

Group ID: 10

Project Topic: Life Insurance Calculator

Student ID	Student Name	Android Project						Report				Viva(15)	Total (100)
		Wireframes(8)	Project Design and Consistency(8)	Integrated application using a repository(3)	Correct calculations for a working app(8)	using SQLite db(20)	Unit Test cases written in Android Project(10)	Explaining connectivity of interfaces(3)	Explaining UI design principles applied to the project (10)	Additional features(5)	mobile test cases(10)		
IT17001908	M.W.S.B Bandara												
IT16521544	B.D.K Samaraweera												

Application project link : - https://drive.google.com/drive/folders/1fWQSWWSmkwQUbk82sBVyGQuCuI07n_MX?usp=sharing

Wireframe design link : - <https://app.moqups.com//SJlvrmn9Bt/edit/page/ad64222d5#>

Table of Contents

01. APPLICATION INTRODUCTION, ACTUAL PROTOTYPE AND USER INTERFACES3

01.1 LOGO3

01.2 PROTOTYPE4

01.3 SPLASH SCREEN5

01.4 MAIN MENU6

01.5 CALCULATOR OPTION7

01.6 HISTORY OPTION8

02. SNAPSHOTS OF RUNNING APPLICATION9

03. CODE SCREENSHOTS10

03.1 CALCULATION.....10

03.2 DATABASE12

04. ADDITIONAL FEATURE.....14

05. MOBILE TEST CASES WITH RESULTS.....15

06. UNIT TEST CASES WITH RESULTS16

01. Application Introduction, Actual prototype and user interfaces

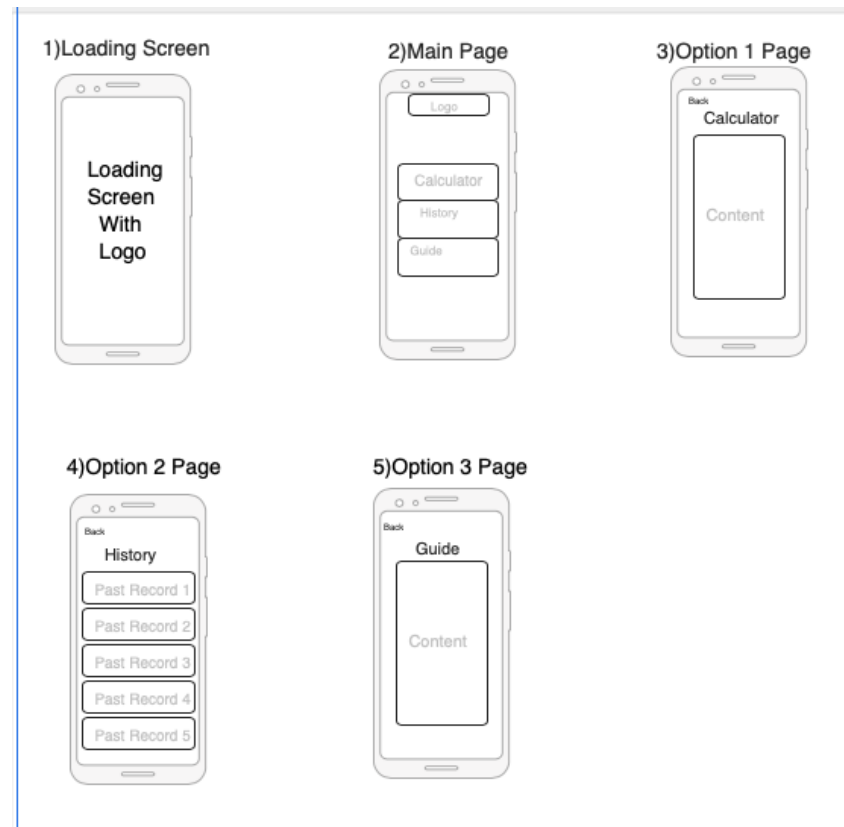
This Application is named as 'MY LIC'. The meaning of LIC is 'Life Insurance Corporation'. We have assigned this name to give a familiar feeling to the user. The specialty of this application is anyone can use without creating an account. So, it will be very convenient for any user who are willing to calculate their life insurance plan.

