CataList

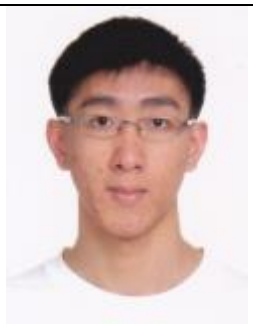
		
MITCHELLE ELAINE JUAW SHIAN FERN Leader, Head Plans, Tester	CHUA CHIN TAT Time Keeper, Head Programmer	KHOR POO SIANG Debugger, Documentation, Planner, Google Group Coordinator

Table of Contents

User Guide

A Catalyst is a substance in which, when present, it aids in the experiment, causing an increase in the rate of the chemical reaction. While our product is not related to chemistry in any form, but our product is a simple tool, such that when used, it increases the productivity of the user and manage the user's time more efficiently, therefore, this manual would briefly explain the functions of CataList.

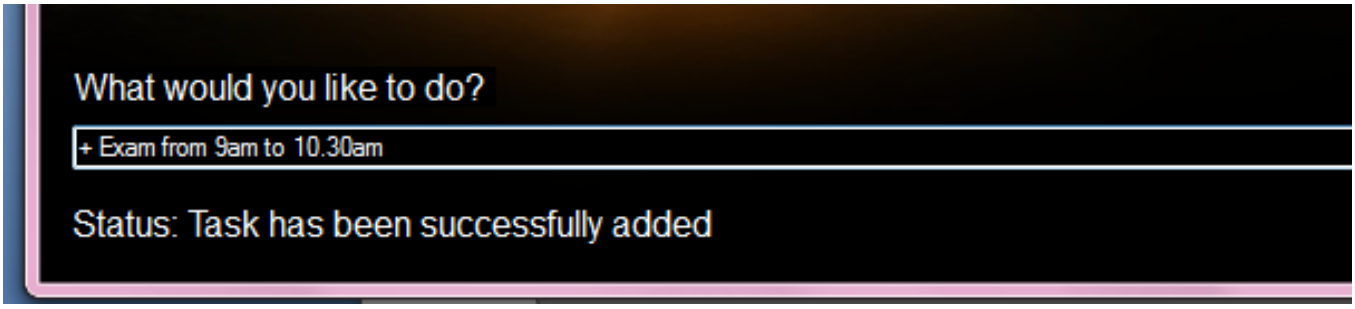
CataList serves to help users ease day to day activities by effectively taking note of tasks and appointments. It would allow the user to spend less time trying to remember what to do, and concentrate on getting things done.

CataList would include the below mentioned features such as:

Add new task	• Add new task with or without a deadline
Delete task	• To remove previously added task or appointment
Update task	• To change or edit previously added task's details
Find task	•Searches list of tasks for requested task and display available matches
Undo	• To undo the changes done previously
Redo	•To revert back to previous operations which have been undone
Block	• To block dates which user would like to reserve
Unblock	• To remove blocked date
Done	• To mark tasks as "done"
Archive	• Allows user to view tasks which have expired
Completed	• To display all tasks marked as done
Display	• To display all added tasks
Save and Exit	• To quit the program after user has finish editing

Command Structures of CataList

Adding a new timed and dated task



To add a task, the user can simply type:

*"**+** dinner with john from 9pm to 10pm 16/11/13"* or

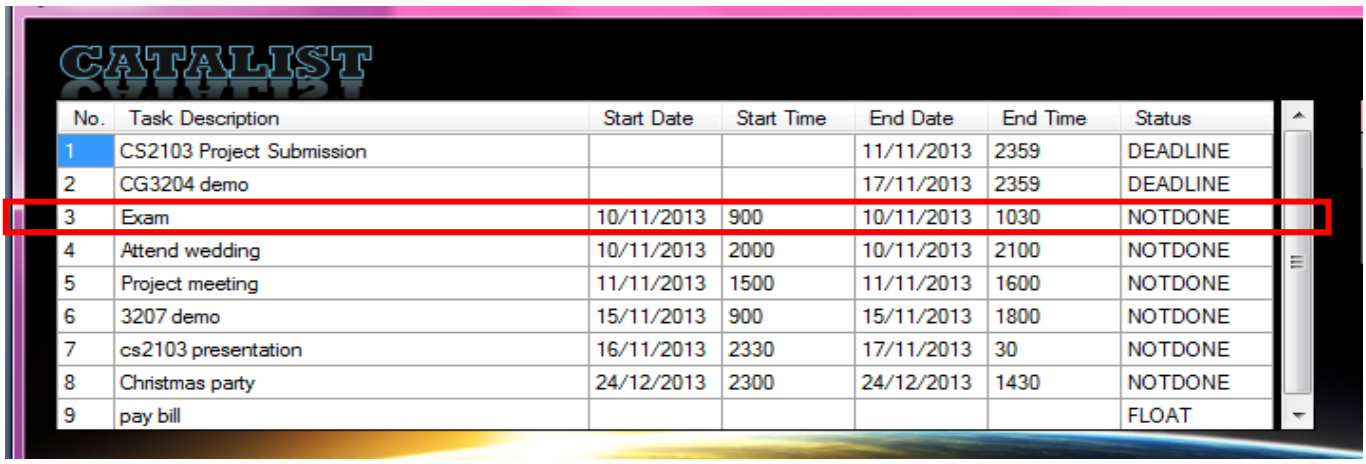
*"**add** attend seminar 1100hrs next sunday"* *or*

*"**todo** reading week next monday to friday"* *or*

"add medical check-up coming thursday"

Catalist provides user with more flexibility for the input of tasks or appointment such that user can also add new tasks or appointments simply by starting sentence with "**todo**" or "**reminder**".

The added task will then be display on the screen ordered according to dates.



No.	Task Description	Start Date	Start Time	End Date	End Time	Status
1	CS2103 Project Submission			11/11/2013	2359	DEADLINE
2	CG3204 demo			17/11/2013	2359	DEADLINE
3	Exam	10/11/2013	900	10/11/2013	1030	NOTDONE
4	Attend wedding	10/11/2013	2000	10/11/2013	2100	NOTDONE
5	Project meeting	11/11/2013	1500	11/11/2013	1600	NOTDONE
6	3207 demo	15/11/2013	900	15/11/2013	1800	NOTDONE
7	cs2103 presentation	16/11/2013	2330	17/11/2013	30	NOTDONE
8	Christmas party	24/12/2013	2300	24/12/2013	1430	NOTDONE
9	pay bill					FLOAT

Adding a new deadline

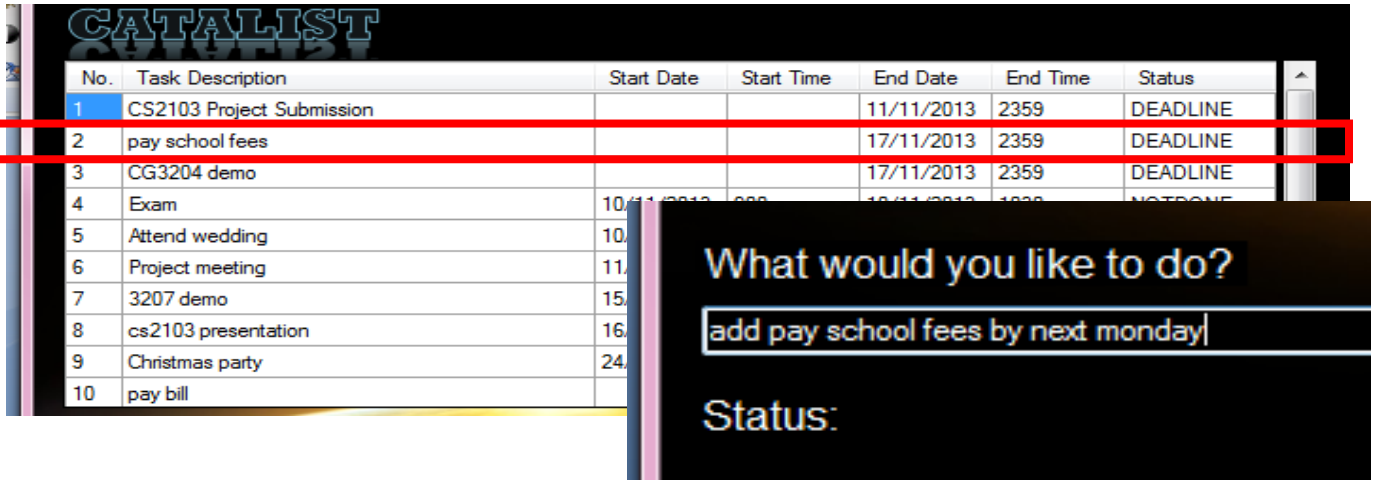
Catalist also allows user to add deadline task without a specified start date.

