Spark Execution Plans Viewer and Metrics Monitoring UI

Configuration Manual

-by

Rutvik Modi

Swarnaditya Maitra

ENSURE that the main directory consists of

1) .vs
2) backend
3) DB-Project-UI-master: This is the complete UI (HTML, CSS - bootstrap, grid and flexbox, JavaCcript and jQuery)
4) demo: This is the Spark Java project, also consists of the API made using Spring Boot
5) node_modules
6) public
7) src
8) package.json
9) package-lock.json

# Following are the REST APIs that have been integrated:

HADOOP APIs

http://hadoop1.example.com:8088/ws/v1/cluster/apps

http://hadoop1.example.com:8088/ws/v1/cluster/info

http://hadoop1.example.com:8088/ws/v1/cluster/metrics

hadoop1.example.com:8088/ws/v1/cluster/apps/{app-id}

(For example:- hadoop1.example.com:8088/ws/v1/cluster/apps/application_1476912658570_0002)

SPARK APIs

Depending on the number of spark applications currently running the port number may vary from 4040 to 4041, 4042 and so on,...

For example, a spark application running on port number 4042 is given below:

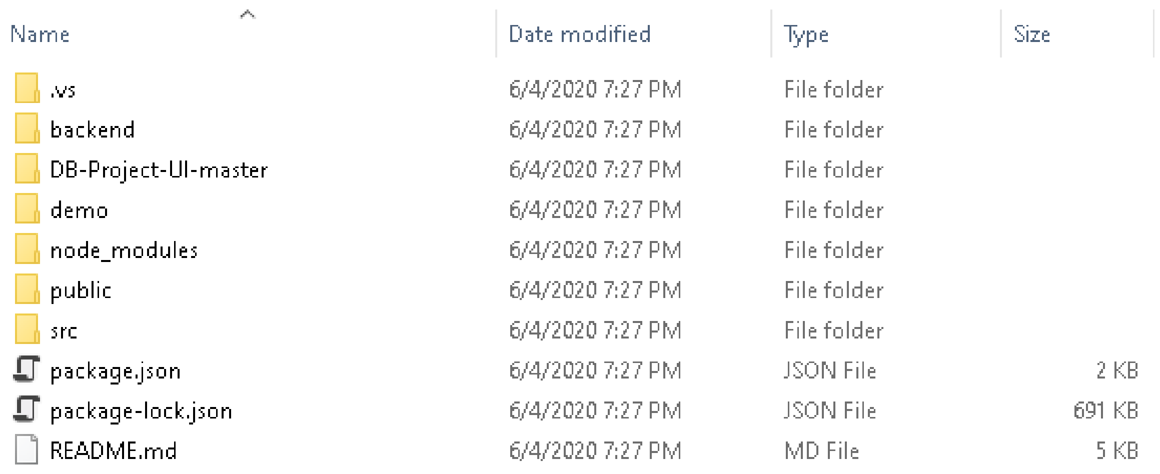
http://hadoop1.example.com:4042/api/v1/applications/

http://hadoop1.example.com:4042/api/v1/applications/local-1590606125695/allexecutors

Instructions:

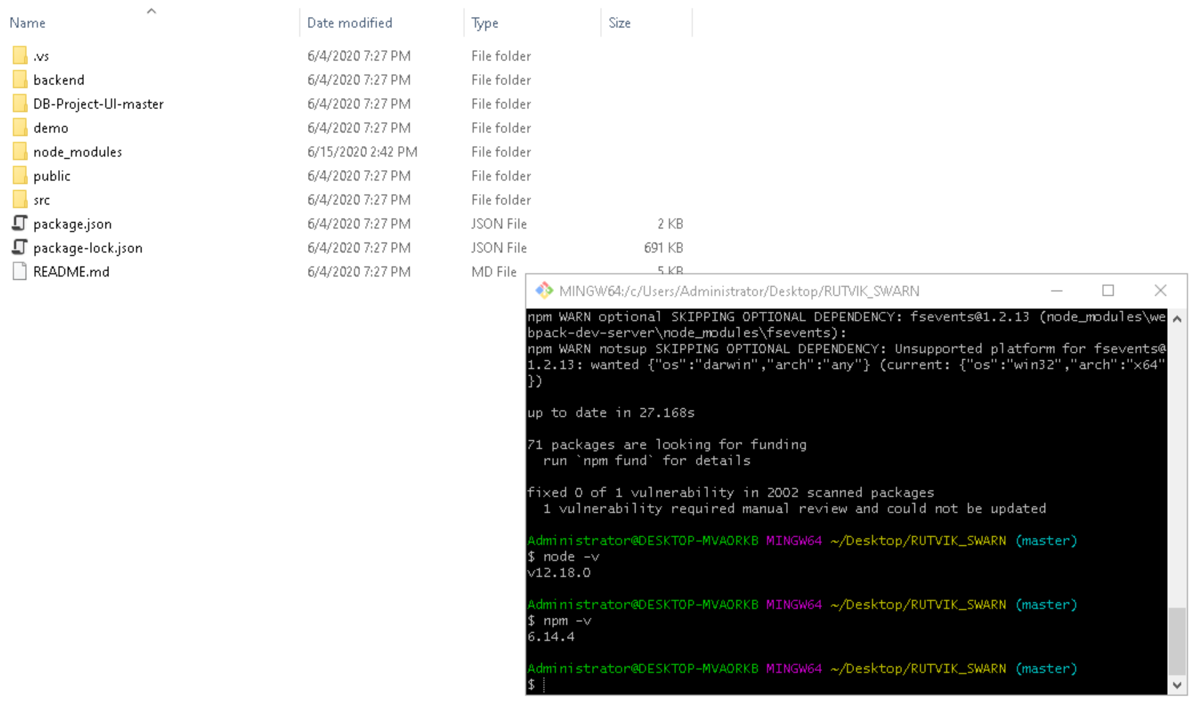
I - React.js form:

1) The main directory structure pertains to the React.js form, the source code for which is present in src folder. package.json will have all the used tech and their versions listed.



Name	Date modified	Type	Size
.vs	6/4/2020 7:27 PM	File folder	
backend	6/4/2020 7:27 PM	File folder	
DB-Project-UI-master	6/4/2020 7:27 PM	File folder	
demo	6/4/2020 7:27 PM	File folder	
node_modules	6/4/2020 7:27 PM	File folder	
public	6/4/2020 7:27 PM	File folder	
src	6/4/2020 7:27 PM	File folder	
package.json	6/4/2020 7:27 PM	JSON File	2 KB
package-lock.json	6/4/2020 7:27 PM	JSON File	691 KB
README.md	6/4/2020 7:27 PM	MD File	5 KB

2) Please ensure that node.js is installed (follow standard procedure for installation of node.js), and the path for node.js is set in the environment variables. It is done automatically on installation of node.js. Open a terminal in this directory, and type
   a. npm install
   b. npm audit fix
   c. All the dependencies and npm packages required for running the react.js form, is present in node_modules, which gets installed with npm install
   d. To check if node is configured, type in the terminal
      i. node –v
      ii. npm –v



