

## Clockwork

Clockwork

Sat 09 Apr 2016

F1  
?

F2  
📅

F3  
📁

Del  
—





Esc  
↩

All Tasks

No	Name	Time	Date
6	buy cat food	10:00 in 17h	Sun 10 Apr 2016
7	cs2103 project submission	23:59	Mon 11 Apr 2016
8	ee2024 project submission	09:00	Tue 12 Apr 2016
0	(Recurring) go swimming	16:00	Sat 16 Apr 2016

Added Deadline "ee2024 project submission" by Tue 12 Apr 2016 at 09:00

Supervisor: Ng Hui Xian Lynette  
Extra feature: Good GUI

 <p>Morgan James Howell</p> <p>Team Lead, Storage API and Testing Lead, Git Expert</p>	 <p>Rajendran Premkumar</p> <p>Logic API Lead, Storage API, Code Quality</p>	 <p>Xu Haoting</p> <p>Logic API Co-Lead, Integrator</p>	 <p>Rebekah Low Jin Yan</p> <p>UI API Lead</p>
---	---	---	---

## Credits

We would like to acknowledge the following libraries for making our product possible:

Joda-Time API

JavaFx

JavaFXtras-Agenda

ControlsFX

FontAwesomeFX

Natty Date Parser API

Gson

Slf4j

Gunit

JUnit

Antlr-Runtime

## User Guide

Introduction: Meet Clockwork.....	4
System Requirements.....	4
Quick Start: the Basics.....	4
Common Commands	
Adding Events.....	6
Editing Events.....	7
Marking Events.....	7
Deleting Events.....	7
Searching for Events.....	8
Undo/Redo.....	8
Exit.....	8
GUI Interaction	
Dynamic Quick Help.....	9
Hot Keys.....	9
Calendar/Agenda View.....	10
Category Buckets.....	11

## Developer Guide

Overall Architecture.....	12
Main Components	
User Interface.....	14
Logic.....	15
Storage.....	18
Parser.....	19
Shared.....	20
Testing.....	20
Notable Features.....	20

## Appendices

Appendix A.....	21
Appendix B.....	23
Appendix C.....	23



# ClockWork User Guide

## Introduction: Meet Clockwork

Are you a command-line addict? Are you looking for a scheduler with the ease and dexterity of performing common tasks on Unix? Look no further. Clockwork is a to-do list scheduler that runs entirely on user typed commands, backed with the option of aesthetically pleasing GUI feedback including Google-like calendar and agenda views. Through the command line interface, you may:

- Add, modify, undo, redo, and delete tasks of all varieties
- Display tasks through calendars, agendas, and text
- Mark tasks to indicate priority, completion, and type
- Search through tasks to retrieve information based off date, name, etc.
- Get help, conveniently at anytime in an accessible format
- Navigate quickly with hot keys and bins that organize tasks

Clockwork does all that, and much more! It is a tool that holds the capacity to serve as a great addition to any developer's suite of tools, holding the ability to have input piped in from users and other running programs so that it can reflect beautiful GUI agendas and calendars without ever having to click the mouse.

## System Requirements

Please ensure that you have the latest version of Java Runtime Environment (JRE) installed on your computer locally. You can pick up the latest release for free by following this link:

<http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

## Quick Start

- 1) Download a free copy of Clockwork at either:  
<https://www.dropbox.com/s/75v07bae6flmn1p/Clockwork.jar?dl=0> or  
<https://github.com/cs2103jan2016-T09-5j/main>
- 2) Double-Click "Clockwork.jar"

## Getting Started

Shows current date

Hotkeys for help, switching of views, minimizing and returning to previous view

CLI to enter commands

Display Box: Supports different views such as calendar and agenda view

Status Bar: Shows feedback based on command entered

The screenshot shows the Clockwork application window. At the top left is the Clockwork logo. Below it is a date box showing 'Sat 09 Apr 2016'. To the right of the date box is a hotkey bar with five buttons: F1 (with a question mark), F2 (with a calendar icon), F3 (with a folder icon), Del (with a minus sign), and Esc (with a return arrow). Below the hotkey bar is a search bar labeled 'All Tasks'. Under the search bar is a table with four columns: 'No', 'Name', 'Time', and 'Date'. The table contains four rows of task data. At the bottom of the window is a status bar showing a message: 'Added Deadline "ee2024 project submission" by Tue 12 Apr 2016 at 09:00'. Annotations with arrows point to these key features.

No	Name	Time	Date
6	buy cat food	10:00 in 17h	Sun 10 Apr 2016
7	cs2103 project submission	23:59	Mon 11 Apr 2016
8	ee2024 project submission	09:00	Tue 12 Apr 2016
0	(Recurring) go swimming	16:00	Sat 16 Apr 2016

Added Deadline "ee2024 project submission" by Tue 12 Apr 2016 at 09:00

## ClockWork Quick Start Guide

Using ClockWork is simple, just use the following commands,

- **add** [task\_name] from [starting period] to [ending period]
- **edit** <ID> [<newName>] [from <newStartTime>] [ to <newEndTime>] [by;on;at <newDeadline>] [every <interval> ] [until <limit>]
- **mark** <ID>
- **delete** [task IDs]
- **search** [<taskName>]
- **undo**
- **redo**
- **exit**

## User Manual for CLI

### Adding an event

e.g. **add** <task\_name> from [starting period] to [ending period]

add do cs2103 dev guide from 10am on 9 april to 10am on 11 april

add swimming every day

add kayaking every saturday at 9 am

add read a book

add finish cs2103 user guide by 10pm today

add cg 2271 lab demo from 2pm to 4pm on 13 apr

add dinner with family on sunday at 7pm every week

#### Compulsory details:

- ➔ Task name entered after the "add" keyword.

Optional: starting and ending period.

#### Note:

Start and end period can just be the time period on a single day e.g. from 10am to 12pm on 11 April.

