Name: Au Chi Chung **Student Number:** 12017765
Course: COMP S363F – Distributed Systems and Parallel Computing
Exam Assignment **Date of submission:** 22 May 2020 (Fri) 23:59

Description of project changes

1. Functionality

First, I have been changed the entries function from original code completely because it is not enough for the entries with different requirements. **[Original Code: server.ex (line 12 - 14, line 33 - 40), list.ex (line 23 - 27)]**

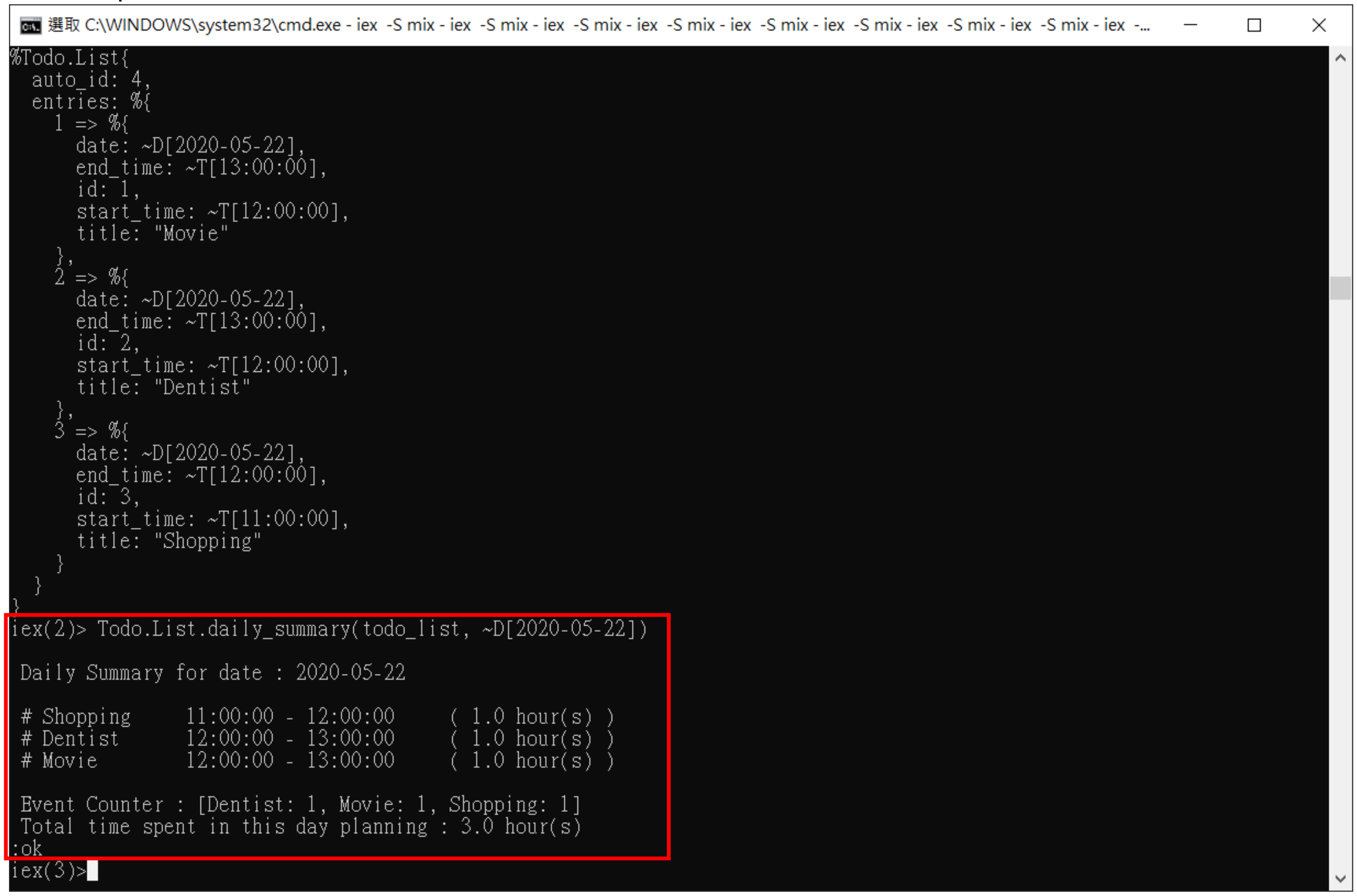
Create a nice report summary (daily report & to-do list report)

