Finance Manager Dashboard

Project Proposal

University: California State University Fullerton
Course: CPSC 490
Members: Jacob Hellebrand, Logan Clampitt, Moksh Patel, Triniti Nguyen
Date: May 15th, 2025

1. Abstract

Personal finance management is a struggle that all individuals and businesses face. The Finance Manager dashboard seeks to alleviate this issue by providing an intuitive, web-based platform that allows users to track expenses, manage budgets, and analyze financial trends. The dashboard aims to implement a user-friendly user interface, a strong backend for handling financial information, and a secure database for storing critical user information. The dashboard will support role-based authentication, interactive visualizations, and AI-driven financial insights. We seek to push out improvements based on user feedback to improve the user experience. This product aims to enhance financial literacy and empower users to make informed decisions by offering real-time financial tracking, automated categorization, and personalized recommendations. This proposal outlines the project's architecture, development plan, metrics, UX design, and prototype, ensuring a structured and efficient approach to implementation.

2. Table of Contents

3. Introduction

3.1. Purpose

Managing money today is more challenging than ever for individuals and businesses. Keeping up with expenses, sticking to a budget, and understanding financial trends can be overwhelming without the right tools. This often results in bad financial decisions and a struggle to grasp basic financial concepts. Many individuals find themselves unprepared to handle unexpected expenses or save for future goals, while businesses frequently miss opportunities for growth or cost-saving due to inefficient financial tracking. Manual financial management is complex because it's time-consuming and prone to error. Without a consolidated view of finances, making informed decisions or predicting future trends is challenging. Additionally, the lack of customized financial advice based on personal or business financial data further complicates financial planning. To address these issues, an automated, comprehensive tool like the Finance Manager Dashboard becomes indispensable, helping to streamline financial processes and improve accuracy.

3.2. Objectives

The main aim of the Finance Manager Dashboard is to make managing money easier with a user-friendly website that helps track spending, manage budgets, and analyze financial data. We're focusing on creating a straightforward interface, setting up a strong backend system to manage complex data securely, and developing a secure database for storing personal information safely. This ensures that all financial activities are logged accurately and easily accessible. Our solution stands out by integrating AI tools that provide customized financial advice and predictive analytics, giving users foresight into their financial health. Role-based authentication will ensure that information is kept secure but accessible to authorized users, making our platform ideal for individual and corporate use. Moreover, integrating machine learning algorithms will enhance the platform's ability to learn from user data, offering increasingly refined insights.

The Finance Manager Dashboard is built for anyone who wants to manage their money better, whether it's individuals or small to medium-sized businesses looking for a dependable tool to keep track of their financial health. The dashboard will offer features like live updates on financial activities, automatic storing of expenses, and detailed reports and analytics. We'll focus on designing and developing the web-based dashboard, integrating AI for smarter data analysis, implementing strong security to protect user data, and conducting user testing to gather feedback and improve the product. Each iteration of the dashboard will incorporate more user feedback, ensuring that the tool evolves to meet the changing needs of its users effectively.

3.3. Scope

The Finance Manager Dashboard will make a big difference for its users by helping them make smarter financial decisions. It's designed to improve financial understanding through easy-to-read, real-time data visuals. For businesses, it will help them plan and allocate resources more efficiently. The main goal is to give users the tools and knowledge they need to be financially stable and grow. This dashboard tackles the tricky parts of managing money with a platform that's easy to use yet powerful. It will help individuals and businesses manage their finances better, improving financial health and greater economic security. By providing a comprehensive view of financial health and easy-to-use tools, the dashboard ensures that users can focus on strategic decisions rather than getting bogged down in financial details.

4. Project Plan

4.1. Milestones

The development of the Personal Finance Dashboard follows a structured milestone-based approach, starting with the Introduction and Abstract phase, where the project scope, objectives, and an initial abstract are defined. Next, the Project Plan is outlined, establishing a schedule, deliverables, and high-level milestones. In the Tracking Project Metrics phase, key performance indicators are identified to monitor progress against initial plans. The Requirement Engineering phase involves defining use cases, developing system models, and creating state and behavioral diagrams to structure the system's functionality. The UX/UI Design phase establishes the user experience framework and develops mockups to refine the interface. Finally, the Prototype and Testing phase includes building an interactive prototype, performing security and usability testing, and gathering user feedback to enhance the final product before deployment.

4.2. High-Level Goals

The Personal Finance Dashboard aims to provide a seamless and intuitive experience for financial tracking through a user-friendly interface that simplifies managing income, expenses, and budgets. Integrating AI-driven automation reduces the need for manual expense tracking, ensuring greater accuracy and efficiency. Strong data security measures, including encryption and secure authentication, will be implemented to comply with industry standards, protecting users' financial information. The platform will be designed for scalability and high performance,

allowing for future feature expansion and ensuring smooth functionality as user needs grow. Additionally, the dashboard will generate actionable financial insights, offering personalized reports and recommendations to help users make informed financial decisions and achieve their financial goals.

Key Focus Areas	High-level Goals
User-Friendly UI/UX Design	Develop an intuitive and responsive dashboard for financial management.
	Provide customizable views with interactive graphs and charts.
	Ensure accessibility compliance for users with disabilities.
Efficient Data Storage and Management	Store financial data in an optimized schema to minimize storage overhead.
	Implement data compression and archiving strategies to manage historical records efficiently.
	Ensure fast and efficient backup & recovery processes to prevent data loss.
Financial Planning	Allow users to set and track financial goals, such as savings.
	Provide budgeting tools that help users plan their spending more effectively.
	Offer forecasting features to predict future expenses based on past trends.
AI Chatbot	Automate expense categorization and anomaly detection to reduce manual tracking.
	Provide personalized financial recommendations based on spending patterns.
	Alert users to prevent overspending or identify unusual transactions.
Security and Compliance	Implement encryption for storing sensitive financial data.
	Enforce role-based access control and multi-factor authentication to prevent unauthorized access.
	Regularly conduct security audits and vulnerability testing.

4.3. Timeline

The project timeline is structured into six key phases, ensuring a systematic approach to development. The project begins in Week 5 with the Introduction and Abstract, where the problem statement, objectives, and project scope are defined, followed by creating an abstract and table of contents. In Weeks 6 and 7, the Project Plan is outlined, detailing the schedule, deliverables, and the technology stack required for implementation. Week 8 focuses on Tracking Project Metrics, identifying key performance indicators, and comparing the planned and actual progress. The Requirement Engineering phase, scheduled for Week 9, involves defining use cases, system models, state diagrams, and a comprehensive storyboard. In Weeks 11–14, the UI/UX design process starts with defining a structured UX framework and developing interactive mockups. Finally, the Prototype and Testing phase in Weeks 15–17 involves building an interactive prototype, conducting security and usability tests, and refining the system based on user feedback. This structured timeline ensures a progressive development cycle, integrating technical and user-focused improvements to build a robust and efficient financial dashboard.

Milestones	Tasks	Estimated Completion
1 - Introduction and Abstract		
1.1	Define problem, objective, and scope of the project	Week 5
1.2	Create the abstract	Week 5
1.3	Create a table of content	Week 5
2 - Define Project Plan		
2.1	Create a schedule and deliverables	Week 6, 7
2.2	Define the milestones and high-level goals	Week 6, 7
3 - Tracking Project Metrics		
3.1	Identify metrics to track project	Week 8
3.2	Illustrate planned vs. actual plan	Week 8
4 - Requirement Engineering		
4.1	Define use cases and scenarios	Week 9, 10
4.2	Develop system models and diagram	Week 9, 10
4.3	Create state and behavioral diagrams	Week 9, 10
4.4	Develop storyboard	Week 9, 10
5 - UX/UI Design		

5.2	Define UX framework and architecture	Week 11, 12
5.3	Develop UX mockups	Week 13, 14
6 - Prototype and Testing		
6.1	Develop interactive prototype	Week 15, 16
6.2	Conduct security, performance, and usability testing	Week 16, 17
6.3	Gather feedback and refine the prototype	Week 16, 17

5. Metrics

5.1. Project Tracking Metrics

To keep our project on track, we'll use GitHub to manage our code and track progress. We plan to organize tasks through GitHub Issues, making it easy to see what's done and still needs work. Our team will also track how often we commit code, merge changes, and resolve issues to keep things moving in the right direction. Since we're using VSCode, we'll regularly ensure our code is clean and efficient through peer reviews. Fixing bugs quickly is our top priority, so we'll monitor how long it takes to find and resolve them. Security is also a big focus, especially since we're dealing with financial data, so we'll include regular checks to keep everything safe. While we haven't fully committed to a development style yet, we plan on taking an agile-inspired approach, using short development cycles to adjust as needed. By tracking these key areas, we'll ensure steady progress and catch any roadblocks early. The project boards function in GitHub and will aid us in visualizing tasks to manage priorities better. The progress reports scheduled for each week will help detect possible delays and regular security audits that will help maintain financial data protection compliance. Measuring development velocity through time tracking for vital features and quick resolution of essential issues helps determine our project's efficiency. The gathered insights enable better workflow optimization and resource maximization to support seamless development throughout the project.

5.2. Planner Chart

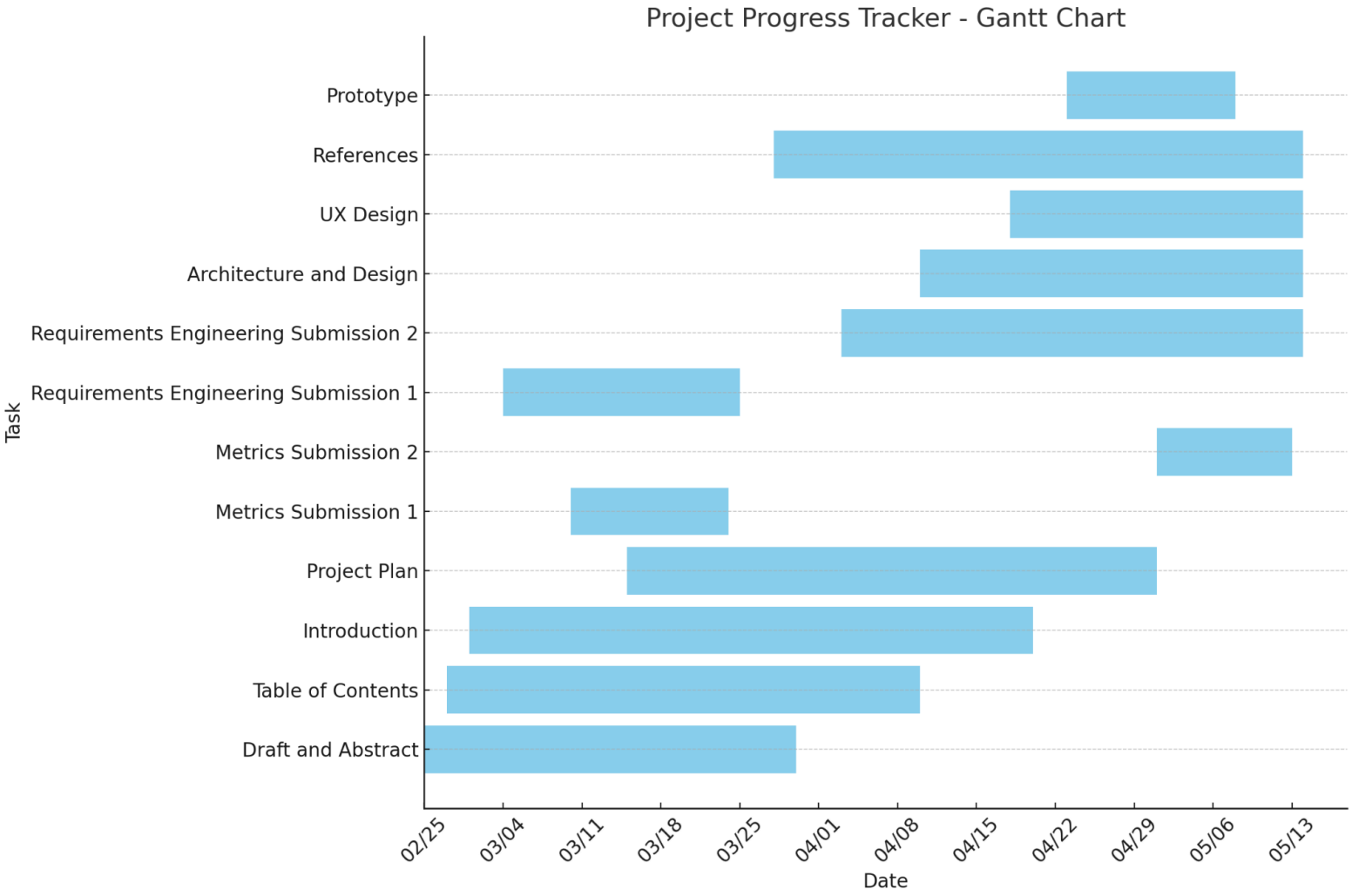
To ensure that development stays on the right track, we will compare our planned schedule with actual progress throughout the project. The Planner Chart will track significant tasks, including dashboard design, backend development, AI integration, and testing. Each task will have start and end dates, and we will regularly update the chart to represent each phase's current state. This will enable us to modify our strategy as necessary by assisting in detecting any delays or obstacles early. We will also monitor task completion times concerning early estimations to ensure that development resources are used efficiently. We can keep a consistent workflow and fulfill project deadlines by carefully monitoring developments and adjusting as needed.

Progress Tracker					
Task	**Assigned To**	**Start Date**	**End Date**	**Status**	**Notes**
1. Draft and Abstract	Jacob	2/25/2025	3/30/2025	Complete	Defined core vision and summarized project goals.
2. Table of Contents	Jacob and Trinity	2/27/2025	4/10/2025	Complete	Created structure for all major sections.
3. Introduction	Moksh	3/1/2025	4/20/2025	Complete	Background, purpose, objectives, and scope covered
4. Project Plan	Triniti	3/15/2025	5/1/2025	Complete	Outlined milestones, timeline, and focus areas.
5. Metrics Submission 1	Logan	3/10/2025	3/24/2025	Complete	Created initial metrics system and planning chart.
6. Metrics Submission 2	Logan	5/1/2025	5/13/2025	Complete	Updated metrics for weeks 10-17.
7. Requirements Engineering Submission 1	Everyone	3/4/2025	3/25/2025	Complete	Documented use-cases and scenarios.
8. Requirements Engineering Submission 2	Everyone	4/3/2025	5/14/2025	Complete	Added updated diagrams and improved descriptions.
9. Architecture and Design	Everyone	4/10/2025	5/14/2025	Complete	Drafted backend design, database planning.
10. UX Design	Everyone	4/18/2025	5/14/2025	Complete	Completed wireframes and UI consistency decisions.
11. References	Everyone	3/28/2025	5/14/2025	Complete	Compiled and formatted all sources used.
12. Prototype	Everyone	4/23/2025	5/8/2025	Complete	Building layout with navigation logic and mockup UI.

5.3. Graphs and Charts

Different graphs and charts will be presented to demonstrate the results and progress achieved from the project. Gantt charts depicting deadlines and interdependencies within the project will be utilized to ensure that key milestones are achieved. Burndown charts, which illustrate the remaining work over a designated period, enable us to track the sprint's advancement and identify potential bottlenecks. Velocity charts will reflect the team's productivity by depicting task completion time. Issue tracking reports will also be used to measure the rate of bug resolution and ensure that the other areas of development are on schedule. Integrated analytics will also be used to analyze user activity and improve usability and functions because we will also track user engagement through the Finance Manager's Dashboard. This aids in enhancing the overall experience. Visual aids will help maintain development efficiency by providing vital information.

Task/Story	Hrs Est.	Wk 5	Wk 6	Wk 7	Wk 8	Wk 9	Wk 10	Wk 11	Wk 12	Wk 13	Wk 14	Wk 15	Wk 16	Act. Hrs.
Draft and Abstract	2		1	1										2
Table of Contents	1		1											2
Introduction	2				2									2
Project Plan	3		1	1	1									3
Metrics Submission 1	3		1	1										2
Metrics Submission 2	2										1	1		2
Requirements Engineering Submission 1	4		1	2	1									4
Requirements Engineering Submission 2	4								1	1	2			4
Architecture and Design	6							2	1	2	1	2		8
UX Design	8							1	2	2	1	2		8
Prototype	6									1	2	2	2	7
References	2												2	2
Remaining Effort	43	43	40	35	31	31	31	28	24	18	12	5	0	
Ideal Burndown	43	40	35	33	30	28	25	24	21	18	10	5	0	


Project Progress Tracker - Gantt Chart
Prototype
References
UX Design
Architecture and Design
Requirements Engineering Submission 2
Requirements Engineering Submission 1
Metrics Submission 2
Metrics Submission 1
Project Plan
Introduction
Table of Contents
Draft and Abstract
Task
02/25
03/04
03/11
03/18
03/25
04/01
04/08
04/15
04/22
04/29
05/06
05/13
Date

6. Requirement Engineering

6.1. Use-Cases

Use Case 1: User views overall account balances

Description	Process of how users will view their account balances.
Actors	Website Users
Preconditions	User is logged into the dashboard
Triggers	The User selects the "Accounts Overview" tab.
Main Flow	1. The dashboard page loads, displaying an overview of the user's financial accounts. 2. Each account type (checking, savings, investments) is shown as a clickable card or tab with a summary balance. 3. The user clicks on one of the account cards 4. The system fetches detailed account information from the financial database. 5. The system displays the selected account's recent transactions, including dates, descriptions, and amounts. 6. The user can scroll through transactions or apply filters such as date range, transaction type, or amount. 7. If needed, the user clicks a **View More** button to load additional transactions.
Postconditions	The user successfully views updated account balances.
Exceptions:	● If there is a connection issue, the system displays an error message. ● If the user has yet to enter any financial information, prompt them with the ability to begin their dashboard.

Use Case 2: User tracks monthly expenses

Description	Process of how users will be able to use the dashboard to keep track of expenses
Actors	Website Users
Preconditions	User is logged in and has financial data stored in the system.
Triggers	The User selects the "Expense Tracker" tab.

