

SCHEDULE MANAGER

CS23333-OBJECT ORIENTED PROGRAMMING USING JAVA

Submitted by

Kailash T S(231001082)

Lalith D(231001098)

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM
(An Autonomous Institution)



DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM ,CHENNAI 600 025

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project titled “SCHEDULE MANAGER” is the Bonafide work of “**KAILASH T S(231001082)**,” **LALITH D (231001098)**” who carried out the project work under my supervision.

SIGNATURE

Dr. P. Valarmathie

HEAD OF THE DEPARTMENT

Information Technology
Rajalakshmi Engineering College,
Rajalakshmi Nagar, Thandalam
Chennai – 602105

SIGNATURE

Mrs. Sangeetha

COURSE INCHARGE

Information Technology
Rajalakshmi Engineering College
Rajalakshmi Nagar, Thandalam
Chennai – 602105

This project is submitted for CS23333 – Introduction to Oops and Java held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

1. SCHEDULE MANAGER

1.1. Abstract----- 5

1.2. Introduction ----- 5

1.3. Purpose ----- 5

1.4. Scope of Project-----5

1.5. Software Requirement Specification ----- 6

2. System Flow Diagrams ----- 11

2.1. Use Case Diagram -----11

2.2. Entity-relationship Diagrams -----11

3. Module Description----- 12

4. Implementation

4.1. Design ----- 13

4.2. Database Design	15
4.3. Code	18
5. Conclusion	21
References	22

SCHEDULE MANAGER

1.1 Abstract

The Schedule Manager Web Application helps users manage personal and professional schedules by allowing them to add, edit, delete, and track task statuses. Built with Java for the backend and MySQL for secure data storage, it offers a responsive frontend for seamless interaction across devices. The app helps improve time management, reduce stress, and adapt to various scheduling needs, whether for daily tasks or complex projects.

1.2 Introduction

The Schedule Manager Web Application was developed to address the challenges of disorganized schedules and poor time management. Traditional tools lack flexibility, prompting the creation of a dynamic task management system that allows users to add, edit, delete, and track tasks efficiently. Built with Java for the backend and MySQL for secure database management, the application ensures reliability, scalability, and high performance. Its clean, intuitive interface and real-time task monitoring empower users to take control of their schedules, reduce stress, and boost productivity, making it an essential tool for both personal and professional time management.

1.3 Purpose

- 2 • **Efficient Task Management:** To provide users with an easy way to manage and track their daily tasks, helping them stay organized and focused on their goals.
- 3 • **Time Optimization:** To help users prioritize tasks, set deadlines, and allocate time effectively to maximize productivity and minimize procrastination.
- 4 • **User-Friendly Interface:** To create an intuitive and easy-to-navigate interface that ensures a seamless experience for users of all technical backgrounds.
- 5 • **Task Status Tracking:** To allow users to update and track the progress of their tasks by marking them as pending, completed, or in-progress, offering better visibility.
- 6 • **Customizable Task Options:** To enable users to add, edit, and delete tasks as needed, giving them full control over their schedule.
- 7 • **Secure Data Storage:** To ensure the secure storage of user data and tasks using a MySQL database, safeguarding personal information and task details.
- 8 • **Scalability:** To create a scalable solution that can accommodate increasing numbers of users and tasks without sacrificing performance.
- 9 • **Cross-Device Accessibility:** To enable users to access and manage their schedule from any device with an internet connection, enhancing convenience and flexibility.

1.4 Scope of the Project

The Schedule Manager System is designed to provide users with a comprehensive tool for managing and organizing their tasks and schedules. Its scope extends to allowing

users to create, update, delete, and track tasks effectively. The system is built to support both personal and professional task management, offering features that enable users to monitor task statuses, set deadlines, and manage multiple tasks at once.

At the heart of this system is a secure database management setup, utilizing MySQL for efficient storage and retrieval of user task data. The system interacts with the database through JDBC (Java Database Connectivity), ensuring secure and reliable data handling. Tasks are organized in a way that allows easy updates and filtering, helping users stay on top of their schedules.

Key functionalities within the system include:

- **Task Management:** Users can add, edit, delete, and categorize tasks.
- **Task Tracking:** The system provides the ability to track tasks by their statuses, allowing users to view tasks based on categories like "Pending," "In-Progress," or "Completed."
- **User Authentication:** The system ensures that only authorized users can access and manage their tasks.
- **Reports:** Future versions of the system can incorporate detailed reports for task progress, completion rates, and performance metrics.

