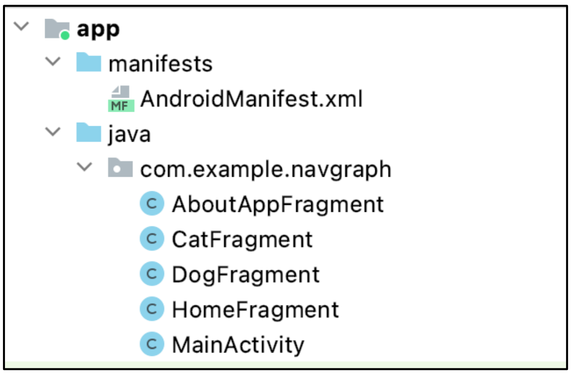
WIA2007 Mobile Application Development
Semester 1, Session 2022/2023
Practical 7 (Navigation)

Task 1: Understanding the Animal Lover App and Source File Preparation

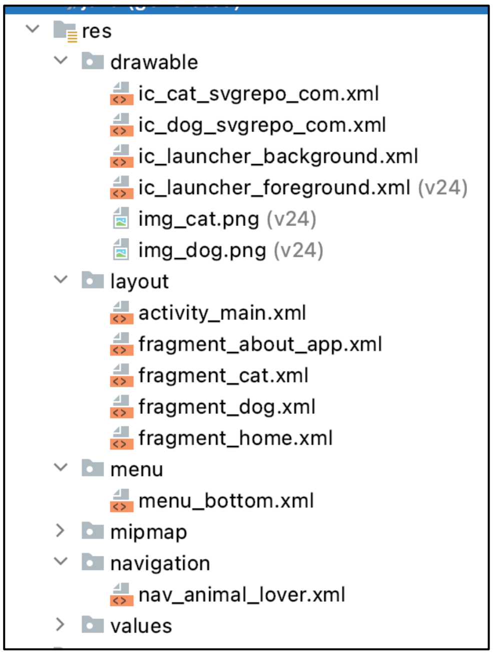
In this practical session, we will build an **Animal Lover App** together. This mobile application contains ***one activity*** (MainActivity) and ***four fragments*** (HomeFragment, CatFragment, DogFragment and AboutAppFragment). The java directory in your Android Project should be similar to the following:



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.navgraph
      AboutAppFragment
      CatFragment
      DogFragment
      HomeFragment
      MainActivity
```

Figure 1: Backend files in Java directory.

You should also include the layout file when you are creating the activity and fragments. Your resource directory should contains the following files:



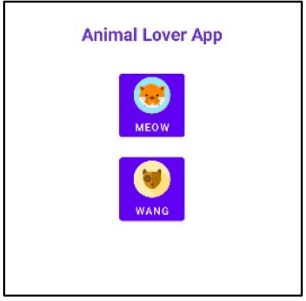
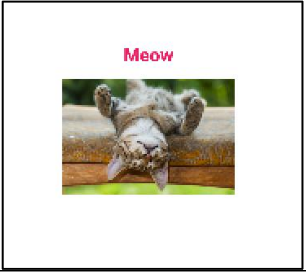
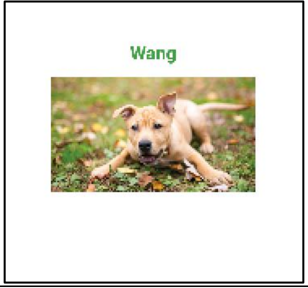
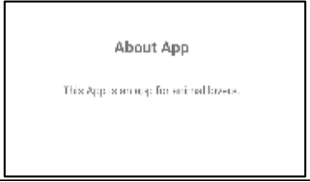
```
res
  drawable
    ic_cat_svgrepo_com.xml
    ic_dog_svgrepo_com.xml
    ic_launcher_background.xml
    ic_launcher_foreground.xml (v24)
    img_cat.png (v24)
    img_dog.png (v24)
  layout
    activity_main.xml
    fragment_about_app.xml
    fragment_cat.xml
    fragment_dog.xml
    fragment_home.xml
  menu
    menu_bottom.xml
  mipmap
  navigation
    nav_animal_lover.xml
  values
```

Figure 2: Frond-end XML files in the resource directory.

Tips: To create fragment files, use the same step as how you create the activity file.

