SP-26 Clyde Budget

FINAL REPORT

CS 4850 – SECTION 04 – FALL 2024

INSTRUCTOR: ARTHUR CHOI

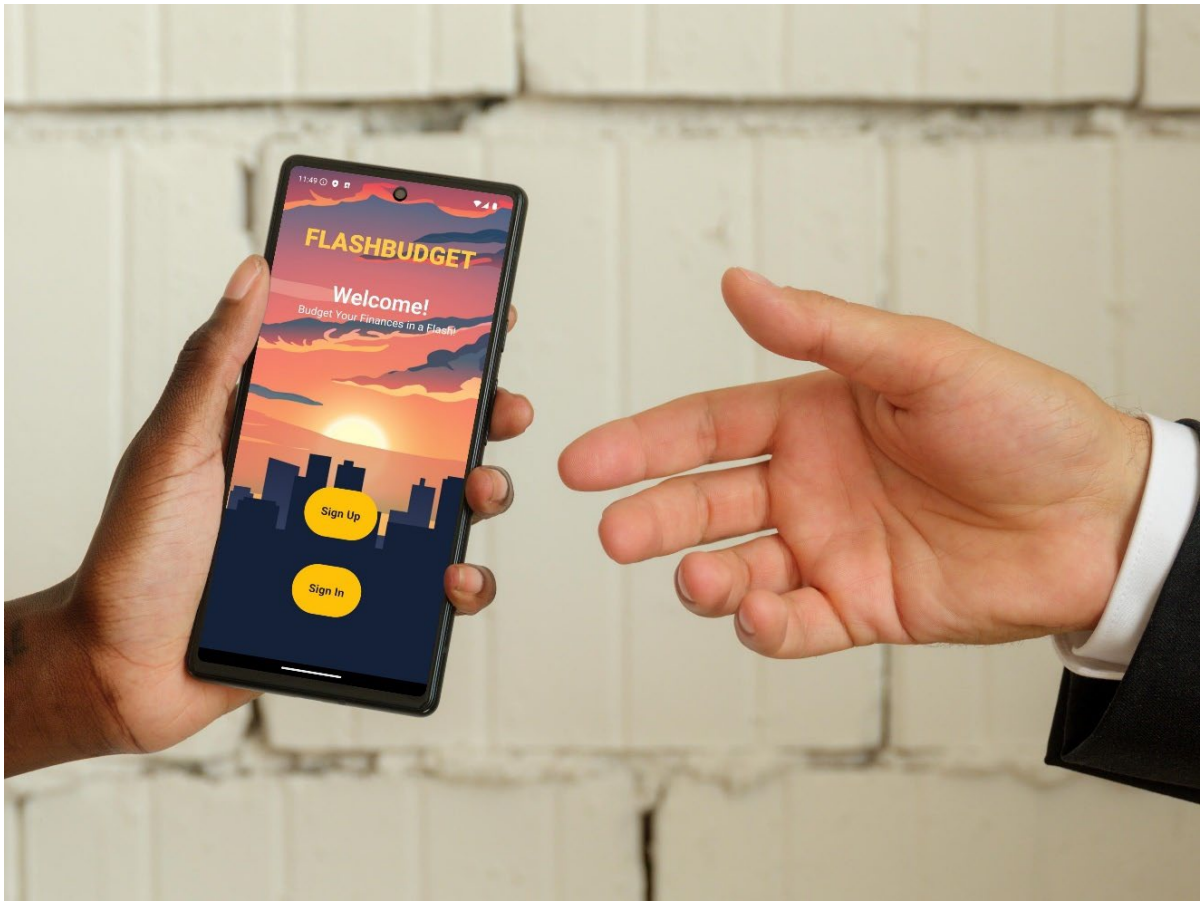
DATE: DECEMBER 2, 2024

WEBSITE LINK: HTTPS://BAGONSS.GITHUB.IO/BUDGETMOBILEAPP/

GITHUB LINK: HTTPS://GITHUB.COM/BAGONSS/BUDGETMOBILEAPP

		
Brandon Johnson **Team Lead/Documentation/Tester**	**Tobi Samaye** **Programmer/Developer**	**Brandon Machipisa** **Researcher/Programmer/ Developer**

Table of Contents



(FlashBudget, the name of our mobile app)

INTRODUCTION

The Mobile Budget App's objective purpose is to provide users with a seamless and sufficient platform to manage their finances efficiently. As mobile technology is becoming an increasing necessity as well as the growing need for financial literacy, this app is primarily designed to help individuals with tracking their own expenses, set financial goals, and make informed financial decisions. This project also focuses on a modern, user-friendly model through a virtual emulation using programming tools.

