

Lab 2: Basic GUI Components

(Due: Mon, Feb 10, 2020 @ 3.25pm)

Introduction:

This lab will show you how to work with the basic GUI components and help you learn how to build a simple user interface in Android. You will learn how to work with buttons, text-fields, toasts and images. You will also create your own currency converter application.

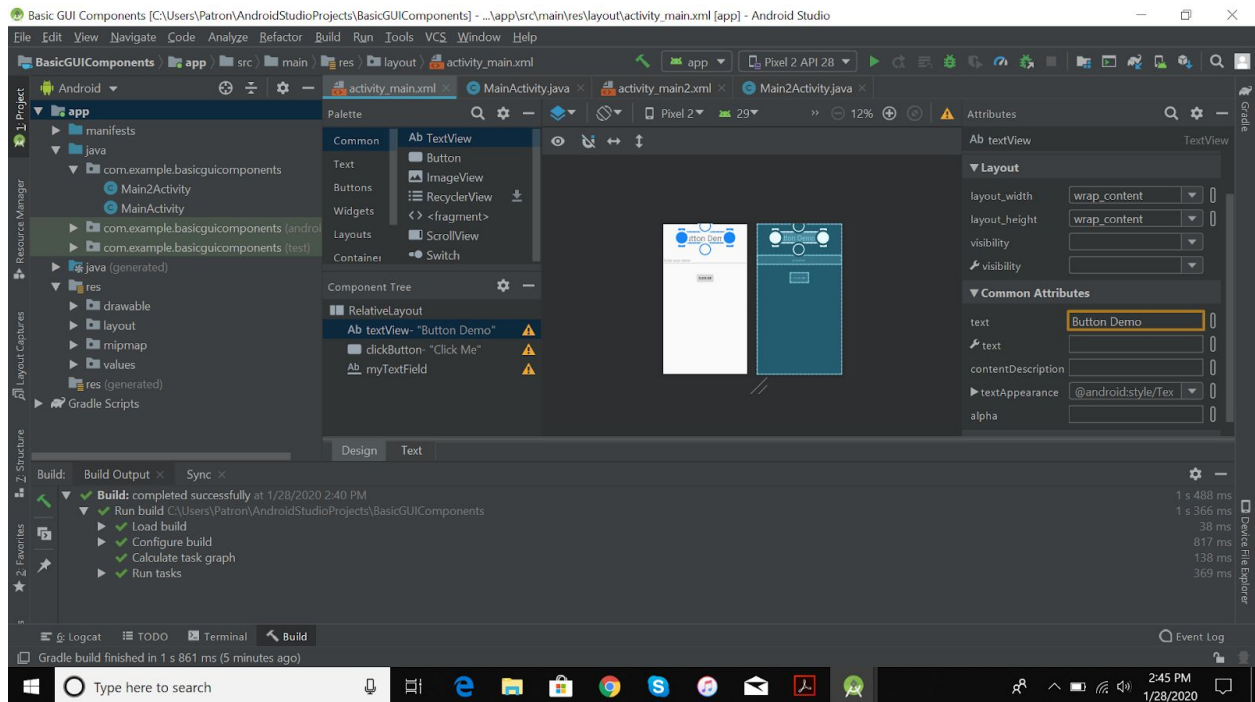
All of your work should be pushed to some git repository. For Milestone 1, you will clone from the provided template in the classroom repository, make all changes and you will have to commit and push them to the cloned repository in github classroom. For Milestone 2, you will be creating a new project, and we expect you to commit and push this new project into your own github account (not to classroom).

Also in many of the tasks below, we describe what you need to do in text briefly, and follow it with a picture in which updated code is shown. In specific cases, you will need to type that code into your project to accomplish the task.

Directions:

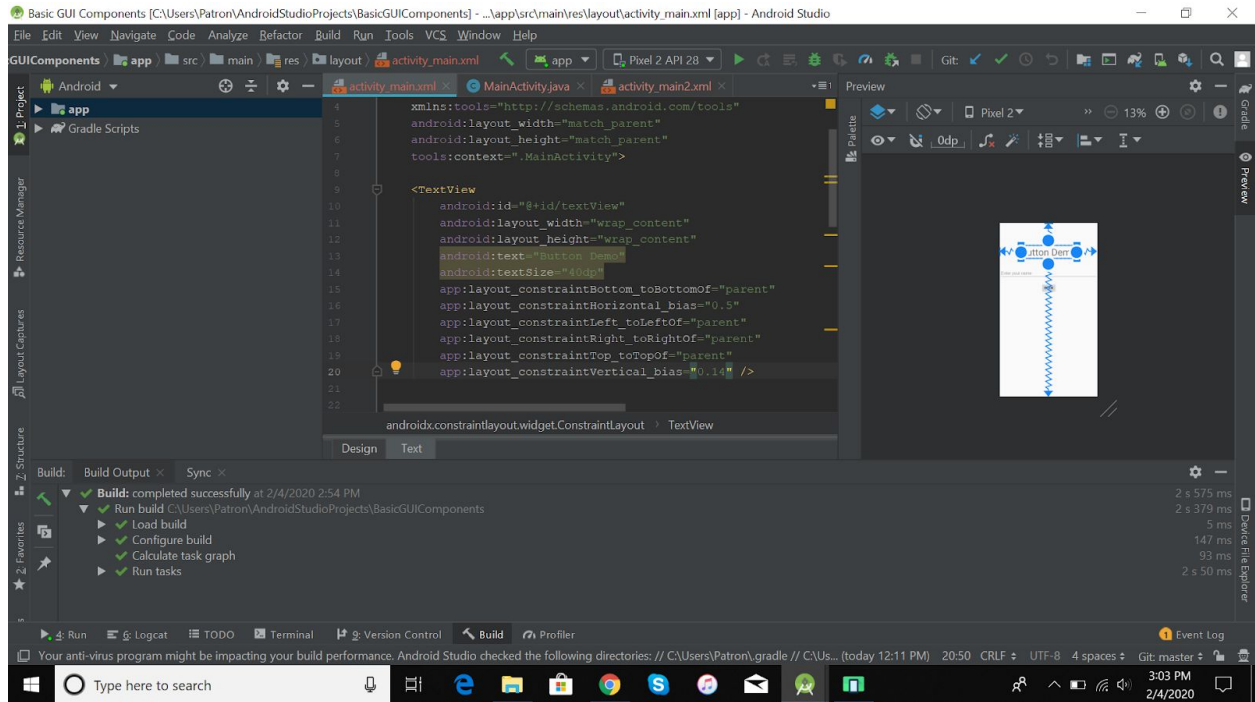
Milestone 1 - Setup your first button and display a toast. In this milestone, we will learn how to add TextViews, TextFields, Buttons and Toasts to your app. We will also learn how to switch activities using intents. We are going to take input using an EditText component, and display our output in (i) toasts and (ii) different activity using intents.

1. As you did in lab 1, clone the Lab2 repository by selecting File > New > Project from Version Control > Git
2. Then click the following link and accept the invitation:
<https://classroom.github.com/a/IH4s86fH>
3. You'll get a link in the form
<https://github.com/CS-407-Spring2020/lab2-yourgithubnamehere>
4. Copy this link in the URL text field in 'Clone' in Android Studio and click 'Clone'.
5. Change the default "Hello World" TextView with a meaningful header for your app e.g. "Button Demo". You can do this in 2 ways:
 - a. Go to the "Design" tab in activity_main.xml. In the preview screen that is being displayed, click on the TextView. Its attributes should appear on the right hand side. Change the text attribute to "Button Demo". This is one way to change attributes of components.



OR

- b. Go to the “Text” tab in activity_main.xml. In the TextView component change the text attribute to display “Button Demo”. You can also change the position of the TextView by either dragging it across the screen or changing its attributes. Remember that we are using Constraint Layout. Thus we need to add the required constraints to our component. Here is an example.



