

Customer Portal Documentation

Overview

The Customer Portal is a web application designed to manage customer data, provide secure user authentication, and offer administrative functionalities. It includes a backend built with Laravel and a frontend using JavaScript, Tailwind CSS, and Axios for API communication. The portal supports features like Multi-Factor Authentication (MFA), password recovery, and user registration, with a focus on user-friendly interfaces and secure interactions.

The project is organized into:

- **Backend API:** Developed using Laravel, providing endpoints for authentication, customer management, and administrative tasks.
- **Frontend Interface:** Built with JavaScript and styled using Tailwind CSS for responsive design.
- **Testing:** Comprehensive PHPUnit tests ensure reliability.
- **Documentation:** Interactive Swagger UI for API exploration and developer use.

Project Features

1. **Authentication System**
 - User login and registration.
 - Password recovery with email integration.
 - Multi-Factor Authentication (MFA) for enhanced security.
 - JWT token-based authentication.
2. **Customer Management**
 - Full CRUD functionality for customer profiles.
 - Role-based access control for secure administrative actions.
3. **Frontend Features**
 - Dynamic forms and interactive elements using JavaScript.
 - Tailwind CSS for responsive and accessible design.
 - Axios for seamless API integration.
4. **Integration and Documentation**
 - Swagger UI for real-time API testing and debugging.
 - Email integration for notifications and password resets.
5. **Testing and Reliability**
 - Comprehensive unit and integration tests using PHPUnit.

Technology Stack

- **Backend Framework:** Laravel
- **Frontend Tools:** JavaScript, Tailwind CSS, Axios
- **Authentication:** Laravel Passport (OAuth2/JWT Tokens)
- **Database:** MySQL
- **Testing:** PHPUnit
- **Interactive Documentation:** Swagger UI

Running the Application

- **Backend:**
 - Start the Laravel application:

```
php artisan serve
```

- The default backend URL is:

```
http://localhost:8000
```

- **Frontend:**
 - Open the HTML file in a browser or use a local server.
 - Ensure the backend API is running and accessible.

Base URL

All API endpoints are relative to the base URL:

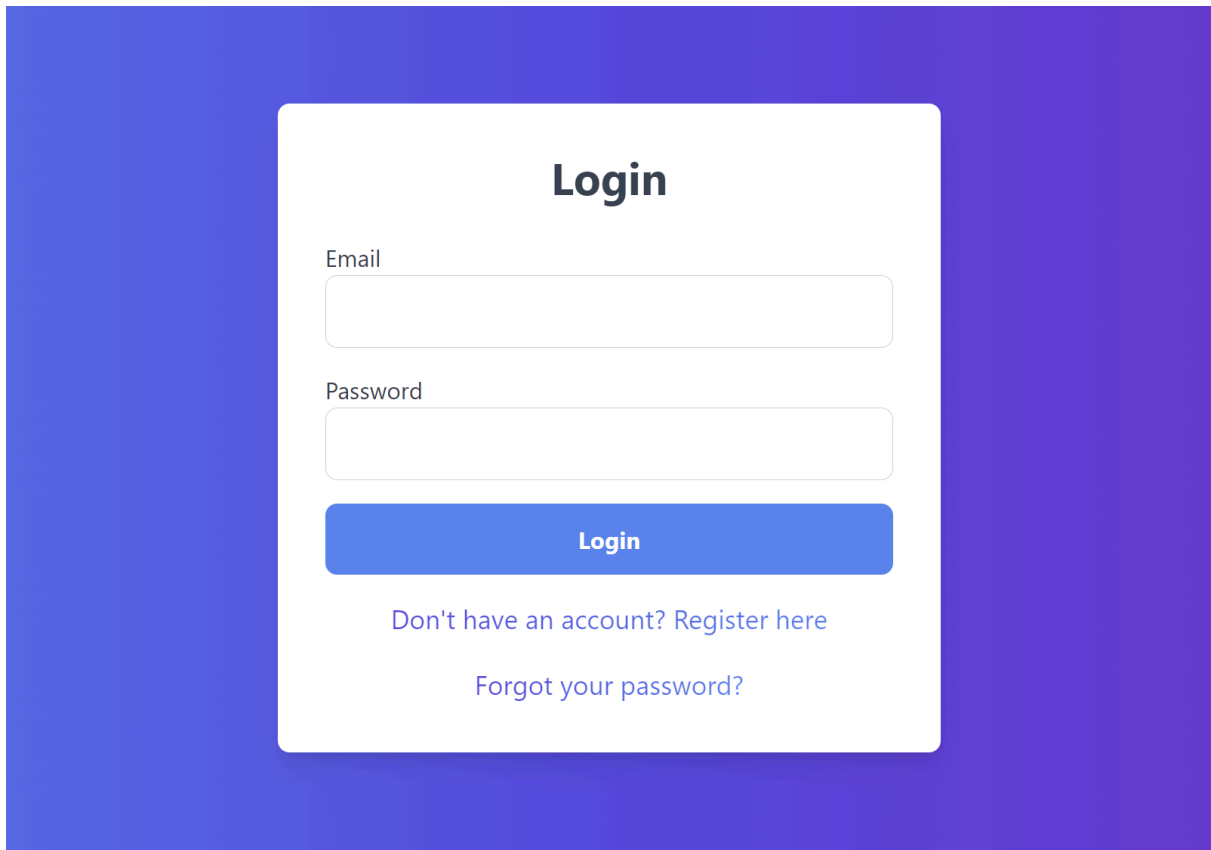
```
http://localhost:8000/api
```

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.

A login form is centered on a blue gradient background. The form is a white rounded rectangle with the title "Login" at the top. Below the title are two input fields: "Email" and "Password". A blue "Login" button is positioned below the password field. At the bottom of the form, there are two links: "Don't have an account? Register here" and "Forgot your password?".

Login

Email

Password

Login

[Don't have an account? Register here](#)

[Forgot your password?](#)

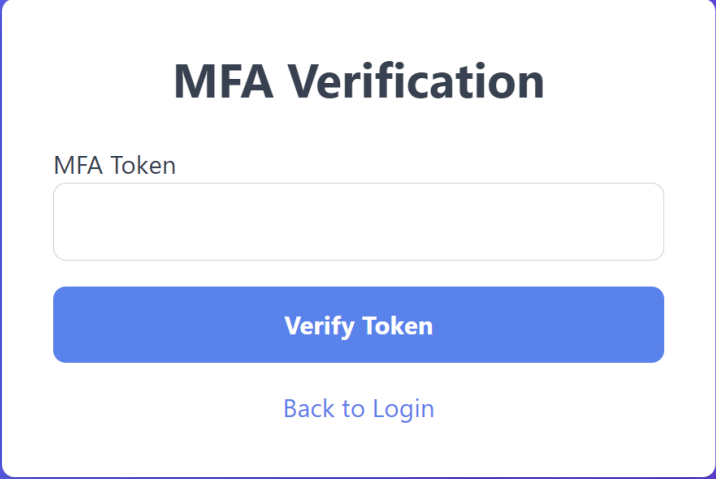
Register

- **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.

A screenshot of a web form titled "MFA Verification" centered on a blue gradient background. The form is a white rounded rectangle containing a label "MFA Token" above a text input field. Below the input field is a blue button with the text "Verify Token". At the bottom of the form is a blue link that says "Back to Login".

