**NATIONAL INSTITUTE OF TECHNOLOGY**



**DEPARTMENT OF COMPUTING AND COMMUNICATION TECHNOLOGY (CCT)**

**TITLE: CHURCH MANAGEMENT SYSTEM**

**SUPERVISOR NAME: EXAUD NOEL KITOMARY**

**STUDENT NAME: EMMANUEL HENRY MSAKI**

**REGISTRATION NUMBER: NIT/BCS/2022/460**

**DATE OF SUBMISSION: 27/02/2025**

# **ABSTRACT**

Managing church operations efficiently is crucial for fostering growth, transparency, and engagement within religious communities. Traditional church management methods often rely on manual record-keeping, leading to inefficiencies, data loss, and difficulty in tracking member contributions and church activities. This project proposes the development of a Church Management System (CMS) to streamline administrative tasks such as member registration, event scheduling, and communication within the church.

The primary objectives of this system are to design a church dashboard, to design notification alerts system, and to design the module to collect user information and authenticate users. The CMS will be a web-based application built using HTML, CSS, JavaScript (React), bootstrap,a secure database (MySQL),python and Django. The system will integrate features such as role-based access control.

A structured Agile methodology will guide the development, ensuring iterative testing and feedback incorporation. The expected outcome is a user-friendly, scalable, and secure system that simplifies church administration, enhances transparency, and improves member engagement. By digitizing church management, this project aims to provide a modern solution that aligns with technological advancements while meeting the unique needs of religious organizations.

Contents

[**ABSTRACT** 2](#_Toc191632160)

[**1. INTRODUCTION** 5](#_Toc191632161)

[**1.1 Background Information** 5](#_Toc191632162)

[**1.2 Problem Statement** 5](#_Toc191632163)

[**1.3 Project Objectives** 5](#_Toc191632164)

[**1.4 Scope of the Project** 5](#_Toc191632165)

[**1.5 Significance of the Project** 6](#_Toc191632166)

[**2. LITERATURE REVIEW** 7](#_Toc191632167)

[**2.1 Review of Existing Projects and Technologies** 7](#_Toc191632168)

[**2.2 Gaps and Limitations in Existing Solutions** 7](#_Toc191632169)

[**2.3 Summary of Key Findings** 7](#_Toc191632170)

[**3. METHODOLOGY** 8](#_Toc191632171)

[**3.1 System Design** 8](#_Toc191632172)

[**3.2 System Architecture & Data Flow** 8](#_Toc191632173)

[**3.3 Technologies Used** 12](#_Toc191632174)

[**3.4 Development Process** 12](#_Toc191632175)

[**3.5 Data Collection and Analysis** 12](#_Toc191632176)

[**4. SYSTEM REQUIREMENTS** 13](#_Toc191632177)

[**4.1 Hardware Requirements** 13](#_Toc191632178)

[**4.2 Software Requirements** 13](#_Toc191632179)

[**4.3 Network Requirements** 14](#_Toc191632180)

[**5. PROJECT SCHEDULE** 15](#_Toc191632181)

[**5.1 Timeline** 15](#_Toc191632182)

[**8. RISK MANAGEMENT** 17](#_Toc191632183)

[**8.1 Potential Risks and Challenges** 17](#_Toc191632184)

[**8.2 Risk Mitigation Strategies** 17](#_Toc191632185)

[**9. EXPECTED OUTCOMES** 18](#_Toc191632186)

[**9.1 Expected Results and Deliverables** 18](#_Toc191632187)

[**9.2 Performance Metrics & Success Criteria** 18](#_Toc191632188)

[**10. BUDGET** 19](#_Toc191632189)

[**10.1 Detailed Budget Breakdown** 19](#_Toc191632190)

[**10.2 Funding Source & Budget Management** 19](#_Toc191632191)

[**11. CONCLUSION** 20](#_Toc191632192)

[**11.1 Summary of Purpose and Objectives** 20](#_Toc191632193)

[**11.2 Reaffirming Expected Benefits and Significance** 20](#_Toc191632194)

[**12. REFERENCES** 21](#_Toc191632195)

# **1. INTRODUCTION**

## **1.1 Background Information**

Churches play a vital role in fostering spiritual growth and community engagement. However, many churches still rely on **manual or outdated record-keeping systems**, which pose challenges in managing member information, event scheduling, and overall administration. With increasing church membership and activities, there is a growing need for **a digital solution** to improve efficiency, transparency, and communication within religious organizations.

A **Church Management System (CMS)** provides an automated platform to handle administrative tasks, ensuring smooth operations and better service delivery to members. By leveraging modern technology, churches can streamline processes such as **membership management, sermons, podcast, articles and event coordination**, thereby reducing administrative workload and improving organizational efficiency.

## **1.2 Problem Statement**

Traditional church management methods often involve **paper-based records or basic spreadsheets**, which can lead to **data loss, inefficiencies, security risks, and difficulties in tracking finances and events**. This lack of an integrated system results in **poor communication, financial mismanagement, and difficulty in accessing historical records**.

This project seeks to **develop a web-based Church Management System** that will **automate and centralize church administrative tasks**, ensuring efficient member data management, financial transparency, and improved communication between church leaders and members.

