UNIVERSITY OF ALASKA ANCHORAGE

College of Engineering

Computer Science and Computer Engineering

Location-Based Reminders Android Application

by

Adrienne Keck

Supervisor:

Prof. Cavalcanti, PhD

A CAPSTONE PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE AND COMPUTER ENGINEERING,
AT UNIVERSITY OF ALASKA ANCHORAGE, FOR THE
DEGREE OF COMPUTER SCIENCE AND COMPUTER ENGINEERING

Anchorage AK, March 2015.

amkeck@alaska.edu

Version 1.1

Abstract

The target of this capstone is to present a new Android application that utilizes GPS to determine a user's location and trigger alerts set up by the user. This capstone thesis will discuss the process of creating such an application including the design of the UI and code, work methods, tools and technologies used on the project, and testing methods. The motivation behind this project is to provide an easier way for people to remember things. It will allow users to be more efficient in their daily lives and eliminate wasted time and hassle.

Acknowledgments

The completion of this thesis would not have been possible without the help of several people to whom I am very thankful.

First, I would like to thank all of my professors at UAA. They have helped me get to where I am today. Without their patience and understanding, I would not have been able to make it this far.

Next, I would like to thank my coworkers and supervisors. They have provided me with the opportunity to get hands on experience which has helped me tremendously in school. I am very grateful for their patience with me as I made mistake after another. I have learned a lot from them.

Finally, I would like to thank my family for supporting me through school and helping to motivate me through the tough times. Without them, I would not have made it.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The Global Positioning System (GPS) is fast growing technology that can be used for a wide range of applications. It is currently used by many major tech giants today. GPS was originally developed for military purposes during the Cold War (Fig.1.1) but has since demonstrated its benefits to the civilian population [1][2]. GPS is most commonly used for navigation directions. However, GPS has become useful for many other applications. This paper will discuss one of the many ways to use this technology.

Figure 1.1: GPS was used by the military at the peak of the Cold War

1.2 Application

Global Positioning Systems have already been developed and utilized by several major tech companies. Companies such as Garmin (Fig.1.2), Apple (Fig. 1.3) [4], and Google (Fig.1.4) have their own navigation devices and applications to help users arrive at their desired destination. Google has developed an Android Location API which can be used by Android developers. The API allows the user to get their current location as well as the location of any other desired location. We will be utilizing this library to build our own application. Our application will be an Android application that will allow users to enter reminders which are triggered when a user is near a specified location. For example, if a user wants to be reminded to buy milk, the alert can be set to trigger when the user is near the grocery store.

