Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

II Year M.Tech Integrated CSE (III Semester)

Academic Year 2024-25

ICS1312 – Java Programming Lab

Mini Project Report

Prepared By :

Harini J **Register Number :** 3122237001015

R Jayasree **Register Number :** 3122237001019

Sno	Title	Pg no
1	Problem Statement	2
2	Motivation for the problem statement	2
3	Scope and Limitations	2-3
4	Design of the Solution (Class Diagram)	4-6
5	Modules Split-up	6
6	Implementation	7-9
7	Output Screenshots	10-14
8	Object Oriented Features Used	15-17
9	Inference and Future Extension	17-18

* **Problem Statement :**

To design and implement a Java-based Task Management System that allows users to efficiently organize and manage their tasks. The system will support functionalities for creating, updating, and categorizing tasks, with CSV files used for persistent storage. Regular users can manage personal tasks, while admin users have additional capabilities to assign and manage work tasks. The objective is to provide a streamlined backend solution, accessible through the terminal, that enhances productivity through organized task handling.

* **Motivation :**

In today's digital world, individuals and teams have to manage multiple tasks and deadlines, making it challenging to stay organized and productive. This project is motivated by the need for a straightforward and simple solution that enables users to manage personal and work-related tasks with ease. By building a Java-based backend application that stores data in CSV files, we aim to create a lightweight, accessible system that offers essential task management functionalities, helping users prioritize, track, and complete tasks efficiently. This project seeks to provide a reliable tool that enhances productivity by simplifying task management in a minimalistic, user-friendly manner.

* **Scope:**

1. The Task Management System enables users to add, update, categorize, and delete tasks, supporting efficient task management for personal and work purposes.

2. CSV files are used for data storage, providing a lightweight and easily manageable alternative to complex databases.

3. Regular users can create and manage personal tasks, while admin users have extended functionalities, including assigning and managing group work tasks.

4. The system supports task prioritization, tagging, deadlines, and recurring task scheduling, making it versatile for various types of task tracking.

5. All interactions occur via the terminal, making it suitable for environments where a GUI is unnecessary or infeasible.

* **Limitations:**

1. The lack of a graphical user interface (GUI) may make the system less accessible for users unfamiliar with terminal commands.

