QUT Capstone Proof-of-Concept Report

QUT | IFB399 | T002

Document Publish Date	17 October 2024
Project ID	P009
Project Title	AR Aircraft Briefing Tool
Industry Partner	CAE Australia
Industry Supervisor(s)	Sung-Jun Lee Chris Schuster
Team members	
n10796461	Kwok, Renee
n11065681	Leonardo Third, Hector
n11338474	Nelson, Keanu
n11029935	Sebastiao, Phillipe

Contents

Report Goals

This report aims to assess the prototype's: *Development Feasibility, Functionality Alignment and User Value* to determine the effectiveness of the AR Aircraft Briefing Tool in its current developed state – the prototype the team has developed thus far – and outline recommendations for its future development. *Note: Refer to Table 12 in Page 15 to view all recommendations that were discussed in this report.*

Development Feasibility

The *Development Feasibility Assessment* evaluates whether the expected features and functionality of the AR Briefing Tool are viable for implementation. As this project did not involve any monetary considerations such as costs and budgeting, the focus will be on the technical requirements to determine the feasibility of development. This section will outline the technical insights the team encountered when developing each feature, their associated issues and risks, and future development considerations.

Software Architecture of the AR Briefing Tool

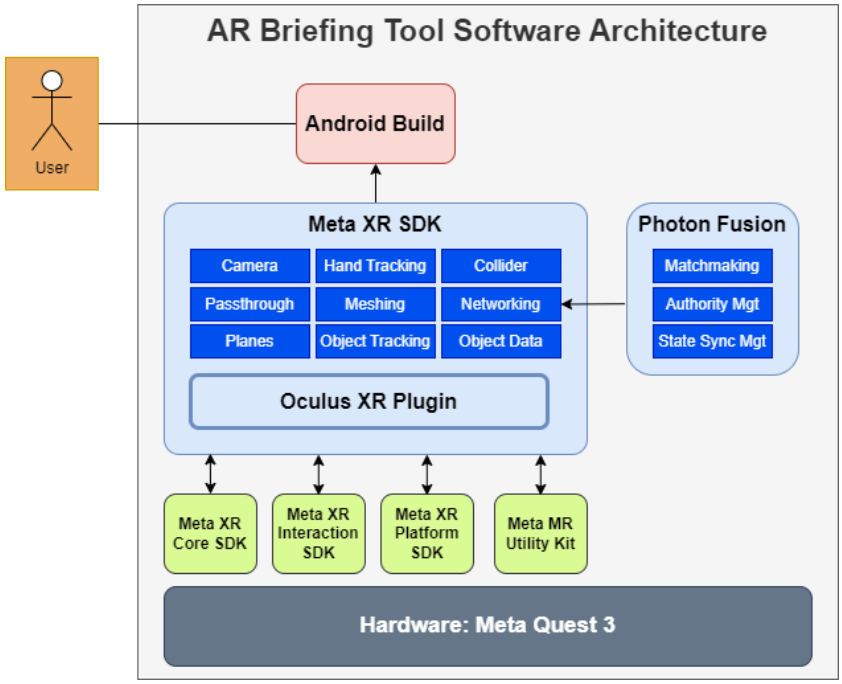

Figure 1 – Software Architecture Diagram of the AR Briefing Tool

As shown in Figure 1, the AR Briefing Tool was developed using the Meta XR Software Development Kits (SDKs) with the Oculus XR Plugin enabled. This enabled the project to be built into the Meta Quest 3 via the Android Build, incorporating essential packages and scripts to support the development of the tool's core features to enable model interactions. These features involve the Meta Quest 3's camera, passthrough, hand tracking inputs, plane detection, and 3D model object tracking, object data, meshing and colliders.

Photon Fusion is an API that enables networking within Unity's game engine. This also incorporates essential packages and scripts to support the networking development, which is crucial in delivering the

Collaboration requirements for this project. Photon Fusion allows players to be connected into the same server with its auto-matchmaking capability, and the management of the 3D model's authority and synchronisation of its state.

