MOBILE – ICP PRESENTATION 2

Dharma Teja K

Email: dkbmy@umsystem.edu

Avinash Reddy T

Email: atfkh@umsystem.edu

Presentation Link: WebDevCourse/ICP Presentation Two.pptx at main · avinashreddy3/WebDevCourse (github.com)

GitHub link: WebDevCourse/MobilePart/ICP Presentation 2 at main · avinashreddy3/WebDevCourse (github.com)

Video Link: https://youtu.be/JQbjO65Wl6s

ICP 8

In this ICP, we are going to

Create the basic Login application, which consists of:

- Username and password should be included in the main action (make appropriate changes for the login app!) with login button.
- If the login credentials are genuine, the screen should navigate to the welcome screen when the 'Login' button is clicked, logout button after login. Otherwise, the app should show the user an appropriate message.
- When clicked on the logout button, the screen should navigate to the login screen

Below was the snippet of main_activity java code , written importing the packages required and used inheritance concepts as well.

- In the mainactivity.java file, we used java code for implementing the functionality of app.
    - We have used textview fields
    - Buttons
    - Edittext fields

We have used editView, Textview, edittext, button. And initialized as the private variables.

```java
//importing all the required packages.
package com.example.basicapp;
import androidx.appcompat.app.AppCompatActivity;

//importing the android packages.
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

//inheritance concept complimented using extends keyword
public class MainActivity extends AppCompatActivity {

    private EditText name;
    private EditText pass;
    private TextView info;
    private Button login;

//    The ability for a subclass to override a method allows a class to inherit from a superclass
//    with "near enough" behavior and then modify it as needed.
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

// code to check the credentials entered
    public void checkCredentials(View v)
```

Here we check and validate the inputs fields entered by the users.

We also deal with the visibility methods in the java, and display the error message if the entered credentials are wrong and also code to redirect user to homepage.

```java
        String password = pass.getText().toString();

        boolean validationFlag = false;
        //Verify if the username and password are not empty.
        if(!userName.isEmpty() && !password.isEmpty()) {
//            as for the sample test, give credentials as below
            if(userName.equals("avinash") && password.equals("umkc")) {
                validationFlag = true;

            }
        }

//        displaying the error message if entered wrong credentials
        if(!validationFlag)
        {
            info.setVisibility(View.VISIBLE);
            Toast.makeText( context: MainActivity.this,  text: "You Entered wrong Credentials! Please try again",
                    Toast.LENGTH_LONG).show();
        }
        else
        {
            //This code redirects the from login page to the home page.
            Intent redirect = new Intent( packageContext: MainActivity.this, HomePage.class);
            startActivity(redirect);
        }

    }


//    navigating to signup page
    public void SignUp(View v)
```

Java Code screenshot of homepage.

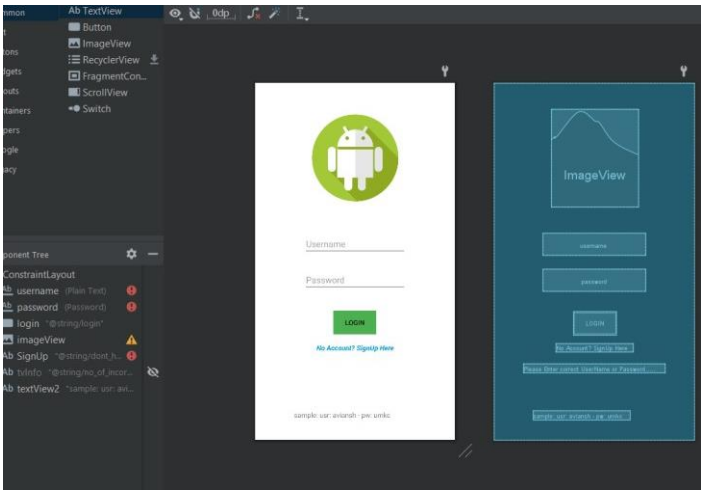
```java
//importing all the required packages.
package com.example.basicapp;
//importing the android packages.
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

//inheritance concept complimented using extends keyword
public class HomePage extends AppCompatActivity {

//    overriding the methods of parent class
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_page);
    }


//    navigating to logout page
    public void logOut(View v)
    {
        Intent i = new Intent( packageContext: HomePage.this, MainActivity.class);
        startActivity(i);
    }
}
```

