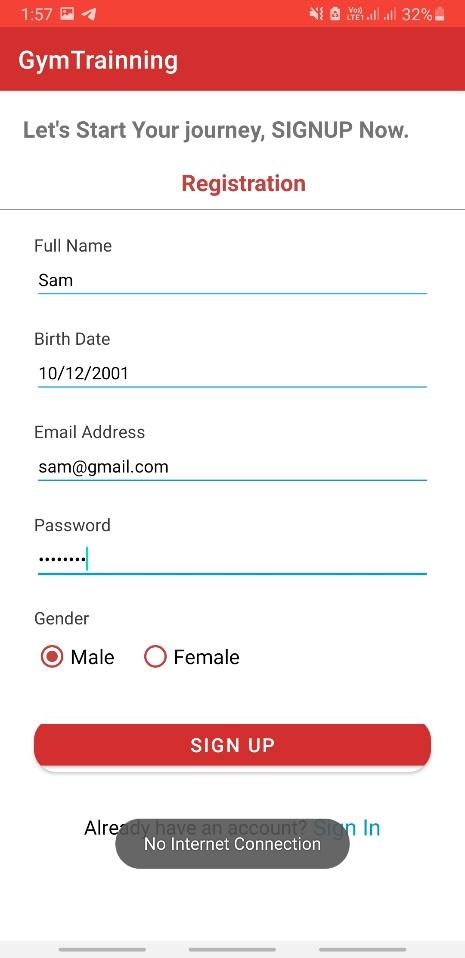
Criteria C: Development

**1. If…else to check Internet Connection**

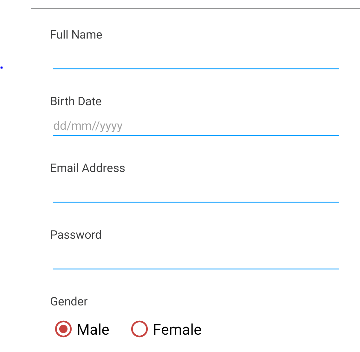
NetworkInfo info = conmag.getActiveNetworkInfo();  
if (info != null && info.isConnected())  
 return true;  
else  
 Toast.*makeText*(context.getApplicationContext(), "No Internet Connection", Toast.*LENGTH\_LONG*).show();



* Here as shown in Registration Screen user fill form and click on Signup button, then code check for internet connection, if connection available, user created else No Internet Connection Toast Shown.

**2. Using Typecasting**

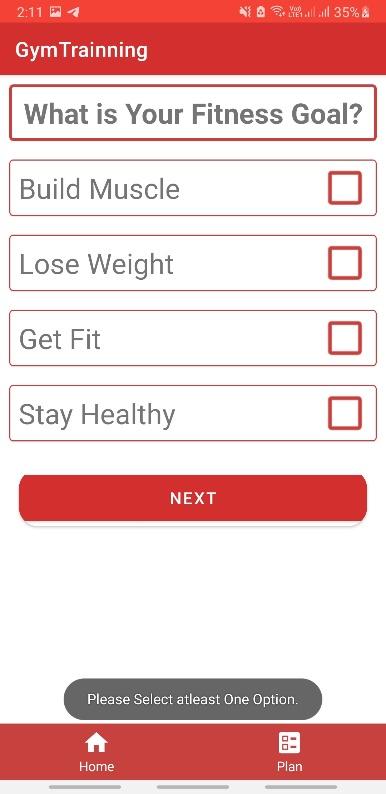
etname = findViewById(R.id.*etfullname*);  
etemail = findViewById(R.id.*etemail*);  
etpass = findViewById(R.id.*etpassword*);  
etbd = findViewById(R.id.*etbdate*);  
rdgendergroup = findViewById(R.id.*rggender*);  
  
utility = new Utility(Register.this);  
btnreg = findViewById(R.id.*btnregister*);



* Users need to cast UIView to Java Object to get text from Edit text and Radio Button.

**3. Multiple if and Logical or**

if (chkbm.isChecked() || chklw.isChecked() || chkgf.isChecked() || chksh.isChecked()) {  
  
 if (chkbm.isChecked()) {  
 plantext.append(tvbm.getText().toString());  
 }  
 if (chklw.isChecked()) {  
 plantext.append(tvlw.getText().toString());  
 }  
 if (chkgf.isChecked()) {  
 plantext.append(tvgf.getText().toString());  
 }  
 if (chksh.isChecked()) {  
 plantext.append(tvsh.getText().toString());  
 }  
  
  
  
 startActivity(new Intent(getApplicationContext(),Activness.class).putExtra("pt",plantext.toString()));  
  
} else {  
 Toast.*makeText*(FitnessGoal.this, "Please Select atleast One Option.", Toast.*LENGTH\_SHORT*).show();  
}



* Using multiple if and Logical or operator to validate fields if the user does not select at least one option. Without selecting any option, the user cannot go to the next step. Users get a Toast message.

