

Task 1

The method ReadText below performs the task of collecting strings of each line from the text file and adding them to an array and returning it.

```
public String[] ReadText()
{
    String[] result = new String[arrayValues];
    int i = 0;

    try {
        InputStream fs = getAssets().open("au_locations.txt");
        BufferedReader br = new BufferedReader(new InputStreamReader(fs));
        String l;
        while((l = br.readLine()) != null)
        {
            result[i] = l.toString();
            i++;
        }
        br.close();
        Log.e("no more lines", "left");
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return result;
}
```

The getData function receives the String Array and splits the values within the line into each variable for the Cities class to assign each city within the array their values.

```
void getData(String[] s)
{
    for(int i = 0; i < s.length; i++)
    {
        Cities[i] = new City();
        String currentLine = s[i];
        Log.d("recieving", currentLine);

        String[] entry = currentLine.split(",");

        Cities[i].suburb = entry[0];
        Cities[i].latitude = Double.parseDouble(entry[1]);
        Cities[i].longitude = Double.parseDouble(entry[2]);
        Cities[i].timeZone = TimeZone.getTimeZone(entry[3]);

        Log.d("suburb is", Cities[i].suburb);
        Log.d("lat is", Double.toString(Cities[i].latitude));
        Log.d("long is", Double.toString(Cities[i].longitude));
        Log.d("timezone is", Cities[i].timeZone.toString());
    }
}
```

This code below contains the way this application assigns all the options for the spinner by adding each suburb in the cities array that can be selected.

```
// Spinner element
Spinner spinner = (Spinner) findViewById(R.id.spinner);

// Spinner click listener
spinner.setOnItemSelectedListener((AdapterView.OnItemSelectedListener) this);

// Spinner Drop down elements
List<String> categories = new ArrayList<String>();
for(int i = 0; c.Cities[i].suburb; i++)
{
    categories.add(c.Cities[i].suburb);
}

// Creating adapter for spinner
ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, categories);

// Drop down layout style - list view with radio button
dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

// attaching data adapter to spinner
spinner.setAdapter(dataAdapter);
```

This code assigns the city selected by clicking on a spinner to the values displayed in the activity.

It does this by using the parameter of position in the spinner to select the appropriate value in the Cities array. If no spinner is selected it will default back to melbourne.

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    // On selecting a spinner item
    String item = parent.getItemAtPosition(position).toString();
    selected = c.Cities[position];
    refreshTime();
    Log.d("position", String.valueOf(position));
    // Showing selected spinner item
    Toast.makeText(parent.getContext(), "Selected: " + item, Toast.LENGTH_LONG).show();
}

