ABSTRACT

Advancement of technology has made people to get things done on the go. Such is the case with mobile applications developed on various mobile operating systems. One application is the use of location based services which provides information about gas stations, places to visit, food outlets, etc. While travelling using a mobile phone, users share their location, photos and videos on social networking websites. In a similar way, drivers could share information with other drivers which could facilitate in saving of time, fuel and life. This android application would allow drivers to share information about traffic, road accidents and any important information based on the driving experience. The data shared by any driver is tagged onto particular location in a map and then displayed as a notification or marker with associated details on the application to all other drivers. The drivers can search for directions from a source to a destination address. If user has lost track from his/her directed path to destination address then user can get directions from current location using reroute option.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>ii</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>vi</td>
</tr>
<tr>
<td>1. BACKGROUND AND RATIONALE</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1.1 What is Android?</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1.2 Introduction</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>1.3 Existing Applications</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1.3.1 Waze</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1.3.2 Inrix Traffic</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1.3.3 Trapster</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>1.4 Proposed Solution</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>2. NARRATIVE</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.1 Problem Statement</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.2 Motivation</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.3 Product Description</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.4 Product Scope</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>2.5 Requirements</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>2.6 System Requirements</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>3. APPLICATION DESIGN</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
3.1 Application Design and Architecture ...........................................................................8

3.1.1 Why Parse? .................................................................................................................9

3.1.2 What are Amazon Web Services? ...............................................................................9

3.2 Use Case Diagram .......................................................................................................11

3.3 Class Diagram ...........................................................................................................12

3.4 Sequence Diagram ....................................................................................................13

4. IMPLEMENTATION OF APPLICATION MODULES .................................................14

4.1 Registration and Login ...............................................................................................14

4.2 Add Update ................................................................................................................16

4.3 Setting the Radius ......................................................................................................17

4.4 Query Database to View an Update ............................................................................18

5. TESTING AND RESULTS .............................................................................................20

5.1 Android Application Testing .....................................................................................20

5.1.1 Register with the Application .................................................................................20

5.1.2 Login to Application ..............................................................................................22

5.1.3 Adding an Update ...................................................................................................23

5.1.4 Setting the Radius ..................................................................................................25

5.1.5 Directions Search ...................................................................................................26

5.1.6 View Updates .........................................................................................................29

5.2 User Study ................................................................................................................30
6. CONCLUSION AND FUTURE WORK ................................................................. 31

BIBLIOGRAPHY AND REFERENCES ...................................................................... 32

Appendix A: Android Application End-User .......................................................... 33

A.1 Screens the End User Will See ........................................................................ 33

A.1.1 LoginFragment.xml ................................................................................. 33

A.1.2 RegisterFragment.xml .............................................................................. 34

A.1.3 MainActivity.xml .................................................................................... 35

A.1.4 AddActivity.xml ....................................................................................... 36

A.1.5 Settings.xml ............................................................................................. 37

A.1.6 DirectionsSearch.xml .............................................................................. 38
LIST OF FIGURES

Figure 3.1 The system design ........................................................................................................... 8
Figure 3.2 Android end user design flowchart ............................................................................... 10
Figure 3.3 Use case diagram .......................................................................................................... 11
Figure 3.4 Class diagram ............................................................................................................... 12
Figure 3.5 Sequence diagram ....................................................................................................... 13
Figure 4.1 Registration module ..................................................................................................... 14
Figure 4.2 Login module ................................................................................................................. 15
Figure 4.3 Adding update module .................................................................................................. 16
Figure 4.4 Setting the radius module ............................................................................................. 17
Figure 4.5 Query database to view an update ............................................................................... 19
Figure 5.1 User registration process ............................................................................................ 21
Figure 5.2 User login process ........................................................................................................ 23
Figure 5.3 Add update process ...................................................................................................... 24
Figure 5.4 Setting radius process .................................................................................................. 26
Figure 5.5 Directions search process ............................................................................................. 27
Figure 5.6 Directions reroute process ......................................................................................... 28
Figure 5.7 View update process .................................................................................................... 29
Figure A.1 LoginFragment.xml .................................................................................................... 33
Figure A.2 RegisterFragment.xml ................................................................................................ 34
Figure A.3 MainActivity.xml ........................................................................................................ 35
Figure A.4 AddActivity.xml .......................................................................................................... 36
Figure A.5 Settings.xml ................................................................................................................ 37
1. BACKGROUND AND RATIONALE

