A Real-Time Machine Learning Based Public Transport Bus-Passenger Information System [⋆]

Menzi Skhosana and Absalom E. Ezugwu

School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Durban, South Africa
Corresponding author: Absalom E. Ezugwu, email: ezugwua@ukzn.ac.za

Abstract. The era of Big Data and the Internet of Things is upon us, and it is time for developing countries to take advantage of and pragmatically apply these ideas to solve real-world problems. Many problems faced daily by the public transportation sector can be resolved or mitigated through the collection of appropriate data and application of predictive analytics. In this body of work, we are primarily focused on problems affecting public transport buses. These include the unavailability of real-time information to commuters about the current status of a given bus or travel route; and the inability of bus operators to efficiently assign available buses to routes for a given day based on expected demand for a particular route. A cloud-based system was developed to address the aforementioned. This system is composed of two subsystems, namely a mobile application for commuters to provide the current location and availability of a given bus and other related information, which can also be used by drivers so that the bus can be tracked in real-time and collect ridership information throughout the day, and a web application that serves as a dashboard for bus operators to gain insights from the collected ridership data. These were integrated with a machine learning model trained on collected ridership data to predict the daily ridership for a given route. Our novel system provides a holistic solution to problems in the public transport sector, as it is highly scalable, cost-efficient and takes full advantage of the currently available technologies in comparison with other previous work in this topic.

Keywords: Bus location tracking · Mobile tracking · Short-term forecasting.

1 Introduction

Public transportation efficiency is essential for economic growth and sustainability of urban areas. However, in developing countries, there are little or no advancements being made in this sector. Among many other benefits, public transportation significantly offers greater affordability, increases mobility, reduces traffic congestion and can lessen the carbon footprint of urban areas. The

[⋆] Supported by Huawei Technologies Africa (Pty) Ltd sizable capacity of public transport vehicles results in lesser trips and reduced fuel consumption.

According to Statistics South Africa, more than 76.7% of South African households rely on public transport for daily commutes [1]. This again highlights the importance of public transport and why problems in this sector need to be addressed. In their article [2], they outline that the development of physical infrastructure, e.g. more highways or bigger roads does not make traffic congestion better but may make it worse; this is a phenomenon known as the Induced Travel Demand (ITD). This leads to the point that the development of proper infostructure - which is defined as an electronic infrastructure that enables the sharing of knowledge and information among various actors in the society [3] - is just as important. This reason has led to the recent trend and drive to modernise public transportation by integrating it with the latest technologies for more viable and eco-friendly environments. However, this trend has only been more prevalent in developed countries around the world and not so much in still developing countries like South Africa. This lag in developing countries is because many cities lack financial resources to implement these technologies [4]. Furthermore, it is noteworthy that approximately 55% of the world's population resides in urban areas, and this figure is expected to increase to 68% by the year 2050 [5].

Moreover, according to the Independent Communications Authority of South Africa (ICASA), smartphone usage has increased rapidly in the past five years. As of 2019 September, 91.2% of the South African population owned a smartphone, which is nearly 10% increase from the previous year. ICASA also reports on the state of the ICT sector in South Africa in terms of internet access coverage across the country, with the national population coverage for 3G mobile connection having increased from 99.5% in 2018 to 99.7% in 2019, and the coverage for 4G/LTE increased from 85.7% in 2018 to 92.8% in 2019 [6]. Given the widespread availability of smartphones and relatively good network coverage; the use of smartphones as a platform for implementing a public transport management system (as part of the infostructure) will allow for easy access to transit information, and lower implementation costs, which favours developing countries.

Commuters - mostly in developing countries - are often left stranded at pick-up spots - clueless about the availability and proximity of their mode of public transportation vehicle hence the bad stigma public transport has. This is a result of the unavailability of real-time information to commuters, poorly managed fleets, varying demands, traffic congestion and rigid schedules. Rapid population growth and urbanisation lead to an overwhelming travel demands [7]. A better understanding of the commuters' behaviour helps transport operators to estimate the demand and to propose better services to improve the commuting experience [8]. However, in developing countries, the public sector's finances are generally so minimal that funding for transportation innovation is insufficient [9]. This calls for a cost-efficient and quickly implementable public transport solution to cater to these specific conditions. Previous work done in this sector [10–13] partially offers solutions to challenges mentioned above but still lacks practicability as they require cumbersome hardware modules to be installed in buses, the use of outdated technologies with complicated implementation and high maintenance and lack of scalability.

The aim of this research is to demonstrate how a pragmatic intelligent public transport management system (Irenbus) can be developed and implemented, especially in the context of South Africa and other developing countries. To achieve our aim, this research has been broken down into the following specific objectives:

- Design and develop an Android mobile application to be mainly used for informing commuters about the current status of a given bus through live location tracking.
- Develop an approach to collect daily ridership data through the use of only a mobile application, without any external hardware modules.
- Develop an approach to incorporate a continuously trained machine learning model to forecast daily ridership per route.
- Design and develop a web-based application to monitor buses, drivers and present the machine learning model's predictions for a given route to bus operators.
- Evaluate the feasibility of our proposed system (Irenbus).

Furthermore, this research is focused on the design and implementation of a low-cost intelligent real-time public transport system that could be easily deployed in developing countries. However, for this research's purpose, the system's implementation and evaluation will be mainly focused on buses, with three user roles - the bus user or commuter, the bus driver, and lastly the bus manager or operator who will be overseeing everything. The proposed Irenbus system, could be considered as a holistic and intelligent real-time public transport system that keeps commuters informed about the current location, estimated arrival time of a given bus while collecting daily ridership data that is used to provide predictive capabilities through the use of machine learning that renders useful insights based on the ridership patterns to public transport authorities.

The technical contributions of this research are as follows: first, a system that provides a communication portal between bus users, drivers and operators and also serves as a data management tool to store and organize collected daily ridership data intelligently. This system is accessible in the form of a mobile application for bus users and drivers, and a web application for bus operators [14]. Second, an approach to incorporate machine learning models into the system to enable a deeper understanding of the collected data and foresight [15]. Third, an approach to efficiently collect daily ridership data - unlike some previous work in this topic [16] - through the use of only a mobile application without the use of cumbersome hardware modules.

The remaining part of this paper is structured as follows: Section 2 looks into related work in both published academia work and software products in the industry.Section 3 describes the overall structure of the proposed system - Irenbus. Section 4 present the results of qualitative and quantitative evaluations of our proposed real-time public transport system. Section 5 provides a summary of our work and describes the contribution of this research. We also explain the resulting Irenbus system prototype and how it relates to previous work and the objectives of this research; and the limitations of our current system and possible improvements in future work.

2 Related Work

2.1 Bus Tracking

Using GPS modules and embedded mini-computer systems, in [10], a bus monitoring system was developed that provided real-time information such as the estimated bus arrival time and the route name to commuters. This information was displayed on an LCD screen mounted at the bus stop. Although this system was stable and provided useful information to commuters it had a few shortfalls, such as being fixed at one place, i.e. it was not able to provide information to commuters who were not at the bus stop. Furthermore, implementing this system would be a costly operation since every bus stop had to have an LCD screen mounted on it.

A Web-based system was developed for tracking buses in transit using a GPS tracker installed on the bus. Users got real-time information straight to their mobile phones. Using Google Maps users could see the bus on the map as it moved [13]. Their system ensured punctuality - which has been proven to be an essential factor in making buses more reliable [17]. Their system also seemed to be less expensive to implement compared to other previously done work but still lacked the ease of use - as it only offered a web application with no native mobile application, which did not give insight to public transport authorities about future demands and ridership patterns.

2.2 Ridership Forecasting

In [18], the authors discussed how a machine learning approach could be used to implement and assess predictive services for the users of a bike-sharing system. The models used in this study were trained on real-world historical usage data comprising of more than 280 000 entries covering all hires in Pisa for two years. Seasonality manifests sharp changes in the usage patterns (e.g. bikes tend to be used more in spring compared to winter). The learning models captured these seasonal patterns through the appropriate encoding of the bike usage time, which explicitly modelled cyclic information such as weekday and holiday.

In [19], a system was proposed that used a camera mounted overhead to count passengers by combining a Convolutional Neural Network detection model and a spatiotemporal context model to address the counting problem in scenes of low resolution and with a variation of illumination, pose and scale. Experimental results showed better performance than other results previously published, and they planned to extend the current method with more in-depth learning algorithms. The only drawback to their system was that it may not work in a very dense and crowded scene, and in that case, they planned to explore crowd density map estimation for their future work.

In [20], the authors explored options for using smart card data for performing simple analyses by using transport planning software. The data was converted to represent passengers per line and matrixes between stops. This matrix was then taken into the network to produce the measured passenger flows. Their method turned out to be valuable to operators to gain insights into small changes but was not able to give accurate insights into long-term changes.

2.3 Implemented Real World Systems

Over the past few years, the City of Cape Town has put in efforts to improve its public transport by introducing the MyCiTi bus service. This service has a mobile application to accompany it that offers access to live updates to track the arrival of buses and allows users to conveniently save their favourite routes, stops and destinations for quick and easy access [21]. This led [22] to claim that Cape Town has the best public transport service in Africa. Nevertheless, it still follows a rigid schedule and does not collect ridership data to use for forecasting future ridership.

A software-as-a-service platform, Optibus, leverages artificial intelligence and historical data to predict and analyze on-time performance. It then automatically generates running times to help schedulers and operations executives create better schedules. Better on-time-performance improves the reliability of the transit service – one of the main factors shown to increase ridership [23]. This software offers excellent insight and flexibility with schedules, but is focused only on scheduling and ridership prediction, and does not cater for commuters directly by providing relevant information and tracking.

According to our best knowledge, there has not been any published work that discusses how a holistic public transportation system can be designed and developed to relay useful transit information amongst commuters, drivers and bus operators while at the same time collecting and studying commuter boarding data to enable the prediction of future ridership patterns, hence allowing for the development of better and more efficient bus schedules by integrating machine learning forecasting abilities.

