Dharmsinh Desai University , Nadiad
Faculty of Technology
Department of Computer Engineering

B. Tech. CE Semester – VI

Subject: System Design Practice

Project Title: Simple to use app for managing earnings and expenses

By: Tejas Dobariya , CE035, 20CEUOS048
Maharshi Mistry, CE004, 20CEUBS043

Guided by:
Prof. Brijesh Bhatt
Professor,
CE Dept, DDIT

Dharmsinh Desai University

College Road, NADIAD - 387001. (Gujarat)

Certificate

This is to certify that the practical / term work carried out in the subject of System Design Practice and recorded in this journal is the bonafide work of

Tejas Dobariya , CE035, 20CEUOS048

Maharshi Mistry, CE004, 20CEUBS043

of B.Tech semester VI in the branch of Computer Engineering during the academic year 2022-2023.

Table of Content

1. Abstract

The Purpose of this project is to develop a solution to everyday problem which is management of budget. To solve this problem, we need to keep in mind that whatever solution is developed it needs to be trackable, user friendly and easily accessible. With the data of solution to be secure and accessible to the rightful data owner.

2. Introduction

General Brief

The project is about to develop an application as a solution to the budget management problem. This application needs to contain all attributes of a budget management solution. These attributes are mainly trackability, security and data depiction.
The traditional solution lacks the trackablity as it cannot manage to list all the expenses of the same type in one page and which may create a problem of loss of data which need to be taken into account. An untimely data in a traditional solution may create a problem of tracking of data while analyzing.

Security is one of the most important attributes of the solution. As it gives a sense of relief to the data owner that there wouldn't be any stealing of information , accessed by other than the rightful owner of the data. The traditional solution can solve this problem as it's all physical. Any unknown person can see the data of the owner. Can use that data wrongfully. This can be solve by placing it on modern devices which can give security to application

We come to one of the most important attributes; data depiction.
The Traditional solution lacks the feature to produce as it has to be done manually which might be a tedious job to do. Data depiction

use the data and create a graphical representation of data which help user to make better decision about the expenses and budgeting

Technology used to build the application are as follows:-

Android Studio
Github - To manage application version
Firebase- As database of application

3. Software Requirement specification

1. Manage Account

R.1.1 :- Register user

- Description: If a new user wants to Track of his expense, user needs to register himself/herself first.

R.1.1.1 :- Sign up

- input : Required Details like Username , email id , contact , number , password etc ….
- output : Account created

R.1.1.2 :- Sign in

-input
Username , password
-output
Logged in

R.1.1.3 :- display user information

-output : Display user's profile

2. Manage Earnings

R.2.1 :- Add INCOME

- Description:
  User will be able to add income
  - input :
  Insert income value
  - output :
  Value will be added.

R.2.2 :- Manage category

R.2.2.1 :- Add Category

-Description: User can add any category in which he wants to spend his income

-input: category name

-output: category created

R.2.2.2 : Select Existing Category

Description :- User can select one or more than one categories from the given list of categories.

Input :-Select Categories

Output:- Categories will be added to home page

R.2.3 Add Expense to categories

Description :- Add the maximum limit of expense of each categories selected by User .

Input:- set maximum limit of categories

Output :- show limit for categories on home page

3. Manage Expense

R.3.1 Add Expense

R.3.1.1 expense details

Description :- user can add his expense amount for the day in the categories of his choice.

Input :- expense amount , category and date

Output: Amount added

R.3.1.2 Bank Details

Description: user can add the bank details , hence the expense will be directly added to categories according to the payment made.

Input: Add bank details

Output: details added

R.3.2 Alert Message

Description: On adding a limit to expenses after a certain amount left for that limit , an alert message will be shown

Input: add expense details

Output: show alert message after user is closer to limit

R.3.3 :- View Expenses

Description: User can view their expense category-wise on daily as well as monthly basis.

R.3.3.1 :- Recent expenses

Description:- user can see all the recent expenses.

R.3.3.2 :- Chart representation

- Description: view the expenses made on particular categories on daily and monthly basis.

Output:- show chart

R.3.4 :- Delete Expense

- Description:
  User can delete particular transaction.

4 :- View Balance

- Description:
  View the total balance left after total expenses.

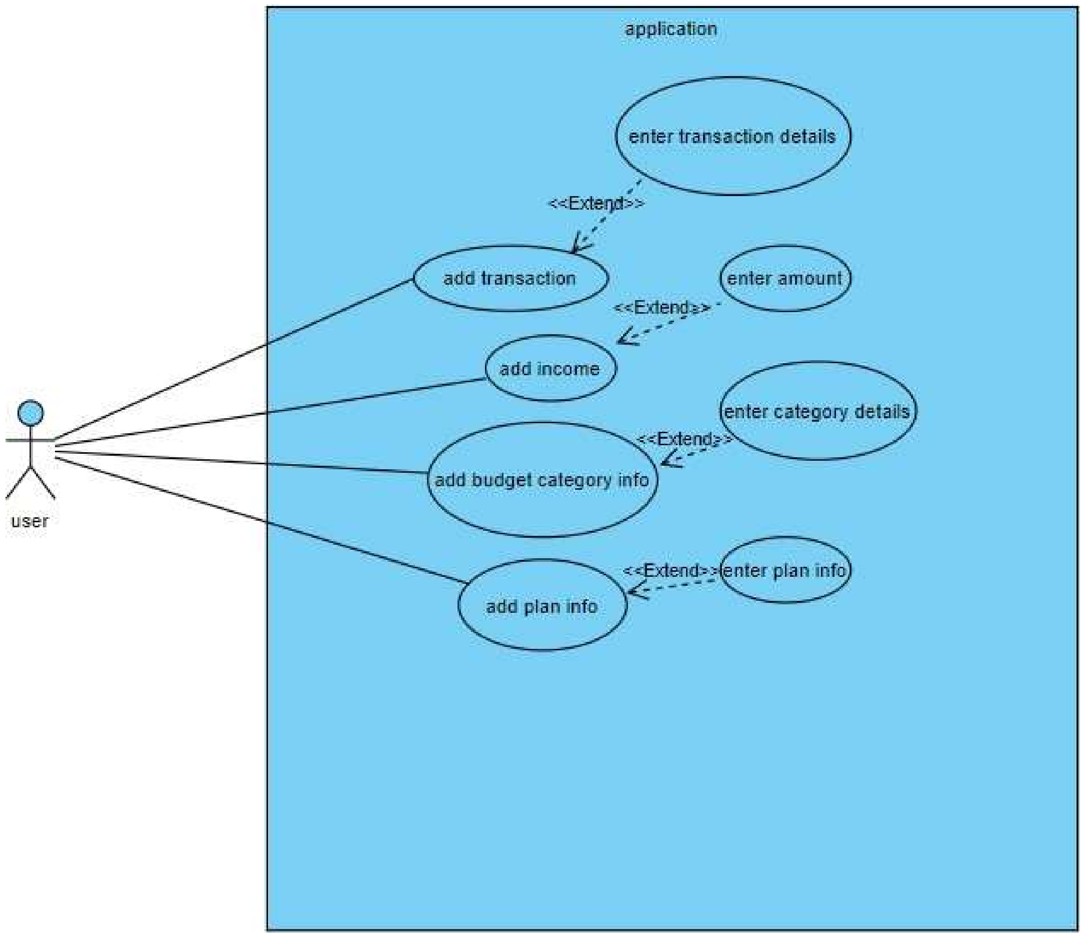
5:- Report
-Description:- Generates a report to give a detailed insight about budgets, income, balance.

Non-functional Requirements:

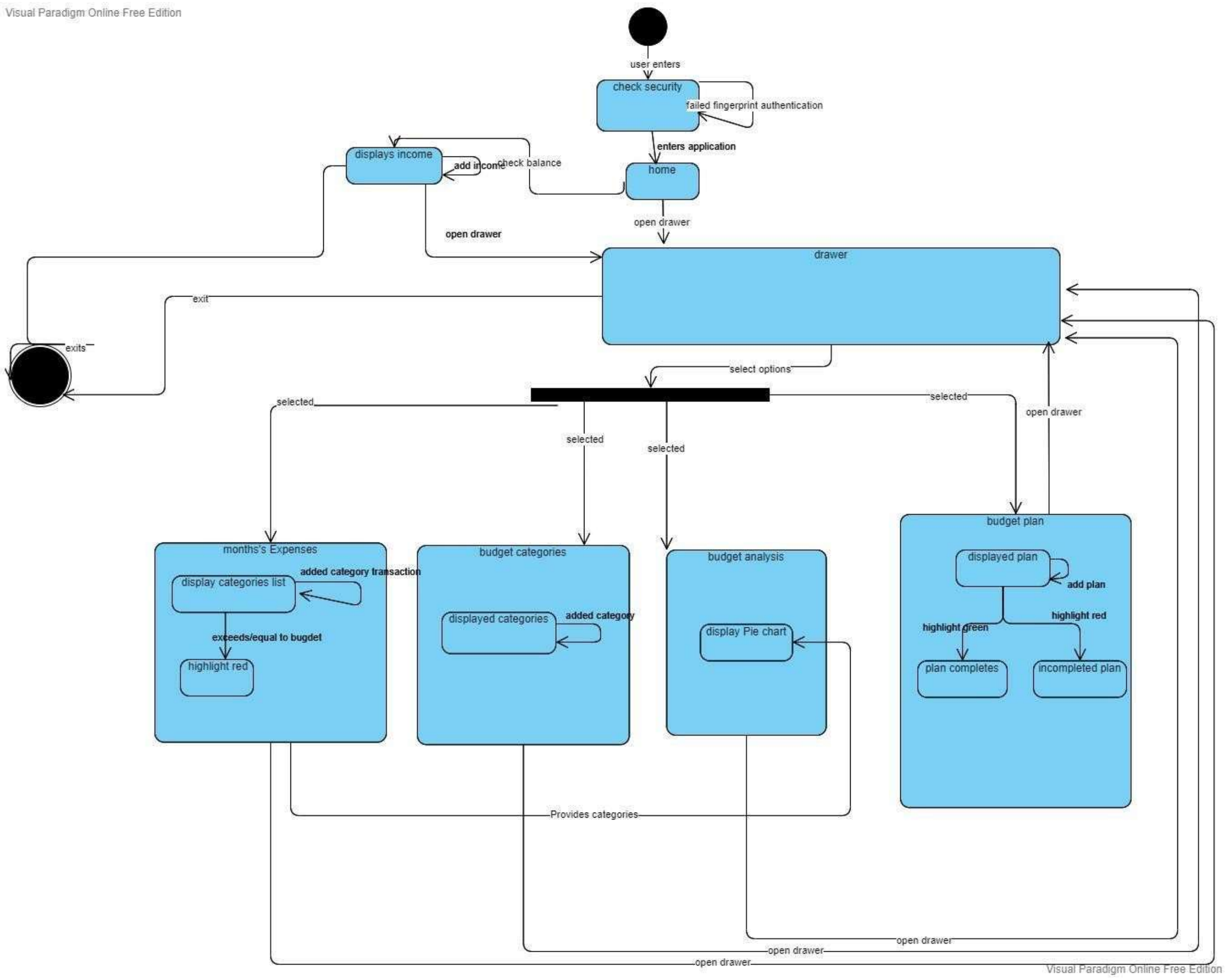
N.1 :- Database :
A database management system that is available free of cost in public domain should be used.

N.2 :- Platform :
android version of software need to be developed.