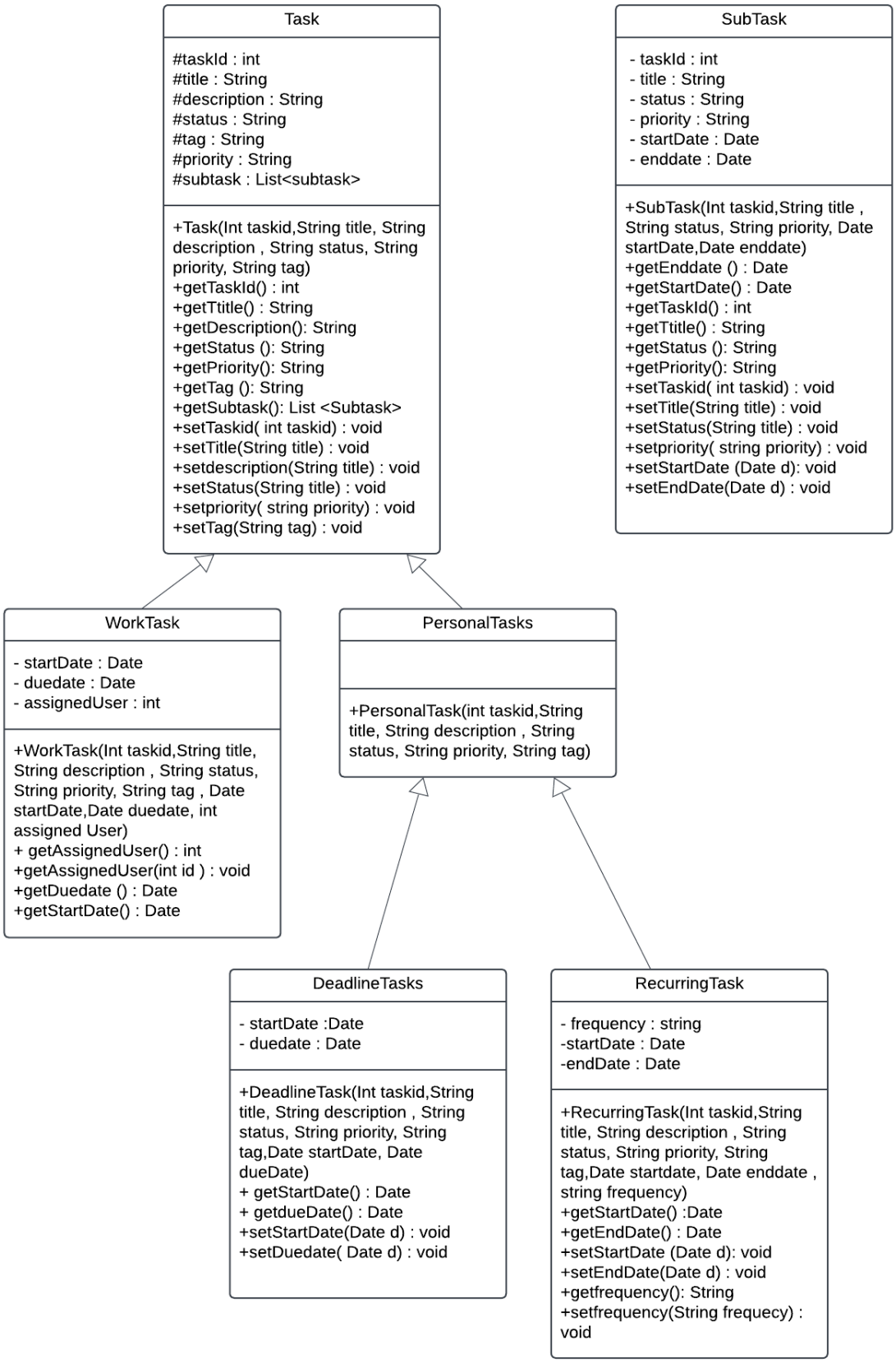
2. Using CSV files for data storage has limitations in handling large volumes of data, as it may affect performance compared to database solutions.

