User Manual 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: 5.3.24

Contents

Introduction

Project Summary

IBM, as represented by John McNamara, has tasked our group to develop software that teaches the concepts of Quantum Computing to a new, younger audience. The motivation behind this project is rooted in the desire to bridge the gap between the transformative potential of quantum technology and the prevailing perception of it as an elusive and almost science-fictional domain. The goal of the proposed solution is to develop an approachable and educational website that displays an interactive lesson on the fundamentals of quantum computing in an AR environment. This innovative approach aims to inspire and educate students to guide them towards further IBM resources, most notably IBM SkillsBuild.

The project can be broken down into four main components, these being:

- The main website component
    - This is the primary platform through which the user will engage with the content of our software, providing access to the various resources and interactive elements.
- The Unity AR Visualisation
    - This component utilises Unity to create the AR experiences, which enables users to interact with the quantum computing concepts in a dynamic and immersive environment.
- The Watson Assistant
    - This is an AI powered chatbot which will facilitate real-time interactive Q&A sessions. This will allow users to inquire about and clarify the quantum concepts presented during the AR lessons.
- The Watson Text-to-Speech
    - This is used alongside the AR lessons with audible narration to convert written information into spoken words, increasing the accessibility of the software and creating an engaging auditory learning experience.

Product Access

Our final product is accessible at this URL:

https://atlantaemrys2002.github.io/ARQuantum/

All code for this product is accessible at:

https://github.com/COMP2281/software-engineering-group-26/tree/main

Please contact us via email (millicent.h.riordan@durham.ac.uk) if there are any problems gaining access to the final product or GitHub repository.

Before the final project handover, our code and final product may be subject to changes/updates. We will update our copy of the User Manual (stored in the Documentation directory in our GitHub repository) to reflect any changes in requirements' statuses, etc.

Dependencies

Please note, if you wish to host this system, you may need to access these technologies:

- Carbon React
- GitHub (especially, GitHub Pages)
- IBM Watson Assistant (Software)
- IBM Watson Text-to-Speech (Software)
- Needle Engine
- Next.js

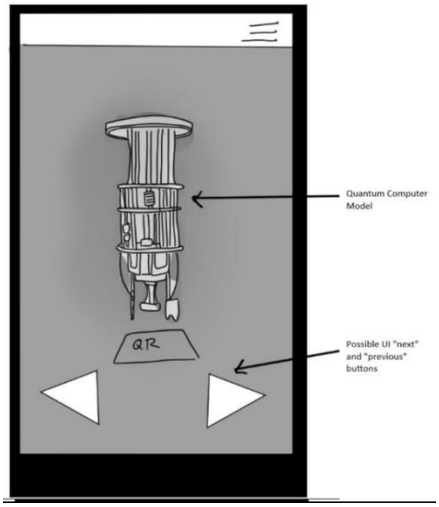
- Node.js
- Vite
- Yarn

Requirement Description and Status

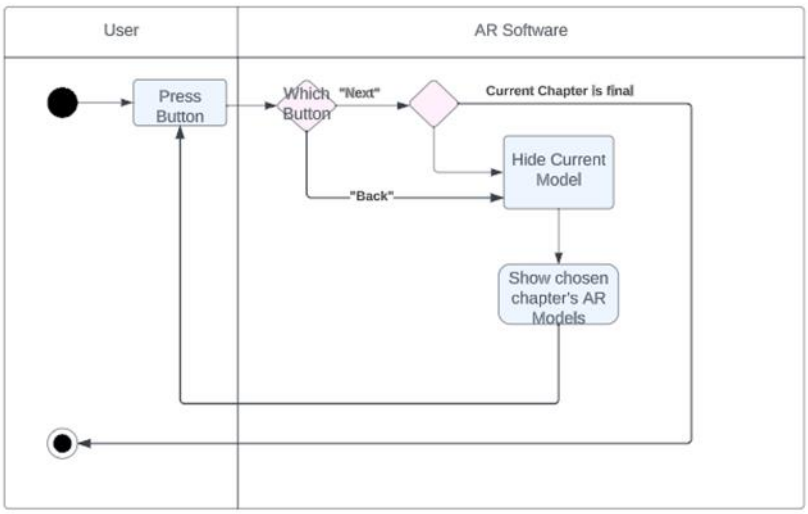
ID and Title	FR1.1.1 - AR Quantum Computer Model Displayed		
Status	Same	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Must Have
(Original) Description	Event driven: Upon the user's mobile device detecting a QR code marker, the system shall activate and display the AR model of a quantum computer		
Justification for Extent of Fulfillment	Expected Results Met		
Dependencies	N/A		
Expected results	The correct model shall be displayed in a timely manner and in a location intuitive to the user.		

ID and Title	FR1.1.2 - AR Quantum Computer Model Orientation and Scale		
Status	Same	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Should Have
(Original) Description	State-driven: While AR Model is displayed, it should be oriented and scaled such that it reflects a real-life Quantum Computer.		
Justification for Extent of Fulfillment	Expected results met.		
Dependencies	FR1.1.1		
Expected results	The Quantum Computer Model should be vertically upright like it would appear in real life. If the real scale of the computer isn't possible it should maintain realistic proportions and size difference explained.		

ID and Title	FR1.1.3 - AR UI Information Buttons		
Status	Same	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Should Have
(Original) Description	State-driven:		

	While the AR quantum model is being displayed, interactive buttons should also be displayed, allowing the user to interact with the AR environment and model. Buttons will include options to go to the next or previous learning 'chapter'.
Visual Representation	
Justification for Extent of Fulfillment	Expected results met.
Dependencies	FR1.1.1
Expected results	If the current working chapter isn't the first one, there will only be the option to proceed. Otherwise, the option should be to go either to the next or previous section. If the current 'chapter' is the final one, the next area shown should be a prompt for the quiz section or a link to SkillsBuild (as requested by the client).

ID and Title	FR1.1.4 - AR Dynamic Presentation		
Status	Same	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Must Have
(Original) Description	Event Driven: When a UI Information Button is clicked, the AR Quantum Computer Model shall be swapped with a more relevant model (and audio description) for a chosen topic.		

Visual Representation	
Justification for Extent of Fulfillment	This requirement has been met with a caveat that certain content is missing. This means the application successfully delivers an AR dynamic presentation. However, the content of the presentation isn't complete in certain areas. So, while the description criterium has been satisfied, a portion of the expected results have not been satisfied as of current.
Dependencies	FR1.1.1, FR1.1.3
Expected results	After the button is clicked ('Next' or 'Previous'), the current model being displayed shall be replaced by a new model, unless the current 'chapter' displayed is the last in the sequence. Relevant models shall include simulations depicting: Quantum Physics, Traditional vs. Quantum Computers, how a Quantum Computer Works, The Future of Quantum Technology. A relevant IBM Watson Text-to-Speech audio for the new model shall also be output alongside the model. Each model must meet the same requirements associated with the main Quantum Model.

ID and Title	FR1.1.5 - Exporting AR Application to Web		
Status	Changed	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Should Have
(Original) Description	Ubiquitous Requirements: The AR application shall be made using Google ARCore technologies such that it can be exported to the website.		
(New) Description	The AR application shall be built using Needle Engine such that it can be exported as a WebXR application that can be hosted on a Vite server and loaded into an iFrame on the main website.		
Justification for Extent of Fulfillment	Updated requirement met - we updated this requirement, as Unity is currently unable to create AR WebGL builds. This third-party package – Needle Engine – allows us to export Unity as WebXR builds, as well as improving our functionality (allowing us to see the model in both AR and VR) and allowing us to conform to our client's requests (no downloads are required). Furthermore, Image Tracking, audio, and interactive functionalities are all maintained.		

Dependencies	FR1.1.1, FR1.1.3
Expected results	The expected outcome is the successful creation of an AR application utilising Needle Engine and WebXR technologies, allowing for seamless exportation to the website. This exportation capability enables users to access and experience the AR application directly through web browsers (with the recommended browser being Chrome, conforming with our aim of primarily targeting Android users, whilst maintaining compatibility with iOS).
Exception handling	No specific exception handling procedures are outlined for this requirement. However, it is advisable to implement robust error-checking mechanisms to address potential issues during the exportation process. This may include providing informative error messages and logging functionalities to facilitate troubleshooting in the event of exportation failures. Additionally, it may be worth testing the application with multiple web-browsers (e.g., Chrome, Firefox, Safari) to check its functionality capabilities cross-platform.

ID and Title	FR1.2.1 IBM Watson Text-to-Speech Audio Description		
Status	Changed	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Should Have
(Original) Description	Event-driven: When text is displayed on the screen to describe what is happening, the IBM Watson Text-to-Speech API shall be called upon to provide an audio description.		
(New) Description	Event Driven: A collection of WAV files shall be created using IBM Text-to-Speech (combined with Python scripts) preemptively. The scripts will be fed, as input, into IBM Text-to-Speech code, which will convert them into WAV files.		
Justification for Extent of Fulfillment	The software written successfully achieves our intended goal of creating a collection of WAV files with the text written to be read out loud to the users. It integrates the IBM Watson Text-to-Speech with customized Python scripting, which quickly makes a collection of well-named WAV files. This leverages advanced Text-to-Speech technology and good coding practices, using the Java style to explain the functionality of the classes.		
Dependencies	FR1.1.1, FR1.1.3, FR.1.1.4		
Expected results (New)	Upon user interaction with quantum elements in the AR Quantum Computer Model, the system should seamlessly play corresponding audio descriptions, generated using IBM Watson Text-to-Speech. The WAV files will follow a standard naming convention, allowing for straightforward and easy access to the files. Additionally, this will allow us to easily swap between voice templates (by changing source directory), should we implement such a feature in future iterations.		
Exception handling	If a text file cannot be read (e.g., file not found, access denied), the script logs an error message indicating the file path and the exception encountered, then continues to the next file. If the IBM Watson Text-to-Speech service returns an error during the conversion process (e.g., authentication failure, service unavailable), the script logs an error message detailing the failed paragraph number and the response from the service.		

ID and Title	FR1.2.2 - Audio Description Subtitles		
Status	Same	Requirement Been Fulfilled?	✓
Priority	Medium	MoSCoW	Could Have
(Original) Description	State Driven: While the audio description of the AR Simulation or its Components is being output, subtitles should be displayed along the bottom of the screen.		
Justification for Extent of Fulfillment	Expected results met.		
Dependencies	FR1.1.1, FR1.1.3, FR1.1.4, FR1.2.1		
Expected results	Audio output from Watson Text-to-Speech should be transcribed at the bottom of the screen in a reasonable font-size and not interfere with the usability of the application. The output rate should match the speed at which the audio is output.		
Exception handling	If no subtitles are displayed along the bottom of the screen, a box should be displayed reading 'Subtitles Not Available.'		

