

A COMPREHENSIVE REPORT ON 'BUGET' AN APP FOR EXPENSE TRACKING AND FINANCIAL GOAL SETTING

—

CSC 420 PROJECT

BY GROUP 6

GROUP 6 MEMBERS

S/N	MATRIC NO.	NAME
1	18/52HA083	KURANGA, Racheal Omolayo
2	18/52HA085	LAWAL, Ibrahim Olatunji
3	18/52HA086	LAWAL, Ibrahim Oluwaseyi
4	18/52HA087	LAWAL, Khadijat Gbemisoke
5	18/52HA088	LAWAL, Mohammed Olanrewaju
6	18/52HA089	LAWAL, Wahab Babatunde
7	18/52HA090	MADUKA, Olivia Chinwendum
8	18/52HA091	MIKE-SANUSI, David Oluwatobi
9	18/52HA092	MOHAMMED, Mufidah Bibiresanmi
10	18/52HA094	MUHAMMAD, Suleiman Folorunsho
11	18/52HA096	MUIBI, Faruq Boluwatife
12	18/52HA097	MUSA, Abdullahi
13	18/52HA098	ODESOLA, Ibrahim Ajibola
14	18/52HA099	ODODO, Samson



TABLE OF CONTENTS

1.0 INTRODUCTION

2.0 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

3.0 FEASIBILITY STUDY

4.0 REQUIREMENTS ANALYSIS AND SPECIFICATION

5.0 DESIGN

6.0 CODING

7.0 TESTING

8.0 MAINTENANCE

9.0 CONCLUSION

1.0 INTRODUCTION

In an era where financial management plays a pivotal role in our lives, imagine an app that transforms the way you track expenses and chase your financial dreams. **Introducing 'Buget' – not just an app, but your personal finance ally.** This comprehensive report delves into the world of 'Buget,' an innovative application designed to empower you with the tools you need to effortlessly track your expenses, set meaningful financial goals, and embark on a journey towards financial success.

In this project, we aim to design and develop a Personal Budgeting Tool to assist individuals in managing their finances effectively. The tool will enable users to track expenses, set financial goals, and generate insightful reports on their spending habits and savings. To achieve this, we will follow the iterative waterfall model, which allows for sequential development with feedback loops for continuous improvement.

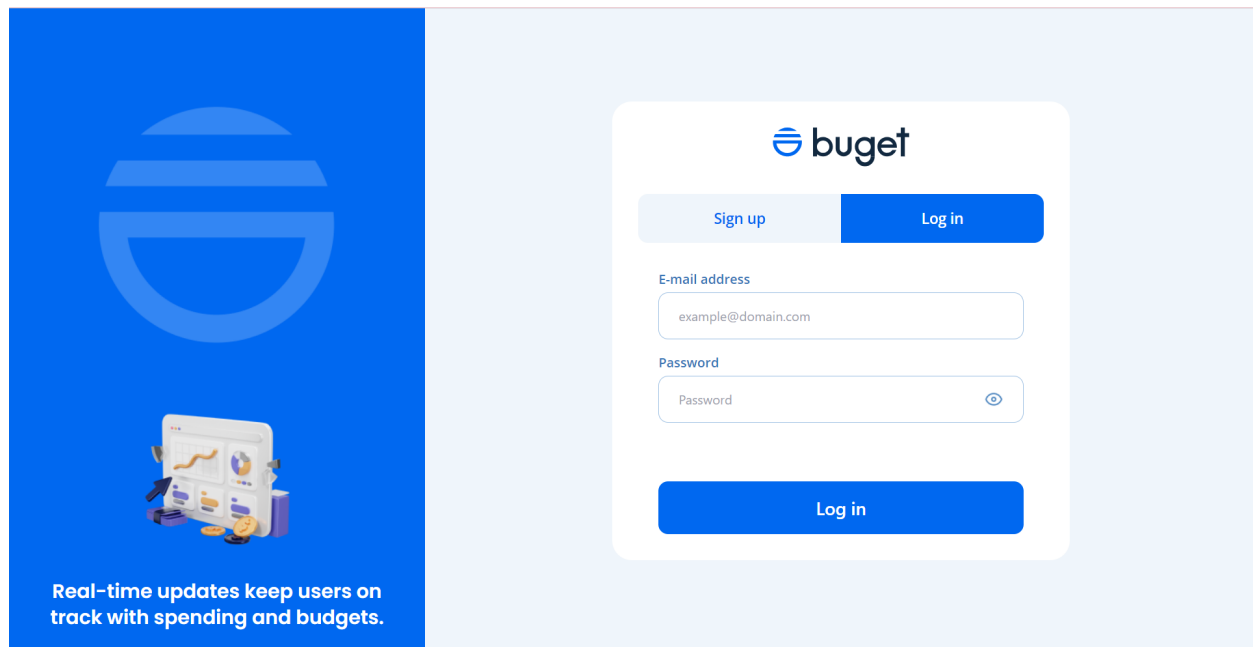


Figure 1. Sign up Page

1.1 Objectives: The Personal Budgeting Tool is a comprehensive web-based application designed to empower individuals in managing their finances effectively. It aims to provide users with a user-friendly platform to track expenses, set financial goals, and generate insightful reports on their spending habits and savings. The project's primary objective is to help users take control of their financial health and make informed decisions to achieve their financial goals.

1.2 Financial Empowerment: The project's core objective is to empower users with the knowledge and tools to make sound financial decisions. By offering a robust budgeting and tracking platform, users can gain better control over their expenses and savings.

1.3 Simplicity and Intuitiveness: The Personal Budgeting Tool focuses on simplicity and intuitiveness to cater to users from diverse financial backgrounds. The user interface will be designed to be easy to navigate, making it accessible to both tech-savvy individuals and those less familiar with financial management applications.

1.4 Insightful Reporting: The tool will generate detailed reports and visualizations, providing users with valuable insights into their financial habits and progress towards their financial goals. The reports will help users identify areas of improvement and make informed decisions for a more stable financial future.

1.5 Scalability and Expandability: While the project starts as an MVP, the design and architecture will be scalable and flexible to accommodate future enhancements and additional features based on user feedback and evolving financial management needs.

1.6 Development Approach: The project will follow the iterative waterfall SDLC model, enabling continuous improvement and feedback loops. Iterative development will allow for early delivery of key features, ensuring timely feedback from users to shape the product's direction.

