LIBRARY MANAGEMENT SYSTEM

19IT452 - JAVA PROGRAMMING LAB

MINI PROJECT REPORT

SUBMITTED BY ,

Hari Hara Sudan M	9517202206021
Kishore N A	9517202206024
Dhanis Ahmed K	9517202206014

DEPARTMENT OF INFORMATION TECHNOLOGY

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

2023 - 2024

CERTIFICATE

This is to certify that it is the bonafide work done by **Hari Hara Sudan M [22BIT043] , Kishore N A [22BIT045] , Dhanis Ahmed [22BIT037]** for their Mini Project on **LIBRARY MANGEMENT SYSTEM** in the 19IT452 - JAVA PROGRAMMING LAB at Mepco Schlenk Engineering College, Sivakasi during the year 2023-2024

SIGNATURE

Mrs. M.Blessa Binolin Pepsi, M.E.,
STAFF IN-CHARGE
Assistant Professor (Sl. G)
Information Technology
Mepco Schlenk Engineering College
Virudhunagar Dt. – 626 005

SIGNATURE

Dr. T. Revathi, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Senior Professor
Information Technology
Mepco Schlenk Engineering College
Virudhunagar Dt. – 626 005

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and our reputed institution Mepco Schlenk Engineering College, Sivakasi. We would like to extend our sincere thanks to our Principal Dr. S. Arivazhagan ME., PhD.

We would like to express my special gratitude and thanks to Dr.T.Revathi ME., PhD. Senior Professor & Head of the Department for giving us such a wonderful experience.

We are highly indebted to Mrs. M. Blessa Binolin Pepsi AP (SL.Grade)/IT, Mrs. M. Prasha Meena AP/IT for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards other teaching and non teaching staff members of IT Department for their kind co-operation and encouragement which help us in all means to complete this project successfully.

ABSTRACT

Our project aims to develop a user interface for establishing Online Voting .This project will make everyone to vote from home itself by just having their EPIC number .

TABLE OF CONTENTS

1.INTRODUCTION

In the digital age, podcasts have emerged as a popular medium for storytelling, education, and entertainment, attracting a diverse audience worldwide. To cater to the growing demand for accessible and engaging podcast platforms, we have developed a state-of-the-art podcast website leveraging React for the frontend and Java Spring Boot for the backend. This project aims to provide users with an exceptional experience in discovering, streaming, and managing their favorite podcasts.

The choice of React for the frontend development ensures a dynamic and responsive user interface, offering features such as user authentication, advanced podcast search and filtering, personalized recommendations, and playlist management. React's component-based architecture enhances the maintainability and scalability of the application, allowing for a modular and efficient development process.

On the backend, Java Spring Boot serves as the foundation for robust server-side operations. It manages user data, podcast metadata, and streaming functionalities with high performance and reliability. Spring Boot's RESTful APIs facilitate seamless communication between the frontend and backend, ensuring a cohesive user experience and enabling future expansion and integration with additional services.

This podcast website exemplifies the integration of modern web technologies to create a comprehensive platform tailored for podcast enthusiasts. It highlights the strengths of React and Java Spring Boot in delivering a user-centric, high-performance web application that meets the evolving needs of its audience

1.1 PURPOSE

The primary purpose of this podcast website is to create a user-friendly and feature-rich platform that enhances the experience of discovering, streaming, and managing podcasts. By leveraging the latest web technologies, this project aims to fulfill several key objectives:

1. Enhanced User Experience: Provide a seamless and intuitive interface for users to explore and enjoy a vast array of podcasts, ensuring ease of navigation and interaction.
2. Personalization: Offer personalized recommendations based on user preferences and listening history to help users discover new content tailored to their interests.
3. Scalability and Performance: Utilize a robust backend infrastructure with Java Spring Boot to ensure reliable performance and scalability, accommodating an increasing number of users and podcasts without compromising on speed or functionality.
4. Comprehensive Features: Incorporate essential features such as user authentication, playlist creation, and advanced search capabilities to meet the diverse needs of podcast listeners.

5. Integration and Extensibility: Design the platform to easily integrate with third-party services for hosting, analytics, and additional functionalities, ensuring the ability to expand and enhance the platform over time.

1.1.1 SIH PROBLEM STATEMENT - SIH 1489

Description:

Submit your ideas to address the growing pressures on the city’s resources, transport networks, and logistic infrastructure

1.2 SCOPE

The scope of the podcast website project encompasses the following aspects:

1. Frontend Development:

- User Interface Design: Creation of a responsive and visually appealing user interface using React, ensuring compatibility across various devices and screen sizes.
- User Authentication: Implementation of user registration, login, and profile management functionalities.
- Podcast Discovery: Development of features for browsing, searching, and filtering podcasts based on categories, popularity, and user preferences.
- Personalized Recommendations: Incorporation of algorithms to provide tailored podcast suggestions based on user behavior and interests.
- Playlist Management: Enabling users to create, manage, and share playlists of their favorite podcasts.

2. Backend Development:

- Data Management: Utilizing Java Spring Boot to manage user data, podcast metadata, and streaming functionalities.
- RESTful APIs: Development of RESTful APIs to facilitate communication between the frontend and backend, ensuring efficient data exchange and synchronization.
- Scalability and Performance Optimization: Ensuring the backend infrastructure can handle increasing loads and provide fast, reliable performance.
- Security Measures: Implementing robust security protocols to protect user data and ensure secure authentication and data transactions.

3. Integration:

- Third-Party Services: Integration with third-party services for podcast hosting, analytics, and additional functionalities such as social sharing and notifications.
- External APIs: Leveraging external APIs to enrich the platform's capabilities, such as fetching podcast metadata and streamlining podcast updates.

4. Testing and Deployment:

- Quality Assurance: Conducting thorough testing, including unit tests, integration tests, and user acceptance tests to ensure the platform is bug-free and performs as expected.
- Continuous Deployment: Setting up continuous integration and deployment pipelines to streamline the deployment process and ensure regular updates and maintenance.

5. Maintenance and Support:

- Documentation: Providing comprehensive documentation for both developers and users to facilitate ease of use and ongoing development.
- Ongoing Enhancements: Planning for regular updates and new feature additions based on user feedback and evolving technological trends.

This scope ensures a comprehensive approach to developing a robust and user-friendly podcast website, addressing both current needs and future growth potential.

1.3 DEFINITIONS , ACRONYMS AND ABBREVIATIONS

1. API (Application Programming Interface): A set of protocols and tools for building software and applications, allowing different software systems to communicate with each other.
2. CI/CD (Continuous Integration/Continuous Deployment): Practices in software development where code changes are automatically built, tested, and deployed to production, ensuring frequent and reliable updates.
3. CRUD (Create, Read, Update, Delete): The four basic operations of persistent storage in databases and applications, representing the fundamental actions for managing data.
4. CSS (Cascading Style Sheets): A stylesheet language used for describing the presentation of a document written in HTML or XML, used to style and layout web pages.

5. HTML (Hypertext Markup Language): The standard markup language for creating web pages and web applications.
6. JWT (JSON Web Token): A compact, URL-safe means of representing claims to be transferred between two parties, commonly used for authentication and information exchange.
7. REST (Representational State Transfer): An architectural style for designing networked applications, relying on stateless, client-server communication via HTTP requests.
8. SPA (Single Page Application): A web application that loads a single HTML page and dynamically updates content as the user interacts with the app, providing a smooth and responsive user experience.
9. UI (User Interface): The point of interaction between the user and a digital device or application, encompassing all visual and interactive elements.

10. UX (User Experience): The overall experience of a person using a product, especially in terms of how easy and pleasant it is to use.
11. IDE (Integrated Development Environment): A software application providing comprehensive facilities to programmers for software development, including a code editor, debugger, and build automation tools.
12. JSON (JavaScript Object Notation): A lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate.
13. SEO (Search Engine Optimization): The process of improving the visibility and ranking of a website or web page in search engine results.
14. SQL (Structured Query Language): A standardized language for managing and manipulating relational databases.
15. UI/UX (User Interface/User Experience): Combined terms referring to the design and user interaction aspects of digital products, focusing on creating effective and enjoyable user experiences.

16. CRUD (Create, Read, Update, Delete): Basic operations for manipulating data in a database, crucial for managing application data.
17. IDE (Integrated Development Environment): A software suite that consolidates basic tools required for software development, such as coding, debugging, and testing.

This section ensures that all technical terms and abbreviations used in the project documentation are clearly defined for better understanding and consistency.

1.4 REFERENCES

1. Official Documentation:
   - React Documentation: [https://reactjs.org/docs/getting-started.html]
   - Java Spring Boot Documentation: [https://spring.io/projects/spring-boot]
2. Tutorials and Guides:
   - React and Spring Boot Integration Tutorial: [Example Tutorial Link]
   - Building RESTful APIs with Spring Boot: [Spring Boot RESTful Tutorial]
3. Books:
   - "Pro Spring Boot 2" by Felipe Gutierrez
   - "Learning React: A Hands-On Guide to Building Web Applications Using React and Redux" by Kirupa Chinnathambi
4. Online Courses:
   - Udemy Course: "Full Stack Development with Spring Boot and React"
   - Pluralsight Course: "Building Full-Stack Apps with React and Spring Boot"

5. GitHub Repositories:
   - Example Spring Boot Repository: [Spring Boot Example Repo]
   - Example React Repository: [React Example Repo]

6. Community Forums:
   - Stack Overflow: [https://stackoverflow.com/]
   - Reddit React Community: [https://www.reddit.com/r/reactjs/]
   - Reddit Spring Boot Community: [https://www.reddit.com/r/springboot

7. Other Resources:
   - Baeldung: [https://www.baeldung.com]Provides in-depth tutorials and articles on Java, Spring Boot, and related technologies.
   - Medium: [https://medium.com Look for articles and tutorials written by developers sharing their experiences with React and Spring Boot.

