APHRODITE

Kainotomia

Aphrodite

Daniel Moore (n01354875)
Alyssa Gomez (n01042777)
Jose Antonio Teodoro (n01384776)
Ryan Black (n01305403)

Table of Contents

Brief Description of Project

The Aphrodite App is an interface between the user and the Aphrodite Smart Mirror. The user will be able to create an account in the app that will store the created layouts. Voice commands are also available to the user to control the mirror, as well as an LED colour switcher for the LEDs that are (will be) built into the smart mirror. Each user creates an account on first start and all this information will be stored in the Firebase Realtime database, with the user account information being linked with Firebase Auth.

Effort Table

Name	ID	Signature	Effort
Alyssa Gomez	N01042777	AG	100%
Daniel Moore	n01354875	DM	100%
Jose Antonio Teodoro	n01384776	JT	100%
Ryan Black	n01305403	RB	100%

GitHub Repo Link

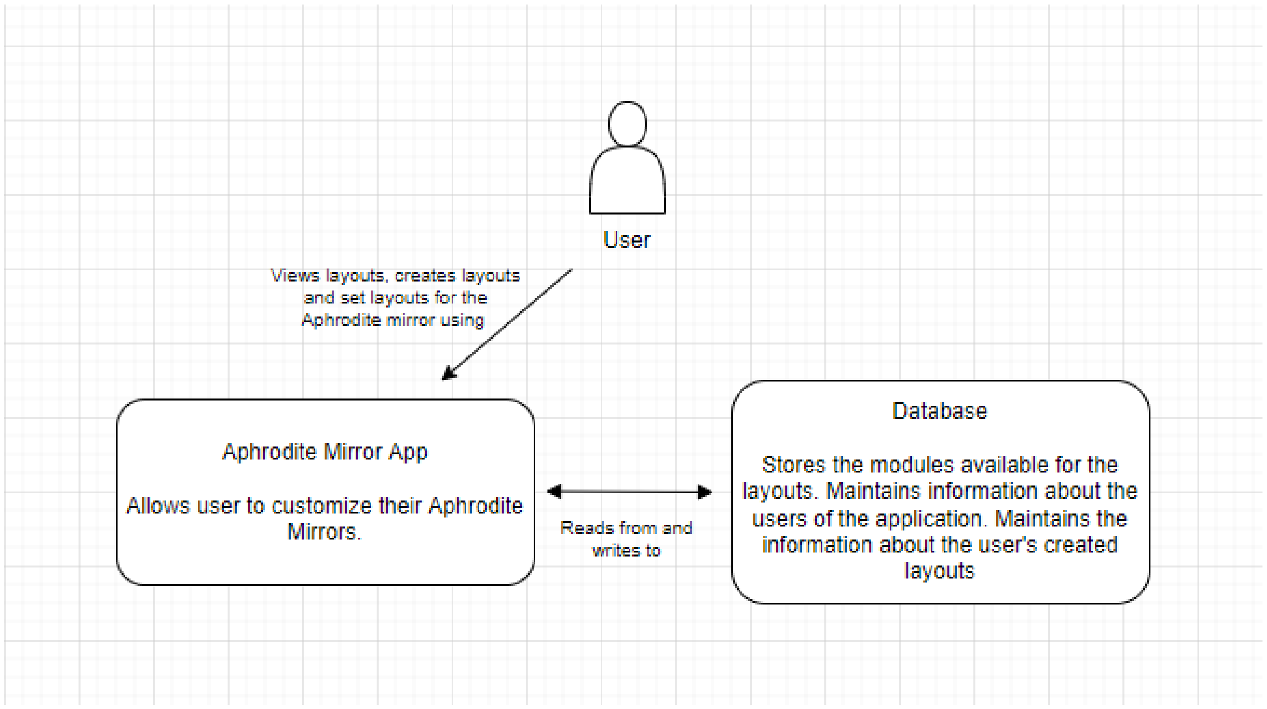
https://github.com/DanielMoore4875/Aphrodite

Sprint Goals

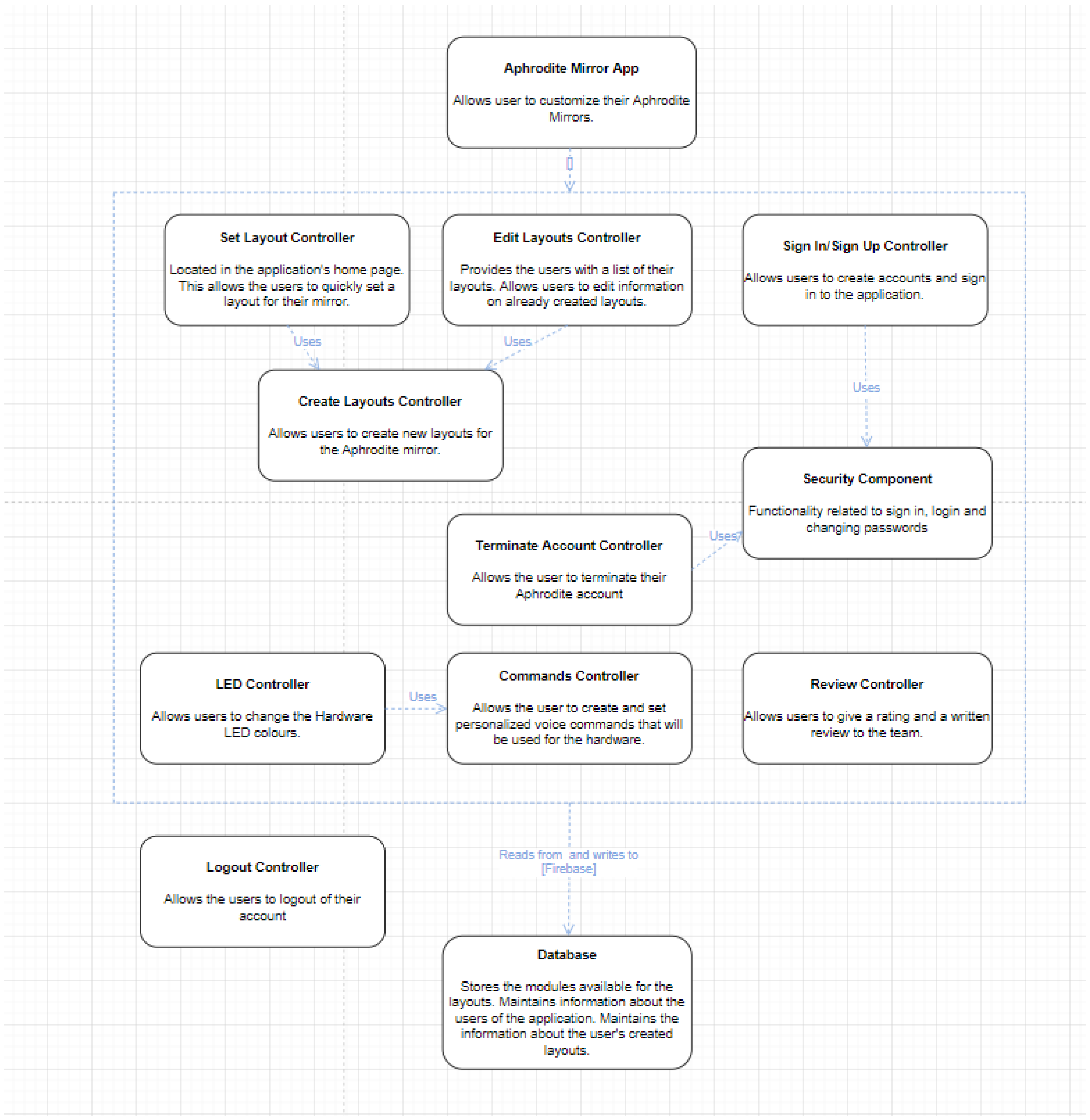
- When user logs in, all layouts that they have created are populated on the home page and the layouts page
- Edit layout functionality for the layouts page
- Test Cases
- Split login and registration into two pages with password criteria implemented
- Finalize the app

