SMARTPHONE-BASED CROWD SOURCING OF BICYCLE AND PEDESTRIAN CONFLICT DATA FOR TRANSPORTATION SAFETY ASSESSMENT

by

RAHUL ANKUSHRAO KAWADGAVE

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2016

Acknowledgements

Firstly, I would like to thank my advisor Dr. Taylor Johnson for the continuous support on my thesis project and motivation. I would like to thank you for your priceless patience and giving me independence to work and learn in course of this project. I would like to thank my thesis committee members Dr. W. Alan Davis and Dr. Ali Davoudi for their questions and reviews.

I would also like to thank Dr. Stephen P Mattingly from Department of Civil Engineering and Dr. Colleen Casey from College of Architecture, Planning and Public Affairs (CAPPA) for their assistance and guidance. I would also like to thank all my professors at UTA. I would also like to thank research team members Ziaur, Dian and Sunil for their help and contributions to build this project. I would like to thank all members from Verification and Validation for Intelligent and Trustworthy Autonomy Laboratory (VeriVITAL), Hoang, Luan, Omar, Prajakta, and Shafiul for their support and help. This research project is supported by Transportation Research Center for Livable Communities (TRCLC).

A special thanks to my father and my brothers Dynaneshwar and Mukesh. Without my brother Dnyaneshwar's support, I would not have possibly completed this degree. Also without my cousin Bhushan's encouragement and support, I would not have decided to pursue this degree. I am also very thankful to Sandip who has been a great help and support for me especially in last two years. I would also like to thank all my friends Aniket, Ashutosh, Chetan, Nachiket, Omkar, Pawan, Rahul, Rohan, Ronit, Sanil, Saurabh, Shashank, Sumet, Suyog, Tejas and many more who helped me every way possible during my stay at Arlington. Finally, I would like to thank everyone who supported me in last two years and I unfortunately missed here to mention specifically.

July 26, 2016

Abstract

SMARTPHONE-BASED CROWD SOURCING OF BICYCLE AND PEDESTRIAN CONFLICT DATA FOR TRANSPORTATION SAFETY ASSESSMENT

Rahul Kawadgave, MS

The University of Texas at Arlington, 2016

Supervising Professor: Taylor Johnson

This thesis presents the idea of mobile application development that aims at data collection for transportation safety assessment using cloud-based database. At present, transportation safety assessment relies only on data available from crash, accident reports. In combination with this crash data, it is possible that conflict data can become a very influential performance measure for transportation safety assessment. A transportation conflict happens when two parties cross a path and one party must take an action to avoid a collision or a crash. Such conflicts can occur between different transportation elements or parties like vehicle drivers, bicyclists, pedestrians. This project helps to collect the conflict data by developing smartphone application.

This application provides a way for crowed-sourced data collection. Nowadays bicyclists, pedestrians and vehicle drivers make use of various smartphone applications in the areas like transportation, navigation systems, maps, health and many more. It makes a smartphone application a straightforward and simple to make choice to collect crowed-sourced conflict data. Android application developed in this project provides simple user interface to select a location using Google Map view and then record a

conflict data by answering to a simple survey. Data derived from submitted survey is stored on Amazon Web Server database for further analysis and use.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Research Background and Motivation

Nowadays, vehicles are growing to have more involvement of different electrical and electronics (E/E) systems. E/E systems such as computing, communication devices are making advances towards making vehicles safer [1]. Largely, these safety systems added to vehicles also ensure transportation safety. This growing involvement of computing devices and E/E systems will make communication between vehicles and other transportation elements possible. There is different standards like IEEE 802.11p present already which will help to grow different applications and systems development in the area of vehicular communication using wireless and mobile technologies like Dedicated Short Range Communication (DSRC) [2]. This capability of communication between transportation elements will make the transportation safety assessment better. As these advancements in vehicles play important part in transportation safety assessment, other two parties in transportation pedestrians and bicyclists, also can play a very important role in transportation safety assessment. Especially nowadays when they are equipped with smartphones. These E/E systems and smartphones are examples of cyber-physical system (CPS). The system using smartphones with network connectivity used in this thesis is a representative distributed cyber-physical system (CPS)[3, 4, 5, 6, 7, 8] and other distributed CPS include for instance mobile swarm robotics applications, which have similar features to transportation systems [9, 10, 11, 12, 13, 14].

In the sense of public health and safety it is very important that, why and which transportation mode is selected by an individual. Transportation mode chosen may

include vehicle, bicycling, walking. Impression of safety level for an individual certainly influences the choice of transportation mode. Safety level can be determined using data available for transportation safety assessment; currently such available data is only from crash or accident reports. In addition, it is possible to collect conflict data using smartphone applications, which can be an input for transportation safety assessment. In addition, it is possible to use the data which can be the performance measure of safety and taken directly from crowed. Such data is possible to derive from conflicts that can occur; conflict happens in transportation is two parties cross a path and one party must take an action to avoid a collision or a crash. This conflict data when recorded can become a very influential performance measure of the transportation safety for most government agencies like United States Department of Transportation (USDOT) and public communities.

1.2 Objective

Transportation infrastructure for conflicts measurement requires real-time data from vehicles, pedestrians and bicyclists. Specifically this project aims at developing and application to capture data on conflicts experienced by pedestrians and bicyclists. This project will also determine effectiveness of using application to crowd-sourced conflict data. Project development involves identifying requirements of the user and develop the application for data collection. This smartphone application with data subscription service will help agencies and users access the collected conflict data.

1.3 Thesis Organization

Thesis documentation contains five chapters in total. Chapter 1 Introduction describes about research background and objective. Chapter 2 gives the overview of project. Chapter 3 documents all application requirements. Chapter 4 talks about overall

development and implantation process in detail; chapter 5 is the conclusion and some explanation on possibilities of future work using new systems and methods.

Chapter 2

Project Overview

2.1 System Overview

This project uses Google Android platform to build an application where using Google Map users can record a conflict at a particular location. The idea is to provide a user interface, which will ask user regarding the information of conflict using survey. Survey includes different questions like what was the type, location, date, time and severity of a conflict. Using Google Map interface user can search a location and record a conflict by answering survey questions. Locations of conflicts accurately recorded with GPS information from Google Maps. This application interfaces with Amazon Web Server to store the conflict data derived from survey in the mobile application. Dynamo DB is easy to use and simple to design NoSQL database service provided in the Amazon Web Services. In addition, Google cloud services provide better location search and secured Sign-In methods with Google client APIs for Android mobile phones.

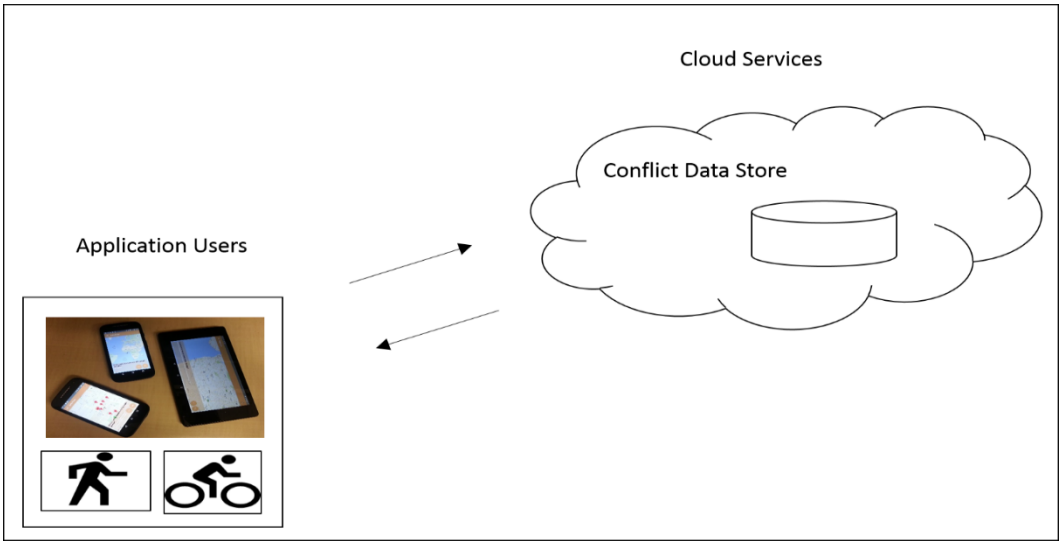

Figure 2-1 System Overview

2.2 Development Process

Figure 2-2 shows the process of this Android application development. Design considerations of different components included in the application come from inputs collected as part of this process. Chapter 3 describes design and development process in detail.

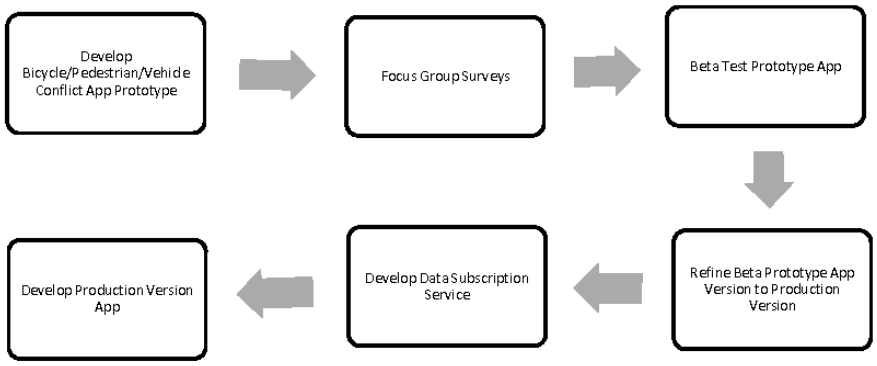

Figure 2-2 Development Process

2.3 Development Environment

This application development uses modern software process. Android smartphones are uses as hardware and software environment as Android has dominant market share. Android studio is an official integrated development environment (IDE) for Android platform. Application source code, documentation is managed using version control repository BitBucket. Cloud platforms used are AWS for cloud-based database and Google for various other services needed in application.

Chapter 3

Application Requirements

Requirements explained in this section are for two major sections of this project. First is mobile application that collects the conflict information and second is the cloud database system. In addition, there is two kinds of users for this application app user and end user. App user is smartphone application users like bicyclists and pedestrians. End users are government agencies who will use data from database. There are different functional and user interface requirements for this application. These different requirements are collection of inputs from two focus group surveys conducted with app users and end users. This section covers all requirements captured for mobile and cloud service for this project.

3.1 Functional Requirements

3.1.1 Application Users and Groups

This section documents the requirements related to mobile application users and groups. Research requires application users be divided in different user groups so that application usage for different scenarios can be evaluated. User groups are differentiated based on ability to show 1) daily reminder 2) notification for a conflict recorded in user's area 3) view conflicts recorded by user 4) view conflicts recorded by other users 5) download conflicts as csv/kml file. This section captures requirements in form of table as shown in Table 3-1. In general, user groups should have different roles admin and other. Group with role admin should have all capabilities and those with role other have different capabilities mentioned as per group in Table 1. Users in Group 0 are all with role admin and groups GROUP 1, GROUP 2, GROUP 3 are with limited capabilities role as other. Application should provide a way for dynamic assignment of group to a user such that

numbers of user in current number of users in each group. Users in GROUP 1, GROUP 2, and GROUP 3 at any instance of time should have nearly equal number of users.

Table 3-1 Application Users and User Groups

Group	Role	Show Reminder	Show Notification	View Own conflicts	View Other Conflicts	Download Conflicts as CSV/KML file
Group 0	Admin	Y	Y	Y	Y	Y
Group 1	Other	-	-	Y	-	-
Group 2	Other	Y	-	Y	-	-
Group 3	Other	Y	Y	Y	Y	-

3.1.2 Reminders and Notifications

Signed in users should be motivated using reminders and notification to record more conflict or crash information using this application. Different groups of users as discussed in section 3.1.1 have different reminders and notification strategies associated. Reminder should be a daily reminder “Would you like to report a conflict today?” if user clicks on the reminder, it should open the application to show applications home screen with map view. In every thirty minutes, if there is any conflict recorded in user’s area, then a user receives the notification. Current postal code area represents the current area. Message shown in notification should be, as “one /more conflict/conflicts was/were reported in your area”.

3.1.3 User Identification

Every conflict recorded should be associated with some unique identification that represents the user who recorded the conflict. Unique id can be something like email id.

