TOLL TAX PAYMENT APPLICATION

Submitted

for

MCA PROJECT FINAL REPORT

in

COMPUTER SCIENCE AND ENGINEERING

by

TANISH SINGH

VISHAL KAUNDAL

RAJA KESHRI

Admission No.: 18SCSE2030077

18SCSE2030076

18SCSE2030040

Under the Supervision of:

Asst. Prof Abhay Kumar

Associate Professor, SCSE

School of Computing Science and Engineering

Galgotias University Greater Noida, Uttar Pradesh

MAY 2020

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
BONAFIDE CERTIFICATE

Certified that this project report "TOLL TAX PAYMENT APPLICATION" is the bonafide work Of "Tanish Singh (18032030092), Vishal Kaundal (18032030091), Raja keshri (18032030060)" Who carried out the project work under my supervision.

SIGNATURE OF HEAD
Dr. Munish Shabarwal
Dean Assistant Professor
School of computer Science
& Engineering
Galgotias University Uttar Pradesh

SIGNATURE OF SUPERVISOR
Mr. Abhay Kumar
Assistant Professor
School of computer Science &
Engineering
Galgotias University Uttar Pradesh

ACKNOWLEDGMENT

We wish to record my deep sense of gratitude and profound thanks to my research supervisor **Asst Prof. Abhay Kumar ,** Associate Professor, School of Computing Science and Engineering department, Galgotias University, Greater Noida for his keen interest, inspiring guidance, constant encouragement with my work during all stages, to bring this dissertation into fruition.

I extend my sincere thanks to Dean SCSE for providing excellent platform and resources to carry out my research projects. Also I would like to thank the panel members for their valuable suggestions and support during presentation of my research projects.

Finally, I extend my sincere thanks to the University Management, All faculty members, non-teaching staff members and Lab Assistants of the SCSE Department, Galgotias University, Greater Noida, for their valuable support throughout the course of my MCA Dissertation.

I thank my friends, fellow researchers and family members who have encouraged me in my research efforts and shouldered me in needy times.

TANISH SINGH

VISHAL KAUNDAL

RAJA KESHRI

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the MCA Dissertation Phase-III, entitled **TOLL TAX PAYMENT APPLICATION** in Computer Science & Engineering and submitted in the School of Computing Science Engineering of the Galgotias University, Greater Noida is an authentic record of my own work carried out during a period from June **2019** to MAY **2020** under the supervision of **Prof. Abhay Kumar,** School of Computing Science & Engineering, Galgotias University, Greater Noida. The Content presented in the dissertation has not been submitted by me for the award of any other degree of this or any other Institute.

(TANISH SINGH)

(VISHAL KAUNDAL)

(RAJA KESHRI)

M.C.A (CSE)

(18032030092)

(18032030091)

(18032030040)

CERTIFICATE

It is to certified that the work contained in the project report titled "TOLL TAX PAYMENT ANDROID APPLICATION" by the following students:

Name of the Student	Enroll Number
TANISH SINGH	18032030092
VISHAL KAUNDAL	18032030091
RAJA KESHRI	18032030060

Has been carried out under my/our supervision and this work has not been submitted or published elsewhere for any degree.

Place: Greater Noida
Dean-SCSE

ABSTRACT

The expressway transportation has become more and more important in today's road network and the manual toll collection system has become outdated due to its number of drawbacks. By employing automated toll collection system, driver of vehicles need not to stop at a window or and waste time for waiting in a long queue to pay their toll. This reduces the consumption of fuel; reduce congestion, increase road safety. The Toll tax Payment system is basically designed for an uninterrupted toll collection, which has become an important part of intelligent transportation system. This paper presents the concept of Toll tax Payment using track system. This work eliminates the need for motorists and toll authorities to manually perform ticket payments and toll fee collections, respectively. Data information are also easily exchanged between the motorists and toll authorities, thereby it is able to eliminate possible human errors for efficient toll collection.

TABLE OF CONTENT

Acknowledgement
Bonafied Certificate
Candidate’s Declaration
Certificate
Abstract
1.1 Introduction
1.4 Overview
2.1 Numerical Study and coding
3.1 Literature Review
4.1 System and Module
4.2 References

CHAPTER 1

1.1 INTRODUCTION

The working principle of our project is based on the track system that are to be used for the tracking the vehicle position and which type of vehicle that was to apply the charges on that vehicle. And also we are going to use the OBU unit which is placed in the vehicle. When the vehicle pass through charging zone then by using the OBU unit the balance was deducted from the account of registered user's account. It's very easy to setup a new toll area and also we are going to remove the old one. DSRC-based Electronic toll system can only use in the district range with in that zone, if charge region need to change, it's difficult to change the whole environment of the system. By altering the toll mode, toll rate and virtual toll node the track base ETC system can change toll area, carry out stretch and mutative toll mode. GPS Toll Collection uses contact less automatic vehicle identification technology for identification of vehicle Owner passing through that particular toll Collection centers. So time required to take the toll manually can be reduced. Saving time for toll collection will save almost maximum time and also saves fuel. The project aims at developing software to collect toll by providing the end user a prepaid wireless cell phone. Users can take this cell phone having GPS technology and according to their need can recharge their cell phones. The user will be provided with prepaid wireless Android cell phone. And toll will be deducted at the end of the session.

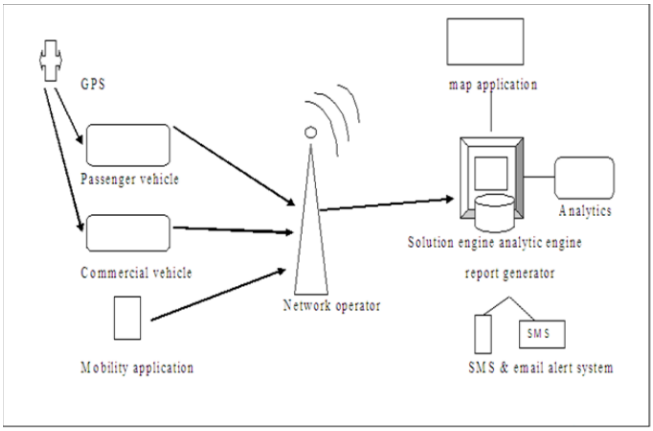


Fig 1.1

1.2 Advantages of Toll Tax Application

- Fewer or shorter queues of vehicles at toll plazas by increasing toll plaza service turnaround rates.
- Fast and more efficient service.
- Ability to make payments by keeping a balance on the card itself.
- Other general advantages include minimization of fuel wastage and reduced emissions by reducing deceleration rate, waiting time of vehicles in queue, and acceleration.

For the toll operators, the benefits include:

- Low toll collection costs.
- Better management by centralized user account

Thus, the Toll Tax Application is useful for both the motorists and toll operators, this is the reason of extended use of ETC system throughout the world.

1.3 Technologies and Terminologies Used

The tools used in this are – NetBeans for server side and Android Studio for front end.
Data base management by MySQL.