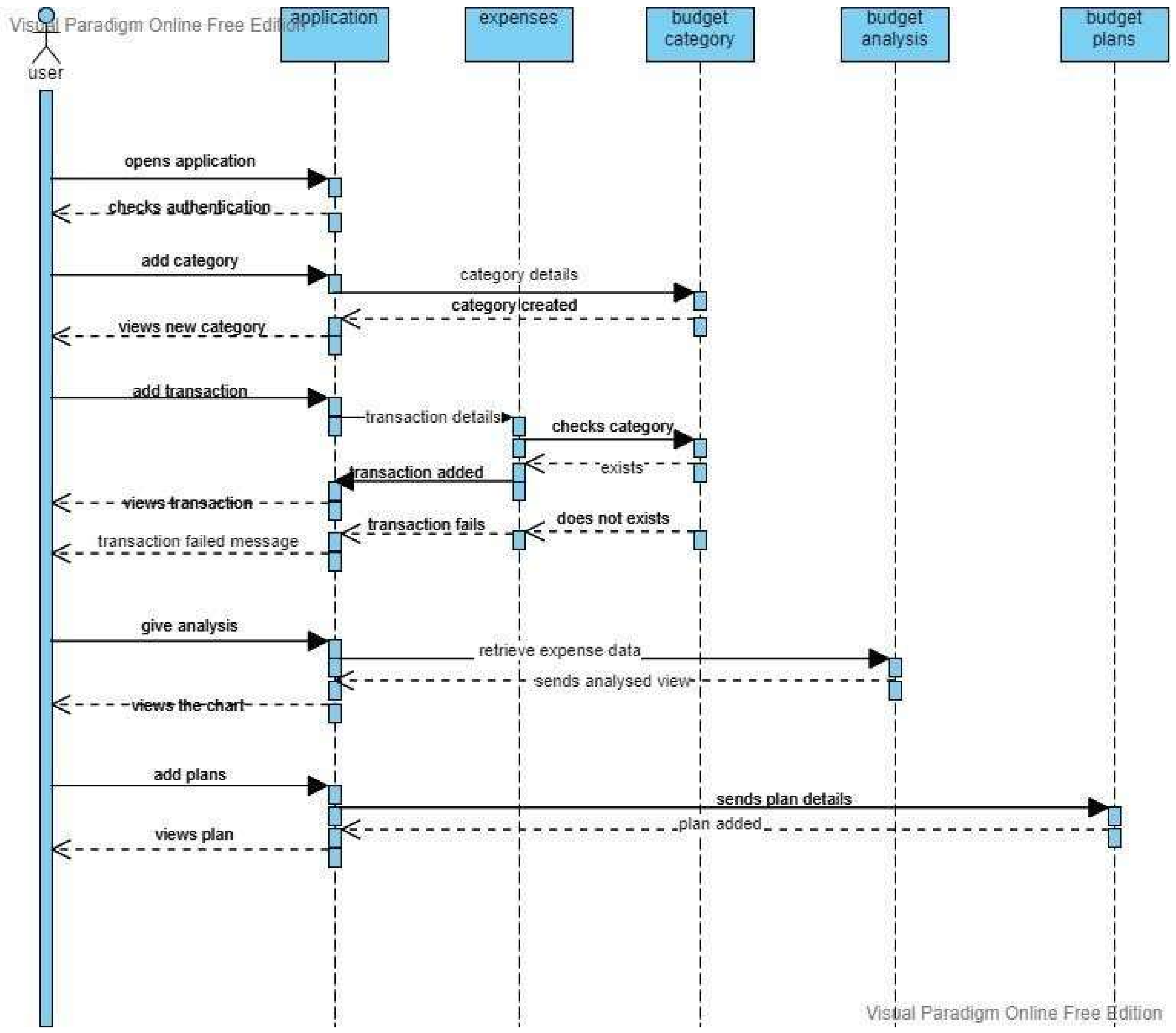
4. Designs



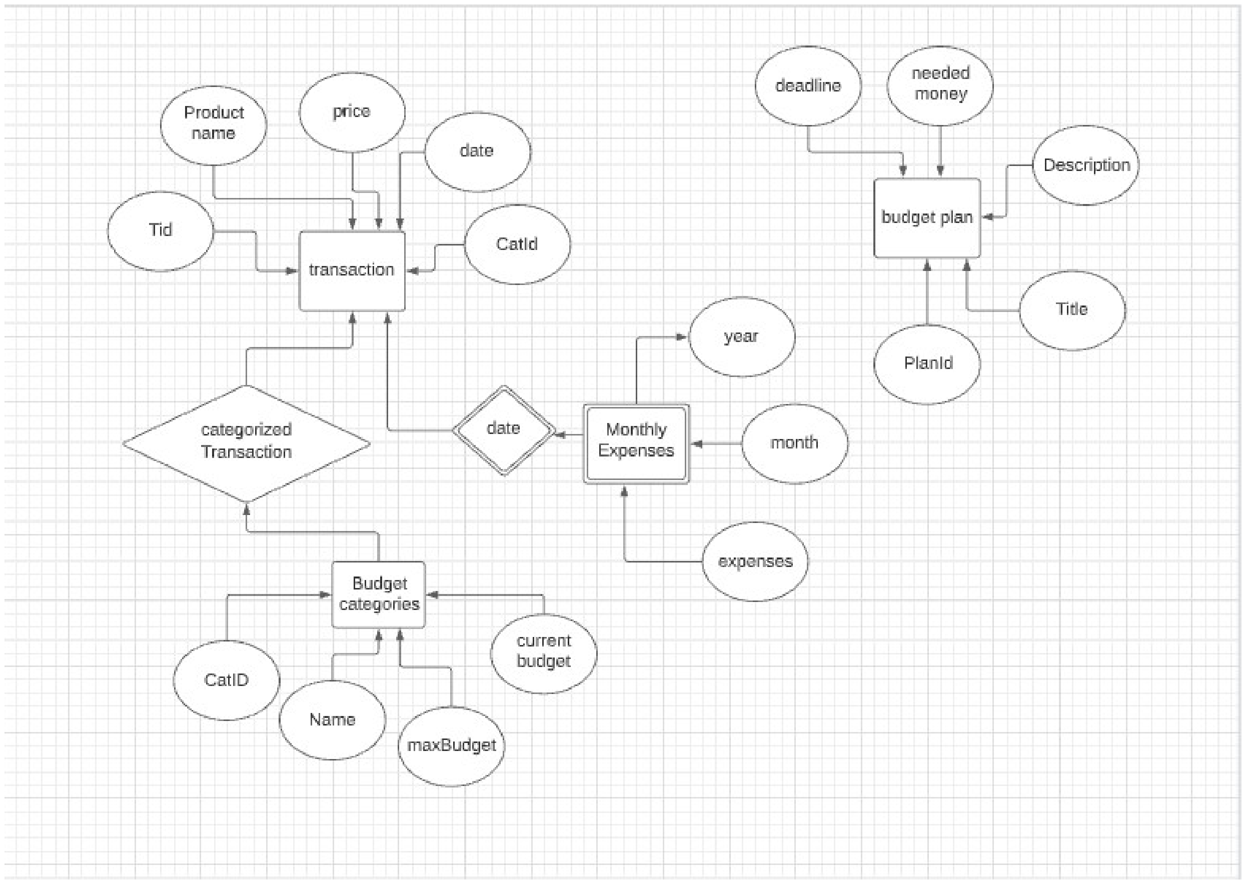
4.1 Use Case Diagram



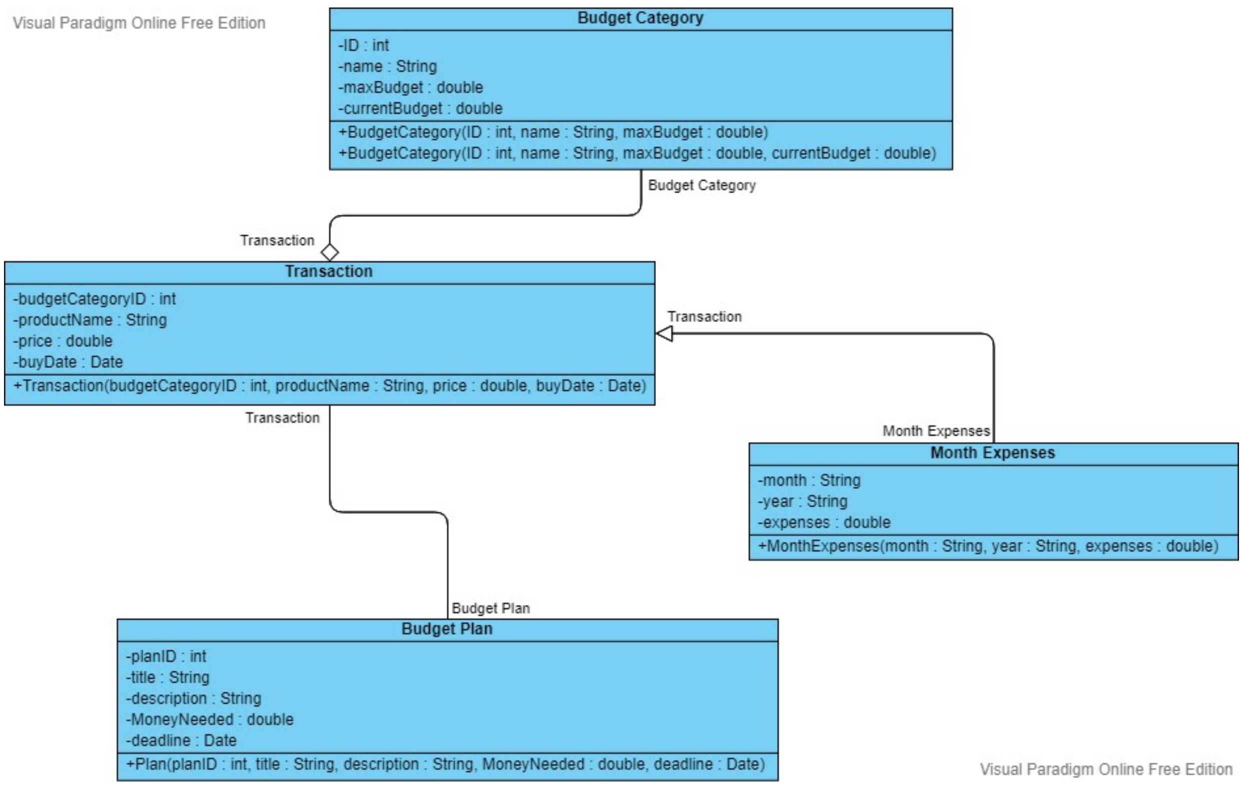
4.2 State diagram



4.3 sequence diagram



4.4 ER Diagram

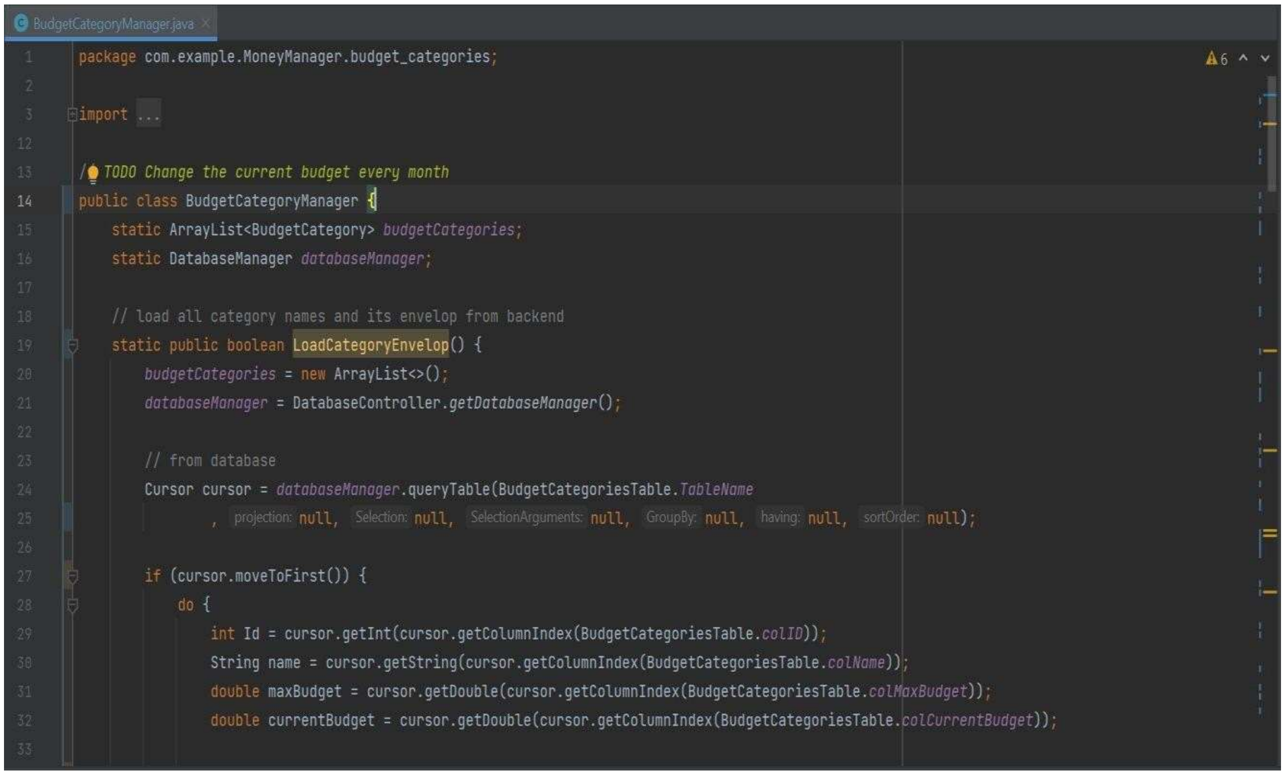

Visual Paradigm Online Free Edition
Budget Category
-ID : int
-name : String
-maxBudget : double
-currentBudget : double
+BudgetCategory(ID : int, name : String, maxBudget : double)
+BudgetCategory(ID : int, name : String, maxBudget : double, currentBudget : double)
Budget Category
Transaction
Transaction
-budgetCategoryID : int
-productName : String
-price : double
-buyDate : Date
+Transaction(budgetCategoryID : int, productName : String, price : double, buyDate : Date)
Transaction
Transaction
Month Expenses
Month Expenses
-month : String
-year : String
-expenses : double
+MonthExpenses(month : String, year : String, expenses : double)
Budget Plan
Budget Plan
-planID : int
-title : String
-description : String
-MoneyNeeded : double
-deadline : Date
+Plan(planID : int, title : String, description : String, MoneyNeeded : double, deadline : Date)
Visual Paradigm Online Free Edition

4.5 Class Diagram

5. Implementation Detail

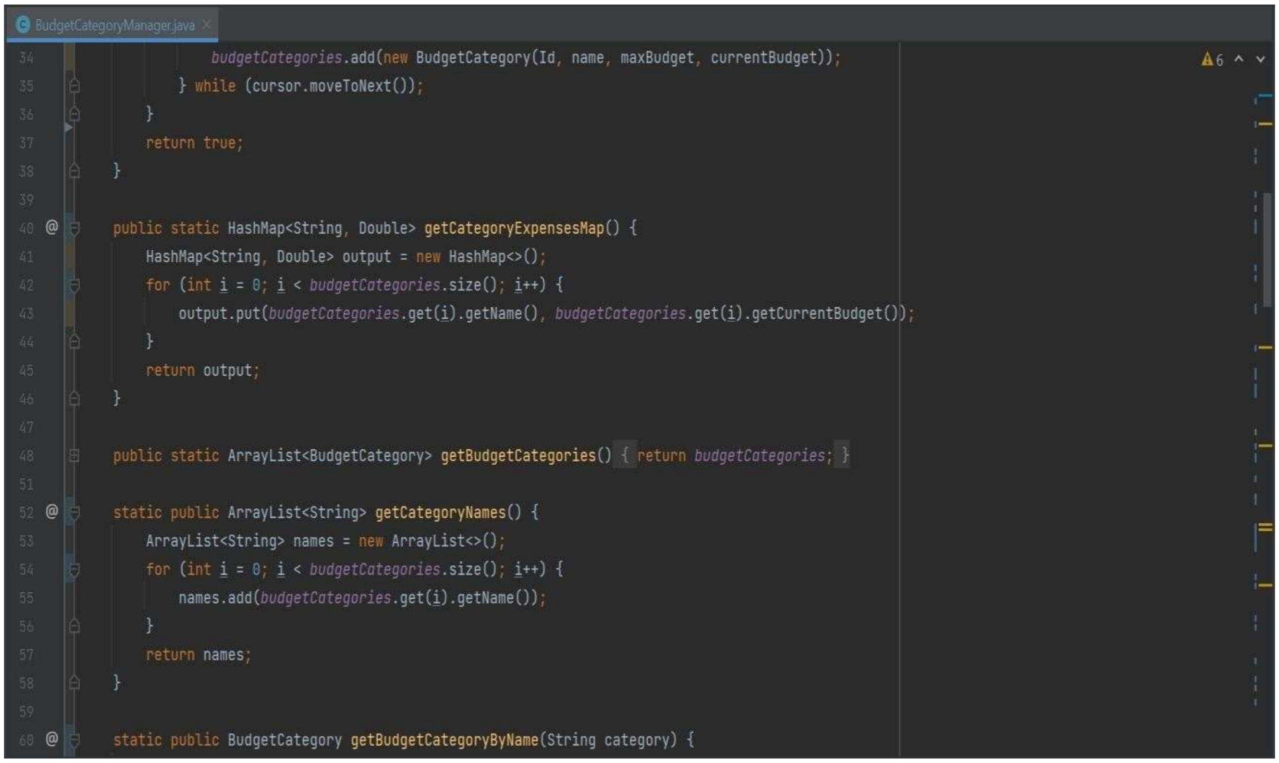
1. Budget Category

The Budget category module is a module where the user first starts with their execution of adding the transaction. Users first add the category and budget for that category in Budget category info and submit which then list into the budget categories table in the database. Now the user can enter the transaction from another module. After adding the category, the user can view it in the tab.The category in listview in tab are editable and its property can be changed.

