CartOptics

Intuitive user interface with AR+Gen AI

Aaron Edward Hamilton: Major in SE

Elijah Esteban Munoz: Major in SE

Kyle Anthony Beck: Major in SE

Mason Thomas Vick: Major in SE

Course Instructor: Simon Fan

Faculty Advisor: Yongjie Zheng

Industry Sponsor: Qualcomm

Project Mentor: Karen Weeks

Project Proposer: Emma Lacey

Project Manager: Simon Fan

A capstone project report submitted to the faculty of

Computer Science and Information Systems

California State University, San Marcos

February 2025

(Version 5.0)

Technical Report Series: CSU-SM-CSIS-30-2025-SE-001-Team-005

1. Abstract

Traditional grocery shopping often lacks immediate, detailed product information, restricting consumers' ability to make fully informed purchasing decisions. Our system effectively addresses this issue by delivering near instant data through interactive augmented reality (AR). We have implemented a software solution using the Snapdragon AR Head-Mounted Display (HMD) designed to significantly enhance the grocery shopping experience.

Our software integrates advanced AI powered object detection and uses the Snapdragon HMD camera to identify grocery items within the user's visual field. Detected items are processed through the Google Gemini API, which instantly retrieves comprehensive product details. Key implemented features include real-time product highlighting, dynamic visual feedback overlays, and immediate access to critical metadata such as pricing, ingredients, and allergen information.

Our system helps consumers make healthier and more informed choices by providing detailed product information that would be hard to obtain otherwise. Users will be able to keep a list of their groceries/products while staying hands free and informed about what exactly is in their cart.

This comprehensive report details the identified business problem, design objectives, technical implementation leveraging Snapdragon Spaces and Unity. It also includes requirements tracking, test case summaries, and test results that show the system can reliably detect objects and give-AI powered details in real time using AR.

2. Report Revision History

Version 1.0 10/12/2024

- Report outline established.
- Abstract and problem statements were added.
- Requirements documented in system.
- Initial use case diagram and system architecture created.
- Lessons learned and conclusion formulated.

Version 2.0 11/25/2024

- Specific architectural designs have been added.
- System testing test suites and test cases have been established.
- Exploratory studies have been modified.
-

Version 2.5 12/09/2024

- Changes summarized in report revision history.
- Fixed grammatical errors in report
- Fixed inconsistencies in technical terms

Version 3.0 2/24/2025

- Updated terminology throughout report
- Added additional test suite
- Expanded test execution reports for 10 total
- UI screenshots are now shown for early model

Version 3.5 3/17/2025

- Added overview diagram
- Referenced figures and provided detailed explanations
- Corrected grammar mistakes

Version 4.0 4/28/2025

- Updated Overview diagram
- Updated UI screenshots
- Update Architectural Design diagram
- Added some new terminology

Version 5.0 5/6/2025

- Added extra detail to objectives section
- Improved testing portion to accurately reflect changes
- Updated grammar to reflect present and past tense rather than future tense since project is complete
- Added extra details to System testing
- Labeled System Testing screenshots

3. Problem Statement

3.1. Business Background

The goal of this project is to create a user-friendly interface combining generative AI and augmented reality (AR) to deliver real-time product information in retail environments. Using AR-enabled HMD, the system recognizes objects in a user's field of view and display relevant information related to the product. This is enhanced through intuitive user interactions using forms of user input. The solution provides seamless feedback to users and highlighting products, all while being privacy-conscious and maintaining user data protection. This cutting-edge system aims to elevate the in-store shopping experience by leveraging AI and AR to deliver real-time insights.

3.2. Needs

- Growing demand for quicker, more efficient access to detailed product information while shopping.
- Current solutions require manual searching, reading labels, or asking for assistance, which can be time-consuming and inefficient.
- Integration of AI and AR streamlines the shopping experience, enabling users to make more informed decisions faster.
- Rise in interest for personalized experiences and the need for transparency regarding product information tailored to specific markets.
- The system meets demand for immediate, personalized, and accurate product details.
- Focus on privacy-conscious object detection to ensure data security while enhancing the shopping experience. enhancing the shopping experience.

3.3. Objectives

- Develop a real-time AR+AI interface that enhances in-store shopping for health- and time-conscious customers.
- Leverage on-device AR with Snapdragon Spaces to instantly recognize products and display relevant details within the shopper's field of view.
- Use the Google Gemini API to provide real-time, personalized product information and respond to user queries through cloud-based AI.
- Design an intuitive interface that supports immersive virtual shopping through gestures and visual interaction.

