

**Developer Guide**

**CS2101/3T**

EFFECTIVE COMMUNICATION FOR COMPUTING PROFESSIONALS  (CS2101) & SOFTWARE ENGINEERING (CS2103T)

**Group POMPOM (F14-3J)**

Ho Wei Lip

Lau Wen Hao

Tan Wee Chen, William

Tay Kai Teng, Jorel

**Date of Submission**

- March 2016

Table of Contents

1. Introduction 3

2. Functionalities 3

3. Setting Up 3

4. Architecture Overview 3

5. Interaction between Components 4

5.1 Adding Task 4

5.2 Editing Task 4

5.3 Deleting Task 5

6. Components 5

6.1 GUI 5

6.2 Logic 6

6.3 Parser 7

6.4 Storage 7

7. Testing 8

8. List of Commands 9

9. Appendix A: User Stories. 10

10. Appendix B: Non Functional Requirements 11

11. Appendix C: Product Survey 11

# Introduction

We, the developers, aim to give users a brand new way of tracking daily tasks. With interactive user design, our software – POMPOM aims to bring user the best user experience among all kinds of task tracking software or application.

# Functionalities

POMPOM allows you to do the following using command line, keyboard hotkeys and mouse click.

* Create task
* Edit task
* View task
* Delete task
* Undo task
* Set task priority
* Set task label
* Set task start date/time
* Set task end date/time

# Setting Up

POMPOM requires Java SE 8 to run. Get Java SE 8 [here](http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html). POMPOM Developers are recommended to use Eclipse IDE as POMPOM is developed using Eclipse. Get Eclipse IDE [here](http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/marsr). A Github account is also required to contribute to POMPOM’s main repository. Sign up for a Github account [here](https://github.com/join). POMPOM’s main repository is hosted [here](https://github.com/cs2103jan2016-f14-3j/main).

POMPOM uses 2 libraries; JavaFX and Gson. The usage of these libraries will be explained further in this document in the respective component it is used in.

# Architecture Overview

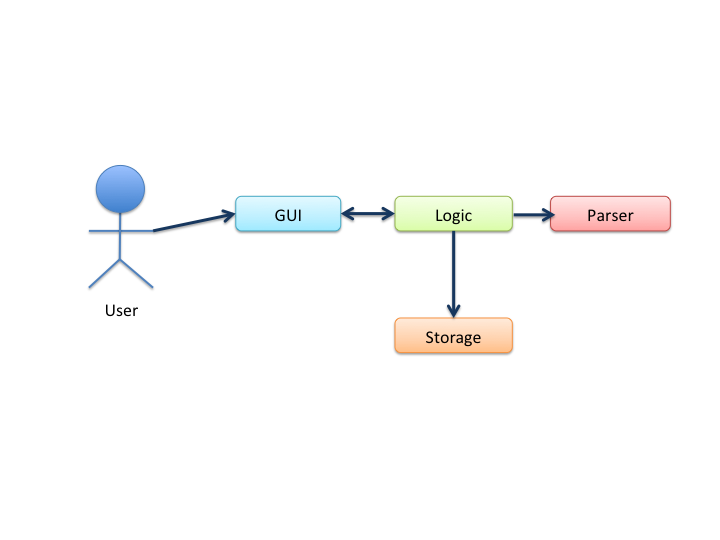


Figure 1. Architecture Overview

This architecture diagram is an overview on how GUI, Logic, Paser, Command, Storage in POMPOM interact with each other. Read on for more information about each component and how the components interact with each other.