Task 2: Get ready with the fragment files

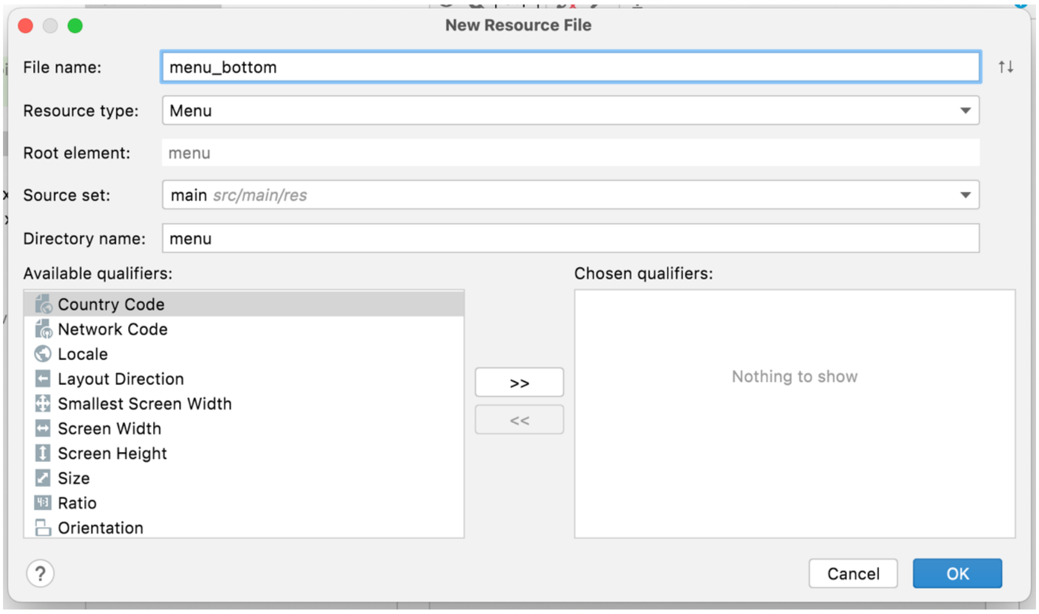
Prepare the layout XML file for the fragments:

	Screenshot	Explanation
1	fragment_home.xml 	The home menu contains: - One TextView named “TVHomeTitle” - Two Buttons named “BtnCat” and “BtnDog” When the BtnCat is clicked, it will show CatFragment. When the BtnDog is clicked, it shows DogFragment.
2	fragment_cat.xml 	This fragment contains only: - One TextView called “TVCatTitle” - One ImageView called “IVCat”
3	fragment_dog.xml 	This fragment contains only: - One TextView called “TVDogTitle” - One ImageView called “IVDog”
4	fragment_about_app.xml  This App is an app for animal lovers.	This fragment has two TextView called “TVAboutAppTitle” and “TVAboutAppDesc”.

Task 3: Get ready with the menu file

The menu_bottom.xml in the menu resource directory will store the options for our bottom navigation bar and also act as secondary options in the toolbar. Prepare the menu resource directory and a menu resource file named menu_bottom.xml:

- On the Project Panel, Right click on the **res** directory.
- Select New → Add Android Resource File.
- On the New Resource File dialog, enter File name as "menu_bottom.xml" and select Resource type as **Menu**. Others remain unchanged. Click "OK" to create the menu resource directory with the new menu_bottom.xml.



New Resource File

File name: menu_bottom

Resource type: Menu

Root element: menu

Source set: main *src/main/res*

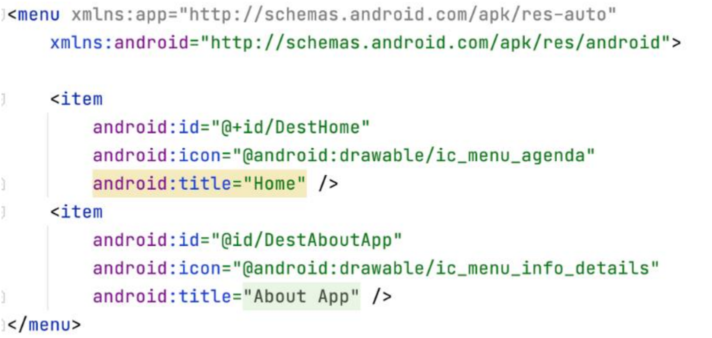
Directory name: menu

Available qualifiers:

Chosen qualifiers: Nothing to show

Figure 3: Add menu directory and resource file.