It shows the daily report summary with the ascending order for time and title from a date request whether the to-do list is inserted orderless. **[New Code: server.ex (line 30 – 32, line 110 - 117), list.ex (line 69 - 84)]**

For example, the data are following that which is orderless in the same date:

```
todo_list =
Todo.List.new([
        %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Movie"},
        %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Dentist"},
        %{date: ~D[2020-05-22], start_time: ~T[11:00:00], end_time: ~T[12:00:00], title: "Shopping"}
])
```

Result Capture Screen:



```
選取 C:\WINDOWS\system32\cmd.exe - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -...
%Todo.List{
  auto_id: 4,
  entries: %{
    1 => %{
      date: ~D[2020-05-22],
      end_time: ~T[13:00:00],
      id: 1,
      start_time: ~T[12:00:00],
      title: "Movie"
    },
    2 => %{
      date: ~D[2020-05-22],
      end_time: ~T[13:00:00],
      id: 2,
      start_time: ~T[12:00:00],
      title: "Dentist"
    },
    3 => %{
      date: ~D[2020-05-22],
      end_time: ~T[12:00:00],
      id: 3,
      start_time: ~T[11:00:00],
      title: "Shopping"
    }
  }
}
iex(2)> Todo.List.daily_summary(todo_list, ~D[2020-05-22])

 Daily Summary for date : 2020-05-22

 # Shopping     11:00:00 - 12:00:00     ( 1.0 hour(s) )
 # Dentist      12:00:00 - 13:00:00     ( 1.0 hour(s) )
 # Movie        12:00:00 - 13:00:00     ( 1.0 hour(s) )

 Event Counter : [Dentist: 1, Movie: 1, Shopping: 1]
 Total time spent in this day planning : 3.0 hour(s)
:ok
iex(3)>
```

