|  |  |  |
| --- | --- | --- |
| **Activity: Socrative (Day-1 & Day-2)** | **Start: 9:10 am** | **Stop: 9:45 am** |
| Done with the socrative activity conducted by our mentor. | | |

|  |  |  |
| --- | --- | --- |
| **Activity(1-12): RecyclerView, ViewHolder, Layouts** | **Start: 9:50 am** | **Stop: 12:30 pm** |
| * Recyclerview plays a vital in a mobile applications. * In presence of a huge data which cannot be fit on screen at a time, recyclerview will be used as scroller. * Recyclerview has a adapter which will be useful to provide new view for recycler when needed. * The adapter gets data from data source and fetches into viewholder. * With the help of viewholder we can find “view by id” of each view. * To accomplish the process of viewholder below steps are to be followed:   + Create the recyclerview in the layout.   + Create the list item layout and viewholder.   + Add the recyclerview adapter.   + Add the layout manager and connect everything together. | | |

|  |  |  |
| --- | --- | --- |
| **Group Discussion (Day-1: Why do we need mobile application)** | **Start: 11:30 am** | **Stop: 12:00 pm** |
| * Mobile Apps Offer Better Personalization. * Ease of Sending Notifications. * Making Use of Mobile Device Features. * Ability to Work Offline. * Users Spend More Time on Apps. * Apps Can Work Faster Than Websites. | | |

|  |  |  |
| --- | --- | --- |
| **Activity(13-17): Adapter, wiring up recyclerview** | **Start: 2:40 pm** | **Stop: 4:30 pm** |
| * The RecyclerView widget is a more advanced and flexible version of ListView. * In the RecyclerView model, several different components work together to display your data. * The views in the list are represented by viewholder objects * The view holder objects are managed by an adapter, which you create by extending RecyclerView.Adapter. * To feed all your data to the list, you must extend the RecyclerView.Adapter class. * The RecyclerView uses a layout manager to position the individual items on the screen and determine when to reuse item views that are no longer visible to the user. | | |

|  |  |  |
| --- | --- | --- |
| **Activity(18-22): Click on items** | **Start: 4:30 pm** | **Stop: 6:00 pm** |
| * A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it. * When the user clicks a button, the Button object receives an on-click event. * To define the click event handler for a button, add the android:onClick attribute to the <Button> element in your XML layout * The appearance of your button (background image and font) may vary from one device to another, because devices by different manufacturers often have different default styles for input controls. | | |

|  |  |  |
| --- | --- | --- |
| **Activity(24-27): Intent intro, framework** | **Start: 6:30 pm** | **Stop: 7:30 pm** |
| * An Intent is a messaging object you can use to request an action from another app component. * Although intents facilitate communication between components in several ways, there are three fundamental use cases starting an activity, starting a service, delivering a broadcast. * Intents are of two types they are:   + **Explicit intents:** It specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name.   + **Implicit intents:** It do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. * An Intent object carries information that the Android system uses to determine which component to start. | | |

|  |  |  |
| --- | --- | --- |
| **Activity(28-31): Creating an Activity** | **Start: 8:20 pm** | **Stop: 9:30 pm** |
| * An activity is a single, focused thing that the user can do. * Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View). * Activities in the system are managed as an activity stack. * When a new activity is started, it is placed on the top of the stack and becomes the running activity. The previous activity always remains below it in the stack | | |

|  |  |  |
| --- | --- | --- |
| **Activity(32-36): Passing data between activities** | **Start: 9:40 pm** | **Stop: 11:20 pm** |
| * To send data to another activity, all you need to do is specify the data and its type, the system will identify compatible receiving activities and display them to the user or immediately start the activity. * Sending and receiving data between applications with intents is most commonly used for social sharing of content. * Intents allow users to share information quickly and easily, using their favorite applications. * The most straightforward and common use of the ACTION\_SEND action is sending text content from one activity to another. * To share multiple pieces of content, use the ACTION\_SEND\_MULTIPLE action together with a list of URIs pointing to the content. | | |

|  |
| --- |
| **Deliverable Status** |

**Module-2:**

**<https://github.com/manojbandari/Android/tree/master/Module%202>**