המכללה האקדמית להנדסה
אורט בראודה
ORT Braude College

personal finance management application

SpendSmart

Capstone Project Phase B - 23-2-D-3

Authors:

Wail Otman – Waelotman211@gmail.com

Ameer Shalata – amer12303g@gmail.com

Supervisor:

Dr. Ronen Zilber – RonenZ@braude.ac.il

TABLE OF CONTENT

1. Abstract

SpendSmart is a user-friendly financial management app that puts you in control of your money. It uses a smart budgeting algorithm to divide your money wisely among different goals, so you can work towards multiple goals at the same time. The app also lets you invest some of your money for long-term growth and sends you reminders and notifications to stay on track.

You can even explore loan options within the app to see how borrowing money would affect your overall financial plan. SpendSmart gives you a clear picture of the loan details, including how much you'll need to pay each month and the total amount you'll repay over time. This helps you make informed decisions about taking out a loan to reach your goals faster, while considering the financial impact.

With SpendSmart, you can easily set your goals, track your progress, and get a comprehensive overview of your finances, including savings, goal progress, and investment performance. It's like having a personal financial assistant that helps you make smart decisions and achieve your goals efficiently, giving you a brighter financial future.

2. Introduction

General Description:

Application:

The Personal Finance Management application is a software engineering project that aims to help users manage their personal finances efficiently. The application will allow users to input their income, expenses, and financial goals. The algorithm will provide personalized recommendations for budgeting, saving money, and tracking progress towards financial goals. The application will also provide investment recommendations, utilizing portfolio optimization algorithms, and tracking the performance of the user's investment portfolio.

Character:

The problem addressed by this project is the difficulty that many people face in managing their personal finances effectively. Many people struggle to budget their income, save money, and track their progress towards financial goals. While there are various financial management applications available in the market, none of them provide a personalized approach to financial management. The need for a more personalized approach to financial management forms the basis for the development of this project.
SpendSmart is characterized by its user-centric design, seamless functionality, and commitment to empowering users with the knowledge and tools they need to make informed financial decisions. Our sign-in/sign-up feature ensures that your financial data remains secure and accessible only to you.

Identification of the User Audience:

SpendSmart caters to a diverse audience of individuals seeking to improve their financial literacy and management skills. Whether you're a college student managing your student loans, a young professional saving for your first home, or a retiree planning for your golden years, SpendSmart is here to help. Our user-friendly interface and personalized recommendations make managing your finances a breeze, no matter where you are on your financial journey.

Current financial apps don't give personalized advice for budgeting and saving, making it hard for users to manage money effectively. They also overlook individual risk tolerance and goals when suggesting investments. We aim to fill this gap with a personalized app that tailors' recommendations to each user.

Proposed Solution Summary:

We're creating a Personal Finance Management app that uses algorithms to give personalized advice on budgeting and investing. Users can input their income, expenses, and goals, and the app will analyze the data to offer tailored recommendations for budgeting, saving, and investing. It also lets users track their progress towards financial goals and monitor their investment portfolio's performance.

3. Background

Personal finance management has become increasingly important in today's society as individuals and households strive to navigate the complexities of managing their income, expenses, and investments. In an era of expanding financial markets, diverse products, and evolving services, it has become crucial for individuals to have the means to take control of their financial affairs and make informed decisions.

Traditionally, managing personal finances involved manual tracking of income and expenses, relying on spreadsheets, and seeking professional financial advice. However, with the rapid advancement of technology, there is a growing demand for intuitive and user-friendly software tools that can streamline the financial management process.

These tools provide users with a centralized platform to track their income, monitor expenses, and analyze spending patterns. They enable users to set financial goals, such as saving for a house, paying off debt, or planning for retirement, and provide features that help users allocate their resources efficiently.

Moreover, the complexities of modern financial markets require individuals to stay informed about investment opportunities, manage portfolios, and assess risk. Financial management software can offer investment tracking, portfolio analysis, and real-time market data, empowering users to make well-informed investment decisions and optimize their returns.

In addition to income and expense management, personal finance software can also assist users in budgeting, debt management, and long-term financial planning. These tools provide visualizations and reports that allow users to gain insights into their financial health, identify areas for improvement, and make adjustments to their financial strategies accordingly.

By utilizing such software, individuals can gain a comprehensive overview of their financial situation, make informed decisions, and work towards achieving their financial goals. With the convenience and accessibility of personal finance management applications, users can take charge of their financial well-being, improve their financial literacy, and secure a stable financial future.

Overall, the increasing complexity of financial management necessitates the use of software tools that simplify the process and empower individuals to take control of their finances. By leveraging these tools, users can make informed decisions, optimize their financial resources, and ultimately achieve their financial aspirations.

4. development process:

4.1. Requirements Discovery:

Rather than gathering requirements directly from a client, we conducted extensive market research and analysis to identify the key features, functionalities, and goals of the SpendSmart system. This involved studying existing financial management apps, conducting surveys, and analyzing user feedback to understand the needs and points of potential users.

4.2. Planning and Design:

Based on the insights gained from the requirements discovery phase, we formulated a plan and created a detailed design for the system. This included defining the architecture, database schema, user interface wireframes, and API specifications.

4.3. Prototyping:

Prototypes or mockups of the user interface were created to visualize the design and gather feedback. This iterative process helped refine the user experience and ensure that the final product met the needs and expectations of the target audience.

4.4 Development:

With the design in place, development of the system commenced. We followed agile methodologies to manage tasks, iterate on features, and adapt to changing requirements. Code was written, reviewed, and tested collaboratively to ensure quality and maintainab ility.

4.5 Backend Development:

Backend development involved implementing the server-side components of the system using technologies like Node.js and Express.js. This included setting up the server, implementing API endpoints for data exchange, integrating with the MongoDB database, and implementing algorithms for financial analysis and recommendations.

4.6 Frontend Development:

Frontend development focused on building the user interface of the mobile application using React Native. This involved creating screens, components, and navigation flows based on the design specifications. UI elements were styled using CSS or styled-components to achieve a cohesive and visually appealing look.

4.7 Integration and Testing:

Once both backend and frontend components were developed, they were integrated to form the complete system. Integration testing was performed to ensure that all parts of the system worked together seamlessly. Unit tests, integration tests, and end-to-end tests were conducted to validate functionality, detect bugs, and ensure reliability.

4.8 Monitoring and Maintenance:

After deployment, the system was monitored for performance, stability, and security. Any issues or bugs discovered were addressed promptly through bug fixes and updates. Regular maintenance activities, such as database backups and security patches, were pe rformed to keep the system running smoothly.

5. Algorithms

5.1. Budgeting algorithms:
Budgeting algorithms are a type of algorithm used in personal finance management applications to provide personalized recommendations for budgeting and saving money. These algorithms take into account the user's income, expenses, financial goals, and other relevant factors to create a customized budget plan.
Budgeting algorithms can also incorporate other factors, such as the user's credit score, investment portfolio, and financial history, to create a more personalized budget plan. Ultimately, the goal of these algorithms is to help users manage their finances more effectively, achieve their financial goals, and improve their overall financial health. In our project we will use the Reverse Budgeting algorithm.

5.1.1. Some Of Budgeting algorithms:

5.1.1.1 Zero-Based Budgeting:
The zero-based budget or Every Dollar budget is a method where every dollar that comes into your bank account is given a specific purpose or job. When you receive money, you allocate it towards specific expenses or savings goals, ensuring that all your income is accounted for. The objective is to reach a point where there is no money left unallocated, meaning you have assigned all your income to different categories. This budgeting approach helps you develop a habit of saving money and prioritizing your financial goals. By using this budget, you gain a clear understanding of where your money is going and can make informed decisions about your spending.

5.1.1.2. Static Budgeting:
A static budget is a pre-planned budget that uses predicted amounts for a specific period without adjusting for changes in revenue and expenses. It remains fixed regardless of any deviations in actual financial outcomes. It is commonly used to assess the performance of a business over time.

5.1.1.3. Why Reverse Budgeting Is Better?
Zero-based budgeting may not be the best fit for personal finance management because it requires assigning every dollar and reaching a "zero" balance, which may not provide the desired flexibility for individuals to prioritize their financial goals. Personal finance involves dynamic income and expenses that can vary month to month, making it difficult to adhere strictly to predetermined allocations.
A static budget, with fixed amounts, may not adapt to changing financial needs and may hinder the ability to adjust and optimize financial plans. Instead, personal finance management often benefits from budgeting approaches that allow for flexibility, adaptability, and customization based on individual circumstances and goals.
the reverse budgeting approach provides a balanced approach to personal finance management by emphasizing savings while allowing flexibility for discretionary spending and adjusting to changing circumstances.

5.2. Reverse Budgeting algorithm:

5.2.1. What Makes Reverse Budgeting Different is:

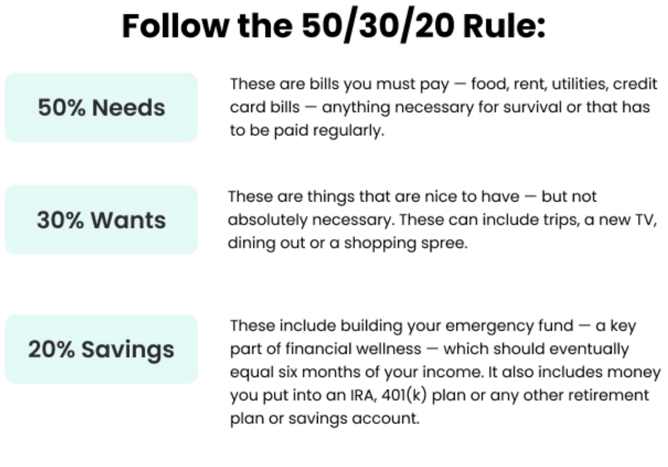
Reverse budgeting is a unique approach to personal finance management that flips the traditional budgeting model on its head. Rather than focusing solely on tracking expenses, reverse budgeting places a priority on saving and investing. The idea behind this method is to allocate a portion of your income towards savings and debt repayment before allocating the remaining funds to expenses. This approach ensures that you are "paying yourself first" and working towards your future financial goals, such as building an emergency fund, making big purchases or planning for retirement.

By prioritizing savings and investments, reverse budgeting can help you establish a balance between meeting your current bills and preparing for your future. It can also help you identify your financial goals, set priorities, manage high-priority expenses and adjust your spending habits. For those who find traditional budgeting too rigid or difficult to follow, reverse budgeting provides a more flexible and customizable alternative.

5.2.2. 4 Steps to Creating a Reverse Budget Plan:

Step 1: Assess and Identify Your Expenses
Before creating a budget, it's important to determine how much money you have available to spend each month. You can start by listing all of your necessary expenses such as rent, mortgage, transportation, and other bills you must pay. This will give you an understanding of your total monthly expenses and how much you can allocate towards other areas of your budget.

Step 2: Set a Savings Goal
Once you have determined your necessary expenses, the next step is to calculate the amount of money you should allocate for meeting your financial objectives. This will involve determining the amount of money that you need to save or invest each month.



