

WeframeTech Backend Hiring Task

Walkthrough of task

<https://youtu.be/oZX6h0DFqTw>

Deadline: **Before 6 PM IST on the 14th August**

Note:

If you have finished watching the YouTube video, you will know what assignment we have prepared. Don't worry if you have never touched Payload CMS before. It is a free, open-source Node.js backend you can explore at <https://payloadcms.com>. It is developer-friendly, highly customizable, and can even be deployed on Vercel at no cost.

We are not expecting you to arrive as a Payload expert, but we are expecting you to learn fast. This challenge is designed to throw you into something new and unfamiliar, because that is exactly the kind of environment you will face here.

We also know the reality: as AI tools get better, anyone can generate a piece of code. That makes it harder than ever for companies to separate someone who can ship solutions from someone who can just paste snippets. Keeping that in mind, we have built this task so that if you rely on AI for every single thing, it will trip you up. You will have to show your actual skills, your reasoning, your debugging, and your ability to connect the dots when the documentation runs out.

This is not about pulling a ready-made repo from GitHub or polishing a side project for show. It is about demonstrating the mindset, grit, and problem-solving ability you will bring to real-world, production-level work—the kind where Stack Overflow does not have your exact answer and ChatGPT will not save you.

Yes, you will probably Google. Yes, you will hit dead ends. And yes, you might have to throw out your first approach entirely. That is the point. This is what actual engineering looks like when things get tough.

If you are ready for a role where you will grow quickly, take ownership, and make an impact from day one, here is your shot. Show us not just that you can code, but that you can think, adapt, and deliver.

Backend Developer Recruitment Task – Advanced Payload CMS Challenge 2 (Duration:2 days)

Multi-Tenant Event Booking System with Waitlist, Notifications, and Organizer Dashboard

Summary

You are building a **multi-tenant event booking backend** using Payload CMS. Multiple organizations (**tenants**) can manage their own events, users, and bookings, with complete data isolation between tenants. The system must enforce event capacity limits, automatically move users to a waitlist when events are full, and promote the oldest waitlisted booking when a seat becomes available. Every booking status change (confirmed, waitlisted, canceled, promoted) must generate an **in-app notification** for the affected user and a corresponding **log entry** for tracking. All business rules, including multi-tenancy and access control, must be enforced on the backend using Payload access control and hooks.

Organizers should also have access to a **Dashboard Page & API** that shows their upcoming events with booking counts, circular progress indicators for capacity usage, a summary of overall booking statistics, and a recent activity feed from booking logs. Attendees can book events, view their own bookings and notifications, and mark notifications as read, but cannot see other users' data. The solution must include proper API endpoints, seed data for testing, and a short demo video showing the booking flow, waitlist promotion, notifications, and dashboard in action.

Overview

You will build a **multi-tenant** backend using **Payload CMS** where multiple organizations (**tenants**) can manage events and bookings.

The system must:

1. **Support multiple tenants** – each tenant's data is isolated.
2. **Enforce booking capacity** – if the event is full, place users on a **waitlist**.
3. **Automatically promote** waitlisted users when seats become available.
4. **Create in-app notifications** every time a booking's status changes.
5. **Log all booking actions** for tracking.
6. Provide an **Organizer Dashboard API** with:
 - Event counters
 - Circular progress data
 - Summary analytics

Multi-Tenancy

- Every record (Users, Events, Bookings, Notifications, BookingLogs) must be linked to a **Tenant**.
- Users can **only** access data belonging to their tenant.
- All access control must be enforced in **Payload backend hooks**.

Collections to Create

1. Tenants

- `name` (string)
- `createdAt` (auto-generated)

2. Users

- `name` (string)
- `email` (string, unique)
- `role` (`attendee` | `organizer` | `admin`)
- `tenant` (relation → Tenants)

3. Events

- `title` (string)
- `description` (rich text)
- `date` (datetime)
- `capacity` (number)
- `organizer` (relation → Users)
- `tenant` (relation → Tenants)

4. Bookings

- `event` (relation → Events)
- `user` (relation → Users)
- `status` (`confirmed` | `waitlisted` | `canceled`)
- `createdAt` (auto-generated)
- `tenant` (relation → Tenants)

