TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	4
1	**INTRODUCTION**	5
	1.1 Overview	
	1.2 About the project	
2	**SYSTEM ANALYSIS**	6
	2.1 Benefits	
	2.1.1 Disadvantages	
3	**SYSTEM SPECIFICATIONS**	7
	3.1 Hardware specification	
	3.2 Software requirements	
	3.3 Mobile requirements	
4	**SOFTWARE DESCRIPTION**	8
	4.1 Front end: Eclipse	
	4.2 Back end: SQLite	
	4.2.1 SQLite features	
5	**PROJECT DESCRIPTION**	11
	5.1 Modules description	
	5.1.1 Admin login	
	5.1.2 User login	
	5.2 System design	
	5.2.1 File design	
	5.2.2 Input design	
	5.2.3 Output design	
	5.2.4 Database design	
	5.2.5 Dataflow diagram	
	5.2.6 Sequence diagram	

6	**SYSTEM TESTING**	**17**
	6.1 Unit testing	
	6.2 Function testing	
	6.3 Integer testing	
	6.4 Validation testing	
	6.5Output testing	
7	**SYSTEM IMPLEMENTTION**	**20**
	7.1 Sample source code	
	7.2 Sample screen shots	

LIST OF ABBREVIATIONS

TERMS	ABBREVIATIONS
CLP	CONSTRAINT LOGIC PROGRAMMING
GUI	GRAPHICAL USER INTERFACE
CHR	CONSTRAINT HANDLING RULES
ISO	INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
PROLOG	PROGRAMMATION EN LOGIQUE
TCL	TOOL COMMAND LANGUAGE
TK	TOOL KIT
ACID	ATOMIC CONSISTENT ISOLATED AND DURABLE
JSON	JAVASCRIPT OBJECT NOTATION
ANSI	AMERICAN NATIONAL STANDARD INSTUITE
BSD	BERKELEY SOFTWARE DISTRIBUTION
XML	EXTENSIBLE MARKUP LANGUAGE
CSV	COMMA SEPARATED VALUES

ABSTRACT

“Voice-based Transport Enquiry System” is a system which works based on the voice input given by the user who intends to know about transportation facilities available from a certain terminal. There is no communication which is understood more properly than voice. This system too uses voice commands to provide input and gives the required information to the user in the form of audio output. Users can get all types of transportation details between any two places. This system will have an additional feature of getting the traffic status of a particular place. The user can get a live video feed of the traffic of that place. As the whole, this system can be operated by speech or voice so, making it easier to use for the blind people.

CHAPTER-1
INTRODUCTION

1.1 OVERVIEW

We have experienced in waiting to a transport terminals for transport controllers to get the information about the transport facility. We encounter so many times there will be no person for providing these information which significantly wastes the time just to know whether there is any facility or not. Here is one solution for such a problem which lessens the human intervention in providing such information in the transport terminal Voice Based Automated Transport Enquiry System is the enquiry system which operates based on the voice input given by the user. There is no communication which is understood more appropriately than voice. This system too uses the voice commands and gives the required information in the form of voice. This system is can be installed in any transport terminal like Bus stands, Railway terminals or airports.

1.2 ABOUT THE PROJECT

Voice Based Automated Transport Enquiry System is developed for providing the information for the enquiry in transport terminals. This project is developed using .Net technology using c# Programming language. This uses sql server for storing the information to be provided to the user. This user Microsoft Speech recognition to detect the voice from the user and uses the speech control to deliver the voice output. This also displays the results on the screen for further verification

CHAPTER 2
SYSTEM ANALYSIS

2.1 BENEFITS

- It works in more interactive way in the form of speech. It needs less or no human intervention.
- It is automated.
- It needs very less maintenance.

2.1.1 DISADVANTAGES

- Heavy noise is a very crowded place can disturb the result.

CHAPTER 3
SYSTEM SPECIFICATION

SYSTEM REQUIREMENTS:

A set of system services and constraints in detail, The System requirements are the more detailed specification of the User Requirements it sometimes serves as a contract between the user and the developer

3.1 HARDWARE SPECIFICATION

System	: IBM-Compatible PC
Processor	: Pentium
Memory	: 4 GB RAM
Hard Disk Drive	: 40 GB
Versions	: 7.0+

3.2 SOFTWARE REQUIREMENTS

Operating System	: Windows 7/8/10
Environment	: Eclipse and Android Studio
Programming language	: JAVA
Database Access Layer	: SQLite

3.3 MOBILE REQUIREMENT

Platform	: Android,
RAM	: 1GB
Memory space	: 200MB

CHAPTER 4
SOFTWARE DESCRIPTION

The selection of appropriate software plays a major important role in the successful implementation of the system. The features of the software should be in part with the requirement of the proposed system. To present such complex system to the idea entry operator, who is basically little knowledge, we need to very effective interface, which make the system user friendly to the optimum can extend. For this, very reason **ECLIPSE** is selected as the application development software, **ECLIPSE** as font end and **SQLITE** as back end.

4.1 PROGRAMMING LANGUAGE: ECLIPSE

Overview

Eclipse is a **Constraint Logic Programming** (CLP) system, consisting of

- a collection of libraries,
- a modelling and control language,
- a development environment,
- interfaces for embedding into host environments,
- Interfaces to third-party solvers.

Eclipse is intended for

- General programming tasks, especially rapid prototyping.
- Problem solving using the available solver libraries and the CLP paradigm.
- Development of new constraint solvers based on the existing solvers and employing ECLIPSE lower-level language features.

The textbook Constraint Logic Programming using by Apt Wallace covers many of these aspects. For further reading, see the introductory paper Eclipse as a Platform for Constraint Logic Programming and the tutorial-style document ECLIPSE - An Introduction. ECLIPSE has also been discussed in a BYTE article about CLP.

Development Environment

Eclipse development GUI on both UNIX and Windows (screen shot (31k)). Traditional command-line based development system also supported. Debugger working on compiled code. Profiling tool is a tool to collect the timings of the execution of an individual predicates. Detailed documentation in printable form, and online-accessible in html format and from the development system.

System Architecture

Incremental compiler Optimizes index selection, unification order, in lining of control constructs and takes mode information into account. The system is designed to impose no unnecessary limits on programs and data. E.g., there is no limit, other than the available memory, on the number or length of atoms and strings, the arity of factors, the code size, and number of procedures, complexity of clauses or stack sizes. Synchronous and asynchronous event handling, used for both error and exception handling as well as for interfacing to external event sources. Fully automatic memory management with garbage collection for stacks and dictionary. Highly configurable syntax and behaviour. ECLIPSE programs can execute in or-parallel on shared-memory multiprocessor hardware (this functionality is currently not actively maintained because of other priorities).A complete I/O system based on the notion of abstract streams which can be connected to files, tots, pipes, sockets, or in-memory strings and queues.

Core Language Features

Eclipse provides comprehensive facilities to implement data-driven control behaviour. These include declarative delay-clauses as well as primitives for meta-programmed control like explicit goal suspension, flexible triggering facilities and execution priorities. The attributed variable data type is the key to many extensions to the basic Logic Programming language. The system calls user-definable event handlers when it encounters attributed variables in certain contexts, e.g. unification. Full module system, controlling the scope of visibility of predicates, non-logical stores, macros and syntax settings. Modules can be used both to structure applications and to restrict access to the implementation details. String data type with garbage-collectable strings. Arithmetic's with unlimited precision integers, rational numbers, single and double precision floats, and floating-point intervals with safe rounding. Logical iteration construct which often eliminates the need for recursive predicates. Logical arrays. Structures with field names. Matching clauses. Global **references** and **variables**.

