**3. Implementation**

**3.1 Data Dictionary**

**3.1.1 USERS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Users | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1 | U\_id | Int | Pk |  |
| 2 | U\_name | Varchar | Unique | 240 |

**3.1.2 REGISTRATION DETAILS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RDetails | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1 | C\_Id | Int | Pk |  |
| 2 | C\_Name | Varchar |  | 240 |
| 3 | C\_Email | Varchar |  | 240 |
| 4 | C\_Contact\_no | Int |  |  |
| 5 | C\_Username | Unique |  | 240 |
| 6 | C\_Password |  |  | 240 |
| 7 | C\_Photo | Varchar |  | 240 |
| 8 | U\_id | Int | Fk |  |

**3.1.3 CUISINES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cuisine | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1 | Cuisine\_Id | Int | Pk |  |
| 2 | CuisineName | Varchar | Unique | 240 |
| 3 | CuisineImage | Varchar |  | 240 |

**3.1.4 ITEM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1 | It\_id | Int | Pk |  |
| 2 | It\_name | Varchar | Unique | 240 |
| 3. | It\_Rate | Numeric |  |  |
| 4. | It\_Image | Varchar |  | 240 |
| 5. | It\_Description | Varchar |  | 240 |
| 6 | Cuisine\_id | Inr | Fk |  |

**3.1.5 TABLE DETAILS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1 | T\_id | Int | PK |  |
| 2 | Status | Boolean |  |  |

**3.1.6 ORDER DETAILS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1 | O\_id | Int | PK |  |
| 2 | Table\_id | Int | Unique,PK |  |
| 3. | Items | String Array |  |  |
| 4. | Total Amount | Int |  |  |

**3.1.7 INVOICE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Invoice | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1. | i\_Id | Int | Pk |  |
| 2. | i\_customer\_name | Varchar |  | 240 |
| 3. | i\_orderdate | Varchar |  | 240 |
| 4. | i\_total | Float |  |  |
| 5. | i\_tax | Float |  |  |
| 6. | i\_discount | Float |  |  |
| 7. | i\_netamount | Float |  |  |
| 8. | i\_paidamount | Float |  |  |
| 9. | i\_due | Float |  |  |
| 10. | i\_paymentmethod | Varchar |  | 240 |

**3.1.8 INVOICE DETAILS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Invoice\_Details | | | | |
| Sr. no | Field name | Type | Constraint | Size |
| 1. | inv\_Id | Int | Pk |  |
| 2. | i\_totalamount | Int |  |  |
| 3. | i\_id | Int | Fk |  |

**3.2 Code**

**3.2.1 activity\_login.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<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:background="@drawable/bck"  
 tools:context=".Login">  
  
 <TextView  
 android:id="@+id/tvloginadmin"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="20dp"  
 android:fontFamily="cursive"  
 android:text="Login"  
 android:textColor="@color/colorPrimaryDark"  
 android:textSize="50dp"  
 android:textStyle="bold" />  
  
 <EditText  
 android:id="@+id/etemailloginadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/tvloginadmin"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/emailiconadmin"  
 android:drawablePadding="5dp"  
 android:hint="Email"  
 android:textSize="22dp" />  
  
 <EditText  
 android:id="@+id/etpassloginadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etemailloginadmin"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="25dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/passiconadmin"  
 android:drawablePadding="5dp"  
 android:hint="Password"  
 android:inputType="textPassword"  
 android:textSize="22dp" />  
  
 <Button  
 android:id="@+id/btnloginadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etpassloginadmin"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginLeft="40dp"  
 android:layout\_marginTop="30dp"  
 android:layout\_marginRight="40dp"  
 android:background="@drawable/roundcornerbtn"  
 android:text="LOGIN"  
 android:textColor="#FFFFFF"  
 android:textSize="21dp" />  
  
 <TextView  
 android:id="@+id/tvforgotpassloginadmin"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/btnloginadmin"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="20dp"  
 android:text="Forgot Password?"  
 android:textColor="@color/colorPrimaryDark"  
 android:textSize="18dp" />  
  
 <TextView  
 android:id="@+id/tvnew"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentRight="true"  
 android:layout\_marginTop="13dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/newusericonadmin"  
 android:drawablePadding="4dp"  
 android:text="New User"  
 android:textColor="@color/colorPrimaryDark"  
 android:textSize="20dp"  
 android:textStyle="italic" />  
  
