الكلية الجامعية للعلوم التطبيقية

رائدة الإبداع

Project #6

Test Results Report

Academic Paper Management System

Software Engineering Project. By:

Hebatallah AbuHarb - 220210448
Shahd El-Thalathini - 220210527
Salma Shaheen - 220210654

Introduction

This document summarizes the results of functional and exploratory testing performed on the Academic Paper Management System. While the Excel file (*QA Test Descriptions Table*) describes tests on details. The goal was to verify that all use cases (UC) related to paper submission, status monitoring, search, recommendations, and alerts work as expected. Testing involved manual, exploratory, and automated approaches where applicable.

Testing Methodology

- **Manual Testing**
  We manually executed predefined test scenarios aligned with each user story, such as submitting papers, updating statuses, searching papers, and managing collections. This helped us confirm that the system's core functionalities and user interface worked as intended.

- **Exploratory Testing**
  Our team performed exploratory testing to uncover unexpected issues, particularly focusing on unstable areas like metadata extraction during submission, concurrency problems during paper revisions, and UI responsiveness in filtering and recommendations.

- **Debugging Tools**
  We utilized browser developer tools, backend logs, and performance profilers to investigate and locate bugs. This enabled us to trace errors both on the frontend and backend, such as race conditions and PDF export timeouts.

- **Logs and User Feedback**
  We analyzed system logs and incorporated user feedback from testing sessions to identify real-world issues, including delayed alerts, inaccurate metadata in notifications, and feedback saving failures in recommendations.

- **Load Testing**
  While we did not conduct formal load testing during this phase, we recommend it for future iterations to ensure the system can handle concurrent paper uploads and updates smoothly without race conditions or performance drops.

Test Results Summary

Use Case	Result Summary
UC-R1	Paper submission mostly successful; metadata extraction needs improvement.
UC-R2	Revision upload functionality affected by concurrency issues.
UC-R3	Paper status viewing stable and reliable.
UC-A1	Monitoring dashboard functional and handles service failures gracefully.
UC-A2	Analytics charts work; PDF export initially timed out but fixed.
UC-A3	Unauthorized access detection working correctly.
UC-A4	User role management functions as expected with proper validation.
UC-A5	Paper status assignment follows business rules with error handling.
UC-S1	Search and reading functionality fixed after addressing filters and preview bugs.
UC-S2	Collection creation and paper addition stable after validation fixes.
UC-S3	Saving and revisiting papers improved with fixes on state and sorting.
UC-S4	Recommendations UI optimized; feedback mechanism repaired.
UC-S5	Alerts on paper updates improved with retry logic and cache fixes.

Bug Analysis

Bug in Test Case UC-R1: Metadata Extraction Failure

Description: Metadata extraction occasionally fails during paper submission, requiring manual metadata entry.
Location/Method: Discovered through exploratory testing when uploads showed missing metadata fields; confirmed via backend logs indicating extraction errors.
Cause: Instability in the metadata extraction algorithm caused by unexpected file formatting variations.
Resolution: Added a manual metadata entry fallback to ensure submissions can proceed. Recommended improving extraction algorithm accuracy and robustness in future updates.

Bug in Test Case UC-R2: Concurrency Issues in Paper Revision Upload

Description: When uploading revised papers, concurrent updates sometimes cause version conflicts or lost updates.
Location/Method: Detected during exploratory testing when multiple revision uploads caused inconsistent version history; backend logs showed race conditions.
Cause: Backend lacked proper concurrency control, leading to race conditions in database update operations.
Resolution: Implemented locking mechanisms and atomic transactions on revision uploads. Recommended conducting load testing to verify stability under concurrent use.

Bug in Test Case UC-A2: PDF Export Timeout

Description: The system failed to export the submission analytics report as a PDF, timing out after 5 seconds.
Location/Method: Identified using the browser developer console showing timeout errors and server logs revealing bottlenecks in the PDF generation library.
Cause: The PDF generation process was overwhelmed by processing a large dataset synchronously without optimization.
Resolution: Optimized the data pipeline by aggregating data points and implemented asynchronous PDF generation. The fix was verified by successful re-testing.

Bug in Test Case UC-S1: Search Filters and Preview Failures

Description: Filters applied during paper search did not work, paper previews failed to load, and the download button was disabled.
Location/Method: Found during manual and exploratory testing; debugging frontend code and backend query logs revealed incorrect query handling and async loading bugs.
Cause: Backend filters were improperly implemented, and the frontend asynchronous preview logic had errors.
Resolution: Fixed backend filter queries, refactored async preview loading code, and corrected download button activation logic.

Bug in Test Case UC-S2: Duplicate Collections and Silent Failures

Description: Users could create duplicate personal collections; form resets improperly and some failures did not show error messages.
Location/Method: Detected through manual UI testing and form behavior observation.
Cause: Missing validation checks for collection names and frontend UI state bugs causing improper form reset and silent API failure.
Resolution: Added collection name uniqueness validation, fixed form reset behavior, and implemented error handling with user notifications.

