# SEA College of Science, Commerce and Arts

A internship report submitted in partial fulfillment of the requirement for Bachelor of Computer Application Degree, Sixth under Bengaluru North University during the academic year 2023-2024

Internship Report on **Money management** at

**DQ TECHNOLOGIES**.

Ernakulam,Kerala

Submitted in Partial Fulfillment of the Requirements of Bachelor of Computer Application Degree of Bengaluru North University ***By***

# IBRAHIM KHALEEL.A

# U19UU21S0070

# Under the guidance of

# Mrs.Ramya M

# Assistant Professor

Department of Computer Science

S.E.A College of Science, Commerce and Arts KR Puram,Bengaluru– 560036.



Department of Computer Science

S.E.A College of Science, Commerce and Arts KR Puram, Bengaluru– 560036.

2023-2024

**CERTIFICATE**

This is to certify that **SIBRAHIM KHALEEL A** bearing Registered No. **U19UU21S0070**. is a student of VI Semester Bachelor of Computer Application of our College

He has prepared Internship report entitled “**Money management at**”, **DQ TECHNOLOGY**. **15/06/2024** to **30/06/2024** towards the partial

fulfillment of the requirement of Bachelors of Computer Application Degree of Bengaluru North University.

**Guide Name : HOD Name: Principal Name:**

**Signature Signature Seal & Signature**

STUDENT DECLARATION

I’m **IBRAHIM KHALEEL.A,** Register Number: **U19UU21S0070**, hereby declare that this report entitled **DQ TECHNOLOGIES ERNAKULAM, KERALA** during the internship period from **15/06/2024** to **30/06/2024** at **DQ TECHNOLOGIES**. under the supervision and guidance of **RAMYA M**, **Head of Department** of Computer Science, S.E.A College of Science, Commerce and Arts, K R Puram, Bengaluru.

Date: Signature

Place:

Name: Ibrahim Khaleel.A Reg No: U19UU21S0070

0

## ACKNOWLEDGEMENT

The successful completion of this internship report required significant guidance and assistance from many individuals, and I am truly grateful for their support throughout this journey.

Firstly, I would like to express my sincere appreciation to Sri/Mr. **Mhummed Syed PM** ,DQ technologles**,** for providing me with the opportunity to intern at their esteemed organization.

I am also deeply grateful to our principal, **Dr. Muthe Gowda T N**, for their unwavering support and for granting me the valuable opportunity to perform the Internship on stage I also express my sincere thanks to guide M**hummed Syed PM,** for his/her valuable guidance and timely suggestion at every stage of this project.

I would like to extend my heartfelt thanks to my parents for their permission and constant encouragement throughout this internship.

Additionally, I am thankful to my friends for their support whenever I needed their assistance during this project.

Lastly, I would like to express my profound gratitude to all individuals who directly or indirectly contributed to the completion of this report.

|  |  |  |
| --- | --- | --- |
| SL No. | Topic | Page No. |
| 01. | Introduction | 08 |
| 02. | About Company | 11 |
| 03. | Software Development usage | 13 |
| 04. | System design and Analysis | 22 |
| 05. | Flutter as the Programming Framework | 45 |
| 06. | Learning outcomes | 48 |
| 07. | Bibliography | 53 |
| 08 | Articles and Online Resources | 55 |

**ABSTRACT**

Money Management App Revolutionizing Personalized personal finances Based Software

In today's fast-paced world, managing personal finances effectively has become increasingly important. The proposed mobile application, developed using the Flutter framework, aims to simplify financial management by providing a comprehensive platform for tracking cash transactions, monitoring income, and managing expenses. The app is designed to be intuitive and user-friendly, catering to individuals who seek to gain better control over their financial activities.

Key features of the application include a seamless interface for logging cash transactions, categorizing them as either income or expenses. Users can easily record their daily transactions, add relevant details, and categorize them under predefined or custom categories. The app provides a real-time overview of the user's financial status, allowing them to track their spending habits, set budgets, and analyze trends over time.

Additionally, the app supports the generation of detailed reports and charts, enabling users to gain insights into their financial behavior. With a focus on simplicity and efficiency, the Flutter-based application ensures smooth performance across various devices, making it accessible to a wide range of users.

By leveraging the power of Flutter, the application offers cross-platform compatibility, ensuring that users can manage their finances on both Android and iOS devices. The proposed app not only aims to enhance financial literacy but also empowers users to make informed decisions about their money, ultimately leading to improved financial well-being.

.

**INTRODUCTION**

In the era of digital transformation, managing personal finances efficiently has become a critical aspect of modern life. With the proliferation of smartphones and the increasing reliance on mobile applications for day-to-day tasks, there has been a growing demand for tools that can assist individuals in tracking and managing their financial activities. The money management app developed as part of this internship project addresses this need by providing users with a comprehensive platform to monitor their income, manage expenses, and gain insights into their financial behavior. This project not only aligns with the current trend towards digitization but also aims to empower users by providing them with the tools necessary to achieve financial stability and literacy.

The primary objective of the money management app is to create a user-friendly and efficient platform that simplifies the process of financial management. Traditional methods of tracking expenses, such as manual ledger entries or the use of spreadsheets, are often cumbersome and prone to errors. This app aims to eliminate these challenges by offering an intuitive interface where users can effortlessly record their income and expenses. By utilizing the Flutter framework, the app is designed to be cross-platform, ensuring that it functions seamlessly on both Android and iOS devices. This cross-platform capability is crucial in today’s market, where users expect a consistent experience regardless of the device they are using.

The app’s functionality extends beyond basic income and expense tracking. It allows users to categorize transactions, making it easier to analyze spending patterns and identify areas where they can save. For instance, users can categorize expenses into different groups such as food, transportation, entertainment, and utilities. This categorization provides a clear picture of where money is being spent and helps users make informed decisions about their finances. The app also supports the setting of budgets for different categories, enabling users to set financial goals and monitor their progress towards achieving them. By providing real-time feedback on spending habits, the app encourages users to develop better financial practices.

The proposed mobile application, developed using the Flutter framework, aims to simplify financial management by providing a comprehensive platform for tracking cash transactions, monitoring income, and managing expenses. The app is designed to be intuitive and user-friendly, catering to individuals who seek to gain better control over their financial activities.

Key features of the application include a seamless interface for logging cash transactions, categorizing them as either income or expenses. Users can easily record their daily transactions, add relevant details, and categorize them under predefined or custom categories. The app provides a real-time overview of the user's financial status, allowing them to track their spending habits, set budgets, and analyze trends over time.

Additionally, the app supports the generation of detailed reports and charts, enabling users to gain insights into their financial behavior. With a focus on simplicity and efficiency, the Flutter-based application ensures smooth performance across various devices, making it accessible to a wide range of users.

By leveraging the power of Flutter, the application offers cross-platform compatibility, ensuring that users can manage their finances on both Android and iOS devices. The proposed app not only aims to enhance financial literacy but also empowers users to make informed decisions about their money, ultimately leading to improved financial well-being.

Another significant feature of the app is its ability to generate detailed reports and visual representations of financial data. Users can access charts and graphs that depict their income and expenses over time, helping them to identify trends and make data-driven decisions. For example, users can compare their monthly expenses and identify any irregularities or patterns that might require attention. This feature is particularly useful for individuals who need to prepare for financial audits or for those who simply want to have a clearer understanding of their financial health. The ability to export these reports also adds a layer of convenience for users who may need to share their financial information with advisors or for tax purposes.

