TEAMMATES:

Pavan Kumar Jonnadula - 16324822 - pvjb6c@umsystem.edu

Vamsi Alapaty - 18230326 - vanny@umsystem.edu

Priya Varsha Tarlapally – 16332852 – ptf2x@umkc.edu

ICP10 – Report

Programming Tools:

GitHub, VS Code

Source Code:

Pavan – 16234822
https://github.com/JVSPAVAN/WebMobile-2022Spring/tree/main/Mobile/ICP10

Vamsi – 18230326
https://github.com/VamsiAlapaty/Web-Mobile-Spring2022/tree/mobile

Priya - 16332852
https://github.com/Privar11/Web_Mobile_Programming_ICP/tree/main/Web_Programming

Objective:

This task concentrates on understanding the concept of Android platform. And create a mobile application which displays the users data which is received from the API.

Task:

Creating an Android application using React-native.

Steps done to achieve the above task:

1. Create a java file which consists of user data structure.
2. Integrate the API into the application.
3. Create a main activity screen to display the user data.
4. Consider all the scenarios of the GitHub API.

User.java screen:

Same as the java application in the backend, here also pojo class is created to structure the data. Two private variables are declared like ‘id’ and ‘username’. Getters and Setters are

also applied for the above variables and encapsulate the data. Using this java class the data is set into this datatype.

```
public class User {
    private int id;

    @SerializedName("login")
    private String userName;

    public int getId() { return id; }

    public String getUserName() { return userName; }
```

API Integration:

An Interface is created to receive the data from the API. This data is collected in the form of a list. And the datatype for this list is the 'user' which was created using the 'user.java' file.

```
public interface AppCollection   {
    @GET ("users")
     Call<List<User>>getData();

}
```

Display of data in the MainActivity screen:

For the API integration, a 'retrofit' package is used to integrate the API calls into the Andorid application. This package manages the creation of http requests. Activities like response handling is also done using 'retrofit' package.

Using the 'OnCreate()', a http call is created to call the API in the backend.

```
protected void onCreate(Bundle  savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textview=findViewById(R.id.textview);
    Retrofit retrofit=new Retrofit.Builder()
            .baseUrl("https://api.github.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build();
```

Using the ‘OnResponse()’ method, the response is obtained from the API call. And this obtained data is set into a list format. And this is in a ‘user’ datatype.

```
public void onResponse(Call<List<User>> call, Response<List<
    if(response.isSuccessful()){
        List<User> users= response.body();
        for(User user:users){
            String data="";
            data +="ID: " + user.getId() +"  ";
            data +="User Name : " +user.getUserName()+"\n";
            textview.append(data);
        }
    }
```

If a http call is failed, this scenario is handled using ‘OnFailure()’ method. This info can be captured by displaying in a text message or toast.

```
public void onFailure(Call<List<User>> call, Throwable t) {
    Toast.makeText( context: MainActivity.this, text: "Data Failed",Toast.LENGTH_SHORT
}
```

Conclusion:

This react-native app can be viewed in the android emulator or any physical device. As this is a react-native app, the users can also be able to operate in the IOS devices as well. Users data is displayed in a list format on the screen. The output is displayed as follows.