

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//SJlvrnm9Bt/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 CALCULATION10
 - 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

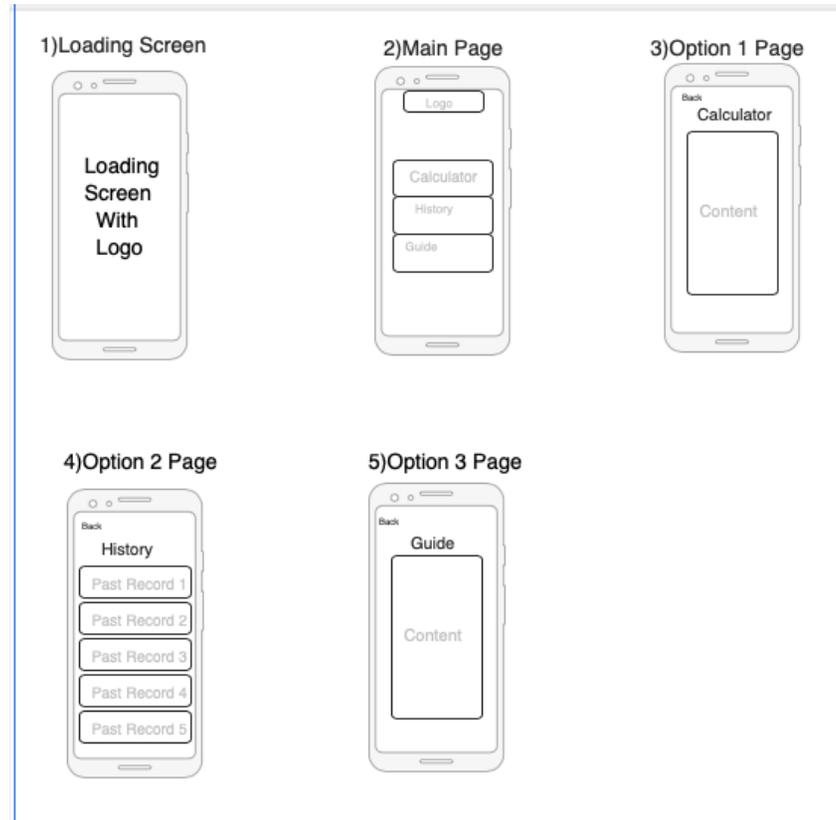
This Application is named as 'MY LIC'. The meaning of LIC is 'Life Insurance Corporation'. We have assigned this name to give a familiar feeling to the user. The specialty of this application is anyone can use without creating an account. So, it will be very convenient for any user who are willing to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

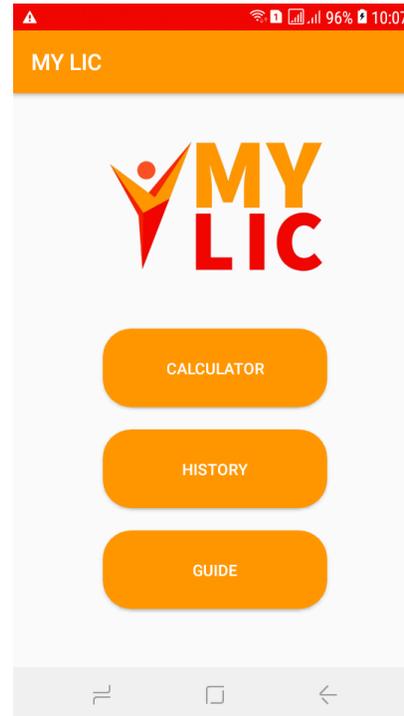
- Calculator – When we input data to the calculator menu, it gives the monthly installment you have to pay when you are applying for a life insurance and, yearly how much you want to pay for the life insurance.
- History – In this option, it shows past records of you have entered. We can retrieve the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