The development of this app was guided by the principles of simplicity, efficiency, and accessibility. The user interface was designed with a focus on ease of use, ensuring that even individuals with limited technical knowledge can navigate the app with ease. The app’s architecture is built to handle a large volume of transactions without compromising on performance, making it suitable for users with varying levels of financial activity. Moreover, the use of Flutter for development ensures that the app is not only responsive but also maintains a consistent user experience across different devices and screen sizes.

In conclusion, the money management app developed as part of this internship project represents a significant step towards simplifying personal financial management. By leveraging modern technology and adhering to best practices in app development, the project successfully addresses the common challenges faced by individuals in managing their finances. The app’s user-centric design, coupled with its robust functionality, makes it a valuable tool for anyone looking to take control of their financial future. This project not only demonstrates the practical application of Flutter in creating cross-platform applications but also highlights the importance of developing tools that cater to the evolving needs of users in a digital world

**DQ Technologies**

DQ technologies is a professionally competent IT company based in india. We are providing comprehensive platform that supports overall software solutions like Mobile applications development and Web developments and Device integration, etc.

DQ team has the best talents in, to offer its clients the best solutions. The dq technologies team of software engineers seeks new and challenging assignments in Business solutions, with the belief that quality and punctuality is the essence of software development

Looking to take your business to the next level? Look no further than DQ tech! Our expert team is here to help you boost the effectiveness of your promotions and polish your branding, ensuring that your business stands out in a crowded digital marketplace.

With years of experience and a customized approach to every client, we’re the perfect partner for businesses of all sizes. Contact us today and see the results for yourself!

**Web development**

Services from Professional web design & web development company, customized website or application development is our expertise. We also have in-depth experience in developing websites on different platforms & technology for business, e-commerce, portal, or any customized applications.

**App development**

We have in-depth experience and expertise in developing high-end Mobile apps for Android and iOS with the customization and user-friendly functionality. The Performance of the Mobile app is the most important parameter, we deliver with the understanding that it is the latest tool to expand

**SEO optimization**

Gain a competitive edge by utilizing the best SEO services, comprising both on-page and off-page optimization. SEO is a combination of knowledge, experience, and technique, and our team of SEO experts can get better, faster, and more sustainable results for your compan

### Software Development usage

### 

****Flutter as the Programming Framework:****

Flutter, developed by Google, has emerged as a revolutionary framework for building cross-platform applications with a single codebase, transforming the landscape of mobile and web development. As a UI toolkit, Flutter offers a powerful and efficient way to create natively compiled applications for mobile, web, and desktop from a single codebase. This cross-platform capability is one of Flutter's most compelling features, enabling developers to write code once and deploy it across multiple platforms, including Android, iOS, Windows, macOS, and even web browsers. This not only accelerates the development process but also ensures consistency in user experience across different devices. Flutter’s architecture is based on the Dart programming language, which is specifically designed for fast performance and provides a smooth development experience with its hot reload feature. Hot reload allows developers to instantly see the results of changes in the code without restarting the entire application, significantly speeding up the iterative development process. The core of Flutter’s power lies in its rich set of pre-designed widgets, which can be customized extensively to create beautiful, responsive user interfaces. Unlike other frameworks that rely on native components, Flutter renders everything using its high-performance rendering engine, Skia, ensuring that applications look the same on every platform. This rendering process provides a high degree of control over every pixel on the screen, making it possible to achieve complex UI designs that are often difficult to implement with other frameworks. Moreover, Flutter’s widget-based architecture promotes the reusability of components, leading to cleaner and more maintainable code. This modular approach not only reduces development time but also simplifies the process of making updates and enhancements. The Flutter framework also includes a comprehensive set of tools and libraries that support various functionalities such as animations, gestures, and internationalization, making it a versatile choice for developing feature-rich applications. Additionally, Flutter's integration with popular backend services and APIs is seamless, allowing developers to easily incorporate functionalities such as cloud storage, real-time databases, and user authentication into their apps. The community around Flutter is vibrant and growing rapidly, with extensive documentation, tutorials, and a wealth of third-party packages available through the Flutter ecosystem. This active community support ensures that developers have access to the resources they need to overcome challenges and implement best practices. In the context of the money management app developed during this internship, Flutter proved to be an ideal choice due to its ability to deliver a consistent user experience across both Android and iOS platforms. The framework’s efficient handling of complex UIs, combined with its fast performance, ensured that the app could manage real-time financial data and provide users with a seamless and responsive interface. Furthermore, Flutter's scalability means that the app can easily be expanded with new features and functionalities as user needs evolve. The choice of Flutter as the programming framework was instrumental in achieving the project’s goals, offering a robust, flexible, and future-proof foundation for the development of a high-quality money management application

****Dart Programming Language:****

Dart is a modern, object-oriented programming language developed by Google, designed to be optimized for client-side development, particularly for building mobile, desktop, and web applications. Dart's syntax is clear and concise, making it accessible to both novice and experienced developers. It supports both ahead-of-time (AOT) and just-in-time (JIT) compilation, providing the dual benefits of fast startup times and high-performance execution. This versatility makes Dart an excellent choice for developing dynamic and responsive applications. One of Dart's standout features is its robust typing system, which allows for both static and dynamic typing. This hybrid approach offers the flexibility of dynamic languages like JavaScript, while also providing the safety and reliability of statically typed languages such as Java or C#. Furthermore, Dart's asynchronous programming model, which includes support for Futures and Streams, makes it particularly well-suited for handling real-time data and complex user interactions, a crucial requirement for modern mobile applications. Dart also has a comprehensive standard library, which includes a rich set of core libraries and packages that simplify common programming tasks, such as working with collections, processing asynchronous tasks, and managing data formats like JSON. In the context of this internship project, Dart played a pivotal role in the development of the money management app, enabling the creation of a responsive and efficient application that can manage complex financial transactions in real-time. The language’s strong support for object-oriented programming principles facilitated the creation of reusable components, contributing to a cleaner and more maintainable codebase. Additionally, Dart’s integration with Flutter provided access to a wide range of customizable widgets, ensuring a smooth and visually appealing user experience across multiple platforms. Dart's seamless integration with Flutter, along with its powerful features, made it the ideal choice for building the money management app, ensuring that it met the project’s performance and

