

Venkata Satyanarayana Vara Prasad

Will Braynen

CS-561 Software Engineering Methods

23 January 2022

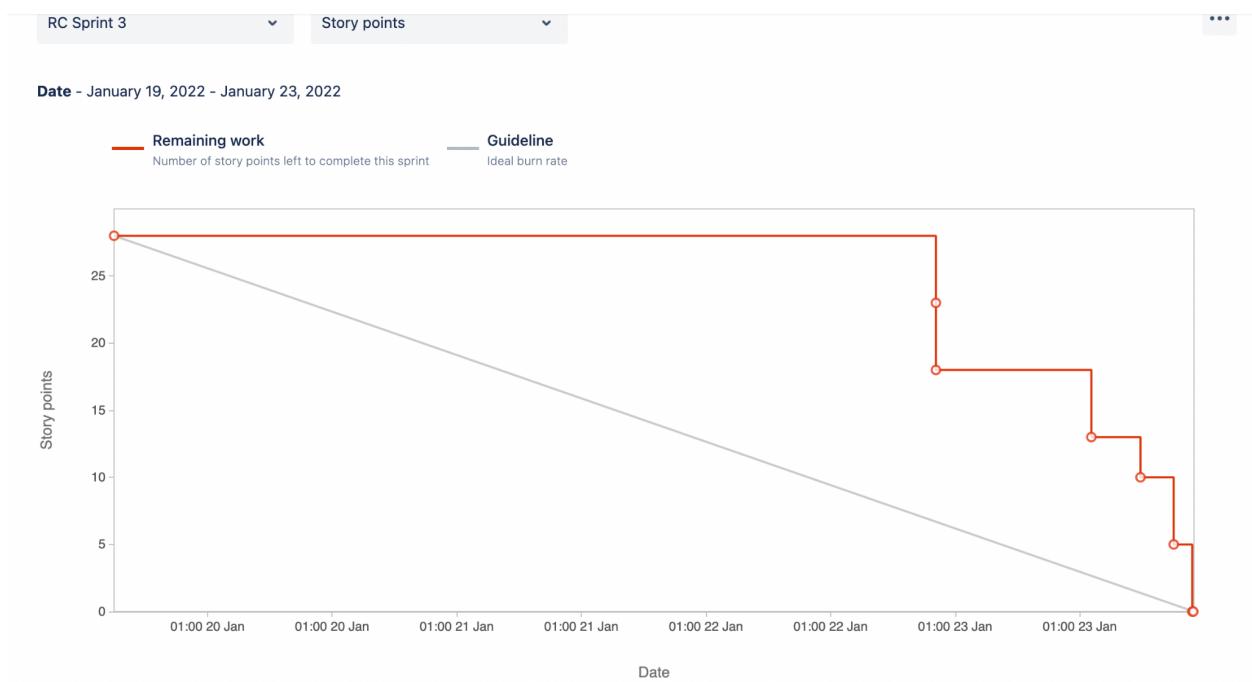
## ASSIGNMENT 3

### 1. Scrum

Board

The screenshot shows the Jira Software interface for a project titled "Rajalingamgari - CS561". The current sprint is "RC Sprint 3". The board is divided into three columns: "TO DO", "IN PROGRESS", and "DONE 6 ISSUES". A screenshot of a terminal window displaying Node.js code is pasted into the "DONE 6 ISSUES" column. At the bottom right, there is a "Quickstart" button.

## 2. Burndown Chart:



### 3. AWS Hands-on Training:

## Section 7 :

The screenshot shows a web browser window for the DigitalCloud AWS Certified Solutions Architect Associate Hands-on Labs. The URL is <https://digitalcloud.training/courses/aws-certified-solutions-architect-associate-hands-on-labs/sections/section-7-amazon-simple-storage-service>. The page title is "Challenge Labs". The main content area displays "SECTION 7 OF 16" and "Section 7: Amazon Simple Storage Service (S3) [1hr 55m]". Below this is a "Take Notes" button. The "Section Content" table lists three lessons: "Section 7—Introduction", "Amazon S3-Overview", and "Amazon S3-Storage Classes", all marked as 100% complete with green checkmarks. To the right, a sidebar titled "Section 7: Amazon Simple Storage Service (S3)" shows a list of 27 lessons, each with a green checkmark indicating completion.