3 Irenbus Architecture, Design and Implementation

3.1 System Architecture

The proposed system is composed of two sub-systems viz. the mobile sub-system and the web sub-system. The mobile system is presented in the form of an

Android application and has two types of users, namely, the commuters and bus drivers. Commuters use the application to get real-time information about the current status of buses in transit. Bus drivers use the application to capture daily ridership data, and also the live location of the device will be sent periodically in the background throughout the bus trip. The web sub-system is in the form of a web application. The bus operators use it to get a detailed view of all buses in transit, perform administrative tasks and view ridership forecast for a given route.

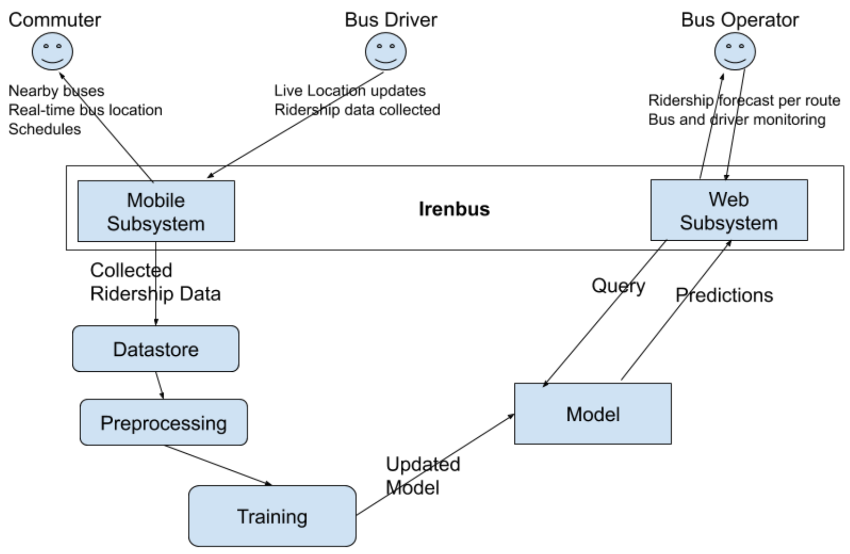


Fig. 1. Irenbus System Overview

The machine learning model is continuously trained with the daily ridership data collected by the mobile application, and the resulting model is used in the web application to forecast ridership, as shown in Figure 1.

The proposed system aimed to take full advantage of cloud-based services. Cloud computing allows for the delivery of different services through the Internet, including data storage, servers, databases and networking infrastructure [24]. A myriad of benefits come with cloud services which include cost savings, security, mobility, competitive edge and sustainability. For the proposed system, we used Google's Firebase, which is a Backend-as-a-Service platform. Firebase was chosen over Amazon Web Services' AppSync and other similar platforms because it allows both mobile and web applications access to shared data and computing infrastructures at lower costs and requires minimal setup and maintenance. Firebase provides real-time syncing of data across all the devices - Android, iOS, and the web.

3.2 Database Design

A NoSQL database was used for the proposed system, which is non-tabular, and stores data differently compared to traditional relational tables. NoSQL databases come in a variety of types, with the main types being document, key-value, wide-column, and graph-based databases. After assessing the data to be collected and stored by our proposed system, we decided to employ the key-value database. A Key-value database is a simpler type of database where each item contains keys and values. A value can typically only be retrieved by referencing its key. The key-value database is great for use cases where we need to store large amounts of data, but we do not need to perform complex queries to retrieve it [25]. Within the Firebase platform, this type of database is offered as the Realtime Database, mentioned in subsection 3.1. The Realtime database additionally offers the following features [26] [27]:

- **Optimization for offline use**: the Realtime Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.
- **Collaboration across devices with ease**: instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds.
- **Accessible from Client Devices**: can be accessed directly from a mobile device or web browser; there is no need for an application server.

Based on the objectives we set out for the Irenbus system, the following nodes resulted from our analysis:

Table 1. NoSQL Database Nodes Description

Node	Description
Bus	bus uniquely identified by a 16 character randomly generated id.
BusLine	bus line attributes, i.e line name.
OnlineBus	buses that are currently in transit.
Ridership	daily ridership for a given route/line
User	basic user information

3.3 Mobile Application

The mobile subsystem of Irenbus is in the form of an Android application. In Android development, the most crucial component of an application is the `Activity` class. In contrast to most programming paradigms in which apps are launched with a `main()` method, the Android system initiates code in an `Activity` instance by invoking specific callback methods that correspond to specific stages of its life cycle [28]. In [29] the `Activity` life cycle is described in detail.

The mobile application has four main activities:

– `StartActivity`: manages other activities and controls which activity is started based on the state of the application when it has launched.
– `LoginActivity`: handles all the user authentication tasks of the application.
– `CommuterMainActivity`: handles all functionality that pertains to the commuter user.
– `DriverMainActivity`: handles all functionality that pertains to the bus driver user.

Authentication : The mobile application has two types of users viz. the commuter and the driver. They each have different tasks they can carry out on the application, so we added an authentication process to give suitable access to different screens based on what type of user was currently logged in. Shown in Figures 2, 3 and 4 are some of the screens that are part of the authentication process.

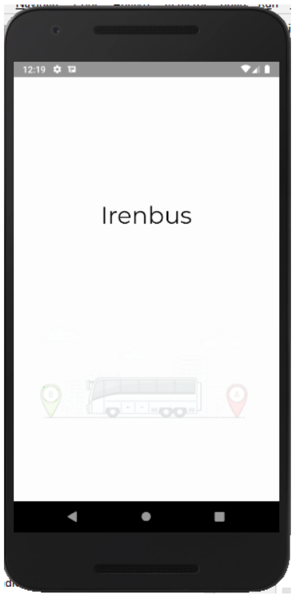


Fig. 2. Splash Screen

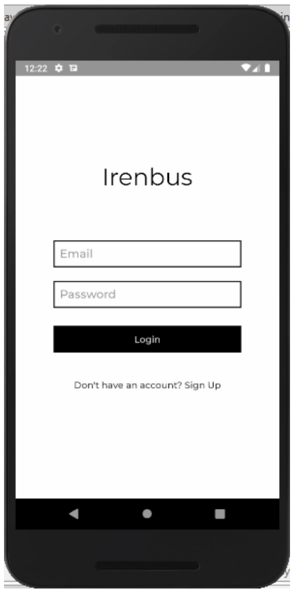


Fig. 3. Login Screen

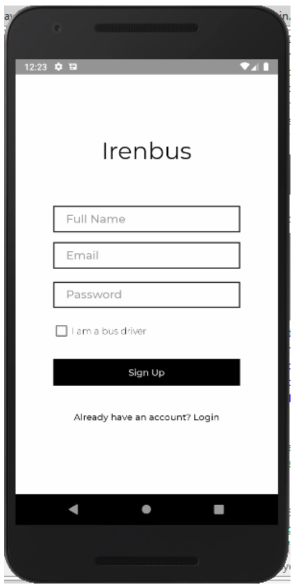


Fig. 4. Signup Screen

The splash screen is shown when the application is being launched, keeping the user company while the application runs a background authentication process. In [30] the authors claim the splash screen dramatically improves the user experience. The login screen allows the user to log in with their email and password. The signup screen allows the user to create a new user account. If the user on sign up checks the 'I am a bus driver' box, a 16 character alphanumeric code for the bus they are assigned to is required, to prevent unauthorized access.

On launch, the mobile application makes a request to the Firebase Authentication SDK to get the current user, as shown in the code snippet below. If the returned user object is null, then the user has not logged in before to the application. They are then presented with a login screen using an instance of the `LoginActivity` class. If the returned user object is not null, then the user has logged in before, and the application will then request the current user's **`userType`** value from the database with their **`userid`** which is obtained using **`firebaseUser.getUid()`**. After the current user's details have been read from the database, the application will give the user access to the appropriate screens based on the type of user they are.

Upon successful login, the authentication token is stored locally on the device; this means the next time the user launches the application they will not be required to enter in their password. Furthermore, this improves the user experience as the user will be able to log in even without an internet connection.

Commuter The primary purpose of the mobile application for the commuter is to keep them informed with real-time information about the current status of all buses currently in transit. There are no strict requirements to use the mobile application as a commuter; there is no personal information that is required other than an email and name, so anyone can sign up without worrying about their privacy. As a fair share of commuters is of age, the user interface is designed to be as minimal as possible and provide simple navigation, which will result in a better experience for the elderly. Additionally, the application is compatible with Google's TalkBack, which is an accessibility service that helps blind and visually impaired users to interact with their devices. When the commuter is logged in, the bottom navigation offers three tabs which each show the fragments in Figures 5, 6 and 7.

