TASK MANAGEMENT SYSTEM

TEAM MEMBERS:

Dharssini K ,2033009

Vidhya Varshany J S ,2033038

Overview of the project

- This application mainly focuses on the efficient improvement on the end users productivity and scheduling his/her daily routine by creating new tasks, assign them a title ,progress of the task, due dates and choosing a project for that task
- It has a text-based user interface in Command Line Interface
- Once the user started using the application, he/she should be able to also edit, mark the task as done or remove tasks.

Functional Requirements

- Initialize the task with a task title, due date , status and project
- Display a collection of tasks that can be sorted by date.
- Enabling the actions like add, sort by date, edit ,mark as done and remove tasks
- Displaying the output in the Human readable format
- Additionally Load and save task list to file for future references

Concepts to be used in the Project

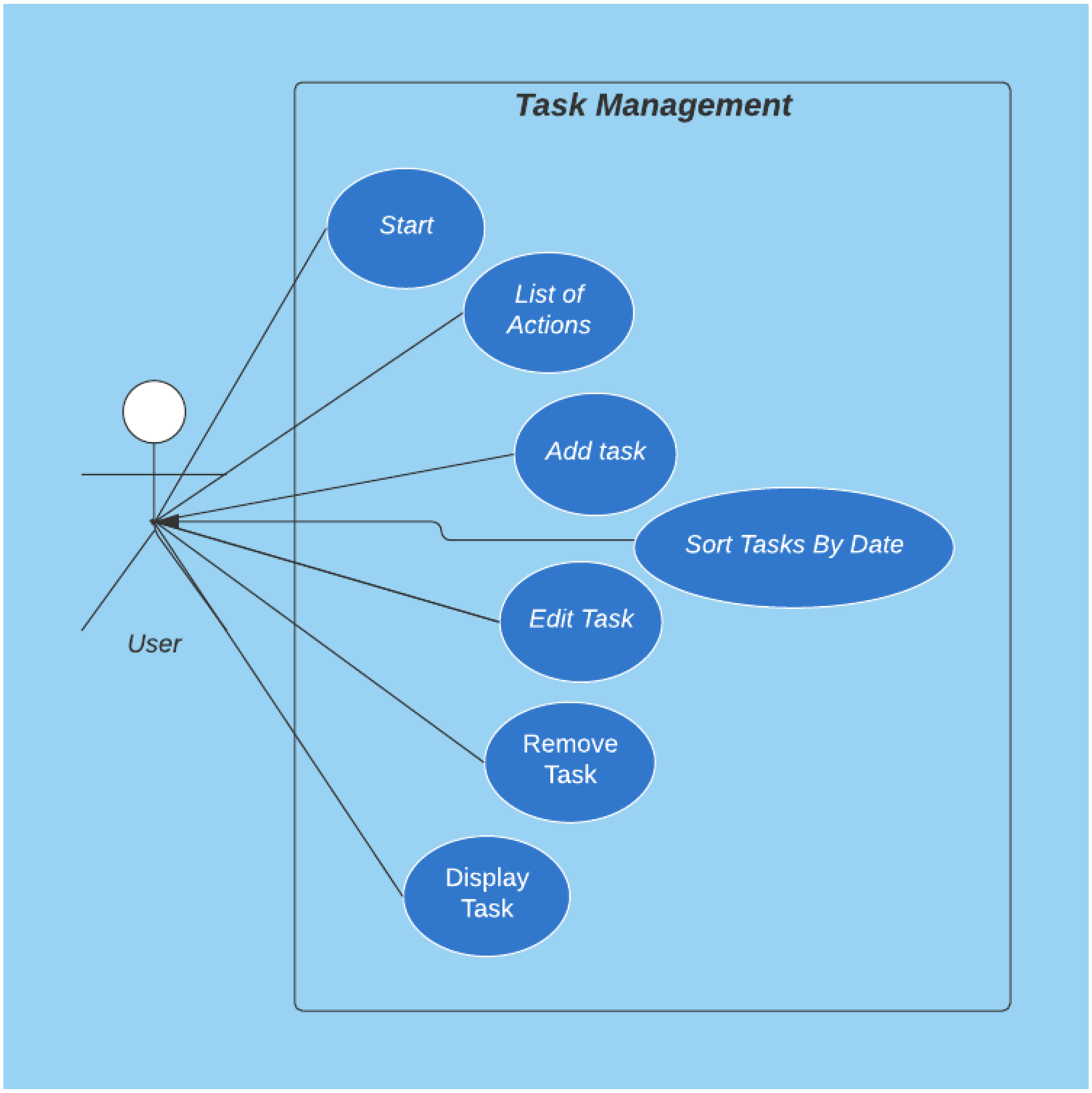
- Use of Data Structures Linked Hash map to store the task by mapping with the id
- Interface,Inheritance , Abstract class

