**House API Documentation**

**Overview**

The House API provides functionality to manage house details, including uploading house information, retrieving lists of houses, getting specific house details by ID, and updating house details such as status, owner phone number, and price.

**Github Repository:**

<https://github.com/Eshwar-Sai-2001/House_api/tree/main>

**Tech Stack Used**

* JavaScript
* Express.js
* Node.js
* MySQL
* JEST (Test Framework)
* GIT
* Docker
* Swagger

**Local Setup**

Clone the repository using the following command:

**bash**

**git clone https://github.com/Eshwar-Sai-2001/House\_api.git**

Navigate to the project directory:

**bash**

**cd House\_api**

Install dependencies:

**bash**

**npm install**

Run the application locally:

**bash**

**npm run start**

The application will start, and you can access it locally at http://localhost:3000.

**Running Commands**

To start the application, use the following command:

**bash**

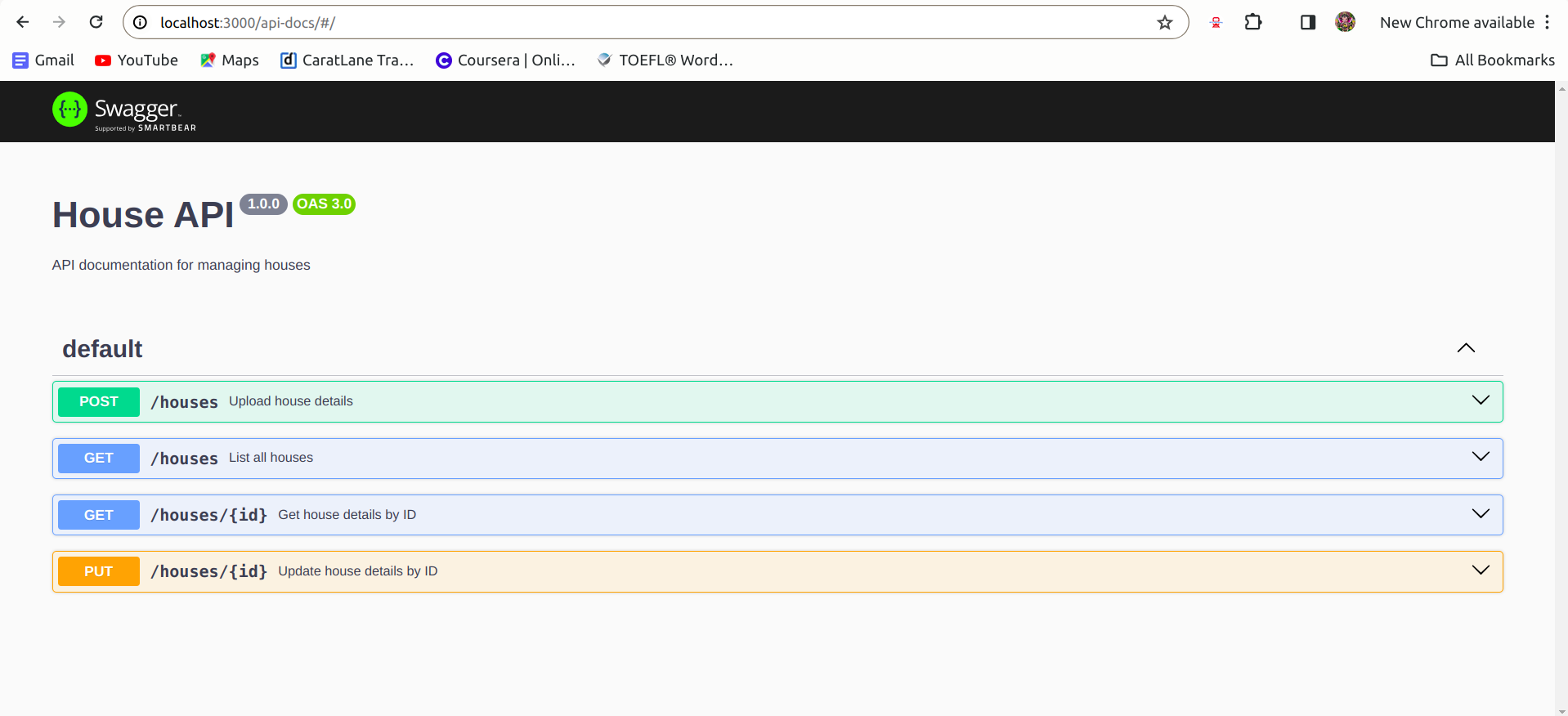
**npm run start**

This will launch the server on port 3000 and establish a connection to the MySQL database.