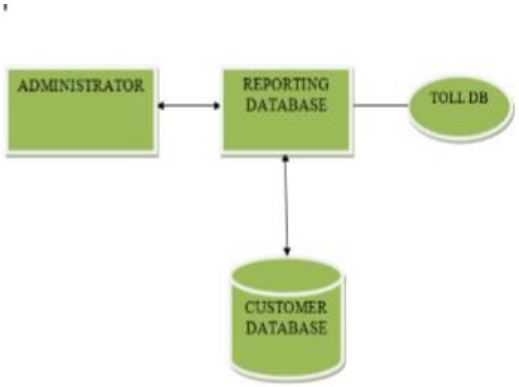


Fig 1.2

1. Administrator: Admin database contains all the details of central database and all toll plazas under Construction.

2. Central database: Centralized system is heart of database. Central database consist records of all toll plazas under that construction. This central database managed by administrator. The customer must register into this account to use ETC system. This account information about is stored into the RTO database. When the registered customer passes through the particular toll plazas then, automatically toll will be deducted from customer's account. This deduction will be updated by central database

3. Integrated database: Integrated database is connected to the central database. Integrated database consist of RTO database. This database will update automatically. RTO database includes all registered vehicles and the details of vehicle such as vehicle owner, vehicle number,

1.4 OVERVIEW

Electronic toll collection (ETC) aims to eliminate the delay on toll roads, HOV lanes, toll bridges, and toll tunnels by collecting tolls without cash and without requiring cars to stop. Electronic toll booths may operate alongside cash lanes so that drivers who do not have transponders can pay a cashier or throw coins into a receptacle. With cashless tolling,[1] cars without transponders are either excluded or pay by plate – a bill may be mailed to the address where the car's license plate number is registered, or drivers may have a certain amount of time to pay with a credit card by phone. Open road tolling is a popular form of cashless tolling without toll booths; cars pass electronic readers even at highway speeds without the safety hazard and traffic bottlenecks created by having to slow down to go through an automated toll booth lane.

Transponders are used to facilitate micropayments from drivers who have typically signed up in advance and loaded money into a declining-balance account which is debited each time they pass a toll point. License plate readers and sensors can be used to detect cars which are evading tolls or which are wanted by law enforcement for other reasons. Electronic tolling is cheaper than a staffed booth, reducing transaction costs for government agencies or private road owners recouping construction or maintenance costs or deriving revenue from a toll road. The ease of varying the amount of the toll and the ability to charge drivers without building a toll booth also makes it easy to implement road congestion pricing, including for high-occupancy lanes, toll lanes that bypass congestion, and city-wide congestion charges.

In 1959, Nobel Economics Prize William was the first to propose a system of electronic tolling for the Washington Metropolitan Area. He proposed that each car would be equipped with a transponder: "The transponder's personalized signal would be picked up when the car passed through an intersection, and then relayed to a central computer which would calculate the charge according to the intersection and the time of day and add it to the car’s bill." In the 1960s and the 1970s, free flow tolling was tested with fixed transponders at the undersides of the vehicles and readers, which were located under the surface of the highway.[3] Modern toll transponders are typically mounted under the windshield, with readers located in overhead gantries.

1.1 About android application –

An Android app is a software application running on the Android platform. Because the Android platform is built for mobile devices, a typical Android app is designed for a smartphone or a tablet PC running on the Android OS.
Although an Android app can be made available by developers through their websites, most Android apps are uploaded and published on the Android Market, an online store dedicated to these applications. The Android Market features both free and priced apps.

Android apps are written in the Java programming language and use Java core libraries. They are first compiled to Dalvik executables to run on the Dalvik virtual machine, which is a virtual machine specially designed for mobile devices.

Developers may download the Android software development kit (SDK) from the Android website. The SDK includes tools, sample code and relevant documents for creating Android apps.

Novice developers who simply want to play around with Android programming can make use of the App Inventor. Using this online application, a user can construct an Android app as if putting together pieces of a puzzle.
Although an Android app can be made available by developers through their websites, most Android apps are uploaded and published on the Android Market, an online store dedicated to these applications. The Android Market features both free and priced apps.

Android apps are written in the Java programming language and use Java core libraries. They are first compiled to Dalvik executables to run on the Dalvik virtual machine, which is a virtual machine specially designed for mobile devices.

Developers may download the Android software development kit (SDK) from the Android website. The SDK includes tools, sample code and relevant documents for creating Android apps.

Novice developers who simply want to play around with Android programming can make use of

the App Inventor. Using this online application, a user can construct an Android app as if putting together pieces of a puzzle.

1.1.1Application Fundamentals

Android apps can be written using Kotlin, Java, and C++ languages. The Android SDK tools compile your code along with any data and resource files into an APK, an Android package, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.

Each Android app lives in its own security sandbox, protected by the following Android security features:

- The Android operating system is a multi-user Linux system in which each app is a different user.
- By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app). The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
- Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.
- By default, every app runs in its own Linux process. The Android system starts the process when any of the app's components need to be executed, and then shuts down the process when it's no longer needed or when the system must recover memory for other apps.

  The Android system implements the principle of least privilege. That is, each app, by default, has access only to the components that it requires to do its work and no more. This creates a very secure environment in which an app cannot access parts of the system for which it is not given permission. However, there are ways for an app to share data with other apps and for an app to access system services:

- It's possible to arrange for two apps to share the same Linux user ID, in which case they are able to access each other's files. To conserve system resources, apps with the same user ID can also arrange to run in the same Linux process and share the same VM. The apps must also be signed with the same certificate.

- An app can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, and Bluetooth. The user has to explicitly grant these permissions. For more information, see Working with System Permissions.

  The rest of this document introduces the following concepts:

- The core framework components that define your app.
- The manifest file in which you declare the components and the required device features for your app.
- Resources that are separate from the app code and that allow your app to gracefully optimize its behavior for a variety of device configurations.

1.1.2 App components

App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

There are four different types of app components:

- Activities
- Services
- Broadcast receivers
- Content providers

Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed. The following sections describe the four types of app components.

Activities

An *activity* is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities if the email app allows it. For example, a camera app can start the activity in the email app that composes new mail to allow the user to share a picture. An activity facilitates the following key interactions between system and app:

- Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.
- Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.
- Helping the app handle having its process killed so the user can return to activities with their previous state restored.
- Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows. (The most classic example here being share.)

Services

A *service* is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it. There are actually two very distinct semantics services tell the system about how to manage an app: Started services tell the system to keep them running until their work is completed. This could be to sync some data in the background or play music even after the user leaves the app. Syncing data in

the background or playing music also represent two different types of started services that modify how the system handles them:

- Music playback is something the user is directly aware of, so the app tells the system this by saying it wants to be foreground with a notification to tell the user about it; in this case the system knows that it should try really hard to keep that service's process running, because the user will be unhappy if it goes away.
- A regular background service is not something the user is directly aware as running, so the system has more freedom in managing its process. It may allow it to be killed (and then restarting the service sometime later) if it needs RAM for things that are of more immediate concern to the user.

  Bound services run because some other app (or the system) has said that it wants to make use of the service. This is basically the service providing an API to another process. The system thus knows there is a dependency between these processes, so if process A is bound to a service in process B, it knows that it needs to keep process B (and its service) running for A. Further, if process A is something the user cares about, then it also knows to treat process B as something the user also cares about. Because of their flexibility (for better or worse), services have turned out to be a really useful building block for all kinds of higher-level system concepts. Live wallpapers, notification listeners, screen savers, input methods, accessibility services, and many other core system features are all built as services that applications implement and the system binds to when they should be running.

  A service is implemented as a subclass of Service. For more information about the Service class, see the Services developer guide.

