Birth Certificate Management System

A MINI-PROJECT BY:

Savitha.J.V 230701300

Smith.N.S 230701323

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **"UG DISSERTATION"** is the
bonafide work of **"SAVITHA J V, SMITH N S"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Mr.G.SARAVANA GOKUL
Assistant Professor (SS),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

SIGNATURE

Ms.V.JANANEE
Assistant Professor(SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINE

ABSTRACT

The **Birth Certificate Management System** is a comprehensive software solution developed in Java to facilitate the efficient management, storage, and retrieval of birth certificate data. The system is designed to digitize and automate the entire process of issuing, updating, and maintaining birth certificates. By leveraging the power of Java programming and a user-friendly interface, the system provides an effective way to manage the birth records of individuals, reducing manual errors, streamlining workflows, and ensuring better accessibility to birth-related data.

In traditional systems, the process of issuing and managing birth certificates is often cumbersome and prone to human errors, leading to delays and inefficiencies. This project aims to address these challenges by creating a centralized and secure platform that provides a seamless experience for both users and administrators. The system allows authorized personnel to input and validate essential information such as an individual's name, date of birth, place of birth, parent details, and more. Once data is entered, the system generates an official birth certificate document that is ready for printing.

The core features of the Birth Certificate Management System include:

1. **User Authentication**: Ensures that only authorized personnel can access and modify sensitive birth record data. This enhances security and prevents unauthorized access.
2. **Data Entry and Validation**: Simplifies the entry of personal information and ensures that all required fields are filled out correctly, thus minimizing errors.
3. **Certificate Generation**: Automatically generates a birth certificate document in a printable format that adheres to the legal standards.
4. **Search and Retrieval**: Provides an efficient search mechanism that allows users to retrieve records based on various criteria such as name, date of birth, or place of birth.
5. **Database Integration**: Utilizes a relational database (such as MySQL or SQLite) to store birth records securely, ensuring that data is organized, easily accessible, and can be backed up regularly.

TABLE O**F CONTENTS**

INTRODUCTION

1.1 INTRODUCTION

A **Birth Certificate Management System** is a software solution designed to automate and streamline the management of birth certificate records. Traditionally, birth certificates are issued, maintained, and stored manually, leading to inefficiencies such as time delays, data inaccuracies, and difficulties in record retrieval. The advent of modern technology, particularly software systems like the one developed in this Java project, aims to address these challenges, making the process of managing birth certificates more efficient, accurate, and accessible.

In this Java-based **Birth Certificate Management System**, the objective is to provide a centralized digital platform for government agencies and institutions responsible for issuing and managing birth certificates. The system allows users to store, retrieve, and update birth certificate data in a secure and user-friendly manner. The system also provides functionalities to generate birth certificates automatically based on the data entered, ensuring that they are legally formatted and ready for distribution.

1.1 IMPLEMENTATION

The **BIRTH CERTIFICATE MANAGEMENT** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.2 WEBSITE FEATURES

- User Authentication
- Birth Certificate Registration
- Administrator Features
- Search and Retrieve Records
- Certificate Generation and Download
- Record Update and Modification
- Security Features.

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR	:	Inteli7
MEMORY SIZE	:	4GB(Minimum)
HARD DISK	:	500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE	:	Java, MySQL
FRONT-END	:	Java Swings
BACK-END	:	MySQL
OPERATING SYSTEM	:	Windows 11

SAMPLE CODE

3.1 FRONTEND CODE:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;

public class BirthCertificateManagement extends JFrame {
    private JTextField nameField, dobField, placeField, certNumberField;

    private Connection connection;

    public BirthCertificateManagement() {
        // Database Connection
        try {
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/birth_certificate_db", "root", "Shalini@2005"); // Replace with your DB credentials
        } catch (SQLException e) {
            e.printStackTrace();
        }

        setTitle("Birth Certificate Management");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Main Panel
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(4, 1, 10, 10));

        // Buttons for each functionality
        JButton addButton = new JButton("Add Certificate");
        addButton.setFont(new Font("Arial", Font.PLAIN, 16));
        addButton.addActionListener(e -> openAddCertificateWindow());
        panel.add(addButton);

