Software Engineering 2
Design Document

Danny Murphy - 100082602

Chanelle Richardson - 100017621

Daniel Banks - 100093180

Sophie Whiteside - 100052770

Table of Contents

1 Introduction

We were given the task to design and develop three UEA open day applications which will all reuse legacy software in the form of the Simple Android Application Framework (SAAF). These include:

- An activity application which provides information about events around the UEA campus and will allow users to set reminders.
- A location finder app, which will provide a list of points of interest around the UEA campus, provide a map and allow users to navigate to any location selected.
- A parking aid app which will help users find their car, track parking ticket information and alert the user if their ticket is about to expire.

2 Management Overview

The current system for the University regarding their Open Days is that any information given to prospective students and parents is in paper form, such as the activity schedule, maps and where to find food and drinks etc. These visitors receive large amounts of information which provides a starting point and a guide for them through the campus. They then use the printed information to complete their day of browsing around the University attending the various lectures and demonstrations that are available. Additionally, with the substantial number of visitors that attend UEA, there are various parking places the University use in different areas and at different points of the campus.

Reviewing the current system, it presently implements a structure where visitors do not have live and on-demand materials available to them to view on their mobile devices, allowing easier access to locate places or their cars on campus. It would also enable the University to not have to provide printed information.

A new system would implement a digital source providing them with access to locations from where they are to a selected point of interest, guide to food and drink on their mobile devices and a parking aid which takes account of your car in the parking space. Additionally, these apps will enable GPS allowing our visitors to use that to locate back to their car or where they would like to go e.g. Walking to a building to attend a lecture or introductory talk.

In the case of the Location Finder app, entering a search term into the search box or browsing through the list will allow the user to click on one of the points of interest. This will then display information about that particular place on selection and allow the user to display it on the campus map.

In the case of the Activity Program App, information with details of the activities that the School of Computing such as title, time and location of the event. To allow the user more flexibility, the app will enable the user to select which event they wish to attend and receive a reminder for it. The App will support different formats of these sources and will be presented in a formal and set way. This information will be loaded from an online location and an online calendar will be imported to integrate any updates.

In the case of the Parking Aid App, keeping track of a visitor's car using a GPS tag of the car and storing this information, as well as determine where the car is located relative to the user's current location. The app would display a visual indication including GPS tag, the current GPS location and the compass to give the direction of where the user should go. Additionally, the time of arrival when the user arrives at UEA should be stored to work with their parking ticket, keeping track of the amount of time remaining before the parking ticket expires, notifying the user on their device before the parking time expires.

As well as being to function on a mobile device, it will also be implemented for tablet devices. It will be executed differently where a tablet lists the points of interest and campus map in a single view and a phone switches between activities.

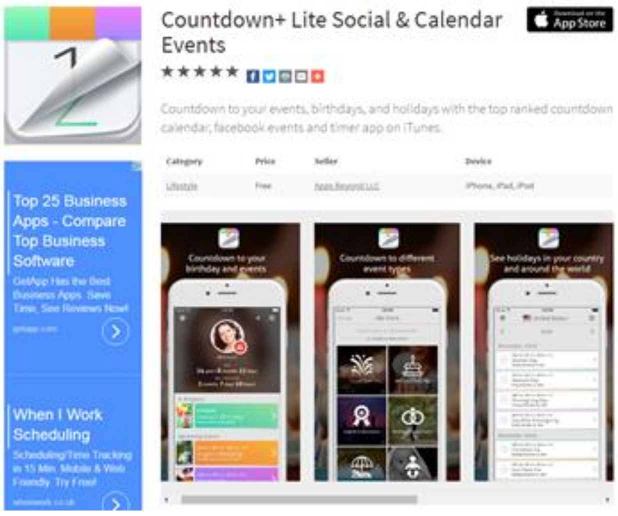
3 Requirements Analysis

3.1 Similar Systems

We are looking to create three mobile applications, which may be used as part of a “UEA Mobile Device Application” system in the future. Upon doing research we found that there aren’t any exact matches for a similar system, however there are many applications that fulfil some of our system requirements in part. Below are some of the examples we have researched.

3.1.1 Activity Program

3.1.1.1 Countdown+ Lite Social & Calendar Events



Countdown+ Lite Social & Calendar Events

Countdown to your events, birthdays, and holidays with the top ranked countdown calendar, facebook events and timer app on iTunes.

Category	Price	Seller	Device
Lifestyle	Free	Apps Beyond LLC	iPhone, iPad, iPod

App Store
Top 25 Business Apps - Compare Top Business Software
When I Work Scheduling
Countdown to your birthday and events
Countdown to different event types
See holidays in your country and around the world

Figure 1 Countdown+ Lite Social & Calendar Events

Description	
Countdown Social & Calendar App features countdown calendar, Facebook events and timer app on iTunes with unlimited ‘countdowns’ and reminders for all your events.	
Pros	**Cons**
- Free-to-download app - Calendar tab which syncs your devices’ calendar and events from Facebook (uses days, minutes, hours and seconds to show upcoming and previous events) - Event codes for every countdown - Login to your account and keep your countdowns safe and easy to download your countdowns from Android to iPhone - Receive notifications more frequently as the event approaches	- Upgrade causes a lot of errors such as incorrect countdowns - The app contains advertisements (However advertisements can be removed by paying for the full application.)

3.1.1.2 Calendar Event

Figure 2 Calendar Event

Description	
Search and order from available of local restaurants, and pay by card, or cash on delivery, or collection	
Pros	Cons
- Free-to-download app - Can override or replace default reminder with "unmissable" ones - Events will ring like a phone call for desired length of time, with their own choice of melody - When installed, will automatically synchronize events and reminders - Available on various smartphones and tablets	- Cannot trust it to trigger on every calendar event notification - Notification for all day events doesn't respect timezone settings - Does not work in conjunction with social media events - User interface not as sophisticated as could be

3.1.2 Location Finder

3.1.2.1 Family Locator

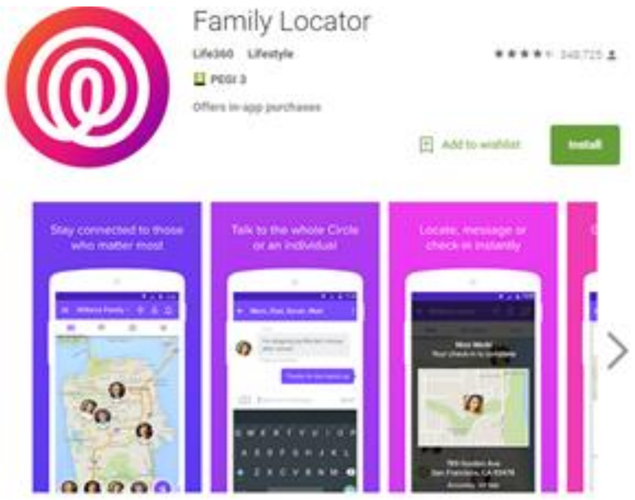

Family Locator
Life360 Lifestyle
PEGI 3
Offers in-app purchases
Add to wishlist
Install
Stay connected to those who matter most
Talk to the whole Circle or an individual
Locate, message or check in instantly

Figure 3 Family Locator

Description	
Family locator & children safety app using GPS Tracker, using instant messages.	
Pros	**Cons**
- Instant messages. - SOS button will immediately send recipients the users exact GPS location. - Set safe and unsafe zones on the map and receive notifications if they leave those zones	- Accuracy – location of entities are not correct - Tracking – inaccurate updated tracking data - Only one device can have the same account

3.1.2.2 Find My Car

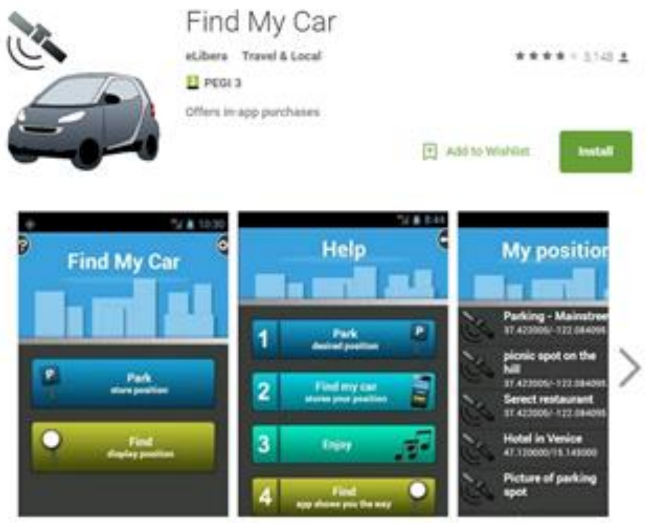


Figure 4 Find My Car

Description	
A car tracking application	
Pros	Cons
- Button to mark the location of your car and GPS tracker of your current location - Free-to-download app - GPS features are very efficient, offering fast, effective tracking tools.	- Number of ads integrated into its design - No date and time stamp for these markers - No way to save locations for future reference

3.2 System Requirements

To analyse the requirements of our chosen applications, we used the MoSCoW system to sort the key requirements by their priority.

3.2.1 Activity Program MoSCoW

Must:

- Show a list of schools
- Show activities available for a selected school
- Each activity must show title, time and location of the event
- Load activity programs from local storage
- Select open day date
- Set one or more reminders by checking a box for each activity
- Pull information from an online location and store in local storage
- Synchronize to most updated activity program
- Have settings to change the synchronization interval (how often the app updates information)
- Must generate a notification 10 minutes before all selected activities
- Users able to access settings from the app

Should:

- Notify user of updates and recent changes
- Be easily extensible with programs for other schools
- Have a setting to choose a default activity program to open when the app is launched
- Have a generic mechanism to deal with online calendars
- Be presented in an easy to read format
- Give description of a selected activity

Could:

- Have a setting to set notifications to sound and/or vibrate
- Have a configurable background image for each school program
- Have a location where the online calendar for a selected school can be found
- Aggregate all selected activities to an easily accessible list on the app

Won't:

- Allow creation or editing of activities
- Transfer activities to an external calendar (e.g. Google calendar)
- Allow users to store any additional data such as images and notes
- Restrict data from being loaded if the user's connection is poor

3.2.2 Location Finder MoSCoW

Must:

- Contain a list of locations on UEA Campus and their locations on the UEA Campus Map.
- Allow users to browse a list of locations
- Have a description for each location
- Allow users to search for locations
- Selected locations must open a campus map
- Have zoom in and out functionality on the map
- Easily configurable input data
- Read location from a textual input file that can be updated
- Input file contains all points of interest, descriptive tags and location coordinates

Should:

- Get the user's current location and provide directions to a selected point of interest
- Provide a visual route to a selected point of interest
- Users able to favourite locations to put them at the top of the list of locations

Could:

- Include support for displaying disabled access routes on campus
- Optimise user interface for different device types and screen sizes (e.g. tablets)
- Access recently selected locations
- Provide an optional textual route to a selected point of interest
- Display an image of the location with the description

Won’t:

- Add custom locations from outside the list of locations provided
- Provide verbal directions
- Display notifications when the app is not in use
- Utilise any map views other than the top down roadmap (e.g. street view, satellite)

3.2.3 Parking Aid MoSCoW

Must:

- Store car location via GPS tag and parking time
- Give visual indication of where the car is located relative to the user's current location
- Keep track of the current location and rotation of the user's device
- Continually update the compass to accommodate for the rotation and orientation of the device relative to the vehicle location

Should:

- Give users the ability to add a note to describe where they parked (e.g. 2nd level)
- Track parking ticket information given by users (e.g. expiration time)
- Notify the user before their parking ticket expires
- Allow users to set the amount of time, before ticket expiration, that they want to be notified

Could:

- Display vehicle location on a map
- Give directions from the user to their vehicle on a map
- Display a fixed timer in the notification tray showing time remaining on the parking ticket
- Give an audio notification when the parking ticket has expired

Won't:

- Support devices without GPS and gyroscope functionality
- Track multiple vehicles

3.3 Assumptions

The following general assumptions were supplied to us in our application requirements:

- Each application will be developed for the Android platform
- Each application will be based on the Simple Android Application Framework
- Each application is a stand-alone application with its own icon
- Each application must consider multiple resolutions and form-factors
- You are free to target phones, tablets or both

In addition to the general assumptions listed above, our group has also assumed that we may implement any method of saving and loading information from local storage, for our initial design we have assumed that we will be saving and loading information from text files stored locally.

4 Use Case Diagrams and Descriptions

The included use case diagrams and descriptions outline the basic actions for both success and alternative scenarios. Both diagrams outline who triggers the action and how the system deals with the request.

All the use case diagrams can be found at Appendices A1 though A3 and all the use case descriptions can be found at Appendices B1 through B12

5 Class Diagrams

5.1 Original SAAF Class Diagram

The original Simple Android Application Framework, SAAF, class diagram was reconstructed by reverse engineering legacy code given to us to analyse. The class diagram, seen at Appendix C1, shows the structure and behaviours of the SAAF alongside the Robot Game Application that implements it. The task of creating this class diagram from the given legacy code was intended to teach us how to analyse old systems for future revision and reuse.

Below are descriptions of some of the SAAF classes that the Robot Game Implements. We have also analysed some Android classes to gain further knowledge.

5.1.1 Assets

The assets class uses the getAudio() function which is part of the AndroidGame framework class, the createMusic() function from the AndroidAudio framework class and also the various music settings methods (setLooping(), setVolume(), play()) from the AndroidMusic framework class.

GameScreen, LoadingScreen, MenuLoadingScreen & SplashLoadingScreen

All of these classes utilise the abstract class Screen found in the framework. This gives each of them the ability to provide their own implementation of each of the functions (update(), paint(), pause(), resume(), dispose() and backButton() as well as a generic constructor.

5.1.2 Tile

The tile class contains a variable tileImage which is of type Image. Image is found in the SAAF framework and provides general image methods such as (getWidth(), getHeight(), getFormat() and dispose().

5.1.3 SampleGame

The sample game class is the main class in the robot game application. It uses the AndroidGame framework class which contains a generic onCreate() method which builds the basics of the app and initialises all the key variables, a few basic functions to determine what will happen during each of the android application states (onResume() and onPause()) and some getters and setters to provide access to the many components such as input, fileIO, graphics, audio and screen.

5.1.4 AndroidAudio

This class allows applications to create music and/or sound by passing in a valid mp3 file.
The file is then opened by an AssetFileDescriptor which creates an object that is passed to create a new AndroidMusic object. From here a new MediaPlayer object is created and the fields from the AssetFileDescriptor object are used to get the music data source, start offset and the length. In the robot game this is first used when loading assets into the game.

5.1.5 AndroidMusic

This class provides some simple music controls such as pause, play, stop, setting the music to loop and setting the volume.

5.1.6 AndroidSound

This class provides controls to play and delete sounds.

5.1.7 AndroidGraphics

This class provides tools to load in images, clear the entire screen and draw simple rectangles and lines with given attributes such as height, width and colour.

5.1.8 AndroidGame

This class aggregates all the generic components in an application such as graphics, audio, screen and input and allows access to each.

5.2 Revised SAAF Class Diagram

Once our analysis of the original SAAF and Robot Game was completed, we then had the task of reconstructing the framework to suit our applications needs. The revised SAAF class diagram (Appendix C2) outlines what classes we believe we can reuse within out applications as well as new classes that can be used across all our applications.

5.3 Application Class Diagrams

There are three class diagrams that represent each application. The class diagrams outline any classes that are needed outside of the SAAF framework, as well as classes that will implement existing classes within the framework. As the project gets further into development stages, there may need to be changes to these class diagrams to better represent the systems.

5.3.1 Activity Program

The class diagram for the Activity Program outlines extra classes that this application will need to implement outside of the SAAF such as main activity screens. Many of the classes also implement or extend existing classes within the SAAF to reuse given code.

To view the Activity Program class diagram, please refer to Appendix C3.

