Bukidnon State University
 Malaybalay City

College of Technologies
Information Technology Department
Bachelor of Science in Information Technology
$1^{\text {st }}$ Semester SY: 2025 - 2026
IT137 - Integrative Programming and Technologies 2

Sparklean Laundry POS & Management System API Documentation

I. Overview

1.1 Project Name

Sparklean Laundry POS & Management System

1.2 Client / Respondents of the System

LaundryPOS Franchise Operations Board

1.3 Description

The Sparklean Laundry POS & Management System is a comprehensive, web-based platform designed to help franchise operators manage daily laundry orders and branch payments. This all-inclusive solution strives to reduce human error while offering a user-friendly interface for handling laundry services, pricing, and branch-level transactions. The system guarantees accurate recording and arrangement of operational data pertaining to every branch by providing an intuitive interface. The platform's capacity to generate digital receipts and invoices is a crucial feature that improves transparency by giving branch managers instant proof of successful payments.

The system incorporates distinct user roles with specific permissions to maintain data integrity and security. Admins are authorized to configure branches, services, employees, and backups, while staff are granted access to create orders, submit expenses, and update customer information. In addition, the franchise owner monitors customers in the system while the platform administrator has the responsibility of maintaining audit logs and backups. This role-based structure ensures that each user can perform designated tasks effectively while maintaining appropriate access controls. Automated notifications further streamline the workflow, reducing manual administrative tasks and providing customers with instant proof of order status. It also generates comprehensive reports on service performance and cashflow for the organization.

The Sparklean Laundry POS & Management System is designed to improve the overall efficiency of laundry business activities. It offers a comprehensive solution for handling customer orders, classifying various services, keeping an up-to-date database of employees, and communicating payment status clearly via automated receipts. By automating several areas of branch operations, the system considerably minimizes the possibility of human error, which improves record accuracy and increases customer satisfaction. This specialized approach allows branch teams to focus on better servicing clients while ensuring a dependable, user-friendly financial tracking system that keeps every member informed and up to date.

1.4 Key Features

- **Secure Login**: Utilizes email/username + password with JWT session tokens so only authorized users can log in.
- **Error Handling**: Implements consistent error objects for validation issues, missing permissions, and expired tokens to enhance troubleshooting.
- **Order Management**: Allows branches to create drafts, finalize orders, apply discounts, and collect partial payments directly from the Admin or Staff apps.
- **Payment Reporting**: Generates per-branch reports, sales dashboards, and exportable summaries for finance reviews.
- **Receipt Management**: Facilitates viewing and emailing of invoices or receipts, with optional PDF attachments for transparency.
- **Calendar & Notifications**: Integrates pickup schedules, branch announcements, and alert banners so staff never miss deadlines.
- **Email Dispatch**: Sends automated notifications to customers and franchise owners covering invoices, payment reminders, and approvals.
- **Cloud Backups**: Provides scheduled and on-demand backups (files and databases) with retention policies managed in the backend.
- **Role-Based Access**: Enforces RBAC so admins, staff, and auditors only see features appropriate to their roles.
- **Archive, Unarchive Records, and Manage Accounts**: Archives orders, customers, and employees instead of deleting them to preserve history.
- **Secure Logout**: Ensures stateless JWT sessions are closed and activity logged after each logout.

1.5 Version

1.6

1.6 Base URL

http://localhost:5000/api

1.7 Authentication

The Sparklean Laundry POS & Management System uses JWT (JSON Web Token) for secure user authentication. Users supply either an email or username plus password to `/auth/login`, the credentials are validated against the MongoDB user store, and a signed token is returned. Optional safeguards—such as email OTP verification and rate limiting—can be toggled per environment. Password-reset OTPs are delivered through the `/auth/forgot-password` and `/auth/reset-password` endpoints, and all sensitive changes are written to the audit log.

II. Endpoints

The following are the detailed endpoints of the Sparklean Laundry POS & Management System API. These include HTTP methods used in the API call, parameters, configuration, request, and response of the API.

Authentication

Authentication Method

All protected endpoints require JWT authentication via the `Authorization` header:

```
Authorization: Bearer <jwt_token>
```

Token Expiration

- Default: 7 days (configurable via `JWT_EXPIRE` environment variable)

Getting a Token

1. Register a new account: `POST /api/auth/register`
2. Login: `POST /api/auth/login`
3. Receive token in response

API Base URL

- **Development HTTP**: `http://localhost:5000`
- **Development HTTPS**: `https://localhost:5443`
- **Production**: Configured via environment variables (`ALLOWED_ORIGINS`)

Response Format

Success Response

```
{
  "success": true,
  "data": { ... },
  "count": 10  // Optional, for list endpoints
}
```

Error Response

```
{
  "success": false,
  "message": "Error description",
  "error": "Detailed error (development only)"
}
```

III. Authentication Requirements

1. Every protected endpoint must include the `Authorization: Bearer <token>` header.
2. JWT tokens expire after seven days by default; enabling `rememberMe` extends the session to thirty days.
3. Accounts are locked for two hours after five failed login attempts.
4. OTP codes expire after ten minutes and are limited to three requests per hour.
5. `POST /auth/logout` records the logout event; disabling the user account is required to fully revoke stateless tokens.

IV. Error Handling

Status	Meaning	Typical Cause
200	OK	Request processed successfully
201	Created	Resource generated (user, OTP, reset entry)
400	Bad Request	Missing required fields or malformed JSON
401	Unauthorized	Missing/invalid token or incorrect credentials
403	Forbidden	Role lacks permission to access the resource
404	Not Found	Resource not available (profile/email mismatch)
409	Conflict	Duplicate user, locked account, or reused OTP
422	Unprocessable Entity	OTP expired or password policy violation
429	Too Many Requests	Rate limit triggered for login/OTP
500	Internal Server Error	Unexpected error (database outage, service failure)

Modules & Endpoints

2.1 AUTHENTICATION AND GENERAL MODULE API

2.1.1 Module Description

The LaundryPOS authentication module backs both the Admin and Staff apps, covering login, session inspection, password changes, OTP-based resets, and logout. Every sensitive action flows through the JWT middleware (server/middleware/auth.js) and RBAC guard (server/middleware/rbac.js). The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/auth/login
http://localhost:5000/api/auth/register
http://localhost:5000/api/auth/forgot-password
http://localhost:5000/api/auth/verify-reset-code
http://localhost:5000/api/auth/reset-password
http://localhost:5000/api/auth/me
http://localhost:5000/api/auth/me (PUT - Update Profile)
http://localhost:5000/api/auth/change-password
http://localhost:5000/api/auth/logout
http://localhost:5000/api/auth/users
http://localhost:5000/api/auth/profile/:userId
http://localhost:5000/api/auth/send-verification-code
http://localhost:5000/api/auth/verify-email-code
http://localhost:5000/api/auth/deactivate/:userId
http://localhost:5000/api/auth/activate/:userId

Response Codes of This API

Code	Message	Description
500	Internal Server Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
429	Too Many Requests	The client sends too many requests within a certain period.
404	Not Found	The requested resource could not be found.
403	Forbidden	Invalid token or token expired.
401	Unauthorized	The request was not successful because it lacks valid authentication credentials.
400	Bad Request	The request was invalid.
201	Created	The request was successful, and a new resource has been created.
200	OK	The request succeeded, and the resource is in the message body.

2.1.1.1 Authenticate User and Create Session

Version: 1.6

Date: November 26, 2025

Description: This API endpoint enables the creation of login sessions for secure access to LaundryPOS. Error responses are generated for invalid login attempts, including incorrect credentials or missing parameters. This ensures that only registered users with valid login details can access the system.

Endpoint: http://localhost:5000/api/auth/login

Method: POST

Configurations:

- The API request requires the Content-Type: application/json header.
- When reCAPTCHA is enabled, the request must include a valid token in the body.
- Successful logins store device and IP metadata in the audit log.

Parameters:

{{content}}gt;$ _email - required if _username is not provided; specifies the user's email.

{{content}}gt;$ _username - required if _email is not provided; specifies the user's username.

{{content}}gt;$ _password - required; user's password.

{{content}}gt;$ _recaptchaToken - optional; token submitted by the user when reCAPTCHA is enabled.

{{content}}gt;$ _rememberMe - optional; indicates whether the session should remain active longer.

{{content}}gt;$ _errorMessage - optional; returned in the response body to provide details about any login issues, such as "Invalid username or password" or "reCAPTCHA verification failed."

Requests:

Valid Request

```
{
    "email": "user@example.com",
    "password": "SecurePassword123!",
    "recaptchaToken": "recaptcha_token_here"
}
```

Not Valid Request

```
{
    "email": "user@example.com",
    "password": "wrongpassword"
}
```

Response Format: JSON

Responses:

Success Response

```
{
    "success": true,
    "data": {
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
        "user": {
            "_id": "user_id",
            "email": "user@example.com",
            "username": "johndoe",
            "role": "staff",
            "fullName": "John Doe",
            "isActive": true,
            "stationId": "station_id"
        }
    }
}
```

Error Response

```
{
    "success": false,
    "message": "Invalid credentials"
}
```

The API returns a 200 OK status for successful login attempts, confirming access to the LaundryPOS system. The response contains the user's username, email, role, and ID within the organization. Failed reCAPTCHA responses return a 400 Bad Request status code with the message "reCAPTCHA verification failed." Additionally, unauthorized access due to missing or invalid authentication credentials triggers a 401 Unauthorized status code with the message "Invalid email or password," ensuring that secure access policies are enforced.

2.1.1.2 Register User

Version: 1.6

Date: November 26, 2025

Description: Bootstrap a user account; typically used by admins only. This endpoint allows administrators to create new user accounts in the system.

Endpoint: http://localhost:5000/api/auth/register

Method: POST

Configurations:

- The API request requires the Content-Type: application/json header.
- The request must include valid user information in the body.

Parameters:

{{content}}gt;$ _email - required; specifies the user's email address.

{{content}}gt;$ _username - required; specifies the user's username.

{{content}}gt;$ _password - required; user's password (must meet security requirements).

{{content}}gt;$ _role - optional; defines the user's role within the system (defaults to "staff").

Requests:

Valid Request

```
{
    "email": "user@example.com",
    "username": "johndoe",
    "password": "SecurePassword123!",
    "role": "staff"
}
```

Not Valid Request

```
{
    "email": "invalid-email",
    "username": "",
    "password": "123"
}
```

Response Format: JSON

Responses:

Success Response

```
{
    "success": true,
    "data": {
        "token": "jwt_token_here",
        "user": {
            "_id": "user_id",
            "email": "user@example.com",
            "username": "johndoe",
            "role": "staff"
        }
    }
}
```

Error Response

```
{
    "success": false,
    "message": "User already exists"
}
```

The API returns a 201 Created status for successful user registration, confirming the creation of a new user account in the LaundryPOS system. The response contains the user's information and a JWT token for immediate authentication. A 400 Bad Request status indicates invalid input, such as missing required fields or invalid email format. A 409 Conflict status is returned if the user already exists in the system.

2.1.1.3 Forgot Password

Version: 1.6

Date: November 26, 2025

Description: Send OTP to email for password reset. This endpoint initiates the password recovery process by sending a verification code to the user's registered email address.

Endpoint: http://localhost:5000/api/auth/forgot-password

Method: POST

Configurations:

- The API request requires the Content-Type: application/json header.
- Rate limiting is applied to prevent abuse (maximum 3 requests per hour per email).

Parameters:

{{content}}gt;$ _email - required; specifies the user's email address for password reset.

Requests:

Valid Request

```
{
    "email": "user@example.com"
}
```

Not Valid Request

```
{
    "email": "nonexistent@example.com"
}
```

Response Format: JSON

Responses:

Success Response

```
{
    "success": true,
    "message": "Password reset code sent to email"
}
```

Error Response

```
{
    "success": false,
    "message": "User not found"
}
```

The API returns a 200 OK status for successful password reset code delivery, confirming that the OTP has been sent to the user's email. A 404 Not Found status indicates that the email address is not registered in the system. A 429 Too Many Requests status is returned if the rate limit has been exceeded.

2.1.1.4 Verify Reset Code

Version: 1.6

Date: November 26, 2025

Description: Validate a previously issued password reset OTP code before allowing the user to set a new password. This endpoint ensures that only the correct and non-expired codes can be used in the reset workflow.

Endpoint: http://localhost:5000/api/auth/verify-reset-code

Method: POST

Configurations:

- The API request requires the Content-Type: application/json header.
- Rate limiting shares the same limits as `/auth/forgot-password`.

Parameters:

{{content}}gt;$ _email - required; email address that received the OTP.

{{content}}gt;$ _code - required; numeric or alphanumeric OTP code sent to the user.

Requests:

Valid Request

```
{
    "email": "user@example.com",
    "code": "123456"
}
```

Not Valid Request

```
{
    "email": "user@example.com",
    "code": "000000"
}
```

Response Format: JSON

Responses:

Success Response

```
{
    "success": true,
    "message": "Code verified"
}
```

Error Response

```
{
    "success": false,
    "message": "Invalid or expired OTP code"
}
```

The API returns a 200 OK status when the OTP is valid and not expired. A 400 Bad Request status indicates an invalid or expired OTP code, while 404 Not Found is returned when the email is not associated with any reset attempt.

2.1.1.5 Reset Password

Version: 1.6

Date: November 26, 2025

Description: Replace password using a valid OTP. This endpoint allows users to reset their password after verifying the OTP code sent to their email.

Endpoint: http://localhost:5000/api/auth/reset-password

Method: POST

Configurations:

- The API request requires the Content-Type: application/json header.
- OTP codes expire after 10 minutes from generation.

Parameters:

{{content}}gt;$ _email - required; specifies the user's email address.

{{content}}gt;$ _code - required; the OTP verification code sent to the user's email.

{{content}}gt;$ _newPassword - required; the new password that meets security requirements.

Requests:

Valid Request

```
{
    "email": "user@example.com",
    "code": "123456",
    "newPassword": "NewSecurePassword123!"
}
```

Not Valid Request

```
{
    "email": "user@example.com",
    "code": "000000",
    "newPassword": "123"
}
```

Response Format: JSON

Responses:

Success Response

```
{
    "success": true,
    "message": "Password reset successfully"
}
```

Error Response

```
{
    "success": false,
    "message": "Invalid or expired OTP code"
}
```

The API returns a 200 OK status for successful password reset, confirming that the user's password has been updated. A 400 Bad Request status indicates invalid input, such as an expired or incorrect OTP code. A 422 Unprocessable Entity status is returned if the new password does not meet security requirements.

2.1.1.6 Get Current Authenticated User

Version: 1.6

Date: November 26, 2025

Description: Retrieve the authenticated user profile associated with the provided JWT token. This endpoint is used by both Admin and Staff apps during startup to hydrate the current session.

Endpoint: http://localhost:5000/api/auth/me

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The token must be signed with the configured `JWT_SECRET`.

Parameters:

{{content}}gt;$ _token - required; JWT access token supplied in the Authorization header.