Libraries

Constraint solvers. Eclipse provides several libraries of constraint solvers which can be used in application programs:

- arithmetic constraints over finite atomic domains (CHIP compatible)
- finite set constraints
- linear rational constraints

- Propria (generalized propagation)
- Interval reasoning over non-linear constraints
- Interface to external simplex solvers
- Repair-based search
- Constraint Handling Rules CHR. CHR itself contains a library of over 20 further constraints solvers.

Language Compatibility

The Eclipse programming language belongs to the Prolog family of languages. This family includes the ISO Standard ISO/IEC 13211-1 (1995/2007/2012) and a number of other popular and influential implementation dialects. Where possible, Eclipse follows the standard and/or common practice. Where conflicts exist, compatibility modules are provided. These make it possible to integrate components written in different dialects into a single Eclipse application. Fully supported dialects are ISO and C-Prolog, partial support exists for Quintus, Scotus and SWI.

Embedding and Interfaces

Tight interface to C and C++. Eclipse code can call C/C++ code, or a C/C++ main program can call Eclipse as a library. Data can be converted between C and Eclipse representations, or alternatively, Eclipse data can be referenced from C/C++ and C/C++ data can be referenced from Eclipse.

Loose interface for embedding or remote connection to Tcl/Tk and Java host applications. The main characteristics of these interfaces are bi-directional event-driven communication and the ability to communicate complex data structures through a language-independent data representation.

4.2 DATABASE ACCESS LAYER: SQLITE

4.2.1 SQLite Features:

- Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- Full-featured SQL implementation with advanced capabilities like partial indexes, indexes on expressions, JSON, common table expressions, and window functions. (Omitted features)
- A complete database is stored in a single cross-platform disk file. Great for use as an application file format.

CHAPTER 5
PROJECT DESCRIPTION

5.1 MODULES DESCRIPTION

5.1.1 Admin login

5.1.2 User login

5.1.1 ADMIN LOGIN

Using admin's credential login ID and password, the authorized person can access admin controls. Admin can to control the bus details and maintain the database and also give permission to the user. Admin will able to use the voice system which has implemented. Admin can able create and delete the users.

5.1.2 USER LOGIN

User can register their information and login using User's login ID and password they can able to access their information and details. Once user can logged in they can view the details about the transportation details and these steps are able to control by voice commands also. User can delete their information when the need.

5.2 SYSTEM DESIGN

5.2.1 FILE DESIGN

System design is the process of planning a new system to complement or al together replace the old system. The purpose of the design phase is the first step in moving from the problem domain to the solution domain. The design of the system is the critical accept that affects the quality of the software. System design is also called top –level design. The design phase translates the logical aspects of the system into physical aspects of the system**.**

5.2.2 INPUT DESIGN

Input design is the process of converting the user-oriented. Input to a computer-based Format. The goal of the input design is to make the data entry easier, logical and free error. Error in the input data is controlled by the input design. The quality of the Input determines the quality of the system output.

The entire data entry screen is interactive in nature, so that the user can directly enter into data according to the prompted messages. The users are also can directly enter into data according to the prompted messages. The users are also provided with option of selecting an appropriate input from a list of values. This will reduce the number of errors, which are otherwise likely to arise if they were to be entered by user itself.

5.2.3 OUTPUT DESIGN

Output design is very important concept in the computerized system, without reliable output the user may feel the entire system is unnecessary and avoids using it. The proper output design is important in any system and facilitates effective decision-making. The output of this system includes various reports.

Computer output is the most important and direct source of information the user. Efficient, intelligible output designing should improve the system's relationships with the user and help in decision making. A major form of output is the hardcopy from the printer. Output requirements are designed during system analysis. A good starting point for the output design is the data flow diagram. Human factors reduce issues for designing involved addressing internals.

5.2.4 DATABASE DESIGN

A database should provide integration, Integrity and a database independence table in a database contains information pertaining to a specific entity. To maintain the tables in an effective way, it should be normalized to ensure that the number of tables does not exceed the optimum level unless it is mandatory.

To prevent unauthorized access, security measures have been provided. This may prevent unauthorized persons using data that is private. The normalization techniques have been used to design the table such that the use of all the tables is easy.

The various relations between different tables. The number of fields in each table and the type. Width of each field were analyses. The name of the fields and tables where so chosen that the users would not face any problem in identifying the table structure.

Table name: admin_Details

Primary Key: id

Column name	Data type	Width	Description
id	Integer	20	Admin id
name	Varchar	20	Admin name
Location	Varchar	80	Location
Mail id	Varchar	30	Admin mail_id
Ph no	Integer	10	Admin contact
pass	Varchar	10	password

Table name: user_Details

Primary key: id

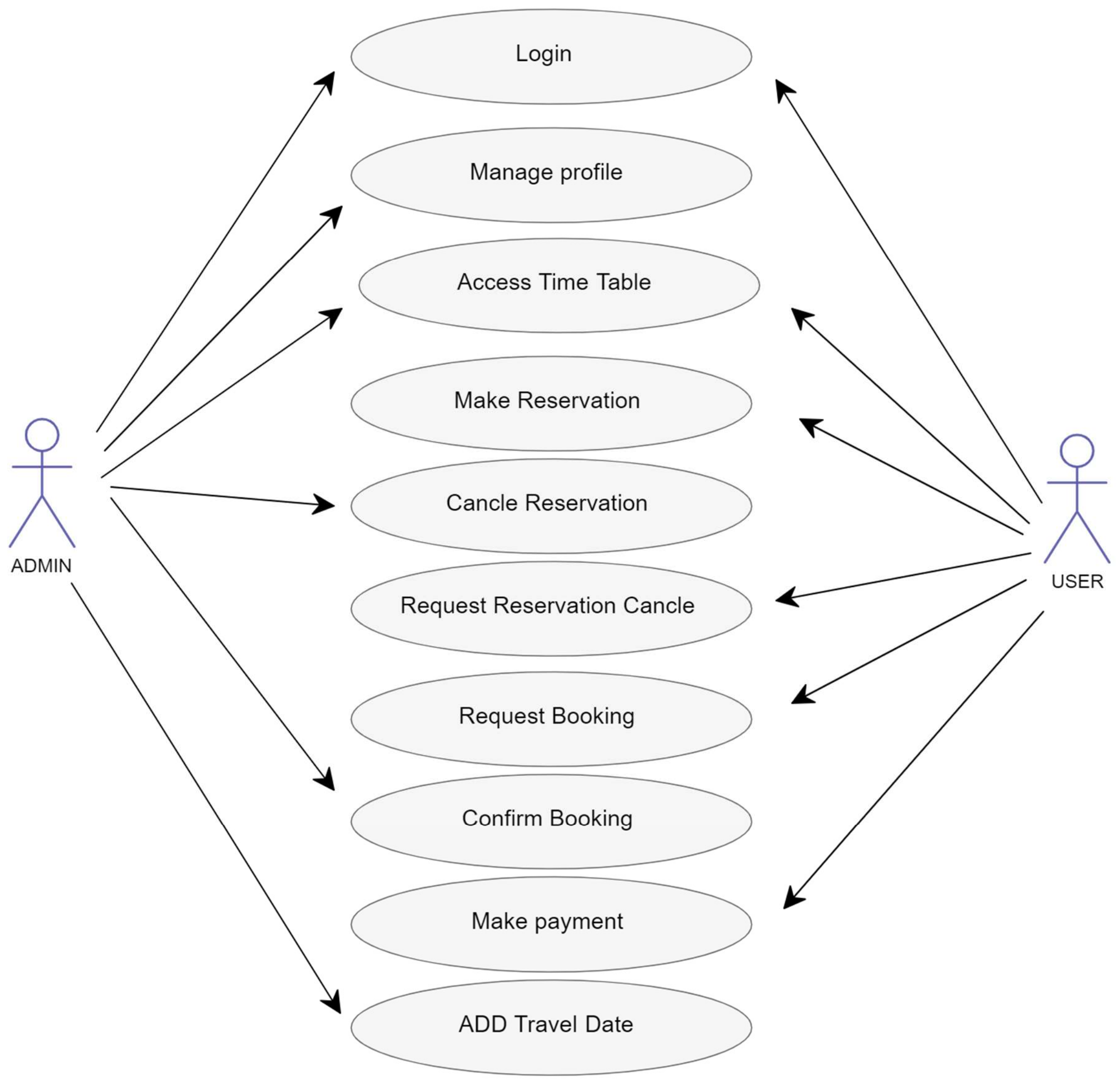
Column name	Data type	Width	Description
id	Integer	20	User id
name	Varchar	20	User name
Dob	Char	15	Date of birth
Gender	Varchar	10	Gender
Address	Varchar	80	Address
Phno	Integer	10	Phone number