public void onNothingSelected(AdapterView<?> arg0) {
    // TODO Auto-generated method stub
    selected = new City("Melbourne", -37.814, 144.96332, TimeZone.getTimeZone("Australia/Melbourne"));
}
```

The pictures below all show the functionality of the sunrise and sunset app.

I have added Walhalla and Bright to the list which are shown in the first 2 screen shots.

Once selected in the spinner the information displayed on the activity is automatically updated.



Task 2

(Sub Task A)

- i) As a photographer, I want to get the sun set times beforehand allowing time to set up so that I have sufficient opportunity to take pictures of the sun as it sets.
- ii) As a Traveller, I would like to generate a table that includes all weather forecasts and sunrise and sunset times for multiple remote locations and be able to email and print this information so I can visit places in good weather in daytime.
- iii) As a motorcyclist, I want to check the weather in my current location and other towns nearby so that if I go for a ride I can dress appropriately for the temperature and not ride in the rain.
- iv) As a camper, I want to use the mobile's GPS to find the sunrise and sunset times for my location and be able to SMS the times with an extra message to friends even with having low mobile signal so I can get the most out of the day and inform my other camping friends in the area.

(Sub Task B)

User stories are lightweight documentation which cut down information compared to scenarios and mainly focus on the Roles of the people who want to use the application, the features that they will use on the app and the objective of why they use the app.

User stories are highly structured scenarios which are easier to write and cover most of the possible use cases. User story normally uses a generic character to explain the features the user will access and their goal.

A scenario describes a real world example with a simple explanation of how and detailed why the user will use the app also including the goal and other details such as location and time. Scenarios have more detail and set the real world context better allowing to inform that the app should be usable under these specific circumstances. A scenario describes the user experience in terms of outcomes not how it will be achieved.

User stories are abstract and scenarios are potential instances of the user story playing out. User stories explain the functionality whereas scenarios describe the potential uses for the functionality.

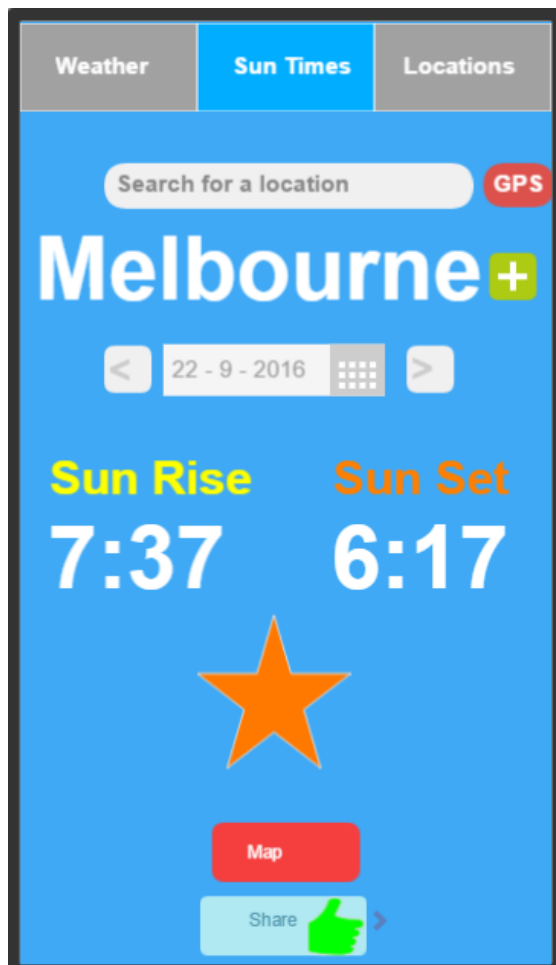
(Sub Task C)

Weather Screen



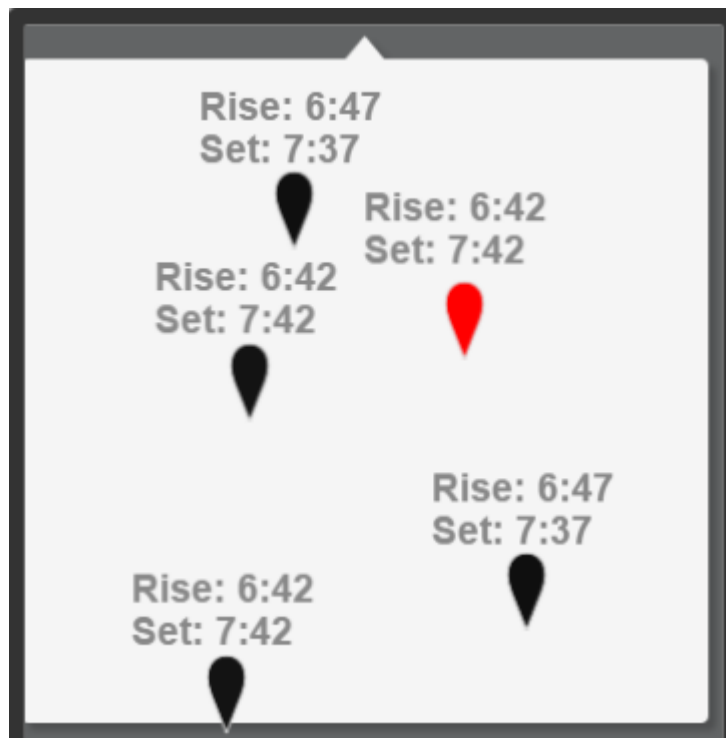
