Introduction

The overall objective for testing this system will simply be to ensure that it meets the requirements and goals outline in other Documents

Test Scope

* Java Database updater
  + The program does not require any inputs and the only modifications possible are through the confing .ini file. I was able to test two different aspects of the completed program. First I would drop the database table and modify the .ini file date section. I would add a date at the bottom row to have 2007. The expected output would be the creation of a download.csv file and the database would have documents 1-3285.
  + Modify config.ini to have -1 as the date. Expected outcome is that the download.csv would printout a copy for the website html, however there is no modification to the database as it is rejected by the input program.
* Microsoft SQL database
  + For the most part, this tool is a robust system and the only testing required was to prove that there were no duplicates and that data can be pulled from the correct database.
  + To test for duplicates, I set a primary key and redundancies were automatically removed.
  + To test for changes, I would clear the database and attempt to add a pre-defined set of datapoints. I would use the auto-updater to do this and load a given series of documents, such as all of 2007. From there I would manually check that all document ID’s from 1-4000 exist and are sequential without any missing documents.
  + To test front-end results, I would use the Query below with a random string or date  
    SELECT \* FROM CombinedTable WHERE desc LIKE=’%fish%’ AND Update\_Date=’2007’;
* Frontend
  + To test the frontend, I simply clicked around and made sure that there were no errors in the console or in the npm terminal in git bash
  + I mainly tested this for completeness