**Lab 1 Report**

黃偉哲

107598019

2019/3/6

1. **Test Plan**
   1. **Test requirements**

The Lab 1 requires to (1) select 15 methods from 6 classes of the SUT (GeoProject), (2) design Unit test cases based on the experience or intuition for the selected methods, (3) develop test scripts to implement the test cases, (4) execute the test script on the selected methods, and (5) report the test result.

In particular, based on the statement coverage criterion, the **test requirements** for Lab 1 are to design test casesfor each selected method so that “*each statement of the method will be covered by at least one test case* and *the minimum statement coverage is 40%*”.

* 1. **Strategy**

To satisfy the test requirements listed in Section 1, a proposed strategy is to

1. select those public methods that are easy to understand and have primitive types of input and output parameters (if possible).
2. set the objective of the minimum statement coverage to be 50% initially and (if necessary) adjust the objective based on the time available.
3. learn the necessary skills and tools as soon as possible.
4. design the test cases for those selected methods by considering
   1. the possible **valid values** and **combinations** of the input parameters.
   2. the **boundary values** of the input parameters.
   3. **Test activities**

To implement the proposed strategy, the following activities are planned to perform.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Activity Name** | **Plan hours** | **Schedule Date** |
| 1 | Study GeoProject | 2 | 2019/3/3 |
| 2 | Learn JUnit | 0.5 | 2019/3/3 |
| 3 | Design test cases for the selected methods | 1 | 2019/3/3 |
| 4 | Implement test cases | 3 | 2019/3/3 |
| 5 | Perform test | 1 | 2019/3/5 |
| 6 | Complete Lab1 report | 2 | 2019/3/6 |

* 1. **Success criteria**

All test cases designed for the selected methods must pass (or "90% of all test cases must pass) and *the statement coverage should have achieved at least 40%.*

1. **Test Design**

To fulfill the test requirements listed in section 1.1, the following methods are selected and corresponding test cases are designed.

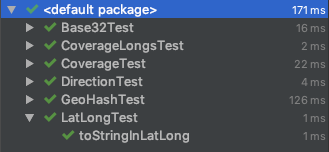
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Class** | **Method** | **Test Objective** | **Inputs** | **Expected Outputs** |
| 1 | Base32 | encodeBase32(long i, int length) | encodeBase32isNegative() | -75314, 4 | -29jk |
| 2 | Base32 | encodeBase32(long i, int length) | encodeBase32isNotNegative() | 75324, 4 | 29jw |
| 3 | Base32 | encodeBase32(long i) | encodeBase32() | 75324 | 0000000029jw |
| 4 | Base32 | decodeBase32(String hash) | decodeBase32isNegative() | -29jk | 75324 |
| 5 | Base32 | getCharIndex(char ch) | getCharIndex() | b | 10 |
| 6 | Base32 | getCharIndex(char ch) | getCharIndexisNull() | a | not a base32 character: a |
| 7 | GeoHash | right(String hash) | right() | 29jw | 29jy |
| 8 | GeoHash | left(String hash) | left() | 29jw | 29jq |
| 9 | GeoHash | top(String hash) | top() | 29jw | 29jx |
| 10 | GeoHash | bottom(String hash) | bottom() | 29jw | 29jt |
| 11 | GeoHash | neighbours(String hash) | neighbours() | 29jw | {29jq, 29jy, 29jx, 29jt, 29jr, 29jm, 29jz, 29jv} |
| 12 | GeoHash | widthDegrees(int n) | widthDegrees() | 13 | 4.190951585769653E-8 |
| 13 | GeoHash | adjacentHash(String hash, Direction direction) | adjacentHash() | 29jw, Direction.LEFT, 3 | 29hy |
| 14 | GeoHash | adjacentHash(String hash, Direction direction) | adjacentHashStepsIsNegative | 29jw, Direction.LEFT, -3 | 29nq |
| 15 | GeoHash | heightDegrees(int n) | heightDegreesGreaterThan12 () | 14 | 5.238689482212067E-9 |
| 16 | GeoHash | heightDegrees(int n) | heightDegreesLessThan12() | 2 | 5.625 |
| 17 | GeoHash | encodeHash(LatLong p, int length) | encodeHash() | 2.3, 6.8 | s0kv4dxw7rpd |
| 18 | Coverage | getHashes() | getHashes() | {29jq, 29jy, 29jx, 29jt, 29jr, 29jm, 29jz, 29jv} | {29jq, 29jy, 29jx, 29jt, 29jr, 29jm, 29jz, 29jv} |
| 19 | Coverage | getRatio() | getRatio() | 6.6 | 6.6 |
| 20 | Coverage | getHashLength() | getHashLength() | 4 | 4 |
| 21 | Coverage | getHashLength() | getHashLengthIfHashEqualZero() | 0 | 0 |
| 22 | Coverage | toString() | toStringInCoverage() | Coverage [hashes=[29jz, 29jy, 29jx, 29jv, 29jt, 29jr, 29jq, 29jm], ratio=6.6] | Coverage [hashes=[29jz, 29jy, 29jx, 29jv, 29jt, 29jr, 29jq, 29jm], ratio=6.6] |

