SILIGURI INSTITUTE OF TECHNOLOGY

PROJ- CS881

Expense Tracker System

BY

IT_PROJ_2023_03

Name of Students	Roll No.
1. Abhi Kumar Sahu	11900219051
2. Alkesh Raj	11900219054
3. Divyam Mishra	11900219036
4. Rudranil Ghosh	11900219032

Under the Guidance

of

DR. ASIT BARMAN

Submitted to the Department of **Information Technology** in partial fulfillment of the requirements for the award of the degree Bachelor of Technology in **Information Technology.**

Year of Submission: 2023

Siliguri Institute of Technology

P.O. SUKNA, SILIGURI, DIST. DARJEELING, PIN: 734009

Tel: (0353)2778002/04, Fax: (0353) 2778003

DECLARATION

-

This is to certify that the Report entitled "**Expense Tracker System**" which is submitted by me in partial fulfillment of the requirement for the award of degree B.Tech. in **Information Technology** at **Siliguri Institute of Technology** under **Maulana Abul Kalam Azad University of Technology**, West Bengal. We took the help of other materials in our dissertation which have been properly acknowledged. This report has not been submitted to any other Institute for the award of any other degree.

Date : 25 May 2023

SN	Name of the Student	Roll No	Signature
1	Abhi Kumar Sahu	11900219051	
2	Alkesh Raj	11900219054	
3	Divyam Mishra	11900219036	
4	Rudranil Ghosh	11900219032	

CERTIFICATE

This is to certify that the project report entitled ____________________ ____________________ **" Expense Tracker System "**________________________________ submitted to **the Department of Information Technology of Siliguri Institute of Technology** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Information Technology** during the academic year **2022-23,** is a bonafide record of the project work carried out by them under my guidance and supervision.

Project Group Number :			
SN	**Name of the students**	**Registration No**	**Roll No**
1.	Abhi Kumar Sahu	035520 OF 2019-20	11900219051
2.	Alkesh Raj	035511 OF 2019-20	11900219054
3.	Divyam Mishra	035826 OF 2019-20	11900219036
4.	Rudranil Ghosh	036059 OF 2019-20	11900219032

--------------------------------------- --

Signature of Project Guide **Signature of the HOD**

Name of the Guide: **Department of Information Technology**

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude and appreciation to the esteemed professors who have played a significant role in the successful completion of this project. Their invaluable guidance, encouragement, and expertise have been instrumental in shaping the project's outcome, and I am truly grateful for their unwavering support throughout this journey.

First and foremost, I would like to extend my deepest gratitude to **Dr. Asit Barman**, whose insightful guidance and mentorship have been indispensable throughout this project. Your vast knowledge, dedication to teaching, and willingness to share your expertise have truly inspired me to push my boundaries and strive for excellence. Your constructive feedback and valuable suggestions have not only improved the quality of this project but have also broadened my understanding of the subject matter.

Lastly, I would like to express my heartfelt thanks to my family and friends for their unwavering support, encouragement, and understanding throughout this project. Their belief in my abilities and their constant motivation has been crucial in overcoming challenges and reaching the finish line.

Signature of all the group members with the date

1.

2.

3.

4.

ABSTRACT

The Expense Tracker System is an Android application developed using Android Studio and Java programming language. It leverages the Firebase backend service provider, specifically Firestore as the real-time database and Firebase Authentication for user authentication. The application aims to provide users with a convenient and efficient solution for tracking and managing their expenses. The system analysis phase involved defining the software requirements, applying agile software engineering paradigms, and creating various diagrams to understand the system's structure and behaviour. The system design incorporated modularization, data integrity measures, database design, and user interface design principles to ensure scalability, maintainability, and usability. Thorough testing techniques and strategies were employed, including unit testing, integration testing, functional testing, and performance testing, to ensure the reliability and functionality of the application. Debugging and code improvement practices were applied to enhance code quality and maintainability. To ensure system security, the Expense Tracker System implemented database and data security measures, such as encryption and access control mechanisms. User profiles and access rights were created using Firebase Authentication. The application generates informative reports to provide users with insights into their expenses, and sample report layouts were provided. The recommendations for the Expense Tracker System include expanding platform compatibility, enhancing security measures, integrating with additional services, optimizing performance, considering localization and internationalization, and allocating resources for ongoing maintenance and support. By implementing these recommendations, the Expense Tracker System can further enhance its functionality, usability, and security, meeting the evolving needs of users and ensuring its competitiveness in the market.

TABLE OF CONTENT

INTRODUCTION

The Expense Tracker System is an innovative Android application designed to provide users with a convenient and user-friendly solution for managing their expenses effectively. With the increasing complexity of modern financial transactions and the need for individuals to maintain a clear understanding of their spending habits, this application offers a comprehensive set of features to track, categorize, and analyze expenses on the go.

The application leverages the capabilities of Android devices to provide a seamless experience for users, allowing them to effortlessly record their expenses and monitor their financial activities. Users can easily input details of their expenses, including amount, date, category, and additional notes, providing a comprehensive record of their financial transactions. The intuitive user interface ensures a smooth user experience, with clear navigation and easy-to-access functionalities.

Key features of the Expense Tracker System include personalized expense categories, allowing users to create custom categories that align with their spending patterns. This flexibility enables users to accurately categorize their expenses, facilitating insightful data analysis and budget planning. Additionally, the application provides visual representations of spending patterns, offering graphical representations and detailed reports that highlight areas of expenditure.

Furthermore, the Expense Tracker System offers a range of financial management tools, such as budget setting and expense notifications. Users can set personalized budgets and receive timely notifications to stay informed about their spending limits and avoid overspending. This proactive approach to financial management empowers users to make informed decisions, save money, and maintain better control over their financial health.

Data security is a top priority in the Expense Tracker System. The application implements robust encryption techniques to protect sensitive financial information, ensuring the privacy and confidentiality of user data. Regular backups and synchronization options across multiple devices provide data redundancy and accessibility.

The Expense Tracker System represents a valuable tool for individuals, families, and small businesses, enabling them to achieve financial stability and make informed financial decisions. By leveraging the power and convenience of Android devices, this application delivers an efficient and user-friendly solution for tracking and managing expenses.

SYSTEM ANALYSIS

1. Identification of Need:

The Expense Tracker System is being developed to address the need for an efficient and user-friendly solution for tracking and managing expenses. Many individuals and organizations face challenges in keeping track of their expenses, leading to financial inefficiencies and difficulties in budgeting. The Expense Tracker System aims to provide a centralized platform that simplifies expense management, enhances financial control, and enables informed decision-making.

2. Preliminary Investigation:

During the preliminary investigation, the current expense management practices and challenges were identified. Interviews, surveys, and discussions with potential users were conducted to gather requirements and understand their pain points. The investigation revealed that manual expense tracking methods were time-consuming, error-prone, and lacked comprehensive reporting capabilities. The need for an automated system that streamlines expense tracking and provides real-time insights became evident.

3. Feasibility Study:

A feasibility study was conducted to assess the viability of developing the Expense Tracker System. The study analyzed technical, economic, operational, and schedule feasibility. It was determined that the system is technically feasible, considering the availability of suitable technologies and resources. Economically, the system is justified as it offers potential cost savings, improved financial management, and increased productivity. The operational feasibility was established by assessing the system's compatibility with existing processes and infrastructure. Lastly, the project was deemed schedule feasible based on resource availability and estimated development timelines.