Ensure to format the references according to the citation style preferred by your institution or organization.

1.5 OVERVIEW

This project entails the development of a dynamic and interactive podcast website utilizing the React framework for the frontend and Java Spring Boot for the backend. The website aims to provide users with a seamless experience in discovering, streaming, and managing podcasts.

The frontend, built with React, ensures a responsive and intuitive user interface. Key features include user authentication, podcast search and filtering, personalized recommendations, and the ability to create and manage playlists. The design prioritizes user experience with a clean, modern aesthetic and straightforward navigation.

On the backend, Java Spring Boot facilitates robust and scalable server-side operations. The backend handles user data, podcast metadata, streaming functionalities, and integrates with third-party services for extended capabilities such as hosting and analytics. The use of Spring Boot's RESTful APIs ensures efficient communication between the frontend and backend, while also enabling future scalability and maintenance.

1.6 CONCEPT OVERVIEW

The technology stack of the podcast web application comprises React.js for the frontend, Java Spring Boot for the backend, and MySQL for the database. This section provides an overview of these technologies and their roles in the application architecture.

React.js (Frontend)

React.js is a popular JavaScript library for building user interfaces. It allows for the creation of dynamic and interactive components that update efficiently when data changes. React.js follows a component-based architecture, making it easy to develop reusable UI elements.

Java Spring Boot (Backend)

Java Spring Boot is a powerful framework for building Java-based web applications. It simplifies the development of production-ready applications by providing a set of tools and conventions for rapid application development. Spring Boot offers features like dependency injection, RESTful web services, and security mechanisms.

MySQL (Database)

MySQL is an open-source relational database management system known for its reliability, scalability, and performance. It is widely used in web applications for storing and managing structured data. MySQL supports SQL queries for data manipulation and retrieval, making it a robust choice for applications requiring data persistence.

Integration

The frontend built with React.js interacts with the backend services developed using Java Spring Boot through RESTful APIs. Data exchanged between the frontend and backend is stored and retrieved from the MySQL database. This seamless integration enables the application to deliver a responsive user experience while efficiently managing data operations.

Benefits

- React.js offers a declarative approach to building UI components, enhancing code maintainability and reusability.
- Java Spring Boot simplifies backend development with its opinionated setup and built-in features like embedded servers and dependency management.
- MySQL provides a reliable and scalable database solution for storing application data securely.

Challenges

- Integrating frontend and backend technologies requires careful coordination to ensure smooth communication between components.
- Managing database interactions and ensuring data consistency across the application can be challenging, especially in complex applications.

The technology stack of React.js, Java Spring Boot, and MySQL forms a robust foundation for the podcast web application, enabling efficient frontend development, backend logic implementation, and secure data storage. By leveraging the strengths of each technology, the application delivers a seamless user experience while maintaining scalability and performance.

1.7 OUR WORK:

Podcast applications have gained popularity for delivering audio content to users. This topic focuses on enhancing user experience in podcast applications by integrating speech recognition and navigation features. By leveraging speech technology, users can interact with the application using voice commands, improving accessibility and usability.

Key Points

1. **Speech Recognition Implementation**: useSpeechRecognition is a React hook that gives a component access to a transcript of speech picked up from the user's microphone. SpeechRecognition manages the global state of the Web Speech API, exposing functions to turn the microphone on and off.

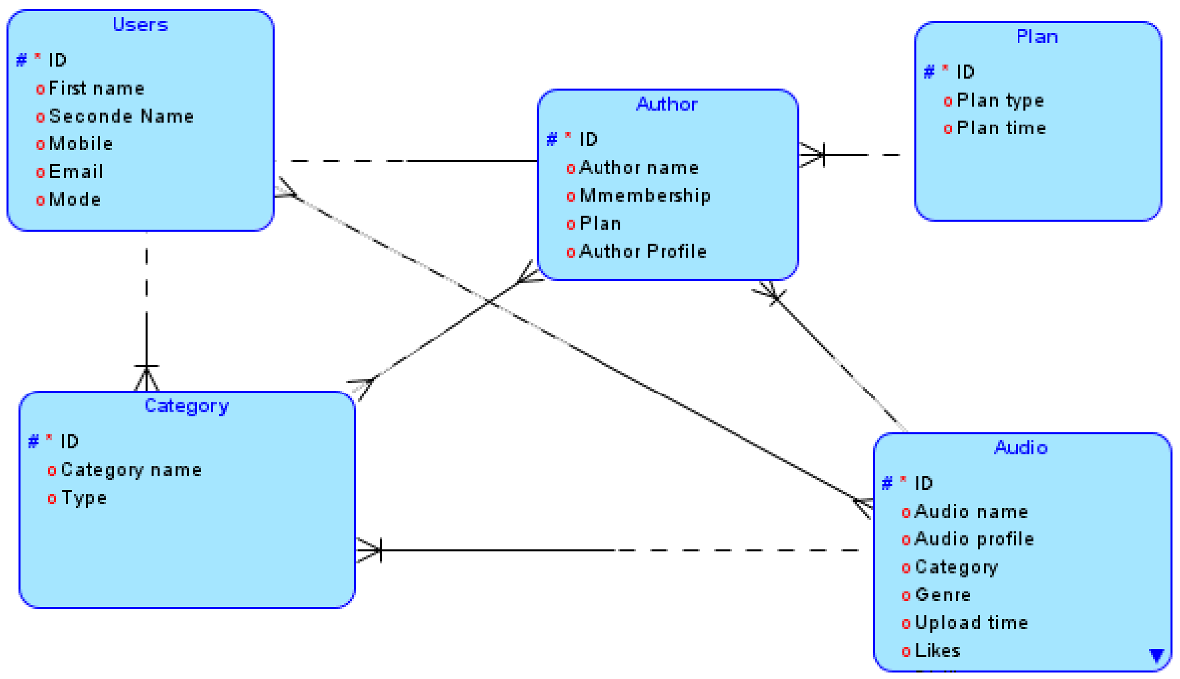
2. **Transcript Generation**: MyMemory is the world's largest Translation Memory. MyMemory is 100% free.It has been created collecting TMs from the European Union, United Nations and aligning the best domain specific multilingual websites.

3. **Speech Navigation Features**: useSpeechRecognition is a React hook that gives a component access to a transcript of speech picked up from the user's microphone. SpeechRecognition manages the global state of the Web Speech API, exposing functions to turn the microphone on and off.

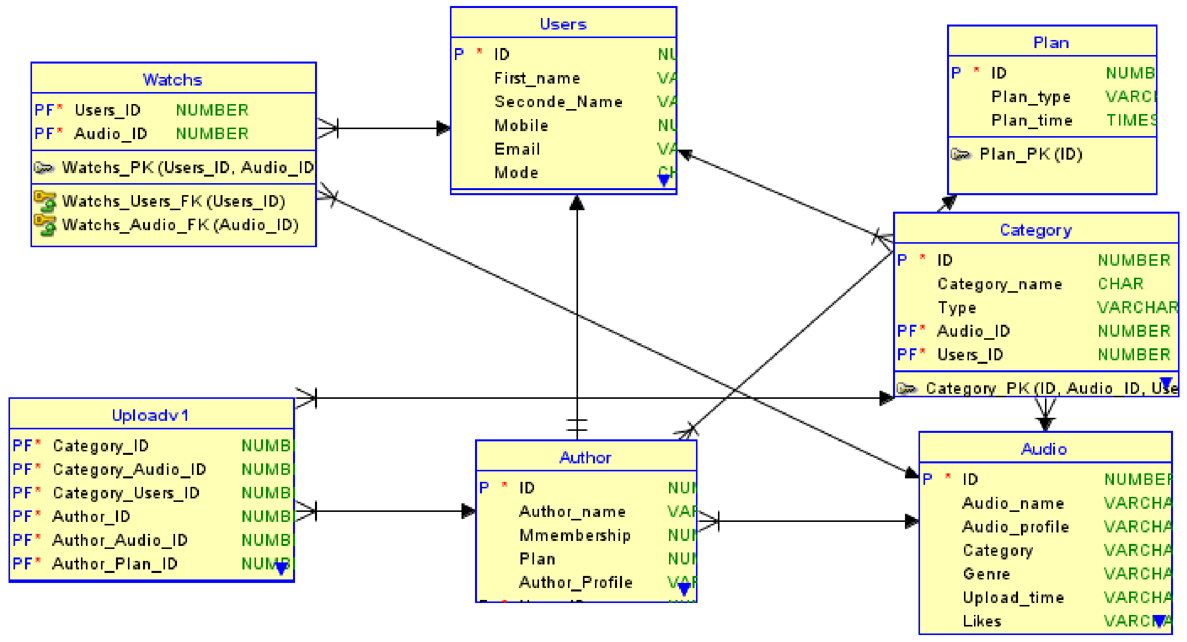
4**. User Interaction Design**: By integrating Material-UI for user interaction design in podcast applications, developers can create visually appealing, user-friendly interfaces that elevate the overall user experience. This topic delves into the benefits, design considerations, and best practices of using Material-UI components to design interactive and engaging interfaces for podcast consumption.

6. **Future Trends and Innovations**: The future of podcast applications is poised for innovation and transformation, driven by advancements in technology, user preferences, and content delivery. By embracing emerging trends such as personalization, interactivity, live streaming, monetization strategies, transcription services, and community building, podcast applications can evolve to meet the changing needs and expectations of users in the digital audio landscape.

