

**Object Oriented
Programming (L2BC)**



**Computer Science Undergraduate Program
Universitas Bina Nusantara
Jakarta**

**Final Project
Stars**

Introduction

Background

The students are expected and challenged to develop and design a complete application that goes beyond the subject matter of the course in Object-Oriented Programming and applies everything learned regarding the Java Object-Oriented Programming language and problem-solving techniques. After much thought and research, I have decided to write a project management system called Stars. This system will develop some programming libraries not covered in the semester of Object-Oriented Programming. I will create a Graphical User Interface using Java Swing and Java AWT for the Stars system.

The system under consideration here is the Stars system, which helps users handle projects and day-to-day tasks efficiently. Project tracking, deadline management, daily tasks management, and user management are the system's main features. Through this application, I am, at this moment, trying to bring forth evidence of my knowledge and competence in advanced object-oriented design principles, Java programming, and practical problem-solving skills.

Project Overview

The “Stars” is a project management system that assists to track the current projects and its progress. The main objective of this application is to provide a user-friendly platform to manage projects, enabling users to create, update, and monitor the project details. Several features are offered, such as project registration, deadline tracking, progress monitoring, user management, and task management. By using technologies such as Java Swing and Java AWT to develop a Graphic User Interface (GUI), Stars offers a visually appealing interface for users to interact with. Through Stars, users can expect improved project organization, enhanced collaboration among team members, and better decision-making based on real-time project data.

Libraries Used

1. Java Swing and Java AWT

These libraries are utilized for creating the graphical user interface (GUI) of the application. They provide lots of components and tools to build interactive and visually appealing interfaces.

2. JSON.simple

This library is employed for reading and writing data in JSON format. It simplifies the handling of project data by providing easy-to-use methods for parsing JSON objects and arrays, enabling efficient storage, retrieval, and manipulation of project details.

3. FlatLaf

This library is utilized for applying modern and consistent look and feel themes to the Stars application. FlatLaf offers a variety of themes and styles, allowing customization of the GUI appearance to suit different preferences and aesthetics.

4. Swing-DateTime-Picker

Incorporating Swing-DateTime-Picker enhances the calendar design aspect of the application. This library provides intuitive date and time picker components, facilitating user-friendly selection and input of dates for project deadlines. It enhances usability and improves the overall user experience by streamlining date-related interactions within the application.

Features

Some features are implemented in the application to support the objectives of the application. It involves:

1. Project Creation and Management

Users can create new projects, specifying details such as project name, leader, and deadlines. They can also edit and update project information as needed.

2. Deadline Tracking

Stars provides users with the ability to set deadlines for projects and tracks the time remaining until each deadline. This feature helps users stay on schedule and prioritize tasks effectively.

3. Progress Monitoring

Users can monitor the progress of each project, tracking completion percentages achieved. This feature enables users to assess project status at a glance and identify areas that require attention.

4. Team Collaboration

Stars facilitates collaboration among team members by allowing them to share project details, assign tasks, and communicate effectively within the platform. This fosters teamwork and enhances productivity.

