**Personal Finance & Schedule Manager**

A full-stack, responsive web application designed for freelancers and gig workers to track work schedules, manage expenses, and visualize financial health.

# **🚀** Features

## • Authentication & Security

• Secure Email/Password registration with OTP verification.

• Social Login via Google and GitHub (Passport.js).

• JWT-based session management.

• Rate limiting and Data sanitization against NoSQL injection.

## • Schedule Management

• Add hourly or fixed-salary work schedules.

• Recurring weekly schedule support.

**• Bulk Upload**: Import schedules via Excel/CSV with data validation preview.

## • Expense Tracking

• Log daily expenses with categories.

• Upload and store digital receipts.

## • Interactive Dashboard

• Real-time KPIs for Weekly/Monthly Income, Expenses, and Net Profit.

• Visual charts for income trends and expense breakdowns (Recharts).

## • Reporting

• Generate and download custom reports in **PDF, CSV, and Excel (XLSX)** formats.

• Custom date range filtering.

# **🛠** Tech Stack

## Frontend

**• Framework**: React (Vite) with TypeScript

**• Styling**: Tailwind CSS, Shadcn UI

**• State/Fetching**: TanStack Query (React Query), Context API

**• Forms**: React Hook Form, Zod validation

**• Charts**: Recharts

## Backend

**• Runtime**: Node.js, Express.js

**• Database**: MongoDB (Mongoose ODM)

**• Auth**: Passport.js, JSON Web Token (JWT)

**• Email**: Resend API

**• Files**: Multer (local storage for receipts/uploads)

**• PDF/Excel**: PDFKit, xlsx

## DevOps

**• Containerization**: Docker, Docker Compose

**• Deployment Ready**: Render (or any container-based host)

# **📸** Gallery

**A screenshot of a login form

AI-generated content may be incorrect.A screenshot of a login form

AI-generated content may be incorrect.**

# **A person standing in front of a sign AI-generated content may be incorrect.A screenshot of a graph AI-generated content may be incorrect.A screenshot of a computer AI-generated content may be incorrect.A screenshot of a schedule AI-generated content may be incorrect.A screenshot of a schedule AI-generated content may be incorrect.A screenshot of a computer AI-generated content may be incorrect.A screenshot of a black screen AI-generated content may be incorrect.A screenshot of a computer AI-generated content may be incorrect.A screenshot of a computer AI-generated content may be incorrect.A screenshot of a computer AI-generated content may be incorrect.A screenshot of a black and purple report AI-generated content may be incorrect.**

# **🏁** Getting Started

## Prerequisites

• Node.js 18+

• MongoDB (Local or Atlas URI)

• Docker Desktop (optional, recommended for easiest setup)

## Environment Variables

Create a .env file in the root directory:

# --- Server Configuration ---

NODE\_ENV=development

PORT=5050

DATABASE\_URL=mongodb+srv://your-mongo-uri

JWT\_SECRET=your\_super\_secret\_jwt\_key

JWT\_EXPIRY=7d

# --- External Services ---

RESEND\_API\_KEY=re\_your\_resend\_key

# --- OAuth (Google/GitHub) ---

GOOGLE\_CLIENT\_ID=your\_google\_id

GOOGLE\_CLIENT\_SECRET=your\_google\_secret

GITHUB\_CLIENT\_ID=your\_github\_id

GITHUB\_CLIENT\_SECRET=your\_github\_secret

# --- URLs (Important for CORS and OAuth Redirects) ---

# For local dev without Docker: http://localhost:5173 and http://localhost:5050

# For Docker local: http://localhost:8080 and http://localhost:5050

CLIENT\_URL=http://localhost:8080

SERVER\_URL=http://localhost:5050

## Option 1: Running with Docker (Recommended)

Build and start the entire application (frontend + backend + database networking) with one command:

docker-compose up --build

• Frontend: http://localhost:8080

• Backend API: http://localhost:5050

**Option 2: Running Locally (Manual)**

## Start Backend

cd server

npm install npm run server

## Start Frontend

cd client

npm install

npm run dev

Frontend: http://localhost:5173

Backend: http://localhost:5050

# **📖** API Documentation

Full API documentation is available via Swagger UI when the server is running.

**Local URL**: http://localhost:5050/api/docs

# **📂** Project Structure

├── client/ # React frontend

│ ├── src/

│ │ ├── api/ # Axios setup

│ │ ├── components/ # Reusable UI components │ │ ├── context/ # Auth global state

│ │ ├── pages/ # Main application views

│ │ └── lib/ # Utilities (tailwind merge, etc.)

│ └── Dockerfile

├── server/ # Express backend

│ ├── config/ # DB and Passport config

│ ├── middleware/ # Auth, upload, and error handling

│ ├── models/ # Mongoose schemas

│ ├── routes/ # API endpoints

│ ├── services/ # Email service (Resend)

│ ├── uploads/ # Local storage for receipts

│ └── Dockerfile

└── docker-compose.yml # Container orchestration