CSE Department – Faculty of Engineering - MSA
Spring 2025
GSE122 GSE122i COM265 PROGRAMMING 2
Course Project
Course Instructor:Dr. Ahmed El Anany

Due Date 9/MAY/2025 11:59 PM on E-learning

Discussion inside lecture 18/May till 23/May inside lab as per lab slot

Student Name	Omar Ayman	Student ID	240033
Student Name	Ahmed Sameh	Student ID	240351
Student Name	Eyad Karam	Student ID	241539
Student Name	Jana Mohamed	Student ID	242319
TA Name	Eng. Dina Magdy	Grade:	/

Diary Management System

Table of Contents

Project Overview

Objectives

This diary management system is designed to help people keep track of their daily activities in a simple and organized way. Like having a digital notebook where you can save all your tasks, appointments, and personal notes. You can add new entries with details like what you need to do, how long it will take, where it needs to happen, and when. The system lets you look back at what you've recorded, make changes if plans shift, or remove entries you don't need anymore. Everything gets saved securely in a database, so your information stays safe and doesn't get lost. Whether you're keeping a personal journal, tracking work tasks, or planning future events, this system helps you stay organized with minimal effort. It's a practical tool that combines good programming with everyday usefulness.

Roles and Responsibilities

Code: worked on by Omar and Ahmed.
Code review and troubleshooting: Eyad and Jana.
Report: worked on by Eyad.
Powerpoint: worked on by Jana.
Github: worked on by all members of group.

Algorithm and external libraries

- Algorithms:

1- User Authentication: *where it matches string for string for validation, and checks for empty or duplicated fields during registration.*

2- Data handling**:** *converts database records into Jtable rows, while also making button actions trigger database operations.*

3- Database operations: *SQL query execution for inserting new diaries, updating them, or retrieving them.*

- Libraries:

1- Java.awt.event.MouseAdapter: *it is an abstract helper that handles mouse activites such as (mouse clicked and mouse released).*

2- Java.awt.event.MouseEvent: *this library represents events triggered by mouse such as clicking, moving, or dragging, works by providing the mouse's coordinates.*

3- Java.sql.*: *this library contains the Java's Database Connectivity(JDBC), enabling interaction with relational databases(executes SQL).*

4- Java.util.ArrayList: *an array which you can add, remove, or get data from, unlike default arrays.*

5- Java.util.list: *used for managing structured data in java, implementation of ArrayList, extends the collection interface and declares methods like get, remove, size.*

6- Java.swing.*: *a set of GUI components for creating cross platform desktop applications, include JFrame, JLabel, etc (The standard library for java GUI).*

7- Javax.swing.table.DefaultTableModel: *used to manage JTable components, by storing data in the form of rows and columns, and provide methods such as addRow, removeRow, simplifies editing and displaying data.*

8- Java.awt.*: *the foundational GUI toolkit for Java, provides core components such as button, label, frame.*

GUI and Database Usage

The User Interface

The application has a simple graphical interface built with Java's Swing library. It starts with a login window containing username and password fields, plus login and register buttons. After successful login, users see the main diary management screen. This screen shows a form for entering new diary entries at the top, with fields for task name, location, duration, date, time and details. Below the form is a table displaying all saved entries, and at the bottom are buttons for adding, editing, deleting entries, changing password, and logging out. The interface uses basic Swing components like JTextField for input, JTable for displaying entries, and JButton for actions.

Database Structure

The application uses a MySQL database with two main tables. The 'user' table stores account information with columns for user ID, username, and password. The 'diary' table contains all diary entries with columns for entry ID, task name, duration, location, date, time, details, and a user ID that links each entry to its owner. The database follows a simple relational design where each diary entry belongs to one user through the user ID foreign key.

Connecting the Interface to Data

When users interact with the GUI, their actions trigger database operations. For example, clicking "Add" takes the form data, creates a new diary object, and inserts it into the database. The table view automatically updates by querying the database for all entries matching the current user's ID. The application uses JDBC to handle all database communications, with prepared statements to safely execute queries and prevent SQL injection. Each database operation opens a connection, performs the query, then closes the connection.

Code explaining

Classes used and their roles in the project:

1- DataBaseConnection: *Class that handles MYSQL and database connections, providing a static connection with URL, username, and password, uses JDBC to connect to MYSQL database.*

2- User and Password manager: *Class that manages user data using their id, username, and password.*

3- Diary: *class that manages diaries using fields of (name, duration, address, and date), containing getters and setter functions for all fields.*

4-Login manager: *class that authenticates users (login, register, change password), and maintains user security.*

5-Record manager: *class that manages records of diaries by adding, editing, deleting, or viewing components in the diary; uses databaseHandler for database operations.*

6-Database handler: *class that contains all SQL Queries and all database operations, it also handles CRUD for diary and user entries.*

7-Diary GUI: *the main gui class containing nested frame classes, also contains loginframe and mainmenuframe, and implements all user interactions and form validations.*