## **1.3 Project Objectives**

The main objectives of this project are:

1. To design a module to collect user information’s and authenticate.
2. To design a notification alert system to notify a user through email about upcoming events.
3. To design a user friendly church dashboard.

## **1.4 Scope of the Project**

* Member registration.
* Event scheduling and notifications
* Communication system (announcements, emails)
* Role-based access control for administrators and members

## **1.5 Significance of the Project**

The Church Management System will **enhance operational efficiency**, **improve financial accountability**, and **foster better communication** within the church. By digitizing administrative processes, the system will **save time, reduce paperwork, and minimize errors**. Additionally, it will **promote transparency** in financial transactions, ensuring trust between church leaders and members.

This project is particularly beneficial for **growing churches** that require an organized system to handle increasing membership and financial transactions. In the long run, the system will **contribute to the church’s overall growth and sustainability** by providing a modern, scalable solution tailored to religious institutions' needs.

# **2. LITERATURE REVIEW**

## **2.1 Review of Existing Projects and Technologies**

Several **Church Management Systems (CMS)** have been developed to assist churches in managing their operations. Some widely used church management solutions include:

1. **Church Community Builder (CCB)** – A cloud-based platform offering membership management, donation tracking, and event scheduling.
2. **Breeze Church Management** – A lightweight, user-friendly CMS designed for small to mid-sized churches, providing membership tracking, financial management, and communication tools.
3. **Tithe.ly Church Management** – Focuses on **online giving**, integrating donation processing, member tracking, and church engagement tools.

## **2.2 Gaps and Limitations in Existing Solutions**

While existing church management systems provide **basic administration,** they exhibit several limitations:

1. **High Cost & Subscription Fees** – Many CMS platforms are commercial products, making them **expensive for small churches** with limited budgets.
2. **Complexity & Usability Issues** – Some systems have **steep learning curves** and require **technical expertise** to operate efficiently.
3. **Limited Customization** – Most existing platforms **lack flexibility** in adapting to unique church needs, especially for churches in **developing regions** with different structures.

## **2.3 Summary of Key Findings**

A review of existing church management solutions highlights the need for a **cost-effective, customizable, and secure system** tailored to the specific needs of churches, particularly in **developing regions like Africa**. This project aims to **bridge the gap** by providing a **web-based CMS** with:

* **Affordable, open-source deployment** to reduce costs.
* **User-friendly design** for easy adoption by non-technical users.

# **3. METHODOLOGY**

## **3.1 System Design**

The **Church Management System (CMS)** will be designed as a **web-based application** following a **three-tier architecture**, comprising:

1. **Presentation Layer (Frontend):**
   * A **responsive user interface (UI)** using bootstrap for smooth user interaction.
   * Bootstrap and CSS for enhanced design and accessibility.
2. **Business Logic Layer (Backend):**
   * **Python** for handling user requests, authentication, and processing data.
   * **Django framework**.
3. **Data Layer (Database):**
   * **MySQL (Relational Database)** for storing user data, financial transactions, and event details.
   * Data encryption and authentication mechanisms to ensure security and privacy.

## **3.2 System Architecture & Data Flow**

The system will follow a **Model-View-Controller (MVC)** architecture:

* **Users interact** with the frontend UI.
* The **backend** processes request and retrieves/stores data from the database.
* Data is displayed dynamically on the UI, ensuring **real-time updates** where necessary.