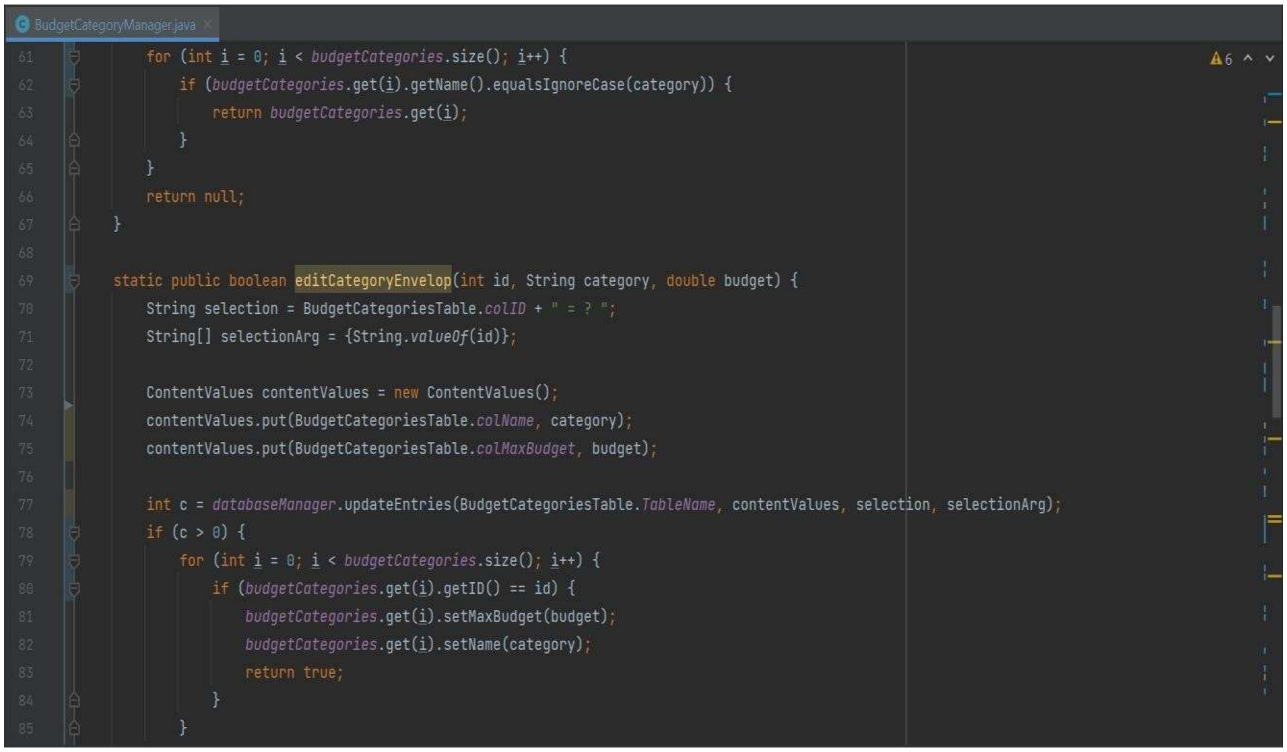


BudgetCategoryManager.java

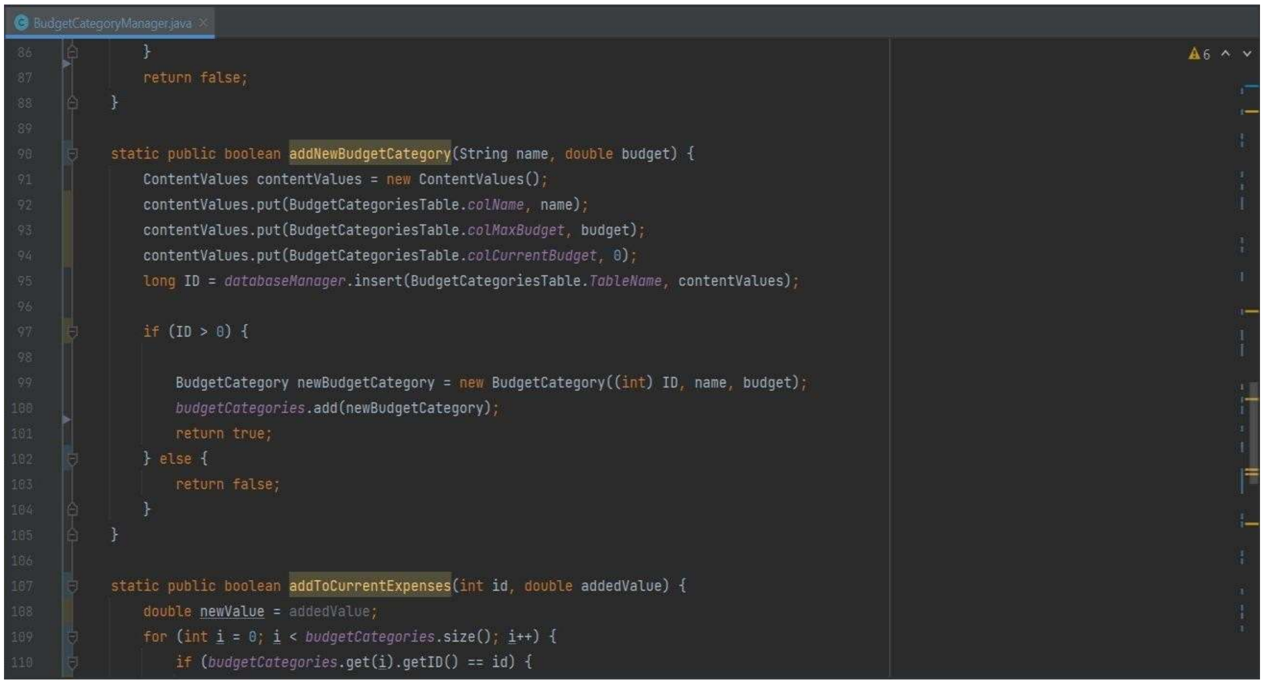
```
package com.example.MoneyManager.budget_categories;

import ...

// TODO Change the current budget every month
public class BudgetCategoryManager {
    static ArrayList<BudgetCategory> budgetCategories;
    static DatabaseManager databaseManager;

    // load all category names and its envelop from backend
    static public boolean LoadCategoryEnvelop() {
        budgetCategories = new ArrayList<>();
        databaseManager = DatabaseController.getDatabaseManager();

        // from database
        Cursor cursor = databaseManager.queryTable(BudgetCategoriesTable.TableName
                , projection: null, Selection: null, SelectionArguments: null, GroupBy: null, having: null, sortOrder: null);

        if (cursor.moveToFirst()) {
            do {
                int Id = cursor.getInt(cursor.getColumnIndex(BudgetCategoriesTable.colID));
                String name = cursor.getString(cursor.getColumnIndex(BudgetCategoriesTable.colName));
                double maxBudget = cursor.getDouble(cursor.getColumnIndex(BudgetCategoriesTable.colMaxBudget));
                double currentBudget = cursor.getDouble(cursor.getColumnIndex(BudgetCategoriesTable.colCurrentBudget));
```


