Project: TaskHub

Overview

TaskHub is a desktop project management application used by project managers to manage projects and their team members in each project. The user interacts with it using a CLI, and it has a GUI created with JavaFX. It is written in Java.

Summary of Contributions

Code Contributed:

My code contributions to TaskHub can be found here.

Enhancements implemented:

- **Upgraded Help Window:**
  - Modified `Ui` for `HelpWindow.java`.
  - Displays a quick guide of all commands.
  - Provides a clickable hyperlink to the user guide at the bottom, instead of the original 'copy to clipboard' feature.
  - Added value to the `help` command.
- **`Task` Model, `addT`, and `deleteT` Command:**
  - Created infrastructure for tasks in both the `model` and `Ui` files - the Ui can be seen in the 'Tasks' section of the `Project`s in the screenshot above.
  - Implemented basic functionality for adding and deleting tasks.
  - Efficiently manages multiple task deletions.
  - Enhanced `addT` to allow task assignment to an employee upon creation using the optional `em/` parameter.

Contributions to the UG:

- **Introductory section**
  - Getting Started
    - Installing and launching TaskHub
    - Understanding the components of TaskHub
    - Quick Start
- **Features section**
  - Explanation for `addT` command.
  - Explanation for `deleteT` command.

Contributions to the DG:

- **Architecture, UI Component, Logic component, Model Component, Employee, Project, Task components**

    - Addition of tasks to the Class Diagram for Ui component
    - Minor updates and diagram changes to other components (e.g., removing attributes section, updating command names/links)
- **Implementation Section**
    - Upgraded Help Feature
    - Add Task feature (created sequence diagram for this)

Contributions to team-based tasks:

- Set up the GitHub team org/repo
- Updated diagrams in DG related to Tasks
- Created tags to help with bug triaging post PE-D
- Generated table for PR review allocations here:

Review/mentoring contributions:

- Within the team, we had decided that each of us were to review 2 of our teammates. I mainly reviewed Anton and Aslam's PRs.
- Helped Anton with centering the TaskHub logo in this PR , by providing a code block which gave advice on handling positioning of components on Java FXML.
- Gave Aslam code quality suggestions in this PR so that we could sync up our implementation of deadlines in both `Tasks` and `Projects`.

Contributions beyond the project team:

Evidence of helping others e.g. responses you posted in our forum, bugs you reported in other team's products,

- Reported an above average bug count of 10 with value-added suggestions to my allocated PE-D team, can be seen from the PE-D repo.

Evidence of technical leadership e.g. sharing useful information in the forum

- Participated in the forum in this issue.