Full timeline:
https://trello.com/b/Bx0Fipkw

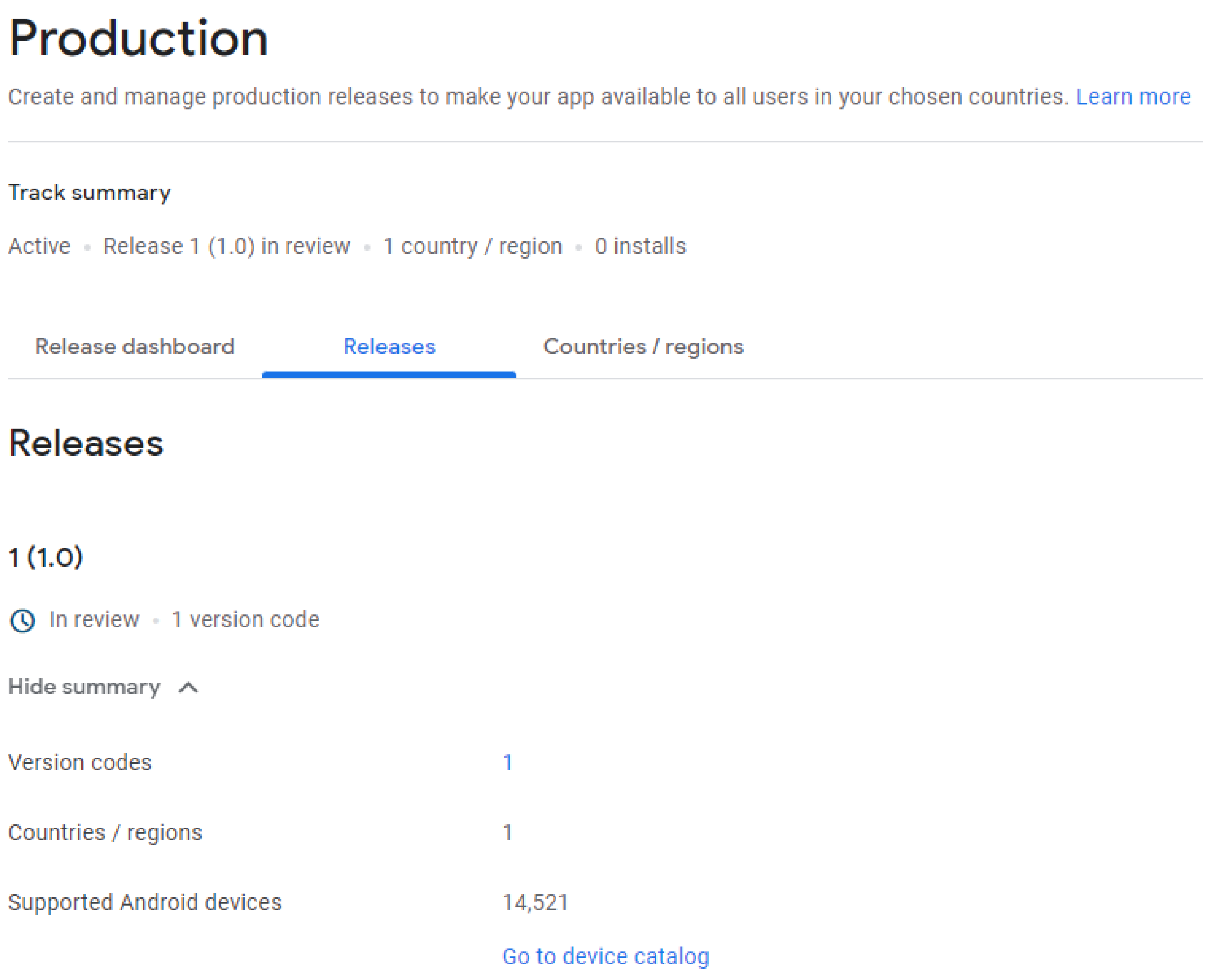
C4 Model Container Diagram


User
Views layouts, creates layouts and set layouts for the Aphrodite mirror using
Aphrodite Mirror App
Allows user to customize their Aphrodite Mirrors.
Reads from and writes to
Database
Stores the modules available for the layouts. Maintains information about the users of the application. Maintains the information about the user's created layouts

C4 Model Component Diagram


Aphrodite Mirror App
Allows user to customize their Aphrodite Mirrors.
Set Layout Controller
Located in the application's home page. This allows the users to quickly set a layout for their mirror.
Edit Layouts Controller
Provides the users with a list of their layouts. Allows users to edit information on already created layouts.
Sign In/Sign Up Controller
Allows users to create accounts and sign in to the application.
Uses
Uses
Uses
Create Layouts Controller
Allows users to create new layouts for the Aphrodite mirror.
Security Component
Functionality related to sign in, login and changing passwords
Terminate Account Controller
Allows the user to terminate their Aphrodite account
Uses
LED Controller
Allows users to change the Hardware LED colours.
Uses
Commands Controller
Allows the user to create and set personalized voice commands that will be used for the hardware.
Review Controller
Allows users to give a rating and a written review to the team.
Logout Controller
Allows the users to logout of their account
Reads from and writes to [Firebase]
Database
Stores the modules available for the layouts. Maintains information about the users of the application. Maintains the information about the user's created layouts.