1.7 Project Timelines: The project timeline is set to ensure the efficient completion of each task. The application is scheduled to commence on the 10th of July, 2023, with the final milestone targeted for the 2nd of August, 2023. The breakdown of the schedule is as follows:

- **Feasibility Study [Duration: 10th July, 2023 – 13th July, 2023]:** During this phase, the project team will conduct a thorough assessment to determine the feasibility of the 'Buget' app. This involves evaluating technical, financial, and operational aspects to ensure that the app is viable and aligns with the project's goals.
- **Requirements Gathering and Analysis [Duration: 14th July, 2023 – 16th July, 2023]:** The project team will engage in extensive research and user interactions to gather detailed requirements for the 'Buget' app. They will document both functional and non-functional requirements, considering user needs, preferences, and industry best practices.
- **Design and Prototyping [Duration: 17th July, 2023 – 19th July, 2023]:** In this phase, the project team will focus on translating the gathered requirements into a user-friendly and visually appealing design. They will collaborate closely with designers to create wireframes, mockups, and prototypes that outline the app's structure and user interactions.
- **Development of the Software [Duration: 20th July, 2023 – 28th July, 2023]:** During this phase, the project team will divide their efforts into front-end and back-end development. Front-end developers will build the user interface and ensure a seamless user experience. Back-end developers will implement the core functionalities, such as expense tracking, categorization, goal setting, and data management.
- **Testing and Project Completion [Duration: 29th July, 2023 – 2nd August, 2023]:** The project team will test the 'Buget' app across various test cases. They will address identified bugs and issues, optimizing the app's performance and user experience. The team will work collaboratively to fine-tune the app, ensuring it meets the highest quality standards before finalizing the project..

2.0 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

A Software Development Life Cycle (SDLC) is a systematic and structured approach used in software development to guide the process of designing, creating, testing, deploying, and maintaining software applications. The SDLC provides a framework for managing the entire software development process from conception to completion, ensuring that the resulting software meets quality standards, fulfills user requirements, and is delivered on time and within budget. The phases of SDLC include:

1. Feasibility Study
2. Requirements Analysis and Specification
3. Design
4. Coding
5. Testing
6. Maintenance

In this project, the software development life cycle model used was the Iterative Waterfall/Incremental Model

The iterative SDLC model was chosen for its adaptability and flexibility in accommodating the changing requirements throughout the development process.. Explained below is a comprehensive description of how the iterative model was employed in the development of this Personal budgeting tool

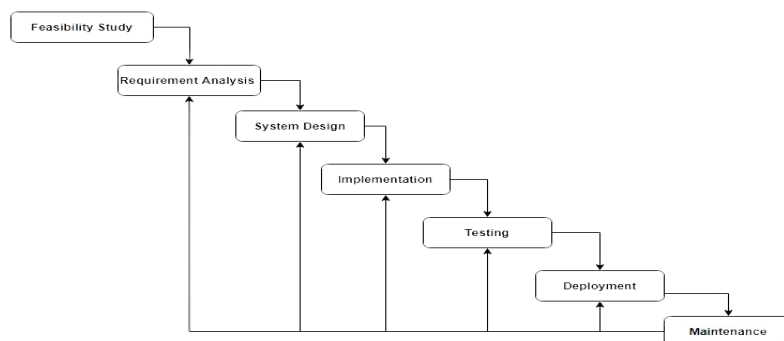


Figure 2. Iterative waterfall Model

3.0 FEASIBILITY STUDY

While conducting a feasibility study for the Personal budgeting tool, several factors were put into consideration to see if the project was viable and worth pursuing. Here's a breakdown of some of the key components that were put into consideration:

3.1 Technical Feasibility:

The technical feasibility study assesses the project's practicality from a technological standpoint. It examines whether the proposed solution is achievable given the available technology, expertise, and resources. In the context of the Personal Budgeting Tool project, technical feasibility played a critical role in determining if the selected technology stack can effectively support the development and implementation of the budgeting application.

- **Technology Stack:** The Personal Budgeting Tool utilizes a modern technology stack to ensure optimal performance and scalability. The frontend was built using Visual Studio Code, Vue JS and Postman. Vue.js, a popular JavaScript library known for its flexibility and responsiveness in creating user interfaces. The backend was developed using Node.js, providing a robust and efficient server-side environment. Other tools used include- Postman, Render, Node.js MongoDB Visual studio code, Express.js, Cors.js, Helmet.js, xss-clean, Winston and MongoDB Compass. MongoDB is a NoSQL database that was employed for secure and efficient data storage and retrieval.
- **Available Expertise:** The technical feasibility of the project was influenced by the availability of skilled developers proficient in the selected technologies. Fortunately, the development team possesses expertise in Node.js, and MongoDB, ensuring that the project's technical requirements can be effectively addressed.
- **Scalability and Performance:** A key consideration for technical feasibility is the potential to scale the application to accommodate a growing user base. The selected technology stack, particularly Node.js and MongoDB, is known for its scalability and ability to handle increasing demands. This ensures that the Personal Budgeting Tool can accommodate a larger number of users as its popularity grows.

In conclusion, the technical feasibility study for the Personal Budgeting Tool project indicated that the selected technology stack, the availability of skilled expertise, scalability, integration capabilities, and security measures all contribute to the project's technical viability. With the proper implementation and adherence to best practices, the technical aspects of the project could be effectively addressed, ensuring the successful development and deployment of the budgeting application.

3.2 Economical Feasibility:

The economic feasibility study evaluates the financial viability of the Personal Budgeting Tool project, considering the costs associated with its development, implementation, and maintenance. As a school project, the economic feasibility assessment focuses on keeping expenses to a minimum while delivering an efficient and functional budgeting application.

- **Development Costs:** As an MVP, the development of the Personal Budgeting Tool primarily relies on open-source technologies, reducing software licensing expenses. The development team utilized free or low-cost development tools and libraries to build the application, minimizing upfront costs.
- **Resource Allocation:** As the project involves school students working collaboratively, the development team members contribute their time and effort voluntarily. While there may be minimal expenses for team meetings or communication, these costs are generally negligible in a school project setting.
- **Infrastructure and Hosting:** The Personal Budgeting Tool was hosted on Netlify- a cloud-based platform, which offers cost-effective hosting solutions.