**5. Use Built-in methods**

Matcher matcher = pattern1.matcher(editText.getText().toString());

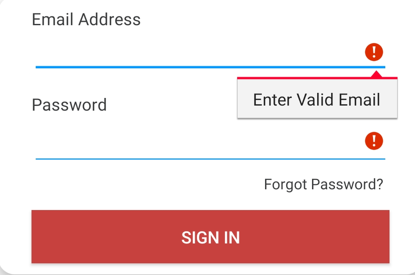
Dialog dialog = new Dialog(context);  
dialog.requestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
dialog.setContentView(resId);  
dialog.setCanceledOnTouchOutside(false);  
dialog.setCancelable(false);

* Here Using Matcher built-in Class and matcher () method matches the text from Edit text and returns an error Message.

**6. User Defined methods with if…else and return value**

static boolean nullValidator(EditText editText) {  
 if (editText.getText().toString().matches("")) {  
 return false;  
 } else {  
 return true;  
 }  
}

if(!Validation.*email\_Validation*(uname))  
{  
 uname.setError("Enter Valid Email");  
 b=false;  
}  
  
if(!Validation.*passwordValidator*(pw))  
{  
 pw.setError("Enter valid Password");  
 b=false;  
}



* Above Code illustrates the user defined method named nullValidator() takes an argument as text of edittext & returns an error message. Else return true.

**7. Using Parameterized Constructor**

public class Utility {  
 Context context;  
  
 Utility(Context context){  
 this.context=context;  
 }

**}**

public class BegExadapter extends RecyclerView.Adapter<BegExadapter.VH> {  
 ArrayList<BegEx> arrayList;  
 Context context;  
  
  
 public BegExadapter(ArrayList<BegEx> arrayList, Context context) {  
 this.arrayList = arrayList;  
 this.context = context;  
 }

**}**

* This code illustrates the parameterized constructor which takes an object of Activity to initiate a class object. Which is used to access properties of Utility class.

**8. Create Database Connection**

DatabaseReference databaseReference;

databaseReference = FirebaseDatabase.*getInstance*().getReference("Users");

* Using the getInstance() method creates a connection of Database and the getReference () method created here is a reference to the same table as the tables in the SQL database.

**9. Create Authentication**

FirebaseAuth firebaseAuth;

firebaseAuth = FirebaseAuth.*getInstance*();

* Using the getInstance() method creates a connection of Authentication. Using Firebase authentication users can login with different features like login with Gmail, Google and more.

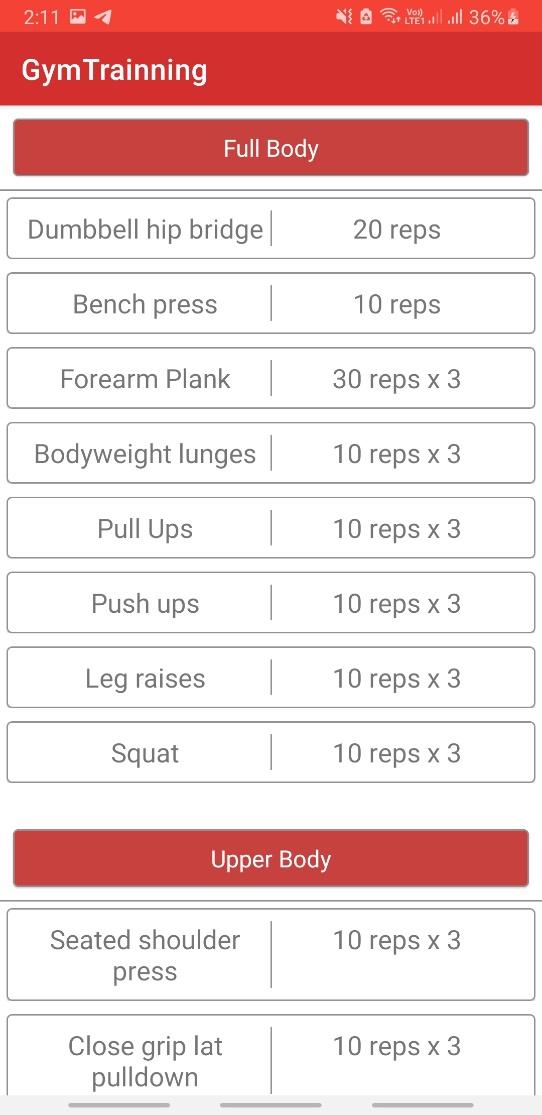
**10. Array List for Store Exercises list Object**