Requests:

Valid Request

```
GET http://localhost:5000/api/auth/me
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/auth/me
(Missing Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "user_id",
    "email": "user@example.com",
    "username": "johndoe",
    "role": "staff",
    "fullName": "John Doe",
    "isActive": true,
    "stationId": "station_id"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns **200 OK** when the token is valid and the user account is active. A **401 Unauthorized** status indicates a missing or invalid token, while **403 Forbidden** is returned when the account has been deactivated.

2.1.1.7 Update Profile

Version: 1.6

Date: November 26, 2025

Description: Update basic profile details for the currently authenticated user, such as email, username, or last known location. This endpoint is used by both web and mobile clients to keep user metadata in sync.

Endpoint: http://localhost:5000/api/auth/me

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the `Content-Type: application/json` header.
- Partial updates are supported; unspecified fields are left unchanged.

Parameters:

{{content}}gt;$ _token - required; JWT access token supplied in the Authorization header.

{{content}}gt;$ _email - optional; new email address.

{{content}}gt;$ _username - optional; new username.

{{content}}gt;$ _location - optional; object containing latitude and longitude for mobile devices.

Requests:

Valid Request

```
{
  "email": "newemail@example.com",
  "username": "newusername",
  "location": {
    "latitude": 14.5995,
    "longitude": 120.9842
  }
}
```

Not Valid Request

```
{
  "email": "not-an-email"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "user_id",
    "email": "newemail@example.com",
    "username": "newusername"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Validation error: invalid email format"
}
```

The API returns **200 OK** when profile information is successfully updated. Invalid payloads result in **400 Bad Request**, while calls without a valid token receive **401 Unauthorized**.

2.1.1.8 Change Password

Version: 1.6

Date: November 26, 2025

Description: Allow an authenticated user to change their password by supplying their current password and a new one. This endpoint is intended for in-session password changes and complements the OTP-based reset flow.

Endpoint: http://localhost:5000/api/auth/change-password

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the `Content-Type: application/json` header.
- New passwords must comply with the configured password policy.

Parameters:

{{content}}gt;$ _currentPassword - required; the user's existing password.

{{content}}gt;$ _newPassword - required; the new password that meets security requirements.

Requests:

Valid Request

```
{
  "currentPassword": "OldPassword123!",
  "newPassword": "NewPassword123!"
}
```

Not Valid Request

```
{
  "currentPassword": "WrongOldPassword",
  "newPassword": "123"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Password changed successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Current password is incorrect"
}
```

Successful changes return **200 OK**, while invalid current passwords result in **401 Unauthorized** and weak new passwords are rejected with **422 Unprocessable Entity**.

2.1.1.9 Logout

Version: 1.6

Date: November 26, 2025

Description: Record a logout event for the current user to support audit logging and optional token blacklisting. Although JWTs are stateless, this endpoint helps track session terminations.

Endpoint: http://localhost:5000/api/auth/logout

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Can be called safely multiple times; redundant logouts are ignored gracefully.

Parameters:

{{content}}gt;$ _token - required; JWT access token supplied in the Authorization header.

Requests:

Valid Request

```
POST http://localhost:5000/api/auth/logout
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
POST http://localhost:5000/api/auth/logout
(Missing Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Logged out successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The endpoint returns **200 OK** when the logout event is stored. Requests without valid credentials receive **401 Unauthorized**.

2.1.1.10 Get All Users (Admin Only)

Version: 1.6

Date: November 26, 2025

Description: Retrieve a paginated list of all user accounts in the system for administrative review. This endpoint is restricted to admin roles and supports optional filtering by role or status.

Endpoint: http://localhost:5000/api/auth/users

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Caller must have the `admin` role and appropriate RBAC permission (`auth:admin`).

Parameters:

{{content}}gt;$ _token - required; admin JWT supplied in the Authorization header.

{{content}}gt;$ _role - optional; filter by role (e.g., `admin`, `staff`).

{{content}}gt;$ _status - optional; filter by active or deactivated accounts.

Requests:

Valid Request

```
GET http://localhost:5000/api/auth/users?role=staff
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/auth/users
(Missing or non-admin token)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "_id": "user_id",
      "email": "user@example.com",
      "username": "johndoe",
      "role": "staff",
      "isActive": true
    }
  ],
  "count": 10
}
```

Error Response

```
{
  "success": false,
  "message": "Forbidden: admin access required"
}
```

The endpoint returns **200 OK** with a list of users for valid admin calls. Missing or non-admin tokens result in **403 Forbidden**, while unauthenticated requests receive **401 Unauthorized**.

2.1.1.11 Get User Profile by ID (Admin Only)

Version: 1.6

Date: November 26, 2025

Description: Fetch a specific user profile by its identifier for administrative review or troubleshooting. This endpoint allows admins to inspect any account's details.

Endpoint: http://localhost:5000/api/auth/profile/:userId

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Caller must have the `admin` role and `auth:admin` permission.

Parameters:

{{content}}gt;$ _userId - required; path parameter identifying the user to fetch.

{{content}}gt;$ _token - required; admin JWT in the Authorization header.

Requests:

Valid Request

```
GET http://localhost:5000/api/auth/profile/6790a2c83f1
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/auth/profile/invalid-id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "user_id",
    "email": "user@example.com",
    "username": "johndoe",
    "role": "staff",
    "fullName": "John Doe"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "User not found"
}
```

Successful lookups return **200 OK** with the user profile. Invalid or non-existent IDs result in **404 Not Found**, while missing permissions can trigger **403 Forbidden**.

2.1.1.12 Deactivate User (Admin Only)

Version: 1.6

Date: November 26, 2025

Description: Disable a user account so that the user can no longer log in. Deactivation is soft and reversible using the activation endpoint.

Endpoint: http://localhost:5000/api/auth/deactivate/:userId

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.

- Caller must have the `admin` role and `auth:admin` permission.

Parameters:

{{content}}gt;$ _userId - required; path parameter of the account to deactivate.

{{content}}gt;$ _token - required; admin JWT in the Authorization header.

Requests:

Valid Request

```
PUT http://localhost:5000/api/auth/deactivate/6790a2c83f1
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/auth/deactivate/invalid-id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "User deactivated successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "User not found"
}
```

The endpoint responds with **200 OK** when the account status is updated. Invalid IDs result in **404 Not Found**, and unauthorized calls receive **403 Forbidden** or **401 Unauthorized**.

2.1.1.13 Activate User (Admin Only)

Version: 1.6

Date: November 26, 2025

Description: Reactivate a previously deactivated user account so that the user can log in again. This is the inverse of the deactivate operation.

Endpoint: http://localhost:5000/api/auth/activate/:userId

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Caller must have the `admin` role and `auth:admin` permission.

Parameters:

{{content}}gt;$ _userId - required; path parameter of the account to activate.

{{content}}gt;$ _token - required; admin JWT in the Authorization header.

Requests:

Valid Request

```
PUT http://localhost:5000/api/auth/activate/6790a2c83f1
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/auth/activate/invalid-id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "User activated successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "User not found"
}
```

The endpoint returns **200 OK** when the user is successfully reactivated. Attempting to activate a non-existent account results in **404 Not Found**, while missing permissions lead to **403 Forbidden**.

2.1.1.14 Send Email Verification Code

Version: 1.6

Date: November 26, 2025

Description: Send a verification OTP to the specified email address to confirm ownership during account registration or email change.

Endpoint: http://localhost:5000/api/auth/send-verification-code

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the `Content-Type: application/json` header.
- Rate limiting is applied per email to prevent abuse.

Parameters:

{{content}}gt;$ _email - required; email address to receive the verification code.

Requests:

Valid Request

```
{
  "email": "user@example.com"
}
```

Not Valid Request

```
{
  "email": "not-an-email"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Verification code sent to email"
}
```

Error Response

```
{
  "success": false,
  "message": "User not found or email invalid"
}
```

The endpoint returns **200 OK** when the verification code is queued or sent successfully. Invalid email formats result in **400 Bad Request**, and excessive requests trigger **429 Too Many Requests**.

2.1.1.15 Verify Email Code

Version: 1.6

Date: November 26, 2025

Description: Verify the email verification OTP code previously sent to the user. Once verified, the account or email address is marked as confirmed.

Endpoint: http://localhost:5000/api/auth/verify-email-code

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the `Content-Type: application/json` header.

Parameters:

{{content}}gt;$ _email - required; email address associated with the verification code.

{{content}}gt;$ _code - required; OTP code that was sent to the user.

Requests:

Valid Request

```
{
  "email": "user@example.com",
  "code": "123456"
}
```

Not Valid Request

```
{
  "email": "user@example.com",
  "code": "000000"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Email verified successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Invalid or expired verification code"
}
```

The endpoint returns **200 OK** when the code is valid and the email is successfully verified. Incorrect or expired codes result in **400 Bad Request**, while unrecognized emails return **404 Not Found**.

2.2 ORDERS MODULE API

2.2.1 Module Description

The Orders module manages all order-related operations including creation, updates, archiving, draft management, and invoice generation. All endpoints require authentication and appropriate RBAC permissions. Staff users are automatically restricted to orders from their assigned station. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints (aligned with the Orders module sections and Postman collection):
http://localhost:5000/api/orders
http://localhost:5000/api/orders/:id
http://localhost:5000/api/orders/draft
http://localhost:5000/api/orders/:id/archive
http://localhost:5000/api/orders/:id/unarchive
http://localhost:5000/api/orders/:id/mark-completed
http://localhost:5000/api/orders/:id/schedule-deletion
http://localhost:5000/api/orders/:id/send-email
http://localhost:5000/api/orders/:id/lock

2.2.1.1 Get All Orders

Version: 1.6

Date: November 26, 2025

Description: Get all orders with filtering and search capabilities. Staff users can only see orders from their assigned station. This endpoint enables retrieval of order lists with various filtering options to help manage and track laundry orders efficiently.

Endpoint: http://localhost:5000/api/orders

Method: GET

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Staff users are automatically filtered to show only orders from their assigned station.
- Results are sorted by date in descending order (newest first).

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _search - optional; search term to filter orders by order ID or customer name.

{{content}}gt;$ _payment - optional; filter by payment status: "Paid", "Unpaid", "Partial", or "All" (default: "All").

{{content}}gt;$ _showArchived - optional; boolean flag to include archived orders (default: false).

{{content}}gt;$ _showDrafts - optional; boolean flag to include draft orders (default: false).

Requests:

Valid Request

```
GET http://localhost:5000/api/orders?search=ORD-001&payment=Paid&showArchived=false
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/orders
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": [
    {
      "_id": "order_id",
      "id": "ORD-001",
      "date": "2024-01-15T10:30:00.000Z",
      "customer": "John Doe",
      "customerPhone": "+1234567890",
      "customerId": "customer_id",
      "payment": "Paid",
      "total": "₱1,500.00",
      "paid": 1500,
      "balance": "₱0.00",
      "items": [
        {
          "service": "Wash & Fold",
          "quantity": "5kg",
          "status": "Completed",
          "amount": 500
        }
      ],
      "isDraft": false,
      "isArchived": false,
      "createdBy": {
        "_id": "user_id",
        "username": "staff1",
        "email": "staff1@example.com"
      }
    }
  ],
  "count": 1
}
```


2.2.1.2 Get Single Order

Version: 1.6

Date: November 26, 2025

Description: Retrieve a single order by its unique order ID, including customer details, payment information, and item breakdown. This endpoint is used by the Admin and Staff apps when viewing order details.

Endpoint: http://localhost:5000/api/orders/:id

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Honors station scoping so staff can only access orders from their assigned station.

Parameters:

{{content}}gt;$ _token – required; JWT supplied in the Authorization header.

{{content}}gt;$ _id – required; path parameter representing the human-readable order ID (e.g., `ORD-001`).

Requests:

Valid Request

```
GET http://localhost:5000/api/orders/ORD-001
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/orders/ORD-99999
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": {
    "_id": "order_id",
    "id": "ORD-001",
    "date": "2024-01-15T10:30:00.000Z",
    "customer": "John Doe",
    "customerPhone": "+1234567890",
    "customerId": {
      "_id": "customer_id",
      "name": "John Doe",
      "email": "john@example.com",
      "phone": "+1234567890"
    },
    "payment": "Paid",
    "total": "₱1,500.00",
    "paid": 1500,
    "balance": "₱0.00",
    "items": [],
    "discount": "10%",
    "discountId": "discount_id",
    "notes": "Handle with care",
    "pickupDate": "2024-01-16T10:00:00.000Z",
    "deliveryDate": null,
    "createdBy": {
      "_id": "user_id",
      "username": "staff1",
      "email": "staff1@example.com"
    },
    "lastEditedBy": null
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Order not found"
}
```


The API returns **200 OK** when the order with the specified ID exists and is visible to the caller. A request with a non-existent ID returns **404 Not Found**, while missing or invalid tokens are rejected with **401 Unauthorized**.

2.2.1.3 Create Order

Version: 1.6

Date: November 26, 2025

Description: Create a new laundry order with items, payment status, and optional customer linkage. This endpoint powers the main "New Order" workflow on the Admin and Staff apps.

Endpoint: http://localhost:5000/api/orders

Method: POST

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- The request must include the Content-Type: application/json header.
- Requires the `orders:create` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _customer - required; customer name for the order.

{{content}}gt;$ _customerPhone - required; customer phone number.

{{content}}gt;$ _customerId - optional; existing customer ID if customer already exists in the system.

{{content}}gt;$ _payment - required; payment status: "Paid", "Unpaid", or "Partial".

{{content}}gt;$ _items - required; array of order items, each containing service, quantity, status, and amount.

{{content}}gt;$ _total - required; total order amount as a formatted string (e.g., "₱1,500.00").

{{content}}gt;$ _paid - optional; amount paid (required if payment is "Partial").

{{content}}gt;$ _balance - optional; remaining balance.

{{content}}gt;$ _discount - optional; discount percentage or amount.

{{content}}gt;$ _discountId - optional; discount ID if a discount is applied.

{{content}}gt;$ _notes - optional; additional notes or instructions.

{{content}}gt;$ _pickupDate - optional; scheduled pickup date and time.

{{content}}gt;$ _deliveryDate - optional; scheduled delivery date and time.

{{content}}gt;$ _stationId - optional; station ID (defaults to user's assigned station).

Requests:

Valid Request

```
{
  "customer": "John Doe",
  "customerPhone": "+1234567890",
  "customerId": "customer_id",
  "payment": "Unpaid",
  "items": [
    {
      "service": "Wash & Fold",
      "quantity": "5kg",
      "status": "Pending",
      "amount": 500
    }
  ],
  "total": "₱1,500.00",
  "paid": 0,
  "balance": "₱1,500.00",
  "discount": "0%",
  "discountId": null,
  "notes": "Handle with care",
  "pickupDate": "2024-01-16T10:00:00.000Z",
  "deliveryDate": null,
  "stationId": "station_id"
}
```


Not Valid Request


```
{
  "customer": "",
  "customerPhone": "",
  "payment": "InvalidStatus",
  "items": []
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "order_id",
    "id": "ORD-001",
    "date": "2024-01-15T10:30:00.000Z",
    "customer": "John Doe",
    "customerPhone": "+1234567890",
    "customerId": "customer_id",
    "payment": "Unpaid",
    "total": "₱1,500.00",
    "paid": 0,
    "balance": "₱1,500.00",
    "items": [
      {
        "service": "Wash & Fold",
        "quantity": "5kg",
        "status": "Pending",
        "amount": 500
      }
    ],
    "isDraft": false,
    "isArchived": false,
    "createdBy": {
      "_id": "user_id",
      "username": "staff1",
      "email": "staff1@example.com"
    }
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Validation error: customer, customerPhone, and items are required"
}
```


The API returns a 201 Created status when the order is successfully created. A 400 Bad Request status indicates invalid input, such as missing required fields or invalid payment status. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.4 Save Draft Order

Version: 1.6

Date: November 26, 2025

Description: Save an in-progress order as a draft that can be completed later. This endpoint is typically called automatically by the Staff app's autosave mechanism or manually when staff needs to pause order creation.

