

BABEŞ-BOLYAI UNIVERSITÄT CLUJ-NAPOCA
FAKULTÄT FÜR MATHEMATIK UND INFORMATIK
INFORMATIK IN DEUTSCHER SPRACHE

BACHELORARBEIT

Optimierte Belegverwaltung mit ML Kit OCR und Firebase - “SmartReceipts”

Betreuer

Lekt. dr. ing. Kuderna-Iulian Bența

Eingereicht von

Feier Mihail-Gheorghe

2024

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA FACULTATEA DE
MATEMATICĂ ȘI INFORMATICĂ SPECIALIZAREA INFORMATICĂ ÎN
LIMBA GERMANĂ

LUCRARE DE LICENȚĂ

Gestionarea optimizată a chitanțelor cu
ML Kit OCR și Firebase -
“SmartReceipts”

Conducător științific

Lect. dr. eng. Kuderna-Iulian Bența

Autor

Feier Mihail-Gheorghe

2024

BABEȘ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN GERMANY

DIPLOMA THESIS

Optimized Receipt Management with
ML Kit OCR and Firebase -
“SmartReceipts”

Supervisor

Lect. dr. eng. Kuderna-Iulian Bența

Author

Feier Mihail-Gheorghe

2024

Abstract

In this thesis, we developed and analyzed an application called SmartReceipts, designed for managing and organizing receipts using Optical Character Recognition (OCR) technology and a Firebase database. The application enables users to efficiently scan, store, and manage their receipts, offering features such as filtering, sorting, and visualizing data in graphs and lists.

SmartReceipts distinguishes itself from other financial management apps through its advanced integration of OCR technology, which automates data entry, thereby reducing manual effort and human errors. The application utilizes Firebase for authentication, data, and image storage, ensuring a secure and high-performance experience for users.

The development process involved modern technologies and tools like Android Studio, Google's ML Kit for text recognition, and an MVVM (Model-View-ViewModel) architecture to separate business logic from the user interface.

The thesis also explores potential extensions of the application, such as adding support for new types of documents, like utility bills, and improving the user interface and reporting functionalities. In conclusion, SmartReceipts represents an innovative and efficient solution for receipt management, with room for future growth and enhancement.

The thesis is structured into three chapters. The first chapter explains the role of financial education and money management, highlighting why financial literacy is important, how it can benefit daily life, and the advantages it provides. The second chapter discusses the role of mobile applications in financial education and money management, exploring their impact on expense tracking and how they help users organize and monitor their finances efficiently and accessibly. This chapter also includes a review of other expense management applications, summarizing their functionalities and concluding with a description of why SmartReceipts is a better choice than the alternatives. The final chapter focuses on our original contribution, presenting the application we developed as a tool to help people better manage their expenses.

Introduction.....	6
CHAPTER 1 - Financial Education and the Importance of Money Management	8
1.1 Introduction.....	8
1.2 The Importance of Financial Education	8
1.3 Financial Management	9
<i>1.3.1 Budgeting</i>	<i>9</i>
<i>1.3.2 Saving.....</i>	<i>9</i>
CHAPTER 2 - Managing Your Budget with Digital Tools	10
2.1 How Applications Assist Us in Money Management.....	10
2.2 Options for Financial Management Apps	11
2.3 Benefits of Using Digital Applications for Money Management.....	12
2.4 Why Choose SmartReceipts Over Other Applications	13
CHAPTER 3 - SmartReceipts.....	16
3.1 Development Stage.....	16
<i>3.1.1 Overview</i>	<i>16</i>
<i>3.1.2 Application Architecture</i>	<i>17</i>
3.2 Tools Used	17
<i>3.2.1 Android Studio</i>	<i>17</i>
<i>3.2.2 Firebase</i>	<i>23</i>
<i>3.2.3 Google ML Kit</i>	<i>27</i>
3.3 Overview of SmartReceipts.....	31
<i>3.3.1 Pages and Functionality</i>	<i>32</i>
3.3.1.1 Start Page - “ReloadActivity”	32
3.3.1.2 Log-in and Register.....	34
3.3.1.3 MainActivity	37
3.3.1.4 Fragment Show (Home).....	39
3.3.1.5 AddFragment.....	42
3.3.1.6 ListFragment	45
3.3.1.7 ReadFragment	47
3.4 Application Testing.....	49
3.5 User Experience and Feedback.....	51
3.6 Future Work and Potential Enhancements for SmartReceipts	54
CONCLUSION	56

Introduction

In today's fast-paced and ever-changing world, the need for effective personal finance management has become more crucial than ever. As financial transactions grow increasingly complex and the importance of maintaining a clear record of expenses rises, individuals often find themselves facing the challenge of efficiently managing and organizing their receipts. This is particularly vital for those who aim to closely monitor their spending habits, stay within budget, and make informed financial decisions.

One of the most common difficulties consumers encounter is the task of systematically collecting, storing, and organizing receipts to accurately track their expenditures. Traditional methods of managing paper receipts, such as manual sorting and storage, can be time-consuming, prone to errors, and difficult to maintain over the long term. Furthermore, as the volume of transactions increases, so does the likelihood of losing or misplacing important receipts, which can lead to gaps in financial records and complications when budgeting or filing taxes.

To address these challenges, this thesis introduces the development of an innovative application designed to transform how individuals manage their receipts. The application leverages advanced technology to scan and digitally store receipts, automatically extracting key information such as the total amount spent, transaction date, and merchant details. This innovative solution not only streamlines the receipt management process but also enhances the user's ability to monitor and analyze spending patterns over time.

The application features a user-friendly interface that allows users to easily view stored receipts, filter them based on specific criteria such as date or category, and generate detailed reports and charts that provide insights into their spending behavior over different time periods. This level of analysis can help users identify spending trends, adjust their budgeting strategies, and ultimately gain better control over their financial health.

The primary objective of this thesis is to design and develop a technological solution that is both accessible and efficient, addressing common pain points associated with receipt management. By offering a comprehensive tool that simplifies the process of organizing and analyzing receipts, this application aims to enhance the overall financial management experience for users. In the following sections of this paper, we will explore the methodologies employed during the application's development, discuss the technical implementation of its various functionalities, and evaluate the results and benefits observed through its practical use. We will also consider the potential implications of this technology on personal finance management and explore future improvements that could further expand its capabilities.*

** I hereby declare that this thesis has been written solely by me, without the use of ChatGPT or any other automated writing tools. All content and ideas presented are the result of my own research and analysis.

CHAPTER 1 - Financial Education and the Importance of Money Management

1.1 Introduction

Financial education and effective money management are crucial for maintaining long-term financial stability and avoiding financial difficulties. In an era where access to information and the complexity of financial products are constantly increasing, the ability to understand and manage personal finances effectively has become increasingly vital. This chapter delves into the significance of financial education and money management, highlighting the benefits they offer to individuals and society [1].

1.2 The Importance of Financial Education

Financial education encompasses understanding basic economic and personal finance concepts, as well as developing the skills necessary to make informed financial decisions. This includes knowledge of income, expenses, savings, investments, and credit [2].

A high level of financial literacy can help reduce debt, increase savings, and improve overall financial well-being. Research indicates that financially educated individuals are less likely to face financial difficulties and are better prepared to handle unexpected events[3]. Additionally, financial education is linked to better retirement planning and smarter investment choices [4].

Benefits of Financial Education:

- *Avoiding Financial Pitfalls:* Financially knowledgeable individuals are better equipped to avoid financial traps such as excessive debt and unplanned spending [5].

- *Increasing Savings:* Financial education promotes regular saving habits, providing a financial cushion for unforeseen circumstances and aiding in achieving long-term financial goals [6].
- *Enhancing Investment Decisions:* Those with strong financial knowledge are more capable of making informed investment decisions, potentially yielding higher returns and minimizing risks[4].
- *Retirement Planning:* A good financial education helps in developing effective strategies for saving and investing for retirement, ensuring long-term financial security[7].

1.3 Financial Management

Financial management involves the processes of planning, organizing, controlling, and monitoring financial resources to achieve personal financial goals. This encompasses activities such as budgeting, saving, investing, and managing debt[8].

1.3.1 Budgeting

Budgeting is a vital tool for effectively managing finances. It entails planning income and expenditures to ensure that financial resources are used optimally. A well-structured budget can help identify and eliminate unnecessary expenses and increase savings. According to a study by the Consumer Financial Protection Bureau (CFPB), individuals who budget their income and expenses tend to have better financial health and are less likely to face unexpected financial difficulties[5].

1.3.2 Saving

Regular saving is essential for creating a financial buffer for unexpected situations and for achieving long-term financial goals, such as buying a home or preparing for retirement. Research by the National Bureau of Economic Research (NBER) indicates that individuals who save regularly experience greater financial security and less financial stress[7].

CHAPTER 2 - Managing Your Budget with Digital Tools

In the digital age, technology plays an integral role in all aspects of life, including personal finance management. Digital applications designed for money management provide users with enhanced control over their income and expenses, making financial planning and informed decision-making more accessible. This chapter explores how digital tools can assist in managing finances and reviews various available options on the market[8].

2.1 How Applications Assist Us in Money Management

Financial management apps are created to help users efficiently organize and monitor their finances. These applications offer a wide range of features that simplify the process of personal budgeting and enhance financial health.

- *Tracking Income and Expenses:* A key feature of these apps is the ability to record and monitor income and expenses. Users can see in real-time how and where they are spending their money, helping them identify areas where they can save. According to a study by the JPMorgan Chase Institute (2019), users of financial apps have a better understanding of their spending habits and can save more[9].
- *Setting and Monitoring Budgets:* These apps allow users to set monthly or annual budgets for various expense categories, such as food, transportation, entertainment, etc. By tracking these budgets, users can see if they are on track to stay within their set limits[9].
- *Automated Expense Categorization:* Many apps can automatically categorize bank transactions. This feature simplifies the process of tracking expenses and helps users quickly identify unplanned or excessive spending[9].
- *Generating Financial Reports:* Financial management apps can generate detailed reports that provide a clear picture of a user's financial situation. These reports may include charts

and graphs that illustrate spending and savings distributions across different categories and time periods[9].

- *Long-Term Financial Planning:* Apps can assist users in planning long-term financial goals, such as saving for a house, a car, or retirement. By setting clear goals and tracking progress, users can remain motivated and disciplined[9].
- *Notifications and Alerts:* To help users avoid late payments or budget overspending, apps can send notifications and alerts. These features help users stay informed about their financial situation and take timely actions when needed[9].

