Automated Testing Using the Sugar Framework

Shaina Mainar, David Spry, AJ Williams

Department of Computer Science, College of Charleston, Charleston, SC

Abstract

A last year Computer Science student needs to be equipped with the proper skillset to launch their careers after graduation. CSCI 362 Software Engineering is the class that enables the student to explore the roles and tools that they will possibly take on in industry. This team project is the culmination of all the tools learned and developed by the students in previous major classes. The students were expected to complete this project from start to finish in teams with guidance from Dr. Bowring.

Goals

- Create a testing framework to test individual methods
- Automate testing using Bash scripting methods
- Inject faults in the source code and observe the results
- Display the results on a stylized HTML file

Motivation

- Computer Science students need to be able to prepare for industry by utilizing tools and enhancing skillsets necessary to succeed.
- Hands-on experience with the Software Engineering Life Cycle.

Methodology

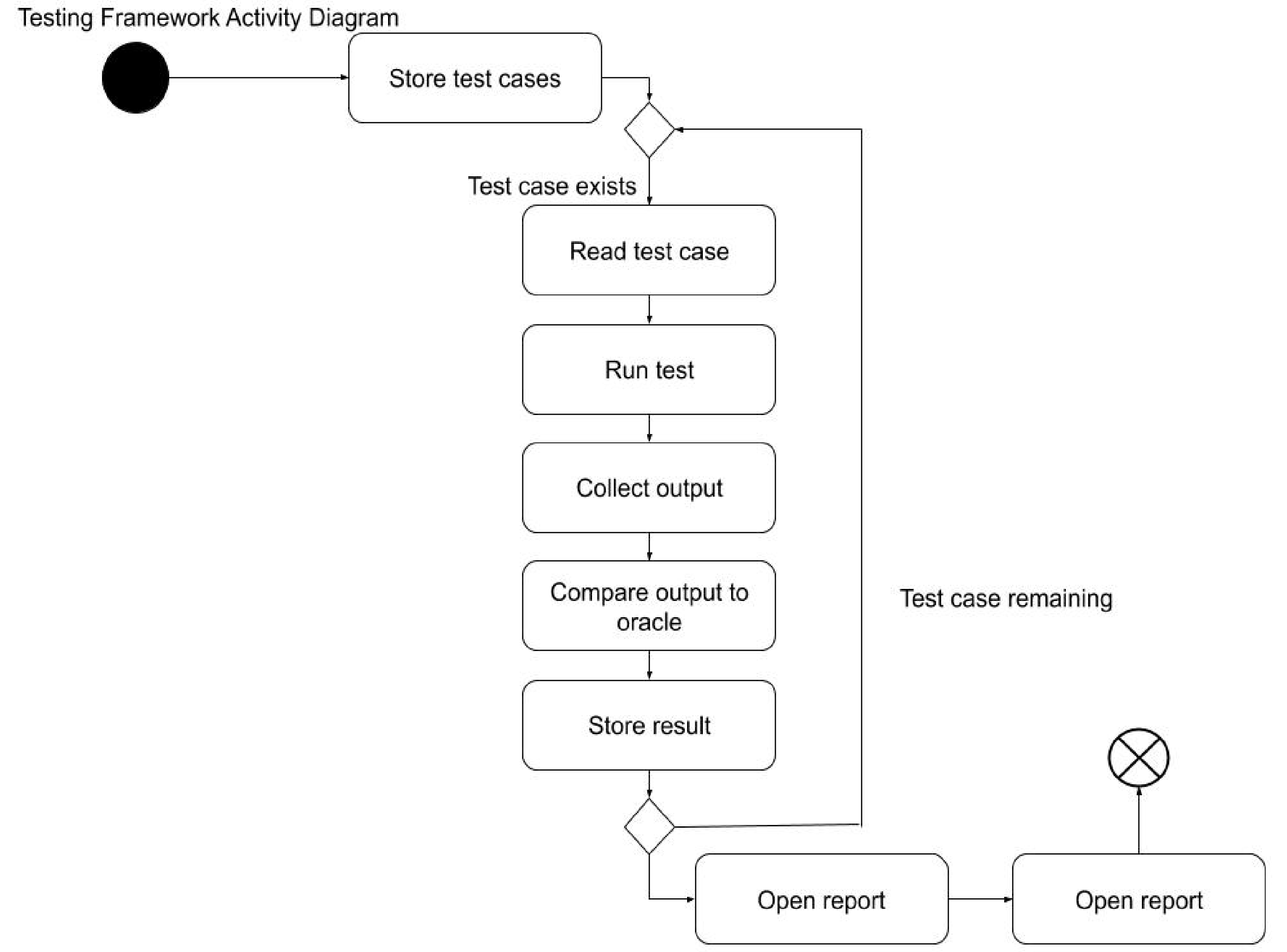
Testing:

- The source code is written almost exclusively in Python 2, therefore we found that it would best to write our test framework in Python.
- Since Python files do not need to be compiled, we were able to access Python methods in the command line using "python -c "import [FILE_NAME]; print([FILE_NAME].[METHOD_NAME]([INPUT]))"
- Compile-time faults were injected in the source code to test robustness.

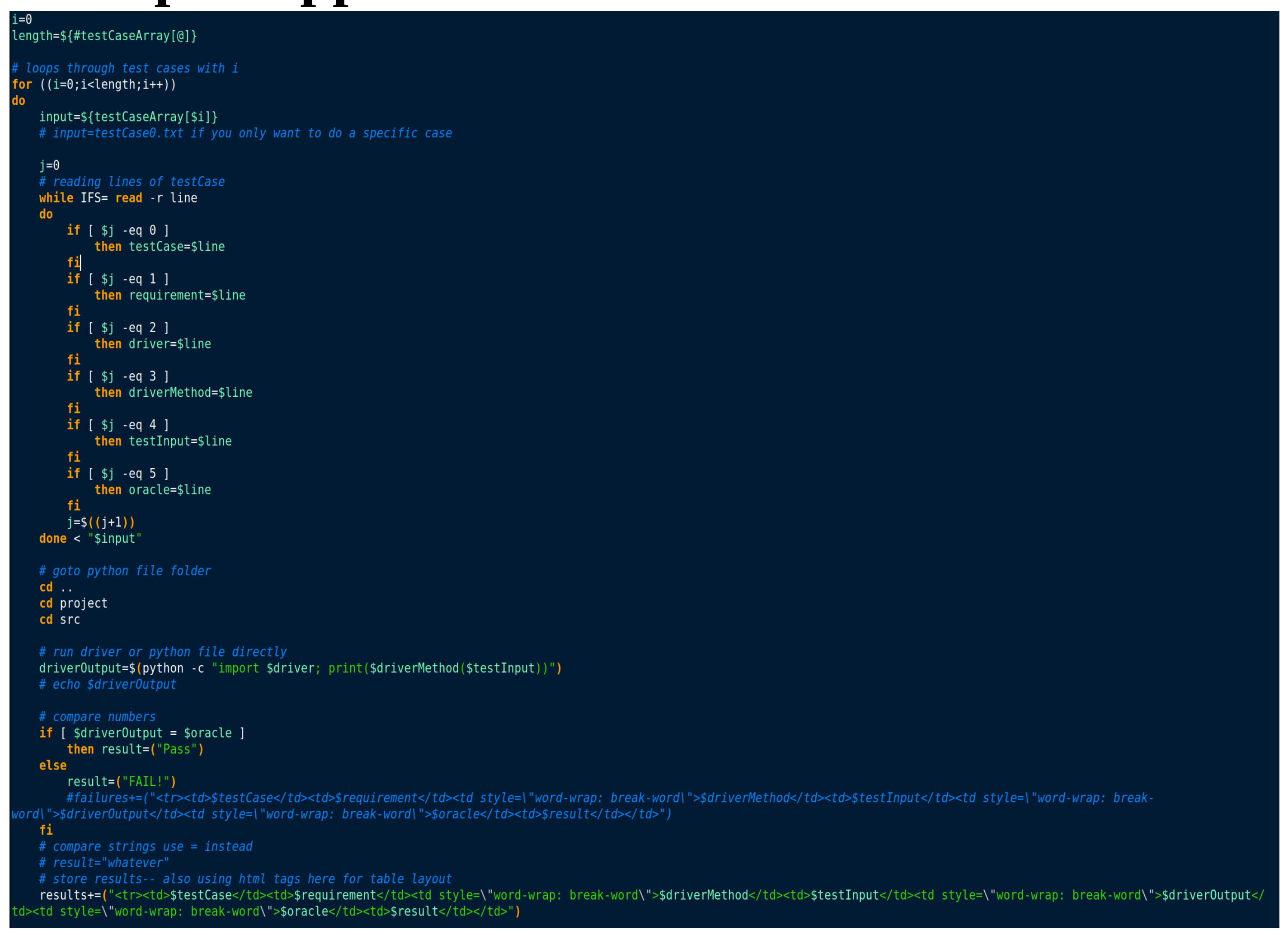
Scripting:

- According to the project specifications, the script is run in Ubuntu, therefore it is easier to write the script in Bash.
- The script is called within the /TestAutomation directory using "bash ./scripts/runAllTests.sh"
- The results are output into a formatted HTML file.

Activity Diagram



Script Snippet

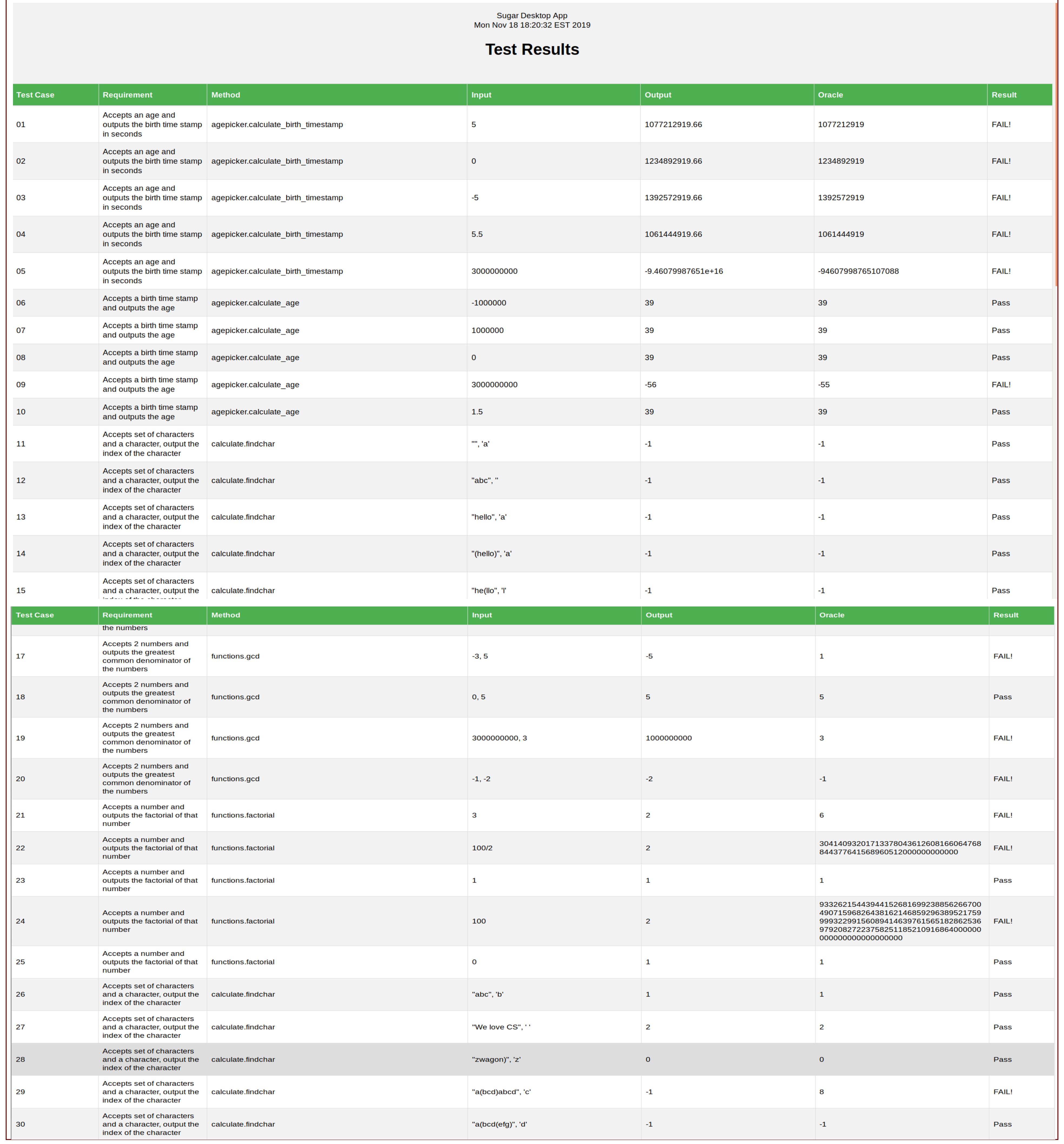


```bash
i=0
length=${#testCaseArray[@]}

# loops through test cases with i
for ((i=0;i<length;i++))
do
    input=${testCaseArray[$i]}
    # input=testCase0.txt if you only want to do a specific case

    j=0
    # reading lines of testCase
    while IFS= read -r line
    do
        if [ $j -eq 0 ]
            then testCase=$line
        fi
        if [ $j -eq 1 ]
            then requirement=$line
        fi
        if [ $j -eq 2 ]
            then driver=$line
        fi
        if [ $j -eq 3 ]
            then driverMethod=$line
        fi
        if [ $j -eq 4 ]
            then testInput=$line
        fi
        if [ $j -eq 5 ]
            then oracle=$line
        fi
        j=$((j+1))
    done < "$input"

    # goto python file folder
    cd ..
    cd project
    cd src

    # run driver or python file directly
    driverOutput=$(python -c "import $driver; print($driverMethod($testInput))")
    # echo $driverOutput

    # compare numbers
    if [ $driverOutput = $oracle ]
        then result=("Pass")
    else
        result=("FAIL!")
        #failures+=("<tr><td>$testCase</td><td>$requirement</td><td style=\"word-wrap: break-word\">$driverMethod</td><td>$testInput</td><td style=\"word-wrap: break-
word\">$driverOutput</td><td style=\"word-wrap: break-word\">$oracle</td><td>$result</td></td>")
    fi
    # compare strings use = instead
    # result="whatever"
    # store results-- also using html tags here for table layout
    results+=("<tr><td>$testCase</td><td>$requirement</td><td style=\"word-wrap: break-word\">$driverMethod</td><td>$testInput</td><td style=\"word-wrap: break-word\">$driverOutput</
td><td style=\"word-wrap: break-word\">$oracle</td><td>$result</td></td>")
```

Conclusions

All project specifications were met within the deadline. The team was able to divide task accordingly and with fairness. We were able to pick up roles that were simultaneously new and catered to our individual strengths. The project followed the Software Engineering life cycle from the creation of the test plan to the final presentation. We followed a plan-driven development approached and it was the best approach for the task at hand. Our skillsets were further developed, and we got a taste of what industry will be like.