# Interaction between Components

## 5.1 Adding Task

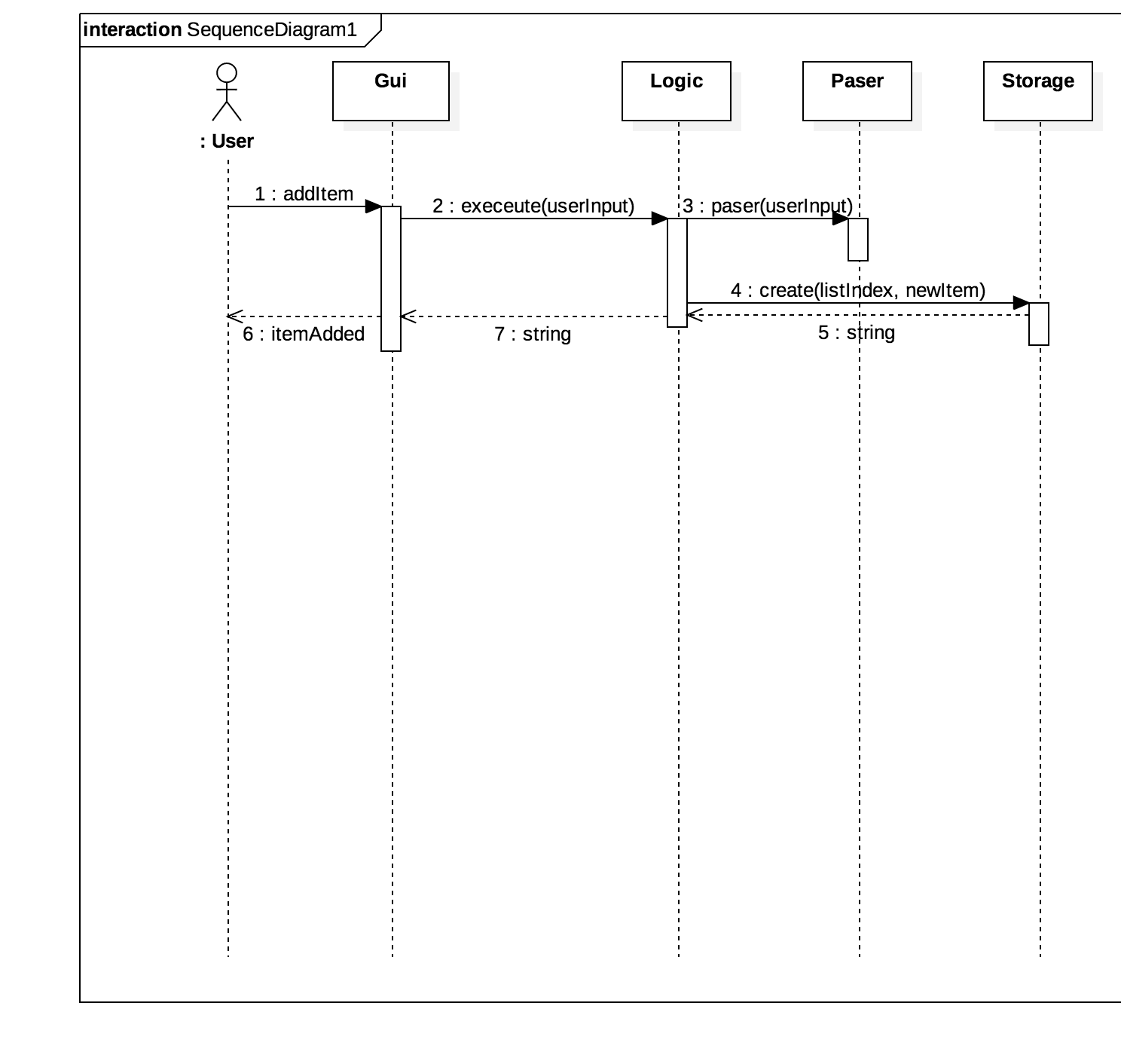


Figure 2. Sequence Diagram for Adding Task

This diagram shows the interaction between the components in POMPOM when items (a task or an event) are added by the user.