1.2 Content providers

A *content provider* manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access. Through the content provider, other apps can query or modify the data if the content provider allows it. For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query the content provider, such as `ContactsContract.Data`, to read and write information about a particular person. It is tempting to think of a content provider as an abstraction on a database, because there is a lot of API and support built in to them for that common case. However, they have a different core purpose from a system-design perspective. To the system, a content provider is an entry point into an app for publishing named data items, identified by a URI scheme. Thus an app can decide how it wants to map the data it contains to a URI namespace, handing out those URIs to other entities which can in turn use them to access the data. There are a few particular things this allows the system to do in managing an app:

- Assigning a URI doesn't require that the app remain running, so URIs can persist after their owning apps have exited. The system only needs to make sure that an owning app is still running when it has to retrieve the app's data from the corresponding URI.
- These URIs also provide an important fine-grained security model. For example, an app can place the URI for an image it has on the clipboard, but leave its content provider locked up so that other apps cannot freely access it. When a second app attempts to access that URI on the clipboard, the system can allow that app to access the data via a temporary *URI permission grant* so that it is allowed to access the data only behind that URI, but nothing else in the second app.

  Content providers are also useful for reading and writing data that is private to your app and not shared. For example, the Note Pad sample app uses a content provider to save notes.

  A content provider is implemented as a subclass of `ContentProvider` and must implement a standard set of APIs that enable other apps to perform transactions. For more information, see the Content Providers developer guide.

A unique aspect of the Android system design is that any app can start another app's component. For example, if you want the user to capture a photo with the device camera, there's probably another app that does that and your app can use it instead of developing an activity to capture a photo yourself. You don't need to incorporate or even link to the code from the camera app. Instead, you can simply start the activity in the camera app that captures a photo. When complete, the photo is even returned to your app so you can use it. To the user, it seems as if the camera is actually a part of your app.

When the system starts a component, it starts the process for that app if it's not already running and instantiates the classes needed for the component. For example, if your app starts the activity in the camera app that captures a photo, that activity runs in the process that belongs to the camera app, not in your app's process. Therefore, unlike apps on most other systems, Android apps don't have a single entry point (there's no `main ()` function).

Because the system runs each app in a separate process with file permissions that restrict access to other apps, your app cannot directly activate a component from another app. However, the Android system can. To activate a component in another app, deliver a message to the system that specifies your *intent* to start a particular component. The system then activates the component for you.

1.2.1 Activating components

Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message called an *intent*. Intents bind individual components to each other at runtime. You can think of them as the messengers that request an action from other components, whether the component belongs to your app or another.

An intent is created with an `Intent` object, which defines a message to activate either a specific component (explicit intent) or a specific *type* of component (implicit intent).

For activities and services, an intent defines the action to perform (for example, to *view* or *send* something) and may specify the URI of the data to act on, among other things that the component being started might need to know. For example, an intent might convey a

request for an activity to show an image or to open a web page. In some cases, you can start an activity to receive a result, in which case the activity also returns the result in an Intent. For example, you can issue an intent to let the user pick a personal contact and have it returned to you. The return intent includes a URI pointing to the chosen contact.

Unlike activities, services, and broadcast receivers, content providers are not activated by intents. Rather, they are activated when targeted by a request from a Content Resolver. The content resolver handles all direct transactions with the content provider so that the component that's performing transactions with the provider doesn't need to and instead calls methods on the Content Resolver object. This leaves a layer of abstraction between the content provider and the component requesting information (for security).

There are separate methods for activating each type of component:

- You can start an activity or give it something new to do by passing an Intent to start Activity() or startActivityForResult() (when you want the activity to return a result).
- With Android 5.0 (API level 21) and later, you can use the JobScheduler class to schedule actions. For earlier Android versions, you can start a service (or give new instructions to an ongoing service) by passing an Intent to start Service (). You can bind to the service by passing an Intent to bindService ().
- You can initiate a broadcast by passing an Intent to methods such as sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().
- You can perform a query to a content provider by calling query () on a ContentResolver.

For more information about using intents, see the Intents and Intent Filters document. The following documents provide more information about activating specific components: Activities, Services Broadcast Receiver, and Content Providers.

Declaring component capabilities

As discussed above, in activating components, you can use an Intent to start activities, services, and broadcast receivers. You can use an Intent by explicitly naming the target component (using the component class name) in the intent. You can also use an implicit intent, which describes the type of action to perform and, optionally, the data upon which you'd like to perform the action.

The implicit intent allows the system to find a component on the device that can perform the action and start it. If there are multiple components that can perform the action described by the intent, the user selects which one to use.

The system identifies the components that can respond to an intent by comparing the intent received to the intent filters provided in the manifest file of other apps on the device.

When you declare an activity in your app's manifest, you can optionally include intent filters that declare the capabilities of the activity so it can respond to intents from other apps. You can declare an intent filter for your component by adding an <intent-filter> element as a child of the component's declaration element.

1.3 What is a Toll Road?

A toll road, also known as a turnpike or tollway, is a public or private road for which a fee (or toll) is assessed for passage. It is a form of road pricing typically implemented to help recoup the cost of road construction and maintenance.

Toll roads have existed in some form since antiquity, with tolls levied on passing travellers on foot, wagon or horseback; but their prominence increased with the rise of the automobile, and many modern tollways charge fees for motor vehicles exclusively. The amount of the toll usually varies by vehicle type, weight, or number of axles, with freight trucks often charged higher rates than cars.

Tolls are often collected at toll booths, toll houses, plazas, stations, bars, or gates. Some toll collection points are unmanned and the user deposits money in a machine which opens the gate once the correct toll has been paid. To cut costs and minimize time delay many tolls today are collected by some form of automatic or electronic toll collection equipment which communicates electronically with a toll payer's transponder. Some electronic toll roads also maintain a system of toll booths so people without transponders can still pay the toll, but many newer roads now use automatic number plate recognition to charge drivers who use the road without a transponder, and some older toll roads are being upgraded with such systems.

Criticisms of toll roads include the time taken to stop and pay the toll, and the cost of the toll booth operators—up to about one third of revenue in some cases. Automated toll paying systems

help minimize both of these. Others object to paying "twice" for the same road: in fuel taxes and with tolls.

In addition to toll roads, toll bridges and toll tunnels are also used by public authorities to generate funds to repay the cost of building the structures. Some tolls are set aside to pay for future maintenance or enhancement of infrastructure, or are applied as a general fund by local governments, not being earmarked for transport facilities. This is sometimes limited or prohibited by central government legislation. Also road congestion pricing schemes have been implemented in a limited number of urban areas as a transportation demand management tool to try to reduce traffic congestion and air pollution.

1.3.1 Why to pay at toll Taxes?

The government do not have enough time, labour and money to built roads that provide uninterrupted ride to you on a Highway.

They open tenders for Infrastructure companies and give them this contract on making the road in a limit of time.
The company starts developing the project and invests from themselves (Most of the times, the money comes from loans) and a part from NHAI.

They then setup Toll Tax on the planned place where the Vehicles have to pay the toll in return of using the roads.

Yes, the earning is transparent as each record in Computerized and saved. The costs include everything - investment cost interest to be paid to the bank+ Salaries for Labours + Cost of Legal dependencies (if any) + Profits+ AMOUNT TO BE PAID TO NHAI + other costs.

A few years back, toll collection at toll plaza was done manually by issuing tokens of various toll rates by single ticket, multiple tickets for different categories of vehicles. And to be frank, they were not transparent and there used to be malpractices, pilferage etc. Nowadays, toll plazas have

become high tech with collection of toll through computerized receipts which are more or less full proof with little possibility of malpractices.

Another aspect regarding toll collection is the risk of lower toll collection due to factors like over projection of toll estimate in the project implementation stage or other factors like loss of toll paying traffic which are diverted to newly developed roads. This has resulted in many PPP projects getting delayed or deferred.

