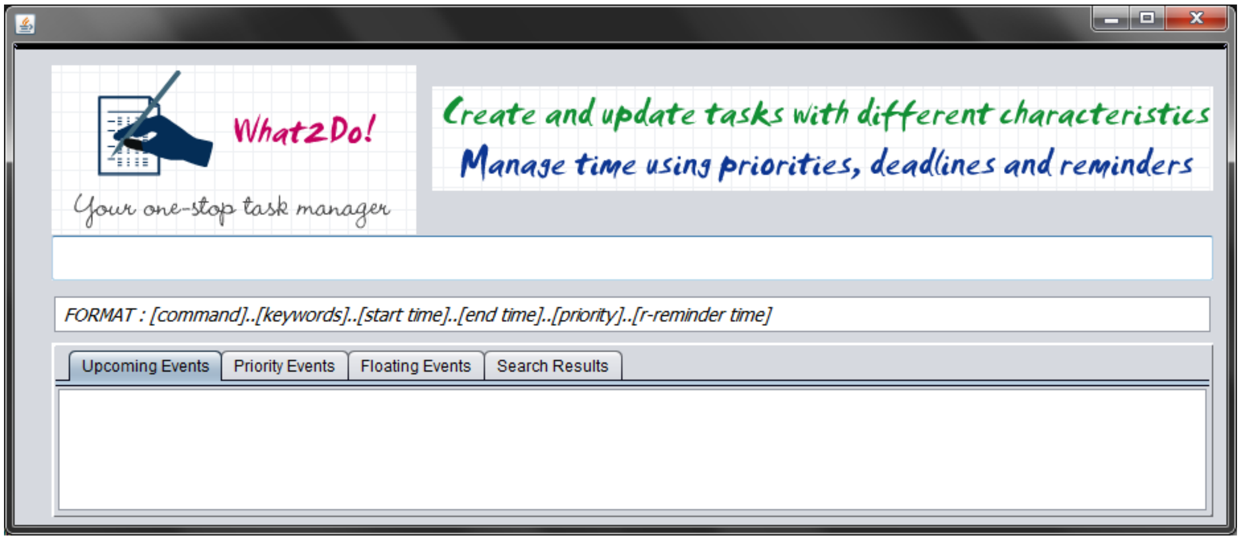
What2Do!


What2Do!
Your one-stop task manager
Create and update tasks with different characteristics
Manage time using priorities, deadlines and reminders
Upcoming Events
Priority Events
Floating Events
Search Results

FORMAT : [command]..[keywords]..[start time]..[end time]..[priority]..[r-reminder time]

Members

Akaash Gupta Team Lead, Logic Integrator	Sandeep Paul GUI	Le Khac Minh Tri Database and Time Analyzing	Anirup Sengupta Executor Class

CREDITS

- We have used the free web-application, LogoMaker to help design our software logo.
- We have also used the Jodatime library for handling Dates and Times.

USER GUIDE

What2Do! is a text based to-do manager which has several features to make it very easy for the user (i.e. you) to manage their schedule.

1. ADDING AN EVENT

You can add an event using the command line interface at the top (i.e. the bar at the top) by typing in the following format -

Format: [add]..[key words]..[time and date]..[priority]..[reminder time]

Key Words:

You can chose to insert a hash-tag. For example, if the entry is:

add meeting prof. damith #cs2103..monday 10/9

The task name would be meeting *prof. damith* and this task would be under the *cs2103* category

The hash tag is

optional.

Optional Components

*Time and Date: If left blank, then classify the task as floating task.

*Priority: In case this field is left blank, the system shall assume normal priority (h/high/l/low)

*Reminder Time: In case left blank, then the system shall not have a reminder for that task.

Examples:

add..meeting with cs2103 teammates..Sunday 9/9 11 am..2 hours

before add..study for cs2103 final exam

add..project submission deadline..26/10..high..10 days before

add..complete tutorial homework..Sunday 16/9..1 day before

2. SEARCHING

You can search for any event using keywords or tags, regardless of whether it is completed or incomplete or even if it has been deleted by typing the following command in the command-line.
The command would be:

[search] [searchwords]

This would result in a point – wise display of all the entries matching the search words.

Example: *search*

meeting Results:

1. Meeting with Prof. Damith on Monday, 10th September 2012, at 11 a.m. *(High priority so it would appear in red)*
2. Meeting with CS2103 Teammates on Sunday, 9 September 2012 at 1 p.m.
3. Meeting with girlfriend on Tuesday, 11th September, 2012 at 8 p.m. *(High priority(duh!))*

Thereafter, you can update or deleted these entries using the following commands: *update* and *delete* and referring to the search result option.

