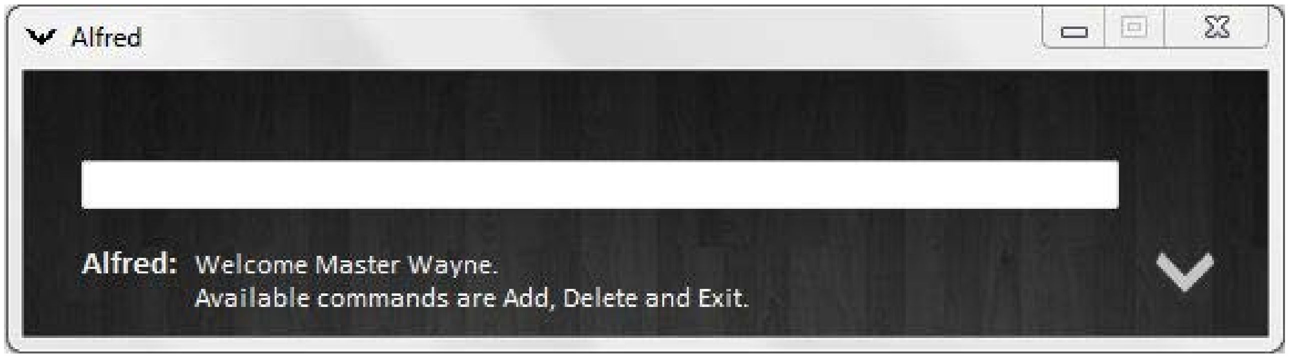
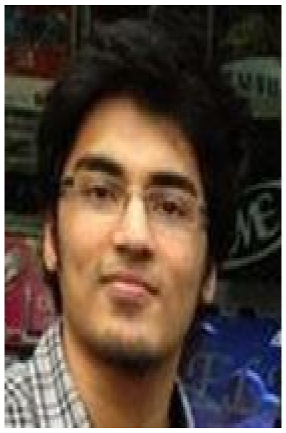
Alfred

He will organize your life



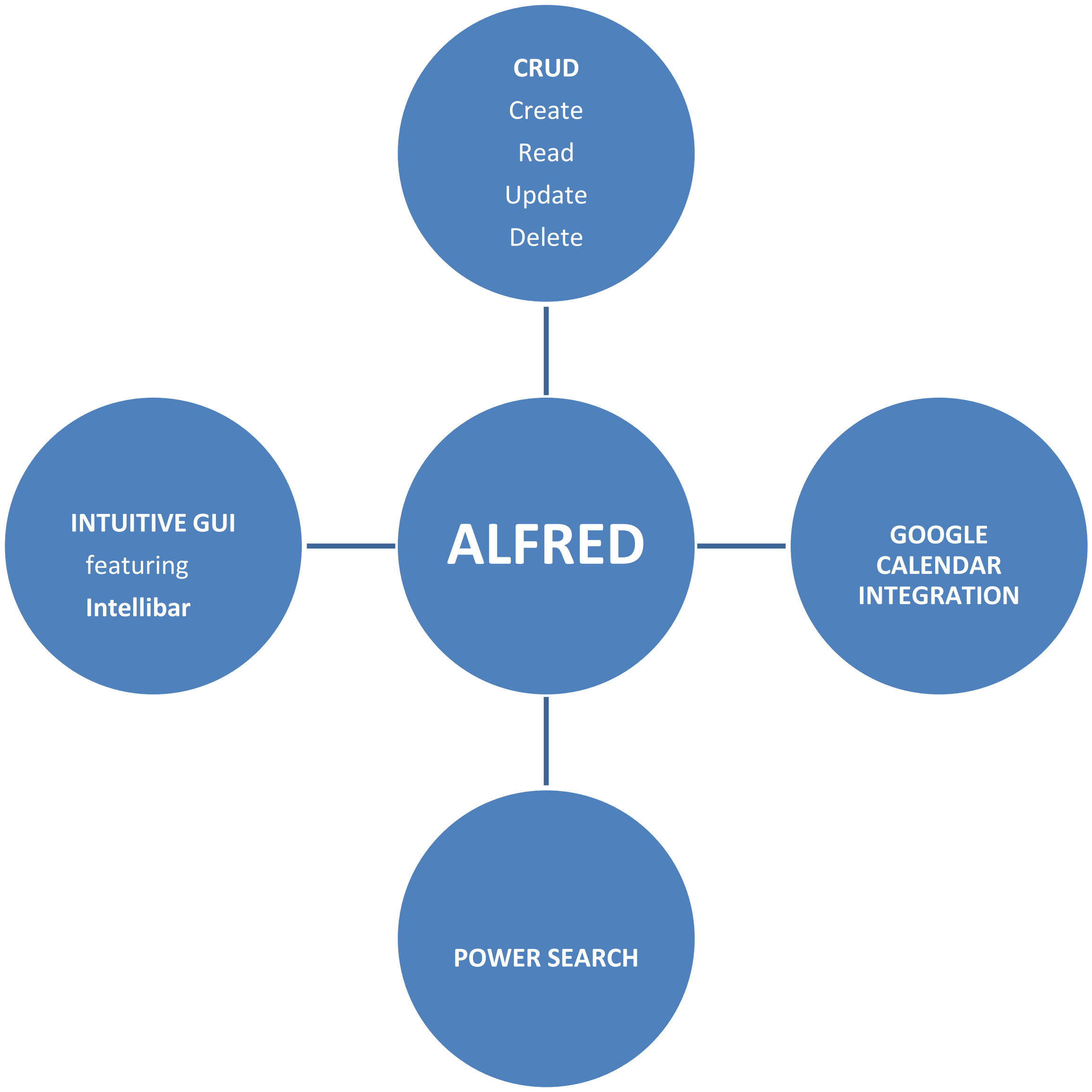
Alfred: Welcome Master Wayne.
Available commands are Add, Delete and Exit.