01.1 Logo



- In this we have decided to use red and orange mixed color combination.
- Because red color is the symbol of energetic and orange color is used to represent confidence, friendliness.
- We have added this logo in png mode. So, it can apply anywhere we want.
- We have added this logo on the middle of the splash screen to emphasize to the user by the first sight.

01.2 Prototype



When the app opens, it is starting with a splash screen (Image - 1). Then it will go to the menu (Image - 2). In the menu, it has 3 options. Calculator, History and Guide options.

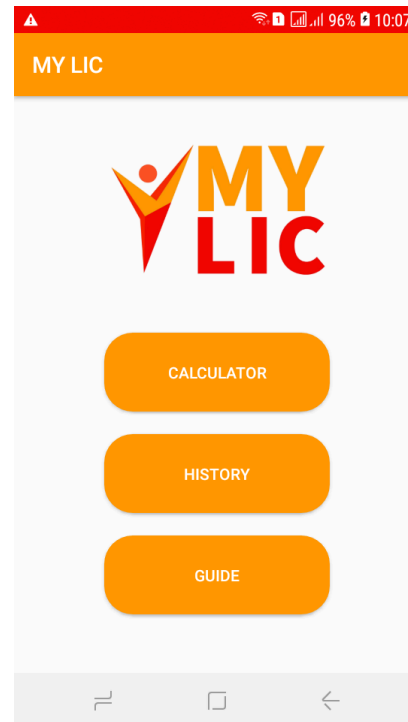
- Calculator – When we input data to the calculator menu, it gives the monthly installment you have to pay when you are applying for a life insurance and, yearly how much you want to pay for the life insurance.
- History – In this option, it shows past records of you have entered. We can retrieve the data.
- Guide – In this option it included small tips on how you should choose a life insurance, what are the types of life insurances likewise.

01.3 Splash Screen



- In this splash screen we have used 3 colors. Red, Orange and Blue. As discussed in the logo section red and orange is used to emphasize energetic and confidence, friendliness.
- This is a life insurance application. So, we have to build a trust between the company and the user. So, Blue color is used to enhance that feeling to the user.
- When a huge incident is happened most of them checks twitter trending. It is because twitter was able to build the trust among people. In the twitter application, they use blue color and white color. It is because of a method of keep engaging the user with the trust.

01.4 Main Menu



- We have decided to avoid the complexity. So, we used a white background and only the option buttons and the logo.
- If we added many colors and buttons it gives the complexity feeling to the user.
- In the app we haven't included create account option at the beginning. Because if it implemented, most of the users will give up using the application at the beginning.

01.5 Calculator option

Monthly Income
Rs.

Monthly Bills Expenses
Rs.

Monthly Rental Fees
Rs.

Monthly Medical Fees
Rs.

Monthly Loan Installments
Rs.