But generally, the amount collected from vehicles after attaining break even, is used for maintenance of the highway such as,

1. Patching up small pot holes.
2. Watering plants in the divider (This is to reduce intensity of high beam lights of opposite lane vehicles)
3. Re-painting the faded lane boundaries
4. Maintaining Truck lay-by shelters

But sometimes toll-plaza staff doesn't tender exact change, where they make money by wider margin. It should be electronic transaction with a smart card which can be made available in fuel pumps, motels in the highways. I believe this brings even more transparency paying exact toll fees.

Some charging methods to do the payment:-

Time Based Charges and Access Fees: In a time-based charging regime, a road user has to pay for a given period of time in which they may use the associated infrastructure. For the practically identical access fees, the user pays for the access to a restricted zone for a period or several days.

Motorway and other Infrastructure Tolling: The term tolling is used for charging a well-defined special and comparatively costly infrastructure, like a bridge, a tunnel, a mountain pass, a motorway concession or the whole motorway network of a country. Classically a toll is due when a vehicle passes a tolling station, be it a manual barrier-controlled toll plaza or a free-flow multi-lane station.

Distance or Area Charging: In a distance or area charging system concept, vehicles are charged per total distance driven in a defined area.

Why there is need of an electronic toll collection?

Electronic toll collection (ETC) aims to eliminate the delay on toll roads, HOV lanes, toll bridges, and toll tunnels by collecting tolls without cash and without requiring cars to stop. Electronic toll booths may operate alongside cash lanes so that drivers who do not have transponders can pay a cashier or throw coins into a receptacle. With cashless tolling, cars without transponders are either excluded or pay by plate – a bill may be mailed to the address where the car's license plate number is registered, or drivers may have a certain amount of time to pay with a credit card by phone. Open road tolling is a popular form of cashless tolling without toll booths; cars pass electronic readers even at highway speeds without the safety hazard and traffic bottlenecks created by having to slow down to go through an automated toll booth lane.

Transponders are used to facilitate micropayments from drivers who have typically signed up in advance and loaded money into a declining-balance account which is debited each time they pass a toll point. License plate readers and sensors can be used to detect cars which are evading tolls or which are wanted by law enforcement for other reasons. Electronic tolling is cheaper than a staffed booth, reducing transaction costs for government agencies or private road owners recouping construction or maintenance costs or deriving revenue from a toll road. The ease of varying the amount of the toll and the ability to charge drivers without building a toll booth also makes it easy to implement road congestion pricing, including for high-occupancy lanes, toll lanes that bypass congestion, and city-wide congestion charges.

In 1959, Nobel Economics Prize William Vickrey was the first to propose a system of electronic tolling for the Washington Metropolitan Area. He proposed that each car would be equipped with a transponder: "The transponder's personalized signal would be picked up when the car passed through an intersection, and then relayed to a central computer which would calculate the charge according to the intersection and the time of day and add it to the car’s bill." In the 1960s and the 1970s, free flow tolling was tested with fixed transponders at the undersides of the vehicles and readers, which were located under the surface of the highway.[3] Modern toll transponders are typically mounted under the windshield, with readers located in overhead gantries.

CHAPTER-2

NUMERICAL STUDY AND CODING

Code for various modules used in the project:

2.1 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.tolltax.external"
android:versionCode="1"
android:versionName="1.0">
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<!--
The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
Google Maps Android API v2, but you must specify either coarse or fine
location permissions for the 'MyLocation' functionality.
-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<activity
android:name=".LoginActivity"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
android:name=".admin.RegisterUser"
android:label="@string/title_activity_register_user"></activity>
<activity
android:name=".admin.BlackListVehicle"
android:label="@string/title_activity_black_list_vehicle"></activity>
<activity
android:name=".admin.RegisterVehicle"
```

```
android:label="@string/title_activity_register_toll_tax"></activity>
<activity
android:name=".admin.RechargeTollBalance"
android:label="@string/title_activity_recharge_toll_balance"></activity>
<activity
android:name=".common.PaymentOption"
android:label="PaymentOption"></activity>
<activity
android:name=".admin.ViewTolls"
android:label="@string/title_activity_view_tolls"></activity>
<activity
android:name=".ForgetPassword"
android:label="Forget Password"></activity>
<activity
android:name=".common.ChangePassword"
android:label="@string/title_activity_change_password"></activity>
<activity
android:name=".common.UpdateMobile"
android:label="@string/title_activity_update_mobile"></activity>
<activity
android:name=".toll.RechargeAccount"
android:label="@string/title_activity_check_vehicle"></activity>
<activity
android:name=".user.MyProfile"
android:label="@string/title_activity_my_profile"></activity>
<activity
android:name=".user.MyVehcile"
android:label="@string/title_activity_my_vehcile"></activity>
<activity
android:name=".user.MyTransactions"
android:label="@string/title_activity_my_transactions"></activity>
<activity
android:name=".user.PayTollTax"
android:label="@string/title_activity_schedule_path"></activity>
<activity
android:name=".user.ViewRoute"
android:label="@string/title_activity_view_route"></activity>
<activity
android:name=".admin.AdminPanel"
android:label="@string/title_activity_admin_panel"></activity>
<activity
android:name=".toll.TollPanel"
android:label="@string/title_activity_toll_panel"></activity>
<activity
android:name=".user.UserPanel"
android:label="@string/title_activity_user_panel"></activity>
<activity
```

```
android:name=".ViewDetails"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".admin.UsersList"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".admin.ViewBlackListVehicles"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".admin.ViewVehicleDetails"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".Profile"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".toll.Scanner"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".admin.ViewAllVehicles"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".user.ViewMyVehicles"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".user.VehicleDetails"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".user.ViewMyTransactions"
android:label="@string/title_activity_user_panel"></activity>
<activity
android:name=".user.TransDetails"
android:label="@string/title_activity_user_panel"></activity>
<!--
The API key for Google Maps-based APIs is defined as a string resource.
(See the file "res/values/google_maps_api.xml").
Note that the API key is linked to the encryption key used to sign the APK.
You need a different API key for each encryption key, including the release key that is used to
sign the APK for publishing.
You can define the keys for the debug and release targets in src/debug/ and src/release/.
-->
<meta-data
android:name="com.google.android.geo.API_KEY"
android:value="@string/google_maps_key" />

<activity
android:name=".user.MapsActivity"
android:label="@string/title_activity_maps"></activity>
```

</application>

</manifest>

2.2 JSONParser.java

```
package com.example.tolltax.external;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.entity.mime.MultipartEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.HttpParams;
import org.json.JSONObject;
@SuppressWarnings("deprecation")
public class JSONParser {
static InputStream is = null;
static JSONObject jObj = null;
static String json = "";
static BufferedReader in=null;
// constructor
public JSONParser() {
}
// function get json from url
// by making HTTP POST or GET mehtod
public String makeHttpRequest(String url, String method,
List<NameValuePair> params) {

// Making HTTP request
try {

// check for request method
if(method == "POST"){
// request method is POST
// defaultHttpClient
```

```
//DefaultHttpClient httpClient = new DefaultHttpClient();
//HttpPost httpPost = new HttpPost(url);

//httpPost.setEntity(new UrlEncodedFormEntity(params));
//ResponseHandler responseHandler = new BasicResponseHandler();
//httpClient.execute(httpPost, responseHandler);
// HttpResponse httpResponse = httpClient.execute(httpPost);


HttpClient client =new DefaultHttpClient();
URI website = null;
try {
website = new URI(url);
} catch (URISyntaxException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

