Test Plan for Recommendation and Review Subsystem

**Introduction**

This test plan outlines the approach and strategy for testing the Recommendation and Review subsystem within the Online Marketplace Platform (OMP). The subsystem is responsible for collecting, processing, and utilizing reviews and ratings from third-party websites and social media platforms to enhance product recommendations and in-house product rankings. Given the large volumes of data and the flow through various technologies involved, testing will focus on data integrity, scalability, and accuracy.

**Test Objectives**

* Verify the correctness of data collection from third-party sources.
* Ensure the accuracy of data processing and extraction.
* Assess the scalability and performance of data storage and retrieval.
* Validate the functionality of API interfaces within the OMP system.
* Fitness of the recommendation Algorithm.

**Scope**

The test plan covers the testing of the Recommendation and Review subsystem, including data collection, processing, storage, and API interfaces. It does not cover testing of the web crawling engine itself.

**Test Approach**

1. **Third-Party Data Collection Validation**:

* Emphasize the testing of the web crawling engine's ability to handle and collect large volumes of data.
* Create test scenarios with data volumes significantly larger than expected in production.
* Monitor memory and CPU utilization during data collection for potential bottlenecks.
* Test for data deduplication to avoid collecting duplicate records when dealing with extensive datasets.

1. **Data Extraction Accuracy**:

* Assess the efficiency of post-processing steps in handling large datasets.
* Create test cases that process and extract data from a substantial dataset.
* Measure the time taken for data extraction and verify that it remains within acceptable limits.
* Verify that data quality is maintained even when processing large volumes of data.

1. **Data Warehousing Testing**:

* Perform stress testing on data warehousing solutions like Google Big Query with datasets of varying sizes.
* Test the scalability of storage infrastructure by gradually increasing data volumes.
* Evaluate the performance impact of indexing strategies, especially when dealing with large tables.
* Assess data retrieval times from transactional databases with significant data loads.

1. **Transactional Database Testing**:

* Execute API interface tests using large datasets as inputs.
* Measure API response times and resource utilization while handling significant data loads.
* Simulate concurrent API requests to assess performance under load.
* Evaluate the effectiveness of API caching mechanisms, especially for frequently requested large datasets.

1. **Internal API Interface Testing**:

* Verify that API interfaces exposed internally within the OMP system provide accurate and timely access to collected data points.
* Test API response times and data consistency.

1. **Test Data**

* Create synthetic test datasets that mimic the characteristics of real-world data with varying volumes.
* Include edge cases where data size exceeds typical operational levels.
* Generate datasets that span different data types (text, numerical, multimedia) to test the versatility of data handling.

1. **Test Environment**

* Allocate sufficient resources in the test environment to replicate production-scale data handling.
* Implement load balancing and clustering solutions for data storage and processing components.
* Utilize performance monitoring tools to capture resource utilization metrics.

**Test Execution**

1. **Functional Test**:

* **Verification of Basic Functionality**: Start with basic functional tests to ensure that the recommendation system is correctly integrated into the website. This includes verifying that recommended products are displayed to users at the appropriate locations on the site.
* **Interaction Testing**: Verify that user interactions trigger recommendations. For example, if a user views a product or adds an item to their cart, the recommendation system should respond by suggesting related or complementary products.
* **Personalization**: Test whether the recommendation system is personalized to individual users. Use multiple user profiles to confirm that recommendations differ based on user behavior and preferences.
* **Content Validity**: Ensure that recommended products are relevant to the context. Verify that recommendations align with the content or products on the page where they are displayed.
* **Error Handling**: Test how the system handles edge cases and errors, such as when there are no recommendations available or when there's a failure in the recommendation algorithm. Verify that appropriate error messages or fallback recommendations are displayed.

1. **Data-Driven Evaluation**:

* **Offline Evaluation**: Conduct offline evaluations using historical user data. This involves running the recommendation algorithms on past user interactions to see how well they would have performed. Use metrics like precision, recall, and mean average precision to assess the quality of recommendations.

1. **Online Evaluation**: Implement online A/B testing to evaluate the recommendation system's performance in a live environment. Deploy multiple versions of the recommendation system with different algorithms or configurations and measure user engagement, conversion rates, and other relevant metrics.
2. **User Feedback Analysis**: Collect and analyze user feedback related to recommendations. Pay attention to user reviews, comments, and ratings to understand how users perceive the recommendations and whether they find them helpful.
3. **Bias and Fairness Assessment**:

* Evaluate the recommendation system for biases, especially those related to gender, age, race, or other sensitive attributes. Ensure that recommendations are fair and do not discriminate against specific user groups.
* Implement fairness-aware metrics and techniques to measure and mitigate biases in recommendations.

1. **Exploratory Testing**:

* Conduct exploratory testing to explore different user profiles and scenarios. Test the recommendation system across various user types, use cases, and geographic regions to identify potential issues or limitations.