4. Project Planning:

i. Project Scheduling:

The project scheduling phase involved breaking down the development process into manageable tasks and establishing a timeline for their completion. A Gantt chart was created to illustrate the project's schedule, milestones, and dependencies. The development tasks included requirements gathering, system design, database creation, user interface development, backend implementation, testing, and deployment. Each task was allocated a specific duration, and their interdependencies were carefully identified and accounted for in the schedule.

ii. Resource Planning:

A comprehensive resource plan was prepared, considering the human resources, hardware, software, and other required resources for the project. The team composition, roles, and responsibilities were defined. The hardware and software infrastructure necessary for development and deployment were identified, ensuring compatibility and scalability.

iii. Risk Assessment:

A risk assessment was performed to identify potential risks that could impact the project's success. Risks such as technology limitations, scope creep, resource constraints, and external dependencies were identified. Strategies to mitigate and manage these risks were devised, including regular progress monitoring, effective communication, contingency planning, and agile development methodologies.

iv. Budgeting:

A budget was developed to estimate the project's financial requirements. It included costs for human resources, hardware, software licenses, infrastructure, training, and contingency. The budget was aligned with the project's objectives and constraints, and it accounted for potential variations and unforeseen expenses.

v. **<u>Quality Assurance:</u>**

Quality assurance processes were defined to ensure that the Expense Tracker System meets the highest standards. Quality checkpoints, code reviews, and comprehensive testing procedures were implemented at various stages of development. User feedback and acceptance testing were incorporated to validate the system's functionality, usability, and performance.

vi. **<u>Documentation:</u>**

A documentation plan was formulated to capture the system's specifications, design, development guidelines, user manuals, and troubleshooting documentation. The plan aimed to facilitate future system maintenance, updates, and knowledge transfer.

5. **<u>Software Requirement Specifications (SRS):</u>**

The Expense Tracker System is an Android application designed to help users track and manage their expenses efficiently. The software requirements for this application are as follows:

i. **<u>Compatibility:</u>** The application should be compatible with Android devices running on version 5.0 (Lollipop) and above.

ii. **<u>User Registration:</u>** The system should allow users to register an account with a unique username and password.

iii. **<u>Expense Tracking</u>:** Users should be able to add, edit, and delete expenses. Each expense entry should include details such as date, category, description, and amount.

iv. **<u>Categorization:</u>** The system should provide predefined categories (e.g., food, transportation, utilities) for users to assign to their expenses. Users should also have the ability to create custom categories.

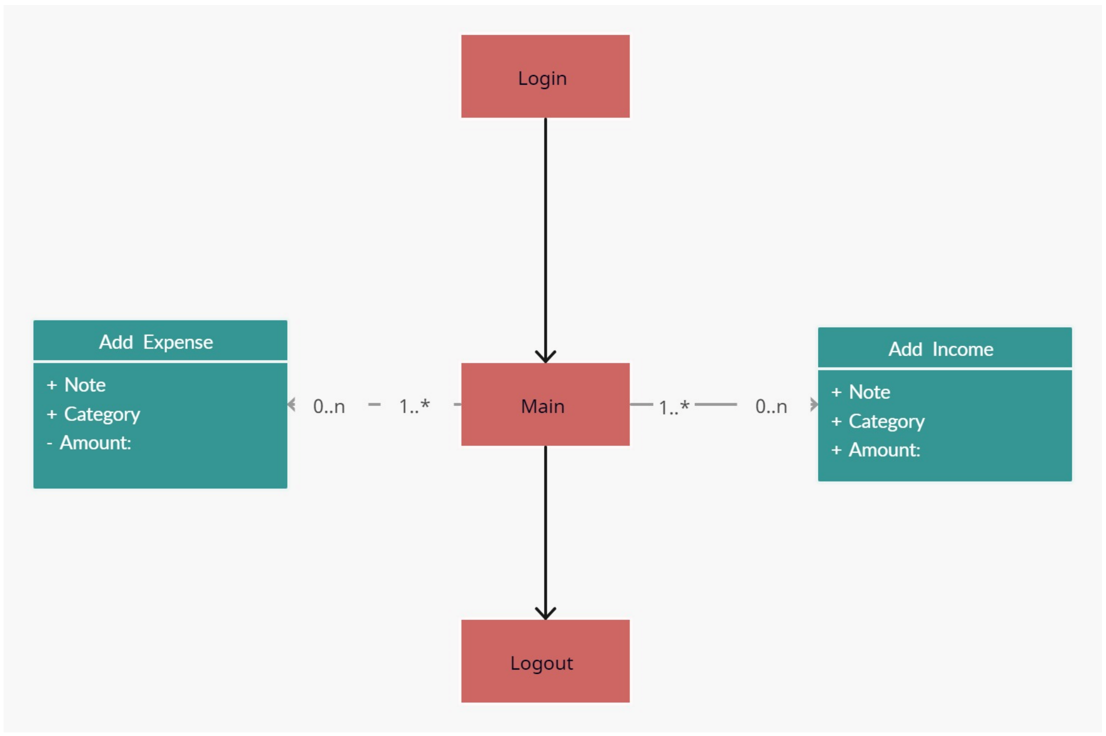
v. **<u>Budget Management:</u>** Users should be able to set monthly or weekly budgets for different expense categories. The system should notify users when they exceed their budget limits.

vi. **<u>Data Backup and Synchronization:</u>** The application should support data backup and synchronization across multiple devices to ensure that users' expense data is always accessible and up to date.

vii. **<u>Reporting and Analysis:</u>** The system should generate reports and visualizations to help users analyze their spending patterns and identify areas for improvement.

viii. **<u>User Interface:</u>** The application should have an intuitive and user-friendly interface, with easy navigation and responsive design.

6. Control Flow diagram:



SYSTEM DESIGN

1. Modularization Details:

The Expense Tracker System is designed using a modular approach to enhance maintainability, scalability, and code reusability. The application can be divided into the following modules:

i. **User Management Module:** Handles user registration, login, and authentication functionalities.

ii. **Expense Tracking Module:** Manages the creation, modification, and deletion of expense entries.

iii. **Category Management Module**: Deals with the management of predefined and custom expense categories.

iv. **Budget Management Module:** Allows users to set budget limits for different expense categories.

v. **Reporting Module:** Generates reports and visualizations based on expense data.

By dividing the application into these modules, it becomes easier to develop and maintain each module separately, facilitating parallel development and efficient bug fixing.

2. Data Integrity and Constraints:

To ensure data integrity and maintain consistency, the Expense Tracker System incorporates the following constraints:

i. **Unique Constraints:** Each user account is associated with a unique username to avoid duplication. Similarly, each expense entry is assigned a unique identifier.

ii. **Foreign Key Constraints:** Expense entries are linked to their respective users and categories through foreign key references, ensuring data consistency and referential integrity.

iii. **Validation Constraints:** The system enforces validation rules on input data, such as checking for valid dates, non-negative amounts, and valid category selections.

These constraints help maintain the integrity of the data stored in the application's database.

3. Database Design/Procedural Design/Object-Oriented Design:

Database Design: The Expense Tracker System utilizes a relational database to store user information, expense details, category data, and budget limits. The database schema is designed with appropriate tables, columns, and relationships to efficiently store and retrieve data. Normalization techniques are applied to eliminate data redundancy and improve database performance.

Procedural Design: The application follows a procedural design approach for implementing business logic and system operations. Each module consists of functions or methods responsible for specific tasks, such as user registration, expense creation, budget management, and reporting. The procedural design allows for better code organization, readability, and maintainability.

Object-Oriented Design: Object-oriented design principles are employed to model entities and their behavior. Objects are created to represent entities such as users, expenses, categories, and budgets. These objects encapsulate data and methods, providing modularity, reusability, and flexibility. Inheritance, polymorphism, and encapsulation are utilized to achieve a robust and extensible design.

4. User Interface Design:

The User Interface (UI) design of the Expense Tracker System focuses on delivering a seamless and intuitive user experience. The following design considerations are taken into account:

i. **Responsive Design:** The UI is designed to adapt to different screen sizes and orientations, providing optimal user experience across various Android devices.

ii. **Intuitive Navigation:** The application employs a well-structured navigation hierarchy with intuitive menus, buttons, and icons, making it easy for users to navigate between different functionalities.

iii. **User-Friendly Forms:** Forms for adding expenses, setting budgets, and managing categories are designed to be user-friendly, with clear labels, input validation, and error handling.

iv. **Visual Feedback:** The system provides visual feedback, such as success messages, error alerts, and progress indicators, to keep users informed about the status of their actions.

v. **Consistent Theme and Branding:** A consistent color scheme, typography, and branding elements are applied throughout the UI to enhance the overall aesthetic appeal and brand recognition.

