Group 19 - IBM Skill Build Application

2023

SKILL BUILD APPLICATION

Test Plan Report

Catherine Doo (vntt37)
Emmanuel-James Bertland (mknd43)
Muhammad Ali Asad (sxgx37)
Muhammed Qasim Qureyshi (jlzc57)
Ngai Tsz Yan Peony (dxgd33)
Nneka Ikeyi (gsnb85)
Olivia Allinson (sncl34)

Section 1 - Introduction and Test Overview

1.1.1 Introduction

A cutting edge online learning platform

A revolutionary approach to personalised online education is being introduced by us. Our platform will lead you to the vast array of IBM's comprehensive course offerings, by factoring your personal interests. We aim to make learning more accessible, personalised, and effective for everyone. We believe a seamless online course recommendation bot would make learning available to the larger public.

How It Works

Our cutting-edge chatbot technology serves as your personal learning assistant. Here's how it enhances your educational journey:

Personalised Interaction: Our chatbot utilises multiple choice questions as inputs for your interests, learning preferences, and what you hope to achieve in your career.

Tailored Course Recommendations: Based on the interests, the chatbot will process a wide range of IBM courses, including data science, artificial intelligence, cloud computing, cybersecurity, and more, to locate the most relevant one.

Instant Access to Resources: You immediately receive links to courses that match your interests, allowing convenient access to enrol and explore them.

Continuous Learning Support: The chatbot is designed to evolve with your learning journey, offering new suggestions and resources as your interests and skills develop.

Our Scope

For Learners: Whether you are a beginner looking to understand the basics or a professional aiming to enhance your skills and specialise, our platform caters to all levels.

Expansive Course Categories: Many topics are available in partnership with IBM, ensuring that you have access to the latest and most relevant courses, and therefore the most up to date knowledge.

Accessibility and Flexibility: Our platform is accessible online through the browser, on any device, making learning easy to fit into your busy schedule effortlessly.

1.1.2 Test Overview

For our IBM Skills Build website project, we're using the Agile framework to guide our development. This approach is well-suited for the ever-changing world of online education. It gives us the flexibility to quickly make changes and improvements as needed. Agile also helps us handle any risks we might have spotted during our project planning. This way, we can develop our educational platform in a smooth and flexible manner.

Furthermore, our development strategy includes the implementation of Test-Driven Development (TDD). This choice is particularly beneficial for a platform like ours, where consistent functionality and reliability is crucial. TDD will guide our coding efforts by requiring specific, predefined test cases to be passed before further development, ensuring high code quality and immediate feedback on functionality. This method promotes an incremental build-up of the platform's logic, clear, maintainable, and scalable code. Our emphasis will be on delivering essential features that meet the minimum requirements for passing these tests, thereby avoiding unnecessary complexity.

Our testing strategy encompasses a comprehensive range of tests, each tailored to different aspects of the IBM Skills Build website. We will start with unit tests, which are crucial for ensuring the integrity of individual components. We will then proceed to integration testing, which is pivotal in verifying the seamless interaction between various components of our platform. This step is important in ensuring a cohesive user experience

System testing will follow, focusing on both functional and non-functional requirements of the integrated system. This phase ensures that the IBM Skills Build website as a whole meets our predefined standards and specifications. Lastly, user acceptance testing will be conducted to confirm that the platform meets the expectations and needs of our end-users, primarily students and educators. Feedback from real users will be instrumental in fine-tuning the final product.

This report, while a preliminary overview, is structured to systematically document the outcomes of our various tests. Through this rigorous and methodical testing approach, we aim to create an educational platform that is robust, user-friendly, and scalable, aligning with the high standards of IBM.

1.2.1 Unit Testing

Our unit tests will primarily focus on the key elements of the website, including the interactive learning modules, user account management, and content delivery systems. We will employ a combination of Black Box and White Box testing methods to thoroughly evaluate each unit. Black Box testing will allow us to assess the website's functionality without the need to understand the underlying code, ensuring that the user experience aligns with our expectations. In contrast, White Box testing will delve into the code itself, identifying any logical errors or flaws in the design that could impact the website's performance or user experience.

By conducting these tests, we aim to cover all aspects of the system – from the front-end user interface developed with modern web technologies to the back-end server and database operations. Our goal is to ensure a high level of quality and reliability, reducing the likelihood of issues post-deployment. To guide our testing process, we have identified a range of unit tests, detailed in Section 2 of this report, which are pivotal for the robust operation of our educational platform. These tests are derived from the specific requirements and objectives outlined in our project specification, ensuring that each aspect of the website not only functions as intended but also delivers a seamless and engaging learning experience to the users.

