1. ***Test Prioritization:***

1. Authentication Flows (Sign-up, Sign-in, Password Reset):

Priority: Critical

Reasoning:

- Authentication is fundamental to user security and experience.
- Anomalies in sign-up, sign-in, or password reset functionalities can affect in a bad way the user experience and introduce security vulnerabilities.
- Ensuring a good authentication flow is necessary to project a sense of stability and credibility of the application/platform.

2. Feeds (Loading and Pagination for User's Feed and Global Feed):

Priority: High

Reasoning:

- Feeds are the main feature of the application, the functionalities of these will impact user engagement profoundly.
- Challenges in loading and pagination directly impact user satisfaction and the perceived performance of the platform.
- Is a high priority to maintaining a smooth content flow for an enjoyable user experience.

3. Articles (Creating, Editing, Reading, and Deleting):

Priority: High

Reasoning:

- Alongside with the feeds, articles are part of the core content of the application/platform.
- Any discrepancies in creating, editing, reading, or deleting articles could lead to data loss, frustration for authors, and an overall poor user experience.
- All article-related features are of high importance for both content creation and consumption.

4. Following Authors (Following, Feed Update, Unfollowing):

Priority: Medium

Reasoning:

- Following authors enriches the user experience but may not carry the same criticality as authentication, feeds, or articles.

- Bugs in this area could affect user engagement, although without immediate severe consequences. Main functionalities of the application would still work as expected.
- Assigning it a medium priority ensures an enhanced user experience without compromising on critical functionalities.

5. Comments (Creating, Reading, Deleting):

Priority: Medium

Reasoning:

- Comments contribute significantly to user interaction, although their criticality may be less than other functionalities.
- Complications with comments can impact user engagement but may not result in immediate severe consequences.
- For most “forum” types of applications a big part of the users-base are trying to find answers instead of giving them.

6. API Contracts:

Priority: Low

Reasoning:

- API contracts form the backbone for system integration but might not carry the same immediacy for end-users.
- While integral to the system, API contract issues may not have a direct and immediate impact on end-users.
- Categorizing it as a low priority allows for a strategic focus on user-facing features, ensuring a better user experience and engagement.

2. Test Plan:

Authentication Flows (Sign-up, Sign-in, Password Reset)

Test Case ID	Steps	Expected Results	Criticality	Impact
AUTH_TC_01	1. Navigate to the sign-up page.	1. Sign-up form is displayed.	Critical	Critical
	2. Fill in valid details and submit.	2. User is registered successfully. 3. User is redirected to the home page.	Critical	Critical

AUTH_TC_02	1. Navigate to the sign-in page.	1. Sign-in form is displayed.	Critical	Critical
	2. Enter valid credentials and sign in.	2. User is logged in successfully.	Critical	Critical
AUTH_TC_03	1. Request a password reset.	1. Password reset email is sent.	High	High
	2. Follow the link in the email and reset.	2. Password is successfully reset.	High	High

Feeds (Loading and Pagination for User’s Feed and Global Feed)

Test Case ID	Steps	Expected Results	Criticality	Impact
FEED_TC_01	1. Navigate to the global feed.	1. Global feed loads without delay.	Critical	Critical
	2. Click one tag.	2. Content is filtered by the selected tag. 3. Chosen tag shows next to the “global feed” tab, indicating that tag is being used by filters.	High	High
	3. Scroll through pages.	4. Pagination works without any type of bugs.	High	High
FEED_TC_02	1. Navigate to the user's feed.	1. User's feed loads smoothly.	Critical	Critical

	2. Check for smooth pagination.	2. Global feed pages load correctly.	High	High

Articles (Creating, Editing, Reading, and Deleting)

Test Case ID	Steps	Expected Results	Criticality	Impact
ARTICLE_TC_01	1. Create a new article.	1. Article is created and visible in the feed.	High	High
ARTICLE_TC_02	1. Edit an existing article.	1. Changes are saved and reflected.	High	High
ARTICLE_TC_03	1. Read an article.	1. Article content is displayed correctly.	High	High
ARTICLE_TC_04	1. Delete an article.	1. Article is removed from the platform.	High	High

Following Authors (Following, Feed Update, Unfollowing)

Test Case ID	Steps	Expected Results	Criticality	Impact
FOLLOW_TC_01	1. Follow a new author.	1. Author is successfully followed.	Medium	Medium
FOLLOW_TC_02	1. Check for feed update after following.	1. User's feed includes updates from the author.	Medium	Medium

FOLLOW_TC_03	1. Unfollow an author.	1. Author is successfully unfollowed.	Medium	Medium

