🔧 Project Overview: Portfolio Website with Admin Dashboard & Backend API

This project is a **developer-focused portfolio website** with a secure, scalable backend architecture and a React-based admin interface. It's designed to showcase personal projects while offering full CRUD functionality for managing content through authenticated routes.

The system includes both a **public frontend** for visitors and a **protected admin backend** for managing project entries.

🧱 Tech Stack

🔹 Frontend:

- Static **HTML/CSS** for homepage and "About Me"
- **React SPA** for:
  - `/my-projects`: users can browse and filter projects
  - `/admin`: secure dashboard to add, edit, and delete projects
- Hosting planned via **AWS S3 + CloudFront**

🔹 Backend:

- **Node.js + Express** REST API
- Deployed to **AWS Lambda** behind **API Gateway**
- Routes include:
  - `GET /projects`: fetch all projects
  - `GET /projects?languages=...`: filter by tech stack

    - `POST /admin/add-project`: add a new project (admin only)
    - `PUT /admin/edit-project/:id`: edit an existing project
    - `DELETE /admin/delete-project/:id`: remove a project

🔹 **Database:**

- **MySQL** hosted on **AWS RDS**
- Three main tables:
    - `projects` (project metadata)
    - `languages` (available tech stack options)
    - `tech_stack` (many-to-many relationship between projects and languages)

🔹 **Authentication:**

- **Firebase Authentication (Email/Password)** is used
- Token verification is handled via **Firebase Admin SDK**
- All `/admin/*` routes require valid ID tokens via `Authorization` headers

🔐 Admin Functionality

➕ `/admin/add-project`

- Accepts `name`, `description`, `link`, and a list of `languages`
- Validates input
- Inserts into `projects` and related `tech_stack` entries

✏️ `/admin/edit-project/:id`

- Allows updating any of the fields from the add route
- Fully replaces associated language links in `tech_stack`

🗑️ `/admin/delete-project/:id`

- Deletes a project by ID
- Cascades to clean up associated entries in `tech_stack`

All admin actions are protected and require a valid Firebase ID token to execute.

🧪 Testing Setup

- Firebase tokens are generated via a local `get-token.js` script
- Postman is used to test all protected endpoints with real tokens
- Body validation and error handling are tested manually

📚 In Progress / Upcoming Enhancements

- **React Admin Panel** to visually manage projects
- **Pagination + filtering** on public-facing project list
- **Validation middleware** using `express-validator` or `zod`
- **Rate limiting** via `express-rate-limit`

- **Logging** via Winston for production observability
- **Test suite** with Jest + Supertest
- **CI/CD pipeline** with GitHub Actions for test-on-push

🤝 Summary

This project simulates what a junior or entry-level backend engineer would build in a real SaaS environment: secure route handling, database joins, cloud deployment, and token-based authentication — all with a focus on modularity and real-world usability.