Project Title: FinWise Manager

Group name: Smart Spend  
Names: Alex Nikirk, Albert Mitchell

Date: April 7, 2024

[1.0 Introduction 3](#_Toc163419587)

[1.1 Purpose 3](#_Toc163419588)

[1.2 Scope 3](#_Toc163419589)

[1.3 Overview 3](#_Toc163419590)

[1.4 Reference Material 3](#_Toc163419591)

[1.5 Definitions and Acronyms 4](#_Toc163419592)

[2.0 System Overview 4](#_Toc163419593)

[3.0 System Architecture 5](#_Toc163419594)

[3.1 Architectural Description 5](#_Toc163419595)

[3.2 Decomposition Description 6](#_Toc163419596)

[3.2.1 User Account Management Module 6](#_Toc163419597)

[3.2.2 Financial Management Module 7](#_Toc163419598)

[3.2.3 Security and Compliance Module 7](#_Toc163419599)

[3.2.4 Data Management Module 7](#_Toc163419600)

[3.3 Design Rationale 8](#_Toc163419601)

[4.0 Data Design 9](#_Toc163419602)

[4.1 Data Description 9](#_Toc163419603)

[4.2 Data Dictionary 10](#_Toc163419604)

[5.0 Component Design 10](#_Toc163419605)

[6.0 Human Interface Design 11](#_Toc163419606)

[6.1 Overview of User Interface 11](#_Toc163419607)

[6.2 Screen Images 12](#_Toc163419608)

[6.3 Screen Objects and Actions 16](#_Toc163419609)

[7.0 Requirements Matrix 18](#_Toc163419610)

# 1.0 Introduction

1.1 Purpose

The purpose of this Software Design Document (SDD) is to provide a comprehensive architectural and design overview of FinWise Manager, a personal finance management application. This document is intended to serve as a roadmap for development teams by detailing the system's architecture, data design, component interactions, and user interface specifics. Additionally, it aims to ensure that all stakeholders have a clear understanding of the design decisions and methodologies applied in developing FinWise Manager.

1.2 Scope

FinWise Manager is designed to empower users with tools for managing their personal finances effectively. It encompasses features such as account management, transaction tracking, budget setting, financial reporting, and insights generation. This document covers the application's design considerations, including the architectural layout, data storage strategies, algorithms, and user interface designs. It outlines the structure that will support the application's requirements as defined in the Software Requirements Specification (SRS) document, ensuring that FinWise Manager is robust, user-friendly, and scalable.

1.3 Overview

This SDD is structured to provide a clear and detailed exposition of FinWise Manager's design. Following this introduction, the document is organized into several key sections:

* **System Overview**: Offers a general description of FinWise Manager, highlighting its objectives and expected functionality.
* **System Architecture**: Describes the system's architectural strategy, including major components and their interactions.
* **Data Design**: Details the data management approach, including the database schema and data flow mechanisms.
* **Component Design**: Explores the design of individual system components, their responsibilities, and operational logic.
* **Human Interface Design**: Presents the user interface design, focusing on usability, navigation, and accessibility.

The document concludes with a Requirements Matrix linking back to the initial SRS document, ensuring that all specified requirements are addressed by the design. Appendices provide additional diagrams, code snippets, and technical details supporting the design choices made.

1.4 Reference Material

IEEE SDD standard and SDD template provided by the professor.

IEEE Std 1016-2009

Software Requirements Specification (SRS) for FinWise Manager

Financial Data Services: [Plaid API documentation](https://plaid.com/docs/)

Database Services: [MongoDB Atlas](https://www.mongodb.com/docs/atlas/)

1.5 Definitions and Acronyms

API: Application Programming Interface

UI: User Interface

UX: User Experience

GUI: Graphic User Interface

FR: Functional Requirements

# 2.0 System Overview

A diagram of a computer code

Description automatically generated

Figure 1 depicts the architectural framework chosen for FinWise Manager, designed to seamlessly interface with web and mobile platforms while ensuring compatibility with a range of technologies. At the core lies the Application Logic Tier, powered by Node.js, facilitating robust communication with web servers like Apache and ensuring compatibility across platforms, including iOS and Android. This tier orchestrates interactions with third-party APIs, enabling vital functionalities such as importing transactions and sending email notifications.

