Ravi Arun Kumarr - Project Portfolio for +Work

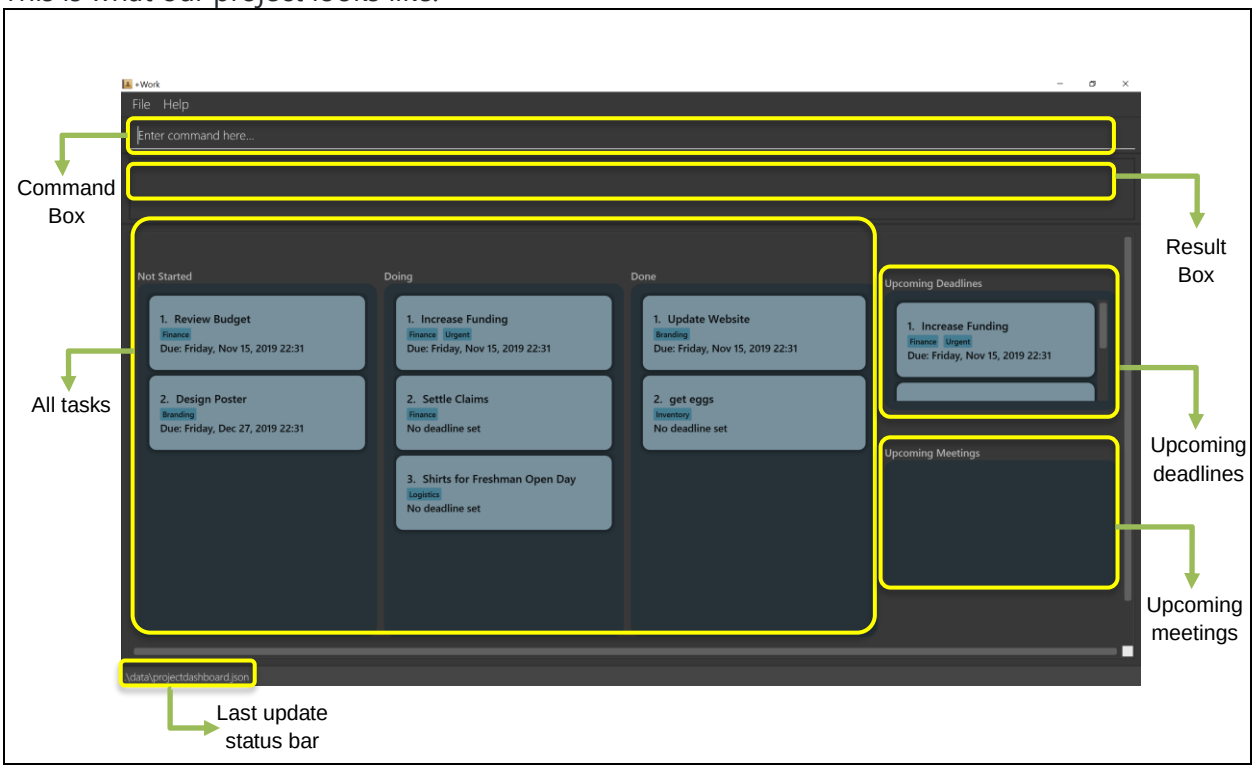
Introduction

This project portfolio details my contributions for a Software Engineering group project taken in National University of Singapore. My team of 4 software engineering students and I built a command line interface (CLI) desktop application, +Work targeted at NUS student leaders.

About the project

We were tasked with enhancing a basic command line interface desktop addressbook application for our Software Engineering project. We chose to morph it into a project management tool called +Work. This enhanced application enables NUS student project leaders to manage team members, meeting times, task allocations as well as equipment purchased.

This is what our project looks like:



My role was to design and write code for the inventory and multiline features. The following sections illustrate these enhancements in more detail, as well as the relevant documentation I have added to the user and developer guides in relation to these enhancements.