Follow the 50/30/20 Rule:

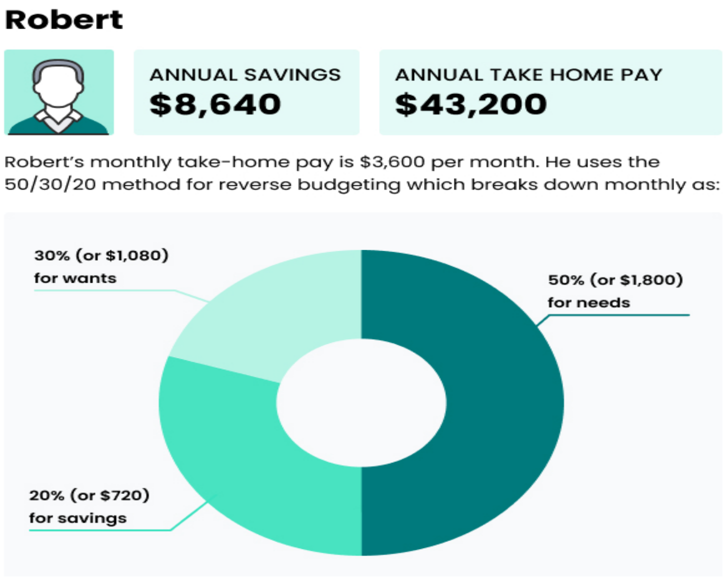
50% Needs	These are bills you must pay — food, rent, utilities, credit card bills — anything necessary for survival or that has to be paid regularly.
30% Wants	These are things that are nice to have — but not absolutely necessary. These can include trips, a new TV, dining out or a shopping spree.
20% Savings	These include building your emergency fund — a key part of financial wellness — which should eventually equal six months of your income. It also includes money you put into an IRA, 401(k) plan or any other retirement plan or savings account.

Step 3: Pay Yourself First
Make it a habit to prioritize your financial goals by allocating a portion of your paycheck into savings as soon as you receive it. By automating your savings and bill payments, you can avoid the hassle of constantly managing your budget on a regular basis. You can easily set up automatic withdrawals, paycheck deductions and payments to ensure that you are meeting your savings goals and paying your bills on time. This way, you can use the remainder of your paycheck to cover other expenses without worrying about overspending.

Step 4: Increase Your Savings and Adjust as Needed
It is important to maintain consistency in the amount you set aside for savings each paycheck. If you find it difficult to stick to the amount, consider cutting down on non-essential expenses before decreasing your savings amount. Although savings should be the last thing you cut, if necessary, focus on cutting back on saving for one goal at a time, such as retirement or a down payment on a house. As time goes on, try to increase the amount you save. You can do this by increasing the amount you contribute to your retirement plan or automatic savings deposits. Gradually increasing your savings can help you reach your long-term financial goals without causing financial strain.

5.2.3. The purpose of reverse budgeting:

Reverse budgeting focuses on prioritizing savings, and one of its main benefits is the ability to automate the process. When you set up automatic deductions for your savings from your paycheck or bank account, you reduce the risk of spending that money on other things. This helps you stay on track towards your financial goals by ensuring that money is consistently going towards your savings.



6. Technology

6.1. Back-End-NODEJS:

Node.js is an open-source, cross-platform JavaScript runtime environment built on Chrome's V8 JavaScript engine. It allows developers to build server-side applications in JavaScript. Node.js uses an event-driven, non-blocking I/O model, which makes it lightweight and efficient for building scalable network applications.

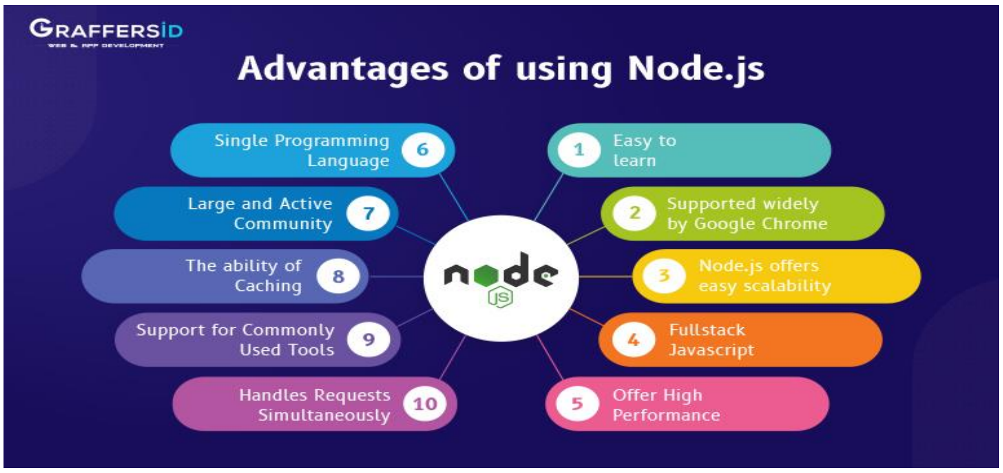
Node.js has a large and active community of developers, which has contributed to its extensive collection of open-source packages available through the Node Package Manager (NPM). These packages provide various functionality and can be easily integrated into Node.js applications.

Node.js is commonly used for building web servers, command-line tools, real-time chat applications, API servers, and microservices. It is also popular for building JavaScript-based build tools, like Gulp and Grunt.

Some of the advantages of Node.js include its scalability, performance, and the ability to use JavaScript on both the client-side and server-side of an application, reducing the need for different programming languages. Additionally, Node.js allows for fast and easy prototyping and development of applications due to its quick iteration times and the availability of numerous open-source packages.

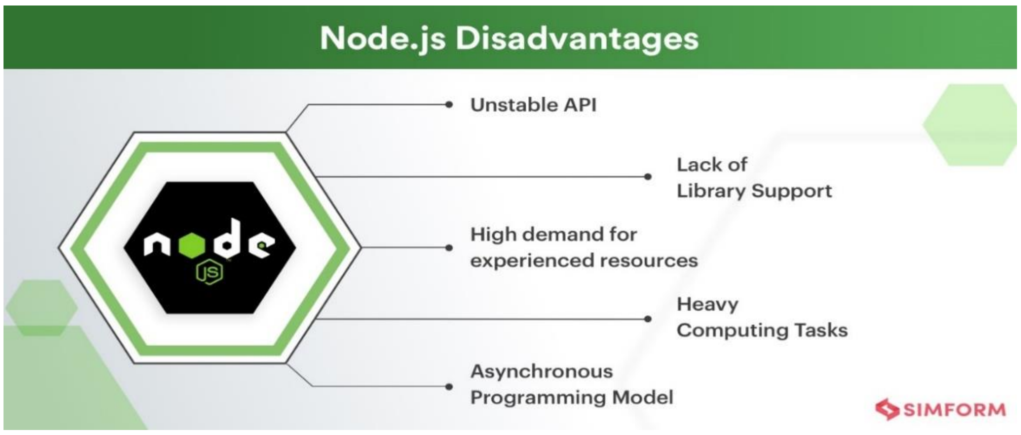
6.1.1 NODEJS Advantages:

- Fast and efficient
- Scalable
- Large and active community
- Cross-platform compatibility
- Easy to learn and use
- Support for real-time web applications
- Excellent for microservices architecture
- Supports JSON
- Great for streaming data
- Extensive library of modules and packages.


GRAFFERSID
WEB & APP DEVELOPMENT
Advantages of using Node.js
1 Easy to learn
2 Supported widely by Google Chrome
3 Node.js offers easy scalability
4 Fullstack Javascript
5 Offer High Performance
6 Single Programming Language
7 Large and Active Community
8 The ability of Caching
9 Support for Commonly Used Tools
10 Handles Requests Simultaneously
node JS

6.1.2 NODEJS Disadvantages:

- Single-threaded nature can limit performance for CPU-intensive tasks
- Limited support for multi-threading
- Relatively new technology compared to other server-side platforms
- Callback-based programming can lead to callback hell and less readable code
- Not ideal for heavy computation or mathematical operations



6.2. BACK-END MONGODB:

MongoDB is a popular open-source NoSQL database that stores data in a document-oriented format, instead of the traditional row-and-column structure used in relational databases. It is designed to be highly scalable, flexible and fast, allowing developers to build and deploy applications faster than with traditional databases.

6.2.1. MONGODB Advantages:

- Flexible and scalable document-based data model
- High performance and availability
- Automatic sharing for horizontal scaling
- Support for ACID transactions in multi-document operations
- Easy to use and learn, with a large and active community
- Open source with a free version available
- Built-in support for indexing, search, and aggregation operations

6.2.2. MONGODB Disadvantages:

- No support for joins or transactions
- Not suitable for complex, relational data models
- Limited analytics and reporting capabilities
- Limited community support and resources
- Requires more memory compared to other databases
- Not as mature as some other databases

6.3. Front-End-React-Native:

React Native is a JavaScript framework for building mobile applications for iOS and Android platforms. It allows developers to use a single codebase to create mobile applications for multiple platforms, which can save time and effort. React Native uses a combination of JavaScript and native platform components, providing a performance that is close to that of a fully native application. It also has a large and active developer community and provides access to a wide range of third-party libraries and tools.

6.3.1. React Native Advantages:

- Cross-platform development for iOS and Android
- Large developer community and support
- Live reload feature for faster development
- Reusable components for efficient coding
- High performance due to native rendering
- Easy integration with existing native code

6.3.2. React Native Disadvantages:

- Limited access to native APIs
- Debugging can be challenging
- Some features may not be supported by React Native
- Learning curve for developers not familiar with React
- May not be suitable for large and complex apps
- Possible performance issues with heavy computations.

6.4. Expo:

Expo is a set of tools and services built around React Native and, while it has many features, the most relevant feature for us right now is that it can get you writing a React Native app within minutes. You will only need a recent version of Node.js and a phone or emulator

6.4.1 Expo Go:

Expo Go is a free client for testing React Native apps on Android and iOS. It eliminates the need for local builds and allows you to quickly start testing your apps. Just use the Expo Go app on your device, and you're good to go. It's the fastest and easiest way to get started with testing React Native apps.

7. Features and Challenges

The project incorporates several unique features and engineering challenges that contribute to its innovative nature. These aspects include:

Algorithm Implementation for Personalized Recommendations:

- Challenge: Designing and implementing algorithms for personalized budgeting, saving, and investment recommendations based on user data was a complex task. It required careful analysis of user financial data and the development of algorithms that could provide tailored recommendations while considering factors such as income, expenses, financial goals, and risk tolerance.
- Solution: We broke down the problem into smaller, manageable components and leveraged existing financial algorithms and models as a foundation. We conducted thorough research on budgeting and investment strategies and developed custom algorithms that could analyze user data and generate personalized recommendations. Additionally, we iteratively tested and refined the algorithms to ensure accuracy and effectiveness.

Integration of Backend and Frontend Components:

- Challenge: Integrating the backend Node.js server with the frontend React Native application posed challenges in terms of data exchange, API integration, and ensuring seamless communication between the two components.
- Solution: We followed best practices for API design and development, ensuring that the backend API endpoints were well-documented, RESTful, and followed a consistent naming convention. We also used libraries like Axios for handling HTTP requests and responses in the frontend, making it easy to interact with the backend API. Regular testing and debugging helped identify and resolve any issues with data exchange and communication between the backend and frontend components.