The front-end is crafted with HTML, CSS, and JS to provide a consistent user interface experience across devices. Its adaptability ensures smooth navigation on both web and mobile platforms, catering to a diverse user base. The Data Management Tier, equipped with MongoDB and MySQL, stores essential user data and financial information, guaranteeing scalability and reliability.

By harmonizing these tiers, FinWise Manager offers a comprehensive financial management solution. Its architecture, designed for compatibility and scalability, ensures seamless operation across platforms, bolstered by integrations with essential third-party APIs. This streamlined approach empowers users with efficient financial management tools, supported by a robust and versatile technological foundation.

# 3.0 System Architecture

## 3.1 Architectural Description

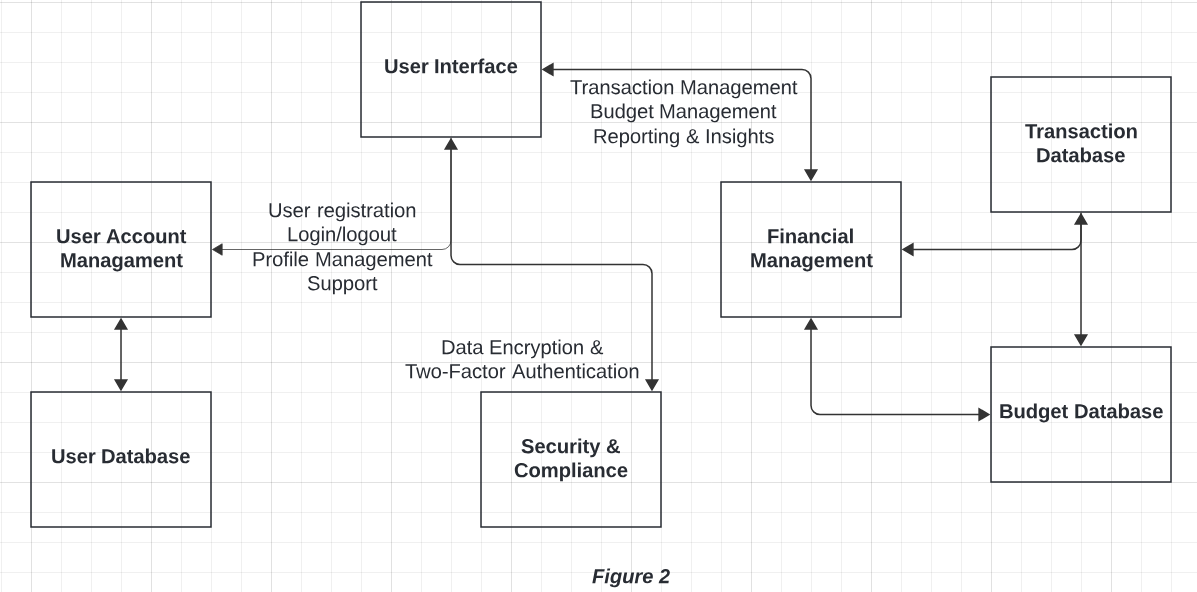


Figure 2 provides a detailed depiction of the system architecture, highlighting the synergies between its core elements. Central to this structure is the User Account Management module, which oversees user registration, login/logout processes, and profile management. Seamlessly interfacing with the User Database ensures the secure storage and retrieval of user data. Additionally, the Financial Management module streamlines transaction handling, facilitating additions, edits, and deletions while interfacing with the transaction and budget databases. This module empowers users with detailed spending reports and personalized financial insights. Ensuring data integrity and security, the Security and Compliance module implements encryption measures and optional two-factor authentication. Throughout this intricate network, the UI integrates seamlessly, offering users effortless access to a myriad of features and functionalities.