</RelativeLayout>

**3.2.2 Login.java**

package com.example.laptop.corasystem;  
  
import android.app.ProgressDialog;  
import android.content.Intent;  
import android.support.annotation.NonNull;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.auth.AuthResult;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.Query;  
import com.google.firebase.database.ValueEventListener;  
  
public class Login extends AppCompatActivity {  
  
 TextView tvforgotpassloginadmin, tvnewuserloginadmin;  
 EditText etemailloginadmin, etpassloginadmin;  
 Button btnloginadmin;  
  
 Utility utility;  
 DatabaseReference databaseReference;  
 ProgressDialog progressDialog;  
 FirebaseAuth logauth;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login*);  
  
 tvforgotpassloginadmin = findViewById(R.id.*tvforgotpassloginadmin*);  
 tvnewuserloginadmin = findViewById(R.id.*tvnew*);  
  
 etemailloginadmin = findViewById(R.id.*etemailloginadmin*);  
 etpassloginadmin = findViewById(R.id.*etpassloginadmin*);  
  
 btnloginadmin = findViewById(R.id.*btnloginadmin*);  
  
 utility = new Utility(Login.this);  
 databaseReference = FirebaseDatabase.*getInstance*().getReference("User");  
 progressDialog = new ProgressDialog(Login.this);  
 logauth = FirebaseAuth.*getInstance*();  
  
 btnloginadmin.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Validation va = new Validation();  
 boolean bl = true;  
  
 if (!va.*email\_Validation*(etemailloginadmin)) {  
 bl = false;  
 etemailloginadmin.setError("Enter Valid Email");  
 }  
  
 if (!va.*passwordValidator*(etpassloginadmin)) {  
 bl = false;  
 etpassloginadmin.setError("Enter Your Password");  
 }  
  
 if (bl) {  
 if (utility.checkInternetConnectionALL()) {  
  
 progressDialog.setTitle("Login User");  
 progressDialog.setMessage("Logining In...");  
 progressDialog.setCancelable(false);  
 logauth.signInWithEmailAndPassword(etemailloginadmin.getText().toString(), etpassloginadmin.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
  
 if (task.isSuccessful()) {  
 progressDialog.show();  
 final Query user = databaseReference.orderByChild("etemailregadmin").equalTo(etemailloginadmin.getText().toString().trim());  
 user.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 if (dataSnapshot.hasChildren()) {  
 for (DataSnapshot usersnapshot : dataSnapshot.getChildren()) {  
 User u = usersnapshot.getValue(User.class);  
 switch (u.getUsertype().toString().trim()) {  
 case "Chef":  
 if (u.isVerify()) {  
 startActivity(new Intent(Login.this, ChefHome.class).addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*));  
 finish();  
 progressDialog.dismiss();  
 } else {  
 utility.toast("You are not Verified !");  
 progressDialog.dismiss();  
 }  
 break;  
  
 case "Manager":  
 if (u.isVerify()) {  
 startActivity(new Intent(Login.this, ManagerHome.class).addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*));  
 finish();  
 progressDialog.dismiss();  
 } else {  
 utility.toast("You are not Verified !");  
 progressDialog.dismiss();  
 }  
 break;  
  
 case "Admin":  
 startActivity(new Intent(Login.this, AddCat.class).addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*));  
 finish();  
 progressDialog.dismiss();  
 break;  
  
 default:  
 utility.toast("User Not Found! Try Again..");  
 progressDialog.dismiss();  
 break;  
 }}}}  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }});  
 progressDialog.dismiss();  
 } else {  
 utility.toast(task.getException().getMessage());  
 progressDialog.dismiss();  
 }}  
 });  
 }}}  
 });  
 tvnewuserloginadmin.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i = new Intent(Login.this, Registration.class);  
 startActivity(i);  
 }  
 });  
 }  
  
