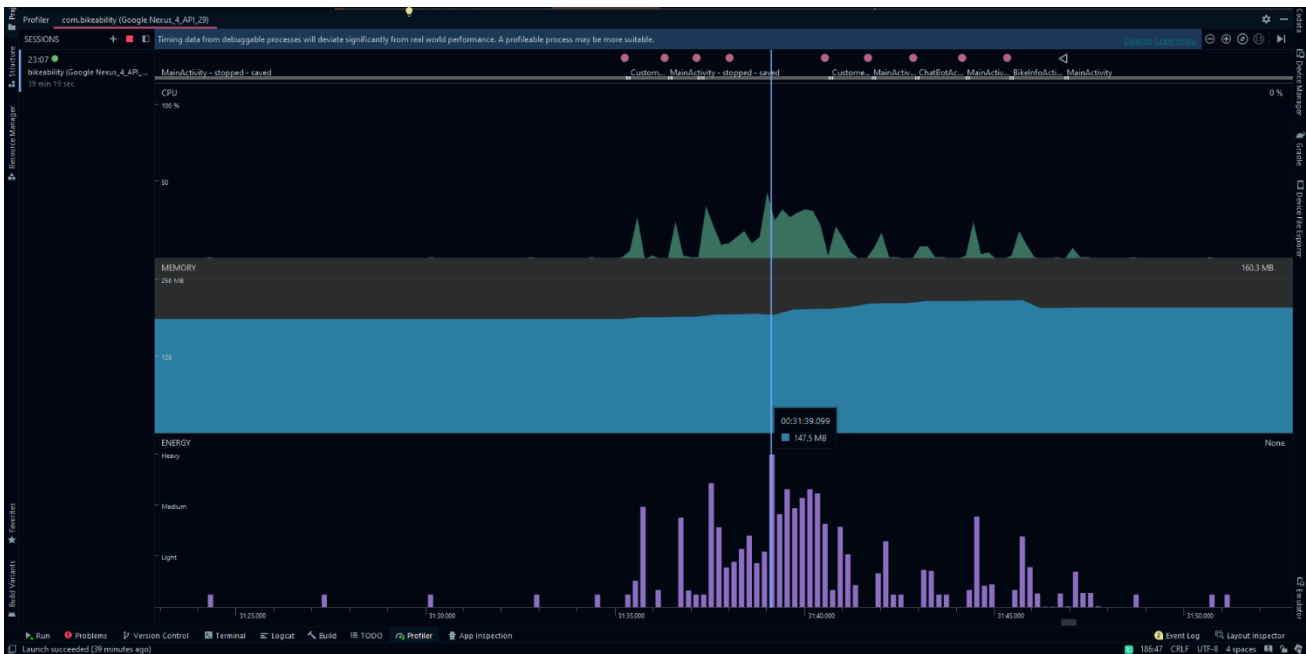


Fig. 41. Bike Ability Test

Fig 41 screenshots the real-time CPU, Network, Energy and Memory performance tested with the Android Profiler tool custom to android studio. The assessment was conducted on Nexus 4. All possible activities and -operation performance was tested. All components being tested resulted in minimum- and maximum performance values.



Fig. 42. Memory performance evaluation

Fig 42 When the memory performance was evaluated to see how much memory the app consumed while in use on the device, maximum memory consumption reached 160.3 MB while at minimum 101.3 MB was consumed. Native consumed the most memory that's why it would be curious to see how the app performs on Android KitKat, because it requires lower resources. Stack remained consistently low with a peak time value of 0.1 MB.

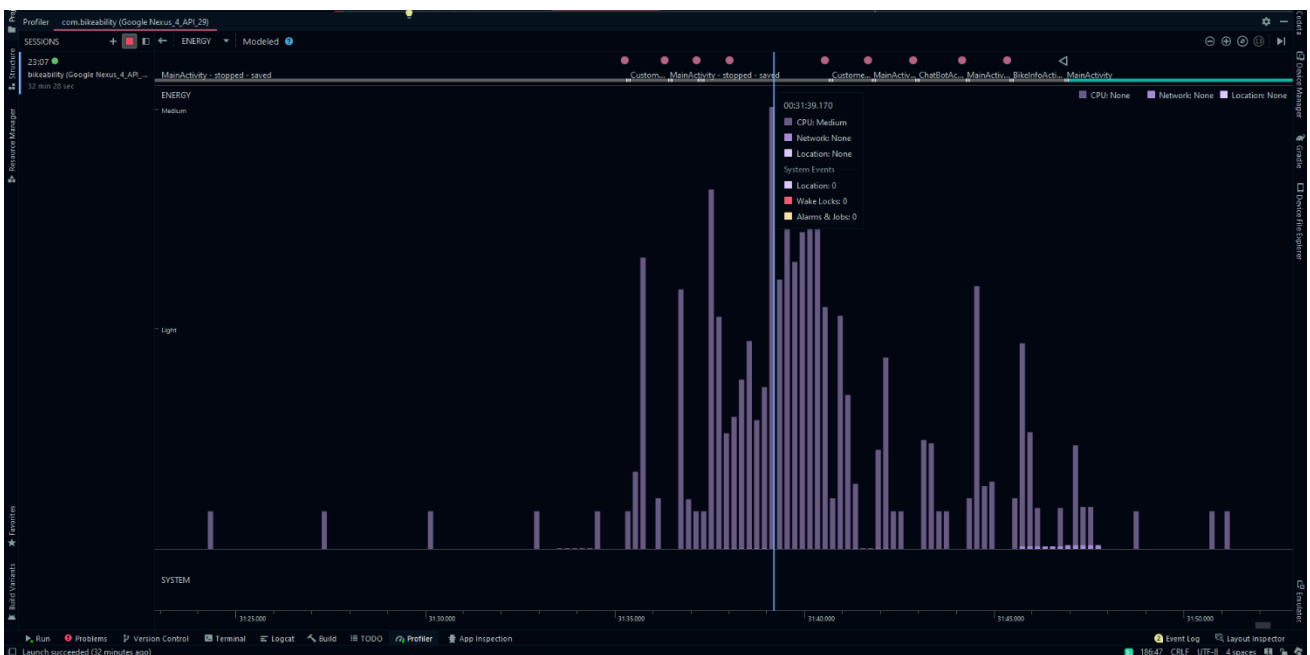


Fig. 43. Energy performance at maximum were medium and at a minimum light energy were consumed

Fig 43 Energy consumption is divided into 3 distributions namely, Heavy, Medium and Light. A medium amount of energy was the Maximum amount of energy used by running the App on a Nexus 8, while the least amounted to light consumption.