BudgetCategoryManager.java

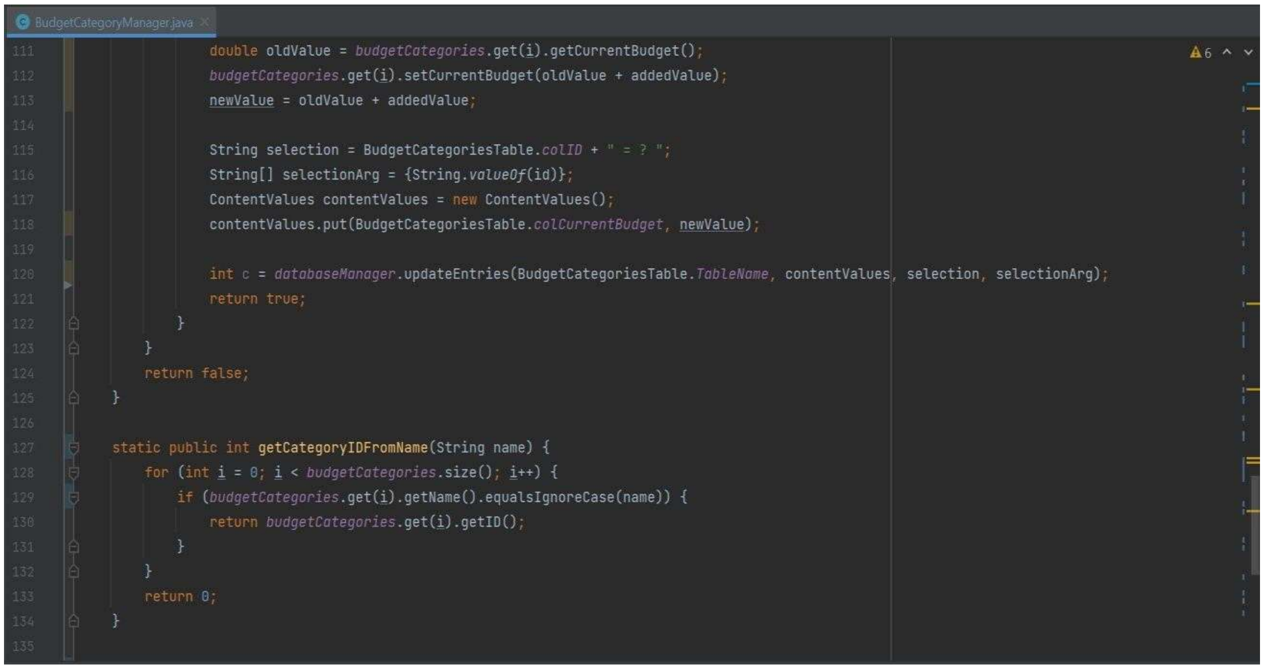
```
                budgetCategories.add(new BudgetCategory(Id, name, maxBudget, currentBudget));
            } while (cursor.moveToNext());
        }
        return true;
    }

    public static HashMap<String, Double> getCategoryExpensesMap() {
        HashMap<String, Double> output = new HashMap<>();
        for (int i = 0; i < budgetCategories.size(); i++) {
            output.put(budgetCategories.get(i).getName(), budgetCategories.get(i).getCurrentBudget());
        }
        return output;
    }

    public static ArrayList<BudgetCategory> getBudgetCategories() { return budgetCategories; }

    static public ArrayList<String> getCategoryNames() {
        ArrayList<String> names = new ArrayList<>();
        for (int i = 0; i < budgetCategories.size(); i++) {
            names.add(budgetCategories.get(i).getName());
        }
        return names;
    }

    static public BudgetCategory getBudgetCategoryByName(String category) {
```


BudgetCategoryManager.java

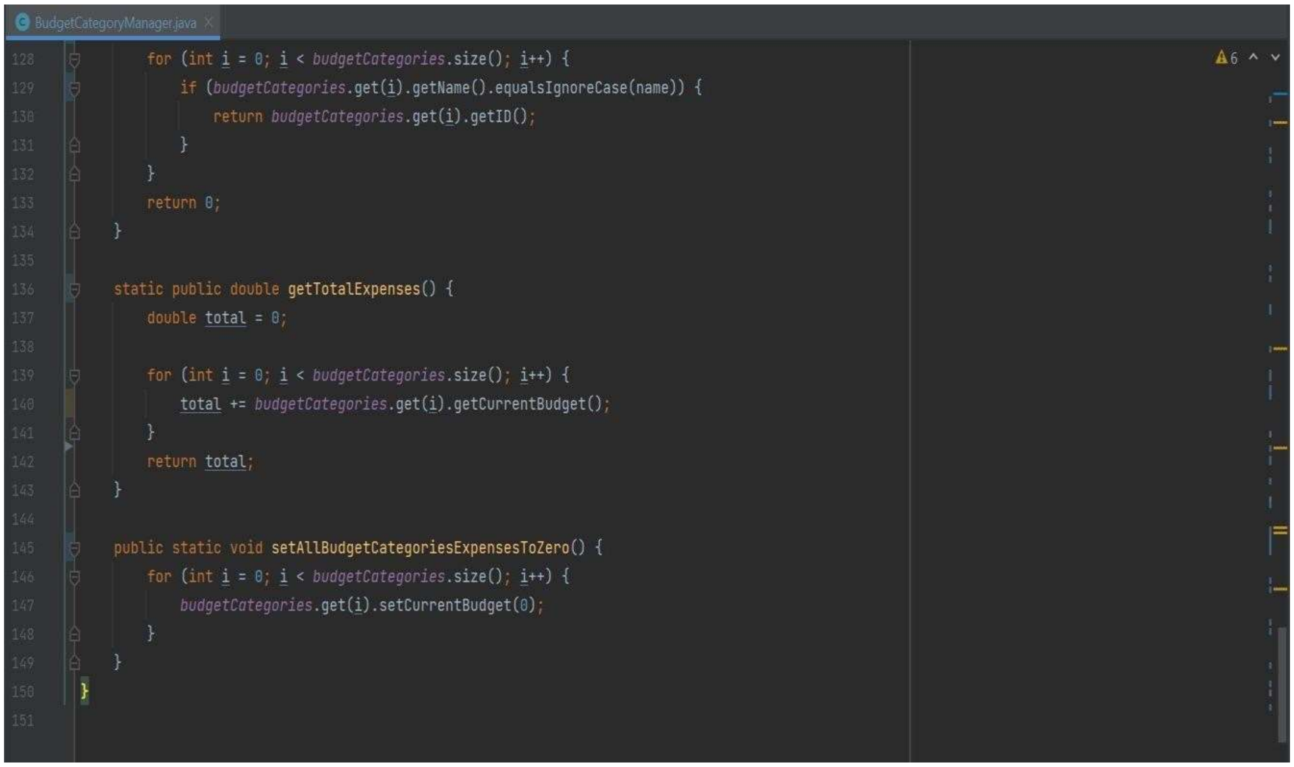
```
        for (int i = 0; i < budgetCategories.size(); i++) {
            if (budgetCategories.get(i).getName().equalsIgnoreCase(category)) {
                return budgetCategories.get(i);
            }
        }
        return null;
    }

    static public boolean editCategoryEnvelop(int id, String category, double budget) {
        String selection = BudgetCategoriesTable.colID + " = ? ";
        String[] selectionArg = {String.valueOf(id)};

        ContentValues contentValues = new ContentValues();
        contentValues.put(BudgetCategoriesTable.colName, category);
        contentValues.put(BudgetCategoriesTable.colMaxBudget, budget);

        int c = databaseManager.updateEntries(BudgetCategoriesTable.TableName, contentValues, selection, selectionArg);
        if (c > 0) {
            for (int i = 0; i < budgetCategories.size(); i++) {
                if (budgetCategories.get(i).getID() == id) {
                    budgetCategories.get(i).setMaxBudget(budget);
                    budgetCategories.get(i).setName(category);
                    return true;
                }
            }
```



```java
        }
        return false;
    }

    static public boolean addNewBudgetCategory(String name, double budget) {
        ContentValues contentValues = new ContentValues();
        contentValues.put(BudgetCategoriesTable.colName, name);
        contentValues.put(BudgetCategoriesTable.colMaxBudget, budget);
        contentValues.put(BudgetCategoriesTable.colCurrentBudget, 0);
        long ID = databaseManager.insert(BudgetCategoriesTable.TableName, contentValues);

        if (ID > 0) {

            BudgetCategory newBudgetCategory = new BudgetCategory((int) ID, name, budget);
            budgetCategories.add(newBudgetCategory);
            return true;
        } else {
            return false;
        }
    }

    static public boolean addToCurrentExpenses(int id, double addedValue) {
        double newValue = addedValue;
        for (int i = 0; i < budgetCategories.size(); i++) {
            if (budgetCategories.get(i).getID() == id) {
```



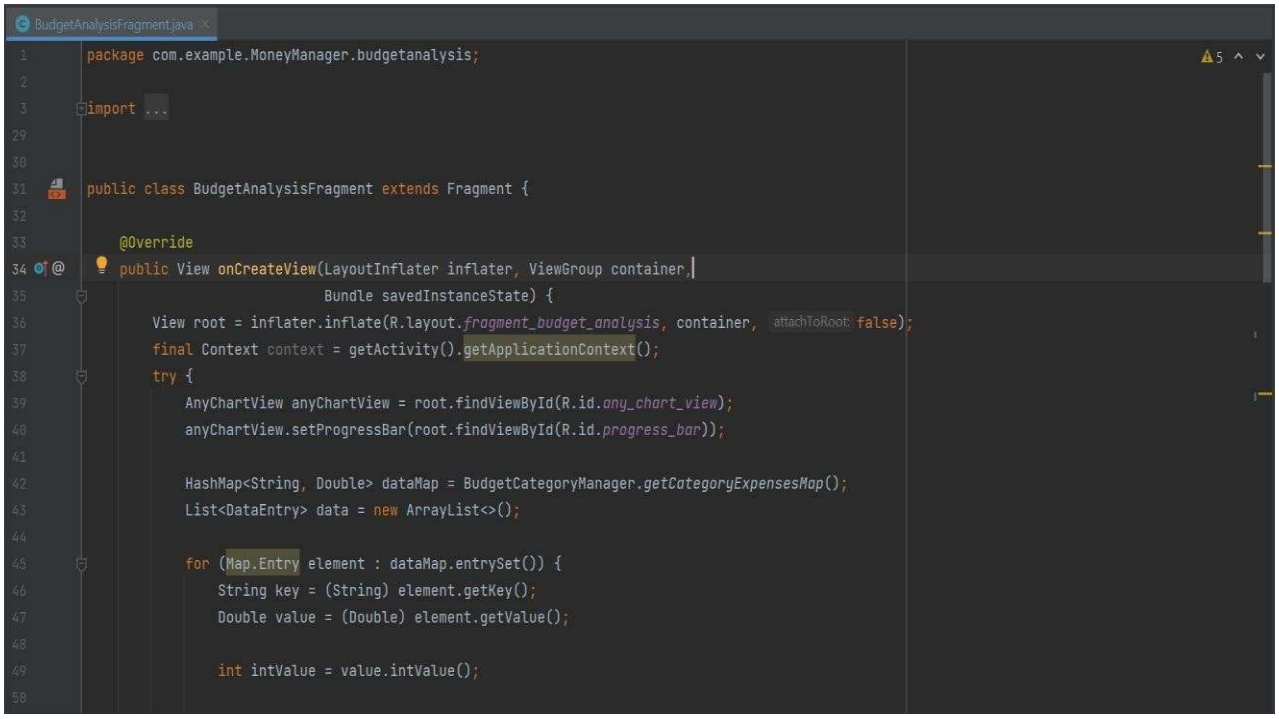
```java
                double oldValue = budgetCategories.get(i).getCurrentBudget();
                budgetCategories.get(i).setCurrentBudget(oldValue + addedValue);
                newValue = oldValue + addedValue;

                String selection = BudgetCategoriesTable.colID + " = ? ";
                String[] selectionArg = {String.valueOf(id)};
                ContentValues contentValues = new ContentValues();
                contentValues.put(BudgetCategoriesTable.colCurrentBudget, newValue);

                int c = databaseManager.updateEntries(BudgetCategoriesTable.TableName, contentValues, selection, selectionArg);
                return true;
            }
        }
        return false;
    }

    static public int getCategoryIDFromName(String name) {
        for (int i = 0; i < budgetCategories.size(); i++) {
            if (budgetCategories.get(i).getName().equalsIgnoreCase(name)) {
                return budgetCategories.get(i).getID();
            }
        }
        return 0;
    }

