Software Testing Analysis

1. Acceptance Testing

To verify whether our client’s specifications for the project were met, we checked the working of 4 major functions/units

* Authenticating Users
* Verifying URL/Links of Articles as Valid/Not
* Running an ensemble ML model to output the validity of an article, whether fake or not
* To show reviews of other users for an article

When we finally displayed the completed webapp to the client, he was very happy with the result and took some time to check all the functionalities and vaildity of the results of the ML models, and sent us a mail expressing his satisfaction on the delivery

2. System Testing

Our system is made of 4 major parts, the UI, the users & articles databases, the scrapper & the ML models. On integrating them all together and deploying the ML models, we checked if all the parts worked in unison and were compatible as we were using a Django framework with angular.js for the frontend and the multiple kinds of ML models we kept updating along with the scrapper. This required multiple system tests as we went on improving our design patterns, ML algorithms & Neural networks.

3. Integration Testing :

As our project consisted of multiple modules, coded by different programmers, integration testing was an integral part of our testing process, as we had to check for multiple possible defects that could arise due to interaction of two modules with another.

The Big-bang test combined with the Bottom-UP approach made more sense for this project as a complete cycle of the main function, i.e to post a review/get the review of an article requires integrating all the 4 components - UI, URL data scraper, the on-the-fly deployed ML models & the backend serving to & from the database, and they need to work in unison.

We could easily find bugs in our code through big-bang testing as wherever throughout the lifecycle of a user’s visit to the website, we’d go lookup in that particular module, as we had already fixed all the localised faults, during the bottom-up testing while working on the individual modules.

While bottom-up testing gave us a closer understanding from the lowest level of abstraction, Big-bang helped us fix the final prototype and how it all worked when put together, as that's what matters most. Deploying the site was a breeze after the integration testing as we had taken care of most of the drawbacks beforehand itself.

4. Unit Testing

We have 4 main units that perform the main 4 functions respectively in the project which are all meant to be tested individually. We tested them all as we went by coding them, as it was a parallel process, as we checked the functionality of each small component as we coded it up, with debugging tools.

The purpose was to validate that each unit of the software performs as designed. We broke them down into the most smallest testable parts and tested accordingly.