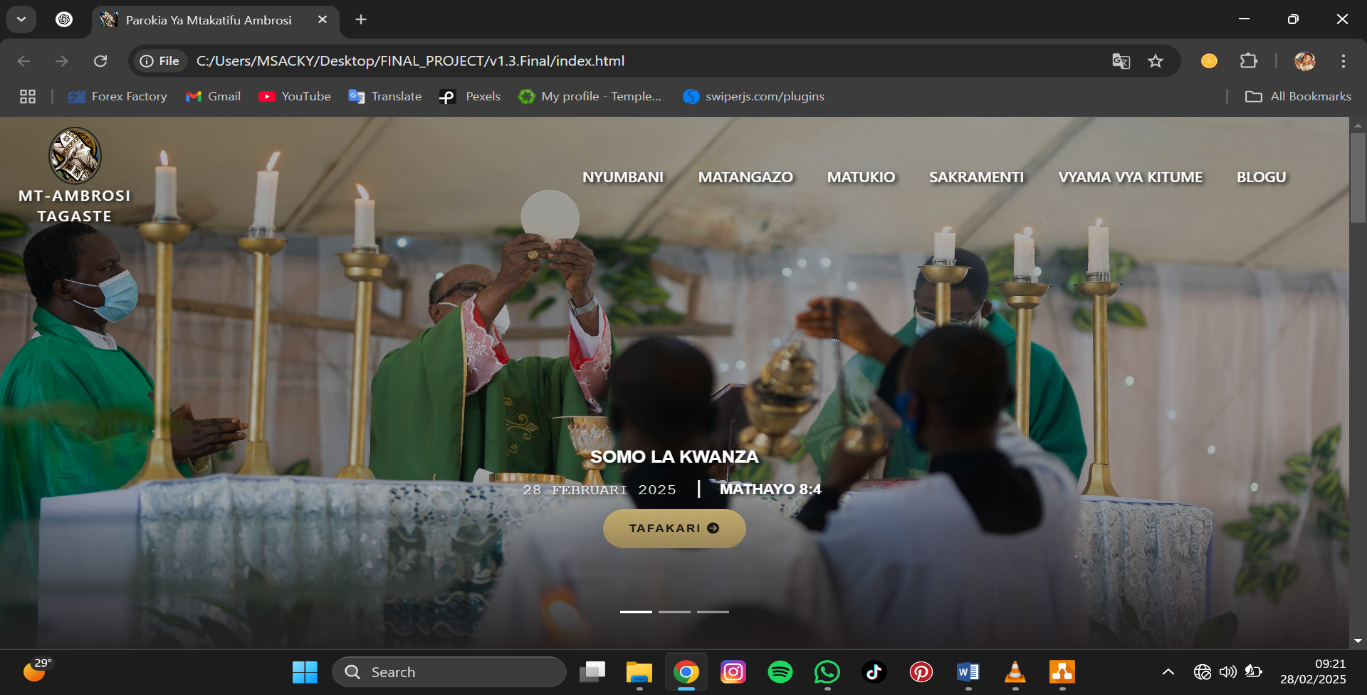


Figure 1. home page

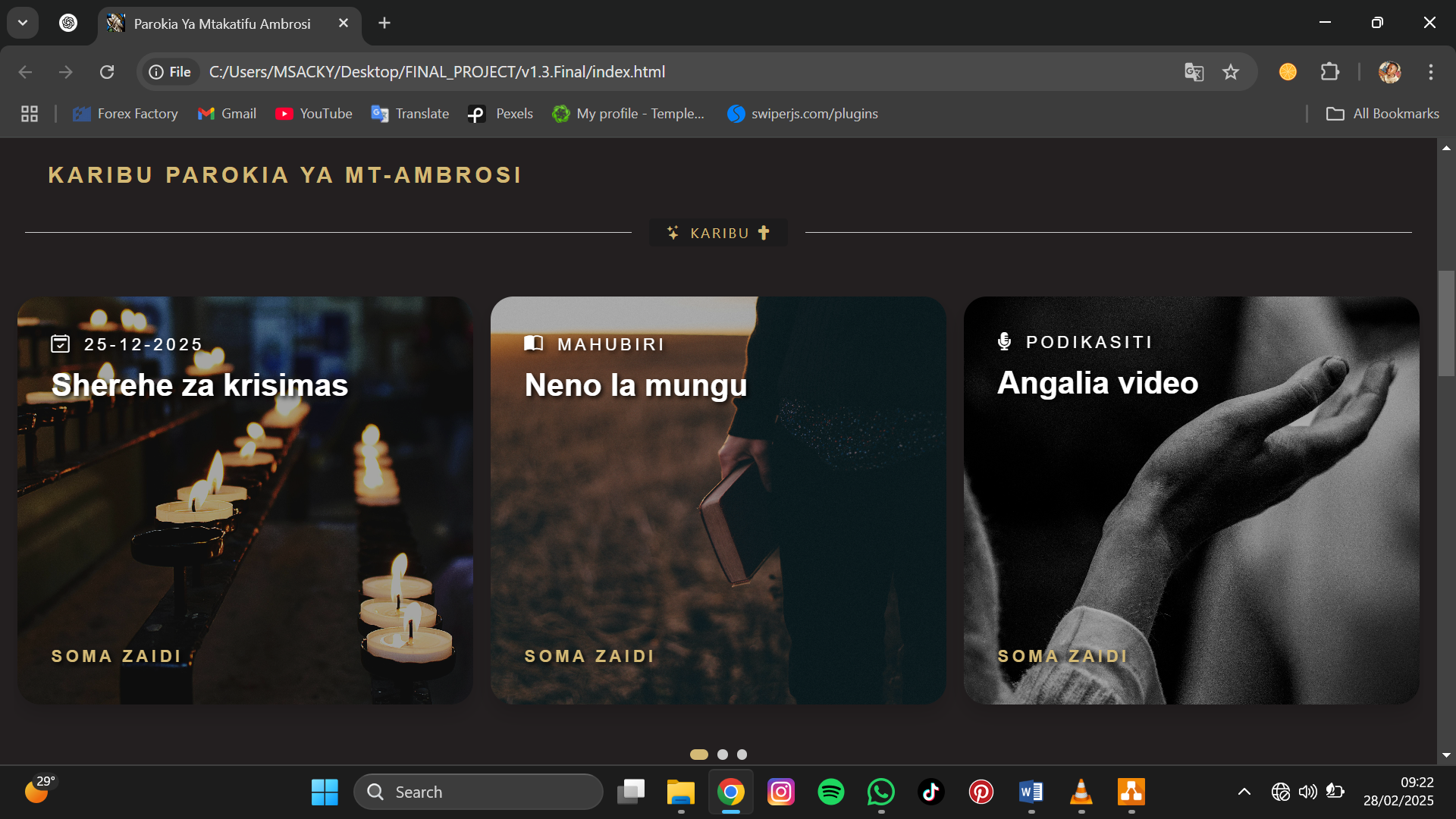


Figure 2. home interface

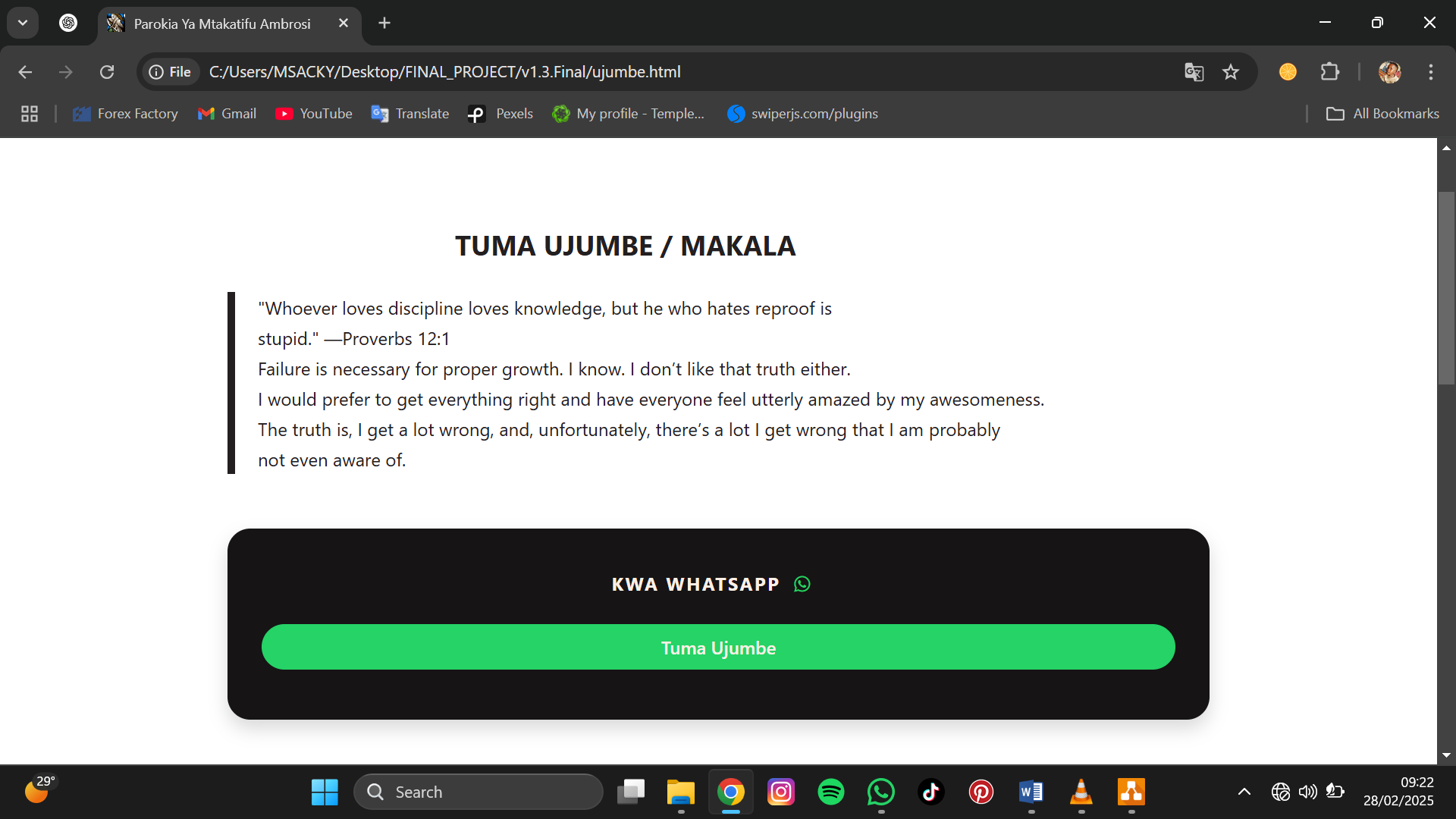


Figure 3. sending message interface

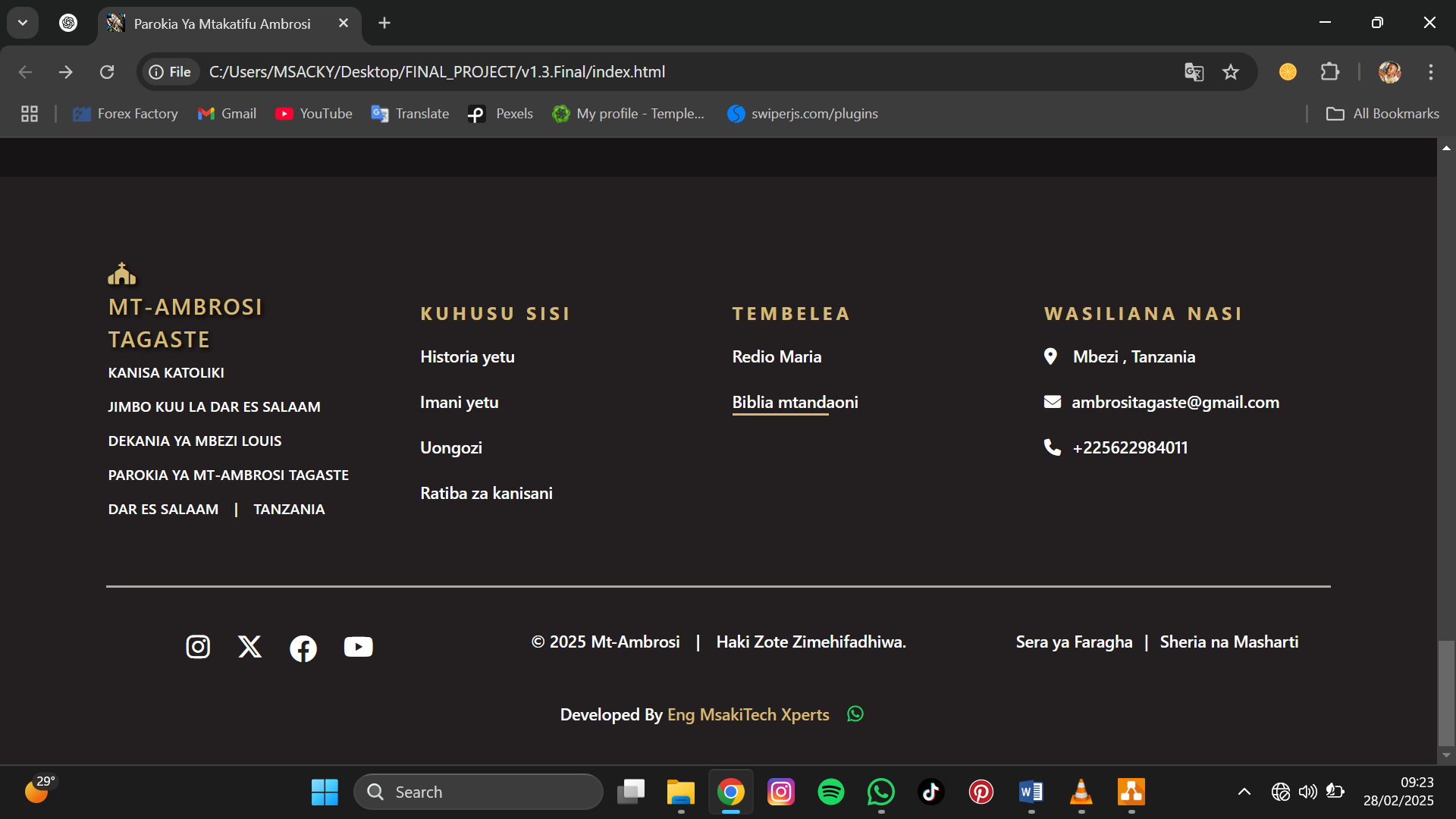


Figure 4. Footer Interface

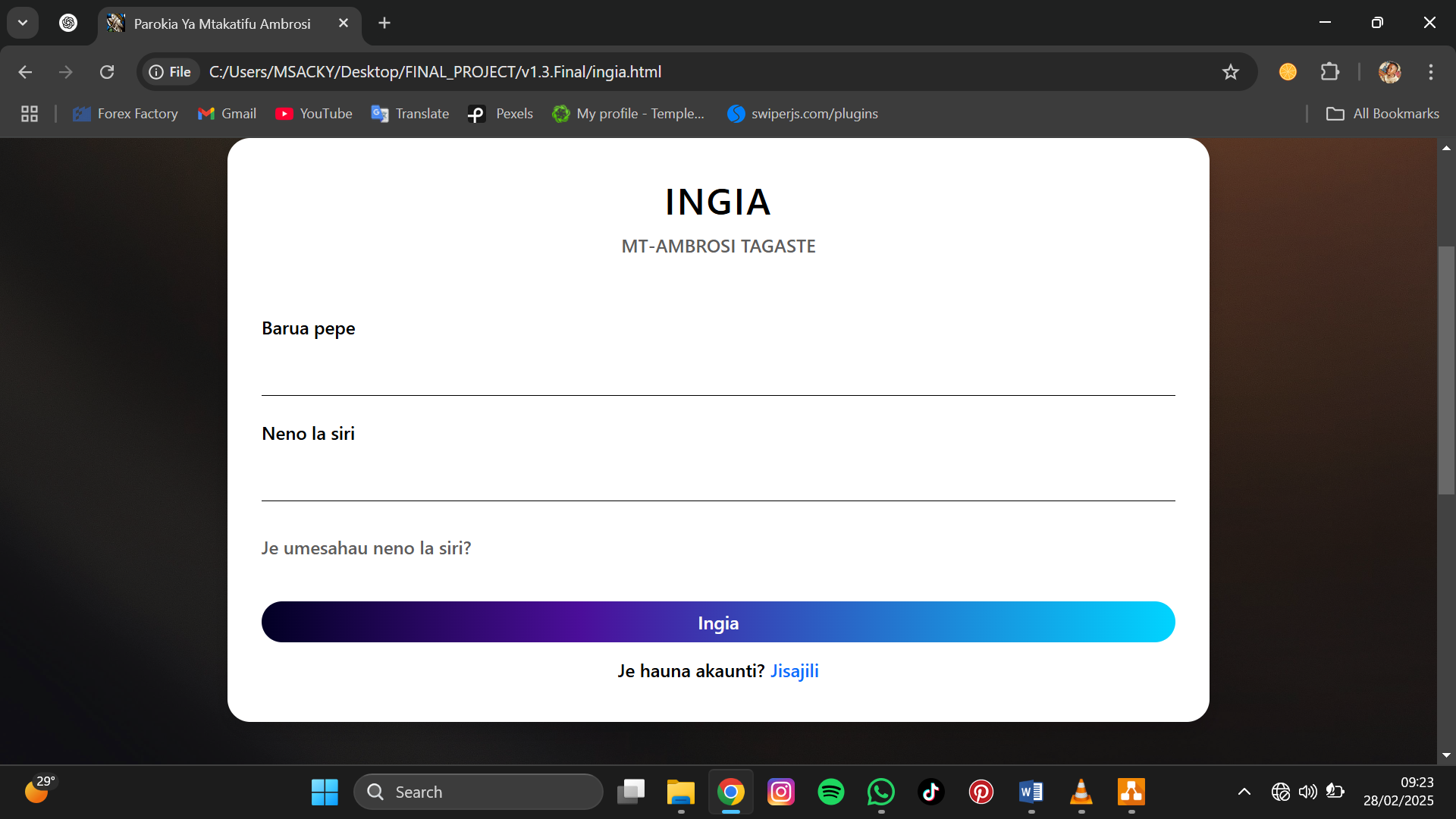


Figure 5. Login interface

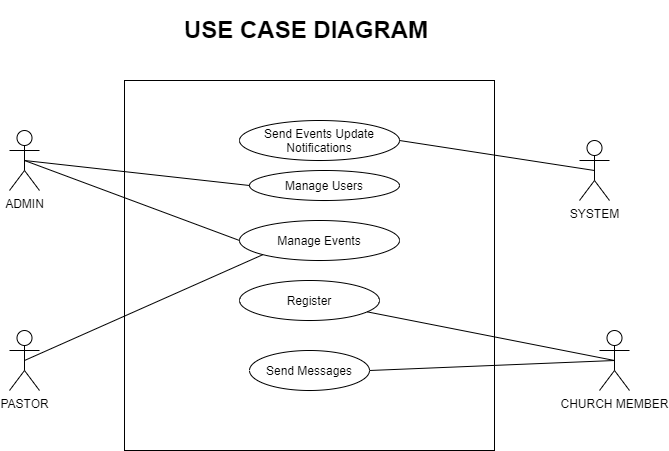


Figure 6. Use case Diagram

ACTORS:

* Admin
* Pastor
* Church Member
* System

USE CASES:

* Manage users
* Manage events
* Register
* Send messages
* Send events update notifications

-The admin can manage users in terms of Add/remove church members, assign roles. And also the admin manages events as can Create, update, or delete church events.