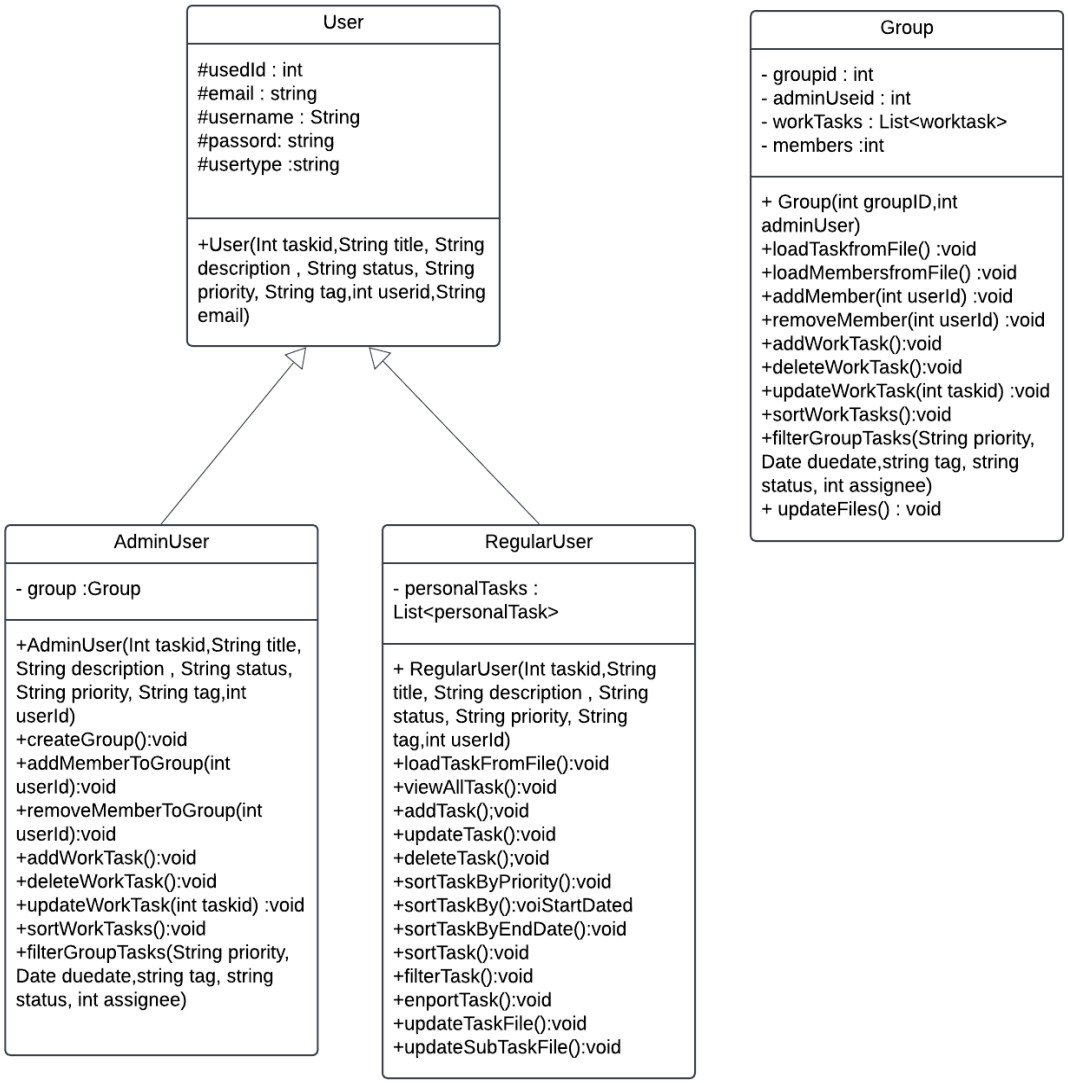
3. There is no real-time collaboration or multi-user support, so tasks cannot be shared or updated by multiple users simultaneously.

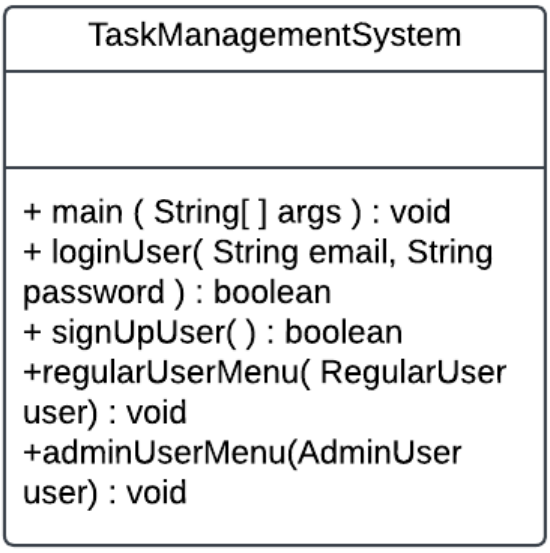
4. Limited customization options are available, as the system is designed primarily for general task management rather than industry-specific needs.

5. Advanced features like notifications, reminders, or integrations with other tools (e.g., email or calendar) are not included, focusing the system on core task management only.