5.3.2 Location Finder

The Location Finder application needs classes outside the SAAF to display different information. Like the Activity Program Class diagram, many of the classes used within the Location Finder application also implement or extend existing classes with the SAFF such as the FileIO class.

To view the Location Finder class diagram, please refer to Appendix C4.

5.3.3 Parking Aid

The Parking Aid class diagram utilises classes with the SAFF by extending or implementing them within its own classes. For example, the RunApp class extends the Screen class within the SAFF to display activities.

To view the Parking Aid class diagram, please refer to Appendix C5.

6 Sequence Diagrams

6.1 Robot Game

This sequence diagram (Appendix D1) models how the Robot Game application classes interact with the SAAF framework. In particular this sequence diagram outlines how the getInitScreen() method within the SampleGame class initialises a new splash loading screen when the application starts up. The method uses the SAAF classes AndroidGame, AndroidMusic and AndroidAudio to pass in a music file to the application, set options to play, loop, and change the volume of the song.
Developing this sequence diagram further enforced our knowledge of how applications interact with the SAAF and how to properly analyse legacy systems.

6.2 Activity Program

The Activity Program sequence diagram (Appendix D2) shows the processes the system must complete to display a list of schools to a user.

6.3 Location Finder

The Location Finder sequence diagram (Appendix D3) shows the processes the system will complete to display the profile of a location. The location profile consists of a title and a description of the location, therefore the information will need to be retrieved from where it has already been stored in memory, inside the SaveLoadLocations lass that extends the FileIO class within the SAAF. We have assumed that all data has been read and stored into memory when the application starts up.

6.4 Parking Aid

The Parking Aid sequence diagram (Appendix D4) shows how the application classes interact with the SAFF to save information about the users parking ticket and where a user has parked their car. This data will be stored within the devices local storage. The SAFF class that this application interacts with is the FileIO class, implemented by our SaveLoadParkingInformation class, as this can use an output stream to write to a text file.

7 State Transition Diagrams

7.1 Activity Program

This diagram shows the main states of the Activity Program application. The entry point for this diagram is when the application is launched, the system will then update the data if required and continue to the school list screen. From here the user can access the settings page as well as view activity programs in more detail.

To view the Activity Program state transition diagram, please refer to Appendix E1.

7.2 Location Finder

This diagram shows the main states of the Location Finder application. The entry point for this diagram is when the application is launched, from there it will load in the location data from a file and display it as a list of locations. From here the user can search the list and select a location to view in more detail. From the location details screen the user can press a button to view the location on a campus map.

To view the Location Finder state transition diagram, please refer to Appendix E2.

7.3 Parking Aid

This diagram shows the main states of the Parking Aid application. The entry point for this diagram is when the application is launched. The app launches straight to the home page where the user has 2 options, add parking info and navigate to car. If the user selects ‘navigate to car’ when there is no stored location information, it will produce an error and return them to the home page.

To view the Parking Aid state transition diagram, please refer to Appendix E3.

8 Application Architecture

The application architecture diagram is listed at Appendix F. Below is a description of what the architecture diagram outlines.
Within the application, the system acts as the controller, switching between the screens and allowing them access to the Model. Inside each screen is a table layout (implemented externally) which contains all the elements to be displayed, these are loaded in using the graphics loader to create the desired view. The screen acts as a controller choosing which data to load into which elements using its parents' access to the Model. The Model contains a Location Manager for getting the current GPS location and storing lists of locations, DataObjects which represent a hierarchical data structure and the Data Input/Output layer which allows for loading and saving of data to and from different source types (using strategy pattern). This is also where the standard file loader is located for working at a lower level with input and output streams.

9 Component Diagram

The component diagram, see Appendix G, consists of five main components. One component is nested inside the Data Layer Component as it represents a subsystem for loading and saving data. The Data Layer consists of location management and data types. The Audio Layer allows access to different audio types that might be required. The Graphics layer allows easier access/creation of graphical elements such as TextViews and a GoogleMap Etc. The Application component contains a screen which uses its parent Application to get access to the external components of the system (mentioned above) that it needs. The Application may at any point swap out the current screen for another which will then use the application to access the external components it needs again.

The Three Application diagrams are extremely similar as the framework is designed to be used in the same way for each. Any significant task the screens may need to perform such as loading in data, getting the GPS location or constructing a Google maps element; are taken care of by the framework itself and do not need to be implemented in the individual applications. This is an example of how the framework reuses code for different applications. To use the framework, the user must create a new screen class which will load in the data from the required source, using the Data Input/Output manager this data is then used to construct the elements it needs using the graphics class.

Both the Graphics and Data Input/Output manager are accessed using the contained parent applications interface. This will usually be done in the constructor of the screen class and/or on the resume method. The paused method may be used to dispose and reconstruct the elements, in case the application is closed temporarily. The only additional work usually required will be methods for the specific tasks a screen will need to perform (such as opening a new screen or saving information). The application interface allows access to a root table layout which allows for a shared area of the application between screens. This means tabs and a search bar can be left on the screen and only the elements within the tab are cleared and reconstructed when a screen is changed. This also allows the screen to switch the tabs selected.

10 User Interface

Initial user interface designs were created using Moqups, an online design development tool. As we are in the early development stages of our applications, we have kept all three initial designs simple as it is highly likely that these designs may change in the future. We also wanted the applications to be simple to understand and use for those with all technical skills. We are aware that family members such as parents often visit UEA with their children during open days and that they may want to download our applications too, we don't want to make the applications unnecessarily complicated by assuming everyone using them has a high level of technical understanding.

All three user interfaces for our applications can be viewed at Appendices H1 though H3.

We have initial wireframe designs for each of the apps to give a better picture of how the user will navigate through them and help to allocate space to each element.

For all the apps, a notification will pop up whenever a button action is triggered such as (starting tracking, setting reminder etc.) to provide confirmation to the user. The same is true if any errors occur.

10.1 Activity Program

On the home screen, we have decided to use a date picker, to allow the user to select the open day and the school on the same activity thus reducing the amount of activities the user must navigate through to reach the information they need. A potential downside to using the date picker is it has been known to not work as well on some devices, therefore during implementation this may have to be changed to two separate select screens.

10.2 Location Finder

On the location information screen, there will be a button to bring up the map and provide navigation to the user. This feature will open another activity so that the map can be viewed in full screen. On a tablet, due to its larger screen it will be possible to have all the information including the map on the same activity. We have considered having a static location map which when clicked on enlarged it to provide an interactive map on phones would be another option.

10.3 Parking Aid

In addition to the designs there will also be notifications and the timer will be visible on the users lock screen. A reset/clear button has also been added in case the user inputs the wrong information by mistake and wishes to try again. The timer may be colour coded to transition from green to red as the timer starts to run out.

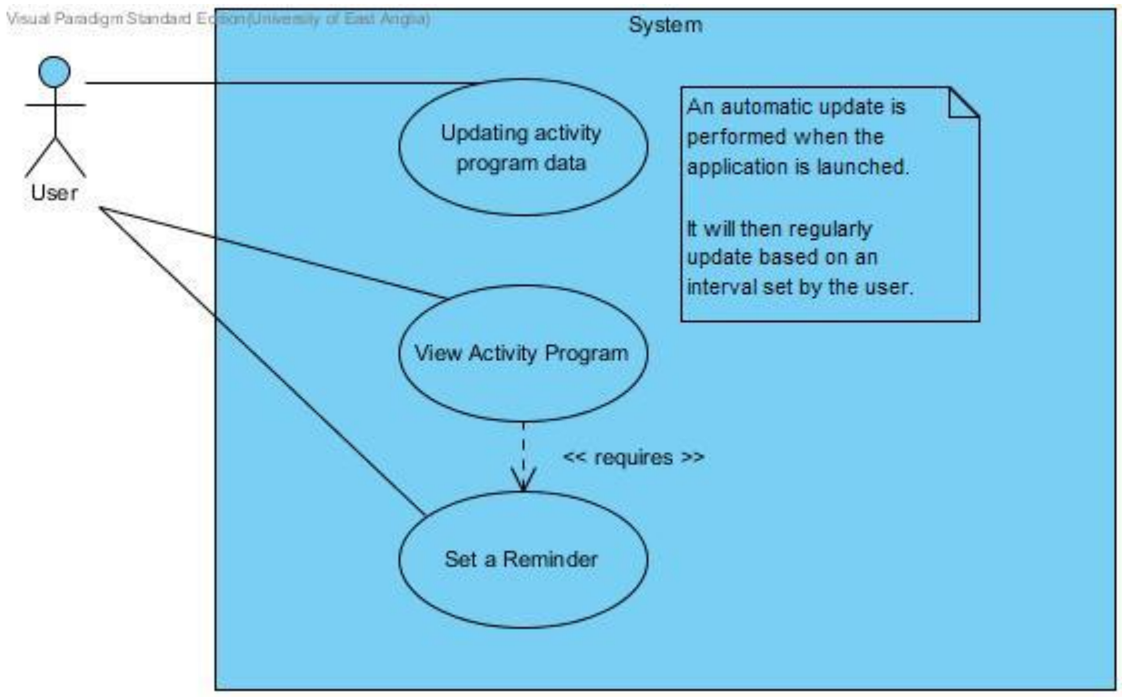
11 Summary

Our designs meet the given requirements for UEA Open Day applications. We have incorporated our research of similar systems and of the Robot Game SAAF into each design. The designs have included reuse wherever possible across the three applications; although each has its own unique features. The user experience has been considered to allow for quick and efficient use of the applications. The designs keep within scope of simple applications with added extras. Overall, all stakeholders have been considered when designing the applications to ensure each application is simple, but effective.

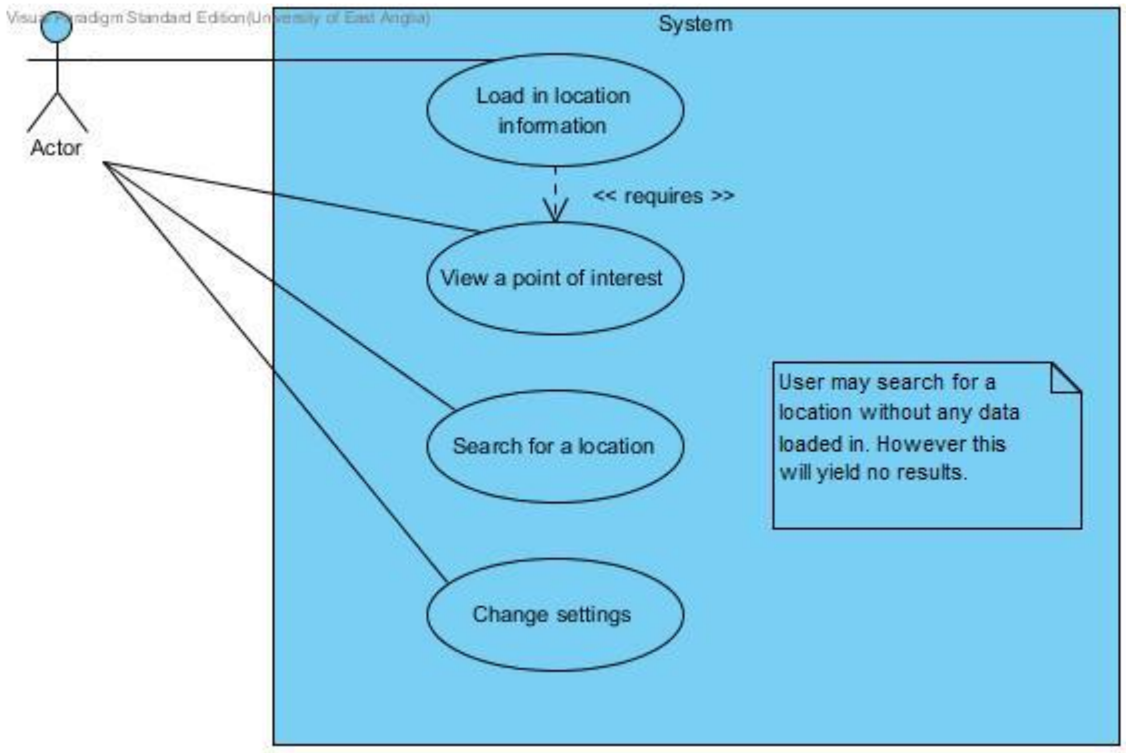
Appendices

Appendix A Use Case Diagrams

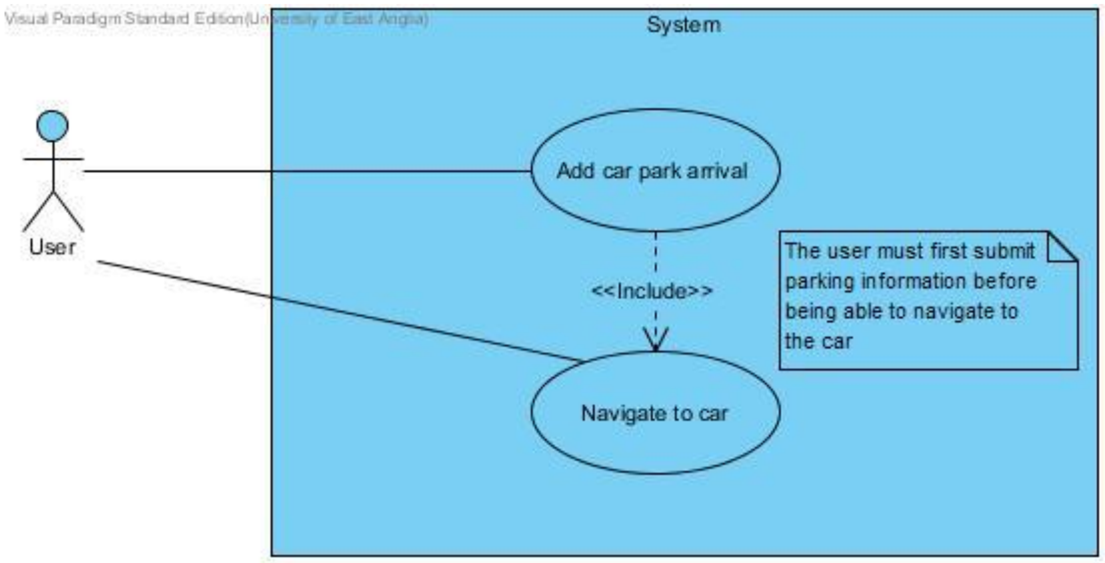
Appendix A1: Activity Program Use Case Diagram


Visual Paradigm Standard Edition(University of East Anglia)
System
User
Updating activity program data
An automatic update is performed when the application is launched.
It will then regularly update based on an interval set by the user.
View Activity Program
<< requires >>
Set a Reminder

Appendix A2: Location Finder Use Case Diagram


Visual Paradigm Standard Edition(University of East Anglia)
System
Actor
Load in location information
<< requires >>
View a point of interest
Search for a location
User may search for a location without any data loaded in. However this will yield no results.
Change settings

Appendix A3: Parking Aid Use Case Diagram


Visual Paradigm Standard Edition(University of East Anglia)
System
User
Add car park arrival
<<Include>>
Navigate to car
The user must first submit parking information before being able to navigate to the car

Appendix B Use Case Descriptions

Appendix B1: Update Activity Program Data

USE CASE NAME	Updating Activity Program data	
Goal in Context	To allow the application to update data for the activity programs	
Scope & Level	Overall System	
Preconditions	Application is running and application has internet access	
Success End Condition	Activity program data is updated	
Failed End Condition	Activity program data fails to update	
Primary Actor	User	
Trigger	The user launches the application.	
SUCCESS SCENARIO	**Step**	**Action**
	1	The system checks if the set update interval had passed.
	2	If the update interval has passed, the system checks if the current version number matches the online source.
	3	If the version numbers do not match, the system fetches the updated data from the online location
	4	The system stores the data locally
	5	The system imports the data into the application
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	1a	The update interval time has not passed yet so the system does not need to update.
	2a	The version numbers match so the data is already up to date.
	3a	System fails to fetch the updated data
	3b	Application displays an error message to the user stating that it failed to fetch updated data.
	4a	The systems storage is full
	4b	Application displays an error to the user stating that it failed to update due to low storage space.
RELATED INFORMATION		
Priority	High Priority	
Performance Target	Display the location on the map in < 2 seconds	

