1. Register

This API allows you to add a new employee to the database. Employee_id will be auto-incremented.The default role when registering is "Employee".

URL	http://localhost:8000/api/users/register
Method	POST
URL Params	<none>
Header	Content-Type: application/json

Data Params	{
	"name": "<user_name>",
	"email": "<user_email>",
	"password": "<user_password>",
	"role": "<user_role>" // Optional, default is "Employee"
	}

Success Response	Code: 201
	Content:
	{
	"message": "User registered successfully!"
	}

Error Response	**Code**: 400 Bad Request
	Content:
	{
	"message": "Missing required fields"
	}
	
	Code: 400 Bad Request (when email or username already exists)
	Content:
	{
	"message": "Email already exists. Username already exists."
	}
	
	Code: 500 Internal Server Error
	Content:
	{
	"message": "Error querying database",
	"error": "<error_details>"
	}

Sample Call	final response = await http.post(Uri.parse('http://localhost:8000/api/users/register'), body: json.encode({ 'name': 'john_doe', 'email': 'john.doe@example.com', 'password': 'securePassword123', 'role': 'Employee' // Optional, default is "Employee" }), headers: {'Content-Type': 'application/json'},);
Notes	<none>

2. Login

This API allows an existing user to log in using their username and password. Upon successful login, a JWT token is generated for authentication.

URL	http://localhost:8000/api/users/login
Method	POST
URL Params	<none>
Header	Content-Type: application/json

Data Params	{
	"name": "<user_name>",
	"password": "<user_password>"
	}

Success Response	Code: 201
	Content:
	{
	"token": "<jwt_token>",
	"role": "<user_role>",
	"id": "<user_id>"
	}

Error Response	**Code**: 400 Bad Request
	Content:
	{
	"message": "Username and Password are required."
	}
	Code: 400 Bad Request (Incorrect username or password)
	Content:
	{
	"message": "Account does not exist or incorrect password."
	}
	Code: 500 Internal Server Error
	Content:
	{
	"message": "An error occurred while querying the database.",
	"error": "<error_details>"
	}

Sample Call	final response = await http.post(Uri.parse('http://localhost:8000/api/users/login'), body: json.encode({ 'name': 'john_doe', 'password': 'securePassword123' }), headers: {'Content-Type': 'application/json'},);
Notes	Ensure that the username and password provided match the credentials in the database. The generated JWT token will expire in 1 hour.

3. Get all user

This API allows an admin to fetch the list of all users from the database. It returns the id, name, email, and role of each user.

URL	http://localhost:8000/api/users/users
Method	GET
URL Params	<none>
Header	Content-Type: application/json Authorization: Bearer <jwt_token>
Data Params	<none>

Success Response	Code: 200 OK
	Content:
	{
	  "token": "<jwt_token>",
	  "role": "<user_role>",
	  "id": "<user_id>"
	}

Error Response	**Code**: 400 Bad Request
	Content:
	[
	  {
	    "id": "<user_id>",
	    "name": "<user_name>",
	    "email": "<user_email>",
	    "role": "<user_role>"
	  },
	  {
	    "id": "<user_id>",
	    "name": "<user_name>",
	    "email": "<user_email>",
	    "role": "<user_role>"
	  }
	]
	Code: 401 Unauthorized
	Content:
	{
	  "message": "Authorization token is missing or invalid."
	}
	Code: 500 Internal Server Error
	Content:
	{
	  "message": "Error fetching users",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.get(

	Uri.parse('http://localhost:8000/api/users'),
	  headers: {
	    'Content-Type': 'application/json',
	    'Authorization': 'Bearer <jwt_token>'
	  },
	);

Notes	This API requires the Admin role for access. The authorization token must be included in the Authorization header in the format: Bearer <jwt_token>.

4. Delete User

This API allows an admin to delete a user from the database by specifying the user's id. It will return a success message if the user is successfully deleted or an error message if the user does not exist.

URL	http://localhost:8000/api/users/users/:id
Method	DELETE
URL Params	id (required): The ID of the user to be deleted.
Header	Content-Type: application/json Authorization: Bearer <jwt_token>
Data Params	<none>

Success Response	Code: 200 OK
	Content:
	{
	  "message": "User deleted successfully"
	}

Error Response	**Code**: 400 Bad Request **Content**: { "message": "Invalid user ID" } **Code**: 404 Not Found **Content**: { "message": "User not found" } **Code:** 401 Unauthorized **Content:** { "message": "Authorization token is missing or invalid." } **Code**: 500 Internal Server Error **Content**: { "message": "Error fetching users", "error": "<error_details>" }
Sample Call	final response = await http.delete(Uri.parse('http://localhost:8000/api/users/1'), // Replace '1' with the actual user ID to delete headers: { 'Content-Type': 'application/json', 'Authorization': 'Bearer <jwt_token>' },);
Notes	This API requires the Admin role for access. The authorization token must be included in the Authorization header in the format: Bearer <jwt_token>. If the user does not exist or the ID is invalid, a 404 error will be returned.