Data Structure and Database Schema Design:

- Challenge: Designing an efficient data structure and database schema to store and manage user financial data, transactions, and recommendations while ensuring scalability and performance.
- Solution: We carefully analyzed the requirements of the system and the types of data that needed to be stored. We opted for MongoDB as the database management system due to its flexibility and scalability, allowing us to store complex, hierarchical data structures efficiently. We designed a schema that could accommodate various types of financial data, such as income, expenses, goals, and investment portfolios, while ensuring optimal performance for data retrieval and manipulation.

User Interface Design and User Experience:

- Challenge: Designing an intuitive and user-friendly interface for the mobile application that effectively presented complex financial data and recommendations to users while maintaining a visually appealing and cohesive design.
- Solution: We utilized Figma for designing the user interface, allowing us to create interactive prototypes and iterate on design concepts collaboratively. We conducted user testing and feedback sessions to gather input from potential users and refine the interface based on their preferences and needs. Additionally, we followed design principles such as consistency, hierarchy, and accessibility to ensure a seamless and engaging user experience.

By addressing these challenges through careful planning, research, collaboration, and iterative development, we were able to overcome obstacles and successfully build SpendSmart, delivering a strong and user-friendly financial management application that met the needs of our target audience.

8. Results and Conclusions:

After months of rigorous development and testing, we have successfully achieved the goals set for the SpendSmart project

8.1. Goal Achievement:

- **Personalized Financial Management:** SpendSmart successfully delivers on its promise of personalized financial management. Through the integration of smart budgeting algorithms and investment allocation strategies, users can make informed decisions tailored to their unique financial situations and aspirations.
- **Smart Budgeting Algorithm:** Implemented an advanced budgeting algorithm that analyzes user income, expenses, and financial goals to provide personalized budgeting recommendations, helping users allocate their funds wisely and achieve their financial objectives.
- **Goal Tracking and Visualization:** Integrated goal tracking features into SpendSmart, allowing users to set, track, and visualize progress towards their financial goals. This feature empowers users to stay motivated and focused on achieving their objectives.
- **Investment Allocation and Growth:** enabling users to invest their money for long-term growth and financial security. SpendSmart provides personalized investment recommendations tailored to each user's risk tolerance and financial goals.
- **Loan Management and Assessment:** Implemented loan management functionalities within SpendSmart, allowing users to explore loan options and assess the financial impact of borrowing money on their overall financial plan. SpendSmart provides users with a clear picture of loan details, repayment schedules, and total repayment amounts to help them make informed decisions.

8.2. Dealing with Challenges:

- **Algorithm Complexity:** We addressed the challenge of implementing complex algorithms by breaking down the problem into smaller, manageable components and leveraging existing models and research. Thorough testing and refinement ensured that the algorithms provided accurate and effective recommendations.
- **Integration Complexity:** Integrating backend and frontend was made easier by planning well, documenting clearly, and testing regularly. We followed best practices for designing APIs, making it simple to fix any issues.
- **Design Considerations:** Throughout the project, we focused on making the app user-friendly, fast, and able to handle lots of users. We did this by researching, getting feedback from users, and improving our designs and implementations as needed.

8.3. Considerations in Decision-Making:

- **User-Centric Approach:** We made decisions based on what our users needed and liked. We did lots of research, got feedback from users, and made changes to our designs and features based on what they told us.
- **Technical Feasibility:** We carefully looked at how hard it would be to build each feature, considering things like how complex it was and how much it would affect the app's performance. We chose solutions that could grow with the app and didn't make it slow or hard to use.
- **Continuous Improvement:** We always tried to make the app better by regularly looking at how we were doing, finding things to improve, and listening to feedback from everyone involved. This helped us keep making the app better over time.

9. Learning Lessons:

Reflecting on the project, we've learned valuable lessons that will inform our future work.

9.1. Project Development Lessons:

- Teamwork Matters: We learned that teamwork is crucial for success. When everyone in the team communicates openly and works together nicely, things tend to go smoothly. Sharing ideas, helping each other out, and being respectful creates a positive environment where we can achieve our goals effectively.
- Stay Flexible: Flexibility is key in any project. We discovered that being open to changes and ready to adapt our plans when needed is important for overcoming unexpected challenges. By staying flexible, we can adjust our approach to meet new requirements or handle unforeseen obstacles without getting stuck or stressed out.
- Think Like Users: Understanding the needs and preferences of our users is essential for creating a product that they will love. We realized the importance of putting ourselves in the shoes of our users and thinking about what they would want. By listening to their feedback and considering their perspectives, we can design a better user experience that meets their expectations.

- **Keep Learning:** The world of technology is constantly evolving, so it's crucial to keep learning and expanding our skills. We recognized that staying up-to-date with the latest tools, technologies, and best practices is essential for staying competitive and delivering high-quality solutions. By investing in continuous learning, we can stay ahead of the curve and continue to grow as professionals.
- **Test Early, Test Often**: Testing is a critical part of the development process. We learned that finding and fixing problems early is much easier and less costly than waiting until later. By conducting thorough testing at every stage of development, from initial prototypes to final products, we can ensure that our solutions are reliable, bug-free, and meet the needs of our users.
- **Plan Better:** Planning is key to success in any project. We realized the importance of having a clear plan and timeline in place to guide our efforts and keep us on track. By setting clear goals, establishing timelines, and regularly monitoring progress, we can ensure that our projects are completed successfully and on time.

9.2. Enhancing User Experience & Features:

- **Adding More Features:** We'll brainstorm and add more cool things users can do in the app, like setting reminders for bills or tracking their spending in different categories. By continuously adding new features, we can make the app even more useful and valuable to our users.
- **Modern UI:** Making the app look cool and easy to use is super important. We'll keep updating the design to stay trendy and make sure it's easy for everyone to use. By incorporating modern design principles and staying up-to-date with the latest UI trends, we can create an engaging and visually appealing user **experience.**
- **Investment Ideas:** We'll come up with more ideas to help users make smart investment choices, like suggesting diversified portfolios or offering educational resources on investing. By providing users with valuable information and tools, we can empower them to make informed decisions and achieve their financial goals.

10. Benchmark Assessment:

As we take a moment to look back on the goals we set for our project, we find both achievements and areas where we can do better:

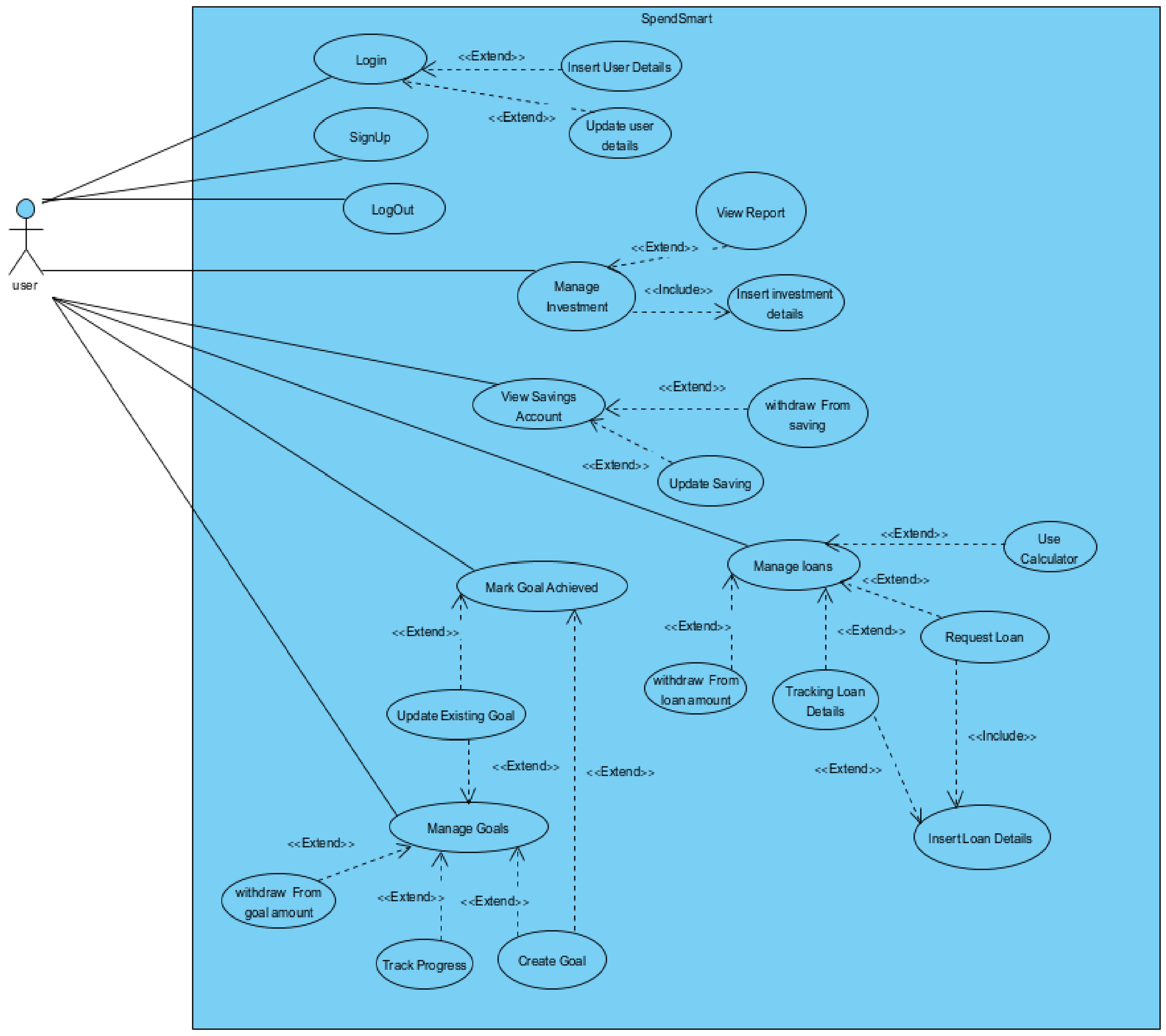
- Teamwork and Communication: We tried to work together and talk openly. Sometimes, though, we could have talked more.
- Flexibility and Adaptability: We tried to be flexible when things changed, but we could have been quicker to adjust our plans sometimes.
- User-Centric Design: We wanted to make our app great for users, but we might have missed some things they wanted. We'll keep trying to make it better for them.
- Continuous Learning and Improvement: We tried to learn new stuff, but there's always more to learn. We'll keep trying to get better.
- Thorough Testing and Quality Assurance: We tested our app, but maybe not as much as we should have. We'll make sure to test everything better next time.
- Effective Project Planning and Management: We had a plan, but things didn't always go exactly as we hoped. We'll work on planning better and dealing with unexpected stuff more smoothly.

In conclusion, while we made progress in meeting our project benchmarks, there were areas where we fell short or encountered challenges. By acknowledging these areas for improvement and committing to ongoing growth and development, we aim to strengthen our project execution and deliver even better results in future endeavors.

