**developer guide for the following class(.cs file):**

Logic.cs, Operation.cs, OperationExecution.cs, OperationHandler.cs, Program.cs,

Storage.cs, Task.cs, TaskList.cs

**Architecture:**

The program serves as the main entry of the whole project. It will call UI and then Logic will be called. The creation of Logic will be followed by creation of OperationHandler.

The Logic control the sequence of command processing: first call commandparser to parse command, then get operation from it and pass operation to OperationHandler to execute then get result, which can serve as feedbacks to user in UI.

The Operation is definition for different types of operations. The OperationHandler is a class which can be inherited by class in OperationExecution, which can choose type for operation to execute.

The OperationExecution execute operation and get feedbacks.

The Task is definition of different types of tasks, TaskList is list of Task.

The storage save information of tasks into xml files.

**important APIs:**

***Logic*:**

Constructor: public Logic().

Function: string ProcessCommand(string input).

***Operation:***

class OperationAdd:

Constructor: OperationAdd(Task setTask) .

class OperationSearch:

Constructor: OperationSearch(string searchCondition).

class OperationDelete:

Constructor: OperationDelete(int DeleteIndex).

class OperationModify:

Constructor: OperationModify(int Previous, Task Revised).

class OperationUndo:

Constructor: OperationUndo().

***OperationHandler***

Constructor: public OperationHandler().

Function: virtual string ExecuteOperation(Operation operation)

***OperationExecution:***

class ExecuteAdd : OperationHandler

class ExecuteDelete : OperationHandler

class ExecuteModify : OperationHandler

class ExecuteUndo : OperationHandler

class ExecuteSearch : OperationHandler

***Task:***

Constructor:

public TaskFloating(string TaskName)

public TaskDeadline(string TaskName, DateTime EndTime)

public TaskTimed(string TaskName, DateTime StartTime, DateTime EndTime)

***Storage:***

Constructor: public Storage().

Function: void WriteXML(TaskList taskList).

**Design descriptions**

**(class diagrams, sequence diagrams, notable algorithms ...)**

**Code examples**

**Instructions for testing**

Since no unit test has been written, simple test can be implemented using code which has been commented in the program.cs.

These codes can check add, display and delete function and get feedback in console.