 @Override  
 protected void onStart() {  
 super.onStart();

if (logauth.getCurrentUser() != null) {  
 progressDialog.show();  
 final Query user = databaseReference.orderByChild("etemailregadmin").equalTo(logauth.getCurrentUser().getEmail().toString());  
  
 user.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 if (dataSnapshot.hasChildren()) {  
 for (DataSnapshot usersnapshot : dataSnapshot.getChildren()) {  
 User u = usersnapshot.getValue(User.class);  
 switch (u.getUsertype().toString().trim()) {  
 case "Chef":  
 if (u.isVerify()) {  
 startActivity(new Intent(Login.this, ChefHome.class).addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*));  
 finish();  
 progressDialog.dismiss();  
 } else {  
 utility.toast("You are not Verified !");  
 progressDialog.dismiss();  
 }  
 break;  
  
 case "Manager":  
 if (u.isVerify()) {  
 startActivity(new Intent(Login.this, ManagerHome.class).addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*));  
 finish();  
 progressDialog.dismiss();  
 } else {  
 utility.toast("You are not Verified !");  
 progressDialog.dismiss();  
 }  
 break;  
  
 case "Admin":  
 startActivity(new Intent(Login.this, AddCat.class).addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*));  
 finish();  
 progressDialog.dismiss();  
 break;  
  
 default:  
 utility.toast("User Not Found! Try Again..");  
 progressDialog.dismiss();  
 break;  
 }}}}  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }});  
 progressDialog.dismiss();  
 }}}

**3.2.3 activity\_registration.xml**

<?xml version="1.0" encoding="utf-8"?>  
<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:background="@drawable/bck"  
 tools:context=".Registration">  
  
 <TextView  
 android:id="@+id/tvregadmin"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:fontFamily="cursive"  
 android:text="Registration"  
 android:textColor="@color/colorPrimaryDark"  
 android:textSize="50dp"  
 android:textStyle="bold" />  
  
 <EditText  
 android:id="@+id/etnameregadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/tvregadmin"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/usericonadmin"  
 android:drawablePadding="5dp"  
 android:hint="Name"  
 android:textSize="24dp" />  
  
 <EditText  
 android:id="@+id/etmobregadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etnameregadmin"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/mobiconadmin"  
 android:drawablePadding="5dp"  
 android:hint="Mobile No."  
 android:inputType="number"  
 android:maxLength="10"  
 android:textSize="24dp" />  
  
 <EditText  
 android:id="@+id/etemailregadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etmobregadmin"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/emailiconadmin"  
 android:drawablePadding="5dp"  
 android:hint="Email"  
 android:textSize="24dp" />  
  
 <EditText  
 android:id="@+id/etpassregadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etemailregadmin"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/passiconadmin"  
 android:drawablePadding="5dp"  
 android:hint="Password"  
 android:inputType="textPassword"  
 android:textSize="24dp" />  
  
 <EditText  
 android:id="@+id/etcpassregadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etpassregadmin"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginRight="10dp"  
 android:drawableLeft="@drawable/passiconadmin"  
 android:drawablePadding="5dp"  
 android:hint="Confirm Password"  
 android:inputType="textPassword"  
 android:textSize="24dp" />  
  
 <Spinner  
 android:id="@+id/spinnerregadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etcpassregadmin"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginRight="10dp"  
 android:textSize="30sp" />  
  
 <Button  
 android:id="@+id/btnregadmin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/etcpassregadmin"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginLeft="40dp"  
 android:layout\_marginTop="60dp"  
 android:layout\_marginRight="40dp"  
 android:background="@drawable/roundcornerbtn"  
 android:text="REGISTER"  
 android:textColor="#FFFFFF"  
 android:textSize="20dp" />  
  
</RelativeLayout>

**3.2.4 Registration.java**

package com.example.laptop.corasystem;  
  
import android.app.ProgressDialog;  
import android.content.Intent;  
import android.support.annotation.NonNull;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Spinner;  
import android.widget.TextView;  
  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.auth.AuthResult;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseAuthUserCollisionException;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
  
public class Registration extends AppCompatActivity {  
  
 EditText etnameregadmin, etmobregadmin, etemailregadmin, etpassregadmin, etcpassregadmin;  
 TextView tvregadmin;  
 Button btnregadmin;  
 Spinner spinner;  
  