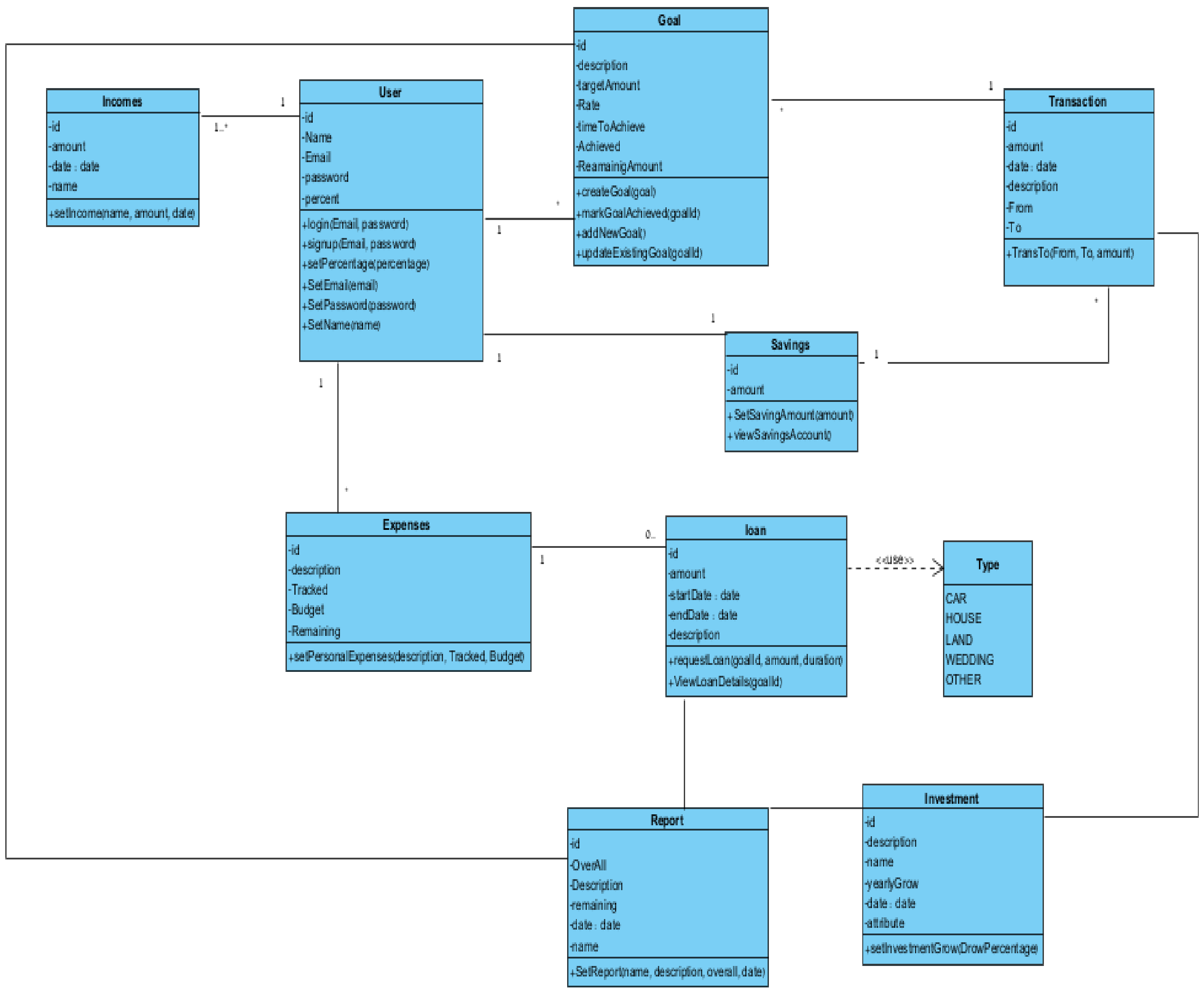
11. Engineer Process

11.1. Products:

11.1.1. Use Case Diagram

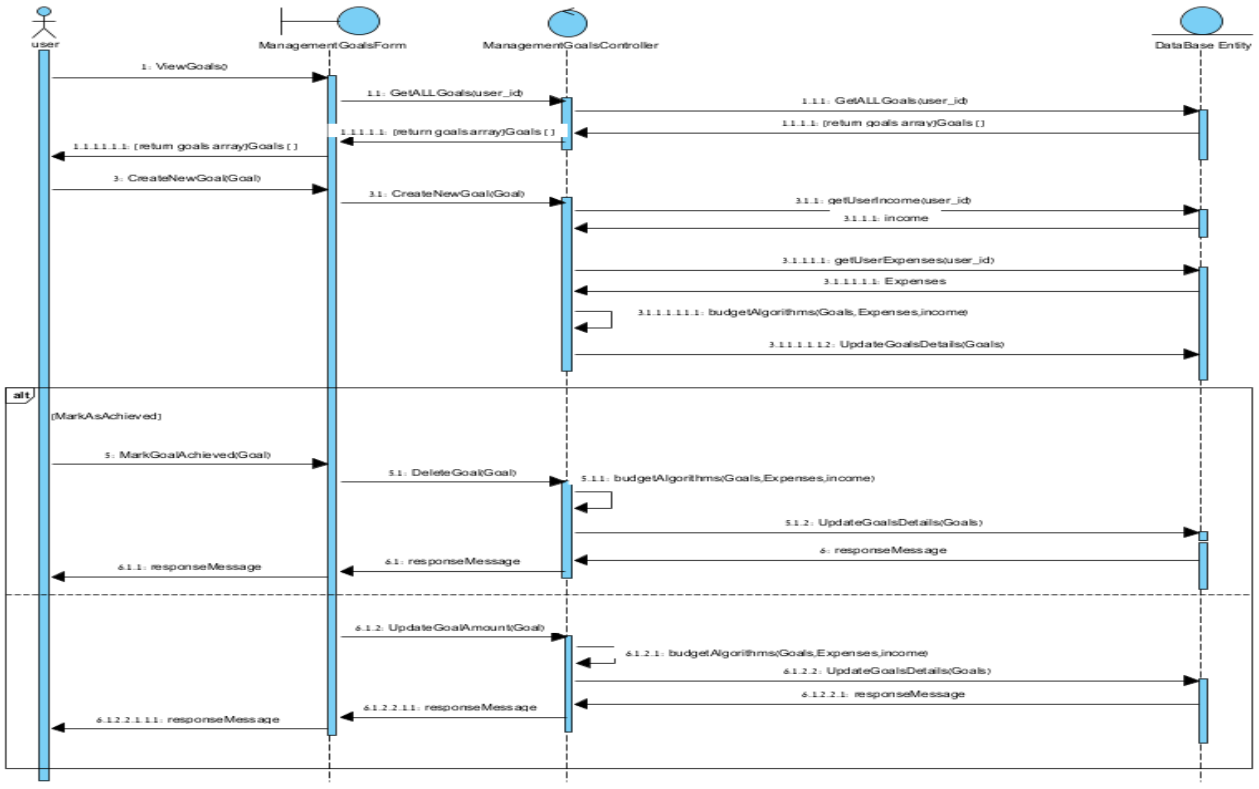


11.1.2. Class Diagram

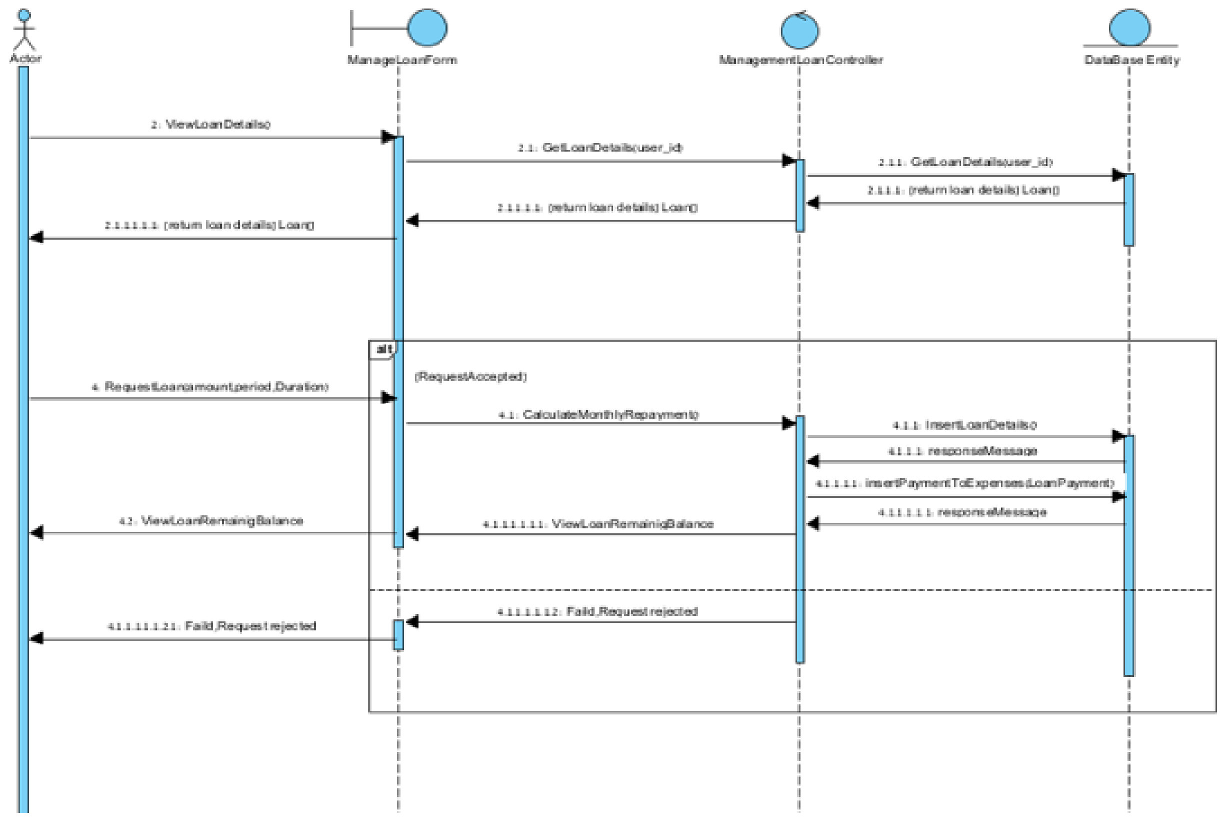


11.1.2. Sequence Diagram

Manage Goal Process:

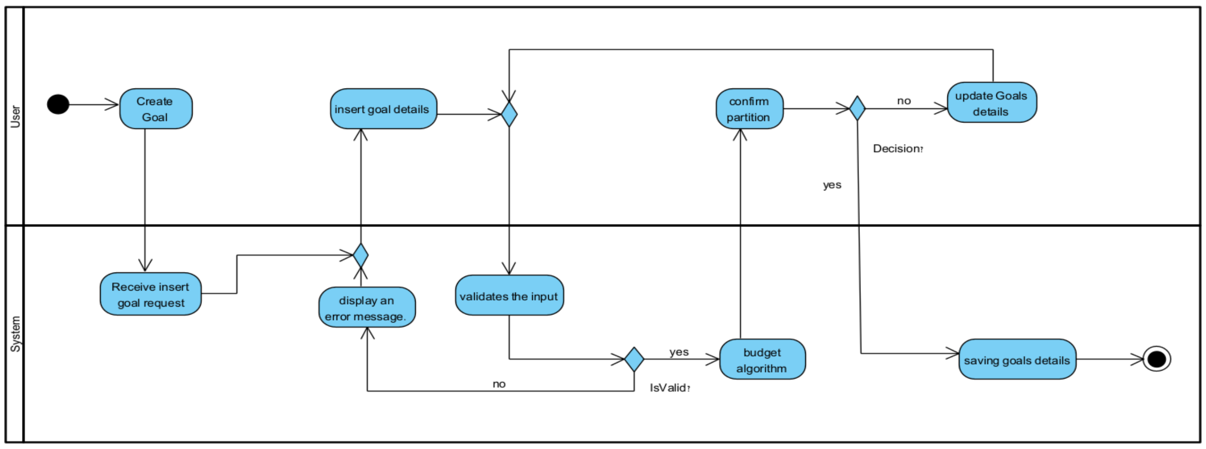

user
ManagementGoalsForm
ManagementGoalsController
DataBase Entity
1: ViewGoals()
1.1: GetALLGoals(user_id)
1.1.1: GetALLGoals(user_id)
1.1.1.1: [return goals array]Goals [ ]
1.1.1.1.1: [return goals array]Goals [ ]
1.1.1.1.1.1: [return goals array]Goals [ ]
3: CreateNewGoal(Goal)
3.1: CreateNewGoal(Goal)
3.1.1: getUserIncome(user_id)
3.1.1.1: income
3.1.1.1.1: getUserExpenses(user_id)
3.1.1.1.1.1: Expenses
3.1.1.1.1.1.1: budgetAlgorithms(Goals,Expenses,income)
3.1.1.1.1.1.2: UpdateGoalsDetails(Goals)
alt
[MarkAsAchieved]
5: MarkGoalAchieved(Goal)
5.1: DeleteGoal(Goal)
5.1.1: budgetAlgorithms(Goals,Expenses,income)
5.1.2: UpdateGoalsDetails(Goals)
6: responseMessage
6.1: responseMessage
6.1.1: responseMessage
6.1.2: UpdateGoalAmount(Goal)
6.1.2.1: budgetAlgorithms(Goals,Expenses,income)
6.1.2.2: UpdateGoalsDetails(Goals)
6.1.2.2.1: responseMessage
6.1.2.2.1.1: responseMessage
6.1.2.2.1.1.1: responseMessage

Loan Request Process:

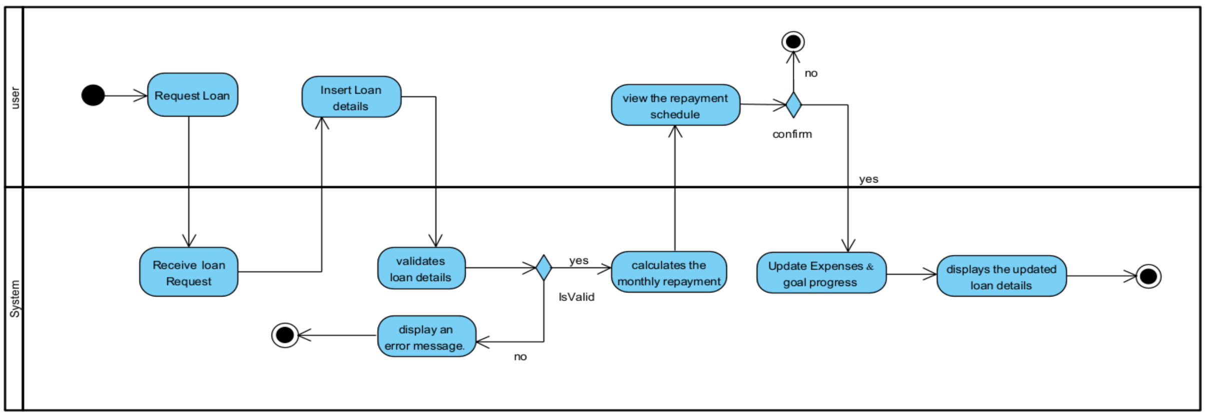

Actor
ManageLoanForm
ManagementLoanController
DataBase Entity
2: ViewLoanDetails()
2.1: GetLoanDetails(user_id)
2.1.1: GetLoanDetails(user_id)
2.1.1.1: [return loan details] Loan()
2.1.1.1.1: [return loan details] Loan()
2.1.1.1.1.1: [return loan details] Loan()
alt
[RequestAccepted]
4: RequestLoan(amount,period,Duration)
4.1: CalculateMonthlyRepayment()
4.1.1: InsertLoanDetails()
4.1.1.1: responseMessage
4.1.1.1.1: insertPaymentToExpenses(LoanPayment)
4.1.1.1.1.1: responseMessage
4.1.1.1.1.1.1: ViewLoanRemainigBalance
4.2: ViewLoanRemainigBalance
4.1.1.1.1.1.2: Faild,Request rejected
4.1.1.1.1.1.2.1: Faild,Request rejected

11.1.3. Activity Diagram:

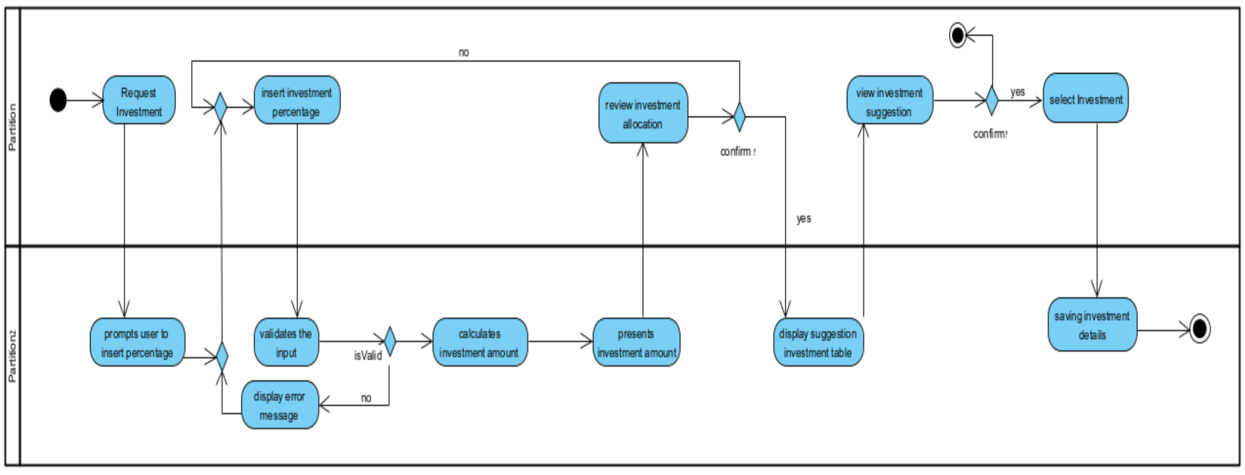
Manage Goal Process



Loan Request Process

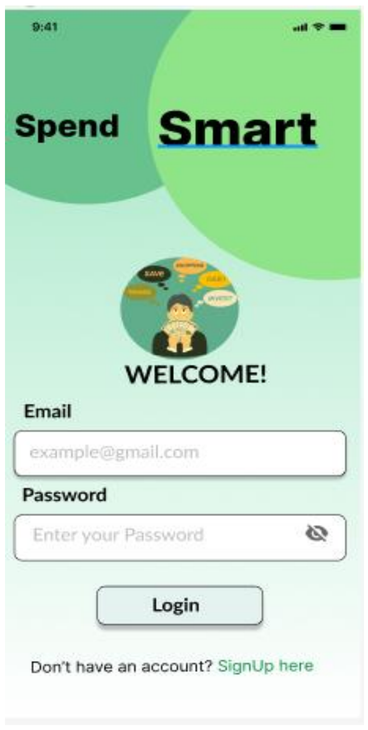

user
System
Request Loan
Receive loan Request
Insert Loan details
validates loan details
yes
IsValid
no
display an error message.
calculates the monthly repayment
view the repayment schedule
no
confirm
yes
Update Expenses & goal progress
displays the updated loan details

Investment Process

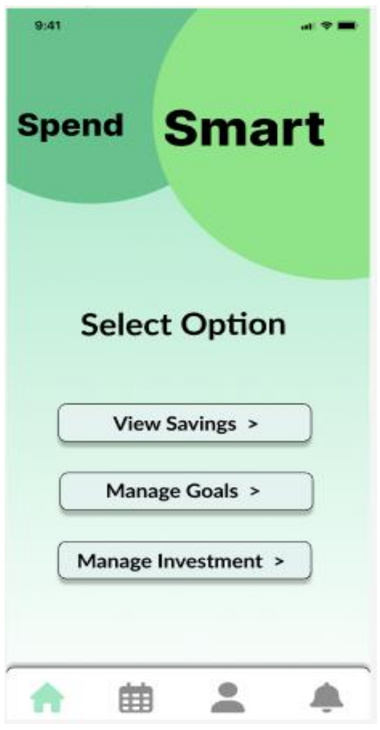

Partition
Partition2
Request Investment
no
insert investment percentage
prompts user to insert percentage
validates the input
isValid
no
display error message
calculates investment amount
presents investment amount
review investment allocation
confirm↑
yes
display suggestion investment table
view investment suggestion
yes
confirm↑
select Investment
saving investment details

11.1.4. GUI

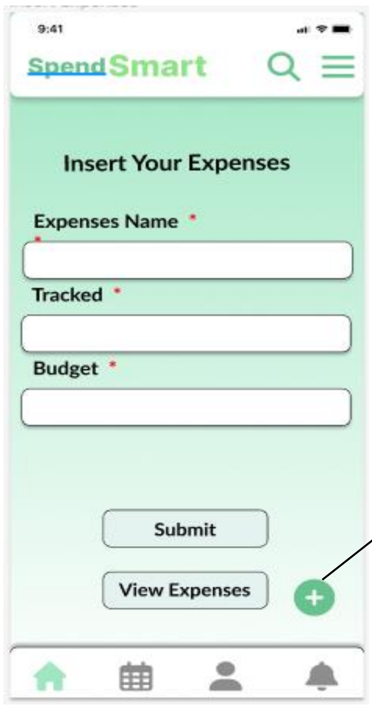
Login Page
User's log into the application based on the username and password:



Home Page
The user can either select to view and manage their savings, manage their goals, or manage their investments on the homepage.



Expenses and Incomes Pages
Users, regardless of whether they are new or existing, can input their expenses and incomes in these pages. They can then save their entries by clicking "Submit".


9:41
SpendSmart

Insert Your Expenses

Expenses Name *

Tracked *

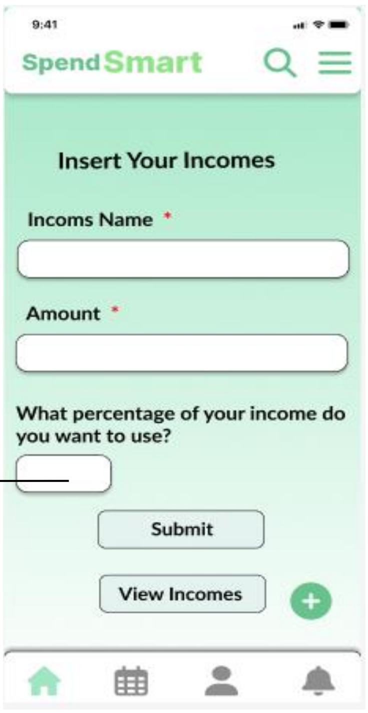
Budget *

Submit

View Expenses

This button to Add Expenses or income.