2.2 Options for Financial Management Apps

There are numerous apps available that can assist with managing finances, varying in features, interfaces, and costs. Below are some of the most popular financial management apps:

- *Mint:* Mint is a well-known and widely used financial management app. It offers a broad range of features, including tracking income and expenses, setting budgets, automatic expense categorization, and generating financial reports. The app integrates with users' bank accounts and credit cards, providing a comprehensive overview of their finances. Mint also sends notifications and alerts for bills and unusual transactions[10].
- *YNAB (You Need A Budget):* YNAB is a budgeting app that focuses on helping users live within their current income and avoid debt. It employs a unique budgeting methodology encouraging users to assign every dollar a purpose. YNAB offers courses and educational resources to help users learn and implement sound financial management principles. Studies show that YNAB users save an average of \$600 in the first two months and over \$6000 in the first year of use[11].
- *Personal Capital:* Personal Capital is a wealth management app that combines financial management with investment tracking features. The app allows users to monitor income, expenses, and savings while providing tools to track investment portfolios. Personal Capital offers detailed analysis of investment performance and recommendations for

portfolio optimization. It is ideal for users who want to manage both their personal finances and investments in one place[12].

- *PocketGuard*: PocketGuard is a simple and intuitive app that helps users monitor spending and manage budgets. The app integrates with users' bank accounts and credit cards, providing an overview of available finances. PocketGuard uses an algorithm to calculate how much money the user has available for spending after deducting bills, savings, and other planned expenses. This feature helps users avoid overspending and stay within their budget[13].
- *Goodbudget*: Goodbudget is a budgeting app based on the envelope method, which helps users manage their income and expenses by allocating money into virtual envelopes. Each envelope represents a spending category, such as food, transportation, or entertainment. Users can add money to envelopes at the beginning of each month and track expenses to ensure they stay within set limits. Goodbudget is ideal for users who prefer a visual and structured approach to budgeting[14].
- *Wally*: Wally is a financial management app that focuses on simplicity and ease of use. The app allows users to track income and expenses, set budgets, and monitor savings. Wally offers features such as receipt scanning and automatic expense categorization, simplifying the transaction tracking process. The app also provides analysis and reports to help users better understand their financial habits and make informed decisions[15].

2.3 Benefits of Using Digital Applications for Money Management

Using digital applications for managing finances offers several benefits that can enhance user's financial health:

- *Accessibility and Convenience*: These apps are available on smartphones and tablets, allowing users to manage their finances from anywhere at any time. This eliminates the need for maintaining physical records or accessing separate websites to track accounts[6].

- *Automation:* Many apps automate the process of recording and categorizing transactions, saving time and reducing errors. They can also send automatic notifications and alerts to help users stick to their budgets and deadlines[6].
- *Analytics and Reports:* The apps provide detailed analytics and reports that help users better understand their financial situation. This information is crucial for making informed decisions and identifying areas for potential savings[6].
- *Long-Term Planning:* By setting financial goals and tracking progress, users can plan more effectively for the long term. These apps help maintain motivation and discipline in saving and investing[6].
- *Security:* Most financial management apps use advanced security technologies to protect users' information, including data encryption, two-factor authentication, and other security measures to ensure the confidentiality and integrity of financial data[6].

2.4 Why Choose SmartReceipts Over Other Applications

In the digital age, managing personal finances effectively is crucial for maintaining a clear and organized view of one's expenses and income. SmartReceipts, the application we have developed, introduces innovative and efficient solutions that make it stand out from other popular financial management apps like Mint, YNAB, Personal Capital, PocketGuard, Goodbudget, and Wally.

What sets SmartReceipts apart is its advanced use of Optical Character Recognition (OCR) technology, which allows users to scan and automatically record receipts and other financial documents. This functionality significantly reduces the manual effort involved in data entry while ensuring accuracy and speed. While other applications often require manual data entry or bank account synchronization such as Mint and Wally, which allow manual data input or receipt scanning SmartReceipts fully automates this process. Users simply take photos of their receipts, and relevant information like the total amount, date, and location are automatically and accurately extracted. This greatly reduces manual labor and minimizes human error, offering a much smoother and more efficient experience.

Another advantage of SmartReceipts is its flexibility and ability to adapt to various types of financial documents. In the future, the app will expand support to include utility bills for electricity, gas, TV, and other services, enabling users to manage all financial aspects in one place. This expansion will make SmartReceipts an indispensable tool for users seeking a comprehensive solution for managing personal finances, as opposed to apps like PocketGuard and Goodbudget, which focus more on budgeting and expenses but do not offer the same extensive document management capabilities.

SmartReceipts also provides an interactive LineChart graph that displays the progression of expenses over time. This feature allows users to identify spending trends and patterns, offering detailed insights into financial behavior. Users can select individual points to view additional details about expenses recorded on specific days, a level of interactivity not typically found in apps like Mint or Personal Capital.

In addition to efficient expense management, SmartReceipts offers advanced data filtering and analysis functionalities. Users can filter expenses based on various criteria, such as date and amount, and generate detailed reports to better understand their spending habits. This detailed analytical capability is essential for users who want to make informed financial decisions and optimize their budgets. While YNAB offers budgeting and expense tracking features, it does not provide the same level of data filtering and customization.

SmartReceipts uses Firebase for user authentication and secure data storage, ensuring that all financial information is protected and accessible only to authorized users. While other applications, such as Personal Capital, offer security for investment data, SmartReceipts provides a comprehensive solution that covers both personal financial data and scanned documents.

Regarding user experience, SmartReceipts stands out with its simple and intuitive interface, making the app easy to use even for those who are not tech-savvy. The process of scanning and recording receipts is fast and efficient, and the user interface is designed to offer easy navigation and quick access to all functionalities.

In conclusion, SmartReceipts provides a range of distinct advantages that make it superior to other financial management apps available on the market. Its advanced OCR technology, flexibility in managing various types of financial documents, detailed data filtering and analysis capabilities,

and simplified user experience make SmartReceipts an indispensable tool for efficient personal finance management. With plans for continuous expansion and improvement, SmartReceipts has the potential to become a leading application in the field of financial management.

Digital money management apps are valuable tools for anyone looking to improve their financial health. They offer a wide range of functionalities that simplify tracking income and expenses, budgeting, and long-term planning. By using these apps, users can gain better control over their finances and make informed decisions that ensure long-term financial stability.

CHAPTER 3 - SmartReceipts

3.1 Development Stage

3.1.1 Overview

SmartReceipts is an Android application designed to help users scan and manage receipts efficiently and intuitively. The app uses the MVVM (Model-View-ViewModel) architecture to clearly separate the application's logic from the user interface, facilitating easier development and maintenance. It utilizes Firebase for authentication and data storage, along with Google's ML Kit for Optical Character Recognition (OCR).

SmartReceipts was developed to assist users in scanning and managing receipts in a practical and intuitive manner. As discussed in previous chapters, personal finance management is crucial for long-term financial stability, and SmartReceipts offers a practical solution for tracking expenses. The app features a simple and user-friendly interface, providing access to a comprehensive yet compact set of functionalities. It is an Android application comprising multiple activities and fragments. The main activity provides access to other fragments named "Add," "Show," "List," and "Read." Additionally, there are two extra activities related to authentication, helping users to sign up or log in to our platform, plus a loading activity.

The app is designed to assist users in scanning, understanding, and managing receipts. The "Add" page allows users to scan receipts and extract key information, such as the total payment, date, and store of purchase, using Google's ML Kit for OCR. The "List" page helps users view all scanned receipts in an organized manner. The "Show" page offers a clear overview of expenses, and users can filter receipts by date for more specific analysis. Finally, the "Read" page allows users to view the full receipt.

3.1.2 Application Architecture

SmartReceipts employs the MVVM architecture to clearly delineate responsibilities among the Model, View, and ViewModel, facilitating easier testing and feature expansion. Firebase is used for authentication and data storage, ensuring efficient and secure user information management.

The Model is represented by the *Receipt*, *ListReceipt*, and *ListReceiptViewModel* classes, which handle the structure and manipulation of receipt data. The View consists of activities and fragments that display data and manage user interactions, while the ViewModel acts as an intermediary between the Model and the View, managing data and synchronizing it with the UI through LiveData. The ViewModel ensures data persistence in case of screen rotations or other configuration changes[16].

This modular structure makes SmartReceipts easy to maintain and extend, while also providing an intuitive and efficient user experience. Users can scan receipts, view and filter expenses, and analyze data through detailed reports, all contributing to better personal finance management.

3.2 Tools Used

3.2.1 Android Studio

Description

Android Studio is the official integrated development environment (IDE) for creating Android applications, built on IntelliJ IDEA's code editor and development tools. Android Studio offers various features that significantly enhance developer productivity[17]. These include:

- *Gradle-based build system:* This flexible system manages and automates the build process, facilitating customization and efficiency.
- *Fast and versatile emulator:* The emulator included in Android Studio is powerful and feature-rich, providing a testing environment that closely simulates real devices.
- *Unified development environment:* It provides an integrated framework for developing applications for all types of Android devices, from phones and tablets to Android Wear, Android TV, and Android Auto.

- *Live Edit*: This feature allows real-time updates to composable in both emulators and physical devices, improving workflow and development efficiency.
- *Code templates and GitHub integration*: Android Studio includes predefined templates and GitHub integrations, helping developers quickly implement common features and import code samples.
- *Extensive testing tools and frameworks*: The IDE includes various tools and frameworks for testing applications, ensuring their quality and performance.
- *Lint tools*: These tools analyze the code to detect issues related to performance, usability, version compatibility, and other critical aspects, contributing to improved code quality.

Activities

In Android application development, activities are crucial components that define the individual screens users interact with. Each activity is tasked with displaying a user interface (UI) and managing user interactions on that screen. Activities are subclasses of the Activity class within the Android SDK and come with a well-defined lifecycle that ensures proper resource management and application behavior[18].

SmartReceipts and Utilized Activities

Within SmartReceipts, several activities are utilized to manage different aspects of the application. Each activity serves a specific purpose and ensures the core functionality of the app.

- *MainActivity*: This is the primary activity of the application, responsible for handling navigation between fragments and initializing the main components of the app.
- *LoginActivity*: Manages user authentication using Firebase Authentication, allowing users to log in or register with an email and password.
- *RegisterActivity*: Similar to LoginActivity, this activity handles the registration process for new users.
- *ReloadActivity*: Used for reloading application data or managing specific user sessions.

These activities are interconnected through intents, enabling users to easily navigate through the various screens and functionalities of the application.

Fragment

In Android application development, fragments are essential components that enable the creation of modular and flexible interfaces. Fragments are reusable parts of the user interface (UI) and application logic that can be combined within activities to build a dynamic and adaptable user interface. Fragments are subclasses of the `Fragment` class and can be managed independently within an activity[19].

SmartReceipts and Utilized Fragments

Within `SmartReceipts`, various fragments are employed to organize and manage the application's functionalities. The fragments used in `SmartReceipts` include:

- *ListFragment*: Displays a list of scanned receipts, utilizing a `RecyclerView` for showing the receipts in a scrollable list.
- *AddFragment*: Allows users to add new receipts either by scanning or manually entering them.
- *ShowFragment*: Shows the details of a specific receipt, where users can view detailed information and edit or delete the receipt.
- *ReadFragment*: Used to display the text extracted from scanned receipts using `ML Kit OCR`.