CALCULATE

Monthly Income
200000

Monthly Bills Expenses
25000

Monthly Rental Fees
30000

Monthly Medical Fees
5000

Monthly Loan Installments
40000

Monthly Installment : Rs.25000.0
Best Suitable Plan : Rs.300000.0

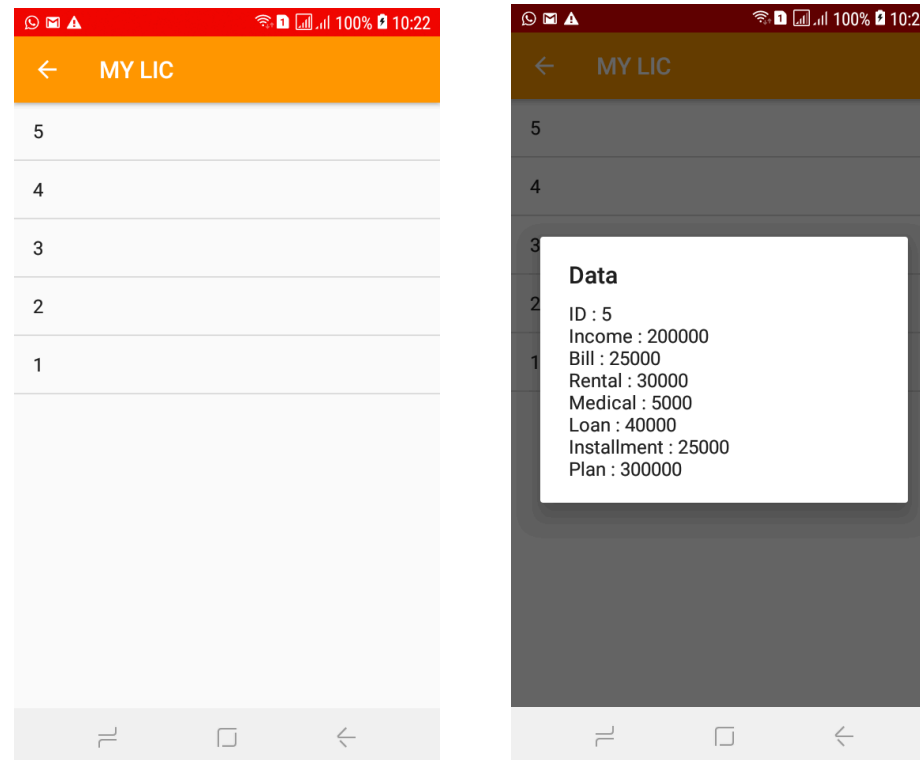
CALCULATE

In calculator option basically it is added 5 fields. After adding data to those fields, it will calculate the best plan to the user. With that user will be able to decide, whether this is suitable for me or not. The specialty in this calculator is it will calculate the monthly installment as well. Calculation is as follows.

- Profit = Monthly Income – (Monthly bill expenses + Rental Fees + Medical Fees + Monthly loan installments)
- Monthly installment = Profit X 25%
- Best suitable plan = Monthly Installment X 12

As a group, we have found out information regarding the calculations how real-world application works. They are using 25% as the standard amount from the profit.

01.6 History Option



- This is the option where past data is stored. There are more than 5 records are storing. But only 5 is showing in the app.
- In the first image it is shown. After tapping any number, it will pop up the result like in image 2.
- In here also we gave the prioritize to avoid the complexity.
- White color in the background gives the calm feeling to the user.

The image displays a sequence of eight mobile application screenshots for "MY LIC".

- Screenshot 1 (Home Screen):** Features a blue header with the "MY LIC" logo. Below the header is an orange navigation bar. The main content area is white and contains three orange buttons: "CALCULATOR", "HISTORY", and "GUIDE".
- Screenshot 2 (Calculator Input):** Shows the "MY LIC" header and navigation bar. The main content area is white and contains four input fields for "Monthly Income" (200000), "Monthly Bills Expenses" (25000), "Monthly Rental Fees" (30000), and "Monthly Medical Fees" (5000). A numeric keypad is visible at the bottom.
- Screenshot 3 (Calculator Output):** Shows the "MY LIC" header and navigation bar. The main content area is white and contains the same input fields as Screenshot 2, but with a "Next" button visible on the right side of the numeric keypad.
- Screenshot 4 (History Screen):** Shows the "MY LIC" header and navigation bar. The main content area is white and contains a list of calculations, including "Monthly Income", "Monthly Bills Expenses", "Monthly Rental Fees", and "Monthly Medical Fees".
- Screenshot 5 (Data Modal):** Shows the "MY LIC" header and navigation bar. The main content area is white and contains a modal window titled "Data" with the following information: ID : 5, Income : 200000, Bill : 25000, Rental : 30000, Medical : 5000, Loan : 40000, Installment : 25000, Plan : 300000.
- Screenshot 6 (What is a life insurance?):** Shows the "MY LIC" header and navigation bar. The main content area is white and contains the text "What is a life insurance?" followed by a paragraph explaining life insurance.
- Screenshot 7 (Types of life insurances):** Shows the "MY LIC" header and navigation bar. The main content area is white and contains the text "Types of life insurances" followed by a list of eight types of life insurance.
- Screenshot 8 (How to select the best insurance?):** Shows the "MY LIC" header and navigation bar. The main content area is white and contains the text "How to select the best insurance?" followed by a list of five steps for selecting the best insurance.

