Project Title: Wallee - Personal Finance and Budgeting App

Team Members:

- Emma Bahr ebahr2022@my.fit.edu
- Kyle Gibson kgibson2021@my.fit.edu
- Joshua Cajuste jcajuste2022@my.fit.edu
- Matteo Caruso mcaruso2022@my.fit.edu

Faculty Advisor: Dr. Siddhartha Bhattacharyya sbhattacharyya@fit.edu

Client:

- Doug Gibson — Small Business Owner
- Dr. Siddhartha Bhattacharyya — SWE/CSE Professor & Advisor

Meeting Dates with Client:

- Meeting 1 with Mr. Gibson: January 21, 2026
- Meeting 1 with Dr. Bhattacharyya:

Goal and Motivation:

Our goal is to create an app to assist with financial goals to help save and manage money. Managing personal finances is a growing challenge for students, freelancers and young professionals who face variable income, recurring bills, and limited financial literacy. Existing budgeting tools often lack personalization, real-time updates, or intelligent guidance, leaving users without actionable insights.

Wallee aims to fill this gap by providing a smart, all-in-one finance app that integrates with user's bank accounts, automates expense tracking, delivers bi-weekly reports and includes an AI-powered financial assistant. Our motivation is to empower users to take control of their finances through simplicity, automation, and tailored insights.

Approach (key features of the system):

- User can refine their ability to manage personal finances
    - Users can connect their bank accounts security to track income and expenses in real time
        - Users can view automatically categorized transactions(e.g. Groceries, rent, subscriptions)
        - Users can also create and adjust custom budget categories specific to their needs.
    - Users can set short-term and long-term financial goals, such as saving for tuition, vacations
    - Users can receive automated budget updates after each paycheck, showing exactly how much money is “safe to spend” after recurring bills are covered.
    - Users can compare planned budgets with actual spending across categories.
    - Users can track progress over time, with adjustments suggested if overspending occurs.
        - For instance a student can set a $12,000 savings goal for a 12 month deadline. If they overspend on dining by $100 during the first month, Wallee can recalculate the remaining budget by:
            - Adjusting the goal deadline, increasing the time to reach the goal based on overspending.
            - Adjusting the budget for the remaining time to account for the overspending (ie: decreased spending each month by 100$ / 11).

- User can view personalized financial summaries from the app
    - Users can access bi-weekly reports showing income, bills, discretionary funds, and savings progress.
    - Users can view weekly and monthly summaries broken into tables with budgeted, spent, and remaining amounts.
    - Users can compare budgeted versus actual spending in visual graphs that highlight trends and overspending.
    - Users can view predictive insights, such as “safe-to-spend” numbers or upcoming bill reminders.

- User can receive proactive financial assistance

    - Users can interact with an AI chatbot that explains financial data in simple terms.
    - Users can receive personalized tips, such as saving more in high-income months or adjusting categories to meet goals.
    - Users can access educational content designed to improve financial literacy.
        - For example, a freelancer with irregular income could receive AI-driven alerts recommending when to save more after a high-earning month, or reminders about upcoming tax payments.

Novel Features/functionalities:

Wallee stands out through its unique mix of automated bi-weekly budgeting, AI-driven guidance, tailored user experiences, and intuitive visualization tools. Competing apps like Mint or PocketGuard often present raw data without context, but Wallee recalculates finances with each paycheck, provides personalized insights through its AI assistant, and adapts to the needs of students, freelancers, retirees, and couples alike. Instead of overwhelming users with transaction lists, it delivers clarity through weekly tables and graphs that show budget limits versus actual spending. This combination of proactive automation and user-focused design makes Wallee more inclusive, actionable, and supportive than existing budgeting applications.

