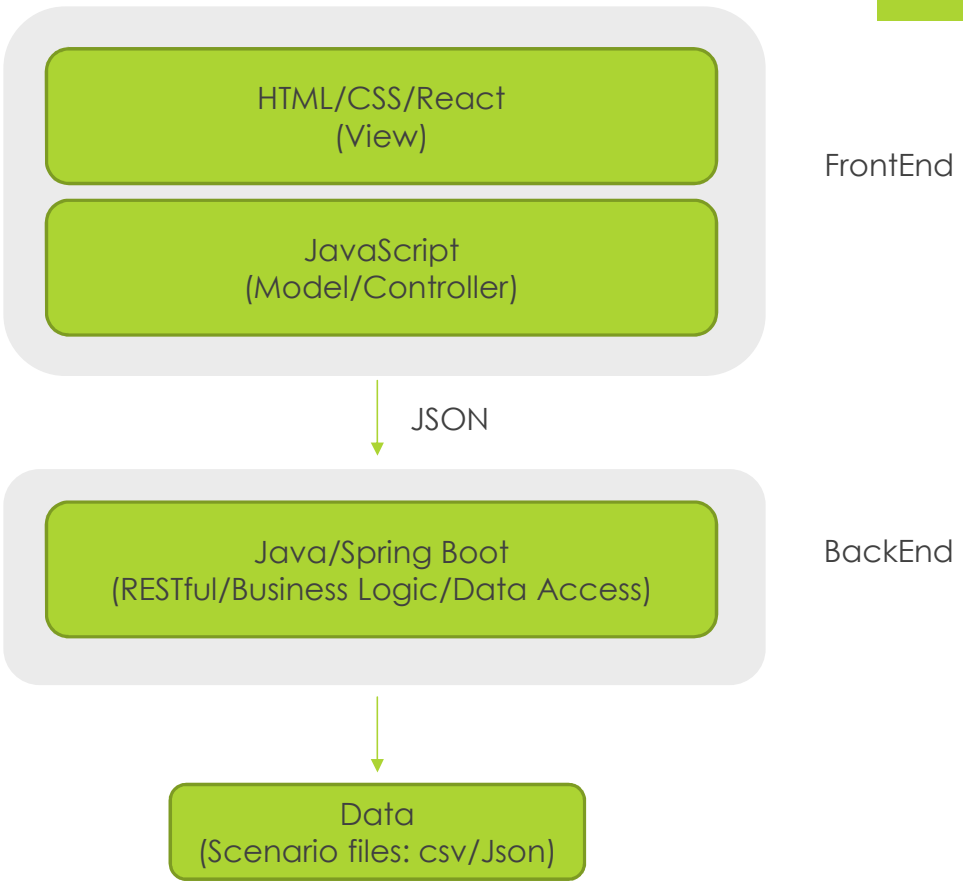
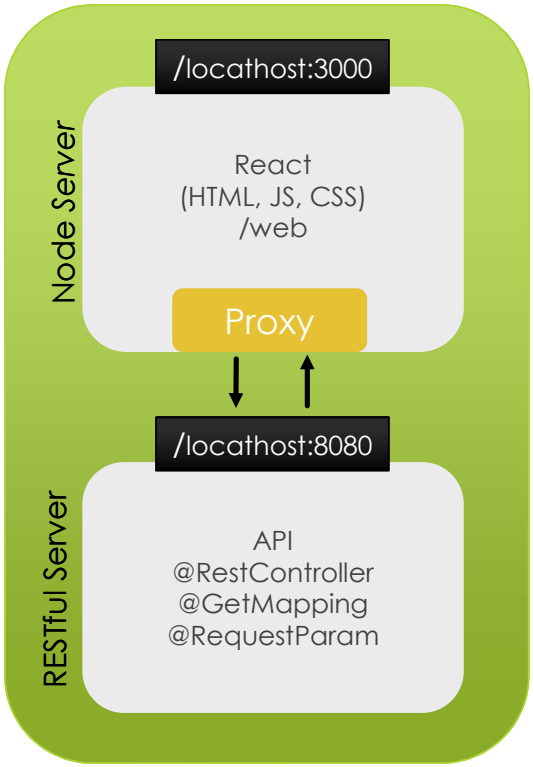
System Design Document

