App: CookerPro
Group : 7

Group Members:

Joji Thomas

Angel Villa

Aaditya Naik

Matthew Widodo

Chase Walsh

Grad Req. (Login Authentication) - This requirement was done by our grad student Aaditya who was instructed to do an additional requirement alongside the ones that were already listed in the project outline. His requirement can be seen upon the initial startup of the app as users are first shown a screen asking the user to sign in. If the user is new they can also click a button that redirects them to a page to create a new user. These users are stored and authenticated through the use of Google Firebase. If the user wants to logout of the app they can do so by clicking the exit button on the top right of the screen.

Call backs - The call backs requirement is fulfilled through our Homefragment.kt file which executes it when the user clicks on the recipe. The onRecipeItemClick interface calls back to the callback execution in different views on the device and onFavoriteItem click are two callbacks that are being called from the recipe Adapter.

Logging - This requirement is fulfilled through the use of Logcat in android studio as any errors in the app are displayed there as well as any processes that occur when interacting with the app. Since there is the existence of unique tags as well we can detect where the issues are being caused in the code by referring to the tag.

Layout - Linear layout user for recipe list top down , ConstraintLayout used in adding ingredients forms. Constrained view layout due to the complexity of the UI views

Resources - We have a couple of different resources in our app mainly depicted by small emoticons that can be seen on the navigation bar, to the right of each recipe, and the top right of the app. Each emoticon when pressed serves a different purpose such as the three emoticons on the navigation bar which would redirect the user to a different page corresponding to what they pressed. The favorite button depicted by the heart alongside each recipe can be pressed by the user to favorite a recipe and can also be pressed again to unfavorite it.

Resource qualifier - According to the requirements provided with the project we were instructed to provide at least three resource qualifiers which in our case involved language change, orientation change, and the change between a light or dark mode. The change of language can be seen when we change the language of the phone through the phone's settings. When altered the language of our app is changed to the corresponding language chosen. The three languages we choose to use are English, Spanish, and Hindi. The second resource qualifier can be seen when we change the orientation of the device since the structure of our app is altered depending on whether the device is vertical or horizontal. Finally, we also included the action of switching

between both a light mode and dark mode of our app. The app on initial startup will be set to light mode, but if the user would prefer a darker tone on the app they can press the moon image on the top right to change it. Alternatively if they want to switch back to light mode they simply have to click the sun image that appears when the app is set to dark mode on the top right.

Persistence - Throughout our app there exists many examples of persistence that can be found. The 3 user preferences we used using the Datastore package are [navigate to Datasource.kt] Search, view count, and dark mode. For the local database we are using a sqlite room where we are saving favorite information and this data persists even if we kill and relaunch the app. So favorite information will not be lost. (Show the AppInspection -> favorite recipe DB)
The app also saves the search history after the user uses the feature which can be seen underneath the search bar. We used datastore saveSearchHistory() to do this task as it allows for the history to remain despite closing the app.
Finally, the app also saves viewCount number or the number of views and the current state of whether the app is set to dark mode or light mode.

User Interfacer - The two fragments when we rotate the screen we can see two layout fragments both the left and right SearchFragment.

fragment_home shows the recycler view and fragment_search

home and search use the same layout

RESTful Interactivity - When the user is on the home and goes to the add item button and submits all entered data we use POST API to add the recipe on server and after that we can search this recipe on the search section which uses GET API. Also on the Home Screen we are using the same GET API. We can show the app is cloudified by turning off wifi and showing no data.myApi.kt files shows the GET and POST calls for the retrofit API