5. Daily Task Management

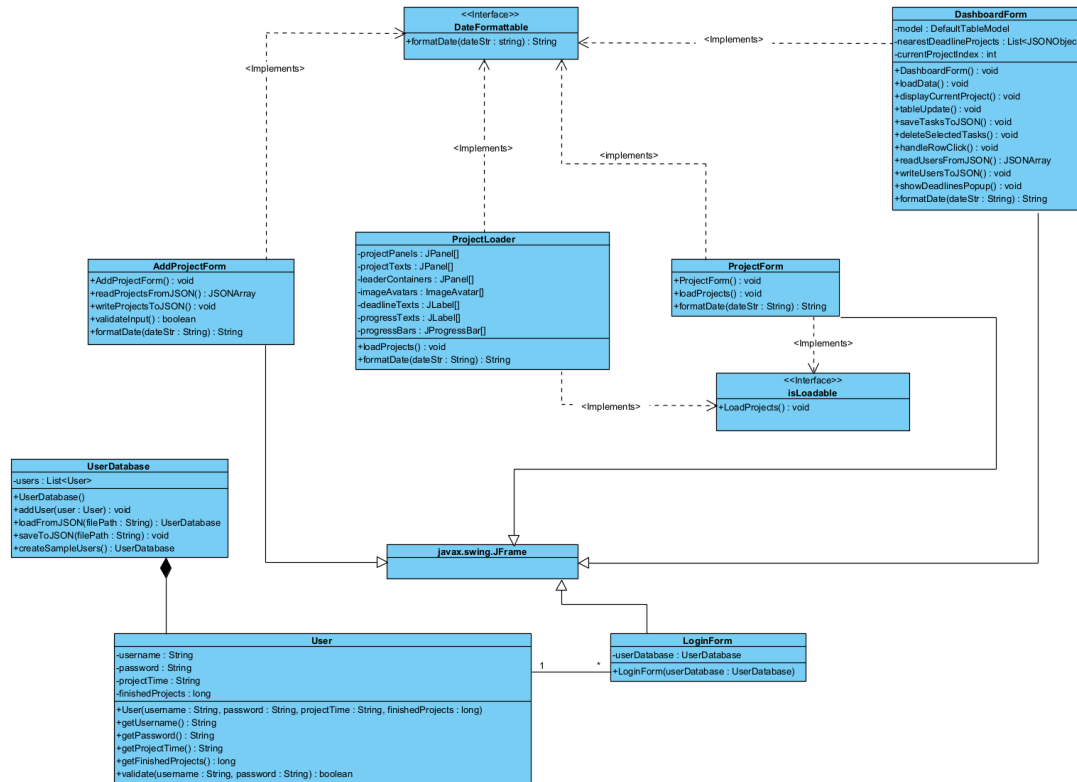
The application includes a daily task management feature that enables users to create and track tasks for each day. Users can also delete the tasks, if they are done or not longer needed.

6. Calendar Integration

Stars integrates a Swing-DateTime-Picker library to provide a user-friendly calendar interface for selecting dates and deadlines. This simplifies date-related interactions and improves usability.

Solution Design

UML Diagram



From the UML diagram shown above, it is shown that there are 2 interfaces used in this application. The first interface, **DateFormattable**, is implemented by 4 classes, **AddProjectForm.java**, **ProjectLoader.java**, **ProjectForm.java**, **DashboardForm.java**. While the other interface used, **isLoadable**, is implemented by 2 classes, **ProjectLoader.java** and **ProjectForm.java**. As I created the GUI for the application, most of the classes used are inheriting the **javax.swing.JFrame**. Furthermore, I have also created the **User** and **UserDatabase** class, and there is a composition relation from **UserDatabase** to **User**, as the **UserDatabase** contains instances of **User**. And lastly, there is a relation of a "1 to many" relationship from **User** to **LoginForm**, as the **LoginForm** may allow multiple logins simultaneously.

Algorithms

1. Formatting Date

```
public String formatDate(String dateStr) {  
    try {  
        SimpleDateFormat inputFormat = new SimpleDateFormat("yyyy-MM-dd");  
        Date date = inputFormat.parse(dateStr);  
        SimpleDateFormat outputFormat = new SimpleDateFormat("dd MMMM yyyy");  
        return outputFormat.format(date);  
    } catch (Exception e) {  
        e.printStackTrace();  
        return "Invalid Date";  
    }  
}
```

This algorithm is used to change the date format so it can convert the date from the database, which is stored in the .json file to be displayed according to what we want.