Development Insights and Future Considerations

Requirement				Priority (MoSCoW)	Development	
Feature	REQ ID	Dependency ID	Requirement	Business Priority	Completed Status	Technical Requirements
App Models	9	12	View Added Model	Must	Done	- .obj or .fbx model file type
	12	N/A	Add Models To Application	Must	Done	- Unity Environment
Switch Models	29	12	Switch Between Models	Should	Done	- Meta XR SDK
	31	12	Show A List Of Models	Should	Done	- Meta XR SDK - UI Menu Sample
Interactions	5	9	Zoom In & Out of Model	Must	Done	- Meta XR SDK - Grabbable - Object Rigid Body & Collider
	15	9	Rotate Model	Must	Done	- Meta XR SDK - Grabbable - Object Rigid Body & Collider
	20	9	Move Model	Must	Done	- Meta XR SDK - Grabbable - Object Rigid Body & Collider
	22	N/A	Pointer	Should	Not Delivered	N/A
	24	9	Pull Apart Individual Parts	Should	Done	- Meta XR SDK - Ray Interactor - Object Rigid Body & Collider
	32	5,15,20,29,31,24	Toggling Various Features	Should	Done	- Meta XR SDK - UI Menu Sample
	21	N/A	Maintenance Manuals	Could	Not Delivered	N/A
Passthrough	17	9	View Model In Mixed Reality	Must	Done	- Meta XR SDK - Passthrough
	30	17	Model Has Collision With Environment	Could	Not Delivered	N/A
	27	17	Model Has Depth With Environment	Could	Not Delivered	N/A
Collaboration	7	N/A	Join Session	Must	Done	- Meta XR SDK & Photon Fusion - Automatchmaking
	23	9,15,20	Synchronised Model Position	Must	Done	- Meta XR SDK & Photon Fusion: Colocation
	26	9,5,22,24	Synchronised Model Interactions	Must	Done	- Meta XR SDK & Photon Fusion: Object State Sync & Authority
Cross-Sections	3	10	Move Cross Section	Should	Not Delivered	N/A
	10	9	Single Cross Section	Should	Not Delivered	N/A
	2	10	Different Axis Cross Sections	Could	Not Delivered	N/A
	11	10	Multiple Cross Sections	Could	Not Delivered	N/A
Exploded View	13	9	Explode Model	Could	Done	- Meta XR SDK - Interactor
Refined Parts	1	9	Associate Refined Part With a Model	Could	Done	- Meta XR SDK
	6	1	Switch Between Refined Parts	Could	Done	- Meta XR SDK

Table 1 - Development Status of Requirements

Table 1 outlines all the requirements the team aimed to develop in Phase 2 (which excludes the Won't-Have requirements). Items highlighted in yellow represent completed requirements, red indicates undelivered requirements, and orange denotes requirements that were attempted but halted due to time constraints. This table also provides a high-level overview of the technical requirements involved in the development. As mentioned in the previous section, most features were developed using the Meta XR SDKs to enable model interactions and functionality of the prototype, while networking was enabled through Photon Fusion.

Completed Requirements

App Models – Add Models to Application & View Added Model

The 3D models currently used in the prototype were sourced from open-source libraries, this includes the *C130 , Blackhawk* and *X-Wing.* Two primary 3D Model file types were utilised in the project: .obj and .fbx files. A key difference between their file types is the capability to retain the material and/or textual data of the 3D model. Importing external .obj file types into Unity results in models with a default solid white colour, requiring the team to manually apply material data. This is evident in the Blackhawk model, which

lacks visual refinement. On the other hand, the C130 and X-Wing models, which are .fbx file types, retained their material data when they were imported into Unity.

Since certain features involve manipulating individual parts, a key requirement for sourcing models is their capability to have selectable individual parts within the Unity Environment. A model's parts are structured as child class components in Unity, allowing them to be referenced and controlled through game scripts. Given these considerations and insights, the team strongly recommends using .fbx models with selectable individual parts for future development.

Although the feasibility of these requirements was dependent on the availability of suitable 3D models, the existing .fbx 3D models with selectable parts demonstrate its implementation is feasible. While designing 3D models was out of scope in this project, CAE should consider using tools like AutoCAD or Autodesk Inventor to create and integrate their own aircraft models into Unity for future development.

Switching Models – Switch Between Models & Show a List of Models

This feature was implemented in response to the Project Owners' requirement for proof that model interactions could be performed with multiple models. As a result, the ability to switch between models via a UI was deemed important. This feature was feasible to develop with support from Meta's sample scenes, which included a scroll bar interface, and Unity's capability to store model files which can be referenced in the interface's game script and logic.

Refined Parts – Associate Refined Part with a Model & Switch Between Refined Parts

This feature is an expansion to the prototype's concept (and hence, was classified as a Could Have requirement) and was found to be highly feasible to develop, as its functionality closely resembles the *Switch Between Models* feature. It involves a UI that displays a model's refined parts – as demonstrated by C130's turbine engine in the prototype – which can be selected and displayed in the environment. A key part of this feature is associating these refined parts in their respective models which was found feasible to implement in the UI's game scripts and logic.

Currently, the C130 is the only model with an associated refined part due to time constraints. While this feature currently serves as a proof of concept for displaying a model's refined parts, it is highly likely to become a core feature in the future, especially for showcasing a model's engines and other specific parts for maintenance operations and briefings/trainings. Therefore, the team recommends adding more refined parts in the other models for future development.