-The pastor can manage events as can Create, update, or delete church events.

-The church member can register and send messages

-The system can send events update notifications to the registered users.

## **3.3 Technologies Used**

The system will be developed using the following technologies:

* **Frontend:** Bootstrap, HTML5, CSS
* **Backend:** python, Django.
* **Database:** MySQL (Relational)

## **3.4 Development Process**

The project will adopt an **Agile development methodology**, allowing for iterative testing and continuous feedback. The key phases include:

1. **Phase 1 – Requirements Analysis (Week 1-2)**
   * Identify key user requirements (Admin, Church Leaders, Members).
   * Define system scope and create initial wireframes.
2. **Phase 2 – System Design (Week 3-4)**
   * Develop database schema and API endpoints.
   * Finalize UI/UX design mockups.
3. **Phase 3 – Development & Implementation (Week 5-10)**
   * Build the frontend.
   * Develop the backend logic and integrate the database.
   * Implement **authentication.**
4. **Phase 4 – Testing & Deployment (Week 11-12)**
   * Conduct unit, integration, and user acceptance testing.
   * Deploy to a cloud platform and conduct final optimizations.

## **3.5 Data Collection and Analysis**

* **User Activity Monitoring:** Track how users interact with the system to improve UI/UX.
* Data analytics will be integrated into the system using **SQL Queries (for MySQL)**, enabling church administrators. By following this methodology, the Church Management System will be **efficient, user-friendly, secure, and scalable**, addressing the administrative challenges faced by churches.

# **4. SYSTEM REQUIREMENTS**

## **4.1 Hardware Requirements**

The **Church Management System (CMS)** will be a web-based application, requiring minimal hardware for end users. However, for deployment and hosting, the following hardware specifications are recommended:

##### **For Server Hosting:**

* **Processor:** Intel Xeon / AMD Ryzen 5 or higher
* **RAM:** Minimum 8GB (16GB recommended for better performance)
* **Storage:** Minimum **100GB SSD** (scalable for database growth)
* **Internet Connection:** At least **10 Mbps upload speed** for cloud hosting
* **Cloud Hosting Options:** AWS, Digital Ocean, Firebase, or a VP.