2. Read and Writing JSON file

```
private JSONArray readUsersFromJSON() throws ParseException {  
    try (FileReader reader = new FileReader("src/stars/data/dailytasks.json")) {  
        JSONParser jsonParser = new JSONParser();  
        JSONObject jsonObject = (JSONObject) jsonParser.parse(reader);  
  
        JSONArray userDataArray = (JSONArray) jsonObject.get("users");  
        if (!userDataArray.isEmpty()) {  
            JSONObject userData = (JSONObject) userDataArray.get(0);  
            return (JSONArray) userData.get("dailyTasks");  
        }  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
    return new JSONArray();  
}  
  
private void writeUsersToJSON(JSONObject userData) {  
    try (FileWriter file = new FileWriter("src/stars/data/dailytasks.json")) {  
        file.write(userData.toJSONString());  
        file.flush();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

These functions are used to either read or write the files from the .json file, in this case, it reads and writes to the dailytasks.json, which shows that this is used to either view, add, or delete the daily tasks shown in the dashboard.

3. Validation

```
private boolean validateInput() {
    // Check if any form field is empty
    if (projectnametext.getText().isEmpty() || leadernametext.getText().isEmpty() ||
        projectdetailtext.getText().isEmpty() || deadlinetext.getText().isEmpty() ||
        progresstext.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please fill in all form fields.");
        return false;
    }

    try {
        Integer.parseInt(progresstext.getText());
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Progress must be an integer.");
        return false;
    }

    // Check if the deadline is before today
    DateTimeFormatter inputFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    LocalDate deadlineDate = null;
    try {
        deadlineDate = LocalDate.parse(deadlinetext.getText(), inputFormatter);
    } catch (DateTimeParseException e) {
        JOptionPane.showMessageDialog(this, "Invalid deadline date format. Please use dd/MM/yyyy.");
        return false;
    }

    LocalDate currentDate = LocalDate.now();
    if (deadlineDate.isBefore(currentDate)) {
        JOptionPane.showMessageDialog(this, "Deadline cannot be set to a date before today.");
        return false;
    }

    return true;
}
```

Validations are also used to check for the input, it checks if the textbox is empty, or if the value inputted is not in the correct format. This is useful to ensure that the data inputted to the database is correct and will not cause any errors.

4. Sorting With Collections