UG Extracts

Understanding the components of TaskHub

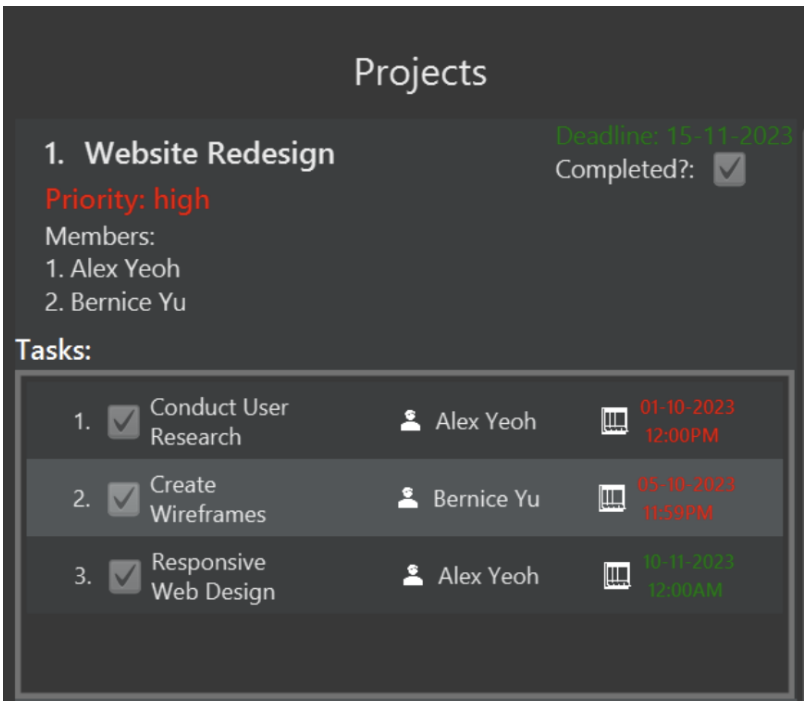
Employees

An `Employee` is someone that you are managing. TaskHub allows you to store their essential details and `tag` them with their strengths and weaknesses so you can allocate them to suitable `Projects` or `Tasks`.

Attributes:

Field	Description	Prefix for addE
Name	Name of the employee.	`n/`
Phone Number	Phone number of the employee.	`p/`
Email	Email address of the employee.	`e/`
Address	Address of the employee.	`a/`
Tags	Tags indicating strengths/weaknesses/position of the employee.	`t/`

Projects

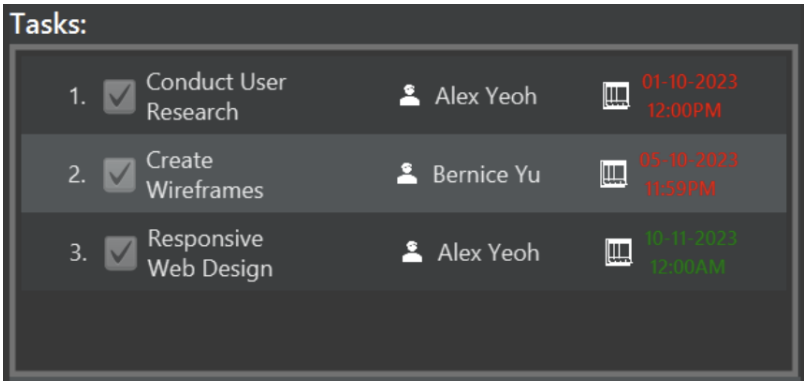


A `Project` in TaskHub is a managerial unit that includes information about `Employees` allocated to the project and an (optionally) set deadline. You can mark a `Project` as done when you deem fit. Additionally, you can add `Tasks` to a `Project`.

Attributes:

Field	Description	Prefix for `addP`	Relevant Command(s)
Name	Name of the project.	`n/`	-
Employees	Employees assigned to the project.	`em/`	`assignP`
Deadline	Deadline for the project.	-	`dlP`
Priority	Priority level of the project.	-	`priorityP`
CompletionStatus	Indicates whether the project is completed or in progress.	-	`markP`, `unmarkP`
Tasks	Tasks associated with the project.	-	`addT`

Tasks



A `Task` in TaskHub represents a specific job within a `Project` that can be assigned to an `Employee` under that `Project`. Tasks are required to have a deadline. Managing `Task`s will be the main way of monitoring the work done within your `TaskHub`!