Table name: Bus_Details

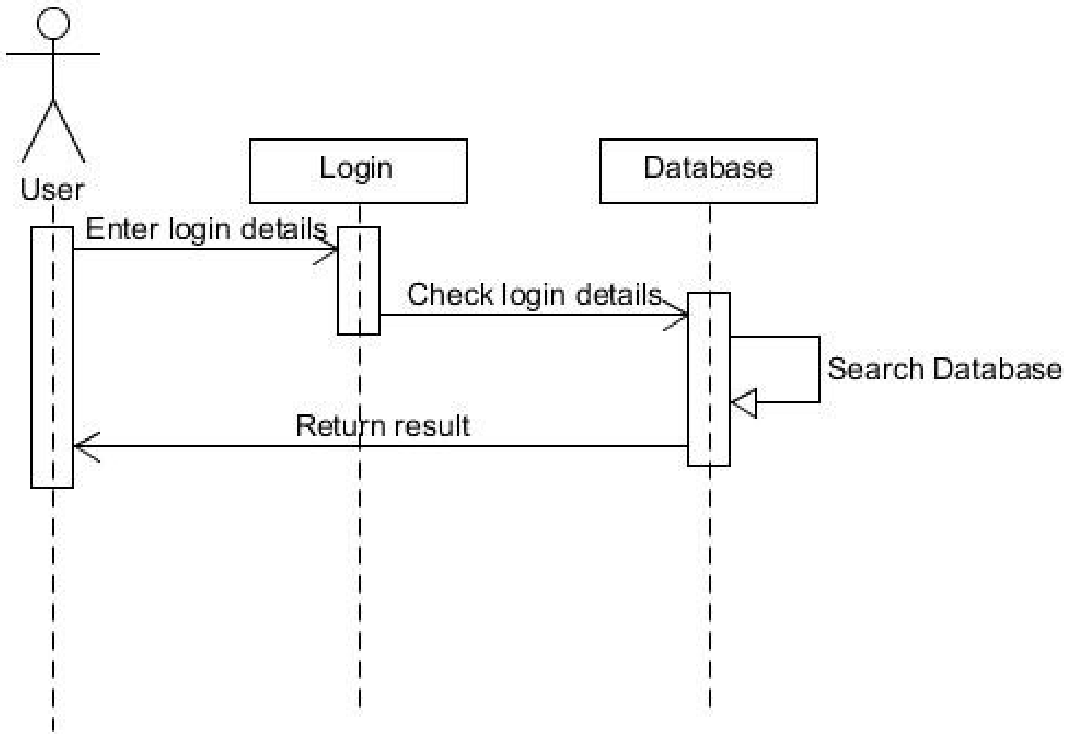
Primary key: bus_id

Column name	Data type	Width	Description
Bus_id	Integer	20	Bus id
Bus_name	Varchar	20	Bus name
Type	Char	15	Type of bus
Source	Varchar	10	Source address
Time	Varchar	80	Time od departure
destination	varchar	10	Destination address
fare	Integer	10	Cost of travell

