

CARLETON UNIVERSITY  
SCHOOL OF COMPUTER SCIENCE

COMP 4905 Honours Project

Life Planner

By: Ali Hamadi  
December 15, 2014

Project Supervisor: Dr. Michel Barbeau  
School of Computer Science

## **Abstract**

The role of smartphones is increasingly becoming an important and vital part of our lives. Mobile data traffic in Canada is expected to grow 900% from 2013 to 2018 [O'Shea, 2014]. Our daily activities that were once recorded on hardcopies (phonebooks and grocery lists) have now shifted into the smartphone domain. Life Planner aims to combine the functions of an alarm application and note application, essentially creating an agenda/planner for the smartphone. This report focuses on the design and thought process taken throughout the development of Life Planner. This paper starts with an introduction that explains the project and its rationale. Following this, is a rundown of the application's design process, as well as a brief look into the features implemented. Next, is the user study plan, which entails the various steps involved in conducting the testing of the application's usability and design. The results of this user study are then illustrated and analyzed to grade the overall application, as well as help improve it through participants' feedback. Finally, this paper concludes by stating the project results, as well as final thoughts, such as limitations, technologies and concepts learned, and future improvements.

## **Acknowledgments**

I would like to express the deepest appreciation to my research project supervisor, Dr. Michel Barbeau, who has given the freedom to select my own topic, which in return, translated into a passionate, exciting adventure in developing this application. Without his assistance and trust, this project would not have been possible.

I would also like to thank Dr. Sonia Chiasson, my COMP 3008 instructor, for teaching me the concepts behind human-computer interactions and the importance of developing an articulate user study.

# Table of Contents

1. Introduction .....	8
1.1 – Motivation .....	8
1.2 – Objectives .....	9
1.3 – Background .....	10
1.3.1 – Java .....	10
1.3.2 – XML .....	10
1.3.3 – Android .....	10
1.3.4 – SQLite Database .....	11
1.3.5 – Support Libraries .....	11
2. Application Design .....	12
2.1 – Android Manifest .....	12
2.2 – Java Classes .....	15
2.2.1 – MainActivity .....	15
2.2.2 – CreateTask .....	17
2.2.3 – ViewTasks .....	18
2.2.4 – Settings .....	20
2.2.5 – About .....	21
2.2.6 – TaskNotification .....	22
2.2.7 – Database .....	23
3. User Study Plan .....	24
3.1 – Methodology .....	24
3.1.1 – Description .....	24
3.1.2 – Tasks for Users to Attempt in Lab Session .....	24
3.1.3 – Order of Steps for Lab Session .....	25
3.2 – Assessment Measures and Metrics Description .....	25
3.3 – Post-test Questionnaire .....	26
3.4 – Pilot Session .....	26
4. User Study Analysis and Results .....	27
4.1 – Qualitative Data Analysis .....	27
4.1.1 – Interpretation & Discussion .....	28

4.2 – Quantitative Data Analysis .....	29
4.2.1 – Perception of the System .....	29
4.2.2 – Usage of the System .....	30
4.2.3 – Interpretation & Discussion .....	30
4.3 – Overall Assessment .....	32
4.3.1 – Interpretation & Discussion .....	32
5. Conclusion.....	33
5.1 – Project Results.....	33
5.2 – Limitations .....	33
5.3 – Future Improvements .....	34
5.4 – Contributions to the Field.....	34
5.5 – Concepts and Technologies Learned .....	34
References .....	36

## List of Figures

Figure 1: Components of an AndroidManifest.xml file.....	12
Figure 2: Example of an AndroidManifest.xml file .....	13
Figure 3: Main screen of Life Planner .....	16
Figure 4: Options context menu .....	16
Figure 5: A field left blank during task creation .....	18
Figure 6: A day with scheduled tasks.....	19
Figure 7: A day with no scheduled tasks .....	19
Figure 8: The Settings screen .....	20
Figure 9: List of available colours .....	20
Figure 10: The About activity.....	21
Figure 11: The system notification .....	22
Figure 12: A visual display of the total scores from Table 2.....	31
Figure 13: A visual display of the total scores from Table 3.....	31

## List of Tables

Table 1: Feedback from the field study .....	27
Table 2: Results of question 2 from the questionnaire with a calculated mean, median, variance and total score .....	29
Table 3: Results of question 3 from the questionnaire with a calculated mean, median, variance and total score .....	30

# 1. Introduction

## 1.1 – Motivation

Smartphones are increasingly becoming an important and vital part of our lives. Mobile data traffic in Canada is expected to grow 900% from 2013 to 2018 [O'Shea, 2014]. Our daily activities that were once recorded on paper have now shifted into the smartphone domain. You will rarely see a personal phonebook being used in this day of age. People have transitioned to storing all of their contacts' information on their phones, which virtually hold no weight as opposed to a hardcopy (phonebook). Another example of a transition from hardcopy to smartphone is grocery lists. There are multiple mobile applications that allow users to share an accessible grocery list, giving them the option to add, edit or delete items on the list. Similarly, I believe the role of agendas and planners can be transferred to the smartphone, which is what this project aims to achieve.

Life Planner will prevent the last-minute panics before a deadline that many people experience in their lifetime. There are currently two types of applications – system applications and user applications. System applications are pre-installed on the smartphone and stored in `/system/app/`, which is a read-only location without root access. On the other hand, user applications can be installed via the Google Play Store, Amazon Appstore, third-party markets, or directly installing the Android application package (APK), also known as sideloading. Currently, there is a system application for setting alarms (Alarm and Clocks), and another application for writing notes (Memo). However, there is no application that combines the two. The application discussed in this report, Life Planner, aims to bring the underlying functionality of these two applications into one.

I chose to develop an application on the Android platform for multiple reasons. The first being for convenience, as I own an Android device and have never used an iOS device. The second, and more notable reason, is that the Android fan base absolutely eclipses the iOS. As of 2014, Android leads the global smartphone market with 84.7% of the market share (255.3 million units sold), while iOS accounts for only 11.7% (35.2 million units sold) [IDC, 2014]. In other words, there are seven Android users for every iPhone. I have always wanted to develop a mobile application and I believe this is the perfect opportunity.



## 1.2 – Objectives

The purpose of this project is to develop a practical Android application for everyday use. I initially planned for my application to be strictly targeted towards students. It was originally designed in a way to allow students to create reminders for deadline such as projects, assignments, and exams. After some thought, I decided to expand my target demographic by allowing the users to customize their tasks. Instead of merely selecting a type of task from a predefined list, users will be able to supply their own type, whether it is exams, appointments, or simply errands. I also wanted to create a simple and user-friendly interface. This would be achieved through continuous feedback from participants that test my application, which brings me to another objective for this project.

Software engineers are often responsible for interacting with the clients to gain a thorough understanding of the requirements needed for the desired software. Since I am in the Software Engineering stream, I wanted to put more of an emphasis on the documentation aspect of this project. I believe the preparation that comes along with software development is as important, if not more important, than programming the software. It is my goal to demonstrate my skills and expertise that I have acquired over the course of my undergrad. Some of the responsibilities that I will take on include requirement gathering, user study sessions, and keeping an open, frequent communication with the participants involved in the user study. Life Planner is, after all, created for the people. Therefore it would make most sense to design the project based on participant feedback.

A personal objective of mine is to demonstrate my capabilities of developing software for a new platform. I have no prior experience with Android development or XML files. This new challenge will guarantee that I will have at least learned something at the end of the project. I believe that learning to adapt quickly to new integrated development environments, platforms, or programming/markup languages is a great asset to have in my field of study.