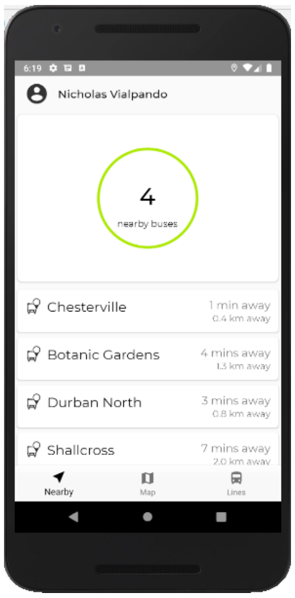

Nicholas Vialpando
4
nearby buses
Chesterville 1 min away 0.4 km away
Botanic Gardens 4 mins away 1.3 km away
Durban North 3 mins away 0.8 km away
Shallcross 7 mins away 2.0 km away
Nearby
Map
Lines

Fig. 5. Nearby Fragment

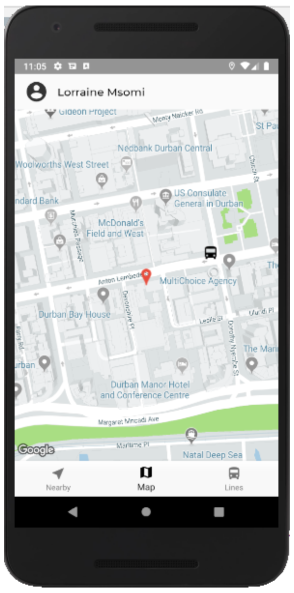


Fig. 6. Map Fragment

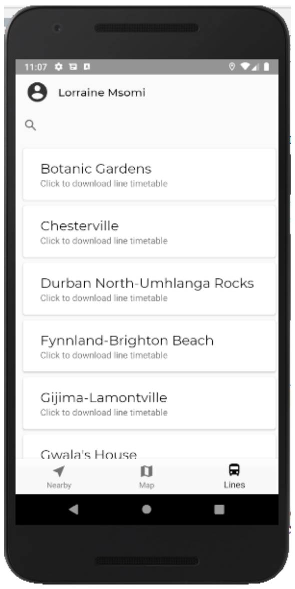
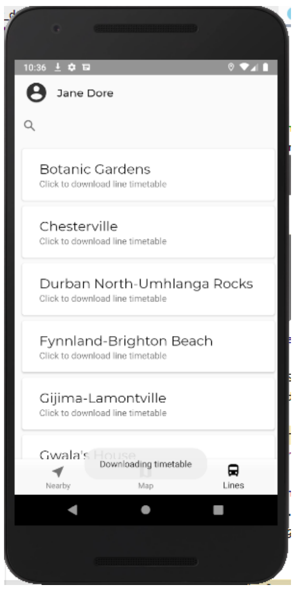

Lorraine Msomi
Botanic Gardens
Click to download line timetable
Chesterville
Click to download line timetable
Durban North-Umhlanga Rocks
Click to download line timetable
Fynnland-Brighton Beach
Click to download line timetable
Gijima-Lamontville
Click to download line timetable
Gwala's House
Nearby
Map
Lines

Fig. 7. Lines Fragment

Each fragments (Figures 5, 6, 7) in the application for the commuter provides the following functionality:

- *Nearby Fragment*: shows all nearby buses, which are buses within a three kilometer radius from the commuter's current location. The distance between each online bus and the commuter is calculated using the Haversine geolocation equation, $d = \left(\sqrt{\sin^2(\frac{\phi_2-\phi_1}{2}) + \cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\lambda_2-\lambda_1}{2})}\right)$. Every three seconds the bus location is updated on the database; this means that the distance is calculated each time the location changes. Additionally through the use of the Google Maps' Directions API - a service that calculates directions between locations and it is accessed through a HTTP interface, with requests constructed as a URL string, and latitude / longitude coordinates to identify the locations [31] - the estimate time of arrival is obtained, which is updated based on real-time traffic conditions.
- *Map Fragment*: graphically shows all the buses currently in transit as they move around, in real-time. The commuter's current location is indicated by a red pin icon and each bus with a black bus icon. When the bus icon on the map is clicked, it will show the bus's route. The user can use a combination of gestures to zoom in and out of the map, to see a detailed view and a broader view. The map was created using the Maps SDK for Android API, which handles access to Google Maps servers, data downloading, map display, and response to map gestures.
- *Lines Fragment*: shows all bus lines available for the commuter to browse. This is shown in the form of a list of bus schedules that can each be downloaded at anytime. To make the navigation easier and improve user experience - a search functionality was added that can be used to filter from hundreds of bus lines to the specified bus lines in the search term.

Shown in the Figures 8, 9 and 10 is the bus lines tab in action.


10:36

Jane Dore

Botanic Gardens
Click to download line timetable

Chesterville
Click to download line timetable

Durban North-Umhlanga Rocks
Click to download line timetable

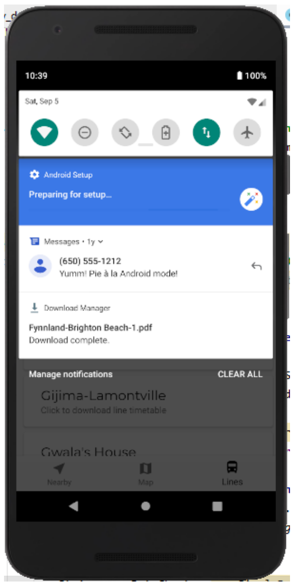
Fynnland-Brighton Beach
Click to download line timetable

Gijima-Lamontville
Click to download line timetable

Gwala's House

Downloading timetable

Fig. 8. Timetable Download


10:39 100%

Sat, Sep 5

Android Setup
Preparing for setup...

Messages • 1y
(650) 555-1212
Yumm! Pie à la Android mode!

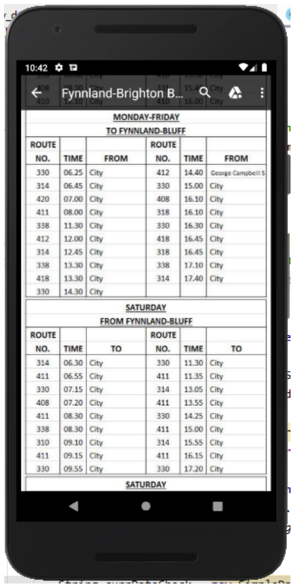
Download Manager
Fynnland-Brighton Beach-1.pdf
Download complete.

Manage notifications CLEAR ALL

Gijima-Lamontville
Click to download line timetable

Gwala's House

Fig. 9. Download Complete


10:42

Fynnland-Brighton B...

MONDAY-FRIDAY
TO FYNNLAND-BLUFF

ROUTE NO.	TIME	FROM	ROUTE NO.	TIME	FROM
330	06.25	City	412	14.40	George Campbell S
314	06.45	City	330	15.00	City
420	07.00	City	408	16.10	City
411	08.00	City	318	16.10	City
338	11.30	City	330	16.30	City
412	12.00	City	418	16.45	City
314	12.45	City	318	16.45	City
338	13.30	City	338	17.10	City
418	13.30	City	314	17.40	City
330	14.30	City			

SATURDAY
FROM FYNNLAND-BLUFF

ROUTE NO.	TIME	TO	ROUTE NO.	TIME	TO
314	06.30	City	330	11.30	City
411	06.55	City	411	11.35	City
330	07.15	City	314	13.05	City
408	07.20	City	411	13.55	City
411	08.30	City	330	14.25	City
338	08.30	City	411	15.00	City
310	09.10	City	314	15.55	City
411	09.15	City	411	16.15	City
330	09.55	City	330	17.20	City

SATURDAY

Fig. 10. Timetable Viewer

When the user has found the bus schedule/timetable they want, they can click on its name, and it will be downloaded in a PDF format, as shown in sequential order in Figures 8, 9 and 10. As there are hundreds of bus schedules PDF files available, this requires a lot of storage space and could make the application bulky. In [32], the authors claim that one of the main reasons users do not download applications is because they do not have enough storage on their devices. If downloading an application means having to sacrifice precious photos or messages, the users are not likely to proceed with that download. So to keep the size of our application at a minimal, we stored those bus schedules on the Firebase Storage service. Each bus schedule will only be downloaded to the device when requested by the user and can be deleted anytime to free up space without having to delete the whole application.

Driver For the bus driver, the main purpose of the mobile application is to record daily ridership and allow the bus's location to be tracked in real-time. The application can be in one of two states, that can be changed with the 'Go Online/Offline' button:

– *Offline State*: in this state, shown in Figure 11, the bus's location is not being tracked, which means it will not be visible to commuters. The bus driver should be in this state when they are not in service. This also means the bus driver will not be able to board commuters or record ridership, as shown in Figure 12.

- *Online State*: in this state, shown in Figure 13, the bus's location is being actively tracked, and the bus is visible to all nearby commuters. The bus driver will be able to record ridership in this state.

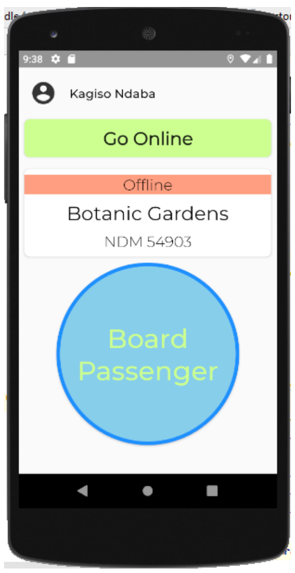


Fig. 11. Offline State

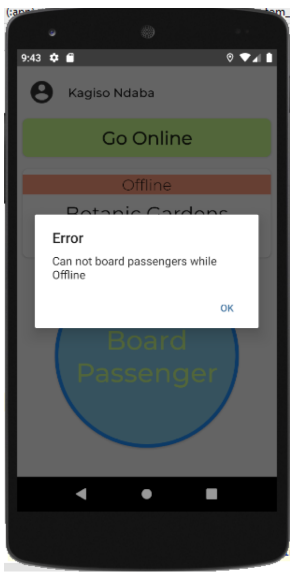


Fig. 12. Board Error

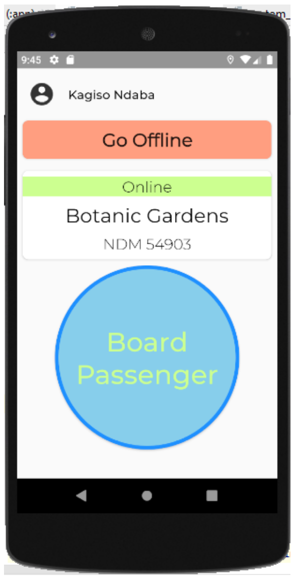


Fig. 13. Online State

The user interface for the bus driver is designed to be as simple as possible so as not to distract the driver. To record ridership, the bus driver simply taps on the 'Board Passenger' circle button the number of times that correspond to the number of passengers being boarded, e.g. when the bus stops and three passenger board, then the bus driver will tap on the button three times.