Endpoint: http://localhost:5000/api/orders/draft

Method: POST

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- The request must include the Content-Type: application/json header.
- Requires the `orders:create` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _customer - optional; customer name for the draft order.

{{content}}gt;$ _customerPhone - optional; customer phone number.

{{content}}gt;$ _customerId - optional; existing customer ID if customer already exists in the system.

{{content}}gt;$ _payment - optional; payment status: "Paid", "Unpaid", or "Partial".

{{content}}gt;$ _items - optional; array of order items, each containing service, quantity, status, and amount.

{{content}}gt;$ _total - optional; total order amount as a formatted string.

{{content}}gt;$ _notes - optional; additional notes or instructions.

{{content}}gt;$ _stationId - optional; station ID (defaults to user's assigned station).

Requests:

Valid Request


```
{
  "customer": "John Doe",
  "customerPhone": "+1234567890",
  "payment": "Unpaid",
  "items": [
    {
      "service": "Wash & Fold",
      "quantity": "5kg",
      "status": "Pending",
      "amount": 500
    }
  ]
}
```


Not Valid Request


```
{
  "customer": "",
  "items": null
}
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": {
    "_id": "order_id",
    "id": "ORD-001",
    "date": "2024-01-15T10:30:00.000Z",
    "customer": "John Doe",
    "customerPhone": "+1234567890",
    "payment": "Unpaid",
    "items": [
      {
        "service": "Wash & Fold",
        "quantity": "5kg",
        "status": "Pending",
        "amount": 500
      }
    ],
    "isDraft": true,
    "isArchived": false,
    "createdBy": {
      "_id": "user_id",
      "username": "staff1",
      "email": "staff1@example.com"
    }
  }
}
```

Error Response


```
{
  "success": false,
  "message": "Authentication required"
}
```


The API returns a 201 Created status when the draft order is successfully saved. A 400 Bad Request status indicates invalid input. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.5 Update Order

Version: 1.6

Date: November 26, 2025

Description: Apply changes to an existing order, such as updating items, totals, payment status, or notes. Only users with update permissions may modify orders. Partial updates are supported.

Endpoint: http://localhost:5000/api/orders/:id

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- The request must include the Content-Type: application/json header.
- Requires the `orders:update` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

{{content}}gt;$ _customer - optional; updated customer name.

{{content}}gt;$ _customerPhone - optional; updated customer phone number.

{{content}}gt;$ _payment - optional; updated payment status: "Paid", "Unpaid", or "Partial".

{{content}}gt;$ _items - optional; updated array of order items.

{{content}}gt;$ _total - optional; updated total order amount.

{{content}}gt;$ _paid - optional; updated amount paid.

{{content}}gt;$ _balance - optional; updated remaining balance.

{{content}}gt;$ _notes - optional; updated notes or instructions.

{{content}}gt;$ _pickupDate - optional; updated scheduled pickup date and time.

{{content}}gt;$ _deliveryDate - optional; updated scheduled delivery date and time.

Requests:

Valid Request


```
{
  "notes": "Handle with care",
  "payment": "Partial",
  "paid": 500,
  "balance": "₱1,000.00"
}
```


Not Valid Request


```
{
  "payment": "InvalidStatus",
  "paid": -100
}
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": {
    "_id": "order_id",
    "id": "ORD-001",
    "date": "2024-01-15T10:30:00.000Z",
    "customer": "John Doe",
    "customerPhone": "+1234567890",
    "payment": "Partial",
    "total": "₱1,500.00",
    "paid": 500,
    "balance": "₱1,000.00",
    "notes": "Handle with care",
    "lastEditedBy": {
      "_id": "user_id",
      "username": "staff1",
      "email": "staff1@example.com"
    },
    "updatedAt": "2024-01-15T11:00:00.000Z"
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Order not found"
}
```


The API returns a 200 OK status when the order is successfully updated. A 404 Not Found status indicates the order ID does not exist. A 400 Bad Request status is returned for invalid input, while 401 Unauthorized indicates missing or invalid authentication credentials. A 403 Forbidden status is returned when the user lacks update permissions.

2.2.1.6 Archive Order

Version: 1.6

Date: November 26, 2025

Description: Soft-delete an order by marking it as archived so it is hidden from day-to-day views while preserved for reporting and audit purposes. Archived orders can be restored using the unarchive endpoint.

Endpoint: http://localhost:5000/api/orders/:id/archive

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `orders:archive` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/orders/ORD-001/archive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/orders/invalid-id/archive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Order archived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found"
}
```

The API returns a 200 OK status when the order is successfully archived. A 404 Not Found status indicates the order ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.7 Unarchive Order

Version: 1.6

Date: November 26, 2025

Description: Restore a previously archived order so that it becomes visible again in the active order lists. This endpoint reverses the archive operation and makes the order available for regular operations.

Endpoint: http://localhost:5000/api/orders/:id/unarchive

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `orders:unarchive` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/orders/ORD-001/unarchive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/orders/invalid-id/unarchive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Order unarchived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found or not archived"
}
```

The API returns a 200 OK status when the order is successfully unarchived. A 404 Not Found status indicates the order ID does not exist or the order is not archived. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.8 Mark Draft as Completed

Version: 1.6

Date: November 26, 2025

Description: Convert a saved draft order into a completed order once payment and details are finalized. This endpoint changes the order status from draft to completed, making it appear in regular order lists.

Endpoint: http://localhost:5000/api/orders/:id/mark-completed

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `orders:update` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/orders/ORD-001/mark-completed
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/orders/invalid-id/mark-completed
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Draft marked as completed",
  "data": {
    "_id": "order_id",
    "id": "ORD-001",
    "isDraft": false,
    "updatedAt": "2024-01-15T11:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found or not a draft"
}
```

The API returns a 200 OK status when the draft order is successfully marked as completed. A 404 Not Found status indicates the order ID does not exist or the order is not a draft. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.9 Schedule Draft Deletion

Version: 1.6

Date: November 26, 2025

Description: Schedule a draft order for automatic deletion after a retention window (e.g., 30 days) to keep the database clean. This endpoint marks the draft for cleanup by automated background processes.

Endpoint: http://localhost:5000/api/orders/:id/schedule-deletion

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `orders:update` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/orders/ORD-001/schedule-deletion
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/orders/invalid-id/schedule-deletion
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Draft scheduled for deletion"
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found or not a draft"
}
```

The API returns a 200 OK status when the draft order is successfully scheduled for deletion. A 404 Not Found status indicates the order ID does not exist or the order is not a draft. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.10 Delete Order

Version: 1.6

Date: November 26, 2025

Description: Permanently delete an order record from the system. This operation is destructive and should be used only for corrections approved by administrators. Deleted orders cannot be recovered.

Endpoint: http://localhost:5000/api/orders/:id

Method: DELETE

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `orders:delete` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
DELETE http://localhost:5000/api/orders/ORD-001
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
DELETE http://localhost:5000/api/orders/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Order deleted successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found"
}
```

The API returns a 200 OK status when the order is successfully deleted. A 404 Not Found status indicates the order ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 409 Conflict status may be returned if the order cannot be deleted due to dependencies.

2.2.1.11 Send Invoice Email

Version: 1.6

Date: November 26, 2025

Description: Send an order invoice to the customer's email address, optionally overriding the stored email. This endpoint triggers the email notification workflow and generates an invoice PDF attachment.

Endpoint: http://localhost:5000/api/orders/:id/send-email

Method: POST

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- The request must include the Content-Type: application/json header.
- Requires the `orders:read` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

{{content}}gt;$ _email - optional; email address to send the invoice to (uses customer email if not provided).

Requests:

Valid Request

```
{
  "email": "customer@example.com"
}
```

Not Valid Request

```
{
  "email": "not-an-email"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Invoice sent successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found or email sending failed"
}
```

The API returns a 200 OK status when the invoice email is successfully sent. A 404 Not Found status indicates the order ID does not exist. A 400 Bad Request status is returned for invalid email addresses. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.12 Acquire Edit Lock

Version: 1.6

Date: November 26, 2025

Description: Acquire an edit lock for a specific order so that only one user can modify it at a time, preventing conflicting updates. The lock is automatically released after a timeout period or can be manually released.

Endpoint: http://localhost:5000/api/orders/:id/lock

Method: POST

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `orders:update` permission.
- Locks automatically expire after a configured timeout period.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
POST http://localhost:5000/api/orders/ORD-001/lock
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
POST http://localhost:5000/api/orders/invalid-id/lock
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Edit lock acquired",
  "data": {
    "isLocked": true,
    "lockedBy": {
      "name": "John Doe",
      "email": "john@example.com"
    },
    "lockedAt": "2024-01-15T10:30:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found or already locked by another user"
}
```

The API returns a 200 OK status when the edit lock is successfully acquired. A 404 Not Found status indicates the order ID does not exist. A 409 Conflict status is returned if the order is already locked by another user. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.2.1.13 Release Edit Lock

Version: 1.6

Date: November 26, 2025

Description: Release a previously acquired edit lock so other users can edit the order again. This endpoint allows the lock owner or an administrator to manually release the lock before it expires.

Endpoint: http://localhost:5000/api/orders/:id/lock

Method: DELETE

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `orders:update` permission.
- Only the lock owner or an administrator can release the lock.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
DELETE http://localhost:5000/api/orders/ORD-001/lock
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
DELETE http://localhost:5000/api/orders/invalid-id/lock
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Edit lock released"
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found or lock not owned by current user"
}
```

The API returns a 200 OK status when the edit lock is successfully released. A 404 Not Found status indicates the order ID does not exist. A 403 Forbidden status is returned if the lock is owned by another user and the current user is not an administrator. A 401 Unauthorized status is returned for missing or invalid authentication credentials.

2.2.1.14 Check Edit Lock Status

Version: 1.6

Date: November 26, 2025

Description: Check whether an order is currently locked for editing and, if so, by which user and at what time. This endpoint is used to display lock status in the UI.

Endpoint: http://localhost:5000/api/orders/:id/lock

Method: GET

Configurations:

- The API request requires the Authorization: Bearer <token> header.

- Requires the `orders:read` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique order ID in the URL path.

Requests:

Valid Request

```
GET http://localhost:5000/api/orders/ORD-001/lock
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/orders/invalid-id/lock
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "isLocked": false,
    "lockedBy": null,
    "lockedAt": null
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Order not found"
}
```

The API returns a 200 OK status with the current lock status. A 404 Not Found status indicates the order ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

Required Permissions:

- `orders:read` - View orders
- `orders:create` - Create orders
- `orders:update` - Update orders
- `orders:delete` - Delete orders
- `orders:archive` - Archive orders
- `orders:unarchive` - Unarchive orders

2.3 CUSTOMERS MODULE API

2.3.1 Module Description

The Customers module manages all customer-related operations including creation, updates, archiving, and retrieval. All endpoints require authentication and appropriate RBAC permissions. Staff users are automatically restricted to customers from their assigned station. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/customers
http://localhost:5000/api/customers/:id
http://localhost:5000/api/customers/:id/archive
http://localhost:5000/api/customers/:id/unarchive
http://localhost:5000/api/customers/:id (DELETE)

2.3.1.1 Get All Customers

Version: 1.6

Date: November 26, 2025

Description: Retrieve a list of all customers with optional filtering by search term, sorting options, and archived status. Staff users are automatically restricted to customers from their assigned station. This endpoint enables efficient customer management and lookup.

Endpoint: http://localhost:5000/api/customers

Method: GET

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Staff users are automatically filtered to show only customers from their assigned station.
- Results can be sorted by name, order count, or total spent in ascending or descending order.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _search - optional; search term to filter customers by name, email, or phone number.

{{content}}gt;$ _sortBy - optional; sort option: "name-asc", "name-desc", "orders-asc", "orders-desc", "spent-asc", "spent-desc" (default: "name-asc").

{{content}}gt;$ _showArchived - optional; boolean flag to include archived customers (default: false).

Requests:

Valid Request

```
GET http://localhost:5000/api/customers?search=John&sortBy=name-asc&showArchived=false
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/customers
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "_id": "customer_id",
      "name": "John Doe",
      "email": "john@example.com",
      "phone": "+1234567890",
      "totalOrders": 5,
      "totalSpent": 7500,
      "lastOrder": "2024-01-15T10:30:00.000Z",
      "isArchived": false,
      "notes": "VIP customer",
      "stationId": "station_id"
    }
  ],
  "count": 1
}
```


Error Response


```
{
  "success": false,
  "message": "Authentication required"
}
```


The API returns a 200 OK status for successful retrieval of customer lists. A 401 Unauthorized status indicates missing or invalid authentication credentials, while 403 Forbidden is returned when the user lacks the required permissions to view customers.

2.3.1.2 Get Single Customer

Version: 1.6

Date: November 26, 2025

Description: Retrieve detailed information for a specific customer by their unique customer ID, including order history, total spent, and contact details. This endpoint is used when viewing or editing customer profiles.

Endpoint: http://localhost:5000/api/customers/:id

Method: GET

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Staff users can only access customers from their assigned station.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique customer ID in the URL path.

Requests:

Valid Request

```
GET http://localhost:5000/api/customers/customer_id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/customers/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": {
    "_id": "customer_id",
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "+1234567890",
    "totalOrders": 5,
    "totalSpent": 7500,
    "lastOrder": "2024-01-15T10:30:00.000Z",
    "isArchived": false,
    "notes": "VIP customer",
    "stationId": "station_id"
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Customer not found"
}
```


The API returns a 200 OK status when the customer is found and accessible. A 404 Not Found status indicates the customer ID does not exist or the user lacks permission to view it. A 401 Unauthorized status is returned for missing or invalid authentication credentials.

2.3.1.3 Create Customer

Version: 1.6

Date: November 26, 2025

Description: Create a new customer record in the system with name, contact information, and optional notes. This endpoint is used when registering new customers during order creation or customer management.

Endpoint: http://localhost:5000/api/customers

Method: POST

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- The request must include the Content-Type: application/json header.
- Requires the `customers:create` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _name - required; the customer's full name.

{{content}}gt;$ _phone - required; the customer's phone number.

{{content}}gt;$ _email - optional; the customer's email address.

{{content}}gt;$ _notes - optional; additional notes or comments about the customer.

{{content}}gt;$ _stationId - optional; station ID (defaults to user's assigned station).

Requests:

Valid Request


```
{
  "name": "John Doe",
  "phone": "+1234567890",
  "email": "john@example.com",
  "notes": "VIP customer"
}
```

Not Valid Request