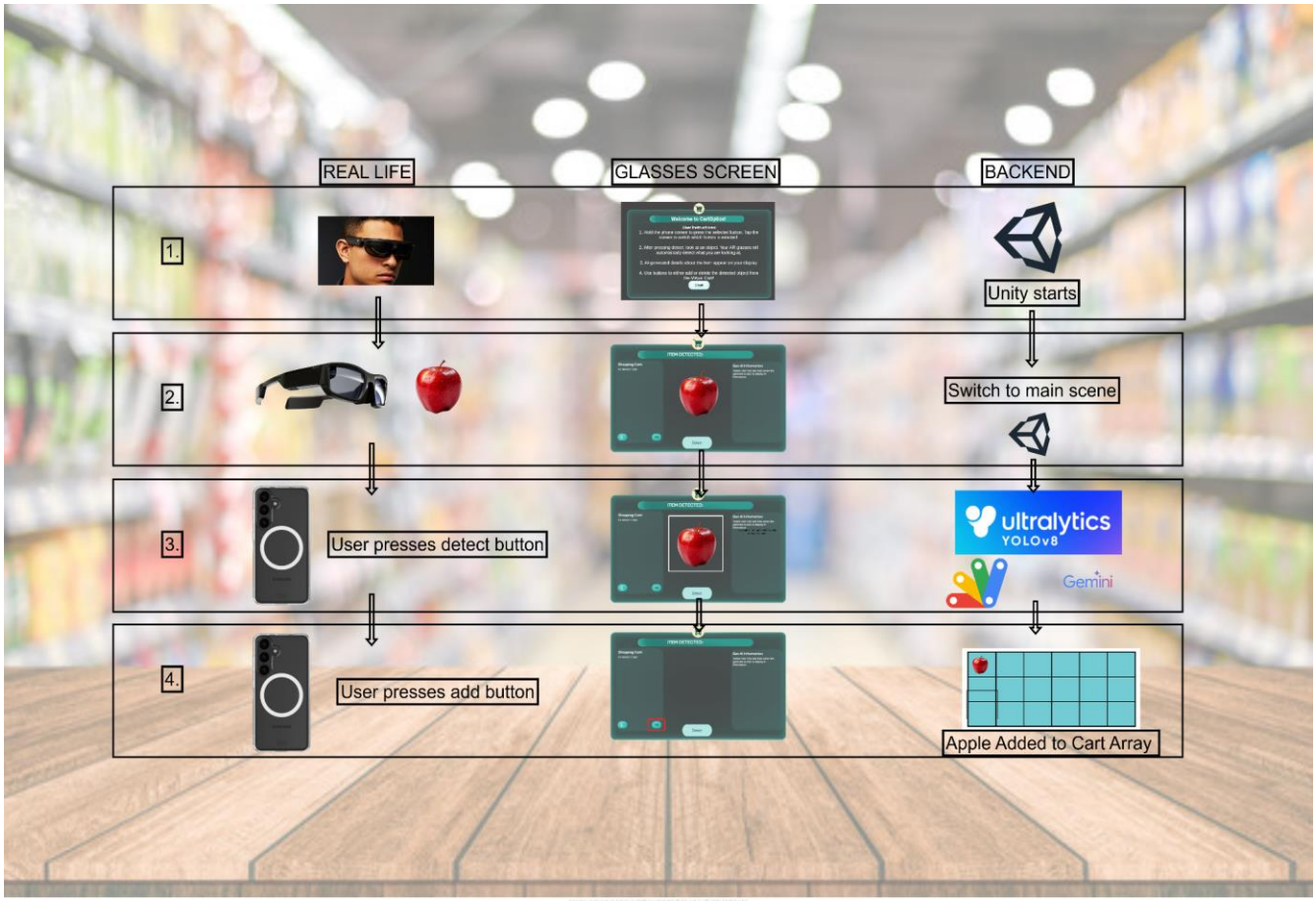


Figure 17.1 System Overview

This diagram showcases how the system functions in a typical use case scenario

4. Requirements

4.1. User Requirement

4.1.1. Glossary of Relevant Domain Terminology

AR (Augmented Reality):

Technology that overlays digital content onto the real world.

XR (Extended Reality):

An umbrella term for AR, VR, and MR, blending real and virtual environments.

ML (Machine Learning):

A subset of AI where systems learn from data to improve over time without explicit programming.

HMD (Head-Mounted Display):

A wearable device that provides immersive visual experiences, used in AR and VR.

Unity:
A cross-platform game engine widely used for creating 2D, 3D, augmented reality (AR), and virtual reality (VR) experiences. It's popular for its versatility and support for real-time interactive content development.

Snapdragon Spaces:
An augmented reality (AR) development platform from Qualcomm that provides tools and frameworks for building immersive AR experiences on Snapdragon-powered devices.

YOLO: Object detection neural network model. The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. (We used YOLOv8n)

Dual Render Fusion: A feature within the Snapdragon Spaces platform that allows developers to simultaneously render a single application on both a mobile phone screen and a connected augmented reality (AR) headset.

API (Application Programming Interface): It is a type of software interface , offering a service to other pieces of software.

Google Gemini: A family of multimodal AI models developed by Google DeepMind, serving as the successor to LaMDA and PaLM 2.

4.1.2. User Groups

Our target audience includes customers who are focused on enhancing their shopping experience by making it more effective and efficient. These individuals value convenience, time-saving features, and streamlined processes that allow them to complete their shopping with minimal hassle. They seek tools or platforms that simplify decision-making, improve navigation, and provide quick access to the products or services they need. By catering to these customers, we aim to deliver a solution that meets their demand for a smoother, more organized, and productive shopping journey.

