# SMARTSPEND
## THE EMOTION-AWARE CASH-FLOW COACH

**Course Name: Capstone 2025**
**Team Name: Foxtrot**
**Date of Submission: December 12, 2025**

**Team Members:**

**Sarvesh Patil**
**811348253**
**spatil8@kent.edu**
**Master's in Computer Science**

**Sandeep Enamandala**
**811350293**
**senamand@kent.edu**
**Master's in Computer Science**

**Prajwal Devaraj**
**811351381**
**pdevaraj@kent.edu**
**Master's in Computer Science**

**Akshara Reddy Nookala**
**811330459**
**anookala@kent.edu**
**Master's in Computer Science**

**Meghamala Samala**
**811362078**
**msamala1@kent.edu**
**Master's in Artificial Intelligence**

**Abhijeet Yalamanchili**
**811354816**
**ayalaman@kent.edu**
**Master's in Computer Science**

**Prasadam Manoj Kumar**
**811374110**
**mprasada@kent.edu**
**Master's in Data Science**

## 1. EXECUTIVE SUMMARY
### 1.1 OVERVIEW

SmartSpend is an intelligent personal finance management system designed to help users understand, track, and optimize their spending behavior. Unlike traditional budgeting applications that focus only on recording income and expenses, SmartSpend combines financial analytics**,** behavioral insights, and machine-learning predictions to provide a deeper understanding of how users spend money and why they make certain financial decisions.

The system enables users to securely sign up and manage their daily financial activities through an interactive dashboard that displays burn rate**,** runway**,** days left**,** insights**,** and upcoming bills. Users can categorize transactions using the NWG model (Need–Want–Guilt) and Mood tagging, enabling smarter behavioral analysis. The platform also includes modules for bills management**,** goal setting**,** achievements tracking**,** and ML-driven spending predictions, making SmartSpend an all-in-one financial wellness tool.

With a robust Flask–MySQL backend, React-based frontend, and integrated machine-learning models, SmartSpend offers a personalized financial experience that adapts to user behavior and provides actionable suggestions to improve financial health.

## 1.2 GOALS AND OBJECTIVES

The primary goal of SmartSpend is to build an intelligent, user-centric financial management system that not only tracks spending but also provides meaningful insights into user behavior. The system aims to bridge the gap between traditional expense trackers and modern ML-driven financial tools by offering predictive analytics, emotional-spending detection, and personalized financial guidance.

### PROJECT GOALS

- To develop a secure and scalable platform where users can manage all aspects of their personal finances in one place.
- To provide users with deeper visibility into their spending patterns through analytics such as burn rate, runway, NWG classification, and mood tagging.
- To incorporate machine learning models that forecast future spending behavior and identify risk indicators.
- To simplify financial planning by automating recurring bill reminders, goal tracking, and achievements.

### OBJECTIVES

- Implement authenticated user flows using JWT-based login/signup with secure backend APIs.
- Enable accurate tracking of income and expenses, with support for categories, merchant names, timestamps, and behavioral tagging.
- Calculate advanced metrics such as daily burn, days left, power-save runway, and next 7-day burn projection.
- Provide ML-driven insights, including emotional spending detection, mood correlations, spending spikes, and risk tier predictions (T1/T2/T3).
- Support recurring bills and reminders, helping users plan upcoming payments and avoid missed dues.
- Allow users to set financial goals and dynamically update runway estimates based on goal horizons.
- Present data visually through an intuitive dashboard featuring charts, tables, and interactive UI components.

- Ensure backend reliability and scalability through a modular Flask API architecture and MySQL database integration.
- Enable seamless integration between frontend, backend, and ML layers for real-time analytics.

### 1.3 MAJOR DELIVERABLES AND FEATURES

The SmartSpend system delivers a complete, end-to-end personal finance platform with ML-driven intelligence, intuitive user interaction, and robust backend support. The following are the major deliverables and core features achieved throughout the project.

1. **User Authentication and Security (JWT-Based)**
   - Secure signup and login flows.
   - Access and refresh token system.
   - Role-based API access ("onboarding" vs. "app" scopes).
   - Automatic token expiration and protection for all financial routes.
2. **Complete Transaction Management**
   - Add, edit, and delete transactions (income and expenses).
   - Categorization into Need / Want / Guilt (NWG).
   - Mood tagging for emotional-spending insights.
   - Merchant, category, timestamp, and late-night detection.
   - Local and server-side validation.
3. **Smart Dashboard**
   - Burn Rate (30-day average)
   - Days Left (Runway) based on spending trends
   - Power-Save Runway for reduced spending mode
   - Next 7 Days Burn prediction
   - NWG breakdown chart
   - Daily burn series graph
   - Achievements preview and insight alerts
4. **Bills & Recurring Payment Manager**
   - Add recurring bills with custom cadences (weekly, bi-weekly, monthly).

- Auto-calculated next due dates.
- Status tracking: active, paid, skipped.
- 7-day upcoming bills preview.
- Alerts for approaching due dates.

**5. Goals & Runway Tracking**

- Set short-term and long-term financial goals.
- Automatic runway calculation based on goal days.
- Real-time updates when transactions change.
- Visual progress indicators and target tracking.

**6. ML-Driven Insights**

Tiered predictive intelligence:

- **Tier-1:** Simple statistical alerts (late-night spending, wants > 50%)
- **Tier-2:** Pattern detection and behavior-based insights
- **Tier-3:** Machine Learning models:
  - Burn rate forecasting
  - Emotional spending detection
  - Category-based risk predictions

Models integrated via ml routes with automated preprocessing.

**7. Responsive Frontend (React + Vite)**

- Modular component structure.
- Clean UI with cards, charts, modals, and animated transitions.
- Dashboard optimized for desktop.
- Smooth onboarding flow with guided steps.

**8. Backend REST API (Flask)**

- Modular blueprint architecture.
- SQLAlchemy ORM with MySQL database.
- Business logic for burn rate, NWG, insights, goals, and bills.
- CORS-enabled for frontend integration.
- Timezone-aware calculations and UTC-safe timestamps.

**9. Deployment Ready Structure**

- Gunicorn entrypoint for production servers.
- Environment-based configuration file.
- Fully container-friendly structure (backend + ML + frontend).

The above deliverables collectively create a powerful AI-enabled financial management system capable of tracking behavior, predicting risks, and guiding users toward healthier financial habits.

### 1.4 SUMMARY OF ACCOMPLISHMENTS AND CHALLENGES

Throughout the development of this project, several important accomplishments were achieved while also encountering meaningful technical challenges that shaped the final outcome. A complete full-stack system was successfully built, integrating a secure JWT-based authentication flow, a modular Flask backend, and a responsive React frontend. The project delivered accurate financial analytics such as burn rate, runway estimation, NWG classification, and next-7-day spending projections through careful handling of timestamps, timezone conversions, and data filtering to ensure reliable results.

A significant accomplishment was the integration of machine learning models that generate user insights, predict risk levels, and identify emotional or impulsive spending behaviors. Additionally, robust features such as transaction management, mood and NWG tagging, recurring bill tracking, financial goals, and achievement badges contributed to a complete user experience. However, the project also faced several challenges, including resolving inconsistencies in ML model versions, handling local versus UTC date logic, managing new-user edge cases for calculations, and overcoming deployment limitations due to restricted server permissions. Despite these obstacles, the system was thoroughly tested with both automated scripts and manual validations, resulting in a stable, user-focused platform that demonstrates strong technical depth and practical usability.

## 2. PROBLEM STATEMENT

### 2.1 WHAT PROBLEM OR NEED DOES THE PROJECT ADDRESS?

In today's digital era, most personal finance applications focus solely on recording expenses and displaying balances, but they fail to explain why individuals overspend. Traditional budgeting tools do not capture emotional triggers, behavioral patterns, or psychological factors behind financial decisions. As a result, users continue to repeat harmful spending habits despite having access to charts, lists, and monthly summaries.

Young adults, students, and working professionals often experience:

- Impulse buying driven by stress, mood swings, or peer influence
- Lack of awareness of their burn rate and long-term financial runway

- Difficulty separating essential spending from emotional or unnecessary purchases
- Poor bill management, leading to late fees and financial uncertainty
- No personalized insights or predictions that help modify daily behavior

Furthermore, existing apps rarely provide machine learning driven forecasting, behavior-based categorization, or real-time runway predictions. The absence of an intelligent assistant that can predict spending risks, analyze emotional consumption, and guide users toward healthier financial habits creates a significant gap in the personal finance ecosystem.

SmartSpend addresses this need by combining budgeting, behavioral analysis, and machine learning predictions to offer users a deeper understanding of their financial actions, enabling them to make informed and emotionally aware spending decisions.

## 2.2 WHO ARE THE STAKEHOLDERS OR INTENDED USERS?

The SmartSpend system is designed to support a wide range of users and stakeholders who are directly or indirectly impacted by financial decision-making and personal budgeting behavior.

**Primary Users**

1. **Students & Young Adult**

   Individuals who are beginning to manage their own finances, often facing impulsive spending habits, lack of budgeting awareness, and irregular income sources.

2. **Working Professionals**

   Users who want to track expenses, control monthly spending, understand emotional triggers, and maintain financial stability while managing bills and savings goals.

3. **Individuals struggling with emotional or impulse spending**

   People who seek behavioral guidance, insights, and personalized recommendations to build healthier spending habits.

**Secondary Stakeholders**

1. **Financial Advisors or Counselors**

   Professionals who may use SmartSpend's analytics and mood-based spending patterns to support clients in financial planning or behavior correction.

2. **Researchers & Data Scientists**

   Stakeholders exploring links between emotional states, behavioral economics, and spending patterns using anonymized data and ML models.

3. **App Developers & Product Teams**

Teams responsible for improving the SmartSpend platform, integrating new features, enhancing ML predictions, and maintaining system performance.

**Tertiary Stakeholders**

1. **Banks or FinTech Companies (Future Scope)**

   Organizations that may integrate SmartSpend insights for budgeting assistance, credit risk assessment, or customer engagement.

2. **Mental Health and Wellness Communities**

   Groups interested in understanding how emotions influence spending behavior and how data-driven insights can support healthier lifestyle choices.

## 2.3 WHY IS THE PROBLEM IMPORTANT OR INTERESTING?

Understanding and managing personal finances has become increasingly challenging in today's fast-paced, emotionally driven environment. Many individuals struggle not because they lack income, but because they lack visibility into how, why, and when they spend money. This makes the problem both important and intellectually interesting for several reasons:

1. **Emotional Spending Is a Growing Behavioural Issue**

Research shows that a significant portion of daily purchases is influenced by emotions such as stress, boredom, guilt, or impulsivity. Traditional budgeting tools only track spending they do not explain the behaviour behind it. SmartSpend addresses this gap by integrating emotional tagging, mood analytics, and ML-driven insights.

2. **Lack of Predictive Tools for Burn Rate and Runway**

Most budgeting apps provide static charts but do not forecast how long a user's money will last. Predicting burn rate, runway days, and future spending risk supports better financial planning and prevents unexpected shortages. This makes the problem fundamentally interesting from a data science and behavioral analytics standpoint.

3. **Managing Bills, Goals, and Cash Flow Is Complex**

People often juggle recurring bills, savings plans, and changing monthly expenses. Without a unified system, this leads to missed payments or poor financial discipline. SmartSpend provides a single platform to handle bills, goals, and projections reducing stress and improving financial clarity.

4. **Real-World Impact on Financial Well-Being**

The problem matters because it directly affects people's:

- Savings
- Mental peace

- Ability to plan
- Overall financial health

A tool like SmartSpend can meaningfully improve day-to-day decision-making, helping users build long-term financial discipline.

## 3. SYSTEM OVERVIEW

The solution unifies four main layers Frontend, Backend, Database, and Machine Learning Models with a Deployment layer for real-world accessibility. The architecture is modular so each component can function independently yet collaborate seamlessly through well-defined APIs and data flows.

### 3.1 HIGH-LEVEL SYSTEM ARCHITECTURE

The architecture of SmartSpend follows a modular, multi-layered design that integrates the frontend, backend, machine-learning engine, and database into a seamless end-to-end workflow. Each layer interacts through well-defined interfaces, ensuring scalability, maintainability, and clear separation of responsibilities as shown in Fig 3.1.

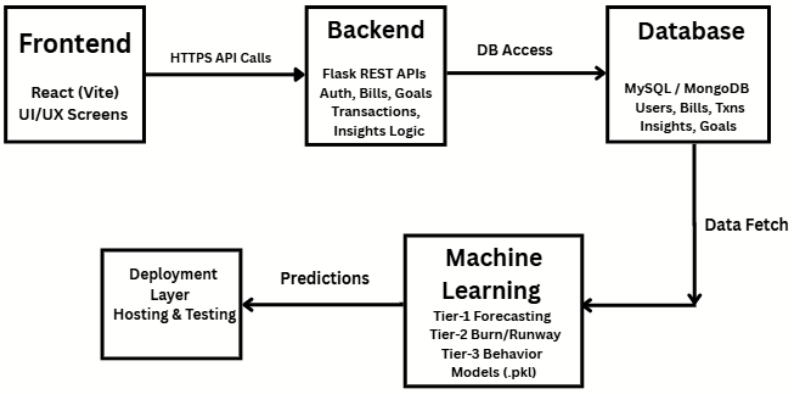


Fig 3.1. System Architecture

#### 3.1.1 Frontend Layer

The frontend represents the user interaction layer of SmartSpend. Built using React with Vite, it provides all UI/UX components including onboarding screens, login, dashboards, transaction views, bill management, goal tracking, and insights visualization. All user actions such as adding expenses, editing bills, or viewing predictions are sent to the backend through secure HTTPS REST API calls. The frontend is lightweight, fast, and optimized for responsive performance, making it suitable for web-based financial applications.

#### 3.1.2 Backend Layer

The backend serves as the central processing engine of the system. Developed using Flask, it exposes multiple RESTful endpoints that manage authentication, bill calculations, goal tracking, transaction classification, NWG logic, insights computation, and orchestration of machine-learning predictions. The backend acts as the mediator between the frontend, the ML models, and the database. It validates user requests, processes business logic, fetches required data, and returns structured JSON responses. This modular API design allows independent scaling of frontend and backend components.