Code:

```
mport java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;

// this is the class that contains all we need for the database connection
class DataBaseConnection
```

```
{
    // storing/initializing URL, USER, PASS as constant variables
    private static final String URL = "jdbc:mysql://localhost:3306/DiarySystem?serverTimezone=UTC";

    private static final String USER = "root";
    private static final String PASS = "";

    // returning a database connection object so that we can use in our program where we need data from the database

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


// this class contain and manage user's data
class User
{
    private final int id;
    private final String username;
    private final PasswordManager passwordManager;

    // class constructor for initializing attributes
    public User(int id, String username, String password)
    {
        this.id = id;
        this.username = username;
        this.passwordManager = new PasswordManager(password);
    }

    // setters and getters
    public int getId() { return id; }
    public String getUsername() { return username; }
    public String getPassword() { return passwordManager.getPassword(); }
    public void setPassword(String password) { this.passwordManager.setPassword(password); }
}

// this class contains what we need from setters and getters to manage passwords and update it
class PasswordManager
{
    private String password;
    // class constructor for initializing attributes
    public PasswordManager(String password) { this.password = password; }
```

```
    // getter for the user's pasword
    public String getPassword() { return password; }
    // setter for setting the password for the user
    public void setPassword(String password) { this.password = password; }
}

// this class is for managing user diaries
class Diary
{
    // defining class attributes
    private final int id;
    private String name;
    private String duration;
    private String address;
    private String date;
    private String time;
    private String details;
    private final int userId;

    // using the class constructor for initializing
    public Diary(int id, String name, String duration, String address, String date, String time, String details, int userId)
    {
        this.id = id;
        this.name = name;
        this.duration = duration;
        this.address = address;
        this.date = date;
        this.time = time;
        this.details = details;
        this.userId = userId;
    }

    // getters
    public int getId() { return id; }
    public String getName() { return name; }
    public String getDuration() { return duration; }
    public String getAddress() { return address; }
    public String getDate() { return date; }
    public String getTime() { return time; }
    public String getDetails() { return details; }
    public int getUserId() { return userId; }

    // setters
    public void setName(String name) { this.name = name; }
```

```
    public void setDuration(String duration) { this.duration = duration; }
    public void setAddress(String address) { this.address = address; }
    public void setDate(String date) { this.date = date; }
    public void setTime(String time) { this.time = time; }
    public void setDetails(String details) { this.details = details; }
}



class LoginManager {// the relation between "LoginManager"& "DatabaseHandler" -> aggregation
    private final DatabaseHandler dbHandler = new DatabaseHandler();                    // creating an instance from
class DatabaseHandler to do database operation to manage users account :

    public User login(String username, String password) throws Exception                // 1- login()is to handle user
login process
    {
        User user = dbHandler.getUserByUsername(username);
        if (user != null && user.getPassword().equals(password))
            return user;
        else
            return null;
    }

    public boolean register(String username, String password) throws Exception {          //register() is to handle user
registeration process
        if (dbHandler.getUserByUsername(username) != null)          //handle the case if username is already used by
another user
            return false;
        else{
        dbHandler.addUser(username, password);
        return true;
        }
    }

    public void changePassword(String username, String newPassword) throws Exception {    // changePassword() is to
handle user changing password process
        dbHandler.updatePassword(username, newPassword);
    }
}



class RecordManager { // the relation between RecordManager class& DatabaseHandler class is aggregation
```

```
    private final DatabaseHandler dbHandler = new DatabaseHandler();              // creating an instance from class
DatabaseHandler to do database operations on user diaries:
    public List<Diary> viewRecord(int userId) throws Exception {            //      1-viewRecord()-> get user diaries from
database for logged in user
        return dbHandler.readDataBase(userId);
    }

    public void addRecord(Diary d) throws Exception {                           //      2- addRecord()-> add a new diary
record
        dbHandler.saveDataBase(d);
    }

    public void updateRecord(Diary d) throws Exception {       //      3- updateRecord()-> edit an existing thing
        dbHandler.updateDataBases(d);
    }

    public void deleteRecord(int id) throws Exception {                          //      4- deleteRecord()-> delete existing
thing
        dbHandler.deleteDiary(id);
    }
}
// DatabaseHandler class
// -handles getting data from database
// -manages database errors
// -opens and closes database
//does the maun database tasks: create new data-read or get data-update existing data-delete data


class DatabaseHandler
{

    public User getUserByUsername(String username) throws SQLException
    {
        String sql = "SELECT * FROM user WHERE username=?";         // sql query to handle user login
        try (Connection conn = DataBaseConnection.getconn();
             PreparedStatement stmt = conn.prepareStatement(sql))
        {
            stmt.setString(1, username);
            ResultSet rs = stmt.executeQuery();
            if (rs.next())
                return new User(rs.getInt("id"), rs.getString("username"), rs.getString("password"));
        }
        return null;
    }
```

```
public void addUser(String username, String password) throws SQLException
{
   String sql = "INSERT INTO user (username, password) VALUES (?, ?)";     //sql query handle user registeration
   try (Connection conn = DataBaseConnection.getconn();
       PreparedStatement stmt = conn.prepareStatement(sql))
   {
      stmt.setString(1, username);
      stmt.setString(2, password);
      stmt.executeUpdate();
   }
}

public void updatePassword(String username, String password) throws SQLException {
   String sql = "UPDATE user SET password=? WHERE username=?";        // sql query handle updating password
   try (Connection conn = DataBaseConnection.getconn();
       PreparedStatement stmt = conn.prepareStatement(sql))
   {
      stmt.setString(1, password);
      stmt.setString(2, username);
      stmt.executeUpdate();
   }
}


public List<Diary> readDataBase(int userId) throws SQLException
{
   List<Diary> list = new ArrayList<>();
   String sql = "SELECT * FROM diary WHERE user_id=?";        // sql query retrive all added diaries
   try (Connection conn = DataBaseConnection.getconn();
       PreparedStatement stmt = conn.prepareStatement(sql))
   {
      stmt.setInt(1, userId);
      ResultSet rs = stmt.executeQuery();
      while (rs.next())
      {
         list.add(new Diary (
               rs.getInt("id"),
               rs.getString("name"),
               rs.getString("duration"),
               rs.getString("address"),
               rs.getString("date"),
               rs.getString("time"),
               rs.getString("details"),
```

```
                rs.getInt("user_id")
            ));
        }
    }
    return list;
}

public void saveDataBase(Diary d) throws SQLException
{
    String sql = "INSERT INTO diary (name, duration, address, date, time, details, user_id) VALUES (?, ?, ?, ?, ?, ?,
?)";
    try (Connection conn = DataBaseConnection.getconn();
         PreparedStatement stmt = conn.prepareStatement(sql))
    {
        stmt.setString(1, d.getName());
        stmt.setString(2, d.getDuration());
        stmt.setString(3, d.getAddress());
        stmt.setString(4, d.getDate());
        stmt.setString(5, d.getTime());
        stmt.setString(6, d.getDetails());
        stmt.setInt(7, d.getUserId());
        stmt.executeUpdate();
    }
}

public void updateDataBases(Diary d) throws SQLException
{
    String sql = "UPDATE diary SET name=?, duration=?, address=?, date=?, time=?, details=? WHERE id=?";
    try (Connection conn = DataBaseConnection.getconn();
         PreparedStatement stmt = conn.prepareStatement(sql))
    {
        stmt.setString(1, d.getName());
        stmt.setString(2, d.getDuration());
        stmt.setString(3, d.getAddress());
        stmt.setString(4, d.getDate());
        stmt.setString(5, d.getTime());
        stmt.setString(6, d.getDetails());
        stmt.setInt(7, d.getId());
        stmt.executeUpdate();
    }
}

public void deleteDiary(int id) throws SQLException
{
```

```
        String sql = "DELETE FROM diary WHERE id=?";
        try (Connection conn = DataBaseConnection.getconn();
            PreparedStatement stmt = conn.prepareStatement(sql))
        {
            stmt.setInt(1, id);
            stmt.executeUpdate();
        }
    }
}

class DiaryGUI {
    private final LoginManager loginManager = new LoginManager();              // creating an instance from class
LoginManager to be able to use login operations on user account:(register- login -change password)
    private final RecordManager recordManager = new RecordManager();          // creating an instance from class
RecordManager to be able to manage user dairies :(add diary-update diary -delete diary)
    private User currentUser;

    // --- entry page ---
    public void showLogin() {
        new LoginFrame();
    }

    // --- LOGIN / REGISTRATION page ---
    class LoginFrame extends JFrame {
        JTextField usernameField = new JTextField(15);
        JPasswordField passwordField = new JPasswordField(15);

        public LoginFrame() {
            setTitle("Login - login page");
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            setSize(400,250);
            setLocationRelativeTo(null);

            JPanel panel = new JPanel(new GridLayout(3,2,5,5));
            panel.add(new JLabel("Username:"));
            panel.add(usernameField);
            panel.add(new JLabel("Password:"));
            panel.add(passwordField);

            JButton loginButton = new JButton("Login");        //login button
            JButton regButton = new JButton("Register");       // register button
            panel.add(loginButton);
            panel.add(regButton);
```

```
        add(panel);

        loginButton.addActionListener(e -> doLogin());
        regButton.addActionListener(e -> doRegister());

        setVisible(true);
    }

    private boolean validateLoginFields(String username, String password)      // boolean function validate user
credintials"username,"password":
    {     // trim()->  ignore accidental spaces                                    //              1- username or password->
null
        if(username == null || username.trim().isEmpty() || password == null || password.trim().isEmpty()) {//        2-
username or password-> empty
            JOptionPane.showMessageDialog(this, "Username and password cannot be empty.");        //        --->>
show error message
            return false;                                                          //              --return false--
        }
        else
            return true;
    }

    private void doLogin() {
        String username = usernameField.getText().trim();             // getText()-> retrieve username from
usernameField
        String password = new String(passwordField.getPassword());
        if (!validateLoginFields(username, password)) return;         // calling validateLoginFields to validate user
credintials
        try {
            User u = loginManager.login(username, password);
            if (u != null) {
                currentUser = u;
                JOptionPane.showMessageDialog(this, "Welcome, " + username + "!");
                dispose();         // --> close login frame
                new MainMenuFrame();
            } else {
                JOptionPane.showMessageDialog(this, "login error check name and password and try again");
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

    private void doRegister() {
```

```
        String username = usernameField.getText().trim();
        String password = new String(passwordField.getPassword());
        if (!validateLoginFields(username, password)) return;
        try {
            if (loginManager.register(username, password)) {
                JOptionPane.showMessageDialog(this, "Registration success! login now");
            } else {
                JOptionPane.showMessageDialog(this, "used username choose another one");
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }
}

// --- MAIN MENU page ---
class MainMenuFrame extends JFrame {
    private final JTextField tfTaskName = new JTextField();                    // text fields
    private final JTextField tfAddress = new JTextField();                    //
    private final JTextField tfDuration = new JTextField();                   //
    private final JTextField tfDate = new JTextField();                       //
    private final JTextField tfTime = new JTextField();                       //
    private final JTextArea taDetails = new JTextArea(3, 20);          //

    private final JTable table;
    private final DefaultTableModel tableModel;

    private final JButton btnAdd = new JButton("Add");                        //BUTTONS
    private final JButton btnEdit = new JButton("Edit");                     //
    private final JButton btnDelete = new JButton("Delete");                  //
    private final JButton btnLogout = new JButton("Logout");                  //
    private final JButton btnPwd = new JButton("Change Password");            //


    private void colorButtons() {
        btnAdd.setOpaque(true);
        btnAdd.setBackground(new Color(76, 175, 80));                         // BUTTON COLOR
        btnAdd.setForeground(Color.WHITE);
        btnAdd.setFocusPainted(false);
        btnAdd.setBorderPainted(false);

        btnEdit.setOpaque(true);
        btnEdit.setBackground(new Color(33, 150, 243));                      // BUTTON COLOR
        btnEdit.setForeground(Color.WHITE);
```

```
    btnEdit.setFocusPainted(false);
    btnEdit.setBorderPainted(false);

    btnDelete.setOpaque(true);
    btnDelete.setBackground(new Color(244, 67, 54));                    // BUTTON COLOR
    btnDelete.setForeground(Color.WHITE);
    btnDelete.setFocusPainted(false);
    btnDelete.setBorderPainted(false);

    btnPwd.setOpaque(true);
    btnPwd.setBackground(new Color(255, 193, 7));                       // BUTTON COLOR
    btnPwd.setForeground(Color.BLACK);
    btnPwd.setFocusPainted(false);
    btnPwd.setBorderPainted(false);

    btnLogout.setOpaque(true);
    btnLogout.setBackground(new Color(255, 0, 0));                    // BUTTON COLOR
    btnLogout.setForeground(Color.WHITE);
    btnLogout.setFocusPainted(false);
    btnLogout.setBorderPainted(false);
}


private List<Diary> loadedDiaries;

public MainMenuFrame() {
    setTitle("Diary Management System - User: " + currentUser.getUsername());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(900, 600);
    setLocationRelativeTo(null);        //--> center page on screen
    getContentPane().setBackground(Color.BLACK);
    JPanel formPanel = new JPanel(new GridLayout(6, 2, 5, 5));
    formPanel.add(new JLabel("name of the task:")); formPanel.add(tfTaskName);
    formPanel.add(new JLabel("Address:"));   formPanel.add(tfAddress);
    formPanel.add(new JLabel("Duration:"));  formPanel.add(tfDuration);
    formPanel.add(new JLabel("Date (YYYY-MM-DD):")); formPanel.add(tfDate);
    formPanel.add(new JLabel("Time (HH:MM:SS):"));  formPanel.add(tfTime);
    formPanel.add(new JLabel("Details:"));   formPanel.add(new JScrollPane(taDetails));

    colorButtons();
    tfTaskName.setBackground(Color.BLUE);
    tfAddress.setBackground(Color.BLUE);
    tfDuration.setBackground(Color.BLUE);
    tfDate.setBackground(Color.BLUE);
```

```
        tfTime.setBackground(Color.BLUE);
        taDetails.setBackground(Color.BLUE);
        tfTaskName.setForeground(Color.WHITE);
        tfAddress.setForeground(Color.WHITE);
        tfDuration.setForeground(Color.WHITE);
        tfDate.setForeground(Color.WHITE);
        tfTime.setForeground(Color.WHITE);
        taDetails.setForeground(Color.WHITE);
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(btnAdd);
        buttonPanel.add(btnEdit);
        buttonPanel.add(btnDelete);
        buttonPanel.add(btnPwd);
        buttonPanel.add(btnLogout);

        // table display added diaries
        String[] columns = {"Task Name", "Address", "Duration", "Date", "Time", "Details"};
        tableModel = new DefaultTableModel(columns, 0) {
            public boolean isCellEditable(int r, int c){          //isCellEditable(rows, columns)--> prevent editing directly
from table[]
                return false;                                 // u must use "edit button " to edit
            }
        };
        table = new JTable(tableModel);        // creating a new table from tableModel
        table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);       // selection mode is single to be sure
that user cannot select more than 1 row
                                                              // which help in edit and delete operations
        JScrollPane tableScroll = new JScrollPane(table);   //adding scrollbars to the table when needed allowing users
to navigate through a large number of added diaries

        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(formPanel, BorderLayout.CENTER);
        topPanel.add(buttonPanel, BorderLayout.NORTH);

        setLayout(new BorderLayout(10,10));
        add(topPanel, BorderLayout.SOUTH);
        add(tableScroll, BorderLayout.CENTER);

        // Button Actions
        btnAdd.addActionListener(e -> addEntry());
        btnEdit.addActionListener(e -> editEntry());
        btnDelete.addActionListener(e -> deleteEntry());
        btnLogout.addActionListener(e -> {
            dispose();
```