****Hive Database:****

Hive is a lightweight, fast, and efficient NoSQL database written in pure Dart, specifically designed for Flutter and Dart applications. It is widely recognized for its excellent performance on mobile devices, making it a popular choice for developers who require an offline-first storage solution with minimal overhead. Hive’s architecture is optimized for speed, with data being stored in binary format, allowing for rapid read and write operations, even on large datasets. Unlike traditional databases that rely on complex schemas and heavy setup processes, Hive is schema-less, enabling developers to store data in a more flexible and dynamic manner. This is particularly advantageous in mobile app development, where the ability to quickly adapt to changing data structures without extensive refactoring is crucial. Hive supports complex data types, such as lists and maps, and allows developers to store custom objects through the use of TypeAdapters, which serialize and deserialize objects for storage. One of Hive’s key features is its built-in encryption support, which ensures that sensitive data, such as user financial information in the context of the money management app, is securely stored and protected from unauthorized access. This is especially important for applications that handle personal or sensitive data, as it adds an extra layer of security without requiring external dependencies. Additionally, Hive’s lazy-loading feature allows for efficient memory usage by loading only the required data into memory, making it highly suitable for resource-constrained environments like mobile devices. The database’s simplicity and ease of use, combined with its powerful capabilities, make it an ideal choice for mobile applications that require fast, secure, and reliable data storage. In the development of the money management app for this internship project, Hive was utilized to manage user data, including transaction histories, budgets, and category information. Its performance and offline capabilities ensured that users could access and manage their financial data seamlessly, even without an active internet connection. Furthermore, Hive’s integration with Flutter was straightforward, allowing for the smooth incorporation of data storage and retrieval functions within the app’s architecture. The result was a highly responsive application that met the demanding requirements of real-time financial management while maintaining the flexibility to evolve alongside the users' needs. Hive’s efficient data handling and secure storage capabilities made it an indispensable component in the development of the money management app, ensuring both performance and security were upheld to the highest standards

**Building the App for Android:**

Building the money management app for the Android platform involved leveraging the unique capabilities and features of Android while ensuring that the application delivered a seamless and high-quality user experience. Android, being one of the most widely used operating systems globally, offers a vast and diverse user base, making it essential to optimize the app for a wide range of devices, screen sizes, and hardware configurations. The development process began with careful planning of the app's architecture, taking into consideration Android's specific requirements, such as its activity lifecycle, permission management, and background process handling. The Flutter framework, which was used to develop the app, provided an excellent foundation for creating a consistent and responsive user interface across different Android devices. However, to fully capitalize on Android's capabilities, the app was further optimized to integrate with native Android features such as notifications, background services, and device-specific functionalities like biometric authentication. One of the critical aspects of Android app development is ensuring that the app performs efficiently across various hardware configurations, from high-end smartphones to more resource-constrained devices. To achieve this, the app was meticulously tested on multiple devices and emulators, focusing on optimizing memory usage, reducing load times, and ensuring smooth animations. Flutter's hot reload feature significantly accelerated this process, allowing rapid iteration and testing. Moreover, Android’s vast ecosystem of devices necessitated rigorous attention to responsive design. The app’s user interface was designed to be adaptive, ensuring that it rendered correctly and efficiently on different screen sizes and resolutions, from small smartphones to large tablets. This adaptability was achieved through the use of Flutter’s flexible widget system, which allowed the app to dynamically adjust its layout based on the device's characteristics. Additionally, Android's Material Design guidelines were closely followed to ensure that the app adhered to platform-specific design conventions, providing users with a familiar and intuitive experience. Building the app for Android also required careful management of permissions and user data security, particularly given the app's focus on financial management. Android’s permission model was used to control access to sensitive features like storage and network communication, ensuring that user data remained secure while maintaining the app's functionality. The app was also optimized for Android’s background processes, allowing it to handle tasks such as data synchronization and notifications without draining the device’s battery. This involved implementing efficient background services and utilizing Android’s WorkManager API to schedule tasks in a battery-efficient manner. The deployment process was streamlined using Android Studio, which provided comprehensive tools for building, testing, and deploying the app. The final step in the Android development process was preparing the app for distribution via the Google Play Store, which included ensuring compliance with Google’s policies, optimizing the app for discoverability, and implementing analytics to monitor app performance post-launch. The app was thoroughly tested for compatibility with different Android versions, ensuring that it functioned correctly on older devices while taking advantage of new features in the latest Android releases. Building the app for Android was a multifaceted process that required a deep understanding of the platform’s intricacies and a commitment to delivering a polished, user-friendly product. The result was a robust and versatile money management app that could meet the needs of Android users, providing them with a powerful tool to manage their finances with ease and efficienc

**Building the App for iOS:**

Building the money management app for iOS was a crucial part of the development process, given the platform's prominence among a significant user base that values high performance, security, and a seamless user experience. The iOS version of the app was developed using Flutter, which provides robust tools to ensure that the app adheres to Apple’s stringent design and performance standards. The development process began with configuring the app in Xcode, Apple’s integrated development environment (IDE), which is essential for building, testing, and deploying iOS apps. This involved setting up the necessary certificates, provisioning profiles, and app identifiers to ensure that the app could be properly signed and distributed through the Apple App Store.

A key consideration in building for iOS was the user interface (UI) design, which needed to align with Apple's Human Interface Guidelines (HIG). These guidelines emphasize simplicity, clarity, and consistency, all of which were meticulously followed to ensure that the app provided a native-like experience. The app’s UI was crafted using Flutter’s Cupertino widgets, which mimic the look and feel of native iOS components, such as navigation bars, buttons, and dialogs. This approach ensured that the app not only looked but also felt natural to iOS users, maintaining the high aesthetic and functional standards they expect.

Performance optimization was another critical aspect of the iOS build. iOS devices are known for their smooth performance, and the app was optimized to run efficiently on a wide range of devices, from the latest iPhone models to older versions still in use. Special attention was given to memory management and battery usage, as these are key factors in maintaining a positive user experience on iOS. The app was rigorously tested using both physical devices and simulators to identify and address any performance bottlenecks, ensuring that it remained responsive even under heavy use.

Security and privacy are paramount in the Apple ecosystem, and the app was designed to meet these standards. Data security features, such as encryption for stored financial data and secure handling of user credentials, were implemented to protect sensitive information. Additionally, the app complied with Apple's privacy policies, ensuring that user data was handled transparently and with consent. The integration of iOS-specific features, such as Face ID and Touch ID for secure authentication, further enhanced the app’s security and provided a seamless and secure user experience.

The deployment process for iOS also involved thorough testing and quality assurance. TestFlight, Apple’s platform for beta testing, was used to distribute the app to a group of testers who provided valuable feedback on usability and performance. This feedback was instrumental in refining the app before its final release. Once testing was complete, the app was submitted to the Apple App Store, where it underwent a rigorous review process to ensure compliance with all of Apple’s guidelines. Successfully navigating this process was a testament to the app’s quality and adherence to best practices in iOS development.

In conclusion, building the money management app for iOS was a comprehensive process that involved meticulous attention to detail in design, performance optimization, security, and compliance with Apple’s ecosystem. The result is a polished and reliable app that meets the high standards of iOS users, providing them with a powerful tool for managing their finances. This process not only highlighted the strengths of Flutter in cross-platform development but also demonstrated the importance of tailoring the app experience to meet the specific expectations of iOS users.