Possible ways of entering a deadline task:

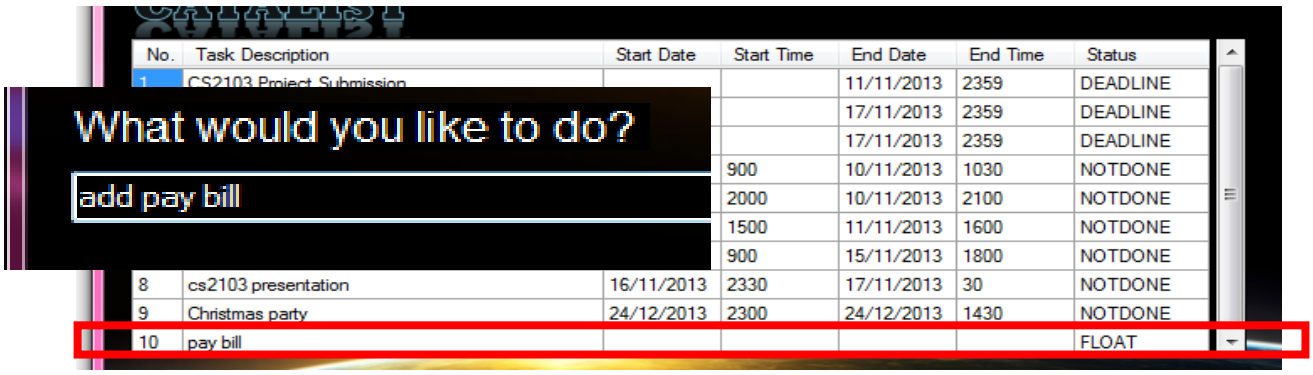
*"**todo** CS1231 Project Submission by next monday"*

*"**add** resume submission by this monday"*

*"**reminder** IPPT by 12/12/2013"*



To add floating task

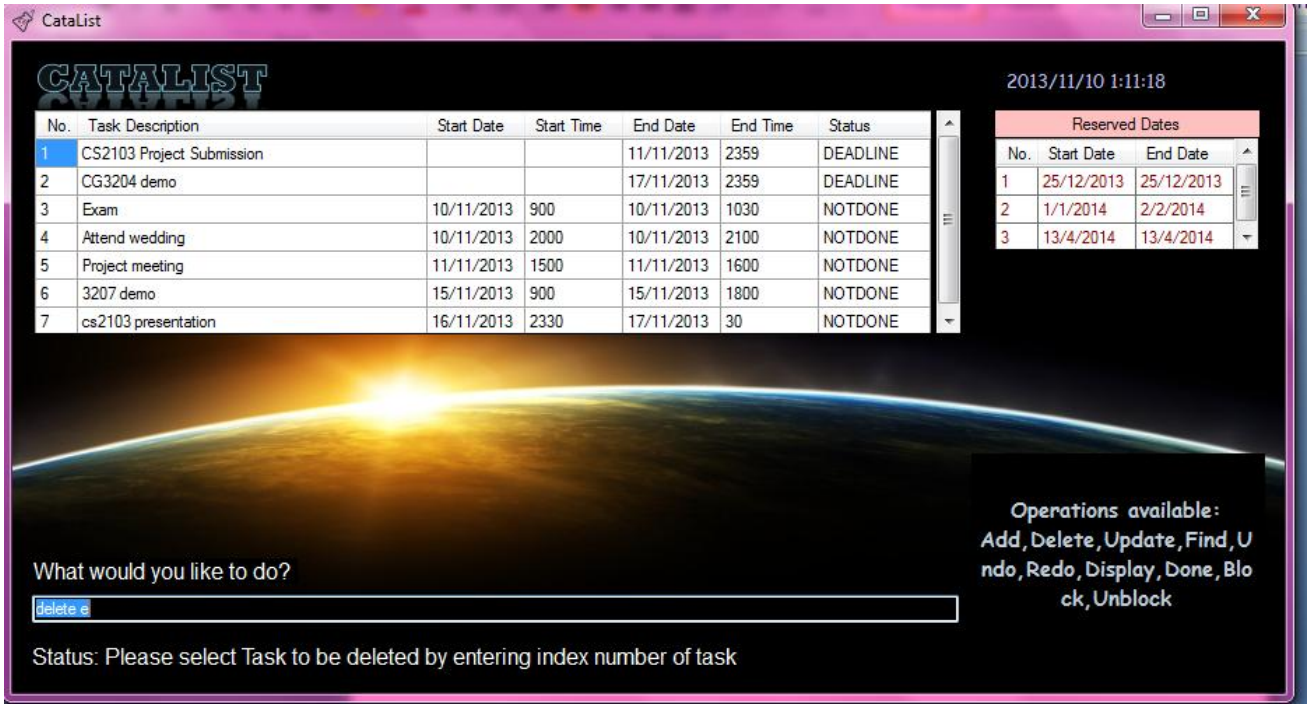


Catalist also supports the addition of floating task (i.e. task with no specific date and time)

For example:

*"**todo** CS2103 tutorial assignments"*

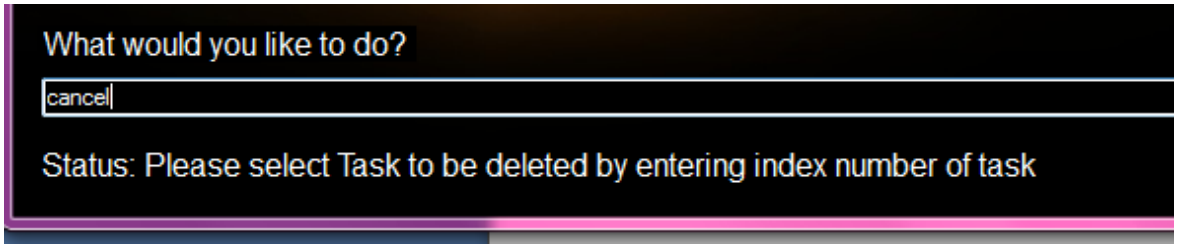
To remove an added task



To remove an entry, CataList provides user with the option to use the symbol "**-**"or simply typing "**delete**" or "**remove**":

*"**-** dinner"* or *"**remove** dinner with family"*

We understand that user might not always remember the full task details, thus the Catalist will attempt to look for related matches to what user wish to delete and allow choosing of task to be deleted simply by entering the index number.

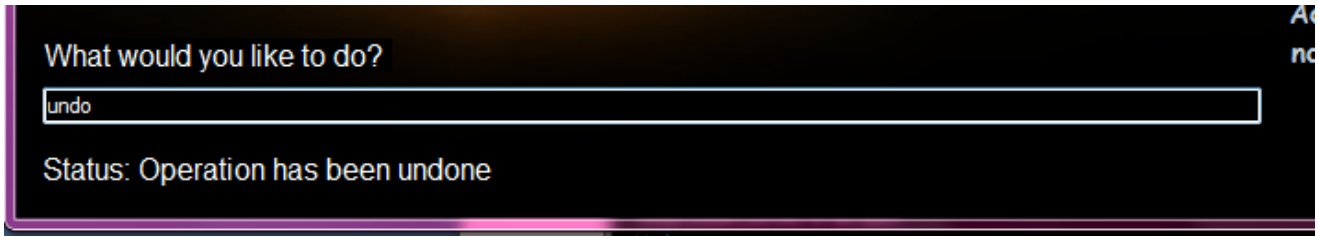


If user wish to cancel the delete operation anytime , they can simply enter "**cancel"** or "**back**" to return to the main display.

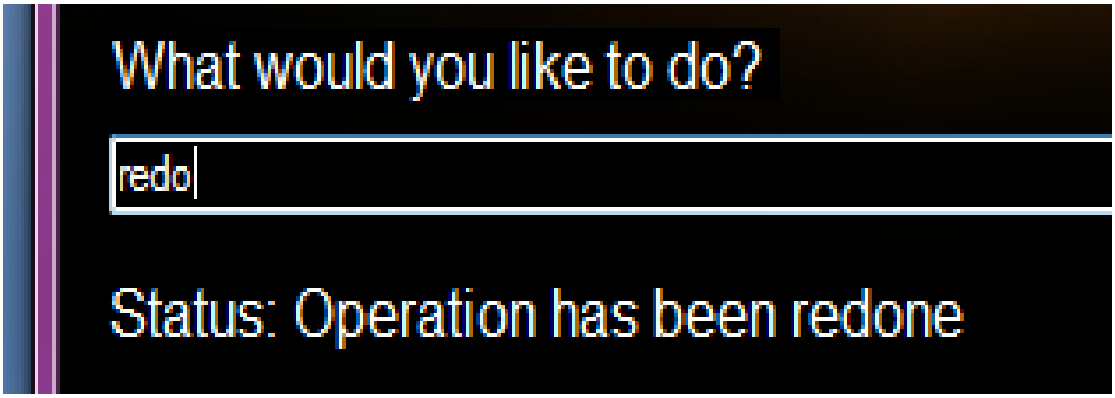
To undo and redo tasks

If user decided to undo his/her last action, simply enter:

"undo"



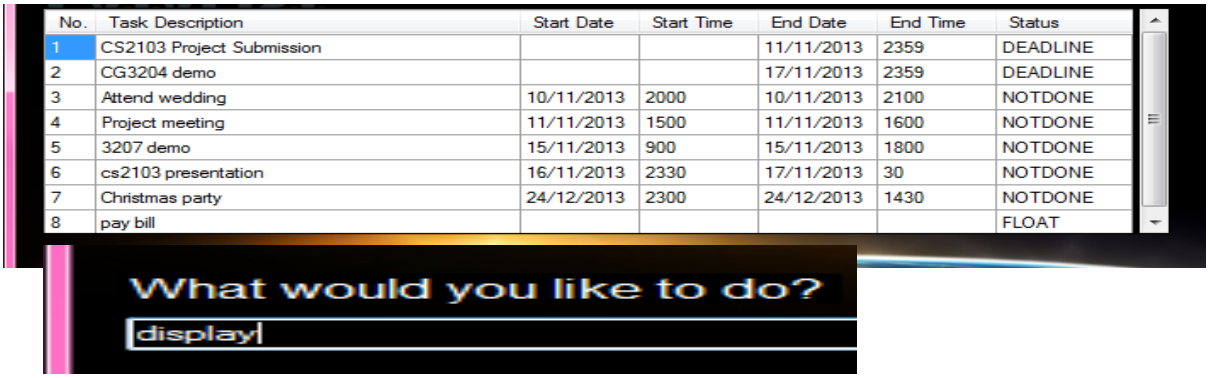
Just in case user decided to change their mind after an operation has been undone, Catalist also provides user with the "**redo**" feature.



