

User Manual

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| Chapter | Content | Page Number |
| 1. **Introduction** | 1.1 What is FINI?  1.2 The FINI Team  1.3 System Requirements |  |
| 1. **Getting Started** | 2.1 Launching FINI  2.2 Adding Tasks 2.3 Undo Function 2.4 Deletion of Task 2.5 Updating a Task 2.6 Completing a Task 2.7 Saving Your Progress 2.8 Getting Help |  |
| 1. **Developer’s Information** | 3.1 Forward  3.2 Architecture 3.3 Command Flow  3.4 Components  3.5 Known Issues  3.6 Future Development  3.7 Links and References for Developers |  |
| 1. **Appendix** | 4.1 Command Cheat sheet Reference  4.2 Acknowledgments |  |

**1 | Introduction**

1.1 | What is FINI?

1.2 | The FINI Team

In order to ensure FINI is produced with quality, a dedicated team comprising of passionate software engineers had to be form

|  |  |
| --- | --- |
| US%20Photo | Venkatesan Harish  Team Leader, Framework & Deliverables In-Charge  Harish has been a passionate software developer for many years. In his free time, he creates new software. |
| https://lh3.googleusercontent.com/DLN77zS_AZGIMx3c-1U1LZBFjpLWj9y0EM46NyhRwr_jKGOdIEQdalkMoR6gNR6oSKdLxZFSgaEAWkD3SI2egynLuXwzV1NVZWLrT6Lej9AzwPLRRYGbJXVN9i1Csws5xE6znVs | Tan Nge Joel Jonas  Integration, Scheduling & Tracking Expert  Jonas has been studying a lot of algorithms recently, and strives to ensure that FINI runs as efficiently and smoothly as possible. |
| ./Google%20Drive/NUS/AY1516%20Sem%201/CS2103T/Photos/Qiyun.jpg | Yu Qiyun  Documentation & Code Quality In-Charge  Qiyun is a perfectionist and demands that things be done with effort and quality. She ensures that FINI’s code is readable and also abides to proper coding conventions. |
| https://lh6.googleusercontent.com/ElPbxlhiMx6_Qf_GfUi6koL-3HhHMU-09iAUJhr0TyjRcJsDxsThziUdDazj5EisQPIjrSv02uBO3EeWFxky1o4Yn7hBXANcrult2GGz0oUfAaeue1Ygcujvng9luPgTkbb0yQY | Wang Jie  Testing & Tools Expert, Lead Developer  He is Guy Po. |

**1 | Introduction**

1.3 | System Requirements

* Windows 8 or later
* Mac OS X Mountain Lion or later
* Java software installed

**2 | Getting Started**

2.1 | Launching FINI

To launch FINI, simply double-click on the FINI icon in the folder where FINI is installed.

2.2 | Adding Tasks

Adding tasks in FINI has never been simpler. The following command shows how you can specify a task along with its details.

>> add <TASK\_TITLE> // from <12/10/2015> to <15/10/2015> with priority <HIGH>

Please note that the parameters given between “< >” can be substituted with your own.

**Floating Tasks**

The first type of tasks that FINI supports is known as the floating task. Floating tasks do not have a start or end date and time. For example, if you have a book to read and there is no specific deadline associated with this task, you may enter:

>> add Read Harry Potter and the Chamber of Secrets

This command will add a new task to FINI and you should see the following screen.

There is a feedback above the command box which confirms that the task you specified has been created and added.

**Tasks with Deadlines**

The second type of task are the ones which you need to complete by a specific date or time. For example, you receive a call today about an upcoming project meeting with your team manager tomorrow evening at 7pm. To add this meeting, simply enter:

>> add Attend project meeting // 16/10/2015 7pm with priority high

**2 | Getting Started**

**Tasks with Start and End Date/Time**

Some tasks may be reminder of events that you might need to attend. For example, if you are attending Bob’s birthday party tomorrow from 7pm to 9pm, you enter:

>> add Bob’s Birthday Party // 11/10/2015 from 7pm to 9pm with priority medium

**Tasks with Recurring Deadlines**

Sometimes you may encounter tasks that you need to do on a recurring basis. For example, you may need to submit a weekly report ever Friday at 9pm. To add such a task, enter:

>> add Submit weekly report // every Friday at 9pm

2.3 | Undo Function

We recognize that the “undo” function is crucial to almost any user. This function will be useful to users in scenarios where a task has accidentally been deleted, a task has been incorrectly updated or if the user wants to quickly cancel a task that has just been added. To undo, simply enter:

>> undo

Quick Tip! FINI supports undo for up to three past commands. To undo up to 3 commands, use:

>> undo <num>

Note that <num> can take values from 1 to 3.