### 3.1.3 Database Layer

SmartSpend uses MySQL to persist user data such as profiles, transactions, recurring bills, goals, and insights history. The backend communicates with the database through optimized SQL queries, ensuring efficient retrieval of financial records and maintaining referential integrity. The database also plays a critical role in the ML workflow by supplying historical data for training, trend analysis, and prediction updates. Its schema is designed to support both transactional consistency and analytical workloads.

### 3.1.4 Machine Learning Layer (Tier-1, Tier-2, Tier-3 Models)

The machine-learning subsystem powers SmartSpend's intelligence. It contains three categories of models:

- Tier-1: Expense forecasting
- Tier-2: Burn-rate and runway prediction
- Tier-3: Behavioral pattern and NWG classification

These models are stored as serialized .pkl files and are invoked by the backend on demand. When predictions are requested, the backend sends user data to the ML layer, which performs preprocessing, runs the model, and returns predicted burn rate, runway, or behavioral risk levels. This ML pipeline as shown in Fig 3.2 ensures that insights are updated dynamically based on user financial activity.

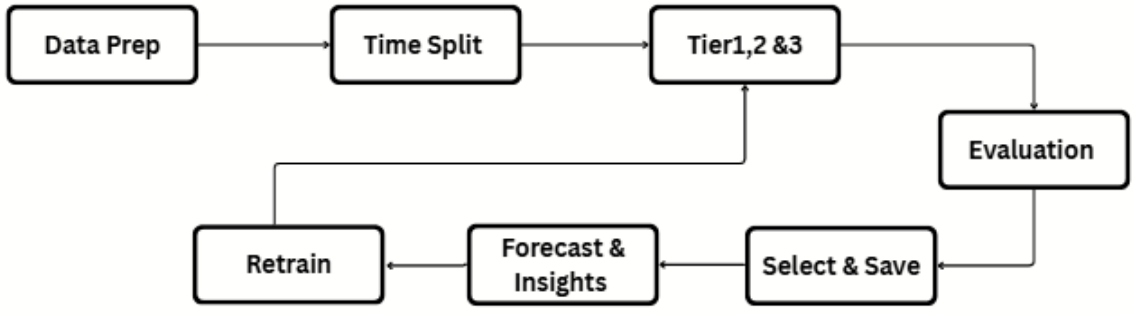


Fig 3.2. ML pipeline

### 3.1.5 Deployment Layer

The deployment layer of SmartSpend is entirely managed through GitHub, making the system easy to maintain, update, and distribute. The complete project including the frontend, backend,

and machine-learning models is hosted in a GitHub repository where version control, collaboration, and automated workflows take place. Developers can push updates, manage branches, and track issues directly through GitHub's integrated ecosystem. The repository also supports GitHub Actions, enabling automated testing and continuous integration whenever new code is committed.

In the complete architecture, the frontend interacts with the backend through HTTPS API calls. The backend processes these requests, fetches data from the database, calls the ML models when predictions are needed, and returns a combined response. The ML models retrieve historical data via the backend and provide predictive insights such as expected expenses, runway days left, or emotional spending risks. The deployment environment ensures that all these layers function cohesively and are accessible to users.

## 3.2 TECHNOLOGIES USED

The SmartSpend system leverages a modern and scalable technology stack designed to support secure authentication, efficient data processing, real-time insights, and an interactive user interface. The selection of technologies was based on performance, maintainability, and compatibility with machine-learning workflows. The following subsections summarize the major technologies used in the project.

### 3.2.1 Backend Technologies

The backend is built using Python, chosen for its extensive ecosystem and compatibility with machine-learning libraries.

- **Flask Framework**: A lightweight yet powerful web framework used to build RESTful API endpoints for authentication, transactions, predictions, and dashboard analytics.
- **Flask-SQLAlchemy**: Manages database interactions using an ORM layer, enabling clean, modular query definitions.
- **PyMySQL**: Provides a direct connection between Flask and the MySQL database.
- **Machine Learning Libraries**:
    - scikit-learn for burn-rate prediction, runway estimation, and classification tasks.
    - pandas & NumPy for preprocessing and analytics computations.
- **JWT** Authentication: Implemented via PyJWT to handle secure login, signup, access tokens, and refresh tokens.

### 3.2.2 Frontend Technologies

The frontend is developed using a React and TypeScript stack for modular, maintainable, and scalable UI development.

- **React:** Provides a component-based architecture for pages such as Dashboard, Transactions, Bills, Goals, and Insights.

- **TypeScript:** Enforces type safety and improves code reliability.
- **Vite:** A fast development server and bundler used for optimized production builds.
- **Recharts:** Used for visualization of burn rate trends, NWG charts, insights, and analytics.
- **Custom UI Components:** Cards, Progress Bars, Modals, and Forms built for consistent design and improved user experience.

### 3.2.3 Database Layer

The system uses MySQL as the primary relational database due to its reliability and structured schema support.

Key stored data includes:

- User profiles
- Transactions with NWG & Mood tags
- Bills and recurring categories
- Goals and runway preferences
- Insights and historical ML predictions

MySQL ensures fast queries, relational integrity, and efficient indexing for analytics operations.

### 3.2.4 Machine Learning Components

SmartSpend integrates Tier-1, Tier-2, and Tier-3 predictive models to estimate user spending behavior.

Technologies used include:

- scikit-learn linear regression
- Custom preprocessing pipelines built with pandas
- Pickle-based model persistence for loading pre-trained models during runtime

These models support:

- Daily burn estimation
- Next-7-days projected expenses
- Mood/NWG classification
- Runway forecasting

### 3.2.5 Version Control & Deployment

GitHub serves as the project's central repository and collaboration platform.

- Manages frontend, backend, and ML folders in a structured repository.
- Enables continuous development using branches and pull requests.
- GitHub Codespaces provides a cloud-based development environment.
- Automates testing and build processes through GitHub Actions.

#### 3.2.6 Development & Testing Tools

- Postman / cURL for testing API endpoints.
- VS Code for integrated development.
- GitHub Codespaces for cloud execution of both frontend and backend.
- Debugging tools in Flask and React DevTools for UI inspection.
- Automated test scripts for signup, login, transactions, bills, and goals.

### 3.3 OVERVIEW OF SUBSYSTEMS OR MODULES

The SmartSpend application is organized into several interconnected subsystems, each responsible for core functionalities such as authentication, financial tracking, predictions, and visualization. This modular architecture ensures scalability, maintainability, and clear separation of concerns across backend, frontend, and machine-learning components. The major subsystems are described below.

#### 3.3.1 Authentication & User Management Module

This module handles all user-related operations including registration, login, onboarding progress, and secure access control.

Key responsibilities include:

- User Signup and Login APIs
- JWT-based authentication (access + refresh tokens)
- Onboarding steps (Balance → Cadence → Bills → Summary)
- Managing user profile settings and notification preferences

This module ensures secure, role-based access and maintains user state across sessions.

#### 3.3.2 Transactions & Spending Module

The Transactions subsystem manages all daily income and spending entries.

Features include:

- Adding, editing, and deleting transactions
- Categorization using NWG (Need–Want–Guilt) and Mood tagging

- Filtering by type, date range, NWG category, or mood
- Automatic balance adjustments upon every update

This module provides the core financial tracking functionality of the application.

### 3.3.3 Bills & Recurring Expenses Module

This subsystem manages monthly or weekly bills, helping users track mandatory expenses. Responsibilities include:

- Adding and editing recurring bills
- Marking payments as completed
- Categorization of bills (essential, utility, subscription, etc.)
- Syncing bill payments with the transaction system

The module helps users stay organized and prevents missed bill payments.

### 3.3.4 Goals & Runway Management Module

The Goals subsystem helps users plan financial targets and estimate how long their balance will last.

Core functions include:

- Setting financial goals (savings, emergency, planned expenses)
- Updating and tracking goal progress
- Predicting runway (days left) using historical spending and ML-driven burn rate
- Power-save mode with lower burn-rate assumptions

This module transforms raw financial data into actionable personal planning insights.

### 3.3.5 Insights & Analytics Module

The Insights subsystem provides ML-driven interpretations of spending patterns. It generates:

- NWG distribution charts
- Mood-based spending patterns
- ML-powered Next-7-Day burn prediction
- Behavioral alerts (overspending, mood–spend correlation, unusual spikes)

This module enhances user self-awareness and supports healthier spending behavior.

### 3.3.6 Machine Learning Prediction Module

This subsystem runs pre-trained models to deliver real-time predictions.

Models include:

- Tier-1: Daily Burn Rate Prediction
- Tier-2: Mood Classification & Spending Category Forecast
- Tier-3: Next-7-Day Expense Projection

It integrates Python ML libraries, preprocessed datasets, and pickle-stored models into the backend.

### 3.3.7 Dashboard & Visualization Module

The Dashboard module consolidates key financial metrics into a single interactive interface. It displays:

- Current balance
- Average daily burn
- Runway (regular and power-save)
- Projected Next-7-Day spending
- Upcoming bills
- Spending insights

Charts are built using Recharts, providing smooth and modern UI visuals.

### 3.3.8 Database & Persistence Layer

This subsystem manages permanent data storage using MySQL.

It supports tables for:

- Users
- Transactions
- Bills
- Goals
- ML insights
- Runway preferences

SQLAlchemy handles ORM mapping, ensuring consistent data access and relational integrity.

### 3.3.9 Frontend UI/UX Module

The frontend subsystem is built with React + TypeScript and includes:

- Page routing (Login, Dashboard, Bills, Goals, Insights, Transactions)
- Component-level state management

- Form validation and user interaction handling
- Reusable UI components like Cards, Modals, Charts, and Notifications

It provides a seamless and intuitive experience for all user operations.

### 3.4 NOTABLE DESIGN DECISIONS AND TRADE-OFFS

Throughout the development of SmartSpend, multiple architectural and technical decisions were made to ensure the system remains scalable, secure, and easy to maintain. Each decision required balancing benefits with trade-offs in performance, complexity, and development speed. The table below summarizes the key design decisions and the reasoning behind them.

| Design Decision | Reason | Trade-offs |
|---|---|---|
| Modular Backend Architecture (Blueprints) | Clear separation of concerns; easier debugging; maintainability; independent module testing. | Slightly more setup complexity; requires coordination between modules (e.g., bills + transactions). |
| Using Flask Instead of FastAPI | Lightweight, flexible, easy integration with ML models; faster prototyping. | Must manually implement structure and security; fewer built-in tools compared to Django. |
| JWT-Based Authentication | Stateless, scalable, secure; works smoothly with React frontend. | Token revocation is difficult; careful handling needed for token expiration. |
| MySQL Relational Database | Strong relational integrity; ideal for financial data; easy analytics queries. | Harder to scale horizontally; schema migrations can be time-consuming. |
| Separate ML Prediction Layer | Models can be updated independently; clean separation from core business logic. | Loading models at runtime increases startup time; version mismatch warnings with pickle. |
| React + TypeScript Frontend | Type-safety reduces UI bugs; reusable components; smooth API integration. | Higher learning curve; complex build process. |
| Vite for Frontend Build | Faster development (HMR); optimized production builds. | Some libraries require additional configuration. |
| Real-time Dashboard Calculations | Provides accurate, dynamic financial insights (burn rate, runway, next-7-day burn). | Increases backend computation; requires data consistency. |

| | | |
|---|---|---|
| GitHub Deployment | Free academic deployment; easy collaboration and updates. | Not production-level; manual configuration of ports & runtime. |

Table 3.1. Notable Design Decisions And Trade-Offs

## 4. DEVELOPMENT PROCESS

### 4.1 TIMELINE OF MAJOR PHASES AND MILESTONES

The project timeline spanned August to December, progressing through distinct development phases. Each phase contributed incremental functionality to the final system.

**August – Project Initiation & Requirement Analysis**

- Identified the core problem of financial mismanagement due to lack of insights.
- Defined project scope, features (JWT login, dashboard, NWG, bills, goals, ML predictions).
- Conducted initial research on financial planning tools and user needs.
- Prepared high-level architecture and technology stack selection.

**September – Backend Foundation & Database Setup**

- Designed MySQL database schema for users, transactions, bills, goals, insights.
- Implemented core backend modules:
  - Authentication (JWT)
  - User onboarding
  - Transactions API
  - Bills & recurring payments API
- Created environment configurations and testing endpoints.
- Integrated SQLAlchemy ORM and error-handling framework.

**October – Frontend Development & UI System**

- Built React components (Login, Dashboard, Transactions, Bills, Goals).
- Integrated TypeScript for type safety and stable UI logic.
- Connected frontend API calls using Axios.
- Implemented NWG tagging, mood tagging, filters, modals, and progress displays.
- Developed charts for burn rate, runway, NWG distribution, and achievements.

**November – Machine Learning Integration & Advanced Features**

- Integrated ML models (Tier-1, Tier-2, Tier-3) for spending predictions.
- Added insights engine for alerts and recommendations.
- Implemented next–7-day burn prediction in dashboard.
- Performed end-to-end connection between backend and ML layer.
- Conducted debugging and optimization of API responses.

**December – Testing, Deployment & Documentation**

- Ran automated backend tests (signup, login, expenses, bills, goals, ML predictions).
- Performed integration testing between frontend and backend.
- Deployed via GitHub environment.
- Completed system documentation, presentation slides, and final project report.

## 4.2 TEAM WORKFLOW

The SmartSpend project followed a hybrid Agile workflow, combining elements of Scrum and Kanban to support continuous development, flexibility, and efficient task management. This mixed approach allowed the team to adapt to academic schedules while maintaining steady project progress.

**Sprint-Based Scrum Structure**

The team organized development into bi-weekly sprints, each including:

- **Sprint Planning:** Setting goals, prioritizing features, and assigning tasks (backend, frontend, ML, testing).
- **Development Phase:** Members independently worked on assigned tasks following a branch-per-feature Git workflow.
- **Sprint Review:** Completed modules were demonstrated to the team, such as authentication, dashboard burn calculations, ML insights, and bills manager.
- **Sprint Retrospective:** The team reflected on challenges, workflow improvements, and next sprint adjustments.