2. ER DIAGRAM



Users
# * ID
o First name
o Seconde Name
o Mobile
o Email
o Mode
Author
# * ID
o Author name
o Mmembership
o Plan
o Author Profile
Plan
# * ID
o Plan type
o Plan time
Category
# * ID
o Category name
o Type
Audio
# * ID
o Audio name
o Audio profile
o Category
o Genre
o Upload time
o Likes



Watchs
PF* Users_ID NUMBER
PF* Audio_ID NUMBER
Watchs_PK (Users_ID, Audio_ID
Watchs_Users_FK (Users_ID)
Watchs_Audio_FK (Audio_ID)
Users
P * ID NU
First_name VA
Seconde_Name VA
Mobile NU
Email VA
Mode CH
Plan
P * ID NUMB
Plan_type VARC
Plan_time TIMES
Plan_PK (ID)
Category
P * ID NUMBER
Category_name CHAR
Type VARCHAR
PF* Audio_ID NUMBER
PF* Users_ID NUMBER
Category_PK (ID, Audio_ID, Use
Uploadv1
PF* Category_ID NUMB
PF* Category_Audio_ID NUMB
PF* Category_Users_ID NUMB
PF* Author_ID NUMB
PF* Author_Audio_ID NUMB
PF* Author_Plan_ID NUMB
Author
P * ID NU
Author_name VA
Mmembership NU
Plan NU
Author_Profile VA
Audio
P * ID NUMBER
Audio_name VARCHA
Audio_profile VARCHA
Category VARCHA
Genre VARCHA
Upload_time VARCHA
Likes VARCHA

3. IMPLEMENTATION

LOGIN PAGE

```
import java.awt.HeadlessException;
import java.sql.*;
import java.time.LocalDate;
import java.util.*;
import javax.swing.JOptionPane;
public class loginpage extends javax.swing.JFrame {

    Connection con=null;
    PreparedStatement st=null;
    ResultSet rs=null;
    int k=0;
    public loginpage() {
        initComponents();
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
    }

    private void jPasswordField1ActionPerformed(java.awt.event.ActionEvent evt){
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        LocalDate currentDate = LocalDate.now();
    LocalDate targetDate = LocalDate.of(2024, 5, 25);
    String username = jTextField1.getText();
    char[] pass = jPasswordField1.getPassword();
    String password = new String(pass);
        try {
        con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"system", "dinesh07");
        if (username.equals("dinesh")) {
            home1 h1 = new home1();
            h1.setVisible(true);
            this.dispose();
        }
else {
            String sql = "SELECT * FROM userlogin WHERE username=? AND
password=?";
            st = con.prepareStatement(sql);
```

```
          st.setString(1, username);
          st.setString(2, password);
          rs = st.executeQuery();
          if (rs.next()) {
             JOptionPane.showMessageDialog(null, "Login Successful");
             jTextField1.setText("");
             jPasswordField1.setText("");
             String checkUvreqSql = "SELECT * FROM uvreq WHERE
username=?";
             PreparedStatement uvreqSt = con.prepareStatement(checkUvreqSql);
             uvreqSt.setString(1, username);
             ResultSet uvreqRs = uvreqSt.executeQuery();
             if (uvreqRs.next()) {
                int status = uvreqRs.getInt("status");
                if (status == 0) {
                   String updateStatusSql = "UPDATE uvreq SET status = 1 WHERE
username = ?";
                   PreparedStatement updateStatusSt =
con.prepareStatement(updateStatusSql);
                   updateStatusSt.setString(1, username);
                   updateStatusSt.executeUpdate();
                   JOptionPane.showMessageDialog(null, "Your voter ID has been
received.");
                } else if (status == 5) {
                   JOptionPane.showMessageDialog(null, "Your voter ID has been
rejected.");
                   k = 1;
                   String deleteUvreqSql = "DELETE FROM uvreq WHERE
username = ?";
                   PreparedStatement deleteSt =
con.prepareStatement(deleteUvreqSql);
                   deleteSt.setString(1, username);
                   deleteSt.executeUpdate();
                }
             }
             if (currentDate.equals(targetDate)) {
                String statusSql = "SELECT status FROM uvreq WHERE
username=?";
                PreparedStatement statusSt = con.prepareStatement(statusSql);
                statusSt.setString(1, username);
                ResultSet statusRs = statusSt.executeQuery();
                if (statusRs.next() && statusRs.getInt("status") == 1) {
                   voting v = new voting();
                   v.setVisible(true);
```

```
                    this.dispose();
                } else {
                    home h = new home();
                    h.setVisible(true);
                    this.dispose();
                }
            } else {
                home h = new home();
                h.setVisible(true);
                this.dispose();
            }
        } else {
            JOptionPane.showMessageDialog(null, "Invalid username or
password");
            jTextField1.setText("");
            jPasswordField1.setText("");
        }
    }
} catch (HeadlessException | SQLException ex) {
    JOptionPane.showMessageDialog(null, ex);
} finally {
    try {
        if (rs != null) rs.close();
        if (st != null) st.close();
        if (con != null) con.close();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, ex);
    }
}
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        newuser n=new newuser();
       n.setVisible(true);
       this.dispose();
}
public static void main(String args[]) {
       java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new loginpage().setVisible(true);
        }
    });
}
}
```

HOME PAGE

```
import javax.swing.JOptionPane;
import java.sql.*;
public class home extends javax.swing.JFrame {
    public home() {
        initComponents();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new candidates().setVisible(true);
        this.dispose();
    }
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        newvoter n=new newvoter();
        n.setVisible(true);
        this.dispose();
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
       new Suggestion().setVisible(true);
       this.dispose();
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
          try (Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");
         Statement stmt = conn.createStatement()) {
        ResultSet rs = stmt.executeQuery("SELECT status FROM result");
        if (rs.next()) {
            int status = rs.getInt("status");
            if (status == 1) {
                new results().setVisible(true);
                this.dispose();
            } else {
                JOptionPane.showMessageDialog(this, "Election results have not been
released yet.");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Election results have not been
released yet.");
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Failed to fetch election results status.");
    }
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

        voterid v=new voterid();
        v.setVisible(true);
        this.dispose();
    }

    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        loginpage l=new loginpage();
        l.setVisible(true);
        this.dispose();
    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new home().setVisible(true);
            }
        });
    }
}
```

ADMIN HOME PAGE

```
import javax.swing.JOptionPane;
import java.sql.*;
import java.time.LocalDate;
public class home1 extends javax.swing.JFrame {
    public home1() {
        initComponents();
    }
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
       newvoterr n=new newvoterr();
```

```
        n.setVisible(true);
        this.dispose();
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
       LocalDate currentDate = LocalDate.now();
        LocalDate targetDate = LocalDate.of(2024, 5, 27);

        if (currentDate.isAfter(targetDate)) {
            int response = JOptionPane.showConfirmDialog(null, "Do you want to
enable the result?", "Enable Result", JOptionPane.YES_NO_OPTION);
            if (response == JOptionPane.YES_OPTION) {
                try {
                    Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");
                    String updateSql = "UPDATE result SET status = 1";
                    PreparedStatement pst = con.prepareStatement(updateSql);
                    pst.executeUpdate();
                    pst.close();
                    con.close();
                } catch (SQLException ex) {
                    JOptionPane.showMessageDialog(null, ex.getMessage());
                }
            }
        }
        new results().setVisible(true);
        this.dispose();
    }

    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
          loginpage l=new loginpage();
        l.setVisible(true);
        this.dispose();
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        new viewq().setVisible(true);
        this.dispose();
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new addcand().setVisible(true);
        this.dispose();
    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new home1().setVisible(true);
            }
        });
    }
}
```