**2 | Getting Started**

2.4 | Deleting a Task

The user may want to delete tasks which details have been keyed incorrectly or for cases such as a meeting that has been cancelled. In this case, the user may delete the task by keying in:

>> delete <task\_id>

Note that <task\_id> takes the value of the task ID which is displayed on the user’s screen.

2.5 | Updating a Task

Updating a task can be useful for users who wish to change minor details of the task that they have already added in. To update:

>> update <task\_id> // <date>

Where <num> is the value of the task ID displayed on the user’s screen and date is the new date for the tasks in the DD/MM/YYYY format.

If there is a need to update the title of the task, the user may input:

>> update <task\_id> New Task // 17/10/2015

2.6 | Completing a Task

It is important for users to be able to indicate that a tasks is completed. To mark a task as completed, key in the command:

>> complete <task\_id>

**2 | Getting Started**

2.7 | Saving your Progress

FINI automatically saves your session after every command. However, if users prefer to manually save your session, the save command can be entered:

>> save

To save your session under a new <file\_name>, you may also enter:

>> save as <file\_name>

2.8 | Getting Help

If the user needs to recall the different commands that FINI accepts, all the user has to do is key in the following command:

>> help

A help screen with an overview of all the commands will be delivered to the user instantly.

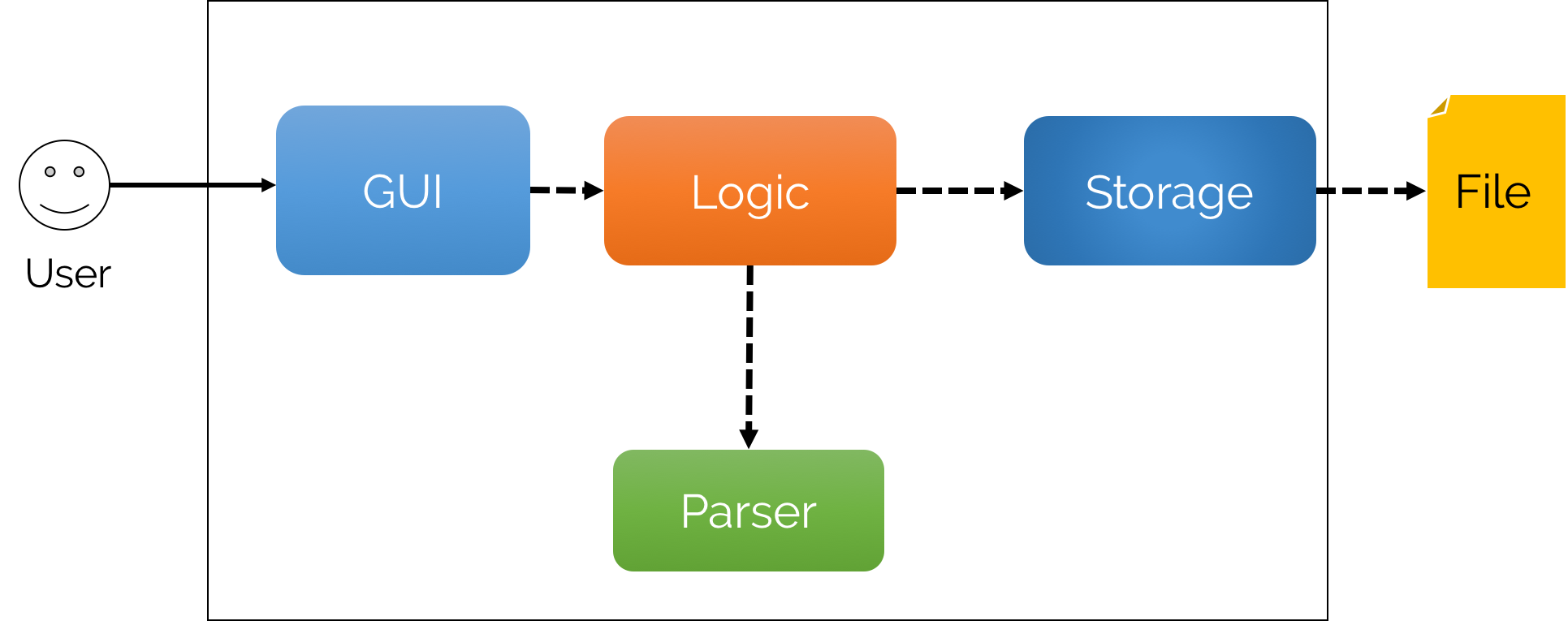
**3 | Developer’s Information**

3.1 | Foreword

FINI is a simple task manager designed for people with busy schedules. FINI is a desktop application written in Java. It utilises JavaFX as the primary GUI library. FINI is designed for interaction via the keyboard only. Hence, the use of mouse clicks is unnecessary except to launch/close FINI through the system icons.

This developer guide provides an overview of FINI’s project architecture as well as the necessary components and interactions involved. With clarity, you will be able to further contribute towards the development of FINI.

3.2 | Architecture

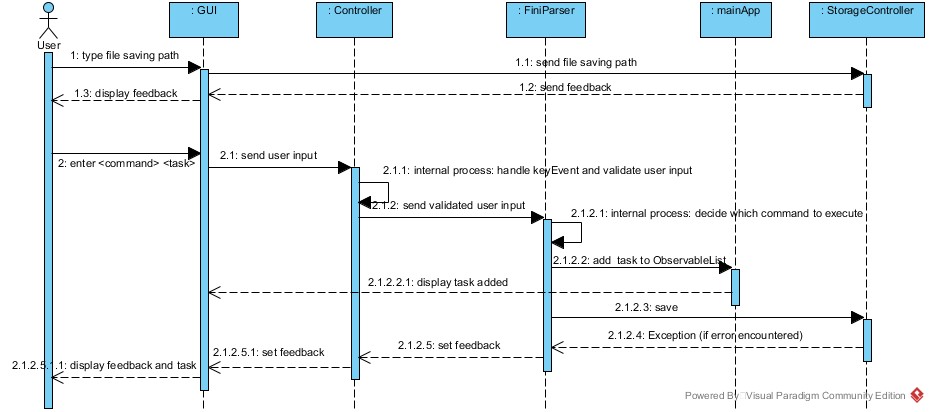


**3 | Developer’s Information**

3.3 | Command Flow

The below sequence diagram illustrates the general process of how the various components of FINI interact with one another. The flow of interactions are simple and are logically guided, starting from the point of user input.

A sample scenario is illustrated below the diagram for greater understanding and clarity.