Example:

delete 2

This would result in the entry pertaining to meeting with cs2103 teammates being cancelled.

update 3

The original entry would appear and you would be free to edit the information. In case of tasks that are completed, you can add a *done* at the end, and this shall result in the task being archived.

3. UNDOING OPERATIONS -

You can undo the last made action/operation by either typing the command undo or by clicking on the undo button in the top-right corner of the interface. The *undo* keyword, when entered in the CLI, cancels the previous action and the

system reverts to the previous state.

4. SOME FLEXIBILITY IN COMMAND FORMAT

You shall have the option to use shorthand notations, if you wish to do so. This will make your work easier and faster. Examples:

add: + OR *a*

delete: - OR *d*

search: *s*

update: *u*

5. VIEWING CALENDAR

The *view [type of calendar]* command will show a calendar in the format specified by you.

Example:

view day shall display the day's schedule of events.

view week shall display the week's schedule of events. *view*

month shall display the month's schedule of events.

OTHER GUIDES

[These features are yet to be implemented but shall be done so in subsequent versions of the project ☺]

1. How to access Help and Troubleshooting -

You can seek help using the help button. The Help button is available on the top right hand corner of the GUI. Clicking on this button shall link to a new side window opening. This window shall have detailed information pertaining to all the features of our system, as well as instructions on how to use them. Also, provided will be troubleshooting for the software.

2. **How to use power-search -**

   You can do a power-search in the same way as a simple search. This utility is a more comprehensive search engine that "Simple Search". It allows you to enter words which match only partially (as opposed to wholly in the case of Simple Search) to previously added tasks and display these tasks, allowing you to choose the one that he wants to access.
   Example

   search cs

   shall lead to the display of all tasks which contain "cs" anywhere in the keyword. It is important that you use specific keywords for power search to ensure that you do not get an unmanageable number of results. (Example *search a*)

3. **Enable Google and Facebook Calendar integration (optional) -**

   This can be enabled in the settings option under the File options. You will also be prompted to integrate on first use of the program. This utility has to be enabled by you before it becomes operational. Once so, the utility shall scourge the your Facebook and Google accounts, access their calendars, and incorporate their entries with the What2Do! calendar. In case of clashes in the timeslots, the system shall prompt you for a confirmation before proceeding.

4. **Suggestions box listing all the possible commands that you can make.**

   Right below the CLI, there shall be a "Suggestion Box" which displays the format of the user entry. (Example: [available operations]..[keywords]..[date and time]..[priority]..

5. **How to use the reminder command - Alarm with snooze intervals -**

   While creating a task, you have the option of setting a reminder, which would be in the form of a audio alarm(if the system is running at that time) or a textual reminder. You have the option of setting the time for the reminder – 20 minutes before deadline, 1 day before the meeting etc. See "ADDING AN EVENT" for command syntax example.

6. **How to find empty slots in the timetable -**

   In case you are unsure of free timeslots in his schedule, you may use this feature which shall display the nearest available timeslots. You have the option of refining this search by specifying the duration of the desired free timeslot.

7. **How to check if there is a clash with other events -**

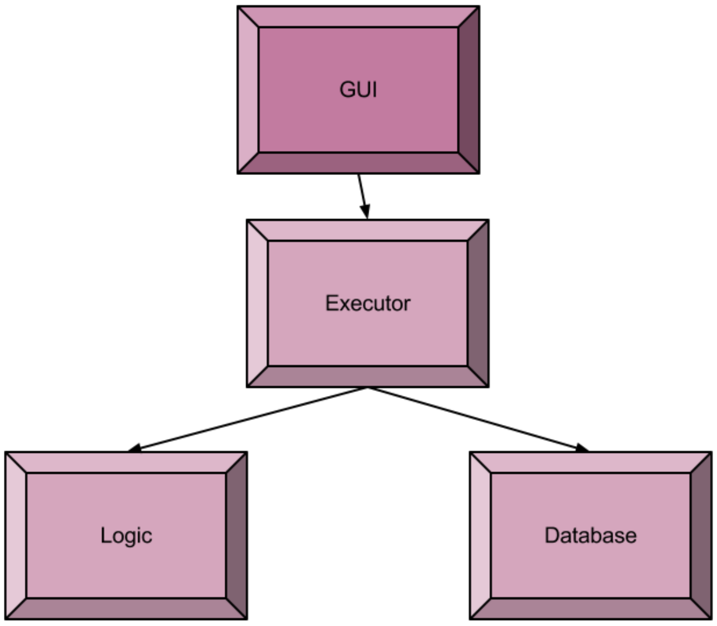
   In case you try scheduling two events whose timeslots overlap, the system shall prompt you for a confirmation before proceeding.

