**TEST PLAN**

**I) Overall test plan**

Our web application has two main functionalities: elastic search and project management. Our plan is to test each functionality separately first and after that, test the whole application in form of a prepared scenario. With elastic search, our test will be divided based on the query input. Input will be varied from blank input, short input, input with complicated medical term, input with special characters, etc. With project management, we focus on testing whether the privilege assigned for each member is correct. In other words, a member of a project should only be able to see what he/she is allowed. With the final test, we will prepare a realistic scenario of our users without revealing it to developers. We will execute the scenario to see if there are any bugs.

**II) Test case descriptions**

ES1.1: Elastic Search Test 1

ES1.2: The test is to ensure that the elastic search works correctly

ES1.3: The test will run the elastic search with blank query

ES1.4: The input will be a blank query

ES1.5: The expect result is the elastic search should not allow the user to carry on the search

ES1.6: Abnormal

ES1.7: Blackbox

ES1.8: Functional

ES1.9: Unit test

ES2.1: Elastic Search Test 2

ES2.2: The test is to ensure that the elastic search works correctly

ES2.3: The test will run the elastic search with short, simple but normal input

ES2.4: Each input will be one word only. Those words are: “japan”, “covid”, “usa”, “vaccine”

ES2.5: The expect result is the elastic search should return the articles respective to keywords

ES2.6: Normal

ES2.7: Blackbox

ES2.8: Functional

ES2.9: Unit test

ES3.1: Elastic Search Test 3

ES3.2: The test is to ensure that the elastic search works correctly

ES3.3: The test will run the elastic search with complicated medical term. Despite their lengths, those words will be expected normal inputs for our project

ES3.4: Each input will be a composed and medical term. Those words are: “covid-19 cardiomyopathy anticoagulation”, “Severe acute respiratory syndrome”, “Droplet transmission”

ES3.5: The expect result is the elastic search should return the articles respective to keywords

ES3.6: Normal

ES3.7: Blackbox

ES3.8: Functional

ES3.9: Unit test

ES4.1: Elastic Search Test 4

ES4.2: The test is to ensure that the elastic search works correctly

ES4.3: The test will run the elastic search with query having special characters

ES4.4: Each input will be word with special characters only. Those words are: “” (double quotation mark), ?, \t, \n.

ES4.5: The expect result is the elastic search should return zero article

ES4.6: Boundary

ES4.7: Blackbox

ES4.8: Functional

ES4.9: Unit test

ES5.1: Elastic Search Test 5

ES5.2: The test is to ensure that the elastic search works correctly

ES5.3: The test will run the elastic search with query having special characters

ES5.4: Each input will be the Pubmed ID of a specific article

ES5.5: The expect result is the elastic search should return the article with such ID or duplicated article

ES5.6: Normal

ES5.7: Blackbox

ES5.8: Functional

ES5.9: Unit test

PM1.1: Project Management 1

PM1.2: The test is to ensure that project management works correctly

PM1.3: A person will create a project and send out invitation of participation to other members

PM1.4: The project creator will input project name, project description, and list of invited members together with their role assignments

PM1.5: The newly created projects with specified name and description should be visible to the creator and the participants. All of them should receive notifications

PM1.6: Normal

PM1.7: Blackbox

PM1.8: Functional

PM1.9: Unit test

PM2.1: Project Management 2

PM.2.2: The test is to ensure that the system of project privilege works correctly

PM2.3: People with different roles (creator, admin, contributor) will try to edit the participated project through our interface and our API (assume our API syntax is leaked to our users)

PM2.4: Everybody in the project will send the request to modify the project information with specified parameters (new project name, new project description)

PM2.5: Creator should be able to modify the project. Request from admins and contributors should be invalidated. The change should be visible to everybody

PM2.6: Normal

PM2.7: Blackbox

PM2.8: Functional

PM2.9: Unit test

PM3.1: Project Management 3

PM.3.2: The test is to ensure that the system of project privilege works correctly

PM3.3: People with different roles (creator, admin, contributor) will try to create repositories through our interface and our API (assume our API syntax is leaked to our users)

PM3.4: Everybody in the project will send the request to create repository in a shared project

PM3.5: Only creator and admin should be able to create repositories. Request from contributors should be invalidated. The change should be visible to everybody

PM3.6: Normal

PM3.7: Blackbox

PM3.8: Functional

PM3.9: Unit test

CST1.1: Composed Scenario Testing

CST1.2: The test is to simulate how our application will work and react on normal environment with multiple interactions and searches

CST1.3: The test is a script of prepared scenario on our application.

CST1.4: The script will be prepared by designer team and not revealed to the developer team. However, it could be described as the combination of all operations from ES1 – ES5 and PM1 – PM3 but in a more realistic structure

CST1.5: The script should be executed smoothly without any bugs/difficulties.

CST1.6: Normal

CST1.7: Blackbox

CST1.8: Functional

CST1.9: Integration test

SQL1.1: SQL Test

SQL1.2: The test is to test how our current SQL queries perform in term of time and network traffic

SQL1.3: We will measure the performance of all SQL calls in our application. We will combine the results with our predicted visitors to determine whether our server is sustainable

SQL1.4: A sample database will be provided as input. The parameters for each call will be also provided. The output will be the performance metrics we decided (time, network traffic)

SQL1.5: The performance metrics should be recorded and calculated

SQL1.6: Normal

SQL1.7: Whitebox

SQL1.8: Performance

SQL1.9: Unit test

**III) Test case matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Normal/abnormal** | **Blackbox/Whitebox** | **Functional/Performance** | **Unit/Integration** |
| ES1 | Abnormal | Blackbox | Functional | Unit |
| ES2 | Normal | Blackbox | Functional | Unit |
| ES3 | Normal | Blackbox | Functional | Unit |
| ES4 | Boundary | Blackbox | Functional | Unit |
| ES5 | Normal | Blackbox | Functional | Unit |
| PM1 | Normal | Blackbox | Functional | Unit |
| PM2 | Normal | Blackbox | Functional | Unit |
| PM3 | Normal | Blackbox | Functional | Unit |
| CST1 | Normal | Blackbox | Functional | Integration |
| SQL1 | Normal | Whitebox | Performance | Unit |

**IV) Results**

ES1: Passed – the system does not carry on the search

ES2: Passed – the system returns 30 articles of articles containing the simple keyword

ES3: Passed – the system returns 30 articles of articles containing the complicated query

ES4: Failed – the system crashed when the user input special characters

ES5: Not tested – this feature is removed from the most recent version of COVID-IQS

PM1: Passed – Both members could see the project at their end

PM2: Passed – Creator has the option to modify the project. Admin and contributor do not have that options

PM3: Passed – Creator and admin have the option to modify the repository. Contributor does not have that options

CST1: Passed – The planned script was executed on our system

SQL1: Passed – an SQL query usually takes 2 seconds to query our huge article database. That is acceptable to us