Name	Date modified	Type	Size
.vs	6/4/2020 7:27 PM	File folder	
backend	6/4/2020 7:27 PM	File folder	
DB-Project-UI-master	6/4/2020 7:27 PM	File folder	
demo	6/4/2020 7:27 PM	File folder	
node_modules	6/15/2020 2:42 PM	File folder	
public	6/4/2020 7:27 PM	File folder	
src	6/4/2020 7:27 PM	File folder	
package.json	6/4/2020 7:27 PM	JSON File	2 KB
package-lock.json	6/4/2020 7:27 PM	JSON File	691 KB
README.md	6/4/2020 7:27 PM	MD File	5 KB

MINGW64:/c/Users/Administrator/Desktop/RUTVIK_SWARN

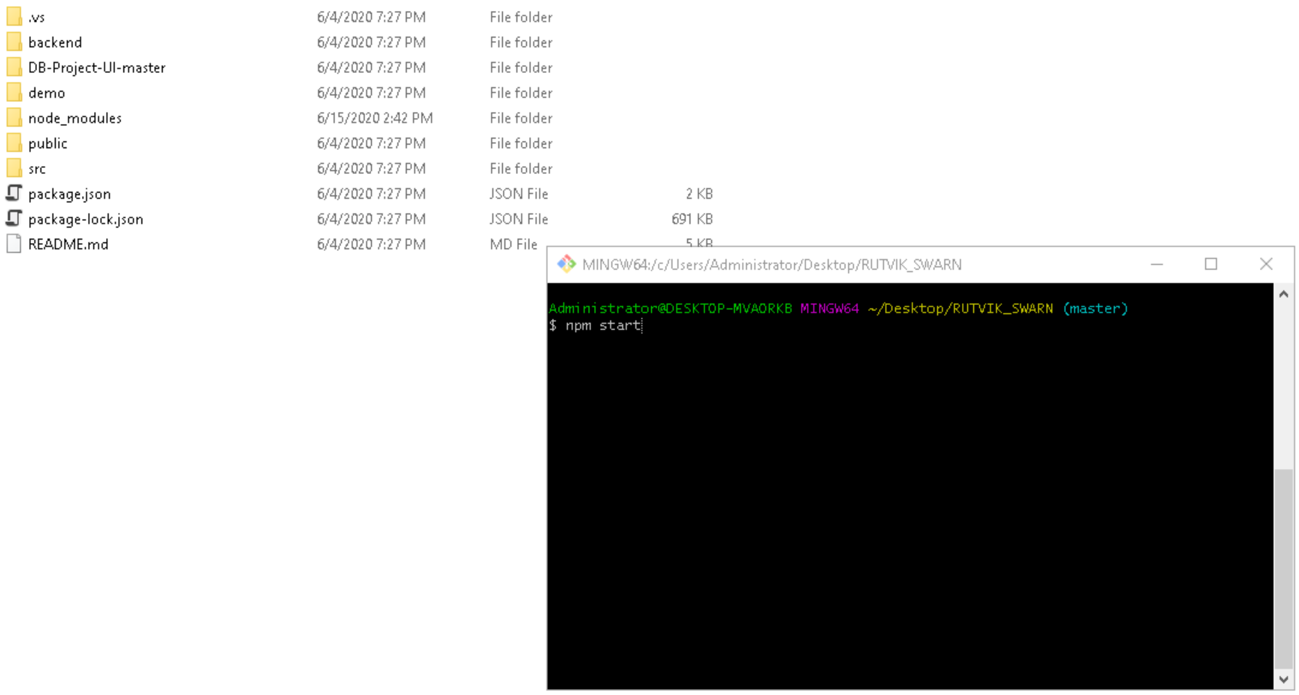
```
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\we
bpack-dev-server\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@
1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"
})

up to date in 27.168s

71 packages are looking for funding
  run `npm fund` for details

fixed 0 of 1 vulnerability in 2002 scanned packages
  1 vulnerability required manual review and could not be updated

Administrator@DESKTOP-MVAORKB MINGW64 ~/Desktop/RUTVIK_SWARN (master)
$ node -v
v12.18.0

Administrator@DESKTOP-MVAORKB MINGW64 ~/Desktop/RUTVIK_SWARN (master)
$ npm -v
6.14.4

Administrator@DESKTOP-MVAORKB MINGW64 ~/Desktop/RUTVIK_SWARN (master)
$
```

   e. node modules contain the react-scripts folder, which has all the scripts for running react.js

f. <Optional> In order to build on openshift, or some other platform, give .bin and react-scripts directory (inside node_modules) all permissions using git update-index --chmod=+x path/to/file

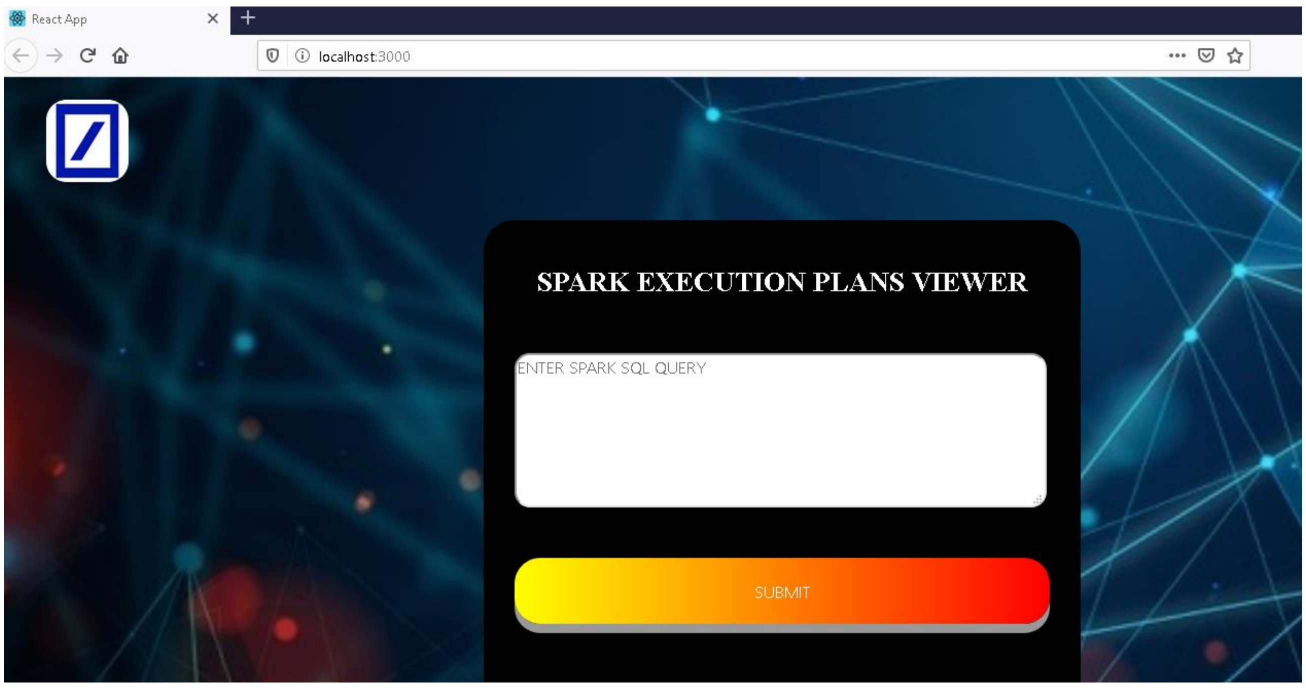
3) src folder contains all the react.js source code. The main react code is in App.js

