METRO BUS TRACKING SYSTEM

A PROJECT REPORT

in the partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

MAY 2019

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO**

**LIST OF FIGURES ii**

**LIST OF ABBREVATIONS iii**

1. **INTRODUCTION**
   1. About the Project 08
2. **SYSTEM ANALYSIS** 
   1. Existing system 09
   2. Proposed system 10

**3 REQUIREMENTS SPECIFICATION**

**3.1** Introduction 11

* 1. Hardware and Software specification 11
  2. Technologies Used 12

**3.3.1** Java 12

**3.3.1.1** Introduction to Java 12

**3.3.1.2** Working of Java 14

**3.3.2** Apache Tomcat Server 18

**3.3.3** Android 20

**4 Project Purpose and Scope**

**4.1** Purpose 24

**4.2** Project Scope 25

**4.3** Product Perspective 26

**4.4** SystemFeatures 26

**4.5** Design and Implementation Constraints 27

**4.6** Other Nonfunctional Requirements 28

**5 SYSTEM DESIGN**

**5.1** System Design

**6 SYSTEM DESIGN – DETAILED**

**6.1** Modules 37

**6.2** Module explanation 37

**7 CODING AND TESTING**

**6.1** Coding 40

**6.2** Coding standards 31

**6.3** Test procedure 42

**6.4** Test data and output 43

**SOURCE CODE** 52

**SNAP SHOTS** 72

**REFERENCES** 78

**LIST OF FIGURES**

5.1 System Design

5.2 Sequence Diagram

5.3 Use Case Diagram

5.4 Activity Diagram

5.5 Collaboration Diagram:

5.6 Data Flow Diagram

5.7 Class Diagram

**LIST OF ABBREVATIONS**

JDK Java Development Toolkit

DEX Dalvik Executables

TCP Transmission Control Protocol

IP Internet Protocol

