Gopi Nelluri – gng75@umsystem.edu

Git Hub Link – https://github.com/gopinelluri9/demo_remote_repository/tree/main/MobilePart

Srikanth Gude – sgcnm@umsystem.edu

Git Hub Link - https://github.com/gudesrikanth/webcourse/tree/main/MobilePart

Video Link – https://www.youtube.com/watch?v=KnxuEEEs408

Presentation Github Link - https://github.com/gopinelluri9/demo_remote_repository/tree/main/MobilePart/ICP%20Presentation%20Two

ICP Presentation II

Introduction:

To complete our ICP's, we employed the following technologies – Android Studio, Java, XML

We have completed the following tasks as part of the ICP's Assignments. They are,

a) Login and Logout in Android App
b) Pizza Ordering App
c) Fetching User details from GitHub using RESTful services
d) Text to Speech Recognition

a) Login and Logout in Android App:

In this Task, we need to create a Login and Logout page in Android APP.

Procedure:

Create an empty Empty activity for the app and name it with the project name. After creating an empty blank activity and naming our project LoginApp, we selected Java as the programming language as previously mentioned and clicked on the finish button, which results in the creation of a Login app.

Login:

activity main.xml has been modified to include two new Text fields and a button for the Login process.

First Text Field: Username - We've included a Text Field with the username, input type @+id/usename is the id of the text field.

Second Text Field: Password - In the second text field we've added, the password is the second one. The id for the text box is @+id/password.

Login Button:

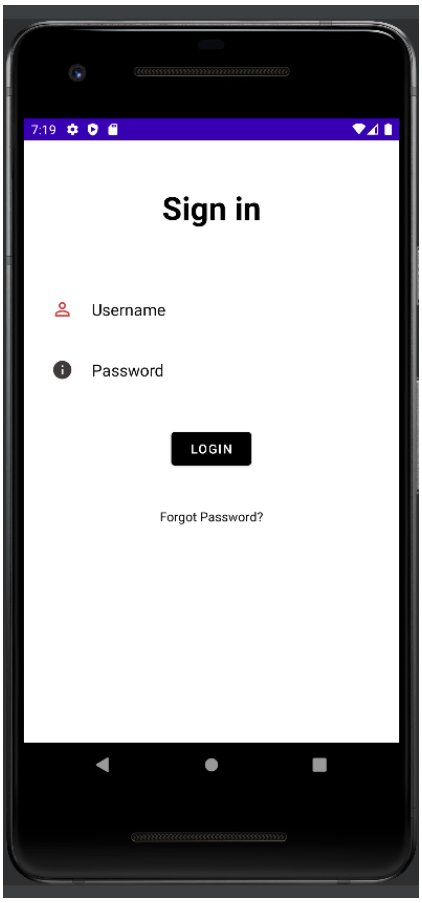
Like this, we have added another element for login button, with the text Login and the onClick event named onClick.

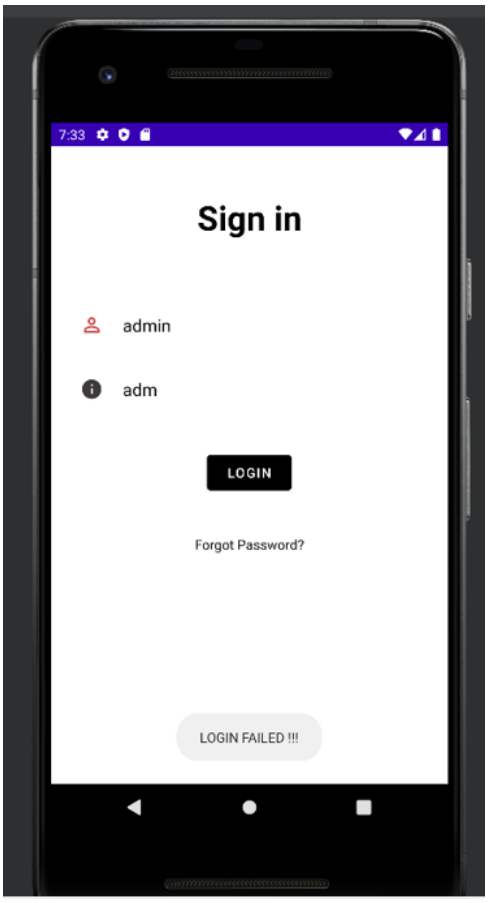
onClick Activity:

