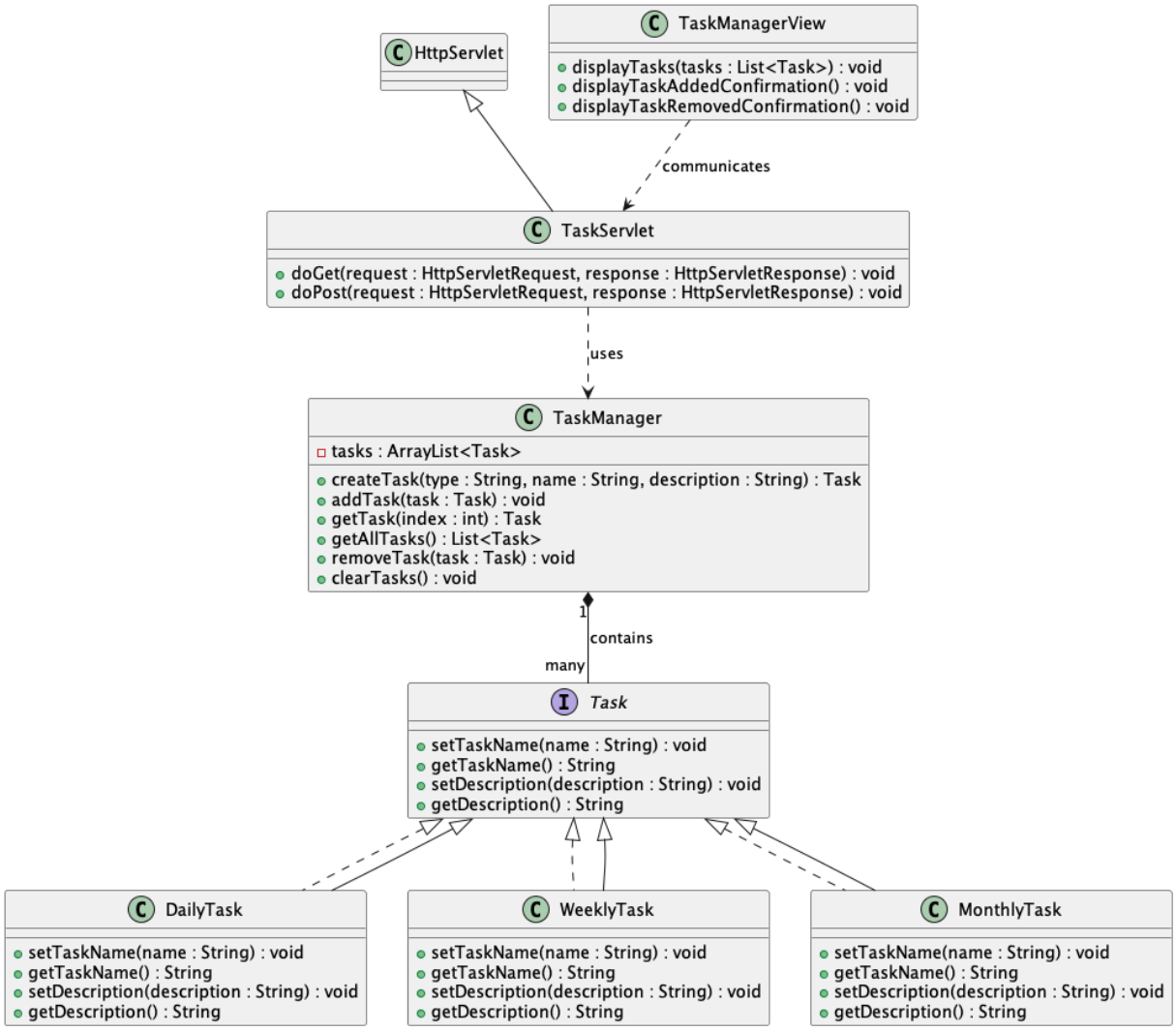
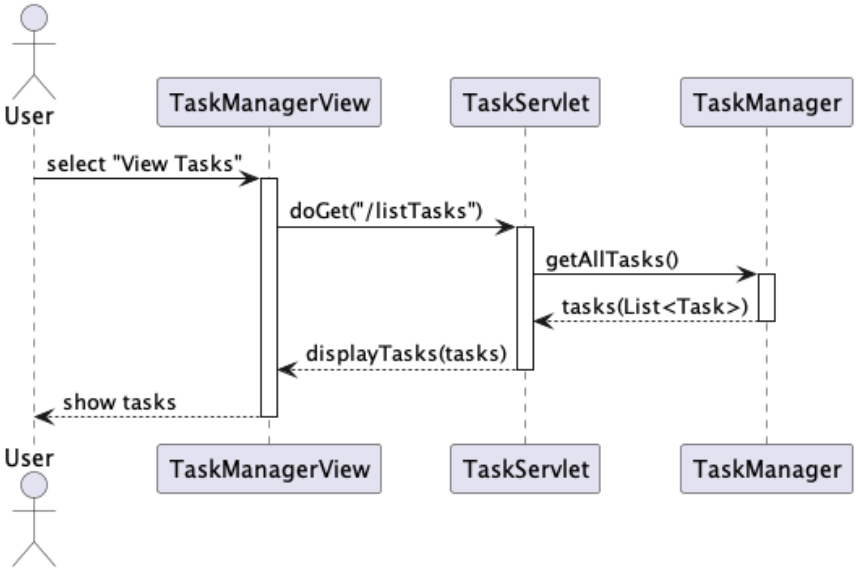
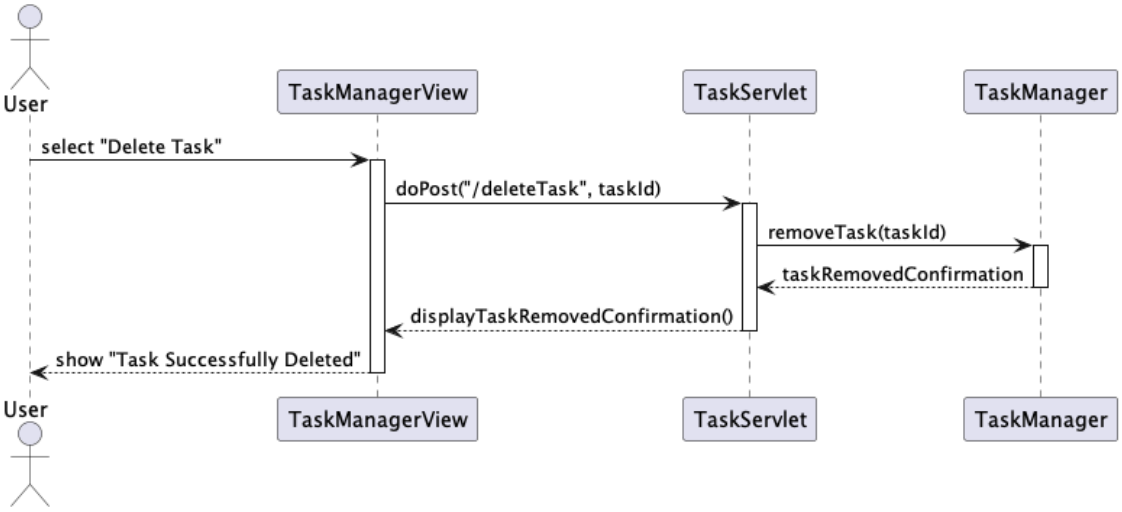