Test Case ID:	Description:	Type:
unit_test-01	Testing that the web-application is functional across different	Black Box

	platform sizes.	
unit_test-02	Testing that the web-application only allows access when the correct password for a certain username is provided.	White Box
unit_test-03	Testing course search functionality - that courses are effectively tagged and categorised for keyword searches,	Black Box
unit_test-04	Testing that the ChatBot provides a course recommendation based on user answers to provided questions.	Black Box
unit_test-05	Testing that the Learning Pathway appears after accepting terms and conditions.	Black Box
unit_test-06	Testing course access - that the hyperlinks in the Learning Pathway are valid.	White Box
unit_test-07	Testing the integration of calendar functionality.	White Box
unit_test-08	Testing the implementation of progress tracking.	Black Box
unit_test-09	Testing the leaderboard function updates after quizzes are completed.	Black Box

1.2.2 System Testing

Functional and performance testing will be performed to verify that the functional and non-functional requirements outlined in the requirements specification are satisfied. This specification was utilised to establish the test oracles.
The tests concentrate on the technical components of the system and assess its performance to guarantee its functionality and stability. As outlined in Section 2, there are eight system tests, with six of them being emphasised and having detailed test cases.

Test case ID:	Description:	Nature of Requirement:	Type:

sys_test-01	User Authentication - Application allows for successful user registration and login.	Functional	Black box
sys_test-02	Accessibility to IBM Online Courses - Users can access IBM online courses through the learning pathway.	Functional	Black Box
sys_test-03	Mobile Compatibility - Application operates seamlessly across different platforms.	Functional	Black Box
sys_test-04	Data Management - The application should have regular data backups and a disaster recovery plan to prevent data loss in case of system failures or data corruption.	Functional	White Box
sys_test-05	Compliance - The application should pass periodic audits to ensure compliance with relevant laws and regulations.	Non-Functional	Black Box
sys_test-06	System Input Limits	Non-functional	Black Box
sys_test-07	Guided Learning Pathway	Functional	Black Box
sys_test-08	Error Handling	Non-Functional	Black Box
sys_test-09	Stress	Non-functional	Black Box
sys_test-10	Concurrency	Non-functional	Black Box
sys_test-11	Course Search Functionality	Functional	Black Box

1.2.3 Integration Testing

We will use a bottom-up integration testing approach. Start by testing individual components such as user authentication, registration system, database management and chatbot to ensure our unit tests work together correctly. This allows us to gradually improve our product's functionality and eliminate issues caused by combining components. The bottom-up method was chosen for its ease of implementation and ability to detect and localise faults.
Although slower than other integration testing methods like the ‘sandwich approach’ and ‘big bang testing’, this method thoroughly tests our product's functionality.
Our development process follows a bottom-up approach, as illustrated in the diagram below:

Test Case ID:	Description:	Type:
int_test-01	Ensure that user registration information is successfully transferred from the registration system to the database.	Black Box
int_test-02	Ensure that the login system correctly validates user authentication credentials using information from the database.	Black Box
int_test-03	Confirm that the chatbot's progress data is accurately stored in the database for future analysis.	Black Box
int_test-04	Validate that the chatbot works seamlessly with the user registration system to gather initial user information.	Black Box
int_test-05	Verify that the guided learning pathway presentation works properly with the user authentication system, allowing only authenticated users to access the pathway.	Black Box
int_test-07	Ensure that educational games and quizzes accurately update the user's progress in the database and are reflected in their learning journal.	Black Box
int_test-08	Ensure that the social integration feature, specifically	Black Box

	the leaderboard, updates dynamically based on user progress data stored in the database.	
int_test-09	Ensure that the social integration feature, specifically the leaderboard, updates dynamically based on user progress data stored in the database.	Black Box
int_test-10	Ensure that the social integration feature, specifically the leaderboard, updates dynamically based on user progress data stored in the database.	Black Box

1.2.4 User Acceptance Testing

The final phase of testing includes User Acceptance Testing (UAT), a critical evaluation process designed to assess the alignment of the developed product with stakeholder requirements and ensure maximum client satisfaction. The web application's usability and efficiency are emphasised, with a focus on minimising disruptions during instructional sessions.

Test Case ID:	Description:	Test Type:
UA_test-01	Responsive window design	Black Box
UA_test-02	Application should be easy to use and provide clear instructions on its usage	Black Box
UA_test-03	Application should have an acceptable response time	Black Box
UA_test-04	Educational games and quizzes functionality	Black Box
UA_test-05	Guided learning pathway presentation	Black Box
UA_test-06	Database management and data tracking	Black Box
UA_test-07	Web app colour and design should be aesthetically pleasing	Black Box

Section 2 - Test Cases

2.1 Unit Testing

Test Case ID	unit_test-02	
Description of Test	Testing that the web-application only allows access when the correct password for a certain username is provided.	
Relevant Requirement and Design Specifications	FRs	FR1.2, FR1.1, FR2.1
	NFRs	NFR 1.1, NFR 2.1, NFR 2.2
Prerequisites for Test	A functional web-application, which passes unit_test-01. A database with the details of registered users.	
Test Type	White Box	

Test Procedure	Testing the user authentication would be automated - tested with POST requests.
	
