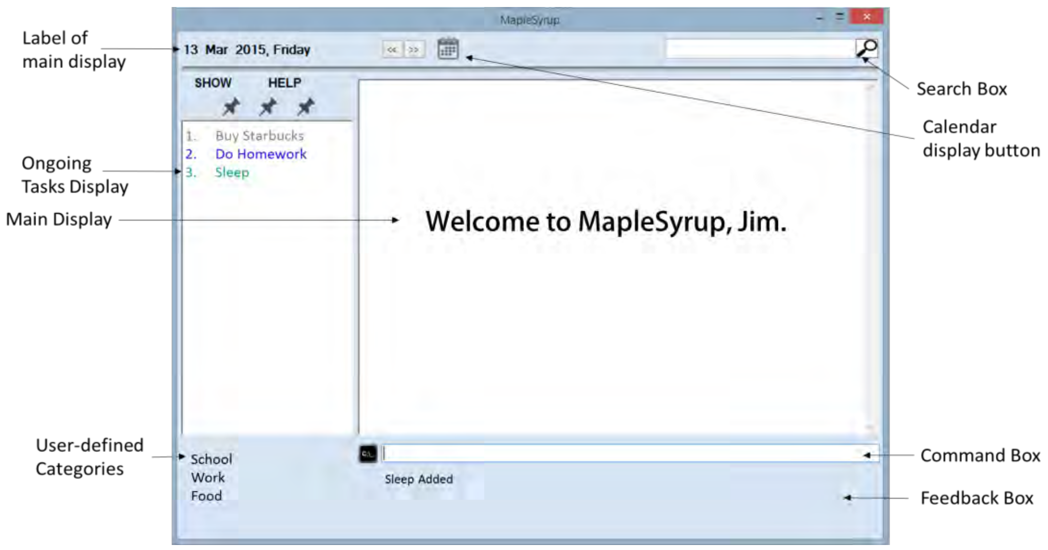
[f11-2c][V0.1]

MapleSyrup



Welcome to MapleSyrup, Jim.

Sleep Added

Supervisor: Lim Yu De Extra feature: GoodGUI

Ong Wei Jee	George Lam Changwei	Ter Yong Kang	Che Jian Yong Joshua
Team Lead	Github Expert	GUI Designer Deliverables and	Integration

Format bonus: Yes

Punctuality bonus: Yes

Overall comments: Good job with the diagrams. They are mostly concise, well done! However, a very bad trap this document has fell into is the one where diagrams are not complemented with textual explanations. Diagrams without textual explanations may either 1. convey the wrong idea to the reader because they decipher it themselves with no guidance, or 2. make the reader just skip the diagram because it is not easily understood/its diagram-drawer's intent is not clear/it is too long and the main point of the diagram is unclear. Also, try to ask the question: is this necessary? What is the point of including it? How does it help me explain what I want to convey to the reader?

I see a TON of effort put into making these awesome diagrams, but I hope you see why textual explanations are necessary. Great job nonetheless! 15 points. (With those textual explanations you could have gotten 16 :( ) Keep it up, and I hope to see your product during the demo looking good :)

Yu De

User guide

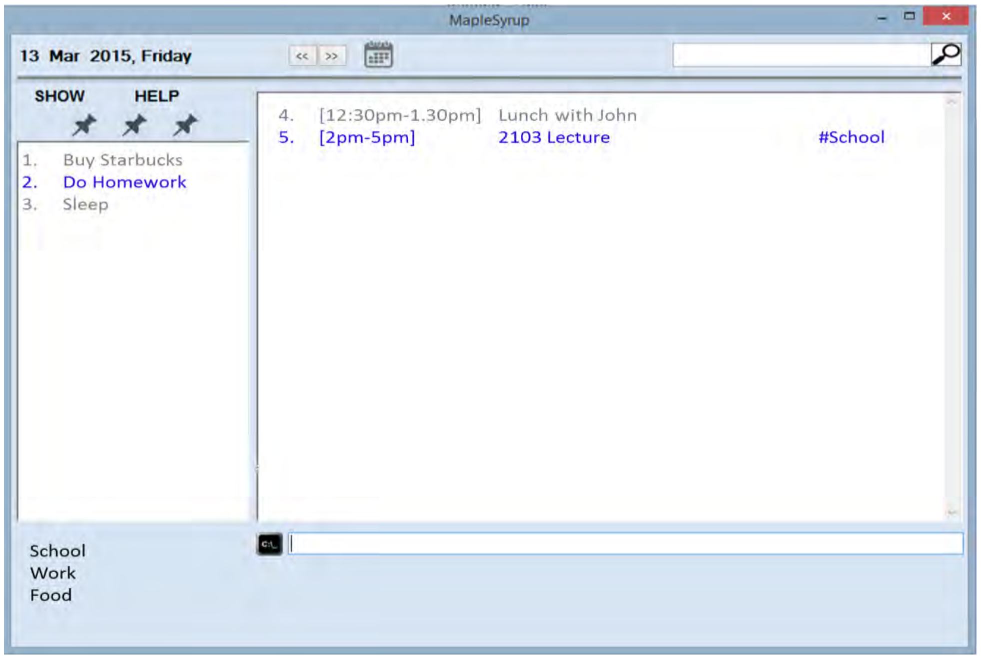
I will not grade this section and will leave it to you to revise it as you deem fit in the coming weeks.

Introduction

Welcome to MapleSyrup, your go to application to keep track of tasks and events. An event in MapleSyrup consists of several details: the name, date, time and additional descriptions. In this guide, we will walk through the process of adding, deleting, editing, displaying events, undoing and redoing.

Getting Started

1. Double click MapleSyrup on your desktop.
2. A welcome message will appear to greet you when the program opens.
3. The main display will show all your tasks and events for today. On-going tasks will be permanently displayed on the left column.

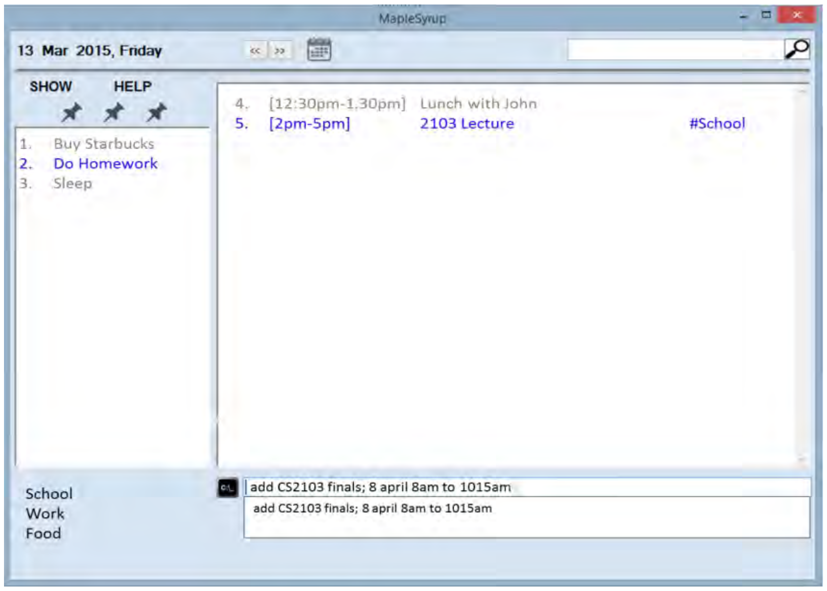


1. Buy Starbucks
2. Do Homework
3. Sleep

4. [12:30pm-1.30pm] Lunch with John
5. [2pm-5pm] 2103 Lecture #School

Functions

ADDING AN EVENT:



MapleSyrup

13 Mar 2015, Friday

SHOW HELP

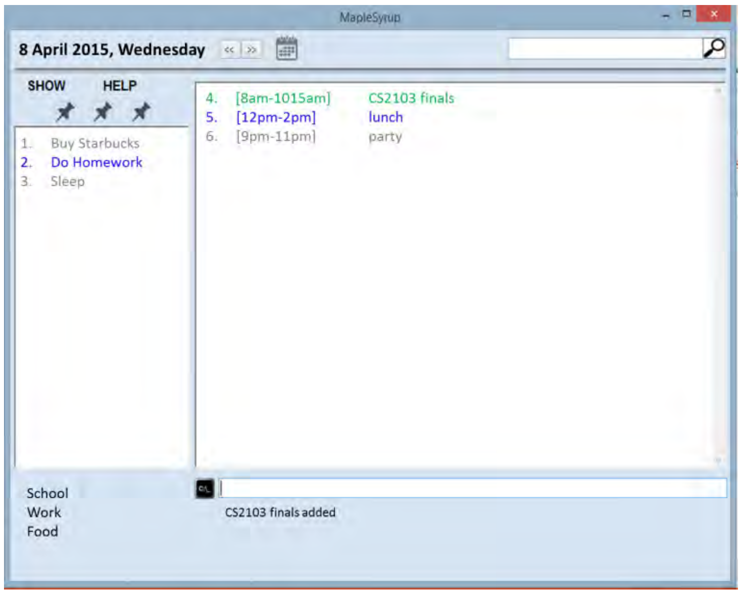
1. Buy Starbucks
2. Do Homework
3. Sleep

4. [12:30pm-1.30pm] Lunch with John
5. [2pm-5pm] 2103 Lecture #School

School
Work
Food

```
add CS2103 finals; 8 april 8am to 1015am
add CS2103 finals; 8 april 8am to 1015am
```

1. For single day events, type “add” followed by the “event name” and “;” followed by the time and duration of the event. For example, “add CS2103 finals; 8 april 8am to 1015am”.
2. A feedback box will display an acknowledgement “CS2103 finals added.”
3. In addition, the main display will show the newly added event in green with the rest of the events you have on that day, as seen in the picture below.
4. When typing a command, a suggestion box will appear to prompt correct input formats. There are a total of 9 types of events that can be added, depending on your event’s characteristic. Below are the 9 types with example.
   1 Event Name only (add Pick up laundry)
   2 Today Event with starting time only (add Go for a run; 9am)
   3 Today Event (add Dinner at Koufu; 8pm-10pm)
   4 Single Day Event with starting time (add Dental Appointment; 18 feb 4pm)
   5 Single Day Event (add Breakfast with Joshua; 8 feb 8am to 1015am)
   6 Full Day Event (add Beach Day; 20mar)
   7 Full Day Event spanning a few days (add Summer Camp; 15 march to 20 april)
   8 Event spanning a few days with starting time (add Queue for new Iphone; 7 dec to 19dec 10am)
   9 Event spanning a few days (add Holiday; 17 april to 19 april from 9.30pm to 11.45pm)



8 April 2015, Wednesday

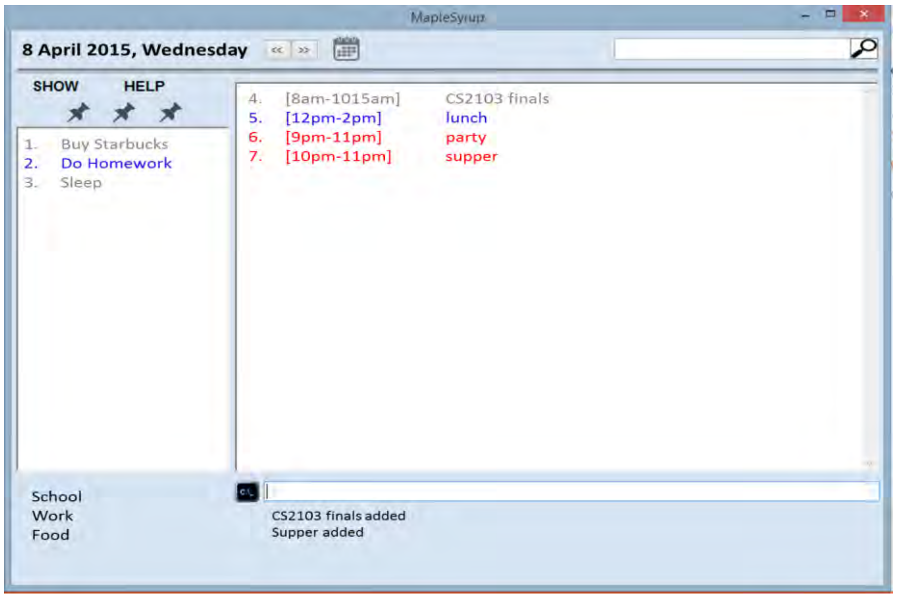
1. Buy Starbucks
2. Do Homework
3. Sleep

4. [8am-1015am] CS2103 finals
5. [12pm-2pm] lunch
6. [9pm-11pm] party

School
Work
Food

CS2103 finals added

5 If the new event clashes with an existing event, both clashed events (old and new) will be shown in red.



8 April 2015, Wednesday

1. Buy Starbucks
2. Do Homework
3. Sleep

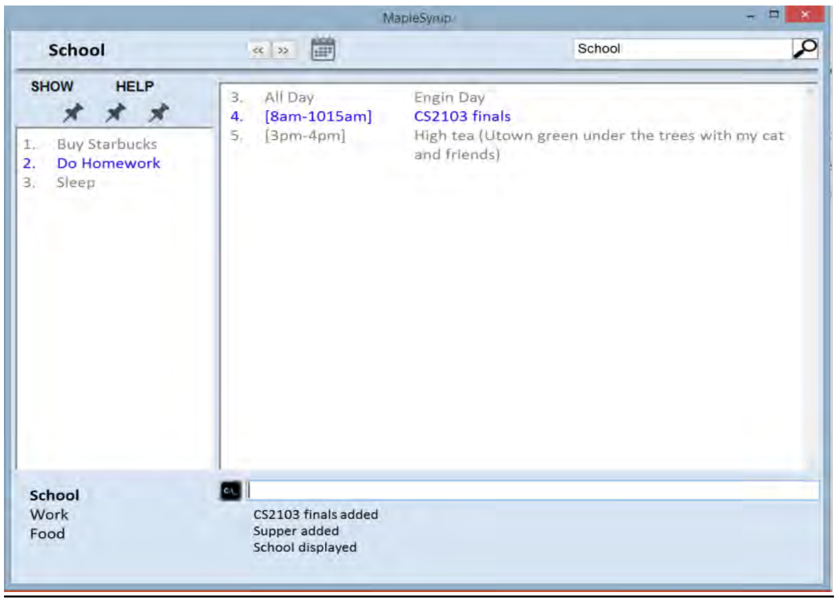
4. [8am-1015am] CS2103 finals
5. [12pm-2pm] lunch
6. [9pm-11pm] party
7. [10pm-11pm] supper

School
Work
Food

CS2103 finals added
Supper added

SEARCHING FOR AN EVENT

1. Type into the search bar at the top right of the window to search for the event. You can search for event name, date, description or tags.
2. The top left box will show the label of what is being displayed on the main display which you have just searched.
3. Search results will be presented on both the main display panel and sub panel.



```
1.  Buy Starbucks
2.  Do Homework
3.  Sleep
```

```
3.  All Day          Engin Day
4.  [8am-1015am]     CS2103 finals
5.  [3pm-4pm]        High tea (Utown green under the trees with my cat
                     and friends)
```