Functionality: When a user clicks on the login button, this function validates the login credentials and will check for correct username & password. We'll use getText() to get the username and password. And validates the inputs, if the validation is successful, proceed to HomeActivity.class.

Output:

We have run the application on the virtual simulator where it has been deployed and is running. The UI Screen of the app looks like this:





activity_home.xml:

After entering the successful login, **activity_home.xml** has a Text field and a Button. Logout Button: This button enables the user to get logged out of the screen. The click functionality of the button is **onLogoutClick**.

HomeActivity.java:

When the method is executed then it should navigate the user to the login page which is present in MainActivity.java. This function has Viewed as a parameter that has button id as a value.

At the end, we covered how to create an Android application using the Android studio, java, and xml in this ICP. Finally, we have successfully designed and created an Android application. We have no serious concerns when doing this ICP.

b) Pizza Ordering App:

We will utilize Android Studio to construct a Pizza Ordering App in this task.

Process:

ORDER PAGE => activity_main.xml

This page contains a text field, three check boxes in Pizza Type, two check boxes in Toppings, and two buttons to increment and decrease the quantity of pizza **activity_main.xml.**

First Text Field: Name:

Here we have added an Edit Text Field with input type as userInput and we have also. The id is of the text field is @+id/user_input.

Type of Pizza Check Boxes: Veggie, Chicken and Other

We have added three checkboxes one is the veggie selection, second is Chicken selection and other selection. @+id/veggie_checked is the id of the Veggie Checkbox, and @+id/chicken_checked is the id of the Chicken Checkbox.

Check Boxes: Mushroom and Extra Cheese (Topping)

We have added two checkboxes one is the Mushroom selection, and the other is for Extra Cheeses selection. @+id/mushroom_checked is the id of the Mushroom Checkbox, and @+id/extra_cheese_checked is the ID of the Extra Cheese Checkbox.

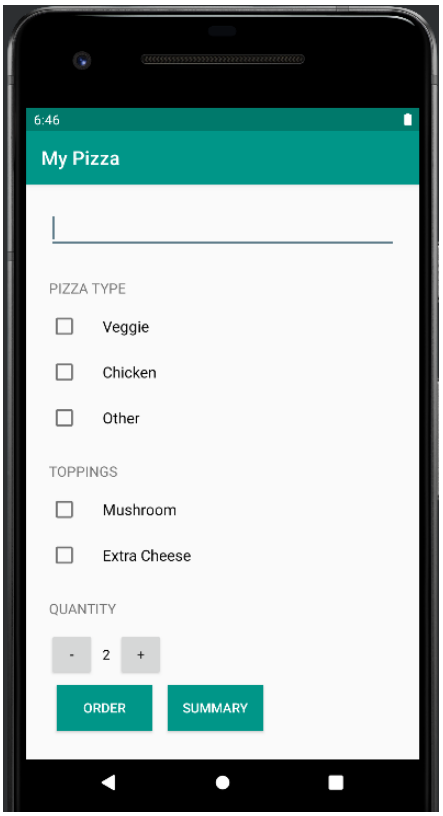
Two Buttons: Increment and Decrement Quantity:

Then you need include two buttons in the file. One is for increasing the quantity of pizza, while the other is for decreasing the quantity of pizza.