If no start or end period is specified, task will be a floating task.

Use "by" keyword to specify deadline

For recurring tasks, every day can be replaced by e.g. every Friday, every week, every month, every year, on Friday every week etc.

## Editing an event

**edit <ID> [<newName>] [from <newStartTime>] [ to <newEndTime>]  
[by;on;at <newDeadline>] [every <interval> ] [until <limit>]**

edit 6 from 10pm on 11 april to 9am on 12 april

edit 6 from 12pm to 1pm on sunday

edit 6 buy condom

edit 6 from 3pm

edit 6 by 5pm

- Enter the id of the task you want to edit
- This is followed by the field that you want to edit, this can be the task name, task start period, end period, or the task deadline.

## Marking an event

**mark <ID>**

mark 7

- Enter "mark" followed by the id of the task to mark the task as completed

Note: marking an event does not delete it from memory.

## Deleting an event

**delete <ID>**

delete 12

- Enter "delete" followed by the id of the task to delete it

Note: deleted tasks can be recovered by using the undo command.

## Searching for an event

**search <task\_name >**

search cat

- ➔ Enter search followed by the name or part of the name and search will display all the tasks with the same search parameters.
- ➔ "search cat" will display all tasks with the name containing cat e.g. buy cat food and buy cat sand

## Undo

undo

- ➔ Type "undo" to undo the previous command

## Redo

redo

- ➔ Type "redo" to roll back on previous undo.

Note: redo will not be possible without prior undo

## Exit

exit

- ➔ Type "exit" to exit the program.



## GUI features

The diagram illustrates the GUI features of the Clockwork application, showing a list of commands, a dynamic help bar, and a detailed help tab.

**Dynamic Help Bar:** Shows Accepted Commands

**Commands:**

- add
- mark
- search

**Dynamic Help Bar:**

- F1: ?
- F2: [Calendar icon]
- F3: [Folder icon]
- Del: -
- Esc: [Return icon]

**F1: Pressing the "F1" key allows the user to open up the help tab which displays the detailed input format for each command.**

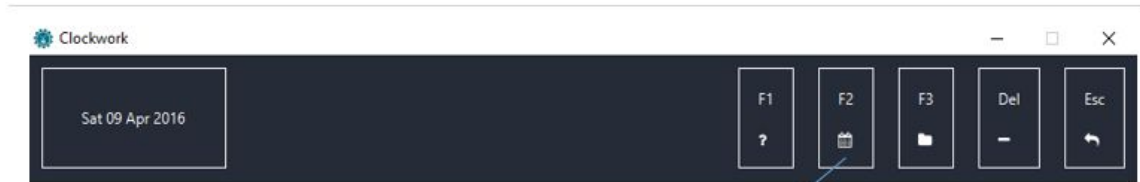
**Help Tab:**

- Add: +
- Delete: -
- Edit: [Pencil icon]
- Search: [Magnifying glass icon]
- Mark: [Checkmark icon]
- Undo: [Undo icon]
- Redo: [Redo icon]
- <Tab>: [Tab icon]

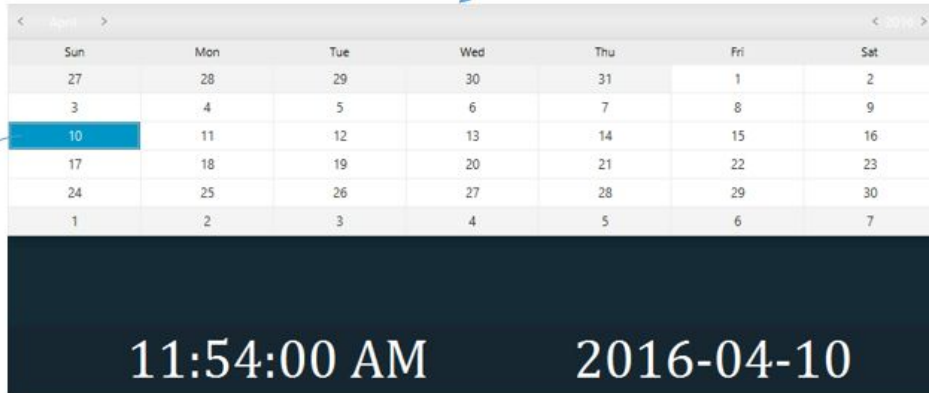
add [task name] from [starting period] to [ending period]  
delete [task IDs]  
edit <ID> [<newName>] [from <newStartTime>] [to <newEndTime>] [by;on;at <newDeadline>] [every <interval>] [until <limit>]  
search [<taskName>; <date>]  
mark <ID>

**Del: Pressing the "del" key minimizes ClockWork to the**

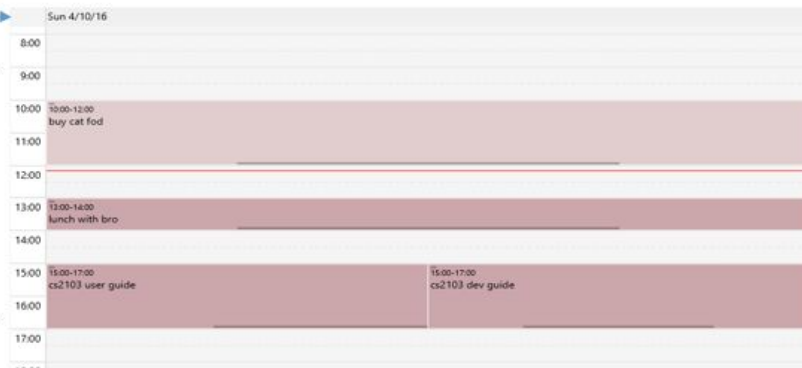
**Esc: Pressing the "Esc" key returns the user to the previous page**