## **1.3 – Background**

### **1.3.1 – Java**

Life Planner is coded in Java. Java is an object-oriented, concurrent programming language that was released in 1995 by Sun Microsystems. Most of Java’s syntax was derived from C and C++; however, it is simplified to prevent common programming errors, such as pointers. Java was developed in a manner that would allow developers to “write once, run anywhere” (WORA), implying that compiled code that has run on a platform can run on another platform without being recompiled. Java applications are generally compiled to bytecode, which serve as the instructions for the Java virtual machine (JVM). The key difference between programming in Java for Android and for a desktop application is that the Android platform does not use Java’s class libraries, and therefore, does not use JVM to execute code. Instead, it translates the Java bytecode to Dalvik “dex-code” (Dalvik Executable), which is then read by the Dalvik virtual machine (DVM) with just-in-time compilation (JIT). JIT, also known as dynamic translation, compiles the code at run-time, rather than prior to execution. Google’s open-source Android operating system utilizes this DVM specifically to execute android applications.

### **1.3.2 – XML**

XML, short for Extensible Markup Language, is a markup language that is used to outline a set of rules for encoding a format that is both machine-readable and human-readable. In this project, XML was responsible for designing the graphical user interface (GUI). I was able to straightforwardly create elements such as buttons, text views, and text fields and position them where I liked by utilizing the attributes associated with each element. Each element can be identified by a unique ID, which can be referenced in the Java classes. This gives the developer the option to modify any attributes of an element in real-time, allowing for the layout to change during run-time.

### **1.3.3 – Android**

Android is a Linux-based mobile operating system that was developed by Google. It was primarily designed for touchscreen mobile devices such as smartphones and tablets. Life Planner was developed on the Android platform via the Eclipse integrated development environment (IDE) by first installing Google’s free Android software development kit (SDK). I then installed the Android development tools (ADT) plugin. This plugin allowed me to create my Android project, create my application’s GUI, add external libraries to my project, export signed APK

files to install my applications on devices without the use of a USB (participants from user study received APK files), and debug code by utilizing the SDK tools which extend from Java. Life Planner targets devices with Android 2.3 or under because that is the version of the device I am using to test my application on. I initially planned to target a higher version by using a virtual device emulator to test my application, however, the emulators provided by the SDK were too slow and often crashed before even launching. Thus, I conducted most of my tests on my Android device by connecting it to a computer via USB cable.

#### **1.3.4 – SQLite Database**

Life Planner utilized SQLite to store the date, time, task ID, name and details. SQLite is a light-weight relational database management system that was developed by Richard Hipp and released in 2000. SQLite was the most optimal choice of database because it is great for local data storage and individual applications and devices. It worked exceptionally well with Life Planner because of its use of powerful SQL queries to access and update data, greatly reducing the complexity of the application code. Updating or editing data in SQLite only overwrites the parts of the file that were modified, rather than the entire file. This improves the performance and reduces wear on the SSD drive as Life Planner allows users to continually edit tasks after creation. Life Planner utilized the *SQLiteOpenHelper* class which was added in application programming interface (API) level 1. It is a helper class that manages database creation and version management.

#### **1.3.5 – Support Libraries**

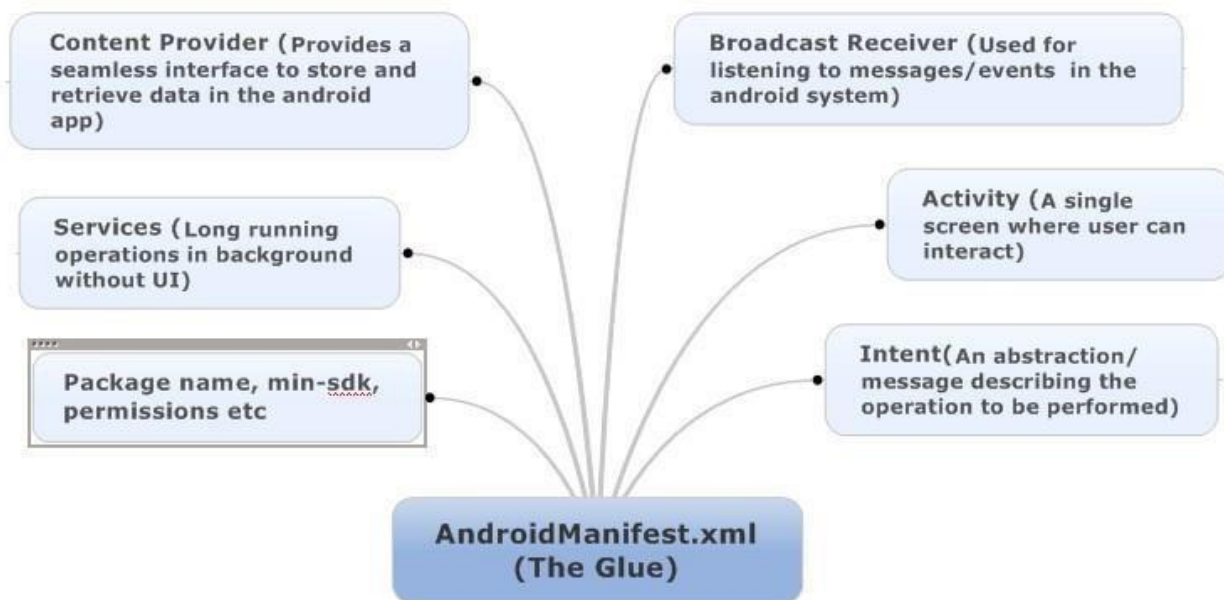
Support libraries allow applications targeted at old android versions to use fragments from newer APIs. All support libraries are backwards-compatible to a certain API level, meaning that my application can use the libraries' features and still be compatible with old devices (2.3, in this case). The *ActionBar* APIs were added in Android 3.0 (API level 11), making it incompatible with my Android device. In order for my application to properly display the action bar at the top of the screen, I had to download the *v4* and *v7 appcompat* support libraries. Instead of extending my main activity class to *Activity*, I extended it to *ActionBarActivity* and imported the *android.support.v7.app.ActionBarActivity* class, while having my *android-support-v4.jar* file in the */LifePlanner/libs/* folder.

## 2. Application Design

This section will explain the object-oriented design and GUI behind Life Planner. It will also explain the rationale behind the features. The Android manifest XML and each of the seven Java classes will be discussed to demonstrate their contribution to the overall completion of the project.

### 2.1 – Android Manifest

The *AndroidManifest.xml* file outlines essential information regarding the application to the Android system. On project creation, an *AndroidManifest.xml* file is automatically created and stored in its root directory. The application will not run without this file, or even if the file is renamed because the Android system traverses to the root folder to find ‘AndroidManifest.xml’, and reads the contents before running any of the application’s code. An android application is comprised of various components, as shown in figure 1. The manifest’s purpose is to basically piece these components together to form a complete project.



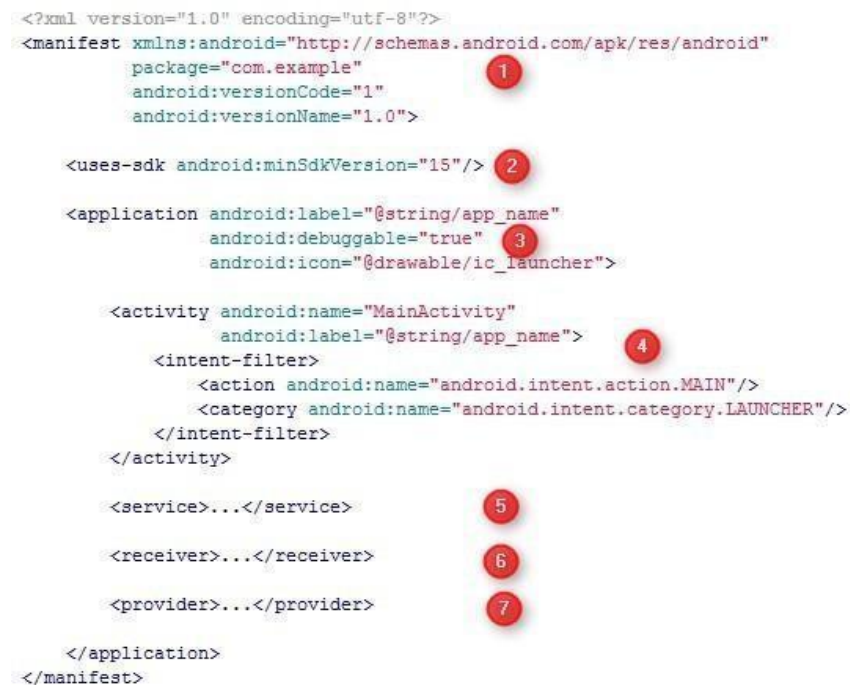
**FIGURE 1:** Components of an *AndroidManifest.xml* file.