1.5 Software Requirement Specification

1. Database Integration:

- **MySQL Database:** The system uses a MySQL database to store user data, including tasks, deadlines, and task statuses. This database is connected using JDBC (Java Database Connectivity), enabling efficient data storage and retrieval.
- **Schema Design:** The database schema includes a single table named tasks with columns for task description, deadlines, statuses (e.g., Pending, In-Progress, Completed), and user IDs to ensure proper user-task mapping.

2. Backend Development:

- **Server Setup:** The backend server is implemented using Java, utilizing popular frameworks like Spring Boot for RESTful APIs and MySQL database interaction. The backend handles various endpoints to manage tasks and user data securely.
- **Endpoints:**
 - **/api/addTask:** Handles the addition of new tasks into the system.
 - **/api/updateTask:** Updates the status and details of an existing task.
 - **/api/deleteTask:** Allows users to delete a task they no longer need.
 - **/api/getTasks:** Retrieves all tasks associated with a specific user.
 - **/api/login:** Manages user authentication, ensuring secure access to the task management features.

3. Frontend Development:

- **HTML and CSS:** The frontend is designed using HTML for structure and CSS (via Tailwind CSS) for styling, ensuring a clean, modern, and responsive user interface for task management.
- **JavaScript:** Utilizes JavaScript for dynamic interactions, such as handling form submissions, rendering tasks, and updating the task status in real-time. This allows a

seamless experience for users.

4. Functionality:

- **Task Management:** Users can add, edit, and delete tasks directly from the interface. They can also set deadlines and mark tasks with different statuses (Pending, In-Progress, Completed).
- **Task Tracking:** The system tracks each task's status, displaying tasks based on their current status for better organization.
- **User Authentication:** The system includes a secure login mechanism, ensuring only authorized users can access and manage their tasks.
- **Task Sorting and Filtering:** Users can sort tasks by deadline or status and filter tasks to view only specific categories (e.g., only completed tasks).

5. Technology Stack:

- **Frontend:** HTML, CSS (Tailwind CSS), JavaScript
- **Backend:** Java (Spring Boot), MySQL, JDBC
- **Tools and Libraries:** JSON for data exchange, JWT for user authentication, Spring Boot for API development and task management functionality.

References and Acknowledgements:

[1] <https://www.javatpoint.com/java-awt>

[2] <https://www.javatpoint.com/java-swing>

Overall Description

The **Schedule Manager Web Application** provides a comprehensive solution for managing personal and professional schedules. It offers users the ability to create, view, update, and delete tasks, as well as track the status and deadlines of each task. The application enhances time management by allowing users to prioritize tasks effectively and stay organized.

Product Perspective

The system follows a client/server architecture, ensuring compatibility with various operating systems. The frontend is developed using HTML, CSS (with Tailwind CSS), and JavaScript, delivering a responsive and intuitive user interface. The backend is implemented in Java using Spring Boot, handling RESTful APIs for task management, and is connected to a MySQL database for storing and retrieving task-related data using JDBC (Java Database Connectivity).

Product Functionality

a) **Add Task:** Allows users to input and store task details, including descriptions, deadlines, and statuses (e.g., Pending, In-Progress, Completed).

- b) **Update Task:** Enables users to modify existing task details, such as status, description, or deadline.
- c) **Delete Task:** Provides the option for users to remove tasks from their schedule.
- d) **View Tasks:** Displays a list of all tasks associated with the user, including filtering options based on status or deadlines.
- e) **Task Status Tracking:** Tracks and updates the status of tasks in real-time to ensure users have an up-to-date overview of their schedules.
- f) **User Authentication:** Ensures secure access to the system by verifying user credentials during login, allowing only authorized users to access the system.

User Characteristics

- **Qualification:** Users should have a basic understanding of time management and scheduling practices.
- **Experience:** Prior experience with basic scheduling tools or task management systems is beneficial but not required.
- **Technical Skills:** Users should be comfortable with interacting with a web-based interface and have basic knowledge of using digital tools for task management.

Hardware Requirements:

- Processor: Any processor with an Intel i3 or higher.
- Operating System: Windows 8, 10, or 11.
- Processor Speed: 2.0 GHz or higher.
- RAM: 4GB or more.
- Hard Disk: 500GB or greater.

Software Requirements:

- Database: MySQL (using JDBC for efficient interaction).
- Frontend: HTML, CSS (with Tailwind CSS), JavaScript (with libraries like Chart.js).
- Backend: Java (with Spring Boot for API development).

Constraints:

- User Access: The system limits access to authorized users, such as administrators and specific users, ensuring only authenticated individuals can manage tasks and view personal data.
- Delete Operation: The ability to delete tasks is restricted to administrators, and extra safeguards are not implemented in this version for simplicity.
- Caution During Deletion: Administrators must be cautious while deleting tasks to ensure data

integrity and avoid unintended data loss.