Project on Google Play



Production

Create and manage production releases to make your app available to all users in your chosen countries. Learn more

Track summary

Active • Release 1 (1.0) in review • 1 country / region • 0 installs

Releases

1 (1.0)

In review • 1 version code

Version codes	1
Countries / regions	1
Supported Android devices	14,521
	Go to device catalog

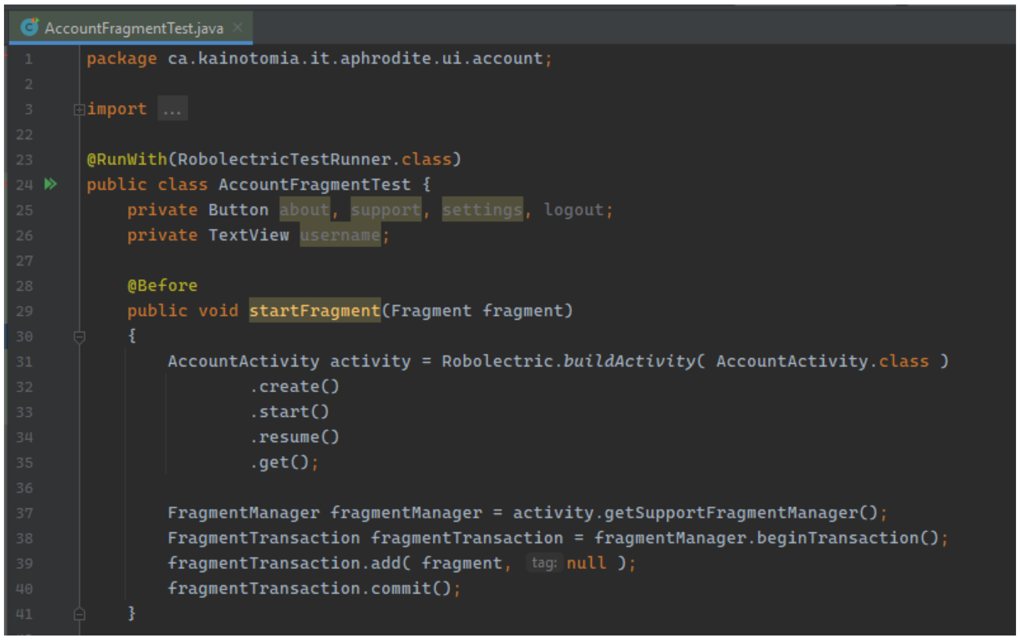
Test Cases

<u>Snippets of each class are shown below. Some classes may show error text but that is just a bug in the git tracking of the file, there are no errors and it runs fine.**</u>**

AccountFragmentTest

The account fragment test runs and checks if all the buttons and text widgets work in the account fragment. This includes all the transversal buttons going from fragment to fragment and also checks if AccountFragment runs in the first place. In addition, this class also checks string input of the user and if the username value will come through the fragment

Startfragment // runs fragment through method as default
shouldnotBeNull // checks if fragment runs
buttonTest // checks if buttons are active
usernameTest // checks if user name is present within the account fragment



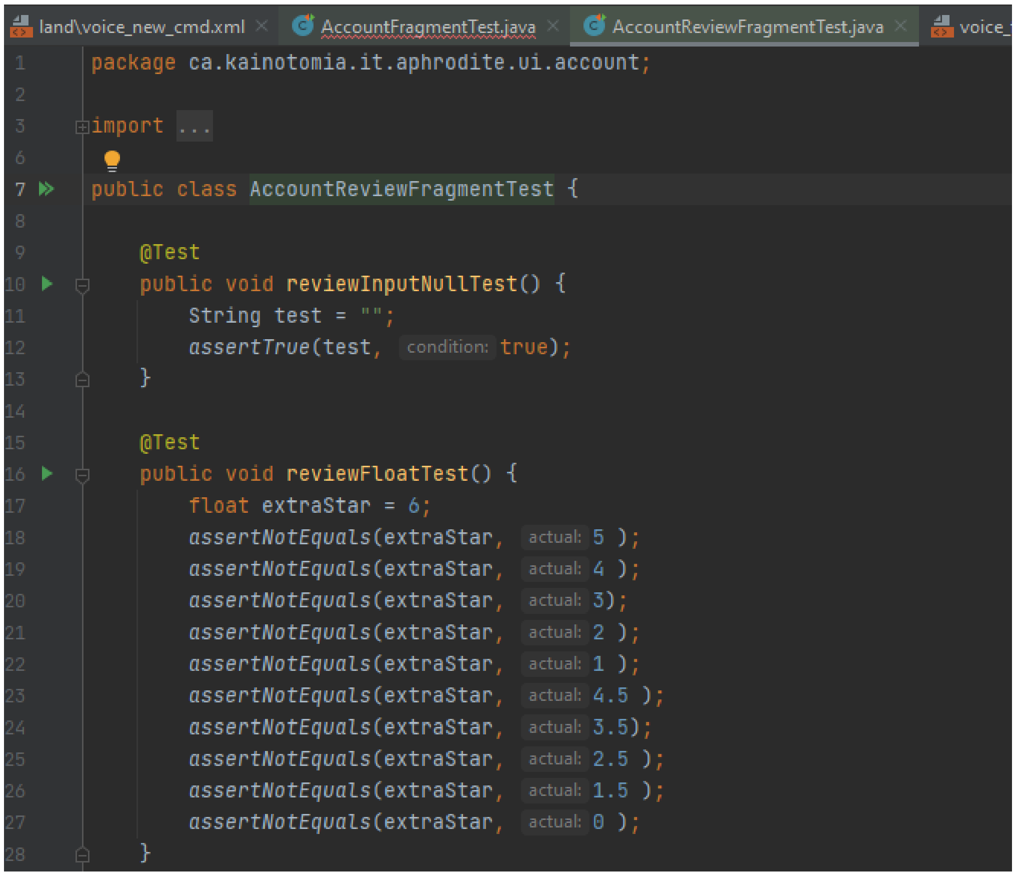
AccountFragmentTest.java

```
package ca.kainotomia.it.aphrodite.ui.account;

import ...

@RunWith(RobolectricTestRunner.class)
public class AccountFragmentTest {
    private Button about, support, settings, logout;
    private TextView username;

    @Before
    public void startFragment(Fragment fragment)
    {
        AccountActivity activity = Robolectric.buildActivity( AccountActivity.class )
                .create()
                .start()
                .resume()
                .get();

        FragmentManager fragmentManager = activity.getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.add( fragment, tag: null );
        fragmentTransaction.commit();
    }
```

