

*Florida International University*  
*Knight Foundation School of Computing and Information Sciences*

Software Engineering Focus

# Final Deliverable

MoneyMap

**Team Members:**

Ronald Blanco  
Jerrick Wong  
Lily Padgett-Pflucker  
Natalie Garcia-Kearly  
Jesus Diaz

**Product Owner(s):** Ronald Blanco

**Mentor(s):** Masoud Sadjadi

**Instructor:** Masoud Sadjadi

The MIT License (MIT)  
Copyright (c) 2016 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### ***Abstract***

*This document provides the necessary information to understand the design, implementation, and functionality of **MoneyMap**, a comprehensive financial management application. MoneyMap was designed with users in mind, granting them the tools to take control of their finances. We do this by providing them with tools to track their income, expenses, savings, and debts while also providing extra features for their financial projections, smart accounting, and currency exchange.*

## Table of Contents

|                                                                                                                           |           |
|---------------------------------------------------------------------------------------------------------------------------|-----------|
| <b>INTRODUCTION .....</b>                                                                                                 | <b>5</b>  |
| CURRENT SYSTEM .....                                                                                                      | 5         |
| PURPOSE OF NEW SYSTEM .....                                                                                               | 5         |
| <b>USER STORIES</b>                                                                                                       |           |
| IMPLEMENTED USER STORIES .....                                                                                            | 6         |
| PENDING USER STORIES .....                                                                                                | 8         |
| <b>PROJECT PLAN</b>                                                                                                       |           |
| HARDWARE AND SOFTWARE RESOURCES .....                                                                                     | 9         |
| SPRINTS PLAN .....                                                                                                        | 10        |
| <i>Sprint 1</i> .....                                                                                                     | 10        |
| <i>Sprint 2</i> .....                                                                                                     | 10        |
| <i>Sprint 3</i> .....                                                                                                     | 11        |
| <i>Sprint 4</i> .....                                                                                                     | 12        |
| <i>Sprint 5</i> .....                                                                                                     | 13        |
| <i>Sprint 6</i> .....                                                                                                     | 14        |
| <i>Sprint 7</i> .....                                                                                                     | 14        |
| <b>SYSTEM DESIGN</b>                                                                                                      |           |
| ARCHITECTURAL PATTERNS .....                                                                                              | 16        |
| SYSTEM AND SUBSYSTEM DECOMPOSITION .....                                                                                  | 17        |
| DEPLOYMENT DIAGRAM .....                                                                                                  | 18        |
| DESIGN PATTERNS .....                                                                                                     | 19        |
| <b>SYSTEM VALIDATION .....</b>                                                                                            | <b>20</b> |
| <b>GLOSSARY .....</b>                                                                                                     | <b>21</b> |
| <b>APPENDIX .....</b>                                                                                                     | <b>22</b> |
| APPENDIX A - UML DIAGRAMS .....                                                                                           | 22        |
| APPENDIX B - USER INTERFACE DESIGN .....                                                                                  | 22        |
| APPENDIX C - SPRINT REVIEW REPORTS .....                                                                                  | 23        |
| APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST<br>DOCUMENT AND OTHER DOCUMENTS ..... | 25        |
| <b>REFERENCES .....</b>                                                                                                   | <b>27</b> |

## INTRODUCTION

MoneyMap aims to help users manage personal finances, an essential aspect of modern life. This is often a challenge for individuals who strive to balance their income, expenses, savings, and debt. MoneyMap was created with a user-friendly environment in mind, with the goal of simplifying these tasks. We do this by helping users organize their financial data, set and track savings goals, and gain valuable insights into their spending and saving habits.

### Current System

MoneyMap offers many features, such as detailed account management, for checking, savings, loans, and credit cards. Furthermore, MoneyMap allows you to track your expenses and income, analyze your income and expense habits, and visualize progress toward financial goals through interactive charts and progress bars. Its smart accounting capabilities offer actionable suggestions, such as debt payoff strategies and savings projections, empowering users to make informed financial decisions. The financial tools allow users to enter random data to calculate what financial decision is best for them and estimate currency exchange by using a live currency exchange estimator

### Purpose of New System

MoneyMap aims to create a centralized, intuitive platform where users can seamlessly manage their finances and plan for the future. By integrating robust functionality with a clean and accessible user interface, MoneyMap seeks to enhance financial literacy and confidence, enabling users to take control of their financial well-being.

## USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the MoneyMap project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

### Implemented User Stories

- *User Story #1:* As a team, we would like to go over the MoneyMap requirements and organize the roles and research required.
  - *Task:* Create a requirements document.
- *User Story #2:* As a team, we will research and discuss what features are going to be included in the program.
  - *Task:* View similar products, research potential resources.
- *User Story #3:* As a team, we discussed and dispersed the responsibilities among ourselves.
  - *Task:* Discussed.
- *User Story #4:* As a team, we would like to have an organized place to store our work so that we can quickly and effectively keep track of work.
  - *Task:* Create and setup a Github repository.
- *User Story #5:* Research datasets needed for the project.
  - *Task:* Find ways to calculate interest rates i.e., APR and APY calculations.
- *User Story #6:* Decide the tools we will use and how we will develop the app.
  - *Task:* Collaborate on the tools and structure that best fit the program.
- *User Story #7:* The users should have an easy-to-use interface that is intuitive and aesthetically pleasing to the eye.
  - *Task:* Sketch out a wireframe for the website/application layout.
- *User Story #8:* As a team, we will conduct surveys of possible users to understand needs and expectations.
  - *Task:* Figuring out the hierarchy of the program and what takes priority in the program.
- *User Story #9:* As a team, we will use tools to begin designing the app layout.
  - *Task:* Use Figma to begin designing the app layout.
- *User Story #10:* As a user, I want to calculate interest and other amounts in the app. So, we will be working on Algorithms for these calculations.

- *Task:* Work on the algorithms needed to calculate interest rates.
- *User Story #11:* Guest vs. Registered User's Features.
  - *Task:* Decide features will be included for guest accounts in comparison to registered Users.
- *User Story #12:* Trend Algorithm.
  - *Task:* Design an algorithm to handle the User data and return spending/income trend.
- *User Story #13:* Currency Exchange API.
  - *Task:* Work and refine our use of Google's Currency Exchange API so it can seamlessly integrate into our App.
- *User Story #14:* Database Setup.
  - *Task:* Set up a database to save User information when an account is connected.
- *User Story #15:* Front-End Design.
  - *Task:* Adapt Figma's template to input to our front-end design.
- *User Story #16:* Interest Rate Algorithm.
  - *Task:* Improve the algorithm to handle the User data and calculate interest rates.
- *User Story #17:* Quick Payoff Algorithm.
  - *Task:* Develop an Algorithm to offer QuickPay options for loans and credit card debt.
- *User Story #18:* Exchange Rate API.
  - *Task:* Work with Exchange Rate API to offer currency exchange option in our App.
- *User Story #19:* Front-End Design Alterations.
  - *Task:* Adapt and Fine-tune our Front-End design.
- *User Story #20:* Database Setup.
  - *Task:* Continue expanding and improving our database to save User information when creating an account.
- *User Story #21:* Expenses Adder Algorithm.
  - *Task:* Develop an Algorithm to add all the expenses registered Users enter.
- *User Story #22:* Database functions and connecting to Python code.
  - *Task:* Add database functionality to the Python code.
- *User Story #23:* Test and find bugs.
  - *Task:* Find and report bugs.
- *User Story #24:* Do presentation material.
  - *Task:* Look over and create presentation material.
- *User Story #25:* Prepare Final Deliverable.
  - *Task:* Get all files finished and ready for delivery.

## **Pending User Stories**

Incomplete: 0

## PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

### Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

1. Hardware:
  - a. Computers for development, testing, and use.
2. Software:
  - a. PyCharm as Integrated Development Environment (IDE) for Python code development.
  - b. GitHub as app repository, version control and collaboration.
  - c. Streamlit as Deployment Platform.
  - d. MongoDB as Database.
  - e. Figma for App Design.
  - f. Exchange Rate API for currency exchange live rate fetching.

## Sprints Plan

### *Sprint 1*

#### Requirements Gathering and Initial Setup

- Team Formation and Organization:
  - Formed the team, defined project scope, and established meeting schedules.
  - Set up a shared Google Drive for documentation and a GitHub repository for collaboration.
  - Delegated roles and responsibilities among team members.
- Requirements and Research:
  - Created a requirements document outlining core features and milestones.
  - Researched similar financial tools feature inspiration.
  - Decided on initial features: account management, expense tracking, budgeting, and savings goals.
- Project Setup:
  - Configured GitHub with a README and branching guidelines.
  - Installed dependencies, including Streamlit for UI and MongoDB for database integration.
- Outcome:
  - Established the foundational structure for MoneyMap and a collaborative workflow.