## 3.2 Decomposition Description

A diagram of a company

Description automatically generated

Figure 3 demonstrates the operational flow and dependencies among the modules of the FinWise Manager, highlighting the system's modular architecture. Central to this design is the User Account Management module, which orchestrates user registration, authentication, and profile management processes. It interacts closely with the User Database, ensuring secure data handling and retrieval. This module is pivotal for initializing user sessions and maintaining accurate user profiles, which are essential for tailoring the application's financial management features to individual needs.

On the financial front, the Financial Management module serves as the backbone for transaction handling, budgeting, and reporting. It processes user transactions, aids in budget setting and monitoring, and generates comprehensive financial insights by interfacing with Transaction and Budget Databases. Meanwhile, the Security and Compliance module underpins the system's integrity, implementing encryption and optional two-factor authentication to protect user data. These modules collectively provide a robust framework for financial planning and management, integrated seamlessly through a user-friendly interface that facilitates intuitive navigation and access to the application's comprehensive financial tools.

3.2.1 User Account Management Module

This module is pivotal to the operation of FinWise Manager, handling all aspects of user interaction with their accounts. It includes:

**Registration Sub-module**: Allows new users to create an account by providing essential details. Incorporates verification mechanisms to ensure data accuracy.

Authentication Sub-module: Manages login and logout processes, employing secure methods to verify user credentials.

**Profile Management Sub-module**: Enables users to view and update their personal information and account settings.

3.2.2 Financial Management Module

Central to FinWise Manager's functionality, this module provides tools for users to manage their financial data effectively.

**Transaction Management Sub-module**: Facilitates the addition, modification, and deletion of financial transactions.

**Budgeting Sub-module**: Allows users to set and adjust budgets for various categories, tracking spending against these thresholds.

Reporting and Insights Sub-module: Generates reports and financial insights based on user data, aiding in financial planning and decision-making.

### 3.2.3 Security and Compliance Module

Ensures the integrity and security of the application and its data, crucial for user trust and regulatory compliance.

**Data Encryption Sub-module**: Implements encryption protocols for sensitive data storage and transmission.

**Authentication and Authorization Sub-module**: Aside from basic login mechanisms, it offers optional two-factor authentication for enhanced security.

**Compliance Sub-module**: Regularly updates and checks against financial regulations and data protection laws to ensure the application remains compliant.

### 3.2.4 Data Management Module

Handles the storage, retrieval, and integrity of data within FinWise Manager.

User Database: Stores user-related data securely, accessible only by authorized components of the application.

**Transaction Database**: Maintains records of all user transactions, ensuring data is accurate and up to date.

**Budget Database**: Keeps track of user-defined budgets and spending limits, interfacing with the Financial Management Module for data retrieval and analysis.

Each module and sub-module are designed to function independently while contributing to the system's overall capabilities, ensuring that FinWise Manager is not only robust and secure but also flexible and scalable. This modular approach simplifies maintenance and future development by allowing individual components to be updated or replaced without impacting the entire system.

## 3.3 Design Rationale

The architectural design choices for FinWise Manager are driven by the goal of creating a highly usable, secure, and scalable personal finance management tool. Here’s the reasoning behind key decisions in our system’s design:

**Modular Architecture**

We adopted a modular design philosophy to ensure each component of FinWise Manager can be developed, maintained, and updated independently. This approach not only facilitates easier debugging and faster development cycles but also allows for scalable expansion in the future as new features or services are introduced.

**Choice of Technology Stack**

Node.js for the Application Logic Tier: Selected for its non-blocking I/O model, which efficiently handles concurrent requests, making it suitable for our application’s potentially high-volume operations. Its vast ecosystem and support for server-side JavaScript also streamline development.

HTML, CSS, and JavaScript for the Front-End: These core web technologies were chosen to guarantee broad compatibility across devices and browsers, ensuring all users have consistent access to FinWise Manager regardless of their platform.