```


BudgetCategoryManager.java

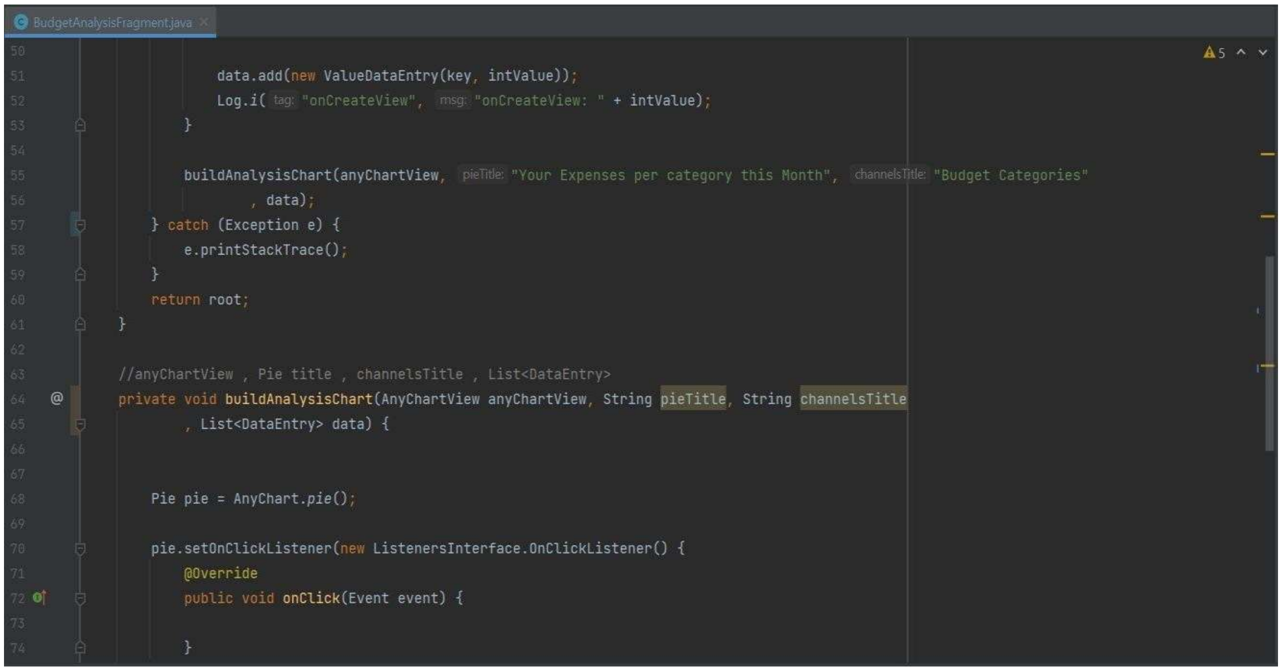
```java
        for (int i = 0; i < budgetCategories.size(); i++) {
            if (budgetCategories.get(i).getName().equalsIgnoreCase(name)) {
                return budgetCategories.get(i).getID();
            }
        }
        return 0;
    }

    static public double getTotalExpenses() {
        double total = 0;

        for (int i = 0; i < budgetCategories.size(); i++) {
            total += budgetCategories.get(i).getCurrentBudget();
        }
        return total;
    }

    public static void setAllBudgetCategoriesExpensesToZero() {
        for (int i = 0; i < budgetCategories.size(); i++) {
            budgetCategories.get(i).setCurrentBudget(0);
        }
    }
}
```

5.2 Budget Analysis

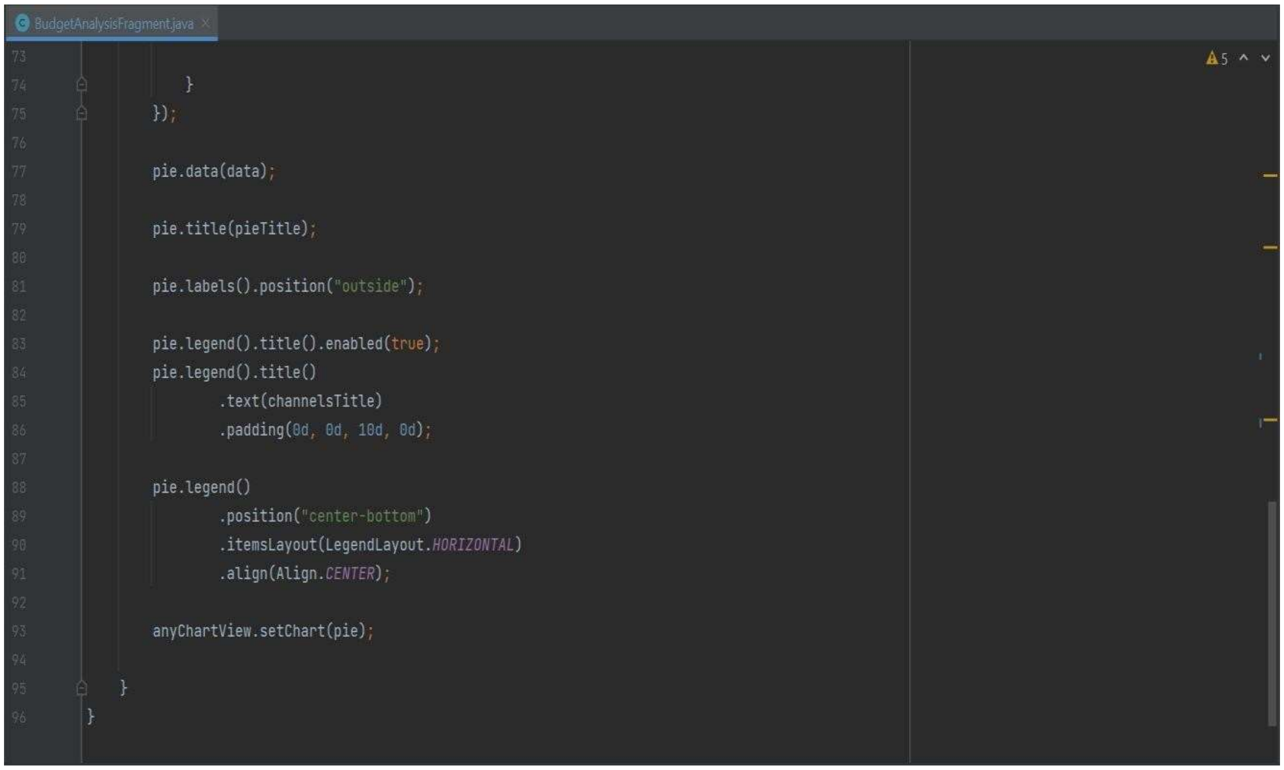
This module consists of developing a pie chart of all the data present in the current month. Everytime a transaction/data is added, the pie chart changes accordingly.



```java
package com.example.MoneyManager.budgetanalysis;

import ...


public class BudgetAnalysisFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_budget_analysis, container,  attachToRoot: false);
        final Context context = getActivity().getApplicationContext();
        try {
            AnyChartView anyChartView = root.findViewById(R.id.any_chart_view);
            anyChartView.setProgressBar(root.findViewById(R.id.progress_bar));

            HashMap<String, Double> dataMap = BudgetCategoryManager.getCategoryExpensesMap();
            List<DataEntry> data = new ArrayList<>();

            for (Map.Entry element : dataMap.entrySet()) {
                String key = (String) element.getKey();
                Double value = (Double) element.getValue();

                int intValue = value.intValue();

```



```java

                data.add(new ValueDataEntry(key, intValue));
                Log.i( tag: "onCreateView",  msg: "onCreateView: " + intValue);
            }

            buildAnalysisChart(anyChartView,  pieTitle: "Your Expenses per category this Month",  channelsTitle: "Budget Categories"
                    , data);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return root;
    }

    //anyChartView , Pie title , channelsTitle , List<DataEntry>
    private void buildAnalysisChart(AnyChartView anyChartView, String pieTitle, String channelsTitle
            , List<DataEntry> data) {


        Pie pie = AnyChart.pie();

        pie.setOnClickListener(new ListenersInterface.OnClickListener() {
            @Override
            public void onClick(Event event) {

            }
```