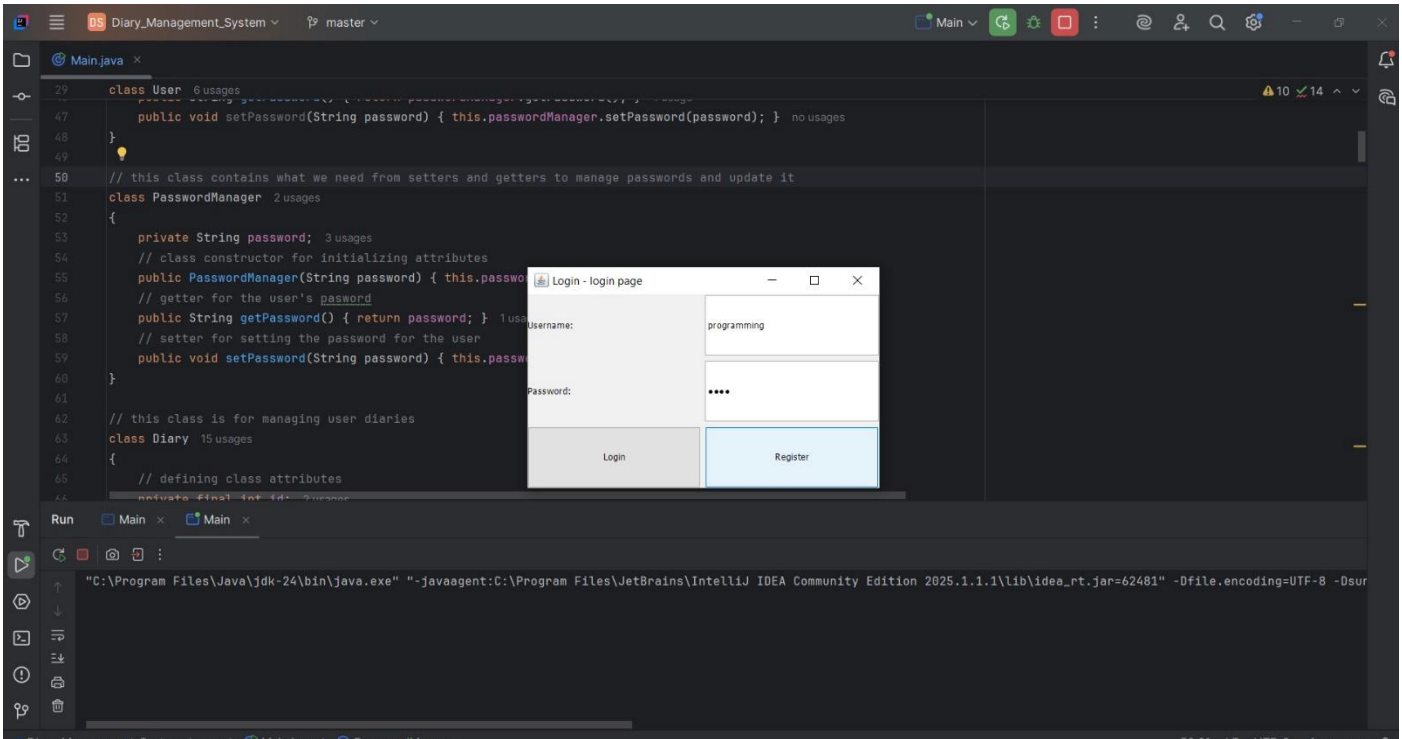
```
            currentUser = null;
            showLogin();
        });
        btnPwd.addActionListener(e -> {
            String newPwd = JOptionPane.showInputDialog(this, "Enter new password:");
            if (newPwd != null && !newPwd.trim().isEmpty()) {
                try {
                    loginManager.changePassword(currentUser.getUsername(), newPwd);
                    JOptionPane.showMessageDialog(this, "Password changed.");
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(this, "Error: "+ex.getMessage());
                }
            }
        });

        // Table select: load into fields
        table.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                int row = table.getSelectedRow();
                if (row >= 0 && loadedDiaries != null && row < loadedDiaries.size()) {
                    Diary d = loadedDiaries.get(row);
                    tfTaskName.setText(d.getName());
                    tfAddress.setText(d.getAddress());
                    tfDuration.setText(d.getDuration());
                    tfDate.setText(d.getDate());
                    tfTime.setText(d.getTime());
                    taDetails.setText(d.getDetails());
                }
            }
        });

        loadEntries();
        setVisible(true);
    }

    private void loadEntries() { //   loadEntries()--> retrive added diaries from the database and displays them in table
in GUI.
        try {
            loadedDiaries = recordManager.viewRecord(currentUser.getId());
            tableModel.setRowCount(0);
            for (Diary d : loadedDiaries) {
                tableModel.addRow(new Object[]{
                        d.getName(), d.getAddress(), d.getDuration(),
                        d.getDate(), d.getTime(), d.getDetails()
```

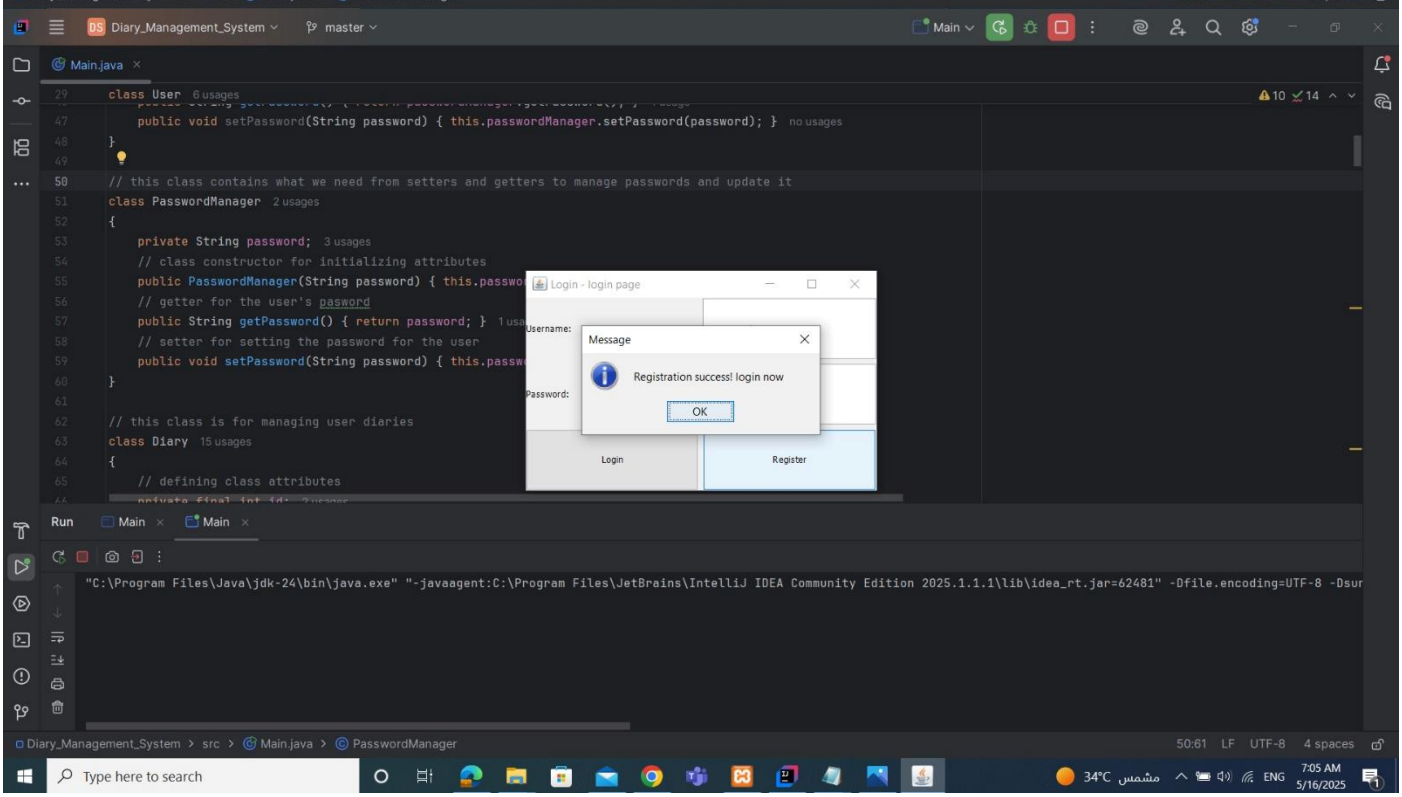
```
            });
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Failed to load entries:\n" + ex.getMessage());
    }
}

private void clearForm() {      // clearForm()--> it clear text fields ,help me in operations :- adding new record
    tfTaskName.setText("");   //                                                - editing existing record
    tfAddress.setText("");    //                                               - deleting existing record
    tfDuration.setText("");   //
    tfDate.setText("");
    tfTime.setText("");
    taDetails.setText("");
}

private boolean validateEntryFields() {                // validateEntryFields()-> verify that required fields are filled
in.
    if (tfTaskName.getText().trim().isEmpty() || tfDate.getText().trim().isEmpty() ||
tfTime.getText().trim().isEmpty()) { //checks if any of the required fields (task name, date, or time) are empty
        JOptionPane.showMessageDialog(this, "Task Name, Date, and Time are required.");                    //if
empty dsiplay mess.
        return false;
    }
    return true;
}

private void addEntry() {
    if(!validateEntryFields()) return;
    try {
        Diary d = new Diary(0, // id is 0 because it is defined by database AUTO INCREMENT
            tfTaskName.getText(), tfDuration.getText(), tfAddress.getText(),
            tfDate.getText(), tfTime.getText(), taDetails.getText(), currentUser.getId());
        recordManager.addRecord(d);
        JOptionPane.showMessageDialog(this, "Entry added!");
        clearForm();
        loadEntries();
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Failed to add entry:\n" + ex.getMessage());
    }
}

private void editEntry() {
```

