Profile Buddy

Software Requirements Specification

Version 1.5

Version History

Version No.	Version comments	Date Released
1.0	**Added:** Purpose, Product Scope, Features, Flow Chart, Functional Requirements, System requirements and Technologies	Feb 02, 2015
1.1	**Added:** Use Case diagram, Application Programming Interfaces, Activity diagram **Modified:** Prototypes, Key Competitors, Technologies	Feb 09, 2015
1.2	**Added:** Test cases, Architecture diagram, Class diagram, Sequence diagrams, Screenshot transition graph, Data structures **Modified:** Use Case diagram, Flow chart	March 02, 2015
1.3	**Added:** Use cases, System Sequence Diagram, Iteration plan **Modified:** Test cases, Data Inputs & outputs, Application screen transitions	March 29,2015
1.4	**Added:** System sequence diagram, Use cases, Sequence diagrams, Activity diagram, Application screen transitions **Modified:** Use Case diagram, Class diagram, Test cases	April 18, 2015
1.5	**Added:** Application screen transition **Modified:** Test cases, Use cases, Project plan	May 02, 2015

Table of Contents

1. Purpose

As mobile devices have taken a top spot in the list of most essential things in our life, it is important to balance between when they are needed and when they are not. So, this application helps us to maintain one aspect of this balance. This application will help to set the right profile at the right location.

2. Product Scope

To design and develop an application which will set the profile of the mobile device based on its current location. The location will be determined using the GPS. The application will relieve the user from changing the profile every time he visits places.

3. Features:

- **Location based profile change:** Changes the profile based on device's location, which user has defined in the application.
- **Calendar event based profile change:** Changes the profile of user's device based on the mode user has set against each event (i.e. events from Google Calendar) in the application.
- **Driving Mode:** Declines calls based on the rules defined, if the user is driving and sends a text message saying "User currently driving".
- **Green Mode:** Gives option to switch off GPS when battery reaches below 20%.
- **Push notification:** Gives a notification to the user, when a new event is created or when the user mode changes.

4. Use Case Diagram

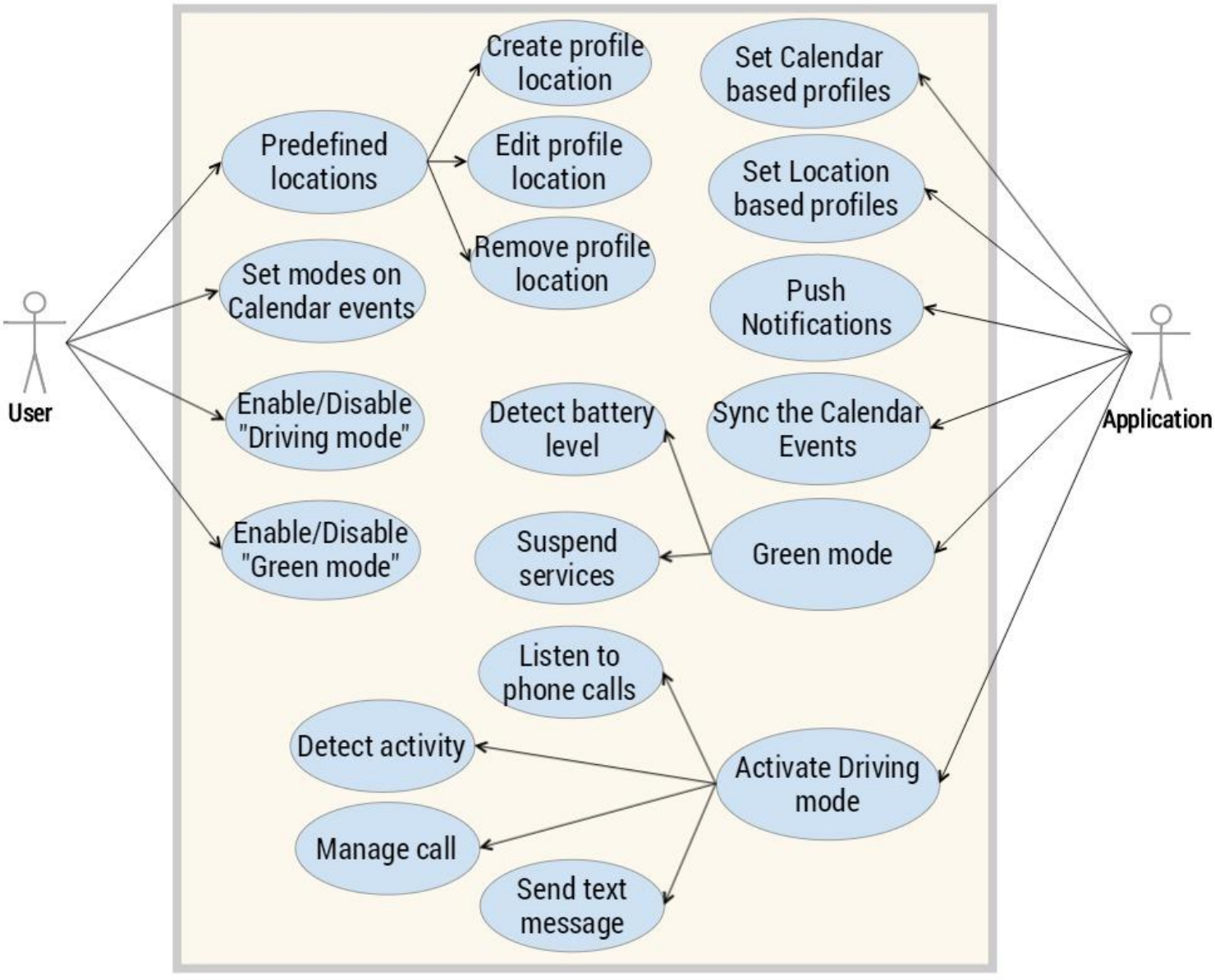


Fig. 4.1: Use case diagram

4.1 Use case (Location based profile):

Name: Set Location

Level: User Goal

Scope: Profile Buddy Application

Primary Actor: Application User

Stakeholders and Interests

Application User: User should be able to set location profile easily and the application should automatically set profile based on location following it.

Developer: Application should run smoothly, provide maximum user satisfaction and provide a simple and interactive GUI.

Precondition: User should install the application.

Post Condition: Location profile saved, application automatically checks location and update the profile

Main Success Scenario:

1. User opens the application.
2. Clicks on crate “new profile”.
3. Enter address and desired profile.
4. Set Radius.
5. Saves the location profile.

Extension: At any time, application fails to find address

1. Application displays error message.
2. User enters correct address and submits it again.
3. Application verifies the address.
4. After verification saves the address, with the respective profile.

4.2 Use case (Event based profile):

Name: Set Calendar Event

Level: User Goal

Scope: Profile Buddy Application

Primary Actor: Application User

Stakeholders and Interests

Application User: User should be able to create event in the calendar and application will import that events in the application and user will be able to set profile respective each event in the calendar.

Developer: Application should run smoothly, should properly import event from calendar and change the profile, provide maximum user satisfaction and provide a simple and interactive GUI

Precondition: User should install the application.

Post Condition: Event based profile saved, application automatically checks events time, date and update the profile

Main Success Scenario:

1. User opens the application.
2. Clicks on events which are in the application.
3. Set the profile for the event.
4. Saves the event profile.

Extension: At any time, if two events are set at the same time

1. Application selected the first event found.
2. Sets the profile for that event.

4.3 Use Case (Driving Mode)

Name: Activate Driving Mode

Level: User Goal

Scope: Profile Buddy Application

Primary Actor: Application User

Stakeholders and Interests

Application User: User shall be able to activate driving mode and application shall be able to listen to oncoming calls, detect the activity of user (e.g. driving, not driving), reject the call if user is driving and send text message to the caller.

Developer: Application shall be robust, reliable & run smoothly. It shall efficiently communicate with Telephony API and provide maximum user satisfaction and interactive GUI.

Pre-Condition: User shall install application on an Android device.

Post-Condition: Incoming call rejected while driving and text message sent to the caller.

Main Success Scenario:

1. User opens the application.
2. Activates the driving mode.
3. Application detects the incoming call.
4. Application detects if user is driving.
5. Application rejects the call and sends text message to the caller.

Extensions:

1. If the application doesn't detect driving, it doesn't reject the call.
2. Application unable to distinguish between driver and rider, it rejects call for both.

4.4 Use Case (Green Mode)

Name: Green Mode

Level: User Goal

Scope: Profile Buddy Application

Primary Actor: Application User

Stakeholder and Interests

Application User: User shall be able to activate green mode and application shall suspend service when battery level gets low.

Developer: Application shall be robust, reliable and run smoothly. It shall communicate with the system effectively to detect battery level and suspend service without any error.

Pre-Condition: User shall install application on an android device.

Post-Condition: Profile Buddy service stopped at low battery level.

Main Success Scenario:

1. User opens the application.
2. User activates green mode.
3. Application detects battery level.
4. Application suspends profile buddy service if low battery level.

Extensions:

If application doesn’t detect battery level, it will never close profile buddy service and service will keep on running.

4.5 Use Case (Default Mode)

Name: Default Mode

Level: User Goal

Scope: Profile Buddy Application

Primary Actor: Application User

Stakeholder and Interest

Application User: User shall be able to activate default mode and application shall activate default mode when predefined profile is completed.

Developer: Application shall be robust, reliable and run smoothly. It shall provide maximum user satisfaction.

Pre-Condition: User shall install application on an android device.

Post-Condition: Profile Buddy activates default mode.

Main Success Scenario:

1. User opens application.
2. User activates default mode.
3. Application activates default mode when user is out of predefined profile.

