Title of the Project

"Kards Keeper"

Name of the group members

Caleon, Philip Jeremiah

Gatchalian, Richmonde

Lirazan, Miguel

Malabanan, Raniel

Date of Submission

August 9, 2024

Abstract

Budgeting finances can be time-consuming and distract from other important tasks, leading to inefficiencies. To address this challenge, "Kards Keeper" was developed as a budgeting application to help users track their finances and make informed decisions about their spending. The application's primary objective is to provide an easy and intuitive way to manage expenses across various categories, including bills, food, transportation, and personal wants. The methodology involved an iterative development process, starting with a basic user interface (UI) and the integration of an SQLite database for secure transaction data storage. Database schemas were tested to ensure accurate data handling, and the UI was enhanced to display calculated totals and support transaction categorization. Rigorous testing, including debugging, was conducted throughout to refine the application's functionality and stability. The results demonstrate that "Kards Keeper" successfully meets its objectives. The application effectively tracks finances, categorizes expenses, and provides users with a clear overview of their spending habits. Users can efficiently manage their budgets, avoid overspending, and make informed financial decisions. The discussion highlights the program's simplicity and effectiveness while also identifying areas for future improvements, such as enhancing the user interface, adding visualizations like expense graphs, and incorporating security features like a login page with authentication. "Kards Keeper" was a successful budgeting tool that aligns with its initial objectives. It offers a practical solution for users seeking to manage their finances more efficiently, though there is potential for further enhancement to increase its value and usability.

Table of Contents

List of Abbreviations and Acronyms

- OOP: Object Oriented Programming
- UI: User Interface
- JDBC: Java Database Connectivity

I. Introduction

Saving money over time gives a person the freedom to spend on more things, but it can quickly become challenging to balance wants and needs while ensuring spending is worthwhile. "Kards Keeper" is a personal expense tracker designed to help users manage their finances with ease. It allows users to keep track of how much money is earned, budgeted, and spent, ensuring they don't lose sight of their spending in various categories. With this app, users can effectively manage their expenses by categorizing transactions, setting budgets, and monitoring their financial health over time. The app provides a clear overview of income and expenses, helping users make informed decisions and avoid overspending. By offering these features, "Kards Keeper" empowers users to take control of their finances, stay organized, and achieve their financial goals more efficiently.

II. Description of the Proposed System

The "Kards Keeper" app is designed to tackle the complexities of personal finance management. This program offers a streamlined approach to budgeting, enabling users to concentrate on their core responsibilities without the burden of intricate financial tasks. By presenting a straightforward snapshot of their financial status, users are empowered to make well-informed choices about their expenditures, ensuring their resources are directed towards achieving their financial objectives.

Thanks to its user-friendly interface, the application can be used by a diverse group of individuals, even those who are unfamiliar with budgeting tools. By utilizing categories such as Bills/Fees, Food, Transportation, and discretionary spending, users are able to monitor their income and expenses. This category system allows users to easily identify their spending habits and pinpoint areas where adjustments could be made to improve savings or spending habits. The emphasis on user feedback is a key feature of "Kards Keeper." By collecting and analyzing user feedback, the development team can constantly improve the program to meet the needs and expectations of its

users. Not only does enhancing the user experience lead to customer loyalty and trust, but it also improves satisfaction levels., making "Kards Keeper" the premier choice for personal finance management.

Although "Kards Keeper" is currently created for a college project, it serves as a fundamental prototype that can be enhanced and expanded upon in the future. This project demonstrates a deep understanding of the importance of financial literacy and the potential for technology to enhance the effectiveness and accessibility of budgeting.

III. Objectives

- **Create a user-friendly budget tracker app using Java.**
  - Making the application as intuitive as possible, it will be easier for the user to navigate through and sort out all their expenses and income efficiently and effectively.

- **Create a database to store the values of the users.**
  - Remembering information is important when it comes to finances. Storing information from the application into a database will benefit the users as they can put more focus on other tasks without worrying about their data being lost.

- **Implement functionality to categorize expenses and income.**
  - Compartmentalizing the users' budgeting allows them to distribute how much money they should be spending and where they should spend it.

The objectives that were kept in mind during the creation of this project was the user friendliness of the application as well as the categorization of expenses to allow for a more detailed breakdown, and tracking of the user's cash flow.