Note the following symbols and formatting used in this document:

This symbol indicates important information.

add-inv	A grey highlight (called a mark-up) indicates that this is a command that can be inputted into the command line and executed by the application.
[i/NAME] (p/PRICE)	Command attributes in square brackets,'[ ]' mean it is required. Meanwhile, attributes in round brackets, '( )' mean it is optional.
	Regions that are surrounded by a rounded box similar to this indicates that the instructions above the picture is regarding the information inside the box.
	A dashed line at the end of the picture indicates that the picture is cut off in the middle, but it still displays the necessary information.

Summary of contributions

This section shows a summary of my coding, documentation, and other helpful contributions to the team project.

Enhancement added: I added the ability to manage inventories.

• What it does: Commands to manage inventory allow user to add, delete, edit and print PDF reports of inventories.

• Justification: +Work is designed for university-level students' projects which usually require inventories. Rather than keeping a separate log of inventories in an excel spreadsheet, +Work allows them to manage it together with project tasks at the same place for extra convenience.

• Highlights: An in-depth analysis of design alternatives was necessary to design the PDF-report feature of inventory management. The decision to create reports in a PDF format rather than other formats was made to increase efficiency and to simplify programming.

• Credits: Code from AddressBook-3 provided was adapted to create the inventory features. iText library was used to create PDF reports for inventory.

Enhancement added: I added the ability to handle multiline commands.

• What it does: Multiline commands ask users questions based on their input, allowing them to add extra attributes.

• Justification: If an user wants to add a task and set deadline or member to it or wants to save a task as inventory after it is done, he/ she has to do it in multiple steps. Multiline commands help to remove this issue by interacting with the user and adding whatever is necessary.

• Highlights: Multiline commands will work with existing as well as future commands. The implementation was challenging since the original code was not designed to ask for user inputs based on previous user inputs. In addition to solving this issue, the multiline features were also created without causing much disruption to the existing code.

Code contributed: Please click these links to see a sample of my code: [Functional code] [Test code]

Other contributions:
• Project Management:
  - Added additional tests to the repository to bump coverage from 40% to 42% (Pull Requests #170)
• Enhancements to existing features:
  - Updated the storage to accommodate inventory and members(Pull requests #29, #61)
• Documentation:
  - Made improvements to the existing User Guide and Developer Guide to make it more reader friendly (Pull Request #161, #165)
• Community:
  - Reviewed Pull Requests (with non-trivial review comments): #12, #17, #63
• Tools:
  - Integrated a third party library (iText) to the project (Pull Request #71)

Contributions to User Guide

We had to update the original addressbook User Guide with instructions for the enhancements that we had added. The following is an excerpt from our +Work User Guide, showing additions that I have made for the inventory and multiline features.

Inventory Commands

Adding an inventory: add-inv

This command allows you to add an inventory bought or retrieved for a specific task by a specific member.

Format: `add-inv [i/ITEM_NAME] (p/PRICE) [ti/TASK_ID] [mi/MEMBER_ID]`

The item name, task id and member id are compulsory inputs. An input without any price value will automatically set the price to $0.

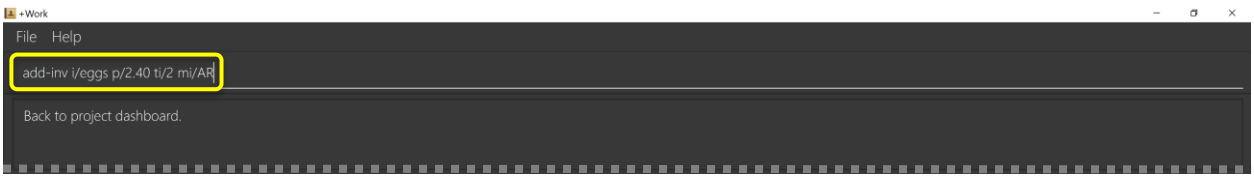
Ensure that an existing task id and member id (as displayed by `list-tasks` and `list-members` respectively) is being typed into the command box.