AccountReviewFragmentTest

The AccountReviewFragmentTest tests if the fragment will run even if there is an empty string, or specifically no review text from the user. in addition it also checks what happens if the float values for the ratingBar will work if it exceeds or stays within range of the 5 star float maximum.

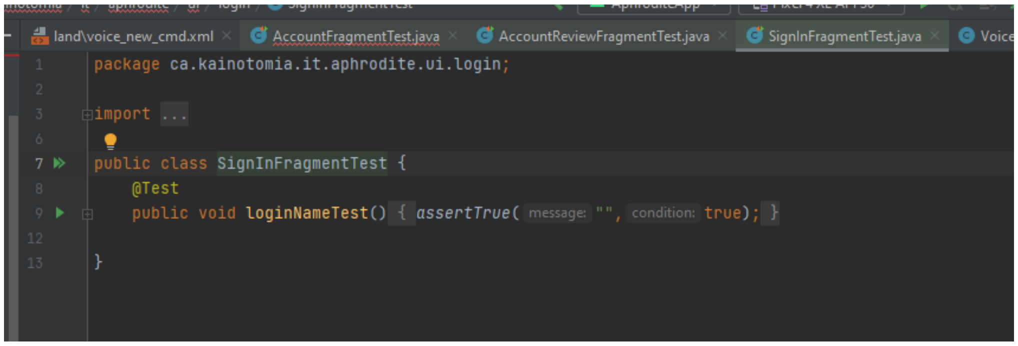
reviewInputNullTest // checks if there is any strong present, will return true regardless
reviewFloatTest // checks if float value exceeds the value of 5.0, it will return false
reviewInputFloatTest // checks if the float value is less than or equal to 5, if it is, it will return true



```java
package ca.kainotomia.it.aphrodite.ui.account;

import ...

public class AccountReviewFragmentTest {

    @Test
    public void reviewInputNullTest() {
        String test = "";
        assertTrue(test,  condition: true);
    }

    @Test
    public void reviewFloatTest() {
        float extraStar = 6;
        assertNotEquals(extraStar,  actual: 5 );
        assertNotEquals(extraStar,  actual: 4 );
        assertNotEquals(extraStar,  actual: 3);
        assertNotEquals(extraStar,  actual: 2 );
        assertNotEquals(extraStar,  actual: 1 );
        assertNotEquals(extraStar,  actual: 4.5 );
        assertNotEquals(extraStar,  actual: 3.5);
        assertNotEquals(extraStar,  actual: 2.5 );
        assertNotEquals(extraStar,  actual: 1.5 );
        assertNotEquals(extraStar,  actual: 0 );
    }
```

SignInFragmentTest

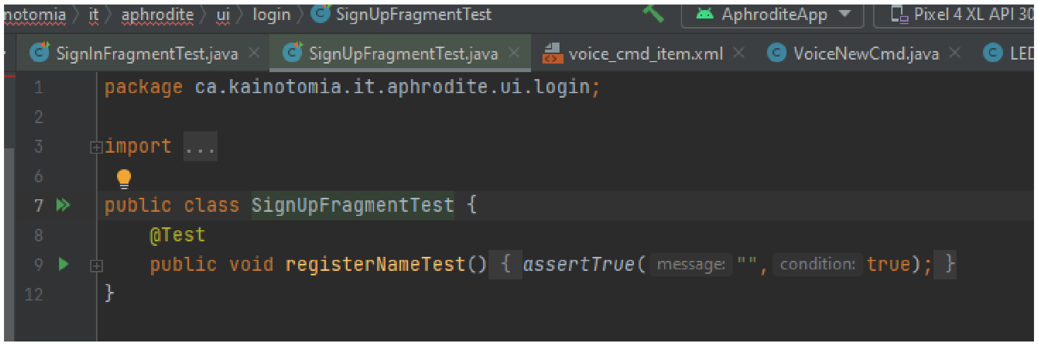
Signup/SigninFragmentTest checks if there is nothing, then it will return false, meaning the app won't continue unless all the text boxes are filled out.



```java
package ca.kainotomia.it.aphrodite.ui.login;

import ...

public class SignInFragmentTest {
    @Test
    public void loginNameTest() { assertTrue( message: "", condition: true); }

}
```