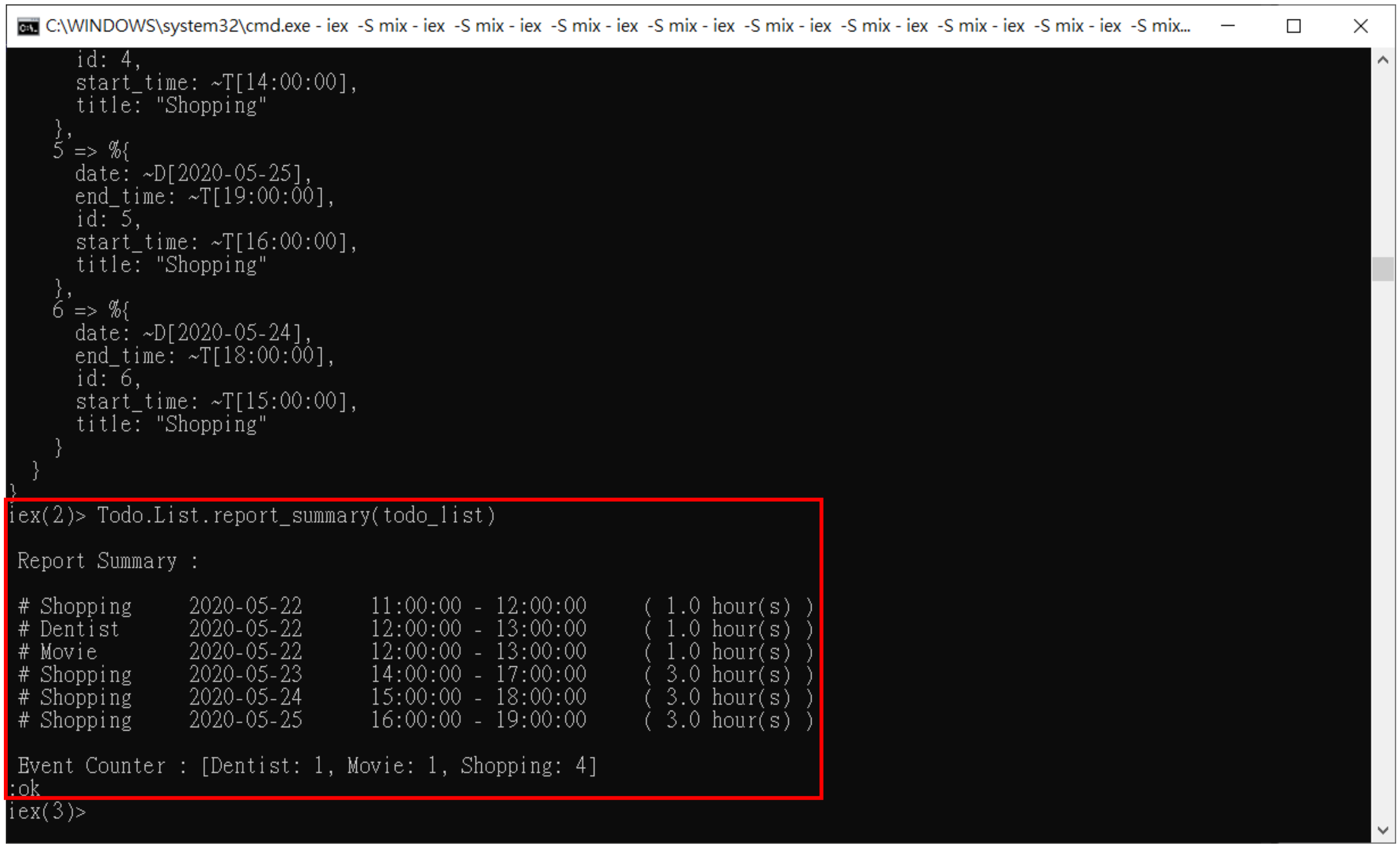
That it can display the information with ascending order, time spent for each item, and the total time spent for the planning in that date.

Also, I have made the function that print all event of the to-do list. **[New Code: server.ex (line 26 – 28, line 101 - 108), list.ex (line 52 - 67)]**

For example, the data are following that which is orderless in different date:

```
todo_list =
Todo.List.new([
        %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Movie"},
        %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Dentist"},
        %{date: ~D[2020-05-22], start_time: ~T[11:00:00], end_time: ~T[12:00:00], title: "Shopping"},
        %{date: ~D[2020-05-23], start_time: ~T[14:00:00], end_time: ~T[17:00:00], title: "Shopping"},
        %{date: ~D[2020-05-25], start_time: ~T[16:00:00], end_time: ~T[19:00:00], title: "Shopping"},
        %{date: ~D[2020-05-24], start_time: ~T[15:00:00], end_time: ~T[18:00:00], title: "Shopping"}
])
```

Result Capture Screen:



```
C:\WINDOWS\system32\cmd.exe - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix...
      id: 4,
      start_time: ~T[14:00:00],
      title: "Shopping"
    },
    5 => %{
      date: ~D[2020-05-25],
      end_time: ~T[19:00:00],
      id: 5,
      start_time: ~T[16:00:00],
      title: "Shopping"
    },
    6 => %{
      date: ~D[2020-05-24],
      end_time: ~T[18:00:00],
      id: 6,
      start_time: ~T[15:00:00],
      title: "Shopping"
    }
  }
}
iex(2)> Todo.List.report_summary(todo_list)

 Report Summary :

 # Shopping     2020-05-22      11:00:00 - 12:00:00     ( 1.0 hour(s) )
 # Dentist      2020-05-22      12:00:00 - 13:00:00     ( 1.0 hour(s) )
 # Movie        2020-05-22      12:00:00 - 13:00:00     ( 1.0 hour(s) )
 # Shopping     2020-05-23      14:00:00 - 17:00:00     ( 3.0 hour(s) )
 # Shopping     2020-05-24      15:00:00 - 18:00:00     ( 3.0 hour(s) )
 # Shopping     2020-05-25      16:00:00 - 19:00:00     ( 3.0 hour(s) )

 Event Counter : [Dentist: 1, Movie: 1, Shopping: 4]
:ok
iex(3)>
```

It can also perform the ascending order, time spent for each item, and the counter.