Objectives

The primary objectives of the Mobile Budget App are:

1. Financial Tracking
2. Goal Setting
3. Insight and Analytics
4. Accessibility

Scope

The project involves the development of a mobile application through an emulator using Flutter and Dart API, which allows it to be compatible with both Android and iOS devices. Firebase UI, a backend database, is implemented into the program to manage data storage, retrieval, and processing.

Key Functionality includes:

- Navigation
- Transaction and Tracking
- Reports and Analytics

Project Timeline

The project is expected to be a semester long (approximately 4.5 months) developmental process.

- Month 1: Research and planning.
- Month 2-3: Design and initial development.
- Month 4: Integration and testing.
- Month 5: Final adjustments and presentations.

REQUIREMENTS

The Mobile Budget App is designed to address three main sets of requirements: functional, non-functional, and external interface requirements, each of which contributes to a seamless user experience.

Functional Requirements

1. **User Authentication**:
    a. **Login**: Users can log in using their email or phone number and a password.
    b. **Create Account**: Users must be able to register an account with email/phone and a secure password.
    c. **Password Recovery**: Users can reset forgotten passwords through a recovery link or code sent via email/phone.
2. **Two-Factor Authentication**:
    a. Users can enable 2FA through SMS or an authenticator app (e.g., Okta Verify, Duo Mobile).
3. **Home Page Features**:
    a. Display an overview of financial information such as account balances, recent transactions, and budget summaries.
    b. Quick links for accessing detailed budgets, adding transactions, setting goals, and viewing reports.
4. **Navigation**:
    a. A clear and intuitive navigation bar gives access to main sections like:
        i. Budgets
        ii. Transactions
        iii. Goals
        iv. Reports
        v. Settings
5. **Transactions and Tracking**:
    a. Users can record, view, and categorize income and expenses (e.g., groceries, bills, entertainment).
    b. Set up recurring payments for rent, subscriptions, and utilities.
6. **Reports and Analytics**:
    a. Generate visual reports to track spending and identify patterns.
    b. Provide insights to help users adjust financial habits.

Non-Functional Requirements

1. **Security**:
    a. Ensure secure login and data storage through encryption.
    b. Display terms of service and privacy policy to meet compliance standards.
    c. Regularly update the app to include the latest security improvements.
2. **Performance**:
    a. Ensure smooth operation on Android and iOS devices using Flutter's cross-platform support.
    b. Design the app to manage large user datasets without performance degradation.
3. **Usability**:
    a. Intuitive user interface with a focus on simplicity and clarity.
    b. Provide error messages that guide users to resolve issues (e.g., invalid email format).
    c. Responsive design for seamless usability on various screen sizes.

External Interface Requirements

1. **User Interface**:
    a. Clean, minimalistic design with intuitive icons and labels.
    b. Support for both dark and light mode themes.
2. **Hardware Interface**:
    a. Compatible with smartphones and tablets running the latest versions of Android or iOS.
3. **Software Interface**:
    a. Backend integrates securely with Firebase or another scalable database solution.
    b. Support for third-party services like SMS gateways for authentication.
4. **Communication Interface**:
    a. Send automated email notifications for account activity, password resets, and monthly summaries.

TESTING

The Mobile Budget App's core features were rigorously tested using a structured methodology to ensure functionality, usability, and reliability. Testing focused on key areas like navigation, transaction tracking, and reporting.

Testing Methodology

- **Unit Testing**: Individual features such as login functionality, account creation, and two-factor authentication were tested to ensure reliability under normal and edge-case conditions.
- **Integration Testing**: Interactions between components, such as navigation between the home page and reports or transactions, were tested for seamless functionality.
- **System Testing**: The app was tested as a whole to verify compliance with requirements and smooth operation on the Android Studio Emulator.
- **User Acceptance Testing (UAT)**: Feedback from team members and peers was integrated to improve usability and address any design or functional issues.

Testing Highlights

- **Demo Testing**:
  - Core features such as Sign Up/Sign In, Home Navigation, and Reports and Analytics were displayed in a demo environment.
  - Transaction tracking, financial goal setting, and visual analytics were tested for accuracy and user-friendliness.
- **Performance Testing**:
  - Evaluated Flutter's cross-platform performance, ensuring smooth operation across Android and iOS devices.
  - Confirmed the app's ability to handle large datasets without compromising speed or reliability.
- **Usability Testing**:
  - Validated that users could navigate seamlessly between sections like Budgets, Transactions, Goals, and Reports.
  - Error messages were tested to ensure clarity and provide guidance for resolving common issues.

Test Results

- **Functional Testing**: All primary features performed reliably, with no significant issues identified during unit and integration testing.
- **Performance Testing**: The app demonstrated optimal performance on the Android Studio Emulator, with smooth transitions and responsive design.
- **Usability Testing**: Feedback from peer reviews emphasized the app's intuitive interface and effective navigation, leading to minor improvements in error messaging and visual elements.

The testing process confirmed that the Mobile Budget App successfully met its functional and non-functional requirements, delivering a secure, dependable, and intuitive financial management solution.

