Laravel Admin Portal

A simple admin portal built with Laravel that includes customer and invoice management with API endpoints.

Features

- User authentication (login/logout)
- Admin dashboard with navigation
- Customer management (list, create, edit)
- Invoice management (list, create, edit)
- API endpoints for customers and invoices

Requirements

- PHP >= 8.0
- Composer
- MySQL or another database supported by Laravel
- Node.js & npm (optional, for asset compilation)

Installation

1. Clone the repository:

```
git clone https://github.com/yourusername/admin-portal.git
cd admin-portal
```

2. Install PHP dependencies:

```
composer install
```

3. Copy the example env file and make the required configuration changes in the .env file:

```
cp .env.example .env
```

4. Configure your database connection in the .env file:

```
DB_CONNECTION=mysql
```

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=admin_portal
DB_USERNAME=your_username
DB_PASSWORD=your_password
```

5. Generate an app encryption key:

```
php artisan key:generate
```

6. Run the database migrations and seed the database:

```
php artisan migrate --seed
```

7. Start the local development server:

```
php artisan serve
```

8. You can now access the application at http://localhost:8000

Default Login Credentials

- Username: admin
- Password: password

Project Structure

- **app/Models**: Contains the database models (User, Customer, Invoice)
- **app/Actions**: Contains Action classes to list and “add new” by API
- **app/Http/Controllers**: Contains the application controllers
- **app/Http/Controllers/API**: Contains API controllers
- **resources/views**: Contains the Blade view files
- **routes/web.php**: Contains the web routes
- **routes/api.php**: Contains the API routes
- **database/migrations**: Contains the database migrations
- **database/seeders**: Contains the database seeders

API Endpoints

The API uses Laravel Sanctum for authentication. You need to obtain an API token by logging in.

Authentication

- **POST /api/login**: Login and get an access token
    - Parameters: username, password
    - Returns: user object and access token
- **POST /api/logout**: Logout (requires authentication)
    - Headers: Bearer token
    - Returns: success message

Dashboard

- **GET /api/dashboard-data**: Get all customers and invoice (requires authentication
    - Headers: Bearer token
    - Returns: List of customers and invoice
- **POST /api/dashboard-data**: Create a new customer, invoices (requires authentication
    - Headers: Bearer token
    - Parameters: type(customer or invoice), name, email,phone address. For add new customer
    - Parameters: type(customer or invoice), customer_id (required), date (required), amount (required), status (required: unpaid/paid/cancelled) for the invoice
    - Returns: Created customer object or created invoice object

Testing the API with Postman

1. Make a POST request to http://localhost:8000/api/login with:

```
{
   "username": "admin",
   "password": "password"
}
```

2. Copy the access token from the response.

3. For subsequent API requests, include the token in the Authorization header:

Authorization: Bearer YOUR_TOKEN_HERE

2. Make a POST request to http://localhost:8000/api/logout with:

   Copy the access token from the response.

3. Make a GET request to http://localhost:8000/api/dasboard-data?modules[]=Customer&modules[]=Invoice with the access bearer token to fetch all the customer and the invoice data. Add new modules to this url to get other modules.

4. Make a Post Request to http://localhost:8000/api/dasboard-data with the access bearer token and parameters :

```
{
    "type": "customer",
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "1234567890",
    "address": "123 Main St"
}
```

For adding new customers.

```
{
    "type": "invoice",
    "customer_id": 2,
    "date": "2025-05-20",
    "amount": 1800,
    "status": "unpaid"
}
```

For adding a new invoice.