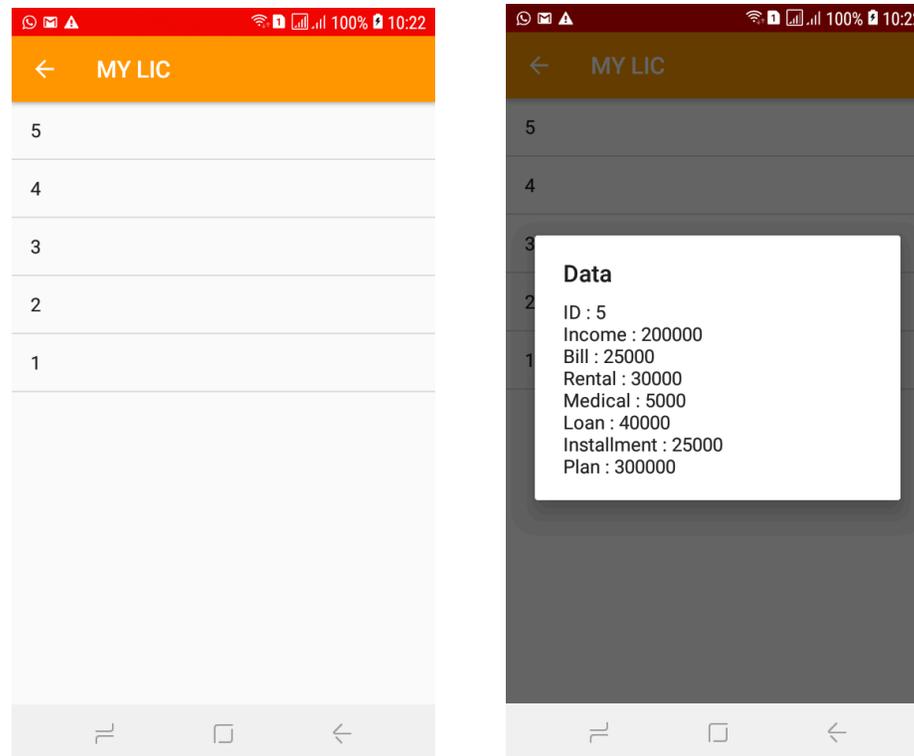
The image displays two screenshots of a mobile application interface for a calculator. The left screenshot shows the input fields for Monthly Income, Monthly Bills Expenses, Monthly Rental Fees, Monthly Medical Fees, and Monthly Loan Installments, all with 'Rs.' as a placeholder. The right screenshot shows the same fields filled with numerical values: 200000, 25000, 30000, 5000, and 40000. Below the fields, the calculated results are displayed: 'Monthly Installment : Rs.25000.0' and 'Best Suitable Plan : Rs.300000.0'. Both screens have an orange 'CALCULATE' button at the bottom.

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

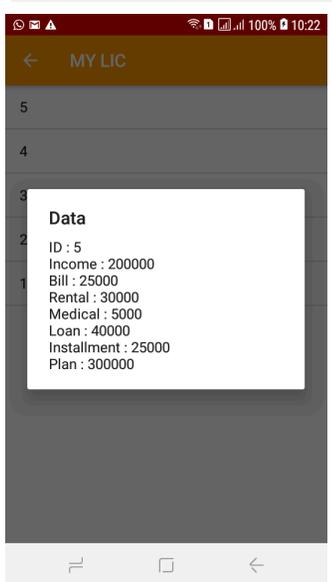
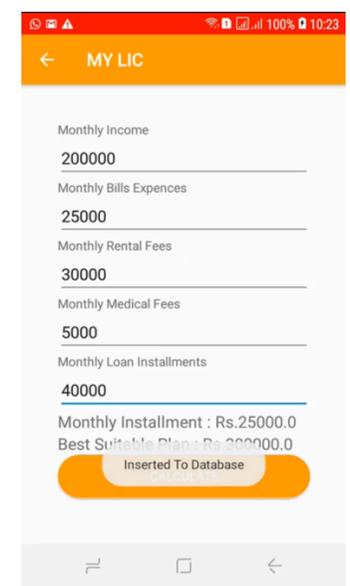
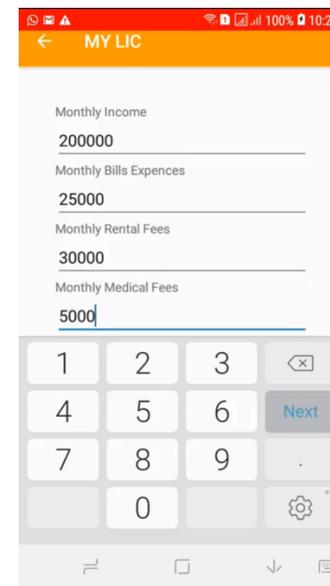
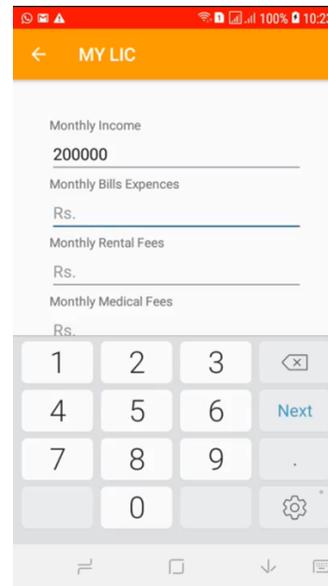
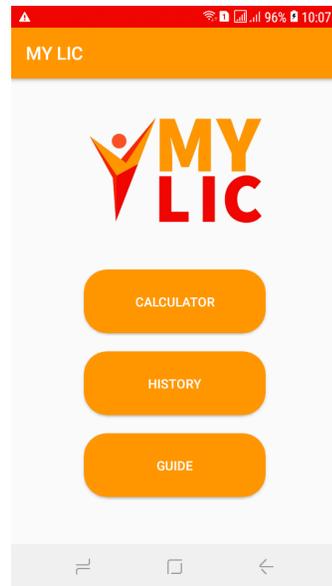
As a group, we have found out information regarding the calculations how real-world application works. They are using 25% as the standard amount from the profit.

