

## A.3 TEST DESIGN SPECIFICATIONS

### 3.1 Feature(s) to be tested, corresponding test cases (T.C) and test logs (T.L)

#### Individual features

- 2.1.0 Search engine
  - T.C 1, 2, 3, 4, 5, 6
  - T.L 1, 2, 3, 4, 5, 6
- 2.1.1 NOML file uploader
  - T.C 7
  - T.L 7
- 2.1.2 Molecule editor
  - T.C 8, 9
  - T.L 8, 9
- 2.1.3 Molecule validator
  - T.C 10
  - T.L 10
- 2.1.4 Chat room—
  - T.C 11
  - T.L 11
- 2.1.5 File sharing and discussion board
  - T.C 12
  - T.L 12
- 2.1.6 Sending email agent
  - T.C 13, 14
  - T.L 13, 14
- 2.1.7 Running time prediction agent
  - T.C 15
  - T.L 15
- 2.1.8 Similar simulation finder
  - T.C 16
  - T.L 16
- 2.1.9 Automatic restarting agent
  - T.C 17
  - T.L 17
- 2.1.10 Manual restarting agent
- 2.1.11 Runtime reports
- 2.1.12 Completed reports
- 2.1.13 Load balancing
  - T.C 18
  - T.L 18
- 2.1.14 Core simulator

### 3.2 Approach refinement

All the individual features and intelligent components will be tested first with valid and invalid output. The combinations of features and components will then be used. The individual and combination of features and components will be tested for fault recovery, utility, reliability, robustness, and error response.

By using *fault recovery* testing we will test the reaction of the application servers when there is a loss of power or an error. An application server shut down means that the connection with databases is lost. We will test the databases capability to rollback and recover data after an error or an exception occurs.

On the *utility* phase we will be focusing on how easy and user friendly are the features individually and combined and weather they perform correctly when subjected to inputs which are valid in terms of the specifications.

The *reliability* approach will measure the frequency and criticality of a system failure. It will consist of testing to know how often the system will fail because of data load, not enough database space, hardware constrains etc. After this test has been done we will observe and document the effects of the failure and time it took to repair.

The *robustness* phase will consist of sending a large volume of data trough the system and see what breaks and how much data and stress can take before it breaks. Error response, this phase consists of testing and observing what is the response of the system and if it displays some sort of message to inform the user of a problem after a system failure or invalid inputs.

We will also be looking at the input constrains where applicable on the features (for example, when creating a new environment set, you can only use real numbers and numbers between 0 and 1, other type of number should not be allowed). Each test case must be tested at lease once by violating the input constrains and observe the response.