To show or display tasks

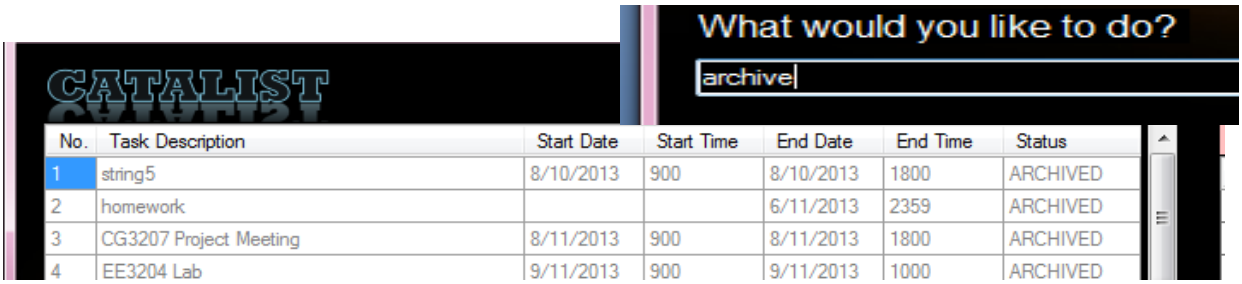
To view all tasks added into the list, user can enter:

"show" *or* ***"display"***



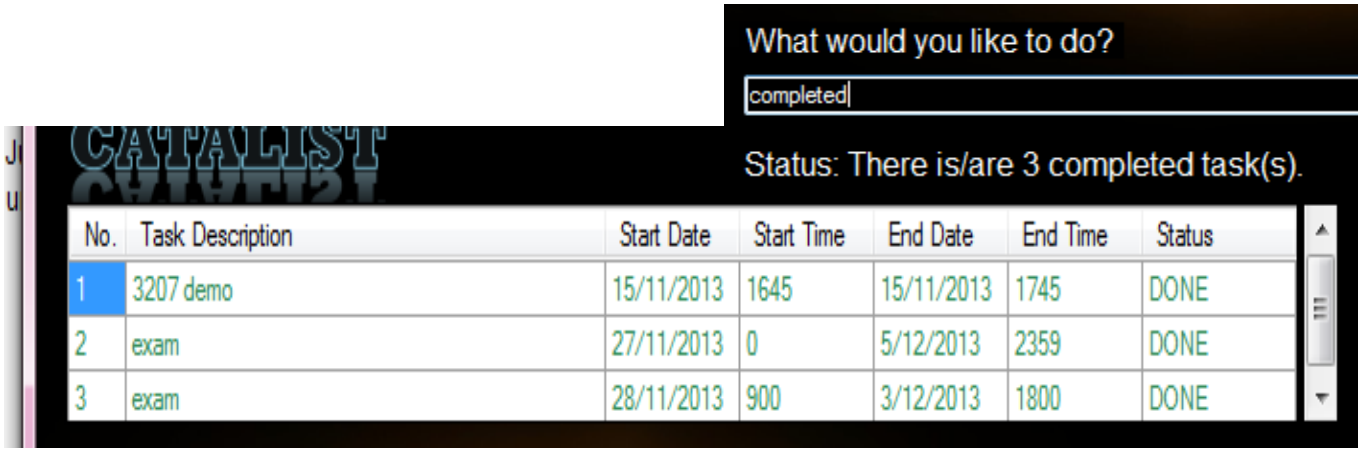
Other than that, user can also view expired tasks by entering:

"archive"



User can view completed tasks by typing:

"completed"



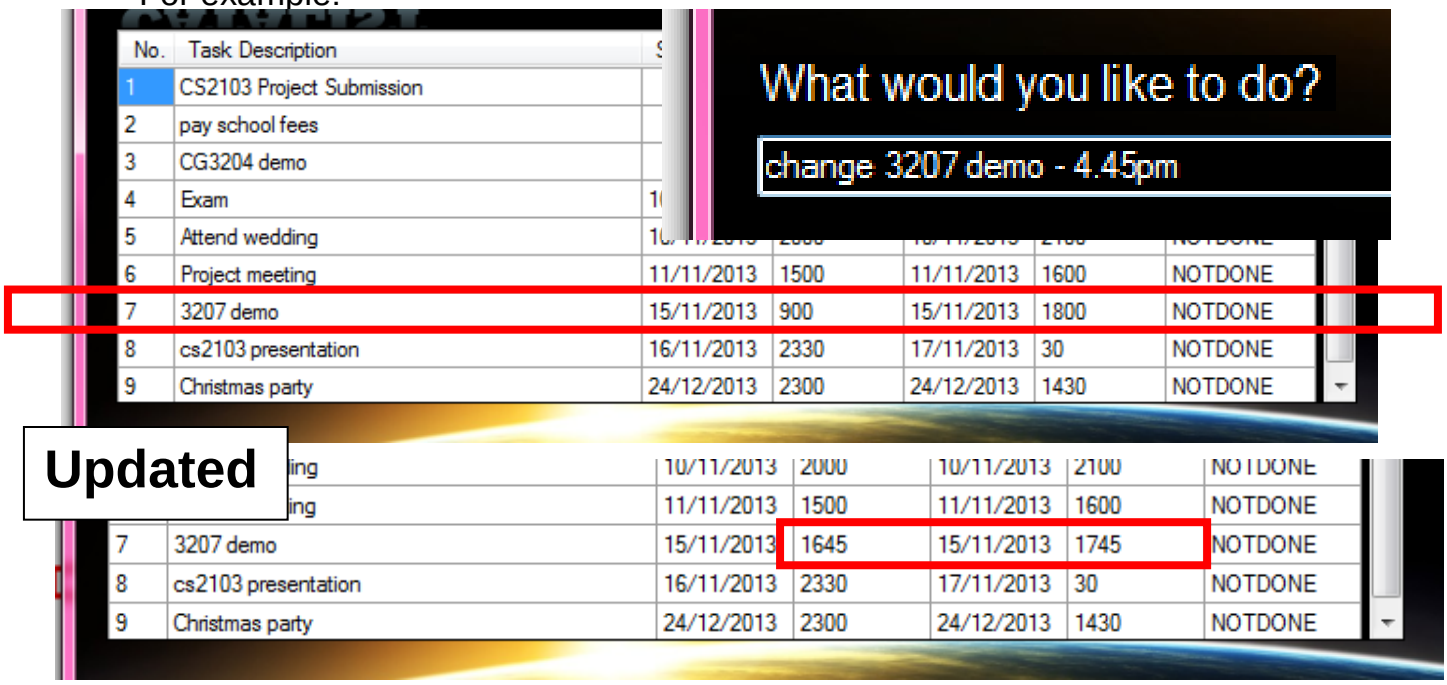
To update an added task

Catalist provides user with the update feature that allows user to update details of a previously added task.

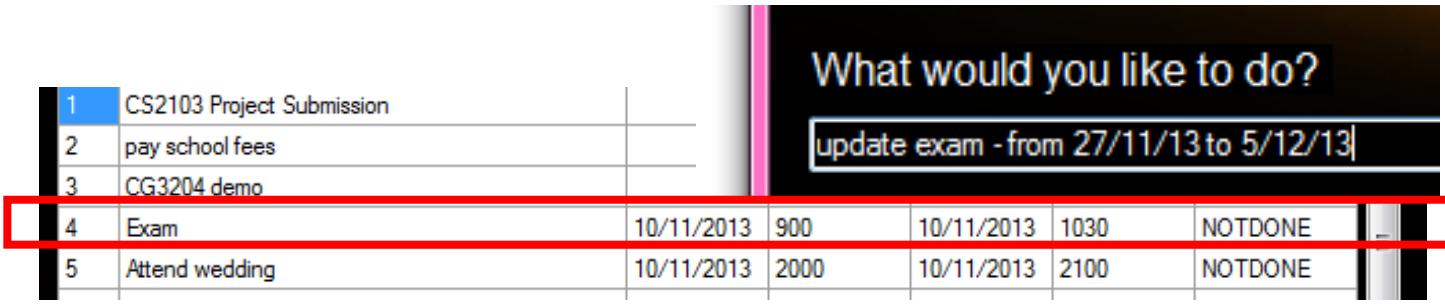
User can update the start date, start time, end date, end time as well as the task description with this feature by simply entering:

"**update** task description – new info" or "**change** task description – new info"

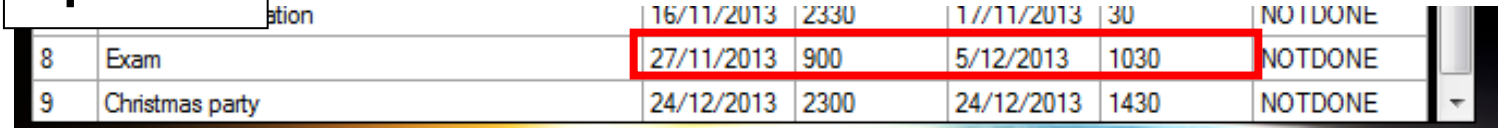
For example:



Updated



Updated

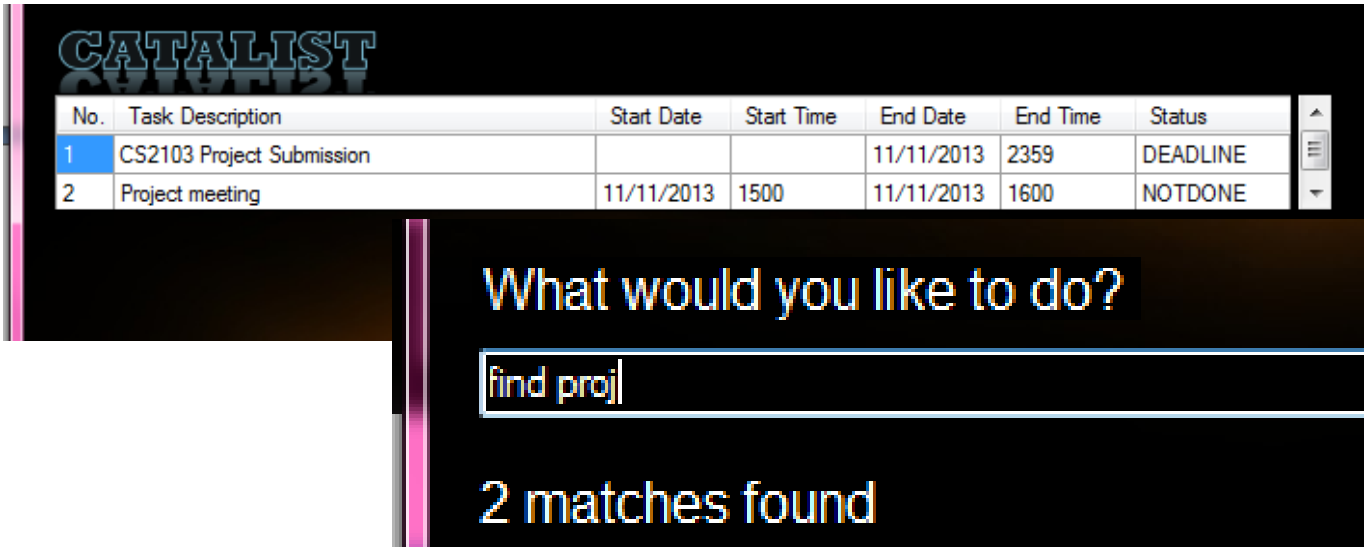


To search for an added task

Catalist also provides an extensive search feature for user to effectively look for any task they wish to view specifically.

User can search for task by entering:

"**Search** abc " or "**find** xyz"



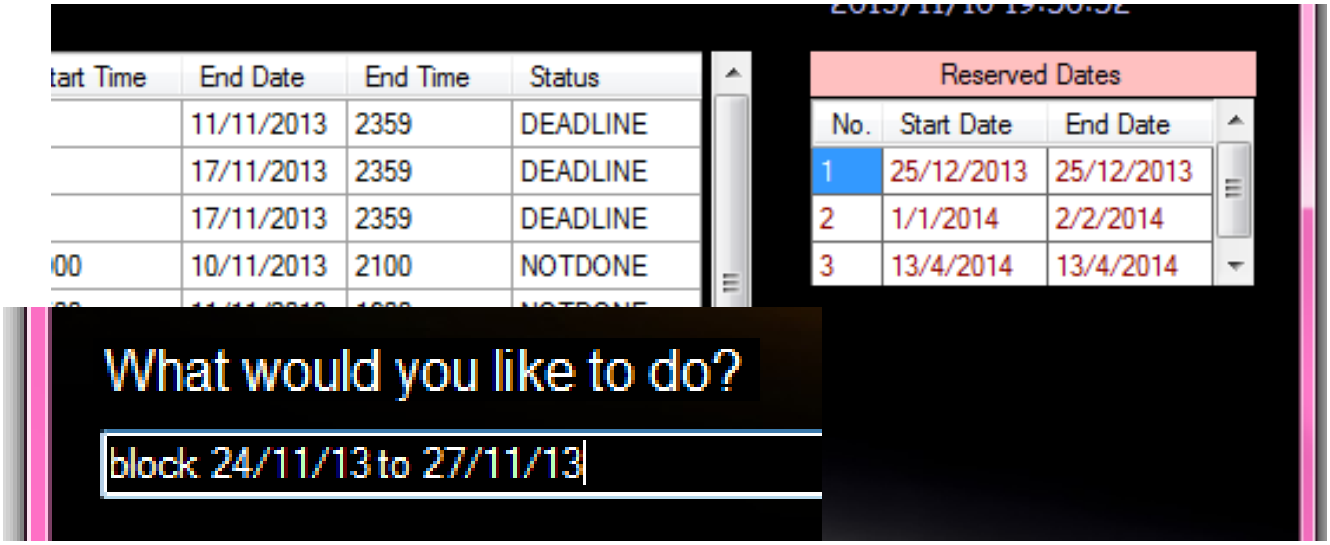
Catalist's search mechanism does not require user to type the full task description. Instead, user can enter any keyword or alphabet related to the desired task.

To block or unblock dates

User is also provided with the feature to block future dates which user wishes to reserve. After the dates have been added into the reserved date list, adding of task with dates that clashes with a reserved date will be disabled.

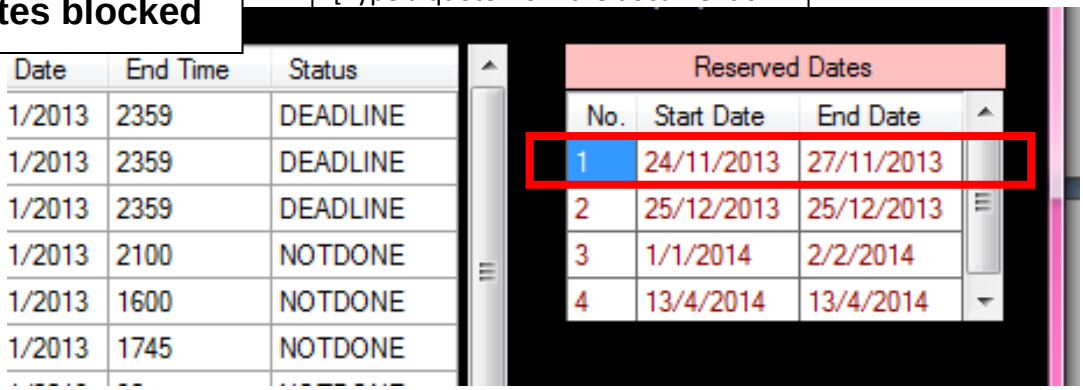
User can do so by entering:

"**block** 21/11/13 to 25/12/13" or "**block** 1/1/2014"



Dates blocked

[Type a quote from the document or



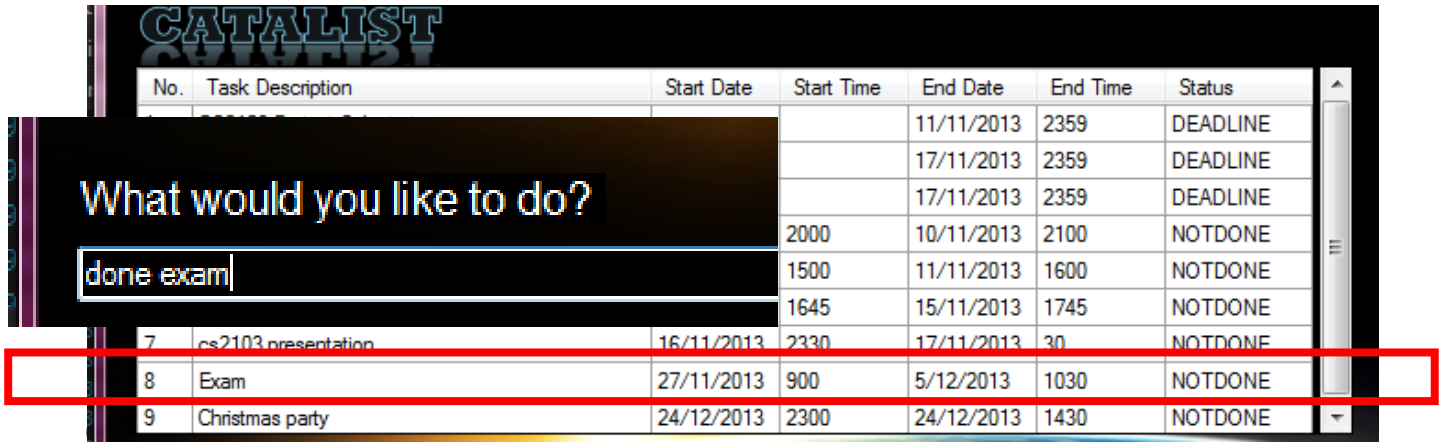
If user wish to remove any reserved date, he can simply enter sentence starting with unblock followed by the index number, for example:

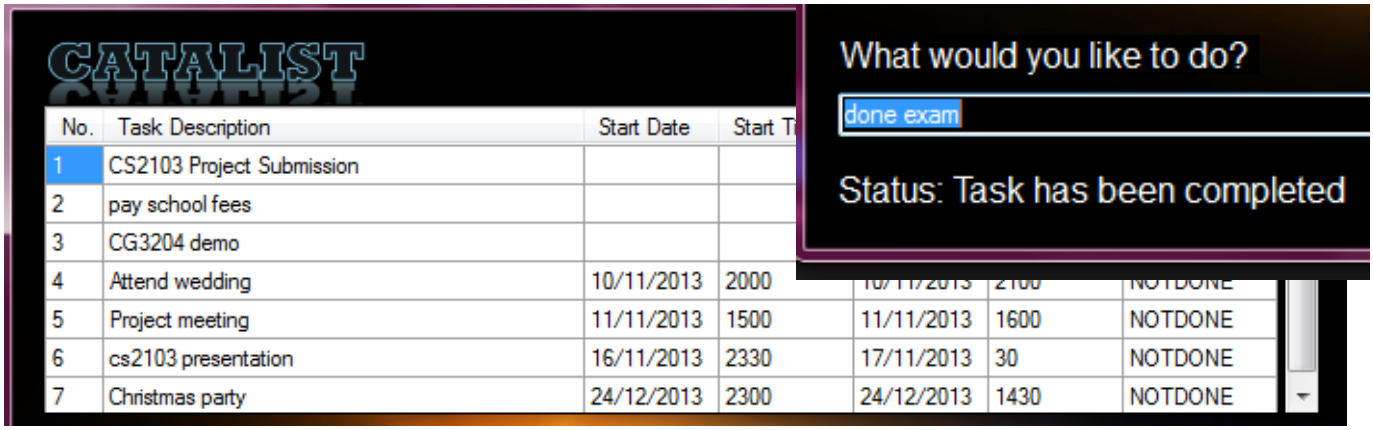
"**unblock** 1"