	`chai.request(server)`
	`        .post('/auth/sign_in')`
	`        .send({`
	`            'email': 'tester@gmail.com',`
	`            'password': 'tester'`
	`        })`
	`        .end((err, res) => {`
	`            console.log('this runs the login part');`
	`            res.body.should.have.property('token');`
	`            var token = res.body.token;`
	
	1. Submit a POST request with a valid email and password.
	2. Submit a POST request with an invalid email and password.
	
	Code snippet from Buddy (Oluyege, 2020).	

Test Materials Used	A computer or mobile device with internet access and one of the common Web Browsers (such as Google Chrome).	
Equivalence Classes	Normal	A registered username is entered with the corresponding password.
	Error	An incorrect username or password is entered.
Test Oracle (Expected Output)	If the correct username and password of a registered user is entered, the User Dashboard is displayed. If the username and password entered are not associated with a registered user, a message informing the user appears on the Login page.	

Comments	The authentication is done with JSONWebToken.
Created by	Catherine
Test Environment(s)	Windows 11, iOS 17, Android 12

Test Case ID	unit_test-04	
Description of Test	Testing that the ChatBot provides a course recommendation based on user answers to provided questions.	
Relevant Requirement and Design Specifications	FR1.3, FR1.1, FR1.8, FR2.1	
Prerequisites for Test	A functional web-application, which passes unit_test-01. A functional database to store users' answers.	
Test Type	Black Box	
Test Procedure	1. Register as a new user. 2. Log in. 3. Answer the questions by clicking an answer directly related to the same course for each question. 4. Answer the questions by choosing options related to different courses. 5. Click submit.	
Test Materials Used	A computer or mobile device with internet access and a search engine.	
Equivalence Classes	Normal	The user provides answers which directly relate to a course.
	Boundary	The user provides answers which relate to multiple courses.
Test Oracle (Expected Output)	If the user provides answers which indicate a particular course, the link to the IBM SkillsShare course is provided. If the answers indicate multiple courses, a list of the courses and their links is provided.	
Comments	The answers are provided in a multiple choice format, based on the courses in the IBM SkillsShare website, so there are never unintelligible or unrelated answers. The chatbot should appear when a new user logs in.	
Created by	Catherine	

Test Environment(s)	Windows 11, iOS 17, Android 12

Test Case ID	unit_test-06	
Description of Test	Testing course access - that the hyperlinks in the Learning Pathway are valid.	
Relevant Requirement and Design Specifications	FRs	FR1.5, FR1.1, FR1.4,
	NFRs	NFR 4.1
Prerequisites for Test	A functional web-application, which passes unit_test-01. A functional Learning Pathway, which passes unit_test-05.	
Test Type	White Box	

Test Procedure	The testing should be automated, as there are many courses and therefore many hyperlinks.
	
	`cy.get('#hyperlink')`
	`  .then((link) => {`
	`   cy.request(link.prop('href')).then((resp => {`
	`       expect(resp.status).to.eq(200)`
	`   })`
	`})`	

Test Materials Used	A computer or mobile device with internet access and a search engine. The IBM SkillsShare course hyperlinks.	
Test Oracle (Expected Output)	1. If the link no longer leads to a working webpage, the test will be failed. 2. If the link leads to a working webpage, the test will be passed as 200 will be returned.	
Comments	The href attributes are tested using Cypress.	
Created by	Catherine	
Test Environment(s)	Windows 11, iOS 17, Android 12	

Test Case ID	unit_test-08
Description of Test	Testing the implementation of progress tracking.
Relevant Requirement and Design Specifications	FR1.6, FR1.1, FR1.4, FR1.5, FR2.1
Prerequisites for Test	A functional web-application, which passes unit_test-01. A working user authentication which passes unit_test-02. The Learning Pathway passes unit-test_05. Course Access, so unit_test-06 was passed. A working database. A registered user who has begun a

	course.	
Test Type	Black Box	
Test Procedure	1. Log in. 2. From the User Dashboard, move to the Learning Pathway. 3. From the Learning Pathway, move to Course Access. 4. Use the Progress Tracking under Course Access. 5. Click the checkbox next to a course on the checklist to indicate it has been completed. 6. Select the button to take the quiz. 7. Fill in the quiz with mostly correct answers. 8. Fill in the quiz with mostly incorrect answers. 9. Submit the quiz.	
Test Materials Used	A computer or mobile device with internet access and a search engine. The correct answers for a quiz - based on a certain course.	
Equivalence Classes	Normal	The user passes the quiz.
	Boundary	The user fails the quiz.
Test Oracle (Expected Output)	If the user passes, the progress tracking checklist should have a tick next to that course. If the user fails, the progress bar should be unchanged and the user should be able to take the quiz again.	
Comments		
Created by	Catherine	
Test Environment(s)	Windows 11, iOS 17, Android 12	