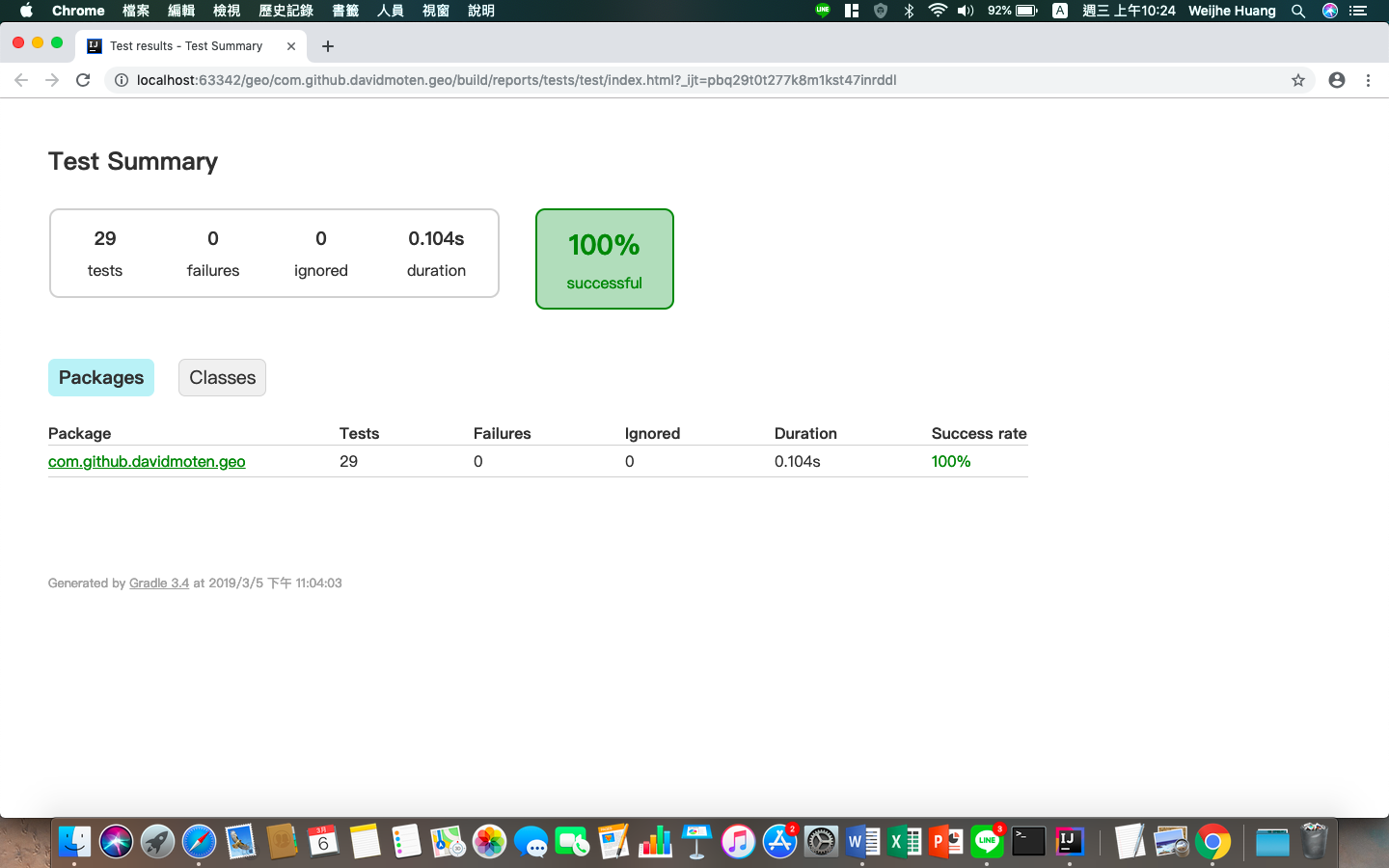
1. **Test Implementation**

The design of test cases specified in Section 2 was implemented using JUnit 4. The test script of 3 selected test cases are given below. The rest of test script implementation can be found in the [link](https://github.com).

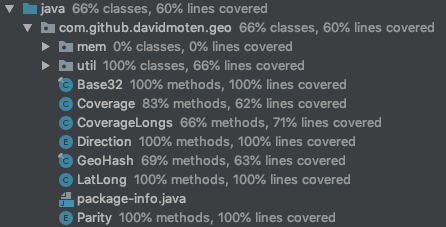
|  |  |  |
| --- | --- | --- |
| **No.** | **Test method** | **Source code** |
| 1 | GeoHash.heightDegrees(int n) | @Test  public void heightDegreesLessThan12() throws Exception{  double result = GeoHash.heightDegrees(2);  assertEquals(5.625, result,0.001);  } |
| 2 | GeoHash .encodeHash(LatLong p, int length) | @Test  public void encodeHash() throws Exception{  String encodeHash = GeoHash.encodeHash(2.3, 6.8);  assertEquals("s0kv4dxw7rpd", encodeHash);  } |
| 3 | Base32.getCharIndex( char ch) | @Test  public void getCharIndex() throws Exception{  Integer result = Base32.getCharIndex('b');  long intgerToLong = result;  assertEquals(10, intgerToLong);  } |