3.1.4 Map View

Application provided map view that shows street view so that user can select different locations on the streets to record a conflict. Map view should have street name and postal code information displayed on it. There should be a way to move map view to user's current location.

3.1.5 Location Search

Application should provide user interface to search particular location using map view. User should be able to search a location by name, zip code.

3.1.6 Sign In

User signs in and share the email id with the application that will serve as user identification. Signing in will provide user's profile information like user name and email id.

3.1.7 Recording Conflicts

Recording conflict is accomplished using survey method. User answers some survey questions whenever user wishes to record a conflict. Survey answers is the information that represents the one entry in conflict database. Section 3.1.12 captures all requirements related to survey.

3.1.8 Recording Activity

User should be given a way to record an activity if user has does not wish to record any conflict in a day. User can record activity like 1) Bike 2) Walk 3) None at any time. Recording activity also is one of the questions in survey. Section 3.1.12 captures all requirements for recording activity as part of all survey questions.

3.1.9 Conflict Information

Conflict information that user should record is mainly derived from survey discussed in section 3.1.12. Table 3-2 describes conflict information fields and example value. Table 3-3 describes the data definition for the conflict records for fields described in Table 3-2.

Field	Value	Field	Value
useremail	abcd@gmail.com	Q51: Who are involved?	a
conflictdatetime	2016-06-08T03:10:00.000Z	Q52: Who is reporting?	b
intersection	N	Q53: Both parties traveling in the same direction?	b
segment	N	Q61: Vehicle Speed	a
latitude	32.73154419	Q62: Bicyclist Speed	N
longitude	-97.13376135	Q71: Distance of the Vehicle	N
StreetAddress	N	Q72: Distance of the Bicyclist	N
IncidenceZipcode	76013	Q73: Lateral Distance	a
Q1: Did you experience a conflict?	a	Q8: Timeforreaction	a
Q2: Was it a collision or a conflict?	b	severity	0
Q3: Activity Type (if Q1=b)	N	Q91: TripPurpose	
Q41: Activity Type (if Q2=a)	N	Q92: RoadCondition	
Q42: Injury Level	N	Q93: Comments	
Q43: Went to hospital by ambulance	N	Q94: HomeZipcode	
Q44: AdmittedToHospital	N		

Table 3-2 Survey Record

Variables	Definition/ Question	Examples/Answers
useremail	The original email address of users that he/she use in google play store account	abcd@gmail.com
conflictdatetime	Time of conlficts; Date format: Year-MM-DD and Time format: HH:MM:SS.00;	2016-06-08T15:25:00.000Z
intersection	Incident happened at an intersection location	Y: yes N: no
segments	Incident happened at a segment location	Yes
latitude	Latitude of incident	32.73154419
longitude	Longitude of incident	-97.11443905
StreetAdreess	Address	416 Yates street, Arlington, TX 76019
IncidentZipcode	Zip code of the incident location	76019
Q1	Did you experience a conflict?	a: Yes; b: No
Q2	Was it a collision or a conflict?	a: An actual hit/crash/collision; b: A conflict/near miss
Q3	In case of no conflict recorded (Q1=b), What type of activity did you do today?	a: Walk/Run; b: Bike; c: None
Q41	In case of collision (Q2=a), What type of activity did you do today?	a: Walk/Run; b: Bike; c: None
Q42	Please identify your injury level.	a: Major Injury (Disabling); b: Minor Injury (non-disabling); c: Minimal Injury (Possible abrasions and bruises); d: No Injruy (property Damage only)

Q43	Were you transported to hospital by an ambulance?	a: Yes; b: No
Q44	Did you got admitted to a hospital?	a: Yes; b: No
Q51	Who were involved?	a: Pedestrian and vehicle; b: Bicyclist and vehicle; c: Pedestrian and bicyclist
Q52	Who is reporting?	a: Bicyclist; b: Pedestrian
Q53	Were both parties traveling in the same direction?	a: Yes; b: No
Q61	Vehicle Speed: when traveling past a pedestrian/bicyclist	a: Very slow (<=10 mph); b: Slow (10-20 mph); c: Moderate (20-30 mph); d: Fast (30-40 mph); e: Very fast (>40 mph)
Q62	Bicyclist Speed: when traveling past a pedestrian	a: Slow (<=10mph); b: Moderate (10-20 mph); c: Fast (>20 mph)
Q71	Distance of the Vehicle: from a pedestrian/bicyclist	a: Greater than one car length (>20ft); b: Between half and one car length (10-20ft); c: Less than half car length (<10 ft)
Q72	Distance of the Bicyclist: from a pedestrian	a: Greater than one bike length (<10 ft); b: between half and one bike length (5-10 ft); c: less than half a bike length (<5ft)
Q73	Lateral Distance (if Q53=a): between	a: Less than 3ft; b: Greater than 3ft

	vehicle and bicyclist or between pedestrian and bicyclist	
Q8	(for all Q6 and Q7 except Q53-a) How much time did you have to take safety measure to avoid a crash?	a: More than enough time, able to think about and select from a variety of safe actions; b: Sufficient time, but made quick decision and acted; c: Barely enough time, only quick reactions avoided a crash
Severity of Conflicts	Severity categories of conflict based on the options selected in Q1-8; Severity is present in Red, Orange, yellow and Green color	Category A: is a serious incident in which a collision is narrowly avoided Category B: is an incident with significant potential for a collision where separation decreases and incident may result in a time critical response to avoid a collision. Category C: is an incident characterized by moderate time and/or distance to avoid a collision. Category D: is an incident with no immediate safety consequences but met the definition of a conflict such as encroachment of the space/area of a roadway surface designated for a single vehicle/person

Q91	What was the purpose of your trip?	a: Home-work-home trip; b: leisure/exercise; c: family errands; d: other, user specified
Q92	What was the road condition?	a: Dry; b: Wait; c: Other, user specified
Q93	Additional Comments	User comment box
Q94	Participants home zipcode	76010
N*	Option not present for participant	

Table 3-3 Survey Data Definition

3.1.10 Search Conflicts

Application should provide a way to search conflicts recorded by different users depending on the capabilities for each user group as mentioned in section 3.1.1. Possible filters for searching conflicts are as 1) user's own conflicts 2) all conflicts recorded in area by zip code 3) all conflicts recorded in area by distance in miles. Zip code and distance in miles are user-entered values.

3.1.11 Download Conflict Information

As per capabilities mentioned in section 3.1.1 for different user groups, users should be able to download the conflicts. Conflicts downloaded in the form of files such as CSV and KML. Download option gives a way to share conflict information with end users. CSV format makes it easy to share information in text format with end users, CSV file works with applications like Microsoft Excel for better view and analysis of conflict data. Applications like Google Earth and Google Maps can view KML files. Requirements for conflict information is captured in section

3.1.12 Survey Questions and Flow diagram

User needs to answer survey to record any conflict or activity. Conflict Information discussed in section 3.1.9 is from this survey questions and answers. Figure 3-1 captures flow for survey and Table 3-4 captures all questions and answers for the survey.

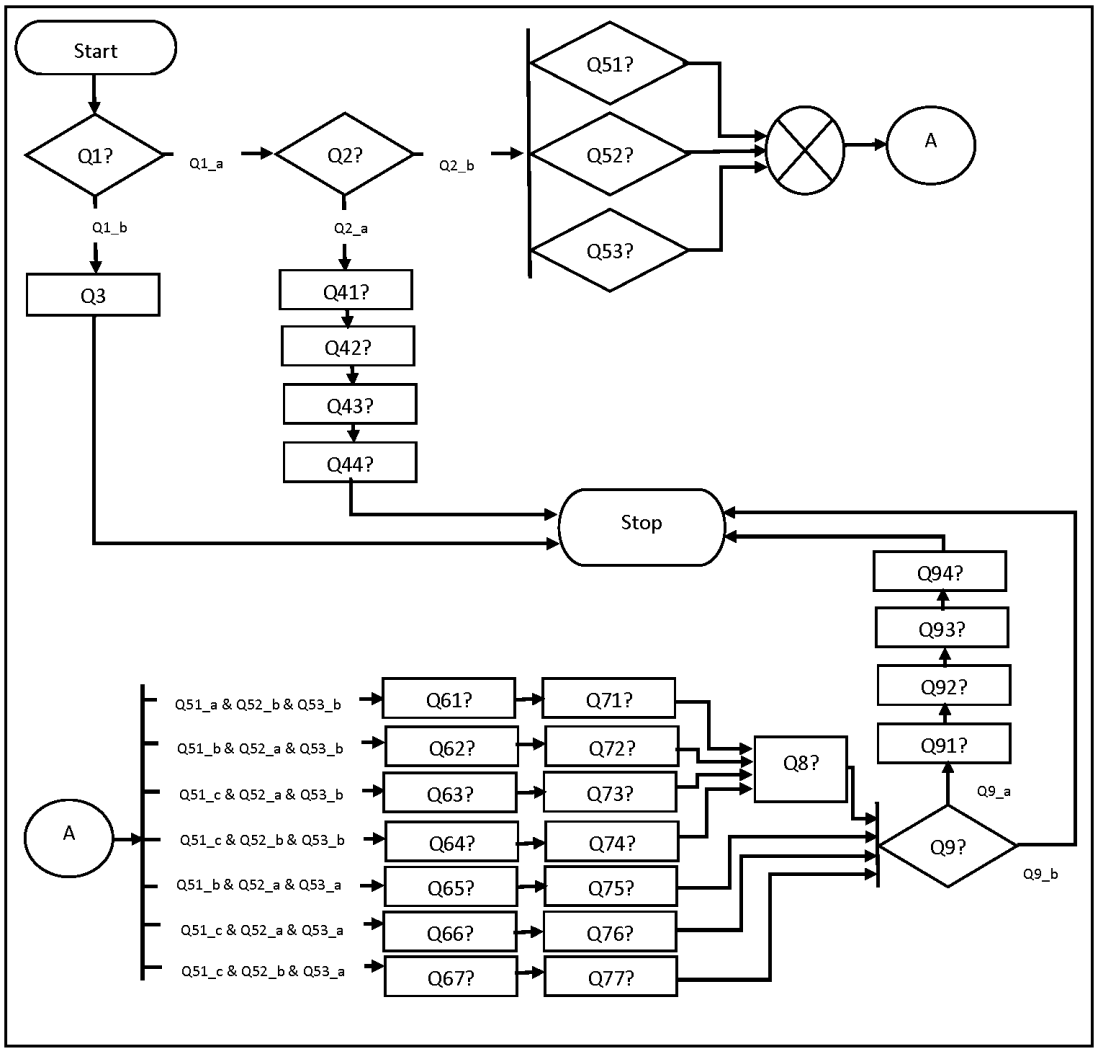

Figure 3-1 Survey Flow Diagram

Question Id	Question	Answer Id	Answer
Q1	Did you experience a conflict?	Q1_a	Yes
		Q1_b	No
Q2	Was it ___?	Q2_a	An actual hit/crash/collision
		Q2_b	A conflict/near miss
Q3	What type of activity did you do today?	Q3_a	Walk/Run
		Q3_b	Bike
		Q3_c	None
Q41	What type of activity did you do today?	Q41_a	Walk/Run
		Q42_b	Bike
		Q43_c	None
Q42	Please identify your injury level	Q42_a	Major Injury (Disabling)
		Q42_b	Minor Injury (non-disabling)
		Q42_c	Minimal Injury (Possible abrasions and bruises)
		Q42_d	No Injury (Property Damage Only)
Q43	Were you transported to hospital by an ambulance?	Q43_a	Yes
		Q43_b	No
Q44	Did you got admitted to a hospital?	Q44_a	Yes
		Q44_b	No
Q51	Who were involved?	Q51_a	Pedestrian and bicyclist
		Q51_b	Bicyclist and vehicle

		Q51_c	Pedestrian and bicyclist
Q52	Were you a__?	Q52_a	Bicyclist
		Q52_b	Pedestrian
Q53	Were both parties travelling in the same direction?	Q53_a	Yes
		Q53_b	No
Q61	At what speed did the vehicle travel past you?	Q61_a	Very Slow (<=10 mph)
		Q61_b	Slow (10-20 mph)
		Q61_c	Moderate (20-30 mph)
		Q61_d	Fast (30-40 mph)
		Q61_e	Very Fast (>40 mph)
Q71	What was the distance between the conflicting vehicle and you?	Q71_a	Greater than one car length (>20 ft)
		Q71_b	Between half and one car length (10-20ft)
		Q71_c	Less than half car length (<10ft)
Q62	At what speed the vehicle traveled past you?	Q62_a	Very Slow (<=10 mph)
		Q62_b	Slow (10-20 mph)
		Q62_c	Moderate (20-30 mph)
		Q62_d	Fast (30-40 mph)
		Q62_e	Very Fast (>40 mph)
Q72	What was the distance between the conflicting vehicle and you?	Q72_a	Greater than one car length (>20 ft.)
		Q72_b	Between half and one car length (10-20 ft.)
		Q72_c	Less than half a car length (<10 ft.)

