PASHA Benefits Portal - Implementation Guide

This document provides a comprehensive guide to the PASHA Benefits Portal implementation using the LAMP (Linux, Apache, MySQL, PHP) stack.

System Overview

The PASHA Benefits Portal is a web application that allows:

1. **Public Users** to browse benefit offers and verify membership status
2. **Admin Users** to manage members, partners, offers, and system settings
3. **Partner Users** to verify member status and manage their profiles

Technology Stack

- **Linux**: Operating system for hosting the application
- **Apache**: Web server with mod_rewrite for clean URLs
- **MySQL**: Database for storing all application data
- **PHP**: Server-side programming language (PHP 7.4+ recommended)

Directory Structure

```
pasha-benefits/
├── config/              # Configuration files
│   ├── config.php       # Main configuration file
│   └── database.php     # Database connection settings
├── public/              # Publicly accessible files
│   ├── index.php        # Main entry point
│   ├── assets/          # CSS, JS, images
│   └── .htaccess        # Apache configuration
├── app/                 # Application code
│   ├── controllers/     # Controller classes
│   ├── models/          # Database models
│   ├── views/           # Templates and UI
│   ├── middleware/      # Authentication middleware
│   └── helpers/         # Utility functions
├── data/                # Data storage (logs, cache)
│   ├── logs/            # Application logs
│   └── cache/           # Cached data
├── vendor/              # Third-party libraries (via Composer)
└── scripts/             # Deployment and maintenance scripts
```

Key Components

Models

1. **MemberModel**: Manages member data and verification
2. **OfferModel**: Handles benefit offers and categories
3. **PartnerModel**: Manages partner organizations
4. **UserModel**: Handles user authentication and management
5. **ActivityLogModel**: Tracks user activity for auditing

Controllers

1. **HomeController**: Public homepage and main navigation
2. **AuthController**: User authentication (login, logout, password reset)
3. **MemberController**: Member verification functionality
4. **OfferController**: Public offer browsing and details
5. **AdminController**: Admin dashboard and management functions
6. **PartnerController**: Partner portal functionality
7. **PageController**: Static pages (about, contact)
8. **ReportController**: Generate and export reports

Views

1. **Public Views**: Homepage, offer listings, verification forms
2. **Admin Views**: Dashboard, member/offer/partner management
3. **Partner Views**: Dashboard, verification, profile management
4. **Authentication Views**: Login, password reset forms

Database Schema

The application uses several relational tables to store data:

1. **members**: Store member organization information
2. **users**: System users (admin, staff, partner)
3. **partners**: Partner organizations offering benefits
4. **offers**: Benefit offers with details
5. **verifications**: Log of membership verifications
6. **activity_logs**: Audit trail of user actions
7. **settings**: System-wide configuration settings

Core Features

Public Features

1. **Offer Browsing**: View available benefits, filter by category
2. **Member Verification**: Verify membership status
3. **Informational Pages**: About and contact information

Admin Portal

1. **Dashboard**: Overview of key metrics and statistics
2. **Member Management**: Add, edit, and deactivate members
3. **Offer Management**: Create and manage benefit offers
4. **Partner Management**: Add and manage partner organizations
5. **User Management**: Create and manage system users
6. **Reports**: Generate and export various reports
7. **Activity Logging**: Comprehensive audit trail
8. **Settings**: Configure system behavior and appearance

Partner Portal

1. **Member Verification**: Verify PASHA membership status
2. **Profile Management**: Update partner information
3. **Verification History**: View past verification activities

Security Measures

1. **Authentication**: Secure login with password hashing
2. **Role-Based Access Control**: Different permissions by user role

3. **CSRF Protection**: Prevent cross-site request forgery
4. **Input Validation**: Server-side validation of all inputs
5. **Prepared Statements**: Protection against SQL injection
6. **Activity Logging**: Audit trail of all important actions
7. **Password Reset**: Secure password reset functionality
8. **Session Management**: Secure handling of user sessions

Installation Guide

1. **Prerequisites**:
   - Linux server with Apache (mod_rewrite enabled)
   - MySQL 5.7+ or MariaDB 10.2+
   - PHP 7.4+ with required extensions
   - Composer (for dependencies)
2. **Database Setup**:
   - Create a new MySQL database
   - Import the database schema from `scripts/db-schema.sql`
   - Create an admin user: `php scripts/create-admin.php`
3. **Application Setup**:
   - Clone the repository to your server
   - Run `composer install` to install dependencies
   - Copy `config/config.example.php` to `config/config.php` and customize
   - Copy `config/database.example.php` to `config/database.php` and set credentials
   - Ensure the `data/` directory is writable by the web server
   - Configure Apache to point to the `public/` directory

- Ensure mod_rewrite is enabled for clean URLs

4. **Configuration**:
   - Update settings in the admin panel
   - Add initial members, partners, and offers
   - Configure email settings if needed

Deployment Recommendations

1. **Production Environment**:
   - Enable HTTPS and configure security headers
   - Set up proper file permissions
   - Configure MySQL for optimal performance
   - Set up regular backups

2. **Performance Optimization**:
   - Enable PHP OpCache
   - Configure browser caching for static assets
   - Consider a CDN for images and other media
   - Set up database indexing for improved query performance

3. **Maintenance**:
   - Set up regular database backups
   - Monitor error logs
   - Update dependencies regularly
   - Consider setting up monitoring for uptime alerts

Customization Options

1. **Branding**:
   - Update logo and site name in settings
   - Customize CSS styles for your organization
   - Modify email templates
2. **Features**:
   - Enable/disable verification functionality
   - Configure offer categories
   - Customize user roles and permissions

Troubleshooting

1. **Common Issues**:
   - Permissions problems: Ensure proper directory permissions
   - Database connection issues: Check credentials and server availability
   - URL rewriting issues: Verify Apache configuration
2. **Logging**:
   - Check application logs in `data/logs/`
   - Check Apache error logs
   - Enable debugging in development environments

Further Development

1. **Potential Enhancements**:
   - API for mobile application integration
   - Email notifications for new offers
   - Member login area for exclusive content

- Advanced analytics and reporting
- Integration with CRM or membership management systems

Support

For questions, issues, or feature requests, please contact the development team.