4) open git bash inside the original directory, when all is configured and type: npm start .....this will host the react form on http://localhost:3000



```
.vs                  6/4/2020 7:27 PM    File folder
backend              6/4/2020 7:27 PM    File folder
DB-Project-UI-master 6/4/2020 7:27 PM    File folder
demo                 6/4/2020 7:27 PM    File folder
node_modules         6/15/2020 2:42 PM   File folder
public               6/4/2020 7:27 PM    File folder
src                  6/4/2020 7:27 PM    File folder
package.json         6/4/2020 7:27 PM    JSON File        2 KB
package-lock.json    6/4/2020 7:27 PM    JSON File      691 KB
README.md            6/4/2020 7:27 PM    MD File          5 KB
```

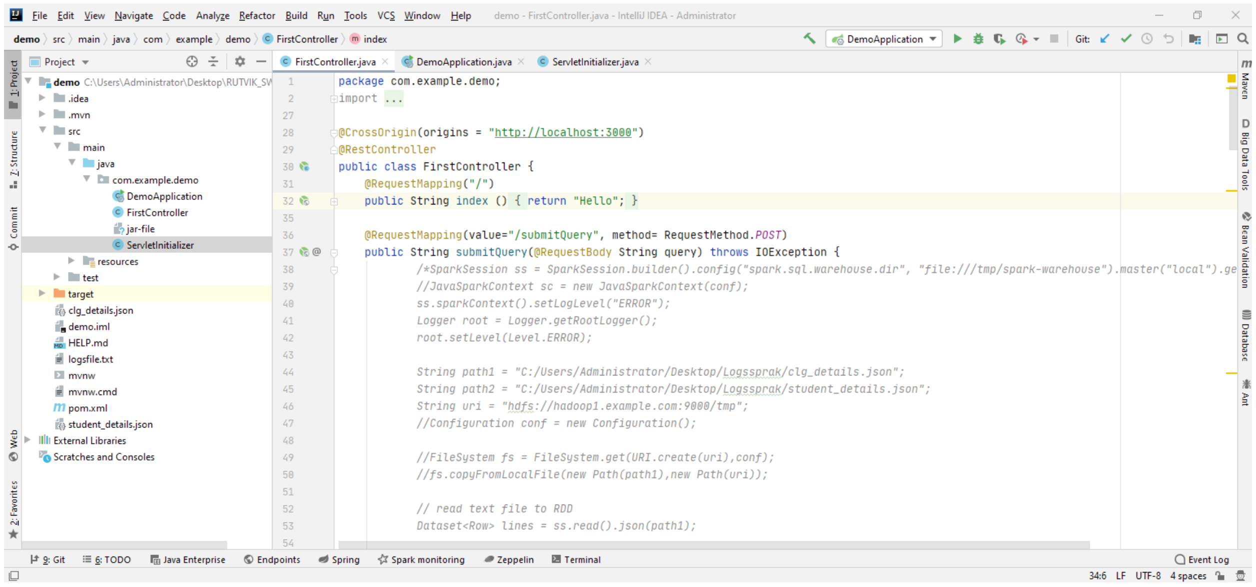
```
MINGW64:/c/Users/Administrator/Desktop/RUTVIK_SWARN

Administrator@DESKTOP-MVAORKB MINGW64 ~/Desktop/RUTVIK_SWARN (master)
$ npm start
```



II - Java Backend

1) demo – a Java Project, present in the original directory, is the major backend component, dealing with all the data processing for user's SQL statements that were entered on the React Form.
   a. NOTE: DO NOT run the project directly from the Run button on the menu-bar.

2) Open up intelliJ, and open the project demo



```
package com.example.demo;
import ...

@CrossOrigin(origins = "http://localhost:3000")
@RestController
public class FirstController {
    @RequestMapping("/")
    public String index () { return "Hello"; }

    @RequestMapping(value="/submitQuery", method= RequestMethod.POST)
    public String submitQuery(@RequestBody String query) throws IOException {
        /*SparkSession ss = SparkSession.builder().config("spark.sql.warehouse.dir", "file:///tmp/spark-warehouse").master("local").ge
        //JavaSparkContext sc = new JavaSparkContext(conf);
        ss.sparkContext().setLogLevel("ERROR");
        Logger root = Logger.getRootLogger();
        root.setLevel(Level.ERROR);

        String path1 = "C:/Users/Administrator/Desktop/Logssprak/clg_details.json";
        String path2 = "C:/Users/Administrator/Desktop/Logssprak/student_details.json";
        String uri = "hdfs://hadoop1.example.com:9000/tmp";
        //Configuration conf = new Configuration();

        //FileSystem fs = FileSystem.get(URI.create(uri),conf);
        //fs.copyFromLocalFile(new Path(path1),new Path(uri));

        // read text file to RDD
        Dataset<Row> lines = ss.read().json(path1);
```

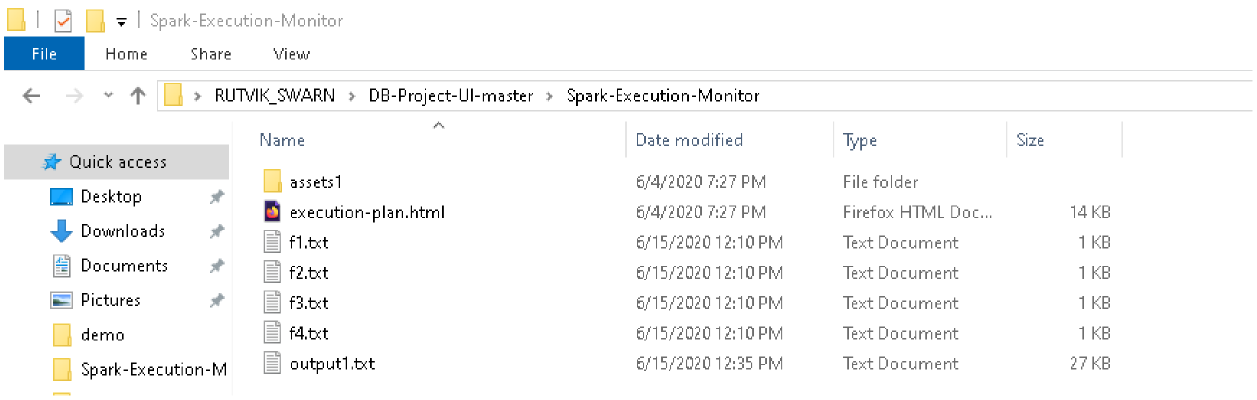
3) All the configuration changes, and importing of all the maven, spark, and spring dependencies has already been done.