Java Code screenshot of signup page.

```java
//importing all the required packages.
package com.example.basicapp;
//importing the android packages.
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

//inheritance concept complimented using extends keyword
public class SignUpPage extends AppCompatActivity {

    //    overriding the methods of parent class
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up_page);
    }

    //    navigating to logout page
    public void logOut(View v)
    {
        Intent i = new Intent( packageContext: SignUpPage.this, HomePage.class);
        startActivity(i);
    }
}
```

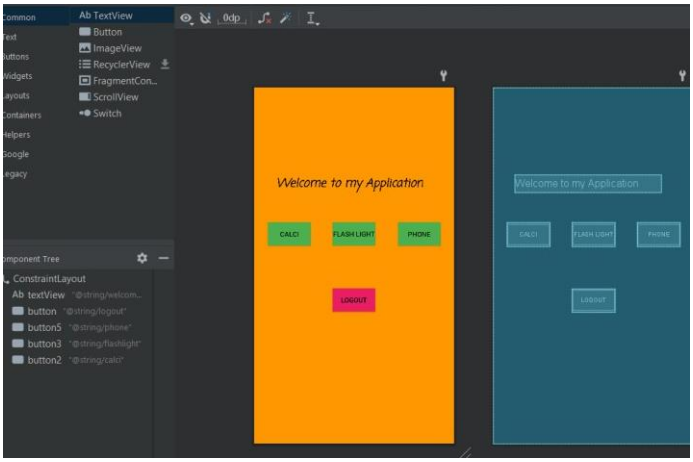
Below was the main login page designed with username and password fields with login button and text displayed below it.

Sample username and password is also displayed as a part of testing



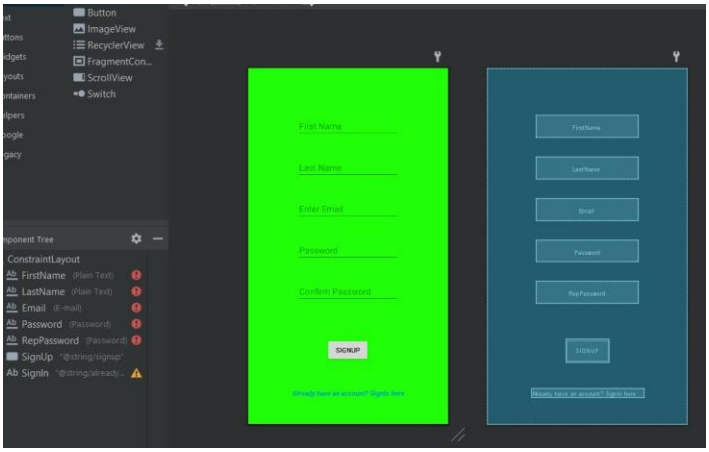

```
<Button
    android:id="@+id/login"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="checkCredentials"
    android:text="@string/login"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/passw
    app:layout_constraintVertical_bias="0.158"
    android:background="#4CAF50"/>

-    profile image in the login page-->
<ImageView
    android:id="@+id/imageView"
    android:layout_width="178dp"
    android:layout_height="200dp"
    android:contentDescription="@string/todo"
    android:src="@drawable/android"
    app:srcCompat="@drawable/android"
    tools:layout_editor_absoluteX="116dp"
    tools:layout_editor_absoluteY="83dp"
    tools:srcCompat="@drawable/android"
    app:layout_constraintBottom_toTopOf="@+id/usern
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    />

-    message displayed under the login button-->
<TextView
    android:id="@+id/SignUp"
```