As mentioned previously, the bus driver is required to enter the bus code when signing up. The bus code is used to uniquely identify each bus, and can only be obtained by the bus drivers from the bus operator or manager. The bus driver may be assigned to a different bus after they have signed up; in that instance, the 'Change Bus' option would be used to change to a different bus, given that the bus code entered is valid. This process is shown in order, in Figures 14, 15 and 16.

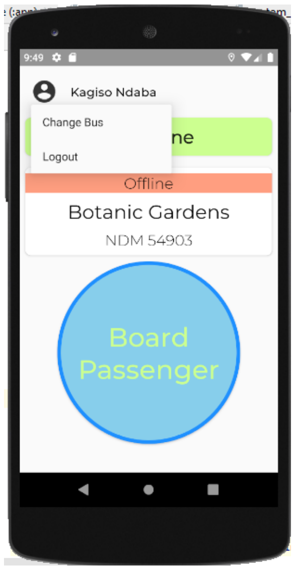


Fig. 14. Bus Change Option

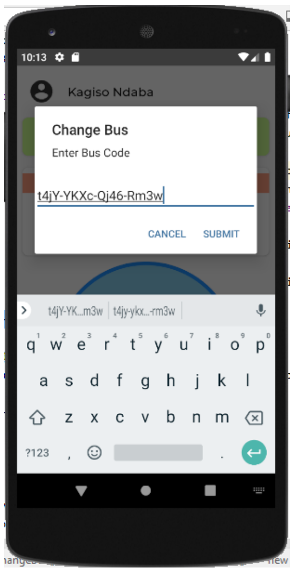


Fig. 15. Bus Code Input

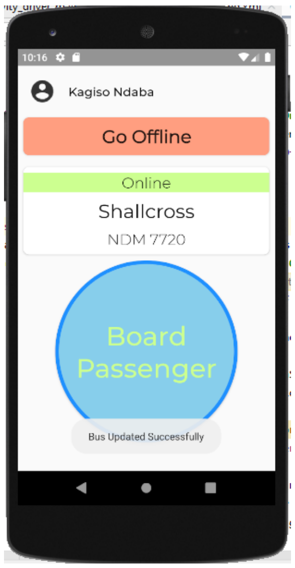


Fig. 16. After Bus Change

3.4 Web Application

The second subsystem of Irenbus is a web application developed mainly with pure JavaScript and a few other technologies. It is hosted on the Firebase Hosting service, which provides production-grade web content hosting and automatically provisions and configures an SSL certificate. The primary purpose of this application is to provide bus managers and operators with the ability to monitor buses, drivers and perform administrative tasks.

Authentication The web application only has one type of user - the bus operator/manager - with full administrative access. Hence proper functioning security features are mandatory for this application. The code snippet below shows how authentication state persistence is handled in the application using the Firebase JavaScript SDK.

Driver and Bus Management The Map tab on the web application provides an overview of all online buses currently in transit. This is presented in a minimally styled map, developed using the Maps JavaScript API, as shown in Figure 17. The bus icons on the map are updated in real-time, and they show the bus as it moves. For a detailed view, the user can click on the bus icon to view where the bus is heading, the number plate and the current driver's full name. This map view will also come in handy when emergency situations arise, as the bus's precise location can be tracked. The user can pan around, zoom in and out and set the map to a full-screen view if required.

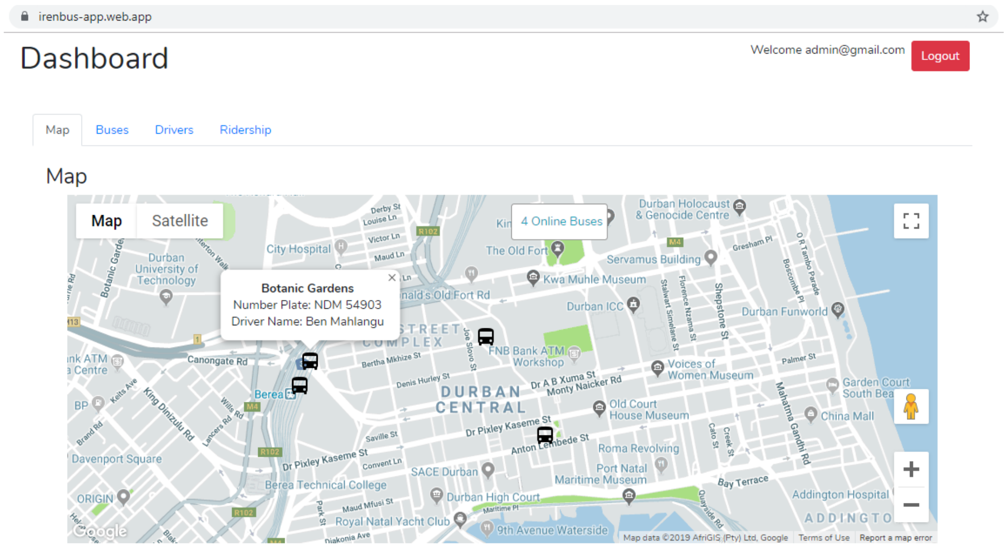


Fig. 17. Online buses overview

Additionally, Google Street View is integrated into the map, which enables users to view and navigate through a 360-degree horizontal and a 290-degree vertical panoramic street-level imagery of cities.

The Buses tab lists all the buses in the system with their corresponding route. This tab also allows new buses to be added and existing ones to be updated; these changes will reflect in real-time across all devices and platforms using Irenbus.

The Drivers tab shows a searchable list of all registered drivers within the Irenbus system, which includes their picture, full name and the bus code of the bus they are currently assigned to.

Ridership Forecasting The Ridership tab shows the predicted ridership figures for a given route. These predictions are made by a continuously trained Keras model, described in detail in subsection 3.5. These predictions help bus managers/operators to dynamically assign drivers and buses to routes based on expected demand. Furthermore, this will potentially improve the availability and service of buses to commuters. An example of these predictions is shown in Figure 18.

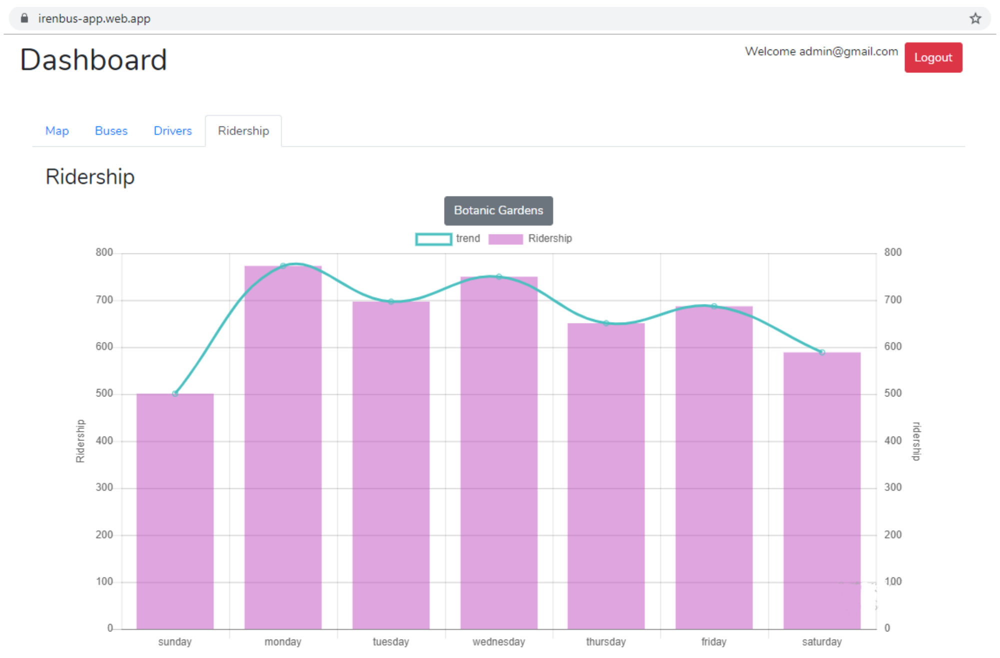


Fig. 18. Ridership forecast for the Botanic Gardens route

The graph representation of the predictions was achieved through the use of `chart.js`, which is an open-source JavaScript library for data visualization. This library provides excellent rendering performance across all modern browsers and redraws charts on window re-size events for perfect scale granularity.

3.5 Machine Learning Pipeline

The machine learning pipeline for the Irenbus system attempts to provide Continuous Delivery for Machine Learning (CD4ML) which is a software engineering approach in which a cross-functional team produces machine learning applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles [33]. Our proposed approach is shown in Figure 19.

Data collection: refers to the collection of daily ridership data, which is carried out by bus drivers through the use of the mobile application. This data is continuously saved on the Realtime Database.

Data Extraction and Analysis: the collected ridership data on the Realtime Database is periodically exported in a JSON format, and then converted into CSV with tools such as `https://konklone.io/json/`. The CSV is then analysed and checked for inconsistencies.

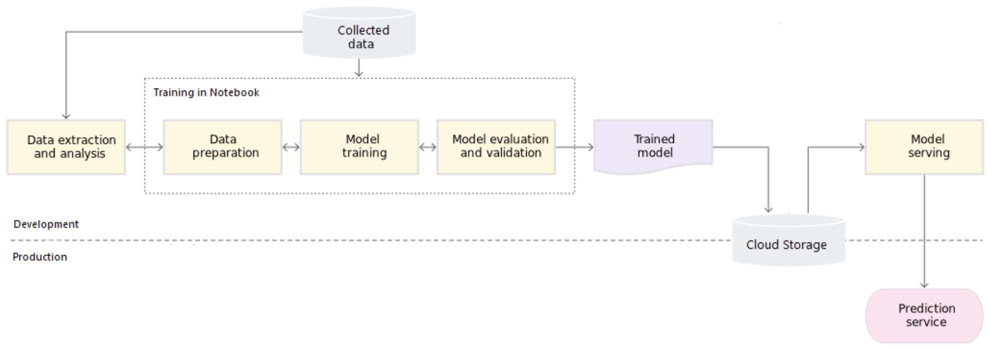


Fig. 19. Irenbus Machine Learning Pipeline

Model Training: is carried out on a Python Notebook using the CSV file with ridership data from the previous step. The resulting Keras model is then converted from the HDF5 format to a JSON format as shown in listing 1.1.