DEVELOPMENT DOCUMENTS

Mobile Files

The mobile application files are written in Dart and structured to handle the functionality, user interface, and design of the app. Below is an overview of each file and its role in the project:

main.dart – its purpose is to hold as the entry point of the app, which is responsible for initializing the application. Its key features are the app-level configurations and calls necessary startup functions.

theme.dart - Defines the color schemes, fonts, and styles for the app. Key features is to produce a consistent and visual appeal.

welcome_screen.dart – its purpose is to view as the first screen users see upon opening the app. It holds navigation options for sign-in or sign-up.

welcome_button.dart – a reusable button widget for the welcome screen. Uses configurable text and on-click actions and a uniform design for call-to-action buttons.

signup_screen.dart - allows new users to register for the app. Its key features are input fields for user details (e.g., name, email, password) and data validation and backend integration.

signin_screen.dart – enables users to log in to their accounts. Its key features are input fields for email and password. Also, errors in handling incorrect credentials.

forget_password_screen.dart – its purpose is to provide users with a way to reset their password securely. The key features are placing email input for password recovery and integration with backend to send password reset links.

custom_scaffold.dart – its purpose is defining a reusable scaffold for consistent layout and navigation across screens. Its key features are having a predefined app bar and bottom navigation. Also ensures a uniform design.

home_screen.dart – its purpose is to serve as the main dashboard for the app. The key features are the overview of budgets, transactions, and goals. Also, a quick navigation to other sections of the app.

budgetanalytics_screen.dart - Displays detailed financial analytics, such as spending patterns and budgeting insights. Its key features show charts and graphs for visual representation. It also filters for date ranges and categories.

transactions_screen.dart - Lists user transactions and allows for detailed exploration. It features sorting and searching transactions. Also, it gives the option to edit or delete entries.

Website Files

The website files include HTML and CSS documents designed to present information about the mobile budget app. These files are structured to give potential users professional and informative browsing experience.

> **about.html** - Provides detailed information about the app, including features and benefits. Practically holds a description of the app's purpose. It also highlights key functionalities with visuals or icons.
>
> **index.html** - Acts as the landing page for the app's website. It is the overview of the app and is linked to the "About" page.
>
> **style.css** - Controls the appearance and layout of the website. Its key features are styling for fonts, colors, and layout. It ensures responsive design for mobile and desktop views.

CHALLENGES

There are several challenges when it comes to developing a budget mobile application, ranging from technical to user experience improvements. Below are some of the challenges that we have come across and how they impact the project:

Data Privacy and Security

- **Challenge:** Protecting sensitive financial information from breaches or misuse.
- **Impact:** Loss of trust and credibility if user data is compromised.
- **Solution:** Implementation of two-factor authentication, and regularly updating the app to address vulnerabilities.

Storing Several User Accounts

- **Challenge:** Allowing the app to store user accounts when signing up and signing in.
- **Potential Impact:** Complexities in API integration and user frustration if accounts are inaccessible.
- **Solution:** Using the Firebase UI database to ensure user accounts are stored properly and retrieved when needed.

VERSION CONTROL

The source code has been placed onto the GitHub page and will be posted in the submissions file.

Lines of Source Code

Approximately a total of 1,351 lines of code and 244 comments within 11 development documents.

