International Information and Engineering Technology Association
Advancing the World of Information and Engineering

International Journal of Sustainable Development and Planning
Vol. 16, No. 7, November, 2021, pp. 1221-1238
Journal homepage: http://iieta.org/journals/ijsdp

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

Menzi Skhosana, Absalom E. Ezugwu*

School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, King Edward Avenue, Pietermaritzburg Campus, Pietermaritzburg 3201, KwaZulu-Natal, South Africa

Corresponding Author Email: Ezugwua@ukzn.ac.za

https://doi.org/10.18280/ijsdp.160703

Received: 19 July 2021
Accepted: 7 September 2021

Keywords:
bus location tracking, mobile tracking, short-term forecasting

ABSTRACT

The problems faced daily by the public transportation sector can be addressed or mitigated by collecting appropriate data and applying predictive analytics. This paper primarily focuses on problems affecting the 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 is developed to address the aforementioned issues. The proposed system comprises two subsystems, namely mobile and web applications interfaces. The mobile application interface provides commuters with the current location and availability of a given bus and other related information, and it is also used by drivers so that the bus can be tracked in real-time and collect ridership information throughout the day. Moreover, the web application serves as a dashboard for bus operators to gain insights from the collected ridership data. The new integrated system was developed using the Firebase Backend-as-a-Service (BaaS) platform and integrated with a machine learning model trained on collected ridership data to predict the daily ridership for a given route. The novel system provides a holistic solution to problems in the public transport sector. It is highly scalable, cost-efficient, and takes full advantage of the current technologies compared to other related application platforms.

1. INTRODUCTION

Public transportation efficiency is essential for the economic growth and sustainability of urban areas. However, in developing countries, there are little or no advancements being made in this sector. The sizable capacity of public transport vehicles results in fewer 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. Stockin, in his article [2], outlines that the development of physical infrastructure, e.g. more highways or bigger roads, does not make traffic congestion better but may worsen it; 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 sharing knowledge and information among various societal actors [3] - is just as important. This has led to the recent trend and drive to modernize 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 September 2019, 91.2% of the South African population owned a smartphone, which is nearly a 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, 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 favouring developing countries.

Commuters - mostly in developing countries - are often left stranded at pickup spots and clueless about the availability and proximity of their mode of the public vehicle hence the bad stigma public transport has. This results from the unavailability of real-time information to commuters, poorly managed fleets, varying demands, traffic congestion and rigid schedules. Rapid population growth and urbanization lead to overwhelming travel demands [4, 7]. A better understanding of the commuters' behaviour helps transport operators estimate the demand and 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 for these specific conditions. Previous work done in this sector [10-13] partially offers solutions to the challenges mentioned above but still lacks practicability. 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.

This research demonstrates how a pragmatic intelligent public transport management system (Irenbus) can be developed and implemented, especially in South Africa and other developing countries. To achieve the aim set for the proposed study, this research has been divided into the following specific objectives:

- Design and develop an Android mobile application to inform commuters about the current status of a given bus through live location tracking.
- Develop an approach to collect daily ridership data using only a mobile application without 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 and drivers and present the machine learning model's predictions for a given route to bus operators.
- Evaluate the feasibility of the 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 mainly focus on buses, with three user roles - the bus user or commuter, the bus driver, and lastly, the bus manager or operator 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 and 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 valuable insights based on the ridership patterns to public transport authorities.

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

The remaining part of this paper is structured as follows: Section 2 looks into related work in both published academic work and software products in the industry. Section 3 describes the overall structure of the proposed system - Irenbus. Section 4 presents the results of qualitative and quantitative evaluation of the new real-time public transport system. Section 5 provides a summary of the current 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, the limitations of the 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 commuters with real-time information such as the estimated bus arrival time and the route name. 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 could not provide information to commuters who were not at the bus stop. Furthermore, implementing this system would be costly since every bus stop had an LCD screen mounted.

A Web-based system was developed for tracking buses in transit using a GPS tracker installed on the bus. Users received 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 than other previously done work but still lacked 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 Bacciu et al. [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 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 than winter). The learning models captured these seasonal patterns by appropriately encoding the bike usage time, which explicitly modelled cyclic information such as weekdays and holidays.

In Hsu et al. [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 low resolution scenes and with a variation of illumination, pose and scale. Experimental results showed better performance than previously published results, and they planned to extend the current method with more in-depth learning algorithms. The only drawback to their system was that it might 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 van Oort et al. [20], the authors explored 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 could not 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 McCann et al. [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 from 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.

To the best of the authors' 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, and 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 subsystems, viz. the mobile subsystem and the web subsystem. The mobile system is presented in an Android application platform and has two users, namely, 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; also, the live location of the device will be sent periodically in the background throughout the bus trip. The web subsystem 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 forecasts for a given route.

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, including 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.

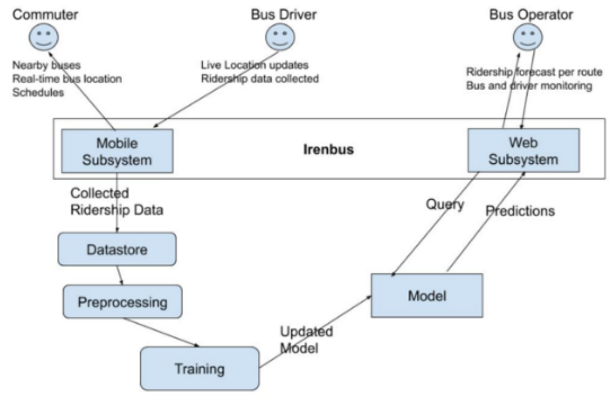


Figure 1. Irenbus system overview

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 various types, with the main types being document, key-value, wide-column, and graph-based databases. After assessing the data collected and stored by the 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 in cases where we need to store large amounts of data, but we do not need to perform complex queries to retrieve it [25]. This type of database is offered as the Realtime Database within the Firebase platform, mentioned in subsection 3.1. The real-time 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: This 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 that resulted from the analysis carried out on the developed system are shown in Table 1.

The proposed Irenbus mobile application dependency graphs and data model representation are presented in Figures 5.1-5.15 under the appendix section.

Table 1. NoSQL database nodes description

Node	Description
Bus	basic bus information 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. The most crucial component of an application in Android development 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 Google LLC [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 application's state when 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: commuters and drivers. They each have different tasks they can carry out on the application, so we added an authentication process to give convenient access to different screens based on what type of user was currently logged in. Figure 2 illustrates some of the proposed system's authentication processes.

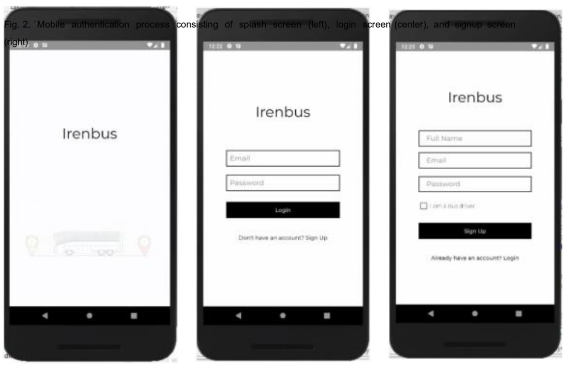


Fig. 2. Mobile authentication process consisting of splash screen (left), login screen (center), and signup screen (right)
Irenbus
Email
Password
Login
Don't have an account? Sign Up
Full Name
I am a bus driver
Sign Up
Already have an account? Login

Figure 2. Mobile authentication process consisting of the splash screen (left), login screen (center), and signup screen (right)

The splash screen is shown when the application is launched, keeping the user company while running a background authentication process. In C¸ORAK [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 enables 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 assigned to is required to prevent unauthorized access.

On launch, the mobile application requests the Firebase Authentication SDK to get the current user, as shown in the code snippet below. If the returned user object is null, the user has not logged in before 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, the user has logged in before. The application will then request the current user's userType value from the database with their userid, which is obtained using firebase user.getUid( ). After the current user's details have been read from the database, the application will access the appropriate screens based on their type.

Upon successful login, the authentication token is stored locally on the device so that 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 can 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; no personal information is required other than an email address 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, an accessibility service that helps blind and visually impaired users interact with their devices. When the commuter is logged in, the bottom navigation offers three tabs which each show the fragments in Figure 3.