5. Flow Chart

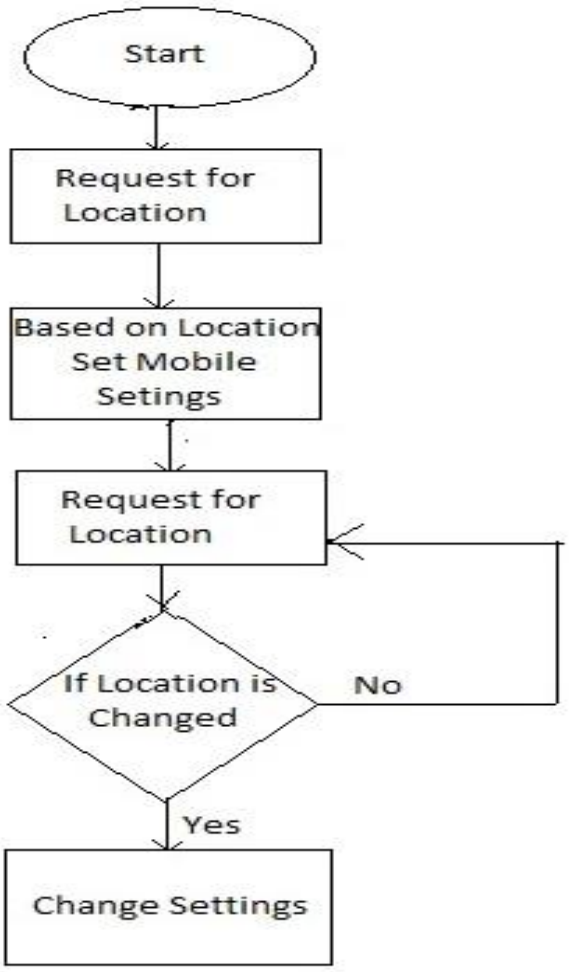


Fig. 5.1: Basic flow diagram of the service

6. Transition Diagram

6.1 Add Location and Edit Location

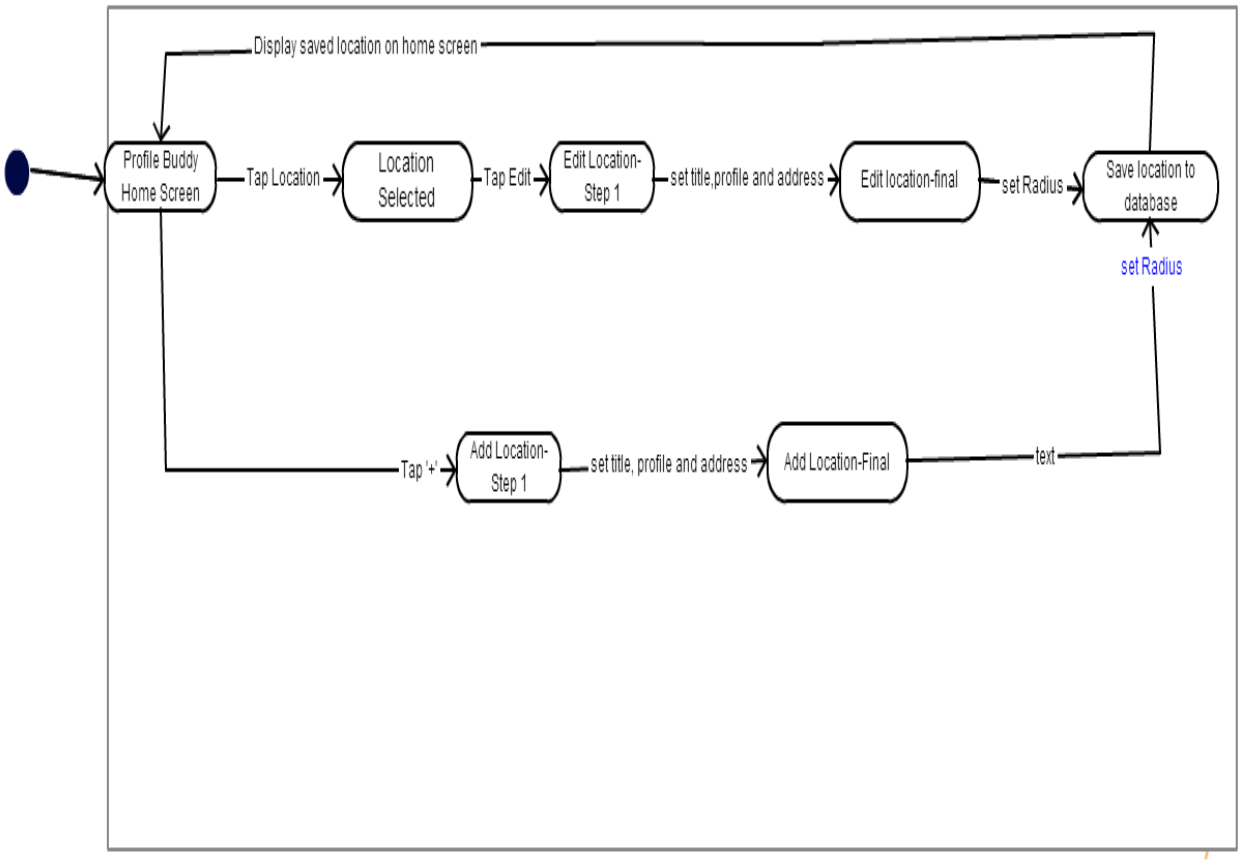


Fig. 6.1: Transition diagram

7. Key Competitors

- **“Llama”** is an application which allows you to set the profile for a specific location and time for which you are going to be at that location. It is purely a location based profiling app. Llama uses cell tower to get the location of the device.[1]
- **“AutomateIt”** does multiple tasks such as switching on/off Wi-Fi and setting profile modes. Have to manually set location and time.[3]
- **“Automagic Automation”** is a flexible app which allows you to draw flowcharts as per your requirements. Have to specify the location manually and diameter to set the boundaries of the location.[4]

7.1 What’s new in Profile Buddy?

- Profile buddy uses GPS compared to cellular radio technology used by its competitors.
- Driving mode, a new concept which uses activity recognition to identify whether user is driving and take actions accordingly.
- Green mode, helps save your battery when running low on battery by switching off GPS.

7.2 How GPS is a better option?

No direct line of sight from cell phone to cellular base station, while GPS has a direct line of sight which gives more accurate estimate of location. [2]

7.3 Why do we have to set the location manually?

Whenever we use any GPS, we prefer to put the address manually, although the Maps will give you suggestion according to place you enter and save the location i.e.: Home, office so, it’s better to do it manually, and give the application perfect location. We don`t have to go to location and save it, we can enter the address just by sitting our home and save it.

7.4 How does Google calendar helps?

Usually all the people store all there appointment and meeting at calendar and they get reminder when it`s time for their meeting, so our idea is that to import that calendar and extract those events and give option to user to set the profile according to the event, because of this user don't have to maintain a different calendar, all management will be done by the one calendar only.

7.5 Driving mode, how does it work?

Driving mode is one of the new feature, in this the application will calculate your speed and when your speed will be above say 15MPH, app will dynamically activate the driving mode, and will give a push notification, so if you are driving you will not check it, but if user is in public transport user will get the notification and can change it again to normal mode, as user is not driving.

7.6 Comparison between Profile Buddy and other applications:

ProfileBuddy	**Llama**	**Automagic Automation**
GPS based profiling	Cell tower based Profiling	GPS based Profiling
Simple Interface	Simple Interface	Complicated Interface
Push Notification	No Push Notification	No Push Notification
Dynamic Radius according to location.	Constant Radius	Constant Radius
Event Based Profile	Event Based Profile	Event Based Profile
Google Calendar Import	No Calendar import	No calendar import
Driving Mode	No this kind of feature	No this kind of Feature

8. Functional Requirements

- Capture user location and set the profile accordingly.
- Read calendar events, notify users about the new events and set profile.
- Detect if user is driving and set “Driving Mode”.
- Detect if device is low on battery and activate “Green mode”.

9. Application Programming Interfaces (API)

9.1 Google Location Services API

The Google location services API is used for finding the GPS coordinates (latitude and longitude) which uniquely identify all locations. Other than this, location services are also helpful in determining current activity of the user, which is important for driving mode in the application. [6]

9.2 Google Geocoding API

The Google Geocoding API is useful is converting the human readable addresses into GPS coordinates. This feature is very useful particularly in this application for defining a particular area. [9]

9.3 Google Calendar API

This API is used for accessing the mobile calendar events, which is need for the application to set profile according to the event. [8]

9.4 Android Telephony API

Google Telephony API is used for accessing and controlling the telephonic features of the phone like Call and SMS functions. For this application this is helpful for setting the mobile in ‘Driving mode’. [9]

9.5 Android Audio Manager API

Android Audio Manager API is used for implementing the key feature of this application i.e., changing the profile of the phone from silent Mode to general or vibrate Mode to Flight Mode as requested by the user. [10]