5. Get all product

This API allows you to retrieve a list of all products in the database. It returns the details of the products in JSON format.

URL	http://localhost:8001/api/products
Method	GET
URL Params	<none>
Header	Content-Type: application/json Authorization: Bearer <jwt_token>
Data Params	<none>

Success Response	Code: 200 OK
	Content:
	[
	  {
	    "id": "<product_id>",
	    "name": "<product_name>",
	    "price": "<product_price>",
	    "quantity": "<product_quantity>"
	  },
	  {
	    "id": "<product_id>",
	    "name": "<product_name>",
	    "price": "<product_price>",
	    "quantity": "<product_quantity>"
	  }
	]

Error Response	**Code:** 500 Internal Server Error
	Content:
	{
	  "message": "Error fetching products",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.get(
	  Uri.parse('http://localhost:8001/api/products'),
	  headers: {
	    'Content-Type': 'application/json',
	    'Authorization': 'Bearer <jwt_token>'
	  },
	);

Notes	This API retrieves all products stored in the database. If an error occurs while fetching the products, a 500 error with details will be returned.

6. Get product by Id

This API allows you to retrieve a single product from the database based on its ID. It returns the product's details in JSON format. Access is secured with a JWT token, which must be included in the Authorization header.

URL	http://localhost:8001/api/products/:id
Method	GET
URL Params	id=[integer]
Header	Content-Type: application/json Authorization: Bearer <jwt_token>
Data Params	<none>

Success Response	Code: 200 OK
	Content:
	{
	  "id": "<product_id>",
	  "name": "<product_name>",
	  "price": "<product_price>",
	  "quantity": "<product_quantity>"
	}

Error Response	**Code: 400 Bad Request**
	Content:
	{
	  "message": "Invalid product ID format."
	}
	Code: 401 Unauthorized
	Content:
	{
	  "message": "Unauthorized access. Please provide a valid token."
	}
	Code: 404 Not Found
	Content:
	{
	  "message": "Product not found"
	}
	Code: 500 Internal Server Error
	Content:
	{
	  "message": "Error fetching product",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.get(
	  Uri.parse('http://localhost:8001/api/products/1'), // Example product ID: 1
	  headers: {
	    'Content-Type': 'application/json',
	    'Authorization': 'Bearer <jwt_token>'
	  },
	);

Notes	This API retrieves a single product based on the provided product ID.

7. Add product

This API allows you to add a new product to the database. The product's details, including name, code, price, quantity, description, and image URL, must be provided in the request body. The API is secured with JWT authentication.

URL	http://localhost:8001/api/products
Method	POST
URL Params	<none>
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	"name": "<product_name>", // Product name
	"code": "<product_code>", // Product code
	"price": <product_price>, // Product price
	"quantity": <product_quantity>, // Available quantity
	"description": "<product_description>", // Optional product description
	"image_url": "<image_url>" // Optional product image URL
	}

Success Response	Code: 200 Created
	Content:
	{
	"message": "Product added successfully",
	"id": "<new_product_id>"
	}

Error Response	**Code: 400 Bad Request**
	Content:
	{
	"message": " Missing required fields"
	}
	Code: 401 Unauthorized
	Content:
	{
	"message": "Unauthorized access. Please provide a valid token."
	}
	Code: 500 Internal Server Error
	Content:
	{
	"message": "Error fetching product",
	"error": "<error_details>"
	}

Sample Call	final response = await http.post(Uri.parse('http://localhost:8001/api/products'), body: json.encode({ 'name': 'Product ABC', 'code': 'ABC123', 'price': 19.99, 'quantity': 50, 'description': 'A great product',

	'image_url': 'http://example.com/image.jpg' }), headers: { 'Content-Type': 'application/json', 'Authorization': 'Bearer <jwt_token>' },);
Notes	This API requires authentication with a valid JWT token in the Authorization header. The name, code, price, and quantity fields are required. If any of them are missing, a 400 error will be returned. If the product is added successfully, the new product's ID will be returned in the response.

8. Update product

This API allows updating the price and quantity of a product based on its ID. Both fields are required to be provided in the request body.