1. **Test Results**
   1. **JUnit test result snapshot**

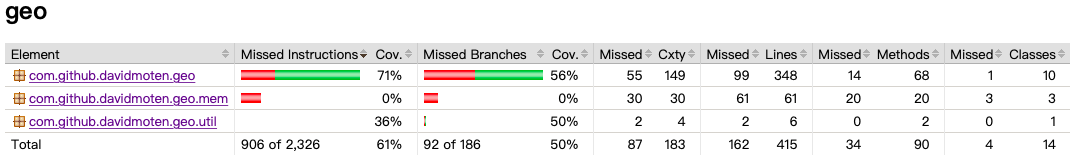
****

****

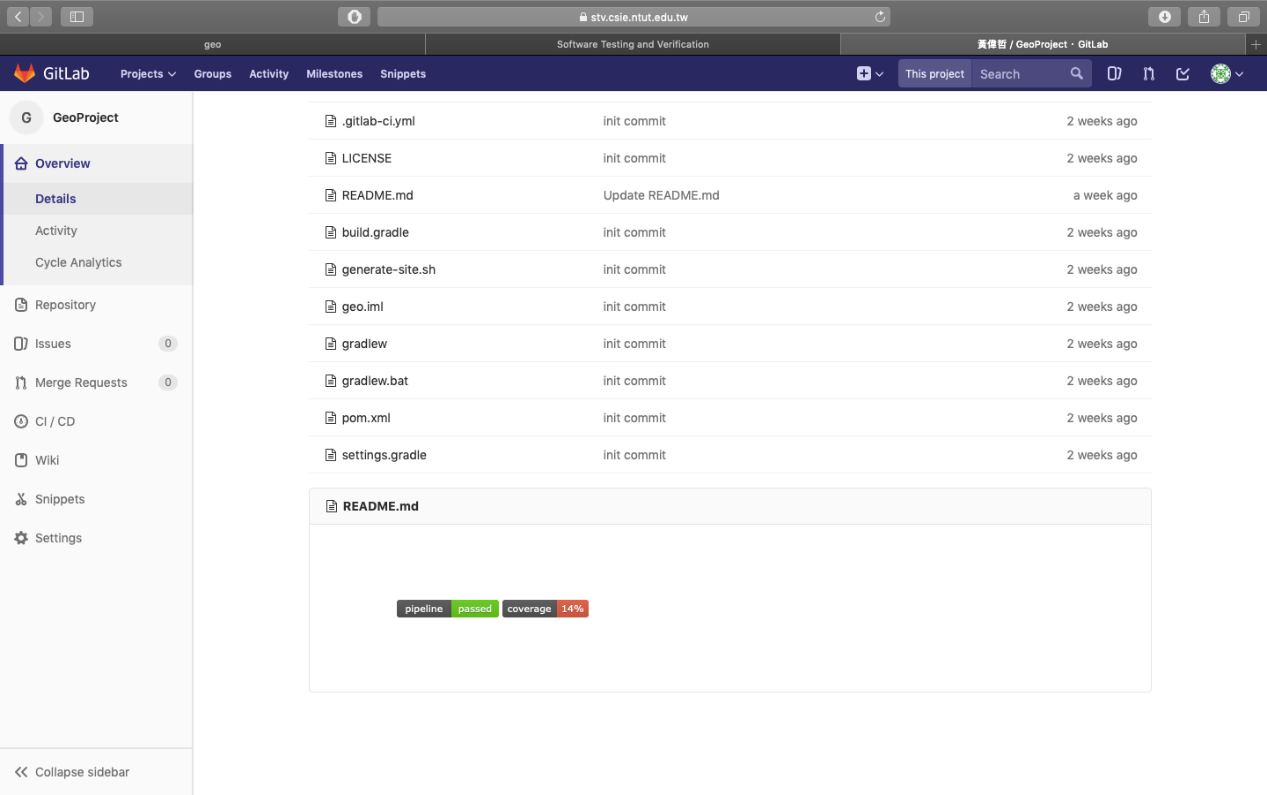
* 1. **Code coverage snapshot**
* Coverage of each selected method