10. Application Architecture

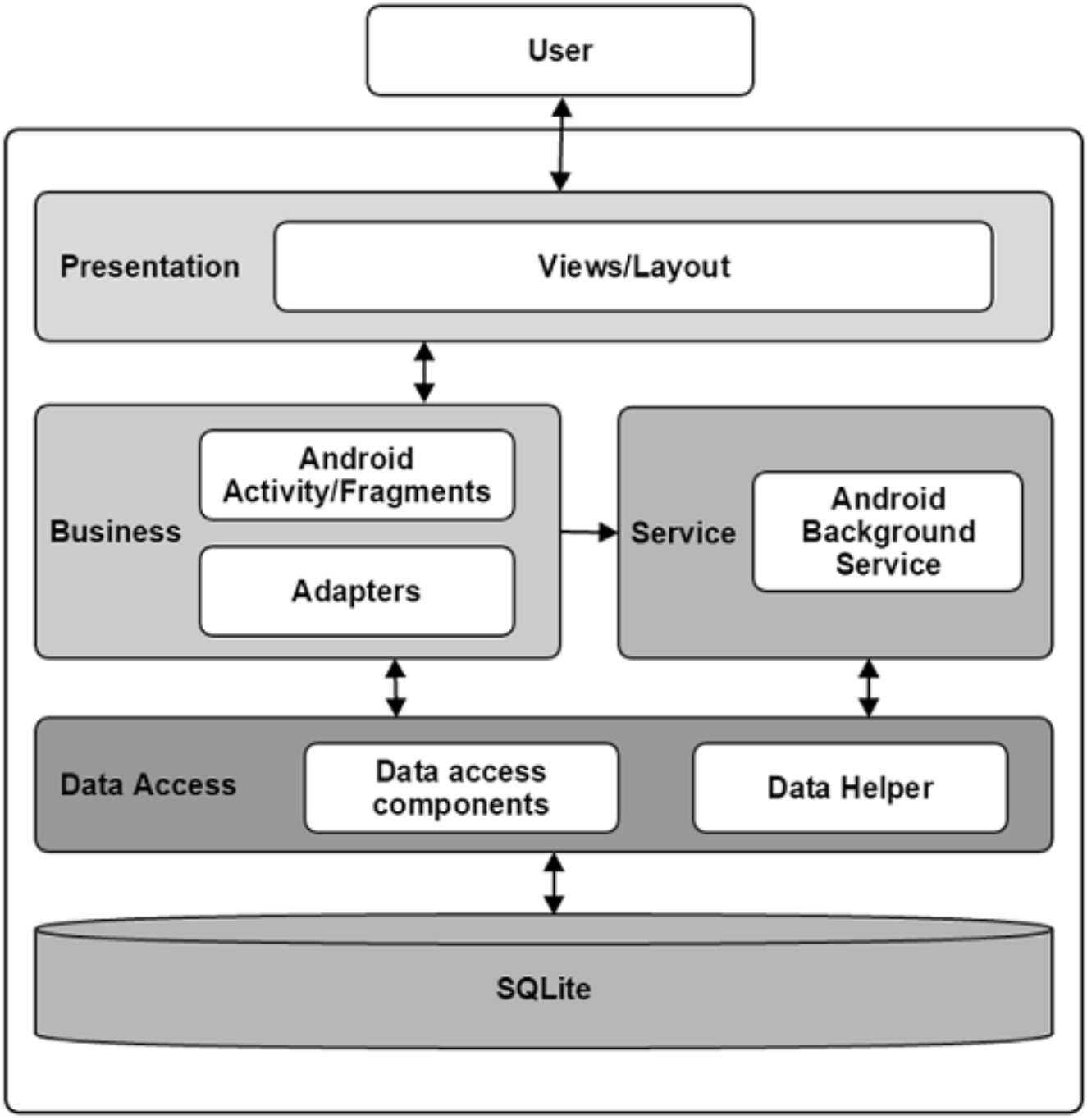


Fig. 10.1: Application architecture

11. Class Diagram

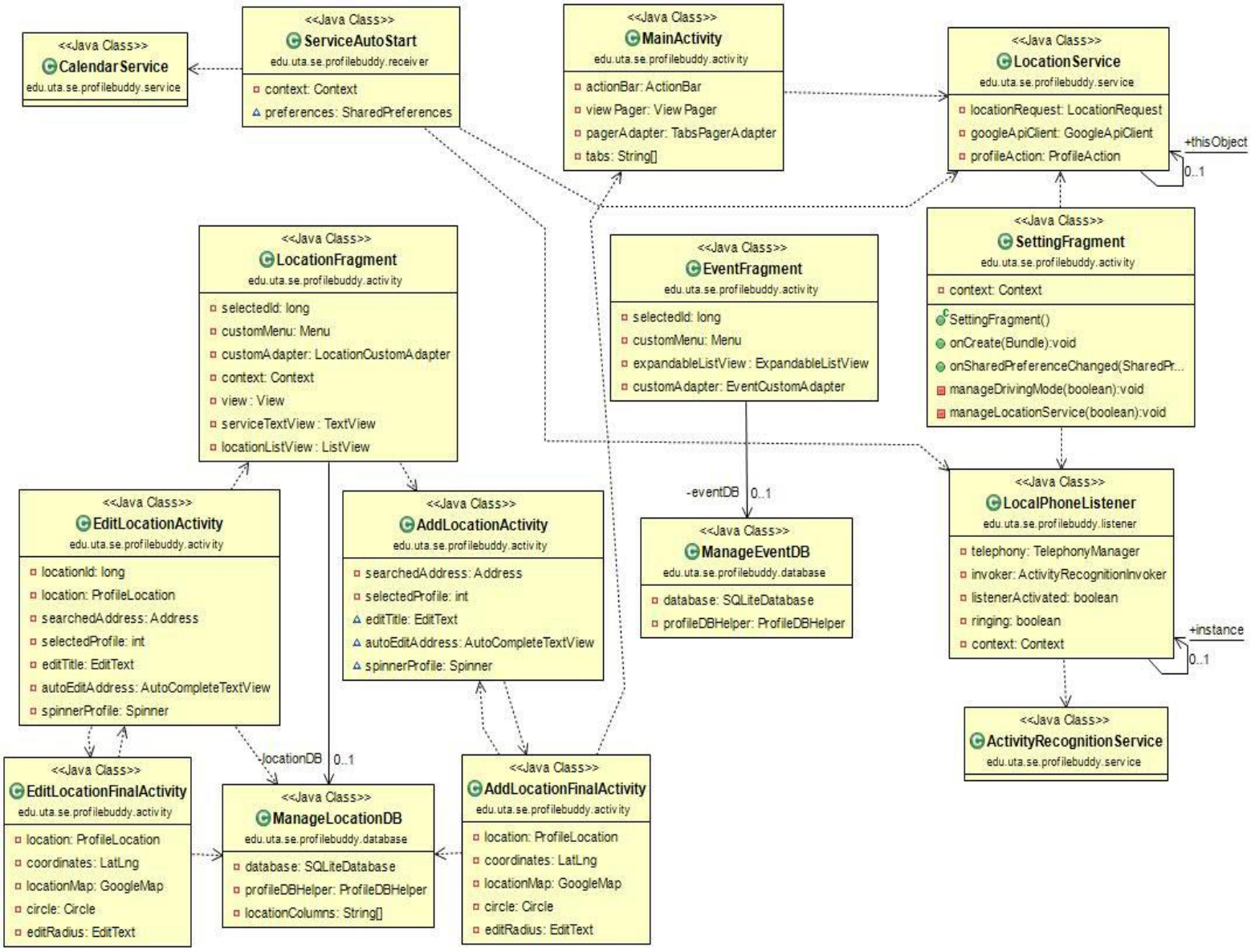


Fig. 11.1: Class Diagram

12. Sequence Diagrams

12.1 Add Location profile

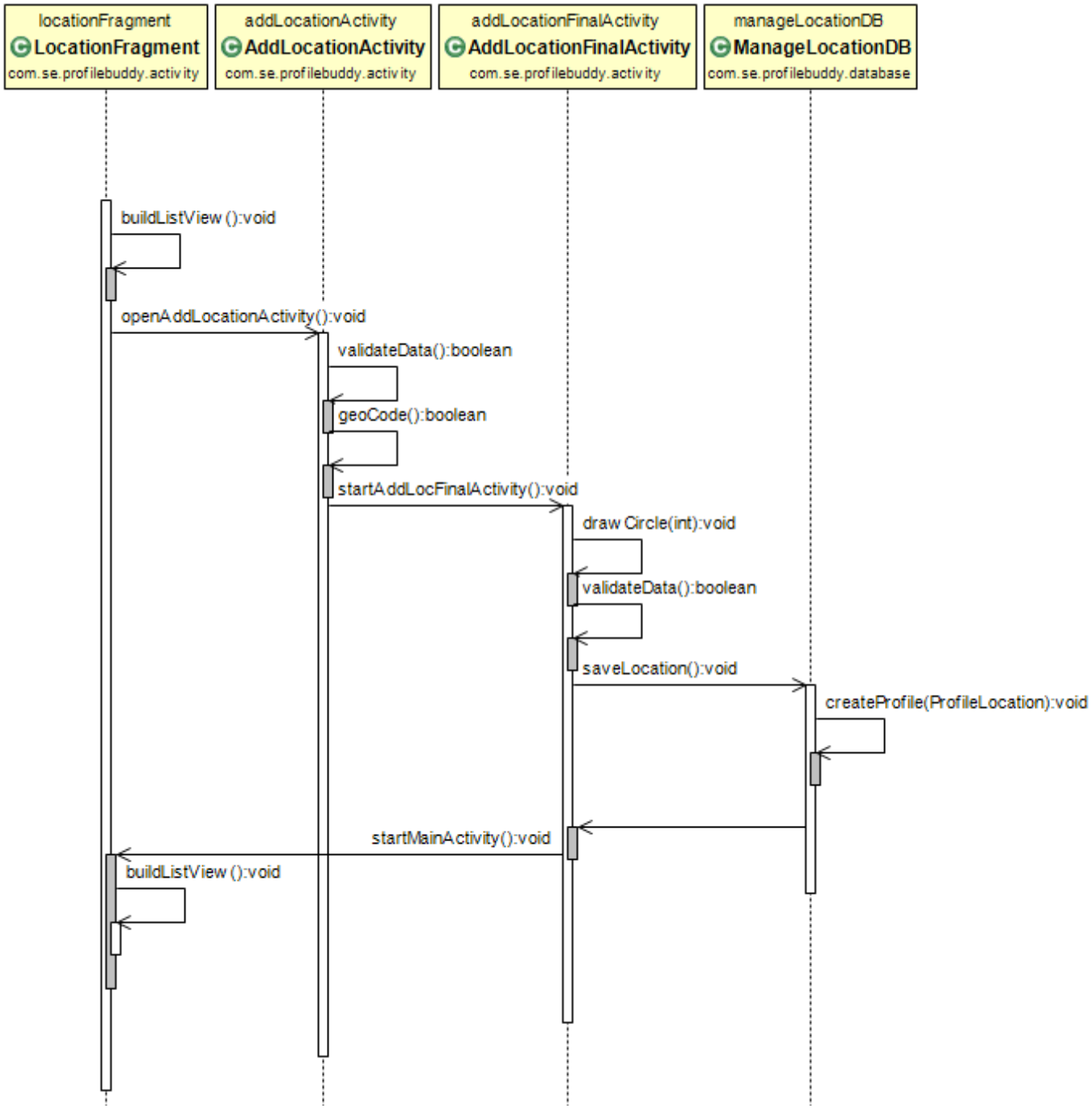


Fig. 12.1: Add Location Sequence Diagram

12.2 Edit Location profile

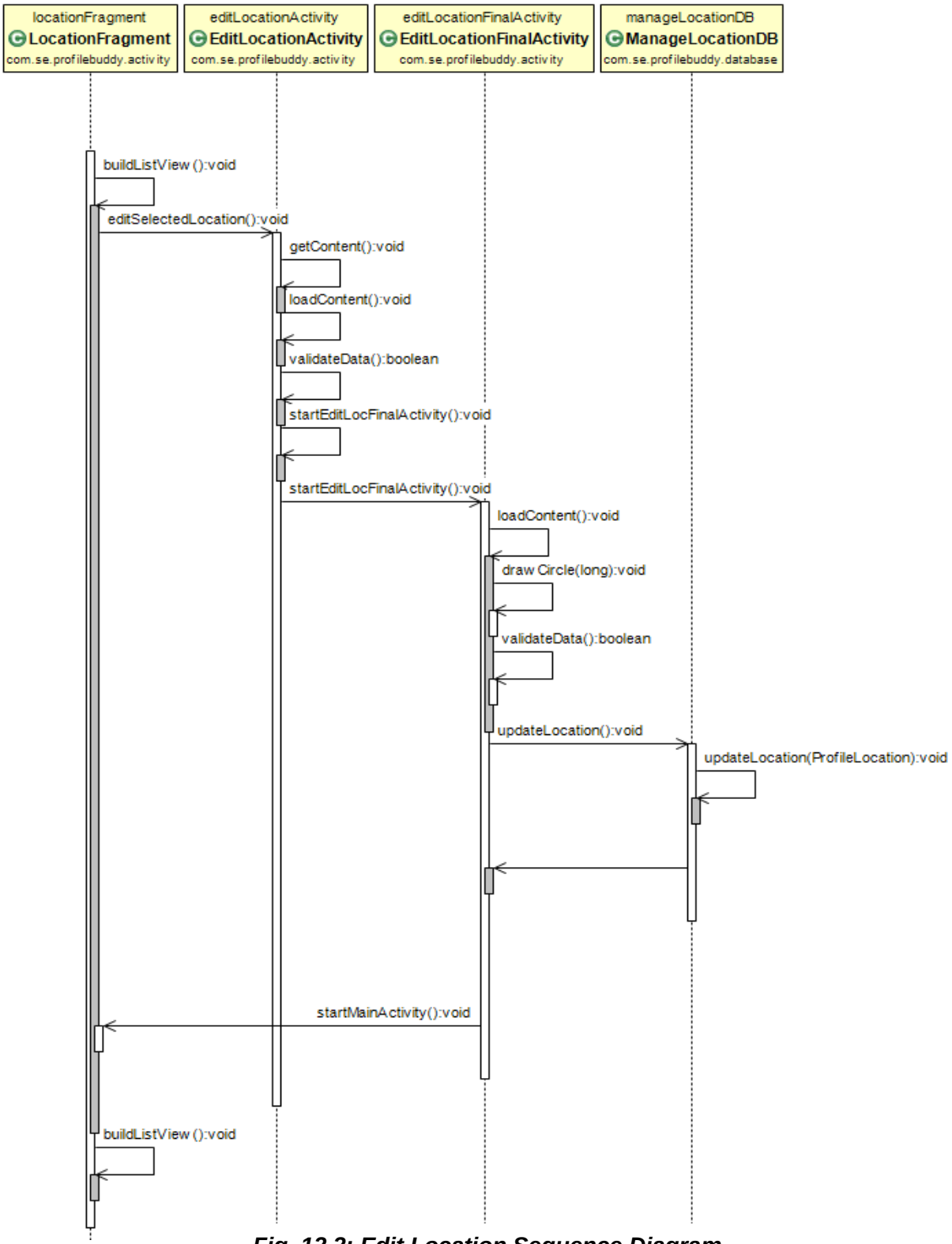


Fig. 12.2: Edit Location Sequence Diagram

12.3 Location Service

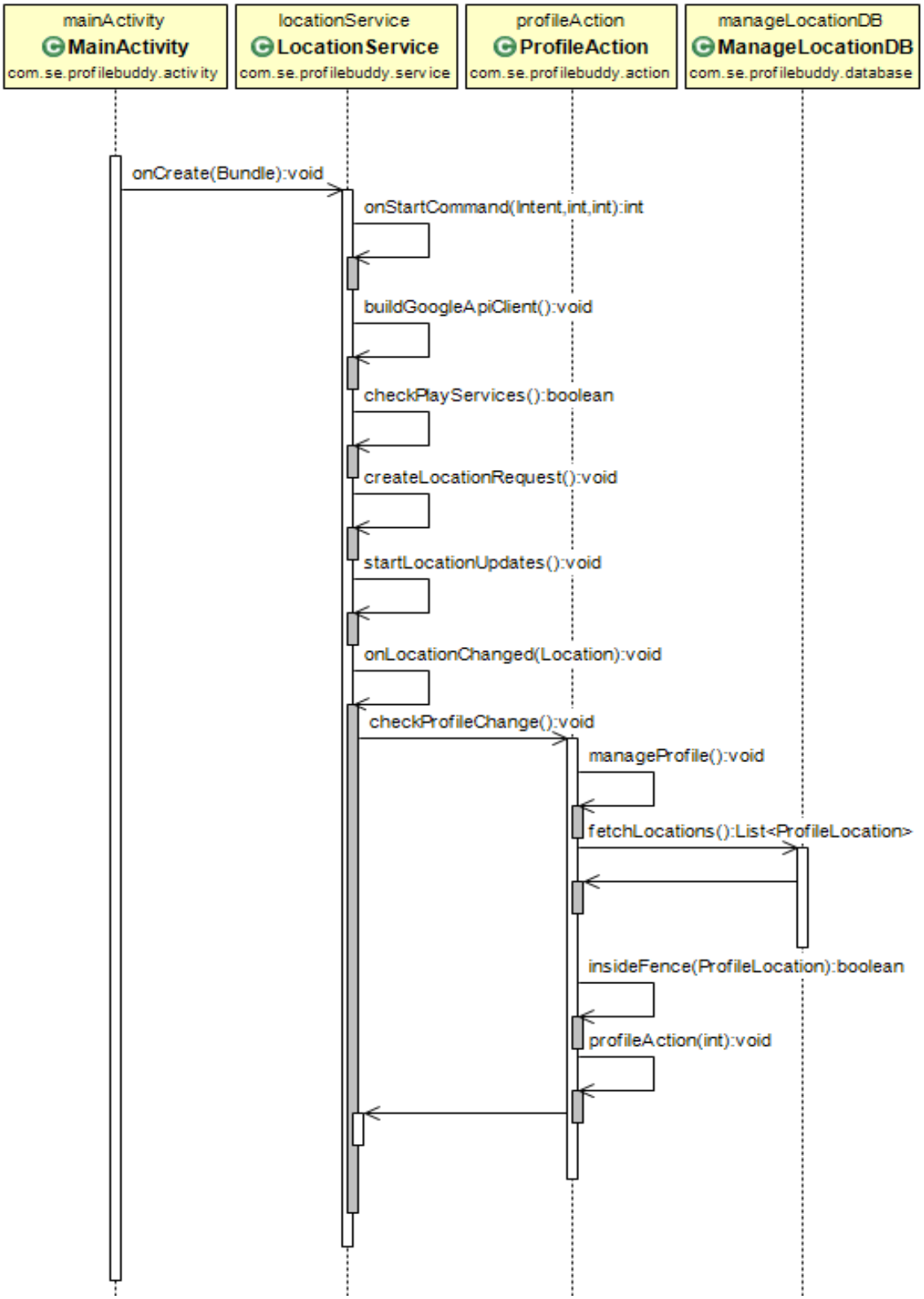


Fig. 12.3: Location Service Sequence Diagram

12.4 Driving Mode

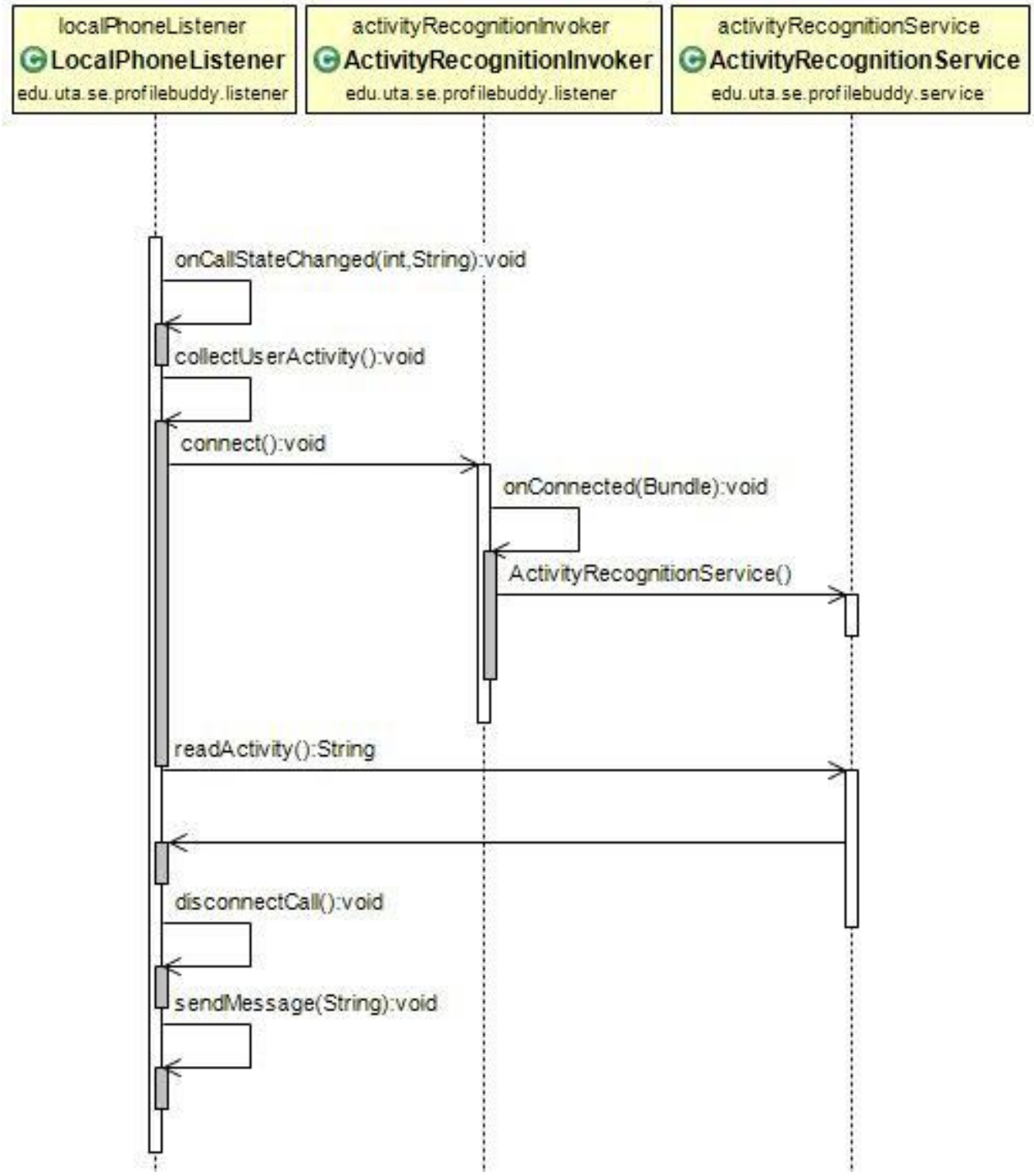


Fig. 12.4: Driving Mode Sequence Diagram

12.5 Green Mode