Test Case ID	unit_test-09
Description of Test	Testing the leaderboard function updates after quizzes are completed.
Relevant Requirement and Design Specifications	FR1.1, FR1.2, FR1.4, FR1.8, FR2.1
Prerequisites for Test	A functional web-application, which passes unit_test-01. A user authentication system which passes unit_test-02. The Learning Pathway passes unit-test_05. A working database.
Test Type	Black Box
Test Procedure	1. A user is registered or logs in. 2. Fill in the quiz with incorrect answers. 3. Fill in the quiz with all correct answers.

	4. Fill in the quiz with mostly correct answers. 5. Submit the quiz. 6. Check the leaderboard from the User Dashboard.	
Test Materials Used	A computer or mobile device with internet access and a search engine. The correct answers for a quiz - based on a certain course.	
Equivalence Classes	Normal	A user who is already on the leaderboard.
	Boundary	- A new user who is not already on the leaderboard. - The user fails the quiz.
Test Oracle (Expected Output)	If the user fails the quiz, the leaderboard should not change. Otherwise, the number of points the user has on the leaderboard should increase, with the increase in points corresponding to how well the user did in the quiz. If the user is not already on the leaderboard, the user should be added - and their position and number of points should be prominently displayed for them.	
Comments		
Created by	Catherine	
Test Environment(s)	Windows 11, iOS 17, Android 12	

2.2 System Testing

Test Case ID	sys_test-01
Description of test	Application allows for successful user registration and login.
Related requirement spec/design spec details	FR1.2
Prerequisites for test	System is operational.
Test Type	Black box
Test procedure	1. Navigate to the registration page 2. Enter valid registration details and submit. 3. Log out and navigate to the login page. 4. Enter login credentials and submit

Test material used	Users details for registration and login
Expected result (test oracle)	User is able to register and log in successfully, accessing their personalised dashboards
Comments	None
Created by	Nneka
Test environment(s)	Cross-platform (Windows 10, various mobile devices).

Test Case ID	sys_test-02
Description of Test	Users can access IBM online courses through the learning pathway.
Relevant Requirement and Design Specifications	FR1.5 - Course Access – Accessibility to IBM Online Courses.
Prerequisites for Test	User is logged in and has a learning pathway set up.
Test Type	Black box
Test Procedure	1. Follow the learning pathway to an IBM online course. 2. Click on the link to access the course.
Test Materials Used	Learning pathway interface.
Test Oracle (Expected Output)	The course link opens successfully, and the user can access the course content.
Comments	None
Created by	Nneka
Test Environment(s)	Cross-platform (Windows 10, various mobile devices).

Test Case ID	sys_test-03
Description of Test	Application operates seamlessly across different platforms.

Relevant Requirement and Design Specifications	NFR 3.2 - Cross-Platform Compatibility.
Prerequisites for Test	Various devices and browsers are available for testing.
Test Type	Black box
Test Procedure	1. Access the application on different browsers. 2. Access the application on different mobile devices. 3. Verify responsiveness and functionality.
Test Materials Used	Multiple browsers and mobile devices.
Test Oracle (Expected Output)	The application functions well across different browsers and mobile devices without loss of features or usability.
Comments	Include a variety of screen sizes and orientations.
Created by	Nneka
Test Environment(s)	Multiple browsers and operating systems.

Test Case ID	sys_test-04
Description of Test	The application should have regular data backups and a disaster recovery plan to prevent data loss in case of system failures or data corruption.
Relevant Requirement and Design Specifications	NFR 4.4 - Backup and Disaster Recovery
Prerequisites for Test	Backup and disaster recovery tools are available.
Test Type	White Box
Test Procedure	1. Perform a backup of the system data. 2. Simulate a disaster scenario and initiate the recovery plan.
Test Materials Used	Backup and disaster recovery tools.
Test Oracle (Expected Output)	The system should be restored quickly after a disaster, and the backup process should be successful.
Comments	None
Created by	Nneka
Test Environment(s)	Windows 11