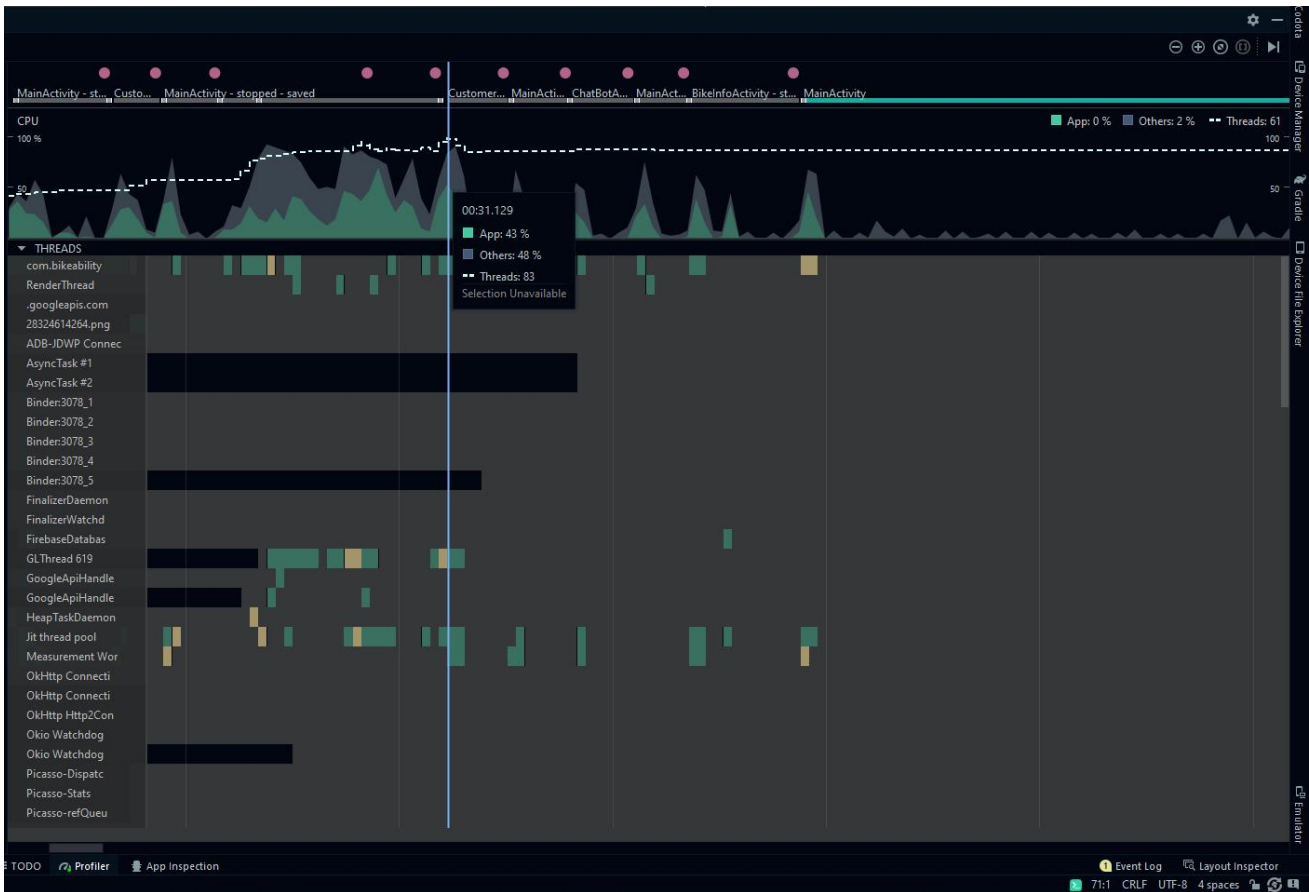


Fig. 44. CPU performance Evaluation

Fig 44 - With 63 active threads, the maximum CPU usage reached circa 43%. In this stress test all the activities were opened in rapid succession, to monitor the behaviour of the app. Considering the number of activities opened in such rapid succession this stress test showed the CPU usage is not high, considering that under normal usage with one activity opened, it would be around 20%.

FUTURE WORK

It is envisaged for the project deliverables to be scaled to include a large number of bike docking stations across Ireland and to expand map coverage. RCNN will be implemented to upscale outputs produced by Raspberry Pi. In the context of this app, it is further envisaged for the periodic scheduled detection training so as to expand the categories of electrically propelled personal transport additions to the list of offerings on hand, such as e-skateboards, scooters etc. When more time becomes available, a weather API is planned for implementation on the Dashboard activity.

More powerful hardware and detection equipment would also be future additions for greater confidence levels in detection. But this is resource dependent. NVidia Xavier is preferable and top of the range but unaffordable for now.

Due to Androids frequent changes of notifications, for Bike-Ability, notifications will be moved to Telegram. In other words, the user will be notified of available bikes via Telegram.

CONCLUSION

One of the limitations of working with Android Studio is that its dependency library versions changes too frequently, and that holds implications for code. When the “Grandle” version got updated for example, method of coding needed to be adapted accordingly. As this change occurred towards the end of the project, when most of the code was almost already scripted, a backtracking through all code, to update changes was needed. This reduced hands on coding time as old features depreciated unnecessarily because the same old features and functionalities had to be approached with new coding methods.

As for detection accuracy, TensorFlow doesn’t have same detection accuracy/ speed compared to Yolo or RCNN due to Hardware limitation of Raspberry Pi.

APPENDICES

- 1 Project Proposal
- 2 Consent Forms
- 3 Reflective Journals
- 4 Usability Testing Report
- 5 Usability Tests and Results
- 6 Other Materials Used (Any other reference material used in the project for example evaluation surveys etc.

1. Project Proposal

PROJECT OVERVIEW

An application given the name Bike Ability will implement a Camera to keep track of the bike availability. Having the application installed will mean that the user's interaction with the bike station is more manageable in that the user is now notified of available bikes at bike stations. A camera that overlooks the parking bay/station is positioned and equipped to detect how many bikes are available through OpenCV Real-time information on the number of bikes on hand and ready for use is then sent to the application end-user who will be online with the application installed on their Android handset.

OBJECTIVE

The objective of the project that this proposal is for is the development of an application that would facilitate the use of rental electric bikes and improve its timeous accessibility for its users. This application is aptly named 'Bike-Ability' because the name captures its function, which is to inform the application end-user of information on the availability of bikes at a particular bike-station in advance of them reaching that station. At its core, the application aims to save users time that would have otherwise been spent needlessly when a bike renter would make his or her way to a bike station that has no available bikes, especially during peak or rush hours. An auxiliary but equally important function of giving the bike rental application user the opportunity to reserve that bike will be implemented as feature of 'Bike-Ability'.

