Team 11: “MiniWatt!”
Unit Testing

Dylan Brashear
Grant Jochums
Craig Wilhite
Adam Worthington
Robbie Mantzoros
Chris Doak

Product	Answer Finder Unit Testing		
Date	10/15/2015		
Author	Craig Wilhite		
Defect #	Description	Severity	How Corrected
1	Supply a null list of sources. Program does not check to see if there is an empty list of sources to pull from before trying to get an answer	1	At beginning of method “findAnswer”, check to see if the sources list is not null or empty.
2	Supply an empty question in method call to “findAnswer”. Program does not know what to do and halts.	1	Have a check to make sure that the question object has valid data before proceeding to try and find an answer.
3	Supply a question that has the question type of “WHY”, and a source that does not have the word “because” included.	3	Currently the program scans for the index of the word “because” in the source without checking to see if the index actually exists. Therefore, include a check to see that the index of “because” does exist.
4	Supply a question that has the question type of “WHEN”, and a source that does not have any dates.	3	Currently the program scans for the index of numbers & dates. Therefore, check to make sure that there is an index of dates before continuing with execution.
5	Supply a question that has the question type of “WHO”, and a source that does not have any words beginning with capital letters.	3	Currently the program scans for presence of words with capital letters. Include a scan that will search for case-irrelevant matches with question subjects.
6	Supply a question that has	2	Add a check at the end of

	the question type of “INVALID”. A null list of answers will be returned		“findAnswers” that will include the answer: “no answer” with 100% certainty if an INVALID question type is asked.

Product	Question Unit Testing		
Date	10/15/2015		
Author	Grant Jochums		
Case#	Description	Severity	How Corrected
1	passing an empty string causes the program to return an empty Question object	2	By checking for empty strings, this can be avoided
2	If a question is passed that does not have a question word in it, the system will mark it as invalid but never report that to the restful api.	2	add a check for invalid questions that will report the error to the backend.
3	Multi word subjects that are not capitalized do not get recognized	3	Add detection for filler words and maybe length analysis to find complete subjects.
4	The question “When did the King of Prussia die?” will cause the word die to be counted as part of the prepositional phrase	3	Change how the end of prepositional phrases is located.
5	If keywords are too close to the subject or prepositional phrases, then they get stuck in with the thing they’re close to	2	This could be fixed by changing the way subjects are located and stop locating prep phrases because they add little to the context.
6	If a sentence is just a question word, an error occurs.	1	change the way words are parsed.

Product	Front End GUI Unit Testing
Date	9/24/2015
Author	Chris Doak & Robbie Mantzoros

Case#	Description	Severity	How Corrected
1	Put in garbled/nonsensical text into questions box and no source material then clicked submit. The program is supposed to signal server error, but it was unhandled.	2	Added exception handling for error returned from server.
2	Passing an invalid URL caused the request to return null because of no communication	1	Passing the correct URL and checking if it is proper
3	Added a pdf file for questions material, and text for source material, and the app crashed.	2	Corrected the if statement that branched based on if the source is input as text or as file.
4	Receiving an empty HttpResponse fails to send error message.	3	Check for “null” response and not an empty response
5	Submitting a questions file and text source throws an error message	2	Fix the logic that checks what format the questions and source are in
6	Tasks can be posted with null or empty parts	2	Trigger an error if any of the parts of the task are null or empty

Product	Back End Server Unit Testing		
Date	10/15/2015		
Author	Adam Worthington & Dylan Brashear		
Case#	Description	Severity	How Corrected
1	Receive garbage data request would cause a null error to halt the server.	2	Changed methods to now require parameters and added checks against null values.

2	Sending files over http would cause google app engine to terminate due to improper data handling (no blobs).	1	Changed parameter inputs from file formats to byte strings or straight question strings.
3	The question “What language did the people living in the village of Llanfairpwllgwyngyllgogerych wyrndrobwllllantysiliogogogoch speak one hundred years ago?” is not stored in the database, as the name is too long for that space in the table	3	Changed the datatype from character to varchar to accommodate such long strings.

Product	Web Crawler Unit Testing		
Date	10/15/2015		
Author	Adam Worthington		
Case#	Description	Severity	How Corrected
1	Feed the web crawler a null request to find. It will cause the current request to return a rather large error report.	1	Check for improper inputs before instantiating a web crawler to search.
2	Feeding in a empty request, “” will result in broken links being returned and no data being found.	3	Check for subject length.
3	Strings that have spaces in them would report a malformed URL error from the scraper of the web crawler.	1	URL encode the string before requesting information from google.