****

**CSCI 5308: Advanced Topics in Software Development**

**GROUP 11**

| **HEMANTH NADIPINENI** | **B00899473** |
| --- | --- |
| **SANGRAMSINH MORE** | **B00903585** |
| **TRUSHITA MAURYA** | **B00913134** |
| **NISHIT MISTRY** | **B00911296** |
| **SUBASH NARAYANAN** | **B00899481** |

**TA:** MANJINDER SINGH

**PROFESSOR:** TUSHAR SHARMA

**Quality Report**

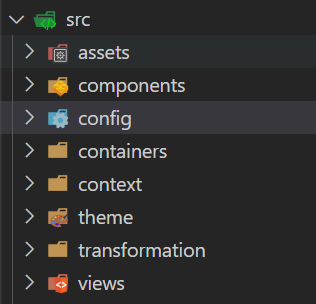
**Structure and architecture sanity**:

Three layered structures have been implemented in the front end and back end.

Front end we have implemented: 1) View 2) Container 3) Component

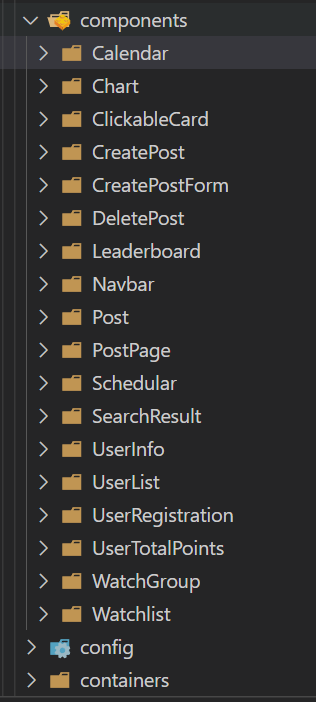
**Front end structure and architecture:**

We have separated all the distinct functionalities into different folders, like configuration info that will be used to connect to API and database, then the entire application in general inside the theme folder, Views, container and components separating data and business logic from views and components, separating all the assets and data transformation.



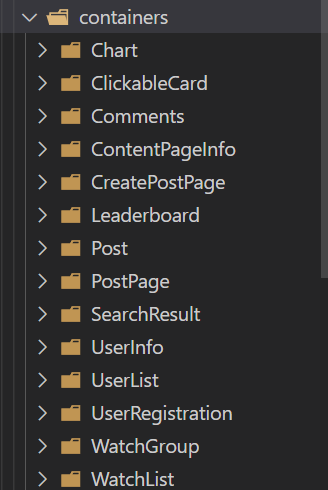
**1) Components**

We have created all the components separately that can be re-rendered easily on passing data, we have a separate folder called components that contains all the elements that can be rendered helping the cohesion.



**2) Containers**:

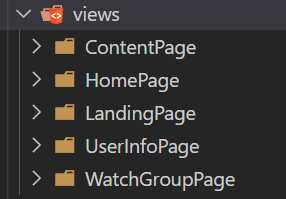
Our business logic is written in containers and calls the corresponding components to be rendered, separating the business logic from components and reducing coupling between components.



**3) Views**:

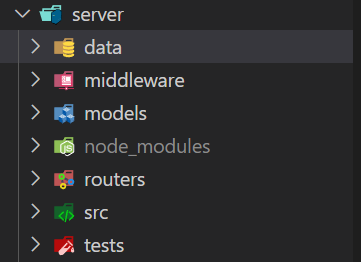
View holds all the containers to gather and renders the containers and in-turn individual components are rendered.

View -> Containers -> Components.



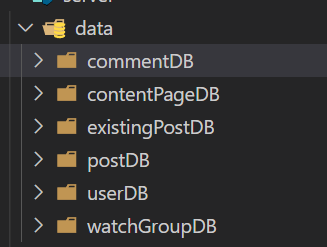
**Back-end structure and architecture:**

We have implemented 3 layered architecture: 1) Data 2) Service [Middleware] 3) Controller [Routers]

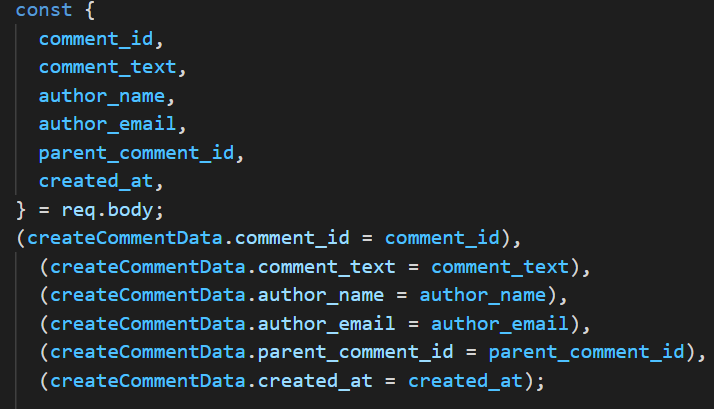


**Data:**

Data has all the connection standards to connect to mongo collections and interact with it.