To mark task as completed

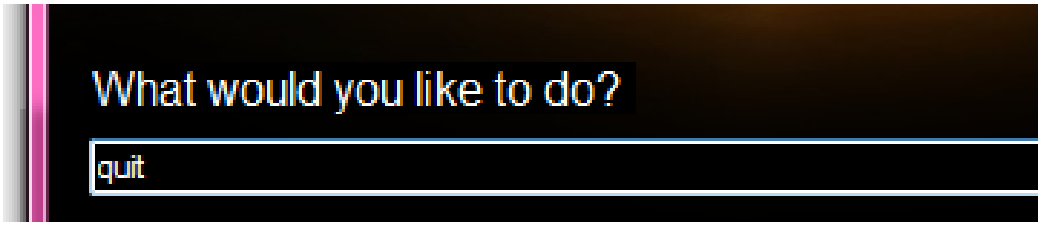
User can mark tasks that are completed by entering “done” followed by task description or keyword:

“**done** exam” or “**done** e”





To quit and save



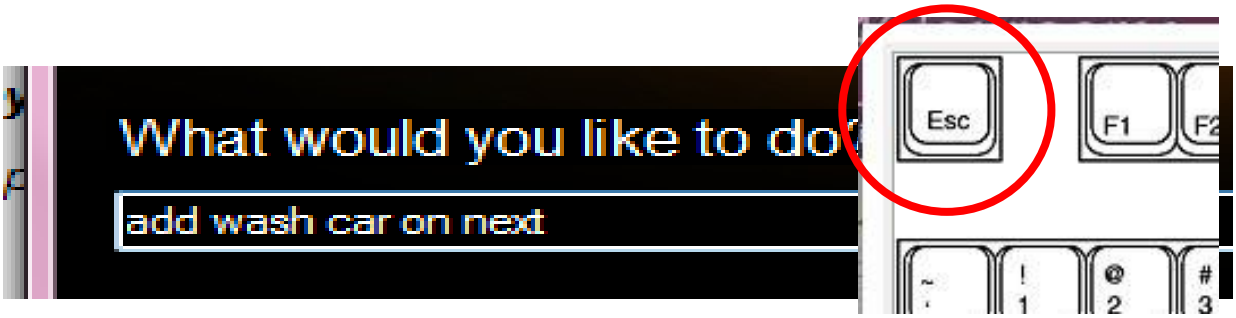
To save and exit the program, user can simply type:

“ **exit** “ or “**close**”or “**quit**”

Additional keyboard shortcuts

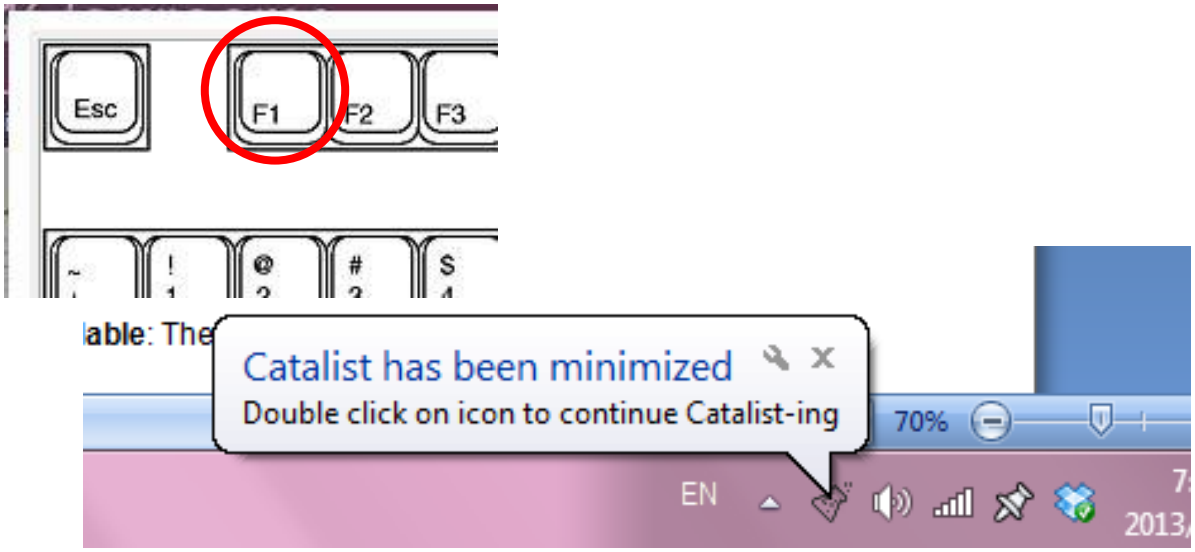
Clearing inputs

To clear the input box during operation, simply press the “Escape” key.



Minimizing Catalist

In case of emergencies, user can press the “F1” key to minimize Catalist to system tray. It will continue running in the background after minimization.



Developer Manual

Scope

CataList is a windows based application that helps user to put their todo items into a systematic process that tracks them and helps the user to decide what and when to do things so that they can keep their mind clear of todo items. All the operations mentioned will be done by user via keyboard.

Project constraints

- **Constraint-Desktop**: CataList works on a stand-alone desktop without Internet connection.
- **Constraint-CLI**: Command Line Interface is the primary mode of input.
- **Constraint-Standalone**: CataList is stand-alone and not a plug-in to another software.
- **Constraint-No-Database**: CataList should not use relational databases. Data storage must be done using text files.
- **Constraint-Readable**: The data is be stored locally and in a human readable text file.
- **Constraint-OO**: A significant part of CataList follows the Object-oriented paradigm. However, some parts of the application can be non-OO.
- **Constraint-Windows**: CataList works on the Windows 7 or Windows Vista OS. It also works on both 32bit and 64bit PCs.
- **Constraint-External-Software**: The use of a third-party framework/library may be allowed only if it is free, does not require installation by user.

Setting Up

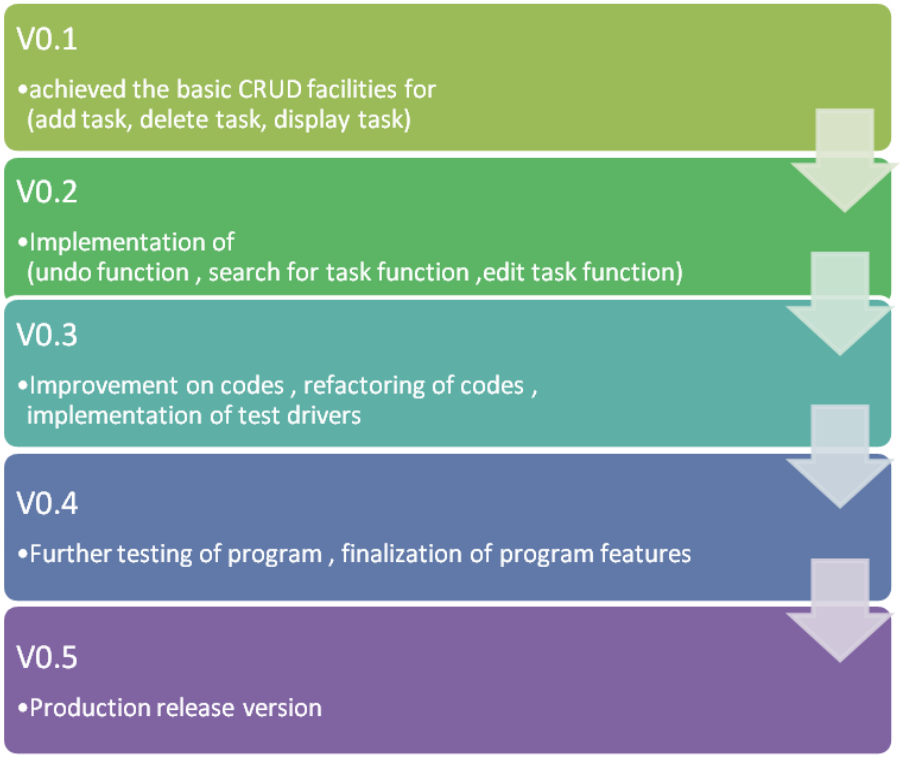
Development environment

- **TortoiseHg**
- **Google Code** project hosting.
- **C++** as primary language
- **Windows Form** for designing of GUI

