Gym Management System - Full Stack Application (Kotlin, SQLite)

Project Report

Aranya Singh Chauhan (CSE, SRM IST)
Sahil (CSE, NIIT UNIVERSITY)

Problem Statement:

The project aims to build a mobile application for gyms, which in turn would help the managerial staff in automating the entire management process.

• The current system in place makes it difficult for the gym staff/trainers to efficiently manage member related activities in the gym.

• The proposed system would not only make it easier for the staff to manage gym related activities, but also make it simpler for them to store and manage details about members/other trainers, pending fees, etc.

Project Description:

This Gym Management System tool is developed to aid the managerial staff to be able to to add members to the gym. The staff shall be able to add the name, date of birth and contact address of the member. The system shall also records additional data such as phone numbers, height and weight data of the member such that they are easily accessible if and when required by the staff. The proposed system shall also have admission dates as well as an option to check whether the member being added is new or an existing one. The system would also store photos of the member.

For members of the gym, the proposed system would be able to remind them when they are due for a payment. It would also be able to let them know about the kind of plans the gym has to offer. Be it monthly, quarterly or annually depending on the choice of the member. The system even holds the receipt number and the amount of fees that has been paid till date by the member. It

shall also be able to generate reports based on the payment of fees. The managerial staff can also be provided with roles on the system that can have access to member records, which if their role permits, can be edited.

System Design:

Activity Diagram:

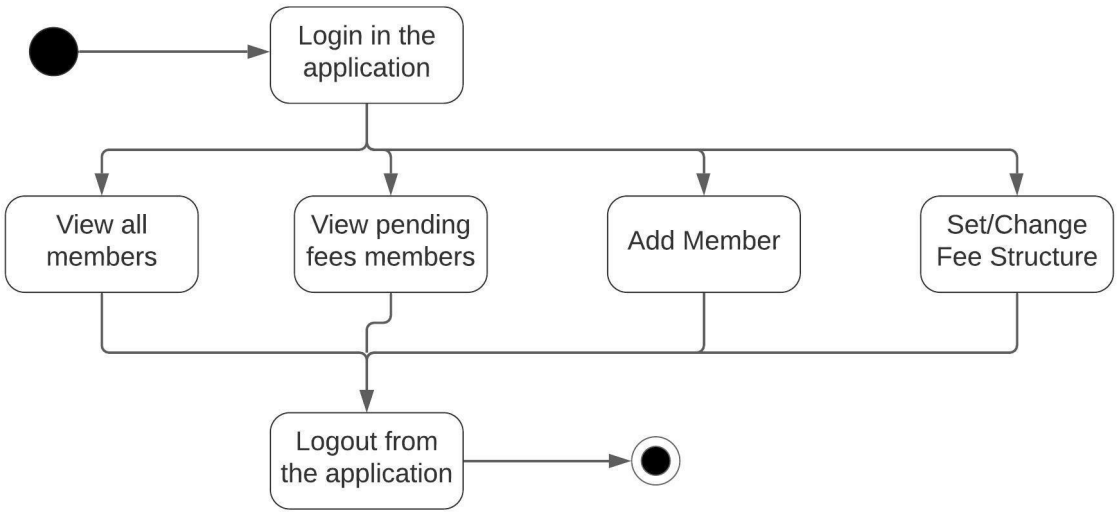
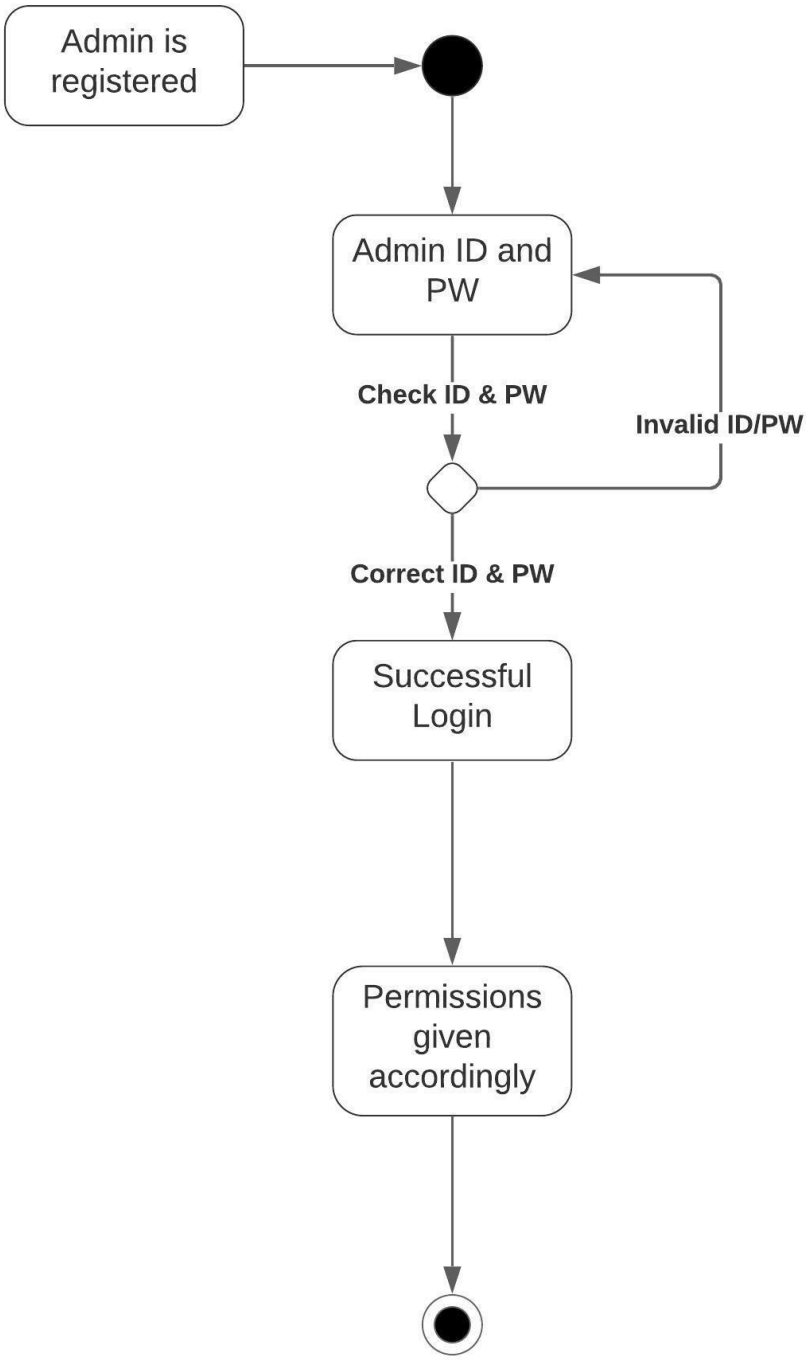
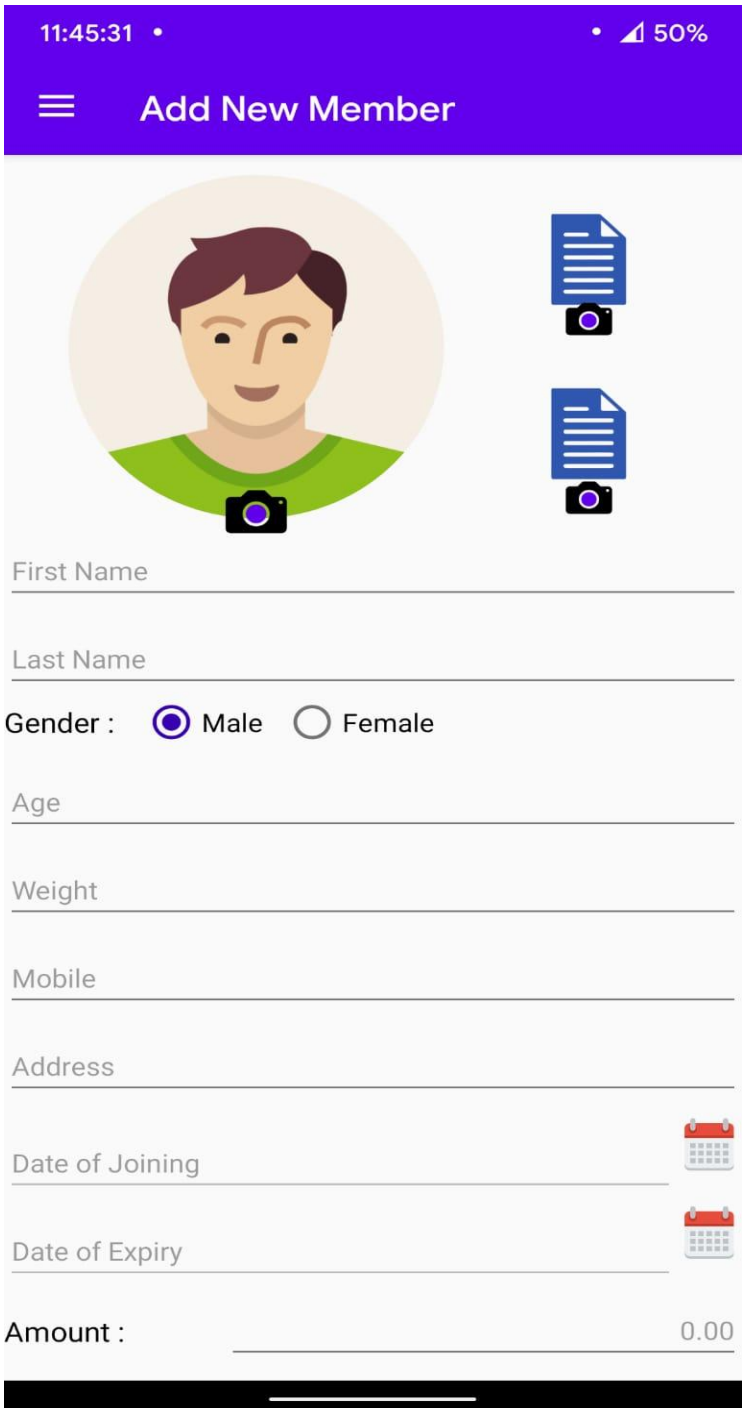