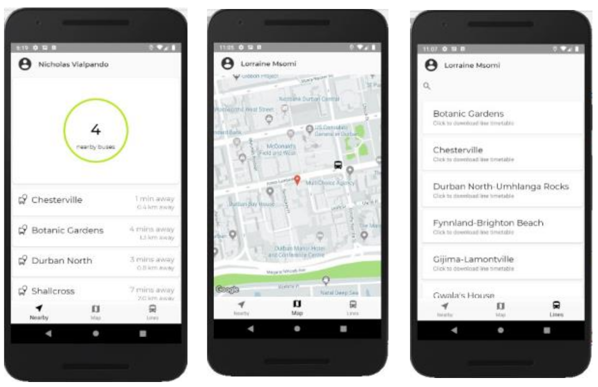


Figure 3. Computer view of the mobile application consisting of the Nearby Fragment (left), Map Fragment (center), and Lines Fragment (right)

Each fragment (Figure 3) in the application for the commuter provides the following functionality:

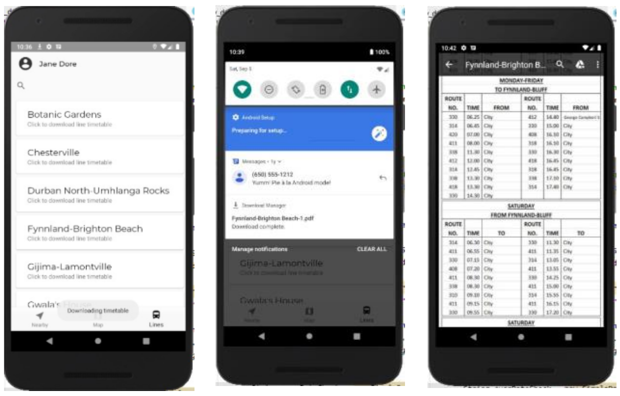
Nearby Fragment: shows all nearby buses, which are buses within a three kilometer radius from the commuter's current location. The distance between each online bus and the commuter is calculated using the Haversine geolocation equation.

$$d = \left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2 \left(\frac{\lambda_2 - \lambda_1}{2}\right)} \right)$$

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 an HTTP interface, with requests constructed as a URL string, and latitude/longitude coordinates to identify the locations [31] - the estimated 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' 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 any time. 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. Figure 4 shows the bus lines tab in action.



MONDAY-FRIDAY TO FYNNLAND-BLUFF					
ROUTE NO.	TIME	FROM	ROUTE NO.	TIME	FROM
330	06.25	City	412	14.40	[illegible]
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

Figure 4. The application bus lines tab consisting of timetable download (left), download complete (center), and timetable viewer (right)

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 Figure 4. As there are hundreds of bus schedules PDF files available, this requires a lot of storage space and could make the application bulky. In Wilson [32], the authors claim that users do not download applications 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 the developed application at a minimum, we stored those bus schedules on the Firebase Storage service. Each bus schedule will only be downloaded to the user's request and can be deleted at any time to free up space without deleting the whole application.

Driver: For the bus driver, the main purpose of the mobile application is to record daily ridership and allow the bus' 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 5, the bus' 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 5.
- Online State: in this state, shown in Figure 5, the bus' 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.

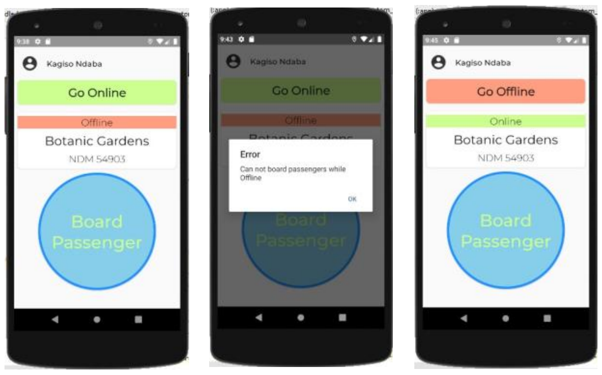


Figure 5. The three application service states: Offline State (left), Board Error (center), Online State (right)

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 taps on the 'Board Passenger' circle button the number of times that correspond to the number of passengers being boarded. For example, when the bus stops and three passengers board, 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 identify each bus uniquely 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 Figure 6.

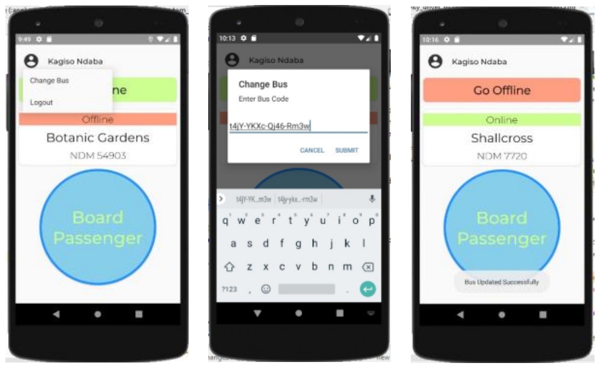


Figure 6. Board Passenger code validation: Bus Change Option (left), Bus Code Input (center), After Bus Change (right)

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 7. 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 emergencies arise, as the bus' 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.

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

The Drivers tab shows a searchable list of all registered drivers within the Irenbus system, including their picture, full name, and bus code of the bus they are 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 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 8.

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]. The study's proposed approach is shown in Figure 9.

Data collection: this refers to the collection of daily ridership data carried out by bus drivers using 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 analyzed and checked for inconsistencies.

Model Training: This is carried out on a Python Notebook using the CSV file with ridership data from the previous step. The resulting Keras model is converted from HDF5 to JSON, as shown in code listing 1.