4.1.3. Functional Requirements

4.1.3.1. Project Scope [Use Case Diagram]

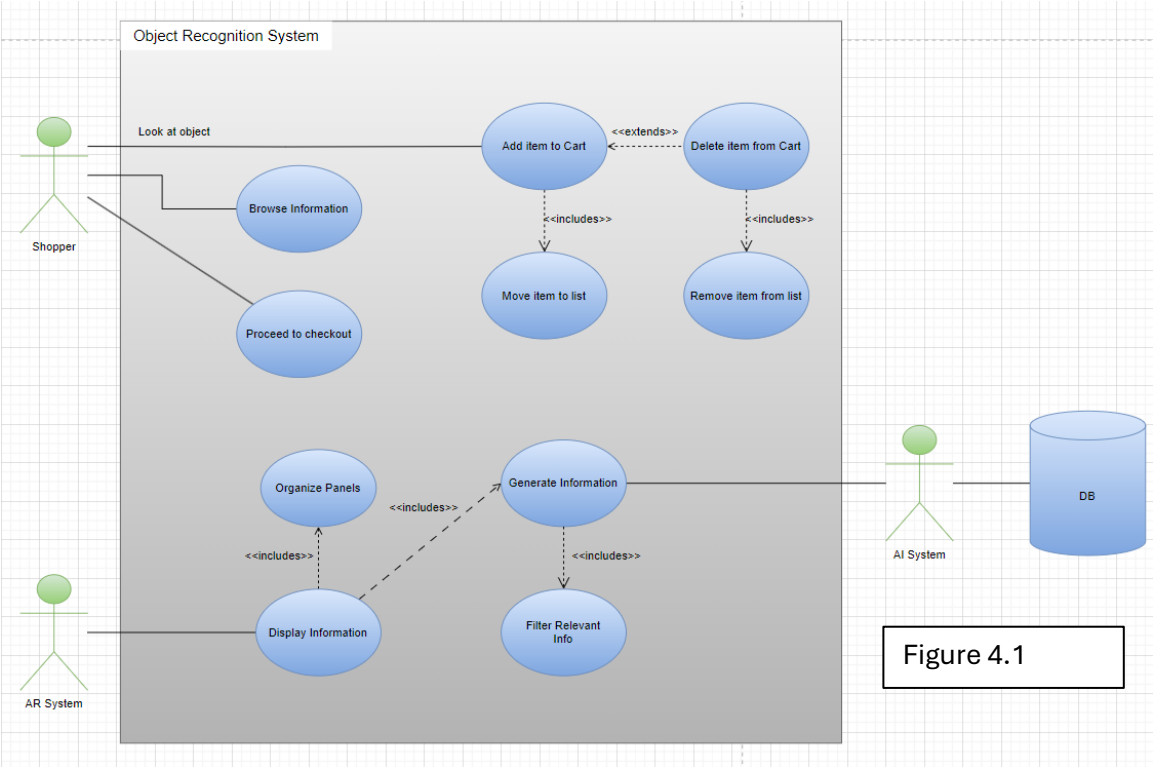


Figure 4.1 Use Case Diagram

Diagram 4.1 demonstrates the use-cases for both the human user and the AR system and how it functions. The AI System is responsible for generating the related details of the product. The shopper can browse information about the detected object, and the AR System displays said information to the user once an object is detected. -

4.1.3.2. User Scenarios

As a shopper, I want to be shown information about the product I am looking at and interested in buying, so that I can be more informed when making a purchase.

As a shopper, I want the information displayed to me to be relevant to what matters to me personally in a product, so that the information is valuable and important to whether I purchase the product.

4.1.3.3. User Functional Requirements

- **UF-A**: Information generated about the object recognized by the HMD should be displayed to the user and easy to view.
- **UF-B**: The HMD must be capable of recognizing the object that the user I currently observing.

4.1.4. Non-Functional Requirements

4.1.4.1. Product: Usability Requirements

- **UP-01**: The user interface should be simple and easy to navigate.
- **UP-02**: Visual elements must be engineered to not obstruct the view and must not become cluttered.

4.1.4.2. Product: Performance Requirements

- **UP-03:** The HMD should respond quickly to user interaction.

4.1.4.3. Product: Availability/Reliability/Security

- **UF-C** : The User data should not be lost in the middle of the session

4.1.4.7. External: Legislative Requirements

- **UE-01:** System does not sell the user's data or perform functions without the user's consent.

4.2. System Requirements

4.2.1. Functional Requirements

4.2.1.1. System Functional Requirements.

- **SF-A-01**: The system should generate information about the recognized object using the LLM.
- **SF-B-01**: The system must be able to recognize objects in the FOV of the cameras using existing libraries compatible with unity and snapdragon spaces.

4.2.2. Non-Functional Requirements

4.2.2.1. Product: Usability Requirements

- **SP-01-01** The system minimizes the number of UI elements used to enhance clarity and usability.
- **SP-02-01:** Visual elements have a timer implemented to close after a set amount of time to avoid cluttering.
- **SP-02-02:** Visual elements must either close or reuse existing visual elements to display new information to avoid cluttering of the view

4.2.2.2. Product: Performance Requirements

- **SP-03-01:** The system minimizes background processes as well and prioritize time and space complexity to ensure a quick response time.

4.2.2.3. Product: Availability/Reliability/Security

- **SF-C-01:** The system should automatically save the user's session as they shop, ensuring that items added to their cart or any relevant data

4.2.2.7. External: Legislative Requirements

- **SE-01-01 :**The system must comply with GDPR Article 5 or CCPA regulations, ensuring that all personal data (e.g., user interactions, shopping habits, and preferences) collected by the AR HMD is not saved without permission

4.3. Requirements Trace Table

Project Name: Intuitive user interface with AR+Gen AI			
User Requirements		System Requirements	
Req ID	Description	Req ID	Description
UE-01	System will not sell the user's data or perform functions without the user's consent.	SE-01-01	The system must comply with GDPR Article 5 or CCPA regulations, ensuring that all personal data (e.g., user interactions, shopping habits, and preferences) collected by the AR glasses is not saved without permission.
UF-A	Information generated about the object recognized by the HMD should be displayed to the user and easy to view.	SF-A-01	The system should generate information about the recognized object using the LLM
UF-B	The HMD must be capable of recognizing the object that the user is currently observing	SF-B-01	The system must be able to recognize objects in the FOV of the cameras using existing libraries compatible with unity and snapdragon spaces.
UF-C	The User data should not be lost in the middle of the session	SF-C-01	The system should automatically save the user's session as they shop, ensuring that items added to their cart or any relevant data
UP-01	The user interface should be simple and easy to navigate.	SP-01-01	The system will minimize the number of UI elements used to enhance clarity and usability
UP-02	Visual elements must be engineered to not obstruct the view and must not become cluttered	SP-02-01	Visual elements will have a timer implemented to close after a set amount of time to avoid cluttering.
		SP-02-02	Visual elements must either close or reuse existing visual elements to display new information to avoid cluttering of the view
UP-03	The HMD should respond quickly to user interaction.	SP-03-01	The system will minimize background processes as well and prioritize time and space complexity to ensure a quick response time.
Acknowledgment: Generated from the CapStone process management system ©2024			

Figure 4.3 Requirements Trace Table

The Requirements Trace Table below provides a clear mapping between the User Requirements and the corresponding System Requirements for the "CartOptics" project. Each user need or functional expectation is identified by a unique ID and description. It is when linked to one or more system-level requirements that define how the software or hardware should be designed to meet those user needs.

The traceability ensures that all users expectations are addressed by technical specifications and provides a reference to verification that the implementation aligns with the stakeholder expectations. For example, the user has privacy concerns are traced to the system behaviors and performance requirements like responsiveness and clarity are supported

through specific UI and processing optimizations. We structured mapping which helps maintain alignment across the development testing and validation phases while reducing the risk of missing critical and functionality.