## 5.2 Editing Task

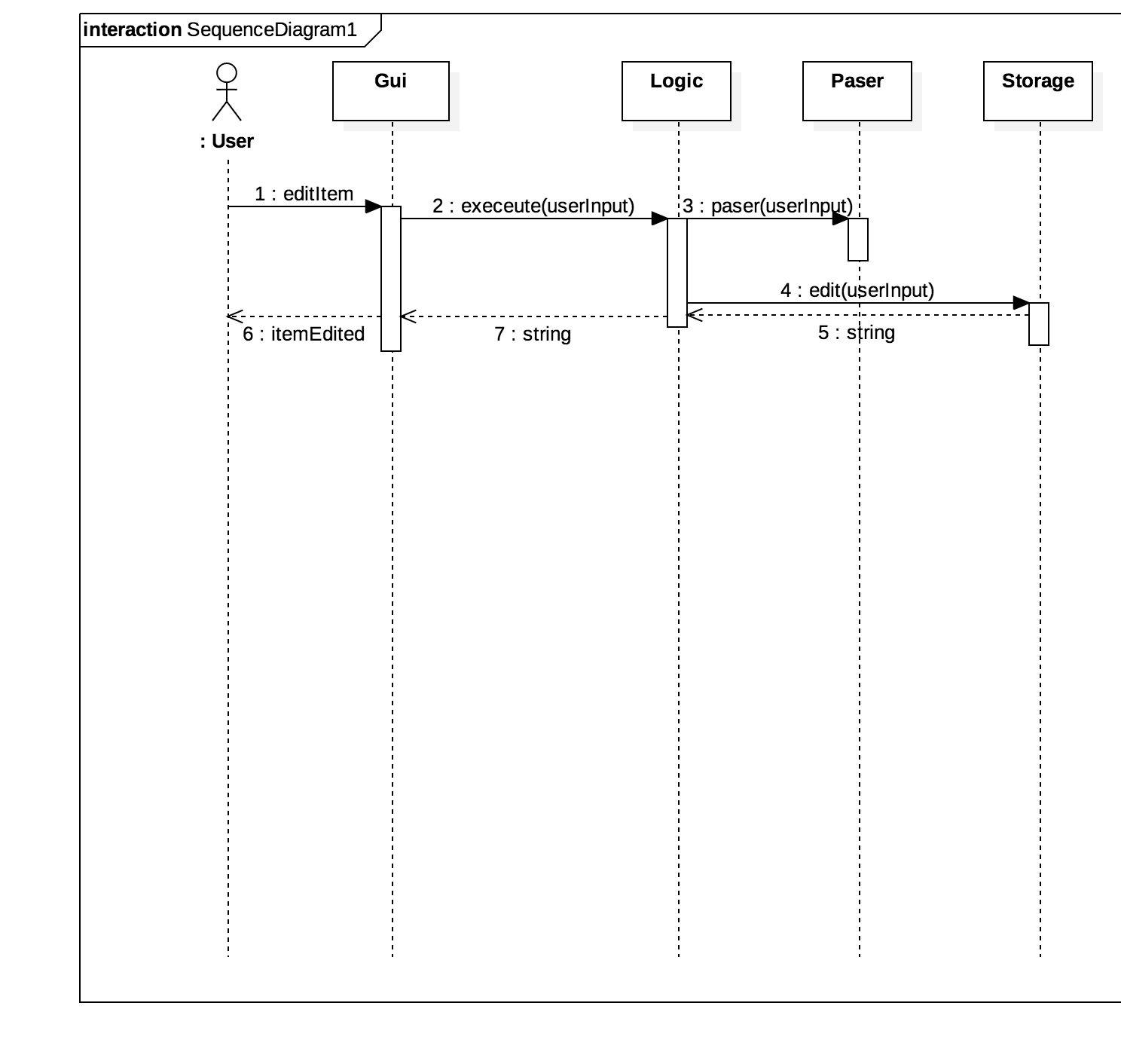


Figure 3. Sequence Diagram for Editing Task

This diagram shows the interaction between the components in POMPOM when items (a task or an event) are edited by the user.

