**CASE STUDY REPORT**

This is the report for our case study. We belong to the fourth group of the second batch of Java Full Stack department in Telstra bootcamp for 2023. Our group members include Tunir Chaudhuri, Akash Ranjan, Goutham Suresh, Rahul and Sunny Kumar.

The topic for our case study is TSE Initializr. According to the instructions we received from our requirements document, it was mentioned that –

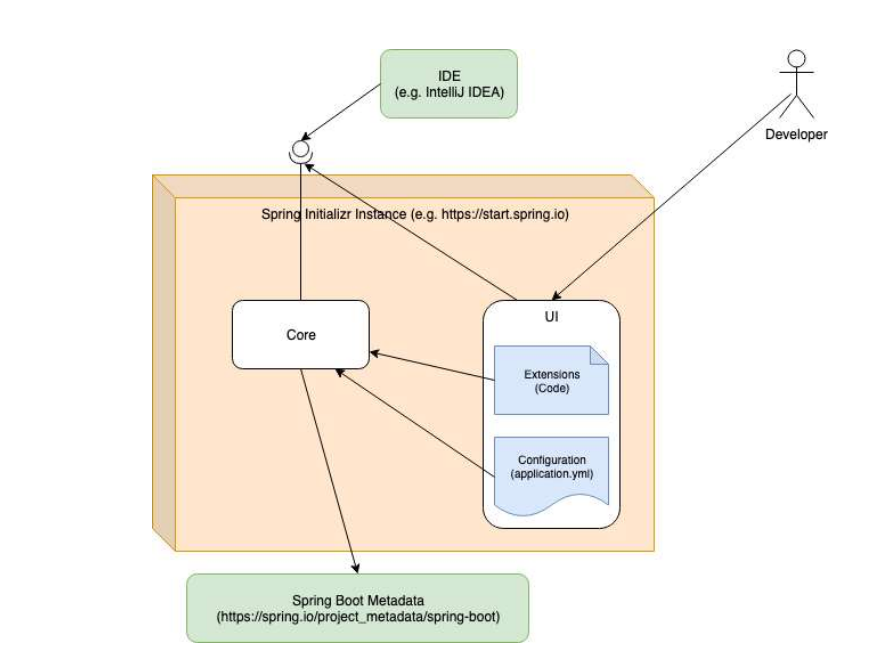
“While https://start.spring.io/ is an excellent start for your next Spring Boot project, sometimes it may be necessary to customize it to your own needs e.g., for the following reasons: You may be required to provide a self-hosted instance of the Spring Initializr within your company e.g., due to network restrictions when downloading archives from the web. You may want to tweak or brand the UI or even build your own UI from scratch. You may want to provide your own project configurations and/or dependencies e.g., company internal Spring Boot starters that are not publicly available on the web.”

So basically, the main aim of our project is to simulate the functioning of Spring Initializr on localhost with some added dependencies and other special features. These are the user stories that we our application has fulfilled –

1. The application should allow us to add dependencies.
2. The application should allow to choose options between war and jar.
3. The application should allow us to choose the appropriate jdk version.
4. The application should allow us to mention the domain, artifact, name of project and the description of a project.
5. The application should allow us to choose Maven or Gradle.
6. REST response should be successful to get the accurate dependencies from Maven or Gradle.
7. ZIP file should be created properly and saved on system.
8. We should be able to load the project in Intellij or Eclipse or any IDE.

We must mention here that the scope for our case study is to deal with custom TSE Initializr, custom dependencies and custom front end user interface only. We are not supposed to invest in the complete backend configuration and complete frontend configuration because these are already available from our source.

Here is the architecture of our TSE Initializr, which is similar to the existing Spring Initializr. We have used Spring Tools mainly.



Our first step was to clone the existing Initializr and Start Spring Io projects from their GitHub repository to our local system.

1. <https://github.com/spring-io/initializr>
2. <https://github.com/spring-io/start.spring.io>

Next, we had to resolve the Spring version conflicts, system configuration problems, Maven build errors and workspace issues.

After this, we added our dependencies to the Application.yml file present in the Start Site project.

The first dependency is CRNK dependency which is taken from JCenter repository. This dependency improves the REST API feature in Spring Boot.

The second dependency is a Hello World dependency. This is a custom user defined dependency which displays Hello World. We used GitHub repository for this dependency.

The third dependency is also a custom dependency. This assists the user in connecting to the database and performing various operations. We used GitHub repository for this dependency.

We created a user guide page that contains almost everything a new end-user or client needs before using our TSE Initializr application. A lot of test cases were already there in the original project. They were in two different files App.js and Initializr.js.