Diagram for Login Activity:

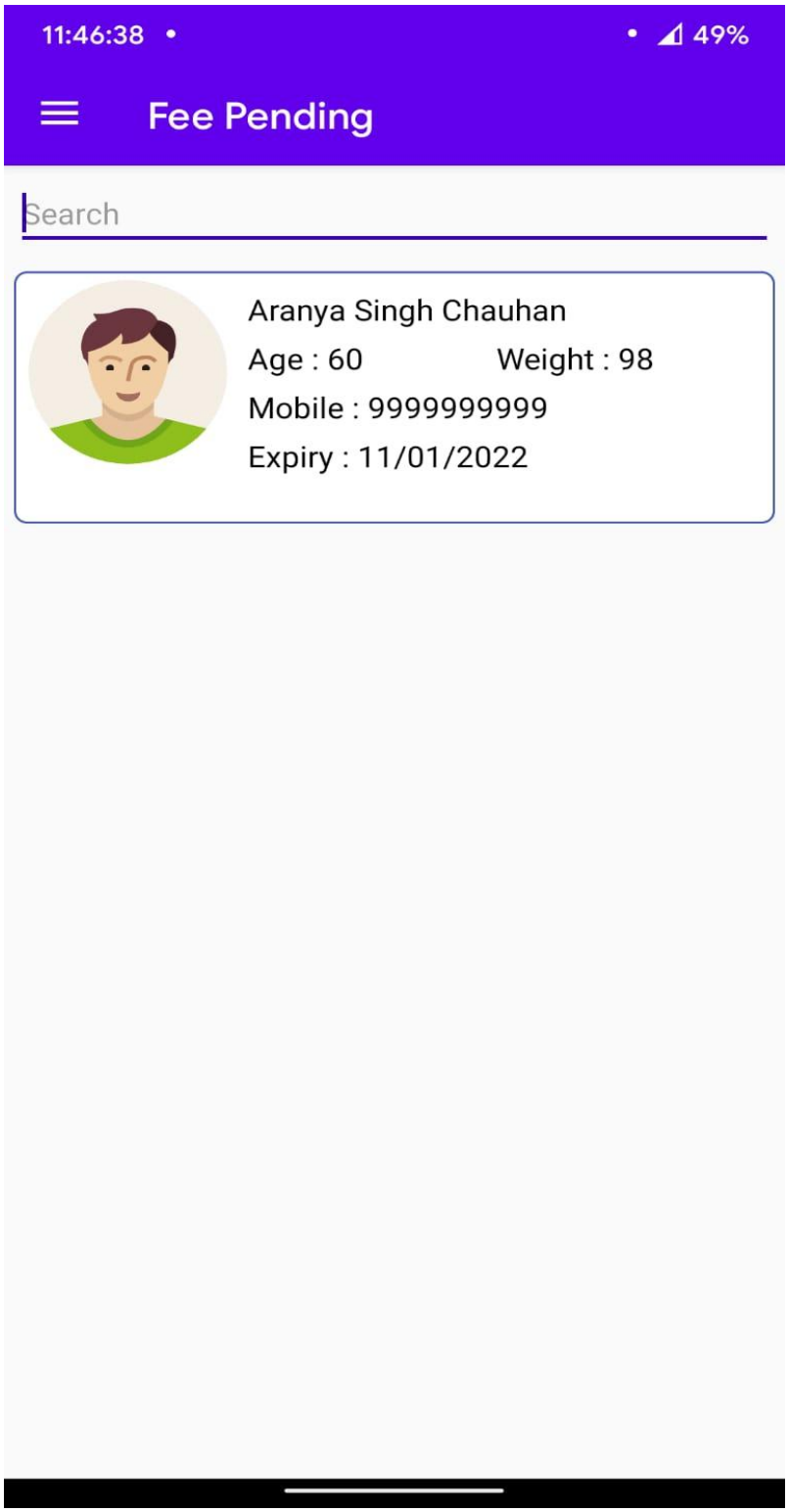


UI/UX:

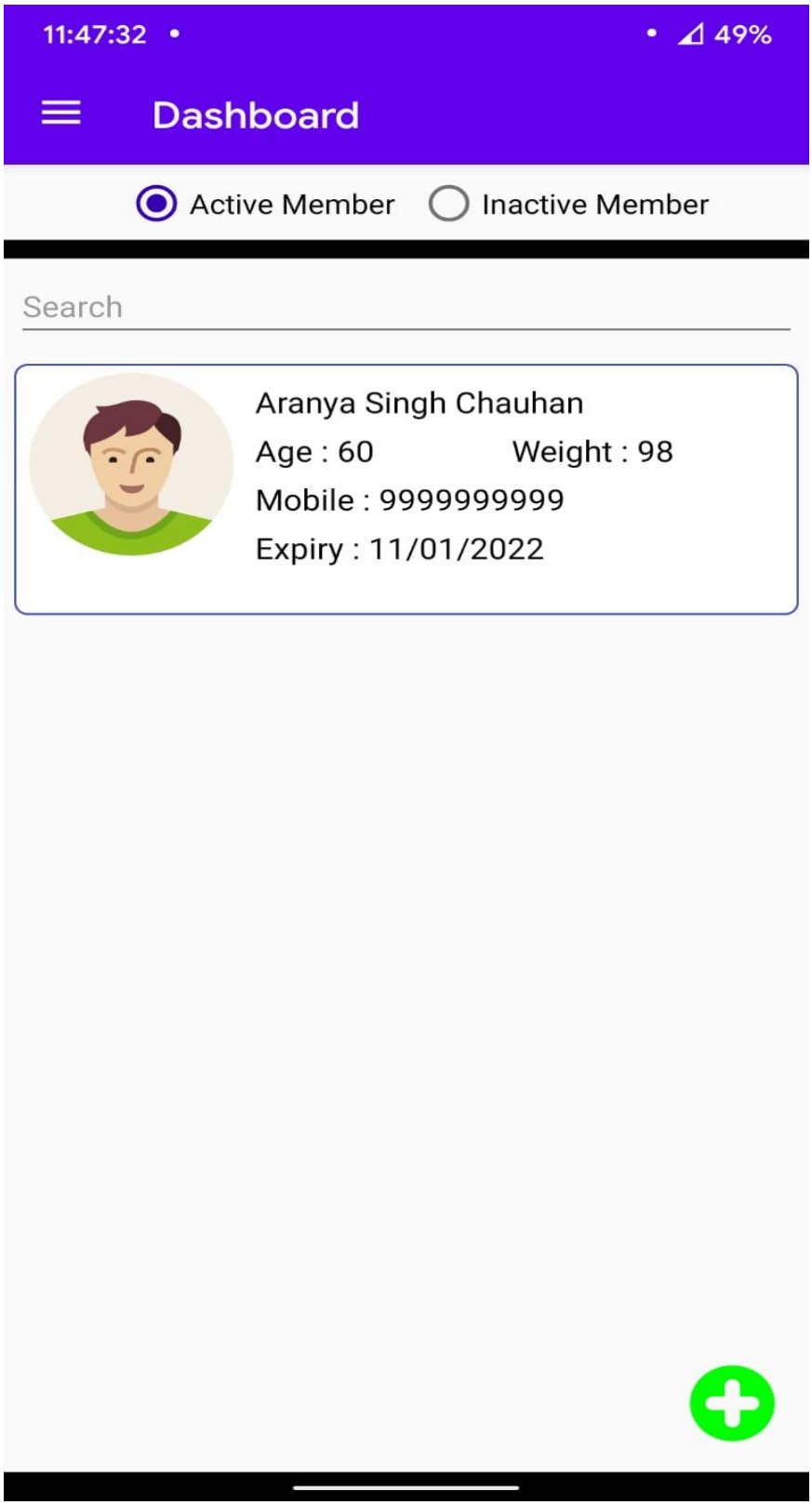
- Adding new gym members:



- Pending fee reminders:



- Dashboard/Active Members list:



-

Coding:

Splash Screen Activity

```
package com.example.primefitness

import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.content.Intent
import android.os.Binder
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.renderscript.ScriptGroup
import android.util.Log
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.primefitness.activity.HomeActivity
import com.example.primefitness.activity.LoginActivity
import com.example.primefitness.databinding.ActivityMainBinding
import com.example.primefitness.global.DB
import com.example.primefitness.manager.SessionManager
import java.lang.Exception

class SplashScreenActivity : AppCompatActivity() {

    private var mDelayHandler: Handler? = null
    private val splashDelay: Long = 1000 // 3 seconds
    var db: DB? = null
    var session: SessionManager? = null
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        db = DB(this)
        session = SessionManager(this)

        insertAdminData()
```

```
        mDelayHandler = Handler()
        mDelayHandler?.postDelayed(mRunnable, splashDelay)
    }

    private val mRunnable: Runnable = Runnable {
        if(session?.isLoggedIn == true){
            val intent = Intent(this, HomeActivity::class.java)
            startActivity(intent)
            finish()
        } else {
            val intent = Intent(this, LoginActivity::class.java)
            startActivity(intent)
            finish()
        }
    }

    private fun insertAdminData() {
        try {
            val sqlCheck = "SELECT * FROM ADMIN"
            db?.fireQuery(sqlCheck)?.use {
                if (it.count > 0) {
                    Log.d("SplashActivity", "data_available")
                } else {
                    val sqlQuery =
                        "INSERT OR REPLACE INTO ADMIN(ID,USER_NAME,PASSWORD,MOBILE)
VALUES('1','admin','5000','9999999999')"
                    db?.executeQuery(sqlQuery)
                }
            }
            val sqlQuery =
                "INSERT OR REPLACE INTO ADMIN(ID,USER_NAME,PASSWORD,MOBILE)
VALUES('1','admin','5000','9999999999')"
            db?.executeQuery(sqlQuery)
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        mDelayHandler?.removeCallbacks(mRunnable)
    }
}
```

Activity:

Home Activity

```
package com.example.primefitness.activity

import android.content.Intent
import android.content.res.Configuration
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.os.PersistableBundle
import android.view.Menu
import android.view.MenuItem
import android.widget.Toast
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.core.view.GravityCompat
import androidx.drawerlayout.widget.DrawerLayout
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
import com.example.primefitness.R
import com.example.primefitness.databinding.ActivityHomeBinding
import com.example.primefitness.fragment.*
import com.example.primefitness.global.DB
import com.example.primefitness.manager.SessionManager
import com.google.android.material.navigation.NavigationView
import java.lang.Exception
import kotlin.math.log

class HomeActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {
    private val TAG = "HomeActivity"
    var session: SessionManager? = null
    var db: DB? = null
    private lateinit var drawer: DrawerLayout
    private lateinit var toggle: ActionBarDrawerToggle
    lateinit var binding: ActivityHomeBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityHomeBinding.inflate(layoutInflater)
```

```
        setContentView(binding.root)

        db = DB(this)
        session = SessionManager(this)
        setSupportActionBar(binding.homeInclude.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        binding.navView.setNavigationItemSelectedListener(this)
        drawer = binding.drawerLayout

        toggle = ActionBarDrawerToggle(
            this, drawer, binding.homeInclude.toolbar,
            R.string.navigation_drawer_open, R.string.navigation_drawer_close
        )

        drawer.addDrawerListener(toggle)
        toggle.syncState()

        val fragment = FragmentAllMember()
        loadFragment(fragment)
    }

    override fun onPostCreate(savedInstanceState: Bundle?, persistentState:
PersistableBundle?) {
        super.onPostCreate(savedInstanceState, persistentState)
        toggle.syncState()
    }

    override fun onConfigurationChanged(newConfig: Configuration) {
        super.onConfigurationChanged(newConfig)
        toggle.onConfigurationChanged(newConfig)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (toggle.onOptionsItemSelected(item)) {
            return true
        }
        if(item.itemId == R.id.logOutMenu){
            logOut()
        }
        return super.onOptionsItemSelected(item)
    }

    override fun onNavigationItemSelected(item: MenuItem): Boolean {
        when (item.itemId) {
```

```
        R.id.nav_home -> {
            val fragment = FragmentAllMember()
            loadFragment(fragment)
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START)
            }
        }
        R.id.nav_add -> {
            loadFragment()
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START)
            }

        }
        R.id.nav_nav_fee_pending -> {
//              Toast.makeText(this, "Fee Pending", Toast.LENGTH_LONG).show()
            val fragment = FragmentFeePending()
            loadFragment(fragment)
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START)
            }
        }
        R.id.nav_update_fee -> {
           Toast.makeText(this, "Update Fee", Toast.LENGTH_LONG).show()
            val fragment = FragmentAppUpdateFee()
            loadFragment(fragment)
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START)
            }
        }
        R.id.nav_change_password -> {
            Toast.makeText(this, "Change Password", Toast.LENGTH_LONG).show()
            val fragment = FragmentChangePassword()
            loadFragment(fragment)
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START)
            }
        }
        R.id.nav_log_out -> {
            Toast.makeText(this, "Log Out", Toast.LENGTH_LONG).show()
            logOut()
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START)
            }
```

```
            }
            R.id.nav_import_export_database -> {
                Toast.makeText(this, "Home", Toast.LENGTH_LONG).show()
                val fragment = FragmentImportExportDatabase()
                loadFragment(fragment)
                if (drawer.isDrawerOpen(GravityCompat.START)) {
                    drawer.closeDrawer(GravityCompat.START)
                }
            }
        }

        return true
    }

    private fun logOut(){
        session?.setLogin(false)
        val intent = Intent(this, LoginActivity::class.java)
        startActivity(intent)
        finish()
    }

    private var doubleBackToExitPressedOnce = false
    override fun onBackPressed() {
        if(drawer.isDrawerOpen(GravityCompat.START)){
            drawer.closeDrawer(GravityCompat.START)
        }else{
            if (doubleBackToExitPressedOnce) {
                super.onBackPressed()
                return
            }
            this.doubleBackToExitPressedOnce = true
            Toast.makeText(this, "Please click BACK again to exit", Toast.LENGTH_SHORT).show()
            Handler(Looper.getMainLooper()).postDelayed(Runnable {
doubleBackToExitPressedOnce = false }, 2000)
        }
    }

    private fun loadFragment(fragment: Fragment){
        var fragmentManager: FragmentManager?= null
        fragmentManager = supportFragmentManager
        fragmentManager.beginTransaction().replace(R.id.frame_container, fragment,
"Home").commit()
    }
```

```
    private fun loadFragment(){
        val fragment = FragmentAddMember()
        val args = Bundle()
        args.putString("ID", "")
        fragment.arguments = args
        val fragmentManager:FragmentManager = supportFragmentManager
        fragmentManager.beginTransaction().replace(R.id.frame_container, fragment,
"FragmentAdd").commit()
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        try{
            val infalter = menuInflater
            infalter.inflate(R.menu.menu_main, menu)
        }catch (e:Exception){
            e.printStackTrace()
        }
        return super.onCreateOptionsMenu(menu)
    }

}
```

Login Activity

```
package com.example.primefitness.activity

import android.annotation.SuppressLint
import android.app.Dialog
import android.content.DialogInterface
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import com.example.primefitness.R
import com.example.primefitness.databinding.ActivityLoginBinding
import com.example.primefitness.databinding.ForgetPasswordDialogBinding
import com.example.primefitness.global.DB
import com.example.primefitness.global.MyFunction
import com.example.primefitness.manager.SessionManager
import java.lang.Exception

class LoginActivity : AppCompatActivity() {
    var db:DB?=null
    var session:SessionManager?=null
    var edtUserName : EditText?=null
    var edtPassword : EditText?= null
    lateinit var binding:ActivityLoginBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding= ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)
        db = DB(this)
        session= SessionManager(this)
        edtUserName = binding.edtUserName
        edtPassword = binding.edtPassword

        binding.btnLogin.setOnClickListener{
            if(validateLogin()){
```

```
            getLogin()
        }
    }

    binding.txtForgotPassword.setOnClickListener {
        showDialog()
    }



}
private fun getLogin(){
    try{
        val sqlQuery = "SELECT * FROM ADMIN WHERE
USER_NAME='"+edtUserName?.text.toString().trim()+"' " +
                "AND PASSWORD = '"+edtPassword?.text.toString().trim()+"' AND ID = '1'"
        db?.fireQuery(sqlQuery)?.use {
            if(it.count>0){
                session?.setLogin(true)
                Toast.makeText(this, "Successfully Logged In", Toast.LENGTH_LONG).show()
                val intent = Intent(this, HomeActivity::class.java)
                startActivity(intent)
                finish()
            } else {
                session?.setLogin(false)
                Toast.makeText(this, "Log In Failed", Toast.LENGTH_LONG).show()
            }
        }
    } catch (e:Exception){
        e.printStackTrace()
    }
}

private fun validateLogin():Boolean{
    if(edtUserName?.text.toString().trim().isEmpty()){
        Toast.makeText(this, "Enter User Name", Toast.LENGTH_LONG).show()
        return false
    } else if(edtPassword?.text.toString().trim().isEmpty()){
        Toast.makeText(this, "Enter Password", Toast.LENGTH_LONG).show()
        return false
    }
    return true
}
```

```
    private fun showDialog(){
        val binding2 = ForgetPasswordDialogBinding.inflate(LayoutInflater.from(this))
        val dialog = Dialog(this, R.style.AlertCustomDialog)
        dialog.setContentView(binding2.root)
        dialog.setCancelable(false)
        dialog.show()

        binding2.btnForgetSubmit.setOnClickListener {
            if(binding2.edtForgetMobile.text.toString().isNotEmpty()){
                checkData(binding2.edtForgetMobile.text.toString().trim(), binding2.txtYourPassword)
            } else {
                Toast.makeText(this, "Enter Mobile Number", Toast.LENGTH_LONG).show()
            }
        }

        binding2.imgBackButton.setOnClickListener {
            dialog.dismiss()
        }
    }

    @SuppressLint("SetTextI18n")
    private fun checkData(mobile:String, txtShowPassword:TextView){
        try {
            val sqlQuery = "SELECT * FROM ADMIN WHERE MOBILE='$mobile'"
            db?.fireQuery(sqlQuery)?.use {
                if(it.count>0){
                    val password = MyFunction.getvalue(it, "PASSWORD")
                    txtShowPassword.visibility = View.VISIBLE
                    txtShowPassword.text = "Your Password is : $password"
                } else {
                    Toast.makeText(this, "Incorrect Mobile Number", Toast.LENGTH_LONG).show()
                    txtShowPassword.visibility = View.GONE
                }
            }
        } catch (e:Exception){
            e.printStackTrace()
        }
    }
}
```