NEW VOTER REGISTRATION

```
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.imageio.*;
import javax.mail.*;
import javax.swing.*;

public class newvoter extends javax.swing.JFrame {
    String ot;
    File selectedFile;
    List<AadharData> aadharDataList;

    int k=0;
    public newvoter() {
        initComponents();
        jLabel9.setText("");
        jTextField7.setEditable(false);
        jTextField1.setEditable(false);
        jTextField4.setEditable(false);
        jTextField5.setEditable(false);
        loadAadharList();
    }
        private void registerVoter() {
        try {
          if (selectedFile == null) {
             JOptionPane.showMessageDialog(null, "Please upload an image.");
             return;
          }

          String epicNo = generateEPICNumber();
          String name = jTextField7.getText();
          String dob = jTextField1.getText();
          String gender = jTextField4.getText();
          String constituency = getConstituency(jTextField5.getText());
          String gmailID = jTextField2.getText();
          String aadharId = jTextField6.getText();

          FileInputStream fis = new FileInputStream(selectedFile);
          ByteArrayOutputStream bos = new ByteArrayOutputStream();
          byte[] buffer = new byte[4096];
          int bytesRead;
          while ((bytesRead = fis.read(buffer)) != -1) {
             bos.write(buffer, 0, bytesRead);
          }
          byte[] imageData = bos.toByteArray();
          fis.close();
          bos.close();

          Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");
          String sql = "INSERT INTO voterrequest (epicno, name, dob, gender,
constituency, gmailid, status, image, aadharid) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
          PreparedStatement pstmt = conn.prepareStatement(sql);
          pstmt.setString(1, epicNo);
          pstmt.setString(2, name);
          pstmt.setString(3, dob);
          pstmt.setString(4, gender);
          pstmt.setString(5, constituency);
          pstmt.setString(6, gmailID);
          pstmt.setInt(7, 0);
          pstmt.setBytes(8, imageData);
          pstmt.setString(9, aadharId);
          pstmt.executeUpdate();
          conn.close();
          JOptionPane.showMessageDialog(null, "Voter request submitted
successfully with EPIC Number: " + epicNo);
       } catch (Exception e) {
          e.printStackTrace();
          JOptionPane.showMessageDialog(null, "Error registering voter.");
        }
    }
    private static final String EPIC_CHARACTERS =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    private static final int EPIC_LENGTH = 8;
    private static final Random random = new SecureRandom();
    private String generateEPICNumber() {
        StringBuilder epicNumber = new StringBuilder(EPIC_LENGTH);
        for (int i = 0; i < EPIC_LENGTH; i++) {

epicNumber.append(EPIC_CHARACTERS.charAt(random.nextInt(EPIC_CHAR
ACTERS.length())));
        }
        return epicNumber.toString();
    }

    public static String generateOTP(int length) {
    String numbers = "0123456789";
    Random rndm_method = new Random();
    char[] otp = new char[length];
    for (int i = 0; i < length; i++) {
        otp[i] = numbers.charAt(rndm_method.nextInt(numbers.length()));
    }
    System.out.print(otp);
    return new String(otp);
}
    private static class AadharData {
        private String aadharId;
        private String name;
        private String dob;
        private String gender;
        private String area;

        public AadharData(String aadharId, String name,String d,String g,String a) {
            this.aadharId = aadharId;
            this.name = name;
            this.dob=d;
            this.gender=g;
            this.area=a;
        }

        public String getAadharId() {
            return aadharId;
        }

        public String getName() {
            return name;
        }

        public String getDob() {
            return dob;
        }

        public void setDob(String dob) {
            this.dob = dob;
        }

        public String getGender() {
            return gender;
        }
        public String getArea() {
            return area;
        }
        public void setGender(String gender) {
            this.gender = gender;
        }
    }
    private void loadAadharList() {
        try {
            aadharDataList = new ArrayList<>();
            Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");
            String sql = "SELECT * FROM aadharlist";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                String aadharId = rs.getString("aadharid");
                String name = rs.getString("name");
                String d=rs.getString("dob");
                String g=rs.getString("gender");
                String a=rs.getString("area");
                AadharData data = new AadharData(aadharId, name,d,g,a);
                aadharDataList.add(data);
            }
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error loading Aadhar data.");
        }
    }
    private boolean checkGmailIDExists(String gmailID) {
    try {
        Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");
        String sql = "SELECT * FROM voterlist WHERE gmailid = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, gmailID);
        ResultSet rs = pstmt.executeQuery();
        boolean exists = rs.next();
        conn.close();
        return exists;
    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error checking Gmail ID.");
        return false;
    }
}

private String getConstituency(String area) {
    try {
        Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");
        String sql = "SELECT cname FROM area WHERE name = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, area);
        ResultSet rs = pstmt.executeQuery();
        String constituency = rs.next() ? rs.getString("cname") : "";
        conn.close();
        return constituency;
    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error determining constituency.");
        return "";
    }
}
```

```
    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
}

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       String recipientEmail = jTextField2.getText();
        if (checkGmailIDExists(recipientEmail)) {
        JOptionPane.showMessageDialog(null, "This Gmail ID is already registered.
Please use a different email address.");
        return;
        }
     else
        {
        String senderEmail = "dineshkesav07@gmail.com";
        String senderPassword = "fyjocqnzsqntgpgy";
          Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");
        Session session = Session.getInstance(props, new Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(senderEmail, senderPassword);
            }
        });
        try {
            ot=generateOTP(6);
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(senderEmail));
            message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipientEmail));
            message.setSubject("VOTER REGISTRATION");
            message.setText("YOUR OTP FOR VOTER REGISTRATION IS  :
"+ot);
            Transport.send(message);
            JOptionPane.showMessageDialog(null,"Email sent successfully to " +
recipientEmail);
        } catch (MessagingException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null,"Failed to send email. Check your
network connection and try again.");
        }}
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        if(k==1)
            registerVoter();
        else
        {
            JOptionPane.showMessageDialog(null,"Verify the otp to register  .");
        }
    }

    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        String ott=jTextField3.getText();
        if(ot.equals(ott))
        {
            k=1;
            JOptionPane.showMessageDialog(null,"OTP VERIFIED.");
        }
        else
        {
            JOptionPane.showMessageDialog(null,"Invalid OTP . Try Again .");
            jTextField3.setText("");
        }
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
    fileChooser.setFileFilter(new FileNameExtensionFilter("Image files",
ImageIO.getReaderFileSuffixes()));
     int returnValue = fileChooser.showOpenDialog(null);
    if (returnValue == JFileChooser.APPROVE_OPTION) {
        selectedFile = fileChooser.getSelectedFile();
        try {
            BufferedImage image = ImageIO.read(selectedFile);
            if (image != null) {
                ImageIcon icon = new
ImageIcon(image.getScaledInstance(jLabel8.getWidth(), jLabel8.getHeight(),
Image.SCALE_SMOOTH));
                jLabel8.setIcon(icon);
            } else {
                JOptionPane.showMessageDialog(this, "Failed to load image.");
            }
        } catch (IOException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Failed to load image.");
        }
    }
    }

    private void jTextField6ActionPerformed(java.awt.event.ActionEvent evt) {
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
                String enteredText = jTextField6.getText();
    boolean found = false;
   for (AadharData data : aadharDataList) {
        if (data.getAadharId().equalsIgnoreCase(enteredText)) {
             jTextField7.setText(data.getName());
            jTextField1.setText(data.getDob());
            jTextField4.setText(data.getGender());
            jTextField5.setText(data.getArea());
            found = true;
            break;
        }
    }
    if(!found)
        JOptionPane.showMessageDialog(this, "Sorry ! You are not eligible for
voting .");
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        home h=new home();
        h.setVisible(true);
        this.dispose();
    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new newvoter().setVisible(true);
            }
        });
    }
}
```

VIEW CANDIDATES

```
import java.sql.*;
import javax.swing.JOptionPane;
import java.awt.Image;
import javax.swing.ImageIcon;
public class candidates extends javax.swing.JFrame {
    public candidates() {
        initComponents();
        jComboBox2.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "ADMK", "DMK", "BJP", "INC" }));
        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Madurai", "Chennai", "Coimbatore", "Vellore" }));
    }

    private void getDetailsActionPerformed(java.awt.event.ActionEvent evt) {
        String selectedConstituency = (String) jComboBox1.getSelectedItem();
        String selectedParty = (String) jComboBox2.getSelectedItem();

        String url = "jdbc:oracle:thin:@localhost:1521:xe";
        String username = "system";
        String password = "dinesh07";
        String query = "SELECT * FROM candidates WHERE constituency = ? AND
pname = ?";
        try (Connection conn = DriverManager.getConnection(url, username,
password);
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, selectedConstituency);
            stmt.setString(2, selectedParty);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                jLabel3.setText(rs.getString("cname"));
                jLabel4.setText(rs.getString("constituency"));
                jLabel6.setText(rs.getString("pname"));
                Blob candidateBlob = rs.getBlob("cimage");

          if (candidateBlob != null) {
                    byte[] candidateImageBytes = candidateBlob.getBytes(1, (int)
candidateBlob.length());
                    ImageIcon candidateImageIcon = new
ImageIcon(candidateImageBytes);
                    Image candidateOriginalImage = candidateImageIcon.getImage();
                    int candidateWidth = jLabel2.getWidth();
                    int candidateHeight = jLabel2.getHeight();
                Image candidateResizedImage =
candidateOriginalImage.getScaledInstance(candidateWidth, candidateHeight,
Image.SCALE_SMOOTH);
                ImageIcon candidateResizedImageIcon = new
ImageIcon(candidateResizedImage);
                jLabel2.setIcon(candidateResizedImageIcon);
             }
             Blob partyBlob = rs.getBlob("pimage");
             if (partyBlob != null) {
                byte[] partyImageBytes = partyBlob.getBytes(1, (int)
partyBlob.length());
                ImageIcon partyImageIcon = new ImageIcon(partyImageBytes);
                Image partyOriginalImage = partyImageIcon.getImage();
                int partyWidth = jLabel7.getWidth();
                int partyHeight = jLabel7.getHeight();
                Image partyResizedImage =
partyOriginalImage.getScaledInstance(partyWidth, partyHeight,
Image.SCALE_SMOOTH);
                ImageIcon partyResizedImageIcon = new
ImageIcon(partyResizedImage);
                jLabel7.setIcon(partyResizedImageIcon);
             }
          } else {
             JOptionPane.showMessageDialog(this, "No candidate found for the
selected constituency and party.", "No Results",
JOptionPane.INFORMATION_MESSAGE);
          }
       } catch (SQLException e) {
          e.printStackTrace();
          JOptionPane.showMessageDialog(this, "Error fetching data. Please try
again.", "Error", JOptionPane.ERROR_MESSAGE);
       }
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
             getDetailsActionPerformed(evt);
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        home h=new home();
       h.setVisible(true);
       this.dispose();
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new candidates().setVisible(true);
            }
        });
    }
}
```

VIEW VOTER ID

```
import java.awt.Image;
import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
public class voterid extends javax.swing.JFrame {

    public voterid(){
        String epicNumber = JOptionPane.showInputDialog(null, "Enter EPIC
Number   :   ");
        initComponents();
        if (epicNumber != null && !epicNumber.isEmpty()) {
            String url = "jdbc:oracle:thin:@localhost:1521:xe";
            String user = "system";
            String password = "dinesh07";
            try (Connection conn = DriverManager.getConnection(url, user, password))
          {
                String query = "SELECT * FROM voterlist WHERE epicno = ?";
                PreparedStatement pstmt = conn.prepareStatement(query);
                String epp = epicNumber.toUpperCase();
                pstmt.setString(1, epp);
                ResultSet rs = pstmt.executeQuery();
                if (rs.next()) {
                    this.setVisible(true);
                    String name = rs.getString("name");
                    String dob = rs.getString("dob");
                    String gender = rs.getString("gender");
                    String constituency = rs.getString("constituency");
                    jLabel1.setText(epp);
                    jLabel3.setText(name);
                    jLabel4.setText(dob);
                    jLabel5.setText(gender);
                    jLabel6.setText(constituency);
                    Blob blob = rs.getBlob("image");
                    if (blob != null) {
                        byte[] imageBytes = blob.getBytes(1, (int) blob.length());
                        ImageIcon imageIcon = new ImageIcon(imageBytes);
                        Image originalImage = imageIcon.getImage();
                        int newWidth = jLabel2.getWidth();
                        int newHeight = jLabel2.getHeight();
                        Image resizedImage = originalImage.getScaledInstance(newWidth,
newHeight, Image.SCALE_SMOOTH);
                        ImageIcon resizedImageIcon = new ImageIcon(resizedImage);
                        jLabel2.setIcon(resizedImageIcon);
                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Incorrect EPIC number or
Voter ID not approved.");
                    this.setVisible(false);
                    this.dispose();
                    home h = new home();
                    h.setVisible(true);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else {
            this.setVisible(false);
            this.dispose();
            home h = new home();
            h.setVisible(true);
        }
    }
  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new home().setVisible(true);
        this.dispose();
    }
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                    new voterid();
            }
        });
    }
}
```