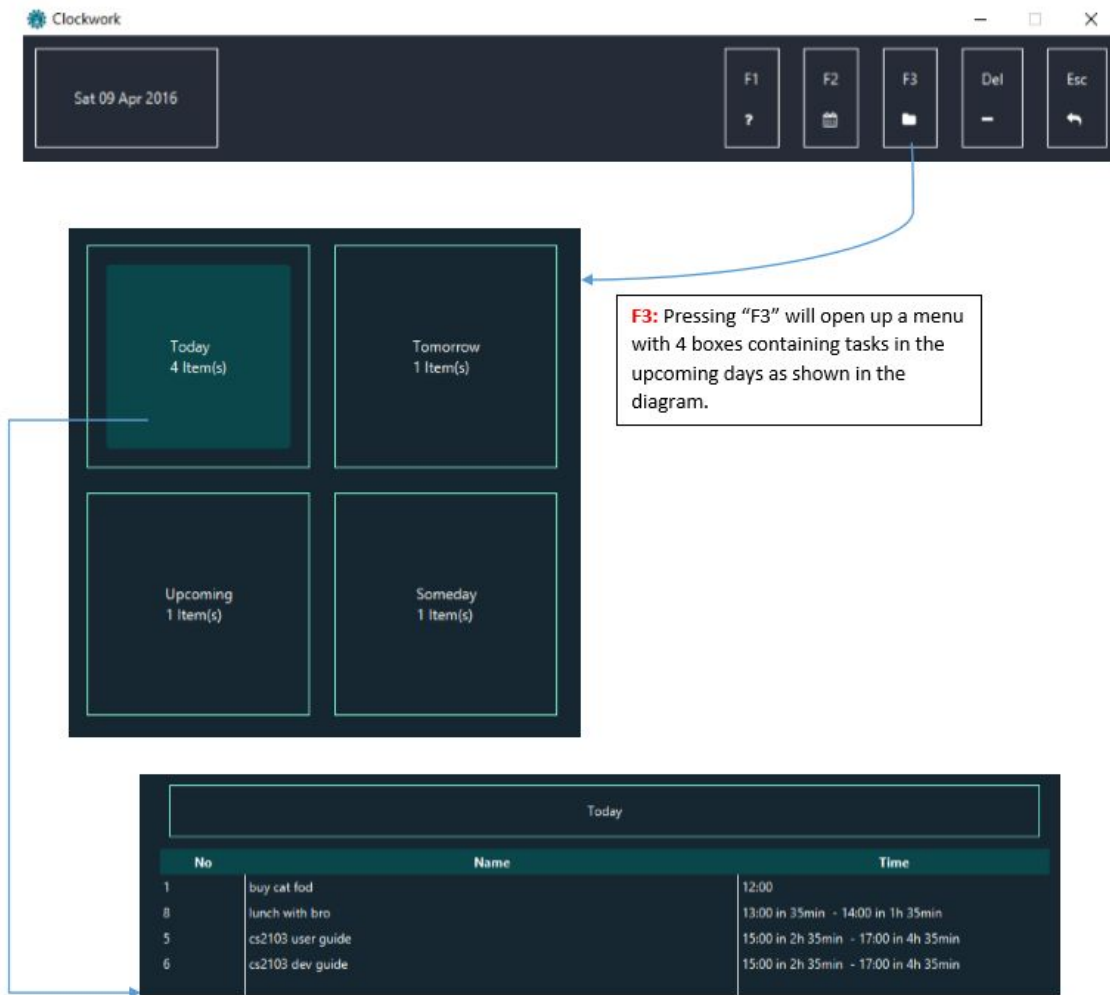


**F2:** Pressing the “F2” key allows the user to navigate to the calendar view



“←↑→↓” followed by “Enter”: From the calendar view, use the arrow keys to navigate through dates. Pressing the “Enter” key on a specified date shows the agenda view for that particular day.



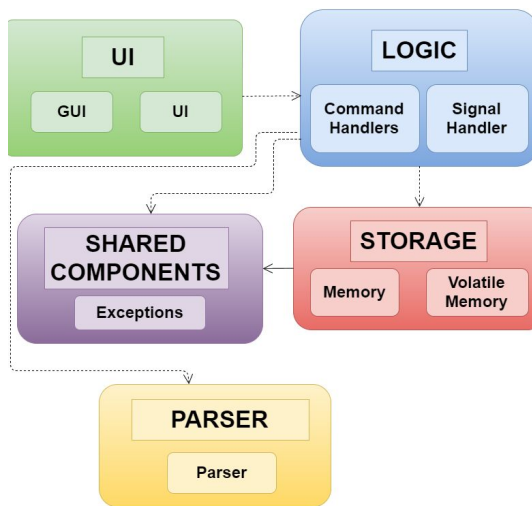


**"Tab"** followed by **"Enter"**: Use the "Tab" key to navigate between the 4 categories. Use the "Enter" key to select the category to display e.g. Today's tasks.

# ClockWork Developer Guide

## Architecture

At an extremely high level, *Figure 1.1* (below) describes the interaction between Clockwork's major components. Upon starting the program, the user may begin adding tasks using the CLI located at the bottom. User input is parsed into a series of key-value pairs intelligently, using the Parser class which employs a variety of third party libraries including 'Natty Date Parser' (dependencies described in *Credits* section) to convert strings into the abstract date type (ADT) 'Todo' or trigger some operation that interacts with the Storage or UI components. Todo objects represent the tasks users collect and manipulate. They hold the power to be a deadline, event, or floating task, with the ability to recur and display within the calendar and agenda views (carried out via the UI component).