Adapter:

Adapter Load Member

```
package com.example.primefitness.adapter


import android.annotation.SuppressLint

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import androidx.recyclerview.widget.RecyclerView

import com.bumptech.glide.Glide

import com.example.primefitness.R

import com.example.primefitness.databinding.AllMemberListResBinding

import com.example.primefitness.model.AllMember


class AdapterLoadMember(var arrayList: ArrayList<AllMember>):
RecyclerView.Adapter<AdapterLoadMember.MyViewHolder>() {


    private var onClick: ((String)->Unit)?=null

    fun onClick(onClick:((String)->Unit)){

        this.onClick = onClick

    }
```

```
    class MyViewHolder(val binding:AllMemberListResBinding):
RecyclerView.ViewHolder(binding.root) {



    }



    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {

        val binding = AllMemberListResBinding.inflate(LayoutInflater.from(parent.context), parent,
false)



        return MyViewHolder(binding)

    }



    @SuppressLint("SetTextI18n")

    override fun onBindViewHolder(holder: AdapterLoadMember.MyViewHolder, position: Int) {

        with(holder){

            with(arrayList[position]){

                binding.txtAdapterName.text = this.firstName + " " + this.lastName

                binding.txtAdapterAge.text = "Age : " + this.age

                binding.txtAdapterWeight.text = "Weight : " + this.weight

                binding.txtAdapterMobile.text = "Mobile : " + this.mobile

                binding.txtAddress.text = this.address

                binding.txtExpiry.text = "Expiry : " + this.expiryDate


                if(this.image.isNotEmpty()){

                    Glide.with(holder.itemView.context)
```

```
                .load(this.image)
                .into(binding.imgAdapterPic)
            } else {
                if(this.gender == "Male"){
                    Glide.with(holder.itemView.context)
                        .load(R.drawable.boy)
                        .into(binding.imgAdapterPic)
                } else {
                    Glide.with(holder.itemView.context)
                        .load(R.drawable.girl)
                        .into(binding.imgAdapterPic)
                }
            }

        binding.layoutMemberList.setOnClickListener {
            onClick?.invoke(this.id)
        }
        }
    }
}

override fun getItemCount(): Int {
    return arrayList.size
}
```

```
    fun updateList(list:ArrayList<AllMember>){
        arrayList = list
        notifyDataSetChanged()
    }
}
```

Fragment:

Base Fragment

```
package com.example.primefitness.fragment;

import android.app.ProgressDialog;
import android.os.Bundle;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public abstract class BaseFragment extends Fragment {

    ProgressDialog mProgress;
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        mProgress = new ProgressDialog(getActivity());
    }

    public void showDialog(String msg){
        try {
            mProgress.setMessage(msg);
            mProgress.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            mProgress.setIndeterminate(false);
            mProgress.setCancelable(false);
            mProgress.show();
        } catch (Exception e){
            e.printStackTrace();
        }
    }

    public void CloseDialog(){
        if(mProgress!=null){
```

```
            mProgress.dismiss();
        }
    }
}
```

