**Project Overview: Pennywise – AI-Powered Personal Finance Co-Pilot (Final Updated Plan)**

**1. Project Summary**

**Name: Pennywise  
One-liner: A smart, AI-powered personal finance dashboard that helps users track spending, set savings goals, and receive intelligent money-saving suggestions.  
Purpose: To showcase your Spring Boot + Next.js (JavaScript) skills while solving a real problem young professionals face — understanding and improving their personal finances — with standout features like AI assistance and visual analytics.**

**2. Target Users**

* **Young professionals in Canada aged 22–35**
* **People with limited financial literacy or interest in spreadsheets**
* **Users who want AI to guide their savings decisions**

**3. Core MVP Features (1 Week)**

1. **User Authentication (Register/Login)**
   * **JWT-based auth with cookie/session or localStorage support**
   * **Optional demo login**
2. **Transaction Feed**
   * **Mock data seeded in PostgreSQL (20k+ entries per user)**
   * **Includes date, merchant, amount, category, location**
3. **Dashboard Visualization**
   * **Monthly breakdown of spend by category (pie chart)**
   * **Spend over time (bar/line chart)**
4. **Savings Goal Setup**
   * **User inputs monthly income, goal target, and categories to focus on**
5. **GPT-4o Advice Generation**
   * **OpenAI API call processes last 30 days of user spend**
   * **Returns 2–3 suggestions in plain English**
6. **Responsive UI / UX**
   * **Built with Tailwind + shadcn/ui**
   * **Dark/light mode toggle**
   * **Mobile-first design**
7. **Deployment**
   * **Vercel for frontend**
   * **Render for backend**
   * **CI/CD via GitHub Actions**

**4. Stretch Enhancements (Future-Ready Design)**

* **PostGIS Heatmaps: Store lat/lng for each transaction to enable geo-based visualization later**
* **Redis Pub/Sub: Abstract a NotificationService to enable future real-time dashboard refreshes**
* **Kafka Stream Simulation: Create a TransactionIngestionService to simulate Kafka-based ingestion**
* **Terraform + AWS ECS: Structure deploy/ folder to support future infrastructure-as-code and scalable hosting**
* **React Native Shell: Keep the front-end decoupled with REST APIs to allow mobile integration in the future**

**5. Backend Stack (Spring Boot)**

* **Java 21 + Spring Boot 3.5.0**
* **Maven**
* **PostgreSQL (hosted on Render or Railway)**
* **REST API**
* **JWT Auth (via Spring Security)**
* **OpenAI integration (GPT-4o)**

**Core Packages:**

* **controller – REST endpoints**
* **service – business logic (AdviceService, TransactionService, etc.)**
* **model – JPA entities**
* **repository – Spring Data JPA**
* **config – CORS, security, OpenAI config**
* **notification – placeholder for Redis pub/sub**
* **ingestion – placeholder for Kafka streaming**

**6. Frontend Stack (Next.js 15)**

* **JavaScript (no TypeScript)**
* **App Router**
* **Tailwind CSS + shadcn/ui**
* **Chart.js or Recharts**
* **Axios or fetch for API calls**

**Core Pages:**

* **/login, /register, /dashboard, /settings, /advice**
* **Layout with Navbar, Sidebar, and Toast feedback**
* **Protected routes using JWT stored in cookies or localStorage**

**7. Database Schema (Lean + Extendable)**

**users**

* **id (UUID), email, password\_hash, created\_at**

**transactions**

* **id, user\_id, date, amount, category, merchant, lat, lng, description**

**goals**

* **id, user\_id, target\_amount, category\_focus, start\_date, end\_date**

**8. REST API Endpoints**

* **POST /api/auth/register – register user**
* **POST /api/auth/login – login user**
* **GET /api/transactions – list user transactions**
* **POST /api/transactions – add a transaction**
* **GET /api/charts/summary – chart data by category & month**
* **GET /api/advice – return GPT-4o suggestions**
* **POST /api/goals – set/update savings goal**

**9. Folder Structure**

**pennywise-app/**

**├── backend/**

**│ ├── src/main/java/com/pennywise/**

**│ │ ├── controller/**

**│ │ ├── service/**

**│ │ │ ├── advice/**

**│ │ │ ├── transaction/**

**│ │ │ ├── notification/ ← future Redis pub/sub**

**│ │ │ ├── ingestion/ ← future Kafka streaming**

**│ │ ├── model/**

**│ │ ├── repository/**

**│ │ ├── config/**

**│ ├── application.yml**

**│ └── Dockerfile**

**├── frontend/**

**│ ├── app/**

**│ │ ├── dashboard/**

**│ │ ├── transactions/**

**│ │ ├── advice/**

**│ │ ├── settings/**

**│ └── public/**

**│ └── tailwind.config.js**

**├── deploy/**

**│ ├── terraform/**

**│ ├── render.yaml**

**│ ├── ci.yaml**

**├── docs/**

**│ ├── project-plan.md**

**│ ├── changelog.md**

**│ ├── current-task.md**

**10. Deployment Plan**

* **Backend: Render service (PostgreSQL + JAR build)**
* **Frontend: Vercel (auto-deploy via GitHub push)**
* **GitHub Actions CI pipeline**
* **Health check endpoint (/health) + uptime badge in README**

**11. Budget & Cost Awareness**

|  |  |  |
| --- | --- | --- |
| **Tool/Service** | **Free Tier?** | **Notes** |
| **Next.js + Tailwind** | **✅ Free** | **Open-source frontend stack** |
| **Vercel** | **✅ Free (Hobby)** | **Ideal for portfolio hosting** |
| **Spring Boot** | **✅ Free** | **Open-source** |
| **Java 21** | **✅ Free** | **LTS, no restrictions** |
| **PostgreSQL (Render)** | **✅ Free (256MB)** | **Paid if DB exceeds usage** |
| **Render backend** | **✅ Free (750 hrs)** | **Paid if uptime/bandwidth exceeds** |
| **OpenAI API (GPT-4o)** | **❌ Paid** | **~$0.005 per query; can mock in dev** |
| **GitHub Actions** | **✅ Free (2k min)** | **Works great for CI/CD** |
| **Redis, Kafka, AWS** | **✅ Limited/free** | **Only needed later for advanced scaling** |
| **Terraform CLI** | **✅ Free** | **Paid only for Terraform Cloud** |

**12. Résumé Pitch Example**

**Built a full-stack AI-powered personal finance dashboard using Spring Boot 3.5.0, Java 21, PostgreSQL, and Next.js 15 (JavaScript). Simulated real-world bank feeds and delivered GPT-4o savings advice with interactive visualizations. Designed a future-ready architecture supporting streaming data, pub/sub updates, and geo analytics.**

**13. Key Outcomes for Recruiters**

* **Demonstrates modern full-stack dev skills (Spring + Next.js)**
* **Includes a real-world problem + valuable solution**
* **Uses AI + charts to elevate beyond CRUD**
* **Clean structure for DevOps, security, and future extensibility**
* **Publicly deployed + polished UI/UX**
* **Prepared to scale with PostGIS, Kafka, Redis, Terraform, and mobile extensions in future iterations**