MOBILE APP DEVELOPMENT
DA-1

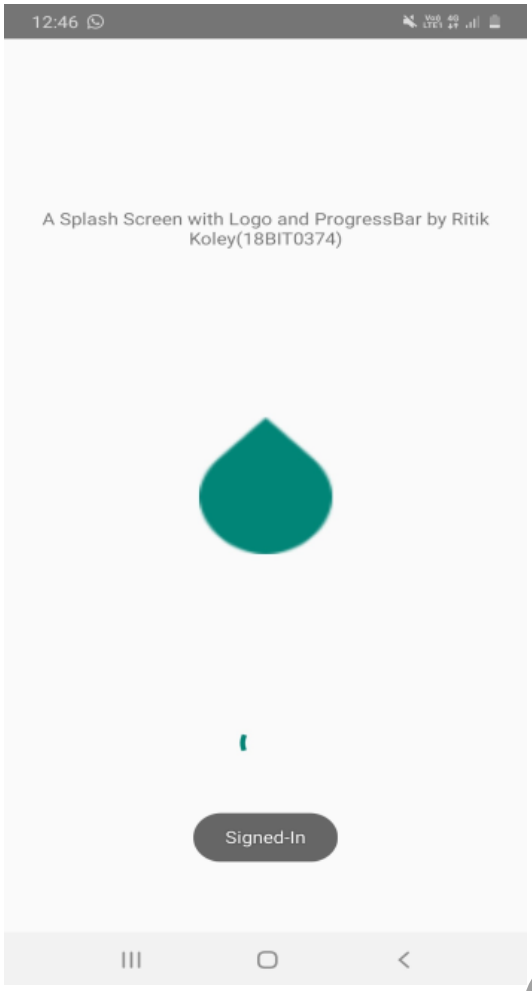
SPLASHCREEN INTERFACE:

```
package com.ritikkoley.myapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.widget.Toast;

import com.firebase.ui.auth.AuthUI;
import com.firebase.ui.auth.IdpResponse;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.Arrays;
import java.util.List;

public class SplashScreenActivity extends AppCompatActivity {

    private static final int RC_SIGN_IN = 23;

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

```
        requestSingIn();
    }

    //Method to start MainActivity
    private void openMainActivity(){
        Intent intent = new Intent(SplashScreenActivity.this, MainActivity.class);
        startActivity(intent);
    }

    public void requestSingIn(){
        // Choose authentication providers
        List<AuthUI.IdpConfig> providers = Arrays.asList(
                new AuthUI.IdpConfig.EmailBuilder().build(),
                new AuthUI.IdpConfig.GoogleBuilder().build());

// Create and launch sign-in intent
        startActivityForResult(
                AuthUI.getInstance()
                        .createSignInIntentBuilder()
                        .setAvailableProviders(providers)
                        .build(),
                RC_SIGN_IN);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == RC_SIGN_IN) {
            IdpResponse response = IdpResponse.fromResultIntent(data);

            if (resultCode == RESULT_OK) {
                // Successfully signed in
                FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                Toast.makeText(this,"Signed-In", Toast.LENGTH_LONG).show();
                //Start Main Activity after % seconds Delay
                final Handler handler = new Handler();
                handler.postDelayed(new Runnable() {
                    @Override
                    public void run() {
```

```
                openMainActivity();
                finish();
            }
        }, 2000);
        // ...
    } else {
        // Sign in failed. If response is null the user canceled the
        // sign-in flow using the back button. Otherwise check
        // response.getError().getErrorCode() and handle the error.
        // ...
        Toast.makeText(this,"Sign-In Failed", Toast.LENGTH_LONG).show();
    }
  }
}
```

Screenshot:



ARCHIVE FUNCTIONALITY:

```
package com.ritikkoley.myapp.callbacks;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffXfermode;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
```

```
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.RecyclerView;

import com.ritikkoley.myapp.R;

public class SwipeToArchiveCallback extends ItemTouchHelper.Callback {

    Context mContext;
    private Paint mClearPaint;
    private ColorDrawable mBackground;
    private int backgroundColor;
    private Drawable deleteDrawable;
    private int intrinsicWidth;
    private int intrinsicHeight;

    public SwipeToArchiveCallback(Context context) {
        mContext = context;
        mBackground = new ColorDrawable();
        backgroundColor = Color.parseColor("#32CD32");
        mClearPaint = new Paint();
        mClearPaint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.CLEAR));
        deleteDrawable = ContextCompat.getDrawable(mContext, R.drawable.ic_archive);
        intrinsicWidth = deleteDrawable.getIntrinsicWidth();
        intrinsicHeight = deleteDrawable.getIntrinsicHeight();

