**📁 Backend Folder Structure — Clean Architecture (Node.js / Express.js)**

This documentation outlines the purpose and responsibility of each directory and utility used in a modern backend project.

**📂 models/**

**Purpose**: Defines the **data structure (schema)** used in the application.

* Contains Mongoose schemas representing collections in MongoDB.
* Each schema is exported and used in controllers/services for CRUD operations.

🧠 Example: User.js, Post.js

// User.js

import mongoose from 'mongoose';

const userSchema = new mongoose.Schema({

name: String,

email: String,

password: String

});

export default mongoose.model('User', userSchema);

**📂 controllers/**

**Purpose**: Contains the **business logic** for each route.

* Acts as a bridge between **routes** and **models**.
* Handles user authentication, data manipulation, and responses.
* Keeps routes clean and organized.

🧠 Example: authController.js, userController.js

**📂 routes/**

**Purpose**: Defines all **API endpoints**.

* Connects HTTP requests to the corresponding controller functions.
* Organized by resource (e.g., /api/users, /api/auth).

🧠 Example: authRoutes.js, userRoutes.js

**📂 middleware/**

**Purpose**: Contains reusable logic that runs **before controllers**.

* Example use cases:
  + JWT authentication
  + Error handling
  + Validation

🧠 Example: authMiddleware.js, errorHandler.js

**📂 lib/**

**Purpose**: Contains **utility functions** and **helper logic**.

* For example, token generation, encryption, cloud upload utilities.

🧠 Example: generateToken.js, cloudinary.js

**🔐 JWT Tokens**

* Used for **user authentication** and **session management**.
* Tokens are created using user credentials (e.g., user.\_id) and a secret key.
* Typically stored in HTTP-only cookies or sent in request headers.

Generated using the jsonwebtoken library.

**🍪 Cookies**

* Used to **store JWT tokens** securely on the client.
* HTTP-only cookies prevent client-side JavaScript from accessing them (more secure).
* Helps maintain user sessions across page reloads.

**☁️ Cloudinary**

* Third-party **cloud storage service** for images, videos, and other files.
* Integrated via SDK or REST API using cloud\_name, api\_key, and api\_secret.
* Often used for uploading user profile pictures or message media.

**🛢️ Mongoose**

* ODM (Object Data Modeling) library for MongoDB.
* Provides schema-based models and methods to interact with the database easily.
* Used in the models/ directory.

**🌐 CORS (Cross-Origin Resource Sharing)**

* Allows the frontend (e.g., running on localhost:3000) to **communicate with the backend** (e.g., localhost:5000).
* Essential for API access control in browser-based apps.
* Handled using the cors package in Express.

!

Front-End Tools and Icons

Icons

React Icons

Lucide Icons

React hot toast

Zustand

Daisy UI