Q63	At what speed you were riding?	Q63_a	Slow (<=10mph)
		Q63_b	Moderate (10-20 mph)
		Q63_c	Fast (>20 mph)
Q73	What was the distance between you and the pedestrian?	Q73_a	Greater than one bike length (<10ft)
		Q73_b	Between half and one bike length (5-10ft)
		Q73_c	Less than half a bike length (<5ft)
Q64	What was the speed of the bicycle when it travelled past you?	Q64_a	Slow (<10 mph)
		Q64_b	Average (10-20 mph)
		Q64_c	Fast (>20 mph)
Q74	What was the distance between you and the conflicting bicyclist?	Q74_a	Greater than one bike length (<10ft)
		Q74_b	Between half and one bike length (5-10ft)
		Q74_c	Less than half a bike length (<5ft)
Q65	What was the speed of the vehicle while overtaking you?	Q65_a	Very Slow (<=10 mph)
		Q65_b	Slow (10-20 mph)
		Q65_c	Moderate (20-30 mph)
		Q65_d	Fast (30-40 mph)
		Q65_e	Very Fast (>40 mph)
Q75	What was the lateral distance between the vehicle and you?	Q75_a	Less than 3ft
		Q75_b	Greater than 3ft
Q66	What was your speed when overtaking the	Q66_a	Slow (<10mph)
		Q66_b	Average (10-20 mph)

	pedestrian?	Q66_c	Fast (20> mph)
Q76	What was the lateral distance between you and the pedestrian?	Q76_a	Less than 3ft
		Q76_b	Greater than 3ft
Q67	What was the speed of the bicyclist when overtaking you?	Q67_a	Slow (<10mph)
		Q67_b	Average (10-20 mph)
		Q67_c	Fast (20> mph)
Q77	What was the lateral distance between you and the bicyclist?	Q77_a	Less than 3ft
		Q77_b	Greater than 3ft
Q8	How much time did you have to take safety measure to avoid a crash?	Q8_a	More than enough time, able to think about and select from a variety of safe actions
		Q8_b	Sufficient time, but made quick decision and acted
		Q8_c	Barely enough time, only quick reactions avoided a crash
Q9	"Thank you for using the app." Would you like to answers some additional questions for extra points?	Q9_a	Yes
		Q9_b	No
Q91	What was the purpose of your trip:	Q91_a	Home-work-home trip
		Q91_b	Leisure/Exercise

		Q91_c	Family errands
		Q91_d	Other, please specify:
Q92	What was the road condition?	Q92_a	Dry
		Q92_b	Wet
		Q92_c	Other, please specify
Q93	Additional Comments:	N/A	
Q94	Your Zip code	N/A	

Table 3-4 Survey questions and Answers

3.1.13 Local Storage

Application requires showing images and GIFs to user to make user interface more intuitive. Images and GIFs are stored on internal storage to application.

3.1.14 Rendering Images and GIFs

Application requires showing images and GIFs to user to make user interface more intuitive.

3.1.15 Interface to Cloud Database

Conflict data is stored on cloud database in this application. Application manages create, read, write, update operations on this remote database.

3.2 User Interface Requirements

3.2.1 Home Screen

Application provides full screen map view to the user. Map view also shows conflicts using markers with different color codes as mentioned in Table 3-2.Color codes represent different severity levels.

3.2.2 Location Search

Application should provide user interface to search particular location using map view. User should be able to search a location by name, zip code.

3.2.3 Recording Conflict and Activity Using Survey

User has the ability to record a conflict using survey questions mentioned in section 3.1.12.

3.2.4 Search Conflicts

User should have user interface that provides menus options to search conflicts by zip code or distance in miles. User should be able to see conflicts recorded by user.

3.2.5 Data Subscriptions Service

Application will have user interface to show all conflicts by search filter in text format in the list view sorted by time. Application should provide interface to access information in CSV [15], KML [16] file formats, to download and share.

Chapter 4

Application Development

This section describes the development of Android application. It is a result of different inputs collected from brainstorming sessions, initial requirements, feedback from prototype testing and two focus group surveys conducted. Development process from includes steps of development from prototype to production version.

4.1 Development of Prototype

This section documents the application prototype development and testing carried out to receive initial feedback from students at UTA. Porotype developed has the simple user interface with map view and survey for students to record conflict information. Recorded data downloaded as CSV format file directly from AWS Dynamo DB service.

4.1.1. Development of Simple User Interface

This stage of development includes designing and implementing simple user interface that gives user a way to record data using map view and survey interface. Cloud interface developed updates this data to AWS Dynamo DB database [17]. Figure 4-1 shows the user interface created in this stage. As much as possible application screen flow and menu options give users to receive feedback from students to get more updates to incorporate in production version of the application.

Prototype user interface enables user to record a conflict with information like description, type, date, time, location and severity of the conflict using map control on app home screen. Each conflict record then showed on a map with all information associated with it.

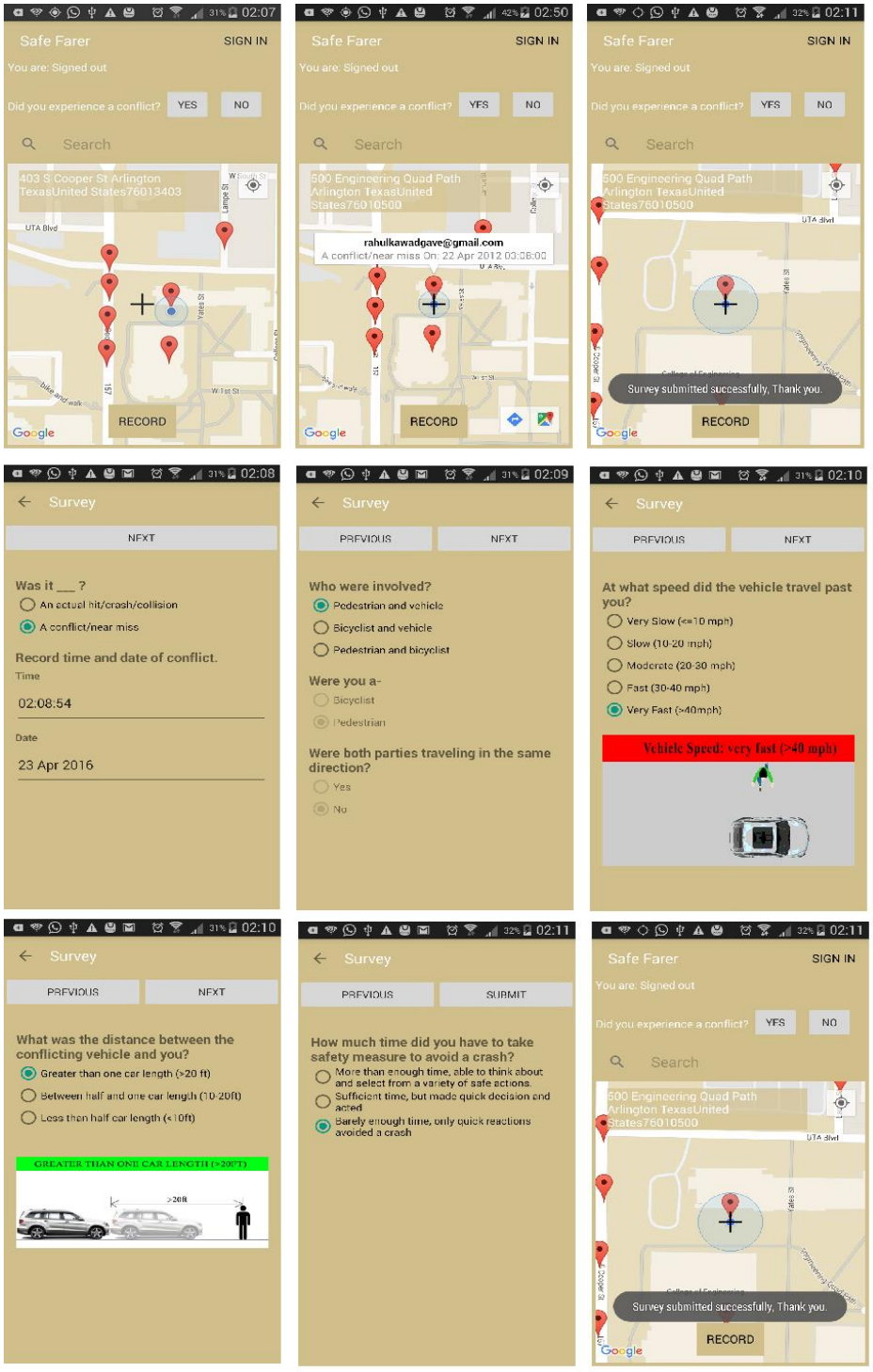

Figure 4-1 Prototype Version User Interface

4.1.2. Testing and Results

Prototype application made available to students for download and use [18] on Google Play Store. Figure 4-2 shows the database snapshot for recorded data by students as part of prototype testing. Recorded data in prototype stage in not fully as per the requirements. Prototype app represents the idea of application to focus group and student members.

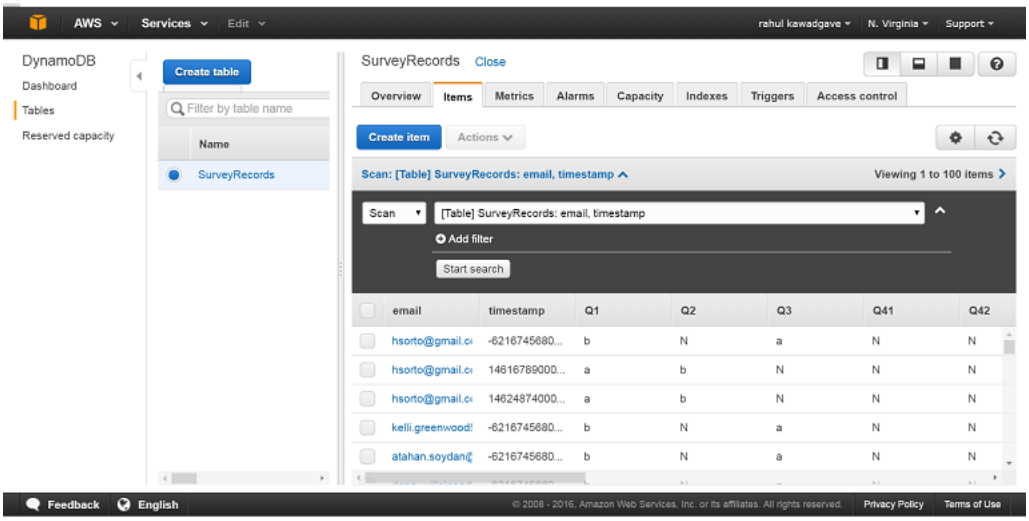

Figure 4-2 Prototype Version Cloud Database View

4.2 Software Architecture

This software architecture contains two main components mobile application and cloud services. Mobile application will collect the required data from user inputs as per the requirements and upload data to cloud based database. Mobile application also interacts with cloud based database service and query data as and when required. Interaction includes querying database with different search queries generated as per the requirement by user. Application also uses cloud services by Google. Mobile application contains user interface and interface to cloud based database service. User interface includes map view, survey user interface, list views, menu options for different search

that user can initiate. Interface to cloud based database service is using Amazon Dynamo DB mapper package [19]. Dynamo DB mapper provided simple and easy way access cloud based database in AWS. Mobile Application also connect to Google cloud to use services like Google Maps [20], Sign in [21], Location APIs [22]. AWS cloud database contains tables in created to store conflict information, user group related Information. Google client APIs provides way to connect to Google cloud services. Figure 4-3 shows the overview of the software architecture with different components. Section 4.3 describes list of components in detail.