##### **For Client Devices (End Users):**

* **PC/Laptop:** Any modern computer with **at least 4GB RAM and 10GB storage**
* **Operating System:** Windows, macOS, or Linux.
* **Browser:** Google Chrome, Mozilla Firefox, Microsoft Edge, Safari

## **4.2 Software Requirements**

The project requires the following software components:

##### **Development Tools:**

* **Visual Studio Code** – Code editor for frontend and backend development

##### **Backend:**

* **Python**
* **Django framework**

##### **Frontend:**

* **Bootstrap and CSS** – UI design framework

##### **Operating System:**

* **Windows 10/11, macOS, or Linux** for development

## **4.3 Network Requirements**

* **For Hosting:**
  + **Stable internet connection (10 Mbps or higher)** for cloud hosting.
  + **Public IP Address (for on premise deployment)** to allow external access.
* **For End Users:**
  + A **basic internet connection (at least 5 Mbps download speed)** for smooth access to the web-based system.
  + Secure HTTPS connection for **encrypted data transfer**.

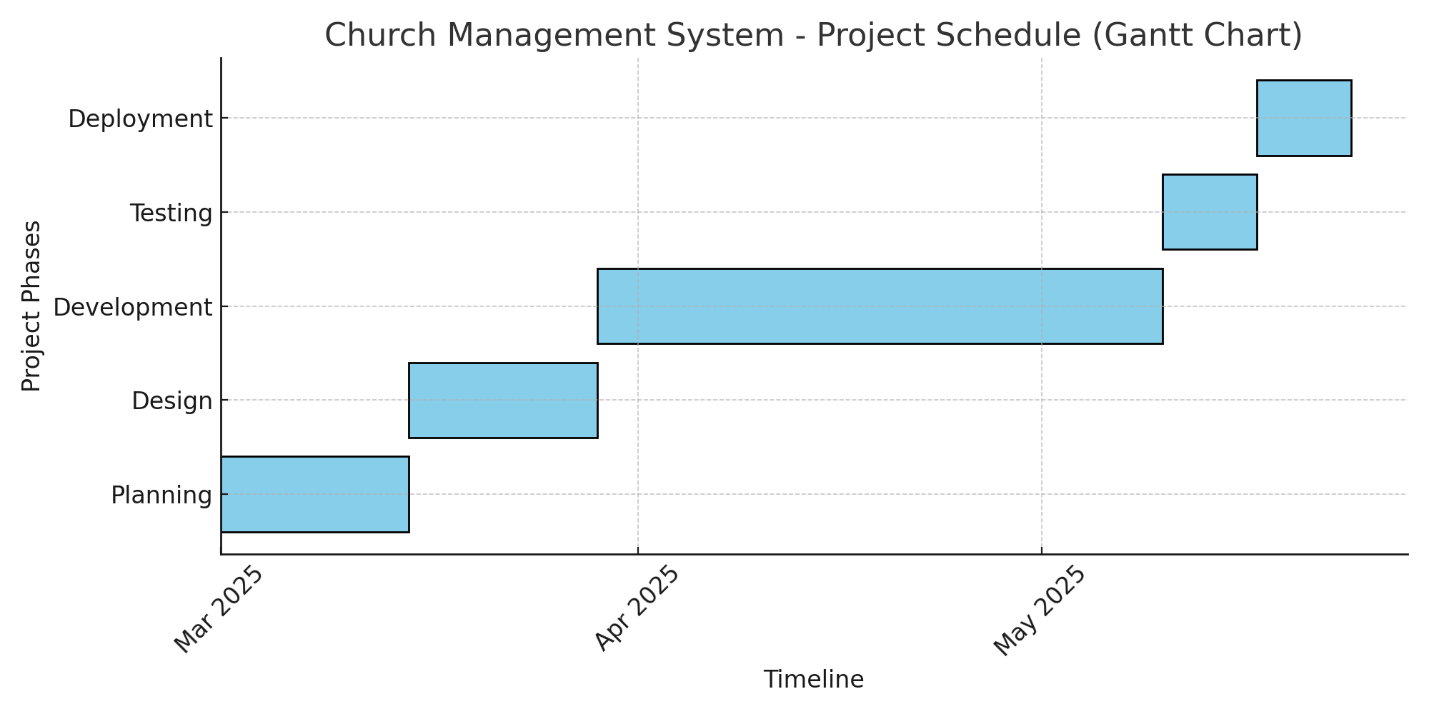
# **5. PROJECT SCHEDULE**

## **5.1 Timeline**

The project will be completed over **12 weeks (3 months)**, following an **Agile methodology** with iterative development and testing. Below is the breakdown of key phases and deliverables:

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Tasks | Duration | Deliverables |
| Planning (Week 1-2) | - Gather requirements - Define system scope - Create wireframes & system architecture | 2 Weeks | System requirements document, initial wireframes |
| Design (Week 3-4) | - Database schema design - UI/UX mockups | 2 Weeks | Database schema, UI/UX designs. |
| Development (Week 5-10) | -Frontend development(Bootstrap) -Backend development (Python, Django) - Database integration (MySQL) -Implement authentication & role-based access | 6 Weeks | Functional frontend & backend, working authentication system |
| Testing (Week 11) | - Unit testing (UI components) - Integration testing - User acceptance testing | 1 Weeks | |  | | --- | |  |  |  | | --- | | Bug reports, optimized system functionality | |
| Deployment (Week 12) | - Deploy on cloud server /local hosting - Final security testing - Documentation & user training | 1 Weeks | Live CMS system, deployment report, user manual |

**5.2 Gantt Chart Representation.**



# **8. RISK MANAGEMENT**

## **8.1 Potential Risks and Challenges**

|  |  |  |
| --- | --- | --- |
| **Risk Category** | **Description** | **Potential Impact** |
| Technical Challenges | Bugs, system crashes, database errors, or API failures | Delays in development and debugging |
| Time Constraints | Project deadlines may be missed due to unforeseen complexities | Reduced system quality due to rushed implementation |
| Resource Availability | Limited access to development tools, hosting servers, or testing environments | Slow development and deployment process |
| Security Risks | Data breaches, unauthorized access, or weak encryption methods | Loss of sensitive user information |
| User Adoption Issues | Church leaders and members may struggle to use the system effectively | Reduced system usage and impact |

## **8.2 Risk Mitigation Strategies**

|  |  |
| --- | --- |
| **Risk Category** | **Mitigation Strategy** |
| Technical Challenges | Conduct **regular code reviews**, implement **unit testing**, and use debugging tools. |
| Time Constraints | Use an **Agile development approach**, set **realistic milestones**, and prioritize essential features |
| Resource Availability | Utilize **open-source technologies**, cloud-based hosting, and ensure **backup plans** for hosting and databases |
| Security Risks | Implement **authentication**, **data encryption**, and regular security audits |
| User Adoption Issues | Provide **user training sessions**, develop a **simple UI**, and create a **user manual** |

# **9. EXPECTED OUTCOMES**

## **9.1 Expected Results and Deliverables**

Upon successful completion of the **Church Management System (CMS)**, the following deliverables will be provided:

1. **A Fully Functional Web-Based CMS**
   * A secure and user-friendly platform for managing church activities.
   * Role-based access control for **admins, church leaders, and members**.
2. **Key Features Delivered:**
   * **Member Management Module** – Registration.
   * **Events & Communication Module** – Announcements, scheduling, and automated notifications.

## **9.2 Performance Metrics & Success Criteria**

To evaluate the **effectiveness** of the project, the following **metrics** will be used:

|  |  |
| --- | --- |
| **Success Criteria** | **Performance Metric** |
| System Usability | At least **80% of users** should be able to complete key tasks without assistance. |
| Performance & Speed | Page load times should be **under 3 seconds** for optimal user experience. |
| System Uptime | The CMS should achieve at least **99% uptime** with minimal downtime. |
| Security & Data Protection | At least **70% of church staff/members** should actively use the system within 3 months. |

# **10. BUDGET**

## **10.1 Detailed Budget Breakdown**

The estimated budget for the **Church Management System (CMS)** is categorized into **software, hardware, hosting, and other essential costs**.

|  |  |  |
| --- | --- | --- |
| **Expense Category** | **Item Description** | **Estimated Cost** |
| Software Costs | Development tools (Visual Studio Code, GitHub ,Python ,Django) | Free |
|  | Database (MySQL) | Free |
|  |  |  |
| Network Costs | Network bandwidth | 210,000/= |
| Hardware Costs | computer | 1,000,000/= |
|  | mouse | 50,000/= |
| Total Estimated Budget |  | 1,260,000/= |

## **10.2 Funding Source & Budget Management**

**Funding Sources:**

* **Church Funds:** Contributions from the church budget.

**Budget Management:**

* The **church administrator/IT team** will oversee the project budget.
* **Regular expense tracking** will be done to ensure cost-effectiveness.

# **11. CONCLUSION**

## **11.1 Summary of Purpose and Objectives**

The **Church Management System (CMS)** project aims to streamline and automate the management of church operations, addressing the need for **efficient administrative tools**. The system will support church members, leaders, and administrators by providing functionalities like **member management**, sermons and **event scheduling**. By implementing this system, churches will be able to manage their resources more effectively, **improve transparency**, and foster better engagement with their congregation.

## **11.2 Reaffirming Expected Benefits and Significance**

This project offers **significant benefits**, including:

* **Improved Efficiency:** The CMS will automate time-consuming administrative tasks, allowing church leaders to focus more on their core responsibilities.
* **Better Communication:** The integrated event scheduling and notification system will improve communication between church leaders and members.
* **User-Friendly Design:** The intuitive interface will ensure ease of use for all church members, regardless of their technical expertise.

# **12. REFERENCES**

1. **Highsmith, J. (2002).** Agile Software Development: A Collaborative Approach to Software Engineering. Addison-Wesley Professional.

2. **Pichler, R. (2010).** Agile Product Management with Scrum: Creating Products that Customers Love. Addison-Wesley Professional.

3. <https://www.mountaingoatsoftware.com>