Listing 1.1. Training in Python

```
# Define Keras model
model = ...

# Train the model
model.fit( ... )

#Convert the Python model to a JavaScript model
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "/content/")
```

Trained model: the Keras model (`model.json`) is produced after training is uploaded to the Firebase's Cloud Storage service. The sample dataset and detailed description, training and evaluation of the model used in our proposed system is described in subsection 4.3.
Prediction Service: is carried out on the web application by a JavaScript function `forecastRidership(...)`, which takes values shown in the listing 1.2 and returns the predicted ridership value through the use of the Keras model saved on Cloud Storage.

Listing 1.2. Ridership forecast function in the Web Application

```
//JavaScript function that makes a prediction using model.json
async function forecastRidership( route , dayOfWeek , dayOfMonth ,
    ↪ dayType){
  const model = await loadLayersModel('https://firebasestorage.
     googleapis.com/v0/b/irenbus-app.appspot.com/o/model.json?
     alt=media&token=b06c88a6-5c29-4266-8a1c-81997ce0de95');
  const prediction = model.predict([[route , dayOfWeek , dayOfMonth ,
      ↪ dayType]]);

  return prediction;
}
```

4 Evaluation, Testing and Results

In this section, we present the results of qualitative and quantitative evaluations of our proposed real-time public transport system.

4.1 Mobile Application

To test our Android application, we used the Android Studio Emulator and Firebase's Test Lab. The Android Emulator simulates Android devices on the computer, which allows applications on a variety of simulated devices and Android API levels without needing to have each physical device. The Emulator provides the ability to simulate incoming phone calls and text messages, simulate different network speeds, specify the location of the device, simulate rotation and other hardware sensors [34]. Firebase Test Lab is a cloud-based app-testing infrastructure that uses real, production devices running in a Google data center [35]. The Test Lab was used to supplement the Emulator with its test automation features and scalable testing.

Location Simulation Since our mobile application depends heavily on location tracking, we needed to test if our application reported valid and accurate location points across all connected devices. We needed a cost-effective and reproducible approach to testing location tracking instead of physically driving around with the device, so we made use of the GPS Data Playback to simulate a bus moving around the city.
Figure 20 shows the simulated route of the bus on an actual map. During the simulation, the bus location updates on the Realtime Database were monitored closely, and there were no abnormalities in the reported location points. The simulation was run for the second time, on which we now monitored the location updates on the web application dashboard map, and no errors on the bus's path were observed. So when the bus driver is online, and the bus's location is being tracked, we expect no errors, as observed in the simulation.

Robo Test To ensure that the mobile application users did not encounter unexpected results or have a poor experience when interacting with our application, we subjected it to several Robo tests. These are automated tests that analyse the structure of the user interface, and they explore it methodically, simulating user activities. Robo tests are made possible by the Espresso and UI Automator

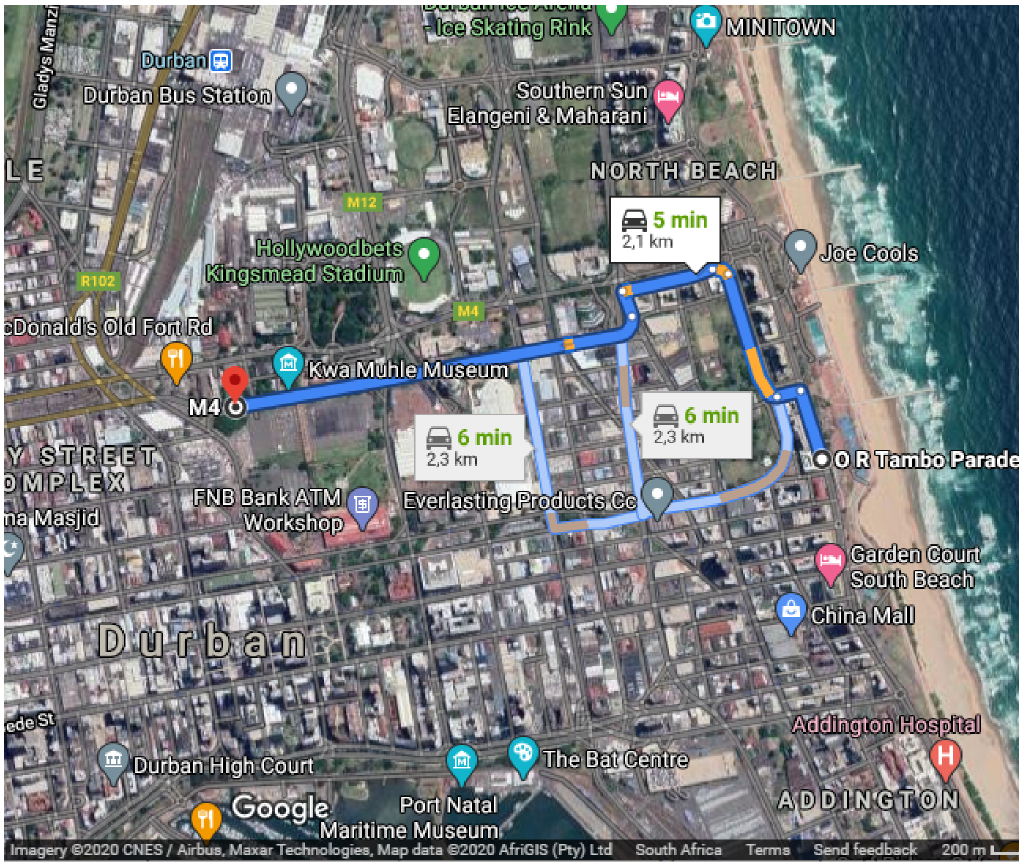


Fig. 20. Played Back Bus Route

2.0 user experience testing frameworks.

Two tests were carried out separately as Robo tests on a Galaxy S7 Edge, Android API Level 23, to assess CPU usage, Memory consumption and Network usage in the two main activities of the Irenbus mobile application, which are the Commuter Activity and the Driver Activity.

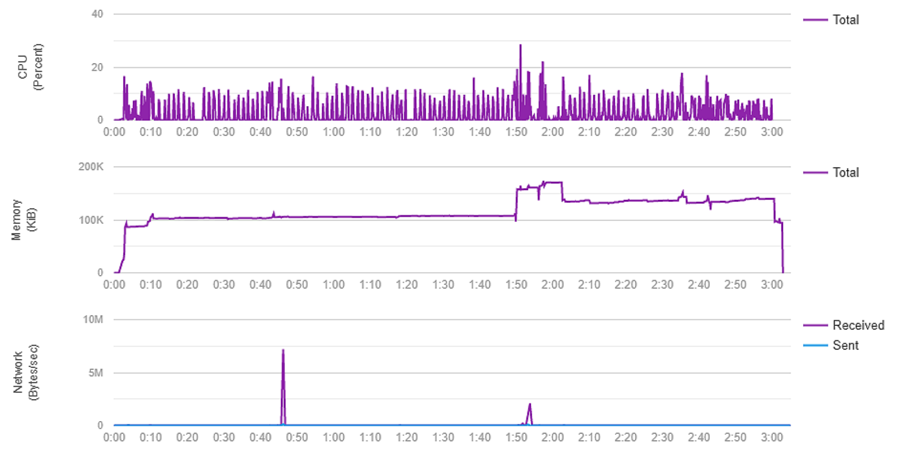


Fig. 21. Commuter Activity CPU, Network and Memory statistics over a period of 3 min 4 sec

Furthermore, as shown in Figures 21 and 22, the Irenbus mobile application does not consume a lot of system resources and utilizes very little network bandwidth. As a result, the application is unlikely to lag or stop responding while being used, and it consumes very little mobile data, which makes it more efficient. These factors will ensure an excellent overall user experience.

4.2 Web Application

The web application performance was evaluated using Google's PageSpeed Insights tool, which reports a performance score of a web application on both mobile and desktop devices. This score is determined by running Lighthouse, which is an opensource, automated tool for improving the quality of web applications.

Six metrics were used to assess the performance of the application. First Contentful Paint (FCP) marks the time at which the first text or image is painted. Time to Interactive is the amount of time it takes for the page to become fully interactive. Speed Index shows how quickly the contents of a page are visibly populated. Blocking Time is the sum of all time periods between FCP and Time to Interactive when task length exceeded 50ms, expressed in milliseconds. Largest Contentful Paint marks the time at which the most extensive text or image is painted. Cumulative Layout Shift measures the movement of visible elements within the viewport. These metrics were selected because they are user-centric, as mentioned in [36].