* **Design of the Solution (Class Diagram) :**


Task
#taskId : int
#title : String
#description : String
#status : String
#tag : String
#priority : String
#subtask : List<subtask>
+Task(Int taskid,String title, String description , String status, String priority, String tag)
+getTaskId() : int
+getTtitle() : String
+getDescription(): String
+getStatus (): String
+getPriority(): String
+getTag (): String
+getSubtask(): List <Subtask>
+setTaskid(int taskid) : void
+setTitle(String title) : void
+setdescription(String title) : void
+setStatus(String title) : void
+setpriority(string priority) : void
+setTag(String tag) : void
SubTask
- taskId : int
- title : String
- status : String
- priority : String
- startDate : Date
- enddate : Date
+SubTask(Int taskid,String title , String status, String priority, Date startDate,Date enddate)
+getEnddate () : Date
+getStartDate() : Date
+getTaskId() : int
+getTtitle() : String
+getStatus (): String
+getPriority(): String
+setTaskid(int taskid) : void
+setTitle(String title) : void
+setStatus(String title) : void
+setpriority(string priority) : void
+setStartDate (Date d): void
+setEndDate(Date d) : void
WorkTask
- startDate : Date
- duedate : Date
- assignedUser : int
+WorkTask(Int taskid,String title, String description , String status, String priority, String tag , Date startDate,Date duedate, int assigned User)
+ getAssignedUser() : int
+getAssignedUser(int id) : void
+getDuedate () : Date
+getStartDate() : Date
PersonalTasks
+PersonalTask(int taskid,String title, String description , String status, String priority, String tag)
DeadlineTasks
- startDate :Date
- duedate : Date
+DeadlineTask(Int taskid,String title, String description , String status, String priority, String tag,Date startDate, Date dueDate)
+ getStartDate() : Date
+ getdueDate() : Date
+setStartDate(Date d) : void
+setDuedate(Date d) : void
RecurringTask
- frequency : string
-startDate : Date
-endDate : Date
+RecurringTask(Int taskid,String title, String description , String status, String priority, String tag,Date startdate, Date enddate , string frequency)
+getStartDate() :Date
+getEndDate() : Date
+setStartDate (Date d): void
+setEndDate(Date d) : void
+getfrequency(): String
+setfrequency(String frequecy) : void


User
#usedId : int
#email : string
#username : String
#passord: string
#usertype :string
+User(Int taskid,String title, String description , String status, String priority, String tag,int userid,String email)
Group
- groupid : int
- adminUseid : int
- workTasks : List<worktask>
- members :int
+ Group(int groupID,int adminUser)
+loadTaskfromFile() :void
+loadMembersfromFile() :void
+addMember(int userId) :void
+removeMember(int userId) :void
+addWorkTask():void
+deleteWorkTask():void
+updateWorkTask(int taskid) :void
+sortWorkTasks():void
+filterGroupTasks(String priority, Date duedate,string tag, string status, int assignee)
+ updateFiles() : void
AdminUser
- group :Group
+AdminUser(Int taskid,String title, String description , String status, String priority, String tag,int userId)
+createGroup():void
+addMemberToGroup(int userId):void
+removeMemberToGroup(int userId):void
+addWorkTask():void
+deleteWorkTask():void
+updateWorkTask(int taskid) :void
+sortWorkTasks():void
+filterGroupTasks(String priority, Date duedate,string tag, string status, int assignee)
RegularUser
- personalTasks : List<personalTask>
+ RegularUser(Int taskid,String title, String description , String status, String priority, String tag,int userId)
+loadTaskFromFile():void
+viewAllTask():void
+addTask();void
+updateTask():void
+deleteTask();void
+sortTaskByPriority():void
+sortTaskBy():voiStartDated
+sortTaskByEndDate():void
+sortTask():void
+filterTask():void
+enportTask():void
+updateTaskFile():void
+updateSubTaskFile():void



* **Modules Split-up :**

1. User Module

- **Classes**: User.java, AdminUser.java, RegularUser.java
- **Description**: Defines user roles. User is the base class, AdminUser has functionalities for managing work tasks , while RegularUser has access to personal task management. Each RegularUser has a List<PersonalTask> to store their tasks.

2. Task Module

- **Classes**: Task.java, PersonalTask.java, WorkTask.java, DeadlineTask.java, RecurringTask.java, SubTask.java
- **Description**: Contains task-related classes. Task is the base class with attributes like title, description, status, and priority. PersonalTask and WorkTask are subclasses representing different task types, while DeadlineTask and RecurringTask extend PersonalTask to handle specific task requirements like deadlines and recurring schedules.

3. Main Module

- **Classes**: TaskManagementSystem.java
- **Description**: Main module is the core entry point, responsible for user login, input handling, and menu navigation. It sets up the environment and initializes

all the key functionalities and data. Also has the CSV files - users.csv, personal_tasks.csv, subtasks.csv, groups.csv, and work_tasks.csv for storing the data.