In conclusion, the economic feasibility study indicated that the Personal Budgeting Tool project was well within the financial capabilities of a school project. The careful selection of cost-effective technologies and resource allocation from team members ensured that the project could be successfully developed and implemented without incurring significant expenses.

3.3 Time Feasibility:

The time feasibility study assesses whether the Personal Budgeting Tool project can be completed within the given time frame, considering the scope of work, development approach, and the availability of resources. As a school project with limited time constraints, effective time management is crucial to ensure the successful delivery of the MVP.

- **Scope Definition:** The project's scope was clearly defined, focusing on the development of an MVP with essential features such as expense tracking, financial goal setting, and insightful reports. By limiting the scope to core functionalities, the development team could better manage the project timeline and meet the deadline.
- **Iterative Waterfall SDLC:** The selected iterative waterfall model enabled the project to be divided into manageable phases. Each phase included specific objectives, milestones, and deliverables. The iterative approach allowed for incremental progress and continuous feedback, ensuring that the MVP could be delivered on time.
- **Resource Availability:** The development team members were committed to the project, and their availability was confirmed. By aligning team members' schedules and dedicating adequate time to the project, the team efficiently collaborated and progressed through each development phase.
- **Prioritization:** During the development process, the team had to prioritize tasks based on their importance and impact on the MVP's core functionalities. This approach ensured that critical components were addressed first, and any potential time constraints could be managed effectively.
- **Project Management:** Effective project management practices, such as setting clear objectives, establishing timelines, and regular progress monitoring, was implemented. Team meetings and communication channels were also utilized to keep everyone informed and on track.

In conclusion, the time feasibility study confirms that the Personal Budgeting Tool project can be completed within the given time frame. With effective project management practices, the project's

time feasibility was well-managed, allowing for the on-time completion of the budgeting application as per the school project requirements.

4.0 REQUIREMENTS ANALYSIS AND SPECIFICATION

The requirements analysis and specification phase of the 'Buget' app project involved a thorough examination of user needs and expectations to define clear and detailed software requirements. This phase served as the foundation for the app's development, ensuring that it addresses users' financial management challenges effectively. The project team conducts user interviews, surveys, and research to gather insights, then translates these findings into the app's functionalities, features, and constraints. The goal was to create a solid roadmap that guides the subsequent design and development stages, aligning the app with user expectations and project objectives.

4.1 Requirements gathering and analysis

Requirement gathering is a crucial phase in the development of the Personal Budgeting Tool, where the project team identifies and documents the functionalities and features needed to meet users' needs. The team made use of various efficient approaches to gather requirements which includes:

- **Exploratory Research:** To kickstart the requirement gathering process, the team conducted exploratory research to understand common financial management challenges faced by individuals. Online articles, blogs, and forums were reviewed to gain insights into users' pain points related to budgeting and financial planning.
- **User Surveys:** The team conducted brief surveys amongst friends, family, and peers. The results allowed the team to gather qualitative feedback about personal finance management practices, preferred features, and potential pain points.



Figure 3. User Survey Results

- Prioritization:** Given the limited time available for the project, the team focused on prioritizing core functionalities. Essential features such as expense tracking, setting financial goals, and generating reports were identified as top priorities based on the exploratory research and user interviews.
- Collaboration with End Users:** As the project team was part of the target audience (students), the members actively collaborated with each other to provide input and feedback during the requirement gathering process. This internal feedback loop ensured that the development aligned with the needs of the intended users.

In conclusion, the requirement gathering process for the Personal Budgeting Tool was conducted efficiently, considering the severe time constraint and absence of formal surveys. Exploratory research, user interviews, and internal collaboration provided valuable insights, helping the team prioritize the most critical features.

4.2 Requirements specification

This section outlines the specific functionalities and features that the app will offer, such as expense tracking, automated categorization, financial goal setting, analytics, and reporting. Requirement specification ensured that the app's development team, including designers and developers, have a clear understanding of what needs to be built. Below are the requirements gathered:

a. Functional Requirements

Requirement ID	Requirement Description
FR1	Users should be able to create new accounts on the platform by providing their personal details such as name, email address, and password. Upon successful sign-up, users will gain access to their personalized dashboard.
FR2	Registered users should be able to log into their accounts using their email and password.
FR3	Logged-in users should be able to securely log out from their accounts to prevent unauthorized access.
FR4	Users should have the ability to manually record various sources of income, such as salary, allowances, or other earnings, to accurately track their financial inflow.
FR5	Users should be able to set financial goals for saving, spending, or debt reduction.

FR6	Users should be able to record their expenses by entering details about the transaction, such as amount, category, date, and description.
FR7	Users should have the ability to create a budget plan by allocating specific amounts to different expense categories. This will help users manage their spending and financial priorities.
FR8	Users should be able to generate comprehensive reports that summarize their progress towards financial goals and provide insights into their spending habits.
FR9	Users should have access to visual representations of their expense data in the form of charts or graphs, allowing for easy visualization of spending patterns.

b. Non-Functional Requirements

Now, there are some other requirements that are not technical per se but also of importance, these are:

1. The user interface should be intuitive and user-friendly, allowing users of varying technical backgrounds to navigate and use the app easily.
2. The app should have responsive design, adapting seamlessly to different screen sizes and orientations on various devices.
3. The app should have fast response times for user interactions, ensuring that actions like data entry and report generation occur quickly.
4. The app should be available and accessible to users consistently, with minimal downtime for maintenance.
5. Data integrity and accuracy should be maintained at all times, preventing loss of user data.
6. User data, including personal financial information, should be stored securely and encrypted to prevent unauthorized access.

7. User authentication and authorization mechanisms should be robust, ensuring that only authorized users can access specific features and data.

5.0 DESIGN

During the design phase of the budgeting tool, the project team focused on creating an intuitive and user-friendly interface that empowers individuals to manage their finances effectively. This phase involved translating the requirements gathered during the initial stages into a concrete design, ensuring that the tool aligns with users' needs and expectations.

5.1 Iterative Prototyping: The iterative prototyping process was a key aspect of the design phase, allowing the project team to continuously improve the user interface of the budgeting tool based on feedback from team members. This iterative approach ensured that the final design met user needs, enhanced usability, and addressed any potential issues or challenges.