Setting up TextView

Attributes you can use:

`layout_constraintTop_toTopOf` — Align the top of the desired view to the top of another.

- `layout_constraintTop_toBottomOf` — Align the top of the desired view to the bottom of another.
- `layout_constraintBottom_toTopOf` — Align the bottom of the desired view to the top of another.
- `layout_constraintBottom_toBottomOf` — Align the bottom of the desired view to the bottom of another.
- `layout_constraintLeft_toTopOf` — Align the left of the desired view to the top of another.
- `layout_constraintLeft_toBottomOf` — Align the left of the desired view to the bottom of another.
- `layout_constraintLeft_toLeftOf` — Align the left of the desired view to the left of another.
- `layout_constraintLeft_toRightOf` — Align the left of the desired view to the right of another.

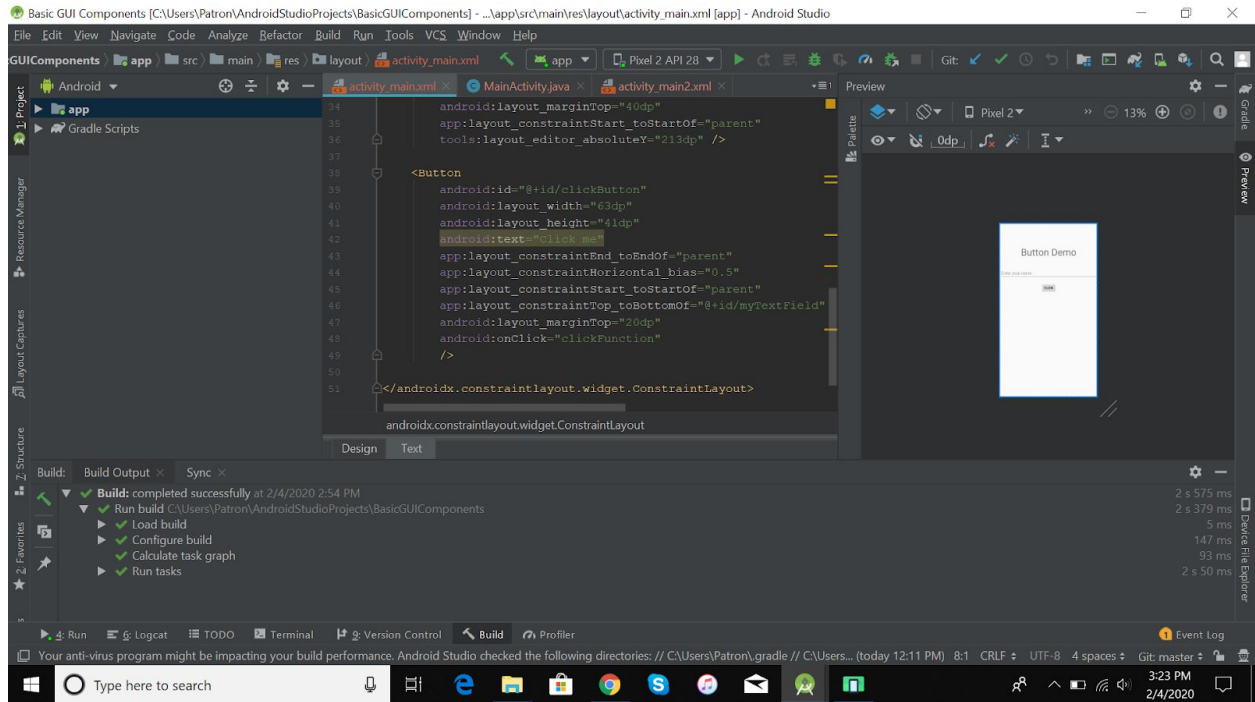
- `layout_constraintRight_toTopOf` — Align the right of the desired view to the top of another.
- `layout_constraintRight_toBottomOf` — Align the right of the desired view to the bottom of another.
- `layout_constraintRight_toLeftOf` — Align the right of the desired view to the left of another.
- `layout_constraintRight_toRightOf` — Align the right of the desired view to the right of another.
- `layout_constraintHorizontal_bias`: This allows us to position a view along the horizontal axis, using a bias value, this will be relative to its constrained position
- `layout_constraintVertical_bias`: This allows us to position a view along the vertical axis, using a bias value, this will be relative to its constrained position

6. Create a new button. You can do this in one of the following two ways:

- a. In the “Design” tab in `activity_main.xml`, drag a button component to the center of the screen. Select this component to view its attributes on the right side of the screen. You can edit these attributes as you like. Use the attributes mentioned in the previous step to position your button in the center horizontally.

OR

- b. In the “Text” tab in `activity_main.xml`, create a new button component and give it the properties as in the following snapshot. Here are the functionalities of some of these attributes. You can use the attributes mentioned in the previous step. Make sure your button is centered horizontally.

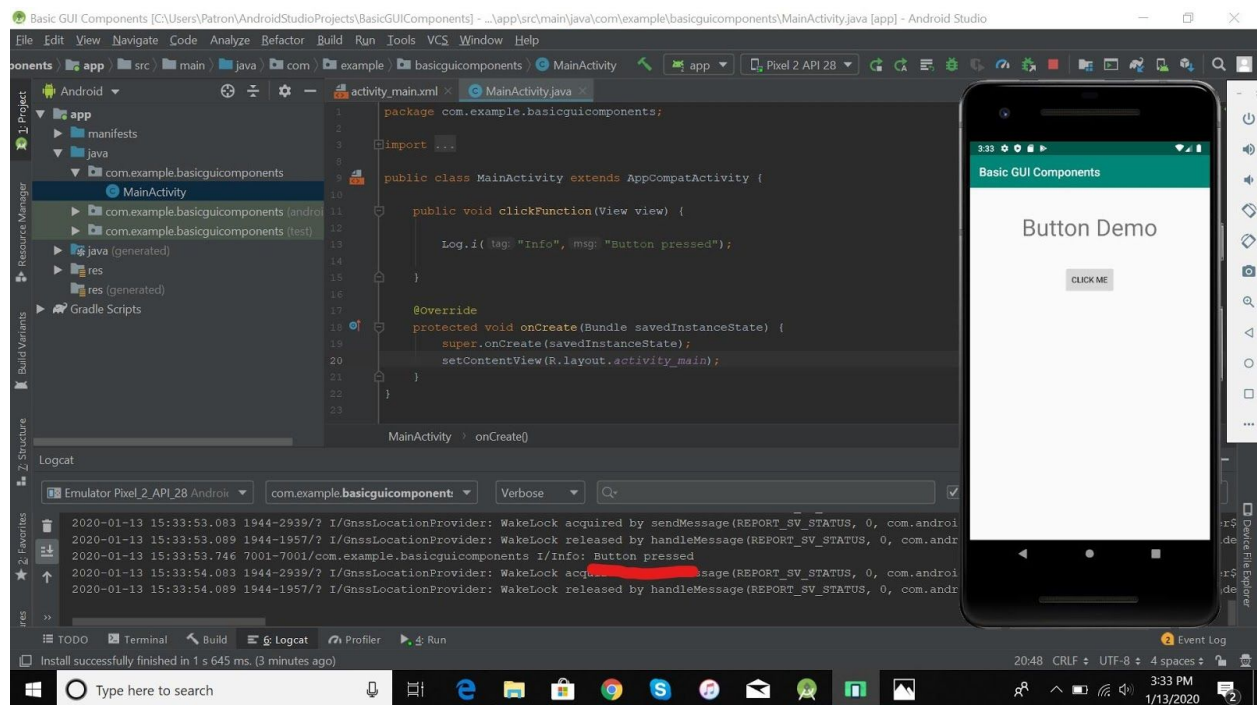


Creating new button

Note: You can assign attributes as you like, but make sure that each component has an id and that your app looks presentable. For buttons, make sure that each button has an `onClick` property. To define the click event handler for a button, add the `android:onClick` attribute to the `<Button>` element in your XML layout. The value for this attribute must be the name of the method you want to call in response to a click event. You can refer to the code in the picture above.

