A Industrial Oriented Miniproject

On

DEVELOPING QR CODE SCANNER FOR SMART BUS TICKETING SYSTEM

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

Kukatpally, Hyderabad-500 085, Telangana, India

In partial fulfilment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

In

Information Technology

By

AKULA SAI JAHNAVI [17E31A1201]

YASMEEN BEGUM [17E31A1233]

Under the guidance of

V.NARESH KUMAR REDDY

Assistant Professor

IT Dept

ESTD:2001

Department of Information Technology

MAHAVEER INSTITUTE OF SCIENCE AND TECHNOLOGY

(Affiliated to JNTU Hyderabad, Approved by AICTE)

Vyasapuri, Bandlaguda, Post: Keshavgiri, Hyderabad-500 005

2020-2021

MAHAVEER INSTITUTE OF SCIENCE AND TECHNOLOGY

(Affiliated to JNTU Hyderabad, Approved by AICTE)

Vyasapuri, Bandlaguda, Post: Keshavgiri, Hyderabad-500 005

ESTD:2001

CERTIFICATE

This is to certify that this miniproject work report entitled ‘‘**DEVELOPING QR CODE SCANNER SMART BUS TICKETING SYSTEM’’** which is being submited by **AKULA SAI JAHNAVI[17E31A1201], YASMEEN BEGUM[17E31A1233]** in partial fulfilment for the award of the Degree of **Bachelor Of Technology** in **Information Technology**, affiliated of **Jawaharlal Nehru Technological University, Hyderabad** and is a record of the bonafide work carried out by them under our guidance during 2020-2021.

Signature of Project Guide
MR. V.Naresh Kumar Reddy

Signature of Head of the Department
Dr. A. Nanda Gopal Reddy

External Examiner

Principal
Dr. B. Nageshwara Rao

AKNOWLEGEMENT

We would like to express our deep felt appreciation and gratitude to **Mr.V. NARESH KUMAR REDDY** Assistant Professor, Department of IT our project guide, for his skilful guidance, constant supervision, timely suggestion,

keen interest and encouragement in completing the individual seminar within the stipulated time.

We wish to express our gratitude to **Mr. V.NARESH KUMAR REDDY,** Project Coordinator ,who has shown keen interest and even rendered his valuable guidance in terms of suggestions and encouragement extended to us with an immense care and zeal.

We express our profound sense of gratitude to **Dr. A. NANDA GOPAL REDDY**, Head of Department, IT ,who has served as a host of valuable corrections and for providing us time and amenities to complete this miniproject.

We express our thanks to **Dr.B.NAGESHWARA RAO**, Principal of our college and the management of Mahaveer Institute of Science and Technology for providing excellent academic environment in the college.

We wish express our gratitude to the **Members of Staff** and all others who helped us in more than one way. We would also like to thank the **Lab assistants and Programmers** for helping us through our Miniproject.

AKULA SAI JAHNAVI **[17E31A1201]**

YASMEEN BEGUM **[17E31A1233]**

DECLARATION

We hereby declare that the Miniproject entitled "**DEVELOPING QR CODE SCANNER FOR SMART BUS TICKETING SYSTEM"** submitted to partial fulfilment of the requirements for award of the degree of **Bachelor of Technology** at **Mahaveer Institute of Science and Technology,** affiliated to **Jawaharlal Nehru Technology University, Hyderabad** in authentic work and has not been submitted to any university institute for award of any degree.

AKULA SAI JAHNAVI [17E31A1201]

YASMEEN BEGUM [17E31A1233]

CONTENTS

ABSTRACT

In this paper, we are proposing QR reader for bus ticket. Users can scan QR reader instead of ticket. In this app, after registration profile, we have to attach our bank details through in this app. Then whenever we are going on bus, we have to select from and to location. Then it will generate amount details for per head. After that we have give passenger details. Passenger's details mean count. Then we can scan QR code. So directly money will transfer from our bank details. Then we can get SMS alert for ticket payment proof. Then admin (Conductor) side, they calculate amount details through using web application. Then they can calculate per day amount details for bus ticket information. Then admin can generate per day 3 hours report for checking that crowd condition

CHAPTER 1

INTRODUCTION

Buses are the foremost wide used public transportation in many cities nowadays. To improve the standard of Bus Company, a period system that can monitor and predict the rider Flow of the running buses is useful. Here, rider Flow denotes the number of on-board passengers of a bus that varies over time and house. The rider flow will partly mirror the collective human quality on a route and therefore the quality of bus service in term of comfort. From a programming perspective, it tells you the way many folks travel or need to travel on a route. This data will guide the operators to allot and schedule the route and timetable dynamically in fine granularity. Current follow in Bus Transit System operators demonstrates that manual data-collection efforts area unit expensive and usually applicable solely in little scale. The utilization of automatic data-collection systems grow speedily and show nice potential. Automatic Fare assortment (AFC) devices that may record payments of rider's exploitation revolving credit, and a GPS embedded On Board Unit (OBU) that may track the bus area unit wide deployed. With the mature of massive knowledge systems, we've got the chance to estimate and predict the rider flow of each bus in urban wide BTS. To depict the matter additional clear, we will think about a concrete example as shown in Figure one. Many buses operate in a line of route wherever we tend to assume that no passing happens among them on their whole journeys. Passengers get on and off at every station, that changes the rider flows of the buses over time and site. The solid lines and circles illustrate the segments and stations that the buses already travelled before current time, and therefore the dash lines represent the rest of the trips they'll travel. the matter is that given the time data of AFC dealing records and therefore the OBU traces of the buses, the way to estimate the quantity of riders on every bus and how to predict the quantity within the remainder of the trip within the near future

CHAPTER 2

LITERATURE SURVEY

1.Z. Wei, Y. Song, H. Liu, Y. Sheng, X. Wang, "**The research and implementation of GPS intelligent transmission strategy based on on-board Android smartphones**", Computer Science and Network Technology (ICCSNT) 2013 3rd International Conference on, pp. 1230-1233, 2013.

Smartphones have been widely integrated with GPS receiver, which may provide accurate location information of vehicles without cost increase. Traditionally, LBS applications obtain vehicle locations then using the Hypertext Transfer Protocol (HTTP) protocol uploaded to central servers with a fixed frequency. In this paper, we exploit an intelligent strategy of GPS sensing and transmitting. Explicitly, we implemented a platform to collect real-time GPS data from vehicles. A common Android Smartphone serves as a GPS sensor in a vehicle. Client Application software is designed to generate GPS location updates with adaptive time stamps once it executed. In the final comparison, MQTT push technology is introduced into GPS transmission in order to effectively reduce mobile traffic.

2.Y. Chen, T. Kunz, "**Performance evaluation of IoT protocols under a constrained wireless access network**", 2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT), pp. 1-7, 2016.

One of the challenges faced by today's Internet of Things (IoT) is to efficiently support machine-to-machine communication, given that the remote sensors and the gateway devices are connected through low bandwidth, unreliable, or intermittent wireless communication links. In this paper, we quantitatively compare the performance of IoT protocols, namely MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), DDS (Data Distribution Service) and a custom UDP-based protocol in a medical setting. The performance of the protocols was evaluated using a network emulator, allowing us to emulate a low bandwidth, high system latency, and high packet loss wireless access network. This paper reports the observed performance of the protocols and arrives at the conclusion that although DDS results in higher bandwidth usage than MQTT, its superior performance with regard to data latency and reliability makes it an attractive choice for medical IoT applications and beyond.

3.K. Tanaka, K. Naito, "**Demo: Implementation of unconscious bus location sensing system with smartphone devices and beacon devices**", 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 280-281, 2016.

This paper demonstrates a new unconscious sensing system for bus location. Our system is a new type of application based on participatory sensing systems. However, it can perform sensing operation without users' operation. Therefore, we can employ the mechanism to realize practical application such as bus location systems. Our sensing system consists of a beacon device, a smartphone application and a cloud service. The beacon device is installed on a bus to activate the smartphone application. The smartphone application can upload a bus location to the cloud service when the smartphone application detects the beacon device. The cloud service manages the bus location and distributes them for smartphone applications. The demonstration shows a prototype system for a bus location system based on the new participatory sensing mechanism.

4.J. Gong, M. Liu, S. Zhang, "**Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time GPS data**", 2013 25th Chinese Control and Decision Conference (CCDC), pp. 972-976, 2013.

Advanced traveler information systems (ATIS) are one component of intelligent transportation systems (ITS), and a major component of ATIS is travel time information. Global positioning system-based automatic vehicle location (AVL) systems have been adopted by many transit agencies for tracking their vehicles and predicting travel time in real time. It is a very important subject to improve the precision and reliability of the prediction model which can attract additional ridership, reduce passengers' anxieties and waiting times at bus stop, and increase their satisfaction. Furthermore, it can promote the development of city public transportation. This paper presents an improved approach to predict the public bus arrival time based on historical and real-time GPS data. After analyzing the components of bus arrival time systematically, the bus arrival time and dwell time at previous stops are chosen as the main input variables of the prediction model. At first, the algorithm of data interpolation and processing is designed to get the real-time GPS data as the input variables of the prediction models. Secondly, the statistical model is obtained based on the historical data of average running time of each link and dwelling time of each stop at given time-of-day and day-of-week, respectively. Thirdly, a hybrid dynamic prediction model is proposed to predict the bus arrival

time. Finally, Actual GPS data from bus route 244 located in Shenyang, CHINA are used as a test bed. The index of Mean Absolute Percentage Error (MAPE) is used to evaluate the three models. The results show that the improved model outperforms the historical data based model in terms of prediction accuracy.

5.L. Singla, P. Bhatia, "**GPS based bus tracking system**", Computer Communication and Control (IC4) 2015 International Conference on, pp. 1-6, 2015.

In this fast life, everyone is in hurry to reach their destinations. In this case waiting for the buses is not reliable. People who rely on the public transport their major concern is to know the real time location of the bus for which they are waiting for and the time it will take to reach their bus stop. This information helps people in making better travelling decisions. This paper gives the major challenges in the public transport system and discuses various approaches to intelligently manage it. Current position of the bus is acquired by integrating GPS device on the bus and coordinates of the bus are sent by either GPRS service provided by GSM networks or SMS or RFID. GPS device is enabled on the tracking device and this information is sent to centralized control unit or directly at the bus stops using RF receivers. This system is further integrated with the historical average speeds of each segment. This is done to improve the accuracy by including the factors like volume of traffic, crossings in each segment, day and time of day. People can track information using LEDs at bus stops, SMS, web application or Android application. GPS coordinates of the bus when sent to the centralized server where various arrival time estimation algorithms are applied using historical speed patterns.

6.Foisal Mahedi Hasan et al., "**RFID-based Ticketing for Public Transport System: Perspective Megacity Dhaka**", 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 6, pp. 459-462, 2010.