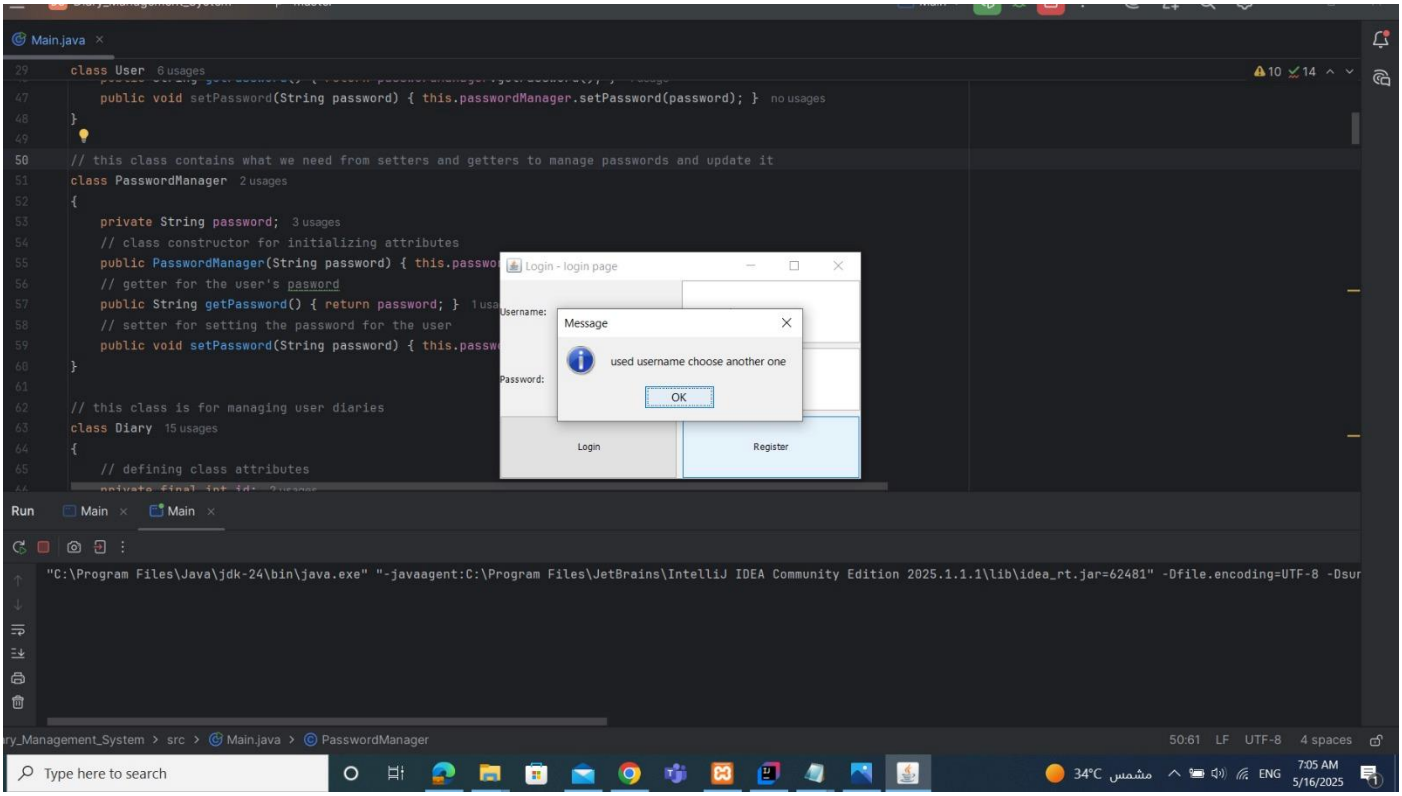
```
        int row = table.getSelectedRow();       //getSelectedRow()-->>Gets the index of the currently selected row in
the table.
        if (row < 0 || loadedDiaries == null || row >= loadedDiaries.size()) {      // If row is negative--> there is no
selection.
            JOptionPane.showMessageDialog(this, "Select a row to edit.");   //If loadedDiaries is null or row is out of
bounds.there is problem in  data
            return;
        }
        if(!validateEntryFields()) return;
        try {
            Diary d = loadedDiaries.get(row);
            d.setName(tfTaskName.getText());
            d.setAddress(tfAddress.getText());
            d.setDuration(tfDuration.getText());
            d.setDate(tfDate.getText());
            d.setTime(tfTime.getText());
            d.setDetails(taDetails.getText());
            recordManager.updateRecord(d);
            JOptionPane.showMessageDialog(this, "Entry updated!");
            clearForm();
            loadEntries();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Failed to edit entry:\n" + ex.getMessage());
        }
    }

    private void deleteEntry() {
        int row = table.getSelectedRow();
        if (row < 0 || loadedDiaries == null || row >= loadedDiaries.size()) {
            JOptionPane.showMessageDialog(this, "Select a row to delete.");
            return;
        }
        try {
            Diary d = loadedDiaries.get(row);
            recordManager.deleteRecord(d.getId());
            JOptionPane.showMessageDialog(this, "Entry deleted!");
            clearForm();
            loadEntries();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Failed to delete entry:\n" + ex.getMessage());
        }
    }
}
```

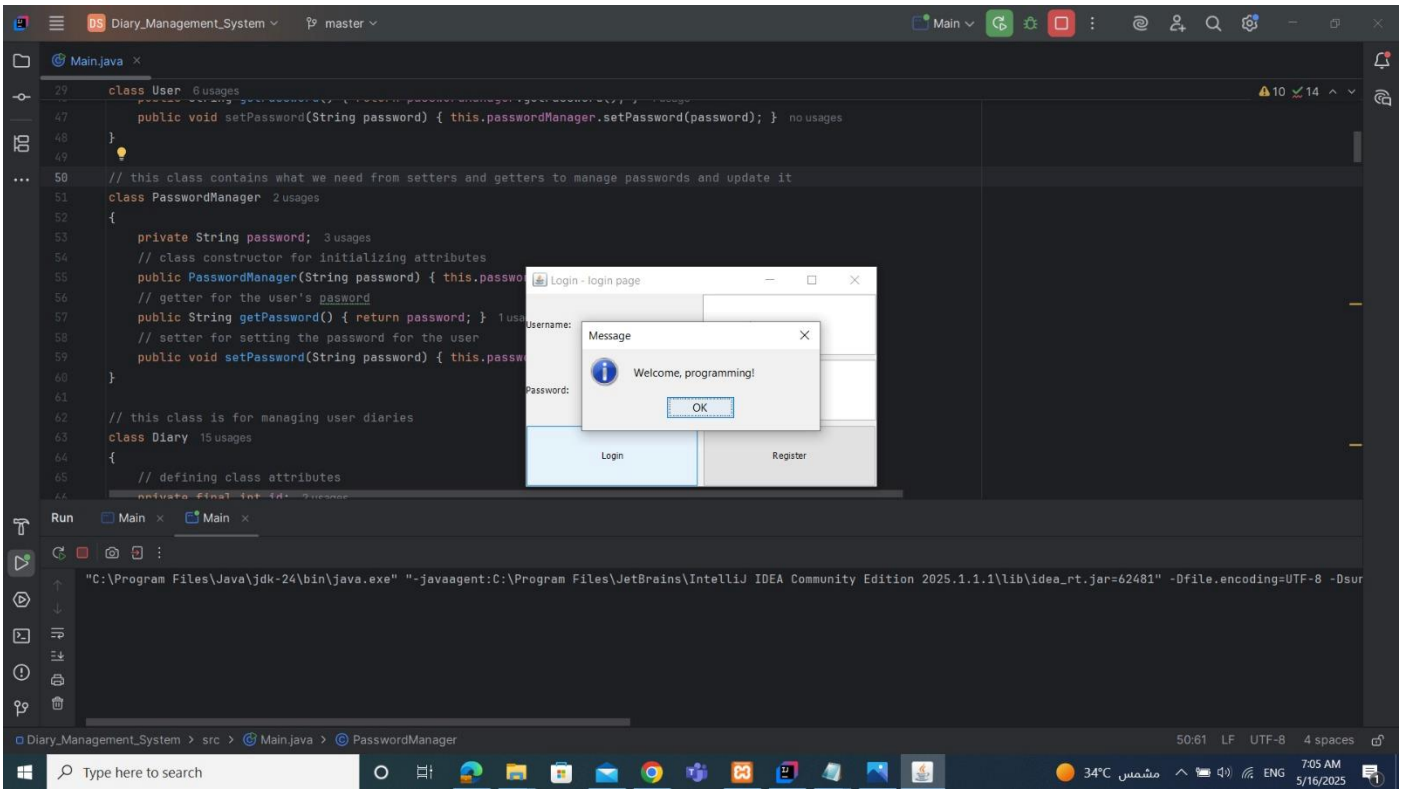
```
}


public class Main
{
   public static void main(String[] args) {
      // Set Look and Feel (optional)
      try { UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()); }
      catch(Exception ignored){}
      SwingUtilities.invokeLater(() -> new DiaryGUI().showLogin());
   }
}
```