Note: When you add certain attributes or properties, Android studio might prompt you to import some packages. Make sure you import them so that you don't run into any errors. For example: import `android.view.View` in `MainActivity.java`

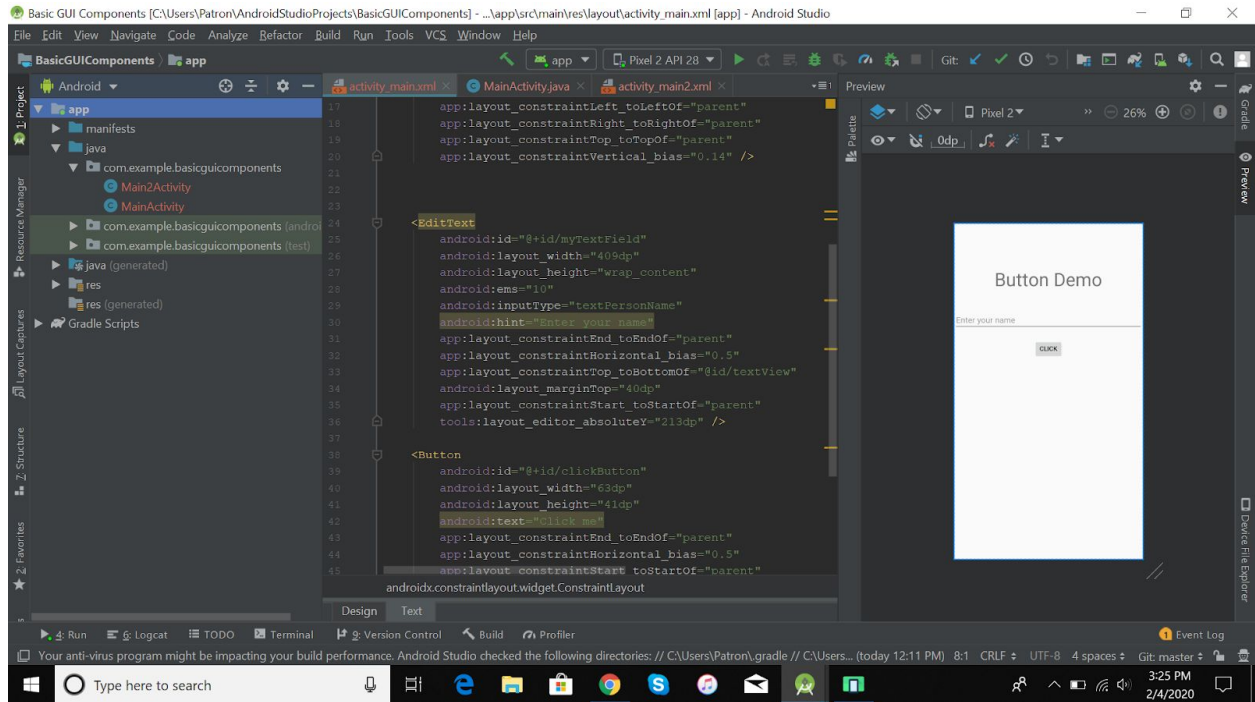
7. Switch to `MainActivity.java` and write the code for the function you specified in the `onClick` property for your button in `activity_main.xml`. This code will tell the app what to do when the button is pressed. For now, we will display a message in Logcat when the button is pressed. Here is an example



Display "Button pressed" in LogCat

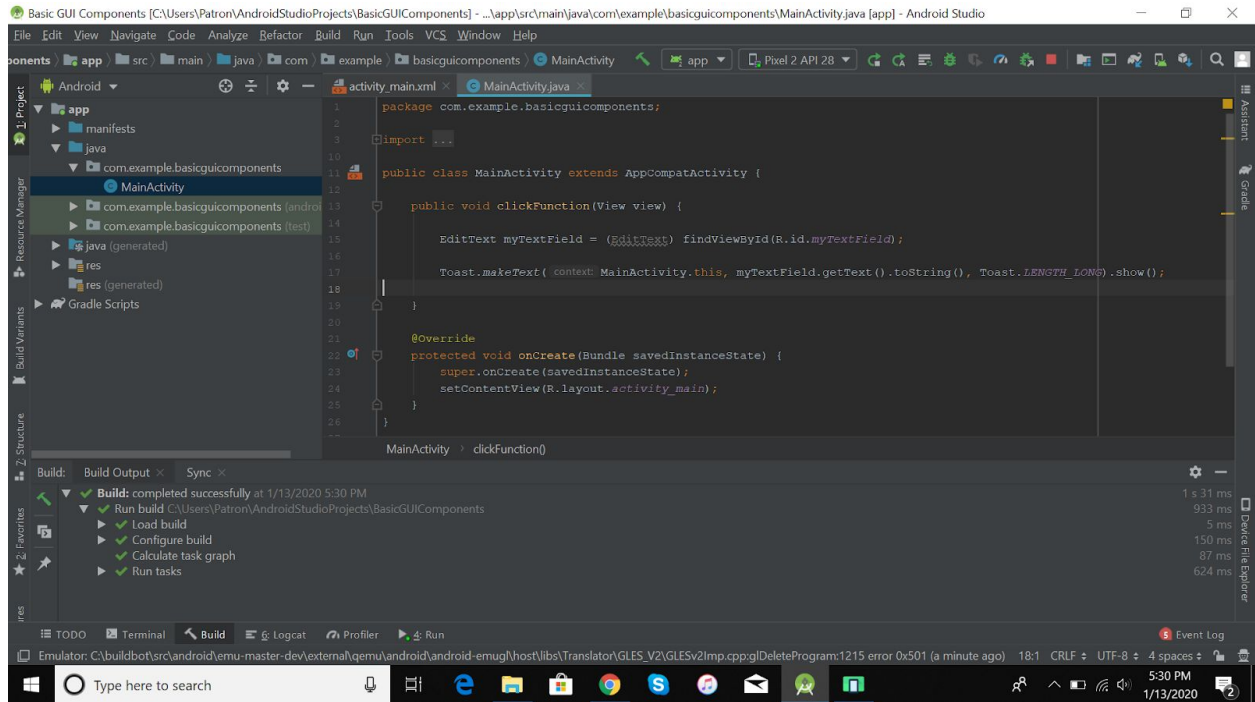
8. Run the app and check if the button is working properly. You can switch to the Logcat tab by selecting "6: Logcat" at the bottom of the screen. You'll be able to see your output in the Logcat when the button is pressed.
9. Now that we know how to work with a button, we will learn how to use text fields to take user input and how to use toasts to display output.
10. To create a text field, switch to activity_main.xml. Follow the same instructions as you did to create a new button.
 - a. If you use the "Design" tab, drag a "Plain Text" component to the center of the screen. Assign attributes as needed.
 - OR
 - b. If you use the "Text" tab, create a new EditText component as shown below. Assign properties as you like.

Remember to assign an id to the component.



