🍽️ MEC Food App — Backend API Documentation

MadrasOne Campus Food Ordering System • Version 1.5.0 • Generated: February 15, 2026

🌐 **Production Base URL:** `https://backend.mec.welocalhost.com`
📖 **Swagger UI:** `https://backend.mec.welocalhost.com/swagger`
📄 **Swagger JSON:** `https://backend.mec.welocalhost.com/swagger.json`
🔑 **API Prefix:** `/api/v1`
🔒 **Auth:** Bearer JWT Token in `Authorization` header

📑 Table of Contents

🔐 Role-Based Access Control (RBAC)

Role	Level	Description
`student`	1	Place orders, manage wallet, view profile
`captain`	2	Shop staff — manage orders, QR verification
`owner`	3	Full shop control — menu, staff, analytics
`accountant`	4	Financial operations, user approvals, wallet credits
`superadmin`	5	Full system access — all operations

1. System Endpoints

GET `/` Public

Root endpoint — returns API info and available endpoints.

GET `/health` Public

Health check — returns server status, DB connection, memory usage, uptime.

GET `/version` Public

Returns API version and operational status.

2. Authentication

POST `/api/v1/auth/register` Public

Register a new student account. Rate limited: 5 req/hour.

Request Body

Field	Type	Required	Validation
`username`	string	Yes	4–20 chars, lowercase + numbers + underscore only
`password`	string	Yes	8–72 chars, must have uppercase, lowercase, number, special char
`name`	string	Yes	2–100 chars
`email`	string	*Optional*	Valid email
`phone`	string	Yes	10-digit Indian mobile (starts with 6-9)
`rollNumber`	string	Yes	1–20 chars, auto-uppercased
`department`	string	Yes	Valid department enum
`year`	number	Yes	1, 2, 3, or 4

POST `/api/v1/auth/login` **Public**

Login with username and password. Rate limited: 10 req/15min. Account locks after 5 failed attempts (15 min lockout).

Request Body

Field	Type	Required	Validation
`username`	string	**Yes**	Auto-lowercased
`password`	string	**Yes**	—
`deviceId`	string	*Optional*	Device identifier
`deviceInfo`	object	*Optional*	platform, language, screenResolution, timezone, brand, model, osVersion, etc.
`location`	object	*Optional*	`{ lat: number, lng: number }`

Response

Returns `accessToken`, `refreshToken`, and user object.

POST `/api/v1/auth/send-otp` **Public**

Send OTP to phone number for login. Rate limited: 5 req/10min.

Request Body

Field	Type	Required	Validation
`phone`	string	**Yes**	10-digit Indian mobile (starts with 6-9)

Response

Returns `sessionId` to be used with verify-otp.

POST `/api/v1/auth/verify-otp` **Public**

Verify OTP and login user.

Request Body

Field	Type	Required	Validation
`phone`	string	Yes	10-digit Indian mobile
`sessionId`	string	Yes	From send-otp response
`otp`	string	Yes	4–6 digits
`deviceId`	string	*Optional*	—
`deviceInfo`	object	*Optional*	Same as login

POST `/api/v1/auth/refresh` **Public**

Refresh access token using refresh token.

Request Body

Field	Type	Required	Validation
`refreshToken`	string	Yes	Valid refresh token

POST `/api/v1/auth/logout` **Auth Required**

Logout current user and invalidate tokens.

GET `/api/v1/auth/me` **Auth Required**

Get current authenticated user's information.

PUT `/api/v1/auth/change-password` **Auth Required**

Change current user's password.

Request Body

Field	Type	Required	Validation
`currentPassword`	string	**Yes**	—
`newPassword`	string	**Yes**	8–72 chars, upper + lower + digit + special

GET `/api/v1/auth/sessions` **Superadmin**

Get login sessions (superadmin only).

3. User Management

Student Routes

GET `/api/v1/student/profile` **Auth Required**

Get current user's profile (name, roll number, department, year, balance, avatar, etc.).

PUT `/api/v1/student/profile` **Auth Required**

Update current user's profile fields (name, email, phone, etc.).

PUT `/api/v1/student/profile/avatar` **Auth Required**

Upload avatar image. Multipart form: field `avatar` (image, max 5MB).

GET `/api/v1/student/leaderboard` **Auth Required**

Get student leaderboard ranked by total spending.

