Needs cover page and table of contents

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Revision Date** | **Author** | **Summary of Changes** |
| 1 | 4-21-2013 | John Gibbons | Initial creation of document and first draft. |
| 2 | 4-24-2013 | John Gibbons | Second draft and added additional items. |
| 3 | 4-26-2013 | John Gibbons | Third draft and added additional items. |
| 4 |  | Roland Heintze | Final revision and modified Risk Assessment |

Introduction:

This document will layout how our development team will conduct all testing pertaining to the Chemistry Molecular Naming and Designing with Animation project. These tests will ensure Dr. Landge’s requirements are fully implemented and fully functional. The tests will begin with ensuring each method and function operate and return the desired results. Following those tests, our team will ensure the entire program as a whole functions properly and as intended on the systems promised. Finally, we will transition into the customer testing the software to ensure all functionality and designs meet or exceed expectations.

Risk Assessment: (add to and change risk lvls if nec)

1: Low Risk - 2: Medium Risk - 3: High Risk

* Algorithm to randomly create bonded molecule up to pentadecane chain: 3
* Algorithm to compare custom made molecule to standard: 3
* GUI interface to custom design algorithm: 2
* Queries from database: 2
* Check optionally entered name by user: 2
* Generate the animation: 2-3
* etc etc

Team Member’s Responsibilities:

* Roland Heintze - Team Leader and Primary Tester
  + Will oversee testing and conduct system, alpha, beta, and acceptance testing. Will ensure product is in release condition within schedule.
* John Gibbons - Quality Assurance Officer
  + Will ensure that testing is properly and fully conducted. All aspects will be run through and all results, especially issues, will be fully documented.
* Tim Elam - Tester
  + Will conduct unit, integration and functional testing to ensure product is fully operational. Will ensure all of the customers’ requirements are implemented and fully functional as well as the program will properly function on all promised operating systems.
* Chris Lansing - Testing Assistant
  + Will ensure all paperwork is properly filled out and all reports are generated. Will ensure all steps and results are properly and fully conducted and results are recorded.

Test Items:

Important Application Areas:

* Graphical User Interface
  + User friendly and in laymen terms.
  + Well designed and laid out.
* Molecule Design Window
  + Contains all necessary components.
  + Easy to understand functionality.
* Database Queries
  + Retrieves correct molecule names and structures.
* Notifications/Error Messages
  + Gives concise and direct error messages
  + Gives users useful information on how to resolve issues
* Proper Animations
  + Animation follows nomenclature rules.
  + Animation meets customers’ requirements.
* Proper Naming
  + All bonded molecules’ name are concise, correct, and are in correct format.

Features to be Tested:

**Feature**: Generating a Random Organic Molecule and Watching the Animation.

A. Input:

* 1. First button click.
  2. Second button click.

1. True Black Box Testing:
   1. Description
      1. Open program.
      2. Click ‘Generate Random Compound’ button.
      3. Click ‘View Animation’ button.
      4. Exit program.
   2. Expected Outcome
      1. Program will open.
      2. A random organic compound up to and including pentadecane is generated.
      3. The compound is passed to the animation generator and the animation plays according to the naming process.
      4. Program exits.
2. False Black Box Testing:
   1. Description
      1. Open program.
      2. Click “Generate Random Compound’ button.
   2. Expected Outcome
      1. Program will open.
      2. Error message displays stating an error has occured while trying to generate the organic compound.
3. True White Box Testing (Same as black as all inputs are automated button clicks)
4. False White Box Testing:
   1. Description
      1. Open program.
      2. Click “View Animation” button
   2. Expected Outcome
      1. Program opens.
      2. Error message is displayed stating that there is no current organic compound to base the animation from.

**Feature:** Generating a Random Organic Molecule, Entering in What is Believed to be the Name and Watching the Animation.

A. Input:

* 1. First button click.
  2. Name of organic molecule.
  3. Second button click.
  4. Third button click.

1. True Black Box Testing:
   1. Description
      1. Open Program.
      2. Click “Generate Random Compound’ button.
      3. Enter in Supposed name of organic compound.
      4. Click ‘Check Name’ button.
      5. Click ‘View Animation’ button.
      6. Exit program.
   2. Expected Outcome
      1. Program opens.
      2. Random organic molecule up to and including pentadecane is generated.
      3. Program informs user their guess is correct.
      4. Program displays correct animation for naming process.
      5. Program Exits.
2. False Black Box Testing:
   1. Description
      1. User clicks ‘Generate Random Compound’ button.
      2. User enters in guess for name of molecule.
      3. User clicks ‘Check Name’ button.
   2. Expected Outcome
      1. Dialog box will appear informing the user their guess for the name is incorrect.
3. True White Box Testing:
   1. Description
      1. User clicks ‘Generate Random Compound’ button.
         1. Makes sure compound generated is up to and including pentadecane and correct format compared to pre gathered data.
      2. User enters in guess for name of molecule.
      3. User clicks ‘Check Name’ button.
         1. Makes sure the guess is correct compared to pre gathered data.
      4. User clicks “View Animation” button.
         1. Makes sure animation follows correct paths according to scientific nomenclature.
      5. Exit program.
   2. Expected Outcome
      1. A correct random organic compound is generated.
      2. User is informed their guess is correct.
      3. Correct animation is displayed.
      4. Program exits.
4. False White Box Testing:
   1. Description
      1. User clicks “Generate Random Compound’ button.
         1. Makes sure compound generated is up to and including a pentadecane and correct format compared to pre gathered data.
   2. Expected Outcome
      1. The generated organic compound is not in the allowed compound list.