8. Priority based colouring – how to set priority for events -

At the time of task creation, you shall have the option to set the priority to high (h/H), low(l/L) or normal(n/N)(default). High priority tasks shall be displayed in RED, and low priority ones in YELLOW.

Developer's Guide

- ***Architecture:*** **n-level architecture –** For our project, we chose a n- level architecture with a top-down approach. So, we started building all the components individually and integrated them continuously as we developed the parts individually. Our Architecture can be demonstrated by this diagram –



- ***Important APIs:*** Listed below are the important APIs of all the classes in a tabular format and what classes they are called by –

1) Logic –

Function Name -	***What It Does -***	***Called By -***
getPriority()	Gets the priority specified in the user input	Executor
getKeyWords()	Returns the keywords	Executor
getReminderTime()	Parses the reminder time and returns the number of milliseconds.	Executor
getStartTime()	Returns the start Time as a String	Executor
getEndTime()	Returns the End Time as a String	Executor
getInteger()	Gets the Integer from the delete Command	Executor
getCommand()	Returns the type of command (add, delete…)	Executor
getHashTags()	Returns the List of Hash Tags as a Vector	Executor
getEventID()	This function returns a unique ID for an event by converting the current time to a string of numbers.	Executor

2) Executor -

Function Name -	***What It Does -***	***Called By -***
analyze()	Analyzes the Input String to get the Command and calls other functions accordingly.	GUI
undoLast()	Undoes the last Action by the User.	Executor.analyze()
analyzeAndSearch()	Analyzes the search Input and performs the search.	Executor.analyze()
analyzeAddInput()	Analyzes the Add parameters and adds to database.	Executor.analyze()
markDone()	Marks a task as Done.	Executor.analyze()
markNotDone()	Marks a task as Undone.	Executor.analyze()
updateEvent()	Updates an event.	Executor.analyze()
printDatabase()	Returns a Formatted String with all the non-floating events in the Database.	GUI
printFloatingDatabase()	Returns a Formatted String with all the floating events in the Database.	GUI
printPriorityDatabase()	Returns a Formatted String with all the non-floating events in the Database sorted in a high to low priority order.	GUI
printSearchResults()	Returns a Formatted String with the search results.	GUI