    }

    @Override
    public int getMovementFlags(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder
viewHolder) {
        return makeMovementFlags(0, ItemTouchHelper.RIGHT);
    }

    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder
viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
```

```
    public void onChildDrawOver(@NonNull Canvas c, @NonNull RecyclerView recyclerView,
RecyclerView.ViewHolder viewHolder, float dX, float dY, int actionState, boolean isCurrentlyActive) {
        super.onChildDraw(c, recyclerView, viewHolder, dX, dY, actionState, isCurrentlyActive);
        View itemView = viewHolder.itemView;
        int itemHeight = itemView.getHeight();

        boolean isCancelled = dX == 0 && !isCurrentlyActive;

        if (isCancelled) {
            clearCanvas(c, itemView.getLeft() + dX, (float) itemView.getTop(), (float) itemView.getLeft(),
                    (float) itemView.getBottom());
            super.onChildDraw(c, recyclerView, viewHolder, dX, dY, actionState, isCurrentlyActive); return;
        }

        mBackground.setColor(backgroundColor);
        mBackground.setBounds(itemView.getLeft() + (int) dX, itemView.getTop(), itemView.getLeft(),
itemView.getBottom());
        mBackground.draw(c);

        int deleteIconTop = itemView.getTop() + (itemHeight - intrinsicHeight) / 2; int deleteIconMargin = (itemHeight
- intrinsicHeight) / 2;
        int deleteIconLeft = itemView.getLeft() - deleteIconMargin ;
        int deleteIconBottom = deleteIconTop + intrinsicHeight;

        deleteDrawable.setBounds(deleteIconLeft, deleteIconTop, deleteIconLeft, deleteIconBottom);
        deleteDrawable.draw(c);
        super.onChildDraw(c, recyclerView, viewHolder, dX, dY, actionState, isCurrentlyActive);

    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {

    }

    @Override
    public float getSwipeThreshold(@NonNull RecyclerView.ViewHolder viewHolder) {
        return 0.7f;
    }
```

```
    private void clearCanvas(Canvas c, Float left, Float top, Float right, Float bottom) {
        c.drawRect(left, top, right, bottom, mClearPaint);
    }
}
```

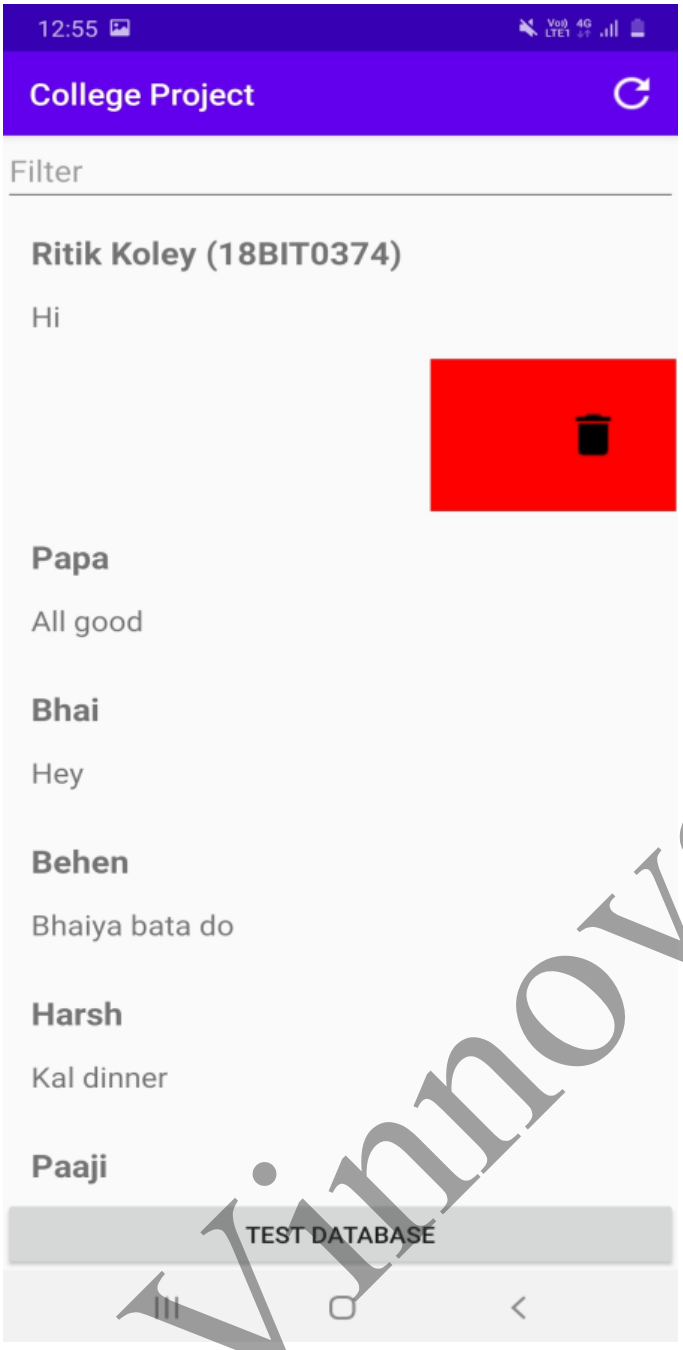
DELETE FUNCTIONALITY:

```
package com.ritikkoley.myapp.callbacks;

import android.content.Context;
```

```
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffXfermode;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.RecyclerView;

import com.ritikkoley.myapp.R;

public class SwipeToDeleteCallback extends ItemTouchHelper.Callback {

    Context mContext;
    private Paint mClearPaint;
    private ColorDrawable mBackground;
    private int backgroundColor;
    private Drawable deleteDrawable;
    private int intrinsicWidth;
    private int intrinsicHeight;

    public SwipeToDeleteCallback(Context context) {
        mContext = context;
        mBackground = new ColorDrawable();
        backgroundColor = Color.parseColor("#FF0000");
        mClearPaint = new Paint();
        mClearPaint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.CLEAR));
        deleteDrawable = ContextCompat.getDrawable(mContext, R.drawable.ic_delete); intrinsicWidth =
deleteDrawable.getIntrinsicWidth();
        intrinsicHeight = deleteDrawable.getIntrinsicHeight();
    }

    @Override
    public int getMovementFlags(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder
viewHolder) {
```

```
        return makeMovementFlags(0, ItemTouchHelper.LEFT);
    }

    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder
viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onChildDraw(@NonNull Canvas c, @NonNull RecyclerView recyclerView, @NonNull
RecyclerView.ViewHolder viewHolder, float dX, float dY, int actionState, boolean isCurrentlyActive) {
        View itemView = viewHolder.itemView;
        int itemHeight = itemView.getHeight();

        boolean isCancelled = dX == 0 && !isCurrentlyActive;

        if (isCancelled) {
            clearCanvas(c, itemView.getRight() + dX, (float) itemView.getTop(), (float)
                    itemView.getRight(), (float) itemView.getBottom());
            super.onChildDraw(c, recyclerView, viewHolder, dX, dY, actionState, isCurrentlyActive); return;
        }

        mBackground.setColor(backgroundColor);
        mBackground.setBounds(itemView.getRight() + (int) dX, itemView.getTop(), itemView.getRight(),
itemView.getBottom());
        mBackground.draw(c);

        int deleteIconTop = itemView.getTop() + (itemHeight - intrinsicHeight) / 2;
        int deleteIconMargin = (itemHeight - intrinsicHeight) / 2;
        int deleteIconLeft = itemView.getRight() - deleteIconMargin - intrinsicWidth;
        int deleteIconRight = itemView.getRight() - deleteIconMargin;
        int deleteIconBottom = deleteIconTop + intrinsicHeight;

        deleteDrawable.setBounds(deleteIconLeft, deleteIconTop, deleteIconRight, deleteIconBottom);
        deleteDrawable.draw(c);

        super.onChildDraw(c, recyclerView, viewHolder, dX, dY, actionState, isCurrentlyActive);
    }
```

```
    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {

    }

    @Override
    public float getSwipeThreshold(@NonNull RecyclerView.ViewHolder viewHolder) {
        return 0.7f;
    }

    private void clearCanvas(Canvas c, Float left, Float top, Float right, Float bottom) {
        c.drawRect(left, top, right, bottom, mClearPaint);
    }
}
```



RESTORE/RECYLER VIEW SCREEN:

```
ackage com.ritikkoley.myapp.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Filter;
import android.widget.TextView;
```