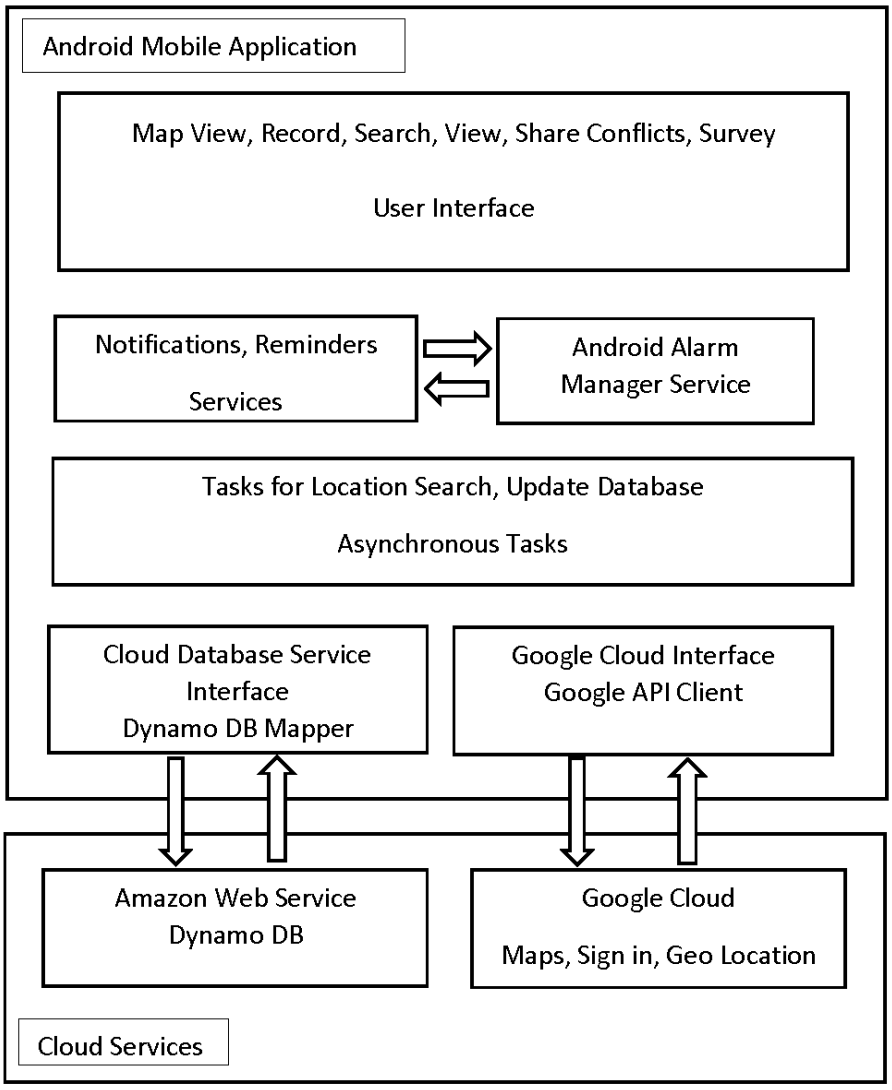

Figure 4-3 High Level Block Diagram of Mobile Application

4.3 Detailed Design and Implementation of Mobile Application

4.3.1 Location Information

Application uses different location related APIs [22]. Functionalities like record search, notify, remind conflicts to user make use of user's location and location services. In this case, there is different methods that provide different location information that this

application requires. This section covers all such methods, ways used in different modules of this application.

4.3.1.1 Knowing Last location using Google play services API

Google play services API client instance gives the ability to application using location services API [22]. Developers documentation online provides all information related on how to use Google Play Services for location information. This application in particular uses location services API to know the users last location. User's last location is required in this application in functionalities like notify user for conflicts in current area of zip code and search conflicts by distance user's current location. This APIs provides location in terms of geographic co-ordinates latitude and longitude.

4.3.1.2 Street Address Using Geocoder Class

Geocode class is part of Android framework location APIs [23]. This class provides two important functions 1) Geocoding 2) Reverse geocoding. Geocoding is process of converting a street address to latitude and longitude i.e. geographical co-ordinates. Contrary, reverse geocoding converts geographical co-ordinates to street address. This application uses reverse geocoding to display current address on home screen when user moves the map view to move to any particular location. Figure 4-4 shows the screenshot.

Figure 4-4 Geocoder View

4.3.1.3 Place Autocomplete for Searching Location by Name

Place autocomplete service in Google Places API provides the list of places in return to user's search query [24]. This application uses the option launching autocomplete activity using intent. Activity returns the results to parent activity using Place is an object that provides information like geographical co-ordinates. Figure 4-5 shows the place autocomplete activity with list of placed searched by user.

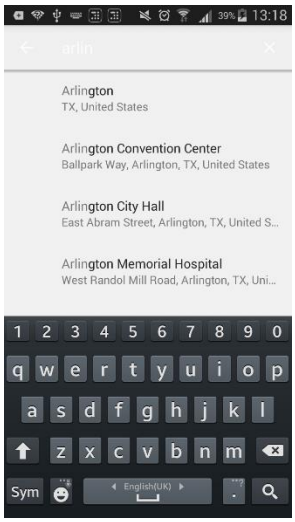

Figure 4-5 Place Autocomplete Activity

4.3.1.4 My Location Button

Google Maps for Android provides in-built feature, which helps user to move map view to its current location [25]. Figure 4-6 shows the button in right-top corner just below the search button.

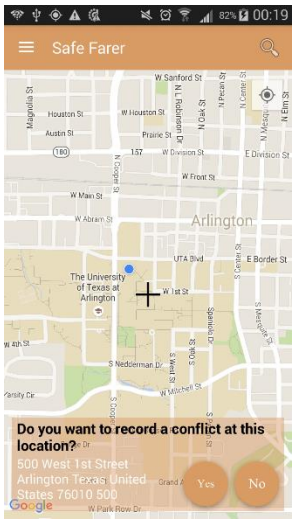

Figure 4-6 My Location Button

4.3.2 Google Sign In

User identification in this application uses Google Sign-In for Android devices [21]. User with Google registers every Android device. Using Google Play Services API client in Android it is possible to integrate Google Sign-In feature to any application [21].

4.3.2.1 Integrating Google Sign-In to application

This process simple and online help at Google developer's site makes it even easier [21]. Google services provide a configuration file that we need to integrate to application to get access to Google Sign-In access to the application. Figure 4-7 shows the sample configuration file used in this application. This file contains information used to create client id with Google for this project like API access related keys and authentication information. This file is located at folder app/main/__.

Figure 4-7 Google Sign-In API Key Configuration

4.3.2.2 Security

Using Google Sign-In for Android provides automatic quick and secure registration system used by various devices and users.

4.3.2.3 Profile Information

Google Sign-In for Android provides access to users profile information [26]. This application uses email id as user identification parameter for each conflict recorded and shows user's full name in the user interface shown in Figure 4-10. Table 4-1 also provides screen flow that shows sign in user interface in the application.

Table 4-1 Google Sign-In View

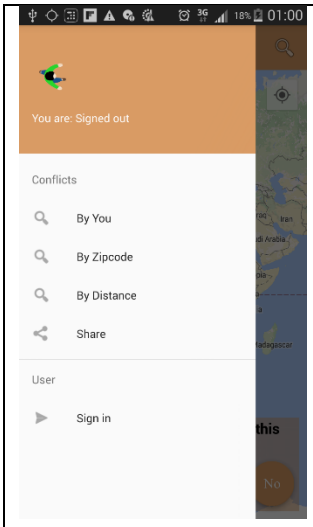	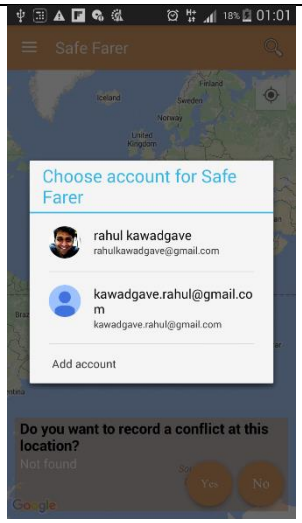	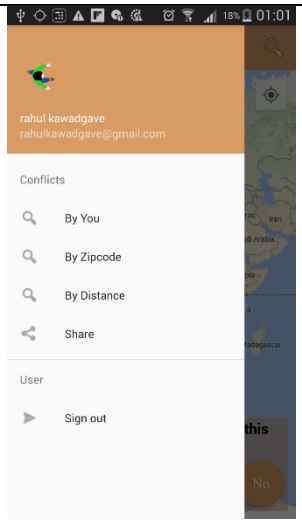
Figure 4-8 Figure Signed Out	Figure 4-9 Sign In Options	Figure 4-10 Signed In

4.3.2.4 Silent Sign-In

Silent Sign-in feature enables application silently sign in using previously granted authorization [27]. Whenever user launches the application silent sign-in feature allows user to sign in immediately and retrieve profile information. If there is no user authorization previously done on this device then user application provides sign-in

options again to user.

4.3.3 Google Maps

Google Maps Android APIs make it is possible to add Google Maps to Android application [20]. Google Maps provides way to add markers and shapes to map view. Google documentation for developers provides online systematic help to add Google map view to Android application. In this application map, view provides user an access to record a location and view for recorded conflicts using markers. Map view also serves as home screen of the application.

4.3.3.1 Configuration

Google Maps in Android use Google Maps APIs [20]. Google Maps APIs in Android uses Google Play Services SDK. Installing and configuring Google Play services SDK is provides this access to application. Application should be registered in Google API console to get Google API key that will added to Android project. Google Services and Google Maps APIs require Android application to have some permissions as mentioned in section. In this application, all required procedure to get configuration uses Google Developers online help.

4.3.3.2 Home Screen

Map view serves as application home screen as shown in Figure 4-11. View provides a way to record a conflict at a particular location and search button on top-right corner used to search any location by name, zip code, street name and address. Map view also provides my location button to center of the map view to the current location.

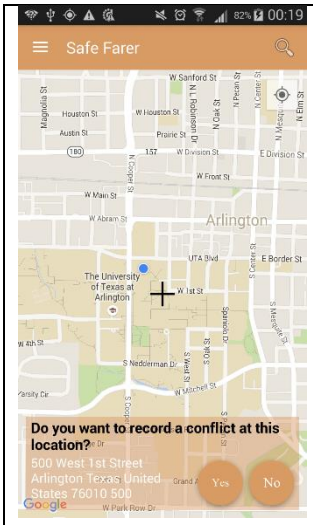

Figure 4-11 Home screen

4.3.3.3 Map View Showing Search Results

Map view also is view for showing search results. User can search conflicts using primarily three filters using menu options and results are as shown in Table 4-2.

