**Birla Institute of Technology and Science, Pilani**



Introduction to DEVOPS (S1-21\_SESAPZG515)

Assignment – Part A

Topic: DevOps Live Project

|  |  |  |
| --- | --- | --- |
| Sr. No. | Name | Bits ID |
| 1 | Allen S | 2020HS70029 |
| 2 | Sugata Kar | 2020HS70001 |
| 3 | Vasundhara | 2020HS70021 |
| 4 | Rajdeep Biswas |  |
| 5 | Sujata Devi |  |
| 6 | Nancy | 2020HS70053 |

Submitted by: Team 8

**Project Description:**

**Project Title**: Amusement Park

**Description:**

We have done this project as a part of DevOps assignment. Various DevOps tools are used in this project as Git, GitHub, Maven, SonarQube and Selenium.

It is a simple microservice, where we have implemented a form to get the details of the ride. We can add rides and can see the rides also.  
we have planned to add another microservice as food/beverages in part 2 as an extension of this project.

**Features:**

1. Create Rides: It allows the admin to add the rides that are available in the park

2. Show Rides: User can get the list of rides that are there in the park. It gives the list of available rides.

3**.** Show Ride by ID: User can get the details of a particular ride by ID.

**Technologies:**

•Frontend: JavaScript, HTML, jQuery

•Backend: Java, SpringBoot

•Automated Testing Framework: Selenium

•Code Quality Check: SonarQube

**Code Editor:**

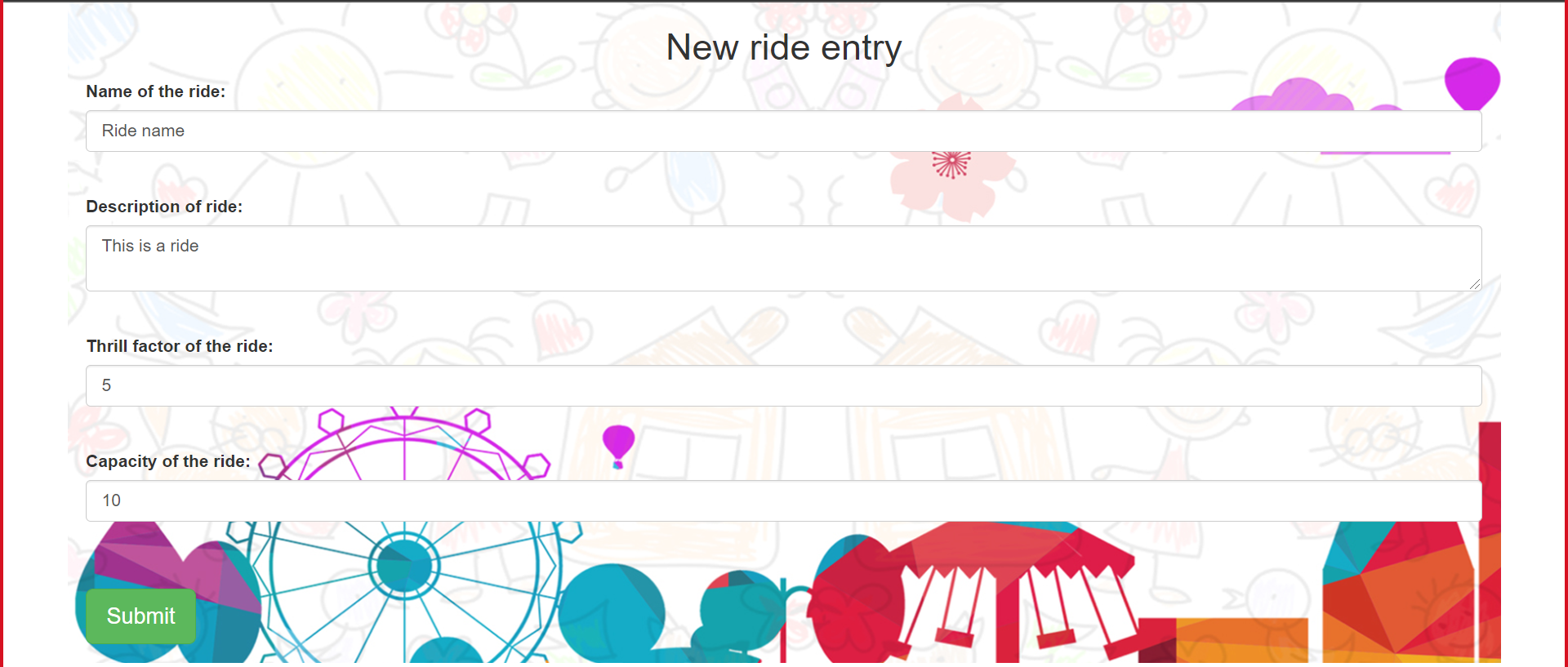
•IntelliJ / Visual Studio

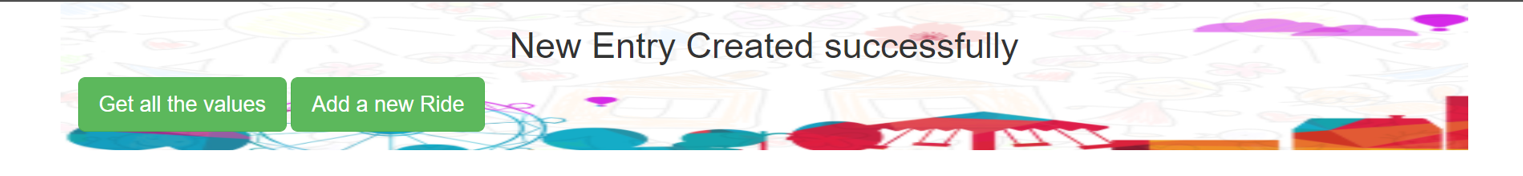
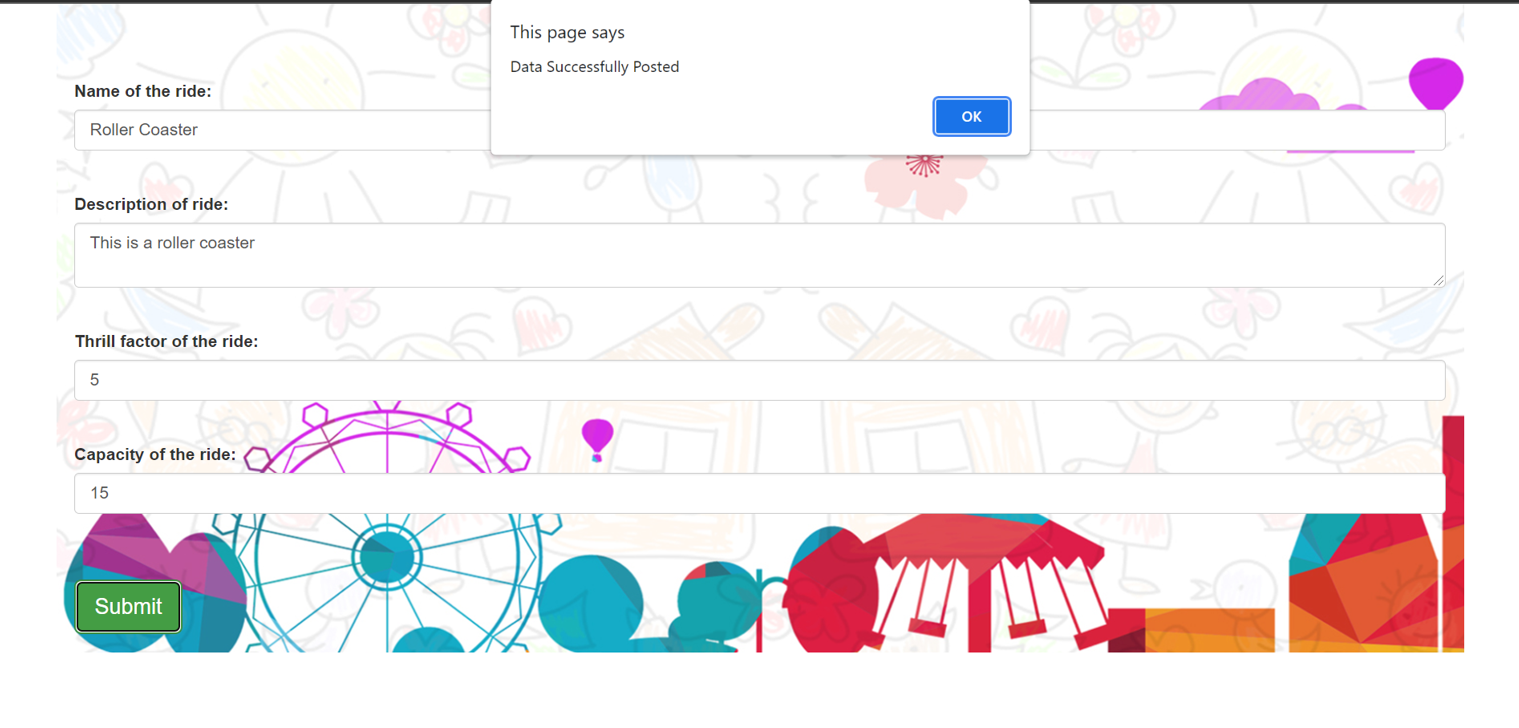
**Web Browser:**