```
{
  "name": "",
  "phone": ""
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "customer_id",
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "+1234567890",
    "notes": "VIP customer",
    "stationId": "station_id",
    "totalOrders": 0,
    "totalSpent": 0,
    "isArchived": false,
    "createdAt": "2024-01-15T10:30:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Validation error: name and phone are required"
}
```

The API returns a 201 Created status when the customer is successfully created. A 400 Bad Request status indicates invalid input, such as missing required fields or invalid phone format. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.3.1.4 Update Customer

Version: 1.6

Date: November 26, 2025

Description: Update existing customer information such as name, contact details, or notes. Partial updates are supported; only provided fields will be updated.

Endpoint: http://localhost:5000/api/customers/:id

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- The request must include the Content-Type: application/json header.
- Requires the `customers:update` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique customer ID in the URL path.

{{content}}gt;$ _name - optional; updated customer name.

{{content}}gt;$ _phone - optional; updated phone number.

{{content}}gt;$ _email - optional; updated email address.

{{content}}gt;$ _notes - optional; updated notes.

Requests:

Valid Request

```
{
  "name": "Jane Doe",
  "phone": "+1234567891",
  "email": "jane@example.com"
}
```

Not Valid Request

```
{
  "email": "not-an-email"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "customer_id",
    "name": "Jane Doe",
    "phone": "+1234567891",
    "email": "jane@example.com",
    "updatedAt": "2024-01-15T11:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Customer not found"
}
```

The API returns a 200 OK status when the customer information is successfully updated. A 404 Not Found status indicates the customer ID does not exist. A 400 Bad Request status is returned for invalid input, while 401 Unauthorized indicates missing or invalid authentication credentials.

2.3.1.5 Archive Customer

Version: 1.6

Date: November 26, 2025

Description: Soft-delete a customer by marking them as archived. Archived customers are hidden from regular views but preserved for reporting and audit purposes.

Endpoint: http://localhost:5000/api/customers/:id/archive

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `customers:archive` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique customer ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/customers/customer_id/archive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/customers/invalid-id/archive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Customer archived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Customer not found"
}
```

The API returns a 200 OK status when the customer is successfully archived. A 404 Not Found status indicates the customer ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.3.1.6 Unarchive Customer

Version: 1.6

Date: November 26, 2025

Description: Restore a previously archived customer so they become visible again in active customer lists.

Endpoint: http://localhost:5000/api/customers/:id/unarchive

Method: PUT

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `customers:unarchive` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique customer ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/customers/customer_id/unarchive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/customers/invalid-id/unarchive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Customer unarchived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Customer not found"
}
```

The API returns a 200 OK status when the customer is successfully unarchived. A 404 Not Found status indicates the customer ID does not exist or is not archived. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.3.1.7 Delete Customer

Version: 1.6

Date: November 26, 2025

Description: Permanently delete a customer record from the system. This operation is destructive and should be used only when absolutely necessary, as it cannot be undone.

Endpoint: http://localhost:5000/api/customers/:id

Method: DELETE

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Requires the `customers:delete` permission.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _id - required; the unique customer ID in the URL path.

Requests:

Valid Request

```
DELETE http://localhost:5000/api/customers/customer_id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
DELETE http://localhost:5000/api/customers/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Customer deleted successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Customer not found"
}
```

The API returns a 200 OK status when the customer is successfully deleted. A 404 Not Found status indicates the customer ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 409 Conflict status may be returned if the customer has associated orders that prevent deletion.

Required Permissions:

- `customers:read` - View customers
- `customers:create` - Create customers
- `customers:update` - Update customers
- `customers:delete` - Delete customers
- `customers:archive` - Archive customers
- `customers:unarchive` - Unarchive customers

2.4 SERVICES MODULE API

2.4.1 Module Description

The Services module manages all service-related operations including creation, updates, archiving, and retrieval. All endpoints require authentication and appropriate RBAC permissions. Services define the types of laundry services offered by the system. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/services
http://localhost:5000/api/services/:id
http://localhost:5000/api/services/:id/archive
http://localhost:5000/api/services/:id/unarchive

2.4.1.1 Get All Services

Version: 1.6

Date: November 26, 2025

Description: Retrieve the complete catalog of laundry services, including pricing, turnaround time, and availability flags. Supports searching and filtering so branch staff can quickly locate the appropriate service while creating an order.

Endpoint: http://localhost:5000/api/services

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Responses are sorted alphabetically by default.
- Staff sees all active services; admins can optionally include archived services.

Parameters:

{{content}}gt;$ _token – required; JWT passed in the `Authorization` header.

{{content}}gt;$ _search – optional; partial text to match against the `name` or `code` field.

{{content}}gt;$ _includeArchived – optional boolean; when `true` includes archived services (default `false`).

Requests:

Valid Request

```
GET http://localhost:5000/api/services?search=wash
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/services
(Missing Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "_id": "service_id",
      "name": "Wash & Fold",
      "category": "Laundry",
      "priceType": "per_kg",
      "basePrice": 150,
      "turnaroundHours": 24,
      "isArchived": false
    }
  ],
  "count": 12
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The endpoint returns **200 OK** when services are fetched successfully. A **401 Unauthorized** status indicates the caller is missing or has an invalid token, while **403 Forbidden** appears when the role lacks `services:read`.

2.4.1.2 Get Single Service

Version: 1.6

Date: November 26, 2025

Description: Retrieve detailed information for a specific service by its unique service ID, including pricing, turnaround time, and availability status. This endpoint is used when viewing or editing service details.

Endpoint: http://localhost:5000/api/services/:id

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Staff can view all active services; admins can also view archived services.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique service ID in the URL path.

Requests:

Valid Request

```
GET http://localhost:5000/api/services/service_id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/services/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "service_id",
    "name": "Wash & Fold",
    "code": "WNF-01",
    "category": "Laundry",
    "priceType": "per_kg",
    "basePrice": 150,
    "turnaroundHours": 24,
    "description": "Standard wash and fold service",
    "isArchived": false,
    "createdAt": "2025-11-26T10:05:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Service not found"
}
```

The API returns a 200 OK status when the service is found and accessible. A 404 Not Found status indicates the service ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.4.1.3 Create Service

Version: 1.6

Date: November 26, 2025

Description: Create a new service record with pricing information so that it becomes selectable in the Admin and Staff order forms.

Endpoint: http://localhost:5000/api/services

Method: POST

Configurations:

- Requires `Authorization: Bearer <token>` and `Content-Type: application/json`.
- Only roles with `services:create` permission can call this endpoint.
- `code` must be unique per tenant; validation occurs server-side.

Parameters:

{{content}}gt;$ _name – required; descriptive service label.

{{content}}gt;$ _code – required; short unique identifier (e.g., `WNF-01`).

{{content}}gt;$ _priceType – required; one of `per_kg`, `per_item`, or `flat`.

{{content}}gt;$ _basePrice – required numeric value greater than zero.

{{content}}gt;$ _turnaroundHours – optional integer for SLA tracking.

{{content}}gt;$ _description – optional long-form explanation shown in the Admin UI.

Requests:

Valid Request

```
{
  "name": "Premium Dry Clean",
  "code": "PDC-001",
  "priceType": "per_item",
  "basePrice": 220,
  "turnaroundHours": 48,
  "description": "Includes stain treatment and pressing"
}
```

Not Valid Request

```
{
  "name": "",
  "code": "PDC-001",
  "priceType": "per_item"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "service_id",
    "name": "Premium Dry Clean",
    "code": "PDC-001",
    "priceType": "per_item",
    "basePrice": 220,
    "turnaroundHours": 48,
    "isArchived": false,
    "createdAt": "2025-11-26T10:05:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Validation error: name is required"
}
```

The endpoint returns **201 Created** when the service is saved. Duplicate codes trigger **409 Conflict**, while invalid payloads return **400 Bad Request**.

2.4.1.4 Update Service

Version: 1.6

Date: November 26, 2025

Description: Modify pricing, descriptions, or SLA details for an existing service.

Endpoint: http://localhost:5000/api/services/:id

Method: PUT

Configurations:

- Requires `Authorization: Bearer <token>` and `Content-Type: application/json`.
- Partial updates are supported; unspecified fields remain unchanged.

Parameters:

{{content}}gt;$ _id – required path parameter referencing the service MongoDB identifier.

{{content}}gt;$ _name/_priceType/_basePrice/_turnaroundHours/_description – optional body fields that will override existing values.

Requests:

Valid Request

```
{
  "basePrice": 180,
  "turnaroundHours": 18
}
```

Not Valid Request

```
{
  "priceType": "unsupported-mode"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "service_id",
    "name": "Wash & Fold",
    "basePrice": 180,
    "turnaroundHours": 18,
    "updatedAt": "2025-11-26T11:20:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Service not found"
}
```

Successful updates return **200 OK**. Non-existent IDs respond with **404 Not Found**, while validation issues return **422 Unprocessable Entity**.

2.4.1.5 Archive Service

Version: 1.6

Date: November 26, 2025

Description: Soft-delete a service by marking it as archived so it no longer appears in order forms while preserved for reporting and audit purposes. Archived services can be restored using the unarchive endpoint.

Endpoint: http://localhost:5000/api/services/:id/archive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `services:archive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique service ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/services/service_id/archive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/services/invalid-id/archive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Service archived successfully",
  "data": {
    "_id": "service_id",
    "isArchived": true,
    "archivedAt": "2025-11-26T12:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Service not found or already archived"
}
```

The API returns a 200 OK status when the service is successfully archived. A 404 Not Found status indicates the service ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 409 Conflict status may be returned if the service is already archived.

2.4.1.6 Unarchive Service

Version: 1.6

Date: November 26, 2025

Description: Restore a previously archived service so it becomes visible again in order forms and service lists. This endpoint reverses the archive operation.

Endpoint: http://localhost:5000/api/services/:id/unarchive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `services:unarchive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique service ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/services/service_id/unarchive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/services/invalid-id/unarchive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Service unarchived successfully",
  "data": {
    "_id": "service_id",
    "isArchived": false,
    "archivedAt": null
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Service not found or not archived"
}
```

The API returns a 200 OK status when the service is successfully unarchived. A 404 Not Found status indicates the service ID does not exist or the service is not archived. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.5 EXPENSES MODULE API

2.5.1 Module Description

The Expenses module manages all expense-related operations including creation, approval workflow, updates, archiving, and retrieval. All endpoints require authentication and appropriate RBAC permissions. Staff users can create expense requests, while admins can approve or reject them. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/expenses
http://localhost:5000/api/expenses/:id
http://localhost:5000/api/expenses/:id/approve
http://localhost:5000/api/expenses/:id/reject
http://localhost:5000/api/expenses/:id/archive
http://localhost:5000/api/expenses/:id/unarchive

2.5.1.1 Get All Expenses

Version: 1.6

Date: November 26, 2025

Description: Retrieve all expense records with optional filtering by status and station. This endpoint powers the Admin and owner review screens for branch reimbursements and cash outflows.

Endpoint: http://localhost:5000/api/expenses

Method: GET

Configurations:

- Requires `Authorization: Bearer <token>`.
- Admins can see all expenses; staff are limited to expenses they submitted.

Parameters:

{{content}}gt;$ _token – required; JWT in the `Authorization` header.

{{content}}gt;$ _status – optional; one of `pending`, `approved`, `rejected`, or `all` (default `pending`).

{{content}}gt;$ _stationId – optional; filter expenses by station.

Requests:

Valid Request

```
GET http://localhost:5000/api/expenses?status=pending
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/expenses
(Missing Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "_id": "expense_id",
      "description": "Detergent purchase",
      "amount": 1200,
      "status": "pending",
      "stationId": "STN-001",
      "submittedBy": "staff1",
      "createdAt": "2025-11-26T08:30:00.000Z"
    }
  ],
  "count": 5
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status for successful retrieval of expense lists. A 401 Unauthorized status indicates missing or invalid authentication credentials, while 403 Forbidden is returned when the user lacks the required permissions to view expenses.

2.5.1.2 Get Single Expense

Version: 1.6

Date: November 26, 2025

Description: Retrieve detailed information for a specific expense by its unique expense ID, including status, approval details, and submission information. This endpoint is used when viewing or processing individual expense requests.

Endpoint: http://localhost:5000/api/expenses/:id

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Staff can only view expenses they submitted; admins can view all expenses.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique expense ID in the URL path.

Requests:

Valid Request

```
GET http://localhost:5000/api/expenses/expense_id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/expenses/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "expense_id",
    "description": "Fabric softener purchase",
    "amount": 850,
    "category": "supplies",
    "status": "pending",
    "stationId": "STN-001",
    "submittedBy": {
      "_id": "user_id",
      "username": "staff1",
      "email": "staff1@example.com"
    },
    "createdAt": "2025-11-26T08:30:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Expense not found"
}
```

The API returns a 200 OK status when the expense is found and accessible. A 404 Not Found status indicates the expense ID does not exist or the user lacks permission to view it. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.5.1.3 Create Expense

Version: 1.6

Date: November 26, 2025

Description: Submit a new expense request for approval, including amount, category, and notes.

Endpoint: http://localhost:5000/api/expenses

Method: POST

Configurations:

- Requires `Authorization: Bearer <token>` and `Content-Type: application/json`.
- Accessible to roles with `expenses:create`.

Parameters:

{{content}}gt;$ _description – required; short explanation of the expense.

{{content}}gt;$ _amount – required; numeric value greater than zero.

{{content}}gt;$ _category – optional; e.g., `supplies`, `maintenance`, `utilities`.

{{content}}gt;$ _stationId – optional; station where the expense occurred (defaults to user's station).

Requests:

Valid Request

```
{
  "description": "Fabric softener purchase",
  "amount": 850,
  "category": "supplies",
  "stationId": "STN-001"
}
```

Not Valid Request

```
{
  "description": "",
  "amount": -10
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "expense_id",
    "description": "Fabric softener purchase",
    "amount": 850,
    "status": "pending",
    "stationId": "STN-001"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Validation error: description and amount are required"
}
```

The API returns a 201 Created status when the expense is successfully created. A 400 Bad Request status indicates invalid input, such as missing required fields or invalid amount. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.5.1.4 Update Expense

Version: 1.6

Date: November 26, 2025

Description: Update an existing expense request before it is approved or rejected. Only pending expenses can be updated. This endpoint allows staff to correct expense details or amounts.

Endpoint: http://localhost:5000/api/expenses/:id

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `expenses:update` permission.
- Only pending expenses can be updated.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique expense ID in the URL path.

{{content}}gt;$ _description - optional; updated expense description.

{{content}}gt;$ _amount - optional; updated expense amount (must be greater than zero).

{{content}}gt;$ _category - optional; updated expense category.

Requests:

Valid Request

```
{
  "description": "Updated expense description",
  "amount": 900,
  "category": "maintenance"
}
```

Not Valid Request