Double click to open the newly created menu resource file, include the following two items into the menu_bottom.xml file:



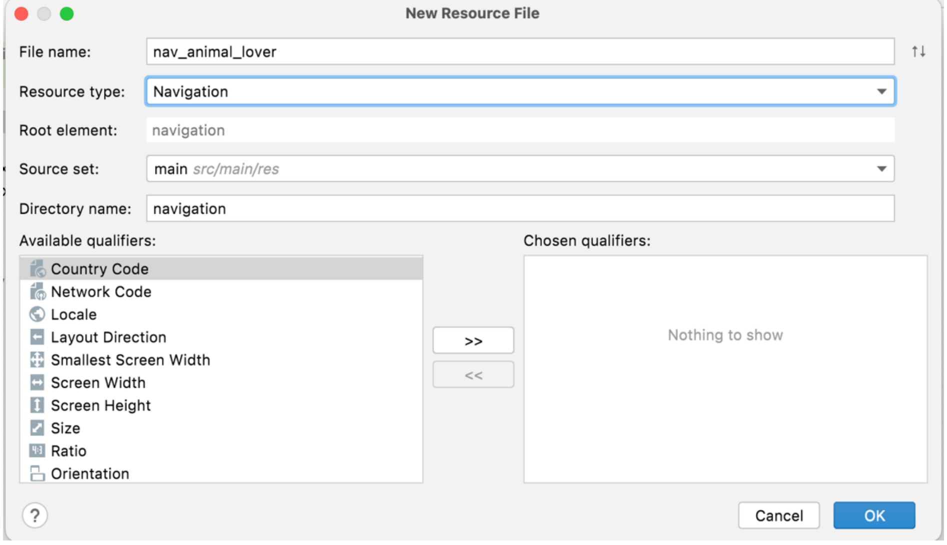
```
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/DestHome"
        android:icon="@android:drawable/ic_menu_agenda"
        android:title="Home" />
    <item
        android:id="@id/DestAboutApp"
        android:icon="@android:drawable/ic_menu_info_details"
        android:title="About App" />
</menu>
```

Figure 4: The menu_bottom.xml file.

Task 4: Use Navigation Graph

Now, let's create the nav_animal_lover.xml file which will store the navigation and naming convention for fragments in the Android Project. Prepare the navigation resource directory and a navigation resource file named nav_animal_lover.xml:

- On the Project Panel, Right click on the **res** directory.
- Select New → Add Android Resource File.
- On the New Resource File dialog, enter File name as "nav_animal_lover.xml" and select Resource type as **Navigation**. Others remain unchanged. Click "OK" to create the menu resource directory with the new menu_bottom.xml.



New Resource File

File name: nav_animal_lover

Resource type: Navigation

Root element: navigation

Source set: main *src/main/res*

Directory name: navigation

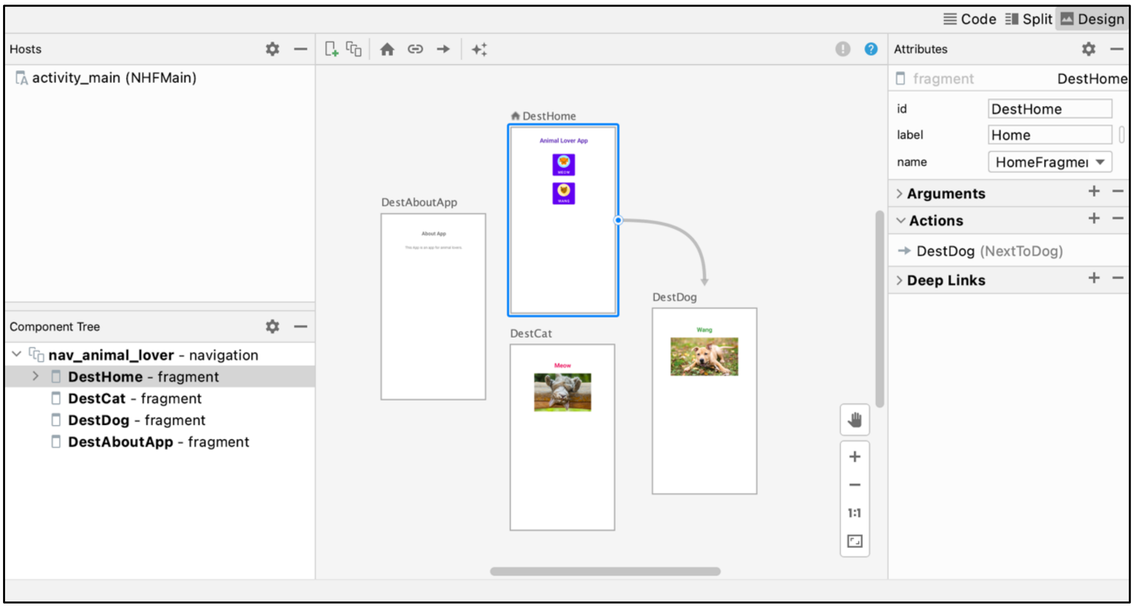
Available qualifiers:
- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation

Chosen qualifiers:

Nothing to show

Figure 5: Add navigation directory and resource file.

- After that, you will be prompted to Add Project Dependency. Click "OK" so that all the required libraries for this Navigation components will be installed and included automatically into your gradle files.



Hosts

activity_main (NHFMain)

Component Tree
