The Back-end was built on the single threaded Python asynchronous web framework Tornado exposed at */api/swathlib/upload/* . The exposed end point has two HTTP methods enabled, GET for checking accessibility and, OPTIONS and PUT for accepting user submission. Default setting allowed connection from all origins with the following headers *Origin, X-Requested-With, Content-Type, Accept.*

0.1 User Query Preprocessing

Upon receiving the query, SWATHLib would attempt to resolve any possible conflict of variable modifications by recursively generate all combination of them at the conflicted positions. If different Ytype transitions are included, only one Ytype transition can exist at the same time. For each combination of modifications, two modifications dictionaries would be generated, one for all static modifications in the combination, the other for all variable modifications in the combination.

Using the dictionaries containing information modifications' positions, the program, in case of multiple modification pattern, would attempt to again recursively iterate through all combination of positions with and without presence of the modifications to generate unique multiple variable modifications modification pattern.

0.2 Theoretical Ion Generation

If there are multiple positions of variable modifications available, using a combination of Python built-in package *itertools*'s functions *permutations* and *combinations*, we could generate a records of all possible states for each modification on the user input. Using a tree data structure, we iterate through all possible combinations of these different modification states.

On every yield, we would perform calculation of Y transitions first if available, then *by.* If the user had not selected any specific b or y, all transitions for them would be calculated, respectively.