MFA Verification

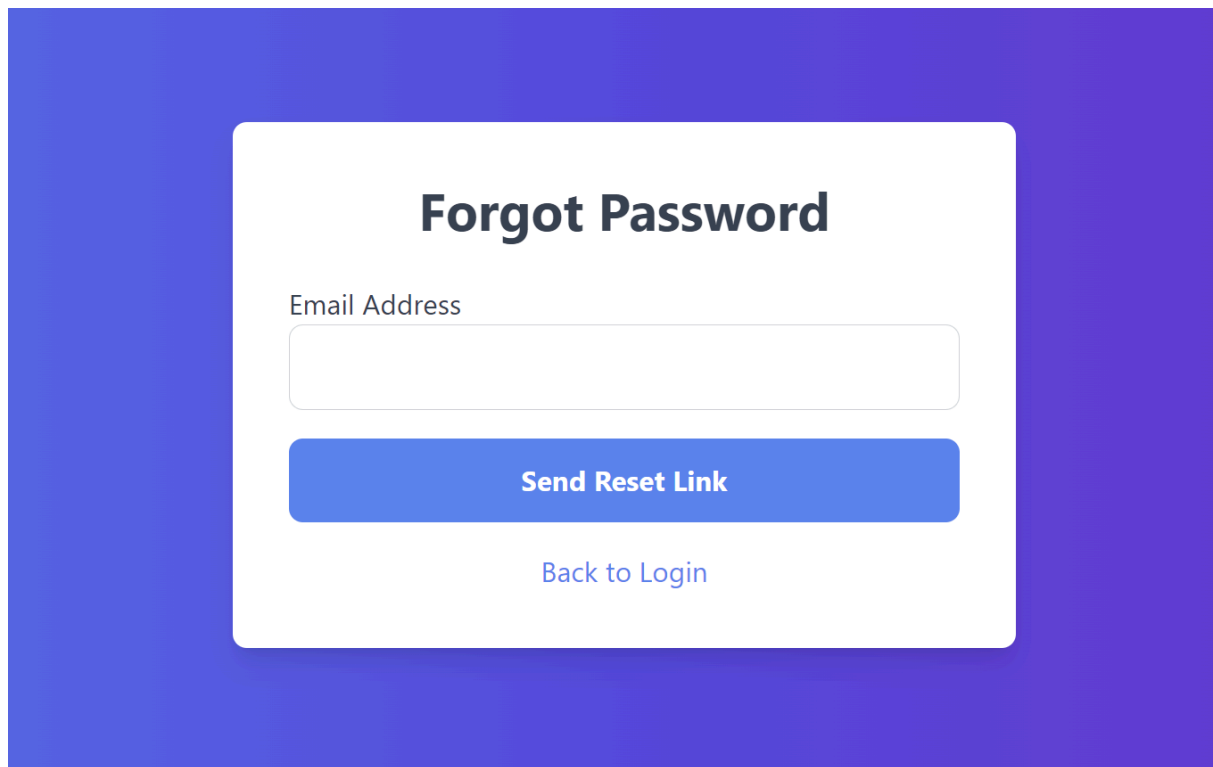
MFA Token

Verify Token

[Back to Login](#)

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

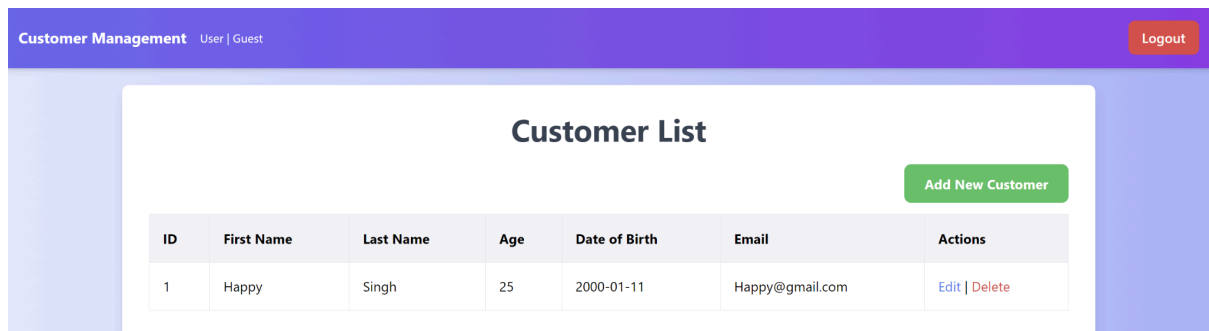
- **Endpoint:** POST /logout
- **Description:** Logs out the authenticated user and invalidates the access token.
- **Authentication Required:** Yes, a valid bearer token is required.
- **Response:** A success message indicating the user has been logged out.

Customer Management API

The **Customer Management** 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.



The screenshot shows a web application interface for 'Customer Management'. At the top, there is a purple header bar with the text 'Customer Management' and 'User | Guest' on the left, and a 'Logout' button on the right. Below the header, the main content area has a light blue background. In the center, there is a white box titled 'Customer List'. To the right of this box is a green button labeled 'Add New Customer'. Below the title, there is a table with the following data:

ID	First Name	Last Name	Age	Date of Birth	Email	Actions
1	Happy	Singh	25	2000-01-11	Happy@gmail.com	Edit Delete

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.

