**Authentication and Authorization Strategy**  
  
**Authentication Strategy (JWT (JSON Web Token)– Laravel Sanctum)**

* Will verify who the user is.
* Will ensure that only the valid users can access the system.

**JWT (JSON Web Token)**

* **Stateless** → No need to store sessions on the server.
* **Secure** → The token is signed and tamper-proof.
* **Scalable** → Ideal for **APIs, mobile apps, and microservices**.
* **Efficient** → No need to query the database on every request.

**Authentication Flow Using JWT**

1. **User logs in** with email & password.
2. **Server verifies credentials** and generates a **JWT token**.
3. **Token is sent to the client** (stored in local storage or HTTP headers).
4. **Client includes token in every API request** (Authorization: Bearer {token}).
5. **Server verifies token** and allows or denies access.

**JWT Structure (Contains User Role for Authorization)**

**{**

"userCode": 23-A-12345,

"role": "Student",

iat": 1675124000,

"exp": 1675127600

**}**

**Authorization Strategy (Using RBAC – Role Based Access Control)**

* **RBAC** strategy can **control what users can do** after they log in.
* Using this will ensure only **authorized users** can perform certain actions.

**RBAC (Role-Based Access Control)**

* **Granular Permissions** → Restrict actions based on user roles.
* **Secure Access Control** → Prevents unauthorized actions.
* **Scalable** → Easily add new roles and permissions.

**RBAC Structure**

|  |  |
| --- | --- |
| **Role** | **Permissions** |
| Student | Take exams, view subjects |
| Faculty | Create questions, request approval |
| Program Chair | Approve questions, print exam papers |
| Dean/Associate Dean | Full access (approve/reject questions, assign faculty) |