Ever since the development of new technologies human has been able to achieve what he intends to, in a short interval of time. It could be communicating with someone in far distant land, monitoring weather conditions, analysis of information about any past event, etc. This could lead to plan better, work and achieve success in future. Traditional mobile phones generally provided features such as make a call, send text message, phonebook, calculator, etc. The development of advanced operating system for mobile phone has widened the scope of functionalities that a phone user could have expected at certain point of time. Over the last decade various mobile operating systems have been developed such as iOS by Apple Inc., Android by Google and Windows phone by Microsoft being the prominent.

1.1 What is Android?

Android is a Linux based operating system developed by Android Inc., later acquired by Google. The first phone to use Android OS was HTC Dream. Android focused on providing better usability to users with features such as wide touch screen, input using swipe across, slide up and down, typing on QWERTY touch keypad, etc. Google being a founder member of Open handset alliance, an alliance of companies to develop open standards for mobile devices, has made Android platform open to all for development of android applications. New Android applications are made available for download at Google Play store.

Different android versions has been released till date, the most recent being the Android 5.0 Lollipop. Applications on Android platform are developed using Java Programming language. Java is a platform independent language, means write once run anywhere. An android application can be developed using Android Integrated Development Environment (IDE) named
Android Studio. Eclipse IDE could be used for development along with Android Software development Kit (SDK) and Android developer tools (ADT) plugin for eclipse.

Eclipse IDE has several versions such as Luna, Helios, Indigo, etc. It provides development environment to develop applications in programming language such as Java, PHP, etc. Android SDK includes various tools used for creating and testing of android applications such as libraries, tools, code for reference, emulator and debugger. Android applications are developed in Java using android SDK in Eclipse IDE.

1.2 Introduction

Connectivity across different cities is one of the driving factors which benefits the economy of a country; goods are shipped from one place to the other, people move across different cities, time of travel is reduced, etc. However as more and more people travel by road one wonders how safe it is travel on a particular route. Someone could face problem such as traffic congestion, accident and any untoward incident which could affect the life and economy. To avoid traffic congestion, methods such as ramp flow meters, installation of cameras, and electronic display boards are used to extract information by using the infrastructure facilities to take appropriate action so as to safe life.

Smart phones are equipped with Global positioning system (GPS) which gets the location information about a particular person. One way is to pull the location information and speed based on GPS from mobile device and use to get real time traffic update. Other way is to get drivers on a particular route to participate and share information about traffic jam, accidents, etc. on to a server which can be accessed by various other users. An example of android application for navigation is Waze [1], which pulls up location and speed information and allows drivers to share information with other drivers.
Generally traffic information is reported by transportation department, the police patrol units, any driver or person via call and by reporting agency. This information is passed on to general public by using communication mediums such as Radio, News agencies and Television channels. Apart from these, public participation can play important role as smart phone users can share information with other drivers. This can affect the selection of route; avoid any hazard and proper time management. A driver travelling from location to X to Y looks for time duration to reach a destination, preferably the shortest distance and least time to travel.

1.3 Existing Applications

1.3.1 Waze

Waze, a navigation application which provides users to post information about traffic jam, accident, and gas stations to a server, that information is used to do real time routing and share traffic update with other users. It allows sharing police traps information which has been subject of debate of recent times [2] as it could lead to a convict being escaped by changing a route or a dangerous encounter with a cop.

1.3.2 Inrix Traffic

Inrix Traffic [3] application uses crowdsourced information about road accidents, construction activity, and traffic jam to share as traffic reports to other users. However, this application is using traffic information to share with businesses.
1.3.3 Trapster

Trapster [4], a community based application which used crowdsourcing to get information about police speed traps, speed monitoring cameras, etc. and alerts drivers. However, Trapster discontinued their service due to financial issues.