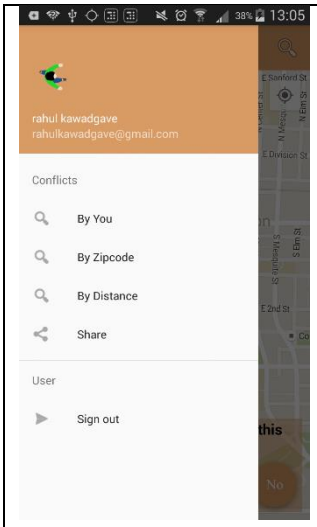	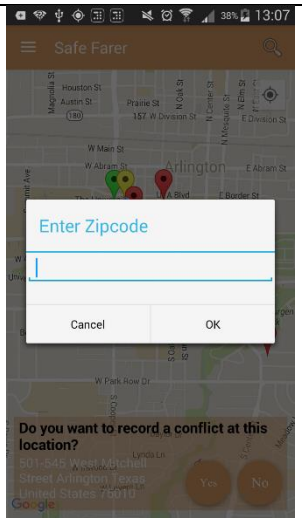	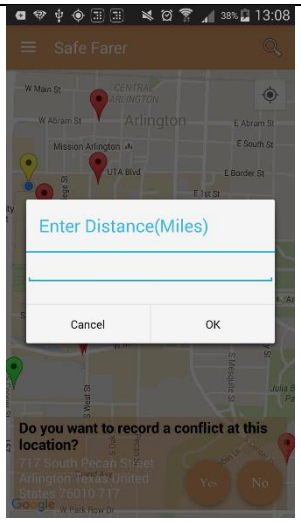
Figure 4-12 Search Menu Options	Figure 4-13 Enter Zip code User Input	Figure 4-14 Enter Miles User Input
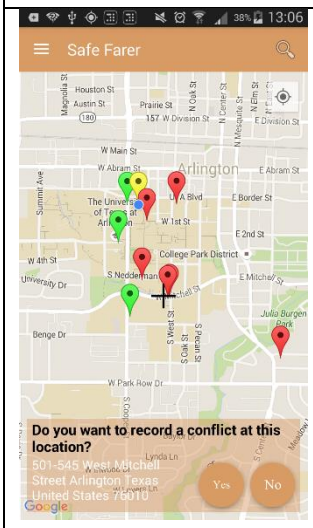	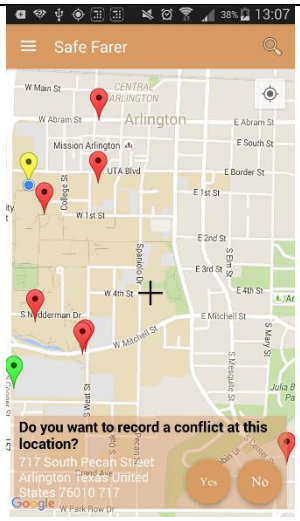	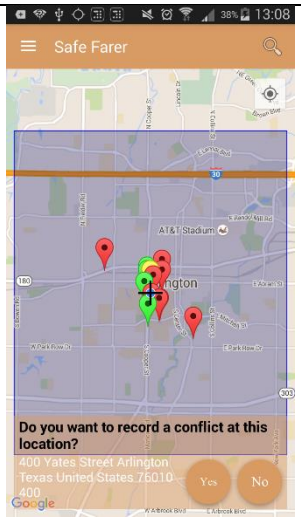
Figure 4-15 User Recorded Conflicts	Figure 4-16 Search by Zip code	Figure 4-17 Search by distance

Table 4-2 Search Results

4.3.3.4 Drawing On The Map View

Markers show the single location on map view. Markers have an option of drawing with different colors [28]. Figure 4-18 shows different markers drawn with info windows. Different colors indicate severity levels of conflicts recorded. MarkerOptions[29] class in Android provides functionality to state the position in latitude and longitude, icons with colors, title and small snippet as shown in Figure 4-19.

Drawing shapes in on map is also possible and rectangle shown in Figure is drawn using addPolygon method [30]. PolygonOptions[31] class is used to specify different options like color to fill, stroke width, and location details in list of latitude and longitude points for a polygon to be drawn. Width and height for drawing rectangle in the map uses simple calculation considering one latitude [32] or one longitude [33] degree equal to 69 miles.

Figure 4-18 Drawing Markers

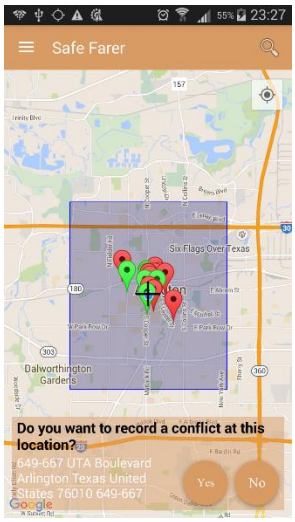

Figure 4-19 Drawing Polygons

4.3.3.5 Google Map Camera Animation

Google Maps Android API also provides map animations using which map view adjusts to different zoom levels or tilts and panned [34]. APIs provide way to animate the camera to specific zoom levels as per the requirement. LatLang[35] and LatLangBounds [36]classes are used to adjust and provide different set of points on map for animations.

4.3.4 User Manager

User Manager in this application works with Dynamo DB Manager and helps to handle requirements mentions as part Application users and user groups for the application.

4.3.5 Cloud Based Database

This application uses cloud based database service to store all conflict data recorded by user and all application users and groups information. Figure 4-10 shows the applications components and interface to cloud in detail. Cloud database used in this application is Dynamo DB. This section describes about configuration and interfacing with Dynamo DB from Android application.

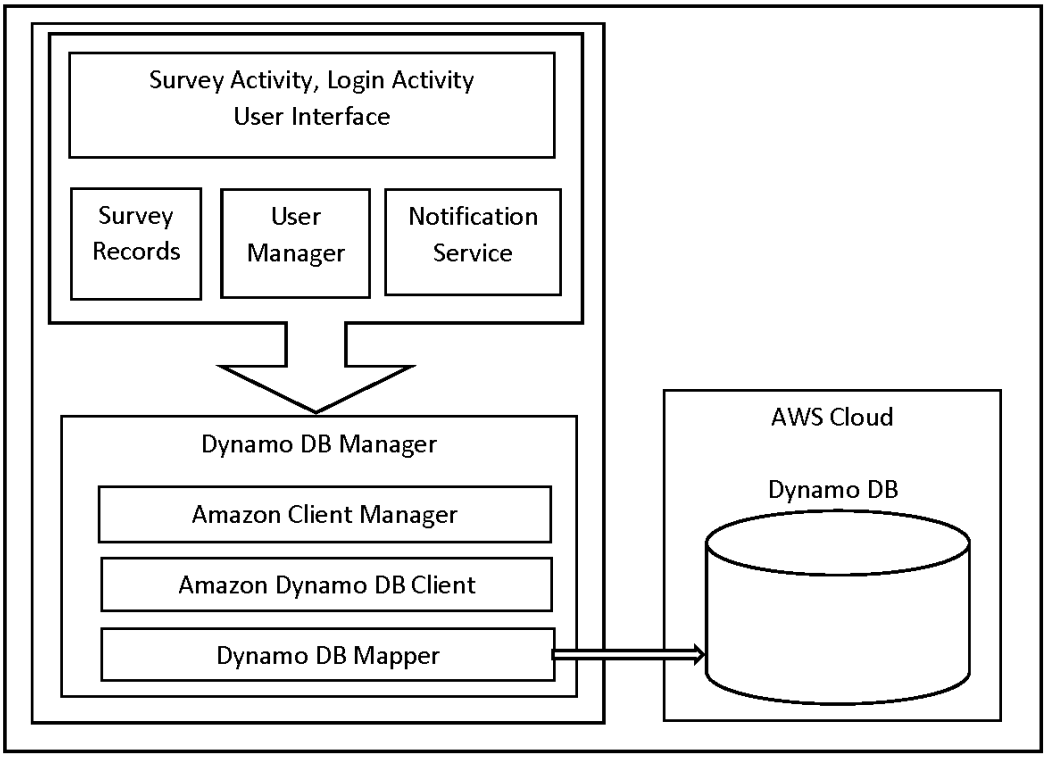

Figure 4-20 Architecture of Interface to Cloud

4.3.5.1 Amazon Dynamo DB

Dynamo DB is NOSQL database service and it is a simple way get fully managed database running in short time. As this application requires secured and easy to handle database service Dynamo DB provides all of it. Amazon provides different access controls to each table created in Dynamo DB. Authentication and access control for Amazon Dynamo DB uses credentials as part of AWS Identity and Access Management [17].

4.3.5.2 Dynamo DB as NoSQL Database

Dynamo DB is NoSQL database as described in Amazon documentation [37]. It is faster, easy to develop non-relational database. NoSQL database provides high

performance needed for real-time applications and on devices like those that mobile phones used in this application. APIs to store and retrieve in Dynamo DB are object based that means it makes development easy for application developers. Data model is key-value based and values retrieved in the form of JSON, XML. Data model is schema-less in Dynamo DB, which makes easy to store records derived from survey that takes different paths, not every record stored in database has data for every key in the database.

4.3.5.3 Database Tables

Application stores conflict data and user group related information in this cloud database. There are three different tables created to handle this requirement. Access to these tables uses component Dynamo DB Manager. This interface enables application with create, read, write, update different records in these tables. Figure 4-21 shows the databases created in Dynamo DB. Database tables have all information described as part of requirements of this application.

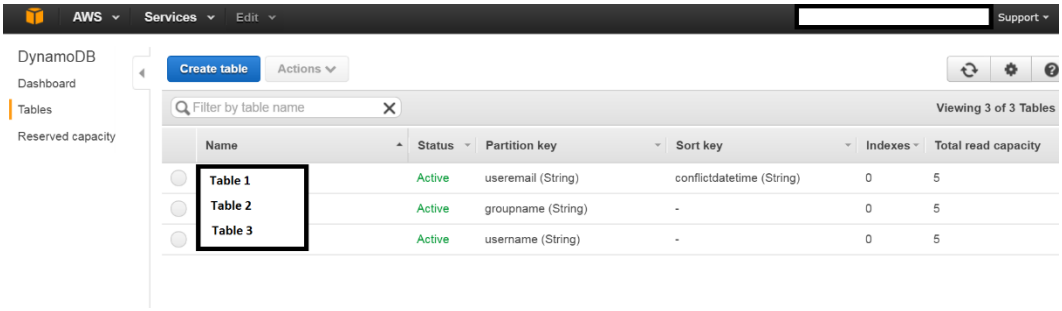

Figure 4-21 Dynamo DB tables.

4.3.5.4 Dynamo DB Manager

Dynamo DB Manager provides service to different components in this application to provide access to database. Survey records uses this interface to store conflict information in database, whereas notification service and user manger uses application

user group's related information from database. Amazon Client Manager and Dynamo DB Client components shown in figure provide the authentication related service. Dynamo DB Mapper class is the interface to Dynamo DB from application. Mapper provides direct access to DB from application. It provides different operations like save delete, scan on tables.

4.3.5.5 Database Operations

Searching conflict information from database with filters like different zip codes, data and time intervals is required. Dynamo DB provides different way to scan and query the database using Dynamo DB Mapper class. This application uses the filter expressions to search different conflict records from database. Table 4-3 lists down different search queries used in this application.

Table 4-3 Search Queries to Dynamo DB

	Search Scenarios	Parameters Used from Database
1	Search records by specific user	username
2	Search records by specific zip code location	sourcezipcode
3	Search location in last 30 minutes in current zip code area	confictdatetime, sourcezipcode
4	Search records by distance miles / by latitude longitude bounds.	Latitude, longitude

4.3.6 Survey Design

Conflict information recorded by application with user inputs using survey questions as mentioned in the requirements. SurveyRecords class represents the one record generated after user entered answers to survey questions in one flow.

SurveyRecord converted to database mapper class used by DynamoDBManager class. Survey user interface creates dynamic screens generated as survey flow goes in particular path with each question answered. Android provides Fragment [38] view, which holds a price of user interface. One Fragment viewed as part of an Activity in Android. Survey uses one Activity and list of fragments to show questions in on that Activity. Using FragmentManger [39]. It is possible to create a stack of fragments in Activity in which supports push and pop operations. Using this stack, survey UI implements next and previous operations. Figure 4-22 shows one such survey flow in the application, which records conflict information.