			
Shourya Moona	**Rohit Mukherjee**	**Lee Jianwei**	**Madhav Kannan**
[GUI, Software Developer, Testing]	**[Software Developer, Testing]**	**[Software developer, Documentation]**	**[Software architect, Documentation]**

User Guide

Who is Alfred?

Alfred is a productivity application that will organize your everyday life and ensure you never miss a deadline. For those on the go, Alfred will make sure you're never burdened with too many tasks at a time. A systematic planner application that will create an inventory of all your tasks, Alfred can add, delete and modify tasks on the go. He will sync your schedule with Google Calendar, allowing you to track your tasks on any portable device. He will also remind you to complete your tasks well in advance. With an elegant, intuitive interface Alfred is at your service 24x7. Granted you're not Batman, but everyone in the 21st century needs an Alfred.



Alfred's Functionality Specifications

Getting started

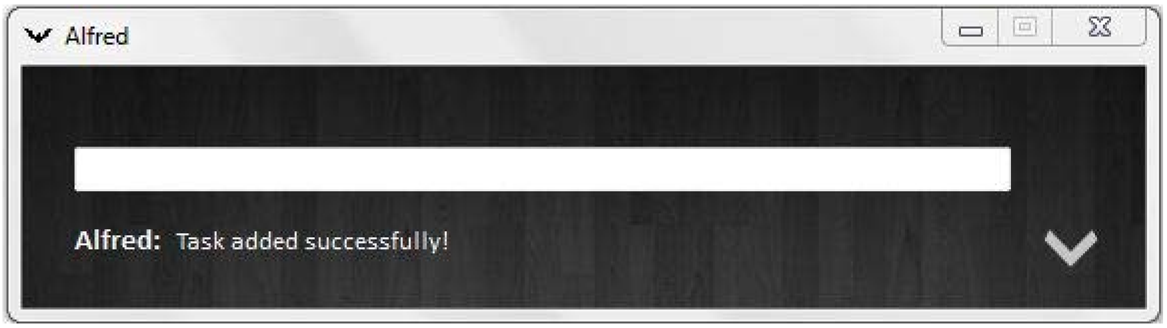
I Creating a Task: (*some parts to be implemented)

The new task can be entered in based on two formats. The task may be prefixed with an “add” keyword, but it is not necessary. “-ed”, “-et” and “-t” must be specified to indicate due date, time and tag of the task respectively:

- **Single day case –**
  A one-time event can be entered in the following ways:
  1. Task name and date and time. (Add Canoe polo –ed 16 October 2012 –et 5pm – Adds the task with the stated task description, with the specified due date and time.)
  2. Task name and date. (Add Canoe polo –ed 16 October 2012 – Adds the task with the stated task description and the specified due date.)
  3. Task name and time. (Add Canoe polo –et 5pm – Adds the task with the stated task description and the specified due time to the current or subsequent day, based on current time.)
  4. Task name (Add Canoe polo – Adds the task with the stated task description, as a floating task.)