****

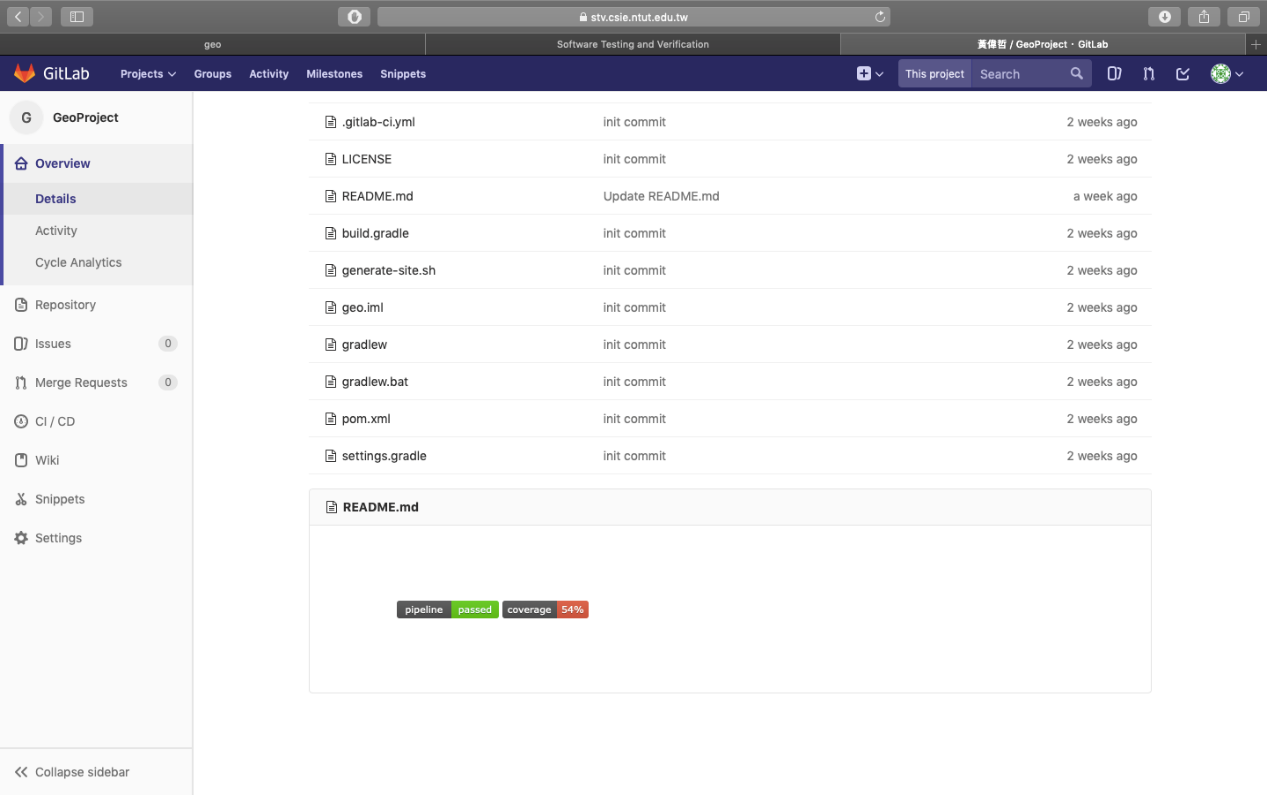
* Total coverage

****