        JButton viewButton = new JButton("View Certificates");
        viewButton.setFont(new Font("Arial", Font.PLAIN, 16));
        viewButton.addActionListener(e -> openViewCertificatesWindow());
        panel.add(viewButton);

        JButton deleteButton = new JButton("Delete Certificates");
        deleteButton.setFont(new Font("Arial", Font.PLAIN, 16));
        deleteButton.addActionListener(e -> openDeleteCertificatesWindow());
```

```
    panel.add(deleteButton);

    add(panel);
}

private void openAddCertificateWindow() {
    JFrame addFrame = new JFrame("Add Certificate");
    addFrame.setSize(500, 350);
    addFrame.setLocationRelativeTo(this);
    addFrame.setLayout(null);

    JLabel nameLabel = new JLabel("Name:");
    nameLabel.setFont(new Font("Arial", Font.PLAIN, 14));
    nameLabel.setBounds(20, 20, 150, 25);
    addFrame.add(nameLabel);

    nameField = new JTextField();
    nameField.setBounds(200, 20, 200, 25);
    addFrame.add(nameField);

    JLabel dobLabel = new JLabel("Date of Birth:");
    dobLabel.setFont(new Font("Arial", Font.PLAIN, 14));
    dobLabel.setBounds(20, 60, 150, 25);
    addFrame.add(dobLabel);

    dobField = new JTextField();
    dobField.setBounds(200, 60, 200, 25);
    addFrame.add(dobField);

    JLabel placeLabel = new JLabel("Place of Birth:");
    placeLabel.setFont(new Font("Arial", Font.PLAIN, 14));
    placeLabel.setBounds(20, 100, 150, 25);
    addFrame.add(placeLabel);

    placeField = new JTextField();
    placeField.setBounds(200, 100, 200, 25);
    addFrame.add(placeField);

    JLabel certNumberLabel = new JLabel("Certificate Number:");
    certNumberLabel.setFont(new Font("Arial", Font.PLAIN, 14));
    certNumberLabel.setBounds(20, 140, 150, 25);
    addFrame.add(certNumberLabel);

    certNumberField = new JTextField();
    certNumberField.setBounds(200, 140, 200, 25);
    addFrame.add(certNumberField);
```

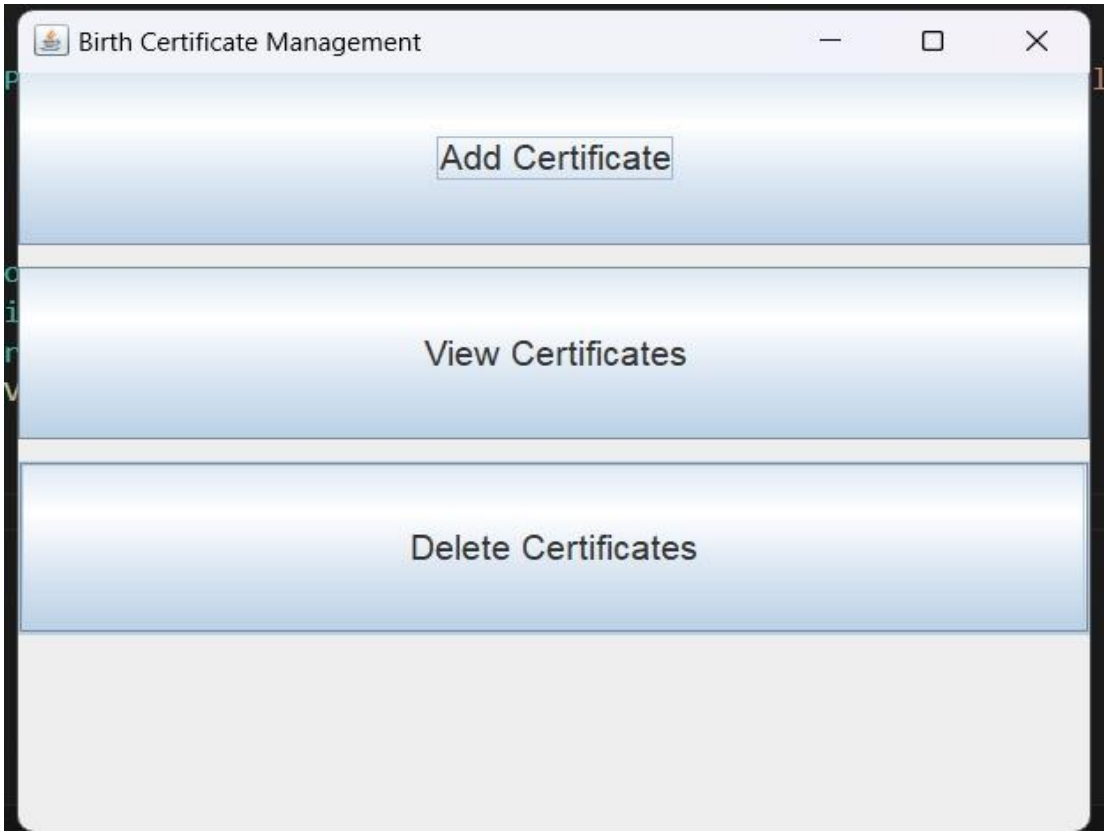
```
        JButton saveButton = new JButton("Save");
        saveButton.setFont(new Font("Arial", Font.BOLD, 14));
        saveButton.setBounds(150, 200, 150, 30);
        saveButton.addActionListener(e -> addCertificate());
        addFrame.add(saveButton);