Our Logic component serves the role of calling Parser through the Command Handler (serving as an effective controller) so that the string can be understood as either the translation of a payload into a Todo object or the triggering of some operation, such as a deletion or undo. Upon the former case, when the command provides a payload that is translated into an ADT, our Storage component serves the purpose of immediately storing it into live, persistent, and volatile memory. Our Storage component serves as the system-wide resource containers for live and persistent memory. Upon persistent storage routines, Memory translates

*Figure 1.1*

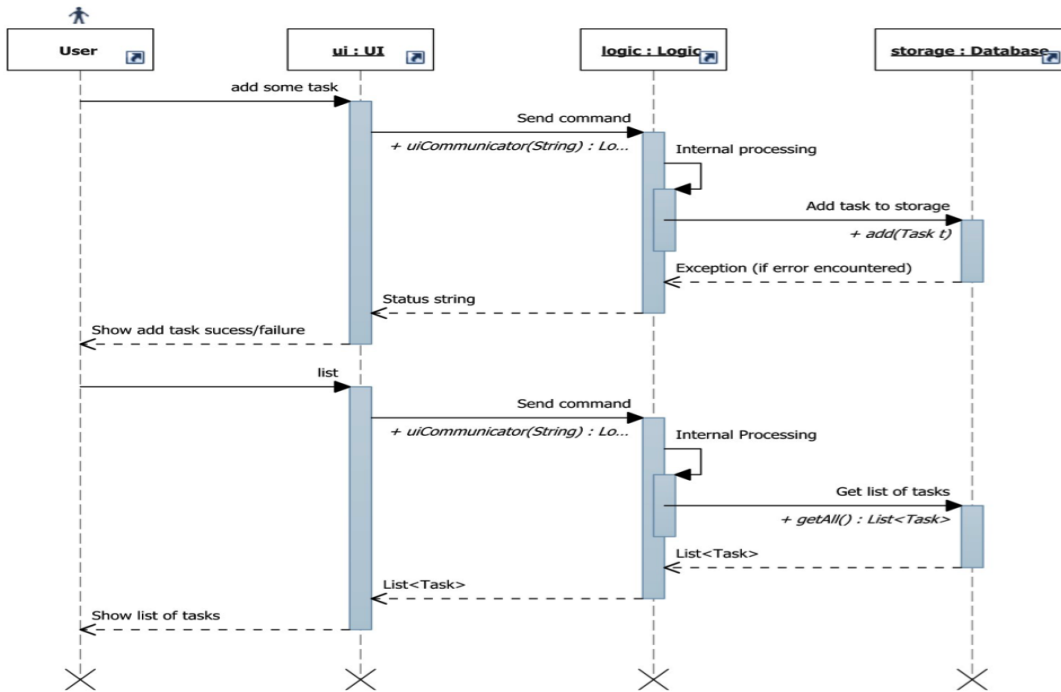


Figure 1.2

the ADTs to a JSON file that can be read in and out according to the standard JSON key delimiters. The JSON file, stored as 'storageFile.json' can also be easily read and edited by the user. Every event has a unique ID, which is associated with a start and end date, as well as many other attributes, in plain (unencrypted) English. Upon volatile memory routines, operations are stored on a stack. Any further undo commands will negate previous commands by popping from the stack. Redoing, in a similar manner, simply reverses the previous undo operation.

Some libraries we found to be useful throughout a variety of other components, such as customized exception classes and common helper methods. We bundled these classes together under a "Shared Component" so that they could be instantiated from any scope. A higher-level sequence diagram, which describes how the components progressively interact with each other through the execution of the program, is also visualized in *Figure 1.2*. Please note that this has been simplified to an overview, and further details of each independent component will be described in future sections.

## ***Component Design***

### **1. UI Component**

The UI component was developed with an emphasis in the MVC (Model-View-Controller) design. The view component, comprising of the Main class, is a container for model components like the Layout classes to pipe input in through the controller (via the input of user text). This way, the view and model will have to go through the controller for communication, living up to the standards of strict MVC. As the current flow holds, the model component comprising of the BoxInput class receives input from the user and updates the model accordingly, which in this case causes the UI display to change. We use the controller as an intermediary that takes the input from the view and communicates with our Logic component. Please note the Logic component has been deemphasized to leave further description within the next section.