Figure 1.2: Garmin Nuvi GPS Navigation System

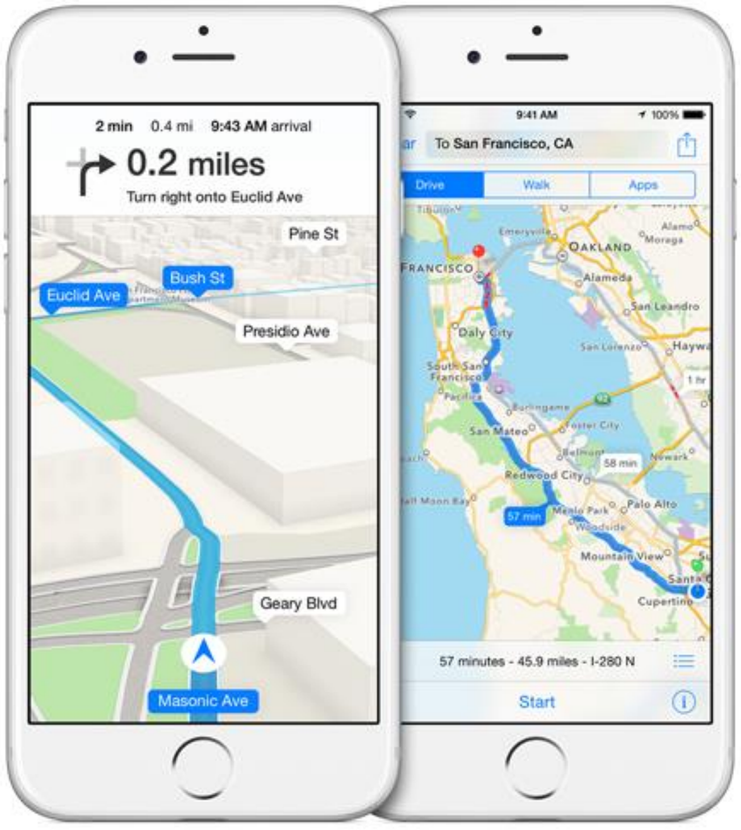


Figure 1.3: Apple Maps

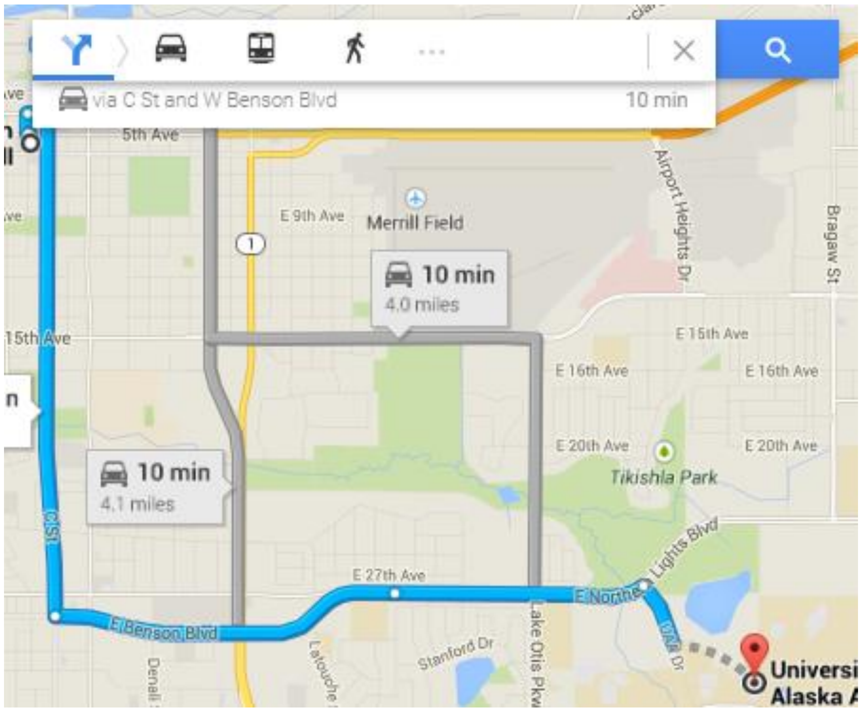


Figure 1.4: Google Maps

1.3 Motivation

The motivation for developing a global positioning system application is ultimately to improve quality of life. GPS will improve our current technologies, making them more useful and beneficial. It will prevent wasted time, whether it is someone looking for the nearest coffee shop or a firefighter trying to navigate a burning building. GPS can eventually be developed to work inside buildings and provide 3D navigation for users. This is called an Indoor Positioning System (Fig. 1.5) and is already under development today [5].

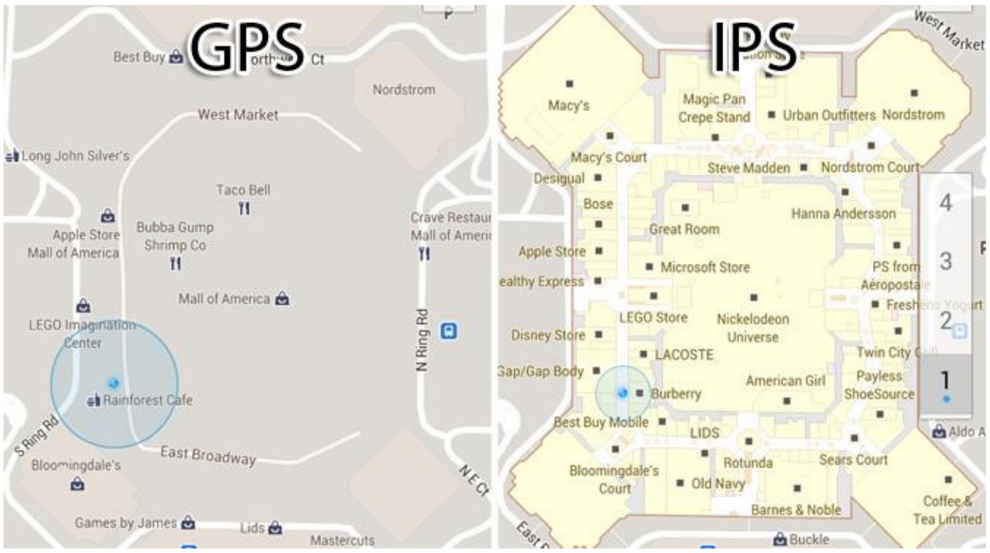


Figure 1.5: GPS vs IPS

Chapter 2

System Integration and Modelling / Methodology

2.1 Introduction

The mobile app will be implemented using agile methodology. This will enhance efficiency and quality. The finished product will be a mobile android application that will allow the user to enter different reminders that are triggered when the user is near a location. The user will also be able to view a map of their current location and see if there are any errands in the area that need to be done.

The application will be built in Eclipse IDE. We will use Google Play services location API to add location awareness. The application will be tested on an android device and within the IDE. All source code will be store in GitHub for source control.

2.2 Testing Devices

The application will be tested using Eclipse IDE and a Nexus 7 tablet (Fig. 2.1). The tablet will be running Android 4.4. Functional changes will be tested on the device while visual changes will mostly be tested in the IDE.

Figure 2.1: The Nexus 7

2.3 Google Play Services

The Google Play Services location API (Fig. 3.2) will provide the app with all the required GPS features. Google Play Services is a proprietary background service and API package for Android devices [5]. The location APIs make it easy for developers to build location-aware applications, without needing to focus on the details of the underlying location technology. They also let you minimize power consumption by using all of the capabilities of the device hardware. Some of the key features of the location API are Fused Location Provider and Geofencing APIs. The Fused Location Provider intelligently manages the underlying location technology and gives you the best location according to your needs. Geofencing APIs let your app setup geographic boundaries around specific locations and then receives notifications when the user enters or leaves those areas.

Figure 2.3: Google Play Services Location API Logo

2.4 SQLite Database

The application will require that data be saved into a database to be retrieved each time the application is opened. In order to do this, we will use a SQLite database. Saving data to a database is ideal for repeating or structured data, such as contact information. SQLite is arguably the most deployed database engine as it is used today by several widespread browsers, operating systems, and embedded systems [6]. SQLite has bindings to many programming languages. Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program.

2.5 Agile Methodology

The agile methodology will be used to develop the application. Agile methodology is an approach to software delivery that is lean, fast, and pragmatic. It is accomplished by breaking big problems into smaller problems. The agile approach is better than the waterfall approach because it allows for changes to be made throughout the project [7]. In the waterfall approach, the whole project is planned at the beginning and the project is delivered in full. This makes it difficult to make changes if a client changes their mind. It is also less efficient because it means that a lot of work will have to be thrown out. It is better to break the project into phases and include the client throughout the process (Fig. 2.3).