 Utility utility;  
 DatabaseReference databaseReference;  
 FirebaseAuth firebaseAuth;  
 ProgressDialog progressDialog;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_registration*);  
  
 etnameregadmin = findViewById(R.id.*etnameregadmin*);  
 etmobregadmin = findViewById(R.id.*etmobregadmin*);  
 etemailregadmin = findViewById(R.id.*etemailregadmin*);  
 etpassregadmin = findViewById(R.id.*etpassregadmin*);  
 etcpassregadmin = findViewById(R.id.*etcpassregadmin*);  
  
 tvregadmin = findViewById(R.id.*tvregadmin*);  
  
 btnregadmin = findViewById(R.id.*btnregadmin*);  
  
 spinner = findViewById(R.id.*spinnerregadmin*);  
 String ut[] = {"Manager","Chef"};  
  
 ArrayAdapter<String> ar = new ArrayAdapter<String>(Registration.this,R.layout.*spinnerdesign*,ut);  
 ar.setDropDownViewResource(R.layout.*spinnerdesign*);  
 spinner.setAdapter(ar);  
  
  
 utility = new Utility(Registration.this);  
  
 databaseReference = FirebaseDatabase.*getInstance*().getReference("User");  
  
 firebaseAuth = FirebaseAuth.*getInstance*();  
  
 progressDialog = new ProgressDialog(Registration.this);  
  
 progressDialog.setTitle("User Registration");  
 progressDialog.setMessage("Please Wait..");  
 progressDialog.setCancelable(false);  
  
 btnregadmin.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 boolean bl = true;  
 Validation va = new Validation();  
  
 if (!va.*nullValidator*(etnameregadmin)) {  
 bl = false;  
 etnameregadmin.setError("Enter Valid Name");  
 }  
  
 if (!va.*email\_Validation*(etemailregadmin)) {  
 bl = false;  
 etemailregadmin.setError("Enter Valid Email");  
 }  
  
 if (!va.*phoneValidator*(etmobregadmin)) {  
 bl = false;  
 etmobregadmin.setError("Enter Valid Mobile Number");  
 }  
  
 if (!va.*passwordValidator*(etpassregadmin)) {  
 bl = false;  
 etpassregadmin.setError("Enter Valid Password");  
 }  
  
 if (!va.*passwordValidator*(etcpassregadmin))  
 {  
 bl = false;  
 etcpassregadmin.setError("Reenter Password");  
 }  
  
 if (!etpassregadmin.getText().toString().equals(etcpassregadmin.getText().toString()))  
 {  
 bl = false;  
 etpassregadmin.setError("Password Not Match");  
 }  
  
 if (bl) {  
 if (utility.checkInternetConnectionALL()) {  
  
 progressDialog.show();  
 firebaseAuth.createUserWithEmailAndPassword(etemailregadmin.getText().toString(), etpassregadmin.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
  
 if (task.isSuccessful()) {  
 String id = firebaseAuth.getCurrentUser().getUid();  
 User user = new User(etnameregadmin.getText().toString(), etmobregadmin.getText().toString(), etemailregadmin.getText().toString(),id,spinner.getSelectedItem().toString(),false);  
 databaseReference.child(id).setValue(user);  
 utility.toast("Registration Successful");  
 firebaseAuth.signOut();  
 progressDialog.dismiss();  
 Intent i = new Intent(Registration.this, Login.class);  
 startActivity(i);  
  
 } else {  
 if (task.getException() instanceof FirebaseAuthUserCollisionException) {  
 utility.toast("Email is Already Registered !!");  
 progressDialog.dismiss();  
 } else {  
 utility.toast(task.getException().getMessage());  
 progressDialog.dismiss();  
 }}}  
 });