•Google Chrome (version 95.x)

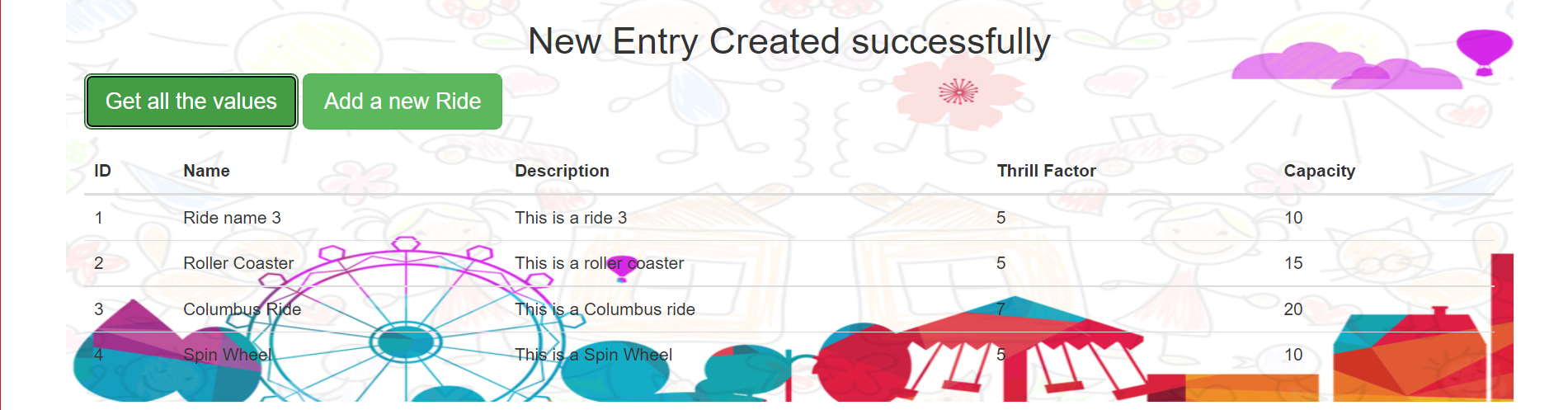
**Project Demo:**

**Creating/Adding Rides:**





**Getting Rides:**



**DevOps Tools:**

* IDE(IntelliJ)
* Git & GitHub
* Maven
* SonarQube
* Selenium

IDE**:**

Visual Studio Code and IntelliJ are used for code editing.

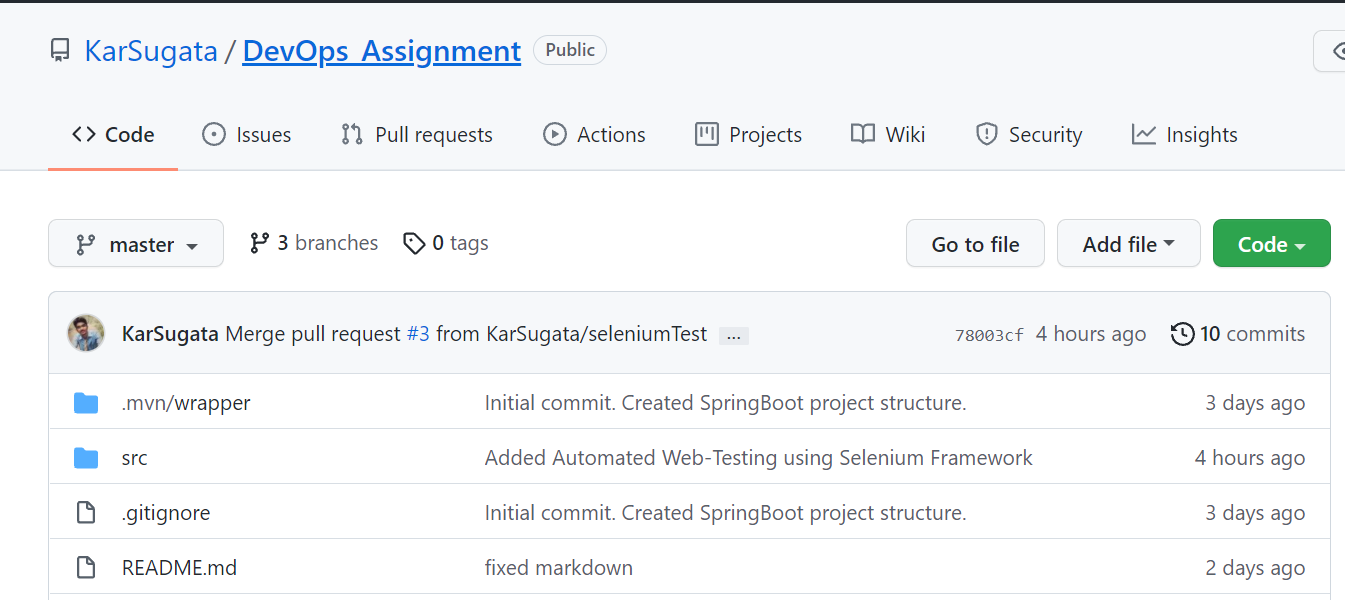
**Git and GitHub:**

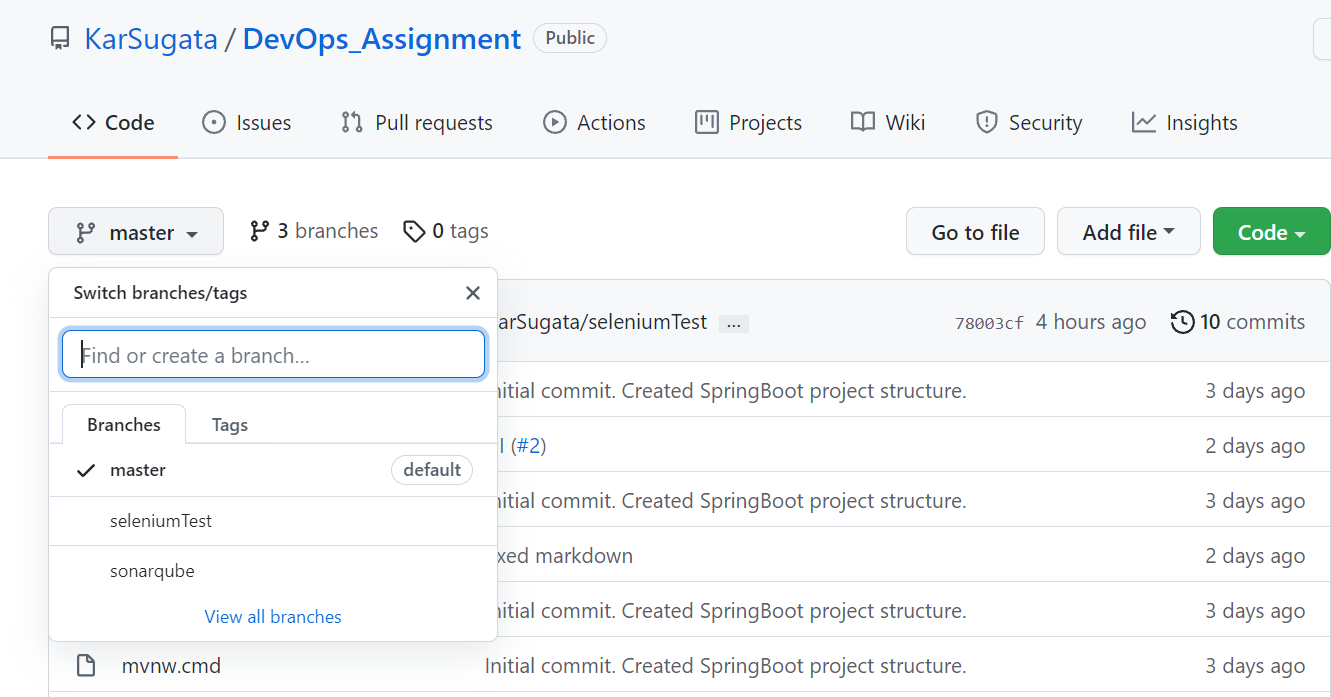
•We have used Git and GitHub for our Source code management.