Each of these fragments is managed via `FragmentManager` within the activities, enabling users to navigate and interact with the application in a modular and efficient manner. We also use `Bundle` to pass data between fragments, ensuring a smooth user experience.

Example of data transfer between fragments:

```

// În ListFragment

override fun onItemClick(receipt: Receipt) {

    val detailFragment = ReadFragment.newInstance(receipt)

    requireActivity().supportFragmentManager.beginTransaction()

        .replace(R.id.fragment_container_view, detailFragment)

        .addToBackStack(null)

        .commit()

}

```

```

// În ReadFragment

companion object {

    private const val STORE_KEY = "store_key"

    fun newInstance(receipt: Receipt): Fragment {

        val fragment = ReadFragment()

        val args = Bundle()

        args.putParcelable(STORE_KEY, receipt)

        fragment.arguments = args

        return fragment

    }

}

```

By using these activities and fragments, SmartReceipts provides a modular, flexible, and easily manageable user interface, allowing users to efficiently manage receipts and access various application functionalities.

RecyclerView / Adapter / ViewHolder Pattern and LiveData

RecyclerView is an advanced widget in Android for displaying data lists in a performance-efficient and flexible manner. It succeeds ListView and GridView, supporting animations and more efficient memory management. RecyclerView is used alongside Adapter and ViewHolder to optimize data display and resource management[20].

The Adapter / ViewHolder pattern is crucial for the functionality of RecyclerView. The Adapter converts data into views for display, while the ViewHolder stores references to UI components, enabling their reuse and enhancing performance [21][22].

LiveData, a component from Android Architecture Components, facilitates reactive data management. It serves as an observable data source, allowing UI components to respond to data changes in sync with their lifecycle[23].

RecyclerView, Adapter, and ViewHolder in SmartReceipts

In SmartReceipts, RecyclerView, Adapter, and ViewHolder are utilized to display the list of scanned receipts. Implementing these components allows for efficient data management and high-performance display.

1. RecyclerView is used to present the list of receipts in a scrollable format. It enables the display of a large number of receipts without compromising the application's performance, as views are recycled and reused as the user scrolls through the list.

2. The AdapterReceipt is responsible for creating individual views for each receipt and binding data to these views. The adapter receives the list of receipts and creates ViewHolders for each item in the list.

3. ViewHolder stores references to the UI components of a receipt view (e.g., date and total amount). This allows RecyclerView to reuse existing views, reducing the number of calls to create new views and thus enhancing performance.

This approach ensures an efficient and high-performance display of receipts, providing users with a smooth and intuitive experience.

LiveData in SmartReceipts

LiveData is employed in SmartReceipts to manage and update the list of receipts in a reactive and efficient manner. It ensures that the application's UI is automatically updated when data changes, without requiring manual intervention from the developer[20].

1. Observability: LiveData allows UI components to observe data changes. In SmartReceipts, the list of receipts is stored in LiveData, and the RecyclerView is automatically updated when the data changes.

2. Lifecycle-awareness: LiveData is aware of the lifecycle of UI components and updates only active observers, preventing memory leaks and crashes. This is essential for ensuring the application's stability and preventing unnecessary resource consumption.

3. Automatic synchronization: LiveData automatically updates the UI when data changes. In SmartReceipts, this means any change in the receipt list (e.g., adding a new receipt or deleting an existing one) is immediately reflected in the RecyclerView, providing users with a real-time experience.

Implementation and Functionality in SmartReceipts

In SmartReceipts, the integration of RecyclerView, Adapter, ViewHolder, and LiveData was crucial for providing users with a robust and high-performing receipt management functionality.

RecyclerView and AdapterReceipt:

- When the application initializes the RecyclerView, it is linked to an AdapterReceipt, which is responsible for transforming each receipt into a displayable view.
- The AdapterReceipt creates and configures ViewHolders, which store references to UI components for each receipt. This allows RecyclerView to recycle views and reuse them to display new data, conserving resources and improving performance.

LiveData and ViewModel:

- ListReceiptViewModel manages the receipt data and stores it in a LiveData object. When data changes (e.g., when a new receipt is added or an existing receipt is deleted), LiveData automatically notifies active observers, and the RecyclerView is updated to reflect these changes.
- By utilizing ViewModels and LiveData, SmartReceipts benefits from efficient lifecycle management and reactive UI updates, ensuring a smooth and consistent user experience.

In SmartReceipts, the use of RecyclerView combined with the Adapter/ViewHolder pattern and LiveData enabled the creation of a responsive and high-performance user interface. RecyclerView was used to display the receipt list in an efficient and manageable way, while LiveData ensured the automatic update of the UI when data changed. This modular and reactive approach improved the user experience and maintained clean and maintainable code.

3.2.2 Firebase

Overview of Firebase

Firebase is a versatile platform providing a wide range of services for app developers. These services include real-time databases, authentication, cloud storage, analytics, notifications, and more. Firebase enables developers to focus on creating exceptional user experiences while handling the complexities of backend infrastructure[24].

In the SmartReceipts application, we utilized Firebase for authentication, data storage, and media resource management, leveraging the platform's flexibility and efficiency.

Firebase Services Used in SmartReceipts

For the development of SmartReceipts, the following Firebase services were integrated:

- Firebase Authentication
- Firebase Realtime Database
- Firebase Storage

Firebase Authentication

Firebase Authentication is a service that simplifies user authentication through various methods such as email and password, social logins (Google, Facebook, etc.), and other methods. In SmartReceipts, Firebase Authentication is used to allow users to register and log in securely[25].

Implementation in SmartReceipts:

- *User Registration:* Users can sign up using their email address and a password. Firebase manages the authentication data, ensuring security and efficient user management.
- *User Login:* Existing users can log in with their credentials. Firebase handles the authentication sessions and the security of user data.

Example Code from SmartReceipts for Authentication:

```
mAuth.createUserWithEmailAndPassword(email, password)
```

```
.addOnCompleteListener(this) { task ->  
    if (task.isSuccessful) {  
        // Înregistrare reușită  
        val user = mAuth.currentUser  
    } else {  
        // Înregistrare eșuată  
    } }
```

Firebase Realtime Database

Firebase Realtime Database is a NoSQL database that enables real-time data synchronization between users and devices. In the SmartReceipts, this database is used for storing and managing scanned receipts.

Firebase Realtime Database is another crucial service used in SmartReceipts, facilitating the storage and real-time synchronization of data among users. This database is employed in SmartReceipts to store information related to receipts, allowing users to access and manage this data from any internet-connected device[26].

Database Structure and Key-Node Logic:

Receipt data is stored in Firebase Realtime Database as nodes. Each user has a dedicated node containing their receipt data.

- **Primary Nodes:** In Firebase Realtime Database, data is organized as nodes. In SmartReceipts, each user is assigned a primary node identified by their unique user ID (uid).
- **Secondary Nodes:** Each primary node contains secondary nodes for each receipt. Each receipt is assigned a unique ID, automatically generated by Firebase using the push() method, which creates a node with a unique key under the user's node.

users

```
|
+-- userId
  |
  +-- receipts
    |
    +-- receiptId
      |
      +-- date: "01.01.2024"
      +-- sum: 50.0
      +-- location: "S.C Supermarket S.R.L"
      +-- imgName: "receipt_image_01.jpg"
      +-- img: "https://firebasestorege....."
```

Creating and Storing Data:

```

val receipt = Receipt(...)
val userId = FirebaseAuth.getInstance().currentUser?.uid
val database = FirebaseDatabase.getInstance().reference
val receiptId = database.child("receipts").child(userId!!).push().key
database.child("receipts").child(userId).child(receiptId!!).setValue(receipt)

```

Reading and Updating Data:

```

database.child("receipts").child(userId!!).addValueEventListener(object : ValueEventListener{
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        val receiptList = mutableListOf<Receipt>()
        for (receiptSnapshot in dataSnapshot.children) {
            val receipt = receiptSnapshot.getValue(Receipt::class.java)
            receiptList.add(receipt!!)
        }
        // Actualizează interfața de utilizator cu lista de bonuri
    }

    override fun onCancelled(databaseError: DatabaseError) {
        // Gestionarea erorilor
    }
})

```

Firebase Storage

Firebase Storage is a cloud storage service that facilitates the uploading and downloading of files, such as images, directly from Firebase applications. In SmartReceipts, Firebase Storage is employed to store images of receipts[27].

Implementation in SmartReceipts:

- *Uploading Receipt Images:* Receipt images are uploaded to Firebase Storage and linked with data stored in the Realtime Database.
- *Downloading and Displaying Images:* Users can view stored receipt images directly within the application.

Here is an example code snippet for file uploads:

```
val storageRef = FirebaseStorage.getInstance().reference
val userId = FirebaseAuth.getInstance().currentUser?.uid
val receiptImageRef = storageRef.child("images/$userId/${receipt.imgName}")
val uploadTask = receiptImageRef.putFile(imageUri)
uploadTask.addOnSuccessListener {
    // Încărcare reușită
}.addOnFailureListener {
    // Încărcare eșuată
}
```

By integrating these services, SmartReceipts ensures a secure and seamless user experience while efficiently managing data and resources.

Firebase is integral to SmartReceipts, providing essential functionalities for user authentication, data storage, and OCR service integration. With Firebase Authentication, the application ensures that only authorized users can access its data and features. Firebase Storage handles image storage, and Firebase Realtime Database facilitates the efficient storage and synchronization of receipt data, giving users quick and real-time access to their information.

3.2.3 Google ML Kit

Description

Google ML Kit is a powerful machine learning platform designed for mobile app developers. It offers a range of APIs that facilitate easy integration of machine learning functionalities into mobile applications without requiring advanced knowledge in the field. In my

application, SmartReceipts, I utilized Google ML Kit for optical character recognition (OCR) to extract information from scanned receipts[28].

Google ML Kit Services Used in SmartReceipts

For developing SmartReceipts, I integrated the Text Recognition service of Google ML Kit, which allows the recognition and extraction of text from images. In SmartReceipts, this functionality is crucial for enabling users to scan receipts and extract relevant information such as the total amount, date, and store.

Implementation in SmartReceipts includes:

- *Scanning Receipts:* Users can scan receipts using their device's camera, and ML Kit OCR automatically extracts text from the image.
- *Extracting and Displaying Information:* The extracted information is displayed in the user interface, allowing users to view and manage relevant data from the receipts.