## 5.3 Deleting Task

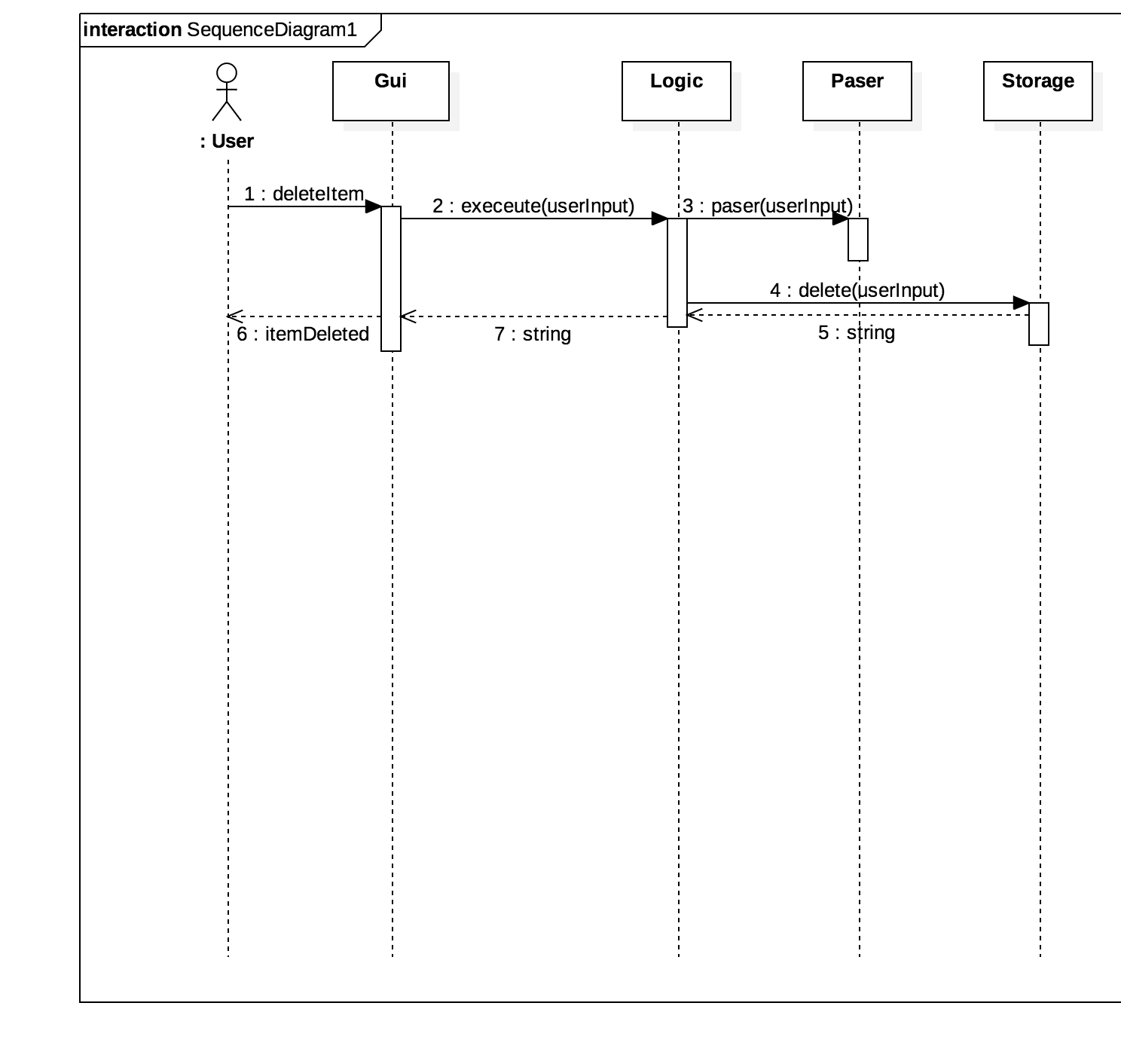


Figure 4. Sequence Diagram for deleting Task

This diagram shows the interaction between the components in POMPOM when items (a task or an event) are deleted by the user.

# Components

## 6.1 GUI

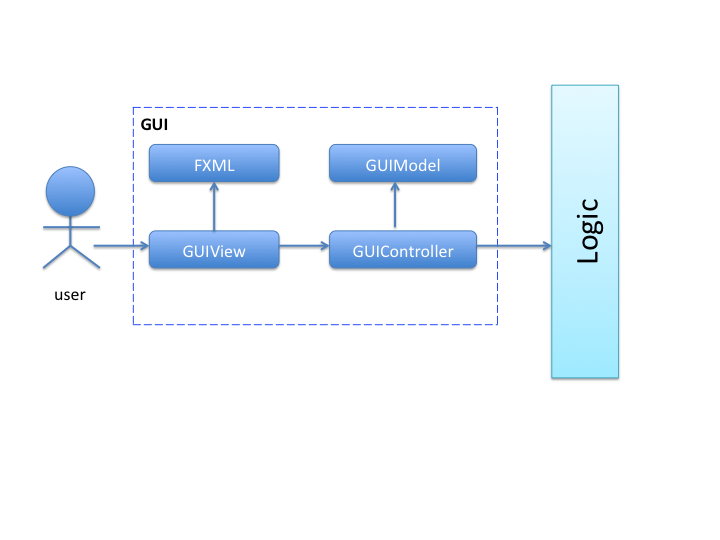


Figure 5. GUI components

The GUI component is to provide user with a graphical interface to enhance the user experience when using POMPOM.

GUI is structured using the model-view-controller framework to facilitate the separation of UI logic and display codes, which includes the FXML and the GUIView classes.

Upon starting POMPOM, it will initialize and GUIController will call logic upon user input for execution of what the user has key in or trigger via user interface.