5. Notifications

- `user` (relation → Users)
- `booking` (relation → Bookings)
- `type` (`booking_confirmed` | `waitlisted` | `waitlist_promoted` | `booking_canceled`)
- `title` (short text)
- `message` (long text)
- `read` (boolean, default: `false`)
- `createdAt` (auto-generated)
- `tenant` (relation → Tenants)

6. BookingLogs

- `booking` (relation → Bookings)
- `event` (relation → Events)
- `user` (relation → Users)
- `action` (`create_request` | `auto_waitlist` | `auto_confirm` | `promote_from_waitlist` | `cancel_confirmed`)
- `note` (string)

- `createdAt` (auto-generated)
- `tenant` (relation → Tenants)

Booking Flow

1. Creating a Booking

- When a booking is made:
 1. Count confirmed bookings for the event.
 2. If `confirmedCount < capacity`:
 - Set status = **confirmed**.
 - Create `Notification` (`booking_confirmed`).
 - Create `BookingLog` (`auto_confirm`).
 3. If event is full:
 - Set status = **waitlisted**.
 - Create `Notification` (`waitlisted`).
 - Create `BookingLog` (`auto_waitlist`).

2. Canceling a Booking

- When a confirmed booking is canceled:
 1. Mark booking as **canceled**.
 2. Create `Notification` (`booking_canceled`).
 3. Create `BookingLog` (`cancel_confirmed`).
 4. Find **oldest waitlisted booking** for the same event.
 5. Promote it to **confirmed**.

6. Create `Notification` (`waitlist_promoted`).
7. Create `BookingLog` (`promote_from_waitlist`).

3. Notifications for Status Changes

- Any time a booking's **status** changes (confirmed, waitlisted, canceled, promoted):
 - Create a new **Notification** for the affected user.
 - This must be done **automatically in Payload hooks**.

Organizer Dashboard

Create an overview page inside the CMS

For the **logged-in organizer's tenant**, return:

1. Upcoming Events:

- List of events with:
 - `title`
 - `date`
 - `capacity`
 - `confirmedCount`
 - `waitlistedCount`
 - `canceledCount`

2. Circular Progress Data for each event:

- `percentageFilled = (confirmedCount / capacity) * 100`

3. Summary Analytics:

- Total events

- Total confirmed bookings
- Total waitlisted bookings
- Total canceled bookings

4. Recent Activity Feed:

- Last 5 **BookingLogs** in the tenant.

Access Rules

- **Attendee:**
 - Can only book for themselves.
 - Can only view their own bookings and notifications.
- **Organizer:**
 - Can manage events for their tenant.
 - Can see all bookings for their tenant's events.
- **Admin:**
 - Full access for their tenant.
- **Cross-tenant access is forbidden.**

Endpoints to Implement

1. **POST /api/book-event**
 - Body: { eventId }
 - Creates booking (confirmed or waitlisted) and triggers notifications/logs.
2. **POST /api/cancel-booking**
 - Body: { bookingId }
 - Cancels booking, promotes waitlist if needed, triggers notifications/logs.

3. `GET /api/my-bookings`
 - Lists bookings for the logged-in user.
4. `GET /api/my-notifications`
 - Lists unread notifications for the logged-in user.
5. `POST /api/notifications/:id/read`
 - Marks notification as read.
6. `GET /api/dashboard`
 - Returns organiser dashboard data.

Deliverables

- Private GitHub repository(Give Access to @sambitraze) with:
 1. Complete Payload CMS setup.
 2. `.env.example`.
 3. **Seed script** that:
 - Creates 2 tenants.
 - Each tenant has:
 - 1 organiser.
 - 3 attendees.
 - 2 events with different capacities.
 4. All collections, hooks, and endpoints implemented.
- **Short unedited video (5–8 min)** showing:
 1. Booking flow until an event is full → user becomes waitlisted.
 2. Cancelling a confirmed booking →the oldest waitlisted booking is promoted.

3. Notifications are generated for each status change.
4. Organiser dashboard showing progress bars, analytics, and recent activity.
5. Access control prevents cross-tenant access.