- ***Recurring case –**
  An event that recurs can be added in the following way:
  1. Entering task and time and frequency. “-r” must be specified to indicate recurrence. (Add lunch –et 2pm –r – Adds it to every day at the mentioned time.)

- ***Setting Alarms –**
  A nifty alarm feature is provided:
  1. Adding an exclamation mark makes an event important and there will be a reminder for the event once a day from 3 days before the event.
  2. A pop-up will remind you of your upcoming event, half an hour before it is to be performed.
  3. Custom alarm tones can be assigned by the user as per their convenience, to create a variation from one task to the other.

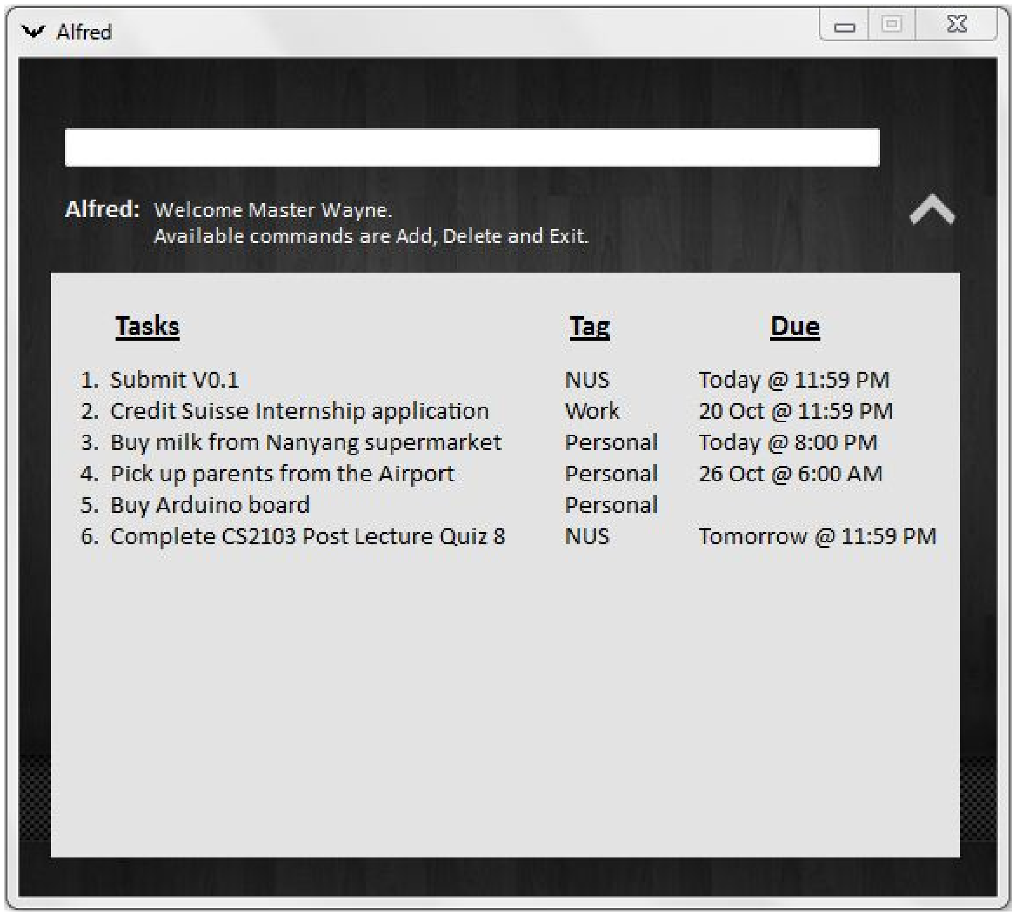


Alfred: Task added successfully!

Creating a task

II Reading a task:

- Default page with drop-down menu shows users what has been updated into planner. It shows the latest status of tasks (whether they have been completed or how much time remains for the task to begin).



Alfred: Welcome Master Wayne.
Available commands are Add, Delete and Exit.