1.4 Proposed Solution

This application creates a social network of drivers on a particular route in a certain radius, who could share information about accident, construction activity, crime, etc. Each driver would be asked to sign up with the application. Each time a user logs in a home screen is shown up with features such as search directions, which can be used to enter source and destination address which he intends to travel. A driver can post an update about traffic jam, accident, construction activity, hazard, etc. and share, this information is sent to a Parse Web server and Database [5] which is accessed by various drivers. Other drivers would refresh and get the updates from the server. This application makes drivers aware so as to get a clear update about the route they will travel before they start on the journey.
2. NARRATIVE

2.1 Problem Statement

Road safety is one of the major concerns for a traveler. Accident and construction activities on a particular route can lead to traffic congestion. There is a need to address the concerns of traveler. At present there are few mobile applications which gather information from travelers on a particular route and gives information to users about traffic jam, construction and other alerts. However, the existing applications do not take into account selection of radius to know the activity around the user and at what distance it is from user’s current location.

2.2 Motivation

Smartphone can be equipped with an application which can help users to know about the available traffic on road, accident information, safety of place, etc. Smart phone users can posts alert of a traffic jam, a short message about accident or any untoward incident on the road or place while travelling which could facilitate the other travelers on that road. This project aims to build an application which could help on to share information to other users. It aims to use crowd sourced data to help other users.

2.3 Product Description

This application considers following points into account –

- Search directions based on user input
- Two way communication between User and Server
- User post’s update about traffic event in a particular radius
- User view’s posts from other user posted in that particular radius
- Crowdsourced information is shared with other users
This application uses crowdsourced data i.e., from users using the application. Users can share traffic updates and tag them on a particular location in the map. This application allows a user to share information within a certain radius, enabling the user to be connected with other users. Finally, users can view posts in order to be aware of what can be expected on the route a traveler intends to travel.

2.4 Product Scope

The product is developed for Android smart phone users where

- A user has internet connectivity
- An Android phone is used with GPS enabled
- The user is registered with the application

The application is connected to Parse Web server and database, where the posts from other users are saved.

2.5 Requirements

1. First time user should be able to register with the application.
2. User already registered with the application should be able to login.
3. Registered user should be able to search source and destination places.
4. Registered user should be able to get directions from source address to destination address based on the address specified in the search field.
5. Directions should be marked from source to destination and displayed as connection between two address points on a map.
6. If user deviates from direction path then he should be provided with option to reroute from his current location to destination address.
7. Registered user should be able to set the distance as a radius measure, which allows user to view any update from other users, those who might be online/offline in that radius.

8. Registered user should be able to post an update on his/her current location about certain traffic event or any other comment.

9. A user should only be able to view post in the radius selected by him/her. Other posts which are not in the radius should not be allowed to view.

10. User logged in should be able to logout from the application.

2.6 System Requirements

To develop this android application below mentioned are the requirements

- Operating System: Windows
- Development Environment: Eclipse IDE for Java Developers
- Plugins: Android Development Toolkit (ADT) plugin for Eclipse
- Software: Android Software Development Kit (Android 5.0.1 API 21), Java SE 7
- Android Device: Google Nexus 5
3. APPLICATION DESIGN

3.1 Application Design and Architecture

The Architecture diagram, as shown in Figure 3.1, describes the proposed application design. The application is installed on Android Mobile device. The user should have internet connectivity on the device being used. When a user registers with the application, user information is sent and stored on Parse Web server and Database, an online cloud service provided by Parse. A user can search a source or destination address, that he/she wants to travel to, as soon as the user types the address, suggestions are given to the user, this is achieved using Google Autocomplete Places API. Once a user selects source and destination address and requests to get directions, the directions are marked on to the Google Map using Google Directions API. The user is allowed to post an update about traffic, road construction, hazard, etc.
3.1.1 Why Parse?

Parse provides cloud-based backend services for mobile application developers. Parse delivers mobile services so that developers can work on developing applications and leave database, server and rest of infrastructure to Parse. Parse handles user account management, data storage and can provide better infrastructure based on usage fluctuation on a day-to-day basis.

3.1.2 What are Amazon Web Services?

Amazon Web Services (AWS) is a collection of remote computing services, also called web services, that makes up a cloud computing platform offered by Amazon. These services are based out of 11 geographical regions across the world. The most central and well-known of these services are Amazon EC2 and Amazon S3. These products are marketed as a service to provide large computing capacity more quickly and cheaper than a client company building an actual physical server farm[6].

Parse uses AWS to build App Services for Mobile App Developers. Parse webserver is responsible for communication between various users on the network, so that each one of them can see traffic updates being posted by other users.