VOTING PAGE

```
import javax.swing.JOptionPane;
import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
public class voting extends javax.swing.JFrame {

    String epicNumber;
    Connection conn;
    public voting() {
        initComponents();
        epicNumber = null;
        boolean voterFound = false;
        conn = null;
        try {
            String url = "jdbc:oracle:thin:@localhost:1521:xe";
            String user = "system";
            String password = "dinesh07";
            conn = DriverManager.getConnection(url, user, password);
            while (!voterFound) {
                epicNumber = JOptionPane.showInputDialog(null, "It's time to
vote.\n\nEnter EPIC Number:");
                if (epicNumber == null || epicNumber.isEmpty()) {
                    int response = JOptionPane.showConfirmDialog(null, "EPIC Number
is required. Do you want to exit?", "Exit Confirmation",
JOptionPane.YES_NO_OPTION);
                    if (response == JOptionPane.YES_OPTION)
                        System.exit(0);
                    else
                        continue;
               }
                String query = "SELECT * FROM voterlist WHERE epicno = ?";
                PreparedStatement pstmt = conn.prepareStatement(query);
                String epp = epicNumber.toUpperCase();
                pstmt.setString(1, epp);
                ResultSet rs = pstmt.executeQuery();
                if (rs.next()) {
                    voterFound = true;
                    String constituency = rs.getString("constituency");
                    String aadharId = rs.getString("aadharid");
                    loadCandidates(constituency);
                    updateUvreqStatus(aadharId);
                } else {
                    JOptionPane.showMessageDialog(null, "Voter not found. Please try
again.");
                }
            }
        } catch (SQLException ex) {
            Logger.getLogger(voting.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
            Runtime.getRuntime().addShutdownHook(new Thread(() -> {
                try {
                    if (conn != null && !conn.isClosed()) {
                        conn.close();
                    }
                } catch (SQLException ex) {
                    Logger.getLogger(voting.class.getName()).log(Level.SEVERE, null,
ex);
                }
            }));
        }
    }
    private void loadCandidates(String constituency) throws SQLException {
        String query = "SELECT cname, cimage, pname, pimage FROM candidates
WHERE constituency = ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, constituency);
        ResultSet rs = pstmt.executeQuery();

        int candidateIndex = 1;
        while (rs.next() && candidateIndex <= 4) {

           String cname = rs.getString("cname");
            Blob cimageBlob = rs.getBlob("cimage");
            String pname = rs.getString("pname");
            Blob pimageBlob = rs.getBlob("pimage");
            byte[] cimageBytes = cimageBlob.getBytes(1, (int) cimageBlob.length());
            ImageIcon cimageIcon = new ImageIcon(cimageBytes);
```

```
            Image cimage =
cimageIcon.getImage().getScaledInstance(jLabel1.getWidth(), jLabel1.getHeight(),
Image.SCALE_SMOOTH);
            byte[] pimageBytes = pimageBlob.getBytes(1, (int) pimageBlob.length());
            ImageIcon pimageIcon = new ImageIcon(pimageBytes);
            Image pimage =
pimageIcon.getImage().getScaledInstance(jLabel6.getWidth(),
jLabel6.getHeight(), Image.SCALE_SMOOTH);
            switch (candidateIndex) {
                case 1:
                    jButton1.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            int confirm = JOptionPane.showConfirmDialog(null, "Are you
sure you want to vote for " + cname + " of " + pname + "?", "Confirmation",
JOptionPane.YES_NO_OPTION);
                            if (confirm == JOptionPane.YES_OPTION) {
                                    addVote(epicNumber, constituency, cname, pname);
                            }
                        }
                    });
                    jLabel1.setIcon(new ImageIcon(cimage));
                    jLabel5.setText(cname);
                    jLabel13.setText(pname);
                    break;
                case 2:
                    jButton2.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            int confirm = JOptionPane.showConfirmDialog(null, "Are you
sure you want to vote for " + cname + " of " + pname + "?", "Confirmation",
JOptionPane.YES_NO_OPTION);
                            if (confirm == JOptionPane.YES_OPTION)
                                        addVote(epicNumber, constituency, cname, pname);
                            }
                        }
                    });
                    jLabel2.setIcon(new ImageIcon(cimage));
                    jLabel7.setText(cname);
                    jLabel14.setText(pname);
                    break;
                case 3:
                    jButton3.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            int confirm = JOptionPane.showConfirmDialog(null, "Are you
sure you want to vote for " + cname + " of " + pname + "?", "Confirmation",
JOptionPane.YES_NO_OPTION);
                            if (confirm == JOptionPane.YES_OPTION)
                                addVote(epicNumber, constituency, cname, pname);
                        }
                    });
                    jLabel3.setIcon(new ImageIcon(cimage));
                    jLabel8.setText(cname);
                    jLabel15.setText(pname);
                    break;
                case 4:
                    jButton4.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            int confirm = JOptionPane.showConfirmDialog(null, "Are you
sure you want to vote for " + cname + " of " + pname + "?", "Confirmation",
JOptionPane.YES_NO_OPTION);
                            if (confirm == JOptionPane.YES_OPTION)
                                        addVote(epicNumber, constituency, cname, pname);
                        }
                    });
                    jLabel4.setIcon(new ImageIcon(cimage));
                    jLabel9.setText(cname);
                    jLabel16.setText(pname);
                    break;
            }
            switch (candidateIndex) {
                case 1:
                    jLabel6.setIcon(new ImageIcon(pimage));
                    break;
                case 2:
                    jLabel10.setIcon(new ImageIcon(pimage));
                    break;
                case 3:
                    jLabel11.setIcon(new ImageIcon(pimage));
                    break;
                case 4:
                    jLabel12.setIcon(new ImageIcon(pimage));
                    break;
            }
            candidateIndex++;
}
```

```
    }

    private void addVote(String epicNumber, String constituency, String cname,
String pname) throws SQLException {
         String insertVotingQuery = "INSERT INTO voting (epicno, constituency,
cname, pname) VALUES (?, ?, ?, ?)";
        PreparedStatement pstmtVoting = conn.prepareStatement(insertVotingQuery);
        pstmtVoting.setString(1, epicNumber);
        pstmtVoting.setString(2, constituency);
        pstmtVoting.setString(3, cname);
        pstmtVoting.setString(4, pname);
        pstmtVoting.executeUpdate();
         String updateResultsQuery = "UPDATE results SET count = count + 1
WHERE cname = ? AND pname = ? AND constituency = ?";
        PreparedStatement pstmtResults =
conn.prepareStatement(updateResultsQuery);
        pstmtResults.setString(1, cname);
        pstmtResults.setString(2, pname);
        pstmtResults.setString(3, constituency);
        pstmtResults.executeUpdate();
        JOptionPane.showMessageDialog(null, "Your vote for " + cname + " of " +
pname + " has been recorded.");
        new home().setVisible(true);
        this.dispose();
    }

    private void updateUvreqStatus(String aadharId) throws SQLException {
        String updateQuery = "UPDATE uvreq SET status = 2 WHERE aadharid =
?";
        PreparedStatement pstmt = conn.prepareStatement(updateQuery);
        pstmt.setString(1, aadharId);
        pstmt.executeUpdate();
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
         new home().setVisible(true);
        this.dispose();
    }
public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new voting().setVisible(true);
            }
        });
    }
}
```

ADD CANDIDATES

```
import java.awt.Image;
import java.io.*;
import javax.imageio.ImageIO;
import javax.swing. *;
import java.sql.*;
public class addcand extends javax.swing.JFrame {

    private File candidateImageFile;
    private File partyImageFile;

    public addcand() {
        initComponents();
        jComboBox2.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "ADMK", "DMK", "BJP", "INC" }));
        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Madurai", "Chennai", "Coimbatore", "Vellore" }));
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        home1 h=new home1();
        h.setVisible(true);
        this.dispose();
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        candidateImageFile = uploadImage(jLabel8);
    }
    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        partyImageFile = uploadImage(jLabel9);
         }
private File uploadImage(javax.swing.JLabel label) {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
        fileChooser.setFileFilter(new FileNameExtensionFilter("Image files",
ImageIO.getReaderFileSuffixes()));
        int returnValue = fileChooser.showOpenDialog(null);
        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            try {
                BufferedImage image = ImageIO.read(selectedFile);
                if (image != null) {
                    ImageIcon icon = new
ImageIcon(image.getScaledInstance(label.getWidth(), label.getHeight(),
Image.SCALE_SMOOTH));
                    label.setIcon(icon);
                    return selectedFile;
                } else {
                    JOptionPane.showMessageDialog(this, "Failed to load image.");
                }
            } catch (IOException e) {
                e.printStackTrace();
                JOptionPane.showMessageDialog(this, "Failed to load image.");
            }
        }
        return null;
    }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
       String name = jTextField1.getText();
        String gender = jRadioButton1.isSelected() ? "MALE" :
jRadioButton2.isSelected() ? "FEMALE" : null;
        String constituency = (String) jComboBox1.getSelectedItem();
        String party = (String) jComboBox2.getSelectedItem();

        if (name.isEmpty() || gender == null || constituency.isEmpty() ||
party.isEmpty() || candidateImageFile == null || partyImageFile == null) {
            JOptionPane.showMessageDialog(this, "Please fill all fields and upload
both images.");
            return;
        }

        try (Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");
             PreparedStatement pstmt = conn.prepareStatement("INSERT INTO
candidates (cname, cimage, pname, pimage, constituency) VALUES (?, ?, ?, ?,
?)")) {
      pstmt.setString(1, name);
            FileInputStream fis = new FileInputStream(candidateImageFile);
            pstmt.setBinaryStream(2, fis, (int) candidateImageFile.length());
            pstmt.setString(3, party);
            fis = new FileInputStream(partyImageFile);
            pstmt.setBinaryStream(4, fis, (int) partyImageFile.length());
            pstmt.setString(5, constituency);
            pstmt.executeUpdate();
            JOptionPane.showMessageDialog(this, "Candidate added successfully.");
        } catch (SQLException | IOException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Failed to add candidate.");
        }
    }

    public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new addcand().setVisible(true);
            }
        });
    }
}
```