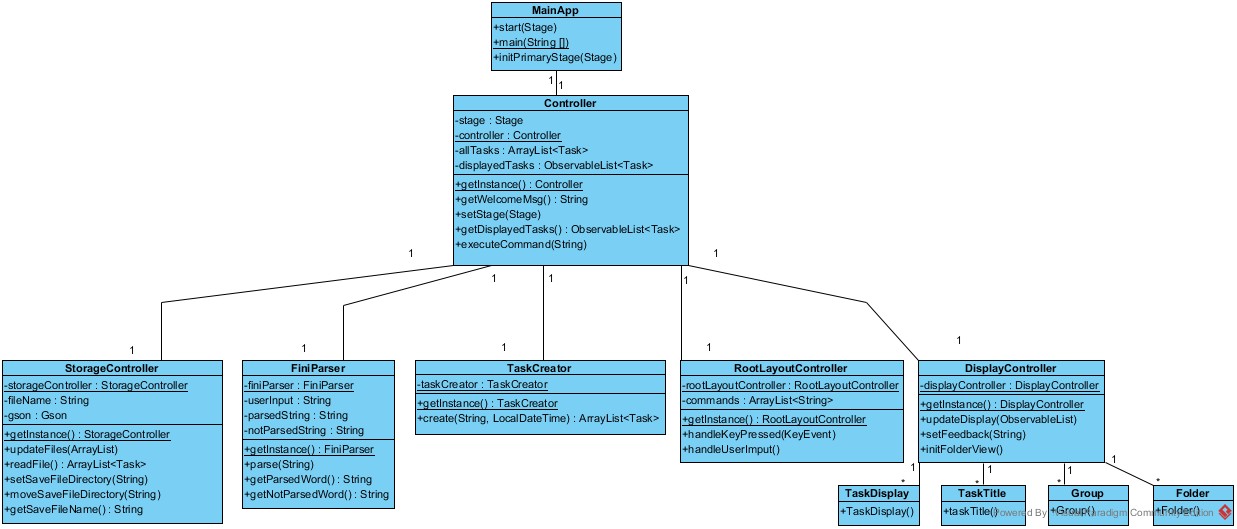


James is a user of FINI. Firstly, he launches FINI and is greeted by the Welcome Screen. He subsequently presses ENTER and he is brought to the main screen of FINI. From here, he decides to add a new task, “Call mother at 7pm”. This command is now received by the Controller which validates the user input and interprets the intended action of the user. Once the Controller interprets that the user wants to add a task, it calls FiniParser. FiniParser parses the user’s input to identify and segregate the various parts of the user’s command and calls the necessary methods to create the task. FiniParser now calls the MainApp component to add the created task to the master-list of tasks. Subsequently, FiniParser calls StorageController to ensure that the new task is updated in the text file of the user, which is the main form of storage.

**3 | Developer’s Information**

3.4 | Components

FINI has been designed from a top-down approach.



As shown in the project’s architecture, FINI is made up of 4 main components.

1. The **Graphical User Interface (GUI)** component consists of JavaFX’s XML files. These files define the layout of the GUI and provide important components for users to interact. A Java controller file is also linked to the FXML file for components which are able to receive inputs from the user.
2. The **Logic** component is responsible for interpreting the user’s actions.
3. The **Parser** component is responsible for parsing the user’s commands and creating or manipulating the relevant models
4. The **Storage** component consists of the user’s text file. This file is the input storage area for FINI to store the user’s tasks. This file is also loaded when the user launches FINI.

**3 | Developer’s Information**

**Logic**

Central to the operation of FINI is the Logic component. The Logic component is crucial in interpreting the user’s commands and then executing these commands correctly. This component is reliant only on the Storage component and works independently from the GUI component.

When the user inputs the commands, <executeCommand(String)> method is called which then parses the user’s input. After interpreting the user’s commands. For example, the command might be to add/delete/update/complete tasks. Once this command is identified, it then calls onto the necessary secondary methods to execute the action. Finally, the Logic component updates its variables present and internal data storage by interacting with the Storage component. The Storage component then writes the updated data to the local file saved on the user’s disk.

**Graphical User Interface (GUI)**

The GUI of FINI mainly consists of 2 “scenes”. Scenes are terms which describe what the user sees on the screen. The first scene is the “Welcome Screen” which greets the user and prompts the user to press “ENTER” to continue. The second scene is FINI’s main layout. The GUI component consists of numerous FXML files. FXML files serve as the backbone of FINI’s GUI as they define the layout of the various components such as the command box (for users to enter their input), task boxes etc. Furthermore, for components such as the command box, the FXML file is linked to a Controller Java file. This Controller class is responsible to provide response to the user’s interactions.

The GUI component also includes the stylesheet, <style.css>. Cascading Style Sheets (CSS) is the main markup language which is used to customise the appearance and style of JavaFX components.

3.5 | Known Issues

There are no known issues as of FINI v0.5

If you encounter any issues, please let contact us at FINIDevTeam@gmail.com and let us know about it. We appreciate any form of feedback as well.

**3 | Developer’s Information**

3.6 | Future Development

The initial release of FINI indicated and overwhelming response with many users giving positive feedback. There has been many requests from FINI users to port FINI to mobile devices such for Android and IOS. The developers are currently looking to expand FINI to those devices in the future.

3.7 | Links & References for Developers

JavaFX - <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm>

**4 | Appendix**

4.1 | Command Cheat Sheet Reference

|  |  |
| --- | --- |
| Command | Description |
| add <task\_title>  add <task\_title> // <start\_date>  add <task\_title> // <start\_date> <end\_date>  add <task\_title> // every <day> at <time>  add <task\_title> // <start\_date> <end\_date> with priority high |  |
| undo  undo <num> |  |
| delete <task\_id> |  |
| update <task\_id> |  |
| complete <task\_id> |  |
| save  save as <new\_file\_name> |  |
| help |  |

4.2 | Acknoledgements

Ms Lim Phay Yen – CS2101

Lynette – CS2103T