```java
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.ritikkoley.myapp.R;
import com.ritikkoley.myapp.dataclasses.ItemData;

import java.util.ArrayList;
import java.util.List;

public class RecyclerViewAdapter extends RecyclerView.Adapter<RecyclerViewAdapter.MyViewHolder> {

    Context mContext;
    List<ItemData> mData;
    List<ItemData> contactListFiltered;
    ArrayList<ItemData> nameArrayList = new ArrayList<>();

    public RecyclerViewAdapter(Context context, List<ItemData> itemData) {
        this.mContext = context;
        this.mData = itemData;
    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View v;
        v = LayoutInflater.from(mContext).inflate(R.layout.item_layout, parent, false);
        return new MyViewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        holder.title.setText(mData.get(position).getItemTitle());
        holder.description.setText(mData.get(position).getItemDescription());
    }

    @Override
    public int getItemCount() {
        return mData.size();
    }
```

```
public Filter getFilter() {
    return new Filter() {
        @Override
        protected FilterResults performFiltering(CharSequence charSequence) {
            String charString = charSequence.toString();
            if (charString.isEmpty()) {
                contactListFiltered = mData;
            } else {
                List<ItemData> filteredList = new ArrayList<>();
                for (ItemData row : mData) {
                    if (row.getItemTitle().toLowerCase().contains(charString.toLowerCase())) {
                        filteredList.add(row);
                    }
                }
                contactListFiltered = filteredList;
            }
            FilterResults filterResults = new FilterResults();
            filterResults.values = contactListFiltered;
            return filterResults;
        }

        @Override
        protected void publishResults(CharSequence charSequence, FilterResults filterResults) {
            contactListFiltered = (ArrayList<ItemData>) filterResults.values;
            notifyDataSetChanged();
        }
    };
}

public void filterList(ArrayList<ItemData> filteredList){
    mData = filteredList;
    notifyDataSetChanged();
}

public void removeItem(int position) {
    mData.remove(position); notifyItemRemoved(position);
}

public void restoreItem(ItemData item, int position) {
    mData.add(position, item); notifyItemInserted(position);
```