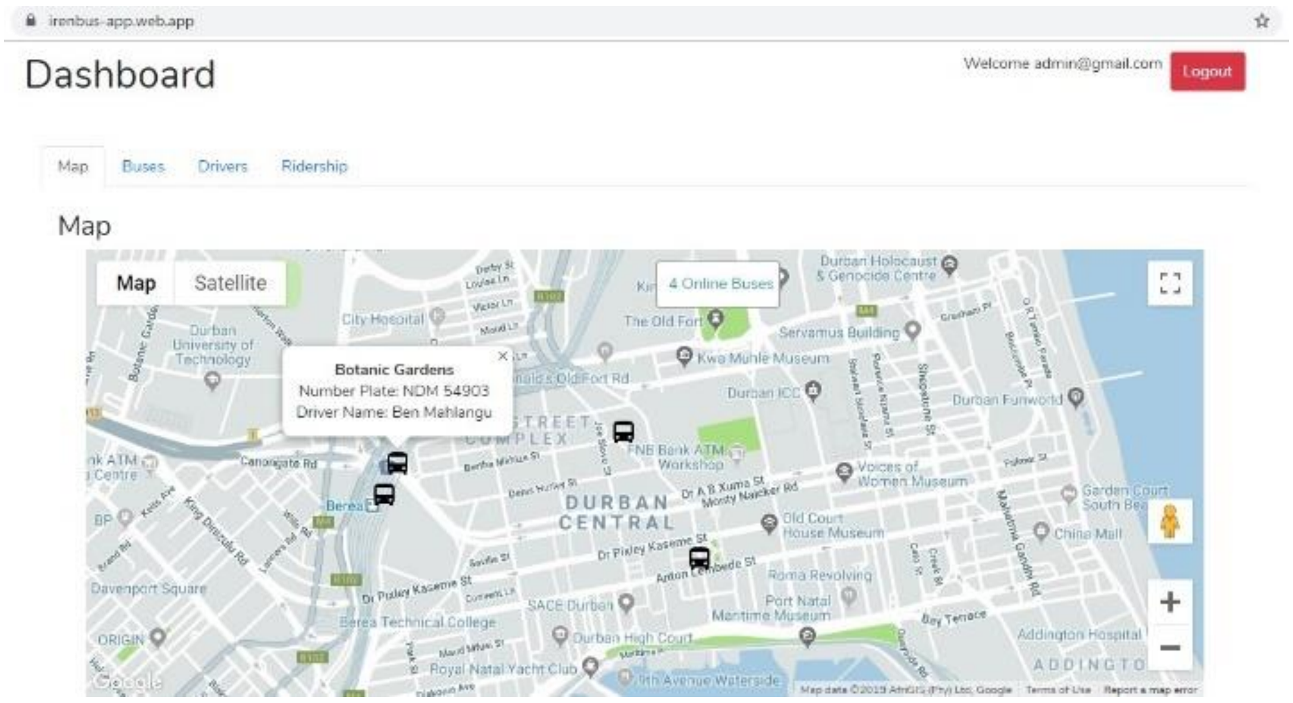


Figure 7. Online buses overview

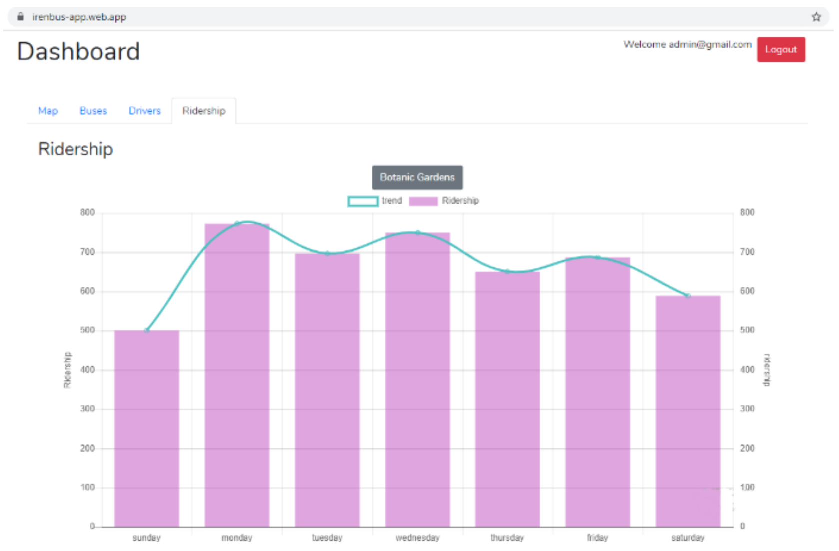


Figure 8. Ridership forecast for the Botanic Gardens route

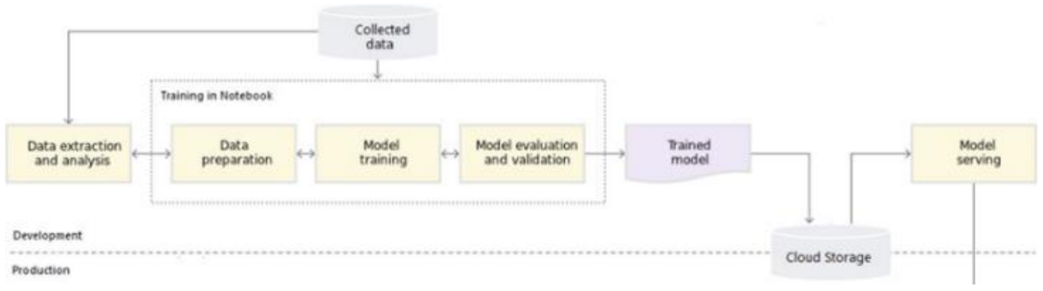


Figure 9. Irenbus machine learning pipeline

Code Listing 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 Firebase's Cloud Storage service. The sample dataset and detailed description, training and evaluation of the model used in the proposed system is described in subsection 4.3.

Prediction Service: this is carried out on the web application by a JavaScript function *forecastRidership(...)*, which takes values shown in code listing 2 and returns the predicted ridership value through the use of the Keras model saved on Cloud Storage.

Code Listing 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

This section presents the qualitative and quantitative evaluations of the newly developed real-time public transport system.

4.1 Mobile application evaluation

We used the Android Studio Emulator and Firebase's Test Lab to test the Android application. The Android Emulator simulates Android devices on the computer, allowing applications on various 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 device's location, and 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.

Figure 10 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' path were observed. So when the bus driver is online and the bus' location is being tracked, we expect no errors, as observed in the simulation.

Location Simulation: Since the mobile application depends heavily on location tracking, we needed to test if the 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 used the GPS Data Playback to simulate a bus moving around the city. The GPS Data Playback allows GPX files to be loaded and played back. A GPX (the GPS Exchange Format) is a lightweight XML data format for the interchange of GPS data (waypoints, routes, and tracks) between applications and Web services on the Internet. The bus' path was loaded into simulated_route.gpx, is as shown in code listing 3.

Code Listing 3: Simulated Route

```
<!--simulated_route.gpx-->
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx  xmlns="http://www.topografix.com/GPX/1/1"
    ↳ xmlns:gpxx="http://www.garmin.com/xmlschemas/GpxExtensions/v3"
    ↳ xmlns:gpxtpx="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
    ↳ creator="mapstogpx.com"  version="1.1"
    ↳  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ↳  xsi:schemaLocation="http://www.topografix.com/GPX/1/1
    ↳ http://www.topografix.com/GPX/1/1/gpx.xsd ...
    ↳ http://www.garmin.com/xmlschemas/TrackPointExtensionv1.xsd">
  <metadata>...
  </metadata>
  <wpt lat="-29.8543894" lon="31.0387423">...
  </wpt>
  <wpt lat="-29.8531588" lon="31.0227668">...
  </wpt>
<trk>
  <name>South Beach to M4</name>
  <number>1</number>
  <trkseg>
  <trkpt lat="-29.85439" lon="31.0387402">
    <name>TP001</name>
  </trkpt>
  .
  .
  .
  <trkpt lat="-29.8531588" lon="31.0227668">
    <name>TP077</name>
  </trkpt>
  </trkseg>
</trk>
</gpx>
```

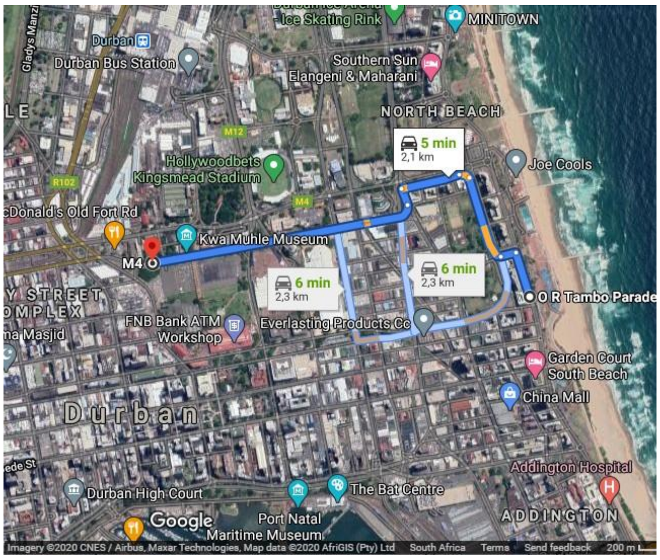


Figure 10. Played back bus route

Robo Test: We subjected it to several Robo tests to ensure that the mobile application users did not encounter unexpected results or had a poor experience when interacting with the newly developed application. These are automated tests that analyze the user interface structure and explore it methodically, simulating user activities. Robo tests are made possible by the Espresso and UI Automator 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. These are the Commuter Activity and the Driver Activity, as shown in Figures 11 and 12.

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.

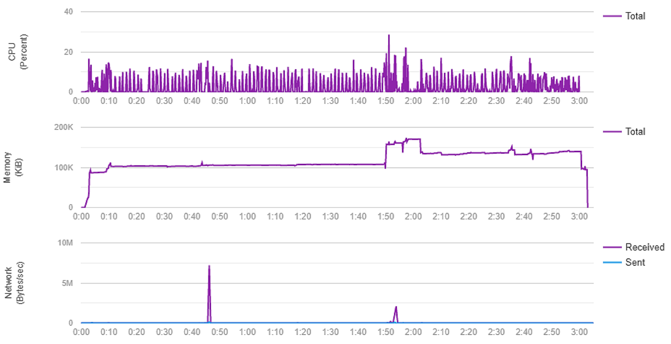


Figure 11. Commuter Activity CPU, Network and Memory statistics for 3 min 4 sec

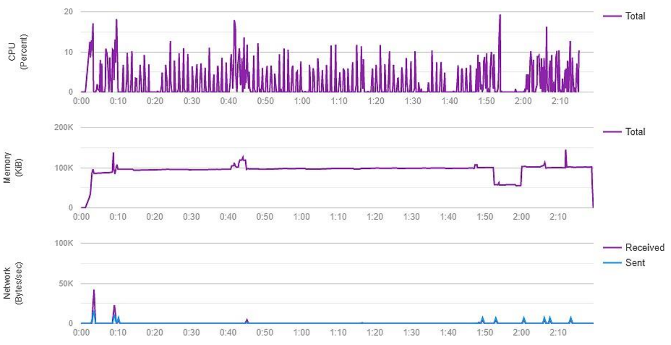


Figure 12. Driver Activity CPU, Network and Memory statistics for 2 min 19 sec

4.2 Web application evaluation

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 open-source, 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 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 Ref. [36].