Examples:

- `add-inv i/scissors ti/4 mi/GS`
  Entering this command adds the inventory "scissors" to the inventory list. The item is tagged to task with id as 4 (id is shown by `list-tasks`) and is provided by member with the member id "GS" for a price of $0.

- `add-inv i/eggs p/2.40 ti/2 mi/AR`
  Entering this command adds the item "eggs" for $2.40 to the inventory list. This item is tagged to task tagged to task with id as 2 (id is shown by `list-tasks`) and was paid for by the member with member id "AR". The following pictures show how this command is to be executed.

  - First, type `add-inv i/eggs p/2.40 ti/2 mi/AR` into the command box as shown below.



  - Then, hit the `Enter` key and you will see that the inventory has been added! You should see a window like the one below.

Creating a report of inventories: pdf

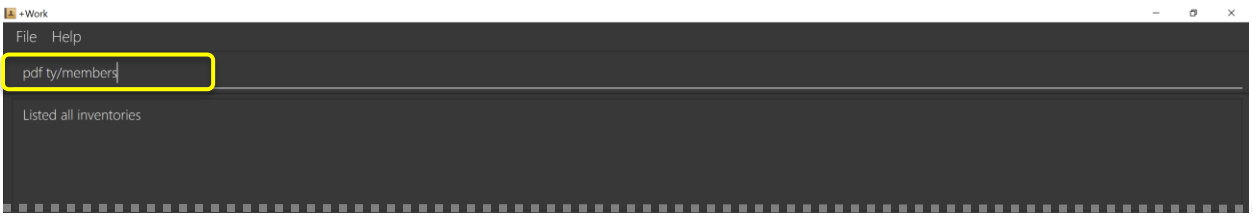
This command allows you to create a PDF report of the existing inventories classified either by the member attached or by the task attached.

Format: pdf [ty/TYPE]

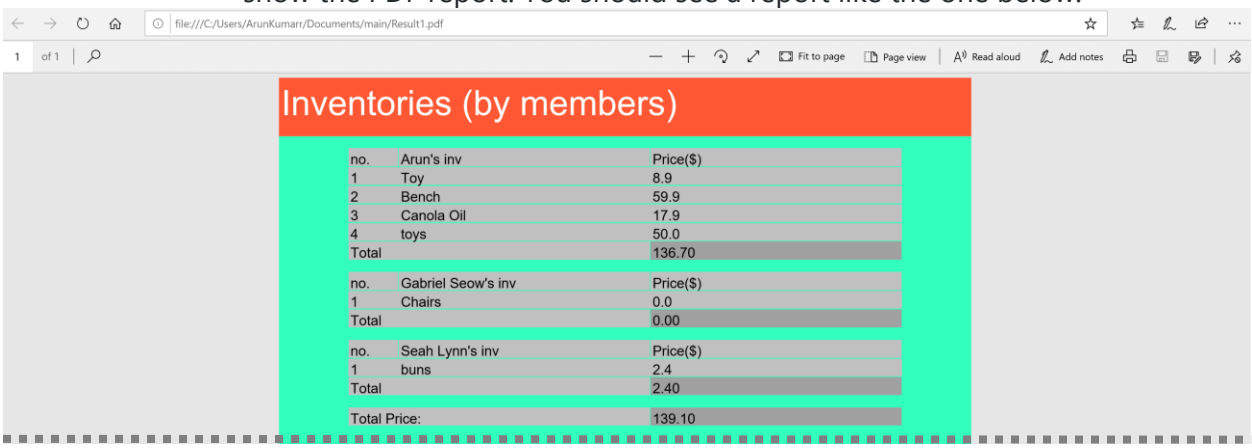
The only two attributes for TYPE are members and tasks. Other inputs will not work.
If a PDF created and is currently open, it has to be closed before another pdf can be created.