Main Flow	1. User enters the expense tracker tab 2. User clicks the track current expenses card 3. The User selects a time period (weekly, monthly, yearly) 4. The system retrieves categorized expenses (such as rent, utilities, groceries). 5. The user can select how they want the data displayed by clicking buttons. 6. The graph/chart the user selects will appear on the current tab 7. The User reviews trends and identifies high-cost areas.
Postconditions	The user successfully tracks expenses with detailed insights.
Exceptions:	If no data is found, the system notifies the user, and prompts them to enter financial information.

Use Case 3: User using AI Chatbot for financial insights

Description	Process of how users will be able to use the dashboard to keep track of expenses
Actors	Website Users, AI Chatbot
Preconditions	● User is logged into their account. ● User has submitted financial data (income, expenses, budget, financial goals). ● AI Chatbot has access to analyze financial trends and provide recommendations
Triggers	● The user interacts with the AI chatbot via the website. ● The user asks for financial insights or suggestions. ● The AI chatbot automatically suggests improvements based on financial trends.
Main Flow	1. The user initiates a conversation with the AI chatbot. 2. The chatbot retrieves the user's financial data (e.g., spending patterns, income, budget compliance). 3. The chatbot analyzes financial trends and identifies key insights. 4. The chatbot generates personalized recommendations 5. The user can ask follow-up questions for more details or alternative suggestions.

	6. The chatbot refines its response and provides additional insights or simulations based on user input.
Postconditions	● The user receives actionable financial insights and recommendations. ● The system logs chatbot interactions for future learning and improvement.
Exceptions:	● If financial data is incomplete, the chatbot prompts the user to input missing information. ● If the chatbot is unable to provide specific advice, it directs the user to financial planning resources or human support.

Use Case 4: User creates a monthly budget

Description	Users can create a monthly budget by setting spending limits for different categories based on their expected income and expenses. The system provides visual progress indicators and real-time alerts to help users track their spending and adjust their budget as needed.
Actors	Website Users
Preconditions	● The user is logged into their account on the dashboard. ● The user has at least one recorded income source or past financial data in the system.
Triggers	● The user selects the “Budget Management” tab from the dashboard. ● The user chooses to create or update their budget for the upcoming month.
Main Flow	1. The system retrieves the user’s financial data, including past income, expenses, and categorized spending. 2. The user sets spending limits for the upcoming month for different budget categories (e.g., groceries, rent, utilities, entertainment). 3. The system calculates the total budget allocation and checks for inconsistencies (e.g., exceeding total income). 4. The system visually represents spending progress using charts, bars, and trend indicators.

	5. As the user spends money, the system updates spending categories in real-time and compares them against budget limits. 6. The system sends alerts and notifications when the user approaches or exceeds their budget in any category. 7. The user can manually adjust budget limits, reallocate funds between categories, or modify goals based on spending patterns. 8. At the end of the month, the system generates a budget summary report, highlighting spending trends and suggesting improvements.
Postconditions	● The user successfully creates and manages their budget. ● The system continuously updates and tracks spending data. ● Users receive timely notifications and insights into their financial habits.
Exceptions:	No Financial History Available: ● If the user has no prior spending records, the system suggests a default budget based on common income and expense patterns. Unrealistic Budget Limits: ● If a user sets a budget limit that is too low or too high based on their income, the system provides recommendations to ensure realistic financial planning.

Use Case 5: User tracks monthly bills and subscriptions

Description	Users can add, track, and manage recurring bills and subscriptions, receive payment reminders, and mark payments as completed.
Actors	Website Users
Preconditions	● The user is logged into their account. ● The user has entered recurring bill details into the system.
Triggers	The user selects the "Bill Management" tab from the dashboard.

Main Flow	1. The system retrieves the user's stored bills and subscription payment details. 2. The user can add new recurring payments (e.g., rent, utilities, Netflix, phone bill). 3. The system automatically updates payment due dates based on the billing cycle. 4. The system sends reminders via email, SMS, or app notifications before due dates. 5. The user can mark a bill as paid manually or enable auto-pay for automatic payment processing. 6. The system records the payment history and updates the user's financial records accordingly.
Postconditions	● The user successfully sets a monthly budget with clear spending limits. ● The system continuously updates and tracks spending data throughout the month. ● Users receive timely notifications and insights into their financial habits.
Exceptions:	No Financial History Available: ● If the user has no prior spending records, the system suggests a default budget template based on common income and expense patterns. Unrealistic Budget Limits: ● If a user sets a budget limit that is too low or too high relative to their income, the system provides recommendations for realistic financial planning.

Use Case 6: User adds financial goals

Description	Users can add financial goals
Actors	Website Users
Preconditions	The user is logged into their account.
Triggers	The user selects the "Financial Goals" tab from the dashboard.

Main Flow	1. The user sets a financial goal (e.g., saving $5,000 for a vacation or paying off a $10,000 loan). 2. The system tracks the user's income, expenses, and savings to monitor progress. 3. The system provides visual indicators (e.g., progress bars, graphs) showing goal achievement status. 4. The system offers personalized suggestions to accelerate goal completion (e.g., reducing non-essential spending, increasing savings contributions). 5. The user can modify, update, or remove the financial goal at any time.
Postconditions	● The user successfully defines and tracks financial goals. ● The system continuously updates goal progress based on real-time financial data. ● The user receives insights on how to reach goals faster.
Exceptions:	No Goal Amount Specified: ● If the user does not enter a specific goal amount, the system suggests realistic goal options based on their income and expenses. Unrealistic Goal: ● If the goal is significantly beyond the user's financial capacity, the system warns the user and suggests a more achievable target.

Use Case 7: User registers for an account

Description	Allow new users to create an account and existing users to log in securely to the personal finance dashboard
Actors	Website users
Preconditions	● User need an internet connection. ● User has access to the registration or login page.
Triggers	● New user decides to create an account. ● User need to click on the "sign up" button on the . ● User need to be on the sign-up page.

Main Flow	1. The user visits the registration or login page. 2. The user enters their credentials (email, password). 3. The system authenticates the user based on the credentials. 4. If successful, the user is redirected to their personal finance dashboard. 5. If registration is selected, the user provides the required information (e.g., email, password) and clicks "Create Account." 6. The system confirms registration via email or SMS and directs the user to the login page.
Postconditions	● The user successfully created an account and can now log in. ● The system securely stores the user's credentials. ● The user receives a confirmation email or SMS with account verification details. ● The system updates the database with the new user's information.
Exceptions:	Invalid Email Format: ● If the user enters an improperly formatted email, the system prompts them to enter a valid email address. Email Already Registered: ● If the provided email is linked to an existing account, the system notifies the user and suggests logging in instead. Weak Password: ● If the password does not meet security requirements (e.g., minimum length, special characters, numbers), the system displays an error message with password guidelines.

Use Case 8: User logs in to the account

Description	Allow existing users to log in securely to the personal finance dashboard
Actors	Website users
Preconditions	● Users need an internet connection. ● User has access to the login page.
Triggers	● New user decides to create an account.

	● An existing user chooses to log in to their account.
Main Flow	1. The user visits the registration or login page. 2. The user enters their credentials (email, password). 3. The system authenticates the user based on the credentials. 4. If successful, the user is redirected to their personal finance dashboard. 5. If registration is selected, the user provides the required information (e.g., email, password) and clicks "Log In." 6. The system confirms registration via email or SMS and directs the user to the login page.
Postconditions	● The user successfully logs into their account and gains access to the personal finance dashboard. ● The system securely stores the user login session. ● The user can now perform financial activities, such as viewing transactions, managing budgets, and tracking expenses.
Exceptions:	Incorrect credentials: ● User will need to re-enter the correct credentials to log in ● User will receive an invalid statement stating that the information is incorrect Unverified account: ● User will receive a message stating that the user credentials are not found

Use Case 9: User reset their password

Description	Existing users with an account forgot their password and want to reset it.
Actors	Website user
Preconditions	● Users need to have an account. ● User need to click on “forgot password” from the login in page ● User need to provide their email address
Triggers	The user selects the "Forgot Password" option on the login page.

Main Flow	1. The user clicks on the "Forgot Password" link. 2. The system prompts the user to enter their registered email address. 3. The user enters their email and submits the request. 4. The system verifies if the email exists in the database and sends a password reset link if it is valid. 5. The user receives an email with a password reset link and clicks on it. 6. The user enters a new password, confirms it, and submits the form to reset their password.
Postconditions	The user successfully resets their password and can log in with the new credentials.
Exceptions:	Invalid Email Address: ● If the email is not found in the system, the user receives an error message stating that the account does not exist. Expired Reset Link: ● If the user clicks on an expired password reset link, the system displays a message asking them to request a new reset email. Weak Password: ● If the new password does not meet security requirements, the system prompts the user to enter a stronger password. Technical Issues: ● If the system fails to send the reset email due to server issues, the user is notified and asked to try again later.

Use Case 10: User receives email notification for due dates

Description	The user will receive an email notification about an upcoming due date.
Actors	Website user
Preconditions	● User is logged into their account ● User is on dashboard ● User has added recurring bills or subscriptions with due dates ● System has scheduled notifications for upcoming payments

Triggers	● System identifies that a bill or subscription is approaching its due date ● User has opted in to receive notification via email
Main Flow	1. The user logs into the Personal Finance Dashboard and adds their recurring bills or subscriptions, including the due dates. 2. The system continuously tracks the due dates of the user's bills. 3. A bill or subscription is identified to be 3 days before its due date. 4. The system generates an email notification detailing the bill's name, amount, due date, and payment options. 5. The email is sent to the user's registered email address. 6. The user receives the email, which prompts them to make the payment before the due date. 7. The user can take action directly from the email, such as linking to the payment portal or marking the payment as complete.
Postconditions	● The user receives a timely email notification about the upcoming payment due date. ● The email notification provides necessary payment details and reminders. ● The user may make the payment or take action based on the reminder.
Exceptions:	Invalid email address: ● The user will not receive the notification if the email address is incorrect or not registered. User opted out of email notifications: ● If users have disabled email notifications in their settings, they will not receive the email. ● If no recurring bills or subscriptions are added by the user, the system will not trigger any notifications.

Use Case 11: User categorizes tags and transactions

Description	Process of how users can manually or automatically categorize their transactions for better financial organization.
Actors	Website Users, Finance Manager System
Preconditions	● The user is logged into their account.

	● The system has access to transaction data from bank feeds or manual inputs.
Triggers	● The user reviews recent transactions. ● The system auto-categorizes transactions based on merchant details and past behavior. ● The user manually edits or reassigns a category or tag
Main Flow	1. The user navigates to the “Transactions” section. 2. The system auto-categorizes transactions (e.g., groceries, utilities, entertainment). 3. The user reviews and modifies categories if necessary. 4. The user adds custom tags for better tracking (e.g., “Work Expense” or “Vacation”). 5. The system updates and saves categorized transactions for financial reports.
Postconditions	● Transactions are correctly categorized and labeled. ● The system provides more accurate insights based on spending categories
Exceptions:	● If a transaction lacks details, the system prompts user to manually categorize it.

Use Case 12: User exports financial data

Description	Process of how users can download financial data for external analysis or tax purposes.
Actors	Website Users, Finance Manager System
Preconditions	● The user is logged into their account. ● The system has stored financial transaction data.
Triggers	● The user navigates to the financial reports section. ● The user selects an export option (CSV, PDF, or Excel).
Main Flow	1. The user selects the “Export Data” option. 2. The system provides file format options (CSV, PDF, Excel). 3. The user chooses the desired format and selects a date range. 4. The system generates the financial report and prepares it for download. 5. The user downloads and saves the file.

Postconditions	● The user successfully obtains a copy of their financial data. ● The exported file is formatted for use in tax filings or external tools (e.g., Excel, QuickBooks).
Exceptions:	● If there is no data available for the selected date range, the system notifies the user and suggests a different range

Use Case 13: User sets up automated savings transfers

Description	Process of how users can automate savings contributions based on income or spending habits.
Actors	Website Users, Finance Manager System, Linked Bank Account
Preconditions	● The user is logged into their account. ● The user has linked a bank account for fund transfers. ● The system has access to income and expense data.
Triggers	● The user navigates to the “Savings Goal” or “Automated Transfers” section. ● The user sets a rule for automatic savings contributions
Main Flow	1. The user selects the “Automate Savings” option. 2. The system prompts the user to set transfer rules (e.g., fixed amount per month, percentage of income, round-up spare change). 3. The user selects a destination account (e.g., savings account, emergency fund). 4. The system confirms the automation settings and schedules transfers accordingly. 5. The system executes transfers based on the set rules and logs them in transaction history.
Postconditions	● The system automatically moves funds based on user-defined rules. ● The user can track automated transfers in financial reports.
Exceptions:	● If the linked bank account has insufficient funds, the system notifies the user and skips the transfer. ● If the bank connection is lost, the system alerts the user to reauthorize the account.

Use Case 14: User sets financial reminders

Description	Users can set custom reminders for financial tasks such as making a loan payment, reviewing budget, or investing.
Actors	Website Users
Preconditions	● User is logged into their account. ● User has access to the Reminders feature from the dashboard.
Triggers	● The user navigates to the “Reminders” section and sets a new reminder.
Main Flow	● The user selects “Add Reminder” from the Reminders section. ● The system prompts the user to enter details: title, date/time, frequency, and optional notes. ● The user configures how they want to be notified (email, in-app, or SMS). ● The system saves the reminder and schedules a notification based on user preferences. ● On the due date/time, the system sends a notification to the user.
Postconditions	● The user is reminded of the scheduled financial task in a timely manner. ● The system logs the reminder for tracking purposes.
Exceptions:	● If the reminder fails to save due to a system issue, the user is prompted to retry. ● If a past date is selected, the system prompts the user to choose a valid future date.

Use Case 15: User links a new bank or credit account

Description	Users can securely link new financial institutions to their dashboard for live updates.

Actors	Website Users, External Financial Institutions
Preconditions	● The user is logged into their account. ● The system integrates with financial data APIs (e.g., Plaid)
Triggers	● The user selects “Link Account” from the Accounts or Settings tab.
Main Flow	● The user clicks “Link a New Account.” ● The system prompts the user to select their financial institution. ● The user is redirected to a secure authentication flow provided by the financial API. ● Upon successful authentication, the account is linked. ● The system imports account balances and transaction history.
Postconditions	● The user's new financial account is added to their dashboard. ● The system begins syncing data in real-time.
Exceptions:	● If authentication fails, the system displays an error and allows retry. ● If the API is down, the system notifies the user and suggests trying later.

Use Case 16: User customized dashboard layout

Description	Users can personalize their dashboard layout by rearranging widgets or hiding sections.
Actors	Website Users
Preconditions	● User is logged into the dashboard.
Triggers	● The user clicks “Customize Layout” on the main dashboard.
Main Flow	● The user selects or drags widgets (e.g., expense chart, recent transactions, budget status) to rearrange their layout. ● The user can enable/disable sections like AI insights or financial goals. ● The system saves the configuration to the user's profile.

	• On next login, the customized layout is applied automatically.
Postconditions	• The dashboard reflects the user's preferred layout and only shows relevant information.
Exceptions:	• If customization fails to save, the user is notified and prompted to retry. • If a widget cannot be displayed due to missing data, the system shows a placeholder with an explanation.

Use Case 17: User enables two-factor authentication (2FA)

Description	Users can add an extra layer of security to their account using 2FA.
Actors	Website Users
Preconditions	• User is logged into their account.
Triggers	• The user navigates to "Account Settings" and selects "Enable Two-Factor Authentication."
Main Flow	• The system prompts the user to enter their phone number or link an authenticator app. • The user selects their preferred 2FA method. • The system sends a verification code to the user. • The user enters the code to confirm. • 2FA is activated and required on the next login.
Postconditions	• The user's account is protected with two-factor authentication. • The system updates the user's security settings.
Exceptions:	• If the verification code is incorrect, the user is prompted to try again. • If 2FA setup fails due to a server error, the system notifies the user.

Use Case 18: User Asks AI Chatbot to Adjust a Current Finance Goal

Description	Users can update their financial goals by informing the AI chatbot of changes in their monthly budget or spending habits. The chatbot responds with a recalculated, personalized goal.

Actors	Website Users
Preconditions	● User is logged into their account. ● At least one finance goal has already been created.
Triggers	● The user accesses the chatbot and states a change in financial circumstances, such as: ● *"I now save $300 more each month"* or *"I've cut back on dining out and want to update my savings goal."*
Main Flow	● The system (AI chatbot) identifies that the user wants to adjust a finance goal. ● The chatbot requests details on the budget or spending habit changes, if not already specified. ● The user provides updated information (e.g., new monthly savings amount, reduced expenses, income change). ● The system analyzes the new inputs and recalculates the finance goal based on the user's preferences and updated budget. ● The system presents the user with a revised goal, including a breakdown of how the changes impact the timeline or amount. ● The user confirms whether to apply the new goal. ● The system updates the finance goal and stores the changes.
Postconditions	● The user's finance goal is updated to reflect their new financial situation. ● The system logs the interaction and updates budgeting projections.
Exceptions:	● If the user's input is too vague or ambiguous, the chatbot prompts for clarification. ● If the recalculation process fails due to a server or logic error, the system notifies the user and does not update the goal. ● If the new budget results in an unachievable goal (e.g., unrealistic savings rate), the chatbot provides suggestions or alternatives.