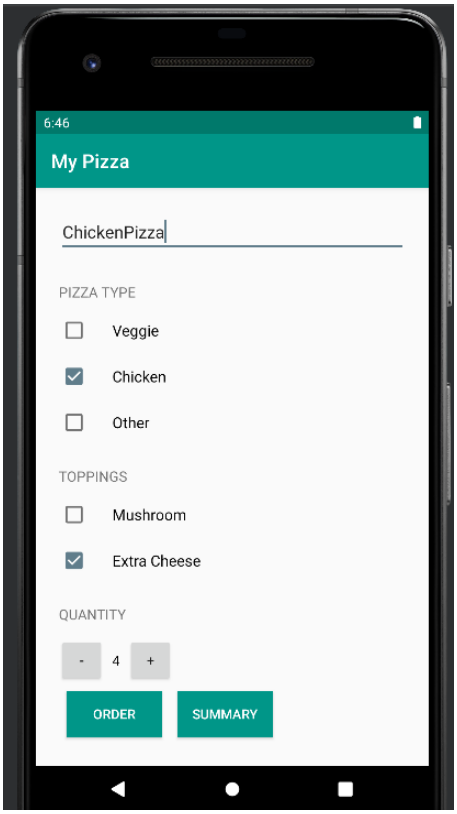
Two Buttons: ORDER and SUMMARY

When the order button is clicked, it enables the user to send the email to the intended recipient more quickly. The Summary button directs the user to a new page that contains the order summary, an image of a pizza, and a GOTO ORDER button. Summary buttons have the id @+id/buttonsummary, which stands for "summary."

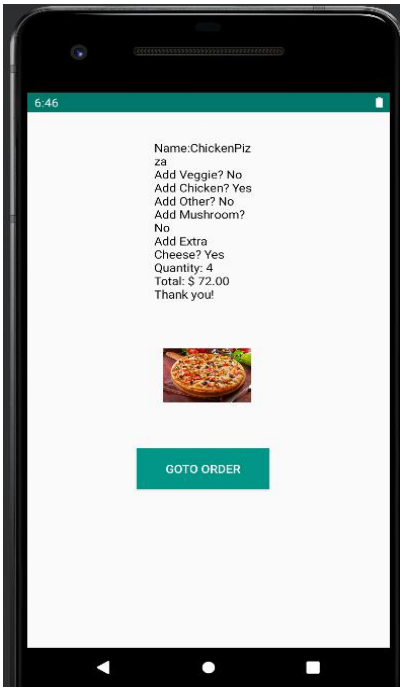
The UI functionality of the app looks like the following images,



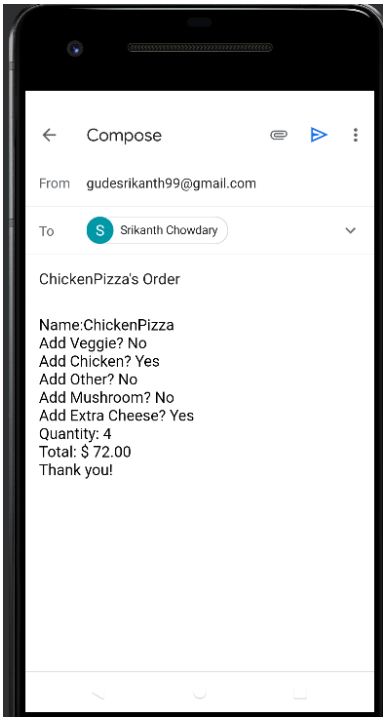
HOME Page



User Input



Summary Page



Clicking on Order

MainActivity.java: The Functionality of the app is described in this file.

SumbitOrder: The functionality of submitOrder is to allow users to send email with the

order description along with the subject of the mail.

sendEmail: The functionality of the sendEmail is to set the Subject of the mail, body of the mail, and recipient of the mail and send the mail to the recipient.

submitSummary(): The functionality of the submitSummary is to navigate to the new page to summary Screen where it will display the description of the pizza order.

Summary Layout: In this we have created a linear layout with a textview, an image, and a button on the screen.