Tasks	Tag	Due
1. Submit V0.1	NUS	Today @ 11:59 PM
2. Credit Suisse Internship application	Work	20 Oct @ 11:59 PM
3. Buy milk from Nanyang supermarket	Personal	Today @ 8:00 PM
4. Pick up parents from the Airport	Personal	26 Oct @ 6:00 AM
5. Buy Arduino board	Personal	
6. Complete CS2103 Post Lecture Quiz 8	NUS	Tomorrow @ 11:59 PM

Display (Expanded view)

III Update/Edit Tasks: (*some parts to be implemented)

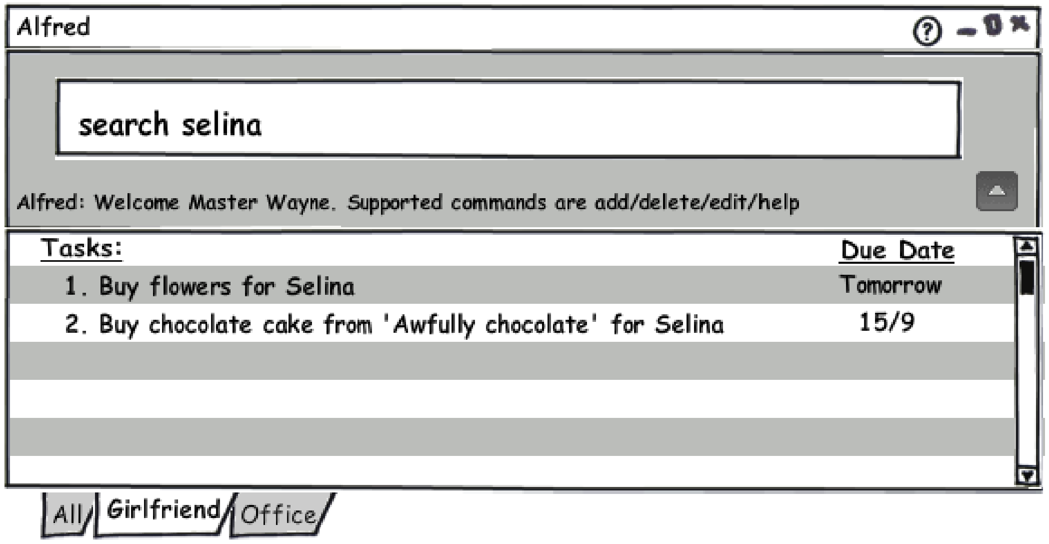
Tasks can be edited in 3 formats. The task is prefixed with different keywords as shown:
1. *Undoing last action. (U – Undoes last action.)
2. *Redoing last action. (R – Redoes last action.)
3. Overwriting a previously entered task. “-ed”, “-et” and “-t” must be specified to indicate due date, time and tag of the task respectively. (Edit canoe polo –ed 16 October 2012 - et 5pm – Updates the task description with the new due date and time.)

IV Delete: (*some parts to be implemented)

Tasks can be deleted in 3 formats. The task is prefixed with a "delete" keyword:
1. Task number. (Delete 2 – Deletes the 2nd task on the display list.)
2. *Task name and date and time OR task name and date OR task day/date and time. (Delete Canoe polo –ed 16 October 2012 - et 5pm – Deletes the task on the specified date, at the mentioned time.)
2. *Task name. (Delete Canoe polo – Deletes the first instance of the mentioned task.)
3. *Recurring task name. (Delete all Canoe polo – Deletes all instances of the mentioned task.)

V Power Search: (*to be implemented)

Tasks can be searched for in 3 formats. The task is prefixed with a "search" keyword:
1. Task name and date and time OR task name and date OR task day/date and time. (Search Canoe polo –ed October 16 2012 - et 5pm – Searches for the task on the specified date, at the mentioned time.)
2. Task name. (Search Canoe polo – Searches for the first instance of the mentioned task.)
3. Recurring task name. (Search all Canoe polo – Returns first 15 instances of the mentioned task.)