•The workflow that has been followed is branch workflow, where the developers in our team have created their own local branches and pushed it to the master branch instead of directly working on main branch.

•We have also leveraged the feature of Pull Requests to merge our feature branches into our main branch in our central repository.

**GitHub Repository:** <https://github.com/KarSugata/DevOps_Assignment>



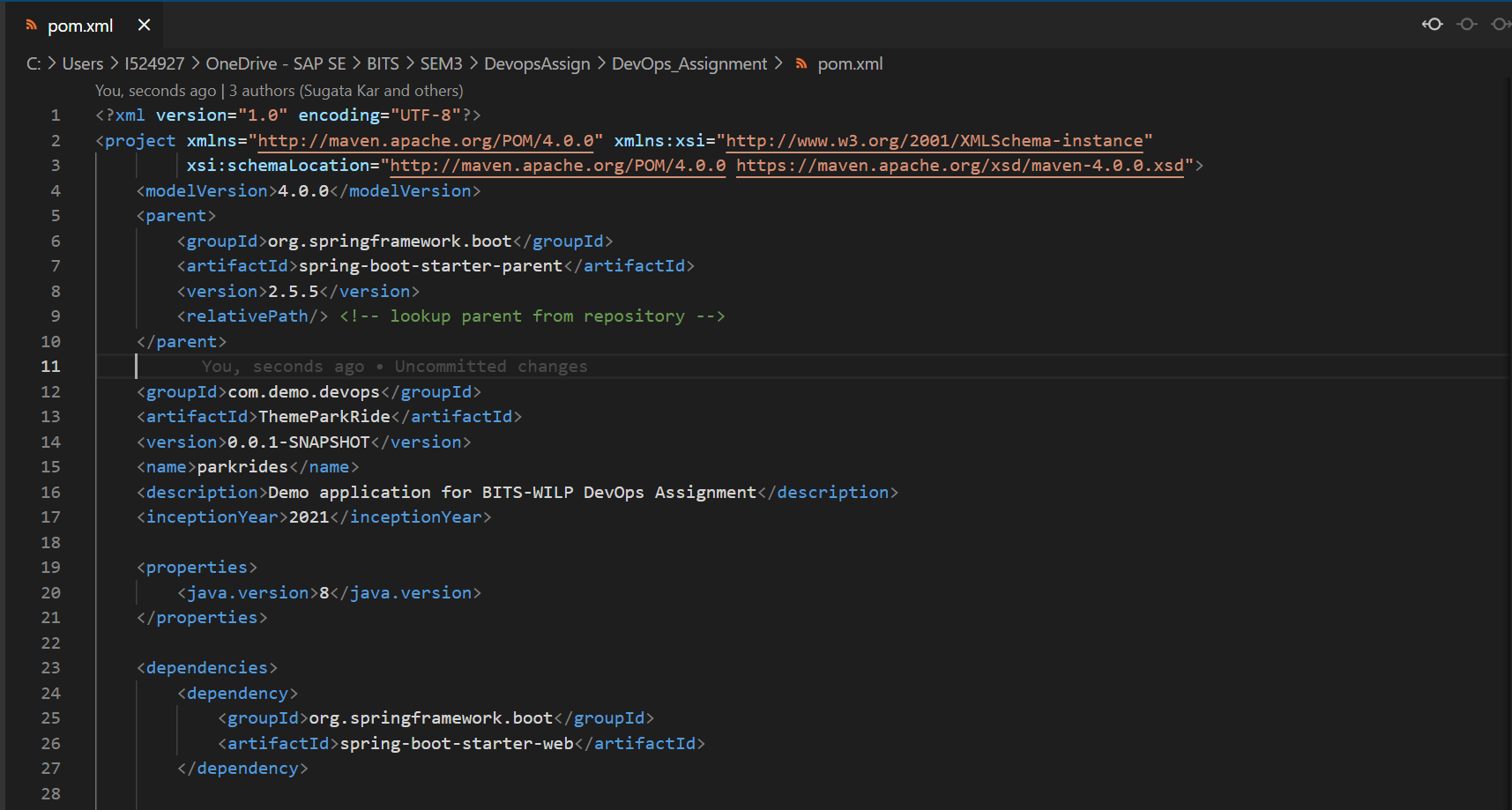