HttpServlet
TaskManagerView
displayTasks(tasks : List<Task>) : void
displayTaskAddedConfirmation() : void
displayTaskRemovedConfirmation() : void
communicates
TaskServlet
doGet(request : HttpServletRequest, response : HttpServletResponse) : void
doPost(request : HttpServletRequest, response : HttpServletResponse) : void
uses
TaskManager
tasks : ArrayList<Task>
createTask(type : String, name : String, description : String) : Task
addTask(task : Task) : void
getTask(index : int) : Task
getAllTasks() : List<Task>
removeTask(task : Task) : void
clearTasks() : void
1
contains
many
Task
setTaskName(name : String) : void
getTaskName() : String
setDescription(description : String) : void
getDescription() : String
DailyTask
setTaskName(name : String) : void
getTaskName() : String
setDescription(description : String) : void
getDescription() : String
WeeklyTask
setTaskName(name : String) : void
getTaskName() : String
setDescription(description : String) : void
getDescription() : String
MonthlyTask
setTaskName(name : String) : void
getTaskName() : String
setDescription(description : String) : void
getDescription() : String

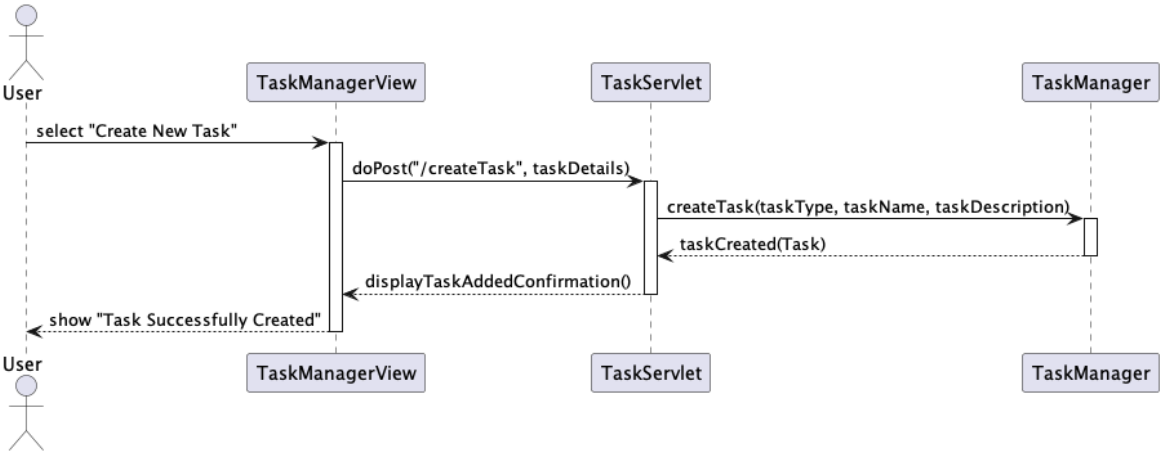
View Task:



Delete Task:



Create Task:



Our Task Tracker Web Application, developed using Java Servlets, leverages the principles of the Model-View-Controller (MVC) architecture, ensuring a structured and efficient user interface and backend interaction.

Central to the application is the Task interface, which defines the essential characteristics of a task. This interface is implemented by three classes: DailyTask, WeeklyTask, and MonthlyTask. This design demonstrates the application of inheritance and allows for flexibility in managing different task types.

The TaskManager class manages these tasks, providing functionalities to add, view, and delete tasks, stored in an ArrayList. It interfaces with the TaskServlet, our gateway for handling HTTP requests and directing them appropriately.

The front end, built with HTML and CSS, is managed by TaskManagerView. It processes user inputs and displays task information, interfacing smoothly with the TaskServlet for data exchange.

Our UML diagrams accurately represent the structure and flow of the application. The class diagram details our design, showing relationships and dependencies clearly. The sequence diagrams for each core functionality (adding, viewing, and deleting tasks) lay out the operational flow, allowing us to make sure that the design is not only theoretical but practically applicable.