Finance Tracker Web Application

# Overview

A web application to track personal finances, available to all users with basic functionality, while offering premium features for subscribed users. Technologies used:

- Frontend: Angular

- Backend: Spring Boot (Java)

- Database: PostgreSQL

# Features

Basic Features (Free Users):

- Add/Edit/Delete income entries

- Add/Edit/Delete expenses

- Categorize income/expenses

- View reports (monthly/yearly summary)

- Manual data entry

Premium Features (Subscribed Users):

- Auto reset of salary every month

- Automatic deduction of fixed expenses (rent, food, etc.)

- Budget recommendations

- Visual analytics (charts, predictions)

- Notifications and reminders

# Step-by-Step Implementation

# 1. Database Design (PostgreSQL)

Tables:

- users: id, name, email, password\_hash, is\_subscribed, created\_at

- incomes: id, user\_id, amount, category, date, description

- expenses: id, user\_id, amount, category, date, description, recurring

- subscriptions: id, user\_id, start\_date, end\_date, payment\_status

- settings: id, user\_id, salary\_amount, auto\_reset, fixed\_expenses (JSON)

Relationships:

- One user can have many incomes and expenses

- One user has one subscription record

# 2. Backend Development (Spring Boot)

Setup:

- Create Spring Boot project with dependencies: Spring Web, Spring Data JPA, Spring Security, PostgreSQL Driver

Structure:

- Controllers: UserController, IncomeController, ExpenseController, SubscriptionController

- Services: Business logic (e.g., IncomeService, ExpenseService, SubscriptionService)

- Repositories: JPA repositories for all entities

- Entities: Map the DB tables to JPA entities

- Scheduled Jobs: Use @Scheduled annotation for monthly resets

APIs:

- POST /api/auth/register

- POST /api/auth/login

- GET /api/user/profile

- CRUD for incomes and expenses

- POST /api/subscription/start

- GET /api/analytics/monthly-summary

Security:

- Use Spring Security with JWT-based authentication

- Role-based access to restrict premium endpoints

# 3. Frontend Development (Angular)

Setup:

- Create Angular project

- Install libraries: @angular/material, ngx-charts, jwt-decode, etc.

Structure:

- Modules: auth, dashboard, subscription, analytics

- Components: LoginComponent, RegisterComponent, IncomeListComponent, ExpenseListComponent, SubscriptionComponent, ChartsComponent

- Services: AuthService, IncomeService, ExpenseService, SubscriptionService

Features:

- Forms to add/edit/delete income/expenses

- Subscription management UI

- Dashboard with monthly summary and charts

- Conditional rendering of premium features

Routing:

- /login, /register, /dashboard, /subscription, /analytics

- Use AuthGuard and RoleGuard to secure routes

# 4. Subscription & Payments (Optional Integration)

- Integrate Razorpay/Stripe for payments

- Update is\_subscribed and subscription tables post-payment

- UI feedback on successful subscription

# 5. Scheduled Tasks (Auto Salary Reset)

- Use @Scheduled(cron = "0 0 0 1 \* ?") in Spring Boot to reset salary

- Auto-deduct fixed expenses by reading settings JSON

- Notify user via email or in-app alert

# Deployment

Frontend:

- Build Angular using ng build --prod

- Host on Firebase, Netlify, or static S3 bucket

Backend:

- Package Spring Boot app as JAR

- Host on AWS EC2, Render, or Heroku

Database:

- Use PostgreSQL instance from AWS RDS, Supabase, or local setup

# Future Enhancements

- AI-based spending suggestions

- Export data to CSV/Excel

- Multi-language support

- Mobile app version

# Conclusion

This document outlines the full-stack development plan for a Finance Tracker web app, balancing free and premium user functionality. By using Angular for the frontend, Spring Boot for the backend, and PostgreSQL for storage, the app ensures modern development standards and scalability.