SignUpFragmentTest

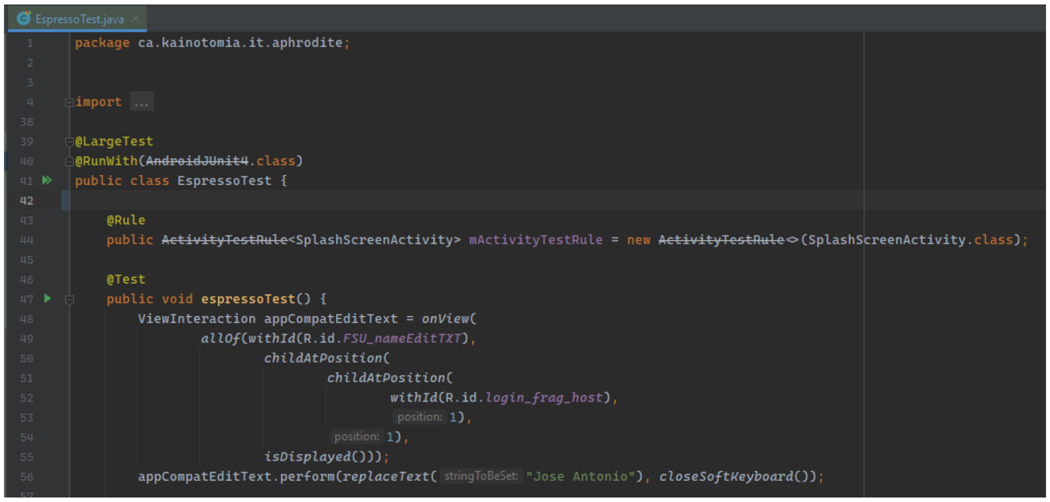
Signup/SigninFragmentTest checks if there is nothing, then it will return false, meaning the app won't continue unless all the text boxes are filled out.



```java
package ca.kainotomia.it.aphrodite.ui.login;

import ...

public class SignUpFragmentTest {
    @Test
    public void registerNameTest() { assertTrue( message: "", condition: true); }
}
```

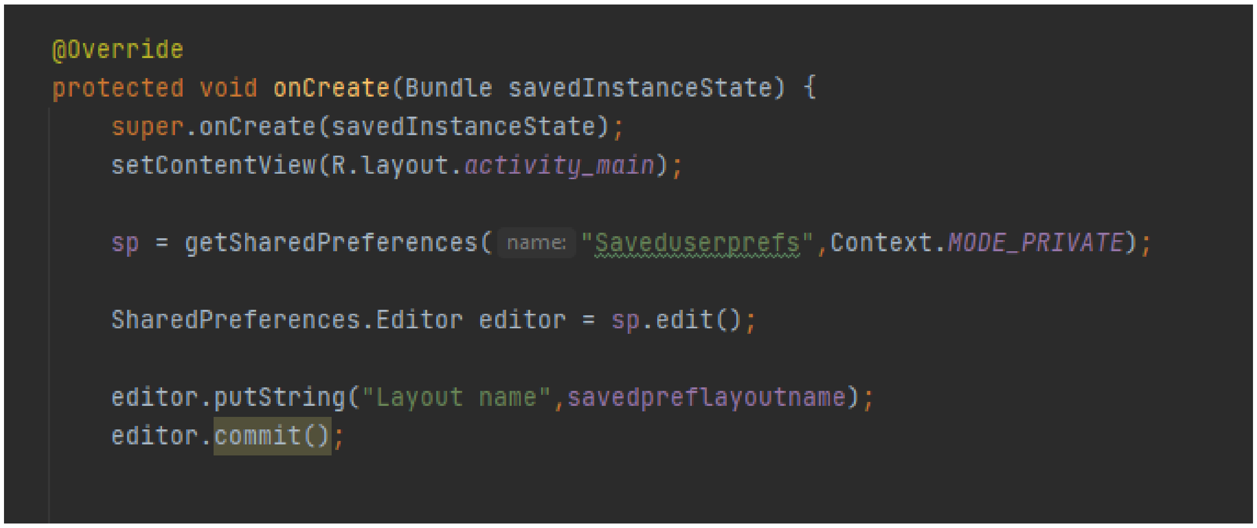
EspressoTest

EspressoTest uses Espresso to check ui input within the app, all separated by multiple screenshot assertions testing login and signup pages if the user exists or doesn't exist in the database, checks if the navigation bar works, and finally checks if the add layout button and add layout fragment runs.

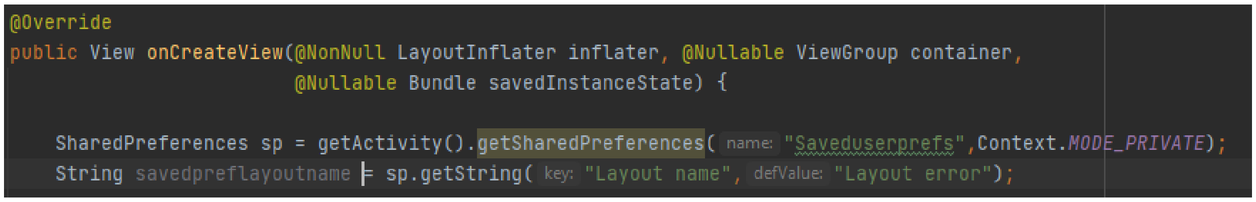


```java
package ca.kainotomia.it.aphrodite;


import ...

@LargeTest
@RunWith(AndroidJUnit4.class)
public class EspressoTest {

    @Rule
    public ActivityTestRule<SplashScreenActivity> mActivityTestRule = new ActivityTestRule<>(SplashScreenActivity.class);

    @Test
    public void espressoTest() {
        ViewInteraction appCompatEditText = onView(
                allOf(withId(R.id.FSU_nameEditTXT),
                        childAtPosition(
                                childAtPosition(
                                        withId(R.id.login_frag_host),
                                        position: 1),
                                position: 1),
                        isDisplayed()));
        appCompatEditText.perform(replaceText( stringToBeSet: "Jose Antonio"), closeSoftKeyboard());
```

Offline Features



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    sp = getSharedPreferences( name: "Saveduserprefs",Context.MODE_PRIVATE);