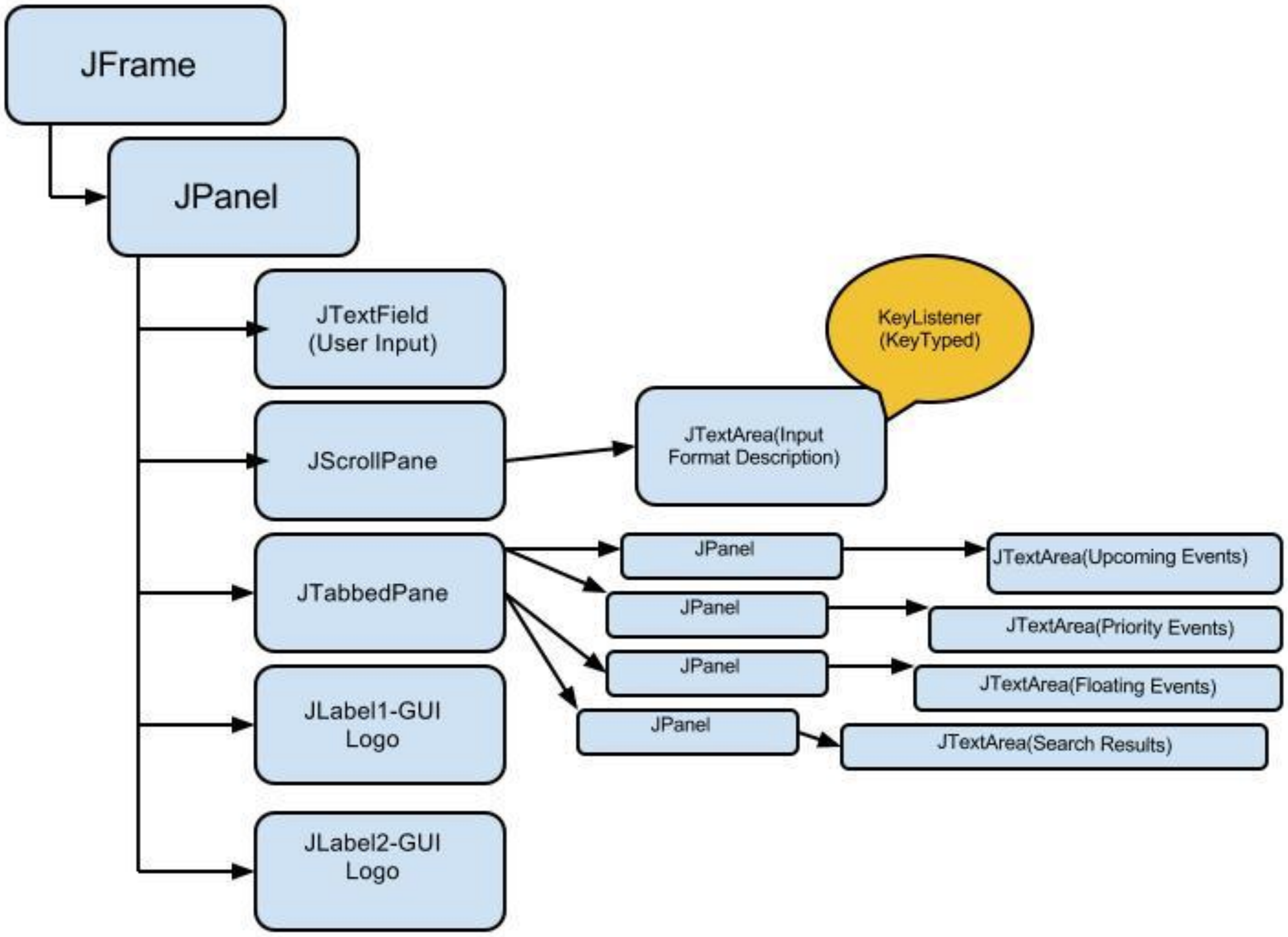
3) ListOfEvent – This is the Database Class.

Function Name -	***What It Does -***	***Called By -***
syncDataToDatabase()	Writes all data to local database.	GUI, Executor
getCurrentListOfEvent()	Returns the entire Database as a Linked List	Executor
size()	Returns the Size	Executor
markDone()	Marks an event as Done	Executor
markUndone()	Marks an event as Undone	Executor
remove()	Removes an event from the database.	Executor
add()	Adds an event to the database.	Executor
setUpDataFromDatabase()	Loads the events into the program from the local database.	GUI
get()	Returns an event at a particular index.	Executor

4) ListOfArchive – This is the List of last actions performed by the User.

Function Name -	***What It Does -***	***Called By -***
add()	Adds the last action performed by the user to the Action Archive.	Executor
undo()	Undo's the last action performed by the User.	Executor

Description of the GUI


JFrame
JPanel
JTextField (User Input)
JScrollPane
JTabbedPane
JLabel1-GUI Logo
JLabel2-GUI Logo
KeyListener (KeyTyped)
JTextArea(Input Format Description)
JPanel
JPanel
JPanel
JPanel
JTextArea(Upcoming Events)
JTextArea(Priority Events)
JTextArea(Floating Events)
JTextArea(Search Results)