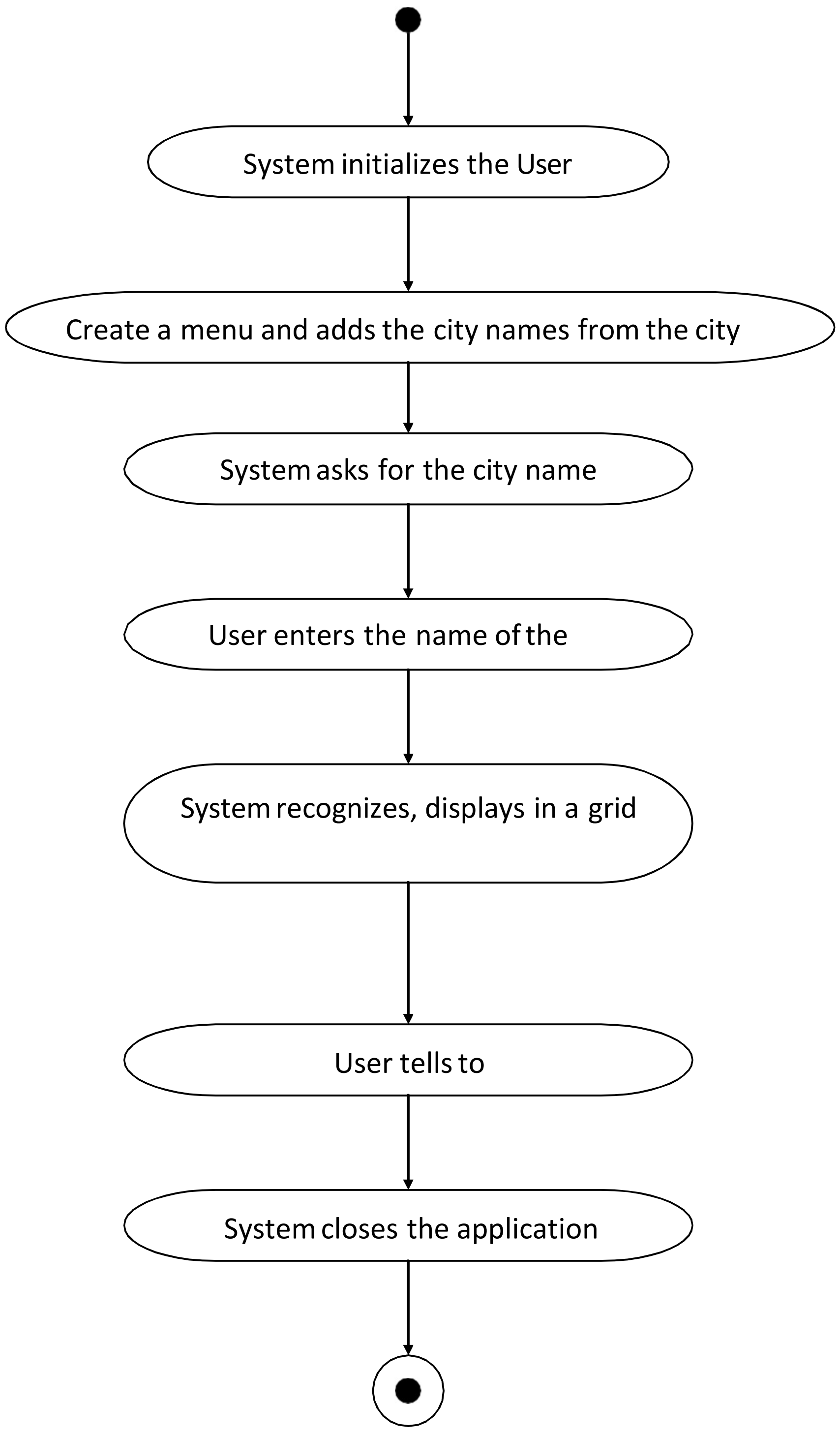
5.2.5 DATA FLOW DIAGRAM



5.2.6 SEQUENCE DIAGRAM



5.2.6 SEQUENCE DIAGRAM


System initializes the User
Create a menu and adds the city names from the city
System asks for the city name
User enters the name of the
System recognizes, displays in a grid
User tells to
System closes the application

CHAPTER 6
SYSTEM TESTING

TESTING

Testing is a technique to establish in an experimental way the reliability and robustness of the software. Testing is an important process, which leads to the success of the system. System testing is mainly performed in the intention of finding error to give the client an error free system. System testing makes a logical assumption that all parts of the system are correct and move towards it to make the system error free. When validating system, number of aspects plays role. First it must be determined whether the software satisfies the original requirements and global set by user, as specified during analysis. Secondly it must be established whether the system meets specification laid down in the design document.

Testing is a series of different tests that whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all work should verify that all system element have been properly integrated and performed allocated function. Testing is the process of checking whether the developed system works according to the actual requirement and objectives of the system. The philosophy behind testing is to find the error. A good test is one that has a high probability of finding an undiscovered error.

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives or met and the user requirements are satisfied. The ultimate aim is quality assurance. Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies a test plan is carried out on each module.

TYPES OF TESTING

- Unit testing
- Function testing
- Integration testing
- Validation testing
- Output testing

6.1 UNIT TESTING

The unit test result are recorded for further references. During unit the functions of the program unit a validation and the limitation are tested. Unit testing changes made in a existing or new program this is carried out during the programming and each module is found to be working satisfactorily. For example in the registration form after entering all the fields we click the submit button.

Unit testing focuses verification effects on the smallest unit of software design i.e, the module. Unit testing is considered an equivalent to the coding step. After the source level code has been developed, reviewed, and verified for correct syntax, unit test case design begins. Test-driven development requires that developers first write falling unit tests. Then they write code and refactor the application until the test passes.

Unit testing involves only those characteristics that are vital to the performance of the unit under test. This encourages developers to modify the source code without immediate concerns about how such changes might affect the functioning of other units or the program as a whole. Once all of the units in a program have been found to be working in the most efficient and error-free manner possible, larger components of the program can be evaluated by means of integration setting.

6.2 FUNCTION TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals. Functional testing is centered on the following items

Valid Input	: identified classes of valid input must be accepted
Invalid Input	: identified classes of invalid input must be rejected
Functions	: identified functions must be exercised
Output	: identified classes of application output must be exercised
System/ procedures	: interfacing systems or procedures must be invoked

6.3 INTERGRATION TESTING

The entire project was split into small program; each of these single programs gives a frame as an output. These programs were tested individually; at last these programs where combined together by creating another program where all these constructors were used. It give a lot of problem by not functioning is an integrated manner.

6.4 VALIDATION TESTING

At the culmination of the black box testing software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of test i.e., Validation succeeds when the software function in a manner that can be reasonably accepted by the customers.

Validation refers to a different set of activities that ensure that the software. That has been built is traceable those customer requirements. Validation testing provides the final assurance software meets all functional, behavioural and performance requirements it checks the quality of the software in both simulated and live environments in simulated approach the developers test the product on their workplace to make the products meet its requirements. In a live environment phase, the product is given to the customer to evaluate the product's functionality. Validation refers to the different set of activities that ensure that software correctly implements a specific function and the software that is built is traceable to customer requirements.

Software validation is achieved through a series of black-box test that demonstrate the conformity with requirement. After each validation check a test has been conducted to check if the test cases pass or fail. The function or performance characteristics conform to the specifications expected. A deviation from specification is uncovered and a deficiency list is created as illustrated in table.

6.5 OUTPUT TESTING

After performing the validation testing the next step is output testing of the proposed system. Since the system cannot be useful if it does not produce the required output. Asking the user about the format in which the system is required tests the output displayed or generated by the system under consideration. Here the output format is considered in two ways, one is on screen and another one is printed format. The output format on the screen is found to be corrected as the format was designed in the system phase according to the user needs.

CHAPTER 7
SYSTEM IMPLEMENTATION

This system implementation is stage in the project where the theoretical design is turned into the working system. The most crucial stage is giving the users confidence that the new system will work effectively and efficiently. The performance of reliability of the system is tested and it gained acceptance. The system was implemented successfully. Implementation is a process that means converting a new system in to operation. Java is a general-purpose, robust, secure, and object-oriented programming language. It is a high-level language, I.e., its syntax uses English like language. It was developed by Sun Microsystems in the year 1995. It is now maintained and distributed by Oracle. Java has its runtime environment and API; therefore, it is also called a platform.

It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of change over methods. Apart from planning major task of preparing the implementation are education and training of users. According to this plan, the activities are carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system. Implementation is the final and the most important phase. The system can be implemented only after through testing is done and if it is found to be working according to the specification.

The implementation view of software requirements presents the real world manifestation of processing functions and information structures. When the physical system has been designed in details, the name stage is to turn the design into a working system and then to monitor the operation of system to ensure that it continues to work efficiently and effectively.

The system will implemented only after the test is done on each of its subsystem and checking for whether it is working according to the specification. The system should provide proper documentation, because any work can passed from one person to another.

Coding

Coding in the process of whereby the physical design specification created by the analysis team turned into working computer code by the programming team.

Testing

Once the coding process is begin and proceed in parallel, as each program module can be tested.

Installation

Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, and documentation and work procedures those consistent with the new system

Documentation

It is result from the installation process, user guides provides the information of how the use system and its flow

Training and support

Training plan is a strategy for training user so they quickly learn to the new system. The development of the training plan probably began earlier in the project. The best-suited application package to develop the system is eclipse under android environment.