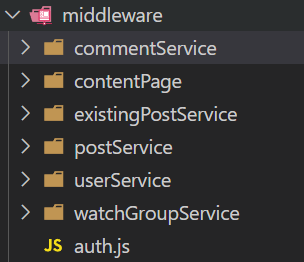


Where we set the object's value to handle request and response.

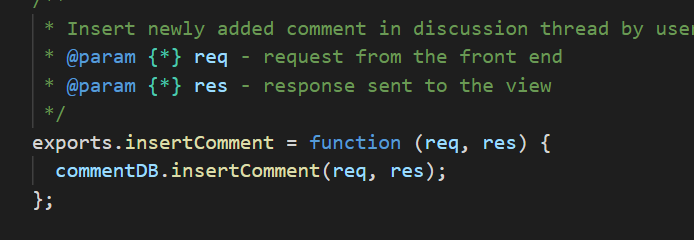


**Middleware:**

Middle is through which the interaction happens with the DB where request and response is passed, we have separated all the services in the middle ware.

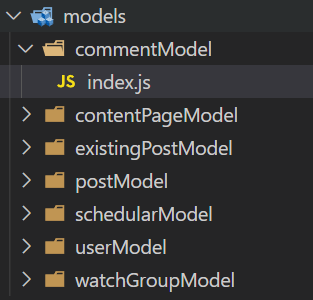


Here is an instance where middleware inserts the comments.

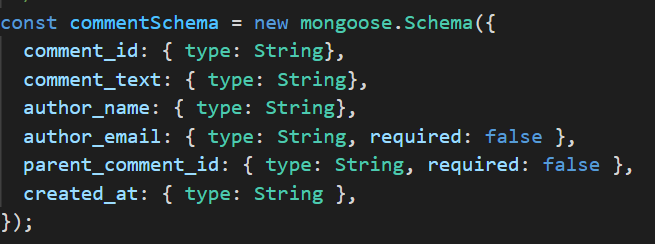


**Models:**

Models are where we declare the object structure that will be sent to the database collection.

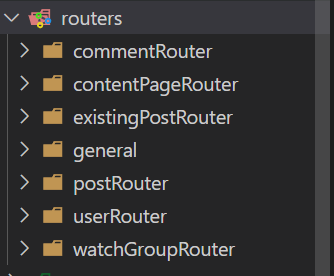
****

Here is the instance of an object declaration:



**Routers:**

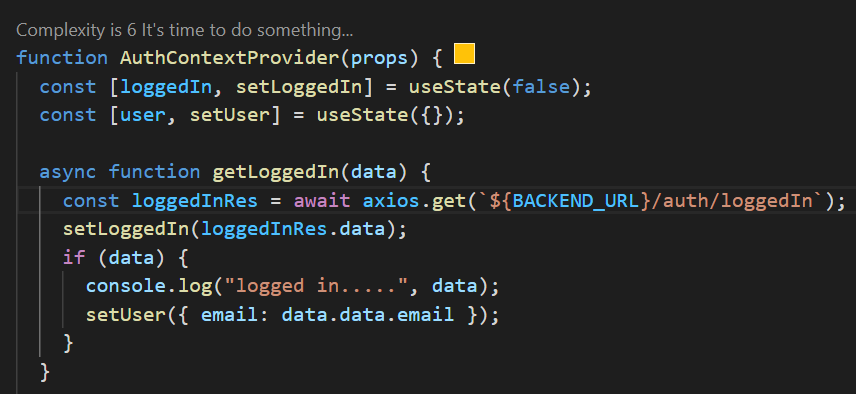
We have set all routers that will be redirected to a particular middleware based on the request from the front end.

****

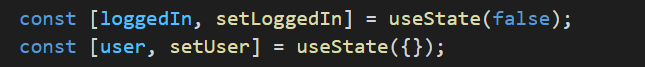
**Other clear code practices**:

We have used the Latest ES7 coding standard in javascript with proper indentation for readability.

We have used the code metrics plugin to see and solve or reduce complex functions.