```
// Sort projects by lastEdited for displaying in Panel 1 and Panel 2
Collections.sort(projects, new Comparator<JSONObject>() {
    @Override
    public int compare(JSONObject o1, JSONObject o2) {
        try {
            Date d1 = sdf.parse((String) o1.get("lastEdited"));
            Date d2 = sdf.parse((String) o2.get("lastEdited"));
            return d2.compareTo(d1);
        } catch (Exception e) {
            return 0;
        }
    }
});
```

This algorithm focuses on sorting the projects by checking the last edit, and will ensure that the first project will be the latest updated projects.

Solution Scheme

1. Object-Oriented Design

I have utilized the object-oriented principles in this application. This approach enhances code reusability, encapsulation, and maintainability. Each component of the application, from project entities to user interfaces, is represented as a class, allowing for easy extension and modification.

2. User Interface

The app incorporates a user-friendly interface designed with Java Swing and Java AWT libraries. This interface provides intuitive navigation and interaction for users, facilitating efficient project management. Additionally, the integration of FlatLaf for themes enhances the visual appeal and customization options for users.

3. Data Management

Effective data management is achieved through organized data structures and algorithms. The app employs data structures such as JSON files to store project information and user data. Algorithms are implemented to parse and manipulate data efficiently, ensuring smooth operation and performance.

4. Event Handling

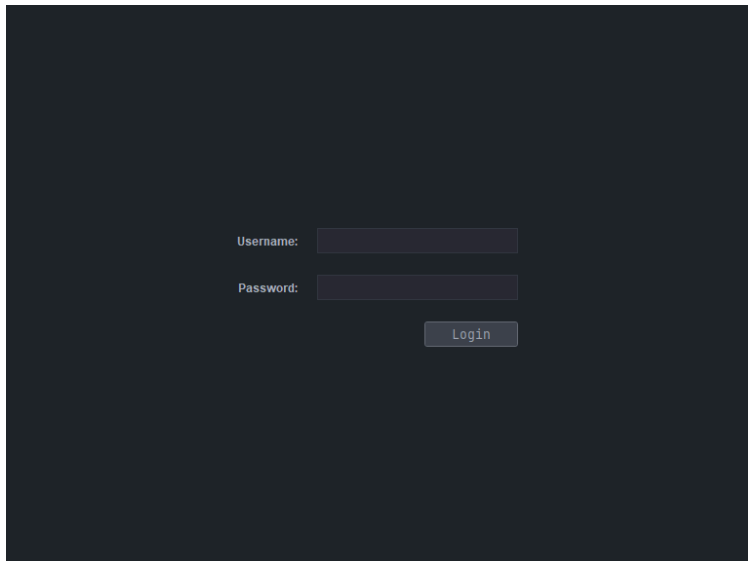
To prevent human errors and the errors in the application, this app implements the event handling mechanisms. This approach ensures a responsive and interactive user experience.