When there are not any bikes available at the time a user arrives at the station in order to avail of a bike, then that means that the renter went to that station for nothing, and will have to go to another station that may be a minimum of 300 meters away in order to get a bike. And if there are no bikes available also at the latter station, it would mean even more time wasted. It is exactly the avoidance of these types of time-wasting scenarios that the app will be developed for. The old adage that came to mind at the genesis of the idea for the app is that time is money, but lost time cannot be bought back.

With this in mind, the value of the application becomes more prominent because it will be saving people the resource of time especially when its scarce, such as in the morning when getting to work on time is a factor that can be impacted greatly by an extra 300 meter walk to the next bike station because there were no bikes available at the first one approached.

Attaching the notification of the number of available bikes will be an option for the end-user to reserve a bike out of the available ones for a limited time. This guarantees the time-saving effort of the application, as, without this function, the point of the application may have only been a moot one, because of situations in which a user may have had the notification that a bike is available five minutes before reaching the station. Another user may arrive subsequently and take that last available bike, meaning that the user would be in the same position as a renter without the application.

In summary, the functional objectives of the project, i.e to develop the app called Bike Ability that will sport 4 use cases, namely: A Check Bike availability function, a reservation tool, a contact us option, and a rental function.

BACKGROUND

An increased need for urban mobility arose over the past couple of decades, as traffic on urban and city roads became increasingly congested. As this need was incrementally met, first with the introduction of Dublin City Bike rental that expanded in a way that welcomed suitable technological accompaniments in the form of software that developed alongside it; software that would keep tread of the latest technological advancements in general, such as the IoT.

Bike sharing industry provides an important means of public urban transport and it is therefore equally important to keep it functioning in line with technology in use. Despite its novelty, the bike-sharing industry opened its doors in the age of smartphones and smart technology, the IOT, and therefore services as such were made available through apps from the onset. Ireland was introduced to its first BSS, Advertising funded Dublin Bike scheme around September of 2009, shortly after major European cities such as Paris and Barcelona launched theirs in 2007. (DeMaio et al., 2020)

Many BSS operators provide apps for the end-user/bike rental customer's convenience in accessing the service. Among other information, the apps provide scheme information, rental payment functions, registration opportunities, and real-time information about station and bikes relative to the user's location. (see also Fig 1)

Relevant research findings

In recent years, numerous and diverse BSSs have been implemented in European cities (Janett B. et All, 2011)

Bike hire schemes operate in many other European cities such as Paris, Barcelona, Berlin, Stuttgart and Milan and are generally either publicly funded or funded through advertising. (KPMG et al., 2011)

Some BSS operators have applications (Apps) for mobile handsets and Smartphones (Figure 27) for use in BSS schemes as early as 2009.



While these apps provide real-time information about stations and bikes depending on the current position of the user (see also Table 4), it is not clear whether these apps provide real-time information to the end-user about the number of bikes available at a particular station, but this service does not yet exist in Ireland.

This is where the idea originated in response to that niche market space in the bike hire market for an app that serves to notify its user about the number of bikes available at a particular station that the user is approaching. The idea to develop the application is based on a functionality that has somehow been overlooked on the applications that exist for current bike sharing schemes such as the one operated by Dublin City Council.

The following table shows most of the basic functions that other application employed for similar purposes hitherto incorporate:

Fig 1

Back-end	Front-end
Station Monitoring	Registration
Redistribution Planning	Rental
Defect Management	Information
Customer Data Management	Customer Data Management
Billing	Payment

Fig 2

The similarities that Bike-Ability share with such apps is in its handling of user registration and rental details. The divergent point for bike-ability is the notification function that it will incorporate. This is not a function seen on any of the other applications investigated for other city bike rental schemes. Another novel use case that Bike-ability will provide the app user with is the option to reserve a bike from the available ones at his or her chosen bike station.

According to a European Community Research project, '[I]n most systems the rental price increases exponentially after the free period and reaches a high daily maximum or fine'. And, fines may issue

for late returns after the first free 30 mins. Late returns are many times impacted by the availability of parking spaces this is discussed under the scalability section of this proposal at heading 10. Most BSSs usually have more than one target group. While the main focus in urban schemes is the daily user who rides to work or to leisure activities, regional schemes often focus on the tourist market. And when people are on leisure time, in a foreign city where on holiday, the app under development would come in especially handy, as it would ensure that users do not arrive at a particular bike station only to find an availability problem, especially during rush hours. The following table indicates the latter as common problem, which is what the app aims to reduce:

	Work + Education	Leisure	Errands	Tourism
Requirements	Dense station network	24/7 service	Dense station network	Stations near PT
	Stations near PT stations and living quarters	Safety during the night	Lock on bike	Stations near points of interest
	Bikes & slots available			
Problems	Lack of rush hour availability	High prices for longer rental	Lack of options to carry goods	High prices for longer rental

Fig 3

Legal Conformity:

Research revealed that at minimum, when working with data of the public, GDPR legislation needs to be conformed to. For this purpose, an ethics form was completed to declare conformance.

PROJECT DELIVERABLES

The main outcome of this project is an Android Application that will notify its user in real-time of the number of bikes available at bike stations. A camera will be implemented through Raspberry Pi for that purpose. The detection function is facilitated by OpenCV a computer vision library and machine learning. There will be four main use cases for this project, namely:

- Check bike Activity
- Reservation Activity
- Rent
- Contact Us

The functions of the app will be developed through the following application pages:

Registration and Create New Account

- A Login Page: This is how the user will access the offerings of the app.
- A Home Page
- Password Reminder: When users lose or forget their passwords they will be able to change it.
- check availability button complete with RecyclerView and CardView (a sort of Displayed list on Android Studio),
- Location indicator / Maps

- Reservation Activity

TECHNICAL APPROACH

The Process/Methodology Overview

In order to achieve the deliverables on offer by this Android application, its main features will be supported by the implementation of a camera to detect available bikes. This use case will be supported by Raspberry Pi as microcontroller incorporating Linux Distro. and OpenCV (Open Source Computer Vision Library) The latter is an “open source computer vision and machine learning software library”. It was built to “provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products”. Open CV is a BSD-licensed product, because it easy for businesses to utilize and modify the code.”(OpenCV team., 2020) Although informing the decision to use it, its open source quality is not the main reason why it was chosen over for example another machine learning tool such as YOLO V3. A comparison between Yolo and Open CV revealed that Yolo v3, in its resource hunger does not perform well on the tiny hardware of Raspberry Pi. Open CV needs less resources to get the same task done to the same standards.

Implementation starts with implementing the designed UI, followed by other iterations in turn, including a real-time database and push notifications up to completion stage.