****Visual Studio Code (VS Code) as the IDE:****

Visual Studio Code (VS Code) was chosen as the primary Integrated Development Environment (IDE) for the development of the money management app due to its flexibility, lightweight nature, and robust feature set. VS Code is an open-source, cross-platform code editor developed by Microsoft, known for its speed and extensive customization options through extensions. For this project, VS Code was configured with the Flutter and Dart extensions, which provided comprehensive support for Flutter development, including syntax highlighting, code completion, and real-time debugging. The integrated terminal allowed for seamless execution of Flutter commands, such as building and running the app on both Android and iOS simulators or devices. Additionally, VS Code’s powerful version control integration with Git facilitated efficient project management, enabling smooth collaboration and tracking of changes throughout the development process. The IDE’s rich ecosystem of extensions, such as those for linting and formatting, further enhanced the coding experience by ensuring code quality and consistency. By using VS Code, the development process was streamlined, allowing for a highly productive environment that supported the rapid iteration and testing necessary to build a robust and user-friendly money management app

System design and Analysis

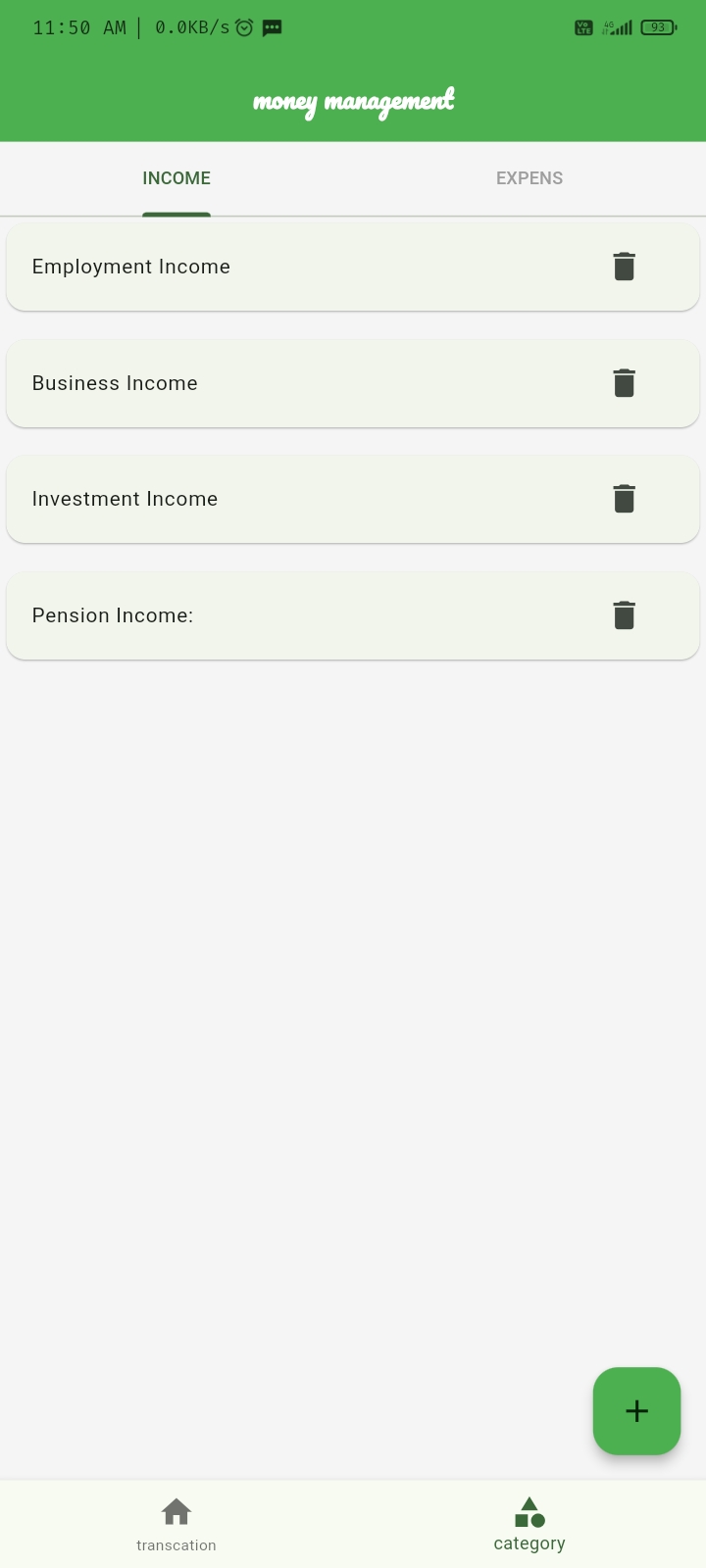
money management app

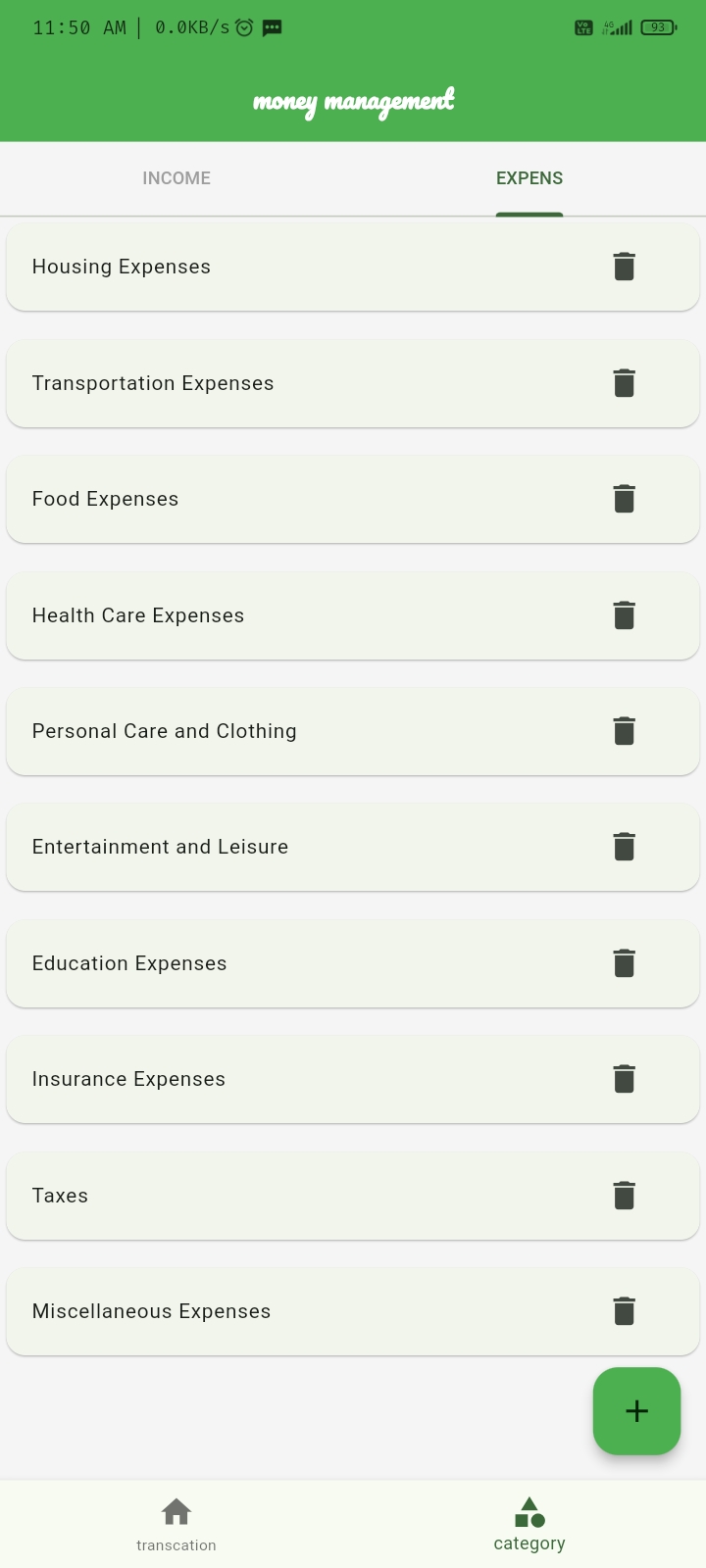
It is a comprehensive Expense Management Software that simplifies personal finance management for everyone. It offers features like automatic expense tracking and management with categorization (for bills, groceries, etc.), customizable budgeting tools, and timely bill reminders. You can even track your investments in mutual funds, stocks, Instant Loans, Pro Saver A/c, Check Credit Score and much more.

