Specification Document For The Project

“FileConsoleApp”

PG FSD Implement OOPS using JAVA with Data Structures and Beyond

Simplilearn / CalTech

MARCH 2022 COHORT

Prepared By

Bakau Onafuwa

May 12, 2022

|  |  |
| --- | --- |
| Name | Bakau Onafuwa |
| Email | Bakau.onafuwa@softgineer.com |
| GitHub Repository | [git@github.com:homozapien/fsdconsoleproject.git](mailto:git@github.com:homozapien/fsdconsoleproject.git)  <https://github.com/homozapien/fsdconsoleproject.git> |
| Project Management | Agile |
| Agile Methodologies | SCRUM |

**Objective**

The main objective of this assessment project is to develop a prototype, Interactive Console based Application in Java to validate our understating and conformance with Software Development Lifecycle.

Core development concepts like exception handling, code reviews, code refactoring, versioning, and SCRUM framework are equally showcased.

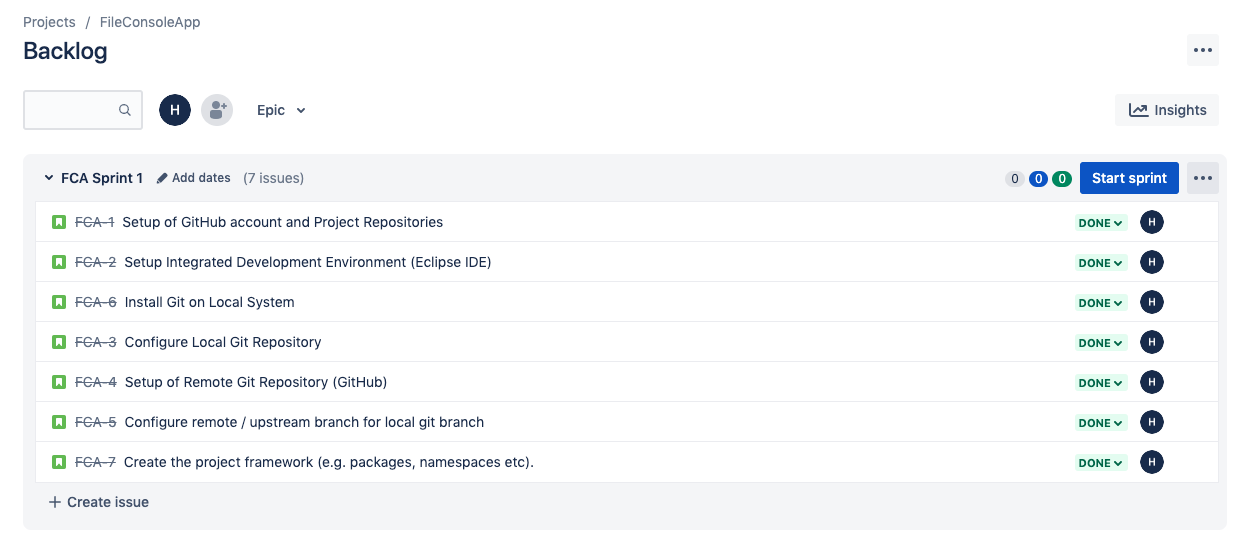
**Assumptions**

1. It is assumed that the reviewer of the source codes, the compiler, the tester and/or runner of this application is conversant with the Java Technologies and possess some level of programing background.
2. This application was developed in Eclipse IDE and based on JDK 1.8; it is assumed that the same environment will be available for the review and testing of this application.
3. It is assumed that the code reviewer and the tester of this application are familiar with Git and can clone the Github repository for this project.
4. The development of this application never considered the security context of the underlying OS with regards to the File I/O and thus, the assumption is made that the user context within that will run or test this application has the full administrative privilege to perform File I/O on the underlying OS.

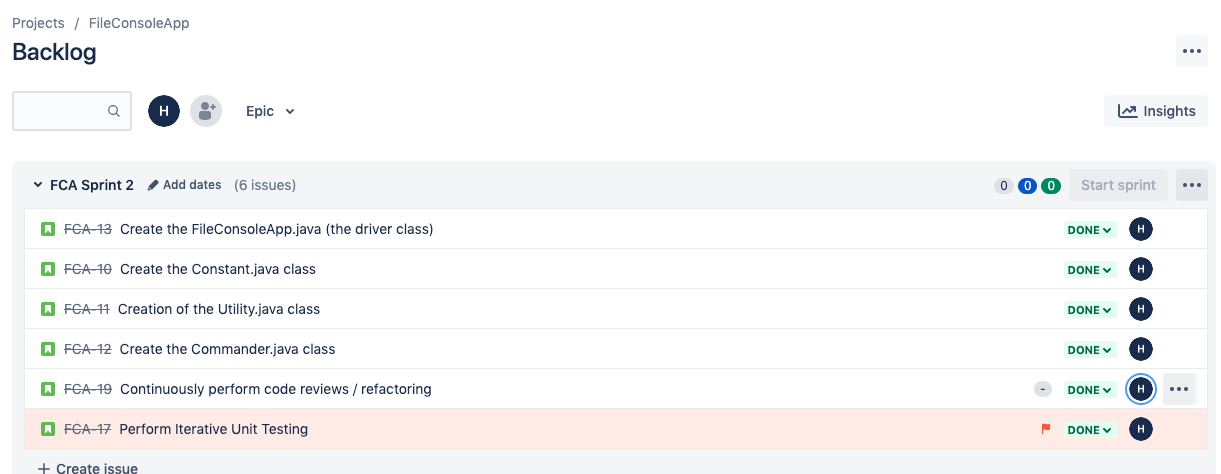
**Project Management**

This application was developed within a period of 1 week however, I had employed the SCRUM framework for an iterative development. The user stories in the Product Backlog are high-level (without task break down for the individual story) and were implemented in four iterations as shown below.

