Software Test Plan for IBM Augmented Reality Quantum with AI

Version 1.0.0

Authors: Sorcha Corrigan, George Gaumond, Ziyi Huang, Aakif Rehman, Millicent Riordan, Dariusz M. Wróblewski, Liyuan Zhang

Group 26

Date Committed: 31 January 2024

Contents

Contents 2
Section 1 - Introduction 3
Project Overview 3
Identification of Components and Units 3
Items Covered by Testing 3
Unit Testing 3
Integration Testing 4
System Testing 4
User Acceptance Testing 4
Test Oracles 5
Unit Testing 5
System Testing 5
Human Oracles in User Acceptance Testing (UAT) 6
Testing Approach 7
Section 2 – Testing 9
Sample Test Cases 9
Unit Tests 9
Integration Tests 15
System Tests 20
User Acceptance Tests (UAT) 24
Testing Halt Condition 29
Section 3 - Test Context 30
Fault Severity Classification 30
Environments and Tools 31
Tools Used/Required 31
Documentation and Analysis 32
Testing Role Assignments 32
Structural Testing 33
Code Walkthroughs 33
Conclusion 33
References 34

Section 1 - Introduction

Project Overview

The aim of our project, as assigned by our client, IBM, is to create an online application that will teach young people in education about quantum computing through interactive AR visualisation, narrated by IBM Watson Text-to-Speech. Upon scanning a QR code or typing the URL of the website into a browser search bar, a user will be prompted to enter the educational AR Session. Upon completion of this AR session, the user will be prompted to complete a quiz on the concepts they have learned, and, finally, they will be encouraged to explore IBM Skillsbuild's Quantum Computing course. At any point in the process, the user will be able to ask a chatbot questions about quantum concepts to further their understanding.

Identification of Components and Units

The system is implemented as an integrated web application that will combine various technologies to deliver a seamless educational experience. Key 'must-have' components (as discussed in our requirement specification) include:

- **Website**
    - This serves as the user's entry point. It offers access to the AR sessions and educational content.
- **Unity AR Visualisation**
    - Provides interactive 3D visualisations (models) of quantum computing concepts. This is where the educational chapters will mainly be taught.
- **Watson Text to Speech**
    - This converts written subtitles into spoken narration to make the learning process more interactive.
- **Watson Assistant**
    - This powers an interactive chatbot feature where the user can ask questions and receive information about quantum computers.

These identified components are fundamental to our unit testing efforts, with each considered an individual unit to be tested for functionality and performance.

Items Covered by Testing

The project will be tested with unit, integration, system, and user acceptance tests. Here, we will cover the main items to be tested in unit, integration, and system testing.

Unit Testing

We use unit testing to ensure that each individual component of our project will work correctly in isolation and identify coding errors. Although we will run numerous unit tests beyond what is detailed in Section 2 on each identified component, the following cases will be outlined within our sample tests:

- **Website**
    - The website will be tested for functionality across different web browsers and devices to ensure compatibility and a seamless user experience for all visitors.
- **Unity AR Visualisation**
    - Evaluates the ability to switch between correct models at the appropriate instances

- **Watson Text to Speech**
    - Watson Text to Speech generates a file with the audio correctly
- **Watson Assistant**
    - Watson Assistant provides the appropriate responses to valid and invalid inputs

Integration Testing

During integration testing, we will focus on the interactions between the integrated units and ensure they work together seamlessly. For our project, this involves verifying the successful integration of our established components – the website, Unity AR visualisation, Watson Text to Speech, and Watson Assistant. The main areas of focus will be:

- **Website to Unity AR Visualisation**
    - Testing will ensure that the transitions from the website to the AR educational chapters are smooth and that the user can easily navigate back and forth without issues occurring.
    - Within our document, we specify two possible tests for this.
- **Unity AR Visualisation and Watson Text to Speech Integration**
    - This test will confirm that the 3D models are correctly paired with their respective Watson-generated audio files
- **Watson Assistant Integration with Website and AR Sessions**
    - This test will validate the seamless incorporation of the Watson Assistant into both the website and AR sessions, ensuring users can access the chatbot at any point to ask questions or seek clarification on quantum computing concepts.

System Testing

For system testing, we will focus on both functional and non-functional requirements. The following outlined systems tests (which will be detailed in Section 2) only make up a sample of the full set of system test cases we will deploy on our system:

- **Functional Requirements (FR)**
    1. AR model presentation
    2. Audio description
- **Non-Functional Requirements (NFR)**
    1. Website's traffic scalability and failure-to-load rate
    2. Components response time

We selected each of these sample tests based off their individual alignments with our identified high-priority requirements, as specified in our requirement specification. This ensures that we focus on critical functionalities.

User Acceptance Testing

User Acceptance Testing (UAT) is crucial to ensuring that the final product meets our client's expectations and goals. The UAT will be structured around key user interactions and experiences within the systems:

- **Interactions with the Chatbot**
- **Accessing and Initialising the AR Experience**

- **Completion of a chapter for the AR Experience**
- **Navigation of the website post AR Experience**

Test Oracles

While automated testing will be employed for relevant test cases, based off our project's focus on immersion and user experience, the utilisation of expert user oracles (consisting of both team members and IBM contacts) aligns better with our client's outlined goals. By emphasising our human oracles in both unit and system testing, the project aligns its validation processes with the inherently interactive and user-focused nature of the software. This approach ensures that the final product is not only technically sound but also highly effective in its educational mission, resonating with its audience and meeting the client's objective. Furthermore, the technical complexity of automating verification of certain behaviours within the system is nearly impossible. Therefore, many of the test case's outputs must be verified by a human.

We will outline the oracles for unit testing, system testing and user acceptance testing in the following sections.

Unit Testing

- **Website**
  - Automated testing will verify functional aspects such as load time, link validity and responsiveness.
  - Humans will serve as the oracle for subjective qualities that automated tests are not suited for, such as evaluating the websites usability (e.g. buttons work), design consistency (e.g. follows IBM Carbon Design Principles (IBM, 2024), and content accuracy (e.g. Skillsbuild link is correct).
- **Unity AR Visualisation**
  - Automated testing is implemented here to ensure that 3D models load correctly, and the system can swap between models as expected.
  - Human evaluators can ensure that the content is correct.
- **Watson Text to Speech**
  - Automated testing will confirm that the text-to-speech conversion occurs without errors and generates the correct audio file format.
  - Human listeners will judge clarity of the outputted speech and test the accuracy of pronunciation for domain-specific terms in quantum computing.
- **Watson Assistant**
  - Automated testing will check the chatbot's ability to recognise and respond to a set of predefined queries.
  - Human testers evaluate the chatbot's effectiveness in handling unexpected questions, the relevance of the response, and its ability to maintain a coherent and helpful dialogue.

System Testing

System testing is a crucial phase in our project and is aimed at ensuring the integrated application not only meets the requirements of our client (as outlined in our requirement specification) but also their vision for immersive and interactive education. This phase is designed to validate the overall

functionality and performance of the system while encompassing both individual components and their interactions within the full application context.

For system tests based on both functional and non-functional requirements, our system requires a diverse set of test oracles. These oracles will provide the basis for determining the correctness of these outcomes, aligned with our high-priority requirements.