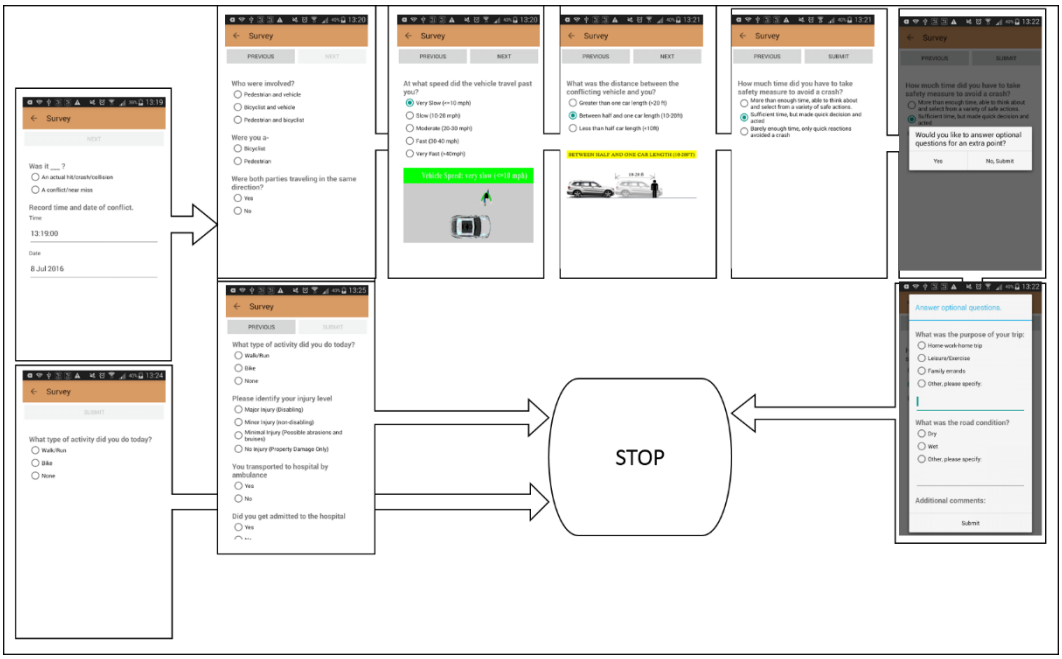

Figure 4-22 Survey Design User Interface

4.3.7 User Interface Views and Controls

This section described the view and controls available in Android and used to implement in this application. Table 4-4 lists the views and controls used to build different components in this application.

Table 4-4 User Interface Controls

	View/Control	Components
1	Activity	Home screen, Survey screen, Share screen
2	Fragment	Survey questions, Map View, GIF Player, Location Search view
3	Alert Dialogs	Popup dialogs, Error Dialogs, User Inputs, and Confirmation messages.
4	Snack Bars	Showing alerts, messages to user.
5	Image View	Survey screens to show extra information
6	Navigation Drawer	Navigation drawer is the navigation options displayed on the left side of the screen
7	Recycler View	List view on the share screen.
8	Date Picker	Date picker view in survey screens.
9	Time Picker	Time picker view in survey screens.

4.3.8 Notification Service

Notifications and reminders as per requirement use different Android framework utilities provided fir implementation. Figure 4-23 shows details on design how notification and reminders generated in this application. Application requires two notifications one for every thirty minutes and one for once a day.

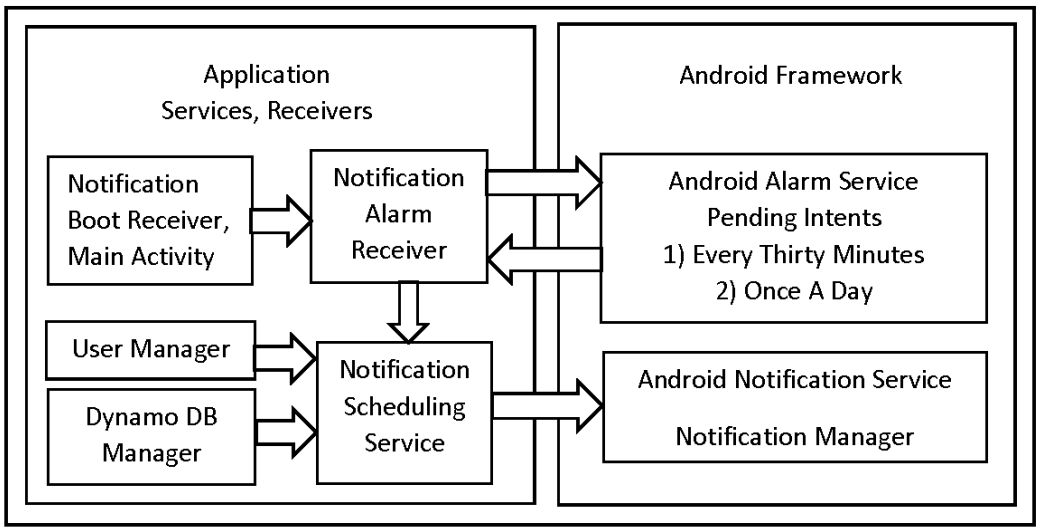

Figure 4-23 Notification Service Architecture

4.3.8.1 Notification Alarm Receiver

Notification alarm receiver interacts with Android Alarm Manager [40] in Alarm service and registers two alarms using Pending Intents [41] as shown in figure. Application components boot receiver and launcher activity of application register alarms using pending intents to Android Alarm receiver. Pending Intent is a way to communicate with exiting services in Android framework.

Pending intents registered with Android alarm service will call the handler in application's notification alarm receiver. This receiver is type of WakefulBroadcastReceiver [42] in Android. This will wakeup device whenever alarm triggers off and even if device is in sleep mode.

4.3.8.2 Notification Boot Receiver

Notification boot receiver with the help of notification alarm receiver registers the required alarms with framework in case of device reboots. Boot receiver implements handler, which triggers in case of device reboots. This step is important as application

exits when device shuts down and user may not launch the application every time after reboot to register the alarms required [43].

4.3.8.3 Android Alarm Service

Scheduling repeating alarms with Android alarm is easy procedure [44]. Alarm manager APIs are simple to use. There are several ways and types to register alarms on Android device. Alarms type used in this application are real time wakeup as application requires to show notifications even when device is in sleep. Repeating alarms are in Android can be exact or inexact. Inexact alarms used in this case allows Android framework to manage power better by aggregating requests from different applications and wakeup device at single point in time to serve all together. Setting inexact repeating alarm type will also help and not burden cloud-based service. This will also help to avoid the concurrency issues with cloud-based services largely.

4.3.8.4 Notification Scheduling Service

Notification scheduling service will handle the requests from Notification Alarm Receiver to actually queueing notification with Android framework [45]. This service runs in the background. Service handles the intent from alarm receiver every thirty minutes and once a day. It takes decision whether to show a notification and reminder depending on application's current user and user group. Service uses the user manager and Dynamo DB Manager to get information about capabilities of current user. Figure 4-25 shows the notification when for any recorded conflict in users' current zip code area. Figure 4-24 shows the notification received by user to motivate user to record any new conflicts that user encountered.

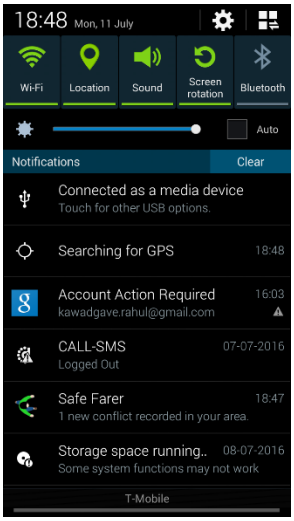

Figure 4-24 Notification View

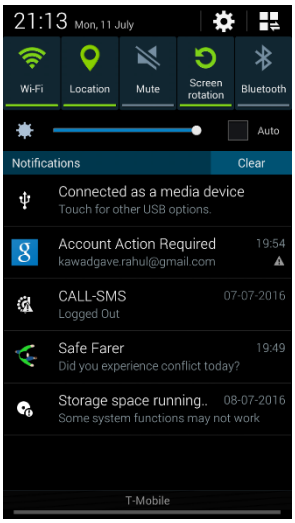

Figure 4-25 Reminder View

4.3.9 Asynchronous Tasks

Asynchronous tasks in Android gives way to handle background operations and then share the result with UI threads [46]. This application uses AysncTask to perform different network operations and update the results in UI threads. Figure 4-26 shows the executions flow of asynchronous tasks.

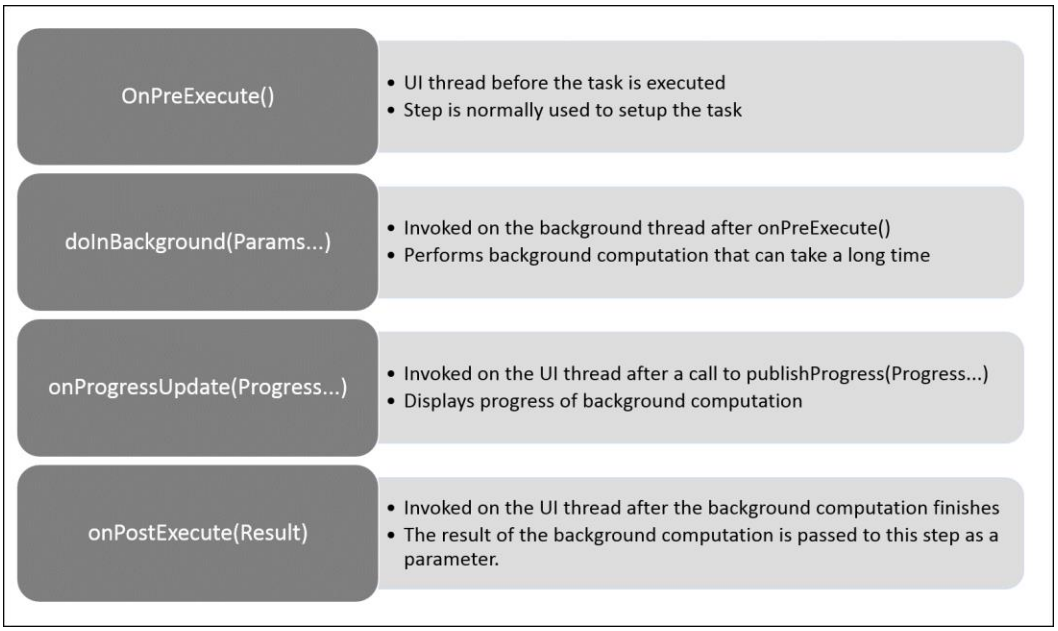

Figure 4-26 AsyncTask Architecture

	Task	Description
1	UpdateUserGroupTask	Updates application users and user groups information to the cloud based data store
2	UpdateLocationTask	Updates location information on home screen
3	SearchDatabseTask	Handles data search queries to cloud database
4	UpdateDatabaseTask	Updates database with new information likes inserting new records.

Table 4-5 List of AyncTasks in Application

4.3.10 File Writers

Application provides way to download and share conflict records using file formats CSV and KML for end users. Location of files created is the external directories. This directory temporary and external to application. In this case, files created are shared

with other applications like Gmail, Dropbox when user selects share option. System takes care of deleting files in external directory when user uninstalls the application or when system requires more space. User has an option to choose download format as CSV or KML. CSV format file is created as per the columns and rows specified in the requirement. KML file format opens in applications like Google Earth and application implements the KML writer to create this format of file.

4.3.11 Android Application Manifest

Application manifest is the file AndroidManifest.xml that is in the root directory of application and it shows the required information of the application [47]. Android platform need this file for every application, which provides in essential information about the application. Information includes like Java package name, application components, and permissions given to the application. Figure 4-27 shows the file uses in this application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.verivitallab.safarer"
    android:versionCode="7"
    android:versionName="1.6">

    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.READ_PROFILE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="com.android.alarm.permission.SET_ALARM" />

    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.READ_PROFILE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:name="android.support.multidex.MultiDexApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar" >
```

```
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

       <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".GoogleMapsActivity"
            android:label="@string/title_activity_google_maps" >
        </activity>
        <activity
            android:name=".ScreenSlideActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:hardwareAccelerated="false"
            android:label="@string/title_screen_slide"
            android:parentActivityName=".MapViewFullScreenActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MapViewFullScreenActivity" />
        </activity>

        <receiver android:name=".NotificationAlarmReceiver" />
        <receiver
            android:name=".NotificationBootReceiver"
            android:enabled="false" >
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>

        <service android:name=".NotificationSchedulingService" />

        <activity
            android:name=".ConflictRecyclerViewActivity"
            android:label="@string/title_activity_conflict_recycler_view"
            android:parentActivityName=".LoginActivity"
            android:theme="@style/Theme.AppCompat.Light.NoActionBar" >

          <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.verivitallab.safarer.MapViewFullScreenActivity" />
        </activity>
        <activity
            android:name=".MapViewFullScreenActivity"
            android:label="@string/title_activity_map_view_full_screen"
            android:launchMode="singleTask"
            android:theme="@style/Theme.AppCompat.Light.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figure 4-27 App Manifest

4.4 Data Subscription Service

Data subscription service in this application provides interface for users to the data recorded using application in the cloud based data store. Application provides interface to data for users two ways 1) Visualization to data using Google Maps, Google Earth application 2) Download and email data as file with format as CSV or KML. Users with permission as defined in application user groups will be able to download the data and share. It is also possible to access the data from AWS Dynamo DB service. As shown in Figure 4-29 user can filter the data and export data to CSV file, this access is only given to limited users.

