CO324 Lab 5: Stateful systems

# **Objectives**

At the end of the lab you should be able to;

# **Instructions**

This lab must be completed using the same code provided for lab 4 part (B)

* Use only grpcio and built-in modules in Python 3 to complete these exercises.
* Solutions to discussion questions should be in a single plain text file named answers.txt. State your E-number at the top of answers.txt.
* Zip all files (including the ones provided) and submit them. No other archive formats please!

Submissions that do not follow instructions will incur a 10% penalty.

# **Exercises**

1. Write two versions of Taskimpl.editTask that implement the different concurrency semantics discussed in class.
2. nondestructive\_editTask: keeps a history of edits made to the task.
3. destructive\_editTask: deletes the task and creates a new task with the edits.
4. Suppose we modified the definition of Task to be able to track the current state of the task (similar to the Github issue tracker.)

|  |
| --- |
| service Taskapi {  // Add a new task and return its id  rpc addTask (google.protobuf.StringValue) returns (Task);  // Delete a task by id  rpc delTask(google.protobuf.UInt64Value) returns (Task);  // Edit an existing task (ignoring write conflicts)  rpc editTask(Task) returns (Task);  // List all tasks in the given states  rpc listTasks (TaskQuery) returns (Tasks);  }  enum TaskState {  OPEN = 0; // may transition to ASSIGNED or CANCELLED.  ASSIGNED = 1; // may transition to PROGRESSING only.  PROGRESSING = 2; // may transition to DONE or CANCELLED.  DONE = 3; // terminal state.  CANCELLED = 4; // terminal state.  }  message Task {  uint64 id = 1;  string description = 2;  TaskState state;  }  message Tasks {  repeated Task pending = 1;  }  // A set of task states to select when listing tasks  message TaskQuery {  repeated TaskState selected;  } |

We would like to add the following rules governing the task lifecycle.

* New tasks are initially in OPEN state.
* OPEN tasks may be ASSIGNED or CANCELLED.
* Once someone begins work on an ASSIGNED task they set the status to PROGRESSING.
* A task that is PROGRESSING is either set to DONE once completed or CANCELLED.

All other state changes should return a suitable error.

1. Draw a state machine to illustrate all possible legal transitions between states. Save your answer as taskstate-fsm.pdf
2. Implement task state update logic using your implementation of editTask in Part(B) question 5. Your implementation must return a suitable error on illegal state transitions.
3. Can a task ever make an illegal transition due to a race condition during update? Explain why (not.)

Modify your listTasks implementation to return a list of tasks that are in the specified states. If TaskQuery.selected is empty, return a list of all tasks. For example listTasks([DONE, CANCELLED]) should return tasks in either the DONE or CANCELLED state. Hint: have a look at [Python comprehensions.](https://www.python101.pythonlibrary.org/chapter6_comprehensions.html)