5. Integration of Third-Party Libraries

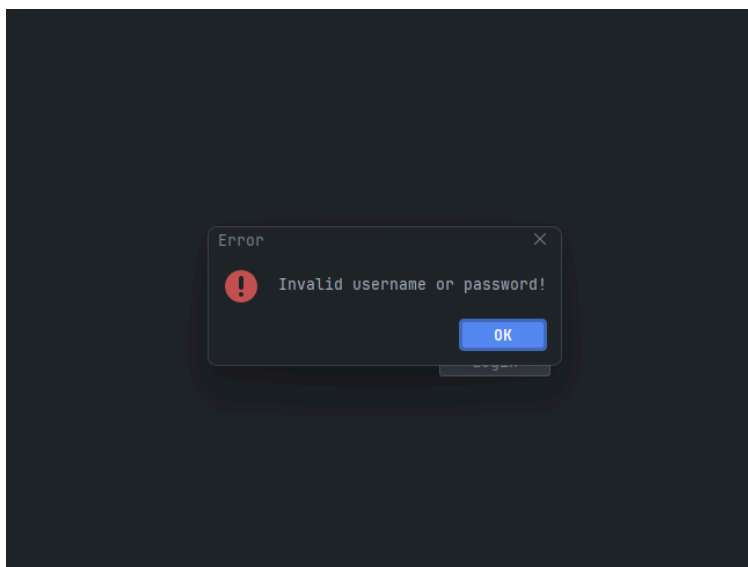
The app leverages third-party libraries such as Swing-Datetime-Picker for calendar design. These libraries extend the functionality of the application, providing additional features and capabilities. Integration with external tools and libraries enhances the app's functionality and versatility.

Evidence

1. Login Form

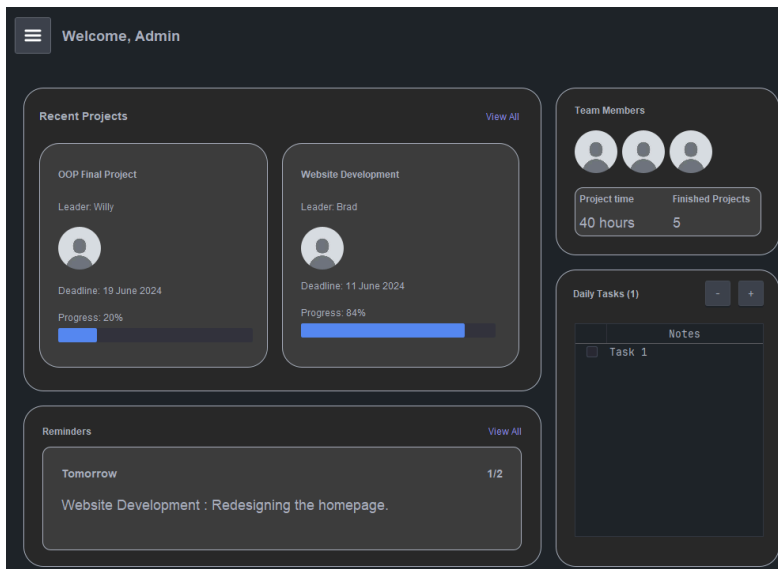
A screenshot of a login form on a dark background. The form consists of two text input fields: the first is labeled 'Username:' and the second is labeled 'Password:'. Below these fields is a button labeled 'Login'.

The figure above shows the initial login page when the run button is clicked.

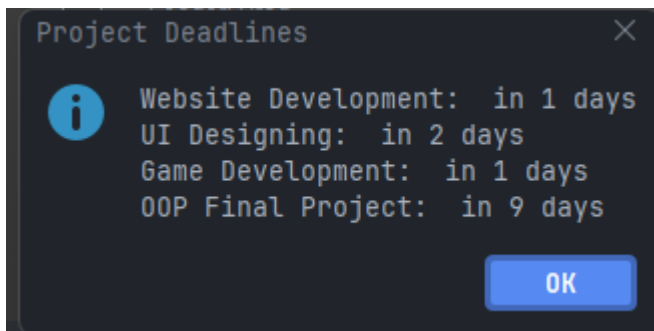


The figure above shows when there is an error when inputting the login credentials.

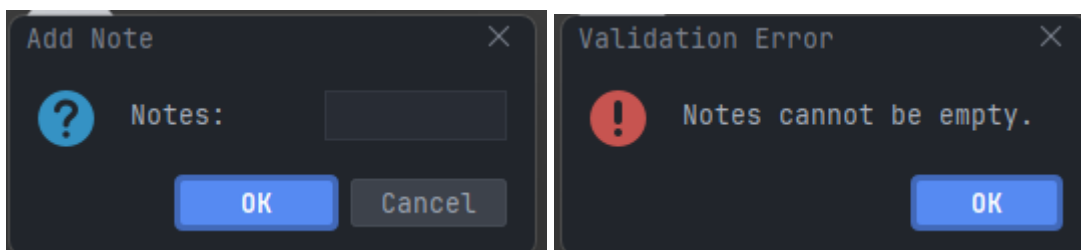
2. Dashboard Form



The figure above shows the initial dashboard form while loading the .json file that has been prefilled. The view all in recent projects will redirect the form to Project Form, which will be explained at number 3.

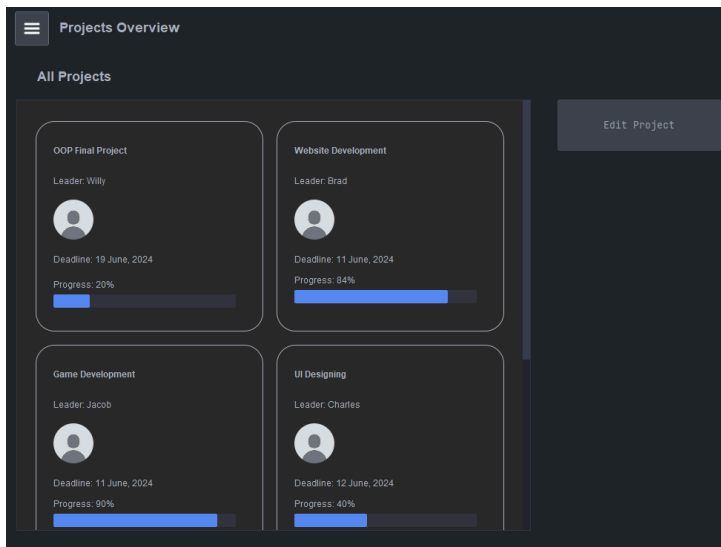


This popup shows when the view all in Reminders is clicked, it shows all the upcoming deadlines.



This popup will show when the “+” button is clicked at the Daily Tasks box, and when the note is still empty, then the validation error will show.

3. Project Form

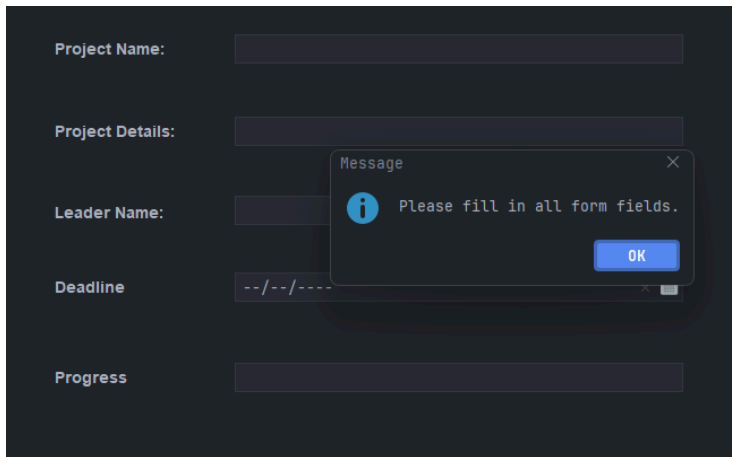


This is the form for Projects when the view for recent projects in dashboard form is clicked, it will show the projects based on the last edit. The scrollpane will show all the projects available in the .json file. And if the edit project is clicked, it will redirect to another form, AddProject Form, and will be explained in the next point.