- **Functional Requirements**
    - For the functional aspects of our system, such as the AR model presentation and audio description, our test oracles will primarily rely on human oracles to assess the immersive quality, educational effectiveness, and user interaction intuitiveness.
- **Non-Functional Requirements**
    - For the non-functional requirements like website scalability and component response times, our approach will combine automated testing tools for objective measurements with expert evaluations for qualitative insights.

In all these cases, the selection of test oracles is guided by the need to ensure that our testing is comprehensive, reliable, and aligned with the project's goals. By combining the precision of automated tools with the nuanced understanding of our expert human evaluators, we aim to ensure that our system not only meets technical specification but also delivers a rich and engaging user experience, consistent with our requirement specifications.

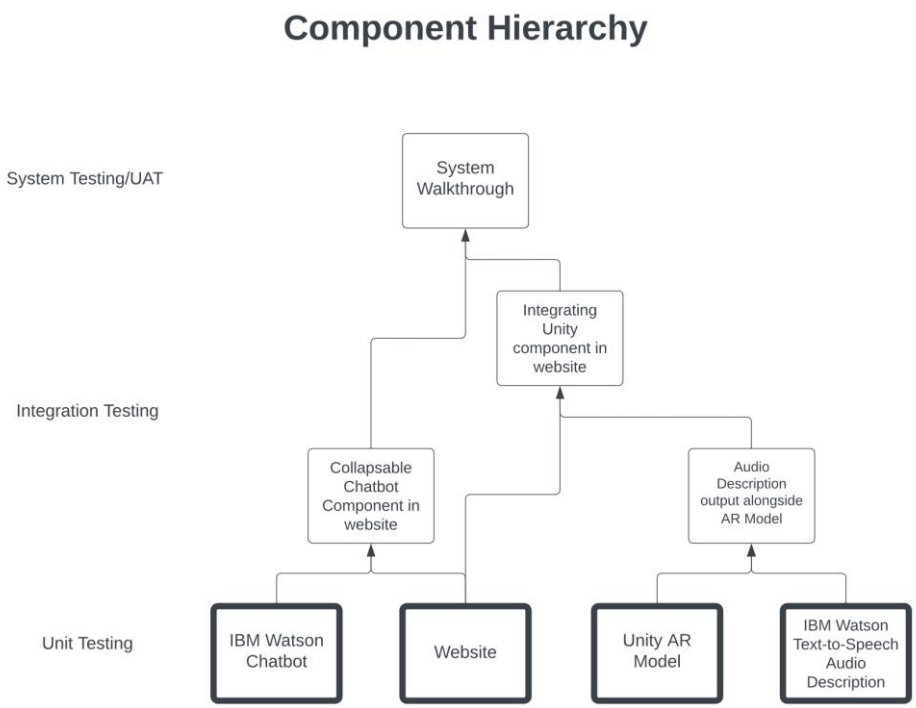
Human Oracles in User Acceptance Testing (UAT)

For this project, we are employing UAT. This is where the end-user/client act as the ultimate test oracle. They validate our system in a real-world context and ensure it meets their expectations and requirements.

- Users engage with and evaluate the application by going through the educational content, interacting with the AR visualisations, communicating with the chatbot, and, finally, end by being redirected to IBM Skillsbuild for further learning.
- Human evaluators can ensure that the content is relevant, engaging and that there is educational value in the system.
- Their feedback on the usability, engagement, and overall satisfaction with the application serves as the primary criterion for acceptance.

This human centric approach ensures that the application meets the technical requirements and client specification and delivers a meaningful and effective educational experience.

Testing Approach



Due to our modular approach and the substantial number of technologies we are utilising, integration testing will play a substantial role in assuring the functionality of our system. Our development process has naturally taken on a bottom-up integration approach with team members working in sub-teams on individual system components that are then gradually integrated together. Some continuous integration practices have also been included in our approach, as we are making use of automatic testing and maintaining a single code repository to increase visibility for all stakeholders and developers.

Within the unit testing phase, we will ensure that component developers and testers remain independent. Effectively, we will each test a component that we have not written ourselves; this is done to prevent any assumptions a developer has made from carrying over from the development phase to the testing phase. As stated above, the project specification was easily broken down into four 'must-have' components that could be worked on concurrently. Each component has at least two team-members (a sub-team) developing it to better improve code quality and redundancy. As we have already divided our team into these individual sub-teams, it will be simple to ensure developer/tester independence, as each sub-team will simply test another sub-team's work. Each member of a sub-team will be trained on two technologies. For example, a team member developing the Watson Chatbot will also be knowledgeable of the Text-to-Speech component, to test it.

Development and Testing Sub-Teams:

Sub-Team	Developing	Testing
1	Unity AR Model	Website

2	Website	Unity AR Model
3	Watson Text-to-Speech	Watson Chatbot
4	Watson Chatbot	Watson Text-to-Speech

Our individual components (such as the Text-To-Speech component or the augmented reality models) will be built first and then their respective unit tests will be created. However, test-driven development (TDD) concepts will be incorporated into our integration testing phase. Our integration tests will be written before we attempt to integrate the components, so that we can develop a more comprehensive overview of how the components will fit together, utilising the benefits of TDD. TDD is not relevant to our system or user acceptance tests, as their oracles consist of sets of expert humans (client contacts and team members) and the majority of system tests and all user acceptance tests cannot be coded.

Although the test cases specified in Section 2 outline our functional testing approach, our structural testing takes the form of several code reviews. Once each test phase has been completed, an informal code walkthrough of all components will be performed at a team review session. Following integration and integration testing, a formal code inspection will take place at another team review session. Another code inspection will take place before the product handover. An in-depth discussion of our code review procedure is in Section 3.

We have also taken regression testing into consideration. As our system is small, we could simply re-run all unit tests when a new fault is discovered and corrected. However, the nature of our system encourages a more optimized approach. During the integration testing phase, a maximum of three components will be tested simultaneously. Therefore, when a new fault is discovered, unit tests will only be re-run for the components involved in the integration test. Furthermore, due to the vastly different functionalities of components, it will often be easy to identify or find a satisfactory approximation of the component set experiencing a fault (e.g. if audio is malfunctioning, the fault is either in the text-to-speech or unity AR components). Whenever a component must be altered in testing phases other than the unit testing phase, all unit tests for that component must be re-run. To further aid regression testing, we will make use of a version control system (GitHub) and maintain thorough testing records (as described in Section 3) for re-use and for our client.

Our test cases within Section 2 (and our testing phases) have been ordered according to the V-Model – beginning with unit testing, progressing through integration and system testing, and ending with user acceptance testing. This ordering is highly compatible with our agile practices and testing approaches. However, contrary to the standard model, there exists a “hidden” development phase situated between the unit and integration testing phases (as the integration tests are written before the components are integrated).

Section 2 – Testing

Sample Test Cases

Following each test case is a results table that details expected results and common failures and how we would classify them (giving us a starting point for how we would approach handling these failures). Within these result tables, the 'Severity of Fault' classification is based on the 'Fault Severity Classification' table in Section 3.

These samples understand the term "coverage" to mean coverage of all aspects of the system (meaning our identified components) and a demonstration of evaluating equivalence classes (as seen in unit_test_01 and unit_test_02) to demonstrate coverage within individual components (as mentioned in lectures).