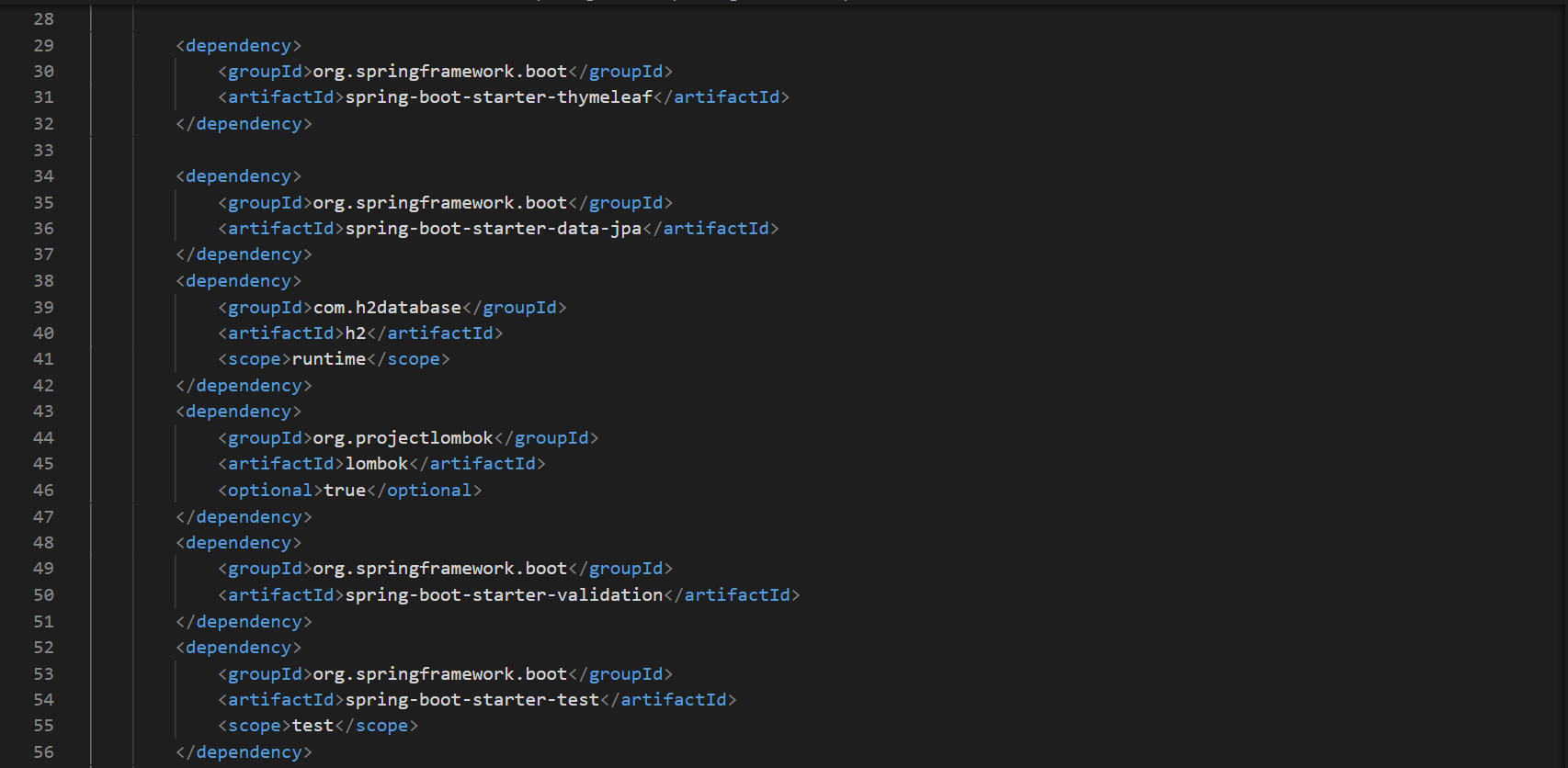
**Maven:**

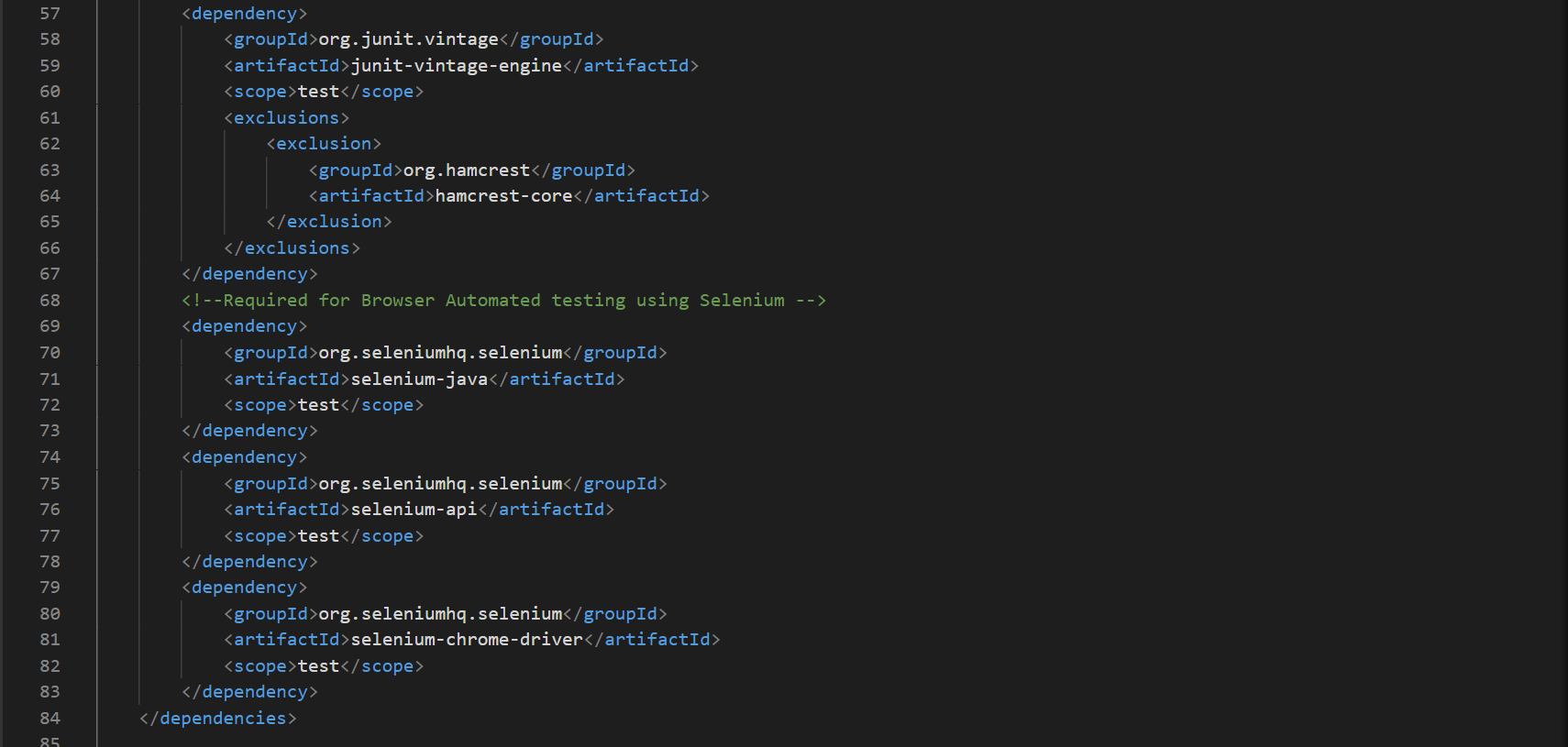
•We have used Maven for maintaining builds and dependencies of our project.

•Maven helped us in making our build process easy by hiding the complexities involved in maintaining the dependencies.

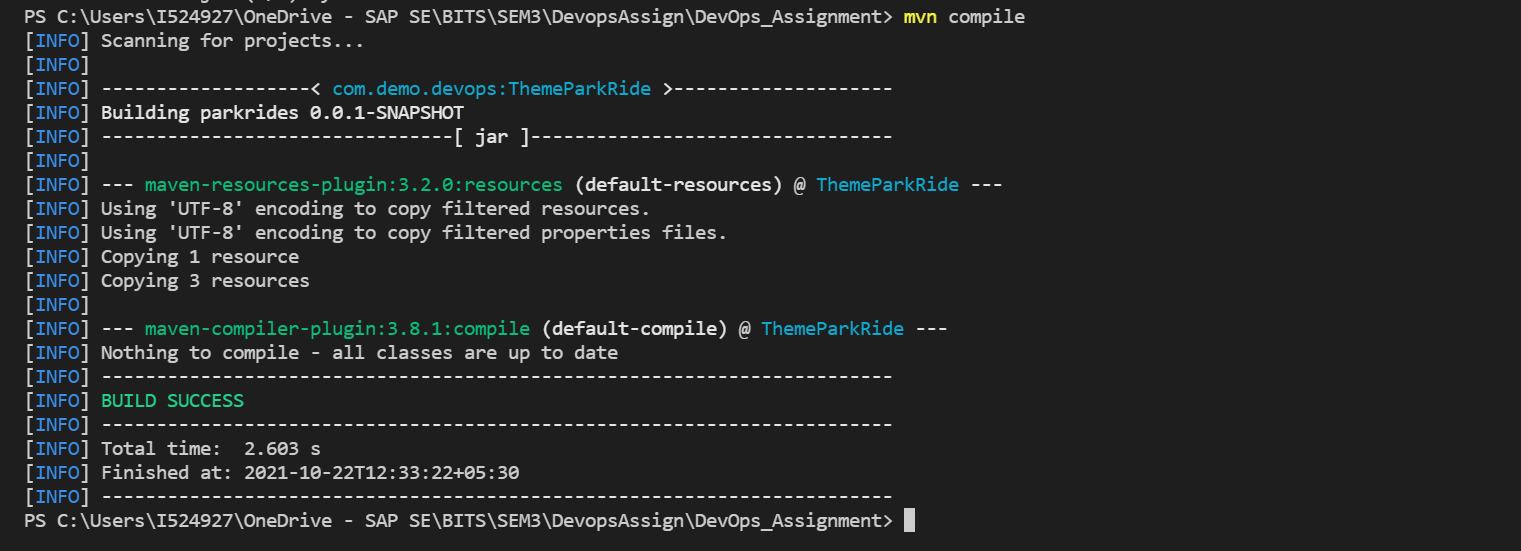
Pom.xml: (Depicting our dependencies)