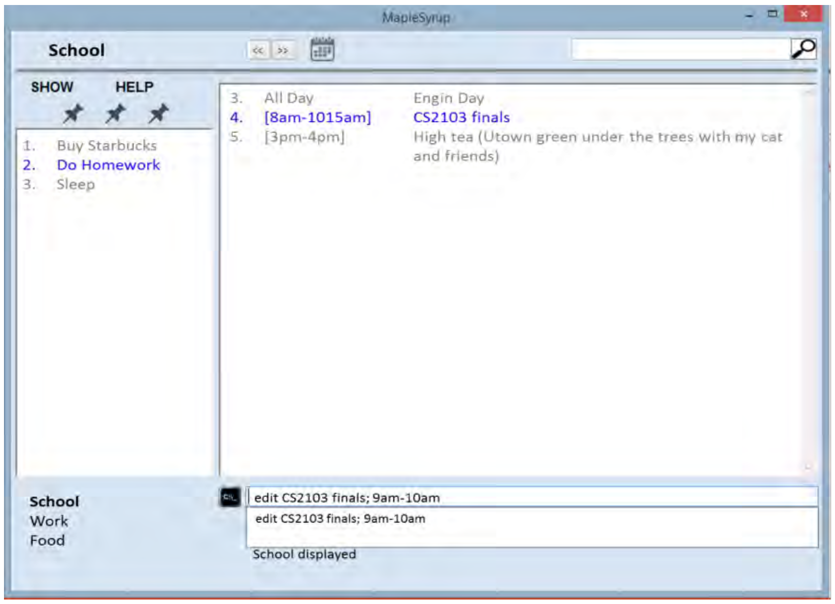
```
School
Work
Food
```

```
CS2103 finals added
Supper added
School displayed
```

EDITING AN EVENT

1. To edit and event, type "edit" and suggestion Box will appear to prompt the correct input format.
2. There are 2 main methods to edit an event:
   1 Edit from display's index.
   2 Edit with event name.
3. These are the recognized methods to edit, with examples in brackets.
   1 Edit from Index (edit 3 CS2103 V0.5)
   2 Edit Name (edit CS2103 finals; CS2103 V0.5)
   3 Edit Time: (edit CS2103 finals; 9am)
   4 Edit Dates: (edit Holiday; 21mar)

   - Multiple Edits
   5 Edit name, dates, time (edit Run; Jog; 10feb 9am-10am)
   6 Edit name, time (edit 1 Dinner at Koufu; 10pm)
   7 Edit name, date (edit Holiday; Bangkok!; 11feb to 15feb)
   8 Edit date, time (edit 3 10 feb 11.30am to 11 feb 11.30pm)

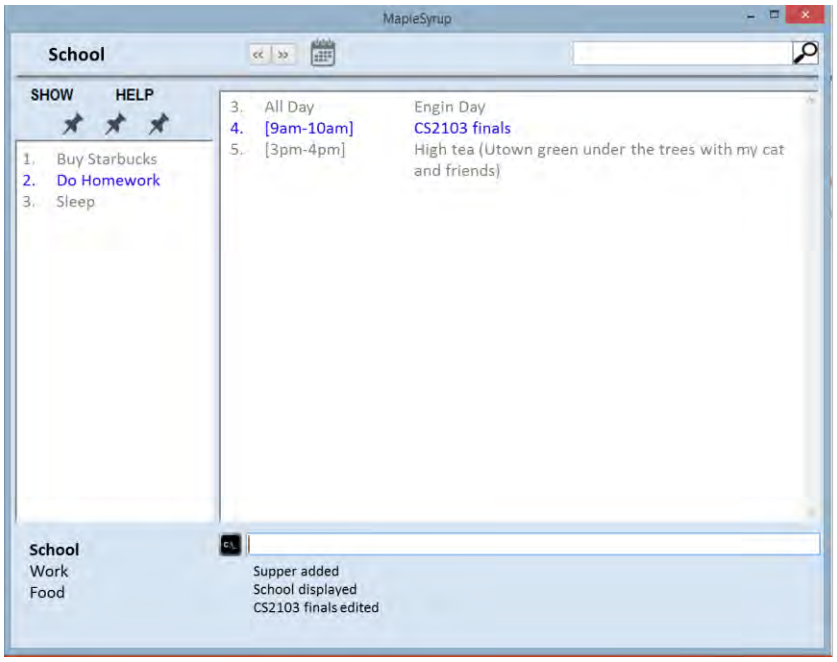


1. Buy Starbucks
2. Do Homework
3. Sleep

```
3.  All Day         Engin Day
4.  [8am-1015am]    CS2103 finals
5.  [3pm-4pm]       High tea (Utown green under the trees with my cat
                    and friends)
```

```
edit CS2103 finals; 9am-10am
edit CS2103 finals; 9am-10am
School displayed
```

4 Feedback box will display an acknowledgement "[Name of event] edited." As seen below.



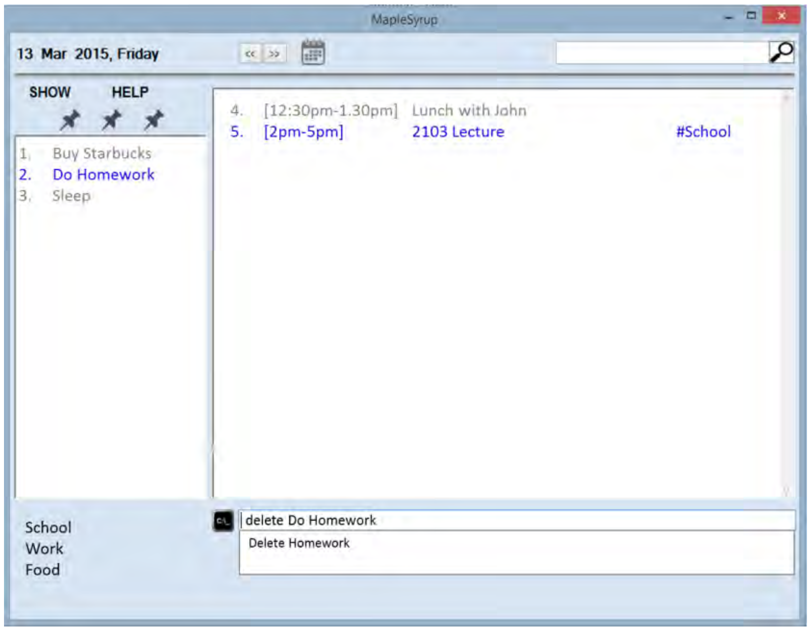
1. Buy Starbucks
2. Do Homework
3. Sleep

```
3.  All Day         Engin Day
4.  [9am-10am]      CS2103 finals
5.  [3pm-4pm]       High tea (Utown green under the trees with my cat
                    and friends)
```

```
Supper added
School displayed
CS2103 finals edited
```

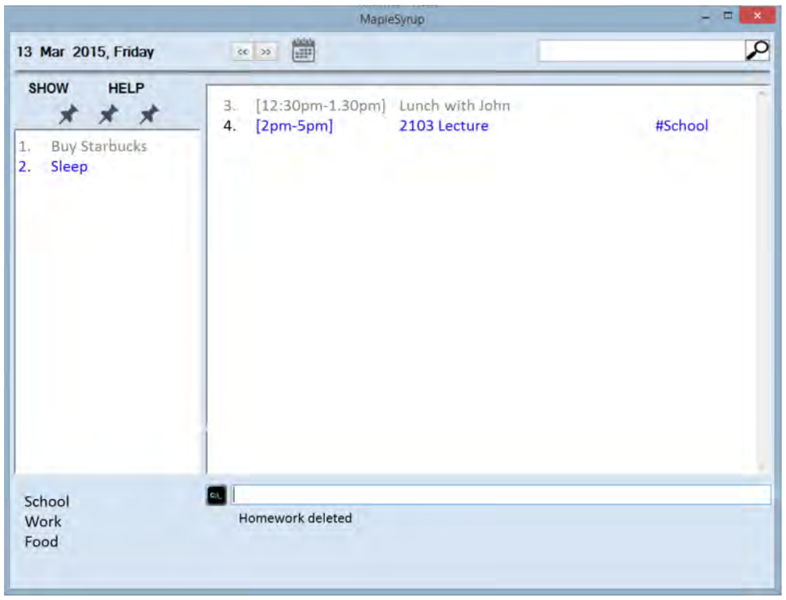
DELETING AN EVENT

1. Type 'delete' and suggestion Box will appear to prompt you the correct format of input.
2. There are 2 main methods to delete an event:
    1. Delete from display's index (delete 5).
    2. Delete with event name (delete 2013 Lecture).



```
delete Do Homework
Delete Homework
```

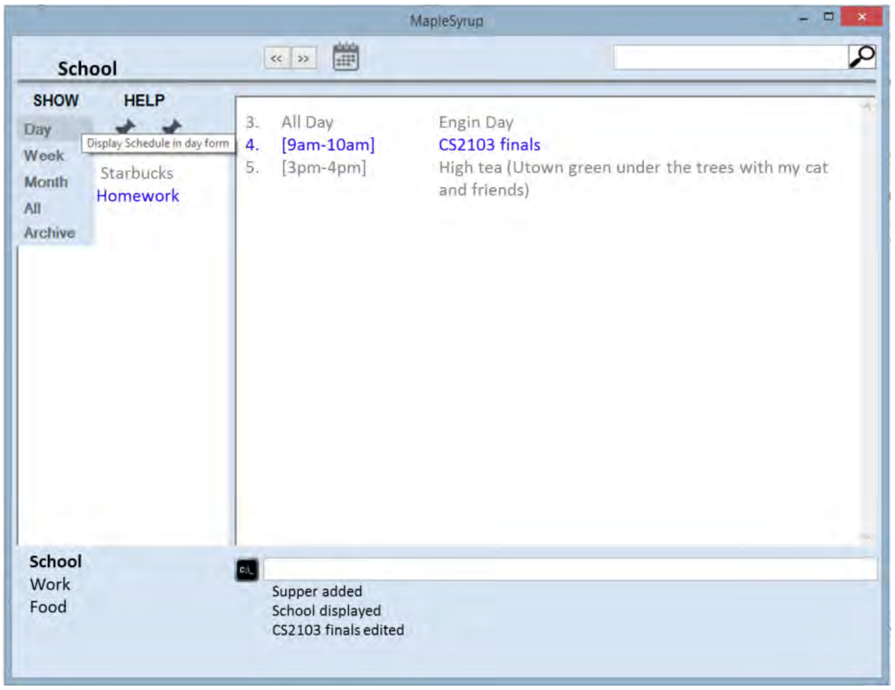
3. Feedback box will display an acknowledgement "2013 Lecture deleted." As seen below.



```
Homework deleted
```

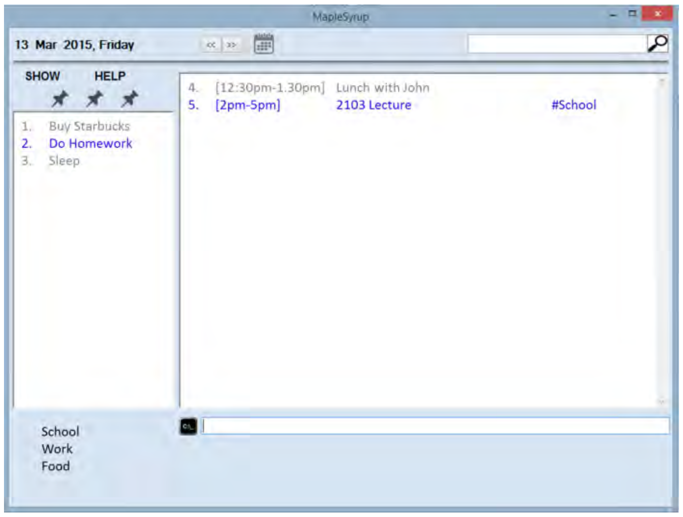
DISPLAYING EVENTS (BY SYSTEM DEFAULT CATEGORIES)- MONTH, WEEK, TODAY

To display a preset category, “Today”,”Week”,”Month”, move your cursor over to “SHOW” icon and press on the desired category. Alternatively, type “show today”, “show week”, “show month” into the command bar.



3. All Day Engin Day
4. [9am-10am] CS2103 finals
5. [3pm-4pm] High tea (Utown green under the trees with my cat and friends)

```
Supper added
School displayed
CS2103 finals edited
```



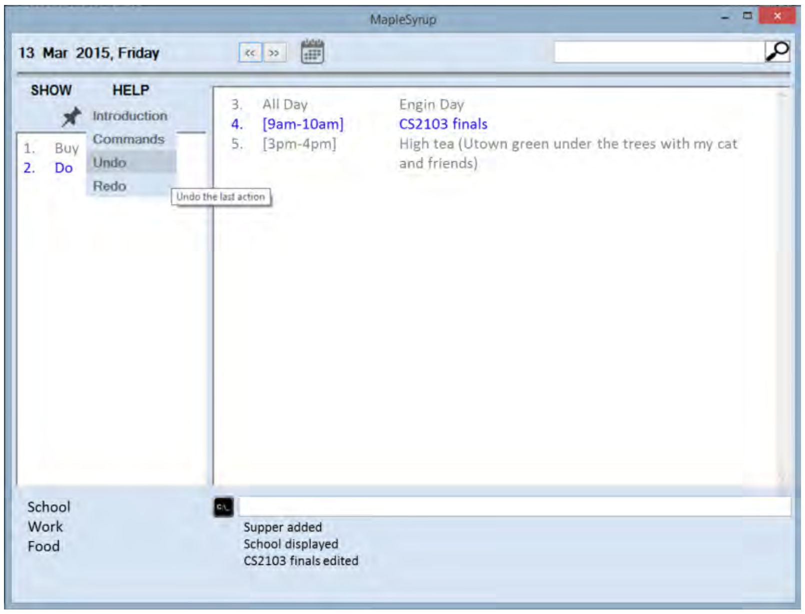
1. Buy Starbucks
2. Do Homework
3. Sleep

4. [12:30pm-1.30pm] Lunch with John
5. [2pm-5pm] 2103 Lecture #School

UNDO THE LAST ACTION

1. Simple move your cursor over to "HELP" and click 'Undo'

2. Alternatively, you can also type "undo" on the command bar.

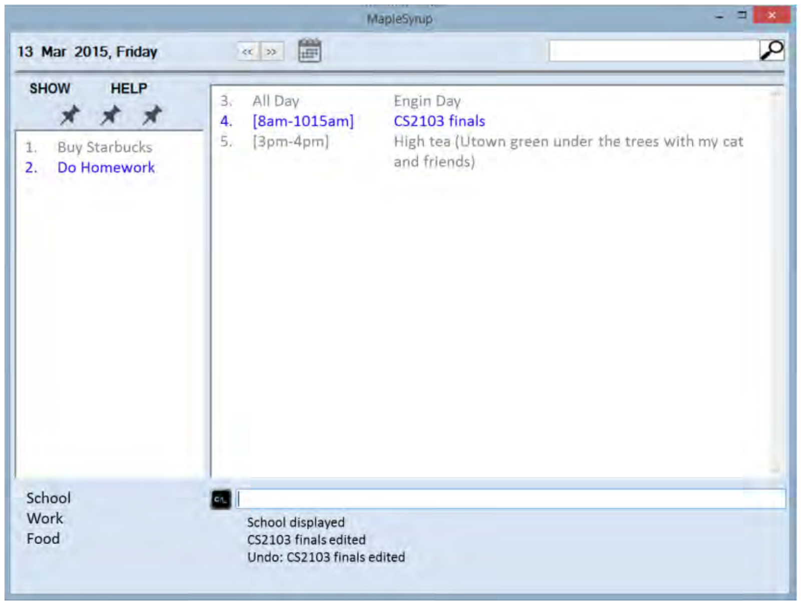


1. Buy
2. Do

3. All Day — Engin Day
4. [9am-10am] — CS2103 finals
5. [3pm-4pm] — High tea (Utown green under the trees with my cat and friends)