Frequency	Depends on user setting – This action is performed periodically depending on the interval set by the user.
OPEN ISSUES	- What happens if the systems firewall blocks the update?
SCHEDULE	Release Build
AUTHOR	Daniel Banks (CMP) – 21/10/2016

Appendix B2: Activity Program – Change settings

USE CASE NAME	Activity Program – Change settings	
Goal in Context	Change settings within the application	
Scope & Level	Overall System	
Preconditions	Settings menu has been selected by the user	
Success End Condition	Application Settings have been updated and saved.	
Failed End Condition	Application settings have not been updated and saved.	
Primary Actor	User	
Trigger	Settings menu has been selected by the user	
SUCCESS SCENARIO	**Step**	**Action**
	1	User opens the application and selects the Settings menu
	2	User selects a setting they want to change
	3	User selects the pre-set options or enters in information.
	4	System saves any changes on exit of the Settings menu
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	1	User opens the application and selects the Settings menu
	2	User cannot change any settings / Users entered changes do not save
	3	Application throws an error and uses default settings.
RELATED INFORMATION		
Priority	Top Priority	
Performance Target	Should be completed within 10 seconds.	
Frequency	Infrequent	
Subordinate Use Cases	-	
Channel to Primary Actor	User Interface	
OPEN ISSUES	-	
SCHEDULE	Release build	
AUTHOR	Chanelle Richardson 22/10/2016	

Appendix B3: Activity Program Set a Reminder

USE CASE NAME	Set a reminder	
Goal in Context	To allow the user to set a reminder for one or more activities	
Scope & Level	Overall System	
Preconditions	The user has checked at least one activity on the list	
Success End Condition	A reminder will be set for all the selected activities and the app will notify the user when the activity is coming up	
Failed End Condition	The user is presented with an error message detailing the problem	
Primary Actor	User	
Trigger	The user wants to set a reminder for an activity	
SUCCESS SCENARIO	**Step**	**Action**
	1	The user checks checkboxes next to all the activates they want a reminder to be set for and then presses the "set reminder" button
	2	The application stores the date/time for each activity and sets a reminder
	3	When an activity is coming up soon the application will notify the user with the activity name and the start time
	4	The user can then press the notification to view more information
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	2a	The application generates an error message because no activities were selected and displays it to the user
RELATED INFORMATION		
Priority	High priority	
Performance Target	The reminders should be set in <1 second	
Frequency	This use case will be used very frequently	
OPEN ISSUES	How much time before the activity is due to begin should the app notify the user? Should this be set by the user?	
SCHEDULE	Release Build	
AUTHOR	Danny Murphy (CMP) – 21/10/2016	

Appendix B4: View Activity Program

USE CASE NAME	**View Activity Program**	
Goal in Context	To allow the actor to view an activity program for a selected school	
Scope & Level	Overall System	
Preconditions	Application is running	
Success End Condition	An activity program is presented on screen to the actor	
Failed End Condition	Application does not display an activity program	
Primary Actor	User	
Trigger	The user wants to view an activity program	
SUCCESS SCENARIO	**Step**	**Action**
	1	The user selects a school from the list of schools displayed
	2	The application displays a list of Open Day dates for the selected school
	3	The user selects an open day date
	4	The application loads the activity program for local storage
	5	The application displays the activity program on the screen
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	2a	The application was unable to find any open days for the selected school
	4a	The application is unable to find the activity program in local storage
	4b	Display an error message stating that the file was not found, and give the user the option to synchronise with the online source.
RELATED INFORMATION		
Priority	High Priority	
Performance Target	Load the activity program from local storage and display it to the user in < 2 seconds	
Frequency	High – Viewing activity programs is the main function of the application	
OPEN ISSUES	- What happens when the user's storage is full?	
SCHEDULE	Release Build	
AUTHOR	Daniel Banks (CMP) – 21/10/2016	

Appendix B5: Location Finder Loading in Location Information

USE CASE NAME	Loading in location information
Goal in Context	Load in location information which can be updated over time with new information
Scope & Level	Overall System

Preconditions	The application must be running	
Success End Condition	All location data will be imported/updated	
Failed End Condition	No new location data is loaded and the previous data is kept. An error message is displayed to the user.	
Primary Actor	User	
Trigger	The user wants to update or import the location data into the application.	
SUCCESS SCENARIO	**Step**	**Action**
	1	User presses the "Import location data" button on the home screen
	2	User selects a textual input file containing location data from a file explorer menu
	3	User confirms the file and presses the "import/update" button
	4	The system imports the data and displays a confirmation window to the user
	5	The application will update to show all the new locations
ALTERNATIVE SCENARIO	**Step**	Branching Action
	4a	There was an error with the file chosen (wrong file type, corrupted data etc.)
	5a	The system is not able to import/update the location data and displays an error message to the user detailing the problem
RELATED INFORMATION		
Priority	High priority – The application will not function without the ability to import location data. This is a key feature	
Performance Target	The data must be imported in < 3 seconds.	
Frequency	Low frequency – Location data is unlikely to change often so this feature will most likely only be used on initial use or when a new location is added	
OPEN ISSUES		
SCHEDULE	Release Build	
AUTHOR	Danny Murphy (CMP) – 27/10/2016	

Appendix B6: Location Finder Loading the Map

USE CASE NAME	Location Finder – Viewing the map
Goal in Context	Display map
Scope & Level	Overall System

Preconditions	Application is running, user has chosen a location to view on the map.	
Success End Condition	Map is displayed	
Failed End Condition	Map is not displayed	
Primary Actor	User	
Trigger	User selects location to view on the map	
SUCCESS SCENARIO	**Step**	**Action**
	1	User opens the application
	2	User selects a location to view on the map
	3	Application displays map with selected location pointer and the users current location
	4	Map is interactive and user can zoom in and out.
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	1	User opens the application
	2	User selects a location to view on the map
	3	Map is unable to open and display locations.
	4	Application throws an error message.
RELATED INFORMATION		
Priority	Top Priority	
Performance Target	Map should be displayed within 10 seconds.	
Frequency	Frequent, every location will display map location.	
Subordinate Use Cases	-	
Channel to Primary Actor	User interface	
OPEN ISSUES	-	
SCHEDULE	Release build	
AUTHOR	Chanelle Richardson 22/10/2016	

Appendix B7: Location Finder Search for a Location

USE CASE NAME	Search for a location
Goal in Context	To allow a user to filter the list of locations using search terms

Scope & Level	Overall System	
Preconditions	Application must be running and location data must be loaded in	
Success End Condition	The user is presented with a list of locations relating to the search terms used	
Failed End Condition	The user is notified that an error has occurred or there are no results found.	
Primary Actor	User	
Trigger	The user wants to find a specific location	
SUCCESS SCENARIO	**Step**	**Action**
	1	User inputs textual search terms into the search bar widget and presses the “search” button
	2	The application generates a list of all locations/points of interest which contain one or more of the search terms.
	3	A list of the related locations are displayed to the user on a new screen. Each location has a brief description and a button to open a
	4.	The user can then return to the home screen via a back button, re-define their search using the search bar again or select one of the locations from the list
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	2a	The application was unable to find any locations relating to the search terms entered
	3a	The user is presented with a notification that “No results were found”. When this message is dismissed the user will remain on the home screen.
RELATED INFORMATION		
Priority	High priority	
Performance Target	The search should process a list of locations in < 2 seconds	
Frequency	This use case will be used very frequently	
OPEN ISSUES		
SCHEDULE	Release Build	
AUTHOR	Danny Murphy (CMP) – 21/10/2016	

Appendix B8: Location Finder Viewing a Point of Interest on the Map

USE CASE NAME	**Viewing A Point Of Interest on the map**	
Goal in Context	To allow the user to view a point of interest on the map	
Scope & Level	Overall System	
Preconditions	Application is running and the user has selected a point of interest from the list	
Success End Condition	A point of interest is displayed on the map	
Failed End Condition	The application does not display a point of interest	
Primary Actor	User	
Trigger	The user wants to view a point of interest on the map	
SUCCESS SCENARIO	**Step**	**Action**
	1	The application displays a description of the point of interest and a “View on Map” button to display the location on a map
	2	The user presses the “View on Map” button
	3	The application fetches the location data
	4	The application opens the campus map with a pin to mark the point of interest
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	3a	The application is unable to fetch location data. Display an error message and return the user to the location description.
RELATED INFORMATION		
Priority	High Priority	
Performance Target	Display the location on the map in < 2 seconds	
Frequency	High – Viewing Point of interest is the main function of the application	
OPEN ISSUES	- What happens if the location data cannot be retrieved?	
SCHEDULE	Release Build	
AUTHOR	Daniel Banks (CMP) – 21/10/2016	

Appendix B9: Parking Aid Set Notification Interval

USE CASE NAME	Parking Aid – Set Notification Interval	
Goal in Context	Display map	
Scope & Level	Overall System	
Preconditions	Application is running and user has selected the setting menu	
Success End Condition	User is able to change the notification interval times.	
Failed End Condition	User is unable to change notification interval times. Default times are used.	
Primary Actor	User	
Trigger	Change notification interval time is selected in the settings menu.	
SUCCESS SCENARIO	**Step**	**Action**
	1	User opens the application and selects the Settings menu
	2	User selects Change Notification Interval and sets a time
	3	System saves the changes
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	1	User opens the application and selects the Settings menu
	2	User selects Change Notification Interval and tries to set a time
	3	System is unable to save the set time and throws an error message.
RELATED INFORMATION		
Priority	High	
Performance Target	Shouldn't take longer than 10 seconds.	
Frequency	Frequent, User may want to change the time every time they use the application.	
Subordinate Use Cases	-	
Channel to Primary Actor	User interface	
OPEN ISSUES	-	
SCHEDULE	Release build	
AUTHOR	Chanelle Richardson 22/10/2016	

Appendix B10: Parking Aid Add Car Park Arrival

USE CASE NAME	**Add car park arrival**	
Goal in Context	**Actor to complete form to process car in the car park**	
Scope & Level	Overall system	
Preconditions	Application is running and is on the "**Home**" screen.	
Success End Condition	Actor enters car park arrival information and system provides a default reminder 10 minutes prior to fee increase.	
Failed End Condition	System does not have car park arrival information.	
Primary Actor	User	
Trigger	Actor enters data on the **"New Parking"** form.	
SUCCESS SCENARIO	**Step**	**Action**
	1	System displays **"New Parking"** form.
	2	System pre-defines **"Name of Car Park"**, **"Time of Arrival"** and **"Fee"** fields.
	3	User enters zone.
	4	User selects **"Add"**.
	5	System processes and generates GPS tag and stores the time.
	6	System displays **"Cost and Time"** screen which includes displaying a pop-up message to indicate the allocated remaining time and the parking fee.
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	2a	User adapts pre-defined information and returns to step 3.
	3a	User does not enter zone information and returns to step 4.
	4a	User adds incorrect information.
	5a	User selects the "<" button and returns to step 1.
	5b	If user does not have GPS switched on in their phone, the system will prompt the user to turn on their GPS tracker.
RELATED INFORMATION		
Priority	Top priority	
Performance Target	System should be quick to process the click of the button and then be quick to respond back to the user	
Frequency	Moderate – occurs when user chooses to complete/update their information of their car	
Subordinate Use Cases	Navigate to Car	
Channel to Primary Actor	User Interface	
Secondary Actors	-	
Channel to Secondary Actors	-	
OPEN ISSUES	- What if the user information entered is invalid?	
SCHEDULE	Due date is version 1.0 release	

AUTHOR	Sophie Whiteside 21/10/2016

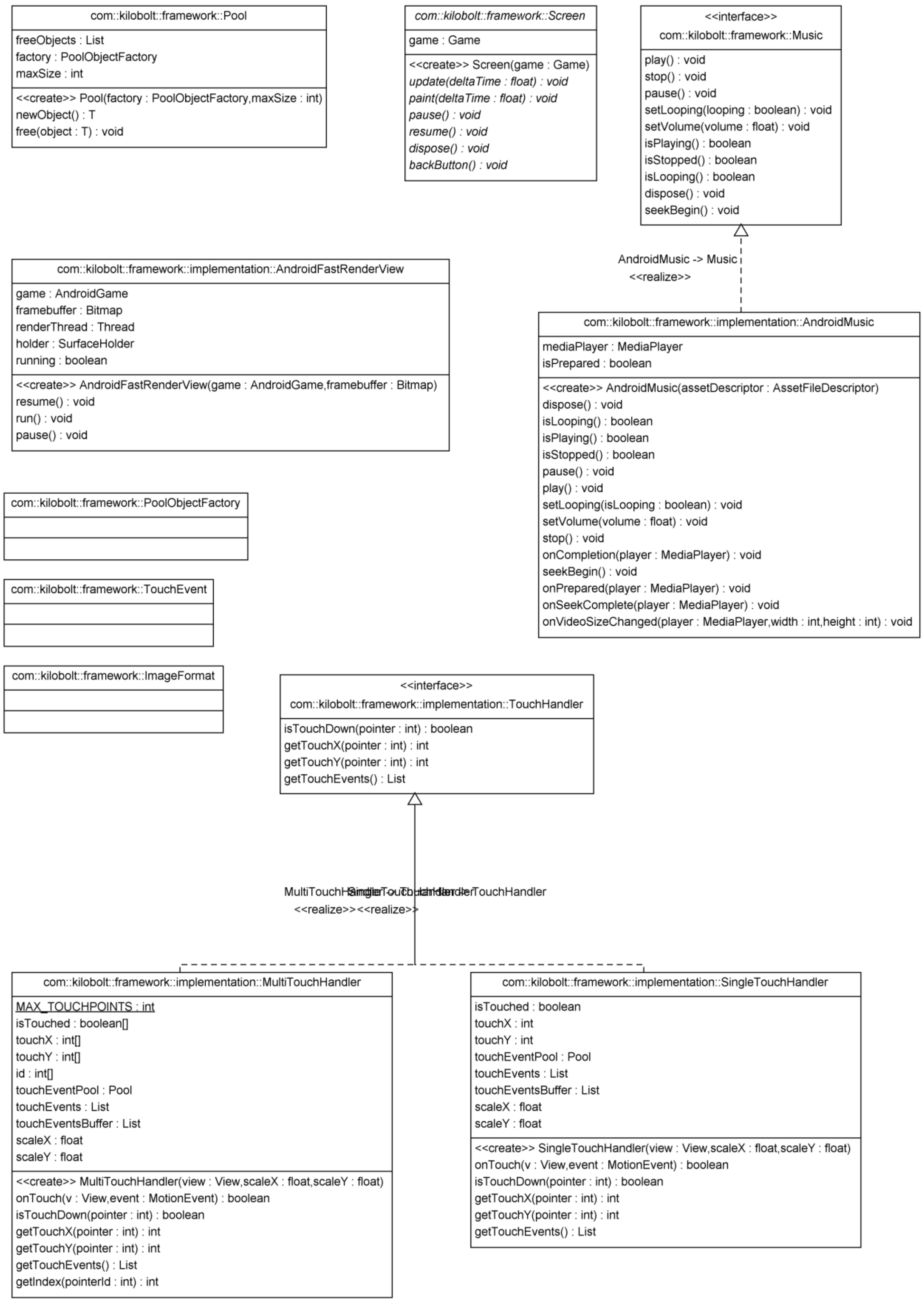
Appendix B11: Parking Aid Navigate to Car