This Scrum-based cycle ensured regular progress and frequent checkpoints.

In addition to Scrum sprints, the SmartSpend team relied heavily on a Kanban workflow to manage daily development activities with clarity and constant visibility. Since team members were working on different modules backend APIs, frontend UI, machine learning models, and database integration Kanban helped maintain smooth progress without blocking one another.

**Task Management and Code Collaboration**

To maintain clarity and avoid conflicts, the team used:

- GitHub Issues to break down each feature into manageable tasks
- Branch-per-feature workflow for clean version control
- Pull Requests with peer review to maintain code quality
- Frequent integration testing, especially during ML + backend and backend + frontend linking

This workflow ensured smooth collaboration and minimized integration conflicts.

**Continuous Feedback Loop**

The team followed an iterative development philosophy:

- Features were built in small increments
- Backend endpoints were tested using automated scripts
- Frontend components were updated after backend validation
- ML models were retrained and re-integrated based on test results
- Regular short discussions (WhatsApp and Discord) resolved quickly

This helped the project evolve steadily from prototype to final working system.

**Communication Rhythm**

To maintain alignment:

- Weekly team meetings were held to review sprint progress
- Daily informal check-ins in group chats helped track micro-tasks
- Shared documentation (API list, architecture diagrams, ML notes) ensured everyone stayed updated

This consistent communication supported smooth collaboration despite varied schedules.

## 4.3 ROLES AND RESPONSIBILITIES

The SmartSpend project was developed by a coordinated team of seven members, each contributing to core areas such as machine learning, backend development, frontend interface design, documentation, and integration workflows. Responsibilities were assigned based on individual expertise, ensuring efficient progress across all modules.

1. **Sarvesh Patil**
   - Developed ML models (Tier-1/2/3 predictors, insights model).
   - Worked on emotional-spend detection and pattern recognition.
   - Co-led planning, task assignment, and technical decision-making.

**2. Sandeep Enamandala**

- Implemented JWT-based authentication (Signup/Login/Refresh).
- Designed backend architecture (Flask) and core user models.
- Ensured database security, indexing, and optimized queries.
- Worked with on backend API connectivity.

**3. Prajwal Devaraj**

- Designed backend architecture (MySQL) and core database schema.
- Developed connection with Dashboard: Burn Rate, Runway, Power-Save, Next-7-Days projection.
- Integrated ML predictions into backend API.
- Co-managed team workflow, progress tracking, and coordination.

**4. Akshara Reddy Nookala**

- Designed onboarding UI flow, profile settings, and notification pages.
- Worked on consistent frontend layouts.
- Co-created project documentation and PowerPoint presentation.

**5. Meghamala Samala**

- Co-developed ML pipeline completely.
- Worked on feature engineering, preprocessing, and evaluation.
- Supported insights generation and testing ML outputs on frontend.

**6. Abhijeet Yalamanchili**

- Built Transactions module (Filters, mood/NWG tagging).
- Implemented Bills Manager, recurring payment logic, and upcoming due alerts.
- Integrated UI with backend endpoints and ensured state synchronization.

**7. Prasadam Manoj Kumar**

- Developed Insights, Accumulate and Achievements UI, Recharts-based visualizations.
- Assisted in styling and aligning UI across modules.
- Helped validate frontend presentation with outputs.

## 4.4 TOOLS USED FOR COLLABORATION AND SOURCE CONTROL

The SmartSpend development team relied on a combination of industry-standard tools to collaboratively build, track, and manage the entire project from August to December. These tools supported version control, communication, task coordination, documentation, and testing workflows.

**Source Control & Version Management**

**GitHub:** GitHub served as the central platform for maintaining the project repository.

It enabled:

- Version control for backend, frontend, and ML code
- Branching & pull-request workflows
- Tracking commits and code reviews
- Issue tracking for bugs and feature requests
- Continuous synchronization across team members

Each module (Frontend, Backend, ML) had dedicated branches, helping avoid merge conflicts and ensuring clean integration cycles.

**Project Management & Workflow Tracking**

**GitHub Projects - Kanban Board**

Kanban boards were used to manage ongoing work through structured lanes. This made daily task prioritization clear and helped monitor progress during weekly sprint reviews. Each card represented a task with assigned owners, due dates, and checklists.

**Communication & Team Coordination**

These platforms supported real-time collaboration via:

- Daily communication
- Quick debugging discussions
- Meeting scheduling
- File sharing (screenshots, logs, test results)

**Google Meet**

Used for weekly sprint calls, milestone reviews, and ML/Backend/Frontend integration discussions.

**Documentation & Shared Resources**

**Google Drive:** All reports, diagrams, screenshots, dataset information, and design decisions were stored in Google Drive.

Google Docs enabled collaborative editing for:

- Project Report
- Presentation Slides
- Requirement Notes
- API Documentation

**Canva - Visual Tools**

Used for: Slide design for the final presentation

**Development Environments & Testing Tools**

**VS Code / GitHub Codespaces**

Primary coding environments for backend, frontend, and ML. GitHub Codespaces provided a consistent cloud-based development environment for the team, reducing setup issues.

**MySQL Workbench**

Used to inspect the database schema, run SQL queries, and verify data consistency.

Together, these tools provided a unified ecosystem for:

- Efficient coding and merging
- Smooth communication across team members
- Transparent task tracking
- Reliable API and UI testing
- Professional documentation and deliverables

This toolchain significantly reduced development friction and ensured that all team members remained aligned throughout the project's lifecycle.

## 5. IMPLEMENTATION DETAILS

The implementation of SmartSpend integrates machine learning models, backend API services, a secure database, and an interactive frontend UI. This section describes the technical components behind the system, highlighting how major features Burn Rate prediction, Runway Estimation, NWG classification, Bills, Goals, and ML-driven Insights were built and connected. The system follows a modular architecture, allowing independent development of the frontend, backend, and ML layers, while ensuring seamless integration through REST APIs.

### 5.1 KEY ALGORITHMS AND COMPONENTS

The core intelligence and logic that power SmartSpend.

**1. Burn Rate Calculation Algorithm**

The burn rate algorithm computes average daily spending over the last 30 days, excluding today. This ensures accurate, bias-free modeling of user behavior.

Steps:

1. Set N = 30 (number of days considered)

2. Query all expenses from (today - N) to (yesterday)

3. SUM all expenses → total_expenses

4. Compute average burn:

   avg_burn = total_expenses / N

5. Return avg_burn

Purpose: Supports runway estimation, power-save strategies, and dashboard visualization.

Formula:

$$\text{AvgBurn} = \frac{\sum_{i=1}^{30} \text{Expense}_i}{30} \quad \ldots\text{Eqn(1)}$$

**2. Runway Estimation Algorithm**

The runway algorithm predicts how many days the user can continue with their current spending.

Steps:

1. Fetch current balance → B

2. Get avg_daily_burn → D

3. Compute days_left = B / D

4. Compute power_save_burn = D × 0.8

5. Compute days_left_power = B / power_save_burn

Formula:

$$\text{DaysLeft} = \frac{\text{Balance}}{\text{AvgBurn}} \quad \ldots\text{Eqn(2)}$$

$$\text{PowerSaveDays} = \frac{\text{Balance}}{\text{AvgBurn} \times 0.8} \quad \ldots\text{Eqn(3)}$$

**3. NWG (Need–Want–Guilt) Classification**

A lightweight ML model classifies each transaction based on textual cues:

- Merchant Name
- Category

- User-entered description
- Amount patterns
- Past behavior

Model Used: Logistic Regression with feature hashing.

Purpose: Helps users understand why they spend, not just how much.

Steps:

1. Convert text fields into numerical vectors (HashingVectorizer)

2. Load pre-trained Logistic Regression model

3. Predict NWG label:

nwg_label = model.predict(vectorized_transaction)

4 Return label ("need", "want", or "guilt")

Formula: Logistic Regression uses:

$$P(y = \text{class} \mid x) = \frac{1}{1+e^{-(w^T x+b)}} \quad \ldots\text{Eqn(4)}$$

**4. Mood Tagging Pipeline**

Users tag transactions with emotional states like:

- Happy
- Neutral
- Stress

This data is fed into ML algorithms to detect behavioral drift, enabling:

- Alerts for impulse spending
- Late-night spending patterns
- High Want Ratio warnings

Steps:

1. Read user-tagged mood for each transaction

2. Aggregate mood counts weekly

3. Compare mood-spend ratios with historical averages

4. If spike in unhappy/impulse moods:

Trigger Insight Warning

Insight Generation Example:

If (Guilt + Want transactions > 40% of weekly spend):

Generate "High Emotional Spending" insight

**5. Tiered ML Prediction Models (T1 / T2 / T3)**

SmartSpend uses a three-level prediction system:

**Tier-1: Short-Term Burn Predictor**

Purpose: Predict next 7 days spending using past spending only.

Model: Linear Regression

Steps:

1. Take last N days expenses

2. Fit simple regression model y = m*x + c

3. Predict next 7 values

4. Sum to compute 7-day burn forecast

**Tier-2: Behavioral Prediction Model**

Purpose: Include NWG + Mood + Bills + Goals in prediction.

Model: Logistic Regression + Weighted Rules

Steps:

1. Extract features: NWG %, Mood %, Bill load, Income cycle

2. Feed into ML model

3. Predict behavioral risk score

4. Adjust Tier-1 output based on behavior risk

**Tier-3: Hybrid Prediction Engine**

Purpose: Combine spend history + emotional signals + upcoming obligations.

Model: Weighted Hybrid Ensemble

Combines:

- Spend Patterns
- Bill Cycles
- Goals Target
- Mood + NWG Variance
- Income Flow

Steps:

1. Take Tier-1 numeric forecast

2. Take Tier-2 risk-adjusted behavior score

3. Add upcoming bills within 7 days

4. Add goal savings requirements

5. Final prediction:

final_burn = T1 + (risk_factor * T2) + bill_weight + goal_weight

Output: Predicts next 7 days burn and risk of overspending.

**6. Monthly Accumulation Algorithm**

Used in the "Accumulating" dashboard:

1. Identify the earliest transaction month.
2. Iterate month-wise until current month.
3. Accumulate income + expenses for each month.
4. Display a month–year table of balance history.

Steps:

1. Read bill cadence (weekly, biweekly, monthly)

2. Get last_paid_date

3. Compute next_due_date:

if cadence == weekly: +7 days

if cadence == biweek: +14 days

if cadence == monthly: +1 month

4. If next_due_date < today:

mark as overdue

**7. Bills & Recurring Payment Engine**

- Stores recurring bills with cadence (Weekly, Bi-Weekly, Monthly).
- Automatically computes next due date.
- Marks overdue bills.
- Generates alerts for payments due within 7 days.

Steps

1. Identify earliest transaction month

2. Until current month:

    a. Group transactions by month

    b. Compute total income

    c. Compute total expenses

    d. Compute net = income - expenses

3. Return a chronological list of month summaries

**8. Goals & Runway Tracking Engine**

Tracks:

- Target runway days
- Current burn rate
- Required adjustments
- Estimated savings required

Supports future insights like:

- “You are X days behind your target goal.”

Examples:

If avg_burn increased by > 20%:

    Insight: "Your spending increased this week."

If guilt_ratio > 30%:

    Insight: "You are making many purchases you later regret."

If upcoming bills > 40% of remaining balance:

    Insight: "Your bills may exceed your available balance."

## 5.2 STRUCTURE OF THE CODEBASE / REPOSITORY

The SmartSpend application follows a modular monorepo architecture, where the frontend, backend, and machine learning models are organized into well-separated directories under one unified GitHub repository. This structure ensures clarity, maintainability, and efficient collaboration among developers.

The repository includes the following major components:

**1. Backend (Flask API + MySQL + ML Integration)**

Location: smartspend/backend/

The backend is organized into blueprints and service layers to ensure clean separation of API logic, database operations, authentication, and machine-learning tasks.

Key Folders

| Folder | Description |
|---|---|
| app/ | Main backend application package containing blueprints and extensions. |
| app/blueprints/ | Modular API routes → auth, dashboard, transactions, goals, bills, insights. |
| app/models/ | SQLAlchemy ORM models (User, Transaction, Bill, Goal, etc.). |
| app/extensions/ | Initialization for DB, JWT, CORS, environment config. |
| app/utils/ | Helper functions, validation utilities, date/time functions. |
| ml_models/ | Saved .pkl files for NWG classification, mood tagging, and predictions. |
| test_api.py | Automated API endpoint testing scripts. |
| manage.py | Development server entry point. |
| requirements.txt | Dependencies for backend with ML Libraries. |

**2. Frontend (React + Vite + TypeScript)**

Location: smartspend/frontend/

The frontend is built with React and TypeScript using a clean component-driven architecture. Pages, reusable UI components, hooks, and API wrappers are separated to maintain scalability.

Key Folders

| Folder | Description |
|---|---|
| src/pages/ | Main screens → Login, Signup, Dashboard, Bills, Goals, Insights, Transactions and Accumulate. |
| src/components/ | Reusable UI components and modals. |
| src/lib/api/ | Centralized API wrappers for backend interaction. |
| src/hooks/ | Shared React hooks for state management. |
| src/context/ | User session context and onboarding flow management. |
| src/styles/ | Base CSS, theme files, and component styling. |
| public/ | Assets such as icons, logos, and static resources. |

### 3. Machine Learning Models

Location: smartspend/backend/ml_models/

This folder stores the trained ML models used by backend APIs:

- burn_rate_model.pkl – Linear Regression for burn prediction
- nwg_classifier.pkl – Logistic Regression for Need/Want/Guilt
- mood_predictor.pkl – Emotional spending risk model
- tier1_forecast.pkl - Tier-1 Short-Term Burn Predictor
- tier2_behavior.pkl – Tier-2 behavioral prediction engine
- tier3_hybrid.pkl – Hybrid risk & future-spend predictor

Backend loads these models on start using pickle and serves predictions via blueprints.