**Note: Ensure that you have Node.js and npm installed on your machine before running the above commands.**

Application will be running on localhost 3000, open it any browser where you will be able to see 4 different API’s

1. Upload house details
2. Get List of houses
3. Get house details by ID ( particular house)
4. Update the details ( status, owner\_phone\_number, price)



The House API has been Dockerized, allowing it to run in a Docker container. This provides an isolated environment and simplifies deployment. Follow the steps below to set up and run the Dockerized House API.

**Docker Commands**

**Build Docker Image**

**bash**

**docker build -t house-api .**

**Description**: Builds a Docker image named house-api based on the Dockerfile in the project directory.

**Run Docker Container**

**bash**

**docker run -p 4000:4000 -d house-api**

Description: Runs a Docker container named house-api in detached mode, mapping port 4000 of the host to port 4000 of the container.

**Check Running Containers**

**bash**

**docker ps**

Description: Lists all running Docker containers. Ensure that the house-api container is running.

Database Setup

Before running the container, follow these additional steps to set up the database for the House API.

**Create Database Container**

**bash**

**docker run -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=house\_api\_db -p 3306:3306 -d mysql:latest**

Description: Runs a MySQL container with the specified environment variables for root password and database name.

**Access MySQL Container**

**bash**

**docker exec -it <mysql-container-id> /bin/bash**

Description: Enters the MySQL container to execute commands.

**Access MySQL CLI**

**bash**

**mysql -u root -p**

Description: Logs into MySQL CLI. Enter the root password when prompted.

Now, you have both the House API and the MySQL database running in Docker containers. You can access the API at http://localhost:4000 and perform operations. Ensure that the database container is running and the house-api container is connected to it.