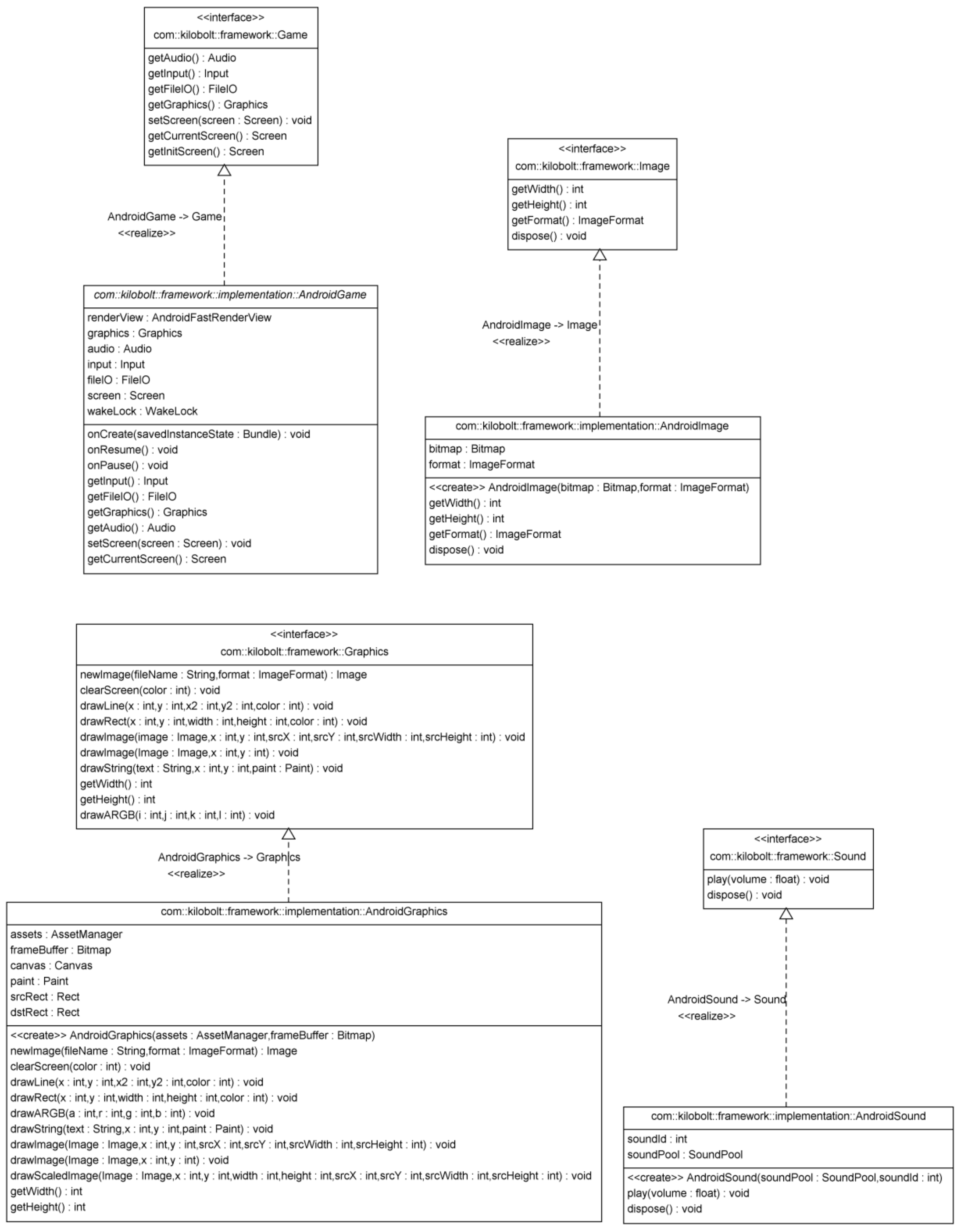
USE CASE NAME	**Navigate to Car**	
Goal in Context	Actor to select the Navigation button in order to then travel back to the location of their car.	
Scope & Level	Level 1	
Preconditions	Application is running and is on the "**Home**" screen. Also, **"Add car park arrival"** use case has been fulfilled.	
Success End Condition	Actor is directed to car.	
Failed End Condition	Actor is not directed to car correctly.	
Primary Actor	User	
Trigger	Actor selects the **"Navigate to car"** button.	
SUCCESS SCENARIO	**Step**	**Action**
	1	System locates current GPS Tag.
	2	System locates the destination.
	3	System presents a compass pointing the User in the direction of the location.
	4	User moves towards destination following the arrow.
	5	System continually updates as User moves towards the destination.
	6	User arrives at destination.
	7	The system notifies actor that they have arrived at their destination.
	8	System returns to the **"Home"** screen.
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	1a	System cannot locate current GPS Tag.
	2	System displays a message **"Cannot locate current position"** and is prompted to refresh the screen.
	2a	System cannot locate destination.
	3	System displays a message **"Cannot locate destination"** and returns to step 4.
	6a	User selects the **"<"** button and returns to step 3.
RELATED INFORMATION		
Priority	Top priority	
Performance Target	System should be quick to process the click of the button and then be quick to respond back to the user	
Frequency	Moderate – occurs when user chooses to use the GPS to locate their car	
Subordinate Use Cases	-	
Channel to Primary Actor	User Interface	

Secondary Actors	-
Channel to Secondary Actors	-
OPEN ISSUES	- What happens if the GPS Tag cannot be located?
SCHEDULE	Due date is version 1.0 release
AUTHOR	Sophie Whiteside 21/10/2016

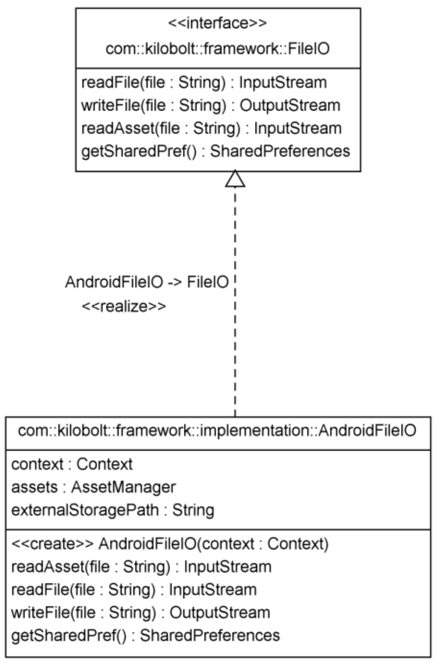
Appendix C Class Diagrams

Appendix C1: Original SAAF Class Diagram

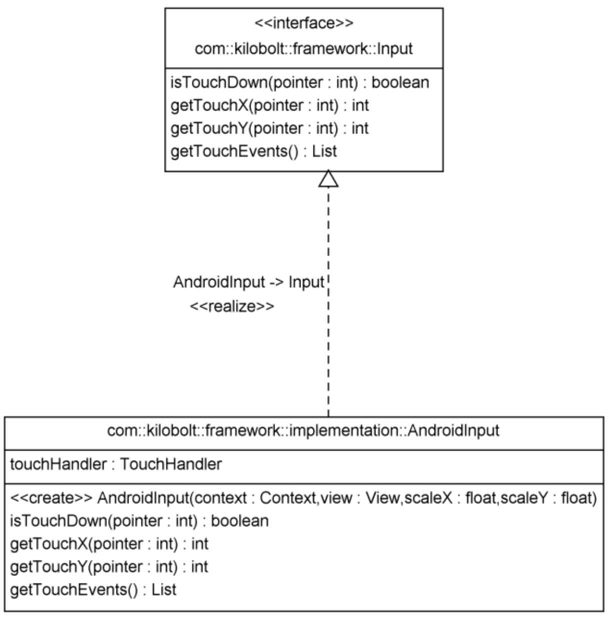

com::kilobolt::framework::Pool
freeObjects : List
factory : PoolObjectFactory
maxSize : int
<<create>> Pool(factory : PoolObjectFactory,maxSize : int)
newObject() : T
free(object : T) : void
com::kilobolt::framework::Screen
game : Game
<<create>> Screen(game : Game)
update(deltaTime : float) : void
paint(deltaTime : float) : void
pause() : void
resume() : void
dispose() : void
backButton() : void
<<interface>>
com::kilobolt::framework::Music
play() : void
stop() : void
pause() : void
setLooping(looping : boolean) : void
setVolume(volume : float) : void
isPlaying() : boolean
isStopped() : boolean
isLooping() : boolean
dispose() : void
seekBegin() : void
AndroidMusic -> Music
<<realize>>
com::kilobolt::framework::implementation::AndroidFastRenderView
game : AndroidGame
framebuffer : Bitmap
renderThread : Thread
holder : SurfaceHolder
running : boolean
<<create>> AndroidFastRenderView(game : AndroidGame,framebuffer : Bitmap)
resume() : void
run() : void
pause() : void
com::kilobolt::framework::implementation::AndroidMusic
mediaPlayer : MediaPlayer
isPrepared : boolean
<<create>> AndroidMusic(assetDescriptor : AssetFileDescriptor)
dispose() : void
isLooping() : boolean
isPlaying() : boolean
isStopped() : boolean
pause() : void
play() : void
setLooping(isLooping : boolean) : void
setVolume(volume : float) : void
stop() : void
onCompletion(player : MediaPlayer) : void
seekBegin() : void
onPrepared(player : MediaPlayer) : void
onSeekComplete(player : MediaPlayer) : void
onVideoSizeChanged(player : MediaPlayer,width : int,height : int) : void
com::kilobolt::framework::PoolObjectFactory
com::kilobolt::framework::TouchEvent
com::kilobolt::framework::ImageFormat
<<interface>>
com::kilobolt::framework::implementation::TouchHandler
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List
MultiTouchHandler -> TouchHandler
SingleTouchHandler -> TouchHandler
<<realize>>
<<realize>>
com::kilobolt::framework::implementation::MultiTouchHandler
MAX_TOUCHPOINTS : int
isTouched : boolean[]
touchX : int[]
touchY : int[]
id : int[]
touchEventPool : Pool
touchEvents : List
touchEventsBuffer : List
scaleX : float
scaleY : float
<<create>> MultiTouchHandler(view : View,scaleX : float,scaleY : float)
onTouch(v : View,event : MotionEvent) : boolean
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List
getIndex(pointerId : int) : int
com::kilobolt::framework::implementation::SingleTouchHandler
isTouched : boolean
touchX : int
touchY : int
touchEventPool : Pool
touchEvents : List
touchEventsBuffer : List
scaleX : float
scaleY : float
<<create>> SingleTouchHandler(view : View,scaleX : float,scaleY : float)
onTouch(v : View,event : MotionEvent) : boolean
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List



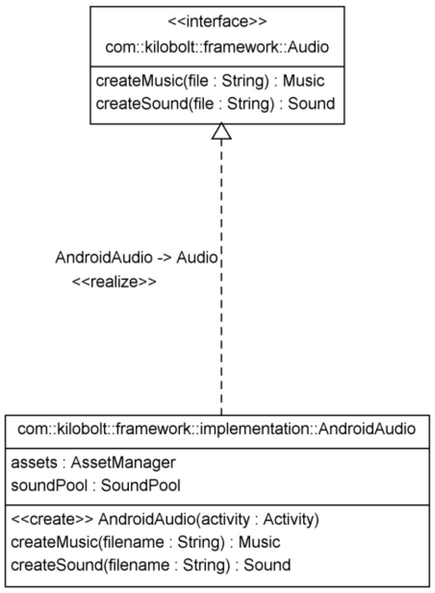
<<interface>>
com::kilobolt::framework::Game
getAudio() : Audio
getInput() : Input
getFileIO() : FileIO
getGraphics() : Graphics
setScreen(screen : Screen) : void
getCurrentScreen() : Screen
getInitScreen() : Screen
AndroidGame -> Game
<<realize>>
com::kilobolt::framework::implementation::AndroidGame
renderView : AndroidFastRenderView
graphics : Graphics
audio : Audio
input : Input
fileIO : FileIO
screen : Screen
wakeLock : WakeLock
onCreate(savedInstanceState : Bundle) : void
onResume() : void
onPause() : void
getInput() : Input
getFileIO() : FileIO
getGraphics() : Graphics
getAudio() : Audio
setScreen(screen : Screen) : void
getCurrentScreen() : Screen
<<interface>>
com::kilobolt::framework::Image
getWidth() : int
getHeight() : int
getFormat() : ImageFormat
dispose() : void
AndroidImage -> Image
<<realize>>
com::kilobolt::framework::implementation::AndroidImage
bitmap : Bitmap
format : ImageFormat
<<create>> AndroidImage(bitmap : Bitmap,format : ImageFormat)
getWidth() : int
getHeight() : int
getFormat() : ImageFormat
dispose() : void
<<interface>>
com::kilobolt::framework::Graphics
newImage(fileName : String,format : ImageFormat) : Image
clearScreen(color : int) : void
drawLine(x : int,y : int,x2 : int,y2 : int,color : int) : void
drawRect(x : int,y : int,width : int,height : int,color : int) : void
drawImage(image : Image,x : int,y : int,srcX : int,srcY : int,srcWidth : int,srcHeight : int) : void
drawImage(Image : Image,x : int,y : int) : void
drawString(text : String,x : int,y : int,paint : Paint) : void
getWidth() : int
getHeight() : int
drawARGB(i : int,j : int,k : int,l : int) : void
AndroidGraphics -> Graphics
<<realize>>
com::kilobolt::framework::implementation::AndroidGraphics
assets : AssetManager
frameBuffer : Bitmap
canvas : Canvas
paint : Paint
srcRect : Rect
dstRect : Rect
<<create>> AndroidGraphics(assets : AssetManager,frameBuffer : Bitmap)
newImage(fileName : String,format : ImageFormat) : Image
clearScreen(color : int) : void
drawLine(x : int,y : int,x2 : int,y2 : int,color : int) : void
drawRect(x : int,y : int,width : int,height : int,color : int) : void
drawARGB(a : int,r : int,g : int,b : int) : void
drawString(text : String,x : int,y : int,paint : Paint) : void
drawImage(Image : Image,x : int,y : int,srcX : int,srcY : int,srcWidth : int,srcHeight : int) : void
drawImage(Image : Image,x : int,y : int) : void
drawScaledImage(Image : Image,x : int,y : int,width : int,height : int,srcX : int,srcY : int,srcWidth : int,srcHeight : int) : void
getWidth() : int
getHeight() : int
<<interface>>
com::kilobolt::framework::Sound
play(volume : float) : void
dispose() : void
AndroidSound -> Sound
<<realize>>
com::kilobolt::framework::implementation::AndroidSound
soundId : int
soundPool : SoundPool
<<create>> AndroidSound(soundPool : SoundPool,soundId : int)
play(volume : float) : void
dispose() : void



<<interface>>
com::kilobolt::framework::FileIO
readFile(file : String) : InputStream
writeFile(file : String) : OutputStream
readAsset(file : String) : InputStream
getSharedPref() : SharedPreferences
AndroidFileIO -> FileIO
<<realize>>
com::kilobolt::framework::implementation::AndroidFileIO
context : Context
assets : AssetManager
externalStoragePath : String
<<create>> AndroidFileIO(context : Context)
readAsset(file : String) : InputStream
readFile(file : String) : InputStream
writeFile(file : String) : OutputStream
getSharedPref() : SharedPreferences