Alfred

search selina

Alfred: Welcome Master Wayne. Supported commands are add/delete/edit/help

Tasks:	Due Date
1. Buy flowers for Selina	Tomorrow
2. Buy chocolate cake from 'Awfully chocolate' for Selina	15/9

Search

VI Google Calendar integration: (*to be implemented)

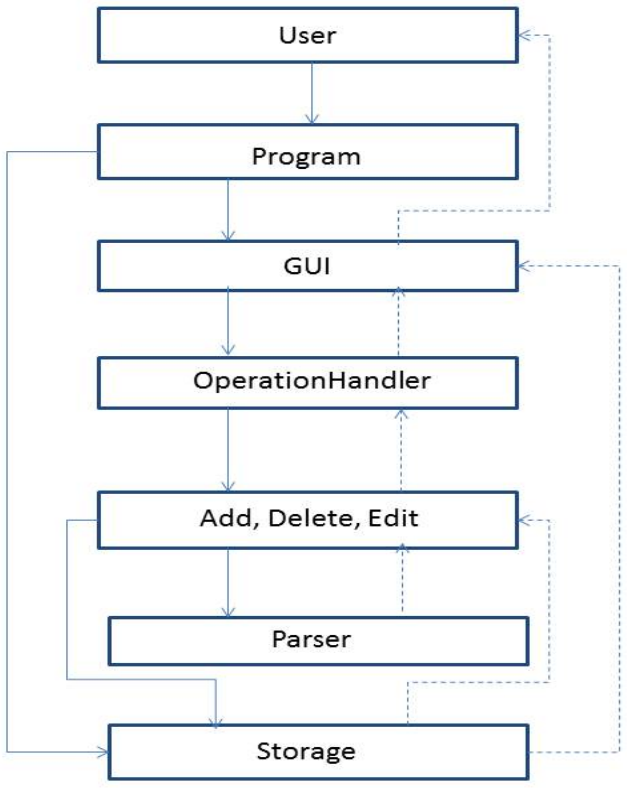
Any new task that has been added or a previously entered task that has been updated will find its way on the user's Google calendar. From there, it can be accessed on the user's portable devices, so that the user can access their schedule on-the-go.

Developer’s Guide

Introduction

Welcome, new developer! Greetings from the Alfred team. This guide has been created just for you, and describes to you exactly how our code is structured. You will find below all the constituents that helps Alfred function. A brief architecture of the software is provided, followed by a comprehensive list of the APIs that have been used. This is followed by a few code snippets to display the flexibility in functionality of our software. You must know that our software is still under development, so we conclude the guide by indicating what we intend to achieve in the coming few versions. Read on to see what makes Alfred tick.

Architecture



Key points of Alfred’s architecture are mentioned below –

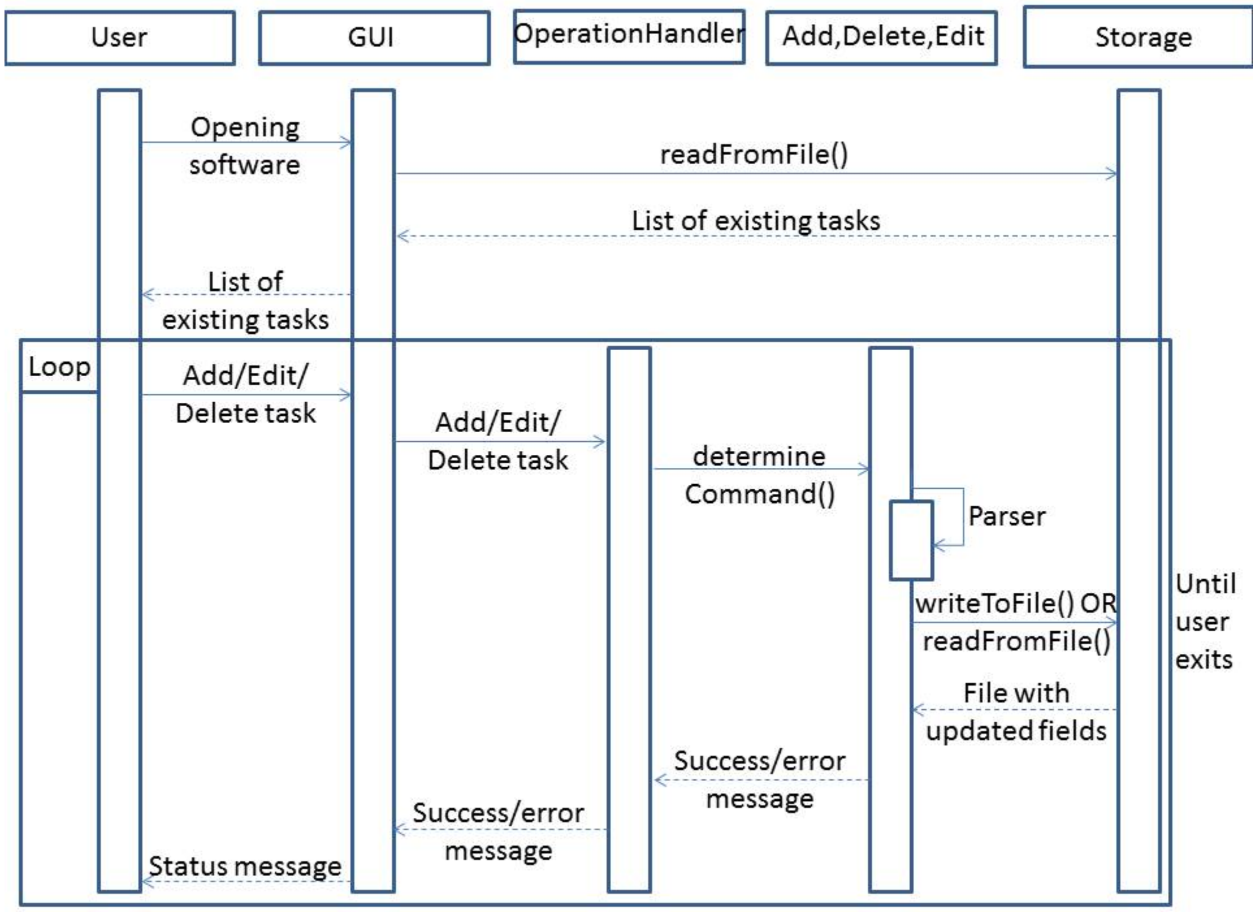
- Alfred possesses an n-tier architecture style. In such a style, as can be seen, each class is linked to the class below it. This holds true for Alfred in all cases but one (*Parser* class doesn’t access *Storage*).