Accountant Routes

GET `/api/v1/accountant/pending-approvals` **Accountant+**

Get list of pending user approvals.

PUT `/api/v1/accountant/approve/:id` **Accountant+**

Approve a user registration. `:id` = User ID (ObjectId).

PUT `/api/v1/accountant/reject/:id` **Accountant+**

Reject a user registration.

GET `/api/v1/accountant/students` **Accountant+**

Get all approved students with their details.

GET `/api/v1/accountant/students/:id` **Accountant+**

Get a specific student with wallet summary.

Superadmin User Routes

GET `/api/v1/superadmin/users` **Superadmin**

Get all users with filters (role, status, search).

PUT `/api/v1/superadmin/users/:id/role` **Superadmin**

Update a user's role. Body: `{ "role": "captain" }`

PUT `/api/v1/superadmin/users/:id/deactivate` **Superadmin**

Deactivate a user account.

PUT `/api/v1/superadmin/users/:id/reactivate` **Superadmin**

Reactivate a deactivated user.

PUT `/api/v1/superadmin/users/:id/toggle-adhoc` **Superadmin**

Toggle ad-hoc payment creation privilege.

PUT `/api/v1/superadmin/users/:id/reset-password` **Superadmin**

Reset a user's password.

4. Wallet & Transactions

GET `/api/v1/student/wallet` **Auth Required**

Get current user's wallet balance.

GET `/api/v1/student/wallet/transactions` **Auth Required**

Get current user's transaction history. Query params: `type` (credit/debit/refund), `startDate`, `endDate`, `page`, `limit`.

POST `/api/v1/accountant/students/:id/credit` **Accountant+**

Credit a student's wallet.

Field	Type	Required	Validation
`amount`	number	Yes	Positive, max 100,000
`source`	string	Yes	cash_deposit | online_payment | complementary | pg_direct | adjustment
`description`	string	*Optional*	Max 500 chars

POST `/api/v1/accountant/students/:id/debit` **Accountant+**

Debit a student's wallet.

Field	Type	Required	Validation
`amount`	number	Yes	Positive, max 100,000
`description`	string	Yes	1–500 chars

GET `/api/v1/accountant/transactions` **Accountant+**

Get all transactions. Query: `userId`, `type`, `source`, `status`, `startDate`, `endDate`, `page`, `limit`.

GET `/api/v1/accountant/vendor-payables` **Accountant+**

Get vendor payable amounts for all shops.

POST `/api/v1/accountant/vendor-transfers` **Accountant+**

Update vendor transfer status.

Field	Type	Required	Validation
`shopId`	string	Yes	ObjectId format
`period`	string	Yes	YYYY-MM format
`amount`	number	Yes	Non-negative
`status`	string	Yes	pending I completed
`notes`	string	*Optional*	Max 500 chars

5. Shop Management

Public Routes

GET `/api/v1/shops` **Public**

List all active shops. Query: `activeOnly`, `category`.

GET `/api/v1/shops/:id` **Public**

Get a specific shop's details by ID.

Superadmin Shop Routes

POST `/api/v1/superadmin/shops` **Superadmin**

Create a new shop.

Field	Type	Required	Validation
`name`	string	**Yes**	2–100 chars
`description`	string	*Optional*	Max 500 chars
`category`	string	**Yes**	Valid shop category enum
`ownerId`	string	*Optional*	Existing user ObjectId
`ownerDetails`	object	*Optional*	Create new owner: `{ name, email, password, phone? }`
`imageUrl`	string	*Optional*	Valid URL
`bannerUrl`	string	*Optional*	Valid URL
`operatingHours`	array	*Optional*	`[{ day, openTime (HH:mm), closeTime, isClosed }]`
`contactPhone`	string	*Optional*	10-digit Indian mobile

PUT `/api/v1/superadmin/shops/:id` **Superadmin**

Update a shop. Same fields as create (all optional).

DELETE `/api/v1/superadmin/shops/:id` **Superadmin**

Deactivate a shop (soft delete).

PATCH `/api/v1/superadmin/shops/:id/toggle` **Superadmin**

Toggle shop active/inactive status.

PATCH `/api/v1/superadmin/shops/:id/toggle-qr` **Superadmin**

Toggle QR payment capability for a shop.

6. Menu Management

Global Public Routes

GET `/api/v1/menu/items` **Public**

Get all menu items from all shops.