The results of the performance assessment are shown in Table 2. 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 FCP, the amount of imagery and styling can be reduced, but all other metrics reported good results on the desktop.

Table 2. Page speed 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
Overall Score	65	95

4.3 Ridership forecasting evaluation

This section describes and evaluates the machine learning model integrated into the newly implemented public transport system. Since the adopted machine learning model required data to be trained on, we utilized 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.

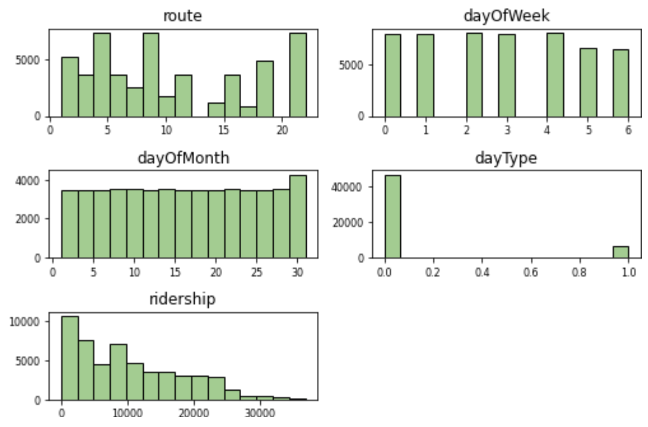


Figure 13. Dataset columns histograms

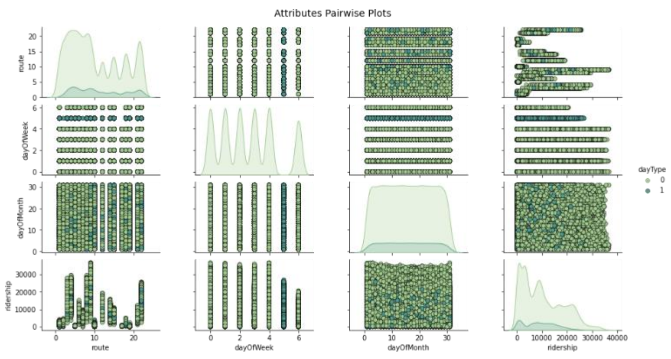


Figure 14. Pairwise correlogram for daytype

Dataset: The dataset used for this study 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 13 shows the value range and distribution for each attribute in the dataset. Specifically, Figures 14 and 15 visually describe the pairwise correlogram dataset for day Type and dataset correlation matrix.

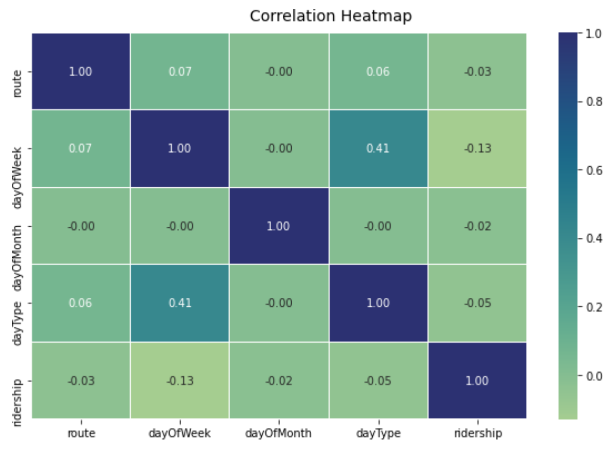


Figure 15. Dataset correlation matrix

Preprocessing: The varying scale of values in the dataset used led us to resort to preprocessing to resolve this. 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*: In this case, the data is rescaled from the original range so 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*: This concerns 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 $z = \frac{x-mean(x)}{standarddeviation(x)}$.

Model Architecture: The adopted machine learning model is 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 the developed system: route, day Type, day Of Week and day Of Month. 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 16.

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 with a different dataset to 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 the new system. The models were trained on Google Colab - an online Python Notebook - with the system specification shown in Table 3.

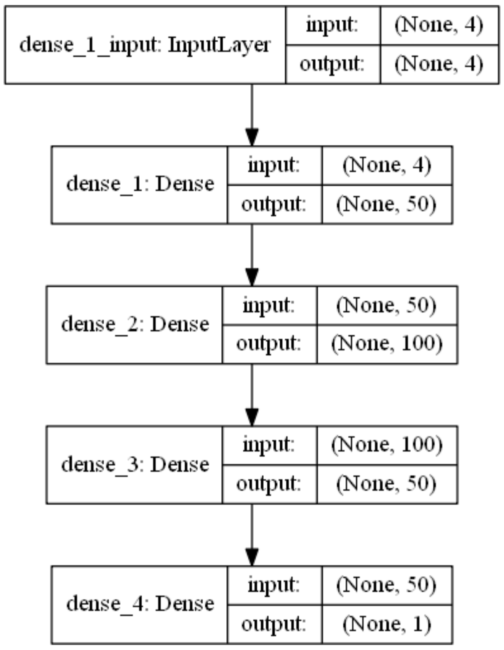


Figure 16. Model architecture

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
Memory	13GB
Hard Disk Space	37GB

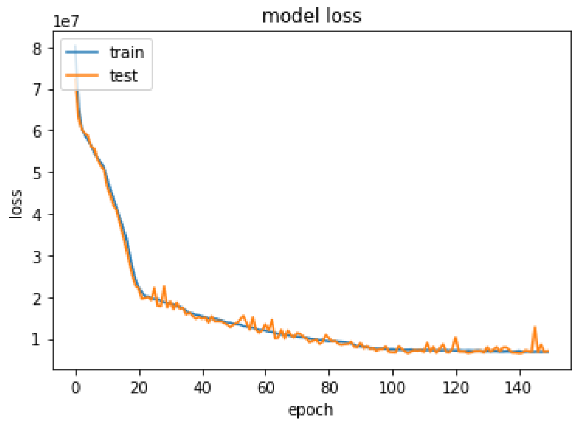


Figure 17. MSE for the model trained with an unscaled dataset

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. The batch size was set to 10 samples. To determine 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 17, 18 and 19 show each model's Mean Square Error fluctuations for a duration of 150 epochs, which indicates how well the model was learning.

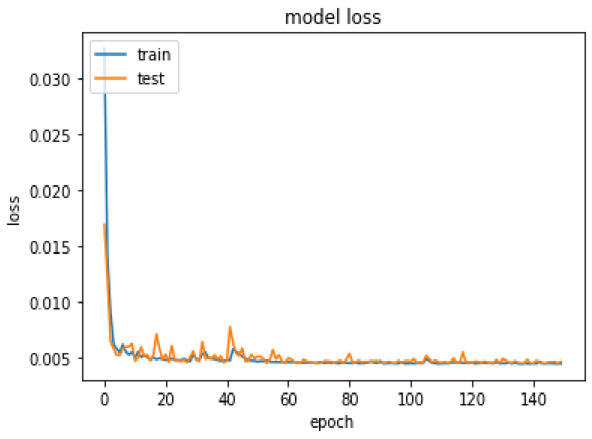


Figure 18. MSE for the model trained with a normalized dataset

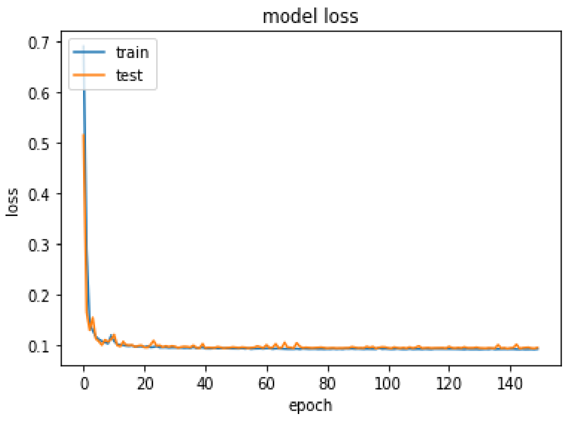


Figure 19. MSE for the model trained with a standardized dataset

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 current models since they were all tested with unseen data. The standardization of the dataset dramatically improved the model's learning ability compared to 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 the proposed 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, Mc-Call, 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: 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 outcome of interaction when a product is used in a particular context. The product quality model comprises 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 the volunteers completed three different assessments, which gave us fifteen assessments in total. The assessments contained sub-characteristics of the aforementioned ISO 25010 characteristics. The volunteers provided ratings 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.