03. Code Screenshots

03.1 Calculation

The screenshot displays the Android Studio interface with the `CalculatorActivity.java` file open. The code implements a calculator activity that calculates monthly installments and the best plan based on user input.

```
public class CalculatorActivity extends AppCompatActivity {

    DatabaseHelper myDb;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calculator2);

        myDb = new DatabaseHelper(context: this);
    }

    public void calculate(View view){

        EditText txtIncome = findViewById(R.id.txtIncome);
        EditText txtBill = findViewById(R.id.txtBills);
        EditText txtRental = findViewById(R.id.txtRental);
        EditText txtMedical = findViewById(R.id.txtMedical);
        EditText txtLoan = findViewById(R.id.txtLoan);
        TextView txtMonthlyInstallment = findViewById(R.id.txtMonthlyInstallment);
        TextView txtBestPlan = findViewById(R.id.txtBestPlan);

        try {
            double income = Double.parseDouble(txtIncome.getText().toString());
            double bills = Double.parseDouble(txtBill.getText().toString());
            double rental = Double.parseDouble(txtRental.getText().toString());
            double medical = Double.parseDouble(txtMedical.getText().toString());
            double loan = Double.parseDouble(txtLoan.getText().toString());

            /*Calculation
            * Profit =Income - (Total Bills + Rent + Medical + Loan)
            * Monthly installment = Profit/4
            * Best plan = Monthly installment * 12
            */

            double monthlyInstallments = calcMonthlyinstallment(income, bills, rental, medical, loan);
            double bestPlan = calcBestPlan(monthlyInstallments);
        }
    }
}
```

The IDE shows a successful build with the following output:

```
Build: Build Output x Sync x
Build: completed successfully at 8/27/20 12:57 PM
Starting Gradle Daemon
Run build /home/chaithura/Downloads/Insurance_Plan
```

The Event Log shows the following messages:

```
12:56 PM NDK Resolution Outcome: Project
12:57 PM Gradle build finished in 48 s 821 ms
```

A notification for IDE and Plugin Updates is visible, stating: "Android Studio is ready to update."

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

com > amarokasia > insurance_plan > CalculatorActivity ExampleUnitTest My Virtual Device (Missing system image)

Android CalculatorActivity.java CalculatorActivityTest.java DatabaseHelper.java ExampleUnitTest.java Insurance_Plan app

Project

- app
 - manifests
 - java
 - com
 - amarokasia
 - insurance_plan
 - CalculatorActivity
 - DatabaseHelper
 - GuideActivity
 - HistoryActivity
 - MainActivity
 - SplashActivity
- com (androidTest)
 - com (test)
 - amarokasia
 - insurance_plan
 - CalculatorActivityTest
 - ExampleUnitTest
- java (generated)
- res
- res (generated)
- Gradle Scripts
 - build.gradle (Project: Insurance_Plan)
 - build.gradle (Module: app)
 - gradle-wrapper.properties (Gradle)
 - proguard-rules.pro (ProGuard Rules)
 - gradle.properties (Project Properties)
 - settings.gradle (Project Settings)
 - local.properties (SDK Location)

Resource Manager

Layout Captures

Structure

7 Favorites

Build Variants

Build: Build Output Sync

Build: completed successfully at 8/27/20 12:57 PM

Starting Gradle Daemon

Run build /home/chatbura/Downloads/Insurance_Plan

Logcat TODO Terminal 9: Version Control Build

Typo: In word 'Monthlyinstallment'

```

47
48
49
50 double monthlyInstallments = calcMonthlyinstallment(income, bills, rental, medical, loan);
51 double bestPlan = calcBestPlan(monthlyInstallments);
52
53 txtMonthlyInstallment.setText("Monthly Installment : Rs."+Double.toString(monthlyInstallments));
54 txtBestPlan.setText("Best Suitable Plan : Rs."+Double.toString(bestPlan));
55
56 saveData(income,bills,rental,medical,loan, monthlyInstallments, bestPlan);
57
58 // Toast.makeText(this, "Clicked"+(Double.toString(income+bills)), Toast.LENGTH_SHORT).show();
59 }catch (Exception e){
60 e.printStackTrace();
61 // Toast.makeText(this, "Clicked"+e.toString(), Toast.LENGTH_SHORT).show();
62 }
63
64
65 public double calcMonthlyinstallment(double income, double bills, double rental, double medical, double loan){
66 double monthlyInstallment = 0.0;
67 double profit = income - (bills+rental+medical+loan);
68 monthlyInstallment = profit/4;
69
70 return monthlyInstallment;
71 }
72
73
74 public double calcBestPlan(double monthlyInstallment){
75 double bestPlan = monthlyInstallment*12;
76 return bestPlan;
77 }
78
79 public void saveData(double income, double bills, double rental, double medical, double loan, double installments, double plan){
80 boolean isInserted = myDb.insertData(income, bills, rental, medical, loan, installments, plan);
81 if (isInserted){
82 Toast.makeText( context: this, text: "Inserted To Database", Toast.LENGTH_SHORT).show();
83 }else{
84 Toast.makeText( context: this, text: "Failed To Database", Toast.LENGTH_SHORT).show();
85 }
86 }
87