**Feature:** Customly Creating an Organic Compound and Viewing the Animation.

A. Input:

* 1. User goes through a series of mouse clicks in a grid.
  2. First button click.
  3. Second button click.

1. True Black Box Testing:
   1. Description
      1. Open program.
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
      4. User clicks ‘View Animation’ button.
      5. Exit program.
   2. Expected Outcome
      1. Program opens.
      2. User forms an acceptable organic compound.
      3. User is informed their custom compound is indeed correct.
      4. User views animation for organic compound.
      5. Exits program.
2. False Black Box Testing:
   1. Description
      1. Open program
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
   2. Expected Outcome
      1. Program opens.
      2. A message box is displayed stating that the user created a compound not allowed by the current functionality of the program.
3. True White Box Testing:
   1. Description.
      1. Open program.
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
         1. Checks pre gathered compound structures to ensure compound structure is indeed correct.
      4. User clicks ‘View Animation’ button.
         1. Checks pre gathered compound information to ensure animation follows the exact nomenclature procedure.
      5. Exit program.
   2. Expected Outcome
      1. Program opens.
      2. User constructed a valid organic compound within the current functionality of the program.
      3. User is informed their custom compound is correct.
      4. User views animation of naming process.
      5. Program closes.
4. False White Box Testing:
   1. Description
      1. Open Program.
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
   2. Expected Outcome
      1. Program opens.
      2. Error message displays the currently constructed compound is not within current constraints.

**Feature:** Customly Creates Organic Compound, Guesses Name and Views Animation.

A. Input.

* 1. User goes through a series of mouse clicks in boxes laid out in a grid.
  2. First button click.
  3. User enters in guess of name for the compound.
  4. Second button click.
  5. Third button click.

1. True Black Box Testing:
   1. Description
      1. Open program.
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
      4. User enters in a guess for the name of the compound.
      5. User clicks ‘Check Name’ button.
      6. User clicks ‘View Animation’ button.
      7. Closes program.
   2. Expected Outcome.
      1. Program opens.
      2. User constructs valid organic compound.
      3. User is informed their compound is indeed valid.
      4. User enters in the correct name for compound.
      5. User is informed their guess was correct.
      6. User views animation for the compound.
      7. Exits program.
2. False Black Box Testing:
   1. Description
      1. Open program.
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
      4. User enters in a guess for the name of the compound.
      5. User clicks ‘Check Name’ button.
   2. Expected Outcome.
      1. Program opens.
      2. User creates a valid organic compound.
      3. User enters in weird characters and numbers.
      4. Error message is displayed informing the user their guess name contained invalid characters.
3. True White Box Testing:
   1. Description
      1. Open program.
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
         1. Checks pre gathered data to ensure the custom organic compound is indeed allowed in current functionality and is indeed a compound.
      4. User enters in a guess for the name of the compound.
      5. User clicks ‘Check Name’ button.
         1. Checks pre gathered data to ensure the name of the organic compound and format is indeed correct.
      6. User clicks ‘View Animation’ button.
      7. Exits program.
   2. Expected Outcome
      1. Program opens.
      2. User indeed creates an acceptable organic compound.
      3. User is informed their custom compound is acceptable.
      4. User enters in a guess for the name of the compound.
      5. User is informed their guess is indeed correct.
      6. User views animation on naming procedure of the compound.
      7. Exits program.
4. False White Box Testing:
   1. Description
      1. Open program.
      2. User clicks in a series of boxes laid out in a grid.
      3. User clicks ‘Validate Compound’ button.
      4. User enters in guess for the name of the compound.
      5. User clicks ‘Check Name’ button.
   2. Expected Outcome
      1. Program opens.
      2. User creates a valid organic compound within the functionality of the program (pentadecane).
      3. User is informed their custom compound is indeed acceptable.
      4. User enters in guess for the name of the compound.
      5. Message box appears informing the user the format of their guess is incorrect.

System Functions:

Interfaces:

* System functions properly on Windows XP, Vista, 7 and 8.
* System functions properly on Linux, Unix and Mac OS.
* System functions properly with standard mouse and keyboard.
* System functions properly with standard monitors and resolutions.