* **Implementation :**

Program Flow :

1. User Login :
    - The program asks to sign up if the user is new to the application.
    - Then the user log in with their username and password which are validated from the users.csv file.
    - After successful login, a role-based menu is displayed:
        - Regular User Menu: Shows options specific to managing personal tasks.
        - Admin User Menu: Shows options with additional admin functionalities for managing work tasks and groups.
2. Input Handling and Menu Options :
    - The user selects an option from the menu, and the system processes the input by invoking corresponding methods.
    - Below are typical options for each user role:

3. Regular User Functionalities :

- View Personal Tasks

  The program reads personal_tasks.csv and displays a list of tasks specific to the logged-in user.

- Add New Personal Task

  Prompts the user to enter task details (title, description, priority, due date, etc.).

  Stores the new task in personal_tasks.csv.

- Update Task

  Lists existing tasks and prompts the user to select one for updates.

Allows the user to modify task attributes like priority, status, or due date.

Saves the updated task back to personal_tasks.csv.

- Delete Task

  Displays a list of personal tasks and allows the user to select one for deletion.

  Removes the selected task from personal_tasks.csv.

- Mark Task as Completed

  Enables the user to mark tasks as completed by updating their status.

  Saves the status update to personal_tasks.csv.

- Add Subtasks to a Task

  Lets the user add subtasks to an existing task.

  Stores subtasks in subtasks.csv, linking them to the main task.

- Filter Personal Tasks:
  Filter tasks based on priority, status, or due date.

- Sort Tasks:

  Sort tasks by priority or due date.

4. Admin User Functionalities :-

- View All Work Tasks

  Reads work_tasks.csv to display all work-related tasks managed by the admin.

- Add New Work Task

  Prompts the admin to enter details for a new work task (title, description, priority, due date, etc.).

  Saves the new task to work_tasks.csv.

- Sort Tasks by Priority or Due Date

  Allows the admin to choose a sorting criterion (priority or due date).

  Reads work_tasks.csv, sorts the tasks, and displays the sorted list.

- Filter Tasks (by Assignee or Status)

  Offers filter options, such as filtering by assignee or status.

  Reads work_tasks.csv, applies the selected filter, and displays the filtered tasks.

- Manage Groups

  Admin can view, add, or delete groups within the system.

  Updates group details in groups.csv.

- Delete Task

  Displays all work tasks, allowing the admin to select a task for deletion.

  Removes the selected task from work_tasks.csv.

5. Saving Changes and Exiting :

- Saving Data

  After any update (add, delete, modify), the system writes changes to the respective CSV file.

  This ensures data persistence between sessions.

- Logout or Exit

  The user can choose to log out or exit the application.

  Logging out returns to the login screen, allowing another user to log in.

  Exiting terminates the program.

6. Error Handling and Validation :

- Input Validation: For all user inputs (dates, priority, etc.), the system checks for valid formats and values.
- File Handling Errors: If a CSV file is missing or corrupt, the program displays an error message and prompts for corrective action.

* **Output Screenshots :**

Test Case 1 :

```
Welcome to the Task Management System!
****************************************

Are you an existing user? (y/n): n
Enter your email: xyz@gmail.com
Enter your username: XYZ
Enter your password: xyz@123
Confirm your password: xyz@123
Enter user type (Admin/Regular): Regular
Sign-up successful! Please log in to continue.
Enter your email: xyz@gmail.com
Enter your password: xyz@123
Login successful!
Welcome, XYZ! Select your mode:
1. Personal Use
2. Work Use
Enter your choice (1 for Personal Use, 2 for Work Use): 1
```

```
PERSONAL USE MODE

Personal Task Management Options:
1. View Tasks
2. Add Task
3. Update Task
4. Delete Task
5. Sort Tasks
6. Filter Tasks
7. Export Tasks
8. Exit
Choose an option (1-8): 2
Add Task selected.
Enter task type (1 for Deadline Task, 2 for Recurring Task):
2
```

```
Enter Task Title:
Os observation
Enter Description:
Upload in lms
Enter Tag:
College
Enter Priority (High/Medium/Low):
High
Enter Start Date (yyyy-MM-dd):
2024-11-04
Enter End Date (yyyy-MM-dd):
2024-11-05
Enter Frequency (daily/weekly/monthly):
Weekly
Does this task have subtasks? (y/n):
y
How many subtasks do you want to add?
1
```

```
Enter Subtask Title:
Finsh code
Enter Subtask Priority (High/Medium/Low):
High
Enter Start Date (yyyy-MM-dd):
2024-04-11
Enter End Date (yyyy-MM-dd):
2024-04-11
Task added successfully!
Choose an option (1-8): 1
View Tasks selected.
```

```
Task ID    Title              Description         Status     Priority   Tag        Start Date      Due Date        Recurring  Frequency
-------------------------------------------------------------------------------------------------------------------------------------
8          Os observation     Upload in lms       Pending    High       College    2024-11-04      2024-11-05      Yes        Weekly
  Subtask :  Finsh code                            Pending    High                  2024-04-11      2024-04-11
Choose an option (1-8): 8

Have a nice day !! :)
Byeee ...
```