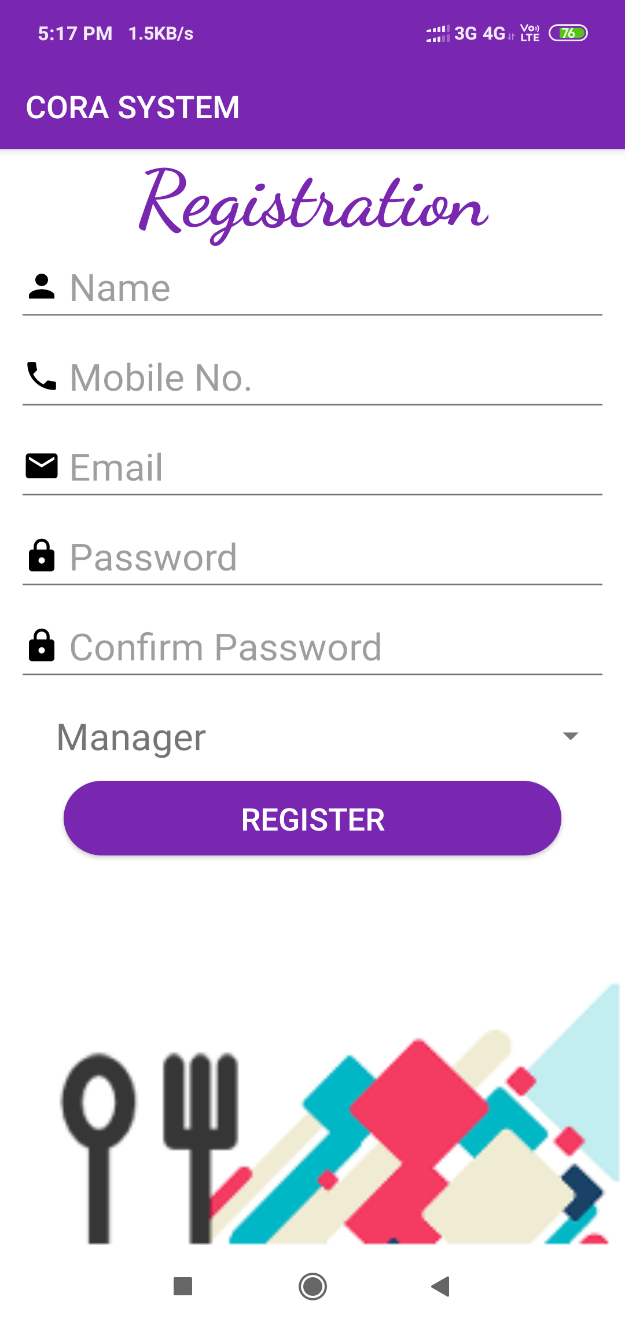
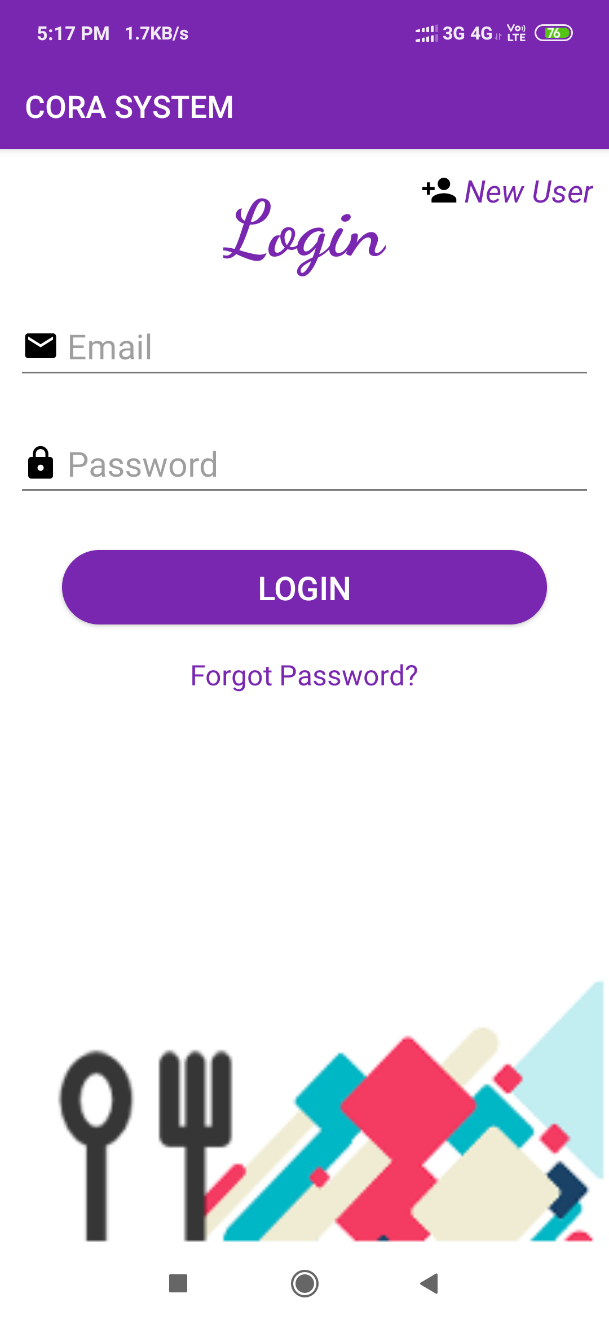
}}}  
 });  
}}