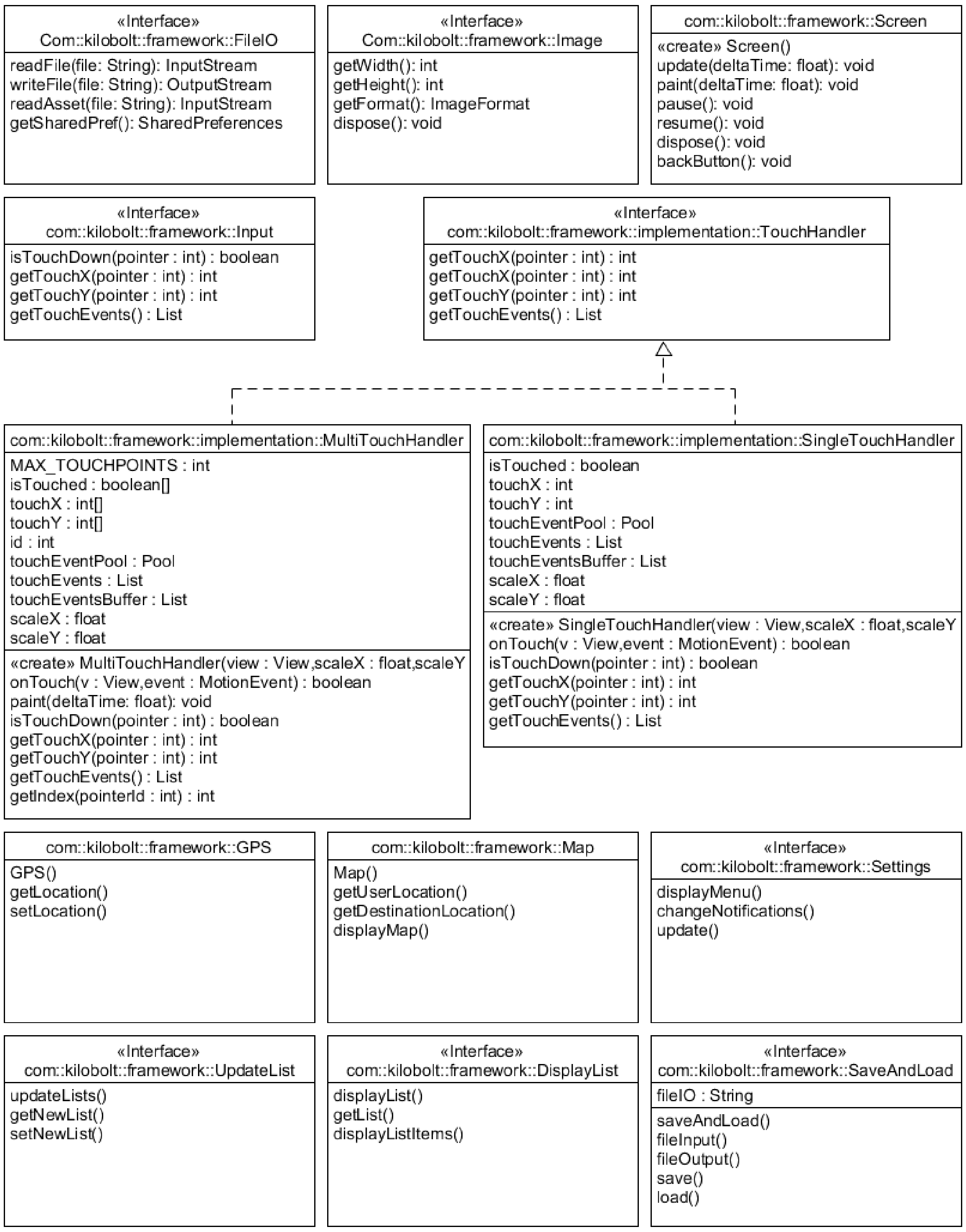


<<interface>>
com::kilobolt::framework::Input
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List
AndroidInput -> Input
<<realize>>
com::kilobolt::framework::implementation::AndroidInput
touchHandler : TouchHandler
<<create>> AndroidInput(context : Context,view : View,scaleX : float,scaleY : float)
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List

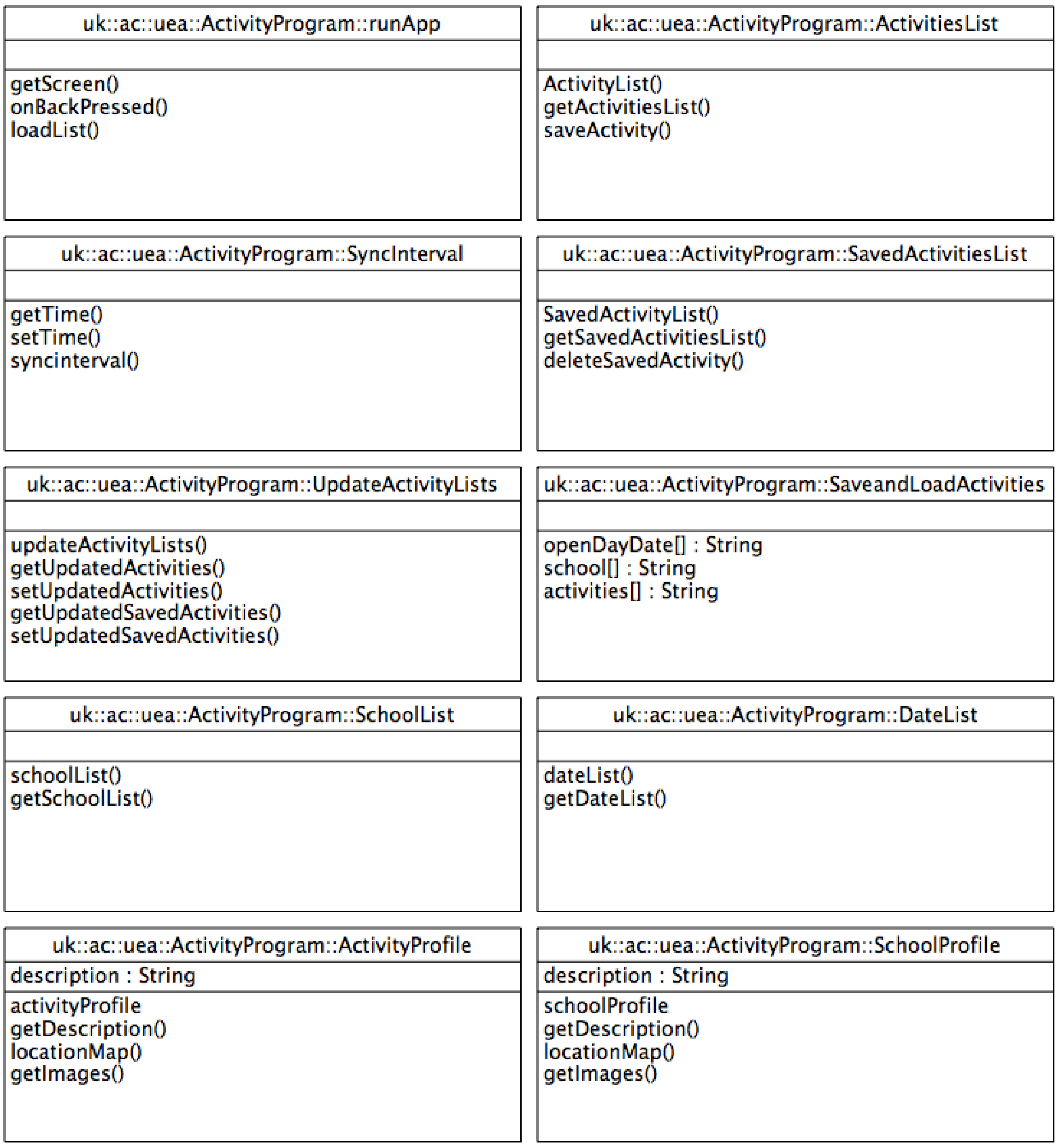


<<interface>>
com::kilobolt::framework::Audio
createMusic(file : String) : Music
createSound(file : String) : Sound
AndroidAudio -> Audio
<<realize>>
com::kilobolt::framework::implementation::AndroidAudio
assets : AssetManager
soundPool : SoundPool
<<create>> AndroidAudio(activity : Activity)
createMusic(filename : String) : Music
createSound(filename : String) : Sound

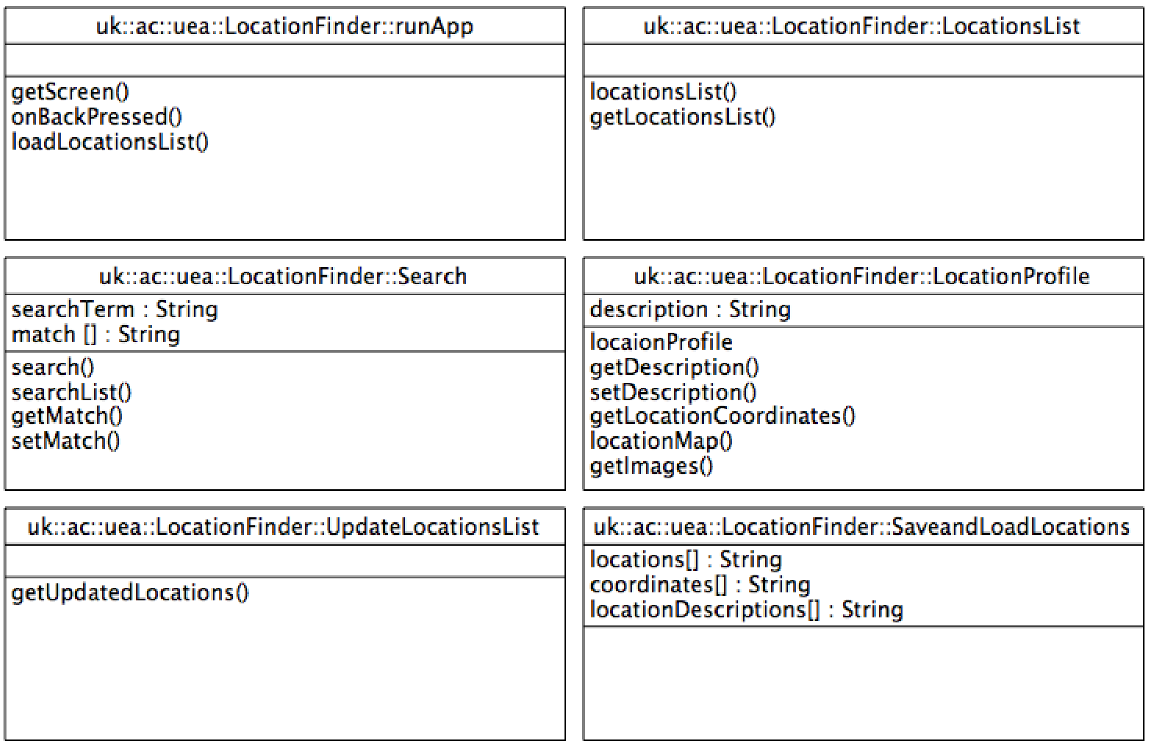
Appendix C2: Revised SAAF Class Diagram


«Interface»
Com::kilobolt::framework::FileIO
readFile(file: String): InputStream
writeFile(file: String): OutputStream
readAsset(file: String): InputStream
getSharedPref(): SharedPreferences
«Interface»
Com::kilobolt::framework::Image
getWidth(): int
getHeight(): int
getFormat(): ImageFormat
dispose(): void
com::kilobolt::framework::Screen
«create» Screen()
update(deltaTime: float): void
paint(deltaTime: float): void
pause(): void
resume(): void
dispose(): void
backButton(): void
«Interface»
com::kilobolt::framework::Input
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List
«Interface»
com::kilobolt::framework::implementation::TouchHandler
getTouchX(pointer : int) : int
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List
com::kilobolt::framework::implementation::MultiTouchHandler
MAX_TOUCHPOINTS : int
isTouched : boolean[]
touchX : int[]
touchY : int[]
id : int
touchEventPool : Pool
touchEvents : List
touchEventsBuffer : List
scaleX : float
scaleY : float
«create» MultiTouchHandler(view : View,scaleX : float,scaleY
onTouch(v : View,event : MotionEvent) : boolean
paint(deltaTime: float): void
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List
getIndex(pointerId : int) : int
com::kilobolt::framework::implementation::SingleTouchHandler
isTouched : boolean
touchX : int
touchY : int
touchEventPool : Pool
touchEvents : List
touchEventsBuffer : List
scaleX : float
scaleY : float
«create» SingleTouchHandler(view : View,scaleX : float,scaleY
onTouch(v : View,event : MotionEvent) : boolean
isTouchDown(pointer : int) : boolean
getTouchX(pointer : int) : int
getTouchY(pointer : int) : int
getTouchEvents() : List
com::kilobolt::framework::GPS
GPS()
getLocation()
setLocation()
com::kilobolt::framework::Map
Map()
getUserLocation()
getDestinationLocation()
displayMap()
«Interface»
com::kilobolt::framework::Settings
displayMenu()
changeNotifications()
update()
«Interface»
com::kilobolt::framework::UpdateList
updateLists()
getNewList()
setNewList()
«Interface»
com::kilobolt::framework::DisplayList
displayList()
getList()
displayListItems()
«Interface»
com::kilobolt::framework::SaveAndLoad
fileIO : String
saveAndLoad()
fileInput()
fileOutput()
save()
load()

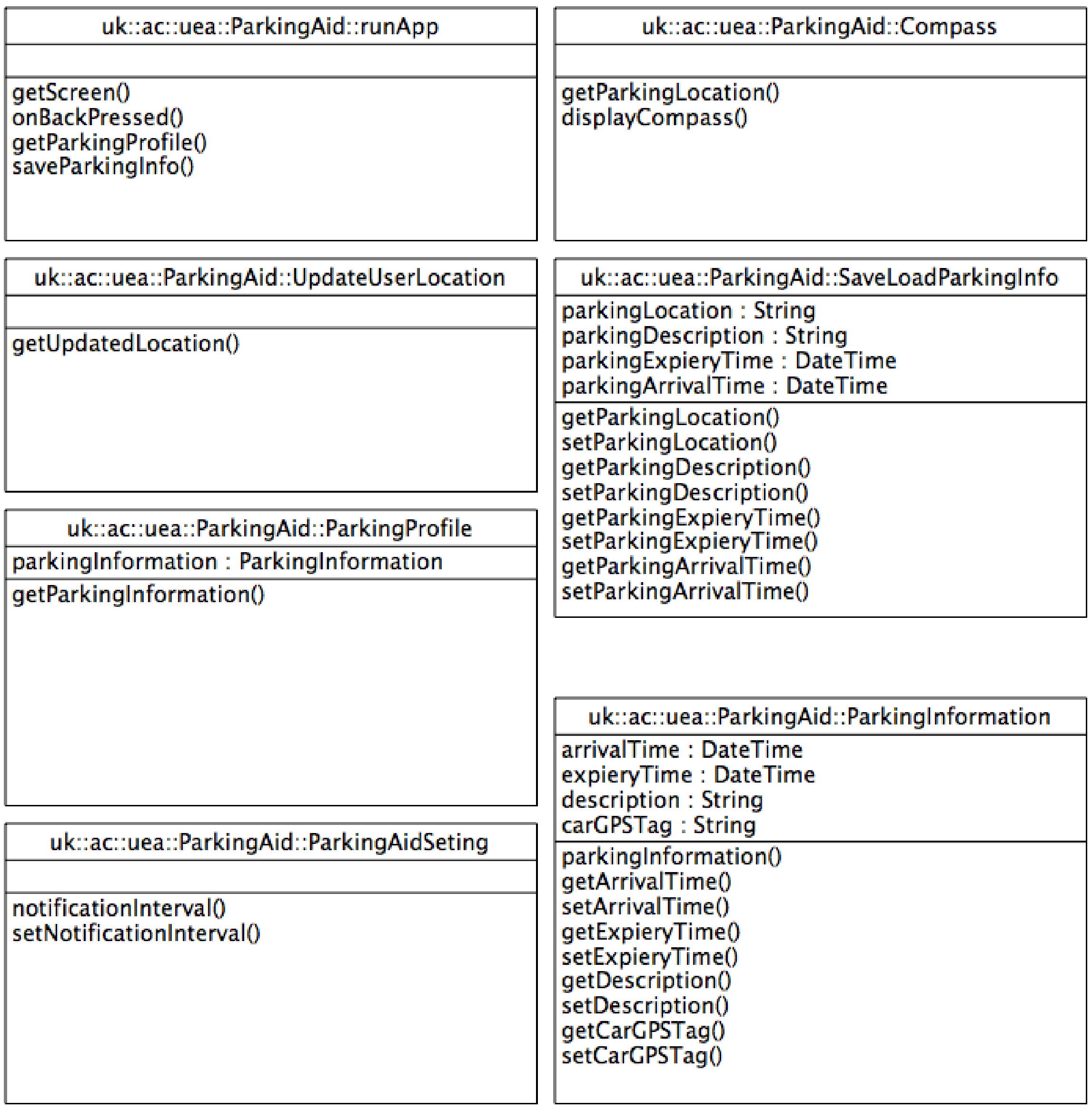
Appendix C3: Activity Program Class Diagram