Assumptions and Dependencies:

- Administrator Responsibilities: Administrators will be in charge of creating login credentials and ensuring secure distribution of IDs and passwords to authorized users.
- User Knowledge: Users are assumed to have basic knowledge of web-based applications and task management principles, making the interface user-friendly for individuals with minimal technical expertise.

Specific Requirements:

User Interface:

The Schedule Manager Web Application provides a simple and intuitive user interface with the following features:

- Login: Secure login for authorized users to access the system.
- Add Task: Allow users to input and store task descriptions, deadlines, and statuses.
- View Tasks: Display all tasks with options for filtering and sorting.
- Update Task: Update existing task details such as status, description, or deadline.
- Delete Task: Allows users to remove tasks that are no longer needed.
- Task Tracking: Real-time status updates and task progress tracking.
- User Authentication: A secure mechanism for managing user access to the system, ensuring data privacy.

Hardware Interface:

- Screen resolution of at least 640 x 480 or higher.
- Compatible with any version of Windows 8, 10, or 11.

Software Interface for Schedule Manager Web Application:

- Operating System: MS-Windows (Windows 8, 10, or 11).
- Frontend Development: HTML, CSS (Tailwind CSS), JavaScript (with libraries such as Chart.js).
- Backend Development: Java (using Spring Boot).
- Database: MySQL (with JDBC integration for data management).
- Integrated Development Environment (IDE): IntelliJ IDEA or Eclipse.

Functional Requirements for Schedule Manager

Login Module (LM):

Users (admins) can access the Login Module via a secure login page.

The system supports login using a username and password.

Passwords are masked to ensure security.

Only authorized users, whose credentials match those in the database, are granted access to the system.

Task Management Module (TMM):

After successful login, users are granted access to the main features of the application.

Users can view detailed information about tasks, including descriptions, deadlines, and statuses.

Users have the ability to add new tasks, update existing ones, and delete tasks when needed.

Task data is securely stored and retrieved from the database, with real-time updates for task status and deadlines.

Administrator Module (AM):

After logging in successfully, the system displays the administrative functions for admins.

Admins can manage task data: adding new tasks, updating existing tasks, and deleting tasks no longer needed.

The "Add Task" function allows admins to input new task details, while the "Update Task" function allows modifications to existing task information.

All add, update, or delete actions trigger communication with the backend (via the Server Module) to ensure the database reflects the changes accurately.

Server Module (SM):

The Server Module serves as the intermediary between the frontend and the database.

It receives and processes requests from the frontend, ensuring proper data formatting and handling the system's functionality.

The Server Module manages communication with the database to validate and execute requests, ensuring data integrity and consistency, especially for tasks, deadlines, and statuses.

It handles secure user authentication and ensures only authorized users can perform administrative actions.