```

CalculatorActivity > calcMonthlyinstallment()

Event Log

12:56 PM NDK Resolution Outcome: Project

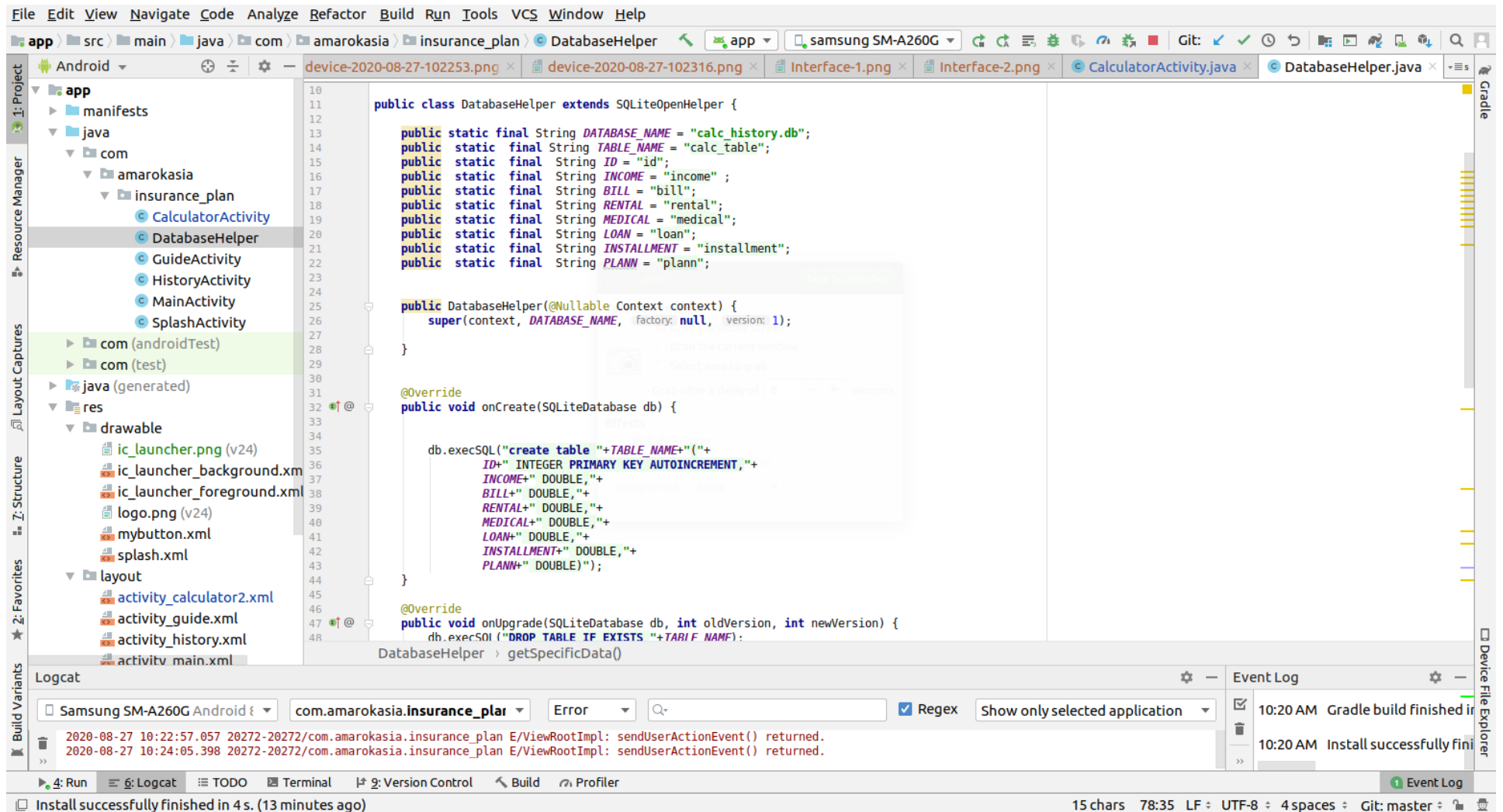
12:57 PM Gradle build finished in 48 s 821 ms

IDE and Plugin Updates

Android Studio is ready to update.