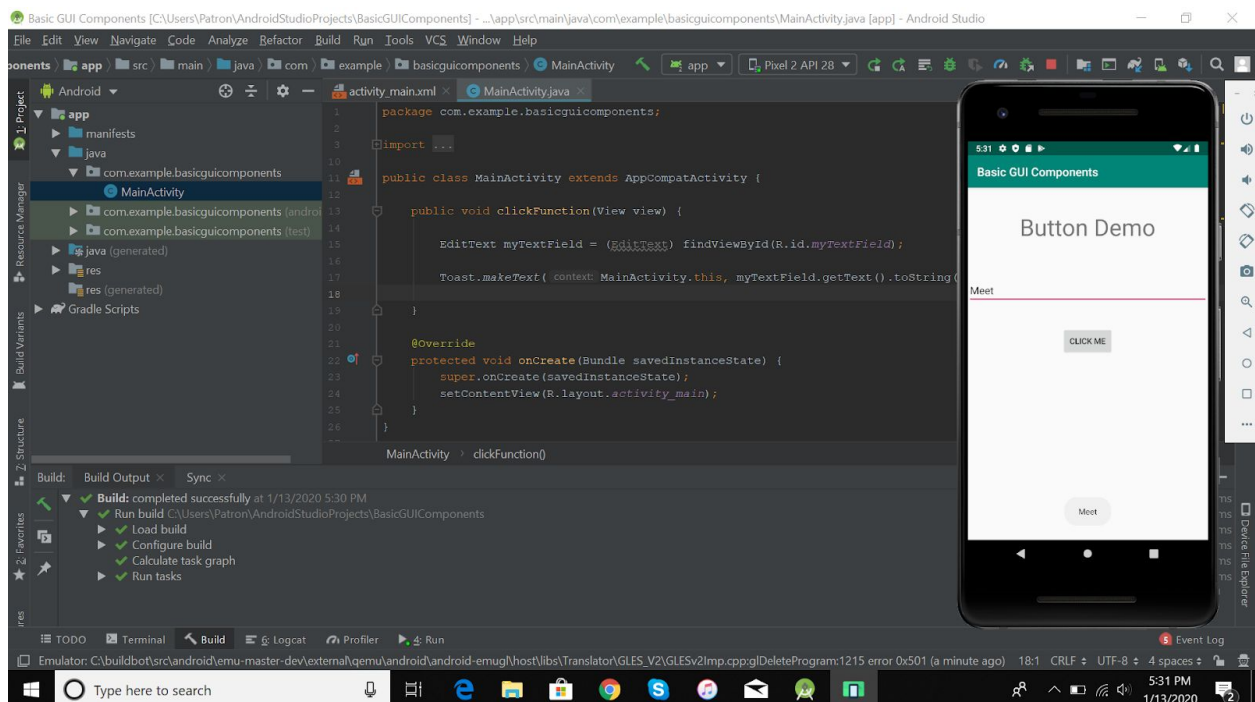
Setting up editText (text field)

11. We want to display the text that the user enters in the text field. We can do this using toasts.
12. To get the text that the user entered in the text field, switch to MainActivity.java and create an EditText variable as shown below. We use the findViewById() method to find the descendant view with the given id. Make sure your id matches the id of the editText component.
13. Finally, we use the getText() method to get the text that the user entered in the text field. We can display that information using Toast as shown in the following snapshot.

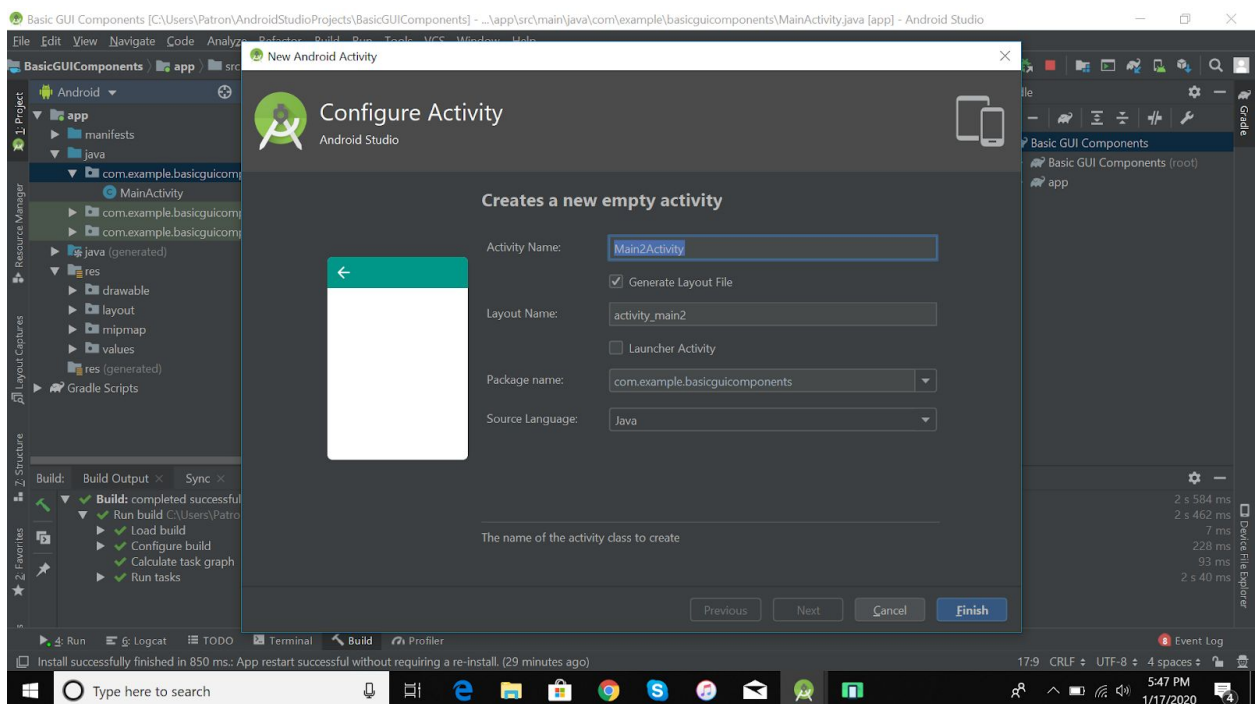
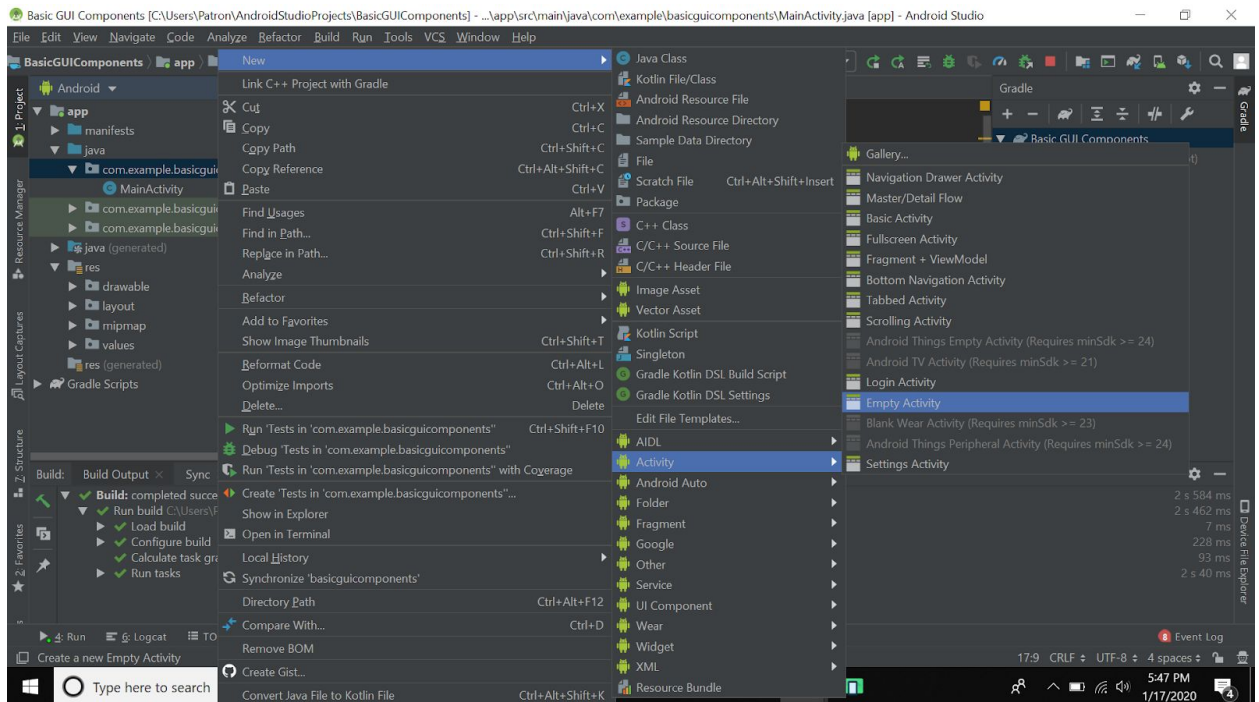