Example code for text recognition and extraction in SmartReceipts [29]:

```
fun recognizeTextFromImage(imageUri: Uri) {  
    val image: InputImage  
    try {  
        image = InputImage.fromFilePath(context, imageUri)  
        val recognizer = TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS)  
        recognizer.process(image)  
        .addOnSuccessListener { visionText ->  
            // Procesare text recunoscut  
            processRecognizedText(visionText)  
        }  
        .addOnFailureListener { e ->  
            // Gestionare eroare  
            e.printStackTrace()  
    }  
}
```

```

    }
} catch (e: IOException) {
    e.printStackTrace()
}
}

fun processRecognizedText(visionText: Text) {
    for (block in visionText.textBlocks) {
        for (line in block.lines) {
            for (element in line.elements) {
                val text = element.text
                // Prelucrarea textului recunoscut
            }
        }
    }
}

```

Integrating Google ML Kit into SmartReceipts was essential to provide users with robust text recognition functionality. Here's how it was implemented:

1. *Configuring ML Kit in the Project:* To use ML Kit, it was necessary to add the relevant dependencies to the project's build.gradle file [30].
2. *Scanning and Recognizing Text:* Users can scan receipts using their device's camera, and the captured image is processed by ML Kit OCR to extract text. The recognized text is then processed and displayed in the user interface.
3. *User Interface:* The user interface was designed to display the extracted information from the receipts in a clear and organized manner. Users can view the total amount, date, and store directly from the app.

Example code for camera integration:

```
import android.content.Intent
```

```
import android.provider.MediaStore
```

```
private val REQUEST_IMAGE_CAPTURE = 1
```

```
private fun dispatchTakePictureIntent() {
```

```
    Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->
```

```
        takePictureIntent.resolveActivity(packageManager)?.also {
```

```
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE)
```

```
        }
```

```
    }
```

```
}
```