Unit Tests

Test Case ID	unit_test_01
Description	Checking Watson chatbot generates accurate answers for *valid* questions (based on IBM Skillsbuild Quantum Computing course and relevant to quantum computing).
Relevant Component	IBM Watson Chatbot
Pre-requisites	- Connected to IBM Watson Assistant.
Test Data (Input)	1. 'What is a qubit?' 2. 'Describe quantum entanglement.' 3. 'What is a quantum circuit?' 4. 'Give an overview of Dirac notation.'
Test Procedure	1. Enter valid question into input text box. 2. Click submit. 3. An answer is returned *or* the chatbot times out and displays error message 'please try again' *or* the chatbot asks the user to rephrase the question *or* chatbot fails to respond and a 'report issue' button is displayed. 4. Repeat for other 3 questions.
Expected Result (Oracle)	For each question, an accurate (no misleading/false information) and relevant (must directly address question) answer is displayed. The answer is of a reasonable length (<= 1000 words).
Equivalence Classes	Normal - all valid questions (that have answers) related to IBM Skillsbuild Quantum course.
Test Environment and Tools Required	- IBM Watson Assistant - Roslyn Analyser Linter - Web Browser (Chrome) - Laptop (with internet connection)

Results:

Test ID	unit_test_01		
Result	**Description of Outcome**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A

Possible Output	Chatbot does not phrase answer correctly (e.g. instead of giving overview of Dirac notation, it describes how to use Dirac notation to represent a certain vector).	Failure	Annoying
Possible Output	Chatbot does not recognize one of questions as valid and asks user to 'rephrase the question.'	Failure	Moderate
Possible Output	Chatbot returns answer not relevant to question (e.g. returns explanation of classical state vs quantum state when asked for definition of qubit).	Failure	Major
Possible Output	Chatbot consistently times out (after multiple re-runs of the test) on one of the questions.	Failure	Major
Possible Output	Chatbot consistently times out (after multiple re-runs of the test) on more than one of the questions.	Failure	Extreme
Possible Output	Chatbot consistently times out on all questions.	Failure	Dangerous
Possible Output	Chatbot crashes and does not respond.	Failure	Dangerous
Possible Output	Causes browser to crash.	Failure	Infectious

Test Case ID	unit_test_02
Description	Checking Watson chatbot generates appropriate responses for *invalid* inputs (*invalid* meaning: not relevant to or based on IBM Skillsbuild Quantum Computing course).
Relevant Component	IBM Watson Chatbot
Pre-requisites	- Connected to IBM Watson Assistant.
Test Data (Input)	1. 'What is a dog?' 2. 'Give a definition of machine learning.' 3. Empty String - '' 4. Random string – e.g. 'qwertyuiopasdfghjklzxcvbnm'
Test Procedure	1. Enter question into input text box. 2. Click submit. 3. An answer is returned *or* the chatbot times out and displays error message 'please try again' *or* the chatbot asks the user to rephrase the question *or* chatbot fails to respond and a 'report issue' button is displayed. 4. Repeat for other 3 questions.
Expected Result (Oracle)	For each question, the chatbot returns the same response, which asks the user to rephrase the question.
Equivalence Classes	Error - all inputs that are not related to quantum computing and the IBM Skillsbuild Quantum course.
Test Environment and Tools Required	- IBM Watson Assistant - Roslyn Analyser Linter - Web Browser (Chrome) - Laptop (with internet connection)

Results:

Test ID	unit_test_02		
Result	**Description of Outcome**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	Chatbot attempts to answer question(s) instead of asking user to rephrase question.	Failure	Moderate
Possible Output	Chatbot consistently times out (after multiple re-runs of the test) on one of the invalid questions.	Failure	Major
Possible Output	Chatbot consistently times out (after multiple re-runs of the test) on more than one of the questions.	Failure	Extreme
Possible Output	Chatbot consistently times out on all questions	Failure	Dangerous
Possible Output	Chatbot crashes and does not respond.	Failure	Dangerous
Possible Output	Causes browser to crash	Failure	Infectious

Test Case ID	unit_test_03
Description	Evaluates the AR visualisation component's ability to change between different models during user interaction. It should stop loading new models at the end of the presentation, and it should manage errors related to the model loading.
Relevant Component	Unity AR Visualisation
Pre-requisites	- Multiple AR models are available for testing.
Test Data (Input)	1. Commands to change models during presentation (i.e. button clicks/mouse movements).
Test Procedure	1. Start AR presentation and issue command to change between models (click 'Next' button). Observe transition and ensure next model loads correctly. 2. At the end of the presentation ensure no further models are called upon, and the presentation concludes smoothly. 3. Simulate model loading failure and observe how system handles error
Expected Result (Oracle)	1. System should smoothly transition between models without errors and display them correctly. 2. System should not attempt to load any further models and should conclude the session gracefully. 3. Upon a loading failure, an error message should be displayed.
Equivalence Classes	This is all valid user clicks/mouse movements within system to navigate between all valid models.
Test Environment and Tools Required	- Mobile device (with access to internet) - Unity editor for monitoring and debugging

Results:

Test ID	unit_test_03		
Result	**Description of Outcome**	**Success/Failure Classification**	**Severity of Fault**

Expected	As above.	Success	N/A
Possible Output	Model transitions are not smooth (slow or stuttering) during changes, detracting from the immersive experience.	Failure	Annoying
Possible Output	System fails to load a specific model, yet no error message or alternative action is provided to the user, but system remains responsive.	Failure	Moderate
Possible Output	After the final model is shown, the system attempts to load additional models, leading to a stuck state.	Failure	Major
Possible Output	The AR visualisation component crashes entirely which leads to a complete cessation of the session and necessitating a restart.	Failure	Dangerous
Possible Output	The system crashes or becomes unresponsive, requiring a restart of the AR session.	Failure	Catastrophic
Possible Output	All model transitions fail due to server connectivity issues, with no fallback mechanism. This renders the AR session ineffective.	Failure	Catastrophic

Test Case ID	unit_test_04
Description	Check the website's navigational abilities are functioning correctly as we ensure that menu items and links lead to the correct pages, enhancing user experience and accessibility.
Relevant Component	Website
Pre-requisites	- Website is live and accessible. - All navigation links and menu items are implemented and visible.
Test Data (Input)	1. Click/mouse movements actions on all navigation menu items.
Test Procedure	1. Open the website's homepage. 2. Systematically click on each item you can navigate to. 3. Record the URL of the page that loads upon clicking each menu item. 4. Compare whether it is the desired destination.
Expected Result (Oracle)	Correct navigation to correct pages.
Equivalence Classes	All user navigational actions within website.
Test Environment and Tools Required	- Web browser (Chrome) - ESLint

Results:

Test ID	unit_test_04

Result	Description of Outcome	Success/Failure Classification	Severity of Fault
Expected	As above.	Success	N/A
Possible Output	Taken to wrong page/link.	Failure	Annoying
Possible Output	User is not redirected to new page (i.e. stays on same page).	Failure	Annoying
Possible Output	Website crashes/becomes non-responsive	Failure	Catastrophic