Results



Sugar Desktop App
Mon Nov 18 18:20:32 EST 2019

Test Results

Test Case	Requirement	Method	Input	Output	Oracle	Result
01	Accepts an age and outputs the birth time stamp in seconds	agepicker.calculate_birth_timestamp	5	1077212919.66	1077212919	FAIL!
02	Accepts an age and outputs the birth time stamp in seconds	agepicker.calculate_birth_timestamp	0	1234892919.66	1234892919	FAIL!
03	Accepts an age and outputs the birth time stamp in seconds	agepicker.calculate_birth_timestamp	-5	1392572919.66	1392572919	FAIL!
04	Accepts an age and outputs the birth time stamp in seconds	agepicker.calculate_birth_timestamp	5.5	1061444919.66	1061444919	FAIL!
05	Accepts an age and outputs the birth time stamp in seconds	agepicker.calculate_birth_timestamp	3000000000	-9.46079987651e+16	-94607998765107088	FAIL!
06	Accepts a birth time stamp and outputs the age	agepicker.calculate_age	-1000000	39	39	Pass
07	Accepts a birth time stamp and outputs the age	agepicker.calculate_age	1000000	39	39	Pass
08	Accepts a birth time stamp and outputs the age	agepicker.calculate_age	0	39	39	Pass
09	Accepts a birth time stamp and outputs the age	agepicker.calculate_age	3000000000	-56	-55	FAIL!
10	Accepts a birth time stamp and outputs the age	agepicker.calculate_age	1.5	39	39	Pass
11	Accepts set of characters and a character, output the index of the character	calculate.findchar	"", 'a'	-1	-1	Pass
12	Accepts set of characters and a character, output the index of the character	calculate.findchar	"abc", ''	-1	-1	Pass
13	Accepts set of characters and a character, output the index of the character	calculate.findchar	"hello", 'a'	-1	-1	Pass
14	Accepts set of characters and a character, output the index of the character	calculate.findchar	"(hello)", 'a'	-1	-1	Pass
15	Accepts set of characters and a character, output the	calculate.findchar	"he(llo", 'l'	-1	-1	Pass
	the numbers					
17	Accepts 2 numbers and outputs the greatest common denominator of the numbers	functions.gcd	-3, 5	-5	1	FAIL!
18	Accepts 2 numbers and outputs the greatest common denominator of the numbers	functions.gcd	0, 5	5	5	Pass
19	Accepts 2 numbers and outputs the greatest common denominator of the numbers	functions.gcd	3000000000, 3	1000000000	3	FAIL!
20	Accepts 2 numbers and outputs the greatest common denominator of the numbers	functions.gcd	-1, -2	-2	-1	FAIL!
21	Accepts a number and outputs the factorial of that number	functions.factorial	3	2	6	FAIL!
22	Accepts a number and outputs the factorial of that number	functions.factorial	100/2	2	30414093201713378043612608166064768844377641568960512000000000000	FAIL!
23	Accepts a number and outputs the factorial of that number	functions.factorial	1	1	1	Pass
24	Accepts a number and outputs the factorial of that number	functions.factorial	100	2	93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000	FAIL!
25	Accepts a number and outputs the factorial of that number	functions.factorial	0	1	1	Pass
26	Accepts set of characters and a character, output the index of the character	calculate.findchar	"abc", 'b'	1	1	Pass
27	Accepts set of characters and a character, output the index of the character	calculate.findchar	"We love CS", ' '	2	2	Pass
28	Accepts set of characters and a character, output the index of the character	calculate.findchar	"zwagon)", 'z'	0	0	Pass
29	Accepts set of characters and a character, output the index of the character	calculate.findchar	"a(bcd)abcd", 'c'	-1	8	FAIL!
30	Accepts set of characters and a character, output the index of the character	calculate.findchar	"a(bcd(efg)", 'd'	-1	-1	Pass

Retrospect

In the beginning, the team was a bit befuddled. We were not sure how to start and where to start. With the guidance of Dr. Bowring and multiple discussions about the project specifications, we slowly, but surely, began to start our project. It seems that it was easier to tackle one problem at a time. Our first issue was the source code was outdated and was in the middle of being ported to Python 3 therefore, it was a bit of a challenge to get it up and running. Once we were able to get it running, everything else was easier. The scripting was a learning curve and so was the overall system framework. We had to understand how the different parts interacted before we created them. Overall, this was a great starting point and a peek at what industry may look like. It would be interesting to see future teams use agile approaches or create their own source code.