I designed the weather screen to be a simple design which is easy to read and understand what is going on. The grey text on the low temperature is designed to be less important than the white high text as more people focus on the high not the low. I include a graphic to show the weather conditions. The drop down calendar allows the user to select any date. The search for a location allows the user to find other locations to check weather from. The plus button next to the town/city name is to save the location to the locations tab. This screen includes a few of the features outlined. The weather page allows you to view the weather forecast for any date set. This page can share information through SMS or Email with the share button at the bottom of the page. It can detect the current location and set the weather report to display your current location with the GPS button.

Sun Times



I designed the sun times screen to be very simplistic and straight to the point on displaying information. The plus button next to the clear town/city name is to add the location to the location tab. The star displayed will inform the user if it is day or night. The text is all large for easy reading. This page displays the sun rise and set times for a specific location on a specific date. You can detect the current location by pressing the GPS button. To display a map of various locations sun times you can press the map button. The share button allows you to text or email the information to anyone.

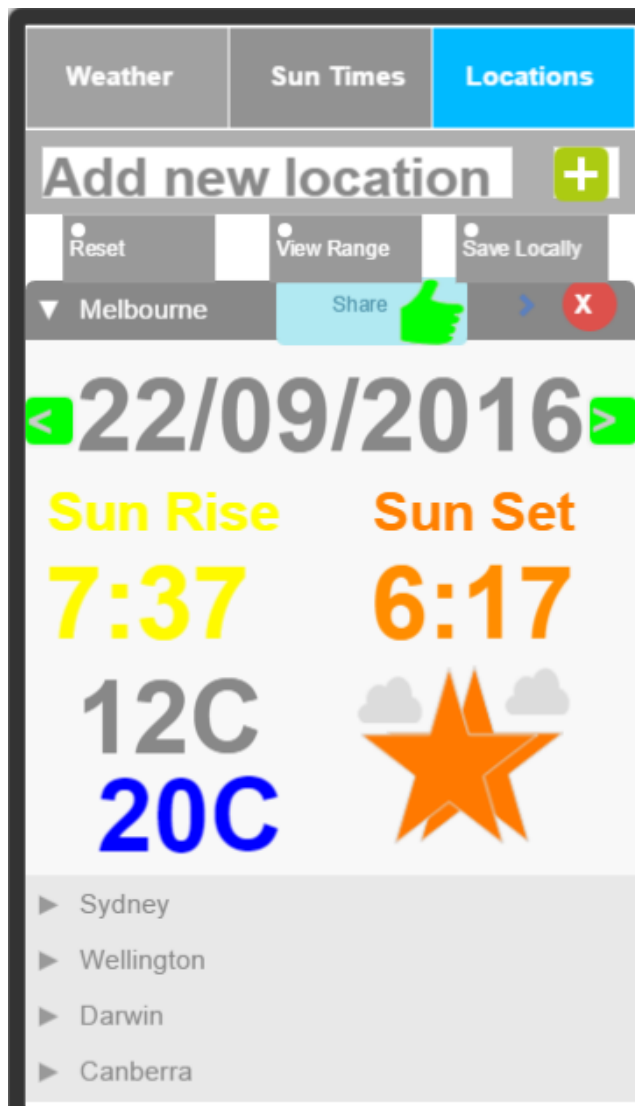
Map Screen



The Map Screen is designed to be extremely simple with not much information only what the user wants with black dots giving nearby towns/city sun times with the red dot indicating the current town. The screen will also display google maps view of the area.

This screen integrates with google maps displaying the town names, roads and other landmarks. The maps screen uses the current location on the sun times tab to find sun rise and set times in the nearby area on the map.

Simple Locations Screen



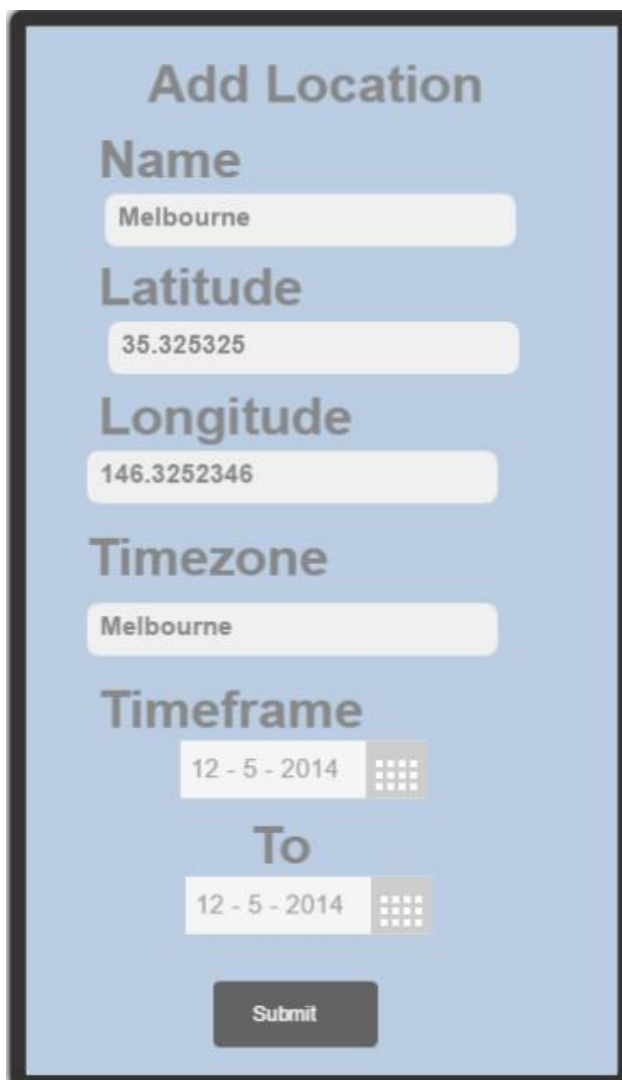
The Locations tab is designed to be very simplistic with clear and concise information display for each town/city that has been saved. You can share the information with the share button. You can remove cities that have been added with the red button. The reset button wipes all data to a clear slate. The save locally button allows the user to save weather and sun times to the device so they do not need a data connection. The view range button takes you to the locations range table screen which displays raw data in a table for a range. You can view sun times and weather forecast on this page for different locations saved.

Locations Range Table Screen

Weather	Sun Times	Locations
Add new location		+
Melbourne 22/09/2016		x
Rise	Set	High Low
7:42	6:52	17 7
Melbourne 23/09/2016		x
Rise	Set	High Low
7:42	6:52	17 7
Melbourne 24/09/2016		x
Rise	Set	High Low
7:42	6:52	35 7
Share		>