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.

02. Snapshots of running application



03. Code Screenshots

03.1 Calculation

The screenshot displays the source code for `CalculatorActivity.java` in an IDE. The code is as follows:

```
10 public class CalculatorActivity extends AppCompatActivity {
11
12     DatabaseHelper myDb;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_calculator2);
18
19         myDb = new DatabaseHelper(context: this);
20     }
21
22     public void calculate(View view){
23
24         EditText txtIncome = findViewById(R.id.txtIncome);
25         EditText txtBill = findViewById(R.id.txtBills);
26         EditText txtRental = findViewById(R.id.txtRental);
27         EditText txtMedical = findViewById(R.id.txtMedical);
28         EditText txtLoan = findViewById(R.id.txtLoan);
29         TextView txtMonthlyInstallment = findViewById(R.id.txtMonthlyInstallment);
30         TextView txtBestPlan = findViewById(R.id.txtBestPlan);
31
32         try {
33             double income = Double.parseDouble(txtIncome.getText().toString());
34             double bills = Double.parseDouble(txtBill.getText().toString());
35             double rental = Double.parseDouble(txtRental.getText().toString());
36             double medical = Double.parseDouble(txtMedical.getText().toString());
37             double loan = Double.parseDouble(txtLoan.getText().toString());
38
39             /*Calculation
40             * Profit =Income - (Total Bills + Rent + Medical + Loan)
41             * Monthly installment = Profit/4
42             * Best plan = Monthly installment * 12
43             */
44
45             double monthlyInstallments = calcMonthlyinstallment(income, bills, rental, medical, loan);
46             double bestPlan = calcBestPlan(monthlyInstallments);
47         }
48     }
49
50 }
```

The IDE interface includes a Project Explorer on the left showing the project structure, a toolbar at the top, and a Build/Event Log panel at the bottom. The build log shows a successful build at 12:57 PM. An IDE update notification is also visible in the bottom right corner.

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

com > amarokasia > insurance_plan > CalculatorActivity

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

Project: 1: Insurance_Plan

- 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)

```

47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
    double monthlyInstallments = calcMonthlyinstallment(income, bills, rental, medical, loan);
    double bestPlan = calcBestPlan(monthlyInstallments);

    txtMonthlyInstallment.setText("Monthly Installment : Rs."+Double.toString(monthlyInstallments));
    txtBestPlan.setText("Best Suitable Plan : Rs."+Double.toString(bestPlan));

    saveData(income,bills,rental,medical,loan, monthlyInstallments, bestPlan);

    Toast.makeText(this, "Clicked"+(Double.toString(income+bills)), Toast.LENGTH_SHORT).show();
} catch (Exception e){
    e.printStackTrace();
    Toast.makeText(this, "Clicked"+e.toString(), Toast.LENGTH_SHORT).show();
}
}

public double calcMonthlyinstallment(double income, double bills, double rental, double medical, double loan){
    double monthlyInstallment = 0.0;
    double profit = income - (bills+rental+medical+loan);
    monthlyInstallment = profit/4;

    return monthlyInstallment;
}

public double calcBestPlan(double monthlyInstallment){
    double bestPlan = monthlyInstallment*12;
    return bestPlan;
}

public void saveData(double income, double bills, double rental, double medical, double loan, double installments, double plan){
    boolean isInserted = myDb.insertData(income, bills, rental, medical, loan, installments, plan);
    if (isInserted){
        Toast.makeText( context: this, text: "Inserted To Database", Toast.LENGTH_SHORT).show();
    }else{
        Toast.makeText( context: this, text: "Failed To Database", Toast.LENGTH_SHORT).show();
    }
}
}

