Amit Peled, Daniel Piro, Shahar Alon, Nofar Rozenberg

BGU / MolOpt

MolOptimizer - Final Project in Software Engineering

Testing Document

Contents

[Testing Functional Requirements 2](#_Toc136092834)

[Testing Non-Functional Requirements 6](#_Toc136092835)

[Random & Automatically-Generated Tests 9](#_Toc136092836)

[Testing The User Interface 11](#_Toc136092837)

[Testing Build, Integration & Deployment 15](#_Toc136092838)

Chapter 1

# Testing Functional Requirements

|  |
| --- |
| Requirement #1: Guest must be able to register to the system. |
| Test:   * Good scenario   Description: The test will be given valid credentials and will call the register function with them.  Expected Result: The new user will be registered successfully.   * Bad scenario   Description: The test will be given invalid credentials OR credentials of an existing user and will call the register function with them.  Expected Result: The system will present an appropriate error. |

|  |
| --- |
| Requirement #2: Guest must be able to login the system if he is registered. |
| Test:   * Good scenario   Description: The test will be given valid credentials and will call the login function with them.  Expected Result: The new user will be logged in successfully.   * Bad scenario   Description: The test will be given invalid credentials OR credentials of a  non-existing user and will call the login function with them.  Expected Result: The system will present an appropriate error. |

|  |
| --- |
| Requirement #3: Guest must not be able to run the Mol-Optimizer. |
| Test:   * Good scenario   Description: The test will try to run an algorithm without logging in before the run execution.  Expected Result: The system will present an appropriate error.   * Note: This scenario will be tested on each algorithm in the system. |

|  |
| --- |
| Requirement #4: User must be able to upload dataset and run the Mol-Optimizer on it. |
| Test:   * Good scenario   Description: The test will be given a user which is registered and logged in and a valid files/dataset for running an algorithm and will execute the algorithm.  Expected Result: The user will run the algorithm successfully.   * Bad scenario   Description: The test will be given a user which is not registered OR not logged in OR an invalid files/dataset for running an algorithm and will execute the algorithm.  Expected Result: The system will present an appropriate error.   * Note: The bad scenario will be tested on each algorithm in the system. |

|  |
| --- |
| Requirement #5: User who is not Admin should not be able to see history of other users |
| Test:   * Good scenario   Description: The test will be given an admin which is registered and logged in and will call the “view history” function.  Expected Result: The admin will be able to see all the runs history.   * Bad scenario   Description: The test will be given a user which is not registered OR not logged in OR not an admin and will call the “view history” function  Expected Result: The system will present an appropriate error. |

|  |
| --- |
| Requirement #6: The system should record details of the runs of the algorithms |
| Test:   * Good scenario   Description: The test will be running an algorithm and will save the results.  Expected Result: The results will be available and matching to the algorithm that ran.   * Note: This scenario will be tested on each algorithm in the system. |

|  |
| --- |
| Requirement #7: User should be able to see history of the Mol-Optimizer runs he did. |
| Test: This has already been tested in Requirement #5 tests. |

|  |
| --- |
| Requirement #8: User should be able to see how many users are using the system at this current time. |
| * Good scenario   Description: The test will be given a user which is registered and logged in and will call the “see-amount-of-users” function.  Expected Result: The user will see the number of users in the system.   * Bad scenario   Description: The test will be given a user which is not registered OR not logged in and will call the “see-amount-of-users” function.  Expected Result: The system will present an appropriate error. |

|  |
| --- |
| Requirement #9: Admin should be able to see all users history details |
| Test: This has already been tested in Requirement #5 tests. |

|  |
| --- |
| Requirement #10: Admin should be able to remove Admin appointment of User |
| Test:   * Good scenario   Description: The test will be given an admin which is registered and logged in and will call the “remove admin” function with an existing admin in the system.  Expected Result: The wanted admin will be removed successfully.   * Bad scenario   Description: The test will be given a user which is not registered OR not logged in OR not an admin and will call the “remove admin” function with an existing admin in the system OR with a non-admin details.  Expected Result: The system will present an appropriate error. |

|  |
| --- |
| Requirement #11: Admin should be able to remove User from the system |
| Test:   * Good scenario   Description: The test will be given an admin which is registered and logged in and will call the “remove user” function with an existing user in the system.  Expected Result: The wanted user will be removed successfully.   * Bad scenario   Description: The test will be given a user which is not registered OR not logged in OR not an admin and will call the “remove admin” function with a non-existing user in the system.  Expected Result: The system will present an appropriate error. |