1. **Performance Test**

* Gradually increase data volumes in test scenarios to assess scalability.
* Continuously monitor system behaviour and resource utilization during testing.
* Pay special attention to memory consumption, CPU usage, and database response times as data sizes grow

**Acceptance Criteria**

* Data Collection Validation:
  + All data collected from third-party sources is accurate and complete.
* Data Extraction Accuracy:
  + Extracted data points match expected values and formats.
* Data Warehousing Testing:
  + Data storage in analytical databases scales to accommodate large volumes.
  + Data integrity is maintained during storage.
* Transactional Database Testing:
  + Data transfer to transactional databases is accurate and efficient.
  + Data retrieval from transactional databases is timely and accurate.
* Internal API Interface Testing:
  + API interfaces provide access to collected data points within acceptable response times.
  + Data consistency is maintained across API requests.

**Defect Management**

* Document and prioritize defects discovered during testing.
* Collaborate with development teams to address and resolve defects promptly.
* Verify defect resolutions through regression testing.

**Test Reporting**

* Generate test reports to document testing activities, results, and any identified issues.
* Provide regular status updates to project stakeholders.

**Evaluation Metrics**

Evaluating the performance of a recommendation system is crucial for its success, as it helps identify areas for improvement and ensures that it meets user needs.

1. **Accuracy Metrics**

Accuracy metrics measure how well a recommendation system can predict user preferences. They are generally applicable to collaborative filtering methods, which leverage similarities between users or items to make recommendations.

* **Mean Absolute Error (MAE)**: Calculates the average absolute difference between predicted and actual ratings.
* **Precision**: Measures the proportion of relevant recommendations out of all the recommended items.
* **Recall**: Measures the proportion of relevant recommendations out of all the relevant items.

1. **Ranking Metrics**

Ranking metrics assess the quality of the ranking of recommended items, ensuring that the most relevant items appear at the top of the list.

* **Mean Reciprocal Rank (MRR)**:
  + Calculates the average of the reciprocal ranks of the first relevant recommendation for each user
  + MRR takes into account the position of the first relevant item in the recommendation list, rewarding systems that rank relevant items higher.
  + It the mean reciprocal rank across all users, this allows for a fair comparison of different algorithms or models, as it considers the average performance rather than specific user cases.
* **Mean Average Precision (MAP)**:
  + Calculates the average precision for each user and takes the mean across all users. It takes into account both the order and the relevance of recommended items.
  + MAP considers the relevance of items by calculating the average precision for each user, which is the average of the precision scores at each relevant item's position.
  + This means that it can distinguish between different levels of relevance when evaluating recommendations.
* **Normalized Discounted Cumulative Gain (nDCG)**:
  + Evaluates the ranking quality by assigning higher importance to relevant items appearing at the top of the recommendation list. It is normalized to ensure comparability across different users and queries.
  + nDCG is normalized against the ideal ranking, which means it can be compared across different queries or users. This allows for a fair evaluation of the recommendation system's performance, even when the number of relevant items varies between users or queries.

1. **Coverage and Diversity Metrics**

Coverage and diversity metrics measure the extent to which a recommendation system can provide a wide range of relevant and novel items, promoting exploration and discovery.

* **Catalog Coverage**:
  + Measures the proportion of items in the catalog that are recommended at least once.
  + A high catalog coverage indicates that the recommendation system is capable of suggesting not only popular items but also less popular or long-tail items.
  + Catalog coverage can help identify the cold-start problem, where the recommendation system struggles to recommend new or less popular items due to a lack of data. A low catalog coverage might indicate that the system is not well-suited for handling such situations, and alternative approaches or additional data sources should be considered.
  + For businesses with a large and diverse catalog, it is essential to ensure that users are exposed to a wide variety of items. A high catalog coverage can contribute to achieving business goals such as increasing sales, user satisfaction, and user retention.
* **Prediction Coverage**:
  + Measures the proportion of possible user-item pairs for which the recommendation system can make predictions.
  + Prediction coverage provides valuable information about the recommendation model's ability to make predictions across the user-item space. It helps identify potential limitations, evaluate model performance, and ensure that the model is well-suited for the intended application.
  + Prediction coverage can also be an indicator of the model's scalability. If a model can make predictions for a large number of user-item pairs, it may be better suited for handling larger datasets or growing catalogs.
* **Diversity**:
  + Evaluates the dissimilarity between recommended items, ensuring that the recommendation list contains a good mix of different types of items.
  + Diversity is important in recommendation systems because it helps ensure personalized and engaging experiences for users, supports exploration, reduces filter bubbles, promotes long-tail items, and enhances the robustness of the system.
  + A diverse set of recommendations is less susceptible to manipulation or bias. By ensuring that a wide variety of items are recommended, the system is more resistant to external factors like spam or targeted promotion of certain items.