High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **FinSight - Personal Finance Dashboard**. FinSight is a modern, open-source web application that visualizes personal spending, budgets, and financial goals using interactive charts and summaries. The solution leverages React, advanced CSS, and dynamic data rendering to deliver an engaging user experience.

---

1. System Architecture Overview

Architecture Description:
FinSight is a single-page application (SPA) built with React. The frontend is modular, with components for data input, visualization, and user management. Data is managed client-side, with potential for future backend/API integration.

Module/Component	Role/Responsibility
App Shell	Main layout, routing, and global state management
Dashboard	Aggregates and displays financial summaries and charts
Data Input	User interfaces for entering transactions, budgets, etc.
Visualization	Renders interactive charts (spending, budgets, goals)
User Profile	Manages user preferences and settings
Data Store	Manages client-side data (state, local storage)

---

2. Component Interactions

Source Component	Target Component	Interaction Description
Data Input	Data Store	User submits data, which updates the data store
Data Store	Dashboard	Dashboard subscribes to data changes for live updates
Dashboard	Visualization	Passes processed data to visualization components
User Profile	App Shell	Updates global settings and preferences

Sequence Flow:

1. User enters data via Data Input.
2. Data Store updates and notifies Dashboard.
3. Dashboard processes and forwards data to Visualization.
4. User Profile changes update App Shell and propagate as needed.

---

3. Data Flow Overview

Data Source	Data Destination	Data Type/Flow Description
User Input	Data Store	Transactions, budgets, goals entered by user
Data Store	Dashboard	Aggregated and filtered financial data
Dashboard	Visualization	Chart-ready datasets
User Profile	App Shell	User settings/preferences

---

4. Technology Stack

Layer/Area	Technology/Frameworks
UI Framework	React (Functional Components, Hooks)
Styling	Tailwind CSS, Flexbox, CSS Grid
Data Management	React Context, useState/useReducer
Charts	Open-source JS charting library (e.g., Chart.js, Recharts)
State Persistence	Local Storage (browser)
Tooling	JavaScript (ES6+), HTML5, npm

---

5. Scalability & Reliability

- **Scalability:** Modular React components enable easy feature extension and future backend/API integration.
- **Reliability:** Client-side state management with local storage ensures data persistence across sessions.
- **Security:** Sensitive data is stored locally; no external data transmission in current scope. Future enhancements may include authentication and secure API integration.

End of Document