5. Exploratory Studies

5.1. Relevant Development Frameworks

5.1.1 Open XR

- We utilized OpenXR, provided by the Snapdragon Spaces development kit in this project. This resource helped us better comprehend how to implement spatial awareness and contextual interactions within the project.

5.1.2 Snapdragon spaces

- We utilized the Snapdragon Spaces development kit, provided by Qualcomm. This is the Qualcomm AR system that we are using for AR development. It is also installed on the Android device, so that we can access the AR application on the Lenovo HMD.

5.1.3 YOLO Neural Network Model

- We used the YOLOv8 Model to detect objects in real time as the XR camera picks them up.

5.2 Relevant Development Frameworks

For this project, Unity was selected as the primary development framework due to its robust capabilities for building real-time 3D applications and its versatility in handling various types of immersive experiences, particularly Augmented Reality (AR). Unity's extensive library of assets, coupled with its compatibility with AR and AI technologies, makes it an ideal choice for integrating both AR and Generative AI features seamlessly. To enhance our understanding of Unity's capabilities in AR, we utilized the Snapdragon Spaces development kit, provided by Qualcomm. This is the Qualcomm AR system that we are using for AR development. It is also installed on the Android device, so that we can access the AR application on the Lenovo HMD.

5.3. Relevant Solution Techniques

Several solution techniques were employed to address the challenges faced in building a system that integrates AR with Generative AI. One of the primary techniques involved the implementation of object detection using YOLO Neural Network Model, which allowed us to identify and classify objects in real time. This was essential in providing relevant information about products on a store shelf based on user queries. We also utilized Unity's AR Foundation to create an interactive user interface that supports AR visualization. To improve our AI capabilities,

We leveraged the Google Gemini API. This allows us the capability of generating real time information about the detected object. Through Snapdragon Spaces, we incorporated spatial tracking and contextual product information, which enhanced the overall user experience by providing immediate insights into the items users interacted with. By utilizing cloud AI, we ensured that more complex data, such as Generative AI responses,

could be processed efficiently while maintaining privacy through on-device processing where necessary.

5.4. Broader Impacts

The broader impacts of this project extend beyond simply enhancing user experience in an AR environment. By integrating privacy-conscious object detection using Barracuda and YOLO, we ensure that users' interactions with products are handled securely, preserving personal data and fostering trust. This focus on privacy aligns with broader societal concerns regarding data protection in AI-driven systems. Additionally, the combination of AR and Generative AI creates a more intuitive and engaging shopping experience. The real-time information provided to users not only improves accessibility to product details but also has potential applications in education, healthcare, and retail sectors, where on-the-go, contextual information is increasingly valuable. The techniques and frameworks utilized in this project, including Google Gemini API, Unity, and Snapdragon Spaces, could serve as a foundation for future innovations in AR-driven user interfaces and personalized, AI-enhanced services.