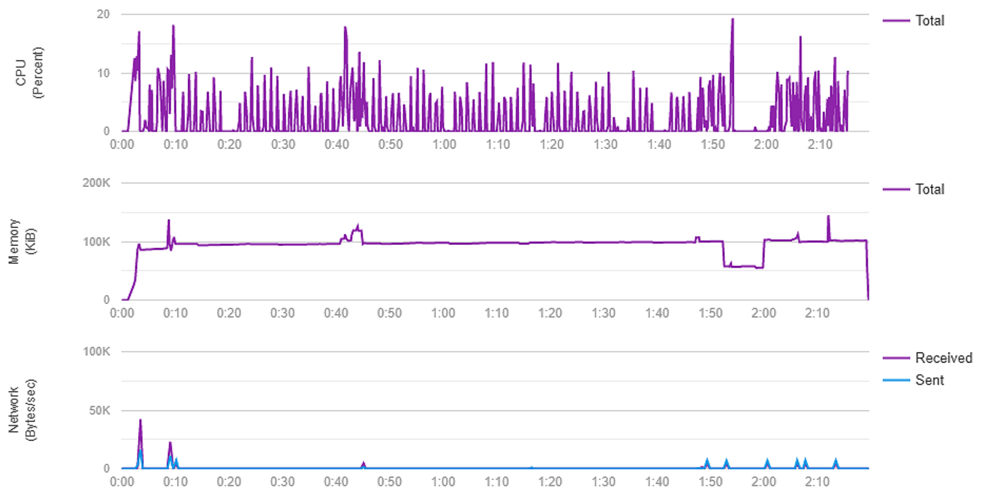


Fig. 22. Driver Activity CPU, Network and Memory statistics over a period of 2 min 19 sec

Table 2. PageSpeed Insights Results

	Device Type	
	Mobile	Desktop
First Contentful Paint	3.9 s	0.9 s
Time to Interactive	5.2 s	1.0 s
Speed Index	4.0 s	0.9 s
Total Blocking Time	300 ms	20 ms
Largest Contentful Paint	5.2 s	1.3 s
Cumulative Layout Shift	0	0
Overral Score	**65**	**95**

The results of the performance assessment are shown in the Table 2 above. The web application obtained a relatively better score on desktop devices than on mobile devices, which was expected as the web application was developed to be used on desktops by bus managers/operators. To improve the LCP the amount of imagery and styling can be reduced, but all other metrics reported good results on desktop.

4.3 Ridership Forecasting

In this section, we aim to describe and evaluate the machine learning model integrated into our proposed public transport system. Since our machine learning model required data to be trained on, we chose to utilize an already existing real-world public transport dataset instead of collecting the ridership data ourselves with the mobile application, which would have been very demanding and costly.

Dataset Our dataset was obtained from the Chicago Transit Authority's open data portal, which is an operator of mass transit in Chicago, Illinois and surrounding suburbs with a fleet of 1879 buses and a 242 million annual bus ridership [37] [38]. The obtained data shows the daily ridership total for each route from mid-2019 back to 2001. The dataset was filtered and only left relevant columns, namely ridership (daily), day type (working day or not), day of the month, day of the week and route (number). Figure 23 shows the value range and distribution for each attribute in the dataset.

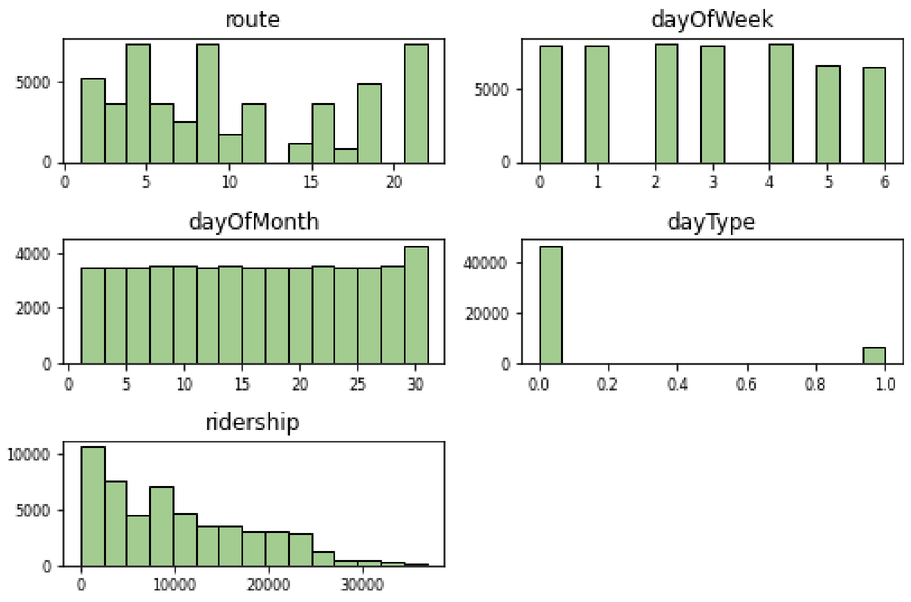


Fig. 23. Dataset Columns Histograms

Preprocessing The varying scale of values in our dataset led to us resorting to preprocessing to resolve this, and it is best practice to prepare the data before modelling it with a neural network model. The quality of data that a machine learning model is trained with greatly affects the resulting model. The following two well-known methods were employed to rescale the dataset's attributes:

- *Normalization*: the data is rescaled from the original range in such a way that all attributes have values within the range of 0 and 1. A normalized value of an attribute is calculated as $z = \frac{x - min(x)}{max(x) - min(x)}$.
- *Standardization*: is rescaling the distribution of values so that the mean of observed values is 0 with a standard deviation of 1. A standardized value is obtained using $z = \frac{x - mean(x)}{standard_deviation(x)}$.

Model Architecture Our machine learning model is the Keras' Sequential model with four densely connected hidden layers, with a single output that returns a continuous value. The input layer takes in four values - for our system; these are route, dayType, dayOfWeek and dayOfMonth. The first hidden layer has four inputs and 50 output nodes. The second hidden layer has 50 inputs and 100 output nodes. The third hidden layer has 100 inputs and 50 output nodes. The output layer has 50 inputs and one output node - which is the ridership prediction - as shown in Figure 24.

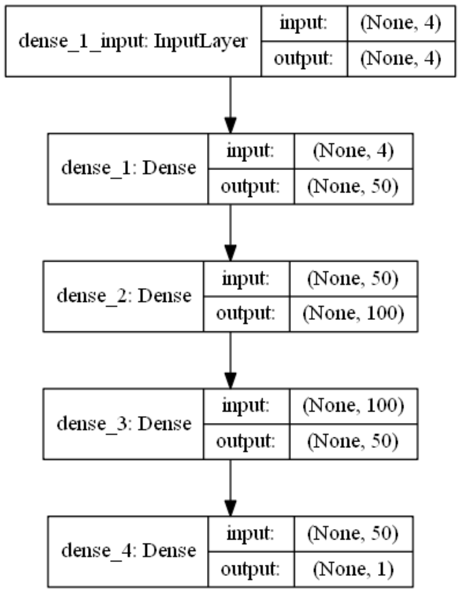


Fig. 24. Model Architecture

The learning algorithm used to adjust weights in the network was the Adaptive moment estimation (Adam) optimization algorithm, chosen because of its computational efficiency, little memory requirements and straightforward implementation.

Training The aforementioned two preprocessed (normalized and standardized) datasets and the unaltered dataset were used in the training process. The training was done three times, each time with a different dataset so that we would have three resulting models. This was done so we could observe the effect of preprocessing the dataset on the model's ability to learn, and we could pick the best model to be used in our system. The models were trained on Google Colab - an online Python Notebook - with the system specification shown in Table 3.

Table 3. Training System Hardware Specifications

GPU Model	NVIDIA Tesla T4 16GB
CPU Model	Intel(R) Xeon(R)
Sockets (CPU Slots)	1
Cores (per Socket)	1
Threads (per Core)	2
L3 Cache	56320K
CPU MHz	2200.00
Memory	13GB
Hard Disk Space	37GB

The model training was carried out for 150 epochs, which was found to be optimal. Each model used the Adam optimization algorithm, which automatically tunes itself to provide better results. The training used 80 per cent of the dataset, and testing used the remaining 20 per cent. For validation, we use 30 per cent of the 80 per cent for training. Our batch size was set to 10 samples. To determine to the optimal number of epochs to be used in the training process, for each model we experimented with the number of epochs starting from 50 and increasing by 50 in each try while observing the amount of loss from each model and when we reached 150, the models showed stability. The graphs in Figures 25, 26 and 27 show each model's Mean Square Error fluctuations for a duration of 150 epochs, which indicates how well the model was learning.

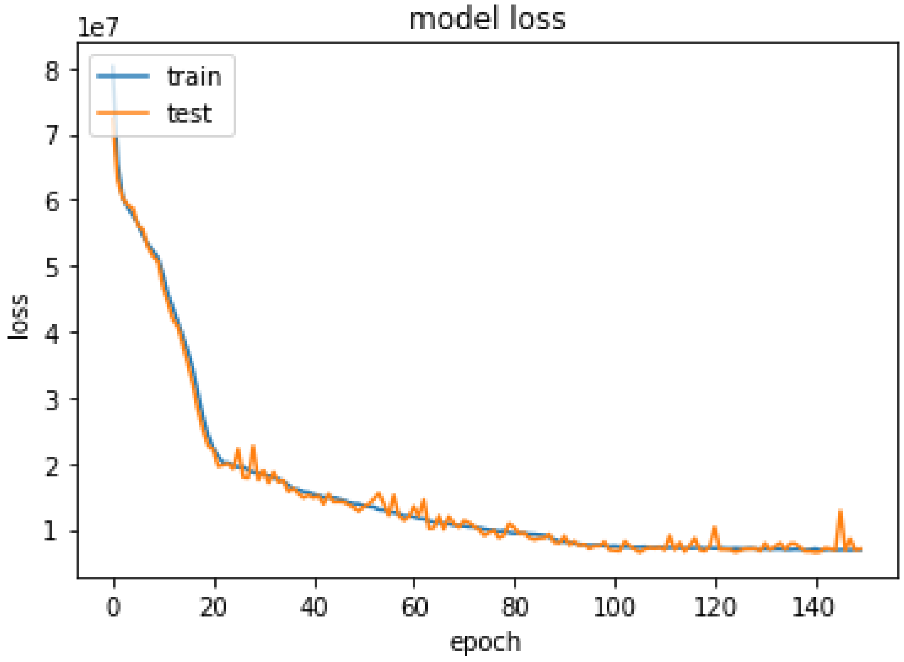


Fig. 25. MSE for the model trained with an Unscaled Dataset

The model's 150 epochs training for the Unscaled, Standardized and Normalized datasets took 14 mins 32 secs, 13 mins 49 secs and 13 mins and 20 secs respectively. Generally, the lower the loss, the better the model is unless the model has overfitted the training data, which was not the case for our models since they were all tested with unseen data. The standardization of the dataset has dramatically improved the model's learning ability in comparison with normalization. The non-preprocessed dataset visibly showed a more significant loss, hence more inferior learning for the model as expected in comparison to the preprocessed ones. The best model, which was trained from the standardized dataset, was used by our web application to forecast future ridership.

