FULL STACK WEB DEVELOPER – TECHNICAL ASSESSMENT

DURATION: 2 Days (48 hours) from the time of receiving the assessment. .

Project Title: "Subscription Management Dashboard"

Objective:
Build a **mini SaaS admin dashboard** that allows users to subscribe to a plan, view their active plan, and manage their profile — with a clean and responsive UI.

Tech Stack

- **Frontend:** React.js (Vite or CRA), TailwindCSS, Redux Toolkit / Zustand
- **Backend:** Node.js + Express.js
- **Database:** PostgreSQL / MongoDB
- **ORM / Query Builder:** Knex.js / Prisma / Mongoose
- **Authentication:** JWT + Refresh Tokens
- **Version Control:** Git (Public GitHub repo)

Backend Requirements (Node.js + Express)

1. **Authentication & Authorization**
    - Implement **JWT authentication** with access & refresh tokens.
    - Implement middleware for **role-based access** (`admin`, `user`).

2. **Subscription Module**
    - Tables/Collections:
        - `plans`: id, name, price, features[], duration (in days)
        - `users`: id, name, email, password, role
        - `subscriptions`: id, user_id, plan_id, start_date, end_date, status
    - APIs:
        - `POST /api/auth/register`
        - `POST /api/auth/login`
        - `GET /api/plans`
        - `POST /api/subscribe/:planId` (authenticated user subscribes)
        - `GET /api/my-subscription` (returns user's active plan)
        - `GET /api/admin/subscriptions` (admin only – all subscriptions list)

3. **Validation & Error Handling**
    - Validate payloads using Joi/Yup/Zod.
    - Return structured error responses with status codes.

4. **Database Seeding**
    - Seed 3–4 sample plans with realistic data.

Frontend Requirements (React.js)

1. **Pages:**
    - `/login` — User login
    - `/register` — User registration
    - `/plans` — List all available plans
    - `/dashboard` — Show user's current plan details
    - `/admin/subscriptions` — Admin dashboard listing all subscriptions

2. **Core Functionality:**
    - Use **Redux Toolkit or Zustand** for global state (auth + user data).
    - Store JWT and refresh tokens securely.
    - Implement auto-logout or silent token refresh.
    - Protect routes based on roles (`/admin/*` accessible only for admin).

3. **UI & UX:**
    - Responsive and professional layout (Tailwind or CSS Modules).
    - Include navigation bar, logout button, and user menu.
    - Show subscription status (active, expired, or none).

Bonus Points (Optional)

- Integrate **Stripe or Razorpay test mode** for payment simulation.
- Implement **dark/light theme toggle**.
- Add **plan upgrade/downgrade** logic.
- Deploy both frontend & backend (Vercel / Render / Railway).

Submission Instructions

1. Create a **public GitHub repository** named `subscription-dashboard-task`.
2. The repo should contain:
    - `/client` – Frontend
    - `/server` – Backend
3. Add a **README.md** with:
    - Setup and run instructions
    - Tech stack used
    - Your name and contact details
4. Submit the GitHub repository link.

Evaluation Criteria

Area	Description	Weight
Code Structure	Clean, modular, well-organized backend & frontend	25%
Functionality	Meets all required features	25%
UI & UX	Clean, responsive, and user-friendly design	20%
API Integration	Smooth frontend-backend communication	20%
Bonus Features	Filters, deployment, extra creativity	10%