7.1 SAMPLE SOURCE CODE

Adapter.java

```
package com.demo.sqlitedababase;
import android.content.Context;
import android.content.DialogInterface;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import java.util.List;
public class EmployeeAdapter extends ArrayAdapter<Employee> {
Context mCtx;
int listLayoutRes;
List<Employee> employeeList;
SQLiteDatabase mDatabase;
public EmployeeAdapter(Context mCtx, int listLayoutRes, List<Employee> employeeList,
SQLiteDatabase mDatabase) {
super(mCtx, listLayoutRes, employeeList);
this.mCtx = mCtx;
this.listLayoutRes = listLayoutRes;
this.employeeList = employeeList;
this.mDatabase = mDatabase;
}
```

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup
parent) {
LayoutInflater inflater = LayoutInflater.from(mCtx);
View view = inflater.inflate(listLayoutRes, null);
final Employee employee = employeeList.get(position);
TextView textViewName = view.findViewById(R.id.textViewName);
TextView textViewUsername = view.findViewById(R.id.textViewDepartment);
TextView textViewEmail = view.findViewById(R.id.textViewSalary);
TextView textViewphno = view.findViewById(R.id.textViewJoiningDate);
textViewName.setText(employee.getMname());
textViewUsername.setText(employee.getMusername());
textViewEmail.setText(String.valueOf(employee.getMemail()));
textViewphno.setText(employee.getMphno());
ImageView buttonDelete = view.findViewById(R.id.buttonDeleteEmployee);
ImageView buttonEdit = view.findViewById(R.id.buttonEditEmployee);
//adding a clicklistener to button
buttonEdit.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
updateEmployee(employee);
}
});
//the delete operation
buttonDelete.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
AlertDialog.Builder builder = new AlertDialog.Builder(mCtx);
builder.setTitle("Are you sure?");
builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialogInterface, int i) {
String sql = "DELETE FROM Student WHERE id = ?";
mDatabase.execSQL(sql, new Integer[]{employee.getId()});
reloadEmployeesFromDatabase();
}
});
builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialogInterface, int i) {
}
});
AlertDialog dialog = builder.create();
dialog.show();
}
});
return view;
}
private void updateEmployee(final Employee employee) {
final AlertDialog.Builder builder = new AlertDialog.Builder(mCtx);
LayoutInflater inflater = LayoutInflater.from(mCtx);
View view = inflater.inflate(R.layout.dialog_update_employee, null);
builder.setView(view);
final EditText editTextName = view.findViewById(R.id.editTextName);
final EditText editusername = view.findViewById(R.id.editUsername);
final EditText editemail = view.findViewById(R.id.editEmail);
final EditText editphno = view.findViewById(R.id.editTextphno);
editTextName.setText(employee.getMname());
editusername.setText(employee.getMusername());
editemail.setText(employee.getMemail());
editphno.setText(employee.getMphno());
final AlertDialog dialog = builder.create();
dialog.show();
// CREATE METHOD FOR EDIT THE FORM
view.findViewById(R.id.buttonUpdateEmployee).setOnClickListener(new
View.OnClickListener() {
@Override
```

```
public void onClick(View view) {
String name = editTextName.getText().toString().trim();
String username = editusername.getText().toString().trim();
String email = editemail.getText().toString().trim();
String phno = editphno.getText().toString().trim();
if (name.isEmpty()) {
editTextName.setError("Name can't be blank");
editTextName.requestFocus();
return;
}
if (username.isEmpty()) {
editusername.setError("Username can't be blank");
editusername.requestFocus();
return;
}
String sql = "UPDATE Student \n" +
"SET Name = ?, \n" +
"Email = ?,\n"+
"Username = ?,\n"+
"PhoneNO= ? \n" +
"WHERE id = ?;\n";
mDatabase.execSQL(sql, new String[]{name, email,username,phno,
String.valueOf(employee.getId())});
Toast.makeText(mCtx, "Student Updated", Toast.LENGTH_SHORT).show();
reloadEmployeesFromDatabase();
dialog.dismiss();
}
});
}
private void reloadEmployeesFromDatabase() {
Cursor cursorEmployees = mDatabase.rawQuery("SELECT * FROM Student", null);
if (cursorEmployees.moveToFirst()) {
employeeList.clear();
do {
```