6. System Design

6.1. Architectural Design

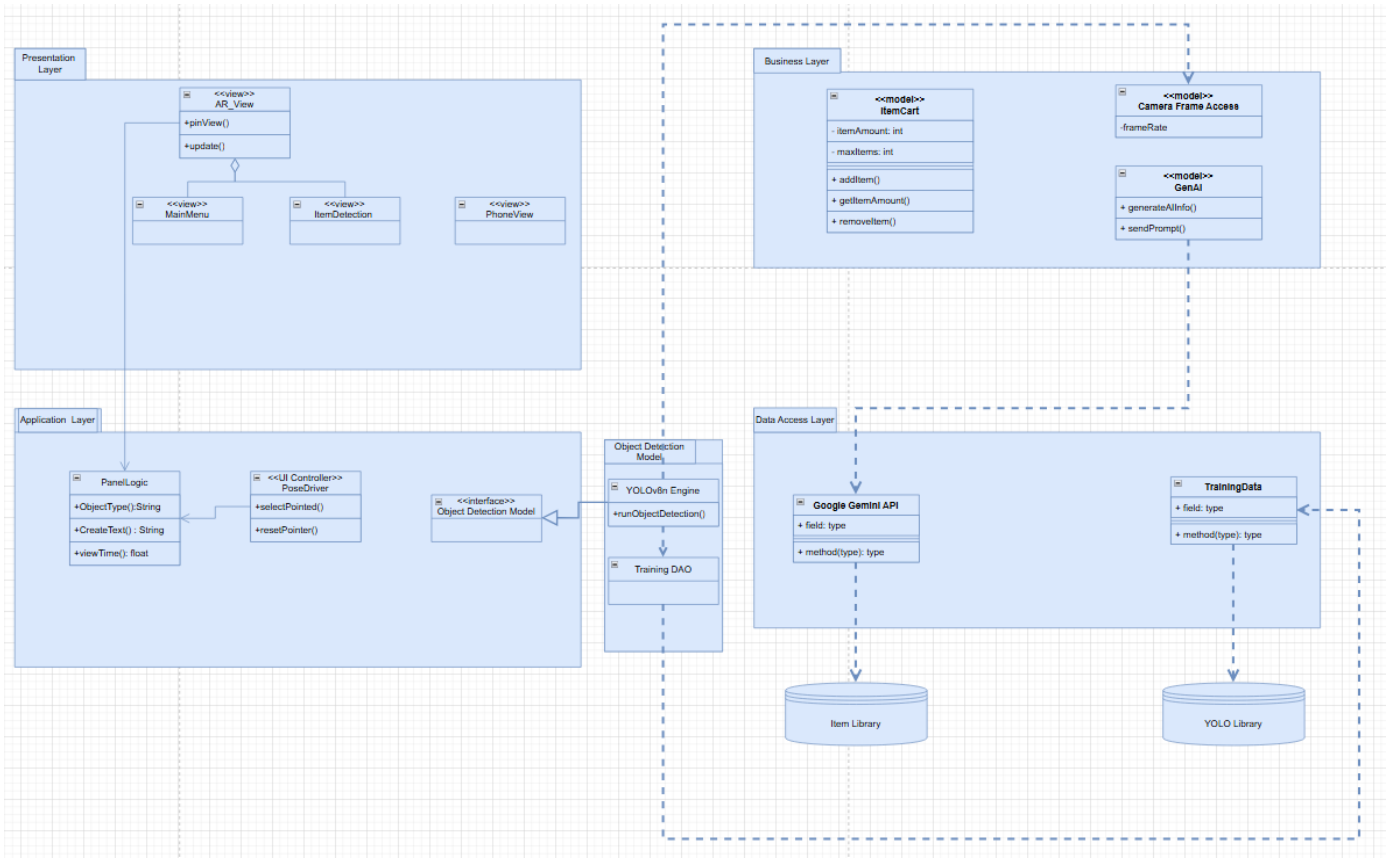
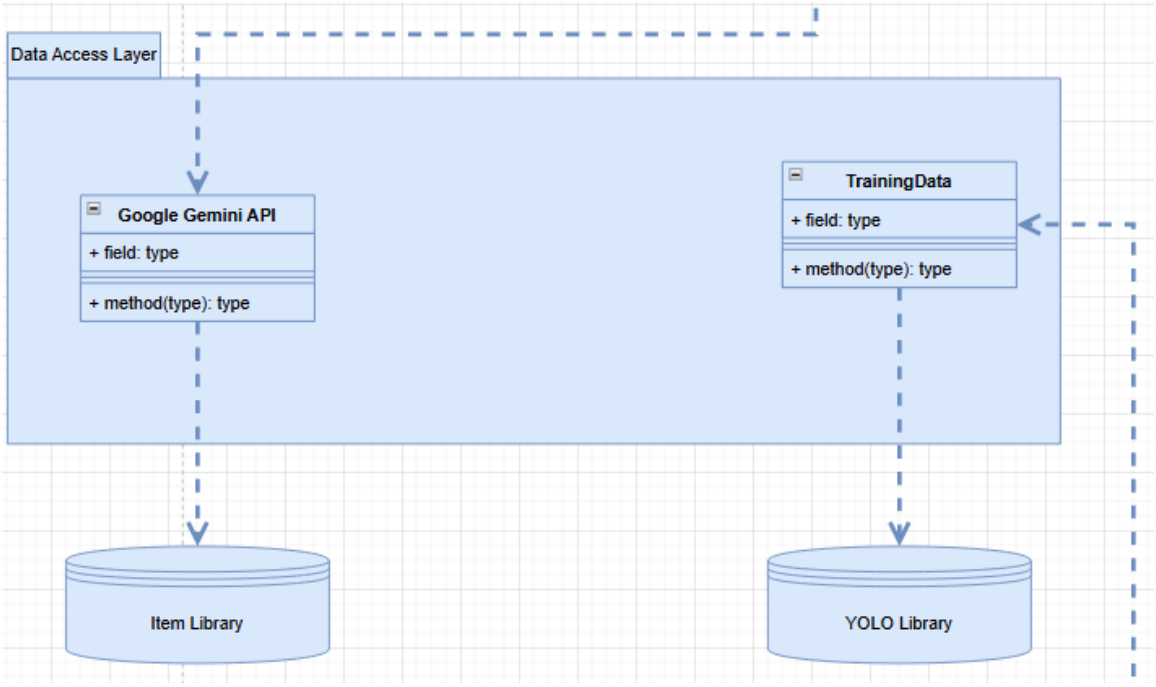


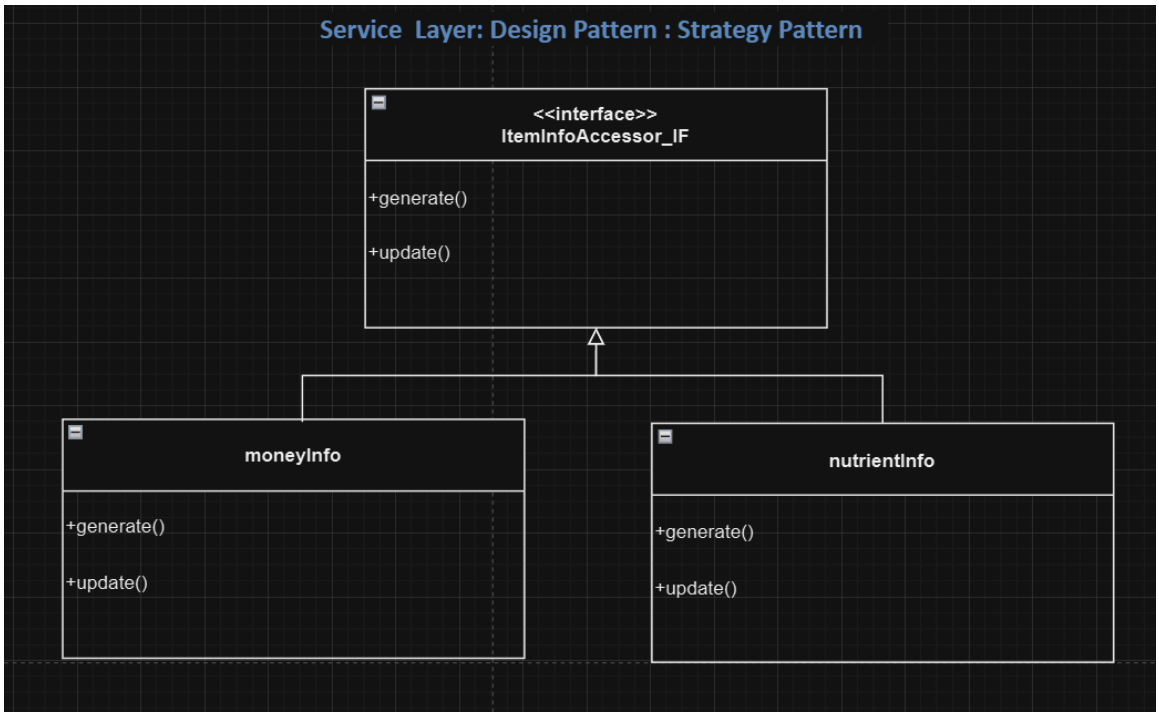
Figure 6.1 Architectural Design

For the design of our system, we decided to use a model view controller design. We took clean architecture into consideration when designing this architectural diagram. The diagram is structured in a counter clockwise orientation as the arrows direct.