    SharedPreferences.Editor editor = sp.edit();

    editor.putString("Layout name",savedpreflayoutname);
    editor.commit();
```



```
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {

    SharedPreferences sp = getActivity().getSharedPreferences( name: "Saveduserprefs",Context.MODE_PRIVATE);
    String savedpreflayoutname = sp.getString( key: "Layout name", defValue: "Layout error");
```

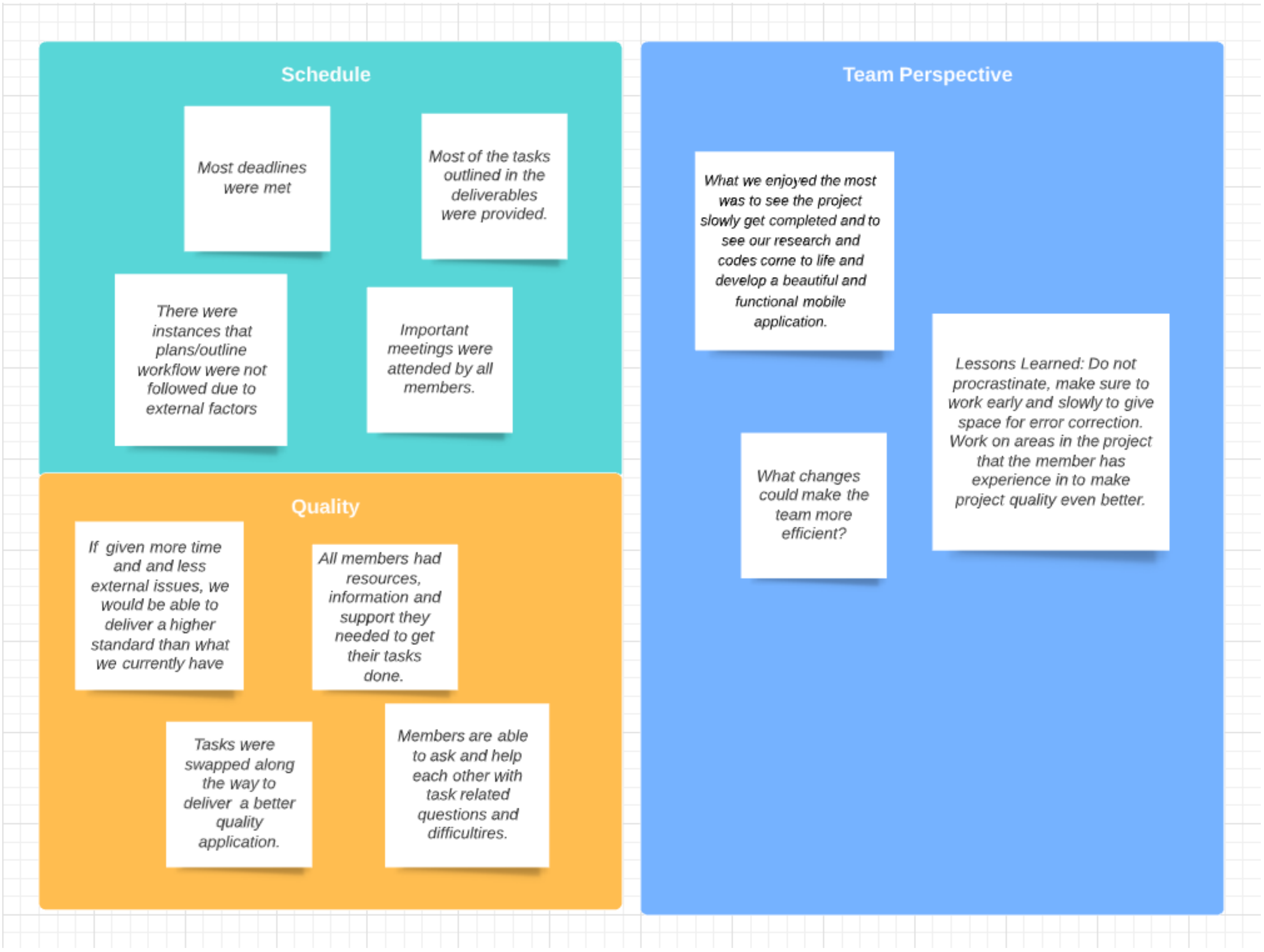
Created the sharedpreferences in order to receive the names of the users layout pages, and save them within sharedpreferences sp. When the user ends up creating and pressing the save button on createlayoutfragment page it will automatically store the name and keep it there for when the device is in offline mode.

Complete Scrum Dashboard

All Stories and tasks on Trello Board

https://trello.com/b/Bx0Fipkw

Post-Mortem Project Review Meeting



Schedule

Most deadlines were met

Most of the tasks outlined in the deliverables were provided.

There were instances that plans/outline workflow were not followed due to external factors

Important meetings were attended by all members.

Quality

If given more time and and less external issues, we would be able to deliver a higher standard than what we currently have

All members had resources, information and support they needed to get their tasks done.

Tasks were swapped along the way to deliver a better quality application.

Members are able to ask and help each other with task related questions and difficultires.

Team Perspective

What we enjoyed the most was to see the project slowly get completed and to see our research and codes come to life and develop a beautiful and functional mobile application.

Lessons Learned: Do not procrastinate, make sure to work early and slowly to give space for error correction. Work on areas in the project that the member has experience in to make project quality even better.

What changes could make the team more efficient?

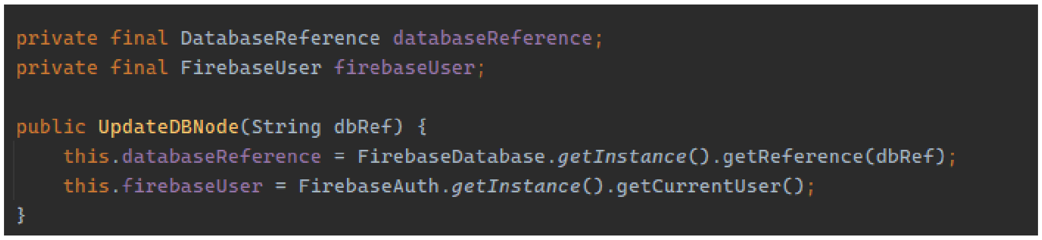
How We Addressed Technical Debt

In the introduction of the app, we wanted to test that the login was possible and able to be done easily. We used the built-in Firebase UI login builders that handled all the login information and combined the registration and login into one page. Once we knew more about how the login functionality would work and the requirements needed, we refactored the login process to be two screens, registration and login. Both the screens meet the password requirements and allow the user to login with ease. If the user closes the app without logging out, they will be auto-logged in when they start the app again for easy and fast access to their layouts.

Two Areas of Refactoring

Using the UpdateDBNode class for all firebase related queries that call on similar data

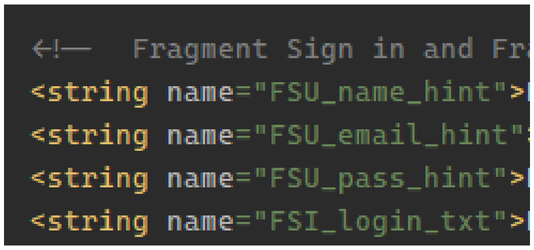
- This was done to make it easier to reference a node and know exactly what is going to be done when it is referenced.



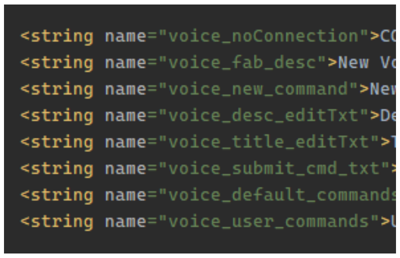
```java
private final DatabaseReference databaseReference;
private final FirebaseUser firebaseUser;

