MINIPROJECT:- PASSWORD MANAGER

CS23332-DATABASE MANAGEMENT SYSTEM

Submitted by:

Abhinav.K – 241001001
Abishek.V.S-241001004
Agathiyan M - 241001007

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM
(An Autonomous Institution)

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this project titled "Password Manger" is the Bonafide work of **"Abhinav.K (241001001),Agathiyan.M(241001007), Abishek(241001004)"**who carried out the project work under my supervision.

SIGNATURE
Dr.Valarmathie
HEAD OF THE DEPARTMENT
Information Technology
Rajalakshmi Engineering College,
Rajalakshmi Nagar, Thandalam
Chennai – 602105

SIGNATURE
Mrs.Ragavardhini
COURSE INCHARGE
Information Technology
Rajalakshmi Engineering College
Rajalakshmi Nagar, Thandalam
Chennai – 602105

This project is submitted for CS23332– DATABASE MANAGEMENT SYSTEM held on

_____

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

TABLE OF CONTENTS

PASSWORD MANAGER:

1.1 Abstract:

This project is a desktop-based Password Manager application developed using Java Swing and MySQL. The system allows users to securely register, log in, and manage their saved passwords in an organized dashboard. Core functionalities include adding, editing, deleting, and viewing stored passwords for various websites. The application uses a MySQL database for persistent storage and ensures multi-user access through user-based authentication. Its simple interface and CRUD operations help users efficiently manage their daily login credentials in one place.

.

1.2 Introduction:

In the digital era, individuals use multiple online services and platforms, each requiring unique login credentials. Remembering all these passwords can be challenging and may compromise security if written down or reused. This project aims to provide a simple, local, and effective Password Manager using Java Swing as the UI framework and MySQL as the backend database. The system allows users to store and manage passwords securely and conveniently. It includes key features such as login/registration, password storage, and CRUD operations through an intuitive graphical interface.

1.3 Purpose:

The purpose of this project is:

- To create a simple and user-friendly password management tool.
- To provide users with an organized dashboard to store and manage their login credentials.
- To ensure password data is securely stored in a database and accessible only to authenticated users.
- To demonstrate Java Swing GUI design and database connectivity using JDBC.
- To implement CRUD operations for a practical desktop application.

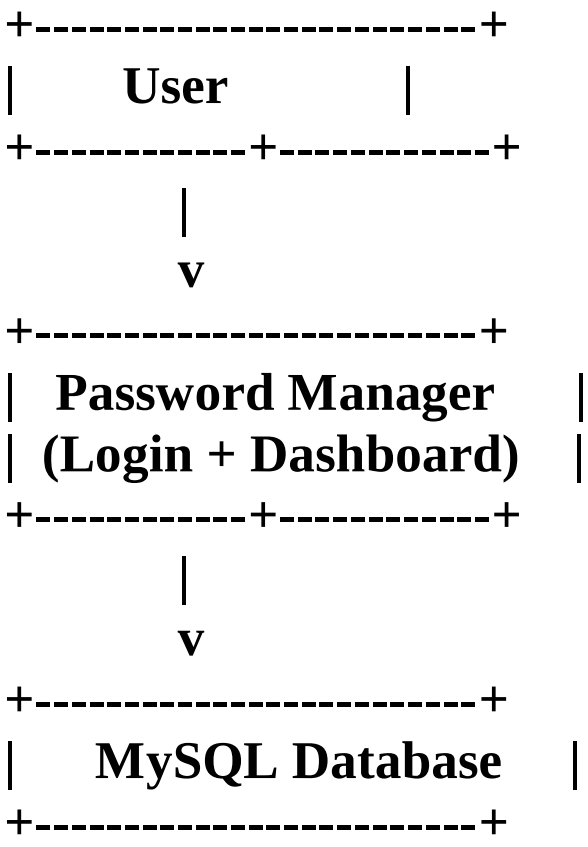
1.4 Scope of the Project:

Included in Scope

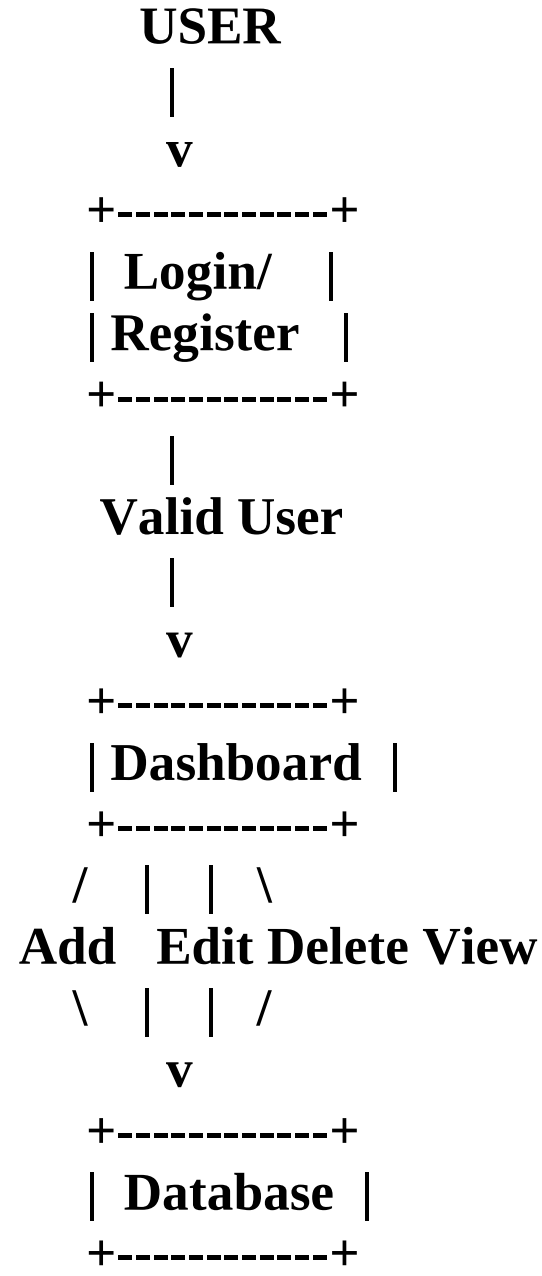
- User registration and login authentication.
- Dashboard to manage stored passwords.
- Add, edit, delete, and view password entries.
- MySQL database integration using JDBC.
- Simple and attractive GUI interface using Java Swing.
- Multi-user support, each user having their own set of passwords.

1.5 System Flow Diagrams:

Level 0 – Context Diagram:



Level 1 DFD


USER
Login/
Register
Valid User
Dashboard
Add
Edit
Delete
View
Database

1.6 software requirement specification :

A. Functional Requirements:

1. **User Registration**
   - User can register with username and password.
   - System stores data in users table.
2. **User Login**
   - User must log in using valid credentials.
   - Invalid login should show an error message.
3. **Dashboard**
   - User can view all saved passwords.
   - Passwords are filtered by user_id.
4. **Add Password**
   - User can add site name, username, and password.
5. **Edit Password**
   - User can edit existing stored passwords.
6. **Delete Password**
   - User can delete any saved credential.
7. **Logout**
   - User is returned to login window.

B. Non-Functional Requirements:

1. **Usability**
   - Interface must be easy to use and understand.
2. **Performance**
   - Application should respond quickly to CRUD operations.
3. **Security**
   - Only authenticated users can access saved passwords.
   - MySQL database handles proper user-specific data.
4. **Reliability**
   - System must maintain stored data even after application closes.
5. **Maintainability**
   - Code is modular and follows Java standards.

C. Software Requirements:

- Programming Language: Java (JDK 8+)
- IDE: IntelliJ IDEA / Eclipse / NetBeans
- GUI Library: Java Swing
- Database: MySQL 5.7 / 8.0

1.7 Module Description:

1. Login & Registration Module

- Allows new users to register.
- Authenticates existing users.
- Connects to users table in database.
- On successful login → opens Dashboard.

2. Dashboard Module

- Displays all stored passwords in JTable.
- Connects to passwords table.
- Fetches data based on logged-in userId.