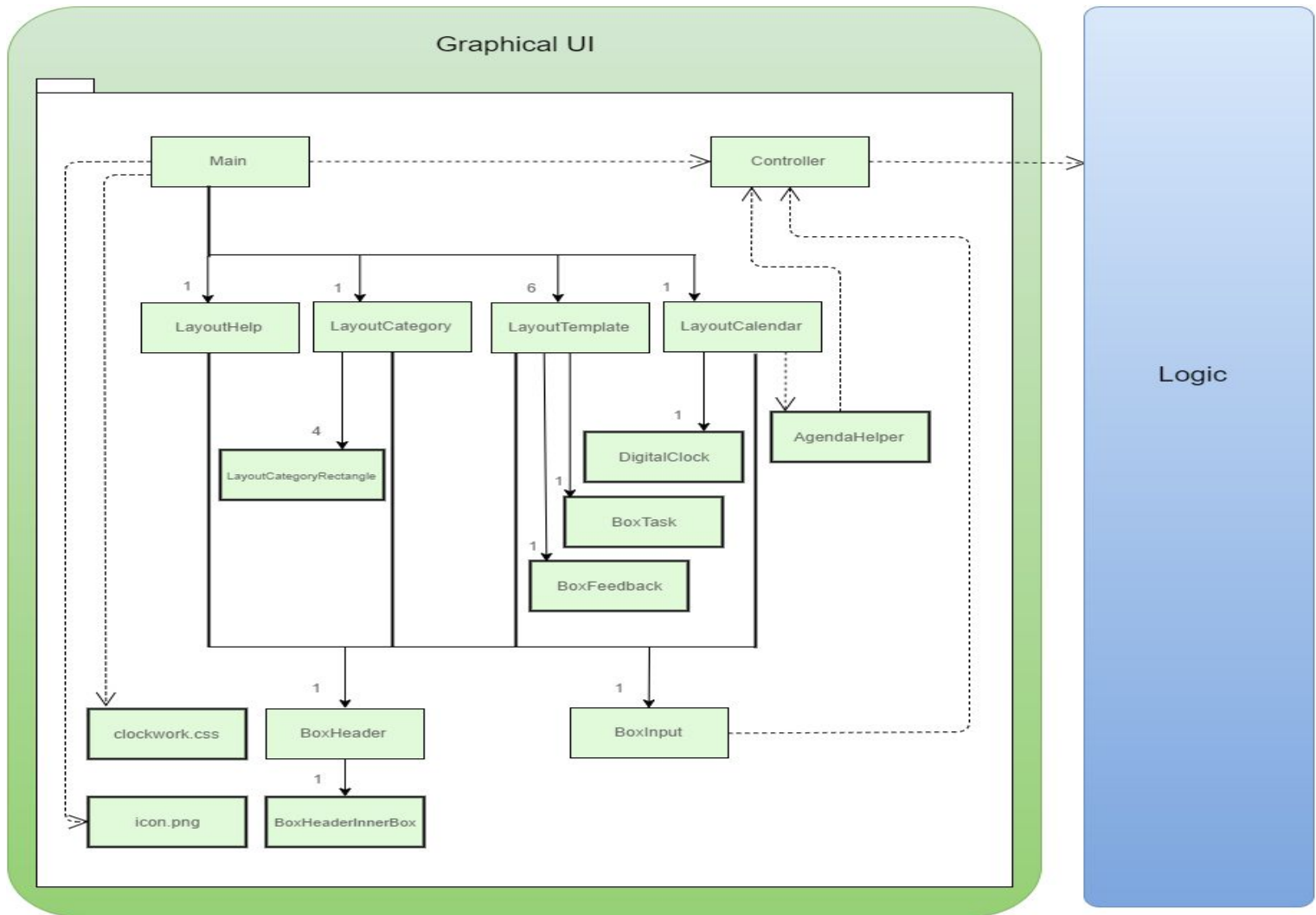


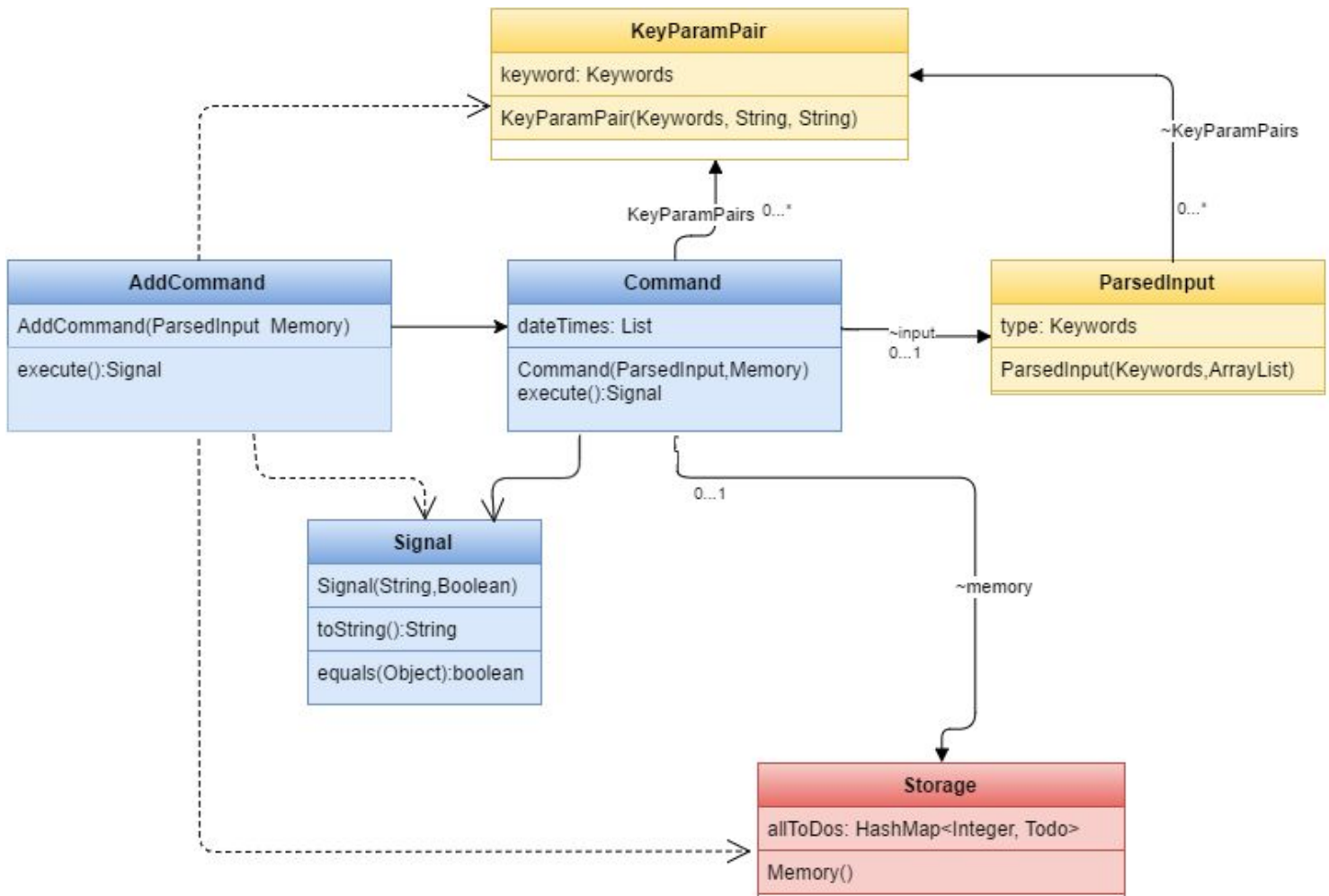
Figure 2.1.