Below is the welcome page which comes after logging in




```
<!--    home screen after login -->
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/welcome_to_basic_application"
    android:textColor="@android:color/black"
    android:textSize="24sp"
    android:textStyle="bold|italic"
    app:fontFamily="casual"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.411"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.258" />

<!-- button to logout -->
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#E91E63"
    android:onClick="logOut"
    android:text="@string/logout"
    android:textColor="#000000"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.603" />
```

Below was the signup page designed with username, password, email address fields with signup button and text displayed below it.



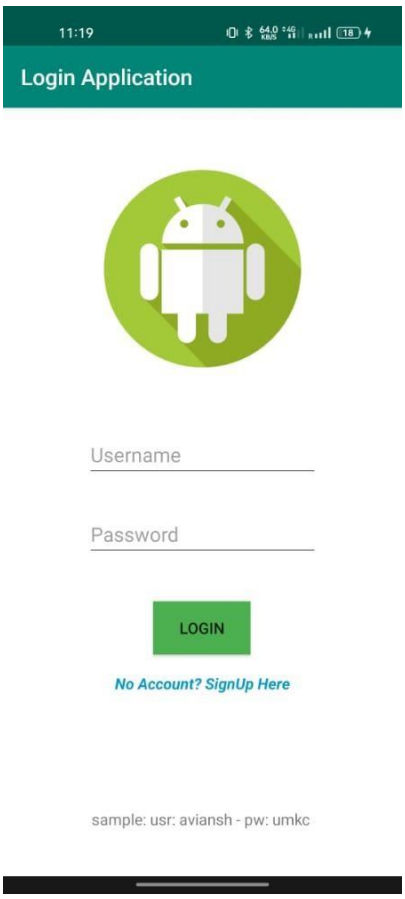

```
<!--    text field for password -->
<EditText
    android:id="@+id/Password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:hint="@string/password"
    android:importantForAutofill="no"
    android:inputType="textPassword"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Email" />

<!--    text field to double check the password entered
<EditText
    android:id="@+id/RepPassword"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:hint="@string/confirm_password"
    android:importantForAutofill="no"
    android:inputType="textPassword"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Password"

    button to signup-->
<Button
    android:id="@+id/SignUp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

Below are the output samples screenshots after the running the code and installing the apk on any of the devices.

First is our homepage of the application ,for user to enter the login credentials.

If the user doesn’t have the credentials, our next page is signup page as shown below where they can create the account and get their credentials for their further login.

Or else user will be redirected to ‘welcome to my application’ page after their successful login, as shown in the last snap below.



ICP 9

In this ICP, we are going to Develop

a pizza ordering application.

Features we added are:

- Submit button, pizza type, quantity, text field, and checkboxes are included in the main page of our application.
- Button to place an order.
- Summary button to view the order details.
- And main feature with our application is that we can share the order details externally with the order button itself.

Home Page:

Below is the screenshots for our application's home page and well commented.

below we have initialized all the global variables with their values as final.

```java
import ...

public class HomePage extends AppCompatActivity {

//  initialization the global variables with final values.

    private static final String MAIN_ACTIVITY_TAG = "MainActivity";
    final int COFFEE_PRICE = 5;
    final int WHIPPED_CREAM_PRICE = 1;
    final int CHOCOLATE_PRICE = 2;
    final int Jalapinoes = 1;
    final int Onions = 1;
    final int Olives = 1;
    final int Tomatoes = 1;
    final int Spinach = 1;
    final int Lettuce = 1;
    final int Bellpeppers = 1;
    String name = "";
    String order = "";
    String price1 = "";
    String quant = "";
    String totalmessage ="";
    int quantity = 3;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_page);
    }
```

Here we have written the code to get the input from the user and also the added extra items to the order and calculated the total price finally.