Similarly, GUIController will have to be initialized before it is functional. FXMLLoader will initialize all the controls from the FXML file, with certain controls wrapped in runnable that is run after initialization to function.

GUI makes use of JavaFX as well as FXML together with CSS style sheet being applied to FXML file for a customized look.

## 6.2 Logic

## Macintosh HD:Users:JorelTay:Dropbox:CS2101-3T:Logic.png

Figure 6. Logic components

Logic is the controller of the application. It acts as an intermediary between GUI and the rest of the classes. Logic is used to execute the user’s requests.

This component contains a single class POMPOM which is coupled with the Parser and Storage. Making Logic a coupled component allows us to make the other components more independent. With a single point-of-failure, we can easily locate bugs in other components.

**Important APIs in Logic:**

|  |  |
| --- | --- |
| Returns | Important APIs |
| void | init()  This method initializes the storage and the undo stack. It calls refreshStatus(). |
| void | RefreshStatus()  This method refreshes and updates the status of every task and event according to current time. |
| String | execute(String userInput)  This method makes use of the Parser to create the relevant command. The command is then executed and its result, a message, is returned. |
| Storage | getStorage()  This method returns the Storage object currently in use by the application. |
| Stack<Command> | getUndoStack()  This method returns the Stack of counter actions currently stored by the application. |