```
}

public List<ItemData> getData() {
    return mData;
}

public static class MyViewHolder extends RecyclerView.ViewHolder {

    private TextView title;
    private TextView description;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        title = itemView.findViewById(R.id.item_title);
        description = itemView.findViewById(R.id.item_description);
    }
}
```

SCREENSHOT:

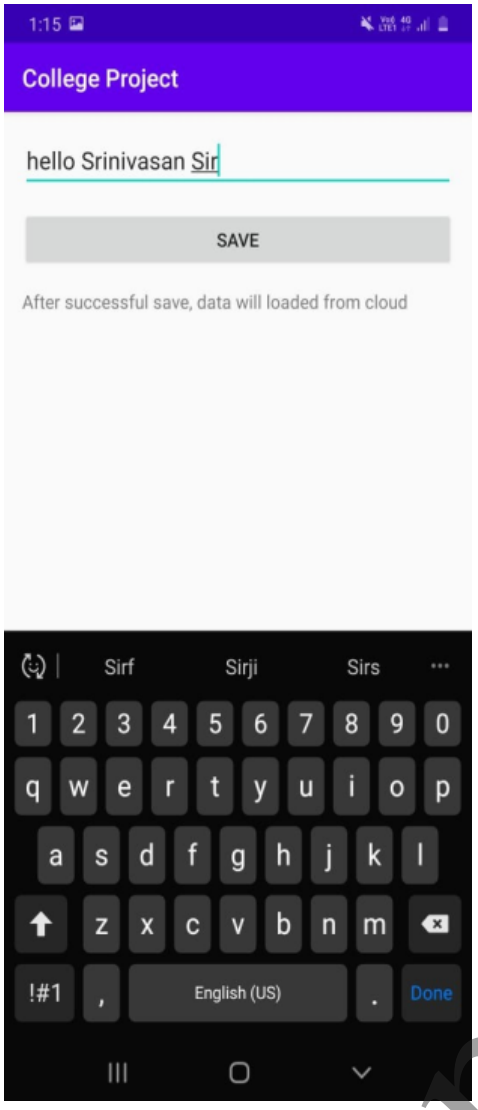
ASSIGNMENT 2:

LINK WITH FIREBASE DATABASE:

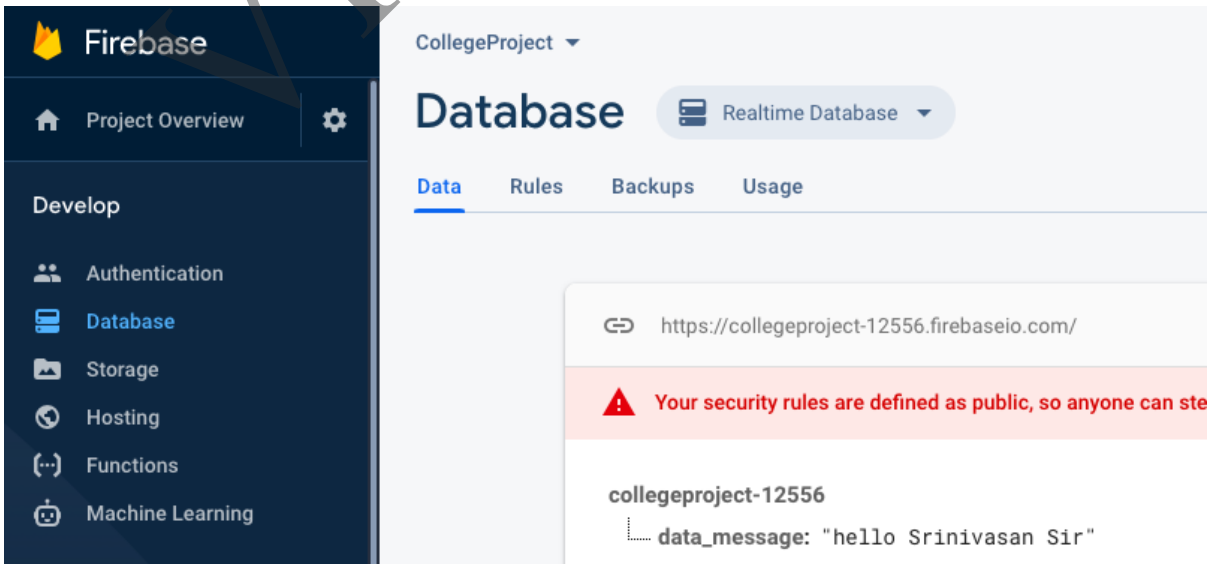
```
package com.ritikkoley.myapp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

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.ValueEventListener;

public class DatabaseActivity extends AppCompatActivity {

    EditText editTextMessage;
    Button buttonSaveData;
    TextView textViewData;

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

        editTextMessage = findViewById(R.id.edit_text_message);
        buttonSaveData = findViewById(R.id.button_save_data);
        textViewData = findViewById(R.id.text_view_retreived);

        buttonSaveData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                writeToFirebaseDatabse(editTextMessage.getText().toString());
            }
```

```
        });
    }

    public void writeToFirebaseDatabse(String message){
        // Write a message to the database
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("data_message");
        myRef.setValue(message);

        myRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                String value = dataSnapshot.getValue().toString();
                Toast.makeText(DatabaseActivity.this,"Loaded from DB: \n" + value,Toast.LENGTH_LONG).show();
                textViewData.setText(value);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Toast.makeText(DatabaseActivity.this,"Failed to read Value",Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

Screenshot of sending a message to firebase:



FIREBASE DATABASE SCREENSHOT.