Model Manipulation – Move, Rotate, Zoom (Scale)

The ability to move, rotate and zoom (scale) the model was essential to the core functionality of the prototype. The team prioritised developing these features using hand inputs and enabling the model to be grabbable in the environment, as we believed this approach would be the most intuitive for users, regardless of their experience or familiarity with AR/VR headsets. Due to Meta's Interaction SDKs, and the extensive documentation available for implementing these features, the team found their development highly feasible. However, issues occurred when scaling the model to a very large size in the environment, which caused a drop in frame rate and making the model appear laggy when being manipulated. Therefore, the team recommends setting a maximum limit for how large a model can be scaled in future development.

Pull Apart Individual Parts

The ability to pull apart a model's individual parts was also essential to the prototype's functionality, enabling end-users to closely examine specific components when needed. The team approached its development using ray grab interactors, which users can toggle if they require pulling individual parts apart. After user testing with the Project Owners, the team also incorporated manipulation of a model's parts by grabbing them directly, providing users the option to use either the ray interactors or their hands to manipulate individual parts. Similar to Model Manipulation, Meta's Interaction SDKs and comprehensive documentations made this feature's development feasible.

However, the functionality for grabbing individual parts directly was implemented just days before the handover, the team did not have sufficient time to conduct user testing for the new UI. Hence, we strongly recommend prioritising its user testing in future development.

Toggling Various Features

Although this feature was not explicitly requested by the Project Owners, the team found its inclusion essential, as it enables users to toggle and activate the functionality of developed features, which includes toggling individual parts, exploded view, switching between models and viewing refined parts. Given the nature of mixed reality, users are most likely to move around the room while interacting with the model. To accommodate this, the team approached its development to avoid a static UI menu, ensuring the UI appears dynamically wherever the user moves. The team initially utilised Meta's Palm Menu UI feature to meet these requirements but was later changed to a pop-up grabbable menu after user testing with the Project Owners. The development of this feature was feasible due to Meta's sample scenes and SDKs.

However, since the menu UI was completed just days before the handover, the team did not have sufficient time to conduct user testing for the new UI. Hence, we strongly recommend prioritising its user testing in future development.

Explode Model

This feature underwent changes to its priority, changing from a *Must Have* requirement to *Could Have.* The team proposed this change as the *Pull Apart Individual Parts* feature offered greater value for viewing individual parts compared to exploding the entire model assembly. Despite the adjustment, our developers considered this feature feasible to develop, given the model's available parts and their prior knowledge of robotics concepts.

Its development was more complex compared to the other requirements however, due to the application of vector mathematics, which includes linear interpolation (to facilitate an object's movement from Point A to Point B). This knowledge was essential for determining how the model's individual parts will move in desired directions. Despite the complexity, this feature was feasible to develop due to some prior knowledge, even with the lack of support from Meta's SDKs and documentation.

It is important to note that the current functionality of the Explode Model feature is highly dependent on the model's file type and the availability of their individual parts. Exploding .obj models is not ideal, as they lack comprehensive detail in the exploded view, which was evident with the Blackhawk model. In contrast, .fbx models provided a more visually appealing result. The availability of individual parts is also crucial, which was evident through C130's refined engine model, which consisted of only three parts in its file. When this model was exploded, its functionality appeared ineffective due to the low number of

individual parts. Therefore, the team strongly recommends utilising .fbx models with a large number of individual parts where possible in future development.

Collaboration – Join Session, Synchronised Model Position, Synchronised Model Interactions

These requirements were crucial for enabling the collaboration aspect of the AR Briefing Tool and proved to be one of the most challenging features to implement. Despite the significant time invested towards its development, the team was only able to complete them in the final week of the project, just before our handover.

To enable Collaboration, the following conditions needed to be met:

- Two headsets must be connected to the same server and view the same model
- Changes to the model's state due to interactions, for instance, model manipulations, pulling apart individual parts and switching between models, must be synchronised in other player's headset in real time
- The model should have an anchor point in the environment, so that players are able to view the model relative to their position – "Colocation", in other words.

As mentioned previously, the team utilised Photon Fusion to enable the networking components in Unity. The availability of an "Auto-matchmaking" component through Meta XR components with Fusion allowed the first condition – connecting players to the same server – to be successfully completed. However, the main issues lie in implementing colocation and synchronising interactions.

The team encountered significant challenges in synchronising model interactions. While basic model manipulation, like moving, rotating and scaling, could be successfully synchronised, more complex interactions, such as pulling apart individual parts and switching models, resulted in errors. For the "pull apart individual part" interaction, although its movement could be synchronised, prevailing issues occurred where only one player was able to perform the action, while the other player was unable to do so. Additionally, when one player switched a model, it resulted in an issue where the other player's environment would display no model. Due to the constant errors, our developers prioritised the development of these features and conducted extensive research using Photon Fusion documentations and engaging in trial and error development to resolve them.