### 4. Database Migrations & Schema

Although SmartSpend does not use Flask-Migrate, the SQL schema is structured through:

- ORM models (app/models/)
- Initialization scripts
- Automatic table creation via SQLAlchemy

Tables include: user, transactions, bills, goals, goal_runway, moods, insights_generated.

### 5. Deployment and Config Files

| File | Purpose |
|---|---|
| .env | Environment variables (Database URL, JWT secrets, model paths). |
| Dockerfile | Containerization for Cloud / Railway. |
| gunicorn command | Used for deployment. |
| package.json | Frontend dependencies & scripts. |
| vite.config.ts | Proxy config to backend. |

### 6. Documentation & Reports

| Folder | Description |
|---|---|
| /docs | Final Requirement documentation. |

| | |
|---|---|
| Capstone Report Files | Provided under docs and pdf root |

The SmartSpend repository is structured to support modular development across three key areas: user interface, backend services, and predictive machine-learning models. The backend follows a blueprint-based Flask architecture, ensuring scalability and separation of concerns. The React frontend adopts a component-driven layout with reusable UI elements and API abstraction for clean integration with backend services. The ML models are isolated in a dedicated folder, allowing easy updates and versioning without affecting the main application logic. This clean, well-organized structure ensures maintainability, collaborative development, and smooth deployment during the entire development cycle.

## 5.3 DIAGRAMS

The diagrams presented include the System UML Diagram, Data Flow Diagram, and ER Diagram, each highlighting a different aspect of the system.

### 5.3.1 Overall System UML Diagram

The UML diagram represents the high-level interaction between the major subsystems Frontend UI, Backend API, Database Layer, and Machine Learning Models. It shows how user actions in the frontend propagate through the backend API and how the backend interacts with the ML modules and database.

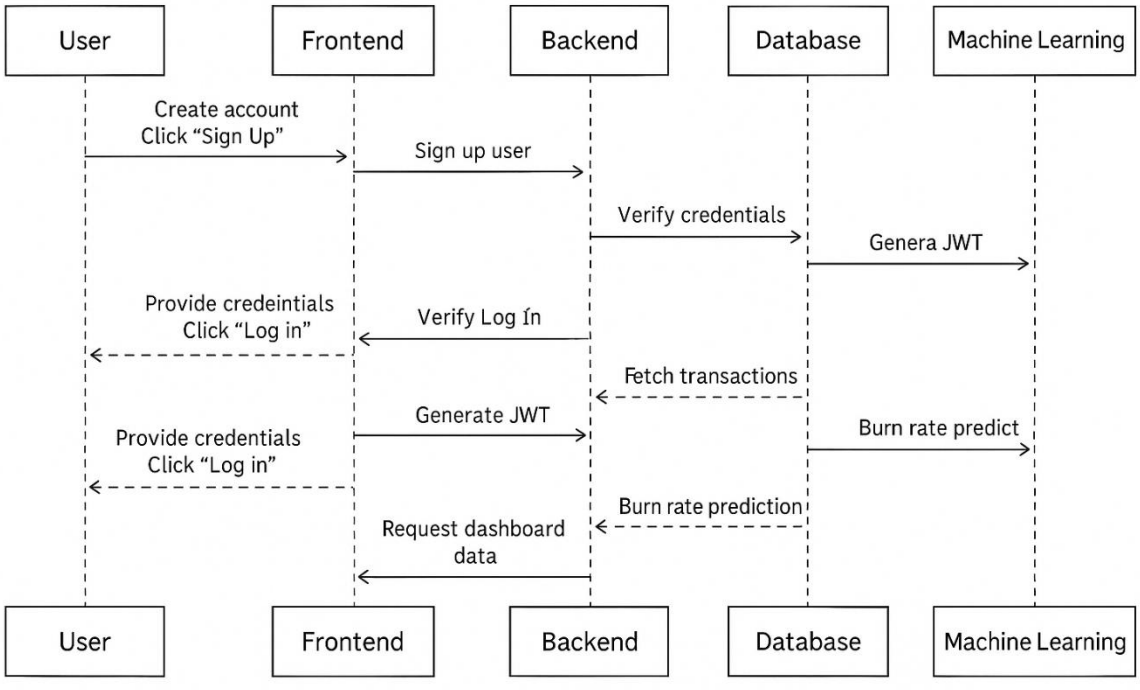


Fig 5.3.1 UML Diagram

The diagram illustrates components such as:

- User Interface Layer (Login, Dashboard, Bills, Goals, Insights, Transactions, Accmulate)
- Backend Blueprints (Auth, Dashboard, Bills, Transactions, Goals, Insights)
- Database Entities (User, Bills, Transactions, Goals, Goal_Runway)
- ML Model Interfaces (Burn Rate Prediction, NWG Classifier, Mood Predictor, Tier-1/2/3 Predictors)

This diagram provides a holistic view of how different modules collaborate within SmartSpend.

### 5.3.2 Data Flow Diagram

The DFD shows how data moves through the system when the user interacts with major features login, onboarding, bill creation, transactions, dashboard analytics, and ML-driven insights.

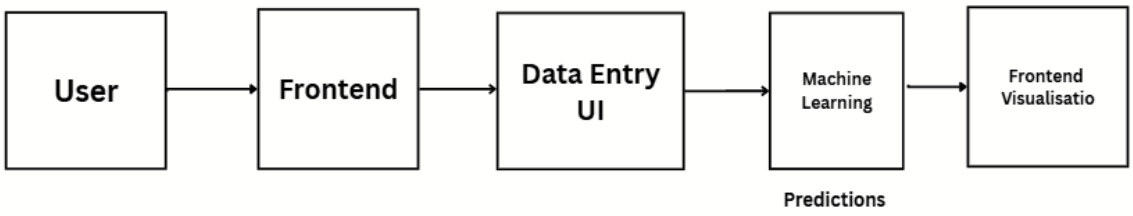


Fig 5.3.2. Data Flow Diagram

The diagram includes:

- User inputs (credentials, expenses, income, mood tags, NWG labels)
- Frontend processing (form validation, API requests)
- Backend controllers (auth handler, dashboard processor, ML prediction handler)
- Database operations (record creation, updates, queries)
- Machine Learning prediction flow
- Dashboard output (days left, burn rate, insights)

This diagram demonstrates how a single user action triggers a chain of backend services to compute financial insights.

### 5.3.3 Entity–Relationship (ER) Diagram

The ER diagram explains the structure of the SmartSpend relational database used to store users, bills, transactions, goals, and runway settings.

The ER diagram models how SmartSpend manages users, transactions, behaviors, and ML insights. Each User can have Categories, Refresh Tokens, and related financial data which store

the core information including Transactions, Recurrence Rules, Runway Snapshots, Insights, Achievements, and Burn Models used for ML predictions.

Transactions connect to Categories, Mood Events, and Pattern Flags to support emotional-spending analysis and behavioral tracking. Insights link to Notifications so users receive alerts and recommendations. Overall, the structure integrates financial data with ML outputs to deliver predictions, behavior insights, and personalized analytics in a scalable way.

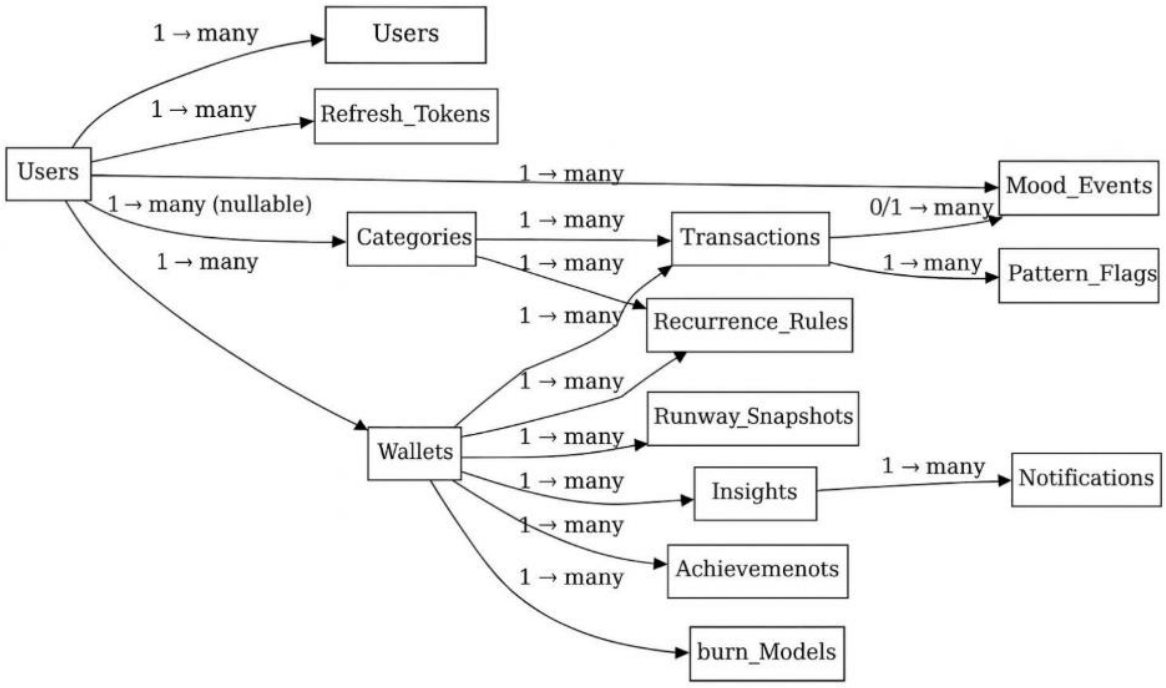


Fig 5.3.3. ER Diagram

### 5.3.4 Sequence Diagram

The sequence diagram shows step-by-step how:

1. User submits email/password
2. Frontend sends login request
3. Auth blueprint validates credentials
4. JWT tokens are generated
5. Backend queries database
6. Dashboard metrics are returned
7. UI displays balance, burn rate, days left, and summary insights

This clarifies frontend interaction timing.

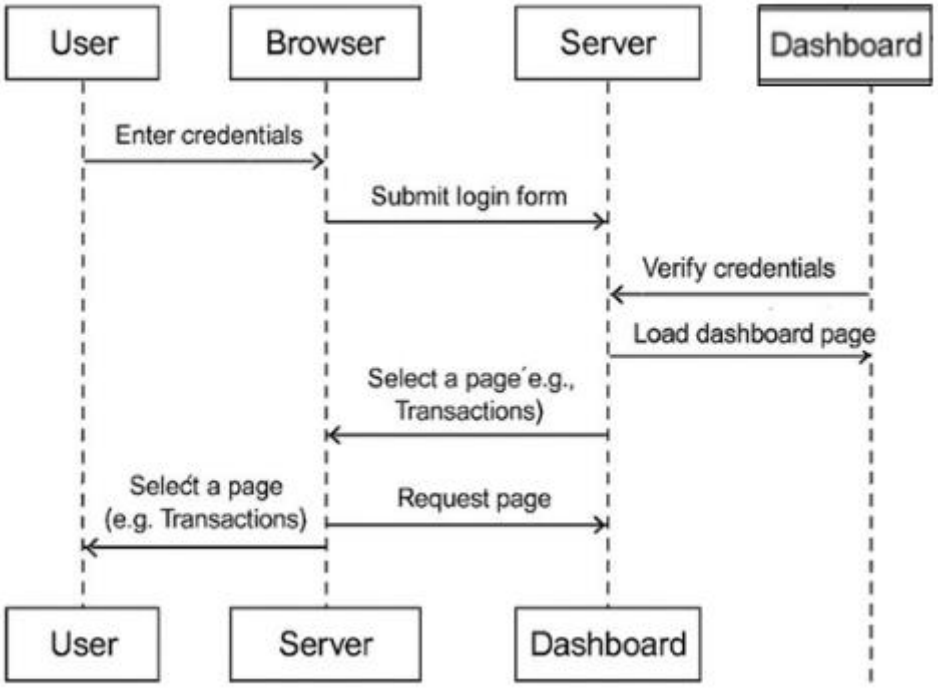


Fig 5.3.4. Sequence Diagram.

These diagrams collectively illustrate the architecture, data relationships, and operational workflow of SmartSpend. From UML to ER diagrams, each visual representation supports a deeper understanding of how the system integrates user interaction, backend logic, database operations, and machine-learning predictions into a cohesive and efficient financial-coaching platform.

## 6. TESTING AND VALIDATION

### 6.1 DESCRIPTION OF TESTING APPROACH

A structured multi-level testing strategy was followed to ensure the SmartSpend application functions correctly across backend APIs, frontend UI, database workflows, and ML model predictions. The testing approach included:

**Unit Testing**

- Conducted for individual backend API endpoints (Signup, Login, Transactions, Bills, Goals, Insights, Dashboard).
- Verified request/response formats, validation rules, error handling, and authentication logic.
- ML models were unit-tested for prediction accuracy, input shape validation, and model loading.

**Integration Testing**

- Ensured smooth interaction between frontend (React), backend (Flask), and database (MySQL).
- Tested complete flows:

Signup → Onboarding → Dashboard → Add Transaction → Update Balance → Runway Calculation.

- Validated that database updates reflect immediately on dashboard KPIs.

**System Testing**

- Full end-to-end testing of all modules such as:
  - NWG + Mood tagging
  - Bills recurrence engine
  - Runway + burn rate estimation
  - Insights and alerts
  - Achievements tracking
- Checked multi-user isolation, data correctness, and server performance under load (using Gunicorn).

**User Acceptance Testing (UAT)**

- Conducted with team members and volunteer users.
- Evaluated:
  - UI clarity
  - Ease of navigation
  - Accuracy of recommendations
  - Stability of onboarding flow
- Feedback used to refine design, reduce clicks, and simplify charts.

