

User Authentication Backend

This project is a basic backend system for user signup/login using:

- OTP verification
- JWT-based authentication
- Refresh tokens
- HTTP-only cookies
- Middleware-protected routes

Features

- Signup with name, email, mobile, password
- OTP generation + expiry (mocked via console)
- OTP verification before login
- Login with email/mobile + password
- Refresh token endpoint
- HTTP-only cookie for access token
- Middleware-protected `/protected` route
- In-memory storage for users and OTPs (DB optional)

Tech Stack

- Node.js
- Express.js
- dotenv
- cookie-parser
- jsonwebtoken
- nodemon (for development)

Folder Structure

```
user-auth-backend/  
├── controllers/  
│   └── authController.js  
├── middleware/  
│   └── authMiddleware.js  
├── routes/  
│   └── authRoutes.js  
├── utils/  
│   └── otp.js  
├── memoryStore.js  
├── index.js  
├── .env  
└── package.json
```

Setup Instructions

1. Clone the repository:

```
git clone <repository-url>
cd user-auth-backend
```

2. Install dependencies:
npm install

3. Create a `.env` file:

```
PORT=3000
JWT_SECRET=your-secret-key
REFRESH_SECRET=your-refresh-secret-key
```

4. Start the server:
Server runs at: `http://localhost:3000`

API Endpoints and curl Commands

1. Signup

```
curl -X POST http://localhost:3000/signup
-H "Content-Type: application/json"
-d '{"name":"xyz","email":"xyz@gmail.com","mobile":"1234567890","password":"1234"}'
```

Response: OTP will be shown in the server console.

2. Verify OTP

```
curl -X POST http://localhost:3000/verify-otp
-H "Content-Type: application/json"
-d '{"email":"sanjana@gmail.com","otp":"123456"}'
```

Replace `123456` with the OTP from console output.

3. Login

```
curl -X POST http://localhost:3000/login
-H "Content-Type: application/json"
-d '{"email":"xyz@gmail.com","password":"1234"}'
```

-c cookie.txt

This saves the access token in `cookie.txt`.
Response includes a refresh token.

4. Access Protected Route

```
curl -X GET http://localhost:3000/protected  
-b cookie.txt
```

Requires login cookie.

5. Refresh Token

```
curl -X POST http://localhost:3000/refresh-token  
-H "Content-Type: application/json"  
-d '{"refreshToken":"<your-refresh-token>"}'  
-c cookie.txt
```

Returns a refreshed access token in the cookie.

OTP

- A 6-digit OTP is generated at signup.
- Stored in memory (`memoryStore`) with a 5-minute expiry.
- Sent via console log (mocked).

Cookies

- Login returns a short-lived JWT access token.
- The token is saved as an HTTP-only cookie.
- Cookies are secure from JavaScript (not accessible by frontend).

Middleware

- Custom middleware (`authMiddleware.js`) verifies JWT tokens.
- Token is extracted from the `req.cookies.token` field.
- If valid, the user is allowed to access protected routes.