Since switching between models involves toggling via the Model List UI, the team implemented remote procedure calls (RPCs) to broadcast that event across all connected clients. This ensures that when one player switches a model, all other users can see the same change. Network components were added to both the model and its individual parts to enable network transforms, with Photon Fusion (specifically Fusion 2) facilitating the synchronisation of the model's network transforms between each clients. Hence, both the RPCs and network components enable synching model interactions. On the other hand, colocation was developed through Unity's Shared Spatial Anchors, which is enabled using Fusion 2. The Shared Spatial Anchors shares the spatial data of an anchor's transform, positioning all other objects relative to the anchor's transform. It is important to note that the project had to be deployed and published as an official app to enable shared spatial anchors. Its development was also found to be highly complex due to the lack of documentations and tutorials to assist with its development. Despite the complexities, it is evident that the technologies offered by Photon Fusion have made the development of the Collaboration features feasible.

However, when the team conducted testing, the following issues were prevalent:

- Control of a model's state is managed through Photon Fusion's *state authority* components, which requires a player to directly touch the model to transfer authority. If a player requires manipulating a model's individual parts, they must first acquire state authority by touching the model, before they are able to pull apart individual parts. This resulted in situations where a second player could not manipulate the model unless they interact with the model directly first, which may cause some inconvenience with their user experience. Therefore, further research is needed to explore how ownership can be transferred through different interactions in future development.
- There were issues where switching models right after the application's initial bootup caused them to disappear when a player attempted to interact with the newly switched model. The team discovered a workaround where a player must first touch each model before a second player joins the environment to prevent the issues to occur. However, this is a potential bug that must be prioritised for resolution in future development.

It is important to note that the features can operate without any issues for local interactions with a single player in the environment. However, the outlined problems can only occur when a second player joins the environment. Therefore, enhancing and troubleshooting the Synchronise Model Interactions feature must be prioritised in future development.

However, since the Collaboration features were completed just days before the handover, the team did not have sufficient time to conduct its user testing with the Project Owners. Hence, we strongly recommend prioritising user testing for the Collaboration features in future development.

Undelivered Requirements

Pointer

The purpose of a pointer was to enhance the collaborative aspect of the AR Briefing Tool by allowing technicians to point out and highlight specific parts of the model during briefings and training sessions. However, its development was put on hold as it relied on the completion of the collaboration features. Implementing a pointer in a single user, local environment would be ineffective without the capability for multiple users to interact collaboratively. The pointer's development also was discontinued due to the significant time spent on developing the collaboration features, leading to time constraints.

Although the team did not explore the concept of a visible pointer in Phase 2, we believe that it is feasible to implement with Meta XR's existing ray interactors. Currently, when individual parts mode is toggled, a user pointing to part becomes highlighted. Therefore, more research and testing are required to enable and synchronise a visible ray as a networked object.

Cross Section (baseline) – Single Cross Section and Move Cross Section

The cross section feature enables a detailed view of a 3D model's internal structures and layers, enhancing the end-user's understanding of an aircraft's design and functionality. The team approached its development by implementing a grabbable cross section pane, allowing users to "cut" into the model and examine its internal components. However, due to the niche nature of this functionality, there was a lack of available documentation and tutorials to assist in developing this feature. While cross section SDKs were available in the Unity Asset Store, they required costly payments to download. Despite our effort to develop this feature from scratch using Unity's shader materials, we encountered persistent issues that

either turned the entire model into one solid colour or completely transparent. The team had to halt this feature's development to prioritise the development of the collaboration features.

Further research is required to implement this feature in future development. CAE may need to explore different development approaches to enable the cross-section functionality. It is also important to monitor for any cross section SDKs or sample scenes that may be published for free. Alternatively, CAE could consider purchasing an existing cross section SDK from the Unity Asset Store, but thorough research should be conducted to ensure it is compatible with mixed reality development before making a purchase.

Could Have Requirements

The team prioritised the development of Must-Have and Should-Have requirements, with some Could-Have requirements completed depending on their feasibility. However, a few remaining Could-Have requirements could not be initiated due to time constraints and dependencies. These include the Maintenance Manual, Model has Collision and Depth with the Environment, and Different and Multiple Cross Section requirements. While the team lacks first-hand development insights into the feasibility of these features, we will outline potential suggestions for the AR Briefing Tool's future development.

Maintenance Manual

The purpose of this feature is to allow technicians to view a maintenance manual while interacting with a 3D model within the mixed the reality space. The existing UIs like the scroll bar interface in Meta makes the feature's development feasible. However, key considerations would include how to integrate the digital copy of CAE's maintenance manuals (a PDF file for instance) into the environment. As the use of maintenance manuals are crucial to assist in maintenance, briefing and training operations, the team considers feature to potentially become important in future development.