GET `/api/v1/menu/offers` **Public**

Get all active offers from all shops.

Shop-Specific Public Routes

GET `/api/v1/shops/:shopId/menu` **Public**

Get menu items for a specific shop. Query: `categoryId`, `search`, `availableOnly`, `vegetarianOnly`, `minPrice`, `maxPrice`.

GET `/api/v1/shops/:shopId/categories` **Public**

Get categories for a specific shop.

GET `/api/v1/shops/:shopId/offers` **Public**

Get active offers for a specific shop.

Owner Menu Routes

POST `/api/v1/owner/categories` **Owner+**

Create category for own shop.

Field	Type	Required	Validation
`name`	string	**Yes**	2–50 chars
`description`	string	*Optional*	Max 200 chars
`icon`	string	*Optional*	—
`sortOrder`	number	*Optional*	Int, min 0, default 0

POST `/api/v1/owner/menu` **Owner+**

Add menu item to own shop.

Field	Type	Required	Validation
`name`	string	**Yes**	2–100 chars
`description`	string	*Optional*	Max 500 chars
`price`	number	**Yes**	Positive
`costPrice`	number	*Optional*	Min 0
`categoryId`	string	*Optional*	ObjectId
`imageUrl`	string	*Optional*	Valid URL
`isVegetarian`	boolean	*Optional*	Default: false
`isInstant`	boolean	*Optional*	Default: false
`preparationTime`	number	*Optional*	1–180 minutes
`tags`	string[]	*Optional*	Array of strings
`nutritionInfo`	object	*Optional*	`{ calories, protein, carbs, fat }`

PUT `/api/v1/owner/menu/:id` **Owner+**

Update a menu item (same fields as create, all optional).

PATCH `/api/v1/owner/menu/:id/availability` **Owner+**

Toggle menu item availability (available/unavailable).

POST `/api/v1/owner/menu/:id/offer` **Owner+**

Set offer on a menu item.

Field	Type	Required	Validation
`offerPrice`	number	**Yes**	Positive
`offerEndDate`	string	**Yes**	ISO 8601 date, must be in the future

DELETE `/api/v1/owner/menu/:id/offer` **Owner+**

Remove offer from a menu item.

DELETE `/api/v1/owner/menu/:id` **Owner+**

Delete a menu item.

Superadmin Menu Routes

GET `/api/v1/superadmin/menu` **Superadmin**

Get all menu items including unavailable ones.

POST `/api/v1/superadmin/categories` **Superadmin**

Create category for any shop.

POST `/api/v1/superadmin/menu` **Superadmin**

Add menu item to any shop.

PUT `/api/v1/superadmin/menu/:id` **Superadmin**

Update any menu item.

DELETE `/api/v1/superadmin/menu/:id` **Superadmin**

Delete any menu item.

7. Orders

POST `/api/v1/orders` **Auth Required**

Create a new food order. Rate limited: 5 req/5min.

Field	Type	Required	Validation
`shopId`	string	**Yes**	ObjectId
`items`	array	**Yes**	1–20 items, each: `{ foodItemId: ObjectId, quantity: 1–50 }`
`notes`	string	*Optional*	Max 500 chars

POST `/api/v1/orders/laundry` **Auth Required**

Create a laundry order.

Field	Type	Required	Validation
`shopId`	string	**Yes**	ObjectId
`items`	array	**Yes**	1–10 items: `{ category: laundry_enum, count: 1–100 }`
`specialInstructions`	string	*Optional*	Max 500 chars

POST `/api/v1/orders/stationery` **Auth Required**

Create a stationery/printing order.

Field	Type	Required	Validation
`shopId`	string	**Yes**	ObjectId
`pageCount`	number	**Yes**	1–1000
`copies`	number	**Yes**	1–100
`colorType`	string	**Yes**	bw I color
`paperSize`	string	**Yes**	Valid paper size enum
`doubleSided`	boolean	*Optional*	Default: false
`specialInstructions`	string	*Optional*	Max 500 chars

GET `/api/v1/orders/my` **Auth Required**

Get current user's orders. Query: `page`, `limit`, `status` (comma-separated), `startDate`, `endDate`.

GET `/api/v1/orders/:id` **Auth Required**

Get a specific order by ID (ownership checked).