Testing Machine’s System Requirements:

Recommended System Requirements:

1. System Hardware

* 1. Windows Systems:
     1. RAM - 128MB (64MB for Windows XP 32bit)
     2. Disc space 124MB.
  2. Linux Systems:
     1. RAM - 64MB.
     2. Disc space 58MB.
  3. MAC Systems:
     1. RAM - 128MB
     2. Disc space 124MB.
  4. Input/Output Devices:
     1. Contain at least 50mb of persistent memory.
     2. Standard monitor that can support 4:3 and 16:10 aspect ratios.
     3. Standard mouse and keyboard.
  5. Additional Software:
     1. Py2exe Python Interpreter
  6. Browsers:
     1. Internet Explorer 7.0 or higher.
     2. Firefox 3.6 for higher.
     3. Google Chrome.
  7. Transfer Medium:
     1. Internet access or USB drive port.

Approach:

Unit Testing:

Unit Testing will be conducted to ensure all methods and functions execute properly and return the desired results. This step will be conducted as White-Box testing since it takes knowledge of how the source code executions and the desired results. This step is being done during implementation to ensure gradual and positive progress. Our development team will be using PyUnit for unit testing in Python. There will be sample code implemented and later disregarded to tests queries from the database to ensure the correct and desired data is being pulled.

Integration Testing:

This session of testing will be conducted to ensure all classes and methods/functions come together to form one fully functioning program. This level of testing will be conducted as Black-Box testing as it requires zero knowledge of how the individual classes/methods/functions are executed in source or pseudo code. If issues occur during this stage, additional White-Box testing will be used to flesh out the issues. After they are resolved the team will resume integration testing. Our testing team will conduct integration testing by using a set of predefined test cases to see if the results match the predetermined outcomes.

Functional Testing:

This stage of testing will be conducted to ensure all requirements set by the customer are fully implemented in the code. Each use case will be executed and the results carefully monitored to see execution style and results. If results do not match the scenarios set forth by the customer, corrections will be made to the code and testing will resume after corrections are completed.

System Testing:

This stage of testing will be conducted to ensure the program functions properly and fluidly on all systems and environments containing the minimum specifications listed in the requirements documents. Our team will be taking the software to the computer labs and classrooms containing computers within the chemistry building and testing functionality to ensure desired results in the environment the software will primarily be used in. Next we will test functionality on laptops similar to or exactly like those used by the majority of students and professors to ensure functionality holds.

Alpha Testing:

This stage of testing will be conducted by having the customer test the software on one of the developer’s machines at our teams location or the customers office. This test will be conducted by Roland who is team leader. In this process, the customer will identify any issues that may exist with the current GUI design or functionality and report them directly to the development team. All issues will be fully documented and fixed according to the customers’ wishes and designs.

Beta Testing:

This stage of testing will be conducted at the customers’ environment of choice, which will be the chemistry buildings’ computer labs or classrooms containing computers. This test will be conducted by Roland. In this process, we will ensure all issues uncovered during Alpha testing have been resolved and the program is functioning to the customers’ wishes. If any other issues arise, they will be fully documented and corrected before release.

Acceptance Testing:

During this final stage of testing our team will deliver the final product to the customer. Along with the finished program, all paperwork and designs pertaining to this project will handed over to the client. The customer will check to ensure all issues have been resolved and all supporting documentation and designs are accounted for. This process will be conducted at the customers’ location and the software will be placed on all machines that the customer requests. If the computers have deep freeze installed, we will leave a hardcopy of the program with the customer on a flash drive or any other device the customer wishes. Finally, our team will ensure any and all questions or concerns the customer may have pertaining to this project are addressed.

Testing Deliverables:

Unit Testing:

For unit testing, our tester will include any functions/methods created as well as all information generated from PyUnit in the notebook. Some examples will be chosen of functions/methods containing the tests and results will also be included.

Integration Testing:

All previously tested modules will be combined to test cohesion. A simple form will be filled out stating any issues found and where those issues may originate. The form shall include the date being performed, the version number of the software, and a list of all key modules and important submodules being tested in unison. The document shall also include a list of test cases performed with their results and the name of the member who performed those tests.

Alpha Testing:

forms will be filled out by the customer stating any issues that may exist with the software from GUI design to functional issues. The issue(s) will be clearly stated by the customer along with how they wish it to be resolved. It will be accompanied by comments from Roland on how these issues will be resolved in accordance with the customers’ wishes, on the technical side, as well as date performed and estimated date of issues being resolved. Before finishing the session, the customer and Roland will sign off on the forms stating the customer found the issues and stated how they should be resolved.

Beta Testing:

Additional forms will be filled out by the customer stating any new issues and unresolved issues from GUI design to functional issues. The issue(s) will be clearly stated by the customer along with instructions on how they wish them to be resolved. It will be accompanied by comments by Roland on how these issues will be resolved in accordance with the customers’ wishes, on the technical side, as well as date performed and estimated date of issues being resolved. Before finishing the session, the customer and Roland will sign off on the forms stating the customer found the issues and gave instructions on how they wish for them to be resolved.

Acceptance Testing:

This form will be filled out by the customer stating the program meets all requirements set forth by them. The client will also verify all documentation and designs are included within the files given to them. Along with physicals items, the customer will confirm that all issues and concerns have been addressed and resolved to the best of the development team’s capabilities.