IV. Object-Oriented Structures and Algorithms

Encapsulation - Bundling data (fields) and methods (functions) that operate on the data into a single class. It blocks direct access to certain components in order to protect the object's internal state.

Example of Private and Public Fields in Program:

```
private static TotalPanel totalPanel;
private static JFrame mainFrame;


public static double calculateTotalAmount(String tableName) {// Some code}
```

```
public static void createAndShowGUI(JFrame mainFrame) {// Some code}
```

Abstraction - includes hiding the complex implementation details while presenting only the relevant functionality.

Example in Program:

```
public static void createAndShowGUI(JFrame mainFrame) {
    JFrame frame = new JFrame("Cash Screen");
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setSize(600, 700);
    frame.setLayout(new BorderLayout());

    String totalAmount = calculateTotalAmount();
    System.out.println("Total amount: " + totalAmount); // Print total amount in terminal

    CustomPanel panel = new CustomPanel("Cash", totalAmount);
    frame.add(panel, BorderLayout.NORTH);// Some Code}
private static String calculateTotalAmount() {
    String sql = "SELECT SUM(amount) AS total FROM cash";
    double total = 0.0;
    try (Connection conn = DriverManager.getConnection("jdbc:sqlite:kards.db");
         PreparedStatement pstmt = conn.prepareStatement(sql);
         ResultSet rs = pstmt.executeQuery()) {
        if (rs.next()) {
            total = rs.getDouble("total");
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    return String.format("%.2f", total);
}
```

Inheritance - class inheriting properties from other classes for reusable code and have structure.

Example in Program:

```
JFrame frame = new JFrame("Cash Screen");
JPanel panel = new JPanel();
JButton button1 = new JButton("Cash");
```

Polymorphism - objects to be treated as instances of their parent class rather than their actual class. It enables one interface to be used for a general class of actions, simplifying code and making it more flexible.

Example in Program:

```
button1.addActionListener(e -> CashPage.createAndShowGUI(mainFrame));
toggleButton.addActionListener(e -> {
    if (toggleButton.isSelected()) {
        toggleButton.setText("Expense");
        showExpensePanel();
    } else {
        toggleButton.setText("Income");
        showIncomePanel();
    }
});
```

V. Methodology

The project followed an iterative development process, beginning with the creation of a basic user interface (UI) for input and output tasks. An SQLite database was then implemented, which required addressing compatibility and configuration concerns. The construction and testing of database schemas guaranteed that data was stored and retrieved accurately. The UI was later improved to show calculated totals and provide a toggle mechanism for transaction categorization. Throughout the development cycle, rigorous testing, including the usage of print statements for debugging, was carried out to fine-tune the applications functionality.

IDE Used: Intellij IDEA Community Edition

Jar Used: sqlite-jdbc-3.34.0.jar | acm.jar

Project Development Process:

1. Setup new project and add JDBC jar to build path
2. Design Main Class
    a. MainDisplay - what the user will see when first open. Seeing the total amounts as well as buttons
3. Create Database Helper
    a. Class that handles all the database operations for the SQLite
4. Setup the GUI Components
    a. For the income/ expense buttons and the input output of text fields
5. Event handling
    a. Switching to different panels
    b. Displaying the correct category buttons
6. Debugging
    a. Error checking
        i. Syntax Error
        ii. Logical Error