Parse web server provides features such as -

- Storage of data on cloud up to 20 Giga Bytes.
- Supports logging to an application through Facebook and Gmail account.
- Local data storage to query when no internet connection is available.
- Configuration changes at server end to change application settings.
- Data sharing across different platforms.
Figure 3.2 Android end user design flowchart

The screenshots for activities of application are listed in Appendix A. When a user first opens the application, the user has two options either to login or register. A new user goes for registration and an existing user for login. If the user does not pass all validations then the user is redirected to Login/Signup page. If all validations are passed, the user is directed to main activity, where he has got options to either navigate to Directions Activity or settings activity or to add an update to the application. Users can logout from any of these activities.
3.2 Use Case Diagram

The use case diagram as shown in Figure 3.3 depicts the interaction between user and different modules of the application. A new user would first register with the application, once registered, the user can access application by logging in. The user can search for directions, add an update to the application and change the settings of the application.

Figure 3.3 Use case diagram
3.3 Class Diagram

Class diagram depicts the relationship among different classes, the variables and methods being used. The class diagram for the application is as shown in Figure 3.4, where the root class is ParseObject class which provides methods to query database, signup and login to parse database. ParseUser class inherits methods from ParseObject class and is used for registration and login. TravellerApp class is used to store data on to Parse database.

![Class Diagram Image](image-url)

Figure 3.4 Class diagram
3.4 Sequence Diagram

The sequence diagram as shown in Figure 3.5 depicts the sequential flow of the application. First, a user requests to login to the application; this user requesting for service has to be authenticated hence it invokes Parse server. Parse then passes a data request on to database, the data is retrieved and returned to the user as response.

Figure 3.5 Sequence diagram
4. IMPLEMENTATION OF APPLICATION MODULES

4.1 Registration and Login

This part of the application deals with registering a new user or validating an existing user at login. ParseUser is a java class provided by Parse which allows user data to be saved and retrieved from the cloud database. While registering as shown in Figure 4.1, setUsername() and setPassword() methods of ParseUser class are invoked with parameters as username and password, input provided by the user. signUpInBackground() method is invoked to sign up a new user and to handle the response from the server. If no exception is found then the user is shown a toast message as “Successfully Registered with TravelWize.”, and is navigated to MainActivity page of the application, else the user is shown appropriate validation error message on the signup page and is not allowed to proceed with registration.

```java
ParseUser newUser = new ParseUser();
newUser.setUsername(uname);
newUser.setPassword(pwd);

newUser.signUpInBackground(new SignUpCallback() {
    @Override
    public void done(ParseException exc) {
        dialog.dismiss();
        if (exc == null) {
            Toast.makeText(getActivity(),
                    "Successfully Registered with TravelWize.",
                    Toast.LENGTH_LONG).show();
            Intent intent = new Intent(getActivity(),
                    MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
                    Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
        } else {
            Toast.makeText(getActivity(), exc.getMessage(),
                    Toast.LENGTH_LONG).show();
        }
    }
});
```

Figure 4.1 Registration module
When an existing user tries to login to the application, logInInBackground() method, as shown in Figure 4.2 is invoked with user provided input of username and password to login and to handle the response from the server. If no exception is found then user is shown a toast message as “Successfully Logged In.”, and is navigated to MainActivity page of the application, else the user is shown an appropriate validation error message on the signup page and is not allowed to proceed with login.

```java
ParseUser.logInInBackground(uname, pwd, new LogInCallback() {
    @Override
    public void done(ParseUser user, ParseException exc) {
        dialog.dismiss();
        if (exc != null) {
            if (exc.getCode() == ParseException.OBJECT_NOT_FOUND) {
                Toast.makeText(getActivity(), "Invalid Credentials", Toast.LENGTH_LONG).show();
                password.selectAll();
                password.requestFocus();
            } else {
                Toast.makeText(getActivity(), "Login request has failed. Try again", Toast.LENGTH_LONG).show();
            }
        } else {
            Toast.makeText(getActivity(), "Successfully Logged In.", Toast.LENGTH_LONG).show();
            Intent intent = new Intent(getActivity(), MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
        }
    }
});
```