This button to view Expenses or income details


9:41
SpendSmart

Insert Your Incomes

Incoms Name *

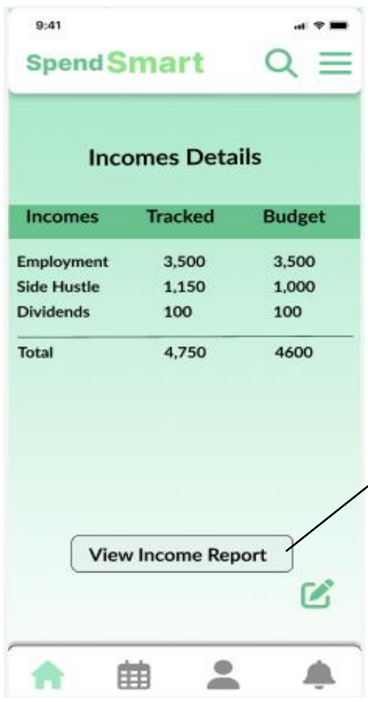
Amount *

What percentage of your income do you want to use?

Submit

View Incomes

Incomes Details Pages
there is a button available to view detailed information about their Incomes, enabling them to track and analyze their financial transactions easily in two ways: List or Report with Graph.

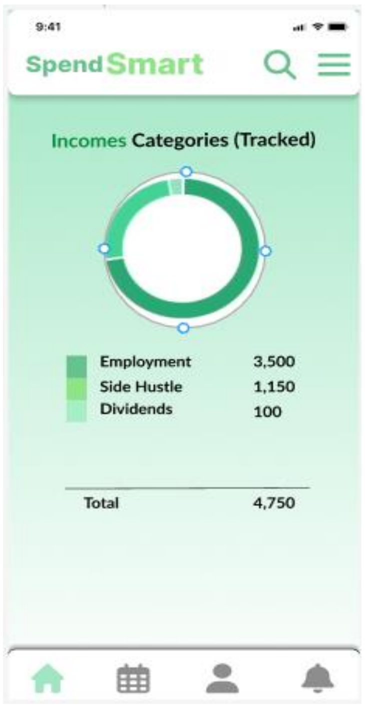

9:41
SpendSmart

Incomes Details

Incomes	Tracked	Budget
Employment	3,500	3,500
Side Hustle	1,150	1,000
Dividends	100	100
Total	4,750	4600

View Income Report

This button to Edit the list of his Incomes

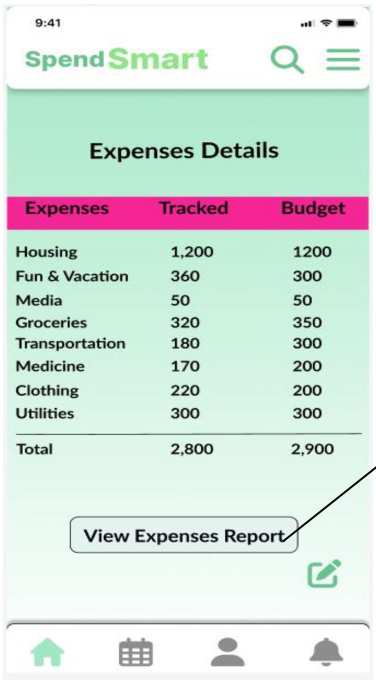

9:41
SpendSmart

Incomes Categories (Tracked)

🟩	Employment	3,500
🟩	Side Hustle	1,150
🟩	Dividends	100
	Total	4,750

Expenses Details Pages

there is a button available to view detailed information about their Expenses, enabling them to track and analyze their financial transactions easily in two ways: List or Report with Graph.



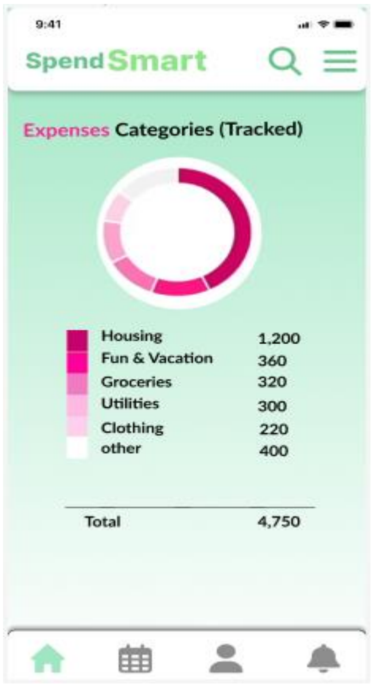
9:41

SpendSmart

Expenses Details

Expenses	Tracked	Budget
Housing	1,200	1200
Fun & Vacation	360	300
Media	50	50
Groceries	320	350
Transportation	180	300
Medicine	170	200
Clothing	220	200
Utilities	300	300
Total	2,800	2,900

This button to Edit the list of his Expenses



9:41

SpendSmart

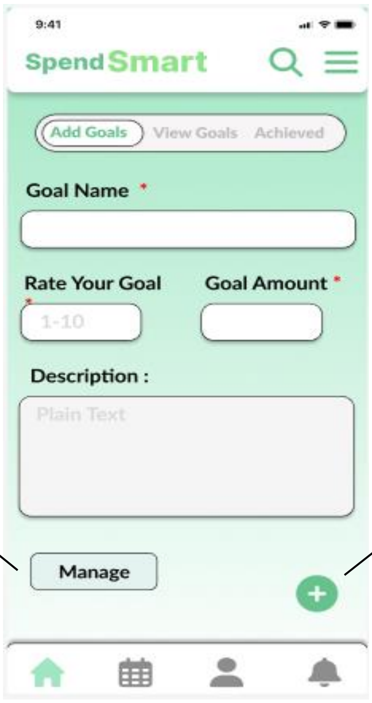
Expenses Categories (Tracked)

Housing	1,200
Fun & Vacation	360
Groceries	320
Utilities	300
Clothing	220
other	400
Total	4,750

Insert Goals Page

In this page, the user can input their goals along with all the relevant details they want to achieve. Once entered, the application saves the goals. From there, the application takes charge of managing the goals using a budgeting algorithm.

Clicking the button triggers the budgeting algorithm to manage the user's goals, considering both the newly inserted goals and the existing ones.



9:41

SpendSmart

Goal Name *

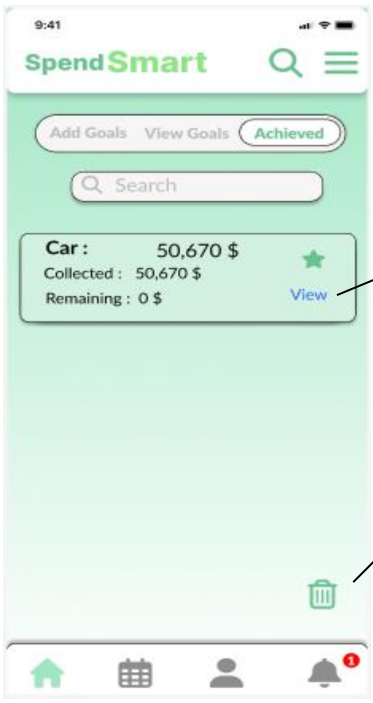
Rate Your Goal *: 1-10

Goal Amount *

Description : Plain Text

This button to Add new Goal

Goals Pages
In these pages, you can view the goals you have set and the goals you have achieved. The top section features three buttons that allow you to navigate between different pages.

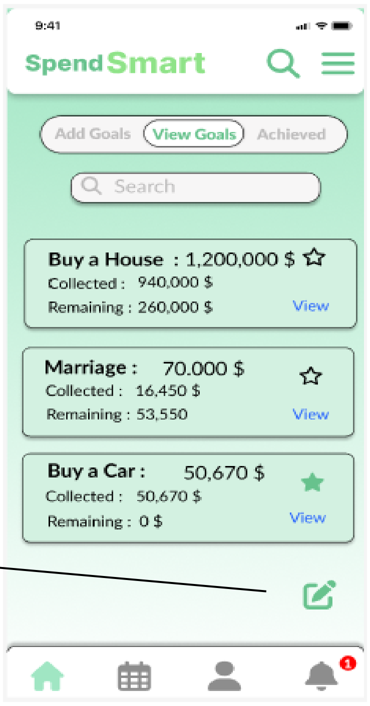


Car : 50,670 $
Collected : 50,670 $
Remaining : 0 $
View

These View buttons enables you to view comprehensive goal details

This button to Delete Goal

This button to Edit your Goal

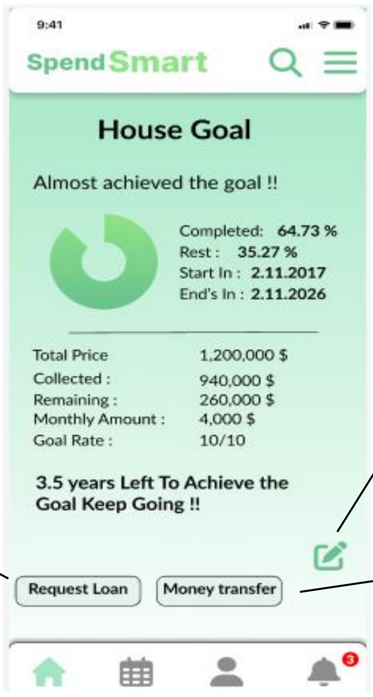


Buy a House : 1,200,000 $
Collected : 940,000 $
Remaining : 260,000 $
View

Marriage : 70.000 $
Collected : 16,450 $
Remaining : 53,550
View

Buy a Car : 50,670 $
Collected : 50,670 $
Remaining : 0 $
View

Goal Details Pages
In this page, you can access and view all the details about a specific goal. It includes a graph that visually represents your progress towards the goal, as well as other relevant information.



House Goal

Almost achieved the goal !!

Completed: 64.73 %
Rest : 35.27 %
Start In : 2.11.2017
End's In : 2.11.2026

Total Price	1,200,000 $
Collected :	940,000 $
Remaining :	260,000 $
Monthly Amount :	4,000 $
Goal Rate :	10/10

3.5 years Left To Achieve the Goal Keep Going !!

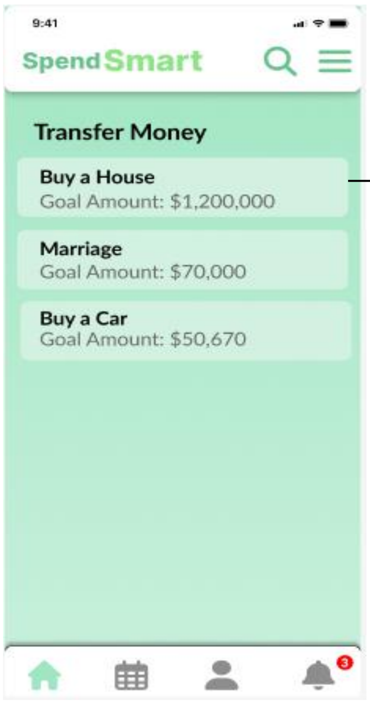
Request Loan
Money transfer

Allows the user to Edit the Goal Details.

