**Project Title: Church Membership Management System**

**Project Overview:**

Your task is to develop a comprehensive web-based application for Inqola Yevangeli to efficiently manage their church membership across branches in different provinces and countries. The system should provide administrators with the tools to track membership numbers, categorize members based on age groups and statuses, and monitor member activity within the church. This project will allow you to apply your knowledge of data modeling, user interface design, application security, web development, and web services/plugins to address the specific challenges faced by Inqola Yevangeli.

**Problem Statement from Inqola Yevangeli:**

Inqola Yevangeli is facing challenges in tracking their membership numbers accurately due to the constant influx of new members, as well as members leaving or passing away. With branches spread across South African provinces and other African countries, keeping track of member information centrally has become a necessity. The organization needs a system that can:

Track the number of members in each branch, categorizing them as new or old, and further classifying old members into age groups (youth, Sunday school, and adult).

Manage membership statuses, including sealed, welcomed, and testifiers, with clear criteria for each status.

Facilitate the process of determining how many members have been sealed in a specific year, how many are waiting for sealing, and how many are ready for baptism.

Identify active and inactive members to provide targeted support and assistance where needed.

**Project Requirements:**

**User Authentication and Authorization: (10 marks)**

* Implementation of secure user authentication using C#.
* Definition of three user roles: Super Admin, Branch Manager, and Regular User.
* Assignment of appropriate access permissions to each user role.

**Data Modeling: (15 marks)**

* Design of a relational database schema using SQL Server to store member information.
* Inclusion of tables for demographics, joining dates, statuses, and activity history.
* Ensuring data integrity through normalization and foreign key constraints.

**User Interface Design: (15 marks)**

* Development of a user-friendly interface using ASP.NET and C#.
* Creation of separate views for administrators and regular users, with intuitive navigation.
* Implementation of features for adding/updating members and viewing membership statistics.

**Membership Tracking: (20 marks)**

* Automatic categorization of members into age groups (youth, Sunday school, adult).
* Tracking of membership statuses (sealed, welcomed, testifiers) and recording relevant dates for each status transition.
* Generation of reports to identify the number of members sealed in a specific year and those awaiting sealing.

**Activity Monitoring: (15 marks)**

* Recording of member attendance and participation in church activities/events.
* Generation of reports to identify active and inactive members within each branch.
* Inclusion of a dashboard with graphical representations of membership trends and demographics.

**Application Security: (10 marks)**

* Implementation of encryption techniques to secure sensitive member data.
* Validation of user input to prevent common security vulnerabilities.
* Integration of role-based access control to restrict unauthorized access to certain features and data.

**Code Quality: (10 marks)**

* Writing clean and well-organized code using C#.
* Documentation of code for clarity and maintainability.
* Adoption of best practices in coding standards and naming conventions.

**Testing and Presentation: (5 marks)**

* Conducting thorough testing of the application to ensure functionality and reliability.
* Effective presentation of the project during the exam, including a demonstration of key features and a discussion of design decisions.

**Final Assessment Criteria:**

Excellent (80-100 marks): Comprehensive implementation of all requirements with exceptional attention to detail and high-quality code. Thorough testing and documentation.

Good (60-79 marks): Solid implementation of most requirements with minor issues in functionality or code quality. Adequate testing and documentation.

Satisfactory (40-59 marks): Basic implementation of core requirements with significant issues in functionality or code quality. Limited testing and documentation.

Needs Improvement (0-39 marks): Incomplete or inadequate implementation with major issues in functionality, code quality, testing, or documentation.