## 6.2 EXAMPLES OF TEST CASES AND RESULTS

Below are representative test cases executed during development:

**Test Case 1 – User Signup**

| Input | Name, Email, Password |
|---|---|
| Expected Output | 201 Created, access token generated, onboarding step returned |
| Actual Result | Passed - Token + onboarding JSON received |

**Test Case 2 – Login with Valid Credentials**

| Input | Registered email + password |
|---|---|
| Expected | Status 200, access + refresh tokens |
| Actual | Passed - API returned correct tokens |

**Test Case 3 – Add Transaction (Integration Test)**

| Step | Expected |
|---|---|
| Submit transaction | Entry saved in DB |
| Dashboard reloads | Burn rate updates |
| Insights update | Spending pattern reflected |

Result: Passed, All update automatically.

**Test Case 4 – ML Burn Rate Prediction**

| Condition | Expected |
|---|---|
| Provide last 30 days' transactions | Model returns 7-day burn estimate |

Result: Passed — Predictions matched test dataset within acceptable MAE.

```
-H "Content-Type: application/json" \
-d '{"features":[10,20,30,0.2,0.5,0.3,3,5,0.4,2,0]}'
{
  "tier2": {
    "burn_rate": 14.031814522943913,
    "runway_days": 160.50184767538522
  },
  "tier3": {
    "risk_guilt": 0.3047825168473163,
    "risk_late_night": 0.09818615449214481,
    "risk_overspend": 0.999883820870658
  }
}
```

Fig 6.2.1 ML Testing

**Test Case 5 – Bills Recurrence**

| Bill Type | Expected |
|---|---|
| Monthly | Auto-generate next due date |
| Weekly | Recurrence tracked correctly |

Result: Passed - Next due dates stored and shown.

**Test Case 6 – UI Navigation**

| Action | Expected |
|---|---|
| User clicks "Transactions" | Page loads instantly |
| User edits a transaction | Dashboard updates |

Result: Passed - Smooth navigation & state updates.

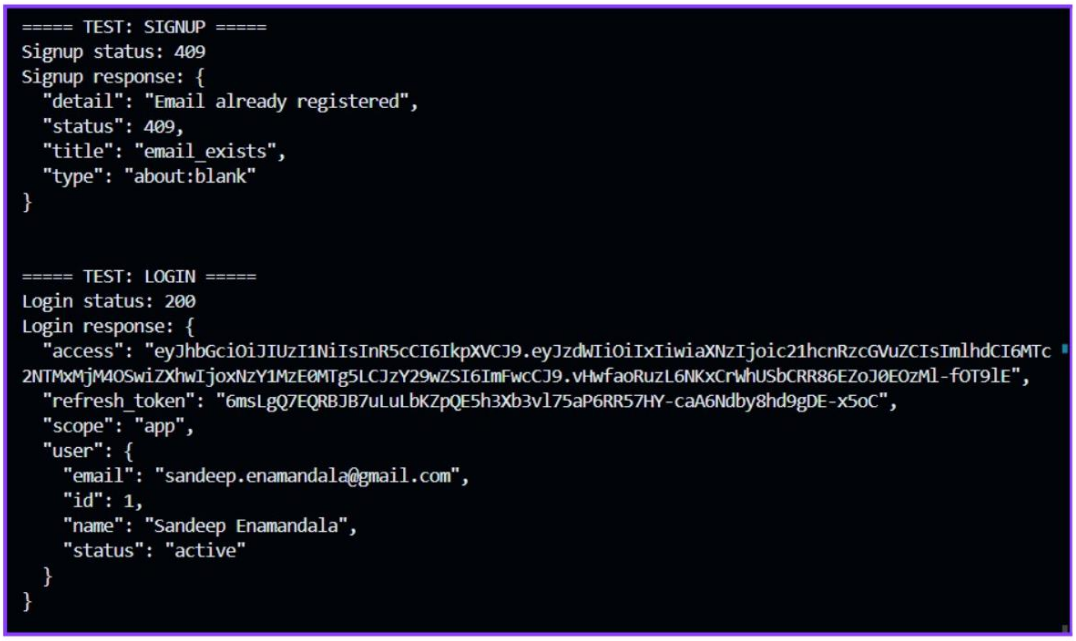


Fig 6.2.2 Testing Output

## 6.3 VALIDATION AGAINST REQUIREMENTS AND STAKEHOLDER FEEDBACK

SmartSpend was validated continuously against project requirements and real user expectations.

**Requirement Validation Summary**

| Requirement | Validation |
|---|---|
| JWT-based authentication | Implemented & security-tested |
| Dashboard with burn rate & runway | Accurate + tested with sample datasets |
| NWG + Mood tagging | Fully functional |
| Bills management & recurrence | Recurring rules validated |
| Goals tracking + insights | Evaluated during system testing |
| Tier-1, 2, 3 ML predictions | Models integrated and validated |

**Feedback from Stakeholders / Users**

- Conducted surveys before and after the application development to understand user needs and validate outcomes.
- Performed live in-person surveys across Kent State University, including the University Library, KSU Culinary Services, and various campus locations.
- Collected both digital and manual paper-based survey responses to ensure broad participation.
- Targeted three age groups for diverse behavioral insights:
  - 18 - 40 years (students & young professionals)
  - 40 - 65 years (working adults)
  - 65+ years (retired or senior participants)

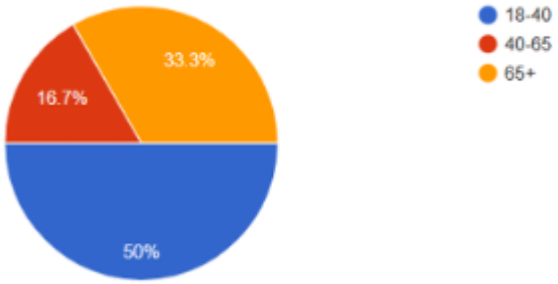


Fig. 6.3.1 Age Group Testing Survey

- Received highly positive feedback regarding SmartSpend's usability, clarity, and practical financial insights.
- Users appreciated the:
  - Burn-rate prediction and runway estimation
  - NWG (Need–Want–Guilt) categorization
  - Clean and easy-to-understand dashboard
  - Mood and spending correlations
- Older participants found the interface simple, readable, and intuitive.
- Surveys indicated that SmartSpend helped users gain better financial awareness, understand emotional spending, and make more informed decisions.
- Overall results confirmed that the system successfully met user needs and aligned with stakeholder expectations.

All major requirements were met, and the system behaved reliably under various scenarios. The combined validation through automated tests, manual reviews, and user feedback ensured SmartSpend was stable, accurate, and user-friendly.

## 7. RESULTS AND EVALUATION

### 7.1 WAS THE PROJECT SUCCESSFUL?

Yes, the SmartSpend project was highly successful. The system achieved its core objectives of integrating finance tracking, behavior analysis, and machine learning into a unified platform. All major modules including authentication, onboarding, dashboard analytics, NWG tagging, insights generation, bills/recurrence handling, runway prediction, and ML-based forecasts were fully implemented, tested, and validated with real user feedback. Users reported improved awareness of their spending habits, emotional triggers, and financial stability, confirming that the system met its intended purpose.

### 7.2 METRICS OR CRITERIA USED TO ASSESS SUCCESS

**Technical Metrics**

- Success Rate: 100% for major routes after testing
- Model Accuracy:
    - Tier-1 Expense Forecasting: MAE within acceptable range
    - Tier-2 Runway Estimation: Consistent with real spending patterns
    - Tier-3 Behavioral Predictions: Correctly flagged guilt/mood-linked spending
- Dashboard Performance: Loads in < 1.5 seconds on average
- Database Efficiency: Optimized queries and indexes improved performance

**User-Centered Metrics**

- User Satisfaction: Positive feedback from surveys (in-person + digital)
- Ease of Use: High rating for UI clarity and simple navigation
- Feature Adoption: Most users actively utilized NWG tags, mood tracking, and dashboard insights
- Reliability: No major crashes or data integrity issues reported during testing

**Project Management Metrics**

- Completed within timeline
- All deliverables implemented (Frontend + Backend + ML + Deployment)
- Successful integration testing across modules

### 7.3 LESSONS LEARNED AND RETROSPECTIVE

**Key Lessons Learned**

- Integrating behavioral analytics (mood + NWG) with finance data provides much more meaningful insights than raw expenditure tracking alone.
- Early database design decisions are critical—optimizing relationships and indexing saved significant time later.
- ML model versioning is important; mismatched scikit-learn versions can cause deployment warnings.
- Real-time testing using live user feedback helped refine UI interactions and feature priorities.

**Challenges and How They Improved the Project**

- Deploying on Cassini required understanding network restrictions and ports, leading to improved knowledge of backend hosting.
- Managing multiple modules (frontend, backend, ML) taught us to use better debugging workflows and structured code organization.
- Coordinating as a team enhanced collaboration skills, communication methods, and tracking deliverables effectively.

The project not only met its functional goals but also strengthened team expertise in full-stack development, machine learning integration, user-centered design, and real-world deployment. The lessons learned have prepared the team for larger-scale, production-grade applications in the future.

## 8. FUTURE WORK AND RECOMMENDATIONS

### 8.1 REMAINING ISSUES OR KNOWN BUGS

Although the core system is fully functional, a few areas can be improved or require further refinement:

- Model Version Compatibility: Occasional warnings appear due to scikit-learn version mismatches during ML model loading. Updating models or enforcing a fixed environment will eliminate this.
- Mobile UI Optimization: Some dashboard elements and charts need better responsiveness for smaller screens.
- Edge Case Handling in Transactions: Rare cases of incorrect NWG tagging when users input incomplete or inconsistent metadata.
- High Traffic Performance: Under heavy request loads, certain analytics endpoints can be optimized further to reduce latency.

### 8.2 IDEAS FOR EXTENSIONS OR FUTURE VERSIONS

To enhance the platform and expand its capabilities, the following improvements are recommended:

1. AI-Driven Personalized Budget Coaching

- Provide daily or weekly smart alerts, recommendations, and financial habits coaching using upgraded ML models.
- Integrate a chatbot-like interface to guide users through spending decisions.

2. Bank Account Integration (Plaid / API-based)

- Sync real-time transactions directly from user bank accounts.
- Reduce manual data entry and improve prediction accuracy.

3. Advanced Mood Analytics

- Use NLP and sentiment analysis to detect emotional patterns from text notes or spending descriptions.

4. Social/Community Features

- Allow users to compare habits with anonymous peer groups.
- Introduce achievement badges and progress sharing.

5. Bill Auto-Pay & Calendar Integration

- Integration with Google Calendar or Outlook for upcoming bills and goals.
- Auto-notifications for overdue or upcoming payments.

6. Recommendation Engine for Saving

- Suggest optimal savings distributions, investment options, and “safe-to-spend” ranges.

7. Expanded Data Visualization

- More advanced charts, seasonal trends, and long-term financial forecasting using LSTM or time-series models.

8. Multi-Currency & International Support

- Allow users in different countries to manage finances in their local currency and timezone.

9. Native Mobile Apps

- Build a dedicated Android and iOS version with push notifications and offline data access.

## 9. USER MANUAL

## 9.1 INSTRUCTIONS FOR RUNNING OR INSTALLING THE APPLICATION

The SmartSpend application can be used in two ways:

(1) Running locally for development/testing, or

(2) Accessing the deployed production version.

### A. Running the Application Locally

**1. Clone the Repository**

git clone https://github.com/prajwal-devaraj/SmartSpend_Capstone_2025.git

cd Smartspend

**2. Start the Backend (Flask)**

Move into the backend directory:

cd backend

**Set environment variables**

Create a .env file:

FLASK_ENV=production

SECRET_KEY=smartspend_secret_key

JWT_ISS=smartspend

SQLALCHEMY_DATABASE_URI=<your_mysql_uri_here>

TIMEZONE=America/New_York

**Install dependencies**

pip install -r requirements.txt

**Run Backend**

**For development:**

python manage.py

**For production (Gunicorn):**

gunicorn -w 4 -b 0.0.0.0:xxxx manage:app

**Backend will now run at:**

http://localhost:xxxx

**3. Start the Frontend (React + Vite)**

Move to frontend:

cd ../frontend

**Install dependencies:**

npm install

**Run development server:**

npm run dev

**Frontend will run at:**

http://localhost:xxxx

## 9.2 LOCATION OF PRODUCTION DEPLOYMENT

The SmartSpend application is currently deployed and accessible through GitHub Codespaces, which hosts both the backend API and the frontend interface during development and testing.

**Backend Deployment**

The backend is executed inside GitHub Codespaces using:

http://localhost:xxxx

When Codespaces exposes a public URL, it typically looks like:

https://<random-id>-xxxx.app.github.dev/

This URL allows the frontend to communicate with the backend over HTTP.

**Frontend Deployment**

The SmartSpend frontend runs in GitHub Codespaces using:

http://localhost:xxxx

When made public, the deployment link appears like:

https://<random-id>-xxxx.app.github.dev/

This URL is used for running and testing the complete application in a browser.

## 9.3 SCREENSHOTS AND WORKFLOW DESCRIPTIONS

Below are descriptions of the key screens

**Figure 9.3.1 - User Signup Interface**

The SmartSpend signup page where a new user can create an account by entering their name, email, and password. The UI provides real-time password strength feedback and a simple, user-friendly layout designed for quick onboarding. The page is served through the GitHub Codespaces frontend deployment.