Test Case ID	sys_test-05
Description of Test	The application should pass periodic audits to ensure compliance with relevant laws and regulations.
Relevant Requirement and Design Specifications	NFR 4.3 - Compliance
Prerequisites for Test	Compliance audit tools are available.
Test Type	Black Box
Test Procedure	1. Conduct a compliance audit to verify adherence to relevant laws and regulations. 2. Check user acceptance of terms of service and privacy policies.
Test Materials Used	Compliance audit tools.
Test Oracle (Expected Output)	The application should pass the compliance audit and show a high rate of user acceptance for terms of service and privacy policies.
Comments	None
Created by	Nneka
Test Environment(s)	Windows 11

2.3 Integration Testing

Three integration tests ensure the product's functionality, including proper communication and integration of frontend, backend, and hardware components. These tests are prioritised because they have the potential to cause errors if not completed.

Test Case ID	int_test-03
Description of Test	Confirm that the chatbot's progress data is accurately stored in the database for future analysis.
Relevant Requirement and Design Specifications	FR1.3 - Chatbot – Chatbot Integration

	FR2.1 - Database Management – Comprehensive database establishment
Prerequisites for Test	- Chatbot integration is enabled and configured correctly - User registration process includes interaction with the chatbot
Test Type	Black Box
Test Procedure	1. Start the user registration process 2. Engage with the chatbot during the registration process to provide initial user information (e.g. professional or educational backgrounds, aspirations). 3. Complete the registration process
Test Materials Used	- Web browser or mobile devices with access to the web application - Chatbot interface
Test Oracle (Expected Output)	- The chatbot collects initial user information correctly during the registration process. - User information collected by the chatbot is stored correctly in the database.
Testing	Positive Testing: · Complete the registration process with valid information and verify successful registration · Check that the chatbot collects and stores user information accurately Negative Testing: · Attempt to register without engaging with the chatbot · Provide invalid or incomplete information during chatbot interaction and verify appropriate error handling
Created by	Peony
Test Environment(s)	Web application environment. Chatbot integration environment

Test Case ID	int_test-08
Description of Test	Validate that the social integration feature, specifically the leader board, updates dynamically

	based on users' progress data stored in the database.
Relevant Requirement and Design Specifications	FR1.6 - Progress Tracking – Progress Tracking through checklist implementation FR1.9 - Social Integration – Social integration with leaderboard feature FR2.1 - Database Management – Comprehensive database establishment
Prerequisites for Test	Multiple registered user accounts exist in the system. Users have made progress in various activities to populate the leader board. Progress data is accurately stored in the database.
Test Type	Black Box
Test Procedure	1. Log in to the system as an authenticated user. 2. Engage in various activities that contribute to progress (e.g., completing courses, participating in quizzes). 3. Verify that the leader board dynamically updates based on users' progress data stored in the database
Test Materials Used	Web browser or mobile devices with access to the web application
Test Oracle (Expected Output)	The leader board updates dynamically based on users' progress data stored in the database. Changes in users' progress reflect accurately in the leaderboard rankings.
Testing	Positive Testing: · Engage in activities and verify that progress updates are reflected in the leaderboard. · Check that the leaderboard rankings change accordingly based on users' progress. Negative Testing: · Attempt to manipulate progress data to affect leaderboard rankings. · Verify that the leaderboard accurately reflects users' genuine progress.
Created by	Peony
Test Environment(s)	Web application environment. Database environment (e.g., MongoDB)

Test Case ID	int_test-10
Description of Test	Verify that all components and features integrated in the system function cohesively during an end-to-end scenario.
Relevant Requirement and Design Specifications	None
Prerequisites for Test	All system components and features are integrated and operational. User accounts exist in the system. Relevant data (courses, progress, etc.) is available in the system.
Test Type	Black Box
Test Procedure	1. Start an end-to-end scenario involving multiple system features (e.g., new user registration, interaction with the chatbot, accessing external courses, completing educational activities, viewing progress on the leaderboard). 2. Verify that all components and features function as expected and interact seamlessly with each other.
Test Materials Used	Web browser or mobile devices with access to the web application
Test Oracle (Expected Output)	All integrated components and features function cohesively during the end-to-end scenario. Users are able to complete the scenario without encountering errors or inconsistencies.
Testing	Positive Testing: · Perform the end-to-end scenario as intended and verify successful completion. · Check that data flows seamlessly between different system components. Negative Testing: · Introduce variations in user input or system conditions to simulate edge cases. · Verify that the system handles unexpected scenarios gracefully without crashing or producing incorrect results.
Created by	Peony
Test Environment(s)	Web application environment. Integrated system environment.