### *Sprint 2*

#### Feature Research and Design Planning

- Feature Research and Planning
  - Researched and finalized key program features.
  - Conducted further analysis on feature functionality, including expense tracking, budgeting, and goal setting.
  - Investigated required datasets.

- Explored methods to calculate interest rates, including APR and APY, and incorporated financial formulas for accuracy.
- Tool Selection and Development Strategy
  - Determined development tools and technologies.
  - Collaborated to finalize Streamlit for the UI, MongoDB for the database, and Firebase for authentication.
- Wireframe and UI Design
  - Ensured an intuitive and visually appealing user interface.
  - Designed a wireframe for the website and application layout to streamline user interactions.
- Outcome
  - Created a wireframe to guide UI development and align team expectations.

### ***Sprint 3***

#### **Feature Refinement and Initial Design**

- Feature Refinement
  - Researched and expanded on program features.
  - Continued to refine and finalize functionality, including financial tracking, reporting, and analytics.
  - Conducted detailed research on required datasets.
  - Focused on developing accurate methods for calculating financial metrics like APR and APY.
- User Insights and Prioritization
  - Conducted surveys to understand user needs and expectations.
  - Organized and prioritized program features based on user input.
  - Initiated app layout design using tools.
  - Utilized Figma to create initial wireframes and design prototypes for the app.
- Algorithm Development
  - Designed algorithms for financial calculations.

- Focused on building and testing algorithms for accurate interest and financial metric computations.
- Outcome
  - Identified user priorities and refined features accordingly.
  - Developed foundational algorithms and app design layouts.

### ***Sprint 4***

#### **Advanced Feature Integration and Design Implementation**

- Feature Differentiation
  - Defined features for guest accounts vs. registered users.
  - Determined unique functionalities available for guest accounts compared to full access for registered users.
- Algorithm Development
  - Developed a trend algorithm.
  - Designed and implemented an algorithm to analyze user data and identify spending and income trends.
- API Integration
  - Enhanced currency exchange API functionality.
  - Refined integration of Google's Currency Exchange API for seamless app usability.
- Database Configuration
  - Established a user database.
  - Set up a database to securely store user information and linked account data.
- Front-End Design
  - Implemented Figma design templates.
  - Adapted finalized Figma designs into the app's front-end interface for a polished user experience.
- Outcome
  - Differentiated user features between guest and registered accounts.

- Achieved integration of currency exchange functionality and trend analysis.
- Progressed on database setup and front-end development.

### ***Sprint 5***

#### **Advanced Algorithm Enhancements and Design Refinements**

- Interest Rate Calculation
  - Enhanced the interest rate algorithm.
  - Improved the algorithm to effectively handle user data and calculate accurate interest rates.
- Debt Payoff Optimization
  - Developed a Quick Payoff algorithm.
  - Created an algorithm to offer quick payoff options for loans and credit card debt.
- API Integration
  - Integrated the Exchange Rate API.
  - Enabled a seamless currency exchange feature within the app.
- Design Improvements
  - Refined front-end design.
  - Made adjustments and fine-tuned the interface to enhance user experience and visual appeal.
- Database Expansion
  - Expanded database functionality.
  - Continued improving the database to securely store and manage user information for newly created accounts.
- Outcome
  - Strengthened algorithmic capabilities for interest rate and debt management.
  - Successfully incorporated currency exchange features via API.
  - Advanced front-end design and database setup for improved usability.

### ***Sprint 6***

#### **Enhanced Algorithms, Front-End Refinements, and Database Integration**

- Algorithm Enhancements
  - Enhanced interest rate algorithms.
  - Improved algorithms to efficiently process user data and calculate accurate interest rates.
  - Developed a Quick Payoff algorithm.
  - Created an algorithm to offer quick payoff options for loans and credit card debt.
  - Implemented an Expenses Adder algorithm.
  - Designed an algorithm to aggregate all expenses entered by registered users.
- Front-End Design Refinements
  - Adapted and fine-tuned the front-end design.
  - Made interface adjustments to improve user experience and maintain consistency with the design template.
- Database Integration
  - Enhanced database functionality and integration.
  - Incorporated database functions into the Python code for seamless interaction between the app and the database.
- Outcome
  - Expanded the app's computational capabilities for financial management.
  - Improved the user interface to ensure a smoother experience.
  - Strengthened the backend with robust database functions integrated into the application logic.