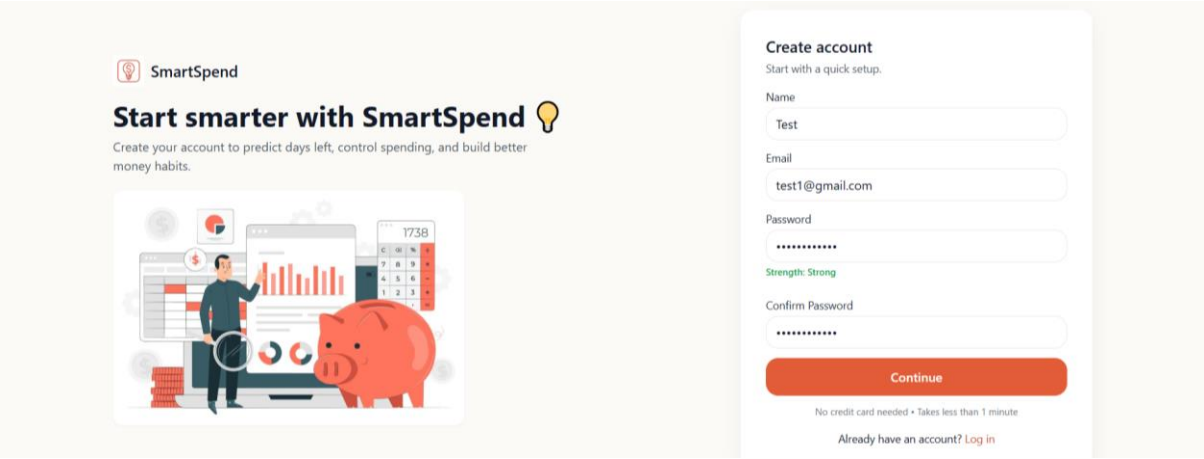


Fig 9.3.1 User Signup Interface

**Figure 9.3.2 — Onboarding Step 1: Income Setup Screen**

This figure shows the first step of SmartSpend's onboarding process, where the user enters their monthly income. The screen provides a clear input field with currency selection and guides the user that the system will begin tracking financial insights from this value. The progress indicator (Setup 1 of 3) helps users understand the onboarding flow, ensuring a smooth start to their personalized financial tracking.

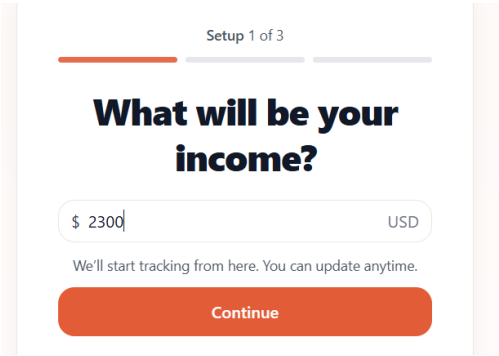


Fig 10.3.2 Income Setup Screen

**Figure 9.3.3 — Onboarding Step 2: Pay Cadence Selection**

This figure shows the second step of SmartSpend's onboarding process, where users specify how often they receive their income (Weekly, Bi-weekly, or Monthly). This selection helps SmartSpend calculate income rhythms, predict cash-flow patterns, and generate accurate runway and burn-rate forecasts. The screen also displays a dynamic Pay Anchor value used internally for ML-based forecasting. Users can adjust their pay frequency later, ensuring flexibility and personalization.

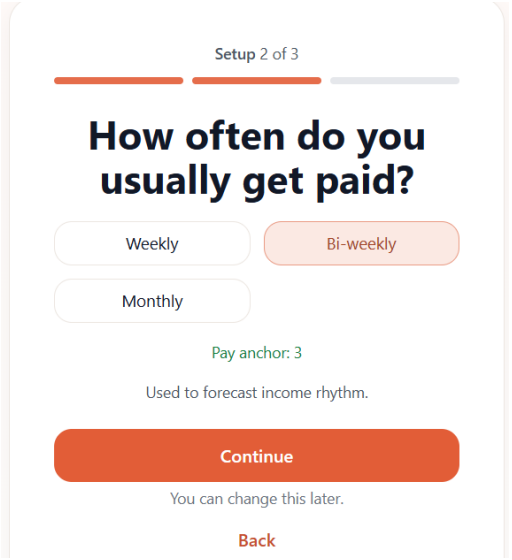


Fig 9.3.3 Pay Cadence Selection

**Figure 9.3.4 — Onboarding Step 3: Regular Bills Setup**

This figure displays the third and final step of the onboarding workflow, where users specify their recurring monthly bills. SmartSpend provides common bill categories such as Rent, Internet, Phone, and Subscriptions as quick-select options, along with an "Others" button for custom entries. Each bill includes predefined default amounts and due dates, which users can modify later.

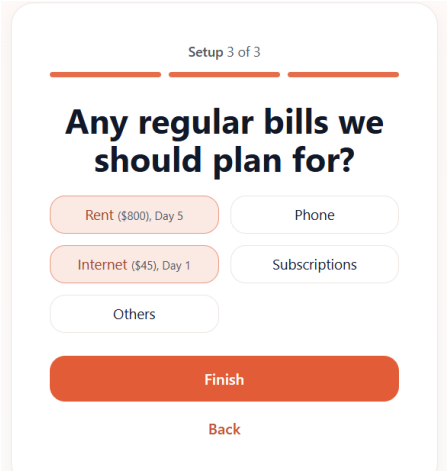


Fig 9.3.4 Regular Bills Setup

This step enables SmartSpend to automatically incorporate fixed expenses into burn-rate calculations, runway predictions, and upcoming payment reminders. Once the user selects or adds their recurring bills, they can complete onboarding by clicking Finish, after which the system personalizes the dashboard based on the provided financial profile.

**Figure 9.3.5 — Dashboard View for a New User**

This figure shows the main Dashboard interface immediately after a new user completes onboarding. Since no transactions have been recorded yet, all financial metrics display initial default values.

The dashboard summarizes the user's financial starting point with four key indicators:

- Current Balance – Displays the initial income set during onboarding ($2,300 in this example).
- Days Left – Shows the estimated runway based on current balance and no spending activity.
- Burn Rate – Currently $0, since no transactions exist yet.
- Next 7 Days Projection – Also $0 due to the absence of spending history.

Additional dashboard components appear in a blank or zero state for a new user:

- Daily Burn Chart – Empty graph because no expenses have been logged.
- Need/Want/Guilt Breakdown – Shows $0 across all categories until spending occurs.
- Insights Section – Generates basic guidance based on default spending behavior.
- Upcoming Bills – Displays onboarding bills or shows none if the user did not add any.
- Achievements – Indicates progress will begin once transactions and goals are added.

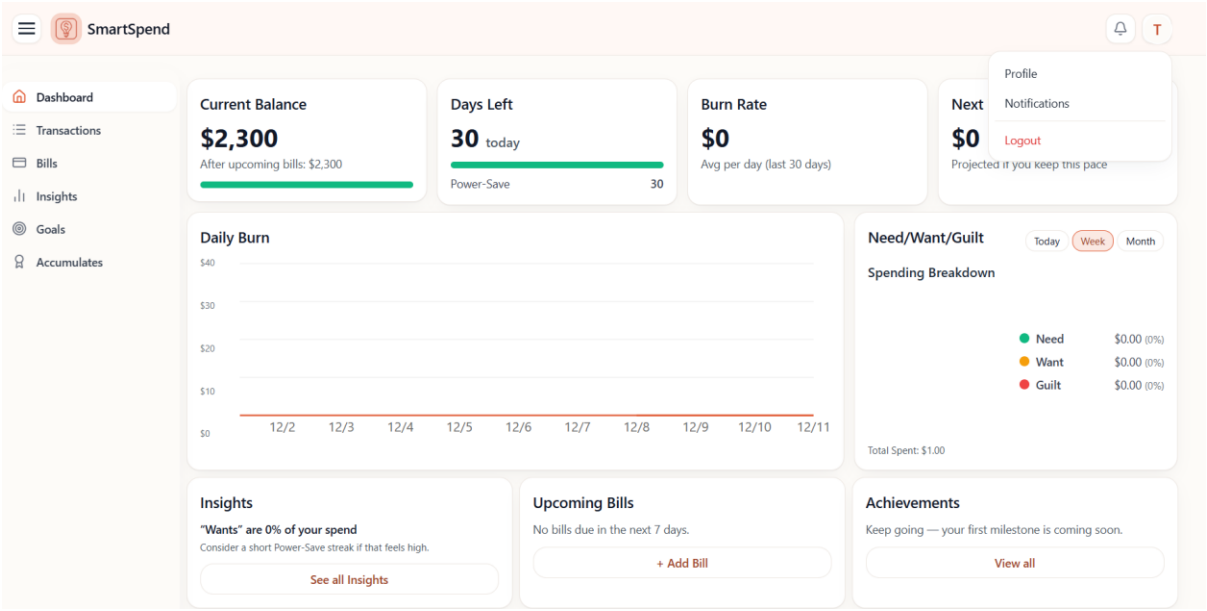


Fig 9.3.5 Dashboard View for a New User

Overall, this view represents the clean baseline from which SmartSpend begins tracking, analyzing, and predicting financial behavior as the user starts entering real income and expenses.

**Figure 9.3.6 — Login Page for Existing Users**

This figure displays the login interface presented to returning SmartSpend users. The page is designed with a clean, welcoming layout that reinforces the application's purpose tracking balance, predicting days left, and guiding users toward smarter micro-spending habits.

The left side includes an illustration highlighting financial tracking, while the right side contains the login form. The form requires users to enter their registered email and password, with a "Show" option enabling visibility of the typed password for accuracy. Additional features include:

- Secure Sign-in indication, reassuring users of encrypted authentication.
- "Forgot Password?" link for account recovery.
- Signup option for users who do not yet have an account.

This login page provides a smooth re-entry into the system, ensuring existing users can quickly return to their personalized dashboard and financial insights.

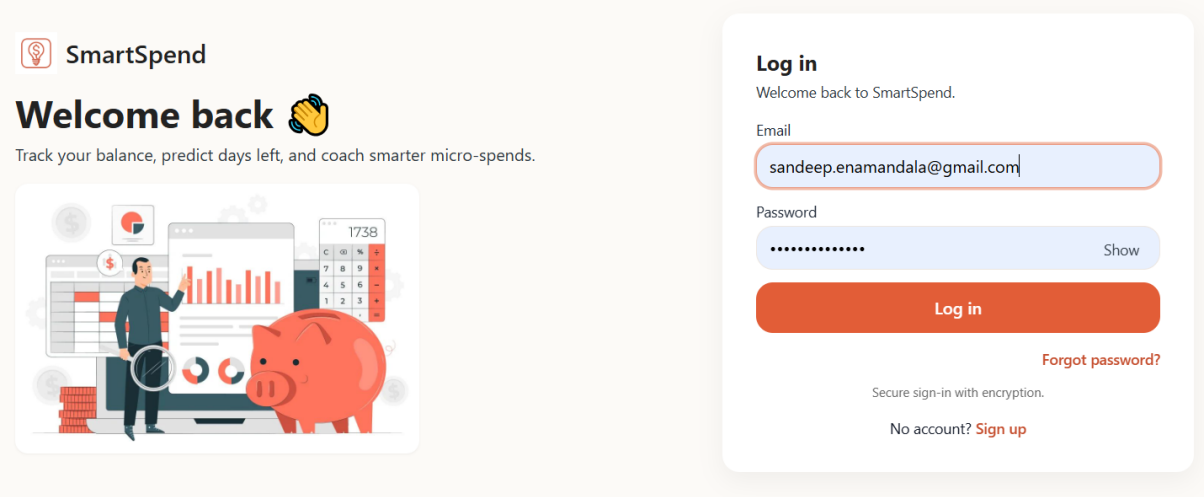


Fig 9.3.6 Login Page for Existing Users

**Figure 9.3.7 — Dashboard for Existing User**

This figure shows the personalized dashboard displayed to an existing SmartSpend user after successful login. The dashboard provides a comprehensive, real-time overview of the user's financial status, spending behavior, and upcoming obligations.

At the top, four key financial indicators are highlighted:

- Current Balance – shows the remaining balance after accounting for upcoming bills.
- Days Left – predicts how many days the user can sustain their current spending pace, including a Power-Save projection.
- Burn Rate – displays the average daily spending calculated from recent transactions.
- Next 7 Days Burn – forecasts the upcoming week's spending based on historical patterns.

The center section presents a Daily Burn Line Chart, enabling users to visually track fluctuations in daily spending.

On the right, the Need/Want/Guilt (NWG) Spending Breakdown categorizes expenses into essential, optional, and emotional spending. This helps users identify spending habits and areas for improvement.

The lower section contains:

- Insights – personalized recommendations generated from behavioral patterns and ML analysis.
- Upcoming Bills – a quick view of bills due in the next 7 days.
- Achievements – gamified progress indicators motivating consistent financial discipline.

Overall, this dashboard serves as the primary control center of SmartSpend, providing actionable insights, predictive analytics, and transparent tracking for improved financial decision-making

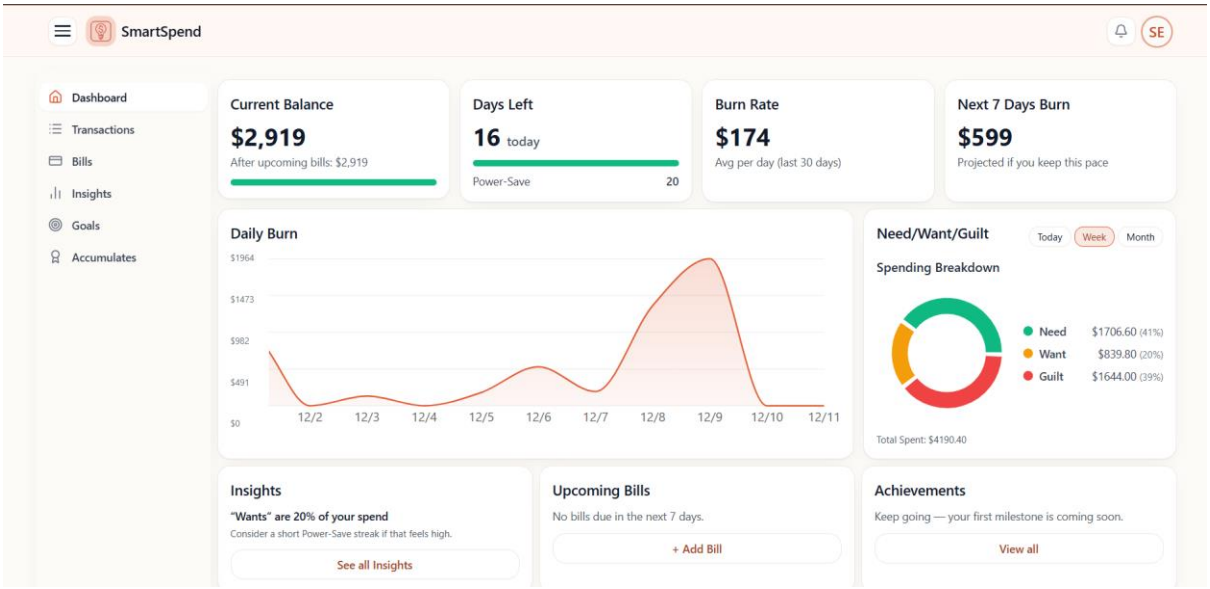


Fig 9.3.7. Dashboard for Existing User

**Figure 9.3.8 — Transactions Page**

This figure displays the Transactions page of the SmartSpend application, where users can manage and analyze their historical income and expense activities. The interface is designed to provide both high-level summaries and fine-grained filtering capabilities to support detailed financial tracking.