2.4 User Acceptance

Test Case ID	UA_test-02
Description of Test	Application should be easy to use and provide clear instructions to the user on its usage
Relevant Requirement and Design Specifications	NFR 3.1 - Usability and Accessibility
Prerequisites for Test	A stable version of the website deployed in a testing environment with back-end functionality Pre-defined user scenarios and tasks to evaluate usability. Access to feedback tools for gathering user input.
Test Type	Usability Testing / User Interface Testing
Test Procedure	1. Select a diverse group of test users, including those with varying levels of technical expertise. 2. Provide them with a series of tasks to complete using the application, ensuring these tasks cover all major functionalities. 3. Instruct users to follow the guidance within the application 4. Collect feedback on the clarity, helpfulness, and accessibility of the instructional materials and assistance features. 5. Analyse feedback to identify areas where users struggle and where guidance is most effective.
Test Materials Used	User feedback forms or tools. Task list for users to complete.
Test Oracle (Expected Output)	Users should be able to complete the tasks with minimal confusion or errors. Feedback indicates that the guidance and instructional materials are clear, helpful, and accessible. The application should demonstrate a user-friendly interface that simplifies the learning curve for new users.
Comments	None
Created by	Qasim

Test Environment(s)	Cross-platform (Windows 11, macOS, iOS, Android) Devices with various screen sizes and resolutions.

Test Case ID	UA_test-03
Description of Test	Application should be responsive, with an acceptable loading time, even under high traffic
Relevant Requirement and Design Specifications	NFR 1.1 - Web application optimisation, NFR 1.2 - Scalability, NFR 1.3 - Availability
Prerequisites for Test	A stable version of the website deployed in a testing environment. Tools for generating and simulating high user traffic. Responsive window design as in UA_test-01
Test Type	Load Testing / Performance Testing
Test Procedure	1. Use load testing tools to simulate a high number of concurrent users accessing the website. 2. Monitor server response times, page load times, and system resource usage. 3. Identify any performance bottlenecks or failures.
Test Materials Used	Use load testing tools to simulate a high number of concurrent users accessing the website. Monitor server response times, page load times, and system resource usage. Identify any performance bottlenecks or failures.
Test Oracle (Expected Output)	The application should pass the compliance audit and show a high rate of user acceptance for terms of service and privacy policies.The application maintains a consistent response time. No system crashes or significant performance degradation under high load.
Comments	None
Created by	Qasim
Test Environment(s)	Windows 11

Test Case ID	UA_test-07

Description of Test	The web application's colour scheme and design should be aesthetically pleasing, consistent across different pages, and enhance user experience without compromising readability or functionality.
Relevant Requirement and Design Specifications	NFR 3.1 - Usability and Accessibility, NFR 3.2 - Cross-Platform Compatibility
Prerequisites for Test	A stable version of the web application deployed in a testing environment. Access to different browsers and devices to test responsiveness and consistency. Design mock-ups and style guides for reference.
Test Type	Design Consistency Testing / Color Scheme Testing
Test Procedure	1. Navigate through all pages of the web application to check for design consistency and alignment with the provided mock-ups and style guides. 2. Evaluate the colour scheme for contrast, brand alignment, and visual appeal on various pages. 3. Test the application on different browsers (like Chrome, Firefox, Safari) and devices (like desktops, tablets, smartphones) to ensure responsiveness and consistent appearance. 4. Check the readability of text, visibility of icons, and usability of interactive elements against the background and colour scheme. 5. Collect feedback from a sample group of users regarding the visual appeal, readability, and overall experience. 6. Document any discrepancies from the design specifications and user feedback for improvements.
Test Materials Used	Design style guides and mock-ups. List of supported browsers and devices. User feedback forms or survey tools.
Test Oracle (Expected Output)	The web application should consistently reflect the design mock-ups and style guides across all pages and on different devices. Colours should enhance readability, align with branding, and be aesthetically pleasing. User feedback indicates a positive response to the design and colour scheme.The application should pass the compliance audit and show a high rate of user acceptance for terms of service and privacy

	policies.The application maintains a consistent response time. No system crashes or significant performance degradation under high load.
Comments	None
Created by	Qasim
Test Environment(s)	Multiple web browsers (Chrome, Firefox, Safari, etc.). Variety of devices (desktop, laptop, tablet, smartphone) with different screen sizes and resolutions.

Section 3 - Testing Context and Tools Collaboration

3.1.1 Testing Context and Tool Selection

In the development of the Skills Build website, the choice of testing tools and the context in which they are used are pivotal for ensuring a robust and user-friendly platform. This section outlines the testing context and the rationale behind the selection of specific tools, aligned with the unique requirements of our educational platform.

3.1.2 Device and Browser Diversity

Considering our target audience, which includes a diverse range of users with varying tech proficiencies and device preferences, our testing environment is designed to be as inclusive as possible.