Use Case 19: User Deletes Account

Description	Users can permanently delete their account, which triggers the removal of their personal data, financial records, and related content from the system and its database (PostgreSQL).
Actors	Website Users
Preconditions	● User is logged into their account. ● User has verified their identity (e.g., via password or 2FA).
Triggers	● The user navigates to “Account Settings” and selects “Delete My Account.”
Main Flow	● The system displays a warning about permanent deletion, including what data will be lost. ● The user confirms the deletion request. ● The system validates the user’s identity (e.g., by prompting for password or a 2FA code). ● Upon successful validation, the system performs the following backend operations: ● Deletes user’s personal data from the users table in PostgreSQL. ● Deletes associated financial goals from the goals table. ● Deletes transaction history from the transactions table. ● Removes any AI interaction logs or stored chatbot sessions from related tables. ● Revokes any active sessions and API tokens tied to the user. ● The system confirms the deletion was successful and logs the event for auditing (excluding deleted user identity). ● The user is logged out and redirected to a goodbye or confirmation page.
Postconditions	● The user’s data is permanently removed from all relevant PostgreSQL tables. ● System no longer stores identifiable information tied to the user. ● The user is no longer able to log in.
Exceptions:	● Validation was failed initially ● The user decides to click no on the confirmation

Use Case 20: User Shares a Budget Report with a Financial Advisor

Description	Users can export and securely share their monthly budget reports with an external financial advisor via email or a temporary shareable link.
Actors	Website Users, External Financial Advisor
Preconditions	● The user is logged into their account. ● The user has generated at least one monthly budget report.
Triggers	● The user clicks “Share Report” from the Budget Summary page.
Main Flow	● The user navigates to the Budget Summary section. ● The user selects the report they wish to share. ● The system provides two options: enter an email address or generate a temporary secure link. ● The user chooses an option and submits. ● If email is selected, the report is sent as a secure PDF attachment. ● If a link is generated, it expires after a predefined time (e.g., 72 hours). ● The system logs the sharing activity for security and audit purposes.
Postconditions	● The report is shared successfully through the chosen method. ● The advisor can view the report without accessing other user data.
Exceptions:	● If the email address is invalid, the user receives an error message. ● If the link expires before it is accessed, the system prompts the user to generate a new one. ● If report generation fails, the user is informed and prompted to retry.

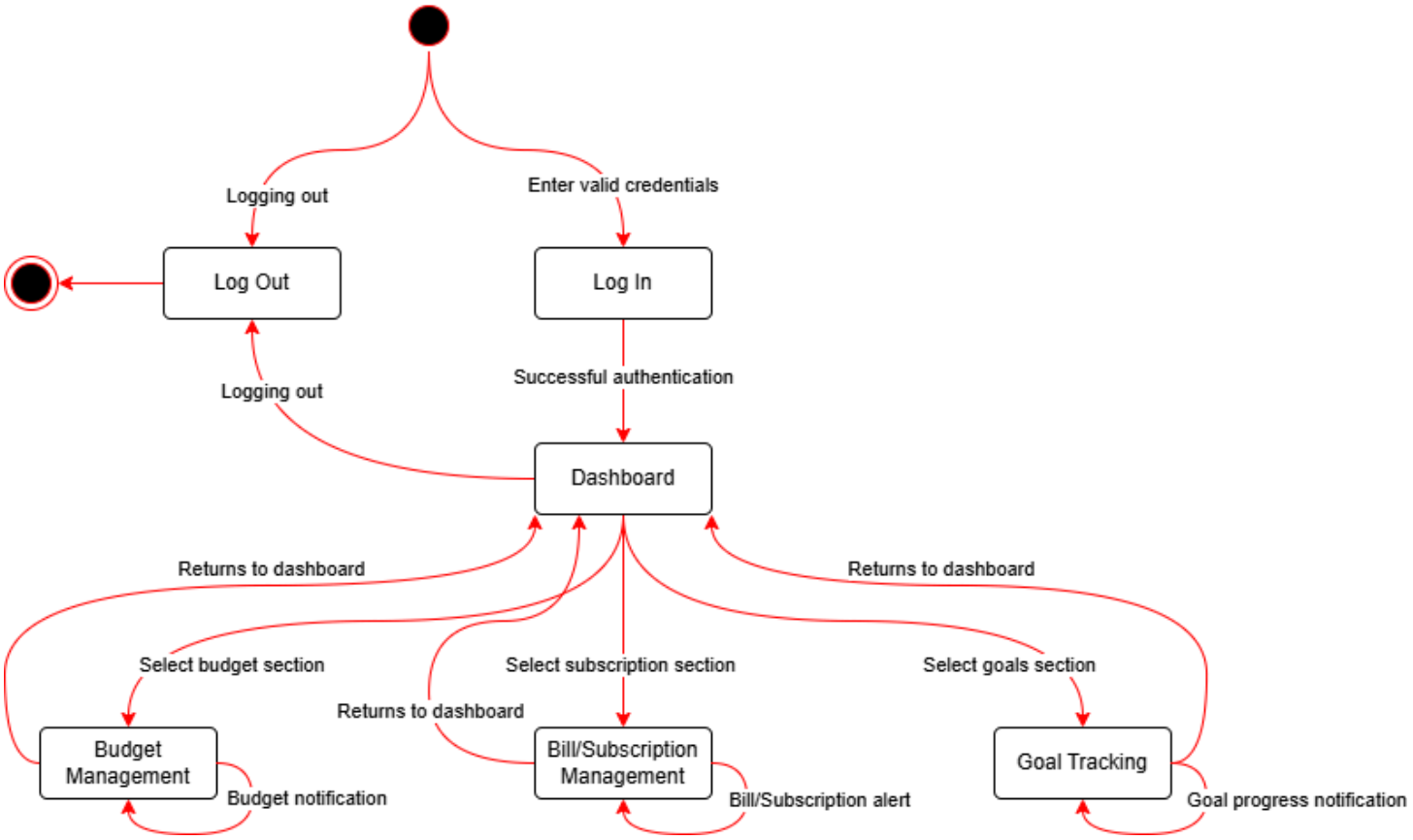
Use Case 21: User enables Dark Mode for Dashboard

Description	Users can toggle between light and dark mode for accessibility and personal preference.
Actors	Website Users
Preconditions	● The user is logged into their account.

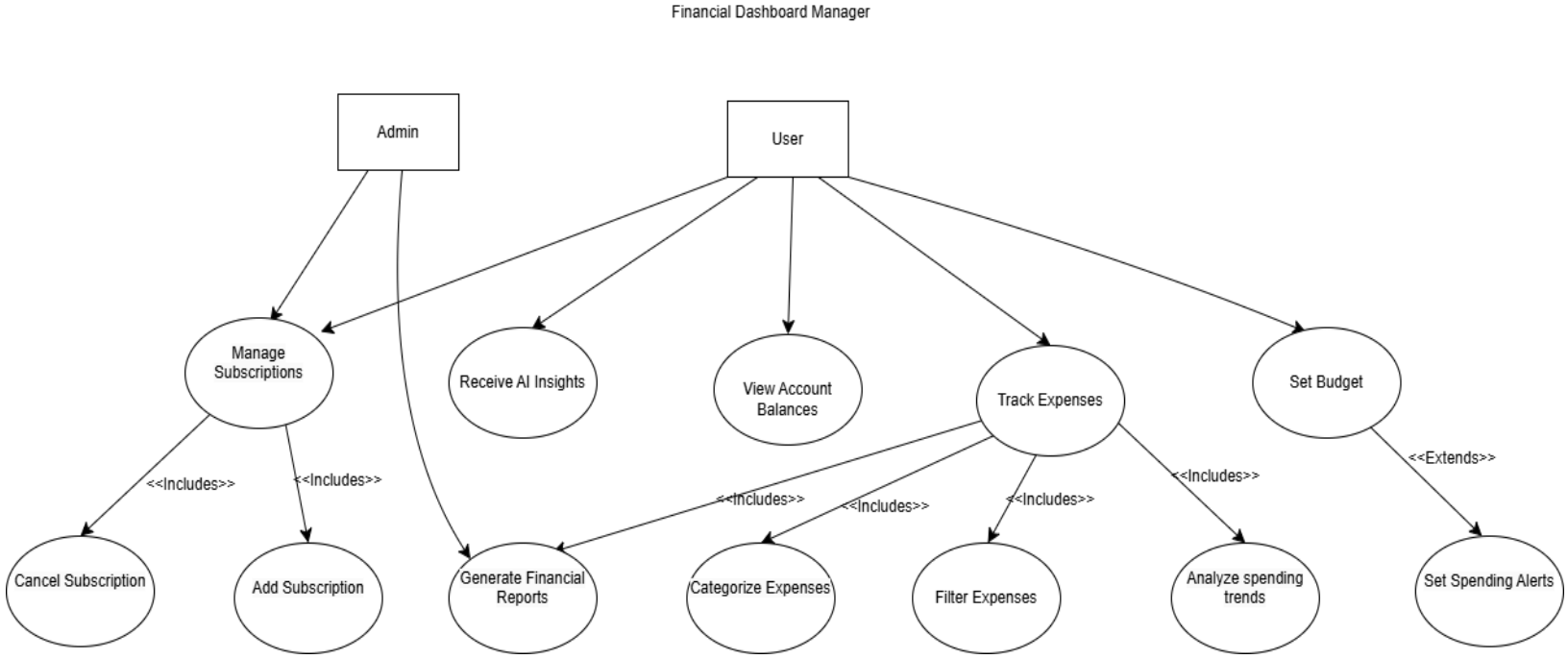
Triggers	● The user selects “Dark Mode” from display settings or clicks a toggle icon.
Main Flow	● The user clicks the toggle for Dark Mode in the settings menu. ● The system applies the dark color theme across the dashboard. ● The user confirms satisfaction or toggles back to light mode. ● The system stores the user’s preference in their profile settings. ● On the next login, the dashboard loads using the saved theme.
Postconditions	● The dashboard is displayed in the preferred theme (light/dark). ● The preference is applied persistently for the account.
Exceptions:	● If theme change fails to save, the system notifies the user. ● If a theme causes readability issues, a reset button is provided.

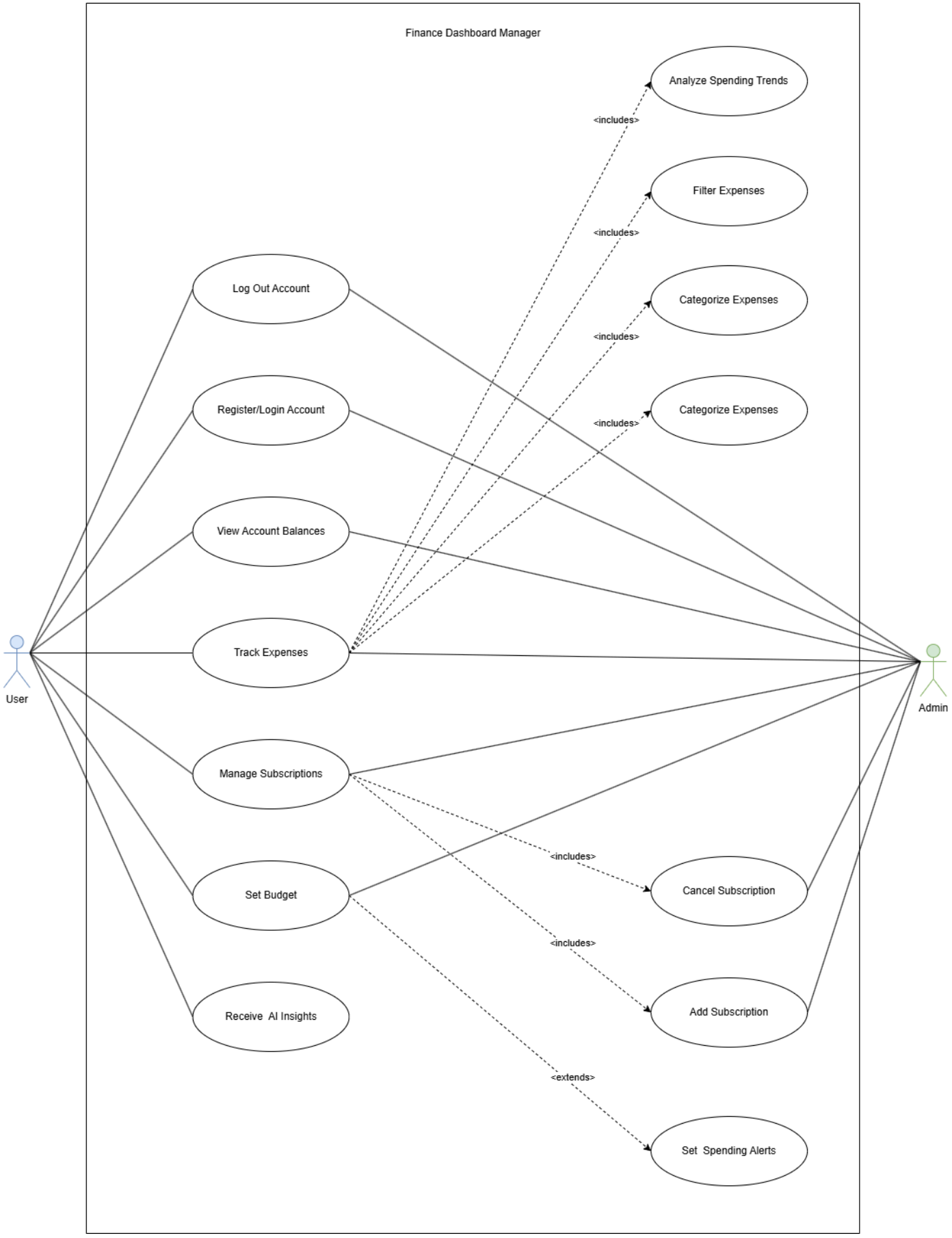
6.2. Modeling

6.2.1. State Diagram

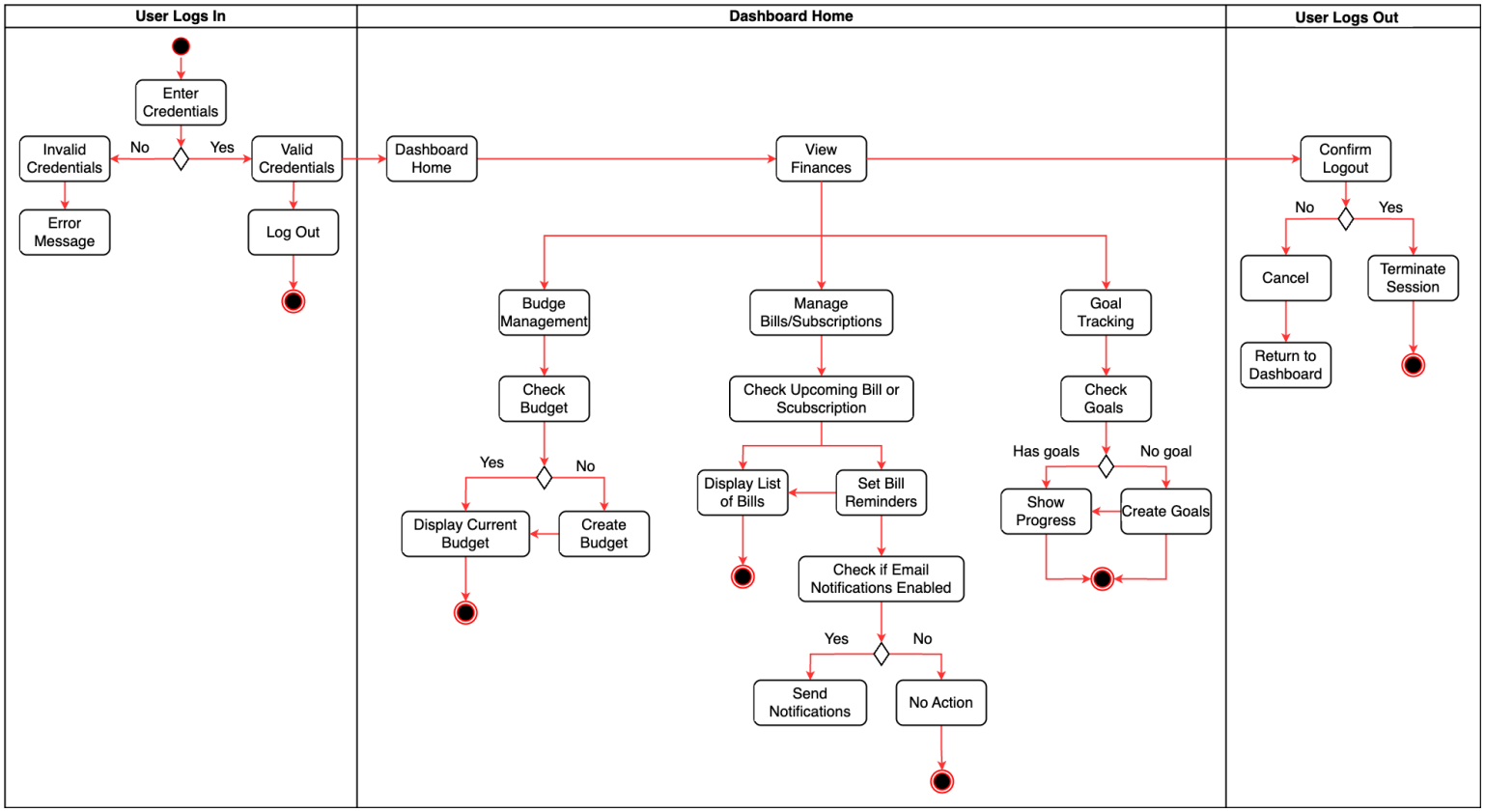


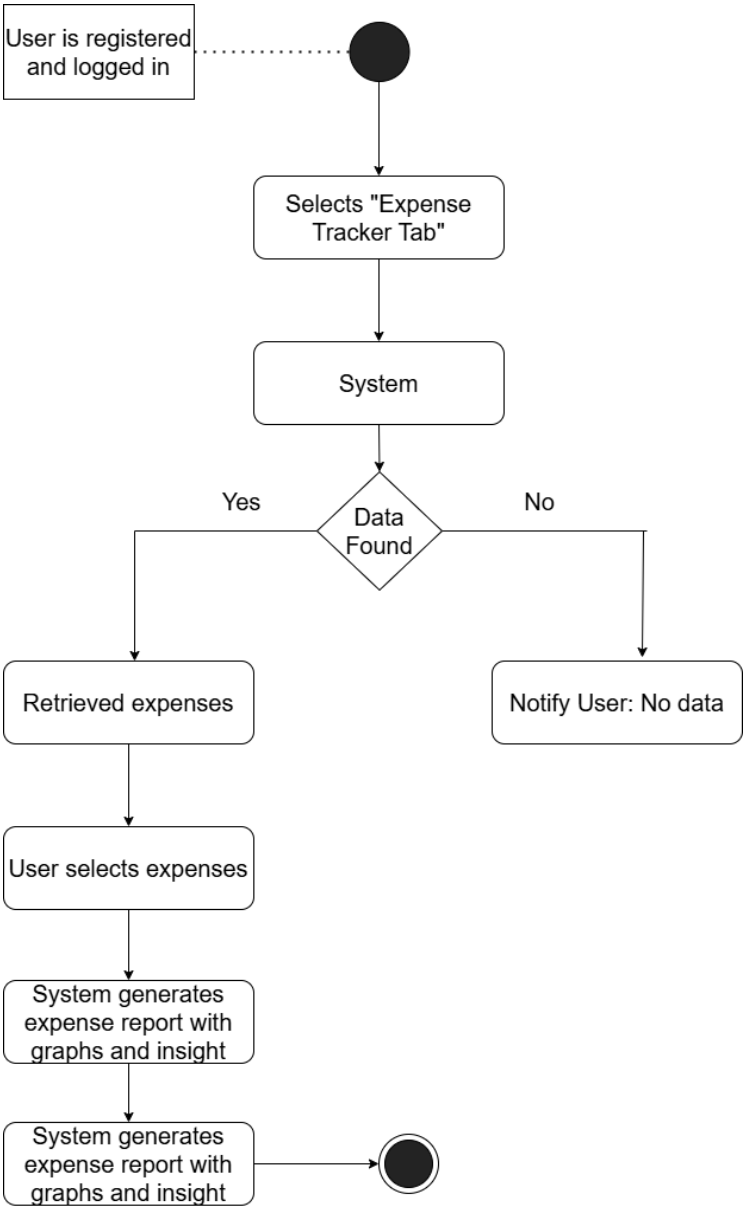
6.2.2. Use-Case Diagram



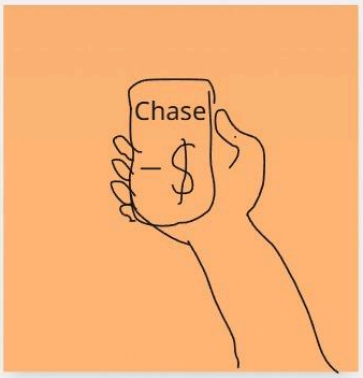

Finance Dashboard Manager
Analyze Spending Trends
Filter Expenses
Categorize Expenses
Categorize Expenses
Log Out Account
Register/Login Account
View Account Balances
Track Expenses
Manage Subscriptions
Set Budget
Receive AI Insights
Cancel Subscription
Add Subscription
Set Spending Alerts
<includes>
<includes>
<includes>
<includes>
<includes>
<includes>
<extends>
User
Admin