3. Password Management Module

Includes:

- Add Password – Inserts new row into database.
- Edit Password – Updates selected row.
- Delete Password – Removes selected entry.
- Uses prepared statements to avoid SQL injection.

4. Database Connection Module

- Contains DBConnection.java
- Manages MySQL connections using JDBC.
- Provides a reusable getConnection() method.

1.8 Implementation:

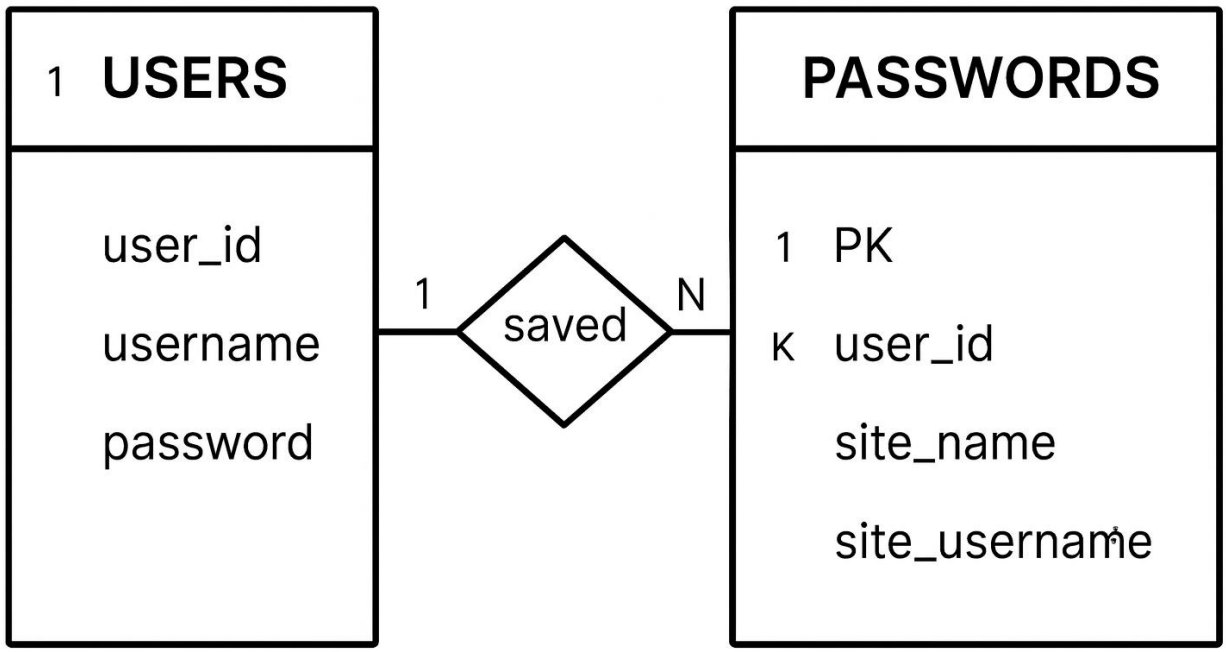
- The project is implemented using Java Swing for UI and MySQL for backend storage.
- LoginRegister.java handles:
  - Login page UI design
  - Register page UI design
  - Query execution for authentication
- Dashboard.java handles:
  - Table creation & styling using JTable
  - CRUD operations (insert, update, delete)
  - Loading password data into table
- DBConnection.java provides:
  - JDBC connection to the local MySQL database
- Uses PreparedStatement for secure database interactions.

- Code is modular, easy to understand, and separated by packages ui and db.

1.9 Entity-relationship diagram:

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in the table.

Entity -relationship diagram:



Introduction:

In the digital era, individuals use multiple online services and platforms, each requiring unique login credentials. Remembering all these passwords can be challenging and may compromise security if written down or reused. This project aims to provide a simple, local, and effective Password Manager using Java Swing as the UI framework and MySQL as the backend database. The system allows users to store and manage passwords securely and conveniently. It includes key features such as login/registration, password storage, and CRUD operations through an intuitive graphical interface.

Document Purpose:

The purpose of this project is:

- To create a simple and user-friendly password management tool.
- To provide users with an organized dashboard to store and manage their login credentials.
- To ensure password data is securely stored in a database and accessible only to authenticated users.
- To demonstrate Java Swing GUI design and database connectivity using JDBC.
- To implement CRUD operations for a practical desktop application.