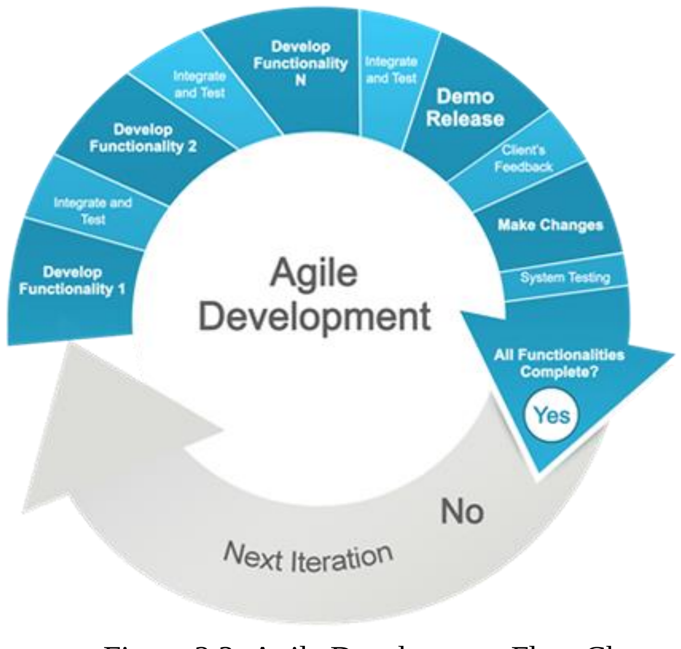


Figure 2.3: Agile Development Flow Chart

The deliverables for this project will be:

- Set up development environment
- Create start page
- Create New Reminder page
- Create All Reminders page

- Create Map page
- Save data to SQLite database
- Display reminders on map
- Add editing/deleting functionality
- Add notifications

2.6 Gantt Chart

The project will be broken into 8 weeks with the final delivery of the project falling on the week of the April 26th. The Gantt chart shows the estimated completion time for each task of the project (Fig. 2.4).

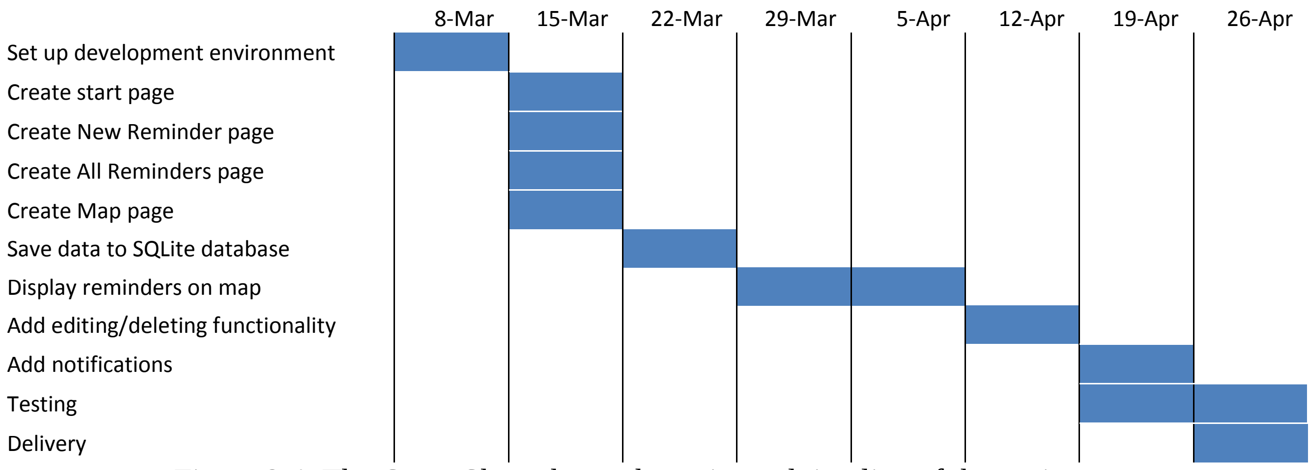


Figure 2.4: The Gantt Chart shows the estimated timeline of the project

Chapter 3

Design and Testing / User Interface

3.1 Introduction

In information technology, the user interface (UI) is everything designed into an information device with which a human being may interact -- including display screen, keyboard, mouse, the appearance of a desktop, illuminated characters, help messages, and how an application program or a Web site invites interaction and responds to it [8]. This application will require a friendly user interface so the user can easily navigate the application and add reminders. It will need to have buttons and editable inputs.

The application will also need to be tested thoroughly for any bugs. This will be done using White Box Testing and Black Box Testing. Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester. White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester [9].

3.2 Design

The application will have five display screens; it will have a home screen which contains the main menu (Fig. 3.1), a View All Reminders page where you can view all active reminders as well as edit and delete them (Fig. 3.3), an Edit Reminder page, an Add New Reminder page (Fig. 3.2), and a Google map where you can see any reminder locations nearby (Fig. 3.4). The home page will have three buttons; one for the Add New Reminder page, View All Reminders page, and the Maps Page. The buttons will be triggered by touch. The maps page will show all nearby alerts. The user can touch the map to move it around and expand the area to display alerts. The Add New Reminder page will have editable text fields to enter the title, location and notes. The distance field will be a Spinner with different preset options and the location field will have autocomplete, which will search for locations matching what the user enters.