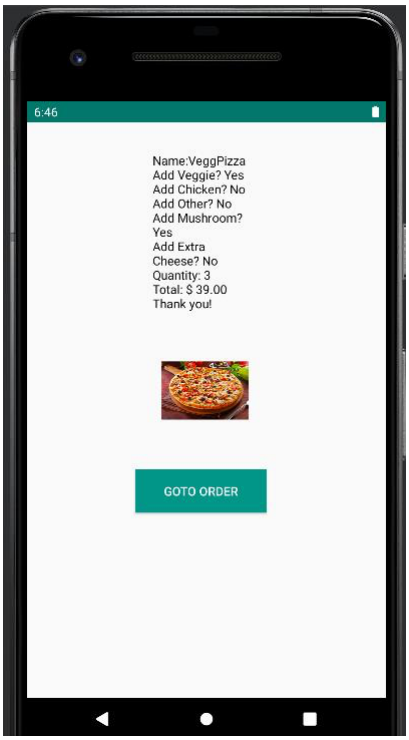
Goto Order Button: When the Goto Order button is clicked, the screen is redirected to the order screen, and navHome is the value supplied to the onclick function, therefore functionality for the same function is written in the java file.

Summary.java: The onCreate function of the Summary class is overridden to get the string message, i.e., the order description.

homeNav: When the Goto Order button is clicked, an intent is created using this and MainActivity parameters.

The UI for the Summary screen as below,

The order summary screen shows the kind of toppings, the quantity of pizzas requested, the total cost, and the user's name, among other things. When you click the Goto Order button, you'll be sent to the order page.



c) Fetching User details from GitHub using RESTful services:

We were requested to use Android Studio to print each GitHub user's ID and User Name, as well as any other information we gathered, from the website https://api.github.com>.

Process: activity_main.xml

We added NestedScrollView for nested scrolling from different positions of view change and will Handle scrolling response to up or down scrollings. And @+id/textView will displays ‘Hello world’ text.

MainActivity.java:

The Functionality of app is defined in this file. findViewById method will find the view by given id. Constructed Retrofit object, and retrofit class is used to retrieve and upload data via REST Client from GitHub service interface and it will provide authorization to access data requested by user, with help of object we can call ApiCollection.class interface and it will retrieve data and stored in list structure.

onResponse:

onResponse function will display the details if response is successful, by using list interface we store retrieved data in structured format and for loop with condition of users will display in organized order.

onFailure:

onFailure function will throw an error message if response is not retrieved and Toast will display the message.

User.java:

user class will Serialize the name and parse from the format by @SerializedName annotation and getId() will return id and getUserName() will return username.

ApiCollections.java:

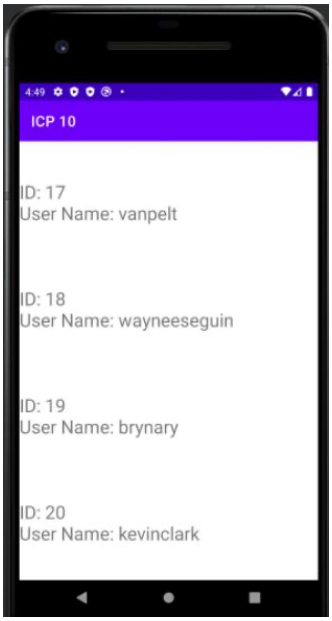
snippet @GET annotation will request HTTP API to get userdata by response.

AndroidManifest.xml:

we need to specify internet permissions in the Androidmanifest.xml file in the uses-permission xml tag, because if we do not mention internet permissions, the response data will not be coming from the url endpoint, and we will not be able to display User information.

Output:

When we run the code, the display screen will show a list of github users from the endpoint, as well as information about each person such as their userid and username.