The UI design focuses on simplicity, ease of use, and visual appeal, ensuring a pleasant and efficient user experience while interacting with the Expense Tracker System.

Overall, the Expense Tracker System's system design encompasses modularization for better maintainability, data integrity constraints for consistency, appropriate database design, a combination of procedural and object-oriented design principles, and a user interface design that prioritizes usability.

CODING

1. Main Activity:

```
package com.example.expensetracker;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.Toast;
import com.example.expensetracker.databinding.ActivityMainBinding;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;


public class MainActivity extends AppCompatActivity {
    ActivityMainBinding binding;
    FirebaseAuth firebaseAuth;
//    ProgressBar pb =  findViewById(R.id.progress_bar);
    private long pressedTime;

      @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding=ActivityMainBinding.inflate(getLayoutInflater());

        //for disabling dark mode
```

```
//for progressbar
//
        setContentView(binding.getRoot());
        firebaseAuth= FirebaseAuth.getInstance();
 //for hiding actionbar
          if (getSupportActionBar() != null) {
              getSupportActionBar().hide();
          }

 //to go to sign-up page
        binding.gotosignupscreen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MainActivity.this,SignUpActivity.class);
                try{
                    startActivity(intent);
                }catch(Exception e){


                }
            }
        });

 //for login button
          binding.loginButton.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {
                  String email=binding.emailLogin.getText().toString().trim();
                  String password =binding.passwordLogin.getText().toString().trim();
                  if(email.isEmpty()||password.isEmpty()){
                      Toast.makeText(MainActivity.this,"Error: This field/s cannot be
empty",Toast.LENGTH_SHORT).show();
                      return;
                  }
                firebaseAuth.signInWithEmailAndPassword(email,password)
                        .addOnSuccessListener(new OnSuccessListener<AuthResult>() {

                            @Override
                            public void onSuccess(AuthResult authResult) {
                                try{

              Toast.makeText(MainActivity.this,"Success!",Toast.LENGTH_SHORT).show();
                                    startActivity(new Intent(MainActivity.this,DashboardActivity.class));

                                }catch(Exception e){

                                }
                            }
                        })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {

Toast.makeText(MainActivity.this,e.getMessage(),Toast.LENGTH_SHORT).show();
                            }
                        });
            }
        });
```

//if already logged directly to dashboard

```
    }
```

//Double press to exit

```
    public void onBackPressed(){
        if (pressedTime + 2000 > System.currentTimeMillis()) {
            super.onBackPressed();
            finish();
        } else {
            Toast.makeText(getBaseContext(), "Press back again to exit",
Toast.LENGTH_SHORT).show();
        }
        pressedTime = System.currentTimeMillis();
    }
}
```

2. <u>DashBoard Activity:</u>

```
package com.example.expensetracker;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;
import android.widget.Toolbar;

import com.example.expensetracker.databinding.ActivityDashboardBinding;
import com.github.mikephil.charting.charts.PieChart;
import com.github.mikephil.charting.components.Description;
import com.github.mikephil.charting.data.PieData;
import com.github.mikephil.charting.data.PieDataSet;
import com.github.mikephil.charting.data.PieEntry;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;

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

public class DashboardActivity extends AppCompatActivity implements OnItemsClick{
    ActivityDashboardBinding binding;
    private ExpenseAdapter expenseAdapter;
    //Intent intent ;
    private long income,expense=0;
    private long pressedTime;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityDashboardBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        expenseAdapter = new ExpenseAdapter(this,this);
        binding.recycler.setAdapter(expenseAdapter);
        binding.recycler.setLayoutManager(new LinearLayoutManager(this));

// intent = new Intent(DashboardActivity.this,AddExpenseActivity.class);

        //for add button
        binding.add.setOnClickListener(new View.OnClickListener() {
            private int count =0;
            @Override
            public void onClick(View view) {
```

```
            count++;
            if(count%2==0){
                findViewById(R.id.add_expense).setVisibility(View.GONE);
                findViewById(R.id.add_income).setVisibility(View.GONE);
            }else {
                findViewById(R.id.add_expense).setVisibility(view.VISIBLE);
                findViewById(R.id.add_income).setVisibility(View.VISIBLE);


            }
        }
    });

    //When Add income is pressed
    binding.addIncome.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(DashboardActivity.this,AddExpenseActivity.class);
            intent.putExtra("type","Income");
            startActivity(intent);
        }
    });

    //When Add Expenses is pressed
    binding.addExpense.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(DashboardActivity.this,AddExpenseActivity.class);
            intent.putExtra("type","Expense");
            startActivity(intent);
        }
    });
}
```

```
    //For menu options
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
       // menu.add("User");
        menu.add("Logout");
        return super.onCreateOptionsMenu(menu);
    }

    //for clicking on menu options
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if(item.getTitle().equals("Logout"));
            getLogout();
        return super.onOptionsItemSelected(item);
    }

    private void getLogout() {
        FirebaseAuth.getInstance().signOut();
        Intent intent = new Intent(DashboardActivity.this,MainActivity.class);
        startActivity(intent);
    }

    //function for double press to exit
    public void onBackPressed() {

        if (pressedTime + 2000 > System.currentTimeMillis()) {
            super.onBackPressed();
            finish();
        } else {
            Toast.makeText(getBaseContext(), "Press back again to exit",
Toast.LENGTH_SHORT).show();
        }
        pressedTime = System.currentTimeMillis();
    }

    @Override
    protected void onResume() {
        super.onResume();
        income=expense=0;
        getData();
    }
```

//For Fetching the data from Firebase

```
    private void getData() {
        FirebaseFirestore
                .getInstance()
                .collection("expenses")
                .whereEqualTo("uid",FirebaseAuth.getInstance().getUid())
                .get()
                .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
                    @Override
                    public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                        expenseAdapter.clear();
                        List<DocumentSnapshot> dsList  =queryDocumentSnapshots.getDocuments();
                        for(DocumentSnapshot ds:dsList){
                            ExpenseModel expenseModel=ds.toObject(ExpenseModel.class);
                            if(expenseModel.getType().equals("Income")){
                                income+=expenseModel.getAmount();
                            }else{
                                expense+=expenseModel.getAmount();
                            }
                            expenseAdapter.add(expenseModel);
                        }
                        setUpGraph();
                    }
                });
    }
```

//For Entry in PieChart

```
    private void setUpGraph() {
        List<PieEntry> pieEntryList = new ArrayList<>();
        List<Integer> colorsList = new ArrayList<>();
        if(income!=0){
            pieEntryList.add(new PieEntry(income,"Income"));
            colorsList.add(getResources().getColor(R.color.teal_700));
        }
        if(expense!=0){
            pieEntryList.add(new PieEntry(expense,"Expense"));
            colorsList.add(getResources().getColor(R.color.red));
        }
        String graph_details;
        if(income>expense){
            graph_details= "| Balance: " +(income-expense);
        }
        else{
            graph_details="| Expense Exceeded: "+(income-expense);
        }
```

//For PieChart Representation

```
        PieDataSet pieDataSet = new PieDataSet(pieEntryList,graph_details);
        pieDataSet.setColors(colorsList);
        pieDataSet.setValueTextColor(getResources().getColor(R.color.white));
        pieDataSet.setValueTextSize(17);

        PieData pieData=new PieData(pieDataSet);
        binding.pieChart.setData(pieData);
        binding.pieChart.invalidate();
        binding.pieChart.getDescription().setEnabled(false);



    }

    @Override
    public void onClick(ExpenseModel expenseModel) {
        Intent intent = new Intent(DashboardActivity.this,AddExpenseActivity.class);
        intent.putExtra("model",expenseModel);
        startActivity(intent);
    }
}
```

