* ***Installation & Setup Guide***

**System Requirements**:

* Node.js (v18+ recommended)
* npm (v9+ recommended)

**Installation Steps:**

1. **Clone Repository:** <https://github.com/rohansb27/Smart-Recipe-Hub.git>
2. **Backend Setup:**

cd backend

npm install

1. **Frontend Setup:**

cd frontend

npm install

1. **Environment Setup:** 
   * + Create .env files in both frontend and backend directories with relevant keys.
     + Backend example:

PORT=8000

MONGO\_URI=your\_mongo\_connection\_string

GCLOUD\_PROJECT\_ID=your\_project\_id

GCLOUD\_CLIENT\_EMAIL=your\_email

GCLOUD\_PRIVATE\_KEY=your\_private\_key

1. **Running the App:**
   * + Backend: npm start
     + Frontend: npm run dev

* ***Developer Guide***

**Folder Structure:**

* **/backend**
  + src/models: Contains Mongoose schemas for Users, Recipes, Comments, etc.
  + src/routes: Express route handlers organized by feature (e.g., recipeRoutes.js, userRoutes.js)
  + src/services: Custom services (e.g., Google Cloud Storage integration, Gemini AI interaction)
  + server.js: Entry point; sets up Express app, middleware, routes, DB connection
* **/frontend**
  + src/components: Reusable UI components (Navbar, RecipeCard, etc.)
  + src/pages: Page-level components (HomePage, UploadPage, RecipeDetailPage)
  + src/routes: Routing definitions and protected route wrappers
  + src/services: Axios-based service layer for API communication
  + layouts: Layout components used for consistent page structure
  + main.jsx: Entry point for the React app; mounts App component

**Codebase Walkthrough:**

* **Backend:**
  + server.js initializes Express, connects to MongoDB, sets up CORS, JSON parsing, and routes.
  + Routes are modularized (e.g., /add-recipe, /like-recipe, /comments) and handle logic for each feature.
  + Multer handles file uploads; images are stored in Google Cloud Storage using a utility in services/gcsUploader.js.
  + Gemini AI integration in a dedicated route under /api/gemini/recipe which sends a prompt and returns a recipe suggestion.
* **Frontend:**
  + App.jsx sets up routing and base layout.
  + Navbar.jsx manages navigation with links to core pages (e.g., Upload, Browse).
  + RecipeForm.jsx in pages handles form submission with file/image uploads.
  + Auth is managed with Clerk’s components (e.g., <SignIn />, <SignedIn />, etc.).
  + RecipeCard.jsx renders individual recipes using props passed from API responses.
  + services/api.js handles all API calls to the backend with axios.