🌿 Arvyax Full Stack Internship Assignment

🧘 Task Title:

Build a Secure Wellness Session Platform with Auth, Drafts, and Auto-Save

---

🎯 Objective

Design and implement a **full-stack application** that allows users to:

- Register and log in securely
- View wellness sessions (like yoga, meditation, etc.)
- Draft and publish their own custom sessions
- Auto-save drafts as they type

This simulates a real-world Arvyax use case and tests your ability to build secure, scalable, and interactive full-stack systems.

🛠️ Tech Stack Guidelines

You are free to use any libraries you're comfortable with. Our preferred stack is:

- **Frontend**: React.js (or Vue, Angular, Next.js)
- **Backend**: Node.js + Express (preferred), or Django/Flask
- **Database**: MongoDB (Atlas preferred)
- **Auth**: JWT (jsonwebtoken + bcrypt)
- **Bonus Deployment**: Render, Railway, Netlify, or Vercel

✅ Core Features to Build

🔐 Authentication

- `POST /register` – User registration (password must be hashed)
- `POST /login` – Login and return JWT token
- Store JWT securely in frontend (e.g., localStorage or cookies)
- Protect routes using JWT middleware

📘 Session Management API

Backend Routes:

Method	Endpoint	Description
GET	`/sessions`	Public wellness sessions
GET	`/my-sessions`	User’s own sessions (draft + published)
GET	`/my-sessions/:id`	View a single user session
POST	`/my-sessions/save-draft`	Save or update a draft session
POST	`/my-sessions/publish`	Publish a session

🧘 Frontend Pages

Page	Description
Login / Register	Auth forms + token handling
Dashboard	View published sessions
My Sessions	View and edit user’s drafts/published sessions
Session Editor	Form to create/edit session with:

- Title
- Tags (comma-separated)
- JSON file URL (text input)
- Save as Draft / Publish buttons
- ✅ Auto-save after 5s of inactivity (bonus)

📦 Database Schema Example

User

```
js
CopyEdit
{
  _id,
  email,
  password_hash,
  created_at
}
```

Session

```
js
CopyEdit
{
  _id,
  user_id: ObjectId,
  title: String,
  tags: [String],
  json_file_url: String,
  status: "draft" | "published",
  created_at,
  updated_at
}
```

✨ Bonus Features (Optional, Preferred)

- Auto-save feedback (toast or message)
- Fully working logout
- Responsive UI
- Deployed demo (frontend and backend)

📊 Evaluation Criteria

Area	What We Look For
Backend	Auth flow, JWT usage, secure data handling, clear API
Frontend	Form handling, protected routes, UI clarity
Database	Schema design, use of indexes or queries
Code Quality	Structure, reuse, naming, comments
System Thinking	Clean separation of concerns
Bonus Points	Debounce logic, deployment, visual polish

📁 Deliverables

- ✅ GitHub repo with:
    - `/backend` folder (Node + Mongo + Express)
    - `/frontend` folder (React/Vue/etc.)
- ✅ `README.md` with:
    - Setup instructions
    - Routes and API documentation

    - Optional: link to live demo or video walkthrough
- ✅ `.env.example` with required env vars

⏱️ Estimated Effort

3–5 days depending on experience. Focus on quality and functionality — not perfection.