```
employeeList.add(new Employee(
cursorEmployees.getInt(0),
cursorEmployees.getString(1),
cursorEmployees.getString(2),
cursorEmployees.getString(3),
cursorEmployees.getString(4)
));
} while (cursorEmployees.moveToNext());
}
cursorEmployees.close();
notifyDataSetChanged();
}
}
```

MainActivity.java

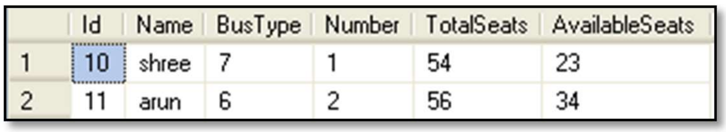
```
package com.demo.sqlitedababase;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Toast;
import com.demo.sqlitedababase.test.MainActivity2;
import com.google.android.material.snackbar.Snackbar;
public class MainActivity extends AppCompatActivity {
EditText e_name, e_email, e_username, e_phnumber;
Button bt_save,viewdata,viewdatall;
public static final String DATABASE_NAME = "StudentDatabases.db";
SQLiteDatabase mDatabase;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
mDatabase = openOrCreateDatabase(DATABASE_NAME, MODE_PRIVATE, null);
createEmployeeTable();
//FindById (Button and Edittxt)
e_name = (EditText) findViewById(R.id.e_name);
e_email = (EditText) findViewById(R.id.e_email);
e_username = (EditText) findViewById(R.id.e_username);
e_phnumber = (EditText) findViewById(R.id.e_phnumber);
bt_save = (Button) findViewById(R.id.btn_save);
viewdatall=(Button)findViewById(R.id.viewdataLL);
viewdatall.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Intent intent= new Intent(MainActivity.this, EmployeeActivity.class);
startActivity(intent);
}
});
viewdata = (Button) findViewById(R.id.viewdata);
viewdata.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Intent intent= new Intent(MainActivity.this, MainActivity2.class);
startActivity(intent);
}
});
//Onclick Btn
bt_save.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View view) {
//Get the Enter data
String name = e_name.getText().toString().trim();
String email = e_email.getText().toString().trim();
String username = e_username.getText().toString();
String phone = e_phnumber.getText().toString();
if (name.isEmpty() || email.isEmpty() || username.isEmpty() || phone.isEmpty()) {
Toast.makeText(MainActivity.this, "Fil the form", Toast.LENGTH_SHORT).show();
} else {
String insertSQL = "INSERT INTO Student \n" +
"(Name, Email, UserName, PhoneNo)\n" +
"VALUES \n" +
"(?, ?, ?, ?);";