|  |
| --- |
| Requirement #12: The Mol-Optimizer should be able to run on every valid dataset. |
| Test: This has already been tested in Requirement #4 tests. |

|  |
| --- |
| Requirement #13: All tasks execution successfully |
| Test: This has already been tested in Requirement #6 tests. |

|  |
| --- |
| Requirement #14: All tasks complete within 10-15 min successfully |
| Test:  Each algorithm will be run, and time stamp will be taken before and after its execution.  Success will be achieved only if the longest algorithm will take less than 15 minutes (Otherwise – the test will fail). |

|  |
| --- |
| Requirement #15: System will support 1000 request per seconds |
| Test:  Stress tests will be created, and few random methods will be activated at the same time.  Success will be achieved only if all the requests will be responded as expected  (Otherwise – the test will fail). |

|  |
| --- |
| Requirement #16: On network/connection failure, system will notify user and rollback changes |
| Test:  First, a random function in the system will be activated.  Then, an intentional network failure will be done.  The test will check both the website recovery and that no change has occurred in the system.  If that will happen, Success will be achieved (Otherwise – the test will fail). |

|  |
| --- |
| Requirement #17: On connection issues, system will do auto retry |
| Test: This has already been tested in Requirement #16 tests. |

|  |
| --- |
| Requirement #18: System will not save any personal data nor expose personal data of users |
| Test:  The test will go on any credentials in the system and will check that all the sensitive data has been encrypted.  If all this data is encrypted Success will be achieved (Otherwise – the test will fail). |

**Note:**

Most of these tests are written as part of the backend code.

You can simply run them by:

* activating the virtual environment of the backend.
* run the command: “python3 manage.py test”

That’s it – the tests will run automatically!

Chapter 2

# Testing Non-Functional Requirements

**Performance (Speed, Capacity, Throughput, etc.).**

Task execution:

Acceptable: All tasks execution successfully.

Unacceptable: 90% of tasks execute successfully, 10% need to retry.

Worse: None of the task execute successfully.

|  |
| --- |
| Performance |
| Test:  Description: The test will run a lot of algorithm runs (of all kinds) and will check if each run got a result (ran successfully).  Expected Result: All the runs finished successfully  (Less than that we’ll be unacceptable). |

Task completion speed:

Acceptable: All tasks complete within 10-15 min successfully.

Unacceptable: All tasks complete within 20 min successfully.

Worse: All tasks complete within more than 20 min successfully.

|  |
| --- |
| Task completion speed |
| Test:  Description: The test will run a lot of algorithm runs (of all kinds) and will check if there’s a run that ran more than 15 min.  Expected Result: All tasks ran less than 15 min each. |

Users’ capacity:

Acceptable: Website support 1000 request per seconds.

Unacceptable: Website support 1000 request per 3 seconds or more.

Worse: Website won’t support 1000 request.

|  |
| --- |
| Users’ capacity |
| Test:  Description: The test will send 1000 API requests at the same time (different kind of API requests, all requests are within one second).  Expected Result: All requests were answered (no one of them were ignored/failed). |

**Reliability & Stability.**

Network failure:

Acceptable: Notify user and rollback changes.

Unacceptable: Rollback changes without notify user.

Worse: Allow partial changes to update, corrupt data.

|  |
| --- |
| Network failure |
| Test:  Description: The test will run a very few API requests while turning the network connection on/off.  Expected Result: Any change made while the network connection was off, wasn’t saved and the status of the system was the same as it was before the disconnection. |

Data recovery:

Acceptable: System will do auto retry on connection issues.

Unacceptable: System will notify user on failure only.

Worse: System abord execution without any notification.

|  |
| --- |
| Data recovery |
| Test:  Description: The test will run a very few API requests which affect the data stored in the system while turning the database connection on/off.  Expected Result: Any change made while the database connection was off, wasn’t saved and the data saved in the system is the same as it was before the disconnection. |

**Safety & Security.**

Privacy & Encryption:

Acceptable: System will not save any personal data nor expose personal data of users.

Unacceptable: System will save personal data for authentication.

Worse: System will save personal data for no reason.

|  |
| --- |
| Privacy |
| Test:  Description: The test will go over any sensitive data in the system (for example – passwords) and will check that it’s written in any kind of encryption.  Expected Result: All sensitive data is stored in the system using encryption. |

Permission:

Acceptable: System will abstract any data based on permission.

Unacceptable: System will show all data but disable access to them based on permission.