Appendix C4: Location Finder Class Diagram

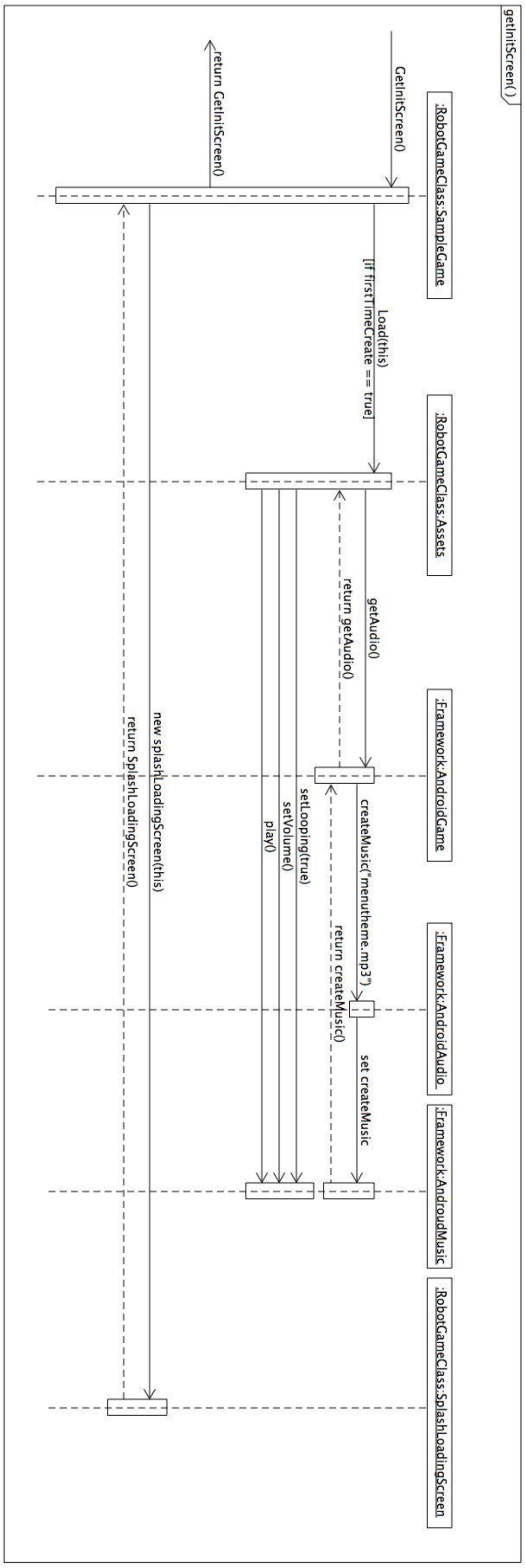


Appendix C5: Parking Aid Class Diagram

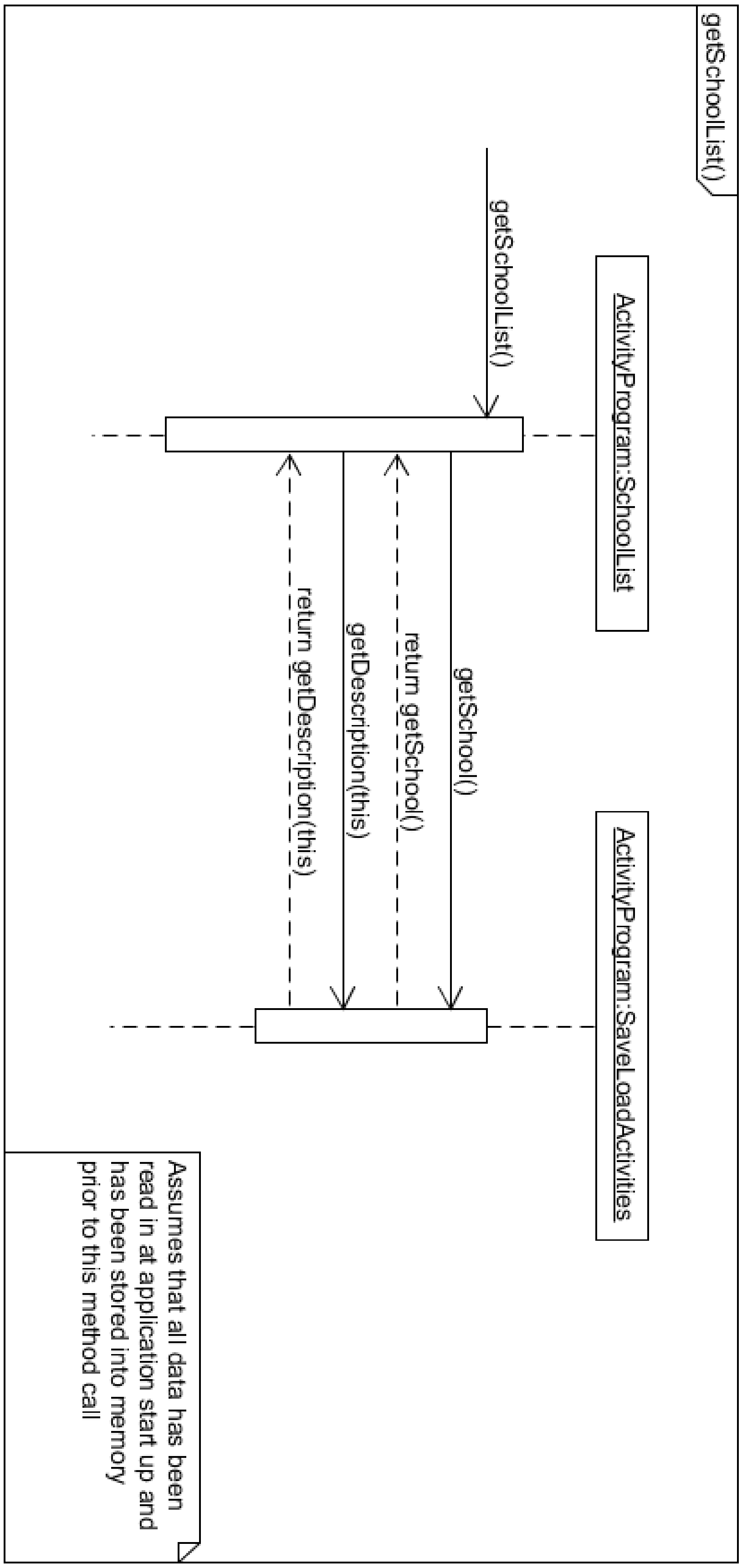


Appendix D Sequence Diagrams

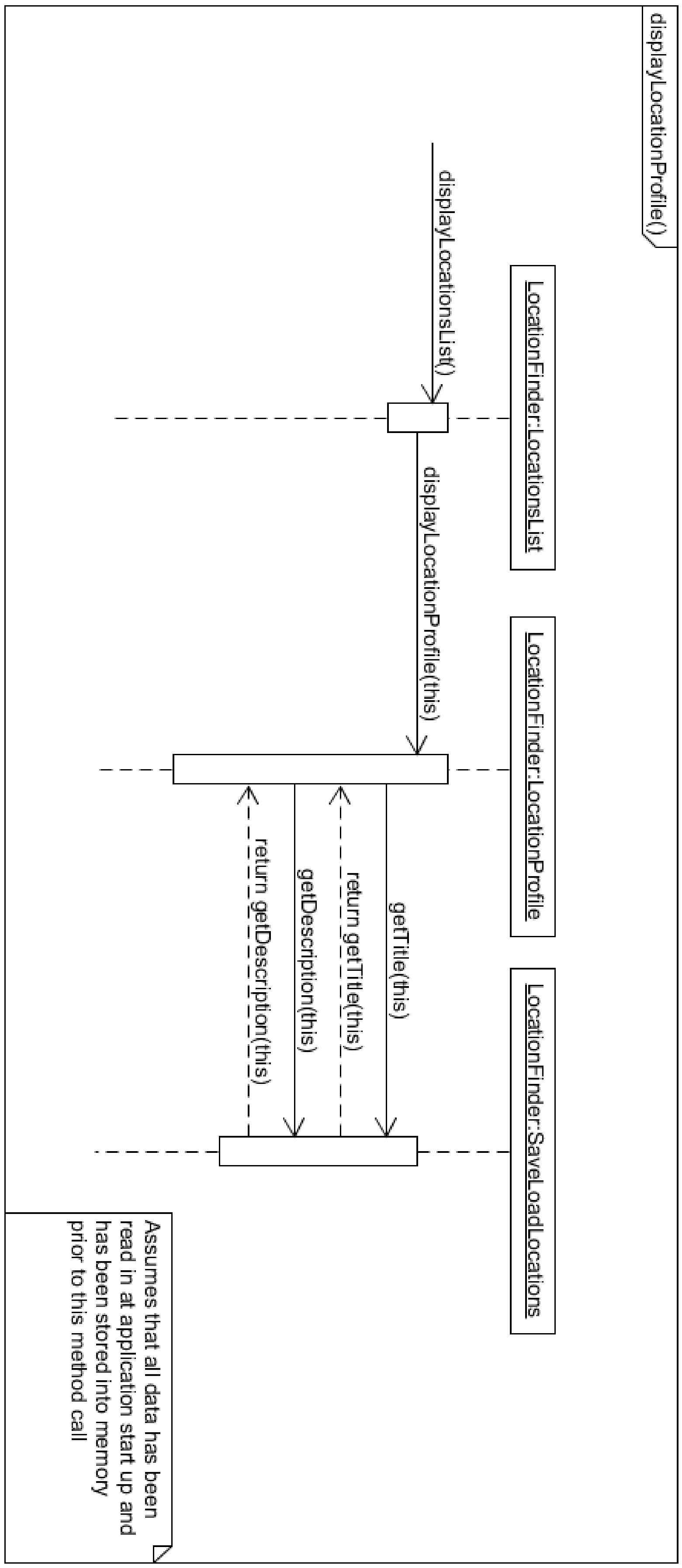
Appendix D1: Robot Game Sequence Diagram



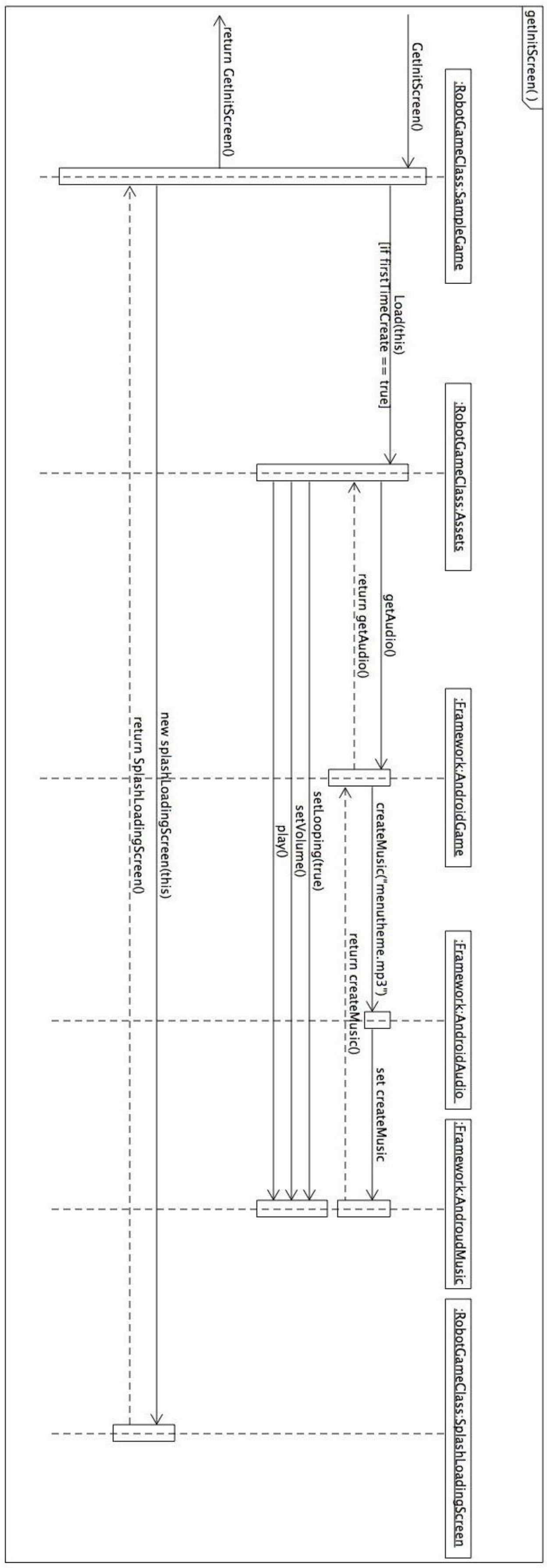
Appendix D2: Activity Program Sequence Diagram


getSchoolList()
ActivityProgram:SchoolList
ActivityProgram:SaveLoadActivities
getSchoolList()
getSchool()
return getSchool()
getDescription(this)
return getDescription(this)
Assumes that all data has been read in at application start up and has been stored into memory prior to this method call

Appendix D3: Location Finder Sequence Diagram

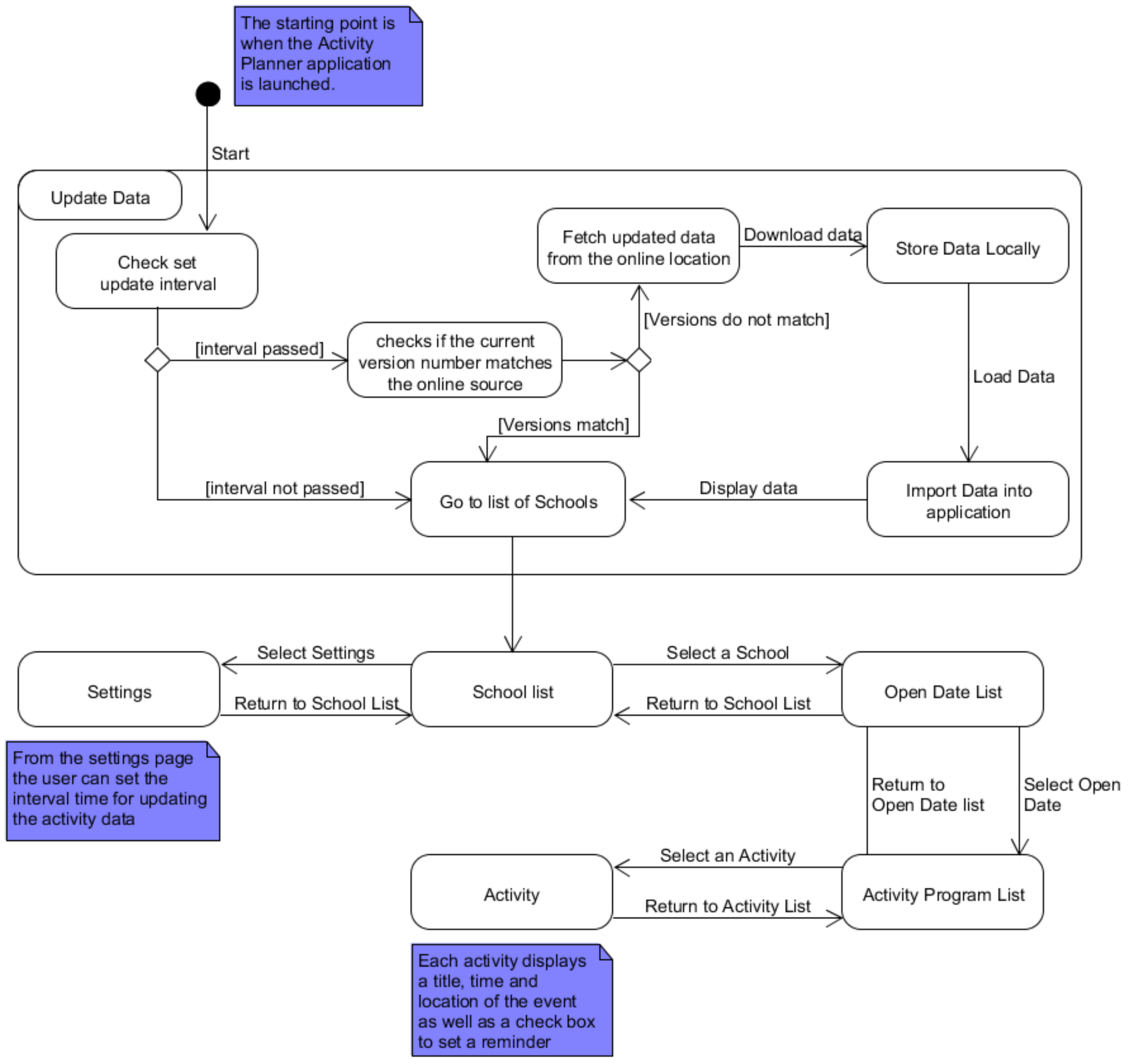

displayLocationProfile()
LocationFinder:LocationsList
LocationFinder:LocationProfile
LocationFinder:SaveLoadLocations
displayLocationsList()
displayLocationProfile(this)
getTitle(this)
return getTitle(this)
getDescription(this)
return getDescription(this)
Assumes that all data has been read in at application start up and has been stored into memory prior to this method call