### ***Sprint 7***

#### **Final Testing, Presentation, and Deliverables**

- Quality Assurance
  - Conducted comprehensive testing.
  - Identified, documented, and resolved bugs to ensure a stable and reliable application.

- Presentation Preparation
  - Developed presentation materials.
  - Reviewed project progress and created materials to effectively present the app's features and achievements.
- Final Deliverables
  - Prepared the final deliverables.
  - Ensured all files were completed, organized, and ready for submission.
- Outcome
  - Delivered a thoroughly tested and polished product.
  - Prepared engaging presentation materials to showcase the app's capabilities.
  - Assembled all necessary documentation and files for the final deliverable.

## SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

### Architectural Patterns

- Frontend (Streamlit)
  - Framework: The app's user interface is built using Streamlit, offering a simple and intuitive web-based experience.
  - Key Features:
    - Modular design with separate tabs for functionalities such as Checking, Savings, Income, Expenses, Loans, Credit Cards, Smart Accounting, and Overview.
    - Interactive forms for user input, real-time visualizations, and progress tracking.
    - Dynamic session states to manage user-specific data efficiently.
- Backend (MongoDB)
  - Database: MongoDB is used to store and manage user data securely.
  - Features:
    - User authentication and account management.
    - Data storage for savings accounts, expenses, income, loans, and credit card details.
    - Custom algorithms leverage the database for analytics and insights.
- API Services (Exchange Rate API)
  - Integration: The Exchange Rate API is integrated to provide real-time currency exchange rates.
  - Key Features:
    - Allows users to perform accurate currency conversions seamlessly.
- Algorithms and Tools
  - Custom Algorithms:

- Interest calculation algorithms for APR and APY.
- Expense tracking and QuickPay payoff algorithms for financial planning.
- Visualizations:
  - Streamlit's interactive charts and progress bars visualize trends, savings goals, and account statuses in real-time.

## System and Subsystem Decomposition

- Codebase Structure:
  - The MoneyMap project is structured for modularity and ease of development. Below are the key directories and their components:
- /pages Directory
  - About.py: Contains the logic and UI for the About section of the app.
  - guestPage.py: Manages the interface and features for guest users.
  - Homepage.py: Serves as the landing page for the application.
  - loggedInUserPage.py: Handles the main dashboard for logged-in users, integrating various account features.
  - signup.py: Manages user registration and authentication.
- Root Directory
  - DatabaseConnection.py: Manages the connection to the database and includes CRUD operations for user data.
  - financial\_tools.py: Implements financial calculators such as APR and APY calculations.
  - helperFunctions.py: Contains helper functions used across different modules for reusable functionality.
  - main.py: Serves as the entry point for the application, initializing the user interface and setting up navigation.
  - README.md: Provides an overview of the project, installation steps, and usage instructions.
- Core Feature Modules
  - checking\_account.py: Handles logic and UI for managing checking accounts.
  - savings\_account.py: Manages savings account features, including deposits and balance tracking.
  - expenses\_account.py: Allows users to track and manage their expenses.

- `income_account.py`: Handles user income entries and income-related features.
  - `loans_account.py`: Manages loan accounts and repayment tracking.
  - `creditcard_account.py`: Provides features for managing credit card balances and transactions.
  - `overview_accounts.py`: Offers an overview of all account balances and activity summaries.
  - `smart_accounting.py`: Contains advanced financial tools like savings goal trackers and spending insights.
  - `savings_goals.py`: Manages the creation, tracking, and modification of user-defined savings goals.
- Media and Assets
  - `MoneyMapLogo.png/JPG`: Logo assets used across the application.
  - `piChart.png`: Sample pie chart visualization for financial data representation.
- Additional Tools
  - `test_connection.py`: Debugging and testing module for database connections.
  - `Calculator.png`, `Coins.png`: Static images used in the UI for visual appeal.