```
{
  "amount": -100,
  "description": ""
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "expense_id",
    "description": "Updated expense description",
    "amount": 900,
    "category": "maintenance",
    "status": "pending",
    "updatedAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Expense not found or cannot be updated"
}
```

The API returns a 200 OK status when the expense is successfully updated. A 404 Not Found status indicates the expense ID does not exist. A 400 Bad Request status is returned for invalid input. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 409 Conflict status may be returned if the expense is already approved or rejected.

2.5.1.5 Approve Expense

Version: 1.6

Date: November 26, 2025

Description: Approve a pending expense request, marking it as approved and optionally providing feedback to the requesting staff member. Only administrators with approval permissions can use this endpoint.

Endpoint: http://localhost:5000/api/expenses/:id/approve

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `expenses:approve` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique expense ID in the URL path.

{{content}}gt;$ _feedback - optional; textual explanation or notes about the approval decision.

Requests:

Valid Request

```
{
  "feedback": "Approved, attach receipt next time."
}
```

Not Valid Request

```
PUT http://localhost:5000/api/expenses/invalid-id/approve
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Expense approved successfully",
  "data": {
    "_id": "expense_id",
    "status": "approved",
    "approvedBy": {
      "_id": "user_id",
      "username": "admin1",
      "email": "admin1@example.com"
    },
    "approvedAt": "2025-11-26T09:00:00.000Z",
    "feedback": "Approved, attach receipt next time."
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Expense not found or already processed"
}
```


The API returns a 200 OK status when the expense is successfully approved. A 404 Not Found status indicates the expense ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 409 Conflict status is returned if the expense is already approved or rejected.

2.5.1.6 Reject Expense

Version: 1.6

Date: November 26, 2025

Description: Reject a pending expense request, marking it as rejected and optionally providing feedback explaining the rejection. Only administrators with rejection permissions can use this endpoint.

Endpoint: http://localhost:5000/api/expenses/:id/reject

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `expenses:reject` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique expense ID in the URL path.

{{content}}gt;$ _feedback - optional; textual explanation of the rejection reason.

Requests:

Valid Request


```
{
  "feedback": "Receipt required for expenses over ₱500"
}
```


Not Valid Request

```
PUT http://localhost:5000/api/expenses/invalid-id/reject
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "message": "Expense rejected successfully",
  "data": {
    "_id": "expense_id",
    "status": "rejected",
    "rejectedBy": {
      "_id": "user_id",
      "username": "admin1",
      "email": "admin1@example.com"
    },
    "rejectedAt": "2025-11-26T09:00:00.000Z",
    "feedback": "Receipt required for expenses over ₱500"
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Expense not found or already processed"
}
```


The API returns a 200 OK status when the expense is successfully rejected. A 404 Not Found status indicates the expense ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 409 Conflict status is returned if the expense is already approved or rejected.

2.5.1.7 Archive Expense

Version: 1.6

Date: November 26, 2025

Description: Soft-delete an expense by marking it as archived so it is hidden from day-to-day views while preserved for reporting and audit purposes. Archived expenses can be restored using the unarchive endpoint.

Endpoint: http://localhost:5000/api/expenses/:id/archive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `expenses:archive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique expense ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/expenses/expense_id/archive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/expenses/invalid-id/archive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Expense archived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Expense not found"
}
```

The API returns a 200 OK status when the expense is successfully archived. A 404 Not Found status indicates the expense ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.5.1.8 Unarchive Expense

Version: 1.6

Date: November 26, 2025

Description: Restore a previously archived expense so it becomes visible again in active expense lists. This endpoint reverses the archive operation.

Endpoint: http://localhost:5000/api/expenses/:id/unarchive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `expenses:unarchive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique expense ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/expenses/expense_id/unarchive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/expenses/invalid-id/unarchive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Expense unarchived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Expense not found or not archived"
}
```

The API returns a 200 OK status when the expense is successfully unarchived. A 404 Not Found status indicates the expense ID does not exist or the expense is not archived. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.6 EMPLOYEES MODULE API

2.6.1 Module Description

The Employees module manages all employee-related operations including creation, updates, performance tracking, archiving, and retrieval. All endpoints require authentication and appropriate RBAC permissions. Admins can manage all employees, while staff can view their own information. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/employees
http://localhost:5000/api/employees/:id
http://localhost:5000/api/employees/:id/performance
http://localhost:5000/api/employees/:id/toggle-account
http://localhost:5000/api/employees/:id/archive
http://localhost:5000/api/employees/:id/unarchive

2.6.1.1 Get All Employees

Version: 1.6

Date: November 26, 2025

Description: Return a list of all employees with role and station details. Used by admins to manage accounts and review staffing.

Endpoint: http://localhost:5000/api/employees

Method: GET

Configurations:

- Requires `Authorization: Bearer <token>`.
- Only roles with `employees:read` may access this endpoint.

Parameters:

{{content}}gt;$ _token – required; JWT in `Authorization` header.

{{content}}gt;$ _stationId – optional; filter by station.

Requests:

Valid Request

```
GET http://localhost:5000/api/employees?stationId=STN-001
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/employees
(Missing Authorization header)
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": [
    {
      "_id": "employee_id",
      "fullName": "Juan Dela Cruz",
      "role": "staff",
      "email": "juan@example.com",
      "stationId": "STN-001",
      "isActive": true
    }
  ],
  "count": 3
}
```


Error Response


```
{
  "success": false,
  "message": "Authentication required"
}
```


The API returns a 200 OK status for successful retrieval of employee lists. A 401 Unauthorized status indicates missing or invalid authentication credentials, while 403 Forbidden is returned when the user lacks the required permissions to view employees.

2.6.1.2 Get Single Employee

Version: 1.6

Date: November 26, 2025

Description: Retrieve detailed information for a specific employee by their unique employee ID, including role, station assignment, account status, and performance metrics. This endpoint is used when viewing or editing individual employee profiles.

Endpoint: http://localhost:5000/api/employees/:id

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Staff can only view their own information; admins can view all employees.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique employee ID in the URL path.

Requests:

Valid Request

```
GET http://localhost:5000/api/employees/employee_id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/employees/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "employee_id",
    "fullName": "Maria Santos",
    "email": "maria@example.com",
    "role": "staff",
    "stationId": "STN-002",
    "isActive": true,
    "createdAt": "2025-11-26T08:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Employee not found"
}
```

The API returns a 200 OK status when the employee is found and accessible. A 404 Not Found status indicates the employee ID does not exist or the user lacks permission to view it. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.6.1.3 Create Employee

Version: 1.6

Date: November 26, 2025

Description: Register a new employee profile that can later be linked to a user account.

Endpoint: http://localhost:5000/api/employees

Method: POST

Configurations:

- Requires `Authorization: Bearer <token>` and `Content-Type: application/json`.
- Restricted to admins with `employees:create`.

Parameters:

{{content}}gt;$ _fullName – required; employee's full name.

{{content}}gt;$ _email – required; email address for communication.

{{content}}gt;$ _role – required; `staff` or `admin`.

{{content}}gt;$ _stationId – optional; default station assignment.

Requests:

Valid Request

```
{
  "fullName": "Maria Santos",
  "email": "maria@example.com",
  "role": "staff",
  "stationId": "STN-002"
}
```

Not Valid Request

```
{
  "fullName": "",
  "email": "not-an-email",
  "role": "staff"
}
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": {
    "_id": "employee_id",
    "fullName": "Maria Santos",
    "email": "maria@example.com",
    "role": "staff",
    "stationId": "STN-002",
    "isActive": true
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Validation error: fullName and email are required"
}
```


The API returns a 201 Created status when the employee is successfully created. A 400 Bad Request status indicates invalid input, such as missing required fields or invalid email format. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.6.1.4 Update Employee

Version: 1.6

Date: November 26, 2025

Description: Update an existing employee's information including name, email, role, and station assignment. This endpoint allows admins to modify employee details while preserving historical records.

Endpoint: http://localhost:5000/api/employees/:id

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `employees:update` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique employee ID in the URL path.

{{content}}gt;$ _fullName - optional; updated employee full name.

{{content}}gt;$ _email - optional; updated email address.

{{content}}gt;$ _role - optional; updated role (`staff` or `admin`).

{{content}}gt;$ _stationId - optional; updated station assignment.

Requests:

Valid Request

```
{
  "fullName": "Updated Employee Name",
  "email": "updated@example.com",
  "stationId": "STN-003"
}
```

Not Valid Request

```
{
  "email": "invalid-email",
  "role": "invalid-role"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "employee_id",
    "fullName": "Updated Employee Name",
    "email": "updated@example.com",
    "role": "staff",
    "stationId": "STN-003",
    "updatedAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Employee not found or validation error"
}
```

The API returns a 200 OK status when the employee is successfully updated. A 404 Not Found status indicates the employee ID does not exist. A 400 Bad Request status is returned for invalid input. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.6.1.5 Toggle Employee Account

Version: 1.6

Date: November 26, 2025

Description: Enable or disable an employee's login access without deleting their historical records.

Endpoint: http://localhost:5000/api/employees/:id/toggle-account

Method: PUT

Configurations:

- Requires `Authorization: Bearer <token>`.
- Restricted to admins with `employees:update`.

Parameters:

{{content}}gt;$ _id – required; employee identifier in the URL.

Requests:

Valid Request

```
PUT http://localhost:5000/api/employees/6790c4de12/toggle-account
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/employees/invalid-id/toggle-account
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Employee account status updated",
  "data": {
    "_id": "employee_id",
    "isActive": false
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Employee not found"
}
```

The API returns a 200 OK status when the employee account status is successfully toggled. A 404 Not Found status indicates the employee ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.6.1.6 Archive Employee

Version: 1.6

Date: November 26, 2025

Description: Soft-delete an employee by marking them as archived so they are hidden from active employee lists while preserved for audit logs and historical reports. Archived employees can be restored using the unarchive endpoint.

Endpoint: http://localhost:5000/api/employees/:id/archive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `employees:archive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique employee ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/employees/employee_id/archive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/employees/invalid-id/archive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Employee archived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Employee not found"
}
```

The API returns a 200 OK status when the employee is successfully archived. A 404 Not Found status indicates the employee ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.6.1.7 Unarchive Employee

Version: 1.6

Date: November 26, 2025

Description: Restore a previously archived employee so they become visible again in active employee lists. This endpoint reverses the archive operation.

Endpoint: http://localhost:5000/api/employees/:id/unarchive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `employees:unarchive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique employee ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/employees/employee_id/unarchive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/employees/invalid-id/unarchive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Employee unarchived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Employee not found or not archived"
}
```

The API returns a 200 OK status when the employee is successfully unarchived. A 404 Not Found status indicates the employee ID does not exist or the employee is not archived. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.7 DISCOUNTS MODULE API

2.7.1 Module Description

The Discounts module manages all discount-related operations including creation, updates, usage tracking, archiving, and retrieval. All endpoints require authentication and appropriate RBAC permissions. Discounts can be applied to orders to provide promotional pricing. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/discounts
http://localhost:5000/api/discounts/:id
http://localhost:5000/api/discounts/:id/reset-usage
http://localhost:5000/api/discounts/:id/archive
http://localhost:5000/api/discounts/:id/unarchive

2.7.1.1 Get All Discounts

Version: 1.6

Date: November 26, 2025

Description: Retrieve all configured discounts, including percentage, fixed amount, and usage limits.

Endpoint: http://localhost:5000/api/discounts

Method: GET

Configurations:

- Requires `Authorization: Bearer <token>`.
- Only roles with `discounts:read` may access this endpoint.

Parameters:

{{content}}gt;$ _token – required; JWT in `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/discounts
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/discounts
(Missing Authorization header)
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": [
    {
      "_id": "discount_id",
      "name": "New Customer 10%",
      "type": "percentage",
      "value": 10,
      "maxUsage": 100,
      "currentUsage": 5,
      "isArchived": false
    }
  ],
  "count": 1
}
```


Error Response


```
{
  "success": false,
  "message": "Authentication required"
}
```


The API returns a 200 OK status for successful retrieval of discount lists. A 401 Unauthorized status indicates missing or invalid authentication credentials, while 403 Forbidden is returned when the user lacks the required permissions to view discounts.

2.7.1.2 Get Single Discount

Version: 1.6

Date: November 26, 2025

Description: Retrieve detailed information for a specific discount by its unique discount ID, including type, value, usage limits, and current usage count. This endpoint is used when viewing or editing individual discount configurations.

Endpoint: http://localhost:5000/api/discounts/:id

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `discounts:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique discount ID in the URL path.

Requests:

Valid Request

```
GET http://localhost:5000/api/discounts/discount_id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/discounts/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": {
    "_id": "discount_id",
    "name": "New Customer 10%",
    "type": "percentage",
    "value": 10,
    "maxUsage": 100,
    "currentUsage": 5,
    "isArchived": false,
    "createdAt": "2025-11-26T08:00:00.000Z"
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Discount not found"
}
```


The API returns a 200 OK status when the discount is found and accessible. A 404 Not Found status indicates the discount ID does not exist or the user lacks permission to view it. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.7.1.3 Create Discount

Version: 1.6

Date: November 26, 2025

Description: Define a new discount that can be applied to orders.

Endpoint: http://localhost:5000/api/discounts

Method: POST

Configurations:

- Requires `Authorization: Bearer <token>` and `Content-Type: application/json`.
- Restricted to roles with `discounts:create`.

Parameters:

{{content}}gt;$ _name – required; discount name.

{{content}}gt;$ _type – required; `percentage` or `fixed`.

{{content}}gt;$ _value – required; numeric percentage or amount.

{{content}}gt;$ _maxUsage – optional; maximum number of times the discount can be used.

Requests:

Valid Request


```
{
  "name": "Holiday Promo 15%",
  "type": "percentage",
  "value": 15,
  "maxUsage": 50
}
```


Not Valid Request

```
{
  "name": "",
  "type": "percentage",
  "value": -5
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "discount_id",
    "name": "Holiday Promo 15%",
    "type": "percentage",
    "value": 15,
    "maxUsage": 50,
    "currentUsage": 0
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Validation error: name and value are required"
}
```

The API returns a 201 Created status when the discount is successfully created. A 400 Bad Request status indicates invalid input, such as missing required fields or invalid discount type/value. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.7.1.4 Update Discount

Version: 1.6

Date: November 26, 2025

Description: Update an existing discount's configuration including name, type, value, and usage limits. This endpoint allows admins to modify discount details while preserving usage history.

Endpoint: http://localhost:5000/api/discounts/:id

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `discounts:update` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique discount ID in the URL path.

{{content}}gt;$ _name - optional; updated discount name.

{{content}}gt;$ _type - optional; updated discount type (`percentage` or `fixed`).

{{content}}gt;$ _value - optional; updated discount value (must be positive).

{{content}}gt;$ _maxUsage - optional; updated maximum usage limit.

Requests:

Valid Request

```
{
  "name": "Updated Discount Name",
  "value": 20,
  "maxUsage": 200
}
```

Not Valid Request

```
{
  "value": -10,
  "type": "invalid-type"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "discount_id",
    "name": "Updated Discount Name",
    "type": "percentage",
    "value": 20,
    "maxUsage": 200,
    "currentUsage": 5,
    "updatedAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Discount not found or validation error"
}
```

The API returns a 200 OK status when the discount is successfully updated. A 404 Not Found status indicates the discount ID does not exist. A 400 Bad Request status is returned for invalid input. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.7.1.5 Reset Discount Usage

Version: 1.6

Date: November 26, 2025

Description: Reset the `currentUsage` counter of a discount back to zero, typically done at the end of a promotion period.