Figure 4.2 Login module
4.2 Add Update

This part of the application deals with adding an update to the application. A user can select options available to the user and add details. TravellerApp is a java class which creates a table on Parse database. TravellerApp class setter methods are invoked with the parameters as user provided input to save data on Parse database. The image before being saved is compressed; ParseAcl class is used to set read access permission in Parse database.

```java
private void addUpdate() {
    String message = tap_et_update.getText().toString().trim();
    String event_type = eventType.getSelectedItem().toString().trim();

    TravellerApp newUpdate = new TravellerApp();

    if (photoFile != null) {
        newUpdate.setUser(ParseUser.getCurrentUser());
        newUpdate.setText(message);
        newUpdate.setLocation(tap_parseGeoPoint);
        newUpdate.setPhotoFile(photoFile);
        newUpdate.setEventType(event_type);
    } else {
        newUpdate.setUser(ParseUser.getCurrentUser());
        newUpdate.setText(message);
        newUpdate.setLocation(tap_parseGeoPoint);

        Bitmap icon = BitmapFactory.decodeResource(context.getResources(),
                R.drawable.no_image_available_icon);
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        icon.compress(Bitmap.CompressFormat.JPEG, 100, stream);
        byte[] image = stream.toByteArray();

        newUpdate.setPhotoFile(new ParseFile("photo.jpg", image));
        newUpdate.setEventType(event_type);
    }

    ParseACL parseAcl = new ParseACL();
    parseAcl.setPublicReadAccess(true);
    newUpdate.setACL(parseAcl);
    newUpdate.saveInBackground(new SaveCallback() {
        @Override
        public void done(ParseException pexc) {
            finish();
        }
    });
}
```

Figure 4.3 Adding update module
4.3 Setting the Radius

This part of the application deals with setting the radius, which allows a user to view an update or a post, posted by other users. In order to use Google APIs provided in Google play services library an instance of GoogleApiClient is created. The API being used is passed as a parameter to addApi() method, such as LocationServices API, as shown in Figure 4.4. When a user selects a radius value, then a circle is added on the map fragment being shown to user, this is achieved using addCircle() method. The center of the circle is considered as the user’s current location latitude and longitude values and is set using setCenter() method. The setRadius() method access the input parameter in terms of meters, hence as the value selected by user is in miles, appropriate conversion from miles to meters is performed and the value is passed on to setRadius() method.

```java
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .addApi(LocationServices.API)
    .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this).build();

mfragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
mfragment.getMap().setMyLocationEnabled(true);
mfragment.getMap().getUiSettings().setZoomControlsEnabled(true);
mfragment.getMap().setTrafficEnabled(true);
mcircle = mfragment.getMap().addCircle(new CircleOptions().center(myLatLng).radius(radius * milesToMeters));
mcircle.setStrokeColor(Color.DKGRAY);
mcircle.setStrokeWidth(2);
mcircle.setFillColor(Color.argb(50,Color.red(Color.DKGRAY),Color.green(Color.DKGRAY), Color.blue(Color.DKGRAY)));
mcircle.setCenter(myLatLng);
mcircle.setRadius(radius * milesToMeters);
```

Figure 4.4 Setting the radius module
4.4 Query Database to View an Update

Whenever a user wants to view an update, the user clicks on marker icon, onMarkerClick() method is invoked and then the user is presented with a dialog window with the information posted by other user. In order to retrieve data from Parse database, ParseQuery is setup to get TravellerApp data. The extracted data is presented in a dialog window with text i.e. message posted by the user, distance from the user’s current location to the marker or event location and an image if any has been uploaded.

```java
mFragment.getMap().setOnMarkerClickListener(new OnMarkerClickListener()
{
    @Override
    public boolean onMarkerClick(Marker marker)
    {
        final Dialog d = new Dialog(context);
        d.setContentView(R.layout.view_posts);
        if(marker.getTitle().equalsIgnoreCase("Get Close to view the post")){
            marker.showInfoWindow();
            return true;
        }
        else{
            ParseQuery<TravellerApp> mQ1 = TravellerApp.getQuery();
            mQ1.getInBackground(marker.getSnippet(),
            new GetCallback<TravellerApp>() {
                @Override
                public void done(TravellerApp arg0,
                ParseException arg1) {
                    ParseFile pf;
                    d.setTitle(arg0.getEventType());
                    double dist = (double)
                    arg0.getLocation().distanceInMilesTo(myPoint);
                    dist = Double.parseDouble(new DecimalFormat("##.##").format(dist));
                    String s = String.valueOf(dist) + " miles away from your location.";
                    pf = arg0.getPhotoFile();
                    if (pf != null) {
                        pf.getDataInBackground(new GetDataCallback() {
```
```java
@Override
public void done(byte[] data,
                 ParseException arg1) {
    if (arg1 == null) {
        ParseImageView pvImage = (ParseImageView) d.findViewById(R.id.image_preview);
        Bitmap bMap = BitmapFactory.decodeByteArray(data, 0, data.length);
        pvImage.setImageBitmap(bMap);
    } else {
    }
}
```

