# Test Strategy Document

[Reddit for Research]
[Team 41]

## Scope

**Who will review the document?**
The scope of the document ranges from the members of the development team to the client and the members of his team. The document will be reviewed by the members of the team along with the client before the testing is taken forward.
This is done so that all members of the team will be on the same page.

**Who will approve this document?**
The document will be approved by the client without whose consent, the given testing approaches will not be implemented.

**Testing activities carried out with timelines**
The timeline for the testing activities which have been carried out has been listed in the Test Plan Document. This document will focus on the tests carried out throughout, with a focus on which aspects of each component were tested.

Unit testing was carried out in earlier sprints, and as the R2 deadline approached, the focus was shifted to integration testing. In the final days of the project, emphasis was laid on system testing, to ensure that no high-severity faults existed prior to the product's release and demonstration.

## Test Approach

### Process of testing
The process of most of the testing systems in place will follow one streamlined pipeline. The generalized process is listed below -

- Build a fully functioning unit(module) as described in SRS.
- Apply unit testing to ensure smooth functioning of the module
- Apply integration testing systematically.
- Apply system testing on the complete module.
- Carry out Acceptance Testing.

Categories of testing:

- Unit testing (ensure code is developed correctly)
- API testing (ensure communication between components works correctly)
- Acceptance testing (to ensure customer's expectations are met)
- Integration testing (to ensure the whole system works when integrated)

Testing approach is manual, and every team member is involved in the same as prescribed by the agile-scrum methodology. Every member working on a feature will take it upon himself to ensure smooth functioning and integration of said feature.

## Examples of tests

- Security Testing on multiple fronts:
    - Hashing the passwords (SHA-256)
    - JWT and Bcrypt authentication
- Performance Testing:
    - Checking multi-user PDF viewer
    - Ensuring document upload time isn't severely impacted by internet speed
    - Benchmarking document retrieval from database, comparing different API requests and database fetch options for best retrieval times
    - Fine-tuning elastic search parameters for optimal search times based on database size
- Accuracy Testing: Checking accuracy of search results (along with advanced search options)

Tests applied across all components of the webapp include:

- **Link tests:** Ensuring that all the outgoing, internal, anchor, and mailTo links in the webapp redirect to the appropriate components
- **HTML/CSS/JS:** Ensuring no syntax errors and console warnings, presence of readable indented blocks of code, and appropriate versioning of NPM packages
- **Error workflows:** Simulating erroneous workflows to ensure appropriate error handling and redirection

## Regression Testing

Regression Testing is a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression testing is used on a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

# Test Environment

- PDF Upload/Viewer:
    - Multer
    - PDF.js
- Annotations/comments/highlight:
    - Hypothes.is client tool (integrated into code)
- Web App:
    - MongoDB
    - Express.js
    - Angular8
    - Node.js
    - Elastic Search

**Backup and Restore Strategy**

As the data is stored using MongoDB, backup can be done on a MongoDB cloud server called Atlas. This will ensure that all the data is safely preserved on the cloud having no risk of local storage problems.

The restore strategy incase of any massive failures will be using Git, since we used GitLab as we completed the project step-by-step. Therefore, in the occurrence of a failure we can revert back to a safe git state and continue working from there.

# Testing Tools

**Tools needed for test executions**

- Postman - required for testing backend API endpoints
- Selenium - testing framework to perform web application testing, supports multiple OSs and multiple web-browsers
- Loadrunner - Load/performance testing tool

# Use Cases

- User Management - user account creation, deletion, and modification. User authentication during sessions (login)
- Profile view - users can view other users' profiles, along with all the documents they have uploaded
- Posting - user posting articles/PDFs/research papers

- PDF viewer - user views uploaded articles/documents/papers in the form of an inbuilt PDF viewer
- Categories - documents are automatically sorted into categories based on the ‘field entered during upload
- Annotations/comments - user highlights particular subsections of article and annotates them
- Search - user searches through annotations and comments to find relevant results
- Reply - user should be able to reply to specific annotations and comments
- Editing - users can edit comments/annotations/replies that they posted
- Deleting - users can delete comments/annotations/replies that they posted

# Test Cases

- **User Management**
  - Check registration and login authentication APIs
  - Check registration and login authentication frontend (in different web browsers and operating systems)
  - Ensure access is granted to other components only if session is authenticated
- **Profile View**
  - Check profile information retrieval API
  - Check profile view frontend - ensure appropriate redirection on clicking a username
  - Repeat for multiple profiles
- **Posting**
  - Check upload API
  - Check upload frontend (in different web browsers and different operating systems)
  - Ensure different fields (DoB, email, username, etc) are stored in appropriate formats and indexed appropriately in database+ElasticSearch
  - Repeat tests for different file sizes
- **PDF Viewer**
  - Test to see if files of different sizes are displayed and rendered correctly
  - Ensure that switching between native PDF.js and Hypothes.is-infused PDF.js is seamless
  - Check fullscreen functionality across browsers and OSs

- **Categories**
    - Ensure correct categorization and retrieval
    - Check buttons to ensure correct link between FormGroup and APIs
- **Annotations**
    - Check annotation API - check if annotations and comments are stored corresponding to correct post
    - Check frontend - ensure correct selection of text to annotate
    - Repeat across multiple posts
    - Repeat multiple annotations and comments on same post
- **Search**
    - Check search API - if JSON retrieved from API corresponds to accurate search results, on passing of input string
    - Check search bar and frontend - if results are retrieved and displayed appropriately
    - Check if search results are consistent across multiple searches of same input string
    - Ensure advanced options are performing as expected
- **Reply**
    - Test reply API - check if replies are stored corresponding to correct initial post
    - Test reply frontend - ensure reply button corresponds to correct post/comment, and calls correct API
    - Repeat multiple replies on same post
    - Repeat across multiple posts

- **Deletion and editing**
    - Test delete API - check if it removes correct elements from database, while preventing any referential integrity issues in database upon removal
    - Test edit API - check if it edits correct element in database
    - Test frontend for deletion and editing - check if the correct corresponding element is deleted/edited