Example:

- pdf ty/members
  This command creates and opens a pdf file of inventories that is classified according to the member attached. The following pictures show how this command is to be executed.
    - First, type pdf ty/members into the command box as shown below.



    - Hit the Enter key and you will see that the default PDF viewer will open to show the PDF report. You should see a report like the one below.

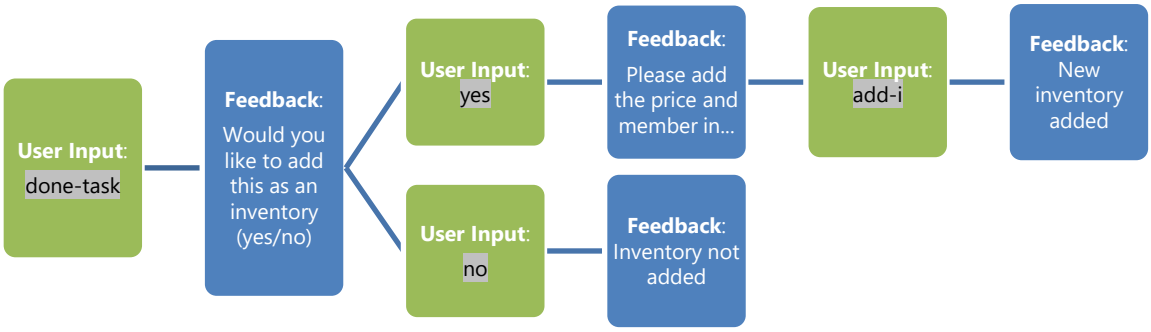


Multiline Commands

Changing a task status to done: done-task (for tasks that are tagged "Inventory")

This feature only applies only for tasks that are tagged "Inventory"

As mentioned in the earlier part of user guide, to change the task status to "done", use the done-task command in the format below. If the task is tagged as "Inventory", this will result in a series of questions that allow you to add the task as an inventory. The following flowchart shows this series of feedbacks from the application and user inputs.



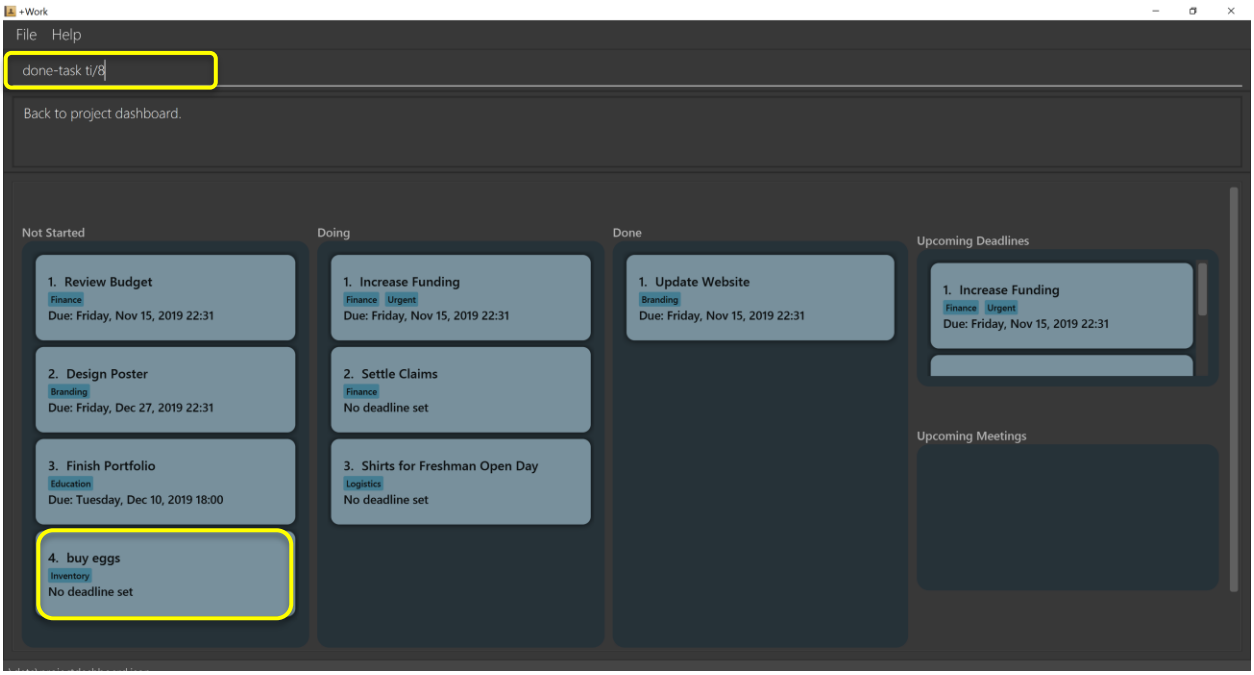
Format (for done task): done-task [ti/TASK_ID]
Format (for yes): yes
Format (for no): no
Format (to add price and member): add-i [p/PRICE] [mi/MEMBER_ID]

