Spend Smart

Software Architecture

2026/03/24

Project Description

SpendSmart, a cloud-based web application, lets university students like you record their expenses quickly and conveniently. SpendSmart also analyses your expenses and gives you ideas for saving money. SpendSmart aims to make expense tracking easier for all university students.

Project Deliverables

An MVP will require the following functionalities:

1. A secure authentication system allowing users to register and manage profiles.
2. Add collaborators to specific expenses or groups, ensuring shared financial data is updated and reflected across all relevant user sessions in real time.
3. Connect to a banking system to query expenses (eg, https://github.com/OpenBankProject/OBP-API, https://github.com/sholanejo/BankApiNodeJs)
4. A search engine providing form-based queries across textual, numerical, and temporal fields, with categorical and state-based filters
5. A visualisation dashboard that aggregates expense data over custom date ranges, presenting findings in tabular formats with multi-category comparison views.
6. An alerting logic that utilises visual indicators to signal shifts in spending behaviour, comparing current expenditure against historical benchmarks to highlight increases or decreases.
7. A state-tracking feature that monitors expenses against user-defined budget thresholds, providing a real-time view of remaining funds.

Quality Attributes

Below are the main quality attributes the application will focus on. In addition to these, implement strong privacy and security measures to protect user data.

1. Interoperability: The system's interoperability is evaluated by its ability to interface with external banking APIs (e.g., Open Bank Project) to ingest transaction data. Evaluation involves testing the integrity of the API request, ensuring that external transaction metadata is accurately mapped and normalised into the platform's internal schema. Evidence must show that the system can handle different data formats from various banking providers without requiring a rewrite of the core system.
2. Reliability: The system is considered reliable if the financial database is replicated across multiple instances, ensuring that individual and group ledger entries are synced correctly. Evaluation involves demonstrating that collaborators view the most up-to-date data and that the system can resolve concurrent update conflicts during group expense edits.

3. Security: The application will be evaluated on the integrity of its data-access policies, specifically ensuring that private expenses remain invisible to non-collaborators.