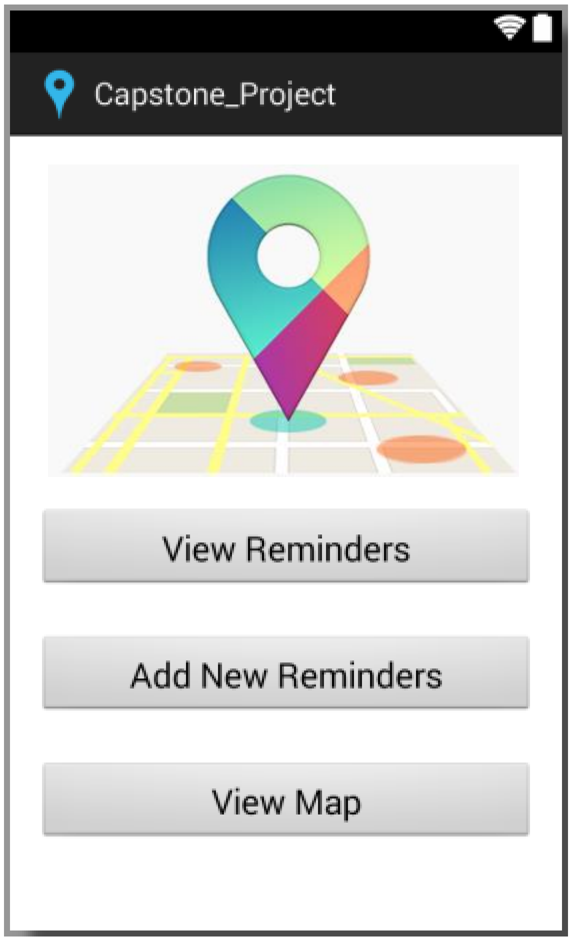


Figure 3.1: The Home screen mock up

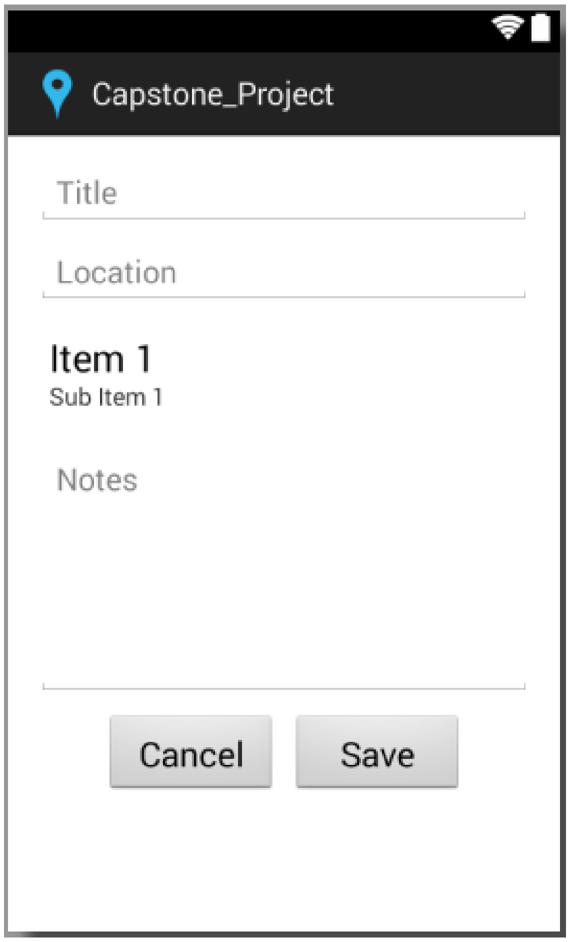


Figure 3.2: The Add New Reminder screen mock up

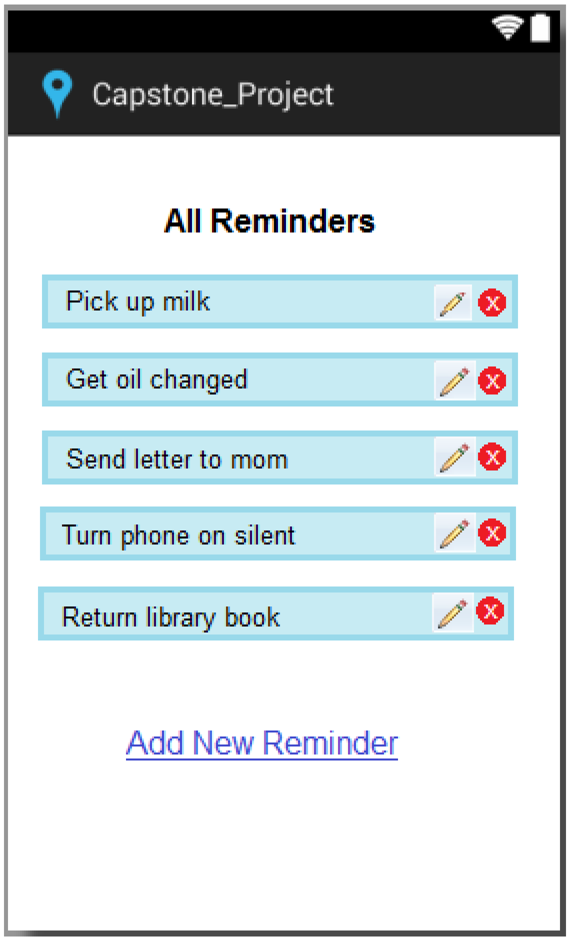


Figure 3.3: The View All Reminders screen mock up

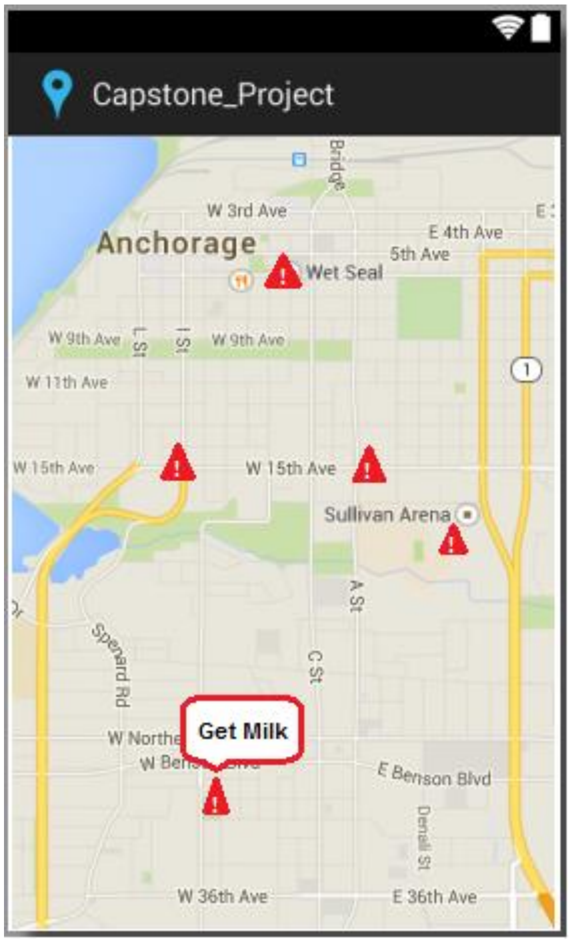


Figure 3.4: Maps screen mock up

3.3 Testing

We will test the application using black box testing and white box testing. To black box test this application, we will enter data into the application and make sure we get the desired output. For example, when we enter a new reminder, we should be able to view the new entry on the Map page and on the View All Reminders page. When we edit an entry, we should see the change in the View All Reminders page and Map page as well. One of the advantages of black box testing is the test is unbiased because the designer and the tester are independent of each other. It also allows for test cases to be made as soon as the specifications are complete [10].

To white box test the application, we will test individual pieces of code, such as loops, conditional statements, and methods, to make sure they are operating correctly and efficiently. We will check to see if all lines of code are being executed, and check for any security holes or errors. One of the advantages of white box testing is that there is no need to wait for the GUI to begin testing. White box testing covers all lines of code so it is a more thorough test of the software [11]. For a complete software examination, both white box and black box tests are required (Fig. 3.5) [12].