Fragment Add Member

```
package com.example.primefitness.fragment


import android.annotation.SuppressLint

import android.app.Activity.RESULT_OK

import android.app.DatePickerDialog

import android.content.Intent

import android.database.DatabaseUtils

import android.graphics.Bitmap

import android.net.Uri

import android.os.Build

import android.os.Bundle

import android.provider.MediaStore

import android.util.Log

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.Toast

import androidx.annotation.RequiresApi

import androidx.fragment.app.Fragment

import com.bumptech.glide.Glide
```

```
import com.example.primefitness.R

import com.example.primefitness.databinding.FragmentAddMemberBinding

import com.example.primefitness.databinding.RenewDialogBinding

import com.example.primefitness.global.CaptureImage

import com.example.primefitness.global.DB

import com.example.primefitness.global.MyFunction

import com.github.florent37.runtimepermission.RuntimePermission

import com.squareup.picasso.Picasso

import java.io.File

import java.text.SimpleDateFormat

import java.util.*



class FragmentAddMember : Fragment() {

    var db:DB?=null

    private lateinit var binding:FragmentAddMemberBinding

    private lateinit var bindingDialog:RenewDialogBinding

    private var captureImage:CaptureImage?=null

    private var captureImageID1:CaptureImage?=null

    private var captureImageID2:CaptureImage?=null

    private var actualImagePath = ""

    private var actualImagePathID1 = ""

    private var actualImagePathID2 = ""

    private var gender = "Male"
```

```
    private var ID = ""

    override fun onCreateView(

        inflater: LayoutInflater,

        container: ViewGroup?,

        savedInstanceState: Bundle?

    ): View? {

        // Inflate the layout for this fragment

        binding = FragmentAddMemberBinding.inflate(inflater, container, false)

        return binding.root

    }


    @SuppressLint("UseRequireInsteadOfGet")

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

        super.onViewCreated(view, savedInstanceState)

        activity?.title = "Add New Member"

        db = activity?.let { DB(it) }

        captureImage = CaptureImage(activity)

        captureImageID1 = CaptureImage(activity)

        captureImageID2 = CaptureImage(activity)


        ID = arguments!!.getString("ID").toString()


        val cal = Calendar.getInstance()

        val dateSetListener = DatePickerDialog.OnDateSetListener{ view1, year, monthOfYear,
dayOfMonth ->
```

```
            cal.set(Calendar.YEAR, year)
            cal.set(Calendar.MONTH, monthOfYear)
            cal.set(Calendar.DAY_OF_MONTH, dayOfMonth)

            val myFormat = "dd/MM/yyyy"
            val sdf = SimpleDateFormat(myFormat, Locale.US)
            binding.edtJoining.setText(sdf.format(cal.time))

        }

        val cal2 = Calendar.getInstance()
        val dateSetListener2 = DatePickerDialog.OnDateSetListener{ view2, year2, monthOfYear2,
dayOfMonth2 ->
            cal2.set(Calendar.YEAR, year2)
            cal2.set(Calendar.MONTH, monthOfYear2)
            cal2.set(Calendar.DAY_OF_MONTH, dayOfMonth2)

            val myFormat2 = "dd/MM/yyyy"
            val sdf2 = SimpleDateFormat(myFormat2, Locale.US)
            binding.edtExpire.setText(sdf2.format(cal2.time))

        }

        binding.radioGroup.setOnCheckedChangeListener { radioGroup, id ->
```

```
    when(id){
        R.id.rdMale -> {
            gender = "Male"
        }
        R.id.rdFemale -> {
            gender = "Female"
        }
    }
}

binding.btnAddMemberSave.setOnClickListener {



    if(validate()){
        saveData()
    }
}

binding.imgPicDate.setOnClickListener {
    activity?.let { it1 -> DatePickerDialog(
        it1, dateSetListener, cal.get(Calendar.YEAR), cal.get(
            Calendar.MONTH
        ), cal.get(Calendar.DAY_OF_MONTH)
    ).show() }
```

```
        }

        binding.imgPicDateExpiry.setOnClickListener {
            activity?.let { it2 -> DatePickerDialog(
                it2, dateSetListener2, cal2.get(Calendar.YEAR), cal2.get(
                    Calendar.MONTH
                ), cal2.get(Calendar.DAY_OF_MONTH)
            ).show() }
        }

        binding.imgTakeImage.setOnClickListener {
//          getImage()
            selectImage()
        }

        binding.imgTakeIDImage1.setOnClickListener {
            selectImageID1()
        }

        binding.imgTakeIDImage2.setOnClickListener {
            selectImageID2()
        }

        binding.btnActiveInactive.setOnClickListener {
```

```
    try {
        if(getStatus() == "A"){
            val sqlQuery = "UPDATE MEMBER SET STATUS='D' WHERE ID='$ID'"
            db?.fireQuery(sqlQuery)
            showToast("Member is Inactive now")

        } else {
            val sqlQuery = "UPDATE MEMBER SET STATUS='A' WHERE ID='$ID'"
            db?.fireQuery(sqlQuery)
            showToast("Member is Active now")
        }
    } catch (e: Exception){
        e.printStackTrace()
    }
}

if(ID.trim().isNotEmpty()){
    if(getStatus() == "A"){
        binding.btnActiveInactive.text = "Inactive"
        binding.btnActiveInactive.visibility = View.VISIBLE
    } else {
        binding.btnActiveInactive.text = "Active"
        binding.btnActiveInactive.visibility = View.VISIBLE
    }
```

```
        loadData()
    } else {
        binding.btnActiveInactive.visibility = View.GONE
    }



    binding.btnDelete.setOnClickListener {
        if(ID.trim().isNotEmpty()){
            deleteEntry()
        }
    }

}

private fun selectImage(){
    val items:Array<CharSequence>
    try {
        items = arrayOf("Take Photo", "Choose Image", "Cancel")
        val builder = android.app.AlertDialog.Builder(activity)
        builder.setCancelable(false)
        builder.setTitle("Select Image")
        builder.setItems(items) { dialoginterface, i ->
            if (items[i] == "Take Photo"){
                RuntimePermission.askPermission(this)
```

```
                .request(android.Manifest.permission.CAMERA)
                .onAccepted {
                    val takePicture = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
                    startActivityForResult(takePicture, 0) //zero can be replaced with any action
code (called requestCode)
                }
                .onDenied {
                    android.app.AlertDialog.Builder(activity)
                        .setMessage("Please accept the permission to capture!")
                        .setPositiveButton("Yes") { dialoginterface, i ->
                            it.askAgain()
                        }
                        .setNegativeButton("No") { dialoginterface, i ->
                            dialoginterface.dismiss()
                        }
                        .show()
                }
                .ask()
            }else if(items[i] == "Choose Image"){


                RuntimePermission.askPermission(this)
                .request(android.Manifest.permission.WRITE_EXTERNAL_STORAGE)
                .onAccepted {
                    val pickPhoto = Intent(
                        Intent.ACTION_PICK,
```

```
                    MediaStore.Images.Media.EXTERNAL_CONTENT_URI
                )
                startActivityForResult(pickPhoto, 1) //one can be replaced with any action code


            }
            .onDenied {
                android.app.AlertDialog.Builder(activity)
                    .setMessage("Please accept the permission to select!")
                    .setPositiveButton("Yes") { dialoginterface, i ->
                        it.askAgain()
                    }
                    .setNegativeButton("No") { dialoginterface, i ->
                        dialoginterface.dismiss()
                    }
                    .show()
            }
            .ask()
        }else{
            dialoginterface.dismiss()
        }
    }
    builder.show()
} catch (e: Exception){
    e.printStackTrace()
```

```
        }
    }


    private fun selectImageID1(){
        val items:Array<CharSequence>
        try {
            items = arrayOf("Take Photo", "Choose Image", "Cancel")
            val builder = android.app.AlertDialog.Builder(activity)
            builder.setCancelable(false)
            builder.setTitle("Select Image")
            builder.setItems(items) { dialoginterface, i ->
                if (items[i] == "Take Photo"){
                    RuntimePermission.askPermission(this)
                        .request(android.Manifest.permission.CAMERA)
                        .onAccepted {
                            val takePicture = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
                            startActivityForResult(takePicture, 2) //zero can be replaced with any action
code (called requestCode)
                        }
                        .onDenied {
                            android.app.AlertDialog.Builder(activity)
                                .setMessage("Please accept the permission to capture!")
                                .setPositiveButton("Yes") { dialoginterface, i ->
                                    it.askAgain()
                                }
```

```
                .setNegativeButton("No") { dialoginterface, i ->
                    dialoginterface.dismiss()
                }
                .show()
            }
            .ask()
        }else if(items[i] == "Choose Image"){


            RuntimePermission.askPermission(this)
                .request(android.Manifest.permission.WRITE_EXTERNAL_STORAGE)
                .onAccepted {
                    val pickPhoto = Intent(
                        Intent.ACTION_PICK,
                        MediaStore.Images.Media.EXTERNAL_CONTENT_URI
                    )
                    startActivityForResult(pickPhoto, 3) //one can be replaced with any action
code


                }
                .onDenied {
                    android.app.AlertDialog.Builder(activity)
                        .setMessage("Please accept the permission to select!")
                        .setPositiveButton("Yes") { dialoginterface, i ->
                            it.askAgain()
                        }
```

```
                    .setNegativeButton("No") { dialoginterface, i ->
                        dialoginterface.dismiss()
                    }
                    .show()
                }
                .ask()
            }else{
                dialoginterface.dismiss()
            }
        }
        builder.show()
    } catch (e: Exception){
        e.printStackTrace()
    }
}

private fun selectImageID2(){
    val items:Array<CharSequence>
    try {
        items = arrayOf("Take Photo", "Choose Image", "Cancel")
        val builder = android.app.AlertDialog.Builder(activity)
        builder.setCancelable(false)
        builder.setTitle("Select Image")
        builder.setItems(items) { dialoginterface, i ->
```

```
            if (items[i] == "Take Photo"){
                RuntimePermission.askPermission(this)
                    .request(android.Manifest.permission.CAMERA)
                    .onAccepted {
                        val takePicture = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
                        startActivityForResult(takePicture, 4) //zero can be replaced with any action
code (called requestCode)
                    }
                    .onDenied {
                        android.app.AlertDialog.Builder(activity)
                            .setMessage("Please accept the permission to capture!")
                            .setPositiveButton("Yes") { dialoginterface, i ->
                                it.askAgain()
                            }
                            .setNegativeButton("No") { dialoginterface, i ->
                                dialoginterface.dismiss()
                            }
                            .show()
                    }
                    .ask()
            }else if(items[i] == "Choose Image"){


                RuntimePermission.askPermission(this)
                    .request(android.Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    .onAccepted {
```

```
                val pickPhoto = Intent(
                    Intent.ACTION_PICK,
                    MediaStore.Images.Media.EXTERNAL_CONTENT_URI
                )
                startActivityForResult(pickPhoto, 5) //one can be replaced with any action
code


            }
            .onDenied {
                android.app.AlertDialog.Builder(activity)
                    .setMessage("Please accept the permission to select!")
                    .setPositiveButton("Yes") { dialoginterface, i ->
                        it.askAgain()
                    }
                    .setNegativeButton("No") { dialoginterface, i ->
                        dialoginterface.dismiss()
                    }
                    .show()
            }
            .ask()
        }else{
            dialoginterface.dismiss()
        }
    }
    builder.show()
```

```
        } catch (e: Exception){
            e.printStackTrace()
        }
    }



    @RequiresApi(Build.VERSION_CODES.KITKAT)
    override fun onActivityResult(requestCode: Int, resultCode: Int, imageReturnedIntent: Intent?)
{
        super.onActivityResult(requestCode, resultCode, imageReturnedIntent)
        when (requestCode) {
            0 -> if (resultCode == RESULT_OK) {
                val selectedImage = imageReturnedIntent?.data
                binding.imgPic.setImageURI(selectedImage)

captureImage(captureImage?.getRightAngleImage(captureImage?.imagePath).toString())
            }
            1 -> if (resultCode == RESULT_OK) {
                val selectedImage = imageReturnedIntent?.data
                binding.imgPic.setImageURI(selectedImage)

captureImage(captureImage?.getRightAngleImage(captureImage?.getPath(imageReturnedInte
nt?.data, context)).toString()
                )
            }
            2 -> if (resultCode == RESULT_OK) {
```

```
            val selectedImage = imageReturnedIntent?.data
            binding.imgIDPic1.setImageURI(selectedImage)
            captureImageID1(
                captureImageID1?.getRightAngleImage(captureImageID1?.imagePath).toString()
            )
        }
        3 -> if (resultCode == RESULT_OK) {
            val selectedImage = imageReturnedIntent?.data
            binding.imgIDPic1.setImageURI(selectedImage)
            captureImageID1(

captureImageID1?.getRightAngleImage(captureImageID1?.getPath(imageReturnedIntent?.data
, context)).toString()
            )
        }
        4 -> if (resultCode == RESULT_OK) {
            val selectedImage = imageReturnedIntent?.data
            binding.imgIDPic2.setImageURI(selectedImage)
            captureImageID2(
                captureImageID2?.getRightAngleImage(captureImageID2?.imagePath).toString()
            )
        }
        5 -> if (resultCode == RESULT_OK) {
            val selectedImage = imageReturnedIntent?.data
            binding.imgIDPic2.setImageURI(selectedImage)
```

```
            captureImageID2(

captureImageID2?.getRightAngleImage(captureImageID2?.getPath(imageReturnedIntent?.data
, context)).toString()

            )

        }

    }

}


    private fun getStatus():String{

        var status = ""

        try {

            val sqlQuery = "SELECT STATUS FROM MEMBER WHERE ID='$ID'"

            db?.fireQuery(sqlQuery)?.use {

                if(it.count>0){

                    status = MyFunction.getvalue(it, "STATUS")

                }

            }

        } catch (e: Exception){

            e.printStackTrace()

        }

        return status

    }


    private fun captureImage(path: String){
```

```
        Log.d("FragmentAdd", "imagePath : $path")
        getImagePath(captureImage?.decodeFile(path))
    }

    private fun getImagePath(bitmap: Bitmap?){
        val tempUri: Uri? = captureImage?.getImageUri(activity, bitmap)
        actualImagePath = captureImage?.getRealPathFromURI(tempUri, activity).toString()
        Log.d("FragmentAdd", "ActualImagePath : ${actualImagePath}")
        context?.let { Picasso.with(it).load(tempUri).into(binding.imgPic) }

    }

    private fun captureImageID1(path: String){
        Log.d("FragmentAdd", "imagePath : $path")
        getImagePathID1(captureImageID1?.decodeFile(path))
    }

    private fun getImagePathID1(bitmap: Bitmap?){
        val tempUri: Uri? = captureImageID1?.getImageUri(activity, bitmap)
        actualImagePathID1 = captureImageID1?.getRealPathFromURI(tempUri,
activity).toString()
        Log.d("FragmentAdd", "ActualImagePath : ${actualImagePathID1}")

        context?.let { Picasso.with(it).load(tempUri).into(binding.imgIDPic1) }
```

```
    }

    private fun captureImageID2(path: String){
        Log.d("FragmentAdd", "imagePath : $path")
        getImagePathID2(captureImageID2?.decodeFile(path))
    }

    private fun getImagePathID2(bitmap: Bitmap?){
        val tempUri: Uri? = captureImageID2?.getImageUri(activity, bitmap)
        actualImagePathID2 = captureImageID2?.getRealPathFromURI(tempUri,
activity).toString()
        Log.d("FragmentAdd", "ActualImagePath : ${actualImagePathID2}")

        context?.let { Picasso.with(it).load(tempUri).into(binding.imgIDPic2) }

    }

    private fun validate():Boolean{
        if(binding.edtFirstName.text.toString().trim().isEmpty()){
            showToast("Enter First Name")
            return false
        }else if(binding.edtMobile.text.toString().trim().isEmpty()){
            showToast("Enter Mobile Number")
            return false
        }
```

```
        return true
    }


    private fun saveData(){
        try {
            var myIncrementId = ""
            if(ID.trim().isEmpty()){
                myIncrementId = getIncrementedId()
            } else {
                myIncrementId = ID
            }
            val sqlQuery = "INSERT OR REPLACE INTO MEMBER(ID, FIRST_NAME,
LAST_NAME, GENDER, AGE, WEIGHT, MOBILE, ADDRESS, DATE_OF_JOINING,
EXPIRE_ON, TOTAL, IMAGE_PATH, STATUS, IMAGE_PATH_ID_1, IMAGE_PATH_ID_2,
DESCRIPTION) VALUES " +
                    "('"+myIncrementId+"','"+DatabaseUtils.sqlEscapeString(
                binding.edtFirstName.text.toString().trim()
            )+"'," +
                    "'"+DatabaseUtils.sqlEscapeString(binding.edtLastName.text.toString().trim())+"',
'"+gender+"', " +
                    "'"+binding.edtAge.text.toString().trim()+"',
'"+binding.edtWeight.text.toString().trim()+"'," +
                    "'"+binding.edtMobile.text.toString().trim()+"','"+DatabaseUtils.sqlEscapeString(
                binding.edtAddress.text.toString().trim()
            )+"'," +
                    "'"+MyFunction.returnSQLdateFormat(binding.edtJoining.text.toString().trim())+"'," +
                    "'"+MyFunction.returnSQLdateFormat(binding.edtExpire.text.toString().trim())+"', " +
```

```
                """+binding.edtAmount.text.toString().trim()+"', '"+actualImagePath+"', 'A',
'"+actualImagePathID1+"', '"+actualImagePathID2+"', "+DatabaseUtils.sqlEscapeString(

            binding.edtDescription.text.toString().trim()

        )+")"

        db?.executeQuery(sqlQuery)

        showToast("Data Saved Successfully!")

        if(ID.trim().isEmpty()){

            clearData()

        }

    } catch (e: Exception){

        e.printStackTrace()

    }

}


private fun getIncrementedId():String{

    var incrementId = ""

    try {

        val sqlQuery = "SELECT IFNULL(MAX(ID)+1, '1') AS ID FROM MEMBER"

        db?.fireQuery(sqlQuery)?.use {

            if(it.count>0){

            incrementId = MyFunction.getvalue(it, "ID")}

        }

    } catch (e: Exception){

        e.printStackTrace()

    }
```

```
    return incrementId
}

private fun clearData(){
    binding.edtFirstName.setText("")
    binding.edtLastName.setText("")
    binding.edtAge.setText("")
    binding.edtWeight.setText("")
    binding.edtMobile.setText("")
    binding.edtAddress.setText("")
    binding.edtJoining.setText("")
    binding.edtExpire.setText("")
    binding.edtAmount.setText("")
    binding.edtDescription.setText("")
    actualImagePathID1=""
    actualImagePathID2=""
    actualImagePath=""

    Glide.with(this)
        .load(R.drawable.boy)
        .into(binding.imgPic)

    Glide.with(this)
        .load(R.drawable.document)
```

```
        .into(binding.imgIDPic1)

    Glide.with(this)
        .load(R.drawable.document)
        .into(binding.imgIDPic2)

}



@SuppressLint("SimpleDateFormat")
private fun loadData(){
    try {
        val sqlQuery = "SELECT * FROM MEMBER WHERE ID='$ID'"
        db?.fireQuery(sqlQuery)?.use {
            if(it.count>0){
                val firstName = MyFunction.getvalue(it, "FIRST_NAME")
                val lastName = MyFunction.getvalue(it, "LAST_NAME")
                val age = MyFunction.getvalue(it, "AGE")
                val gender = MyFunction.getvalue(it, "GENDER")
                val weight = MyFunction.getvalue(it, "WEIGHT")
                val mobileNo = MyFunction.getvalue(it, "MOBILE")
                val address = MyFunction.getvalue(it, "ADDRESS")
                val dateOfJoin = MyFunction.getvalue(it, "DATE_OF_JOINING")
                val expiry = MyFunction.getvalue(it, "EXPIRE_ON")
```

```
val total = MyFunction.getvalue(it, "TOTAL")
val description = MyFunction.getvalue(it, "DESCRIPTION")
actualImagePath = MyFunction.getvalue(it, "IMAGE_PATH")
actualImagePathID1 = MyFunction.getvalue(it, "IMAGE_PATH_ID_1")
actualImagePathID2 = MyFunction.getvalue(it, "IMAGE_PATH_ID_2")

binding.edtFirstName.setText(firstName)
binding.edtLastName.setText(lastName)
binding.edtAge.setText(age)
binding.edtWeight.setText(weight)
binding.edtMobile.setText(mobileNo)
binding.edtAddress.setText(address)
binding.edtJoining.setText(MyFunction.returnUserdateFormat(dateOfJoin))
binding.edtExpire.setText(MyFunction.returnUserdateFormat(expiry))
binding.edtDescription.setText(description)

if(actualImagePath.isNotEmpty()){
    val uri = Uri.fromFile(File(actualImagePath))
    context?.let { it1 -> Picasso.with(it1).load(uri).into(binding.imgPic) }
}
else {
    if(gender=="Male"){
        Glide.with(this)
            .load(R.drawable.boy)
```

```
            .into(binding.imgPic)
    } else{
        Glide.with(this)
            .load(R.drawable.girl)
            .into(binding.imgPic)
    }
}

if(actualImagePathID1.isNotEmpty()){
    val uri = Uri.fromFile(File(actualImagePathID1))
    context?.let { it1 -> Picasso.with(it1).load(uri).into(binding.imgIDPic1) }
}

else {
        Glide.with(this)
            .load(R.drawable.document)
            .into(binding.imgIDPic1)
    }

if(actualImagePathID2.isNotEmpty()){
    val uri = Uri.fromFile(File(actualImagePathID2))
    context?.let { it1 -> Picasso.with(it1).load(uri).into(binding.imgIDPic2) }
}
else {
```

```
                Glide.with(this)
                    .load(R.drawable.document)
                    .into(binding.imgIDPic2)
                }

            if (gender == "Male"){
                binding.radioGroup.check(R.id.rdMale)
            } else {
                binding.radioGroup.check(R.id.rdFemale)
            }



            binding.edtAmount.setText(total)
        }
    }
  } catch (e: Exception){
    e.printStackTrace()
  }
}

private fun deleteEntry(){
  try {
    val sqlQuery = "DELETE FROM MEMBER WHERE ID='$ID'"
    db?.executeQuery(sqlQuery)
```

```
            showToast("Member deleted successfully!")
            clearData()
        } catch (e: Exception){
            e.printStackTrace()
        }
    }
}
```