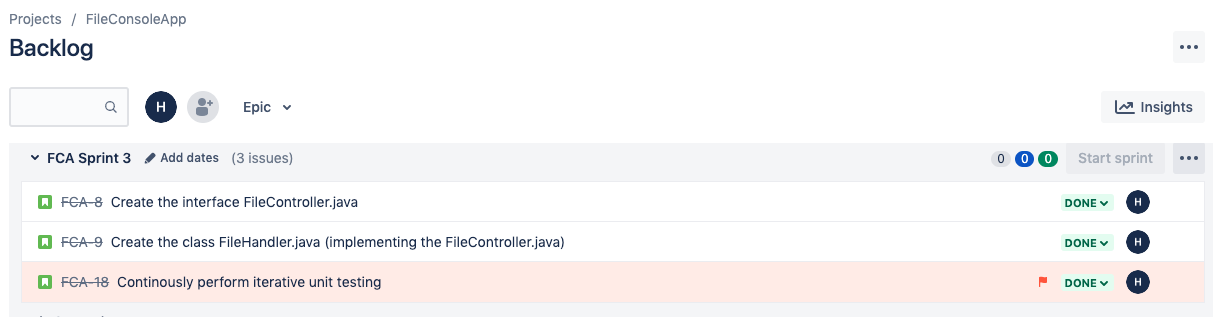
Sprint 1 Goal: Get the environment and infrastructure ready



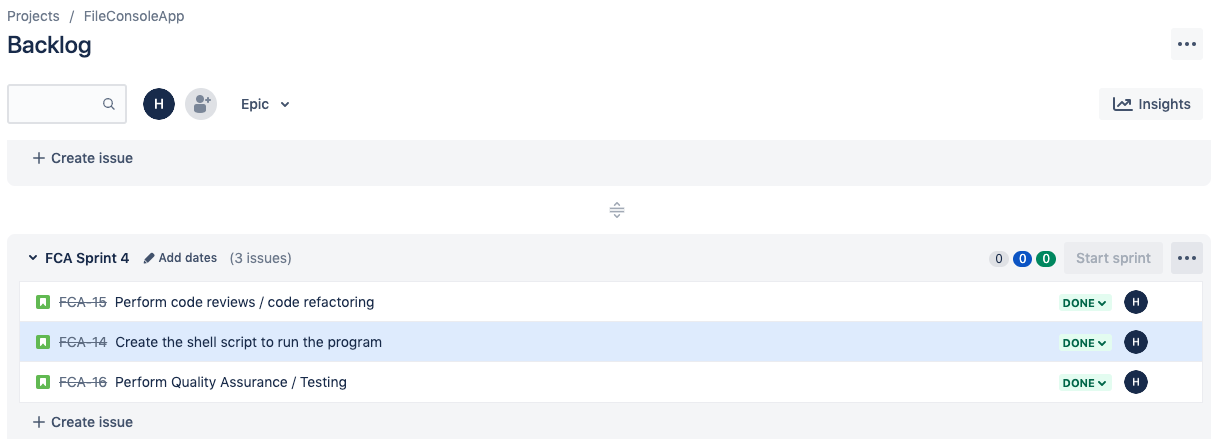
Sprint 2 Goal: Build Application Framework



Sprint 3 Goal: Build the Business Logic



Sprint 4: Improve Easy of Use



**Execution Flow**

This is a console based application and requires inputs from the user to control and direct the flow of execution.

High-Level Flow

1. The runner may pass a command line argument during start-up for a suitable working directory or folder on the system.
2. The application leverages the “user.dir” as the working directory if no argument was supplied in step 1 above.
3. Based on user input, initial set of seed files may be created in the “working directory/foobar”. Up to 6 files may be created at this point.
4. The current state of the working directory is displayed automatically.
5. The runner is presented with a console input request to control operations on the files (if any was created in Step 3 above) in the working directory.
   1. User inputs: ADD to add a file to the working directory
   2. User inputs: DELE to delete a file from the directory if it exists.
   3. User inputs: DISP to display the contents of a file if it exists in the working directory
   4. User inputs: SEAR to search for a file in the working directory
   5. User inputs: RETR to display the current contents of the working directory
   6. User Inputs: EXIT to terminate the running application.

**Data Structure & Algorithms**

The program uses the standard java.util.TreeMap<String, Path> to maintain current state of the working directory where the keys to the Map object are maintained as the “filenames” and corresponding values maintained as the individual java.nio.file.Path objects.

For each physical file in the working directory, an entry will be mainatined in the TreeMap as thus: treeMap.put(“filename”, FilePath).

All mutable operations on the working directory (e.g. Add file or Delete file) will change the current state of the map and the immutable operations (e.g. Search, Retrieve, and CheckExistence) do not need to interrogate the underlying file system but rather query the map object. This approach introduces huge performance gain when compared to directly querying the file system.

**Testing**

To successfully compile and run this program, minimum of JDK 1.8 must be running in the test environment.

To test the program without an IDE, follow these steps below

1. Navigate to the project directory from the command line
2. Compile the source file using the javac command
3. Run the program with the Java command as demonstrated below.

$......fsdconsoleproject> javac -d ./bin -cp ./src ./src/com/simplilearn/fsd/main/FileConsoleApp.java

$......fsdconsoleproject> java -cp ./bin com.simplilearn.fsd.main.FileConsoleApp