**API Endpoints**

1. **Upload House Details**

**Endpoint**: POST /houses

**Description**: Upload details for a new house.

**Request Body**:

{

"owner\_name": "Eshwar",

"owner\_phone\_number": "+91 9090909090",

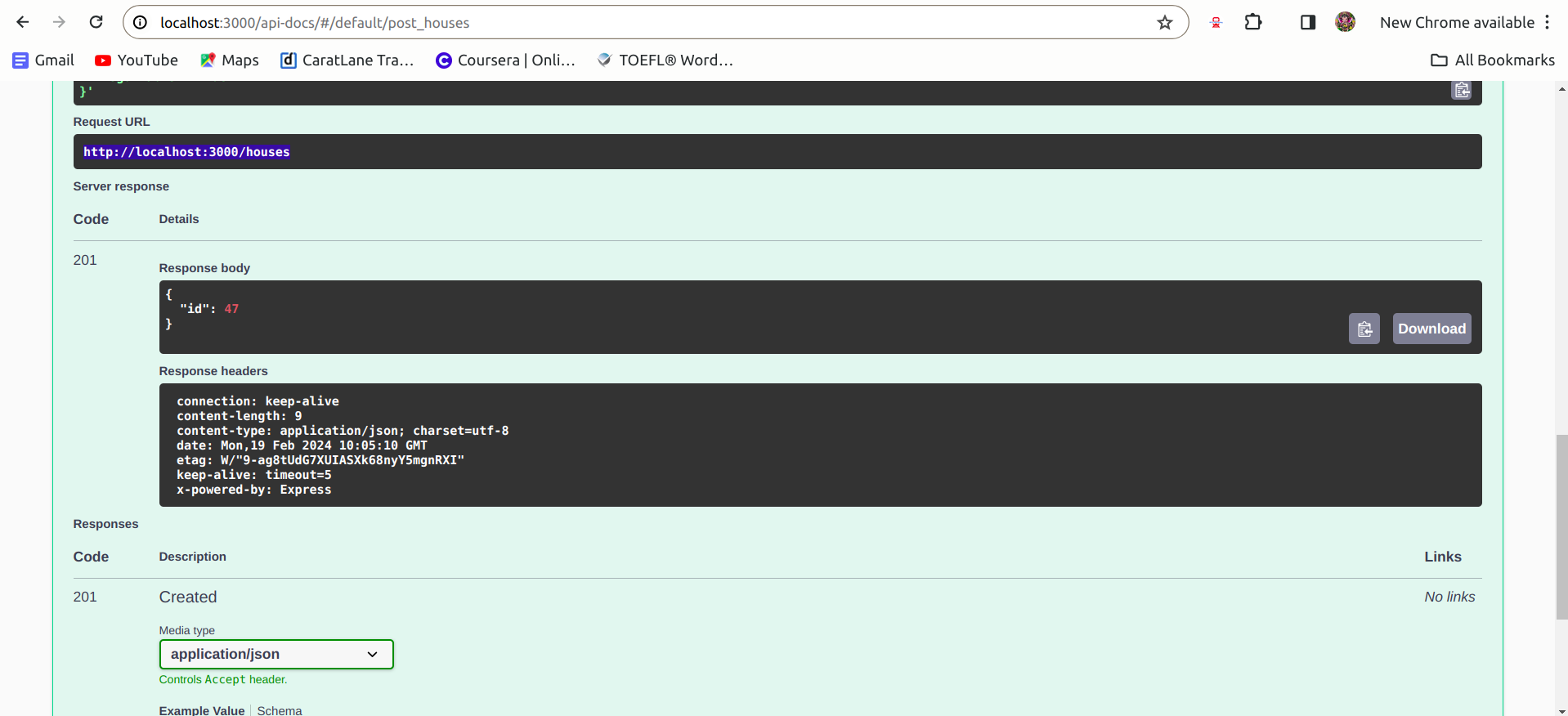
"area": "500 sq.yards",

"sale\_price": 30000000,

"negotiable": true

}

**Response**:

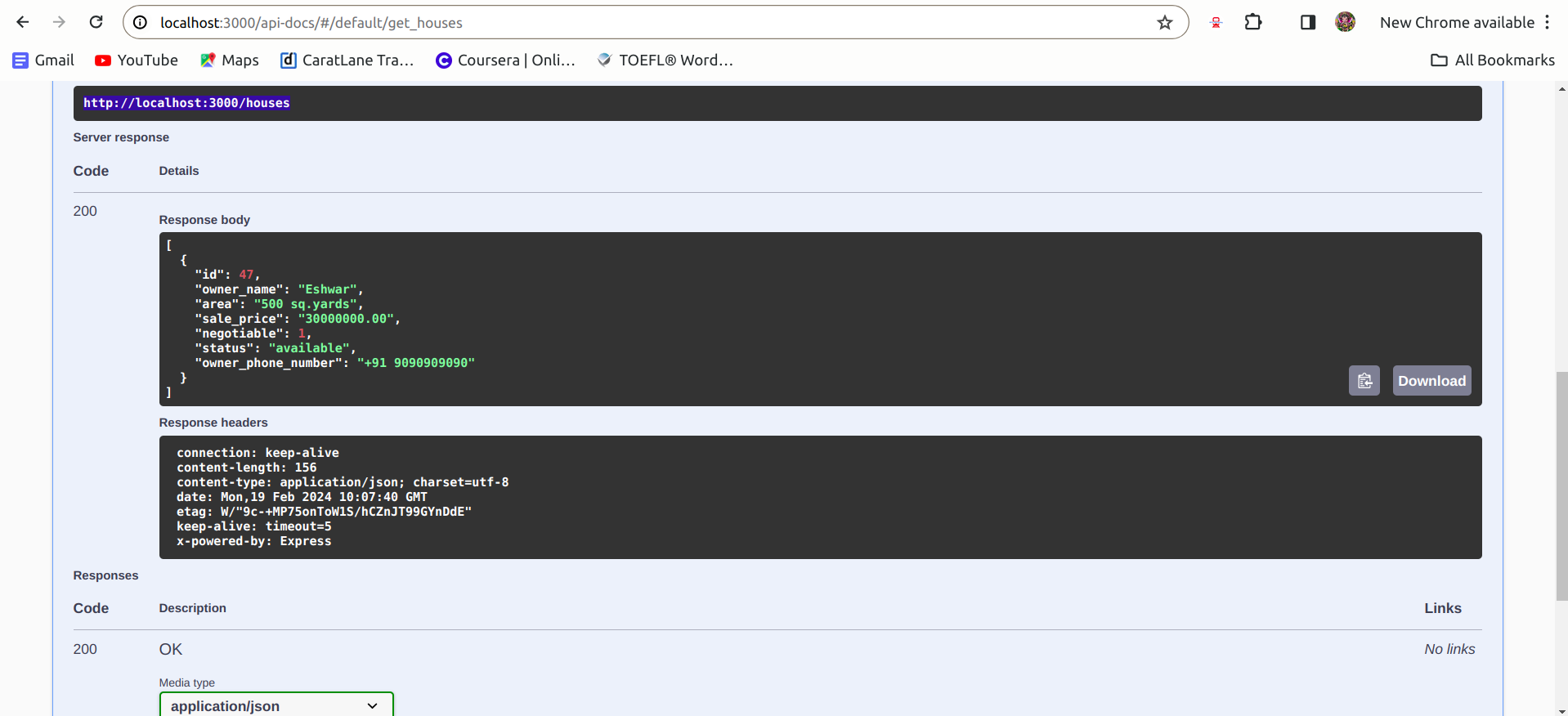


**2. Get List of Houses**

**Endpoint**: GET ((<http://localhost:3000/houses>))

**Description**: Retrieve a list of all houses.

**Response**:



3: **Get House Details by ID**