Research

Research plays an important role to avoid the underestimation of planned iteration and to minimize the probability of errors occurring at later stages of the project; in that way, the success of the project can be realized. Comprehensive and adequate research will also underpin a thorough understanding requirements capture that encompasses the entirety of the deliverables and outcomes of the project. Further research will be ongoing into finer details such as the colors-chemes of the UI and other front end iterations such as the ergonomics of screen real-estate. The main sources for research are college library and google scholar to check future implementations.

Use Case and Architecture Diagram

For the use case Rational Rose will be the Software, used for diagrams that are needed in visualizations of different components and functions of the app. Draw Io is another important visualization tool that will help to better understand the project functionally in all the aspects.

Mock-up Prototype

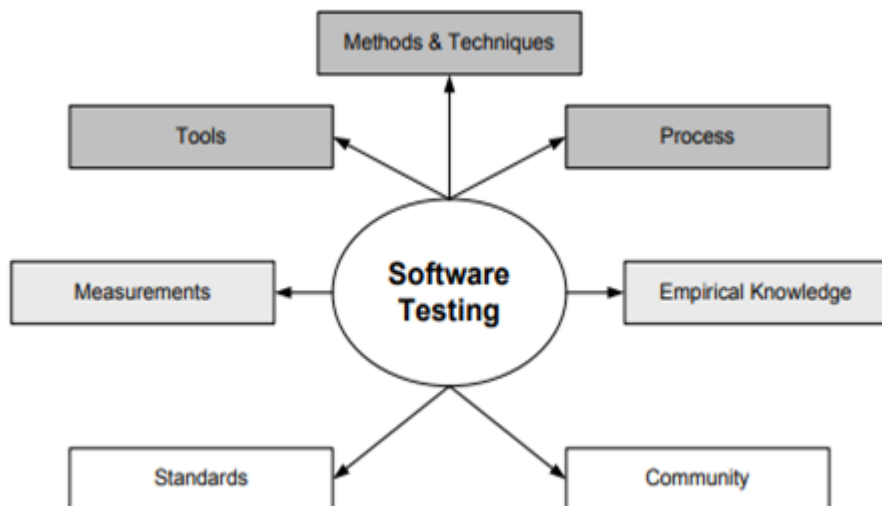
For advance visualization of the project structure Balsamiq Mockups 3 is used in order to keep track of minor adjustments and to maintain an understanding of the desired outcomes that will ensure overall project success.

Testing

The importance of testing cannot be overstated in its role to greatly reduce the incidence of bugs in the application and will happen on JUnit and Espresso test an integrated framework for testing in Android.

Testing will happen throughout the different phases of the project and is projected to be performed at the end of each milestone reached. Testing is a vital component of the project that will show the shortcomings of both process and product that can be met before product launch or demonstration.

A model that forms the basis for the model of testing employed in this project looks as follows:



Android Studio Application

The application will be developed on Android Studio using Java programming language, while Python programming language will be used for the Raspberry PI integration, with Firebase acting as an intermediary between Python and Android.

Raspberry PI (RP) /Camera (BackEnd Functionality.

RP is a single board computer with a microcontroller. This is the main component that will be used to monitor with the help of the camera the bike availability that will inform the alert that will issue to the user.

Firestore Database (FB)

Firestore database is a real-time database, this is exactly the function that I need to display information in real time. FB will work in conjunction with the Raspberry Pi that will save the information on FB. In this case, Android will act as a reader that will read from FB. For the login and registration stored on FB, android will be used as both reader and writer to Firestore,

FB, therefore, serves as intermediary communicator between Raspberry Pi and the Android application. While the Raspberry pi camera captures bike availability information to be saved on FB,

the Android application will access and retrieve the information and display it in a List called Recyclerview for the end-users perusal.

IMPLEMENTATION

Technical Details

To implement the components necessary for the successful delivery of the project in accordance of Industrial Standards, the following main considerations given prominence to were the synthesis of the main Hardware- and Software components. :

Hardware

- Raspberry PI Implementation

Raspberry OS formerly known as Raspian is the operating system that is chosen for this. Another called Noobs was considered, but Noobs eventually leads to Raspian. Raspian can work in headless mode without the use of screen or keyboard. And this is what is needed for this project because the Raspberry Pi will be accessed through a mobile phone. Another reason that underpinned the decision to use Raspberry Pi OS is because with it comes official support.

- Mobile Phones with different Android versions, ranging from Kit-Kat version 4 to Pie version 9 for testing purposes.
- Camera (Make and Model)

Software

- Raspberry PI Implementation: Raspberry Pi OS

SPECIFICATIONS

Hardware Specs

Raspberry Pi Camera Specification: 8 MP

Raspberry Pi - A microcontroller for the camera being implemented.

Android Mobile Phone with various API and models (from KitKat to Android Q) (Also for testing)

Software Specs

Firebase Database: For storing user data and bike availability data

Raspberry Pi OS (Raspbian): Used for the implementation of camera.

Balena Etcher software: To write the Raspberry Pi OS on the SD card

- Rational Rose for use case Diagram
- Balsamiq Mockups 3
- Android Studio
-

Programming Languages used

Java for Android

- Python for Raspberry PI
- XML for Android to design the page
- PNG File containing Pictures
- JSON for animation and to load firebase
-

SPECIAL RESOURCES REQUIRED

- Camera - Make and Model Number and when ordered and expected delivery date.
- Research materials from books and online sources such as Tutorials Point
- API : Open Source Library

Further research is planned into the use of colour in android applications user interface and how these play a role in guiding user perception when using the application. The Findings of that research that will form part of the documentation will be used in the implementation of the user interface.

An Ethics form was uploaded in compliance with requirements by GDPR because the application will incorporate the use and storing of data that is of public consequence.

Evaluation

Mobile Phones with different Android versions ranging from Kit-Kat version 4 to Pie version 9 will be used to test the solution. Testing will be the most important phase of the project, because this will be the phase that determines whether the project is a success. Code will be tested step by step though, so

that testing takes place throughout and intermittently to see if specific project goals that were set for specific times were met.

Scalability

Notwithstanding the time constraints on users/renter where bikes has to be returned to stations before the specified paid for time period lapses, pressure mounts when such a renter needs to go from station to station in search of a vacant spot for the return of the rental. Vacant spots are not always available near renters' destinations or within a convenient radius of their destinations. This may not only add avoidable wasted minutes to an already near exhausted predefined or prepaid rental time-slot, but it may also add minutes to arrival times and deadlines. This is where the application being developed may be scaled to accommodate the the function of the existing indicate to users in advance of reaching a particular bike station whether there is a parking lot / space available. The current application being developed will provide accurate real-time information on the number of bikes available.