```java
//    This method is called when the order button is clicked.

    public void submitOrder(View view) {

        // get user input
        EditText userInputNameView = (EditText) findViewById(R.id.user_input);
        String userInputName = userInputNameView.getText().toString();

        // check if whipped cream is selected
        CheckBox whippedCream = (CheckBox) findViewById(R.id.whipped_cream_checked);
        boolean hasWhippedCream = whippedCream.isChecked();

        // check if chocolate is selected
        CheckBox chocolate = (CheckBox) findViewById(R.id.chocolate_checked);
        boolean hasChocolate = chocolate.isChecked();

        // calculate and store the total price
        float totalPrice = calculatePrice(hasWhippedCream, hasChocolate);

        // create and store the order summary
        String orderSummaryMessage = createOrderSummary(userInputName, hasWhippedCream, hasChocolate, totalPrice);
        Intent redirect = new Intent( packageContext: HomePage.this, SummaryPage.class);
        redirect.putExtra( name: "summary", orderSummaryMessage);
        redirect.putExtra( name: "order", order);
        redirect.putExtra( name: "price", price1);
        redirect.putExtra( name: "quant", quant);
        redirect.putExtra( name: "name", name);
        startActivity(redirect);
        // Write the relevant code for making the buttons work(i.e implement the implicit and explicit intents
```

Below snippet code written using java programming language to store the order details or summary message which can be shared externally via any platform like messaging app, email, etc.

And we also store the details in the application as well.

```
 chocolate is selected
ocolate = (CheckBox) findViewById(R.id.chocolate_checked);
Chocolate = chocolate.isChecked();

e and store the total price
Price = calculatePrice(hasWhippedCream, hasChocolate);

nd store the order summary
rSummaryMessage = createOrderSummary(userInputName, hasWhippedCream, hasChocolate, totalPrice);
new Intent(Intent.ACTION_SEND);
text/plain");
Intent.EXTRA_EMAIL  , new String[]{"sumanthmedavarapu628@gmail.com"});
Intent.EXTRA_SUBJECT,  value: "subject of email");
Intent.EXTRA_TEXT   , orderSummaryMessage);

tivity(Intent.createChooser(i,  title: "Send mail..."));
droid.content.ActivityNotFoundException ex) {
okeText( context: HomePage.this,  text: "There are no email clients installed.", Toast.LENGTH_SHORT).show();



boolToString(boolean bool) {
```

```
private void display(int number) {
    TextView quantityTextView = (TextView) findViewById(R.id.quantity_text_view);
    quantityTextView.setText("" + number);
}

/**
 * This method increments the quantity of coffee cups by one
 * @param view on passes the view that we are working with to the method
 */

public void increment(View view) {
    if (quantity < 100) {
        quantity = quantity + 1;
        display(quantity);
    } else {
        Log.i( tag: "MainActivity",  msg: "Please select less than one hundred cups of coffee");
        Context context = getApplicationContext();
        String lowerLimitToast = "Please select less than one hundred pizza";
        int duration = Toast.LENGTH_SHORT;
        Toast toast = Toast.makeText(context, lowerLimitToast, duration);
        toast.show();
        return;
    }
}

/**
```

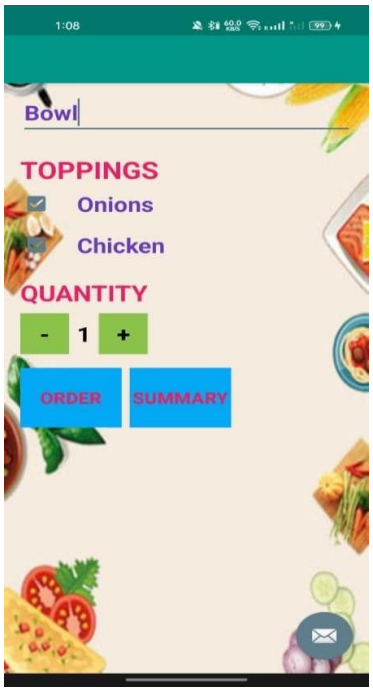
Summary Page:

Code screenshot below shows the summary page designed using the java code, to store the details of the order or the summary of the order, which can be viewed by using summary button in our application.