UI SCREEN **UI SCROLL DOWN SCREEN**

At the end, we covered how to create an Android application using the Android studio, RESTful services, Retrofit ListView, and Adapter in this ICP. Finally, we have successfully designed and created an Android application. We have no serious concerns when doing this ICP.

d)Text to Speech Recognition:

In this task, we will create a Text to Speech Recognition program that will say what we have supplied as a text input.

Process:

activity_main.xml:

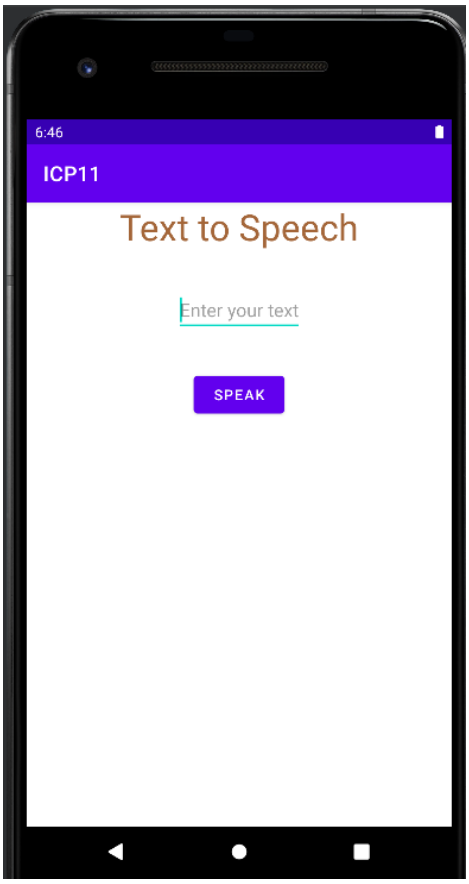
In above code snippet we have created TextView widget to display the text in UI screen and given id as '@+id/textViewTTS'. EditText is used to enter text into input box and given id as '@+id/editTextTTS'. And Button is used to create clickable speak button and given id as '@+id/btnTT'.

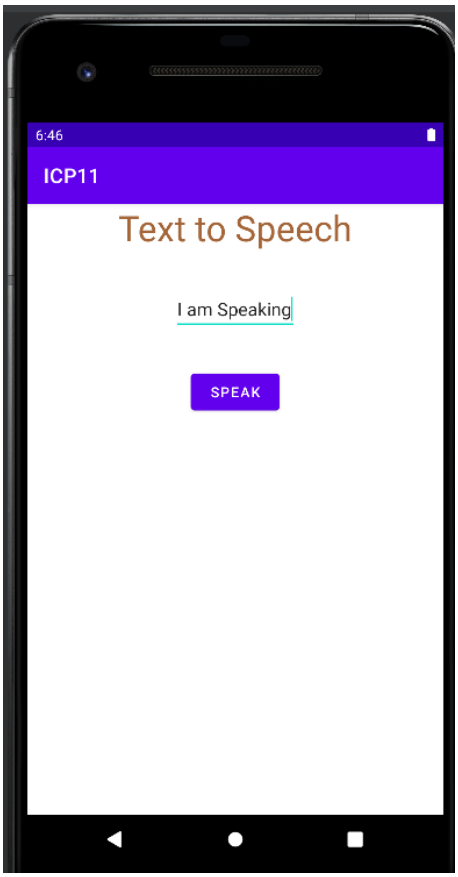
MainActivity.java:

OnCreate method bundle instance will create the user interface and savedInstanceState will hold the information from previously saved instance. And created variables and find their id by findViewById for input text and speak button, setOclickListener attribute will listen to click element.

TextToSpeech is initialized using OnInitListener and OnInit() method will check for if condition by taking parameter status and if it is equals to success it will proceed speak by verifying the text given in input box and if not matched with status and it will move to else condition and print log message.

Output:





UI SCREEN UI SCREEN INPUT

Conclusion:

At the end, we covered how to create an Android application using OnCreate, savedInstanceState, setOclickListener, findViewById, TextToSpeech, OnInitListener.