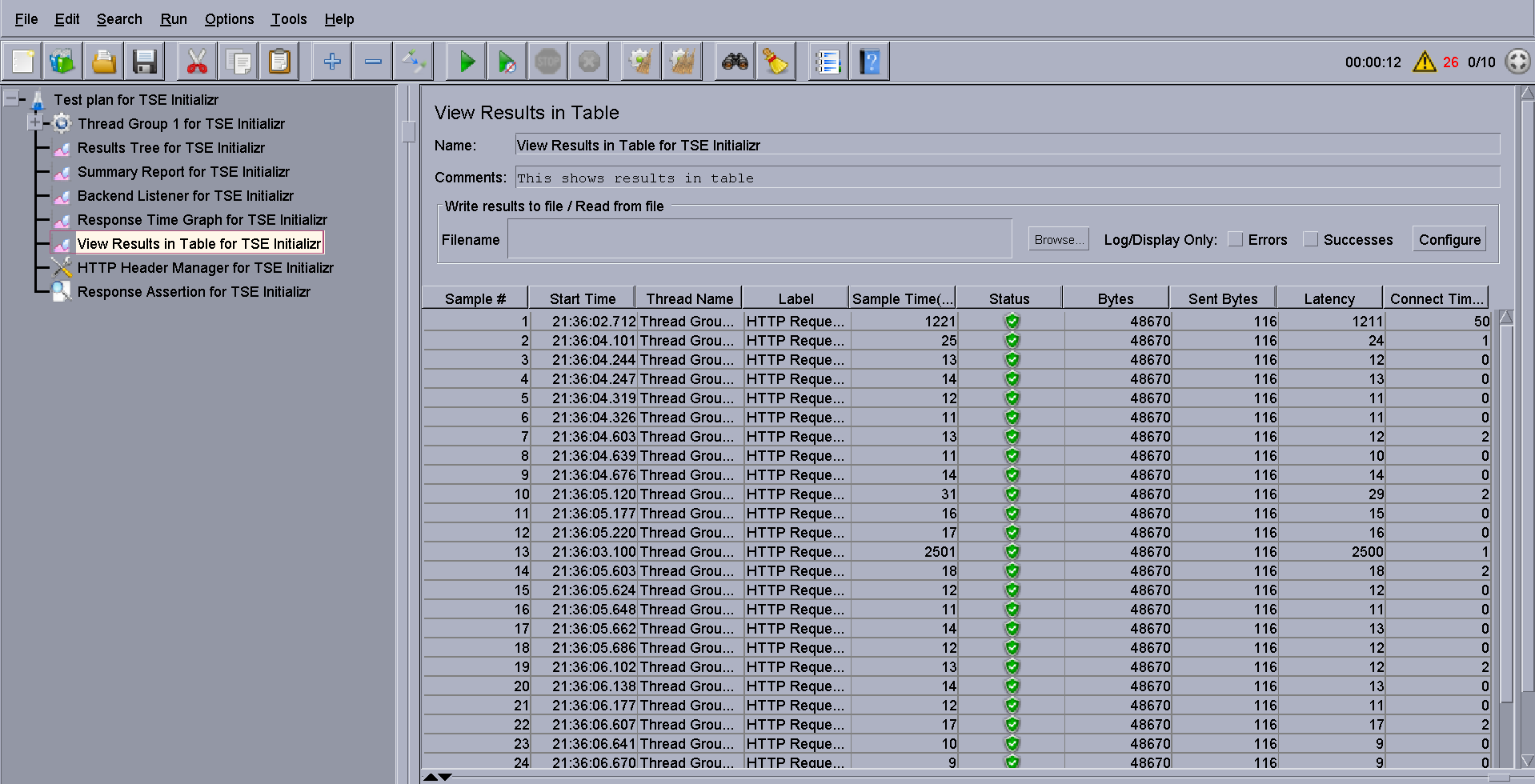
The existing test cases cover a lot of areas including-

1. Colour theme – whether it is selected as light or dark and whether it is getting reflected.
2. Update error – whether it is getting displayed when the user demands a feature or support not available for that version but is available for a higher version.
3. Group conditions – whether the groups can be added, removed or toggled.
4. Dependency conditions – whether the dependencies are getting displayed, added, removed or used in the zip file.
5. Project settings and configuration – whether the details regarding project name, language, Spring Boot version, group id, artifact id, description, JAR/WAR, package name, Java version are getting selected and applied correctly.
6. Errors and Warnings – whether the errors and warnings are shown correctly and whether the warnings are not shown when the user attempts to hide them.

We added a few more test cases in our performance testing tool – Jmeter.

A screenshot of a computer

Description automatically generated



Then we modified our front-end user interface. We changed the default green colour of Spring Initializr to Telstra’s blue colour. We also changed the default Spring Initalizr logo to Telstra’s logo. We removed the GitHub and Twitter links for Spring Initializr and added our own GitHub link and user guide page respectively. We removed the Spring Initializr advertisements and added a link that redirects to Telstra’s website. Here is the updated front-end user interface. This can be viewed in our localhost = <http://localhost:8080/> . This is currently the only end point we have in our application.