The following list breaks down some of the elements involved in an *AndroidManifest.xml* file and how they impact the application:

1. The first element in the file is the manifest element. The `<xmlns:android>` attribute defines the android namespace and should always be set to the same web address, as indicated in figure 2. The second attribute is the package, which serves two main purposes:
  - Uniquely identifies the application. The package name defines the application's identity; therefore, changing the package name after publishing the application will cause the system to read the application as a different one. This prevents users from updating the application to the new version, since they are viewed as two unrelated applications.
  - Allows classes to be organized in various folders. For example, a client/server system can have a client package and a server package. Another example is a project with a model, view, and controller (MVC) architectural pattern can have a package for each part. Classes in one folder cannot have the same name, but if they were placed in separate packages they can.

The version code attribute is used to determine whether one version is more recent than another. This allows the application to know whether there is an update to be done. The version code is set as an integer and stored internally. A higher version code implies a more recent version. The version name attribute, set as a string, is the version code that the users see; therefore, it is not stored internally.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="15"/>
    <application android:label="@string/app_name"
        android:debuggable="true"
        android:icon="@drawable/ic_launcher">
        <activity android:name="MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <service>...</service>
        <receiver>...</receiver>
        <provider>...</provider>
    </application>
</manifest>
```



The diagram shows an example of an *AndroidManifest.xml* file. Seven red circles with white numbers are placed next to specific elements in the XML code to highlight them:

- 1. Points to the `xmlns:android` attribute in the `<manifest>` tag.
- 2. Points to the `<uses-sdk>` tag.
- 3. Points to the `android:debuggable="true"` attribute in the `<application>` tag.
- 4. Points to the `<activity>` tag.
- 5. Points to the `<service>` tag.
- 6. Points to the `<receiver>` tag.
- 7. Points to the `<provider>` tag.

**FIGURE 2:** Example of an *AndroidManifest.xml* file.

2. The `<uses-sdk>` element allows me to express the compatibility levels of an application based on the API levels. The `miniSdkVersion` attribute takes an integer and sets it as the minimum API level required by the Android device to run the application. A device with an API level less than the `miniSdkVersion` will fail to install. Applications that do not declare this attribute in the manifest will have a default minimum API level of 1, indicating that any device can run the application. However, if this is not the case, the application will crash at runtime when it attempts to access the unavailable APIs. There are also the `<targetSdkVersion>` attribute, which specifies the exact API level, and the `<maxSdkVersion>`, which is primarily used during updates to check compatibility.
3. The most important element in the manifest file is the `<application>` element because it describes and sets the behaviour of the activities, services, receivers and providers of the application. Since the virtual machine emulator provided by Android Eclipse was not efficient in terms of speed, I tested the application via an Android device. The `<label>` attribute takes a string and sets it as the name of the application, with the `<icon>` attribute taking the file path of the icon for the application, generally in the *drawable* folder.
4. The `<activity>` element is responsible for listing the activities the application can run, along with their intents. If an activity is not declared in the manifest it will never be run because the system would not recognize it. The `<name>` attribute indicates the name of the Java class, while the `<label>` attribute indicates the title to be displayed (in the action bar). We can specify which activity we want the application to display upon launch time inside the `<intent-filter>` element by assigning it a *LAUNCHER* category as displayed in figure 2. My *AndroidManifest.xml* file contained 5 activities in the *ah.lifeplaner* package, namely, *MainActivity*, *CreateTask*, *Settings*, *About* and *ViewTasks*.
5. The `<service>` element lists all the services that are utilized by the application. Services have no user interface. They run in the background, regardless of what is being performed on the screen. Similar to activities, any services not listed in this element will never be run. Life Planner has one service class, *TaskNotification*.
6. The `<receivers>` element lists all broadcast receivers, which are typically implemented to respond to events/intents that occur in the system. Each receiver can be set to respond to certain events. For example, an event is broadcast to all the receivers by the system when an SMS is received, but one of these receivers can be configured for handling the SMS and performing an operation upon receiving it. Receivers can be declared to the system by adding it to the manifest, or by creating the receiver dynamically in code and registering it with the *Context.registerReceiver()* method. After researching the practical uses of broadcasting receivers, I did not find it necessary to add one for Life Planner.

7. The `<providers>` element lists the content provider components, which are subclasses of *ContentProvider*. These providers offer structured access to data that is managed by the application. For example, Android shares a user's contact information as a content provider, allowing developers to use this information in their application via content provider APIs. Similar to services, any content provider that is not listed in the `<providers>` element will not run. Since Life Planner creates its local data from scratch and does not require information from the Android system, I deemed it unnecessary to use a content provider.

## 2.2 – Java Classes

Life Planner consists of seven classes:

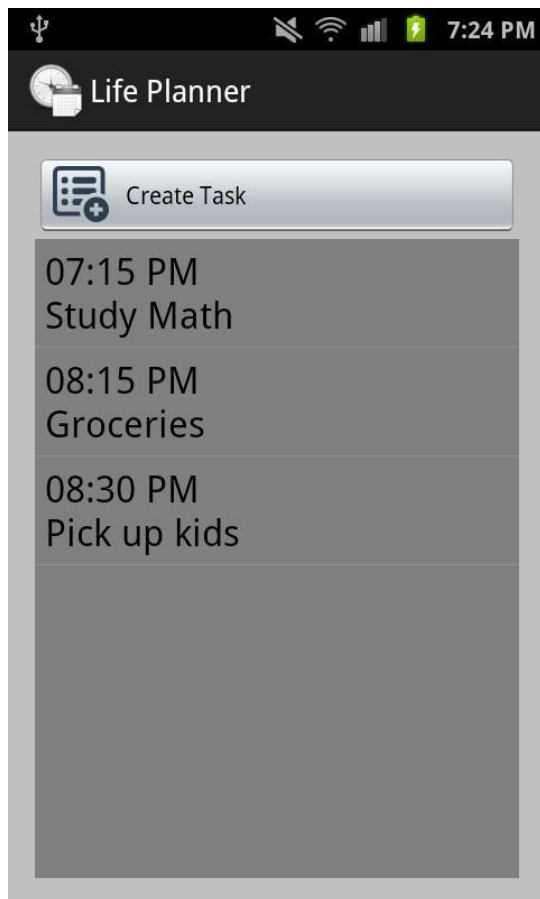
- *MainActivity*
- *CreateTask*
- *ViewTasks*
- *Settings*
- *About*
- *TaskNotification*
- *Database*

### 2.2.1 – MainActivity

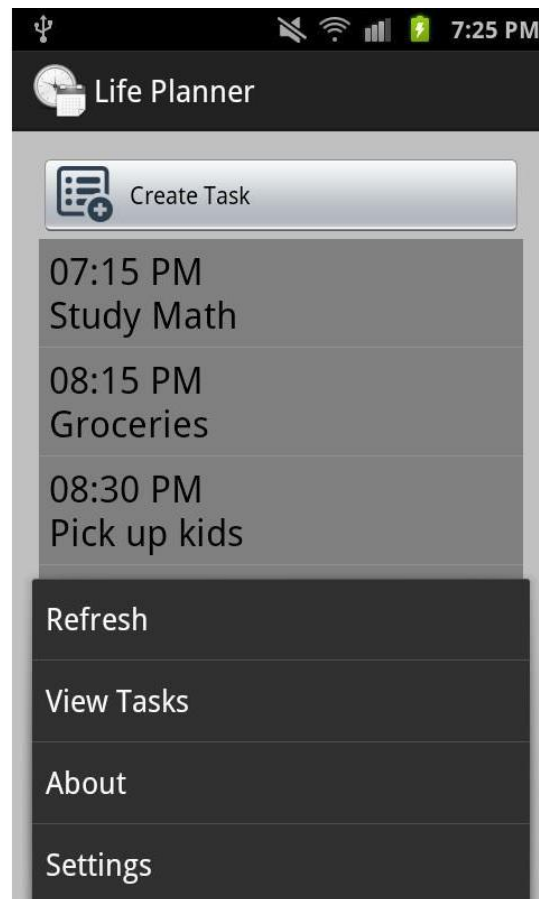
The *MainActivity* class, like the name suggests, is the main class of the project. It is the executable file that launches the applications as indicated in the manifest file. This activity class extends *ActionBarActivity*, supplied by the *appcompat v7* support library, and initializes the GUI by reading the *main.xml* file located in the */res/layout/* folder. It is responsible for the main screen of the application. The button and list view on this screen are initialized in the *onCreate()* method. This method can be compared to a constructor method, in that it is run on application launch. . It is responsible for the main screen of the application. At this point, the user has multiple options to choose from, however, the most visible option would be the *Create Task* button placed at the top of the screen. The other options require the user to click the *Settings* button on their Android device, which would open a menu item list via the *onOptionsItemSelected()* method.

I chose this design because I want the user to only worry about the essentials of the applications, which would be creating a task. I wanted to minimize the elements on the page to give a feel of simplicity, as cluttering the action bar with buttons would make the user ponder on scenarios that stray from the main function of this application. The default button supplied by Android was not appealing, so I opted to customize it to give the user a better experience. I added an icon of a list to the *Create Task* button to help the user interpret its function. I was also displeased with the default layout of the button so I added padding and alignment attributes to give it an organized design. Ultimately, this activity class serves 3 main functions:

1. Display a method (button) for the user to navigate to the *CreateTask* screen (see Fig 3).
2. Display a list of today's tasks (see Fig. 3).
3. Display a context menu upon clicking the *Options* button (see Fig. 4).



**FIGURE 3:** Main screen of Life Planner.



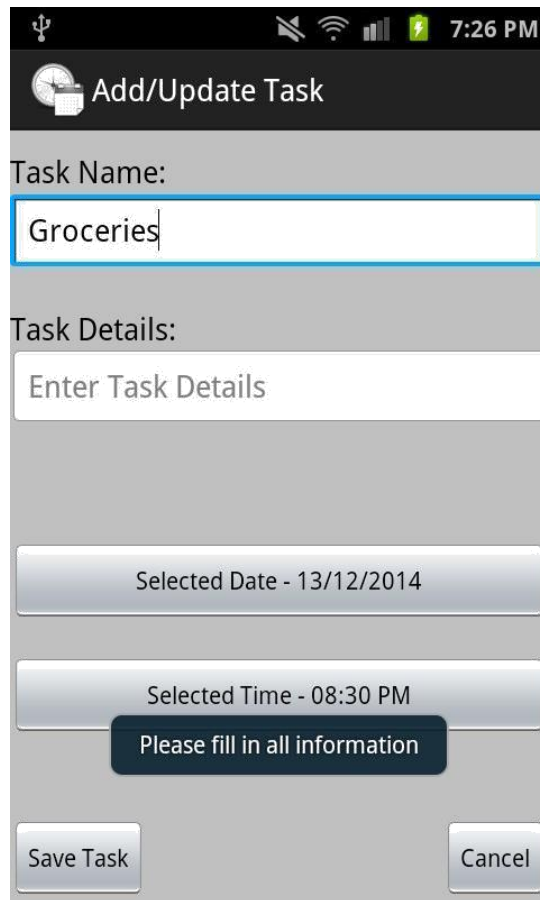
**FIGURE 4:** Options context menu.



### 2.2.2 - CreateTask

The *CreateTask* class is responsible for reading the user's input and storing it in a database. Of course, the information entered by the user is validated before added to the database. The user must select the *Create Task* button from the main screen to get to this activity. There are four pieces of information that the user is required to fill in before adding a task. The first would be the name of the task being created. This name basically summarizes the task details in one or two words, ideally. It has a maximum character length of 30. The second piece of information, the task details, is used to give the user a better understanding of the task to plan for. The reason I added this attribute is because users would have a difficult time remembering specifics about the tasks when the number of tasks got too high. Task details are meant to be a brief breakdown of what the user wants to accomplish. The third piece of information is the date of when the user wants to be reminded of the task. I needed to ensure that the application did not allow users to create tasks for the past, as that would not fully optimize the function (alarm feature) of Life Planner. I say fully because reminding the user is only part of the core purpose of this application, since users can also use Life Planner as a catalogue that keeps track of all tasks, even past tasks. The alarm notification is still equally as important.

New methods for setting the minimum and maximum dates of a *Date Picker* were added in API level 11. Unfortunately, the Android device used for testing this application was below API level 11, therefore I had to improvise and write code to check whether the date was valid. The fourth and last piece of information required from the user is the time of day. This button is initially disabled until a valid date is supplied. The reason for this is because the date must be known for the code that checks the time validity to work properly and free of bugs. I chose to use the 12-hour clock system because it was easier to use than the 24-hour clock system, based on participants' feedback. I created a *convertTime()* method that took the hour and minutes as arguments, and converted them to the AM/PM format. After filling in the task name, details, date and time, the user can click the *Save Task* button to add the task to the list. If successful, a short message is displayed on the screen notifying the user that the task was added and redirects the user to the main screen. If unsuccessful, a message with an explanation pertaining to the error is displayed (see Fig. 5).



**FIGURE 5:** A field left blank during task creation.

### 2.2.3 - ViewTasks

Since people typically tend to make plans for the future, I added this activity to display a list of tasks that are scheduled for a specified date. To get to this screen, the user must be on the main screen and press the *Options* button on the Android device and select *View Tasks* from the menu item list that appears. At this point, a *Date Picker Dialog* will appear for the user to set a day, month and year. I also wanted this application to serve as a catalogue, so I allowed users to enter past dates. These values are then stored in the 'day', 'month' and 'year' variables in the *ViewTasks* activity by adding the following code in *MainActivity*:

```

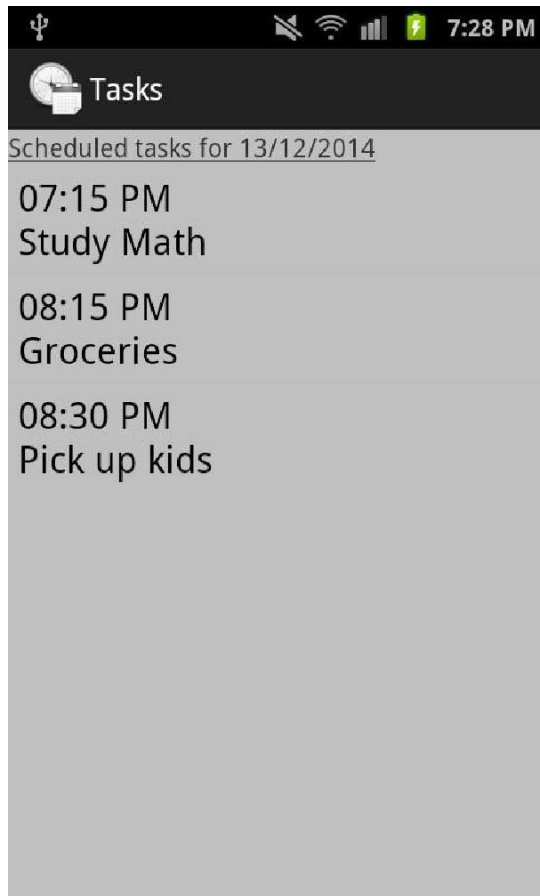
DatePickerDialog.OnDateSetListener dateSetListener = new DatePickerDialog.OnDateSetListener() {

    @Override
    public void onDateSet(DatePicker view, int year, int month, int day)
    { updateColours();