**HTTP** Hyper Text Transfer Protocol

**ADT** Android Development Tool

**APK** Android Package Kit

**SDK** Software Development Kit

**API** Application Program Interface

**JSON** JavaScript Object Notation

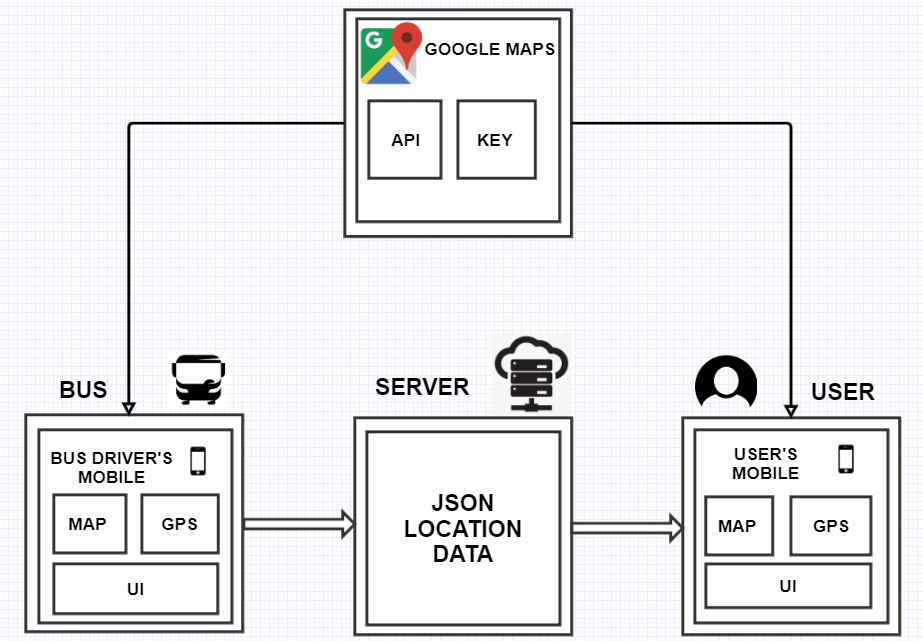
**APP** Application

**GPS** Global Positioning System

**UI** User Interface

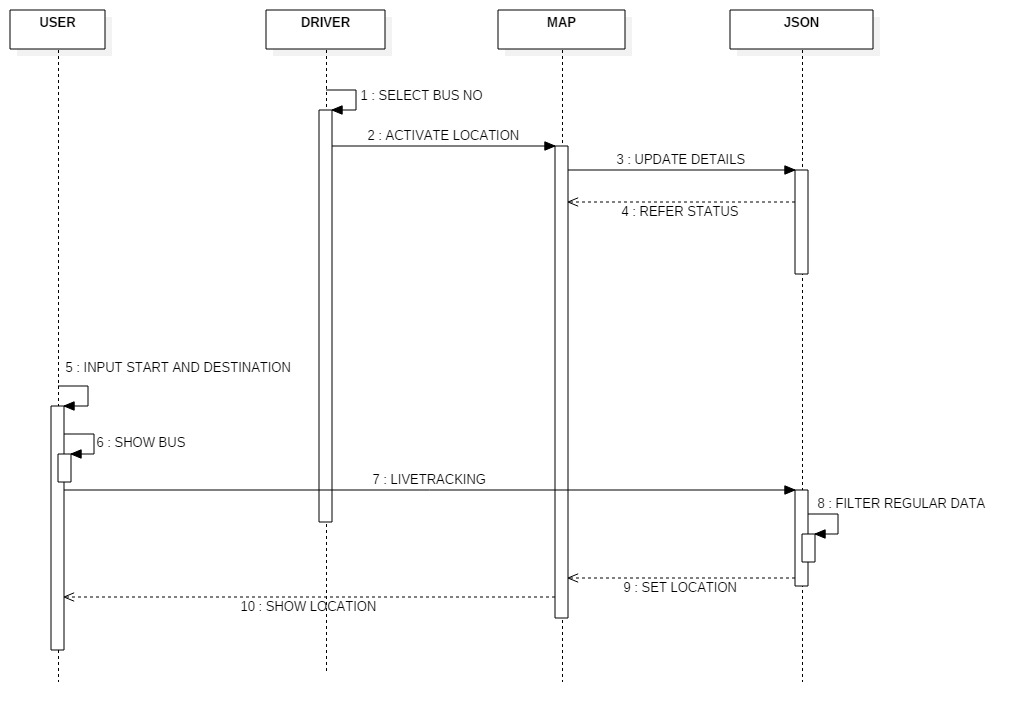
**CHAPTER 5**

**5.1 Architecture Diagram:**

****

**5.2 Sequence Diagram:**

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

****

**5.3 Use Case Diagram:**

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

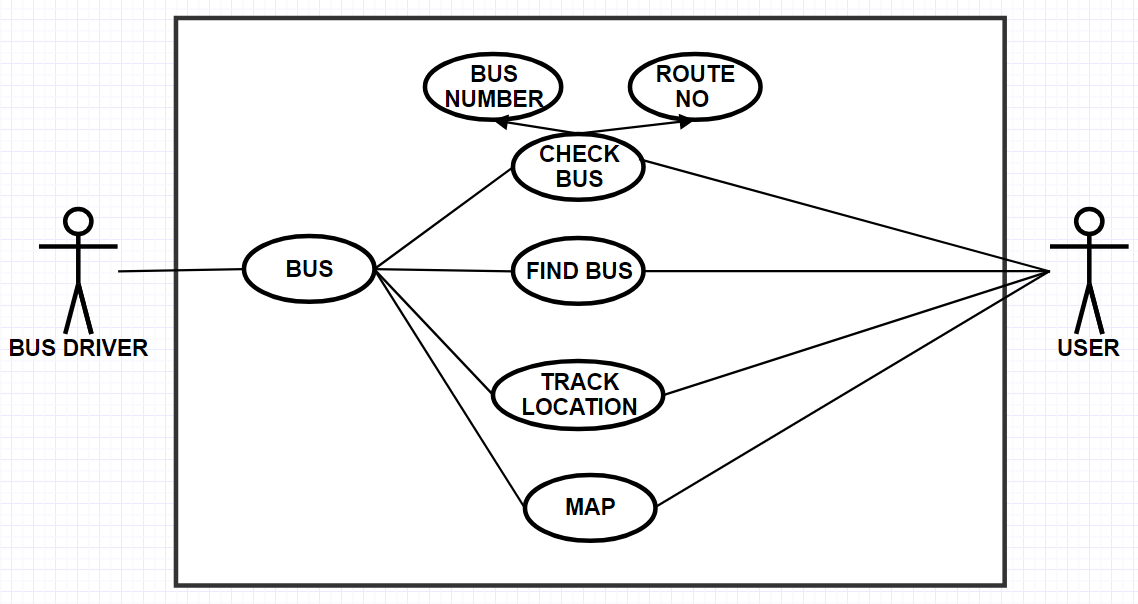
**5.3.1. USECASE DIAGRAM**

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.

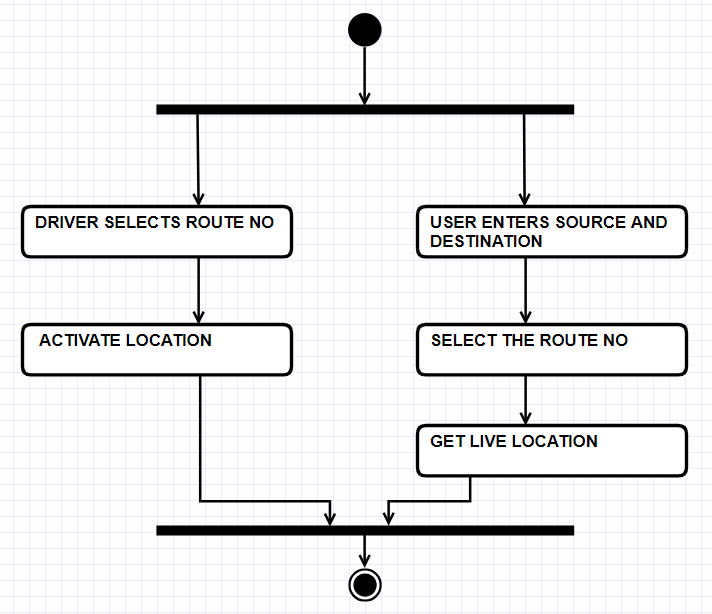
****

**5.4 Activity Diagram:**

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

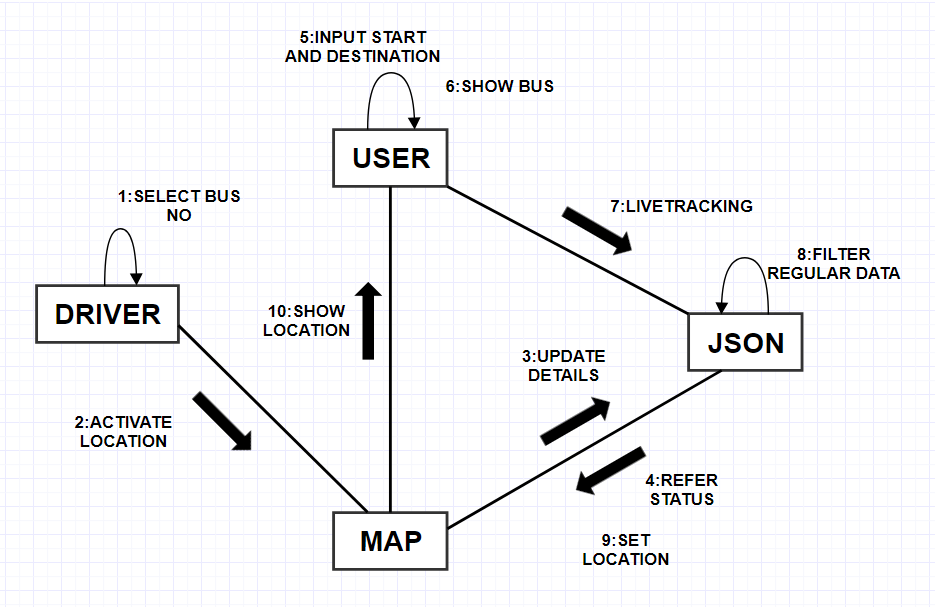
The most important shape types:

* Rounded rectangles represent activities.
* Diamonds represent decisions.
* Bars represent the start or end of concurrent activities.
* A black circle represents the start of the workflow.
* An encircled circle represents the end of the workflow.

****

**5.5 Collaboration Diagram:**

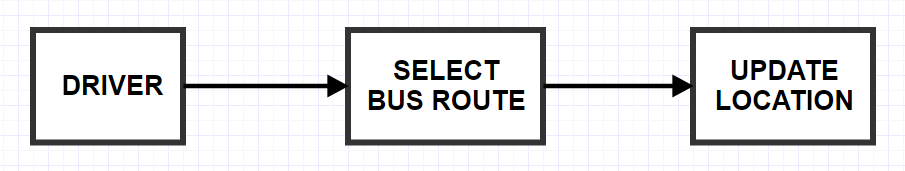
UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

****

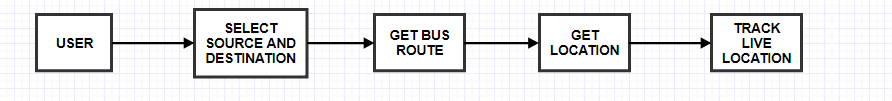
**5.6 DATA FLOW DIAGRAM:**

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

**Level 0:**

****

**Level 1:**



**CHAPTER 7**

**CODING AND TESTING**

**7.1 CODING**

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

**7.2 CODING STANDARDS**

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

**7.2.1 NAMING CONVENTIONS**

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

**Class names**

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

**Member Function and Data Member name**

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

7**.2.2 VALUE CONVENTIONS**

Value conventions ensure values for variable at any point of time. This involves the following:

* Proper default values for the variables.
* Proper validation of values in the field.
* Proper documentation of flag values.

**7.2.3 SCRIPT WRITING AND COMMENTING STANDARD**

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

**7.2.4 MESSAGE BOX FORMAT**

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

* X – User has performed illegal operation.
* ! – Information to the user.

**7.3 TEST PROCEDURE**

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

* Static analysis is used to investigate the structural properties of the Source code.
* Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

**7.4 TEST DATA AND OUTPUT**

**7.4.1 UNIT TESTING**

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

**7.4.2 FUNCTIONAL TESTS**

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

* Performance Test
* Stress Test
* Structure Test

**7.4.3 PERFORMANCE TEST**

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

**7.4.4 STRESS TEST**

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

**7.4.5 STRUCTURED TEST**

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

* Exercise all logical decisions on their true or false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structures to assure their validity.
* Checking attributes for their correctness.
* Handling end of file condition, I/O errors, buffer problems and textual errors in output information

**7.4.6 INTEGRATION TESTING**

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

**7.5 TESTING TECHNIQUES / TESTING STRATERGIES**

**7.5.1 TESTING**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

**7.5.1.1 WHITE BOX TESTING**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* Flow graph notation
* Cyclometric complexity
* Deriving test cases
* Graph matrices Control

**7.5.1.2 BLACK BOX TESTING**

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**7.5.2 SOFTWARE TESTING STRATEGIES:**

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

**7.5.2.1 INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

**7.5.2.2 PROGRAM TESTING:**

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

**7.5.2.3 SECURITY TESTING:**

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

**7.5.2.4 VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

\* The function or performance characteristics confirm to specifications and are accepted.

\* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

**7.5.2.5 USER ACCEPTANCE TESTING**

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

* Input screen design.
* Output screen design.

**Screenshots:**