ID and Title	FR1.2.3 - Audio Description AR Integration		
Status (same or changed)	Changed	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Must Have
(Original) Description	Ubiquitous: IBM Watson Text-to-Speech and Unity must be able to communicate such that any required audio description is output alongside the correct model. This will require an SDK, such as the Watson Developer Cloud's (Watson Developer Cloud, 2022).		
(New) Description	Event-driven: When a model is displayed, the system triggers the respective audio description (generated by IBM Watson Text-to-Speech and stored as a Unity asset) when the user interacts with it, so the auditory information seems well timed and follows the subtitles.		
Justification for Extent of Fulfillment	Based on changes to FR1.2.1, this requirement had to be adapted. Therefore, we have managed to integrate audio source components (as Unity assets) directly into Unity itself. These files were generated by IBM Watson Text-to-Speech.		
Dependencies	FR1.1.1, FR1.2.1		
Expected results	Should any model require an audio description, the relevant audio will be provided by IBM Watson Text-to-Speech to Unity and Unity should be able to use this audio alongside the model.		

Exception handling	If audio file cannot be accessed, an error message must be displayed stating 'Audio cannot be found.' The relevant quantum model must be displayed, even if the audio is not available.

ID and Title	FR2.1 - Website Access		
Status	Same	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Must Have
(Original) Description	Event-driven: When a user scans the QR code with a phone's camera, it allows the user to enter the website. The website should also have a unique URL that can be typed if there is no QR Code.		
Justification for Extent of Fulfillment	Expected results met.		
Dependencies	FR1, FR2, FR3		
Expected results	User should be able to access the website when they scan the correct QR Code provided, or if they type in the specific URL into the address bar.		
Exception handling	N/A		

ID and Title	FR2.2 - Hosting		
Status	Changed	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Must Have
(Original) Description	Ubiquitous: The web application shall be hosted on the IBM Cloud using IBM Virtual hosting/ servers.		
(New) Description	Ubiquitous: The web application shall be hosted on GitHub Pages, using secure HTTPS protocol and Vite/Next.js server technologies.		
Justification for Extent of Fulfillment	Updated requirement met - the website is now hosted on Github pages, as we had difficulty gaining full access to the required IBM Cloud features.		
Dependencies	FR2.1, FR1.1.5		
Expected results	The web application will be correctly hosted on Github pages and follow expected server behaviors.		

Exception handling	If there be any issues (e.g., server disconnections) with GitHub Pages, they should be handled gracefully displaying useful error messages.

ID and Title	FR3.1 - Quantum Quiz Initialization		
Status	Same	Requirement Been Fulfilled?	x
Priority (New)	Low	MoSCoW	Could Have
(Original) Description	Event-Driven: When a user clicks the start button, the quiz shall begin. Questions will be generated using the material from 'IBM SkillsBuild Introductory Quantum' course, utilizing IBM Watson Assistant and IBM Cloud		
Justification for Extent of Fulfillment	The 'Quantum Quiz' component is still in the development phase, as the backend and frontend have not been fully developed or integrated. However, this may change before the Product Handover and this document will be updated (in our GitHub 'Documentation' directory) accordingly.		
Dependencies	FR2.1		
Expected results	When a user starts the quiz, the website shall publish 5 multiple choice questions sent by IBM Watson Assistant from IBM Cloud. The quiz interface should be user-friendly, clear, and accessible.		
Exception handling	If the quiz fails to initialize, the website shall display an error message and offer a retry option.		

ID and Title	FR3.2 - Quantum Quiz User Input		
Status	Same	Requirement Been Fulfilled?	x
Priority (New)	Low	MoSCoW	Could Have
(Original) Description	Event-Driven: When a user interacts with the quiz, the answers selected by the user shall be saved and marked for correctness.		
Justification for Extent of Fulfillment	The 'Quantum Quiz' component is still in the development phase, as the backend and frontend have not been fully developed or integrated. However, this may change before the Product Handover and this document will be updated (in our GitHub 'Documentation' directory) accordingly.		
Dependencies	FR2.1		
Expected results	The website shall track and save users' progress of the quiz and allow them to resume.		
Exception handling	If the UI is not responding, the website shall display an error message and allow the user to reload the current page. If the user would like to move onto the next question without answering the		

	current question, they should be prompted to finish the question before moving onto the next one.

ID and Title	FR3.3 - Quantum Quiz Response Output		
Status	Same	Requirement Been Fulfilled?	x
Priority (New)	Low	MoSCoW	Could Have
(Original) Description	Event-Driven: IBM Watson Assistant shall process the answer, check if the answer is correct, then return the output of whether the user got the question correct. Then, the quiz will move onto the next question.		
Justification for Extent of Fulfillment	The 'Quantum Quiz' component is still in the development phase, as the backend and frontend have not been fully developed or integrated. However, this may change before the Product Handover and this document will be updated (in our GitHub 'Documentation' directory) accordingly.		
Dependencies	FR2.1		
Expected results	The website shall show the correctness of the quiz question answered by the user and then allow the user to move onto the next question.		
Exception handling	If IBM Watson Assistant is non-responsive and the correctness of the question is not displayed, the website shall display an error message and an option to reload the quiz page.		

ID and Title	FR3.4 - Quantum Quiz Finish		
Status	Same	Requirement Been Fulfilled?	x
Priority (New)	Low	MoSCoW	Could Have
(Original) Description	Event-Driven: When the user has answered 5 questions generated by the Watson Assistant, the user's final score and feedback shall be displayed. The user shall be able to either click a button 'Retake' or follow the recommended link to SkillsBuild for more information.		
Justification for Extent of Fulfillment	The 'Quantum Quiz' component is still in the development phase, as the backend and frontend have not been fully developed or integrated. However, this may change before the Product Handover and this document will be updated (in our GitHub 'Documentation' directory) accordingly.		
Dependencies	FR2.1		
Expected results	The user's score and feedback shall be displayed. Two options - 'Retake' and 'SkillsBuild' - shall be displayed. When the user clicks the 'Retake' button, the system shall reset the quiz. When the user clicks the 'SkillsBuild' link, it shall direct the user to the appropriate IBM Quantum Computing resources.		

Exception handling	If there is a delay in calculating the score and generating the feedback, then there should be a loading indicator to inform the user. If the button and link are not accessible, then there should be displayed an option to report the problem and an option to refresh the page.

ID and Title	FR3.5 - User Input Question		
Status	Changed	Requirement Been Fulfilled?	✓
Priority	High	MoSCoW	Should Have
(Original) Description	Event-Driven: When the quiz finishes, the user will be able to ask questions to a chatbot that has been trained by the IBM SkillsBuild Quantum Computing course.		
(New) Description	Event-Driven: When the user clicks on the Watson Assistant icon, the user will be able to ask questions to a chatbot that has been trained by the IBM SkillsBuild Quantum Computing course.		
Justification for Extent of Fulfillment	Expectations met. We changed the functionality of the Watson Assistant – we wished to make the Chatbot available at all times, throughout the process, and introduced the icon.		
Dependencies	FR3.4		
Expected results	The chatbot shall answer the questions asked by the user correctly and with 90% accuracy.		
Exception handling	If there is a delay in generating answers, then there should be a sign to inform the user that the chatbot is still generating. If IBM Watson Assistant fail to response, then there should be displayed an option to report the issue and an option to restart generating.		

Non-Functional Requirements

ID and Title	NFR1.1 – Performance Requirement - Load Times		
Status	Same	Requirement Been Fulfilled?	✓
MosCoW	Should Have		
(Original) Description	Upon entering the application, the amount of time the user must spend waiting for the application to load should not be excessive to promote user functionality.		
Justification for Extent of Fulfillment	After testing with JMeter and running 5000 times, none of the tests took longer than 1 minute to load the page. The average response time was 3921ms (and the maximum was 12068ms). Therefore, this indicates that our website satisfies the requirement.		
Metrics/Fit Criteria	The application must be fully rendered and operational within 1 minute of initiating the loading process. This sets a clear performance standard for the application's loading time. In case of a failure the application should distinguish between a server timeout and whether it is related to the system specifications of the user's device. In the latter case, it will suggest to		

	the user to look at the IBM Introductory Quantum Course instead while maintaining the graphical style of the site.
Security	N/A
Constraints	We do not have control over the user's Wi-Fi speed or if their device's specification is appropriate for loading the app.

ID and Title	NFR1.2 – Performance Requirement – Website Reliability		
Status	Same	Requirement Been Fulfilled?	x
MosCoW	Must Have		
(Original) Description	The website has to offer consistent and reliable access to the user. It is crucial to the positive user experience that the website remains accessible and performs effectively.		
Justification for Extent of Fulfillment	After testing with JMeter and running 5000 times, 25.47% of the access requests failed. Therefore, this indicates that our website does not meet this requirement. However, as this is an educational resource that will not receive 5000 access requests within a 12 second period, we have deemed this an acceptable failure rate. Additionally, Git		
Metrics/Fit Criteria	The website fails to load due to a website related issue a maximum of once every 5000 access requests.		
Security	N/A		
Constraints	We are bound by potential challenges arising from the IBM Cloud services. This constraint stems from the fact that any issues on IBM's end directly impact the performance of our website, subjecting it to the duration required for IBM to resolve the underlying problems.		

ID and Title	NFR1.3 - IBM Watson Response Time		
Status	Changed	Requirement Been Fulfilled?	✓
MosCoW	Should Have		
(Original) Description	Within the Quantum Quiz, when IBM Watson Assistant processes answer, the response should be delivered within 45 seconds		
(New) Description	Within the chatbot, when IBM Watson Assistant processes answer, the response should be delivered within 45 seconds		
Justification for Extent of Fulfillment	Watson chatbot answers within 10 seconds during the test.		
Metrics/Fit Criteria	IBM Watson Assistant must process the user's questions and return results within 45 seconds.		

Security	The processing of IBM Watson Assistant should adhere to security best practices to ensure the confidentiality and integrity of user interactions and model descriptions.
Constraints	The system should operate within the defined time constraints. Any delays should be communicated to users with clear messages encouraging patience.