Endpoint: http://localhost:5000/api/discounts/:id/reset-usage

Method: PUT

Configurations:

- Requires `Authorization: Bearer <token>`.
- Restricted to admins with `discounts:update`.

Parameters:

{{content}}gt;$ _id – required; discount identifier in the URL.

Requests:

Valid Request

```
PUT http://localhost:5000/api/discounts/6790e6ff44/reset-usage
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/discounts/invalid-id/reset-usage
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Discount usage reset successfully",
  "data": {
    "_id": "discount_id",
    "currentUsage": 0
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Discount not found"
}
```

The API returns a 200 OK status when the discount usage is successfully reset. A 404 Not Found status indicates the discount ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.7.1.6 Archive Discount

Version: 1.6

Date: November 26, 2025

Description: Soft-delete a discount by marking it as archived so it is hidden from active discount lists and cannot be applied to new orders, while preserving historical usage data. Archived discounts can be restored using the unarchive endpoint.

Endpoint: http://localhost:5000/api/discounts/:id/archive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `discounts:archive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique discount ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/discounts/discount_id/archive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/discounts/invalid-id/archive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Discount archived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Discount not found"
}
```

The API returns a 200 OK status when the discount is successfully archived. A 404 Not Found status indicates the discount ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.7.1.7 Unarchive Discount

Version: 1.6

Date: November 26, 2025

Description: Restore a previously archived discount so it becomes visible again in active discount lists and can be applied to new orders. This endpoint reverses the archive operation.

Endpoint: http://localhost:5000/api/discounts/:id/unarchive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `discounts:unarchive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique discount ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/discounts/discount_id/unarchive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/discounts/invalid-id/unarchive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Discount unarchived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Discount not found or not archived"
}
```

The API returns a 200 OK status when the discount is successfully unarchived. A 404 Not Found status indicates the discount ID does not exist or the discount is not archived. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.8 DASHBOARD MODULE API

2.8.1 Module Description

The Dashboard module provides aggregated statistics and key performance indicators for the LaundryPOS system. All endpoints require authentication and appropriate RBAC permissions. The dashboard displays revenue trends, order statistics, customer metrics, and recent activity. The API employs comprehensive response codes: 200 (OK) for successful operations, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/dashboard/stats

2.8.1.1 Get Dashboard Statistics

Version: 1.6

Date: November 26, 2025

Description: Get comprehensive dashboard statistics and metrics including revenue, orders, customers, and trends. This endpoint provides aggregated data for the admin dashboard view.

Endpoint: http://localhost:5000/api/dashboard/stats

Method: GET

Configurations:

- The API request requires the Authorization: Bearer <token> header.
- Staff users see statistics only for their assigned station.
- Results are calculated based on the specified time range.

Parameters:

{{content}}gt;$ _token - required; must be included in the request header for user authentication.

{{content}}gt;$ _timeRange - optional; time range for statistics: "today", "week", "month", or "year" (default: "today").

Requests:

Valid Request

```
GET http://localhost:5000/api/dashboard/stats?timeRange=month
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/dashboard/stats?timeRange=invalid
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "stats": {
      "orders": 25,
      "revenue": 37500,
      "pending": 5,
      "customers": 150
    },
    "trends": {
      "orders": 15,  // Percentage change from previous period
      "revenue": 20,
      "pending": -10,
      "customers": 5
    },
    "orderStatus": [
      { "name": "Pending", "value": 5 },
      { "name": "In Progress", "value": 10 },
      { "name": "Completed", "value": 10 }
    ],
    "revenueTrend": [
      {
        "name": "Mon",
        "value": 5000,
        "target": 3000
      },
      {
        "name": "Tue",
        "value": 6000,
        "target": 3000
      }
      // ... last 7 days
    ],
    "recentActivity": [
      {
        "id": "order_id",
        "type": "order",
        "message": "New order ORD-001 from John Doe",
        "time": "2 hours ago",
        "orderId": "ORD-001"
      }
    ],
    "pendingExpenses": 3  // Admin only
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when dashboard statistics are successfully retrieved. A 400 Bad Request status indicates invalid time range parameter. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. Staff users will only see statistics for their assigned station, while admins see system-wide statistics.

2.9 REPORTS MODULE API

2.9.1 Module Description

The Reports module generates comprehensive reports for orders, revenue, customers, expenses, services, employees, and branch performance. All endpoints require authentication and appropriate RBAC permissions. Staff users can only generate reports for their own data, while admins can generate system-wide reports. The API employs comprehensive response codes: 200 (OK) for successful operations, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/reports/orders
http://localhost:5000/api/reports/revenue
http://localhost:5000/api/reports/customers
http://localhost:5000/api/reports/expenses
http://localhost:5000/api/reports/services
http://localhost:5000/api/reports/employee
http://localhost:5000/api/reports/sales-per-branch
http://localhost:5000/api/reports/cashflow-per-branch

2.9.1.1 Generate Orders Report

Version: 1.6

Date: November 26, 2025

Description: Generate a detailed orders report for the specified date range, including order counts, totals, payment status breakdown, and individual order lines. Staff users can only generate reports for orders they created, while admins can generate system-wide reports.

Endpoint: http://localhost:5000/api/reports/orders

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```

Not Valid Request

```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "reportType": "orders",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "summary": {
      "totalOrders": 100,
      "totalRevenue": "150000.00",
      "paidOrders": 80,
      "unpaidOrders": 15,
      "partialOrders": 5
    },
    "orders": [
      {
        "id": "order_id",
        "date": "2024-01-15",
        "customer": "John Doe",
        "customerPhone": "+63 900 123 4567",
        "payment": "Paid",
        "total": "₱1,500.00",
        "paid": "₱1,500.00",
        "balance": "₱0.00",
        "items": [
          {
            "service": "Wash & Fold",
            "quantity": 5,
            "status": "Completed",
            "amount": "₱1,500.00"
          }
        ],
        "pickupDate": "2024-01-20",
        "deliveryDate": "2024-01-22",
        "notes": "Handle with care",
        "createdBy": "staff1"
      }
    ]
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Date range is required"
}
```


The API returns a 200 OK status when the orders report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.9.1.2 Generate Revenue Report

Version: 1.6

Date: November 26, 2025

Description: Generate a comprehensive revenue report with profit/loss analysis, daily breakdowns, and expense categorization for the given period. This report combines order revenue and expense data to provide financial insights.

Endpoint: http://localhost:5000/api/reports/revenue

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.

- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```

Not Valid Request

```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "reportType": "revenue",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "summary": {
      "totalRevenue": 150000,
      "totalExpenses": 30000,
      "profit": 120000,
      "profitMargin": 80
    },
    "dailyBreakdown": [
      {
        "date": "2024-01-01",
        "revenue": "5000.00",
        "expenses": "1000.00",
        "profit": "4000.00",
        "orders": 10
      }
    ],
    "expenseByCategory": {
      "supplies": 10000,
      "maintenance": 8000,
      "utilities": 12000
    }
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Date range is required"
}
```

The API returns a 200 OK status when the revenue report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.9.1.3 Generate Customers Report

Version: 1.6

Date: November 26, 2025

Description: Generate customer statistics and a list of customers active within the requested date range, including new and returning customer metrics.

Endpoint: http://localhost:5000/api/reports/customers

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```

Not Valid Request

```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "reportType": "customers",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "summary": {
      "totalCustomers": 150,
      "newCustomers": 25,
      "returningCustomers": 125
    },
    "customers": [
      {
        "id": "customer_id",
        "name": "John Doe",
        "email": "john@example.com",
        "phone": "+63 900 123 4567",
        "totalOrders": 5,
        "totalSpent": "₱7,500.00",
        "firstOrderDate": "2024-01-15",
        "lastOrderDate": "2024-01-28"
      }
    ]
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Date range is required"
}
```


The API returns a 200 OK status when the customers report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.9.1.4 Generate Expenses Report

Version: 1.6

Date: November 26, 2025

Description: Generate a comprehensive expenses report showing totals, categorized breakdowns, and individual expense records for the specified date range.

Endpoint: http://localhost:5000/api/reports/expenses

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```

Not Valid Request

```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "reportType": "expenses",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "summary": {
      "totalExpenses": 30000,
      "categories": [
        { "name": "supplies", "amount": 12000 },
        { "name": "utilities", "amount": 8000 },
        { "name": "maintenance", "amount": 10000 }
      ]
    },
    "expenses": [
      {
        "id": "expense_id",
        "description": "Fabric softener purchase",
        "amount": 850,
        "category": "supplies",
        "status": "approved",
        "date": "2024-01-15",
        "requestedBy": "staff1"
      }
    ]
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Date range is required"
}
```

The API returns a 200 OK status when the expenses report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.9.1.5 Generate Services Report

Version: 1.6

Date: November 26, 2025

Description: Generate service usage statistics for the period, including top services by order count and revenue generated per service type.

Endpoint: http://localhost:5000/api/reports/services

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```

Not Valid Request

```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "reportType": "services",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "topServices": [
      {
        "name": "Wash & Fold",
        "count": 80,
        "revenue": "₱120,000.00"
      },
      {
        "name": "Dry Clean",
        "count": 40,
        "revenue": "₱80,000.00"
      }
    ],
    "summary": {
      "totalServicesUsed": 120,
      "uniqueServiceTypes": 5
    }
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Date range is required"
}
```

The API returns a 200 OK status when the services report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.9.1.6 Generate Employee Report

Version: 1.6

Date: November 26, 2025

Description: Generate employee performance metrics including orders handled, revenue generated, and productivity statistics for each employee within the specified date range.

Endpoint: http://localhost:5000/api/reports/employee

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```

Not Valid Request

```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "reportType": "employee",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "employees": [
      {
        "id": "employee_id",
        "name": "staff1",
        "fullName": "Maria Santos",
        "ordersHandled": 40,
        "revenueGenerated": 60000,
        "averageOrderValue": "₱1,500.00",
        "stationId": "STN-001"
      }
    ],
    "summary": {
      "totalEmployees": 5,
      "totalOrders": 200,
      "totalRevenue": 300000
    }
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Date range is required"
}
```


The API returns a 200 OK status when the employee report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.9.1.7 Generate Sales Per Branch Report

Version: 1.6

Date: November 26, 2025

Description: Generate sales totals grouped by station/branch, providing comparative performance metrics across all branches for the specified date range. This report helps identify top-performing locations.

Endpoint: http://localhost:5000/api/reports/sales-per-branch

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```


Not Valid Request


```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": {
    "reportType": "sales-per-branch",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "branches": [
      {
        "stationId": "STN-001",
        "stationName": "Downtown Branch",
        "totalRevenue": 75000,
        "totalOrders": 50,
        "averageOrderValue": "₱1,500.00"
      },
      {
        "stationId": "STN-002",
        "stationName": "Mall Branch",
        "totalRevenue": 75000,
        "totalOrders": 50,
        "averageOrderValue": "₱1,500.00"
      }
    ],
    "summary": {
      "totalBranches": 2,
      "totalRevenue": 150000,
      "totalOrders": 100
    }
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Date range is required"
}
```


The API returns a 200 OK status when the sales per branch report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.9.1.8 Generate Cashflow Per Branch Report

Version: 1.6

Date: November 26, 2025

Description: Generate comprehensive cashflow analysis per branch summarizing revenue, expenses, and profit margins. This report provides financial performance comparison across all branches.

Endpoint: http://localhost:5000/api/reports/cashflow-per-branch

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `reports:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - required; start date of the report period in YYYY-MM-DD format.

{{content}}gt;$ _dateTo - required; end date of the report period in YYYY-MM-DD format.

Requests:

Valid Request

```
{
  "dateFrom": "2024-01-01",
  "dateTo": "2024-01-31"
}
```

Not Valid Request

```
{
  "dateFrom": "2024-01-01"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "reportType": "cashflow-per-branch",
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    },
    "branches": [
      {
        "stationId": "STN-001",
        "stationName": "Downtown Branch",
        "revenue": 75000,
        "expenses": 15000,
        "profit": 60000,
        "profitMargin": 80
      },
      {
        "stationId": "STN-002",
        "stationName": "Mall Branch",
        "revenue": 80000,
        "expenses": 20000,
        "profit": 60000,
        "profitMargin": 75
      }
    ],
    "summary": {
      "totalRevenue": 155000,
      "totalExpenses": 35000,
      "totalProfit": 120000,
      "averageProfitMargin": 77.42
    }
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Date range is required"
}
```


The API returns a 200 OK status when the cashflow per branch report is successfully generated. A 400 Bad Request status indicates missing or invalid date parameters. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.10 STATIONS MODULE API

2.10.1 Module Description

The Stations module manages all branch/station-related operations including creation, updates, archiving, and retrieval. All endpoints require authentication and appropriate RBAC permissions except for the public endpoint. Stations represent physical branch locations in the franchise network. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/stations/public
http://localhost:5000/api/stations
http://localhost:5000/api/stations/:id
http://localhost:5000/api/stations/:id/archive
http://localhost:5000/api/stations/:id/unarchive

2.10.1.1 Get Public Stations

Version: 1.6

Date: November 26, 2025

Description: Return a list of active stations for public use on landing pages and marketing materials. This endpoint does not require authentication and provides basic station information including name, address, and contact details.

Endpoint: http://localhost:5000/api/stations/public

Method: GET

Configurations:

- No authentication required.
- Returns only active (non-archived) stations.

Parameters:

None required.

Requests:

Valid Request

```
GET http://localhost:5000/api/stations/public
```

Not Valid Request

```
POST http://localhost:5000/api/stations/public
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "_id": "station_id",
      "name": "Downtown Branch",
      "code": "STN-001",
      "address": "Main St., Malaybalay",
      "phone": "+63 900 000 0000"
    }
  ]
}
```

Error Response

```
{
  "success": false,
  "message": "No stations found"
}
```

The API returns a 200 OK status when stations are successfully retrieved. A 500 Internal Server Error status may be returned for unexpected server issues.

2.10.1.2 Get All Stations

Version: 1.6

Date: November 26, 2025

Description: Retrieve all stations including archived ones for use in admin configuration, report filters, and management interfaces. This endpoint provides complete station information with archive status.

Endpoint: http://localhost:5000/api/stations

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `stations:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/stations
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/stations
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "_id": "station_id",
      "name": "Downtown Branch",
      "code": "STN-001",
      "address": "Main St., Malaybalay",
      "phone": "+63 900 000 0000",
      "isArchived": false,
      "createdAt": "2025-11-26T08:00:00.000Z"
    }
  ]
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when stations are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.10.1.3 Create Station

Version: 1.6

Date: November 26, 2025

Description: Create a new station record in the franchise network. Stations represent physical branch locations and must have unique codes.

Endpoint: http://localhost:5000/api/stations

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires the `stations:create` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _name - required; station/branch name.

{{content}}gt;$ _code - required; unique station code (e.g., "STN-001").

{{content}}gt;$ _address - optional; physical address of the station.

{{content}}gt;$ _phone - optional; contact phone number.

Requests:

Valid Request

```
{
  "name": "New Branch",
  "code": "STN-002",
  "address": "Barangay 1",
  "phone": "+63 900 111 2222"
}
```

Not Valid Request

```
{
  "name": "",
  "code": "STN-001"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "station_id",
    "name": "New Branch",
    "code": "STN-002",
    "address": "Barangay 1",
    "phone": "+63 900 111 2222",
    "isArchived": false,
    "createdAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Station code already exists"
}
```

The API returns a 201 Created status when the station is successfully created. A 400 Bad Request status indicates invalid input, such as missing required fields or duplicate station code. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.10.1.4 Archive Station

Version: 1.6

Date: November 26, 2025

Description: Soft-delete a station by marking it as archived so it no longer appears in active station selectors while keeping historical data and reports intact. Archived stations can be restored using the unarchive endpoint.

Endpoint: http://localhost:5000/api/stations/:id/archive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `stations:archive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique station ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/stations/station_id/archive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/stations/invalid-id/archive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Station archived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Station not found"
}
```

The API returns a 200 OK status when the station is successfully archived. A 404 Not Found status indicates the station ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.10.1.5 Unarchive Station

Version: 1.6

Date: November 26, 2025

Description: Restore a previously archived station so it becomes visible again in active station lists and can be selected for new orders and assignments. This endpoint reverses the archive operation.