Test Case ID	unit_test_05
Description	Check the Watson Text-to-Speech component returns a clear, accurate audio file when given a valid input with accurate pronunciation of all domain-specific terms to assess audio accuracy.
Relevant Component	Watson Text-to-Speech
Pre-requisites	- Active and functional connection to the Watson Text-to-Speech API - Watson Text-to-Speech service is configured and operational
Test Data (Input)	1. 'This is a quantum computer. Please observe. A qubit is a quantum bit. Dirac notation is useful. The quick brown fox jumps over the lazy dog.'
Test Procedure	1. Feed test data into Text-to-Speech (automated testing of function). 2. Open audio file generated (.wav extension). 3. Listen to file to evaluate quality of response.
Expected Result (Oracle)	- The audio is clear and free of distortion or background noise. - The pronunciation of all words, especially domain specific terminology is accurate. - The speech sounds natural with appropriate pacing and intonation for educational tool.
Equivalence Classes/Coverage	Normal - Sentences with a mixture of technical terms regarding quantum computing and common words to evaluate natural language processing.
Test Environment and Tools Required	- Access to the Watson Text-to-Speech service. - Laptop (with internet connection). - Automated testing scripts (in python or C#). - Audio playback software compatible with .wav files (e.g. Audacity). - Headphones or speakers to assess audio quality.

Results:

Test ID		unit_test_05	
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	Pronunciation is not correct for quantum computing domain-specific terminology (e.g. 'qubit').	Failure	Minor
Possible Output	Poor quality (as judged by human tester) audio file is returned.	Failure	Annoying

Possible Output	Audio file is not returned.	Failure	Extreme
Possible Output	Component does not respond and crashes.	Failure	Dangerous

Test Case ID	unit_test_06
Description	Evaluate the Watson Text-to-Speech component's error handling and response when providing invalid inputs. This includes empty strings, inputs and unsupported characters or symbols and inputs exceeding the maximum character limit
Relevant Component	Watson Text-to-Speech
Pre-requisites	- Active and functional connection to the Watson Text-to-Speech API - Watson Text-to-Speech service is configured and operational
Test Data (Input)	1. Empty string: "" 2. String with unsupported characters " 🐞 🐞 🐞 🐞 " 3. Excessively long string over the API's limit (e.g. 100,000 characters) - can be randomly generated.
Test Procedure	1. For each type of invalid input, use an automated testing script to send the data to the Watson Text-to-Speech component 2. Attempt to generate an audio file for each input 3. Record the systems response and notice any error messages or system behaviour 4. Assess whether the system gracefully handles the invalid input by providing appropriate error feedback or by safely declining to process the input
Expected Result (Oracle)	- For the empty string, the system should return silent audio file or an error message - For the unsupported characters, the system should return an error message specifying that some characters are not supported - For the excessively long string the server should return an error message
Equivalence Classes/Coverage	Error – invalid inputs to Watson Text-to-Speech component. These include: - Empty inputs to test the systems response to lack of content - Inputs with unsupported characters to assess the system's ability to recognise and reject invalid characters - Overly long inputs to evaluate how the system manages inputs that exceed character limits
Test Environment and Tools Required	- Access to the Watson Text-to-Speech service - Laptop (with internet connection) - Automated testing scripts (in python or C#) - Audio playback software compatible with .wav files (e.g. Audacity) - Headphones or speakers to assess audio quality

Results:

Test ID		unit_test_06	
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	System returns a silent audio file for the empty string indicating a lack of error message	Failure	Minor
Possible Output	The system returns a generic error message for all types of invalid inputs failing to specify the nature of the input error.	Failure	Annoying
Possible Output	The component crashes or becomes unresponsive when provided with unsupported characters.	Failure	Dangerous

Integration Tests

Test Case ID	integration_test_01
Description	Checks Watson Text-to-Speech-generated audio file correctly plays alongside a Unity AR Model when no user input interruptions are present.
Relevant Components	Watson Text-to-Speech, Unity AR Visualisation
Pre-requisites	- Watson Text-to-Speech component has passed all unit tests - AR Visualisation component has passed all unit tests - Connected to Watson Text-to-Speech - The browser (Chrome) is equipped with necessary audio playback capabilities (which are enabled) - The website includes the UI elements for audio control - Audio files prepared and inserted into Unity
Test Data (Input)	1. Text for text-to-speech: 'This is what a modern quantum computer looks like. Let us take a closer look at how it works.' 2. Mouse movements
Test Procedure	1. User clicks on icon to change to next AR model (assuming user is not viewing concluding chapter). 2. AR model loads. 3. Audio description begins to play once AR model has fully loaded. 4. The user does *not* interact with the system 5. Audio description finishes. AR model remains for user to interact with.
Expected Result (Oracle)	- Upon clicking on icon, AR model is loaded within 1 minute. - Within 5 seconds, the correct audio description (intended for that model) begins to play alongside AR model. - The audio description concludes. - The AR model remains, giving the user time to explore. - Audio narration is well-synced with subtitles. - Audio quality and control responsiveness are consistent across all tested devices.
Test Environment and Tools Required	- Unity Editor

	- Watson Text-to-Speech - Roslyn Analyser - Mobile phone (with internet connection)

Results:

Test ID	integration_test_02		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	Audio quality is not consistent across all devices.	Failure	Minor
Possible Output	AR does not load within 1 minute but does fully load (eventually).	Failure	Annoying
Possible Output	Audio does not play within 5 seconds but does play (eventually).	Failure	Annoying
Possible Output	Audio plays, but it is not the correct audio for the model	Failure	Moderate
Possible Output	AR Model does not load, but components still respond to some limited inputs (e.g. mouse movements).	Failure	Extreme
Possible Output	Audio does not play.	Failure	Dangerous
Possible Output	Either or both components crash and do not respond to user inputs.	Failure	Dangerous
Possible Output	Causes browser to crash.	Failure	Infectious

Test Case ID	integration_test_02
Description	Checks Watson Text-to-Speech-generated audio file correctly plays alongside a Unity AR Model *when user input interruptions are present*.
Relevant Components	Watson Text-to-Speech, Unity AR Visualisation
Pre-requisites	- integration_test_01 must be successful - Watson Text-to-Speech component has passed all unit tests - AR Visualisation component has passed all unit tests - Connected to Watson Text-to-Speech - The browser is equipped with necessary audio playback capabilities - The website includes the UI elements for audio control - Audio files prepared and inserted into Unity
Test Data (Input)	1. Text for Text-to-Speech (Model 1): 'This is what a modern quantum computer looks like. Let us take a closer look at how it works.' 2. Text for Text-to-Speech (Model 2): 'To understand quantum computers, it is easier to look at classical computers first and make a comparison' 3. Mouse movements.