ID and Title	NFR1.4 - IBM Watson Assistant Answers		
Status	Same	Requirement Been Fulfilled?	✓
MosCoW	Must Have		
(Original) Description	IBM Watson Assistant must consistently deliver precise and pertinent answers to user queries concerning Quantum Computing.		
Justification for Extent of Fulfillment	All metrics met.		
Metrics/Fit Criteria	Achieve a minimum accuracy rate of 90% in responding to user queries related to Quantum Computing.		
Security	Aligns with FR1.2.1 security requirements.		
Constraints	Users must receive informative, helpful, and accurate responses to their inquiries, ensuring a positive user experience and effective engagement with Quantum Computing topics.		

ID and Title	NFR2.1 - Security Requirement – Code Back Ups		
Status	Same	Requirement Been Fulfilled?	✓
MosCoW	Must Have		
(Original) Description	Code will be pushed initially to GitHub. All code from the GitHub Repo will be backed up to a separate cloud location and cloned to all local developer devices at least once per month during all phases of the Software Development Lifecycle. This means that if one location of the code is shut down, we will always have back-ups.		
Justification for Extent of Fulfillment	All metrics met.		

Metrics/Fit Criteria	We will successfully complete this if there are monthly backups that have been committed to a cloud location and that GitHub shows frequent cloning onto local developer device.
Security	This is an activity that only the developers can do.
Constraints	We will constrain ourselves to the use of GitHub as the primary code repository. We will also exclusively use Google Drive. Finally, we trust the developers to make regular local downloads.

ID and Title	NFR2.2 – Security Requirement – Malicious Attacks		
Status	Changed	Requirement Been Fulfilled?	✓
MosCoW	Could Have		
(Original) Description	If a malicious attack, website functionality should be restored within 24 hours (using code back-ups). Any reported vulnerabilities should be found and repaired within 2 days. If a malicious attack, communication plan between IBM Cloud will be in place.		
Justification for Extent of Fulfillment	All metrics met, as restoration hasn't been required.		
Metrics/Fit Criteria	Website restoration within 24 hours of an attack.		
Security	We will implement an update and patch management protocol to ensure that when the website is down, we have strict rules to follow to get it back and running and ways to patch the security breach effectively. We will also ensure that only developers can perform these updates and patches.		
Constraints (New)	Our primary constraint is that if a malicious attack is launched on GitHub pages and through that attack our security is breached; then the security of our program and any patching is for GitHub pages to accomplish which we cannot guarantee would be done in a timely manner.		

ID and Title	NFR2.3 – User Responses to IBM Watson		
Status	Changed	Requirement Been Fulfilled?	x
MosCoW	Could Have		
(Original) Description	The user's response to the Quantum Quiz will only be kept until IBM Watson Assistant finishes processing it and the result is recorded. If IBM Watson Assistant does not analyse it successfully, the voice audio recording will be deleted within 5 minutes. The same is true for IBM Assistant when it answers user's questions about Quantum Computing.		
Justification for Extent of Fulfillment	IBM Watson Assistant takes text input rather than vocal input from the user, this was deemed necessary to increase accessibility for groups with speech impairments or are non-verbal etc. as they would not be able to use this resource. If vocalization is necessary, most mobile		

	devices have an option to take audio input through the keyboard and this method is encouraged for users for whom this would be necessary.
Metrics/Fit Criteria	If IBM Watson Assistant does not answer the users' questions successfully, the record will be deleted within 5 minutes.
Security	To ensure data privacy and security, user responses are securely stored until processed.
Constraints	The retention of user responses is contingent on the successful processing by IBM Watson Assistant. If processing fails, recordings are subject to immediate deletion within the specified 5-minute period.

ID and Title	NFR3.1 - Scalability Requirement - Application Traffic Scaling		
Status	Changed	Requirement Been Fulfilled?	✓
MosCoW	Could Have		
(Original) Description	IBM Cloud will be used to host the application and will automatically scale the website to be able to cope with varying user traffic and resource requests.		
(New) Description	GitHub Pages will be used to host the application and will automatically scale the website to be able to cope with varying user traffic and resource requests.		
Justification for Extent of Fulfillment	This requirement has been updated to reflect changes made to FR2.2 (the change to GitHub Pages from IBM Cloud). We have met this requirement's fit criteria. However, this is conditional on the fact that we do not exceed the bandwidth limit of 100GB. Additionally, we will contact GitHub via email if any failures occur, due to their failure to scale down.		
Metrics/Fit Criteria	GitHub Pages will scale the website resources used to cope with user traffic of between 1 and 1000 users at any one time without using any more resources than necessary. If the website fails to scale up correctly, the website should display useful error messages and keep making requests until the website loads. If the website fails to scale down, GitHub must be notified of failure.		
Security	N/A		
Constraints	Application must not attempt to handle traffic where there are 0 users, as this may lead to application shutting down and an application cold start when traffic grows again (Schweigert and Hadas, 2022).		

ID and Title	NFR3.2 – Availability & Scalability Requirement – Website/ AR Access		
Status	Same	Requirement Been Fulfilled?	✓
MosCoW	Must Have		
(Original) Description	The objective is to ensure that the website application is accessible and functional across both mobile and tablet devices. This includes ensuring optimal user interface and navigation experiences tailored for a variety of screen sizes, resolutions, and operating systems.		

Justification for Extent of Fulfillment	The website and AR components work on iOS and Android with dynamic adjustment to screen size for an optimal user interface. Drop down menus replace full website headers for phone screens.
Metrics/Fit Criteria	The measure of success will be the 'as expected' functionality of the website and AR features with major mobile operating systems, namely iOS and Android, as defined in the functional requirements.
Security	N/A
Constraints	The primary constraints are that development and testing will be confined to Android and iOS. In addition to this, we will have to adhere to the guidelines and best practices set forth by the respective mobile device manufacturers. This includes compliance with any specific standards that these manufacturers advocate that would create an optimal app performance and user experience on their devices.

ID and Title	NFR4.1- Technology Requirement – AR Coding		
Status	Same	Requirement Been Fulfilled?	✓
MosCoW	Should Have		
(Original) Description	The main website component (excluding Unity) should be built using HTML, CSS, and JavaScript with the option of using REST API (Application Programming Interface) if needed.		
(New) Description	The main website component (excluding Unity) should be built using HTML, SCSS, CSS and JavaScript with the option of using REST API (Application Programming Interface), Vite, Node.js, and Next.js, if needed.		
Justification for Extent of Fulfillment	All metrics met		
Metrics/Fit Criteria	This requirement will be assessed after the completion of the website on whether it has been successfully using HTML, CSS, and JavaScript. When testing is conducted, ESLint (Wikipedia, 2023b) developer tool must find no errors in the JavaScript code and our chosen HTML5 validator (W3C, no date) must find no issues in the HTML.		
Security	Any C# written for the project shall meet expected Common Code Standards as set out by Microsoft (Microsoft, 2023).		
Constraints	All AR component code files committed to GitHub code must have extension '. unity' or '.cs.' If any code used in AR components is not written in Unity or C#, it must be converted by a team member to C# or Unity files with no functionality lost.		

ID and Title	NFR4.2- Technology Requirement - Building Website		
Status	Changed	Requirement Been Fulfilled?	✓
MosCoW	Must Have		

(Original) Description	The main website component (excluding Unity) should be built using HTML, CSS, and JavaScript with the option of using REST API (Application Programming Interface) if needed.
(New) Description	The main website component (excluding Unity) should be built using HTML, SCSS, CSS and JavaScript with the option of using REST API (Application Programming Interface), Vite, Node.js, and Next.js, if needed.
Justification for Extent of Fulfillment	The requirement was updated as IBM standards were taken into consideration (especially with regards to SCSS, Vite for Unity integration, and Next.js). When linting the website JavaScript, no errors were found (the ESLint plugin 'next/core-web-vitals' was used, as it is recommended by Next.js). Ultimately, the majority of the HTML was generated through builds and the HTML written manually did conform to standards (to the extent that it could in a React/Next.js app).
Metrics/Fit Criteria	This requirement will be assessed after the completion of the website on whether it has been successfully using HTML, CSS, and JavaScript. When testing is conducted, ESLint (Wikipedia, 2023b) developer tool must find no errors in the JavaScript code and our chosen HTML5 validator (W3C, no date) must find no issues in the HTML.
Security	The website must not have security exploitable flaws with regards to code.
Constraints	All JavaScript code within the website must meet Core Web Vitals (web.dev.. n.d.) and all HTML within the website must meet HTML Living Standard (WHATWG, 2023).

ID and Title	NFR4.3 - Technology Requirement Layout and Uniformity		
Status	Same	Requirement Been Fulfilled?	✓
MosCoW	Must Have		
(Original) Description	The website must feature a uniform design, including footers (where relevant), a drop-down menu, and consistent CSS styles. The layout and design should remain constant across all pages.		
Justification for Extent of Fulfillment	We have built this website utilizing Carbon React components and Sass. For accessibility and best practices (for Google Lighthouse), our website was classified 'Green' on the traffic-light system. Additionally, we have employed IBM's Carbon Design practices to ensure consistency across the entire website and several components.		
Metrics/Fit Criteria	The desired consistency in design can be achieved by implementing either Bootstrap or custom CSS. This approach aims to ensure a seamless and visually cohesive user experience. The user experience will be assessed through Google Lighthouse (Chrome for Developers, 2016), serving as a metric for evaluating the effectiveness of the chosen design implementation.		
Security	No specific security considerations are outlined for this requirement.		
Constraints	This consistency can be achieved by utilising either Bootstrap or custom CSS, ensuring a seamless and visually cohesive user experience. User experience will be evaluated through Google Lighthouse (Chrome for Developers, 2016).		

ID and Title	NFR4.4 – Technology – Documents of Application

Status	Changed	Requirement Been Fulfilled?	✓
MosCoW	Should Have		
(Original) Description	Postman software can be used to build documentation of all code. This documentation should cover the full range of functionalities offered by the application, supplying clear, detailed, and easily accessible information for developers and users.		
(New) Description	Postman, Doxygen, JSDoc, and Unity recommendations can be used to build documentation of all code. This documentation should cover the full range of functionalities offered by the application, supplying clear, detailed, and easily accessible information for developers and users.		
Justification for Extent of Fulfillment	The documentation has been limited to the private Github repository where all our code is stored. We have used professional tools/standards where possible. Our website, the Python and C#-related code for our audio component, and Unity components have all been appropriately documented (90% is considered approximate and may not reflect actual documentation levels). Several automatically generated pieces of code (e.g., Unity builds, .next directory) have not been documented, as they are already documented elsewhere and can be automatically loaded/generated on any system.		
Metrics/Fit Criteria	The completeness of the documentation will be assessed by the metric of 90% of the applications functions being clearly described including formats and expected behaviours.		
Security	Access to the documentation should be restricted to the developers only, and potentially the client if necessary.		
Constraints	The documentation must be created using Postman, JSDoc, Doxygen, and all other relevant documentation tools. It should adhere to industry standards for technical documentation.		