We focus on:

Laptops and Mobile Phones:
Predominantly used by our target demographic of university students and educators.
Major Operating Systems:
Windows, MacOS for laptops, and Android, iOS for mobile devices, reflecting the broad market coverage.
Key Web Browsers:
Chrome (122.0.6261.18 x64), Safari (17.2, x64), and Edge (120.0.2210.167, x64), selected based on their market dominance and relevance to our audience.

3.1.3 Testing Tools and Technologies

Automated Testing Frameworks: Tools like Selenium and Cypress are chosen for their ability to automate browser-based tests, crucial for ensuring consistency and efficiency in testing.
API Testing with Postman: Essential for validating the functionality of our back-end services and their integration with the front end.
Responsive Design Testing: Utilising built-in tools in Chrome and Safari to ensure our application's UI adapts seamlessly across different screen sizes.
Cross-Platform Compatibility Tools: Emulators and simulators to test the application on various devices and operating systems that we might not physically possess.

3.2 Collaborative Testing Approach

Shared Testing Environments: BrowserStack will be used for cross-browser and cross-platform testing, allowing team members to collaborate and share results in real-time.
Continuous Integration (CI) Pipeline: Implementing a CI/CD pipeline using the tools Jenkins or GitHub Actions. This ensures that new code changes are automatically tested, maintaining the code quality throughout the development process.
Code Quality and Security Scans: Utilising tools SonarQube and OWASP ZAP to regularly scan the codebase for potential quality issues and security vulnerabilities, ensuring adherence to best practices.

3.3 Rationale for Tool Selection

Diversity and Inclusivity: By covering a wide range of devices and browsers, we ensure that our platform is accessible and functional for all users.
Efficiency in Testing: Automated tools significantly reduce the time and resources required for comprehensive testing, allowing for more frequent and thorough testing cycles.
Collaboration and Consistency: Cloud-based tools and CI pipelines facilitate seamless collaboration among team members and maintain consistency in testing standards.
Proactive Problem-Solving: Regular quality and security scans help in proactively identifying and resolving potential issues before they impact users.

3.4 Test Failure Severity Classification

To effectively prioritise and address potential issues within the Skills Build website, we've adopted a detailed severity classification system. This system ranges from 'mild' to 'infectious', covering a spectrum of possible issues that could arise during testing. This classification aids in promptly identifying and resolving the most critical issues, ensuring the stability and reliability of the website.

Severity Level Classification:

Severity Level:	Examples:	Relevant Tests:
Mild	Typos, difficult-to-read fonts, misaligned buttons or texts.	UA_test-01 UA_test-02 UA_test-07
Moderate	Content loading delays, minor response time issues.	int_test-08 UA_test-03
Serious	Inability to load content, navigation failures, forms not submitting.	int_test-03 sys_test-01 unit_test-02 unit_test-06 sys_test-02 sys_test-07 UA_test-05
Very Serious	Erratic behaviour in interactive elements, critical feature malfunctions.	unit_test-04 unit_test-08 unit_test-09 UA_test-04
Extreme	Session crashes preventing user re-entry, significant functionality loss.	int_test-10 sys_test-07
Infectious	Backend server failure causing systemic breakdown, multi-component malfunctions.	sys_test-05 UA_test-06

Prioritization Strategy for Addressing Test Failure Severity:

Mild to Moderate Failures

Importance:

Mild to moderate failures, while not critical, still require attention as they can impact the user experience to some extent. These failures are typically considered low priority due to their minimal impact on the overall functionality of the system. The system remains largely operational, but aspects of the user experience might be slightly compromised.

Impact on the System

Mild Failures: These might include minor user interface glitches or delays in non-critical functions, which do not significantly hinder system performance or usability.
Moderate Failures: Issues like occasional slow response times or intermittent errors in less crucial features. While they don't cause system breakdowns, they can lead to user inconvenience.

Resolution Strategy

Mild Failures: Focus on resolving these as part of the routine development process, leveraging standard debugging and optimization techniques.
Moderate Failures: Require a more structured approach, often involving targeted troubleshooting and code refinement, but can generally be addressed within the normal development workflow.

Mitigation Strategy

1. Proactive Identification: Implement tools and practices to identify these issues early in the development cycle.
2. User Feedback Integration: Regularly incorporate user feedback to identify and rectify these failures.
3. Iterative Testing: Continuously test the system to ensure that these minor issues are caught and fixed promptly.

Post-Deployment Consideration:

Regular testing and maintenance are essential to address these issues as they arise post-deployment. The focus should be on continuous improvement, ensuring that these mild to moderate failures do not escalate or accumulate, thereby maintaining a high standard of user experience.