Lesson	Status
Section 7—Introduction	Completed
Amazon S3-Overview	Completed
Amazon S3-Storage Classes	Completed
[HOL] Create Amazon S3 Bucket	Completed
IAM Policies; Bucket Policies and...	Completed
[HOL] Access Control Lists (ACLs)	Completed
[HOL] Bucket and User Policy Pra...	Completed
S3-Versioning; Replication and Lif...	Completed
[HOL] Versioning and Replication	Completed
[HOL] Lifecycle Rules	Completed
MFA with Amazon S3	Completed
S3-Encryption	Completed

The screenshot shows a macOS desktop environment with two terminal windows open and the Dock at the bottom.

**Terminal 1:** The title bar says "Terminal Shell Edit View Window Help". The command run is "docker inspect 7fa1bf1713a3". The output shows a container configuration with fields like Id, Created, Path, Args, and State (Status: running).

```
prasadrajalingamgari@Prasads-MacBook-Pro SE % docker inspect 7fa1bf1713a3
[{"Id": "7fa1bf1713a3b9d7b5d2aba173f48e4965c012812e10dc8052e2e0b776420166",
 "Created": "2022-01-23T23:58:38.503411627Z",
 "Path": "bash",
 "Args": [],
 "State": {
     "Status": "running",
     "Running": true,
     "Paused": false,
     "Restarting": false,
```

**Terminal 2:** The title bar says "Sun Jan 23 4:45 PM". The command run is "com.docker.cli - docker-compose run rust-client". The output shows a stack trace for a module compilation error, followed by commands being run in the container (vim app.js and node app.js), and a message about the application listening on localhost.

```
at Module._compile (internal/modules/cjs/loader.js:963:27)
at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10)
at Module.load (internal/modules/cjs/loader.js:863:32)
at Function.Module._load (internal/modules/cjs/loader.js:708:14)
at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:60:12)
at internal/main/run_main_module.js:17:47
root@7fa1bf1713a3:/code# vim app.js
bash: command not found
root@7fa1bf1713a3:/code# node app.js
Example app listening at http://localhost:${port}
```

**Bottom:** The Dock contains icons for various applications including Finder, Mail, Safari, and others.

```

SyntaxError: Invalid or unexpected token
  at wrapSafe (internal/modules/cjs/loader.js:915:16)
  at Module._compile (internal/modules/cjs/loader.js:963:27)
  at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10)
  at Module.load (internal/modules/cjs/loader.js:863:32)
  at Function.Module._load (internal/modules/cjs/loader.js:708:14)
  at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:60:12)
  at internal/main/run_main_module.js:17:47
root@7fa1bf1713a3:/code# vim app.js
bash: vim: command not found
root@7fa1bf1713a3:/code# node app.js
Example app listening at http://localhost:${port}

Compiling weather v0.1.0 (/code/cs561-rust)
  Finished dev [unoptimized + debuginfo] target(s) in 1m 57s
root@7fa1bf1713a3:/code/cs561-rust# cargo run
  Finished dev [unoptimized + debuginfo] target(s) in 0.35s
    Running `target/debug/weather`

Weather from a JSON we hard-coded locally:
Weather { main: Main { temp: 30.94 } }

Weather from openweathermap.org:
Weather { main: Main { temp: 281.59 } }
root@7fa1bf1713a3:/code/cs561-rust#

```

#### 4. MockWeatherService (*let myLibrary = MyLibrary(weatherService: mockWeatherService)*)

class doesn't need an async call because it is restricted to our localhost and there exists no api calls but while using a WeatherServiceImpl (*let myLibrary = MyLibrary()*) requires a communication with the openweathermap api.

In the first instance when we debug the console displays the output as

One

Two

Three

Four

While in the Second scenario, we can see an output as

One

Three

Two

Four

This is because at line 22 :

```
myLibrary.isLucky(number, completion: { lucky in  
    print("Two")  
  
    isLuckyNumber = lucky  
  
    expectation.fulfill()  
})
```

This code block is redirected to an asynchronous wait for it to hit the openweathermap endpoint skipping that code block, the execution continues further. And a wait count is initiated and when the wait count exceeds 5, an exception is invoked.