Product Scope:

- User registration and login authentication.
- Dashboard to manage stored passwords.
- Add, edit, delete, and view password entries.
- MySQL database integration using JDBC.
- Simple and attractive GUI interface using Java Swing.
- Multi-user support, each user having their own set of passwords.

Overall Description:

The Password Manager is a standalone desktop application developed using Java for the front-end and MySQL for data storage. Its primary purpose is to help users securely store, manage, and retrieve their website login credentials in one convenient location. The system offers an easy-to-use interface that allows users to register, log in, and maintain a personal database of their passwords.

Product Perspective:

The Password Manager system is a **standalone desktop application** designed to help users securely store and manage their website login credentials. It is developed using **Java Swing** for the graphical interface and **MySQL** for backend data storage. The system functions independently and does not rely on any external services, making it suitable for offline usage.

Product Functionality:

The Password Manager system provides a set of core functionalities that allow users to securely manage their login credentials through an intuitive Java Swing interface.

User and Characteristics:

The Password Manager system is designed for individuals who need a secure and organized way to store their website login credentials. The users of this system are non-technical or moderately technical, and the interface is developed to be simple and easy to understand.

Operating Environment:

Hardware Requirements:
- Processor: Any Processor over i3
- Operating System: Windows 8, 10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

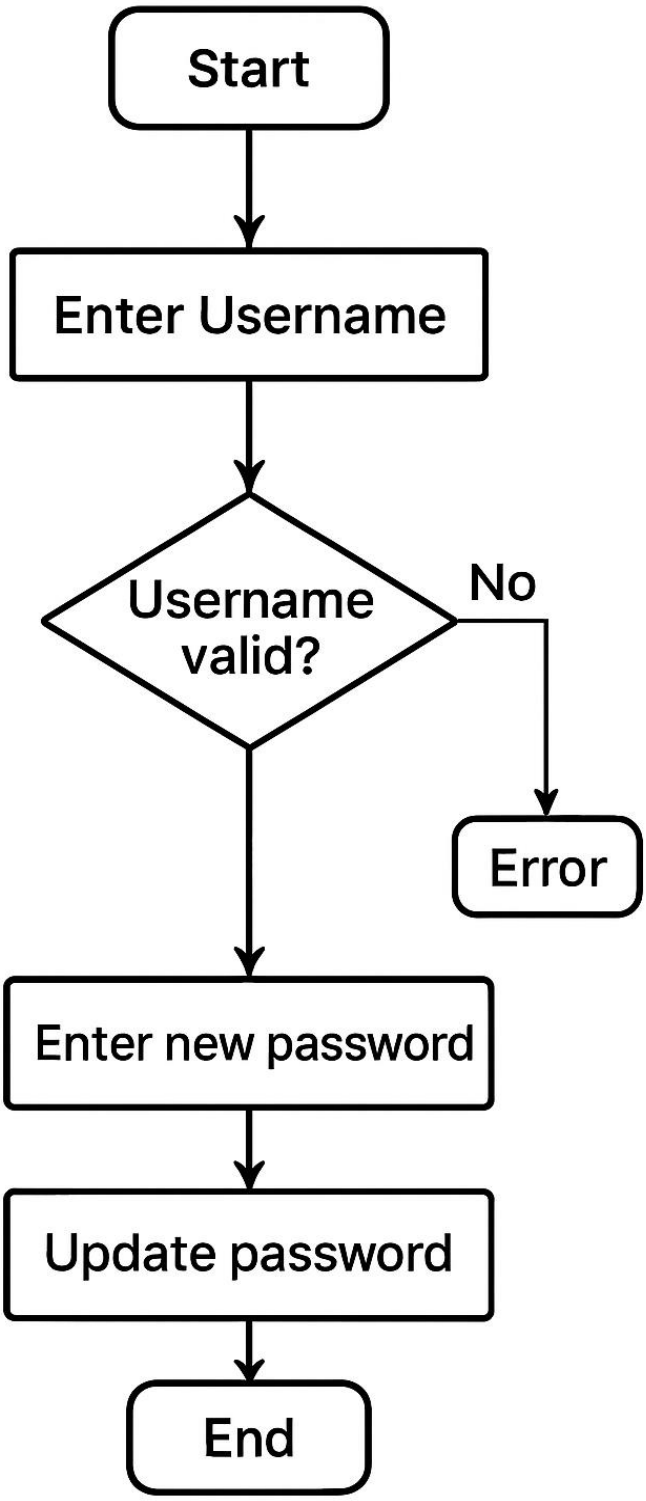
Software Requirements:

- Database: MySQL
- Frontend: Java (SWING, AWT)
- Technology: Java (JDBC)

Software Interface:

a) MS-Windows Operating System
b) Java AWT and SWING for designing the front end
c) MySQL for the backend
d) Platform: Java Language
e) Integrated Development Environment (IDE): NetBeans

2.0 Data flow diagram:



2.1 CODES :

LOGIN REGISTER.JAVA:

```
package ui;

import db.DBConnection;

import javax.swing.*;
import java.awt.*;
import java.sql.*;

public class LoginRegister extends JFrame {

    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginBtn, registerBtn;

    public LoginRegister() {
        setTitle("🔐 Password Manager - Login/Register");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JLabel title = new JLabel("Password Manager", SwingConstants.CENTER);
        title.setFont(new Font("Arial", Font.BOLD, 24));
        title.setForeground(new Color(0, 102, 204));
        add(title, BorderLayout.NORTH);

        JPanel formPanel = new JPanel(new GridLayout(2, 2, 10, 10));
        formPanel.setBorder(BorderFactory.createEmptyBorder(30, 50, 30, 50));
        formPanel.setBackground(new Color(224, 224, 224));

        formPanel.add(new JLabel("Username:"));
        usernameField = new JTextField();
        formPanel.add(usernameField);

        formPanel.add(new JLabel("Password:"));
        passwordField = new JPasswordField();
        formPanel.add(passwordField);
```

```
        add(formPanel, BorderLayout.CENTER);

        JPanel btnPanel = new JPanel();
        btnPanel.setLayout(new FlowLayout());
        btnPanel.setBackground(new Color(224, 224, 224));

        loginBtn = new JButton("Login");
        loginBtn.setBackground(new Color(0, 153, 76));
        loginBtn.setForeground(Color.WHITE);
        loginBtn.setFocusPainted(false);

        registerBtn = new JButton("Register");
        registerBtn.setBackground(new Color(255, 102, 102));
        registerBtn.setForeground(Color.WHITE);
        registerBtn.setFocusPainted(false);

        btnPanel.add(loginBtn);
        btnPanel.add(registerBtn);

        add(btnPanel, BorderLayout.SOUTH);

        loginBtn.addActionListener(e -> loginUser());
        registerBtn.addActionListener(e -> registerUser());

        setVisible(true);
    }

    private void loginUser() {
        String username = usernameField.getText();
        String password = String.valueOf(passwordField.getPassword());

        try (Connection conn = DBConnection.getConnection()) {
            String sql = "SELECT * FROM users WHERE username=? AND
password=?";
            PreparedStatement pst = conn.prepareStatement(sql);
            pst.setString(1, username);
            pst.setString(2, password);
            ResultSet rs = pst.executeQuery();

            if (rs.next()) {
                int userId = rs.getInt("user_id");  // ✅ corrected
                JOptionPane.showMessageDialog(this, "🎉 Login Successful!");
                new Dashboard(userId);           // open dashboard
```

```
            this.dispose();                 // close login window
        } else {
            JOptionPane.showMessageDialog(this, " ✗ Invalid credentials!");
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void registerUser() {
    String username = usernameField.getText();
    String password = String.valueOf(passwordField.getPassword());

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "INSERT INTO users(username,password) VALUES(?,?)";
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setString(1, username);
        pst.setString(2, password);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, " ✓ Registration Successful!");
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, " ✗ Username already exists!");
    }
}

public static void main(String[] args) {
    new LoginRegister();
}
}
```