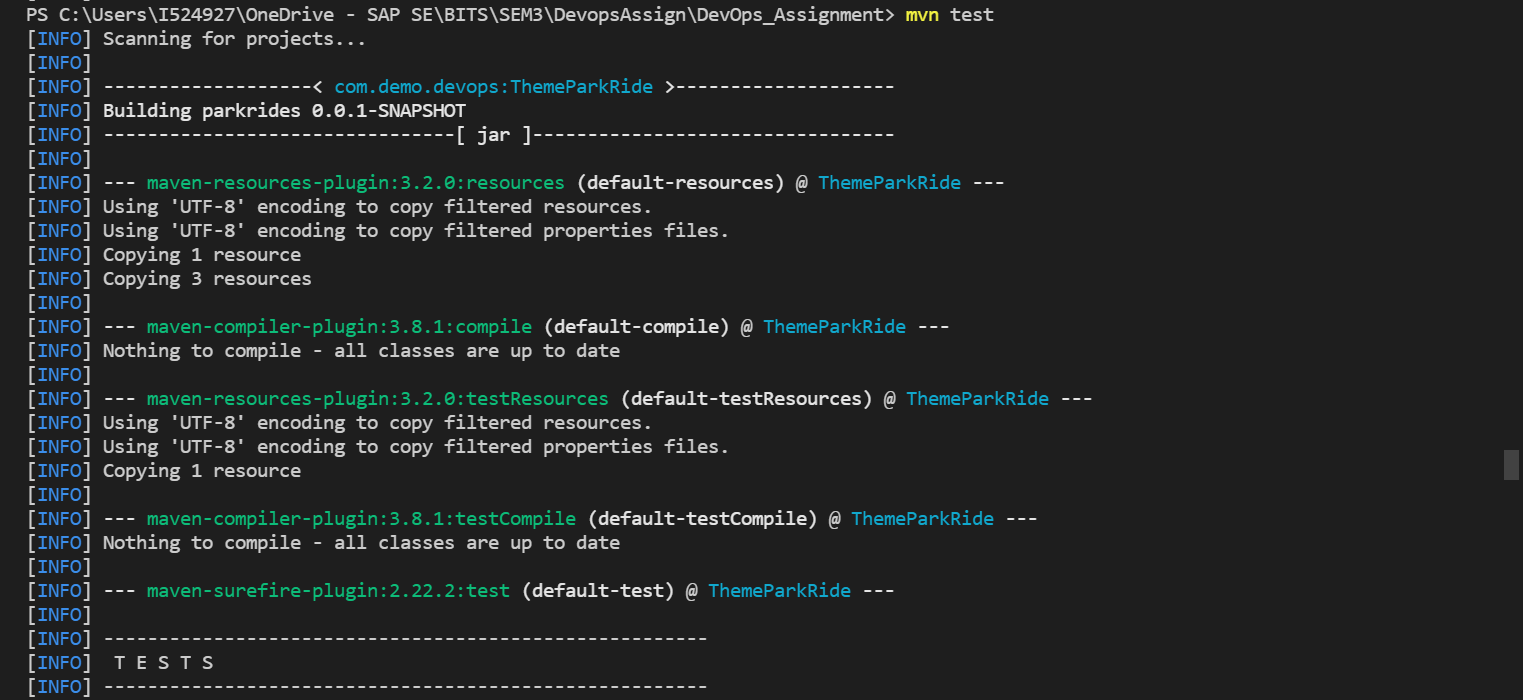


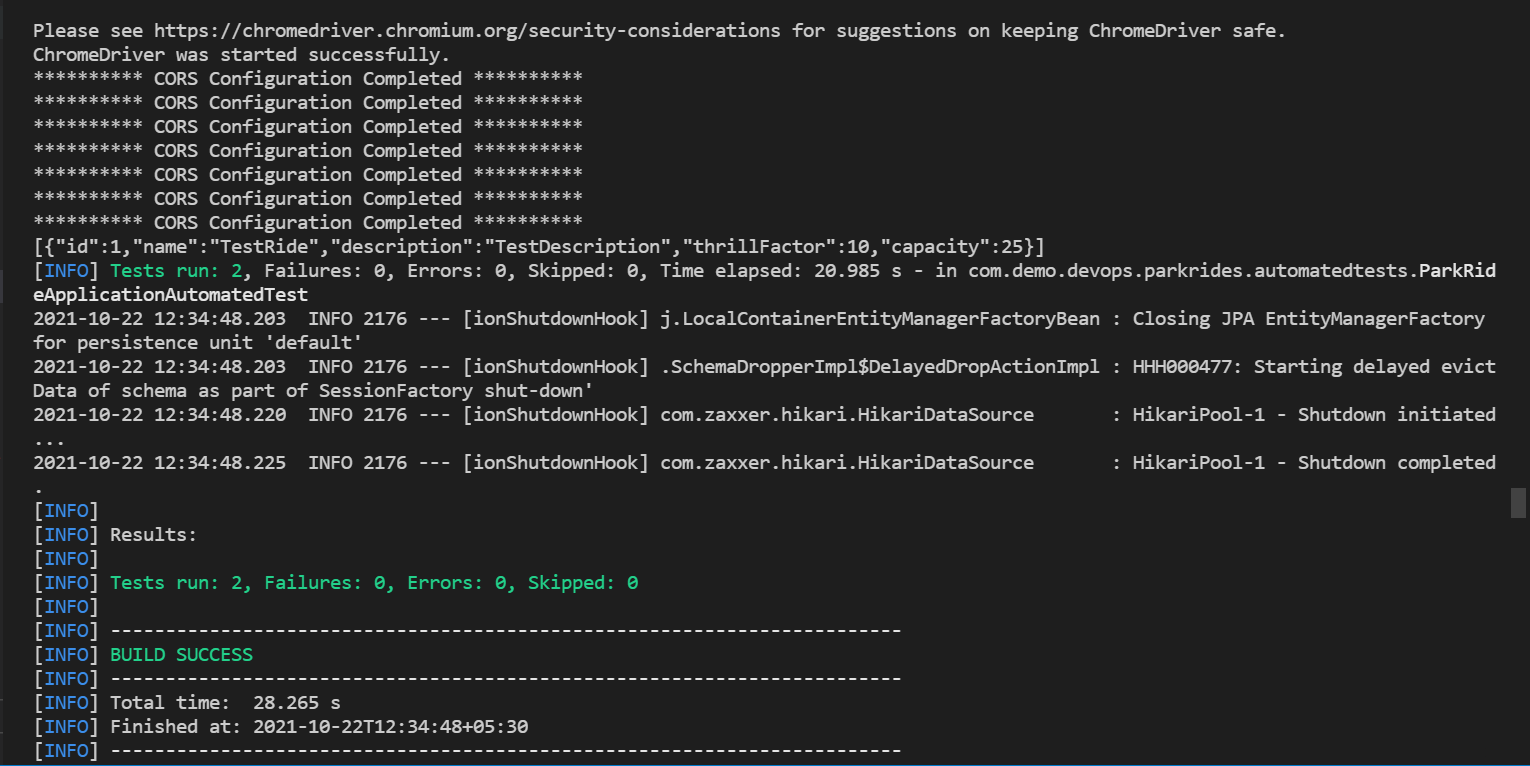


Compilation using Maven (mvn compile):



Unit testing using Maven (mvn test):