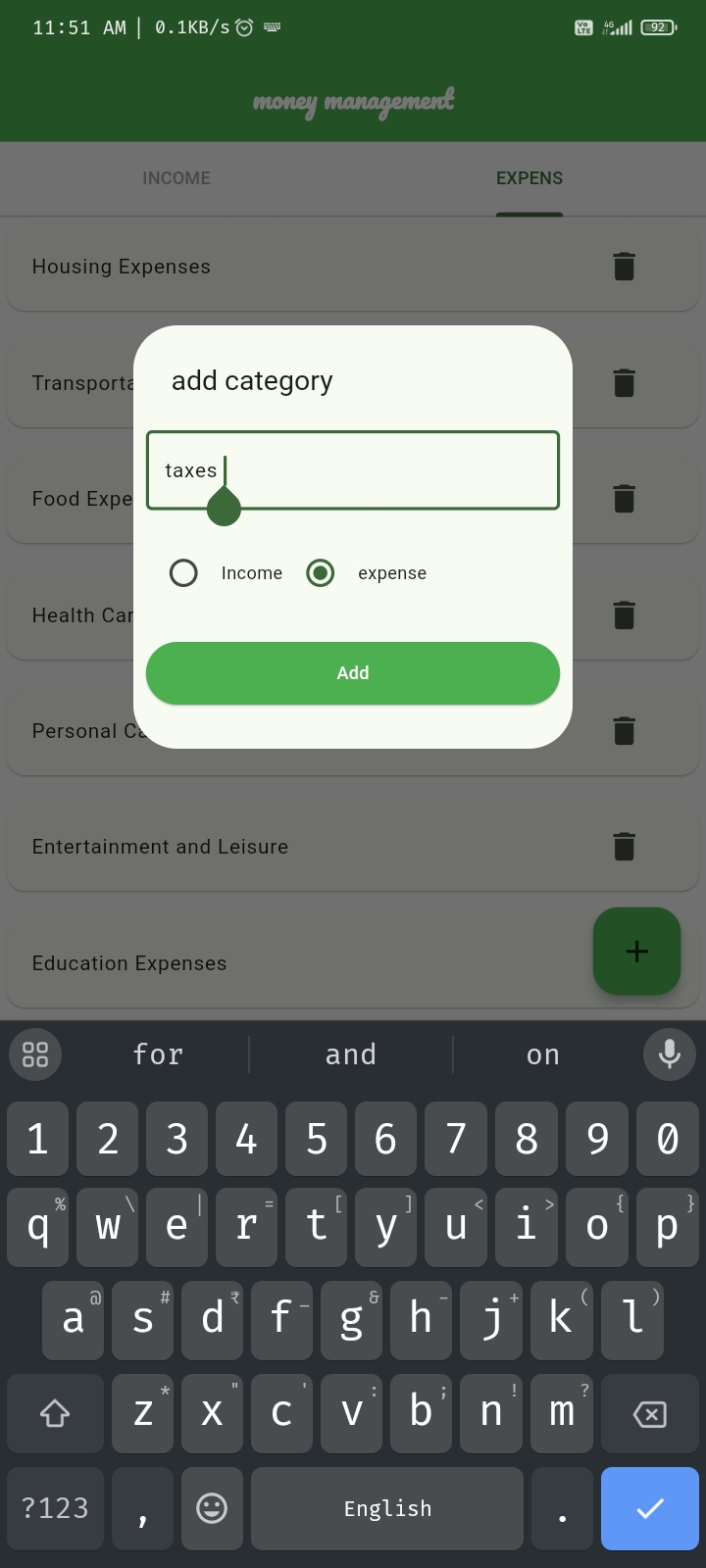
## App Screenshots

The “Pictures” page view in the money management app is designed to showcase images related to user experiences, such as screenshots of app features, financial tips, or motivational images. This feature enhances the visual appeal of the app and provides users with an engaging way to interact with content beyond traditional financial data.



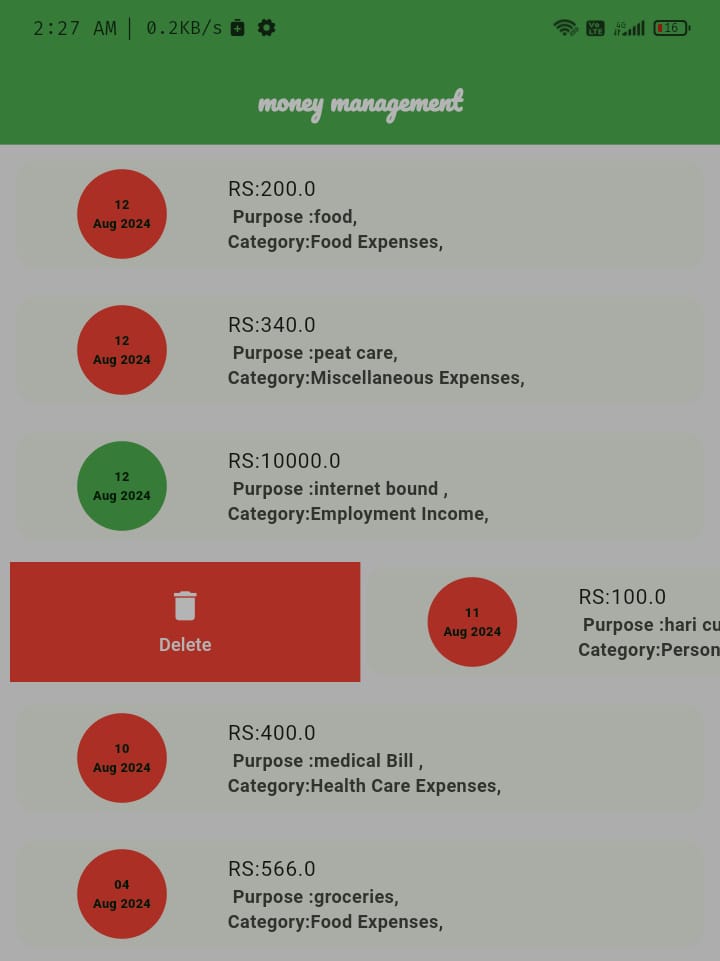










****

**Delete the purpose**

The money management app is designed with a strong emphasis on user-centric features, ensuring that individuals can effortlessly track and manage their financial activities. One of the core functionalities of the app is its transaction management system, which provides users with a clear and organized overview of their financial data. Each transaction entered into the app is displayed in a list format, where the user can easily view critical details such as the date, the amount involved, the purpose of the transaction, and the category it falls under. This categorization is crucial as it allows users to better understand their spending patterns, making it easier to budget and plan for future expenses.