Test case 2 :

```
Welcome to the Task Management System!
****************************************

Are you an existing user? (y/n): y
Enter your email: harini@gmail.com
Enter your password: harini@123
Login successful!
Welcome, Harini! Select your mode:
1. Personal Use
2. Work Use
Enter your choice (1 for Personal Use, 2 for Work Use): 1

PERSONAL USE MODE
```

```
Personal Task Management Options:
1. View Tasks
2. Add Task
3. Update Task
4. Delete Task
5. Sort Tasks
6. Filter Tasks
7. Export Tasks
8. Exit
Choose an option (1-8): 1
View Tasks selected.
```

```
Choose an option (1-8): 1
View Tasks selected.
Task ID   Title          Description    Status    Priority   Tag      Start Date   Due Date     Recurring   Frequency
-------------------------------------------------------------------------------------------------------------------
3         Java           assi           Pending   High       study    2024-11-12   2024-11-23   No
4         os             oosss          Pending   Medium     clg      2024-12-13   2024-12-24   Yes         weekly
5         data scienec   do ass         Pending   Medium     study    2024-12-23   2024-12-26   No
6         project        task manager   Pending   High       clg      2024-10-06   2024-11-04   No
7         meditation     do daily       Pending   High       self     2024-11-04   2024-12-31   Yes         daily
```

```
Choose an option (1-8): 3
Update Task selected.
Enter Task ID to update: 3
What would you like to update?
1. Status
2. Due Date
3. Start Date
4. Priority
Choose an option (1-4): 4
Enter new priority: Low
Task updated successfully!
```

```
Choose an option (1-8): 1
View Tasks selected.
```

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
3	Java	assi	Pending	Low	study	2024-11-12	2024-11-23	No	
4	os	oosss	Pending	Medium	clg	2024-12-13	2024-12-24	Yes	weekly
5	data scienec	do ass	Pending	Medium	study	2024-12-23	2024-12-26	No	
6	project	task manager	Pending	High	clg	2024-10-06	2024-11-04	No	
7	meditation	do daily	Pending	High	self	2024-11-04	2024-12-31	Yes	daily

```
Choose an option (1-8): 3
Update Task selected.
Enter Task ID to update: 4
What would you like to update?
1. Status
2. Due Date
3. Start Date
4. Priority
Choose an option (1-4): 1
Enter new status: Completed
Task updated successfully!
```

```
Choose an option (1-8): 5
Sort Tasks selected.
Sort tasks by:
1. Priority
2. Start Date
3. Due Date
1
Tasks sorted successfully!
```

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
6	project	task manager	Pending	High	clg	2024-10-06	2024-11-04	No	
7	meditation	do daily	Pending	High	self	2024-11-04	2024-12-31	Yes	daily
4	os	oosss	Completed	Medium	clg	2024-12-13	2024-12-24	Yes	weekly
5	data scienec	do ass	Pending	Medium	study	2024-12-23	2024-12-26	No	
3	Java	assi	Pending	Low	study	2024-11-12	2024-11-23	No	

```
Choose an option (1-8): 5
Sort Tasks selected.
Sort tasks by:
1. Priority
2. Start Date
3. Due Date
3
Tasks sorted successfully!
```

```
Tasks sorted successfully!
```