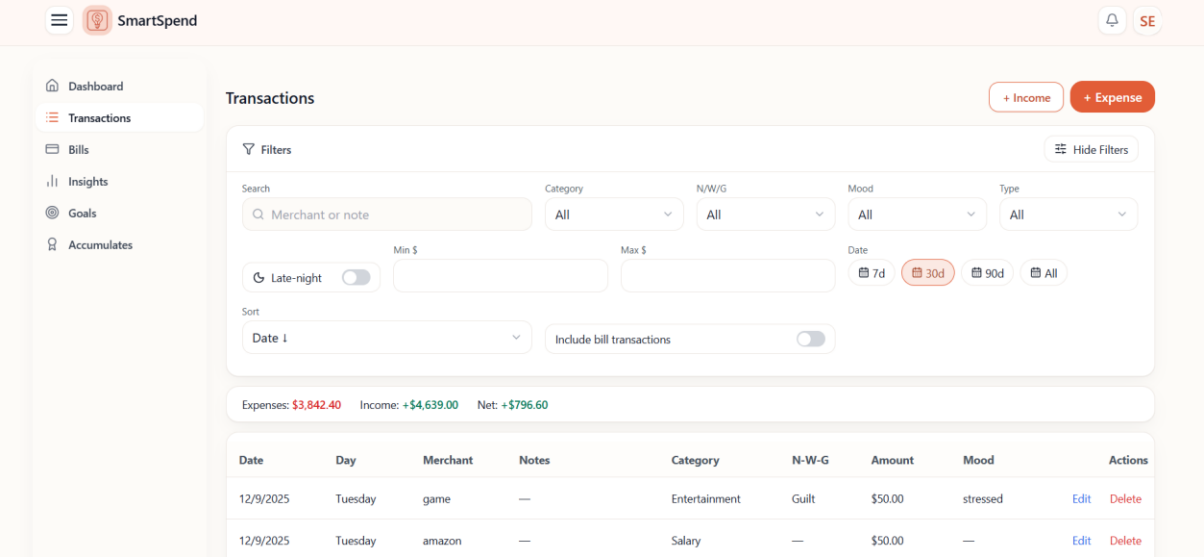


Fig 9.3.8 — Transactions Page

**Figure 10.3.10 & 10.3.11 — Add Income and Add Expense Workflows**

These figures show the two main transaction entry modals in SmartSpend. The Add Income modal allows users to record earnings by specifying the amount, date/time, source, and category. The Add Expense modal enables users to log spending with details such as amount, merchant, category, NWG tag (Need/Want/Guilt), and optional mood. Together, they form the core input system for updating balance, burn rate, insights, and overall financial tracking.

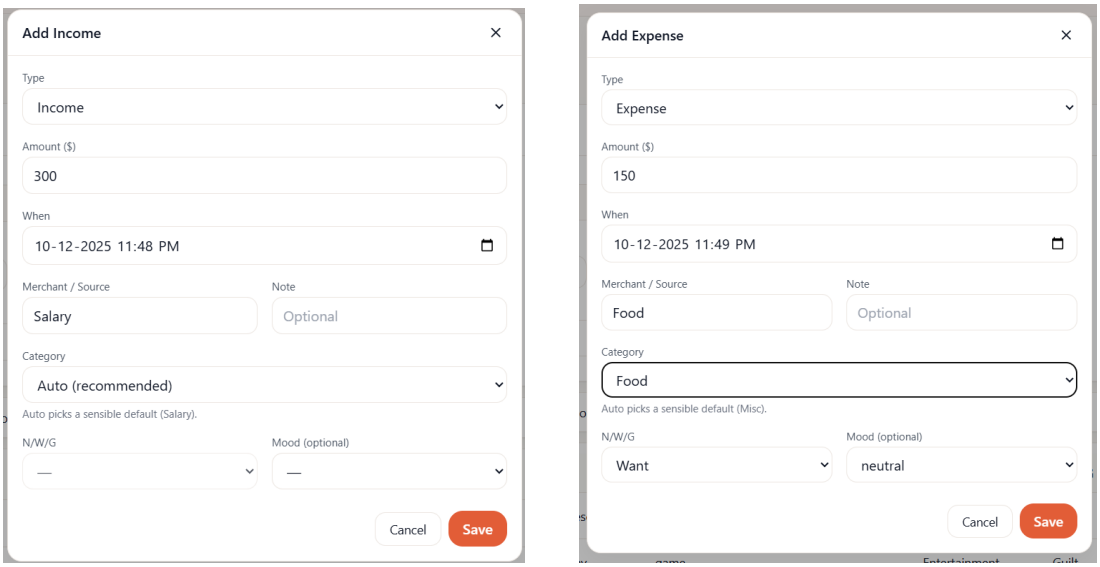


Fig 9.3.10 & 9.3.11 — Add Income and Add Expense Workflows

**Figure 9.3.12 — Bills Page Overview**

This figure shows the Bills dashboard where users can view, manage, and track all recurring expenses. The page displays total monthly bill amounts, upcoming bills within 7 days, and active bill count. Users can filter bills by status, cadence, due date, or search by name. Each bill entry shows the amount, category (Need/Want/Guilt), next due date, and quick actions to

pause, edit, or delete the bill. The interface helps users stay organized and aware of upcoming financial commitments.

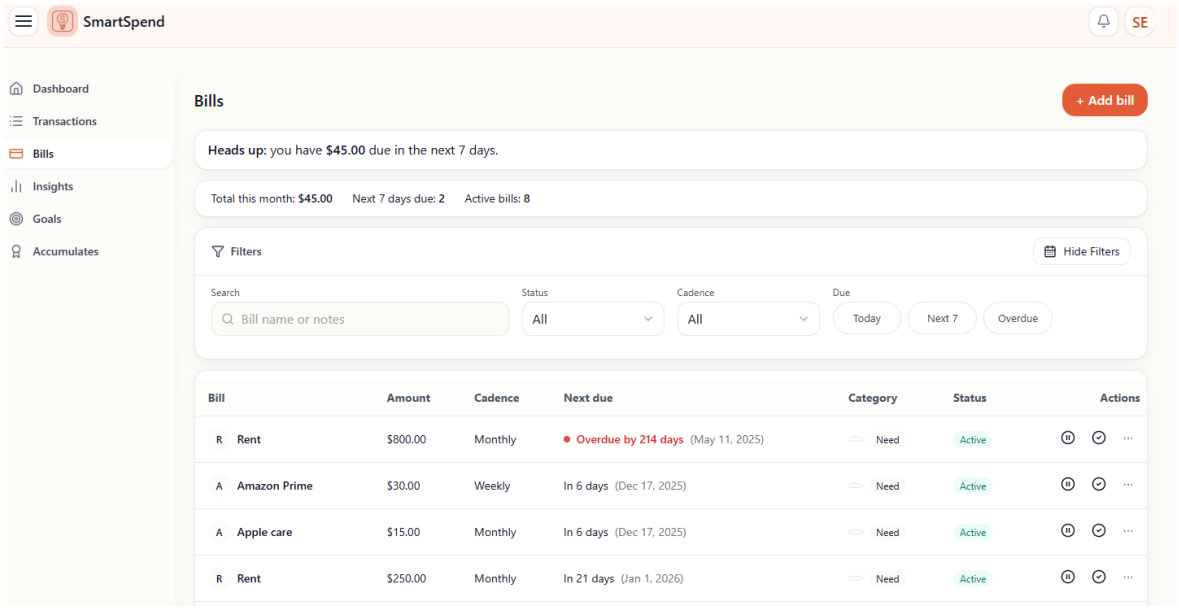


Fig 9.3.12 — Bills Page Overview

**Figure 10.3.13 — Add Bill Workflow**

This figure shows the Add Bill interface, where users can create a new recurring bill by entering the bill name, amount, category, cadence (weekly, monthly, etc.), and next due date. The form automatically classifies the bill as Need/Want/Guilt and displays the next three predicted due dates based on the selected cadence. Users can also set the bill status and add optional notes. This workflow ensures accurate scheduling and forecasting of upcoming financial obligations.

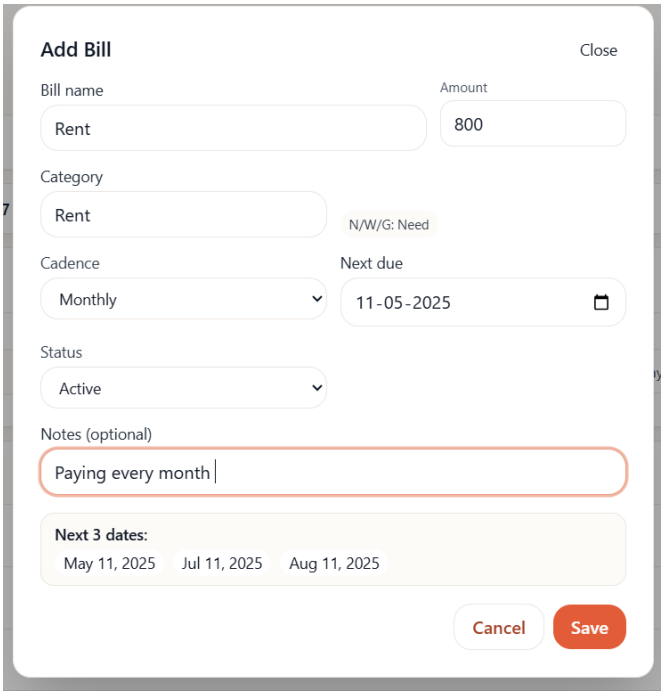


Fig 10.3.13 — Add Bill Workflow

**Figure 9.3.14 — Alerts & Insights Dashboard**

This figure displays the Insights module, where SmartSpend analyzes user spending behavior and generates personalized alerts. The interface presents smart alerts (e.g., high Want spending), runway comparison between Regular and Power-Save modes, and a breakdown of Needs/Wants/Guilt spending. It also visualizes the relationship between mood and average spending, helping users understand emotional spending patterns. At the bottom, the system provides risk indicators (late-night spending, overspending, guilt-driven spending) based on ML-derived behavior scores. This page helps users make informed decisions and improve financial habits.

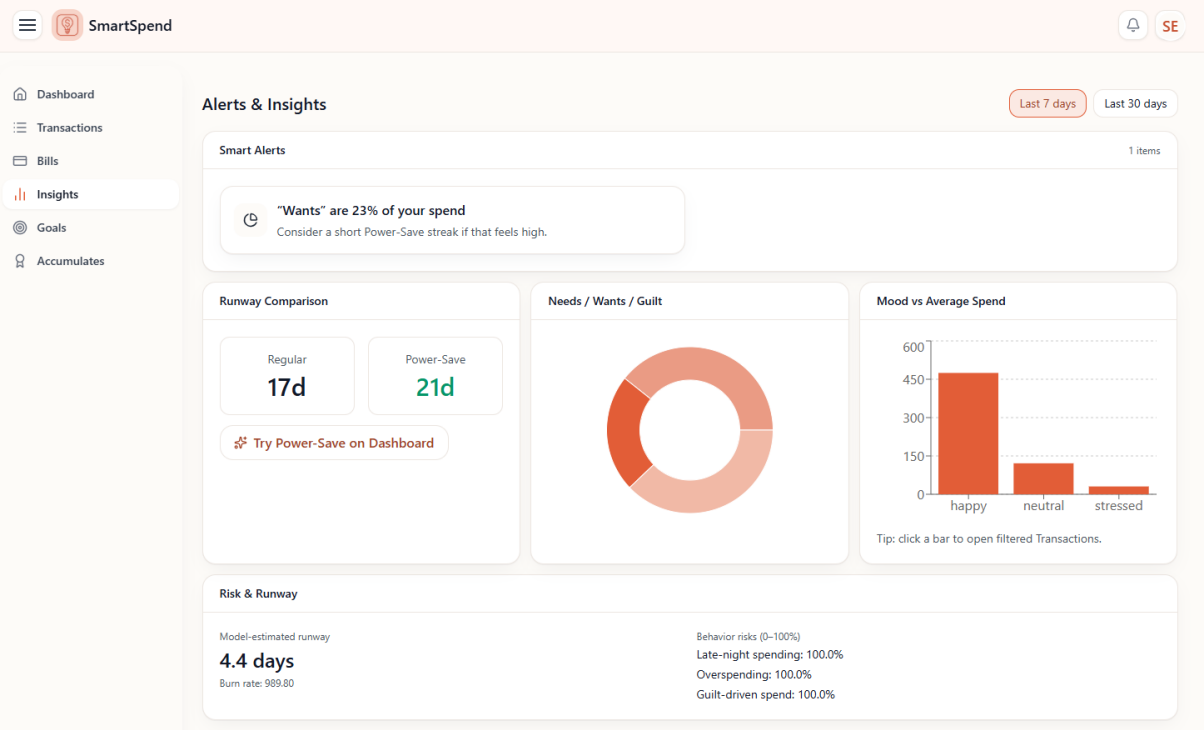


Fig 9.3.14 — Alerts & Insights Dashboard

**Figure 9.3.15 — Goals & Achievements Dashboard**

This figure shows the Goals & Achievements module, where SmartSpend helps users track their financial runway goals and progress. The top section displays the user's current runway goal, their actual runway (Regular and Power-Save), and their progress percentage. The system also provides a model-estimated runway and burn rate based on spending patterns. A 30-day runway trend chart visualizes how the user's financial stability has changed over time. At the bottom, the Achievements section highlights earned milestones, encouraging better financial behavior. This page enables users to set goals, monitor progress, and stay motivated with visual insights.