Future Work

Future additions to the app will issue in features such as additional administrator features so as to expand the app's allowable for administrators, such as updating and inventory management of the bike store facilities. Once enough momentum is gained resource wise, with the new Nvidia Xavier on board, which is quite expensive for now, new capabilities possible for this app. The Nvidia Xavier NX Series is able to increase computation, with increased accuracy in detection and will therefore be an improvement on the raspberry Coral. With raspberry pi and coral the libraries are limited to tensor flow. With the Nvidia Xavier neural networking is possible that would facilitate faster, better and more accurate identification and detectability. By implementing said newer technologies such as the Xavier, the project becomes infinitely more scalable.

Project Plan/Milestones

Weekly reflection sessions to look at goals is worked into the schedule of this project and this is important in order to fulfill project milestones timeously and to complete the project in the projected time-frame. A gant chart is included for this purpose under topic 11 of this proposal.

A Gantt chart using Microsoft Project with details on implementation steps and timelines:

		Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Adc
1			1 Initializing	45 days	Mon 28/09/20	Fri 27/11/20		Software Project	
2			1.1 Brainstorming	3 days	Mon 28/09/20	Wed 30/09/20		Software Project	
3			1.2 Project proposal Pitch video	22 days	Wed 30/09/20	Fri 30/10/20	2	Software Project	
4			1.3 Review Content	20 days	Mon 02/11/20	Fri 27/11/20	3	Software Project	
5			1.4 Project Proposal Documentation	11 days	Mon 02/11/20	Mon 16/11/20	3	Software Project	
6			2 Planning	29 days	Wed 04/11/20	Mon 14/12/20	4	Software Project	
7			2.1 Create Preliminary Scope/work Statement	5 days	Tue 17/11/20	Mon 23/11/20	5	Software Project	
8			2.2 Prepare Work Breakdown Structure	11 days	Tue 17/11/20	Tue 01/12/20	5	Software Project	
9			2.3 Estimate time to develop the Application	16 days	Tue 17/11/20	Tue 08/12/20	5	Software Project	
10			2.4 Define user requirements	10 days	Tue 17/11/20	Mon 30/11/20	5	Software Project	
11			2.5 Develop schedule and tasks	10 days	Tue 17/11/20	Mon 30/11/20	5	Software Project	
12			2.6 Determine Project App development	1 day	Tue 17/11/20	Tue 17/11/20	5	Software Project	
13			2.7 Develop App Project Plan	10 days	Tue 17/11/20	Mon 30/11/20	5	Software Project	
14			2.8 Project Plan Approval Agreement by supervisor	10 days	Tue 17/11/20	Mon 30/11/20	5	Software Project	
15			3 Executing	85 days	Mon 02/11/20	Fri 26/02/21	5	Software Project	
16			3.1 Define system requirements	6 days	Tue 17/11/20	Tue 24/11/20	5	Software Project	
17			3.2 Define Application Requirements	16 days	Wed 18/11/20	Wed 09/12/20	5	Software Project	
18			3.3 Create Concept and Prototype	14 days	Wed 18/11/20	Mon 07/12/20	5	Software Project	
19			3.4 Define Application functionality	35 days	Fri 20/11/20	Thu 07/01/21	5	Software Project	
20			3.5 Define server hosting requirements Firebase	22 days	Sat 21/11/20	Mon 21/12/20	5	Software Project	
21			3.6 Procure Hardware/Software	70 days	Mon 23/11/20	Fri 26/02/21	5	Software Project	
22			3.6.1 Install Development Software System [Android Studio]	1 day	Mon 23/11/20	Mon 23/11/20	5	Software Project	
23			3.6.2 Design/Build application System	69 days	Tue 24/11/20	Fri 26/02/21	5	Software Project	
24			3.6.2.1 Design Home Page Content	6 days	Wed 25/11/20	Wed 02/12/20	5	Software Project	
25			3.6.2.2 Design Log in Register Content	3 days	Thu 26/11/20	Mon 30/11/20	5	Software Project	
26			3.6.2.3 Design Menu Form Content	3 days	Fri 27/11/20	Tue 01/12/20	5	Software Project	
27			3.6.2.4 Design Contact Us Content	4 days	Mon 30/11/20	Thu 03/12/20	5	Software Project	
28			3.6.2.5 Design Forgotten Password Content	4 days	Sun 01/11/20	Wed 04/11/20	5	Software Project	
29			3.6.2.6 Implement Camera	1 day	Mon 02/11/20	Mon 02/11/20	5	Software Project	
30			3.6.2.7 Implement Bike detection	4 days	Tue 03/11/20	Fri 06/11/20	5	Software Project	
31			3.7 Application test Phase	39 days	Tue 17/11/20	Fri 08/01/21	5	Software Project	
32			3.8 Demonstration of Application [Presentation week 13]	1 day	Tue 22/12/20	Tue 22/12/20	5	Software Project	
33			4 Monitoring & Control	99 days	Tue 22/12/20	Fri 07/05/21	5	Software Project	
34			4.1 Project Status Monitoring	66 days	Wed 23/12/20	Wed 24/03/21	5	Software Project	
35			4.2 Analyze and evaluate Performance	66 days	Sun 27/12/20	Fri 26/03/21	5	Software Project	
36			4.3 Evaluate Risks	66 days	Mon 28/12/20	Mon 29/03/21	5	Software Project	
37			4.4 Update Request changes	66 days	Tue 29/12/20	Tue 30/03/21	5	Software Project	
38			4.5 Update Files/Records	52 days	Wed 30/12/20	Thu 11/03/21	5	Software Project	
39			4.6 Archive Files/Documents	28 days	Mon 04/01/21	Wed 10/02/21	5	Software Project	
40			5 Closing	87 days	Fri 22/01/21	Mon 24/05/21		Software Project	
41			5.1 Confirm Application is completed in accordance to requirements	85 days	Mon 25/01/21	Fri 21/05/21	5	Software Project	
42			5.2 Project Completed	1 day	Fri 21/05/21	Fri 21/05/21	5	Software Project	
43			5.3 Final Video Presentation	1 day	Mon 24/05/21	Mon 24/05/21	5	Software Project	
44			5.4 Online Project Showcase	1 day	Mon 24/05/21	Mon 24/05/21		Software Project	

2.Ethics Approval Application/ Consent Forms



National College of Ireland

DECLARATION OF ETHICS CONSIDERATION

School of Computing

Antonio Penna

Student Name:

Student ID: x01417383.....

Programme BSCSD4... **Year:**2021/22.....

Module: Software Project.....
Bike Ability

Project Title:

Please circle (or highlight) as appropriate

This project involves human participants	Yes / No
--	----------

INTRODUCTION

Secondary data refers to data that is collected by someone other than the current researcher. Common sources of secondary data for social science include censuses, information collected by government departments, organizational records and data originally collected for other research purposes. Primary data, by contrast, is collected by the investigator conducting the research.