## Deployment Diagram

- Frontend Deployment:
  - The frontend is built using Streamlit, which is a Python-based web application framework.
  - The user interacts with the application via a web browser.
- Backend Deployment:
  - The backend consists of Python scripts that handle business logic, calculations (e.g., APR, savings goals), and database operations.
  - Backend logic integrates with MongoDB for database management, ensuring data is stored and retrieved efficiently.
- Database:
  - A MongoDB Atlas Cloud Instance is used to store user data, financial records, and savings goals.
  - Data is securely accessed via MongoClient, with authentication and SSL encryption to protect sensitive information.
- API Integrations:

- APIs, such as Exchange Rate API, are used for dynamic data like currency conversions.
  - Cloud-based APIs connect to external services, providing functionality without the need for extensive server infrastructure.
- Deployment Environment:
  - The system is deployed in a Python virtual environment using PyCharm for development.

## Design Patterns

- Model-View-Controller (MVC):
  - Model: The MongoDB database serves as the model, managing user data and application state.
  - View: The Streamlit interface acts as the view, presenting data to the user and capturing input.
  - Controller: Backend Python scripts (e.g., `smart_accounting.py`, `savings_goals.py`) process user input, manage application logic, and interact with the database.
- Singleton Pattern:
  - The `DatabaseConnection.py` file implements a singleton-like behavior, ensuring only one connection instance to MongoDB is created, reducing resource usage.
- Observer Pattern:
  - Notifications for users, such as updates on savings progress, are designed using an observer pattern where changes in data trigger specific updates in the UI.
- Strategy Pattern:
  - Algorithms for APR/APY calculations, savings goal tracking, and expense management are modularized to allow swapping or enhancing algorithms without affecting the system's overall design.
- Factory Pattern:
  - Modular functions (e.g., `get_savings_accounts()`, `save_expense_account()`) act like factories, abstracting database operations for easy use across multiple modules.
- Template Pattern:
  - Reusable templates are applied in the UI (e.g., repeating layouts for checking accounts, savings accounts, and expenses) for consistent user experience.

## **SYSTEM VALIDATION**

System validation for MoneyMap ensured that all functionalities, such as savings goal tracking, APR/APY calculations, and income/expense management, operate accurately and efficiently. Security measures, including robust authentication and data protection, were verified. Comprehensive end-to-end testing ensured the system reliably meets user expectations, providing a secure and efficient financial management tool.

## GLOSSARY

**APR (Annual Percentage Rate):** The annual rate charged for borrowing or earned through an investment, expressed as a percentage.

**APY (Annual Percentage Yield):** The annual rate of return earned on an investment, including compound interest.

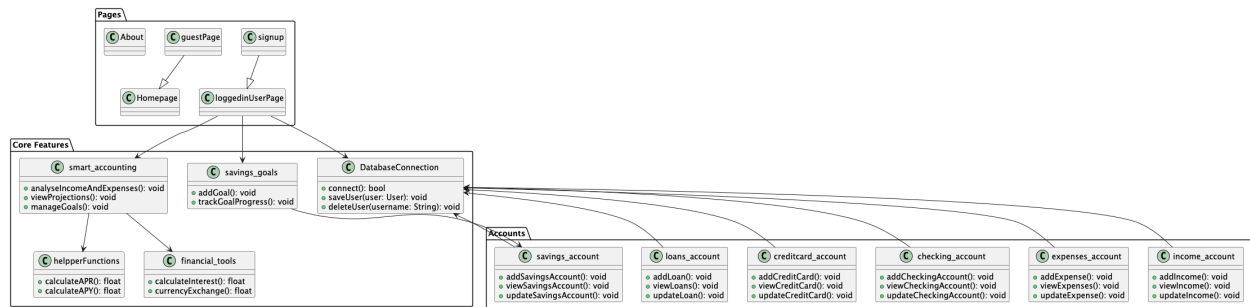
**Currency Exchange API:** A software interface that provides real-time exchange rates for converting currencies.