Supper added
School displayed
CS2103 finals edited

3. The feedback bar will display an acknowledgement statement as seen below.



1. Buy Starbucks
2. Do Homework

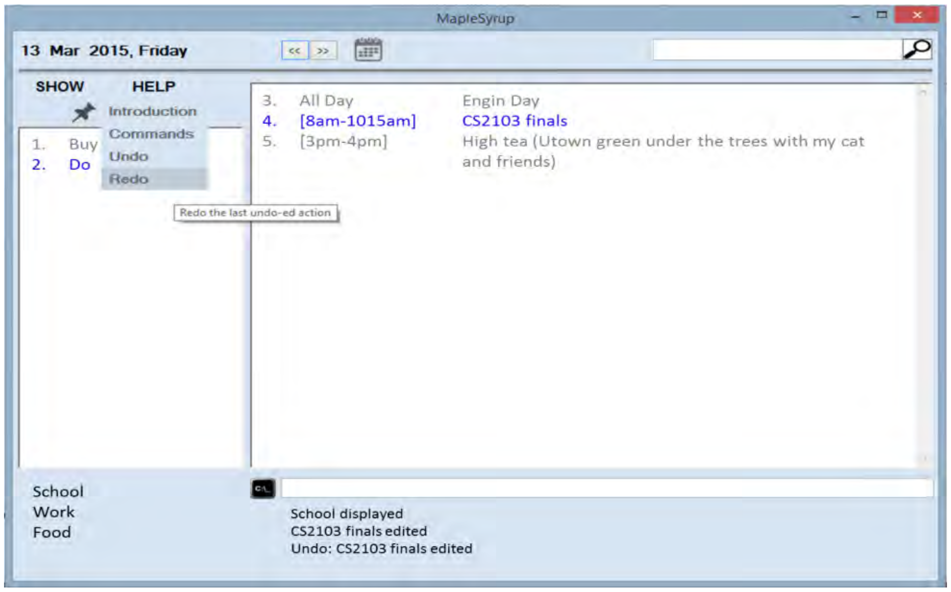
3. All Day — Engin Day
4. [8am-1015am] — CS2103 finals
5. [3pm-4pm] — High tea (Utown green under the trees with my cat and friends)

School displayed
CS2103 finals edited
Undo: CS2103 finals edited

REDO THE LAST UNDO-ED ACTION

1. Simple move your mouse over to "HELP" and press 'Redo

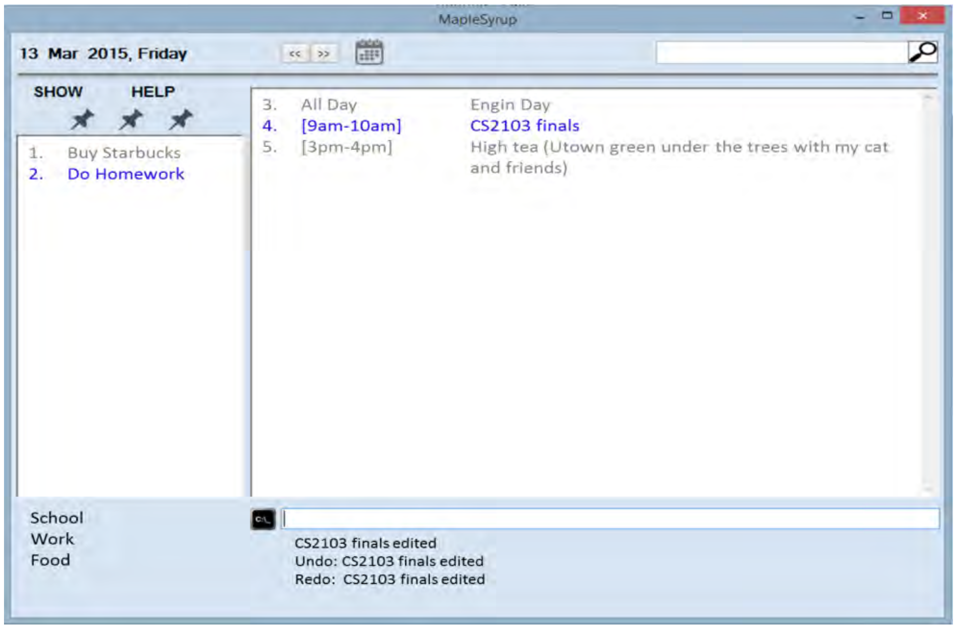
2. Alternatively, you can also type "Redo" on the command bar.



3. All Day Engin Day
4. [8am-1015am] CS2103 finals
5. [3pm-4pm] High tea (Utown green under the trees with my cat and friends)

School displayed
CS2103 finals edited
Undo: CS2103 finals edited

3. The feedback bar will display an acknowledgement statement as seen below.



3. All Day Engin Day
4. [9am-10am] CS2103 finals
5. [3pm-4pm] High tea (Utown green under the trees with my cat and friends)

CS2103 finals edited
Undo: CS2103 finals edited
Redo: CS2103 finals edited

Appendix A: User stories. As a user, …

[Likely]

ID	I can … (i.e. Functionality)	so that I … (i.e. Value)
addFloating	add a task by specifying a task description only	can record tasks that I want to do some day
addLocation	Add location to an event	Will know where I need to be for that event
setRecurring	Set events to be recurring every day/week/month	Do not have to replicate similar events that occur every day/week/month
anyFormat	Enter new task in any format	Do not have to follow strictly to a certain format
setCategories	Organize my events/tasks by categories through tagging	Can search for any event/task easily
addAllDay	Add all day events	Do not have to specify time period spanning the whole day
setImportance	Rank my task by importance	Can sort tasks by importance and perform more important task first
viewAll	View all my task/events in an entire calendar	Can see how busy the week/month is going to be
setAbbreviations	Type in commands in the form of abbreviations	Save time by reducing amount of typing
setKeyboardCtrl	Open/close the software with keyboard commands	Have a quicker and simpler way of opening and closing the software
viewFreeSlots	View all free slots available	Better
setAutoSave	Save my inputs after every command	No loss of information if software accidentally closes
viewCompleted	View all completed tasks	Have a sense of accomplishment
reportClashing	Notified if there is any clashing	Reschedule events/tasks to not have

	events/tasks	any clash
allowEdit	Edit my task information	Change any information at any time

[Unlikely]

ID	I can … (i.e. Functionality)	so that I … (i.e. Value)
addAlarm	Alarm/reminder when event is coming up	Will not miss important tasks
autoReschedule	Let the software auto reschedule my task if I fail to complete it during the set time	Do not need to set a new deadline again
setBufferTime	Set buffer time between events at different locations	Do not have a situation where I do not have time to travel from one event to the other
setAutoStart	Have the software auto start up when I turn on my computer	Can use the software immediately and be reminded of today's event right away

Appendix B: Product survey

Product: Google Calendar **Documented by**: Ter Yong Kang

Strengths:

1. Event colour - allow user to classify events based on colour. This enhance the user experience.

2. Same event can be repeated more than 1 time.

3. Find time function - helps user to find a free time for the event to be added

4. Search function - allows user to search for events according to its details (name etc. etc.)

5. Display density allows user to choose the style of displaying calendar - enhances user experience

6. Displays only the important details of the event (Name & Time) on the summary page. Upon clicking the event, the full details of the event are being displayed. This provides a quick overview for the user without information overloading he/she.

7. A "help" section to provide a quick guide to new users on how to use the program

Weaknesses:

1. Unable to help user prioritise which event is most important given they overlaps

2. Unable to prioritise to user what event/activity to be complicated first

3. Does not allow user to add "floating tasks" without entering date and time

4. Unable to categorise events of the same type (e.g. work, family) together (except through colour)

5. Only able to display in Day Week Month 4Day. Should have more display options such as "display event according to priority" or "events of same type" etc.

Product: Macbook Calendar **Documented by**: George Lam Changwei

Strengths:

1. Able to add new events easily with detailed information such as location, day and time

2. Able to set alarms and reminder for to do events

3. Able to add invitees, notes, URL, attachments

4. Able to tag an item to sort by different categories.

5. Can sync to Iphone

Weaknesses:

1. Only on mac platform

2. Unable to recognise shortforms for users that prefers to type fast

Product: Wunderlist **Documented by**: Che Jian Yong Joshua

Strengths:

1. Able to set due date, and has options for recurring events daily, weekly, monthly, yearly, and even custom (every 2 days, or every 5 days)

2. Able to add additional details to the event

3. Able to star events to label as high priority.

4. Able to search for events based on event title as well as additional info typed on the event

5. Has sorting availability by due dates, and priority.

6. Has automatic grouping of events, by today and week.

7. Allows attaching of file to the event, such as emails or documents that contain additional information on the current event can be linked to

8. Has shortcuts available for quick addition, starring, completing, selecting

9. Keeps records of completed tasks for reference in the future if need be

Weaknesses:

1. Does not allow copying and pasting of events. Not able to set multiple due dates that does not follow a standard recurring structure.

2. Does not allow previewing of all the events in the form of a calendar for easier viewing. (Such as seeing how busy a certain week is)

3. Front page of website does not show all the events. Unable to set Today tab as the first tab viewed.

4. Events for the week cannot be sorted, fixed by due dates.

Product: Sony Calendar **Documented by**: Ong Wei Jee

Strengths:

1. Can view in day/week/month

2. Option to input location and detailed event description

3. Can specific exact time for events

4. Can set repetition for events

Weaknesses:

1. Setting time by scrolling through numbers is not efficient

2. Reminders only occur 10 minutes before the event

3. Must navigate to respective dates to view events, and it is not always obvious in the GUI

MapleSyrup Developer Guide

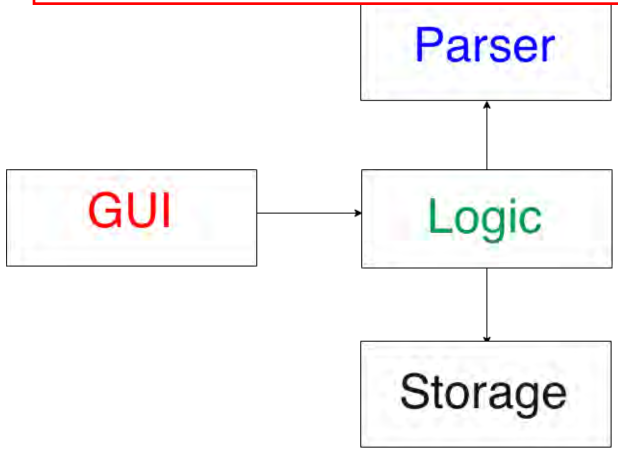
Table of Contents

1. Overview of Architecture 1
2. GUI 3
    1. UI
    2. CommandSuggestion
3. Logic 4
    1. Logic
    2. Display
4. Parser 5
    1. Parser
    2. InputStringSplit
    3. ParserProcessor
5. Storage 6
    1. EventStorage
    2. Search
    3. EventArchive
6. Global Classes 7
    1. Conversion
    2. Event
7. Sequence Diagrams 7
    1. Add 7
    2. Delete 8
    3. Edit 9
8. Important APIs 10
    1. GUI
    2. Logic
    3. Parser
    4. InputStringSplit
    5. ParserProcessor
    6. EventStorage
9. Appendix 11
    1. Sequence Diagrams
    2. APIs

To show that GUI is thin and is the gateway to the other components, draw it like a thin wall on the left of each other component, with its height 'shielding' them away. Using space wisely can make a lot of difference in diagramming. Also, Logic can be made bigger than Parser and maybe slightly bigger than Storage to show the size of the expected codebase.

1. Overview of Architecture

MapleSyrup utilizes a transaction processing architecture style with 4 components. They are the GUI, which is in charge of all user displays and interactions; Logic, which is responsible for the internal processing of commands; Parser, which is responsible for translating user input into understandable program commands; and Storage, which handles the reading and writing of the internal data to savefiles. All tasks/events entered by the user are saved as Event objects which may be passed between components (except GUI). This is illustrated in the overall architecture diagram on the right.



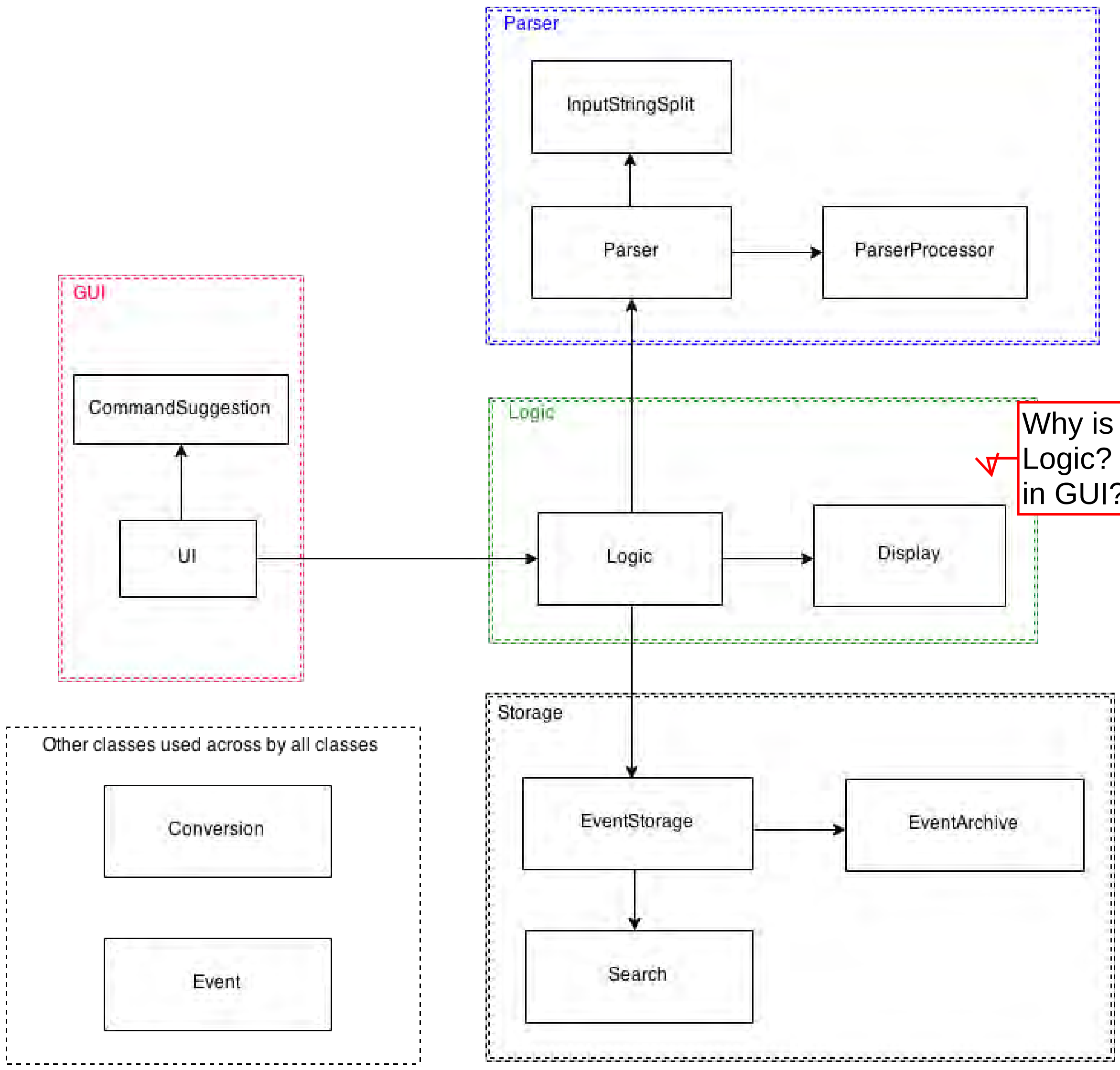
GUI: Users interact only with the GUI mainly in the form of text input, though there are certain buttons that provide alternatives to text input. Text typed in the command bar will be detected and the appropriate command suggestion will be received from CommandSuggestion class, and displayed to the user. The GUI captures user input and conveys it to Logic through a one-point contact, and receives a cue from Logic through this same point. It moves on to obtain the specific data from Logic based on this cue and displays them to the user.