DB CONNECTION.JAVA:

```
package db;

import java.sql.*;

public class DBConnection {
    private static final String URL =
"jdbc:mysql://localhost:3306/password_manager";
    private static final String USER = "root";
    private static final String PASSWORD = "harish2006";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

DASHBOARD.JAVA:

```
package ui;

import db.DBConnection;

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

public class Dashboard extends JFrame {
    private JTable table;
    private DefaultTableModel model;
    private int userId;

    public Dashboard(int userId) {
        this.userId = userId;

        setTitle("🌈 Password Manager Dashboard");
        setSize(700, 450);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Table
        model = new DefaultTableModel(new String[]{"ID", "Site", "Username", "Password"}, 0);
        table = new JTable(model);
        table.setRowHeight(25);
        table.setFillsViewportHeight(true);
        table.setBackground(new Color(240, 248, 255));
        table.setGridColor(new Color(200, 200, 200));
        add(new JScrollPane(table), BorderLayout.CENTER);

        // Bottom panel
        JPanel bottomPanel = new JPanel(new GridLayout(2, 5, 5, 5));
        bottomPanel.setBackground(new Color(224, 224, 224));

        JTextField siteField = new JTextField();
        JTextField siteUserField = new JTextField();
        JTextField sitePassField = new JTextField();

        JButton addBtn = new JButton("Add");
        addBtn.setBackground(new Color(0, 153, 76));
        addBtn.setForeground(Color.WHITE);
        addBtn.setFocusPainted(false);
```

```
JButton editBtn = new JButton("Edit Selected");
editBtn.setBackground(new Color(255, 153, 0));
editBtn.setForeground(Color.WHITE);
editBtn.setFocusPainted(false);

JButton deleteBtn = new JButton("Delete Selected");
deleteBtn.setBackground(new Color(255, 51, 51));
deleteBtn.setForeground(Color.WHITE);
deleteBtn.setFocusPainted(false);

JButton logoutBtn = new JButton("Logout");
logoutBtn.setBackground(new Color(102, 102, 255));
logoutBtn.setForeground(Color.WHITE);
logoutBtn.setFocusPainted(false);

bottomPanel.add(new JLabel("Site:"));
bottomPanel.add(siteField);
bottomPanel.add(new JLabel("Username:"));
bottomPanel.add(siteUserField);
bottomPanel.add(new JLabel("Password:"));
bottomPanel.add(sitePassField);
bottomPanel.add(addBtn);
bottomPanel.add(editBtn);
bottomPanel.add(deleteBtn);
bottomPanel.add(logoutBtn);

add(bottomPanel, BorderLayout.SOUTH);

loadPasswords();

// Add new password
addBtn.addActionListener(e -> {
    String site = siteField.getText();
    String username = siteUserField.getText();
    String password = sitePassField.getText();
    if (site.isEmpty() || username.isEmpty() || password.isEmpty()) {
        JOptionPane.showMessageDialog(this, "❌ Fill all fields!");
        return;
    }
    addPassword(site, username, password);
    loadPasswords();
    siteField.setText("");
    siteUserField.setText("");
    sitePassField.setText("");
});

// Edit selected password
editBtn.addActionListener(e -> {
    int selectedRow = table.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "❌ Select a row to edit!");
```

```
            return;
        }
        int id = (int) model.getValueAt(selectedRow, 0);
        String site = JOptionPane.showInputDialog(this, "Enter new Site:",
model.getValueAt(selectedRow, 1));
        String username = JOptionPane.showInputDialog(this, "Enter new Username:",
model.getValueAt(selectedRow, 2));
        String password = JOptionPane.showInputDialog(this, "Enter new Password:",
model.getValueAt(selectedRow, 3));
        if (site != null && username != null && password != null) {
            editPassword(id, site, username, password);
            loadPasswords();
        }
    });

    // Delete selected password
    deleteBtn.addActionListener(e -> {
        int selectedRow = table.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "❌ Select a row to delete!");
            return;
        }
        int id = (int) model.getValueAt(selectedRow, 0);
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure to delete this
password?", "Confirm", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            deletePassword(id);
            loadPasswords();
        }
    });