Getting input from text field and creating a Toast

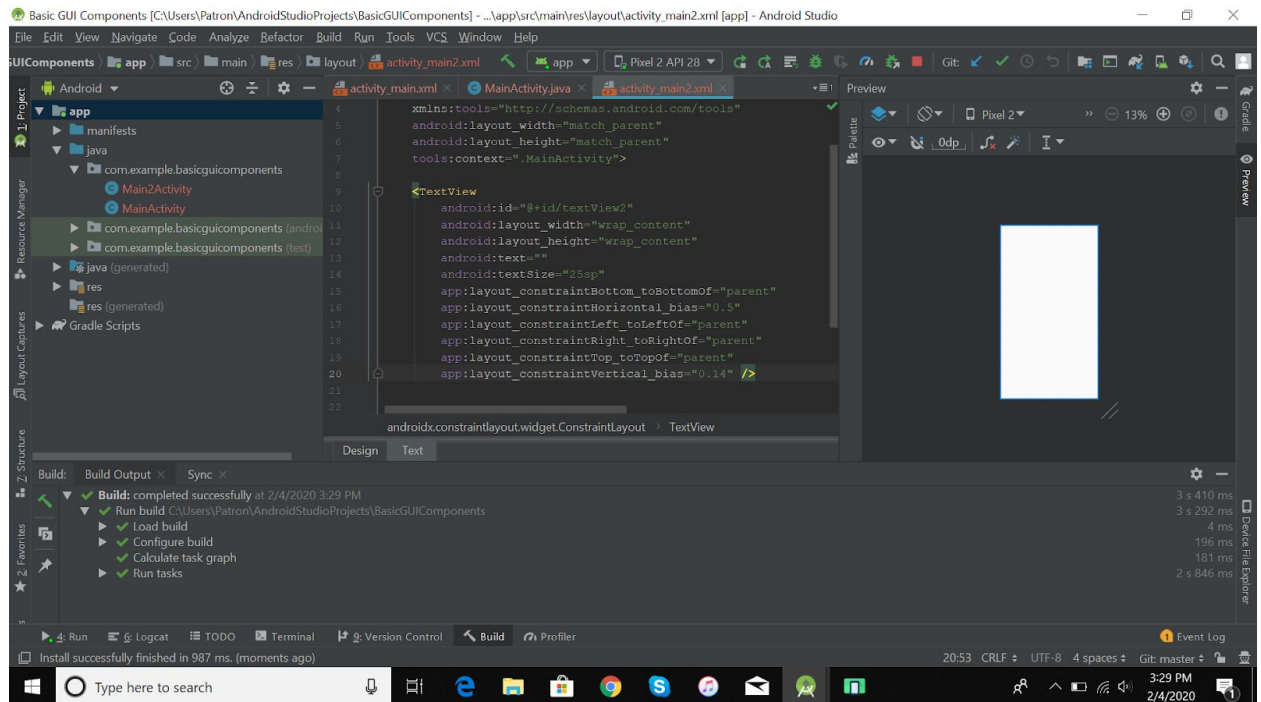
DEMO OUTPUT



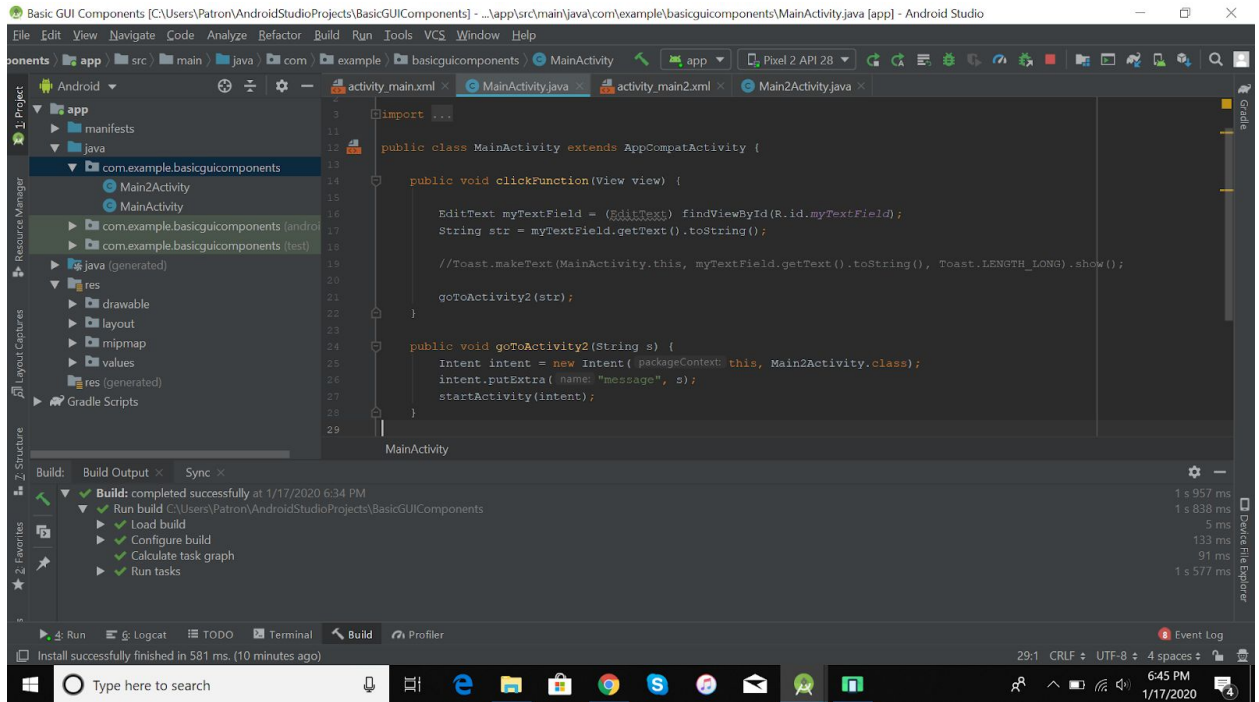
14. Now we will learn to use intents in our mobile apps and how to pass some data from one activity to another.
15. First, let's create a new activity in the same project that we are doing. To create a new activity, right-click on `com.example.your-project-name` in the "Project" tab on the left side of Android studio. Select `New -> Activity -> Empty Activity`. Here are some snapshots to help you.