Test Procedure	1. User clicks on icon to change to next AR model (assuming user is not viewing concluding chapter). 2. AR model (Model 1) loads. 3. Audio description begins to play once AR model has fully loaded. 4. The user clicks on icon ('Next') again to go ahead to the next model (Model 2) *before* the audio description has finished playing. 5. Audio description halts immediately. 6. Next AR model (Model 2) loads. 7. Audio description for this new model begins.
Expected Result (Oracle)	- Upon clicking on icon, Model 1 is loaded. - The correct audio description (intended for that model) begins to play alongside Model 1. - When the 'Next' button is clicked prematurely, the current audio description immediately halts, and the next AR model (Model 2) is loaded, and the new audio description begins to play within 5 seconds.
Test Environment and Tools Required	- Unity Editor - Watson Text-to-Speech - Roslyn Analyser - Mobile phone (with internet connection)

Results:

Test ID	integration_test_02		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	Audio plays for Model 2 (from its beginning), but it is not the correct audio for Model 2.	Failure	Moderate
Possible Output	Audio descriptions from Model 1 and Model 2 overlap, as Model 1's audio description is never halted.	Failure	Major
Possible Output	Model 2 does not load, but components still respond to some limited inputs (e.g. mouse movements).	Failure	Extreme
Possible Output	Audio does not play for Model 2.	Failure	Dangerous
Possible Output	Either or both components crash and do not respond to user inputs.	Failure	Dangerous

Test Case ID	integration_test_03
Description	Check chatbot functions correctly (and completely) within designated website area.
Relevant Components	Website, Watson Chatbot
Pre-requisites	- Website has passed all unit tests - Chatbot has passed all unit tests

Test Data (Input)	1. Mouse movements/clicks 2. Test Data from unit_test_01 3. Test Data from unit_test_02
Test Procedure	1. User clicks on chatbot icon to maximise chatbot interface. 2. User inputs valid and invalid questions (as described in unit_test_01's test data). 3. An answer is returned for each question *or* the chatbot times out and displays error message 'please try again' *or* the chatbot asks the user to rephrase the question *or* chatbot fails to respond and a 'report issue' button is displayed. 4. User clicks on chatbot icon again and the chatbot interface is minimised.
Expected Result (Oracle)	User opens chatbot, gives various input and receives correct responses for those inputs, user then closes chatbot.
Test Environment and Tools Required	- ESLint - Watson Assistant - Roslyn Analyser - Laptop with internet connection

Results:

Test ID	integration_test_03		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	The report issue button fails to function or does not log the reported issue correctly	Failure	Annoying
Possible Output	The chatbot repeatedly asks the user to rephrase questions that should be valid (indicating poor natural language processing)	Failure	Moderate
Possible Output	The chatbot fails to open or maximise upon user interaction with the chatbot icon	Failure	Major
Possible Output	The chatbot times out frequently failing to provide any response to user queries	Failure	Major
Possible Output	The chatbot provides incorrect or irrelevant answers to user queries	Failure	Extreme
Possible Output	Chatbot component crashes, but website still responds.	Failure	Extreme
Possible Output	Website component crashes, but chatbot component still responds.	Failure	Extreme
Possible Output	Browser crashes when attempting to use chatbot (at any time during the procedure).	Failure	Infectious

Test Case ID	integration_test_04
Description	Validates the integration between the Unity AR model component and the website to ensure users can interact with AR models directly from the web interface without issues. The AR models must be accessible

	and fully functional within the website environment, providing an interactive and immersive user experience without needing to leave the website needing to leave the website or use additional software.
Relevant Components	- AR Visualisation component - Website
Pre-requisites	- The Unity AR model component passes all unit tests. - The website passes all unit tests. - The website has designated areas or pages where AR models are to be displayed. - Necessary permissions and plugins (especially Web player) are enabled on the web browser to support AR content.
Test Data (Input)	1. Mouse movements (e.g. selecting a chapter from the website, moving through the chapter, and moving around the environment)
Test Procedure	1. Navigate to the chapter selection of the website and select a chapter. 2. Find the AR marker with phone camera which will spawn the model. 3. Go through the chapter, attempting to move around. 4. Attempt to trigger any specific actions available for the current model 5. Exit the AR session and observe the transition back to the website 6. Repeat the process across different devices
Expected Result (Oracle)	- The models change without significant delays - The user interactions are responsive, and function as intended - Exiting the AR session returns the user to the previous web content without errors or disruptions - The integration performs consistently across all supported devices and browsers
Test Environment and Tools Required	- Web hosting - Web Browser (Chrome) - Various mobile devices (Apple and Android smartphones, tablets with internet connection) - Web performance monitor (such as NetLimiter). - N.B. Various devices are needed to test compatibility to test responsiveness.

Results:

Test ID	integration_test_04		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	Model spawning is slow or sluggish	Failure	Annoying
Possible Output	Models load but with noticeable delays, impacting the user experience	Failure	Moderate
Possible Output	AR content fails to load on certain devices despite passing compatibility checks	Failure	Dangerous

Possible Output	The AR session disrupts website functionality, preventing users from navigating away or accessing other site content	Failure	Catastrophic
Possible Output	Transitioning back to the website after exiting the AR session causes the web browser to freeze or crash	Failure	Infectious

System Tests

Test Case ID	sys_test_01
Description	Checks website failure rate – website should not fail to load more than once for every 5000 access requests.
Relevant Components	Website, IBM Cloud
Related Requirement Specification Requirement(s)	NFR1.2
Pre-requisites	- Website will correctly load for some access requests - All individual components are integrated into website. - Reliable internet connection.
Test Data (Input)	1. 'Refresh website' command
Test Procedure	1. Enters website address into browser search bar or refreshes page. 2. Website processes user access request (and scales accordingly). 3. Website (and all components within its framework) loads successfully *or* website fails to load and error message is displayed. 4. This is repeated 5000 times.
Expected Result (Oracle)	- The above steps are repeated a suitable number of times (e.g. 20) to get an accurate understanding of the failure rate and identify outliers. - Website loads correctly at least 4999 out of 5000 times for all runs of the procedure detailed above.
Test Environment and Tools Required	- Web browser (Chrome) - Mobile device or laptop (with internet connection)

Results:

Test ID	sys_test_01		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above	Success	N/A
Possible Output	Website fails more often than once every 5000 access requests, but less than or equal to 10 times for every 5000-access request.	Failure	Minor
Possible Output	Website fails more often than once every 5000 access requests, but less than or equal to 500 times for every 5000	Failure	Dangerous

	access requests (<= 1% of access requests)		
Possible Output	Website fails more often than once every 5000 access requests and more than 500 times for every 5000 access requests (> 1% of access requests)	Failure	Catastrophic

Test Case ID	sys_test_02
Description	Checks that the Unity AR presentation works correctly with audio when navigating back and forth between chapters that do not begin or end the presentation and that the user can move around each model displayed, viewing it from different angles.
Relevant Components	Unity AR Visualisation
Related Requirement Specification Requirement(s)	FR1.1.3, FR1.1.4
Pre-requisites	- Unity AR component fully integrated into website - Unity AR component has passed all unit tests - Text-to-Speech component has passed all unit tests
Test Data (Input)	1. Mouse movements
Test Procedure	1. User clicks on 'Next' button. 2. Different model loaded and displayed. Audio description begins. 3. User walks around model, viewing it from different angles. 4. User clicks on 'Previous' button. 5. Different model loaded and displayed. Audio description begins 6. User walks around new model, viewing it from different angles.
Expected Result (Oracle)	When user clicks on 'Next,' the next chapter in the AR presentation begins, a new model is displayed, and the correct audio description begins. When the user clicks on 'Previous,' the previous chapter in the AR presentation is replayed, the previous model is displayed, and correct audio description begins. The user can "look closer" at the model and the displayed model adapts to user movements (e.g. different angles, resolution of details, etc.).
Test Environment and Tools Required	- Web browser (Chrome)