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
6	project	task manager	Pending	High	clg	2024-10-06	2024-11-04	No	
3	Java	assi	Pending	Low	study	2024-11-12	2024-11-23	No	
4	os	oosss	Completed	Medium	clg	2024-12-13	2024-12-24	Yes	weekly
5	data scienec	do ass	Pending	Medium	study	2024-12-23	2024-12-26	No	
7	meditation	do daily	Pending	High	self	2024-11-04	2024-12-31	Yes	daily

```
Choose an option (1-8): 6
Filter Tasks selected.
Enter priority to filter (or leave empty to skip): Medium
Enter due date to filter (yyyy-MM-dd) (or leave empty to skip):
Enter tag to filter (or leave empty to skip):
Enter status to filter (or leave empty to skip):
Filter recurring tasks only? (yes/no):
Filtered Tasks:
```

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
4	os	oosss	Completed	Medium	clg	2024-12-13	2024-12-24	Yes	weekly

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
5	data scienec	do ass	Pending	Medium	study	2024-12-23	2024-12-26	No	

```
Choose an option (1-8): 6
Filter Tasks selected.
Enter priority to filter (or leave empty to skip):
Enter due date to filter (yyyy-MM-dd) (or leave empty to skip):
Enter tag to filter (or leave empty to skip): study
Enter status to filter (or leave empty to skip):
Filter recurring tasks only? (yes/no):
Filtered Tasks:
```

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
3	Java	assi	Pending	Low	study	2024-11-12	2024-11-23	No	

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
5	data scienec	do ass	Pending	Medium	study	2024-12-23	2024-12-26	No	

```
Choose an option (1-8): 4
Delete Task selected.
Do you want to delete by ID or by Name? (Enter 'ID' or 'Name')
id
Enter Task ID to delete: 7

Task deleted successfully !!
```

```
Choose an option (1-8): 1
View Tasks selected.
```

Task ID	Title	Description	Status	Priority	Tag	Start Date	Due Date	Recurring	Frequency
6	project	task manager	Pending	High	clg	2024-10-06	2024-11-04	No	
3	Java	assi	Pending	Low	study	2024-11-12	2024-11-23	No	
4	os	oosss	Completed	Medium	clg	2024-12-13	2024-12-24	Yes	weekly
5	data scienec	do ass	Pending	Medium	study	2024-12-23	2024-12-26	No	

```
Choose an option (1-8): 8

Have a nice day !! :)
Byeee ...
```

* **Object Oriented features used :**

1. Classes and Objects

- Classes are blueprints for creating objects, defining attributes and behaviors. Objects are instances of these classes, holding specific data and interacting with each other.
- Classes Used:
    - Task, PersonalTask, WorkTask, DeadlineTask, and RecurringTask represent different types of tasks.
    - User, AdminUser, and RegularUser represent different types of users.
- Objects Created:
    - When the program runs, instances of RegularUser or AdminUser are created based on login credentials. Similarly, tasks are instantiated as PersonalTask or WorkTask objects, each with unique data.

2. Encapsulation

- Encapsulation is used to protect and bundle data within classes. Each class (e.g., Task, PersonalTask, AdminUser, etc.) has private or protected attributes, accessed and modified through public methods (getters and setters).
- Example: In the Task class, attributes like title, description, status, etc., are protected meaning they can be accessed only by its subclasses, with public methods provided to access or modify these fields.

3. Inheritance

- Inheritance is used to create a hierarchy of classes that share common properties, reducing code redundancy.
- Example: PersonalTask and WorkTask classes inherit from the Task superclass, reusing its attributes and methods. Similarly, DeadlineTask and RecurringTask inherit from PersonalTask.
- Additionally, RegularUser and AdminUser extend the User class, sharing common user functionality but also adding specific features for each user type.

4. Polymorphism

Polymorphism is an OOP concept that allows methods or collections to work flexibly with different types.In the project ,

1. Method Overloading: The Task class has methods with the same name but different parameters, like setTaskDetails(String title, String description) and setTaskDetails(String title, String description, int priority), allowing different inputs.
2. Using List<Task> for Multiple Task Types: A List<Task> can store any subclass of Task, such as PersonalTask or WorkTask, enabling unified handling of different task types in one collection.