Total Number of Man Hours/Gantt Chart

A total of 200-man hours. 183 Planned Hours and 17 Delayed Hours

Project Name: SP-26-Clyde
Report Date: Dec 2nd 2024

					Milestone #1				*Milestone #2*				*Milestone #3*				Final Report	
Deliverable	**Tasks**	**Complete%**	**Current Status Memo**	**Assigned To**	**09/02**	**09/09**	**09/16**	**09/23**	**09/30**	**10/07**	**10/14**	**10/21**	**10/28**	**11/04**	**11/11**	**11/18**	**11/25**	**12/02**
Discovery and Plann	Develop Project Plan	100%	Delayed but Completed	Bran(IL)	5	1												
	Implement Functional Requiren	100%	Completed	Tobi	3	2												
	Write Software Requirements S	100%	Completed	All	10													
	Write Software Design Docume	100%	Completed	Brandon	3	1												
Project design	Implement Functional Requiren	100%	Completed	Tobi		1	4	1										
	Compile Brainstorming Notes	100%	Completed	Bran(IL)		2	2	2										
	Finalize Design Documentation	100%	Completed	Brandon		1	3	1										
	Define User Characteristics	100%	Completed	Tobi			5	1										
	Establish Operation Environmer	100%	Completed	Tobi				7										
Prototype Developm	Develop Initial Prototype	100%	Completed	All					2	4	1	2						
	Prepare for Peer Review Presen	100%	Completed	Bran(IL)							2	5						
	Implement Functional Requiren	100%	Completed	Bran(IL)							1	2						
Actual Development	Code Mobile Budget App	100%	Completed	Tobi				1	1	2	4	2		2				
	Integrate Core Functionalities	100%	Completed	Brandon, Tobi					1	2	3		1	3				
	Prepare for Initial Testing	100%	Completed	Tobi					3	2	1							
	Develop Navigation Menu	100%	Completed	Tobi					2		2	1		4	2			
	Implement Non-Functional Req	100%	Completed	Bran(IL)					1	2	2	4	1	2				
External Interface Re	Implement User Interface Requi	100%	Completed	Tobi					2	1		3						
	Integrate Hardware Interface Re	100%	Completed	Tobi, Bran (IL)					2	2		2	1					
	Ensure Software Interface Requ	100%	Completed	Brandon					3	1		4						
	Establish Communication Interf	100%	Completed	Brandon					4	2	2	1						
Testing and Final Ad	Conduct Unit Testing	100%	Completed	ALL								1		3	3	1		
	Perform User Acceptance Testi	100%	Completed	Tobi									1	3	4			
	Make Necessary Adjustments	100%	Completed	Bran(IL)									3		2	2		
	Review and Revise all Interface	100%	Completed	Brandon									1	1	2	3		
Final Report and C-I	Compile Final Report Package	100%	Finally Completed	All												1	7	3
	Create Video Demo	100%	Finally Completed	Tobi, Bran (IL)													3	1
			Total work hours	**200**	**21**	**8**	**14**	**13**	**21**	**18**	**18**	**27**	**8**	**18**	**13**	**7**	**10**	**4**

*Developed from flutter and source code is transferred to GitHub for management/version control

Legend
Planned 183
Delayed 17
Number Work: 200 man hours

Below is a link to the source code on the GitHub page:

https://github.com/bagonss/BudgetMobileApp

SUMMARY

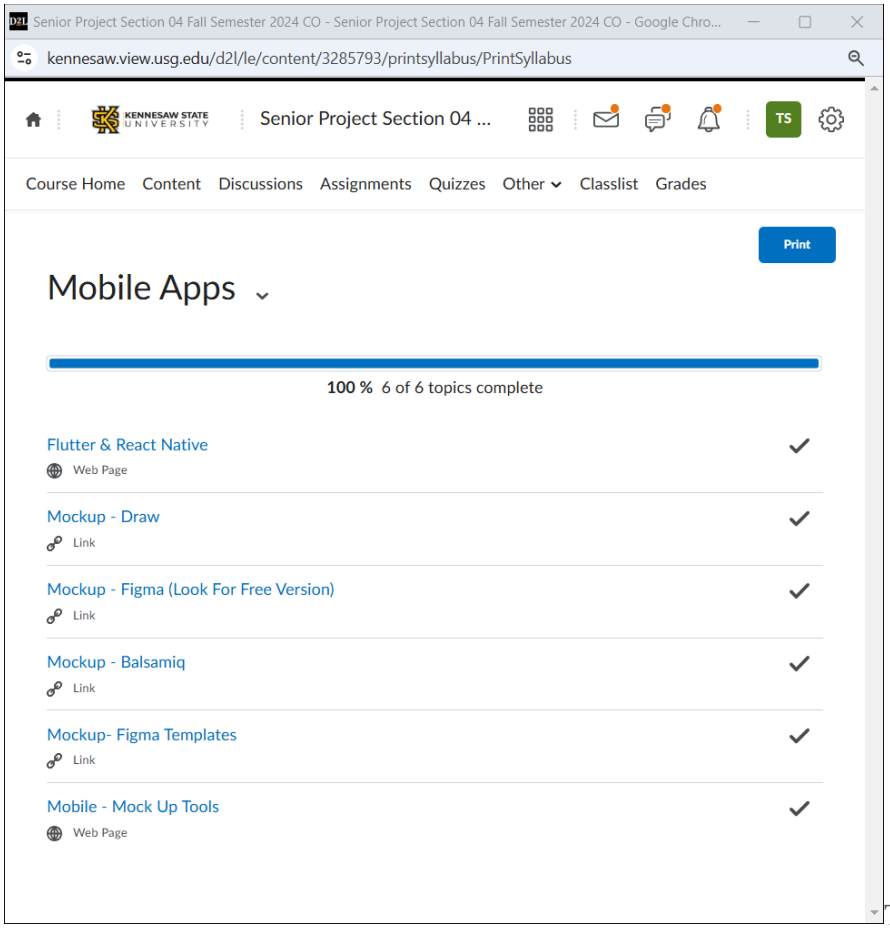
Our mobile budget app, compatible with both Android and IOS, is a useful app that is sure to improve users' organization when it comes to their finances. It is equipped with transaction history tracking, charts, and graphs to make meeting your financial goals much easier.