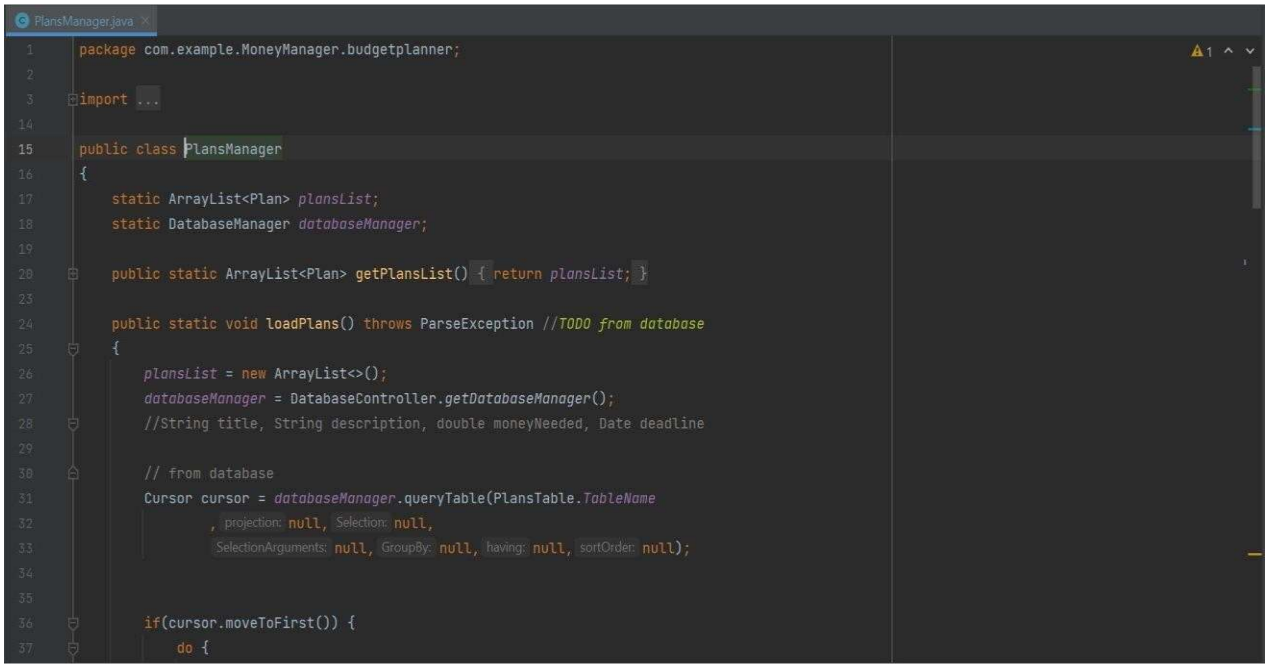


BudgetAnalysisFragment.java

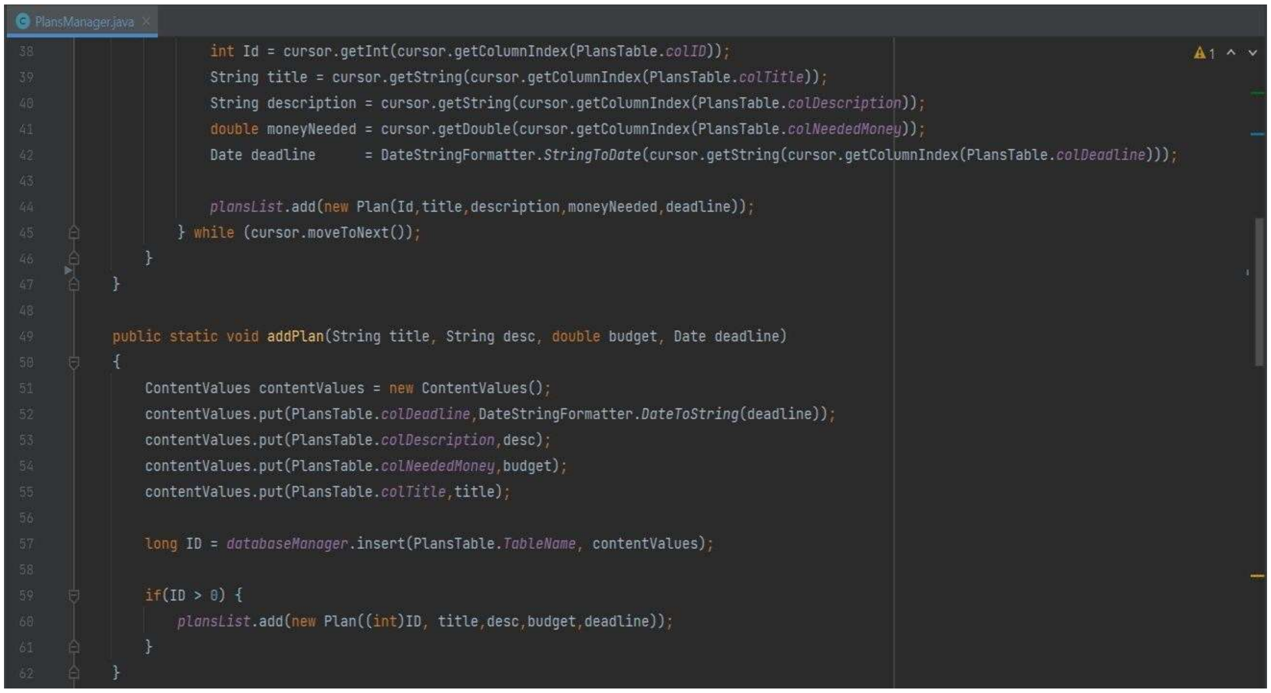
```
                }
            });

            pie.data(data);

            pie.title(pieTitle);

            pie.labels().position("outside");

            pie.legend().title().enabled(true);
            pie.legend().title()
                    .text(channelsTitle)
                    .padding(0d, 0d, 10d, 0d);

            pie.legend()
                    .position("center-bottom")
                    .itemsLayout(LegendLayout.HORIZONTAL)
                    .align(Align.CENTER);

            anyChartView.setChart(pie);

        }
    }
```

5.3 Budget Planner

The primary task of this module is to set a reminder for expenses which are not priority but need to be completed. Here users can add their plan to make a purchase by entering its title, description, budget and date. This data gets stored in its table in the database and displayed in the budget plan tab. After completion of all the expenses of the month, if the user finds a plan in budget plans highlighted to be green, it indicates the user has enough balance to make that expense. If it is highlighted as red, the user does not have enough balance. This module proposes a saving to buy theory to users.


PlansManager.java

```java
package com.example.MoneyManager.budgetplanner;

import ...

public class PlansManager
{
    static ArrayList<Plan> plansList;
    static DatabaseManager databaseManager;

    public static ArrayList<Plan> getPlansList() { return plansList; }

    public static void loadPlans() throws ParseException //TODO from database
    {
        plansList = new ArrayList<>();
        databaseManager = DatabaseController.getDatabaseManager();
        //String title, String description, double moneyNeeded, Date deadline

        // from database
        Cursor cursor = databaseManager.queryTable(PlansTable.TableName
                , projection: null, Selection: null,
                SelectionArguments: null, GroupBy: null, having: null, sortOrder: null);


        if(cursor.moveToFirst()) {
            do {
```


PlansManager.java

```java
                int Id = cursor.getInt(cursor.getColumnIndex(PlansTable.colID));
                String title = cursor.getString(cursor.getColumnIndex(PlansTable.colTitle));
                String description = cursor.getString(cursor.getColumnIndex(PlansTable.colDescription));
                double moneyNeeded = cursor.getDouble(cursor.getColumnIndex(PlansTable.colNeededMoney));
                Date deadline      = DateStringFormatter.StringToDate(cursor.getString(cursor.getColumnIndex(PlansTable.colDeadline)));

                plansList.add(new Plan(Id,title,description,moneyNeeded,deadline));
            } while (cursor.moveToNext());
        }
    }

    public static void addPlan(String title, String desc, double budget, Date deadline)
    {
        ContentValues contentValues = new ContentValues();
        contentValues.put(PlansTable.colDeadline,DateStringFormatter.DateToString(deadline));
        contentValues.put(PlansTable.colDescription,desc);
        contentValues.put(PlansTable.colNeededMoney,budget);
        contentValues.put(PlansTable.colTitle,title);

        long ID = databaseManager.insert(PlansTable.TableName, contentValues);

        if(ID > 0) {
            plansList.add(new Plan((int)ID, title,desc,budget,deadline));
        }
    }
```

6. Testing

6.1 Months expenses

Sr. No	Test Scenario	Expected Result	Actual Result	Status
1	Open expense activity from drawer	Open and shows informati on	Opened and showed informati on	success
2	Add Transacti on	Shows on home fragment	Showed on home fragment	success
3	Category exceed budget	Shows red highlighte d budget	Showed red highlighte d budget	success
4	Transacti on informati on	Shows info of transactio n	showed	success

6.2 Budget Categories

Sr. No	Test Scenario	Expected Result	Actual Result	Status
1	Adding Category	Shows category with budget on budget categories fragment	Shows the info on fragment	success
2	Adding Category without budget	Shows message of enter budget	Showed the message	success

6.3 Budget Analysis

Sr. No	Test Scenario	Expected result	Actual result	Status
1	Show pie chart of expenses	Displays Pie chart	Displayed Pie chart	success

6.4 Budget Plans

Sr. No	Test Scenario	Expected result	Actual result	Status
1	Add plan info	Shows on plan fragment	showed	success
2	Plan with deadline passed	Shows red highlighted plan	Showed the red highlighted plan	success
3	Plan which can be completed	Shows green highlighted plan	showed	success

4	Plan with no enough money and not near deadline	Shows only on plan fragment	showed	success

6.5 Income

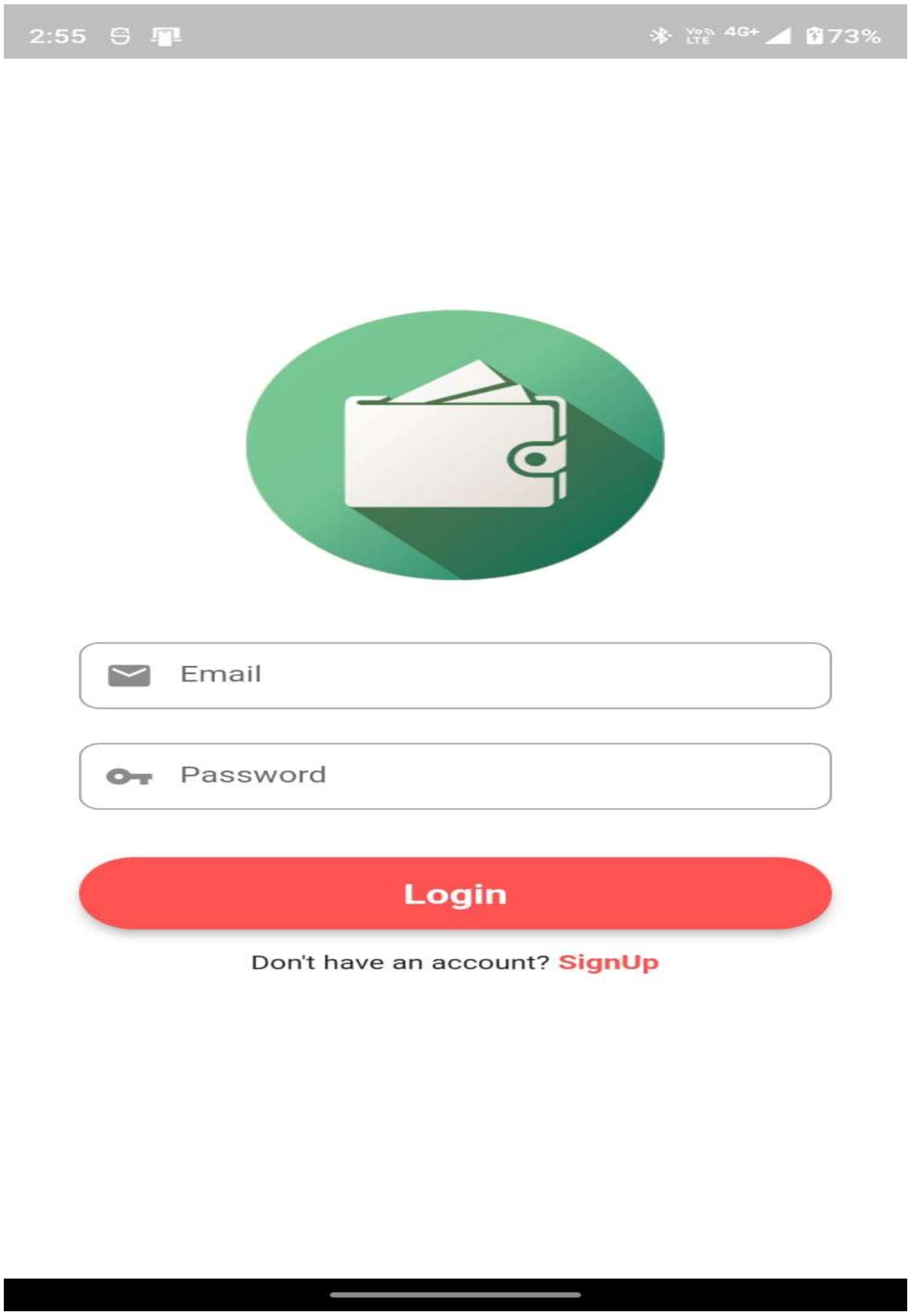
Sr.No	Test scenario	Expected result	Actual result	Status
1	Add money	Shows in wallet	Showed in wallet	success

6.6 Security

Sr.No	Test scenario	Expected result	Actual result	Status
1	Place Fingerprint	Authorized and enters in application	Authorized and entered the application	success