A project that does not involve human participants requires ONLY completion of Declaration of Ethics Consideration Form and submission of the form on module's Moodle page

A project that involves human participants requires ethical clearance and an Ethics Application Form must be submitted through the module's Moodle page. Please refer to and ensure compliance with the ethical principles stated in NCI Ethics Form available on the Moodle page.

The following decision table will assist you in deciding if you have to complete the Declaration of Ethics Consideration Form or/and the Ethics Application Form.

Public Data	Y	Y	Y	Y	N	N	N	N
Private Data	Y	Y	N	N	Y	Y	N	N
Human Participants	Y	N	Y	N	Y	N	Y	N
Declaration of Ethics Consideration Form	x	X	x	X	X	X	x	
Ethics Application Form	X		X		X		X	

Please circle (or highlight) as appropriate

The project makes use of secondary dataset(s) created by the researcher	Yes / No
The project makes use of public secondary dataset(s)	Yes / No
The project makes use of non-public secondary dataset(s)	Yes / No
Approval letter from non-public secondary dataset(s) owner received	Yes / No

Sources of Data:

It is students' responsibility to ensure that they have the correct permissions/authorizations to use any data in a study. Projects that make use of data that does not have authorization to be used, will not be graded for that portion of the study that makes use of such data.

Public Data

A project that makes use of public secondary dataset(s) **does not need ethics permission**, but **needs a letter/email from the copyright holder** regarding potential use.

Some websites and data sources allow their data sets to be used under certain conditions. In these cases, a letter/email from the copyright holder is NOT necessary, but the researcher should cite the source of this permission and indicate under what conditions the data are allowed to be used. See Appendix I for examples of permissions granted by Fingal Open Data, and Eurostat website.

Where websites or data sources indicate that they do not grant permission for data to be used, you will still need a letter/email from the copyright holder. For example, see Appendix II for an example from the Journal of Statistics Education.

Private Data

A project that makes use of non-public (private) secondary dataset(s) must receive data usage permission from School of Computing.

An approval letter/email from the owner (e.g. institution, company, etc.) **of the non-public secondary dataset must be attached to the Declaration of Ethics Consideration**. The letter/email must confirm that the dataset is anonymised and permission for data processing, analysis and public dissemination is granted.

Evidence for use of secondary dataset(s)

Include dataset(s) owner letter/email or cite the source for usage permission

N/A	
-----	--

CHECKLIST

Non-public/private secondary dataset(s) -Owner letter/email is attached to this form OR Citation and link to the web site where permission is granted – provided in this form	Yes / No N/A Yes / No N/A
--	--

The Ethics Application Form must be submitted on Moodle for approval prior to conducting the work.


Considerations in data collection

- Participants will not be identified, directly or through identifiers linked to the subjects in any reports produced by the study
- Responses will not place the participants at risk of professional liability or be damaging to the participants' financial standing, employability or reputation
- No confidential data will be used for personal advantage or that of a third party

Informed consent

- Consent to participate in the study has been given freely by the participants
- participants have the capacity to understand the project goals.
- Participants have been given information sheets that are understandable
- Likely benefits of the project itself have been explained to potential participants
- Risks and benefits of the project have been explained to potential participants
- Participants have been assured they will not suffer physical stress or discomfort or psychological or mental stress
- The participant has been assured s/he may withdraw at any time from the study without loss of benefit or penalty
- Special care has been taken where participants are unable to consent for themselves (e.g children under the age of 18, elders with age 85+, people with intellectual or learning disability, individuals or groups receiving help through the voluntary sector, those in a subordinate position to the researcher, groups who do not understand the consent and research process)
- Participants have been informed of potential conflict of interest issues
- The onus is on the researcher to inform participants if deception methods have to be used in a line of research

I have read, understood, and will adhere to the ethical principles described above in the conduct of the project work.

Signature: 

Date:01/02/2022.....

Reflective Journals

Reflective Journal October

Student name: Antonio Penna x01417383

Programme BSc in Computing – Software Development

My Achievements

The aim is to draft a Project proposal that may be updated gradually and incrementally. This means that its content may be increased incrementally as different project phases complete over the course of the semesters.

In its current form, the project idea grew out of two prior project ideas that were discarded due to time constraints and ‘Ethical Approvals’ that needed to be obtained, but decided against. As they were both very good ideas, I have decided to try to salvage some of their features and iterations for implementation in this finally settled.

Starting the new academic year running, with pressure already mounting and either parts of- or full assignment. The thing that turns the pressure up even more is that nearly all projects this semester and indeed for this year are ‘solo Projects’, that would otherwise be worked in teams of at least two in the real-world. And albeit challenging and with the success of these projects being my sole responsibility that I look forward to maintaining to a very high standard and degree. To that end, pristine planning and time-management based on finely planned schedules, complete with rest-breaks and breaks for exercise will be adhered to. The one thing that is advantageous to my plans is the fact that this semester is completed from home, meaning that less time will be spent travelling to and from classes. The missing 2 hours plus that the missing travel time add to my weekly calendar has already been allotted to a deliberately reflective recap and progress monitoring session at the end of each week.

The main project is now moving from an idea phase into the initiation of its more realized phase of appearing on paper into its planning phase. The supervisor assigned for this project is Paul Stynes. with whom the first, introductory meeting is scheduled online for Monday 2 November 2020. In addition to meeting the supervisor for the first time the agenda for the meeting includes a discussion and questions that I have compiled regarding iterations such as ‘Deep learning’ that I am excited to implement with the project.

Reflective Journal for November

All the preparation for the planning phase of the project has now been done with the frame of the project proposal starting to take shape. Slots were entered onto my weekly planner and calendar also for the adding of text to the project proposal which I plan to spend about an hour per day on.

As regards to the project itself, the software that I will use to write the coding and programming of the application with is Android Studio, because the application under development is a mobile telephone app. I have also been working on some planning sketched ideas of the applications UI that foregoes the Project Proposal on paper and I am deliberating whether to include a section into the proposal that showcases these initial sketches.

Upon preparing user flow charts of the tasks ahead, I started scripting the code for the app and thus far I have not been stuck on any sections of it the realization of which motivates me to swiftly arrive at the other planned features for the app. With the operating system 'Raspbian' downloaded and installed, the initial testing phase of the equipment that includes the Raspberry Pie element and camera that I acquired timeously for implementation has now completed successfully also meaning that more time and attention can now focus on some of the Front-End developments and the implementation of the Login page of the application.

Time has also been allocated on my weekly scheduler for the building of the prototype of my project, that will be completed for demonstration by 22 December 2020, which marks another exciting new phase on the project calendar.