        addFrame.setVisible(true);
    }

    private void openViewCertificatesWindow() {
        JFrame viewFrame = new JFrame("View Certificates");
        viewFrame.setSize(800, 500);
        viewFrame.setLocationRelativeTo(this);

        DefaultTableModel tableModel = new DefaultTableModel(new String[]{"Serial",
"Name", "Date of Birth", "Place of Birth", "Certificate Number"}, 0);
        JTable table = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(table);

        JButton refreshButton = new JButton("Refresh");
        refreshButton.setFont(new Font("Arial", Font.PLAIN, 14));
        refreshButton.addActionListener(e -> loadCertificates(tableModel));

        viewFrame.setLayout(new BorderLayout());
        viewFrame.add(scrollPane, BorderLayout.CENTER);
        viewFrame.add(refreshButton, BorderLayout.SOUTH);

        loadCertificates(tableModel);
        viewFrame.setVisible(true);
    }

    private void openDeleteCertificatesWindow() {
        JFrame deleteFrame = new JFrame("Delete Certificates");
        deleteFrame.setSize(800, 500);
        deleteFrame.setLocationRelativeTo(this);

        DefaultTableModel tableModel = new DefaultTableModel(new String[]{"Serial",
"Name", "Date of Birth", "Place of Birth", "Certificate Number"}, 0);
        JTable table = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(table);

        JButton deleteButton = new JButton("Delete");
        deleteButton.setFont(new Font("Arial", Font.PLAIN, 14));
        deleteButton.addActionListener(e -> deleteCertificate(table, tableModel));

        deleteFrame.setLayout(new BorderLayout());
        deleteFrame.add(scrollPane, BorderLayout.CENTER);
```

```
        deleteFrame.add(deleteButton, BorderLayout.SOUTH);

        loadCertificates(tableModel);
        deleteFrame.setVisible(true);
    }

    private void addCertificate() {
        try {
            PreparedStatement ps = connection.prepareStatement(
                    "INSERT INTO birth_certificates (name, date_of_birth, place_of_birth, certificate_number) VALUES (?, ?, ?, ?)"
            );
            ps.setString(1, nameField.getText());
            ps.setString(2, dobField.getText());
            ps.setString(3, placeField.getText());
            ps.setString(4, certNumberField.getText());
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Certificate added successfully.");
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error adding certificate.");
        }
    }