Model has Collision and Depth with the Environment

The purpose of these features was to enhance realism when displaying a 3D model within the user's environment. This would allow the model to interact and collide with real-world objects – such as resting on a table, for instance, and add depth to the user's perspective – for example, partially obscuring the model when it is positioned behind a chair. As Unity and the Meta XR SDK have physics and passthrough capabilities, these features could be feasible to develop. However, further research for implementing object physics and changes to the passthrough capabilities may be required as their development could be complex. However, as these features are purely for aesthetic purposes, the team recommends making its development the least prioritised for future development.

Cross Section (expansion) – Different and Multiple Cross Sections

These features are an expansion to the baseline Cross Section features by enabling cross section panes to appear on different axes and allowing users to add multiple cross sections panes in the environment. However, given the halted development of the baseline Cross Section features, it is recommended to complete the baseline Cross Section before adding these features. Further research and testing may be required if this concept is feasible to develop in future development.

<u>Other Development Considerations</u>

Improved UI

The current prototype design only displays the model in the environment, with no additional guidance or interface. Therefore, the team recommends adding more user-friendly features in future development. This could include instructional pop-up interfaces to guide users on enabling features and providing

instructions on how to manipulate the model for different features, for example, the hand motion required to scale a model.

How AI can be Implemented

The prototype's current functionality depends on developers manually adding the necessary components and game scripts in each model. If users want to add more models, the process of adding the components and scripts would always occur. Therefore, Implementing AI could streamline this process by automating the addition of the components, allowing the application to integrate new models seamlessly into the environment. This automation could significantly reduce manual intervention and should be considered in the later stages of future development.

Team's Lack of Game Engine Development Experience

It is important to note that the team had no prior experience developing software with game engines. We relied on our current knowledge of the C# language, as well as extensive Meta documentation and tutorials to assist us through the development process. While we successfully delivered majority of the features we aimed to implement, our lack of expertise also presented significant challenges for some features. This should be taken into consideration for future development, especially if CAE plans to pursue more advanced features.

Future Development Consideration – Summary

The following table provides a summary of all future development considerations and recommendations discussed in this section.

Requirement	Future Development Considerations & Recommendations
App Models – Add Models to Application & View Added Model	• Utilise .fbx models with selectable individual parts • Consider using AutoCAD or Autodesk Inventor for integrating custom aircraft models (CAE's aircrafts for instance) into Unity
Switching Models – Switch Between Models & Show a List of Models	• N/A
Refined Parts – Associate Refined Part with a Model & Switch Between Refined Parts	• The team considers this feature to become a core feature in the future to showcase models' respective refined parts • This feature must be explored further by adding more refined parts to the other existing models
Model Manipulation – Move, Rotate, Zoom (Scale)	• Implement maximum limit for how large a model can be scaled to avoid dropping frame rates and lag
Pull Apart Individual Parts	• Conduct user testing for pulling apart individual parts when they are directly gabbed by a user
Toggling Various Features	• Conduct user testing for the current development of the Collaboration Features
Explode Model	• Utilise .fbx models with many individual parts as possible
Collaboration – Join Session, Synchronised Model Position & Interactions	• Conduct user testing for the current development of the Collaboration Features • Explore how to effectively manage transferring a model's state authority through other interactions (other than touching a model)

	• Troubleshoot potential bugs that is causing a model to disappear when switching between models upon the application's initial bootup.
Pointer	• Explore Meta's Ray Interactors to enable visible rays • Explore how visible rays can become synchronised as a networked object so other users view the pointer as well
Cross Section (Baseline) – Single Cross Section and Move Cross Section	• Further research is required to implement this feature • Alternative development approaches may need to be explored • Monitor future publishing of free cross section SDKs or sample scenes • Consider purchasing existing cross section SDKs in the Unity Asset Store; thorough research should be conducted if these would be compatible with mixed reality developing before making a purchase
Cross Section – Different and Multiple Cross Section	• Ensure the baseline cross section features are implemented first before exploring these requirements • Further research and testing required to implement these concepts
Maintenance Manual	• Utilise Meta's existing scroll bar UI to implement this feature • Key considerations include how the digital copy of CAE's maintenance manuals (PDF files for instance) can be integrated into the environment – PDF converter API? Etc.
Model has Collision and Depth with the Environment	• Explore Unity's physics features to enable object collision with real-world object • Explore Meta's passthrough features to explore depth in the AR view • Further research may be required due to possible complexities with its implementation • These features should be the least prioritised as it is purely for aesthetic purposes
Other Considerations	• Adding more user-friendly features to guide users on how to utilise the product • Possible AI integration in the later stages of future development to streamline manual development whenever a new model is added to the application • Expertise with Unity game development may be required to implement more complex features

Table 2 – Summary of future development considerations and recommendations based on development insights

Functionality Alignment

The *Functionality Alignment Assessment* evaluates whether the developed features and functionality of the AR Briefing Tool is aligned with the project's scope and objectives, and if its quality, functionality and usability (ease-of-use) is aligned with user expectations.