4) Ensure that the path of all the files, specified inside the program, is where the files are actually present. This is important. Ensure that the path of the files f1, f2, f3, f4 and output.txt(in DB-Project-UI-master) is the same as that specified in the code. Ensure that files are pasted at these locations, if not already, or change the path to the appropriate locations of the files. (In the pictures below, the repo is named RUTVIK_SWARN, which can of course vary. Make sure to reflect the changes in the code)

```
String path1 = "C:/Users/Administrator/Desktop/RUTVIK_SWARN/demo/clg_details.json";
String path2 = "C:/Users/Administrator/Desktop/RUTVIK_SWARN/demo/student_details.json";
```

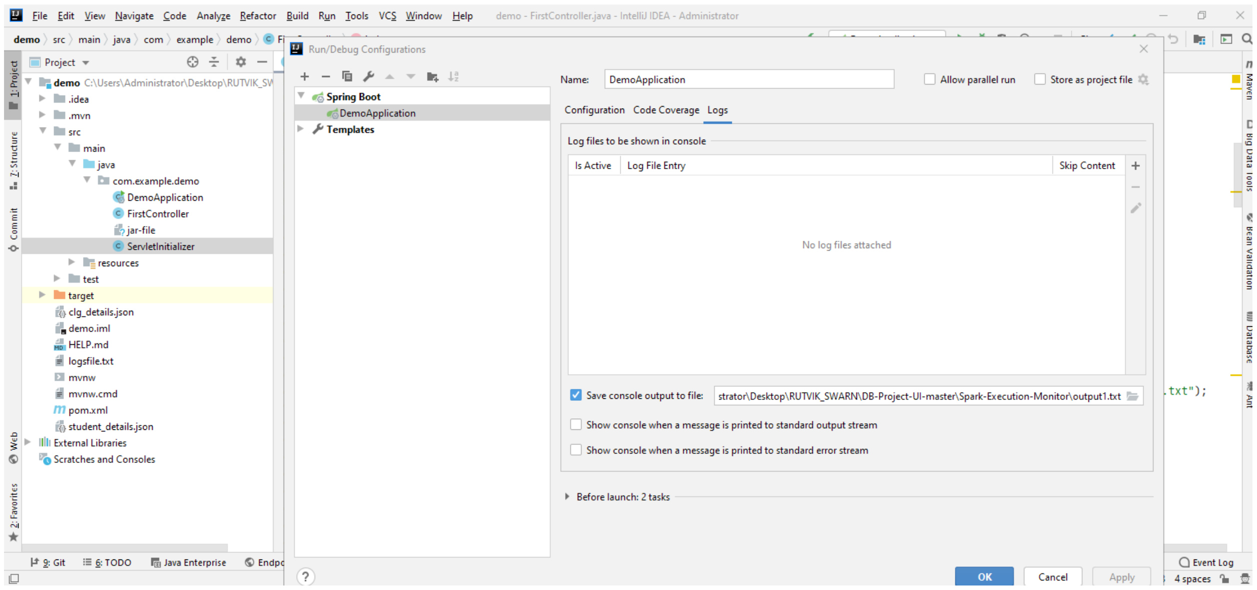
```
String f1 = "C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Spark-Execution-Monitor/f1.txt";
String f2 = "C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Spark-Execution-Monitor/f2.txt";
String f3 = "C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Spark-Execution-Monitor/f3.txt";
String f4 = "C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Spark-Execution-Monitor/f4.txt";
```

```
( fileName: "C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Spark-Execution-Monitor/output1.txt");
```



Name	Date modified	Type	Size
assets1	6/4/2020 7:27 PM	File folder	
execution-plan.html	6/4/2020 7:27 PM	Firefox HTML Doc...	14 KB
f1.txt	6/15/2020 12:10 PM	Text Document	1 KB
f2.txt	6/15/2020 12:10 PM	Text Document	1 KB
f3.txt	6/15/2020 12:10 PM	Text Document	1 KB
f4.txt	6/15/2020 12:10 PM	Text Document	1 KB
output1.txt	6/15/2020 12:35 PM	Text Document	27 KB

5) Click on Run button -> Edit Configuration -> logs -> save console output to file. Enter the path of the output file here (ENSURE that the path is the same as DB-Project-UI-master). This is necessary to route the console output stream to the required output file.



Name: DemoApplication

Allow parallel run ☐ Store as project file ☐

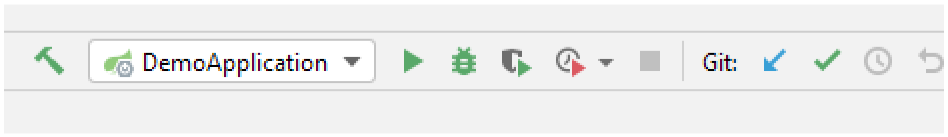
Log files to be shown in console

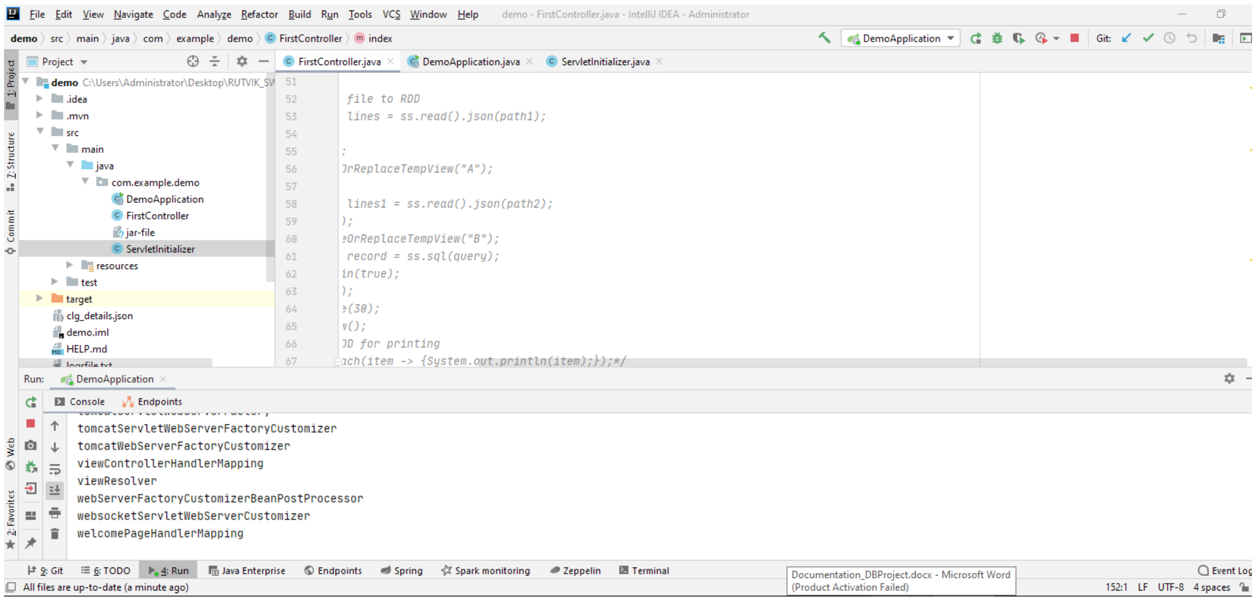
☑ Save console output to file: strator\Desktop\RUTVIK_SWARN\DB-Project-UI-master\Spark-Execution-Monitor\output1.txt

☐ Show console when a message is printed to standard output stream

☐ Show console when a message is printed to standard error stream

6) To start up the project, on the bar below the title bar, at the right-hand side, beside "demo-application", there would be a green play button. Click it. The spring application should run. Check the console output





```
 file to RDD
 lines = ss.read().json(path1);

;
OrReplaceTempView("A");

 lines1 = ss.read().json(path2);
);
eOrReplaceTempView("B");
 record = ss.sql(query);
in(true);
);
e(30);
v();
DD for printing
ach(item -> {System.out.println(item);});*/
```

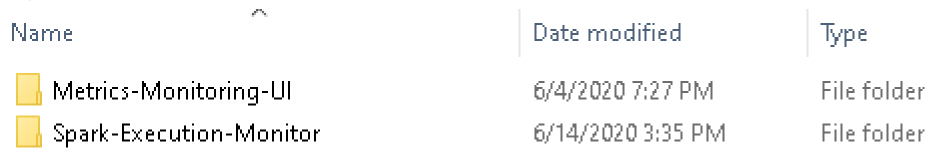
```
tomcatServletWebServerFactoryCustomizer
tomcatWebServerFactoryCustomizer
viewControllerHandlerMapping
viewResolver
webServerFactoryCustomizerBeanPostProcessor
websocketServletWebServerCustomizer
welcomePageHandlerMapping
```

All files are up-to-date (a minute ago)

7) Now, this application runs infinitely, and goes active whenever the user submits statements in the React Form.