4. AddProject Form

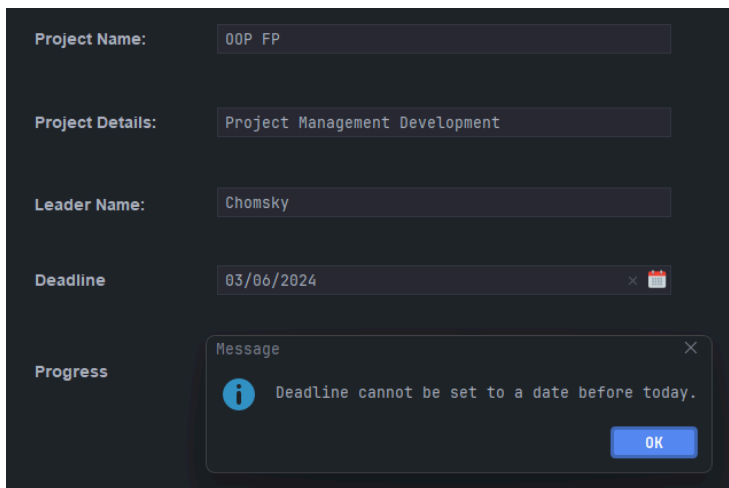
The screenshot shows the 'AddProject Form'. It has a 'BACK' button in the top left corner. The form contains five input fields: 'Project Name', 'Project Details', 'Leader Name', 'Deadline' (with a date picker icon), and 'Progress'. On the right side of the form, there are two buttons: 'Register' and 'Delete'.

This form will be used to add more projects. And, the initial form shown will be like this, and the user will need to input for every field.



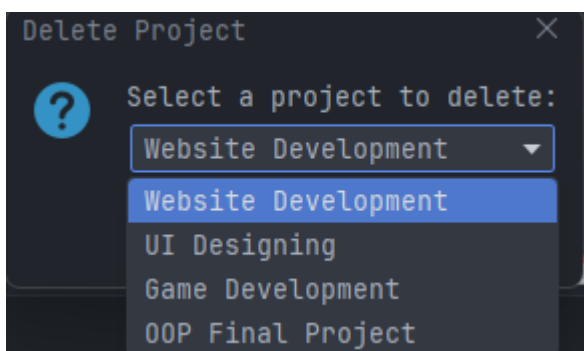
The screenshot shows a registration form with the following fields: Project Name, Project Details, Leader Name, Deadline, and Progress. A modal message box is displayed over the form, containing an information icon, the text "Please fill in all form fields.", and an "OK" button.

This message will popup if you click register while all fields are not filled.



The screenshot shows the same registration form, but now the fields are filled: Project Name is "OOP FP", Project Details is "Project Management Development", Leader Name is "Chomsky", and Deadline is "03/06/2024". A modal message box is displayed, containing an information icon, the text "Deadline cannot be set to a date before today.", and an "OK" button.

Another error message will be displayed if the deadline set is before today.



The screenshot shows a "Delete Project" dialog box. It features a question mark icon and the text "Select a project to delete:". Below this is a dropdown menu with the following options: "Website Development", "Website Development", "UI Designing", "Game Development", and "OOP Final Project". The first "Website Development" option is currently selected and highlighted.

And when you click the delete button, it will show a popup with a dropdown containing all the project names.