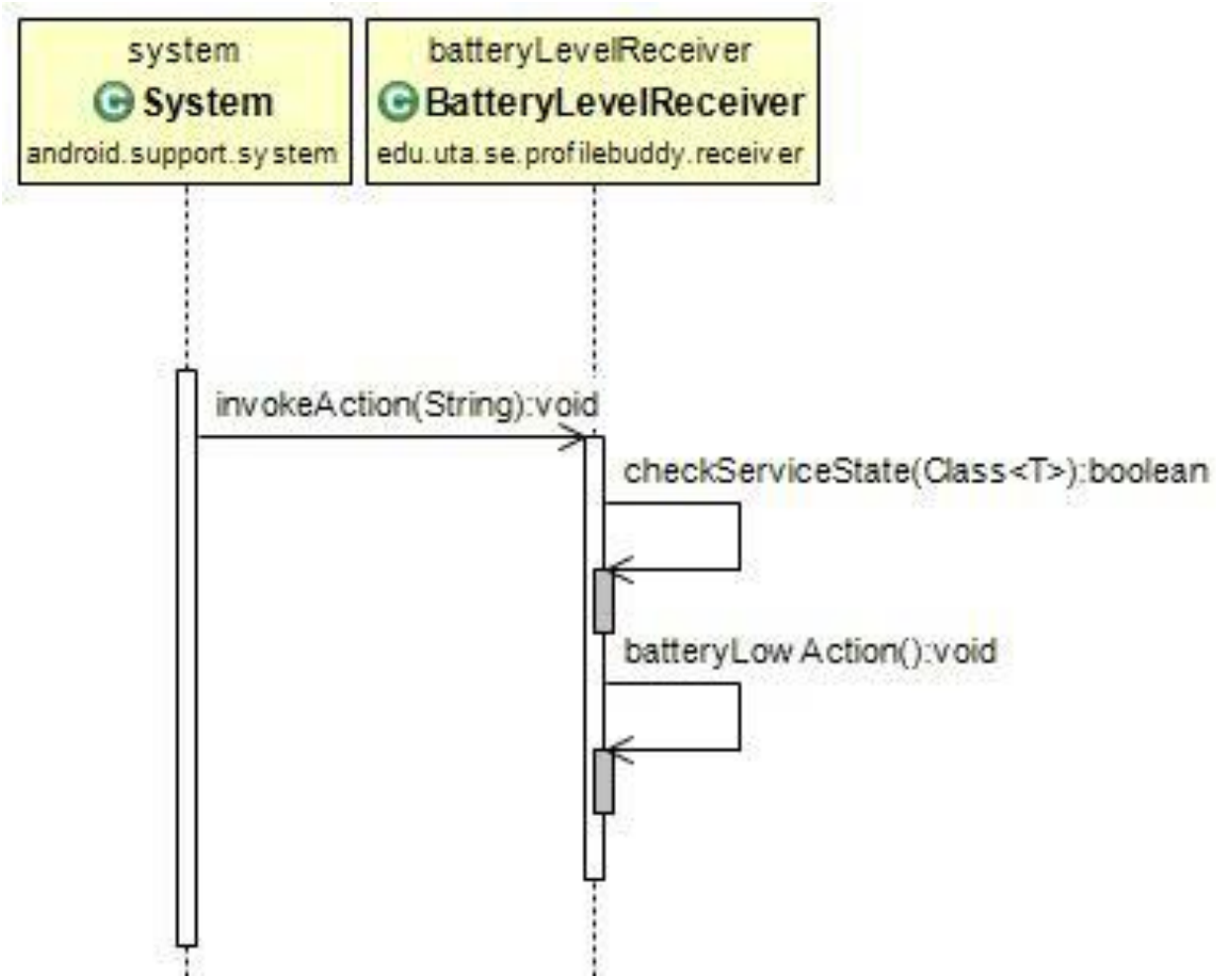


Fig. 12.5: Green Mode Sequence Diagram

13. System Sequence Diagram

13.1 Add Location:

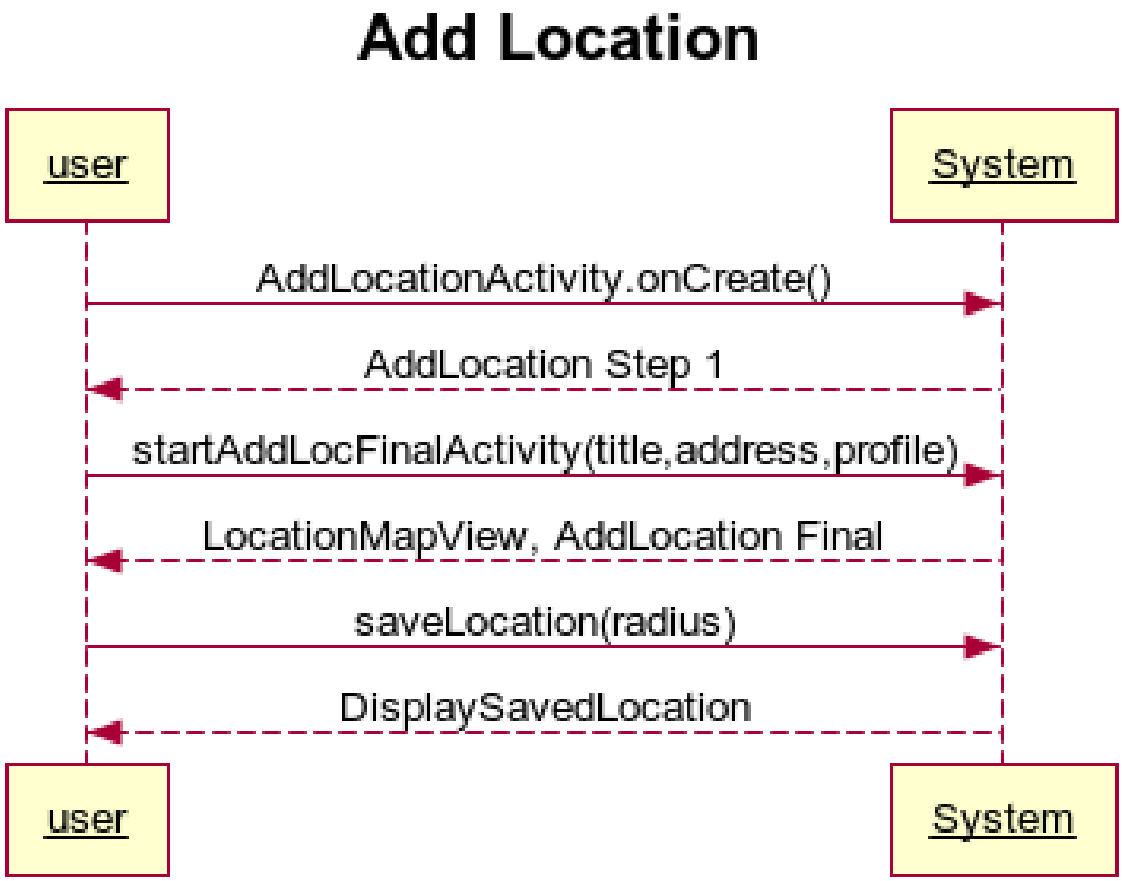


Fig. 13.1: Add Location System Sequence Diagram

13.2 Edit Location:

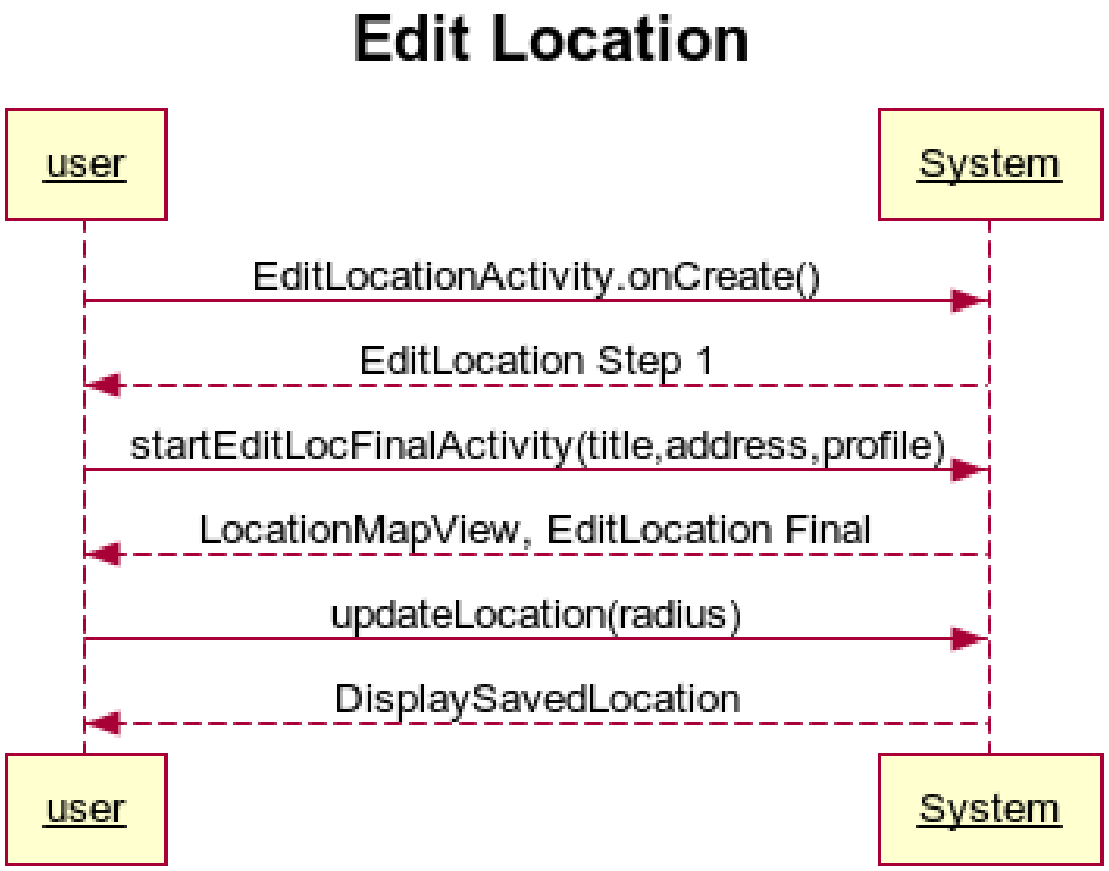


Fig. 13.2: Edit Location System Sequence Diagram

13.3 Manage Calendar Events:

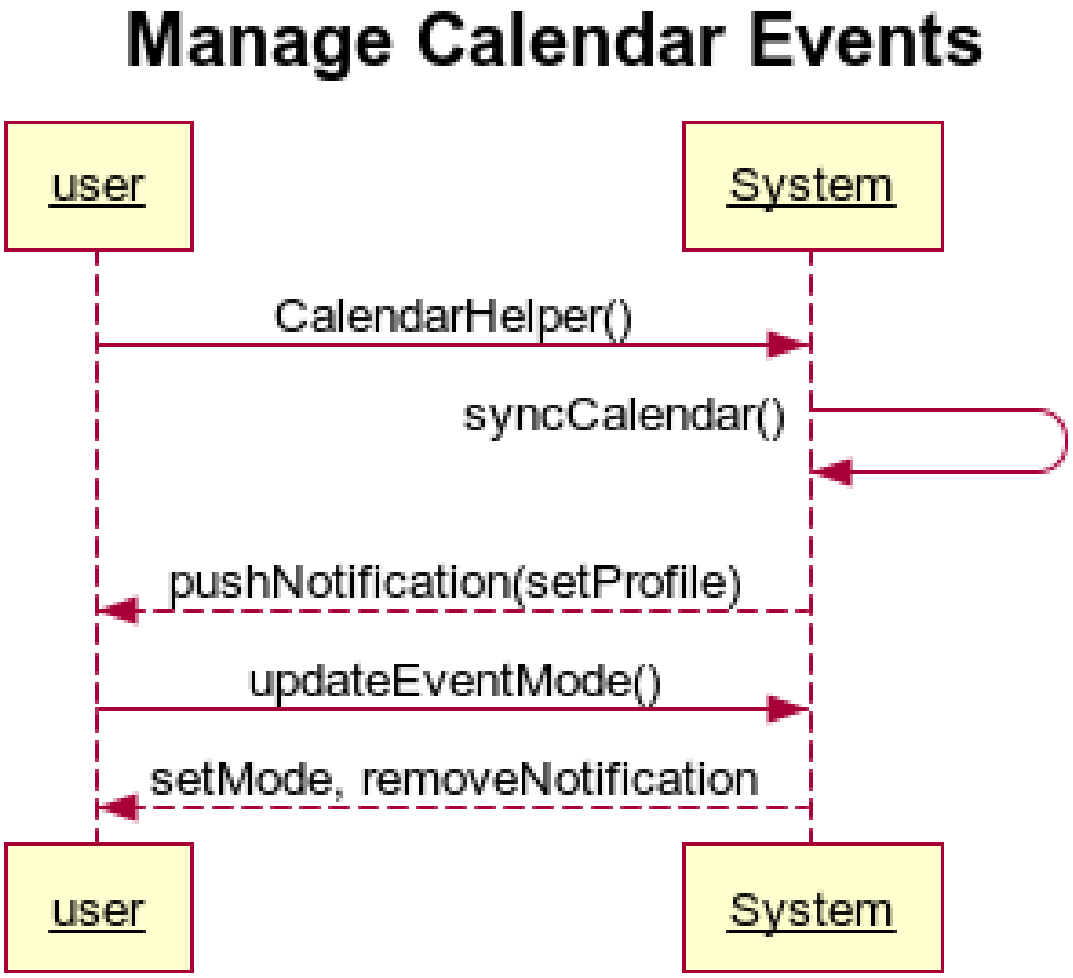


Fig. 13.3: Calendar Events System Sequence Diagram

13.4 Driving Mode:

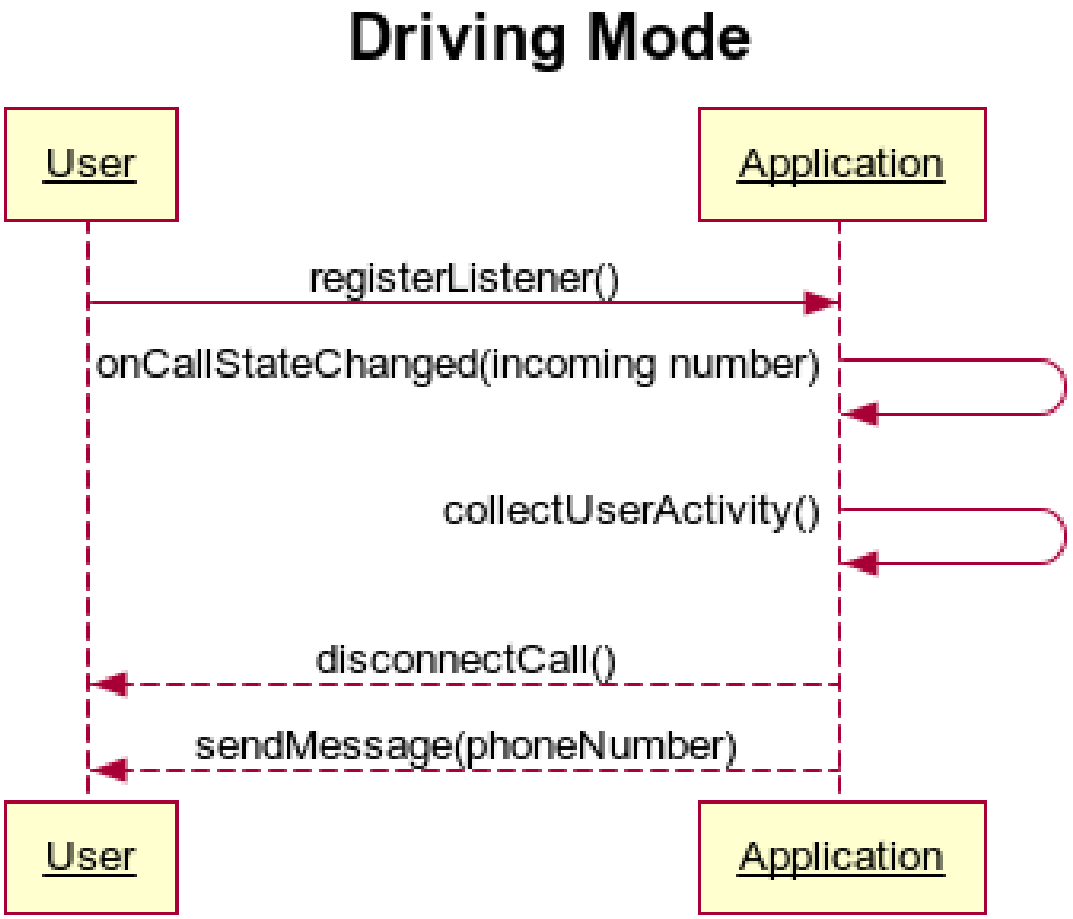


Fig. 13.4: Driving Mode System Sequence Diagram

13.5 Green Mode:

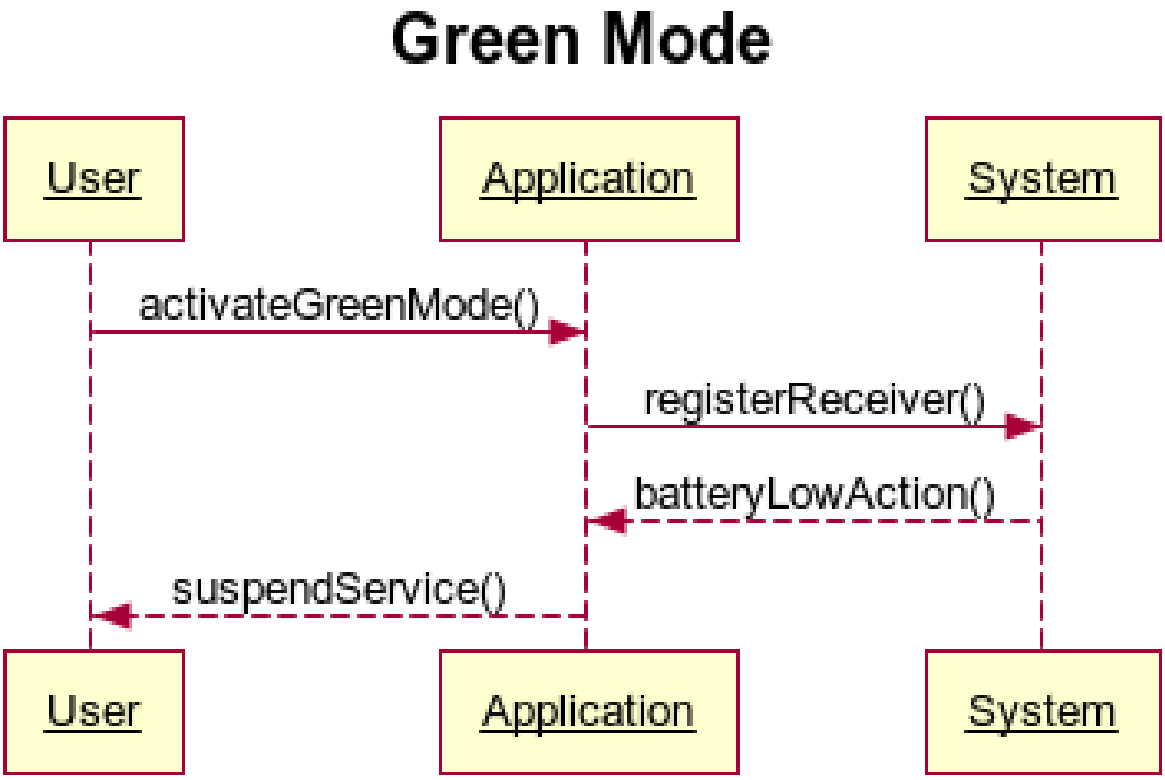


Fig. 13.5: Green Mode System Sequence Diagram