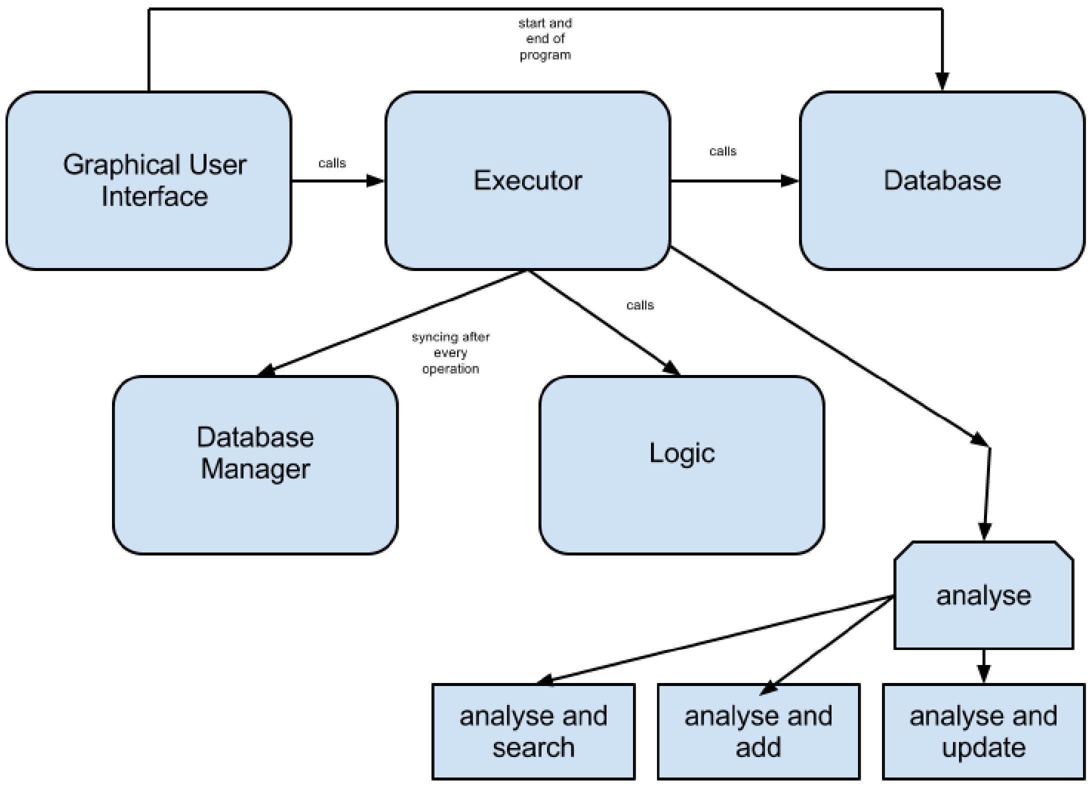
- The JTextField contains a **KeyListener**-it “listens” for the keyword “\n” or “Enter” which signifies that the user has completed his input.
- The GUI then sends the user input to the Executor class, which analyses, parses the input and updates the database.
- These databases values are then returned to the GUI and are displayed in the three JTextAreas under the JTabbedPane-one each for Upcoming, Priority and Floating Events. There is another such JTabbedPane for Search Results to be displayed(the search results are obtained from the Executor class)

- The GUI has been developed as a pure **keyboard version of Siri.** Therefore, we have deliberately omitted buttons for mouse clicks(ex. “Done”, “Exit”). Instead these actions are performed by keyboard commands (“\n” and “exit” respectively)
- The four tabs(Upcoming, Priority and Floating Events and Search Results) can be accessed by the keyboard as well. The shortcut for each of these tabs is Alt+1, Alt+2, Alt+3 and Alt+4 respectively.
- The GUI also has a JTable which has not been utilized (yet).
- Code Example for displaying data obtained from Executor Class:

```
private void textField1KeyTyped(java.awt.event.KeyEvent evt) throws
IOException {// GEN-FIRST:event_textField1KeyTyped
            jTabbedPane3.setMnemonicAt(0, evt.VK_1);
            jTabbedPane3.setMnemonicAt(1, evt.VK_2);
            jTabbedPane3.setMnemonicAt(2, evt.VK_3);
            if (evt.getKeyChar() == '\n') {
                  String data = textField1.getText();
                  if (data.equals("exit") == true) {
                        System.exit(0);
                  }
                  Executor.analyze(data);
                  String upcomingEvents=Executor.printDataBase();
                  String priorityEvents=Executor.printPriorityDataBase();
                  String floatingEvents=Executor.printFloatingDataBase();
                  //String searchResults=Executor.printSearchResults();
                  upcomingEvents=format(upcomingEvents);
                  priorityEvents=format(priorityEvents);
                  floatingEvents=format(floatingEvents);

                  jTextArea1.setText(upcomingEvents);
                  jTextArea2.setText(priorityEvents);
                  jTextArea3.setText(floatingEvents);
                  //jTextArea4.setText(searchResults);
                  textField1.setText("");

            }


      }
```

- ***Design Description -*** So, the GUI class is the interface between the user and system. The GUI then calls functions in Executor Class to analyse the user Input – to do this, the executor class calls functions in the Logic Class to analyse all the input parameters and get the processed results. The Executor class then calls the database to perform the required tasks. It also adds the last performed task to an Archive of Actions by the user which is used to undo operations. After the control returns to the GUI, it updates all the fields with the new Databases.

  The GUI also accesses the ListOfEvents Class to load and write to local databases.



- ***Instructions For Testing –*** We have done individual unit testing for all the individual parts as well as carried out GUI testing for the implemented functions.