CHAPTER 2

System Flow Diagrams

2.1 Use Case Diagrams :

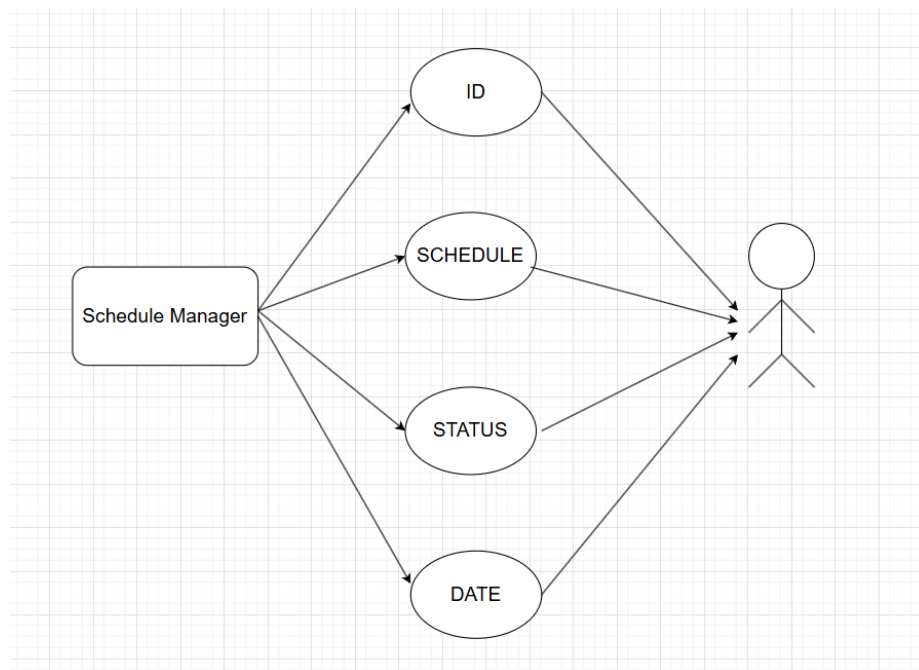


Fig.2.1.Case Diagram

2.2 Entity-relationship diagram

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

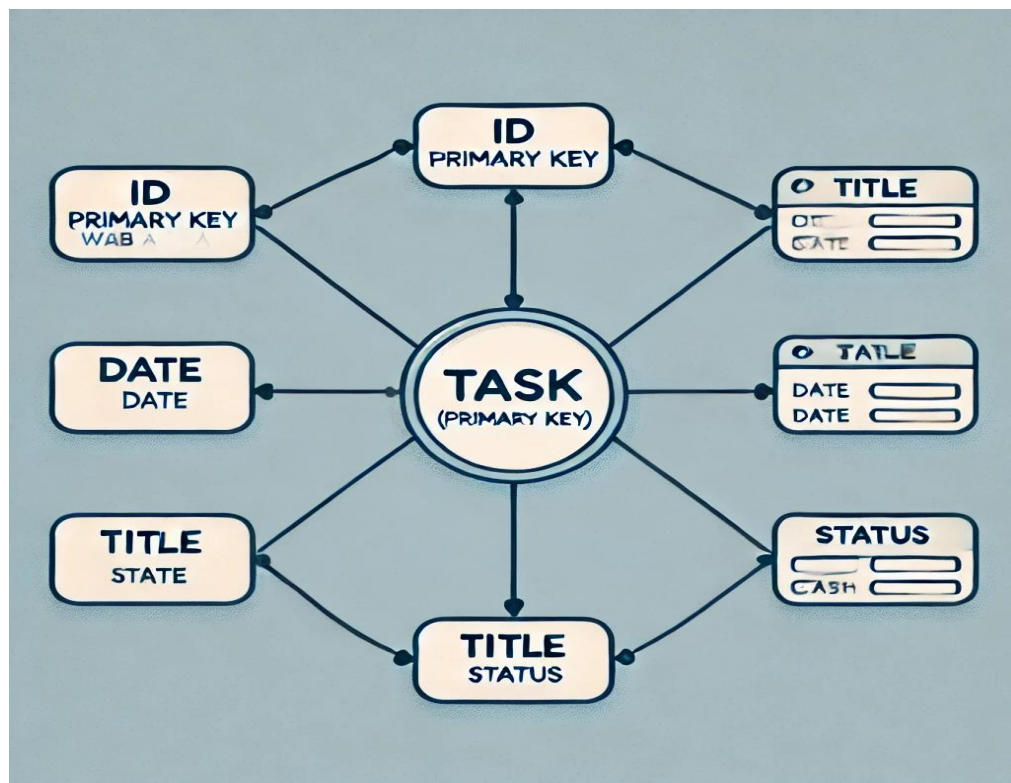


Fig 3.2 ER DIAGRAM OF SCHEDULE MANAGER

CHAPTER 3

Module description:

❖ View List:

- The admin can view a list of all tasks in the system, displaying task details such as the title, due date, and current status. This page allows the admin to monitor task progress and see important deadlines and statuses.

❖ Add List:

- The admin can add new tasks to the system by entering details like the task title, due date, and status. The system saves this data to the database for future reference and task tracking.