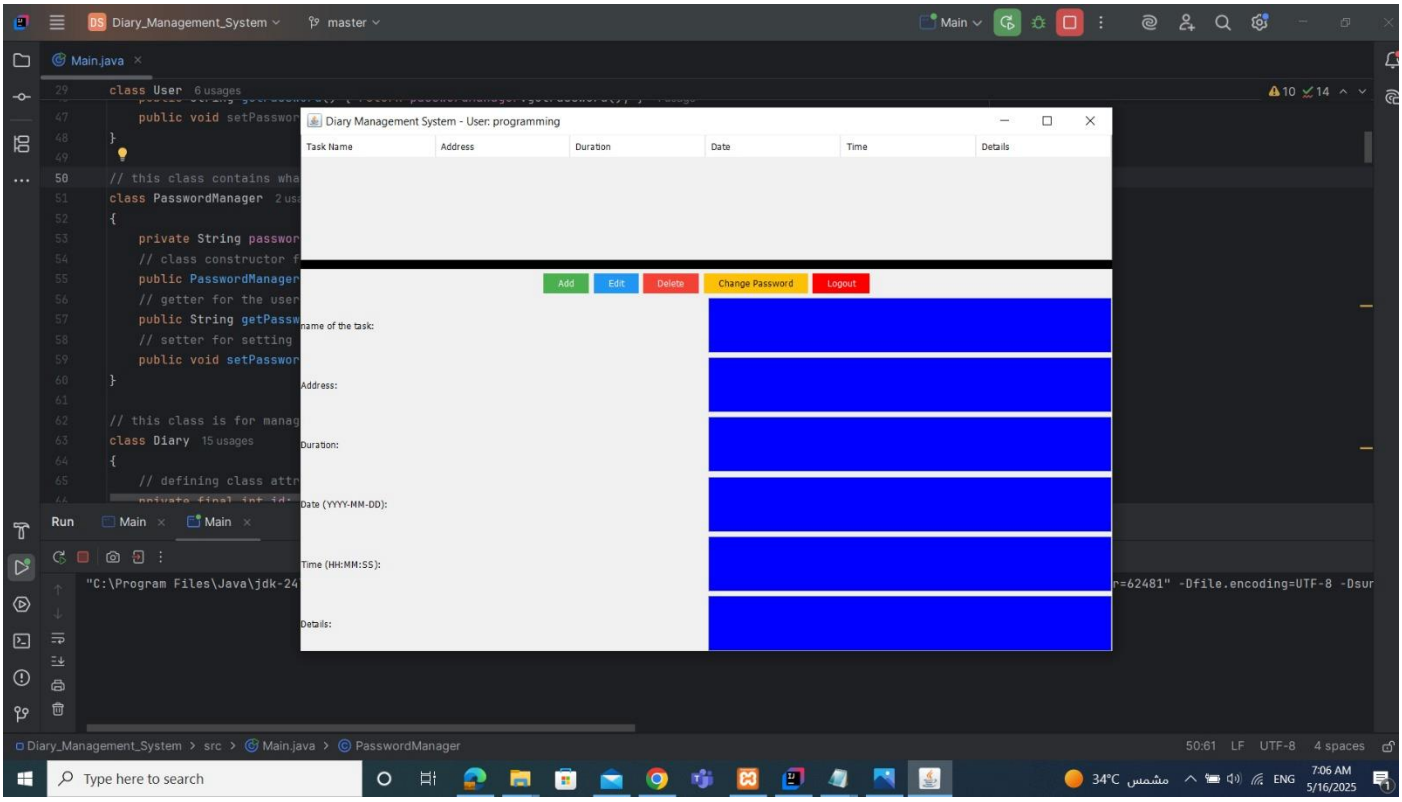
Output and results

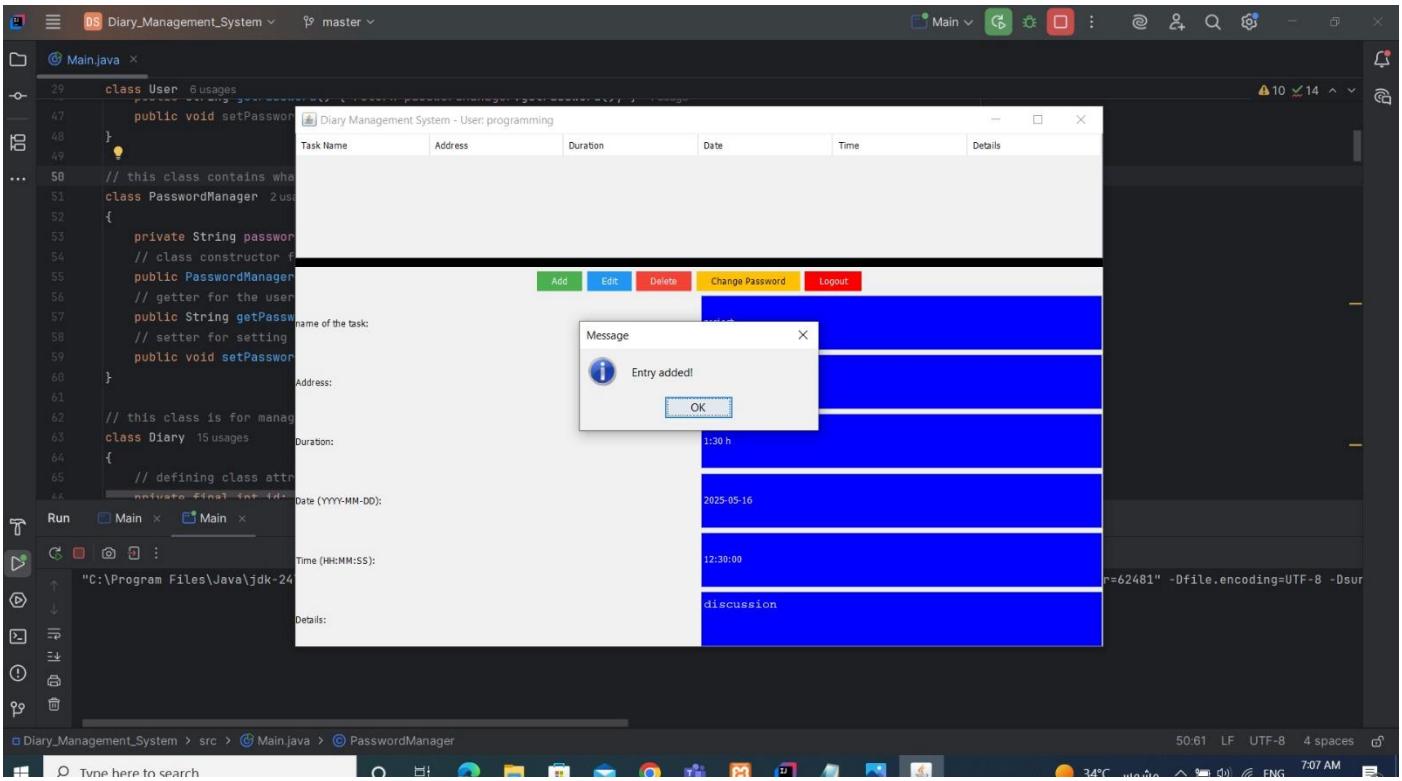

Diary_Management_System
master
Main.java
class User 6 usages
public void setPassword(String password) { this.passwordManager.setPassword(password); } no usages
// this class contains what we need from setters and getters to manage passwords and update it
class PasswordManager 2 usages
private String password; 3 usages
// class constructor for initializing attributes
public PasswordManager(String password) { this.passwo
// getter for the user's pasword
public String getPassword() { return password; }
// setter for setting the password for the user
public void setPassword(String password) { this.passw
// this class is for managing user diaries
class Diary 15 usages
// defining class attributes
Login - login page
Username:
programming
Password:
Login
Register
Run
Main
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1.1\lib\idea_rt.jar=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces

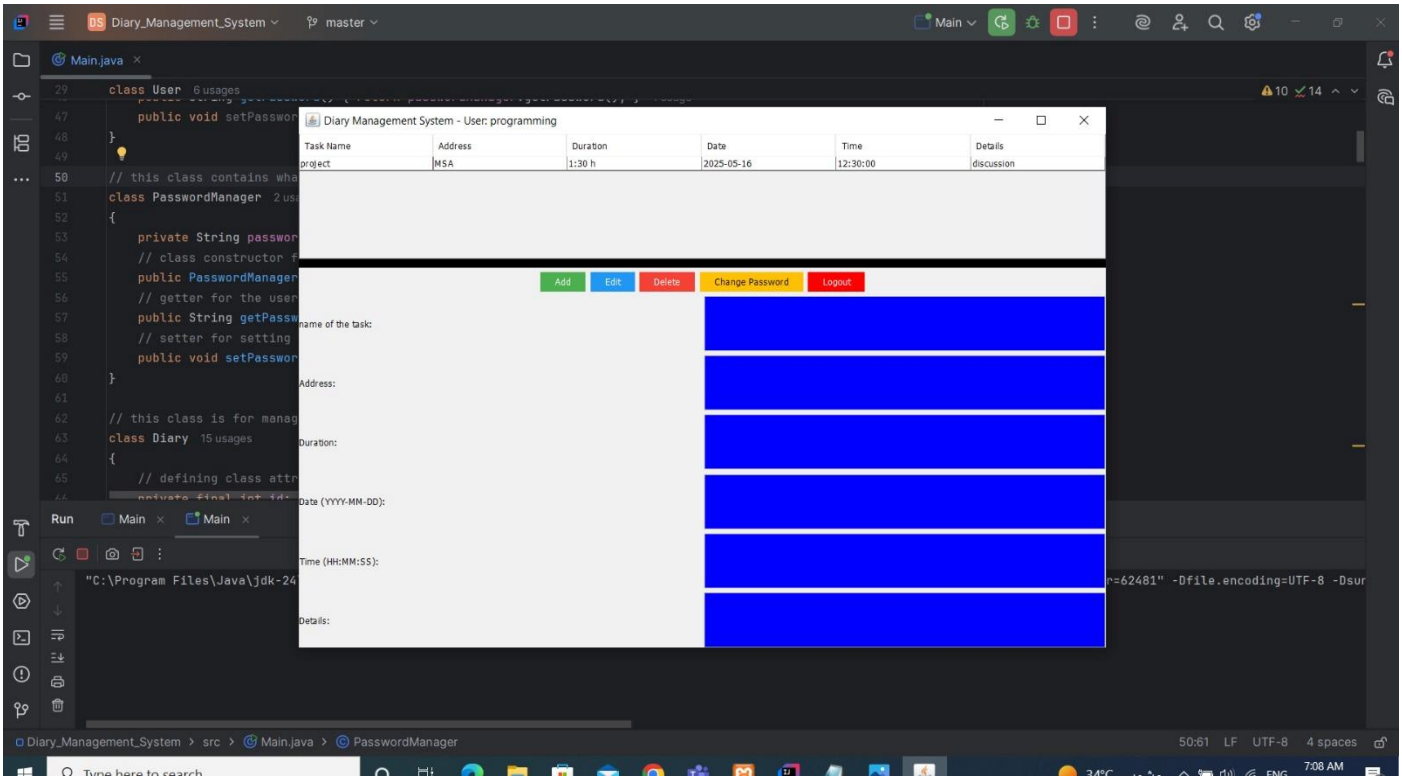

Diary_Management_System
master
Main.java
class User 6 usages
public void setPassword(String password) { this.passwordManager.setPassword(password); } no usages
// this class contains what we need from setters and getters to manage passwords and update it
class PasswordManager 2 usages
private String password; 3 usages
// class constructor for initializing attributes
public PasswordManager(String password) { this.passwo
// getter for the user's pasword
public String getPassword() { return password; }
// setter for setting the password for the user
public void setPassword(String password) { this.passw
// this class is for managing user diaries
class Diary 15 usages
// defining class attributes
Login - login page
Username:
Password:
Message
Registration success! login now
OK
Login
Register
Run
Main
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1.1\lib\idea_rt.jar=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C مشمس
ENG
7:05 AM
5/16/2025