```

Build: Build Output | Sync | Event Log

- Build: completed successfully at 8/27/20 12:57 PM (48 s 817 ms)
- Starting Gradle Daemon (3 s 58 ms)
- Run build /home/chatbura/Downloads/Insurance_Plan (20 s 402 ms)
- 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.

Type: In word 'Monthlyinstallment'

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

03.2 Database

The screenshot displays the Android Studio IDE with the following components:

- Project Explorer (Left):** Shows the project structure for 'app', including 'manifests', 'java', 'res', and 'layout' folders. The 'DatabaseHelper' class is highlighted in the 'java' folder.
- Code Editor (Center):** Displays the source code for 'DatabaseHelper.java'. The code defines a class that extends 'SQLiteOpenHelper' and includes static constants for database and table names, as well as methods for 'onCreate' and 'onUpgrade'.
- Logcat (Bottom):** Shows the output of the application, including the message 'Install successfully finished in 4 s. (13 minutes ago)'.

```
10
11
12
13 public class DatabaseHelper extends SQLiteOpenHelper {
14
15     public static final String DATABASE_NAME = "calc_history.db";
16     public static final String TABLE_NAME = "calc_table";
17     public static final String ID = "id";
18     public static final String INCOME = "income";
19     public static final String BILL = "bill";
20     public static final String RENTAL = "rental";
21     public static final String MEDICAL = "medical";
22     public static final String LOAN = "loan";
23     public static final String INSTALLMENT = "installment";
24     public static final String PLANN = "plann";
25
26     public DatabaseHelper(@Nullable Context context) {
27         super(context, DATABASE_NAME, Factory: null, version: 1);
28     }
29
30     @Override
31     public void onCreate(SQLiteDatabase db) {
32
33         db.execSQL("create table "+TABLE_NAME+"("
34             +ID+" INTEGER PRIMARY KEY AUTOINCREMENT,"
35             +INCOME+" DOUBLE,"
36             +BILL+" DOUBLE,"
37             +RENTAL+" DOUBLE,"
38             +MEDICAL+" DOUBLE,"
39             +LOAN+" DOUBLE,"
40             +INSTALLMENT+" DOUBLE,"
41             +PLANN+" DOUBLE)");
42     }
43
44     @Override
45     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
46         db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
47     }
48
49     DatabaseHelper > getSpecificData()
```

Logcat Output:

```
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.
```

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

app src main java com amarokasia insurance_plan DatabaseHelper app samsung SM-A260G