**3.3 Snapshots**

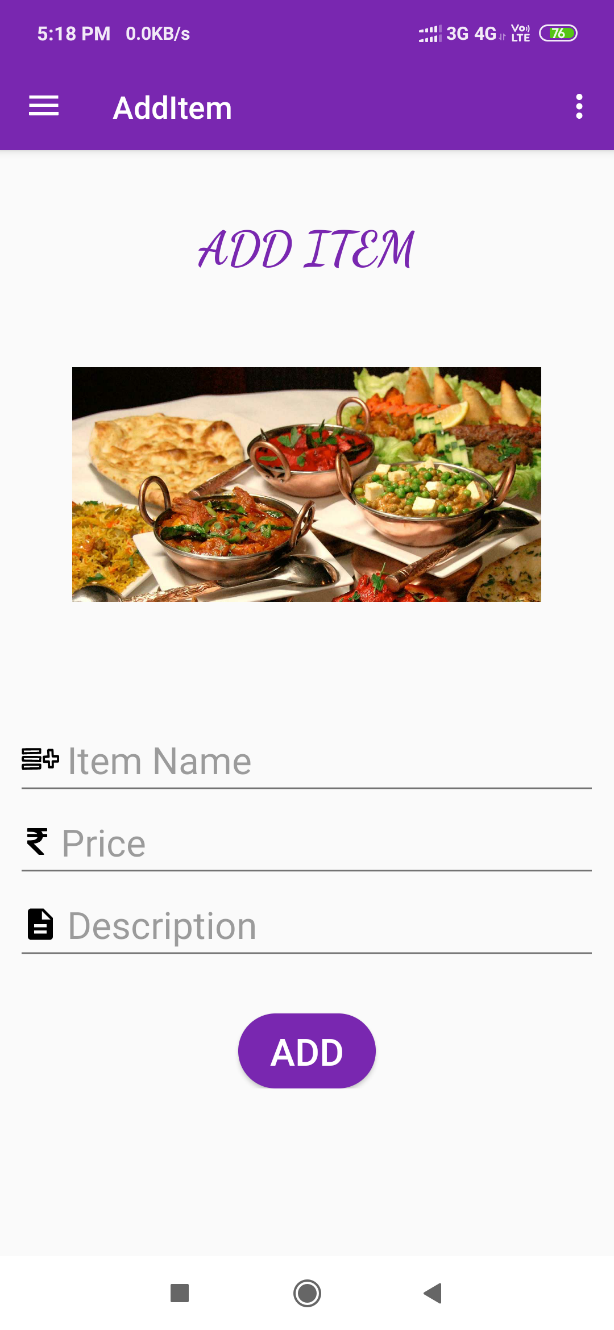
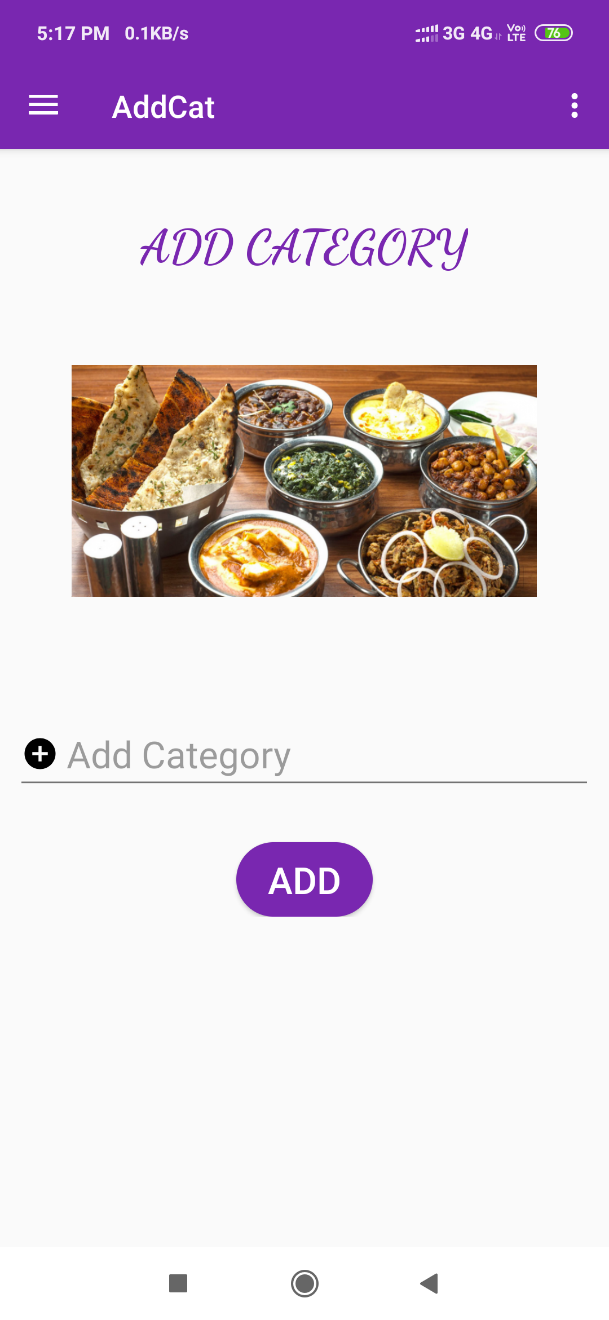
**Splash Screen:**