- nav_animal_lover - navigation
  - DestHome - fragment
  - DestCat - fragment
  - DestDog - fragment
  - DestAboutApp - fragment

Attributes

fragment DestHome

id: DestHome

label: Home

name: HomeFragme

Arguments

Actions
- DestDog (NextToDog)

Deep Links

Figure 6: The Navigation Editor.

Once all the components are ready, the Navigation Editor will be shown on the Android Studio. On the Navigation Editor, let's add all the fragments (using the button on the Navigation

Editor. Then, modified the ID attribute of each fragment to DestHome, DestCat, DestDog and DestAboutApp. All these IDs will be used as destination reference in the java code later.

Task 5: Prepare the MainActivity layout file

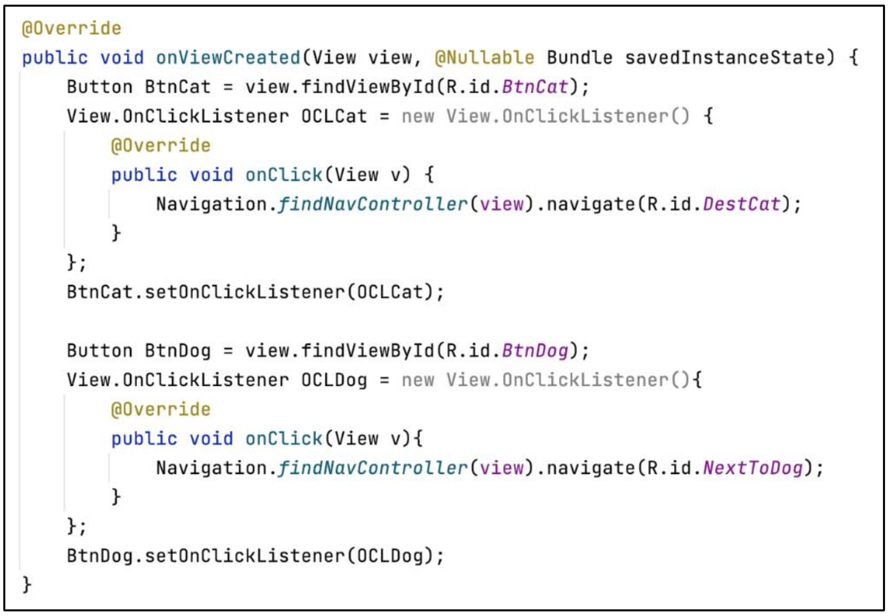
Convert the layout of the XML file to Drawer Layout. Drawer Layout is top-level container that allows the child component to appear as a “drawer”, for example slide-in from left, slide-out to right, etc. In the Drawer Layout, nest a Linear Layout which contains the following widgets: (1) Toolbar, (2) NavHostFragment, and (3) Bottom Navigation View. You may refer to the following XML code to complete the MainActivity layout file:

```xml
<?xml version="1.0" encoding="utf-8"?>

<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/DLMain"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/TBMainAct"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/NHFMain"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_animal_lover" />

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_nav_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:menu="@menu/menu_bottom" />
    </LinearLayout>

</androidx.drawerlayout.widget.DrawerLayout>
```

Task 6: Navigate using Destination and Action

Now, let’s link the first fragment!