User Manual 1 - System Use

End-User Use (Non-Technical)

Getting Started

This product was made to be simple and easy for anyone to use, without requiring any downloads. To do so, simply scan the provided QR code using your tablet or smartphone or type the website URL manually at the top of the address bar (in the browser) and reach the same location.

Fig.1 - Website QR Code (takes you to https://atlantaemrys2002.github.io/ARQuantum/)

As soon as you have reached our homepage, with just one click, the quantum world opens itself to you for your first lesson in this revolutionary and immersive field.

Navigating the Website

On the homepage, you are presented with multiple options – from the dropdown menu in the top left, you can choose to start with the AR (remaining on the home page) or consult our 'References' page to view the resources that were used to create the content of the AR presentation. If you chose to look at the AR presentation on Quantum Computing first, you can select either the 'Open on Quest' or 'Open in Quicklook' button. If you are using a mobile phone/device, select 'Open in Quicklook.' This will open the AR component in full screen mode. Both portrait (Fig. 2) and landscape (Fig. 3) orientations of the website's home page are displayed below.

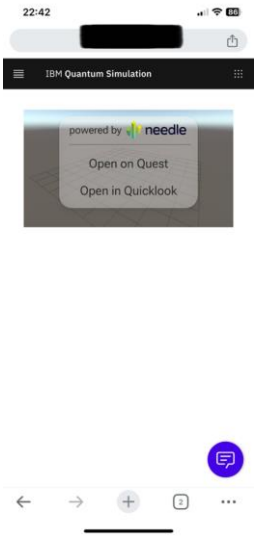


Fig. 2 – Website Homepage (Portrait Mode)

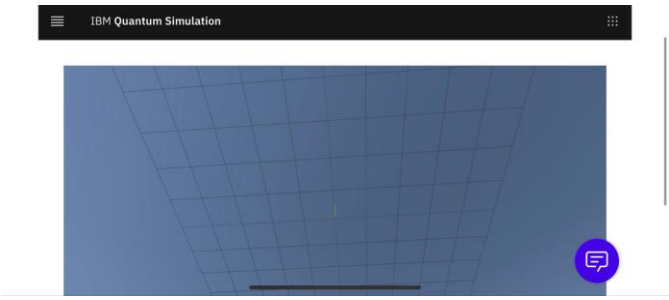


Fig. 3 – Website Homepage (Landscape mode)

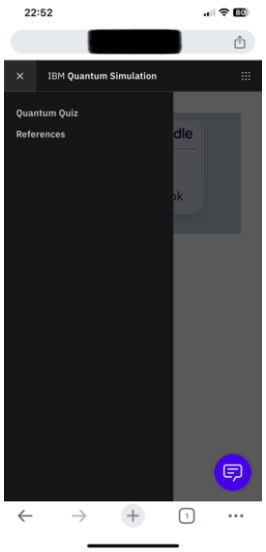


Fig. 4 – Dropdown menu (on all website pages). Allows user to navigate between pages on the website.

Starting your first AR Chapter

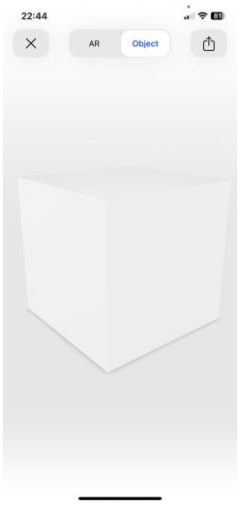
To start your first AR chapter, you will need to find a marker – the QR code. Move your mobile device around, pointing the camera at the physical copy of the QR code, as instructed on the screen.

Fig. 5 – Prompt displayed on screen while marker is not found (i.e. QR code)

Once the marker has been detected, a virtual object should appear (fixed to the marker). An audio description of the object will commence. There are accompanying subtitles for accessibility and for those who may prefer reading along or need any form of language support (see Fig. 8). You will now be able to move around with the object remaining in place (in the example below, the object is a cube). If you wish to make the object bigger or smaller, pinch your fingers together on the screen (pinch in) to make it smaller. Pinch out (the opposite) to zoom in on the object. If you click anywhere on the screen, aside from where the buttons and object are described, the 'AR/Object,' 'x' and 'Export' (top right) buttons will disappear. They will return if you click the scene again.

Fig. 6 – Object (cube) displayed on a marker.

If you wish to take a picture of the object, click the circular button at the bottom of the screen. To view the object in virtual reality (not overlaid on the real world), click 'Object' at the top of the screen, instead of AR. To leave the presentation at any point, click the 'X' in the top left corner to leave the AR experience and return to the home page.



Navigating the AR Chapter

Once a chapter loads, you will be able to navigate it with 2 buttons labelled 'Next Scene' and 'Previous Scene' as seen below. To move between chapters, use the buttons. Using these, you can go through the presentation, enabling you to navigate forwards, as well as backwards, in case you miss anything! If you are currently viewing the first chapter and click 'Previous scene', nothing will happen. Similarly, if you are currently viewing the last chapter, nothing will happen if you click 'Next scene'.

Fig. 8 – Example chapter in AR presentation – with 'Next Scene' & 'Previous Scene' buttons, as well as subtitles to accompany audio.

Finishing the AR Chapter

As described above, once you have concluded the presentation and want to leave the experience and return to the home page, click on the 'X' in the top left.

Watson Assistant

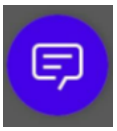


Fig. 9 – Watson Assistant icon (click in website)

If you have any questions regarding what you have learned or are simply curious about what the little blue dot at the bottom of your screen is (on any of the website pages) - give it (Fig. 9) a click!

Fig. 10 – Location of Watson Assistant Icon on Website pages.

This will now open (maximize) the Watson Assistant Chatbot where you can ask your questions.

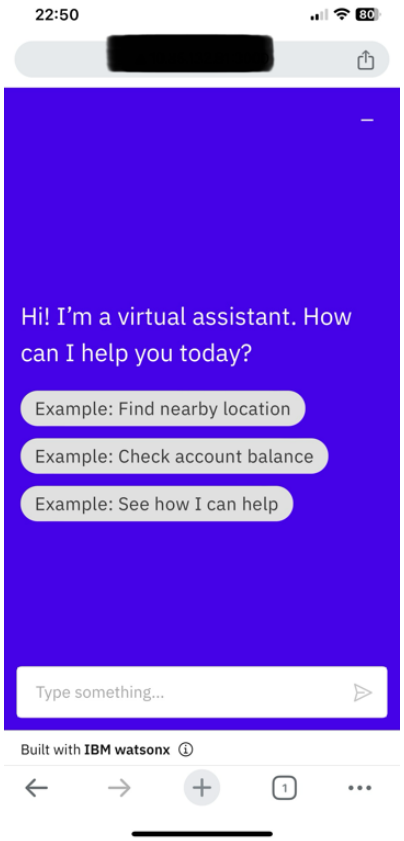


Fig. 11 – Maximized Watson Assistant, displaying input box for any questions, as well as common questions above.

To ask a question, you can either click on a commonly asked question or type it into the box labelled 'Type something' and press 'Enter.'

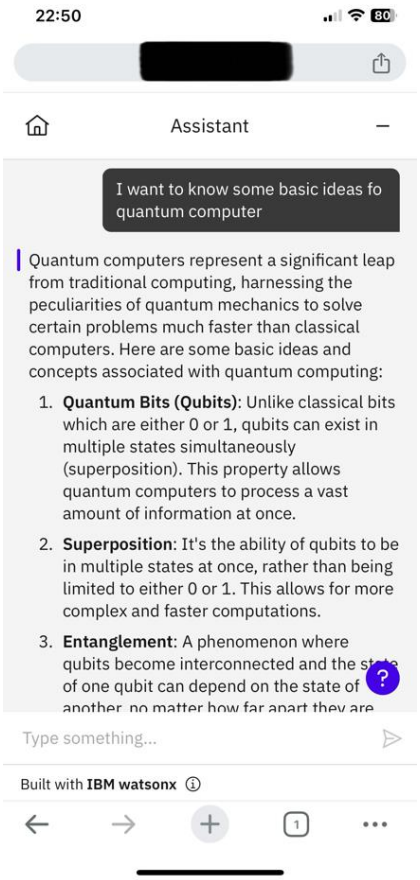


Fig. 12 – Watson Assistant chatbot replying to a user's question.

The example query above is 'I want to know some basic ideas fo quantum computer' – upon being asked, Watson returns a relevant answer. If at any point you want to leave the Chatbot and return to the home page, click on the '-' (minimize) button in the top right-hand corner.

Further Resources

After sufficient exploration of the website, you may want to go to the link (displayed below the main AR component and in the final chapter of the AR presentation), where you can further read and study the quantum concepts you have learnt by going to the IBM SkillsBuild course. You will also be able to earn several free certificates if you complete the Skillsbuild Quantum courses.

Developer Use (Technical)

Starting The Website Server

It is recommended that, if you run the application locally, you use a Linux/Unix/macOS-based system.

Once you have downloaded our code, open a terminal, and navigate to the directory 'website_framework.' We would recommend using yarn. First, enter the command 'yarn' to install all dependencies. Then, enter 'yarn @carbon/react@1.33.0,' followed by 'yarn sass@1.63.6,' and, finally, 'yarn @carbon/icons-react' (based on recommendations of Carbon Design Tutorial (IBM Carbon, n.d.).

You should now run 'yarn build' to build the application, then run it with 'yarn dev.' Navigate to 'localhost:3000' on your desktop and it should compile and load. You can now find your IP address with the 'ifconfig' command and navigate to your IP address with port number 3000 (localhost) on your phone (check your desktop and mobile device are on the same network). E.g., if your IP address were 0.0.0.0, you should navigate to 0.0.0.0:3000 on your mobile device. The app should be displayed and have full functionality.

Compilation Errors

If any issues should arise with regards to the compilation/running of the application, first, check you have navigated to 'http://' (not 'https://,' as GitHub Pages handles HTTPS). If this does not work, first delete the 'node_modules' directory, the '.next' directory, and the file 'yarn.lock.' Complete all the instructions described in the above section up till 'yarn@carbon/icons-react.' Then, create a Next.js app (using 'yarn create next-app'). Name your application anything (e.g., 'abc') and continue through all the questions with the following answers:

- TypeScript – NO
- ESLint – YES
- Tailwind CSS – NO
- /src Directory – YES
- App Router – YES
- Default Import Alias – NO

Delete the directory with the Next app's project name (for example, if you named your application 'abc,' delete 'abc'). Then, try to run the application again (i.e., 'yarn build' followed by 'yarn dev').