APPROVAL OF VOTER ID

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableModel;
public class newvoterr extends javax.swing.JFrame {
    Connection con=null;
    PreparedStatement st=null;
    ResultSet rs=null;
    String lowerCaseNamePart;
    public newvoterr() {
        initComponents();
        try {
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","di
nesh07");
```

```
          String sql="select * from voterrequest where status=?";
          st=con.prepareStatement(sql);
          st.setInt(1,0);
          rs=st.executeQuery();
          DefaultTableModel model = (DefaultTableModel) histable.getModel();
          model.setRowCount(0);
          while (rs.next()) {
             Object[] row = {
                    rs.getString("aadharid"),
                    rs.getString("name"),
                    rs.getString("dob"),
                    rs.getString("gender"),
                    rs.getString("constituency"),
             };
             model.addRow(row);
          }

       } catch (SQLException ex) {
          Logger.getLogger(newvoterr.class.getName()).log(Level.SEVERE, null,
ex);
       }
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       int selectedRow = histable.getSelectedRow();
    if (selectedRow != -1) {
       DefaultTableModel model = (DefaultTableModel) histable.getModel();
       String aadharId = model.getValueAt(selectedRow, 0).toString();
       String name = model.getValueAt(selectedRow, 1).toString();
       String dob = model.getValueAt(selectedRow, 2).toString();
       String gender = model.getValueAt(selectedRow, 3).toString();
       String constituency = model.getValueAt(selectedRow, 4).toString();
         lowerCaseNamePart = name.split(" ")[0].toLowerCase();
         String epicNo = null;
       String gmailId = null;
       byte[] image = null;

       try {
         con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"system", "dinesh07");
         String query = "SELECT * FROM voterrequest WHERE aadharid = ?";
          st = con.prepareStatement(query);
          st.setString(1, aadharId);
          rs = st.executeQuery();
          if (rs.next()) {
            epicNo = rs.getString("epicno");
            gmailId = rs.getString("gmailid");
            image = rs.getBytes("image");
         }
         con.setAutoCommit(false);
        String insertUVReqSQL = "INSERT INTO uvreq (username, aadharid,
status) VALUES (?, ?, ?)";
          st = con.prepareStatement(insertUVReqSQL);
          st.setString(1, lowerCaseNamePart);
          st.setString(2, aadharId);
          st.setInt(3, 0);
          st.executeUpdate();
        String insertVoterListSQL = "INSERT INTO voterlist (aadharid, name, dob,
gender, constituency, epicno, gmailid, image) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
          st = con.prepareStatement(insertVoterListSQL);
          st.setString(1, aadharId);
          st.setString(2, name);
          st.setString(3, dob);
          st.setString(4, gender);
          st.setString(5, constituency);
          st.setString(6, epicNo);
          st.setString(7, gmailId);
          st.setBytes(8, image);
          st.executeUpdate();
          String deleteSQL = "DELETE FROM voterrequest WHERE aadharid = ?";
          st = con.prepareStatement(deleteSQL);
          st.setString(1, aadharId);
          st.executeUpdate();

          con.commit();
          model.removeRow(selectedRow);
      } catch (SQLException ex) {
          try {
              if (con != null) {
                  con.rollback();
              }
          } catch (SQLException e) {
              Logger.getLogger(newvoterr.class.getName()).log(Level.SEVERE, null,
e);
          }
          Logger.getLogger(newvoterr.class.getName()).log(Level.SEVERE, null,
ex);
      } finally {
          try {
              if (rs != null) rs.close();
              if (st != null) st.close();
              if (con != null) con.close();
          } catch (SQLException ex) {
              Logger.getLogger(newvoterr.class.getName()).log(Level.SEVERE, null,
ex);
          }
      }
  }
  }
  private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
         int selectedRow = histable.getSelectedRow();
  if (selectedRow != -1) {
      DefaultTableModel model = (DefaultTableModel) histable.getModel();
      String aadharId = model.getValueAt(selectedRow, 0).toString();
      String name = model.getValueAt(selectedRow, 1).toString();
        lowerCaseNamePart = name.split(" ")[0].toLowerCase();

      try {
          con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"dinesh07");

          // Delete the record from voterrequest
          String deleteSQL = "delete from voterrequest where aadharid = ?";
          st = con.prepareStatement(deleteSQL);
          st.setString(1, aadharId);
          st.executeUpdate();

          // Insert into uvreq
          String insertUVReqSQL = "INSERT INTO uvreq (username, aadharid,
status) VALUES (?, ?, ?)";
          st = con.prepareStatement(insertUVReqSQL);
          st.setString(1, lowerCaseNamePart);
          st.setString(2, aadharId);
          st.setInt(3, 5); // Status for rejected is 0
          st.executeUpdate();

          model.removeRow(selectedRow);
      } catch (SQLException ex) {
          Logger.getLogger(newvoterr.class.getName()).log(Level.SEVERE, null,
ex);
      } finally {
          try {
```

```
            if (st != null) {
                st.close();
            }
            if (con != null) {
                con.close();
            }
        } catch (SQLException ex) {
            Logger.getLogger(newvoterr.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
  }
  }
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      new home1().setVisible(true);
      this.dispose();
  }
  public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
              new newvoterr().setVisible(true);
          }
      });
  }
}
```

RESULTS

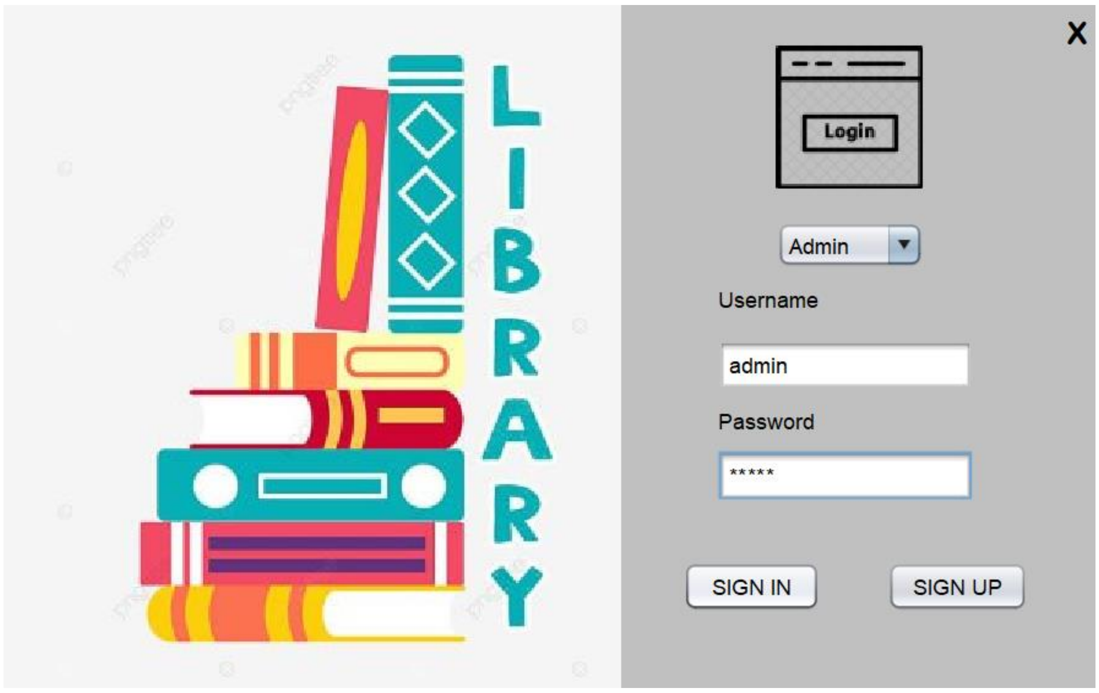
```
import java.sql.*;
import javax.swing.ImageIcon;
import javax.swing.*;
import java.awt.*;
import javax.swing.table.DefaultTableModel;
public class results extends javax.swing.JFrame {

  DefaultTableModel model;
  public results() {
      initComponents();
      jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[]{"Madurai", "Chennai", "Coimbatore", "Vellore"}));
      model = new DefaultTableModel(new Object[]{"CANDIDATE", "PARTY",
"VOTE COUNT"}, 0);
```