Figure 4.5 Query database to view an update
5. TESTING AND RESULTS

This section describes the end user application testing details.

The main functionalities of the application are –

- Register with the Application
- Login to Application
- Add/View an Update
- Setting the Radius
- Directions Search

5.1 Android Application Testing

5.1.1 Register with the Application

A user downloads the application on his Android mobile/tablet device. When user runs the application, the user is presented with startup screen as shown in Figure 5.1 (a), which gives user two options either to Login or Register with the application. New user, who wants to use the application, would not have username and password to login to the application. The user needs to register with the application. The user clicks on sign up page, user is presented with register screen as shown in Figure 5.1 (b). Each edit text fields on the signup page has certain validations. When a user does not enters any data and tries to register with the application then user is shown a message to enter username and password as shown in Figure 5.1 (c). If a user enters a username but does not enter password then user is shown to enter password as shown in Figure 5.1 (d). User is required to enter same password in password and re-enter password fields, if user does not then an error message is shown on the screen as shown in Figure 5.1 (e). User enters Username and password and clicks on register button. When the registration is successful the user is presented with main screen as shown in Figure 5.1 (f).
Figure 5.1 User registration process

(a) Startup screen of TravelWize application
(b) Signup page where user can register
(c) Scenario – when user leaves input fields blank
(d) Scenario – when user enters username but password as blank
(e) Scenario – when user enters different password in both password fields
(f) Scenario – when user is successfully registered
5.1.2 Login to Application

When a user tries to login to the application, if user does not enters username and password in the edit text field’s then application prompt’s an error message to enter Username and password as shown in the figure 5.2 (a). If user enters username but password field is left blank then an error message to enter password is prompted on screen as shown in figure 5.2 (b). If user enters password but username field is left blank then an error message to enter username is prompted on screen as shown in figure 5.2 (c). If user enters wrong username and password then an error message as Invalid credentials is prompted on screen as shown in figure 5.2 (d). If user enters correct login details and clicks on login button then user is directed to main page of the application with a toast message as shown in figure 5.2 (e).
5.1.3 Adding an Update

When a user wants to add an update, then the user clicks on add update button on the main page of application, the user is directed to Add an update page. The user enters the text message to post as shown in Figure 5.3(c), selects the event type of what he wants to report about and clicks on save button. The user is then navigated to the main page of TravelWize application where a marker icon is displayed on the map. The marker icon varies based on the type of event the user selects to report.
Figure 5.3 Add update process

(a) Add an update page
(b) Scenario – application allows speech input to add text
(c) Scenario – speech input provided is displayed as text
(d) Scenario – user selects what he wants to report
(e) Scenario – user selects to add an photo
(f) Scenario – The entered information is displayed on the map
5.1.4 Setting the Radius

When a user wants to update the radius then, the user can change it from the settings page. The user is presented with options as radio buttons to select a particular radius. In this scenario, a user first selects 2 miles as radius as shown in Figure 5.4 (a), then navigates to the main page. The radius is shown as circle as shown in Figure 5.4 (b). When a user wants to change the radius from a previous selected value then the user can change the radius to 10 miles radius as shown in Figure 5.4 (c), and then navigates to the main page. The radius is shown as an updated circle as shown in Figure 5.4 (d).
Figure 5.4 Setting radius process

(a) Scenario – when user selects radius as 2 miles
(b) Scenario – output as when radius is 2 miles
(c) Scenario – when user selects radius as 10 miles
(d) Scenario – output as when radius is 10 miles