The paper based public transport ticketing system, prevailing in the megacity Dhaka (Bangladesh), introduces severe malfunction in the system, malicious argument among public, corruption and most of all traffic jam. This paper actually suggests a much more public friendly, automated system of ticketing as well as the credit transaction with the use of RFID based tickets. The total system mainly acts to bring out the consistency among various bus agencies that will conclude in uniform access of passengers in daily rides through an automated server being updated every single time the passengers travel by carrying the RFID based tickets.

7.R. Hamon, P. Borgnat, P. Flandrin, C. Robardet, "**Networks as signals with an application to bike sharing system**" in Global SIP 2013, Austin, Texas, USA:, Dec. 2013.

Dynamic graphs are commonly used for describing networks with a time evolution. A method has been proposed to transform these graphs into a collection of signals indexed by vertices. This approach is here further explored in a number of different directions. First, the importance of a good indexing of a graph is stressed, and a solution is proposed using a node labeling algorithm which follows the structure of the graph. Second, a spectral analysis of identified signals is performed to compute features linked to graph properties such as regularity or structure in communities. Finally, these features can be tracked over time to evidence the structure evolution of the graph. As a case study, the approach is applied to a dynamic graph based on a dataset of trips made using the bike sharing system Vlov in use in Lyon, France. This is shown to offer specific insights on behaviors of bike users over time in two districts of the city.

8.B. Danila, Y. Yu, J. K. Marsh, K. Bassler, "**Optimal transport on complex nets**", Phys. Rev. E, vol. 74, pp. 046106–6, October 2006.

We present a heuristic algorithm for the optimization of transport on complex networks. Previously proposed network transport optimization algorithms aim at avoiding or reducing link overload. Our algorithm balances traffic on a network by minimizing the maximum node betweenness with as little path lengthening as possible, thus being useful in cases when networks are jamming due to node congestion. By using the resulting routing, a network can sustain significantly higher traffic without jamming than in the case of shortest path routing.

9.R. Hua-Ling, "**Origin-Destination Demands Estimation in Congested Dynamic Transit Networks**", International Conference on Management Science & Engineering (14th), 2007.

This paper investigates the problem of estimation of time-dependent passenger origin-destination (OD) matrices in congested transit networks where real-time updated passenger counts and prior OD matrices are available. A bilevel programming model is proposed for the dynamic estimation of passenger OD matrix. The upper level minimizes the sum of error measurements in dynamic passenger counts and time-dependent OD matrices, and the lower level is a new schedule-based dynamic transit assignment model that can determine simultaneously the dynamic average travel costs and route choices of passengers in congested

transit networks. The lower-level problem can be formulated as a variational inequality problem. A heuristic solution algorithm is adapted for solving the proposed bilevel programming model. Finally, a numerical example is used to illustrate the applications of the proposed model and solution algorithm.

10.P. Verma, J.S.Bhatia, "**Design and Development of GPS-GSM based Tracking System with Google Map based Monitoring**", International Journal of Computer Science, Engineering and Applications (IJCSEA), vol. 3, no. 3, pp. 33-40, 2013.

GPS is one of the technologies that are used in a huge number of applications today. One of the applications is tracking your vehicle and keeps regular monitoring on them. This tracking system can inform you the location and route travelled by vehicle, and that information can be observed from any other remote location. It also includes the web application that provides you exact location of target. This system enables us to track target in any weather conditions. This system uses GPS and GSM technologies. The paper includes the hardware part which comprises of GPS, GSM, Atmega microcontroller MAX 232, 16x2 LCD and software part is used for interfacing all the required modules and a web application is also developed at the client side. Main objective is to design a system that can be easily installed and to provide platform for further enhancement.

11.J. Gong, M. Liu, S. Zhang, "**Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time GPS Data**," 2013 25th Chinese Control and Decision Conference (CCDC), pp. 972 976, 2013.

Advanced traveller information systems (ATIS) are one component of intelligent transportation systems (ITS), and a major component of ATIS is travel time information. Global positioning system-based automatic vehicle location (AVL) systems have been adopted by many transit agencies for tracking their vehicles and predicting travel time in real time. It is a very important subject to improve the precision and reliability of the prediction model which can attract additional ridership, reduce passengers' anxieties and waiting times at bus stop, and increase their satisfaction. Furthermore, it can promote the development of city public transportation. This paper presents an improved approach to predict the public bus arrival time based on historical and real-time GPS data. After analyzing the components of bus arrival time systematically, the bus arrival time and dwell time at previous stops are chosen as the main input variables of the prediction model. At first, the algorithm of data interpolation and

processing is designed to get the real-time GPS data as the input variables of the prediction models. Secondly, the statistical model is obtained based on the historical data of average running time of each link and dwelling time of each stop at given time-of-day and day-of-week, respectively. Thirdly, a hybrid dynamic prediction model is proposed to predict the bus arrival time. Finally, Actual GPS data from bus route 244 located in Shenyang, CHINA are used as a test bed. The index of Mean Absolute Percentage Error (MAPE) is used to evaluate the three models. The results show that the improved model outperforms the historical data based model in terms of prediction accuracy.

12.W. El-Medany, A. Al-Omary, R. Al-Hakim, S. Al-Irhayim, M. Nusaif, "**A Cost Effective Real-Time Tracking System Prototype Using Integrated GPS/GPRS Module**", Sixth International Conference on Wireless and Mobile Communication, pp. 521-525, 20-25 September, Valencia, 2010.

This paper presents a real time tracking system that provides accurate localizations of the tracked vehicle with low cost. The system is implemented using GM862 cellular quad band module. A monitoring server and a graphical user interface on a website have also been implemented using Microsoft SQL Server 2003 and ASP.net to view the current location of a vehicle on a specific map. The system provides information regarding the vehicle status such as speed, mileage. The prototype has been tested experimentally and the results are analyzed and discussed. The experiments are conducted in different areas on Kingdom Of Bahrain using Google maps.

SOFTWARE ENVIRONMENTS

ANDROID

It is a free, open source mobile platform. Linux-based, multi process, Multithreaded OS. Android is not a device or a product It's not even limited to phones You could build a DVR, a handheld GPS, an MP3 player, etc.Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

- Makes mobile development easy.
- Full phone software stack including applications
- Designed as a platform for software development
- Android is open
- Android is free
- Community support
- **July 2005**
    - Google acquired Android Inc.
- **5 Nov 2007**
    - Open HandSet Alliance formed-
    - Google, HTC, Intel, Motorola, Qualcomm,T-Mobile
- Android is the OHA first product
- **12 Nov 2007**
    - OHA released a preview of the Android OHA
- **Oct-2008**
    - First Device – **T-Mobile G1** was released.
    - Latest – **Motorola Milestone** released in India.

Features

- **Application framework** enabling reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices
- **Integrated browser** based on the open source WebKit engine

- **Optimized graphics** powered by a custom 2D graphics library; 3Dgraphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQLite** for structured data storage
- **Media support** for common audio, video, and still image formats(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth,EDGE, 3G, and WiFi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator ,tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

Linux Kernel

Android relies on Linux version 2.6 for core system services such as

- security
- memory management
- process management
- network stack
- driver model
- The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

Android Runtime

- Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.
- Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently.
- The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint.
- The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Libraries

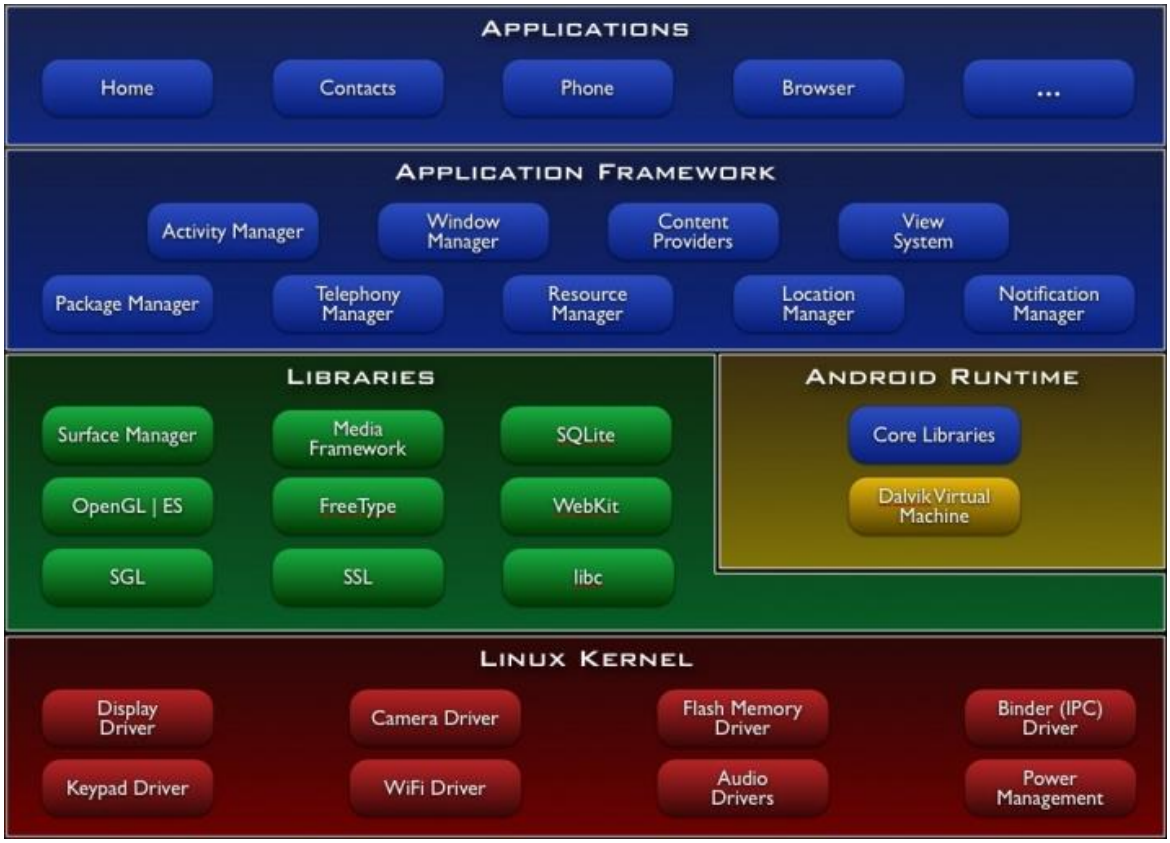
Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.

- System C Library
- Media Library
- Surface Manager
- LibWebCore
- SGL
- 3D libraries
- Free Type
- SQLite

Application Framework

- Being a open development platform, Android offers developers the ability to build extremely rich and innovative applications.
- Developers have full access to the same framework APIs used by the core applications.
- **Views** – used to build applications (lists, grid, buttons, text boxes and even embeddable web browser)
- **Content providers** – enable applications to access data from other applications or share their own data.
- **Resource manager** – provides access to non-code resources such as localized strings, graphic and layout files.
- **Notification manager** – enables applications to display custom alerts in status bar
- **Activity manager** – manages lifecycle of applications and provides navigation backstack
- Android ships with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others.
- All applications are written using the Java programming language.