HttpGet request=new HttpGet();
request.setURI(website);
request.setParams((HttpParams) params);
HttpResponse response=client.execute(request);
in=new BufferedReader(new InputStreamReader(response.getEntity().getContent()));


}else if(method == "GET"){
// request method is GET
DefaultHttpClient httpClient = new DefaultHttpClient();
String paramString = URLEncodedUtils.format(params, "utf-8");
url += "?" + paramString;
HttpGet httpGet = new HttpGet(url);
System.out.println("url"+url);
HttpResponse httpResponse = httpClient.execute(httpGet);
//httpResponse.ge
int status=httpResponse.getStatusLine().getStatusCode();
if(status==200){

is = httpResponse.getEntity().getContent();
System.out.println(is);
}
// RequestLine s=httpGet.getRequestLine();
// System.out.println(s);
//is = httpEntity.getContent();
}
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
} catch (ClientProtocolException e) {
```

```
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
return convertStreamToString(is);

}
public InputStream makeHttpRequestForImage(String url, String method,
List<NameValuePair> params) {


// Making HTTP request
try {

// check for request method
if(method == "POST"){
// request method is POST
// defaultHttpClient

//DefaultHttpClient httpClient = new DefaultHttpClient();
//HttpPost httpPost = new HttpPost(url);

//httpPost.setEntity(new UrlEncodedFormEntity(params));
//ResponseHandler responseHandler = new BasicResponseHandler();
//httpClient.execute(httpPost, responseHandler);
// HttpResponse httpResponse = httpClient.execute(httpPost);


HttpClient client =new DefaultHttpClient();
URI website = null;
try {
website = new URI(url);
} catch (URISyntaxException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

HttpGet request=new HttpGet();
request.setURI(website);
request.setParams((HttpParams) params);
HttpResponse response=client.execute(request);
in=new BufferedReader(new InputStreamReader(response.getEntity().getContent()));


}else if(method == "GET"){
// request method is GET
DefaultHttpClient httpClient = new DefaultHttpClient();
```

```
String paramString = URLEncodedUtils.format(params, "utf-8");
url += "?" + paramString;
HttpGet httpGet = new HttpGet(url);

HttpResponse httpResponse = httpClient.execute(httpGet);

//httpResponse.ge
int status=httpResponse.getStatusLine().getStatusCode();
if(status==200){

is = httpResponse.getEntity().getContent();
System.out.println(is);
}
// RequestLine s=httpGet.getRequestLine();
// System.out.println(s);
//is = httpEntity.getContent();
}

} catch (UnsupportedEncodingException e) {
e.printStackTrace();
} catch (ClientProtocolException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
return is;
}
public String makeHttpRequestMultiType(String url, String method,
MultipartEntity entity) {
```

2.3 LoginActivity.java

```
package com.example.tolltax.external;
import java.io.BufferedReader;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import java.util.TreeMap;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;

import com.example.tolltax.external.R;
import com.example.tolltax.external.admin.AdminPanel;
import com.example.tolltax.external.toll.Scanner;
import com.example.tolltax.external.toll.TollPanel;
```

```
import com.example.tolltax.external.user.UserPanel;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.widget.EditText;
import android.widget.TextView;

/**
* Activity which displays a login screen to the user, offering registration as
* well.
*/
public class LoginActivity extends Activity {
/**
* The default email to populate the email field with.
*/
public static final String EXTRA_EMAIL =
"com.example.android.authenticatordemo.extra.EMAIL";
private static String url_validate_login =URL.url_validate_login;

/**
* Keep track of the login task to ensure we can cancel it if requested.
*/
private UserLoginTask mAuthTask = null;

// Values for email and password at the time of the login attempt.
private String mEmail;
private String mPassword;

// UI references.
private EditText mEmailView;
private EditText mPasswordView;
private View mLoginFormView;
private View mLoginStatusView;
private TextView mLoginStatusMessageView;
```

```
String fileName = "rememberme.txt";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

setContentView(R.layout.activity_login);

// Set up the login form.
mEmail = getIntent().getStringExtra(EXTRA_EMAIL);
mEmailView = (EditText) findViewById(R.id.email);
mEmailView.setText(mEmail);

mPasswordView = (EditText) findViewById(R.id.password);
mPasswordView
.setOnEditorActionListener(new TextView.OnEditorActionListener() {
@Override
public boolean onEditorAction(TextView textView, int id,
KeyEvent keyEvent) {
if (id == R.id.login || id == EditorInfo.IME_NULL) {
attemptLogin();
return true;
}
return false;
}
});

mLoginFormView = findViewById(R.id.login_form);
mLoginStatusView = findViewById(R.id.login_status);
mLoginStatusMessageView = (TextView) findViewById(R.id.login_status_message);

findViewById(R.id.sign_in_button).setOnClickListener(
new View.OnClickListener() {
@Override
public void onClick(View view) {
attemptLogin();

}
});
}


@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
getMenuInflater().inflate(R.menu.login, menu);
return true;
}
```

```
@Override
public boolean onMenuItemSelected(int featureId, MenuItem item) {
if(item.getItemId()==R.id.action_forgot_password){
Intent i = new Intent(getBaseContext(), ForgetPassword.class);

startActivity(i);
}
return super.onMenuItemSelected(featureId, item);

}

/**
* Attempts to sign in or register the account specified by the login form.
* If there are form errors (invalid email, missing fields, etc.), the
* errors are presented and no actual login attempt is made.
*/
public void attemptLogin() {
if (mAuthTask != null) {
return;
}

// Reset errors.
mEmailView.setError(null);
mPasswordView.setError(null);

// Store values at the time of the login attempt.
mEmail = mEmailView.getText().toString();
mPassword = mPasswordView.getText().toString();

boolean cancel = false;
View focusView = null;

// Check for a valid password.
if (TextUtils.isEmpty(mPassword)) {
mPasswordView.setError(getString(R.string.error_field_required));
focusView = mPasswordView;
cancel = true;
} else if (mPassword.length() < 4) {
mPasswordView.setError(getString(R.string.error_invalid_password));
focusView = mPasswordView;
cancel = true;
}

// Check for a valid email address.
if (TextUtils.isEmpty(mEmail)) {
mEmailView.setError(getString(R.string.error_field_required));
```

```
focusView = mEmailView;
cancel = true;
} else if (!mEmail.contains("@")) {
mEmailView.setError(getString(R.string.error_invalid_email));
focusView = mEmailView;
cancel = true;
}

if (cancel) {
// There was an error; don't attempt login and focus the first
// form field with an error.
focusView.requestFocus();
} else {
// Show a progress spinner, and kick off a background task to
// perform the user login attempt.
mLoginStatusMessageView.setText(R.string.login_progress_signing_in);
showProgress(true);
mAuthTask = new UserLoginTask();
mAuthTask.execute((Void) null);


}
}

/**
* Shows the progress UI and hides the login form.
*/
@TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
private void showProgress(final boolean show) {
// On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow
// for very easy animations. If available, use these APIs to fade-in
// the progress spinner.
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
int shortAnimTime = getResources().getInteger(
android.R.integer.config_shortAnimTime);

mLoginStatusView.setVisibility(View.VISIBLE);
mLoginStatusView.animate().setDuration(shortAnimTime)
.alpha(show ? 1 : 0)
.setListener(new AnimatorListenerAdapter() {
@Override
public void onAnimationEnd(Animator animation) {
mLoginStatusView.setVisibility(show ? View.VISIBLE
: View.GONE);
}
});
```