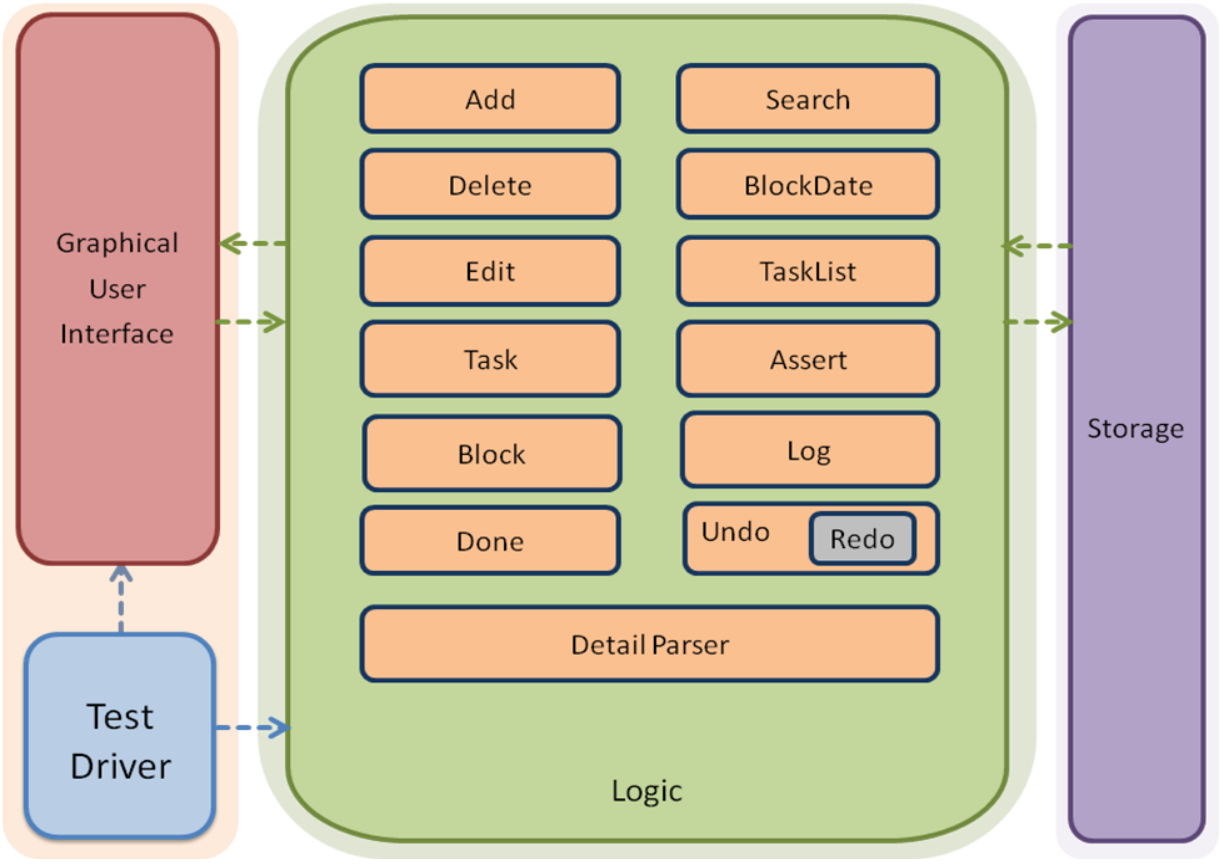
Prerequisites

- Install TortoiseHg latest stable version.
- To clone the source code, request permissions from current project members to be added into google code project.
- http://code.google.com/p/cs2103aug2013-t15-1c/
- If you are using TortoiseHg to clone, simply copy the source URL and paste it in the Source field. Alternatively, you can use the command line to do the cloning.
- Install the latest C++ IDE.
- Install the latest googletest testing framework
- For more information regarding project request access to the team wiki space. http://cs2103.wikispaces.com/cs2103
- Happy coding!

Progress of project



Main Architecture



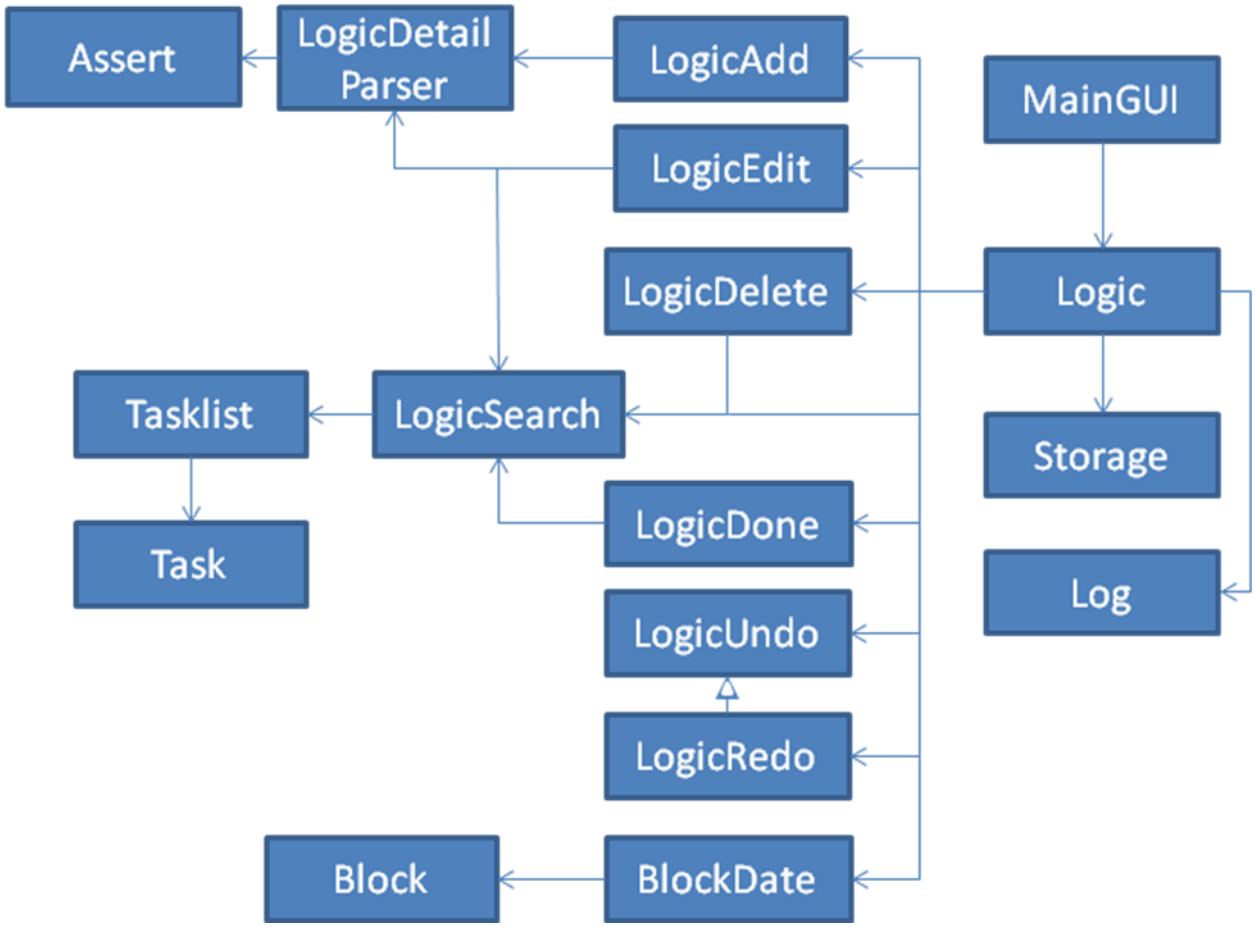
The design of the Catalist architecture is based on the **façade pattern**, where Logic shields all the internal components from GUI.CataList is a scaled-down version of 'a Siri for keyboards'. I.e. tool that is designed to imitate natural language processing (NLP) via keyboard. Given above is an overview of the main components.

- **Graphical User Interface**: The GUI seen by users handles operation to request for user input, display of the output and status of operations.
- **Logic**: The main logic of the program is situated here. Logic handles all interaction between internal components and GUI.
- **Storage**: Storage consists of 4 text files. Namely,PendingTask.txt; archived.txt; completedTask.txt;blockDetails.txt
- **Test Driver**: CataList will be utilizing Gtest framework for the unit testing of all components

Class Diagram

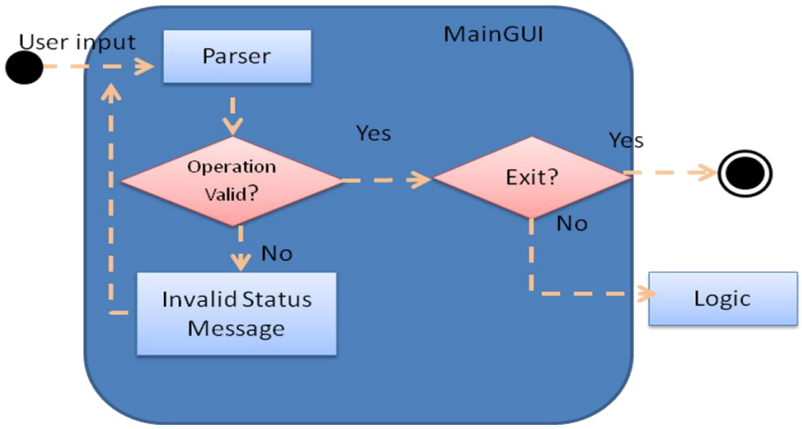
This class diagram depicts the navigability between the classes and also provides an overview to the classes available in Catalist as well as the interaction between them.

The main principle applied in the design of Catalist is the principle of Separation of Concerns. The application of this principle can be seen from the following class diagram where different feature of Catalist has been separated into different classes as they are of different concerns and objective thus should be taken as different components.



Graphical User Interface

Activity diagram



Parser: this is the main function in MainGUI that parses the user input into "operation" and "taskdetails" which will then initiate functions in Logic Class accordingly.