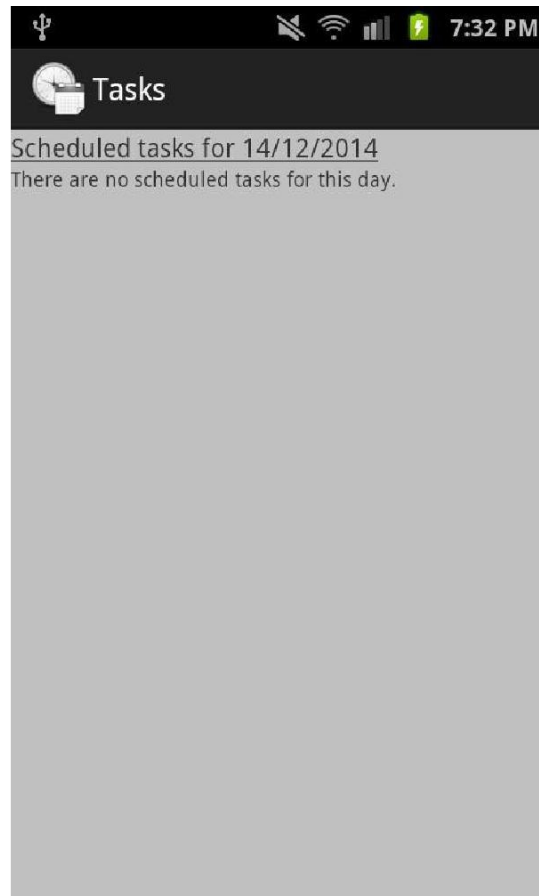
        //Create a new intent for the ViewTasks class and pass in the values of the
        //date. Intent intent = new Intent("ah.lifeplanner.ViewTasks");
        intent.putExtra("day", day);
        intent.putExtra("month",
            month); intent.putExtra("year",
            year); startActivity(intent);
    }
};

```

This activity is then started and the tasks of that day will be listed (see Fig. 6). Pressing on an item in the list opens a menu that gives the user an option to delete or view/update a task. If there are no scheduled tasks for that day, there will be a message stating so to avoid any confusion or misunderstanding from the user (see Fig. 7).



**FIGURE 6:** A day with scheduled tasks.



**FIGURE 7:** A day with no scheduled tasks.

## 2.2.4 – Settings

This class is comprised of bonus features I wanted to include in the project. The user must select *Settings* from the option menu on the main screen to reach the *Settings* screen. The *Settings* class extends *PreferenceActivity* and loads all its contents from the *settings.xml* file. It is important to know that all of these elements are not essential for the application to fulfil its purpose, they are merely preferences. The user can set a username, background colour and list colour (see Fig. 8). I found that adding these customizable options for the user would make the application unique for each individual. The notification message will display the username to give it a personalized feeling. Anthropomorphism is attributing human characteristics to inanimate objects. I wanted my application to be anthropomorphic for users who adore that type of relationship, and being called by your name (via notification) is definitely a trait of human behaviour. I wanted to ensure that I chose a wide array of colours for the user to choose from (see Fig. 9), so I picked the 17 most common colours. It is difficult to choose one default colour and try to justify it because colour is subjective. This was the underlying reason for giving the user the power to choose their preferred colour.

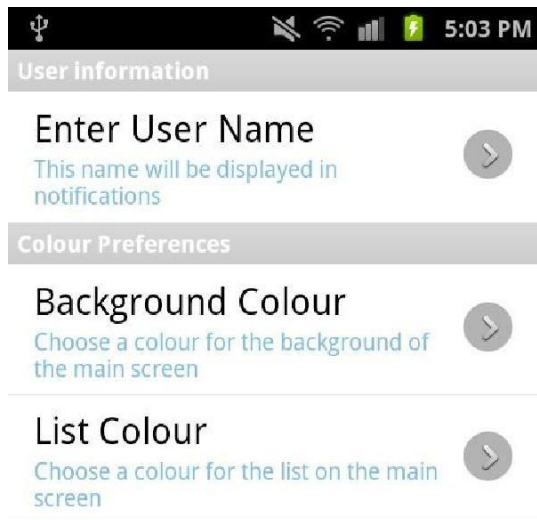


FIGURE 8: The Settings screen.

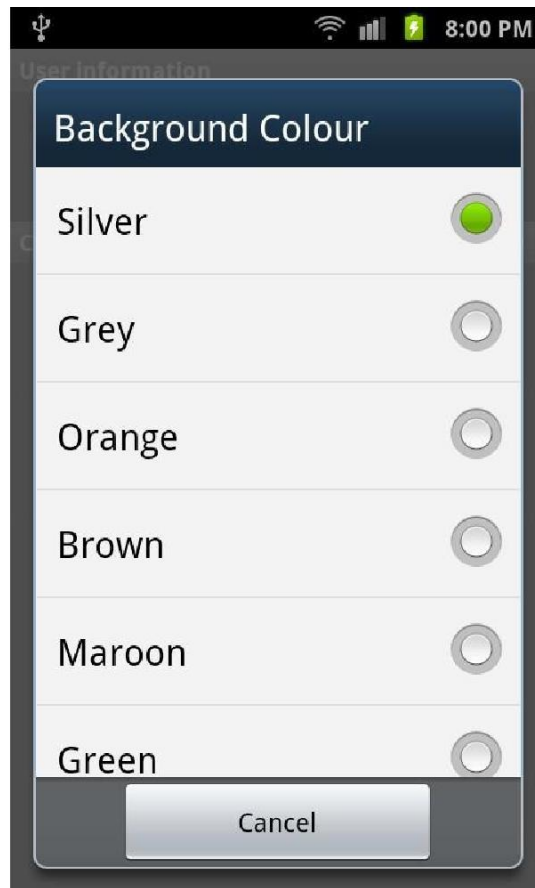
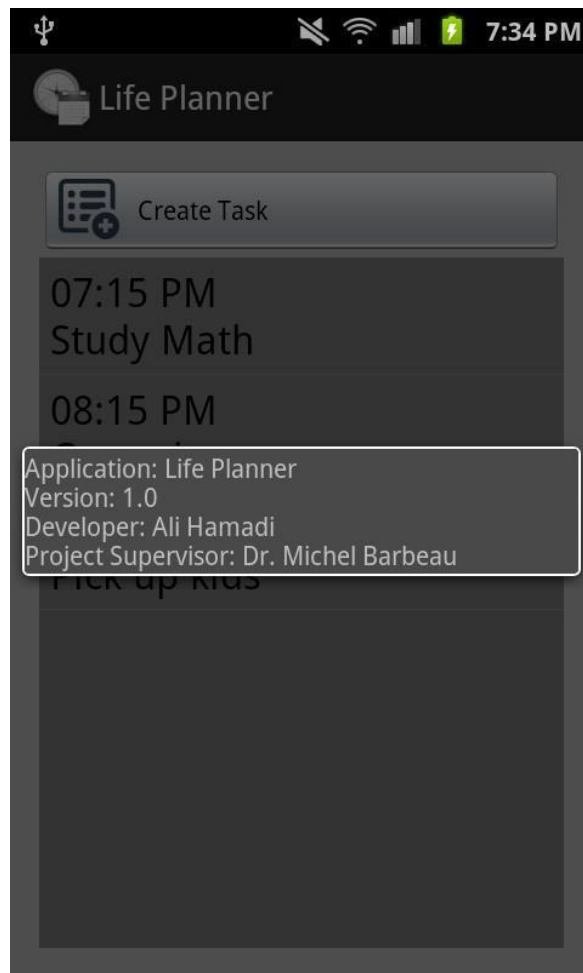


FIGURE 9: List of available colours.

### 2.2.5 – About

This activity simply displays a small text box with general information relevant to the application. This information includes the name of the application, version, developer and project supervisor (see Fig. 10). This activity is started upon selecting the *About* item in the options menu from the main screen. Since the purpose of this screen is to be brief and informational, I did not want to take up the entirety of the screen with so little text. I contained the text in a small box by using the graphical layout tool supplied by Android's ADT plugin.



**FIGURE 10:** The *About* activity.

## 2.2.6 - TaskNotification

This class is responsible for displaying the notification message at the right time and date. The *TaskNotification* class extends *Service*. *Service* is an application component that allows my application to run long-lasting operations in the background, without the need for interaction from the user. To use this service, I used a `<service>` declaration in *AndroidManifest.xml*. Basically, the *Notification()* method in this class checks if the date and current time of each task is equal to the current time and date by utilizing a while loop. If a match is found, the method makes a call to the *showNotification()* method with the information I wish to display in the message, which in this case is the task name and details (see Fig. 11).

I initially had it so that whenever a notification was displayed for a task, that task was automatically deleted from the list. I felt that it would free up space to prevent a cluttered home screen, and that completed tasks serve no purpose. However, the feedback I received from the user study indicated that roughly half the users did not want the tasks to be deleted. They wanted to keep a catalogue of their tasks, as I mentioned earlier. They also stated that at the end of the day, they wanted to look back at all the tasks they had completed for that particular day. Thus, instead of having the task automatically deleted upon its notification, I gave the users the option to delete a task from the list.

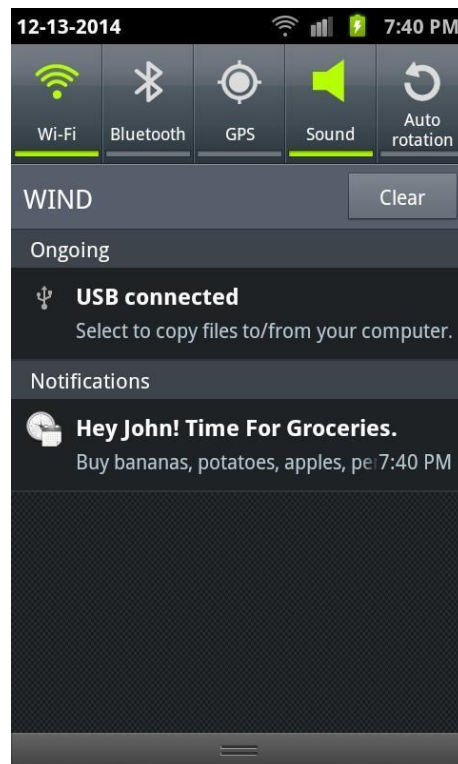


FIGURE 11: The system notification.

### 2.2.7 - Database

The *Database* class extends *SQLiteOpenHelper* and is for creating the database if it does not already exist, opens the database if it does, and upgrades it as necessary. The database consists of 8 attributes (also known as variables). The *\_id* variable is the unique key of the database schema. The *\_id* variable is incremented every time a task is created, as seen in the code fragment below.

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE TASKS (_id INTEGER PRIMARY KEY
        AUTOINCREMENT, NAME text, DETAILS text, MINUTE integer, HOUR integer,
        DAY integer, MONTH integer, YEAR integer)");
}
```