The mobile budget app project is a one-stop solution to the increasingly important task of financial education and individual finance today. The app is powered with tools such as Flutter, Dart API, and Firebase UI to provide secure, easy, and user-friendly interface to keep track of your expenses, set financial targets, and get insights through rich analytics. This is a semester-long development effort which consists of careful planning, thorough testing, and consideration of both functional and non-functional requirements for smooth user experience.

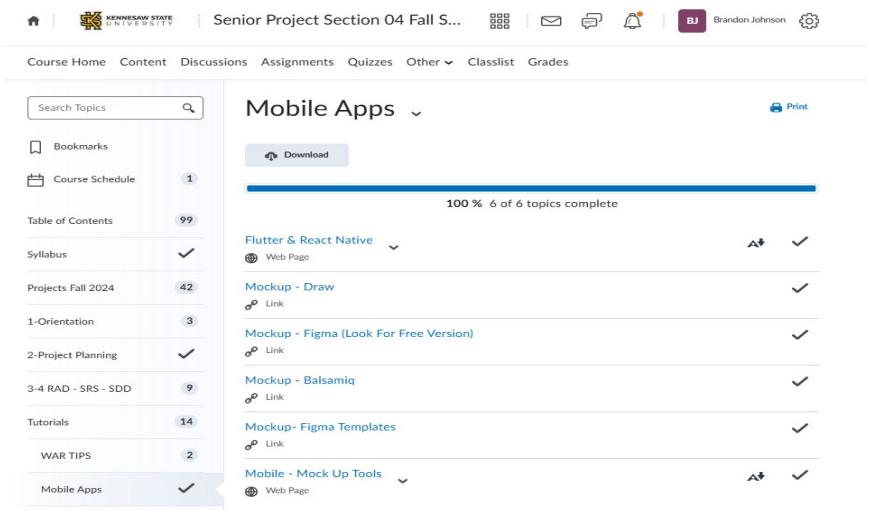
Data privacy, banking integration, storing multiple accounts and other issues caused some challenges but with proactive steps like two-factor authentication and integration with Firebase the app is better built for the future. The structured test methodology also verified the functionality, usability, and performance of the app to ensure it was ready to be used.

Not only does this project demonstrate the possibilities for the budget app but also creates a roadmap for future improvement so that it will remain relevant in a world with ever more competitive digital information.