Customer Management

User | Guest

Logout

Add New Customer

First Name

Last Name

Age

Date of Birth

mm/dd/yyyy

Email

Add Customer

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
```

```
Tests\Unit\AuthControllerTest
✓ login with valid credentials          34.63s
✓ login with invalid credentials        1.01s
✓ login validation errors               1.68s
✓ mfa verification with valid token     5.41s
✓ mfa verification with invalid token   1.05s
✓ logout                               2.26s

Tests\Unit\CustomerControllerTest
✓ list all customers                   2.06s
✓ show customer                       1.01s
✓ show non existent customer          1.18s
✓ create customer                     1.39s
✓ update customer                     1.17s
✓ update non existent customer         1.09s
✓ delete customer                     1.13s
✓ delete non existent customer         1.22s

Tests\Unit\ExampleTest
✓ that true is true                   0.06s

Tests\Feature\ExampleTest
✓ the application returns a successful response 1.91s

Tests: 16 passed (35 assertions)
Duration: 68.21s

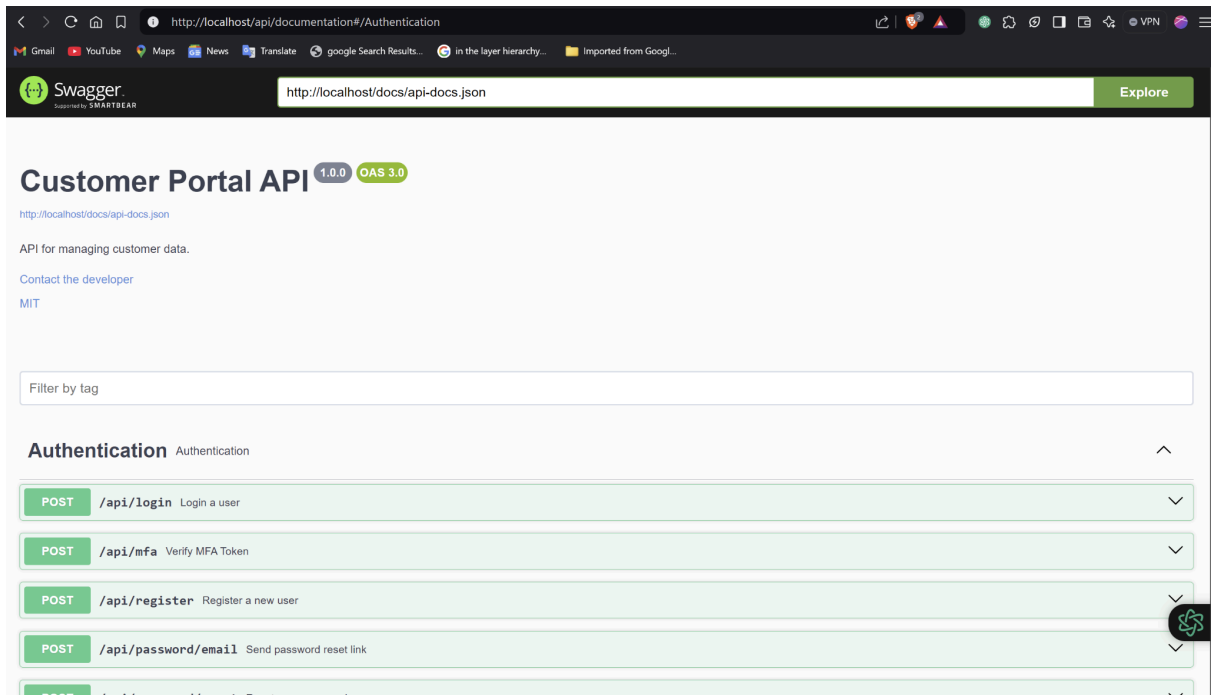
sukhpal@Happy:/mnt/d/laravel project/customer-porta$
```

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.

Frontend Integration

The frontend for the Customer Portal is built using:

- **JavaScript:** Handles dynamic content and API calls.
- **Tailwind CSS:** Ensures responsive and accessible design.
- **Axios:** Simplifies HTTP requests to the API.

Key Features

- **Responsive Design:** Tailwind CSS enables a mobile-first approach.
- **Dynamic Interactions:** JavaScript ensures smooth user interactions and validations.
- **API Integration:** Axios handles secure and efficient communication with the backend.

Running the Frontend

- Ensure the backend is running at `http://localhost:8000`.
- Open the frontend HTML file in a browser or serve it using a local server.
- Update the Axios base URL in the JavaScript configuration if needed.

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.