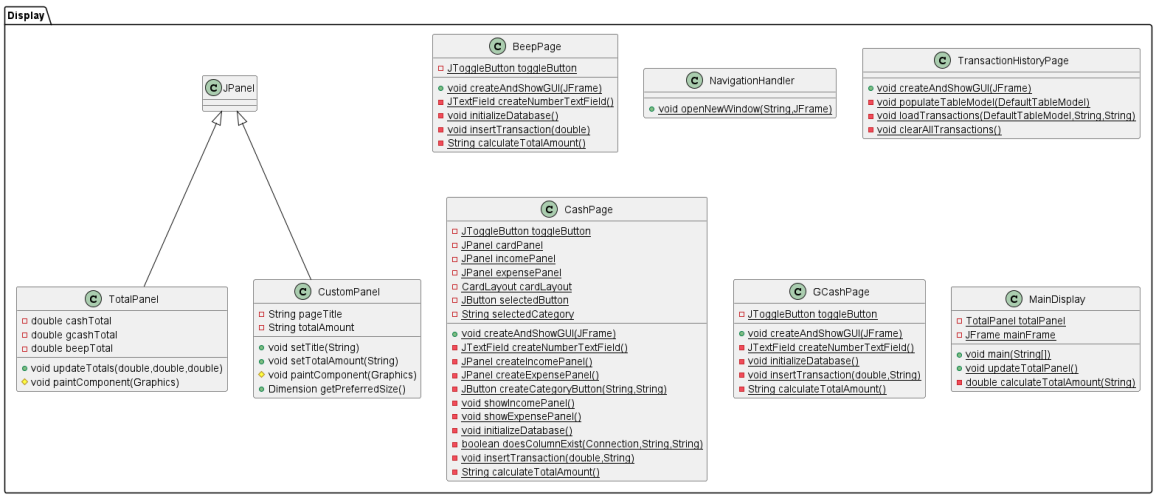


Figure 5.1. UML Diagram

table(cash, gcash, beep)		
Category	Amount	Income/ Expense
Salary	100	Income
Food	-25	Expense

Queery: SELECT SUM(amount) AS TOTAL FROM + tablename

MainDisplay table	
Transaction	Total
Cash	75
Gcash	0
Beep	0

Figure 5.2. SQL Query

VI. RDC (Results-Discussion-Conclusion)

- **Results**

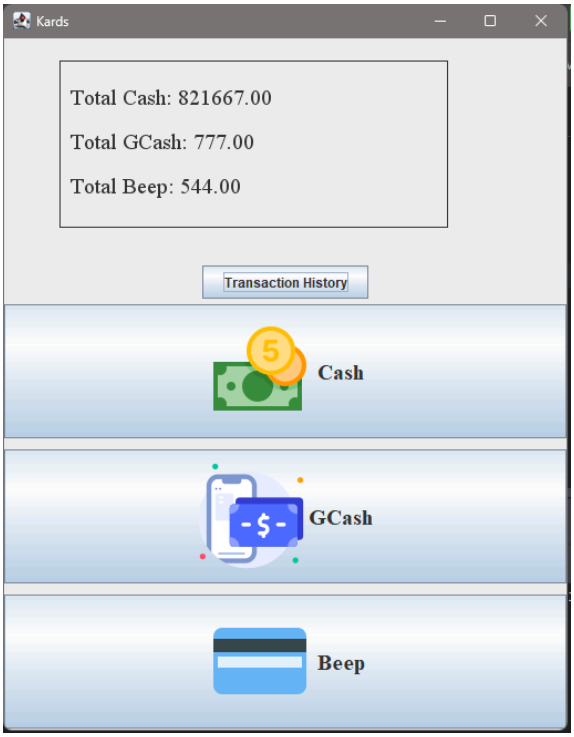


Figure 1. Main Page

This page appears when the user runs the program. They will be directed to the main page of the app wherein they can navigate through the different features of cash transactions including cash, GCash, and Beep. Additionally, there is a panel at the top of the screen that shows the total balance left for each of their transactions. Below the panel is a transaction history button that displays the summary of transactions.