14. API - Activity Diagram

14.1 Location based profiles

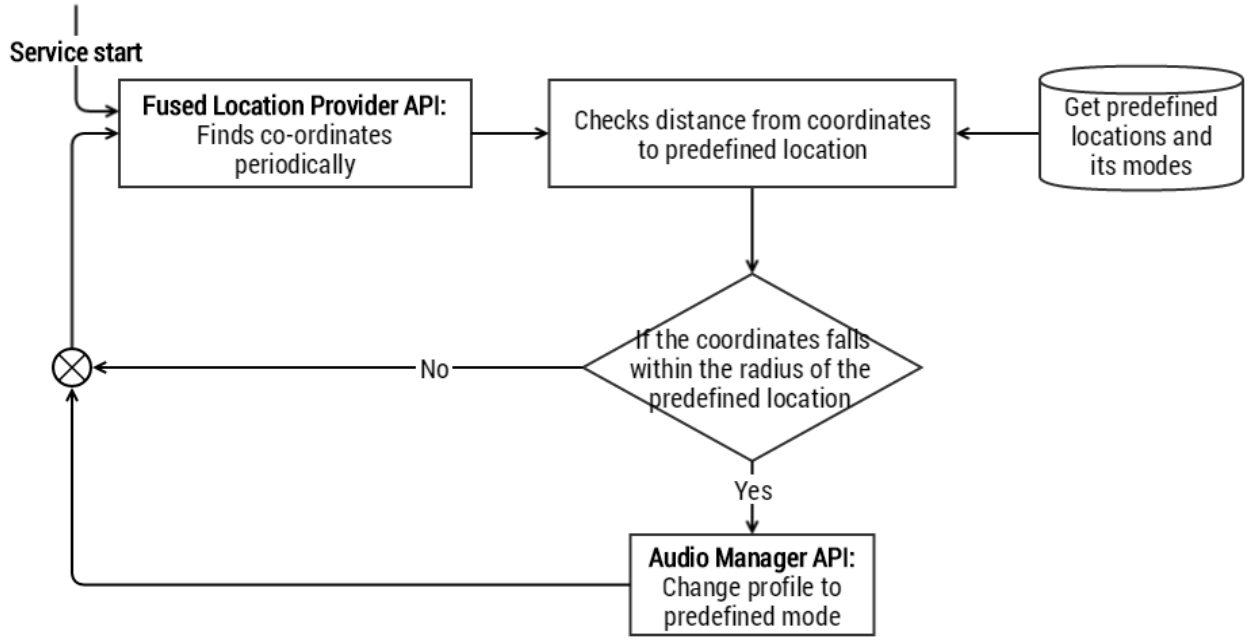


Fig. 14.1: Activity flow for Location based profiles

14.2 Driving Mode

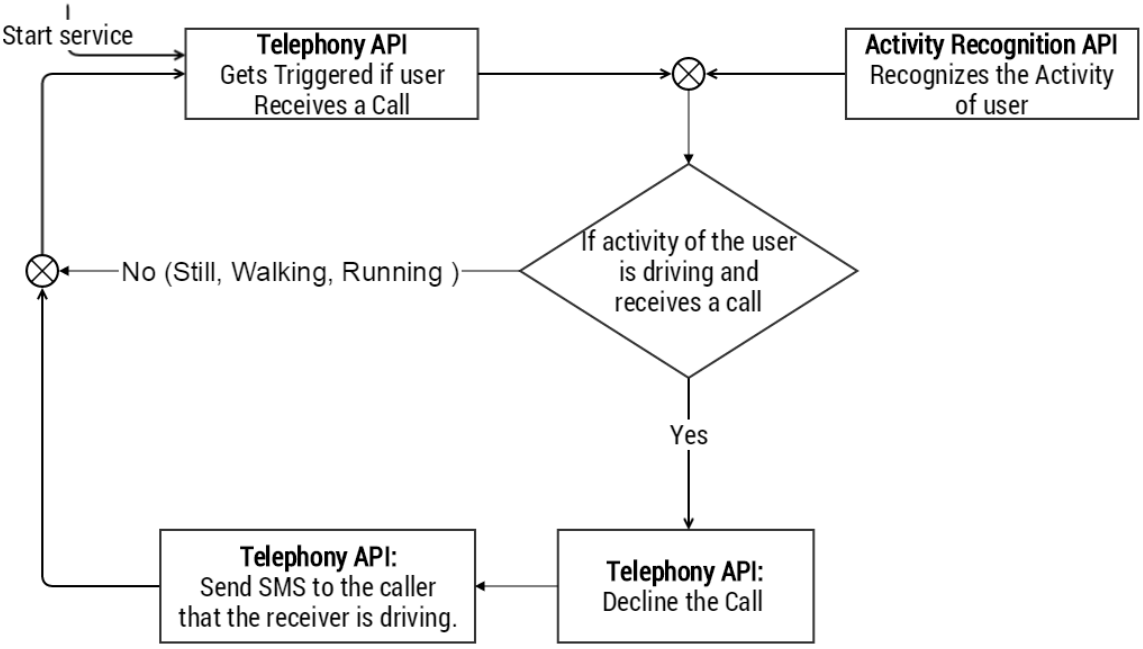


Fig. 14.2: Driving Mode Activity diagram

14.3 Event based Profile

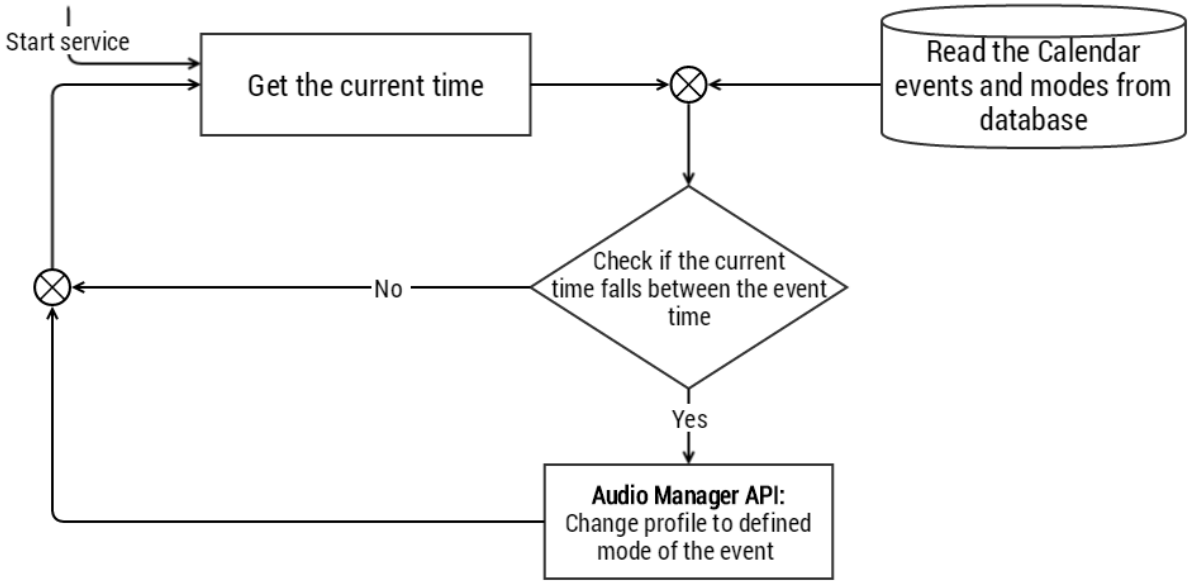


Fig. 14.3: Event based profile Activity diagram

14.4 Green Mode

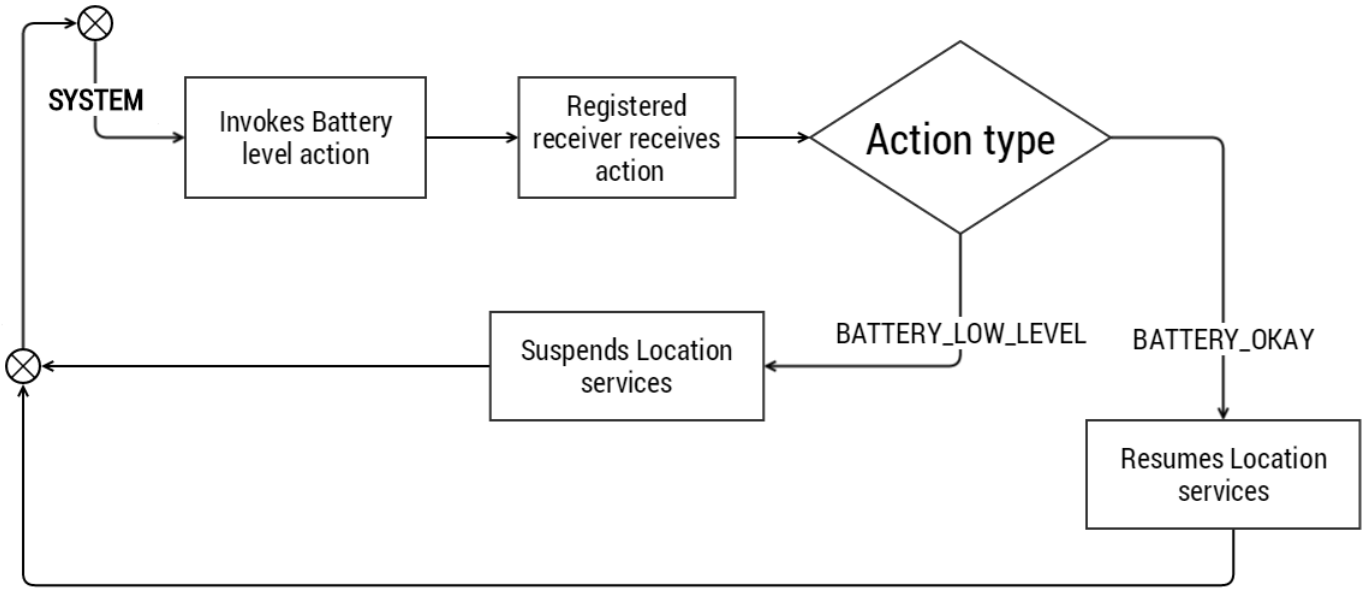


Fig. 14.4: Green Mode Activity Diagram

15. Data

15.1 Inputs

- Location and profile – Set by user.
- GPS coordinates – provided by API.
- Google Calendar event – created by user.
- Profile – Set by the user against the event.

15.2 Outputs

Profile change

15.3 Data Structures and Schema

- Location table – stores location id, coordinates and corresponding mode.
- Event table – stores event id, title, start time, end time, duration, type of the event (recursive or one time), status and profile.

Location Table Schema	
Column Name	**SQLite Data Type**
_ID	**Integer**
TITLE	**Text**
ADDRESS	**Text**
LATTITUDE	**Real**
LONGITUDE	**Real**
MODE	**Integer**
RADIUS	**Integer**

Event Table Schema	
Name	**SQLite Data Type**
_id	**Integer**
FOREIGN_EVENT_ID	**Integer**
TITLE	**Text**
START_TIME	**Integer**
END_TIME	**Integer**
DURATION	**Integer**
MODE	**Integer**
RECURSIVE	**Integer**
STATUS	**Integer**