List of classes identified

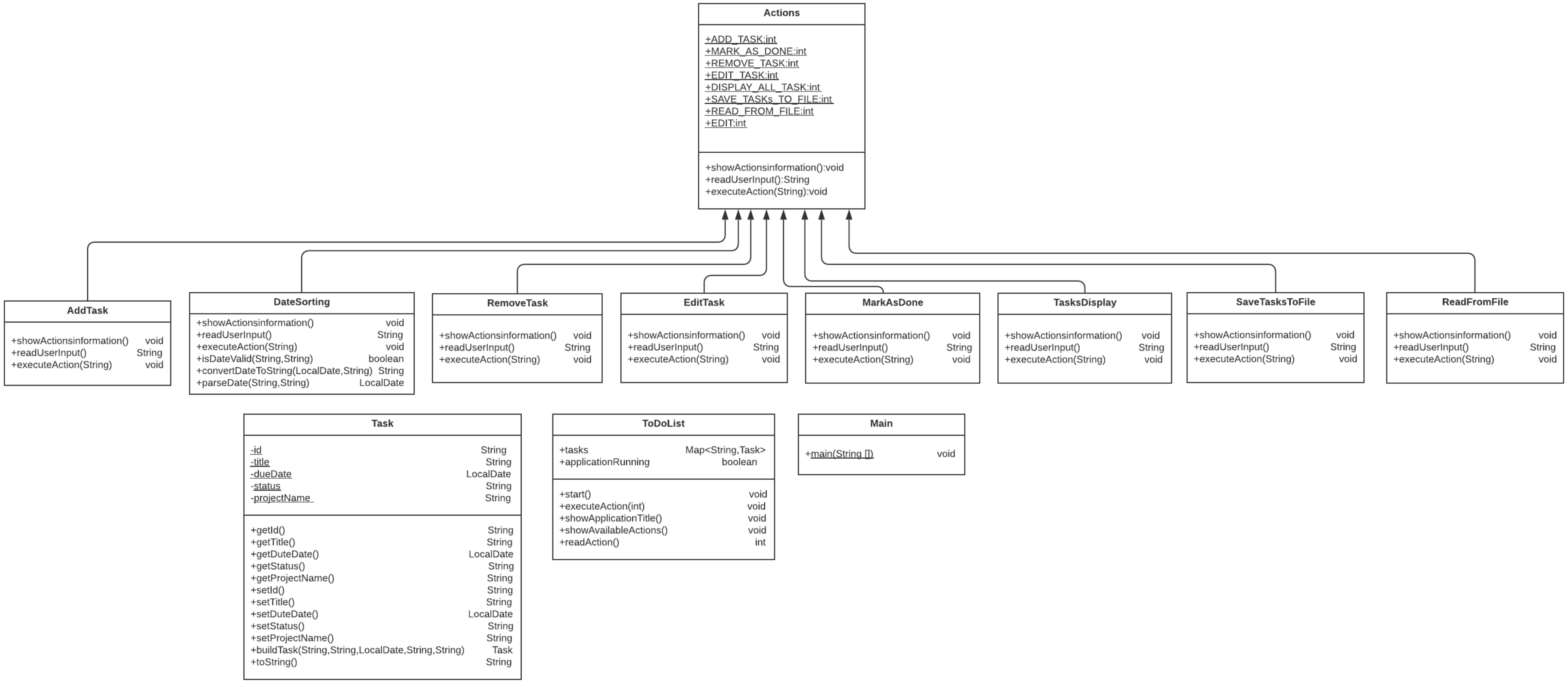
- ❖ **TodoList** This Class is the Parent class which contains all the menu for the dynamic functionality of the other sub classes are acted upon just by calling the specific feature option.
- ❖ **Actions** The Action abstract class used to perform the features like add tasks,marks as done,remove task,edit task, display all task by assigning static final variable and abstract void methods to access among the other classes.
- ❖ **AddTask** This Class is specifically to do the functionality to show Actions Information ,reading user input and execute Action
- ❖ **Sort By Date** This Class enables the user to have the ability to organize tasks by Date,to give better visualization and a better reading
- ❖ **EditTask** This Class will give the user ability to search and identify a certain task using the ID number and edit some or all parts of the task as desired and the update the list itself
- ❖ **MarkAsDone** This Class enables the user to set the task as done once they feel they are satisfied with the task.