- While *Program* initiates the software, the user interacts only with the GUI.
- The *OperationHandler* class links the *CRUD* classes (*Add*, *Edit* and *Delete*) with the GUI. These classes are independently related to the *Parser* class.
- The *Storage* class is accessed by a number of classes (*Program*, *CRUD* classes) simultaneously.

Design descriptions

Overall design:

The sequence diagram to display the overall working of the software is shown below –



It is further explained in text below –

- When the .exe file is opened by the user, an instance of UI is created, which displays the condensed version of the UI at start-up.
- The Storage class returns the list of existing of tasks to the UI at start-up, so that on pressing the drop-down icon, the user can see the same.
- The user now enters some input, for adding, editing or deleting a task in the ways mentioned.

- The command is now sent to the *OperationHandler* class, which determines the type of command. It is then sent to the class that handles the respective operation.
- Under the respective *CRUD (Add, Delete,Edit)* classes, the command is sent to the *Parser* class which converts input string to a readable form.
- The software-readable task is now sent back to the *CRUD* classes, which link with the *Storage* class.
- An operation success/error message is then sent to the UI, to display to the user.

Important Application Programming Interfaces

Add.cs: Takes in the task that must be added, and creates an instance of the Parser class to break the command down into attributes of the newTask object. It then adds the task to Data.txt.

public void addTask(string input)	Adds a new task into Data.txt.
private void isTaskDescriptionNull()	Checks if task contains any description.

Delete.cs: Takes in the task number of the task that must be deleted, and deletes it on locating within Data.txt, provided the task number entered is valid.

public int getIndex(string input)	Gets index of task to be deleted in file.
public void deleteTask(string input)	Deletes task using index in file.

Edit.cs: Takes in the task number of the task that must be edited, and overwrites the task on locating within Data.txt, provided the command entered is valid.

public int getIndex(string input)	Gets index of task to be edited in file.
public void editTask(string input)	Edits task using index in file.

OperationHandler.cs: Takes in the first word of the command that has been entered by the user, and hands over control to the class of the respective operation.

static string extractFirstWord(string input)	Extracts first word of input to get command type.
public static void determineCommand(string input)	Determines which class to hand input to be processed.

