AIU Smart Café Assistant - API Documentation

Base URL: `http://localhost:3000/api` (Local) / `https://your-project.vercel.app/api` (Production) **Authentication:** All secured endpoints require a header: `Authorization: Bearer <token>`

---

1. Authentication Module

Used by: Frontend (Login/Register Pages)

Register New Student

Create a new user account.

- **Endpoint:** `POST /auth/register`
- **Auth Required:** No

Request Body:
JSON
```
{
  "name": "Ali Student",
  "email": "ali@student.aiu.edu.my",
  "password": "password123"
}
```

-
- **Success Response (201):** `{ "message": "User created successfully" }`

Login

Authenticate a user and receive a JWT token.

- **Endpoint:** `POST /auth/login`
- **Auth Required:** No

Request Body:
JSON
```
{
  "email": "ali@student.aiu.edu.my",
  "password": "password123"
}
```

-

Success Response (200):
JSON
```
{
  "token": "eyJhbGciOiJIUzI1...",
  "user": {
    "id": "654abc...",
    "name": "Ali Student",
    "role": "student"
  }
}
```

- **Frontend Note:** Save the `token` in LocalStorage/Cookies. Send it in the Header for all future requests.

---

2. Menu Module

Used by: Frontend (Student Menu, Owner Dashboard)

Get Public Menu (Student View)

Fetch all available menu items. Supports filtering.

- **Endpoint:** `GET /menu`
- **Auth Required:** No
- **Query Parameters (Optional):**
  - `?category=Main Course` (Filter by category)
  - `?maxPrice=10` (Filter items under RM 10)
  - `?search=nasi` (Search by name)

Success Response (200):
JSON
```
[
  {
    "_id": "item_123",
    "name": "Nasi Lemak",
    "price": 650,
    "imageUrl": "https://...",
    "category": "Main Course"
  }
]
```

-

Add Menu Item (Admin Only)

- **Endpoint:** `POST /admin/menu`
- **Auth Required:** Yes (Role: Admin)

Request Body:

```JSON
{
  "name": "Chicken Rice",
  "price": 800,
  "category": "Main Course",
  "description": "Roasted chicken with rice",
  "isAvailable": true
}
```

-

Update Menu Item (Admin Only)

- **Endpoint:** `PUT /admin/menu/[id]`
- **Auth Required:** Yes (Role: Admin)
- **Request Body:** `{ "price": 850, "isAvailable": false }`

Delete Menu Item (Admin Only)

- **Endpoint:** `DELETE /admin/menu/[id]`
- **Auth Required:** Yes (Role: Admin)

---

3. Order Module

Used by: Frontend (Cart, Order History) & ML (Data Source)

Place Order

- **Endpoint:** `POST /orders`
- **Auth Required:** Yes

Request Body:

```JSON
{
  "items": [
    { "itemId": "item_123", "quantity": 2 },
    { "itemId": "item_456", "quantity": 1 }
  ]
}
```

-
- **Success Response (201):** Returns the created Order object with `status: "Pending"`.

Get My Orders

Fetch history for the logged-in student.

- **Endpoint:** `GET /orders`
- **Auth Required:** Yes
- **Success Response (200):** List of orders sorted by newest first.

Get All Orders (Admin/Staff)

Fetch all orders for the kitchen display.

- **Endpoint:** `GET /admin/orders`
- **Auth Required:** Yes (Role: Staff/Admin)
- **Query Params:** `?status=Pending` (Optional)

Update Order Status (Admin/Staff)

- **Endpoint:** `PUT /admin/orders/[id]`
- **Auth Required:** Yes (Role: Staff/Admin)

Request Body:
JSON
{ "status": "Ready" }

- **Valid Statuses:** `Pending`, `Preparing`, `Ready`, `Completed`, `Cancelled`

---

4. Feedback & AI Module

Used by: Frontend (Reviews) & Python ML Service

Submit Feedback

Saves review and triggers Sentiment Analysis.

- **Endpoint:** `POST /feedback`
- **Auth Required:** Yes

Request Body:
JSON
{

```
  "itemId": "item_123",
  "orderId": "order_999",
  "rating": 5,
  "textReview": "The food was amazing!"
}
```

-
- **Behavior:**
  1. Saves feedback to MongoDB.
  2. **Backend** automatically calls Python ML Service.
  3. Updates feedback with `sentimentCategory` (Positive/Negative).

Get Personalized Recommendations

Fetches food suggestions for the user.

- **Endpoint:** `GET /recommendations`
- **Auth Required:** Yes
- **Behavior:**
  1. **Backend** calls Python ML Service: `GET /recommend/<userId>`
  2. Python Service calculates preferences based on order history.
  3. **Backend** returns full menu item details (Name, Image, Price).

Success Response (200):
JSON
```
{
  "recommendations": [
    { "name": "Nasi Lemak", "category": "Main Course", ... },
    { "name": "Teh Tarik", "category": "Drink", ... }
  ]
}
```

-

---

5. Integration Notes for ML Engineer (Fawad)

How the Backend calls your Python Service:

1. **Sentiment Analysis:**
   - Next.js calls: `POST https://your-render-url.com/predict/sentiment`
   - Payload sent to you: `{ "text": "The food was amazing" }`
   - Expected Response from you: `{ "sentiment": "Positive", "score": 0.95 }`
2. **Recommendations:**

- Next.js calls: `GET https://your-render-url.com/recommend/<userId>`
- Expected Response from you: `{ "recommendations": ["item_id_1", "item_id_2"] }`
- **Note:** You must connect to the **Same MongoDB Atlas Cluster** to read the user's Order History for training/prediction.