Building a New Chapter

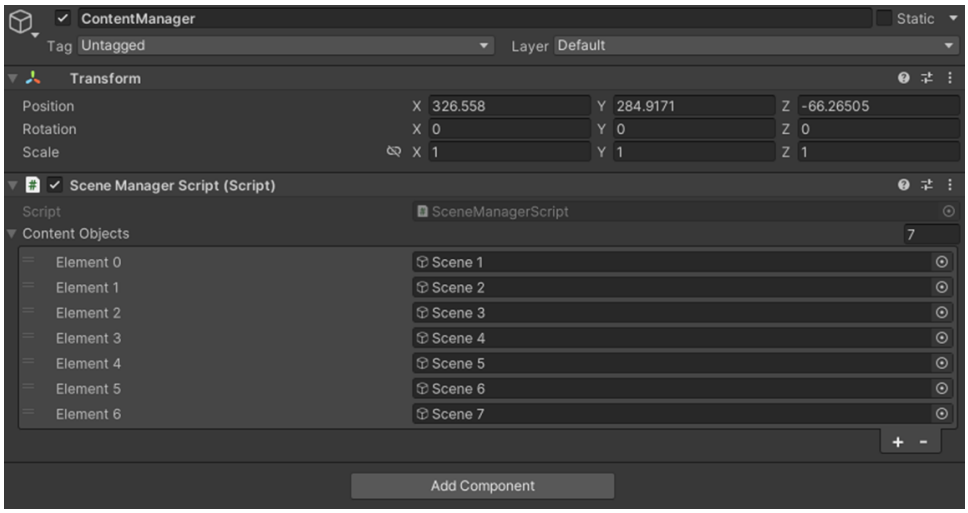
The way we have structured scenes and scene changes utilizes Unity's subscenes. This means we are not actively rendering new scenes in and out. Instead, we have child objects we associate with 'Scene 1' and, when 'Scene 1' is 'active', we load in all the objects within 'Scene 1'. When scene changes occur, for instance, we switch to 'Scene 2' and we make all the objects in any scene other than 'Scene 2' inactive while setting 'Scene 2' to 'active', thereby rendering them out and rendering 'Scene 2' in.

The essential components you need to build a new chapter are:

- A Content Manager object
- Scene Manager Script.cs (this is in project/assets/scripts/*)
- A blank game object which we will call 'Scene 1', which will contain everything that happens within that scene.

The hierarchy view should look similar to this:

To make 'Scene 1' recognizable as a subscene we will have to add it to the array of objects in the content manager (in this case you can see we have 7 scenes):



The script here is the SceneManagerScript.cs from earlier.

In essence, this is enough for it to be considered a subscene. However, it will have no functionality for switching between these.

Transitioning between scenes.

This is where we will need buttons and that script.

Firstly, we require an event system that recognizes basic user input for the buttons. This can be achieved by importing the default supplied Unity 'XRUIInputModule'.

Now, we will elaborate on the script that will connect to the buttons and manage the scene changes.

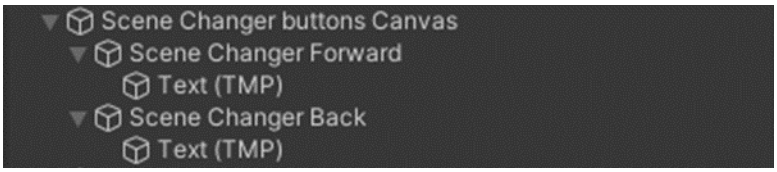
```
using UnityEngine;

0 references
public class SceneManagerScript : MonoBehaviour
{
    3 references
    public GameObject[] contentObjects;

    5 references
    private int currentIndex = 0;

    0 references
    private void Start()
    {
        // Ensure only the first content is active at start
        UpdateSceneVisibility();
    }

    0 references
    public void ShowNextContent()
    {
        if (currentIndex + 1 < contentObjects.Length)
        {
            currentIndex++;
            UpdateSceneVisibility();
        }
    }

    0 references
    public void ShowPreviousContent()
    {
        if (currentIndex > 0)
        {
            currentIndex--;
            UpdateSceneVisibility();
        }
    }

    3 references
    private void UpdateSceneVisibility()
    {
        for (int i = 0; i < contentObjects.Length; i++)
        {
            contentObjects[i].SetActive(i == currentIndex);
        }
    }
}
```

This script operates through a counter, which we will increment +1 and -1 depending on which button is pressed ('Next Scene' and 'Previous Scene') respectively.

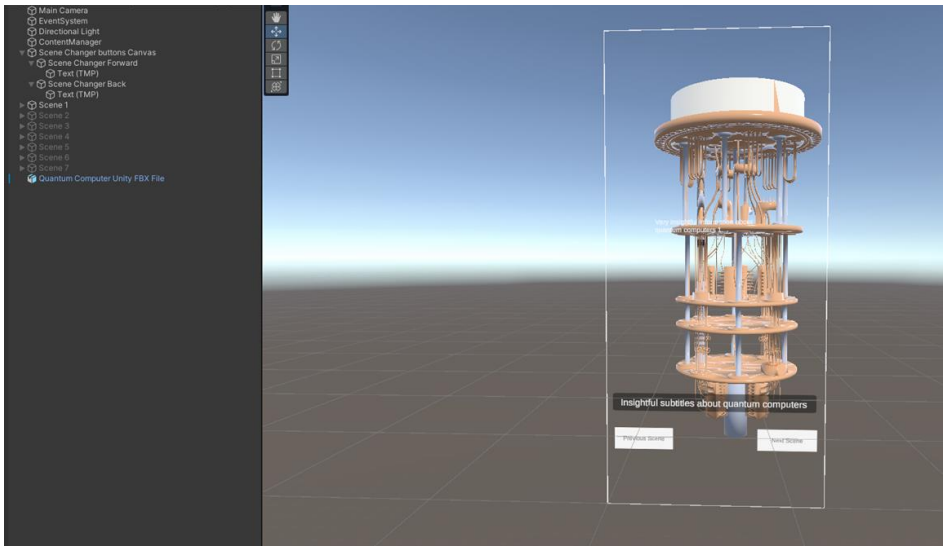
In other words, 'Next Scene' calls upon the function 'ShowNextContent' and 'Previous Scene' calls upon the function 'ShowPreviousContent'.

After any function is called, we run the function 'UpdateSceneVisibility', which will do just that. It will go through all the objects in the array of 'ContentManager' and set to 'active' only the one that the counter is on.

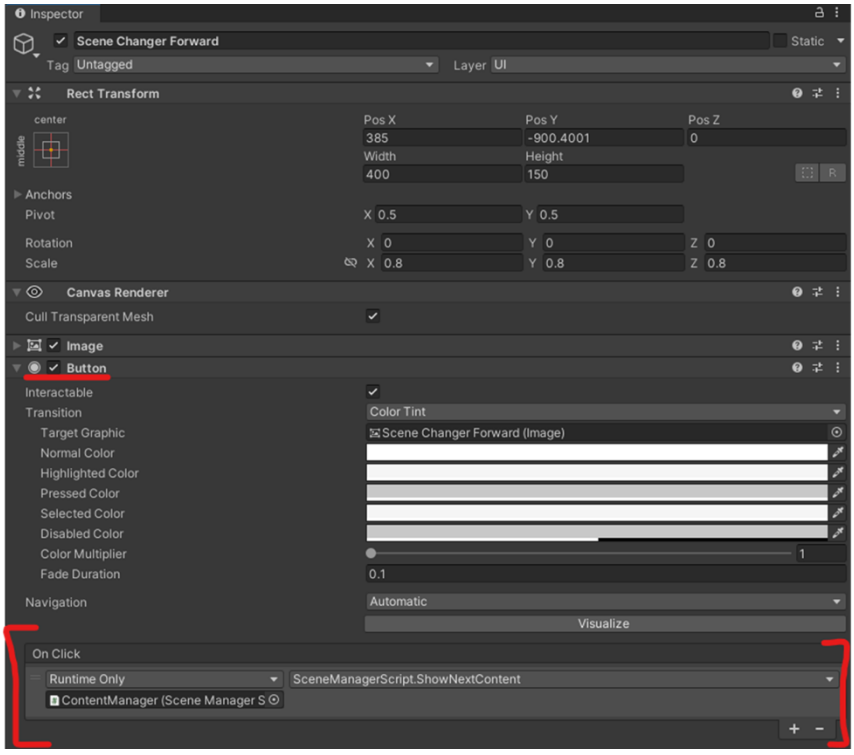
Following on from this, we will require buttons to attach the functions to:



By setting the objects above to the correct types you can create a UI like this:



Then, all that is left is connecting the buttons (panels) to the relevant function and how it should be interacted with in the inspector to the right. You select 'Button' and set it to be 'On Click' and then you import the SceneManagerScript and call the relevant function for the specific button you are working on.



From here, simply import your texts, animations, and audio into the subscene and that's a chapter done.

Generating Audio Files

Audio files used in AR simulations are dynamically generated using the IBM Watson Text to Speech service. The actual generation involves the scripting in Python, the reading of text input from files, and the API discussed above is used to convert this text into spoken audio with nearly 100% accuracy in a WAV file format. The audio is thus finalized and placed into the appropriate folders, such that different types of speech and sounds fall into different folders.

- Text Preparation

    - This is preparation of the input – it is stored in the form of plain text in a text file. The main content of the text file can be explained as a series of paragraphs, separated by two new lines.
- Script Execution
    - The text files' execution requires a script while iterating in Python. The script is developed to work with an IBM Watson API Key and Watson Service URL, which must be provided for authentication.
- Audio Files Creation
    - The audio content received is then written into a WAV file. In case a text file has many paragraphs, each of the paragraphs is saved as separate sound files with naming conventions which obviously describe each paragraph and allow it to be recognized and organized.
- Error Handling
    - The script incorporates error handling in case the text file cannot be read or if some error appears from the text-to-speech service itself.