    private void loadCertificates(DefaultTableModel tableModel) {
        tableModel.setRowCount(0); // Clear existing rows
        try {
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM birth_certificates");
            int serial = 1; // Start serial numbers from 1
            while (rs.next()) {
                tableModel.addRow(new Object[]{
                        serial++,
                        rs.getString("name"),
                        rs.getString("date_of_birth"),
                        rs.getString("place_of_birth"),
                        rs.getString("certificate_number")
                });
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error fetching certificates.");
        }
    }

    private void deleteCertificate(JTable table, DefaultTableModel tableModel) {
        int selectedRow = table.getSelectedRow();
```

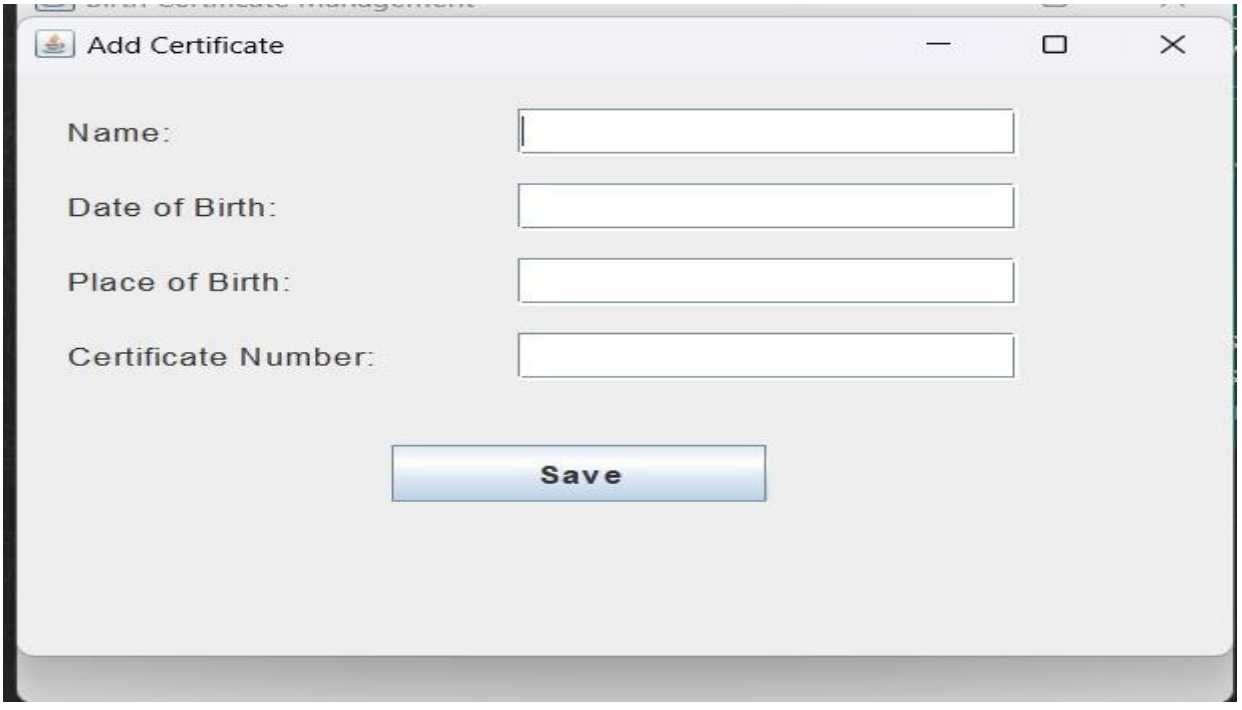
```
        if (selectedRow != -1) {
            String certNumber = tableModel.getValueAt(selectedRow, 4).toString();
            try {
                PreparedStatement ps = connection.prepareStatement("DELETE FROM
birth_certificates WHERE certificate_number = ?");
                ps.setString(1, certNumber);