Name: Au Chi Chung **Student Number:** 12017765
Course: COMP S363F – Distributed Systems and Parallel Computing
Exam Assignment **Date of submission:** 22 May 2020 (Fri) 23:59

Add time in addition to date

The entries method changed to 5 ways for search (search all, search by title, search by date, search by time, and search by date and time). And those method can also perform the ascending order to display the result.
[New Code: server.ex (line 34 – 52, line 119 - 162), list.ex (line 86 - 124)]

Track time spent as well as planning

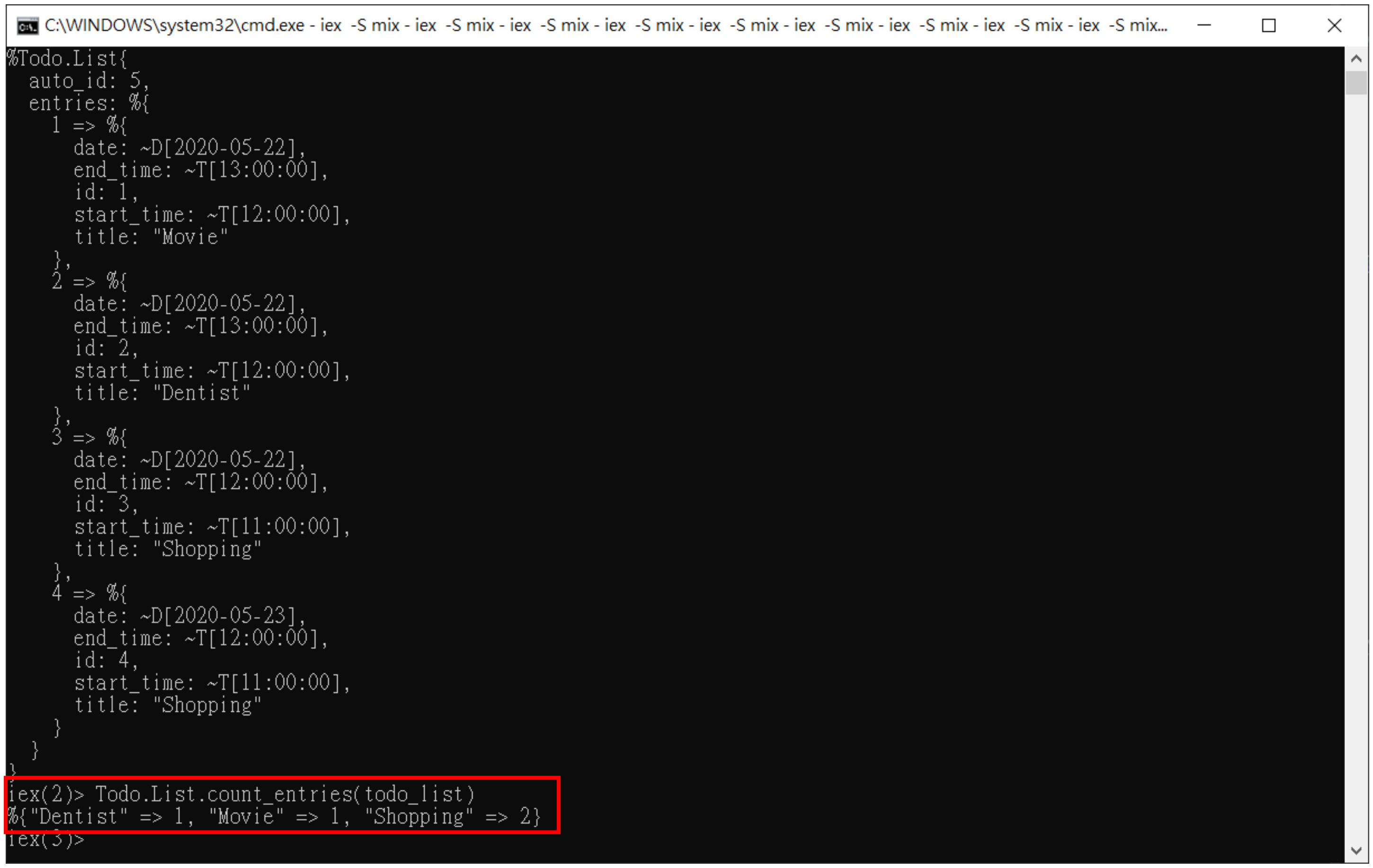
The time divided into start time and end time that it shows the processing time of each event planning. And it can track by date, track for title, and track by date and title.

