Author: Mathew Kamande Macharia

MAMA MBOGA PROJECT DOCUMENTATION

Overview

Mama Mboga is a PHP and MySQL web application designed to manage credit-based customer relationships. It facilitates customer management, sales and payment tracking, and automated payment reminders via email. The application features distinct user roles for vendors and administrators. It is currently deployed and accessible at: http://169.239.252.114:341/~mathew.macharia/

Technology Stack

The application is built on a standard LAMP-like stack. The backend utilizes PHP 8.x with PDO for database interactions. Data is stored in a MySQL/MariaDB database. The frontend is constructed with HTML5, custom CSS, and Bootstrap 5 for layout and responsive design, supplemented with minimal JavaScript for basic interactions. The email notification system is implemented using the SendGrid API (v3) with a dynamic HTML template. The application is hosted on an Apache web server.

Project Structure

The codebase is organized into logical directories. The root contains the main entry points (index.php, auth.php) and a helper script for local development. Configuration files for the database and SendGrid are located in the config/ directory, utilizing environment variables for sensitive data. Database schema is defined in db/schema.sql. Reusable page components and helper functions are in the includes/ folder. All static assets (CSS, JavaScript, images) are under assets/. Core application logic for processing forms and actions is contained within the actions/ directory. The vendor/ and admin/ directories house the respective dashboards and role-specific interfaces. The HTML template for email reminders is stored in templates/.

Authentication and Access

For evaluation and demonstration purposes, a default administrative account is available. The credentials are:

1. Email: admin@example.com
2. Password: Admin@123

   Vendor accounts can be created through the public registration form

Database Design

The application's data model is built around four core tables. The users table stores authentication details for all users (admins and vendors), including their role and active status. The customers table holds information for each vendor's clients, including a running current_balance. Financial transactions are recorded in two tables: sales for credit purchases and payments for received payments. Foreign key relationships ensure that all sales and payments are linked to both a customer and the responsible vendor.

Core Application Workflows

3. User Authentication: The registration process validates input, enforces password strength, and ensures unique usernames and emails before storing a hashed password. The login process verifies credentials and user active status, creating a session upon success.
4. Vendor Operations: The vendor dashboard displays key metrics, such as total outstanding debt and a list of top debtors. Vendators can add customers, record new credit sales (which increase the customer's balance), and log payments (which decrease the balance). These updates to the current_balance field are handled within database transactions to ensure data consistency.
5. Email Reminders: The system allows vendors to send payment reminders via email. The reminder action validates customer ownership, prepares personalized data, and sends requests to the SendGrid API using a pre-designed dynamic template. NOTE: if you do not receive the email, check the spam folder of the test customer email that you added.

Administrative Features

Administrators have a dedicated dashboard showing system-wide metrics. A key function is vendor management, which includes searching vendors by username or email, viewing their status, and performing actions. Administrators can directly edit a vendor's contact information (username/email) and toggle their account's active status. Deactivated vendors cannot log into the system.

Security and Validation Measures

Several measures are implemented to protect the application. User input is sanitized using custom helper functions. Access to all protected pages is gated by role-checking functions. Passwords are hashed using PHP's `password_hash()` function and verified with `password_verify()`; plain-text passwords are never stored. Essential server-side validation is performed on all form submissions.

Limitations and Future Considerations

The current implementation has several areas identified for future enhancement. These include the absence of CSRF protection tokens on forms, no pagination for large data sets, and a simple trust model for administrators without granular permissions. A planned SMS reminder feature was not implemented due to cost constraints; email reminders were selected as a viable alternative using SendGrid's free tier. These features represent potential avenues for further development.