add-i command does not work unless it is typed after done-task and yes.

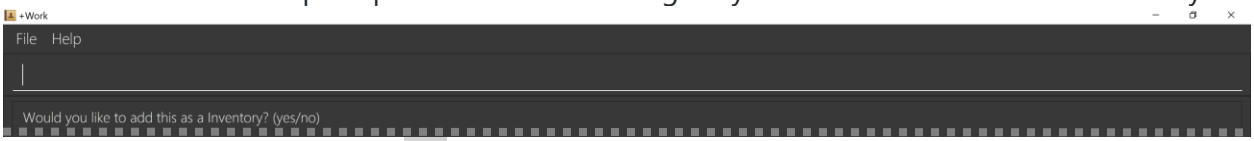
Unlike add-task, in this case both attributes price and member id are required.

Example:

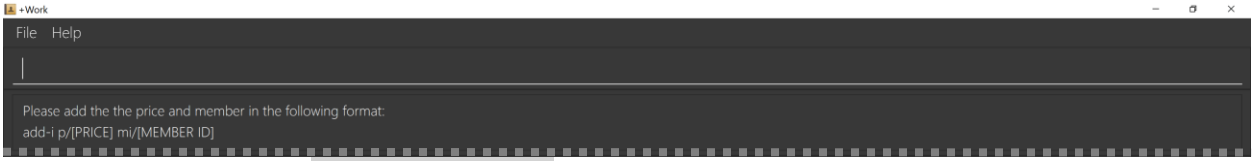
- done-task ti/8
  The task with index 8 will have its task status changed to done. Since the task is tagged as "Inventory", this will prompt a series of questions, as shown in the pictures below.
    - First, type done-task ti/8



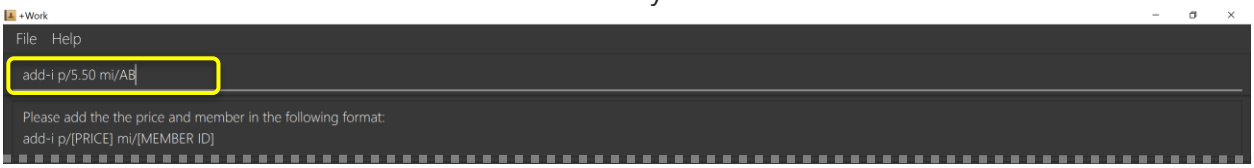
- This prompts the feedback asking for your choice to add it as an inventory.