3. <u>**Time and Date Maintaining Activity:**</u>

```
public class ExpenseAdapter extends RecyclerView.Adapter<ExpenseAdapter.MyViewHolder> {
    private Context context;
    private OnItemsClick onItemsClick;
    private List<ExpenseModel> expenseModelList;

    public ExpenseAdapter(Context context,OnItemsClick onItemsClick){
        this.context=context;
        expenseModelList = new ArrayList<>();
        this.onItemsClick=onItemsClick;
    }
    public void add(ExpenseModel expenseModel){
        expenseModelList.add(expenseModel);
        notifyDataSetChanged();
    }
    public void clear(){
        expenseModelList.clear();
        notifyDataSetChanged();
    }
    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.expense_row,parent,false);
        return new MyViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        ExpenseModel expenseModel=expenseModelList.get(position);

        //for time

       holder.date.setText(String.valueOf(getTimeDate(expenseModel.getTime())));
        holder.note.setText(expenseModel.getNote());
        holder.category.setText(expenseModel.getCategory());
        holder.amount.setText(String.valueOf(expenseModel.getAmount()));
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onItemsClick.onClick(expenseModel);
            }
        });
    }

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

    public class MyViewHolder extends RecyclerView.ViewHolder{
        private TextView note,category,amount,date;
        public MyViewHolder(@NotNull View itemView){
            super(itemView);
            date=itemView.findViewById(R.id.date);
            note=itemView.findViewById(R.id.note);
            category=itemView.findViewById(R.id.category);
            amount=itemView.findViewById(R.id.amount);
        }
    }

    //For Date
    public static String getTimeDate(long timestamp){
        try{
            Date date = (new Date(timestamp));
            //new trial
            SimpleDateFormat sdf= new SimpleDateFormat("EEE, MMM d, yyyy HH:mm",
Locale.getDefault());
            return sdf.format(date);
        }catch(Exception e){
            return "date";
        }
    }
}
```

4. SignUp Activity:

```
package com.example.expensetracker;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import com.example.expensetracker.databinding.ActivityMainBinding;
import com.example.expensetracker.databinding.ActivitySignUpBinding;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class SignUpActivity extends AppCompatActivity {
    ActivitySignUpBinding binding;
    FirebaseAuth firebaseAuth;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivitySignUpBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        //for hiding action bar
        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }

        firebaseAuth = FirebaseAuth.getInstance();

        //for going to log in page
        binding.gottologin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(SignUpActivity.this,MainActivity.class);
                try{
                    startActivity(intent);
                }catch(Exception e){

                }
            }
        });

        //For creating user
        binding.buttonSignup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email=binding.emailSignup.getText().toString();
                String password = binding.passwordSignup.getText().toString();
                if(email.trim().length()<=0||password.trim().length()<=0){
                    return;
                }

firebaseAuth.createUserWithEmailAndPassword(email,password).addOnSuccessListener(new
OnSuccessListener<AuthResult>() {
                    @Override
                    public void onSuccess(AuthResult authResult) {
                        Toast.makeText(SignUpActivity.this,"User
Created",Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(SignUpActivity.this,MainActivity.class));
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {

Toast.makeText(SignUpActivity.this,e.getMessage(),Toast.LENGTH_SHORT).show();

                    }
                });
            }
        });
    }
    public void onBackPressed(){
            Intent intent = new Intent(SignUpActivity.this,MainActivity.class);
            startActivity(intent);
            super.onBackPressed();
            finish();

    }
}
```

5. Activity_Main XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:orientation="vertical"
    android:background="@drawable/loginbackground"
    >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="135dp"
        android:text="Expense Manager"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:layout_marginTop="20dp"
        android:textStyle="bold"
        android:textColor="@color/white"
        android:textAlignment="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="LogIn"
        android:textAllCaps="true"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/email_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:background="@drawable/edit_text_background"
        android:hint="Write your email here"
        android:padding="15dp"
        android:textAlignment="center" />

    <EditText
```

```
    android:id="@+id/password_login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:background="@drawable/edit_text_background"
    android:hint="Write your password"
    android:inputType="textPassword"
    android:padding="12dp"
    android:textAlignment="center" />

  <TextView
    android:id="@+id/forgotpassword"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:layout_marginTop="8sp"
    android:gravity="center"
    android:padding="10dp"
    android:text="Forgot Password?"
    android:textSize="14dp"
    android:textStyle="bold" />



  <ProgressBar
    android:background="@drawable/progressbar_style"
    android:id="@+id/progress_bar"
    android:padding="8dp"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:elevation="10dp"
    android:layout_gravity="center"
    android:visibility="invisible"/>
```

```
<!--    <RelativeLayout-->
<!--        android:layout_width="wrap_content"-->
<!--        android:layout_height="wrap_content"-->
<!--        android:layout_gravity="center"-->
<!--        android:background="@drawable/progressbar_style">-->
<!--        <ProgressBar-->
<!--            android:id="@+id/progress_bar"-->
<!--            android:padding="2dp"-->
<!--            android:layout_width="40dp"-->
<!--            android:layout_height="40dp"-->
<!--            android:elevation="10dp"-->
<!--            android:visibility="gone"/>-->

<!--    </RelativeLayout>-->
    <Button
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="58dp"
        android:layout_gravity="center"
        android:layout_marginTop="-30dp"
        android:backgroundTint="@color/white"
        android:text="Login"
        android:textColor="@color/black" />

    <TextView
        android:id="@+id/gotosignupscreen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="20dp"
        android:padding="20dp"
        android:text="New User? Register here!"
        android:textColor="@color/white"
        android:textSize="20sp" />

</LinearLayout>
```

6. <u>DashBoard Activity XML:</u>

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DashboardActivity"
    android:padding="10dp"
    >

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="300dp"
            android:layout_margin="5dp">
            <com.github.mikephil.charting.charts.PieChart
                android:id="@+id/pieChart"
                android:layout_width="match_parent"
                android:layout_height="match_parent"/>
        </androidx.cardview.widget.CardView>

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recycler"
            android:layout_width="match_parent"
            android:layout_height="350dp"
            android:layout_marginTop="10dp" />
    </LinearLayout>


    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="64dp"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal">

        <TextView

            android:id="@+id/add_income"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/add"
            android:layout_alignParentBottom="true"
            android:layout_marginLeft="-130dp"
            android:layout_marginRight="-142dp"
            android:layout_marginBottom="14dp"
            android:layout_weight="1"
            android:background="@drawable/rounded_corners"
            android:gravity="center"
            android:padding="15dp"
            android:text="Add Income"
            android:textColor="@color/white"
            android:textSize="16sp"
            android:visibility="gone" />

        <TextView
            android:id="@+id/add_expense"
            android:layout_width="wrap_content"
```

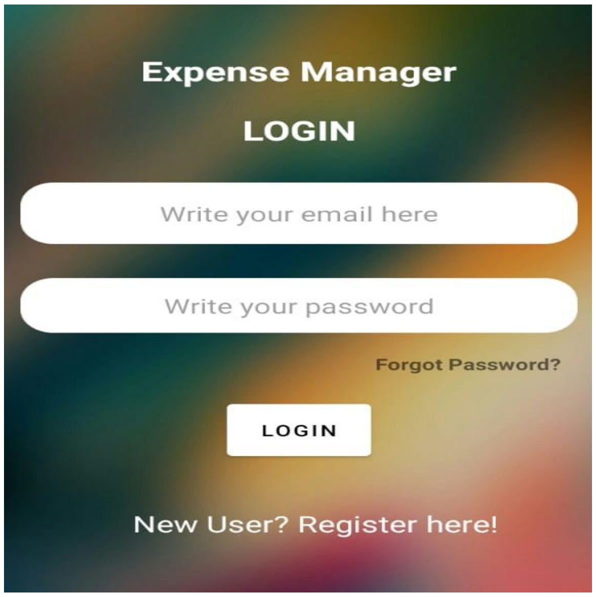
```
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginLeft="10dp"
            android:layout_marginBottom="14dp"
            android:layout_weight="1"
            android:background="@drawable/rounded_corners"
            android:gravity="center"
            android:padding="15dp"
            android:text="Add Expense"
            android:textColor="@color/white"
            android:textSize="16sp"
            android:visibility="gone" />

        <ImageView
            android:id="@+id/add"
            android:layout_width="77dp"
            android:layout_height="67dp"
            android:layout_alignRight="@+id/add_expense"
            android:layout_alignParentRight="true"
            android:layout_alignParentBottom="true"
            android:layout_gravity="center"
            android:layout_marginRight="0dp"
            android:layout_marginBottom="0dp"
            android:baselineAlignBottom="true"
            android:src="@drawable/add_view" />
    </RelativeLayout>

</RelativeLayout>
```

