Step 1: Importing Dependencies

First, the necessary modules and libraries are imported:

javascript Copy code

```
import { Router } from "express"; import dotnet from "dotenv"; import user from
"../helpers/user.js"; import jwt from "jsonwebtoken"; import chat from
"../helpers/chat.js"; import OpenAI from "openai"; import { db } from
"../db/connection.js"; import collections from "../db/collections.js"; import multer
from "multer"; import fs from "fs"; import { ObjectId } from "mongodb";
dotnet.config();
```

These imports include Express.js for routing, dotenv for environment variables, jwt for JSON Web Token handling, and more.

Step 2: Setting Up Middleware and Instances

The router instance and middleware are set up:

javascript Copy code

```
let router = Router(); const upload = multer({ dest: "uploads/" }); const CheckUser =
async (req, res, next) => { // Middleware for checking the user token };
```

`CheckUser` middleware verifies the user's JWT token to ensure they are authenticated.

Step 3: Defining the OpenAI Client

javascript Copy code

```
const client = new OpenAI({ apiKey: "sk-
EYunmiF6ERSCWcl4Fgu7T3BlbkFJbrUzlWaAmd9XBsacMctG", });
```

An OpenAI client is instantiated with an API key.

Step 4: Defining Routes

1. Welcome Route

javascript Copy code

```
router.get("/", (req, res) => { res.send("Welcome to chatGPT api v1"); });
```

- **Method:** GET
- **Path:** `/`
- **Description:** Responds with a welcome message.

2. Upload File Information Route

javascript Copy code

```
router.get("/upload", CheckUser, async (req, res) => { // Route implementation });
```

- **Method:** GET
- **Path:** `/upload`
- **Middleware:** `CheckUser`
- **Description:** Retrieves information about a previously uploaded file based on `chatId`.

3. Upload File Route

javascript Copy code

```
router.post("/upload", upload.single("file"), CheckUser, async (req, res) => { // Route implementation });
```

- **Method:** POST
- **Path:** `/upload`
- **Middleware:** `upload.single("file")`, `CheckUser`
- **Description:** Uploads a file to OpenAI and stores file metadata in MongoDB.

4. Chat Prompt Route

javascript Copy code

```
router.post("/", CheckUser, async (req, res) => { // Route implementation });
```

- **Method:** POST
- **Path:** `/`
- **Middleware:** `CheckUser`
- **Description:** Sends a chat prompt to OpenAI and saves the response to the database.

5. Update Chat Route

javascript Copy code

```
router.put("/", CheckUser, async (req, res) => { // Route implementation });
```

- **Method:** PUT
- **Path:** `/`
- **Middleware:** `CheckUser`
- **Description:** Updates an existing chat session with a new prompt and retrieves responses from OpenAI.

6. Retrieve Saved Chats Route

javascript Copy code

```
router.get("/saved", CheckUser, async (req, res) => { // Route implementation });
```

- **Method:** GET
- **Path:** `/saved`
- **Middleware:** `CheckUser`
- **Description:** Retrieves saved chat sessions based on userId and optionally chatId.

7. Retrieve Chat History Route

javascript Copy code

```
router.get("/history", CheckUser, async (req, res) => { // Route implementation });
```

- **Method:** GET
- **Path:** `/history`
- **Middleware:** `CheckUser`
- **Description:** Retrieves the chat history for the authenticated user.

8. Delete All Chats Route

javascript Copy code

```
router.delete("/all", CheckUser, async (req, res) => { // Route implementation });
```

- **Method:** DELETE
- **Path:** `/all`
- **Middleware:** `CheckUser`
- **Description:** Deletes all chat sessions for the authenticated user.

9. Get Attached Files Route

javascript Copy code

```
router.post("/getfile", async (req, res) => { // Route implementation });
```

- **Method:** POST
- **Path:** `/getfile`
- **Description:** Retrieves files attached to a chat session.

10. Delete File Route

javascript Copy code

```
router.post("/deletefile", CheckUser, async (req, res) => { // Route implementation
});
```

- **Method:** POST
- **Path:** `/deletefile`
- **Middleware:** `CheckUser`
- **Description:** Deletes a specific file from a chat session.

Summary

- **GET** `/`: Welcome route.
- **GET** `/upload`: Retrieves file info based on `chatId`.
- **POST** `/upload`: Uploads a file to OpenAI and stores metadata.
- **POST** `/`: Sends a chat prompt to OpenAI and stores the response.
- **PUT** `/`: Updates an existing chat session with a new prompt.
- **GET** `/saved`: Retrieves saved chat sessions.
- **GET** `/history`: Retrieves chat history.
- **DELETE** `/all`: Deletes all chat sessions.
- **POST** `/getfile`: Retrieves attached files.
- **POST** `/deletefile`: Deletes a specific file from a chat session.

Each route is designed to handle specific aspects of the application's functionality, such as file uploads, chat interactions, and user authentication.