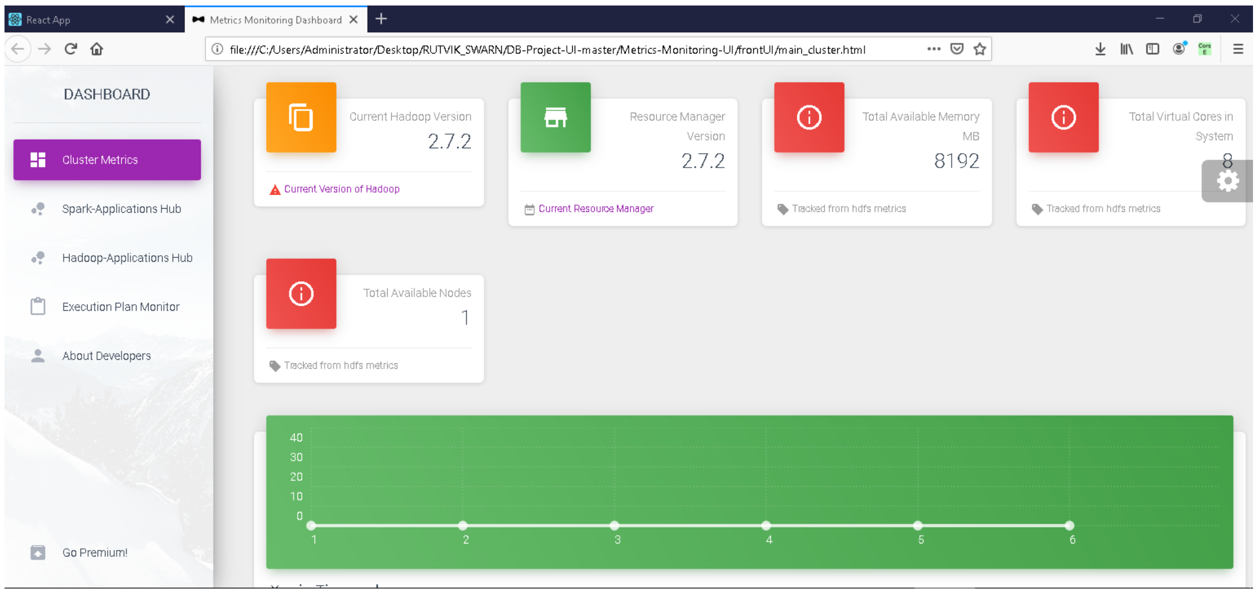
III - DB-Project-UI-master

1) There are two parts:
   a. Metrics-Monitoring-UI: Contains the Metrics UI that displays all the cluster metrics (for the cluster as a whole), as well has info for the individual apps running on Hadoop (general metrics). In addition, it displays all running spark applications, and the metrics details of each individual application
   b. Spark Execution Monitor: Displays unresolved logical plan, resolved logical plan, optimized logical plan and physical implementation plan, corresponding to the SQL query entered by the user. The plans are stored in f1-f4.txt and console output of project is redirected to output.txt (truncation)



Name	Date modified	Type
Metrics-Monitoring-UI	6/4/2020 7:27 PM	File folder
Spark-Execution-Monitor	6/14/2020 3:35 PM	File folder

2) Inside Metrics-Monitoring-UI
   a. The frontUI contains all the html files. Open main_cluster.html



file:///C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Metrics-Monitoring-UI/frontUI/main_cluster.html

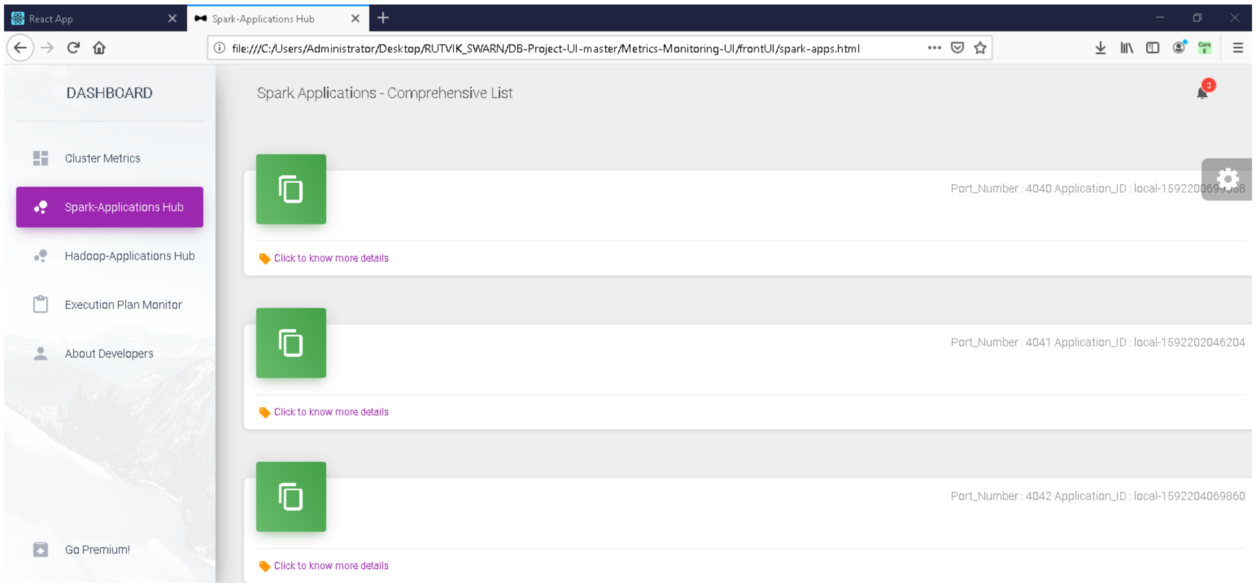
Current Hadoop Version 2.7.2 — Current Version of Hadoop
Resource Manager Version 2.7.2 — Current Resource Manager
Total Available Memory MB 8192 — Tracked from hdfs metrics
Total Virtual Cores in System 8 — Tracked from hdfs metrics
Total Available Nodes 1 — Tracked from hdfs metrics

   b. The Dashboard contains
      i. Cluster Metrics (main_cluster.html)
      ii. Spark-Applications Hub (spark-apps.html)
      iii. Hadoop-Applications Hub (hadoop-apps.html)
      iv. Execution Plan Monitor
      v. About Developers (developers.html)
      vi. Go Premium!

   c. Note that the CSS and JS files for the webpages in all of DB-Project-UI-master, have the same names as corresponding HTML files. The implementation of functionalities can be seen there.

d. Cluster Metrics: Displays information about the hadoop cluster as a whole. Cards and graphs have been used to showcase the data.

e. Spark-Applications Hub: Is a dynamically updating page, that shows all the currently running spark applications. User can click on any to see the details.