5. Abstraction

- Abstraction is achieved by defining high-level classes that represent general concepts, like User and Task, with subclasses specializing these general concepts.
- Example: Task represents an abstract idea of a task, while PersonalTask and WorkTask add specific details, like whether the task is personal or work-related.

6. Composition

- Composition is used to build complex types by combining objects of other classes.
- Example: The RegularUser class has a List<PersonalTask> attribute, meaning each RegularUser manages their own list of PersonalTask objects, but they are independent of the RegularUser class itself.

7. File Handling :

- Example: Each class, like User or Task, have methods that read or write data to CSV files, users.csv, personal_tasks.csv, work_tasks.csv, groups.csv and subtasks.csv

8. Exceptions

Exception Handling is implemented to manage unexpected conditions and errors , such as:

- Invalid Login Attempts: Throws custom or standard exceptions if user credentials don’t match.
- File Handling Errors: Handles errors during file reading/writing (e.g., if a CSV file is missing).
- Input Validation: Validates task attributes (like dates or priorities), throwing exceptions if values are invalid. This ensures that the program runs smoothly and provides feedback for incorrect actions.

9. Packages

Packages organize related classes, improving structure and reusability:

- main: Contains the TaskManagementSystem.java file, handling user login, input processing, and menu operations. It also manages CSV files: users.csv, personal_tasks.csv, subtasks.csv, groups.csv, and work_tasks.csv.
- users: Contains user-related classes (User, AdminUser, RegularUser) and manages user-specific functionalities like login and task permissions.
- tasks: Contains task-related classes (Task, PersonalTask, WorkTask, DeadlineTask, RecurringTask), defining task attributes, types, and behaviors.

10. Generics

Generics are used in methods such as exportTasks to handle both work and personal tasks in a single, type-safe method, demonstrating the flexibility of generic programming in Java.

11. Collections

The project makes extensive use of Java Collections, especially ArrayList, to manage lists of tasks, users, and group members dynamically. Collections provide a flexible and efficient way to handle variable-sized data, enabling robust management of tasks and users in the system.

* **Inferences :**

  - Enhanced Flexibility through Polymorphism: Using polymorphism, especially with method overloading and List<Task>, enables the system to handle different types of tasks and input variations, making the project adaptable to future requirements or additional task types.

  - Code Reusability with Inheritance: By defining common task and user attributes in superclasses (Task and User), and extending them in subclasses, the project minimizes redundancy. This leads to a cleaner, more modular codebase, which can be easily updated or expanded.

  - User-Specific Functionalities: The separation of AdminUser and RegularUser with distinct access rights demonstrates efficient role-based control. Admin

functionalities (like assigning and managing work tasks) are restricted to admin users, ensuring data integrity and clear user responsibilities.

- Simplified Task Management with CSV Files: Using CSV files for data storage allows for straightforward data access and persistence, which, while simple, offers an efficient way to manage user and task records without needing complex databases.

- Scalability Potential: With organized packages (main, users, and tasks) and encapsulated data, the system is structured to scale. New features, user roles, or task types can be added with minimal restructuring, supporting the project's long-term maintainability.

* **Future enhancements :**

- Database Integration: Moving from CSV files to a database (e.g., MySQL or MongoDB) could improve data management, scalability, and support for larger datasets. This would also allow for more complex queries and data retrieval options.

- Graphical User Interface (GUI): Adding a GUI would make the system more user-friendly and accessible. With a GUI, users could interact with the application through forms, buttons, and visual displays, enhancing the overall experience.

- Notification System: Implementing notifications for upcoming deadlines or high-priority tasks would help users stay on top of their assignments. This could be achieved with email notifications or pop-up alerts.

- User Authentication and Security: Adding a more secure authentication system with hashed passwords, multi-factor authentication, and role-based access control would enhance security, especially for admin accounts.

- Mobile App Version: Developing a mobile version of the application could make it more accessible, enabling users to manage tasks on the go.