PHONE GAP SIMPLE APP

Index.html:-

Recycler Adapter:-

```
<!DOCTYPE html>
<html>
<head>
<!--Stylesheet Files : jQuery Mobile CSS File, Customized CSS File and Font Awesome for
icons -->
<link rel="stylesheet" href="css/jquery.mobile-1.4.5.css">
<link rel="stylesheet" href="css/my.css">
<link rel="stylesheet" href="font-awesome/css/font-awesome.min.css">
</head>
<!--Beginning of the Body-->
<body>
<div data-role="page">
<!--Header Bar-->
<div data-role="header" data-position="fixed" class="ui-header ui-bar-a ui-header-fixed
slidedown" role="banner">
<h1>Store Me</h1>
</div>
<!--Beginning of the Name and Email Field Div-->
<div data-role="main" class="ui-content" id="main">
<center><p id="heading">CRUD Operations Using SQLite Plugin</p></center>
<center><img src="img/phonegap-sqlite-plugin-img.png"/></center>
<div class="ui-grid-a">
<div class="ui-block-a"><div class="ui-bar ui-bar-a"
<center><label for="text-basic">Name</label></center><input type="text" name="text-basic"
id="name" placeholder="Name">
</div></div>
<div class="ui-block-b"><div class="ui-bar ui-bar-a">
<center><label for="text-basic">Email</label></center>
<input type="text" name="text-basic" id="email" placeholder="Email">
</div></div> </div>
<!--End of the Name and Email Field Div-->
<!--Beginning of the Create and Clear All Div-->
<fieldset class="ui-grid-a" id="btndiv">
<div class="ui-block-a"><div class="ui-bar ui-bar-a">
<a href="#" id ="creat" onclick="create()">
<center><i class="fa fa-pencil"> Create</i></center>
</a></div>
</div>
<!--End of the Create and Clear All Button Div-->
<!--Dialog Box when clicked on Clear All button-->
<div class="ui-block-b"><div class="ui-bar ui-bar-a">
<a id ="bt" href="#popupDialog" data-rel="popup" data-position-to="window" data-
transition="pop" class="ui-btn ui-corner-all ui-shadow " >
<center><i class="fa fa-scissors"> Clear All</i></center>
```

```
</a></div>
</div>
</fieldset>
<div data-role="popup" id="popupDialog" data-dismissible="false" style="max-width:400px;"> <div data-
role="header" data-theme="a">
<h1>Clear All?</h1>
</div>
<div role="main" class="ui-content">
<h3 class="ui-title">Are you sure you want to clear local storage?</h3>
<center><b><p>This action cannot be undone.</p></b></center>
<center> <a href="#" id ="clearall" class="ui-btn ui-corner-all ui-shadow ui-btn-inline ui-
btn-b" data-rel="back" data-transition="flow">Yes! I'm Sure</a><a href="#" class="ui-btn
ui-corner-all ui-shadow ui-btn-inline ui-btn-b" data-rel="back">Cancel</a></center>
</div>
</div>
<!-- End of the Clear All Dialog Box-->
<!--Table Displaying Database Rows-->
<table data-role="table" data-mode="reflow" class="ui-responsive ui-shadow" id="myTable">
</table>
<!--Dialog Box when clicked on update button-->
<div data-role="main" class="ui-content">
<div data-role="popup" id="myPopupDialog">
<div data-role="header">
<h1>Update</h1>
</div>
<div data-role="main" class="ui-content">
<form method="post">
<div class="ui-field-contain">
<table id="utable">
<tr>
<td><label for="id">Id:</label></td>

<td><input disabled="disabled" type="text" name="id" id="id" placeholder="Id"

value=""></td>

</tr>

<tr>
<td><label for="name">Name:</label></td>
<td><input type="text" name="name" id="uname" placeholder="Name" value=""></td>
</tr>
<tr>
<td><label for="email">Email:</label></td>
<td><input type="email" name="email" id="uemail" placeholder="Enter Your New Email"></td>
</tr>
</table>
</div>
<center><a href="#" class="ui-btn ui-corner-all ui-shadow ui-btn-inline ui-btn-b" data-
rel="back" id="upd" data-transition="flow">Update</a> <a href="#" class="ui-btn ui-corner-
all ui-shadow ui-btn-inline ui-btn-b" data-rel="back">Cancel</a>
</center>
</form>
</div>
```

```
</div>
</div> </div>
<!--End of the Update Dialog Box -->
<!--jQuery File : Library, Mobile Library, Cordova JS, SQLite Plugin JS and Customized JS
File -->
<script type="text/javascript" charset="utf-8" src="js/jquery-1.11.3.min.js"></script>
<script type="text/javascript" charset="utf-8" src="js/jquery.mobile-1.4.5.js"></script>