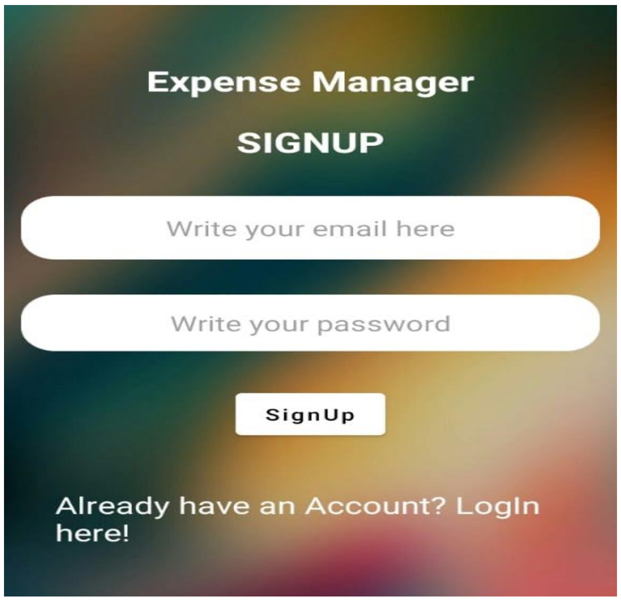
TESTING

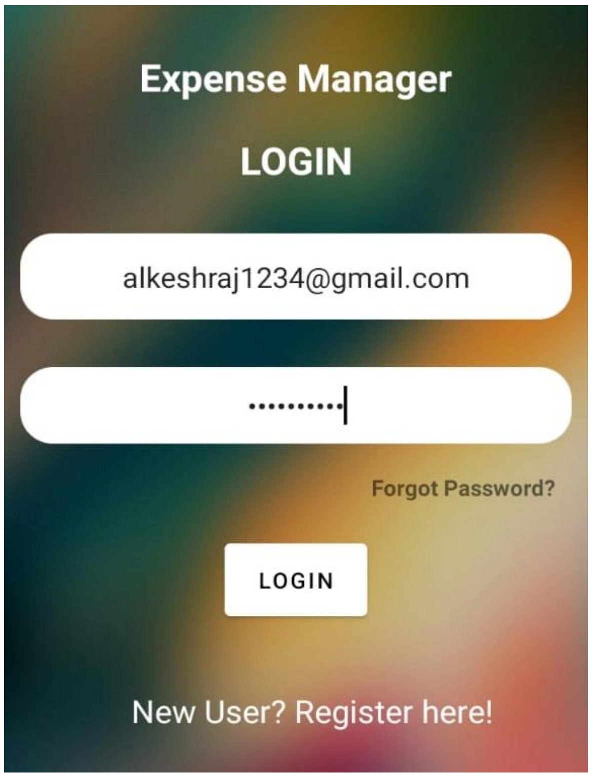
1. Some Screenshots of our Application during Testing

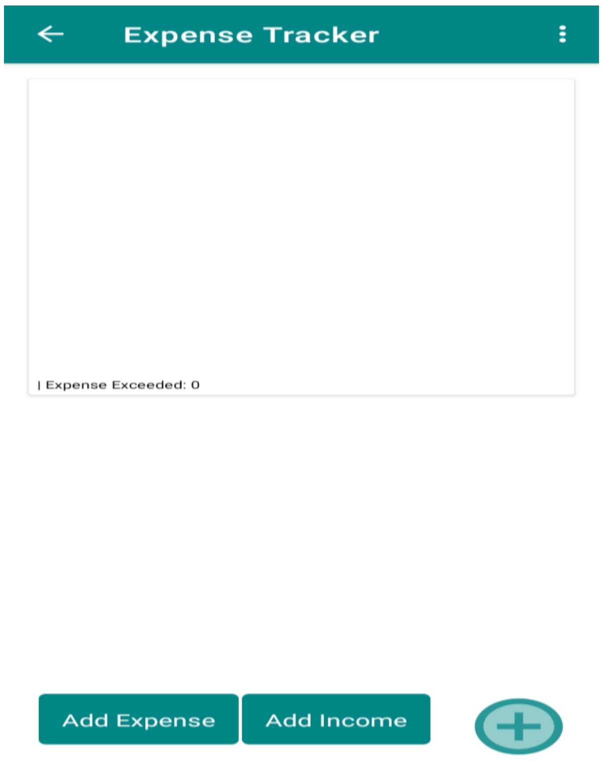
Step 1. : First of all, we open the application.

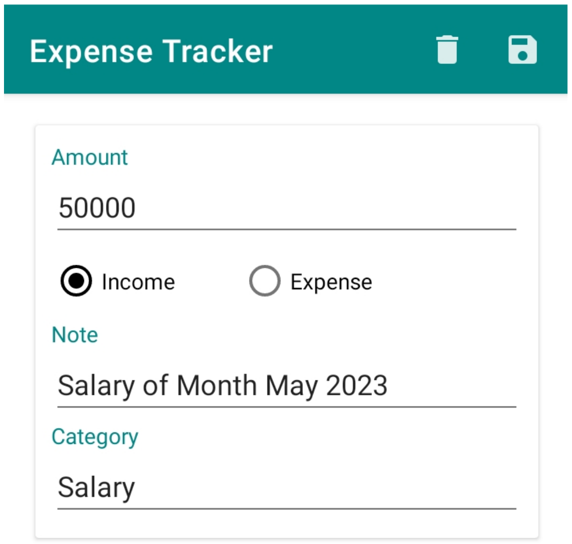
Step 2. : Then the following interfaces come up –