public UpdateDBNode(String dbRef) {
    this.databaseReference = FirebaseDatabase.getInstance().getReference(dbRef);
    this.firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
}
```

Renaming strings in strings.xml to reflect what page they exist on

- This was done for easy access to strings when their reference is needed in the code.



```xml
<!-- Fragment Sign in and Fr
<string name="FSU_name_hint">
<string name="FSU_email_hint"
<string name="FSU_pass_hint">
<string name="FSI_login_txt">
```

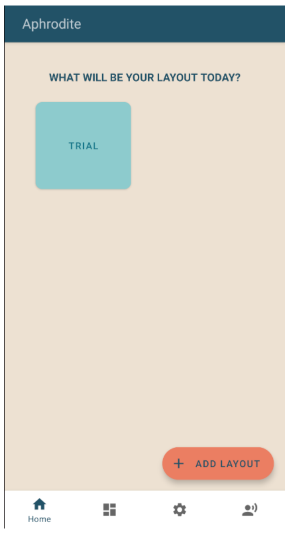


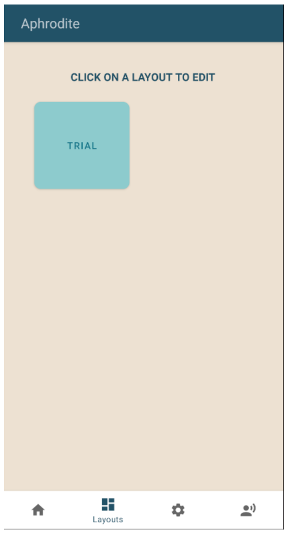
```xml
<string name="voice_noConnection">C
<string name="voice_fab_desc">New V
<string name="voice_new_command">Ne
<string name="voice_desc_editTxt">D
<string name="voice_title_editTxt">
<string name="voice_submit_cmd_txt"
<string name="voice_default_command
<string name="voice_user_commands">
```

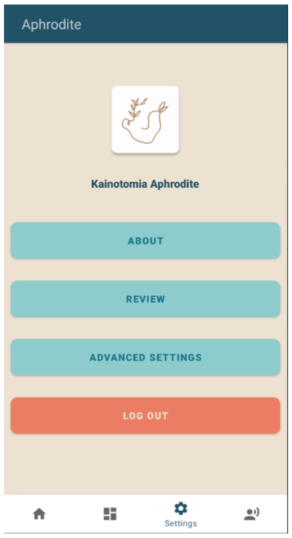
Suggestions to the instructor

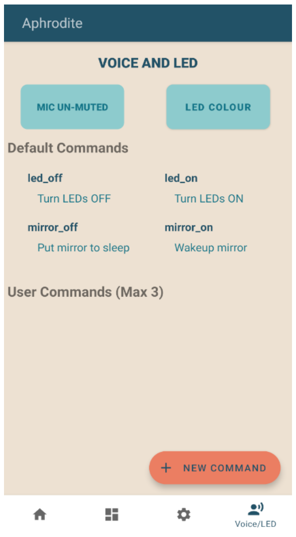
- Clearer information on the deliverables.
- Less tasks on each deliverable.
- Have less commits requirement on the final deliverable as some teams may be close to finishing or are done with their application development. Thus causing them to not have enough commits by the end of the deliverable.

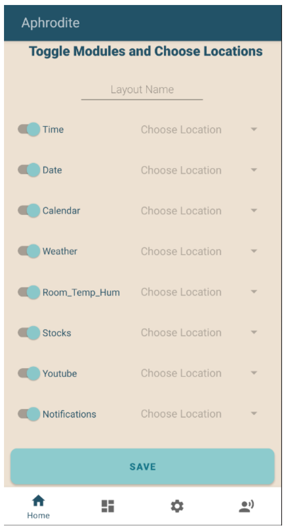
