Automated Debt Collection System Documentation

# Acknowledgement

Primarily, we are grateful to the almighty God for the strength, health, and ability to successfully complete of this project

We would like to thank the DMI – St. John the Baptist University Malawi, for providing us the opportunity to do the project work as part of our curriculum. I also like to thank our university founder **Rev. Fr. Dr. J.E. ARULRAJ** founder of MMI and DMI

We are also thankful to our director of international operations **Dr. IGNATIUS A. HERMAN –**Director of Education for having provided these facilities to carry on my project. We accordially give thanks to **Rev.Fr.G. Sundar**, Secretory university council, **Dr.Jerin Leno**, Vice Chancellor, **Rev.Sr. J Rani**, Management Representative, **Dr. Njoloma**, The Principal, **Ms. Agnes Msonda** Campus Dean, Ms**. Fanny**, Head of Computer Science and Information Technologyfor their kind help during out project work by providing an opportunity to enhance our career.

We also sincerely thank **Dr.G.Glorindal**, Dean of Research Computer Science and Information Communication Technology, our internal guide and staff members of the department for their valuable support to finish our project

Finally yet importantly, we thank our family members and all of those who in one way or another have given us a hand of support throughout the time we have been working on this project.

Table of Contents

[Acknowledgement 1](#_Toc137174860)

[Introduction 2](#_Toc137174861)

[System Requirements 2](#_Toc137174862)

[Architecture Overview 2](#_Toc137174863)

[Database Design 2](#_Toc137174864)

[Frontend Development 2](#_Toc137174865)

[Backend Development 2](#_Toc137174866)

[Deployment 2](#_Toc137174867)

[Conclusion 2](#_Toc137174868)

# Introduction

The Automated Debt Collection System is designed to streamline and automate the manual debt recovery process, facilitating efficient communication and information management between lenders and debt collectors. This system replaces the traditional approach where lenders and debt collectors engage in direct communication to discuss loan details, debtor information, security of the loan, documentation, commission agreements, and operating costs.

By leveraging this system, lenders can easily register their details and input relevant case information, which can later be assigned to debt collectors. The system allows debt collectors to take action by sending automated emails to the debtors directly through the platform. Additionally, every action performed by the debt collector is logged within the system, ensuring a comprehensive record of all interactions. The system also notifies the lender of the debt collector's actions, keeping them informed throughout the process.

One of the core benefits of this web application is the digital storage of information, eliminating the need for manual record-keeping. Lenders can access and retrieve the stored information at any time by utilizing the search functionality, enabling quick and easy retrieval of case details and related documents.

The primary objectives of the Automated Debt Collection System are:

* Automating and streamlining the debt recovery process.
* Enabling efficient communication between lenders and debt collectors.
* Storing digital records and facilitating easy retrieval of information.
* Providing transparency and accountability through comprehensive action logging.
* Improving the overall efficiency and effectiveness of the debt collection workflow.

The target audience for this system includes lenders, debt collectors, and administrators responsible for managing and overseeing the debt collection process. By leveraging the system, these stakeholders can enhance their operations, reduce manual effort, and optimize the debt recovery workflow.

# System Requirements

## 2.1 Hardware Requirements

* Computing device
* Server infrastructure capable of hosting a web application
* Adequate storage capacity for storing user data and documents
* Sufficient processing power to handle concurrent user requests
* Reliable internet connectivity to ensure uninterrupted access to the system

## Software Requirements

* Operating System: Linux, Windows, or macOS
* Web Server: Apache, Nginx, or Microsoft IIS
* PHP version 7.4 or higher
* MySQL database server version 5.7 or higher
* SMTP server for sending emails
* SSL certificate for secure data transmission (HTTPS)
* Text editor or integrated development environment (IDE) for code development
* Version control system (e.g., Git) for source code management
* Deployment tools for deploying the application to the production environment
* Firewall and security measures to protect against unauthorized access

## 2.3 Supported Browsers

The Automated Debt Collection System should be compatible with the following web browsers:

* Google Chrome (latest version)
* Mozilla Firefox (latest version)
* Safari (latest version)
* Microsoft Edge (latest version)
* Internet Explorer 11 (or gracefully degrade functionality)

## 2.4 Dependencies

* PHP frameworks and libraries for web development (Laravel)
* JavaScript frameworks and libraries for frontend development (React)
* SMTP library or service for sending emails (PHPMailer)
* MySQL database driver for PHP (MySQLi)
* CSS frameworks for frontend styling (Bootstrap)
* Validation libraries for input validation and sanitization (Laravel Validation)

## 2.5 Security Requirements

* Strong encryption for sensitive user data (e.g., passwords) using industry-standard hashing algorithms (e.g., bcrypt)
* Proper input validation and sanitization to prevent SQL injection, cross-site scripting (XSS), and other security vulnerabilities
* User authentication and authorization mechanisms to ensure only authorized users can access the system
* Role-based access control (RBAC) to define different levels of system access for lenders, debt collectors, and administrators
* Secure transmission of data over HTTPS to protect against eavesdropping and tampering
* Regular backups of the database to prevent data loss
* Regular software updates and patches to address security vulnerabilities

These system requirements provide a foundation for developing and deploying the Automated Debt Collection System.

# Architecture Overview

The Automated Debt Collection System follows a typical client-server architecture, where the frontend interacts with the user through a web browser, and the backend handles the processing of requests, data storage, and business logic. The system architecture consists of the following components:

## 3.1 Client-Side (Frontend)

The client-side of the system is responsible for presenting the user interface and handling user interactions. It is built using HTML, CSS, and JavaScript, and interacts with the backend via APIs. The frontend is designed to be user-friendly and intuitive, providing lenders and debt collectors with an efficient and seamless experience. Key components include:

* User Interface (UI): The UI components allow users to register, log in, manage cases, view case details, and perform actions. It includes forms, tables, search functionality, and notification mechanisms.
* User Authentication: The frontend handles user authentication, securely sending login credentials to the backend for verification, and managing user sessions.
* API Communication: The frontend communicates with the backend via RESTful APIs to retrieve and send data. It handles API requests, responses, and error handling.

## 3.2 Server-Side (Backend)

The server-side of the system consists of the backend components responsible for processing requests, managing data, and implementing business logic. It is developed using PHP and interacts with the frontend, database, and external services. Key components include:

* Web Server: The web server (e.g., Apache, Nginx) receives HTTP requests from the frontend, routes them to the appropriate endpoints, and serves static files.
* API Endpoints: The backend provides RESTful API endpoints to handle various functionalities such as user registration, authentication, case management, action logging, and email sending.
* Business Logic: The backend implements the necessary business logic, including validating user inputs, processing actions, performing database operations, and generating email notifications.
* Database Interaction: The backend communicates with the MySQL database to store and retrieve user data, case information, action logs, and other relevant data. It uses SQL queries to interact with the database.

## 3.3 Database

The database component stores the system's data using a MySQL relational database. It maintains tables such as **lenders**, **cases**, **debt\_collectors**, and **actions**. The database stores lender and debt collector details, case information, action logs, and other relevant data. The backend interacts with the database through SQL queries to retrieve, insert, update, and delete data.

## 3.4 External Services

The system may rely on external services to enhance its functionality. For example, an SMTP service can be used for sending emails to debtors and notifications to lenders. These external services are accessed by the backend through appropriate APIs or libraries.

## 3.5 Security Considerations

To ensure the security of the system, several measures should be implemented, including:

* Properly validating and sanitizing user inputs to prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS) attacks.
* Applying appropriate encryption techniques (e.g., bcrypt) to store sensitive data, such as passwords.
* Implementing user authentication and authorization mechanisms to control access to the system's functionalities.
* Enforcing secure communication over HTTPS to protect data transmission between the client and server.
* Regularly updating the system with security patches and fixes to address any vulnerabilities.

The architecture outlined above provides a high-level overview of the Automated Debt Collection System. It is designed to ensure efficient communication, secure data storage, and streamlined debt collection processes.

# Database Design

# Frontend Development

# Backend Development

# Deployment

# Conclusion