Fragment All Members

```
package com.example.primefitness.fragment


import android.os.Bundle

import android.text.Editable

import android.text.TextWatcher

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import androidx.fragment.app.FragmentManager

import androidx.lifecycle.lifecycleScope

import androidx.recyclerview.widget.LinearLayoutManager

import com.example.primefitness.R

import com.example.primefitness.adapter.AdapterLoadMember

import com.example.primefitness.databinding.FragmentAllMemberBinding

import com.example.primefitness.global.DB

import com.example.primefitness.global.MyFunction

import com.example.primefitness.model.AllMember

import kotlinx.coroutines.CoroutineScope

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.launch

import kotlinx.coroutines.withContext

import java.util.*
```

```
import kotlin.collections.ArrayList


class FragmentAllMember : BaseFragment() {

    private val TAG = "FragmentAllMember"

    var db:DB?=null

    var adapter:AdapterLoadMember?=null

    var arrayList:ArrayList<AllMember> = ArrayList()

    private lateinit var binding: FragmentAllMemberBinding


    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {

        // Inflate the layout for this fragment

        binding = FragmentAllMemberBinding.inflate(inflater, container, false)

        return binding.root

    }


    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

        super.onViewCreated(view, savedInstanceState)

        activity?.title = "Dashboard"

        db = activity?.let { DB(it) }


        binding.rdGroupMember.setOnCheckedChangeListener { radioGroup, id ->

            when(id){

                R.id.rdActiveMember -> {

                    loadData("A")
```

```
        }
        R.id.rdInActiveMember -> {
            loadData("D")
        }
    }
}

binding.imgAddMember.setOnClickListener {
    loadFragment("")
}

binding.edtAllMemberSearch.addTextChangedListener(object : TextWatcher {
    override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {

    }

    override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {

    }

    override fun afterTextChanged(s: Editable?) {
        myFilter(s.toString())
    }
```

```
    })
}

override fun onResume() {
    super.onResume()
    loadData("A")
}

fun <R> CoroutineScope.executeAsyncTask(
    onPreExecute: () -> Unit,
    doInBackground: () -> R,
    onPostExecute: (R) -> Unit
) = launch {
    onPreExecute()
    val result = withContext(Dispatchers.IO){
        doInBackground()
    }
    onPostExecute(result)
}

private fun loadData(memberStatus:String){
    lifecycleScope.executeAsyncTask(onPreExecute = {
        showDialog("Processing....")
```

```
}, doInBackground = {
    arrayList.clear()
    val sqlQuery = "SELECT * FROM MEMBER WHERE STATUS='$memberStatus'"
    db?.fireQuery(sqlQuery)?.use {
        if(it.count>0){
          it.moveToFirst()
            do {
                val list = AllMember(id = MyFunction.getvalue(it, "ID"),
                    firstName = MyFunction.getvalue(it, "FIRST_NAME"),
                    lastName = MyFunction.getvalue(it, "LAST_NAME"),
                    age = MyFunction.getvalue(it, "AGE"),
                    gender = MyFunction.getvalue(it, "GENDER"),
                    weight = MyFunction.getvalue(it, "WEIGHT"),
                    mobile = MyFunction.getvalue(it, "MOBILE"),
                    address = MyFunction.getvalue(it, "ADDRESS"),
                    image = MyFunction.getvalue(it, "IMAGE_PATH"),
                    dateOfJoining = MyFunction.returnUserdateFormat(MyFunction.getvalue(it,
"DATE_OF_JOINING")),
                    expiryDate = MyFunction.returnUserdateFormat(MyFunction.getvalue(it,
"EXPIRE_ON")),
                    description = MyFunction.getvalue(it, "DESCRIPTION"))


                arrayList.add(list)
            } while (it.moveToNext())
```

```
            }
        }
    }, onPostExecute = {
        if(arrayList.size>0){
            binding.recyclerViewMember.visibility = View.VISIBLE
            binding.txtAllMemberNDF.visibility = View.GONE
            adapter = AdapterLoadMember(arrayList)
            binding.recyclerViewMember.layoutManager = LinearLayoutManager(activity)
            binding.recyclerViewMember.adapter = adapter



            adapter?.onClick {
                loadFragment(it)
            }

        } else {
            binding.recyclerViewMember.visibility = View.GONE
            binding.txtAllMemberNDF.visibility = View.VISIBLE
        }
        CloseDialog()
    })
}

private fun loadFragment(id:String){
```

```
        val fragment = FragmentAddMember()

        val args = Bundle()

        args.putString("ID", id)

        fragment.arguments = args

        val fragmentManager:FragmentManager?= fragmentManager

        fragmentManager!!.beginTransaction().replace(R.id.frame_container, fragment,
"FragmentAdd").commit()

    }



    private fun myFilter(searchValue:String){

        val temp:ArrayList<AllMember> = ArrayList()

        if(arrayList.size>0){

            for(list in arrayList){


if(list.firstName.toLowerCase(Locale.ROOT).contains(searchValue.toLowerCase(Locale.ROOT)
) || list.lastName.toLowerCase(

                        Locale.ROOT).contains(searchValue.toLowerCase(Locale.ROOT))){

                    temp.add(list)

                }

            }

        }

        adapter?.updateList(temp)

    }

}
```