## 2. The Logic Component

The logic component, melded with a variety of external libraries and novel in-house functionality, stands as Clockwork's secret sauce: providing the underlying foundation for the smooth and speedy user experience that offers a plethora of delightful features. *Figure 2.1* reveals the complexity of our network: many inner-working classes that create the Logic component's flow. After the UI component sends the user input through the UI Controller, Logic (Clockwork.java) receives a String and uses the Parser class to intelligently extract meaning from the String. Different substrings, after passing through the Handlers, will assimilate into different ADTs, such as CType (Command.java) or ToDoType (Todo.java). These ADTs, now with much more meaning and identification, are tagged with unique IDs and remain to be sent off into the Storage component (mainly in Memory.java) to live in live data structures (from ArrayLists to







*Class Diagram for AddCommand.*

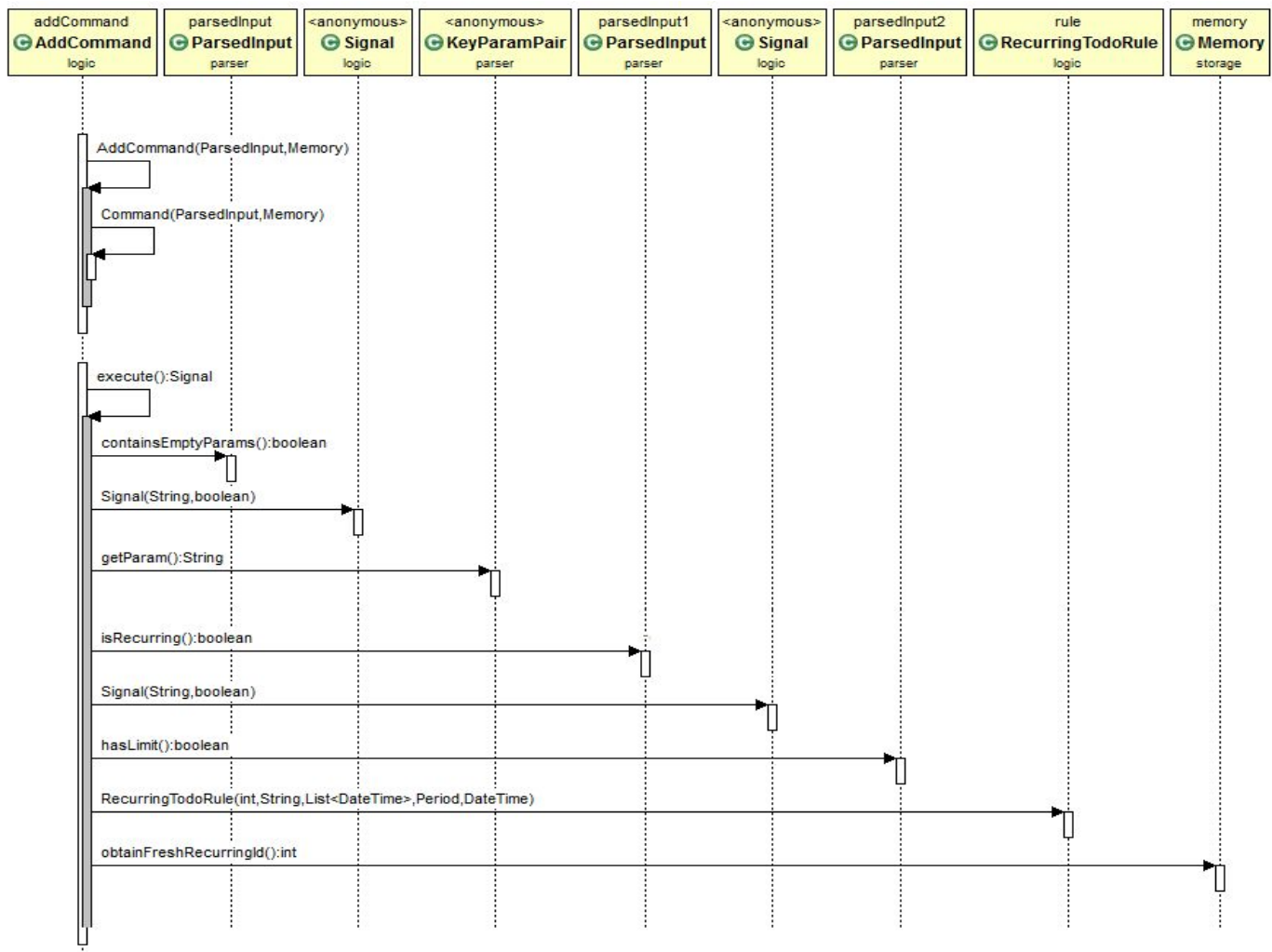
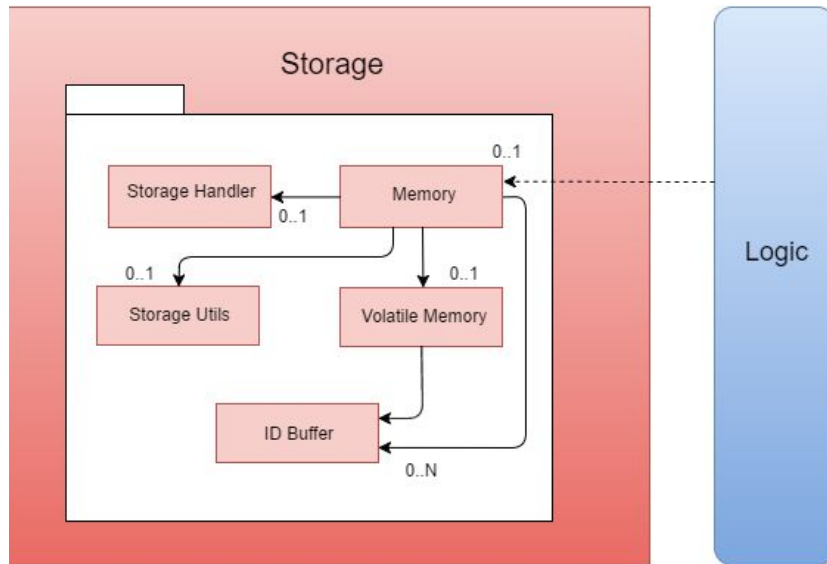