Architecture



Anatomy of an Android Application

There are four building blocks for an Android application:

- **Activity** - a single screen
- **Broadcast Receiver**- to execute in reaction to an external event(Phone Ring)
- **Service** - code that is long-lived and runs without a UI(Media Player)
- **Content Provider** - an application's data to be shared with other applications

Android Building Blocks

These are the most important parts of the Android APIs:

- **AndroidManifest.xml**
  - the control file-tells the system what to do with the top-level components
- **Activities**
  - an object that has a life cycle-is a chunk of code that does some work.

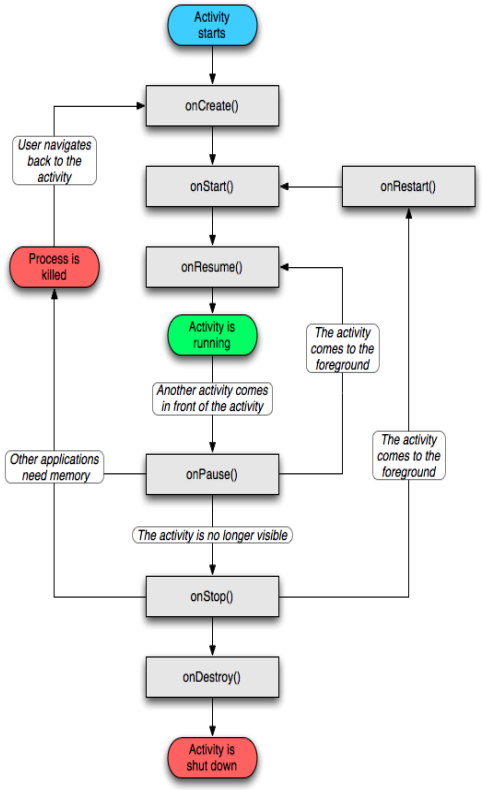
- **Views**
  - an object that knows how to draw itself to the screen
- **Intents**
  - a simple message object that represents an "intention" to do something
- **Notifications**
  - is a small icon that appears in the status bar(SMS messages)
  - for alerting the user
- **Services**
  - is a body of code that runs in the background

Development Tools

The Android SDK includes a variety of custom tools that help you develop mobile applications on the Android platform. Three of the most significant tools are:

- **Android Emulator** -A virtual mobile device that runs on our computer -use to design, debug, and test our applications in an actual Android run-time environment
- **Android Development Tools Plugin** -for the Eclipse IDE - adds powerful extensions to the Eclipse integrated environment
- **Dalvik Debug Monitor Service (DDMS)** -Integrated with Dalvik -this tool let us manage processes on an emulator and assists in debugging
- **Android Asset Packaging Tool (AAPT)** – Constructs the distributable Android package files (.apk)
- **Android Debug Bridge (ADB)** – provides link to a running emulator. Can copy files to emulator, install .apk files and run commands.

Lifecycle of activity

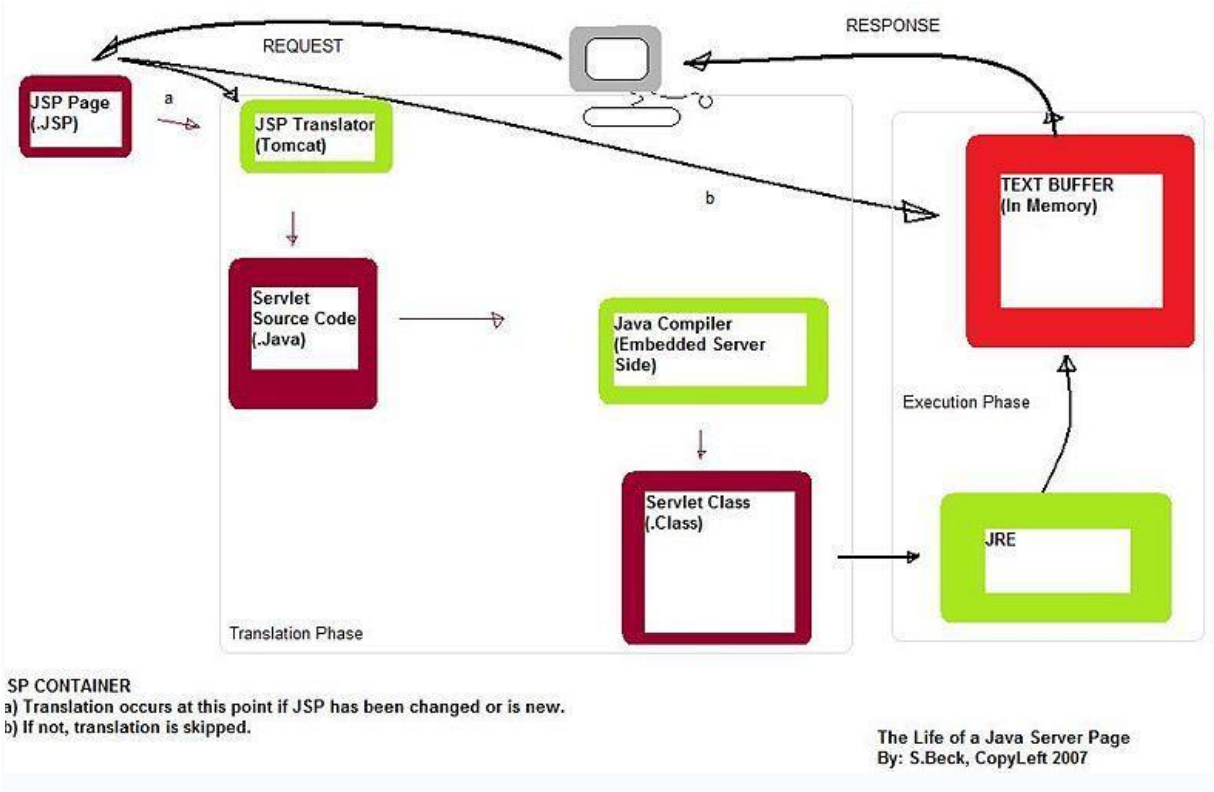


JSP:

JavaServer Pages (JSP) is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

JSPs are compiled into Java Servlets by a JSP compiler. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, or it may generate byte code for the servlet directly. JSPs can also be interpreted on-the-fly reducing the time taken to reload changes JavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server- and platform-independent.

Architecture OF JSP:



The Advantages of JSP:

- Active Server Pages (ASP). ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
- Pure Servlets. JSP doesn't give you anything that you couldn't in principle do with a servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people on different tasks: your Web page design experts can build the HTML, leaving places for your servlet programmers to insert the dynamic content.
- Server-Side Includes (SSI). SSI is a widely-supported technology for including externally-defined pieces into a static Web page. JSP is better because it lets you use servlets instead of a separate program to generate that dynamic part. Besides, SSI is

really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

- JavaScript. JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment. With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like.
- Static HTML. Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

SERVLETS

The Java Servlet API allows a software developer to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets are the Java counterpart to non-Java dynamic Web content technologies such as PHP, CGI and ASP.NET. Servlets can maintain state across many server transactions by using HTTP cookies, session variables or URL rewriting.

The Servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of a Web container and a servlet. A Web container is essentially the component of a Web server that interacts with the servlets. The Web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

A Servlet is an object that receives a request and generates a response based on that request. The basic servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the Web server and a client. Servlets may be packaged in a WAR file as a Web application.

Servlets can be generated automatically by JavaServer Pages (JSP), or alternately by template engines such as WebMacro. Often servlets are used in conjunction with JSPs in a pattern called "Model 2", which is a flavor of the model-view-controller pattern.

Servlets are Java technology's answer to CGI programming. They are programs that run on a Web server and build Web pages. Building Web pages on the fly is useful (and commonly done) for a number of reasons:

- The Web page is based on data submitted by the user. For example the results pages from search engines are generated this way, and programs that process orders for e-commerce sites do this as well.
- The data changes frequently. For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date.
- The Web page uses information from corporate databases or other such sources. For example, you would use this for making a Web page at an on-line store that lists current prices and number of items in stock.

The Servlet Run-time Environment:

A servlet is a Java class and therefore needs to be executed in a Java VM by a service we call a servlet engine.

The servlet engine loads the servlet class the first time the servlet is requested, or optionally already when the servlet engine is started. The servlet then stays loaded to handle multiple requests until it is explicitly unloaded or the servlet engine is shut down.

Some Web servers, such as Sun's Java Web Server (JWS), W3C's Jigsaw and Gefion Software's LiteWebServer (LWS) are implemented in Java and have a built-in servlet engine. Other Web servers, such as Netscape's Enterprise Server, Microsoft's Internet Information Server (IIS) and the Apache Group's Apache, require a servlet engine add-on module. The add-on intercepts all requests for servlets, executes them and returns the response through the Web server to the client. Examples of servlet engine add-ons are Gefion Software's WAICoolRunner, IBM's WebSphere, Live Software's JRun and New Atlanta's ServletExec.

All Servlet API classes and a simple servlet-enabled Web server are combined into the Java Servlet Development Kit (JSDK), available for download at Sun's official Servlet site.To get started with servlets I recommend that you download the JSDK and play around with the sample servlets.

Life Cycle OF Servlet

The Servlet lifecycle consists of the following steps:

1. The Servlet class is loaded by the container during start-up.
2. The container calls the init() method. This method initializes the servlet and must be called before the servlet can service any requests. In the entire life of a servlet, the init() method is called only once.
3. After initialization, the servlet can service client-requests. Each request is serviced in its own separate thread. The container calls the service() method of the servlet for every request. The service() method determines the kind of request being made and dispatches it to an appropriate method to handle the request. The developer of the servlet must provide an implementation for these methods. If a request for a method that is not implemented by the servlet is made, the method of the parent class is called, typically resulting in an error being returned to the requester.

Finally, the container calls the destroy() method which takes the servlet out of service. The destroy() method like init() is called only once in the lifecycle of a Servlet**.**

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the general way, every bus is controlled by a conductor. The conductor will collect money from each passenger and issue ticket. Initially, printed papers or tokens are used as tickets. Nowadays, handheld machines are used to print tickets. This system has many disadvantages. The passenger have to carry the ticket till the end of travel, the conductor should ensure that everyone has got the ticket, the time taken for ticketing is comparatively more and more amount of paper is needed to print the Ticket. Nowadays conductors are trained to operate the handheld ticketing machine. For example, if a passenger wish to travel in bus. He has to carry money with him. Then conductor will collect the money and he will give ticket. This has to repeat for all passengers. This will take more time and waste of human resource as well as energy. Even handheld ticketing machine is comparatively slow and need trained person to operate it.

In Existing system RFID Reader is used to read the RFID tag but destination should be entered by passenger in keyboar , So that amount will be debited automatically from the tag. Here if once destination is arrived, bus stops automatically and intimate with buzzer sound. Fairly such arrangement consumes more time in case of accessing of tag by every individual, so to overcome that, implementation of ticketing system without access is developed in this proposal with addition of application to transfer information about accident occurrence.