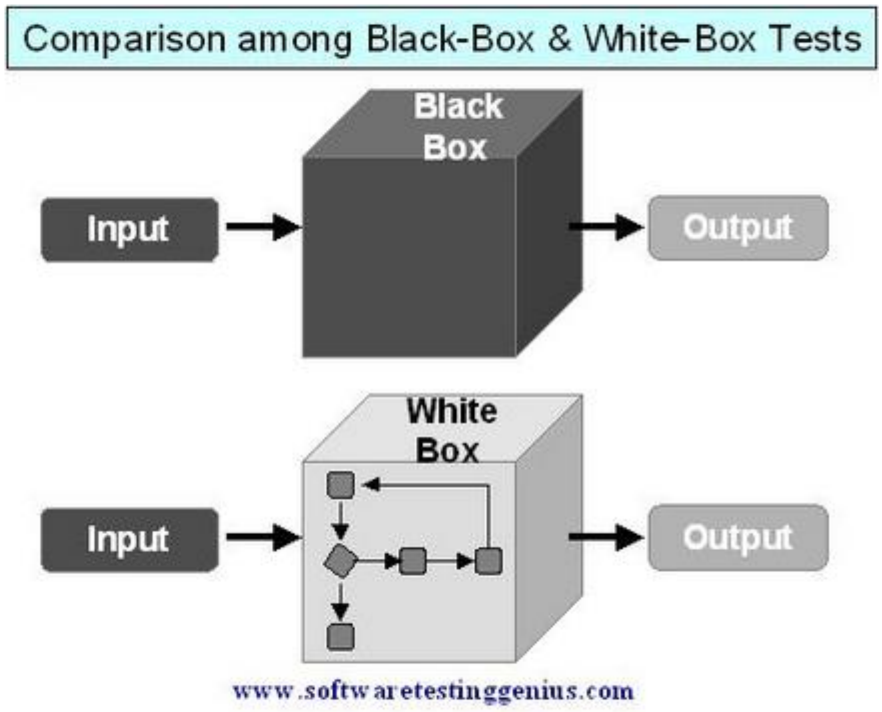

Comparison among Black-Box & White-Box Tests
Black Box
Input
Output
White Box
Input
Output
www.softwaretestinggenius.com

Figure 3.5: Comparison of Black Box and White Box Testing

Chapter 4

User's Manual

4.1 General Information

General Information section explains in general terms the system and the purpose for which it is intended.

4.1.1 System Overview

GeoReminders is an application, which allows for alerts to be triggered based on a specified location. The application provides an electronic form to enter reminders and a dynamic map. The application saves data collected to a database. Its operational status is under development. GeoReminders operates on mobile devices with an Android operating system.

4.1.2 Organization of the Manual

The user's manual consists of five sections: General Information, System Summary, Getting Started, and Using the System.

General Information section explains in general terms the system and the purpose for which it is intended.

System Summary section provides a general overview of the system. The summary outlines the uses of the system's hardware and software requirements, system's configuration, user access levels and system's behavior in case of any contingencies.

Getting Started section explains how to get GeoReminders and install it on the device. The section briefly presents the system menu.