Main display components

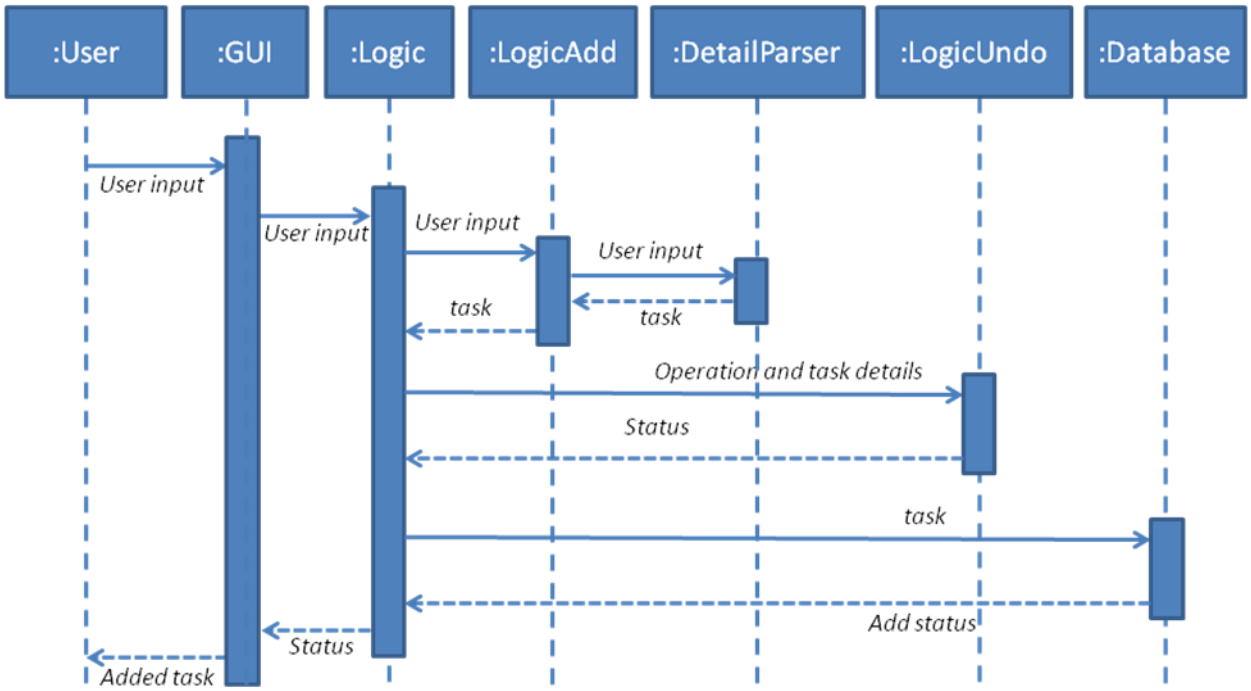
GUI utilizes Data Grid View Table for the main display of items in list pass as reference from Logic Class.

Some example of Data Grid View codes:

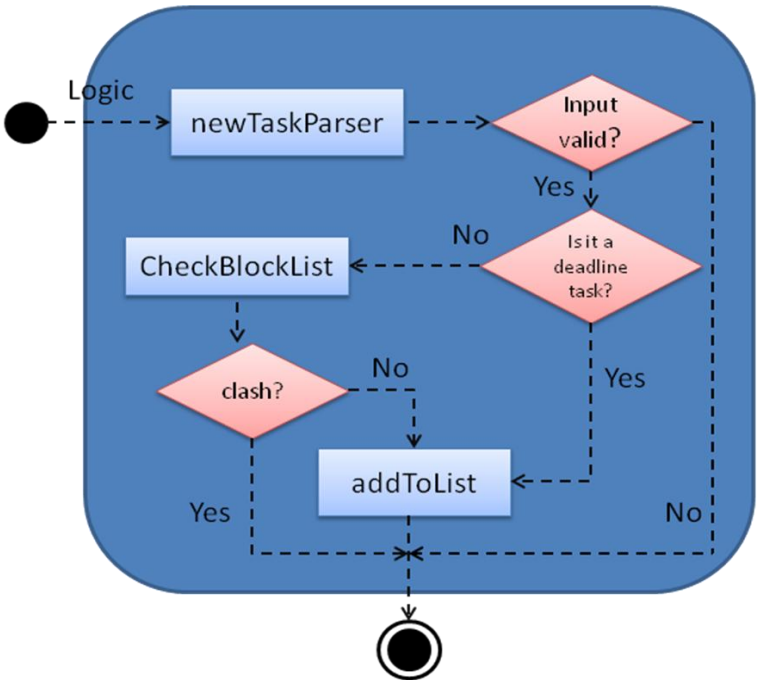
```
void MainGUI::showBlockData(list<Block> blockInfoList){
    this->dataGridView2->ForeColor= System::Drawing::Color::DarkRed;
    dataGridView2->Rows->Clear();
    //label3->Text = "Status:";
    BL= gcnew ArrayList;
    int listSize = blockInfoList.size();

    for(int i=0;i<listSize;i++)
    {
        _block = new Block();
        *_block = blockInfoList.front();

        array<String^>^row = gcnew array<String^>{
            gcnew String((i+1).ToString()),
                gcnew String(_block->getStartDay().ToString()+"/"+_block->getStartMonth().ToString()+"/
                gcnew String(_block->getEndDay().ToString()+"/"+_block->getEndMonth().ToString()+"/"+_b
        };
        dataGridView2->Rows->Add(row);
        blockInfoList.pop_front();
    }
```

