4 Testing

Some of the unique testing challenges in our project include the complexity of testing communications between the frontend and backend. As well as testing the accuracy of our text extractor module. Below is our testing plan to handle these challenges.

4.1 Unit Testing

As shown in our design diagrams in section 3, there are several distinct modules in our system. Each of these modules have identifiable units of work that are planned to be a part of the final deliverable.

Below is a list of the identifiable elements:

1. Web Application
    a. All components and functionality displayed to the user such as:
        i. Search bar queries and filters
        ii. Proper presentation of results
        iii. Uploading of new documents
2. Tika Text Extractor
    a. Extraction of metadata from newly uploaded files. It should be able to handle all of the file types specified in our constraints (docx, pdf, pptx, etc.)
    b. Be able to identify file types and determine what pieces of metadata to send to Elasticsearch. (e.g. text body for pdf, resolution for jpg, etc.)
3. Elasticsearch Handler
    a. Proper indexing of the metadata from Tika
    b. Searching of nodes based on queries received from the web application

Below is a list of tools that will help us test the above elements:

1. Web Application
    a. We plan on using Jest and React Test Library packages for unit testing. Since they are React's recommended testing tools, they will integrate well into the project. Jest will allow us to see the code coverage of our tests and will allow us to mock data inputs. React Testing Library provides the virtual DOMs for testing React components without a browser.

2. Tika Text Extractor
    a. We plan on using JUnit for unit testing. Since the primary package we are using to support this component is a Java package, JUnit gives us the ability to easily test our code. JUnit is also well documented and will integrate into our development environment easily.

3. Elasticsearch Handler

    a. Since the primary purpose of this component is to handle communication between components see 4.2 Interface Testing for more information.

4.2 Interface Testing

There are several identifiable levels of interface testing:

1. Web Application to Elasticsearch Handler
    a. Is responsible for fetching and retrieving query data
2. Web Application to Tika Text Extractor
    a. Is responsible for sending a selected file to be extracted by the Tika Text Extractor component
3. Tika Text Extractor to Elasticsearch Handler
    a. Is responsible for inserting a files metadata and text content into Elasticsearch

Elasticsearch provides a REST API for inserting data. We plan on using Postman to validate data insertions into Elasticsearch are successful. To test our components' interaction with Elasticsearch, interface 1 and 3, we plan on using Jest and JUnit, respectively.

4.3 Integration Testing

Integration testing will be critical for our project as we will have several components communicating with each other through various endpoints. Incremental integration testing will be the approach for this project, allowing us to test individual modules prior to integrations. It will be important to test and validate the functionality between the elasticsearch endpoint and the tika application as well as the integration between the frontend and elasticsearch; this will be the core of the application. Additionally, if the group reaches the stretch goal, integration of the application into a cloud-based environment would require additional integration testing as there would be additional components to take into consideration during development(VPC, Cloud DB, etc.). However, our project will primarily be focused on the local implementation prior to integrating with the cloud. As this project serves as a proof-of-concept, we anticipate that Buildertrend will also be integrating this product into their own application, implying additional testing during this stage.

4.4 System Testing

Considering that our overall system is composed of 3 major components with minimal communication among them, our system testing should be fulfilled by our unit and integration tests. The individual components should be tested via the unit tests to ensure that they are functional before testing the communication among them. Following the unit tests, there are three interprocess communications which are specified in the interface testing section that need to be tested. These communications will be tested using the methods specified in the integration testings section which will complete our system testing.

4.5 Regression Testing

It is important that new additions made to the system shouldn't break it and instead enhance the user experience. Our most essential features are web application, elasticsearch handler, and tika text extractor and we need to make sure they are robust enough. For this, we would do enough unit and integration testing on our critical components and then add a new feature and see if we still get the same results. This way we can have our core system to be robust and can easily figure out what new feature is breaking the old functionality.

4.6 Acceptance Testing

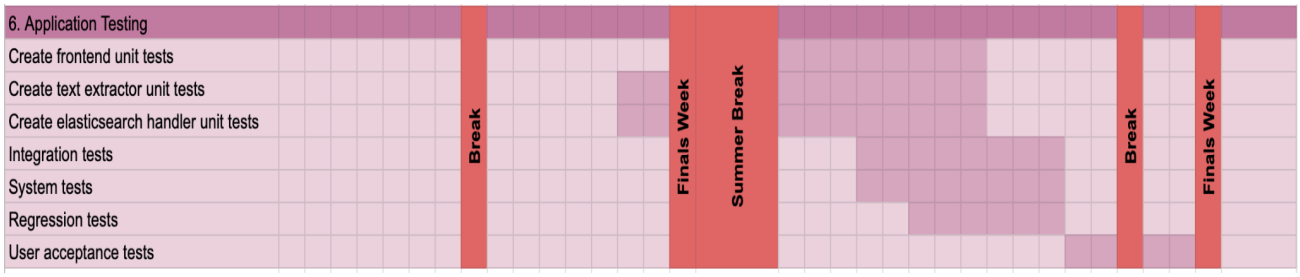
Requirements agreed upon by the client and the team should be fully visible and usable in our minimal viable product - the search web application. We plan on having several iteration of acceptance testing as we expect to receive feedback from the client while making progress toward the final deliverable. The final iteration will include documentation in order to aid users and developers, as well as the working application based on the feedback received, showcasing the requirements and features.