Using The System section provides a detailed description of system functions.

4.2 System Summary

System Summary section provides a general overview of the system. The summary outlines the uses of the system’s hardware and software requirements, system’s configuration, user access levels and system’s behavior in case of any contingencies.

4.2.1 System Configuration

GeoReminders operates on mobile devices with an Android operating system. It is compatible with Android 1.5 API level 3 and higher versions. The application requires connection to Internet in order to save data to the database and an internal GPS receiver in order to obtain coordinates automatically. After installation on the device, location tracking will need to be enabled in the device settings. Once this is done, GeoReminders can be used immediately without any further configuration.

4.2.2 User Access Levels

Everyone can use the application. No registration is required to save data to the application database.

4.2.3 Contingencies

In case of power outage data are not saved in internal memory of the operating device. In case there is no Internet connection available data cannot be saved in internal memory of the operating device.

4.3 Getting Started

Getting Started section explains how to get GeoReminders and install it on the device. The section briefly presents the system menu.

4.3.1 Installation and Logging In

The newest installation version currently available can be downloaded from the Google Play Store. For specific instructions on how to install application on specific device, refer to device’s manual.

4.3.2 System Menu

GeoReminders has a main menu with three buttons (Fig. 3.1). The View Reminders button allows the user to view all reminders as a list (Fig. 3.2). The Add New Reminders button takes the user to a form that can be used to add a new reminder (Fig. 3.3). The View Map button takes the user to a Google map that displays all reminders within specified vicinity (Fig. 3.4). The user can expand or shrink the map area using their fingers.

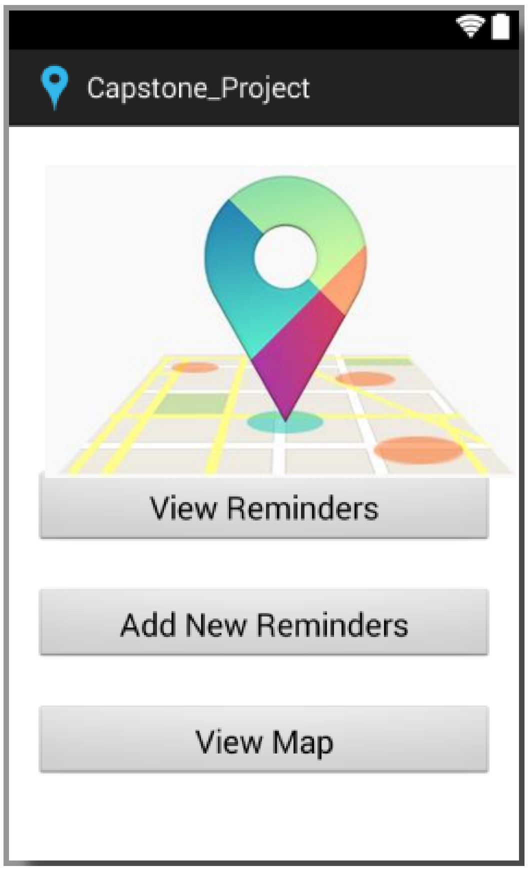


Figure 4.1: Main Menu

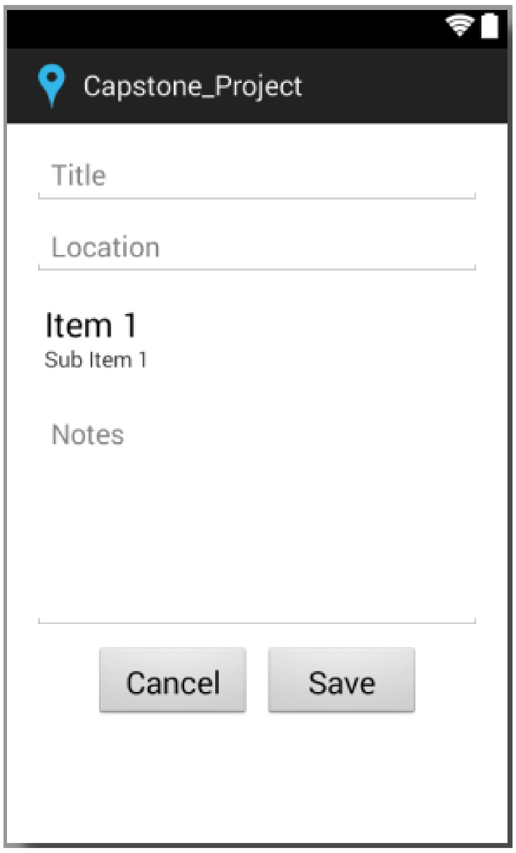
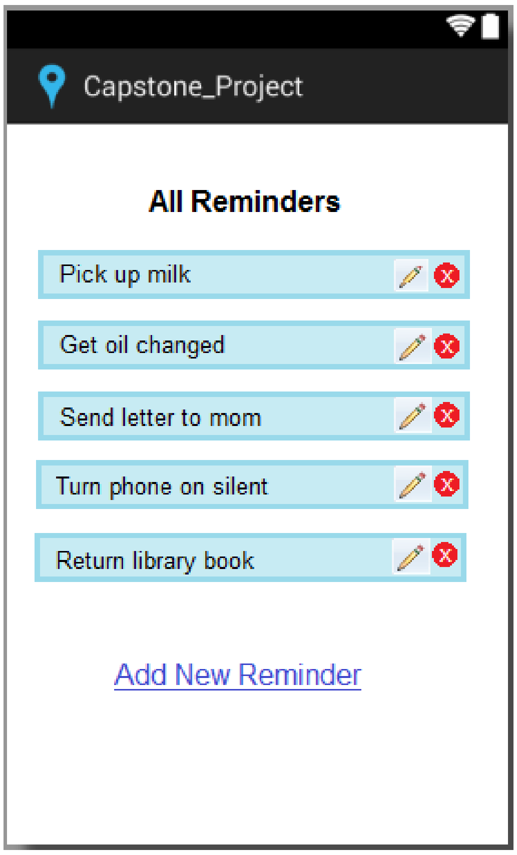


