Fullstack Developer Assignment

> **Spaces by Fanpit** is a novice platform that acts as a networks for listing event spaces, co-working areas, and casual third spaces where anyone can simply walk-in, get productive, or just hang out.

1 | Task Overview

To develop a production-ready prototype that demonstrates the full booking life-cycle while showcasing an eye for user experience, clean code, and a solid architecture.

2 | Objectives & Scope

Develop and deploy a minimal yet functional platform where:

- **Consumers** browse spaces, reserve time-slots, complete payment via Razorpay, and check-in/out with ease.
- **Brand Owners** list & manage their own spaces, configure `sophisticated` pricing models, and monitor reservations.
- **Staff** validate check-ins and supervise day-to-day occupancies.

3 | User Roles & Authorization

Role	Core Capabilities
Consumer	Browse spaces (public route), sign-up/login, see real-time availability (protected), place reservations, pay via Razorpay, view/cancel bookings.
Brand Owner	CRUD own spaces, upload images, configure pricing rules, view/filter reservations, mark no-shows, basic analytics.
Staff	View daily reservation list, scan/enter booking code, mark guests checked-in/out, flag issues.

Role-based access control must be enforced in **both the API and the UI**.

-

4 | Core Functional Requirements

1. **Authentication**
    - Email + password, JWT access + refresh tokens, password reset.
2. **Space Management** (*Brand Owner*)
    - Fields → name, description, address (Google Maps or Open Street Maps Search), capacity, amenities, images (these are just basic fields)
    - **Pricing Engine** → free, hourly/day rates, peak/off-peak multipliers, **time-block bundles, monthly passes, promo codes**, and the ability to override pricing for special events.
3. **Reservation Workflow** (*Consumer*)
    - Availability calendar, dynamic price breakdown, checkout with Razorpay (test keys), confirmation e-mail.
4. **Check-In Dashboard** (*Staff*)
    - Real-time list, search/filter, quick status update (checked-in, no-show, checked-out).
5. **Payments & Refunds**
    - Webhook handling for successful, failed, or refunded payments; status must stay consistent in MongoDB.

5 | Tech Stack (*must be used)*

Layer	Tech
Frontend	**Next.js 15** (App Router, SSR/SSG/ISR), TypeScript, Tailwind CSS, React-Hook-Form/Zod. Deployed on **Vercel**.
Backend	**NestJS** (Node 18, TypeScript) + **Mongoose** ODM. **MongoDB Atlas** connection string will be supplied. **REST API**
Payments	**Razorpay**

Make sensible use of **Server-Side Rendering (SSR)** and other Next.js best practices (e.g., route-level caching, edge functions where relevant).

-

6 | Deliverables

#	Stage	Item	Details
1	Start	**GitHub repository(ies)**	Invite `@fanpitapp`, `@fanpitabd`, `@sanjai-balajee` as collaborator
2	Start & Incremental	**README**	**Start** - Setup & approach; **Incrementally** - env vars, architectural decisions, known limitations, "What I'd do next" section.
3	End	**API collection**	Postman/Insomnia file covering all endpoints with pre-filled JWT tokens.

7 | Evaluation Criteria

- **Code quality & structure** (SOLID, modularity, linting, typed)
- **Security** (Auth Checks, rate-limits, input validation)
- **Database design & query efficiency**
- **UI/UX polish & responsiveness**
- **Edge-case handling** (over-booking, payment failures)

9 | Stretch Goals (Bonus)

- iCal / Google Calendar export.
- Admin analytics dashboard (charts).
- Misc.

10 | Resources

1. **https://events.fanpit.live** – Reference application
2. Razorpay Docs – https://razorpay.com/docs/
3. NestJS Docs – https://docs.nestjs.com/

4. Next.js 15 – https://nextjs.org/

Submission

Email demo URL to **tech@fanpit.live**

Good luck!