```
mLoginFormView.setVisibility(View.VISIBLE);
mLoginFormView.animate().setDuration(shortAnimTime)
.alpha(show ? 0 : 1)
.setListener(new AnimatorListenerAdapter() {
@Override
public void onAnimationEnd(Animator animation) {
mLoginFormView.setVisibility(show ? View.GONE
: View.VISIBLE);
}
});
} else {
// The ViewPropertyAnimator APIs are not available, so simply show
// and hide the relevant UI components.
mLoginStatusView.setVisibility(show ? View.VISIBLE : View.GONE);
mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
}
}

/**
* Represents an asynchronous login/registration task used to authenticate
* the user.
*/
String username="";
String usertype="";
public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

private String password;

private String fname;

private String userid;

@Override
protected Boolean doInBackground(Void... params) {
// TODO: attempt authentication against a network service.
Boolean check = false;
try {
// Simulate network access.
Thread.sleep(2000);

} catch (InterruptedException e) {
return check;
}

username=mEmail;
password=mPassword;
```

```
String para = validateLogin(username, password);
if (para.contains("success")) {
check=true;
try {
setProperties(fileName, para);
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
fname = getMapElement(fileName, "firstName");
userid = getMapElement(fileName, "userid");
usertype = getMapElement(fileName, "userType");
System.out.println(para);
}




return check;
}

@Override
protected void onPostExecute(final Boolean success) {
mAuthTask = null;
showProgress(false);

if (success) {
if(usertype.equalsIgnoreCase("admin")){
Intent i = new Intent(getBaseContext(), AdminPanel.class);
i.putExtra("userid", userid);
i.putExtra("password", password);
i.putExtra("firstname", fname);
startActivity(i);
}else if(usertype.equalsIgnoreCase("Toll Tax")){
Intent i = new Intent(getBaseContext(), TollPanel.class);
i.putExtra("userid", userid);
i.putExtra("firstname", fname);
i.putExtra("password", password);
System.out.println("here");
startActivity(i);
}else if(usertype.equalsIgnoreCase("End User")){
Intent i = new Intent(getBaseContext(), UserPanel.class);
i.putExtra("userid", userid);
i.putExtra("firstname", fname);
i.putExtra("password", password);
System.out.println("here");
```

```
startActivity(i);
}


} else{
mPasswordView
.setError(getString(R.string.error_incorrect_password));
mPasswordView.requestFocus();
}


}
@Override
protected void onCancelled() {
mAuthTask = null;
showProgress(false);
}
}
@Override
public void onBackPressed() {
// TODO Auto-generated method stub

finish();
}

public String validateLogin(String username, String password) {
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("txtusername", username));
params.add(new BasicNameValuePair("txtpassword", password));


String st = "";
try {
JSONParser jp = new JSONParser();
st = jp.validateUser(url_validate_login, "POST", params);
System.out.println(st);
} catch (Exception e) {
e.printStackTrace();
}
return st;
}

public String getMapElement(String fileName, String key) {//to get a particular elemnt from the
file whtih the given key
TreeMap<String, String> map = null;
try {
map = getProperties(fileName);
```

```
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

return map.get(key);

}

public TreeMap<String, String> getProperties(String infile)//returns returns treemap in valu pair
throws IOException {
final int lhs = 0;
final int rhs = 1;

TreeMap<String, String> map = new TreeMap<String, String>();
// BufferedReader bfr = new BufferedReader(new FileReader(new
// File(infile)));
BufferedReader bfr = new BufferedReader(new InputStreamReader(
openFileInput(infile)));
String line;
while ((line = bfr.readLine()) != null) {
System.out.println(line);
if (!line.startsWith("#") && !(line.equalsIgnoreCase(""))) {
String[] pair = line.trim().split("=");
map.put(pair[lhs].trim(), pair[rhs].trim());
System.out.println(map);
}
System.out.println("inside" + map);
}

bfr.close();
System.out.println("returned" + map);
return (map);
}

public void setProperties(String infile, String paramName, String paramValue)
throws IOException {
String content = "";
content = paramName + "=" + paramValue;
FileOutputStream fos;
fos = openFileOutput(infile, Context.MODE_PRIVATE);
fos.write(content.getBytes());
System.out.println("donnnn");
fos.close();// to save first no.
}
```

```
public void setProperties(String infile, String param) throws IOException {//to write inside the
file
String content = param;

FileOutputStream fos;
fos = openFileOutput(infile, Context.MODE_PRIVATE);

fos.write(content.getBytes());
System.out.println("donnnn");
fos.close();// to save first no.
}
public void image(View v){
Intent intent = new Intent(getApplicationContext(), Scanner.class);
intent.putExtra("", "");
startActivity(intent);
}
}
```

2.4 URL.java

```
package com.example.tolltax.external;

public class URL {
//private static String url="http://10.0.2.2:8080/TollTaxApp/";
private static String url="http://172.20.10.4:8080/TollTaxApp/";
//private static String url="http://expansetracker.com/";
public static String url_get_users=url+"get_user_names.jsp";
public static String sendReport = url+"sendReport.jsp";
public static String upload = url+"uploadFile.jsp";
public static String url_get_reports = url+"Android/SMSReports.jsp";
public static String url_validate_login=url+"validateUser";
public static String url_save_location=url+"saveLocation.jsp";
public static String saveUser=url+"saveUser.jsp";
public static String saveAcademics=url+"saveAcademicDetails.jsp";
public static String url_save_event=url+"saveEvents.jsp";
public static String url_view_user=url+"view_active_users.jsp";
public static String url_userList=url+"userList.jsp";
public static String url_view_all_users=url+"view_all_users.jsp";
public static String url_get_user_details=url+"userDetails.jsp";
public static String url_get_user_profile=url+"profile.jsp";
public static String downloadFile=url+"displayImage.jsp";
public static String url_location=url+"blockList.jsp";
public static String url_location_det=url+"map.jsp";
public static String url_events=url+"eventList.jsp";
public static String url_events_details=url+"eventDetails.jsp";
public static String sendMessage=url+"saveMessage.jsp";
public static String sentMessage=url+"messageSent.jsp";
public static String received_messages=url+"messageReceived.jsp";
```

```
public static String url_manage_users=url+"manageStatus.jsp";

public static String url_manage_vehicles=url+"manageVehicleStatus.jsp";

public static String url_view_tolls=url+"view_active_tolls.jsp";
public static String url_view_blacklists=url+"view_blacklist_vehicles.jsp";
public static String url_view_all_vehicles=url+"view_all_vehicles.jsp";
public static String url_view_my_vehicles=url+"view_my_vehicles.jsp";
public static String url_view_my_trans=url+"view_my_transactions.jsp";
public static String url_get_vehicle_details=url+"vehicleDetails.jsp";
public static String url_get_payment_details=url+"getPaymentDetails.jsp";
public static String saveVehicle=url+"saveVehicle.jsp";
public static String url_get_tolls=url+"view_active_tolls_address.jsp";
public static String view_tolls_loc=url+"view_tolls_location.jsp";
public static String check_vehicle=url+"check_vehicles.jsp";
public static String url_save_payment=url+"save_payment.jsp";
public static String url_get_vehicles=url+"view_user_vehicles.jsp";
public static String url_get_vehicle_location=url+"get_vehicle_location.jsp";
public static String addAmount=url+"recharge.jsp";
public static String forgetPassword=url+"forgetPassword.jsp";
}
```

2.5AdminPanel.java