Figure 2.2

### 3. The Storage Component

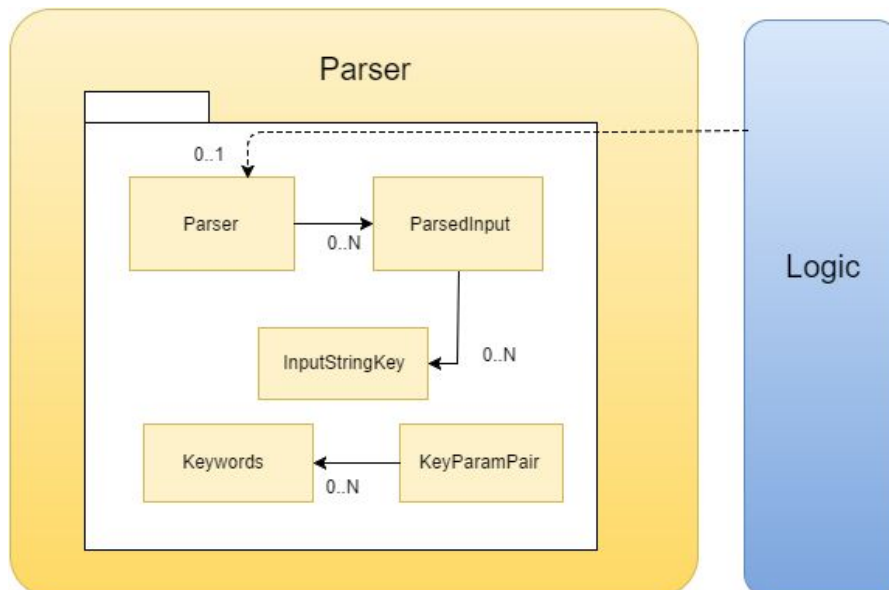
Clockwork's memory component serves as the solution to all information storage, including live, persistent, and volatile memory. The memory class keeps a slew of data structures that represent the live representation of Todo items. When 'Memory.java' is instantiated it reads from 'StorageFile.json' and begins to construct Array Lists, Binary Search Trees, and Hash Maps from the object notation. Then, the Logic component can consequently use the reference it has to memory to commit CRUD operations. 'VolatileMemory.java' serves as our solution for being able to undo and redo actions, Clockwork keeps a stack of most recent operations. Popping from the stack triggers the reverse of what was previously done. When a user is done with execution, the program can be terminated, where all the live data structures can then be saved in a user-friendly JSON format.

### Architecture Diagram for Storage



### 4. Parser Component

Parser helps Logic by doing its fair share of backend work. As shown by Figure 19 on the left, it consists of classes that are able to interpret user String inputs to create `LocalDate` or `LocalTime` objects. These objects may then be utilised by various classes from within the Parser component to create other objects, which will then be returned to Logic for further processing.



## 5. The Shared Component

The Shared component of Clockwork describes all the modules that have functionality useful outside of one particular class. For instance, custom Exception classes can be logically instantiated and thrown throughout any component or class. An in-house custom Search class serves useful for searching memory, commands that are visualized on the UI, and parsed operations in Logic that needs the identification of particular substrings.

## Testing

Testing is of utmost importance to us. We use JUnit to test every component in ClockWork. Every method is unit tested to ensure that everything works as intended. Equivalent partition of most test cases has been achieved, with extra effort to include an exhaustive collection of boundary cases. The JUnit codes used for testing are placed in the src/testcases folder. We recommend regression testing with the JUnit test files after every modification made to the program. In fact, some of the existing test files not only serve as unit tests of specific methods, but also qualify as integration tests to verify connections between backend components. If you discover a failed test case that has not been accounted for by the tests, feel free to append it to the relevant JUnit test file.

## Notable Features

### 1. Calendar and Agenda View

Clockwork allows for the illustrative representation of your data in a variety of ways. Arguably the most appealing representation would be seeing tasks through the Calendar and Agenda views. Clockwork allows the user to navigate through a Gregorian calendar and select a specific date to view an agenda with that day's tasks. Tasks are represented by colored spans within the agenda that show the start time, end time, and description. Clashing events within the agenda are able to utilize their space intelligently, splitting and overlapping with each other depending on the interaction of their times.

### 2. Recurring Task

Clockwork provides extensive support for recurring tasks. When adding a task, the user has to simply specify the keyword "every" followed by how often he/she would like the specified task to recur. These key words can take the form of "every day", "on Tuesday every week", "on 21 april every month" and " 22 april every year". The tasks are added starting from the current date and

are stored as reoccurring for a year. This recurrence is displayed on the default display as well as the agenda view.

### 3. Clash Detection and Alert

When editing or adding a new task, it's possible that a user could potentially overwrite or overlap an existing task in memory. Within Clockwork, 'ClashDetector.java' provides a convenient solution to this problem by detecting such potential events. Our clash detector has a suite of tools to scan memory or a potentially dangerous command to warn the user of unintended consequences.

### 4. Category Buckets

By pressing F3, users can easily browse through their tasks by category. The four category buckets, Today, Tomorrow, Upcoming and Someday, can be selected by pressing Tab and Enter to display the corresponding list. With a minimalistic design that displays only the category name and number of tasks in each category, the user can quickly check for the number of tasks left to do from a category point-of view.