On the HomeFragment Activity, override the OnViewCreated callback method. The navigation steps are similar with our usual routine, which is: (a) Get the frontend Button (BtnCat and BtnDog), (b) Create the onClickListener, and (c) Bind the onClickListener to the button. (Or the other methods taught in the previous class before.)



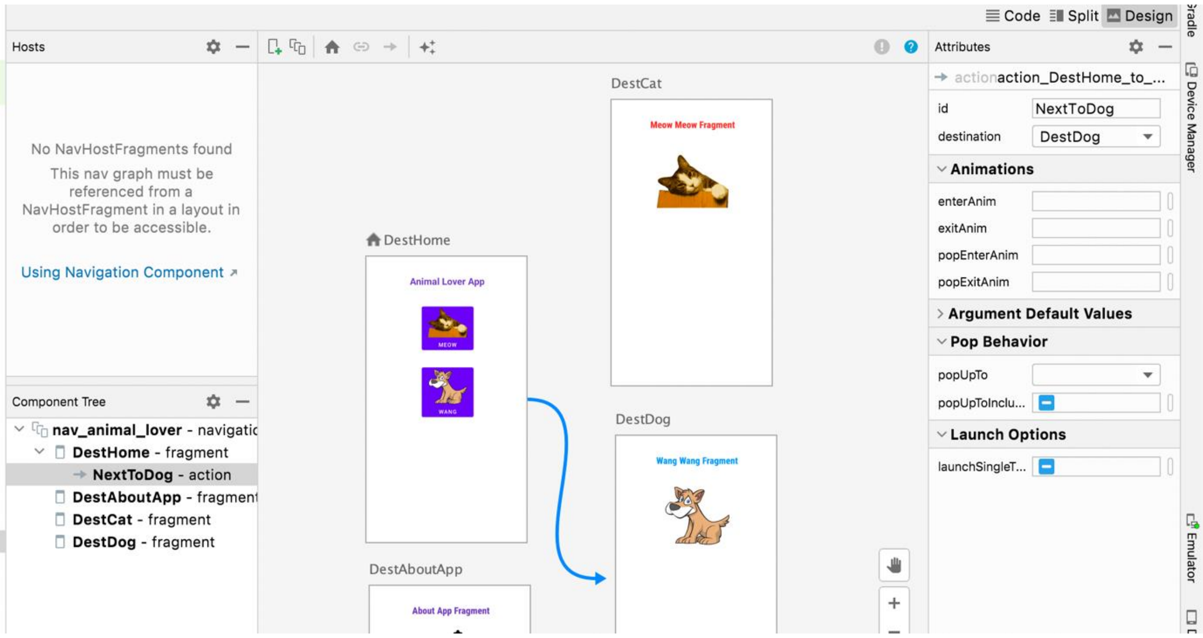
```
@Override
public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
    Button BtnCat = view.findViewById(R.id.BtnCat);
    View.OnClickListener OCLCat = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Navigation.findNavController(view).navigate(R.id.DestCat);
        }
    };
    BtnCat.setOnClickListener(OCLCat);

    Button BtnDog = view.findViewById(R.id.BtnDog);
    View.OnClickListener OCLDog = new View.OnClickListener(){
        @Override
        public void onClick(View v){
            Navigation.findNavController(view).navigate(R.id.NextToDog);
        }
    };
    BtnDog.setOnClickListener(OCLDog);
}
```

Figure 7: Navigation to other fragments.

There are two navigation methods using NavGraph, which is:
Method (1): via NavGraph Destination using Fragment ID, and
Method (2): via Action between two fragments.

