Project Documentation

Testing of Educational Platform Admin Account

Groub code:

GHR1_SWD7_M1e

SW Tester

Team Members:

- Mai Atef Abdallah Ebrahim
- Tasneem Mahmoud Rizk Ibraheem
- Radwa Mohamed Abdo Abdo
- Mariam Kamal Faheem Saad
- Hagar Abdallah Mahmoud

1. Introduction

This document outlines the testing efforts carried out for an educational platform's admin account. The platform is designed to assist educational institutions in managing various academic, administrative, and operational tasks. Our testing focused primarily on critical functionalities within the admin account, which contains numerous pages and modules, ensuring that the key features operate as expected.

1.1. Scope

The primary goal of this project was to validate the critical functionalities of the admin account in terms of performance, usability, and robustness. The testing process included manual testing, automated testing, API testing, and performance testing using a range of tools and techniques. **The testing process was divided into the following categories:**

- **Manual testing** for initial validation and exploratory testing.
- **Automated testing using Cypress** to automate repetitive and critical test cases.
- **API testing using Postman** to verify the backend interactions.
- **Performance testing using JMeter** to assess system stability under load and stress conditions.
- **Test management using Jira** to log, track, and manage test cases, bugs, and project progress.
- **Reports creation** for different types of testing (API, Performance) with the use of different tools like Newman for creating API report

1.2. used Technologies

In order to achieve comprehensive testing coverage, we utilized a combination of modern testing tools and frameworks:

- Cypress for automating end-to-end tests and ensuring key user flows functioned smoothly.
- Postman for API testing, with additional reporting generated through Newman.
- JMeter for performance testing, including both load and stress testing of APIs, to ensure the platform could handle expected user traffic.
- Jira for tracking the testing progress, logging bugs, and managing the overall workflow.
- Jira AIO Test Management Tool for managing manual test cases and organizing test plans.
- Fetch/XHR in browser developer tools for manual inspection of API requests and responses, due to incomplete API documentation.

These technologies allowed us to systematically verify the platform’s robustness and report detailed findings throughout the development cycle.

2. Educational Platform Admin Account features Tested

The admin account of the educational platform comprises a wide variety of pages and modules designed to handle different aspects of school administration. The following list represents the key sections within the admin interface that we tested:

- Dashboard: A central overview of key metrics and updates.
- Sidebar Manager: Manage and configure the sidebar for easier navigation.
- Administration: Core administrative tasks, including managing user roles and permissions.
- Academics: Features related to academic management, such as course assignments and schedules.
- Download Center: Provides access to downloadable resources.
- Study Material: Upload and manage academic study material for students.
- Lesson Plan: Plan and structure lessons for teachers.
- Student: Comprehensive management of student data.
- Student Info: Access to detailed student profiles and records.
- Behavior Records: Track and record student behavior incidents.
- Homework: Manage homework assignments and submissions.
- Examination: Tools for creating and managing exams.
- Exam Plan: Scheduling and organizing exam timetables.
- Online Exam: Conducting and managing online exams.
- HR (Human Resource): Management of staff and human resources.
- Teacher Evaluation: Evaluate teacher performance based on predefined criteria.
- Leave: Manage leave requests for staff and teachers.

There were additional features but due to the limited time we posterized the features and tested these features first.

While time constraints limited testing to specific functionalities, the combination of manual, automated, and performance testing ensured that the platform could support its intended use cases effectively.

3. Testing Approach

3.1. Manual Testing

Given the vast number of features available in the admin account, we prioritized the testing of critical functionalities. These areas were selected based on their importance to the overall system's performance and user experience. The critical areas included:

- **User Authentication and Authorization**: Ensuring only valid users with the correct credentials and roles could access the system.
- **Change password:** ensuring admin able to change his password successfully
- **Admin profile**: Verifying the correctness of the admin logged information and ability to edit profile data
- **Dashboard:** Verifying the data visualizations, reports, and real-time metrics displayed on the dashboard were accurate and up to date.
- **Administration:** Testing the ability of the admin to Provide resources for students and faculty to download, allow educators to upload and share academic materials, and to Manage course syllabi and weekly lesson plans.
- **Student:** Testing the ability of the admin to manage and test critical student-related features to ensure smooth functionality. These features include the Download Center, where students and faculty can access essential academic resources; Study Material, which enables educators to upload and share important academic content like lecture notes and textbooks; and Lesson Plan Management, allowing the oversight of course syllabi and weekly lesson plans. Admins test these features to ensure that uploads, file sharing, and real-time updates are accessible to students, enhancing the overall academic experience.
- **Online Exam:** Examining the workflow for creating and administering exams, including the submission process for students.
- **HR:** verify the ability of the administrators to manage staff-related operations effectively. This includes handling employee records, managing teacher evaluations, and overseeing leave requests. Admins conducted manual testing on these critical HR features to ensure the system supports proper functionality, streamlining administrative tasks and improving staff management.

During manual testing, we logged bugs and tracked the overall progress using Jira, a comprehensive project management tool. This ensured clear communication across the team and allowed for efficient tracking of issue resolution.

3.2. Automated Testing

Cypress was utilized to automate the test cases for the main functionalities of the admin account. This involved:

- Writing test cases for critical functionalities, such as user authentication, navigation between pages, form submissions, and data validation.
- Automating repetitive tests, such as checking the integrity of the Dashboard, navigating through Sidebar Manager, and verifying CRUD operations (Create, Read, Update, Delete) in modules like Student Info, Lesson Plan, and Exam Plan.

