Finance App

Ethan Gandiboyina, Alex Huet

December 20, 2024

1 Introduction

This report describes the functionality and structure of an Android application designed to manage user transactions. The app allows users to add, filter, and graph transaction data. It uses Room for database management and MPAndroidChart for visualizing transaction data. The app provides features for user registration and login, transaction entry, displaying graphs of transaction data, and database management. The app uses several activities to manage different functionalities. The activities are used in the following order:

1. **LoginActivity**: The first activity where users log in to their accounts.
2. **RegisterActivity**: Allows users to create new accounts.
3. **MainActivity**: Displays the list of transactions and allows users to add new transactions. It also provides options to filter the transaction history.
4. **AddTransactionActivity**: Displays a form where users can enter new transaction details (amount, category, date, type).
5. **TransactionHistory**: Displays a list of all transactions. It allows the user to filter by dates.
6. **StatisticsActivity**: Shows a pie chart to visualize income versus expenses.
7. **FilterActivity**: Allows users to filter transactions by date or date range.

2 Transaction Handling in MainActivity

The main functionality of the app resides in `MainActivity`, where users can add transaction details. The user inputs the transaction amount, category, and type (either income or expense), and the data is stored in the database.

```java
addButton.setOnClickListener(v -> {
    // Get transaction details from input fields
    double amount =
        Double.parseDouble(amountEditText.getText().toString());
    String category = categoryEditText.getText().toString();
    // Determine if the transaction is income or expense
    String type = (typeGroup.getCheckedRadioButtonId() ==
        R.id.incomeRadioButton) ? "income" : "expense";
    long date = System.currentTimeMillis();  // Current timestamp for
       the transaction date

    // Create a new transaction entity
    TransactionEntity transaction = new TransactionEntity(amount,
       category, date, type);

    // Insert the transaction into the database using the repository
    repository.insertTransaction(transaction);
});
```

Listing 1: Adding a Transaction

The transaction is inserted into the database by the `TransactionRepository`, which abstracts all database interactions.

3 Graphing Transaction Data

Users can graph their transaction data, displaying income and expenses as a pie chart. This is done using the MPAndroidChart library.

```
PieChart chart = findViewById(R.id.pieChart);

// Prepare data entries for the Pie chart (total income and total
   expenses)
List<PieEntry> entries = new ArrayList<>();
entries.add(new PieEntry(totalIncome.floatValue(), "Income"));
entries.add(new PieEntry(totalExpense.floatValue(), "Expense"));

// Create a PieDataSet with the data and define colors for the segments
PieDataSet dataSet = new PieDataSet(entries, "Transaction Summary");
dataSet.setColors(Color.GREEN, Color.RED);

// Set the data to the chart and refresh it
PieData data = new PieData(dataSet);
chart.setData(data);
chart.invalidate(); // Refresh the chart to display updated data
```

Listing 2: Updating Pie Chart

This code generates a pie chart showing the breakdown of total income and total expenses. The chart is refreshed using `chart.invalidate()` after the data is updated.

4 Database Structure

The app uses Room to store transaction data, which is stored in a local SQLite database. The main database entity is `TransactionEntity`, which represents a transaction record.

```
@Entity(tableName = "transactions")
public class TransactionEntity {
    @PrimaryKey(autoGenerate = true)
    private int id;  // Unique identifier for each transaction
    private double amount;  // Amount of the transaction
    private String category;  // Category for the transaction (e.g.,
       Food, Salary)
    private long date;  // Date stored as a timestamp
    private String type;  // "income" or "expense"

    // Constructor, getters, and setters omitted for brevity
}
```

Listing 3: Transaction Entity

The app's database structure is accessed through a `TransactionDao` interface, which defines methods to interact with the transaction data, such as inserting and querying transactions.

```
@Dao
public interface TransactionDao {
    @Insert
    void insertTransaction(TransactionEntity transaction);  // Insert
        a new transaction

    @Query("SELECT * FROM transactions WHERE date = :date")
    List<TransactionEntity> getTransactionsByDate(String date);  //
        Get transactions by date

    @Query("SELECT SUM(amount) FROM transactions WHERE type =
        'income'")
    Double getTotalIncome();  // Get total income amount

    @Query("SELECT SUM(amount) FROM transactions WHERE type =
        'expense'")
    Double getTotalExpense();  // Get total expense amount
}
```