Evaluation Criteria

Area	Weight
Multi-tenancy enforcement	20%
Booking + waitlist logic	20%
Status-change notifications	20%
Organizer dashboard API	20%
Access control	10%
Code clarity & README	10%

Submission Expectations (Private)

Please submit the following in a **private GitHub repository** (do not share it publicly or on social platforms such as LinkedIn):

What to include:

1. Codebase

A working Payload CMS project with

2. README with:

- **Setup instructions** (step-by-step installation and running)
- **Brief explanation of the architecture** (file/folder structure, plugin logic)
- **Sample workflows** (examples of how workflows work)
- **Demo credentials** (for both admin and reviewer roles)

- **Deployment guide** (how to deploy to Vercel or other platforms)

3. 🚨 **MANDATORY: Loom Walkthrough Video**

CRITICAL SUBMISSION REQUIREMENT:

Your Loom video submission **MUST** include:

- ✓ **Your face must be visible throughout the video** (camera on)
- ✓ **Proper task overview and demonstration** (2-3 minutes minimum)
- ✓ **Clear explanation of your approach and architecture**
- ✓ **Live demonstration of all features working**
- ✓ **Screen recording of the admin interface and workflow system**

⚠️ **REJECTION CRITERIA:**

- Video without your face visible = **AUTOMATIC REJECTION**
- Video without proper task overview = **AUTOMATIC REJECTION**
- Video that's just screen recording without explanation = **AUTOMATIC REJECTION**
- Generic or rushed walkthrough = **AUTOMATIC REJECTION**

What to cover in your Loom video:

1. **Introduction** (with your face visible)
2. **Architecture explanation** (show your code structure)
3. **Live demo** (payload)
4. **Technical challenges** you faced and how you solved them
5. **Bonus features** you implemented (if any)
6. **Deployment demonstration** (show it working live)

Important Notes:

- Keep the repository **private**.
- Do **not** share this project on LinkedIn or any social media platforms.
- Please grant repo access to: **majhisambit2@gmail.com**

🧠 **Evaluation Criteria:**

- 🛠️ **Architecture & modularity**
- 🔁 **Reusability & dynamic config**
- 🔒 **Security & role-based access**
- ⚙️ **Payload-specific expertise**
- 🧩 **Problem-solving under constraint**

-  **Quality of Loom presentation and explanation**
-

Recommended Learning Path

Day 1 Preparation:

1. **Morning (2-3 hours):** Set up Payload locally, go through basic tutorials
2. **Afternoon (2-3 hours):** Study collections, hooks, and plugins documentation
3. **Evening (1-2 hours):** Plan your architecture and approach

Development Day:

1. **Early Morning:** Set up project structure and basic collections
 2. **Morning:** Implement core logic
 3. **Afternoon:** Build admin UI components and integration
 4. **Late Afternoon:** Implement APIs and testing
 5. **Evening:** Create Loom video and finalize submission
-

Submit your assignment here

<https://forms.gle/B7DMe7ZSoVUQ7fLo7>


Why Join Us at WeFrametech?


At WeFrametech, we're a fully remote company working with global clients and building digital products using the latest technologies. We're not just another dev shop — we focus on creating solutions that actually matter, used by millions of people across the world.


We're a young, dynamic team that values creativity, accountability, and integrity above all else. We believe great teams are built on trust and ownership, not just skillsets.


If you're looking for a place where your work has real impact and you're constantly learning and growing — you'll love it here.


What to Expect When You Join weframetech:


 **Real-World Impact** Build production-ready products that reach millions of users, from startups to billion-dollar companies.


 **Global Exposure** Work directly with international clients, giving you real-world experience with diverse teams, cultures, and markets.


 **Work with Cutting-Edge Tech** From Jamstack and serverless to AI and commerce ecosystems, you'll always be working with modern, high-demand tools.


 **Open & Transparent Culture** Weekly team calls, feedback loops, and async communication — we keep things clear and collaborative.

 **Career Growth & Ownership** You'll be given real responsibilities from Day 1, and a path to grow into leadership or specialization roles.

 **100% Remote & Flexible** Work from anywhere, on your schedule. We care about results, not time clocks.

 **A Team That Actually Cares** We're here to support each other, push boundaries, and have fun doing it.

 **Young, Driven & Fun Culture** No corporate BS. Just a passionate team building cool things and solving real problems.

 **Mental Wellness & Balance** We're serious about work-life balance and maintaining a healthy, respectful work environment.

About The company: <https://weframetech.com/> (Weframetech Solutions PVT Ltd)

LinkedIn: <https://www.linkedin.com/company/weframetechcom>

Best regards,

Vipul Uthaiah

CSO, Weframe Tech