A *User Testing Feedback s*urvey was created to evaluate the functionality alignment. Since there were no end-users – such as technicians and trainees for instance – available for user testing, the Project Owners completed the survey instead. This section will focus on the *Quality of Deliverables, Project Scope and Goals* and *Functionality and Usability* aspects evaluated in the survey. Please refer to the *User Testing*

Feedback excel spreadsheet to view the survey insights. It should be noted that the feedback was only gathered for the completed features listed below:

Feature	Completed Requirements
App Model	• View Added Model • Add Models to Application
Switch Models	• Switch Between Models • Show a List of Models
Interactions	• Zoom in & Out of Model • Rotate Model • Move Model • Pull Apart Individual Parts • Toggling Various Features (Palm Menu)
Passthrough	• View Model in Mixed Reality
Exploded View	• Explode Model
Refined Parts	• Associate Refined Part with a Model • Switch Between Refined Parts

Table 3 – List of completed requirements

As discussed previously, although the Collaboration features were completed, user testing for *Join Session, Synchronise Model Interactions and Position* could not be conducted due to time constraints; therefore, these aspects will not be discussed in this section.

Summary of Results and Implications

To calculate the summary of results, the response for each requirement from their corresponding survey question were counted. For more details of how each requirement were answered, please refer to the *User Testing Feedback* spreadsheet.

Quality of Deliverables

Q – How would you rate the quality of this deliverable

Requirement Answer	Count Result for Sung-Jun Lee	Count Result for Chris Schuster
Excellent	9	6
Good	1	4
Fair	0	0
Poor	0	0

Table 4 – Count result for the Quality of Deliverables question

Both Project Owners found the quality of the completed requirements to be positive as shown from the results above, indicating that they were satisfied with the end results and there were no major bugs, errors, or glitches prevalent.

Project Scope and Goals

Q – Is this deliverable aligned with the project's scope and goals?

Requirement Answer	Count Result for Sung-Jun Lee	Count Result for Chris Schuster
Yes	10	10
No	0	0

Table 5 – Count result for the Project Scope and Goals question

The results above reveal that both Project Owners agree that all the deliverables developed were aligned with the project's scope and goals.

Functionality Alignment

Q – The functionality/design of this deliverable aligns with my expectations

Requirement Answer	Count Result for Sung-Jun Lee	Count Result for Chris Schuster
Strongly Agree	7	5
Agree	1	4
Disagree	1	0
Strongly Disagree	0	0

Table 6 – Count result for the Functionality Alignment question

Majority of the deliverables met the Project Owners' expectations. However, one deliverable – the *Toggling Various Feature* requirement – had a disagreed response from one Project Owner. As commented by Sung-Jun Lee, he had a preference towards a pop-up menu for toggling features instead of a palm menu. He notes that it "takes time to get used to the Palm Menu's action", and its interaction is too "intimate" which may be disruptive for briefings and trainings.

Intuitiveness (Ease of Use)

Q – I found the deliverable intuitive and easy to use

Requirement Answer	Count Result for Sung-Jun Lee	Count Result for Chris Schuster
Strongly Agree	2	7
Agree	6	1
Disagree	1	1
Strongly Disagree	0	0

Table 7 – Count result for the Intuitiveness question

The Project Owners found that majority of the deliverables were intuitive and easy to use. However, there were disagreed responses for *Pulling Apart Individual Parts* and *Toggling Various Features* from Sung-Jun Lee and Chris Schuster respectively. Sung-Jun Lee suggests touching the model to pull apart individual parts instead of the current ray interactors. Meanwhile, Chris Schuster found that swiping through the features in the Palm Menu was inconsistent, occasionally requiring multiple attempts to successfully swipe. He recommends adding a "left" and "right" button near the palm menu UI to scroll through the features as an alternative to swiping. The team later changed the Palm Menu to toggle features with a main scroll bar UI menu instead, as mentioned previously in the Development Feasibility section based on the suggestions and remarks given from the Project Owners.

Learning Complexity

Q – I found that I needed to learn a lot of things before using the deliverable

Requirement Answer	Count Result for Sung-Jun Lee	Count Result for Chris Schuster
Strongly Agree	0	0
Agree	0	0
Disagree	6	1
Strongly Disagree	3	8

Table 8 – Count result for the Learning Complexity question

As shown from the results above, majority of the Project Owners responses to this question were *Disagree* and *Strongly Disagree* for the developed requirements; indicating that using the deliverables were straightforward and not overly complex to learn.

Potential Future Considerations

The following table summarises future development considerations based on the insights gathered from the *Functionality Alignment* assessment survey questions and additional comments from the Project Owners which have not yet been addressed.

