Task: Build a Secure User Authentication System with JWT and Role-Based Access

Objective:

Create a complete backend authentication system using Node.js and Express.js that supports:

- User registration and login
- Authentication using JWT access & refresh tokens
- Authorization using role-based access control
- Input validation
- Rate limiting
- Password hashing
- Use of environment variables

Functional Requirements:

User Registration:

- Endpoint: POST /register
- Required fields: name, email, password
- Validate all inputs using express-validator
- Hash the password using bcrypt before storing
- Return a success message after creating the user

User Login:

- Endpoint: POST /login
- Validate inputs
- If successful:
- Return an access token in the JSON response body
- Return a refresh token as an httpOnly cookie
- Tokens must be signed using JWT

Refresh Token:

- Endpoint: POST /refresh
- Validate the refresh token from the cookie
- Return a new access token if valid

Protected Route:

- Endpoint: GET /profile
- Require a valid access token
- Return user information from the token

Admin-only Route:

- Endpoint: GET /admin

- Only accessible to users with the role "admin"
- Return a message like "Welcome, admin"

Technical Requirements

- Use dotenv to manage environment variables:
    - PORT, JWT_SECRET, JWT_REFRESH_SECRET, MONGO_URI, etc.
- Use bcrypt for password hashing
- Use cors to allow cross-origin requests
- Apply rate limiting to /login and /register routes using express-rate-limit (e.g., max 5 requests per minute)
- Implement role-based access control:
    - Users have a role field: "user" by default, "admin" manually set in DB
    - Use a simple MongoDB model via Mongoose:
        - name, email, password, role, etc.

Useful Modules:

npm install express mongoose dotenv bcryptjs jsonwebtoken express-validator cookie-parser express-rate-limit cors

Bonus (Optional for Extra Credit)

- Token rotation: issue a new refresh token and invalidate the old one
- Store refresh tokens in memory or in MongoDB
- Implement logout endpoint /logout to clear cookies or invalidate refresh token

API Summary

Method	Route	Description	Protected?	Role Required
POST	/register	Register a new user	False	None
POST	/login	Login user	False	None
POST	/refresh	Get new access token via refresh	True (cookie)	None
GET	/profile	Get user profile	True	User or Admin
GET	/admin	Admin-only route	True	Admin only

Testing

- Use Postman or Thunder Client to test all endpoints
- Access token should be sent as Authorization: Bearer <token>
- Refresh token should be sent as an httpOnly cookie
- Test both successful and failing cases