[New Code: server.ex (line 54 – 64, line 164 - 189), list.ex (line 126 - 146)]

To-do list counter

Provide the counter for each event planning. And it can count for all in to-do list or count for date.

[New Code: server.ex (line 14 – 20, line 76 - 92), list.ex (line 18 - 27)]

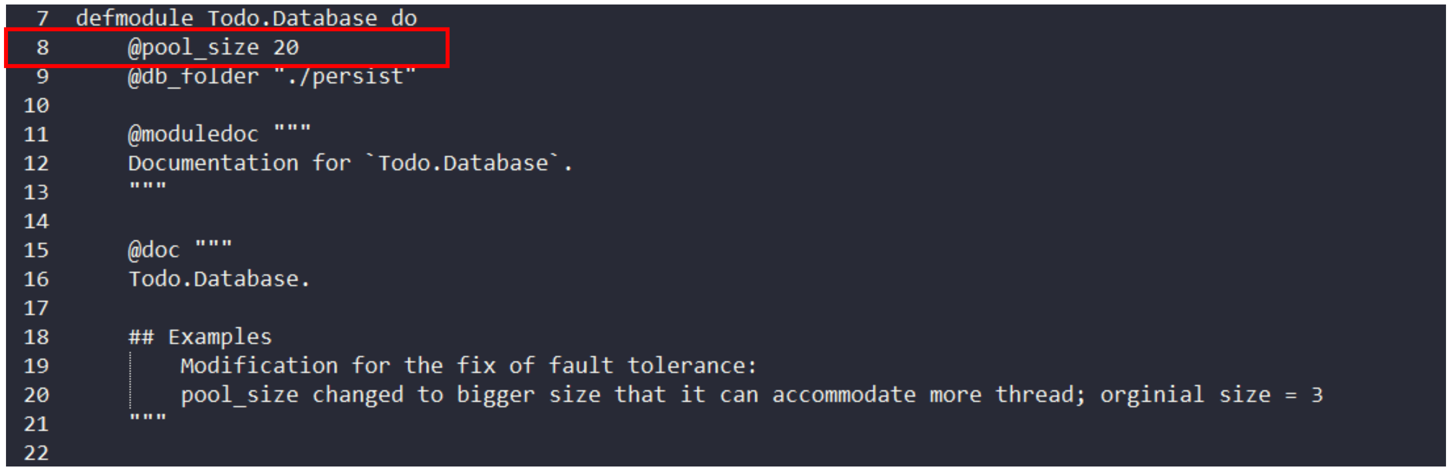


```
C:\WINDOWS\system32\cmd.exe - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix - iex -S mix...
%Todo.List{
  auto_id: 5,
  entries: %{
    1 => %{
      date: ~D[2020-05-22],
      end_time: ~T[13:00:00],
      id: 1,
      start_time: ~T[12:00:00],
      title: "Movie"
    },
    2 => %{
      date: ~D[2020-05-22],
      end_time: ~T[13:00:00],
      id: 2,
      start_time: ~T[12:00:00],
      title: "Dentist"
    },
    3 => %{
      date: ~D[2020-05-22],
      end_time: ~T[12:00:00],
      id: 3,
      start_time: ~T[11:00:00],
      title: "Shopping"
    },
    4 => %{
      date: ~D[2020-05-23],
      end_time: ~T[12:00:00],
      id: 4,
      start_time: ~T[11:00:00],
      title: "Shopping"
    }
  }
}
iex(2)> Todo.List.count_entries(todo_list)
%{"Dentist" => 1, "Movie" => 1, "Shopping" => 2}
iex(3)>
```