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

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 public Cursor getAllData(){
73     SQLiteDatabase db = this.getWritableDatabase();
74     Cursor result = db.rawQuery( sql: "select * from "+TABLE_NAME+" order by id desc limit 5 ", selectionArgs: null);
75     return result;
76 }
77
78 public Cursor getSpecificData(String id){
79     SQLiteDatabase db = this.getWritableDatabase();
80     Cursor result = db.rawQuery( sql: "select * from "+TABLE_NAME+" where "+ID+"="+id+" ", selectionArgs: null);
81     return result;
82 }
83 }

```

DatabaseHelper > getSpecificData()

Logcat

Samsung SM-A260G Android 8.0.0 com.amarokasia.insurance_plar 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 4s (13 minutes ago)

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

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

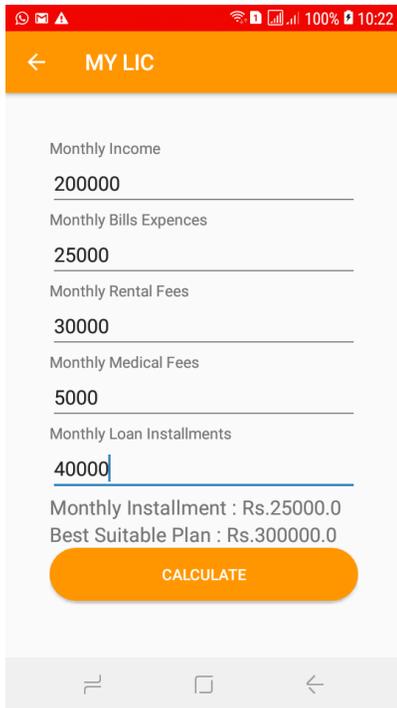


Image 1 – Adding Data

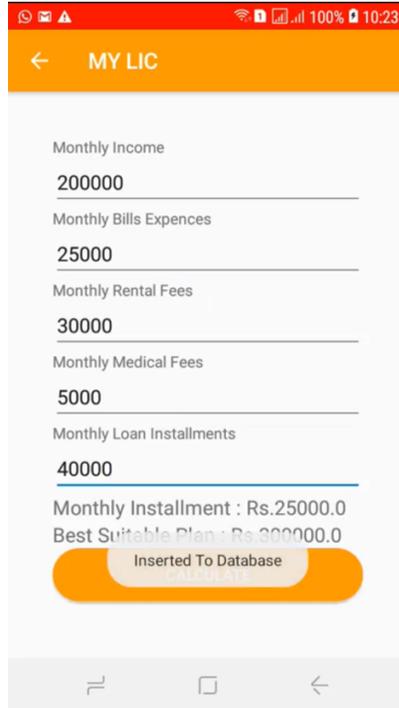


Image 2 – Store calculation

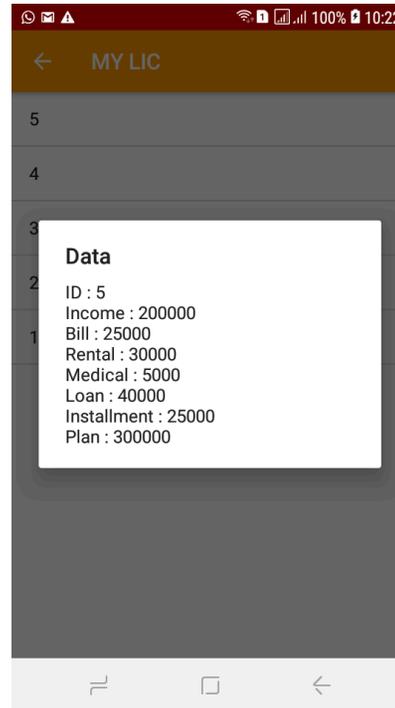


Image 3 – Result

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 an IDE interface with the following components:

- Project Explorer:** Shows the project structure with folders for manifests, java, and resources. The test class `CalculatorActivityTest` is highlighted under the `com (test)` package.
- Code Editor:** Displays the source code for `CalculatorActivityTest.java`. It contains two test methods:
 - `calcMonthlyinstallment()`: A test method that calls `calcMonthlyinstallment()` with various parameters and asserts the output. The test cases include:
 - Income: 200000, Bills: 25000, Rental: 30000, Medical: 5000, Loan: 40000. Expected output: 25000.
 - Income: 250000, Bills: 30000, Rental: 50000, Medical: 10000, Loan: 10000. Expected output: 37500.
 - Income: 300000, Bills: 40000, Rental: 70000, Medical: 12000, Loan: 20000. Expected output: 39500.
 - Income: 500000, Bills: 50000, Rental: 90000, Medical: 8000, Loan: 30000. Expected output: 80500.
 - Income: 1000000, Bills: 75000, Rental: 100000, Medical: 5000, Loan: 40000. Expected output: 195000.
 - `calcBestPlan()`: A test method that calls `calcMonthlyinstallment()` and `calcBestPlan()` with the same parameters and asserts the output. The test cases include:
 - Income: 200000, Bills: 25000, Rental: 30000, Medical: 5000, Loan: 40000. Expected output: 300000.
 - Income: 250000, Bills: 30000, Rental: 50000, Medical: 10000, Loan: 10000. Expected output: 450000.
- Run Panel:** Shows the execution results for the tests. It indicates that 2 of 2 tests passed in 107 ms. The output shows the test process finished with exit code 0.
- Event Log:** Shows the execution tasks: `[:app:generateDebugSources, :app:compileDebugSources, :app:createMockable`.
- Notification:** A notification at the bottom right states "IDE and Plugin Updates" and "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) Git

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

Project: 1: 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_P
 build.gradle (Module: app)
 gradle-wrapper.properties (Gradl
 proguard-rules.pro (ProGuard Rul
 gradle.properties (Project Proper

```

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

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-