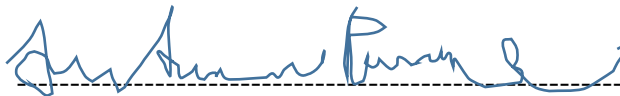
Supervisor Meetings

Date of Meeting 02/11/2020

Upcoming Meeting Agenda: Project feedback, the scope of the Project, possible implementation of Deep learning, Minutes of meeting to be recorded.

Action item: Write project proposal, write the journal, start the project Plan.

Signature: _____



Date: 31/10/2020-----

Reflective Journal December

Student name: Antonio Penna x01417383

Programme BSc in Computing – Software Development

My Achievements

At this point the project proposal is drafted and used, together with Gant Chart to follow every phase step by step. It is being updated and have been amended Model bicycles that was order in the first week of November arrived. This will be used for the demonstration phase of the prototype.

Camera

After arrival of camera its set up was preceded by a basic test that yielded positive results; the camera is working and functioning as should but Open CV to detect the bicycle has yet to be implemented. As far as testing went, the camera did not arrive with any tutorial and therefore I had to research a Youtube tutorial to set the camera up properly. The next step as far as the camera is concerned is to “train” it to identify the bicycle.

Rasberry Pie -was set up.

Altered Schedule due to Emergency

Back injury is causing me to alter time management plans and schedules as it prevents me from sitting up for hours or long periods of time that is requisite for the scheduled work needed on the project. This means that I am working and progressing at a much slower pace for now. Emergency equipment in the form of laptops were therefore set up to accommodate for my posture and ergonomics so as for this misfortune to have minimal impact on the progress of the project. After the risks to the timeous completion of scheduled tasks was assessed, I thought it necessary to look at the schedule to see what smaller tasks that would not require me to physically sit up at the computer could be done now instead of later.

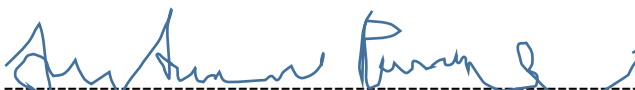
This is why I opted over the past two weeks to perform smaller tasks now such as setting up the camera, that would have been done at a later stage in the project so that less time is impacted upon by my back injury. What I’ve learned through this is that this sort of thing needs to be accommodated for in the planning phase, where a week or two excess time for the event of misfortune such as injury or convalescence; health is a resource that can not be discounted or forgotten about. Additionally tasks should be able to be swopped around on the schedule and therefor a degree of flexibility on the schedule needs to be maintained since the inception or planning phase for the sake of crisis and risk management.

Supervisor Meetings

Date of Meeting

Upcoming Meeting Agenda: Project feedback, the scope of the Project, possible implementation of Deep learning, Minutes of meeting to be recorded.

Action item: Write project proposal, write the journal, start the project Plan.

Signature: -----

Date: ---01/12/2020-----

Reflective Journal January

Student name: Antonio Penna x01417383

Programme BSc in Computing – Software Development

January 2022

My Achievements

- Bicycle and Bicycle Detection

A second bicycle was ordered and arrived for building the prototype. I ensured that the bicycle is a not too small otherwise CNN computer Vision will not detect it. This taught me how important even the tiniest detail in a project like this is.

Additionally, Yolo was tested but was crashing because Yolo was too resource hungary and impacted upon performance. The strategy was adjusted accordingly and Yolo was substituted with Open CV, but not before extensively researching it. I looked at what other people used for vision detection projects. Open CV is less resource demanding and performed much better. Bicycle detection function for Bike-Ability is not yet complete and is still under construction. Considerable time was spent writing the technical report for the project also. Use Case Diagrams and the Structure of the report took some time to match with this project.

- Back Injury and Convalescence

Due to back injury and recovery time, the mid-term presentation originally scheduled for 21 December has been postponed until 6 January. The application for the planned midpoint presentation is ongoing as I could not perform at 100%. The examination and many other written submissions are factored into the schedule for a successful presentation/submission on 6 January. This was also finalized in the supervisor meeting for December.

Planned for next month

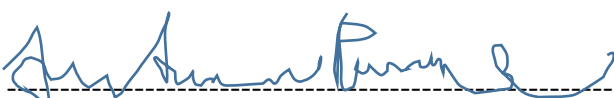
It is planned for next month to implement the Computer Vision, the major components of the project. The other tasks focus on completion of the code for the detection of the available bikes. using Python. It is hoped for the start the testing phase by the end of January.

Supervisor Meetings

Date of Meeting: A next supervisor meeting is more than likely scheduled for after the examination, the date for which is not available at the submission of this journal.

Upcoming meeting agenda:

Project Progress Feedback with Supervisor; Very Important to discuss possibility of getting this application patented.

Signature: -----

Date: 01-01--2021-----

Reflective Journal February

Student name: Antonio Penna x01417383

Programme BSc in Computing – Software Development

February 2022

My Achievements


During this month the advice resulting from the January supervisor meeting was researched and acted on. It was discussed during the meeting to look at implementing analytics that would track the interactions customers had with bike stations. The aim with this would be to produce statistics on these interactions such as the busiest time of bike station usage along with the personalized usage for a particular app user. Another achievement was to write the code for the unique user identifier number that is generated for the bike-reservation functionality. Despite this month being one full of tight deadlines, I completed the code for two main use cases of the app successfully. A skeletal draft of the report that accompanies the project has been completed and from March on, adjustments will feature as needed and as the final stages of the project draws to a close.

Planned work for next month

Implement analytics recommendation based on user choice. Code testing with Espresso will commence during the second week of March, as part of the test-driven development methodology that is being followed with this project. The FPS of Raspberry pie also needs to be increased and this is a priority on the assigned research hour list for the first week of this month; this will require for my own model to be trained instead of the pre-trained detection model. It is aimed with this to increase the FPS, as the bicycles will be the only images being detected. The corollary effect is that detection accuracy increases in doing so.

Supervisor Meetings

Some information is needed on the insertion of functional requirements on the template for the report. A number of usability tests and interviews were conducted as part of the Usability Design module; find out from the supervisor as to whether I can use some of the material, especially in terms of non-functional requirements and test results.

Signature: 

Date: 28-02--2022-----

Reflective Journal March

Student name: Antonio Penna x01417383

Programme BSc in Computing – Software Development

March 2022

My Achievements

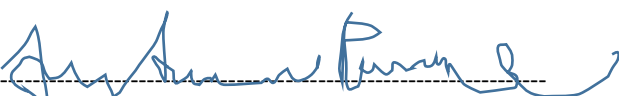
As work progressed, more was added to the draft of the report that accompanies the project. Found out that data science will take too much out of the schedule for this project and may jeopardize its on time delivery. The analytics idea now got added to the future work of the project. In lieu of the analytics moving to the future works heading for this project, a fifth use case, in the form of a chat bot was planned and the script for the chat bot was successfully written. Since the chat bot consist of if statements, no test cases as part of test-driven development methodology was applicable. The FPS was increased with Google Coral. An extra CPU was added for extra computation power.