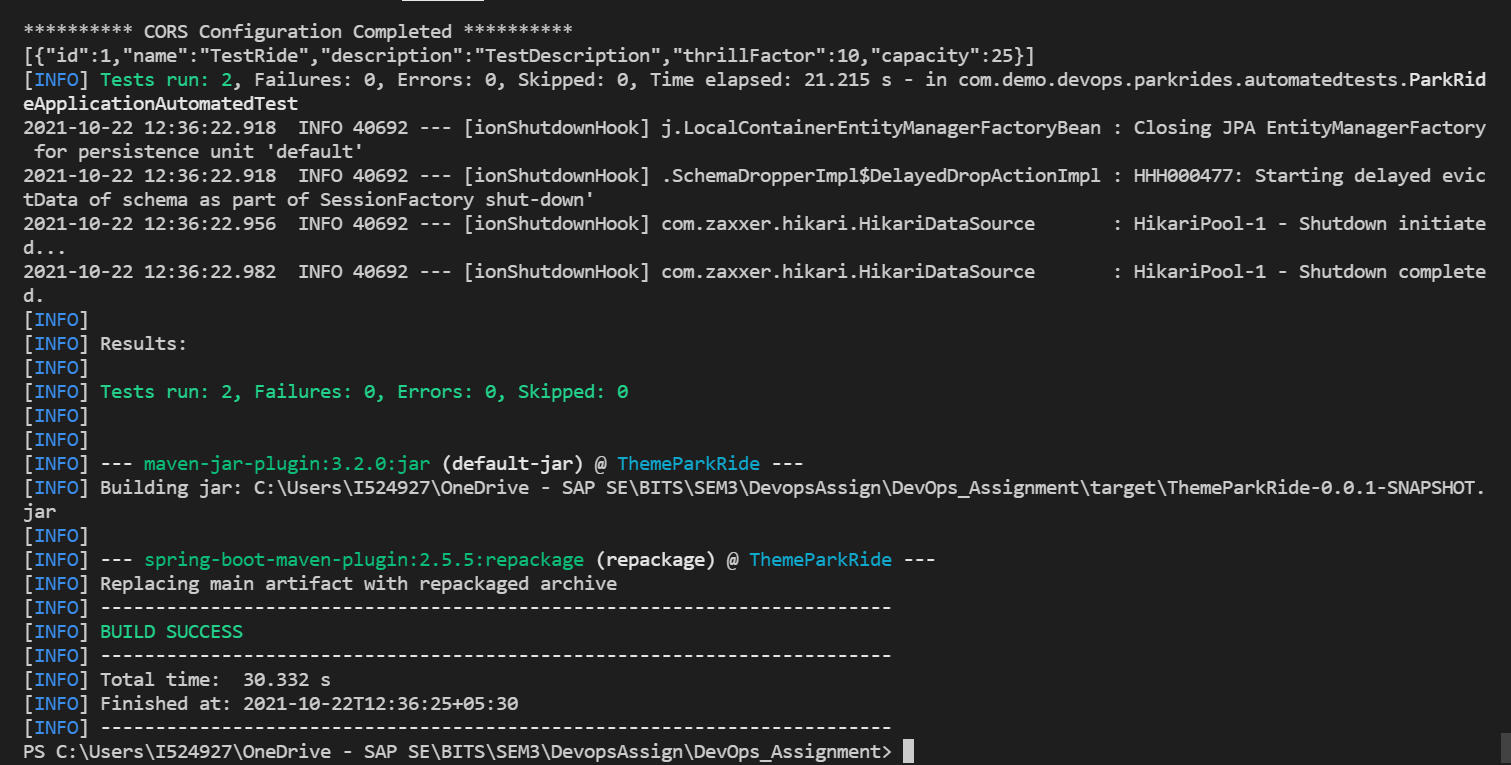




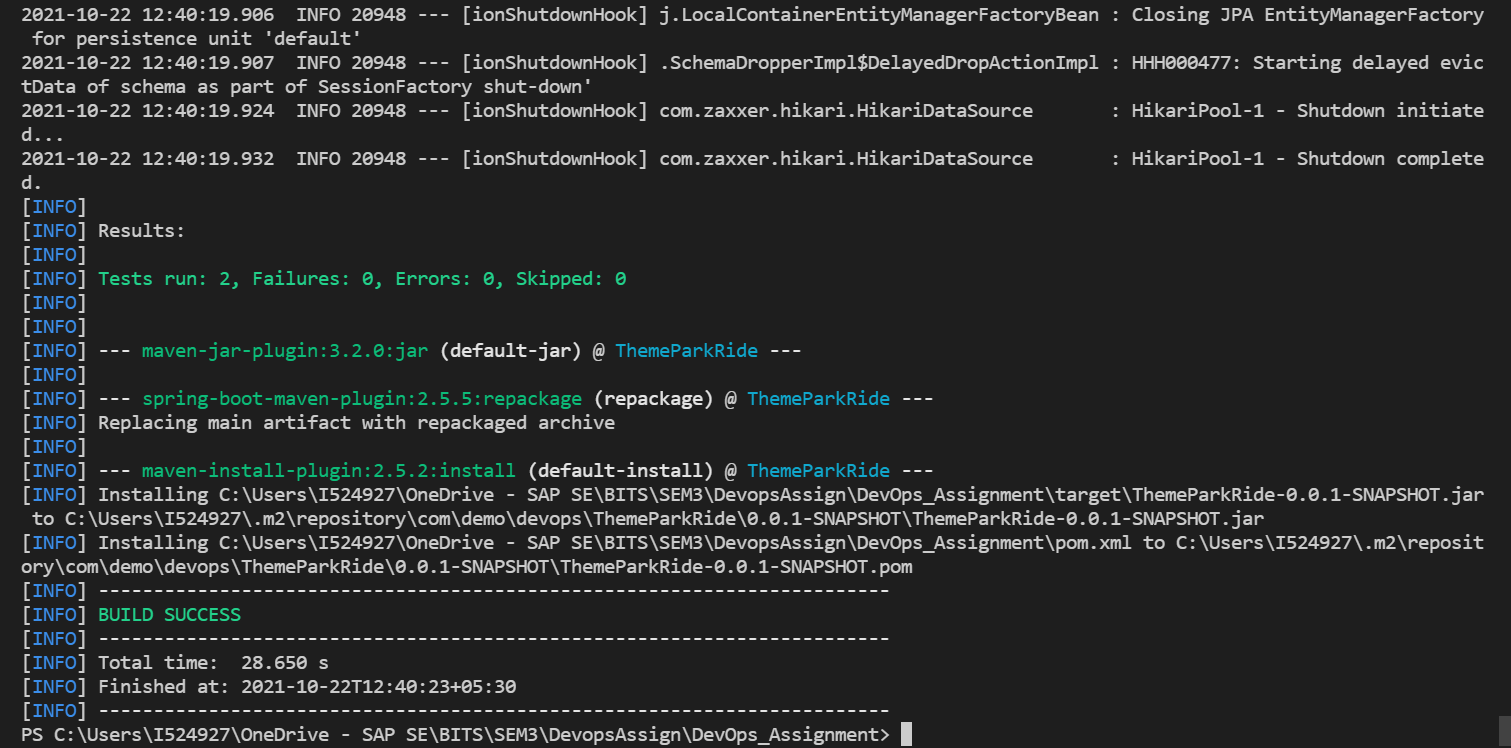
Packaging using Maven (mvn package):

We are packaging our project into jar file, so mvn package does this job for us.

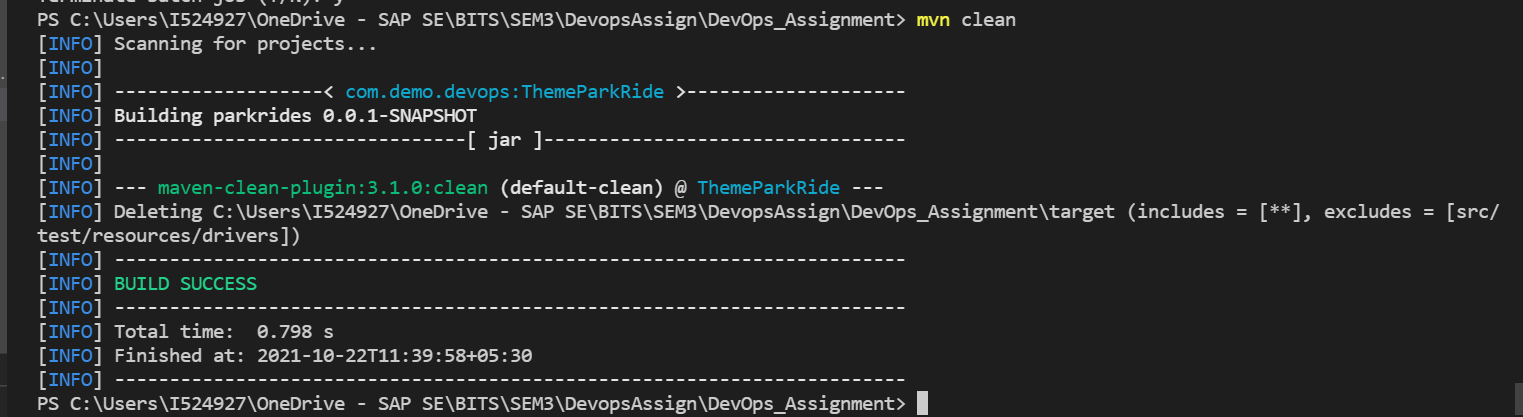


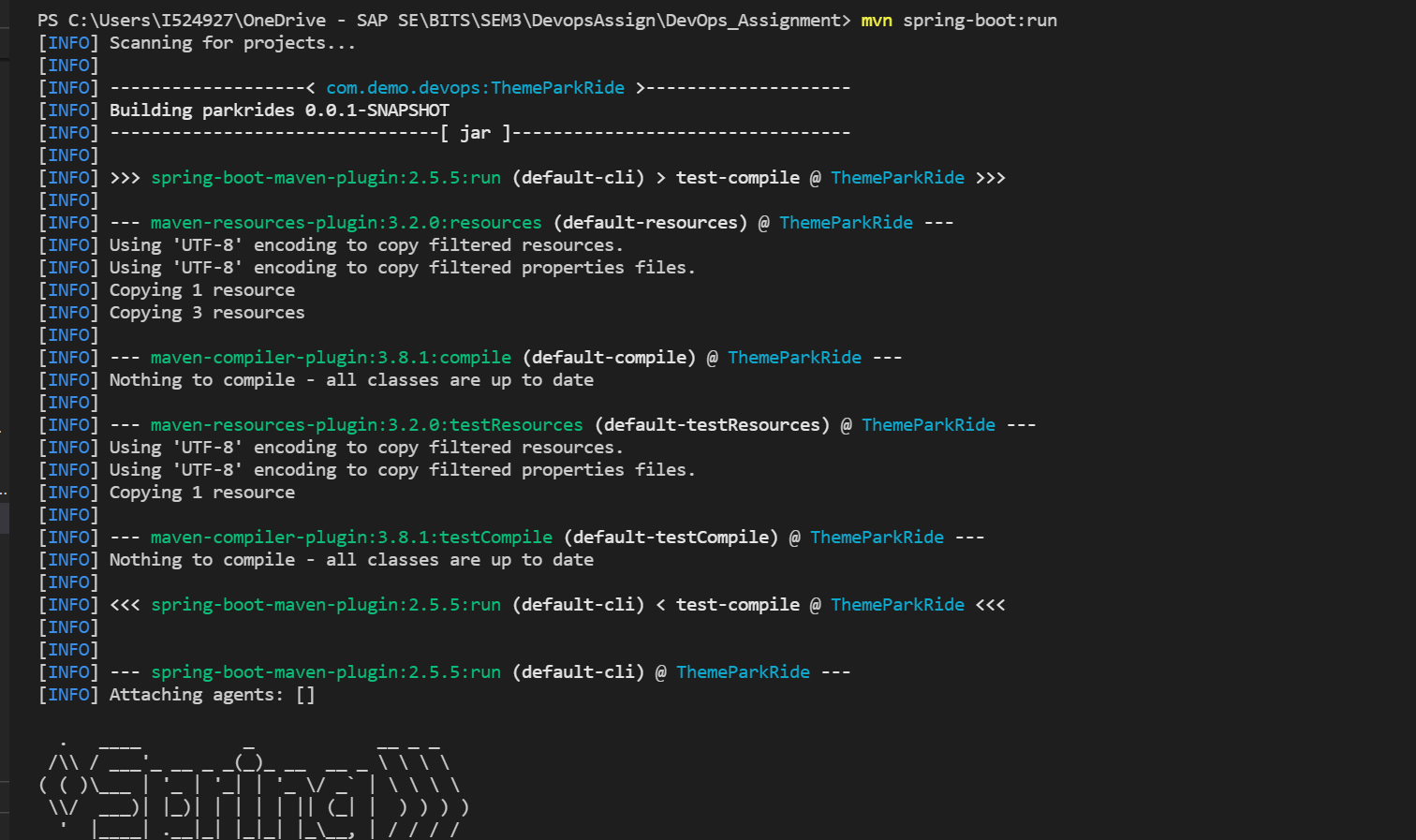


Installing in Repository using maven (mvn install):



Clean up using Maven (mvn clean):





**SonarQube**

We have used SonarQube for checking the quality of the code.