The async call sends back a response or exceeds the wait time to throw an exception and the execution further continues for the assertions.

The assertions are then validated to pass the tests.

MyLibraryTests

```
let myLibrary = MyLibrary(weatherService: mockWeatherService)
let myLibrary = MyLibrary()
let number = 8

let expectation = XCTestExpectation(description: "We asked about the number 8 and heard back One")
var isLuckyNumber: Bool?
print("One")

// When
myLibrary.isLucky(number, completion: { lucky in
    print("Two")
    isLuckyNumber = lucky
    expectation.fulfill()
})
```

Test Suite 'Selected tests' started at 2022-01-23 21:37:55.529  
Test Suite 'MyLibraryTests.xctest' started at 2022-01-23 21:37:55.529  
Test Suite 'MyLibraryTests' started at 2022-01-23 21:37:55.530  
Test Case '-[MyLibraryTests.MyLibraryTests testIsLuckyBecauseWeAlreadyHaveLuckyNumber]' started.  
One  
Three  
Two  
Four  
Test Case '-[MyLibraryTests.MyLibraryTests testIsLuckyBecauseWeAlreadyHaveLuckyNumber]' passed (12.737 seconds).  
Test Case '-[MyLibraryTests.MyLibraryTests testIsLuckyBecauseWeatherHasAnEight]' started.  
Five  
Six  
Seven  
Eight  
Test Case '-[MyLibraryTests.MyLibraryTests testIsLuckyBecauseWeatherHasAnEight]' passed (28.943 seconds).  
Test Case '-[MyLibraryTests.MyLibraryTests testIsNotLucky]' started.  
Test Case '-[MyLibraryTests.MyLibraryTests testIsNotLucky]' passed (0.001 seconds).  
Test Case '-[MyLibraryTests.MyLibraryTests testIsNotLuckyBecauseServiceCallFails]' started.  
explicitlyCancelled  
Test Case '-[MyLibraryTests.MyLibraryTests testIsNotLuckyBecauseServiceCallFails]' passed (0.001 seconds).  
Test Suite 'MyLibraryTests' passed at 2022-01-23 21:38:37.258.  
 Executed 4 tests, with 0 failures (0 unexpected) in 41.681 (41.705) seconds  
Test Suite 'MyLibraryTests.xctest' passed at 2022-01-23 21:38:37.247.  
 Executed 4 tests, with 0 failures (0 unexpected) in 41.681 (41.717) seconds  
Test Suite 'Selected tests' passed at 2022-01-23 21:38:37.258.  
 Executed 4 tests, with 0 failures (0 unexpected) in 41.681 (41.729) seconds  
Program ended with exit code: 0

## 5. <https://rajalingamgariassignment3.s3.amazonaws.com/weather.json>

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various navigation options like 'Buckets', 'Access Points', 'Object Lambda Access Points', etc. The main area displays the contents of the 'rajalingamgariassignment3' bucket. A table lists one object: 'weather.json'. The table includes columns for Name, Type, Last modified, Size, and Storage class. The 'weather.json' file is a json file of size 670.0 B and Standard storage class. Below the table is a search bar labeled 'Find objects by prefix'.

```
{
  "coord": {
    "lon": -123.262,
    "lat": 44.5646
  },
  "weather": [
    {
      "id": 741,
      "main": "Fog",
      "description": "fog",
      "icon": "50n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 275.76,
    "feels_like": 275.76,
    "temp_min": 274.04,
    "temp_max": 281.59,
    "pressure": 1027,
    "humidity": 51
  },
  "visibility": 402,
  "wind": {
    "speed": 0,
    "deg": 0
  },
  "clouds": {
    "all": 100
  },
  "dt": 1642998744,
  "sys": {
    "type": 1,
    "id": 3727,
    "country": "US",
    "sunrise": 1642952435,
    "sunset": 1642986538
  },
  "timezone": -28800,
  "id": 5720327,
  "name": "Corvallis",
  "cod": 200
}
```