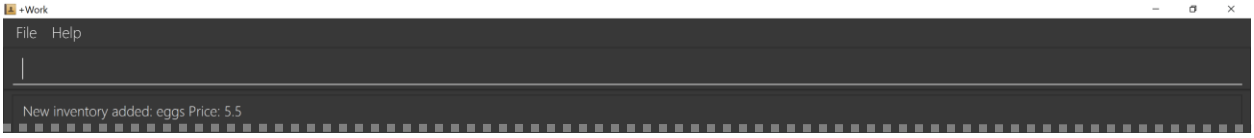
- After typing yes as the user input, feedback will prompt you to add the details.



- Type add-i p/5.50 mi/AB to set the price at $5.50 and to assign the member with member id "AB" to the inventory



- This sets the price and assigns the member successfully! You should see a window like this.



Contributions to Developer Guide

The following section shows my additions to the +Work Developer Guide for inventory feature.

Inventory Feature

Proposed Implementation

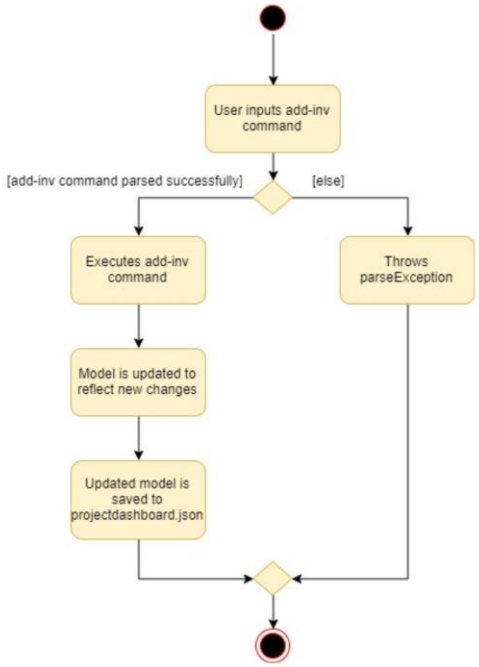
This feature was implemented to allow users to add inventories when using +Work.

The commands introduced by this feature include; `add-inv`, `delete-inv`, `edit-inv`. The commands are facilitated by UniqueInventoryList class which resides in Model. The UniqueInventoryList class implements the following operations:

- `UniqueInventoryList #add(Inventory toAdd)` — This command adds the Inventory toAdd to the inventory list of +Work.
- `UniqueInventoryList #remove(Inventory toAdd)` — This command removes the Inventory toAdd from the inventory list of +Work.
- `UniqueInventoryList #setInventory (Inventory target, Inventory editedInventory)` — This command replaces the target Inventory with the new editedInventory.

These operations are exposed in the Model interface as `Model#addInventory(Inventory inventory)`, `Model#deleteInventory(Inventory target)`, `Model#setInventory(Inventory target, Inventory editedInventory)`