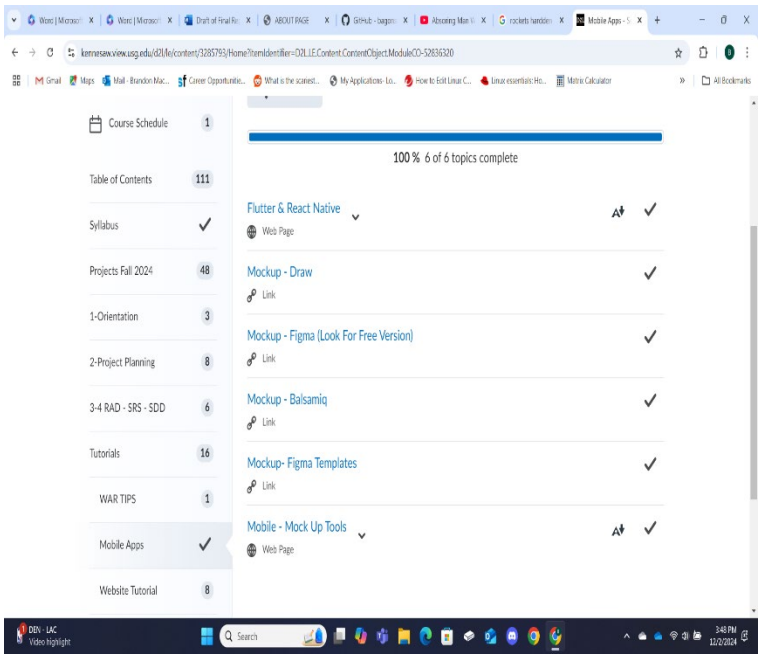
APPENDIX A. TUTORIALS



Tobi Samaye



Brandon Johnson



Brandon Machipisa

APPENDIX B. TESTING DEMO

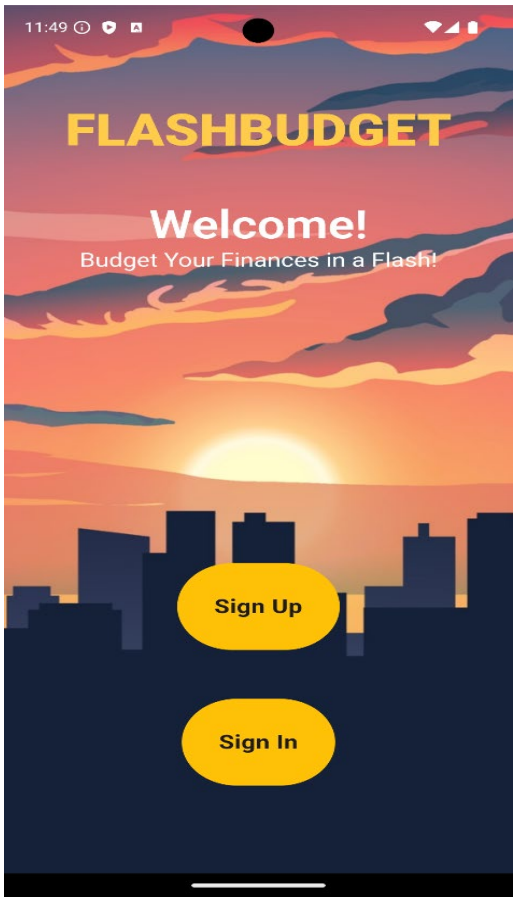
The welcome screen serves as the first impression for users, embodying the app's aesthetic and functionality. The background image was selected from an auto-generated template website that ensures the image is free of copyright restrictions. The app's name, "FlashBudget," was brainstormed with the goal of being unique and relevant to the app's purpose. To avoid potential legal issues, the name was cross-referenced with existing trademarks and found to have no copyright conflicts. The tagline, "Budget Your Finances in a Flash!" adds a personal touch, immediately conveying the app's value proposition. The design itself was inspired by various online examples, but these inspirations were creatively adapted into an original format.

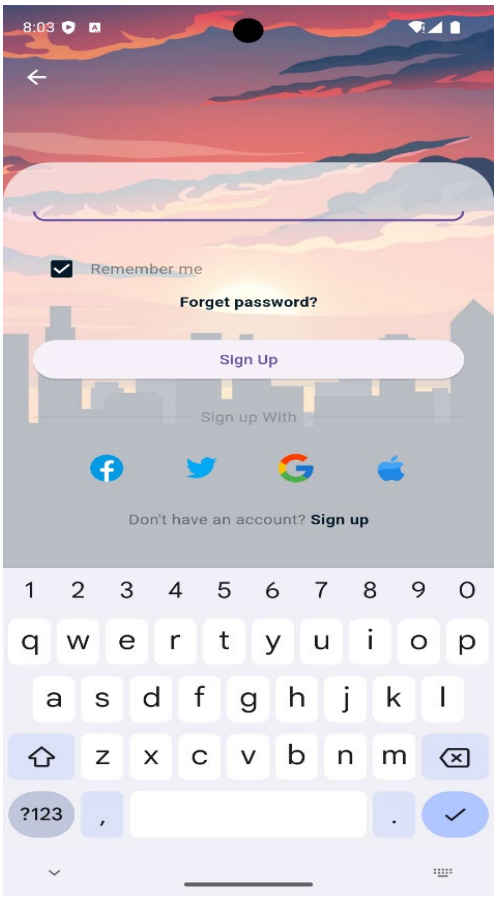
The home screen was designed to provide straightforward access to the app's primary features. It offers two main options: "Report and Analytics" and "Transaction and Tracking." The layout ensures simplicity, making navigation intuitive for users. The intention behind this design was to quickly guide users to the app's core tools for financial management.