- **RemoveTask** The Class enables the feature to delete the current file by choosing the Remove option in the menu. The File will be taken out of the list after the use had searched and identified the specific folder to be deleted
- **TasksDisplay** This Class displays all the tasks, by choosing Display All Tasks option, this will fetch all related data to tasks given in the to-do list
- **SaveTasksTo File** This Class can save any file by choosing the **Save Tasks** option, this will make sure the user doesn't miss any of his/her tasks
- **ReadFromFile** This Class displays all the tasks, by reading or fetching the tasks information form a local existing file.

Use-Case Diagram



Class Diagram



Actions

```
+ADD_TASK:int
+MARK_AS_DONE:int
+REMOVE_TASK:int
+EDIT_TASK:int
+DISPLAY_ALL_TASK:int
+SAVE_TASKs_TO_FILE:int
+READ_FROM_FILE:int
+EDIT:int

+showActionsinformation():void
+readUserInput():String
+executeAction(String):void
```

AddTask

```
+showActionsinformation()     void
+readUserInput()            String
+executeAction(String)        void
```

DateSorting

```
+showActionsinformation()                       void
+readUserInput()                              String
+executeAction(String)                          void
+isDateValid(String,String)                  boolean
+convertDateToString(LocalDate,String)  String
+parseDate(String,String)                  LocalDate
```

RemoveTask

```
+showActionsinformation()     void
+readUserInput()            String
+executeAction(String)        void
```

EditTask

```
+showActionsinformation()     void
+readUserInput()            String
+executeAction(String)        void
```

MarkAsDone

```
+showActionsinformation()       void
+readUserInput()              String
+executeAction(String)          void
```

TasksDisplay

```
+showActionsinformation()       void
+readUserInput()              String
+executeAction(String)          void
```

SaveTasksToFile

```
+showActionsinformation()         void
+readUserInput()                String
+executeAction(String)            void
```

ReadFromFile

```
+showActionsinformation()         void
+readUserInput()                String
+executeAction(String)            void
```

Task

```
-id                                              String
-title                                           String
-dueDate                                      LocalDate
-status                                          String
-projectName                                     String

+getId()                                         String
+getTitle()                                      String
+getDuteDate()                                LocalDate
+getStatus()                                     String
+getProjectName()                                String
+setId()                                         String
+setTitle()                                      String
+setDuteDate()                                LocalDate
+setStatus()                                     String
+setProjectName()                                String
+buildTask(String,String,LocalDate,String,String)  Task
+toString()                                      String
```