URL	http://localhost:8001/api/products/:id
Method	PUT
URL Params	id = Product ID (required)
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	"price": <new_price>,
	"quantity": <new_quantity>
	}

Success Response	Code: 200 OK
	Content:
	{
	"message": "Product updated successfully"
	}

Error Response	**Code: 400 Bad Request**
	Content:
	{
	  "message": " Missing price or quantity"
	}
	Code: 404 Not Found
	Content:
	{
	  "message": "Product not found"
	}
	Code: 500 Internal Server Error
	Content:
	{
	  "message": "Error fetching product",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.put(
	  Uri.parse('http://localhost:8001/api/products/1'),
	  body: json.encode({
	    'price': 29.99,
	    'quantity': 100
	  }),
	  headers: {
	    'Content-Type': 'application/json',
	    'Authorization': 'Bearer <jwt_token>'
	  },
	);

Notes	<none>

9. Delete product

This API deletes a product from the database based on its ID.

URL	http://localhost:8001/api/products/:id
Method	DELETE
URL Params	id = Product ID (required)
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	  "price": <new_price>,
	  "quantity": <new_quantity>
	}

Success Response	Code: 200 OK
	Content:
	{
	  "message": "Product deleted successfully"
	}

Error Response	**Code: 404 Not Found**
	Content:
	{
	  "message": "Product not found"
	}
	Code: 500 Internal Server Error
	Content:
	{
	  "message": "Error fetching product",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.delete(
	  Uri.parse('http://localhost:8001/api/products/1'),
	  headers: {
	    'Content-Type': 'application/json',
	    'Authorization': 'Bearer <jwt_token>'
	  },
	);

Notes	<none>

10. Update Product Quantity

This API updates the product quantity after a sale (or payment). It decreases the product's stock based on the provided id and quantity.

URL	http://localhost:8001/api/products/update-quantity
Method	POST
URL Params	<none>
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	  "id": <product_id>,
	  "quantity": <quantity_to_decrease>
	}

Success Response	Code: 200 OK
	Content:
	{
	"message": "Product quantity updated successfully"
	}

Error Response	**Code: 400 Bad Request**
	Content:
	{
	  "message": "Missing productId or quantity"
	}
	Code: 400 Bad Request(if stock is insufficient)
	Content:
	{
	"message": "Not enough stock"
	}
	Code: 500 Internal Server Error
	Content:
	{
	  "message": "Error fetching product",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.post(
	  Uri.parse('http://localhost:8001/api/products/update-quantity'),
	  body: json.encode({
	    'id': 1,
	    'quantity': 10
	  }),
	  headers: {
	    'Content-Type': 'application/json',
	    'Authorization': 'Bearer <jwt_token>'
	  },
	);

Notes	<none>

11. Get All Invoices

This API allows fetching all invoices from the database. It returns a list of invoices, with each invoice's created_at and updated_at formatted.

URL	http://localhost:8002/api/invoices
Method	GET
URL Params	<none>
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	  "id": <product_id>,
	  "quantity": <quantity_to_decrease>
	}

Success Response	Code: 200 OK
	Content:
	[
	  {
	    "invoice_id": <invoice_id>,
	    "user_id": <user_id>,
	    "total_amount":<total_amount>,
	    "created_at":<created_at>,
	    "updated_at:<updated_at>
	  }
	]

Error Response	**Code: 500 Internal Server Error**
	Content:
	{
	  "message": "Error fetching product",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.get(
	  Uri.parse('http://localhost:8002/api/invoices'),
	  headers: {
	    'Content-Type': 'application/json',
	    'Authorization': 'Bearer <jwt_token>',
	  },
	);

Notes	<none>

12. Get Invoice Details by Invoice ID

This API retrieves details of a specific invoice by its ID. It also calculates the total price for each product based on quantity and unit_price.

URL	http://localhost:8002/api/invoices/details/:invoice_id
Method	GET
URL Params	invoice_id = Invoice ID (required)
Header	Content-Type: application/json Authorization: Bearer <jwt_token>
Data Params	{ "id": <product_id>, "quantity": <quantity_to_decrease> }
Success Response	Code: 200 OK Content: [{ "product_name": "Product A", "product_code": "A001", "quantity": 2, "unit_price": 50.0, "total_price": 100.0 }]
Error Response	**Code: 404 Not Found** **Content:** { "message": "Không tìm thấy chi tiết hóa đơn cho mã hóa đơn này" } **Code: 500 Internal Server Error** **Content:** { "message": "Lỗi khi lấy chi tiết hóa đơn", "error": "<error_details>" }
Sample Call	final response = await http.get(Uri.parse('http://localhost:8002/api/invoices/details/12345'), // Example invoice_id headers: { 'Content-Type': 'application/json', 'Authorization': 'Bearer <jwt_token>', },);
Notes	<none>

13. Get Invoice by User ID

This API allows fetching all invoices for a specific user by their ID, including grouped invoice details.