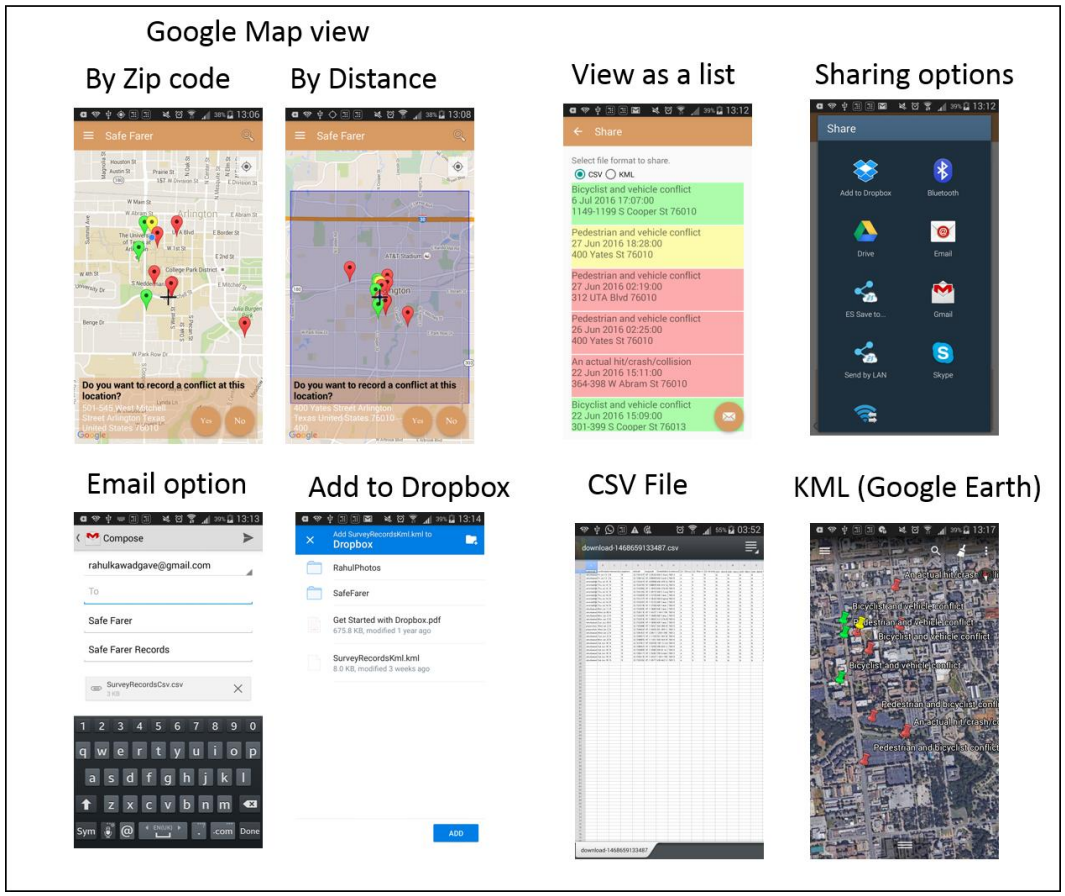

Figure 4-28 Data Subscription Service User Interface

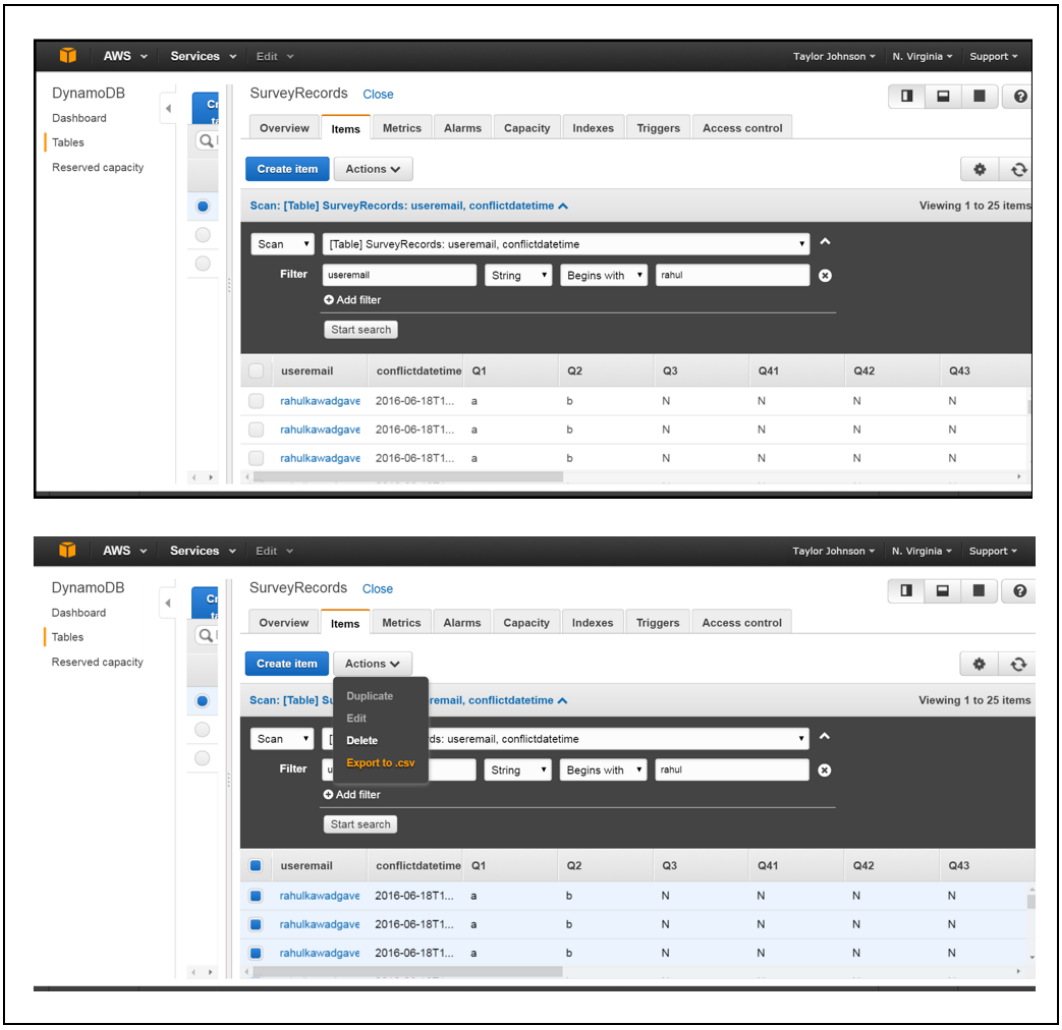

Figure 4-29 Data Subscription Cloud Database View

4.5 Verification and Validation

Verification involved activities like ensuring product is in line with requirements specifications or not. It includes reviewing requirements and updating requirement to include feedbacks from different elements like internal and external reviewers. Validation involves testing the actual product works correctly as per the requirements and design.

This application developed as per the process defined in the Figure 2-2. Verification for this product includes developing prototype from initial requirements and receive the feedback from different focus group surveys and students at UTA testing prototype. In combination with this feedback, brainstorming sessions with internal team are done to refine and verify requirements. To some extent, manual testing prototype application with some primary test cases is also involved in this process of verification. Validation testing happens on production version of the application with test plan mentioned in this section. This section documents test setup involved to test prototype and production versions of the applications at one place.

4.5.1 Test Setup

Setup includes Android devices with different versions of Android installed to include all supported API versions, applications permissions, screen sizes. Table 4-6 shows different devices used for testing this application.

	Device Name	Android version	Quantity
1	Samsung GT-N7100 - Phone	4.4.2	1
2	Google Nexus 7 - Tablet	5.0.2	1
3	Google Nexus 7 - Tablet	6.0.1	1
4	Motorola Moto E - Phone	5.0.2	2

Table 4-6 List of Devices Used for Testing

4.5.2 Testing

Table 4-7 shows the overview of some manual/automated and positive/negative test cases run as part of unit/regression test phases/plans.

No.	Test scenarios
1	Launching application with location settings enabled/disabled.

2	Launching application with internet connectivity settings enabled/disabled.
3	Tested working of survey user interface for different possible combinations of user inputs.
4	Recording conflicts of all combinations possible and check cloud database updated with expected information recorded.
5	Test application users and groups information updated properly in cloud-based database.
6	Test with every new installation user assigned with expected user group as per requirement.
7	Test notifications and reminders generated using set of devices mentioned in the setup.
8	Test AWS Dynamo DB interface for handling of requests and response scenarios using automated/manual tests for different scenarios.
9	Test data subscriptions services like CSV and KML files generated as expected using share menu options provided.
10	Devices having different emails logins application users and user groups are tested.
11	Different searches of conflict records from database using zip codes, distance requirements tested using manual using menu options in user interface and automated using test stubs in source code.

Table 4-7 List of Test Scenarios

4.5.3 Bug fixing and Issues faced

This section documents some important issues identified and fixed as part of verification and validation of the application.

4.5.3.1 GIF Rendering

GIF rendering in this application requires hardware acceleration in disabled mode. Starting with Android 3.0 it is possible to use GPU to perform drawing operation on the view's canvas. By default hardware acceleration is enabled which forces view rendering to use GPU. This application in the survey user interface displays GIF animation to make user interface more intuitive for user to understand the conflict scenario better. GIF file sizes provided were large in sizes and GPU rendering was not working as expected. Therefore, this application uses hardware acceleration in disabled state to avoid using GPU to render GIF animations.

4.5.3.2 View Pager Vs Fragment Manger in survey user interface

Prototype version of survey user interface uses View Pager, which also supported left-right swipe functionality to navigate through survey questions. As this survey interface requires user to move to next questions only when user answers all currently displayed questions. It is not possible to disable left-right swipe functionality in View Pager control to implement this requirement. Therefore, idea of using this View Pager control for survey is not very good and Fragment Manager is used. Fragment Manager APIs make it easy to handle different operations on fragments providing more advantage to implement survey user interface,

4.5.3.3 Android Permissions

Android system permission model [48] has updates starting with Android 6.0. Application is required to check run-time permissions from user along with file permissions mentioned in manifest file of the application. One such issue identified and fixed. Implementation for permission to show "my location button on Google Map View" was updated to be in line with this new requirement of Android Framework.

4.5.3.4 Notification Service

Notification service is dependent on Google API client connection [49]. Google API client connection is possible in two ways synchronous and asynchronous. Notification service was using asynchronous way to connect which caused the service missing to show notifications. Implementation is changed to connect API client in synchronous way using blocking connect method for Google API client.

4.6 Publishing Application on Google Play Store

Publishing on Google play store is very simple procedure and all the help is available online to do so [18]. Application name is “Safe Farer” and published on Google Play Store. Android phone users can search application in Google play store to download the production version of application. Figure 4-30 shows application in Play Store.

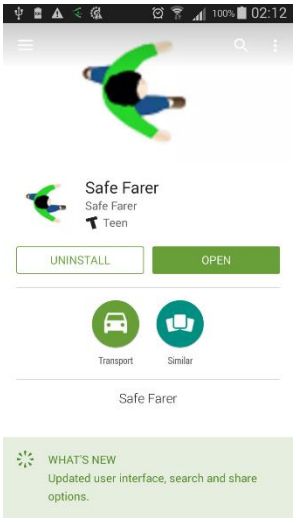

Figure 4-30 Google Play Store View

Chapter 5

Conclusion and Future work

5.1 Conclusion

This thesis project implements Android application with given requirements successfully. Process includes different steps covered from prototype to production version of application. Production version is available to users on Google Play store. Requirements reviewed and refined with different feedbacks from focus group surveys conducted and development process followed to achieve the objective of the project. Application features and functionalities validated on production version to fix some very important issues. Using the application some conflict data is collected and is present in the cloud database that is available to app and end users. This application shows that crowed sourcing of conflict data using smartphone-based app is possible. Data collected using this method in future will be available for further assessment of transportation safety.