65:27 LF UTF-8 4 spaces Git: master

03.2 Database



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

app > src > main > java > com > amarokasia > insurance_plan > DatabaseHelper

device-2020-08-27-102253.png x device-2020-08-27-102316.png x Interface-1.png x Interface-2.png x CalculatorActivity.java x DatabaseHelper.java x

Project: app

- manifests
- java
 - com
 - amarokasia
 - insurance_plan
 - CalculatorActivity
 - DatabaseHelper
 - GuideActivity
 - HistoryActivity
 - MainActivity
 - SplashActivity
- com (androidTest)
- com (test)
- java (generated)
- res
 - drawable
 - ic_launcher.png (v24)
 - ic_launcher_background.xml
 - ic_launcher_foreground.xml
 - logo.png (v24)
 - mybutton.xml
 - splash.xml
 - layout
 - activity_calculator2.xml
 - activity_guide.xml
 - activity_history.xml
 - activity_main.xml

```

45
46
47 @Override
48 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
49     db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
50     onCreate(db);
51 }
52
53 public boolean insertData(double income, double bill, double rental, double medical, double loan, double installment, double plan){
54     SQLiteDatabase db = this.getWritableDatabase();
55     ContentValues contentValues = new ContentValues();
56     contentValues.put(INCOME,income);
57     contentValues.put(BILL,bill);
58     contentValues.put(RENTAL,rental);
59     contentValues.put(MEDICAL,medical);
60     contentValues.put(LOAN,loan);
61     contentValues.put(INSTALLMENT,installment);
62     contentValues.put(PLANN,plan);
63
64     long result = db.insert(TABLE_NAME, nullColumnHack: null,contentValues);
65
66     if (result==-1){
67         return false;
68     }else{
69         return true;
70     }
71 }
72
73 public Cursor getAllData(){
74     SQLiteDatabase db = this.getWritableDatabase();
75     Cursor result = db.rawQuery( sql: "select * from "+TABLE_NAME+" order by id desc limit 5 ", selectionArgs: null);
76     return result;
77 }
78
79 public Cursor getSpecificData(String id){
80     SQLiteDatabase db = this.getWritableDatabase();
81     Cursor result = db.rawQuery( sql: "select * from "+TABLE_NAME+" where "+ID+"="+id+" ", selectionArgs: null);
82     return result;
83 }

```

DatabaseHelper > getSpecificData()

Logcat

Samsung SM-A260G Android 10 | com.amarokasia.insurance_plai | Error

2020-08-27 10:22:57.057 20272-20272/com.amarokasia.insurance_plan E/ViewRootImpl: sendUserActionEvent() returned.

2020-08-27 10:24:05.398 20272-20272/com.amarokasia.insurance_plan E/ViewRootImpl: sendUserActionEvent() returned.

Event Log

10:20 AM Gradle build finished in 4 s. (13 minutes ago)

10:20 AM Install successfully finished in 4 s. (13 minutes ago)

4: Run | Logcat | TODO | Terminal | Version Control | Build | Profiler

15 chars 78:35 LF UTF-8 4 spaces Git: master

04. Additional Feature



- Extra function is this Guide function.
- This will give some information to the user. If a user has no experience regarding the Life insurances, this will be very helpful.
- To emphasize the content, we decided to use black color font.