6.2.3. Activity Diagram (Lucid Software, n.d.)

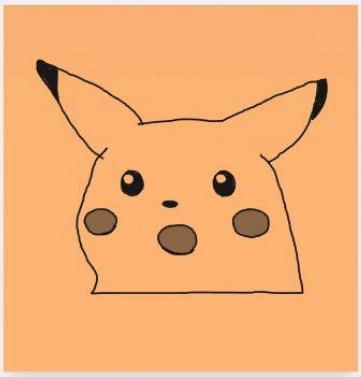



User is registered and logged in
Selects "Expense Tracker Tab"
System
Data Found
Yes
No
Retrieved expenses
Notify User: No data
User selects expenses
System generates expense report with graphs and insight
System generates expense report with graphs and insight

6.2.4. Storyboard (DevSquad, n.d.)

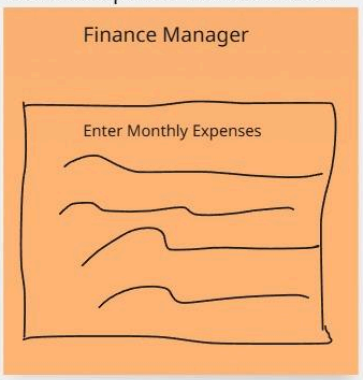


Actor checks bank account and finds their suboptimal financial status



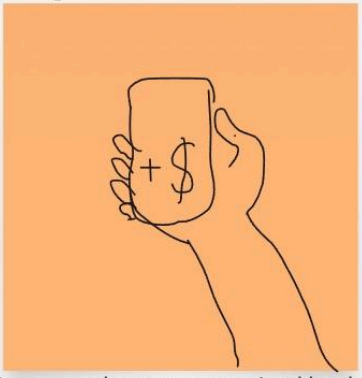
Actor is shocked because they have no idea how this happened

Actor googles a personal Finance manager



Actor enters their expenses for their desired time frame as well as income.

Actor receives a detailed report of high-cost expenses and how to improve



Actor now has a more optimal bank balance after applying their new finance strategy

7. Architecture and Design

We wanted to create a dashboard that genuinely helps people stay on top of their finances without having to deal with clunky spreadsheets or overwhelming apps. To make that happen, we needed a solid, flexible system that could grow with user needs while staying simple and secure. In this section, I'll walk you through the architecture and design choices we made, why they work for our project, and how everything fits together. We'll also show diagrams to help visualize the structure.

7.1 Architecture Pattern: Microservices

Instead of stuffing all the features into one massive application, we decided to break everything down into smaller pieces that work independently but talk to each other. This is what's called Microservices Architecture. Each microservice focuses on a specific feature: tracking expenses, managing budgets, handling authentication, sending notifications, and so on.

Why microservices?

- If something breaks, the rest of the app still works. This makes the system more reliable.
- We can scale services individually. If the AI chatbot gets popular, we can give it more power without affecting the rest of the app.
- It lets different team members work on different parts of the system without stepping on each other's toes.
- We can mix and match technologies—for example, using Python for AI while sticking with Node.js for backend APIs.
- It supports continuous delivery and deployment. We can push updates to individual services without taking the entire system offline.
- Services can be containerized using Docker and managed with Kubernetes if the project grows to that scale.

7.2 Design Pattern: Model-View-Controller (MVC)

For the design pattern within each service, especially the frontend and backend, we're going with the Model-View-Controller (MVC) pattern. It breaks the application into three layers:

- Model: This part handles the data, things like your transaction history, saved budgets, and user goals.
- View: This is the user interface, the actual screens and components users interact with.
- Controller: This handles input from the user. It figures out what the user is trying to do and then tells the model or view what to do next.

Using MVC helps us keep the codebase clean and makes it easier to manage over time. If we want to update how a chart looks, we only need to touch the view. If we want to change how we store transactions, that's a model change. It keeps things neat.

We chose MVC because it separates concerns effectively. Teams working on the front-end UI don’t have to worry about business logic or data persistence, and back-end developers don’t need to worry about how the page renders. It improves both development speed and maintainability.

Major System Components:

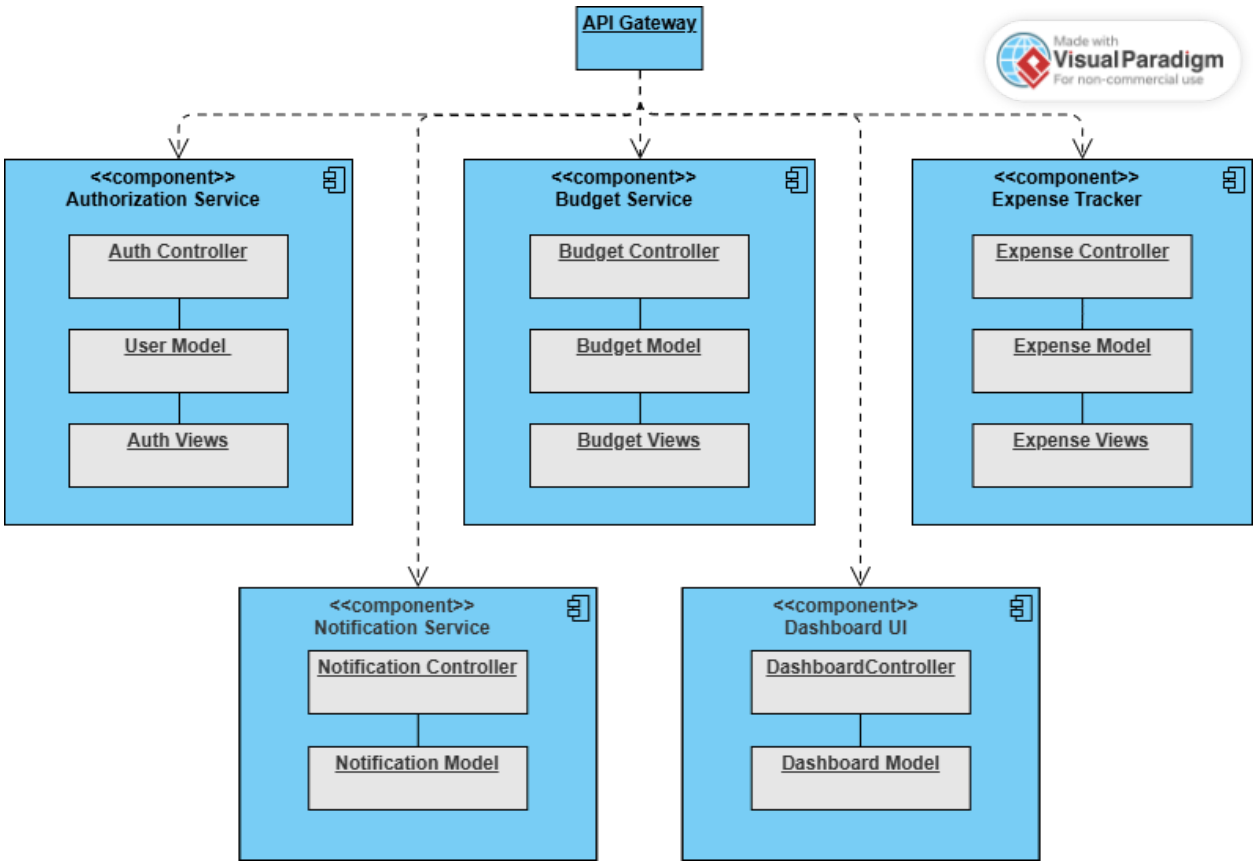
Service	Function
Authorization service	Handles user login, signup, and two-factor authentication
Budget service	Manages user-set spending limits, tracks actual vs. goal
Expense Tracker	Logs daily spending, categorizes it automatically
AI insight service	Gives smart suggestions, like “ You're spending more on food. “
Notification service	Sends email and SMS reminder for bills and budgets
Dashboard UI	Pulls everything together and shows it in a nice and clean layout
API gateway	Direct requests from users to the correct service

7.2.1 Model-View-Controller Diagram (UML Diagram)

The diagram illustrates how the MVC architecture helps maintain clean separation between data management, business logic, and presentation layers across your financial system's components.

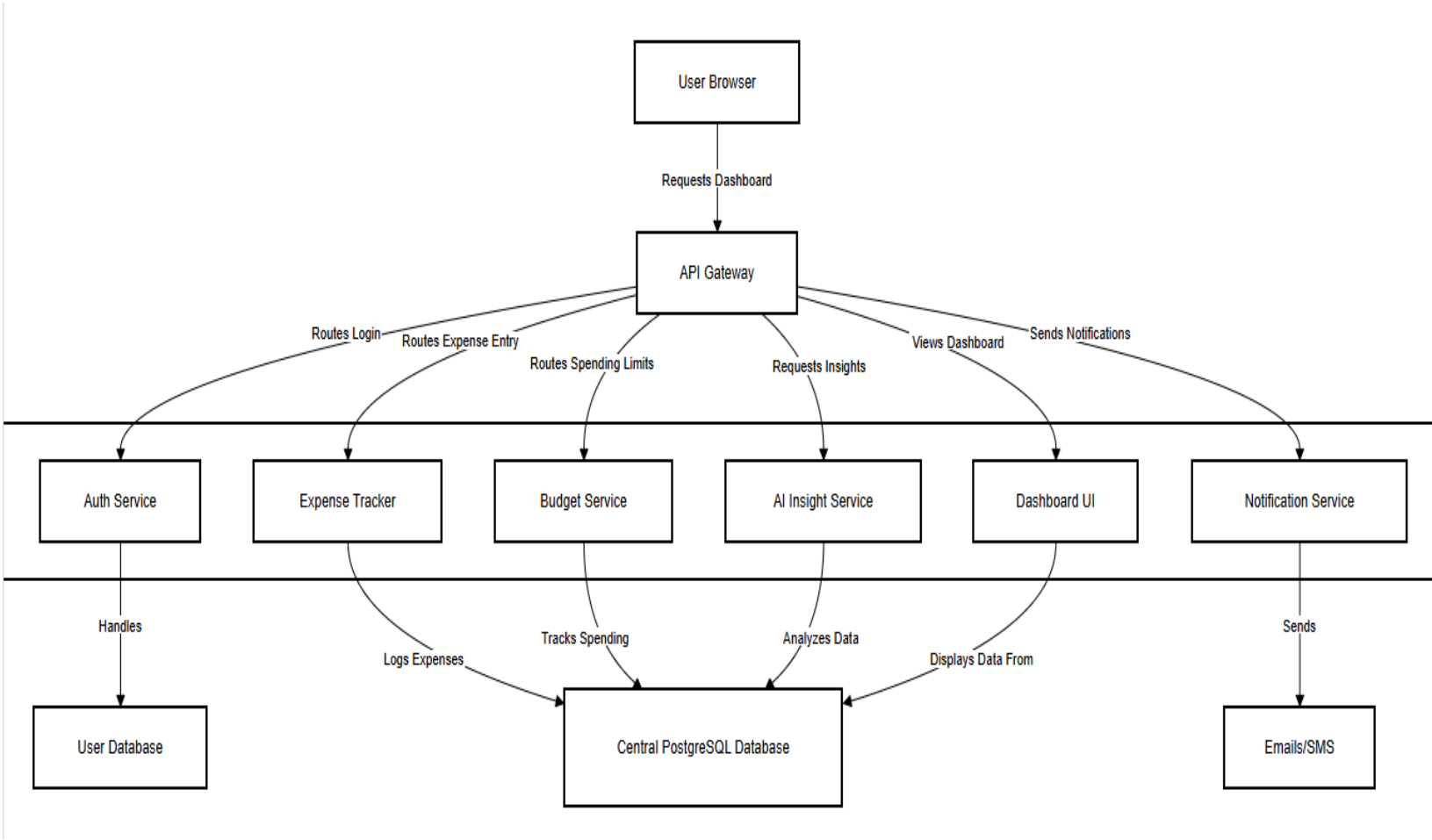
Each service implements the MVC pattern with appropriate components:

- **Controllers**: Handle user input and determine what actions to take
- **Models**: Manage data, business logic, and persistence
- **Views**: Present data to users and handle UI interactions

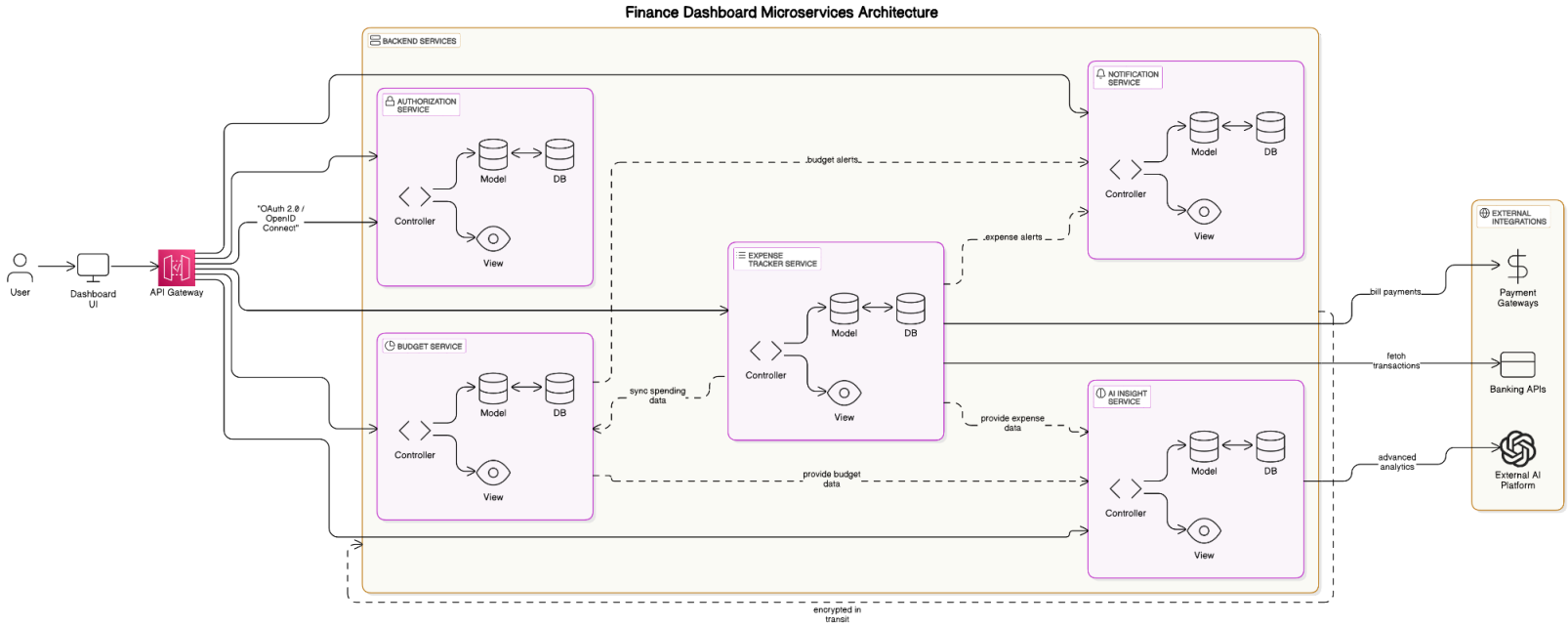


7.3 Architecture Diagram

Imagine the user visiting our dashboard from their browser. That request hits our API Gateway first, which then decides where the request should go. If they're logging in, it routes them to the Auth Service. If they're entering expenses, it goes to the Expense Tracker. Each service talks to a central database (PostgreSQL) but also has its own small local data store for performance.