Logic: This acts as a facade shielding the more complex implementations from the GUI. It comprises the Logic class which interacts with Parser and Storage to implement the actual user command, and the Display class which stores all output required to be displayed by the GUI.

Parser: Parser translates raw user input into commands intelligible by the program by converting inputs into Event objects. This is done by passing inputs through classes ParserProcessor and InputStringSplit. Generally, Parser is able to track an Event and a command (which are part of its attributes). Logic gets this information through Parser's API, then uses it to call the APIs from Storage to execute the desired command.

Storage: In Storage, the class EventStorage keeps track of a vector of Events that have been entered by the user but not deleted. All actual adding, deleting and editing of Events is implemented by manipulating this vector. The information in the vector is written to a text file after every command is executed, and read from the text file whenever the program is started. The EventArchive class stores similar information but with additional information for command type. This is to facilitate undo operations. Finally, the Search class implements various search functions required by EventStorage to find certain Events.

Detailed Architecture Diagram



Why is display in Logic? Shouldn't it be in GUI?

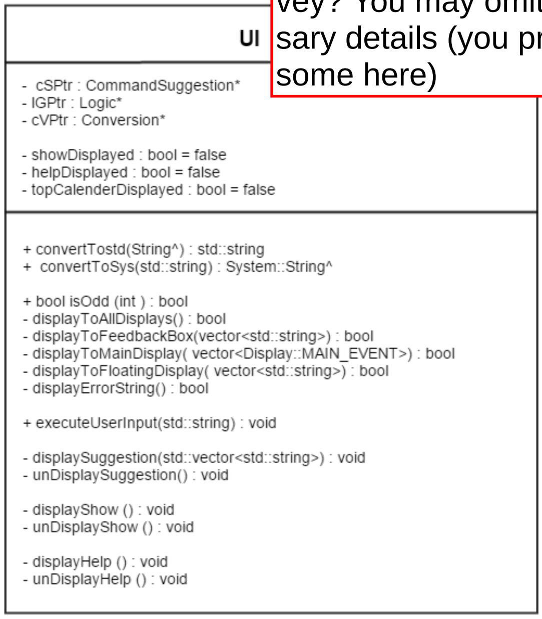
Diagrams should be complemented with textual explanations. At the very least, the purpose of the diagram's existence should be explained. Better, the main ideas that should be grasped from the diagrams. (Same comment for all the diagrams below with no textual explanation...)

2. GUI

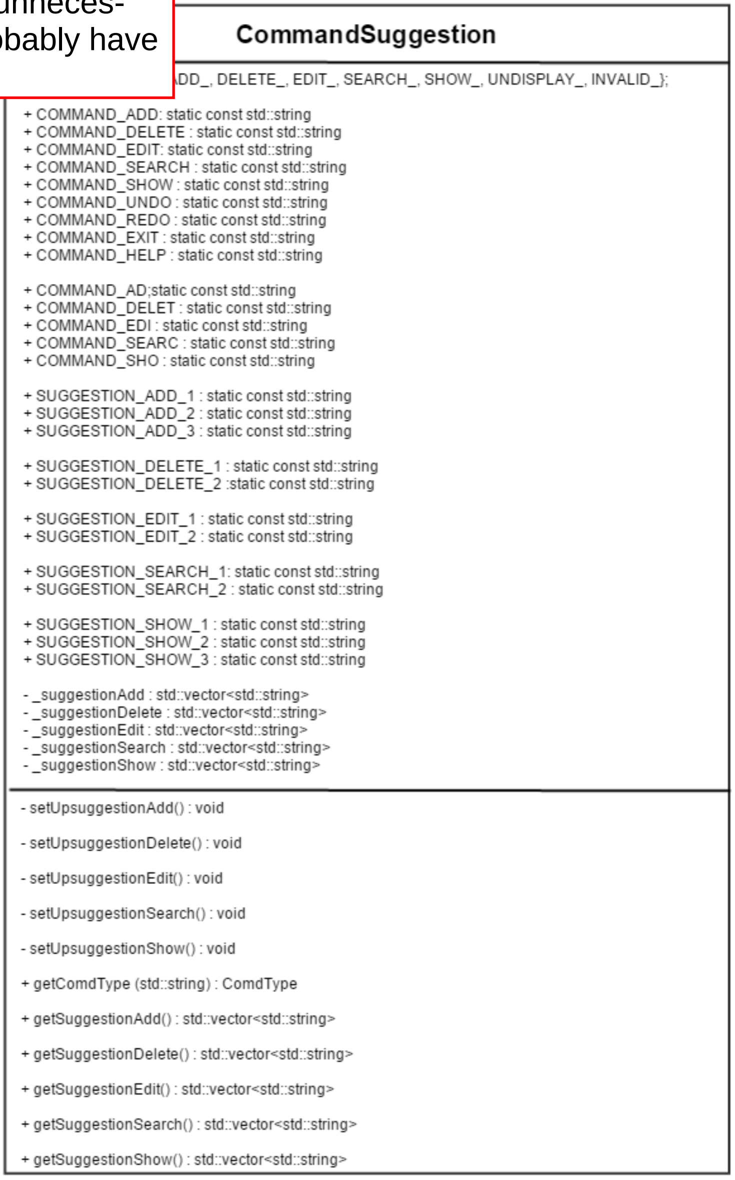
Is it necessary to show everything here? What is/are the main idea(s) you wish to convey? You may omit unnecessary details (you probably have some here)

2.1 UI

dSuggestion



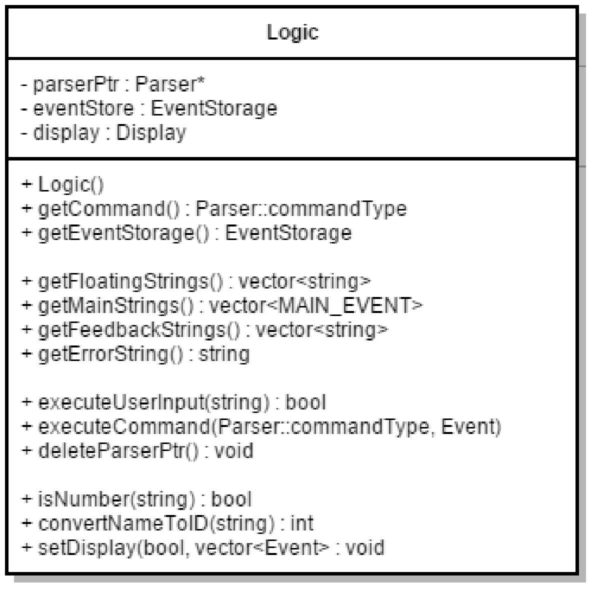
```
UI

-  cSPtr : CommandSuggestion*
- lGPtr : Logic*
- cVPtr : Conversion*

- showDisplayed : bool = false
- helpDisplayed : bool = false
- topCalenderDisplayed : bool = false

+ convertTostd(String^) : std::string
+  convertToSys(std::string) : System::String^

+ bool isOdd (int ) : bool
- displayToAllDisplays() : bool
- displayToFeedbackBox(vector<std::string>) : bool
- displayToMainDisplay( vector<Display::MAIN_EVENT>) : bool
- displayToFloatingDisplay( vector<std::string>) : bool
- displayErrorString() : bool

+ executeUserInput(std::string) : void

- displaySuggestion(std::vector<std::string>) : void
- unDisplaySuggestion() : void

- displayShow () : void
- unDisplayShow () : void

- displayHelp () : void
- unDisplayHelp () : void
```



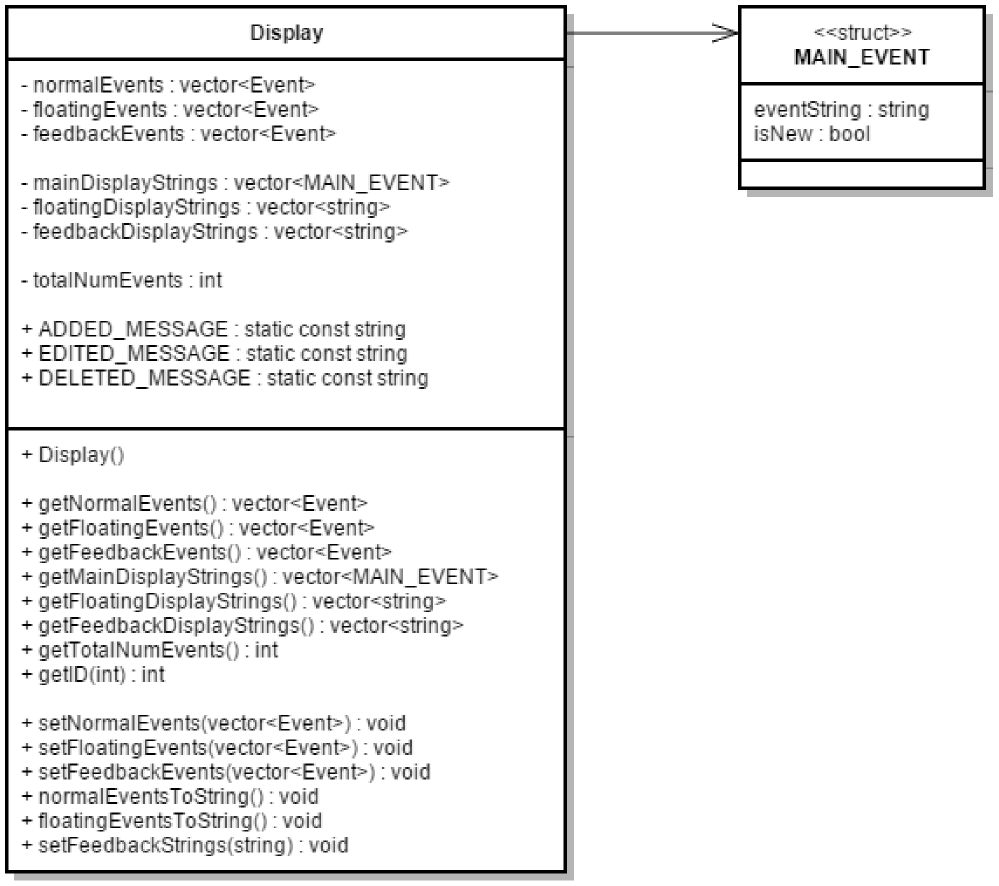
```
CommandSuggestion

DD_, DELETE_, EDIT_, SEARCH_, SHOW_, UNDISPLAY_, INVALID_};

+ COMMAND_ADD: static const std::string
+ COMMAND_DELETE : static const std::string
+ COMMAND_EDIT: static const std::string
+ COMMAND_SEARCH : static const std::string
+ COMMAND_SHOW : static const std::string
+ COMMAND_UNDO : static const std::string
+ COMMAND_REDO : static const std::string
+ COMMAND_EXIT : static const std::string
+ COMMAND_HELP : static const std::string

+ COMMAND_AD;static const std::string
+ COMMAND_DELET : static const std::string
+ COMMAND_EDI : static const std::string
+ COMMAND_SEARC : static const std::string
+ COMMAND_SHO : static const std::string

+ SUGGESTION_ADD_1 : static const std::string
+ SUGGESTION_ADD_2 : static const std::string
+ SUGGESTION_ADD_3 : static const std::string

+ SUGGESTION_DELETE_1 : static const std::string
+ SUGGESTION_DELETE_2 :static const std::string

+ SUGGESTION_EDIT_1 : static const std::string
+ SUGGESTION_EDIT_2 : static const std::string

+ SUGGESTION_SEARCH_1: static const std::string
+ SUGGESTION_SEARCH_2 : static const std::string

+ SUGGESTION_SHOW_1 : static const std::string
+ SUGGESTION_SHOW_2 : static const std::string
+ SUGGESTION_SHOW_3 : static const std::string

- _suggestionAdd : std::vector<std::string>
- _suggestionDelete : std::vector<std::string>
- _suggestionEdit : std::vector<std::string>
- _suggestionSearch : std::vector<std::string>
- _suggestionShow : std::vector<std::string>

- setUpsuggestionAdd() : void

- setUpsuggestionDelete() : void

- setUpsuggestionEdit() : void

- setUpsuggestionSearch() : void

- setUpsuggestionShow() : void

+ getComdType (std::string) : ComdType

+ getSuggestionAdd() : std::vector<std::string>

+ getSuggestionDelete() : std::vector<std::string>

+ getSuggestionEdit() : std::vector<std::string>

+ getSuggestionSearch() : std::vector<std::string>

+ getSuggestionShow() : std::vector<std::string>
```

3. Logic

3.1 Logic

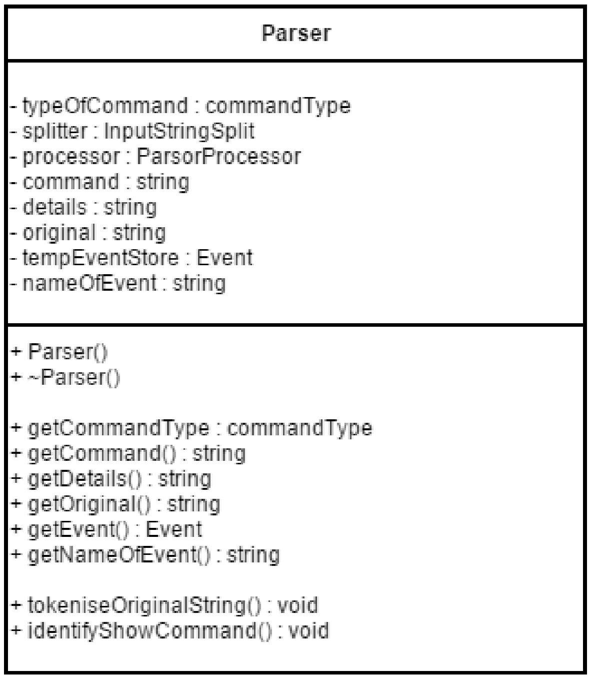

Logic
- parserPtr : Parser*
- eventStore : EventStorage
- display : Display
+ Logic()
+ getCommand() : Parser::commandType
+ getEventStorage() : EventStorage
+ getFloatingStrings() : vector<string>
+ getMainStrings() : vector<MAIN_EVENT>
+ getFeedbackStrings() : vector<string>
+ getErrorString() : string
+ executeUserInput(string) : bool
+ executeCommand(Parser::commandType, Event)
+ deleteParserPtr() : void
+ isNumber(string) : bool
+ convertNameToID(string) : int
+ setDisplay(bool, vector<Event> : void

3.2 Display


Display
- normalEvents : vector<Event>
- floatingEvents : vector<Event>
- feedbackEvents : vector<Event>
- mainDisplayStrings : vector<MAIN_EVENT>
- floatingDisplayStrings : vector<string>
- feedbackDisplayStrings : vector<string>
- totalNumEvents : int
+ ADDED_MESSAGE : static const string
+ EDITED_MESSAGE : static const string
+ DELETED_MESSAGE : static const string
+ Display()
+ getNormalEvents() : vector<Event>
+ getFloatingEvents() : vector<Event>
+ getFeedbackEvents() : vector<Event>
+ getMainDisplayStrings() : vector<MAIN_EVENT>
+ getFloatingDisplayStrings() : vector<string>
+ getFeedbackDisplayStrings() : vector<string>
+ getTotalNumEvents() : int
+ getID(int) : int
+ setNormalEvents(vector<Event>) : void
+ setFloatingEvents(vector<Event>) : void
+ setFeedbackEvents(vector<Event>) : void
+ normalEventsToString() : void
+ floatingEventsToString() : void
+ setFeedbackStrings(string) : void
<<struct>>
MAIN_EVENT
eventString : string
isNew : bool