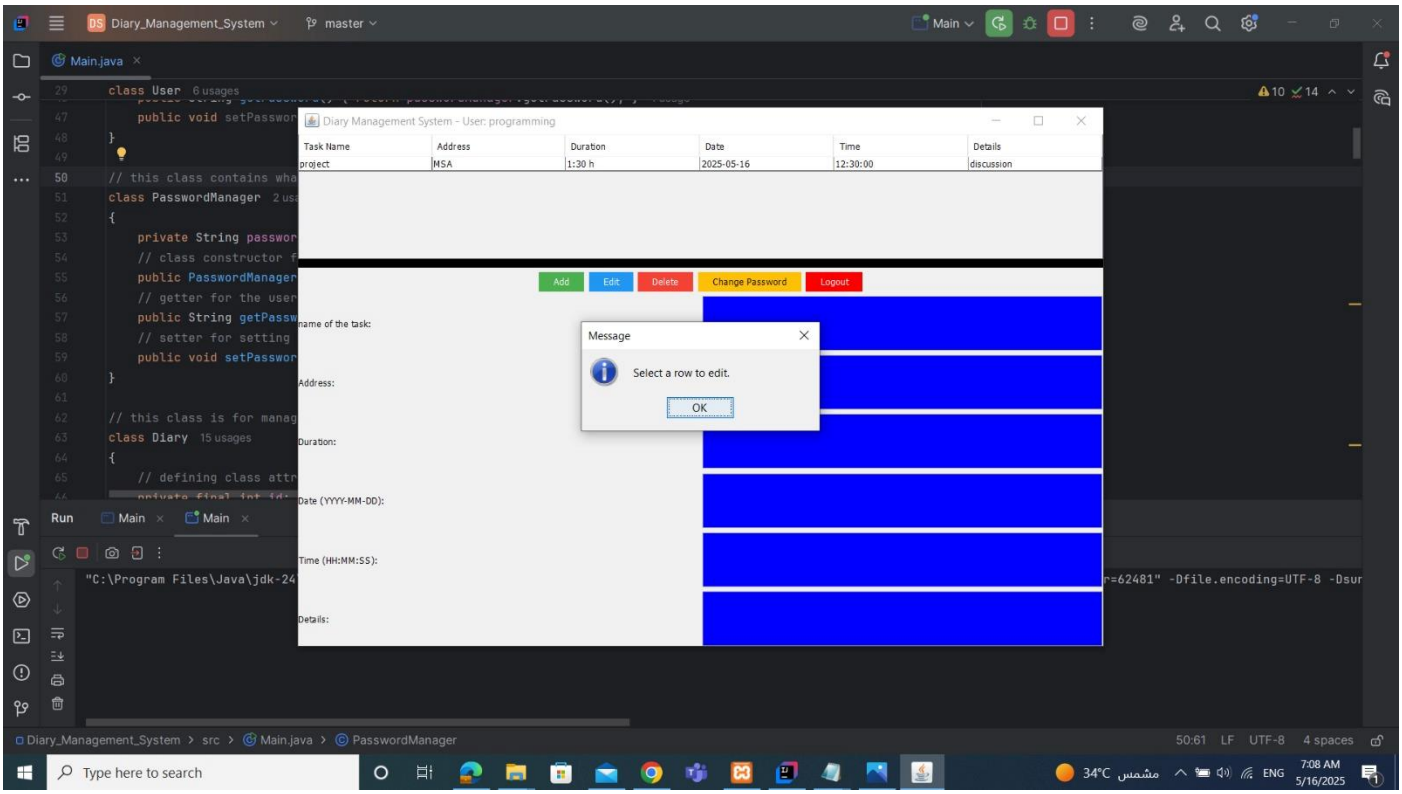

Main.java
class User 6 usages
public void setPassword(String password) { this.passwordManager.setPassword(password); } no usages
// this class contains what we need from setters and getters to manage passwords and update it
class PasswordManager 2 usages
private String password; 3 usages
// class constructor for initializing attributes
public PasswordManager(String password) { this.passwo
// getter for the user's pasword
public String getPassword() { return password; }
// setter for setting the password for the user
public void setPassword(String password) { this.passw
// this class is for managing user diaries
class Diary 15 usages
// defining class attributes
Login - login page
Username:
Password:
Login
Register
Message
used username choose another one
OK
Run
Main
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1.1\lib\idea_rt.jar=62481" -Dfile.encoding=UTF-8 -Dsur
PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C
ENG
7:05 AM
5/16/2025

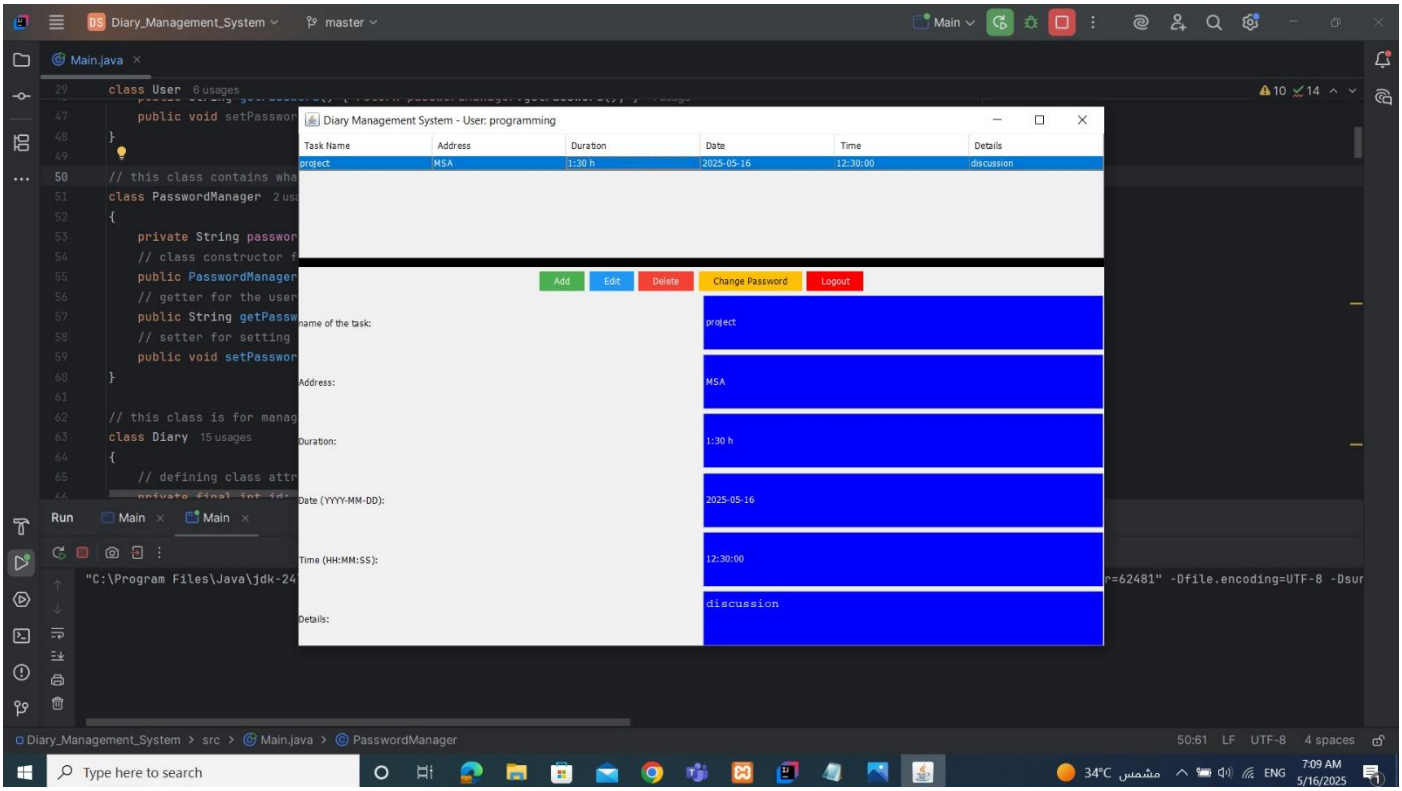

Diary_Management_System
master
Main.java
class User 6 usages
public void setPassword(String password) { this.passwordManager.setPassword(password); } no usages
// this class contains what we need from setters and getters to manage passwords and update it
class PasswordManager 2 usages
private String password; 3 usages
// class constructor for initializing attributes
public PasswordManager(String password) { this.passwo
// getter for the user's pasword
public String getPassword() { return password; }
// setter for setting the password for the user
public void setPassword(String password) { this.passw
// this class is for managing user diaries
class Diary 15 usages
// defining class attributes
Login - login page
Username:
Password:
Login
Register
Message
Welcome, programming!
OK
Run
Main
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1.1\lib\idea_rt.jar=62481" -Dfile.encoding=UTF-8 -Dsur
PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C
ENG
7:05 AM
5/16/2025

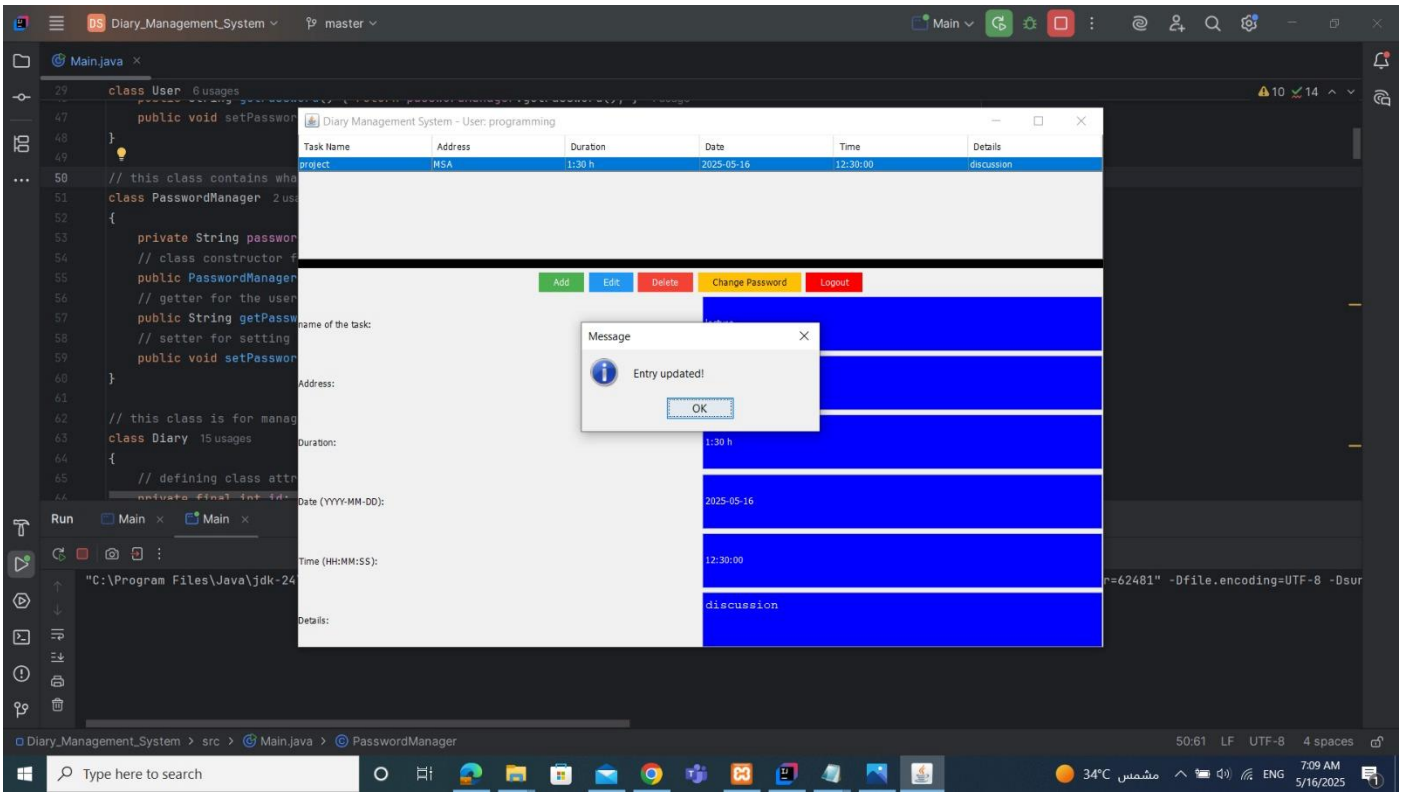

Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 us
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Run
Main
"C:\Program Files\Java\jdk-24
r=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C مشمس
ENG
7:06 AM
5/16/2025