The volunteers evaluated the new system and scored the system on each characteristic's feasibility in the South African context. For the obtained results shown in Table 4, the ratings in all characteristics had a very low standard deviation. They were all less than one, which implies that the mean ratings resemble individual volunteer ratings closely. Based on the results, the new Irenbus is functionally suitable for the needs of commuters, bus drivers and bus operators. The performance efficiency of the system was rated high except for resource utilization, where volunteers reported heavy battery drainage on their mobile devices when logged in as bus drivers. This was due to background location tracking processes on which the GPS receiver, a small chip antenna 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 Ref. [47] how they are at- tempting 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. The system's maintainability and portability characteristics achieved decent ratings; this implies that the new system can be deemed feasible and cost-efficient. Overall, the ratings show that the volunteers were fairly satisfied with the current system.

Table 4. Irenbus system evaluation

		Rating	
Characteristic	Sub-Characteristic	μ	σ
Functional Suitability	Functional completeness	4.73	0.46
	Functional correctness	4.6	0.63
	Functional appropriateness	4.67	0.62
Performance Efficiency	Time behavior	4.27	0.8
	Resource utilization	3.47	0.52
	Capacity	4.67	0.49
Compatibility	Co-existence	4.8	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	4.8	0.56
Reliability	Maturity	4.67	0.49
	Availability	4.8	0.41
	Fault tolerance	4.47	0.63
	Recoverability	4.6	0.51
Security	Confidentiality	4.87	0.35
	Integrity	4.73	0.46
	Non-repudiation	4.8	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	4.2	0.86

5. CONCLUSIONS

This paper aimed to demonstrate a practical and implementable intelligent public transport management system in the context of developing countries. Five objectives were listed as part of this research. The first involved 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 to develop an approach to collect daily ridership by using only a mobile application and no external hardware modules. Third, developing an approach to incorporate 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 the newly developed system.

As this research aimed to develop the new system so 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 the proposed system being purely cloud-based and run on just a mobile application and requiring no extra external hardware components. We chose smartphones as the platform to implement the new system because more than 91% of the South African population owned smartphones as of 2019, 3G coverage was almost 100%, and 4G/LTE coverage was nearly 93% in 2019 [6]. Furthermore, the cloud-based system means system updates and new features can be rolled out effortlessly without physically tinkering 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; the exploration and development of a daily ridership collection approach; and the use of a machine learning model to predict future ridership based on the data collected by the system itself. The developed system took full advantage of cloud computing services while requiring only a smartphone to work, thereby eliminating the need and costs of installing and maintaining separate GPS trackers on the vehicles.

This research was limited to demonstrating the current implemented system on only one mode of public transport - buses; this was done to simplify the scope of the system for now. However, it will be interesting to see how the new system can be implemented for other public transport modes and training and testing done on a large scale.

For future work, the functionality of the Irenbus can be improved by way of 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 to improve the ridership predictions.

ACKNOWLEDGMENT

This work is supported by the Huawei Technologies Africa (Pty) Ltd, Masters Bursary (Menzi Skhosana, student number: 216032734).

DATA AVAILABILITY

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] Stats, S.A. (2015). Measuring household expenditure on public transport. http://www.statssa.gov.za/?p=5943.

[2] Stockin, D. (2020). Does adding an extra driving lane make traffic worse? https://drivetribe.com/p/does-adding-an-extra-driving-lane-E6FPiVJnQSCPun1-pS-Q-A?iid= LEiNsk8oQqOPNAOQ1{_}hVqg.

[3] Hanna, N. (2010). e-Transformation: Enabling new Development Strategies, New York: Springer, pp. 281-309. ISBN: 9781441911858.

[4] Skhosana, M., Ezugwu, A. (2021). A Real-Time Machine Learning Based Public Transport Bus-Passenger Information System. https://doi.org/10.31224/osf.io/2ubv9

[5] United Nations Department of Economic and Social Affairs. (2020). 68% of the World Population Projected To Live in Urban Areas By 2050. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.

[6] ICASA. (2020). The state of the ICT sector in South Africa. Independent Communications Authority of South Africa, 83. https://www.icasa.org.za/uploads/files/State-of-ICT-Sector-Report-March-2018.pdf.

[7] Motta, R.A., Da Silva, P.C.M., Santos, M.P.D.S. (2013). Crisis of public transport by bus in developing countries: a case study from Brazil. International Journal of Sustainable Development and Planning, 8(3): 348-361. https://doi.org/10.2495/SDP-V8-N3-348-361

[8] Ngoc, A.M., Hung, K.V., Tuan, V.A. (2017). Towards the development of quality standards for public transport service in developing countries: Analysis of public transport users' behavior. Transportation Research Procedia, 25: 4560-4579. https://doi.org/10.1016/j.trpro.2017.05.354

[9] Pucher, J., Korattyswaropam, N., Mittal, N., Ittyerah, N. (2005). Urban transport crisis in India. Transport Policy, 12(3): 185-198. https://doi.org/10.1016/j.tranpol.2005.02.008

[10] Sungur, C., Babaoglu, I., Sungur, A. (2015). Smart bus station-passenger information system. In 2015 2nd International Conference on Information Science and Control Engineering, pp. 921-925. https://doi.org/10.1109/ICISCE.2015.209

[11] Manikandan, R., Niranjani, S. (2014). Implementation on real time public transportation information using GSM query response system. Contemporary Engineering Sciences, 5(7): 509-514.

[12] Shirisha, K., Sivaprasad, T. (2016). Acquire bus information using GSM technology. International Journal of Advancements in Technology.

[13] Kumbhar, M., Survase, M., Mastud, P., Salunke, A., Sirdeshpande, S. (2016). Real time web based bus tracking system. International Research Journal of

Engineering and Technology, 3(2): 632-635.

[14] Skhosana, M. (2020). Irenbus: A real-time public transport management system. In 2020 Conference on Information Communications Technology and Society (ICTAS), pp. 1-7. https://doi.org/10.1109/ICTAS47918.2020.234000

[15] Skhosana, M., Ezugwu, A.E., Rana, N., Shafi'i, M.A. (2020). An intelligent machine learning-based real-time public transport system. In International Conference on Computational Science and Its Applications, pp. 649-665. https://doi.org/10.1007/978-3-030-58817-5_47

[16] Ryu, S., Park, B.B., El-Tawab, S. (2020). WiFi sensing system for monitoring public transportation ridership: A case study. KSCE Journal of Civil Engineering, 24(10): 3092-3104. https://doi.org/10.1007/s12205-020-0316-7

[17] Monchambert, G., De Palma, A. (2014). Public transport reliability and commuter strategy. Journal of Urban Economics, 81: 14-29. https://doi.org/10.1016/j.jue.2014.02.001

[18] Bacciu, D., Carta, A., Gnesi, S., Semini, L. (2017). An experience in using machine learning for short-term predictions in smart transportation systems. Journal of Logical and Algebraic Methods in Programming, 87: 52-66. https://doi.org/10.1016/j.jlamp.2016.11.002

[19] Hsu, Y.W., Chen, Y.W., Perng, J.W. (2020). Estimation of the number of passengers in a bus using deep learning. Sensors, 20(8): 2178. https://doi.org/10.3390/s20082178

[20] van Oort, N., Brands, T., de Romph, E. (2015). Short-term prediction of ridership on public transport with smart card data. Transportation Research Record, 2535(1): 105-111. https://doi.org/10.3141/2535-12

[21] MyCiti. (2020). Using MyCiti on your phone. Available from: https://www.myciti.org.za/en/discover-myciti/using-myciti-on-your-phone/.

[22] McCann A. (2020). Cities with the best & worst public transportation. https://wallethub.com/edu/cities-with-the-best-worst-public-transportation/65028/{#}main-findings.

[23] Optibus. (2020). The optibus platform. https://www.optibus.com/product/platform/.

[24] Frankenfield, J. (2020). What is cloud computing? - Types of Cloud Computing Services & More. https://www.trianz.com/insights/revolution-that-is-cloud-computing.