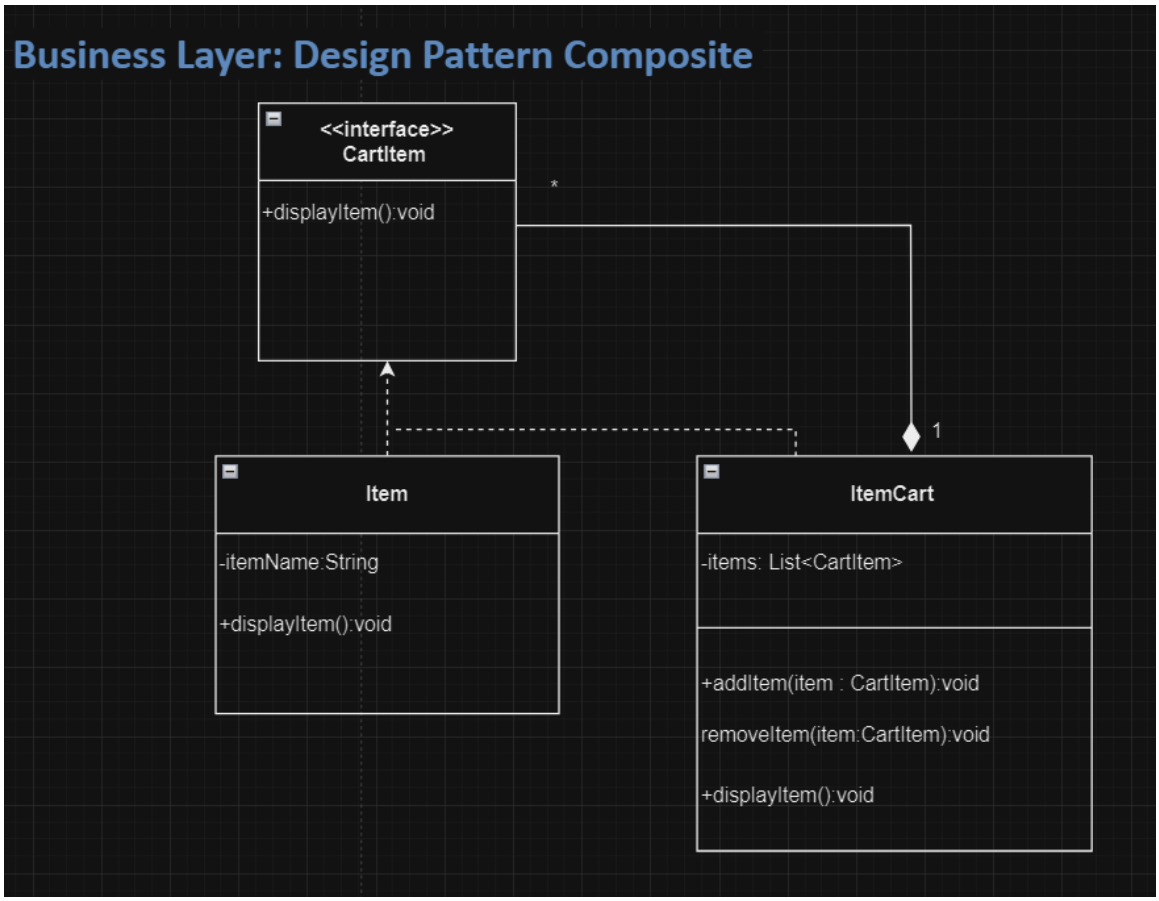
6.2. Structural Design



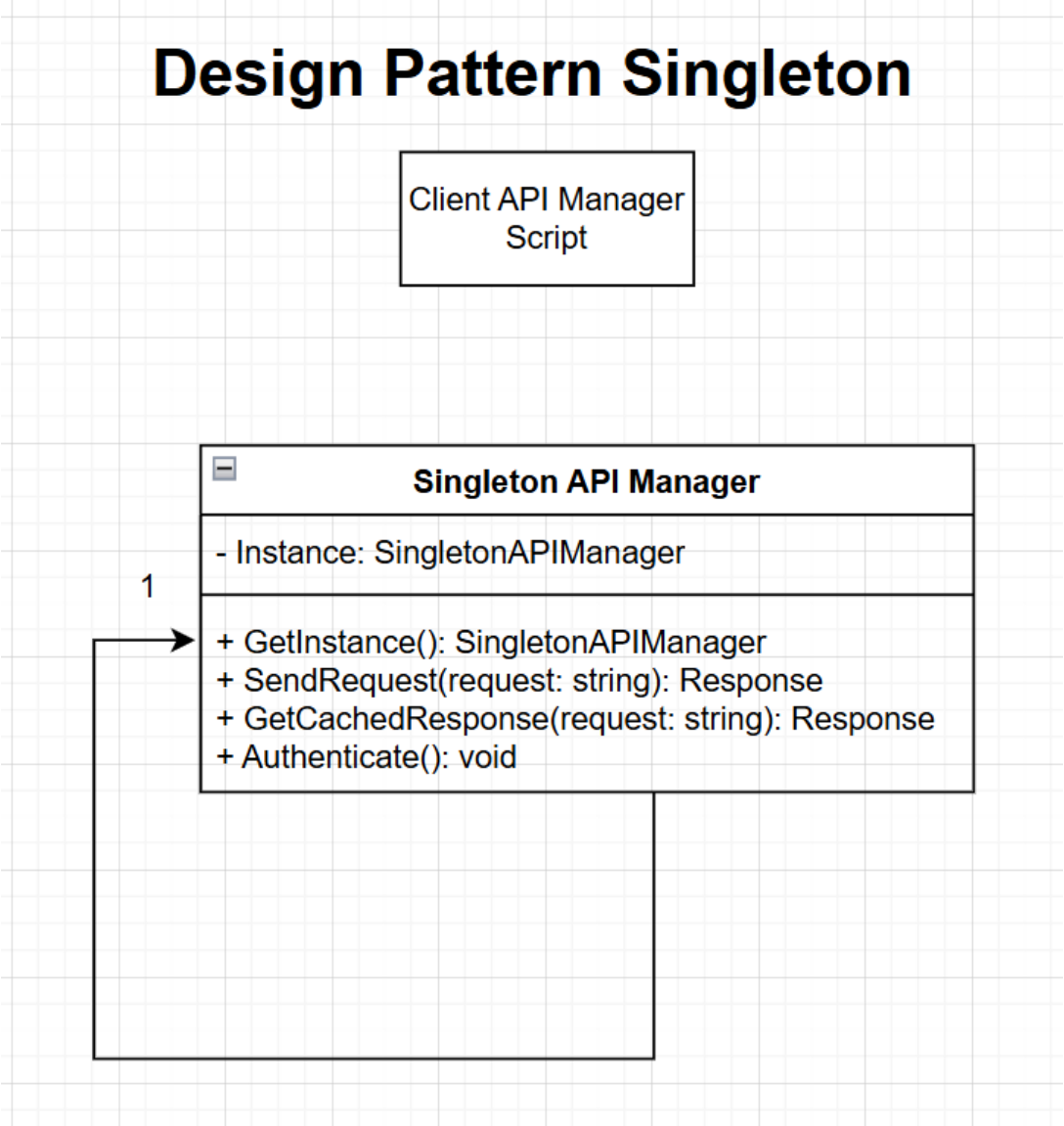
We separated the database into its own layer.