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
```

```
    super.onActivityResult(requestCode, resultCode, data)
```

```
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
```

```
        val imageUri = data?.data
```

```
        if (imageUri != null) {
```

```
            recognizeTextFromImage(imageUri)
```

```
        }
```

```
    }
```

```
}
```

3.3 Overview of SmartReceipts

SmartReceipts is a cutting-edge mobile application designed to streamline personal financial management by leveraging Optical Character Recognition (OCR) technology to automate the entry of financial data. The app enables users to scan receipts and automatically extract key information (such as the total amount, date, and location) while securely storing this data in a Firebase database. SmartReceipts is built to be user-friendly and efficient, eliminating the need for manual data entry and providing users with a comprehensive solution for tracking and analyzing their expenses.

Application Functionality

The application initiates with `ReloadActivity`, which checks if the user is already authenticated. If authenticated, the user is automatically redirected to `MainActivity`; otherwise, they are directed to `LoginActivity`. In `LoginActivity`, users can enter their login credentials or navigate to `RegisterActivity` to create a new account. Upon successful authentication, users are redirected to `MainActivity`. `MainActivity` provides access to various fragments, such as `ShowFragment` (for viewing scanned receipts and analyzing data), `AddFragment` (for adding new receipts), and `ListFragment` (for viewing a detailed list of receipts). `AddFragment` allows users to capture images of receipts, scan them using OCR technology, and save the extracted data to Firebase. `ShowFragment` displays financial data in an interactive graph and detailed list, enabling users to analyze and filter their expenses. `ListFragment` offers a detailed view of all stored receipts, with filtering and search options. From `ListFragment`, users can select a receipt, which redirects them to `ReadFragment`, where all the receipt's details, including the image, are presented.

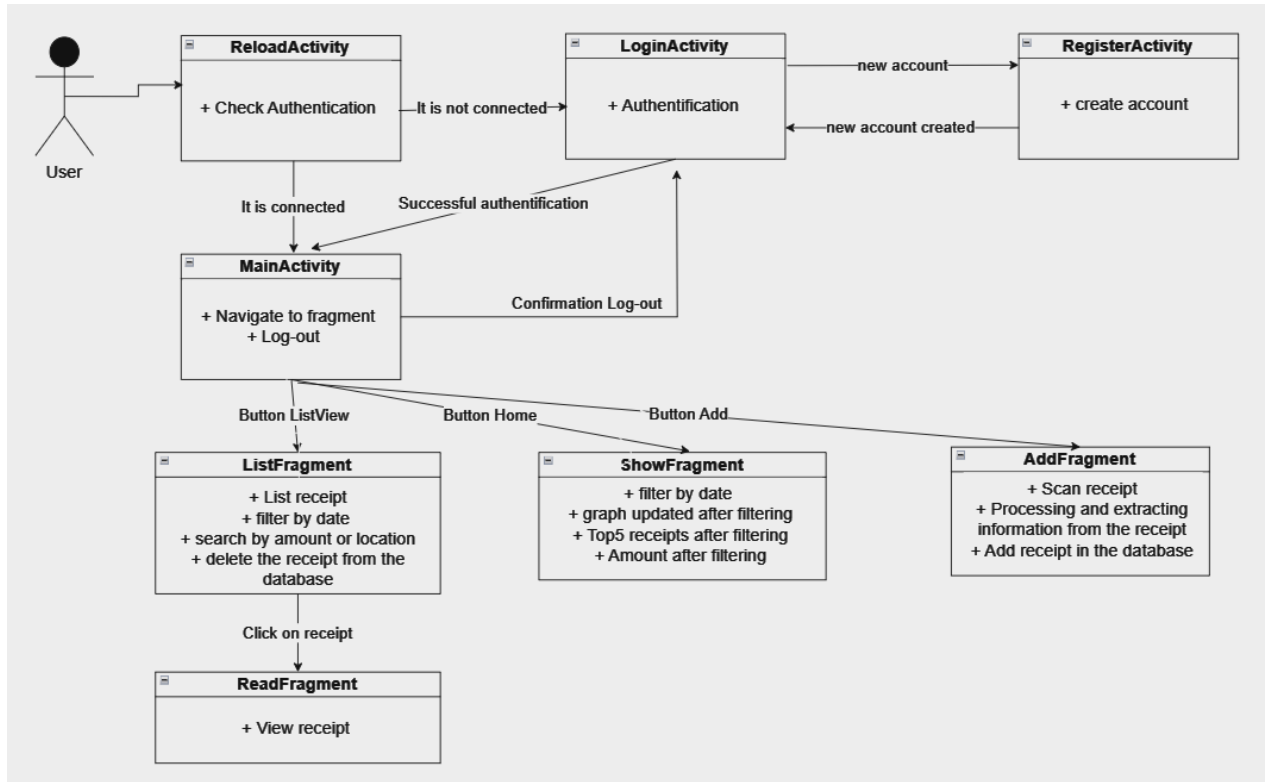


Figure 3.1: Diagram

3.3.1 Pages and Functionality

3.3.1.1 Start Page - "ReloadActivity"

The ReloadActivity in SmartReceipts is designated as the main activity in the manifest. It is responsible for verifying user authentication and redirecting them to either the main screen or the login screen, depending on their session status. This activity functions as a "splash screen," displaying a loading animation and managing the initial user navigation.

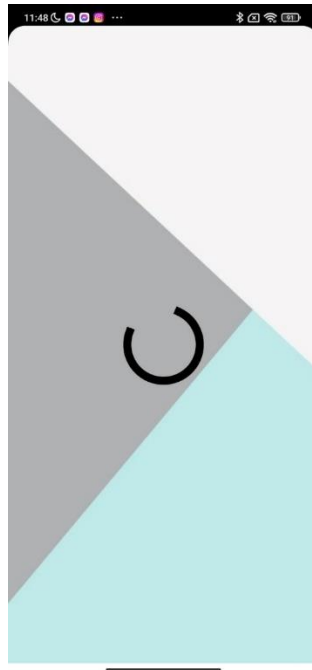


Figure 3.2: ReloadPage

As shown in the Figure 3.4, there is a ProgressBar centered on the screen. This ProgressBar appears to indicate that the application is performing a background check or loading process. In the code, a thread is created to manage the loading animation and to add a brief delay before checking user authentication. The thread is paused for 2 seconds, simulating a loading animation and then a data check.

ReloadActivity is a crucial component of SmartReceipts, managing user authentication status and directing them to the appropriate screen. It ensures a smooth and intuitive experience by accurately routing users based on their authentication state.

Authenticated users are navigated to MainActivity, while unauthenticated users are taken to LoginActivity. By using a thread to handle a short delay and loading animation, this activity enhances the user experience by providing visual feedback during the authentication check.

3.3.1.2 Log-in and Register

Log-in Page Interface

In the SmartReceipts application, LoginActivity serves as the initial access point for users, responsible for managing the authentication process. This activity is crucial for securing access to the app's functionalities, ensuring that only authorized users can access sensitive data and features. LoginActivity is designed to provide a simple and efficient authentication experience, utilizing Firebase Authentication to validate user credentials.

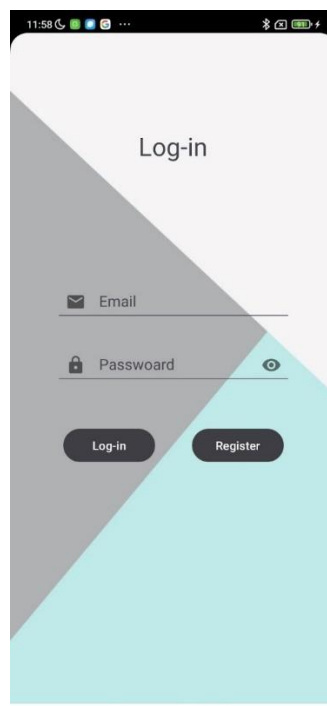


Figure 3.3: Log-in Page

As shown in the Figure 3.5, the main screen of LoginActivity is built using a ConstraintLayout, allowing for flexible and adaptable arrangement of UI elements. The screen background is defined by a custom drawable, giving the application a distinctive appearance.

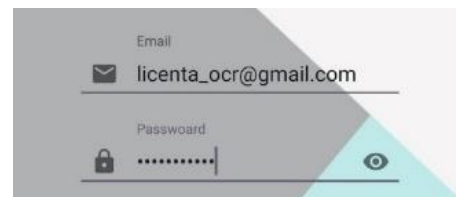
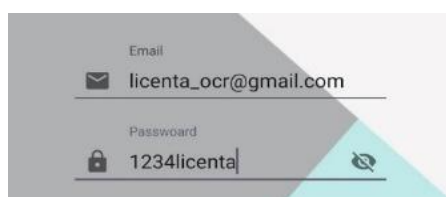


Figure 3.4: Credentials with password visible Figure 3.5: Credentials with password hide

The user interface in LoginActivity is intuitive and easy to navigate, featuring two input fields, each managed by a TextInputLayout that includes a TextInputEditText. The first field is for entering the email address, with an "Email" hint and an email icon on the left. The second field is for entering the password, with a "Password" hint, a password icon, and a toggle function to show or hide the password.

The “Register” button redirects the user to the registration screen, RegisterActivity, and closes LoginActivity to prevent returning to this screen, while the “Log-in” button is essential for user authentication, ensuring a smooth user experience.

By integrating with Firebase Authentication, LoginActivity provides a quick and secure login process, managing both the authentication of existing users and redirecting new users to the registration screen.

Register Page Interface

The RegisterActivity page serves as the entry point for new users, handling the account creation process. This activity is crucial for expanding the application's user base, allowing new users to register and create accounts securely. RegisterActivity is designed to offer a simple and intuitive registration experience, using Firebase Authentication to validate and create new accounts.

Through a user-friendly interface, RegisterActivity ensures that users can quickly and easily complete the necessary information to create an account, including email address and password, and ensures that the entered passwords are correct and match. This activity contributes to a smooth and secure transition to using the application, starting with the registration process.

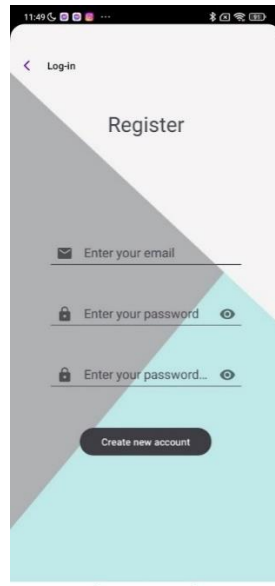


Figure 3.6: Register Page

The `RegisterActivity` interface is similar to that of `LoginActivity`, with an additional `TextInputLayout` containing a `TextInputEditText` for re-entering the password, allowing for its verification. This activity includes a button that directs the user to the login page (`LoginActivity`), closing the current activity to prevent returning to it. There is also a "Create New Account" button responsible for creating a new account using Firebase Authentication.

Authentication Implementation

In `LoginActivity`, Firebase Authentication is used to verify user credentials. When the user enters their email and password and clicks the login button, the `signInWithEmailAndPassword` method from `FirebaseAuth` is called. This checks if the credentials are correct and authenticates the user. Upon successful authentication, the user is redirected to the main screen of the application (`MainActivity`). If authentication fails, an error message is displayed, and the input fields are reset.

In `RegisterActivity`, the process of creating a new account is managed through the `createUserWithEmailAndPassword` method from `FirebaseAuth`. When the user enters their email and password, and these are validated, a new account is created in Firebase Authentication. After the account is successfully created, the user is redirected to `LoginActivity` for authentication.

Both RegisterActivity and LoginActivity are essential components of SmartReceipts, ensuring efficient user access management. By using Firebase Authentication, these activities provide a secure and robust solution for user registration and login. The interfaces are designed to be intuitive and user-friendly, facilitating quick and secure access to the app's features. By integrating these essential activities, SmartReceipts ensures a smooth and secure user experience, contributing to user satisfaction and safety.

3.3.1.3 MainActivity

Within SmartReceipts, MainActivity serves as a central component, acting as the primary interface for users after they log in. This activity is crucial for managing navigation between the app's various fragments and for loading user data from Firebase. MainActivity ensures an intuitive user experience, providing quick and easy access to the app's essential features, such as viewing receipts, adding new ones, and navigating between the app's main sections.

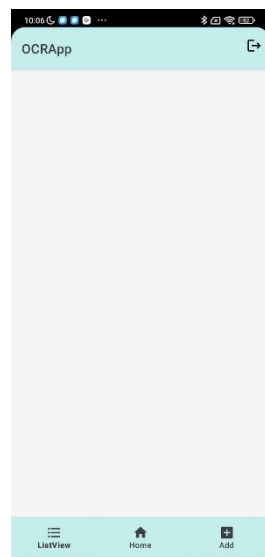


Figure 3.7: Main Page

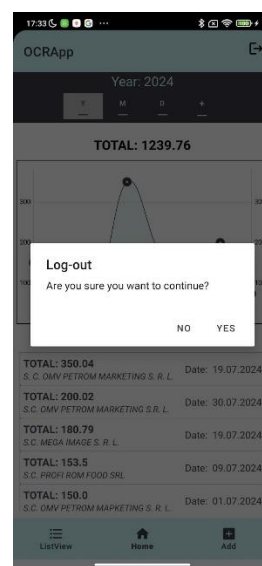


Figure 3.8: Dialog log-out

User Interface Overview

The user interface in MainActivity (Figure 3.9) is designed to be straightforward and easy to navigate, featuring a vertical LinearLayout that includes three main components: a Toolbar, a FragmentContainerView, and a BottomNavigationView. The toolbar, positioned at the top of the screen, includes functionality for logging out and is customized with a blue background to match the app's overall theme. Below the toolbar, the fragment container occupies most of the screen, displaying various fragments based on user interactions. At the bottom of the screen, the BottomNavigationView provides users with quick access to the main sections of the app, facilitating easy and intuitive navigation between the screens for viewing, adding, and listing receipts.

Functionality and Role of MainActivity

