I'll assume your project structure is the usual **MERN alumni portal**.

🧠 OVERALL ARCHITECTURE (BIG PICTURE)

- **client/** → Frontend (React)
- **server/** → Backend (Node + Express)
- **MongoDB** → Database (models live in server)

Frontend talks to backend using **Axios + JWT auth**.

📁 CLIENT (FRONTEND – React)

client/

├── src/

📁 src/

This is where **everything the user sees** lives.

📁 assets/

👉 Static files

- logo.png
  → College / portal logo
  → Used in Navbar

Nothing dynamic. Just images/icons.

📁 components/

👉 Reusable UI blocks

Navbar.js

- Top navigation bar
- Shows:
  - Logo
  - Portal name

    - Logout button
- Handles:
    - Logout (localStorage.removeItem("token"))
    - Navigation

Used on **every protected page**.

ImageCarousel.js

- Homepage image slider
- Auto-scrolls images
- Shows:
    - Event banners
    - Announcements (visual only)
- Fetches images from backend

Pure UI component.

📁 **pages/**

👉 **Actual screens (routes)**

Each file = **one page**

Dashboard.js

- Main landing page **after login**
- Role-based UI:
    - Admin → approvals, carousel, events
    - Alumni → profile, directory, jobs
- Fetches carousel images
- Uses jwtDecode to identify role

This is your **control center**.

Login.js

- Login form
- Sends credentials to backend
- Receives JWT token
- Saves token in localStorage

Register.js

- Alumni registration form
- Sends data to backend
- Status usually = pending approval

AdminCreateEvent.js

- **Admin-only**
- Create alumni events
- Sends event data to backend
- Stored in MongoDB

PostJob.js

- **Alumni-only**
- Post job openings
- Stored in database
- Displayed to all alumni

Events.js

- Shows all events
- Data fetched from backend
- Read-only for alumni

Jobs.js

- Shows job postings
- Alumni can view
- Posting controlled by role

📁 **styles/**

👉 **Design system**

theme.js

- Central color palette
- Fonts
- Card styles
- Used everywhere

This prevents **hardcoding colors**.

📁 **utils/**

(if present)

api.js

- Axios instance
- Central place for:
    - Base URL
    - Authorization headers (JWT)

Cleaner than repeating axios config everywhere.

📄 **App.js**

👉 **Routing brain**

- Defines routes:
    - /dashboard
    - /events

    - /jobs
    - /admin/...
- Protects routes based on token
- Wraps pages with Router

Without this → app won't navigate.

📄 **index.js**

👉 **React entry point**

- Renders <App />
- Mounts React into HTML

You rarely touch this.

📁 **SERVER (BACKEND – Node + Express)**

server/

Handles:

- Auth
- Database
- Business logic
- Security

📁 **models/**

👉 **MongoDB schemas**

User.js

- Alumni/admin structure
- Fields:
    - name
    - email

- password
- role
- approvalStatus

Defines **how users are stored**.

Event.js

- Event data structure
- title, date, description, location

Job.js

- Job posting structure
- company, role, location, description

Carousel.js

- Image URL + title
- Used by homepage carousel

📁 **routes/**

👉 **API endpoints**

Each file = **URL group**

authRoutes.js

- /login
- /register
- JWT generation

---

adminRoutes.js

- Approve alumni
- Admin-only actions

---

eventRoutes.js

- Create event
- Get all events

---

jobRoutes.js

- Create job
- Get job listings

---

carouselRoutes.js

- Upload image
- Fetch carousel images

📁 **controllers/**

👉 **Logic layer**

Routes call controllers.

Example:

- Route: /api/jobs
- Controller:
    - Validates request
    - Saves to DB
    - Sends response

Keeps routes clean.

📁 **middleware/**

👉 **Security + checks**

authMiddleware.js

- Verifies JWT token

- Blocks unauthorized access

---

roleMiddleware.js

- Checks:
    - admin vs alumni
- Prevents role abuse

📄 **server.js**

👉 **Backend entry point**

- Starts Express server
- Connects MongoDB
- Registers routes
- Applies middleware

Equivalent of index.js for backend.

📄 **.env**

👉 **Secrets**

- MongoDB URI
- JWT secret
- Port

Never commit this to GitHub.

🔐 **AUTH FLOW (IMPORTANT)**

1. Login → backend issues JWT
2. JWT stored in localStorage
3. Axios sends JWT in headers
4. Middleware validates JWT

5. Role-based access granted