Results:

Test ID	sys_test_02		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	
Possible Output	There is a significant delay (>5 second) in Text-to-Speech audio	Failure	Minor

	being output following a new model being loaded.		
Possible Output	Model does not promptly adapt to user movements, resulting in awkward AR experience.	Failure	Annoying
Possible Output	Wrong model is displayed when 'Next' button is clicked (based on ordered sequence of chapters).	Failure	Moderate
Possible Output	Wrong model is displayed when 'Previous' button is clicked (based on ordered sequence of chapters).	Failure	Moderate
Possible Output	Wrong audio is played when 'Previous' or 'Next' button is clicked (based on ordered sequence of chapters).	Failure	Moderate
Possible Output	No model is displayed when 'Next' button is clicked.	Failure	Major
Possible Output	No model is displayed when 'Previous' button is clicked.	Failure	Major
Possible Output	No audio description is output alongside model when 'Next' or 'Previous' button clicked.	Failure	Major
Possible Output	Model does not "adapt"/update to user movements and remains static on screen	Failure	Extreme
Possible Output	AR presentation crashes and does not respond to user input when any button is clicked.	Failure	Dangerous

Test Case ID	sys_test_03
Description	Checks Unity AR presentation works correctly for navigating between chapters that begin or end presentation (i.e. first and final chapters).
Relevant Components	Unity AR Visualisation
Related Requirement Specification Requirement(s)	FR1.1.3, FR1.1.4
Pre-requisites	- Unity AR component fully integrated into website - Unity AR component has passed all unit tests. - Text-to-Speech component has passed all unit tests
Test Data (Input)	1. Mouse movements
Test Procedure	1. User clicks on 'Previous' button when viewing first chapter. 2. Model/audio does not change. 3. User clicks on 'Next' button when viewing first chapter. 4. Different model loaded and displayed. Audio description begins. 5. User clicks on 'Previous' button when viewing concluding chapter. 6. Different model loaded and displayed. Audio description begins 7. User clicks on 'Next' button when viewing concluding chapter. 8. User is presented with menu.
Expected Result (Oracle)	When user clicks on 'Previous' in first chapter, nothing happens. When user clicks on 'Next' in first chapter, user is taken to next chapter, new model and audio loaded. When user clicks on 'Previous'

	in concluding chapter, user is taken to next chapter, new model and audio loaded. When user clicks on 'Next' in concluding chapter, user is presented with menu.
Test Environment and Tools Required	- Web browser (Chrome)

Results:

Test ID	sys_test_03		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	There is a significant delay (>5 second) in Text-to-Speech audio being output following a new model being loaded.	Failure	Annoying
Possible Output	Unity component attempts to display model when 'Previous' button clicked in first chapter or 'Next' button is clicked in last chapter.	Failure	Moderate
Possible Output	Text-to-Speech component attempts to play audio when 'Previous' button clicked in first chapter or 'Next' button is clicked in last chapter.	Failure	Moderate
Possible Output	Wrong model is displayed when 'Next' button is clicked when in first chapter or when 'Previous' button is clicked when in concluding chapter (based on ordered sequence of chapters).	Failure	Major
Possible Output	Wrong audio is played when 'Next' button is clicked when in first chapter or when 'Previous' button is clicked when in concluding chapter (based on ordered sequence of chapters).	Failure	Major
Possible Output	No model is displayed when 'Next' button is clicked in first chapter or when 'Previous' button is clicked in concluding chapter.	Failure	Major
Possible Output	No audio description is played when 'Next' button is clicked in first chapter or when 'Previous' button is clicked in concluding chapter.	Failure	Major
Possible Output	AR presentation crashes and does not respond to user input any button is clicked.	Failure	Dangerous

Test Case ID	sys_test_04
Description	Ensures audio descriptions are generated, output correctly with the AR model, and that subtitles match the speech speed.
Relevant Components	Watson Text-to-Speech, Subtitle System

Related Requirement Specification Requirement(s)	FR1.2.1, FR1.2.2, FR1.2.3
Pre-requisites	All relevant components are integrated.
Test Data (Input)	1. Give string to generate audio description file "the fox jumped over the lazy dog". 2. Give empty string ""
Test Procedure	1. Trigger the AR course. 2. Confirm the audio description file plays alongside the AR model. 3. Observe and confirm that subtitles are displayed in sync with the audio.
Expected Result (Oracle)	- Audio descriptions and subtitles work in sync with the AR model, providing a synchronised multimodal experience.
Test Environment and Tools Required	- Hardware with audio playback and subtitle display capability, e.g. Android phone - Web browser (chrome).

Results:

Test ID	sys_test_04		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above	Success	N/A
Possible Output	Website fails to generate audio description.	Failure	Minor
Possible Output	Website fails to generate audio description in sync with the audio.	Failure	Annoying
Possible Output	Subtitles are not displayed in sync with audio description	Failure	Annoying
Possible Output	Subtitles are not displayed alongside audio description	Failure	Moderate
Possible Output	Website causes a system crash due to a failure of handling audio description.	Failure	Catastrophic

User Acceptance Tests (UAT)

Test Case ID	uat_01
Description	Checks chatbot performs correctly and answers user's question with satisfactory speed and accuracy.
Relevant Components	IBM Watson Chatbot, Website
Client Specification	"Use Watson Assistant to allow the user to ask questions around quantum computing"
Pre-requisites	- Chatbot functions correctly as isolated component - all unit tests on chatbot are successful. - Chatbot fully integrated into website. - Chatbot is accessible from AR component.

Test Data (Input)	1. Question: 'What is a qubit?'
Test Procedure	1. Chatbot interface is maximized by user. 2. User enters relevant quantum computing question. 3. Chatbot responds with accurate answer *or* asks for more information/a different question *or* times out. 4. User minimizes chatbot interface 5. This procedure is repeated when user is "within" AR course and other part of website.
Expected Result (Oracle)	1. Chatbot appears after user clicks on icon and icon disappears – the system is intuitive for the user, and they can locate the chatbot easily. 2. An accurate (no incorrect/misleading statements) and brief description of a qubit (<1000 words) is output after question entered. 3. Chatbot disappears and icon reappears after user clicks on minimization button. 4. User is satisfied with accurate answer and now understands a new concept of quantum computing. They are satisfied with the speed of response.
Test Environment and Tools Required	- Web browser (Chrome) - IBM Watson Assistant

Results:

Test ID	uat_01		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above	Success	N/A
Possible Output	Chatbot fails to appear when icon clicked (user does not have access to component)	Failure	Moderate
Possible Output	Chatbot fails to disappear when icon clicked (confusing user)	Failure	Major
Possible Output	Chatbot times out (on multiple runs of test), contributing to poor user experience	Failure	Major
Possible Output	Chatbot crashes and fails to respond (user experience is not satisfactory).	Failure	Dangerous

Test Case ID	uat_02
Description	Verifying the web application's ability to open and navigate to the AR model using a QR code. This should be intuitive and easy for the user to do.
Relevant Components	Website, AR Model
Client Specification	All specifications related to AR Quantum Model (see FR1)
Pre-requisites	- Website and AR Model are deployed and accessible. - Website passes all unit tests - AR Visualisation component passes all unit tests
Test Data (Input)	1. QR code provided for AR model access.
Test Procedure	1. Scan the QR code with a mobile device.