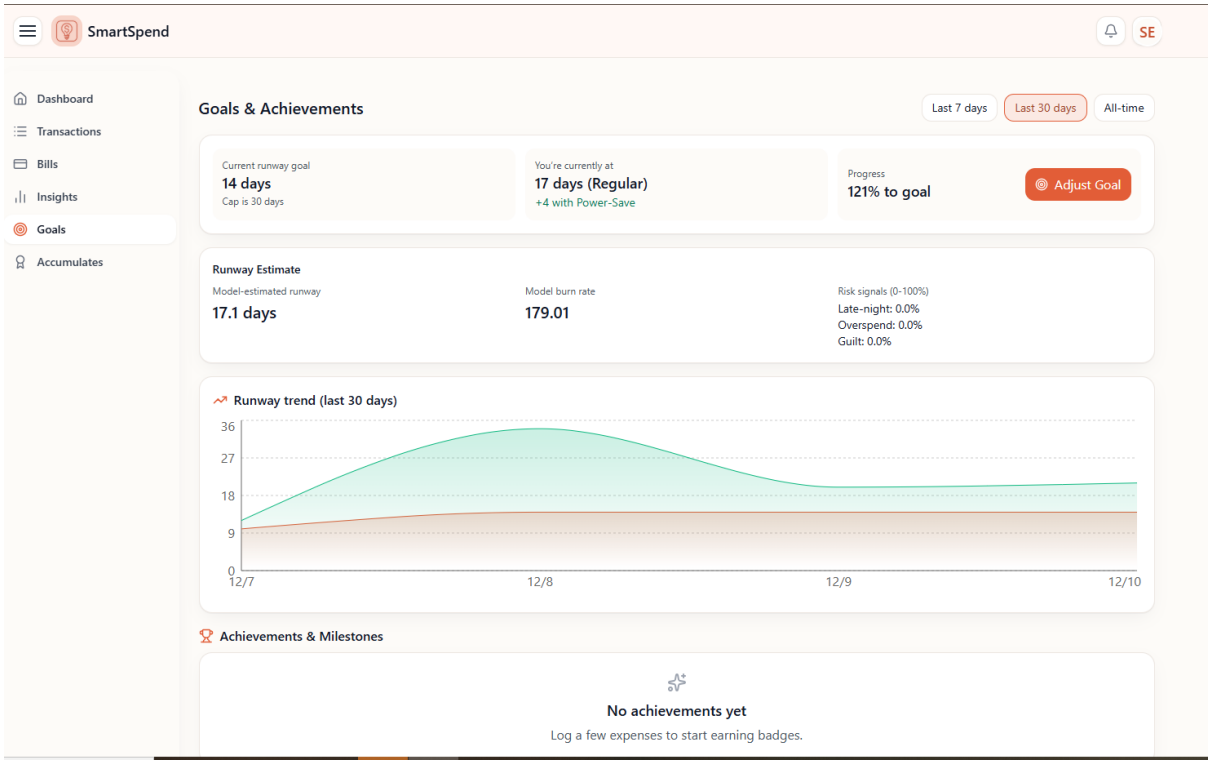


Fig 9.3.15 — Goals & Achievements Dashboard

**Figure 9.3.16 — Adjust Runway Goal Modal**

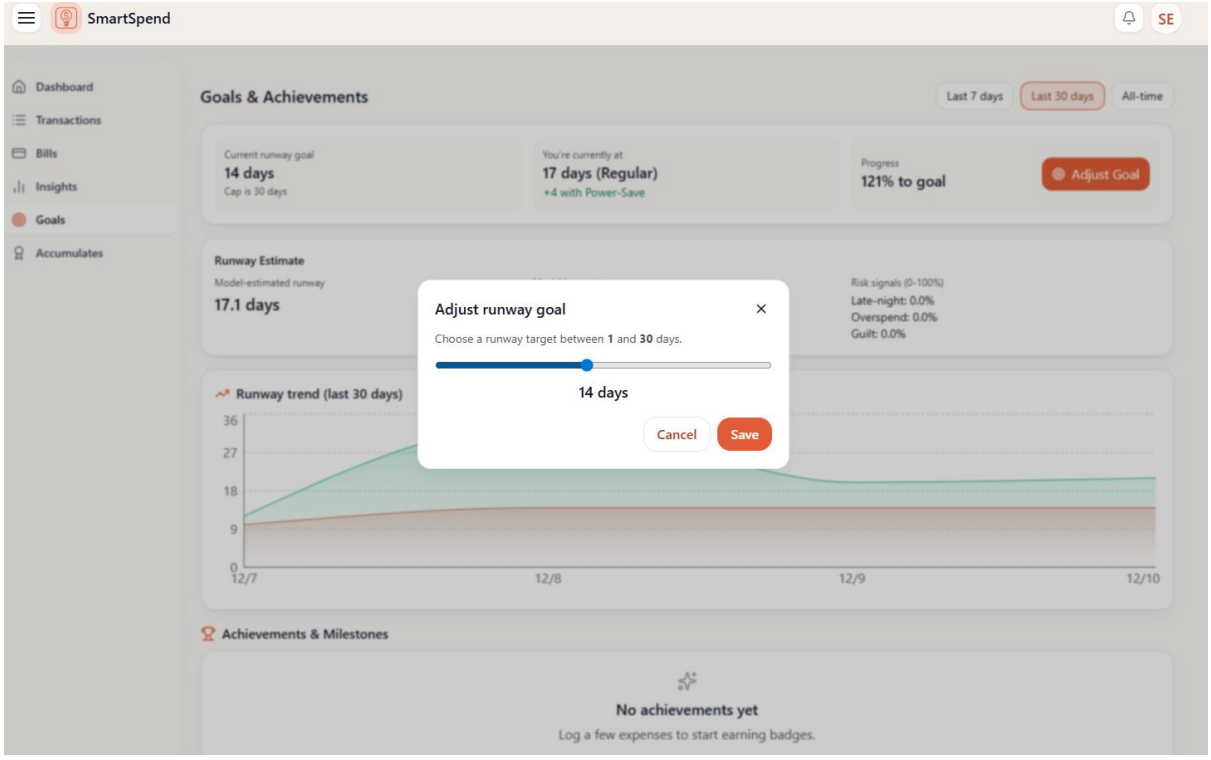


Fig 10.3.16 — Adjust Runway Goal Modal

This figure displays the modal window used to adjust a user's financial runway goal. The interface provides a slider allowing the user to choose a target between 1 and 30 days, offering

full flexibility based on spending habits and savings preferences. The selected value is shown dynamically (e.g., 14 days) to help the user make precise adjustments. The modal includes Save and Cancel options to confirm or discard changes. This feature allows users to continuously refine their goals as their financial behavior evolves.

**Figure 9.3.17 — Accumulates Page (Monthly Savings Summary)**

This figure illustrates the Accumulates section of SmartSpend, which provides a monthly breakdown of how much money the user has saved over time. At the top, a large highlighted card displays the Total Money Saved to date, offering users an immediate sense of their financial progress.

Below this, the Balance_Total table summarizes savings month-by-month for the selected year. Each entry shows the month, year, and the final remaining balance after all expenses and incomes were processed. This helps users understand their financial patterns, track improvements, and compare spending behavior across months. The feature gives a clear historical view of savings trends, supporting better long-term budgeting decisions.

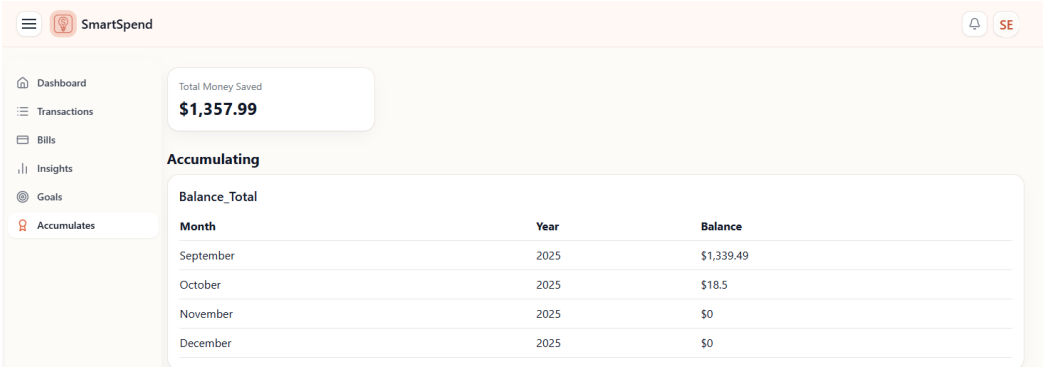


Fig 9.3.17 — Accumulates Page

**Figure 9.3.18 & Figure 9.3.19 — Profile and Account Management**

These figures illustrate the user profile and account settings workflow in SmartSpend. Figure 9.3.18 shows the profile menu accessed from the dashboard, allowing users to navigate to Profile, Notifications, or Logout. Figure 9.3.19 presents the Profile page, where users can view and update personal details such as name, email, timezone, and password. This section ensures secure account management and allows users to customize their preferences for a personalized SmartSpend experience.

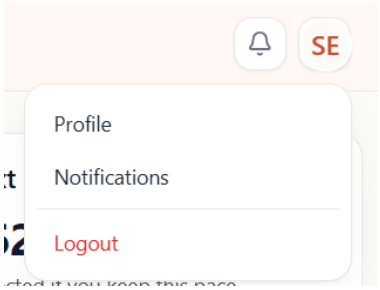


Fig 9.3.18 Profile

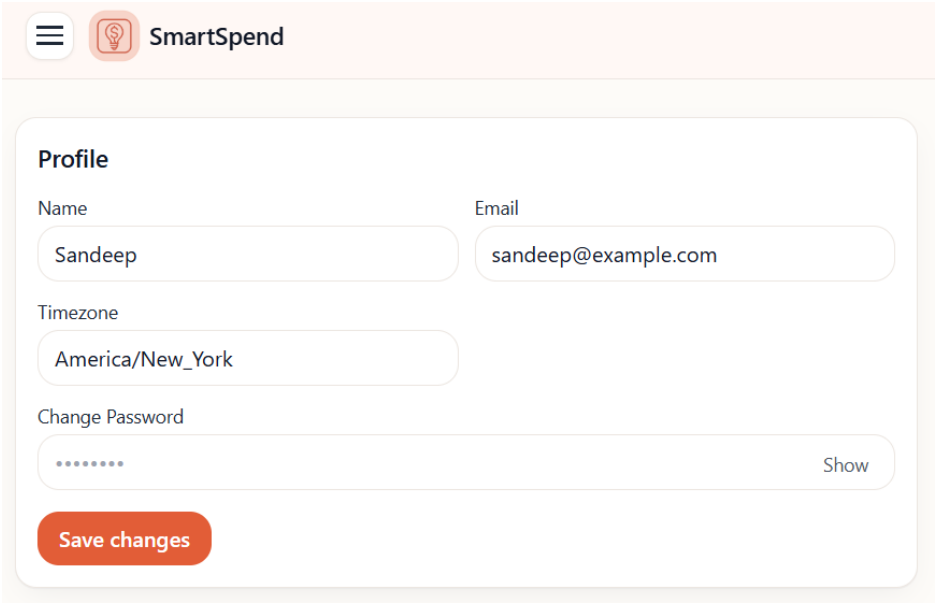


Fig 9.3.19 Account Management

## 10. SAMPLE INPUT/OUTPUT DATA

**Sample Input — Add Expense**

{
"type": "expense",
"amount": 150,
"merchant": "Food",
"category": "Food",
"nwg": "Want",
"mood": "neutral",
"timestamp": "2025-12-10T23:49:00"
}

**Sample Output — Add Expense Response**

{
"id": 91,
"user_id": 1,
"amount": 150,
"category": "Food",

"nwg": "Want",
"mood": "neutral",
"timestamp": "2025-12-10T23:49:00",
"status": "added"
}

**Sample ML Model Input — Burn Rate Prediction**

{
"daily_expenses": [25.5, 30.0, 40.75, 18.90],
"bill_upcoming": 250,
"balance": 2300
}

**Sample ML Model Output — Predicted Burn**

{
"avg_daily_burn": 82.15,
"projected_next7days_burn": 575.05
}

## 11. CONCLUSION

The SmartSpend system successfully demonstrates how financial tracking, behavioral insights, and machine learning can be combined into a single, user-friendly platform. The project achieved all major objectives—from secure authentication and intuitive onboarding to real-time dashboards, NWG-based spending analysis, bill management, and predictive ML models for burn rate and runway estimation.

Through iterative development, testing, and refinement, the system matured into a complete end-to-end solution covering frontend (React), backend (Flask + MySQL), and ML pipelines (Python models). The integration between these components worked smoothly, delivering accurate calculations, real-time updates, and personalized insights for both new and existing users.

User feedback collected during surveys at the Kent State University library, KSU dining areas, and digital forms showed strong approval for the platform's simplicity, visual clarity, and practical usefulness in daily financial decision-making. This validation confirmed that SmartSpend addresses a genuine need—helping individuals understand spending behavior, anticipate financial risks, and make smarter budgeting choices.

In conclusion, SmartSpend stands as a comprehensive financial wellness tool that unifies technology, analytics, and behavioral science. The project not only met its technical milestones but also provided meaningful value to users. With future enhancements such as mobile app support, advanced ML personalization, and AI-driven conversational insights, SmartSpend has the potential to evolve into a powerful, full-scale intelligent finance companion.

## REFERENCES

1. Kim, J. (2021). Machine Learning Approaches for Personal Finance Forecasting. Journal of Financial Data Science.
2. Thaler, R. H., & Sunstein, C. R. (2008). Nudge: Improving Decisions About Health, Wealth, and Happiness. Yale University Press.
3. Kahneman, D. (2011). Thinking, Fast and Slow. Farrar, Straus and Giroux.