05. Mobile test cases with results

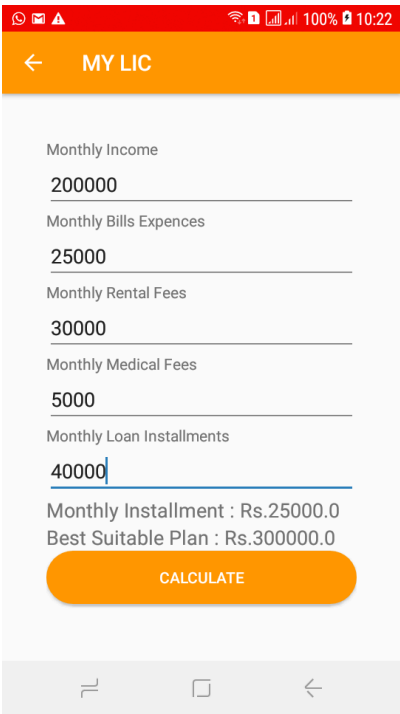


Image 1 – Adding Data

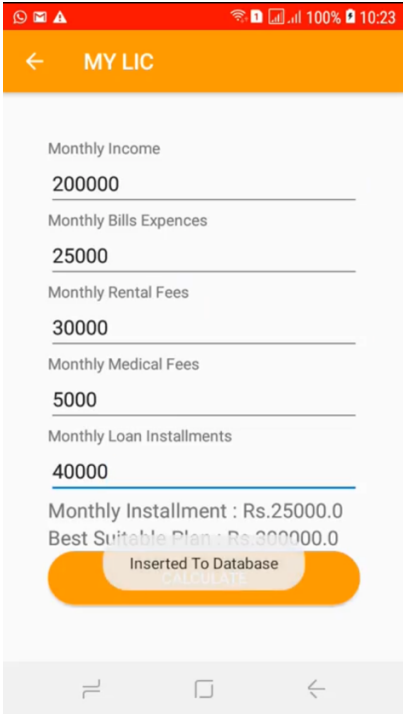


Image 2 – Store calculation

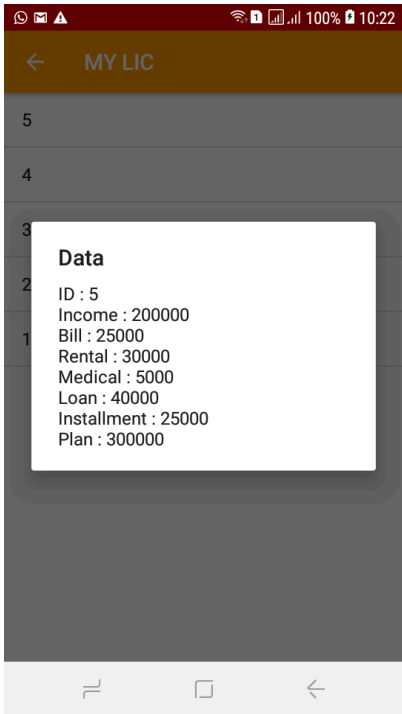


Image 3 – Result

Following data sets are currently stored in the database.

Test Cases	
1.	Monthly Income:- 250,000/= Monthly bills expense:- 30,000/= Monthly rental fees :- 50,000/= Monthly Medical fees :- 10,000/= Monthly loan installments :- 10,000/=
	Monthly installment is :- 37,500/= Best suitable Plan is :- 450,000/= (for one year)
2.	Monthly Income:- 300,000/= Monthly bills expense:- 40,000/= Monthly rental fees :- 70,000/= Monthly Medical fees :- 12000/= Monthly loan installments :- 20,000/=
	Monthly installment is :- 39,500/= Best suitable Plan is :- 474,000/= (for one year)
3.	Monthly Income:- 500,000/= Monthly bills expense:- 50,000/= Monthly rental fees :- 90,000/= Monthly Medical fees :- 8000/= Monthly loan installments :- 30,000/=
	Monthly installment is :- 80,500/= Best suitable Plan is :- 966,000/= (for one year)
4.	Monthly Income:- 1,000,000/= Monthly bills expense:- 75,000/= Monthly rental fees :- 100,000/= Monthly Medical fees :- 5000/= Monthly loan installments :- 40,000/=
	Monthly installment is :- 195,000/= Best suitable Plan is :- 2,340,000/= (for one year)
5th test case already added	

06. Unit test cases with results

The screenshot displays the Android Studio interface with the `CalculatorActivityTest` class open. The code includes two test methods: `calcMonthlyinstallment` and `calcBestPlan`. The `calcMonthlyinstallment` method contains five test cases, each with specific input values for income, bills, rental, medical, and loan, and an expected output. The `calcBestPlan` method contains two test cases that use the `calcMonthlyinstallment` method's output as input for `calcBestPlan`.