```
package com.example.tolltax.external.admin;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;

import com.example.tolltax.external.Profile;
import com.example.tolltax.external.R;

public class AdminPanel extends Activity {
String userid;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_admin_panel);
Intent i = getIntent();
userid = i.getStringExtra("userid");
TextView tvWelcome = (TextView) findViewById(R.id.tvWelcome);
tvWelcome.setText("Welcome " + getIntent().getStringExtra("firstname"));
```

```
}

public void registration(View v) {
Intent i = new Intent(getBaseContext(), RegisterUser.class);
i.putExtra("userid", userid);
startActivity(i);
}
public void vehicleRegistration(View v) {
Intent i = new Intent(getBaseContext(), RegisterVehicle.class);
i.putExtra("userid", userid);
startActivity(i);
}

public void profile(View v) {
Intent i = new Intent(getBaseContext(), Profile.class);
i.putExtra("userid", userid);
startActivity(i);
}



public void viewAllVehicles(View v) {
Intent i = new Intent(getBaseContext(), ViewAllVehicles.class);
i.putExtra("userid", userid);

i.putExtra("from", "view_all_vehicles");
startActivity(i);
}

public void listActiveUsers(View v) {
Intent i = new Intent(getBaseContext(), UsersList.class);
i.putExtra("userid", userid);
i.putExtra("from", "view_active_users");
startActivity(i);
}
public void listAllUsers(View v) {
Intent i = new Intent(getBaseContext(), UsersList.class);
i.putExtra("userid", userid);
i.putExtra("from", "view_all_users");
startActivity(i);
}public void listAllTolls(View v) {
Intent i = new Intent(getBaseContext(), ViewTolls.class);
i.putExtra("userid", userid);
i.putExtra("from", "view_active_users");
startActivity(i);
}
```

```
public void logout(View v){
Intent i1 = new Intent(getBaseContext(), com.example.tolltax.external.LoginActivity.class);
i1.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(i1);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.admin_panel, menu);
return true;
}