- **Initial Prototyping:** The design process began with the creation of initial prototypes, including wireframes and low-fidelity mockups. These prototypes were used to visualize the layout and basic functionality of the budgeting tool.
- **Internal Feedback Sessions:** The project team conducted regular internal feedback sessions, where team members reviewed the prototypes and provided valuable insights. These sessions encouraged open discussions and brainstorming, allowing the team to identify areas for improvement and potential enhancements.
- **Usability Testing:** To validate the design decisions and assess the tool's usability, the project team conducted usability testing with team members acting as end-users. Usability testing involved simulating real-life scenarios to observe how team members interacted with the prototype and identify any usability issues.
- **Gathering User Feedback:** The project team also sought feedback from potential end-users outside of the development team. This feedback was collected through informal discussions and surveys, allowing the team to gain perspectives from individuals who would ultimately use the budgeting tool.

- **Incorporating Feedback:** Based on the feedback received from both team members and potential end-users, the project team made iterative improvements to the user interface. Changes ranged from minor adjustments to major redesigns, all aimed at enhancing user experience and addressing pain points.
- **Validation and Testing:** After each round of iterative improvements, the project team validated the changes through further usability testing and feedback sessions. This validation process ensured that the refinements were effective and aligned with user expectations.
- **Continuous Refinement:** The iterative prototyping process was repeated multiple times throughout the design phase. Each iteration brought the user interface closer to the final design, incorporating valuable insights from feedback sessions and usability testing.
- **Final Design Selection:** By the end of the iterative prototyping process, the project team collectively arrived at a final user interface design that encompassed the most effective and user-friendly elements. This design was chosen based on its usability, visual appeal, and alignment with project requirements.

5.2 User Interface Design: The Design team began by sketching wireframes and creating mockups to visualize the layout and flow of the budgeting tool. Special attention was given to simplicity and clarity to ensure users can easily navigate and understand the application.

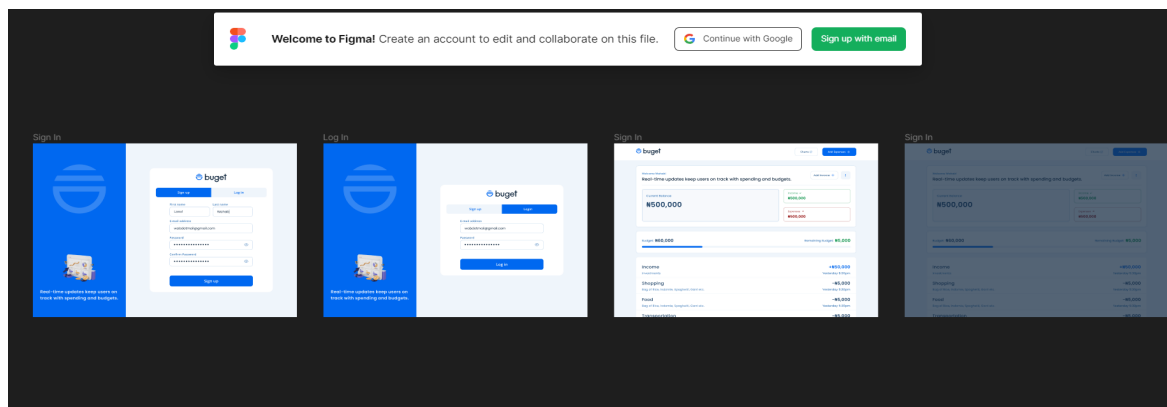


Figure 4. Budget Wireframes

- **Expense Tracking Dashboard:** The core feature of the budgeting tool is an interactive dashboard that allows users to input and categorize their expenses. The dashboard displays an overview of spending patterns, enabling users to track their financial health at a glance.

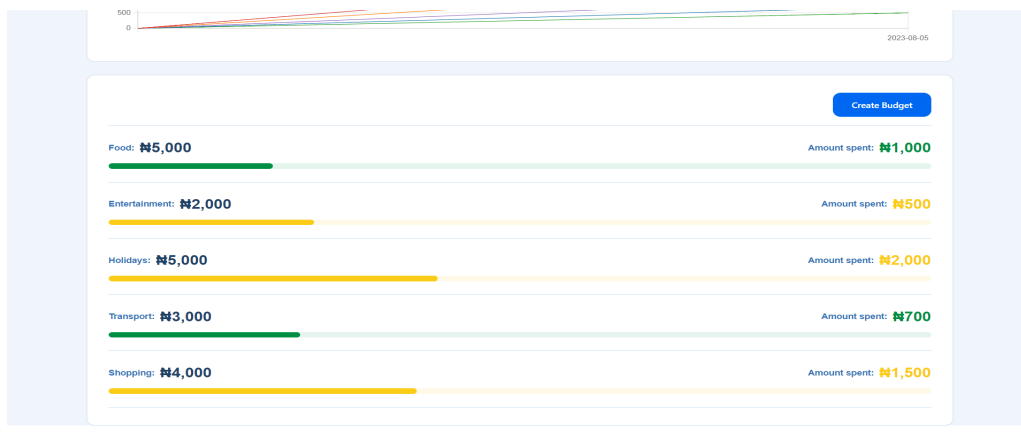


Figure 5. Expense Tracking Dashboard

- **Financial Goal Setting:** To assist users in achieving their financial goals, the design included a dedicated section for setting and managing personal financial targets. Users can input their goals, such as saving for a vacation or paying off debts, and track their progress over time.