### **3. User Study Plan**

#### **3.1 – Methodology**

##### **3.1.1 – Description**

The methodology for this user study will use a hybrid approach. This approach will incorporate a field-based study, followed by a lab session. The field-based approach allows the user to test the application in a regular setting for more than a day. It is potentially more ecologically valid than other approaches because it offers real-life, practical use of the application. I deployed the Life Planner APK file, along with a tutorial on how to set it up to the participants via email and received feedback from the users twice a week. Following the field-based study will be one lab session that will take place in a controlled environment. This approach includes direct observation while the user “thinks-aloud”. This will help me record the participant’s comments, and give insight into the user’s decision process or attitude. The session will start off with closed questions because they are straight-forward and simpler to answer. This will give the user confidence and comfort. I will then eventually move on to open questions where the user has the opportunity to elaborate and/or express his/her thoughts. I will use simple language, no compound questions, and remain neutral to avoid giving the participant the illusion of a correct answer. The questions will be ordered logically to avoid confusion. I will be using semi-structured interviews, one of the most common interview formats in human computer interactions. Unlike a structured interview, a semi-structured interview will combine a set of pre-determined questions with giving the user the opportunity to freely discuss and explore the application at will.

##### **3.1.2 – Tasks for Users to Attempt in Lab Session**

1. Add a task
2. Update a Task
3. View the tasks of a specific day (future or past day)
4. Set a username and background colours for the application
5. Delete a task

The tasks will be given in the same order to all participants to keep the procedure consistent. This minimizes the effect of outside factors on their results.



### **3.1.3 – Order of Steps for Lab Session**

1. Greet the user.
2. Introduce the purpose of the study (carefully so that I remain neutral). Make small talk with the user to give a casual and comfortable atmosphere to make him/her feel at ease.
3. Hand the consent form to the user and respectfully ask him/her to read, and then sign it.
4. Get general participant background information.
5. Conduct the interview (including the 5 tasks to attempt).
6. Administer post-test questionnaire.
7. Allow users to give an opinion and/or suggestions.
8. Finish with some easy, non-controversial questions.
9. Wrap up, debrief.
10. Thank the participant.

### **3.2 – Assessment Measures and Metrics Description**

I will be assessing the users' input on the interface and usage of my application through questionnaires and general discussions. The interface's purpose is to attract people to the system, while its functionality's purpose is to keep those same people using the system. One will not work without the other, therefore it is important to have an appealing interface with easy to use features. I will ask the users specific questions which will allow me to translate their results into a mean likert-scale so that they can be visually displayed. These questions will be administered to the participant during the lab session. Users will be asked to give a (numerical) score for each of these specific questions. Users will then be asked about their perception, opinion and preference with respect to the interface. I will then ask them about the simplicity, flexibility and usage with respect to the functional system. Quantitative data, best illustrated through visual graphs, will be gathered after the user study.

The usability of my application will be deemed acceptable if the overall score is above 50%. Participants will be told to answer questions using a scale of 1-5, with 5 being the best score. Since a score of 3 indicates an indifferent opinion on the question, it marks the neutral line on the mean likert-scale. Anything below a score of 3 will be viewed as a failure.

### **3.3 – Post-test Questionnaire**

1. On a scale of 1-5, 1 being not appealing at all, 3 being indifferent and 5 being very appealing; how was the (1) font, (2) size, (3) colours and (4) overall layout of buttons and text?
2. On a scale of 1-5, 1 being very difficult, 3 being indifferent and 5 being very easy, how easy was it to complete each of the 5 tasks?
3. Are there any aspects of the application that need to be simplified or improved in any way?
4. Are there any features you would recommend to add or remove from the application? If so, what are they?
5. Which of the following best describes how many times you see yourself using this application in the future:
  - a. 1-3 times a day
  - b. 4-7 a day
  - c. 8+ times a day
  - d. Never

### **3.4 – Pilot Session**

My session took 11 minutes to complete. The questions were straight-forward and easy to understand. I found myself going back to look at the application when answering about specifics, such as the font colour, size or organization of the various buttons. The task descriptions were intuitive and gave a clear objective to follow immediately. The application was easy to navigate alongside the list of tasks to attempt.

## 4. User Study Analysis and Results

### 4.1 – Qualitative Data Analysis

The following notes were taken from participants' feedback during the field study. Participants sent me their feedback twice a week, for 2 weeks (total of 4 evaluations). The android application was modified after each evaluation based on the feedback, and the new updated version was sent to the participants for further evaluations.