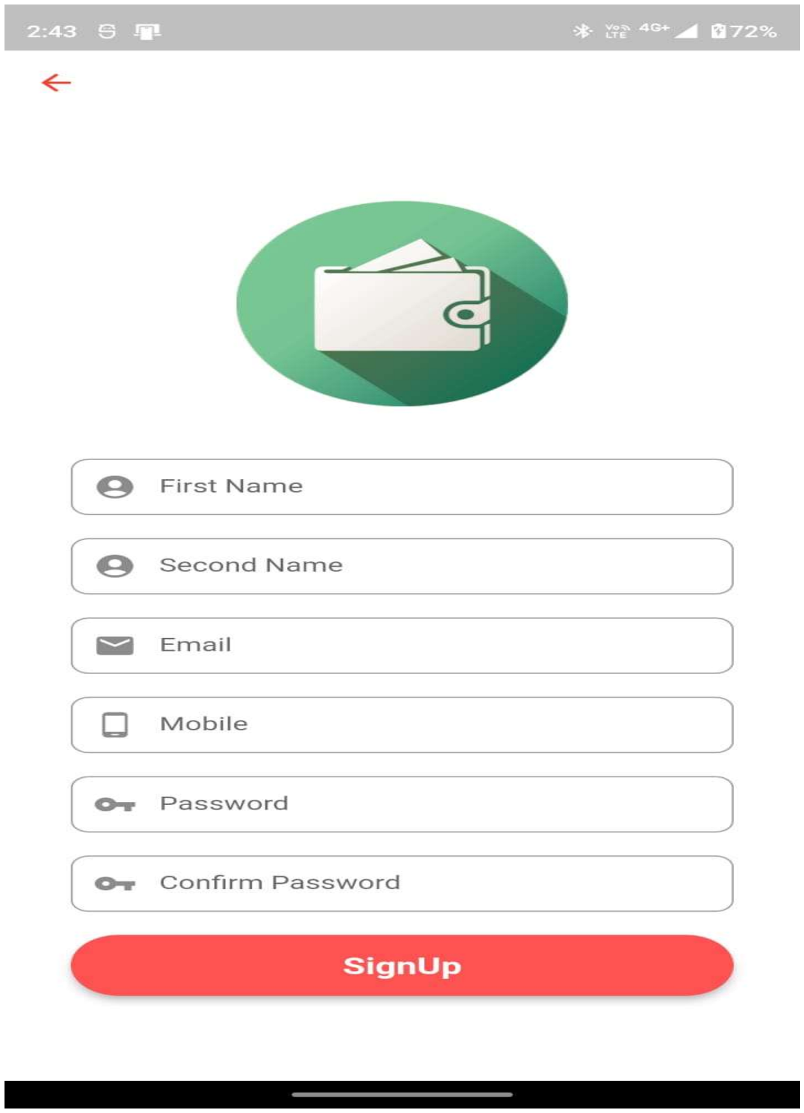
7. Screen-shots

1. Home

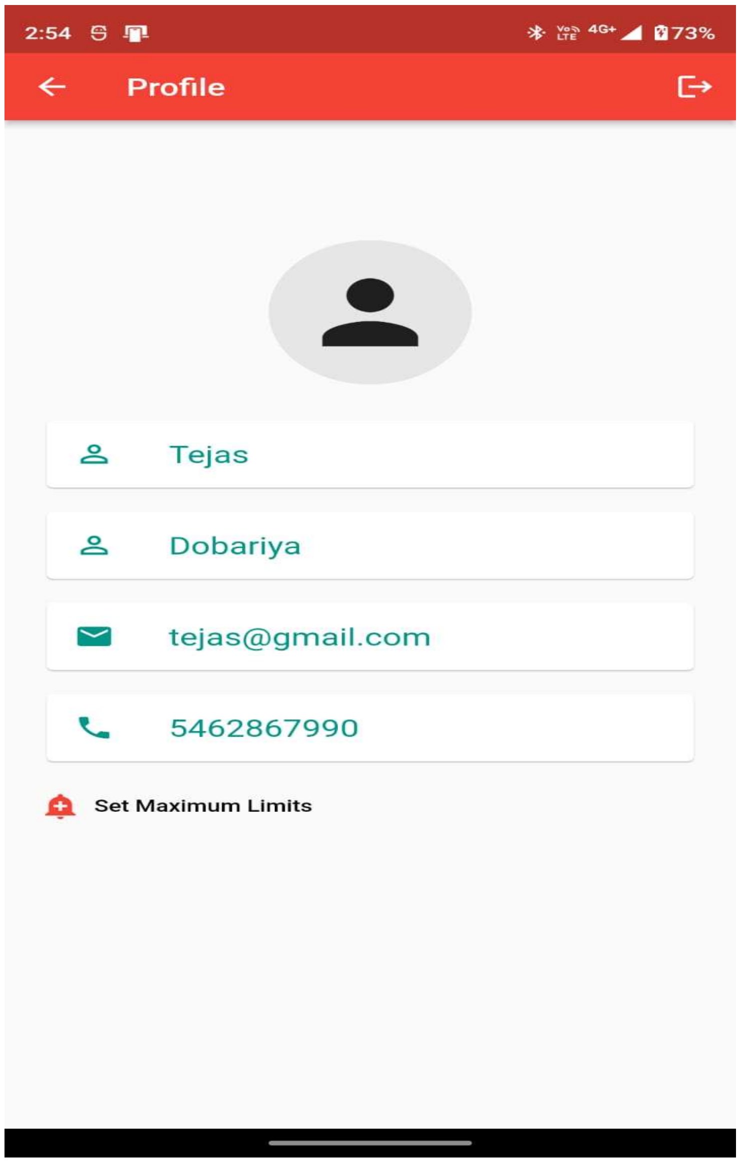


Don't have an account? SignUp

7.2 Sign up page



7.3 My Profile



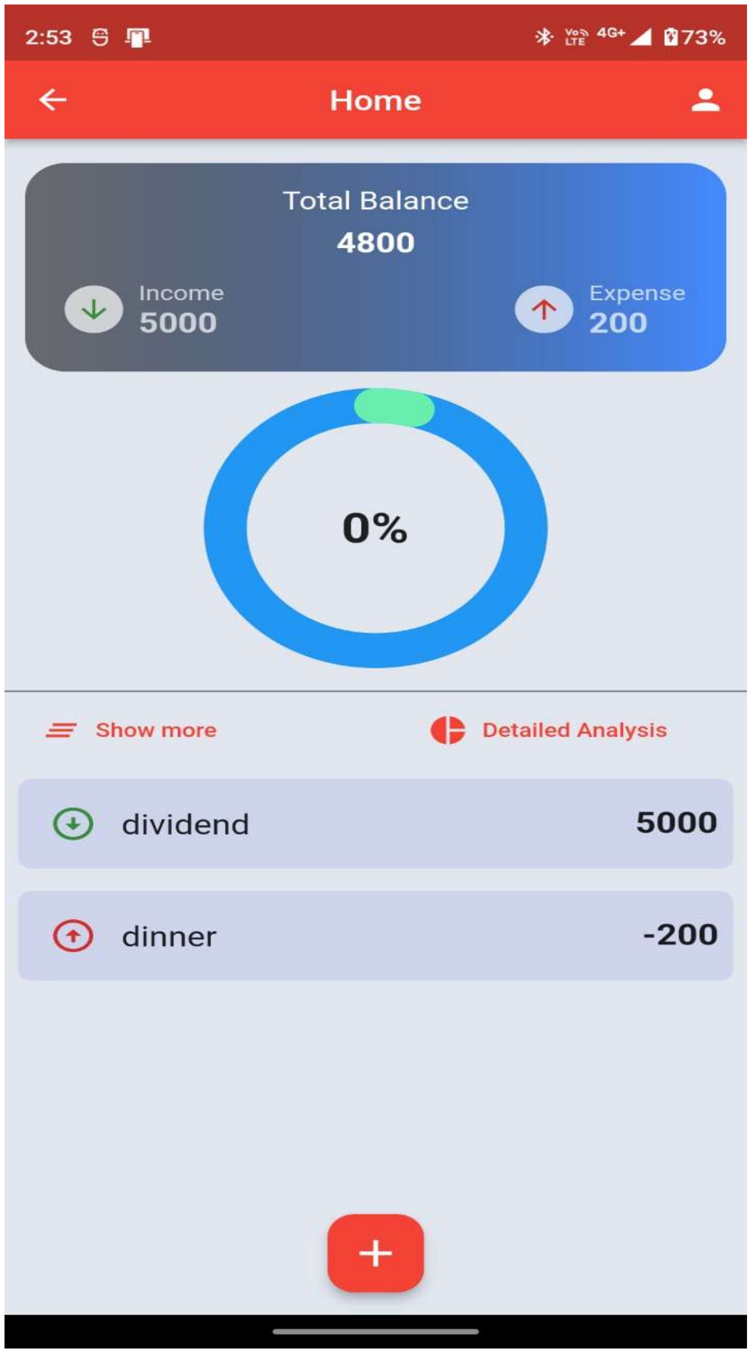
Tejas

Dobariya

tejas@gmail.com

5462867990

7.4 Home Page



Total Balance
4800

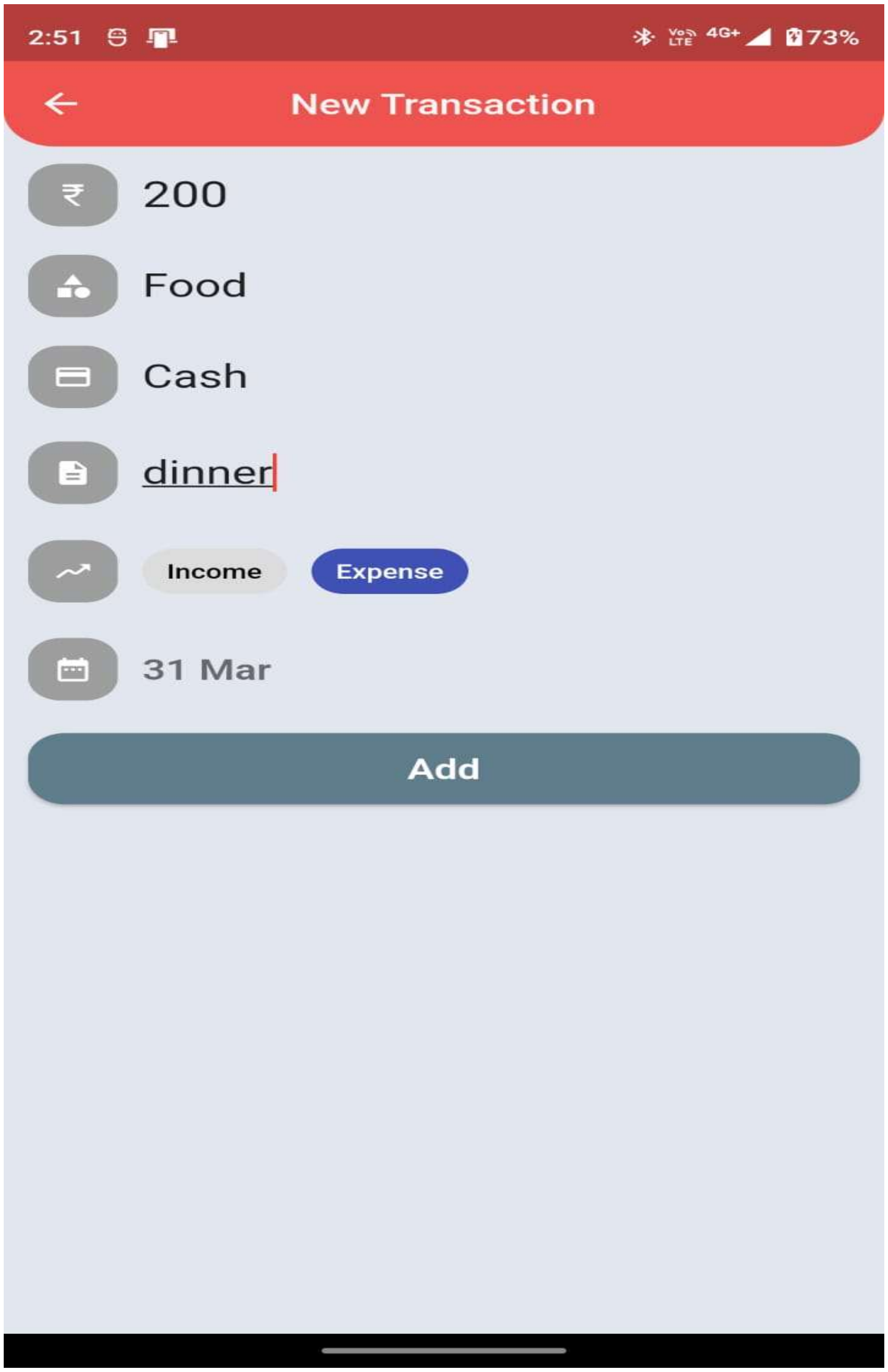
Income
5000

Expense
200

0%

dividend	5000
dinner	-200

7.5 Add Expense


2:51 73%

New Transaction

200

Food

Cash

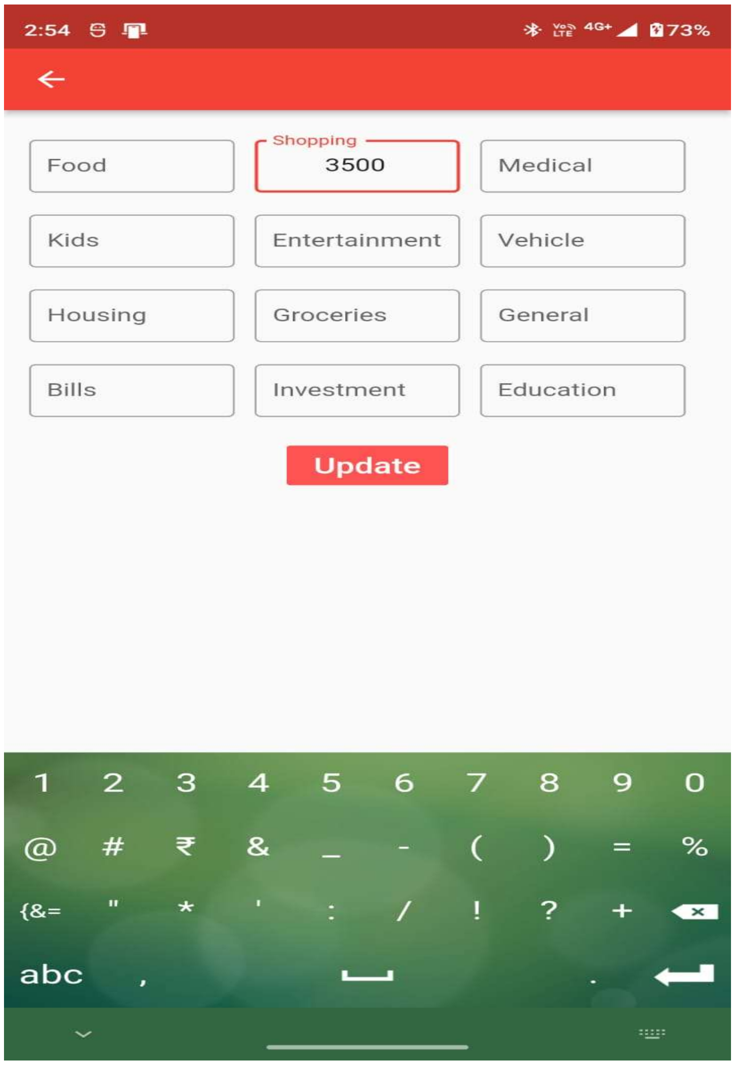
dinner

Income [✓ Expense]

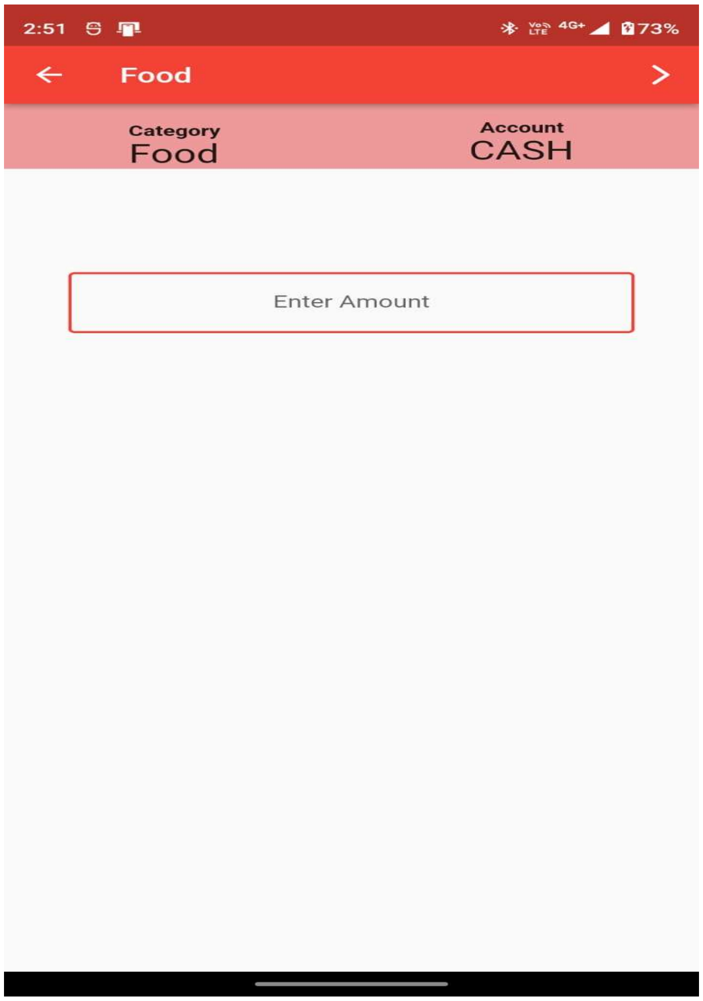
31 Mar

Add

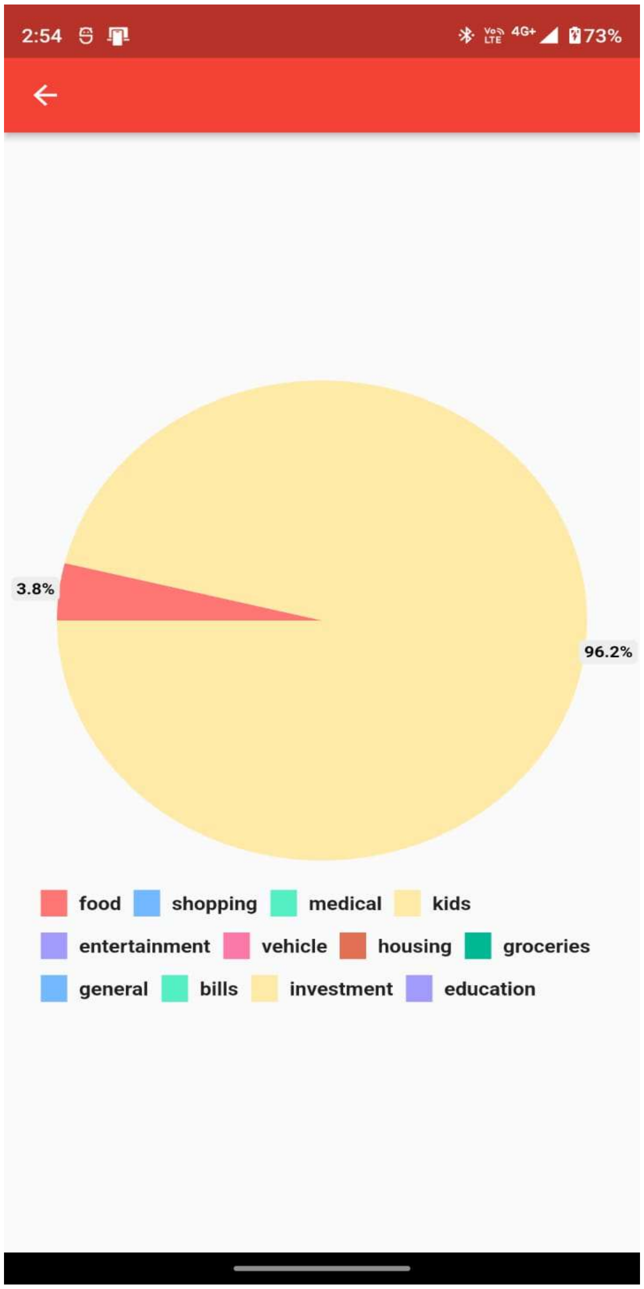
7.6 Update Limit



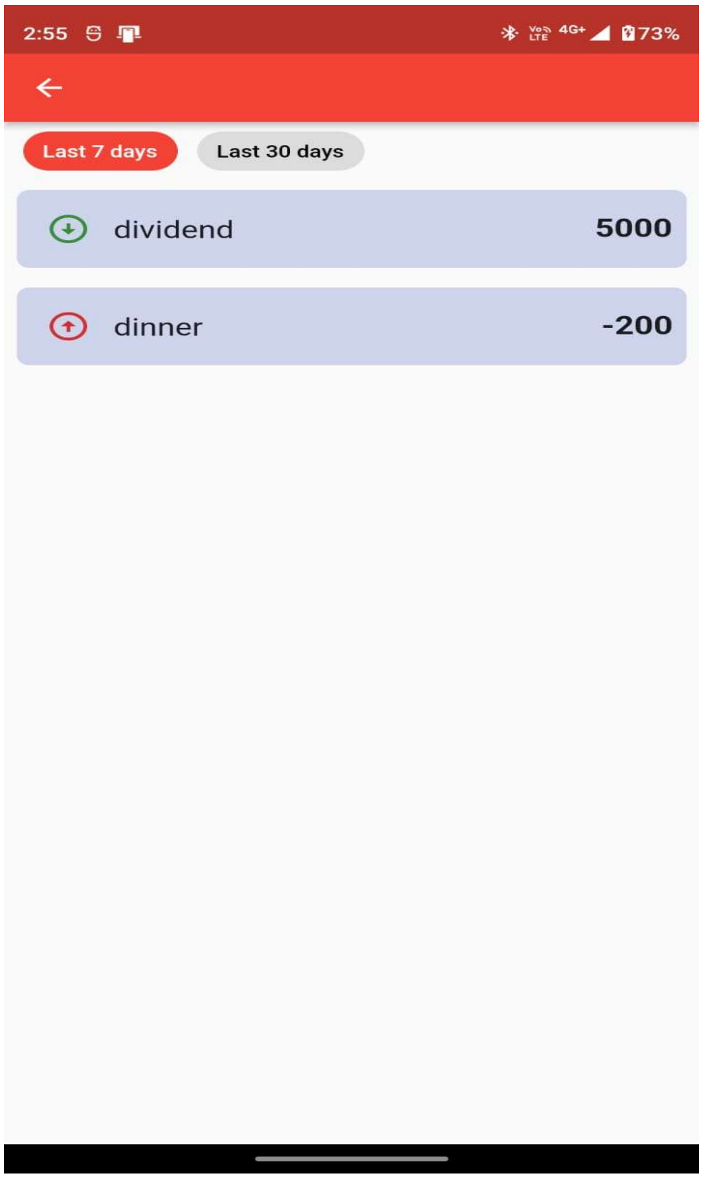
7.7 Set Limit



7.8 View PieChart



7.9 View Expenses



dividend 5000

dinner -200

8. Conclusion

As per the Software Requirement Specification, we were able to make all major features in our application.

The Add transaction was done with precision as we also want to set the categories. We set the add transaction in such a way that it gets associated with its category. We also took the date of the transaction into account. Previous dates transaction can also be added in later on days

The Add Category was built in such a way that the budget gets added to the category to check later on that a particular category does not get overspend by the user and the user reamined informed.

The Add Plan is a functionality which provides users an enhanced experience in application by checking whether the user can buy or not a product.

The Add Income is functionality in which user add money. This will get added to the wallet of the user.

9. Limitation and Future Extension

The Limitation to the projects are as follows:-

- The user cannot add income as a transaction.
- One time expense cannot be added if there is no category for the expense.

Future Extension of the application can be as follows:-

- Add income as Transaction
- Provide a excel file of transaction
- One time expense can be built.

10. Bibliography

Smyth, Neil. Android Studio 3.0 Development Essentials - Android 8 Edition. Payload Media, Inc., 2017.

“Documentation.” Android Developers, https://developer.android.com/docs. Accessed 12 February 2022.

Stack Overflow - Where Developers Learn, Share, & Build Careers, http://www.stackoverflow.com. Accessed 28 March 2022.

“Android - Studio.” Tutorialspoint, https://www.tutorialspoint.com/android/android_studio.htm. Accessed 28 March 2022.