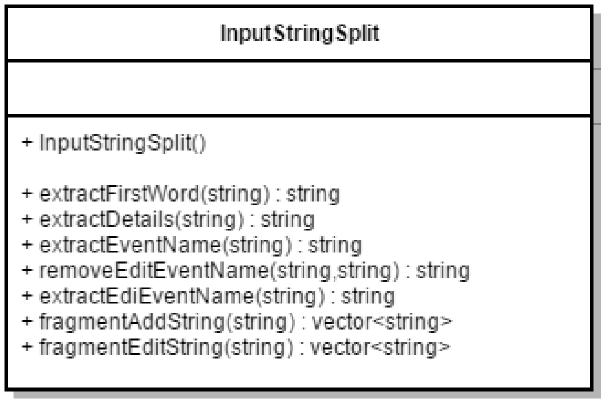
4. Parser

4.1 Parser



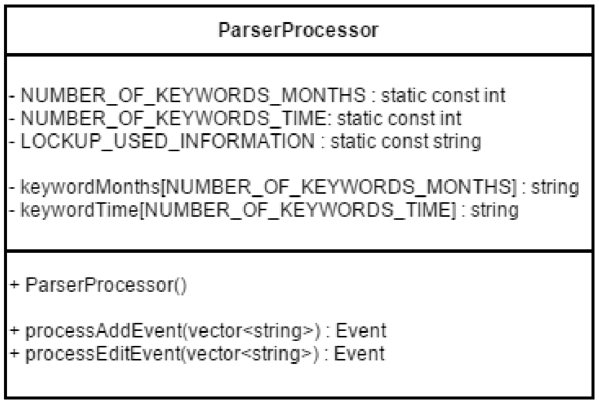
```
Parser

- typeOfCommand : commandType
- splitter : InputStringSplit
- processor : ParsorProcessor
- command : string
- details : string
- original : string
- tempEventStore : Event
- nameOfEvent : string

+ Parser()
+ ~Parser()

+ getCommandType : commandType
+ getCommand() : string
+ getDetails() : string
+ getOriginal() : string
+ getEvent() : Event
+ getNameOfEvent() : string

+ tokeniseOriginalString() : void
+ identifyShowCommand() : void
```

4.2 InputStringSplit



```
InputStringSplit

+ InputStringSplit()

+ extractFirstWord(string) : string
+ extractDetails(string) : string
+ extractEventName(string) : string
+ removeEditEventName(string,string) : string
+ extractEdiEventName(string) : string
+ fragmentAddString(string) : vector<string>
+ fragmentEditString(string) : vector<string>
```

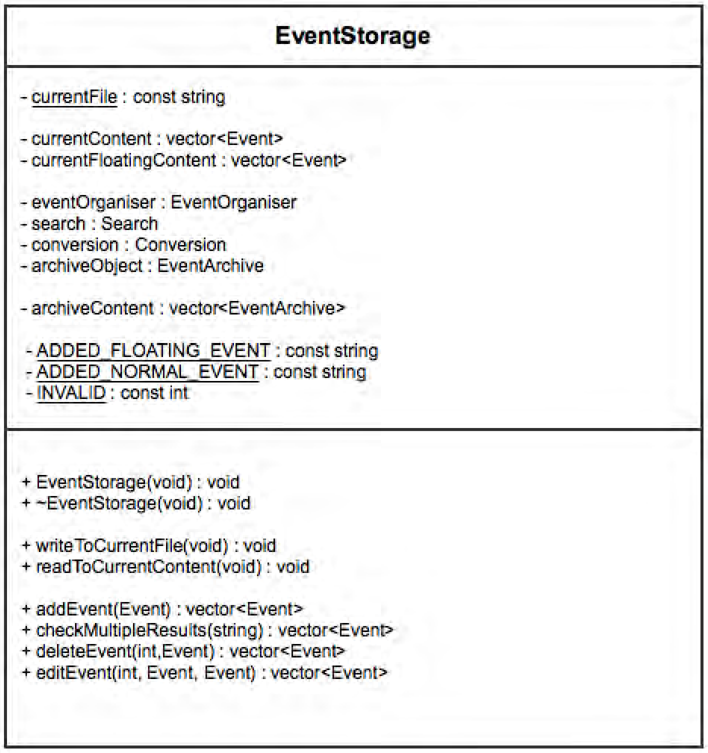
4.3 ParserProcessor



```
ParserProcessor

- NUMBER_OF_KEYWORDS_MONTHS : static const int
- NUMBER_OF_KEYWORDS_TIME: static const int
- LOCKUP_USED_INFORMATION : static const string

- keywordMonths[NUMBER_OF_KEYWORDS_MONTHS] : string
- keywordTime[NUMBER_OF_KEYWORDS_TIME] : string

+ ParserProcessor()

+ processAddEvent(vector<string>) : Event
+ processEditEvent(vector<string>) : Event
```

5. Storage

5.1 EventStorage



EventStorage

- currentFile : const string

- currentContent : vector<Event>
- currentFloatingContent : vector<Event>

- eventOrganiser : EventOrganiser
- search : Search
- conversion : Conversion
- archiveObject : EventArchive

- archiveContent : vector<EventArchive>

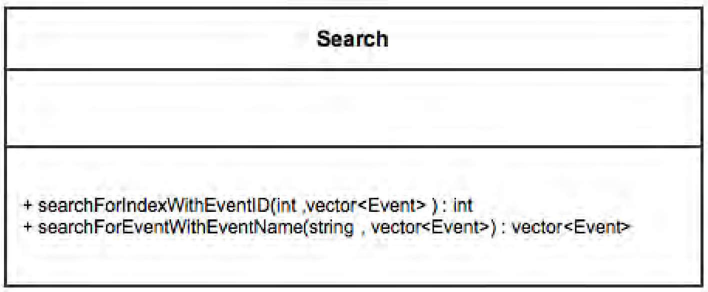
- ADDED_FLOATING_EVENT : const string
- ADDED_NORMAL_EVENT : const string
- INVALID : const int

+ EventStorage(void) : void
+ ~EventStorage(void) : void

+ writeToCurrentFile(void) : void
+ readToCurrentContent(void) : void

+ addEvent(Event) : vector<Event>
+ checkMultipleResults(string) : vector<Event>
+ deleteEvent(int,Event) : vector<Event>
+ editEvent(int, Event, Event) : vector<Event>

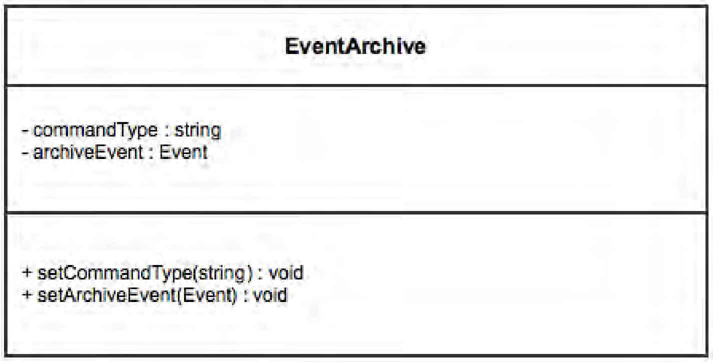
5.2 Search



Search

+ searchForIndexWithEventID(int ,vector<Event>) : int
+ searchForEventWithEventName(string , vector<Event>) : vector<Event>

5.3 EventArchive



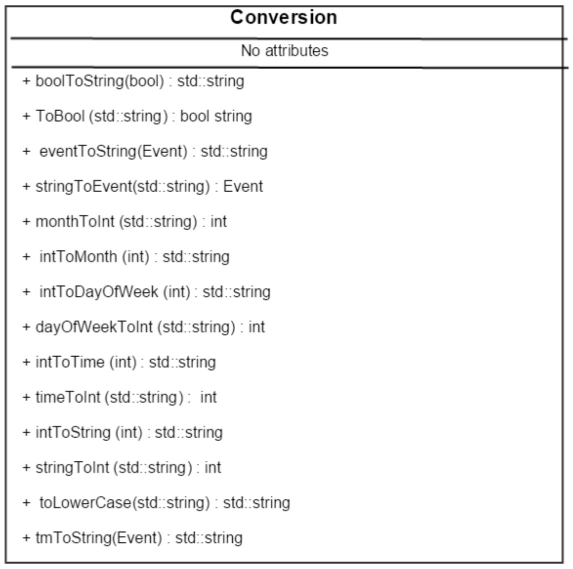
EventArchive

- commandType : string
- archiveEvent : Event

+ setCommandType(string) : void
+ setArchiveEvent(Event) : void

6. Global Classes

6.1 Conversion

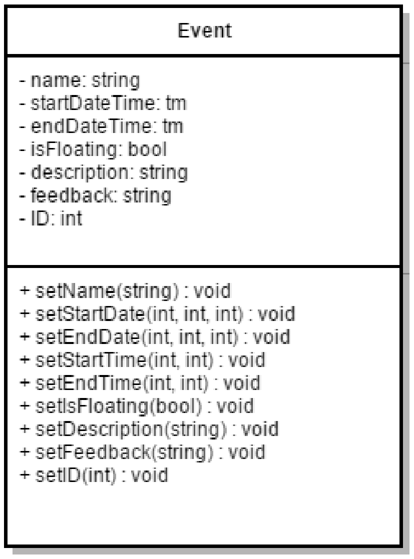


Conversion

No attributes

+ boolToString(bool) : std::string
+ ToBool (std::string) : bool string
+ eventToString(Event) : std::string
+ stringToEvent(std::string) : Event
+ monthToInt (std::string) : int
+ intToMonth (int) : std::string
+ intToDayOfWeek (int) : std::string
+ dayOfWeekToInt (std::string) : int
+ intToTime (int) : std::string
+ timeToInt (std::string) : int
+ intToString (int) : std::string
+ stringToInt (std::string) : int
+ toLowerCase(std::string) : std::string
+ tmToString(Event) : std::string

6.2 Event



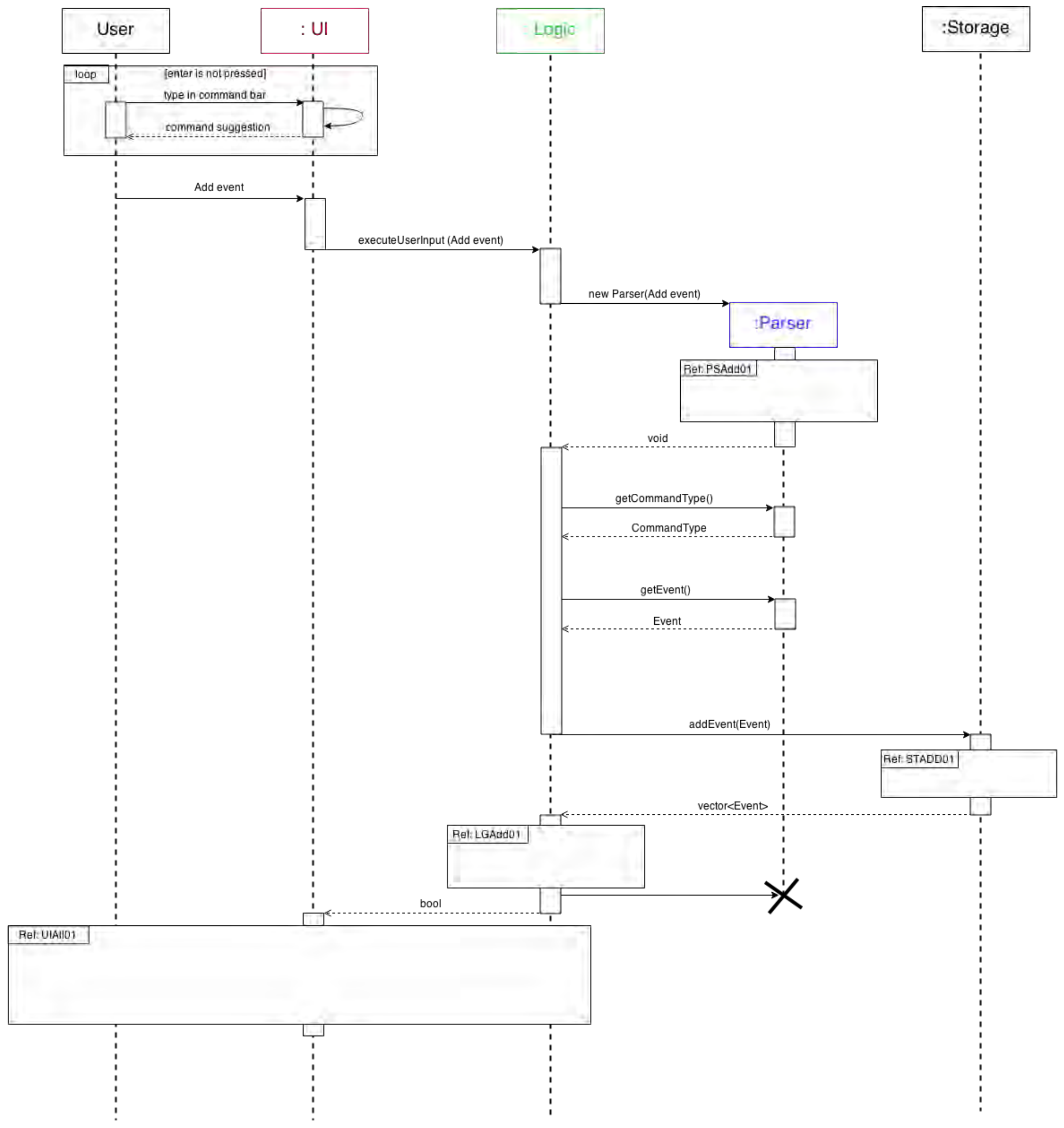
Event

- name: string
- startDateTime: tm
- endDateTime: tm
- isFloating: bool
- description: string
- feedback: string
- ID: int

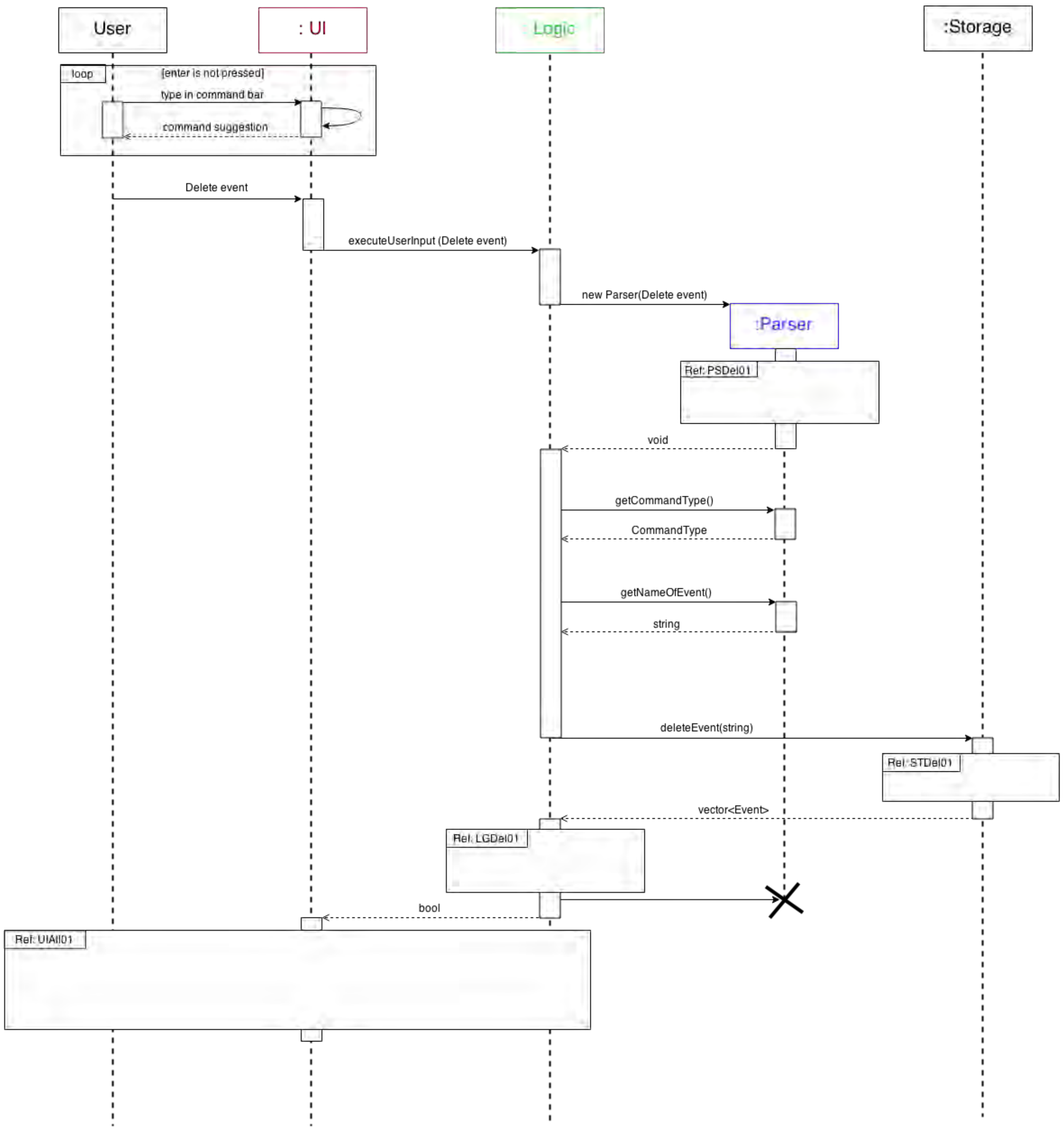
+ setName(string) : void
+ setStartDate(int, int, int) : void
+ setEndDate(int, int, int) : void
+ setStartTime(int, int) : void
+ setEndTime(int, int) : void
+ setIsFloating(bool) : void
+ setDescription(string) : void
+ setFeedback(string) : void
+ setID(int) : void