Figure 4.2: The Add New Reminder screen


Capstone_Project

All Reminders

Pick up milk

Get oil changed

Send letter to mom

Turn phone on silent

Return library book

Add New Reminder

Figure 4.3: The View All Reminders screen

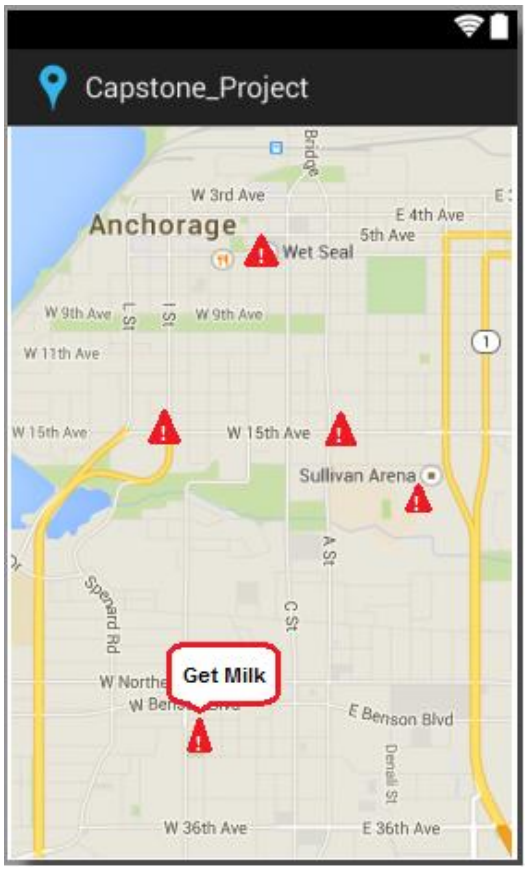


Figure 4.4: Maps screen

4.3.3 Exit System

GeoReminders can be closed by selecting Back action on the device.

4.4 Using the System

This section provides a detailed description of system functions.

4.4.1 View Reminders Page

The View Reminders page consists of a list of items containing details about each reminder. Each item contains the title and location of the item. Each item also has an edit button and a delete button (Fig. 4.1). If a user would like to edit an item in the list, they can simply click the Edit button and they will be redirected to the Edit page. If they would like to delete an item, they click the Delete button. The user will then be prompted with the message “Are you sure you want to delete this item?” If the Delete action was intentional, the user can click “yes” and the item will be removed from the list. If the user selects “no”, then the item will remain in the list. An item cannot be recovered after deletion.

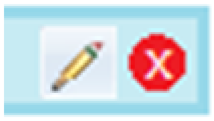


Figure 4.5: The Edit and Delete Buttons

4.4.2 Edit Reminders Page

The Edit page consists of a form similar to the Add New Reminder form. It contains four editable fields. The user may change any of the details in the form and press the Save button. This will update the item and redirect the user back to the list. If any fields have an error, they will be highlighted in red. The page will not redirect on an error.

4.4.3 Add Reminder Page

The Add Reminder page consists of four editable fields. The fields are initially blank. The user can enter a title, location, and notes for a specific item. Once all the information has been entered, the user can press Save. If there are no errors, the user will be redirected to the View Reminders page. If there are errors, the fields with errors will be highlighted in red.

4.4.4 View Maps Page

The View Maps page consists of a dynamic Google Map. It defaults to the user’s current location with a 10 mile radius. The user can expand and shrink the map area using a pinching motion with their fingers. They can also view areas away from their current location by using a dragging motion with their fingers. All reminders with locations within the map area will be displayed on the screen. Clicking on the one of the reminder icons will take the user to the Edit page for the respective item.

Chapter 5

Conclusion

5.1 Implications

The popularity of mobile phones and the ubiquity of their networks make them a promising platform for personal applications such as location-based reminders (Fig. 5.1). Location proved to be a successful trigger for the reminders as it gave the user the ability to input different types of information that can be hard for other systems to detect. The convenience and ubiquity of location-sensing provided by mobile phones outweighs some of their current weaknesses as a sensing platform. This bodes well for the use of mobiles phones as a personal ubiquitous computing platform. Our study revealed unexpected uses of location-aware reminders. We found that GeoReminders was often used for creating motivational reminders to perform activities that would vary in priority over time. This is similar to using Post-It notes in highly visible areas for motivation (Fig. 5.2). It is hoped that with continued usage of the application along with future modifications, more uses of the application will be discovered. The application has proven to improve the quality of life for its users by saving them time and money, as well as giving them any extra motivation they desire.

Figure 5.1: The use of mobile phones has become very popular.