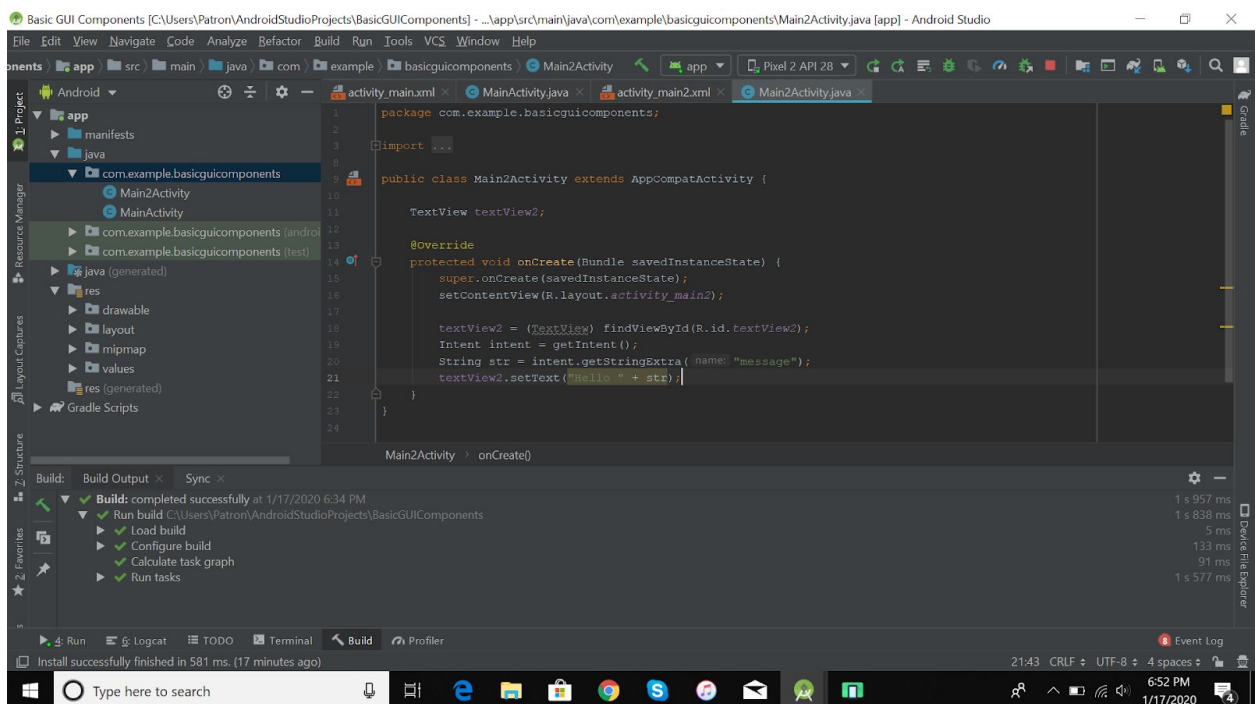
16. Next, we want to change what happens when the button is clicked. Instead of showing a toast like before, we will switch to a new activity when the “Click Me” button is clicked. Instead of showing our output as a toast, we will show it in a new activity.
17. To do this, let’s create a new TextView in our new Activity’s xml file, activity_main2.xml, where we’ll show the output. Keep the text attribute empty for now. Here’s a snapshot:



18. Next we will create an intent in our Main Activity which will help us switch activities. Go to MainActivity.java. You do not need to remove the toast. You can comment it if you want. Create a new function called `goToActivity2()` like in the snapshot below. Create an intent in the function like in the snapshot. The `putExtra()` function helps us pass data to the second activity. In this case, we are using `putExtra()` to pass the string that we want to output to the new activity. Call this `goToActivity2()` method in the `clickFunction()` method so that it is called when the button is clicked.



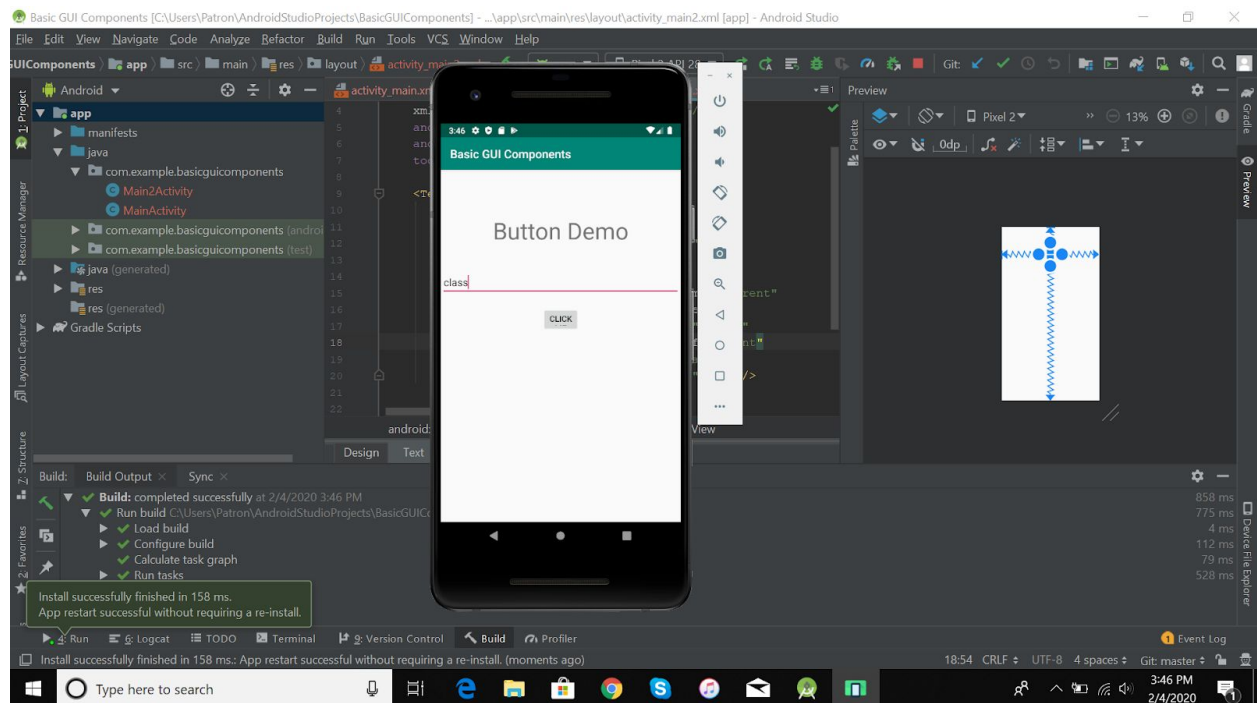
19. Now go to the second activity, i.e. `Main2Activity.java`. Create a `TextView` instance to show the string we are going to get from the intent. In the `onCreate()` method, create an instance of the `TextView` we created earlier in `activity_main2.xml`. We will use the `getStringExtra()` method to get the string from previous activity. And then we will use the `setText()` method to set the text for the `TextView`. Here's what the code would look like.

