□ WhatsApp Auth API — Node.js + Express + MongoDB

This documentation provides an overview of the WhatsApp-based authentication system for user **registration**, **login**, and **OTP verification, built using Node.js**, **Express.js**, **MongoDB Atlas**, and **JWT**.

□ Deployment Info (AWS EC2 + Elastic IP)

The API is deployed on an *Amazon EC2 instance* with a permanently assigned *Elastic IP* to ensure a fixed public address for the backend server.

□ What is an EC2 Instance?

An EC2 (Elastic Compute Cloud) instance is a virtual server in Amazon's cloud infrastructure where you can run applications. It behaves like a physical server and supports custom configurations.

□ What is an Elastic IP?

An *Elastic IP address* is a static, public IPv4 address provided by AWS. Unlike the dynamic public IP addresses assigned to EC2 instances by default, Elastic IPs do not change across instance restarts, ensuring a consistent endpoint for accessing services.

□ Why Elastic IP?

- Ensures a *permanent endpoint* for API usage
- Eliminates the need to update DNS or documentation after restarts
- Ideal for *production deployments*

□ Current Elastic IP:

```
http://15.206.226.196
```

This IP address hosts our API services and can be used for all endpoint requests. - The Elastic IP is the same as the Public IP, but it doesn't change even if the server restarts. - That's why we use it — to keep one fixed IP for our server.

□ Registration Flow

1. □ Register User (Send OTP)

Endpoint:

POST

```
http://15.206.226.196/api/users/register
```

This endpoint begins the registration process by sending a *One-Time Password (OTP)* to the user's provided mobile number using the 2Factor API.

Request Body:

```
{
  "phoneNumber": "9108547263",
  "name": "Balaji S"
}
```

Response:

```
{
  "message": "OTP sent successfully for registration"
}
```

Status: 200 OK

---

2. □ Verify OTP (Complete Registration)

Endpoint:

POST

```
http://15.206.226.196/api/otp/verify
```

Use this endpoint to verify the OTP sent during registration. The mode should be "register" to confirm the user is registering, not logging in.

Request Body:

```
{
  "phoneNumber": "9108547263",
  "otp": "88402",
  "mode": "register"
}
```

> □ *Note:*
> The mode value determines the context of the OTP verification:
>
> - "register": Registers a new user upon successful OTP verification.
> - "login": Logs in an existing user after OTP verification.

Response:

```
{
  "message": "User registered successfully"
}
```

Status: 201 Created

□ Login Flow

3. □ Login User (Send OTP)

Endpoint:

POST

```
http://15.206.226.196/api/users/login
```

This endpoint sends an OTP to an existing user's phone number to start the login process.

Request Body:

```
{
  "phoneNumber": "9108547263"
}
```

Response:

```
{
  "message": "OTP sent successfully for login"
}
```

Status: 200 OK

---

4. □ Verify OTP (Login)

Endpoint:

POST

```
http://15.206.226.196/api/otp/verify
```

After receiving the OTP via the login endpoint, verify it here. The mode must be "login" to trigger the login flow and return a JWT token.

Request Body:

```
{
  "phoneNumber": "9108547263",
  "otp": "73455",
  "mode": "login"
}
```

Response:

```
{
  "message": "Login successful",
  "user": {
    "_id": "67f7739e1c930cc6820e2983",
    "phoneNumber": "9108547263",
    "name": "Balaji S",
    "isVerified": true,
    "createdAt": "2025-04-10T07:30:38.140Z",
    "updatedAt": "2025-04-10T07:36:05.601Z",
    "__v": 0
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Status: 200 OK

> *JWT Token:*
> A secure JSON Web Token is issued upon successful login. This token can be used for authenticated API requests in future sessions.

Get All Users

Endpoint:

GET

```
http://15.206.226.196/api/users/get-all-users
```

This endpoint retrieves a list of all registered users in the system.

Response:

```
{
  "users": [
    {
      "_id": "67f740163b9a5b0bd0d18a36",
      "phoneNumber": "9876543210",
      "name": "Rahul Kumar",
      "isVerified": true,
      "createdAt": "2025-04-10T03:50:46.795Z",
      "updatedAt": "2025-04-10T03:50:46.795Z"
    },
    {
      "_id": "67f74787b53912256585a87d",
      "phoneNumber": "9988776655",
      "name": "Asha Mehta",
      "isVerified": true,
      "createdAt": "2025-04-10T04:22:31.669Z",
      "updatedAt": "2025-04-10T04:23:20.674Z"
    },
    {
      "_id": "67f7739e1c930cc6820e2983",
      "phoneNumber": "9108547263",
      "name": "Balaji S",
      "isVerified": true,
      "createdAt": "2025-04-10T07:30:38.140Z",
      "updatedAt": "2025-04-10T07:36:05.601Z"
    }
  ]
}
```

Status: 200 OK

□ Background Process Management

- To ensure the backend runs 24/7 on the EC2 instance without stopping even if the terminal session ends or the server reboots, we used a tool called P2m (Package Production Manager).
- What is P2m? P2m is a lightweight process manager that allows Node.js applications to:
- Run in the background
- Auto-restart on crashes
- Stay active across SSH disconnections and reboots
- This makes our EC2-hosted authentication API stable, resilient, and production-ready.

□ Notes

- All OTPs are powered by 2Factor SMS API integration.
- OTPs are required for both *registration* and *login*, differentiated using the mode field (register or login).
- JWT tokens are returned on successful login to support secure, session-less authentication for your app or frontend.
- Responses are clean, developer-friendly, and ready for integration with mobile or web clients.