## Appendix A: User stories

### *[Likely]*

ID	I can ... (i.e. Functionality)	so that I ... (i.e. Value)
addFloating	add a task by specifying task description only	can record tasks that I want to do some day
addWithTime	add a task by specifying task description and time only	can record tasks that I want to do today
addWithDate	add a task by specifying task description and date only	can record tasks that I want to do that day
addWithPriority	add a task by specifying task description and priority	can record tasks that I want to do first
add	add a task by specifying task description, priority, date and time	can see the tasks to do
flexibleCommand	key in commands by their other aliases as well	can type commands more naturally
see ID	see tasks by the task ID	can make changes to the task

editWithDescription	edit a task given only the task ID and description	can make changes to the task if I only remember the description
editWithDeadline	edit a task given only the task ID and deadline	can makes changes to the task if I only remember the deadline
editWithPriority	edit a task given only the task ID and priority	can makes changes to the task if I only remember the priority level
displayCompleted	see completed tasks	can know what tasks I have done
displayIncomplete	see incomplete tasks	can know what tasks I have yet to do
displayAll	see all tasks	can decide what task to do in the list
markTask	mark tasks as completed or incomplete	can categorise what I have done or not done
markAll	mark all tasks as completed	can easily categorise everything in a single command
removeTask	delete a task	can make up for any error I made in creating the task
setGeneric	set generic email replies	can personalize the way I want my generic message templates to read
sendGeneric	send generic email replies	can save time crafting email to messages that require standard replies
autofill	autofill commands	can save time typing commands
emailNotification	be notified of tasks by email	can be reminded of the tasks I need to do in another platform
setEmail	set the email address linked to my Google account	can directly send generic emails to a specified email address
quickAdd	get support for CalQuick Add command format	can better integrate the two platforms together
uploadToGcal	upload to do items to Calfrom >\$Clockwork	can see my tasks ongCalafter keying them in >\$Clockwork
getFromGcal	updateto do items from Calto >\$Clockwork	can see the tasks keyed in on Calon >\$Clockwork

**[Unlikely]**

<b>ID</b>	<b>I can ... (i.e. Functionality)</b>	<b>so that I ... (i.e. Value)</b>
integrateFacebook	integrate>\$Clockwork with my Facebook account	can share, like and comment on tasks with my social network
hideHotkey	have a hotkey to hide >\$Clockwork	can have more working space on my desktop while being able to easily check >\$Clockwork
exceedAlarm	have an alarm to notify me if I have set too many tasks in a day	know that I may be over planning too many activities in the day

**Appendix B: Non-Functional Requirements**

- The software should run on Windows 7 or later.
- The software able to perform even without network connections.
- Disaster recovery, the file can be saved to the disk after each user operation.
- The software should work stand-alone and work without requiring an installer.
- The software should have shorter running time.
- The software should be reliable and fault tolerant.
- The software should not violate other constraints.
- The software should be free for all users.

**Appendix C: Product survey**

<b>Product:</b> Doodle <b>Documented by:</b> Rajendran Premkumar
<b>Strengths:</b>
Once registered, connect the user's calendar and keep track of all the polls.
Offers a native app for all mobile devices.
Able to show the user's available date and how the user wants to be contacted.
Connect calendars which abletheuser to edit meeting requests directly in the calendar interface.
Can be used forBusiness or Enterprisewhich has addedfunctionality and theming possibilities.
Users can receive the schedule link to administer his/her poll at provided e-mail addresses.
Good user interface.
<b>Weaknesses:</b>
Lack of market exposure for mass usage.
Too many email threads.
Limited functionality.

Users have to pay to integrate with desktop schedulers.
---

Do not have functionality to prevent double-bookings.
---

<b>Product:</b> Time bridge <b>Documented by:</b> Xu Haoting (Regine)
---

<b>Strengths:</b>
-------------------

Once registered, connect user's calendar with Google or My Outlook.
---

Organizers can share a public calendar to their friends.
--

Invitees can select a preferred time.
---------------------------------------

Measures available to prevent double-bookings.
--

Users can customize events and share them in the calendar.
--

No spam.
----------

Excellent user interface.
---------------------------

Good market reach - more than 250000 people and businesses around the world use this product.
---

Phone conferencing and video conferencing functionality.
--

SMS feature for notes, attachments and reminders.
---

<b>Weaknesses:</b>
--------------------

Registration required for organizers.
---------------------------------------

Does not sync with Yahoo! Calendars.
--------------------------------------

No functionality to designate certain attendees as VIP or required.
---

Time glitch with the download for Outlook appointment.
--

Weak web conferencing functionality.
--------------------------------------

<b>Product:</b> Google Calendar <b>Documented by:</b> Rebekah Low Jin Yan
---

<b>Strengths:</b>
-------------------

Free for use, and users may connect to their Gmail friends
--

Organizer can share a public calendar to their friends and groups.
--

Public holiday is automatically marked.
---

Options available to make events public or private.
---

Users can customize events and share in the calendar.
---

Users can create multiple calendar for different groups.
--

Excellent user interface.
---------------------------

Fully cloud-enabled with offline tools and device synchronization.
--

Sync across each and every Google account-linked devices with Google Calendar
---

<b>Weaknesses:</b>
--------------------



Requires registration for organizers.
Compatibility issues with sending events to non-Google calendar users.
Calendar events cannot be added from non-primary Google email addresses.
GUI appear less elegant and usable on the iOS.

<b>Product:</b> Sunrise Calendar	<b>Documented by:</b> Morgan James Howell
<b>Strengths:</b>	
Easy-to-navigate user interface.	
Calendar will automatically be imported into the linked Facebook or Google accounts if they are used for registration.	
Users may view reminders and calendar entries which are imported from iCloud Calendar and Reminder on the same screen.	
Pioneering calendar app for Android and web devices.	
Allows user to see Google Maps previews for maps and directions and Facebook events integration.	
<b>Weaknesses:</b>	
Users cannot search for a specific entry in their agenda.	
Can go one year ahead of the present time.	
Month view does not show the days the user has an event.	
Widget only shows agenda view.	
No "quick add" for adding events by typing commands like "Dinner with John at 8pm".	