## 6.3 Parser

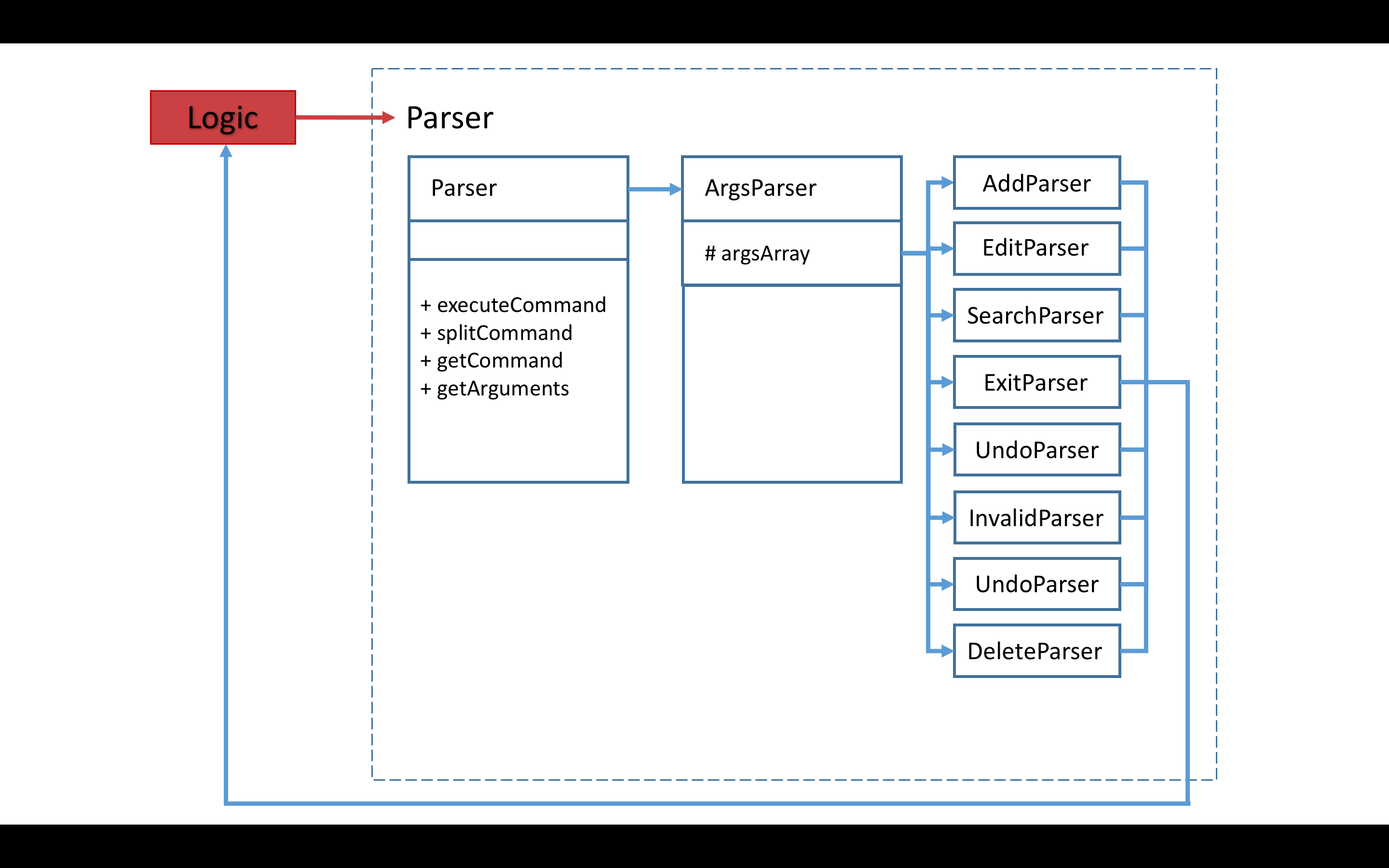


Figure 7. Parser components

The Parser reads the input from the user, breaks the input up into several parts, determines which functions in Logic to call and passes the arguments into the function that was selected to be called.

The user will also enjoy a certain degree of flexibility when they type their commands with the help of the Parser. For instance, the user does not have to strictly specify the start date before the end date in their commands as the parser is also able to understand the command if the end date is specified before the start date.

The Parser employs the Single Responsibility design principle. Every class in the parser has one responsibility. If an “add” function is required, the main Parser class will use an AddParser object to split the arguments up and pass it into the Logic component.

**Important APIs in Parser:**

|  |  |
| --- | --- |
| Returns | Important APIs |
| Parser | getInstance()  This method creates a Parser object if it was not created before. Else, it returns the previously created Parser object. |
| Command | executeCommand()  This method breaks the user input into two parts. One is the command type and the other is the command arguments. The appropriate parser is called according to the command type determined and the command arguments are passed into it. |