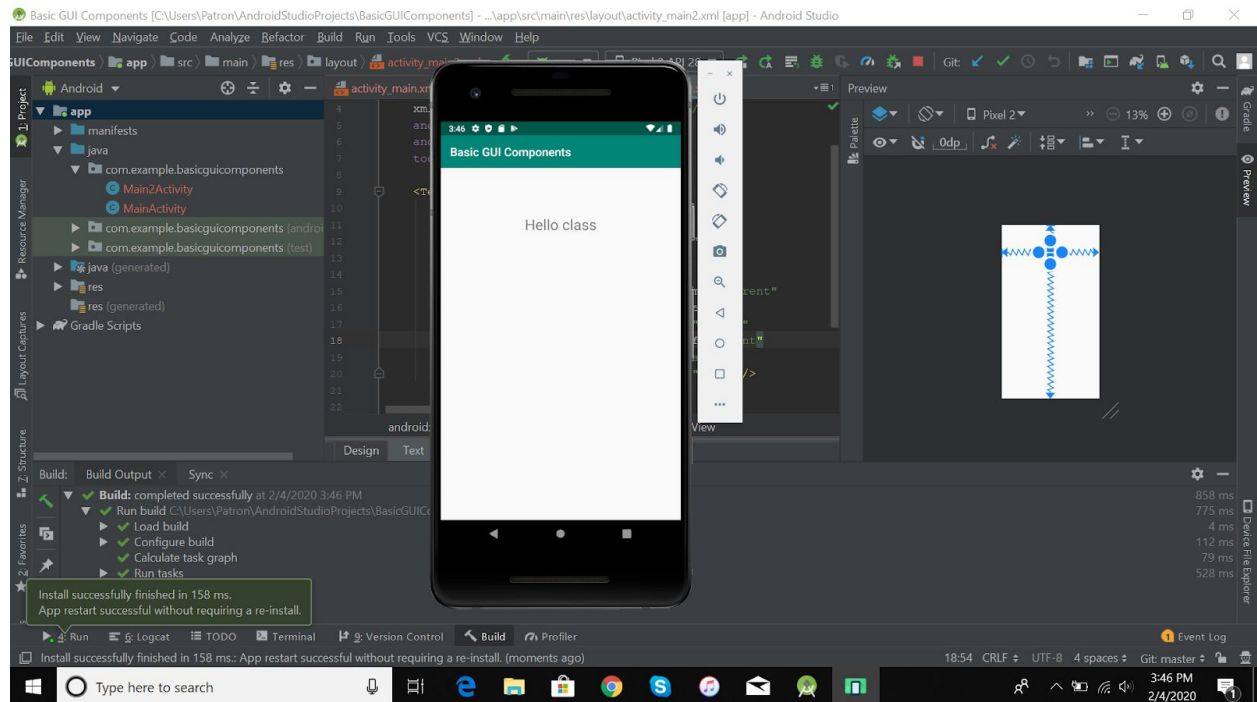


DELIVERABLES:

Good job, you have finished milestone 1. You need to show the following things to your TA:

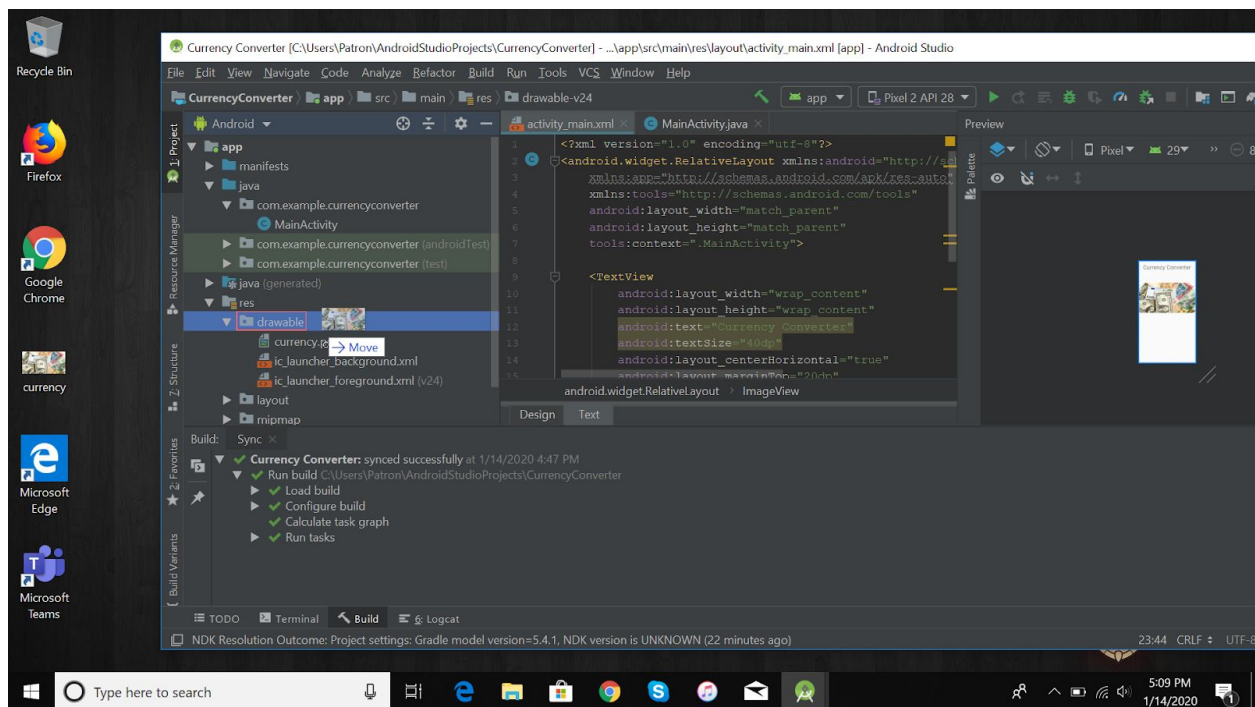
- The TextView, EditText and Button in a presentable format on your emulator
- Correctly being able to enter input in the EditText component
- On clicking the button, it should take us to a new activity where the output is displayed.
- The output displayed must be taken from the user-entered text in the EditText Component in the first activity.
- Commit and push changes to the repo
- **Here is a Demo Output:**





Milestone 2 - Currency converter app. Using the components we learnt in the previous milestone, we will create a currency converter app which converts dollars to pounds.

1. In this section, we are going to create a currency converter app
2. First, let's see how to add images to android apps
3. Create a new project in Android Studio to create the currency converter app
4. Download a suitable image for the currency converter app from the web.
5. Copy the image to the drawable folder found inside the res folder. You can do this by either dragging the image in the folder as shown in the following snapshot or copying the image in the drawable folder found by the path
C:\Users\Your-PC-Name\AndroidStudioProjects\CurrencyConverter\app\src\main\res\drawable-v24