**TABLE 1:** Feedback from the field study, over a span of 2 weeks.

Date	Feedback
17/11/2014	<ul style="list-style-type: none"><li>- The application's main requirement is satisfied; users can add and remove tasks.</li><li>- A visual bug when selecting a single digit minute in the time button. The button does not display the correct information. For example, selecting 10:03 on the dialog shows 10:3 on the button.</li><li>- Changing the colours in the settings menu crashes the application.</li></ul>
20/11/2014	<ul style="list-style-type: none"><li>- There should be a wider range of colours to choose from.</li><li>- The background colours only update after closing and re-opening the application.</li><li>- When deleting a task, the application should double check to confirm if the user wants to delete the task.</li><li>- Displaying the task details and date is redundant since the list only displays tasks for today. Task details are sometimes too long to be displayed in the list.</li><li>- Change the time from a 24-hour system to a 12-hour system with AM/PM.</li><li>- The user wants to be able to edit tasks from the View Tasks page.</li></ul>
24/11/2014	<ul style="list-style-type: none"><li>- The Refresh button is a nice addition.</li><li>- The TOAST messages need to be more accurate. For example, trying to add a task while the task name field is empty should state so.</li><li>- A TOAST message should be added on the Settings page to let users know to refresh the main page after selecting a colour.</li><li>- Tasks can be created for the past. Participants are not sure if that is intended. It does not really make sense with the functionality of the application since the alarm will never occur if it is set for the past.</li></ul>
27/11/2014	<ul style="list-style-type: none"><li>- A task is removed from the list when the notification is displayed. Participants want to have the power to delete tasks manually and be able to look at tasks for previous days.</li><li>- Buttons look smoother. The new icon on the Create Task button is intuitive. Font is neat and organized.</li><li>- When adding a task, users cannot set a past date, however, they can set a past date when updating a task.</li><li>- Overall design layout is simple and easy to use.</li><li>- A bonus feature could allow the user to share a task with people in the phone's contacts list.</li></ul>

#### **4.1.1 – Interpretation & Discussion**

The participants were very helpful in voicing an outside opinion. They offered constructive criticism over a period of 2 weeks which allowed me to modify my application for the better. One of the most common themes brought up was the ability to view the tasks completed for previous days. Participants adored the idea of permanently storing their daily tasks on a light-weight, practical application, and this was no surprise. In today's technological society, the mobile phone has become essential to our everyday lifestyle. It is a vital instrument that keeps us informed and connected to the world. Another recurring comment was the option to set a username and colour to the application was a great addition. I underestimated the importance of this feature as I had initially added it as a small bonus. Participants acknowledged that customizing the application gave a sense of belonging.

I was pleased with the overall acceptance of the application by the users, as no major problems were encountered. Some of the issues encountered by a couple of the participants were insignificant compared to the purpose and goal I had in mind for my app. For example, one participant was displeased that there were no ways to connect with people over social media, such as posting a task on Twitter or Facebook. Another miniscule issue that was raised was the inability to add a personal ring tone for the alarm. These features may be something to add in the future.

## 4.2 – Quantitative Data Analysis

### 4.2.1 – Perception of the System

The following table displays the recorded answers given for question 1 of the questionnaire in the lab session.

**TABLE 2:** Results of question 1 from the questionnaire with a calculated mean, median, variance, and total score.

<b>Participant #</b>	<b>Font</b>	<b>Size</b>	<b>Colour</b>	<b>Overall Layout</b>	<b>Total Score</b>
1	5	4	5	4	18
2	5	3	4	4	16
3	4	5	4	4	17
4	5	5	4	3	17
5	5	4	3	3	15
<b>Mean</b>	4.8	4.2	4.0	3.6	16.6
<b>Median</b>	5	4	4	4	17
<b>Variance</b>	0.2	0.7	0.5	0.3	1.3

#### 4.2.2 – Usage of the System

The following table displays the recorded answers given for question 2 of the questionnaire in the lab session.

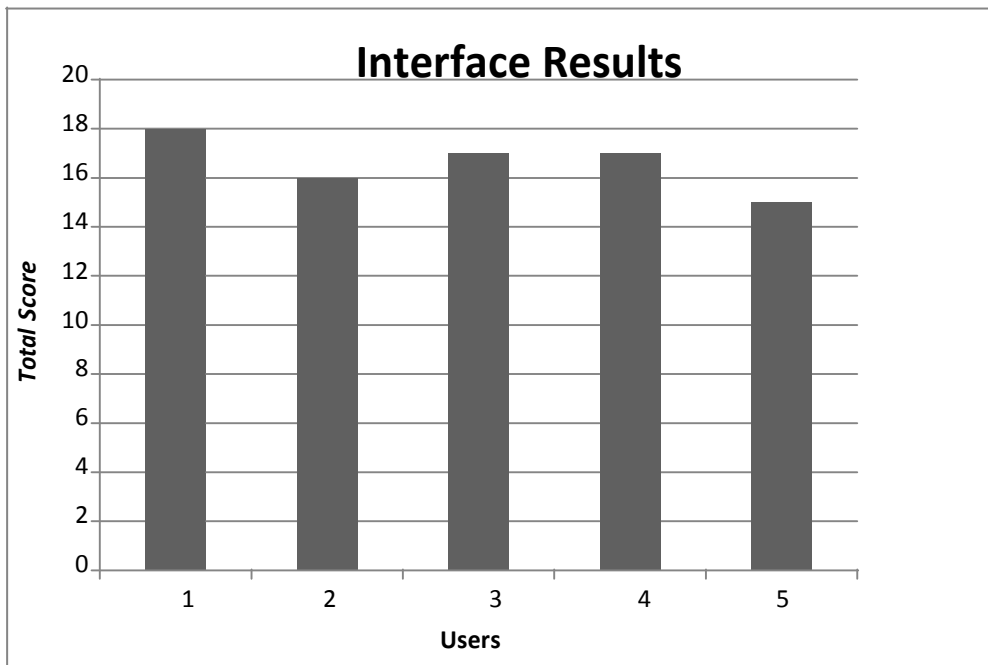
**TABLE 3:** Results of question 2 from the questionnaire with a calculated mean, median, variance and total score.

Participant #	Task 1	Task 2	Task 3	Task 4	Task 5	Total Score
1	5.0	5	5	4	5	24
2	5	5	4	4	5	23
3	5	5	4	4	5	23
4	5	5	5	4	5	25
5	4	5	4	5	5	23
<b>Mean</b>	4.8	5	4.4	4.2	5	23.6
<b>Median</b>	5	5	4	4	5	23
<b>Variance</b>	0.2	0	0.3	0.2	0	0.8

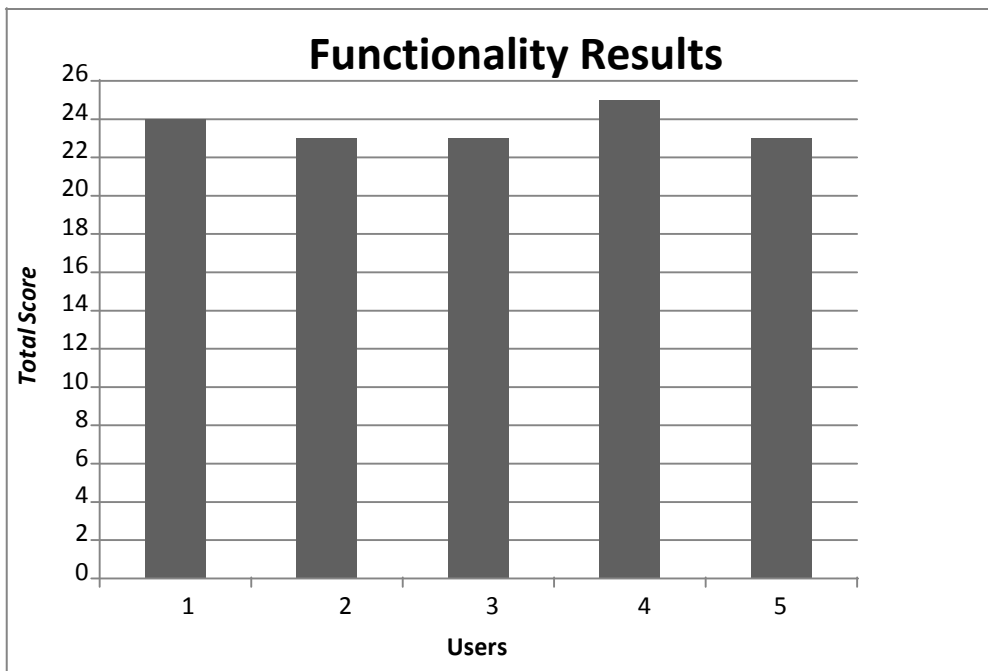
#### 4.2.3 – Interpretation & Discussion

The interface received an overall average score of 16.6/20, a median of 17 and a variance of 1.3. The quantitative results indicate that the interface of my application was fairly visually appealing, as illustrated in figure 12. I expected that participants would appreciate the design of Life Planner because of the small and simple things that I took into consideration, such as keeping the number of widgets to a minimum. The functional usage received an overall average score of 23.6/25, a median of 23 and a variance of 0.8. These results indicate that the general sample of participants were able to successfully add a task and navigate through the other features, as shown in figure 13. Although the results are generally positive in terms of both perception and usage, I expected higher scores for the user interface. I think the reason for why it was not higher is that people have grown accustomed to mainstream applications that are

developed by large companies, who would typically have abundant resources at their disposal. I also expected the functionality results to be higher, but this is most likely because I already have an idea and set of steps to take prior to launching the application, which gives me an advantage over participants who see this application for the very first time.