7. Sequence Diagrams

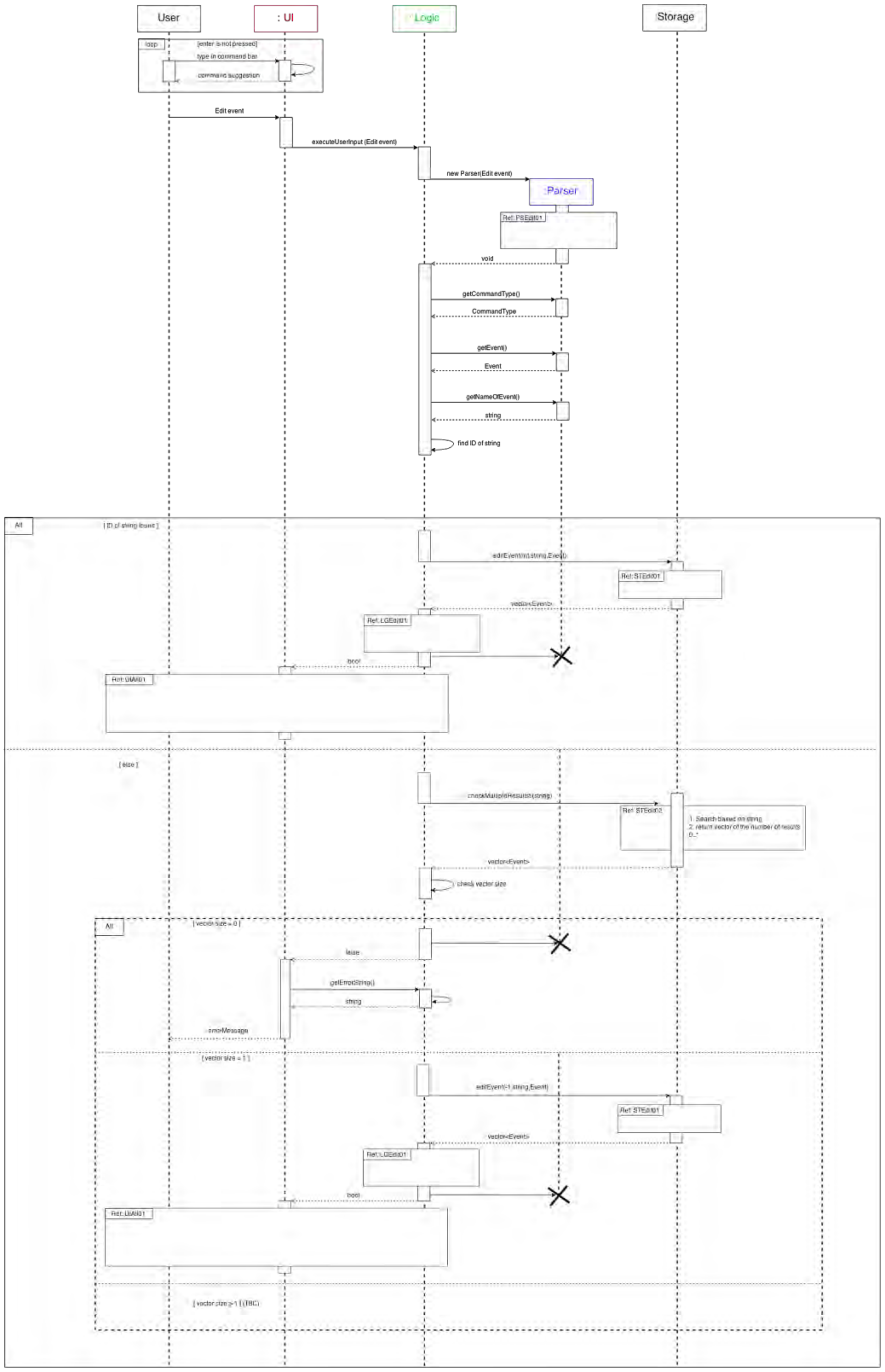
7.1 ADD



7.2 DELETE


User
: UI
Logic
:Storage
loop
[enter is not pressed]
type in command bar
command suggestion
Delete event
executeUserInput (Delete event)
new Parser(Delete event)
:Parser
Ref: PSDel01
void
getCommandType()
CommandType
getNameOfEvent()
string
deleteEvent(string)
Ref: STDel01
vector<Event>
Ref: LGDel01
bool
Ref: UIAll01

7.2 EDIT


User
: UI
:Logic
Storage
loop
[enter is not pressed]
type in command bar
command suggestion
Edit event
executeUserInput (Edit event)
new Parser(Edit event)
:Parser
Ref: PSEdit01
void
getCommandType()
CommandType
getEvent()
Event
getNameOfEvent()
string
find ID of string
Alt
[ID of string found]
editEvent(int,string,Event)
Ref: STEdit01
vector<Event>
Ref: LGEdit01
bool
Ref: UIAll01
[else]
checkMultipleResults(string)
Ref: STEdit02
1. Search based on string
2. return vector of the number of results 0..*
vector<Event>
check vector size
Alt
[vector size = 0]
false
getErrorString()
string
errorMessage
[vector size = 1]
editEvent(-1,string,Event)
Ref: STEdit01
vector<Event>
Ref: LGEdit01
bool
Ref: UIAll01
[vector size > 1] (TBC)

8. 8. Important APIs

8.1 GUI
void executeUserInput(string input): Centralises all calls from various parts/event handlers from the UI to Logic for execution. Thereafter, based on the boolean variable received from Logic::executeUserInput(), it proceeds to call functions to display the relevant information to the various displays on the GUI.

bool displayToAllDisplays(): Gets display vectors from Logic when invoked by function executeUserInput(), then displays these vectors to main display, floating tasks display and feedback box. Returns true upon successful display.

bool displayErrorString(): Get error string from Logic when invoked by function executeUserInput(), then displays the string on the GUI's main display. Upon successful display, it returns true to caller.

8.2 Logic
bool Logic::executeUserInput(string input): Called with the exact user input string. Creates Parser object to determine the correct action to take, then calls another Logic::executeCommand to execute this action. Returns true by default; will be able to return false when future error cases are implemented.

void Logic::executeCommand(Parser::commandType command, Event userEvent): Calls APIs from EventStorage to execute the desired command based on the input parameters (e.g. if command = ADDFLOAT, it will call EventStorage to add the floating event denoted by userEvent).

8.3 Parser
void Parser::tokenizeOriginalString(): Separates input string into a command and additional details. Based on the command, calls InputStringSplit object to further split the remaining string, then calls ParserProcessor object to process the split string. Command type will be determined and additional information will be stored in Event object within the Parser object.

8.4 InputStringSplit
vector<string> InputStringSplit::fragmentAddString(string input): Splits input string into components by removing spaces and ".-" symbols, then stores them in a vector<string>. Returns this vector.

8.5 ParserProcessor
Event ParserProcessor::processAddEvent(vector<string> fragmentedWords): Identifies names, dates and time in their respective formats from argument vector<string>, stores them in an Event object. Dates and time will be converted from string to integer and missing details filled in. Returns completed Event.

8.6 EventStorage
void EventStorage::writeToCurrentFile(): Processes events and stores them in an external storage as strings. Implements commands from Logic (eg, add, delete, edit) using internal vector<Event> (currentContent and currentFloatingContent) followed by saving the new data in the external txt file.

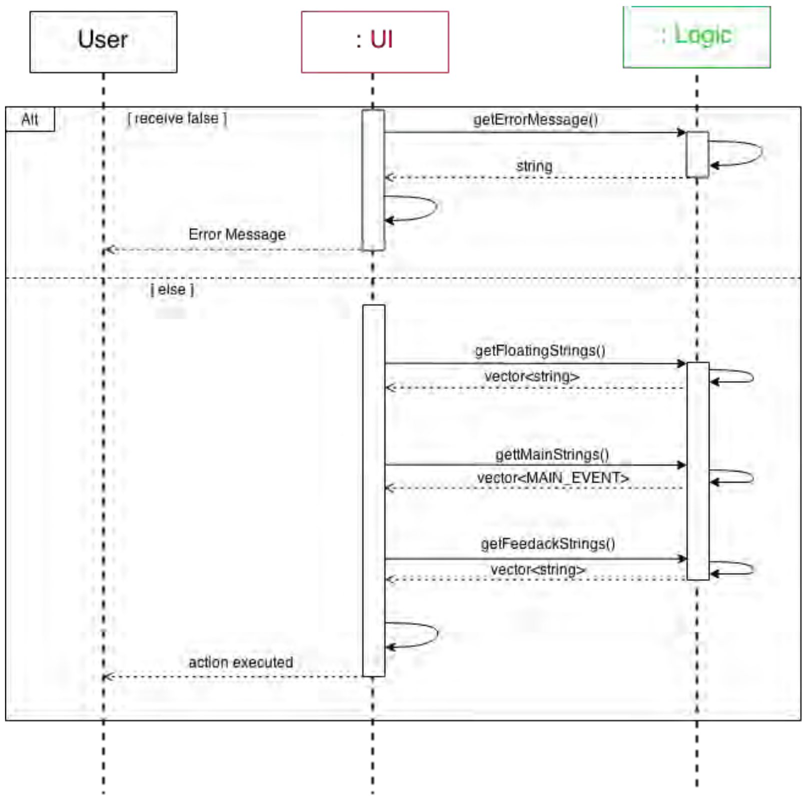
void EventStorage::readToCurrentContent(): Used when EventStorage object is created to read all data from external txt file into internal vector<Event>. To do this, event components that were saved, as strings of text will be individual transferred into an event format and saved in the internal storage (currentContent or currentFloatingContent).

Ok. But do mention which component may want to use a certain API and when/ for what it may use it. That gives the reader a better idea of when he might need the API. Otherwise, it is just a bunch of functions.

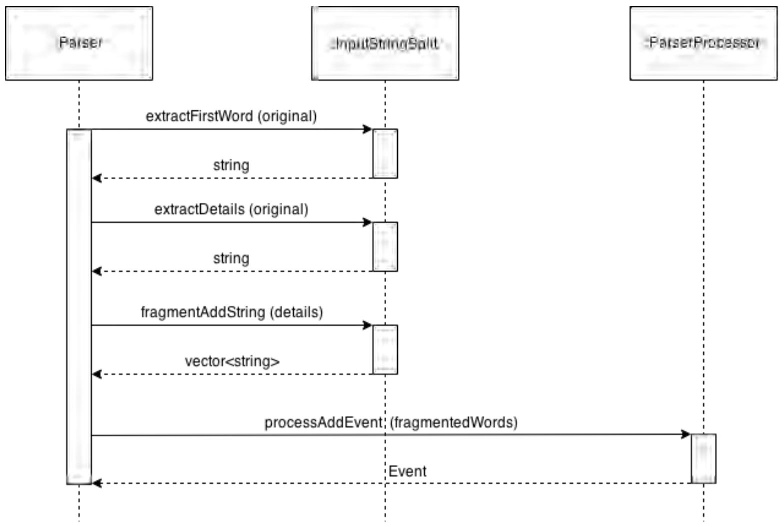
9. Appendix

9.1 Sequence Diagrams

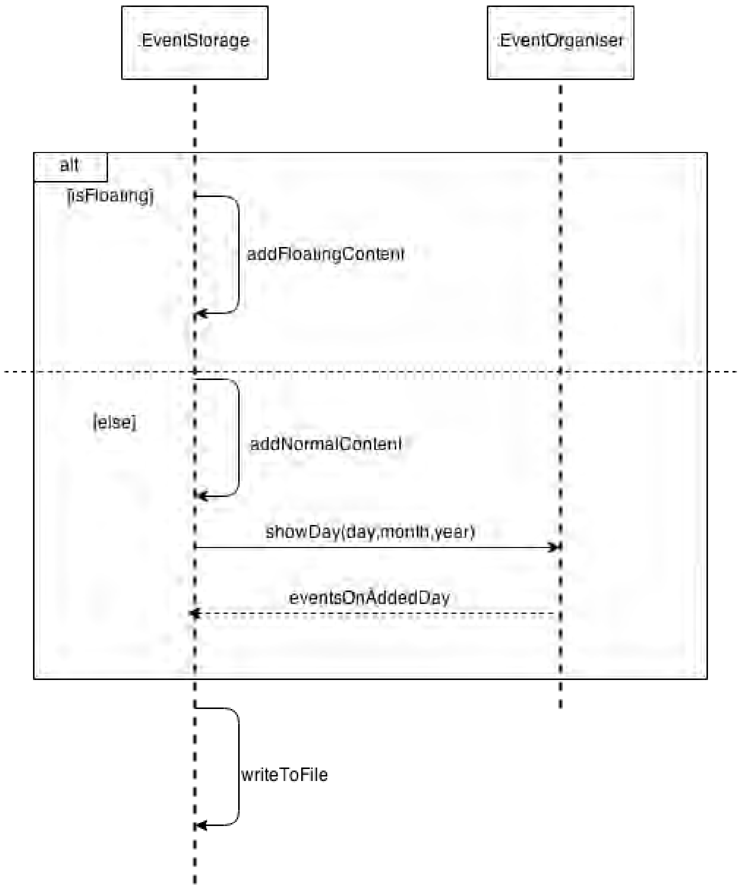
UIAll01



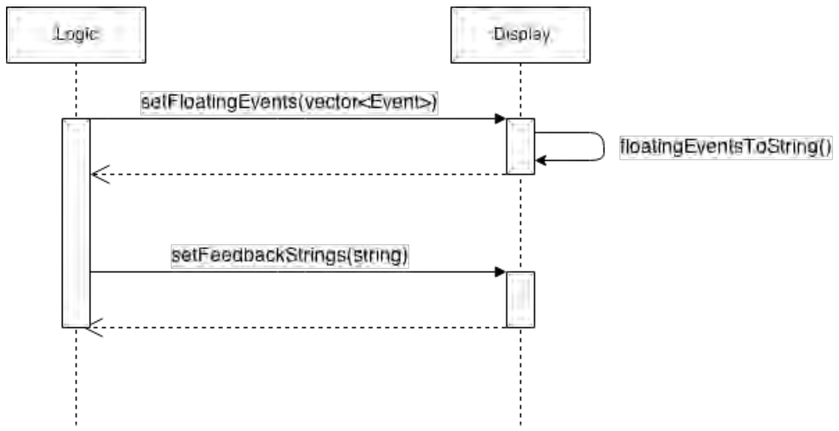
PSAdd01


:Parser
:InputStringSplit
:ParserProcessor
extractFirstWord (original)
string
extractDetails (original)
string
fragmentAddString (details)
vector<string>
processAddEvent (fragmentedWords)
Event