Unity/Web-XR Build

The Unity/WebXR build is based on two tutorials and many of the instructions you must follow are the same/paraphrased from these tutorials (feel free to consult them alongside these instructions):

- Needle Tutorial: https://www.youtube.com/watch?app=desktop&v=QYd2XPL0WW0
- GitHub/Vite Integration Tutorial: https://www.youtube.com/watch?v=yo2bMGnIKE8

If you wish to rebuild our Unity code and host it yourself, first install Unity, Needle Engine, Node, and VSCode. Then, download the 'Needle Engine for Unity' package from this link: https://engine.needle.tools/docs/getting-started/#prerequisites. Open a new, empty Unity 3D Core package and import Needle Engine as a custom package.

Download our Unity code (not the build, but the original files). Open the files in a new window, then export all scenes as a custom package. Import this custom package into the same project as Needle Engine was. Within 'Export,' agree to EULA educational licenses (make sure your situation apply), etc. Then, click 'Generate' within the 'Export' asset, running the Vite development server. The Needle Engine tutorial (listed above) also describes how to set up Image Tracking – our code will already fulfill this functionality. However, if this functionality fails, the tutorial is a useful resource to consult.

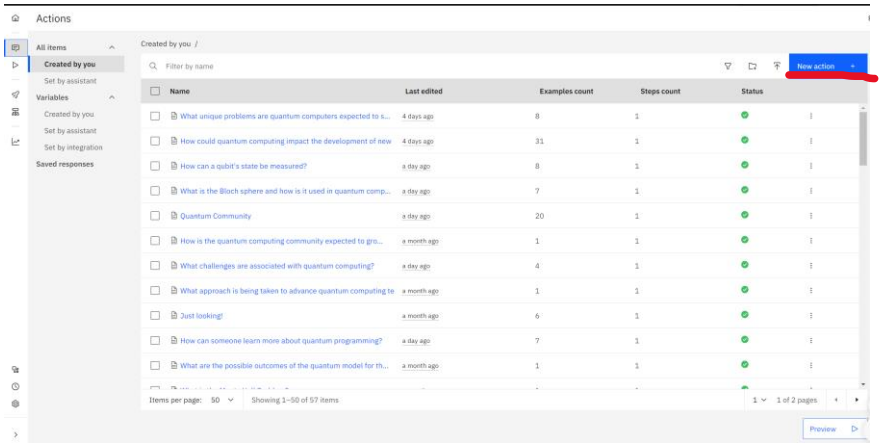
To deploy on GitHub pages, create a GitHub repository and commit the *generated* Unity build (which has the same name as your Unity) to the repository. Edit the 'vite.config.js' file in the generated Unity build. Within the 'export default defineConfig' function, add 'base: '/nameofyourbuild' (i.e., the name of your build). Make sure '/dist' has been committed, as it is vital for your project. Then, in your terminal (in the directory where your Unity build is located), run 'git subtree push –-prefix dist origin gh-pages'. Check your code has been published by navigating (on GitHub) from the code view (home page) of your repository:

1. Go to 'Settings.'
2. Under 'Code and Automation,' click on 'Pages.'
3. At the top, it should say 'Your site is live at [link].'

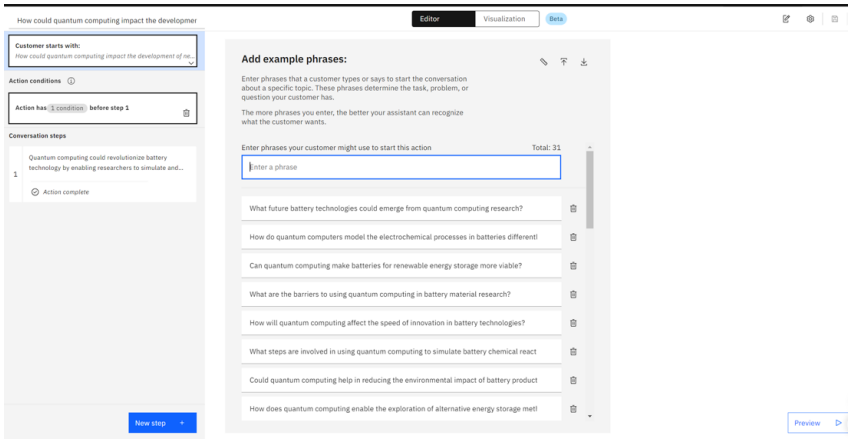
Training Watson Assistant

To make an ideal chatbot with the appropriate and correct answers, here is the way (under the lite account provided by IBM Watsonx Assistant).

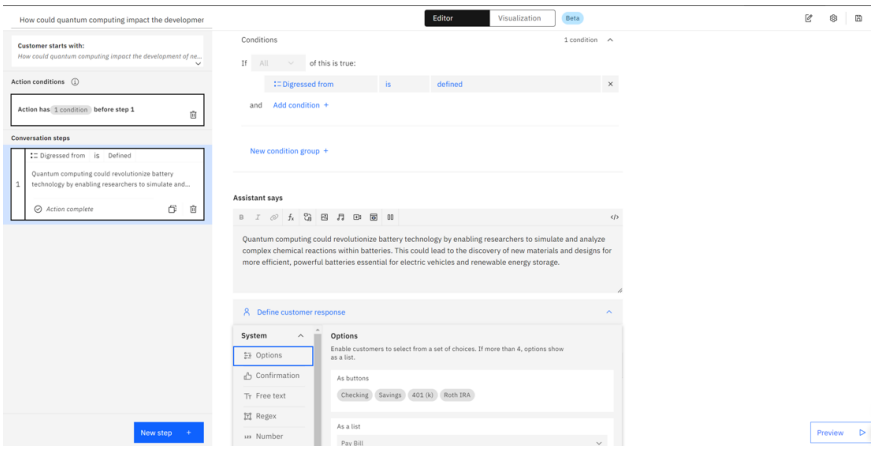
Firstly, use the "New action" function to create a scenario that contains a topic and answers about that topic.



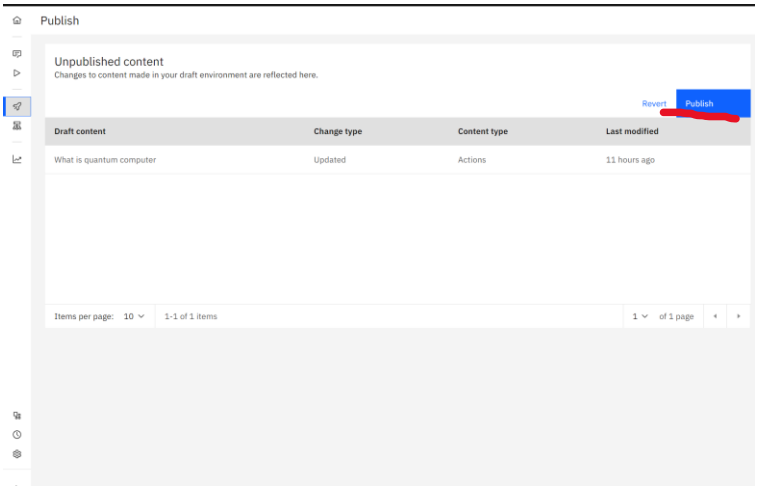
Then according to the topic, set how the user may ask questions and create the phrases (as many as possible because the more phrases you create, the more accurate the chatbot will be).



The next step is to set the conditions in which the answer will be shown and include more options for interaction with the user (according to your aim).



Finally, publish this action to the entire chatbot as a new version, using this publish function.


Publish
Unpublished content
Changes to content made in your draft environment are reflected here.
Revert
Publish
Draft content
Change type
Content type
Last modified
What is quantum computer
Updated
Actions
11 hours ago
Items per page: 10
1-1 of 1 items
1 of 1 page

User Manual 2 - System Development

Source Materials

Our client, IBM, provided us with access to the IBM Skillsbuild courses and, throughout the course, we would send weekly updates of our progress on these resources to our client contact. These courses also helped us understand IBM technologies better, including IBM Watson.

Additionally, for work on Unity the use of standard Unity badges was used as tutorials, notably the AR Marker tutorial. The Unity modules WEBXR and WEBGL were incorporated during development of the AR presentation.

Whilst building the website framework, Carbon Design modules (specifically, built for React) were imported, using yarn, as were 'Sass' and other modules. Documentation for Carbon Design, JavaScript, React, Next.js, Unity, Needle Engine (to export to WebXR to ensure website compatibility), Yarn, and IBM Watson Assistant was consulted, as well as several valuable tutorials, such as IBM Carbon Design's React Tutorial (IBM Carbon. n.d.).

In addition to the key resources discussed above, we have also consulted numerous other sources for our code and product. Please see our documentation/references in GitHub for the full list (references may be subject to change before product handover).

Resources References Link:

https://github.com/COMP2281/software-engineering-group-26/blob/main/Code/Website/website_framework/References.txt

Functionality Development Process

Functionality	Development	Challenges
Marker Based AR Quantum Computer	Having built the Unity Assets through Autodesk Inventor in Unity, the QR code being used as a marker is selected and put into the 'Reference Image Library'. Simply using the AR tracked image manager the Quantum Computer model is selected to be the 'Tracked Image Prefab' and adjust accordingly through rotation and scale to reflect real life.	Challenges faced were finding the correct settings and finding up-to date tutorials for the most recent version of Unity and relevant APIs.
AR Chapter	Development of the AR chapter involved several key steps: Scene Management: we utilized subscenes to manage content efficiently. This method avoids the overhead of loading and unloading entire scenes, instead toggling the active state of child objects associated with specific subscenes. Content and Interaction Setup: A Content Manager object was introduced to keep track of all scenes, including the AR chapter. Scene Manager Script.cs, located in assets/scripts/, was used to manage scene transitions. This script facilitated the smooth switch between AR content and other educational materials within the application. User Interaction: To interact with the AR chapter, an event system based on Unity's default XRUIInputModule was imported, enabling basic user inputs for navigating	The largest challenge in development was knowing and understanding how each component interplays with each other. This could be argued is the fault of the developer having little relevant starting knowledge on Unity and C#. This means that the whole task took longer to accomplish than was reasonable. However, the finished product successfully met expectations.