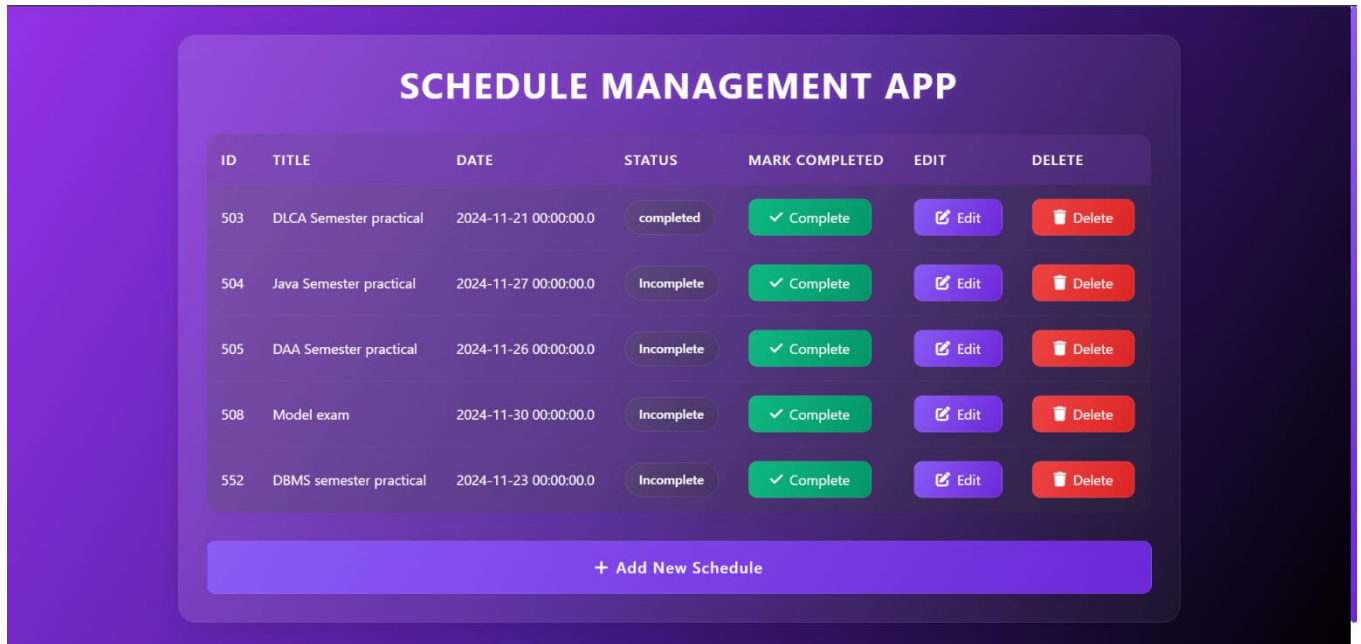
❖ Edit List:

- The admin can edit the details of existing tasks, including updating the task title, changing the due date, or modifying the task status (e.g., from "Pending" to "In-Progress" or "Completed"). This allows the admin to keep the task list up-to-date and manage tasks effectively.

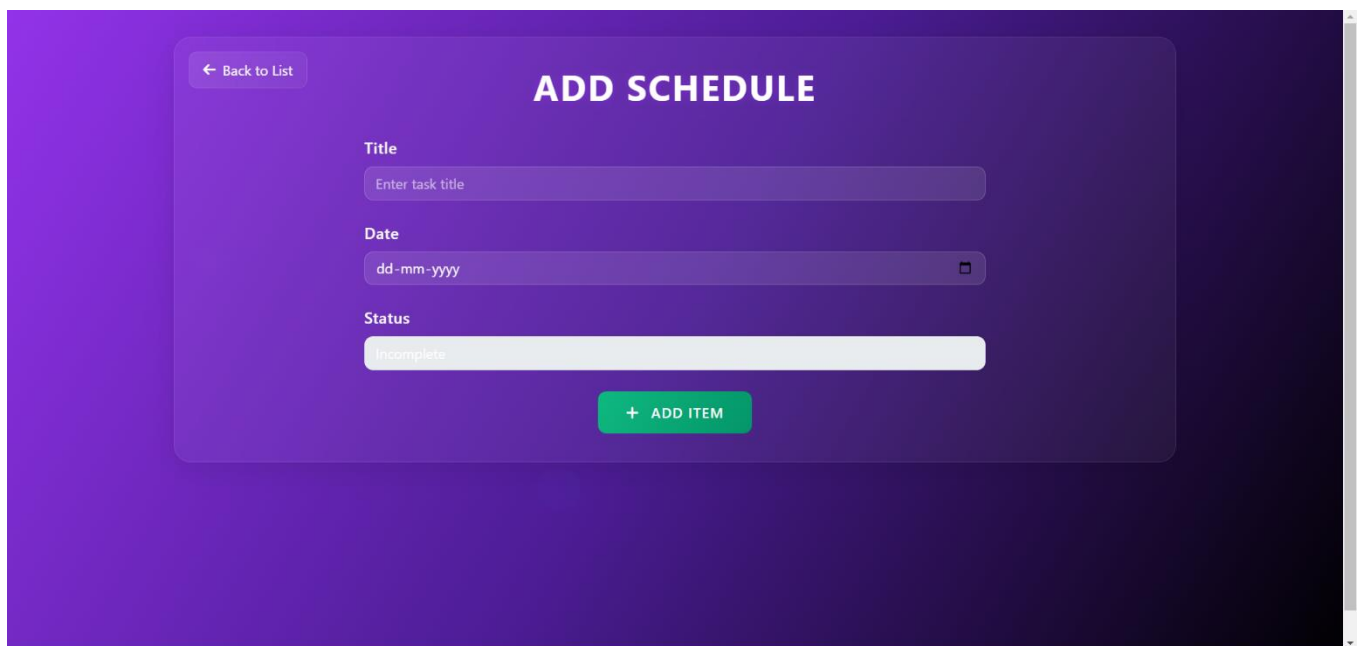
CHAPTER 4

4.1 DESIGN

1.View Schedule page



2.Add schedule page



3.Edit Schedule page

[← Back to List](#)