}
```

CHAPTER -3

3. LITERATURE REVIEW

Here are some objectives about the Toll Tax Payment App which tells us about purpose behind selecting this topic & the requirement of this type of project in our day to day life.

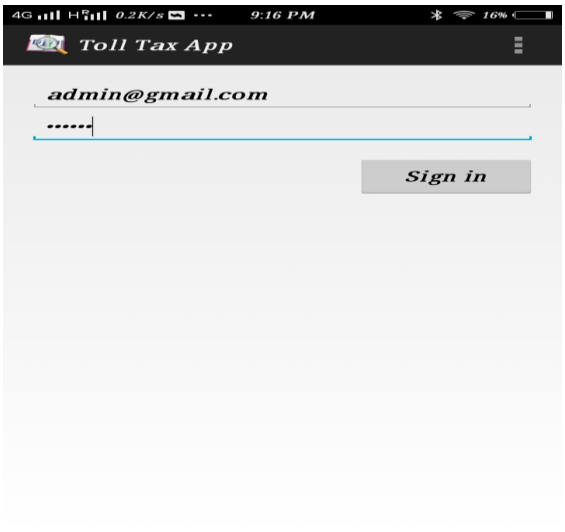
- To avoid the fuel loss.
- To save the time in collecting toll at toll plaza.
- To avoid financial loss.
- To control the traffic.

According to the survey of Karnataka Government, in Sept.2012 they have proposed to get the annual toll collection about 2500 crores/year .But in the present situation they are able to collect only 900 corers of the toll value. Means there is loss of 600 cores due to human errors. So, in this situation we have to control this leakage.
If there would be an automated technology to collect the toll fund, then 1500 crores rupees would be saved.

CHAPTER -4

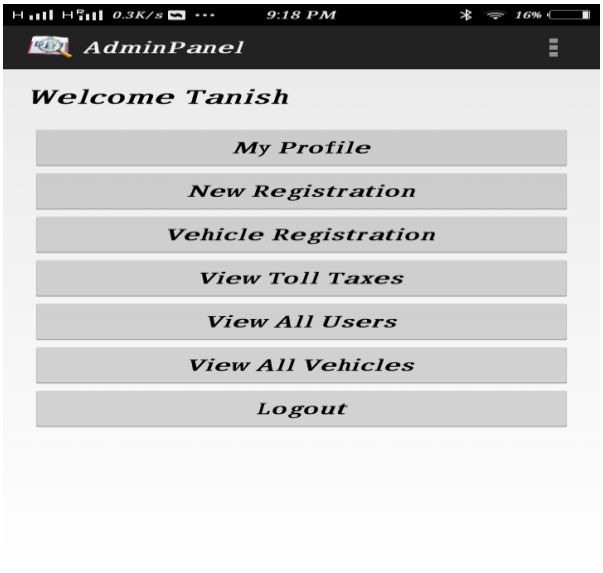
4.1 SYSTEM AND MODULE

Following are the screenshots of our Toll Tax payment Android Application:



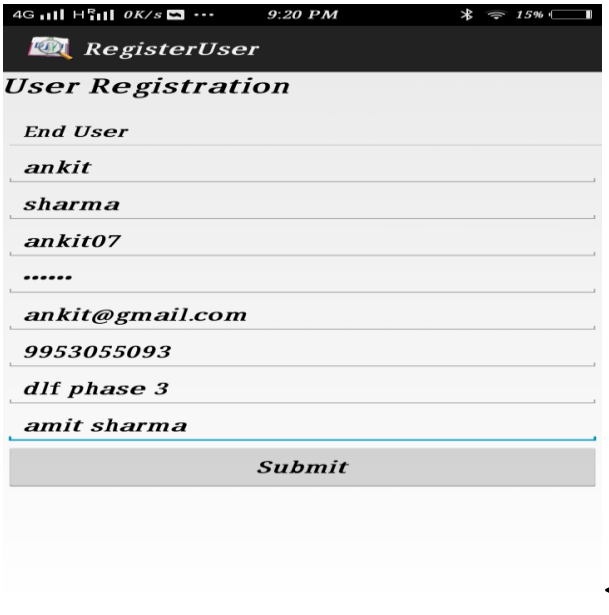
This is the Admin's Panel of Our App.
As soon as the user or admin run the app he/she get this panel opened where he/she can login using his/her registered e-mail ID and password.

Fig-4.1



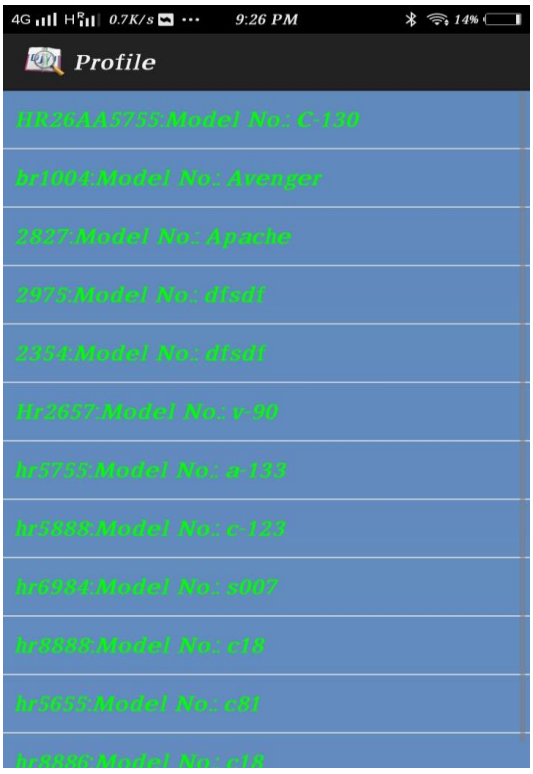
This is the Module of the Admin's panel.
Admin can register any user's information and any toll plaza's details.
Admin can also view its users and their registered vehicles easily and can modify them

Fig-4.2



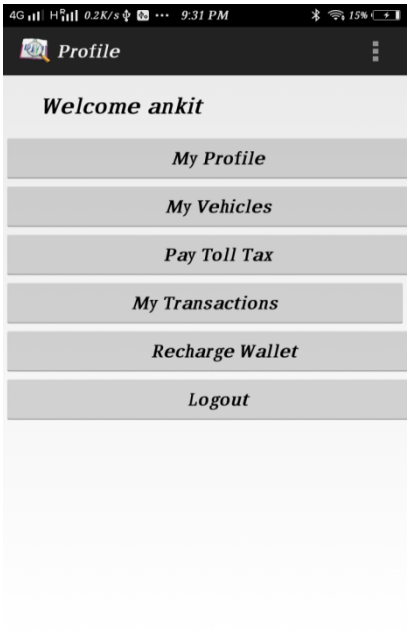
This is the registration module for the Admin's panel.
Admin need to define the kind of user or any other admin he is going to register for.

Fig-4.3



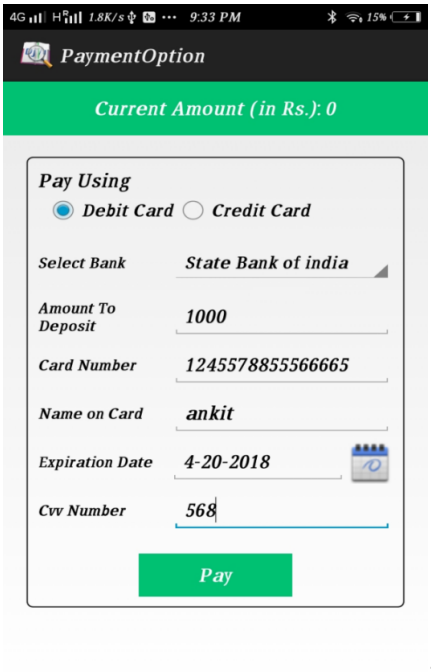
The Admin can view the registered users along with their registered vehicle

Fig-4.4



This is the User's Panel of Our App.
As soon as the user login with his registered email ID he/she will get this panel opened where he/she can do the payment, register his /her wallet and also can view his/her past payments

Fig-4.5

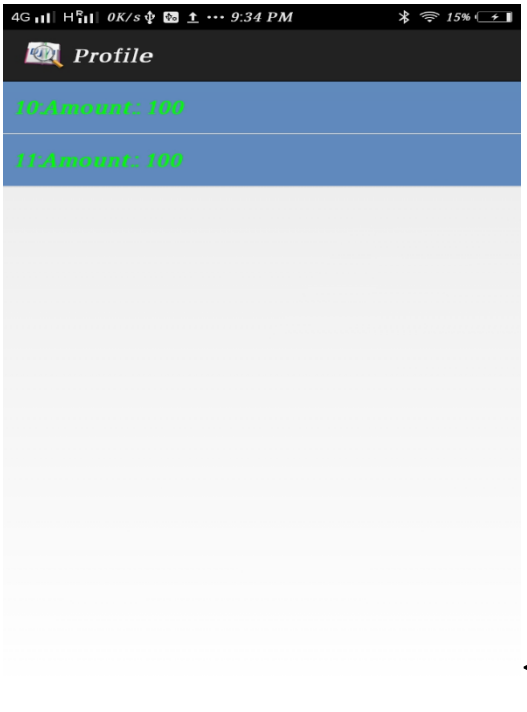


This is the module which is opened in user's panel where he /she can make payment after filling the credentials

Fig-4.6

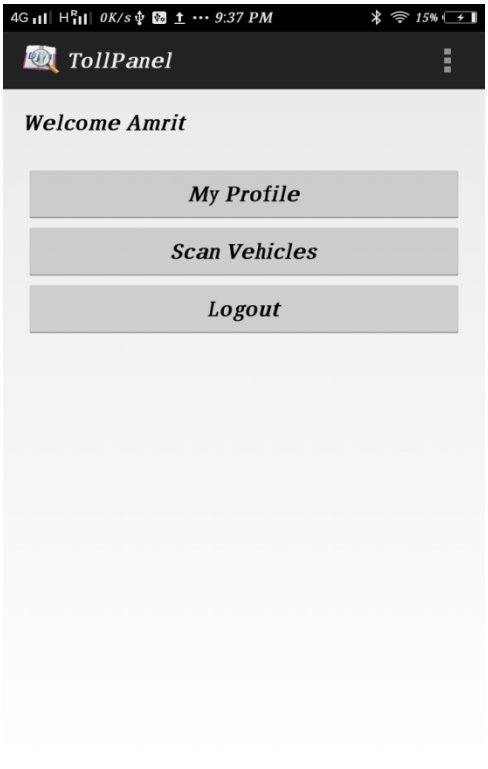
This is panel where user can do the payment for the toll gateways.

Fig-4.7



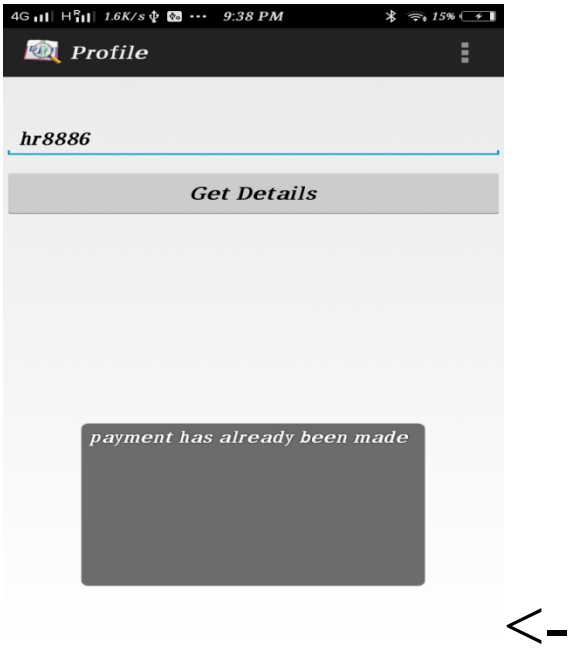
Like the adjacent panel user also get a panel where he/ she can check the payment he have done to the toll gates.

Fig-4.8



This is the Toll gate admin's Panel of Our App.
As soon as the toll admin login with his registered email ID he/she will get this panel opened where he/she can check if payment had been made by the user passing through the toll gate.

Fig-4.9



Like the adjacent figure toll admin can just enter the vehicle details and can check if the payment of that vehicle had been made or not.

Fig-4.10

4.2 REFERENCES

Books:
Android a programmer's guide (Jerome (J.f)) dimerize.
O'Reilly Head First Servlet and JSP.
http://www.codeproject.com/Android/
http://www.vogella.com/tutorials/android.html
http://developer.Android.com/reference/org/json
http://www.javatpoint.com/android-json-parsing-tutorial
http://www.go4expert.com/articles/step-step-guide-sending-data-android-t30182/
http://www.javatpoint.com
http://www.journal.dev.com/2114/servlet-jsp-tutorial
https://www.qoncious.com/questions/creating-new-mysql-datbase-using-phpmyadmin/