•Initially before creating our Jenkins pipeline we have downloaded SonarQube Zip file and ran the server in our local for checking the code quality.

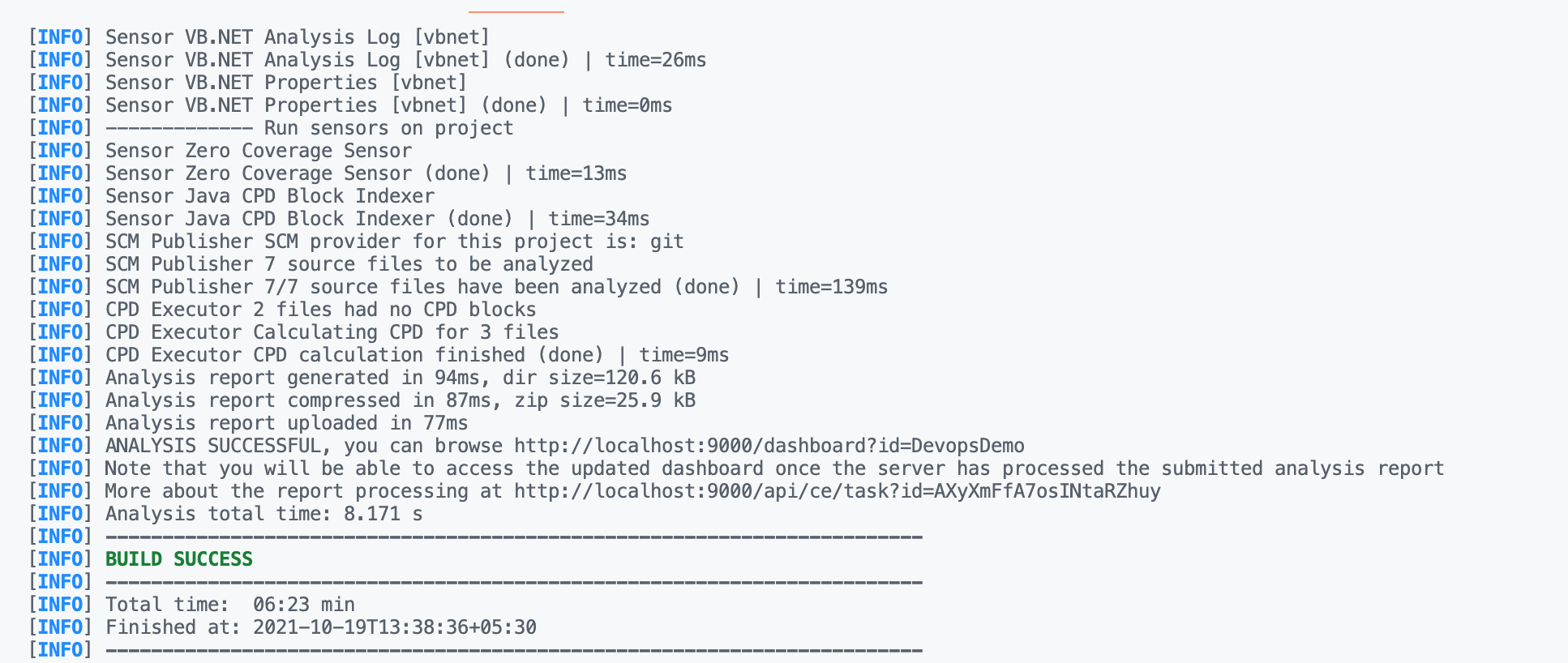
•After downloading the SonarQube Zip file we have started the Sonar server then created new project and then generated token for our project and our initial analysis is as shown below.

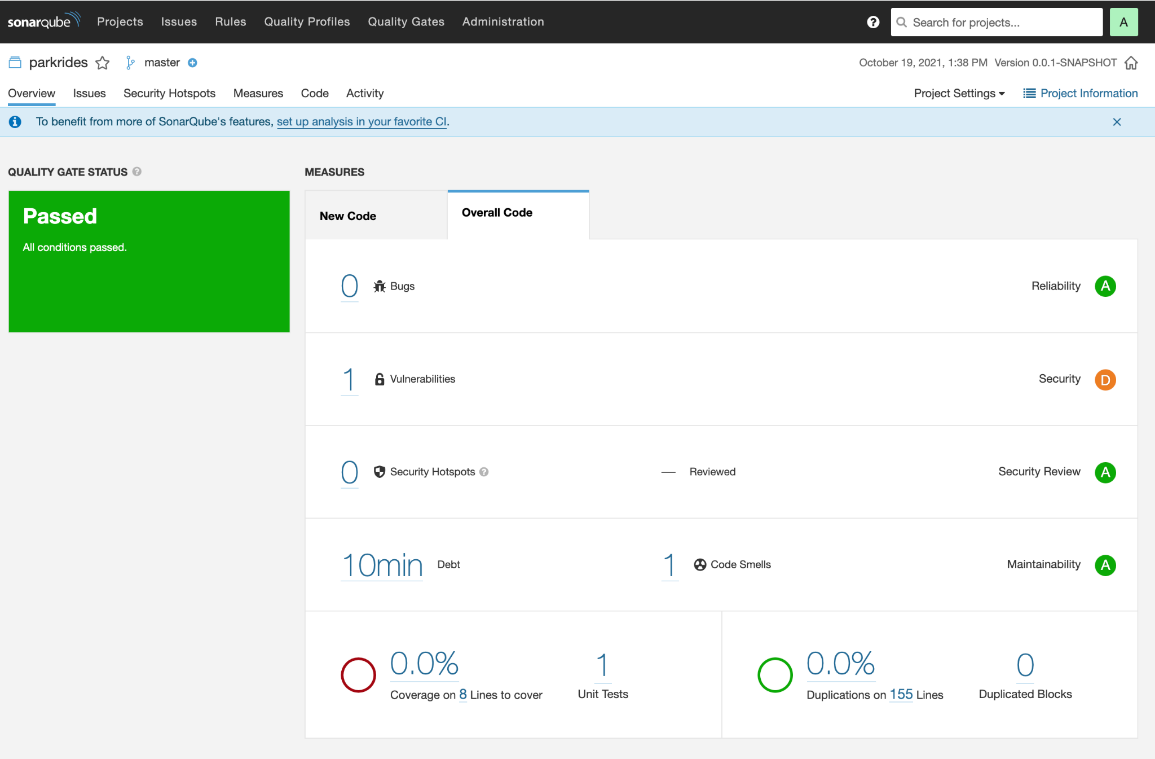
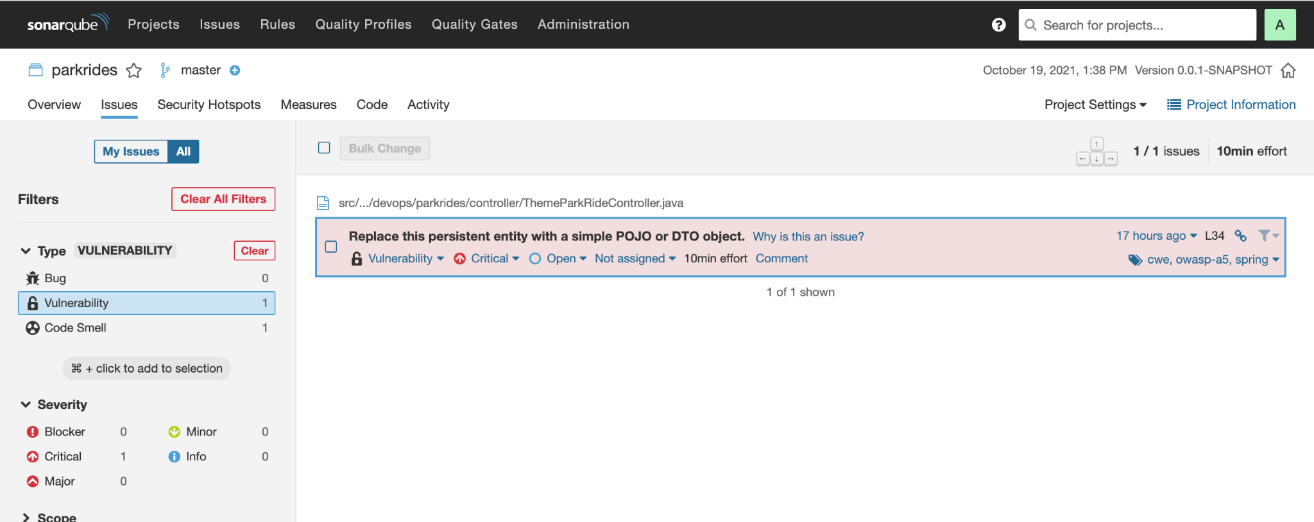
On running this command, maven will build the project and a SonarQube scan will be run on it, with which a bunch of console log will be generated as shown below.  
  