EDIT SCHEDULE

Title

DLCA Semester practical

Date

21-11-2024

Status

Incomplete

SAVE CHANGES

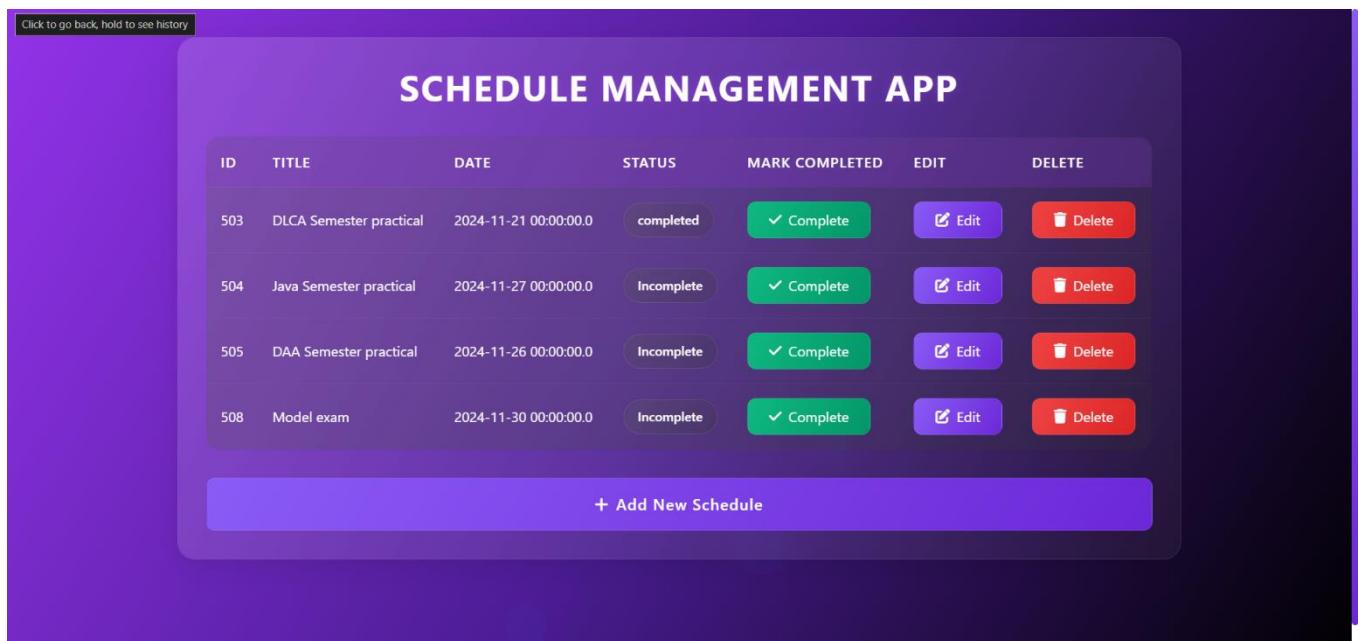
4.Delete Operation on Schedule manager

localhost:8080 says
deleted

ID	TITLE	DATE	STATUS	MARK COMPLETED	EDIT	DELETE
503	DLCA Semester practical	2024-11-21 00:00:00.0	completed	✓ Complete	Edit	Delete
504	Java Semester practical	2024-11-27 00:00:00.0	Incomplete	✓ Complete	Edit	Delete
505	DAA Semester practical	2024-11-26 00:00:00.0	Incomplete	✓ Complete	Edit	Delete
508	Model exam	2024-11-30 00:00:00.0	Incomplete	✓ Complete	Edit	Delete
652	aef	2024-11-07 00:00:00.0	Incomplete	✓ Complete	Edit	Delete

+ Add New Schedule

5. Page after Deletion of data



4.2 Database Design

For the Schedule Manager System, the data is stored and retrieved from a MySQL database, chosen for its efficiency in managing structured data and supporting relational operations. The system uses the database to store task details such as task title, deadlines, statuses, and the user ID for task management.

