SP-26 PURPLE BUDGET APP

SOFTWARE DESIGN DOCUMENT (SDS)

Course: Senior Project Section 01 CS4850
Semester: Spring 2024
Professor Perry
Date: 2/8/2024

Team:


Cameron Lowry


Ann Nguyen


Alexandra Clark

1. Introduction

1.1. Document Description

1.1.1. Introduction

The Software Design Document (SDD) describes the system and the architecture of a mobile budgeting app known as MoneyFlow. MoneyFlow is a mobile application that will help users budget their money and make smart financial decisions. This document is designed to help Project Member and our project leader plan and implement MoneyFlow. The scope of the SDD will focus on all major design decisions and major components of the system. This will include design considerations such as constraints and assumptions, our strategies for our designs as well as a detailed look into our design.

1.1.2. System Overview

The software is a budget management application designed to help users track and manage their finances effectively. The app will provide various features to assist users in budget and expenses tracking. It is developed by React Native, allowing users to access the app both on Android and iOS:

Functionalities:

- Expenses tracking allows users to track their expenses by linking to the bank account or enter by their own.
- Income tracking: users can also track their income sources and view their total income over a specified period.
- User authentication and Security: to ensure the security of user data. The app includes user authentication features such as login/logout functionality and data encryption.

Design approach:

- The system is divided into two subsystems: budget management and user management subsystem. Each subsystem is responsible for handling the specific aspect of the application functionality.
- The overall system focuses on a user-friendly interface, with intuitive navigation and clear visualization of financial data. The app design also emphasizes security to protect user database and profile.
- The system is structured to allow scalability and flexibility. New features can be added easily, and an existing feature can be modified.
- Overall, the app is designed to provide users with comprehensive tools for managing their finances effectively, with a focus on security, usability and scalability.

2. Design Considerations

2.1. Assumptions and Dependencies

There are several assumptions that we as the team of MoneyFlow make about this system and its implementation. The first assumption is that this system will be used to manage someone's budget. Our next assumption is that the user will be using a mobile device. For these devices we will be focusing on the Android and IOS operating systems. We will also assume that the user is using a bank that is compatible with our system. The end-user is also assumed to be able to use basic mobile features such as a touch screen and will be able to read the English language.

2.2. General Constraints

Due to the limited size and scope of this project there are several core constraints that will need to be accounted for in the design of the system's software.

Hardware environment

The constraints of the hardware of multiple different types of mobile devices from a variety of different companies that run on IOS and Android. This limits the development of this software due to having to account for both IOS and Android system requirements.

Availability of resources

As a small un-funded group our resources will be very limited. This means the use of high-level programs and hardware will be extremely limited.

Interoperability

This software system must communicate between multiple systems including our database and various banks' systems to function. Therefore, the application must be able to send and receive information from a variety of different sources.

Security

This project must be able to protect the data of the end-user from various bad actors. This will require security both for the database and for the application.

End-user Environment

The end user will want to use a budgeting app on the move and be able to quickly check and change their budget. This constrains us to mobile applications that need to be able to connect with a user's bank account and

2.3. Goals and Guidelines

The goal for this system is to create an easy and fun to use application to budget money and reach user's financial goals. To do this we follow a collection of guidelines and rules. One of our first guidelines is simplicity of design. Our team wants to focus on giving users a fast and optimized experience. Focusing on a simple design allows us to streamline our code for better optimization and streamline interface for users. Another guideline we have for coding and development is readability. All our code should be easily readable with the use of comments and correct naming conventions. This is to allow seamless maintenance and other developers to understand what we have written. We hope these guidelines will help us achieve our goal to create a fast, streamlined user-friendly budget app.

2.4. Development Methods

The development approach we are using for this project is the Waterfall Model. This is a software development model that heavily uses documents and a sequential approach to development. Waterfall models have six steps requirements gathering, design, implementation, integration and testing, deployment and finally maintenance. Models like this allow us to take a structured approach to fulfill this system's requirements. We are also using the idea of design sprints from the Agile method. This allows our team to accomplish goals on time and with a well-documented and planned production schedule.

3. Architectural Strategies

1. **Programming language and framework**

- Choice of programming language: JavaScript for its compatibility with React Native, used to build cross-platform mobile applications for mobile.
- Framework: react native was chosen for its ability to create native mobile application.

2. **User Interface**
   - Mobile app interface is designed as mobile application using React native, providing a native look and feel in both iOS and Android devices.
   - Component-Based UI

3. **Data Management**
   - Data storage: we plan to use AsyncStorage for local storage and integrate with cloud-based database
   - State management: context API will be used for managing App wide state, ensuring efficient data flow and consistent UI updates.
4. **Communication**
   - API integration to communicate with backend server, allow data exchange and interaction with server component.
5. **Error Handling**
   - Debugging: using react native's debugging tools and libraries to identify and fix the issue during development.
6. **Scalability and Future Plans:**
   - The architecture will be designed to scale, ensuring the app can handle increased user traffic and data load as it grows.
   - The app will be designed with modularity in mind, allowing easy integration of new features and enhancement in the future.