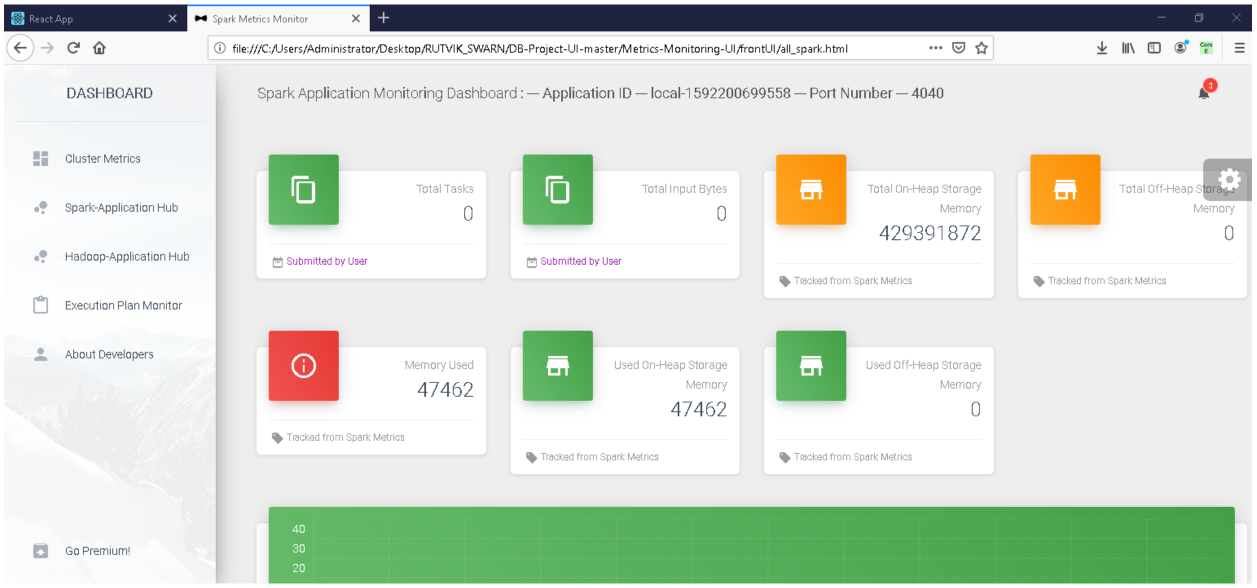
Spark Applications - Comprehensive List

Port_Number : 4040 Application_ID : local-159220069[illegible]8
Click to know more details

Port_Number : 4041 Application_ID : local-1592202046204
Click to know more details

Port_Number : 4042 Application_ID : local-1592204069860
Click to know more details

Below is the dynamically generated page (all_spark.html), on clicking the 1st link:



Spark Application Monitoring Dashboard : — Application ID — local-1592200699558 — Port Number — 4040

Total Tasks
0
Submitted by User

Total Input Bytes
0
Submitted by User

Total On-Heap Storage Memory
429391872
Tracked from Spark Metrics

Total Off-Heap Storage Memory
0
Tracked from Spark Metrics

Memory Used
47462
Tracked from Spark Metrics

Used On-Heap Storage Memory
47462
Tracked from Spark Metrics

Used Off-Heap Storage Memory
0
Tracked from Spark Metrics

f. Hadoop-applications Hub works exactly the same way as Spark Applications Hub (hadoop-apps.html and hadoop_general.html). Hence, for brevity, we aren't repeating the instructions for the same.

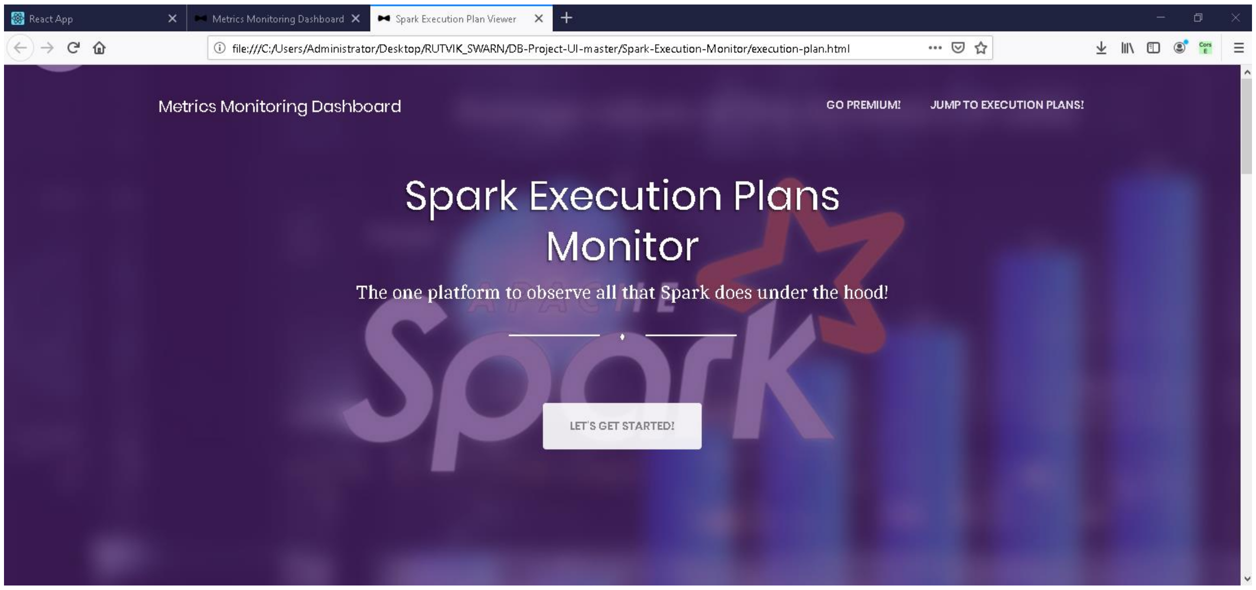
g. About Developers: Simply displays basic info about developers.

h. Premium feature hasn't been developed yet.

