TE3201 Introduction to Software Engineering  
AY2021 Semester 2

**Project report**

**Name**: Lee Weihan Darren

**Student Number**: A0194169R

|  |
| --- |
| **User stories** Give some user stories (5-10) that match your project. Hint: refer to the textbook to find the meaning of ‘user stories’  [Epic] As a user, I want a tool to help me note down my tasks:   1. As a user, I want to be able to quickly add basic tasks so that I will remember what I need to do at different times of the day. 2. As a user, I want to be able to add more complex tasks with a deadline in the future so that I can keep track of what I have to do by a certain date 3. As a user, upon accomplishment of a certain task, I want to be able to mark it as done. 4. As a user, sometimes a task marked ‘done’ will have to be reopened again. I would have to have a way of changing the task status to undone. 5. As a user, I want to be given a choice to delete tasks so that my task list is streamlined.   [Epic] As a student, I have different needs for my task tool:   1. As a student, I can be on the move frequently. I want to my tasks saved when I shut down my computer, and yet view them again when I switch it on. 2. As a student, I want to be able to view my task progress at the end of the day so that I can track my performance. 3. As a student, I want to able to perform multiple operations on them so that I do not have to spend time doing repetitive actions. 4. As a student, I would like an interface to manage all tasks without delving into programming or the command line interface (CLI). |

|  |
| --- |
| **NFRs** Give some non-functional requirements (3-5) for your project.  Technical requirements:   1. Users should be able to add task list of arbitrary length. 2. Distinct error messages should be provided if the user enters in the wrong input.   Quality requirements:   1. Tool should be usable by a novice with no prior knowledge of how the tool would work. 2. Tool should be robust to unexpected input.   Product should be scalable and extensible by other developers if required. |
| **Welcome** Show the output Duke shows when you launch the program. |
| **Adding** Describe adding different types of task Note: ‘describe’ here (and in the subsequent sub-sections) means give examples of user commands and the app’s response for those commands. You may use screenshots.  Regular task:   * todo + TASK. (e.g. todo read book)     Deadline task:   * deadline + ‘by:’ + TASK. (e.g. deadline return book by: Tuesday) |
| **Listing** Describe listing tasks.  No specific command to list tasks because tasks are persistent in task pane.  As shown: |

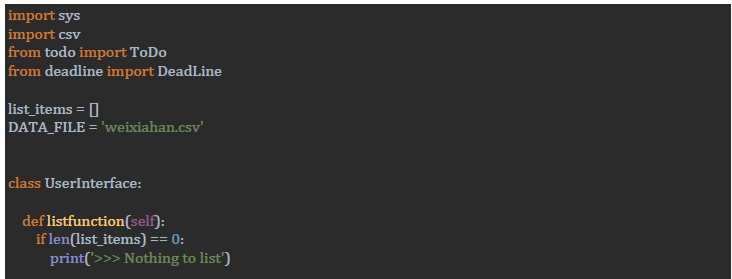
|  |
| --- |
| **Marking** Describe marking/unmarking tasks as done.  Marking task 3 as done:   * ‘done’ + TASK NUMBER     Marking task 3 as undone:   * ‘pending’ + TASK NUMBER     What if user tries to mark a done task as done again or a pending task as pending?   * Mark pending task as pending      * Mark ‘done’ task as done. |

|  |
| --- |
| **Help** Describe how Monty provides a helpful instructions to users e.g., the result of a ‘help’ command  Helpful instruction panel is launched if user types ‘help’ explicitly. |
| **Saving** Give a sample of the tasks as they are stored in the hard disk.  Stored on harddisk (CSV):  T,read book,True  D,return book,False,Tuesday  T,watch movie,False |
| **ErrorHandling** Give different types of incorrect commands the app can handle and the corresponding error message given by the app.  Todo without a specific task (e.g. blank).  Todo without a specific task (e.g. blank).  Deadline without task, deadline, or both.  Trying to mark non-existent tasks as pending or done.  Trying to delete, mark as done, or mark as pending non numerical input.  Trying to mass execute action using non-numerical inputs, or non-existent tasks.  All other non-permitted commands. |
| **Deleting** Describe deleting tasks. |
| **GUI**/**individual feature** If you implemented a GUI, give some screenshots. If you implemented an individual feature, describe that feature. |
| **Other features** Describe other features you implement (i.e., not described above), if any e.g., optional increments. |
| **OOP/UML** Give a class diagram to match your code. Include examples of (if applicable) in the diagram: classes, some attributes/methods, associations, inheritance, navigability, association labels and roles, multiplicity, class-level members  Give at least one object diagram illustrating the state of your program when the user has added at least 2 tasks. |
| **UnitTests**: Give the code of 2-3 unit tests (if any) from your code. |
| **Suggested test commands** Give a list of commands a tester can execute in sequence to examine your product. Cover all features in a reasonable order. E.g.,  list  todo borrow book  deadline return book /by Sunday  done 2 |

# Appendix: Code

Give all your code here. Indicate filename too. Some examples given below.

**main.py**



**todo.py**