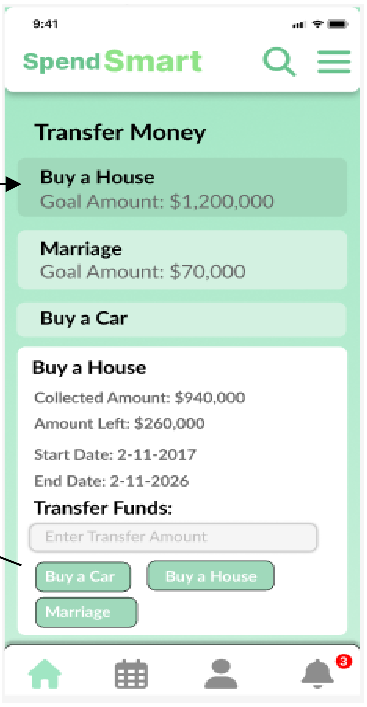
Allows the user to navigate to the loan page.

Allows the user to navigate to the Transfer Money page.

Transfer Money Pages
In this page you have all your goals, you can transfer money from one selected goal to another by specifying the amount and choosing the destination goal.

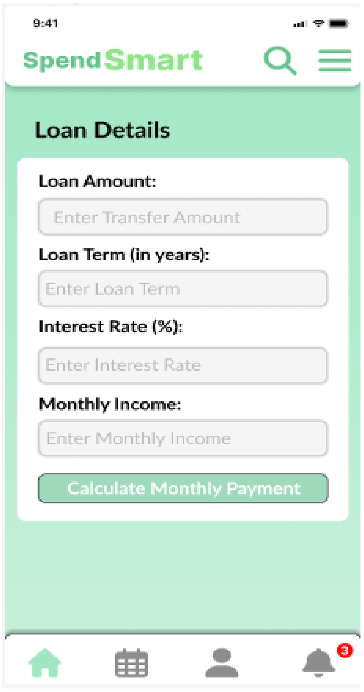

9:41
SpendSmart
Transfer Money
Buy a House
Goal Amount: $1,200,000
Marriage
Goal Amount: $70,000
Buy a Car
Goal Amount: $50,670

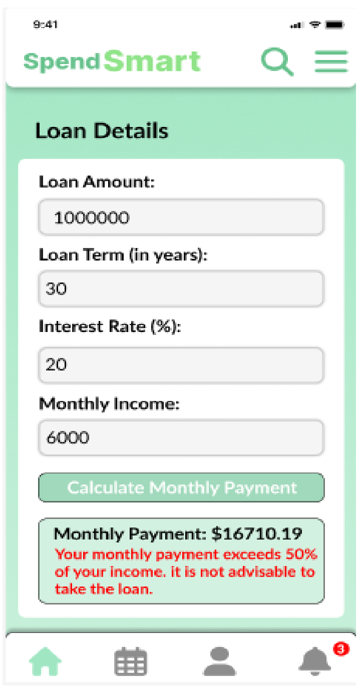
After clicking on this button, you will see the following

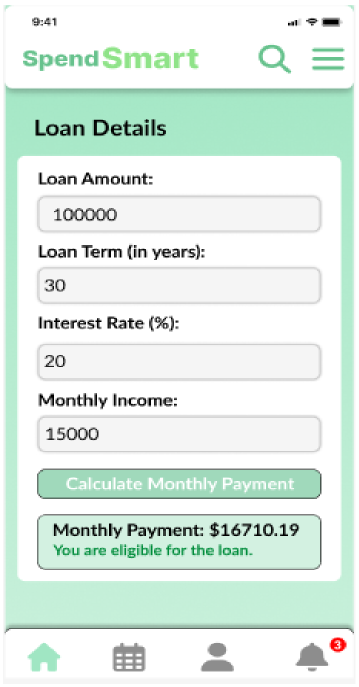

9:41
SpendSmart
Transfer Money
Buy a House
Goal Amount: $1,200,000
Marriage
Goal Amount: $70,000
Buy a Car
Buy a House
Collected Amount: $940,000
Amount Left: $260,000
Start Date: 2-11-2017
End Date: 2-11-2026
Transfer Funds:
Enter Transfer Amount
Buy a Car
Buy a House
Marriage

Here you can insert the amount the you want to transfer from the selected goal to another that you also select.

Loan Request Pages
In this page, you can request a loan by entering the loan details, including the desired amount and interest rate. The application will calculate the monthly amount you will receive and provide a message indicating loan eligibility.

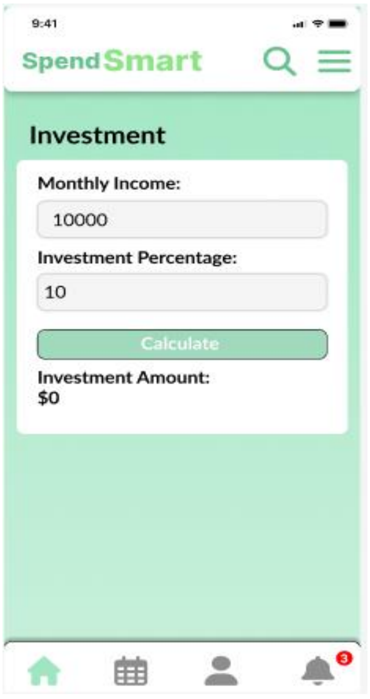

9:41
SpendSmart
Loan Details
Loan Amount:
Enter Transfer Amount
Loan Term (in years):
Enter Loan Term
Interest Rate (%):
Enter Interest Rate
Monthly Income:
Enter Monthly Income
Calculate Monthly Payment


9:41
SpendSmart
Loan Details
Loan Amount:
1000000
Loan Term (in years):
30
Interest Rate (%):
20
Monthly Income:
6000
Calculate Monthly Payment
Monthly Payment: $16710.19
Your monthly payment exceeds 50% of your income. it is not advisable to take the loan.


9:41
SpendSmart
Loan Details
Loan Amount:
100000
Loan Term (in years):
30
Interest Rate (%):
20
Monthly Income:
15000
Calculate Monthly Payment
Monthly Payment: $16710.19
You are eligible for the loan.

Investment Pages

In this page, you can invest a portion of your monthly income by specifying the income amount and desired percentage to invest. The app will calculate the corresponding investment amount for you.


9:41

SpendSmart

Investment

Monthly Income:

10000

Investment Percentage:

10

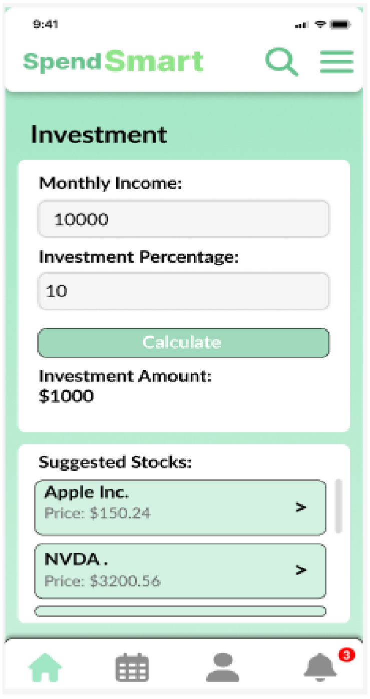
Calculate

Investment Amount:

$0

Investment Pages After Button Clicked

After clicking the calculate button, the app will show the investment amount. It will also provide suggestions for the top 5 stocks in the market, allowing the user to consider investing in them if interested.


9:41

SpendSmart

Investment

Monthly Income:

10000

Investment Percentage:

10

Calculate

Investment Amount:

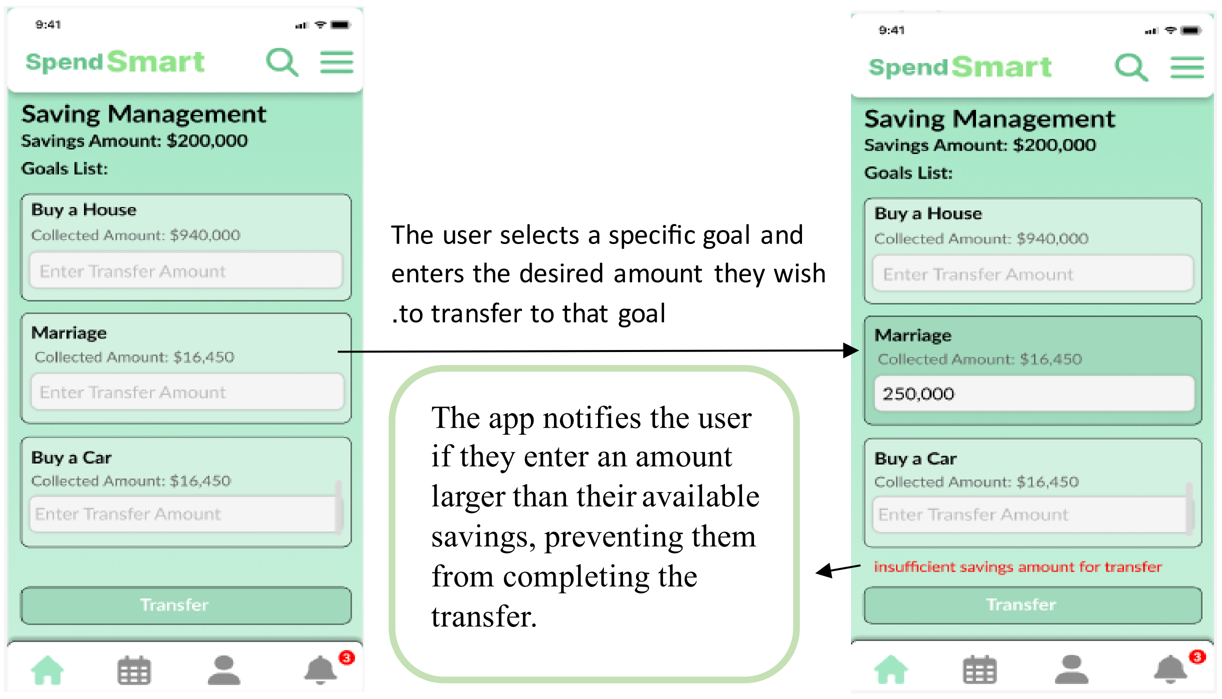
$1000

Suggested Stocks:

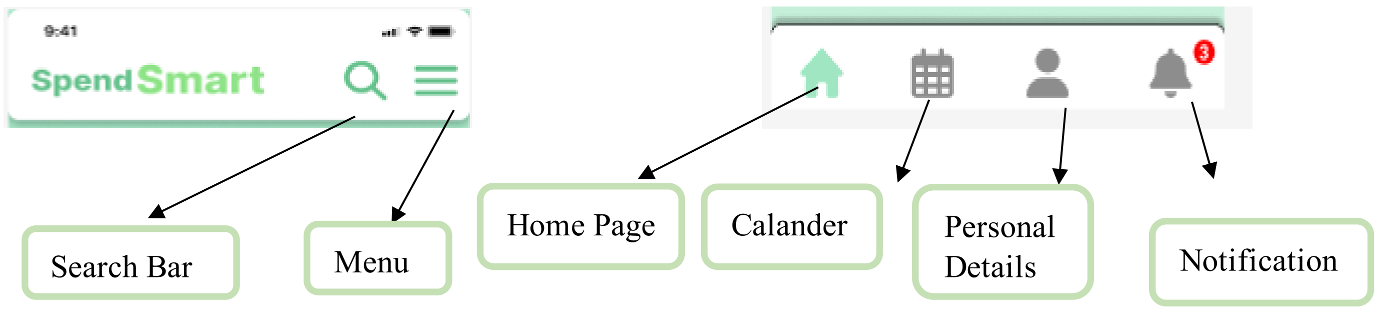
Apple Inc.
Price: $150.24 >

NVDA .
Price: $3200.56 >

Savings Management Page
In this page, the user can view their savings amount and transfer money from savings to selected goals by specifying the desired transfer amount per goal.