Disadvantage:

1. Hardware debugging is the major problem.
2. We fabricated all the components in PCB and test power supply; input and output.
3. If the ports are not working then check the code and rework in hardware.

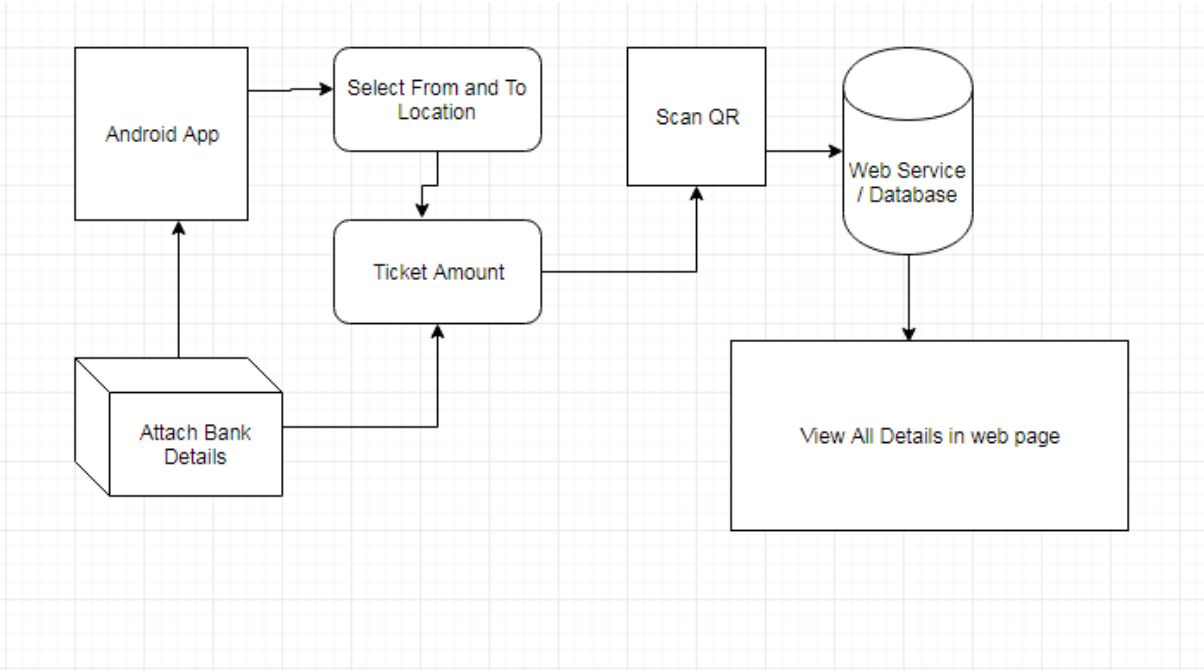
3.2 PROPOSED SYSTEM

In proposed method, we are introducing QR reader. Here, we will create one android application for select travelling route and generate amount. After generating amount, user has to read that QR image. Then automatically it will send amount from our bank details or wallet. Each conductor having one QR reader and after reading that values automatically it will store in database. Then user will get message for travelling ticket.

Advantage:

1. No hardware debugging.
2. No change (Amount) problem.

3.3 SYSTEM ARCHITECTURE



3.4 FEASIBILITY STUDY

This paper investigates the problem of estimation of time-dependent passenger origin-destination (OD) matrices in congested transit networks where real-time updated passenger counts and prior OD matrices are available. A bilevel programming model is proposed for the dynamic estimation of passenger OD matrix. The upper level minimizes the sum of error measurements in dynamic passenger counts and time-dependent OD matrices, and the lower level is a new schedule-based dynamic transit assignment model that can determine simultaneously the dynamic average travel costs and route choices of passengers in congested transit networks. The lower-level problem can be formulated as a variational inequality problem. A heuristic solution algorithm is adapted for solving the proposed bilevel programming model. Finally, a numerical example is used to illustrate the applications of the proposed model and solution algorithm.

3.5 SYSTEM REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS

1. OS – Windows 7, 8
2. RAM – Min 4GB
3. Android Mobile

3.5.2 SOFTWARE REQUIREMENTS

1. JDK
2. Android Eclipse (ADT-Bundle)
3. Net beans IDE
4. Mysql and SQLyog

3.5.3 FUNCTIONAL REQUIREMENTS

3.5.3.1 Function1: User Registration:

Purpose: The purpose of registration is to attach all the images of the moves of unauthorized intruder.

Inputs: The user will enter details in the registration form according to required fields, some of the fields include are username, password, confirm password, first name etc

Processing: On the server side the servlet is used to capture the details of the user and then store the details in the database.

Output: After the registration the user will be directed to the main home

Error Handling: If error is encountered than try to handle it with some suitable exception handling method

3.5.4 NONFUNCTIONAL REQUIREMENTS

Some of the non-functional requirements are as follows:

1. **Performance**-is high as only it gets processed when necessary.

2. **Reliability** – is high as only registered users will get the intruder's images

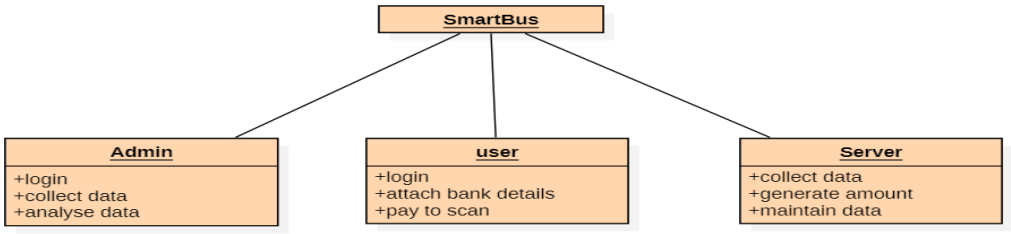
3. **Security**- It provides security to the data by authentication, only authorized user access it.

4. **Logical Database Requirements** – data base is needed for storing the details of the users

CHAPTER 4

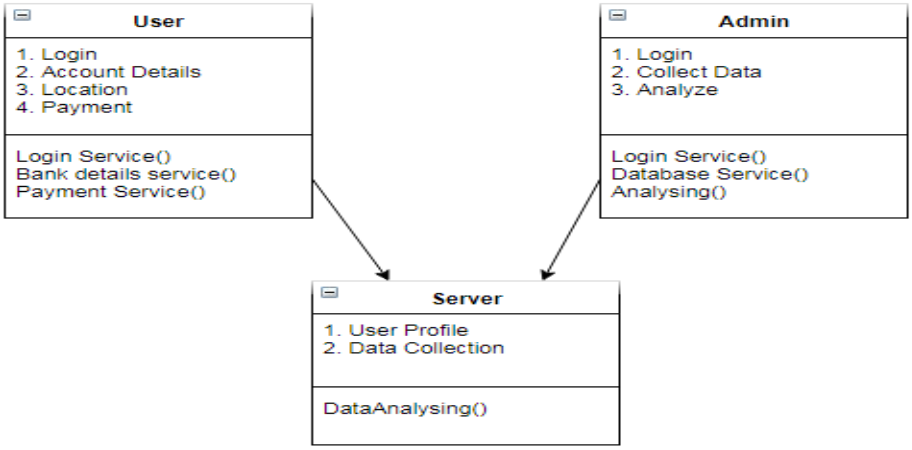
SYSTEM DESIGN

Object Diagram:



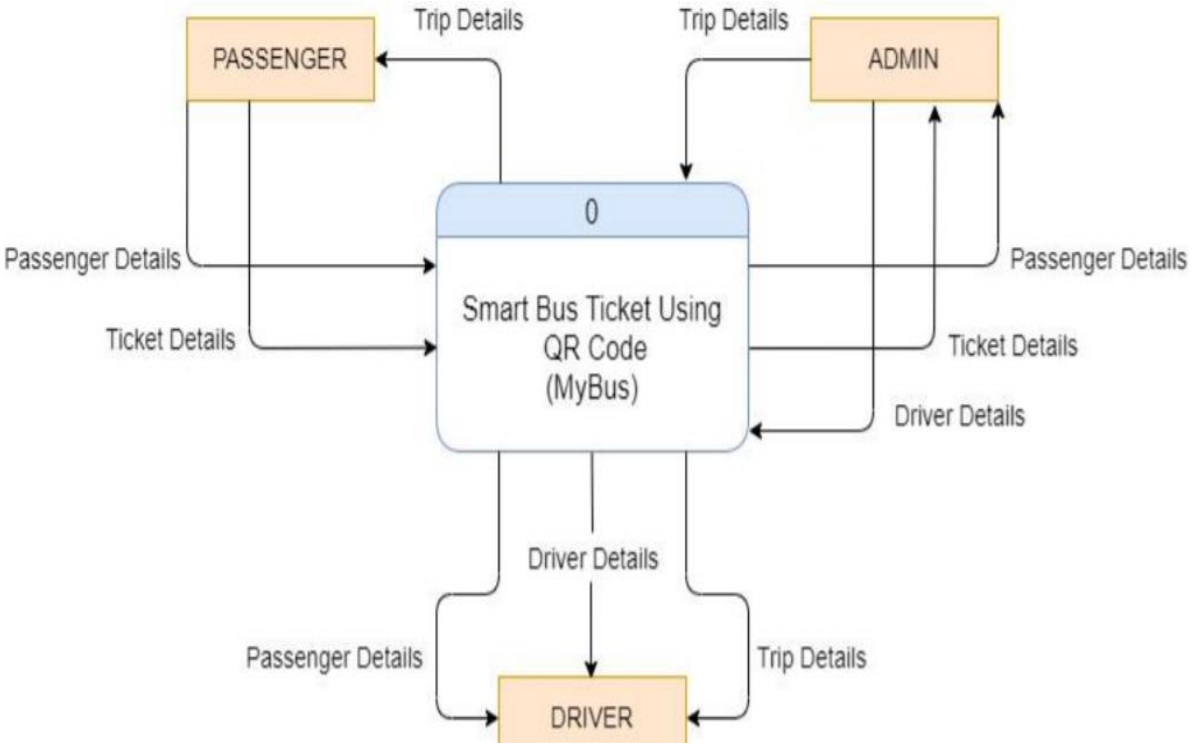
The figure represents the object diagram. An object diagram shows a complete or partial view of the structure of a modeled system at a specific time.Here objects are admin, user, driver. Admin,,user and driver can view the SmartBus.

Class Diagram:



The figure represents class diagram. A class diagram is static structure diagram that specifies the structure of system by showing the systems classes ,their attributes, operations and the relationships among objects. Here user, admin and server are the classes ,their attributes and operations are shown in the figure.