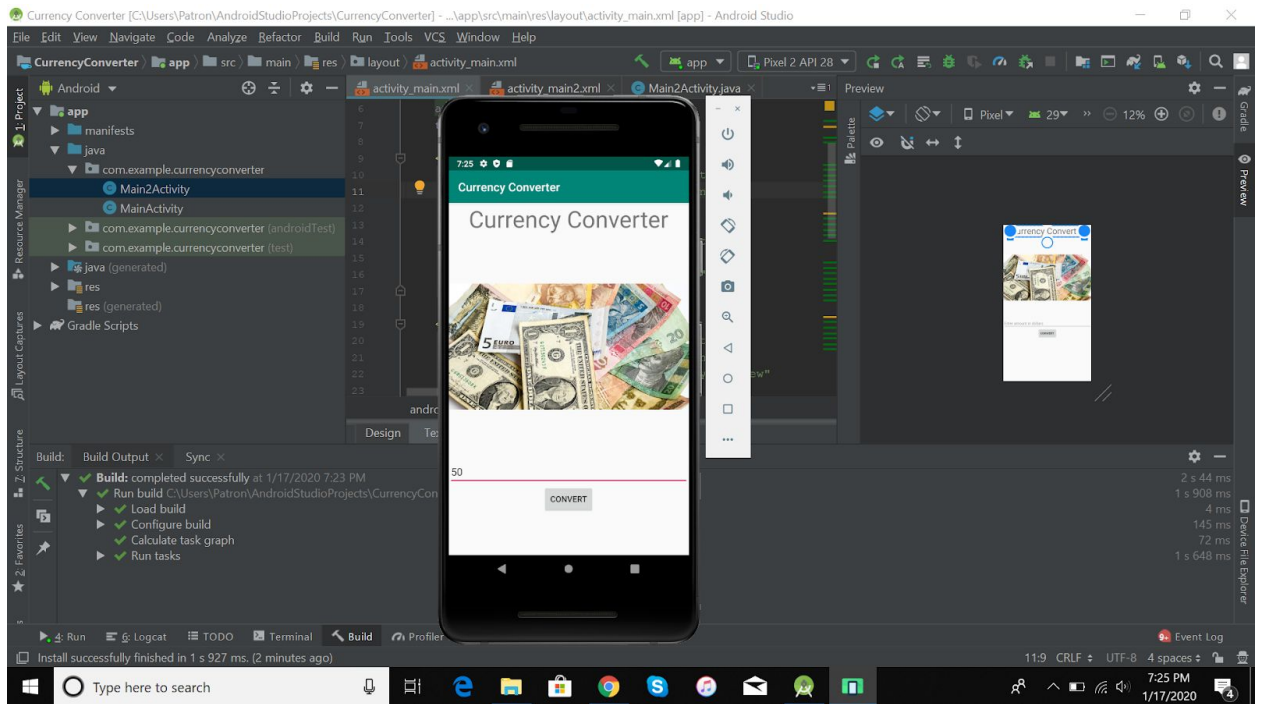
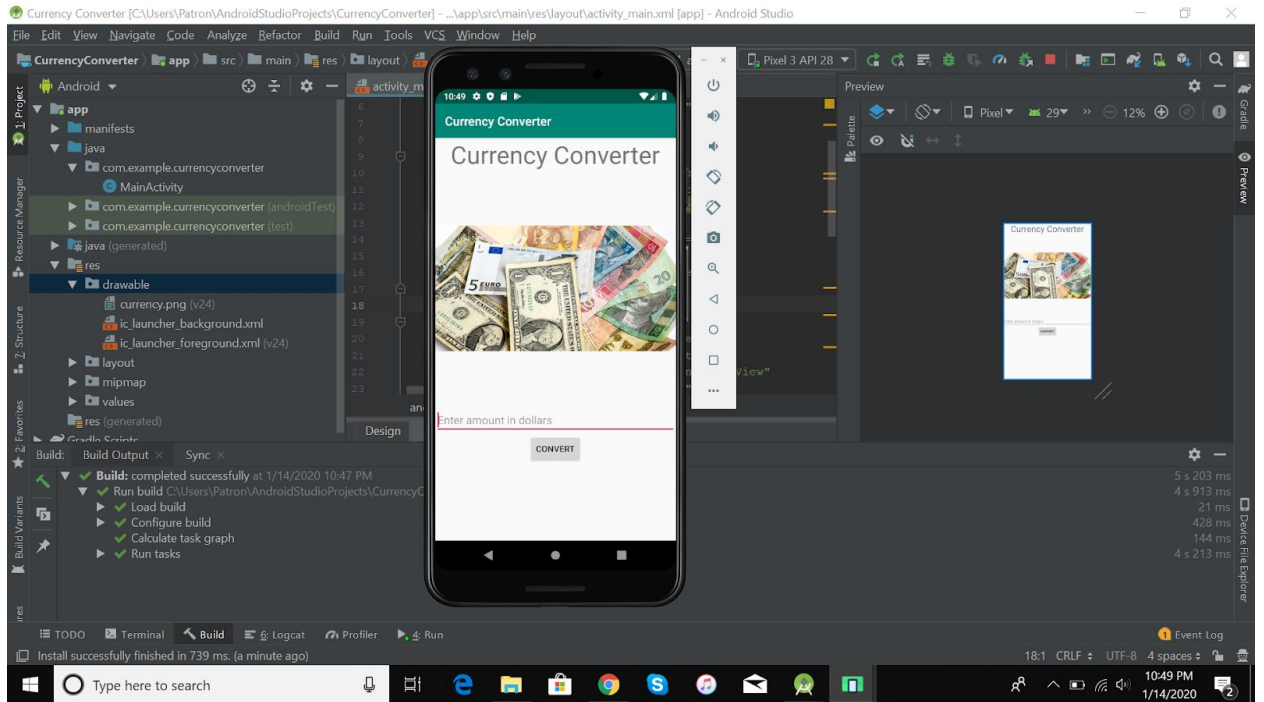


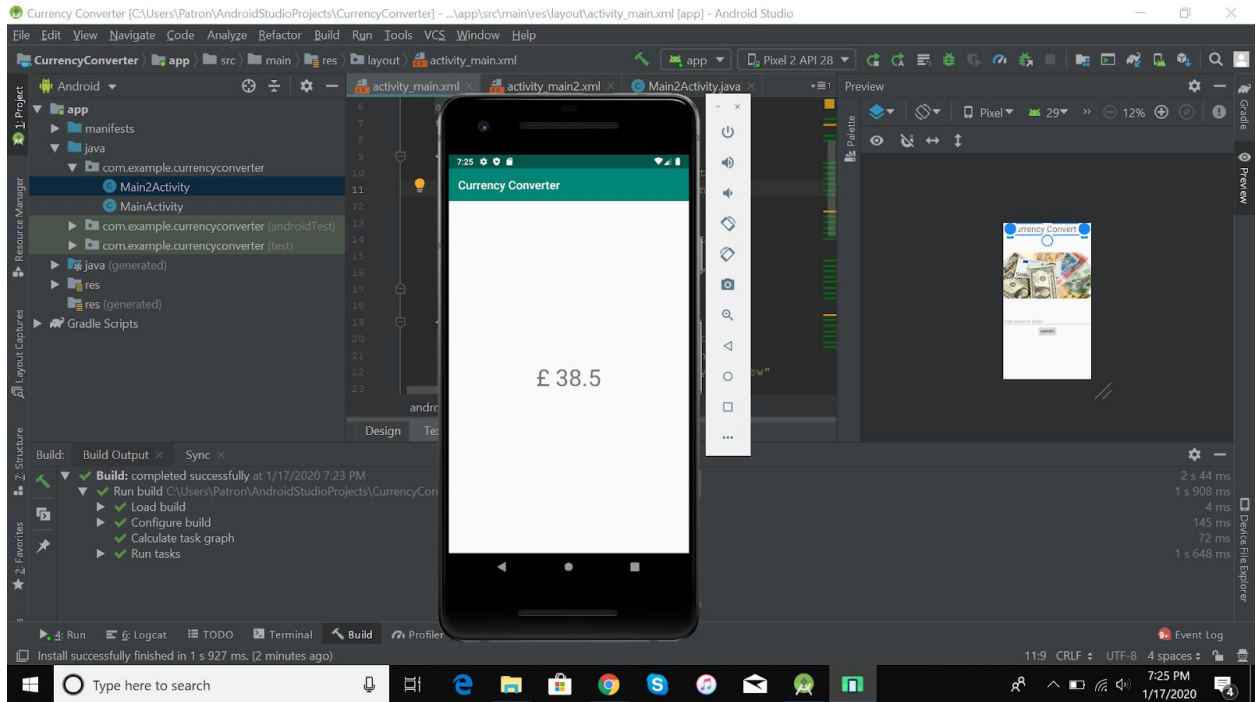
6. Next create an ImageView in activity_main.xml by either dragging an ImageView component on the screen in the “Design” tab or by creating an ImageView in the “Text” tab. To add the Image to the ImageView, under the “src” attribute, write “@drawable/name-of-your-image”. Before doing so, make sure the image was copied into the drawable folder in the previous step.
7. Using this and the components we learnt in the Milestone 1, create a currency converter app that converts dollars to pounds. Use the following conversion rate: 1 dollar = 0.77 pounds

DELIVERABLES:

Good job on finishing the milestones! After completion, you need to show the following things to a TA:

- The currency converter app in a presentable format
- PlainText text-field is taking input correctly
- The output is being displayed in a new activity
- The conversion from dollars to pounds is correctly done.
- The fully functioning currency converter app
- **Add this project to your own github repo (not to classroom repo)**
- **Here are the Demo Output screens:**





Conclusion:

Nice job getting through this lab! You have created your first interactive app! There are a lot more things you can do with the Android components. You can learn more about these components at the following link:

<https://developer.android.com/training/basics/firstapp/building-ui>

References:

- <https://developer.android.com/studio/write/layout-editor>
- <https://developer.android.com/training/basics/firstapp/building-ui>