**MongoDB:** A backend service used for database management, authentication, and hosting.

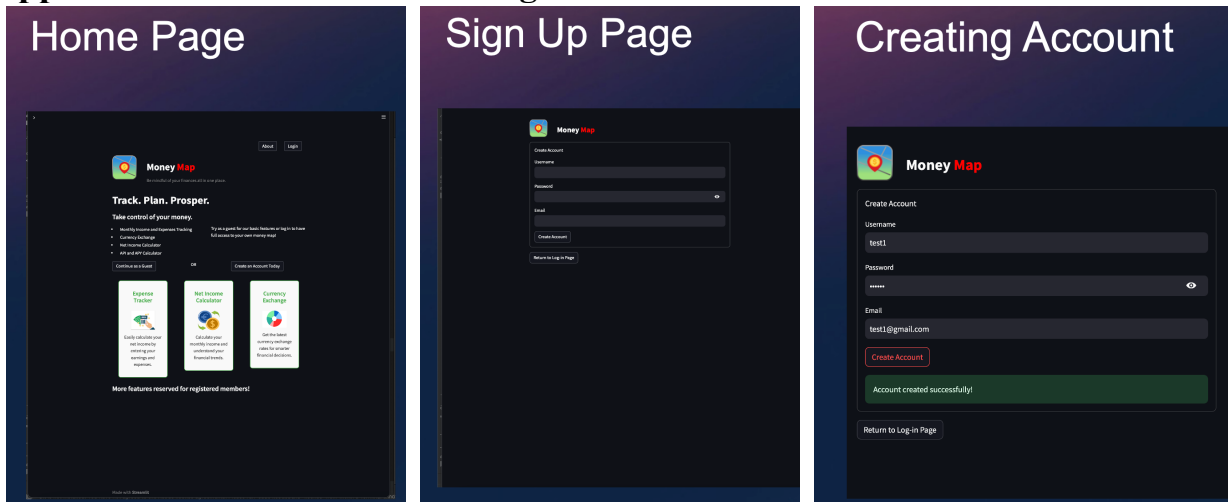
**Interest Rate Algorithm:** A formula or method used to calculate interest rates for loans or savings.