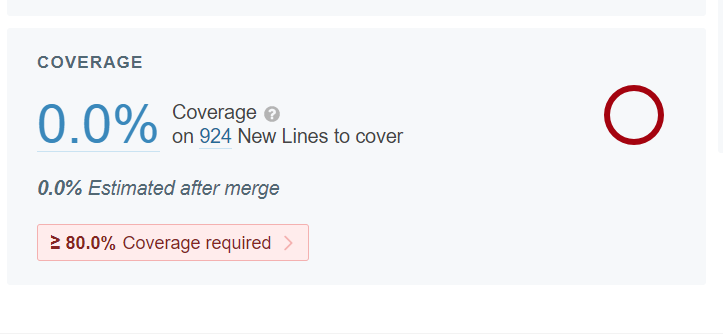


We have used **Const** and **Let** instead of **Var** and avoided global variables at all places.

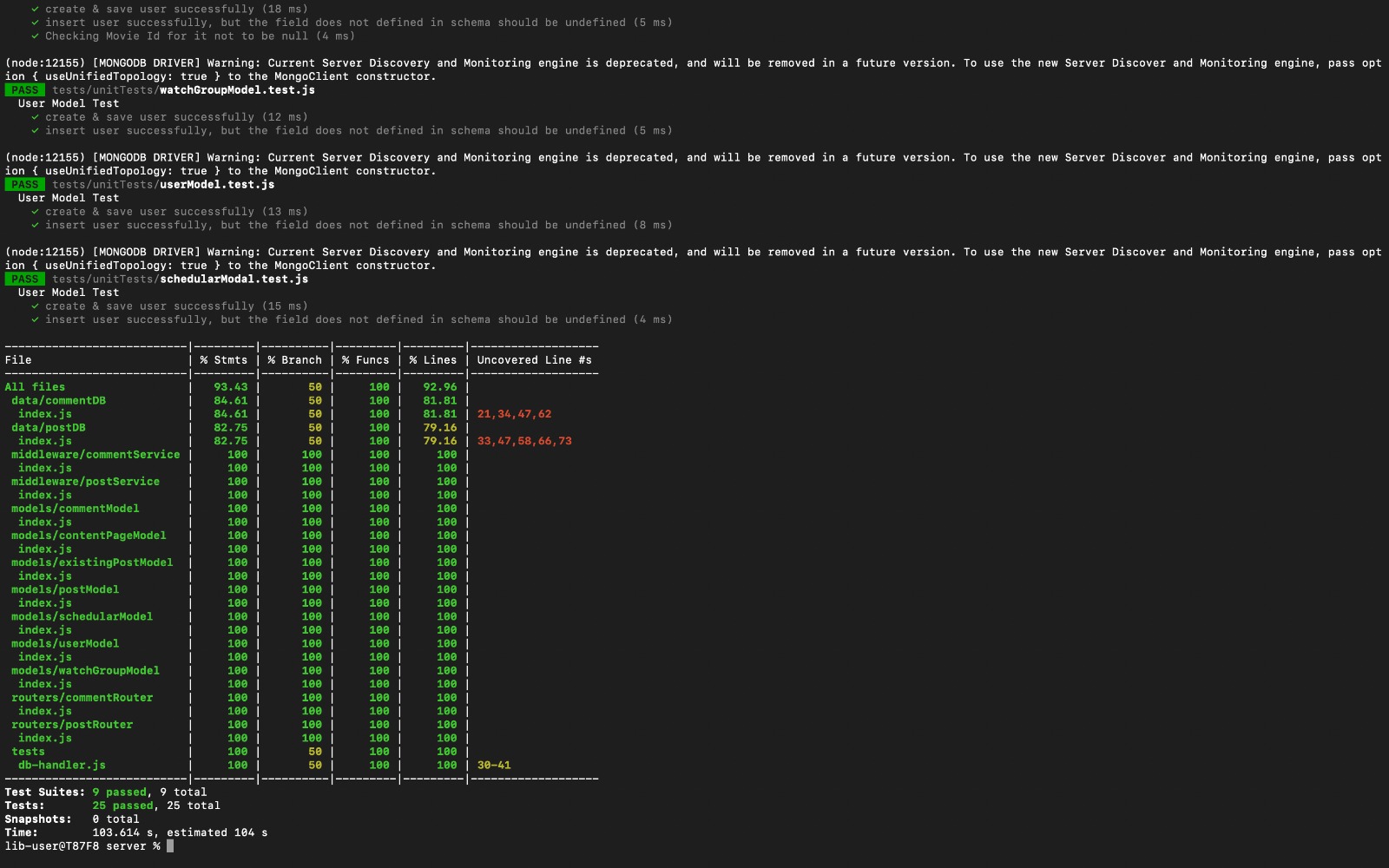


**SonarCloud**:

Coverage was not running for the new line and we faced difficulties in setting up the sonar cloud to identify new lines to cover.