MongoDB and MySQL for Data Management: Utilizing both NoSQL and SQL databases allows us to optimize storage and querying strategies for different types of data, from unstructured transaction records (MongoDB) to structured user information (MySQL), enhancing performance and scalability.

**Security Measures**

Implementing robust security protocols, including data encryption and optional two-factor authentication, was non-negotiable. These measures protect users' sensitive financial data and personal information, addressing growing concerns around data privacy and security in digital tools.

**User-Centric Interface Design**

The decision to craft a responsive and intuitive user interface comes from a commitment to accessibility and ease of use. Ensuring that FinWise Manager provides a seamless experience across web and mobile platforms is crucial for user satisfaction and engagement.

The rationale behind these design choices reflects our desire to deliver a reliable, user-friendly, and adaptable finance management tool. Each decision aligns with our core objectives: enhancing user experience, ensuring data security, and supporting future growth.

# 4.0 Data Design

## 4.1 Data Description

**User Entity**

Attributes: UserID, Name, Email, Password, Date Created

Description: Represents individual users of FinWise Manager. Each user has a unique identifier, personal information, and login credentials.

**Transaction Entity**

Attributes: TransactionID, UserID, Amount, Category, Date, Description

Description: Captures financial transactions made by the user. This includes income and expenses, each tied to a user via UserID. Transactions are categorized for easier management and reporting.

**Budget Entity**

Attributes: BudgetID, UserID, Category, Limit, StartDate, End Date

Description: Defines budget limits set by users for different categories over specific time frames. Budgets help users control their spending according to their financial goals.

**Report Entity**

Attributes: ReportID, UserID, Period, Generated Date

Description: Generated reports based on user transactions and budgeting information. Reports provide insights into spending habits and financial health over time.

**Relationships:**

User to Transaction: One-to-Many. A single user can have multiple transactions, but each transaction is associated with only one user.

User to Budget: One-to-Many. Users can set multiple budgets for different categories, but each budget is specific to one user.

User to Report: One-to-Many. Users can generate multiple reports; however, each report is specific to one user and their financial data.

This data model is structured to ensure scalability, supporting a growing number of users and transactions. Relationships between entities are designed to maintain data integrity and provide a holistic view of each user’s financial landscape within FinWise Manager.

## 4.2 Data Dictionary

Provide a detailed data dictionary listing and describing all the data elements used by FinWise Manager.

# 5.0 Component Design

Each component within FinWise Manager is designed with clear interfaces, well-defined dependencies, and efficient processing logic to fulfill its specific responsibilities while seamlessly integrating with other components to provide users with a comprehensive and intuitive financial management experience.

Here's a breakdown of some key components:

1. User Account Management Component

* Interface: Provides endpoints for user registration, login/logout, and profile management.
* Dependencies: Relies on the Database Systems for storing user data and the Security and Compliance module for authentication and data security.
* Processing Logic: Implements robust validation mechanisms for user input during registration and profile updates. Utilizes secure authentication protocols to verify user identities during login/logout operations.

1. Financial Management Component

* Interface: Offers endpoints for managing transactions, budget setting, and generating financial reports.
* Dependencies: Interacts with the Transaction Database for storing transactional data and the Budget Database for budget-related information. May also utilize third-party APIs for additional financial data services.
* Processing Logic: Enforces validation rules for transaction entries to ensure data integrity. Utilizes efficient algorithms for budget calculations and analysis to provide users with accurate financial insights. Implements mechanisms for fetching and processing external financial data via APIs.

1. Security and Compliance Component:

* Interface: Provides interfaces for encryption/decryption operations, user authentication, and optional two-factor authentication.
* Dependencies: Relies on cryptographic libraries for encryption/decryption operations and interfaces with the User Database for user authentication data.
* Processing Logic: Implements strong encryption algorithms to safeguard sensitive user data. Utilizes secure authentication protocols such as OAuth or JWT for user authentication. Optionally integrates two-factor authentication mechanisms for enhanced security.