## 6.4 Storage

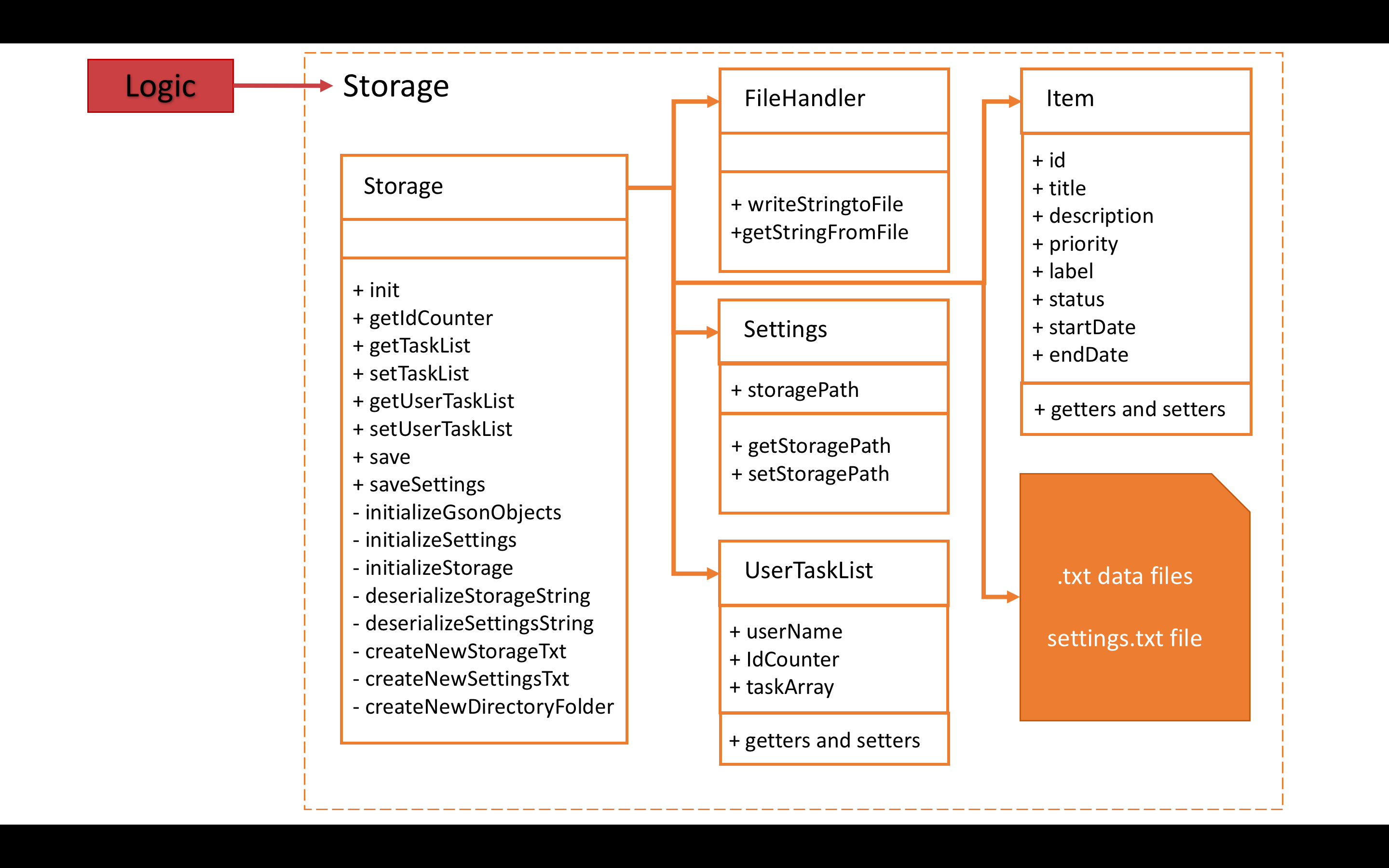


Figure 8. Storage components

# Testing

At the current point of development (v0.2), testing has yet to be implemented.

In the upcoming versions, POMPOM will use jUnit 3 testing built into the Eclipse IDE to test the program. All available and up-to-date unit test cases are on POMPOM’s GitHub main repository. POPOM developers will be able to run available jUnit test cases on Eclipse IDE to test the various components.

# Commands

**List of Prefixes**

**f:**

Specifies “from”. Indicates the start date.

e.g. f:next monday

**l:**

Specifies “label”. Indicates the label of the task.

e.g. l:homework

**p:**

Specifies “priority”. Indicates the priority of the task.

e.g. p: high

**s:**

Specifies “status”. Indicates the status of the task.

e.g. s:overdue

**List of Commands**

**Add**

1. add <task>

*e.g. add do project*

2. add <task> <mmm dd>

*e.g. add do project march 28, add do project mar 28*

3. add <task> < end date>

*e.g. add do project 28/03/2016*

4. add <task> <end date> <f:start date>

*e.g. add do project march 28 f:march 16*

5. add <task> <end date> <f:start date> <l:label>

*e.g. add do project march 28 f:march 16 l:cs2103*

6. add <task> <end date> <f:start date> <l:label> <p:priority>

*e.g. add do project march 28 f:march 16 l:cs2103 p:high*

7. add <task> <end date> <f:start date> <l:label> <p:priority> <s:status>

*e.g. add do project march 28 f:march 16 l:cs2103 p:high s:opened*

**Delete**

1. del <task id>

*e.g.* del *1*

2. del <task name>

*e.g.* del *do project*

**Edit**

1. edit <task id> <field> <new data>

*e.g. edit 1 title do cs2103*

**Done**

1. done <task id>

e.g. *done 1*

**Show**

1. show <task id>

e.g. show *1*

**Undo**

1. undo

e.g. *undo*

**Search**

1. search <task name>

*e.g. search do project*

**Help**  
1. help

*e.g. help*

**Exit**

1. exit

e.g. *exit*

# Appendix A: User Stories.

***[Likely]***

|  |  |  |
| --- | --- | --- |
| **ID** | **I can … (i.e. Functionality)** | **so that I … (i.e. Value)** |
| addTask | add tasks that has a start time and/or end time | can record tasks that I want to do with specific deadlines |
| deleteTask | delete a task | am able remove tasks that are no longer relevant |
| editTask | edit a task | make changes to an already added task |
| undoAction | undo my actions | can undo my mistakes |
| searchTask | search for a task | look for a specific task quickly |
| filterTask | filter the tasks | can differentiate between tasks that are done and tasks that are not |
| chooseLocation | specify the location of  storage | can manage where the app stores data locally |
| snoozeTask | snooze a task | can allow more time for the task to be done, delay the deadline |
| deleteMultipleTask | delete more than a task | don’t have to delete them one by one, more efficient |
| selectPriority | set the level of priority for a task | can keep track of things that are more important |
| setAlarm | set a alarm and a reminder for a task | can be reminded when the task is close to the deadline |
| multipleGUI | change the look of my app | can use it as it as I fancy |
| setShortcut | set custom shortcuts | can customise the app to my own usage |
| setSnooze | set custom snooze timer | can set different snooze for different tasks |