LogicAdd

Sequence diagram for adding new task

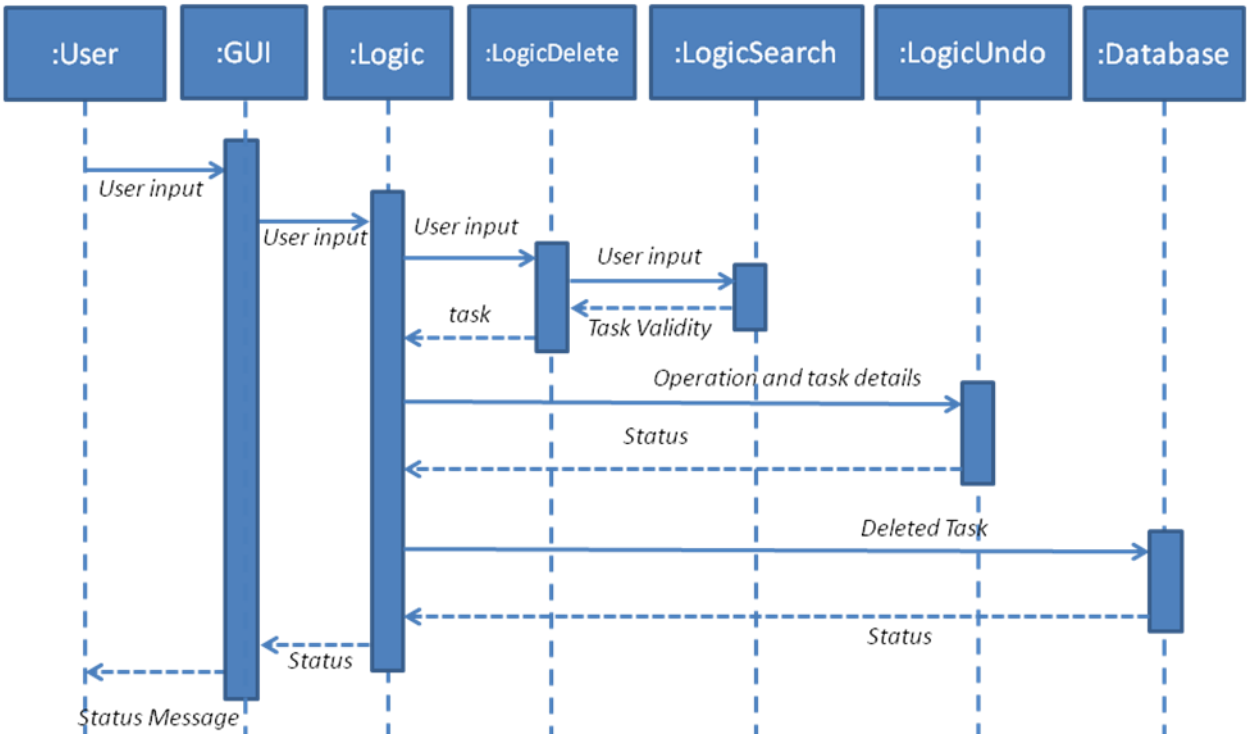


Activity Diagram

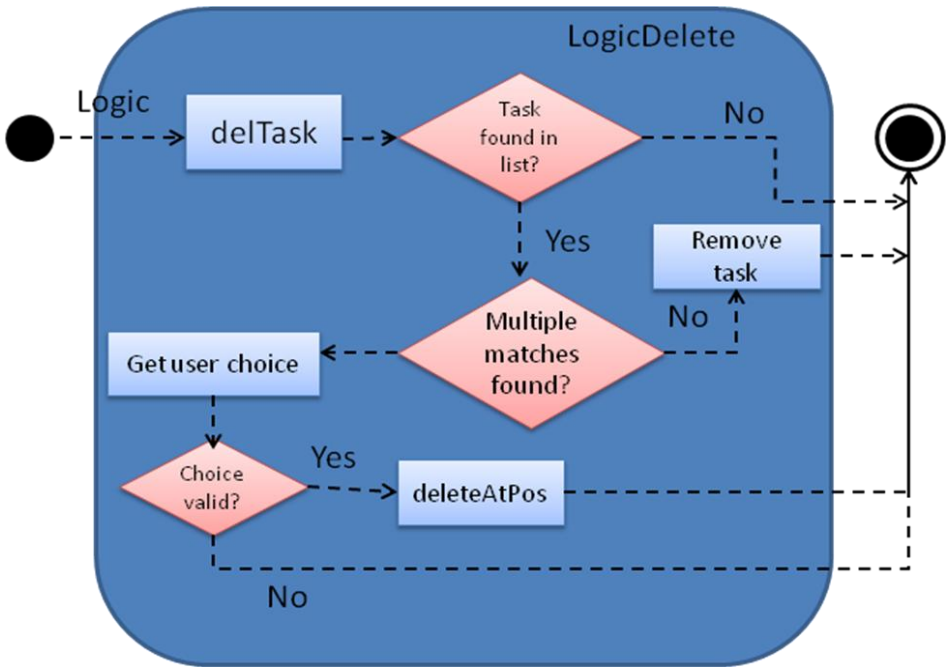


LogicDelete

Sequence diagram for deleting task

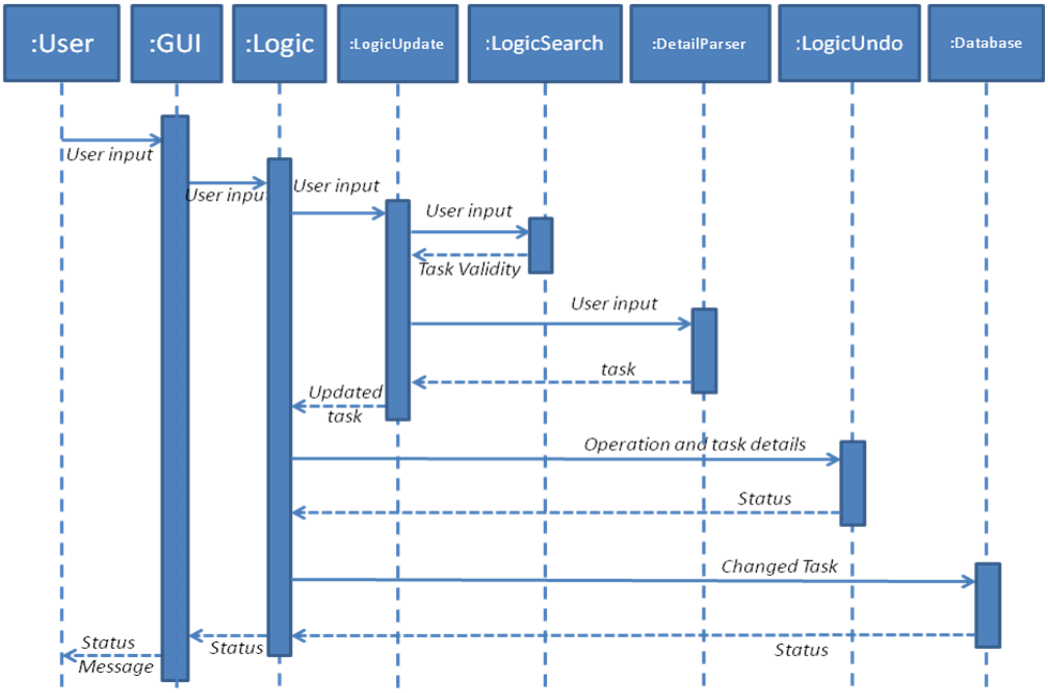


Activity Diagram

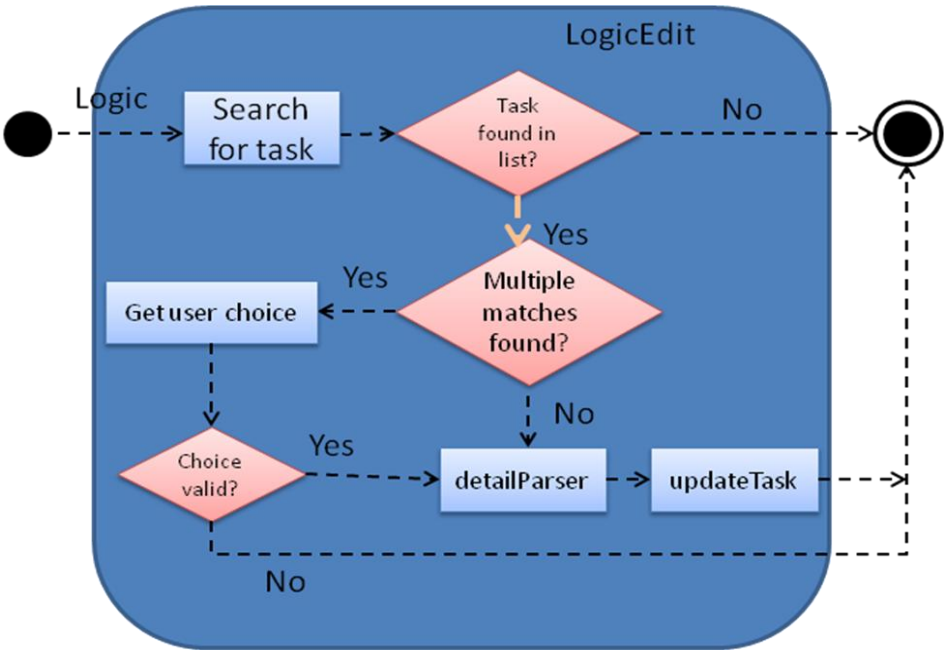


LogicEdit

Sequence diagram for Modifying added task

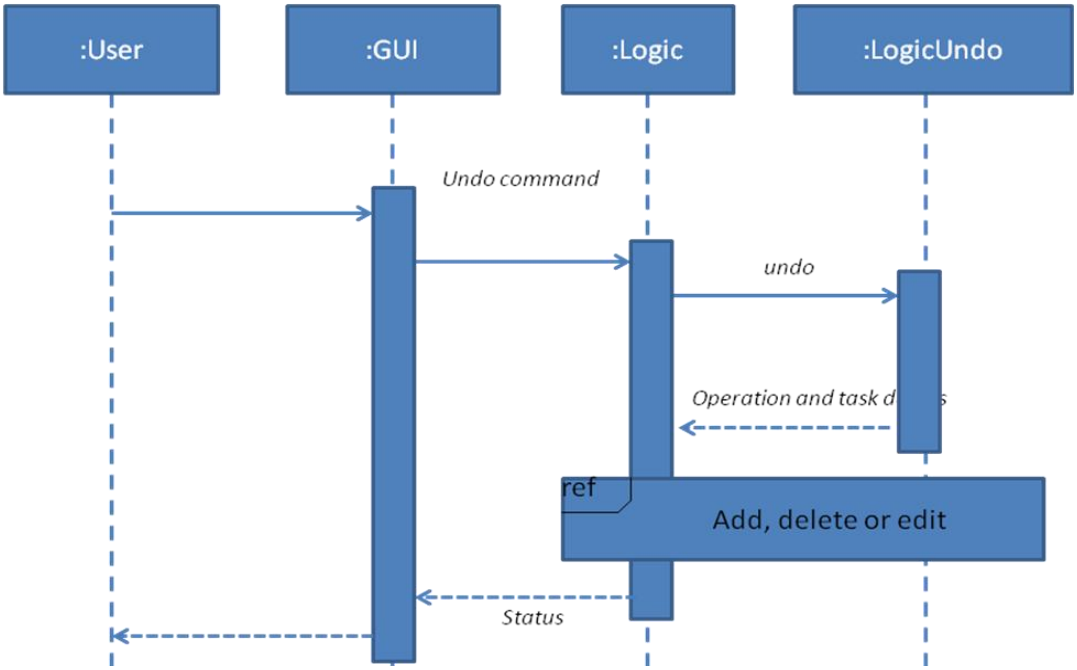


Activity Diagram

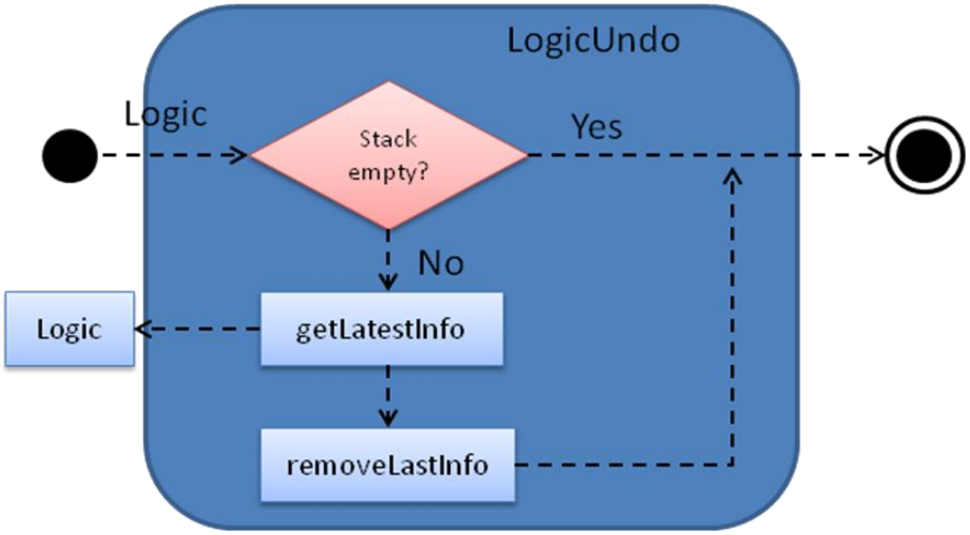


LogicUndo

Sequence diagram for task undo



Activity Diagram



Tool used in testing

Testing of CataList will be done done using the Google Gtest. Following is the steps to setting up of the gtest.

Google Test

Get Google C++ Testing Framework

1. Download the latest gtest framework
2. Unzip to `C:\gtest`

Build the Framework Libraries

1. Open `C:\gtest\msvc\gtest.sln` in Visual Studio
2. Set Configuration to "Debug"
3. Build Solution

Create and Configure Your Test Project

1. Create a new solution and choose the template Visual C++ > Win32 > Win32 Console Application
2. Right click the newly created project and choose Properties
3. Change Configuration to Debug.
4. Configuration Properties > C/C++ > General > Additional Include Directories: Add `C:\gtest\include`
5. Configuration Properties > C/C++ > Code Generation > Runtime Library: If your code links to a runtime DLL, choose Multi-threaded Debug DLL (/MDd). If not, choose Multi-threaded Debug (/MTd).
6. Configuration Properties > Linker > General > Additional Library Directories: Add `C:\gtest\msvc\gtest\Debug` or `C:\gtest\msvc\gtest-md\Debug`, depending on the location of gtestd.lib
7. Configuration Properties > Linker > Input > Additional Dependencies: Add `gtestd.lib`

Verifying Everything Works

1. Open the cpp in your Test Project containing the `main()` function.
2. Paste the following code:

```
#include "stdafx.h"
#include <iostream>

#include "gtest/gtest.h"
```