Add New Goal

Description:

Amount:

Target Date:

Start amount (optional):

Add New Goal

Figure 6. Financial Goal Setting page 1

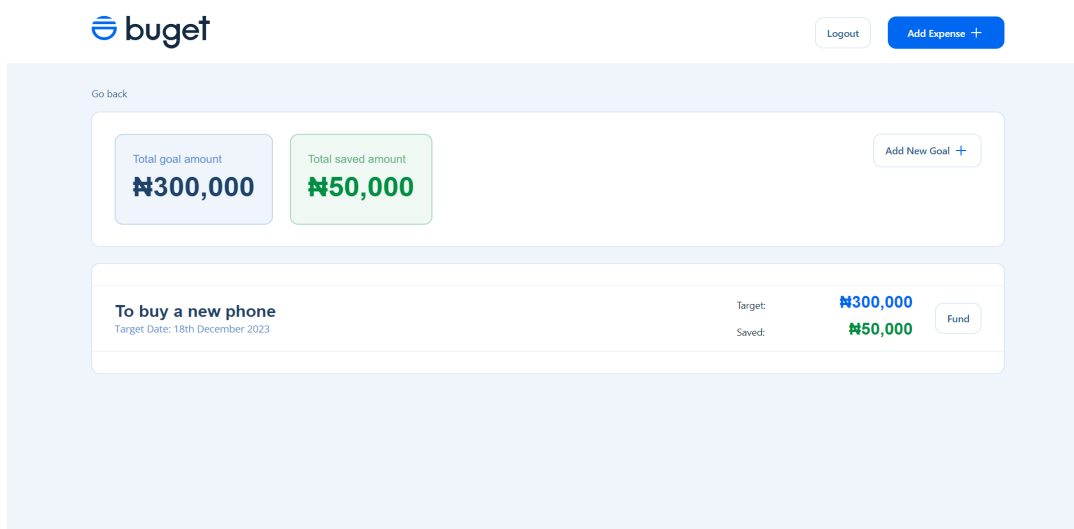


Figure 7. Financial Goal Setting page 2

- **Reports and Insights:** To enhance financial awareness, the tool generates detailed reports and insights based on users' spending habits and savings. These reports offer a clear picture of expenses and identify areas where adjustments can be made to improve financial planning.

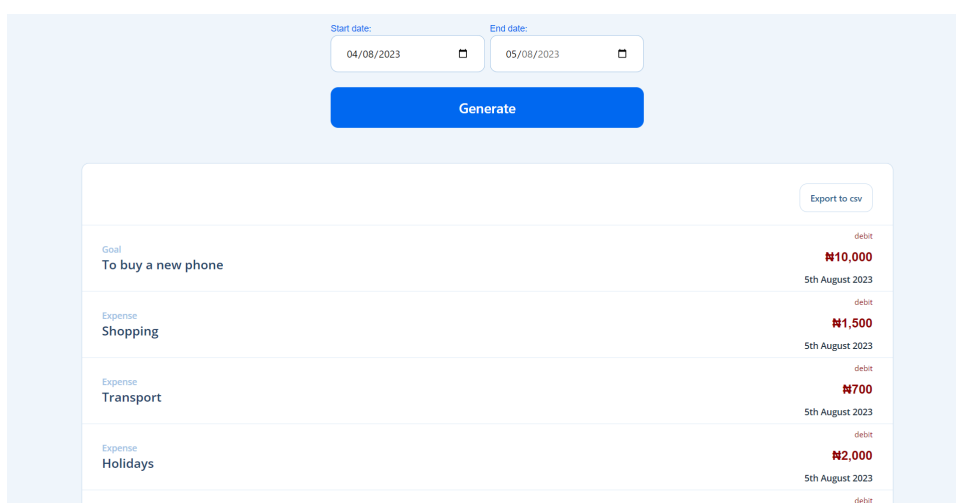


Figure 8. Report Generation

- **Visualizations and Charts:** To enhance data comprehension, the budgeting tool incorporated visualizations and charts. Pie charts and bar graphs were used to present expense distributions and savings progress, making financial data more accessible and engaging.

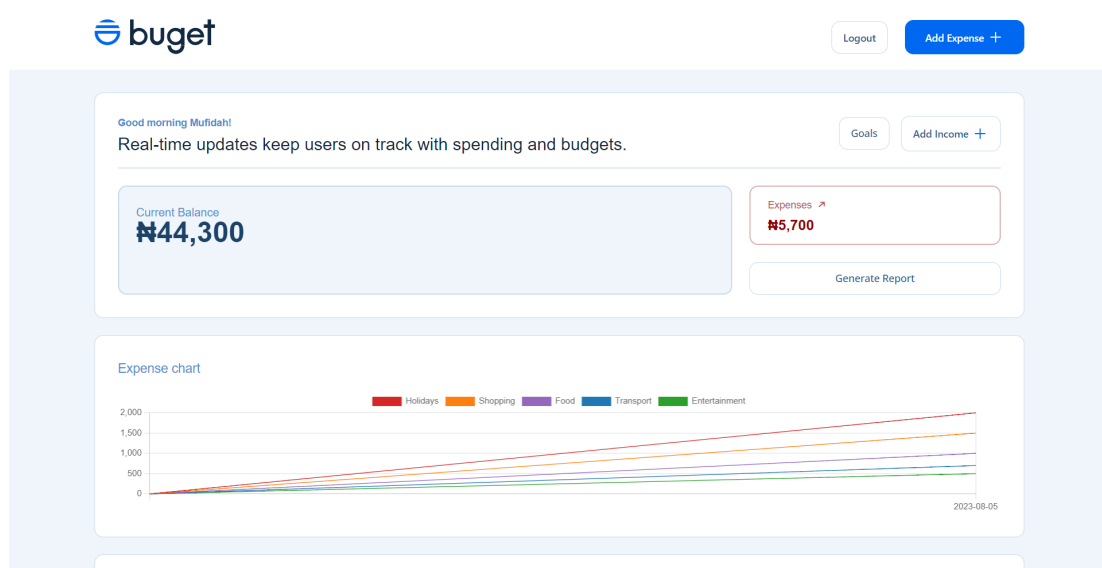


Figure 9. Expense Chart

In conclusion, the design phase of the budgeting tool was focused on creating a user-centered and feature-rich application. The user interface was designed with simplicity and personalization in mind, while data visualizations and reports provide valuable insights for financial management. The iterative approach and usability testing ensured that the design resonates with users, offering a comprehensive and intuitive budgeting solution.

6.0 CODING

The coding phase of the 'Buget' app marks the transition from design concepts to functional reality. It's during this pivotal stage that the carefully crafted designs and user experience came to life through lines of code. This phase was the heart of the software development process, where the vision of an app for expense tracking and financial goal setting transformed into a tangible digital solution.

6.1 Transforming Designs into Code: The meticulous UI/UX designs from the previous phase serve as the blueprint for the coding process. Our talented software developers translated these designs into code, utilizing programming languages and frameworks that best suit the app's requirements.