mvn clean verify sonar:sonar \

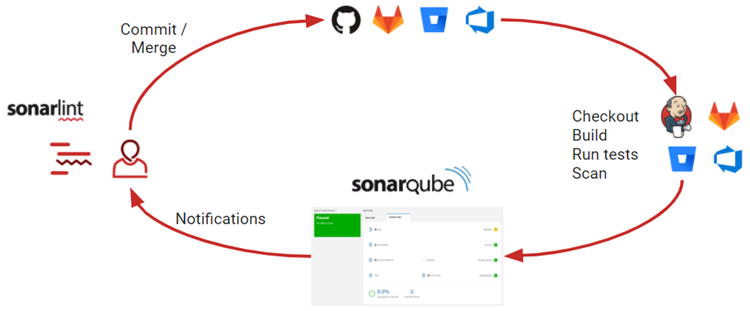
-Dsonar.projectKey=DevopsDemo2 \

-Dsonar.host.url=http://localhost:9000 \

-Dsonar.login=988a3a87402a8facb357a7a9724593990c45e60f



Post scan, a Scan report will be generated in the SonarQube WebUI.  
  
  
  
Here, you will find various Bugs, Vulnerabilities, etc that have been found via the scan.  
In our example, we had 0 (zero) Bugs and 1 Vulnerability.  
Upon clicking the Vulnerability number, the UI presents you with the specific Vulnerability/ies that has/ve been found.  
  
  
  
It describes the issues in a brief one liner and also estimates the effort duration that will be required to resolve it.  
  
  
  
Clicking on it further describes the exact line of code that caused the vulnerability.

All of this was run on a local test server, but these can also be integrated into a complete CI/CD pipeline. Once vulnerabilities, etc. are found, they are to be reported to the developer. Post which, the dev will provide with a resolution and it will go back into the CI/CD pipeline. The workflow is illustrated below.  
  
  


**Selenium**

We have used selenium for testing the front-end code. It is automating the testing on the google web browser.

Selenium is an open-source automation testing tool which is used for automating tests carried out on different web-browsers.

Selenium automation framework is very easy-to-use tool. It provides a user-friendly interface. We have used it to create and execute test scripts easily and effectively. We can also watch while tests are running. We can analyse detailed reports of Selenium tests and take follow-up actions.

**Jenkins**

Jenkins is a very crucial tool used In Continuous Integration and Continuous Delivery. It’s the widely used automation server which helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

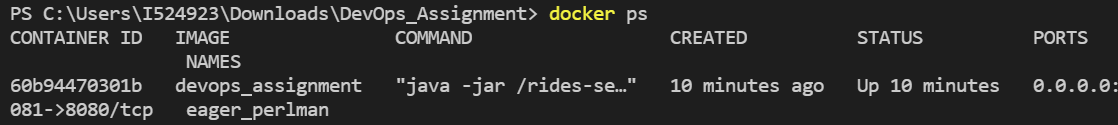
In our project we have set up a simple pipeline to automate the maven build process and the testing.

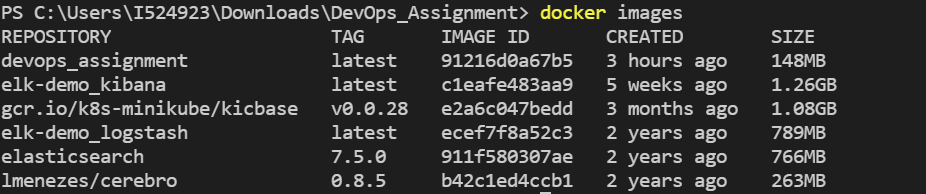
**Docker**

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels

For a full fledged application, docker images can be created for the frontend and the backend and communication can be defined between them. This makes it easy to scale up the application as required.

For our application we created a simple container to host the application.





**Critique Report**

The application is title Theme Park Rides, which Is a simple application designed to store the details about the rides that are present in any theme park. Along with it we also store some basic details about each ride, along with the ratings for each ride.

This application is aimed to simplify the task of maintenance of data. We can deploy this application for user stories which involves presenting users with the details of rides made available for them in a given amusement park or any fair being setup at any location.

The users of this application can easily add any ride according to it availability, remove any rides and display the list of rides that anyone can enjoy.

It is a very basic application with a scope of improvement by adding or extending the application to cater to new user stories. Meaning the basic framework along with the basic CRUD operation is designed.

Another critical aspect to be mentioned is the use of DevOps practices right from the initial beginning of this project. We used the Agile software management methodologies to decide on the user story, based on which the requirements were decided which led to the definition of backlogs for feature development.

The decision to include the DevOps practices was crucial to ensure smooth process before, while and after development. Use cases were carefully analysed to define the done criteria’s and the come up with the DoR(Definition of Ready) and DoD(Definition of Done).

After this each developer in the team took individual tasks to develop the features of both the frontend and backend while carefully testing each feature as and when completed to ensure quality code from the very beginning.

Various DevOps tools were used by the team, which made things easier for the team while development. Version control was implemented using the source control framework of git.

Git repository was created where the code was being pushed by the team continuously enabling the tracking of progress easy and efficient.

The application was designed using the maven framework for dependency management and continuous build. Some of the dependencies used are H2 repository to store values, Spring Web for frontend etc..

Giving equal priority to quality of the application we used some widely used tools tom monitor the application. Sonarcube was setup for the application to identity the possible code smells and setting the standard for quality checks. Suggestions for the various code smells was helpful to build the application matching the market standards for quality to ensure bug-free stable code for the customers.

To boost the quality further, different types of testing was equipped to test the code from the functional level to UI testing. JUnits were written to test the working of the code at the functional level and the most popular tool of selenium was used to automate the UI testing.

Another important idea was to automate the process as much as possible to ensure continuous delivery while keeping the standards intact. Jenkins pipeline was setup with git as SCM which triggered the various stages defined for each commit and code change.

Docker container was also setup for hosting the application giving us flexibility to handle user requests and any failure in the application resulting in direct impact to the users by using the various scaling features supported by Docker.

The team acknowledged the importance of DevOps practices to ensure continuous and progressed feature delivery. Timely meetings to discuss the status of the application, to check the delivery of features,to acknowledge the completion of development, to test the feature being delivered were setup with a focus to build reliable code.

Some of the challenges faced by the team with the development of the application includes, picking up the right framework among the available many to provide variety of features. Ensuring proper development practices to build quality code. Using the best industry practices to avoid failures and bugs. Enhancements to the existing features and deciding on the new features aimed to provide solution for the various user stories.

Testing the application on regular basis, after code changes and ensuring all features are working as expected and designed i.e reliability and maintainability. Usage of the various DevOps tools and configuring it with the application.

The application has a lot of scope of improvement, we currently have limited logs management which can be a crucial resource to identify and prevent application failure by constantly monitoring the usage of the application. The ELK stack can be used to capture these logs and monitor the various factors of the application. Active monitoring can also be helpful to improve the standards of the application, to improve the response time and provide faster and better experience to the user by giving the regular updates.

Visualizations can be included to make the application more user friendly and attractive to grab their attentions. Security can be enhanced to ensure confidentiality to the user data and avoid any chances of cyber attacks is one major gap that we have to work on. Code to improvise the application to make it faster with less efforts is something that we can always focus.

Focus on more user stories to provide feature rich application to increase the userbase. Scaling the application to ensure performance under heavy load.