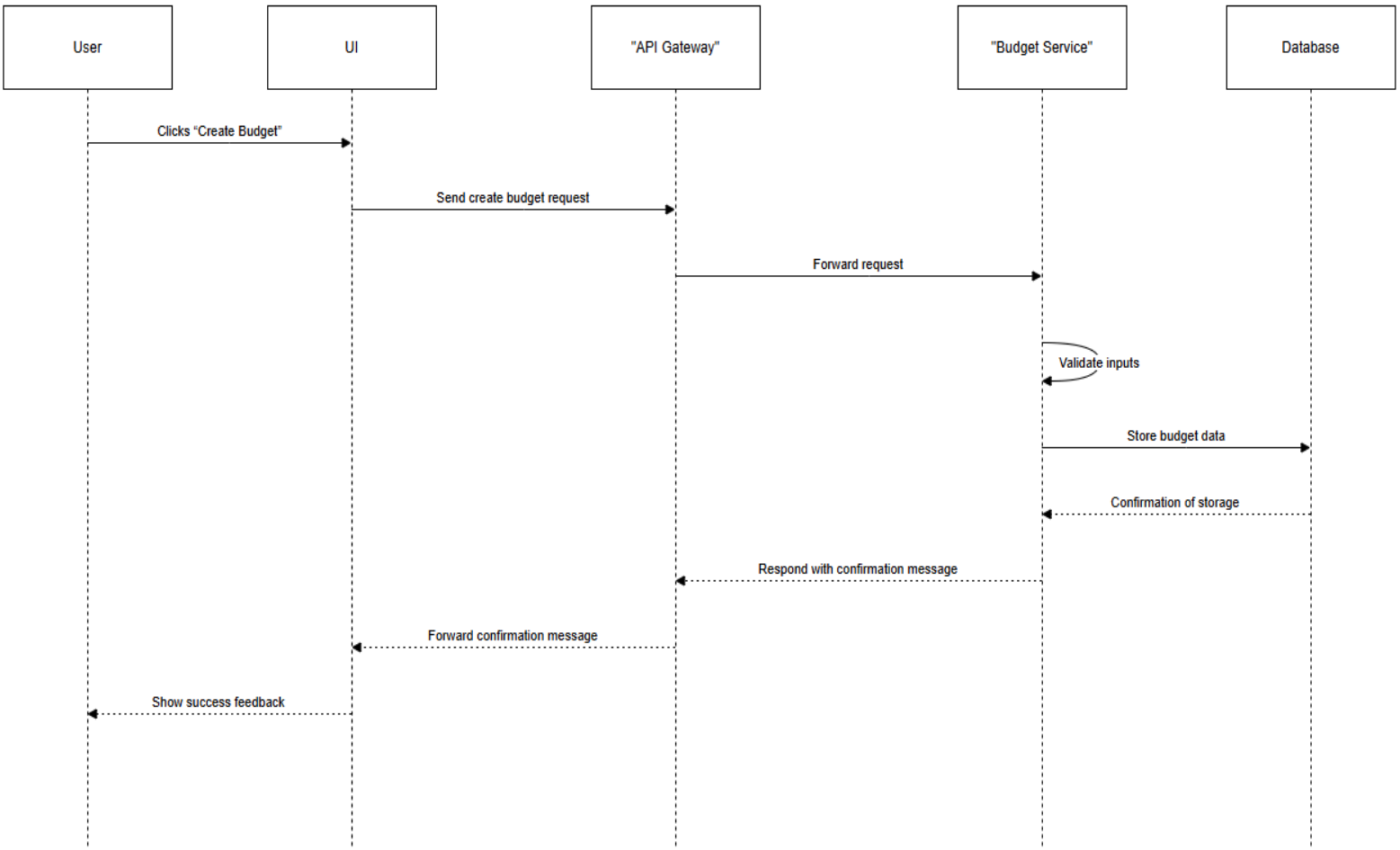


7.4 Microservice Architecture Diagram



7.5 Sequence Diagram: Creating a Budget

- The user clicks “Create Budget”
- The request is sent to the API Gateway
- API Gateway forwards it to the Budget Service
- Budget Service validates the inputs and stores them in the database
- It responds with a confirmation message, and the UI shows success feedback



7.6 Security Design

Input and Query Security

- All users attempting to access the website and interact with any forms will at the minimum be required to register and verify an email
- Next since we will be using PostgreSQL it will be essential we prevent SQL injection by using parameterized queries in our backend for any method used to query our database.
- We will also ensure that the server rejects malformed inputs early.

Data Encryption

- Any sensitive user data such as passwords and tokens will be hashed using Argon2
- **In transit** we will be using HTTPS (TLS 1.2+) across all services and APIs.
- **Token security:** JWTs are used for session management, and tokens are stored securely with expiration times and rotation policies.
- Although at the moment the most sensitive information we will have to handle are user passwords and tokens, in the future if we expand to integrating banking APIs into our product then we will also have to scale security measures.

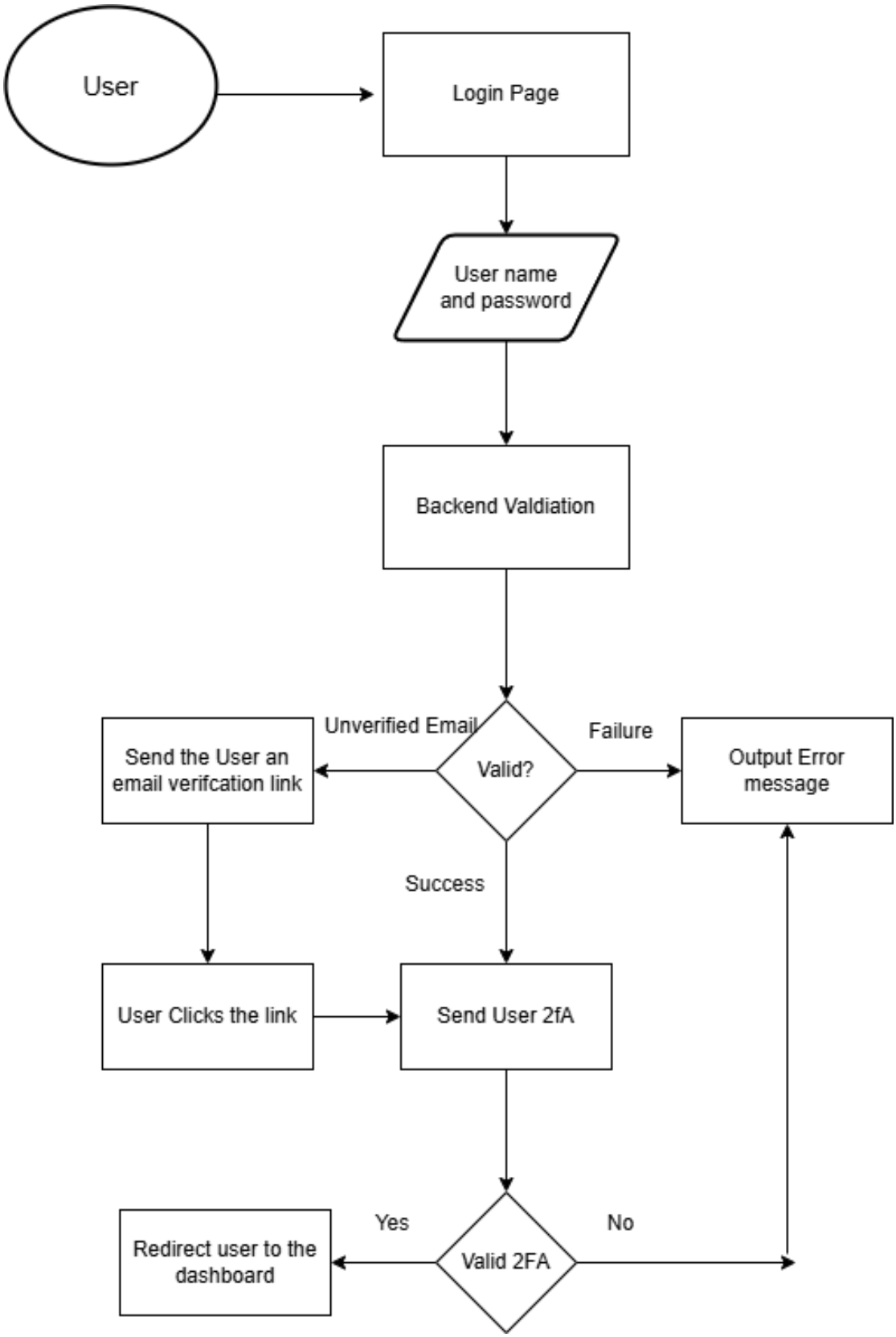
Authentication & Authorization

- **User Signup/Login:**
  - Email and password required (password is hashed).
    Email verification required before the account is activated.

- **Two-Factor Authentication (2FA):**
  - Users must enter a code sent via email or authenticator app.
  - 2FA is enforced on login and sensitive operations (like connecting bank accounts)

- **Authorization:**
  - Role-based access (admin, user).
  - Certain routes are protected using JWT scopes or middleware guards.

7.6 Security Flowchart



7.7 Error Handling and Recovery Design

Our error handling will build on top of our microservices design. By utilizing microservices we can potentially isolate failures across the website.
For example, if our AI chatbot were to go down, then the rest of our features would ideally still be available.

Database Failover

- Since the main feature of our website is to be able to store and display customized recommendations as well as data imputed by the user, it's essential we have a method to ensure these features are available in the event we encounter an issue
- One way we can achieve this is by implementing a **Primary–Replica Setup** with PostgreSQL.
- We will have our primary database which handles all writing to the client side.
- We will also have one or more replica databases which only have read access in normal circumstances, and are constantly copying and saving data from the primary database.
- In the event our primary database goes down, we can easily promote a replica to the primary database and continue operation.

If we decide to deploy our website utilizing AWS RDS then this will all be automatically handled behind the scenes. Otherwise we will need to find an alternative method.

Connection Pool Management
Since creating database connections to PostgreSQL are a limited resource, as well as potentially costly, it will be important to have a connection pool to manage this.

Connection pools manage lots of open database connections and reuse them, so they are not being opened and closed constantly.

We could utilize **PgBouncer** to easily achieve this. It's a lightweight connection pooler that helps greatly with the connection pooling problem, as well as handling failovers.

Connection pools also help in the event of a database failover, as they can attempt a reconnection, or if that fails they can switch to a replica, or even just switch to reading from a replica if that's all that is required.

7.8 Deployment Architecture

To ensure that our web app can be deployed reliably, maintained, and scaled, we've chosen a cloud native deployment architecture built on top of AWS, using Docker containers for service encapsulation and Github actions for CI/CD automation. This setup aligns well with our microservices approach to our web app, and provides flexibility, performance, and security for both development and production environments.

Key Components

Component	Purpose
Github Actions	CI/CD automation (build, test, deploy pipelines)
Docker	Containerize each microservice
AWS EC2	Host services in the cloud
AWS RDS	Managed database hosting with high availability
AWS S3	Store static assets
AWS IAM & Secrets Manager	Manage access control and environment secrets securely
Nginx / API Gateway	Route external requests to internal services
Monitoring Tools like cloud watch	Log and monitor app performance

Deployment Workflow (CI/CD Pipeline)

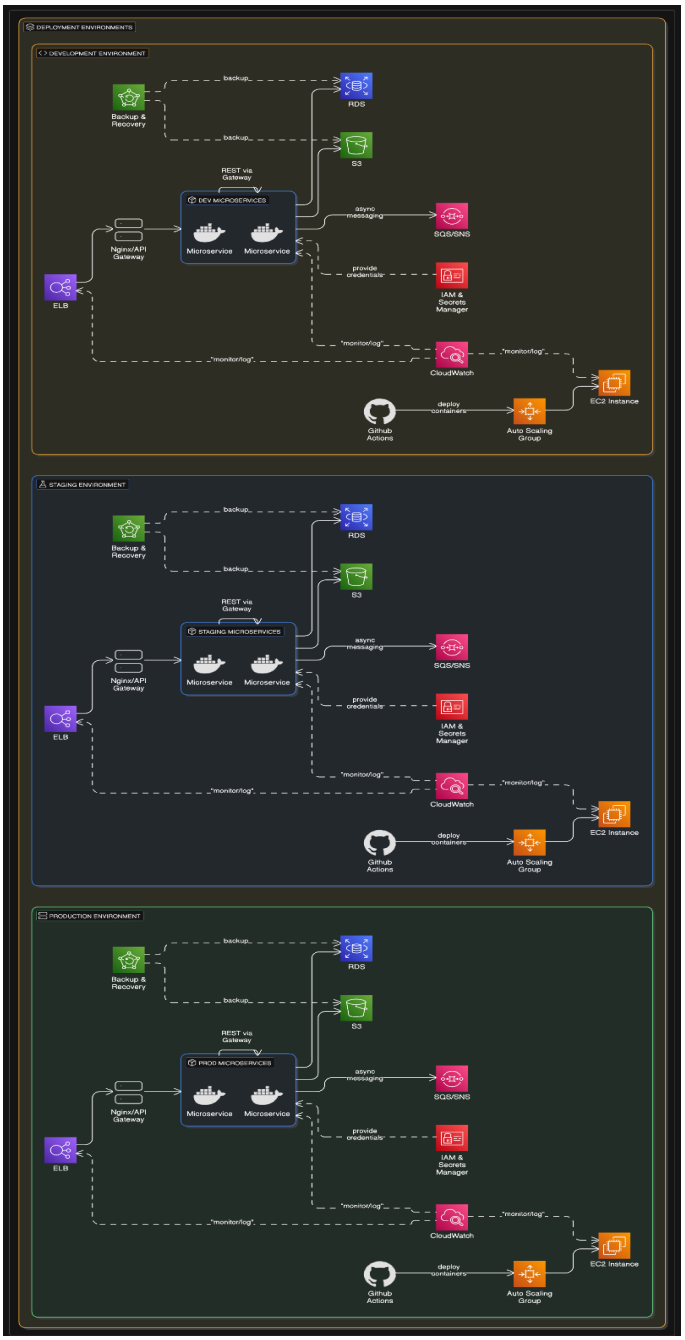
We use GitHub Actions to automate code integration, testing, and deployment with the following steps:

1. **Code Push -** Developers push code to GitHub repositories.
2. **Build & Test Phase**
    - Each microservice is built into a Docker image.
    - Unit integration tests are automatically executed.
3. **Docker Image Push**
    - Successful builds are tagged and pushed to **Docker Hub** and **AWS ECR (Elastic Container Registry).**
4. **Deployment Phase**
    - Services are deployed to **EC2** instances (initially) using Docker Compose or directly to **AWS ECS** if scaled.
    - Configuration files (e.g., environment variables, secrets) are pulled from AWS Secrets Manager.
5. **Post-Deployment Tests**
    - Health checks and smoke tests confirm that deployment succeeded.
    - Alerts are sent if any service fails.

Infrastructure and Orchestration Strategy

While we currently anticipate deploying services to AWS EC2 with Docker Compose, our architecture supports scaling to **Kubernetes (EKS)** if needed. Using **Infrastructure-as-Code (IaC)** with tools like **Terraform** or **AWS CloudFormation** is also under consideration for repeatable and consistent environment setup.

7.8 Deployment Architecture Diagram



7.9 Why These Choices Fit Our Project

- **User-Centered Design**: Our system is built for flexibility, but also clarity. Users want a smooth, clean experience, and these patterns help us deliver that while keeping the backend smart and scalable. MVC helps ensure that changes to the user interface don't disrupt the underlying logic, and vice versa. With clean separation between concerns, we can focus on delivering features that feel intuitive and responsive, like customizable dashboards or guided setup flows. Accessibility and responsiveness are also key pillars—ensuring that users on different devices or with different needs all enjoy a seamless experience.
- **Team Collaboration**: With each part of the project split out into Model, View, and Controller layers, our team can contribute without stepping on each other's toes. Frontend developers can refine the UI and UX within the View layer, while backend developers work on data models and business logic. Controller logic serves as the handshake between the two, enabling parallel development and faster iteration. This separation also makes it easier to onboard new team members or delegate specific tasks without losing momentum.
- **Future Growth**: As more users join or features are added—like bank account syncing, predictive financial forecasting, or real-time spending alerts—our system can scale without needing a full rewrite. MVC and modular design make it easy to plug in new components, APIs, or services without impacting existing functionality. We're also planning for microservice compatibility, allowing individual services (like budgeting tools or analytics engines) to scale independently and efficiently.
- **Security & Reliability**: With 2-Factor Authentication (2FA), secure JWT-based session management, data encryption both at rest and in transit, and database failover strategies, the system is designed to protect user data and stay reliably available. We also implement role-based access control, rate-limiting, and continuous vulnerability scanning to ensure only authorized users interact with sensitive features. Automated testing and logging help us catch errors before they affect users, and downtime is minimized with automated backup and restore mechanisms.

8. UX Design

8.1 Aesthetic (UI Design)

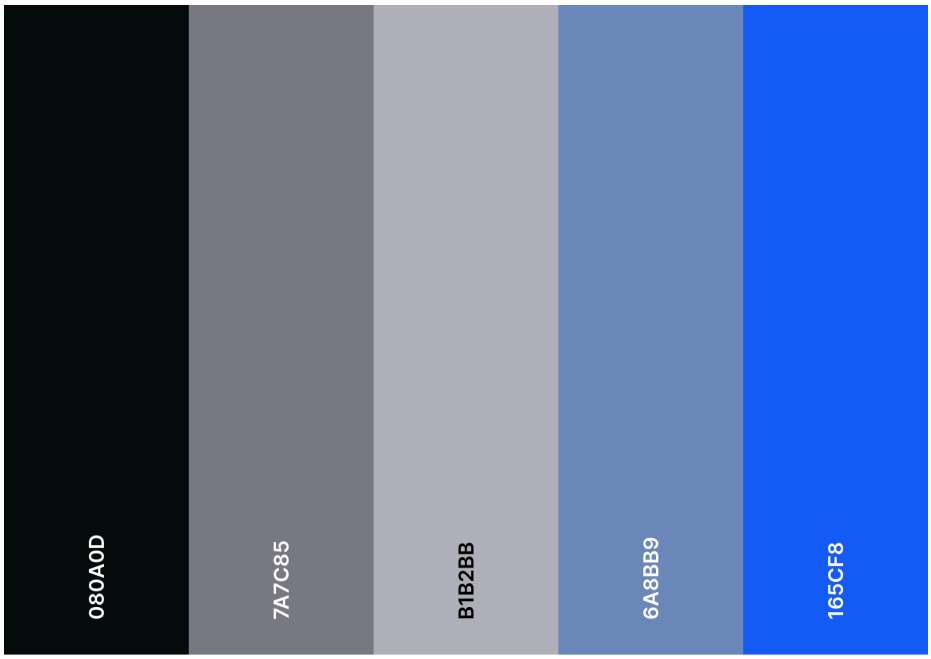
The finance dashboard will adopt a minimalistic visual style to promote a seamless and distraction-free user experience. By focusing on simplicity and clarity, the design will allow users to engage with key features effortlessly and stay focused on their financial goals.

