CS2103: Week 2 Tutorial

From the Handbook,

“The software should help the user track these to-do items in a systematic way so as to facilitate the user in deciding what tasks to do and when to do them”

**Must-have features:**

1. Support for **events** (i.e., has a start time and end time), **deadlines** (tasks that have to be done before a specific deadline), and **floating tasks** (tasks without specific times). Note that these category names are temporary names only. You should figure out what are the best names for these categories. Are these names intuitive to end-users? Should they even care about how the tasks are categorized?
2. **CRUD** (i.e., Create, Read, Update, Delete) support for tasks.
3. **Undo** operations (at least for the most recent action)
4. Some **flexibility in the command** format, i.e. support a few natural variations of the command format.
5. **Simple search**: A simple text search for finding an item if the user remembers some keywords from the item description.
6. Some way to **keep track of which items are done** and which are yet to be done.
7. The ability to **specify a specific folder as the data storage location**. With this feature, Jim can choose to store the data file in a local folder controlled by a cloud syncing service (e.g. dropbox), allowing him to access task data from multiple computers.

**Extra features (choose one feature bucket):**

Difficulty-low

* **AutomatedTesting:** Achieve a very high level of automated testing.
* **RecurringTasks:** Support for managing recurring tasks. e.g. CS2103 lectures are a recurring task from week 1 to week 13 except in recess week.

Difficulty-medium

* **PowerSearch:** More powerful and intelligent search. e.g. search for empty slots, near-match search, auto-complete (similar to Google search box), filters for other attributes (e.g. start time).
* **FlexiCommands:** High flexibility in command format. e.g., non-strict ordering of keywords, ability to specify aliases for commands, support for more ‘natural’ language input, multiple undo/redo. *How flexible should the format be?* As flexible, intuitive, and user-friendly as you can make it.

Difficulty-high

* **GoodGui:** A good GUI to give visual feedback to the user e.g., in-built guidance for new users, good visual feedback for actions, feedback while typing commands, hotkey to activate/hide, notifications, etc. conveying more information visually (e.g. more important/urgent tasks stand out from the rest, longer events look different from shorter events, easy to see free slots, etc.)
* **GoogleIntegration:** Google Calendar integration. e.g. upload todo items to GCal, two-way sync, Support for GCal quick add command format, etc. You may also use the Google Tasks API for this.
* Propose your own (subject to approval).

**Constraints:**

**Desktop**: The software should work on a desktop without network/Internet connection. It should not be a mobile app or a cloud-based application.

**CLI**: Command Line Interface is the primary mode of input. If you implement a GUI, it should be primarily for output. That is, the GUI is used to give visual feedback rather than to collect input. Some minimal use of the mouse is fine (e.g. to click the minimize button), but the primary input should be command-driven. Mouse actions should have keyboard alternatives and typing is preferred over key combinations. Design the app in a way that you can do stuff faster by typing compared to mouse or key combinations.

**Standalone**: The software should work stand-alone. It should not be a plug-in to another software. However, you can build optional extensions that integrate your application with other existing software. Furthermore, you are allowed to build extensions that can plug into your software.

**No-Database**: The software should not use relational databases. Data storage must be done using text files you create yourself.

**Human-Editable-File**: The data should be stored locally in the form of a human editable text file. The intention of this constraint is to allow advanced users to manipulate the task list by editing the data file directly.

**OO**: A significant part of the software should follow the Object-oriented paradigm. However, some parts of the application can be non-OO only if it can be justified.

**Windows**: The software should work on the Windows 7 or Windows Vista OS. It should work on both 32bit and 64bit PCs.

**No-Installer**: The software should work without requiring an installer. Having an optional installer is fine as longs as the portable (non-installed) version has all the critical functionality.

**External-Software**: The use of third-party frameworks/libraries is allowed but only if they,

are free.

do not require any installation by the user of your software.

do not violate other constraints.

and subject to prior approval by us. We will not allow third-party software that can interfere with the learning objectives of the module.

From our discussion,

Features (To Have):

1. Auto-minimize to tray function when not in use after (XXX) time
2. Move function to move the event during reminder alarm or anytime to another date/time
3. As a user, I want to have a checklist at the end of the day (user set) (time)
4. Auto-complete for commands (hints)
5. As a user, I will want to type in informal language to set in my schedule
6. Alarm for clashes in time (warning); can still add at time slot
7. Be able to set different themes (eg. classy, cutesy etc)
8. Categorize the events by colours
9. Be able to see by day, month and year
10. Cancel events (still shown but marked as deleted), for last minute changes
11. Alarm to limit schedule (include workload rate tracker)
12. Search function (by String/Date)
13. As a user, there is a “schedule” feature where a calendar would pop up, and I could see my planned activities (alarm)
14. As a user, there is an archive option so that I can move it out of the inbox

Features (Maybe):

1. As a user, I would like to have a time zone feature to record the schedule
2. Make another feature to add on notes to events scheduled

Extra points about Jim,

Workflow:

1. Decides the follow up action required by that email
   1. Can do immediately – Do right away and archive (move out of Inbox)
   2. Cannot do immediately – Schedules the follow up action in calendar and archives. If cannot decide a good time to do the action, schedule it in a relatively free area in calendar.
2. Free time: looks at calendar and selects tasks that he can do at that time. Once a task is done, mark as ‘done’. If there is a further follow up action required, schedules in the calendar.
3. Periodically reviews calendar by picking tasks that could not be completed and either reschedules them or discards them as ‘cannot do’.
4. Non-email related tasks also entered in calendar.

Problems to Solve:

1. Rather have desktop software than can activate quickly, preferably (though not a must) by pressing a keyboard shortcut. Eg, Launchy tool, launched by pressing Alt+Space
2. Can capture tasks that need to be done before a specific date/time, or after a specific date/time, or tasks without specific times
3. Prefer ‘one-shot’ approach. e.g., to be able to type in commands such as "add July 10, 5-6, project meeting", Eg GCal’s ‘quick add’ feature
4. Can use without internet connection
5. Do certain things important to his workflow e.g. look for suitable slot to schedule an item, mark item as done, decide what to-do item to do next, postpone item, etc
6. Facilitate multiple slots to be “blocked” when the exact timing of a task is uncertain, and release the blocked slots once the timing is finalized, e.g. block Mon 2-3pm and Tue 2-3pm for a meeting with boss → the meeting is confirmed for Tue → automatically release Mon slot.