The user interface is designed to be intuitive and user-friendly, with each transaction represented in a visually distinct format. The use of color-coded icons and clear labels helps users quickly identify whether a transaction is an expense or income. For example, expenses may be marked with red icons, while income transactions may be marked with green icons, providing an at-a-glance summary of the user's financial status. This visual differentiation is particularly beneficial in helping users maintain financial discipline, as they can instantly see whether their expenditures are outweighing their income.

In addition to simply viewing transactions, the app empowers users with the ability to edit and manage their transaction history actively. Recognizing that errors and changes are a natural part of managing personal finances, the app includes a robust deletion feature. This feature is designed to give users complete control over their financial records, allowing them to remove any transaction that may have been entered incorrectly or is no longer relevant. The deletion option is seamlessly integrated into the transaction list, represented by a trash icon next to each entry. This intuitive design ensures that users can easily delete transactions without needing to navigate through multiple screens or menus, making the process as straightforward and efficient as possible.

The implementation of the deletion feature is not just about convenience; it is also about maintaining the integrity of the user’s financial data. By allowing users to remove erroneous entries, the app helps to ensure that the financial summaries and reports generated by the app are accurate and reliable. This accuracy is essential for users who rely on the app to make informed financial decisions, as even small errors in transaction records can lead to significant discrepancies in budgeting and financial planning.

Moreover, the deletion feature is implemented with user safety and security in mind. To prevent accidental deletion, the app may include a confirmation prompt, asking users to confirm their decision before a transaction is permanently removed from the record. This additional step helps to safeguard against unintended data loss, ensuring that users can trust the app to manage their financial data responsibly.

Overall, the transaction management and deletion feature of the money management app is a testament to the app’s focus on delivering a user-friendly and efficient tool for personal finance management. By providing users with a clear overview of their financial activities, combined with the flexibility to edit and manage their records as needed, the app supports users in achieving better financial control and awareness. The thoughtful design of these features not only enhances the user experience but also aligns with the app’s broader goal of empowering individuals to take charge of their finances with confidence and ease.

**Codes**

Main .dart

import 'dart:async';

import 'package:flutter/material.dart';

import 'package:hive\_flutter/adapters.dart';

import 'package:money\_management/home/screen\_home.dart';

import 'package:money\_management/model/category/categoryModel.dart';

import 'package:money\_management/model/transaction/transction\_model.dart';

import 'package:money\_management/transaction/transaction\_form.dart';

Future<void> main() async {

WidgetsFlutterBinding.ensureInitialized();

await Hive.initFlutter();

// data base adapter register in hive

if (!Hive.isAdapterRegistered(CategoryModelAdapter().typeId)) {

Hive.registerAdapter(CategoryModelAdapter());

}

if (!Hive.isAdapterRegistered(CategoryTypeAdapter().typeId)) {

Hive.registerAdapter(CategoryTypeAdapter());

}

if (!Hive.isAdapterRegistered(transactionModeAdapter().typeId)) {

Hive.registerAdapter(transactionModeAdapter());

}

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false, // Remove the debug banner

theme: ThemeData(

useMaterial3: true,

// elevated Button Theme

elevatedButtonTheme: ElevatedButtonThemeData(

style: ElevatedButton.styleFrom(

foregroundColor: Colors.white, backgroundColor: Colors.green)),

// Define the default brightness and colors.

colorScheme: ColorScheme.fromSeed(seedColor: Colors.green

//brightness: Brightness.dark,

),

appBarTheme: const AppBarTheme(

backgroundColor: Colors.green,

titleTextStyle: TextStyle(

color: Colors.white,

fontSize: 20, // add font size

fontWeight: FontWeight.bold, // add font weight

fontFamily:

'Pacifico-Regular', // add font family (make sure to import the font package)

),

),

floatingActionButtonTheme: const FloatingActionButtonThemeData(

backgroundColor: Colors.green, // Optional: Set FAB color

),

),

home: ScreenHome(),

routes: {transaction\_form.routename: (ctx) => transaction\_form()},

);

}

}

Screen\_home.dart

import 'package:flutter/material.dart';

import 'package:money\_management/category/Category\_pop\_page.dart';

import 'package:money\_management/category/screen\_category.dart';

import 'package:money\_management/home/botton\_navigation.dart';

import 'package:money\_management/transaction/screen\_transaction.dart';

import 'package:money\_management/transaction/transaction\_form.dart';

// home screen

class ScreenHome extends StatelessWidget {

const ScreenHome({super.key});

static ValueNotifier<int> selectedIndexNotifier =

ValueNotifier(0); //select naviction bottn varibale

final \_pages = const [

ScreenTranssction(),

ScreenCategory()

]; // Home page view category & transaction pagss varibel array list

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.grey[100],

appBar: AppBar(

centerTitle: true,

title: const Text(

'money management',

),

),

bottomNavigationBar: const MoneyMangementBottonNavigation(),

body: SafeArea(

child: ValueListenableBuilder(

valueListenable: selectedIndexNotifier,

builder: (BuildContext context, int updateIndex, \_) {

return \_pages[updateIndex];

}),

),

// forme action button

floatingActionButton: FloatingActionButton(

onPressed: () {

if (selectedIndexNotifier.value == 0) {

Navigator.of(context).pushNamed(transaction\_form.routename);

} else {

Showcategoryaddpopu(context);

}

},

child: Icon(Icons.add)),

);

}

}

BottonNavigation .dart

import 'package:flutter/material.dart';

import 'package:money\_management/home/screen\_home.dart';

// transcation & category navigation botton class

class MoneyMangementBottonNavigation extends StatelessWidget {

const MoneyMangementBottonNavigation({super.key});

@override

Widget build(BuildContext context) {

return ValueListenableBuilder(

valueListenable: ScreenHome.selectedIndexNotifier,

builder: (BuildContext ctx, int updateIndex, Widget? \_) {

return BottomNavigationBar(

currentIndex: updateIndex,

onTap: (newIndex) {

ScreenHome.selectedIndexNotifier.value = newIndex;

},

items: [

BottomNavigationBarItem(

icon: Icon(Icons.home), label: 'transcation'),

BottomNavigationBarItem(

icon: Icon(Icons.category), label: 'category')

]);

});

}

}

ScreenTranssction

import 'package:flutter/material.dart';

import 'package:flutter\_slidable/flutter\_slidable.dart';

import 'package:intl/intl.dart';

import 'package:money\_management/db/category/category\_db.dart';

import 'package:money\_management/db/transaction/transaction\_db.dart';

import 'package:money\_management/model/category/categoryModel.dart';

import 'package:money\_management/model/transaction/transction\_model.dart';

class ScreenTranssction extends StatelessWidget {

const ScreenTranssction({super.key});

@override

Widget build(BuildContext context) {

TransactionDB.instance.UIrefersh();