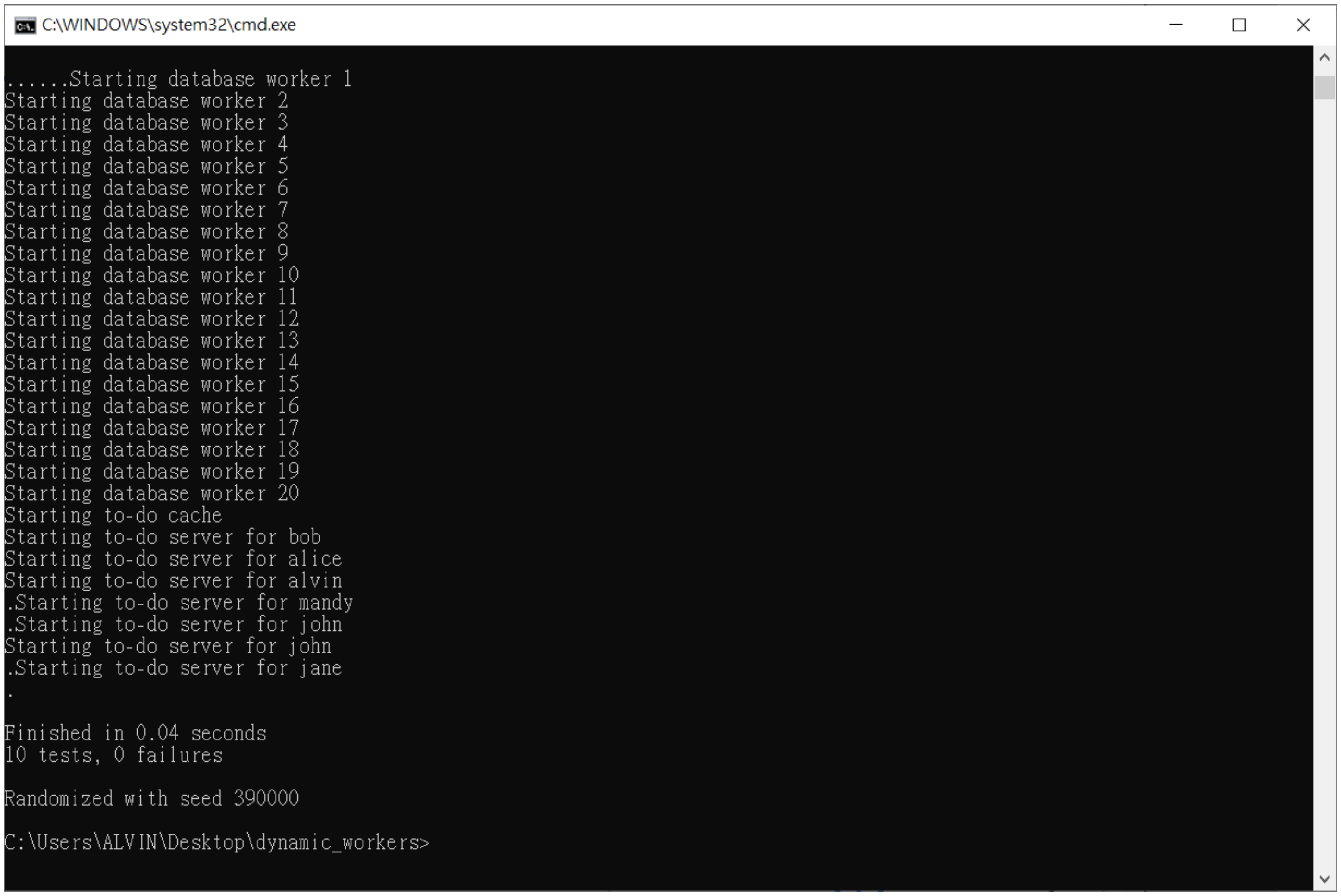
Name: Au Chi Chung **Student Number:** 12017765
Course: COMP S363F – Distributed Systems and Parallel Computing
Exam Assignment **Date of submission:** 22 May 2020 (Fri) 23:59