Context Diagram:

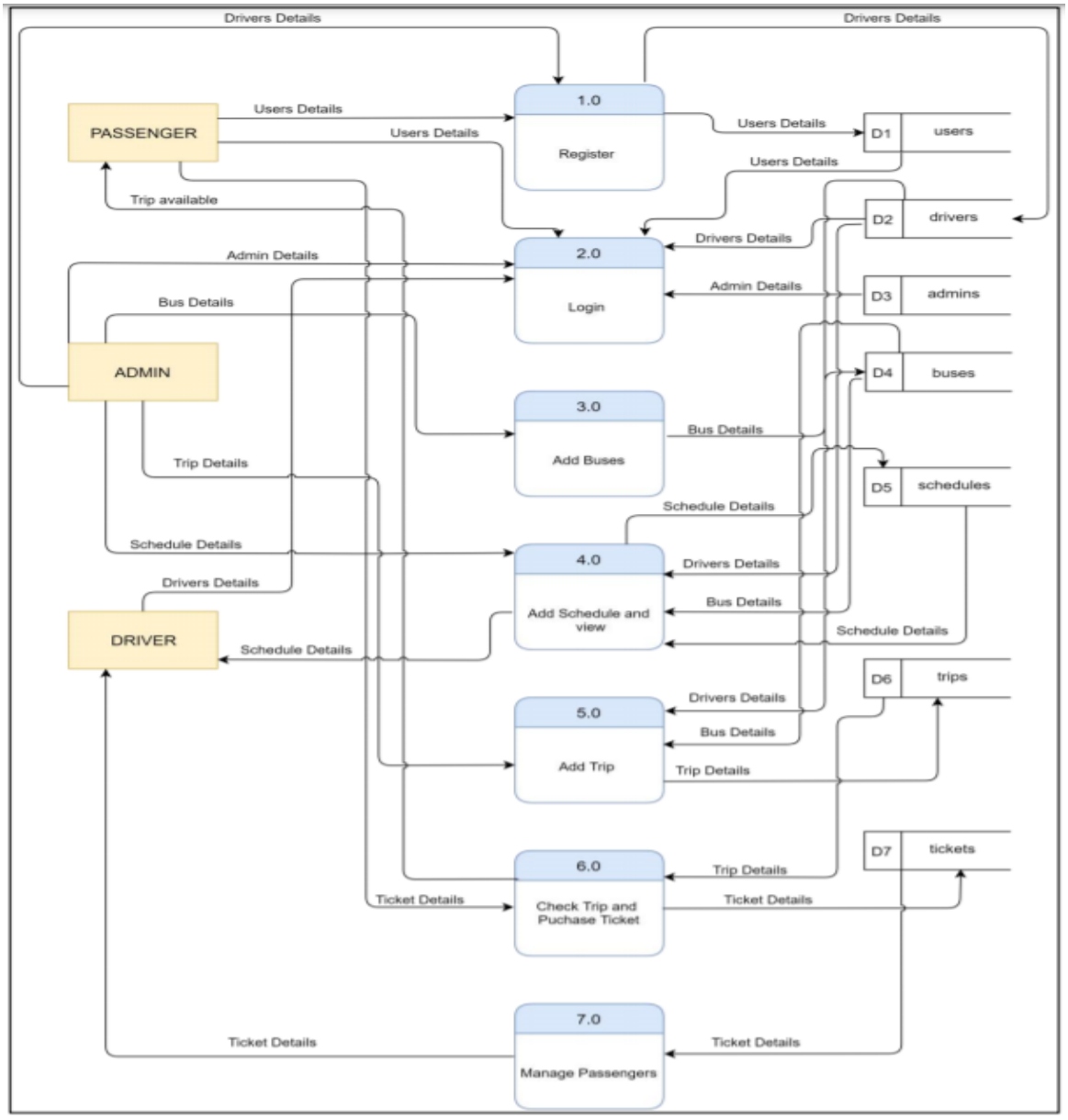


The figure reperesents context diagram,as shown in figure as there are three entities involved in the Smart Bus Ticket Using QR Code which is Passenger, Driver, and Admin. The data flow coming from the Passenger shows the data that passenger provides to the system in order to complete the process of registering, checking trips, and purchasing tickets. The trip details will be given to the Passenger once they bought the ticket.

For the Driver, the registration process does not require for drivers because the admin will register the driver. Other than that,in order to complete the process of getting trip details.Driver will be provided with trip details and passenger details. This is to ensure the number of passengers who is supposed to board and get off from the bus.

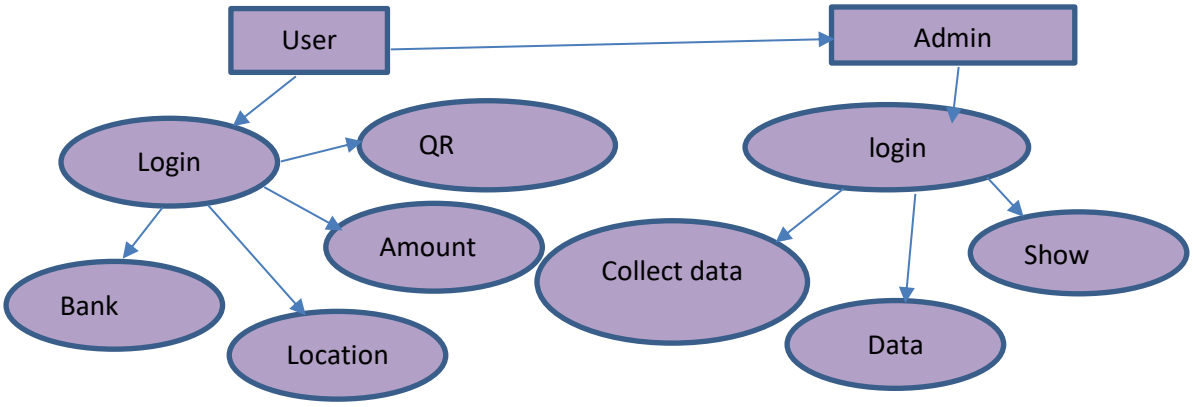
For the Admin, registering process for bus driver requires Driver's details from the Driver into the system. For report management,Admin will be provided with trip details and passenger detail. Lastly, admin will add new destination if have request from regular customer.

DataFlow Diagram:



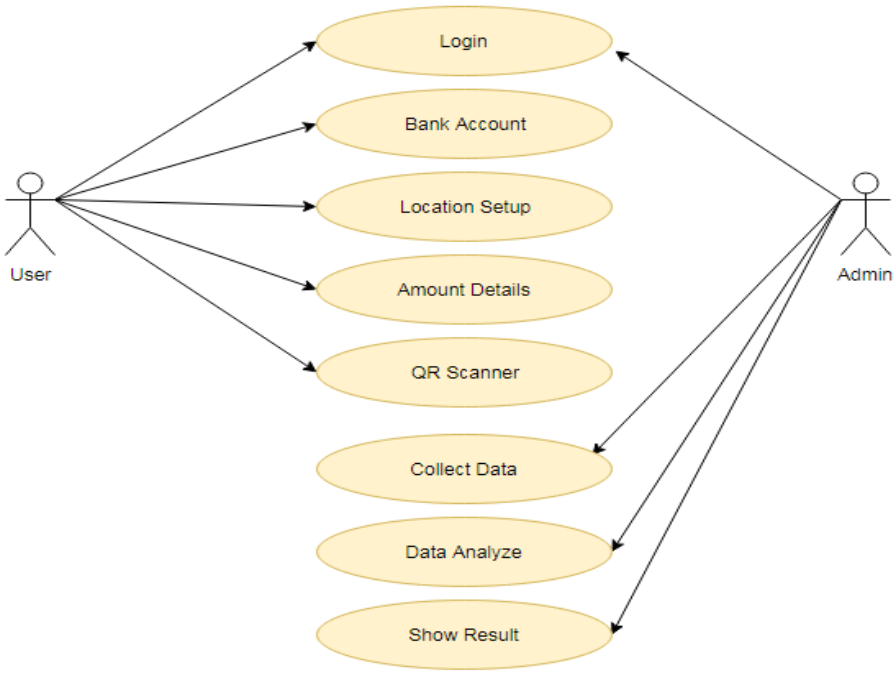
The figure shows the DataFlow diagram.It shows the processes and data stores involve under the process purchase ticket in data flow diagram level 0. There are five processes and two data stores involved. The processes are check date, check board location, check destination,check trip time, select seat and payment. On the other hand, the data stores are D6 Trips and D7 Tickets

ER Diagram:



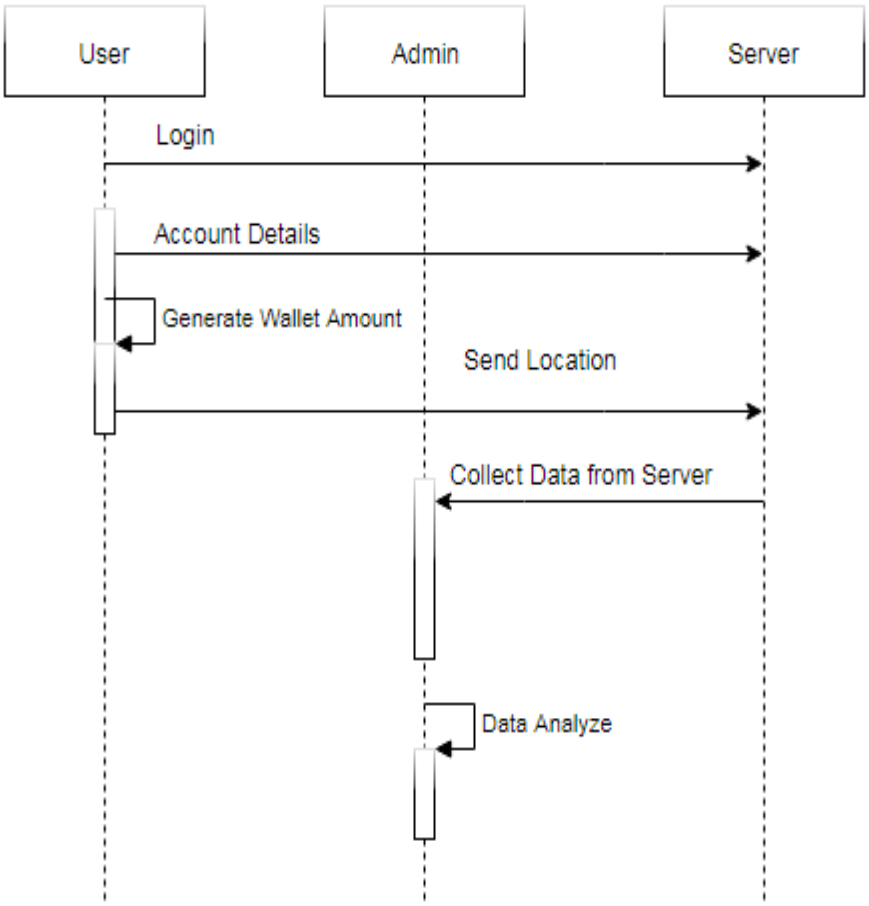
The figure show the entity relationship diagram. The relation among the user and admin is shown. User is an one entity ,user can login or do registration by providing the details. The user by selecting to and from location ,scanning the Qr code ,the amount will be directly deducted as per the location he selected, amount will transferred from his bank. Admin is another entity. Admin can login ,collect the data, show the destination, analyse the users details, manages the users.