5.2 Future work

Application with useful functionalities and features can always motivate more users. One such feature, which provides interface to existing exercise apps, is possible. Frequently used exercise app Strava by Bicyclists and Runners also provides cloud API interface to get data on user's trips and activities. This application with extended interface to Strava will provide better real-time conflict data for assessment. In addition, user interface showing activities from Strava at locations of user's day-to-day interest will motivate users to record conflicts.

The USDOT ITS plan shows the development of V2X related ITS operations; this will change the transportation infrastructure in many ways [50]. Connected vehicles will

run DSRC enabled devices to broadcast safety related messages. V2N will enable vehicle to pedestrian communication using smartphones. Smartphones already have mobile and wireless technologies to support this infrastructure. In V2X enabled infrastructure, methods for collection of conflict data is possible using automation. Application implemented in this project has manual survey method to collect the data. Smartphone-based applications in V2X, V2I enabled infrastructure can make use of broadcasted safety messages and collect the conflict data.

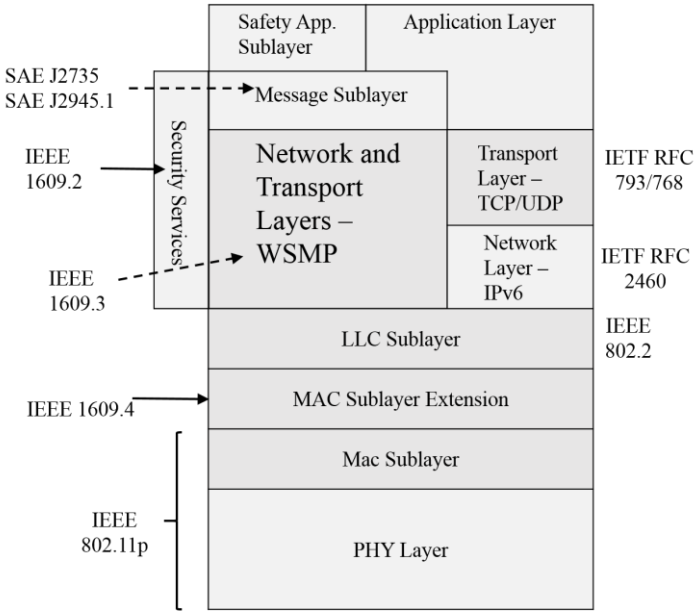

Figure 5-1 Layered network architecture for DSRC communication in the US[2]

The layered network architecture for DSRC is shown in Figure 5-1 and illustrates that it will provide support for different Safety Applications for V2V and V2I communication. Safety applications for V2V and V2I include applications like vehicle collision avoidance using various emergency alerts as hard braking, stop line violations and many more, which will help to collect conflict data for transportation safety assessment.

References

[1] T. T. Johnson, R. Gannamaraju, and S. Fischmeister, “A survey of electrical and electronic (e/e) notifications for motor vehicles,” in *24th International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, Gothenburg, Sweden, 2015, pp. 1–15.

[2] J. Kenney, “Dedicated short-range communications (dsrc) standards in the united states,” *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[3] T. T. Johnson and S. Mitra, “Anonymized reachability of rectangular hybrid automata networks,” in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2014.

[4] T. T. Johnson, “Uniform verification of safety for parameterized networks of hybrid automata,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, Electrical and Computer Engineering, Urbana, IL 61801, 2013.

[5] T. T. Johnson and S. Mitra, “Invariant synthesis for verification of parameterized cyber-physical systems with applications to aerospace systems,” in *Proceedings of the AIAA Infotech at Aerospace Conference (AIAA Infotech 2013)*, Boston, MA, Aug. 2013.

[6] T. T. Johnson and S. Mitra, “A small model theorem for rectangular hybrid automata networks,” in *Proceedings of the IFIP International Conference on Formal Techniques for Distributed Systems, Joint 14th Formal Methods for Open Object-Based Distributed Systems and 32nd Formal Techniques for Networked and Distributed Systems (FMOODS-FORTE)*, ser. LNCS. Springer, Jun. 2012, vol. 7273.

[7] T. T. Johnson and S. Mitra, “Parameterized verification of distributed cyber-physical systems: An aircraft landing protocol case study,” in *ACM/IEEE 3rd International Conference on Cyber-Physical Systems*, Apr. 2012.

[8] T. T. Johnson, S. Mitra, and C. Langbort, “Stability of digitally interconnected linear systems,” in *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC ECC 2011)*, Orlando, Florida, USA, Dec. 2011, pp. 2687–2692.

[9] T. T. Johnson and S. Mitra, “Safe and stabilizing distributed multi-path cellular flows,” *Theoretical Computer Science A*, pp. 1–39, Oct. 2015.

[10] L. Bobadilla, T. T. Johnson, and A. LaViers, “Verified planar formation control algorithms by composition of primitives,” in *AIAA SciTech*. Kissimmee, Florida: AIAA, Jan. 2015.

[11] T. T. Johnson and S. Mitra, “Safe flocking in spite of actuator faults using directional failure detectors,” *Journal of Nonlinear Systems and Applications*, vol. 2, no. 1-2, pp. 73–95, Apr. 2011.

[12] T. T. Johnson, “Fault-tolerant distributed cyber-physical systems: Two case studies,” Master’s thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, May 2010.

[13] T. T. Johnson, S. Mitra, and K. Manamcheri, “Safe and stabilizing distributed cellular flows,” in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS)*. Genoa, Italy: IEEE, Jun. 2010, pp. 577–586.

[14] T. T. Johnson and S. Mitra, “Safe flocking in spite of actuator faults,” in *12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2010)*, ser. Lecture Notes in Computer Science, S. Dolev, J. Cobb, M. Fischer, and M. Yung, Eds. Springer Berlin / Heidelberg, Sep. 2010, vol. 6366, pp. 588–602.

[15] “Comma-separated values - wikipedia, the free encyclopedia,” https://-en.wikipedia.org/wiki/Comma-separated_values, (Accessed on 07/12/2016).

[16] “Kml tutorial | keyhole markup language | google developers,” https://-developers.google.com/kml/documentation/kml_tut, (Accessed on 07/12/2016).

[17] “Amazon dynamodb nosql cloud database service,” https://aws.amazon.com/-dynamodb/, (Accessed on 07/12/2016).

[18] “Get started with publishing | android developers,” https://-developer.android.com/distribute/googleplay/start.html, (Accessed on 07/12/2016).

[19] “The dynamodbmapper class-amazon dynamodb,” http://docs.aws.amazon.com/-amazondynamodb/latest/developerguide/DynamoDBMapper.Methods.html, (Accessed on 07/12/2016).

[20] “Google maps android api | google developers,” https://developers.google.com/-maps/documentation/android-api/, (Accessed on 07/12/2016).

[21] “Start integrating google sign-in into your android app|google sign-in for android|google developers,” https://developers.google.com/identity/sign-in/android/start-integrating, (Accessed on 07/12/2016).

[22] “Making your app location-aware | android developers,” https://-developer.android.com/training/location/index.html, (Accessed on 07/12/2016).

[23] “Developer’s guide | google maps geocoding api | google developers,” https://-developers.google.com/maps/documentation/geocoding/intro, (Accessed on 07/12/2016).

[24] “Place autocomplete | google places api for android | google developers,” https://-developers.google.com/places/android-api/autocomplete, (Accessed on 07/12/2016).

[25] “Controls and gestures | google maps android api | google developers,” https://-developers.google.com/maps/documentation/android-api/controls, (Accessed on 07/12/2016).

[26] “Getting profile information | google sign-in for android | google developers,” https://developers.google.com/identity/sign-in/android/people, (Accessed on 07/12/2016).

[27] “Googlesigninapi | google apis for android | google developers,” https://-developers.google.com/android/reference/com/google/android/gms/auth/api/signin/-GoogleSignInApi, (Accessed on 07/12/2016).

[28] “Markers | google maps android api | google developers,” https://-developers.google.com/maps/documentation/android-api/marker, (Accessed on 07/12/2016).

[29] “Markeroptions | google apis for android | google developers,” https://-developers.google.com/android/reference/com/google/android/gms/maps/model/-MarkerOptions, (Accessed on 07/12/2016).

[30] “Shapes | google maps android api | google developers,” https://-developers.google.com/maps/documentation/android-api/shapes, (Accessed on 07/12/2016).

[31] “Polygonoptions | google apis for android | google developers,” https://-developers.google.com/android/reference/com/google/android/gms/maps/model/-PolygonOptions, (Accessed on 07/12/2016).

[32] “Latitude - wikipedia, the free encyclopedia,” https://en.wikipedia.org/wiki/-Latitude, (Accessed on 07/12/2016).

[33] “Longitude - wikipedia, the free encyclopedia,” https://en.wikipedia.org/wiki/-Longitude, (Accessed on 07/12/2016).

[34] “Camera and view | google maps android api | google developers,” https://-developers.google.com/maps/documentation/android-api/views, (Accessed on 07/12/2016).

[35] “Latlng | google apis for android | google developers,” https://-developers.google.com/android/reference/com/google/android/gms/maps/model/LatLng, (Accessed on 07/12/2016).

[36] “Latlngbounds | google apis for android | google developers,” https://-developers.google.com/android/reference/com/google/android/gms/maps/model/-LatLngBounds, (Accessed on 07/12/2016).

[37] “What is nosql? amazon web services (aws),” https://aws.amazon.com/nosql/, (Accessed on 07/12/2016).

[38] “Fragment | android developers,” https://developer.android.com/reference/-android/app/Fragment.html, (Accessed on 07/12/2016).

[39] “Fragmentmanager | android developers,” https://developer.android.com/-reference/android/app/FragmentManager.html, (Accessed on 07/12/2016).

[40] “Alarmmanager | android developers,” https://developer.android.com/reference/-android/app/AlarmManager.html, (Accessed on 07/12/2016).

[41] “Pendingintent | android developers,” https://developer.android.com/reference/-android/app/PendingIntent.html, (Accessed on 07/12/2016).

[42] “Wakefulbroadcastreceiver | android developers,” https://developer.android.com/-reference/android/support/v4/content/WakefulBroadcastReceiver.html, (Accessed on 07/12/2016).

[43] “Broadcastreceiver | android developers,” https://developer.android.com/-reference/android/content/BroadcastReceiver.html, (Accessed on 07/12/2016).

[44] “Scheduling repeating alarms | android developers,” https://-developer.android.com/training/scheduling/alarms.html, (Accessed on 07/12/2016).

[45] “Notificationmanager | android developers,” https://developer.android.com/-reference/android/app/NotificationManager.html, (Accessed on 07/12/2016).

[46] “Asynctask | android developers,” https://developer.android.com/reference/-android/os/AsyncTask.html, (Accessed on 07/12/2016).

[47] “App manifest | android developers,” https://developer.android.com/guide/topics/-manifest/manifest-intro.html, (Accessed on 07/12/2016).

[48] “System permissions | android developers,” https://developer.android.com/guide/-topics/security/permissions.html, (Accessed on 07/12/2016).

[49] “Googleapiclient | google apis for android | google developers,” https://-developers.google.com/android/reference/com/google/android/gms/common/api/-GoogleApiClient, (Accessed on 07/12/2016).

[50] J. Barbaresso, G. Cordahi, D. Garcia, C. Hill, A. Jendzejec, and K. Wright, “ITS 2015-2019 strategic plan,” Intelligent Transportation Systems Joint Program Office, US Department of Transportation, Tech. Rep. FHWA-JPO-14-145, 2015. [Online]. Available: http://www.its.dot.gov/strategicplan/offline/download.pdf

Biographical Information

Rahul graduated with Bachelor of Engineering degree in Electronics and Telecommunications from University of Pune, India in 2006. He earned Post-Graduate Diploma in Wireless and Mobile Computing from CDAC Pune in 2007. Rahul worked in industry at Divitas Networks for 2 years (2007 to 2009) to develop Windows mobile applications for a product on unified communication. Rahul also worked for Samsung R&D Bangalore (SRI-B) for 4+ years (2009-2014) to develop applications and power optimization techniques for smartphones running on Windows Phone and Android platforms. He earned Master of Science degree in Electrical Engineering from University of Texas at Arlington (UTA) in 2016. He worked on different projects in area the embedded systems and smartphone applications at UTA. His interests are design and development of applications for embedded systems, networked embedded systems, and smartphones.