```
        jTable2.setModel(model);
        loadDataForConstituency((String) jComboBox1.getSelectedItem());
    }
    private void loadDataForConstituency(String constituency) {
    model.setRowCount(0);
    try (Connection conn = 
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", 
"dinesh07")) {
        String query = "SELECT cname, pname, count FROM results WHERE 
constituency = ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, constituency);
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            String cname = rs.getString("cname");
            String pname = rs.getString("pname");
            int count = rs.getInt("count");
            model.addRow(new Object[]{cname, pname, count});
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
          String selectedConstituency = (String) jComboBox1.getSelectedItem();
            loadDataForConstituency(selectedConstituency);
    }
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new results().setVisible(true);
            }
        });
    }
}
```

4. SCREENSHOTS

LOGIN PAGE



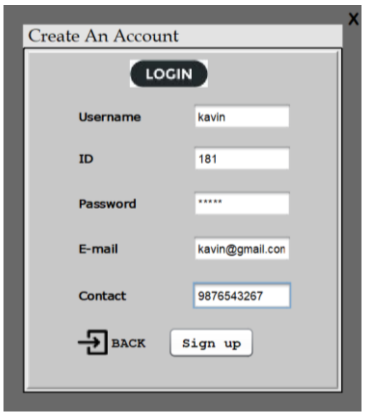
Username

admin

Password

*****

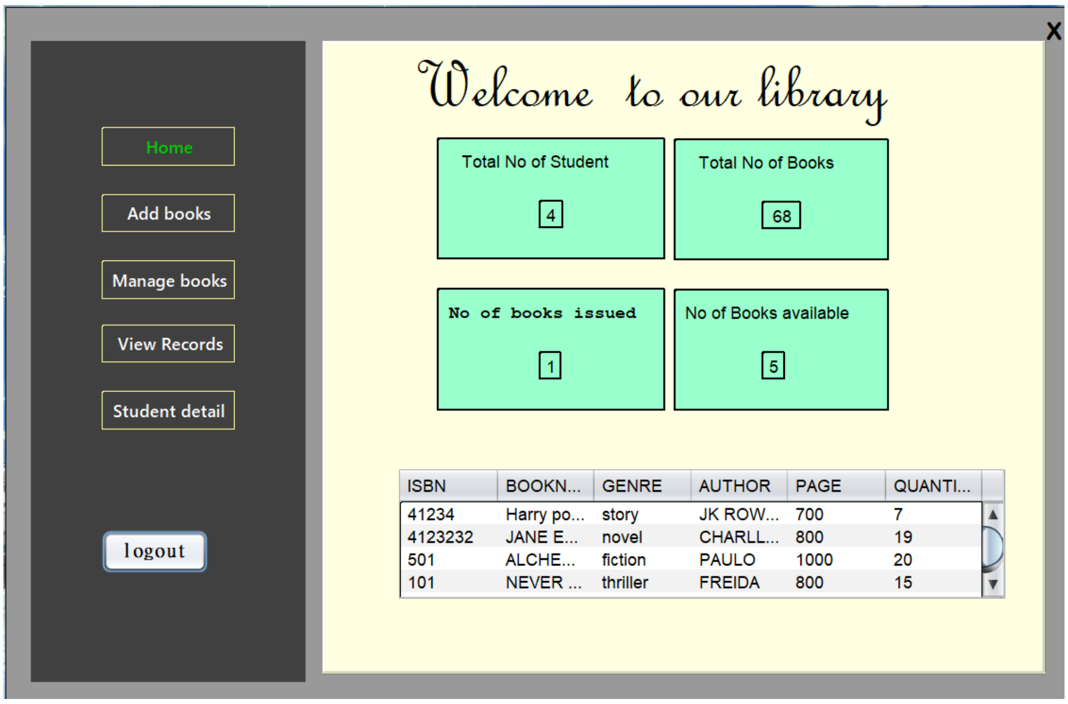
SIGN UP PAGE:



Create An Account

Username	kavin
ID	181
Password	*****
E-mail	kavin@gmail.con
Contact	9876543267

ADMIN HOME PAGE :



Welcome to our library

Total No of Student 4

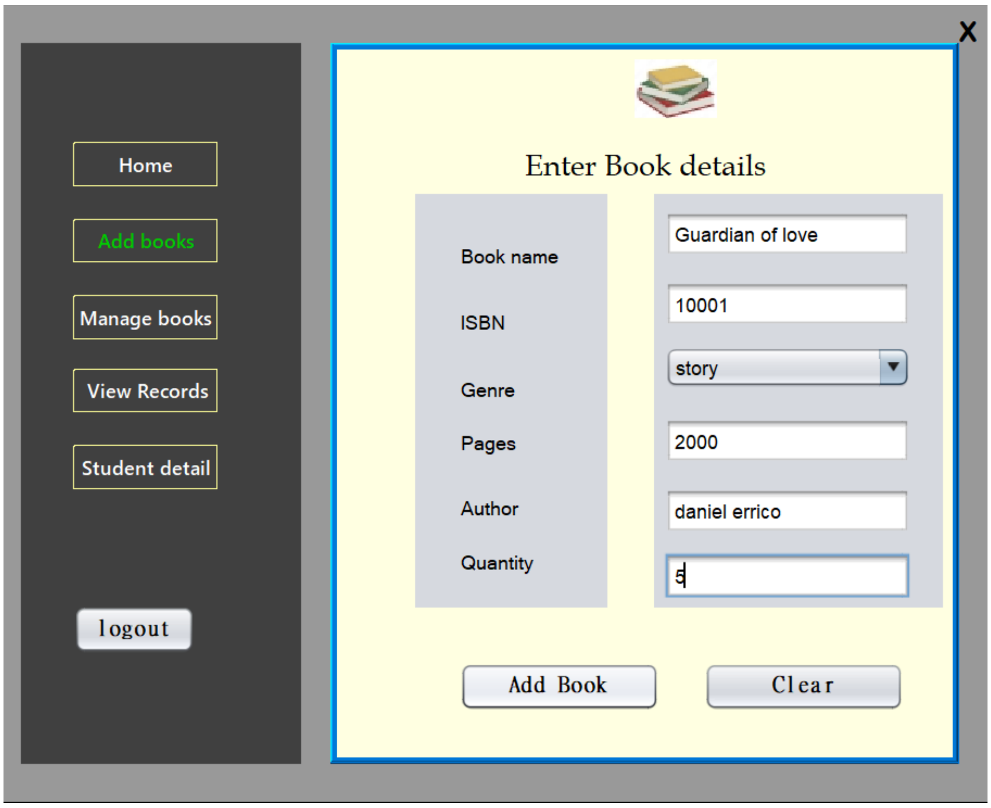
Total No of Books 68

No of books issued 1

No of Books available 5

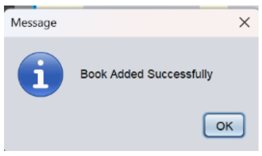
ISBN	BOOKN...	GENRE	AUTHOR	PAGE	QUANTI...
41234	Harry po...	story	JK ROW...	700	7
4123232	JANE E...	novel	CHARLL...	800	19
501	ALCHE...	fiction	PAULO	1000	20
101	NEVER ...	thriller	FREIDA	800	15

ADDING BOOK :



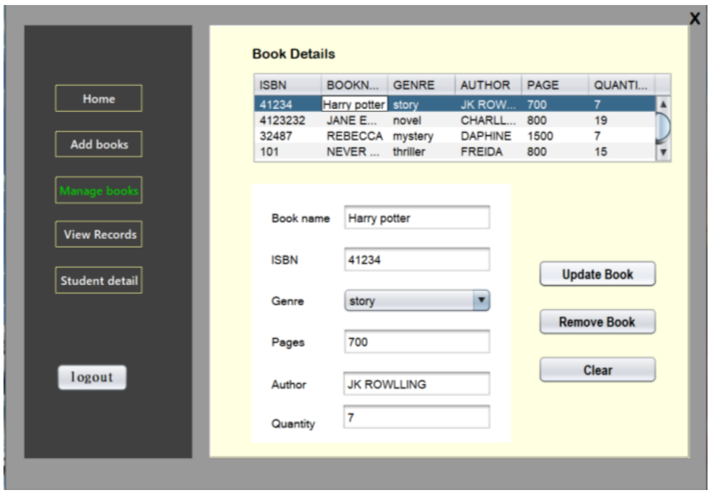
Enter Book details

Book name	Guardian of love
ISBN	10001
Genre	story
Pages	2000
Author	daniel errico
Quantity	5



Book Added Successfully

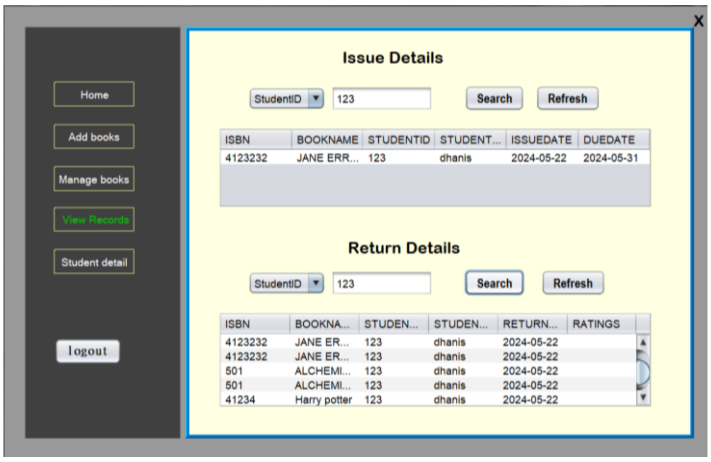
MANAGING BOOK :



Book Details

ISBN	BOOKN...	GENRE	AUTHOR	PAGE	QUANTI...
41234	Harry potter	story	JK ROW...	700	7
4123232	JANE E...	novel	CHARLL...	800	19
32487	REBECCA	mystery	DAPHINE	1500	7
101	NEVER ...	thriller	FREIDA	800	15

Book name: Harry potter
ISBN: 41234
Genre: story
Pages: 700
Author: JK ROWLLING
Quantity: 7



Issue Details

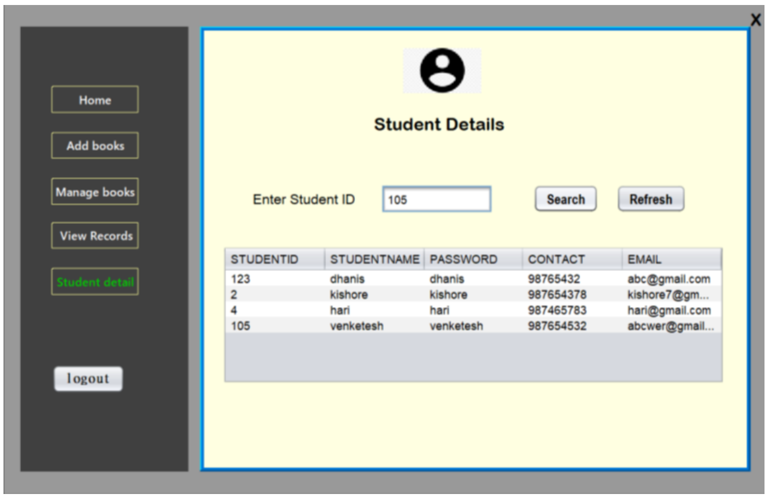
StudentID: 123

ISBN	BOOKNAME	STUDENTID	STUDENT...	ISSUEDATE	DUEDATE
4123232	JANE ERR...	123	dhanis	2024-05-22	2024-05-31

Return Details

StudentID: 123

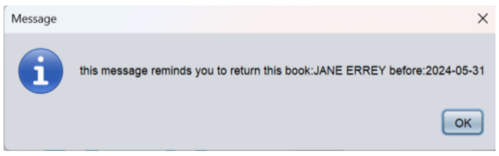
ISBN	BOOKNA...	STUDEN...	STUDEN...	RETURN...	RATINGS