Endpoint: http://localhost:5000/api/stations/:id/unarchive

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `stations:unarchive` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique station ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/stations/station_id/unarchive
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/stations/invalid-id/unarchive
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Station unarchived successfully"
}
```

Error Response

```
{
  "success": false,
  "message": "Station not found or not archived"
}
```

The API returns a 200 OK status when the station is successfully unarchived. A 404 Not Found status indicates the station ID does not exist or the station is not archived. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.11 BACKUPS MODULE API

2.11.1 Module Description

The Backups module manages database and file backup operations including creation, restoration, listing, and cleanup. All endpoints require authentication and appropriate RBAC permissions. Backups ensure data safety and enable disaster recovery. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/backups
http://localhost:5000/api/backups/stats
http://localhost:5000/api/backups/:backupName/restore
http://localhost:5000/api/backups/cleanup

2.11.1.1 Create Backup

Version: 1.6

Date: November 26, 2025

Description: Create a new backup snapshot of the database and file storage. This endpoint initiates a full system backup that includes all database collections and uploaded files. The backup is stored with a timestamped name for easy identification.

Endpoint: http://localhost:5000/api/backups

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `backups:create` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
POST http://localhost:5000/api/backups
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
POST http://localhost:5000/api/backups
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Backup created successfully",
  "data": {
    "backupName": "backup-2024-01-31T23-59-59",
    "createdAt": "2024-01-31T23:59:59.000Z",
    "size": 52428800
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Failed to create backup"
}
```

The API returns a 201 Created status when the backup is successfully created. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 500 Internal Server Error may be returned if the backup process fails.

2.11.1.2 List Backups

Version: 1.6

Date: November 26, 2025

Description: Retrieve a list of all available backup snapshots with their creation dates and sizes. This endpoint helps administrators manage and monitor backup files.

Endpoint: http://localhost:5000/api/backups

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `backups:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/backups
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/backups
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "name": "backup-2024-01-31T23-59-59",
      "createdAt": "2024-01-31T23:59:59.000Z",
      "size": 52428800
    },
    {
      "name": "backup-2024-01-30T23-59-59",
      "createdAt": "2024-01-30T23:59:59.000Z",
      "size": 51200000
    }
  ]
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when backups are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.11.1.3 Get Backup Statistics

Version: 1.6

Date: November 26, 2025

Description: Retrieve aggregate statistics about all backups including total count and combined storage size. This endpoint provides an overview of backup storage usage.

Endpoint: http://localhost:5000/api/backups/stats

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `backups:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/backups/stats
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/backups/stats
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "totalBackups": 10,
    "totalSizeBytes": 524288000,
    "totalSizeMB": 500,
    "oldestBackup": "2024-01-01T00:00:00.000Z",
    "newestBackup": "2024-01-31T23:59:59.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when backup statistics are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.11.1.4 Restore Backup

Version: 1.6

Date: November 26, 2025

Description: Restore the system from a specific backup snapshot. This operation will replace the current database and files with the data from the specified backup. Use with caution as this operation is irreversible.

Endpoint: http://localhost:5000/api/backups/:backupName/restore

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `backups:restore` permission.
- This is a destructive operation that should be performed with caution.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _backupName - required; the name of the backup to restore (from the backup list).

Requests:

Valid Request

```
POST http://localhost:5000/api/backups/backup-2024-01-31T23-59-59/restore
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
POST http://localhost:5000/api/backups/invalid-backup-name/restore
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Backup restore initiated",
  "data": {
    "backupName": "backup-2024-01-31T23-59-59",
    "status": "restoring"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Backup not found"
}
```

The API returns a 200 OK status when the restore operation is successfully initiated. A 404 Not Found status indicates the backup name does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 500 Internal Server Error may be returned if the restore process fails.

2.11.1.5 Cleanup Backups

Version: 1.6

Date: November 26, 2025

Description: Remove old backup files based on retention policies to free up storage space. This endpoint typically removes backups older than a specified number of days or keeps only the most recent N backups.

Endpoint: http://localhost:5000/api/backups/cleanup

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `backups:delete` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
POST http://localhost:5000/api/backups/cleanup
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
POST http://localhost:5000/api/backups/cleanup
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Old backups cleaned up successfully",
  "data": {
    "deletedCount": 5,
    "freedSpaceBytes": 262144000
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Cleanup failed"
}
```

The API returns a 200 OK status when old backups are successfully cleaned up. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 500 Internal Server Error may be returned if the cleanup process fails.

2.12 RBAC (ROLE-BASED ACCESS CONTROL) MODULE API

2.12.1 Module Description

The RBAC module manages role-based access control permissions for the system. Most endpoints require authentication and admin role, except for the emergency recovery endpoint. This module allows administrators to configure and manage permissions for different user roles. The API employs comprehensive response codes: 200 (OK) for successful operations, 201 (Created) for resource creation, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/rbac/emergency-recover-admin
http://localhost:5000/api/rbac/resources
http://localhost:5000/api/rbac/me
http://localhost:5000/api/rbac/:role
http://localhost:5000/api/rbac/initialize

2.12.1.1 Emergency Recover Admin

Version: 1.6

Date: November 26, 2025

Description: Emergency endpoint to restore admin permissions when RBAC system is misconfigured or locked out. This endpoint bypasses all RBAC checks and requires a secret key for security. Use only as a last resort for system recovery.

Endpoint: http://localhost:5000/api/rbac/emergency-recover-admin

Method: POST

Configurations:

- No authentication required (bypasses RBAC).
- Requires a secret key in the request body for security.
- Should only be used in emergency situations.

Parameters:

{{content}}gt;$ _secretKey - required; emergency recovery secret key (configured via environment variable `EMERGENCY_RECOVERY_KEY`).

Requests:

Valid Request

```
{
  "secretKey": "EMERGENCY_ADMIN_RECOVERY_2024"
}
```

Not Valid Request

```
{
  "secretKey": "wrong-key"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Admin role recovered successfully",
  "data": {
    "role": "admin",
    "permissions": [
      {
        "resource": "orders",
        "actions": ["read", "create", "update", "delete"]
      }
    ]
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Invalid secret key"
}
```

The API returns a 200 OK status when admin permissions are successfully recovered. A 400 Bad Request status indicates missing or invalid secret key. A 500 Internal Server Error may be returned if the recovery process fails.

2.12.1.2 Get RBAC Resources

Version: 1.6

Date: November 26, 2025

Description: Retrieve a list of all available resources and actions in the RBAC system. This endpoint provides the complete list of permissions that can be assigned to roles, useful for building permission management interfaces.

Endpoint: http://localhost:5000/api/rbac/resources

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires the `rbac:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/rbac/resources
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/rbac/resources
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "orders": ["read", "create", "update", "delete"],
    "customers": ["read", "create", "update", "delete"],
    "services": ["read", "create", "update", "delete"],
    "expenses": ["read", "create", "update", "approve", "reject"],
    "employees": ["read", "create", "update", "delete"],
    "discounts": ["read", "create", "update", "delete"],
    "reports": ["read"],
    "stations": ["read", "create", "update", "delete"],
    "backups": ["read", "create", "restore", "delete"],
    "rbac": ["read", "update"],
    "system-settings": ["read", "update"]
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when resources are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.12.1.3 Get Current User Permissions

Version: 1.6

Date: November 26, 2025

Description: Retrieve the permissions for the currently authenticated user based on their role. This endpoint allows users to see what actions they are allowed to perform in the system.

Endpoint: http://localhost:5000/api/rbac/me

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Returns permissions based on the authenticated user's role.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/rbac/me
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/rbac/me
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "role": "staff",
    "permissions": [
      {
        "resource": "orders",
        "actions": ["read", "create", "update"]
      },
      {
        "resource": "customers",
        "actions": ["read", "create"]
      }
    ]
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Unable to determine user role"
}
```

The API returns a 200 OK status when permissions are successfully retrieved. A 400 Bad Request status indicates the user role cannot be determined. A 401 Unauthorized status is returned for missing or invalid authentication credentials.

2.12.1.4 Manage Role Permissions

Version: 1.6

Date: November 26, 2025

Description: Update permissions for a specific role (admin or staff). This endpoint allows administrators to configure which actions each role can perform. Only admins can modify role permissions.

Endpoint: http://localhost:5000/api/rbac/:role

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires admin role to update permissions.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _role - required; the role to update (`admin` or `staff`) in the URL path.

{{content}}gt;$ _permissions - required; array of permission objects with `resource` and `actions` properties.

{{content}}gt;$ _description - optional; description of the role permissions.

Requests:

Valid Request

```
{
  "permissions": [
    {
      "resource": "orders",
      "actions": ["read", "create", "update"]
    },
    {
      "resource": "customers",
      "actions": ["read", "create"]
    }
  ],
  "description": "Staff role with limited permissions"
}
```

Not Valid Request

```
{
  "permissions": "invalid"
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Role permissions updated successfully (RBAC cache may need refresh)",
  "data": {
    "_id": "role_permission_id",
    "role": "staff",
    "permissions": [
      {
        "resource": "orders",
        "actions": ["read", "create", "update"]
      }
    ],
    "updatedBy": "admin_user_id",
    "updatedAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Invalid role. Must be admin or staff."
}
```

The API returns a 200 OK status when role permissions are successfully updated. A 400 Bad Request status indicates invalid role or permissions format. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions (only admins can update permissions).

2.12.1.5 Initialize RBAC

Version: 1.6

Date: November 26, 2025

Description: Initialize or reload the RBAC system, refreshing the permission cache. This endpoint is useful after updating role permissions to ensure changes take effect immediately without server restart.

Endpoint: http://localhost:5000/api/rbac/initialize

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires admin role.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
POST http://localhost:5000/api/rbac/initialize
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
POST http://localhost:5000/api/rbac/initialize
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "RBAC system initialized successfully",
  "data": {
    "rolesInitialized": ["admin", "staff"],
    "timestamp": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Failed to initialize RBAC system"
}
```

The API returns a 200 OK status when the RBAC system is successfully initialized. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions. A 500 Internal Server Error may be returned if initialization fails.

2.13 SYSTEM SETTINGS MODULE API

2.13.1 Module Description

The System Settings module manages system-wide configuration settings including inactivity timeouts and other system parameters. All endpoints require authentication and typically admin role. The API employs comprehensive response codes: 200 (OK) for successful operations, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/system-settings/inactivity

2.13.1.1 Get Inactivity Settings

Version: 1.6

Date: November 26, 2025

Description: Retrieve the current inactivity timeout settings for user sessions. This setting determines how long a user can be inactive before being automatically logged out for security purposes.

Endpoint: http://localhost:5000/api/system-settings/inactivity

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires admin role or `system-settings:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/system-settings/inactivity
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/system-settings/inactivity
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "maxInactivityMinutes": 30,
    "updatedAt": "2025-11-26T08:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when inactivity settings are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.13.1.2 Update Inactivity Settings

Version: 1.6

Date: November 26, 2025

Description: Update the inactivity timeout setting for user sessions. This allows administrators to configure how long users can remain inactive before being automatically logged out.

Endpoint: http://localhost:5000/api/system-settings/inactivity

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.
- Requires admin role or `system-settings:update` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _maxInactivityMinutes - required; number of minutes before inactive users are logged out (must be positive).

Requests:

Valid Request

```
{
  "maxInactivityMinutes": 45
}
```

Not Valid Request

```
{
  "maxInactivityMinutes": -10
}
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Inactivity settings updated successfully",
  "data": {
    "maxInactivityMinutes": 45,
    "updatedAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Invalid inactivity minutes value"
}
```

The API returns a 200 OK status when inactivity settings are successfully updated. A 400 Bad Request status indicates invalid input, such as negative or zero values. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.14 UPLOAD MODULE API

2.14.1 Module Description

The Upload module handles file upload operations for images used in receipts, profiles, and other system features. All endpoints require authentication. The API employs comprehensive response codes: 200 (OK) for successful operations, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 413 (Payload Too Large) for files exceeding size limits, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/upload/image
http://localhost:5000/api/upload/images

2.14.1.1 Upload Single Image

Version: 1.6

Date: November 26, 2025

Description: Upload a single image file for use in receipts, profiles, or other system features. The image is validated, processed, and stored on the server. Supported formats typically include JPEG, PNG, and GIF.

Endpoint: http://localhost:5000/api/upload/image

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must use `multipart/form-data` content type.
- File size limits apply (typically 5MB per image).

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _image - required; image file in the form data (field name: `image`).

Requests:

Valid Request

```
POST http://localhost:5000/api/upload/image
Headers:
  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
  Content-Type: multipart/form-data
Body:
  image: [binary file data]
```

Not Valid Request

```
POST http://localhost:5000/api/upload/image
(No Authorization header or missing image file)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "url": "https://cdn.example.com/uploads/receipt-1.png",
    "filename": "receipt-1.png",
    "size": 245760,
    "mimetype": "image/png"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "File too large or invalid file type"
}
```

The API returns a 200 OK status when the image is successfully uploaded. A 400 Bad Request status indicates invalid file type or missing file. A 401 Unauthorized status is returned for missing or invalid authentication credentials. A 413 Payload Too Large status is returned when the file exceeds size limits. A 500 Internal Server Error may be returned if the upload process fails.

2.14.1.2 Upload Multiple Images

Version: 1.6

Date: November 26, 2025

Description: Upload multiple image files in a single request. All images are validated, processed, and stored on the server. Useful for bulk uploads or when multiple images need to be associated with a single record.

Endpoint: http://localhost:5000/api/upload/images

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must use `multipart/form-data` content type.
- File size limits apply per image (typically 5MB per image).
- Maximum number of images per request may be limited (e.g., 10 images).

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _images - required; array of image files in the form data (field name: `images` or multiple fields with same name).

Requests:

Valid Request

```
POST http://localhost:5000/api/upload/images
Headers:
  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
  Content-Type: multipart/form-data
Body:
  images: [binary file data 1]
  images: [binary file data 2]
```

Not Valid Request