Fragment App Update Fee

```
package com.example.primefitness.fragment


import android.os.Bundle

import androidx.fragment.app.Fragment

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.Toast

import com.example.primefitness.R

import com.example.primefitness.databinding.FragmentAddMemberBinding

import com.example.primefitness.databinding.FragmentAppUpdateFeeBinding

import com.example.primefitness.global.DB

import com.example.primefitness.global.MyFunction

import java.lang.Exception



class FragmentAppUpdateFee : Fragment() {


    private lateinit var binding: FragmentAppUpdateFeeBinding

    var db : DB?=null

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {

        // Inflate the layout for this fragment

        binding = FragmentAppUpdateFeeBinding.inflate(inflater, container, false)
```

```
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    super.onViewCreated(view, savedInstanceState)
    activity?.title = "Fee"
    db = activity?.let { DB(it) }
    binding.btnAddMemberShip.setOnClickListener {
        if(validate()){
            saveData()
        }
    }
    fillData()
}

private fun validate():Boolean{
    if(binding.edtOneMonth.text.toString().isEmpty()){
        showToast("Enter One Month Fees")
    } else if(binding.edtThreeMonth.text.toString().isEmpty()){
        showToast("Enter Three Month Fees")
    }
    return true
}
```

```
    private fun saveData(){


        try{

            val sqlQuery = "INSERT OR REPLACE INTO FEE(ID, ONE_MONTH, THREE_MONTH)
VALUES ('1', '"+binding.edtOneMonth.text.toString().trim()+"',
'"+binding.edtThreeMonth.text.toString().trim()+"')"

            db?.executeQuery(sqlQuery)

            showToast("Membership Data Saved Successfully")

        } catch (e:Exception){

            e.printStackTrace()

        }

    }


    private fun fillData(){

        try {

            val sqlQuery = "SELECT * FROM FEE WHERE ID = '1'"

            db?.fireQuery(sqlQuery)?.use {

                if(it.count>0){

                    it.moveToFirst()

                    binding.edtOneMonth.setText(MyFunction.getvalue(it, "ONE_MONTH"))

                    binding.edtThreeMonth.setText(MyFunction.getvalue(it, "THREE_MONTH"))

                }

            }

        } catch (e:Exception){
```

```
            e.printStackTrace()
        }
    }


    private fun showToast(value:String){
        Toast.makeText(activity, value, Toast.LENGTH_LONG).show()
    }
}
```

Fragment Change Password

```
package com.example.primefitness.fragment


import android.database.DatabaseUtils

import android.os.Bundle

import androidx.fragment.app.Fragment

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.Toast

import com.example.primefitness.R

import com.example.primefitness.databinding.FragmentAddMemberBinding

import com.example.primefitness.databinding.FragmentChangePasswordBinding

import com.example.primefitness.global.DB

import com.example.primefitness.global.MyFunction

import java.lang.Exception



class FragmentChangePassword : Fragment() {

    private lateinit var binding: FragmentChangePasswordBinding

    private var db:DB?=null

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {

        // Inflate the layout for this fragment

        binding = FragmentChangePasswordBinding.inflate(inflater, container, false)
```

```
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    activity?.title = "Change Password"
    db = activity?.let { DB(it) }

    fillOldMobile()

    binding.btnChangePassword.setOnClickListener {
        try {
            if(binding.edtNewPassword.text.toString().trim().isNotEmpty()) {
                if (binding.edtNewPassword.text.toString()
                        .trim() == binding.edtConfirmPassword.text.toString().trim()
                ) {
                    val sqlQuery =
                        "UPDATE ADMIN SET PASSWORD=" + DatabaseUtils.sqlEscapeString(
                            binding.edtNewNumber.text.toString().trim()
                        ) + ""
                    db?.executeQuery(sqlQuery)
                    showToast("Password changed successfully")
                } else {
                    showToast("Passwords do not match!")
```

```
                }
            } else {
                showToast("Enter new password")
            }
        } catch (e:Exception){
            e.printStackTrace()
        }
    }


    binding.btnChangeMobile.setOnClickListener {
        try {
            if(binding.edtNewNumber.text.toString().trim().isNotEmpty()){
            val sqlQuery = "UPDATE ADMIN SET
MOBILE='"+binding.edtNewNumber.text.toString().trim()+"'"
            db?.executeQuery(sqlQuery)
            showToast("Mobile number changed successfully")
            } else {
                showToast("Enter new mobile number")
            }
        } catch (e:Exception){
            e.printStackTrace()
        }
    }
}
```

```
    private fun fillOldMobile(){
        try {
            val sqlQuery = "SELECT MOBILE FROM ADMIN"
            db?.fireQuery(sqlQuery)?.use {
                val mobile = MyFunction.getvalue(it, "MOBILE")
                if(mobile.trim().isNotEmpty()){
                    binding.edtOldNumber.setText(mobile)
                }
            }
        } catch (e:Exception){
            e.printStackTrace()
        }
    }

    private fun showToast(value:String){
        Toast.makeText(activity, value, Toast.LENGTH_LONG).show()
    }
}
```