Amount

50000

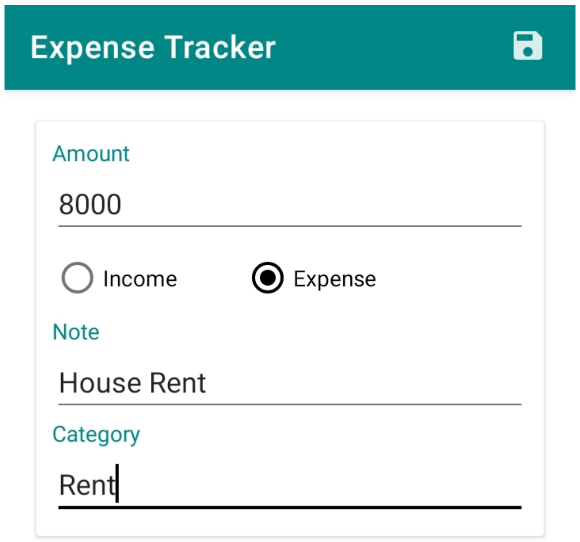
[✓ Income] Expense

Note

Salary of Month May 2023

Category

Salary



Amount

8000

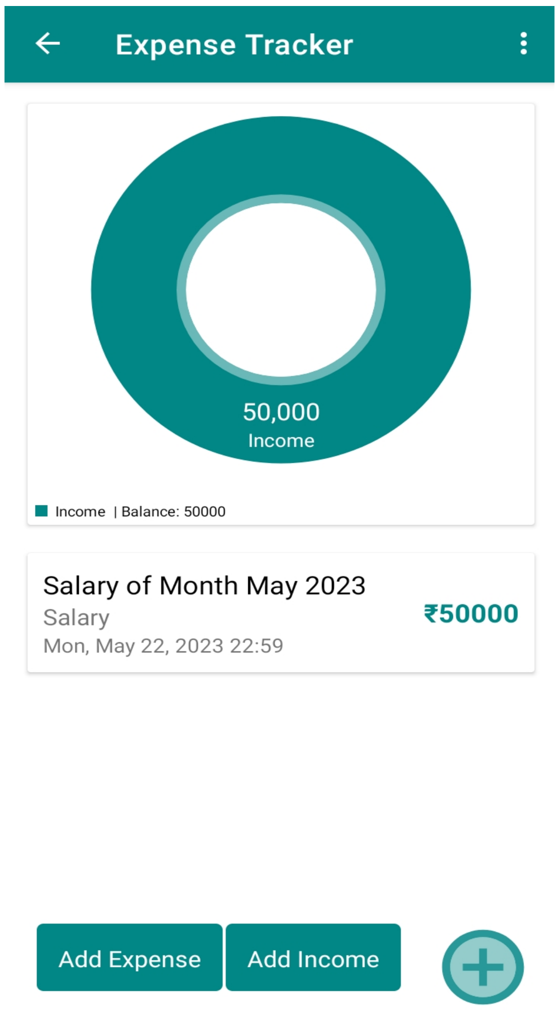
Income [✓ Expense]

Note

House Rent

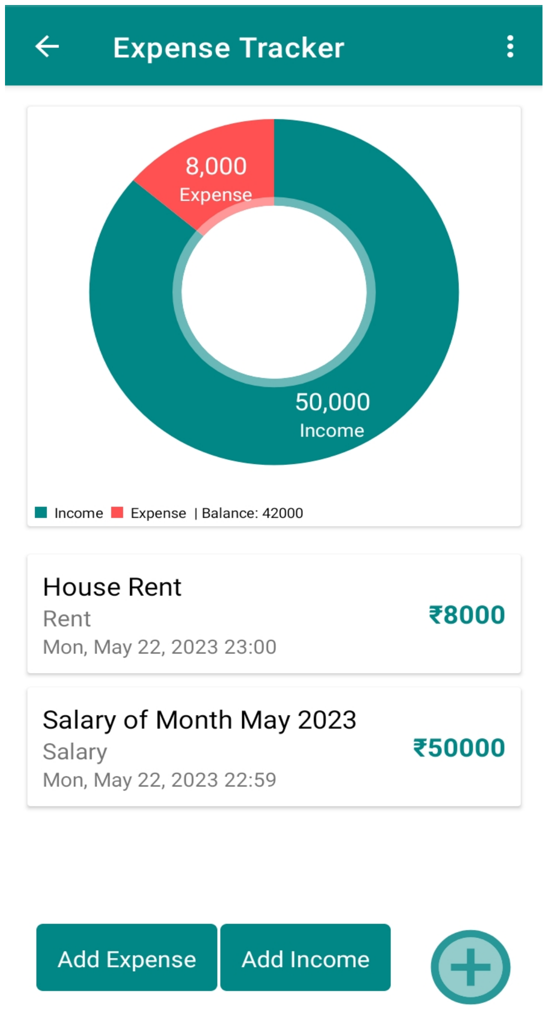
Category

Rent



🟩 Income | Balance: 50000

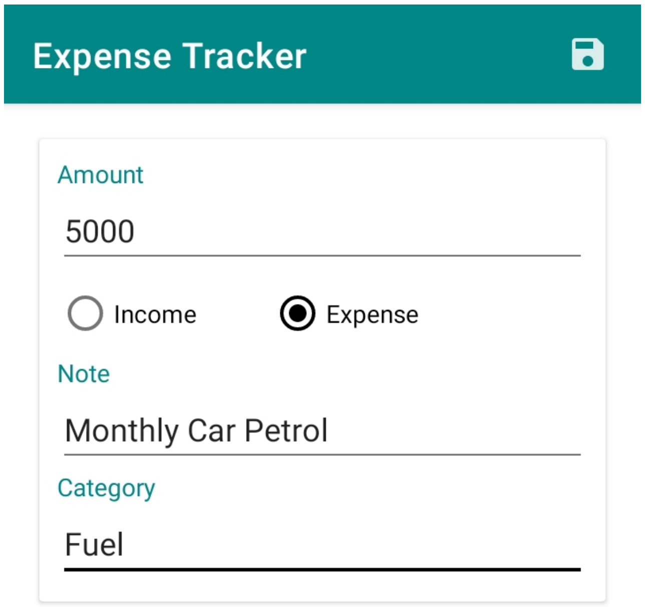
Salary of Month May 2023
Salary
Mon, May 22, 2023 22:59
₹50000



🟩 Income 🟥 Expense | Balance: 42000

House Rent
Rent
Mon, May 22, 2023 23:00
₹8000

Salary of Month May 2023
Salary
Mon, May 22, 2023 22:59
₹50000

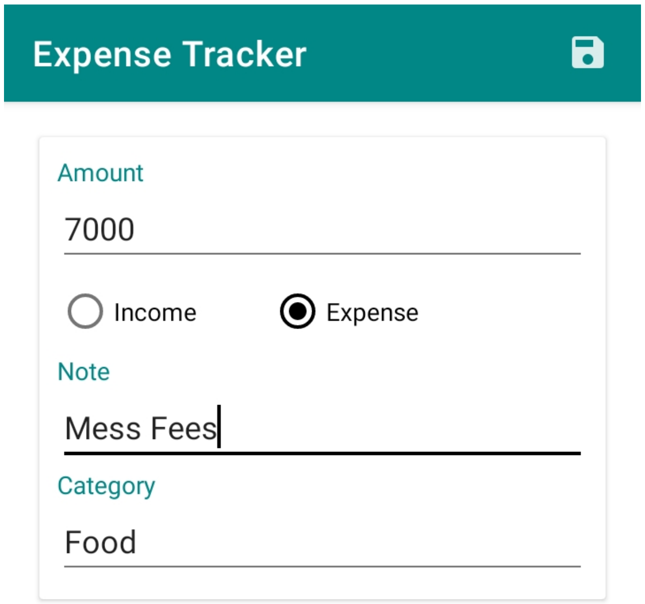

Expense Tracker

Amount
5000

Income [✓ Expense]

Note
Monthly Car Petrol

Category
Fuel

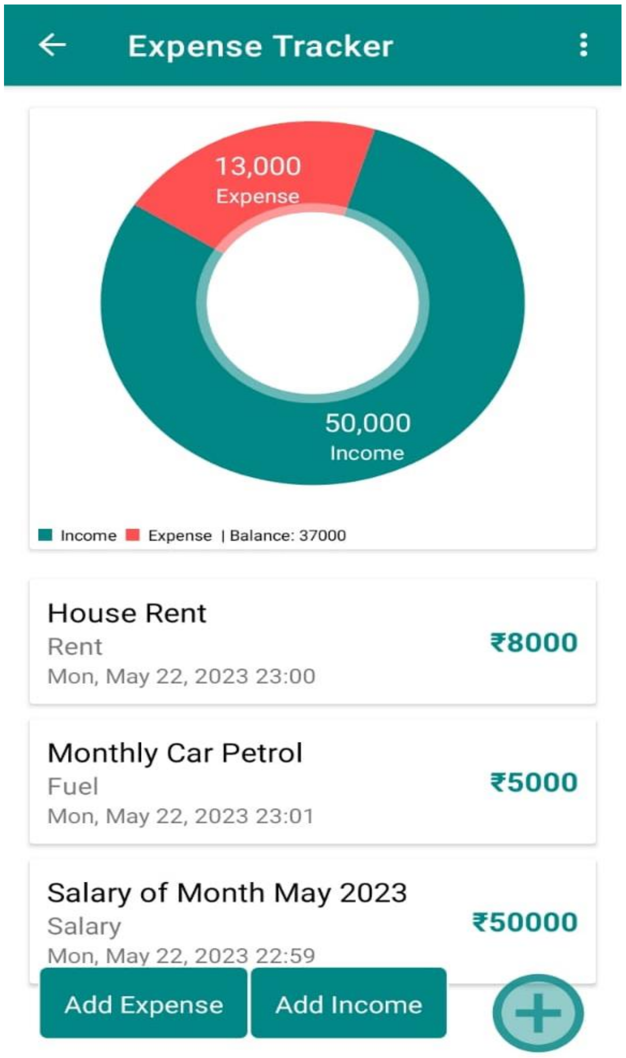

Expense Tracker

Amount
7000

Income [✓ Expense]