Comments (Creating, Reading, Deleting)

Test Case ID	Steps	Expected Results	Criticality	Impact
COMMENT_TC_01	1. Create a new comment.	1. Comment is visible under the article.	Medium	Medium
COMMENT_TC_02	1. Read existing comments.	1. Comments are displayed correctly.	Medium	Medium
COMMENT_TC_03	1. Delete a comment.	1. Comment is removed from the article.	Medium	Medium

API Contracts

Test Case ID	Steps	Expected Results	Criticality	Impact
API_TC_01	1. Validate API contracts for integration.	1. Contracts align with specifications.	Low	Low

3. ***Testing Strategy:***

- Regression Testing Landscape

Authentication Flows (Sign-up, Sign-in):

Execution Pipeline: High
Reasoning: Authentication is critical, and any issues here directly impact user experience and security. Include these tests in the main execution pipeline.

Feeds (Loading and Pagination for User's Feed and Global Feed):

Execution Pipeline: High

Reasoning: High impact on user satisfaction and platform performance. Included in the main execution pipeline.

Articles (Creating, Editing, Reading, and Deleting):

Execution Pipeline: High
Reasoning: Core content creation and consumption features. Included in the main execution pipeline.

Following Authors (Following, Feed Update, Unfollowing):

Execution Pipeline: Medium
Reasoning: Enhances user experience but may not be as critical as authentication or core functionalities. Can be included in a secondary pipeline.

Comments (Creating, Reading, Deleting):

Execution Pipeline: Medium
Reasoning: Significant for user interaction but may not have immediate severe consequences. Included in a secondary pipeline.

API Contracts:

Execution Pipeline: Low
Reasoning: Vital for system integration but doesn't directly impact end-users. Included in a separate pipeline, considering it has a low priority.

- Techniques to Reduce Regression Tests:

Risk-based Testing: Focus on critical and high-impact areas, reducing the number of tests for low-priority features. Prioritize tests based on business-critical functionalities and potential impact on users.

Selective Testing: Prioritize tests that cover the most common user scenarios.

Test Impact Analysis: Identify impacted areas after code changes and execute only tests related to those areas.

- Multi-Browser Environment:

Authentication Flows: Ensure users can sign up, sign in, and reset passwords across different browsers.

Feeds and Articles: Check for consistent rendering and pagination on various browsers.

Following Authors and Comments: Verify that social interactions work seamlessly on different browsers.

4. ***Postman Collection:***

Some of the test cases in folders “Articles” and “Articles, Favorite, Comments” seem to be duplicated. So I would suggest checking those and erase whatever tests are duplicated. Also would be preferable to separate test cases in more folders, folder “Articles, Favorite, Comments” seems to have multiple test cases that are not necessarily related to each other so could be located in separate folders. From a brief glance and the test suite it doesn’t seem to have negative test cases which could also be added to increase testing coverage.

In case some of the tests should be included I would select the ones that are critical to the business:

1. Auth test scenarios.
2. Creating/Editing/Reading/Delete of articles.
3. Creation of new users.

5. ***QA Road Map:***

https://roadmap.sh/qa?s=65a6b8d28240aa21a4a6ac47

For handling website tests, I'd suggest going with Playwright. It's a really cool tool that makes automating e2e intuitive and smooth. You can run tests on different browsers and devices, test multiple windows and tabs and also comes with the possibility of doing API automation parallel to UI Automation.

When it comes to testing APIs, Postman is still one of the best options out there. It's user-friendly, and you can easily manage and test your APIs. Plus, it supports scripting in case there are more complex scenarios that need to be taken into account.

For assertions library I would suggest Chai since it comes along with playwright. Also Chai library supports multiple styles, including “expect”, “should”, and “assert”, it has a lot of flexibility that allows each developer to choose a style that aligns with his/hers preferences.

And to keep things organized, I would suggest Jira Xray because it integrates smoothly with Jira, providing a nice setup for managing our tests and keeping track of any bugs we might find. This way, everything's in one place.

6. ***Shift left or shift right?:***

The “trend” of Shift-Left has gained popularity for multiple reasons. In my opinion it is more of a proactive approach, preventing defects early which ends up saving time and resources. Also the Shift-Left practices align well with Agile and DevOps practices but it's not necessarily about choosing one over the other. Both Shift-Left and Shift-Right have their pros and cons, the key is finding the right balance based on the project's needs, the team's dynamics, and the overall company philosophy when it comes to development. All of these “frameworks” we hear of (TDD, SCRUM, KANBAN, Shift-left, BDD, etc..) have a definition of their own but each company should adapt them for their specific projects.