3.2.1. Key features tested

- **Login/Logout and User Roles**: Automated tests were developed to verify that users can log in and out successfully and access pages relevant to their roles.
- **Critical Workflow Testing:** Automated scripts tested workflows such as:
  - View&edit profile
  - Change password
  - Administrations
  - Exam.
  - HR
- **Form Validations:** Automated tests checked form input fields (e.g., login, user data entry) for validation rules to ensure appropriate error messages were triggered for invalid inputs.

3.3. API Testing

Due to the lack of API documentation, we relied on fetching API requests via Fetch/XHR to retrieve endpoints. This allowed us to perform Create and Read operations, which were the only available functions.

- We developed API test scripts in Postman to verify the creation of new data entries and the reading of student records and leave requests.
- Using Newman, we ran the Postman collections and created reports to validate the API testing process. This provided insights into potential issues with API performance and response times.

These tests ensured that the platform's backend interacted with the frontend effectively, even with limited API documentation.

3.4. Performance Test

To assess the website's performance, we conducted load testing and stress testing using JMeter. This helped us evaluate how well the platform performed under different user loads, which is critical for determining scalability and reliability. Performance testing focused on:

- Load Testing: Simulating normal user behavior to assess the website’s stability during peak hours.
- Stress Testing: Overloading the system with an increasing number of concurrent users to identify the maximum load it can handle before failing.
- API Performance: Measuring the response time and throughput for key API endpoints, ensuring that the website performs well under a high number of simultaneous requests.

The JMeter reports provided key insights, including response times, throughput, error rates, and system bottlenecks that could impact user experience.

3.5. Jira for Project and Test Management

Jira played a pivotal role in managing our testing activities. We utilized Jira to:

- **Log and track bugs:** Every issue encountered during testing was logged into Jira, categorized, and assigned for resolution.
- **Create test cases:** Using the AIO Testing tool integrated into Jira, we created and managed manual test cases, ensuring that each functionality of the system was covered.
- **Track project progress:** Jira allowed us to maintain visibility over the testing progress, including task assignments, deadlines, and overall project status.
- **Collaboration:** The platform helped foster collaboration between testers, developers, and project managers by keeping everyone aligned on the current state of the project.

4. Key challenges and solutions

API Documentation Gaps

- **Challenge**: The lack of API documentation required additional effort in reverse engineering API endpoints.
- **Solution**: Postman's Fetch/XHR feature was used to capture and analyze API requests from the frontend. This allowed us to test API interactions effectively despite the documentation gap.

Synchronization Issues in Automated Testing

- **Challenge**: Handling dynamic data, such as real-time exam schedules ,uploading images posed challenges for test automation.
- **Solution**: Cypress's built-in capabilities for automatic waiting and retrying actions ensured that the test cases ran smoothly even with dynamically loaded content.

Testing Coverage

- **Challenge**: Testing the entire platform was not feasible due to time constraints.
- **Solution**: We prioritized critical functionalities like Leave Management, Dashboard, and exam. Non-essential features like Chat and Style were deferred for future testing phases.

5. Summary

This project focused on testing the admin account of an educational platform, ensuring that critical features functioned as intended. The platform consists of multiple modules supporting administrative tasks, communication, and academic management. Given the complexity and importance of these features, the testing approach combined manual and automated methods to cover a wide range of functionalities.

Manual testing was prioritized for the platform's most critical features due to time constraints. The goal was to ensure that essential workflows and processes were free from critical issues. Jira was employed to track test cases and log bugs, utilizing AIO Testing to document the manual tests.

To enhance testing efficiency, Cypress was selected for automating key test cases, enabling efficient and repeatable validation of important workflows. This approach significantly reduced the time spent on regression testing while maintaining high accuracy. The focus was on automating the platform's core functionalities, which would otherwise require extensive manual testing.

API testing was conducted using Postman, given the lack of formal API documentation. This involved manually capturing GET and POST requests through browser developer tools (Fetch/XHR) and writing test scripts to validate the correctness of the responses, focusing on status codes, headers, and response payloads. Newman was utilized to automate the execution of API tests, allowing for consistent testing across different environments while generating detailed reports to track test outcomes.

Additionally, performance testing was performed using JMeter to evaluate the platform’s backend services. Load and stress tests simulated multiple users interacting with the APIs simultaneously, ensuring that the platform could effectively scale under high user demand and identify any bottlenecks or performance issues.

Reports generated from manual testing in Jira, automated test reports from Cypress, API testing reports from Postman/Newman, and performance testing results from JMeter were all documented and shared with the team. These reports provided detailed insights into test coverage, highlighting any issues discovered and ensuring that key functionalities performed as expected under varying conditions.

The project revealed important learnings and challenges, including time constraints that led to prioritizing key features for manual testing while less critical features remained uncovered. The lack of formal API documentation complicated the API testing process, requiring the team to capture requests from browser tools. Balancing manual and automated testing proved crucial, as automation helped reduce testing time for key features, but manual testing was still necessary for areas that couldn’t be automated due to time or complexity.

In conclusion, the combination of manual, automated, and performance testing ensured comprehensive coverage of the educational platform within the available timeframe. This approach validated the functionality and stability of key features, delivering a reliable experience for end users. Future work may focus on extending automation to less critical features and improving API documentation for easier testing.