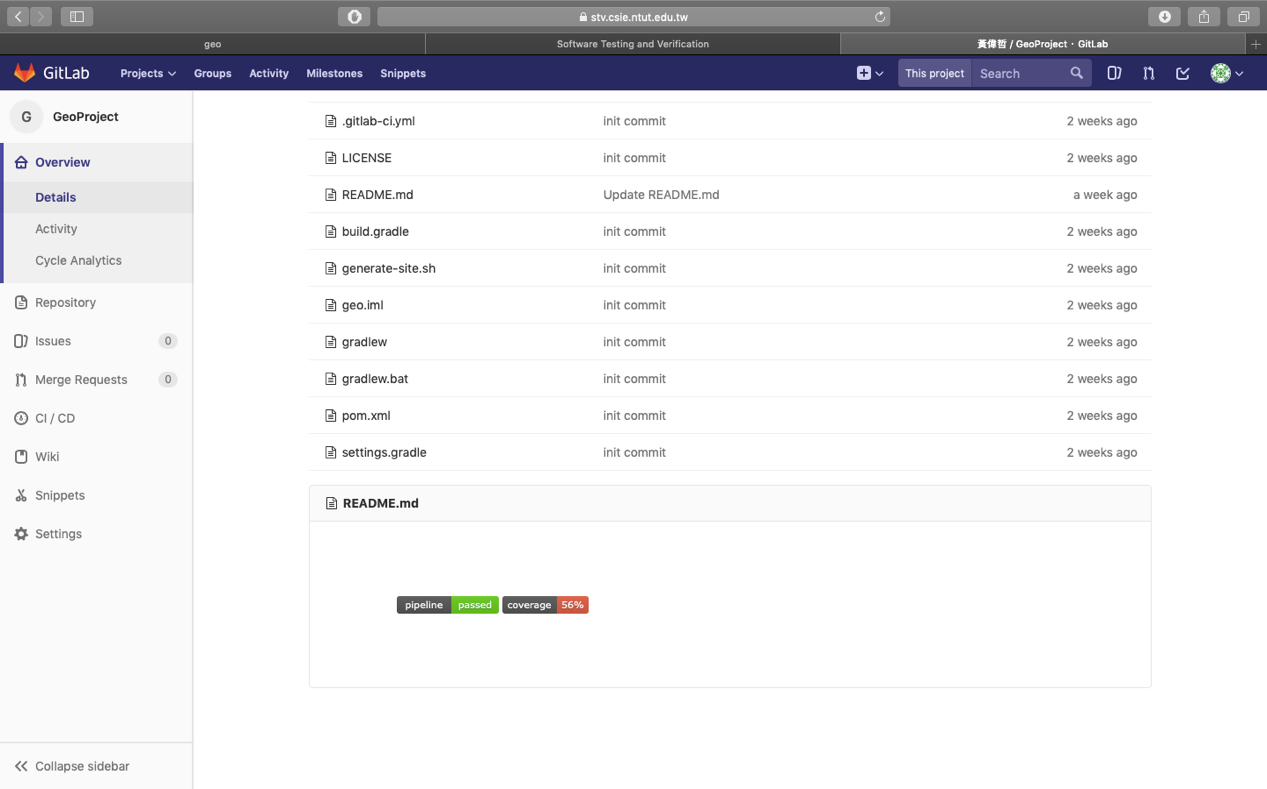
* 1. **CI result snapshot (3 iterations for CI)**
* CI#1