**FIGURE 12:** A visual display of the total scores from Table 2.



**FIGURE 13:** A visual display of the total scores from Table 3.

## 4.3 – Overall Assessment

### 4.3.1 – Interpretation & Discussion

The results of the user study indicate that the users found Life Planner exceptional. They had an average score of 16.6/20 (83%) for the interface's design and 23.6/25 (94%) for the usage of the system. These results exceed the 50% minimum requirement that I had set for my usability metrics. Four out of five of the participants said they would use the application. I was pleased with the functionality results' variance being as small as it was because it shows that there was no major confusion during the usage of the system. This assures the goal I had in mind for my application has been met.

#### *Areas of success:*

- User able to successfully add and delete tasks
- A working tracking feature to find the scheduled tasks on a specific date
- Ability to add personal preference to the application via Settings
- Visual presentation of tasks on the main page
- A simple and clean interface

#### *Areas for improvement:*

- Allow users to add a personal ring tone
- Allow users to share tasks via Contacts list
- Allow users to post tasks on social media



## 5. Conclusion

### 5.1 – Project Results

I am pleased with the results of this project. All the features I had initially planned to be in my application were successfully implemented. I did not expect to have extra time for bonus features, but I was wrong. I was able to implement features that expanded on the underlying function of my application. For example, searching for a specific day and having a list of tasks displayed for that day. Other minor features were customizing the colours and assigning a username. I am happy with my decision of building a simple, intuitive user interface like the pre-installed system applications on Android. A famous quote regarding software engineering by Antoine de Saint-Exupery says it best, “Perfection (in design) is achieved not when there is nothing more to add, but when there is nothing more to take away” [Saint-Exupery, n.d.]. My goal for this project was not just to create any application, but a practical one for everyday usage. I am extremely satisfied with my decision of broadening the demographic I was targeting to more than just students. As of this point in time, I have a high school student, college student, a middle-aged employee and a parent using my application.

### 5.2 – Limitations

Having no prior knowledge of Android development, I went into this project with high expectations with respect to the design. The tools and language took a large amount of start-up time because I had to learn and research a wide range of tools. The more I learned, the more options I had to choose from. Near the end of the development of my project, I realized the difficulty in creating a unique and stylistic interface. I had thought that Android Eclipse would provide me with a wide range of tools for designing my buttons, text and overall layout. I now understand the role and importance of a graphic designer in a development team. Most of my application’s widgets have the default design supplied by the ADT plugin, giving it a bland taste. Another minor limitation is that my user study was conducted with only 5 participants, which is too small of a sample to be able to extract accurate and concrete conclusions.

Another limitation I had to deal with was my Android device being outdated. There were multiple APIs I wanted to make use of but couldn’t because of its incompatibility with my device, such as setting a minimum to the *Date Picker Dialog*. I spent hours into looking for alternative ways, not just for the *Date Picker*, but for getting my action bar to work. Having a more recent Android device would definitely have saved me time and given me a wider range of features to choose from.

### **5.3 – Future Improvements**

There will always be room for improvement with any application. Since my application already delivers in terms of its main objective, in the future I plan on expanding on the main functionality of the system. I will begin by purchasing a new Android device, preferably Android KitKat (version 4.4, API level 19). This will allow me to use methods from all API levels. I would also like to improve the design of my application by taking online graphic design courses. Ideally, I would create my own icon and visual graphics.

Another personal, but long-term goal of mine is to have my application compete with the best on the Google Playstore. To achieve this, I will need to advance my application into a multi-purpose-based system. I plan to integrate a shared grocery list feature that allows multiple users to access the same list through the internet. This feature would work well with my current application because I already have a system that can add, remove and edit items. For a grocery list, I would simply create an Object class of Food which would be substituted with tasks on the list.

Adding additional elements to a task can diversify the application for specific scenarios. For example, an element I plan on adding is the estimated time it will take to complete a task and how much time has been already spent on it. This would utilize a timer which the user will start before working on the task. This feature would come in handy for contractors that charge by the hour.

### **5.4 – Contributions to the Field**

Inherently, there were no contributions to the field of Computer Science as the intention behind this project was to be a learning experience for myself and for future students. The purpose of this project was to build a practical application, along with a thorough report that exemplifies my thought process throughout the development cycle. I strongly believe this report will be useful for anyone looking to build their first Android application.

### **5.5 – Concepts and Technologies Learned**

I learned many concepts and technologies throughout the development of this project. I went into this project with no experience in the Android development platform, XML files, or implementing a SQLite database within a project. I am now comfortable with developing applications on the Android platform. I have gained enough experience with SDK tools and ADT

plugin, as well as creating and adding external libraries to a project. Initially, the use of XML files to create the layout was confusing because of all the attributes included for each element, but I can now easily create XML files using both the graphical layout tool or manually filling in the attributes. Near the end of the project I began creating custom *Drawables* for distinguishing my user interface elements from the default elements. For example, adding custom icons on a button, or shifting the text of a button towards an icon. I discovered that the UI development took just as long, if not longer than the functional development phase. I learned that planning is a crucial aspect of developing a project from scratch. This was an invaluable concept I learned. It is important to set enough time for receiving feedback from the participants in the user study, making changes to the application, and deploying the updated version to the participants again for more testing. This process of testing could take longer than expected, so it is important to make leeway for these scenarios.

## References

- Android. “*Layouts*.” Retrieved on October 23, 2024 from:  
<http://developer.android.com/guide/topics/ui/declaring-layout.html>
- Android. “*Support Library*.” Retrieved on November 2, 2014 from:  
<http://developer.android.com/tools/support-library/index.html>
- Anon. “*HTML color codes and names*.” Retrieved on November 10, 2014  
from: <http://www.computerhope.com/htmcolor.htm>
- Anon. “*SQLite vs MySQL*.” Retrieved on November 2, 2014 from:  
<http://stackoverflow.com/questions/3630/sqlite-vs-mysql>
- Icons8. “*New icons*.” Retrieved on November 23, 2014 from:  
<http://icons8.com/web-app/category/all/Very-Basic>
- IDC. “*Worldwide Mobile Phone Tracker*.” Retrieved on December 3, 2014  
from: <http://www.idc.com/getdoc.jsp?containerId=prUS25037214>
- O’Shea, Chris. “*What is mobile marketing anyway?*” Retrieved on December 1, 2014  
from: <http://www.bdc.ca/EN/blog/Pages/mobile-marketing.aspx>
- Saint-Exupery, de Antoine. “*Quotation Details*.” Retrieved on December 3, 2014  
from: <http://www.quotationspage.com/quote/26979.html>
- Wikipedia. “*Android (operating system)*.” Retrieved on December 2, 2014  
from: [http://en.wikipedia.org/wiki/Android\\_%28operating\\_system%29](http://en.wikipedia.org/wiki/Android_%28operating_system%29)
- Wikipedia. “*Java (programming language)*.” Retrieved on December 3, 2014  
from: [http://en.wikipedia.org/wiki/Java\\_%28programming\\_language%29](http://en.wikipedia.org/wiki/Java_%28programming_language%29)
- Wikipedia. “*XML*.” Retrieved on December 3, 2014 from:  
<http://en.wikipedia.org/wiki/XML>