[25] Mongo, D.B. (2020). NoSQL Explained. https://www.mongodb.com/nosql-explained.

[26] Firebase. (2020). Store and sync data in real time. https://firebase.google.com/products/realtime-database.

[27] Firebase. (2020). Firebase realtime database. https://firebase.google.com/docs/database.

[28] Google, LLC. (2020). Introduction to Activities. https://developer.android.com/guide/components/activities/intro-activities.html.

[29] Google, LLC. (2020). Developer guides — android developers. https://developer.android.com/reference/android/app/Activity.

[30] C¸ORAK, A. (2020). Splash screen is more important than you think how to use. https://uxplanet.org/splash-screen-is-more-important-than-you-think-855e78da3c2c.

[31] Google, LLC. (2020). Get started - Directions API. https://developers.google.com/maps/documentation/directions/start.

[32] Wilson, M. (2020). App download stats reveal reason behind low number of apps downloaded. https://www.zipwhip.com/blog/app-download-statistics-reveal-why-people-dont-download-apps/.

[33] Sato, D., Wider, A., Windheuser, C. (2019). Continuous delivery for machine learning. https://martinfowler.com/articles/cd4ml.html.

[34] Android Developers. (2020). Run apps on the android emulator. https://developer.android.com/studio/run/emulator.

[35] Firebase. (2020). Firebase test. https://firebase.google.com/docs/test-lab.

[36] Google Developers. (2020). About pagespeed insights. https://developers.google.com/speed/docs/insights/v5/about.

[37] Chicago Transit Authority. (2020). CTA - Ridership - bus routes - daily totals by route about this dataset columns in this dataset. https://data.cityofchicago.org/Transportation/CTA-Ridership-Bus-Routes-Daily-Totals-by-Route/jyb9-n7fm.

[38] Chicago Transit Authority. (2020). Annual Ridership Report Calendar Year 2015. https://www.transitchicago.com/assets/1/6/2018_Annual_Report_-_v3_04.03.2019.pdf.

[39] SYSQA. (2020). Iso 25010: 2011. http://www.gripoprequirements.nl/downloads iso-25010-2011-een-introductie-v1{_}0.pdf.

[40] Boehm, B.W., Brown, J.R., Lipow, M. (1976). Quantitative evaluation of software quality. In Proceedings of the 2nd International Conference on Software Engineering, pp. 592-605.

[41] Walters, G.F., McCall, J.A. (1979). Software quality metrics for life-cycle cost-reduction. IEEE Transactions on Reliability, 28(3): 212-220. https://doi.org/10.1109/TR.1979.5220569

[42] Saini, R., Dubey, S.K., Rana, A. (2011). Analytical study of maintainability models for quality evaluation. Indian Journal of Computer Science and Engineering, 2(3): 449-454.

[43] Dromey, R.G. (1995). A model for software product quality. IEEE Transactions on Software Engineering, 21(2): 146-162. https://doi.org/10.1109/32.345830

[44] Coallier, F. (2001). Software engineering–product quality–part 1: quality model. International Organization for Standardization: Geneva, Switzerland. 2(1): 1-25.

[45] Messer-Misak, K., de Bruin, J.S., Hanke, S. (2020). A systematic approach to quality requirement management in medical software. In dHealth 2020–Biomedical Informatics for Health and Care (pp. 129-136). IOS Press.

[46] Liao, S. (2018). Why GPS-dependent apps deplete your smartphone battery. https://www.theverge.com/2018/8/17/17630872/smartphone-battery-gps-location-services.

[47] Hartanto, Y., Attwell, B. (2020). Activity / Service as a dependency: Re-thinking android architecture for the uber driver app. https://eng.uber.com/activity-service-dependency-android-app-architecture/.