    // Logout
    logoutBtn.addActionListener(e -> {
        new LoginRegister();
        this.dispose();
    });

    setVisible(true);
}

private void loadPasswords() {
    model.setRowCount(0);
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM passwords WHERE user_id=?";
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setInt(1, userId);
        ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            model.addRow(new Object[]{
                rs.getInt("id"),
                rs.getString("site_name"),
```

```
                rs.getString("site_username"),
                rs.getString("site_password")
            });
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void addPassword(String site, String username, String password) {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "INSERT INTO passwords(user_id, site_name, site_username,
site_password) VALUES(?, ?, ?, ?)";
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setInt(1, userId);
        pst.setString(2, site);
        pst.setString(3, username);
        pst.setString(4, password);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "✅ Password added!");
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "❌ Error adding password: " +
ex.getMessage());
    }
}

private void editPassword(int id, String site, String username, String password) {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "UPDATE passwords SET site_name=?, site_username=?, site_password=?
WHERE id=?";
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setString(1, site);
        pst.setString(2, username);
        pst.setString(3, password);
        pst.setInt(4, id);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "✅ Password updated!");
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "❌ Error updating password: " +
ex.getMessage());
    }
}

private void deletePassword(int id) {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "DELETE FROM passwords WHERE id=?";
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setInt(1, id);
```

```
            pst.executeUpdate();
            JOptionPane.showMessageDialog(this, "✅ Password deleted!");
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "❌ Error deleting password: " +
ex.getMessage());
        }
    }
}
```

ggyu

2.1 Database Design:

The database design for the Password Manager system focuses on securely storing user credentials and their corresponding password entries. The design follows a **relational database model** using **MySQL**, ensuring data integrity, normalization, and efficient access.

The database consists of two main tables: **Users** and **Passwords**. These tables are linked through a **one-to-many relationship**, where a single user can have multiple password entries saved in the system. The design ensures separation of user information from the stored passwords, allowing secure, scalable storage.

Users Table:

This table stores the login information of each registered user.

Field Name	Data Type	Description
user_id (PK)	INT (Auto Increment)	Unique identifier for each user
username	VARCHAR(100)	User's login name
password	VARCHAR(255)	User's login password

Passwords Table:

This table stores the credentials of websites or services saved by the user.

Field Name	Data Type	Description
id (PK)	INT (Auto Increment)	Unique identifier for each password record
user_id (FK)	INT	References user_id from Users table
site_name	VARCHAR(200)	Name of the website/service
site_username	VARCHAR(200)	Username used on the site
site_password	VARCHAR(200)	Password used on the site

Relationship Between Tables:

- Users (1) —— (N) Passwords
  One registered user can store multiple password entries.
  This relationship is implemented using a foreign key (user_id) in the passwords table.

Primary User: General Users

These are individuals who will actively use the password manager to save and retrieve login information.
Characteristics

- Basic computer knowledge.
- Able to operate simple desktop applications.
- Not required to have technical knowledge of Java or MySQL.
- Expect easy navigation and a simple UI.
- Want quick access to stored passwords.

New Users

These are people creating an account for the first time.
Characteristics

- May not be familiar with password managers.
- Need clear guidance during the registration process.
- Expect simple instructions and error messages.

Returning Users

These users have already created an account and will log in regularly.
Characteristics

- Expect fast login and quick access to their stored data.
- Will use features like Add, Edit, Delete, and View passwords frequently.
- Prefer stable and reliable performance.

System Administrator (Optional)

If included in the system setup, a basic administrator may manage the MySQL database.
Characteristics

- Has technical knowledge of MySQL.
- Responsible for database backup and maintenance.
- Ensures the system runs smoothly for end-users.

2.2 Conclusion:

The Password Manager project successfully demonstrates Java Swing GUI development, MySQL database integration, and core CRUD operations. It provides users with a simple yet effective solution to store and manage their passwords securely. The project is modular, scalable, and can be enhanced further by adding features such as encryption, password generation, cloud backup, and two-factor authentication. Overall, this project serves as an excellent example of integrating Java desktop applications with database technology to solve real-world problems.

Reference links:

[1] https://www.javatpoint.com/java-awt
[2] https://www.javatpoint.com/java-swing