6.2 Front-End Development: Front-end developers focused on creating the visual elements that users interact with directly. They utilized a combination of languages and frameworks to craft the user interface and interactions according to the design specifications, ensuring a seamless and intuitive experience. The key technologies employed include:

- **HTML and CSS:** These foundational languages are used to structure the content and style the visual elements of the application's user interface.
- **Vue.js:** A progressive JavaScript framework, Vue.js, is employed to build dynamic and interactive front-end components, enhancing user engagement and responsiveness.
- **Visual Studio Code:** This integrated development environment (IDE) provides a robust platform for front-end developers to write, edit, and debug code efficiently.
- **Postman:** The Front-end developers use Postman to test and interact with APIs, ensuring smooth communication between the front-end and back-end components.

These technologies collectively enable front-end developers to create a visually appealing, user-friendly, and functional user interface that aligns with the design vision and enhances the overall user experience.

6.3 Back-End Development: Back-end developers handled the behind-the-scenes functionalities that power the app. This included setting up databases to store expense records, and creating APIs that allow the front-end and back-end components to communicate. The Back-end developers employed a range of languages and frameworks to manage the underlying functionalities that drive the application. The prominent technologies utilized in this context comprise:

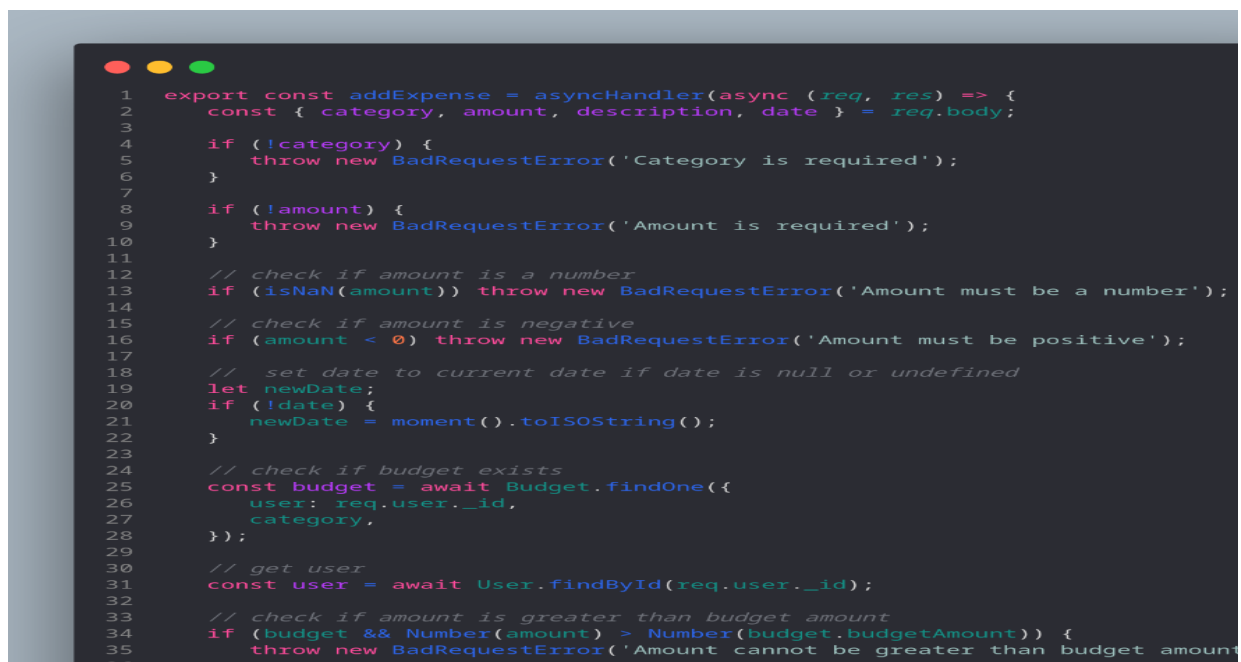
- **Node.js:** A runtime environment that enables back-end developers to execute JavaScript on the server-side, facilitating efficient and scalable application development.
- **Express.js:** A minimalistic web application framework for Node.js, Express.js simplifies the creation of robust APIs and routing structures.
- **MongoDB:** A NoSQL database, MongoDB stores and manages expense records effectively, providing flexibility in data storage.
- **MongoDB Compass:** A graphical user interface for MongoDB, MongoDB Compass aids in database management and visualization.
- **Visual Studio Code:** A versatile integrated development environment (IDE), Visual Studio Code offers tools for back-end developers to write, edit, and debug code efficiently.
- **Cors.js:** Middleware for Express.js, Cors.js ensures secure cross-origin resource sharing, enabling controlled data exchange between different domains.
- **Helmet.js:** A security-focused middleware, Helmet.js enhances application security by setting various HTTP headers.
- **xss-clean:** A module that prevents cross-site scripting (XSS) attacks, xss-clean mitigates vulnerabilities related to user input.
- **Winston:** A logging library for Node.js, Winston facilitates proper error handling and debugging.
- **Postman:** Back-end developers utilize Postman to test and validate APIs, ensuring seamless communication between front-end and back-end components.

- **Render:** Render provides cloud hosting services, allowing back-end developers to deploy and manage the server-side components of the application.

By leveraging these technologies, the back-end developers created a robust, secure, and efficient foundation that supports the overall functionality of the application while maintaining data integrity and communication with the front-end.

6.4 Implementing Functionality: The 'Buget' app's core functionalities were brought to life during the coding phase. These functionalities include:

- **Expense Tracking:** Developers wrote codes to allow users to enter expenses and categorize them.



```

1  export const addExpense = asyncHandler(async (req, res) => {
2    const { category, amount, description, date } = req.body;
3
4    if (!category) {
5      throw new BadRequestError('Category is required');
6    }
7
8    if (!amount) {
9      throw new BadRequestError('Amount is required');
10   }
11
12   // check if amount is a number
13   if (isNaN(amount)) throw new BadRequestError('Amount must be a number');
14
15   // check if amount is negative
16   if (amount < 0) throw new BadRequestError('Amount must be positive');
17
18   // set date to current date if date is null or undefined
19   let newDate;
20   if (!date) {
21     newDate = moment().toISOString();
22   }
23
24   // check if budget exists
25   const budget = await Budget.findOne({
26     user: req.user._id,
27     category,
28   });
29
30   // get user
31   const user = await User.findById(req.user._id);
32
33   // check if amount is greater than budget amount
34   if (budget && Number(amount) > Number(budget.budgetAmount)) {
35     throw new BadRequestError('Amount cannot be greater than budget amount');
36   }
37
38   // create expense
39   const expense = await Expense.create({
40     user: req.user._id,
41     category,
42     amount,
43     description,
44     date: newDate,
45   });
46
47   res.status(201).json(expense);
48 }