MainActivity operates with a multifaceted functionality. First, it manages navigation across the app's sections via the BottomNavigationView, allowing users to seamlessly switch between viewing receipts (Show fragment), adding new ones (Add fragment), and listing existing ones (List fragment). Utilizing a fragment manager, MainActivity updates the fragment container's content according to user selections, ensuring smooth and uninterrupted transitions between screens.

Secondly, MainActivity handles user data management by integrating Firebase Realtime Database. Upon initialization, it checks whether the user's receipt list is empty and, if so, retrieves data from Firebase. This involves listening for database changes and updating the ViewModel with new data, ensuring that users have access to the latest information in real-time, thereby enabling efficient management of their receipts.

Additionally, MainActivity oversees user logout through a menu option in the toolbar. When the logout option is selected, a confirmation dialog is displayed (Figure 3.10), and upon confirmation, the Firebase authentication session is terminated, redirecting the user to the login screen (LoginActivity). This functionality is vital for securing user data by preventing unauthorized access.

In summary, MainActivity in SmartReceipts is pivotal for managing navigation and user data, offering a clear interface and robust features that contribute to a seamless and efficient user experience. By integrating with Firebase and utilizing fragments, MainActivity ensures that users

can quickly and securely access the app's essential features, enhancing overall efficiency and user satisfaction.

3.3.1.4 Fragment Show (Home)

The ShowFragment component in SmartReceipts serves as a crucial part of the application, designed to present users with data and analyses related to their receipts in a clear and easily interpretable manner. This fragment is intended to offer a user-friendly and intuitive interface, enabling users to view their expenses through detailed charts and a list of the most significant receipts.



Figure 3.9: Home Page

User Interface Overview

The user interface for ShowFragment is arranged vertically within a LinearLayout and consists of several key components, see Figure 3.11. At the top, a TextView displays the current month and year, providing context for the presented data. Below this, a MaterialButtonToggleGroup allows users to filter data based on year, month, day, or a selected date range. These buttons are styled to seamlessly integrate with the app's design and offer intuitive navigation between different filtering options.

Further down, another TextView shows the total expenses for the selected period, giving users a quick overview of their total spending. The central element of the interface is a LineChart that illustrates the evolution of expenses over time. This chart is customized for easy interpretation and highlights trends in user spending.

At the bottom of the interface, a TextView introduces the "Top 5 receipts" list, followed by a RecyclerView displaying the five most significant receipts from the selected period. These components are arranged to provide users with detailed information in a structured and easily accessible manner.

ShowFragment Functionality and Role

The functionality of ShowFragment focuses on displaying and filtering user receipt data. The fragment uses ViewModel to access and observe the receipt list stored in Firebase, ensuring that the displayed data is always up-to-date. Upon initialization, ShowFragment loads the current data and configures the user interface to display this information.

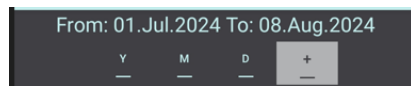


Figure 3.10: Filter date label

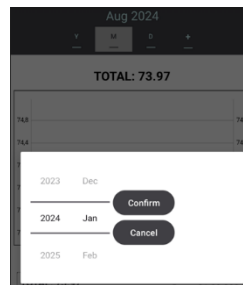


Figure 3.11: Calendar month



Figure 3.12: Calendar date

A key aspect of ShowFragment's functionality is its ability to filter data based on the selected year, month, day, or date range, see Figure 3.12. Users can select different filtering options using the toggle button group, and the fragment will automatically update the displayed data according to the user's selections. For instance, if the user selects a monthly filter, ShowFragment will display the relevant data for the current month and year. This functionality allows users to analyze their expenses in a detailed and personalized way.

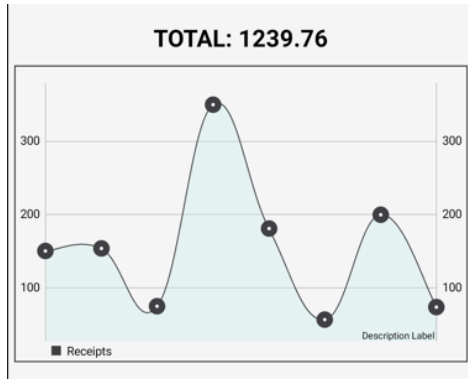


Figure 3.13: View graph receipt after period

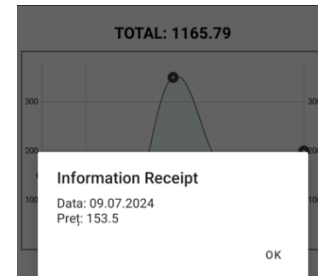


Figure 3.14: Point of graph information

The LineChart within SmartReceipts plays a crucial role in providing a clear and concise visualization of financial data. It allows users to monitor and analyze the evolution of their expenses over time, which is essential for efficient budget management. The chart presents financial data as a line graph, with the X-axis representing time (days, months, or years) and the Y-axis representing the amount spent. This visual representation enables users to observe how their spending has varied over time, making it easier to identify trends and spending patterns. For example, users can quickly spot periods of high or low spending and correlate these fluctuations with specific events or time periods, Figure 3.15.

An important feature of the LineChart is its interactivity, see Figure 3.16. Users can interact with the chart by selecting individual points on the line. When a user selects a point, additional details about the expenses recorded on that specific day are displayed, including the exact amount spent, the purchase date, and other relevant information. This interactive feature not only enhances the user experience but also facilitates a more detailed and in-depth analysis of the data.

In addition, the "Top 5 receipts" list displayed in the RecyclerView provides users with information about their most significant receipts. This feature is useful for quickly identifying the largest expenses and gaining insight into the most costly purchases. The RecyclerView is configured to display this data clearly and structured, ensuring easy navigation between list items.

ShowFragment in SmartReceipts is a central component that significantly enhances the user experience by offering an intuitive interface and robust functionality for viewing and analyzing receipt data. Through the use of well-structured UI components and advanced data filtering and

visualization features, ShowFragment empowers users to efficiently manage and better understand their spending. This fragment exemplifies how a well-designed application can combine attractive design with practical functionality to deliver real value to users.

3.3.1.5 AddFragment

AddFragment within the SmartReceipts is a key component designed to allow users to input new receipts into the application. It features an intuitive interface and advanced capabilities, including image capture, text recognition from images, and data storage in Firebase. The primary purpose of this fragment is to streamline the process of adding receipts, providing users with a straightforward and efficient experience.



Figure 3.15: AddFragmnet

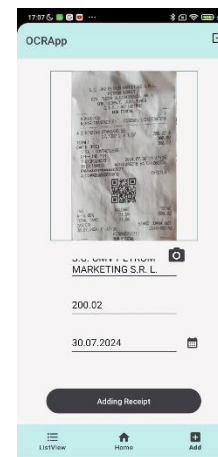


Figure 3.16: AddFragmnet with picture scan

User Interface Overview

The UI for AddFragment is structured using a ConstraintLayout, ensuring a flexible and adaptive arrangement of the UI components, Figure 3.17. At the top of the screen, there is an ImageView that displays the captured receipt image, bordered by an ImageButton which enables users to take new pictures using the device's camera.

Below the ImageView, three EditText fields are provided for entering essential receipt details: total amount, date, and location. These fields are clearly labeled to guide users in entering the

information correctly. Next to the date field, an ImageButton opens a date picker dialog, facilitating quick and accurate date entry.

At the bottom of the screen, a prominent "Adding Receipt" button allows users to save the receipt information to the Firebase database. This button is highly visible and easily accessible, ensuring a simple and intuitive use of the fragment.

Functionality and Role of AddFragment

AddFragment's functionality focuses on capturing and processing receipt images and storing this data in Firebase. Upon initializing the fragment, necessary instances for Firebase are configured, and the UI components are linked to the corresponding variables.

When the user presses the capture button within AddFragment, a document scanner integrated via Google ML Kit is launched. This scanner activates the device's camera, allowing the user to capture an image of the receipt. Once the image is captured, it is processed to extract relevant information.

The process starts by converting the captured image into a format that optical character recognition (OCR) algorithms can analyze. Google ML Kit uses its TextRecognition technology to scan and interpret the text in the image. This technology identifies and extracts text segments, converting them into a readable digital format.

After text recognition, the application uses regular expressions (regex) to analyze and filter the extracted information. Initially, the recognized text is divided into blocks and lines, with each line being evaluated to identify specific patterns. For instance, to extract the total amount, the application searches for text segments that match a currency format, such as a number followed by two decimal places. This is achieved through a regex that recognizes and validates the numerical format.

Similarly, the date is identified by searching for specific date formats, such as day/month/year. Regular expressions are used to locate and validate these formats, ensuring that the extracted dates are correct and consistent. For the location, the application searches for text patterns that match the format of a company name, such as "Company Name Ltd." This involves using regular expressions to detect these patterns and extract them appropriately.

For example:

```
regexSuma = """"d+[.,]\d{1,2}"""
```

```
regexData = """"b(?:0?[1-9]|[12][0-9]|3[01])[,./-](?:0?[1-9]|1[0-2])[,./-]\d{4}\b"""
```

```
regexLocation = """"(?:S[.,]?s*C[.,]?s*)?.*?s*S[.,]?s*R[.,]?s*L[.,]?"""
```

The extracted information total amount, date, and location is then automatically mapped to the corresponding fields in the user interface. This allows the recognized data to be displayed in the relevant text fields, eliminating the need for manual entry by the user. This automation not only improves the efficiency of adding receipts but also reduces the risk of human error.

The "Adding Receipt" button plays a crucial role in validating the user-entered data and initiating the process of uploading the receipt image and information to Firebase. When the user presses this button, the application first checks that all required fields—total amount, date, and location—are correctly filled out. This includes verifying the format of the amount and date using regular expressions, ensuring that the entered values adhere to the expected formats.

If the validation is successful, the application begins uploading the receipt image to Firebase Storage. First, the captured image is uploaded to a directory specific to the current user in Firebase Storage. This is done by creating a reference to the location in Firebase Storage and then uploading the image using this reference. Once the image is successfully uploaded, Firebase Storage generates a unique URL for the image, which can be used to access the stored image.

Once the image URL is obtained, the application creates a Receipt object containing all the relevant receipt information: total amount, date, location, and the image URL. This Receipt object is a complete representation of the receipt data and is ready to be stored in the Firebase Database.

To add the receipt information to the Firebase Database, the application creates a unique reference in the database using an automatically generated key. This unique key ensures that each receipt is stored as a separate record in the database. The Receipt object is then uploaded to this reference, storing all the receipt data in a well-organized and easily accessible structure.

The process of uploading data to Firebase is managed through asynchronous methods, meaning that the application is not blocked during the upload. Status messages are displayed to the user to indicate progress, such as a success message when the receipt is successfully added or an error message if something goes wrong.

AddFragment in SmartReceipts provides an efficient and user-friendly interface for adding receipts. By integrating advanced technologies like text recognition and cloud storage, this fragment ensures a smooth and effective user experience. The robust functionality and intuitive design make AddFragment an essential component for managing receipts in SmartReceipts, helping to keep financial data organized and accessible.

3.3.1.6 ListFragment

The ListFragment plays a critical role in managing and displaying receipts. This component enables users to access and filter their receipts, offering an efficient way to organize and analyze expenses. By utilizing Firebase for data storage and LiveData for real-time user interface updates, ListFragment ensures a seamless and integrated user experience.

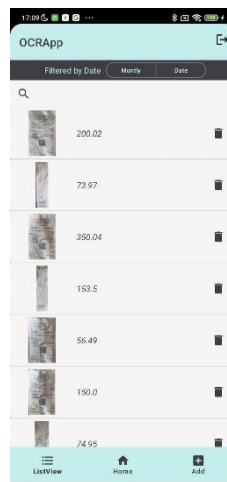


Figure 3.17: ListFragment

User Interface Presentation

The user interface (UI) for the ListFragment is designed for ease of use, see Figure 3.19. At the top of the screen, a horizontal LinearLayout contains a TextView displaying the current filter criteria and a MaterialButtonToggleGroup for selecting the filter type. Users can choose to filter receipts by month or a specific date.

Beneath the horizontal LinearLayout, a SearchView is included, allowing users to search receipts by price or date. The SearchView provides an intuitive and interactive search bar that filters results

in real-time as the user types. This feature is crucial for users who wish to quickly find specific receipts without manually browsing through the entire list.

The main part of the interface is occupied by a RecyclerView that displays the receipt list in a vertical format, with each item styled according to a specific design. Each item includes essential details such as the total amount, date, and location. A delete button is also provided for removing a receipt, accompanied by a confirmation dialog. This modular structure allows for a clear and easily navigable view of receipts.

Functionality and Role of ListFragment

The ListFragment is responsible for displaying and managing the receipts stored by users. Upon initialization, the fragment sets up Firebase and LiveData to retrieve and display receipt data in real-time. The RecyclerView is configured with a LinearLayoutManager to arrange items in a vertical list and with a custom adapter that handles the display of each receipt.

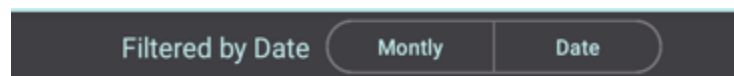


Figure 3.18: Date filter for ListFragment

Users can interact with the UI elements to filter and search receipts, see Figure 3.20. The ToggleButtonGroup allows for quick selection of the desired filter (month or date), triggering the display of a DatePickerDialog or a MonthYearPickerDialog for the relevant period selection. After the filter is chosen, the fragment updates the TextView to reflect the filtering criteria and uses filtering methods to narrow down the displayed list of receipts.

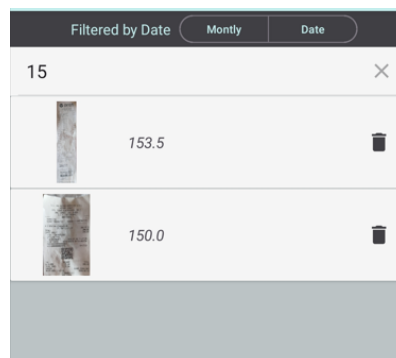


Figure 3.19: SearchView for ListFragment

The SearchView enables searches by price or date, updating the displayed list as the user types in the search text, Figure 3.21. The filtering functionality uses regular expressions to validate and match the search text with receipt data.

ListFragment also implements the AdapterReceipt interface, specifically an OnItemClickListener function, to manage click events on list items. When a user selects a receipt, the fragment replaces the current fragment with the ReadFragment, using supportFragmentManager to navigate between fragments. This allows the user to view the details of the selected receipt.

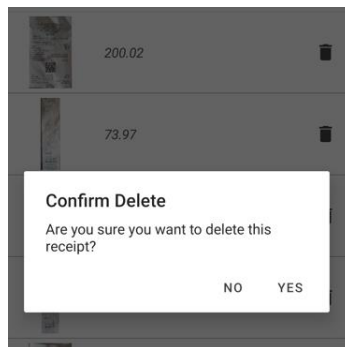


Figure 3.20: Confirmation Dialog from the delete button

Additionally, ListFragment includes functionality for deleting receipts, Figure 3.22. When the user clicks the delete button associated with a receipt, the fragment removes the receipt from the Firebase Database and, if applicable, deletes the associated image from Firebase Storage. This is handled through asynchronous methods to ensure the application remains responsive during network operations.

In conclusion, ListFragment is a vital component of the SmartReceipts, providing users with the necessary tools to efficiently manage and analyze their receipts. The use of Firebase and LiveData ensures real-time data updates, offering a smooth and integrated experience for users.

3.3.1.7 ReadFragment

Within the SmartReceipts, ReadFragment is responsible for displaying the full details of a selected receipt. This fragment provides users with a detailed view of the receipt, including

The image shows a mobile application interface for OCR (Optical Character Recognition). At the top, the status bar shows the time 17:10 and various icons. The app's header is light blue with the text 'OCRApp' and a share icon. Below the header, the OCR result is displayed as a receipt from 'S.C. OMV PETROM MARKETING S.R.L.'. The receipt text includes: 'TOTAL: 200.02', '30.07.2024', 'S.C. OMV PETROM MARKETING S.R.L.', 'PETROM IERNUT', 'STR. TUDOR VLADIMIRESCU NR.1', 'ORS. IERNUT, J.D. MURES', 'C. I. P. 1 RD1281961', 'ROM FISCAL', 'NUMAR POS: 1', 'NUMAR TRANZACTIE: 1330301/1/240730/210', '* 2 BENZINA STANDARD B5', '27,1397 L X 7,37', '200.02 A', 'TOTAL: 200.02', 'CARTE CREDIT', 'VISA - CONTACTLESS', 'OFF-LINE PIN', 'TID:0R020629', '2024.07.30 11:47:35', 'TXID:154519', 'AUTH:406316', 'BATCH:003754', 'PAN:*****2627', 'ATD:A0000000031010', 'EN:CTLS', a QR code, 'TVA A=10.00%', 'VALDARE 31,94', 'TOTAL 200.02', 'TOTAL TAXE: 31,94', 'CASSIER: 30.07.2024 11:47:39', 'SIZARD: IDANA 001', '30.07.2024 11:47:39', '2549-00170', 'FINANZ:11711', and 'ROM FISCAL'. At the bottom, there is a navigation bar with three icons: 'ListView', 'Home', and 'Add'.

User Interface Presentation

The main user interface is built using a `ConstraintLayout` that spans the entire screen, containing a vertically oriented `LinearLayout`. This `LinearLayout` hosts several subcomponents, arranged to provide a clear and easily navigable view of the receipt details.

At the top of the screen, a horizontal `LinearLayout` includes a `TextView` displaying the Total Amount, highlighted with larger, bold text, another `TextView` showing the Location of the transaction, styled in italics for distinction, and a Date `TextView` positioned to the right of the `LinearLayout`, featuring medium-sized text for visibility.

Below the horizontal `LinearLayout`, an `ImageView` occupies the remaining screen space to display the receipt image, shown in a large size to allow users to clearly view the receipt details.

Functionality and Role of ReadFragment

ReadFragment is responsible for displaying the details of a specific receipt selected from the list. Upon initialization, the fragment receives a Receipt object via a Bundle, containing all necessary information about the receipt. These details are then displayed in the corresponding UI elements.

Using the Glide library, the receipt image is loaded from the URL stored in the Receipt object. Glide provides an efficient and straightforward way to load and display images from network sources, ensuring a smooth user experience. The receipt image is loaded asynchronously using Glide, which effectively manages memory and performance during the image loading process. If the image cannot be loaded, Glide displays a fallback image, ensuring that the UI remains complete and visually consistent.

ReadFragment provides an efficient and clear method for viewing receipt details in the SmartReceipts. Through the use of an intuitive UI design and efficient libraries like Glide, the fragment ensures an optimized user experience, allowing users to quickly access and view essential receipt information. This contributes to better expense management and detailed transaction analysis.

3.4 Application Testing

The SmartReceipts was tested using a manual approach. Test cases were designed based on the possible navigation paths users might follow within the app. The aim was to verify all major functionalities and ensure a smooth, error-free user experience.

This manual testing approach allowed us to identify and fix any potential issues, ensuring the SmartReceipts functions correctly and efficiently.

INPUT	EXPECTED	ACTUAL RESULT
PreCondition: Start from the Login page.		OK

1. Enter a valid email and password and Click the "Login" button.	1. The MainActivity page is rendered.	
PreCondition: Start from the Login page. 1. Enter an invalid email and password and Click the "Login" button.	1. An error message is displayed indicating invalid credentials.	OK
PreCondition: Start from the Register page 1. Enter a valid email, password, and confirm password. 2. Click the "Create new account" button.	1. The account is created. 2. The Login page is rendered.	OK
PreCondition: Start from the Register page. 1. Enter an invalid email or mismatched passwords. 2. Click the "Create new account" button.	1. An error message is displayed indicating the issue.	OK
PreCondition: Go to the MainActivity page. 1. Click the "Logout" button from the menu.	1. The user is logged out. 2. The LoginActivity page is rendered.	OK
PreCondition: Go to the MainActivity page. 1. Click the "Add" button in the bottom navigation. 2. Capture or select an image of a receipt. 3. Enter the total, date, and location. 4. Click the "Add Receipt" button.	1. The receipt is added to the database. 2. The receipt is displayed in the list.	OK
PreCondition: Go to the ListFragment page. 1. Use the search bar to enter a date or total amount.	1. The list is filtered to show receipts that match the search criteria.	OK
PreCondition: Go to the ListFragment page 1. Use the toggle buttons to filter by month or date.	1. The list is filtered to show receipts for the selected month or date.	OK
PreCondition: Go to the ListFragment page 1. Click on a receipt item in the list.	1. The ReadFragment page is rendered. 2. The details of the selected receipt are displayed.	OK

PreCondition: Go to the ShowFragment page 1. Select a year, month, or date from the toggle buttons. 2. Observe the LineChart for the selected period.	1. The LineChart updates to display the data for the selected period. 2. Data points are interactive and display details on selection.	OK