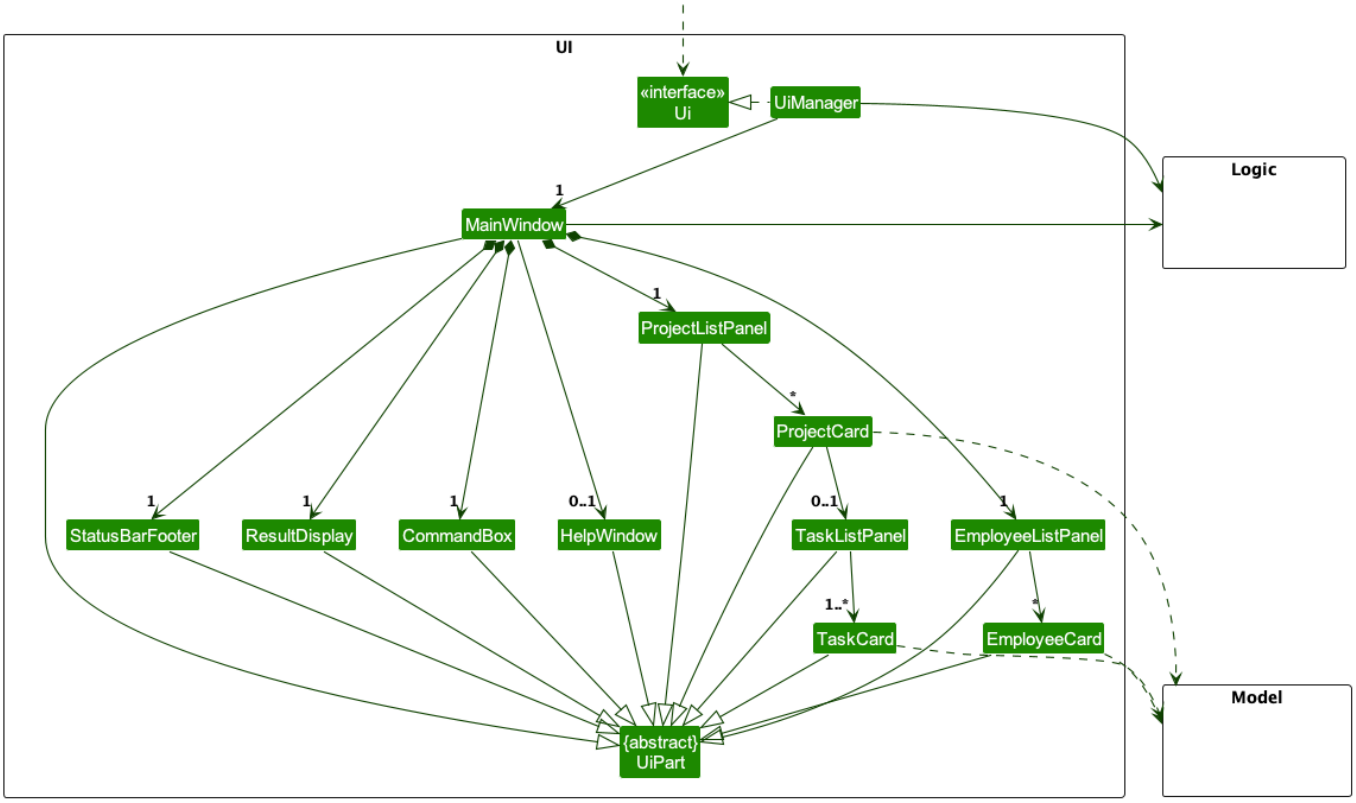
Attributes:

Field	Description	Prefix for `addT`	Relevant Command
Name	Name of the task.	`n/`	-
Employee	Employee assigned to the task.	`em/`	`assignT`
Deadline	Deadline for completing the task.	`d/`	-
isDone	Indicates whether the task is complete.	N.A.	`markT`, `unmarkT`

DG Extracts

UI component

The **API** of this component is specified in `Ui.java`



The UI consists of a `MainWindow` that is made up of parts e.g.`CommandBox`, `ResultDisplay`, `EmployeeListPanel`, `ProjectListPanel`, `StatusBarFooter` etc. All these, including the `MainWindow`, inherit from the abstract `UiPart` class which captures the commonalities between classes that represent parts of the visible GUI.

The `UI` component uses the JavaFx UI framework. The layout of these UI parts are defined in matching `.fxml` files that are in the `src/main/resources/view` folder. For example, the layout of the `MainWindow` is specified in `MainWindow.fxml`