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" charset="utf-8" src="SQLitePlugin.js"></script>
<script type="text/javascript" src="js/my.js"></script>
</body>
<!--End of the Body-->
</html> My.js:-
<!--Calling onDeviceReady method-->
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
<!--window.sqlitePlugin.openDatabase creates/open a non existing/existing database-->
var db = window.sqlitePlugin.openDatabase({name: "my.db"});
show();
db.transaction(function(tx) {
tx.executeSql('CREATE TABLE IF NOT EXISTS mydata (id integer primary key, name text, email
text)');
});
<!--Method to insert new row in the database-->
$(document).on('click', '#creat', function(){
var name = $("#name").val();
var email = $("#email").val();
db.transaction(function(transaction) {
var executeQuery = "INSERT INTO mydata (name, email) VALUES (?,?)";
transaction.executeSql(executeQuery, [name,email]
, function(tx, result) {
show();
},
function(error){
//filter(function(aSome) {alert('Error occurred');
});

});

});

<!--Display all rows stored in the database-->
function show(){
db.transaction(function(transaction) {
transaction.executeSql('SELECT * FROM mydata', [], function (tx, results) {
var key = "";
<!--Display the table head-->
var pair="<tr><th data-priority=\"1\"><center>Id</center></th><th data-
priority=\"1\"><center>Name</center></th><th data-
priority=\"2\"><center>Email</center></th><th><center>Update</center></th><th><center>Delet
e</center></th></tr>";
var i=0;
<!--results.rows.length to get the total number of rows stored in the database--> var len =
results.rows.length, i;
```

```
for (i=0; i<=len-1; i++) {
<!--Fetching the 'name' from the database-->
key = results.rows.item(i).name;
<!--Fetching the 'id' from the database-->
id = results.rows.item(i).id;
<!--Displaying all rows of the database in the table-->
pair +=
"<tr><td><center>"+id+"</center></td><td><center>"+key+"</center></td><td><center>"+results
.rows.item(i).email+"</center></td><td><a class=\"update\" href=\"#myPopupDialog\" data-
custom="+"'"+ id+ "'" +"data-rel=\"popup\" data-position-to=\"window\" data-
transition=\"pop\"><center><i class='fa fa-pencil-square-o'></i></center></a></td><td><a
id=\"delete\" data=\""+id+"\"><center><i class='fa fa-trash'></i></center></a></td></tr>";
}
if (pair == "<tr><th>Name</th><th>Email</th></tr>") {
pair += "<tr><td><i>empty</i></td><td><i>empty</i></td></tr>";
} $("#myTable").html(pair);
}, null);
});
}
<!--Method to delete any row from the database-->
$(document).on('click', '#delete', function(){
var id = $(this).attr("data");
db.transaction(function(transaction) {
transaction.executeSql("DELETE FROM mydata where id=?", [id],
function(tx, result) {
show();
},
function(error){
// alert('Something went Wrong');
});
});
});
<!--Method to update the values of any row in the database-->
$(document).on('click', '#upd', function(){
var id = $("#id").val();
var name = $("#uname").val();
var email = $("#uemail").val();
db.transaction(function(transaction) {
var executeQuery = "";
transaction.executeSql("UPDATE mydata SET name=?, email=? WHERE id=?", [name,email,id],

function(tx, result) {alert('Updated successfully');

show();
},
function(error){alert('Something went Wrong');});
});
});
$(document).on('click', '.update', function(){
var id = $(this).attr('data-custom');
$("#id").val(id);
db.transaction(function(transaction) {
transaction.executeSql('SELECT name,email FROM mydata where id=?', [id], function (tx,
results) {
```

```
var name = results.rows.item(0).name;
var email = results.rows.item(0).email;
$("#uname").val(name);
$("#uemail").val(email);
},
function(error){
alert('Something went Wrong');
});
});
});
<!--Method to clear all rows from the database-->
$(document).on('click', '#clearall', function(){
db.transaction(function(transaction) {
transaction.executeSql("DELETE FROM mydata", [],
function(tx, result) {alert('Delete successfully');
show();
},
function(error){alert('Something went Wrong');});
});
});
}
```