APPENDIX

Irenbus Mobile Application Dependency Graphs

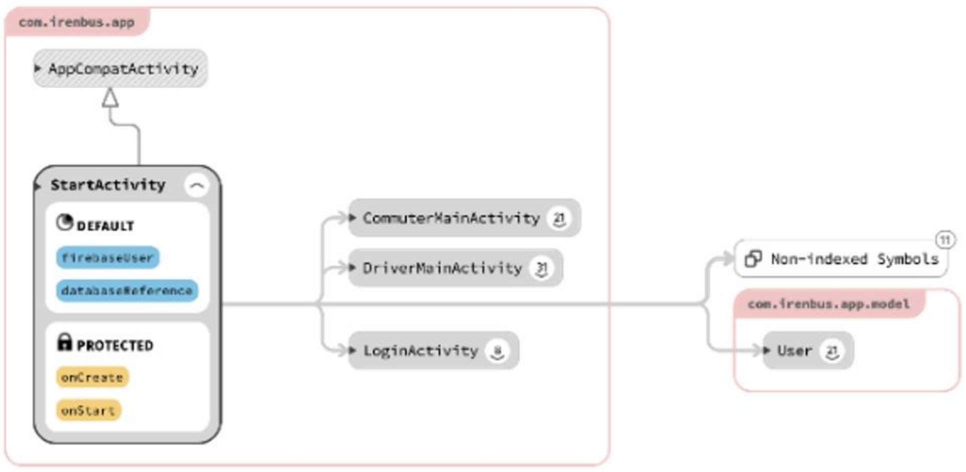


Figure 5.1. Start activity

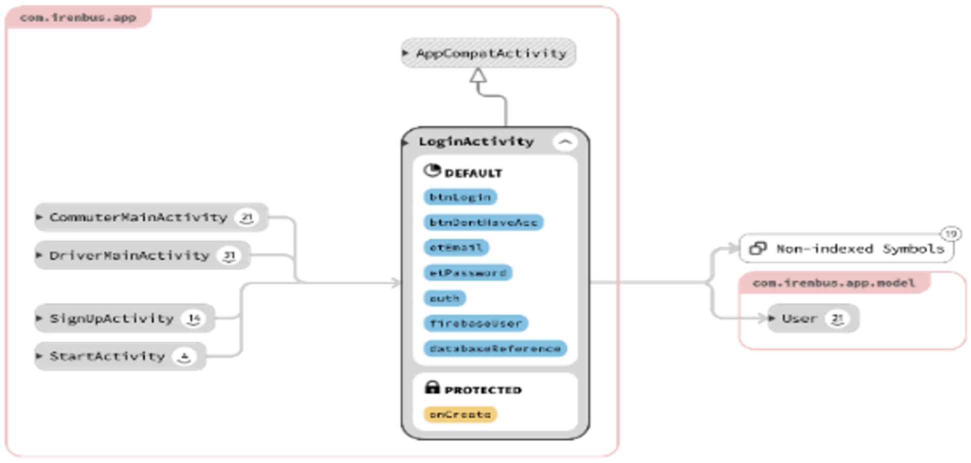


Figure 5.2. Login activity

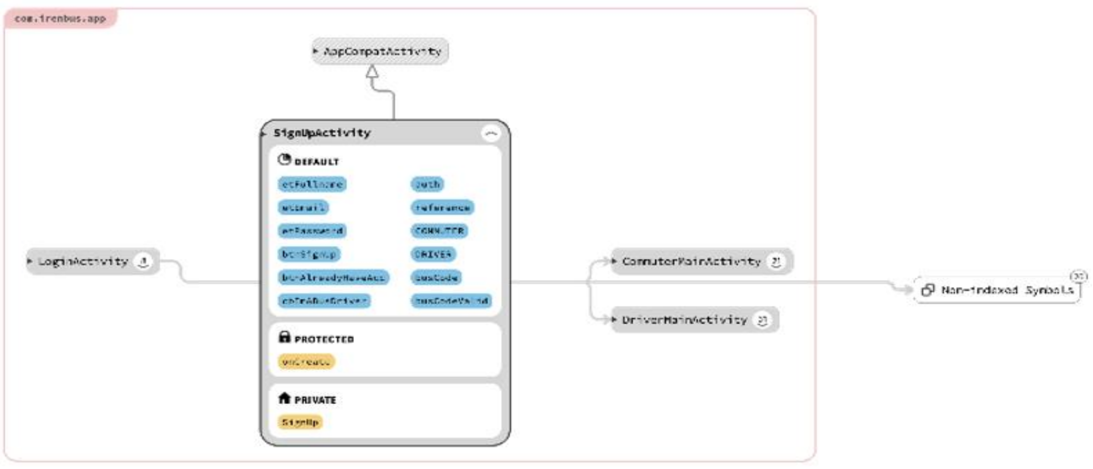


Figure 5.3. SignUp activity

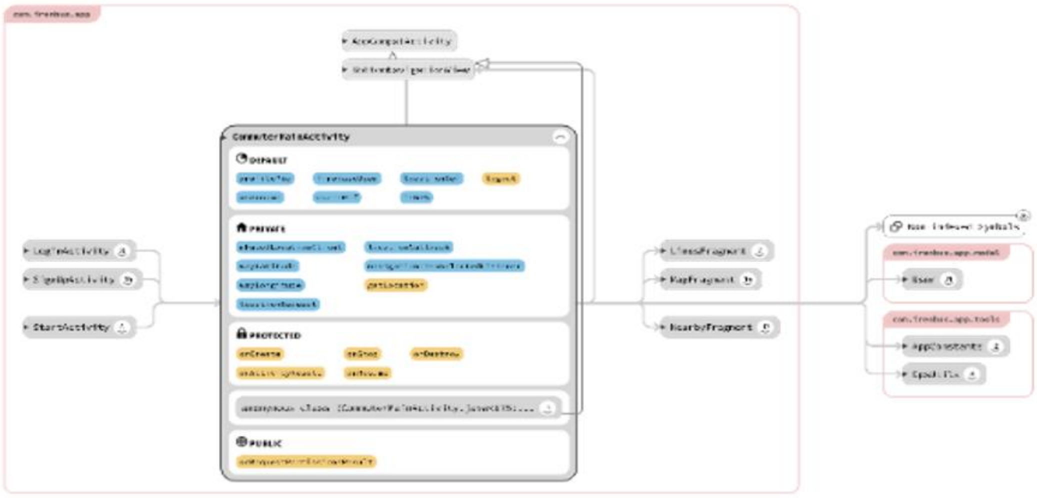


Figure 5.4. Commuter main activity

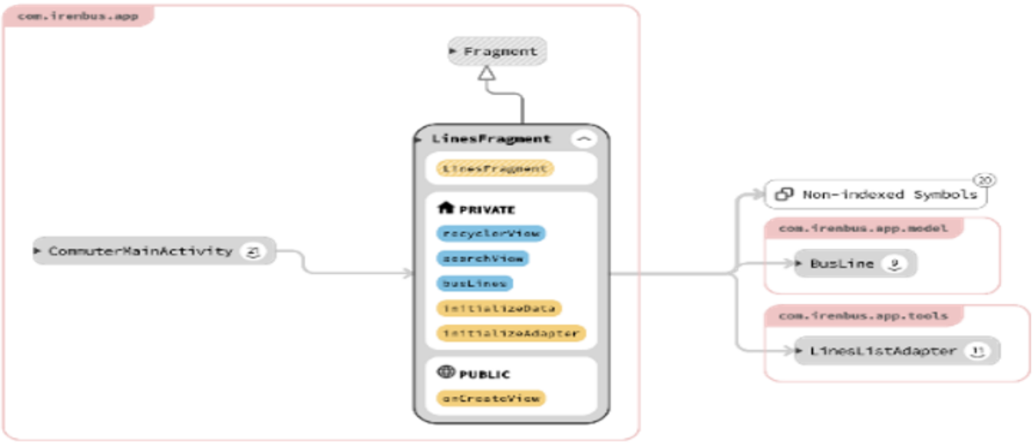


Figure 5.5. Lines fragment activity

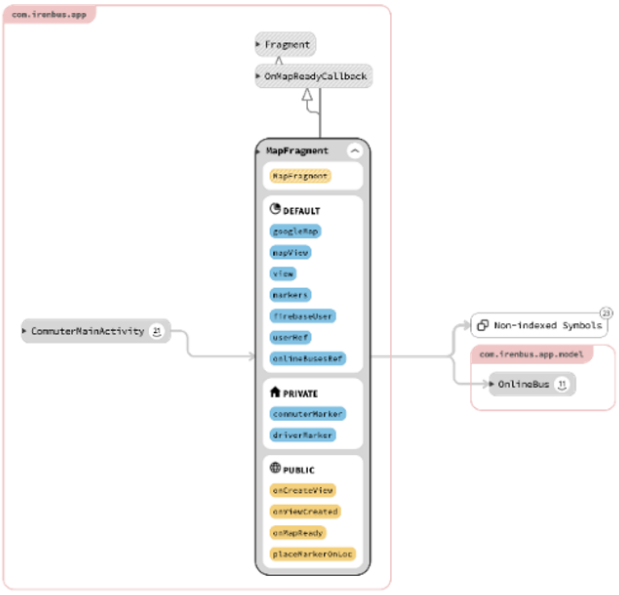


Figure 5.6. Map fragment activity

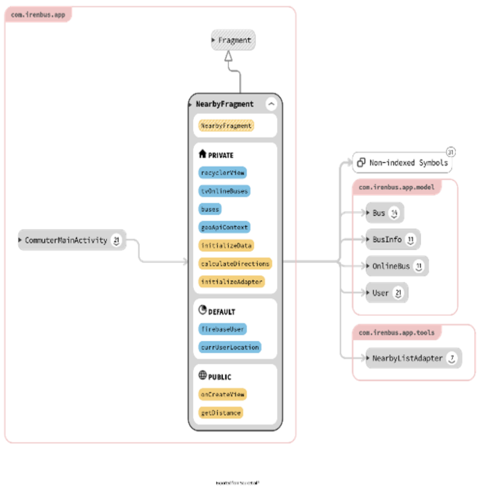


Figure 5.7. Nearby fragment activity

