Fake WhatsApp Chatbox – MongoDB CRUD Project

1. Project Overview

This project is a **Fake WhatsApp Chatbox** built to understand and practice **CRUD (Create, Read, Update, Delete)** operations using **MongoDB** along with **Node.js, Express.js, and EJS**.

The application allows users to:

- View chat messages in a WhatsApp-like interface
- Add new chat messages
- Edit existing messages
- Delete chat messages

This project is mainly developed for **learning backend fundamentals**, **MongoDB integration**, and **server-side rendering using EJS**.

---

2. Tech Stack Used

- **Node.js** – JavaScript runtime environment
- **Express.js** – Web framework for Node.js
- **MongoDB** – NoSQL database
- **Mongoose** – ODM for MongoDB
- **EJS (Embedded JavaScript)** – Template engine for dynamic HTML
- **CSS** – Styling the chat UI

3. Folder Structure Explanation

Day-04/mongodb/mongo3

```
├── models/
│   └── chats.js
├── node_modules/
├── public/
│   └── stylesheet.css
├── views/
│   ├── index.ejs
│   ├── new.ejs
│   └── edit.ejs
├── ExpressError.js
├── index.js
├── init.js
├── package.json
└── package-lock.json
```

Explanation of Each Folder/File

`models/`

- Contains **Mongoose schema and model**
- `chats.js` defines the structure of a chat document (from, to, message, createdAt)

`views/`

- Contains all **EJS template files**
- Responsible for rendering dynamic HTML pages

Files inside views:

- `index.ejs` → Displays all chat messages
- `new.ejs` → Form to create a new chat
- `edit.ejs` → Form to edit an existing chat

`public/`

- Contains static assets
- `stylesheet.css` is used to style the fake WhatsApp chat UI

`index.js`

- Main entry point of the application
- Sets up Express server, routes, middleware, and database connection

`init.js`

- Used to initialize the database with sample chat data
- Helpful for testing and demo purposes

`ExpressError.js`

- Custom error handling class
- Helps manage application-level errors cleanly

`package.json`

- Contains project metadata and dependencies

4. Database Schema (MongoDB)

The chat data is stored in MongoDB using Mongoose.

Chat Schema Fields (Example)

- `from` → Sender name
- `to` → Receiver name
- `message` → Chat message text
- `createdAt` → Timestamp

Each document represents a single chat message.

5. CRUD Operations Explained

Create (C)

- Users can add a new chat message
- Form rendered using `new.ejs`
- Data is saved to MongoDB using Mongoose

Read (R)

- All chats are fetched from MongoDB
- Displayed on `index.ejs` in a chatbox format

Update (U)

- Existing messages can be edited
- Edit form rendered using `edit.ejs`
- Changes are updated in MongoDB

Delete (D)

- Chat messages can be deleted
- Corresponding document removed from MongoDB

6. EJS Rendering Flow

1. User requests a route
2. Express fetches data from MongoDB
3. Data is passed to an EJS file
4. EJS dynamically renders HTML
5. Browser displays updated UI

7. Styling (Fake WhatsApp UI)

- CSS is written inside `public/stylesheet.css`
- Chat layout mimics WhatsApp-style message bubbles
- Clean and minimal UI for better readability

8. How to Run the Project

Step 1: Install Dependencies

npm install

Step 2: Start MongoDB

Make sure MongoDB is running locally.

Step 3: Initialize Database (Optional)

node init.js

Step 4: Run the Server

node index.js

Step 5: Open Browser

http://localhost:3000

9. Learning Outcomes

- Understood CRUD operations using MongoDB
- Learned Mongoose schema and models
- Practiced EJS templating
- Implemented Express routing
- Improved backend project structure

10. Future Enhancements

- User authentication
- Real-time chat using Socket.io
- Message timestamps UI
- Responsive design
- Database validation

11. Conclusion

This Fake WhatsApp Chatbox project is a beginner-friendly full-stack application that demonstrates how **MongoDB CRUD operations** work with **Express and EJS**. It serves as a strong foundation for learning backend development and MVC architecture.