Appendix D4: Parking Aid Sequence Diagram

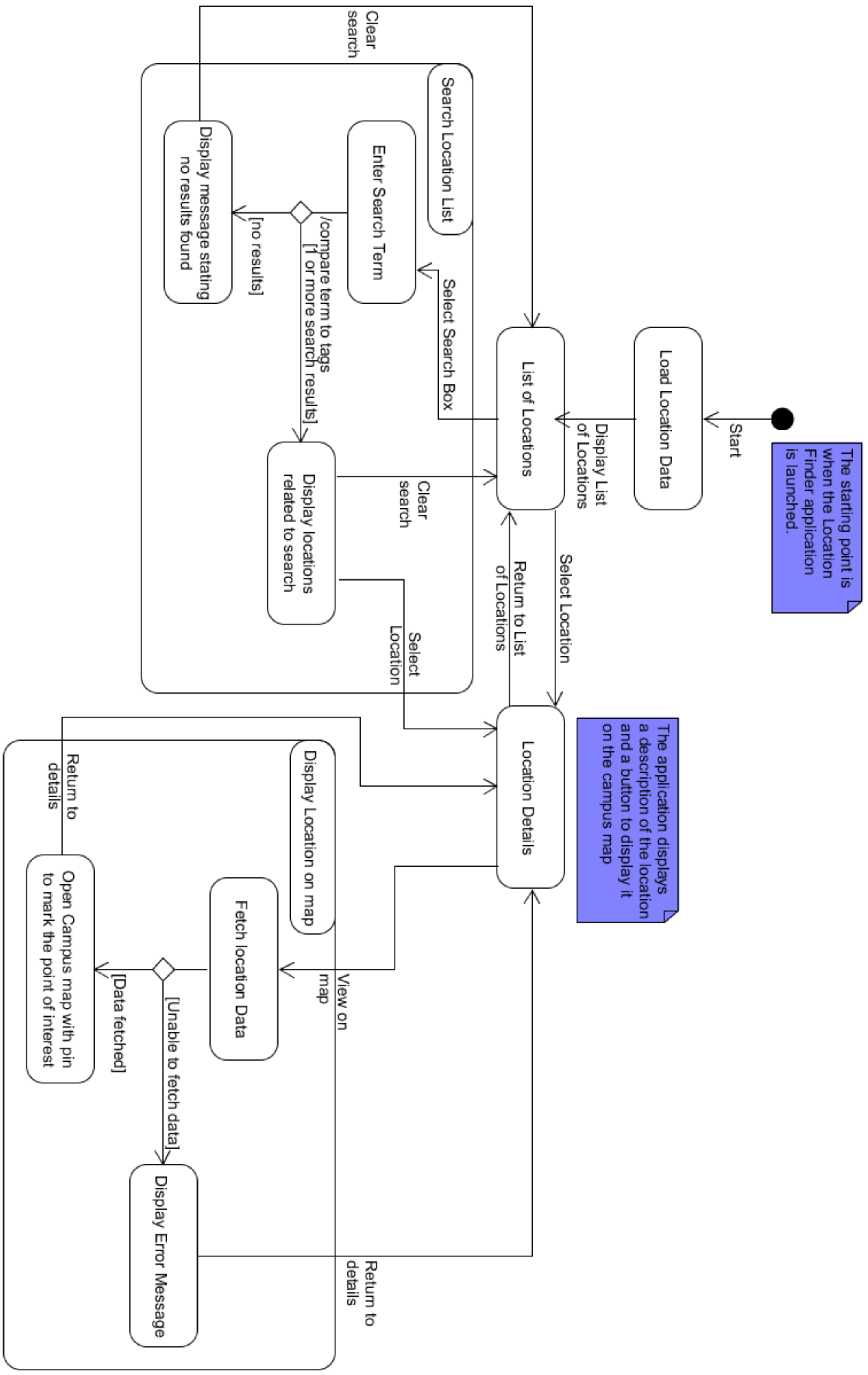


Appendix E State Transition Diagrams

Appendix E1: Activity Program State Transition Diagram

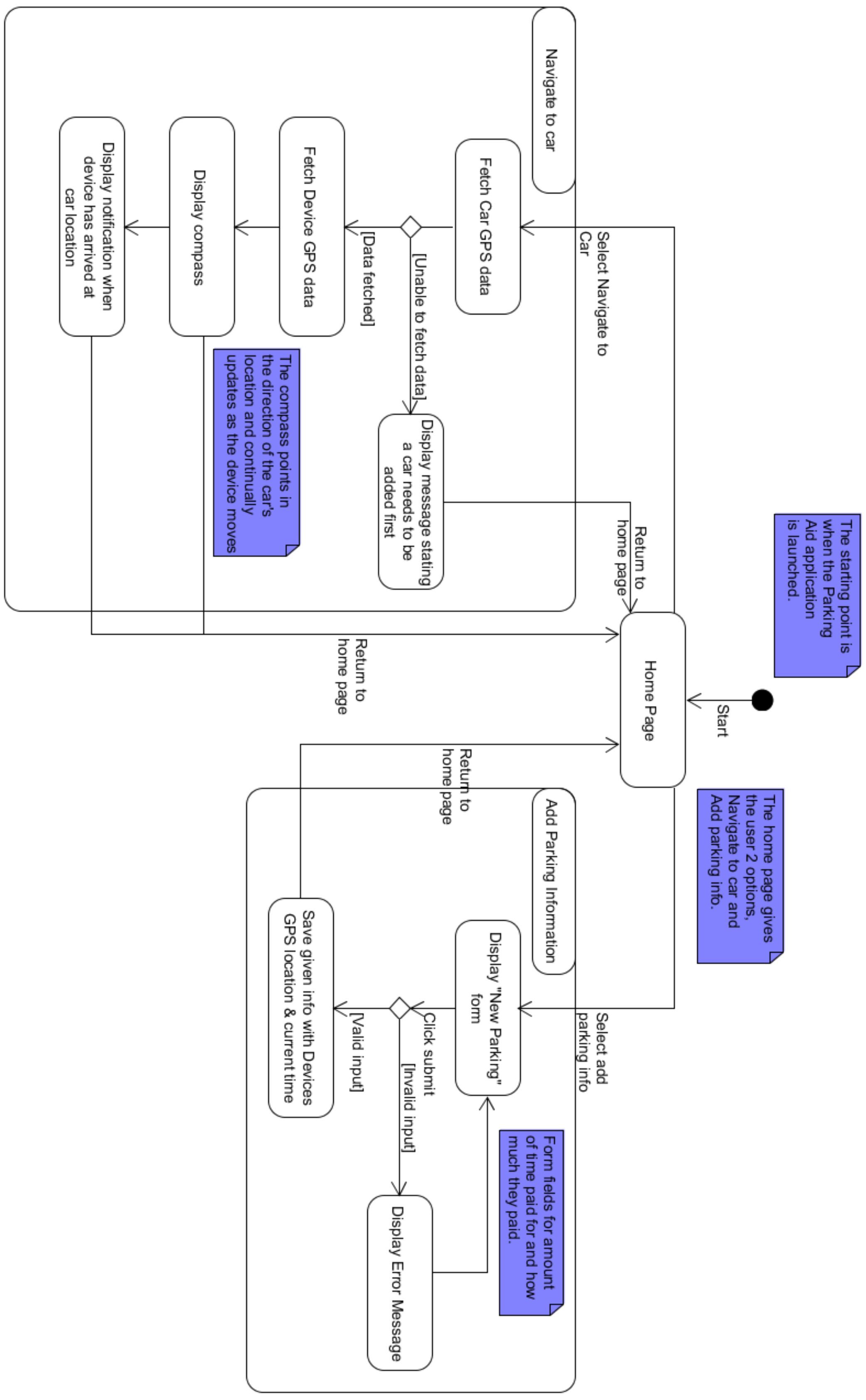

The starting point is when the Activity Planner application is launched.
Start
Update Data
Check set update interval
[interval passed]
checks if the current version number matches the online source
[Versions do not match]
Fetch updated data from the online location
Download data
Store Data Locally
Load Data
Import Data into application
Display data
[Versions match]
[interval not passed]
Go to list of Schools
Select Settings
Return to School List
Settings
School list
Select a School
Return to School List
Open Date List
From the settings page the user can set the interval time for updating the activity data
Return to Open Date list
Select Open Date
Select an Activity
Return to Activity List
Activity
Activity Program List
Each activity displays a title, time and location of the event as well as a check box to set a reminder

Appendix E2: Location Finder State Transition Diagram



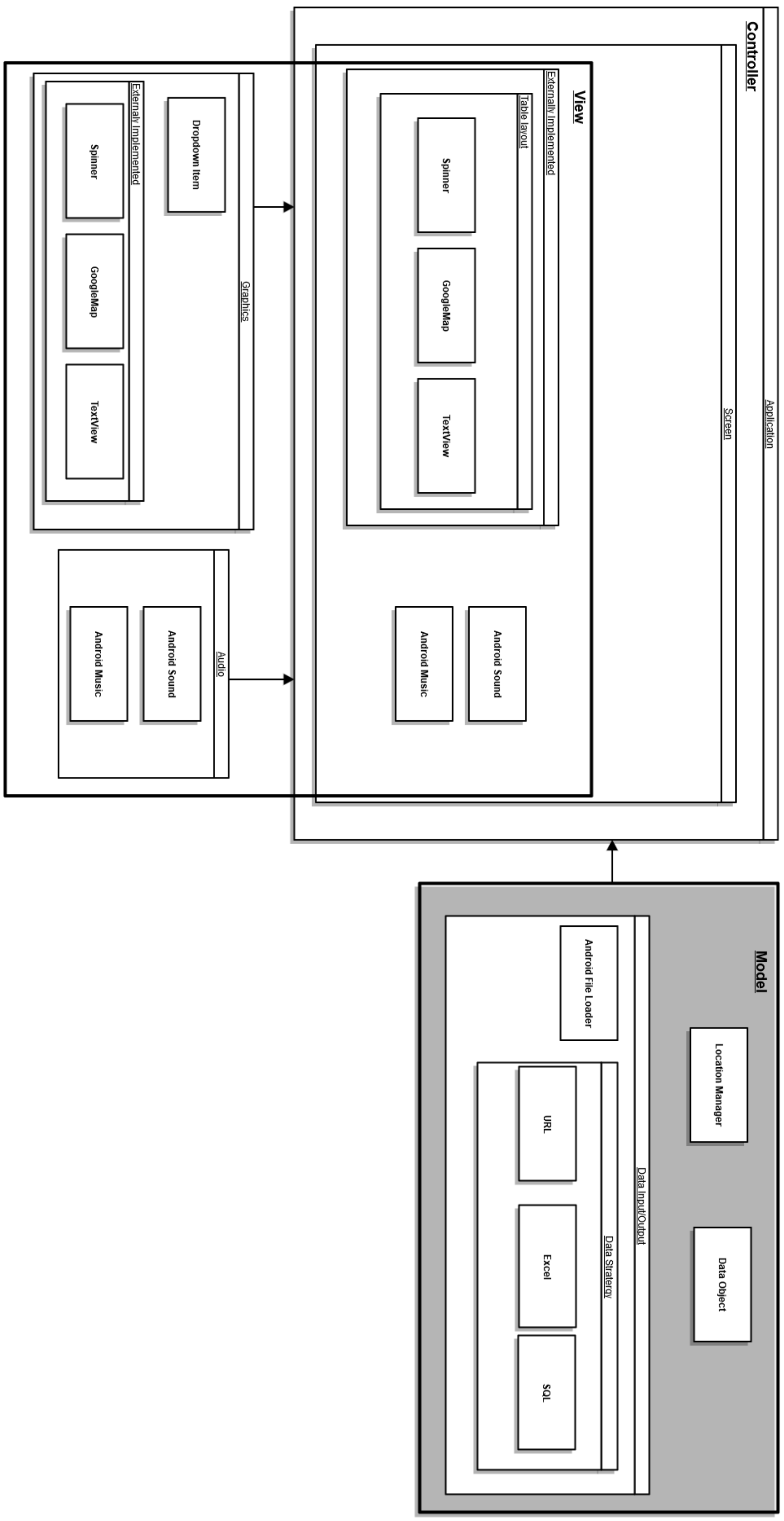
Start
The starting point is when the Location Finder application is launched.
Load Location Data
Display List of Locations
List of Locations
Select Search Box
Search Location List
Enter Search Term
/compare term to tags
[1 or more search results]
[no results]
Display message stating no results found
Clear search
Display locations related to search
Clear search
Select Location
Select Location
Return to List of Locations
Location Details
The application displays a description of the location and a button to display it on the campus map
Display Location on map
View on map
Fetch location Data
[Data fetched]
[Unable to fetch data]
Open Campus map with pin to mark the point of interest
Return to details
Display Error Message
Return to details