```

Figure 10. Code snippet for Expense Tracking

- **Financial Goal Setting:** Code was implemented to enable users to set financial goals, specifying target amounts, dates, and types of goals.

```

1  export const createGoal = asyncHandler(async (req, res) => {
2    const { goalAmount, targetDate, description, creditAmount } = req.body;
3
4    // Check if user balance is greater than targetDate
5    const userFromDb = await User.findById(req.user._id);
6
7    if (creditAmount > Number(userFromDb.balance)) {
8      throw new BadRequestError(
9        'Credit amount cannot be greater than user balance'
10     );
11   }
12
13   // create a new goal
14   const goal = await new Goal({
15     user: req.user._id,
16     goalAmount: goalAmount,
17     targetDate: moment(targetDate).toISOString(),
18     description: description,
19     savedAmount: +creditAmount,
20   }).save();
21
22   // debit user balance
23   const user = await User.findById(req.user._id);
24   user.balance -= Number(creditAmount);
25   user.save();
26
27   // create new report
28   if (creditAmount) {
29     await new Report({
30       user: req.user._id,
31       reportType: 'Goal Report',
32       date: moment().toISOString(),
33       summaryStatistics: {
34         transactionId: goal._id,
35         category: goal.description,
36         amount: creditAmount,
37         type: 'debit',
38       },
39     }).save();
40   }
41
42   res.status(StatusCodes.CREATED).json({
43     message: 'Goal created successfully',
44     goal,
45   });
46 });

```

Figure 11. Code snippet for Goal Setting

- **Analytics and Reporting:** Code was written to generate graphs, charts, and reports that provide insights into users' spending habits, savings progress, and financial performance.

```

1  export const generateExpenseReport = asyncHandler(async (req, res) => {
2    const { startDate, endDate } = req.body;
3
4    if (!startDate || !endDate)
5      throw new BadRequestError('Start and End date is required');
6
7    if (endDate < startDate)
8      throw new BadRequestError('endDate cannot be earlier than startDate');
9
10   const startDateq = moment(startDate).startOf('day').toISOString();
11   const endDateq = moment(endDate).endOf('day').toISOString();
12
13   const report = await Report.find(
14     {
15       user: req.user._id,
16       date: {
17         $gte: startDateq,
18         $lte: endDateq,
19       },
20     },
21     { __v: 0 }
22   ).sort({ date: 'asc' });
23
24   const reports = {
25     data: report,
26     startDateq,
27     endDateq,
28     length: report.length,
29   };
30
31   res.status(StatusCodes.OK).json({
32     reports,
33   });
34 });

```

Figure 12. Code snippet for Report Generation

6.5 Ensuring Data Security and Privacy: Developers prioritize the security of users' financial data. They implement a bcrypt package for hashing the password which makes use of Blowfish cipher algorithm to protect sensitive information and ensure that user data remains private and confidential.

6.6 Iterative Development: Coding is an iterative process. Developers frequently collaborate with designers, testers, and other team members to ensure that the code aligns with the intended functionality and user experience. Regular code reviews and testing help identify and rectify issues early in the development process.

6.7 Quality Assurance: Throughout the coding phase, quality assurance (QA) engineers test the app to identify bugs, glitches, and inconsistencies.

The coding phase was the heartbeat of the 'Buget' app's development, where lines of code give life to designs and functionalities. Through meticulous coding, iterative development, and rigorous testing, the 'Buget' app inched closer to becoming a valuable tool that champions financial control.

7.0 TESTING

At the testing phase of the 'Buget' app project. The testing team aimed at ensuring the app's reliability, functionality, security, and user experience. This phase encompassed a range of testing methodologies and activities to identify and rectify any issues before the app's deployment.

7.1 Testing Strategies: The testing team employed various testing methodologies during this phase. They include:

- **Unit Testing:** Here we tested individual components to ensure their correctness.

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	1.Go to site https://grp6-budge-tapp.netlify.app/ 2.Enter Email Address 3.Enter Password 4.Click Login	Email Address= Ywzmail@gmail.com Password= polarisbank1	User should Login into an application	As Expected	Pass
TU02	Check Customer Login with invalid Data	1.Go to site https://grp6-budge-tapp.netlify.app/ 2.Enter Email Address 3.Enter Password 4.Click Login	Email Address=Ywzmail@gmail.com Password= polaris1	User should not Login into an application	As Expected	Pass
TU03	Adding Income	1.Click on Add Income on the dashboard 2. Input amount 3.Click on Add to balance	Amount = 250,000 And Amount = 50,000.123456	Should add input amount to current balance	As Expected	Pass

TU04	Setting Goals	1.Click on goals on the dashboard 2.Go to goals page 3.Click on add new goals 4.Input description 5.Input amount 6.Set Target date 7.(optional) input start amount 8.Click on the Add new goal	Description = New laptop Amount= 200000 Target date = 1 Jan 2024 Start amount = 0	Should add input amount into total goal amount. Should add input start amount into total saved amount.	As Expected	Pass
TU05	Adding Expenses	1.Click on add expense on the dashboard 2.Choose category 3. Input amount 4. Input description 5. Click on add expense	Category = bills Amount = 25000 Description = electricity bills	Should deduct input amount from current balance. Should update the expenses tab on the dashboard by adding input amount to it. Should show on the expense chart the expense added.	As expected	Pass
TU06	Creating budget	1.Click on create budget on the dashboard 2.Choose category 3. Input amount 4.Click on add budget	Category = bills Amount = 30,000	Should add the created budget to the dashboard(budget name and amount spent).	As expected	Pass
TU07	Generate report	1. Click on generate report on the dashboard 2.Go to generate report page 3. Choose start date 4.Choose end date 5.Click on generate	Start date = 08/03/23 End date = 08/04/23	Should show all transactions from each feature that occurred from start date to end date.	As expected	Pass

TU08	Logging out	1.Click on logout button on the dashboard		Should take user back to login page	As expected	Pass
------	-------------	---	--	-------------------------------------	-------------	------

- **Integration Testing:** We verified interactions between different components to ensure seamless operation.