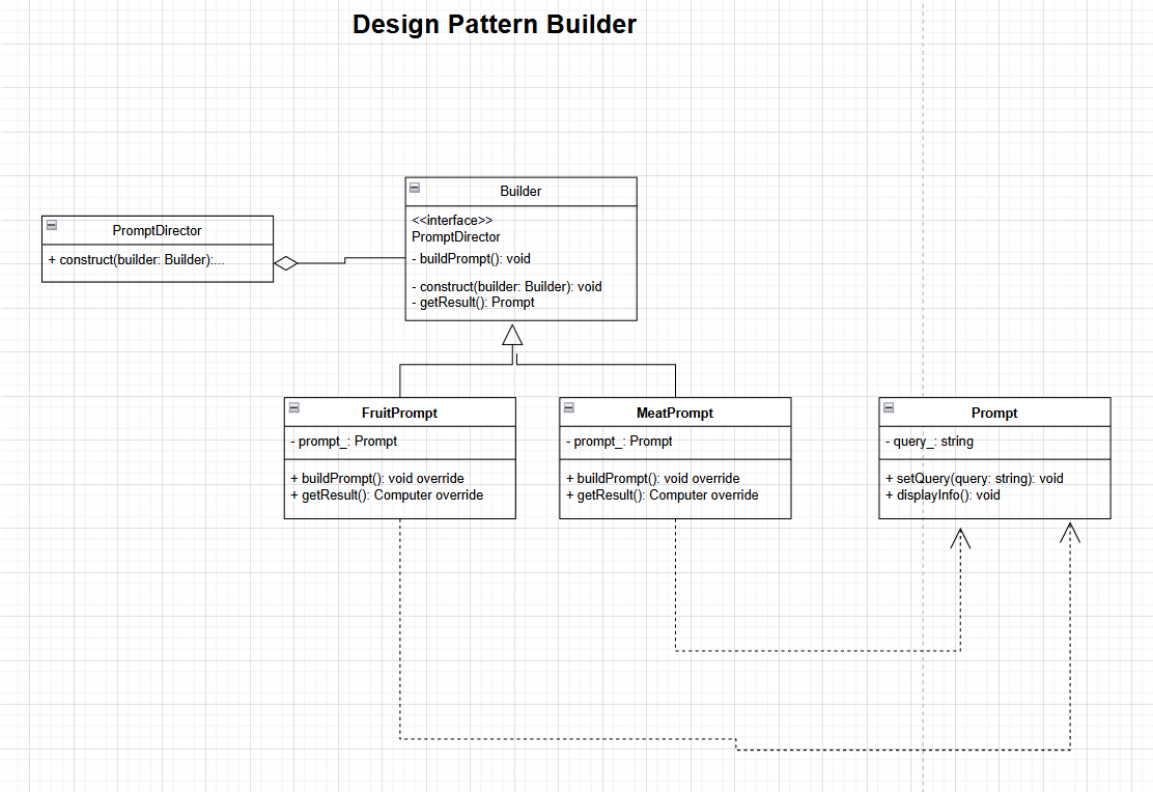
The ItemInfoAccessor_IF interface features a generate and an update function. This information is about the detected object. There are different variants to this info that focus more on money and pricing, while the other focuses more on the nutritional value of the object. Each has a variant version of the generate() and update() functions. To do this, we utilize the Strategy pattern.



We decided to use a composite design for the item cart to show the 1 to many relationship between the ItemCart and the items. We used an interface to represent that the ItemCart is also a CartItem.



We used the Singleton Design pattern for our API manager component of the project. This revolves around the calling of the Google Gemini API for our generative AI to function properly. Since this request and API Manager object is very important and should only have one instance in the entire project, then we can utilize singleton to ensure there is only one instance and the important logic within is not called multiple times.



We decided to use the builder design pattern for constructing specific prompt within the generative AI portion of our project. We first understood that we wanted to seprate the different objects we can recognize into categories, for example: oranges and apples go under the fruit category, while there are also other categories such as dairy or meat products. The builder, based on the category of the detected object, feeds a different prompt that is specific to the category to the Google Gemini API. So, for fruits it can showcase the specific vitamins involved, dairy products can involve an expiration date, etc.