My.css

```
.ui-bar-a, .ui-page-theme-a .ui-bar-inherit, html .ui-bar-a .ui-bar-inherit, html .ui-body-
a .ui-bar-inherit, html body .ui-group-theme-a .ui-bar-inherit {
border: 1px solid #005994 !important;
background: #0093EA !important;
color: #fff !important;
font-weight: bold !important;
text-shadow: 0 0 #eee !important;
background-image: -webkit-gradient(linear, left top, left bottom, from( #0093EA), to(
#007dcd ));
background-image: -webkit-linear-gradient( #0093EA , #007dcd );
background-image: -moz-linear-gradient( #0093EA, #007dcd );
background-image: -ms-linear-gradient( #0093EA , #007dcd );
background-image: -o-linear-gradient( #0093EA , #007dcd );
background-image: linear-gradient( #0093EA , #007dcd );
}
.ui-page-theme-a .ui-btn:hover, html .ui-bar-a .ui-btn:hover, html .ui-body-a .ui-
btn:hover, html body .ui-group-theme-a .ui-btn:hover, html head + body .ui-btn.ui-btn-
a:hover{
border: 1px solid #007dcd;
background: #333 ;

font-weight: bold;
text-shadow: 0 0 #eee !important;
color: #fff !important;
background-image: -webkit-gradient(linear, left top, left bottom, from( #0093EA ), to(
#0093EA ));
background-image: -webkit-linear-gradient( #0093EA , #0093EA );
background-image: -moz-linear-gradient( #0093EA , #0093EA );
background-image: -ms-linear-gradient( #0093EA , #0093EA );
background-image: -o-linear-gradient( #0093EA , #0093EA );
background-image: linear-gradient( #0093EA , #0093EA );
}
```

```
.ui-page-theme-a .ui-btn.ui-btn-active, html .ui-bar-a .ui-btn.ui-btn-active, html .ui-
body-a .ui-btn.ui-btn-active, html body .ui-group-theme-a .ui-btn.ui-btn-active, html head
+ body .ui-btn.ui-btn-a.ui-btn-active, .ui-page-theme-a .ui-checkbox-on:after, html .ui-
bar-a .ui-checkbox-on:after, html .ui-body-a .ui-checkbox-on:after, html body .ui-group-
theme-a .ui-checkbox-on:after, .ui-btn.ui-checkbox-on.ui-btn-a:after, .ui-page-theme-a .ui-
flipswitch-active, html .ui-bar-a .ui-flipswitch-active, html .ui-body-a .ui-flipswitch-
active, html body .ui-group-theme-a .ui-flipswitch-active, html body .ui-flipswitch.ui-bar-
a.ui-flipswitch-active, .ui-page-theme-a .ui-slider-track .ui-btn-active, html .ui-bar-a
.ui-slider-track .ui-btn-active, html .ui-body-a .ui-slider-track .ui-btn-active, html body
.ui-group-theme-a .ui-slider-track .ui-btn-active, html body div.ui-slider-track.ui-body-a
.ui-btn-active { background-color: #0093EA !important ;
border-color:#0093EA !important; color: #fff ;
text-shadow: 0 1px 0 #005599 ;
}
img{

padding: 25px;

}

button.ui-btn, .ui-controlgroup-controls button.ui-btn-icon-notext {

border-radius: 5px !important;

}

#searchbutton{

margin-bottom: 25px;

}

#main{

margin-top: 12% !important ;

}

.ui-collapsible-inset.ui-collapsible-themed-content .ui-collapsible-content

{ background-color: #ddd;

color: #111;

}
.ui-collapsible-content {
-webkit-transition: all 0.5s;
-moz-transition: all 0.5s;
-ms-transition: all 0.5s;
-o-transition: all 0.5s;
transition: all .5s;
//height: 2em;
overflow: hidden;
}.ui-collapsible-content-collapsed {
display: block;
height: 0;
padding: 0 16px;
```

```
}
#bt i{
font-weight: bold;
}
th {
border-bottom: 1px solid #d6d6d6 !important;
}
tr:nth-child(even) {
background: #e9e9e9 !important;
}
.ui-table {
margin-top: 5% !important;
border: 1px solid grey !important;
border-radius: 5px !important;
border-collapse: initial !important;
}
label{
font-weight: bold !important;
}
#label{
border: 1px solid #0093EA !important;
background: #fff !important;
color: #005994 !important;
font-weight: bold !important;
text-shadow: 0 0 #eee !important;
background-image: -webkit-gradient(linear, left top, left bottom, from( #0093EA), to(
#007dcd ));
background-image: -webkit-linear-gradient( #0093EA , #007dcd );
background-image: -moz-linear-gradient( #0093EA, #007dcd );
background-image: -ms-linear-gradient( #0093EA , #007dcd );

background-image: -o-linear-gradient( #0093EA , #007dcd );

background-image: linear-gradient( #0093EA , #007dcd );
}
#utable tr:nth-child(even){ background: inherit !important;
}
#heading{
font-weight: bold;
font-size: 40px;
}
#btndiv{
margin-top: 3%;

}

@media ( max-width: 35em ) {

.ui-table-reflow.ui-responsive td, .ui-table-reflow.ui-responsive th {

width: auto;

float: none;

clear: none; display: table-cell;

margin: 0;
```

```
padding:0;

}

}

#btndiv .ui-bar-a{

width: 50% !important;

margin: auto !important;
```

SCREENSHOT