Color Scheme:

The color scheme will use a clean and straightforward palette to reduce visual clutter and enhance usability. The design will draw inspiration from OpenAI's interface, offering both light and dark modes to accommodate user preferences.

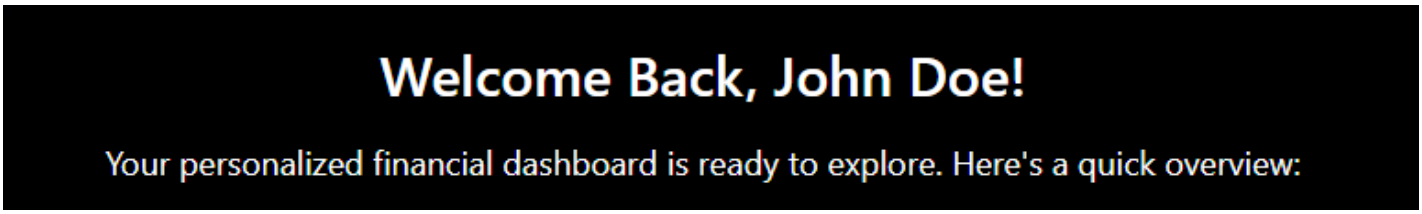
- Light Mode will feature a white background with black text, ensuring high contrast and readability.

- Dark Mode will incorporate a dark grey background with white text, providing a visually comfortable alternative for low-light environments.



Typography:

The selected typography prioritizes readability and visual simplicity. A sans-serif font such as Open Sans or Helvetica Neue will be used to maintain a professional and user-friendly appearance across all screens. These fonts are widely regarded for their legibility on digital interfaces and align well with the overall minimalist design approach of the application.



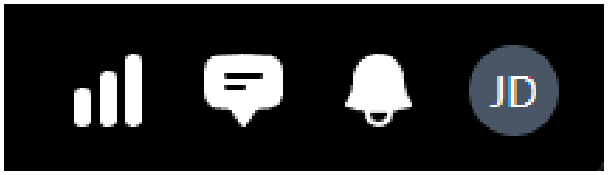
Layout:

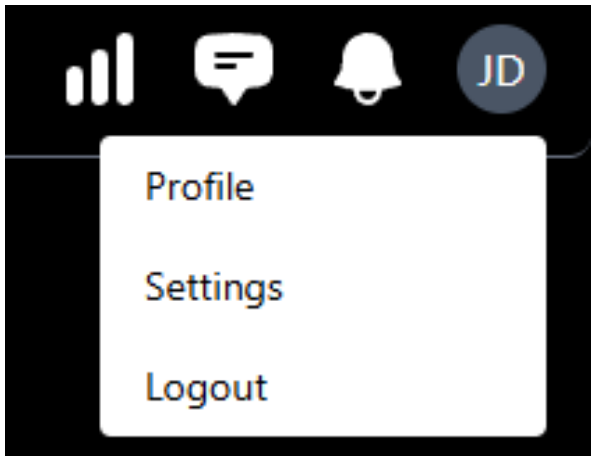
The layout will be modeled after other personal finance dashboards like Mint, which are known for their clean, intuitive design. Key information will be clearly organized and easy to locate, ensuring a streamlined navigation experience for users. This familiar structure helps reduce the learning curve and supports efficient user interactions.

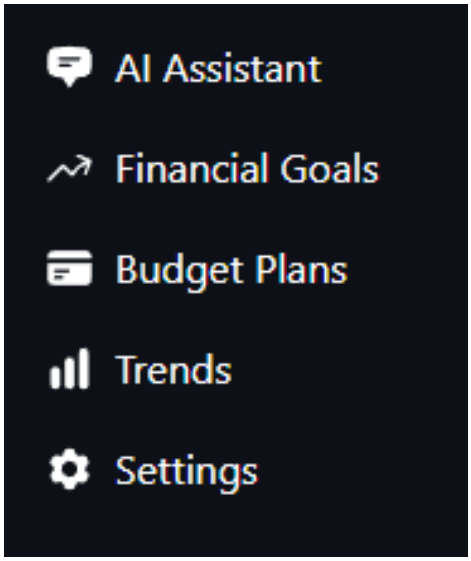
Iconography:

We will incorporate simple and intuitive icons—such as dollar signs, calendars, and pie charts—to enhance visual clarity and quick recognition. These familiar symbols support efficient

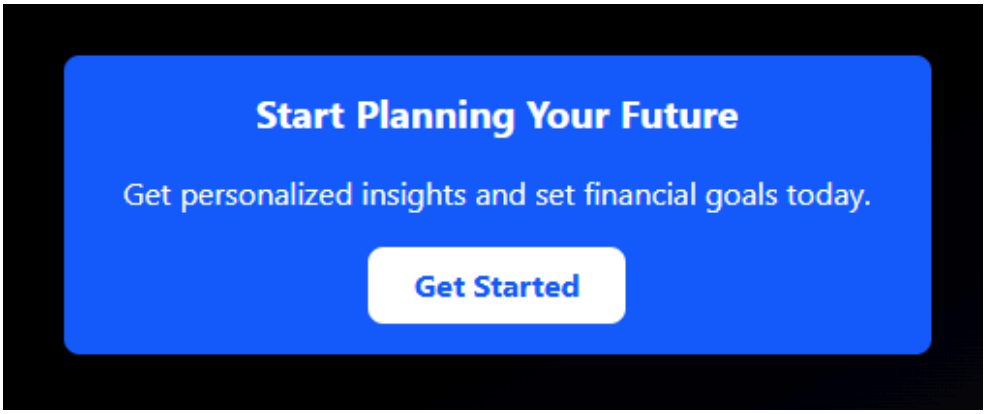
visual scanning throughout the interface, helping users locate key features and actions with minimal cognitive effort.







Subtle animations, such as button feedback, loading indicators, and toggle transitions, will be integrated to guide user interactions and provide visual cues.



8.2 User Research

To define our target users, we conducted research by analyzing competitors (Mint, YNAB) and user reviews. We identified three primary user groups: young professionals, families, and small business owners. Each group requires a dashboard that is intuitive, easy to navigate, and provides actionable insights. We noticed that many users across these groups shared similar frustrations, mainly cluttered interfaces, confusing charts, and a lack of personalization. Based on this, we focused on creating a clean and simple layout with just the right amount of detail. We also looked into common features people wanted, such as easy budget tracking, notifications for upcoming bills, and the ability to set and monitor financial goals.

To better understand what users cared about most, we also paid attention to app store reviews, blog comments, and finance forums. Many users asked for faster load times, fewer pop-ups, and a smoother mobile experience. This helped us prioritize performance and mobile responsiveness early in our design process.

By taking in both qualitative and quantitative feedback, we were able to shape a dashboard that feels helpful, not overwhelming, one that adapts to real-life needs instead of trying to do too much at once.

8.3 User Personas

Persona 1: Sarah Johnson

Attribute	**Details**
Age	29
Occupation	Marketing Specialist
Income	$100,000/year
Tech Comfort Level	Moderate

Goals	● Wants a simple method to track her monthly expenses ● Could benefit from cost cutting insights
Frustrations	● Using spreadsheets is too tedious ● Current apps are too cluttered and intimidating
Behavior	● Uses a budgeting app several times a week ● Likes cross platform dashboards browser/mobile

Persona 2: David Lee

Attribute	Details
Age	48
Occupation	Small Business Owner
Income	$85,000/year
Tech Comfort Level	Moderate
Goals	● Track personal and business expenses separately. ● Forecast cash flow month-to-month. ● Quickly view upcoming bill due dates and loan repayments.
Frustrations	● Overcomplicated finance apps with too many irrelevant features. ● Hard-to-read graphs or cluttered dashboards.
Behavior	● Logs in weekly to update expenses. ● Wants mobile notifications for important financial deadlines.

Persona 3: Jasmine Patel

Attribute	Details
Age	22
Occupation	College Student
Income	High
Tech Comfort Level	$12,000/year
Goals	● Build an emergency fund. ● Understand basic budgeting and credit management.

Frustrations	● Apps that assume users already know about personal finances. ● Overwhelming number of charts and options.
Behavior	● Uses mobile apps daily. ● Prefers colorful, simple, gamified interfaces

Persona 4: Luis Ramirez

Attribute	**Details**
Age	35
Occupation	Freelance Graphic Designer
Income	~$50,000/year
Tech Comfort Level	High
Goals	● Smoothly manage irregular income ● Separate personal vs business expenses
Frustrations	● Most apps assume consistent monthly income ● Difficult to track tax-deductible expenses
Behavior	● Logs in every few days ● Prefers mobile-friendly apps with cloud sync

Persona 5: Brenda Thompson

Attribute	**Details**
Age	63
Occupation	Retired / Part-time caregiver
Income	$28,000/year
Tech Comfort Level	Low
Goals	● Track medical and utility bills ● Avoid overdraft fees or late payments
Frustrations	● Navigation-heavy apps feel overwhelming

	● Small text and unfamiliar icons
Behavior	● Logs in 1–2 times a week ● Uses tablet with accessibility settings turned on

Persona 6: Ali Reza

Attribute	**Details**
Age	31
Occupation	Uber/Lyft Driver
Income	~$40,000/year
Tech Comfort Level	Moderate
Goals	● Track daily earnings and fuel expenses ● Set and monitor weekly savings goals
Frustrations	● Hard to find tools that match gig work lifestyle ● Wants faster input options while on the go
Behavior	● Uses mobile daily, especially after shifts ● Prefers voice input and simplified dashboards

Persona 7: Emily Tran

Attribute	**Details**
Age	26
Occupation	Software Engineer
Income	$125,000/year
Tech Comfort Level	Very high
Goals	● Automate recurring financial tasks (e.g., rent, subscriptions) ● View detailed analytics for saving and investing habits
Frustrations	● Limited customization of financial categories ● Apps that don’t sync well with investment platforms
Behavior	● Logs in daily on both desktop and mobile

	• Expects API integration with external tools like Robinhood and TurboTax

Persona 8: Marco Delgado

Attribute	Details
Age	42
Occupation	Truck Driver
Income	$90,000/year
Tech Comfort Level	Low to Moderate
Goals	• Track gas expenses and food costs while on the road • Get reminders for bills and important financial dates
Frustrations	• Most apps don't work well offline or in areas with poor reception • Complicated interfaces that are hard to use on the go
Behavior	• Uses mobile during breaks • Prefers simple visuals and voice alerts instead of charts

Persona 9: Aisha Khan

Attribute	Details
Age	39
Occupation	Single Part and Nurse
Income	$60,000/year
Tech Comfort Level	Moderate
Goals	• Set monthly budgets for childcare, groceries, and utilities • Share financial progress with a trusted family member
Frustrations	• Apps without shared access or family view modes • Overwhelmed by financial jargon and too many charts
Behavior	• Logs in several times a week after work • Prefers clean UI with clear icons, summaries, and friendly nudges

Persona 10: Carlos Rivera

Attribute	Details
Age	19
Occupation	Part Time Retail Worker and College Student
Income	$10,000 a year
Tech Comfort Level	High Level
Goals	● Track small, day-to-day expenses like meals, transit, and books ● Start saving for emergencies and future tuition costs
Frustrations	● Most finance apps assume higher income or full-time work ● Feels discouraged by complex dashboards and long setup processes
Behavior	● Uses mobile apps daily, often between classes or during work breaks ● Prefers fast-loading, minimal interfaces with goal tracking features ● Interested in short tips or reminders that help build good money habits over time

8.4 UX Frameworks

To guide the design of our Finance Manager Dashboard, we are adopting the Double Diamond UX framework. This model ensures that we understand user needs before developing solutions, using the following four stages:

- **Discover:** Conduct user research and identify pain points in personal finance apps.
- **Define:** Narrow down core features like budgeting, bill tracking, and chatbot assistance.
- **Develop**: Create wireframes and mockups, testing early designs for usability.
- **Deliver**: Finalize the interface and launch with an iterative feedback loop.

We also apply Design Thinking during brainstorming and prototyping sessions to stay focused on real user needs. This helps us explore a wide range of ideas and refine them based on user feedback before building the final version.

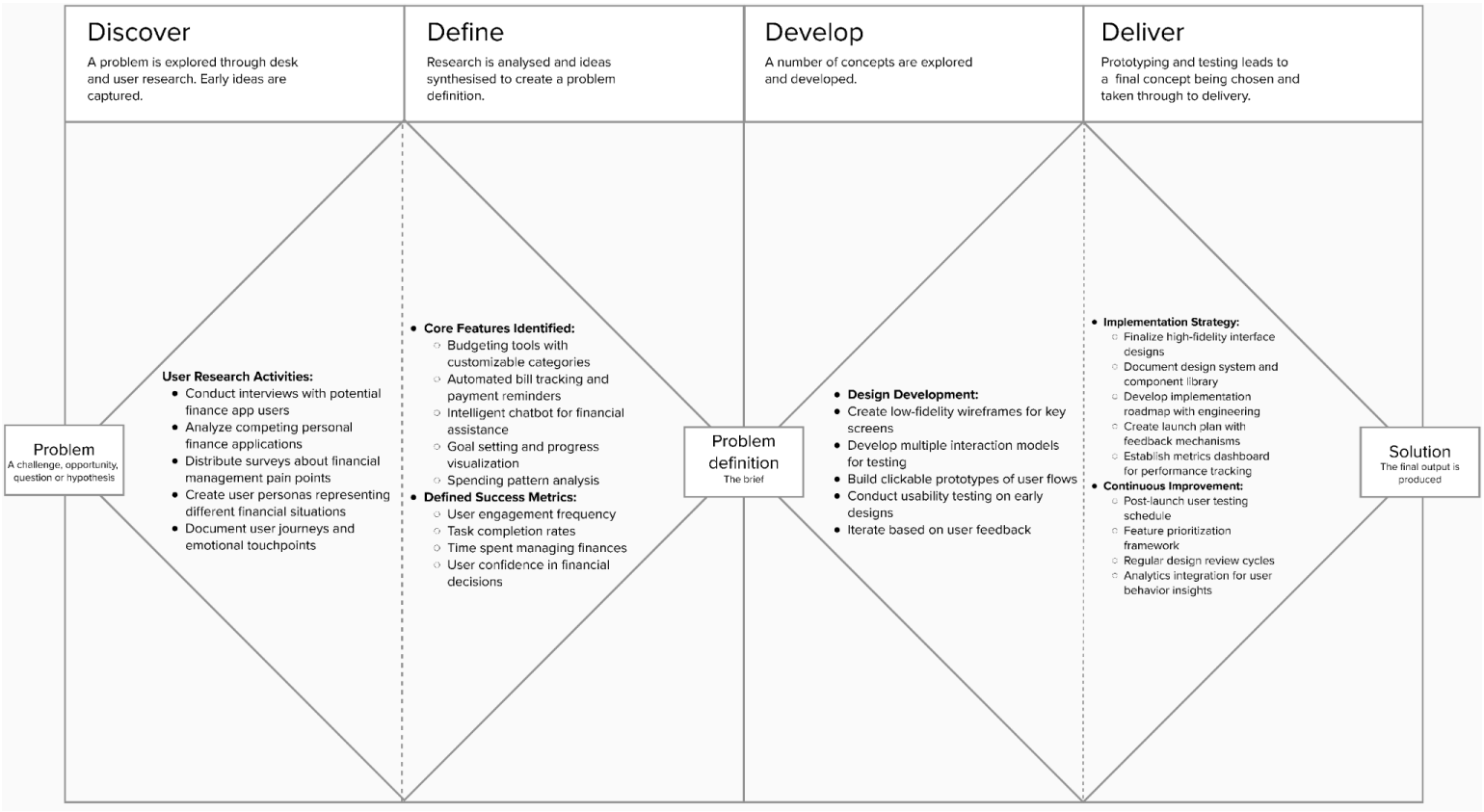
To support this process, we also use:

- Low-fidelity wireframes to quickly test ideas without overinvesting in full designs too early.

- User feedback loops throughout the development process to catch issues early and make improvements continuously.
- Simple usability testing with everyday users to see how they interact with the dashboard and where they might get stuck.
- Agile design principles to stay flexible and make changes as we go, instead of waiting until the end to fix problems.
- Click-through prototypes to simulate real user flows before coding, which helps spot confusion points early.
- Collaborative design reviews, where team members give feedback on each version to improve clarity, consistency, and functionality.

By combining these approaches, our team can build a dashboard that feels intuitive, works smoothly, and meets the real needs of our users.

8.4.1 Framework Diagram (Double Diamond Diagram)



8.5 UX Architecture and Design

Our dashboard is built around a simple, layered structure that makes it easy for users to find what they need and take action quickly. The top navigation bar gives access to the main sections: Dashboard, Expenses, Goals, Insights, and Settings. Each section shows a clean workspace with only the most important info up front.

We use quick-add buttons, pop-up confirmations, and progress bars to guide users as they interact with the system. For example, adding a new expense or goal only takes a few clicks, and users get instant feedback.