//using the same method execsql for inserting values
//this time it has two parameters
//first is the sql string and second is the parameters that is to be binded with the query
mDatabase.execSQL(insertSQL, new String[]{name, email, username, phone});
Toast.makeText(MainActivity.this, "Great! Data Saved", Toast.LENGTH_SHORT).show();
}
}
});
}
private void createEmployeeTable() {
mDatabase.execSQL("CREATE TABLE IF NOT EXISTS Student " +
"(\n" +
"id INTEGER NOT NULL CONSTRAINT employees_pk PRIMARY KEY
AUTOINCREMENT,\n" +
" Name varchar(200) NOT NULL,\n" +
" Email varchar(200) NOT NULL,\n" +
" UserName varchar(200) NOT NULL,\n" +
"PhoneNo Varchar(200) NOT NULL\n" +
");
);
}
}
```

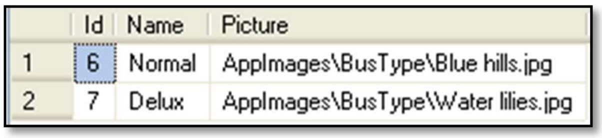
DATABASE:

BUS



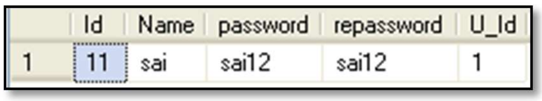
	Id	Name	BusType	Number	TotalSeats	AvailableSeats
1	10	shree	7	1	54	23
2	11	arun	6	2	56	34

BUS TYPE



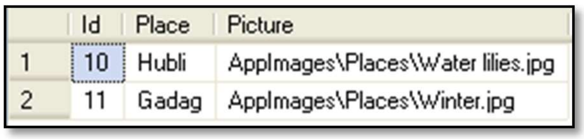
	Id	Name	Picture
1	6	Normal	AppImages\BusType\Blue hills.jpg
2	7	Delux	AppImages\BusType\Water lilies.jpg

CREATE NEW USER



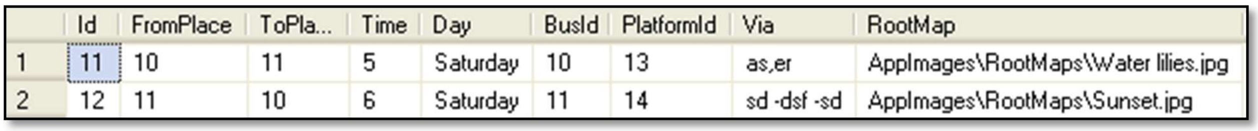
	Id	Name	password	repassword	U_Id
1	11	sai	sai12	sai12	1

PLACES



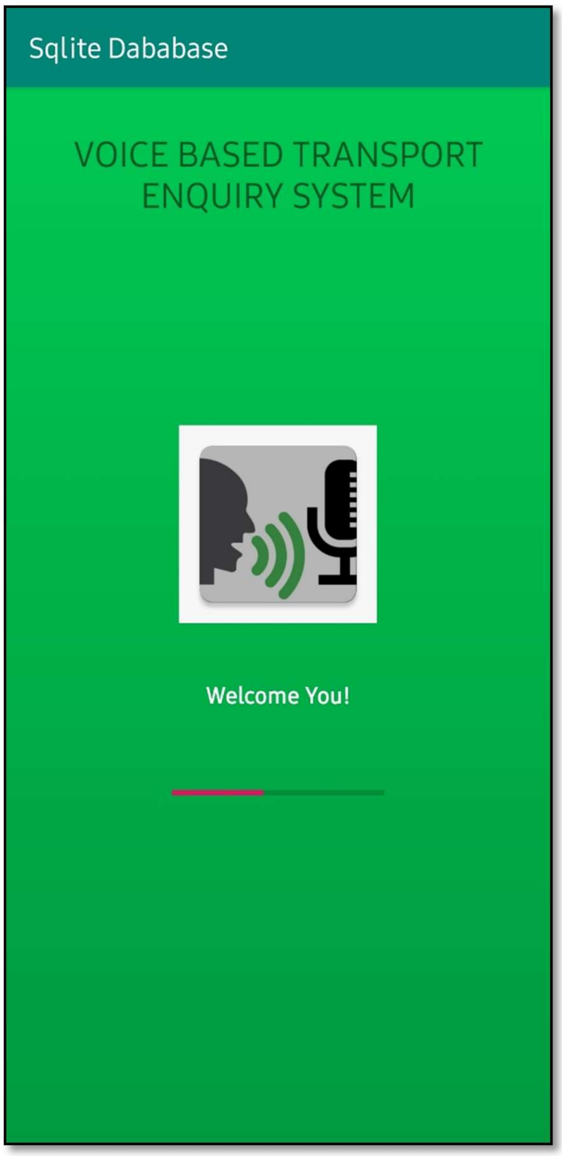
	Id	Place	Picture
1	10	Hubli	AppImages\Places\Water lilies.jpg
2	11	Gadag	AppImages\Places\Winter.jpg

TIMING



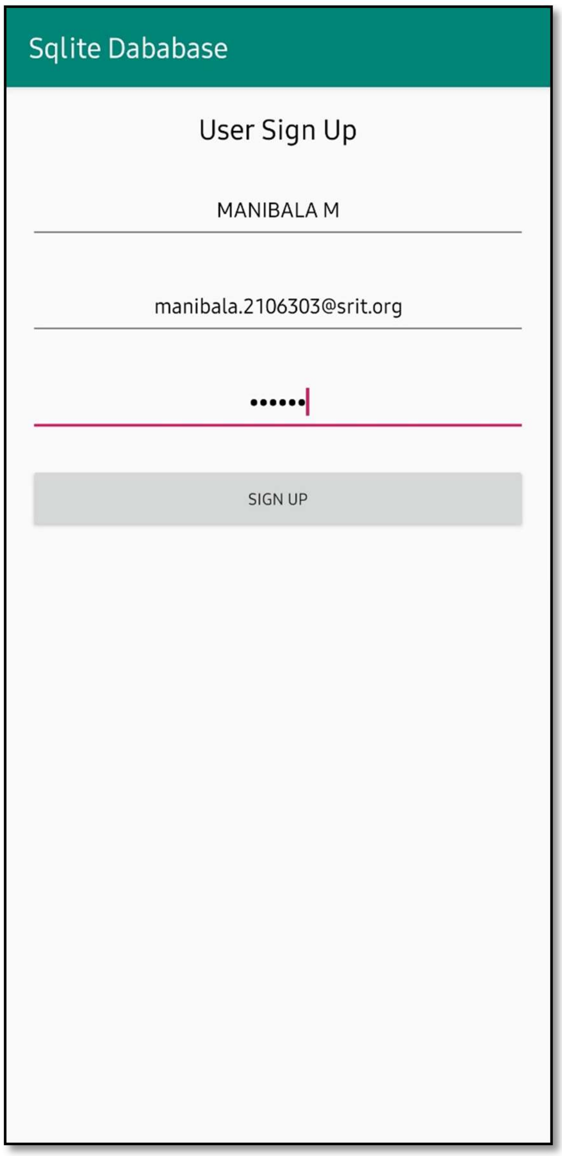
	Id	FromPlace	ToPla...	Time	Day	BusId	PlatformId	Via	RootMap
1	11	10	11	5	Saturday	10	13	as,er	AppImages\RootMaps\Water lilies.jpg
2	12	11	10	6	Saturday	11	14	sd -dsf -sd	AppImages\RootMaps\Sunset.jpg

7.2 SAMPLE SCREEN SHOTS


Sqlite Dababase

VOICE BASED TRANSPORT ENQUIRY SYSTEM

Welcome You!

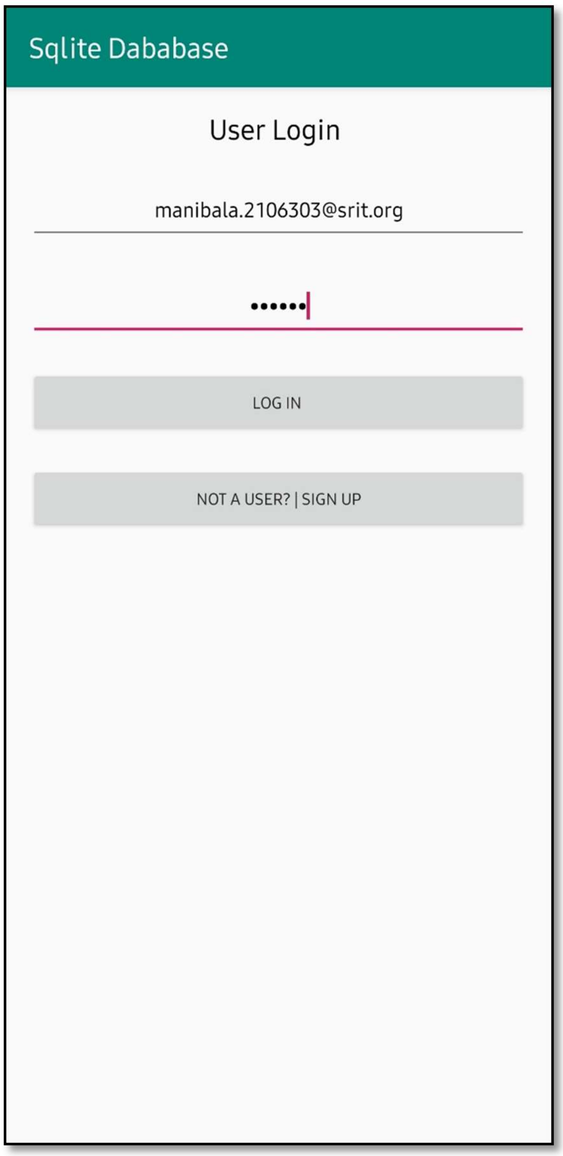

Sqlite Dababase

User Sign Up

MANIBALA M

manibala.2106303@srit.org

••••••


Sqlite Dababase

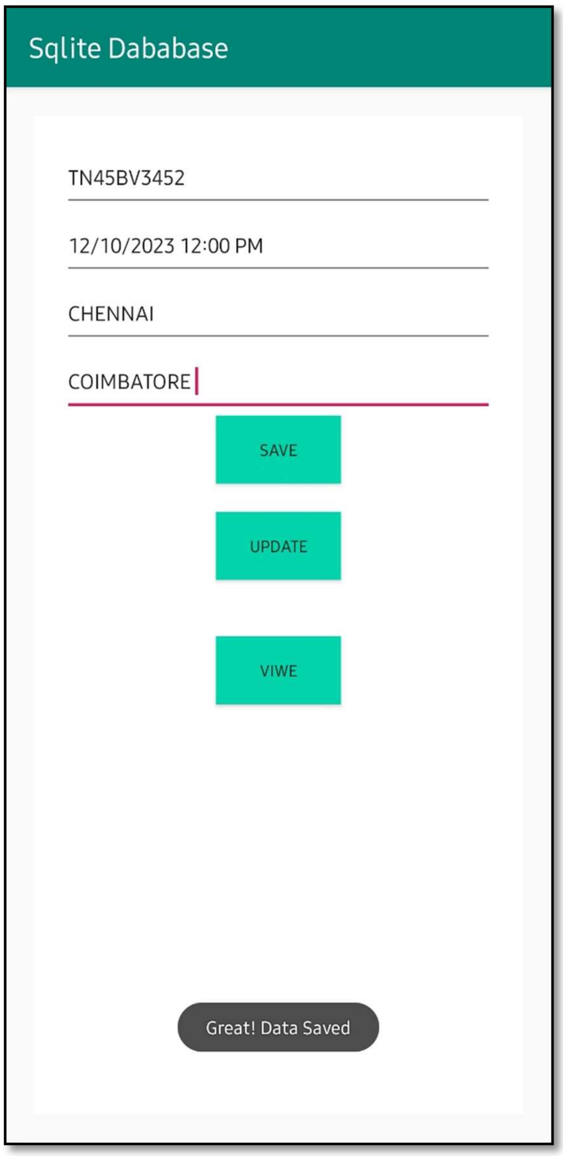
User Login

manibala.2106303@srit.org

••••••

LOG IN

NOT A USER? | SIGN UP


Sqlite Dababase

TN45BV3452

12/10/2023 12:00 PM

CHENNAI

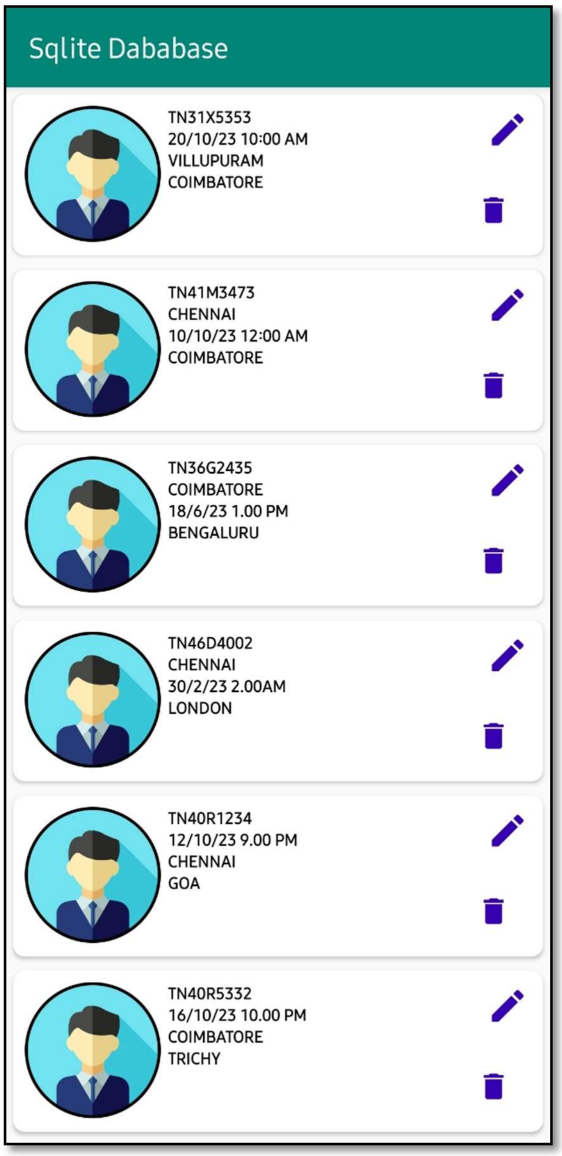
COIMBATORE

SAVE

UPDATE

VIWE

Great! Data Saved


Sqlite Dababase

TN31X5353
20/10/23 10:00 AM
VILLUPURAM
COIMBATORE

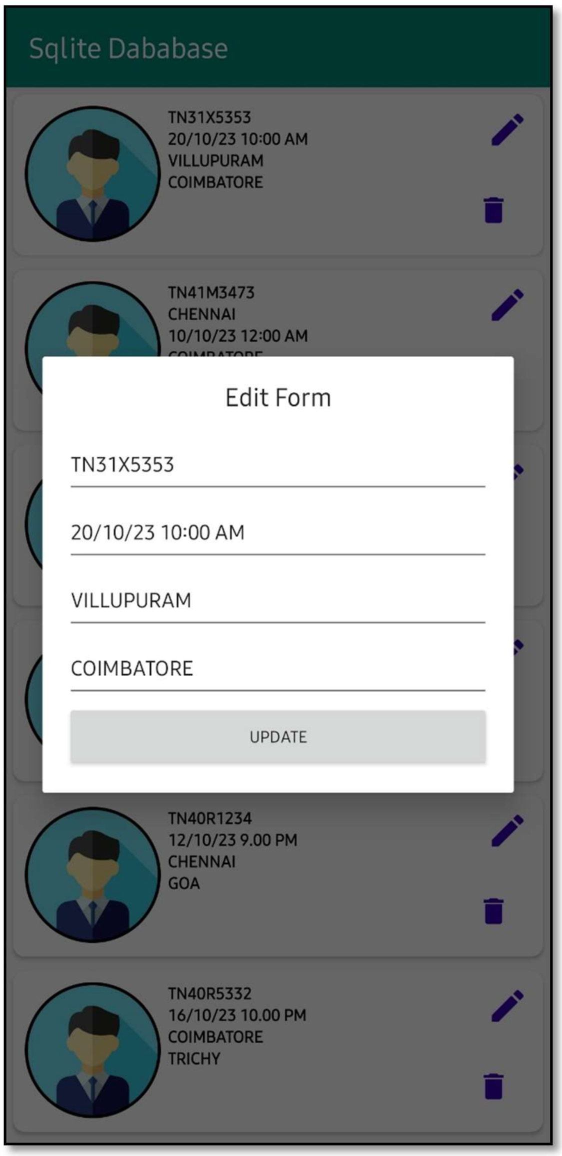
TN41M3473
CHENNAI
10/10/23 12:00 AM
COIMBATORE

TN36G2435