ToDoList

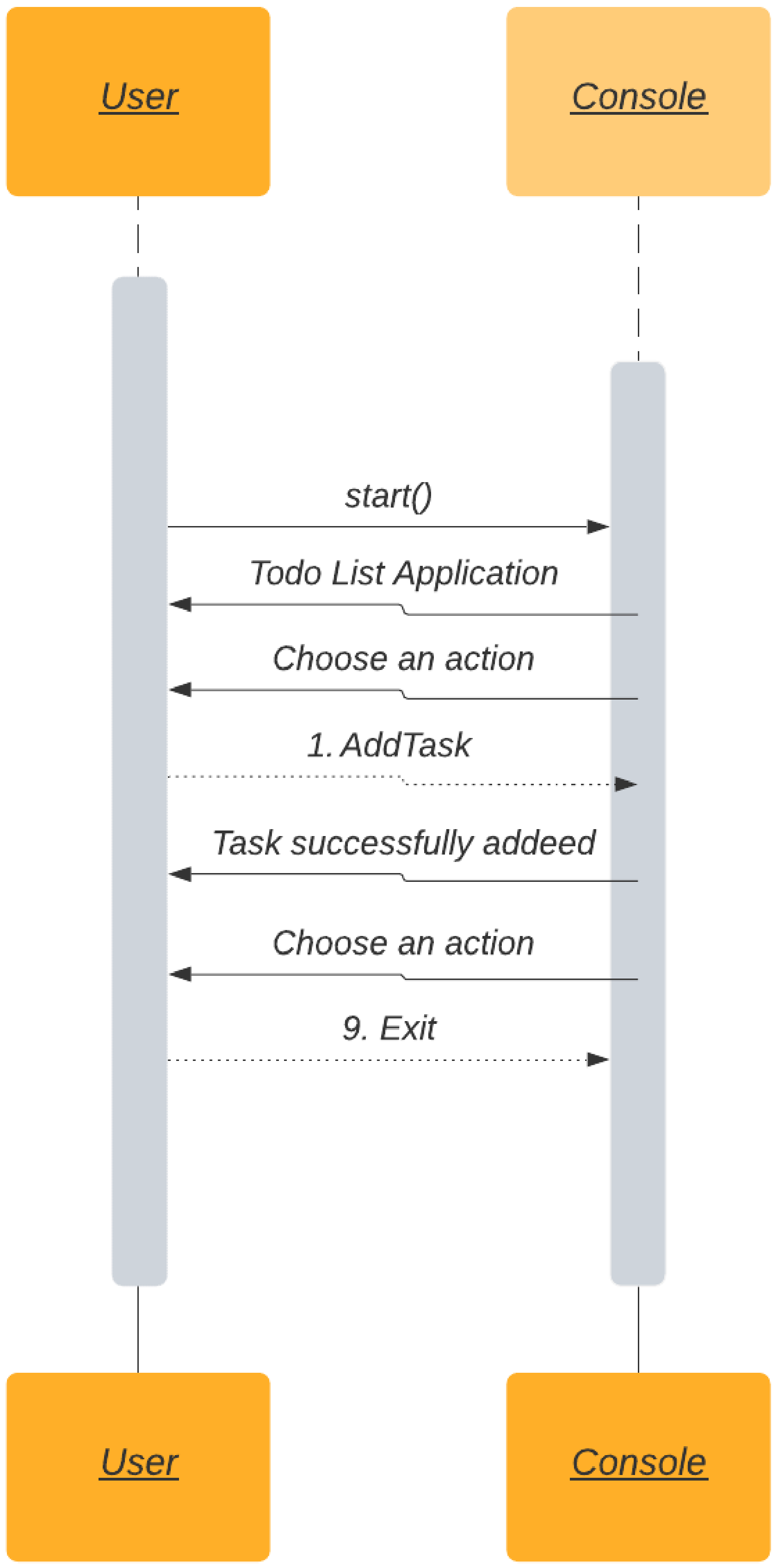
```
+tasks                      Map<String,Task>
+applicationRunning                  boolean

+start()                                void
+executeAction(int)                     void
+showApplicationTitle()                 void
+showAvailableActions()                 void
+readAction()                            int
```

Main

```
+main(String [])            void
```

Sequence Diagram



List of Actions

1.Add a task
2.Sort By Date
3.Remove a task
4.Mark as done
5.Edit a task
6.Display all tasks
7.Save Tasks to file
8.Read from File
9.Exit