Data Elements and Structures:

The required data elements for the Schedule Manager include task titles, deadlines, statuses (e.g., Pending, In-Progress, Completed), and user IDs. These elements are structured in a way that allows easy retrieval and modification, ensuring efficient task management.

Normalization:

The database schema is normalized to ensure the elimination of data redundancy and to improve data integrity. This normalization process organizes the data into efficient tables, helping in reducing data duplication and making the database scalable. It also ensures that tasks can be managed effectively as the system grows.

Data Integrity:

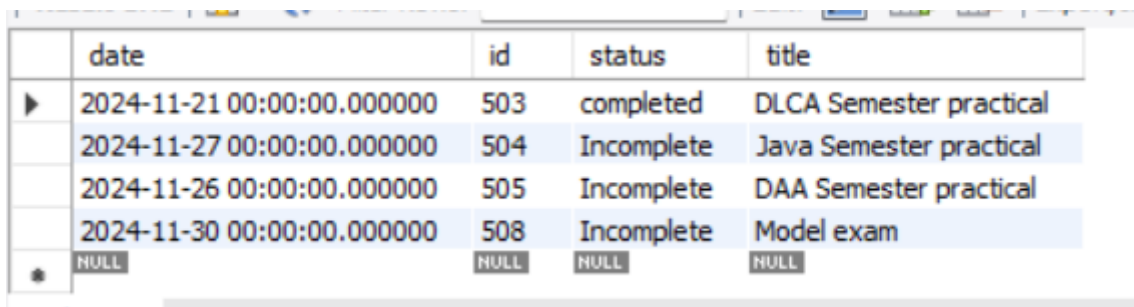
To maintain consistency and accuracy, the relationships between tasks, their statuses, and user IDs are defined. The normalization helps in maintaining data integrity by avoiding data anomalies and ensuring easy updates or deletions.

MySQL Database:

MySQL is chosen for its flexibility, scalability, and reliability in handling relational data. It ensures quick and efficient access to the data, allowing for seamless management of tasks while minimizing the risk of data inconsistencies.

The database design is optimized to ensure that storage is used efficiently, and the system remains stable as it scales.

Schedule manager example table:



The image shows a screenshot of a MySQL database interface displaying a table with the following data:

	date	id	status	title
▶	2024-11-21 00:00:00.000000	503	completed	DLCA Semester practical
	2024-11-27 00:00:00.000000	504	Incomplete	Java Semester practical
	2024-11-26 00:00:00.000000	505	Incomplete	DAA Semester practical
	2024-11-30 00:00:00.000000	508	Incomplete	Model exam
•	NULL	NULL	NULL	NULL

Fig 4.2. SQL Database

4.2 IMPLEMENTATION(CODE)

//JDBC CONNECTIVITY

```
spring.port = 8080
spring.mvc.view.prefix = /WEB-INF/jsp/
spring.mvc.view.suffix = .jsp

spring.datasource.url = jdbc:mysql://localhost:3306/todo_app
spring.datasource.username = root
spring.datasource.password = tskailashts0807@ @

spring.jpa.hibernate.ddl-auto = update
```

//CONTROLLER

```
package com.example.ToDo.App.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.example.ToDo.App.model.ToDo;
import com.example.ToDo.App.service.ToDoService;

@Controller
public class ToDoController {

    @Autowired
    private ToDoService service;

    @GetMapping("/{", "viewToDoList"})
    public String viewAllToDoItems(Model model, @ModelAttribute("message") String
message) {
        model.addAttribute("list", service.getAllToDoItems());
        model.addAttribute("message", message);

        return "ViewToDoList";
    }

    @GetMapping("/updateToDoStatus/{id}")
    public String updateToDoStatus(@PathVariable Long id, RedirectAttributes
redirectAttributes) {
        if(service.updateStatus(id)) {
            redirectAttributes.addFlashAttribute("message", "Update Success");
        }
    }
}
```

```

        return "redirect:/viewToDoList";
    }

    redirectAttributes.addFlashAttribute("message", "Update Failure");
    return "redirect:/viewToDoList";
}

@GetMapping("/addToDoItem")
public String addToDoItem(Model model) {
    model.addAttribute("todo", new ToDo());

    return "AddToDoItem";
}

@PostMapping("/saveToDoItem")
public String saveToDoItem(ToDo todo, RedirectAttributes redirectAttributes) {
    if(service.saveOrUpdateToDoItem(todo)) {
        redirectAttributes.addFlashAttribute("message", "save success");
        return "redirect:/viewToDoList";
    }

    redirectAttributes.addFlashAttribute("message", "save failure");
    return "redirect:/addToDoItem";
}

@GetMapping("/editToDoItem/{id}")
public String editToDoItem(@PathVariable Long id, Model model) {
    model.addAttribute("todo", service.getToDoItemById(id));

    return "EditToDoItem";
}

@PostMapping("/editSaveToDoItem")
public String editSaveToDoItem(ToDo todo, RedirectAttributes redirectAttributes) {
    if(service.saveOrUpdateToDoItem(todo)) {
        redirectAttributes.addFlashAttribute("message", "Edit success");
        return "redirect:/viewToDoList";
    }

    redirectAttributes.addFlashAttribute("message", "Edit failure");
    return "redirect:/editToDoItem/"+todo.getId();
}

@GetMapping("/deleteToDoItem/{id}")
public String deleteToDoItem(@PathVariable Long id, RedirectAttributes
redirectAttributes) {
    if(service.deleteToDoItem(id)) {
        redirectAttributes.addFlashAttribute("message", "Delete success");
        return "redirect:/viewToDoList";
    }

    redirectAttributes.addFlashAttribute("message", "Delete failure");
    return "redirect:/viewToDoList";
}
}