5.1.5 Directions Search

When a user wants to search the directions, the user searches for a destination address place, the place is shown using Google Places API as shown in Figure 5.5 (a). The user sets the source address as current location and the destination address as Dallas, TX, United States then input is provided to Google Directions API and the result returned is plotted as polyline on the map with the blue color as shown in Figure 5.5 (c).
Figure 5.5 Directions search process

(a) Scenario – user searches for a particular destination address
(b) Scenario – user selects a place from given options
(c) Scenario – output plotted as polyline based on user input
(d) Scenario – turn by turn navigation shown by google maps
If a user has deviated from the original path then by clicking Reroute button, the user can find the direction to the destination from the current location, as shown in Figure 5.6.

Figure 5.6 Directions reroute process

(a) Scenario – user is shown direction from current location
(b) Scenario – user deviates from path
(c) Scenario – reroute gets directions from current location
5.1.6 View Updates

When a user wants to view an update posted by other users, then the user clicks on the marker icon on the map; the appropriate data associated with the post is displayed as a dialog window as shown in Figure 5.7.

![Figure 5.7 View update process](image-url)
5.2 User Study

As part of user study the application was shared with android users. The application was tested by the users and as per the inputs received, the users had trouble in entering the password on the signup page and clicking ‘Cancel’ on the add an update page. On observation, it was found that as the application was tested on Android tablet device the layout of any page was visible clearly, however, on a small display screen when compared to a tablet device, the complete layout was not visible. Hence, scrollview was added which allows users to scroll on each page which lets them to access the whole layout with the scroll feature.

Another issue reported by a different user was to allow the user to add a picture which is already stored on a device. When a user is driving a car, the user might take a picture, but the user might report the incident after some duration, then this feature ‘to add stored image’ is helpful. Prior to that a user was allowed only to take picture through camera and upload.
6. CONCLUSION AND FUTURE WORK

The goal of this application was to build an android application which could assist or alert users for their safety while they are on the road. This was applied on to the scenario of road travelers where travelers could be informed about any danger while they are travelling. This Android application uses Google Places API, Google Maps API and Google Directions API in order to retrieve information about places and routes. Users are allowed to add an update about any incident such as traffic jam, closure, construction activity or any other event. Input from a user are stored on Parse webservice and cloud database, which can be accessed by all the users registered with the application, thereby creating a social network of travelers and helping users with crowdsourced data.

Some future work for this project can include:

- Betterment of user interface design
- Implementation for iOS platform
- Implementation of text to voice notification when user clicks on to view an update
- Implementation of proximity alert based on certain radius
- Presenting users with safety and security tips
BIBLIOGRAPHY AND REFERENCES


Appendix A: Android Application End-User

A.1 Screens the End User Will See

This part of the document is about the screen shots of the application running on device.

A.1.1 LoginFragment.xml

This screen gives the option to a user to either go to login or sign up with the application. It uses the sliding tab layout android material design feature for a better graphical user interface. Figure A.1 shows the startup screen for the application.

Figure A.1 LoginFragment.xml
A.1.2 RegisterFragment.xml

This screen allows a new user to register with the application. It has three EditText fields where any user can enter the username, the password and re-enter the password. The password entered in the password and the re-enter password fields should match. It also includes a button to register with the application, as shown in Figure A.2.

Figure A.2 RegisterFragment.xml
A.1.3 MainActivity.xml

This is the main page of the application where a user can view his current location on a map, the updates posted by other registered users of the application, a button which on clicked navigates the user to AddActivity.xml page and an action item list at the top which allows the user to navigate to Directions.xml or Settings.xml page or Logout of the application.

Figure A.3 MainActivity.xml
A.1.4 AddActivity.xml

This screen allows the user to add information about any incident witnessed to the application.

![Add Activity XML Screen](image)

Figure A.4 AddActivity.xml
A.1.5 Settings.xml

This page allows the user to set a radius in miles as shown in Figure A.5. The user is allowed to view any updates posted by the users in that radius only. It also has a button which allows the user to logout of the application.

Figure A.5 Settings.xml
A.1.6 DirectionsSearch.xml

This page allows the user to search for a particular address location as shown in Figure A.6. It also has a button labeled as Load Directions which directs the user to the main screen of the application.

Figure A.6 DirectionsSearch.xml