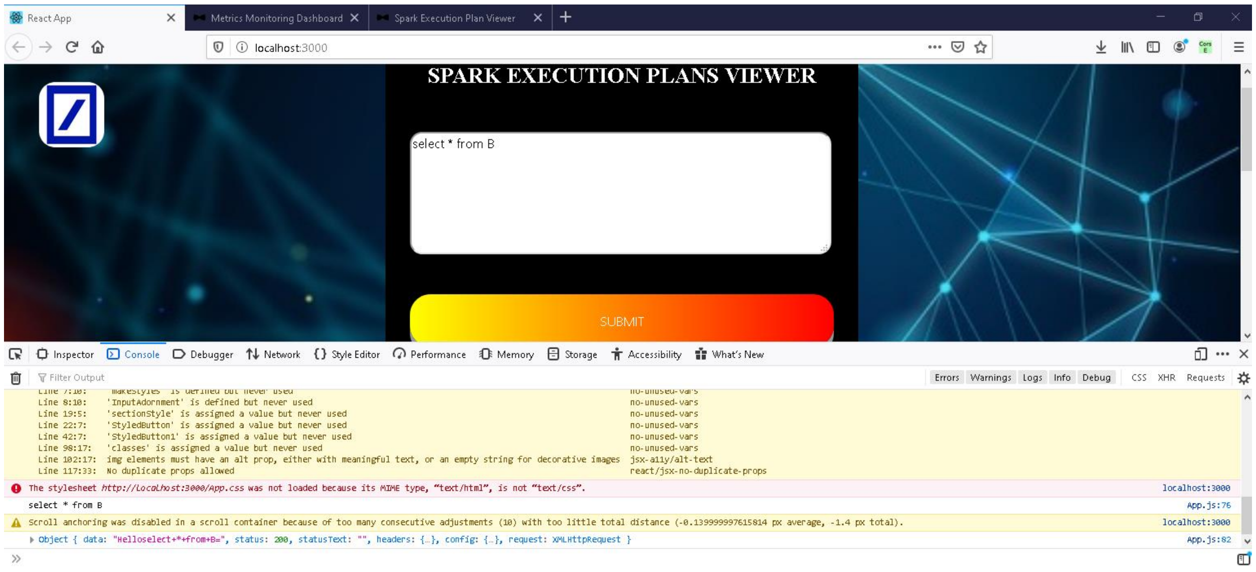
i. All the css, js and image files are present in assets folder.

3) Inside Spark Execution Monitor:

a. Open execution-plan.html. If the java sub-project (demo) is correctly configured, the execution plans corresponding to user's SQL query (entered in react.js form), will show up in the Execution plans section on the page.


React App
Metrics Monitoring Dashboard
Spark Execution Plan Viewer
file:///C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Spark-Execution-Monitor/execution-plan.html
Metrics Monitoring Dashboard
GO PREMIUM!
JUMP TO EXECUTION PLANS!
Spark Execution Plans Monitor
The one platform to observe all that Spark does under the hood!
LET'S GET STARTED!

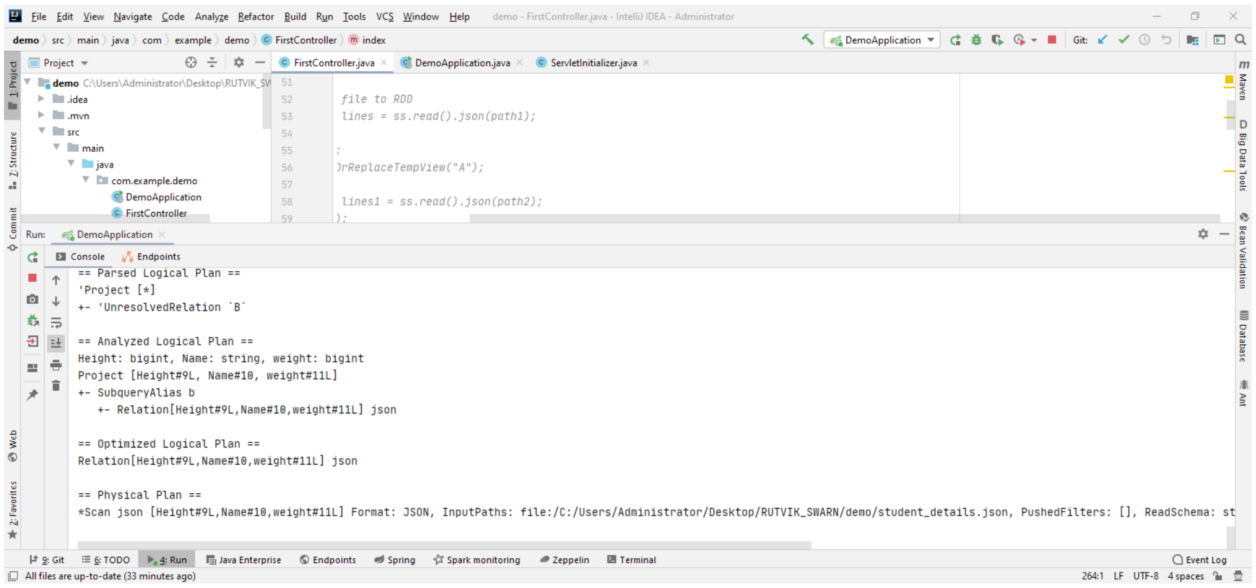
b. Sequence of Working (with a simple SQL query):



```
  Line 7:10:    'makeStyles' is defined but never used                                    no-unused-vars
  Line 8:10:    'InputAdornment' is defined but never used                                no-unused-vars
  Line 19:5:    'sectionStyle' is assigned a value but never used                         no-unused-vars
  Line 22:7:    'StyledButton' is assigned a value but never used                         no-unused-vars
  Line 42:7:    'StyledButton1' is assigned a value but never used                        no-unused-vars
  Line 98:17:   'classes' is assigned a value but never used                              no-unused-vars
  Line 102:17:  img elements must have an alt prop, either with meaningful text, or an empty string for decorative images  jsx-a11y/alt-text
  Line 117:33:  No duplicate props allowed                                                react/jsx-no-duplicate-props
The stylesheet http://localhost:3000/App.css was not loaded because its MIME type, "text/html", is not "text/css".   localhost:3000
select * from B                                                                                                      App.js:76
Scroll anchoring was disabled in a scroll container because of too many consecutive adjustments (10) with too little total distance (-0.139999997615814 px average, -1.4 px total).   localhost:3000
▸ Object { data: "Helloselect+*+from+B=", status: 200, statusText: "", headers: {…}, config: {…}, request: XMLHttpRequest }   App.js:82
```