16. Prototypes

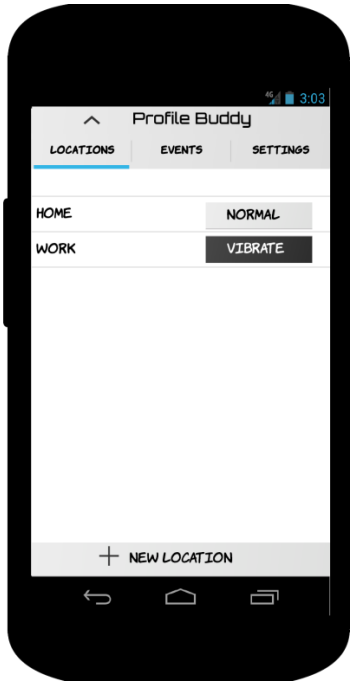


Fig. 16.1: Locations screen

Fig. 16.2: Events screen

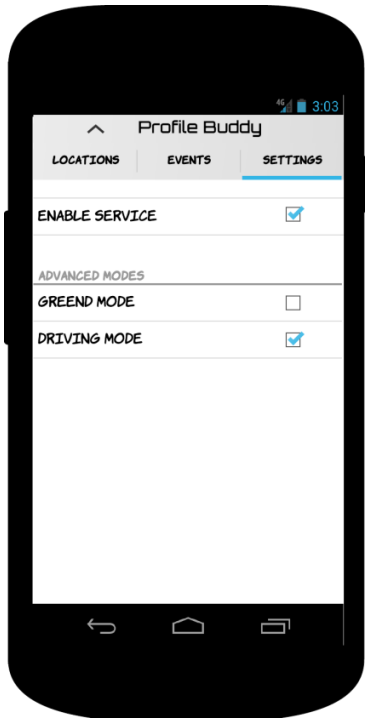


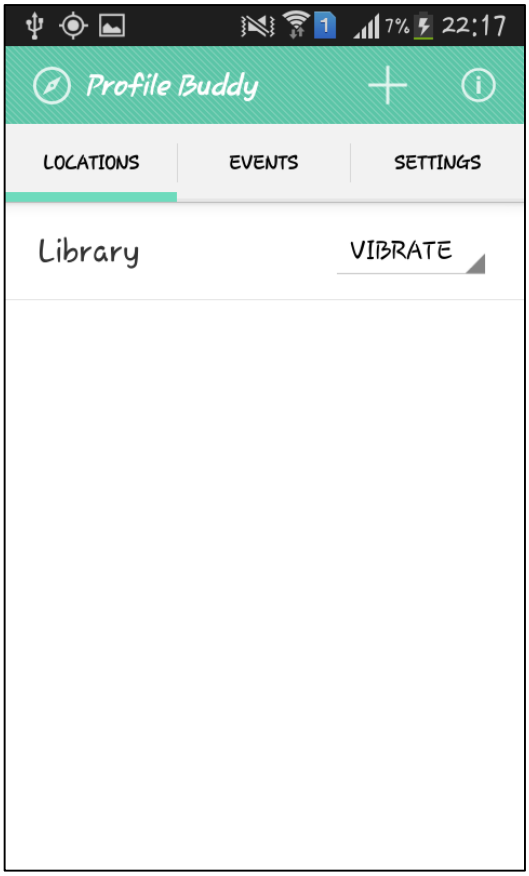
Fig. 16.3: Settings Screen

Fig. 16.4: Notification Screen

17. Application Screen Transitions

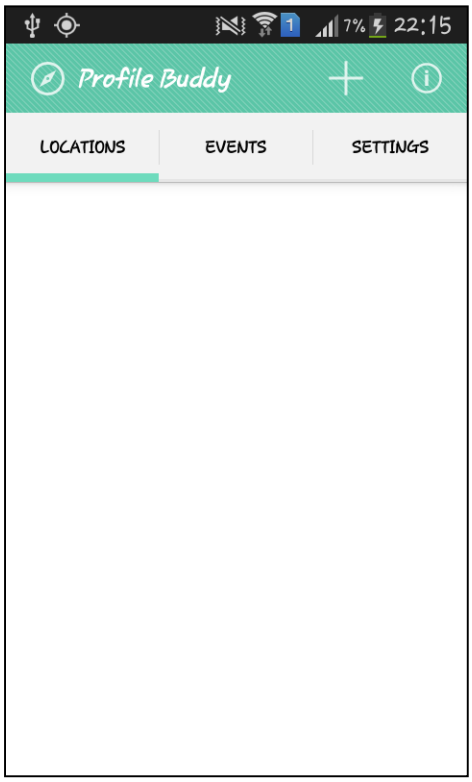
17.1 Main Activity Screen

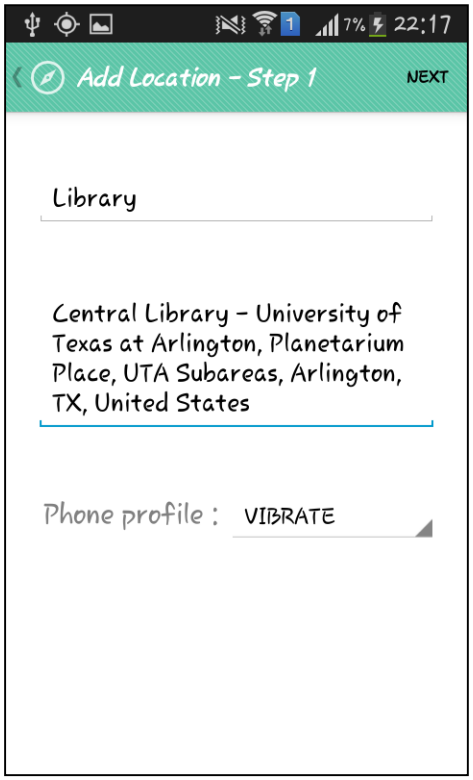
Splash Screen

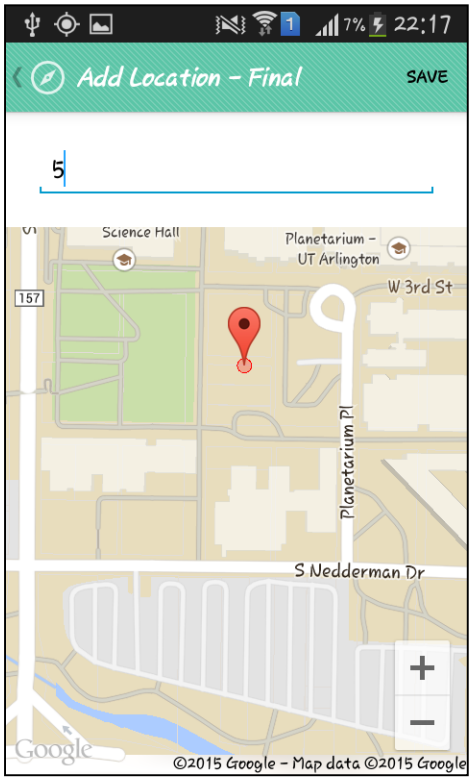


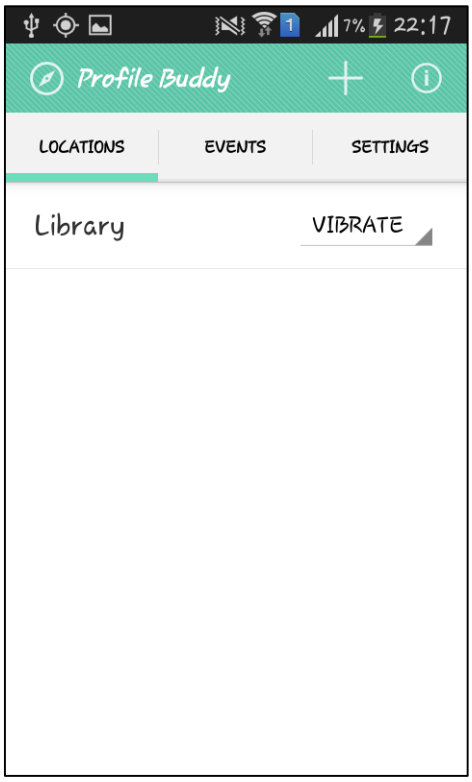
Locations Screen

17.2 Add Location profile

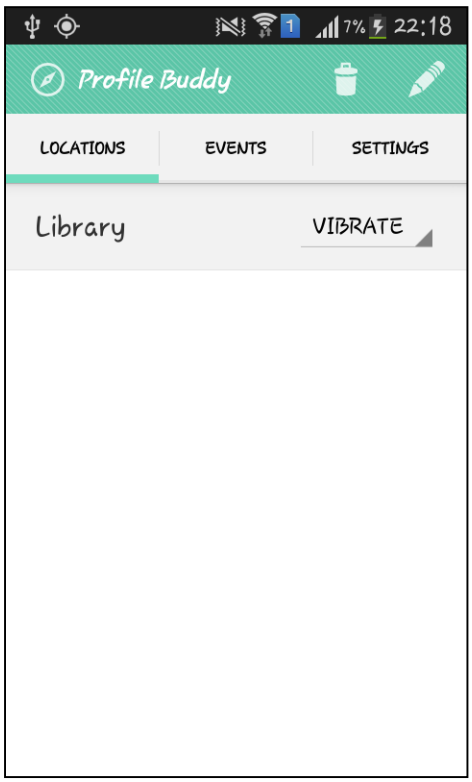


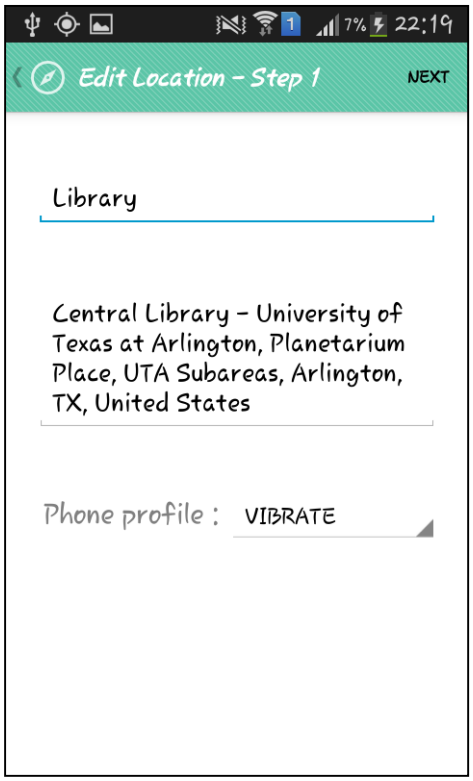


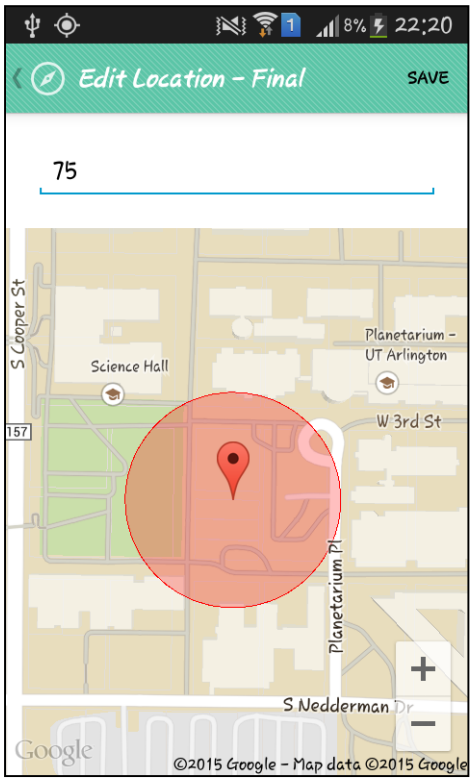


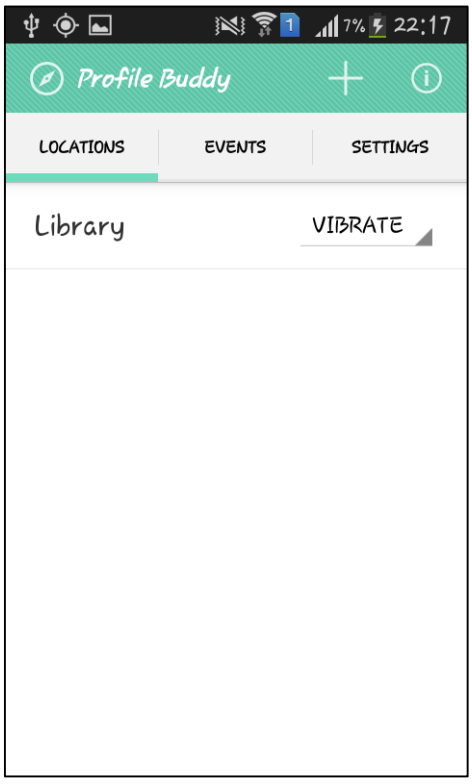


17.3 Edit Location profile

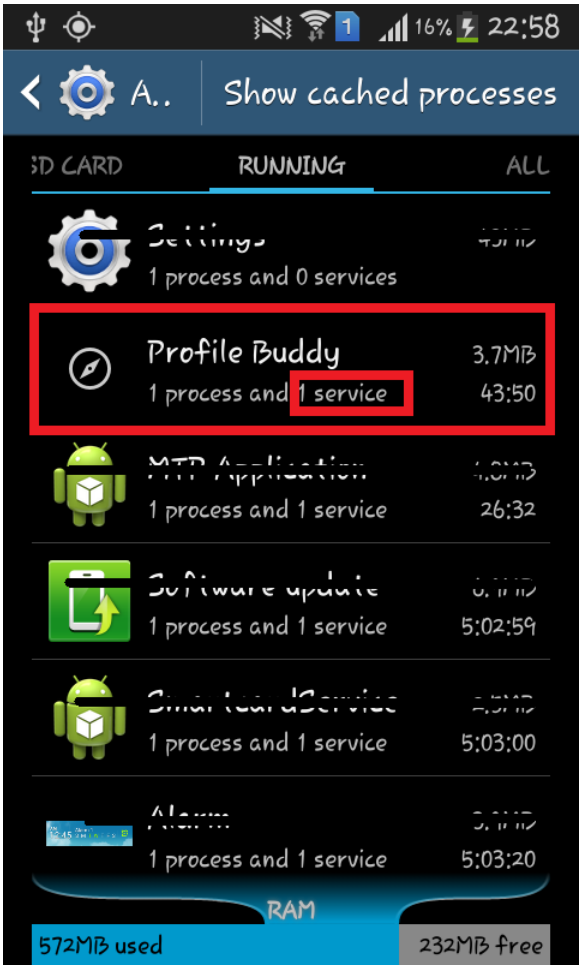




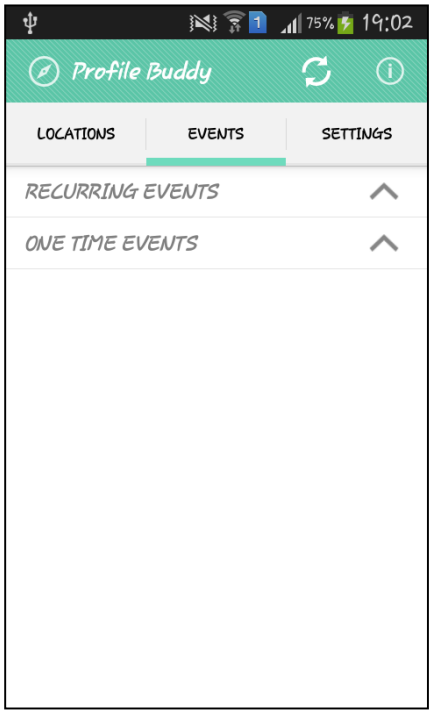




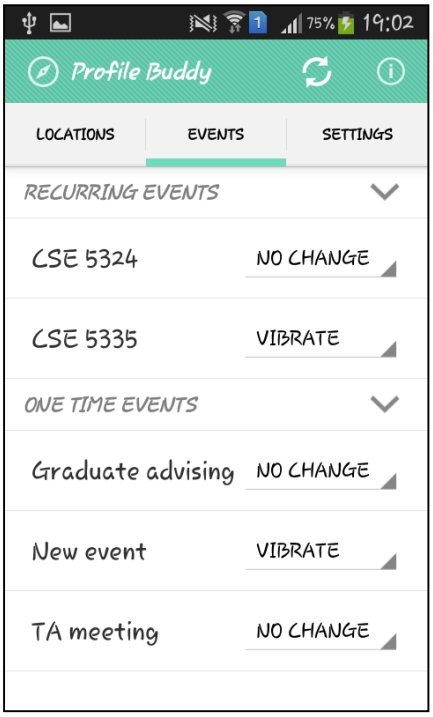
17. 4 Background service (part of application)



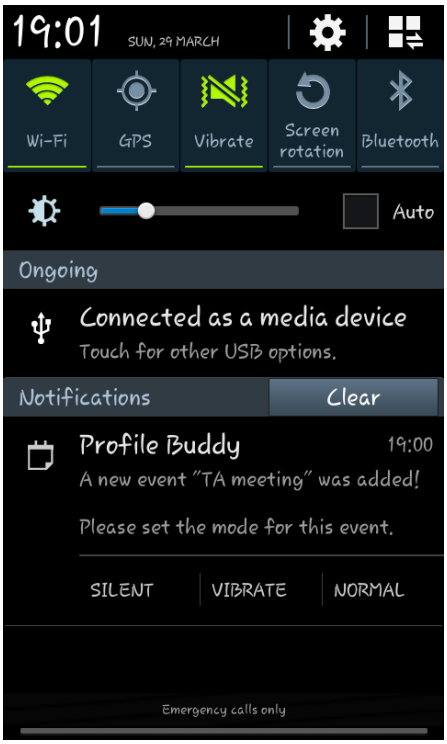
17.5 Calendar Events



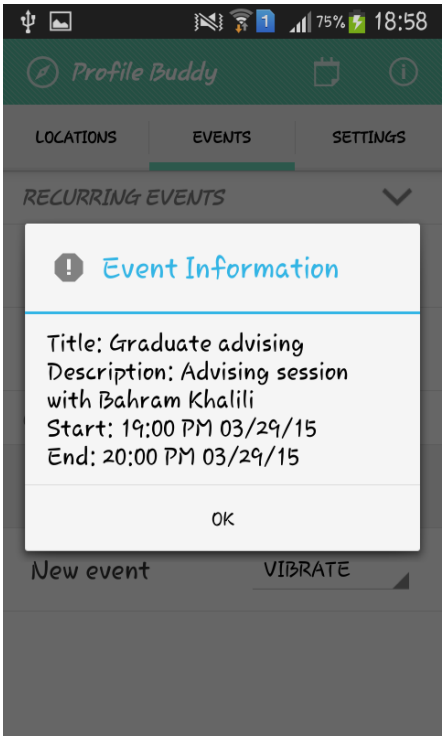
Event Screen Collapsed



Event Screen Expanded

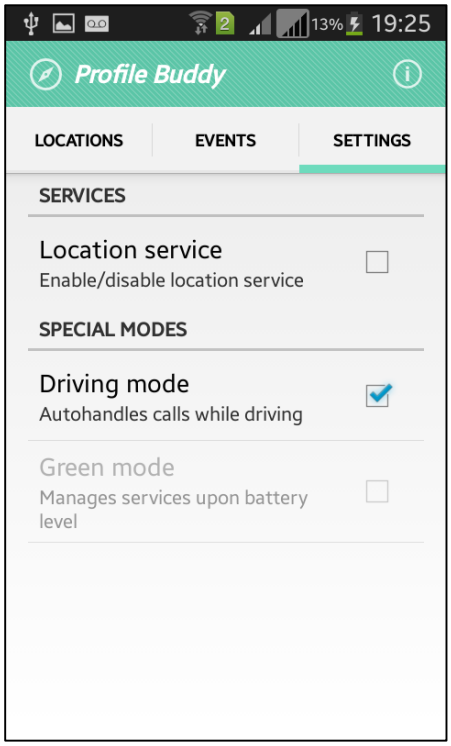


Push Notification

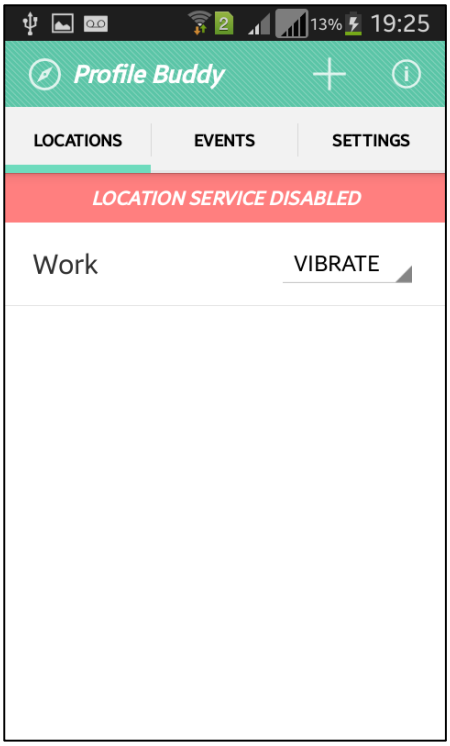


Event Information

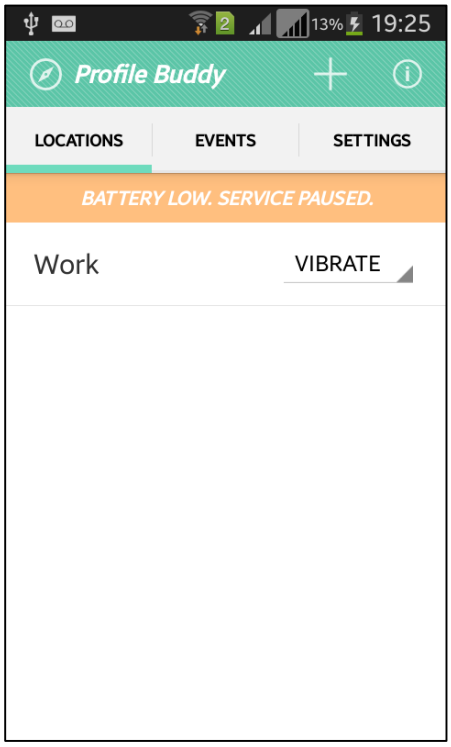
17.6 Settings – Green Mode and Location Services



Settings



Location Services disabled



Battery low. Service paused

18. Project Risks

18.1 Technical:

- Usage of GPS for a long time may drain your battery.
- If there is no data connection, then this app may be less effective.
- Differentiating between travelling in a public transport and in a personal vehicle. Predefined locations might have different areas, and using defined rule on them may reduce the effectiveness.
- There can be two events in the calendar set at the same time.
- Clash between Location based profile and Event based profile.

Mitigation Plan:

- Use event based profiles to reduce dependency on GPS, when on “Green mode”.
- Have a notification to let user know "Driving mode" is activated.
- User need to specify the radius for each predefined coordinates.
- Set the profile of the first event that was found.
- Use Event based profile in case of clash, which is more specific.

18.2 Resources:

- Lack of experience in android application development.
- Two team members are not from Computer Science background.
- Two team member’s works 20 hours/week.

Mitigation Plan:

- Each member assigned a few API’s related to the project to learn more about them and gain a good hand’s on before starting with the actual development
- Planning meets up and dividing work according to the schedule of the team members

19. Operating Environment

- **Software/OS:** Android OS
- **Hardware**: Any device capable of running android OS.

20. Assumptions and Dependencies

- Wi-Fi or mobile data connection is available.
- GPS should be enabled.

21. Technologies

- Java
- Android
- SQLite
- Eclipse / Android Studio.

22. Iteration Plan:

Features:
Iteration one:
Location Based Profile:
--Enter the address of the location and profile to set for it.
--Display the location in google maps and set radius of the location.
--running application in background (as service).