Usecase Diagram:



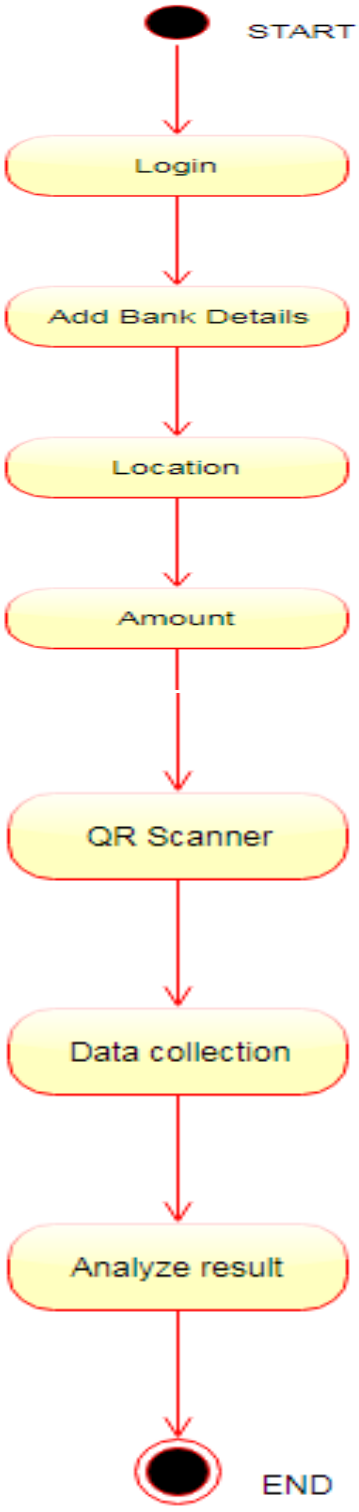
The figure shows the Usecase diagram. Here the actors are user and admin.Usecases represents the actions performed by the both user and admin. Actor as a user can login, attach the bank details, location setup, add the amount details, scans the QR code. Actor as a admin can performed action such as collect data of different users, analyse the data, show the result based on location they selected.

Sequence Diagram:



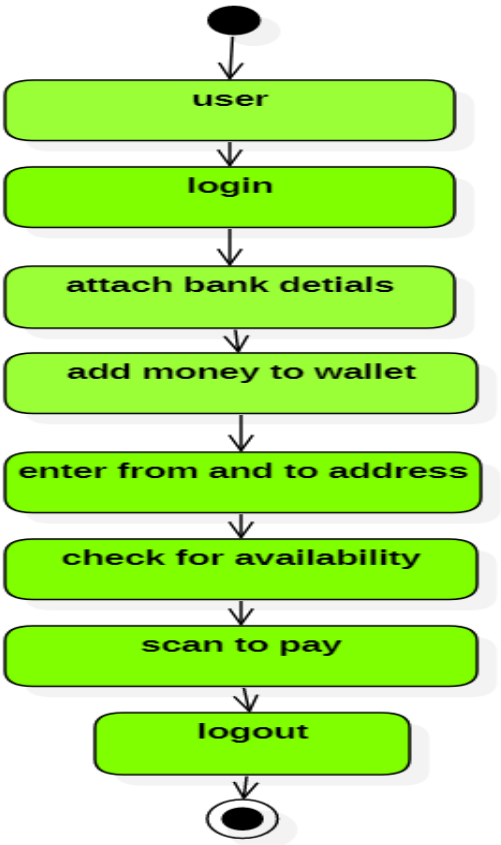
The figure represents the Sequence diagram.It shows the object interaction arranged in time sequence.The parallel vertical line are lifelines ,different objects that live simultaneously and horizontal lines are messages.The sequence of messages exchanged among object.Here user,admin and server are the objects.The messages are exchanged between them.For the user ,user can login,add the account details ,send location to the server and generate the wallet amount.For the admin,admin can analyse the data.Server can collect he data from different users and show the results to both admin and user.

Activity Diagram:



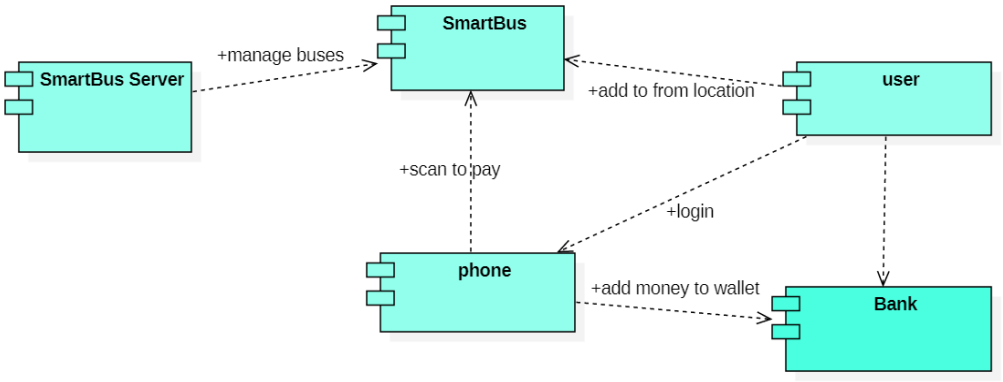
The figure represents the Activity diagram. It is a behavourial diagram .It depicts behaviour of the system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. Here the activity start with the login, the user or admin can login with their details.The user activities are user can attach bank details, select location, scans the Qr code, add money from wallet. The admin activities such admin can collect the data,analyse the data and show the result.

StateChart Diagram:



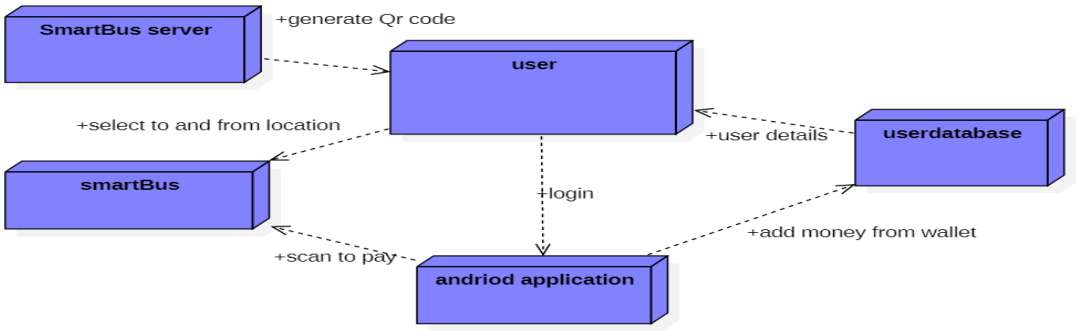
The figure represents the State chart diagram. A Sate chart diagram is used to represent the condition of the system or part of the system at the finite instances of time. It is a behavioural diagram and it represents the behaviour using finite state transitions. It specifies the sequences of states that an object can be in event and condition. Here the user behaviour can be represented as state transitions. The user perform action can be represented in sequence of states. The states are the action done by the user such as user can login, attach the bank details, add money to wallet, enter from and to address, check the availability of the buses,scan to pay and transfer money and logout .

Component Diagram:



The figure represents the component diagram. Component diagram are used for modelling subsystems. Its represents how each and every components acts during execution and running of the system program. They are also used to show and represent structure and organization of all components. Here in figure SmartBus,SmartBus server,user,phone,bank are the components are involved .Each components has its own task to perform.

Deployment Diagram:



The figure represents the Deployment diagram. The deployment diagram visualizes the physical hardware on which the software will be deployed. It involves the nodes and their relationship. It maps the software architecture,where the software will be executed as node. since it involves many nodes and their relations is shown by utilizing communication path. Here the nodes such as SmartBus server, smartBus,user,userdatabase , android application and their relationship shows the involvement of each components to execute.

CHAPTER 5

IMPLEMENTATION AND CODING

5.1 MODULES

- Admin
- User
- Server

5.2 MODUES DESCRIPTION

User registration

We will create one android application for users. Users can register them in android application. Then user can add bank details with them profile. Users can select from and to location using that android application when users are going to local or government bus and user can generate amount according to that bus.

Location Selection:

A user has to select from and to location and it will generate fare details for based on that location. Then we have enter the count of passengers and we get total amount. After that, we have to use QR scanner for mobile payment.

Web Service

Web service is like connecting android application and server. Server should run 24 hours and it has to give all the details to database which data's we are getting from users. Then using SOAP protocol we can connect android application to server. If we are using SOAP protocol, it will collect all the details from android application and it will send to server.

Database

Admin can see all the details of users like where they are rode local bus.Then admin has to analyse that details like username, from location,to location, amount for bus fare and admin id.

Classification

We have classified that each and every 3 hours using SVM algorithm.Becuase whenever reaching bus from one place to another place, it has to collect all the details from

users who are all using QR scanner in bus. Then we have analyzed the data like when and where we can give another or extra bus for according to that place.