The bottom panel shows the **Run** tab with the following output:

```
Run: MainActivity x CalculatorActivityTest x
Tests passed: 2 of 2 tests - 107 ms
Calculation: 107 ms
calcBestPlan: 107 ms
calcMonthlyinstallment: 0 ms
Process finished with exit code 0
```

The **Event Log** shows the following messages:

- 6:49 PM Executing tasks: [:app:generateDebugSources, :app:compileDebugSources, :app:createMockableJar]
- 6:49 PM Gradle build finished in 4 s 242 ms
- 6:49 PM Tests passed: 2

A notification at the bottom right states: **IDE and Plugin Updates** - Android Studio is ready to update.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

com \ amarokasia \ insurance_plan \ CalculatorActi CalculatorActivityTest My Virtual Device (Missing system image)

Android app MainActivity.java AppCompatActivity.java CalculatorActivity.java CalculatorActivityTest.java DatabaseHelper.java

Project Structure: app, manifests, java, com, amarokasia, insurance_plan, CalculatorActivity, DatabaseHelper, GuideActivity, HistoryActivity, MainActivity, SplashActivity, com (androidTest), com (test), amarokasia, insurance_plan, CalculatorActivityTest, ExampleUnitTest

```

26 output = calc.calcMonthlyinstallment( income: 1000000, bills: 75000, rental: 100000, medical: 5000, loan: 40000);
27 assertEquals( expected: 195000,output, delta: 0.1);
28 }
29
30
31 @Test
32 public void calcBestPlan() {
33     CalculatorActivity calc = new CalculatorActivity();
34     double monthlyinstallment;
35     double output;
36
37     monthlyinstallment = calc.calcMonthlyinstallment( income: 200000, bills: 25000, rental: 30000, medical: 5000, loan: 40000);
38     output = calc.calcBestPlan(monthlyinstallment);
39     assertEquals( expected: 300000,output, delta: 0.1);
40
41     monthlyinstallment = calc.calcMonthlyinstallment( income: 250000, bills: 30000, rental: 50000, medical: 10000, loan: 10000);
42     output = calc.calcBestPlan(monthlyinstallment);
43     assertEquals( expected: 450000,output, delta: 0.1);
44
45     monthlyinstallment = calc.calcMonthlyinstallment( income: 300000, bills: 40000, rental: 70000, medical: 12000, loan: 20000);
46     output = calc.calcBestPlan(monthlyinstallment);
47     assertEquals( expected: 474000,output, delta: 0.1);
48
49     monthlyinstallment = calc.calcMonthlyinstallment( income: 500000, bills: 50000, rental: 90000, medical: 8000, loan: 30000);
50     output = calc.calcBestPlan(monthlyinstallment);
51     assertEquals( expected: 966000,output, delta: 0.1);
52
53     monthlyinstallment = calc.calcMonthlyinstallment( income: 1000000, bills: 75000, rental: 100000, medical: 5000, loan: 40000);
54     output = calc.calcBestPlan(monthlyinstallment);
55     assertEquals( expected: 2340000,output, delta: 0.1);
56 }
57

```

Run: MainActivity CalculatorActivityTest

Tests passed: 2 of 2 tests - 107 ms

Calculation results:

Test Case	Income	Bills	Rental	Medical	Loan	Expected Output
1	1000000	75000	100000	5000	40000	195000
2	200000	25000	30000	5000	40000	300000
3	250000	30000	50000	10000	10000	450000
4	300000	40000	70000	12000	20000	474000
5	500000	50000	90000	8000	30000	966000
6	1000000	75000	100000	5000	40000	2340000

Event Log: 6:49 PM Executing tasks: [:app:generateDebugSources, :app:compileDebugSources, :app:createMockable...

6:49 PM Gradle build finished in 4 s 242 ms

6:49 PM Tests passed: 2

IDE and Plugin Updates: Android Studio is ready to update.

Tests passed: 2 (a minute ago)

35:23 LF UTF-8 4 spaces Git: master

-End of the document-