Authentication System (Express + Cookies + JWT)

- **express** – server
- **cookie-parser** – read cookies from request
- **jsonwebtoken (JWT)** – identity proof (more on this later)
- **bcryptjs** – password security (more on this later)
- **dotenv** – hide secrets like `MONGO_URI` and `JWT_SECRET`

Core Concepts

Authentication

User kon hai?

Example:
"Is this request coming from *Ankur* or someone else?"

Authorization

User kya kya kar sakta hai?

Example:
"Can this user delete posts or only view them?"

Validation

Data ka format sahi hai ya nahi?

Example:

- Email looks like email?
- Password length >= 6?
- Role is `user` or `admin`?

Verification

Data sahi hai ya nahi?

Example:

- Email exists in DB?
- Password matches stored password?
- Token is valid or expired?

Why we need to know

"Request kis user ne ki thi/hai?"

If you don't know **who made the request**, your backend is blind.

Bank Example

- `/api/bank/withdraw`
- Question backend must answer:
    - **Which account?**
    - **Which user?**
    - **Is balance enough?**

If backend doesn't know the user →
Anyone can withdraw from anyone's account.

Instagram Example

- `/api/posts/create`
- Backend must know:
    - Who is posting?
    - Whose profile should show this post?

If user identity is missing →
Post will have no owner → system breaks.

Conclusion:
Every serious backend feature depends on
“request kis user ne ki thi”

Authentication Flow (Big Picture)

```
User → Login → Get Token → Store in Cookie
Every Request → Cookie → Token → User Identity
```

APIs We Will Build

API	Purpose
`/api/auth/register`	New user create
`/api/auth/login`	User login
`/api/auth/get-me`	Logged-in user info

Step-by-Step Implementation (Sequence)

Step 1: User Register (`/api/auth/register`)

Role of Register API

- Create new user
- Save user in database
- Store password **securely**

What happens inside

1. Validate data (email, password)
2. Check user already exists or not
3. Hash password using `bcryptjs`
4. Save user in DB
5. Return success response

Register does NOT log user in
It only **creates identity**

Step 2: User Login (`/api/auth/login`)

Role of Login API

- Verify user
- Give proof of identity (JWT)

What happens inside

1. Validate input
2. Find user by email
3. Compare password using `bcryptjs`
4. Create JWT (userId, role)
5. Send JWT in **HTTP-only cookie**

Login = **authentication happens here**

Step 3: Store JWT in Cookie

Why cookie?

- Automatically sent with every request
- Safer than localStorage (for beginners)

Cookie contains:

```
token = JWT
```

Step 4: Auth Middleware (Very Important)

Middleware job:

1. Read cookie using `cookie-parser`
2. Get token
3. Verify token using `jsonwebtoken`
4. Extract userId
5. Attach user to `req.user`

If token missing or invalid → reject request

Step 5: Get Logged-in User (`/api/auth/get-me`)

Purpose

- Check who is currently logged in

Flow

1. Request comes with cookie
2. Middleware runs
3. `req.user` is already available
4. Return user data

No email/password needed here
Identity already proved by token

How Each Tool Is Used (Simple)

Tool	Why
express	Create APIs
cookie-parser	Read token from cookie
jsonwebtoken	Create + verify user identity
bcryptjs	Protect password
dotenv	Hide JWT secret

One-Line Summary (Remember This)

- **Register** → create user
- **Login** → prove user
- **JWT** → identity proof
- **Cookie** → carry identity
- **Middleware** → guard routes