2. Efficiency or Capacity

To holding larger amount of workload or data for the capacity part, the original pool size is not enough for handle more than 3 workload even doing the auto test or using the function, that I changed a little bit larger for handle more of threads. **[Modified Code: database.ex (line 8)]**



```
 7  defmodule Todo.Database do
 8      @pool_size 20
 9      @db_folder "./persist"
10
11      @moduledoc """
12      Documentation for `Todo.Database`.
13      """
14
15      @doc """
16      Todo.Database.
17
18      ## Examples
19          Modification for the fix of fault tolerance:
20          pool_size changed to bigger size that it can accommodate more thread; orginial size = 3
21      """
22
```



C:\WINDOWS\system32\cmd.exe

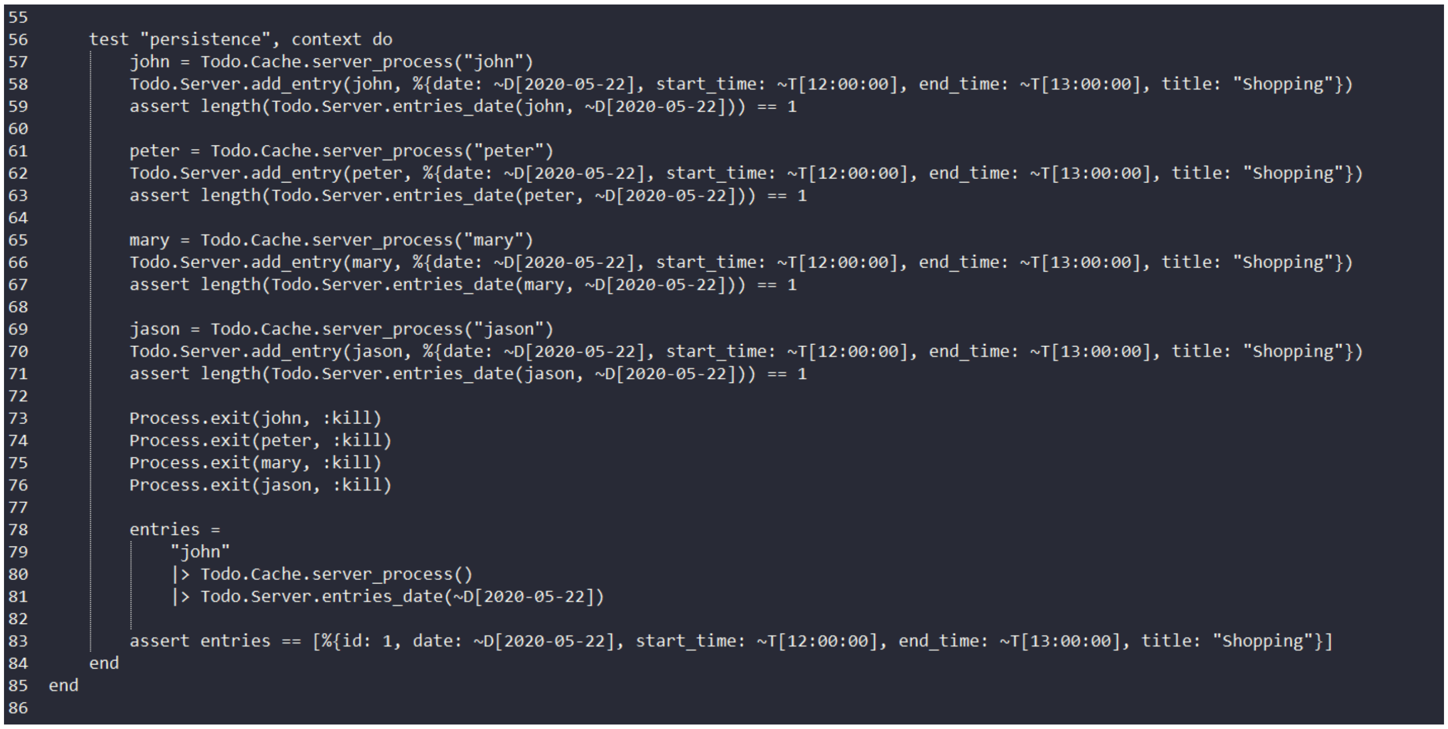
```
......Starting database worker 1
Starting database worker 2
Starting database worker 3
Starting database worker 4
Starting database worker 5
Starting database worker 6
Starting database worker 7
Starting database worker 8
Starting database worker 9
Starting database worker 10
Starting database worker 11
Starting database worker 12
Starting database worker 13
Starting database worker 14
Starting database worker 15
Starting database worker 16
Starting database worker 17
Starting database worker 18
Starting database worker 19
Starting database worker 20
Starting to-do cache
Starting to-do server for bob
Starting to-do server for alice
Starting to-do server for alvin
.Starting to-do server for mandy
.Starting to-do server for john
Starting to-do server for john
.Starting to-do server for jane
.

Finished in 0.04 seconds
10 tests, 0 failures

Randomized with seed 390000

C:\Users\ALVIN\Desktop\dynamic_workers>
```