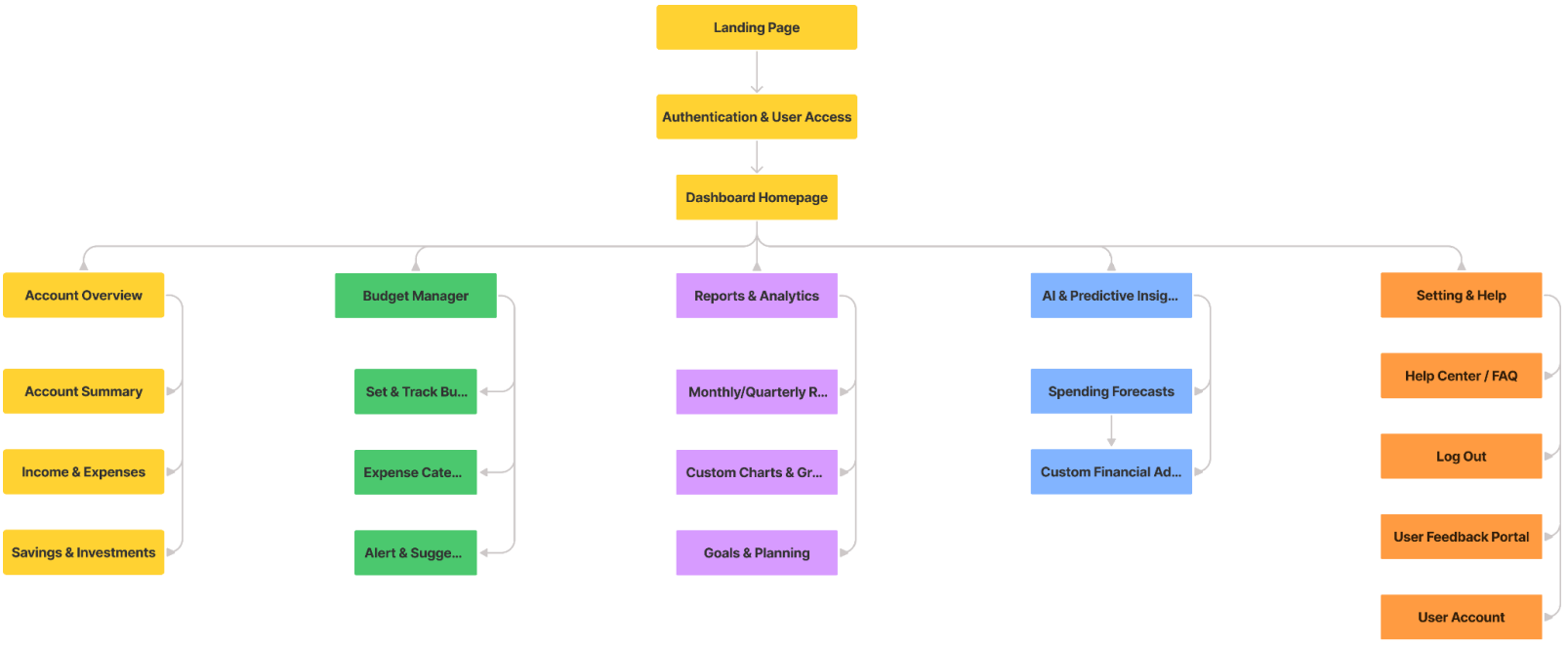
The layout follows natural reading patterns (top-down, left to right) and adjusts automatically for different screen sizes. Whether someone's on a laptop or phone, the design stays clear and easy to use.

We also built in accessibility features like keyboard navigation, screen reader support, and high-contrast options. Everything is organized so users can get things done with as little clutter and confusion as possible.

To improve user flow and reduce frustration:

- Each section loads independently, which keeps the app fast and responsive.
- Helpful tooltips and labels are included next to icons or new features to explain what they do.
- We added a collapsible sidebar option to give users more screen space when needed.
- Confirmation messages and undo options help prevent mistakes and boost confidence when taking actions.
- Overall, the architecture is built to support both new users who want simplicity and experienced users who need efficiency.

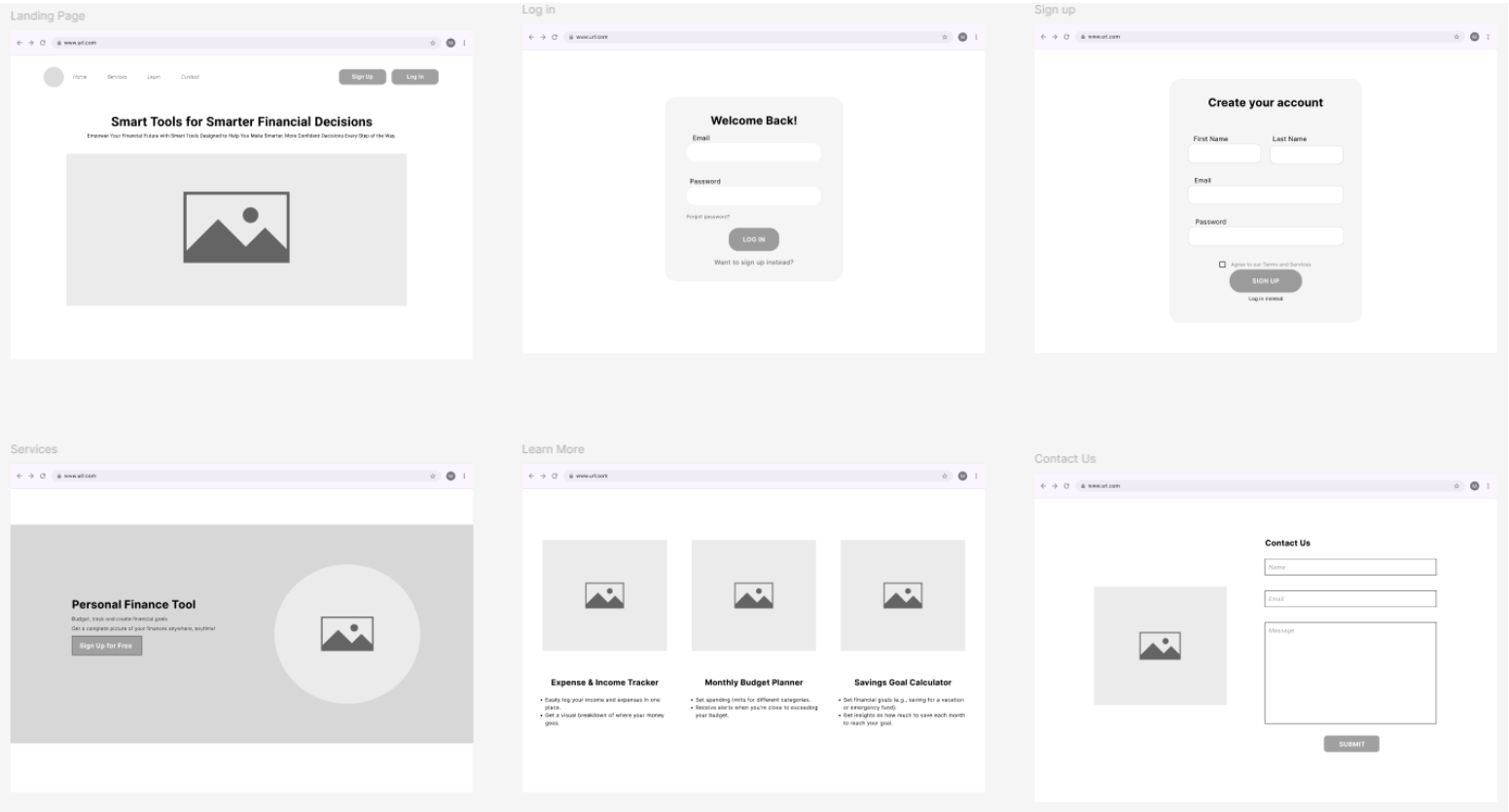
8.5.1 UX Architecture Diagram (Information Architecture Diagram)



8.6 UX Mockup

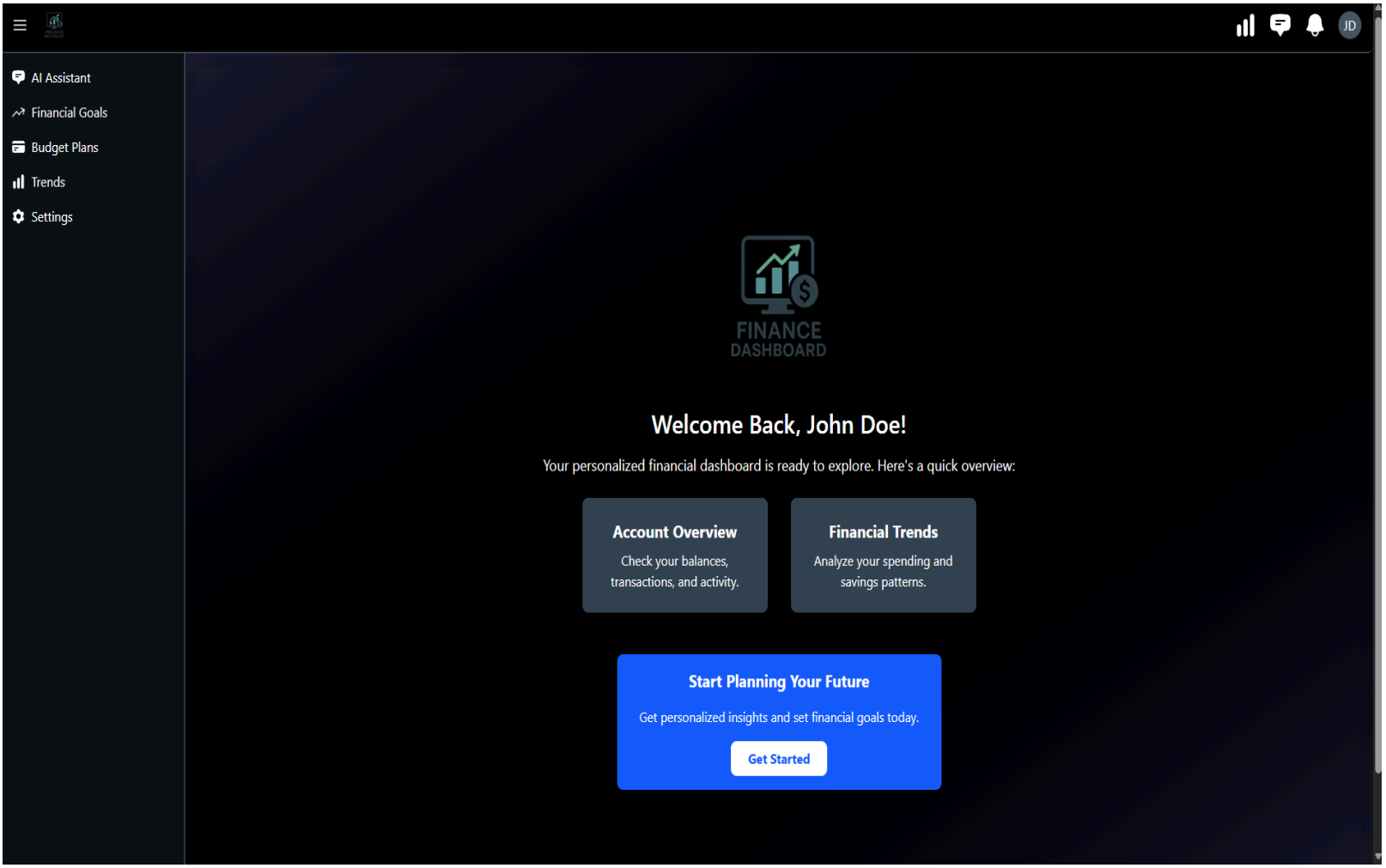
Our initial mockup presents a streamlined, mobile-responsive dashboard. Key features include:

- **Top Navigation Bar**: Home, Expenses, Goals, AI Chatbot, Settings
- **Sidebar or Dashboard Tiles**: Each major function is accessible via visually distinct cards (e.g., “View Budget,” “Ask Chatbot,” “Track Bills”)
- **Central Area**: Displays active charts (monthly spending, cash flow trends), updated in real-time
- **AI Chatbot Widget**: Docked in the bottom right for ongoing financial insights and support
- **Dark Mode Toggle**: Instantly changes color theme based on user preference
- **Responsive Design**: Scales smoothly across devices from desktops to phones
- **Quick Actions Panel:** One-click access to add expenses, set savings goals, or check recent transactions
- **Daily/Weekly Highlights:** A banner or tile at the top of the dashboard showing helpful summaries, like "You’re $50 under budget this week!"
- **Customizable Widgets:** Users can drag-and-drop cards to rearrange sections based on what matters most to them
- **Status Icons and Alerts:** Small icons and color indicators (e.g., red for overdue bills, green for goal progress) provide quick visual updates

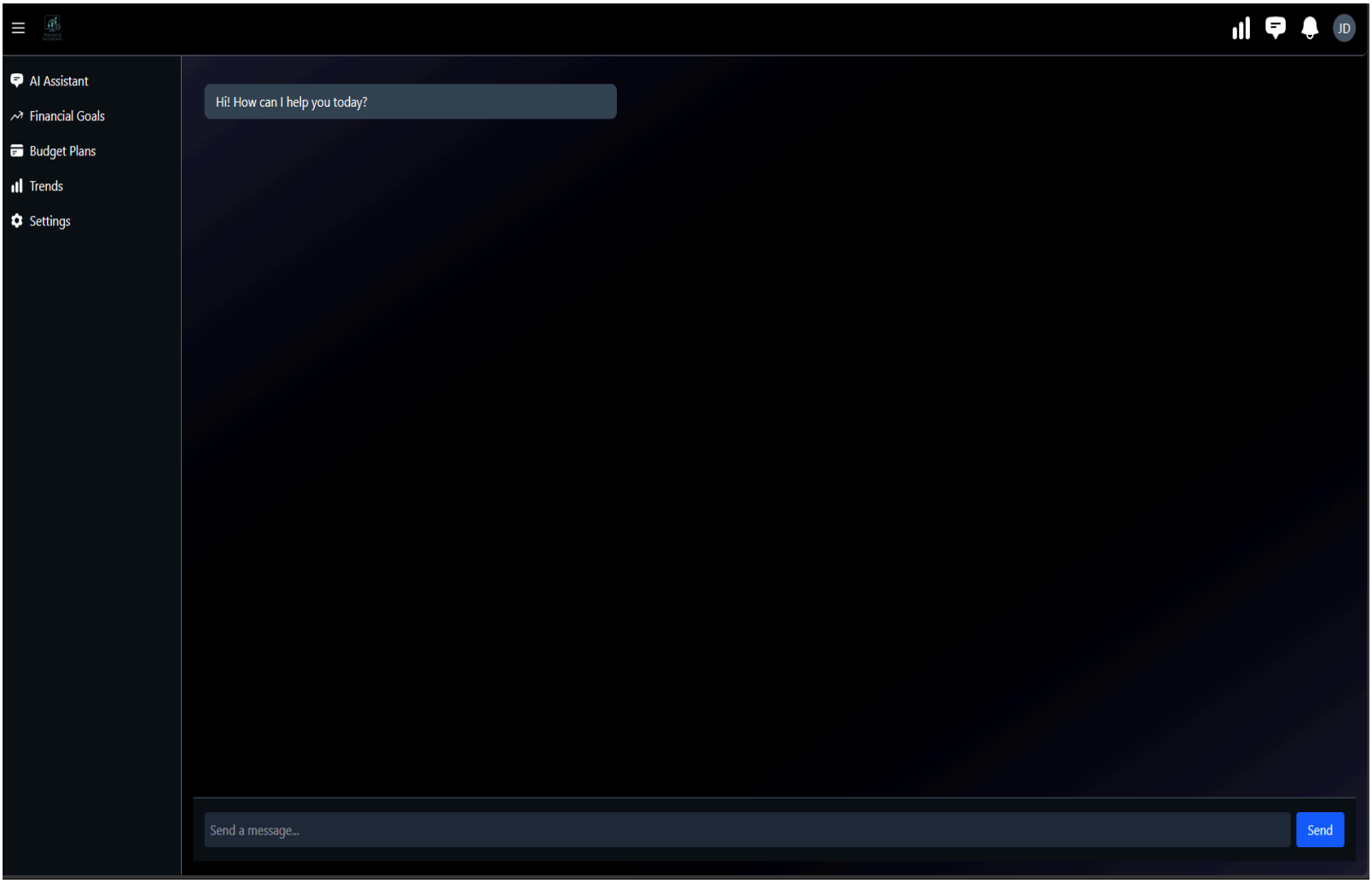


9. Prototype

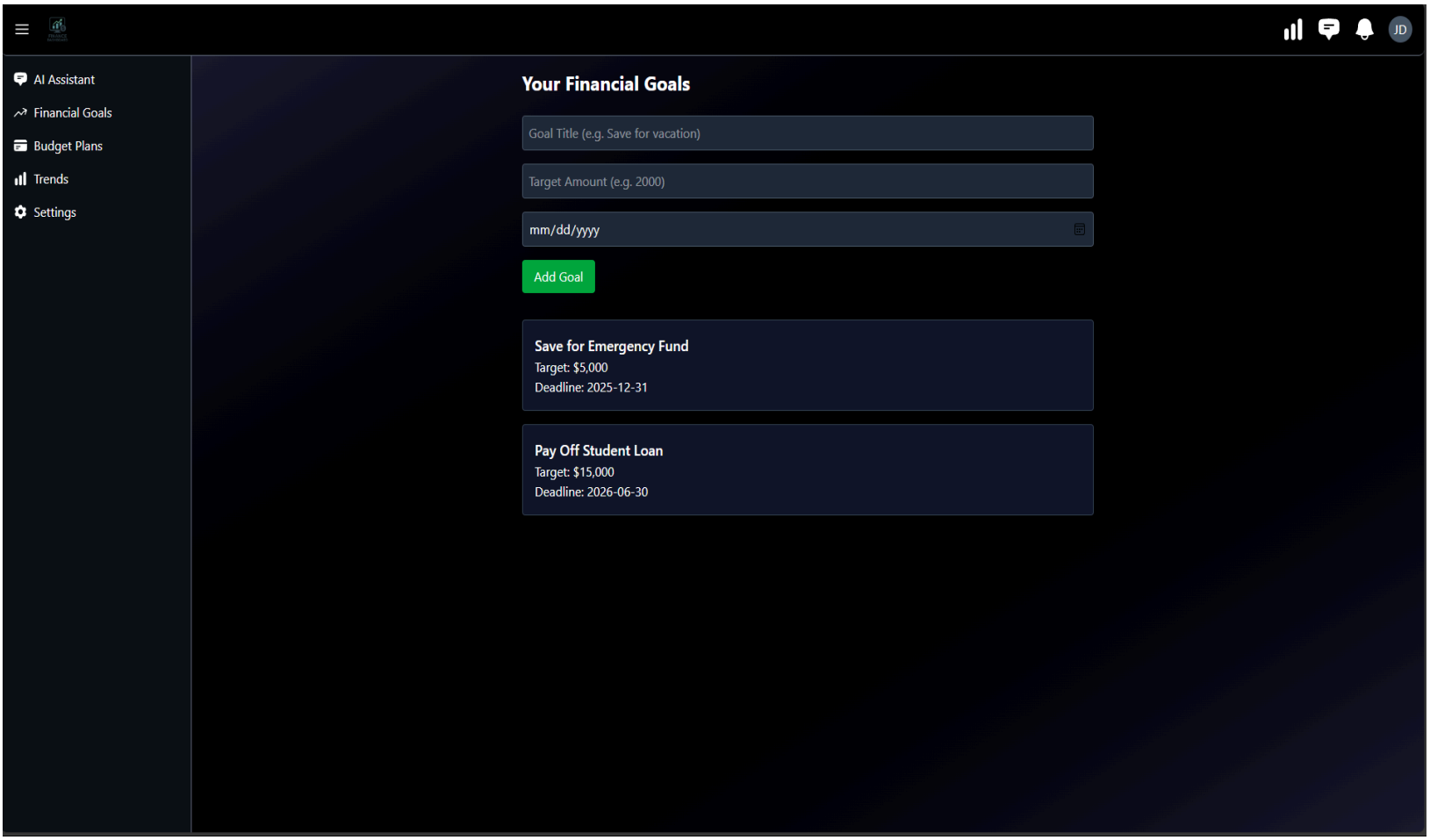
This is the landing page for our Finance manager dashboard which provides quick links to pages