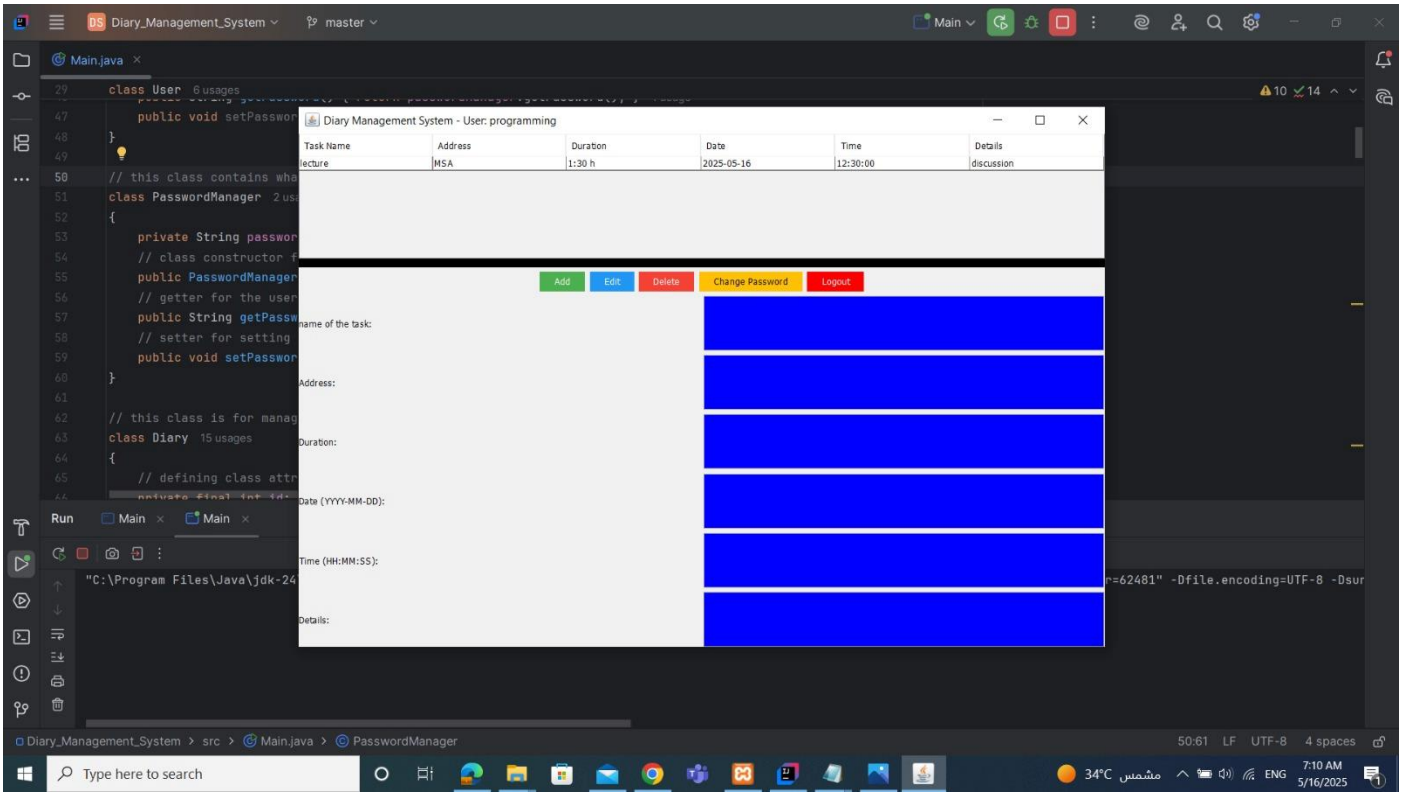

Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 us
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
1:30 h
Date (YYYY-MM-DD):
2025-05-16
Time (HH:MM:SS):
12:30:00
Details:
discussion
Message
Entry added!
OK
Run
Main
"C:\Program Files\Java\jdk-24
r=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
7:07 AM

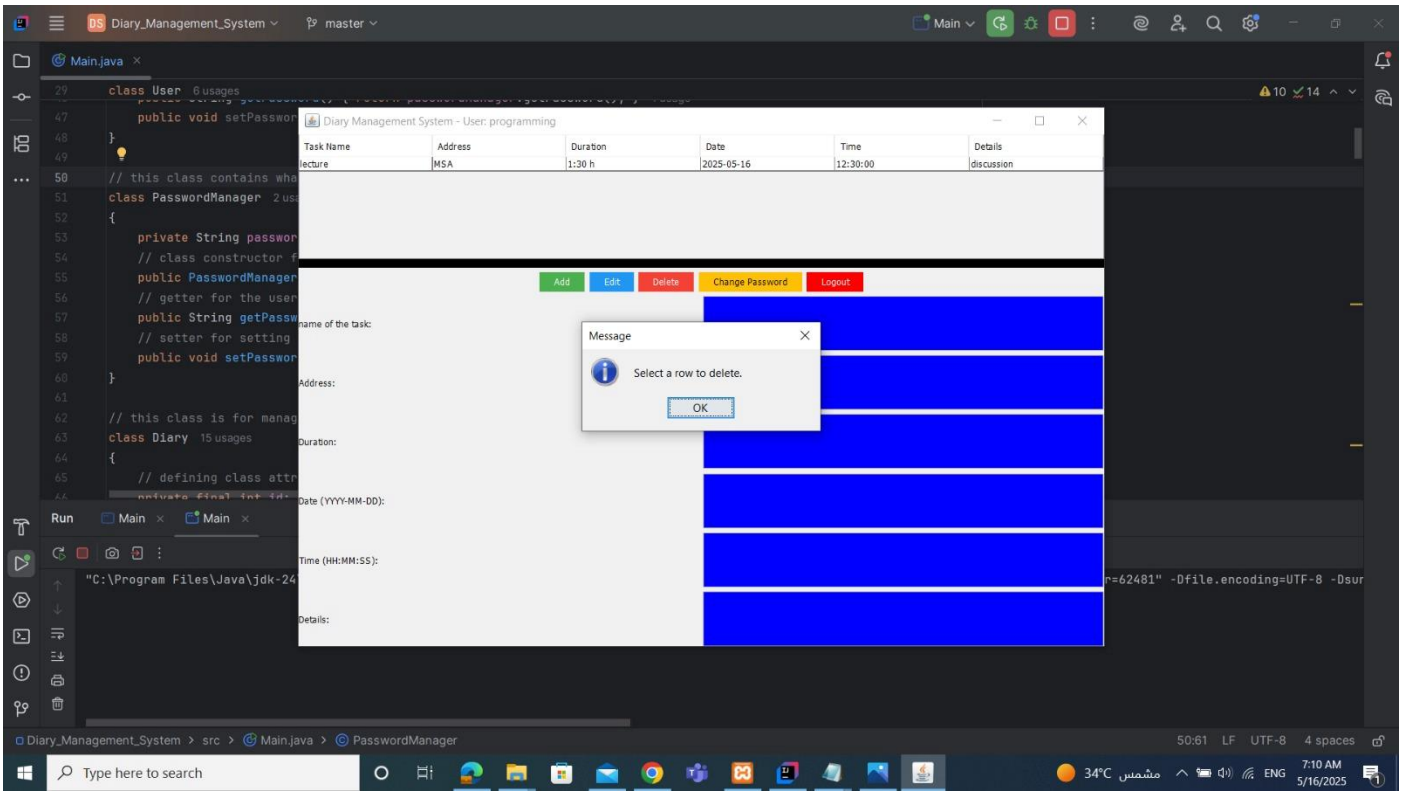

Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 usa
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
project
MSA
1:30 h
2025-05-16
12:30:00
discussion
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Run
Main
"C:\Program Files\Java\jdk-24
=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
7:08 AM