*i) Select * from B has been successfully submitted above.*

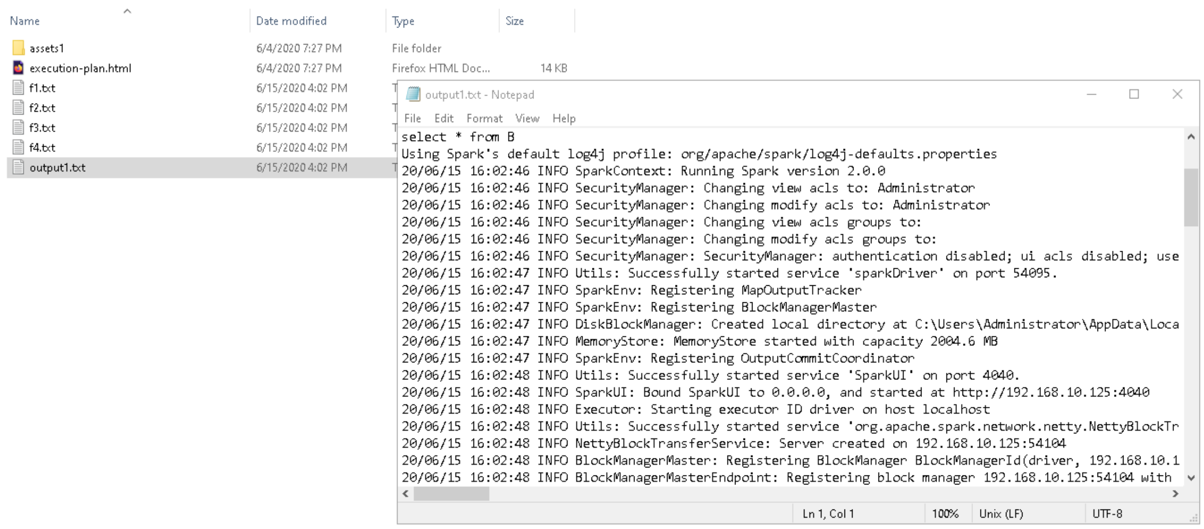
ii) The spring Boot API in the java backend, has received it using the dispatcher servlet (MVC) *iii) Since the program is running in infinite loop, it has passed the query to the Spark Engine and console output can be used to check that.*



```
file to RDD
lines = ss.read().json(path1);

;
)rReplaceTempView("A");

lines1 = ss.read().json(path2);
);
```

```
== Parsed Logical Plan ==
'Project [*]
+- 'UnresolvedRelation `B`

== Analyzed Logical Plan ==
Height: bigint, Name: string, weight: bigint
Project [Height#9L, Name#10, weight#11L]
+- SubqueryAlias b
   +- Relation[Height#9L,Name#10,weight#11L] json

== Optimized Logical Plan ==
Relation[Height#9L,Name#10,weight#11L] json

== Physical Plan ==
*Scan json [Height#9L,Name#10,weight#11L] Format: JSON, InputPaths: file:/C:/Users/Administrator/Desktop/RUTVIK_SWARN/demo/student_details.json, PushedFilters: [], ReadSchema: st
```

iv) We can also check in Spark-Execution-Monitor, if the output has been redirected:

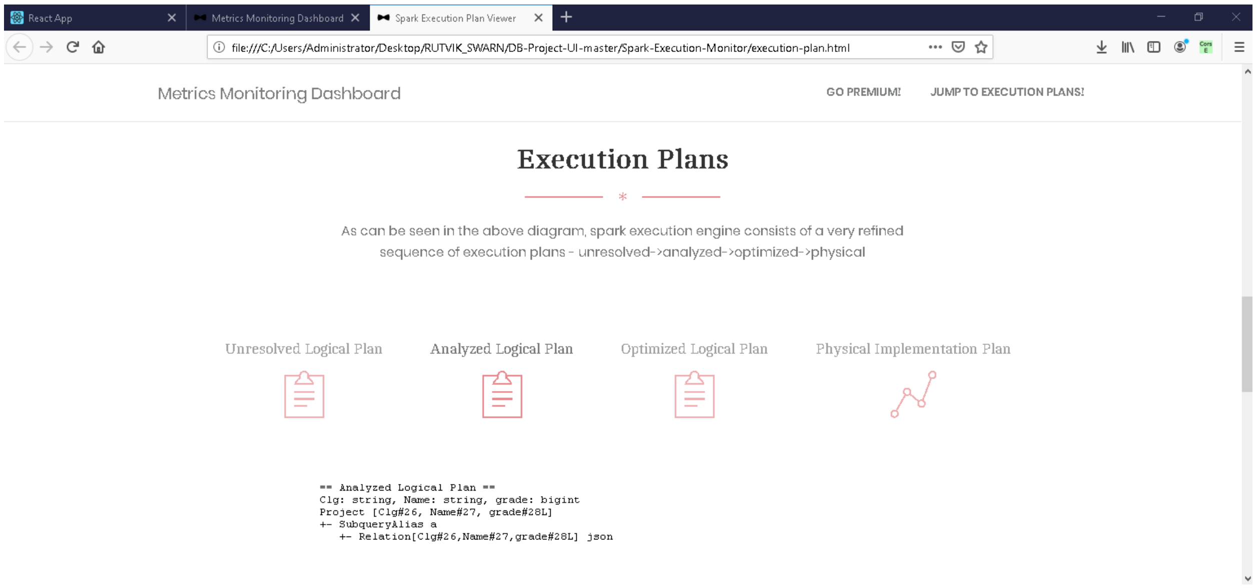


Name	Date modified	Type	Size
assets1	6/4/2020 7:27 PM	File folder	
execution-plan.html	6/4/2020 7:27 PM	Firefox HTML Doc...	14 KB
f1.txt	6/15/2020 4:02 PM	T	
f2.txt	6/15/2020 4:02 PM	T	
f3.txt	6/15/2020 4:02 PM	T	
f4.txt	6/15/2020 4:02 PM	T	
output1.txt	6/15/2020 4:02 PM	T	

```
select * from B
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
20/06/15 16:02:46 INFO SparkContext: Running Spark version 2.0.0
20/06/15 16:02:46 INFO SecurityManager: Changing view acls to: Administrator
20/06/15 16:02:46 INFO SecurityManager: Changing modify acls to: Administrator
20/06/15 16:02:46 INFO SecurityManager: Changing view acls groups to:
20/06/15 16:02:46 INFO SecurityManager: Changing modify acls groups to:
20/06/15 16:02:46 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; use
20/06/15 16:02:47 INFO Utils: Successfully started service 'sparkDriver' on port 54095.
20/06/15 16:02:47 INFO SparkEnv: Registering MapOutputTracker
20/06/15 16:02:47 INFO SparkEnv: Registering BlockManagerMaster
20/06/15 16:02:47 INFO DiskBlockManager: Created local directory at C:\Users\Administrator\AppData\Loca
20/06/15 16:02:47 INFO MemoryStore: MemoryStore started with capacity 2004.6 MB
20/06/15 16:02:47 INFO SparkEnv: Registering OutputCommitCoordinator
20/06/15 16:02:48 INFO Utils: Successfully started service 'SparkUI' on port 4040.
20/06/15 16:02:48 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.10.125:4040
20/06/15 16:02:48 INFO Executor: Starting executor ID driver on host localhost
20/06/15 16:02:48 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTr
20/06/15 16:02:48 INFO NettyBlockTransferService: Server created on 192.168.10.125:54104
20/06/15 16:02:48 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.10.1
20/06/15 16:02:48 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.10.125:54104 with
```

v) The Java program (later part of FirstController.java) processes this file and generates the contents for f1.txt, f2.txt, f3.txt and f4.txt.

vi) The files are linked to the spark execution plans viewer (execution-plan.html). Hence, UI should work perfectly, as can be seen below:


React App
Metrics Monitoring Dashboard
Spark Execution Plan Viewer
file:///C:/Users/Administrator/Desktop/RUTVIK_SWARN/DB-Project-UI-master/Spark-Execution-Monitor/execution-plan.html
Metrics Monitoring Dashboard
GO PREMIUM!
JUMP TO EXECUTION PLANS!

Execution Plans

As can be seen in the above diagram, spark execution engine consists of a very refined sequence of execution plans - unresolved->analyzed->optimized->physical

Unresolved Logical Plan
Analyzed Logical Plan
Optimized Logical Plan
Physical Implementation Plan

```
== Analyzed Logical Plan ==
Clg: string, Name: string, grade: bigint
Project [Clg#26, Name#27, grade#28L]
+- SubqueryAlias a
   +- Relation[Clg#26,Name#27,grade#28L] json
```

c. Assets1 has all the css, js and image files, with same names as corresponding HTML files.

---THE END---