Parser.cs: Determines the way that the command entered by the user is interpreted by the software. It breaks down the command into the attributes of the Task class, and returns the Task object to the class from where the Task object had been called.

private string shortcutsToString(Utility.shortcuts current)	Adds hyphen to input.
private string parseEndDate(string input)	Gets end date from input.
private string parseEndTime(string input)	Gets end time from input.
private string parseTaskDescription(string input)	Gets task description from input.

parseTag(string input)	Gets tag from input.
private void combineDateTime(string input)	Combines date and time into a single string.
public Task returnTask(string input)	Gets Task object from parsed input.
private static void isEndDateTodayOrTomorrow(ref string endDate)	Checks if task's end date is today or tomorrow, based on system time.
private static int indexBeforeNextHyphen(string input, int startPosition, int endPosition)	Locates next hyphen in string.

Storage.cs: Links the software with the Data.txt and is used to write to and delete data from the aforementioned file. It contains a working vector.

public void addData(Task newTask)	Adds new tasks to taskList.
public void removeData(int index)	Removes task from taskList.
public List<Task> readFromFile()	Reads data from text file and returns data as a List of Task objects.
public void writeToFile()	Writes data from given List of Task objects and stores it as a text file.

Task.cs: Defines what a task is supposed to contain, and is used to set/get the attributes that a normal task contains.

public void setEnd()	Sets date of task due date.
public void setTaskDescription()	Sets bulk of task info.
public void setIsStarred()	Sets importance of task.
public void setCategory()	Sets type of task.
public void setTag	Sets tag.

Program.cs: Sets the program running when the .exe file is opened. Control is then handed over to the UI class.

UI.cs: Contains Windows form object initializations, event triggers and consequent functions, keyboard shortcuts and windows form component declarations.

UI.Designer.cs: Consists of the design for Alfred's GUI.

Utility.cs: Contains constants and enum types, to enhance code readability.

Code features & Instructions for testing

The features that have been implemented in Alfred thus far have been designed keeping in mind the heavy-keyboard user's convenience. That is why we have tried to keep it as simple and as command-line oriented as possible. The following features can be tried out in the ways mentioned in the following page –

I Creating a task:

- The "Add" keyword doesn't have to be mentioned.
- While there is some rigidity in mentioning "-ed", "-et" and "-t" to enter due date, time and tag respectively, the due date itself can be mentioned in a number of ways. Multiple formats such as 16 October 2012, 16 Oct 2012, 16/10/2012 and 16/10/12 can be used.
- While adding a task, if and when the end date, end time and tag are added, it can be done so in any order.
- Certain level of natural language processing (e.g – today, tomorrow) is also provided by the software.
- If the task is added correctly, a status message will be displayed – "Task added successfully!"

II Deleting a task:

- In the current version, only deletion by task number is supported.
- An error message supplied if an invalid delete command ("Please enter Delete <task number>") or an invalid task number ("Please enter a valid task number") is entered.
- If the task is deleted correctly, a status message will be displayed - "Task deleted successfully!"

III Editing a task:

- In the current version, only editing by task number is supported, and the entire task must be rewritten in the format used to add tasks.
- An error message is supplied if an invalid edit command ("Please enter Edit <valid task number> <-ed valid end date> <-et valid end time>") or an invalid task number ("Please enter a valid task number") is entered.
- If the task is deleted correctly, a status message will be displayed - "Task edited successfully!"

IV Miscellaneous:

- In general, an error message is supplied if an invalid command ("Please enter a valid command") is entered.
- An error message is supplied if invalid due date or time ("Please enter a valid deadline") is entered.
- Nifty keyboard shortcuts have been implemented to make Alfred easy-to-use. Ctrl + M minimizes the application to the system tray, with a notification. This can be brought back up again by double-clicking the icon in the tray. Ctrl + S expands/collapses the display, so that the list of existing tasks can either be seen or hidden.

The developer is expected to take note of the above and test each case rigorously.

Appendix

Logic for interpreting end date and end time:

End Date	End Time	Task Type	End
0	0	Floating Task	Maxvalue of DateTime
0	1	Deadline Task	End time of the current day or following day, based on current system time.
1	0	Deadline Task	End date, 23:59 of the given day.
1	1	Deadline Task	End date and end time as per input

Updated prospective project timeline:

Version No.	Implementation
0.2	Power Search, Undo, Redo, Support for recurring tasks.
0.3	Alarm Reminders, Google Calendar integration.
0.4	Intuitive GUI, Keyboard Shortcuts, Auto complete, Prompts.
0.5	Final Revision, Bug Fixes, Optimization.