POST `/api/v1/orders/verify-qr` **Captain/Owner**

Verify QR code for order pickup.

Field	Type	Required	Validation
`qrData`	string	**Yes**	QR code string

GET `/api/v1/orders/shop` **Captain/Owner+**

Get shop orders. Query: `page`, `limit`, `status`, `startDate`, `endDate`.

GET `/api/v1/orders/shop/active` **Captain/Owner+**

Get active (in-progress) orders for the shop.

GET `/api/v1/orders/shop/stats` **Captain/Owner+**

Get shop order statistics (counts by status, revenue, etc.).

GET `/api/v1/orders/shop/analytics` **Captain/Owner+**

Get shop analytics data (trends, top items, peak hours).

PUT `/api/v1/orders/:id/status` **Captain/Owner**

Update order status.

Field	Type	Required	Validation
`status`	string	**Yes**	pending | preparing | ready | completed | cancelled
`cancellationReason`	string	*Optional*	Max 500 chars (for cancellations)

PATCH `/api/v1/orders/:id/items/:itemIndex/deliver` **Captain/Owner**

Mark individual item as delivered/undelivered (toggle).

POST `/api/v1/orders/:id/complete` **Captain/Owner**

Complete order (after QR verification).

8. Razorpay Payments

POST `/api/v1/razorpay/create-order` **Auth Required**

Create a Razorpay order for wallet top-up. Rate limited: 3 req/5min.

Field	Type	Required	Validation
`amount`	number	**Yes**	Rs. 1 – Rs. 50,000

POST `/api/v1/razorpay/verify-payment` **Auth Required**

Verify Razorpay payment after completion.

Field	Type	Required	Validation
`razorpay_order_id`	string	**Yes**	Format: `order_*`
`razorpay_payment_id`	string	**Yes**	Format: `pay_*`
`razorpay_signature`	string	**Yes**	64-char hex HMAC

POST `/api/v1/razorpay/webhook` **Webhook**

Razorpay webhook handler. Receives raw body for HMAC signature verification. Registered before `express.json()` .

9. Ad-hoc Payments

Superadmin Payment Routes

POST `/api/v1/superadmin/payments` **Superadmin**

Create a new payment request.

Field	Type	Required	Validation
`title`	string	**Yes**	3–100 chars
`description`	string	**Yes**	10–500 chars
`amount`	number	**Yes**	Positive, max 100,000
`targetType`	string	**Yes**	all | selected | department | year
`targetStudents`	string[]	If selected	Array of user ObjectIds
`targetDepartment`	string	If department	Valid department enum
`targetYear`	number	If year	1–4
`dueDate`	string	*Optional*	Valid date
`isVisibleOnDashboard`	boolean	*Optional*	Default: true

GET `/api/v1/superadmin/payments` **Superadmin**

Get all payment requests. Query: `status`, `page`, `limit`.

GET `/api/v1/superadmin/payments/:id` **Superadmin**

Get a single payment request by ID.

PUT `/api/v1/superadmin/payments/:id` **Superadmin**

Update a payment request (title, description, isVisibleOnDashboard, dueDate).

POST `/api/v1/superadmin/payments/:id/close` **Superadmin**

Close or cancel a payment request. Body: `{ "status": "closed" | "cancelled" }`

GET `/api/v1/superadmin/payments/:id/students` **Superadmin**

Get students with payment status. Query: `status` (paid/pending/all), `search`, `page`, `limit`.

POST `/api/v1/superadmin/payments/:id/remind` **Superadmin**

Send payment reminders to unpaid students.

Student Payment Routes

GET `/api/v1/student/payments/pending` **Auth Required**

Get pending payments for logged-in student.

POST `/api/v1/student/payments/:id/pay` **Auth Required**

Pay a pending payment request (deducts from wallet).

GET `/api/v1/student/payments/history` **Auth Required**

Get payment history. Query: `status` (paid/pending/all), `page`, `limit`.

Privileged User Payment Routes

POST `/api/v1/user/payments/create-onetime` **canCreateAdhoc**

Create a one-time payment (requires canCreateAdhoc privilege). Body: `{ title, description, amount }`.

GET `/api/v1/user/payments/my-requests` **Auth Required**

Get payment requests created by current user.

10. Owner Operations

GET `/api/v1/owner/shop` **Captain+**

Get owner's shop details.