Planned work for next month

The If Statements for the chatbot needs added during the first week of the new month. Methods for counting the number of Bicycles available v the number of spaces available will need to be researched and implemented, as this is one of the main functions of the app. A method to prevent over-booking of bicycles are on the cards and has to be researched and implemented from midway through the month onwards to completion at month end.

Supervisor Meetings

Further possibilities and Questions were raised about analytics. The value of bookings history was discussed and it was decided that this will serve the purpose of the app well. Plans are in place for researching and implementing the usability feature.

Signature: 

Date: 28-03--2022-----

Reflective Journal April

Student name: Antonio Penna x01417383

Programme BSc in Computing – Software Development

April 2022

My Achievements

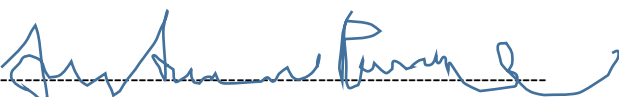
The chatbot was implementation completed on schedule by the end of the first week of the month. Methods for counting the number of bicycles on hand as well as the number of spaces that the app may notify the customer of was researched and implemented successfully by the end of the 2nd week of the month. For the remainder of the month a method for preventing over-booking of bicycles was researched and partially implemented.

Planned work for next month

Booking history methods of implementation needs to be researched and coded for. The all important app testing phase is set to begin mid-way through the month of May and testing will involve validation and unit testing. The report will also have to move to its final stages where it will be polished and proof-read. A poster for the app needs to be designed and made, as well as a video recording in which the app is presented completes the final week of May.

Supervisor Meetings

It is on the agenda to find out certain details about the structure of the report such as that of the placement of functional requirements v non-functional requirements. Questions are pending on whether login and registration actually has to be included as use-cases as some lecturers says yes and others say it does not qualify as use-case.

Signature: 

Date: 28-04--2022-----

Reflective Journal May

Student name: Antonio Penna x01417383

Programme BSc in Computing – Software Development

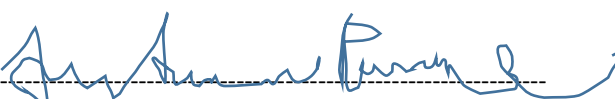
May 2022

My Achievements

The implementation of the over-booking feature was completed. Validation and Unit testing of use cases completed. This was difficult as absolutely no testing is ever taught in college even though this is a very important part of the project. Screenshots of test were inserted into the report. A poster for the app was designed and saved. A Video presentation of the app completed. The report for the app is at proof-reading and polishing stage.

Supervisor Meetings

The final meeting for this project laid to rest the pie-chart idea for good as this was found not to be fully supported by android. The uncertainty over whether login and registration is to be included in use-cases was cleared up when the supervisor confirmed that these in fact qualify as use-cases. It was decided upon that to include them. I ensured in this last meeting to confirm all details for the uploading of the video presentation and the project in general and thanked the supervisor for their support during this difficult project.

Signature: 

Date: 28-05--2022-----

BIBLIOGRAPHY

Android Developers. 2022. *Write automated tests with UI Automator* | *Android Developers*. [online] Available at: <<https://developer.android.com/training/testing/other-components/ui-automator>> [Accessed 5 April 2022].

clipartkey. 2021. clipartkey. [online] Available at: <https://www.clipartkey.com/view/iRxJbmi_road-bike-cycling-vector> [Accessed 1 April 2022].

DeMaio, P., 2020. The Bike-Sharing Blog. [online] [Bike-sharing.blogspot.com](http://bike-sharing.blogspot.com). Available at: <http://bike-sharing.blogspot.com/2007/> [Accessed 8 November 2021].

Firebase. 2020. Service Level Agreement for Hosting and Realtime Database | *Firebase*. [online] Available at: <<https://firebase.google.com/terms/service-level-agreement>> [Accessed 5 May 2022].

Great Learning Team. 2021. Real-Time Object Detection Using TensorFlow. [online] Available at: <<https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/>> [Accessed 5 April 2022].

Irish Times. 2022. [online] Available at: <<https://www.irishtimes.com/special-reports/business-ireland-magazine/trillion-euro-plan-aims-to-transform-eu-into-low-carbon-economy-by-2030-1.4417199>> [Accessed 1 August 2022].

Janett B. et All, 2011. [ebook] Available at: https://ec.europa.eu/energy/intelligent/projects/sites/iee-projects/files/projects/documents/obis_handbook_en.pdf [Accessed 8 November 2021].

KPMG et al., 2011. [ebook] Available at: <<https://www.nationaltransport.ie/wp-content/uploads/2011/12/Further-attachments-provided-with-response.pdf>> [Accessed 8 November 2021].

Medium. 2019. Google Coral USB Accelerator Introduction. [online] Available at: <<https://towardsdatascience.com/google-coral-usb-accelerator-introduction-18989a962d25>> [Accessed 5 August 2022].

Mender, 2022. *Bicycle Bike Clipart 6 Bikes Clip Art 3 Image - Huffy 26 Alpine Womens Mountain Bike*, *Free Transparent Clipart - ClipartKey*. [online] [clipartkey.com](https://www.clipartkey.com). Available at:

<https://www.clipartkey.com/view/oiwRR_bicycle-bike-clipart-6-bikes-clip-art-3/> [Accessed 1 May 2022].

Mike van Drongelen, *Android Studio Cookbook*, page. 22 ‘*Design, debug and test your apps using Android Studio*’

OpenCV team, Courses, A., I, C., II, C., PyTorch, D., Partnership, D., Sponsorship, G. and Kit, M., 2020. About. [online] [Opencv.org](https://opencv.org/about/). Available at: <https://opencv.org/about/> [Accessed 5 November 2021].

Tripadvisor. 2022. [online] Available at: <https://www.tripadvisor.com/Attraction_Review-g186605-d6439815-Reviews-Dublin_Bikes-Dublin_County_Dublin.html#REVIEWS> [Accessed 2 July 2022].

United Nations. 2011. *BICYCLE-SHARING SCHEMES: ENHANCING SUSTAINABLE MOBILITY IN URBAN AREAS*. 1st ed. [ebook] Bruxelles. Available at: <https://www.un.org/esa/dsd/resources/res_pdfs/csd-19/Background-Paper8-P.Midgley-Bicycle.pdf> [Accessed 11 March 2022].