Note
Mess Fees

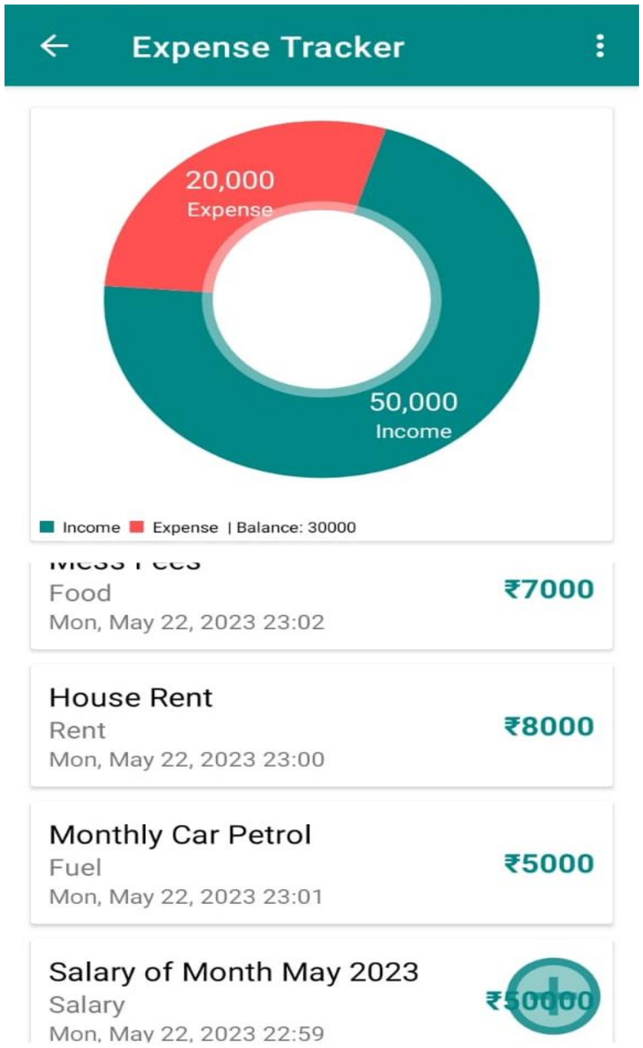
Category
Food


Expense Tracker

🟩 Income 🟥 Expense | Balance: 37000

House Rent Rent Mon, May 22, 2023 23:00	₹8000
Monthly Car Petrol Fuel Mon, May 22, 2023 23:01	₹5000
Salary of Month May 2023 Salary Mon, May 22, 2023 22:59	₹50000

Add Expense Add Income


Expense Tracker

🟩 Income 🟥 Expense | Balance: 30000

Food Mon, May 22, 2023 23:02	₹7000
House Rent Rent Mon, May 22, 2023 23:00	₹8000
Monthly Car Petrol Fuel Mon, May 22, 2023 23:01	₹5000
Salary of Month May 2023 Salary Mon, May 22, 2023 22:59	₹50000

ii. <u>Debugging and Code Improvement:</u>

During the testing process, debugging techniques are applied to identify and resolve defects or issues within the Expense Tracker System. Debugging involves analyzing error logs, examining code, and using debugging tools to locate and fix errors, ensuring the proper functioning of the application.
Code improvement is an ongoing process that aims to enhance the quality, readability, and maintainability of the codebase. This includes:

i. **Refactoring:** The code is optimized and restructured without changing its external behavior. This improves code clarity, eliminates code duplication, and enhances code maintainability.

ii. **Code Review:** Peer code reviews are conducted to ensure adherence to coding standards, identify potential issues, and provide suggestions for code improvement. This helps identify and correct any code-related issues early in the development process.

iii. **Performance Optimization:** The code is analyzed to identify performance bottlenecks, and optimizations are implemented to improve the application's speed and efficiency. This may involve optimizing database queries, reducing resource usage, or implementing caching mechanisms.

iv. **Error Handling and Exception Handling:** Proper error handling mechanisms are implemented to gracefully handle exceptions and prevent application crashes. Error logs are generated to aid in debugging and provide insights into potential issues.

By incorporating effective debugging techniques and continuously improving the codebase, the Expense Tracker System ensures a robust and reliable application.
Overall, testing techniques, strategies, and test case designs are employed to validate the Expense Tracker System's functionality, while debugging and code improvement practices help enhance the quality and stability of the application.

SYSTEM SECURITY MEASURES

The Expense Tracker System incorporates various security measures to ensure the protection of user data and maintain the integrity of the application. The following security measures are implemented:

i. Database/Data Security:

1. **Encryption:** User data, including passwords and sensitive information, is encrypted using strong encryption algorithms before storing them in the database. Encryption ensures that even if the database is compromised, the data remains secure and inaccessible to unauthorized users.

2. **Secure Database Connection:** The communication between the application and the database is secured using secure protocols, such as HTTPS or SSL/TLS. This encryption ensures that data transmitted between the application and the database remains confidential and cannot be intercepted or tampered with.

3. **Data Backup and Recovery:** Regular backups of the database are taken to prevent data loss in case of any unforeseen incidents. The backups are stored securely, following best practices, and can be used to restore the system to a previous state if necessary.

4. **Database Access Control:** Access to the database is restricted to authorized personnel only. Strong authentication mechanisms, such as username/password combinations or multi-factor authentication, are implemented to prevent unauthorized access to the database.

ii. Creation of User Profiles and Access Rights:

1. **User Authentication**: The Expense Tracker System employs a secure authentication mechanism to verify the identity of users. Users are required to provide valid credentials, such as a unique username and password, to access the system. This authentication process ensures that only authorized users can log in and access their expense data.

2. **User Access Rights**: Different access levels and permissions are assigned to user profiles

based on their roles and responsibilities within the system. This prevents unauthorized access to sensitive functionalities and ensures that users can only perform actions for which they have proper authorization.

3. **Password Policies:** The system enforces strong password policies, including password complexity requirements and regular password expiration. This ensures that users create and maintain secure passwords, reducing the risk of unauthorized access due to weak or compromised credentials.

4. **Session Management:** The system implements secure session management techniques to manage user sessions effectively. This includes using session tokens, enforcing session timeouts, and securely managing session data to prevent session hijacking or unauthorized session reuse.

Overall, the Expense Tracker System incorporates database and data security measures, such as encryption and secure connections, to protect user data. The creation of user profiles with appropriate access rights, along with strong authentication mechanisms and role-based access control, ensures that only authorized users can access and manipulate the system's functionalities. These security measures help safeguard user information and maintain the confidentiality and integrity of the Expense Tracker System.

COST ESTIMATION OF THE PROJECT

Cost estimation is an essential aspect of project planning and management. The Expense Tracker System's cost estimation takes into account various factors, including development resources, infrastructure, and ongoing maintenance. The following cost components are considered:

i. **Development Costs:** This includes the cost of development resources, such as software engineers, designers, and testers. The estimated effort required to complete the project, based on the scope and complexity, is multiplied by the appropriate hourly rates to determine the development cost.

ii. **Infrastructure Costs:** The infrastructure costs cover the hardware and software resources required to develop and deploy the application. This includes the cost of development tools, software licenses, servers, and hosting services.

iii. **Maintenance Costs:** Ongoing maintenance costs are estimated to ensure the system's smooth operation, bug fixes, and updates. These costs may include the hosting fees and the cost of supporting software.

iv. **Miscellaneous Costs:** Miscellaneous costs, such as training, documentation, and customer support, are also considered in the cost estimation.