Name: Au Chi Chung **Student Number:** 12017765
Course: COMP S363F – Distributed Systems and Parallel Computing
Exam Assignment **Date of submission:** 22 May 2020 (Fri) 23:59

3. Fault Tolerance

In this area, the existing fault tolerance is that the failures when keep doing the killing process within a short time that it due to not enough workload. And the issue has been relaxed after I changed the pool size in the database. And there is the test rewrite of “persistence” from the original code for testing the fault tolerance mentioned in the exam assignment question (A limit of 3 failures in 5 minutes), it shows the fault tolerance of the workload by kill many process at the same time.

[Modified Code: todo_cache_test.exs (line 56 - 84)]



```
55
56      test "persistence", context do
57          john = Todo.Cache.server_process("john")
58          Todo.Server.add_entry(john, %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Shopping"})
59          assert length(Todo.Server.entries_date(john, ~D[2020-05-22])) == 1
60
61          peter = Todo.Cache.server_process("peter")
62          Todo.Server.add_entry(peter, %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Shopping"})
63          assert length(Todo.Server.entries_date(peter, ~D[2020-05-22])) == 1
64
65          mary = Todo.Cache.server_process("mary")
66          Todo.Server.add_entry(mary, %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Shopping"})
67          assert length(Todo.Server.entries_date(mary, ~D[2020-05-22])) == 1
68
69          jason = Todo.Cache.server_process("jason")
70          Todo.Server.add_entry(jason, %{date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Shopping"})
71          assert length(Todo.Server.entries_date(jason, ~D[2020-05-22])) == 1
72
73          Process.exit(john, :kill)
74          Process.exit(peter, :kill)
75          Process.exit(mary, :kill)
76          Process.exit(jason, :kill)
77
78          entries =
79              "john"
80              |> Todo.Cache.server_process()
81              |> Todo.Server.entries_date(~D[2020-05-22])
82
83          assert entries == [%{id: 1, date: ~D[2020-05-22], start_time: ~T[12:00:00], end_time: ~T[13:00:00], title: "Shopping"}]
84      end
85  end
86
```

4. Automated Testing

First, I have setup the time and changed the assert function from “=” to “==” with all of the original test case that ensure it can work well for the time data and make sure the assertion is completely correct.

Then, I added 3 test case in the list test including count entries, entries, and track time. Those performed different checking for any situation.

count_entries [New Code: todo_list_test.exs (line 54 - 68)]

entries [New Code: todo_list_test.exs (line 70 - 101)]

track_time [New Code: todo_list.exs (line 103 - 119)]

After that, I also have setup the test for all the entries method by testing the server process of the users

entries [New Code: todo_cache_test.exs (line 31 - 54)]