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Adding expense and creating a budget	<p>Add Expense:</p> <ol style="list-style-type: none"> 1. Click on add expense on the dashboard 2.Choose category 3. Input amount 4.Input description 5.Click on add expense <p>Create budget for expense:</p> <ol style="list-style-type: none"> 1. Click on create budget on the dashboard 2.Choose category 3.Input amount 4.Click on add budget 	<p>Category = bills</p> <p>Amount = 25000</p> <p>Description = electricity bills</p> <p>Category = bills</p> <p>Amount = 30,000</p>	Visual representation showing budgeted amount and amount spent for chosen category.	As Expected	Pass
TU02	Funding goals	<p>(after setting goal)</p> <ol style="list-style-type: none"> 1.Click on fund button next to goal 2.Input amount 3.Click on add to "selected goal" 	Input amount = 50,000	<p>Input amount should be added to the total saved amount on the goals page.</p> <p>Input amount should be deducted from the current balance on the dashboard.</p>	As Expected	Pass
TU03	Adding expense and expense chart	1.Add expense	<p>Category = bills</p> <p>Amount = 25000</p> <p>Description = electricity bills</p>	Expense added should reflect on the chart	As expected	Pass

- **System Testing:** Our final goal here was to ensure that the developed system conforms to its requirements.
 - **Alpha Testing:** The application was tested by the development team.
 - **Beta Testing:** The application was tested by some of our target audience.
 - **Acceptance testing:** The application was tested by some set of customers to determine whether to accept or reject the delivered product.

7.2 Use-case Diagram

A use case diagram is used to illustrate how a system is dynamic. They are used to compile a system's needs, considering both internal and external factors. Most of these needs are for the design. Consequently, when a system is examined, actors are chosen, use cases are created, and its functionalities are gathered.

In a nutshell, use case diagrams are used for the following reasons:

1. Employed to compile a system's needs.
2. Employed to obtain a system's external perspective.
3. Determine the system's external and internal affecting forces.
4. Show how the requirements interact with the actors

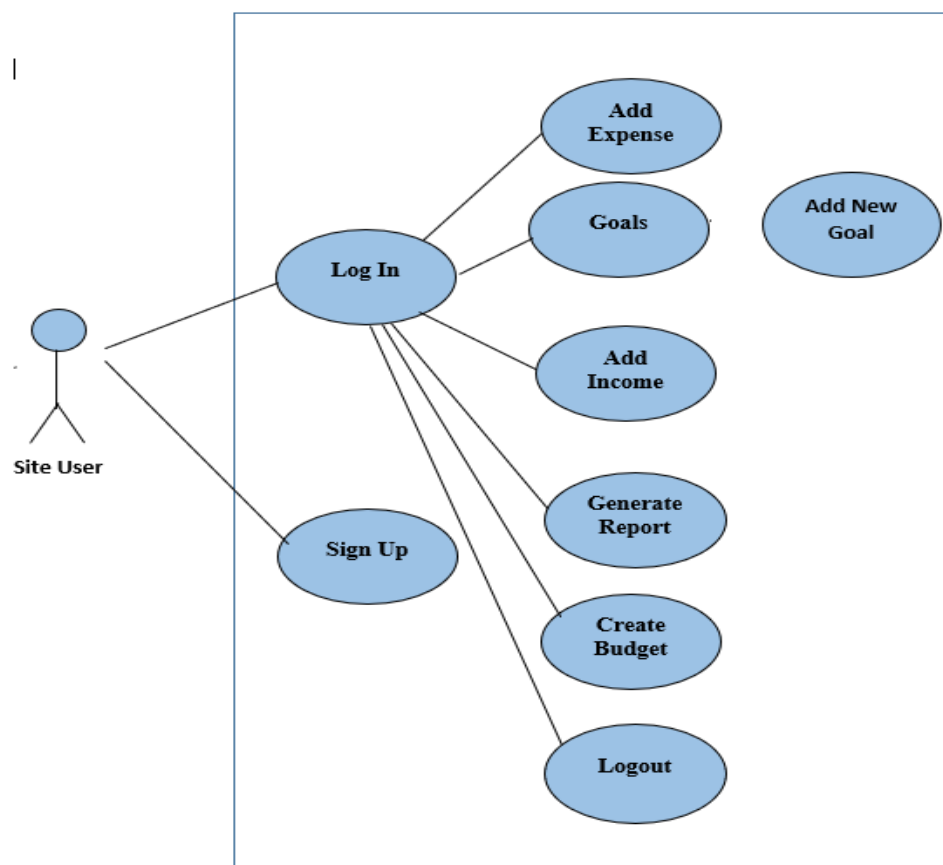


Figure 14. Use-case Diagram

8.0 MAINTENANCE

Buget App's journey doesn't end with the app's launch. In this phase, the project team will be dedicated to nurturing, enhancing, and evolving the app to ensure that users continue to experience its full potential. Our maintenance plan will be revolved around this 3 key activities:

8.1 Corrective maintenance: Just as a well-tended garden requires vigilant attention, our maintenance team will promptly identify, categorize, and address any bugs or glitches that users encounter. We understand that a seamless experience is vital for user satisfaction.

8.2 Perfective maintenance: We recognize that user needs and preferences can evolve over time. Our maintenance team will actively monitor user interactions, analyze feedback, and conduct surveys to identify areas for improvement.

8.3 Adaptive maintenance: The 'Buget' app is designed to adapt to user needs. We'll introduce new features and functionalities aligned with evolving financial management trends, ensuring the app remains a valuable tool for users' goals.

In the Maintenance phase, our focus remains unwavering: to create an app that not only meets but exceeds users' expectations. With continuous improvements, rigorous testing, and an unyielding dedication to excellence, we ensure that the 'Buget' app is not just a tool but a trusted companion in the pursuit of financial well-being.

9.0 CONCLUSION

In this project, we embarked on a mission to create a tool that resonates with users' aspirations for better financial control. Through meticulous planning, collaborative efforts, and unwavering dedication, we've successfully designed an app that allows individuals to track their expenses, set meaningful financial goals, and gain valuable insights into their spending habits and savings patterns.