5.3 SAMPLE CODE

Sample code for QRScan

```
package com.example.smartbus;

import android.net.Uri;

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.ActivityNotFoundException;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.DialogInterface.OnClickListener;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

public class QRScan extends Activity

{

protected static final int ACTIVITY_RESULT_QR_DRDROID = 0;

        TextView report;

        Button scan;

        String pamount;
```

```
        String from,username;
        String to;
        String time,busno,cost,count;
        protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_qrscan);
        report = (TextView) findViewById(R.id.textView2);
        scan = (Button) findViewById(R.id.button1);
        username=getIntent().getStringExtra("username");
         pamount=getIntent().getStringExtra("pamount");
         from=getIntent().getStringExtra("from");
         to=getIntent().getStringExtra("to");
         time=getIntent().getStringExtra("time");
         busno=getIntent().getStringExtra("busno");
         cost=getIntent().getStringExtra("resultamount");
         count=getIntent().getStringExtra("count");
    //   Toast.makeText(getApplicationContext(), pamount, Toast.LENGTH_LONG).show();
     //   scan.setOnClickListener(new View.OnClickListener() {
        public void onClick(View arg0) {
        Intent i = new Intent("la.droid.qr.scan");
try
{
      startActivityForResult(i, ACTIVITY_RESULT_QR_DRDROID);
```

```
}
        catch (ActivityNotFoundException activity)
{
         QRScan.qrDroidRequired(QRScan.this);
}
}
});
}
protected void onActivityResult(int requestCode, int resultCode, Intent data)
 {
      super.onActivityResult(requestCode, resultCode, data);
    if(ACTIVITY_RESULT_QR_DRDROID == requestCode && data != null && data.getExtras() != null )
{
   String result = data.getExtras().getString("la.droid.qr.result");
    report.setText(result);
    report.setVisibility(View.VISIBLE);
    Intent in=new Intent(QRScan.this,Wallet.class);
    in.putExtra("qrvalue", result);
    in.putExtra("amount", pamount);
    in.putExtra("to", to);
    in.putExtra("from", from);
    in.putExtra("busno", busno);
    in.putExtra("time", time);
    in.putExtra("cost", cost);
```

```
   in.putExtra("count",count);
   in.putExtra("username", username);
  startActivity(in);
}
}
protected static void qrDroidRequired(final QRScan qrScan) {
AlertDialog.Builder AlertBox = new AlertDialog.Builder(qrScan);
AlertBox.setMessage("QRDroid Missing");
AlertBox.setPositiveButton("Direct Download", new OnClickListener() {
public void onClick(DialogInterface arg0, int arg1)
 {
qrScan.startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://droid.la/apk/qr/")));
}
});
AlertBox.setNeutralButton("From Market", new OnClickListener()
{
public void onClick(DialogInterface dialog, int which)
{
      qrScan.startActivity(new Intent(
      Intent.ACTION_VIEW, Uri.parse("http://market.android.com/details?id=la.droid.qr")));
}
});
AlertBox.setNegativeButton("Cancel", new OnClickListener() {
```

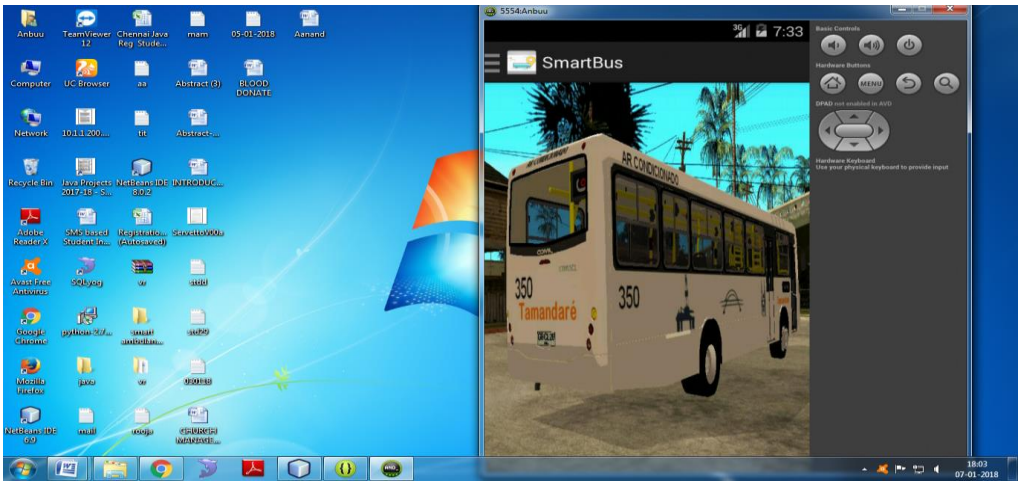
```
public void onClick(DialogInterface dialog, int which) {
dialog.cancel();
}
});
AlertBox.create().show();
}
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.qrscan, menu);
return true;
}}
```

CHAPTER 6

OUTPUT SCREENS

Screen No:1

Screen Title: SmartBus App Interface

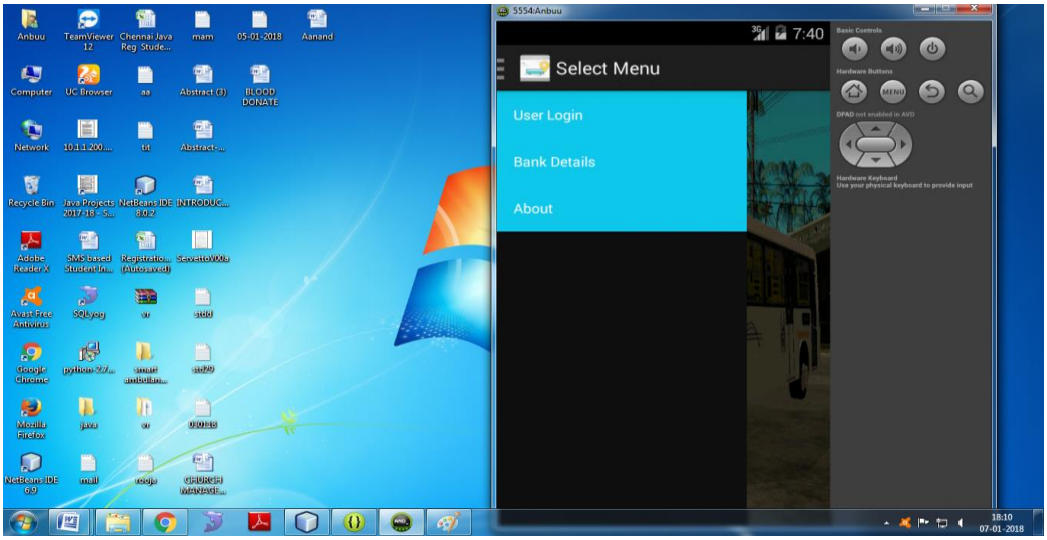


SmartBus App Description:

The users can use this application for travelling at different location.The application contain route details and different users details. SmartBus server maintain the database of different users.

Screen No:2

Screen Title: Select Menu

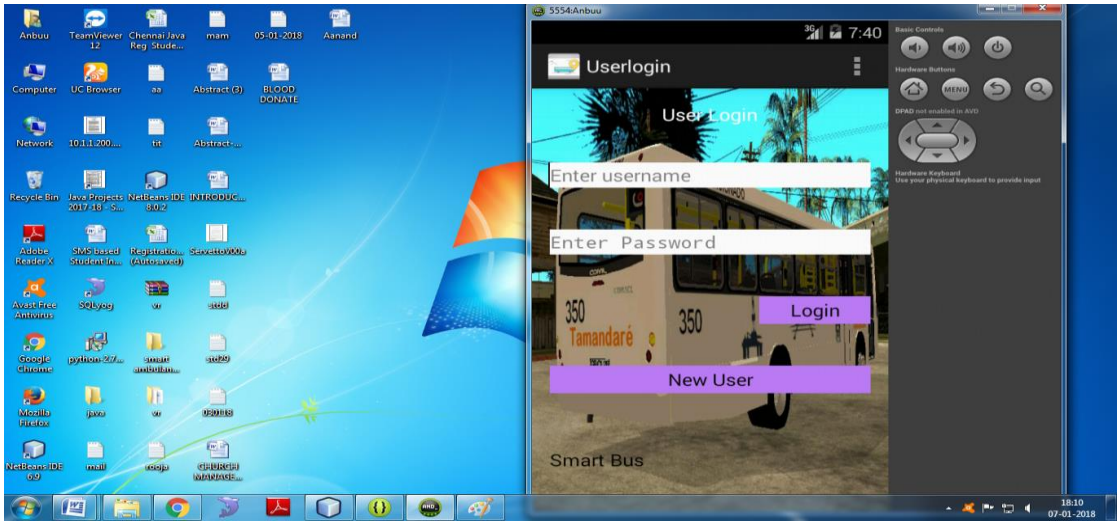


Select Menu Description:

The users has provide with the menu,so the user by selecting menu ,user can login ,add bank details,knows about the app details.

Screen No:3

Screen Title: Userlogin



Userlogin Description:

The user has enter in userlogin option by selecting user login option.The userlogin page contain username,password,login,New User options.Soif the new user using this app ,user by selecting New User option can enter the details.if there is existing user,then user can simply by selecting login user can login to the app by providing username and password details.

CHAPTER 7

SYSTEM TESTING

Testing

The various levels of testing are

1. White Box Testing
2. Black Box Testing
3. Unit Testing
4. Functional Testing
5. Performance Testing
6. Integration Testing
7. Objective
8. Integration Testing
9. Validation Testing
10. System Testing
11. Structure Testing
12. Output Testing
13. User Acceptance Testing

White Box Testing

White-box testing (also known as **clear box testing**, **glass box testing**, **transparent box testing**, and **structural testing**) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code.

These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Levels

1. Unit testing. White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.
2. Integration testing. White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

3. Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

White-box testing's basic procedures involve the understanding of the source code that you are testing at a deep level to be able to test them. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analysed for test cases to be created. These are the three basic steps that white-box testing takes in order to create test cases:

1. Input, involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.
2. Processing Unit, involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
3. Output, prepare final report that encompasses all of the above preparations and results.

Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration,system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well

Test procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

Test cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily *functional* in nature, *non-functional* tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output without any knowledge of the test object's internal structure.

Test design techniques

Typical black-box test design techniques include:

- Decision table testing
- All-pairs testing
- State transition tables
- Equivalence partitioning
- Boundary value analysis

Unit testing

In computer programming, **unit testing** is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process.

Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper)to being formalized as part of build automation.

Testing will not catch every error in the program, since it cannot evaluate every execution path in any but the most trivial programs. The same is true for unit testing.

Additionally, unit testing by definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system-level errors (such as functions performed across multiple units, or non-functional test areas such as performance).

Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a complete absence of errors. In order to guarantee correct behaviour for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behaviour.

Software testing is a combinatorial problem. For example, every Boolean decision statement requires at least two tests: one with an outcome of "true" and one with an outcome of "false". As a result, for every line of code written, programmers often need 3 to 5 lines of test code.

This obviously takes time and its investment may not be worth the effort. There are also many problems that cannot easily be tested at all – for example those that are nondeterministic or involve multiple threads. In addition, code for a unit test is likely to be at least as buggy as the code it is testing. Fred Brooks in The Mythical Man-Month quotes: *never take two chronometers to sea. Always take one or three.* Meaning, if two chronometers contradict, how do you know which one is correct?

Another challenge related to writing the unit tests is the difficulty of setting up realistic and useful tests. It is necessary to create relevant initial conditions so the part of the application being tested behaves like part of the complete system. If these initial conditions are not set correctly, the test will not be exercising the code in a realistic context, which diminishes the value and accuracy of unit test results.

To obtain the intended benefits from unit testing, rigorous discipline is needed throughout the software development process. It is essential to keep careful records not only of the tests that have been performed, but also of all changes that have been made to the source code of this or any other unit in the software. Use of a version control system is essential. If a later version of the unit fails a particular test that it had previously passed, the version-control software can provide a list of the source code changes (if any) that have been applied to the unit since that time.

It is also essential to implement a sustainable process for ensuring that test case failures are reviewed daily and addressed immediately if such a process is not implemented and

ingrained into the team's workflow, the application will evolve out of sync with the unit test suite, increasing false positives and reducing the effectiveness of the test suite.

Unit testing embedded system software presents a unique challenge: Since the software is being developed on a different platform than the one it will eventually run on, you cannot readily run a test program in the actual deployment environment, as is possible with desktop programs.