Refer to the implementation of BtnCat for Method (1) to navigate to DestCat (this is the id we set for the Cat Fragment in the navigation graph)

Then, for the implementation of BtnDog, we will use Method (2). On the Navigation Editor, connect the DestHome to DestDog by dragging an arrow from DestHome pointing to DestDog. Let's name this action as "NextToDog" by changing the id of the action. Then, on the backend code, instead of navigating to the Destination fragment as in Method (1), we will navigate using action (refer to the backend code in figure above).

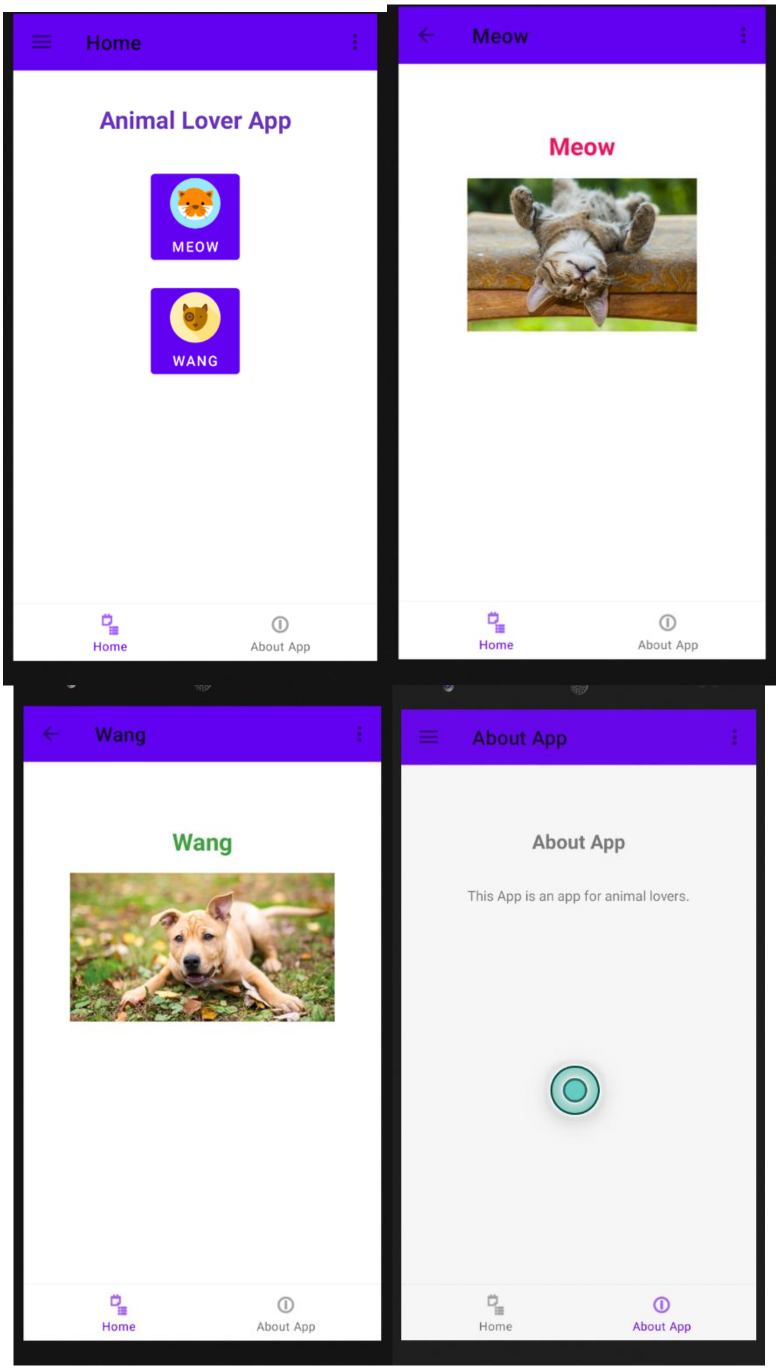


id NextToDog
destination DestDog

Figure 8: Create action in Navigation Editor.

Complete all the tasks in this practical exercises, beautify the fragments (e.g., adding descriptions, add more animals, etc.) and most importantly, make sure the navigation works! We shall continue on the Action Bar, Bottom Navigation and others in the coming practical exercise.

Example output:



This App is an app for animal lovers.

Submission

You are required to submit the Android Studio Project to the Spectrum before the next Tutorial class.