Requirement	Future Development Considerations / Comments
Toggling Various Features	• Preference towards a main pop-up menu for toggling features instead of a Palm Menu. It takes time to get used to the action and its interaction is too "intimate" with how it works which may be disruptive for briefings and trainings • Swiping through the features in the Palm Menu is inconsistent, occasionally requiring multiple attempts to successfully swipe. Suggestion to add a “left and right” button for scrolling through the features as an alternative to swiping. • UI can be confusing at times – Selecting the ‘X’ on the Palm Menu is not as intuitive, makes it seem you are ‘disabling’ the feature instead of enabling it. Suggestion to make it blank instead, and show a tick (like normal) when selecting • *Due to these remarks, the Palm Menu was changed to a grabbable pop-up menu. Hence, user testing for the new UI must be conducted*
Pull Apart Individual Parts	• Preference for touching the model to pull apart individual parts instead of a ray interactor • *Due to the above remark, the team added another functionality to directly grab a model’s individual parts* • Potential issues for green models as it would be hard to see what individual part is highlighted. Suggestion to change the colour when a part is highlighted into another colour • Add instruction pop-ups in the environment when individual parts mode is toggled to show how the ray interactor works • Preference to see visible rays when pointing to a part
Explode View	• Suggestion to use hand gestures (similar to zooming in/out) to explode the model as an alternative to using a slider • Suggestion to set a maximum boundary for how far the model can be exploded as some parts tend to go further away in the room

Switching Between Models	• This feature may not be utilised often, as one aircraft tends to be retained for a long term before a newly aircraft is made. This feature is a good thing to have regardless • Based on above comment, further enhancements may not be required for this feature and hence can be least prioritised in future development

Table 9 – Future Development Considerations and Comments outlined in the Functionality Alignment assessment

Based on the insights gathered, future development should prioritise changes to the current functionality and design for *Toggling Various Features, Pull Apart Individual Parts* and *Exploded View*. This is due to the numerous suggestions made from the Project Owners for improving these areas; as well as the disagreed responses for their functionality alignment and intuitiveness from the *Pull Apart Individual Parts* and *Toggling Various Features* requirements.

User Value

The *User Value Assessment* evaluates whether the AR Briefing Tool would provide end-user value for CAE's maintenance, teaching or maintenance operations. Insights gained from this section, along with the findings from the previous section can help gauge the user acceptance upon the tool's final implementation.

This section will focus on the *User Value* aspect of the *User Testing Feedback* survey. As mentioned previously, the Project Owners completed the survey instead as there were no available end-users for user testing. User Value feedback is only gathered for completed deliverables outlined in Table [ADD number] in the previous section. It should be noted that the *View Added Model, Add Models to Application, View Model in Mixed Reality* and *Associate Refined Parts* deliverables were excluded in the User Value assessment as they are built-in features.

Use Case

The following table describes the intended use of the AR Briefing Tool, and the features that enable the use case. Note: Model Manipulation includes the *Move, Rotate and Zoom* features and Collaboration includes the *Join Session, Synchronised Model Positions and Interactions* features.

Users	Use Case	Features Enabling Use Case
Technicians	Maintenance Operations – Complete aircraft maintenance tasks for flight preparations with the support of the AR Briefing Tool	Model Manipulation Cross Sections Toggling Various Features Explode Model Switch Between Refined Parts
	Conduct Maintenance and Aircrew Trainings and Briefings for aircraft parts and engines through the AR Briefing Tool	Model Manipulation Cross Sections Toggling Various Features Explode Model Switch Between Refined Parts Collaboration Pointer

Trainees	View the aircraft being manipulated during maintenance and/or aircrew trainings for learning	Collaboration

Table 10 – Use Case Description and the Features that enable the Use Case

Summary of Results and Implications

Q – I believe this feature will bring value to its end users, either for teaching operations, briefings or maintenance operations.

Requirement Answer	Count Result for Sung-Jun Lee	Count Result for Chris Schuster
Strongly Agree	7	9
Agree	2	0
Disagree	0	0
Strongly Disagree	0	0

Table 11 – Count result for the User Value assessment

As shown from the results above, both the project owners believe that the developed features will bring value to the end users of the AR Briefing Tool based on their interaction with the prototype.

Potential Future Considerations

Although the User Value assessment results were positive, it is crucial to involve actual end-users, such as technicians and trainees, in testing the AR Briefing Tool prototype for future development. End-users have practical insights from their real-world maintenance, briefing and training operations, which allows them to identify usability issues, functional gaps and/or areas for improvement that may not be prevalent in the user testing conducted by the Project Owners. Their practical feedback ensures the tool can be tailored to their needs, ultimately improving its effectiveness and user adoption for their intended use case.

Final Recommendations for Future Development and Conclusion

Summary of All Future Recommendations

The following table outlines all the future development consideration and recommendations that were discussed in the entire report.