Bug in Test Case UC-S3: Saved Papers List Not Refreshing and Sorting Issues

Description: The saved papers list did not refresh correctly; sorting options malfunctioned and sometimes the wrong document opened.
Location/Method: Identified via manual and regression testing; frontend state and sorting logic debugging.
Cause: Frontend state management bugs and incorrect sorting algorithm implementation.
Resolution: Fixed frontend state update triggers, corrected sorting logic, and fixed navigation to open correct documents.

Bug in Test Case UC-S4: Stale Recommendations, UI Freeze, and Feedback Not Saved

Description: AI-generated paper recommendations were stale, the UI occasionally froze, and user feedback on recommendations was not saved.
Location/Method: Found through logs analysis and user feedback reports.
Cause: Profile synchronization issues, inefficient rendering code causing UI lag, and broken feedback submission endpoints.
Resolution: Updated profile synchronization logic, optimized UI rendering performance, and repaired feedback API endpoint.

Bug in Test Case UC-S5: Alerts Delayed and Incorrect Metadata in Notifications

Description: Alerts about paper updates were delayed, notification preferences did not always work, and metadata in alerts was sometimes incorrect.
Location/Method: Diagnosed through system logs and user feedback.
Cause: Notification system downtime, outdated metadata cache causing stale data in alerts, and bugs in preference management.
Resolution: Added retry mechanisms for notification delivery, fixed API bugs related to preference handling, and refreshed metadata caches regularly.

Bugs Encountered and Resolution Details

During testing of the academic paper management system, several bugs were identified across various functional areas. These included issues with metadata extraction, concurrency in paper uploads, PDF export failures, search and filtering bugs, duplicate collections, saved papers management, stale AI recommendations, and delayed alerts. Each bug was analyzed, located, and resolved through targeted fixes involving backend improvements, frontend code refactoring, and process optimizations.

Bug Location Methods

- **Manual Testing:** Step-by-step execution of predefined user scenarios to reproduce and observe bugs firsthand.
- **Exploratory Testing:** Unscripted, focused testing especially around edge cases such as metadata extraction failures and concurrency problems.
- **Debugging Tools:** Utilized browser developer consoles for frontend debugging; backend logs and performance profiling to trace server-side errors and bottlenecks.
- **System Logs & User Reports:** Monitored real-time logs and analyzed user feedback to uncover issues related to AI recommendations and alert delivery delays.
- **Regression Testing:** Verified that previously fixed bugs did not reoccur and that system stability was maintained after updates.

Root Causes of Bugs

- Unstable or insufficiently robust third-party/custom metadata extraction tools leading to inconsistent data capture.
- Race conditions stemming from lack of proper locking mechanisms or atomic transactions in backend data handling.
- Large datasets processed synchronously causing timeouts and performance bottlenecks in PDF export and analytics.
- Missing validation checks and improper UI state management on the frontend resulting in duplicates and silent failures.
- Inefficient asynchronous data fetching and rendering logic affecting UI responsiveness and functionality.
- Cache inconsistencies and network interruptions impacting timely notifications and alert accuracy.

Bug Resolution Actions

- Introduced **manual fallback mechanisms** for metadata entry where automated extraction failed, ensuring uninterrupted workflow.
- Enhanced backend with **concurrency controls** such as locking and atomic operations, coupled with recommendations for load testing under concurrent usage.
- Optimized data processing pipelines and switched to **asynchronous handling** for large export tasks to prevent timeouts.
- Strengthened frontend validation logic and **corrected UI state management** to prevent duplicates, enable proper form resets, and surface errors clearly.
- Refactored asynchronous frontend code to ensure **correct loading and sorting** behaviors for search and saved papers features.
- Improved AI recommendation system by updating **profile synchronization** logic and optimizing UI rendering performance.
- Added **retry mechanisms** and implemented **cache refresh** strategies in notification services to guarantee timely and accurate alert delivery.

Conclusion and Recommendations

The Academic Paper Management System has undergone comprehensive testing, demonstrating stable core functionalities and successful handling of key use cases such as login, paper search, downloading, submission, and collection management. Most major features operate as intended, and identified bugs related to filtering, subscriptions, previews, concurrency, and metadata extraction have been effectively debugged and resolved.

However, some challenges remain, primarily in the areas of performance under concurrent usage, metadata extraction accuracy, and UI responsiveness during asynchronous operations.

To further enhance the system’s robustness and user experience, we recommend:

- Enhancing the metadata extraction algorithms by improving current methods or integrating more reliable third-party extraction libraries.

- Conducting formal load and concurrency testing to ensure backend stability and performance under heavy simultaneous use.

- Implementing additional user experience (UX) feedback mechanisms and validation prompts throughout the application to reduce errors and improve user guidance.

- Establishing continuous monitoring of system logs and actively analyzing user feedback to detect and resolve issues proactively, facilitating ongoing system improvements.

With these improvements, the system will be well-positioned for reliable deployment and scalable operation in academic environments.