**📘 Project Requirements & Architecture: *Pennywise***

**🧱 1. Project Overview**

**Pennywise** is a cross-platform personal finance tracker designed to simplify expense tracking, group-based budgeting, and financial analytics. It will be accessible via web and mobile (via PWA), with a clean, intuitive interface and collaborative features.

**🧰 2. Tech Stack**

**✅ Frontend**

* **Framework**: [Next.js](https://nextjs.org/)
* **UI Library**: [Material UI](https://mui.com/)
* **Language**: TypeScript
* **PWA Support**: via next-pwa
* **State Management**: Redux
* **Authentication**: Google OAuth 2.0

**✅ Backend**

* **Language**: Python
* **Framework**: FastAPI
* **API Style**: RESTful (JWT-based)
* **Auth Verification**: Google ID Token via google-auth

**✅ Database**

* **DB Engine**: PostgreSQL (hosted on [Supabase Free Tier](https://supabase.com/))
* **Access Control**: Role-based group permissions
* **ORM**: SQLAlchemy or Tortoise ORM

**✅ DevOps & Hosting**

| **Layer** | **Platform** | **Notes** |
| --- | --- | --- |
| Frontend | Vercel | SSR-ready, PWA-friendly |
| Backend | Render/Railway | Python backend hosting |
| Database | Supabase | Free PostgreSQL + Auth & Admin panel |
| Domains/SSL | Vercel / Cloudflare | HTTPS out of the box |

**🧾 3. Functional Requirements**

**3.1 Expense Tracking (Month-wise)**

* Default to **current month view**
* Users can **navigate forward/backward** by month
* All expenses shown in clean list format

**3.2 Group Management**

* Users can **create groups** and invite others via Gmail address
* Invited users must **authenticate via Google**
* Groups act as **containers** for shared expenses
* Group roles: **Admin** (creator), **Member**
* Access control:
  + Admins can modify permissions
  + Members can view/add/edit expenses if allowed

**3.3 Date Navigation**

* Add **month/year jump** using dropdown or calendar selector
* Preserves selected group and view type (monthly, weekly)

**3.4 Main UI**

* Clean, minimalist layout
* Expense/income items listed with summary at top
* Two primary action buttons:
  + **“Cash In”** (Income)
  + **“Cash Out”** (Expense)

**3.5 Add Expense / Income**

Both forms will contain:

* **Amount** (required)
* **Remark/Note**
* **Category** (dropdown)
* **Payment Mode** (e.g., UPI, cash, card)
* Buttons:
  + **Save**
  + **Save & Add New**

**3.6 Reporting & Analytics**

* Users can generate:
  + **Weekly**, **Monthly**, **Yearly** reports
* Reports to include:
  + Totals, category breakdown, income/expense ratio
  + Simple charts using recharts or chart.js
* Analytics (future scope):
  + Trend lines
  + Auto-categorization
  + Savings suggestions

**3.7 Settings Panel**

* User Profile
* Default currency and time zone
* Group management interface
* Export data (CSV)
* Delete account/group

**🧩 4. Non-Functional Requirements**

* Responsive & mobile-first (PWA)
* Performance optimized with SSR/SSG from Next.js
* Secure: token validation, role-based access, HTTPS enforced
* Accessibility: WCAG-compliant UI elements

**🗃️ 5. Data Model Outline (Simplified)**

**User**

* id, name, email, google\_id
* created\_at

**Group**

* id, name, owner\_id
* created\_at

**GroupMember**

* user\_id, group\_id, role (admin/member), permissions

**Transaction**

* id, group\_id, user\_id, amount, note
* type: INCOME or EXPENSE
* category, payment\_mode, date

**🔐 6. Auth Flow (Google OAuth)**

1. User logs in with Google (client-side)
2. Frontend sends Google ID token to backend
3. Backend verifies ID token using Google's OAuth2 API
4. If valid, backend issues own JWT for session use

**🛠️ 7. Tools & Libraries**

* **Frontend**: @mui/material, react-hook-form, react-query, next-pwa
* **Backend**: fastapi, uvicorn, sqlalchemy, google-auth, pydantic
* **Testing**: Jest (frontend), Pytest (backend)
* **API Docs**: Auto-generated via FastAPI's OpenAPI
* **DevOps**: GitHub Actions or Vercel Auto Deploy

**🧪 8. Milestone-Based Development Plan (Optional)**

| **Phase** | **Deliverables** |
| --- | --- |
| **Phase 1** | Basic auth, create group, add expense |
| **Phase 2** | Monthly navigation, basic reports |
| **Phase 3** | PWA, mobile testing, offline cache |
| **Phase 4** | Group permissions, export & settings |
| **Phase 5** | Charts, analytics, polish UI & UX |

**✅ Final Notes**

* The architecture supports both **solo** and **shared budgeting**
* Can scale to multiple users/groups
* Mobile-ready via PWA without separate native app
* Clean UI design focus with future-proof extensibility