9:41
SpendSmart
Saving Management
Savings Amount: $200,000
Goals List:
Buy a House
Collected Amount: $940,000
Enter Transfer Amount
Marriage
Collected Amount: $16,450
Enter Transfer Amount
Buy a Car
Collected Amount: $16,450
Enter Transfer Amount
Transfer
The user selects a specific goal and enters the desired amount they wish .to transfer to that goal
The app notifies the user if they enter an amount larger than their available savings, preventing them from completing the transfer.
250,000
insufficient savings amount for transfer

Notifications and
Our application utilize notifications to keep users informed and on track with their financial management. Notifications serve as reminders for users to input necessary information or notify them of important milestones or achievements.
The search functionality in the app's top bar allows users to easily find specific pages or features they are looking for. This improves user navigation and enhances the overall user experience by providing quick access to desired information or functionality.



12. Verification

12.1. Login Tests

Test Name	Description	Expected Result	Precondition	Comment
Successful Login	This test verifies that a user can successfully log in with valid credentials.	The user should be granted access to their account dashboard.	User account exists in the system with valid login credentials.	This test ensures that the login functionality is working correctly and allows users to access their accounts securely.
Failed Login - Incorrect Password	This test checks if the application rejects login attempts with an incorrect password.	The application should display an error message indicating that the password is incorrect.	User account exists in the system with a valid username but an incorrect password.	This test ensures that the application properly handles failed login attempts and provides appropriate feedback to the user.
Failed Login - Invalid Username	This test verifies that the application rejects login attempts with an invalid username.	The application should display an error message indicating that the username is not recognized.	User attempts to log in with a username that does not exist in the system.	This test ensures that the application handles invalid usernames correctly and prompts the user to enter a valid username.
Failed Login - Missing Username	This test checks if the application handles login attempts without entering a username.	The application should display an error message indicating that the username is required.	User attempts to log in without entering a username.	This test ensures that the application enforces the requirement for a username and provides appropriate validation.
Failed Login - Missing Password	This test verifies that the application requires the user to enter a password for login.	The application should display an error message indicating that the password is required.	User attempts to log in without entering a password.	This test ensures that the application enforces the requirement for a password and prompts the user to enter it.

12.2. Goal Management Tests

Test Name	Description	Expected Result	Precondition	Comment
Create Goal	This test verifies that a user can successfully create a new goal. Expected Result: The goal should be added to the user's goal list.	The goal should be added to the user's goal list.	User is logged in and on the goal management page.	This test ensures that the goal creation functionality is working correctly and allows users to define their financial objectives.
Update Goal Progress	This test checks if the application accurately tracks and updates the progress of a goal.	The goal progress should be updated based on the user's monthly contributions and the assigned rate.	User has a goal created and has made contributions towards it.	This test ensures that the application correctly calculates the progress towards each goal based on the allocated amount and the monthly contributions.
Delete Goal	This test verifies that the user can delete a goal from their goal list. Expected Result: The goal should be removed from the user's goal list.	The goal should be removed from the user's goal list.	User has a goal created and is on the goal management page.	This test ensures that the application allows users to remove goals that they no longer wish to pursue.
Add New Goal	This test checks if the user can add a new goal after achieving a previous goal.	The new goal should be added to the user's goal list.	User has achieved a goal and is on the goal management page.	This test ensures that the application enables users to set new financial objectives once they have accomplished previous goals.
Set Goal Amount	This test verifies that the user can specify the target amount for a goal.	The goal amount should be saved and associated with the goal.	User has a goal created and is on the goal management page.	This test ensures that the application correctly captures and stores the desired amount for each goal, representing the target to be achieved.

Set Goal Rate	This test checks if the user can set a rate for a specific goal.	The goal rate should be saved and associated with the goal.	User has a goal created and is on the goal management page.	This test ensures that the application correctly captures and stores the rate for each goal, which determines the contribution towards the goal each month.
Update Goal Allocation	This test verifies that the budgeting algorithm updates the allocation amount for each goal based on the user's income, rate, and desired goal amount.	The allocation amount for each goal should be calculated accurately.	User has set goals with rates, desired amounts, and income entered.	This test ensures that the budgeting algorithm considers the user's income, rates, and desired goal amounts to determine the monthly allocation for each goal.
Check Monthly Allocation Total	This test checks if the total allocation amount for all goals matches the user's income.	The total allocation amount should equal the user's income.	User has set goals with rates, desired amounts, and income entered.	This test ensures that the budgeting algorithm correctly calculates the total allocation amount and ensures that it does not exceed the user's income.
Validate Updated Goal Amount	This test verifies that the goal amount to be achieved is updated correctly based on the allocated monthly amounts and the desired goal amount.	The goal amount to be achieved should reflect the updated monthly contributions.	User has set goals with rates, desired amounts, and income entered.	This test ensures that the budgeting algorithm accurately updates the goal amount based on the allocated monthly contributions, providing an updated target for goal achievement.
Handle Changes in Income	This test checks if the budgeting algorithm adjusts the allocation amounts when the user's income changes.	The allocation amounts for each goal should be recalculated based on the new income value.	User has set goals with rates, desired amounts, and initial income entered. User updates the income value.	This test ensures that the budgeting algorithm can adapt to changes in the user's income and recalculates the allocation amounts accordingly.

12.3. Loan Part Tests

Test Name	Description	Expected Result	Precondition	Comment
Loan Calculation Accuracy	This test verifies that the loan calculation is accurate based on the loan amount, interest rate, and loan term.	The calculated monthly payment and total repayment amount should match the expected values.	User has entered a loan amount, interest rate, and loan term.	This test ensures that the loan calculation algorithm correctly determines the monthly payment and total repayment amount based on the input values.
Loan Eligibility Check	This test checks if the application correctly determines the user's eligibility for a loan based on their financial situation and goals.	The application should display an error message indicating that the password is incorrect.	User has provided their financial information and goals.	This test ensures that the loan eligibility check algorithm properly assesses the user's financial situation and goals to determine if they meet the criteria for obtaining a loan.
Loan Details Display	This test checks if the application accurately displays all relevant details of a loan, including the monthly payment amount, interest rate, loan term, and total repayment amount.	The loan details displayed should match the information entered by the user and the calculated values.	User has entered loan details and received loan offers.	This test ensures that the loan details are correctly stored and retrieved, providing the user with an accurate overview of the loan terms and repayment information.

12.4. Investment Part Tests

Test Name	Description	Expected Result	Precondition	Comment
Investment Allocation	This test verifies that the application accurately allocates the user's chosen percentage of income to investments.	The allocated investment amount should match the user's specified percentage of income.	User has specified the percentage of income to allocate towards investments.	calculates and allocates the desired percentage of income for investments.
Income Update After Investment Allocation	This test verifies that the user's income is correctly updated after allocating a percentage for investments.	The user's income should be reduced by the allocated percentage for investments.	User has specified the percentage of income to allocate towards investments and has a valid income value.	This test ensures that the application accurately updates the user's income by deducting the allocated percentage for investments.

13. User Documentation

13.1. User manual Operating instructions

To access SpendSmart, you'll need to create an account or log in if you already have one. Follow the prompts to register with your email address and create a secure password. Once registered, you can log in using your credentials to access your financial dashboard.

Navigate through SpendSmart features from the home page, including managing expenses and incomes, setting and tracking financial goals, receiving investment recommendations, managing savings and budgeting, assessing loan options, and using the calendar feature for reminders.

13.2. Maintenance guide

13.2.1. Operating Environment:

SpendSmart comprises two main components: SpendSmartServer and SpendSmartClient. SpendSmartServer serves as the backend, developed using Node.js and Express, and connects to a MongoDB database. SpendSmartClient is the frontend, built using React Native with Expo.

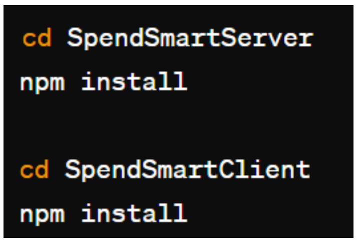
For maintaining the application and keep the development process updated you need to use the expo application to release new updates and fix bugs.

Start by cloning the SpendSmartServer and SpendSmartClient repositories from GitHub:

SpendSmartClient: https://github.com/WaelOtman211/SpendSmartClient.git

SpendSmartServer: https://github.com/WaelOtman211/SpendSmartServer.git

Once the repositories are cloned, navigate into each directory's Then, install the required dependencies by:



```
cd SpendSmartServer
npm install

cd SpendSmartClient
npm install
```

Then start the servers for both the backend and frontend by: npm start

13.3. GitHub Repositories URL:

For the Client Side: https://github.com/WaelOtman211/SpendSmartClient

For the Server Side: https://github.com/WaelOtman211/SpendSmartServer

14. References

1. What is revers Budgeting,
https://www.annuity.org/personal-finance/financial-wellness/reverse-budgeting/
2. What is Static Budgeting,
https://www.datarails.com/finance-glossary/static-planning-static-budget/
3. Budgeting types,
https://www.meettally.com/blog/budgeting-one-size-wont-fit-all
4. Mint Application
https://www.thebalancemoney.com/mint-com-manages-accounts-budgets-and-more-online-1293882
5. Personal Capital Application,
https://www.cnbc.com/select/personal-capital-budgeting-app-review/
6. Quicken Application,
https://en.wikipedia.org/wiki/Quicken
7. Expo go Explaining,
https://docs.expo.dev/get-started/expo-go/
8. React Native and what is Advantages and Disadvantages,
https://pagepro.co/blog/react-native-pros-and-cons/
9. What is Node.js and what is the Advantages and Disadvantages,
https://www.simform.com/blog/nodejs-advantages-disadvantages/
10. About the Investment and Stocks,
https://www.financestrategists.com/wealth-management/investments/?gclid=Cj0KCQjwnMWkBhDLARIsAHBOftoA6zb_QplgwhUe2R6z2oDbD1HjGzLStjGFcSLQIerPGpjkvivBj_QaAqCXEALw_wcB