Figure 5.2: Motivational Post-It notes

5.2 Future Scope

- Make it available on other smartphone markets such as iOS and Windows RT
- Customize the application, implement more branding
- Encourage unique and more opportunistic use

5.2.1 Modifications

- Implement a way of associating audio messages or pictures with reminders in order to offer more convenience and encourage more unique uses of the application
- Add the ability to trigger reminders by date and time, allow user to set priority
- Allow user to choose ringtones
- Add support for recurring reminders

References

[1] CNN. The History of the Cold War, 2014.
http://www.cnn.com/2014/03/04/world/gallery/cold-war-history/

[2] A. Drira. GPS Navigation for Outdoor and Indoor Environments, 2006.
http://imaging.utk.edu/publications/papers/dissertation/Anis_Pilot.pdf

[3] Wikipedia Contributors. Indoor Positioning System, 2015.
http://en.wikipedia.org/wiki/Indoor_positioning_system

[4] iOS 8 Maps. https://www.apple.com/ios/maps/

[5] S. Anthony. Think GPS is cool? IPS will blow your mind, 2012.
http://www.extremetech.com/extreme/126843-think-gps-is-cool-ips-will-blow-your-mind

[6] Location APIs. https://developer.android.com/google/play-services/location.html

[7] Wikipedia contributors. SQLite, 2015.
http://en.wikipedia.org/w/index.php?title=SQLite&oldid=651598035

[8] J. Rasmusson. The Agile Samurai, 2010.

[9] M. Rouse. User Interface (UI) Definition, 2005.
http://searchsoa.techtarget.com/definition/user-interface

[10] STF. Differences between Black Box Testing and White Box Testing, 2010.
http://softwaretestingfundamentals.com/differences-between-black-box-testing-and-white-box-testing/

[11] V. Beal. Black Box Testing, 2015.
http://www.webopedia.com/TERM/B/Black_Box_Testing.html

[12] What is a White Box Testing?, 2012 http://www.softwaretestingclass.com/white-box-testing/

Appendix A: Licensing

BSD License

Appendix B: Source Code

MainActivity.java: This is where all the functionality is located

```
package com.example.capstone_project;

import android.support.v7.app.ActionBarActivity;
import android.app.Activity;
import android.location.Location;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

import com.google.android.gms.common.ConnectionResult;
//import com.google.android.gms.common.api.GoogleApiClient.Builder;
import com.google.android.gms.common.api.*;
import com.google.android.gms.common.api.GoogleApiClient.ConnectionCallbacks;
import com.google.android.gms.common.api.GoogleApiClient.OnConnectionFailedListener;
import com.google.android.gms.drive.*;
import com.google.android.gms.location.*;

public class MainActivity extends ActionBarActivity implements ConnectionCallbacks,
OnConnectionFailedListener {
   private GoogleApiClient mGoogleApiClient;
   private Location mLastLocation;
   private static final String LOG_TAG = "ExampleApp";

   private static final String PLACES_API_BASE = "https://maps.googleapis.com/maps/api/place";
   private static final String TYPE_AUTOCOMPLETE = "/autocomplete";
   private static final String OUT_JSON = "/json";

   private final String API_KEY = getString(R.string.key);
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // Create a GoogleApiClient instance
      mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addApi(Drive.API)
            .addScope(Drive.SCOPE_FILE)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .build();
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.main, menu);
```

```
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
    public void View_Reminders(View view){
            setContentView(R.layout.view_reminders);
    }
    public void Add_New(View view){
            setContentView(R.layout.add_new);
            Spinner spinner = (Spinner) findViewById(R.id.alert);
            // Create an ArrayAdapter using the string array and a default spinner layout
            ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
                    R.array.alert, android.R.layout.simple_spinner_item);
            // Specify the layout to use when the list of choices appears
            adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            // Apply the adapter to the spinner
            spinner.setAdapter(adapter);
    }
    public void View_Map(View view){
            setContentView(R.layout.view_map);
    }
    @Override
    public void onConnected(Bundle connectionHint) {
        mLastLocation = LocationServices.FusedLocationApi.getLastLocation(
                mGoogleApiClient);
        if (mLastLocation != null) {
                                    //View mLatitudeText.setText(String.valueOf(mLastLocation.getLatitude()));
            //View mLongitudeText.setText(String.valueOf(mLastLocation.getLongitude()));
        }
    }

    @Override
    public void onConnectionSuspended(int cause) {
        // The connection has been interrupted.
        // Disable any UI components that depend on Google APIs
        // until onConnected() is called.
    }

    @Override
    public void onConnectionFailed(ConnectionResult result) {
        // This callback is important for handling errors that
        // may occur while attempting to connect with Google.
        //
        // More about this in the next section.

    }
}
```

```
class SpinnerActivity extends Activity implements OnItemSelectedListener {

    public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
        // An item was selected. You can retrieve the selected item using
        // parent.getItemAtPosition(pos)
    }

    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
}
```

activity_main.xml: This is the layout for the home page

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.capstone_project.MainActivity" >

    <Button
        android:id="@+id/view"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="200dp"
        android:layout_marginBottom="25dp"
        android:background="@android:drawable/btn_default"
        android:onClick="View_Reminders"
        android:paddingBottom="10dp"
        android:paddingLeft="20dp"
        android:paddingRight="20dp"
        android:paddingTop="10dp"
        android:text="@string/view"
        android:textSize="20sp" />

    <Button
        android:id="@+id/add"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/view"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="25dp"
        android:background="@android:drawable/btn_default"
        android:onClick="Add_New"
        android:paddingBottom="10dp"
        android:paddingLeft="20dp"
        android:paddingRight="20dp"
        android:paddingTop="10dp"
        android:text="@string/add"
        android:textSize="20sp" />
    <Button
        android:id="@+id/map"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/add"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="25dp"
        android:background="@android:drawable/btn_default"
        android:onClick="View_Map"
        android:paddingBottom="10dp"
        android:paddingLeft="20dp"
        android:paddingRight="20dp"
        android:paddingTop="10dp"
        android:text="@string/map"
        android:textSize="20sp" />

</RelativeLayout>
```