ArrayList<BegEx> alfb,alub,allb;

alfb = new ArrayList<>();  
alub = new ArrayList<>();  
allb = new ArrayList<>();

if (snapshot.exists()) {

for (DataSnapshot data : snapshot.getChildren()) {  
  
 BegEx bb = data.getValue(BegEx.class);  
 alfb.add(bb);  
 }  
 BegExadapter begExadapter = new BegExadapter(alfb,AutomatedPlan.this);  
 rvfb.setAdapter(begExadapter);  
  
  
}

****

* All data fetched from database stores into Arraylist and sent to Custom Adapter, Custom Adapter loads data into view dynamically.

**11. Using Enhanced for loop**

for (DataSnapshot data : snapshot.getChildren()) {  
  
 BegEx bb = data.getValue(BegEx.class);  
 alfb.add(bb);  
}

* Here I don’t use the traditional loop for adding data into arraylist because enhanced for loop is far better than traditional loop. Enhanced for loop works with objects too and objects are stored one by one into an array list automatically till the end of data.

**12. Encapsulation (Model Class)**

public class BegEx {  
  
 String name,reps;  
  
 public BegEx() {  
 }  
  
 public BegEx(String name, String reps) {  
 this.name = name;  
 this.reps = reps;  
 }  
  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getReps() {  
 return reps;  
 }  
  
 public void setReps(String reps) {  
 this.reps = reps;  
 }  
}

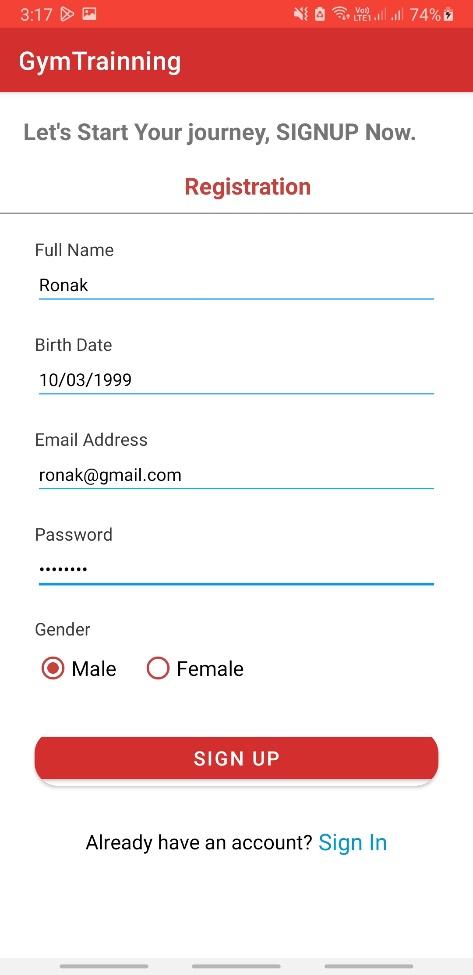
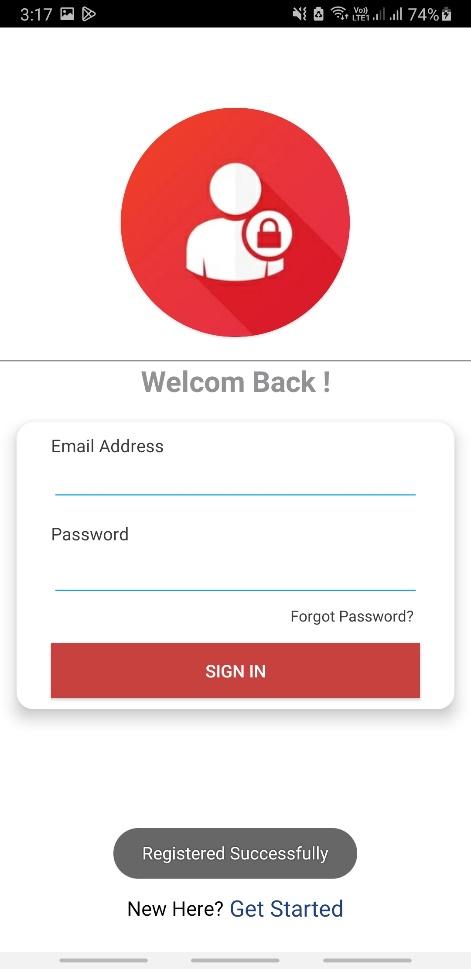
* Encapsulation contains their own methods (such as getParent(), addParent(), loadFromDatabase(), etc.) in order to facilitate code reusability and modularity through encapsulation . This, along with documentation comments, led to a much easier maintenance time when an error was found.