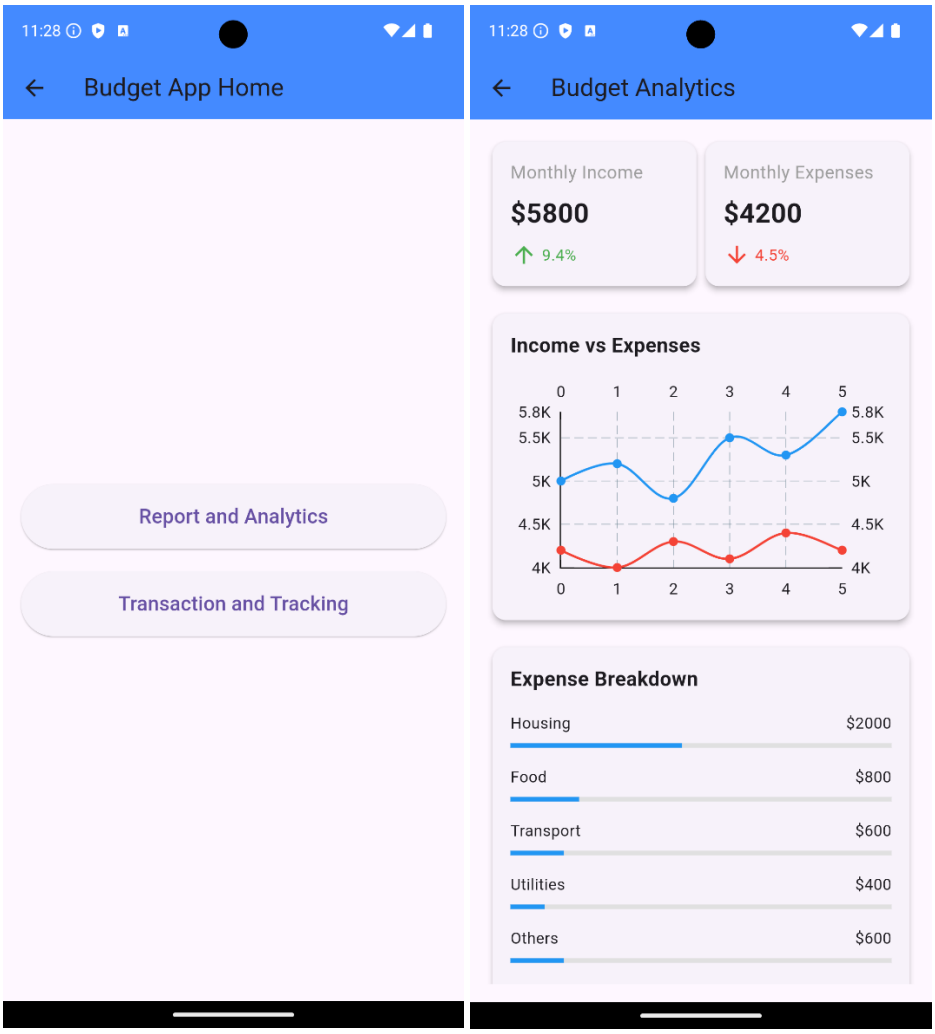
The transaction tracking screen focuses on helping users manage daily financial activities. This screen displays the total income and expenses for a given period, along with a list of individual transactions. A "+" button was installed to allow users to easily add new transactions. The green and red color scheme for income and expenses was chosen to intuitively differentiate between positive and negative cash flow.

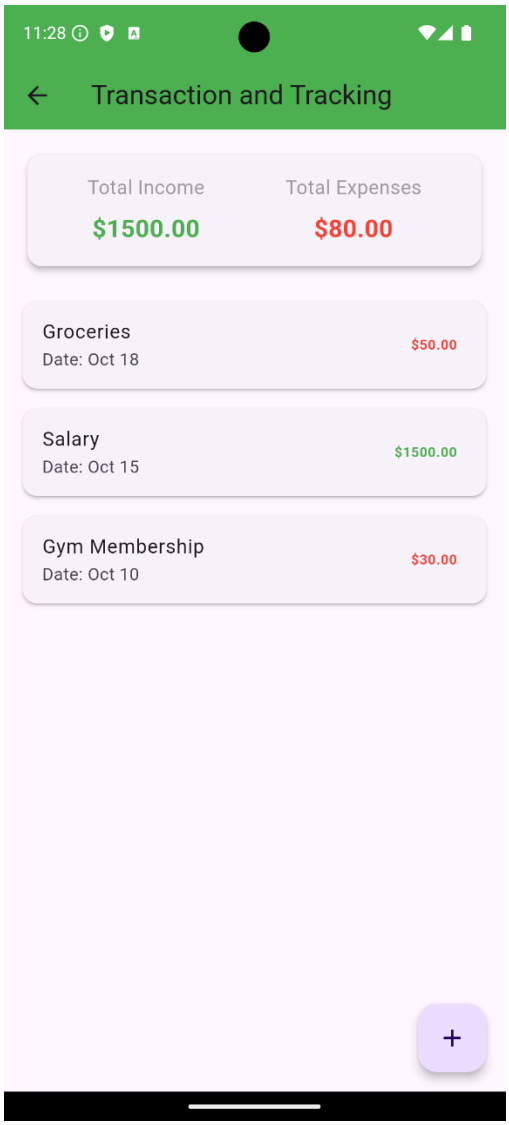
The report and analytics screen displays monthly income and expenses along with percentage changes to give users a quick overview of their financial status. Additionally, an expense breakdown by categories such as housing, food, and utilities is included to help users understand their spending habits in detail. It is adjustable as inputs are being made through the transaction screen.

Below are images of each display:






11:28
Budget App Home
Report and Analytics
Transaction and Tracking
11:28
Budget Analytics
Monthly Income
$5800
9.4%
Monthly Expenses
$4200
4.5%
Income vs Expenses
0 1 2 3 4 5
5.8K
5.5K
5K
4.5K
4K
Expense Breakdown
Housing $2000
Food $800
Transport $600
Utilities $400
Others $600