4.4 Overall System Evaluation

The quality of a system is the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value [39]. The most prominent models for assessing software quality include the Boehem, McCall, FURPS, Dromey, ISO 9126 and the recent ISO 25010 which is defined as the cornerstone of a product quality evaluation [40–44]. The ISO 25010 is an international standard that defines software quality in two models, viz. quality in use and product quality. The quality in use model is composed of five characteristics (further subdivided into sub-characteristics) that relate to the

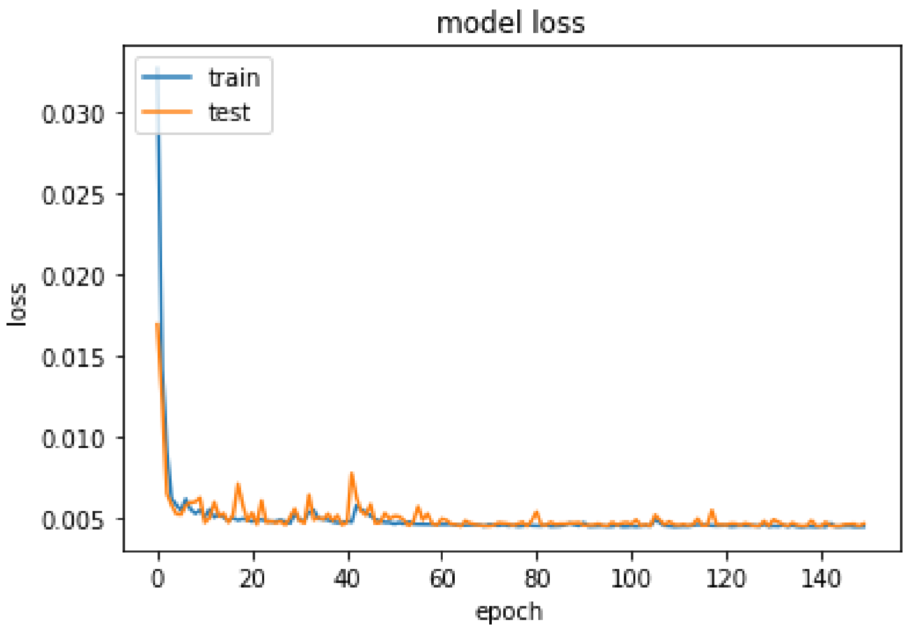


Fig. 26. MSE for the model trained with an Normalized Dataset

outcome of interaction when a product is used in a particular context of use. The product quality model is composed of eight characteristics (further subdivided into sub-characteristics) that relate to static properties of software and dynamic properties of the computer system [45].

The Irenbus system was rated based on the ISO 2501O standards by five volunteers. For an accurate assessment, the selected volunteers were experienced in software design and development. Experienced volunteers were selected because most of the ISO 25010 standard metrics are technical. Due to the broadness of the Irenbus system - as it has three types of users viz. the commuter, the bus driver and the bus manager - each volunteer was provided with authentication credentials for all three system users and asked to assess the system from the perspective of each of the users. So each of our volunteers completed three different assessments, and that gave us fifteen assessments in total. The assessments contained sub-characteristics of the aforementioned IS0 25010 characteristics, and the volunteers provided rating using the Likert Scale (1-Not Satisfied, 2-Less Satisfied, 3-Neutral, 4-Satisfied, 5-Very Satisfied) on how much each sub-characteristic was satisfied by the Irenbus system. All assessments' results were compiled into Table 4, with μ being the mean and σ being the standard deviation of ratings for each sub-characteristic.

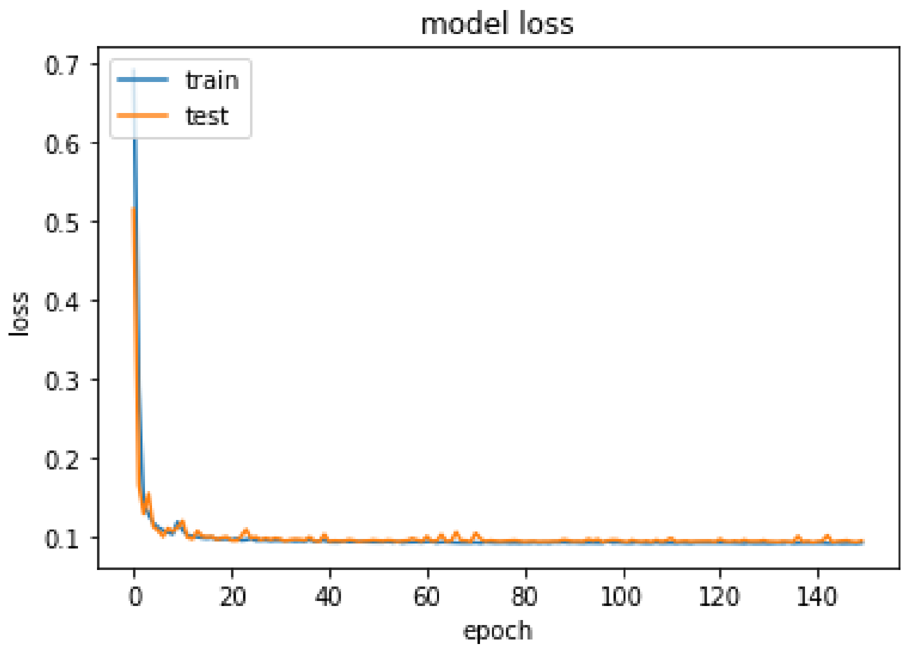


Fig. 27. MSE for the model trained with an Standardized Dataset

Table 4: Irenbus System Evaluation

		Rating	
Characteristic	**Sub-Characteristic**	μ	σ
Functional Suitability	Functional completeness	4.73	0.46
	Functional correctness	4.60	0.63
	Functional appropriateness	4.67	0.62
Performance Efficiency	Time behavior	4.27	0.80
	Resource utilization	3.47	0.52
	Capacity	4.67	0.49
Compatibility	Co-existence	4.80	0.41
	Interoperability	4.87	0.35
Usability	Appropriateness recognizability	4.47	0.74
	Learnability	4.87	0.35
	Operability	4.67	0.72
	User error protection	4.67	0.62
	User interface aesthetics	4.93	0.26
	Accessibility	4.80	0.56
Reliability	Maturity	4.67	0.49
	Availability	4.80	0.41
	Fault tolerance	4.47	0.63

	Recoverability	4.60	0.51
Security	Confidentiality	4.87	0.35
	Integrity	4.73	0.46
	Non-repudiation	4.80	0.41
	Accountability	4.73	0.46
	Authenticity	4.87	0.52
Maintainability	Modularity	4.67	0.62
	Reusability	4.73	0.59
	Analyzability	4.67	0.72
	Modifiability	4.67	0.49
	Testability	3.87	0.92
Portability	Adaptability	4.87	0.35
	Installability	4.13	0.83
	Replaceability	4.20	0.86

The volunteers evaluated our proposed and scored the system on each characteristic on its feasibility in the South African context. It is worth noting that in our results in Table 4, the ratings in all characteristics had a very low standard deviation, they were all less than one, which implies that our mean ratings resemble individual volunteer ratings closely. Based on the results, our proposed system (Irenbus) is functionally suitable for the needs of commuters, bus drivers and bus operators. The performance efficiency of our system was rated high except for resource utilization where volunteers reported heavy battery drainage on their mobile devices when logged in as a bus driver; this was due to background location tracking processes on which the GPS receiver - the small chip antennae in the mobile device - was always listening to the cell towers to decide where the device was located geographically at all times [46]. Furthermore, this is a prevalent issue in most location tracking mobile applications, for example, the popular cab-hailing application - Uber - explains in [47] how they are attempting to solve this problem. The Irenbus system achieved relatively good ratings on compatibility, reliability, security and the usability sub-characteristic - user interface aesthetics scoring the highest rating. Our system's maintainability and portability characteristics achieved decent ratings; this implies that our system can be deemed feasible and cost-efficient. Overall, the ratings show that the volunteers were fairly satisfied with our system.

5 Discussion and Conclusion

This research was aimed at demonstrating a practical and implementable intelligent public transport management system in the context of developing countries.

Five objectives were listed as part of this research. First, designing and developing an Android application to inform commuters about the current status of a given bus through live location tracking. The second objective was developing an approach to collect daily ridership through the use of only a mobile application and no external hardware modules. Third, developing an approach to incorporate a continuously trained machine learning to forecast ridership per route. Fourth, designing and developing a web-based application to monitor buses, drivers and present the machine learning model's predictions to bus operators. The fifth objective was evaluating the feasibility of our proposed system.

As this research aimed to develop our system in a way that it could be easily implemented in South Africa and other developing countries, we needed to make sure that unnecessary implementation and maintainability costs were reduced as much as possible. This then led to our proposed system being purely cloud-based and run on just a mobile application and requiring no extra external hardware components. The reason we chose smartphones as the platform to implement our system is because more than 91% of the South African population owned as smartphone as of 2019, 3G coverage was almost a 100% and 4G/LTE coverage was almost 93% in 2019 [6]. Furthermore, with our system being cloud-based, this means system updates and new features can be rolled out effortlessly without having to physically tinker with the device. The system evaluation in subsection 4.4 showed that experienced software developers who volunteered to assess the Irenbus system deemed it feasible within the South African context.

The contribution of this research was to fill in the gap in previously published work with the development of a 360 degree and cohesive public transportation system that caters for the needs of commuters and bus operators; and the exploration and development of a daily ridership collection approach; the use of a machine learning model to predict future ridership based on the data collected by the system itself. Our system took full advantage of cloud computing services while requiring only a smartphone to work - eliminating the need and costs of installing and maintaining separate GPS trackers on the vehicles.