Algorithms and Tools: potentially useful algorithms and software tools

- **Backend runtime for APIs & webhooks:**
    - Node.js (TS) for public API/webhooks, Python (FastAPI) for analytics/budgeting workers. Best of both worlds.
- **Budgeting/analytics engine:**
    - Python worker for budgeting/forecasting; expose results via API.
- **Banking integration:**
    - Plaid for US launch; design adapters to add TrueLayer for EU later.
- **Database & data modeling:**
    - PostgreSQL (core ledger, users, budgets, goals)
    - Supa Base Database Engine

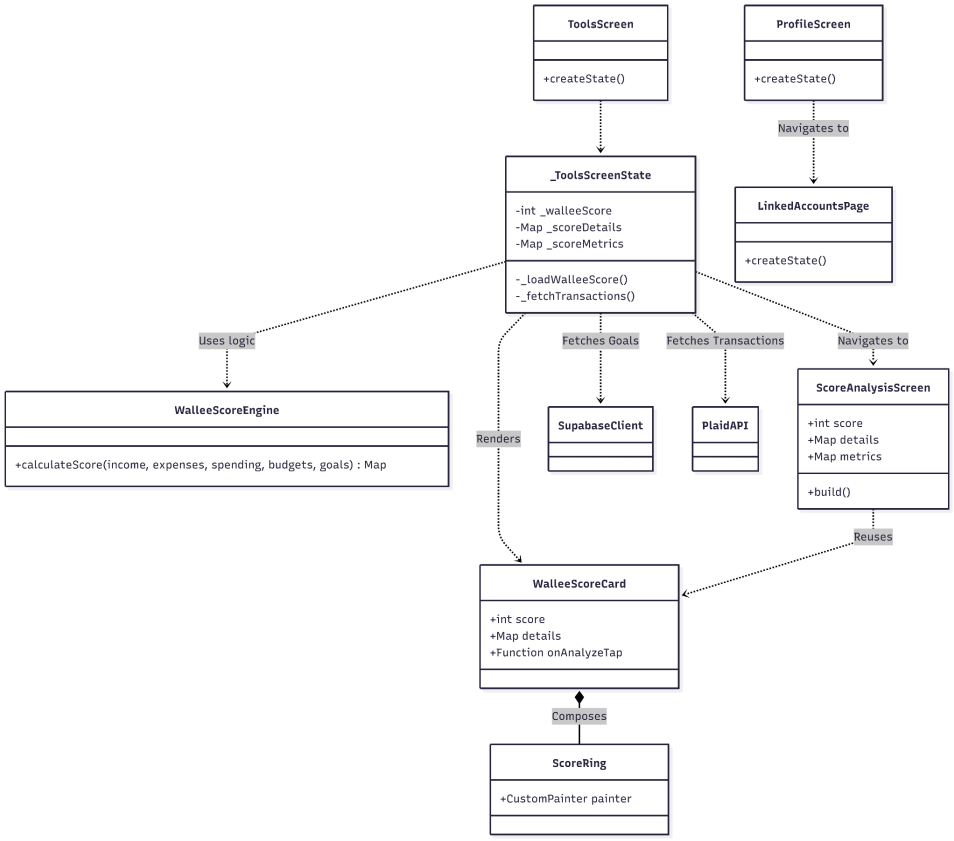
- **Job orchestration & scheduling:**
    - Celery + Redis/SQS for analytics tasks; optional cron to trigger batch runs.
- **AI assistant & NLU:**
    - OpenAI for conversational guidance + spaCy for lightweight extraction
- **API framework & validation:**
    - NestJS for API; FastAPI for analytics service.

Technical Challenges:

- We plan to create an adaptive budgeting engine that will be the bulk of the application using Node.js and python but are not very familiarized with it
- We plan to develop an ethical and reliable AI Assistant using Google AI studio API but we are not very familiar with using API's
- We plan to develop a mobile app, but we are not overly familiar with mobile app development
- We plan to use Flutter for app development, but we are not overly familiar with flutter.
- We plan to add security measures to the application but are not familiar with implementing security precautions to application

Design:

Class Diagram for the Software Design



Architecture Description:

Wallee follows a modular, service-oriented architecture designed to support scalability, security, and clear separation of concerns. The system is centered around a mobile-first experience, with backend services handling data processing, analytics, and integrations with third-party platforms.

The mobile client, built using Flutter, serves as the primary user interface. It handles user authentication, data visualization, and user interactions such as setting budgets, viewing reports, and chatting with the AI assistant. The client communicates exclusively with backend services through secure HTTPS REST APIs, ensuring sensitive financial data is never processed directly on the device.