4.7 Security Testing

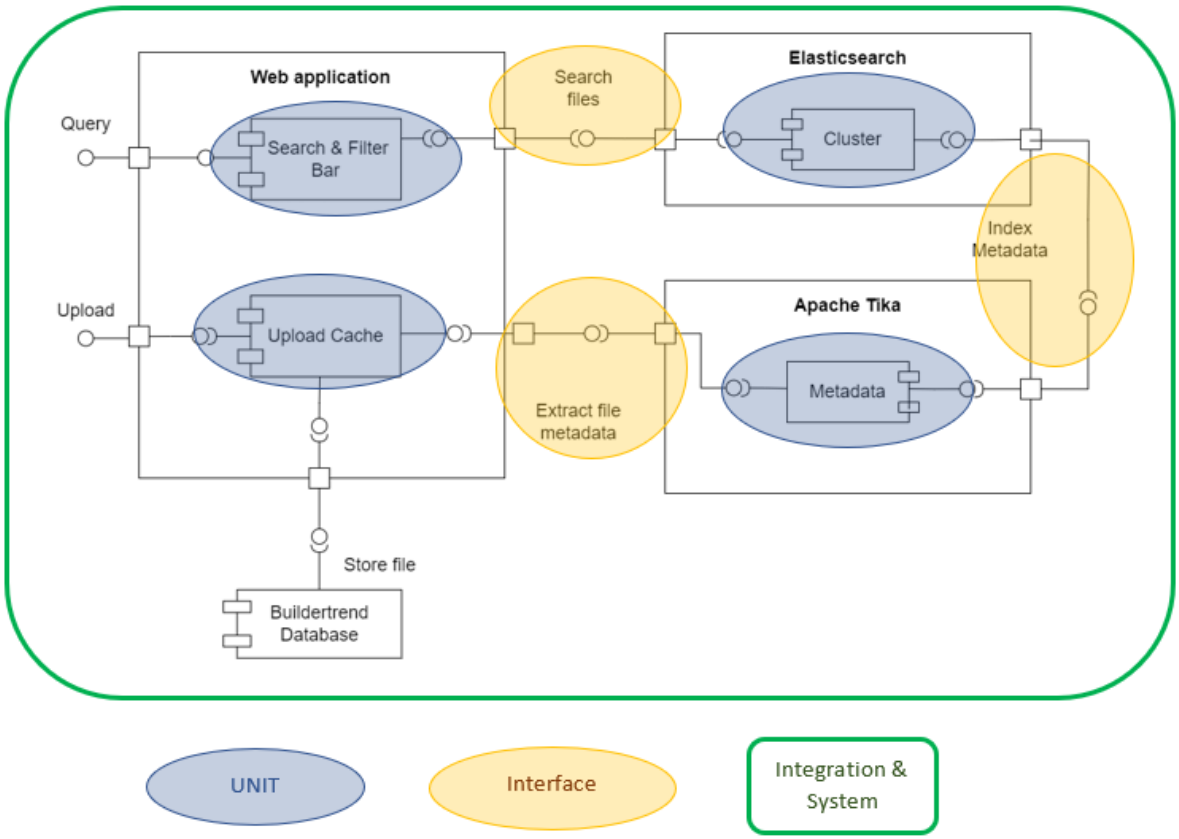
Security testing is now more important than ever, however, it won't be applicable to our project as it's just a proof of concept. We expect BuilderTrend to work on the security when they integrate it into their system.

4.8 Results

We won't have any results this semester as we aren't performing any tests (we may create some unit tests). In the next semester, we shall follow this schedule.



Here is an overview of how the testing strategies overlay the project's components.


Web application
Query
Search & Filter Bar
Upload
Upload Cache
Store file
Buildertrend Database
Search files
Elasticsearch
Cluster
Index Metadata
Apache Tika
Metadata
Extract file metadata
UNIT
Interface
Integration & System