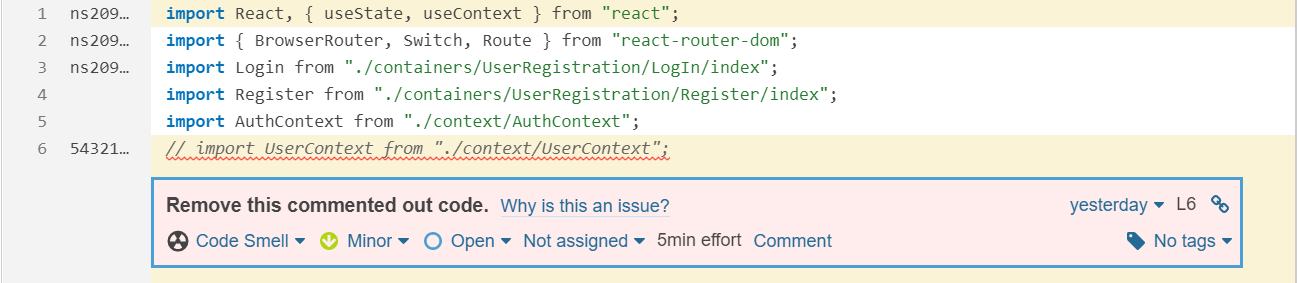


We defined our own quality gate to ignore coverage, but we have written all the test cases both unit and integration testing inside the application with coverage of 100%



**Smells:**

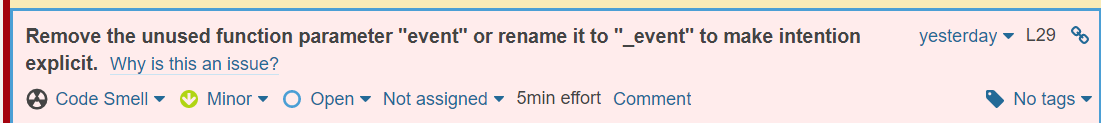
We had smells related to removing unused imports and comments that exist.



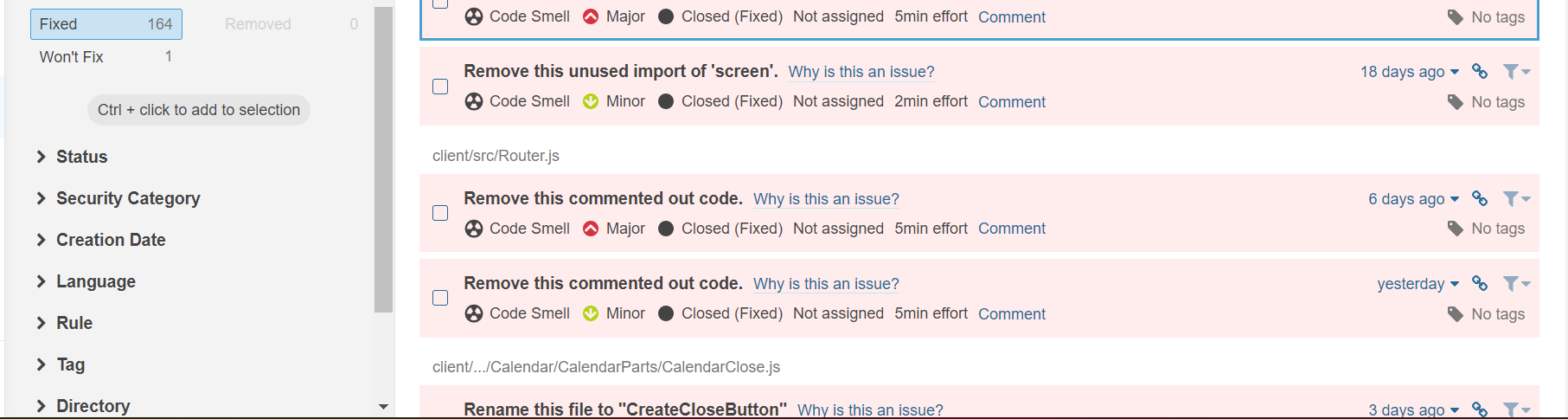
Solving redundant imports :



Removed Unused parameters:



Solved issues : 164 issues resolved.



**SonarCloud Quality Gate Passed :**