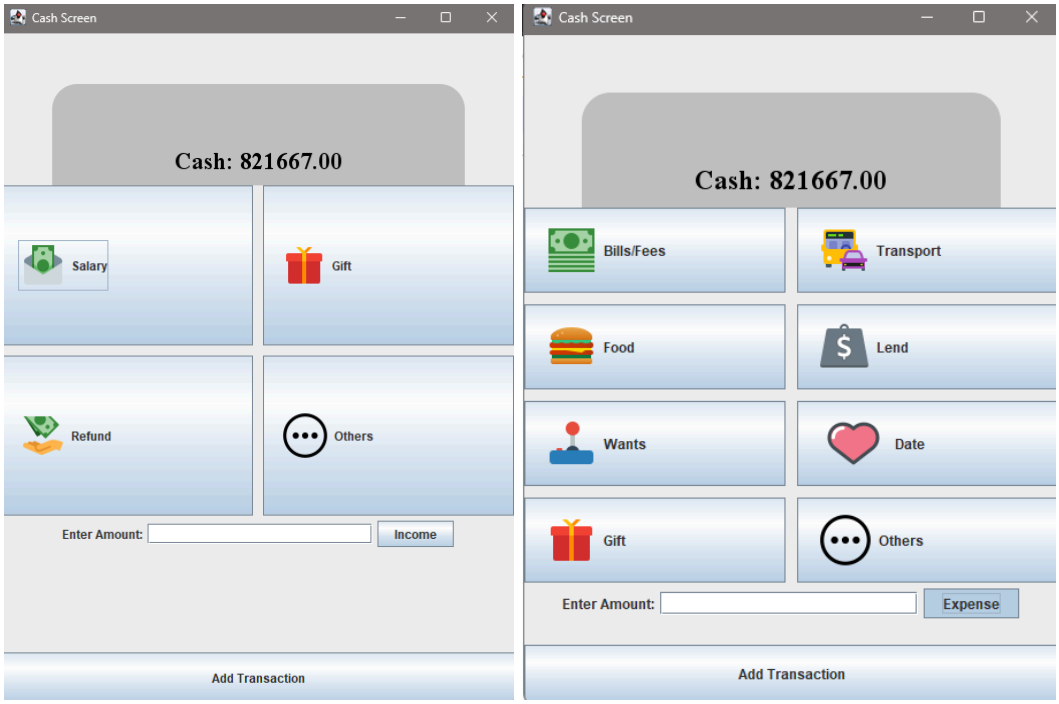


Figure 2. Cash Page

The cash page appears once the user clicks on ‘Cash” in the main page. Here, they can input any amount that they want and label it off as income or expense. On this page, there are categories depending on what you label your value (income/expense). Labeling can help the user easily identify what they are spending on when they track in the transaction history page.

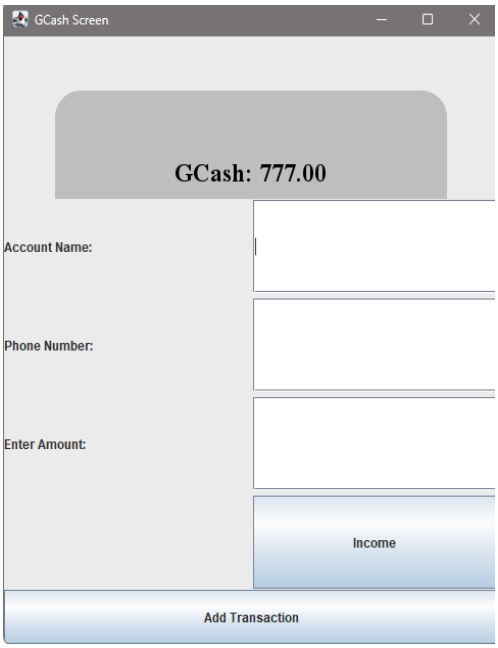


Figure 3. GCash Page

On the GCash page, it allows the user to input the account name and phone number of the person that they have transacted with. This adapts the real features in an existing application since the user will be required to enter the account information before making a transaction.

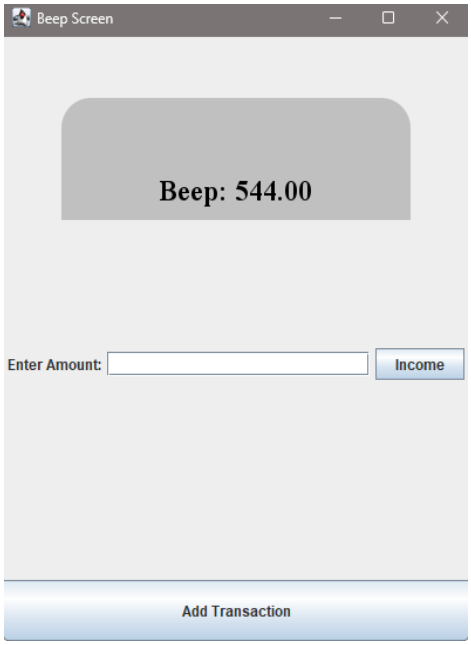


Figure 4. Beep Page

The Beep page displays a simple interface that allows the user to track their balance on their beep cards. This page can be improved and developed in future developments of the program.