```
HomePage.java   SummaryPage.java
package com.webproject.hiresh.pizzaorder;

import ...

public class SummaryPage extends AppCompatActivity {

//  this page is used to display the summary of order.
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.summarypage);
        String summary = getIntent().getStringExtra( name: "summary");
        TextView summaryText = (TextView) findViewById(R.id.summary);
        summaryText.setText(summary);
    }

    public void goBack(View view) {
        Intent redirect = new Intent( packageContext: SummaryPage.this, HomePage.class);
        //redirect.putExtra("summary", orderSummaryMessage);
        startActivity(redirect);
    }

}
```

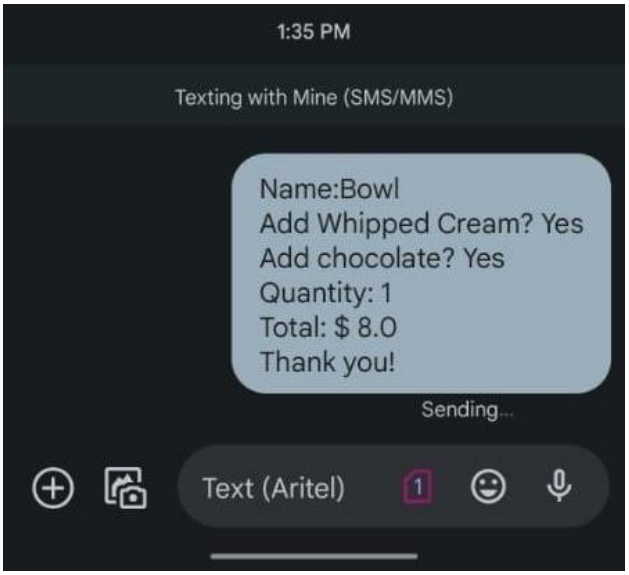
Output:

Finally we run our application by installing the apk version on our desired device. Below is the Output of our app's homepage view where there is an input field, toppings as an additionals to be added and two buttons.

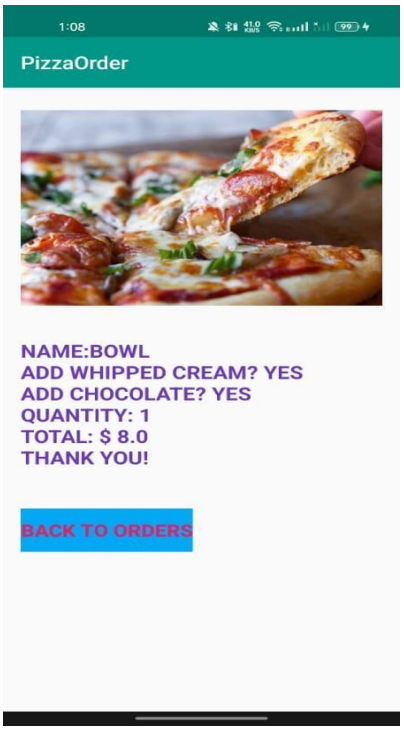


After placing an order we can share the details externally as said before.

Below was the sample where I shared details through messaging app, as a part of testing.



This is our Summary page where the order summary is stored and viewed by using the summary button on the homepage.



ICP 11

In this ICP, we are going to utilize android studio, java programming, xml languages for mobile application development,

To create an app with Text-to-Speech functionality which convert the text that is entered on the field into speech using android studio.

Procedure:

First we have created a new project with empty activity using android studio application.

This provides .java, .xml files which are used to convert the text entered by the user into speech.

1. Creating a TextView, button & EditText fields in android manifest.xml file.

We've added TextView, EditText, and Button fields to the 'activity main.xml' file to collect user input and transform it to voice.