**Login: Registration:**



**Add Category: Add Item:**



**3.4 Testing**

**3.4.1 White-Box Testing**

White-box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, the software engineer can derive test cases that (1) guarantee that all independent paths within a module have been exercised at least once, (2) exercise all logical decisions on their true and false sides, (3) execute all loops at their boundaries and within their operational bounds, and (4) exercise internal data structures to ensure their validity. A reasonable question might be posed at this juncture: "Why spend time and energy worrying about (and testing) logical minutiae when we might better expend effort It is not possible to exhaustively test every program path because the number of paths is simply too large. White-box tests can be designed only after a component-level design (or source code) exists. The logical details of the program must be available. Ensuring that program requirements have been met?" Stated another way, why don't we spend all of our energy on black-box tests? The answer lies in the nature of software defects:

• Logic errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed. Errors tend to creep into our work when we design and implement function, conditions, or controls that are out of the mainstream. Everyday processing tends to be well understood (and well scrutinized), while "special case" processing tends to fall into the cracks.

• We often believe that a logical path is not likely to be executed when, in fact, it may be executed on a regular basis. The logical flow of a program is sometimes counterintuitive, meaning that our unconscious assumptions about flow of control and data may lead us to make design errors that are uncovered only once path testing commences.

• Typographical errors are random. When a program is translated into programming language source code, it is likely that some typing errors will occur. Many will be uncovered by syntax and type checking mechanisms, but others may go undetected until testing begins. It is as likely that a typo will exist on an obscure logical path as on a mainstream path. Each of these reasons provides an argument for conducting white-box tests. Black box testing, no matter how thorough, may miss the kinds of errors noted here. White box testing is far more likely to uncover them.

**3.4.2 Black-Box Testing**

Black-box testing, also called behavioral testing, focuses on the functional requirements of the software. That is, black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black-box testing is not an alternative to white-box techniques. Rather, it is a complementary approach that is likely to uncover a different class of errors than white-box methods. A class of loops you can’t test unstructured loops effectively. Black-box testing attempts to find errors in the following categories: (1) incorrect or missing functions, (2) interface errors, (3) errors in data structures or external data base access, (4) behavior or performance errors, and (5) initialization and termination errors. Unlike white-box testing, which is performed early in the testing process, black box testing tends to be applied during later stages of testing. Because black-box testing purposely disregards control structure, attention is focused on the information domain. Tests are designed to answer the following questions:

• How is functional validity tested?

• How is system behavior and performance tested?

• What classes of input will make good test cases?

• How are the boundaries of a data class isolated?

• What data rates and data volume can the system tolerate?

• What effect will specific combinations of data have on system operation?

By applying black-box techniques, we derive a set of test cases that satisfy the following criteria: (1) test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing and (2) test cases that tell us something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand. We have used Black Box testing in our project.