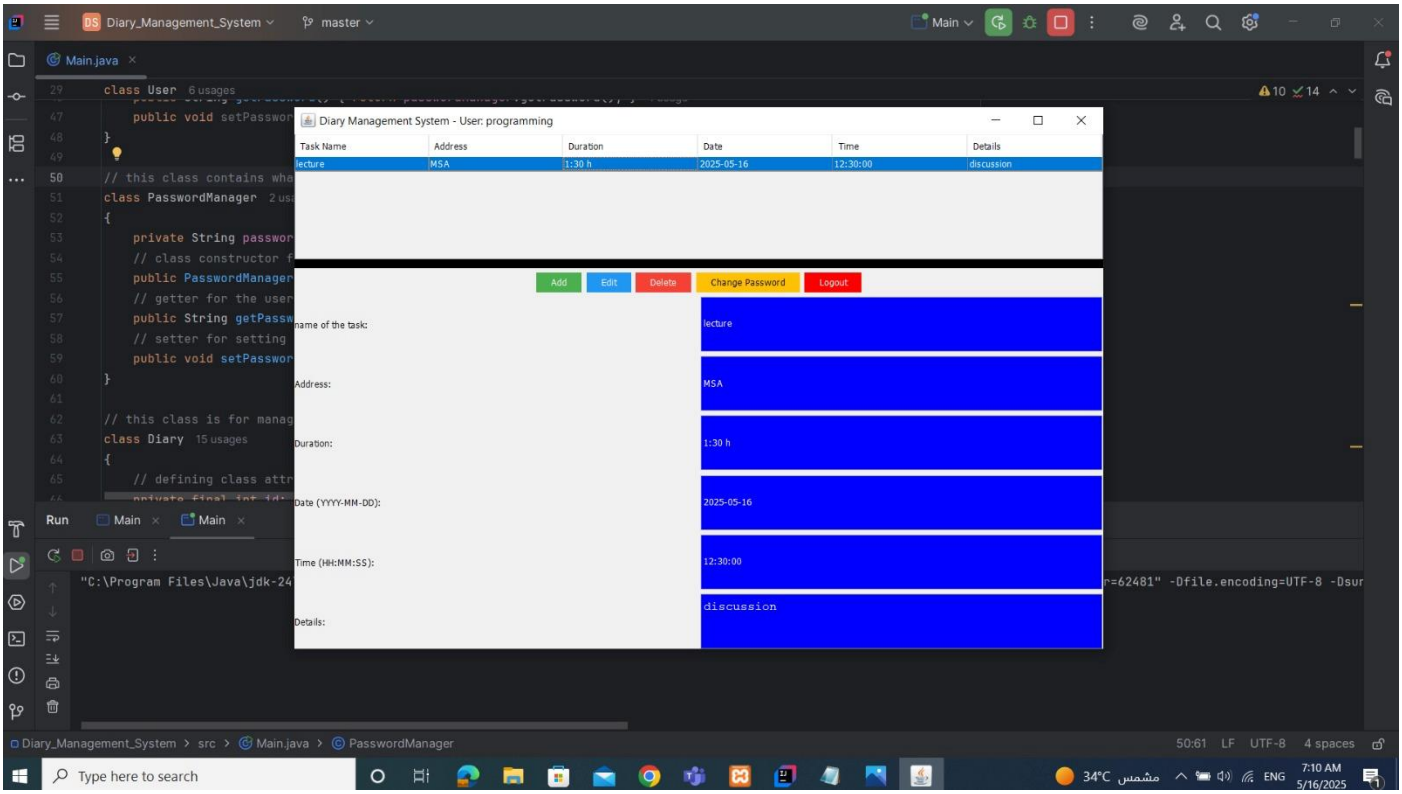

Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 usa
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
project
MSA
1:30 h
2025-05-16
12:30:00
discussion
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Message
Select a row to edit.
OK
Run
Main
"C:\Program Files\Java\jdk-24
=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C مشمس
ENG
7:08 AM
5/16/2025

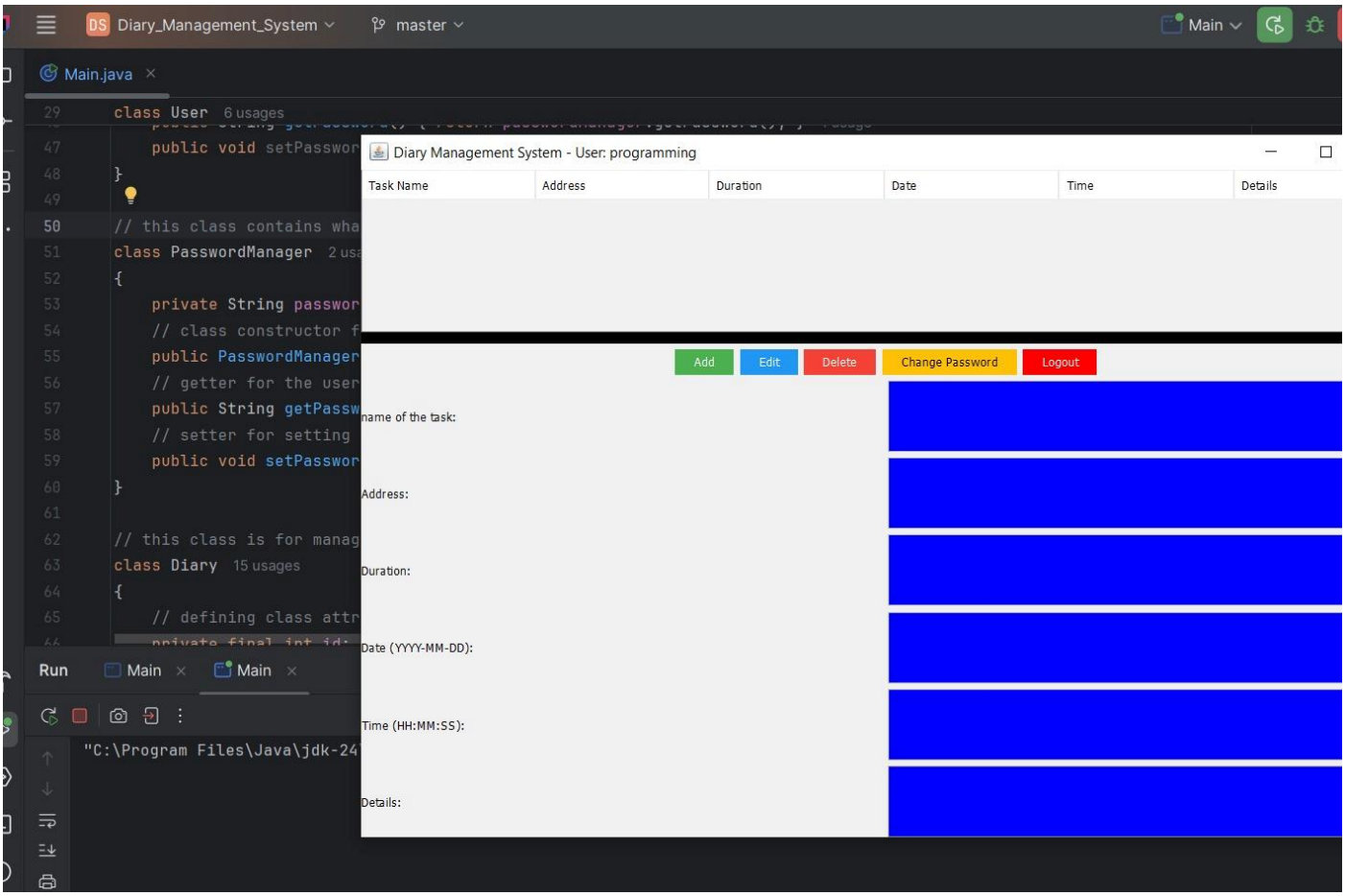

Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 us
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Run
Main
"C:\Program Files\Java\jdk-24
Diary_Management_System - User: programming
Task Name
Address
Duration
Date
Time
Details
project
MSA
1:30 h
2025-05-16
12:30:00
discussion
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C
ENG
7:09 AM
5/16/2025


Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 us
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Run
Main
"C:\Program Files\Java\jdk-24
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
project
MSA
1:30 h
2025-05-16
12:30:00
discussion
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Message
Entry updated!
OK
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C
ENG
7:09 AM
5/16/2025


Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 usa
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
lecture
MSA
1:30 h
2025-05-16
12:30:00
discussion
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Run
Main
"C:\Program Files\Java\jdk-24
=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C مشمس
ENG
7:10 AM
5/16/2025


Diary_Management_System
master
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 usa
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
lecture
MSA
1:30 h
2025-05-16
12:30:00
discussion
Add
Edit
Delete
Change Password
Logout
name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Message
Select a row to delete.
OK
Run
Main
"C:\Program Files\Java\jdk-24
=62481" -Dfile.encoding=UTF-8 -Dsur
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C مشمس
ENG
7:10 AM
5/16/2025


Diary_Management_System
master
Main
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 usa
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Run
Main
Main
"C:\Program Files\Java\jdk-24
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
lecture
MSA
1:30 h
2025-05-16
12:30:00
discussion
Add
Edit
Delete
Change Password
Logout
Name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:
Diary_Management_System > src > Main.java > PasswordManager
50:61 LF UTF-8 4 spaces
Type here to search
34°C مشمس
ENG
7:10 AM
5/16/2025


Diary_Management_System
master
Main
Main.java
class User 6 usages
public void setPasswor
}
// this class contains wha
class PasswordManager 2 usa
{
private String passwor
// class constructor f
public PasswordManager
// getter for the user
public String getPassw
// setter for setting
public void setPasswor
}
// this class is for manag
class Diary 15 usages
{
// defining class attr
Run
Main
Main
"C:\Program Files\Java\jdk-24
Diary Management System - User: programming
Task Name
Address
Duration
Date
Time
Details
Add
Edit
Delete
Change Password
Logout
Name of the task:
Address:
Duration:
Date (YYYY-MM-DD):
Time (HH:MM:SS):
Details:

localhost/phpmyadmin/index.php?route=/sql&db=diarysystem&table=diary&pos=0

Server: 127.0.0.1 » Database: diarysystem » Table: diary

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Showing rows 0 - 3 (4 total, Query took 0.0010 seconds.)

```
SELECT * FROM `diary`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	id	name	duration	address	date	time	details	user_id
Edit Copy Delete	2	jcsjfjed	1m	jbscjfbsjnfsjk	2005-06-06	01:01:01	jdnfkjwkwnf	2
Edit Copy Delete	3	jbcdhwe	1h	bcfhebfueb	2005-06-06	01:02:01	ncksnkfcn	1
Edit Copy Delete	7	kjdkfkwnfk	1h	MSA	2006-06-06	12:04:03	,cncvkd	7
Edit Copy Delete	8	cghchchmj	1h	hgdhtghgh	2006-12-23	03:05:00	,fnsknfks	7

Check all | With selected: Edit | Copy | Delete | Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

Print | Copy to clipboard | Export | Display chart | Create view

Bookmark this SQL query

Label: | Let every user access this bookmark

Console

Type here to search | 34°C مشمس | ENG | 7:26 AM 5/16/2025

localhost/phpmyadmin/index.php?route=/database/sql&db=diarysystem

Server: 127.0.0.1 » Database: diarysystem

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Tracking | Designer | Central columns

Run SQL query/queries on database diarysystem:

```
USE DiarySystem;

CREATE TABLE user (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE diary (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    duration VARCHAR(50),
    address VARCHAR(255),
    date DATE,
    time TIME,
    details TEXT,
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES user(id) ON DELETE CASCADE
);
```

Clear | Format | Get auto-saved query

Bind parameters

Bookmark this SQL query:

GitHub(optional)

Include link of github repo containing your project code and report and add screenshot of repo with commit logs

References

1. https://www.youtube.com/playlist?list=PLCRogJ_v4BQUuRtS3t_s3TpGFHk8Rsraz&si=-KjVpCGQbufVEcNu
2. https://www.youtube.com/watch?v=Kmgo00avvEw