1. UI Component:

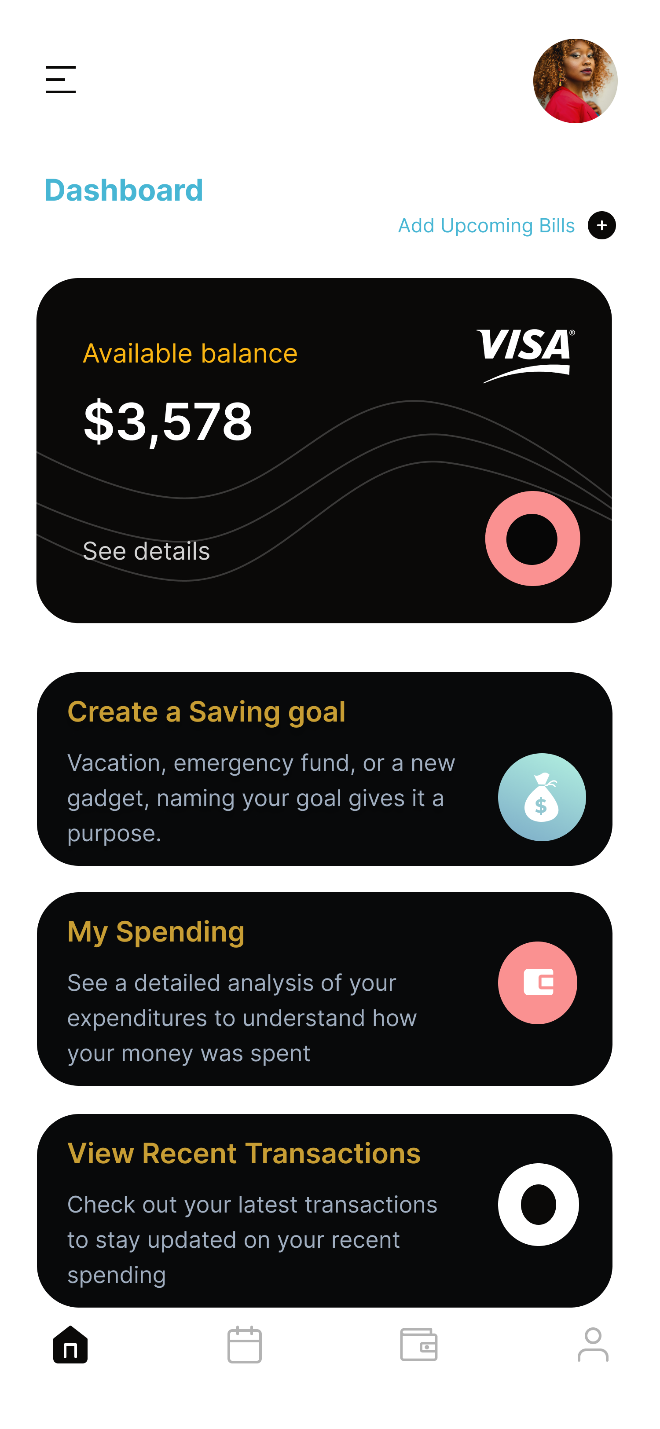
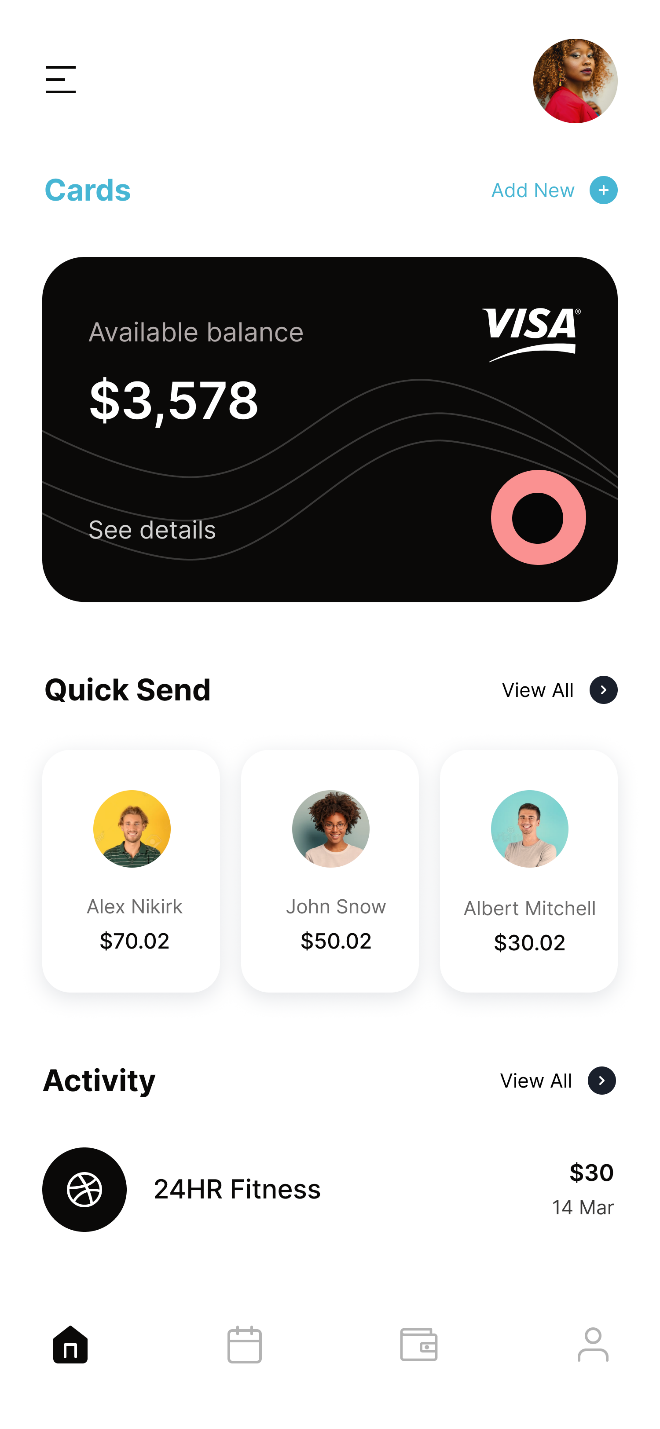
* Interface: Presents a user-friendly interface for interacting with FinWise Manager functionalities.
* Dependencies: Interfaces with all other components for data retrieval and processing.
* Processing Logic: Implements client-side logic for rendering UI elements, handling user input, and making requests to backend APIs. Utilizes responsive design principles to ensure optimal user experience across different devices and screen sizes.