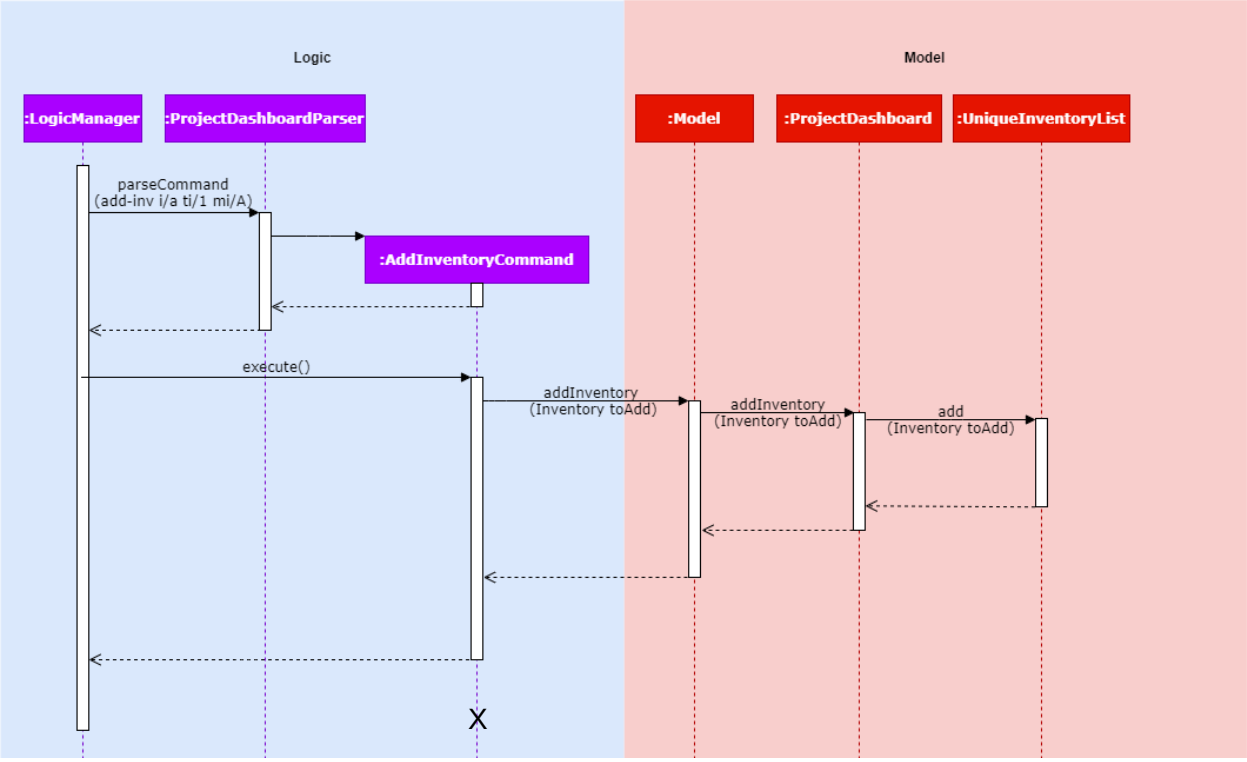
The activity diagram below summarises the process of executing an `add-inv` command.



Given below is an example usage scenario and how add-inv command behaves at each step:

Step 1. The user launches the application. The inventories will be initiated by Model based on the inventories previously saved.
Step 2. The user executes `add-inv` command.

Step 3. Logic#execute() calls Model#addInventory(Inventory inventory), which calls UniqueInventoryList #add(Inventory toAdd). This adds the inventory to UniqueInventoryList.
Step 4. The UI will be updated to reflect the changes. This can be viewed using the list-inv command.
Step 5. The settings have been updated and stored in projectdashboard.json.
The following sequence diagram shows how the add-inv command works.



Design Considerations

This section explores how the design can affect the inventory features in +Work.

Aspect: Storage of an inventory

- **Alternative 1 (current choice):** Each inventory only stores the inventory name and price. The task and member attached to each inventory is stored in UniqueMappingManager.
    - Pros: Better design as it is more modular. If task or member is deleted, it only needs to be changed in UniqueMappingManager for the expected behaviour to be achieved.
    - Cons: Retrieving of the mappings (task and member attached) is more difficult and may result in bugs if not implemented accurately.
- **Alternative 2:** Each inventory also contains the attribute for task attached and member attached.

    - Pros: Retrieving the mappings is easier and faster.
    - Cons: When a task or member is deleted, all the inventories need to be checked and updated. This would be a very slow method.

Aspect: Display of inventories list

- **Alternative 1 (current choice):** The inventories are listed without any classification and is not sorted by any attributes.
    - Pros: This would be easier to implement and maintain UI components.
    - Cons: User must use the pdf method to see any statistics, making for a less user-friendly experience.
- **Alternative 2:** More statistics, classifications and sorting methods available to customize the inventories list.
    - Pros: The inventories list is more user-friendly and more provides more details.
    - Cons: Implementation is harder, and we have to ensure minimal errors or bugs.