```
POST http://localhost:5000/api/upload/images
(No Authorization header or no image files)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "url": "https://cdn.example.com/uploads/image1.png",
      "filename": "image1.png",
      "size": 245760,
      "mimetype": "image/png"
    },
    {
      "url": "https://cdn.example.com/uploads/image2.png",
      "filename": "image2.png",
      "size": 189440,
      "mimetype": "image/png"
    }
  ],
  "count": 2
}
```

Error Response

```
{
  "success": false,
  "message": "No files uploaded or invalid file types"
}
```

The API returns a 200 OK status when images are successfully uploaded. A 400 Bad Request status indicates invalid file types or missing files. A 401 Unauthorized status is returned for missing or invalid authentication credentials. A 413 Payload Too Large status is returned when files exceed size limits. A 500 Internal Server Error may be returned if the upload process fails.

2.15 SUPPORT MODULE API

2.15.1 Module Description

The Support module handles user feedback and support ticket submissions. All endpoints require authentication. The API employs comprehensive response codes: 200 (OK) for successful operations, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/support/feedback

2.15.1.1 Submit Feedback

Version: 1.6

Date: November 26, 2025

Description: Submit user feedback or support ticket to the system administrators. This endpoint allows users to report issues, request features, or provide general feedback about the system.

Endpoint: http://localhost:5000/api/support/feedback

Method: POST

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- The request must include the Content-Type: application/json header.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _topic - required; category or topic of the feedback (e.g., "Bug Report", "Feature Request", "Feedback", "Support").

{{content}}gt;$ _message - required; detailed feedback message or description.

{{content}}gt;$ _email - optional; contact email for follow-up (defaults to user's email if authenticated).

Requests:

Valid Request

```
{
  "topic": "Feature Request",
  "message": "Please add dark mode for the admin dashboard.",
  "email": "user@example.com"
}
```

Not Valid Request

```
{
  "topic": "",
  "message": ""
}
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "message": "Feedback submitted successfully",
  "data": {
    "_id": "feedback_id",
    "topic": "Feature Request",
    "message": "Please add dark mode for the admin dashboard.",
    "submittedBy": {
      "_id": "user_id",
      "username": "user1",
      "email": "user@example.com"
    },
    "status": "pending",
    "createdAt": "2025-11-26T09:00:00.000Z"
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Topic and message are required"
}
```


The API returns a 200 OK status when feedback is successfully submitted. A 400 Bad Request status indicates missing or invalid input fields. A 401 Unauthorized status is returned for missing or invalid authentication credentials. A 500 Internal Server Error may be returned if the submission process fails.

2.16 NOTIFICATIONS MODULE API

2.16.1 Module Description

The Notifications module manages real-time notifications for users including Server-Sent Events (SSE) streaming, notification retrieval, and read status management. All endpoints require authentication. The API employs comprehensive response codes: 200 (OK) for successful operations, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/notifications/stream
http://localhost:5000/api/notifications
http://localhost:5000/api/notifications/:notificationId/read
http://localhost:5000/api/notifications/read-all

2.16.1.1 Notifications Stream (SSE)

Version: 1.6

Date: November 26, 2025

Description: Open a Server-Sent Events (SSE) stream to receive real-time notifications. This endpoint maintains a persistent connection and pushes notifications to the client as they occur, enabling live updates without polling.

Endpoint: http://localhost:5000/api/notifications/stream

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Uses Server-Sent Events (SSE) protocol with `text/event-stream` content type.
- Connection remains open until closed by client or timeout.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
GET http://localhost:5000/api/notifications/stream
Headers:
  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
  Accept: text/event-stream
```

Not Valid Request

```
GET http://localhost:5000/api/notifications/stream
(No Authorization header)
```

Response Format: Server-Sent Events (SSE)

Responses:

Success Response (Stream)

```
data: {"type":"notification","data":{"_id":"notification_id","title":"New Order","message":"Order ORD-001 has been created","read":false,"createdAt":"2025-11-26T09:00:00.000Z"}}

data: {"type":"notification","data":{"_id":"notification_id2","title":"Order Updated","message":"Order ORD-001 status changed","read":false,"createdAt":"2025-11-26T09:05:00.000Z"}}
```

Error Response

```
data: {"success":false,"message":"Authentication required"}
```

The API returns a 200 OK status with `text/event-stream` content type when the stream is successfully established. A 401 Unauthorized status is returned for missing or invalid authentication credentials. The connection may close automatically after a timeout period.

2.16.1.2 Get All Notifications

Version: 1.6

Date: November 26, 2025

Description: Retrieve all notifications for the authenticated user, including both read and unread notifications. Results are typically sorted by creation date with newest first.

Endpoint: http://localhost:5000/api/notifications

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Returns only notifications for the authenticated user.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _limit - optional; maximum number of notifications to return (default: 50).

{{content}}gt;$ _offset - optional; number of notifications to skip for pagination (default: 0).

{{content}}gt;$ _read - optional; filter by read status (`true` or `false`).

Requests:

Valid Request

```
GET http://localhost:5000/api/notifications?limit=20&read=false
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/notifications
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response


```
{
  "success": true,
  "data": [
    {
      "_id": "notification_id",
      "title": "New Order",
      "message": "Order ORD-001 has been created",
      "type": "order",
      "read": false,
      "createdAt": "2025-11-26T09:00:00.000Z"
    },
    {
      "_id": "notification_id2",
      "title": "Order Updated",
      "message": "Order ORD-001 status changed to Completed",
      "type": "order",
      "read": true,
      "createdAt": "2025-11-26T08:00:00.000Z"
    }
  ],
  "count": 2,
  "unreadCount": 1
}
```


Error Response


```
{
  "success": false,
  "message": "Authentication required"
}
```


The API returns a 200 OK status when notifications are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials.

2.16.1.3 Mark Notification as Read

Version: 1.6

Date: November 26, 2025

Description: Mark a specific notification as read by its ID. This updates the notification's read status and timestamp.

Endpoint: http://localhost:5000/api/notifications/:notificationId/read

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Only the notification owner can mark their own notifications as read.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _notificationId - required; the unique notification ID in the URL path.

Requests:

Valid Request

```
PUT http://localhost:5000/api/notifications/notification_id/read
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/notifications/invalid-id/read
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "Notification marked as read",
  "data": {
    "_id": "notification_id",
    "read": true,
    "readAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Notification not found"
}
```

The API returns a 200 OK status when the notification is successfully marked as read. A 404 Not Found status indicates the notification ID does not exist or belongs to another user. A 401 Unauthorized status is returned for missing or invalid authentication credentials.

2.16.1.4 Mark All Notifications as Read

Version: 1.6

Date: November 26, 2025

Description: Mark all unread notifications for the authenticated user as read in a single operation. This is useful for bulk read status updates.

Endpoint: http://localhost:5000/api/notifications/read-all

Method: PUT

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Updates all unread notifications for the authenticated user.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

Requests:

Valid Request

```
PUT http://localhost:5000/api/notifications/read-all
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
PUT http://localhost:5000/api/notifications/read-all
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "message": "All notifications marked as read",
  "data": {
    "updatedCount": 5,
    "readAt": "2025-11-26T09:00:00.000Z"
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Failed to update notifications"
}
```

The API returns a 200 OK status when all notifications are successfully marked as read. A 401 Unauthorized status is returned for missing or invalid authentication credentials. A 500 Internal Server Error may be returned if the update process fails.

2.17 AUDIT LOGS MODULE API

2.17.1 Module Description

The Audit Logs module provides access to system audit logs for tracking user actions and system events. All endpoints require authentication and appropriate RBAC permissions. Audit logs capture user activities, IP addresses, timestamps, and action details for security and compliance. The API employs comprehensive response codes: 200 (OK) for successful operations, 400 (Bad Request) for invalid input, 401 (Unauthorized) for missing or invalid tokens, 403 (Forbidden) for insufficient permissions, 404 (Not Found) for unavailable resources, and 500 (Internal Server Error) for unexpected server issues, ensuring secure and clear communication of request outcomes.

The API have the following endpoints:
http://localhost:5000/api/audit-logs
http://localhost:5000/api/audit-logs/stats
http://localhost:5000/api/audit-logs/:id

2.17.1.1 Get Audit Logs

Version: 1.6

Date: November 26, 2025

Description: Retrieve audit logs for system activities and user actions. This endpoint provides a comprehensive log of all tracked events including user actions, IP addresses, timestamps, and action details for security and compliance purposes.

Endpoint: http://localhost:5000/api/audit-logs

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires admin role or `audit-logs:read` permission.
- Supports filtering and pagination.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _limit - optional; maximum number of logs to return (default: 50).

{{content}}gt;$ _offset - optional; number of logs to skip for pagination (default: 0).

{{content}}gt;$ _action - optional; filter by action type (e.g., "LOGIN", "CREATE", "UPDATE", "DELETE").

{{content}}gt;$ _userId - optional; filter by user ID.

{{content}}gt;$ _dateFrom - optional; filter logs from this date (YYYY-MM-DD).

{{content}}gt;$ _dateTo - optional; filter logs to this date (YYYY-MM-DD).

Requests:

Valid Request

```
GET http://localhost:5000/api/audit-logs?limit=20&action=LOGIN&dateFrom=2024-01-01
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/audit-logs
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": [
    {
      "_id": "log_id",
      "action": "LOGIN",
      "resource": "auth",
      "user": {
        "_id": "user_id",
        "username": "admin",
        "email": "admin@example.com"
      },
      "ip": "127.0.0.1",
      "userAgent": "Mozilla/5.0...",
      "details": {
        "method": "POST",
        "endpoint": "/api/auth/login"
      },
      "createdAt": "2024-01-15T10:30:00.000Z"
    }
  ],
  "count": 1,
  "total": 150
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when audit logs are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.17.1.2 Get Audit Log Statistics

Version: 1.6

Date: November 26, 2025

Description: Retrieve aggregate statistics about audit logs including total count, action breakdowns, and activity trends. This endpoint provides insights into system usage and security patterns.

Endpoint: http://localhost:5000/api/audit-logs/stats

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires admin role or `audit-logs:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _dateFrom - optional; start date for statistics (YYYY-MM-DD).

{{content}}gt;$ _dateTo - optional; end date for statistics (YYYY-MM-DD).

Requests:

Valid Request

```
GET http://localhost:5000/api/audit-logs/stats?dateFrom=2024-01-01&dateTo=2024-01-31
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/audit-logs/stats
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "totalLogs": 1500,
    "actionBreakdown": {
      "LOGIN": 200,
      "CREATE": 500,
      "UPDATE": 600,
      "DELETE": 100,
      "READ": 100
    },
    "topUsers": [
      {
        "userId": "user_id",
        "username": "admin",
        "actionCount": 300
      }
    ],
    "dateRange": {
      "from": "2024-01-01",
      "to": "2024-01-31"
    }
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Authentication required"
}
```

The API returns a 200 OK status when audit log statistics are successfully retrieved. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

2.17.1.3 Get Audit Log By ID

Version: 1.6

Date: November 26, 2025

Description: Retrieve detailed information for a specific audit log entry by its unique ID. This endpoint provides complete details of a single logged event including all metadata and action details.

Endpoint: http://localhost:5000/api/audit-logs/:id

Method: GET

Configurations:

- Requires the `Authorization: Bearer <token>` header.
- Requires admin role or `audit-logs:read` permission.

Parameters:

{{content}}gt;$ _token - required; JWT passed in the `Authorization` header.

{{content}}gt;$ _id - required; the unique audit log ID in the URL path.

Requests:

Valid Request

```
GET http://localhost:5000/api/audit-logs/log_id
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Not Valid Request

```
GET http://localhost:5000/api/audit-logs/invalid-id
(No Authorization header)
```

Response Format: JSON

Responses:

Success Response

```
{
  "success": true,
  "data": {
    "_id": "log_id",
    "action": "CREATE",
    "resource": "orders",
    "user": {
      "_id": "user_id",
      "username": "staff1",
      "email": "staff1@example.com"
    },
    "ip": "192.168.1.100",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36",
    "details": {
      "method": "POST",
      "endpoint": "/api/orders",
      "requestBody": {
        "customer": "John Doe",
        "items": []
      },
      "responseStatus": 201
    },
    "createdAt": "2024-01-15T10:30:00.000Z"
  }
}
```


Error Response


```
{
  "success": false,
  "message": "Audit log not found"
}
```


The API returns a 200 OK status when the audit log is successfully retrieved. A 404 Not Found status indicates the audit log ID does not exist. A 401 Unauthorized status is returned for missing or invalid authentication credentials, while 403 Forbidden indicates insufficient permissions.

System Utility Endpoints

Health Check

- **Endpoint**: `GET /api/health`
- **Description**: Health check endpoint for monitoring system status
- **Authentication**: Not required
- **Response** (200):


```
{
  "status": "healthy",
  "timestamp": "2024-01-15T10:30:00.000Z",
  "uptime": 3600,
  "memory": {
    "used": "150 MB",
    "total": "200 MB",
    "rss": "180 MB"
  },
  "database": "connected",
  "version": "1.0.0",
  "environment": "production"
}
```


SMS Test (Development/Debug)

- **Endpoint**: `GET /api/test-sms`
- **Description**: Test SMS functionality (for debugging)
- **Authentication**: Not required
- **Query Parameters**:

    - `phone` (string, required) - Phone number (e.g., `+1234567890`)
    - `message` (string, optional) - Test message (default: "Test SMS from Laundry POS")

- **Response** (200):

```
{
  "success": true,
  "message": "SMS sent successfully",
  "data": {
    "messageId": "sms_id",
    "status": "sent"
  }
}
```

V. Sample Requests and Responses

1. **Login**: Use the payload shown in Section 2.1.1.1.
2. **Send Reset OTP**: Follow the `/auth/forgot-password` request in Section 2.1.1.3.
3. **Reset Password**: Submit the payload in Section 2.1.1.4 to `/auth/reset-password`.
4. **Draft Order**: Use the Staff autosave payload from Section 2.2.1.2.
5. **Email Invoice**: Call `/orders/:id/send-email` as detailed in Section 2.2.1.3.

VI. Security Notes

- All passwords are hashed with bcrypt (12 salt rounds).
- Environment variables required: `JWT_SECRET`, `MONGODB_URI`, SMTP credentials, reCAPTCHA keys, and `ALLOWED_ORIGINS`.
- HTTPS should be enforced in production using certificates stored in `server/certs/`.
- Audit logs capture user, IP, user-agent, and payload metadata for every sensitive endpoint.
- Rate limiting is enforced via `middleware/rateLimiter.js` to protect authentication workflows.

VII. Changelog

Date	Changes
26 Nov 2025	Reformatted documentation to mirror the BuKSU template and embedded comprehensive endpoint details.
15 Nov 2025	Added OTP verification, password reset, and logout coverage.

VIII. References

- LaundryPOS backend source (`server/routes`, `server/controllers`)
- Internal API Reference (`docs/API_REFERENCE.md`)
- Thunder Client captures stored under `docs/images/`
- SBO Fee Collection Management System API Documentation (Sample PDF)
- API Documentation Rubric.docx

Last Updated: November 26, 2025
Documentation Version: 1.6
API Version: 1.6