**User Guide**

The CalenderApp is an application that has a calendar in which the user can view the time from a year, month, and day perspectives. It also allows the used to add events/reminders to a list that the application keeps in memory. The user can remove events from the event at any point of time without removing the entire list of events too. In order to use the project, the user needs to download the files from the GitHub Repository named (3354---Team-Rocket). Place the .java files and the .xml in the AndroidStudio project files to have the complete project present. Build then run on the virtual machine of your choosing.

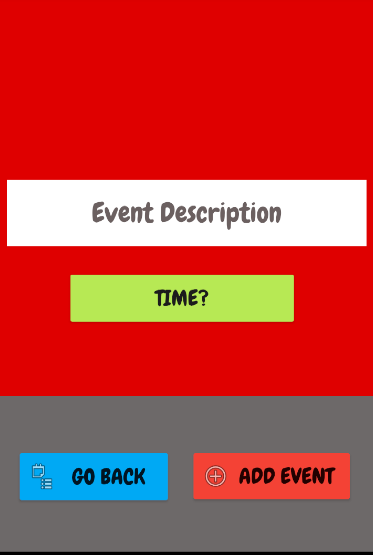
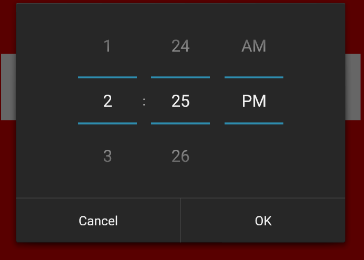
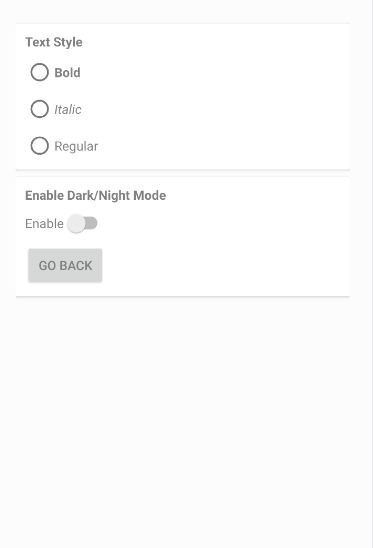
1. The application boots up in Month View. This acts as a central point of connection to the rest of the application’s components. It has buttons for both the settings menu, which allows the user to change the font, and the Year View.
2. Clicking on any of the days will take the user to the Event View (shown right), in which a user can add events and view already existing ones.
   1. The red button is used to add events
   2. The blue button takes the user back to Month View
3. Clicking on the red button moves the user further into Event View. The event can be named and assigned a specific time in which it triggers. The interface is shown below

Figure 1 Setting the time for each individual event

Figure 2 The Event View allows the users to input the name of the reminder and assign a time unique to it

1. The Settings View allows the user to change a few novel things such as the font properties. Italicizing the font or making it bold can be applied to the text. It can also apply dark mode to the user interface for an alternative look.

**Location of the Test Cases**

**java.com.example.calenderApp (android test)**

5 test cases split across 3 locations (classes)

1. eventViewTest.java

2. myDBAdapterTest.java

3. TimePickerFragmentTest.java

**The Design Pattern used and its Location**

Design pattern used and the corresponding code location

Factory Pattern: The theme of the app changes based on the user picked settings.

if(AppCompatDelegate.getDefaultNightMode() == AppCompatDelegate.MODE\_NIGHT\_YES)

{

setTheme(R.style.DarkTheme);

}

else setTheme(R.style.AppTheme);

/////////////////////////////////////////////////////////////////////////////////////////

if(textStyle.equals("Bold")) {

eventText.setTypeface(eventText.getTypeface(), Typeface.BOLD);

}

if(textStyle.equals("Regular")) {

eventText.setTypeface(eventText.getTypeface(),Typeface.NORMAL);

}

if(textStyle.equals("Italic")) {

eventText.setTypeface(eventText.getTypeface(),Typeface.ITALIC);

}

Creational Pattern: Creating an object that helps access the database.

helper = new myDBAdapter(this); //new database helper object

Behavioral Pattern- Each Android Activity contains this onCreate methods, however it is used in it own way.

protected void onCreate(Bundle savedInstanceState) {.....................}

Visitor Pattern: The class is implement the onClickListener, but will be defined based on its own use within the class. So it does not have to be the same as in another class.

//dayView class used to view all the events for a selected Date

public class dayView extends AppCompatActivity implements View.OnClickListener {.........}

@Override

public void onClick(View view) {

Intent i = new Intent(dayView.this, eventView.class); //going from dayView to eventView

startActivity(i);

}

Singleton Pattern: The Calendar is a class and only one instance of that class is being created.

calender = (CalendarView) findViewById(R.id.calender);