CategoryDB.instance.refreshUI();

return ValueListenableBuilder(

valueListenable: TransactionDB.instance.transactionListNotifier,

builder: (BuildContext ctx, List<transactionMode> newlist, Widget? \_) {

return ListView.separated(

padding: EdgeInsets.all(6), //////////////

itemBuilder: (ctx, index) {

final \_khal = newlist[index];

return Slidable(

key: Key(\_khal.id!),

startActionPane: ActionPane(

motion: const ScrollMotion(),

children: [

SlidableAction(

onPressed: (ctx) {

TransactionDB.instance.deleteTransactiony(\_khal.id!);

},

backgroundColor: Colors.red,

foregroundColor: Colors.white,

icon: Icons.delete,

label: 'Delete',

),

],

),

child: Card(

elevation: 0,

child: ListTile(

leading: CircleAvatar(

radius: 50,

child: Text(

parseDate(\_khal.date),

textAlign: TextAlign.center,

style:

TextStyle(fontSize: 8, fontWeight: FontWeight.w900),

),

backgroundColor: \_khal.type == CategoryType.income

? Colors.green

: Colors.red,

),

title: Text('RS:${\_khal.amout}'),

subtitle: Text(

' Purpose :${\_khal.purpos},\nCategory:${\_khal.category.name},',

style: TextStyle(

fontWeight: FontWeight.bold,

),

),

/\* contentPadding:

EdgeInsets.symmetric(vertical: 0, horizontal: -30),\*/

),

),

);

},

separatorBuilder: (ctx, index) {

return SizedBox(height: 10);

},

itemCount: newlist.length);

},

);

}

String parseDate(DateTime date) {

final \_date = DateFormat('dd MMM yyyy').format(date);

final \_splitDate = \_date.split(' '); // Splitting by space

return '${\_splitDate[0]}\n${\_splitDate[1]} ${\_splitDate[2]}';

}

}

DATABASE Hive .dart

//import 'dart:ffi';

import 'package:flutter/material.dart';

import 'package:hive\_flutter/adapters.dart';

import 'package:money\_management/model/transaction/transction\_model.dart';

const TRANSACTIOM\_DB\_NAME = 'transaction-DB';

abstract class TransactionDBfunction {

Future<List<transactionMode>> getallTranscation();

Future<void> addTransaction(transactionMode obj);

Future<void> deleteTransactiony(String id);

}

class TransactionDB implements TransactionDBfunction {

TransactionDB.\_internal();

static TransactionDB instance = TransactionDB.\_internal();

factory TransactionDB() {

return instance;

}

ValueNotifier<List<transactionMode>> transactionListNotifier =

ValueNotifier([]);

@override

Future<void> addTransaction(transactionMode obj) async {

final \_DB = await Hive.openBox<transactionMode>(TRANSACTIOM\_DB\_NAME);

//Hive

await \_DB.put(obj.id, obj);

// getallTranscation();

}

@override

Future<List<transactionMode>> getallTranscation() async {

final \_DB = await Hive.openBox<transactionMode>(TRANSACTIOM\_DB\_NAME);

return \_DB.values.toList();

}

Future<void> UIrefersh() async {

final \_list = await getallTranscation();

\_list.sort((fiste, second) => second.date.compareTo(fiste.date));

transactionListNotifier.value.clear();

transactionListNotifier.value.addAll(\_list);

transactionListNotifier.notifyListeners();

}

@override

Future<void> deleteTransactiony(String id) async {

final \_DB = await Hive.openBox<transactionMode>(TRANSACTIOM\_DB\_NAME);

await \_DB.delete(id);

UIrefersh();

}

}

//import 'dart:js\_interop';

import 'package:flutter/material.dart';

import 'package:flutter/widgets.dart';

import 'package:hive/hive.dart';

import 'package:money\_management/model/category/categoryModel.dart';

const CATEGORY\_DB\_NAME = 'category database';

abstract class CategoryDBfunctions {

Future<List<CategoryModel>> getCategories();

Future<void> insertCategory(CategoryModel value);

Future<void> deletecategory(String categoryID);

}

class CategoryDB implements CategoryDBfunctions {

/\*\_internal() wich is called exactly once, and since it is private it can only be called in

this Class and we also prevent it from being instantiated outside of here. This constructor

can be used to initialize logic.\*/

CategoryDB.\_internal();

static CategoryDB instance = CategoryDB.\_internal();

factory CategoryDB() {

return instance;

}

ValueNotifier<List<CategoryModel>> incomecategoryList = ValueNotifier([]);

ValueNotifier<List<CategoryModel>> expensecategoryList = ValueNotifier([]);

@override

Future<void> insertCategory(CategoryModel value) async {

final \_CATEGORY\_DB = await Hive.openBox<CategoryModel>(CATEGORY\_DB\_NAME);

await \_CATEGORY\_DB.put(value.Id, value);

refreshUI();

}

@override

Future<List<CategoryModel>> getCategories() async {

final \_CATEGORY\_DB = await Hive.openBox<CategoryModel>(CATEGORY\_DB\_NAME);

return \_CATEGORY\_DB.values.toList();

}

Future<void> refreshUI() async {

final \_allcategory = await getCategories();

incomecategoryList.value.clear();

expensecategoryList.value.clear();

Future.forEach(\_allcategory, (CategoryModel category) {

if (category.type == CategoryType.income) {

incomecategoryList.value.add(category);

} else {

expensecategoryList.value.add(category);

}

/\*refreshUI();\*/

});

incomecategoryList.notifyListeners();

expensecategoryList.notifyListeners();

}

@override

Future<void> deletecategory(String categoryID) async {

final \_CategoryDB = await Hive.openBox<CategoryModel>(CATEGORY\_DB\_NAME);

await \_CategoryDB.delete(categoryID);

refreshUI();

}

}

**Flutter as the Programming Framework**

· **Cross-Platform Development:**

* · Flutter enables the creation of applications for both Android and iOS using a single codebase, significantly reducing development time and costs.

· **Single Codebase:**

* · By writing code once, developers can deploy applications across multiple platforms without the need for platform-specific code, ensuring consistency and ease of maintenance.

· **High Performance:**

* · Flutter applications are compiled directly to native ARM code using Dart's AOT (Ahead-of-Time) compilation, resulting in high-performance apps with smooth animations and fast startup times.

· **Rich Set of Pre-Built Widgets:**

* · Flutter provides an extensive library of customizable widgets that adhere to Material Design (for Android) and Cupertino (for iOS) standards, allowing for the creation of beautiful, native-like user interfaces.

· **Hot Reload:**

* · The "hot reload" feature allows developers to see the effects of code changes instantly without restarting the app, greatly accelerating the development process and enabling quick experimentation.

· **Dart Language:**

* · Flutter is built on the Dart programming language, which offers modern, object-oriented features and strong typing, making it easy to learn and productive for developers.

· **Expressive and Flexible UI:**

* · Flutter’s widget-based architecture allows for the creation of highly customizable, flexible, and expressive user interfaces that can adapt to different screen sizes and orientations.

· **Strong Community and Ecosystem:**

* · Flutter has a rapidly growing community and a rich ecosystem of plugins and packages, providing solutions for common development tasks and enabling easy integration of third-party services.

· **Comprehensive Documentation:**

* · Flutter is well-documented, with extensive resources available for developers, including tutorials, API references, and community-contributed content, making it accessible even for those new to mobile development.