Worse: System will allow all users access to all website content.

|  |
| --- |
| Permission |
| Test:  Description: The test will use 3 different type of users – Guest, User & Admin.  The test will check that each type, has its matching permissions and accesses.  Expected Result: Each user will be able to do only what he’s permitted to do in the system. |

**Usability & Availability.**

Acceptable: Website will work on all known browser types.

Unacceptable: Website will work only on chrome and Firefox.

Worse: Website will work only on chrome.

|  |
| --- |
| Usability & Availability |
| Test:  Description: This test will be done manually!  This test will run few scenarios in each browser – Chrome, Firefox, Explorer.  Expected Result: All scenarios ran flawlessly in all browsers. |

Chapter 3

# Random & Automatically-Generated Tests

Most of MolOpt Testing is strict and unambiguous – except for 2 procedures that happens in the system:

Auto-process of the algorithms runs, and user’s password reset (in the “Forgot Password” section in the Login procedure in the system).  
  
Except these 2 procedures mentioned above, all actions in the system are   
straight-forward – which means that for a specific input/action of the user, there’s only one expected result.  
Therefore, any action in the system (except these 2 procedures mentioned above)   
is tested using unit tests / functional requirements tests / non-functional requirements tests / UI tests.

For testing the Auto-process of the algorithms runs and User’s password reset, these are the tests:

|  |
| --- |
| Auto-Process Algorithm Run |
| Random/Automated Procedure:  The test will generate in its set-up a random auto setting for an auto mode run process - it will pick the Alignment algorithm, at least 1 of the Feature Extraction algorithms, at least 1 of the Machine Learning algorithms.  Test:   * Good scenario   Description: The test will check that the Auto mode run, will run successfully when the initial input is valid.  Expected Result: The Auto process will run successfully.   * Bad scenario   Description: The test will check that the Auto mode run, won’t run when the initial input is invalid.  Expected Result: The user will be notified by a message that the data that he inserted is not valid. |

|  |
| --- |
| User’s Password Reset / “Forgot Password” Action |
| Random/Automated Procedure:  The test will generate in each iteration a new password for the user and will use that password only for the validation of the test.  Test:   * Single scenario   Description: The test will check (for a huge number of iterations) that resetting the password and using it as the new user’s password successfully.  Expected Result: The password reset will work flawlessly in all the test’s iterations. |

Chapter 4

# Testing The User Interface

MolOpt UI testing is following a list of all the API calls that the Frontend side is capable of doing during any type of usage in the MolOpt system, which means that every API call have to be tested extra carefully and with attention to details.  
In addition, some results of actions used in the system including results outside of the system (for example – sending emails to the user’s email).

Rather than that, some of these results (for example – the results an algorithm run) is very context-sensitive and most be double-checked.

Therefore, MolOpt UI testing is must include both manual and automated tests!

**Here Below is a list of all system APIs and their tests:**

Manual Tests:

|  |
| --- |
| API #1: Register |
| Test:   * Good scenario   Description: The test will check that for a valid user credentials given by the user, the user will be registered successfully.  Expected Result: The user will be registered successfully.   * Bad scenario   Description: The test will check that for any invalid user credentials given by the user, the user won’t be able to register to the system.  Expected Result: The user will be notified by a message, about the invalid credentials that he has inserted. |

|  |
| --- |
| API #2: Login |
| Test:   * Good scenario   Description: The test will check that for a valid user credentials given by the user, the user will be logged-in successfully.  Expected Result: The user will be logged-in successfully.   * Bad scenario   Description: The test will check that for any invalid user username (email) or unmatched password given by the user, the user won’t be able to log in to the system.  Expected Result: The user will be notified by a message that his email/password are not valid. |

|  |
| --- |
| API #3: Logout |
| Test:   * Single scenario   Description: The test will check that logged in user, can logout from the system (in the UI, this is the only scenario that can be tested).  Expected Result: The user will logout from the system successfully. |

|  |
| --- |
| API #4: Update User Details |
| Test:   * Good scenario   Description: The test will check for each user detail in the system (email, password, first name, last name, etc.) that updating any user’s details can be done successfully, when the new details are valid.  Expected Result: The user’s details will be updated successfully.   * Bad scenario   Description: The test will check for each user detail in the system (email, password, first name, last name, etc.) that updating any user’s details can’t be done, when the new details are invalid.  Expected Result: The user will be notified by a message, about the invalid credentials that he has inserted. |

|  |
| --- |
| API #5: View All Runs Of A User In The System |
| Test:   * Single scenario   Description: The test will check that each user can see its own runs (all of them) in the system.  Expected Result: The user will be able to see all his runs in the Tasks/Runs page in the system. |