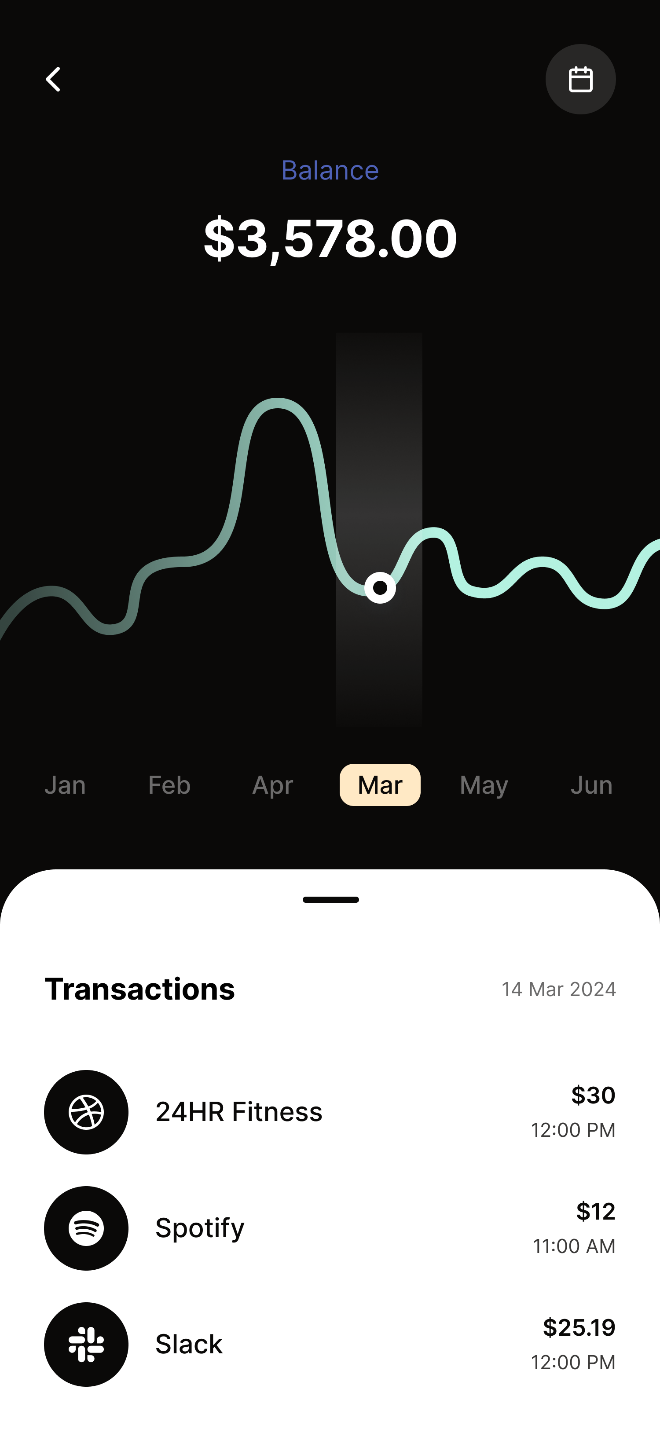
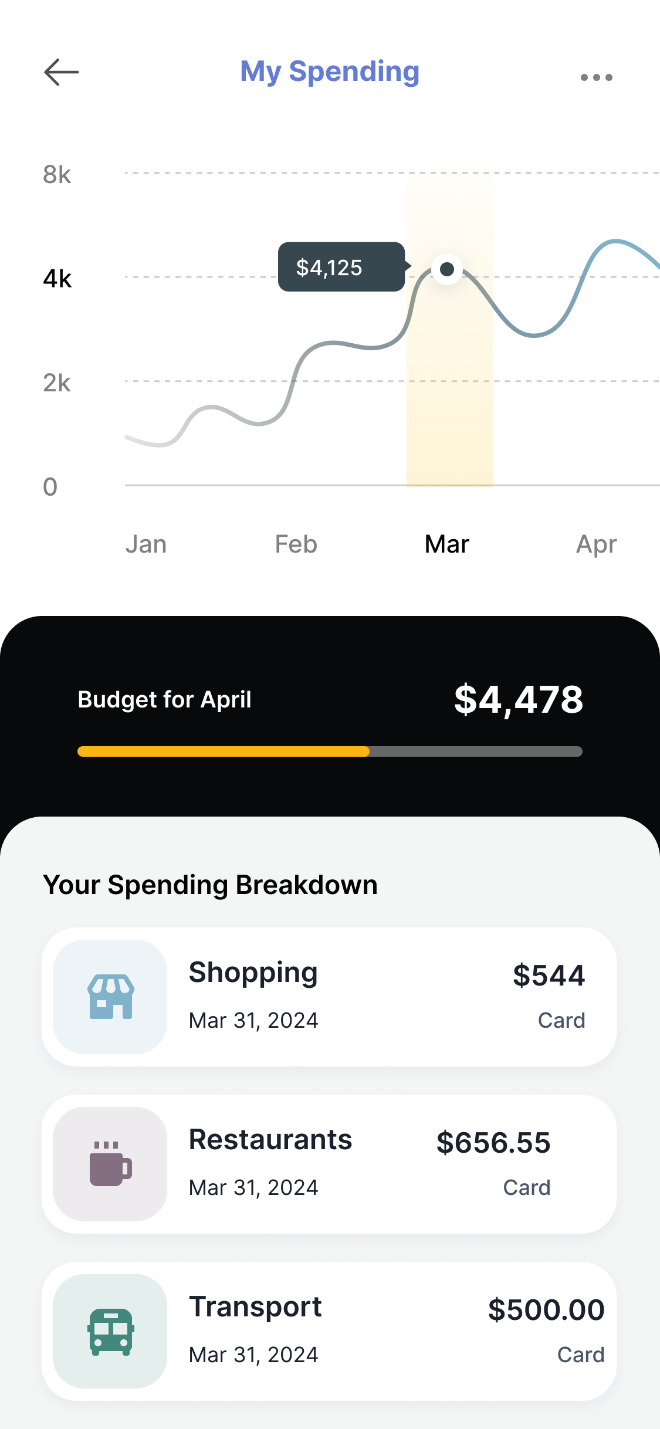
# 6.0 Human Interface Design