The backend API layer, implemented using Node.js with NestJS, acts as the central gateway for all client requests. This layer manages authentication, authorization, request validation, and communication with external services such as Plaid. It

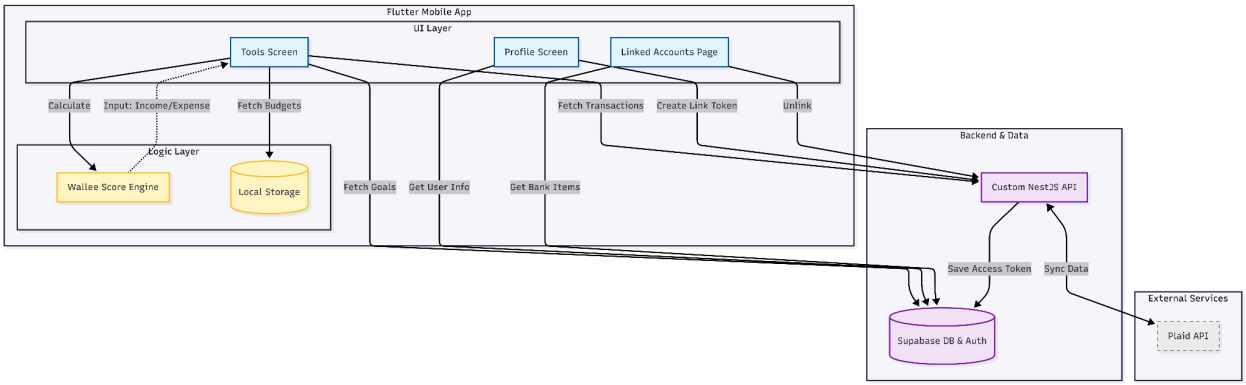
exposes endpoints for retrieving transactions, budgets, goals, and reports while enforcing security and rate limits.

The analytics and budgeting engine, implemented as a separate Python FastAPI service, performs computationally intensive tasks such as budget recalculations, forecasting, and overspending adjustments. By isolating analytics logic from the main API, the system remains responsive while allowing advanced financial algorithms to evolve independently.

All persistent data is stored in a PostgreSQL database, which maintains a normalized financial ledger including users, transactions, budgets, goals, and historical reports. This centralized data store ensures consistency across features and supports accurate long-term analysis.

Finally, third-party integrations provide essential functionality. Plaid supplies secure access to bank transaction data, while the OpenAI API powers the conversational financial assistant. These integrations are abstracted behind backend services to allow future replacements or extensions without impacting the client application.

Block System Architecture Diagram:



Evaluation:

System success will be evaluated using:

- Accuracy: Correct categorization and budgeting calculations.
- Reliability: Consistent performance across repeated budget recalculations.
- Performance: Timely updates after transactions and paychecks.
- User Surveys: Ratings (1–5) on usability, clarity, and perceived usefulness of features.

Progress Summary:

Module / Feature	Completion %	To Do
GUI	80%	The overall theme is set, needs to finish the little details and make sure the whole thing is following the guidelines
Backend APIs	90%	Last changes and testing
Budget Engine	90%	Last changes and testing
Database	60%	Need to finish implementation of bank accounts and profile pictures.

Milestone 4 (Feb. 23rd):

- Implement and test all core features
- Implement and test a calendar and time feature
- Implement and test core features
- Implement and test all new database features

Milestone 5 (Mar. 30th):

- Have the app functional.
- UI and bug testing will be needed
- Start working on apple integration if able to
- Conduct evaluation and analyze results
- Create poster for Senior Design Showcase

Milestone 6 (Apr. 20th):

- Have the app fully functional and presentable with all UI and bug fixes implemented.
- Test/demo of the entire system
- Conduct evaluation and analyze results

- Create user/developer manual
- Create demo video

Task Matrix for Milestone 4:

Task	Emma	Matteo	Kyle	Joshua
Hide api keys	**0%**	**0%**	**0%**	**100%**
Link budget and goal pages	**0%**	**10%**	**90%**	**0%**
Finish user profile page and settings page	**30%**	**30%**	**30%**	**10%**
Finish user reports	**0%**	**100%**	**0%**	**0%**
Finish implementing AI	**0%**	**20%**	**0%**	**80%**
Start implementing time and calendar	**0%**	**0%**	**50%**	**0%**
Connect bank accounts in the database	**100%**	**0%**	**0%**	**0%**

Approval from Faculty Advisor:

- I have discussed with the team and approved this project plan. I will evaluate the progress and assign a grade for each of the three milestones.
- Signature: _____ Date: _____