```
<?xml version="1.0" encoding="utf-8"?>

<!--relative layout is view group which shows the child view in their relative positions-->
<!-- here we added top padding to our android app-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="65dp"
    tools:context=".MainActivity">
```

To all the fields, we have added height, weight, color, alignment properties and all ,as show in below snippets.

EditView:

To accept the input text from the user to convert it into speech

```
<!--    added one edittext field and created an id -->
    <EditText
        android:id="@+id/editTextTTS"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:hint="enter your text"
        android:layout_below="@+id/textViewTTS"
        android:layout_centerHorizontal="true"
        android:layout_margin="35dp"
        />
```

TextView:

This acts like main heading or title of our android application

```
<!--    added one text view feature with below parameters-->
    <TextView
        android:id="@+id/textViewTTS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Text to Speech"
        android:textColor="#ff1c1c"
        android:textSize="35dp"
        android:layout_centerHorizontal="true"
        />
```

Button:

The button field assisted in the conversion of written text to voice. When the "Speak" button is triggered, the text entered is converted into speech.

```
<!--    added a user interactive speak button -->
    <Button
        android:id="@+id/btnTTS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editTextTTS"
        android:layout_centerHorizontal="true"
        android:text="Speak"
        />

</RelativeLayout>
```

2. (Adding Text-to-Speech capabilities to turn text into speech): MainActivity.java:

First, we had given the id's for each of the fields we created above.

And create the variables for the three text **Text-to-Speech** functionalities which are EditView, Button, TextToSpeech objects as shown below:

After we've finished creating the object, we've implemented validations for both success and failure scenarios.

If no errors are identified, it will execute properly and pick "Locale.US" as the language of speech. Validations for language selection have also been included.

If the specified language is incorrect or not functioning correctly, an error notice will appear. Otherwise, the text will be successfully transformed. This is shown in below screenshot.

```
activity_main.xml   MainActivity.java
public class MainActivity extends AppCompatActivity {

    EditText editText;
    Button btn;
    TextToSpeech tts;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editText= findViewById(R.id.editTextTTS);
        btn= findViewById(R.id.btnTTS);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                tts= new TextToSpeech( context: MainActivity.this, new TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status== TextToSpeech.SUCCESS){
                            int result= tts.setLanguage(Locale.US);
                            if (result== TextToSpeech.LANG_NOT_SUPPORTED || result== TextToSpeech.LANG_MISSING_DATA){
                                Log.e( tag: "message",  msg: "Language is not supported");
                            }else {
                                speak();
                            }
                        }else {
                            Log.e( tag: "message",  msg: "TTs is not supported");
```

And the next main feature is speak() method/ function which is to be triggered when clicked on the speak button on the home page.

Here we have also added the setSpeechRate() which is used to control the rate of speed on voice.

And we have given the two scenarios here as well,

-If the text entered is empty, display a message to user to enter the text to speak -

Else convert the text entered to voice.

The "stop()" method was also used to interrupt the current sentence (whether it was being played or stored to a file) and discard the remaining sentences in the queue.

We also utilized the shutdown() method, which is intended to free up native TextToSpeech resources.

```java
    void speak() {
        String s = String.valueOf(editText.getText());
        tts.setSpeechRate(0.1f);
        if (isEmpty(s)){
            tts.speak( text: "please enter something to speak", TextToSpeech.QUEUE_ADD,  params: null);
        }else{
            tts.speak(s, TextToSpeech.QUEUE_ADD,  params: null);
        }
    }

    @Override
    protected void onDestroy(){
        super.onDestroy();
        tts.stop();
        tts.shutdown();
    }
}
```

Output pages on the android device:

This is homepage of Text-To-Speech application

We have entered the text to speech, as shown below, to test. When clicked on 'SPEAK', you can hear the voice.