The locations range table screen is very basic with raw data with not much visuals to display day to day information on sun times and the weather forecast. This screen generates a table of sun times for a date range. You can also add new custom locations or pre set locations with the add new location text box and plus button.

Add Location Screen



The image shows a mobile app screen titled "Add Location". It features a light blue background with a dark border. The form contains the following fields and labels:

- Add Location** (Title)
- Name** (Label) with a text input field containing "Melbourne".
- Latitude** (Label) with a text input field containing "35.325325".
- Longitude** (Label) with a text input field containing "146.3252346".
- Timezone** (Label) with a text input field containing "Melbourne".
- Timeframe** (Label) with a date range selector showing "12 - 5 - 2014" and a calendar icon.
- To** (Label) with a date range selector showing "12 - 5 - 2014" and a calendar icon.
- Submit** (Button) at the bottom.

The add locations screen includes all the information you need to fill out to include a custom location to be able to access the sun times. This screen allows you add pre defined locations as you fill out the name category if the area exists in the system it will fill out all the other information for you. Also the form will allow you to add custom locations which you fill out the information and the phone will generate the sun time data depending on the time zones, longitude and latitude.

Sub Task C part d – indicate sequence of screens for the user scenarios

i)Brad will only need to view the Sun Times screen to check the sun times

ii)For Sachin to create the table of sun rise and set times he must go the Sun Times screen search for the locations, set the times he is visiting and then press the green add button to add it to the table. He must then move to the Locations screen and then press the View Range button to access the Locations range table screen, in which he can also add more custom locations with the Add locations screen for his remote places he is visiting and ranges with the Add new location and green button bar. He must then click the share button to email the table so he can print it.

iii)For Li to check sun rise times for Sydney She must Visit the Sun Times screen which gets her location automatically, the she must press the right arrow to move the date to the next day then she can check the sun rise time for the following day.

iv)Justin and Mary need to find the sun times for their remote location. They must be on the Sun Times screen and then press the gps button to get the current location's sun times. They must then press the right arrow next to the date to select tomorrows sun times and then they want to press the share button at the bottom to be sending the sun times intent to the phone's message activity so they can add the extra note to the message.