Requirement	Future Development Considerations & Recommendations
App Models – Add Models to Application & View Added Model	• Utilise .fbx models with selectable individual parts • Consider using AutoCAD or Autodesk Inventor for integrating custom aircraft models (CAE's aircrafts for instance) into Unity
Switching Models – Switch Between Models & Show a List of Models	• The Project Owners considers this feature to become the least utilised due to the rollout of newer aircrafts • Further enhancements for this feature can be the least prioritised in the future
Refined Parts –	• The team considers this feature to become a core feature in the future to showcase models' respective refined parts

Associate Refined Part with a Model & Switch Between Refined Parts	• This feature must be explored further by adding more refined parts to the other existing models
Model Manipulation – Move, Rotate, Zoom (Scale)	• Implement maximum limit for how large a model can be scaled to avoid dropping frame rates and lag
Pull Apart Individual Parts	• Consider changing the colour when a part is highlighted by the ray interactor to avoid issues with green models • Add instruction pop-ups in the environment when individual parts mode is toggled to show how the ray interactors work • Consider implementing visible rays when pointing to a part • Conduct user testing for pulling apart individual parts when they are directly gabbed by a user
Toggling Various Features	• Prioritise user testing for the newly implemented menu UI for toggling features
Explode Model	• Utilise .fbx models with many individual parts as possible • Explore using hand gestures (similar to zooming in/out) to explode the model as an alternative to using a slider • Implement maximum boundary for how far the model can be exploded
Collaboration – Join Session, Synchronised Model Position & Interactions	• Prioritise user testing for the collaboration features • Explore how to effectively manage transferring a model's state authority through other interactions (other than touching a model) • Troubleshoot potential bugs that is causing a model to disappear when switching between models upon the application's initial bootup.
Pointer	• Explore Meta's Ray Interactors to enable visible rays • Explore how visible rays can become synchronised as a networked object so other users view the pointer as well
Cross Section (Baseline) – Single Cross Section and Move Cross Section	• Further research is required to implement this feature • Monitor future publishing of free cross section SDKs or sample scenes • Consider purchasing existing cross section SDKs in the Unity Asset Store; thorough research should be conducted if these would be compatible with mixed reality development before making a purchase
Cross Section – Different and Multiple Cross Section	• Ensure the baseline cross section features are implemented first before exploring these requirements • Further research and testing required to implement these concepts
Maintenance Manual	• Utilise Meta's existing scroll bar UI to implement this feature • Key considerations include how the digital copy of CAE's maintenance manuals (PDF files for instance) can be integrated into the environment
Model has Collision and Depth with the Environment	• Explore Unity's physics features to enable object collision with real-world object • Explore Meta's passthrough features to explore depth in the AR view • Further research may be required due to possible complexities with its implementation • These features should be the least prioritised as it is purely for aesthetic purposes

Other Considerations	• Adding more user-friendly features to guide users on how to utilise the product • Possible AI integration in the later stages of future development to streamline manual development whenever a new model is added to the application • Expertise with Unity game development may be required to implement more complex features • Involve actual end-users of the prototype (technicians and trainees) to gain practical feedback

Table 12 – Combined Future Development Considerations and Recommendations based on the insights gathered from Development Feasibility, Functionality Alignment and User Value assessments.

Based on the insights from this report, the team strongly recommends prioritising the completion of the Must-Have and Should-Have requirements that were not completed within this project lifecycle. Therefore, features such as Collaboration, Pointer and Cross-Section should be pursued in the next development phase. It is also strongly recommended to prioritise the user testing of the Collaboration features and the Grabbable Pop-Up Menu UI for toggling various features in the next project lifecycle. We also recommend prioritising enhancing the Synchronised Model Interactions feature through exploring effective ways to transfer state authority through other interactions and troubleshoot the issues that is causing models to disappear upon the application's bootup when switching between models. Additionally, certain Could-Have requirements, such as the Refined Part and Maintenance Manual features, could become increasingly important in the future and should be considered for future prioritisation.

In contrast, the *Switching Between Models* and *Model has Collision and Depth with the Environment* features may become lower priorities to further enhance and develop respectively. The table above also provides detailed alternatives and enhancements to some of the current functionalities, offering valuable options for refinement. We also stress the importance of meeting the suggested model requirements, including their file types and parts structures, as these factors play a crucial role in both the model's quality and the interactions with individual parts.

Overall, the goal of this project was to assess whether an AR Briefing Tool would be effective to assist in CAE's maintenance operations, briefings and trainings. Based on the team's current progress of the prototype and the insights gathered from development and user testing, the concept has proven feasible to develop and resulted in a positive user experience, functionality alignment and user value. Therefore, it is essential for CAE to consider the recommendations outlined in this report to further enhance the prototype's functionality and testing in future development.