# Customer Portal API Documentation

## Overview

The **Customer Portal** API is designed to manage customer data, perform user authentication, and support features like Multi-Factor Authentication (MFA), password recovery, and user registration. It is built using **Laravel**, with **Laravel Passport** for API authentication and **MySQL** as the database. The API allows for secure management of customer profiles and provides an administrative interface to handle users.

The API is organized into several key sections:

- **Authentication**: User login, registration, MFA verification, and password management.
- **Customer Management**: Admin operations for viewing, creating, updating, and deleting customer records.
- **Testing**: PHPUnit tests to ensure API reliability and correctness.
- **Swagger UI Documentation**: Interactive documentation for developers to test endpoints.

This API serves as the backend for a customer portal that allows users to log in, register, manage their accounts, and access their customer data, all while ensuring secure interactions using **JWT tokens**.

# Base URL

All API endpoints are relative to the base URL:

```
http://localhost/api
```

For testing or interacting with the API, replace `localhost` with the actual host URL where your Laravel application is running, if different.

# Authentication API

The **Authentication API** handles user login, registration, password resets, and Multi-Factor Authentication (MFA). All of these actions are essential to securing the Customer Portal, ensuring that only authorized users can access customer data and services.

## Login

- **Endpoint**: `POST /login`
- **Description**: Authenticates the user and generates an access token. This token is required for making authenticated requests to other API endpoints.
- **Request Parameters**:
    - `email`: The user's email address.
    - `password`: The user's password.
- **Response**:
    - A successful login returns a JWT access token and a refresh token.
    - The user must include the access token in the `Authorization` header for future requests to protected routes.

## Register

- **Endpoint**: `POST /register`
- **Description**: Registers a new user in the system.
- **Request Parameters**:
    - `name`: Full name of the user.
    - `email`: The user's email address.
    - `password`: The user's password.
    - `password_confirmation`: A confirmation of the password to ensure accuracy.
- **Response**:
    - A successful registration returns a JWT access token for immediate authentication.

## Verify MFA

- **Endpoint**: `POST /verify-mfa`
- **Description**: Verifies the MFA token provided by the user. This feature is essential for ensuring two-factor authentication (2FA).
- **Request Parameters**:
    - `mfa_token`: The one-time passcode sent to the user via their MFA provider.
- **Response**:
    - A success response indicates the MFA token is valid, and the user is granted access.

## Forgot Password

- **Endpoint**: `POST /forgot-password`
- **Description**: Sends a password reset link to the user's registered email address.
- **Request Parameters**:
    - `email`: The email address associated with the account.
- **Response**:
    - A success message confirming the reset link has been sent.

### Reset Password

- **Endpoint**: `POST /reset-password`
- **Description**: Resets the user's password using the token received from the "forgot password" request.
- **Request Parameters**:
  - `token`: The reset token sent to the user's email.
  - `password`: The new password.
  - `password_confirmation`: Confirmation of the new password.
- **Response**:
  - A success message indicating that the password has been updated.

### Logout

- **Endpoint**: `POST /logout`
- **Description**: Logs out the authenticated user and invalidates the access token.
- **Authentication Required**: Yes, a valid bearer token is required.
- **Response**: A success message indicating the user has been logged out.

## Customer Management API

The **Customer Management API** allows authenticated users (admin or specific roles) to perform CRUD (Create, Read, Update, Delete) operations on customer records. These actions are essential for managing the customer database effectively.

### Get All Customers

- **Endpoint**: `GET /customers`
- **Description**: Retrieves a list of all customers in the system. Admin users can view this data.
- **Authentication Required**: Yes, a valid bearer token is required to access this endpoint.
- **Response**: A list of customers, including details such as `id`, `name`, `email`, and other relevant data.

### Create Customer

- **Endpoint**: `POST /customers`
- **Description**: Creates a new customer record in the system.
- **Request Parameters**:

    - `name`: Full name of the customer.
    - `email`: Email address of the customer.
    - `phone_number`: Phone number.
    - `address`: Address of the customer.
    - `other_fields`: Any additional customer data required by the system.
- **Authentication Required**: Yes, a valid bearer token is required to access this endpoint.
- **Response**: A success message with the created customer's details.

### Get Customer by ID

- **Endpoint**: `GET /customers/{id}`
- **Description**: Retrieves a specific customer's details by their ID.
- **Authentication Required**: Yes, a valid bearer token is required to access this endpoint.
- **Response**: Customer details like `name`, `email`, and other stored attributes.

### Update Customer

- **Endpoint**: `PUT /customers/{id}`
- **Description**: Updates an existing customer's details by their ID.
- **Request Parameters**:
    - `name`, `email`, `phone_number`, `address`, etc.
- **Authentication Required**: Yes, a valid bearer token is required to access this endpoint.
- **Response**: A success message confirming that the customer record has been updated.

### Delete Customer

- **Endpoint**: `DELETE /customers/{id}`
- **Description**: Deletes a customer record by their ID.
- **Authentication Required**: Yes, a valid bearer token is required to access this endpoint.
- **Response**: A success message indicating that the customer has been deleted.

# Testing

## Unit Testing

The API supports unit tests written using **PHPUnit** to ensure that all routes and actions are working as expected. The tests include:

- **Authentication Tests**: Verifying login, registration, password resets, and MFA.

- **Customer Tests**: Validating CRUD operations for customer records.
- **Integration Tests**: Ensuring that the system works as expected with external services like email for password resets and MFA.

You can run the tests by executing the following command:

```
php artisan test
```

---

## API Documentation with Swagger

Swagger UI is integrated into the application to provide an interactive API documentation for developers and testers. This documentation allows you to test the API endpoints directly from the browser, making integration and debugging easier.

To access the Swagger API documentation, navigate to:

```
http://localhost/api/documentation
```

- 
- This page will list all available API routes along with their details, request parameters, and response formats. You can interact with the API directly by making test requests from this UI.

---

## Conclusion

The **Customer Portal API** provides essential functionality for managing users and customer data. This API allows for secure authentication, customer data management, and integration with third-party services for password recovery and Multi-Factor Authentication (MFA). The API is built with **Laravel**, making use of **Laravel Passport** for secure token-based authentication and **Swagger UI** for user-friendly documentation and interaction.

With thorough testing using **PHPUnit** and an interactive **Swagger UI** interface, this API offers a reliable and secure backend solution for the **Customer Portal** project.