4123232	JANE ER...	123	dhanis	2024-05-22	
4123232	JANE ER...	123	dhanis	2024-05-22	
501	ALCHEMI...	123	dhanis	2024-05-22	
501	ALCHEMI...	123	dhanis	2024-05-22	
41234	Harry potter	123	dhanis	2024-05-22	



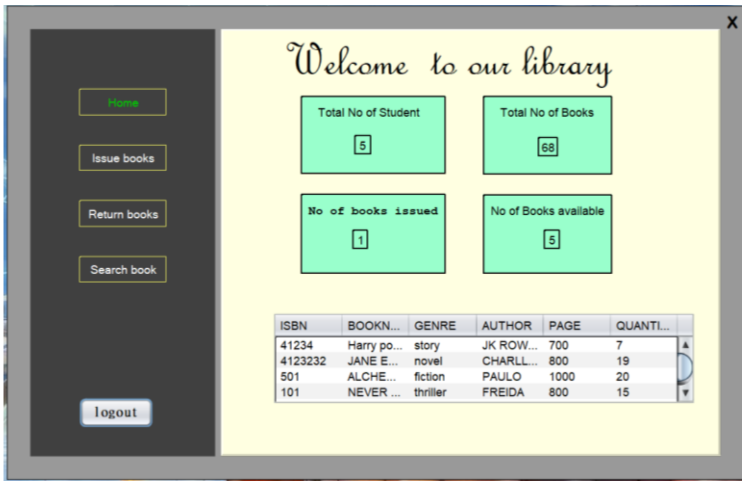
Student Details

Enter Student ID: 105

STUDENTID	STUDENTNAME	PASSWORD	CONTACT	EMAIL
123	dhanis	dhanis	98765432	abc@gmail.com
2	kishore	kishore	987654378	kishore7@gm...
4	hari	hari	987465783	hari@gmail.com
105	venketesh	venketesh	987654532	abcwer@gmail...



this message reminds you to return this book:JANE ERREY before:2024-05-31



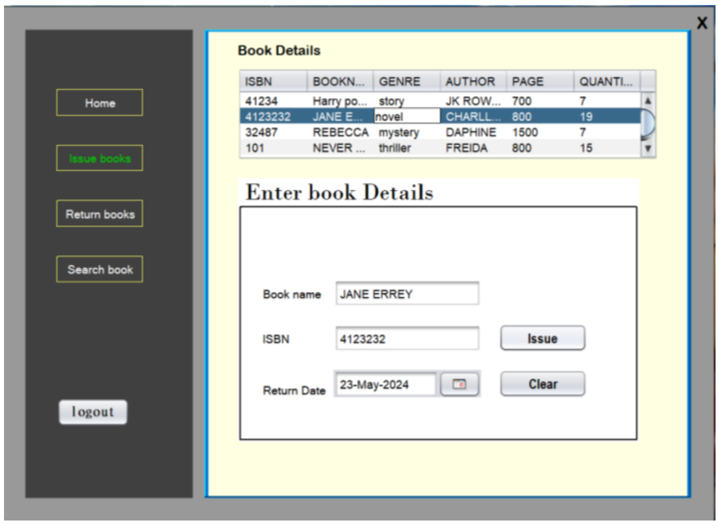
Total No of Student: 5

Total No of Books: 68

No of books issued: 1

No of Books available: 5

ISBN	BOOKN...	GENRE	AUTHOR	PAGE	QUANTI...
41234	Harry po...	story	JK ROW...	700	7
4123232	JANE E...	novel	CHARLL...	800	19
501	ALCHE...	fiction	PAULO	1000	20
101	NEVER ...	thriller	FREIDA	800	15



Book Details

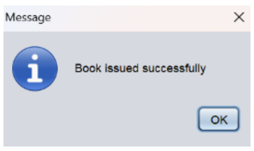
ISBN	BOOKN...	GENRE	AUTHOR	PAGE	QUANTI...
41234	Harry po...	story	JK ROW...	700	7
4123232	JANE E...	novel	CHARLL...	800	19
32487	REBECCA	mystery	DAPHINE	1500	7
101	NEVER ...	thriller	FREIDA	800	15

Enter book Details

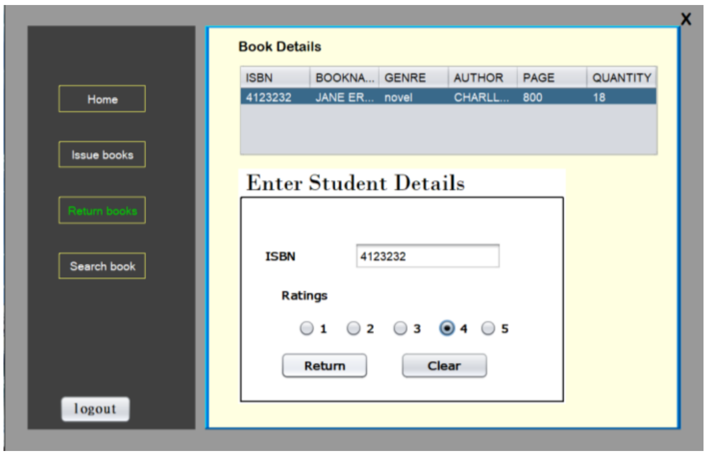
Book name: JANE ERREY

ISBN: 4123232

Return Date: 23-May-2024



Book issued successfully



Book Details

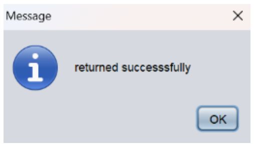
ISBN	BOOKNA...	GENRE	AUTHOR	PAGE	QUANTITY
4123232	JANE ER...	novel	CHARLL...	800	18

Enter Student Details

ISBN 4123232

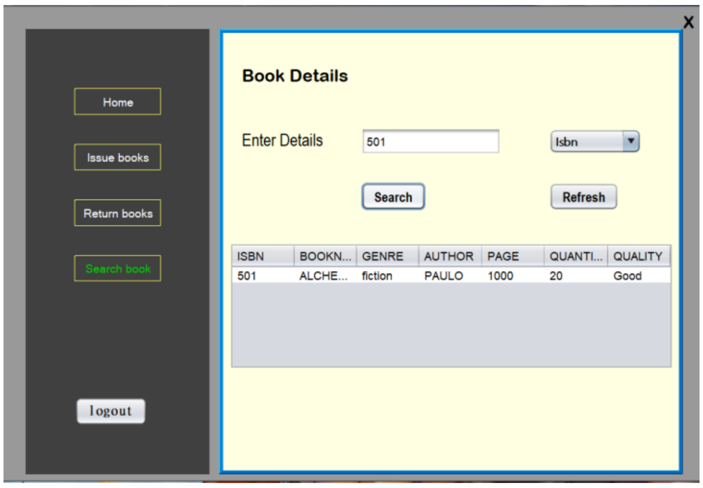
Ratings

1 2 3 [✓ 4] 5



Message

returned successsfully



Book Details

Enter Details 501 Isbn

ISBN	BOOKN...	GENRE	AUTHOR	PAGE	QUANTI...	QUALITY
501	ALCHE...	fiction	PAULO	1000	20	Good

5.CONCLUSION

The development of this podcast website using React for the frontend and Java Spring Boot for the backend represents a significant achievement in creating a modern, user-friendly platform for podcast enthusiasts. By leveraging the strengths of these technologies, we have successfully built a scalable, responsive, and feature-rich application that meets the diverse needs of its users.

In conclusion, this project underscores the successful integration of modern web technologies to deliver a superior podcasting experience. The collaboration between React and Java Spring Boot has resulted in a dynamic, user-centric platform poised to cater to the evolving needs of podcast listeners and content creators alike. Moving forward, continuous improvements and feature expansions will ensure the platform remains relevant and valuable to its user base.