PATCH `/api/v1/owner/shop/toggle` **Captain+**

Toggle shop open/closed status.

POST `/api/v1/owner/captains` **Captain+**

Create a new captain (shop staff).

Field	Type	Required	Validation
`name`	string	Yes	2–100 chars
`email`	string	Yes	Valid email
`password`	string	Yes	8+ chars, upper + lower + digit
`phone`	string	*Optional*	10-digit Indian mobile

GET `/api/v1/owner/captains` **Captain+**

List all captains for the owner's shop.

DELETE `/api/v1/owner/captains/:id` **Captain+**

Remove a captain (deactivate).

POST `/api/v1/owner/qr-payments` **Captain+**

Create a QR payment.

Field	Type	Required	Validation
`title`	string	Yes	3–100 chars
`description`	string	Yes	10–500 chars
`amount`	number	Yes	Positive, max 100,000

GET `/api/v1/owner/qr-payments` **Captain+**

List QR payments for the shop.

11. File Uploads

GET `/api/v1/uploads/status` **Public**

Check if storage (Garage S3) is configured and operational.

POST `/api/v1/uploads/image` Captain+

Upload a base64 encoded image. Rate limited: 10 req/10min.

Field	Type	Required	Validation
`image`	string	Yes	Base64 encoded image
`filename`	string	Yes	—
`folder`	string	*Optional*	Storage folder

POST `/api/v1/uploads/presigned` Captain+

Get a presigned URL for direct upload to Garage S3.

Field	Type	Required	Validation
`filename`	string	Yes	—
`folder`	string	*Optional*	Storage folder

DELETE `/api/v1/uploads/:key` Captain+

Delete an uploaded file by key (URL encoded).

12. Image Proxy

GET `/api/v1/images/:folder/:filename` Public

Proxy image from Garage S3 storage. Example: `/api/v1/images/meccanteen/idly.png`. Path validated with whitelist to prevent traversal attacks.

13. Superadmin Dashboard

GET `/api/v1/superadmin/dashboard/stats` Superadmin

Get dashboard statistics (total users, orders, revenue, shops).

GET `/api/v1/superadmin/orders` Superadmin

Get all orders across all shops.

GET `/api/v1/superadmin/orders/stats` Superadmin

Get order statistics across all shops.

GET `/api/v1/superadmin/transactions/collections` Superadmin

Get transaction collection status (monthly collections info).

POST `/api/v1/superadmin/transactions/migrate` Superadmin

Migrate existing transactions to monthly collections.

GET `/api/v1/superadmin/diagnose/owner-shop` Superadmin

Diagnose owner-shop relationships and find issues.

POST `/api/v1/superadmin/fix/owner-shop` Superadmin

Link an owner to a shop (fix broken relationships). Body: `{ "ownerId": "...", "shopId": "..." }`

POST `/api/v1/superadmin/users/create` Superadmin

Create a new user with any role. Body: `{ username, password, name, email?, role, phone?, rollNumber?, department?, year?, shopId? }`

14. WebSocket (Real-time via Socket.IO)

URL: `wss://backend.mec.welocalhost.com`
Auth: JWT token required in handshake — `{ auth: { token: "Bearer ..." } }`
Room Patterns:

- `order:<orderId>` — Subscribe to updates for a specific order
- `vendor:<shopId>` — Subscribe to shop's incoming order feed

Events Emitted:

- `orderUpdate` — When an order's status changes
- `newOrder` — When a new order is placed at a shop

15. Error Handling

All error responses follow the format:

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable message",
    "details": [...] // optional, for validation errors
  }
}
```

HTTP Status	Code	Description
400	VALIDATION_ERROR	Request body/params/query validation failed
400	INVALID_ID	Invalid MongoDB ObjectId format
401	AUTH_ERROR	Missing or invalid JWT token
403	FORBIDDEN	Insufficient role/permissions
404	NOT_FOUND	Resource not found
409	DUPLICATE_KEY	Record with this value already exists
422	VALIDATION_ERROR	Mongoose schema validation error
429	RATE_LIMIT_EXCEEDED	Too many requests — rate limit hit
500	INTERNAL_ERROR	Unexpected server error

503	SERVICE_UNAVAILABLE	Database disconnected

MEC Food App • Backend API Documentation v1.5.0
Production: `https://backend.mec.welocalhost.com`
Generated: February 15, 2026