URL	http://localhost:8002/api/invoices/user/:id
Method	GET
URL Params	id = User ID (required)
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	  "id": <product_id>,
	  "quantity": <quantity_to_decrease>
	}

Success Response	Code: 200 OK
	Content:
	[
	  {
	    "invoice_id": 1,
	    "user_id": 101,
	    "total_amount": 100.0,
	    "created_at": "2025-01-14 12:00:00",
	    "updated_at": "2025-01-14 12:00:00",
	    "details": [
	      {
	        "product_name": "Product A",
	        "product_code": "A001",
	        "quantity": 2,
	        "unit_price": 50.0,
	        "total_price": 100.0
	      }
	    ]
	  }
	]

Error Response	**Code: 404 Not Found**
	Content:
	{
	  "message": "Không tìm thấy hóa đơn cho mã hóa đơn này"
	}
	Code: 500 Internal Server Error
	Content:
	{
	  "message": "Lỗi khi lấy hóa đơn",
	  "error": "<error_details>"
	}

Sample Call	final response = await http.get(
	  Uri.parse('http://localhost:8002/api/invoices/user/101'), // Example user_id
	  headers: {

	'Content-Type': 'application/json', 'Authorization': 'Bearer <jwt_token>', },);
Notes	<none>

14. Create New Invoice

This API creates a new invoice for a user. It requires a user ID, total amount, and invoice details to be provided.

URL	http://localhost:8002/api/invoices
Method	POST
URL Params	<none>
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	"user_id": 101,
	"total_amount": 100.0,
	"details": [
	{
	"product_name": "Product A",
	"product_code": "A001",
	"quantity": 2,
	"unit_price": 50.0
	}
	]
	}

Success Response	Code: 200 Created
	Content:
	{
	"message": "Hóa đơn đã được tạo và chi tiết đã được thêm",
	"invoiceId": 1
	}

Error Response	**Code: 404 Bad Request**
	Content:
	{
	"message": "Chi tiết hóa đơn không hợp lệ"
	}
	Code: 500 Internal Server Error
	Content:
	{
	"message": "Lỗi khi tạo hóa đơn",

	"error": "<error_details>" }
Sample Call	
Notes	<none>

15. Update Invoice

This API allows updating an existing invoice based on its ID.

URL	http://localhost:8002/api/invoices/:id
Method	PUT
URL Params	id = Invoice ID (required)
Header	Content-Type: application/json Authorization: Bearer <jwt_token>

Data Params	{
	  "total_amount": 120.0,
	  "details": [
	    {
	      "product_name": "Product B",
	      "product_code": "B001",
	      "quantity": 3,
	      "unit_price": 40.0
	    }
	  ]
	}

Success Response	Code: 200 OK
	Content:
	{
	  "message": "Hóa đơn đã được cập nhật"
	}

Error Response	**Code: 500 Internal Server Error**
	Content:
	{
	"message": "Lỗi khi cập nhật hóa đơn",
	   "error": "<error_details>"
	}

Sample Call	final response = await http.post(
	  Uri.parse('http://localhost:8002/api/invoices'),
	  headers: {
	    'Content-Type': 'application/json',

	'Authorization': 'Bearer <jwt_token>',
	},
	body: json.encode({
	'user_id': 101,
	'total_amount': 100.0,
	'details': [
	{
	'product_name': 'Product A',
	'product_code': 'A001',
	'quantity': 2,
	'unit_price': 50.0
	}
	]
	}),
	);

Notes	<none>

16. Delete Invoice

This API deletes an invoice based on its ID.

URL	http://localhost:8002/api/invoices/:id
Method	DELETE
URL Params	id = Invoice ID (required)
Header	Content-Type: application/json Authorization: Bearer <jwt_token>
Data Params	<none>

Success Response	Code: 200 OK
	Content:
	{
	"message": "Hóa đơn đã được xóa"
	}

Error Response	**Code: 500 Internal Server Error**
	Content:
	{
	"message": "Lỗi khi xóa hóa đơn",
	"error": "<error_details>"
	}

Sample Call	final response = await http.delete(Uri.parse('http://localhost:8002/api/invoices/1'), headers: { 'Content-Type': 'application/json',

	```
	    'Authorization': 'Bearer <jwt_token>',
	  },
	);
	```
Notes	<none>