IT314 - Software Engineering

Group - 28

Coincious: Smart Expense Tracker and Splitter

System Design Document

Dhirubhai Ambani
University

COMBINED SYSTEM DESIGN:

To manage user security, financial tracking, and intelligent features, the Coincious software is split into three main subsystems.

Subsystem 1: Security & User Management

This subsystem handles secure authentication, user profiles, and session/security management, leveraging Supabase for robust data protection.

Components and Features:

1. **Authentication & Authorization:**
   - Secure user registration and login (Email/Password).
   - Session management via JWT / Supabase Auth.
   - Password reset and recovery flows.
2. **Profile Management:**
   - Management of user details (Name, Avatar, Phone, Preferences).
   - Synchronization of profile data across dashboard, groups, and AI assistant.
3. **Security Enforcement:**
   - Row Level Security (RLS) policies ensuring users access only their own data or data shared in groups.
   - Access control for sensitive operations (delete account, modify group data).

Data Flow:

1. User enters credentials → Auth service verifies with Supabase Auth → Session token created → User redirected to Dashboard.

2. User updates profile → Profile data stored in database → Reflected across Dashboard, Groups, Notifications and AI Assistant.

3. Unauthorized access attempt → RLS policies block the query → Error/denied message shown to the user.

Subsystem 2: Expense Management

This subsystem manages creation, storage, filtering, and retrieval of both **personal** and **group-linked** expenses.

Components and Features:

1. **Expense Creation:**
    - Add personal expenses with title, amount, category, date, and notes.
    - Attach receipt images (upload to Supabase storage).
    - Support for personal vs group mode (toggle).

2. **Expense Storage & Retrieval:**
    - Store expense records with relations to User and optionally Group.
    - Fetch expenses based on date range, category, or type (personal/group).

3. **Expense History & Editing:**
    - View detailed expense list and individual expense records.

  - Edit or delete existing expenses owned by the user.

Data Flow:

1. User submits new expense → ExpenseController validates data → Expense stored in `Expenses` table → Dashboard totals updated.

2. User uploads receipt → File stored in Supabase Storage → URL saved in Expense record.

3. User opens Expense History → Filter parameters sent → Matching expenses queried and rendered in UI.

Subsystem 3: Group & Split Management

This subsystem manages user groups, shared expenses, and splitting logic among group members.

Components and Features:

1. **Group Lifecycle Management:**
   - Create group with name and description.
   - Add/remove members (current scope: same application users).
   - View all groups a user is part of.
2. **Shared Expense Handling:**
   - Add a group expense and associate it with a specific group.
   - Automatically tag expenses as “Group” type.

3. **Split Calculation:**
   - Equal split among all members.
   - Store per-member share (amount owed) in Split entity.
   - Basis for future “settle up” and balance views.

Data Flow:

1. User creates group → GroupController stores `Group` record → Creator set as admin/member → Group appears in Group List.
2. User adds a group expense → Expense saved and linked to Group → Split records generated per member.
3. User views group details → GroupController fetches members, expenses, and splits → Group summary rendered.

Subsystem 4: Dashboard, Budget & Analytics

This subsystem provides a consolidated overview of the user’s finances, including metrics, charts, and budget tracking.

Components and Features:

1. **Overview Metrics:**
   - Display total monthly expense and remaining budget.
   - Show quick stats such as total expenses this month vs previous month.

2. **Budget Management:**
    - ○ User sets monthly budget limit.
    - ○ Track used amount vs limit and show progress (circular chart / progress bar).
    - ○ Trigger alerts when budget exceeds threshold.
3. **Analytics & Visualization:**
    - ○ Category-wise expense breakdown (bar/pie chart).
    - ○ Time-based expense trends using date filters.

Data Flow:

1. User opens Dashboard → Dashboard module aggregates data from Expense & Budget tables → KPI cards and charts are rendered.
2. User sets/updates budget → Budget entity updated → Dashboard recalculates utilization and warnings.
3. User changes date range → Filter passed to Expense subsystem → New aggregated stats returned and visualizations updated.

Subsystem 5: AI Assistant

This subsystem offers intelligent, conversational access to expense data and simple financial insights.

Components and Features:

1. **Natural Language Query Handling:**

    - User asks questions like “Show my expenses for last week”.
    - Understands intent (time range, category, type of query).

2. **Insight Generation:**
    - Summarizes spending by category or time period.
    - Highlights overspending categories and unusual spikes.

3. **Quick Actions & Suggestions:**
    - Suggests filters (“Do you want to see only Food?”).
    - Provides shortcuts to open Dashboard, Expenses, or Groups with applied filters.

Data Flow:

1. User sends message in chat → ChatController forwards text to NLPService / AI API.
2. NLPService extracts intent (e.g., time window, category) → Calls Expense / Dashboard subsystems to fetch required data.
3. Retrieved data → Formatted into natural language answer → Returned to user as chat response.

Subsystem 6: Notifications & Reminders

This subsystem handles system-generated alerts related to budgets, groups, and important account events.

Components and Features:

1. **Notification Generation:**
    - Budget threshold exceeded alerts.
    - Group expense added/updated notifications.
    - Important account/security alerts (e.g., password change).
2. **Notification Delivery & Storage:**
    - Store all notifications in `Notifications` table.
    - Provide unread/read status for each notification.
3. **Notification Center UI:**
    - List all notifications in descending time order.
    - Allow marking notifications as read/cleared.

Data Flow:

1. Event occurs (e.g., expense pushes spending above budget) → NotificationController creates a notification entry.
2. User opens Notifications page → All notifications for that user fetched → Unread ones highlighted.
3. User taps “Mark as read” → Notification status updated → UI refreshes to reflect changes.

Subsystem 7: Help & Support

This subsystem assists users in understanding how to use the application and resolve basic issues.

Components and Features:

1. **FAQ Management:**
   - List of common questions and answers.
   - Grouped into categories: Getting Started, Expenses, Groups, AI Assistant, Security.
2. **Help Content Delivery:**
   - Expandable/collapsible FAQ items.
   - Search or filter FAQs by keyword (optional future enhancement).
3. **Support Contact Entry Point:**
   - Provide “Need more help?” action (mailto/form – future scope).
   - Link to privacy policy and terms.

Data Flow:

1. User opens Help section → FAQ items loaded from static data / database.
2. User expands a question → Answer content displayed inline.
3. User clicks on support/contact link → Redirect to external email or support page (if implemented).

Subsystem 8: Application Settings & Account Controls

This subsystem manages global app preferences and critical account-level actions.

Components and Features:

1. **General Settings:**
    - Toggle UI preferences (e.g., currency format; future: theme).
    - Manage notification preferences.
2. **Account Controls:**
    - Change password, email, and other security-related fields (via Security subsystem).
    - “Danger Zone” actions like deactivate/delete account.
3. **App Information & Policies:**
    - Show app version, build info.
    - Show links to Terms and Privacy Policy.

Data Flow:

1. User changes a setting (e.g., notification preference) → SettingsController updates stored preferences → Affects future behavior of Notifications/UX.
2. User initiates “Delete Account” → Confirmation modal → Security subsystem validates and performs cleanup → User logged out.
3. User views App Info → Static details rendered from config/metadata.