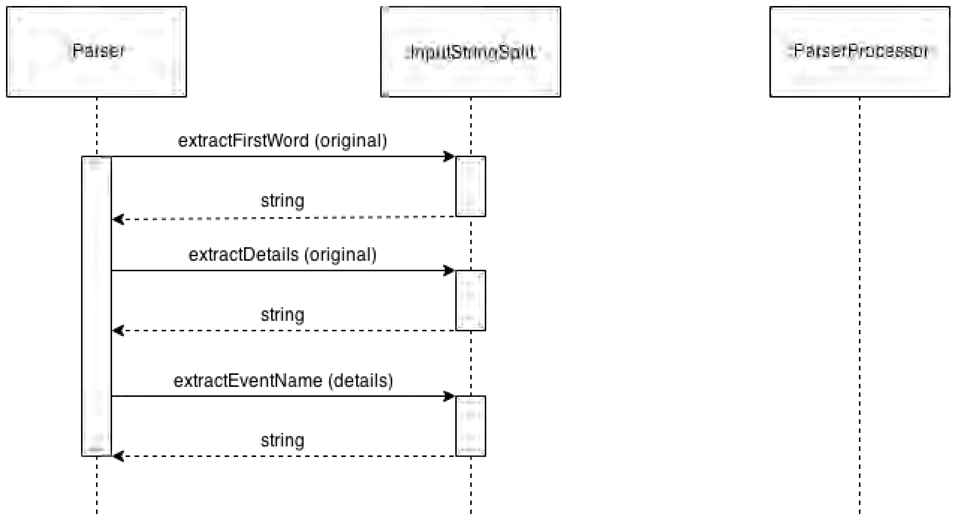
STADD01


:EventStorage
:EventOrganiser
alt
[isFloating]
addFloatingContent
[else]
addNormalContent
showDay(day,month,year)
eventsOnAddedDay
writeToFile

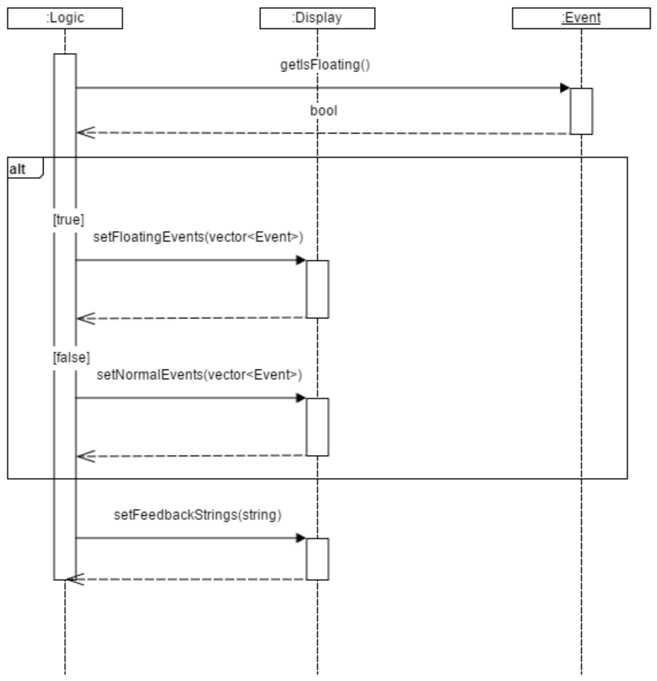
LGAdd01


Logic
Display
setFloatingEvents(vector<Event>)
floatingEventsToString()
setFeedbackStrings(string)

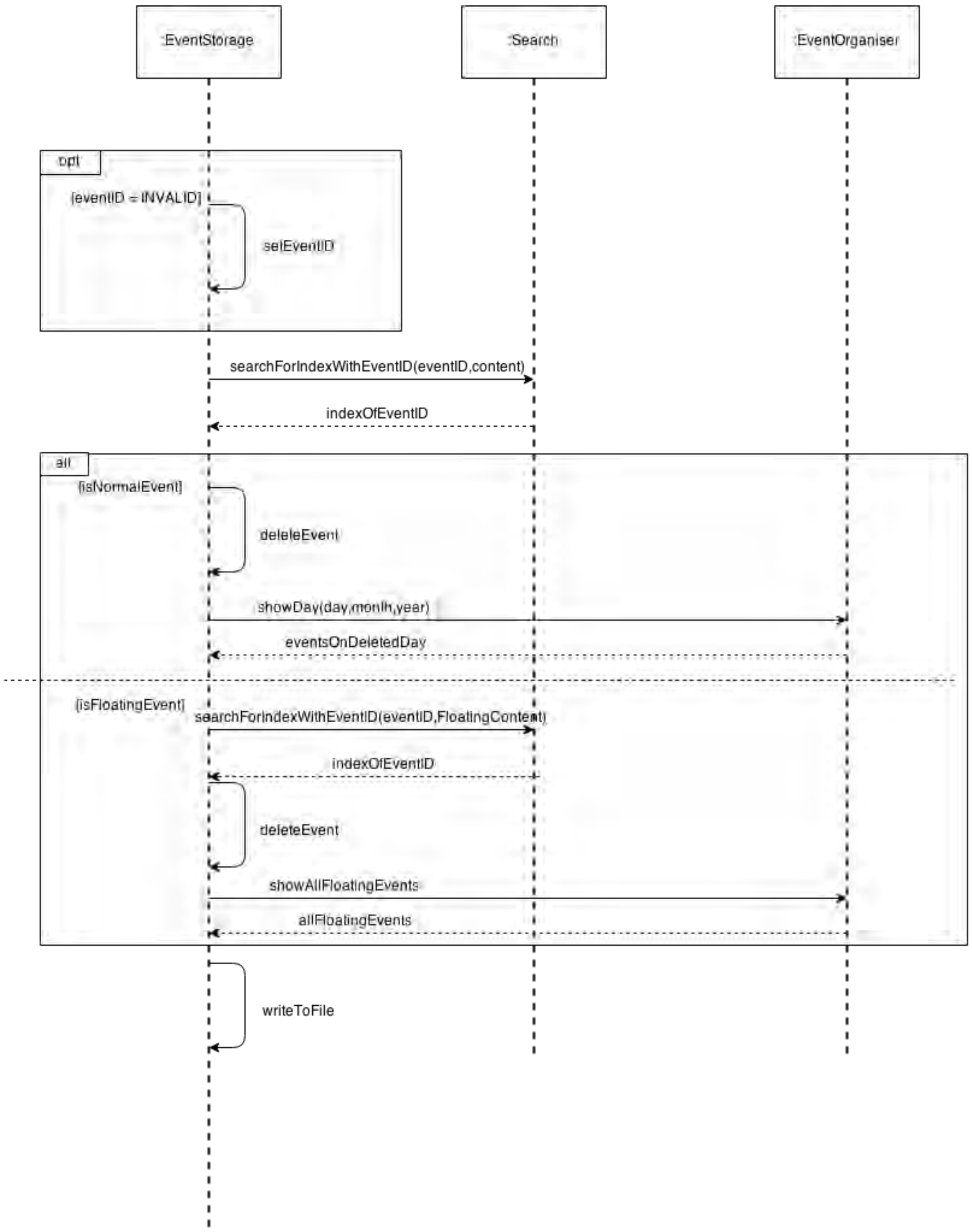
PSDel01


Parser
:InputStringSplit
:ParserProcessor
extractFirstWord (original)
string
extractDetails (original)
string
extractEventName (details)
string

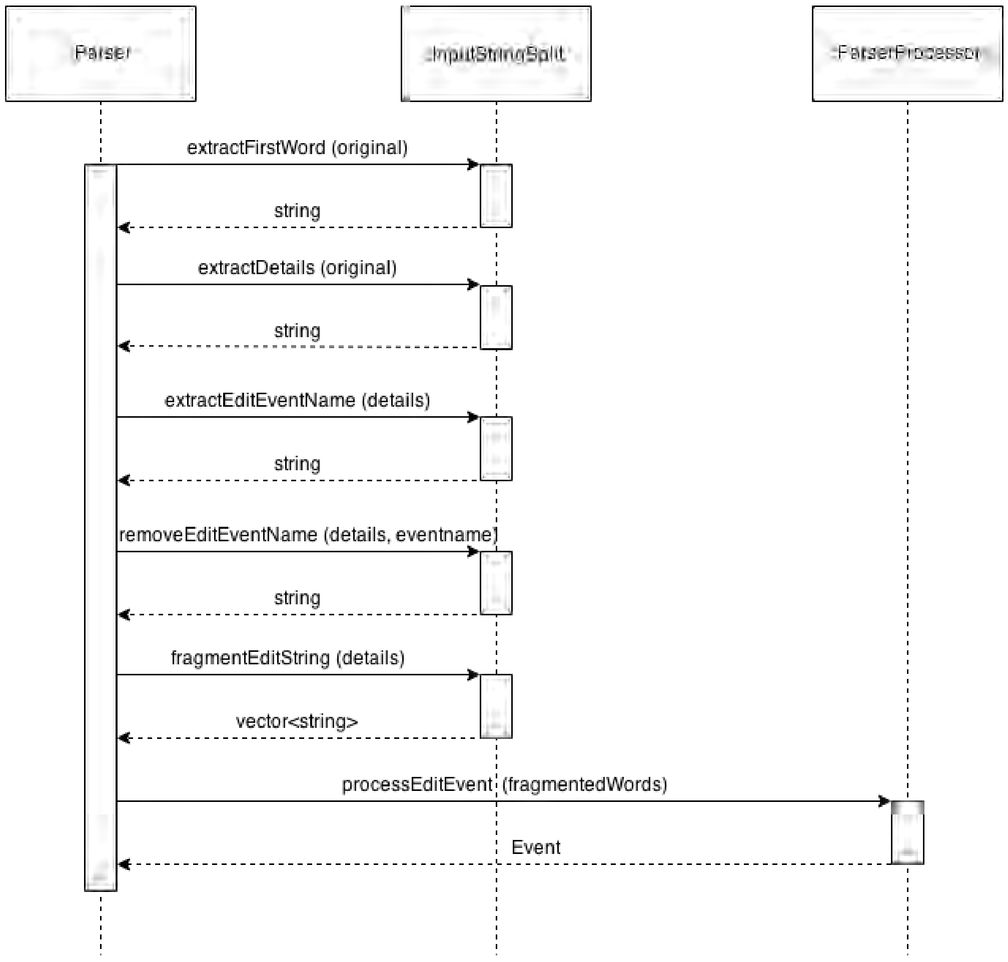
LGDel01


:Logic
:Display
:Event
getIsFloating()
bool
alt
[true]
setFloatingEvents(vector<Event>)
[false]
setNormalEvents(vector<Event>)
setFeedbackStrings(string)

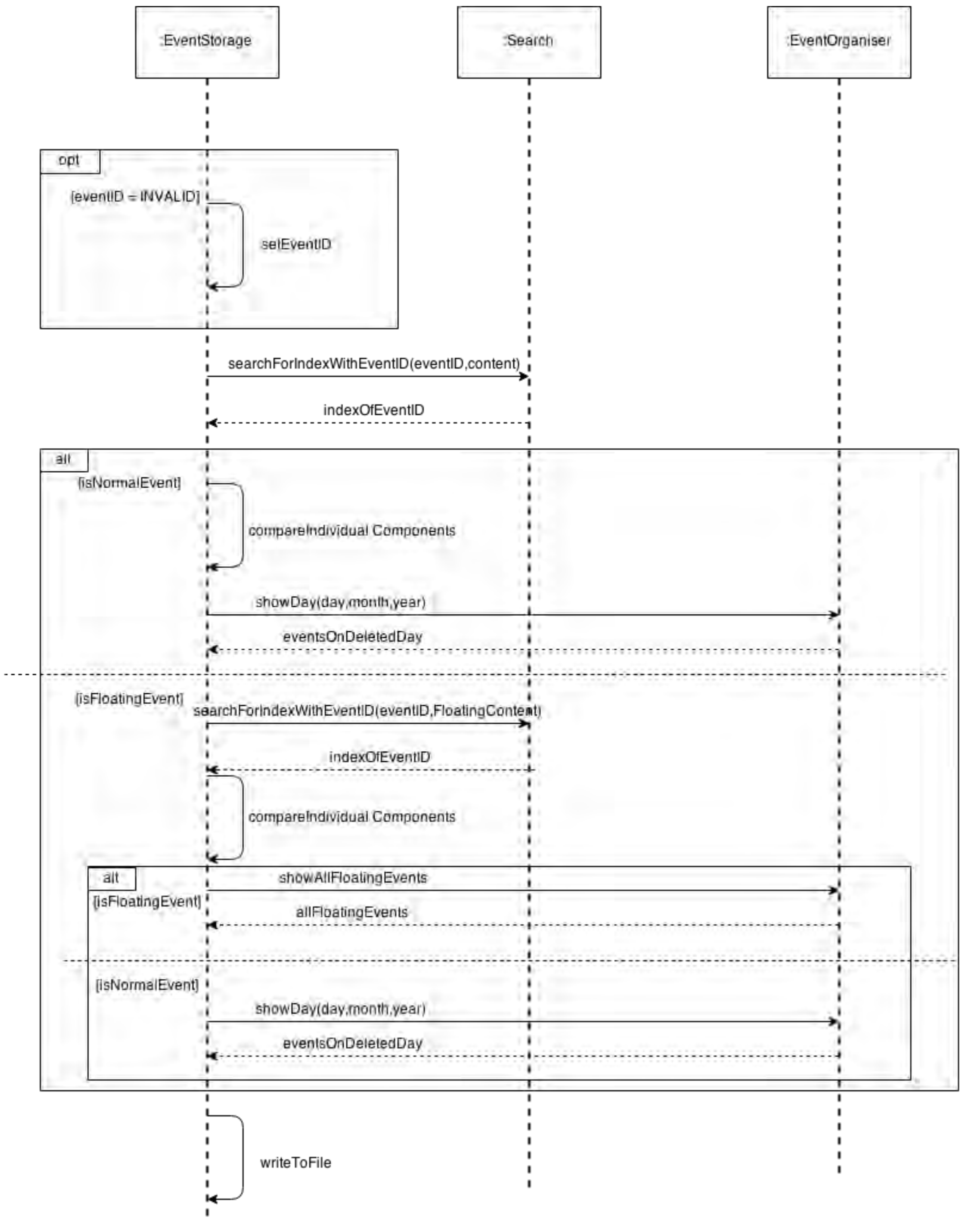
STDel01


:EventStorage
:Search
:EventOrganiser
opt
[eventID = INVALID]
setEventID
searchForIndexWithEventID(eventID,content)
indexOfEventID
alt
[isNormalEvent]
deleteEvent
showDay(day,month,year)
eventsOnDeletedDay
[isFloatingEvent]
searchForIndexWithEventID(eventID,FloatingContent)
indexOfEventID
deleteEvent
showAllFloatingEvents
allFloatingEvents
writeToFile