***[Unlikely]***

|  |  |  |
| --- | --- | --- |
| **ID** | **I can … (i.e. Functionality)** | **so that I … (i.e. Value)** |
| socialSharing | share my tasks on social media | share with my friends on what I have to do |
| googleCal | sync my tasks with my Google calendar | can sync my tasks with multiple devices with cloud storage |
| widget | have a widget for the app | conveniently use on my on desktop |
| delayTask | delay my task | reschedule my tasks |
| shortHandLanguage | use natural language to enter commands and inputs | can easily carry out an action |
| setPassword | set passwords for specific tasks | can restrict access to more private tasks |

# Appendix B: Non Functional Requirements

* The software should work on Windows 10 or MACOS.
* The program does not require Internet connection.
* The program should be very responsive to the user
* GUI is interactive and easy to navigate
* Commands are intuitive and auto-complete for commands
* Computer should have sufficient memory and ram to install the program and run the program smoothly

# Appendix C: Product Survey

|  |
| --- |
| **Product**: WunderList  **Documented by**: Ho Wei Lip  **Strengths**   1. Display of task is list-based instead of calendar-based. Easier to see and keep track of all tasks 2. Compatible with wearable technology 3. User Interface is simple and intuitive enough such that it is immediately usable 4. Customizable background 5. Able to easily interact with other users using the app, such has having conversations and sharing tasks 6. Adding new tasks is fast and simple 7. Synchronizes with cloud storage (Gmail account)   **Weakness**   1. Not for users who prefer to see their tasks displayed under certain dates like calendars. 2. There is no “reward system” within the app, making it less interesting to use. 3. The feature to label and organize tasks by priorities is weak. It only allows tasks to be “starred”. Only one level of priority is available to the user. |
| **Product**: Any.do  **Documented by**: Tan Wee Chen, William  **Strengths**   1. Able to take photos as notes for the tasks 2. The reward system in the app, where users earn points for completing tasks. Reward systems helps to create incentives to complete tasks 3. Easy to create and access different categories of tasks 4. Makes use of swipes to quickly add deadline to tasks 5. “Someday” option allows user to add tasks they know they need to do but they do not know the deadline. Easy to see tasks categorized under “Someday” also. 6. Easy to collaborate with other users 7. Synchronizes with cloud storage (Gmail account)   **Weakness**   1. UI is not intuitive. The user will need to take some time to get used to the UI before he can use the application effectively 2. UI design makes the list of tasks to be completed look messy and less simple to differentiate tasks above it or below it. 3. Can be hard to add tasks, swiping feature makes it easy to fumble around and add tasks wrongly |
| **Product**: Todoist  **Documented by**: Lau Wen Hao  **Strengths**   1. Can use Natural Language Processing (NLP) to execute the basic features 2. NLP feature is well-implemented. Able to give accurate suggestions to the commands the user has typed halfway 3. Attractive and Simple GUI 4. Able to set different level of priorities for tasks added 5. The reward system in the app, where users earn points for completing tasks. Reward systems helps to create incentives to complete tasks 6. Synchronizes with cloud storage (Gmail account)   **Weakness**   1. User interface is not very intuitive. Users will need time to get familiar on how to perform basic commands. 2. Icons used in place of labels for buttons. Layman users may not understand what the icons mean and get frustrated when they just started using the application. 3. Not optimised for different devices, some layouts are not viewable in some device |
| **Product**: Google Calendar  **Documented by**: Tay Kai Teng Jorel  **Strengths**   1. UI is simple and attractive enough for anyone to use straight away. 2. Tasks are listed differently when holding the device in a horizontal or vertical orientation. Holding the device in the vertical orientation shows the tasks as a list. Holding the device in the horizontal orientation shows the tasks in a calendar 3. Allows cloud syncing with multiple Gmail accounts, allowing the user to show tasks added to different accounts. 4. Easy to differentiate and filter tasks belonging to different accounts (by font color) 5. Can use Natural Language Processing (NLP) to execute the basic features   **Weakness**   1. User is unable to set priorities for the tasks added 2. Difficult to label and categorize tasks 3. There is no “reward system” within the app, making it less fun to use. |