ProGYM Project Documentation

# 📁 Project Folder Structure

ProGYM/  
├── docker-compose.yml  
├── progym-frontend/  
│ ├── Dockerfile  
│ ├── package.json  
│ ├── public/  
│ ├── src/  
│ │ ├── App.js // Main app with routes and layout  
│ │ ├── index.js // Entry point that wraps app in BrowserRouter and UserContext  
│ │ ├── context/  
│ │ │ └── UserContext.js // Global context for login state  
│ │ ├── components/  
│ │ │ ├── Navbar.js // Dynamic navbar based on user role  
│ │ │ └── Footer.js // Basic footer  
│ │ ├── styles/  
│ │ │ ├── Home.js // Inline JS styles for Home page  
│ │ │ └── Register.js // Styles for Register page  
│ │ ├── pages/  
│ │ │ ├── Home.jsx // Dynamic homepage with role-based sections  
│ │ │ ├── Login.jsx // Login form, sets context + localStorage  
│ │ │ ├── Register.jsx // Registration form, also logs in user  
│ │ │ ├── Profile.jsx // Placeholder user profile page  
│ │ │ ├── Upgrade.jsx // Premium upgrade trigger  
│ │ │ └── PremiumPage.jsx // Page only premium users can access  
├── progym-backend/  
│ ├── Dockerfile  
│ ├── package.json  
│ ├── server.js // Main Express server entry  
│ ├── config/  
│ │ └── db.js // MySQL database connection logic  
│ ├── routes/  
│ │ └── auth.js // Defines /register, /login, /upgrade routes  
│ └── controllers/  
│ └── authController.js // Register/login/upgrade logic with MySQL  
├── db/  
│ └── progymdb.sql // MySQL schema for users, workout\_sessions

# ⚙️ Tech Stack Used

Frontend: React.js  
State: React Context API  
Routing: React Router DOM  
HTTP Client: Axios  
Styling: CSS + JavaScript inline styles  
Backend: Node.js + Express  
Database: MySQL  
ORM/Driver: mysql2 (Promise-based)  
Auth Persistence: localStorage  
Dev Tools: Docker, Docker Compose

# 📦 Important Libraries

react-router-dom – For routing and route protection  
axios – HTTP request handler  
bcryptjs – Password hashing  
mysql2 – MySQL database integration with Promises  
cors, express.json() – Middleware for CORS + body parsing  
dotenv – Loads environment variables  
nodemon – Auto-reload dev server

# 🧠 Logic Overview

User Roles:  
- guest: not logged in  
- user: registered but not premium  
- premium: upgraded user  
  
Login/Register: validates fields, stores hashed passwords, returns user object (with role)  
Upgrade: updates backend role, updates UI  
Role-based UI: Home and Navbar update based on user.role

# 🚀 How to Start

1. Using Docker (recommended):  
docker-compose up --build  
  
2. Manual Setup:  
- Backend:  
 cd progym-backend  
 npm install  
 node server.js  
  
- Frontend:  
 cd progym-frontend  
 npm install  
 npm start

# ✅ Feature Checklist

Guest:  
- Register / Login  
- Read homepage  
- View testimonials  
- Browse features  
  
Logged-in User:  
- Track workouts  
- Meal plans  
- Calorie tracking  
- Upgrade to premium  
  
Premium User:  
- Progress charts  
- AI GymBot  
- Exclusive plans (bulking, cutting, etc.)  
- Premium dashboard

# ChatBot Logic

* User sends a message → API POST /api/gymbot
* Backend forwards it to **Google Gemini API**
* Gemini returns a reply → rendered in chat
* API key stored in .env, accessed via process.env.GOOGLE\_API\_KEY