Appendix E3: Parking Aid State Transition Diagram

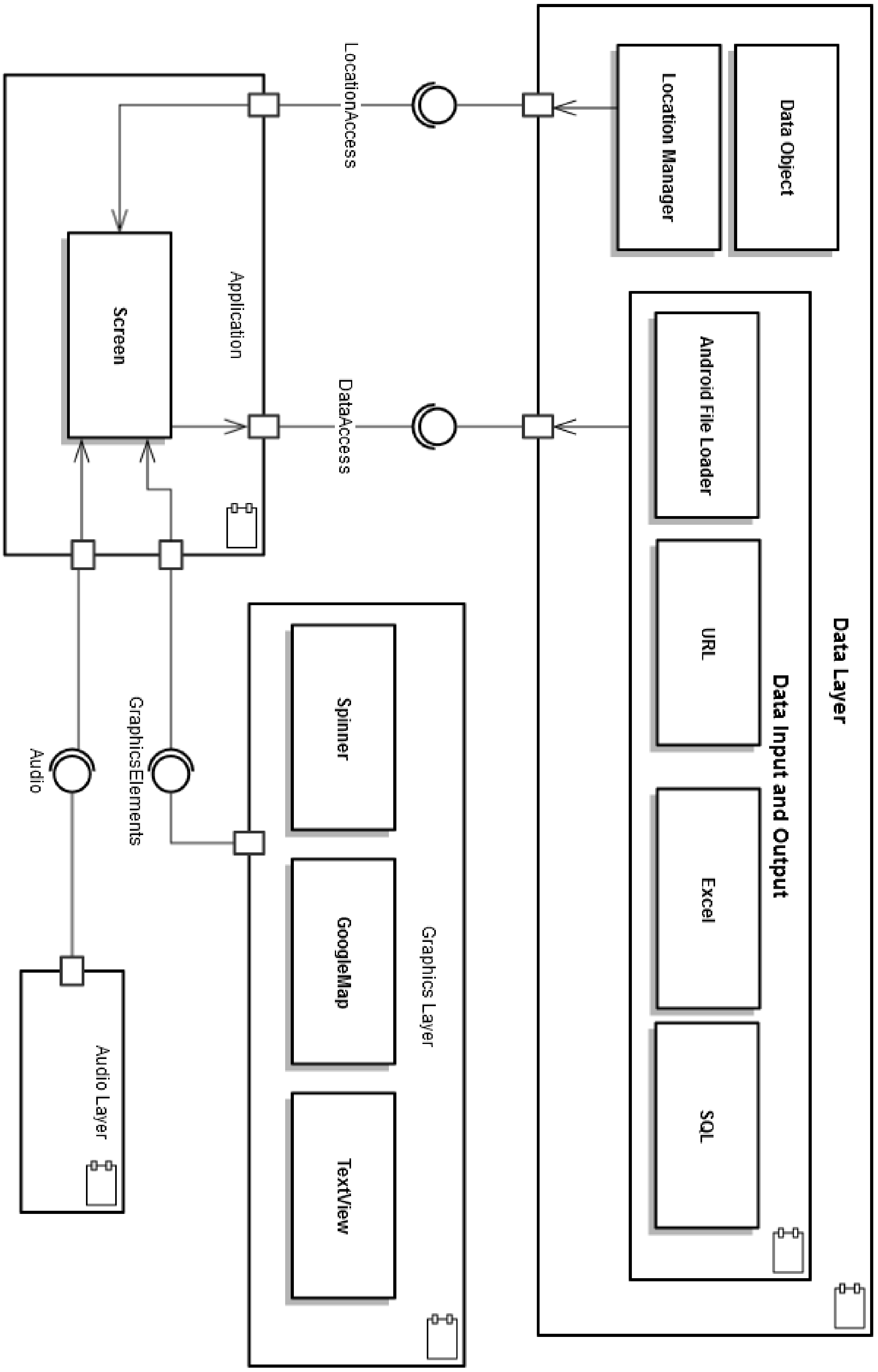


Start
The starting point is when the Parking Aid application is launched.
Home Page
The home page gives the user 2 options, Navigate to car and Add parking info.
Select Navigate to Car
Navigate to car
Fetch Car GPS data
[Unable to fetch data]
[Data fetched]
Display message stating a car needs to be added first
Return to home page
Fetch Device GPS data
Display compass
The compass points in the direction of the car's location and continually updates as the device moves
Display notification when device has arrived at car location
Return to home page
Select add parking info
Add Parking Information
Display "New Parking" form
Form fields for amount of time paid for and how much they paid.
Click submit
[Invalid input]
[Valid input]
Display Error Message
Save given info with Devices GPS location & current time
Return to home page

Appendix F Application Architecture Diagram

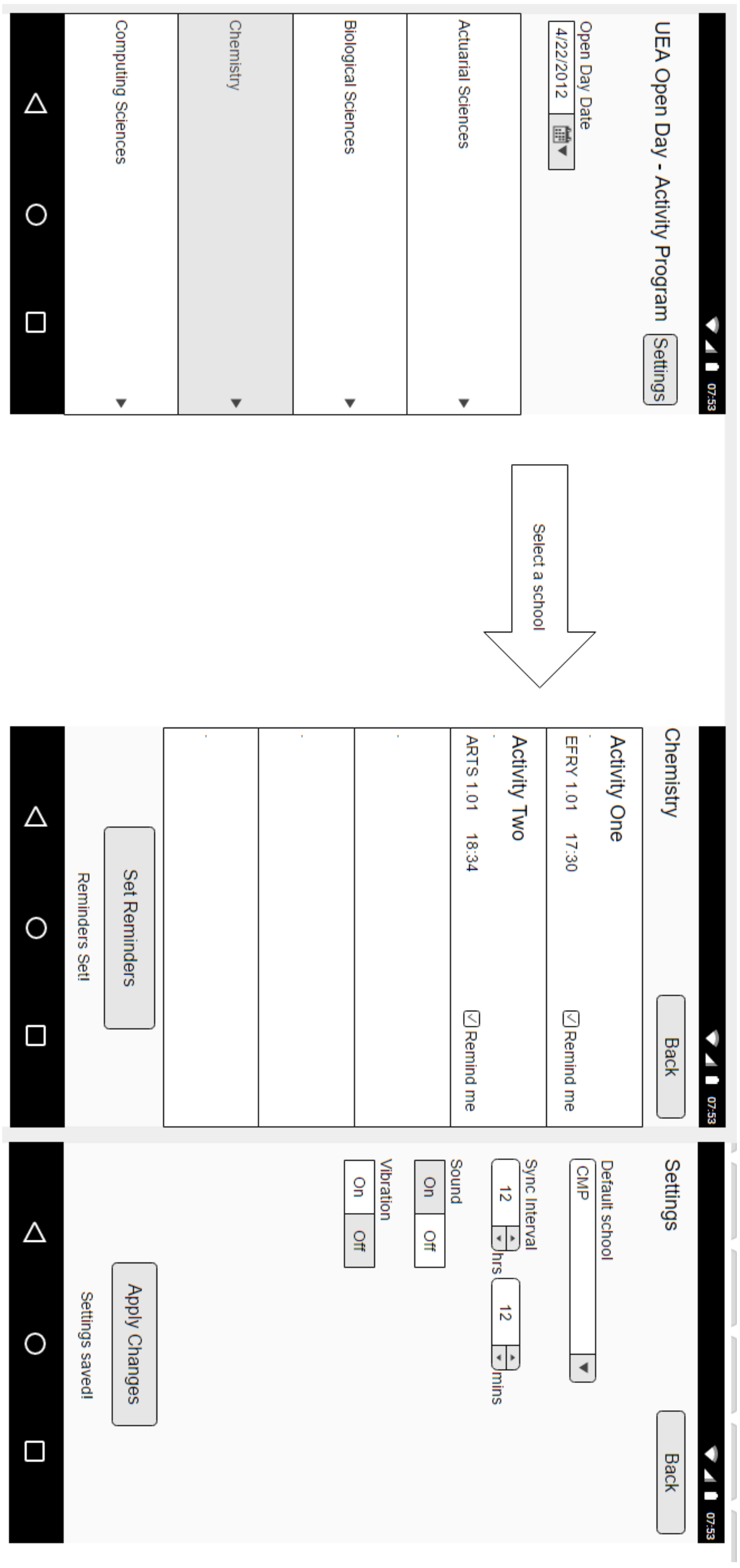


Appendix G Component Diagram

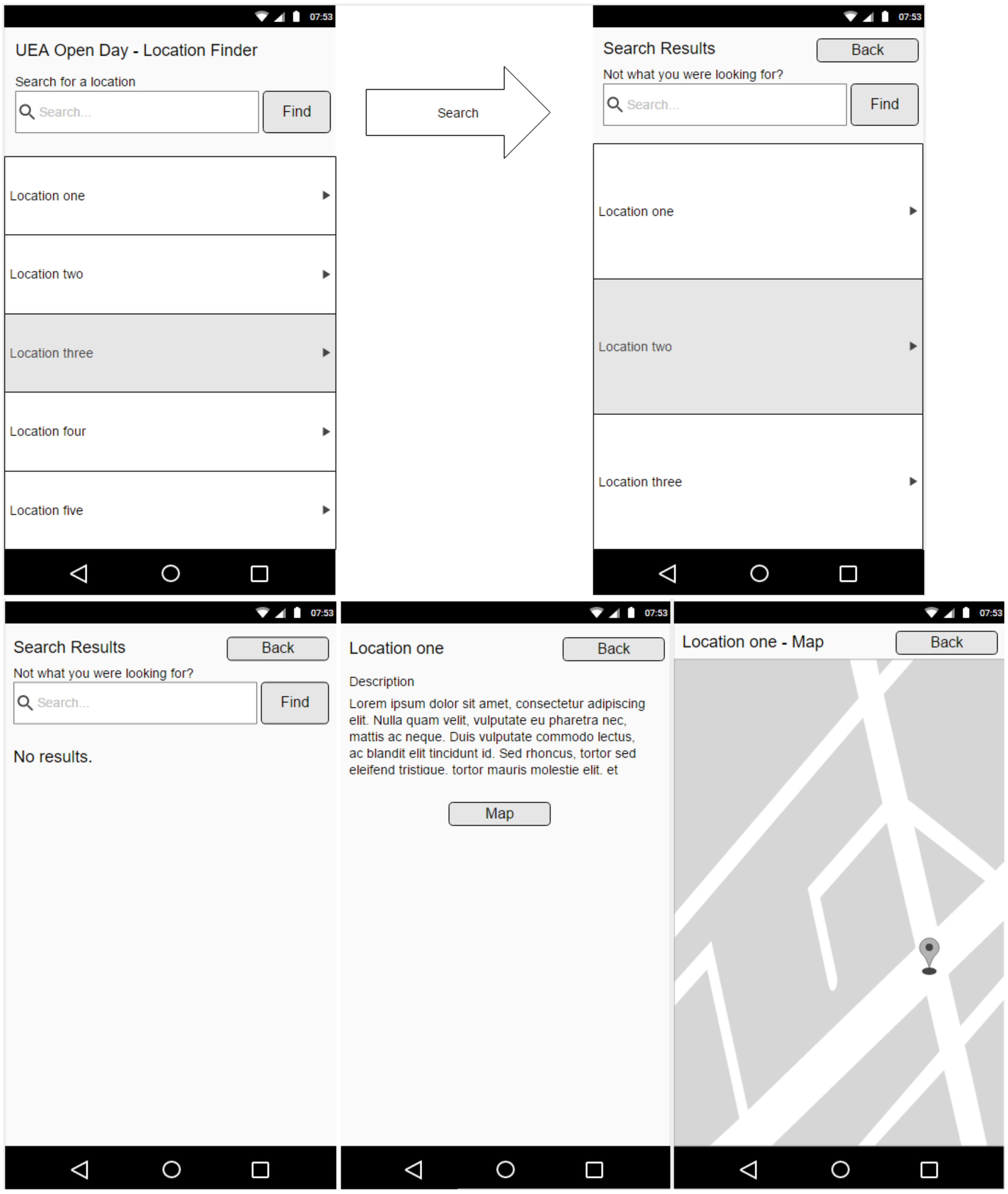


Appendix H User Interface

Appendix H1: Activity Program UI

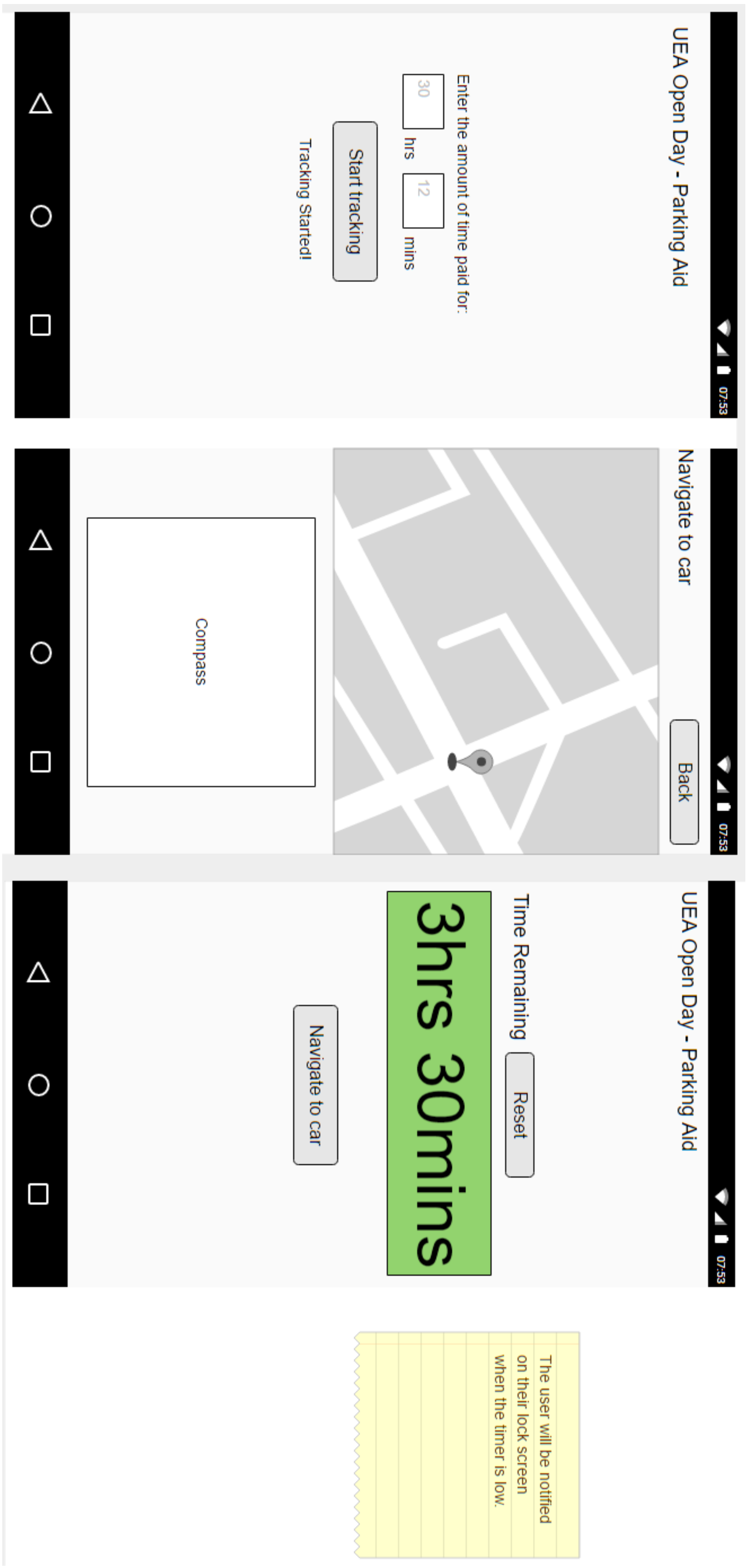


Appendix H2: Location Finder UI



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et

Appendix H3: Parking Aid UI



The user will be notified on their lock screen when the timer is low.

Appendix I Glossary of terms

Below is a list of clarified specific terms and definitions regarding our system.

SAAF: Simple Android Application Framework. A collection of classes used to develop Android applications.

Emerging technology: the ability to support advanced features such as enhancing GPS communication between modes of transport and tagging the GPS system to locate it.

Requirements: what the potential visitor would need from the system.

Gadget: a type of technical device. E.g. Phone, tablet.

Visitor: visitor to the UEA whose car is parked.

Point of interest: a specific point or location that a visitor may find useful or interesting.

Online Location: content offered via directories and links over the Internet and the World Wide Web.

GPS: stands for Global Positioning System. Is a navigation system that is used to determine the users exact location.

Interface: connect with and use the system by an interface.

URL: stands for Uniform Resource Locator. It is a reference (an address) to a resource on the Internet.

Compass: pointer which shows the direction of magnetic north and bearings from it.

Visual Indication: view of the location area that the user is in and giving directions to get to your car using the GPS tag, the current GPS location and the compass.

Parking period: a set time where the users can park their car in the parking area, depending on the amount of money the user put in for their parking ticket.

Imported: bring an idea/source/service etc. from a different place or context. In this circumstance, importing a program from another source into the app.

Societal trends: users' expectations of the system.

Synchronise: in the context of computing, it is the process of a set of data or files to remain identical in more than one location.

Generic mechanism: a system made of efficient and reusable software interacting with processes that produce one or more effects or operations.