Testing types:

Unit testing: During development It can be easy to separate my 4 use cases (empty search, main name, secondary name, misspelled name) since none of them rely on each other to function just that the ones before it fails and as such I can test each one individually to make sure output is correct. Main name and secondary name searches are easy to make test cases for since they just look for exact matches in the database and return a list of matched objects. Misspelled names will need extensive unit test to ensure proper function and output e.g. If I put in qpumpkin I should get about the same results as searching pumpkin. I can split up the misspelled search into more parts and test them individually like the edit distance algorithm and weighted queue function to make sure each piece works as it should before moving into integration testing.

Integration testing: This will be done by top down integration testing as the most important/main control modules are finished first. The main module is the main name search once this is done we integrate the secondary name search and then the misspelled name search. The misspelled name search is itself broken into different parts like the edit distance algorithm and its weighted queue and must be put together piece by piece while being tested in between revisions. This is good since the driver to test the component won’t need to change very much just what string it inputs for the search

System testing: I don’t need to worry about most parts of system testing since my project is an expansion of a select part of an established web app. Recovery is unnecessary as putting in some recovery functionality into a search function might take up too much resources for no foreseeable gain. Security might be an issue but if input is sanitized correctly there should be no problem. Stress testing is important however since the misspelled names function could fail strangely on some specific inputs requiring much more testing than the other parts. There might also be a huge performance hit from repeatedly calling the misspelled name function since the algorithm used to calculate some of the data is very expensive computation wise, time might need to be spent in finding an alternate solution should it prove to be too much for the server to handle. Last is deployment testing and since this is a rails web app it should already run on most browsers which already run on most operating systems, so the only thing left for it is to make sure it runs correctly under some of the most popular web browsers.

Debugging:

Ruby on rails has an inbuilt debugger and ruby itself has a gem called ruby-debug which I will be using. Since they are purpose built for ruby and rails specifically they should help me catch stray values and what talks to what in growstuff’s application itself without having to look at every file in the growstuff repository.

Test cases:

|  |  |  |  |
| --- | --- | --- | --- |
| Functionality tested | Inputs | Expected output | Actual output |
| No input | NULL | Every item in database |  |
| Main name search | pumpkin | The search results page with every plant that has pumpkin in its name: pumpkin, Japanese pie pumpkin, Atlantic giant pumpkin, baby bear pumpkin, crown pumpkin |  |
| All caps main name | PUMPKIN | pumpkin, Japanese pie pumpkin, Atlantic giant pumpkin, baby bear pumpkin, crown pumpkin |  |
| Secondary name search | puha | The search results page with every plant that has the searched term as a secondary name: pūhā |  |
| All caps secondary name | PUHA | pūhā |  |
| Misspelled name | Tormato (misspelling of tomato) | The search results page with the top 10 results closest to the entered term: tomato, potato, beefsteak tomato… |  |
| All caps Misspelled name | Tormato (misspelling of tomato) | tomato, potato, beefsteak tomato |  |
| main name with special characters | a!!pp:le}}” | Same output for apple: Apple, apple mint… |  |
| Secondary name with special characters | Purpl$$%@#$^%#e granadilla | Passion fruit |  |
| Misspelled name with special characters | Gral^&\*e (misspelling of grape) | Grape, grapefruit, grape tomato, borage(3 matches with 3 indel’s), … should have less matches/more indel’s as the list progresses |  |