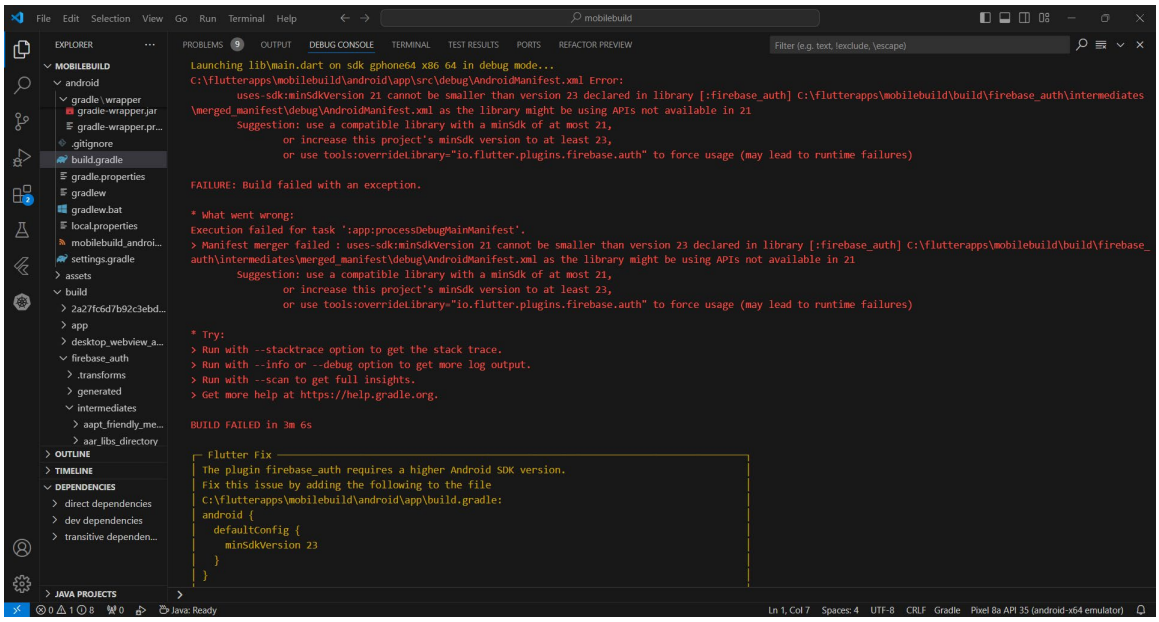


APPENDIX C. CHALLENGES

The biggest challenge was being able to fully install firebase API into the flutter app. There were small complications within each part of the app with the installation which majorly delayed the process of the app.

The main problem was being able to update the firebase authentication to be compatible with the 'minSDK' of the 'build.gradle' within the android properties. It was solved through various troubleshooting, eventually being able to manually update to the correct version.

Below is an image of this debugging challenge:


File Edit Selection View Go Run Terminal Help
mobilebuild
EXPLORER
MOBILEBUILD
android
gradle\wrapper
gradle-wrapper.jar
gradle-wrapper.pr...
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
mobilebuild_androi...
settings.gradle
assets
build
2a27fc6d7b92c3ebd...
app
desktop_webview_a...
firebase_auth
.transforms
generated
intermediates
aapt_friendly_me...
aar_libs_directory
OUTLINE
TIMELINE
DEPENDENCIES
direct dependencies
dev dependencies
transitive dependen...
JAVA PROJECTS
PROBLEMS 9
OUTPUT
DEBUG CONSOLE
TERMINAL
TEST RESULTS
PORTS
REFACTOR PREVIEW
Filter (e.g. text, !exclude, \escape)
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
C:\flutterapps\mobilebuild\android\app\src\debug\AndroidManifest.xml Error:
uses-sdk:minSdkVersion 21 cannot be smaller than version 23 declared in library [:firebase_auth] C:\flutterapps\mobilebuild\build\firebase_auth\intermediates
\merged_manifest\debug\AndroidManifest.xml as the library might be using APIs not available in 21
Suggestion: use a compatible library with a minSdk of at most 21,
or increase this project's minSdk version to at least 23,
or use tools:overrideLibrary="io.flutter.plugins.firebase.auth" to force usage (may lead to runtime failures)
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':app:processDebugMainManifest'.
> Manifest merger failed : uses-sdk:minSdkVersion 21 cannot be smaller than version 23 declared in library [:firebase_auth] C:\flutterapps\mobilebuild\build\firebase_
auth\intermediates\merged_manifest\debug\AndroidManifest.xml as the library might be using APIs not available in 21
Suggestion: use a compatible library with a minSdk of at most 21,
or increase this project's minSdk version to at least 23,
or use tools:overrideLibrary="io.flutter.plugins.firebase.auth" to force usage (may lead to runtime failures)
* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.
> Get more help at https://help.gradle.org.
BUILD FAILED in 3m 6s
Flutter Fix
The plugin firebase_auth requires a higher Android SDK version.
Fix this issue by adding the following to the file
C:\flutterapps\mobilebuild\android\app\build.gradle:
android {
defaultConfig {
minSdkVersion 23
}
}
Java: Ready
Ln 1, Col 7
Spaces: 4
UTF-8
CRLF
Gradle
Pixel 8a API 35 (android-x64 emulator)