****

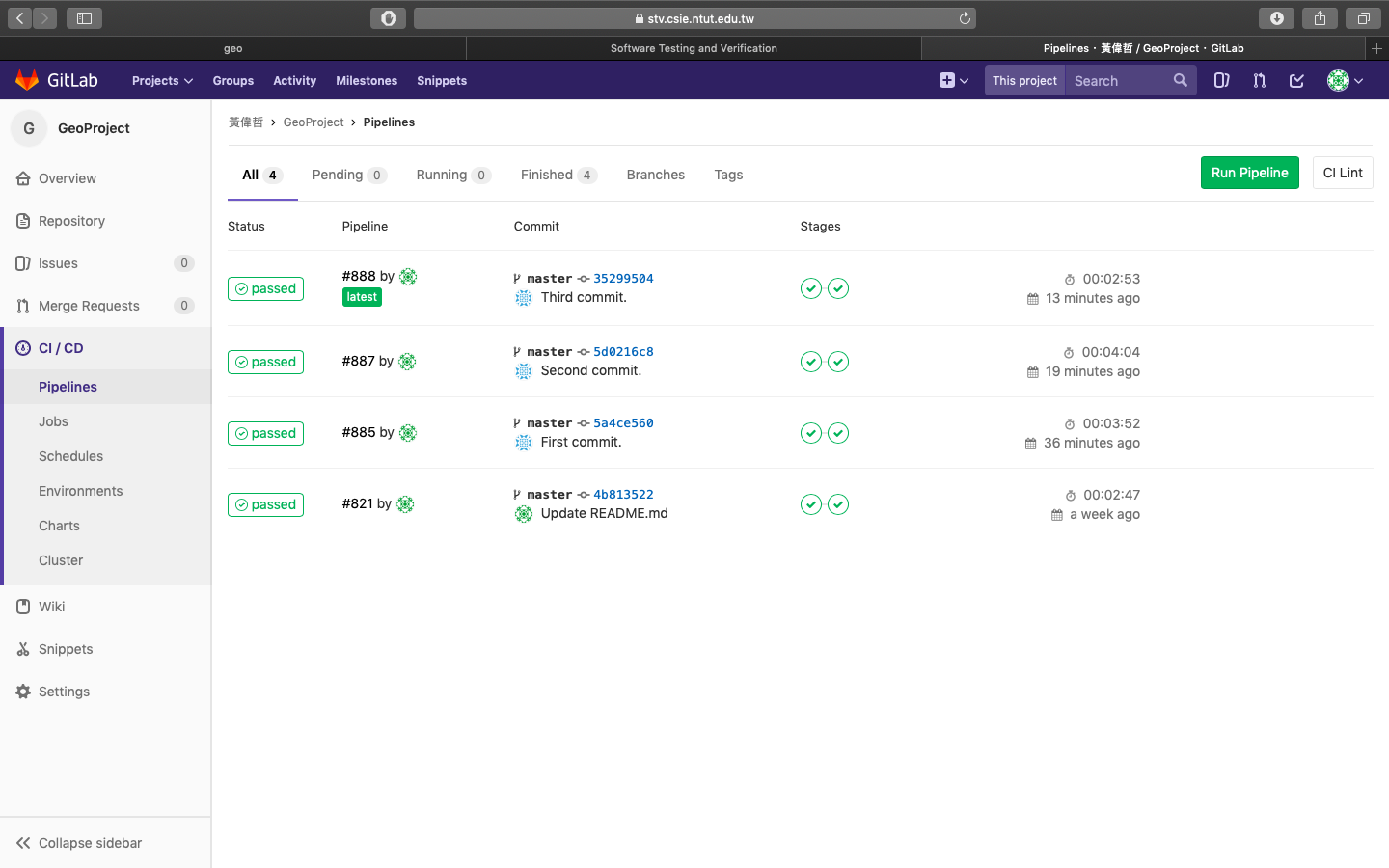
* CI#2



* CI#3

****

* CI Pipeline

****

1. **Summary**

In Lab 1, **15 test cases have been designed and implemented using JUnit**. The test is conducted in 3 CI and **the execution results of the 15 test methods are all passed**. **The total statement coverage of the test is 40%.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are …