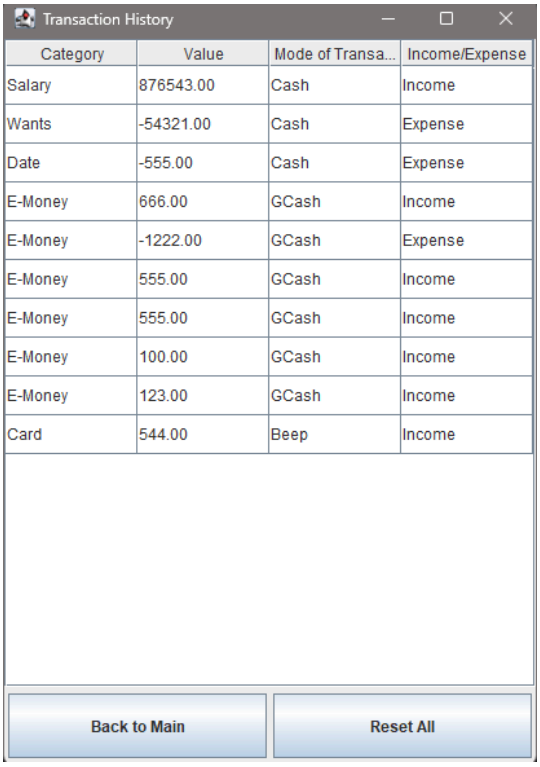


Figure 5 Transaction History Page

The purpose of the transaction history page is to help the user visualize the summary of their income and expenses. This will help them make an informed decision on how they would responsibly spend their remaining balance. Additionally, from this, they can be aware on what

For the performance of the app, it handles the existing features according to what it is supposed to do. All transaction data is stored in an SQLite database, ensuring persistence and reliability. The app also includes error handling mechanisms to manage database connectivity issues and other potential runtime errors, ensuring a stable user experience.

- **Discussion**

The budget tracker application, "Kards Keeper," was developed with the goal of providing users with a simple yet effective way to manage their finances across different platforms, including cash transactions, GCash, and Beep cards. Additionally, a key objective was to create a database that stores user transaction data, allowing users to view their transaction history even after exiting the program. The project successfully achieved these objectives, as demonstrated by the app's functionality and performance.

As shown in the results, "Kards Keeper" effectively integrates multiple financial platforms into a single interface, providing a streamlined user experience. The app's main page serves as a central hub, enabling users to quickly navigate through various features and view their overall financial status. The cash, GCash, and Beep pages allow for precise tracking and categorization of transactions, while the transaction history page provides a clear summary of all financial activities, helping users make informed decisions about their spending.

The use of an SQLite database ensures that all data is securely stored and remains accessible across sessions, enhancing the app's reliability and user convenience. Overall, the successful implementation of these features demonstrates that "Kards Keeper" is an effective tool for personal finance management, meeting the project's goals of simplicity, functionality, and reliability.

- **Conclusion**

Ultimately, this program can be considered a success as the results align with the initial objectives of the project.The program effectively connects different financial platforms, reliably tracks transactions, and securely stores data for easy access across sessions. It has proven to be a simple yet effective budget tracker, fulfilling its primary goal of aiding users in managing their finances.

However, there is clear room for improvement. While the current GUI is designed to be "user-friendly," enhancing its visual appeal and functionality could significantly improve the overall user experience. Future developments could focus on refining the user interface to make it more intuitive and aesthetically pleasing. Moreover, adding a graph in the app to visualize expenses over time would provide users with a more comprehensive understanding of their spending patterns, making the app even more useful for financial planning since it promotes the users to make an informed decision. Another improvement would be the introduction of a security feature, such as a login page with authentication, to protect users' financial data and increase their confidence in using the app.

These recommendations suggest a clear path forward for making "Kards Keeper" an even more reliable and valuable tool for personal finance management.

VII. References

W3Schools.com. (n.d.). https://www.w3schools.com/sql/sql_syntax.asp

Microsoft SQL Server LocalDB | IntelliJ IDEA. (n.d.). IntelliJ IDEA Help. https://www.jetbrains.com/help/idea/connecting-to-sql-server-express-localdb.html

Mohideen, N. (2023, November 30). *5 budgeting apps that every student should have*. https://www.urbanstudentlife.com/blog/5-budgeting-apps-that-every-student-should-have