|  |
| --- |
| API #6: Remove A User From The System (Admin Action) |
| Test:   * Good scenario   Description: The test will check that an admin in the system, can remove/delete any user in the system (when this user isn’t an admin).  Expected Result: The user will be removed successfully by the admin.   * Bad scenario #1   Description: The test will check that an admin in the system, can’t remove/delete another admin user in the system.  Expected Result: The user will be notified by a message, that the deletion isn’t possible due to user’s permission level.   * Bad scenario #2   Description: The test will check that an admin in the system, can’t remove/delete if the user’s details isn’t valid.  Expected Result: The user will be notified by a message, that the user that he wants to remove isn’t exist in the system. |

|  |
| --- |
| API #7: Promote User To Admin (Admin Action) |
| Test:   * Good scenario   Description: The test will check that an admin can promote user to be an admin in the system.  Expected Result: The user will be promoted successfully.   * Bad scenario   Description: The test will check that an admin user can’t be promoted (because he’s already an admin)  Expected Result: The user will be notified by a message that this user is already an admin. |

|  |
| --- |
| API #8: Downgrade Admin User To Regular User (Admin Action) |
| Test:   * Good scenario   Description: The test will check that an admin can downgrade an admin in the system, when the downgraded admin didn’t was the one who promoted the downgrading admin and when both are admin in the system.  Expected Result: The user will be downgraded back to regular user in the system successfully.   * Bad scenario #1   Description: The test will check that an admin can’t downgrade an admin in the system, if the downgraded admin was the one who promoted the downgrading admin.  Expected Result: The user will be notified by a message that this action can’t be done.   * Bad scenario #2   Description: The test will check that an admin can downgrade an admin in the system, when one of them is not an admin in the system.  Expected Result: The user will be notified by a message that this action can’t be done. |

|  |
| --- |
| API #9: View All Runs In The System (Admin Action) |
| Test:   * Single scenario   Description: The test will check that an admin can view all the runs’ status in the system.  Expected Result: The admin will be able to see al runs in the Tasks/Runs page in the system. |

|  |
| --- |
| API #10: Statistics Validation (Admin’s Page) |
| Test:   * Single scenario   Description: The test will check that the statistics page of the admin, have all the updated data of all the runs, users, and usage in the system.  Expected Result: The admin will see all the updated data. |

|  |
| --- |
| API #11: Running Algorithm In Manual Mode  (Alignment / Feature Extraction / Any Machine Learning Algorithm) |
| Test:   * Good scenario   Description: The test will check that each user can run any kind of algorithm exists in the system when each input data inserted is valid.  Expected Result: The user will be able to run each algorithm selected successfully.   * Bad scenario   Description: The test will check that each user can’t run any kind of algorithm exists in the system when an input data inserted is invalid.  Expected Result: The user will be notified by a message that the invalid data that he has inserted. |

Automated Tests:

The UI tests are running in the system using “Cypress” package.

“Cypress” allows us to check the test like a real user is using the system.

These tests include most of the tests mentioned above.

In order to run them, all you need to do is:

* Run the backend.

(for more information, read the README.md file in the Server folder of this project).

* Run in the Client side: “npm run tests”
* Then, choose one of the browsers you want to run the tests on.
* Go into the index.cy.js file in the “Cypress” software.

That’s it – the tests will run automatically!

Chapter 5

# Testing Build, Integration & Deployment

In MolOpt web-application, there are few measurements for validating that the Build, Integration & Deployment processes were run successfully.

**Build**

* The backend side is running using Django.

Therefore, installing few packages and using virtual environment is a must for it to run.

So, when the system will be uploaded by the user, Django’s validations will ensure that the system won’t be up unless all its requirements and installations were done successfully.

In addition, the same process will happen with any failure related to the application’s database (therefore, we can ensure that any issue we might face in the backend is covered and taken care before the application is loaded).

* The frontend side is running using React JS.

Therefore, installing few packages is a must for it to run.

So, when the system will be uploaded by the user, React’s validations will ensure that the system won’t be up unless all its requirements and installations were done successfully.

**Note:** Any unsuccessful installation of package in both backend and frontend will be notified to the user immediately by the system.

**Integration**

* MolOpt system has a notification system.

One of the main purposes of it, is to ensure that the system is functioning only when all the parts of the system are working well.

* MolOpt also have a logger.

Its main purpose is to follow and find new hazards and problems that might happen in the future.

That way, the system is always tracking its functioning, improving for future uses and avoiding system failures.

**Deployment**

MolOpt is running on the BGU servers.  
Therefore, any deployment validation that any BGU’s system in doing, is also relevant and related to the MolOpt web-application.