· **Continuous Updates and Support:**

* · Backed by Google, Flutter receives regular updates, ensuring it stays up-to-date with the latest mobile development trends and continues to improve in terms of features and performance.

· **Integration with Firebase and Other Tools:**

* · Flutter seamlessly integrates with Firebase, a comprehensive platform for building and managing mobile applications, offering backend services like authentication, database, and analytics out-of-the-box.

· **Customizability and Control:**

* · Flutter provides developers with complete control over every pixel on the screen, enabling the creation of highly tailored user experiences that can differentiate apps from others on the market.

**Learning Outcomes**

1. User-Centered Design (UCD): Throughout my internship, I developed a deep understanding of user-centered design principles by focusing on the needs and preferences of the end-users of the money management app. I applied user research techniques, such as surveys and interviews, to gather insights and ensure that the app’s features were designed to enhance user experience and satisfaction.
2. Usability Principles: I learned to apply key usability principles—simplicity, consistency, feedback, and error prevention—to the app’s design. This involved creating an intuitive interface that minimizes user errors and provides clear feedback, resulting in a user-friendly experience.
3. Interaction Design: I gained insights into designing effective interactions between users and the app. This included creating seamless navigation flows and interactive elements that facilitate user engagement and ensure efficient task completion.

**Practical Skills**

1. Wireframing and Prototyping: I honed my skills in wireframing and prototyping by developing visual representations of the app’s interface and user flows. Creating prototypes allowed me to test design concepts, gather user feedback, and iterate on design solutions before final implementation.
2. Information Architecture (IA): I learned to organize and structure content effectively, ensuring that users could navigate the app intuitively. This involved designing a logical hierarchy for information and features, which improved the app’s usability and accessibility.
3. Visual Design: I acquired knowledge in visual design principles, including layout, typography, and color theory. Applying these principles helped me create an aesthetically pleasing and cohesive design that aligned with the app’s branding and enhanced user engagement

**Tools and Technologies**

1. Flutter Framework: I became proficient in using Flutter to develop cross-platform mobile applications. Flutter’s widget-based framework enabled me to build a responsive and visually consistent user interface for both iOS and Android platforms.
2. Dart Programming Language: I used Dart for programming the app’s logic and functionality. Dart’s features, such as its strong typing and asynchronous programming capabilities, facilitated efficient and reliable code development.
3. Hive Database: I implemented Hive as the local database for managing user data, including transactions and budgets. Hive’s lightweight and performant key-value store allowed for efficient data handling and offline access.

Research and Analysis

1. User Research: I conducted user research through methods such as interviews and surveys to gather feedback on user needs and preferences. This research informed the design decisions and ensured that the app addressed real user pain points.
2. Usability Testing: I performed usability testing to evaluate the app’s functionality and identify areas for improvement. By observing users interacting with the app, I was able to gather insights and make data-driven design refinements.
3. Personas and User Journeys: I created user personas and mapped user journeys to better understand user goals and behaviors. This analysis helped in designing features that aligned with user needs and improved the overall user experience.

Communication and Collaboration

1. Effective Communication: I developed strong communication skills, learning to present design ideas and project updates clearly to stakeholders and team members. Effective communication was crucial for aligning expectations and ensuring project success.
2. Collaboration Skills: I collaborated with cross-functional teams, including developers and product managers, to integrate design solutions with technical requirements. This collaboration ensured that the app’s design was feasible and aligned with project goals.
3. Presentation Skills: I improved my presentation skills by showcasing design concepts and findings to various audiences. Creating compelling presentations and articulating design rationale helped in gaining stakeholder buy-in and support.

Ethical and Inclusive Design

1. Accessibility: I focused on designing the app to be accessible to users with diverse abilities. This included implementing accessibility features such as screen reader compatibility and ensuring sufficient contrast for users with visual impairments.
2. Ethical Considerations: I considered the ethical implications of design decisions, including data privacy and user consent. Ensuring that the app handled user data responsibly and transparently was a priority.
3. Cultural Sensitivity: I designed the app with cultural sensitivity in mind, considering diverse user backgrounds and preferences. This approach helped ensure that the app was inclusive and relevant to a global audience.

**Career Development**

1. Portfolio Development: I created a professional portfolio showcasing the money management app project and other design work. This portfolio highlights my skills in UI/UX design and demonstrates my ability to tackle complex design challenges.
2. Industry Trends: I stayed updated with the latest trends and best practices in mobile app development and UI/UX design. Keeping abreast of industry developments allowed me to apply contemporary techniques and tools to my work.

# Bibliography

# .

1. Google. (2024). Flutter: Beautiful native apps in record time. Retrieved

from <https://flutter.dev>

2. Nguyen, T. (2023). Mastering Flutter: A comprehensive guide to building cross-platformapplications. Tech Books Publishing.

3 Hartman, K. (2023). Dart Programming for Beginners: An easy-to-followguide to building apps with Flutter. CodeHouse Publishing.

4 Johnson, M. (2024). Building Mobile Apps with Flutter and Dart.AppPress.

5 Google Developers. (2023). Dart Language Overview. Retrieved from <https://dart.dev>

6 Fitzgerald, L. (2023). Modern Mobile Development: Strategies for Android and iOS, MobileTech Publishing

7 Stack Overflow Community. (2024). Common issues and solutions in Flutter development. Retrieved from <https://stackoverflow.com>

8 Robinson, S. (2023). Understanding Hive: Lightweight database forFlutter. AppDev Magazine, 15(3), 42-50

.

9 Google Developers. (2024). Integrating Firebase with Flutter. Retrieved from <https://firebase.google.com/docs/flutter/setup>

10. Microsoft. (2023). Visual Studio Code: Code editing. Redefined. Retrieved from https://code.visualstudio.com

# Articles and Online Resources

#### Google Developers. (2024). Rutter: Beautiful native apps in record time. Retrieved from https://flutter.dev

#### Hartman, K. (2023). Building Responsive User Interfaces with Flutter. Mobile Dev Weekly. Retrieved from https://mobiledevweekly.com/flutter-responsive-ui

#### Nguyen, T. (2023). Best Practices for Cross-Platform Mobile Development with Flutter. Code Journal. Retrieved from https://codejoumal.dev/flutter-cross-platform-best-practices

#### Robinson, S. (2023). Understanding Hive: Lightweight datubase for Flutter. AppDev Magazine, 15(3), 42-50.Retrieved from https://sppdevmagazine.com/hive-flutter-database

#### Johnson, M. (2024). Exploring Dart's Role in Modern Mobile Development. Tech Talks. Retneved from https/techtalks.iordart-modem-mobile-development

#### Stack Overflow Community. (2024). Common Issues and Solutions in Flutter Development. Retrieved from https://stackoverflow.com

#### . Fitzgerald, L. (2023). Integrating Firebase with Flutter for Enhanced App Functionality. Firebase Dev Blog. Retrieved from https://firebase.googleblog.com/flutter-integration

#### Microsoft. (2023). Visual Studio Code: Code Editing. Redefined. Retrieved from https://code.visualstudio.com

#### Google Developers. (2024). Dart Language Overview. Retrieved from https://dart.dev

#### 10. PerformanceInsights.Retrievedfrom https://appperformanceinsights.com/flutter-optimization

#### 11Williams, R. (2024). Optimizing Flutter Apps for Performance on Android