Page Screenshots

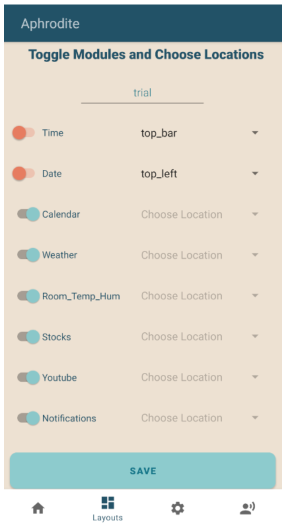


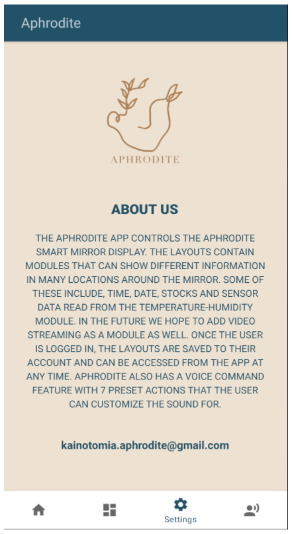








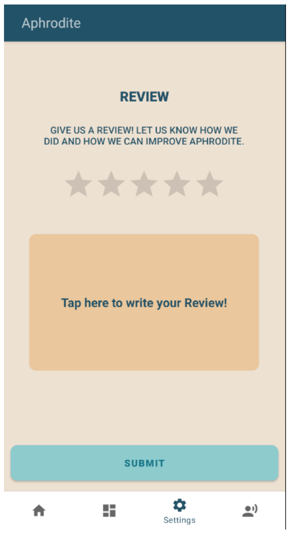


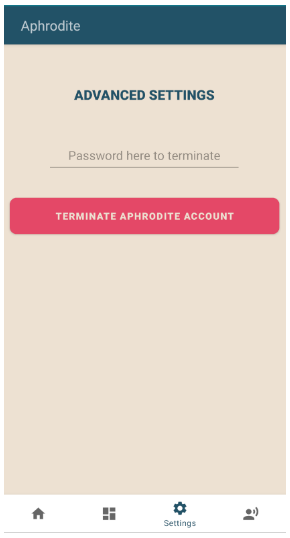


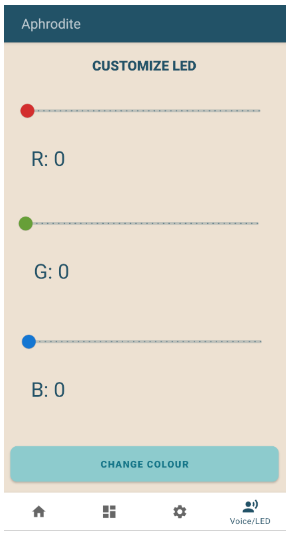
ABOUT US

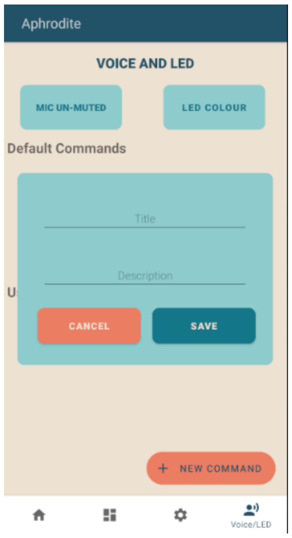
THE APHRODITE APP CONTROLS THE APHRODITE SMART MIRROR DISPLAY. THE LAYOUTS CONTAIN MODULES THAT CAN SHOW DIFFERENT INFORMATION IN MANY LOCATIONS AROUND THE MIRROR. SOME OF THESE INCLUDE, TIME, DATE, STOCKS AND SENSOR DATA READ FROM THE TEMPERATURE-HUMIDITY MODULE. IN THE FUTURE WE HOPE TO ADD VIDEO STREAMING AS A MODULE AS WELL. ONCE THE USER IS LOGGED IN, THE LAYOUTS ARE SAVED TO THEIR ACCOUNT AND CAN BE ACCESSED FROM THE APP AT ANY TIME. APHRODITE ALSO HAS A VOICE COMMAND FEATURE WITH 7 PRESET ACTIONS THAT THE USER CAN CUSTOMIZE THE SOUND FOR.

kainotomia.aphrodite@gmail.com


Aphrodite
REVIEW
GIVE US A REVIEW! LET US KNOW HOW WE DID AND HOW WE CAN IMPROVE APHRODITE.
Tap here to write your Review!
SUBMIT
Settings

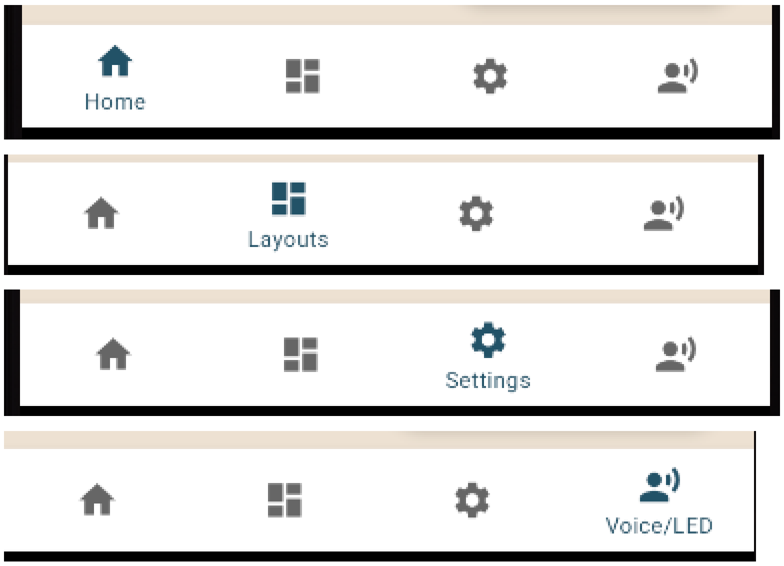




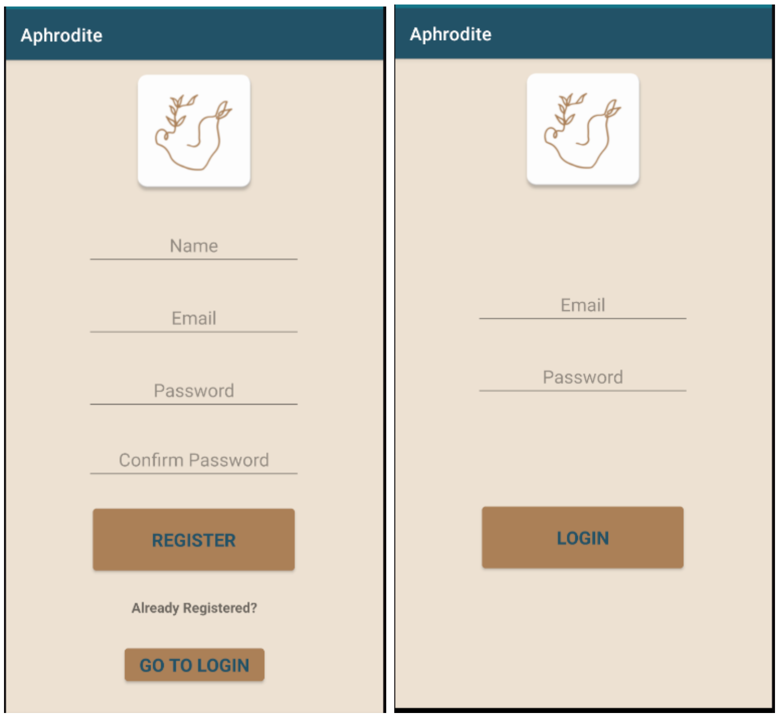


Items from other Deliverables

Bottom Navigation menu



Split Login and Registration Pages


Aphrodite
Name
Email
Password
Confirm Password
REGISTER
Already Registered?
GO TO LOGIN
Aphrodite
Email
Password
LOGIN