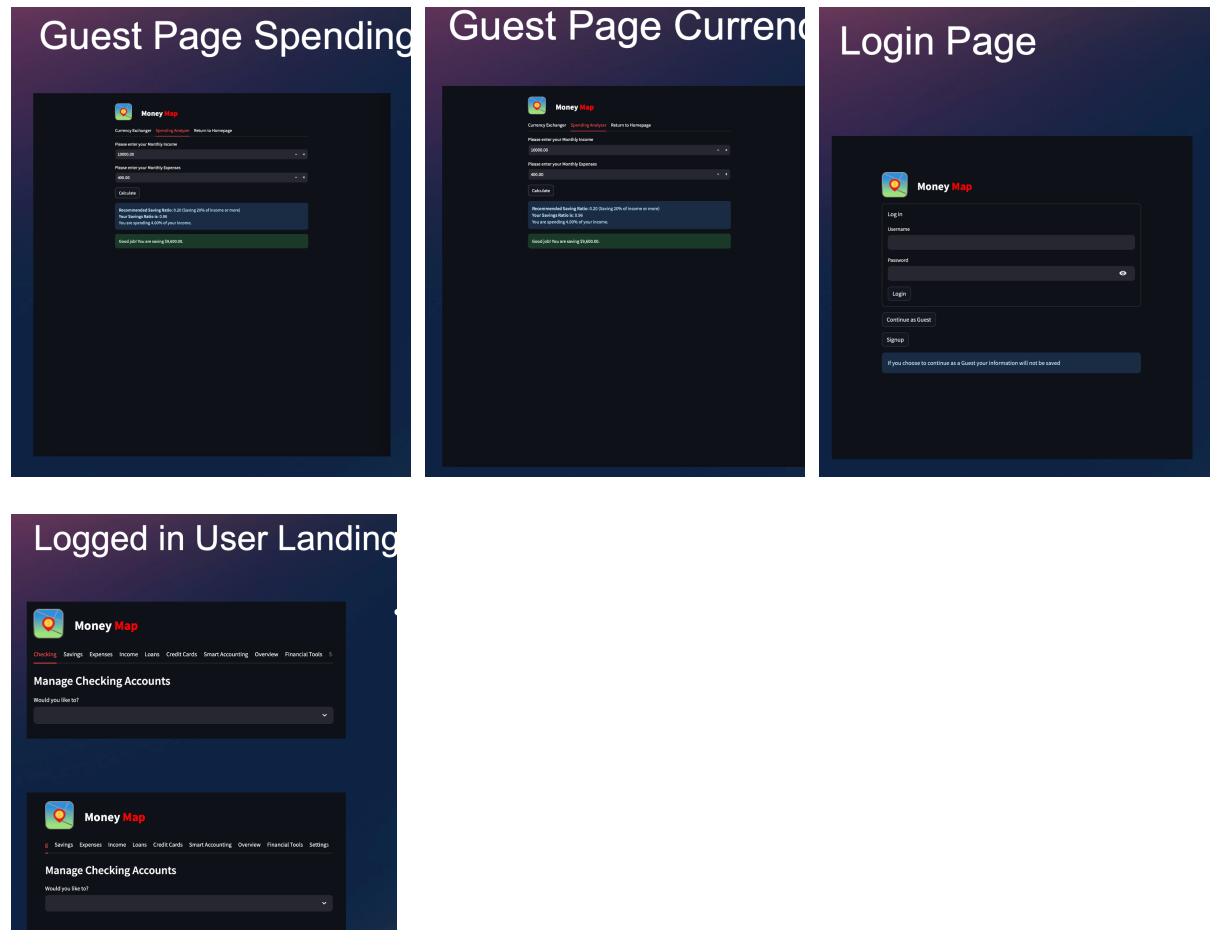
## APPENDIX

### Appendix A - UML Diagrams



### Appendix B - User Interface Design





## Appendix C - Sprint Review Reports

The MoneyMap project progressed through six focused sprints, starting with team formation, setting up resources like GitHub and Firebase, and initial research into features and requirements. Subsequent sprints saw advancements in developing algorithms for interest calculations, integrating the Exchange Rate API, and designing a user-friendly interface with Figma. The team refined the database, added functionality for expense tracking, and differentiated features for guest and registered users. In later sprints, rigorous testing resolved

bugs and ensured smooth functionality across modules, while final deliverables were polished, validated, and prepared for presentation, demonstrating a collaborative and methodical approach throughout.

## **Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents**

- User Manual
- Introduction
  - Access the app at <https://money-map-fokubi2sabueoupjdkajhz.streamlit.app>
  - MoneyMap is designed to assist users in tracking expenses, managing budgets, setting financial goals, and analyzing financial trends.
- Features Overview
  - Expense Management: Add and categorize expenses.
  - Savings Goal Tracker: Create, track, and update financial goals.
  - Income Analysis: Visualize income trends and compare them with expenses.
  - QuickPay Options: Plan loan or credit card debt payoff strategies.
  - Currency Exchange: Convert currencies using real-time exchange rates.
- Step-by-Step Guide
  - Access the App:
    - Open a web browser and navigate to the app URL .(<https://money-map-fokubi2sabueoupjdkajhz.streamlit.app>)
  - Register/Login/Use Guest:
    - Sign up using your email or log in with existing credentials.
  - Navigate Tabs:
    - Use the tabs (e.g., Savings, Expenses, Income) to explore specific features.
  - Add Data:
    - Enter financial data such as expenses, income, or savings goals.
  - Track Progress:
    - Use graphs and progress bars to monitor your financial trends.
  - Customize Goals:
    - Set or update financial goals under the Savings Goal Tracker.
- Installation/Maintenance
  - Deployment
    - The app is hosted online at <https://money-map-fokubi2sabueoupjdkajhz.streamlit.app>

- Deployment uses Streamlit Cloud, removing the need for local installation.
- Maintenance
  - Database: Regularly monitor MongoDB for performance and backup user data.
  - Updates:
    - Push updates to the GitHub repository.
    - Changes auto-deploy to Streamlit Cloud.
  - Environment Variables:
    - Ensure API keys and sensitive data remain updated.
  - Troubleshooting
    - Connectivity Issues:
      - Check the internet connection.
      - Verify Streamlit Cloud status.
    - Data Loss:
      - Restore from MongoDB backups.
    - Feature Errors:
      - Debug and test locally before redeploying.
- Shortcomings/Wishlist
  - Shortcomings
    - Offline Access: The app requires internet connectivity.
    - Customization: Limited ability to customize interface layouts.
    - API Dependence: Reliant on third-party APIs (e.g., Exchange Rates).
    - Scalability: The free Streamlit Cloud plan might face performance issues with heavy traffic.
  - Wishlist
    - Mobile App: Build a dedicated mobile app for Android/iOS.
    - Offline Mode: Allow users to input data offline and sync later.
    - Enhanced Analytics: Add more advanced financial analysis tools.
    - Integration: Link with bank accounts for automated data imports.
    - User Personalization: Offer themes or layout adjustments.

## REFERENCES

<https://www.exchangerate-api.com> (Exchange Rate API)

<https://www.grammarly.com/> (Grammar and text editing)