PSEdit01


Parser
:InputStringSplit
:ParserProcessor
extractFirstWord (original)
string
extractDetails (original)
string
extractEditEventName (details)
string
removeEditEventName (details, eventname)
string
fragmentEditString (details)
vector<string>
processEditEvent (fragmentedWords)
Event

STEdit01


:EventStorage
:Search
:EventOrganiser
opt
[eventID = INVALID]
setEventID
searchForIndexWithEventID(eventID,content)
indexOfEventID
alt
[isNormalEvent]
compareIndividual Components
showDay(day,month,year)
eventsOnDeletedDay
[isFloatingEvent]
searchForIndexWithEventID(eventID,FloatingContent)
indexOfEventID
compareIndividual Components
alt
[isFloatingEvent]
showAllFloatingEvents
allFloatingEvents
[isNormalEvent]
showDay(day,month,year)
eventsOnDeletedDay
writeToFile

9.2 APIs

UI

```
void executeUserInput(std::string input){
        bool isExecuted = lGPtr->executeUserInput(input);

        bool isAllDisplayed;
        bool isErrorStringDisplayed;
        if(isExecuted){
                isAllDisplayed = displayToAllDisplays();
        } else {
                isErrorStringDisplayed = displayErrorString();
        }
        // if (isAllDisplayed && isErrorStringDisplayed), Error
        //Else carry on
}

bool displayToAllDisplays(){
        vector<std::string> displayToFloating = lGPtr->getFloatingStrings();
        vector<Display::MAIN_EVENT> displayToMain = lGPtr->getMainStrings();
        vector<std::string> displayToFeedback = lGPtr-> getFeedbackStrings();
        bool checkAllDisplayed;
        bool displayFeedback = displayToFeedbackBox (displayToFeedback);
        bool displayFloating = displayToFloatingDisplay (displayToFloating);
        bool displayMain = displayToMainDisplay (displayToMain);
        if (!displayFeedback || !displayFloating || !displayMain){
                checkAllDisplayed = false;
        } else {
                checkAllDisplayed = true;
        }
        return checkAllDisplayed;
}

bool displayErrorString(){
        bool isErrorStringShown;
        std::string tempErrorString = lGPtr->getErrorString();
        String^ errorString = convertToSys(tempErrorString);
        display->Text = errorString;
        //if there is error, isErrorStringShown = false;
        isErrorStringShown = true;
        return isErrorStringShown;
}
```

Logic

```
bool Logic::executeUserInput(string input) {
        parserPtr = new Parser(input);

        Parser::commandType command = getCommand();
        Event userEvent = getEvent();
        executeCommand(command, userEvent);
        deleteParserPtr();
        return true;
}
void Logic::executeCommand(Parser::commandType command, Event userEvent) {
        string eventName = parserPtr->getNameOfEvent();//can be index in integer form (e.g. "1") or actual
name of event (e.g. "lunch")
        int index, id;
        vector<Event> tempEvents;
        switch (command) {
        case Parser::ADDFLOAT: {
                display.setFloatingEvents(eventStore.addEvent(userEvent));
                display.setFeedbackStrings(userEvent.getName() + ADDED_MESSAGE);
                break;
                                           }
        case Parser::DELETE_: {
```

```
			if (isNumber(eventName)) {
				index = std::stoi(eventName);
				id = display.getID(index);
				eventName = display.getEventName(index);
			} else {
				id = INVALID_NUMBER;
			}
			tempEvents = eventStore.deleteEvent(id, eventName);
			bool isFloat = tempEvents[0].getIsFloating();
			if (isFloat) {
				display.setFloatingEvents(tempEvents);
			} else {
				display.setNormalEvents(tempEvents);
			}
			display.setFeedbackStrings(eventName + DELETED_MESSAGE);
			break;
									}
		case Parser::EDIT: {
			Event tempEvent = parserPtr->getEvent();
			if (isNumber(eventName)) {
				index = std::stoi(eventName);
				id = display.getID(index);
				eventName = display.getEventName(index);
			} else {
				id = INVALID_NUMBER;
			}
			tempEvents = eventStore.editEvent(id, eventName, tempEvent);
			bool isFloat = tempEvents[0].getIsFloating();
			if (isFloat) {
				display.setFloatingEvents(tempEvents);
			} else {
				display.setNormalEvents(tempEvents);
			}
			display.setFeedbackStrings(userEvent.getName() + EDITED_MESSAGE);
			break;
							}
		default:
			break;
		}
}
```

Parser

```
void Parser::tokenizeOriginalString(){
		command = splitter.extractFirstWord(original);
		details = splitter.extractDetails(original);
		std::vector<std::string> fragmentedWords;
		if(command == "add"){
			fragmentedWords = splitter.fragmentAddString(details);
			tempEventStore = processor.processAddEvent(fragmentedWords);
			if(tempEventStore.getIsFloating() == true){
				typeOfCommand = Parser::ADDFLOAT;
			} else {
				typeOfCommand = Parser::ADD;
			}
		}
		else if(command == "delete"){
			nameOfEvent = splitter.extractEventName(details);
			typeOfCommand = Parser::DELETE_;
		}
		else if(command == "edit"){
			nameOfEvent = splitter.extractEditEventName(details);
			details = splitter.removeEditEventName(details,nameOfEvent);
			fragmentedWords = splitter.fragmentEditString(details);
			tempEventStore = processor.processEditEvent(fragmentedWords);
			typeOfCommand = Parser::EDIT;
		}
```

```
        return;
}
```

InputStringSplit

```
std::vector<std::string> InputStringSplit::fragmentAddString(std::string input){
        std::string::size_type strCutIndex;
        std::vector<std::string> fragmentedWords;
        std::string tempString;
        bool endOfString = false;
        strCutIndex = input.find_first_of(";");  // ; indicates end of event name
        tempString = input.substr(0,strCutIndex);
        fragmentedWords.push_back(tempString + ";");
        strCutIndex = input.find_first_not_of(" -.;",strCutIndex);
        if(strCutIndex == std::string::npos){
                endOfString = true;
        }

        while(!endOfString){
                input = input.substr(strCutIndex);
                strCutIndex = input.find_first_of(" -.");
                fragmentedWords.push_back(input.substr(0,strCutIndex));
                strCutIndex = input.find_first_not_of(" -.",strCutIndex);
                if(strCutIndex == std::string::npos){
                        endOfString = true;
                }
        }
        return fragmentedWords;
}
```

ParserProcessor

```
Event ParserProcessor::processAddEvent(std::vector<std::string> fragmentedWords){
        Conversion convertor;
        Event tempEventStore;
        std::string strMonth;
        int tempInt;
        int day = 0, month = 0, year = 0, hour = 0, minute = 0;
        bool matchFound = false;
        bool startDayFound = false;
        bool endDayFound = false;
        bool startTimeFound = false;
        bool endTimeFound = false;
        bool afterTwelve = false;
        bool nameFound = false;

        int tempi = 0;
        unsigned int i;
        //finding all the names of event
        for(i = 0; i < fragmentedWords.size() && !nameFound; i++){
                if(fragmentedWords[i].find(";") != std::string::npos){

        tempEventStore.setName(fragmentedWords[i].substr(0,fragmentedWords[i].find_last_of(";")));
                        tempi++;
                        nameFound = true;
                }
        }
        int j;
        for(i = tempi; i < fragmentedWords.size(); i++){
                //finding date
                for (j = 0; j < NUMBER_OF_KEYWORDS_MONTHS && !matchFound; j++){
                        if(fragmentedWords[i].find(keywordMonths[j]) != std::string::npos){
                        matchFound = true;
                        tempi = i;
                        strMonth = keywordMonths[j];
                        }
```

```
}

if(matchFound){
        try {
        auto tempStoi = std::stoi(fragmentedWords[tempi]);
        fragmentedWords[tempi] = LOCKUP_USED_INFORMATION;
        tempInt = tempStoi;
        } catch (const std::invalid_argument& e){
        tempi--;
        auto tempStoi = std::stoi(fragmentedWords[tempi]);
        fragmentedWords[tempi] = LOCKUP_USED_INFORMATION;
        tempInt = tempStoi;
        }
        day = tempInt;
        month = convertor.monthToInt(strMonth);
        year = 2015-1900;
        if(!startDayFound){
        startDayFound = true;
        tempEventStore.setStartDate(day,month,year);
        } else {
        endDayFound = true;
        tempEventStore.setEndDate(day,month,year);
        }
}
matchFound = false;
//finding time
for (j = 0; j < NUMBER_OF_KEYWORDS_TIME && !matchFound; j++){
        if(fragmentedWords[i].find(keywordTime[j]) != std::string::npos){
        matchFound = true;
        tempi = i;
        if(keywordTime[j] == "pm"){
                afterTwelve = true;
        }
        }
}
if(matchFound){
        try {
        auto tempStoi = std::stoi(fragmentedWords[tempi]);
        fragmentedWords[tempi] = LOCKUP_USED_INFORMATION;
        tempInt = tempStoi;
        } catch (const std::invalid_argument& e){
        tempi--;
        auto tempStoi = std::stoi(fragmentedWords[tempi]);
        fragmentedWords[tempi] = LOCKUP_USED_INFORMATION;
        tempInt = tempStoi;
        }
        if(tempInt >= 100){
        minute = tempInt%100;
        if(afterTwelve){
                hour = tempInt/100 + 12;
        } else {
                hour = tempInt/100;
        }
        if(!startTimeFound){
                tempEventStore.setStartTime(hour,minute);
                startTimeFound = true;
        }
        else {
                tempEventStore.setEndTime(hour,minute);
                endTimeFound = true;
        }
        } else if(tempInt < 100){
        tempi--;
        try {
                hour = std::stoi(fragmentedWords[tempi]);
                fragmentedWords[tempi] = LOCKUP_USED_INFORMATION;
                minute = tempInt;
```

```
				} catch (const std::invalid_argument& e){
					hour = tempInt;
					minute = 0;
				}
				if(afterTwelve){
					hour = hour + 12;
				}
				if(!startTimeFound){
					startTimeFound = true;
					tempEventStore.setStartTime(hour,minute);
				}
				else {
					endTimeFound = true;
					tempEventStore.setEndTime(hour,minute);
				}
				}
			}
			matchFound = false;
		}
		if(!startDayFound && !startTimeFound){
			tempEventStore.setIsFloating(true);
		}
		if(startDayFound && !startTimeFound){
			tempEventStore.setStartTime(0,0);
			tempEventStore.setEndTime(23,59);
		}
		if(!startDayFound && startTimeFound){
			time_t t = time(0);
			struct tm* now = localtime(&t);
			day = now->tm_mday;
			month = now->tm_mon;
			year = now->tm_year;
			tempEventStore.setStartDate(day,month,year);
			tempEventStore.setEndDate(day,month,year);
		}
		if(!endDayFound){
			tempEventStore.setEndDate(day,month,year);
		}
		if(!endTimeFound){
			tempEventStore.setEndTime(hour+1,minute);
		}
		if(endDayFound && !endTimeFound){
			tempEventStore.setEndTime(23,59);
		}
		struct tm* temptmPtr;
		temptmPtr = &tempEventStore.getStartDate();
		mktime(temptmPtr);
		tempEventStore.setStartDate(temptmPtr->tm_mday,temptmPtr->tm_mon,temptmPtr->tm_year);
		tempEventStore.setStartTime(temptmPtr->tm_hour,temptmPtr->tm_min);
		temptmPtr = &tempEventStore.getEndDate();
		mktime(temptmPtr);
		tempEventStore.setEndDate(temptmPtr->tm_mday,temptmPtr->tm_mon,temptmPtr->tm_year);
		tempEventStore.setEndTime(temptmPtr->tm_hour,temptmPtr->tm_min);
		return tempEventStore;
}
```

EventStorage

```
void EventStorage::writeToCurrentFile(){
		std::ofstream writeFile(currentFile);

		for(auto i=0;i<currentContent.size();i++){
			writeFile
				<< conversion.boolToString(currentContent[i].getIsFloating()) << std::endl
				<< currentContent[i].getName() << std::endl
				<< conversion.tmToString(currentContent[i]) << std::endl
				<< currentContent[i].getDescription() << std::endl
```

```
                        << currentContent[i].getFeedback() << std::endl
                        << currentContent[i].getID() << std::endl;
        }
        for(auto i=0;i<currentFloatingContent.size();i++){
                writeFile
                        << conversion.boolToString(currentFloatingContent[i].getIsFloating()) <<
std::endl
                        << currentFloatingContent[i].getName() << std::endl
                        << '\n' << '\n' << '\n' << '\n' << '\n' << '\n' << '\n' << '\n' << '\n' <<
std::endl
                        << currentFloatingContent[i].getDescription() << std::endl
                        << currentFloatingContent[i].getFeedback() << std::endl
                        << currentFloatingContent[i].getID() << std::endl;
        }
        writeFile.close();
}
void EventStorage::readToCurrentContent(){

        std::ifstream readFile(currentFile);
        std::string textLine, name, description, feedback, tags, startDateYear, startDateMonth,
startDateDay, startDateHour, startDateMin, endDateYear, endDateMonth, endDateDay, endDateHour, endDateMin,
id;
        getline(readFile, textLine);
        while(!readFile.eof()){
                Event* tempEvent = new Event;
                if(textLine == "0"){            //Normal case
                        //getinfo from textfile;
                        getline(readFile, name);
                        //cin event times
                        getline(readFile, startDateYear);
                        getline(readFile, startDateMonth);
                        getline(readFile, startDateDay);
                        getline(readFile, startDateHour);
                        getline(readFile, startDateMin);
                        getline(readFile, endDateYear);
                        getline(readFile, endDateMonth);
                        getline(readFile, endDateDay);
                        getline(readFile, endDateHour);
                        getline(readFile, endDateMin);

                        getline(readFile, description);
                        getline(readFile, feedback);
                        getline(readFile, id);
                        //createEvent
                        tempEvent->setIsFloating(false); //stringToBool
                        tempEvent->setName(name);
                        tempEvent->setDescription(description);
                        tempEvent->setFeedback(feedback);
                        tempEvent->setID(atoi(id.c_str()));
                        tempEvent->setStartTime(atoi(startDateHour.c_str()),atoi(startDateMin.c_str()));
                        tempEvent-
>setStartDate(atoi(startDateDay.c_str()),atoi(startDateMonth.c_str()),atoi(startDateYear.c_str()));
                        tempEvent->setEndTime(atoi(endDateHour.c_str()),atoi(endDateMin.c_str()));
                        tempEvent-
>setEndDate(atoi(endDateDay.c_str()),atoi(endDateMonth.c_str()),atoi(endDateYear.c_str()));

                        currentContent.push_back(*tempEvent);
                } else if(textLine == "1"){     //floatingEvent
                        //getinfo from textfile
                        getline(readFile, name);
                        getline(readFile, startDateYear);
                        getline(readFile, startDateMonth);
                        getline(readFile, startDateDay);
                        getline(readFile, startDateHour);
                        getline(readFile, startDateMin);
                        getline(readFile, endDateYear);
                        getline(readFile, endDateMonth);
                        getline(readFile, endDateDay);
                        getline(readFile, endDateHour);
                        getline(readFile, endDateMin);
                        getline(readFile, description);
                        getline(readFile, feedback);
                        getline(readFile, id);
                        //createEvent
                        tempEvent->setIsFloating(true);
                        tempEvent->setName(name);
                        tempEvent->setDescription(description);
                        tempEvent->setFeedback(feedback);
                        tempEvent->setID(atoi(id.c_str()));
                        currentFloatingContent.push_back(*tempEvent);
                }
                delete tempEvent;
                getline(readFile, textLine);//takes in 0/1 for isFloating check
        }
        readFile.close();
}
```