Priority in Multiple Failures:

In scenarios with multiple types of failures, mild to moderate failures should be addressed after more critical issues have been resolved. The priority is to first tackle failures that have a greater impact on the user experience and system functionality. Once these are contained or resolved, attention can shift to resolving mild to moderate failures to enhance overall system quality and user satisfaction.

Serious to Very Serious Failures

Importance:

Serious and very serious failures are critical issues that have a substantial impact on the functionality, security, and overall user experience of the system. They represent the most urgent problems that need immediate attention due to their potential to cause significant disruption and harm.

Impact on the System:

Serious Failures: These significantly impair core functionalities, leading to user frustration and reduced efficiency.
Very Serious Failures: These pose severe risks to data integrity and user security, potentially leading to catastrophic outcomes like data loss or exposure of sensitive information.

Resolution Priority:

Serious Failures: High priority for prompt fixing; should be addressed immediately upon detection during testing and post-deployment.
Very Serious Failures: The highest priority with an immediate and effective response required to prevent severe long-term consequences.

Mitigation Strategy:

1. Extensive Testing: Rigorous testing to uncover potential failures early in the development cycle.
2. Rapid Response Teams: Dedicated teams for immediate handling of serious and very serious failures.
3. Regular Updates and Patches: To prevent security vulnerabilities and functionality malfunctions.
4. Real-Time Monitoring Post-Deployment: To quickly detect and address any issues.
5. Contingency Planning: For quick action in case of very serious failures.
6. Transparent User Communication: To maintain trust and credibility.

Post-Deployment Consideration:

Continuous monitoring and an immediate response system are essential. Automated alerts for unusual system behaviours and regular system audits should be in place, alongside a skilled technical team ready to address issues as they arise.

Priority in Multiple Failures:

When facing multiple failures, the priority should be given to resolving very serious failures first due to their potential for greater harm. Once these are contained or resolved, attention should shift to serious failures to restore full system functionality and ensure user satisfaction.

Extreme and Infectious Failures

Importance:

Extreme and infectious failures represent the most critical and alarming issues within a system. They not only cause immediate and widespread disruption but also pose a significant risk of long-term damage to the system's integrity and user trust.

Impact on the System:

Extreme Failures: Completely debilitate the system's functionality, impacting all users and all aspects of the system's operation.
Infectious Failures: Characterised by their propensity to spread, these failures, such as backend server failures, lead to systemic breakdowns and multi-component malfunctions, rapidly escalating the severity of the situation.

Resolution Priority:

Extreme Failures: Addressed with the utmost urgency due to their total system shutdown implications.
Infectious Failures: Require immediate containment and resolution strategies to prevent widespread system collapse and further propagation of the issue.

Mitigation Strategy:

1. Advanced Monitoring Systems: Implement state-of-the-art monitoring tools for early detection of signs that could lead to extreme or infectious failures.
2. Rapid Response and Containment: Develop and rehearse rapid response procedures specifically designed to quickly contain and rectify infectious failures.
3. Robust Backup Systems: Ensure strong and regularly tested backup systems are in place to maintain essential functionalities during extreme failures.
4. Collaboration with Experts: Maintain relationships with external IT specialists and organisations for emergency support and guidance.

Post-Deployment Consideration:

Constant monitoring and immediate response capabilities are crucial. This includes setting up automated alerts for unusual system behaviour, conducting regular comprehensive system audits, and maintaining a team of skilled professionals ready to tackle these failures as they arise.

Priority in Multiple Failures:

When faced with multiple failures, extreme and infectious failures must be prioritised above all others. Their potential to cause immediate and extensive harm to the system and users necessitates a focus on rapid stabilisation and resolution of these critical issues before addressing other types of failures.

Conclusion:

This prioritisation strategy ensures that our response to potential failures is measured, efficient, and aligned with the overarching goals of the Skills Build platform. By classifying and addressing issues based on their severity and impact, we aim to maintain a high-quality, reliable, and engaging educational environment. Regular testing and maintenance, coupled with a readiness to tackle critical issues promptly, are key to achieving this objective.

References:

Cypress Documentation. (2024, January 16). request | Cypress Documentation. Why Cypress? Retrieved January 26, 2024, from https://docs.cypress.io/api/commands/request

Hric, F. (2022, January 18). Quicker way to check links in Cypress. YouTube. Retrieved January 26, 2024, from https://www.youtube.com/watch?v=6rTkQwJLmhc

Oluyege, P. (2020, November 3). Unit Testing JWT Secured Node and Express RESTful API with Chai and Mocha. Buddy. Retrieved January 24, 2024, from https://buddy.works/tutorials/unit-testing-jwt-secured-node-and-express-restful-api-with-chai-and-mocha