REPORTS

The Expense Tracker System generates various reports to provide users with insights into their expenses and financial management. Sample layouts for the reports can include:

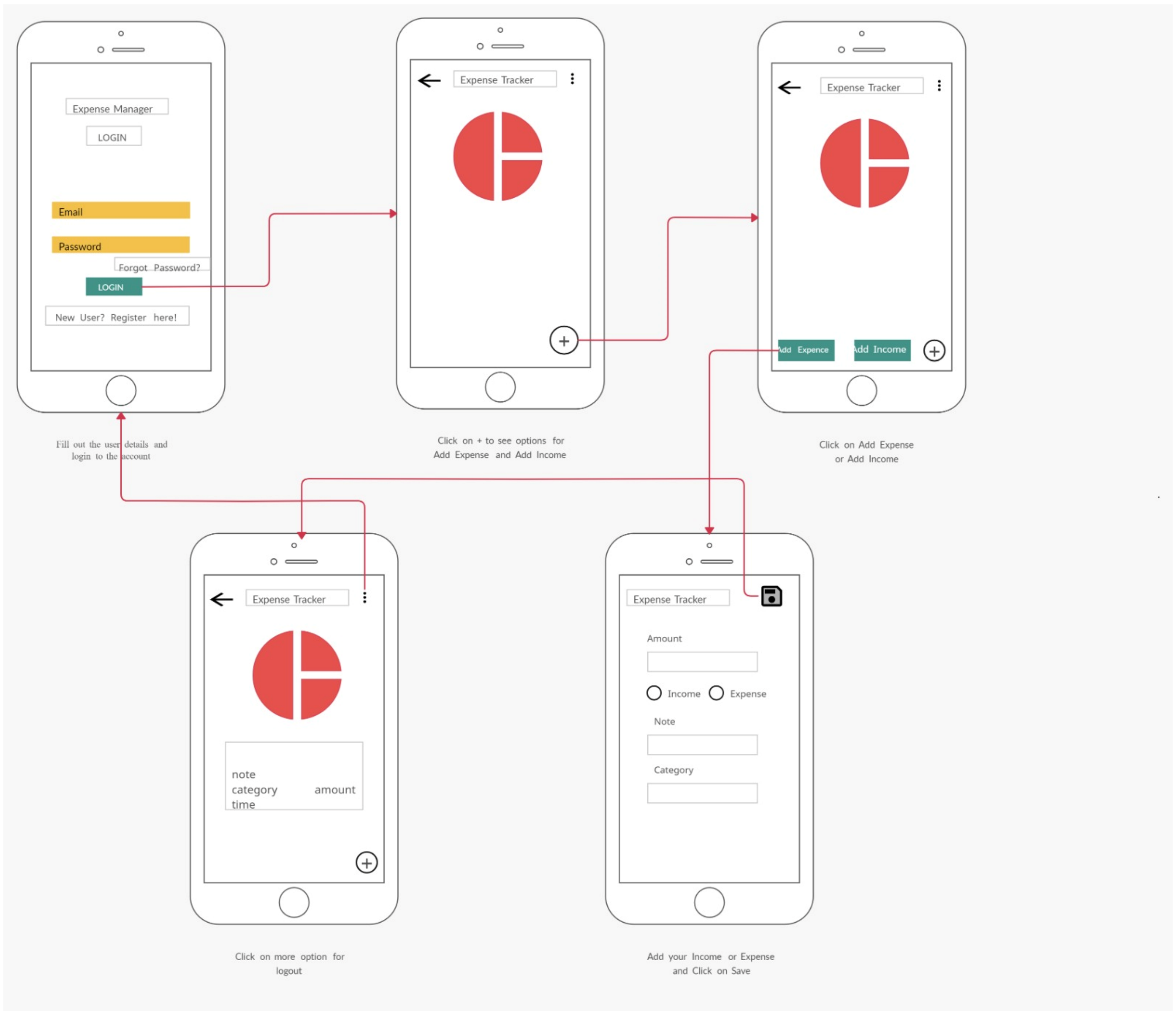
i. **Expense Summary Report:** This report provides an overview of the user's total expenses, categorized by different expense categories (e.g., food, transportation, utilities). It includes a pie chart to visualize the distribution of expenses.

ii. **Budget Analysis Report:** This report compares the user's actual expenses against their set budget limits. It highlights categories where the user has exceeded the budget and provides recommendations for better budget management.

iii. **Monthly Expense Trend Report:** This report shows the user's monthly expenses over a specific period, allowing them to identify spending patterns and trends.

iv. **Category-wise Expense Report:** This report provides a breakdown of expenses by category, allowing users to analyze their spending habits in detail. It may include a table or bar chart displaying the total expenses for each category.

SAMPLE LAYOUT



Expense Manager
LOGIN
Email
Password
Forgot Password?
LOGIN
New User? Register here!
Fill out the user details and login to the account
Expense Tracker
Click on + to see options for Add Expense and Add Income
Expense Tracker
Add Expence
Add Income
Click on Add Expense or Add Income
Expense Tracker
note
category
time
amount
Click on more option for logout
Expense Tracker
Amount
Income
Expense
Note
Category
Add your Income or Expense and Click on Save

CONCLUSION

The Expense Tracker System is an Android application developed using Android Studio and Java programming language, with Firebase as the backend service provider. The project successfully addresses the need for a convenient and efficient expense-tracking solution for users. Throughout the development process, various system analysis and design techniques were applied, including software requirement specifications, modularization, data integrity, database design, and user interface design.

The testing phase played a crucial role in ensuring the reliability and functionality of the Expense Tracker System. Testing techniques such as unit testing, integration testing, functional testing, and performance testing were employed to identify and resolve any defects. Debugging and code improvement practices were implemented to enhance the quality and maintainability of the codebase.

System security measures were implemented to protect user data and maintain data integrity. This included database and data security through encryption and access control mechanisms, as well as the creation of user profiles and access rights. Firebase Authentication was utilized for secure user authentication.

The cost estimation of the project considered various factors such as development costs, infrastructure costs, maintenance costs, and miscellaneous costs. This estimation provided an understanding of the financial aspects associated with the development and maintenance of the Expense Tracker System.

The Expense Tracker System generates informative reports, allowing users to gain insights into their expenses and make informed financial decisions. Sample layouts for reports were provided, including expense summary reports, budget analysis reports, and customizable reports with data visualizations.

REFERENCE

[1] Creating Gantt Charts. (2016). Retrieved 09 02, 2016, from Gantt Charts Web site: http://www.gantt.com/creating-gantt-charts.html

[2] Entity Relationship. (2016). Retrieved 09 02, 2016, from Creately Website: https://creately.com/app/#

[3] Fowler, M. (2004).UML Distilled Third Edition A Brief Guide To TheStandard Object Modelling Language.Addison-Wesely. Retrieved12 14, 2016

[4] Krutchten, P. (2000).The Rational Unified Process An IntroductionSecond Edition.Addison-Wesely. Retrieved 12 13, 2016

[5] Larman, C. (2008).Applying UML and Patterns. Pearson Education, Inc.Retrieved 08 28, 2016

[6] UML-Diagram. (2016). Retrieved 12 16, 2016, from UML-DiagramWebsite: http://www.uml-diagrams.org/class-diagrams-examples.html

[7] MP Android chart created by Philipp Jahoda .

Website : https://github.com/PhilJay/MPAndroidChart

[8] Firebase basics learning. Website : https://firebase.google.com/docs

[9] Android Studio Learning. Website: https://developer.android.com/docs

[10] Geeks For Geeks. Website: https://www.geeksforgeeks.org/android-projects-from-basic-to-advanced-level/

[11] Stack overflow: https://stackoverflow.com/questions/tagged/android-studio

[12] YouTube Video link : https://www.youtube.com/watch?v=qPlQrjDe-yE&pp=ygUXbG9naW4gaW4gYW5kcmlvZCBzdHVkaW8%3D