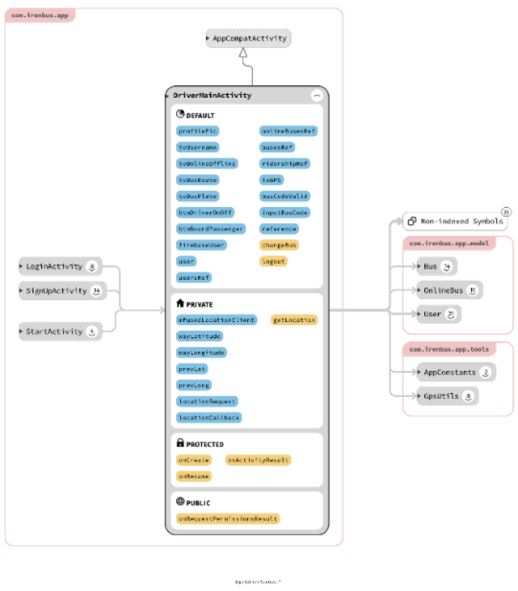


Figure 5.8. Driver main activity

IRENBUS DATA MODELS

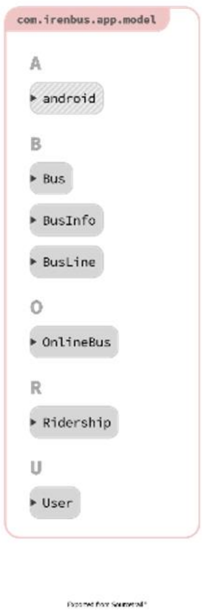


Figure 5.9. All data models

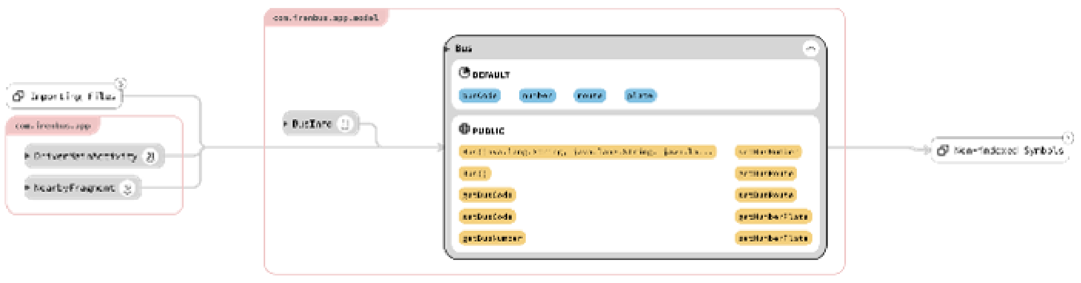


Figure 5.10. Bus data model

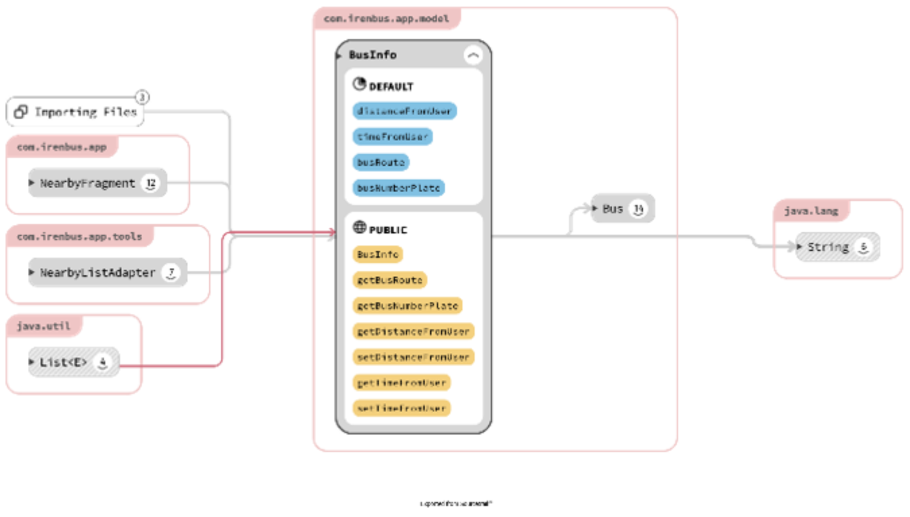


Figure 5.11. Bus info model

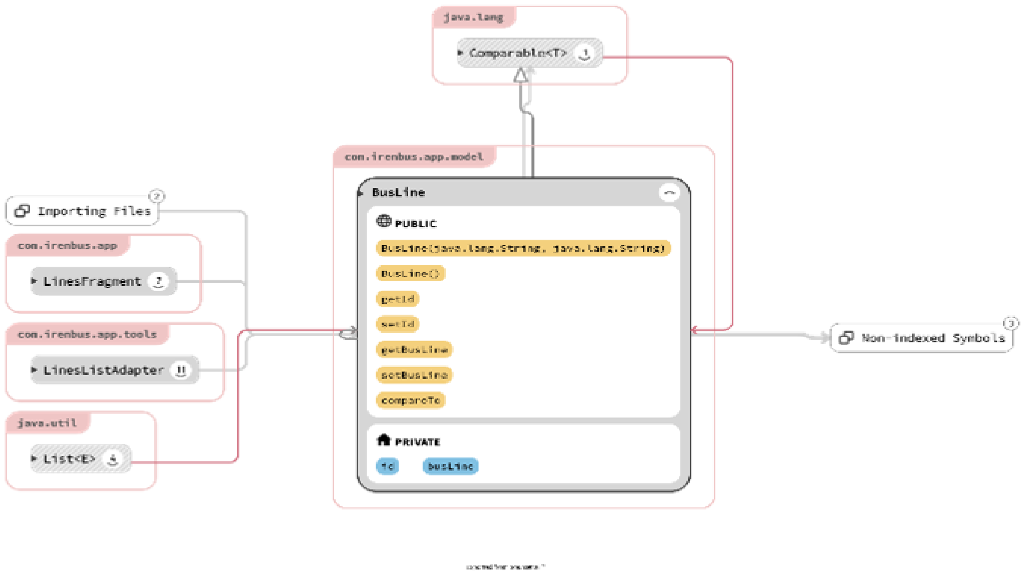


Figure 5.12. Bus line model

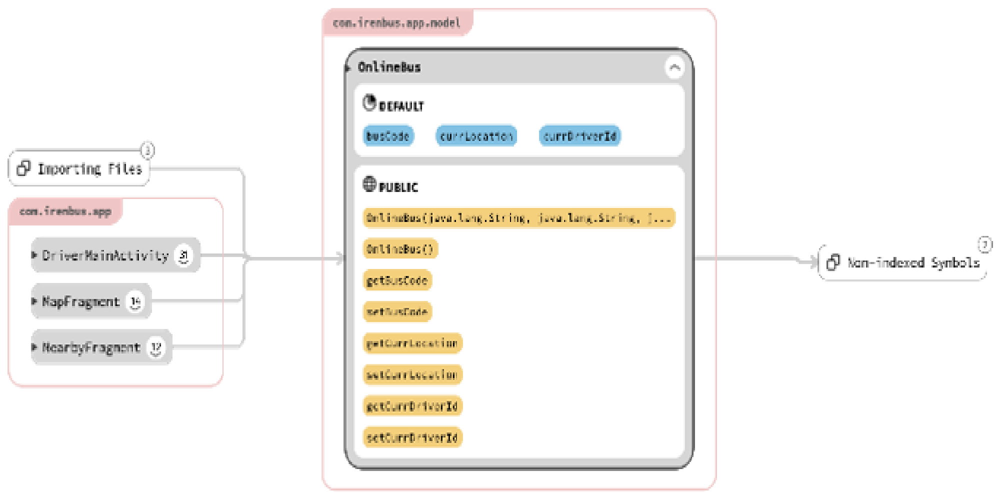


Figure 5.13. Online bus model

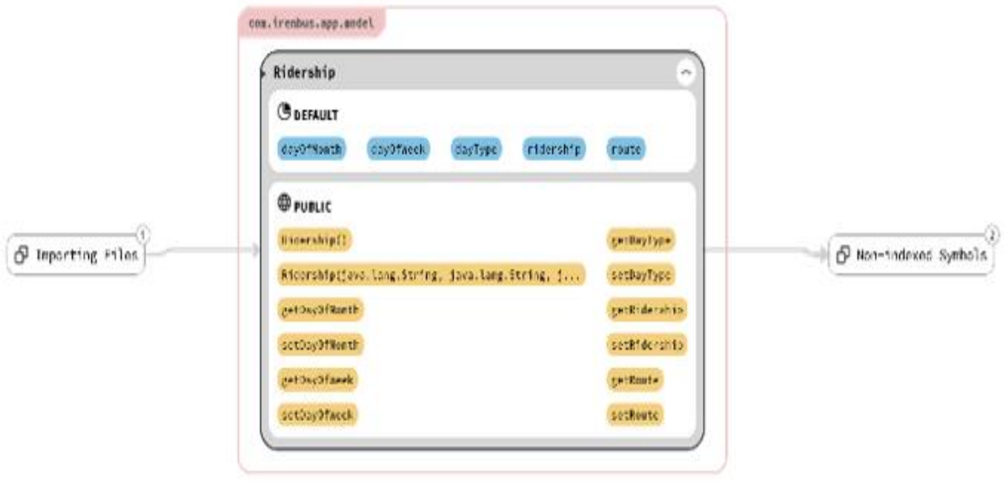


Figure 5.14. Ridership model

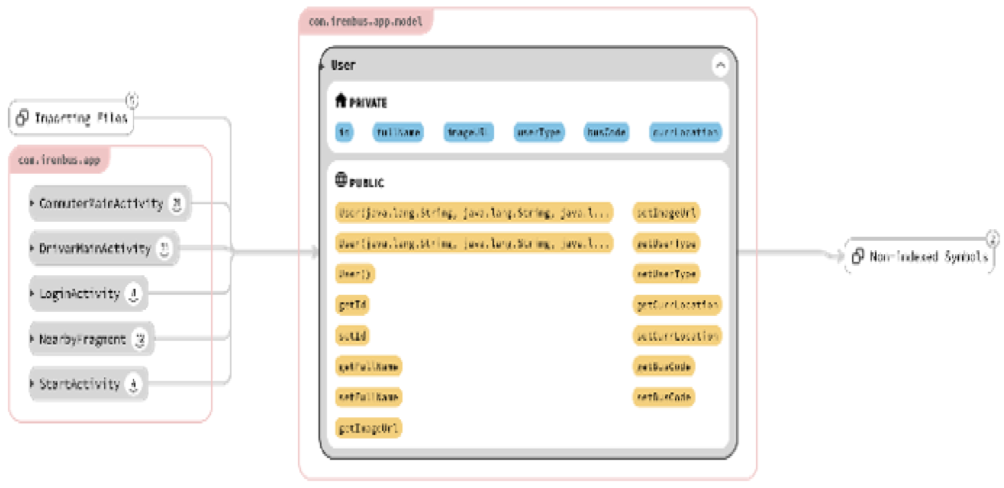


Figure 5.15. User model