--Display all the predefined locations and corresponding profiles.
Iteration Two:
Calendar Event Based Profile
--Access event in the calendar and set profile for it.
Push Notification on Event creation
--Set profile for the event from notification bar.
Iteration Three:
Driving Mode
--Detects if the user is Driving.
--If user receives any call ends it and Sends the caller a custom message.
Green Mode
--Detects the Battery Level.
--Suspend profile buddy service if low battery level.
Additions and Improvements (Final)
Default mode
Splash screen
Broadcast based Event profile service

23. Setting up project in IDE

Assumptions:

- Eclipse Android plugin and SDK Manager are installed.
- API versions 14 – 21, Google Play Services Library, Android Support Library and Google USB Driver are installed using the Android SDK Manager.
- Profile Buddy code is available.
- Testing device has internet access.

Step 1 (Google Play services dependency setup):

- Click on “File” menu in Eclipse and select “Import”.
- In the import screen, select “Existing Android Code Into Workspace”.
- Click on browse and select directory “{Android SDK installation path}\android-sdk\extras\google\google_play_services” as the project root.
- Click on Finish to import the project into Workspace.
- Once the project is visible in the Workspace, right click on the project and select “Properties”.
- Select “Android” from the side list, and check “Is Library” option in the screen.

Step 2 (Profile Buddy project setup):

- Using the first two steps given in the Step 1, import Profile Buddy (as Android project) into the workspace.
- Right click on the project, and select “Properties”.
- Select “Android” from side list, and click on “Add” to add google_play_services_lib” as dependency library for the project.

Step 3 (Plug and run):

- Plugin the device on which you want the application to be run.
- Enable “USB debugging” in your device from Settings -> Developer options
- Right click on the project, select “Run As”, and click on “Android Application” to launch the application on your device.
- Setup is complete.

24. References

[1] **Llama** - Location Profiles, Google “Play Store” : Description. Retrieved January 31, 2015

[2] **Priyank Patil (August 03, 2012)**, Location based services, Location based services: Article. Retrieved Feb 06, 2015

[3]**Tarandeep Singh (April 05, 2013)**, AutomateIt, Automate everything on Android: Article. Retrieved January 31, 2015

[4] **Automagic Automation, Google “Play Store”**: Description. Retrieved January 06, 2015

[5] **Dan Moren (May 15 2013)**, Google I/O 2013 keynote, Location-based and notification APIs highlight new Google Play services: Article. Retrieved Feb 07, 2015

[6] **Location APIs, Google Play Service APIs**: Key Features. Retrieved Feb 06, 2015

[7] **Google Maps API Web Services, The Google Geocoding API**: What is Geocoding? Retrieved Feb 06, 2015

[8] **Google Calendar API** - Google Developers, Google Calendar API: Description. Retrieved Feb 06, 2015

[9] **Telephony API**, Android Telephony Manager API: Class overview. Retrieved Feb 07, 2015

[10] **Audio Manager API**, Android Audio Manager API: Class overview. Retrieved Feb 08, 2015