This research was limited to demonstrating our system on only one mode of public transport - buses; this was done to simplify the scope of our system for now. However, our system can be implemented for other modes of public transport.

The functionality of our proposed system (Irenbus) can be improved by incorporating other modes of public transport; creating a communication channel between the bus managers and commuters in the form of a noticeboard for public announcements; by having a mobile application for not only Android but iOS devices too; and by also exploring other machine learning models with the aim of improving our ridership predictions.

Appendix

The source code, documentation, Python Notebook and training data for Irenbus is available in the following GitHub repository: https://github.com/m3n2ie/Irenbus

References

1. Statistics South Africa. *Measuring Household expenditure on public transport*;. [Accessed 3rd March 2020]. [Online]. Available from: http://www.statssa.gov.za/?p=5943.
2. Stockin D. *Does Adding an Extra Driving Lane Make Traffic Worse?*;. [Accessed 29th September 2020]. [Online]. Available from: https://drivetribe.com/p/does-adding-an-extra-driving-lane-E6FPiVJnQSCPun1-pS-Q-A?iid=LEiNsk8oQqOPNAOQ1{_}hVqg.
3. Hanna NK. In: E-Transformation: Enabling New Development Strategies; 2010. p. 281–309.
4. European Union. *How to get cities moving: Public transport challenges in developing countries*;. [Accessed 25th July 2020]. [Online]. Available from: https://europa.eu/capacity4dev/articles/how-get-cities-moving-public-transport-challenges-developing-countries.
5. United Nations Department of Economic and Social Affairs. *68% of the World Population Projected To Live in Urban Areas By 2050*;. [Accessed 17th March 2020]. [Online]. Available from: https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.
6. ICASA. The state of the ICT sector in South Africa. Independent Communications Authority of South Africa. 2020;(March):83. Available from: https://www.icasa.org.za/uploads/files/State-of-ICT-Sector-Report-March-2018.pdf.
7. Almeida Motta R, Da Silva PCM, De Sequeira Santos MP. Crisis of public transport by bus in developing countries: A case study from Brazil. International Journal of Sustainable Development and Planning. 2013;8(3):348–361.
8. Ngoc AM, Hung KV, Tuan VA. Towards the Development of Quality Standards for Public Transport Service in Developing Countries: Analysis of Public Transport Users' Behavior. Transportation Research Procedia. 2017;25:4560–4579. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.
9. Pucher J, Korattyswaropam N, Mittal N, Ittyerah N. Urban transport crisis in India. Transport Policy. 2005 05;12:185–198.
10. Sungur C, Babaoglu I, Sungur A. Smart Bus Station-Passenger Information System. In: 2015 2nd International Conference on Information Science and Control Engineering; 2015. p. 921–925.
11. Manikandan R, Niranjani S. Implementation on Real Time Public Transportation Information Using GSM Query Response System. Contemporary Engineering Sciences. 2014 05;7:509 – 514.
12. Shirisha K, Sivaprasad T. Acquire Bus Information using GSM Technology. International Journal of Advancements in Technology. 2016 01;7.
13. Kumbhar M, Survase M, Mastud P, Salunke A. Real Time Web Based Bus Tracking System. International Research Journal of Engineering and Technology. 2016;5(10):632–635.

14. Skhosana M, Ezugwu AE. Irenbus: A Real-Time Public Transport Management System. In: 2020 Conference on Information Communications Technology and Society (ICTAS); 2020. p. 1–7.
15. Skhosana M, Ezugwu AE, Rana N, Abdulhamid SM. An Intelligent Machine Learning-Based Real-Time Public Transport System. In: O G, editor. Computational Science and Its Applications – ICCSA 2020. Cham: Springer International Publishing; 2020. p. 649–665.
16. Ryu S, Park BB, El-Tawab S. WiFi Sensing System for Monitoring Public Transportation Ridership: A Case Study. KSCE Journal of Civil Engineering. 2020;24(10):3092–3104.
17. Monchambert G, de Palma A. Public transport reliability and commuter strategy. Journal of Urban Economics. 2014;81(C):14–29.
18. Bacciu D, Carta A, Gnesi S, Semini L. An experience in using machine learning for short-term predictions in smart transportation systems. Journal of Logical and Algebraic Methods in Programming. 2017;87:52–66.
19. Hsu YW, Chen YW, Perng JW. Estimation of the Number of Passengers in a Bus Using Deep Learning. Sensors (Switzerland). 2020 04;20:2178.
20. van Oort N, Brands T, de Romph E. Short-Term Prediction of Ridership on Public Transport with Smart Card Data. Transportation Research Record: Journal of the Transportation Research Board. 2015 01;2535:105–111.
21. MyCiti. *Using MyCiti on your phone*;. [Accessed 21st September 2020]. [Online]. Available from: https://www.myciti.org.za/en/discover-myciti/using-myciti-on-your-phone/.
22. McCann A. *Cities with the Best & Worst Public Transportation*;. [Accessed 5th July 2020]. [Online]. Available from: https://wallethub.com/edu/cities-with-the-best-worst-public-transportation/65028/{#}main-findings.
23. Optibus. *The Optibus Platform*;. [Accessed 27th November 2020]. [Online]. Available from: https://www.optibus.com/product/platform/.
24. Frankenfield J. *What is Cloud Computing? - Types of Cloud Computing Services & More*;. [Accessed 3rd November 2020]. [Online]. Available from: https://www.trianz.com/insights/revolution-that-is-cloud-computing.
25. MongoDB. *NoSQL Explained*;. [Accessed 11th August 2020]. [Online]. Available from: https://www.mongodb.com/nosql-explained.
26. Firebase. *Store and sync data in real time*;. [Accessed 3rd July 2020]. [Online]. Available from: https://firebase.google.com/products/realtime-database.
27. Firebase. *Firebase Realtime Database*;. [Accessed 24th October 2020]. [Online]. Available from: https://firebase.google.com/docs/database.
28. Google LLC. *Introduction to Activities*;. [Accessed 26th November 2020]. [Online]. Available from: https://developer.android.com/guide/components/activities/intro-activities.html.
29. Google LLC. *Developer Guides — Android Developers*;. [Accessed 26th November 2020]. [Online]. Available from: https://developer.android.com/reference/android/app/Activity.
30. ÇORAK A. *Splash Screen is More Important Than You Think How to Use*;. [Accessed 21th July 2020]. [Online]. Available from: https://uxplanet.org/splash-screen-is-more-important-than-you-think-855e78da3c2c.
31. Google LLC. *Get started - Directions API*;. [Accessed 27th November 2020]. [Online]. Available from: https://developers.google.com/maps/documentation/directions/start.

32. Wilson M. *App Download Stats Reveal Reason Behind Low Number of Apps Downloaded*;. [Accessed 12th October 2020]. [Online]. Available from: https://www.zipwhip.com/blog/app-download-statistics-reveal-why-people-dont-download-apps/.
33. Sato, Danilo and Wider, Arif and Windheuser, Christoph. *Continuous Delivery for Machine Learning*;. [Accessed 3rd September 2020]. [Online]. Available from: https://martinfowler.com/articles/cd4ml.html.
34. Android Developers. *Run apps on the Android Emulator*;. [Accessed 14th October 2020]. [Online]. Available from: https://developer.android.com/studio/run/emulator.
35. Firebase. *Firebase Test*;. [Accessed 15th November 2020]. [Online]. Available from: https://firebase.google.com/docs/test-lab.
36. Google Developers. *About PageSpeed Insights*;. [Accessed 15th August 2020]. [Online]. Available from: https://developers.google.com/speed/docs/insights/v5/about.
37. Chicago Transit Authority. *CTA - Ridership - Bus Routes - Daily Totals by Route About this Dataset Columns in this Dataset*;. [Accessed 29th March 2020]. [Online]. Available from: https://data.cityofchicago.org/Transportation/CTA-Ridership-Bus-Routes-Daily-Totals-by-Route/jyb9-n7fm.
38. Chicago Transist Athority. *Annual Ridership Report Calendar Year 2015*;. [Accessed 27th April 2020]. [Online]. Available from: https://www.transitchicago.com/assets/1/6/2018_Annual_Report_-_v3_04.03.2019.pdf.
39. SYSQA. *Iso 25010: 2011*;. [Accessed 2nd October 2020]. [Online]. Available from: http://www.gripoprequirements.nl/downloads/iso-25010-2011-een-introductie-v1{_}0.pdf.
40. Boehm BW, Brown JR, Lipow M. Quantitative evaluation of software quality. Proceedings of the 2nd international conference on Software engineering. 1976.
41. Walters GF, McCall JA. Software Quality Metrics for Life-Cycle Cost-Reduction. IEEE Transactions on Reliability. 1979;R-28(3):212–220.
42. Saini R, Dubey SK, Rana A. Analytical Study of Maintainability Models for Quality Evaluation. Indian Journal of Computer Science and Engineering. 2011 06;2.
43. Dromey RG. A model for software product quality. IEEE Transactions on Software Engineering. 1995:146–162. IEEE Transactions on Software Engineering.
44. ISO/IEC. Software engineering - Product quality - Part 1: Quality model. Software Process: Improvement and Practice. 2001;2(1):1–25.
45. ISO/IEC. BSI Standards Publication Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. BSI Standards Publication. 2011:1–4.
46. Shannon L. *Why GPS-dependent apps deplete your smartphone battery*;. [Accessed 9th November 2020]. [Online]. Available from: https://www.theverge.com/2018/8/17/17630872/smartphone-battery-gps-location-services.
47. Hartanto Y and Attwell B. *Activity / Service as a Dependency : Rethinking Android Architecture for the Uber Driver App*;. [Accessed 12th September 2020]. [Online]. Available from: https://eng.uber.com/activity-service-dependency-android-app-architecture/.