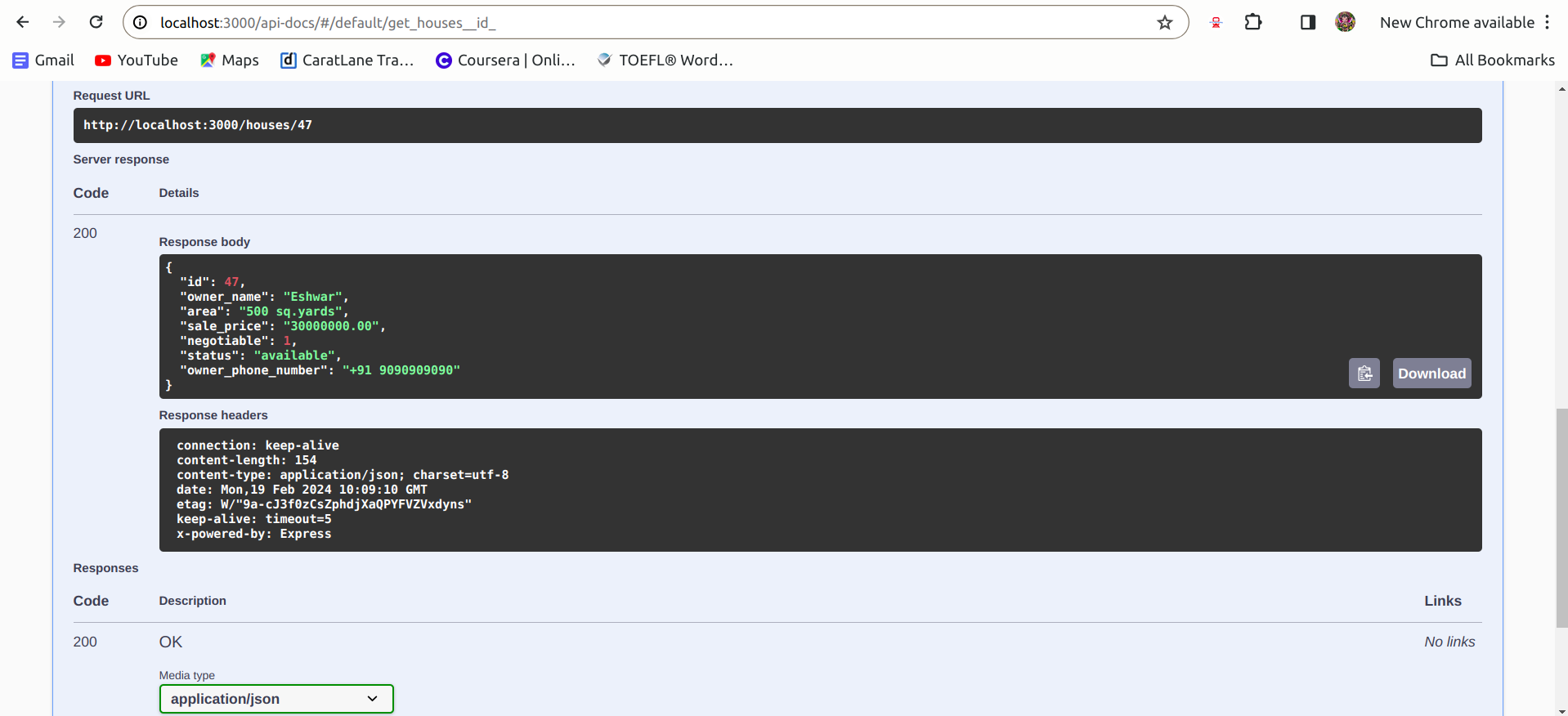
**Endpoint**: GET /houses/:id

**Description**: Retrieve details of a specific house by its ID.

**Example**:

curl http://localhost:3000/houses/1

**Response**:



4. **Update House Details**

**Endpoint**: PATCH /houses/:id

**Description**: Update details of a specific house identified by its ID.

**Request Body** (Optional):

{

"status": "available",

"owner\_phone\_number": "+91 9191919191"

}

**Response:**

