API Technical Documentation: Owner and Order Management Endpoints

---

Overview:

This API is designed for managing owner, menu, and order-related operations in a restaurant system. It allows users to perform tasks such as logging in, updating orders, editing menu items, and managing owner details. The views are rendered using **EJS templates**, and the database interactions are handled using **Mongoose**.

Base URL:

`/owner`

Endpoints

---

1. Login Page & Dashboard

GET /

- **Description:** Renders the login page or the owner dashboard if the user is logged in.
- **Request:**
  - Method: `GET`
  - URL: `/owner/`
- **Response:**
  - If logged in: Renders the `owners/layout.ejs` with a list of all orders.
  - If not logged in: Renders the `owners/login.ejs`.
- **Dependencies:**
  - Populates customer, driver, and menu data using the `Order` model.

---

2. Logout

GET /logout

- **Description:** Logs the user out by destroying the session and redirects to the login page.
- **Request:**
  - Method: `GET`
  - URL: `/owner/logout`

- **Response:**
    - Redirects to `/owner`.

---

3. User Login

POST /login

- **Description:** Authenticates the owner using email and password, creates a session for the user, and renders the dashboard.
- **Request:**
    - Method: `POST`
    - URL: `/owner/login`

Payload:
json
Copy code

```
{
  "email": "string",
  "password": "string"
}
```

    -
- **Response:**
    - On success: Renders the `owners/layout.ejs` with order data.
    - On failure: Renders the `owners/login.ejs` with an error message.
- **Errors:**
    - 400: Invalid email or password.
    - 500: Server error.

---

4. View Order Details

GET /orders/

/view

- **Description:** Fetches and displays details of a specific order by order ID.
- **Request:**
    - Method: `GET`
    - URL: `/owner/orders/:id/view`
- **Response:**

    - Renders `owners/order_view.ejs` with order details populated (customer, driver, and menu).

---

5. Edit Order

GET /orders/

/edit

- **Description:** Fetches an order and displays the edit form.
- **Request:**
    - Method: `GET`
    - URL: `/owner/orders/:id/edit`
- **Response:**
    - Renders `owners/order_edit.ejs` with order details populated (customer and driver).

---

6. Update Order Status

POST /orders/

/update

- **Description:** Updates the status of a specific order.
- **Request:**
    - Method: `POST`
    - URL: `/owner/orders/:id/update`

Payload:
json
Copy code

```
{
  "orderStatus": "string"
}
```

    -
- **Response:**
    - Redirects to `/owner/layout` after updating the order status and fetching the latest data.

---

7. View Menu

GET /menu

- **Description:** Fetches and displays all menu items associated with the owner.
- **Request:**
  - Method: `GET`
  - URL: `/owner/menu`
- **Response:**
  - Renders `owners/owner_view.ejs` with the owner and their restaurant menus (populated with associated image URLs).

---

8. Edit Owner Info

GET /info/edit

- **Description:** Fetches owner information and renders the edit form.
- **Request:**
  - Method: `GET`
  - URL: `/owner/info/edit`
- **Response:**
  - Renders `owners/owner_edit.ejs` with owner data populated.

---

9. Update Owner Info

POST /info/update

- **Description:** Updates the owner's personal information such as name, email, password, and restaurant name.
- **Request:**
  - Method: `POST`
  - URL: `/owner/info/update`

Payload:
json
Copy code

```
{
  "ownerId": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
```

```
  "password": "string",
  "restaurant_name": "string"
}
```

    -
- **Response:**
    - Redirects to `/owner/menu` after updating owner info.

---

10. Edit Menu

GET /menu/

/edit

- **Description:** Fetches menu details for editing.
- **Request:**
    - Method: `GET`
    - URL: `/owner/menu/:id/edit`
- **Response:**
    - Renders `owners/menu_edit.ejs` with the menu details (including image URL).

---

11. Update Menu

POST /menu/update

- **Description:** Updates a specific menu item, including its image.
- **Request:**
    - Method: `POST`
    - URL: `/owner/menu/update`

Payload:
json
Copy code

```
{
  "menuId": "string",
  "name": "string",
  "sku": "string",
  "description": "string",
  "price": "number",
```

```
  "inStock": "boolean"
}
```

    -
    - **File Upload:**
        - Field: `menuImage`
        - Type: Image (JPEG, PNG, etc.)
- **Response:**
    - Redirects to `/owner/menu` after successfully updating the menu item.

Data Models:

---

1. **Owner Model**
    - Fields: `firstName`, `lastName`, `email`, `password`, `restaurant_name`, `restaurant_menus`.
2. **Menu Model**
    - Fields: `name`, `sku`, `description`, `price`, `inStock`, `menu_images_url`.
3. **Order Model**
    - Fields: `customer`, `driver`, `order_Menus`, `orderStatus`.

Error Codes:

- **400 Bad Request:** Invalid input (e.g., incorrect email or password).
- **404 Not Found:** Resource not found (e.g., order, menu, or owner).
- **500 Internal Server Error:** Unexpected server failure.

Security:

- **Sessions:** Owner login status is maintained using session cookies (`req.session.loggedInUser`).
- **Password Hashing:** Owner passwords are hashed using bcrypt (`bcrypt.hash()`).

Dependencies:

- **Mongoose:** For database operations with MongoDB.

- **Bcrypt:** For hashing passwords.
- **Multer:** For handling file uploads (e.g., menu images).
- **Express-Session:** For managing user sessions.
- **FS Module:** To handle file system operations for storing image files.