4. System Architecture

Main Parts

Budget Management: Budget Managment will take care of tracking money and budget settings. This will include storing the user's budget to allow them to quickly access and change their budget quickly. It will also let users keep an eye on their spending, set budgets, and see reports on their finances.

User Management: User Management will contain User sign-up and log-in information. This will also include security to make sure that their information is held safely.

How It Works

Handling Data: Users tell the app about their spending and earnings, either on their own or by linking their bank account.

Talking to Banks: MoneyFlow talks to banks to get the latest transactions, making sure users have the newest info for tracking and budgeting.

4.1. Subsystem Architecture

Budget Management Subsystem

Components:

Expense Tracker: Records and categorizes the user's expenses.
Budget Planner: Users set and adjust their financial goals.
Financial Reporting: Offers insights through reports.
Functionality: Detailed tracking and management of the user's finances. It integrates with external banking APIs.

User Management Subsystem
Components:

Authentication Module: Secures user access using a login.
Profile Management: Users can maintain their personal and financial information.
Functionality: Focuses on securing user data and personalizing the app experience.

5. Detailed System Design

5.1. Classification

1. Subsystem:
   - Budget management subsystem
   - User management subsystem
2. Module:
   - User Authentication module
   - Income and expenses tracking module
   - Reporting module
3. Class:

- User class
- Transaction class
- Budget class

4. Function/ method
   - calculateTotalIncome()
   - calcualteExpenses()
   - categorizeExpenses()
   - addExpenses()
   - addIncome()
5. Package
   - Budget_planning
   - User_authentication

5.2. Definition

Subsystem:

- Budget management subsystem: this subsystem serves as the central hub for all budget-related operations within the application. It encompasses some functionalities such as income tracking, expenses tracking, budget planning, and financial analysis.
- User management subsystem: responsible for handling user-related functionalities such as registration, authentication, profile management, and access control. This subsystem ensures that only authorized users can access and interact with the functionalities.

The module: within the budget app subsystem, there can be several modules:

- User authentication module: this module handles user login, registration, password management, and authentication processes.
- Income tracking modules: this module manages the tracking of various income sources, including salaries, etc.
- Expenses tracking module: handling the tracking of distinct types of expenses such as groceries, bills, entertainment, etc.
- Reporting module: this generates various reports based on user data, such as spending trends, budget performance, etc.

Class:

- User class: this class is for user information such as username, password, email, etc.
- Transaction class: this stands for individual income or expense transactions, including attributes like amount, category, date, etc.
- Budget class: this class defines the structure of a budget, including the allocated amount for each category.

5.3. Responsibilities

1. **Budget Management Subsystem:**

- Assist users in setting up budgets for different expense categories.
- Provide tools for monitoring budget adherence and analyzing spending patterns.
- Generate reports and visualizations to help users gain insights into their financial habits and make informed decisions.
- Handle interactions with external financial APIs or services, if applicable.

2. **User Management Subsystem:**
- Enable users to register accounts securely within the application.
- Authenticate users during login to ensure access control and security.
- Manage user profiles, allowing users to update personal information and preferences.
- Implement access control mechanisms to restrict unauthorized access to sensitive functionalities and data.
- Handle password management tasks such as reset and recovery.

3. **Income tracking module:**
- Record the various income for user.
- Categorized the income sources.
- Allow users to input income or integrate with external sources like bank accounts.
- Calculate total income over time.

4. **Expenses tracking module:**
- Categorize different types of expenses.
- Enable user to input expenses manually.
- Tracking spending habit and pattern over time
- Generate reports and visualization to help users understand their spending.

5. **Reporting module:**
- Generate various reports and summaries based on user data.
- Present insights into spending trends, income sources, budget performance
- Visualize data by using charts and pie graphs.

6. **User class**
- Representing user data: some user essential information: username, password, email.
- Have access control to certain part of the application.
- Provide functionalities for password reset, recovery, and ensuring the password is secure.
- Allow users to update their information.

7. **Transaction class**
- Representing individual financial transaction: amount, date, category, etc.
- Categorization and tagging provide methods to create new transactions, updating existing transactions.

8. **Budget class**
- Define structure of the budget

- Analyze the historical spending patterns to provide spending habits.
- Allow users to adjust their budget.