**13. Insert data into database**

DatabaseReference databaseReference;

databaseReference = FirebaseDatabase.*getInstance*().getReference("Users");

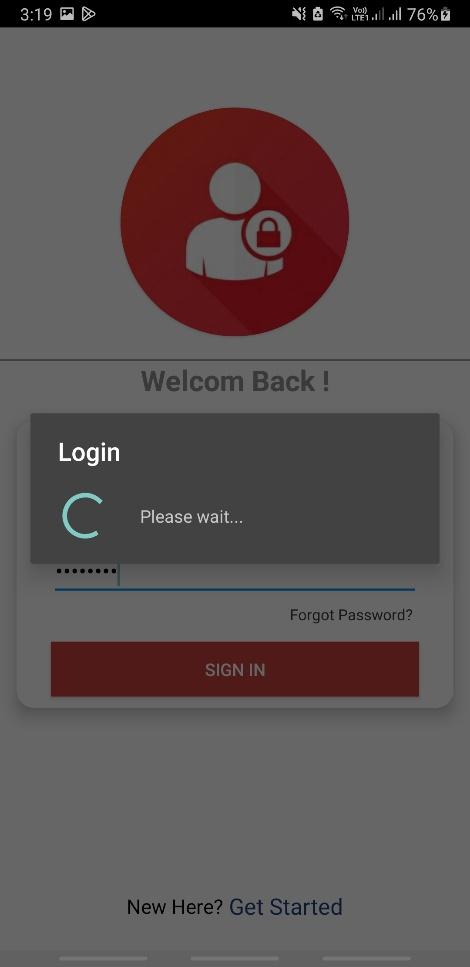
String id = firebaseAuth.getCurrentUser().getUid();  
User users = new  
 User(id,etname.getText().toString().trim(),etbd.getText().toString().trim(),  
 etemail.getText().toString().trim(),etpass.getText().toString().trim(),rdgen.getText().toString());  
databaseReference.child(id).setValue(users);  
utility.toast("Registered Successfully");

**14. Custom Dialog Box**

final ProgressDialog pg = new ProgressDialog(Login.this);  
  
pg.setTitle("Login");  
pg.setMessage("Please wait...");  
pg.setCancelable(false);

pg.dismiss();



* Here the process Dialog box is used to show the background process. Here I used the process dialog box to wait & check user information for validate and redirect to the home page.

**15. Inheritance & Polymorphism**

public class BegExadapter extends RecyclerView.Adapter<BegExadapter.VH> {  
 ArrayList<BegEx> arrayList;  
 Context context;  
  
  
 public BegExadapter(ArrayList<BegEx> arrayList, Context context) {  
 this.arrayList = arrayList;  
 this.context = context;  
 }  
  
 @NonNull  
 @Override  
 public BegExadapter.VH onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.*from*(context).inflate(R.layout.*cardbegex*, parent, false);  
 return new VH(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull BegExadapter.VH holder, int position) {  
  
 BegEx b = arrayList.get(position);  
 holder.tvreps.setText(b.getReps());  
 holder.tvname.setText(b.getName());  
  
 }  
  
 @Override  
 public int getItemCount() {  
 return arrayList.size();  
 }  
  
 public class VH extends RecyclerView.ViewHolder {  
  
 TextView tvname, tvreps;  
  
 public VH(@NonNull View itemView) {  
 super(itemView);  
  
 tvname = itemView.findViewById(R.id.*tvname*);  
 tvreps = itemView.findViewById(R.id.*tvreps*);  
  
  
 }  
 }  
}

* To use overridden methods of Recyclerview and its functionality, it requires inheritance. Then, polymorphism is used to get super class methods and functionalities that I had programmed in for the Custom Adapter to execute. Here, it inherits from RecyclerView.Adapter(which all activities must inherit from in order to inherit the onCreateViewHolder, onBindViewHolder).