COIMBATORE
18/6/23 1.00 PM
BENGALURU

TN46D4002
CHENNAI
30/2/23 2.00AM
LONDON

TN40R1234
12/10/23 9.00 PM
CHENNAI
GOA

TN40R5332
16/10/23 10.00 PM
COIMBATORE
TRICHY


Sqlite Dababase

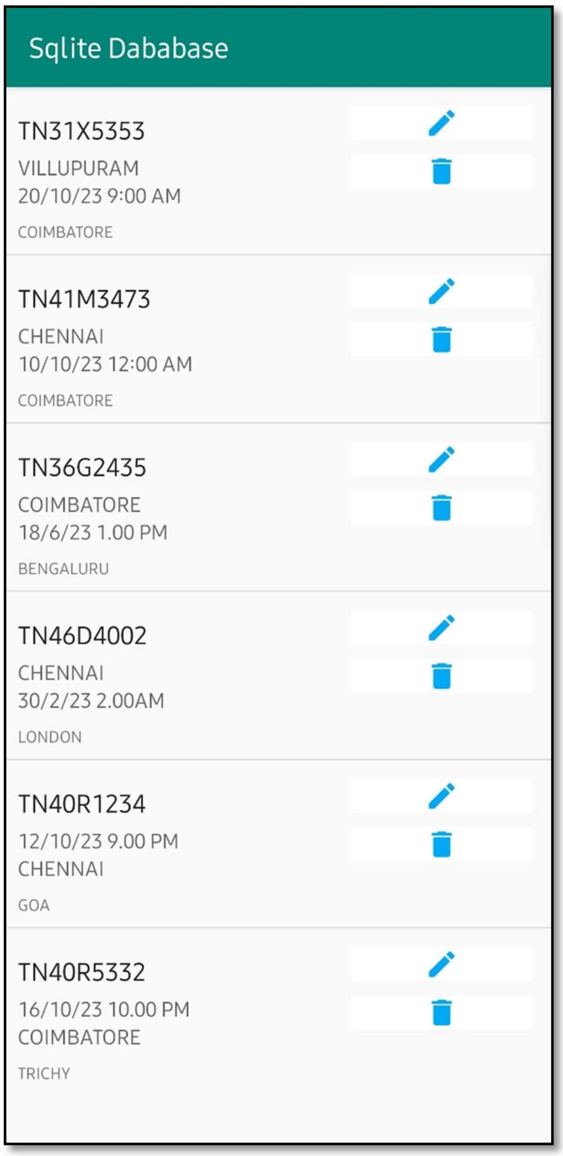
Edit Form

TN31X5353

20/10/23 10:00 AM

VILLUPURAM

COIMBATORE



Sqlite Dababase

TN31X5353
VILLUPURAM
20/10/23 9:00 AM
COIMBATORE

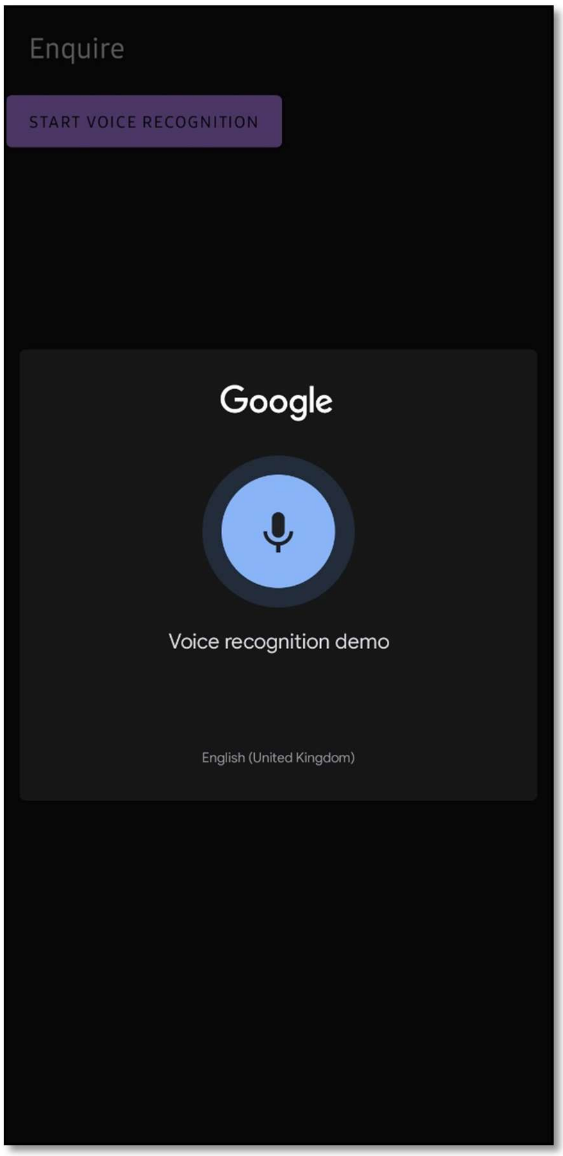
TN41M3473
CHENNAI
10/10/23 12:00 AM
COIMBATORE

TN36G2435
COIMBATORE
18/6/23 1.00 PM
BENGALURU

TN46D4002
CHENNAI
30/2/23 2.00AM
LONDON

TN40R1234
12/10/23 9.00 PM
CHENNAI
GOA

TN40R5332
16/10/23 10.00 PM
COIMBATORE
TRICHY



CONCLUSION

With increase in world population, traffic and congestion is also increasing, and so the difficulty to find a place is increasing too. Moreover, if a person visits a new place it is really hard for him/her to find a route and available transport to reach their destination. If anyone visit to a new place without knowing the local language then it is a real problem for them to reach the proper destination. This system will be helpful for user to overcome all these flaws. Moreover, human voice is the most useful and easiest way of communication and as this system takes voice as input and gives voice output, it is easy to understand for any one and hence solves the entire problem for a traveller. It will need a working internet plan and all the information of traffic around the whole world will be available to the user.

8.1 FEATURE ENHANCEMENT

- Cost effective
- Does not require any extra hardware component rather than micro phone(Mice).
- This same technology can be used in other fields like hotels etc.

BIBLIOGRAPHY

TEXT BOOKS

1. Herbert Schildt.2008 ,"**JAVA COMPLETE REFERENCE**", Tata McGraw-Hill , 7th Edition, pp. 177-180.
2. GradyBrooch,James Rambaugh.1998,"**UNIFIED MODELING LANGUAGE USER GUIDE**", Addison Wesley Publishing, chapter 8-31.
3. Elias Awath, "**SYSTEM ANALYSIS AND DESIGN**", Tata Mc Graw Hill Publication, Sixth Edition, 2003.
4. S. Ramachandran, "**COMPUTER AIDED DESIGN**", Air Walk Publication, Third Edition, 2003.

WEBSITESS

1. www.android.com
2. www.google.com
3. http://developer.android.com/index.html
4. http://en.wikipedia.org/wiki/SQLite