5.4. Constraints

1. User class: Must adhere to password security constraints such as minimum length and complexity requirements to ensure protect the user credentials.
2. Transaction class: must adhere to data integrity constraints to maintain consistency and accuracy.
3. Income tracking module: the module must be combined with data privacy regulation to ensure that the sensitive data information is securely stored.
4. Expenses tracking information: the app needs to link to the bank account, so it must handle data synchronization challenges such as data consistency to ensure accurate expenses tracking.
5. Reporting module: must efficiently aggregate and process a large volume of transaction data to generate reports.

5.5. Composition

1. User class:
   - Have some attributes such as username, password, email, firstName, lastName which represents the user identity and profile with a system.
   - It also includes some methods like user authentication, profile management, password reset, etc.
2. Transaction class:
   - Including some transaction amount, categories, date, description, and tags which capture the detail of the individual transaction.
   - Providing some method for creating new transactions, retrieving transactions, deleting transactions, etc.
3. Budget class:
   - This class will have some attributes representing budget categories, allocated amount, which defined the structure of the user budget.
   - Some methods: different expenses categories tracking budget, analyze the budget performance

5.6. Uses/Interactions

1. User class
   - Is used by user management subsystem during the authentication process to verify the user credentials and grant access to the system.
   - User class is also used by the authentication module, which interacts with user class to retrieve and update user information.
2. Transaction class
   - Expenses and income tracking module: the transaction class serves as backbone of this module.

- Reporting module: transaction data is captured by transaction class is utilized by the reporting module to generate various financial reports and insights.

3. Budget class
   - Used by reporting module to generate the report on budget adherence, saving goals and financial forecasts.
   - It uses the transaction class to retrieve the data transaction, which is then used to analyze the spending habit.

5.7. Resources

- External APIs: the budget app needs to interact with the APIs provided by the third-party services to link and access the bank account information.
- Data storage: linking banking account information needs to be securely stored in the app's database. This requires resources such as storage space and database processing.

5.8. Processing

1. User class:
   - For the user authentication, the user class will use cryptographic algorithm to hash and salt password for secure storage.
   - The user class undergoes state change during user registration where user record is created and during profile update.
   - Time/space complexity: Time complexity will be O(1) or O(logn) with respect to the number of users. The space complexity will be O(n)
2. Transaction class
   - Transaction recording: this provides methods to create new transaction records, which give the details as amount, category, date, and description.
   - State change occurs when a new transaction is created.
   - Time complexity will be O(1) and O(logn). Space complexity is O(n) with respect to the number of transactions stored.
   - The transaction will handle exceptional conditions such as invalid transaction.

5.9. Interface/Exports

1. User class:
   - createUser (username, email, password): create a new user record in the database
   - updateUserProfile (user, newData): update profile information
   - auhenticationUser (username, password): authentication a user based on the provide username and password.

- Interface will provide method for user registration, authentication, and profile management.

2. Transaction class:
   - CreateTransaction (user, amount, category, data, description): create new transaction record for specified user.
   - updateTransaction: update an existing transaction record.
   - DeleteTransaction: delete a transaction record
   - The interface will provide methods to for creating, updating, and deleting the transaction

3. Budget class:
   - createBudget: create a new budget plan.
   - updateBudget: update an existing budget plan

5.10. Detailed Subsystem Design

In the detailed subsystem design, we will focus on the budget management subsystem, which is the core component responsible for managing all budget related functionalities. This subsystem includes the budget class and its interaction with other component: transaction module and the user class:

1. Budget class:
   - Id: unique identifier
   - Categories: list of budget categories
   - Amount: list of budget amount for each category
   - Period: time period
   - Method: I have discussed the method in the interface part.
   - Behavior:
     The budget plan interacts with database to store and retrieve the budget data. It validates input data for budget creation and update, ensuring the budget data is positive and the user has not exceeded their income.

2. Interaction with transaction class
   - When a transaction is recorded, the transaction class checks if the transaction exceeds the valid amount.

3. Interact with user class:
   - Retrieve the user information from user class.

4. Resources managed:
   - The budget manages the budget plan data, including the categories, amount, and period.
   - The budget connects with the database to store and retrieve budget data

- Notification the user about the budget

6. Glossary

AsyncStorage: An asynchronous, key-value storage system integrated within React Native.
- stores and retrieves data locally on the user's device

Authentication: A security mechanism to verify user identities using login credentials.

Context API: A tool in React that makes it easier to share data, like user settings, across different parts of the app without complicated coding.

Interoperability: Money Flow's ability to work smoothly with different banks.

React Native: Allows the creation of native mobile applications for both Android and iOS platforms using JavaScript.

Waterfall Model: A step-by-step approach used in developing MoneyFlow, where each phase like planning, designing, building, testing, and deploying happens in order.

7. Bibliography