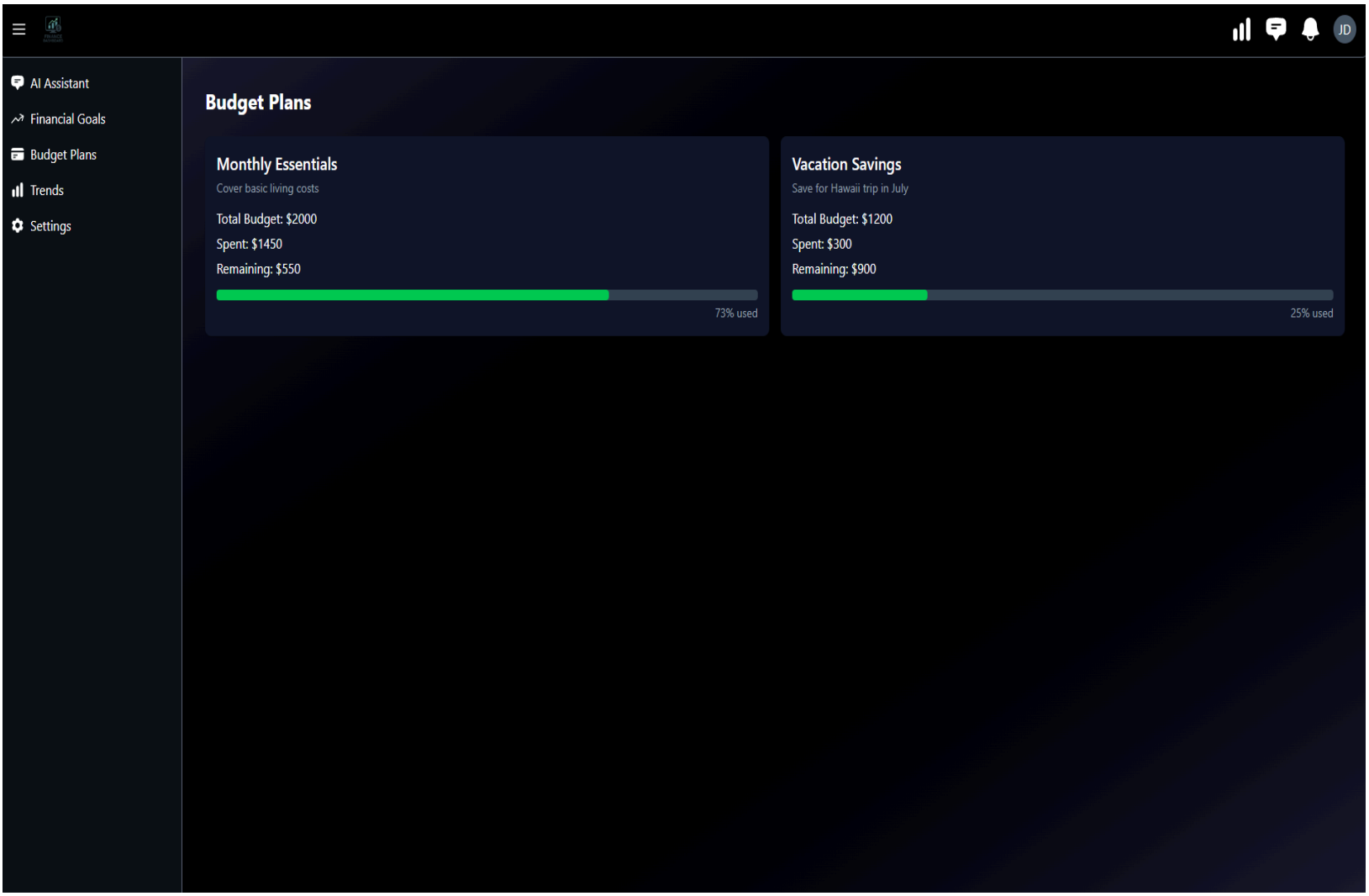
they might want, as well as a get started link to provide important information.



This is the AI assistant page which will be able to provide the user with tailored information regarding the information that they submit from the other pages/forms.

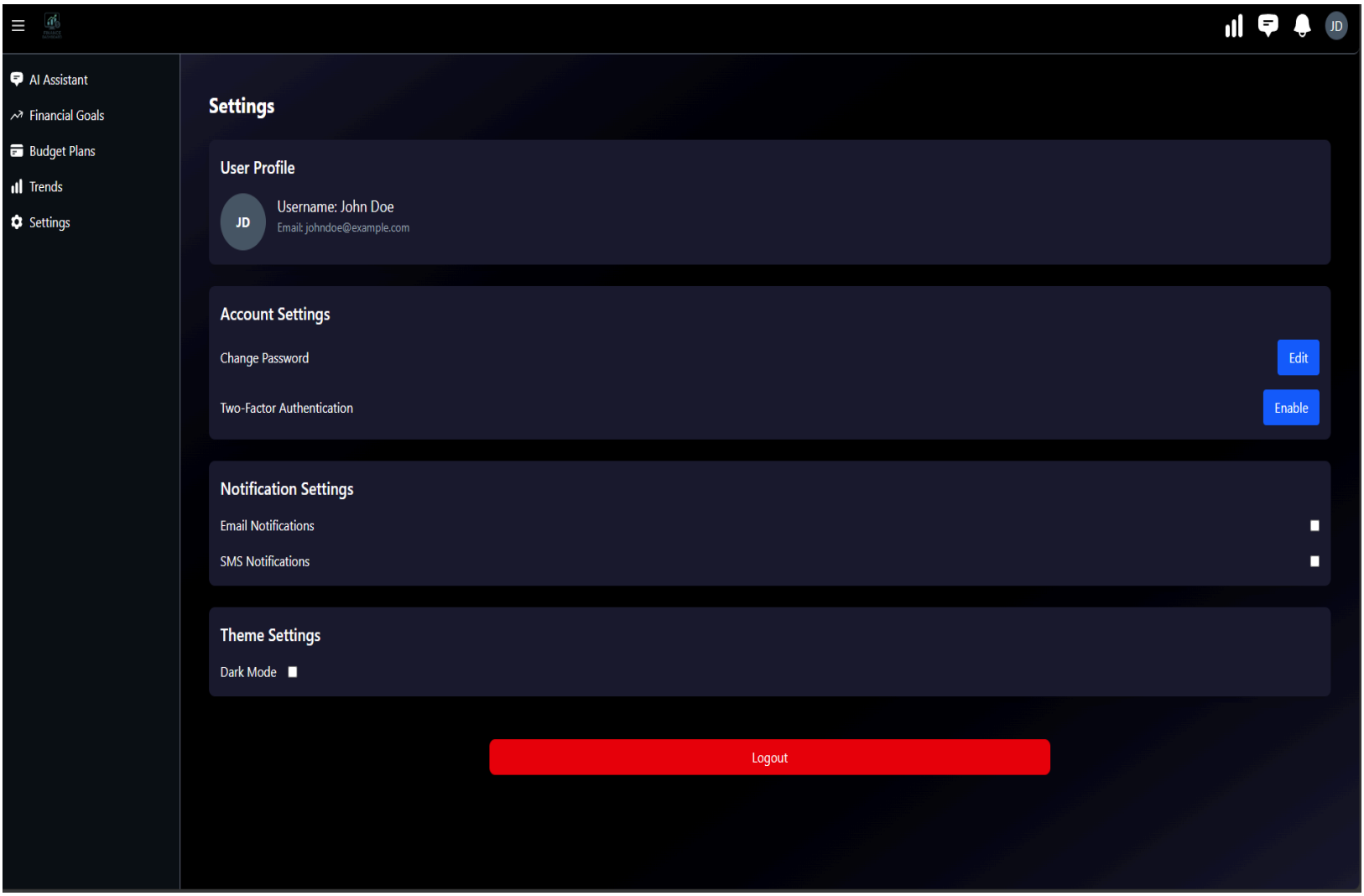


This is the Finance goals page where users will be able to insert their information and generate custom savings goals for specific things.

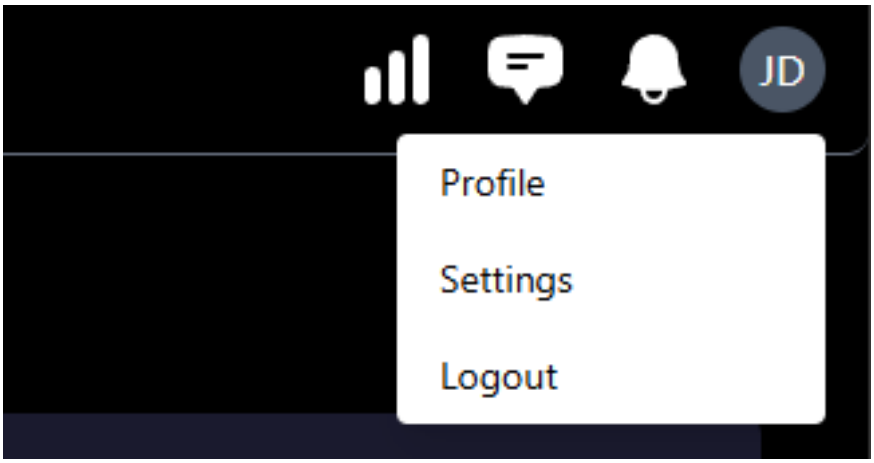


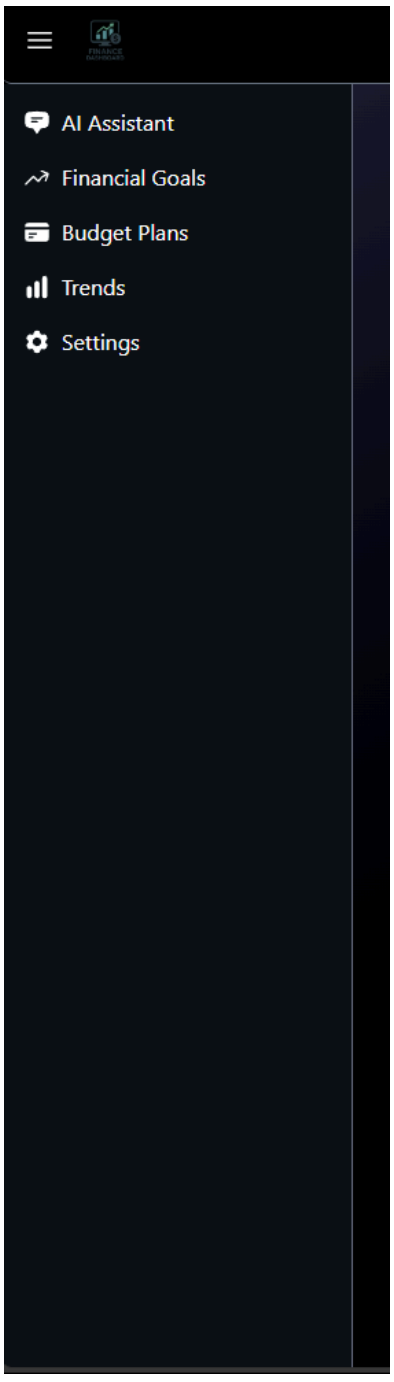
This is the Budget plans Page which will display the users current status in relation to their savings goals.

This is the Financial trends page which will display how the user allocates their funds for the time frame that they specify.

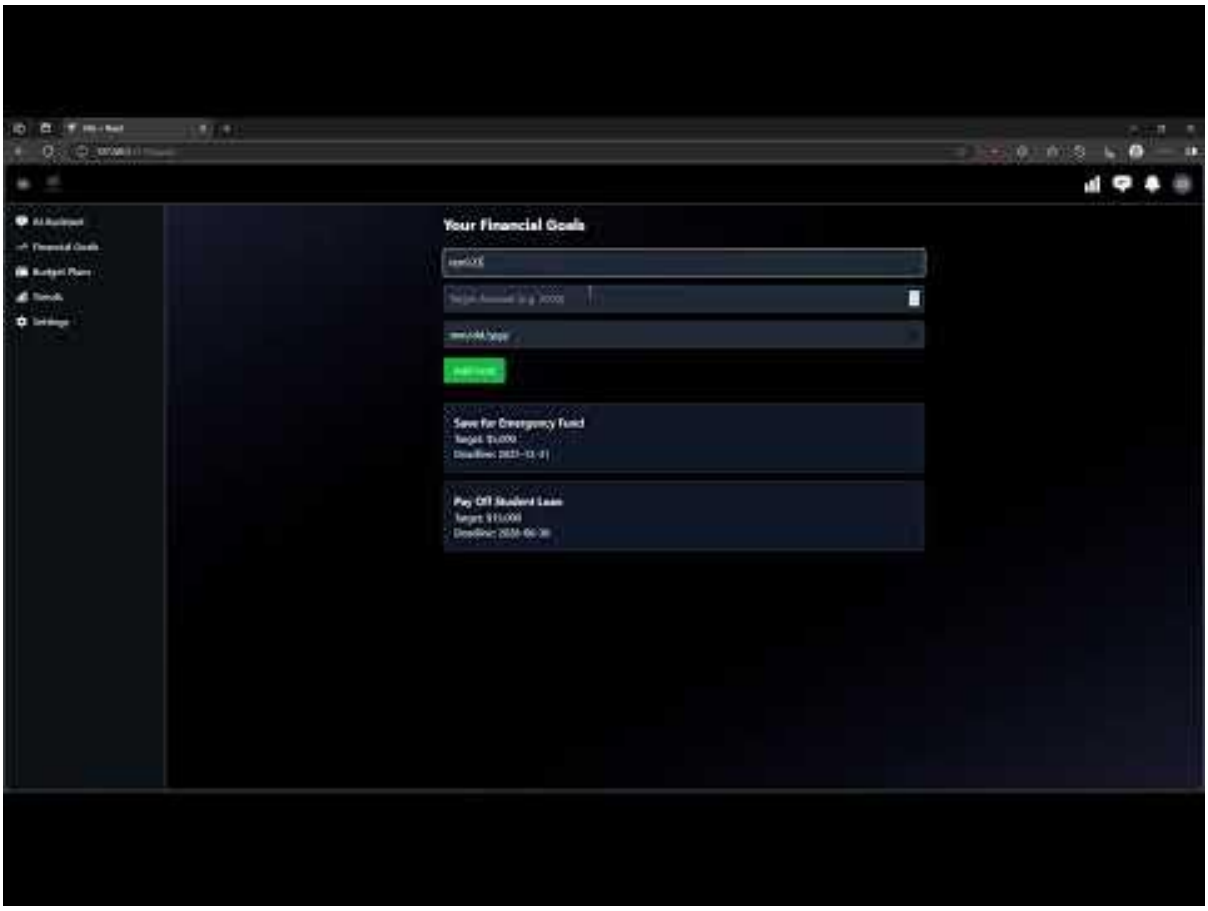


This is a simple settings page where the user can change basic account information, display preferences, as well as log out, and delete their account.


JD
Profile
Settings
Logout


AI Assistant
Financial Goals
Budget Plans
Trends
Settings

Youtube Video Prototype: https://youtu.be/iktkCm6eEO4



Github Link: JacobH123/CPSC-490-Prototype

References

Addison Wesley - UML Distilled, 3rd Ed - 2003. (n.d.). UAH. Retrieved May 15, 2025, from http://www.cs.uah.edu/~rcoleman/CS307/Announcements/UML%20Distilled.pdf

The Double Diamond template. (n.d.). Mural. Retrieved May 15, 2025, from https://www.mural.co/templates/double-diamond

Gemayel, T. (n.d.). *Information Architecture Diagram*. Figma. Retrieved May 15, 2025, from https://www.figma.com/community/file/1007804018270351087

Nielsen, J. (1994, April 24). *10 Usability Heuristics for User Interface Design - NN/g*. Nielsen Norman Group. Retrieved May 15, 2025, from https://www.nngroup.com/articles/ten-usability-heuristics/

Pikover, J. (n.d.). *The UX Process for Information Architecture*. Toptal. Retrieved May 15, 2025, from https://www.toptal.com/designers/ia/guide-to-information-architecture

Storyboards & Design Thinking. (2024, September 26). DevSquad. Retrieved May 15, 2025, from https://devsquad.com/blog/storyboard-design

UML Activity Diagram Tutorial. (n.d.). Lucidchart. Retrieved May 15, 2025, from https://www.lucidchart.com/pages/uml-activity-diagram

UX Daily: The World's Largest Open-Source UX Design library. (n.d.). The Interaction Design Foundation. Retrieved May 15, 2025, from https://www.interaction-design.org/literature/article/overview