	through the AR content. Buttons were integrated to allow users to move to the next or previous scenes. This was accomplished by using a counter within the SceneManagerScript to manage scene visibility dynamically.	
Website Framework (Basic)	The Carbon Design React Tutorial was a major inspiration for the development of the website – the layout of the page header was created after following the tutorial. Additionally, the recommended modules for carbon design components from the tutorial were imported. The Watson chatbot was soon integrated by calling on the JavaScript code to generate it	Our developer had little knowledge of React at the beginning and found it difficult to learn in such a short time frame.
Integrating Unity and React Website	IFrame HTML tags were heavily researched and, eventually, an iFrame allowed a Unity VR WebGL build to be displayed on the website.	There were some struggles with the iFrame tag, as attempts were made to test it, while it was calling a 'WebGL AR' from Unity, which, obviously, consistently failed. Additionally, some of its attributes were difficult to correctly define.
Unity WebGL and AR	As Unity does not currently allow for AR WebGL builds, Needle Engine was used to build (from Unity) a WebXR application (with a Vite development server). This technique was learnt from a tutorial [YouTube, n.d.a.]. This was placed in a separate repository to our main website, and, through subtrees on Github/git, we could host the AR application on GitHub pages, This Github page was then called into the website via iFrame. This technique, too, was influenced by a tutorial (YouTube, n.d.b.).	The incompatibility of WebGL AR and Unity was the main struggle when developing the website. The majority of third-party softwares that would allow us to integrate the two concepts were outside our financial budget. However, Needle Engine (a relatively new tool) allowed us to accomplish our task, although the research period was longer than desired.
Hosting	The website is hosted on GitHub pages. Configuration of the Next.js application was done through GitHub actions. Several edits and formatting (e.g., changes from '/' to './') had to be made in order for GitHub pages to function correctly.	Incorporating the Next.js application, using GitHub actions, proved challenging, as there were few resources available to consult. Several precise changes had to be made to certain configuration files and the process was incredibly time-consuming. Additionally, we had to wait for a GitHub Educational license to be itHub functionalities
Text-to-Speech	We implemented IBM Watson Text-to-Speech API for the creation of audio files from already written text. A Python script was made to read text inputs, converting them to speech, and organizing the outputs into categorized folders. Plans for future development include real-time speech rendering within Unity to enhance user engagement.	Difficulties with Unity and managing the API limits. We overcame the difficulties in selecting the different voices and saving the files in the very specific format required by creating Python code that did it all automatically
Watson Assistant	We built the web live chatbot with the online software IBM Watson Assistant provided by IBM Cloud. The chatbot was built and trained step by step following the tutorial provided by IBM Watson Assistant.	The content for setting and training the chatbot was not provided. We had to collect, filter, and edit the valid information ourselves. Therefore, this became a very

		challenging and time-consuming task.

Non-Functional System Aspects

Code Back-ups

Our code backups are a crucial aspect of the system's development, ensuring that all progress is securely stored and can be retrieved or restored in the case of unexpected data loss, corruption, or when reverting to previous versions, as necessary. Our team has outlined a strategy for maintaining robust code backups, utilizing GitHub as our primary version control system.

Backup Strategy

- Manual backups:
  - Our backup strategy utilizes manual backups, where developers are encouraged to periodically (and manually) download a ZIP of the current repository stat or use Git to clone the repository in a secure location on their local device.
  - It is encouraged for this process to be carried out at least once a month, as this ensures that a recent copy of the entire project is always available offline.
- Repository Security:
  - The security of the repository is tightly controlled with permissions only granted to team members and those who require them.
  - No sensitive data is being worked with and, therefore, encryption is not necessary as it is already publicly available.

Best Practices on GitHub Usage

Along with this, the team follows best practices for working with GitHub repositories. Some of these repositories are kept locally and other remote repositories are synchronized to avoid conflicts, using informative and understandable descriptions for each commit, frequent commits, and working in separate branches before merging into the main branch after comprehensive reviews at our weekly team sessions.

Special Considerations for Unity Code and Website Hosting

The Unity project code is managed in a distinct branch separate from the main branch of the repository to avoid conflicts with the .gitignore settings, which are tailored to the project's web components.

Once the AR Unity project segment is finalized, it will be exported and subsequently merged into the 'code' folder within the main branch to be called upon by the website, ensuring seamless integration with the rest of the project.

An exact duplicate of the website code is held in a private repository by one of our developers, so that it can be deployed to GitHub pages.

Malicious Attacks

Our plan to counter the effects of malicious attacks on our product is outlined below.

1.Communication plan

1.1 Internal communication

Establish an emergency team to respond to the crisis.

Notify internal stakeholders, including IT staff and managers about the attack and the steps being taken.

1.2 External communication

Contact GitHub Pages support through the designated emergency contact channels

Inform users of the incident (via website) and the expected resolution timeline, maintaining transparency.

2.Restoration and recovery

2.1 Utilize code back-ups

Follow a pre-defined protocol to restore website functionality from the most recent secure backup within 24 hours of the attack.

2.2 Diagnosing the problem

Collaborate with GitHub Pages and web technology experts to conduct a thorough investigation into the attack, identifying the entry points and methods used by the attackers.

3.Safeguard and vulnerability identification

3.1 Update and Patch management

Implement an update and patch management protocol to address the identified vulnerabilities and prevent similar attacks in the future.

3.2 Identifying and repairing vulnerabilities

Within 2 days of the attack, identify all vulnerabilities that were exploited and potential weaknesses that could be targeted.

Develop and implement fixes for these vulnerabilities, prioritizing them based on the risk they pose to the system.

Layout and Uniformity

Web Technologies

The web technologies have (mostly) met all desired standards (as required by our Requirement Specification). Due to changes in expectations, we have had to adapt and format as best we could with regards to HTML components within React, especially.

- HTML
    - Where possible, all HTML elements loaded on the website (built using React) conform to HTML Living Standard. All HTML across webpages is consistent with regards to duplicated components.
- Style Sheets (CSS/SCSS)
    - all formatting for the website consists of .scss style sheets (Sass). This is to ensure more readability and robustness in our styling. Furthermore, it helps us to cope with complexity, as several functionalities are implemented in Sass that are not in standard CSS. This not only helps us now, but in the future, as we develop the website, and it becomes more complex. Sass also indicates if there are any errors in the syntax of a styling sheet (Sass, n.d.). Therefore, we can be sure there are no syntax errors.
- JavaScript
    - all website JavaScript have been linted to ensure it meets Web Core Vitals standards (meeting Google/Android's standards, as well as Next.js').
- Carbon Design
    - Carbon Design react components and icons have been incorporated into the website, matching the style and layout favored by our client.

Python

The Python code for creating sound files that utilizes the IBM Watson Text to Speech API is written with an eye on PEP 8 to assure clean readability, maintainability, and consistency in the coding style. This commitment is visible in consistent indentation at four spaces, proper snake case, effective use of spaces, and constraining the length of lines for wider legibility across editors. There are extensive docstrings with clean organization of imports for all classes without assertions on standard library imports. An illustrative example of careful exception handling is all these practices make the code in line with the best practices followed by the wider python community and thereby guide such scalable future development.

Unity/C#

When developing the Unity AR component, ensuring that the code adheres to common and understandable standards is crucial for maintaining the readability, consistency, and overall code quality for collaboration in our project. Unity doesn't have a universally enforced style guide therefore our team has adopted a set of well-defined coding guidelines and best practices focusing on consistency, naming conventions, and spacing as laid out by Avangarde Software in this document: https://avangarde-software.com/unity-coding-guidelines-basic-best-practices.

- Naming Conventions
    - Namespaces shall be PascalCase
    - Classes and Methods shall be PascalCase
    - Fields are camelCase (including public fields)
    - Static Fields shall be PascalCase
    - Parameters should be camelCase
    - Callback suffixes shall be added to delegates
    - Events and actions shall have an 'On' prefix
- Basic Best Practices
    - Declarations always use access level modifiers
    - Single declaration per line
    - Only one class/interface per source file (exception being internal classes)
- Spacing
    - Utilize basic Visual Studio spacing conventions (4 spaces per tab, etc.)
    - Use Vertical Spacing
- Brace Style
    - All braces get their own line

Documentation (Documents of Application)

Our documentation is currently situated in a directory - 'Documentation' - alongside our code on Github. Although we initially thought manually annotated Postman-generated documents would satisfy any documentation requirements, we realized that the software would not be satisfactory and could not cover all the technologies/languages that we had employed. We have used Postman to document certain components, e.g., the Quantum Quiz. However, we have also utilized Doxygen (Wikipedia, 2022) to document our C# and Python code, and the main website framework has been documented through the JSDoc mark-up language (@use JSDoc, n.d.). Additionally, we have conformed to Unity Technologies' documentation recommendations, regarding packages (Technologies, U., n.d.).

User Manual 3 – Deployment

System Maintenance

As our product consists of 4 independently coded components that have been integrated together (the website framework, Unity AR component, Watson chatbot, and Watson Text-to-Speech), the maintainability index has been calculated for each available component to better illustrate the complexity of our system, how it could be improved, and how it could be maintained.

Although the Watson Text-to-Speech code is not currently called in our product, it is used to generate audio files for our Unity AR component and, based on our plans for future development, could eventually be called by the Watson Chatbot. Therefore, we have also taken this component's complexity into consideration. The Watson chatbot, however, has been built using IBM's systems and, therefore, its complexity cannot truly be evaluated, due to proprietary software issues and, therefore, has not been evaluated code complexity-wise. Only one Unity C# file has been coded manually – the rest has been generated during builds. We believe this does not reflect the extent of our product as standard code complexity metrics would be ill-suited to measuring the complexity of our product. The Quantum Quiz is not fully developed yet and, therefore, has not been included, for the time being.

Component	Halstead Volume	Cyclomatic Complexity	Lines of Code	CM	Maintainability Index
Watson Text-to-Speech	28.53	Varying ratings for complexity (ranging from 1 – 10)	77	40%	A (as returned by Python Radon library)
Website Framework	N/A	N/A	2984	0.007%	N/A

As we could not find any tools available for automating the metric evaluation of cyclomatic complexity, Halstead volume, or maintainability index in JavaScript, we have decided to rely on the LOC (Lines of Code) and CM (Comment Percentage) to evaluate maintainability. Based on the Watson Text-to-Speech metrics, the code to generate audio files is easily maintainable. The 2984 LOC in the website framework are readable and 2471 of them are contained within the auto-generated 'yarn.lock' file. However, we will have completed thorough documentation of the JavaScript (via JSDoc techniques) by the Product Handover deadline. This will increase the LOC, but also increase the av. percentage of comment lines – this will make it easier for future developers to understand the code and, should a tool be found to automatically assess the maintainability index of the site, contribute to the maintainability index score. Ultimately, we intend to perform some refactoring of our code, regardless of metrics (we have already performed some refactoring, eliminated unhealthy package dependencies, and tidied our code).

Refactoring our code, following deployment, will certainly benefit our product and improve our Google Lighthouse performance ratings (as we move away from using older libraries). We aim to update any imported modules and libraries regularly and consistently check for any deprecations.