PreCondition: Go to the ShowFragment page. 1. Select a specific range of dates. 2. Observe the total sum and top 5 expenses.	1. The total sum and top 5 expenses update to reflect the selected date range.	OK

3.5 User Experience and Feedback

Understanding the user experience is crucial for refining and enhancing the functionality of any application. To evaluate the effectiveness and user satisfaction of the SmartReceipts app, a survey was conducted, gathering insights from users regarding various aspects of the app's performance. This section presents the feedback obtained from the users, highlighting their opinions on the design, ease of navigation, utility in managing personal expenses, and the effectiveness of specific features such as receipt scanning, filtering, and interactive chart creation. The feedback collected not only provides a snapshot of the current user satisfaction but also offers valuable suggestions for future improvements and feature enhancements. The results, as illustrated in the following charts, reflect the overall reception of the SmartReceipts app and guide the direction for future developments

1. Do you find SmartReceipts useful for effectively managing personal expenses?
13 răspunsuri

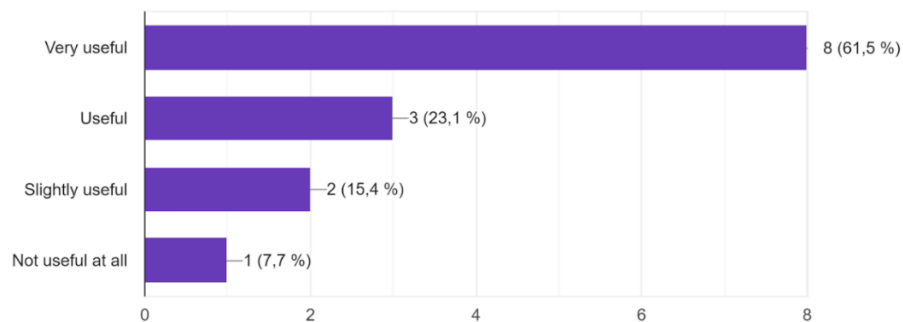


Figure 3.22

2. Is the design of the SmartReceipts app attractive and pleasing?

13 răspunsuri

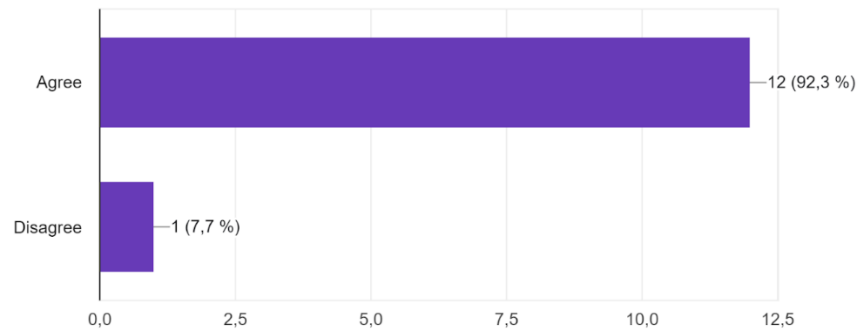


Figure 3.23

3. How easy and intuitive is it to navigate through the SmartReceipts app?

13 răspunsuri

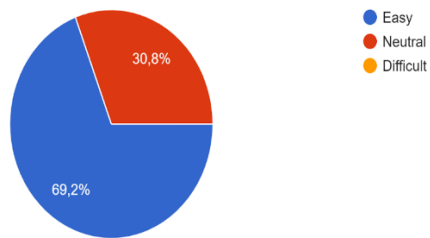


Figure 3.24

4. Is the filtering feature for receipts in SmartReceipts effective ?

13 răspunsuri

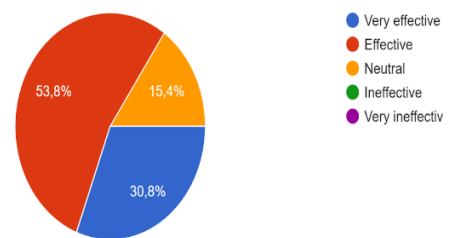


Figure 3.25

5. Which feature do you think is the best in the app?

13 răspunsuri

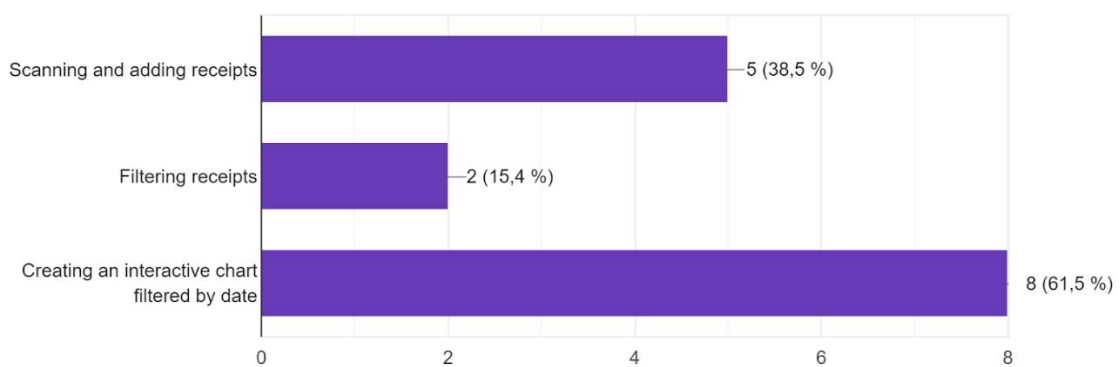
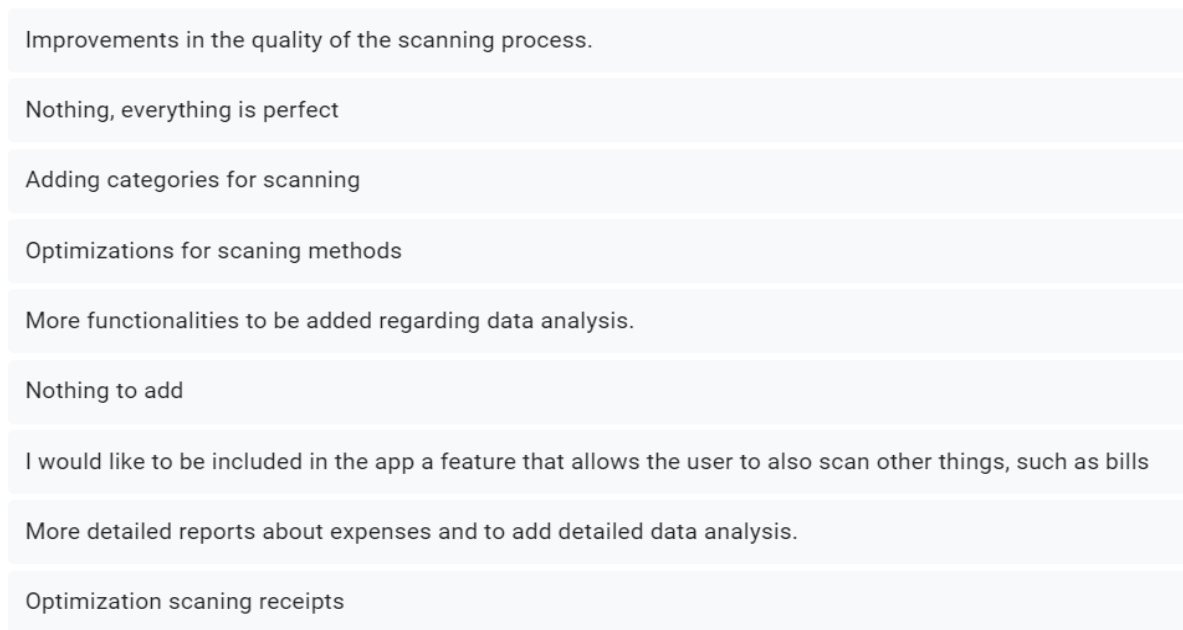


Figure 3.26

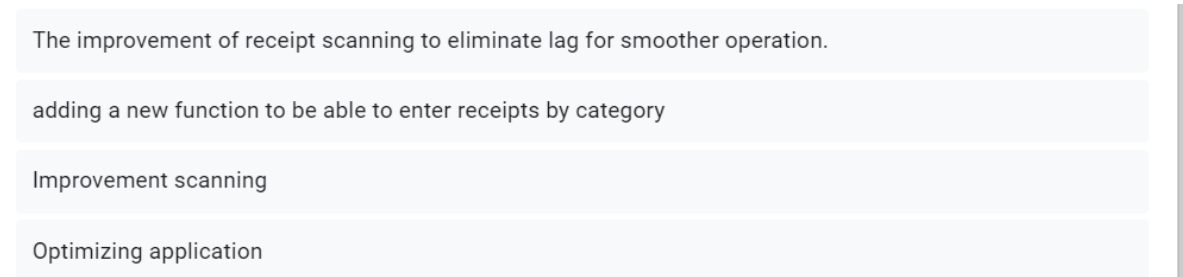
6. What could be improved in the app, and what future features would you like to see added?

13 răspunsuri



Improvements in the quality of the scanning process.
Nothing, everything is perfect
Adding categories for scanning
Optimizations for scanning methods
More functionalities to be added regarding data analysis.
Nothing to add
I would like to be included in the app a feature that allows the user to also scan other things, such as bills
More detailed reports about expenses and to add detailed data analysis.
Optimization scanning receipts

Figure 3.27



The improvement of receipt scanning to eliminate lag for smoother operation.
adding a new function to be able to enter receipts by category
Improvement scanning
Optimizing application

Figure 3.28

The insights gathered from user feedback have provided a clear picture of how the SmartReceipts app is perceived in terms of design, usability, and functionality. Overall, the app has been well-received, with users appreciating its core features, such as receipt scanning, filtering, and the creation of interactive charts. However, the feedback also highlighted areas where further improvements can be made, particularly in optimizing the scanning process and expanding the app's functionality.

The suggestions and recommendations provided by users will serve as a foundation for the continued development of the SmartReceipts app. The final image, which captures user input on desired features and enhancements, directly informs the direction for the next phase of development. These insights will be integrated into the "Future Work" chapter, where potential upgrades and new features will be explored to further enhance the app's effectiveness and user satisfaction.

3.6 Future Work and Potential Enhancements for SmartReceipts

As SmartReceipts continues to develop, there are several avenues for future work and potential improvements that could enhance its functionality, user experience, and overall performance. This section highlights some of the key areas that could be explored to ensure that SmartReceipts remains a robust and user-friendly application.

Enhancing OCR Accuracy: One major area for improvement in SmartReceipts is the accuracy of text recognition. While the current implementation using Google ML Kit provides satisfactory results, there is always room for improvement. Future work could involve integrating more advanced machine learning models, such as those based on deep learning techniques, to increase the accuracy and reliability of text extraction from receipts. Additionally, implementing a feedback mechanism where users can correct incorrectly recognized text could help train the model and improve its performance over time.

Expanding Receipt Types and Adding Invoice Support: Currently, SmartReceipts is optimized for standard receipts. Expanding the application's capabilities to recognize and process various types of receipts, including those from different countries with diverse formats and languages, would significantly enhance its usability. Moreover, adding support for utility bills, such as electricity, gas, TV, and other services, would be a valuable addition. These documents typically have specific formats and critical information that need to be accurately extracted, necessitating the development of specialized algorithms for each type of invoice.

Advanced Filtering and Analysis Features: In the future, SmartReceipts could offer advanced filtering and analysis features to help users better manage their expenses. For instance,

integrating detailed charts and reports that show spending trends over different time periods or categories could provide users with deeper insights into their finances.

Optimizing Application Performance: As SmartReceipts evolves and new features are added, optimizing the application's performance becomes essential. Ensuring a fast and smooth user experience, reducing load times, and optimizing device resource usage will contribute to overall user satisfaction.

As SmartReceipts continues to evolve, focusing on enhancing OCR accuracy, expanding its capability to handle various receipt types and invoices, and introducing advanced filtering and analysis features will be crucial for maintaining its relevance and effectiveness. Optimizing performance to ensure a seamless user experience will further contribute to the app's success. By addressing these areas, SmartReceipts can become a powerful and user-friendly tool for managing personal finances, providing users with valuable insights and an improved experience.

CONCLUSION

Given the significance of efficiently managing personal finances in today's society, it is clear that the development of technological solutions for money management should not be overlooked.

SmartReceipts is a contribution to this crucial area. SmartReceipts was created with the goal of helping users to efficiently and intuitively scan and manage receipts, utilizing advanced optical character recognition (OCR) techniques. The application allows users to quickly capture and store essential information from receipts, eliminating the need for manual data entry and reducing the likelihood of human error. By integrating with Google ML Kit, SmartReceipts provides accurate and reliable text recognition, while using Firebase for data storage and authentication ensures a high level of security and accessibility. The app enables users to track and analyze their daily expenses, giving them better visibility and control over their personal finances.

Further development of this application and expansion of its features, such as including utility bills for electricity, gas, TV, and other services, could provide significant benefits to users. Additionally, incorporating gamification concepts could encourage users to engage more actively in managing their finances, turning a routine task into a more enjoyable and motivating experience.

Through the development of SmartReceipts, deeper understanding was gained of the importance of effective financial management in everyday life. I strongly believe that software solutions dedicated to this field can play a vital role in facilitating this process and educating users on responsible money management.

In conclusion, SmartReceipts demonstrates how modern technology can be leveraged to transform and optimize daily processes, providing users with a powerful and efficient tool for managing expenses. By continuing to innovate and improve, SmartReceipts has the potential to become an essential part of users' daily lives, contributing to better financial organization and time savings.

Bibliography

- [1] **Lusardi, A., & Mitchell, O. S. (2014).** *The Economic Importance of Financial Literacy: Theory and Evidence*. Journal of Economic Literature, 52(1), 5-44. doi:10.1257/jel.52.1.5
- [2] **Huston, S. J. (2010).** *Measuring financial literacy*. Journal of Consumer Affairs, 44(2), 296-316. doi:10.1111/j.1745-6606.2010.01170.x
- [3] **Atkinson, A., & Messy, F.-A. (2012).** *Measuring financial literacy: Results of the OECD/International Network on Financial Education (INFE) pilot study*. OECD Publishing. doi:10.1787/5k9csfs90fr4-en
- [4] **Lusardi, A., & Tufano, P. (2015).** *Debt literacy, financial experiences, and overindebtedness*. Journal of Pension Economics & Finance, 14(4), 332-368. doi:10.1017/S1474747215000232
- [5] **Consumer Financial Protection Bureau (CFPB). (2017).** *Financial Well-Being: The Goal of Financial Education*. Retrieved from <https://www.consumerfinance.gov> in 06.05.2024
- [6] **Fernandes, D., Lynch, J. G., & Netemeyer, R. G. (2014).** *Financial literacy, financial education, and downstream financial behaviors*. Management Science, 60(8), 1861-1883. doi:10.1287/mnsc.2013.1849
- [7] **National Bureau of Economic Research (NBER). (2019).** *The Role of Financial Literacy in Financial Decision Making*. Retrieved from <https://www.nber.org> in 12.05.2024
- [8] **Garman, E. T., & Forgue, R. (2011).** *Personal finance*. Cengage Learning.
- [9] **JPMorgan Chase Institute. (2019).** *The evolution of financial tracking apps: How they help consumers save*. Retrieved from <https://www.jpmorganchase.com> in 12.05.2024
- [10] **Mint. (2021).** *Mint: Your Money, Effortlessly Organized*. Retrieved from <https://www.mint.com> in 03.06.2024
- [11] **YNAB (You Need A Budget). (2020).** *The Benefits of Budgeting: YNAB Users' Savings Statistics*. Retrieved from <https://www.youneedabudget.com> in 03.06.2024

- [12] **Personal Capital. (2021).** *Personal Capital: Manage and Grow Your Wealth*. Retrieved from <https://www.personalcapital.com> in 03.06.2024
- [13] **PocketGuard. (2023).** *PocketGuard: Personal Finance and Budgeting App*. Retrieved from <https://pocketguard.com> in 03.06.2024
- [14] **Goodbudget. (2023).** *Goodbudget: Budgeting & Personal Finance App*. Retrieved from <https://goodbudget.com> in 03.06.2024
- [15] **Wally. (2023).** *Wally: Smart Personal Finance*. Retrieved from Google Play Store or Apple App Store.
- [16] **ViewModel** Overview. Retrieved from <https://developer.android.com/topic/libraries/architecture/viewmodel> in 18.06.2024
- [17] **Android Studio** Overview. Retrieved from <https://developer.android.com/studio/intro> in 18.06.2024
- [18] **Managing the Activity Lifecycle**. Retrieved from <https://developer.android.com/guide/components/activities/activity-lifecycle> in 18.06.2024
- [19] **Fragments**. Retrieved from <https://developer.android.com/guide/components/fragments> in 18.06.2024.
- [20] **RecyclerView**. Retrieved from <https://developer.android.com/guide/topics/ui/layout/recyclerview> in 18.06.2024.
- [21] **LiveData** Overview. Retrieved from <https://developer.android.com/topic/libraries/architecture/livedata> in 18.06.2024.
- [22] **Adapter**. Retrieved from <https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter> in 18.06.2024.
- [23] **ViewHolder**. Retrieved from <https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.ViewHolder> in 18.06.2024.
- [24] **Firebase** documentation. <https://firebase.google.com/docs> in 23.06.2024.

- [25] **Firestore** authentication. <https://firebase.google.com/products/auth> in 23.06.2024.
- [26] **Firestore** realtime database. <https://firebase.google.com/products/realtime-database> in 23.06.2024.
- [27] **Firestore** storage. <https://firebase.google.com/products/storage> in 23.06.2024.
- [28] **ML Kit** documentation. <https://developers.google.com/ml-kit> in 04.07.2024.
- [29] **ML Kit** text recognition. <https://developers.google.com/ml-kit/vision/text-recognition> in 04.07.2024
- [30] **Implementing OCR in your app**. <https://firebase.google.com/docs/ml-kit/android/recognize-text> in 04.07.2024