## 6.1 Overview of User Interface

The FinWise Manager UI aims to provide a seamless, intuitive experience that allows users to manage their finances with ease. It emphasizes clarity, minimizing cognitive load with a clean layout and straightforward navigation. Accessibility is also a core principle, ensuring the application is usable by as wide an audience as possible, including those with disabilities.

## 6.2 Screen Images

## 6.3 Screen Objects and Actions

**Navigation Menu Icon**

* Object: Hamburger icon on the top left
* Action: Tapping this icon expands the main navigation menu, allowing users to access different areas of the app such as transactions, budgets, reports

**Profile Image and Add Upcoming Bills**

* Object: User profile picture on the top right with an "Add Upcoming Bills" button next to it.
* Action: Clicking the profile picture takes the user to their profile settings. The "+" button opens a form to add upcoming bill details like due dates and amounts.

**Available Balance Display**

* Object: Card showcasing the available balance with a "See details" link.
* Action: A click reveals a detailed view of the account balances across different accounts and/or a more comprehensive breakdown of recent transactions contributing to the current balance.

**Create a Saving Goal Button**

* Object: Button labeled "Create a Saving Goal".
* Action: Launches a step-by-step wizard or form that guides the user through setting up a new financial goal, including naming the goal, setting a target amount, and establishing a timeline.

**My Spending Button**

* Object: Button labeled "My Spending".
* Action: Directs users to a detailed spending analysis page, which includes graphs and charts that break down expenses by category, time period, etc.

**View Recent Transactions Button**

* Object: Button labeled "View Recent Transactions".
* Action: Takes the user to a page or section where they can browse through a detailed list of all recent transactions, offering options to filter, sort, and search through their financial activity.

**Bottom Navigation Bar**

* Objects: Icons representing home (dashboard), transactions, budgets, and more.
* Actions: Each icon serves as a shortcut to its respective page within the app. The currently active section (e.g., home/dashboard) is highlighted to indicate where the user is currently located in the app.

# 7.0 Requirements Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| SRS Requirement ID | Requirement Description | Design Component | Design Section |
| FR1 | User Registration | User Account Management Module | 3.2.1 |
| FR2 | User Login/Logout | User Account Management Module | 3.2.1 |
| FR3 | Profile Management | User Account Management Module | 3.2.1 |
| FR4 | Add Transactions | Financial Management Module | 3.2.2 |
| FR5 | View Transactions | Financial Management Module | 3.2.2 |
| FR6 | Edit/Delete Transactions | Financial Management Module | 3.2.2 |
| FR7 | Set Budgets | Financial Management Module | 3.2.2 |
| FR8 | View Budgets | Budget Management Module | 3.2.2 |
| FR9 | Budget Notifications | Budget Management Module | 3.2.2 |
| FR10 | Spending Reports | Reporting Module | 3.2.2 |
| FR11 | Financial Insights | Insights Module | 3.2.2 |