	2. Observe if the web application is launched and navigates to the AR model. 3. Confirm that the AR model loads successfully
Expected Result (Oracle)	- Upon scanning the QR code, the web application opens, navigates to, and loads the AR model *without delay or error*.
Test Environment and Tools Required	- Mobile Device with camera, QR code scanning capability, internet access - Web browser (chrome).

Results:

Test ID	uat_02		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above	Success	N/A
Possible Output	AR model loads but with significant delay, impacting user engagement and experience.	Failure	Annoying
Possible Output	Web application launches but fails to navigate to the correct AR model leading to user confusion.	Failure	Moderate
Possible Output	QR code redirects correctly but the AR model does not load, displaying an error message instead.	Failure	Dangerous
Possible Output	QR code scan does not redirect to the web application, failing to initialise the AR model.	Failure	Dangerous

Test Case ID	uat_03
Description	Testing the completion of an AR chapter by the user.
Relevant Components	AR Model, Text-to-Speech, Interactive Quiz
Client Specification	FR1.1, FR1.2, FR3.1
Pre-requisites	- All components are functional and integrated within the web application - AR Model has passed all unit tests - Watson Text-to-Speech component has passed all unit tests
Test Data (Input)	1. User beginning an AR chapter.
Test Procedure	1. Initiate the AR chapter through the web application after scanning the QR code. 2. Interact with the AR and accompanying features model (such as pressing buttons) to complete the chapter. 3. Engage with the interactive quiz at the end of the AR chapter.
Expected Result (Oracle)	The user should be able to go through the AR experience, understand the content via text-to-speech explanations without issues.
Test Environment and Tools Required	- Mobile device or laptop (with internet access) and has AR-capable hardware - Audio output (speakers/headphones) - Web Browser (Chrome)

Test ID	uat_03		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above	Success	N/A
Possible Output	User interactions trigger unexpected actions (model changing).	Failure	Annoying
Possible Output	Interactive quiz at the end of the chapter does not load or fails to register user response.	Failure	Extreme
Possible Output	User interactions with the AR model are unresponsive.	Failure	Dangerous
Possible Output	The web application becomes unresponsive or crashes during the AR chapter, preventing the user from completing it.	Failure	Catastrophic

Test Case ID	uat_04
Description	Verifies that upon completion of the AR model presentation the user can successfully conclude the session and seamlessly navigate to either the chatbot or further inquiries to the skillsbuild platform for additional learning resources
Relevant Components	- AR Model Visualisation - Website - Watson Assistance Chatbot Integration - SkillsBuild Platform Link
Client Specification	The user must have a smooth transition from the AR model presentation to either engaging with the chatbot for questions *or* proceeding to the SkillsBuild platform for further educational content, enhancing the learning experience without technical interruptions.
Pre-requisites	- AR Model passes all unit tests - Website passes all unit tests - Chatbot passes all unit tests - AR chapters are completed - The chatbot is integrated into the website - The user has completed an AR model presentation session
Test Data (Input)	1. Completion of AR model presentation 2. Selection inputs by the user to navigate to the chatbot or IBM SkillsBuild (click/mouse movements) 3. Random question for chatbot (e.g. 'Hello?')
Test Procedure	1. Engage with the AR presentation on the website until completion 2. Upon completion observe the options presented for further actions 3. Select the option to engage with the chatbot and verify the transition to the chatbot interface 4. Initiate a query (random) with the chatbot and confirm it responds appropriately.

	5. Return to the AR model completion screen and select the option to navigate to the SkillsBuild platform 6. Confirm that the link correctly redirects to the skillsbuild platform, and the website loads successfully
Expected Result (Oracle)	- User is presented with clear options to either engage with the chatbot *or* proceeds to the skillsbuild platform upon completing the AR model presentation - Selection of the chatbot option leads to a functional chat interface where the user can interact and speak with it without issues - Selection of the SkillsBuild link correctly redirects the user to the SkillsBuild platform, where they can access further learning materials - Transition between the AR model
Test Environment and Tools Required	- Web browser (Chrome) - Unity editor - Mobile device (with internet connection) - Watson Assistant

Results:

Test ID	sys_test_03		
Result	**Description**	**Success/Failure Classification**	**Severity of Fault**
Expected	As above.	Success	N/A
Possible Output	Options for further actions post AR presentation are not clearly presented or are missing.	Failure	Moderate
Possible Output	The chatbot interface loads, but responses to queues are delayed or irrelevant.	Failure	Moderate
Possible Output	Selecting the chatbot option does not lead to the chat interface, or the chatbot fails to initiate.	Failure	Major
Possible Output	Selecting the SkillsBuild link results in a broken link or incorrect redirection hindering access to additional learning material.	Failure	Major
Possible Output	Transition from the AR presentation either the chatbot or Skillsbuild platform causes the web application to freeze or crash	Failure	Dangerous
Possible Output	Both the chatbot interaction and skillsbuild redirection fail, completely blocking the user's path to further engage and learn.	Failure	Catastrophic

Testing Halt Condition

We will halt our testing when all system tests based on non-functional requirements meet agreed upon performance levels (e.g. all components respond within an agreed upon period) and all other tests do not produce failures that are caused by faults with classifications greater than 'Major.' Furthermore, all unit tests must be successful to meet our standards for adequate coverage. Our success rates (e.g. website loading failures) and boundaries may not be perfect; this is acceptable (if our performance expectations are met), as our system is not life critical.

Section 3 - Test Context

Fault Severity Classification

The following matrix helps in prioritizing issues based on their impact on system functionality and user experience. It recognises that not all faults are equal and provides a framework to allocate resources effectively while outlining what constitutes a certain level of failure. This ensures that the most critical issues are addressed first.

It is useful to state that a component of assessing this severity was how quickly it could be fixed. Where code 1 can be fixed in less than 10 minutes, code 8 could take days, or weeks, to fully solve.

Code	Severity Level	Effect on System	Potential User Impact	Example(s)
1	Minor	Affects small, non-critical component(s), overall functionality of system unaffected.	Minor annoyance, even unnoticed	Spelling errors
2	Annoying	Affects navigational or minor interactive component(s), overall functionality of system unaffected.	Annoying, disrupts flow but not critical functionality	Links to wrong pages, navigational dead ends without a clear return path
3	Moderate	Affects partial functionality of component(s)	Noticeable but not obstructing core functionality	Noncritical buttons not working correctly, audio not outputting for the correct model
4	Major	Affects major functionality of component(s)	Significant impact on user experience	AR Model not displayed (but still responds), chatbot consistently takes too long to respond to question (times out)
5	Extreme	Affects most functionality of component(s)	Severe impact, user might not be able to complete certain tasks	Several occurrences of errors classified between 1 and 4
6	Dangerous	Affects critical functionality of component(s)	Critical impact, user unable to use major features	Major component(s) (such as text-to-speech or AR model) shuts down, major component(s) frequently becomes unresponsive. Critical buttons not working.
7	Catastrophic	Affects entire system	System unusable, complete failure of user experience	Entire site shuts down, site frequently becomes unresponsive
8	Infectious	Causes severe issues beyond the web application, affecting the user's device	System crash, potential data loss, affects user's trust and safety	Causes user's device or system to crash, potential data loss or security vulnerabilities

Minor: These issues, like spelling errors, are minor in nature and don't affect the functionality of the system. They can be irritating but will have negligible effect on the overall user experience. They're easy to fix and low impact, making them low priority.