Functional testing

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes *what* the system does.

Functional testing differs from system testing in that functional testing "*verifies* a program by checking it against ... design document(s) or specification(s)", while system testing "*validate* a program by checking it against the published user or system requirements" (Kane, Falk, Nguyen 1999, p. 52).

Functional testing typically involves five steps.The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications
2. The determination of output based on the function's specifications
3. The execution of the test case
4. The comparison of actual and expected outputs

Performance testing

In software engineering, **performance testing** is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

Testing types

Load testing

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.

Stress testing

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

Soak testing

Soak testing, also known as endurance testing, is usually done to determine if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.

Spike testing

Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behaviour of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

Configuration testing

Rather than testing for performance from the perspective of load, tests are created to determine the effects of configuration changes to the system's components on the system's performance and behaviour. A common example would be experimenting with different methods of load-balancing.

Isolation testing

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

Integration testing

Integration testing (sometimes called **integration and testing**, abbreviated **I&T**) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Some different types of integration testing are big bang, top-down, and bottom-up. Other Integration Patterns are: Collaboration Integration, Backbone Integration, Layer

Integration, Client/Server Integration, Distributed Services Integration and High-frequency Integration.

Big Bang

In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

A type of Big Bang Integration testing is called **Usage Model testing**. Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use.

Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment.

For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

Top-down and Bottom-up

Bottom Up Testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all

or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Top Down Testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

Sandwich Testing is an approach to combine top down testing with bottom up testing.

The main advantage of the Bottom-Up approach is that bugs are more easily found. With Top-Down, it is easier to find a missing branch link

Verification and validation

Verification and Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it full fills its intended purpose. These are critical components of a quality management system such as ISO 9000. The words "verification" and "validation" are sometimes preceded with "Independent" (or IV&V), indicating that the verification and validation is to be performed by a disinterested third party.

It is sometimes said that validation can be expressed by the query "Are you building the right thing?" and verification by "Are you building it right?"In practice, the usage of these terms varies. Sometimes they are even used interchangeably.

The PMBOK guide, an IEEE standard, defines them as follows in its 4th edition

- "**Validation.** The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with *verification*."
- "**Verification**. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with *validation*."
    - Verification is intended to check that a product, service, or system (or portion thereof, or set thereof) meets a set of initial design specifications. In the development phase, verification procedures involve performing special tests to model or simulate a portion, or the entirety, of a product, service or system, then performing a review or analysis of

the modelling results. In the post-development phase, verification procedures involve regularly repeating tests devised specifically to ensure that the product, service, or system continues to meet the initial design requirements, specifications, and regulations as time progresses. It is a process that is used to evaluate whether a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process.

- Validation is intended to check that development and verification procedures for a product, service, or system (or portion thereof, or set thereof) result in a product, service, or system (or portion thereof, or set thereof) that meets initial requirements. For a new development flow or verification flow, validation procedures may involve modelling either flow and using simulations to predict faults or gaps that might lead to invalid or incomplete verification or development of a product, service, or system (or portion thereof, or set thereof). A set of validation requirements, specifications, and regulations may then be used as a basis for qualifying a development flow or verification flow for a product, service, or system (or portion thereof, or set thereof). Additional validation procedures also include those that are designed specifically to ensure that modifications made to an existing qualified development flow or verification flow will have the effect of producing a product, service, or system (or portion thereof, or set thereof) that meets the initial design requirements, specifications, and regulations; these validations help to keep the flow qualified. It is a process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders. This is often an external process.
- It is sometimes said that validation can be expressed by the query "Are you building the right thing?" and verification by "Are you building it right?". "Building the right thing" refers back to the user's needs, while "building it right" checks that the specifications are correctly implemented by the system. In some contexts, it is required to have written requirements for both as well as formal procedures or protocols for determining compliance.

- It is entirely possible that a product passes when verified but fails when validated. This can happen when, say, a product is built as per the specifications but the specifications themselves fail to address the user's needs.

Activities

Verification of machinery and equipment usually consists of design qualification (DQ), installation qualification (IQ), operational qualification (OQ), and performance qualification (PQ). DQ is usually a vendor's job. However, DQ can also be performed by the user, by confirming through review and testing that the equipment meets the written acquisition specification. If the relevant document or manuals of machinery/equipment are provided by vendors, the later 3Q needs to be thoroughly performed by the users who work in an industrial regulatory environment. Otherwise, the process of IQ, OQ and PQ is the task of validation. The typical example of such a case could be the loss or absence of vendor's documentation for legacy equipment or do-it-yourself (DIY) assemblies (e.g., cars, computers etc.) and, therefore, users should endeavour to acquire DQ document beforehand. Each template of DQ, IQ, OQ and PQ usually can be found on the internet respectively, whereas the DIY qualifications of machinery/equipment can be assisted either by the vendor's training course materials and tutorials, or by the published guidance books, such as *step-by-step* series if the acquisition of machinery/equipment is not bundled with on- site qualification services. This kind of the DIY approach is also applicable to the qualifications of software, computer operating systems and a manufacturing process. The most important and critical task as the last step of the activity is to generating and archiving machinery/equipment qualification reports for auditing purposes, if regulatory compliances are mandatory.

Qualification of machinery/equipment is venue dependent, in particular items that are shock sensitive and require balancing or calibration, and re-qualification needs to be conducted once the objects are relocated. The full scales of some equipment qualifications are even time dependent as consumables are used up (i.e. filters) or springs stretch out, requiring recalibration, and hence re-certification is necessary when a specified due time lapse Re-qualification of machinery/equipment should also be conducted when replacement of parts, or coupling with another device, or installing a new application software and restructuring of the computer which affects especially the pre-settings, such as on BIOS, registry, disk drive partition table, dynamically-linked (shared) libraries, or an ini file etc., have been necessary. In such a situation, the specifications of the

parts/devices/software and restructuring proposals should be appended to the qualification document whether the parts/devices/software are genuine or not.

Torres and Hyman have discussed the suitability of non-genuine parts for clinical use and provided guidelines for equipment users to select appropriate substitutes which are capable to avoid adverse effects. In the case when genuine parts/devices/software are demanded by some of regulatory requirements, then re-qualification does not need to be conducted on the non-genuine assemblies. Instead, the asset has to be recycled for non-regulatory purposes.

When machinery/equipment qualification is conducted by a standard endorsed third party such as by an ISO standard accredited company for a particular division, the process is called certification. Currently, the coverage of ISO/IEC 15408 certification by an ISO/IEC 27001 accredited organization is limited; the scheme requires a fair amount of efforts to get popularized.

System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification

Types of tests to include in system testing

The following examples are different types of testing that should be considered during System testing:

- Graphical user interface testing
- Usability testing
- Software performance testing
- Compatibility testing
- Exception handling
- Load testing
- Volume testing
- Stress testing
- Security testing
- Scalability testing
- Sanity testing
- Smoke testing
- Exploratory testing
- Ad hoc testing
- Regression testing
- Installation testing
- Maintenance testing Recovery testing and failover testing.
- Accessibility testing, including compliance with:
    - Americans with Disabilities Act of 1990
    - Section 508 Amendment to the Rehabilitation Act of 1973
    - Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

Although different testing organizations may prescribe different tests as part of System testing, this list serves as a general framework or foundation to begin with.

Structure Testing:

It is concerned with exercising the internal logic of a program and traversing particular execution paths.

Output Testing:

- Output of test cases compared with the expected results created during design of test cases.
- Asking the user about the format required by them tests the output generated or displayed by the system under consideration.
- Here, the output format is considered into two was, one is on screen and another one is printed format.
- The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.
- The output comes out as the specified requirements as the user's hard copy.

User acceptance Testing:

- Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.
- The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.
- It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document.

Two set of acceptance test to be run:

1. Those developed by quality assurance group.
2. Those developed by customer.

CHAPTER 8

CONCLUSION

The paper summarizes the current issues in bus ticketing system. To overcome from this we are working towards android platform. We have identified the current gaps and open research areas. Our research will focus on these open problems and propose effective solutions for the same. This paper introduces on how to secure passenger information. To overcome the drawbacks of manual ticketing system we are using QR-Code for security purpose of passengers information in the propose system

FUTURE ENHANCEMENT

When it comes to taking the public transportation, time and patience are of essence. In other words, many people using public transport buses have experienced time loss because of waiting at the bus stops. In this paper, we proposed smart bus tracking system that any passenger with a smart phone or mobile device with the QR (Quick Response) code reader can scan QR codes placed at bus stops to view estimated bus arrival times, buses' current locations, and bus routes on a map. Anyone can access these maps and have the option to sign up to receive free alerts about expected bus arrival times for the interested buses and related routes via SMS and e-mails. We used algorithm for the estimation of bus arrival times to minimize the passengers waiting time. GPS (Global Positioning System) and Google Maps are used for navigation and display services, respectively.

CHAPTER 9

BIBLOGRAPHY

[1].Z. Wei, Y. Song, H. Liu, Y. Sheng, X. Wang, "The research and implementation of GPS intelligent transmission strategy based on on-board Android smartphones", Computer Science and Network Technology (ICCSNT) 2013 3rd International Conference on, pp. 1230-1233, 2013.

[2].J. Gong, M. Liu, S. Zhang, "Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time GPS data", 2013 25th Chinese Control and Decision Conference (CCDC), pp. 972-976, 2013.

[3].Saurabh Chatterjee, Prof. Balram Timande, "Public Transport System Ticketing system using RFID and ARM processor Perspective Mumbai bus facility B.E.S.T", International Journal of Electronics and Computer Science Engineering.

[4].Md. Foisal Mahedi Hasan, Golam Tangim, Md. Kafiul Islam, Md. Rezwanul Haque Khandokar, Arif Ul Alam, "RFID-based Ticketing for Public Transport System: Perspective Megacity Dhaka".

[5].M. Bhuvaneswari, S. Sukhumar, N. Divya, S. Kalpanadevi, N. SuthanthiraVanitha, "Embedded System Based Automatic Ticket Vending Machine for Modern Transport System", International Journal of Advanced Research in Computer and Communication Engineering, November 2013.

[6]. Pankaj Verma, J.S Bhatia, "Design and Development of GPS GSM Based Tracking System with Google Map Based Monitoring", International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.3, No.3, June 2013.

[7].R. Ramani, S. Valarmathy, Dr. N. SuthanthiraVanitha, S. Selvaraju, M. Thiruppathi, R. Thangam, " Vehicle Tracking and Locking System Based on GSM and GPS ", MECS I.J. Intelligent Systems and Applications, 2013, 09

[8].B. Danila, Y. Yu, J. K. Marsh, K. Bassler, "Optimal transport on complex nets", Phys. Rev. E, vol. 74, pp. 046106–6, October 2006.

[9].R. Hua-Ling, "Origin-Destination Demands Estimation in Congested Dynamic Transit Networks", International Conference on Management Science & Engineering (14th), 2007.