User Module Explanation (Node.js + Express + MongoDB)

1. user.controller.js (Controller Layer)

Handles incoming HTTP requests for user actions (register, login, etc.).
- Validates input using express-validator.
- Calls userService to create a user in DB.
- Hashes the password before saving.
- Generates JWT token after user creation.
- Sends JSON response with token and user details.

2. user.model.js (Database Schema / Mongoose Model)

Defines MongoDB schema for users.
- Fields: fullname (firstname, lastname), email, password, socketId.
- Adds helper methods:
• generateAuthToken() – creates JWT.
• comparePassword() – compares entered vs hashed password.
• hashpassword() – hashes password using bcrypt.
- This is the data model directly interacting with MongoDB.

3. user.routes.js (Routing Layer)

Defines API endpoints for user-related routes using express.Router().
- POST /register → calls userController.registerUser with validations.
- POST /login → calls userController.loginUser (to be implemented).
- Connects HTTP routes to controllers.

4. user.service.js (Business Logic Layer)

Acts as intermediate between controller and model.
- Handles business logic for user creation.
- createUser function checks required fields and inserts user into MongoDB.
- Keeps controllers slim and avoids direct DB handling.

5. Overall Flow (User Registration)

1. Client sends POST /register with user data.
2. Validations checked in routes.
3. Controller hashes password, calls service.
4. Service calls model to create user in DB.
5. Model saves to MongoDB and provides helpers.
6. Controller generates JWT token.
7. Response returned with token and user info.

■ Structure Recap

- Routes → define endpoints + validation
- Controller → handles request/response
- Service → business logic (DB operations)
- Model → schema & DB helpers