Listing 4: Transaction DAO

The functions in the `TransactionRepository` class provide methods for interacting with the database:

1. **'TransactionRepository(Context context)'** - Initializes the Room database and sets up the `TransactionDao`.
2. **'insertTransaction(TransactionEntity transaction)'** - Inserts a new transaction into the database in a background thread.
3. **'getTotalIncome()'** - Retrieves the total income amount from the database.
4. **'getTotalExpense()'** - Retrieves the total expense amount from the database.

5 Gradle Setup for External Libraries

To enable the graphing functionality and database management, the app requires external dependencies, such as MPAndroidChart for charting and Room for database management. The following dependencies are added to the `build.gradle` file:

```
dependencies {
    implementation 'androidx.room:room-runtime:2.4.3'  // Room for
        database management
    annotationProcessor 'androidx.room:room-compiler:2.4.3'  // Room
        annotation processor
    implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'  //
       MPAndroidChart for graphing
    implementation 'androidx.appcompat:appcompat:1.3.1'  // AppCompat
       for backward compatibility
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
       // ConstraintLayout for UI layout
    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'  //
       Lifecycle components for data binding
}
```

Listing 5: Gradle Dependencies

These dependencies are required to enable the charting and database features. **The app will NOT run without properly loading external files.**

6 Conclusion

This Android app provides a comprehensive solution for managing transactions. It allows users to log in, register new accounts, enter transaction data, and visualize their financial information through pie charts or as a list. By using "Room" for database management and MPAndroidChart for graphing, the app effectively helps users track their income and expenses. The app is structured around distinct activities, each responsible for different tasks such as user authentication, transaction management, and data visualization.

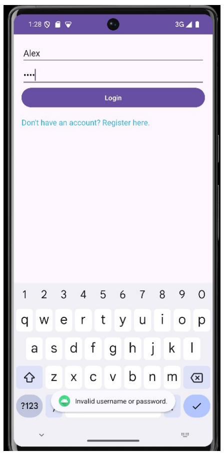
Alex
Login
Don't have an account? Register here.
Invalid username or password.

Figure 1: Login Activity

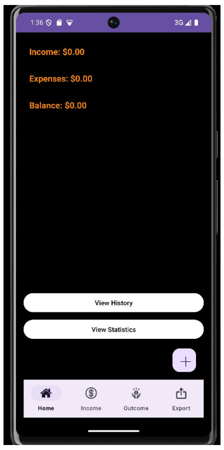

Figure 2: Main Activity

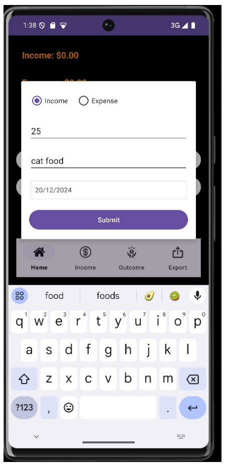

Figure 3: Add Transaction

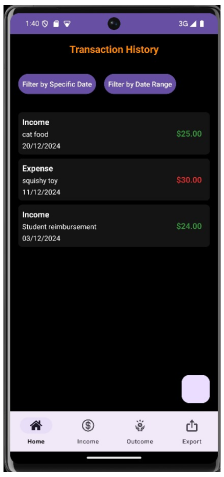

Figure 4: Transaction

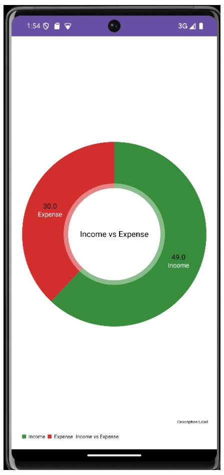

Figure 5: Graphing

Figure 6: Activities of the Finance App.

7 Contributions

- **Ethan**: Database management, Login/Registration Activities
- **Alex**: Graphing Activity, Transaction/Main page Logic
- **Both**:UI