Additionally, as discussed above, we hope our thorough and standardized documentation will aid any future developers working on this project to quickly comprehend the full functionality of the product (program comprehension), although we recognize this would only be beneficial with regards to corrective, perfective, and preventive maintenance). Any changes to code can be merged with the main branches of the relevant GitHub repositories and these changes will be automatically reflected in the main website. Any git commits should be pushed (initially) to the branches we have created (e.g., a change to the website code would be pushed to the 'Main-Website' branch) and will undergo reviews before being merged with the main branch.

We would recommend completing a summative evaluation before any future changes are made and non-essential system maintenance is conducted, utilizing controlled and natural settings (field studies) to understand the effectiveness of the system. Usability testing in a controlled setting, such as a living lab, may improve our understanding of how the user interface could be improved and identify any small faults (e.g., buttons not working correctly) before the system is fully deployed. Field studies in natural settings would also be beneficial (especially if the test users were the intended user demographic – university and high school students). The users would be asked to record observations through diaries and notes. Not only would this be less obtrusive than living labs and, more generally, controlled settings, it seems to reflect IBM's approach to system maintenance, as they requested our group to document our experiences with IBM Skillsbuild courses.

By using up-to-date and relatively new technologies (e.g., Needle Engine, recent Unity versions, and supported node modules), we aim to extend the lifespan of the product, preventing it from becoming a legacy system in the near future. Additionally, this will help shift the focus of maintenance from re-engineering to refactoring. If we were to also work on maintaining this software, we would consider the number of requests for corrective maintenance and number of outstanding change requests (both of which could be measured by viewing the 'Issues' section on our GitHub repository – once it is made public). This would allow us to assess the maintainability of the product.

Future Development

Dynamic Audio Generation

For audio generation, coding was developed in python with the IBM Watson Text to Speech. Generally, it converts text into audio files and then automatically places them in a folder created for this purpose. These files are then given to the Unity project for narration. In the future, to make it smoother, direct text-to-speech rendering will need to be made in real-time rather than generating massive WAV files. The reason it will be an advantage is that we can have text created by AI and spoken out loud, instead of creating the script ourselves and turning them into WAV files, leading to a somewhat limited script for users to hear.

Shift to IBM Cloud

The implementation of the application utilizing IBM cloud as per the client's request is a task for the future. There are two avenues we could take, one harder but arguably more suitable. The first is using the IBM cloud foundry for easy deployment of our web application. It supports the necessary programming languages and frameworks compatible with Unity web builds. This would be the most straightforward path to get our Unity application running on the cloud.

The second option would be the IBM Kubernetes service which is typically reserved for applications requiring both high scalability and high availability. It would allow us to' containerize' our Unity application and manage deployment using Kubernetes, the main advantage this offers is robust scaling and management features at the cost of extra time to deploy and develop.

We believe going forward we should utilize Kubernetes for our project.

There are additional features that we would be interested in implementing given IBM's extensive library of features. These are:

- The utilization of IBM cloud's CDN to ensure lower latency and faster loading times, enhancing the user experience.
- IBM Cloud Object Storage: IBM Cloud Object Storage is used for storing and managing large volumes of data, such as user interaction logs or educational content updates.
- IBM Db2 can be used for data management and analytics. This can be used to analyze user engagement and performance, helping refine and personalize the application, such as monitoring and rectifying when people leave the application prematurely.

The technologies required to do this are:

- Unity WebGL/Needle Engine
- Docker
    - Since we'd be using IBM Kubernetes Service, Docker is essential for containerizing the Unity application. This involves creating a Docker image of the application that can be deployed and managed within a Kubernetes cluster.
- IBM Cloud CLI and SDKs
    - IBM Cloud Command Line Interface (CLI) and Software Development Kits (SDKs) for management of IBM Cloud services within the application development workflow.

Other Skillsbuild Courses

Our client, IBM, has provided us with useful resources via 'SkillsBuild Courses' from which if we integrate insights and knowledge from these courses into the system's future can significantly enhance the project's scope. Efficiency and innovation. These courses are designed to offer in-depth knowledge and skills from technological domains which can help the team's continued learning.

The client had asked us to identify and review relevant courses and review each one, sending weekly reports on our progress. Given more time and continued efforts to complete each course, the team could fill knowledge gaps and enhance expertise in critical areas that can benefit the systems development.

Continued progress through these SkillsBuild courses encourages the team to leverage innovative ideas for the cutting-edge technologies featured in the courses. Some of these features include Artificial Intelligence, Cloud-based Technologies, and Quantum Computing.

By systematically incorporating the knowledge and skills acquired from IBM's SkillsBuild courses, the team can ensure that the systems development is aligned with the latest technological advances of the client, industry best practices and innovative trend. This would enhance the system's current functionalities in addition to positioning the software in a good position for future growth and adaptability.

Ethical and Societal Impacts

Although the system does not pose much risk (legally) to either its users, developers, or our client, we have still taken into consideration the potential ethical impacts of the system (for example, consumer privacy) and its societal impacts too (for example, defective software liabilities).

As our system takes an input (user questions) through the Watson chatbot and processes it (using IBM Watson Assistant), we have considered data protection regulations. Based on IBM Watson's Privacy Policy and GDPR, we will protect any user inputs to the Watson Chatbot to ensure user trust and conformation to data protection laws. We will highlight IBM Watson's Privacy Policy to increase transparency and act lawfully (both out of legal obligation and to improve our users' understanding before they consent to processing of their data). This will also be beneficial once the Quantum Quiz is fully developed and deployed on the application as well (as that also takes user input). IBM Watson may use user inputs to train its model further (IBM, n.d.), but we will indicate that (as a purpose limitation) to the user before they enter any information and Watson will not keep any data longer than is necessary, incorporating a storage limitation.

We have built our system for a specific client – IBM. This introduces ethical questions around ownership and property. We may have to negotiate who can claim Intellectual Property rights in the future with regards to the system, as there is no formal contract to enforce or negate copyright infringement, due to this being academic experience/coursework. We may have a claim to automatic copyright and could include a copyright notice (e.g., 'Copyright © Software_Engineering_Group_26 2024'). Furthermore, we could explore licensing in the future. IBM may like to develop our system further and create a new system, based off ours, that is relevant to other IBM Skillsbuild courses (see *Future Developments* section) and that may require a license to develop. If our software is defective in some way, it will not pose a large risk to any user's rights or safety. However, should IBM deploy our software in different settings, we must determine who is liable should it break.

As our software may benefit many educational institutions and the greater public, we are inclined to make our product open source (although, possibly subject to 3-clause BSD (Open Source Initiative, 2011) or MIT licensing (Open Source Initiative, 2006). This would allow IBM to both use it in its current state and produce derivative works from it (for other Skillsbuild courses). Once again, however, this will need to be negotiated with IBM, as current ownership may be viewed as ambiguous.

Although we have attempted to apply significant measures to prevent discrimination, as defined by the Equality Act 2010, in our software (e.g. by including gender neutral language where applicable, by including subtitles) and have discussed plans for further prevention of discrimination in future iterations of software, we have created a plan on how to improve our software should we receive any reports of discrimination from users:

1. Report is submitted.
2. Report is addressed and processed within 24 hours.
3. Response is sent to report.
4. Planning/brain-storming Stage – discuss how to counteract this discrimination in our system.
5. Design phase – planning how to integrate changes.
6. Implementation phase – code discrimination-prevention measure.
7. Deploy phase – integrate discrimination-prevention measures.

References

- Wikipedia. (2022). Doxygen. [online] Available at: https://en.wikipedia.org/wiki/Doxygen.
- Wikipedia. (2024a). Software documentation. [online] Available at: https://en.wikipedia.org/wiki/Software_documentation#Technical_documentation [Accessed 4 Mar. 2024].
- Wikipedia. (2024b). JSDoc. [online] Available at: https://en.wikipedia.org/wiki/JSDoc [Accessed 4 Mar. 2024].
- Sass. (n.d.). Sass: Parsing a Stylesheet. [online] Available at: https://sass-lang.com/documentation/syntax/parsing/ [Accessed 4 Mar. 2024].
- Open Source Initiative. (2006). The MIT License. [online] Available at: https://opensource.org/license/mit.
- Open Source Initiative. (2011). The 3-Clause BSD License. [online] Available at: https://opensource.org/license/bsd-3-clause [Accessed 4 Mar. 2024].
- web.dev. (n.d.). Web Vitals | Articles. [online] Available at: https://web.dev/articles/vitals.
- IBM Carbon (n.d.). Carbon Design System. [online] Available at: https://carbondesignsystem.com/developing/react-tutorial/overview/ [Accessed 4 Mar. 2024].
- YouTube (n.d.a). Needle Engine - AR Image Tracking Tutorial. [online] Available at: https://www.youtube.com/watch?app=desktop&v=QYd2XPL0WW0 [Accessed 4 Mar. 2024].
- YouTube (n.d.b). How to Deploy Your Vite App to Github Pages. [online] Available at: https://www.youtube.com/watch?v=yo2bMGnIKE8 [Accessed 4 Mar. 2024].
- IBM (n.d.) Watson Privacy, Compliance, & Security. [PDF] Available at: https://www.ibm.com/watson/assets/duo/pdf/Watson-Privacy-and-Security-POV_final_062819_tps.pdf [Accessed 4 Mar. 2024]

Documentation

- Technologies, U. (n.d.). Unity - Manual: Documenting your package. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/Manual/cus-document.html [Accessed 4 Mar. 2024].
- @use JSDoc (n.d.). Use JSDoc: Index. [online] Available at: https://jsdoc.app/.
- GitHub Docs. (n.d.). About GitHub Pages. [online] Available at: https://docs.github.com/en/pages/getting-started-with-github-pages/about-github-pages.
- nextjs.org. (n.d.). Configuring: ESLint | Next.js. [online] Available at: https://nextjs.org/docs/app/building-your-application/configuring/eslint [Accessed 4 Mar. 2024].

Watson Assistant Documentation

- Avangarde-Software. (2019). [Unity] Coding guidelines & Basic Best Practices. [online] Available at: https://avangarde-software.com/unity-coding-guidelines-basic-best-practices [Accessed 4 Mar. 2024].
- IBM Cloud (n.d.). IBM Cloud Docs. [online] Available at: https://cloud.ibm.com/docs/assistant?topic=assistant-skill-add#skill-add-dialog-skill [Accessed 4 Mar. 2024].
- van Rossum, G., Warsaw, B. and Coghlan, N. (2001). PEP 8 – Style Guide for Python Code | peps.python.org. [online] peps.python.org. Available at: https://peps.python.org/pep-0008/.