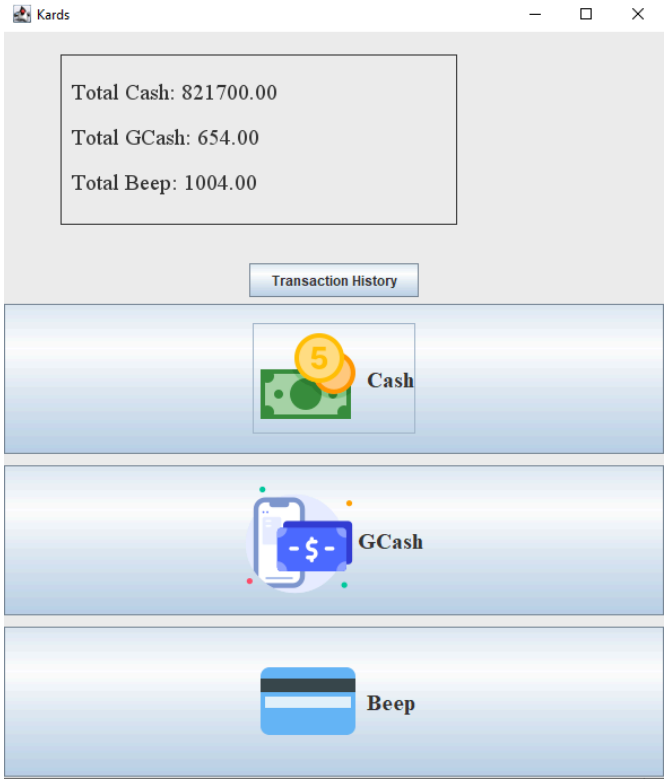
VIII. Appendices:

Project Source Code:

LBYCPEI_PROJECT

User manual of the system:

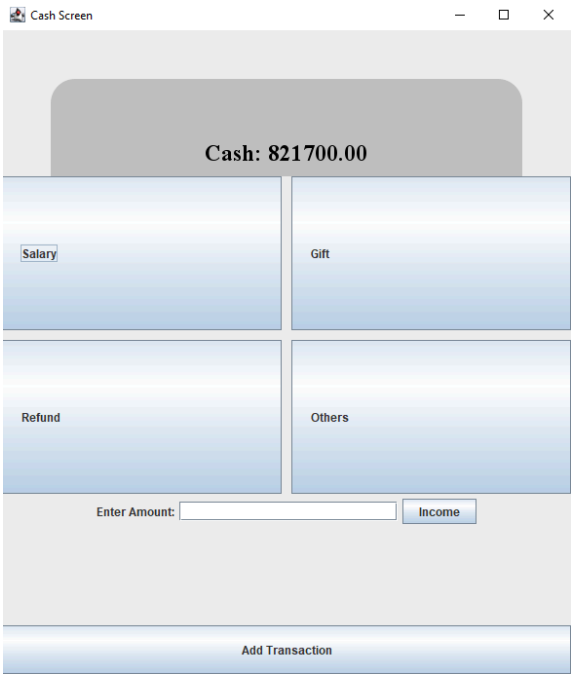
Upon opening the app, the user will be greeted with a screen as shown in the picture below



As seen in the picture above, the main menu displays information about the balance of each compatible payment account present in the app, a button to display transaction history, as well as the respective payment accounts present in the app.

Upon selecting any of the 3 payment accounts available, The user will be greeted with a similar screen as this shown on the right.

This menu is where the user will select on whether or not they are tracking down earnings or expenditures, as well as the category of either.



The photo above shows the available income options, while the photo on the right shows the options for expenses tracking.

Upon selecting either the expense or income option, as well as selecting the category for either option, the User will then be brought back to the main menu where they are once again able to track their personal finance on any of the available options.

Alternatively, if the user wishes to return to the main menu without changing the balance of the selected account they may opt to press the “Add Transaction” Button Without inputting any numerical value in the “Enter Amount” bracket, which all the same will lead the user back to the main menu.

As for the Transaction history, as mentioned in the previous paragraphs, if the user is to press this button they will be greeted with this menu:

Category	Value	Mode of Transaction	Income/Expense
Salary	876543.00	Cash	Income
Wants	-54321.00	Cash	Expense
Date	-555.00	Cash	Expense
Gift	90.00	Cash	Income
Food	-25.00	Cash	Expense
Others	-32.00	Cash	Expense
E-Money	666.00	GCash	Income
E-Money	-1222.00	GCash	Expense
E-Money	555.00	GCash	Income
E-Money	555.00	GCash	Income
E-Money	100.00	GCash	Income
Card	544.00	Beep	Income
Card	460.00	Beep	Income

Here the user is able to see a breakdown and compilation of their personal finances on all the available accounts present within the app.

The user is also given the option of either returning to the main menu, or to reset the logged transaction history saved in the application.