```

//MySQL SCHEMA

```
package com.example.ToDo.App.model;
import java.util.Date;

import org.springframework.format.annotation.DateTimeFormat;

import jakarta.annotation.Nonnull;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Table;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
@Table (name="todo")

public class ToDo {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Nonnull
    private Long id;

    @Column
    @Nonnull
    private String title;

    @Column
    @Nonnull
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date date;

    @Column
    @Nonnull
    private String status;

    public ToDo() {
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }
}
```

```
    public void setTitle(String title) {  
        this.title = title;  
    }  
  
    public Date getDate() {  
        return date;  
    }  
  
    public void setDate(Date date) {  
        this.date = date;  
    }  
  
    public String getStatus() {  
        return status;  
    }  
  
    public void setStatus(String status) {  
        this.status = status;  
    }  
  
}
```

CHAPTER 5

Conclusion

The **Schedule Manager System** serves as an effective tool for users to manage their tasks, deadlines, and statuses in an organized manner. By offering features such as adding, editing, and deleting tasks, the system enables users to track and update their tasks seamlessly. The secure user authentication ensures that only authorized individuals can access and manage the tasks, while the database structure is designed for scalability and efficient data retrieval.

The integration of **MySQL** as the database management system guarantees that task data is stored in a reliable and consistent manner, minimizing redundancy and improving data integrity. Through a combination of **Java (Spring Boot)** for the backend and a **responsive frontend** built using **HTML, CSS (Tailwind CSS), and JavaScript**, the system offers an intuitive interface for users to interact with the application, making it easier to track progress and meet deadlines. The use of **RESTful APIs** ensures smooth communication between the frontend and backend, enhancing the overall user experience.

The successful implementation of the **Schedule Manager System** demonstrates the potential for streamlining task management across various sectors. Whether for personal or professional use, this system provides a robust and flexible platform to manage tasks and schedules effectively. As technology continues to evolve, further enhancements can be made to the system, such as integrating advanced features like notifications, task prioritization, and synchronization with external calendars, providing even more convenience and efficiency for users.

The development of this system has not only improved personal time management but also highlighted the importance of effective software architecture, data integrity, and secure user access in web-based applications. Moving forward, the system can be further optimized by adding features like mobile responsiveness, multi-user access, and machine learning-based task suggestions, expanding its utility for a wider range of users.

References

1. **Java MySQL Database Connectivity (JDBC):**

<https://www.javatpoint.com/java-mysql>

This reference provides detailed information on how Java interacts with MySQL databases using JDBC, which is used in the backend of the Schedule Manager System.

2. **Spring Boot Documentation:**

<https://spring.io/projects/spring-boot>

A detailed guide on how to use **Spring Boot** for building web applications, which is used for backend development in this system.

3. **Tailwind CSS Documentation:**

<https://tailwindcss.com/docs>

Official documentation for **Tailwind CSS**, used for designing the frontend of the Schedule Manager System with a responsive, modern user interface.

4. **Chart.js Documentation:**

<https://www.chartjs.org/docs/latest/>

For integrating dynamic, interactive charts into the web app, especially if adding visual task progress tracking in future updates.

5. **MySQL Documentation:**

<https://dev.mysql.com/doc/>

Comprehensive resources and documentation for **MySQL**, explaining how to manage databases, queries, and optimization techniques.

6. **HTML and CSS for Beginners:**

<https://www.w3schools.com/>

This site provides foundational knowledge on **HTML** and **CSS**, essential for frontend development in the project.