Annoying: Issues classified as "Annoying", such as wrong page links or navigational dead ends, are more disruptive than minor issues. The idea here is that they can frustrate users by interrupting the flow of interaction, all the while they don't impede core functionalities. These are higher priority than minor issues because they affect the user experience more directly but are not critical to the overall system operation.

Moderate: These faults, like noncritical buttons malfunctioning or audio issues, affect the functionality of specific components but don't prevent the overall application from working. They are noticeable to the user and can hinder the functionality of certain features.

Major: Issues like these would impact the core functionality of specific components, taking away from the user experience typically quite significantly. These problems would need prompt attention, as they can frustrate users and negatively impact the perception of the application.

Extreme: When most functionalities of a component are affected, the severity is classified as extreme. We believe this would mostly occur as a combination of several different code 1-4 issues at the same time. These issues can prevent users from completing tasks and severely impact the application's usability. Quick resolution would be advisable.

Dangerous: This level is for issues that shut down critical components like text-to-speech or AR models or make them unresponsive. Such problems can make key features of the application unusable effectively making our application equally unusable.

Catastrophic: This category consists of issues that render the entire system unusable. These faults would lead to a complete failure of the user experience and system. This would require immediate attention to avoid long-term damage to the project's reputation and user base.

Infectious: The most severe category, "Infectious", is reserved for faults that extend beyond the application, potentially causing system crashes, data loss, or data leaks on the user's device. These are extremely unlikely given our structure; however, they are critical as they transcend the boundaries of the application, potentially compromising the security of the user's device. They require the most urgent and thorough response to prevent serious repercussions.

Environments and Tools

Tools Used/Required

Web Browser: Google Chrome is chosen as the primary browser due to its widespread use and compatibility with Android devices. This is the case as our AR application is primarily targeted towards Android users. Chrome's extensive developer tools makes it well suited for testing web-based AR applications like ours.

Operating System: While the application is web-based and largely platform-independent, the initial focus is on Android, due to Google ARCore's compatibility constraints. Having said that, we will be testing it on both iOS and Android OS.

AR Development Tools: Google ARCore is used for AR development and offers plenty of in-built testing tools. What makes it stand out is its compatibility with Android and web-based platforms, which aligns with our project's requirements. Unity is also heavily employed, particularly for integrating AR features into the web application. Unity's play mode will be utilised to test features.

IBM Tools: IBM Watson Assistant and Watson Text-to-Speech are mostly utilised to enhance user interaction and accessibility. These tools will provide auditory explanations and, possibly, facilitate the interactive Q&A segment. We will be hosting our system on IBM Cloud, which will provide us with the ability to simulate traffic and measure load times accurately.

Linting Tools: Will be utilised for maintaining code quality and overall consistency. C#'s Roslyn Analyser and JavaScript's ESLint will be utilised.

Hardware: Testing will be conducted on both Android and Apple phones; in addition, we will also test our system on laptops to ensure compatibility and responsiveness across different devices. However, the system will not be available for laptops in the final product. These devices must be run on the most recent OS (Operating Systems) versions for their respective platforms. The choice of phones and laptops for testing reflects our intention to cater to a diverse user base, ensuring the application's performance across various device types.

Documentation and Analysis

Results Documentation: The outcomes of tests will be recorded automatically, where appropriate. This will involve recording data in a database, capturing warning boxes, and generating .txt files for record-keeping. The results of tests that require human oracles will be captured as screenshots or videos and stored in an organised file system in our Github repository.

Version Control System (VCS): GitHub will be utilised for tracking changes, maintaining previous test results, and documenting failures. This ensures a comprehensive history of the development process, allowing for easy version backtracking should a severe enough error arise.

Traffic Simulation: We plan to simulate traffic on IBM Cloud to test scalability. Though full access to IBM Cloud is currently unavailable, this aspect is under consideration for demonstrating our preparedness to test under varying loads.

Testing Role Assignments

Role	Name
UAT Team Leader	Millicent Riordan
Development Team Leader	Sorcha Corrigan
User Representative	Darek Wroblewski
Tester 1	Aakif Rehman
Tester 2	Ziyi Huang
Tester 3	George Gaumond
Tester 4	Liyuan Zhang

As well as assigning sub-team roles for unit testing (see Section 1), we have also allocated roles to team members for the user acceptance testing phase. We have decided to have several different (and independent) testers, as it's beneficial to have multiple testers to cover different testing scenarios and to bring diverse perspectives to the testing process.

Structural Testing

Code Walkthroughs

We will schedule regular walkthroughs during the testing phases to evaluate various parts of the code function correctly for all situations. At the conclusion of each test phase, we will have an informal walkthrough of our code in a team review session.

Code inspections

We have elected a moderator for the code inspections; this will be Sorcha, as she's the Development Team Leader. Firstly, we will create a checklist based on coding standards outlined by the best practices manuals of the technologies we are using, and this checklist will also incorporate parts of our project requirements. Two code inspections will take place over the course of this project. One inspection will take place at the end of the integration phase to ensure "on track" functionality and one just before the final product handover to ensure further design modifications have not damaged functionality.

In summary, walkthroughs will be an ongoing, less formal review process to catch issues early, while code inspections will be more rigorous and structured, typically occurring after significant milestones like the formal "finishing" of components. Both are integral to the development cycle and will be documented thoroughly with the help of version control systems.

Conclusion

Our test plan details our thought process when it comes to testing our project. We aim to present an up-to-date version of this test plan, as well as all test records, to our client during our product handover, highlighting the extensive coverage of our tests and providing evidence of our success in building this system. This works to better establish confidence in our product's functionality.

References

- IBM (2024). *Carbon Design System*. https://carbondesignsystem.com/ (Accessed: 31 January 2024).
- Jama Software (no date a) *Best Requirements for Verification and Validation in Product Development.* Available at: https://www.jamasoftware.com/requirements-management-guide/requirements-validation-and-verification/best-practices-for-verification-and-validation-in-product-development (Accessed: 27 January 2024).
- Jama Software (no date b) *Requirements Verification and Validation for Product Teams.* Available at: https://www.jamasoftware.com/requirements-management-guide/requirements-validation-and-verification/requirements-verification-and-validation-for-product-teams (Accessed: 27 January 2024).
- Jama Software (no date c) *What is MBSE? Model-Based Systems Engineering Explained.* Available at: https://www.jamasoftware.com/requirements-management-guide/systems-engineering/what-is-mbse-model-based-systems-engineering-explained (Accessed: 27 January 2024).
- Unity (2024). *Configurable Enter Play Mode.* Available at: https://docs.unity3d.com/Manual/ConfigurableEnterPlayMode.html (Accessed: 31 January 2024).

N.B. Diagrams were created using LucidChart (https://www.lucidchart.com/pages/).