Fragment Fee Pending

```
package com.example.primefitness.fragment


import android.app.Notification

import android.app.NotificationChannel

import android.app.NotificationManager

import android.app.PendingIntent

import android.content.Context

import android.content.Intent

import android.graphics.BitmapFactory

import android.graphics.Color

import android.os.Build

import android.os.Bundle

import android.text.Editable

import android.text.TextWatcher

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.RemoteViews

import androidx.annotation.RequiresApi

import androidx.core.app.NotificationCompat

import androidx.core.app.NotificationManagerCompat

import androidx.core.content.ContextCompat.getSystemService

import androidx.fragment.app.FragmentManager
```

```
import androidx.lifecycle.lifecycleScope

import androidx.recyclerview.widget.LinearLayoutManager

import com.example.primefitness.R

import com.example.primefitness.adapter.AdapterLoadMember

import com.example.primefitness.databinding.FragmentFeePendingBinding

import com.example.primefitness.global.DB

import com.example.primefitness.global.MyFunction

import com.example.primefitness.model.AllMember

import kotlinx.coroutines.CoroutineScope

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.launch

import kotlinx.coroutines.withContext

import java.util.*

import kotlin.collections.ArrayList


class FragmentFeePending : BaseFragment() {


    private lateinit var binding: FragmentFeePendingBinding

    private var db:DB?=null

    var adapter:AdapterLoadMember?=null

    var arrayList:ArrayList<AllMember> = ArrayList()

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {

        // Inflate the layout for this fragment

        binding = FragmentFeePendingBinding.inflate(inflater, container, false)
```

```
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    activity?.title = "Fee Pending"
    db = activity?.let { DB(it) }

    binding.edtPendingSearch.addTextChangedListener(object : TextWatcher{
        override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {

        }

        override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {

        }

        override fun afterTextChanged(s: Editable?) {
            myFilter(s.toString())
        }

    })
```

```
}

override fun onResume() {
    super.onResume()
    loadData()
}

fun <R> CoroutineScope.executeAsyncTask(
    onPreExecute: () -> Unit,
    doInBackground: () -> R,
    onPostExecute: (R) -> Unit
) = launch {
    onPreExecute()
    val result = withContext(Dispatchers.IO){
        doInBackground()
    }
    onPostExecute(result)
}

@RequiresApi(Build.VERSION_CODES.O)
private fun loadData(){
    lifecycleScope.executeAsyncTask(onPreExecute = {
        showDialog("Processing....")
    }, doInBackground = {
```

```
        arrayList.clear()

        val sqlQuery = "SELECT * FROM MEMBER WHERE STATUS='A' AND
EXPIRE_ON<='"+MyFunction.getThreeDaysAfterDate()+"' ORDER BY FIRST_NAME"  //three
days before notification

        db?.fireQuery(sqlQuery)?.use {

            if(it.count>0){

                it.moveToFirst()

                do {

                    val list = AllMember(id = MyFunction.getvalue(it, "ID"),

                        firstName = MyFunction.getvalue(it, "FIRST_NAME"),

                        lastName = MyFunction.getvalue(it, "LAST_NAME"),

                        age = MyFunction.getvalue(it, "AGE"),

                        gender = MyFunction.getvalue(it, "GENDER"),

                        weight = MyFunction.getvalue(it, "WEIGHT"),

                        mobile = MyFunction.getvalue(it, "MOBILE"),

                        address = MyFunction.getvalue(it, "ADDRESS"),

                        image = MyFunction.getvalue(it, "IMAGE_PATH"),

                        dateOfJoining = MyFunction.returnUserdateFormat(MyFunction.getvalue(it,
"DATE_OF_JOINING")),

                        expiryDate = MyFunction.returnUserdateFormat(MyFunction.getvalue(it,
"EXPIRE_ON")),

                        description = MyFunction.getvalue(it, "DESCRIPTION"))



                    arrayList.add(list)

                } while (it.moveToNext())
```

```
            }
        }
    }, onPostExecute = {
        if(arrayList.size>0){
            binding.recyclerViewPending.visibility = View.VISIBLE
            binding.txtPendingNDF.visibility = View.GONE
            adapter = AdapterLoadMember(arrayList)
            binding.recyclerViewPending.layoutManager = LinearLayoutManager(activity)
            binding.recyclerViewPending.adapter = adapter



            adapter?.onClick {
                loadFragment(it)
            }

        } else {
            binding.recyclerViewPending.visibility = View.GONE
            binding.txtPendingNDF.visibility = View.VISIBLE
        }
        CloseDialog()
    })
}

private fun loadFragment(id:String){
```

```
        val fragment = FragmentAddMember()

        val args = Bundle()

        args.putString("ID", id)

        fragment.arguments = args

        val fragmentManager: FragmentManager?= fragmentManager

        fragmentManager!!.beginTransaction().replace(R.id.frame_container, fragment,
"FragmentAdd").commit()

    }



    private fun myFilter(searchValue:String){

        val temp:ArrayList<AllMember> = ArrayList()

        if(arrayList.size>0){

            for(list in arrayList){


if(list.firstName.toLowerCase(Locale.ROOT).contains(searchValue.toLowerCase(Locale.ROOT)
) ||
list.lastName.toLowerCase(Locale.ROOT).contains(searchValue.toLowerCase(Locale.ROOT))){

                    temp.add(list)

                }

            }

        }

        adapter?.updateList(temp)

    }

}
```

Database:

SQL Table

```
package com.example.primefitness.global


object SqlTable {

    const val admin =

        "CREATE TABLE ADMIN(ID INTEGER PRIMARY KEY AUTOINCREMENT, USER_NAME
TEXT DEFAULT '', PASSWORD TEXT DEFAULT '', MOBILE TEXT DEFAULT '')"

    const val member = "CREATE TABLE MEMBER(ID INTEGER PRIMARY KEY
AUTOINCREMENT, FIRST_NAME TEXT DEFAULT '', LAST_NAME TEXT DEFAULT '',
GENDER TEXT DEFAULT '', AGE TEXT DEFAULT '', WEIGHT TEXT DEFAULT '', MOBILE
TEXT DEFAULT '', ADDRESS TEXT DEFAULT '', DATE_OF_JOINING TEXT DEFAULT '',
EXPIRE_ON TEXT DEFAULT '', TOTAL TEXT DEFAULT '', IMAGE_PATH TEXT DEFAULT '',
STATUS TEXT DEFAULT 'A', IMAGE_PATH_ID_1 TEXT DEFAULT '', IMAGE_PATH_ID_2
TEXT DEFAULT '', DESCRIPTION TEXT DEFAULT '')"

    const val fee = "CREATE TABLE FEE(ID INTEGER PRIMARY KEY AUTOINCREMENT,
ONE_MONTH TEXT DEFAULT '', THREE_MONTH TEXT DEFAULT '')"

}
```

DB

```
package com.example.primefitness.global


import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

import android.util.Log

import java.lang.Exception
```

```
class DB (val context:Context) : SQLiteOpenHelper(context, DB_NAME, null, DB_VERSION){



    override fun onCreate(db: SQLiteDatabase) {

//        Log.d("Table", "admin")

//        val db =
SQLiteDatabase.openOrCreateDatabase("/mnt/sdcard/Download/PrimeFitness.DB", null) // not
working

        db.execSQL(SqlTable.admin)

        db.execSQL(SqlTable.member)

        db.execSQL(SqlTable.fee)

    }



    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {



    }



    // to run the query

    fun executeQuery(sql:String):Boolean{

        try {

            val database = this.writableDatabase

            database.execSQL(sql)

        } catch (e:Exception){

            e.printStackTrace()
```

```
        return false
    }
    return true
}
//retrieve data
fun fireQuery(sql:String):Cursor?{
    var temCursor:Cursor?=null
    try {
        val database = this.writableDatabase
        temCursor = database.rawQuery(sql, null)
        if (temCursor!=null && temCursor.count>0) {
            temCursor.moveToFirst()
            return temCursor
        }
    } catch (e:Exception){
        e.printStackTrace()
    }
    return temCursor
}
companion object{
    private const val DB_VERSION = 1
    private const val DB_NAME = "PrimeFitness.DB"
}
```

}