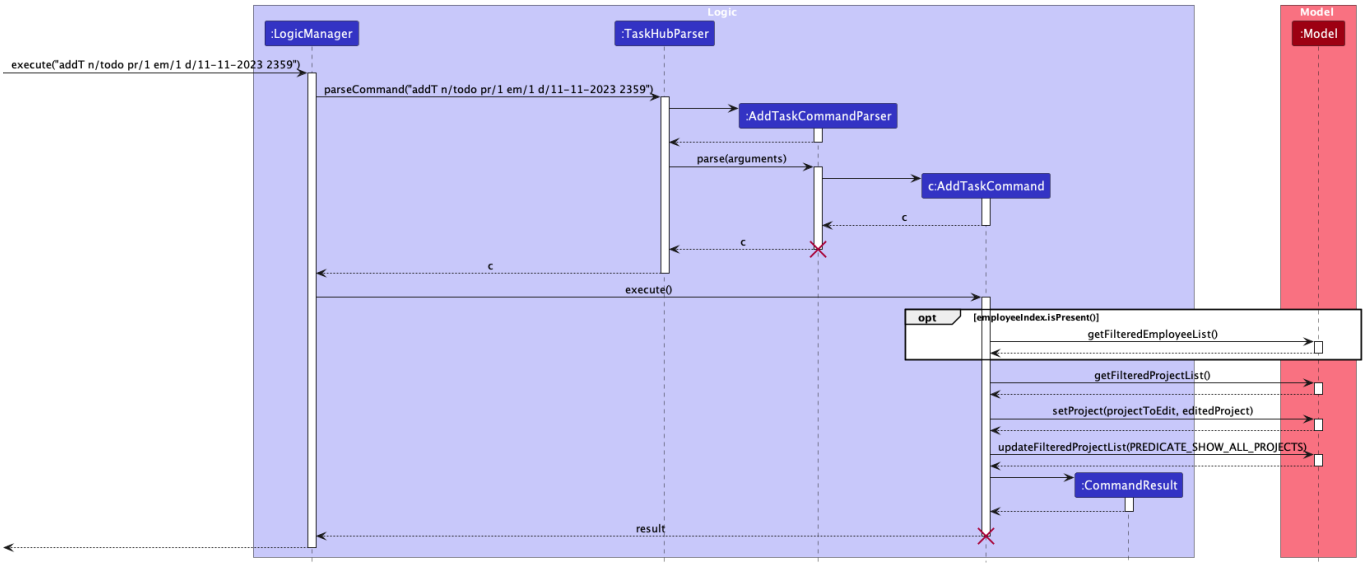
The diagram above briefly demonstrates the hierarchy of the Ui components. A few more notes are as follows:

- At the level just below `MainWindow`, there are the main components, like `CommandBox`, `ResultDisplay`, `EmployeeListPanel` and `ProjectListPanel`.
- `EmployeeListPanel` contains some number of `EmployeeCard`s.
- `ProjectListPanel` contains some number of `ProjectCard`s
    - which contain a `TaskListPanel` with their `TaskCard`s if the relevant `Project` in the `Model` contains a `Task`.

The `UI` component,

- executes user commands using the `Logic` component.
- listens for changes to `Model` data so that the UI can be updated with the modified data.
- keeps a reference to the `Logic` component, because the `UI` relies on the `Logic` to execute commands.
- depends on some classes in the `Model` component, as it displays `Employee` object residing in the `Model`.

Add Task feature



When creating a new task using the `addT` command, the `TaskList` of the specified `Project` is updated, and the `Project` is hence updated too.

Given below is an example usage scenario and the internal changes that happen at each step.

Step 1. The user launches the application. All employees and projects will be shown to the user.

Step 2. The user executes `addT n/todo pr/1 em/1 d/11-11-2023 2359` to add a new `Task` called `todo` to the first currently listed `Project`, assigned to the first `Employee` within that `Project`. `LogicManager` will call `TaskHubParser#parse(input)` to extract the parameters and pass it to an `AddTaskCommandParser`.

Step 3. `TaskHubParser` will call `AddTaskCommandParser#parse(arguments)` to produce a `AddTaskCommand` to be executed by the `LogicManager`.

Step 4. `LogicManager` calls `AddTaskCommand#execute(model)` to produce a `CommandResult` to be logged.

Step 5. During the execution of the `AddTaskCommand`, a new `Project` copy is created, with an updated `TaskList` that contains the newly created `Task`. If an `employeeIndex` was specified by the command (in this case it was), then `Model::getFilteredEmployeeList` is called to assign the new `Task` to the specified `Employee`. Then, the `Model#setProject` and `Model#updateFilteredProjectList` is called, to trigger a `Ui` update, as the specified `Project` has been updated with an updated `TaskList`.

Step 6. A `CommandResult` is produced based on whether the execution was a success or not and returned to the `LogicManager`.
