**User Authentication (authController)**

signUp

Purpose: Registers a new user.

Process:

Validates that all required fields (username, email, password, role) are provided.

Checks if the user already exists by email.

Hashes the password using bcrypt.

Creates and saves a new user document.

Returns a success message upon completion.

signIn

Purpose: Authenticates a user and logs them in.

Process:

Validates that email and password are provided.

Finds the user by email.

Compares the provided password with the stored hashed password using bcrypt.

If valid, generates a JWT token (with user ID, email, and role) that expires in 5 hours.

Returns the token along with a success message.

**Menu Management (menuController)**

getMenu

Purpose: Retrieves the latest menu.

Process:

Uses a database query (findOne) to fetch the current menu document.

Returns the menu details to the client.

updateMenu

Purpose: Updates the menu items.

Process:

Validates that an array of menu items is provided in the request body.

If a menu document exists, it updates the items and refreshes the last updated timestamp; otherwise, it creates a new menu document.

Saves the changes and returns the updated menu.

**Order Management (orderController)**

createOrder

Purpose: Creates a new order.

Process:

Expects table number, order items, and total amount in the request.

Sets the order status to "pending" and saves the new order document.

Returns a success message along with the order details.

updateOrderStatus

Purpose: Updates the status of an existing order.

Process:

Expects the order ID (from URL parameters) and new status (from the request body).

Uses the order ID to find and update the order document.

Returns the updated order.

getOrderById

Purpose: Retrieves the details of a specific order.

Process:

Accepts an order ID as a URL parameter.

Queries the database to find the order and returns its details.

getOccupiedTables

Purpose: Retrieves a list of tables that are currently occupied.

Process:

Searches the QR Code collection for documents with a status of "occupied."

Returns the list of occupied table entries.

deleteAllTables

Purpose: Deletes all table entries from the QR Code collection.

Process:

Executes a delete operation on all QR Code documents.

Returns a message indicating that the tables have been deleted.

**QR Code Operations (qrController)**

generateStaticQR

Purpose: Generates a static QR code for a table (typically one-time per table).

Process:

Accepts a table number from the request body.

Checks if a QR code for that table already exists.

If not, it creates a QR URL (pointing to the order page), generates a PNG image using the QRCode library, saves the image, creates a new QR code document with a status of "available," and returns the QR details.

scanQR

Purpose: Marks a table as occupied when its QR code is scanned by a customer.

Process:

Expects table number and additional customer data (IP, phone OS, battery percentage, email) in the request.

Finds the corresponding QR code document.

Checks if the table is already occupied.

Assigns a session ID, updates the status to "occupied," sets an estimated vacant time (45 minutes later), stores customer data, and saves the changes.

transferSession

Purpose: Transfers an active session (and related orders) from one table to another.

Process:

Accepts a new table number and an existing session ID in the request body.

Finds the QR document for the session and retrieves the previous table number.

If the session exists and the table is different, updates the new table’s document with the session and customer data, resets the previous table's status to "available," and updates any related orders to reflect the new table number.

markTableAvailable

Purpose: Marks a table as available once it is no longer occupied.

Process:

Expects the table number in the request body.

Finds the corresponding QR code document.

Resets its status to "available," clears the session ID and customer data, and saves the document.

**Middleware**

authMiddleware / authenticate

Purpose: Validates JWT tokens to authenticate users.

Process:

Extracts the token from the "Authorization" header.

Verifies the token using the secret key from environment variables.

If valid, attaches the user data to the request (req.user) and allows the request to proceed.

authorizeAdmin

Purpose: Restricts certain routes to admin users only.

Process:

Checks if the user’s role (from req.user) is "admin."

If not, returns an error message denying access; otherwise, allows the request to continue.

**Data Models (Mongoose Schemas)**

User Model

Fields: username, email, password, and role (admin or staff).

Menu Model

Fields:

items: An array of objects (each with name, price, description, category, and availability).

lastUpdated: A timestamp set to the current date/time in the "Asia/Kolkata" timezone.

Order Model

Fields:

sessionId: For session tracking.

tableNumber: The table number associated with the order.

items: An array containing order items (with item ID, name, price, and quantity).

totalAmount: Total cost of the order.

status: Enum field with values like pending, preparing, completed, or cancelled.

QrCode Model

Fields:

tableNumber: Unique identifier for the table.

qrCodeUrl: URL/path of the generated QR code image.

sessionId: Tracks the active session if the table is occupied.

status: Indicates if the table is "occupied" or "available."

lastUpdated: Timestamp for the last update.

customerData: Stores additional details such as IP address, device OS, battery percentage, and email.

previousTable: Optionally stores the previous table number when sessions are transferred.

**Route Definitions**

Auth Routes (/api/auth)

POST /register: Calls signUp to register a new user.

POST /login: Calls signIn to authenticate the user and provide a JWT token.

Menu Routes (/api/menu)

GET /: Calls getMenu to fetch the current menu.

PUT /update: Calls updateMenu to update the menu; this route is protected (only accessible by admins).

Order Routes (/api/orders)

POST /: Calls createOrder to place a new order.

PUT /:orderId/status: Calls updateOrderStatus to change an order’s status.

GET /:orderId: Calls getOrderById to retrieve a specific order’s details.

GET /tables/occupied: Retrieves a list of occupied tables.

DELETE /tables/delete: Deletes all table entries.

QR Routes (/api/qr)

POST /generate: Calls generateStaticQR to generate a QR code for a table (requires authentication and admin privileges).

POST /scan: Calls scanQR to mark a table as occupied when its QR code is scanned.

POST /scan-table: Calls transferSession to move an active session from one table to another.

PUT /mark-available: Calls markTableAvailable to mark a table as available again.

GET /:tableNumber: Retrieves details of a specific table’s QR code entry.

DELETE /:tableNumber: Deletes a QR code entry for a table (admin-only).

Server Setup & Real-Time Integration

Main Server File

Initializes Express, applies middleware (like body-parser and CORS), and sets up static file serving (for QR code images).

Connects to MongoDB with error handling.

Defines API routes for auth, menu, orders, and QR code operations.

Creates an HTTP server and attaches Socket.IO for real-time communication (for example, to emit events when user data is updated).