CS6310-A6 GROUP10

System Architecture

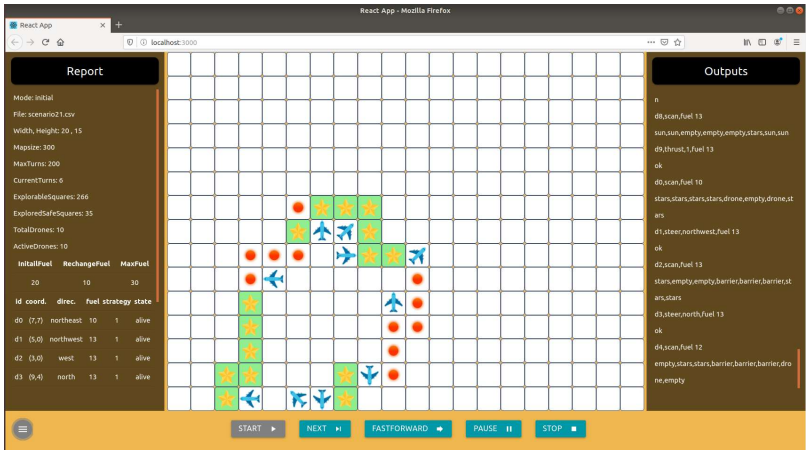


Development



npm start

Update



Define RESTful API

Client

GET:

1. /starSearch/?filename=filename&mode=mode
2. /files?filename=fileName
3. /getStrat
4. /nextAction
5. /nextActionByUser/?action=action&param=param
6. /stop

RESPONSE :

1. {"width","height", "maxTurns", "turns", "fuels", "exploreableSquares", "squares", "safeSquares", "totalDrones", "drones", "outputs"}
2. {"filesToInitial", "filesToResume"}
3. "strat" (strategy)
4. {"turns", "squares", "safeSquares", "drones", "outputs"}
5. {"turns", "squares", "safeSquares", "drones", "outputs"}
6. {"finalReport"}

Web Service

Step-by-step Set up

- Download virtual machine

  Team_10.ova: https://drive.google.com/open?id=16u1_TMGzOWvij-xwguOu52lv8VLBoa11

  (Md5sum: c11ffe3f4b0105c262471c521257c008 Password: group10)
- Download and unzip **source_code.zip** from our group submission and store it in VM
- Open terminal and go to **/source_code**
- Run the command below to build and run file in the current directory: **gradle run**
- Build and run successfully if you see the information below

```
INFO Tomcat started on port(s): 8080 (http)
```

```
<==========----> 75% EXECUTING [3m 26s]
> :run
```

Two Ways to Start GUI

Once backend build and run successfully, there are two ways to start GUI:

- Production Mode

- Open the browser (Firefox), enter **http://127.0.0.1:8080**

- Development mode (OR you can start it with development mode)

- Open Visual Studio Code, Click **File -> Add folder to Workspace** and select the folder **source_code**
- Click **Terminal -> New Terminal, cd web** to go to the folder **/web**
- To Install the dependencies in the local folder node_modules, run the command (once): **npm install**
- To run server file and app in the development mode, use the command: **npm start**
- The browser(Firefox) will open automatically with an address **http://127.0.0.1:3000**

Fuel & Energy Impacts

"Recharge" action is implemented as part of new functionality, which refuels the drones. Initial fuel, recharge fuel and max fuel are read from each scenario files. Drones will automatically take recharge action if their fuels are less than 3. Drones will carry a number of units of fuel but not exceeding the maximum number of fuels in the beginning. Each action costs a different unit of fuel, for example, Pass and Recharge cost 0 fuel, Steer and Scan cost 1 fuel, and thrust costs 1,2,3 fuel based on the steps the drones actually thrust.

	Initial fuel	Recharge fuel	Max Fuel	Cost Fuel				
				Pass	Recharge	Steer	Scan	Thrust
Drone	Read from scenario files	Read from scenario files	Read from scenario files	0	0	1	1	thrust costs 1,2,3 fuel based on the steps

Fuel Test

- To regenerate star_search.jar (already generated under source_code), run the command under source_code :

  gradle shadowjar

- To save output to a scenario_results file, run the command:

  java -cp star_search.jar simulator.Main scenario<N>.csv > scenario<N>_results.csv

- Test one file as following (only test scenario files 2) :

  java -jar fuel_test.jar 2 2

- Test mulitple files as following (only test scenario files 3 to 5) :

  java -jar fuel_test.jar 3 5

- Or you can test all scenario files and save it to report.txt, run the command:

  ./test.sh > report.log

References

- Building a RESTful Web Service: https://spring.io/guides/gs/rest-service/
- Visual Studio Code: https://code.visualstudio.com/
- Create a New React App: https://reactjs.org/docs/create-a-new-react-app.html
- Gradle User Manual: https://docs.gradle.org/current/userguide/userguide.html
- Materialize: https://materializecss.com/