                ps.executeUpdate();
                loadCertificates(tableModel); // Refresh the table with reordered serial numbers
                JOptionPane.showMessageDialog(this, "Certificate deleted successfully.");
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Error deleting certificate.");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Select a certificate to delete.");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            BirthCertificateManagement app = new BirthCertificateManagement();
            app.setVisible(true);
        });
    }
}
```

SNAPSHOTS

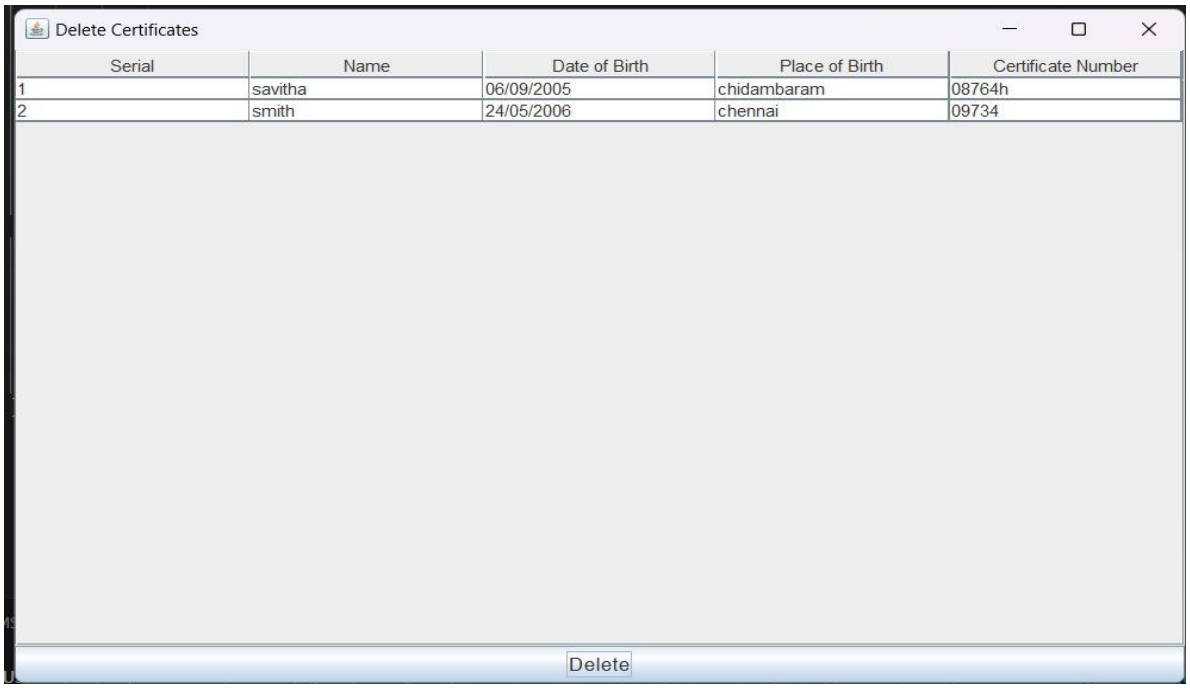
4.1 DASHBOARD PAGE



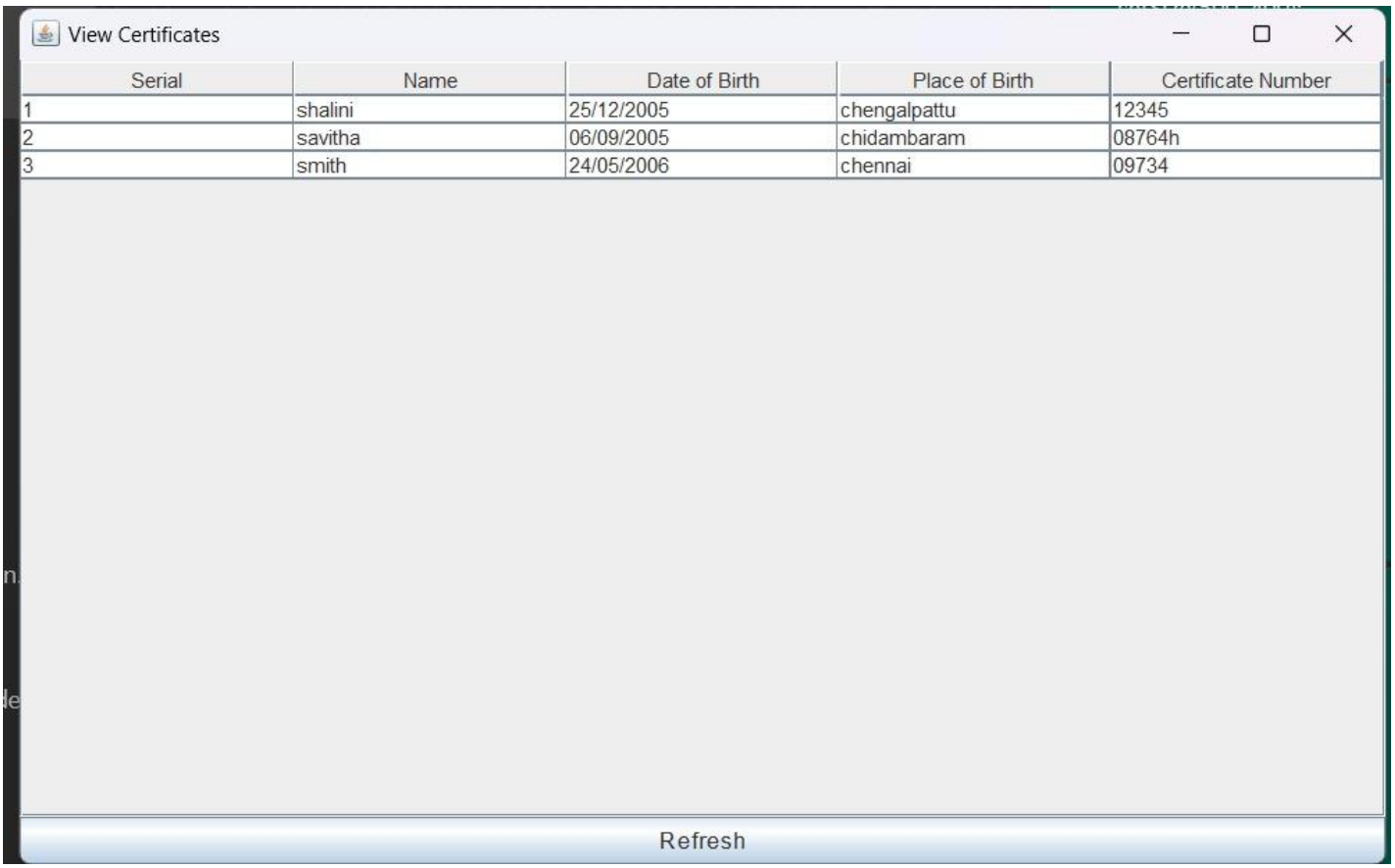
4.2 ADD BIRTH CERTIFICATE DATA PAGE



4.3 DELETE BIRTH CERTIFICATE DATA PAGE



4.4 VIEW BIRTH CERTIFICATE DATA PAGE



CONCLUSION

In conclusion, the **Birth Certificate Management System** developed using Java represents a modern, efficient, and secure solution to the traditionally paper-based and cumbersome process of issuing, storing, and managing birth certificates. By leveraging the power of Java technologies, such as Spring Boot, MySQL, and a user-friendly web interface, this system effectively addresses the challenges associated with record-keeping, data retrieval, and document generation in the public sector.

The system's key features, such as automated birth certificate generation, secure user authentication, data validation, and the ability to search and retrieve records quickly, make it a highly efficient tool for both administrative authorities and citizens. By eliminating manual errors, improving the accuracy of birth records, and enabling faster document issuance, the system enhances the overall efficiency of government services. Additionally, with robust security measures to protect sensitive personal information, the system ensures privacy and confidentiality for all users

.

Moreover, the **Birth Certificate Management System** is scalable and flexible, meaning it can handle increasing volumes of data as the population grows, and can be easily extended to accommodate future requirements. The integration of database backup mechanisms further guarantees the safety of records, while the ability to generate reports and manage users allows administrators to monitor and control the system effectively.

REFERENCES

1. **https://www.javatpoint.com/java-tutorial**
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/
4. SQL | Codecademy