6.3. User Interface Design

For the user interface design, we have the start screen, displaying the title and start button. Then we transition to the main screen which illustrates the AI slider. The AI slider displays the information about the item nutrition and cost and other important information about the item. The item list comes with a slider would display objects that have been previously looked at by the customer. Now when the shopper wants to add something to the cart, they can click on the shopping cart button then it takes them to a list of items previously looked at and can simply checkout.

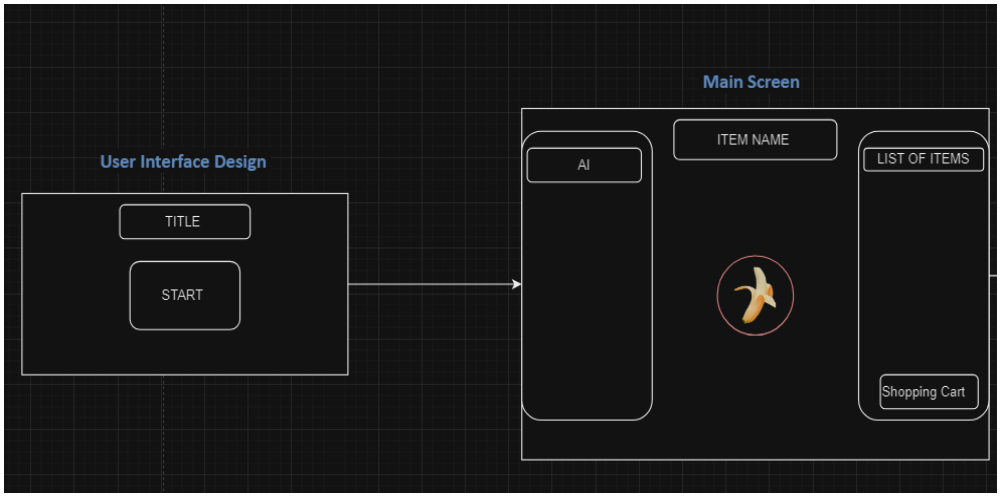


Figure 6.3 AR UI Prototype

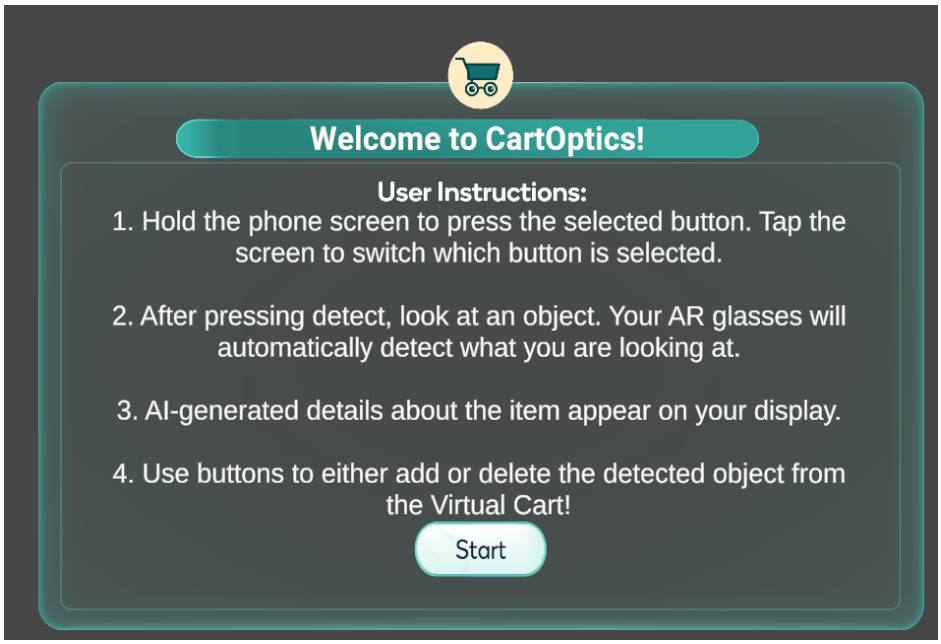
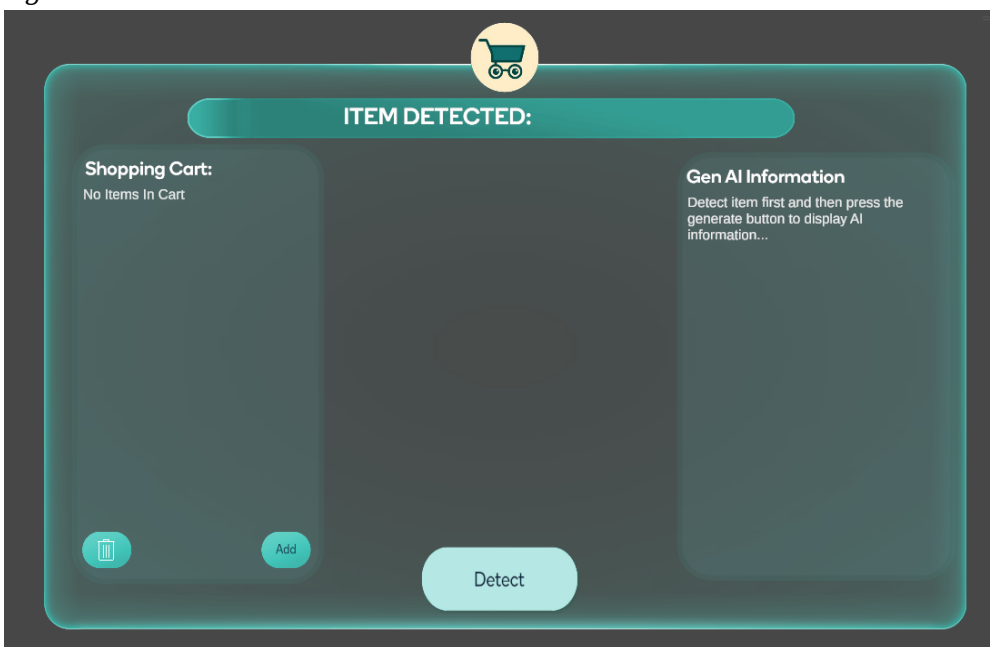
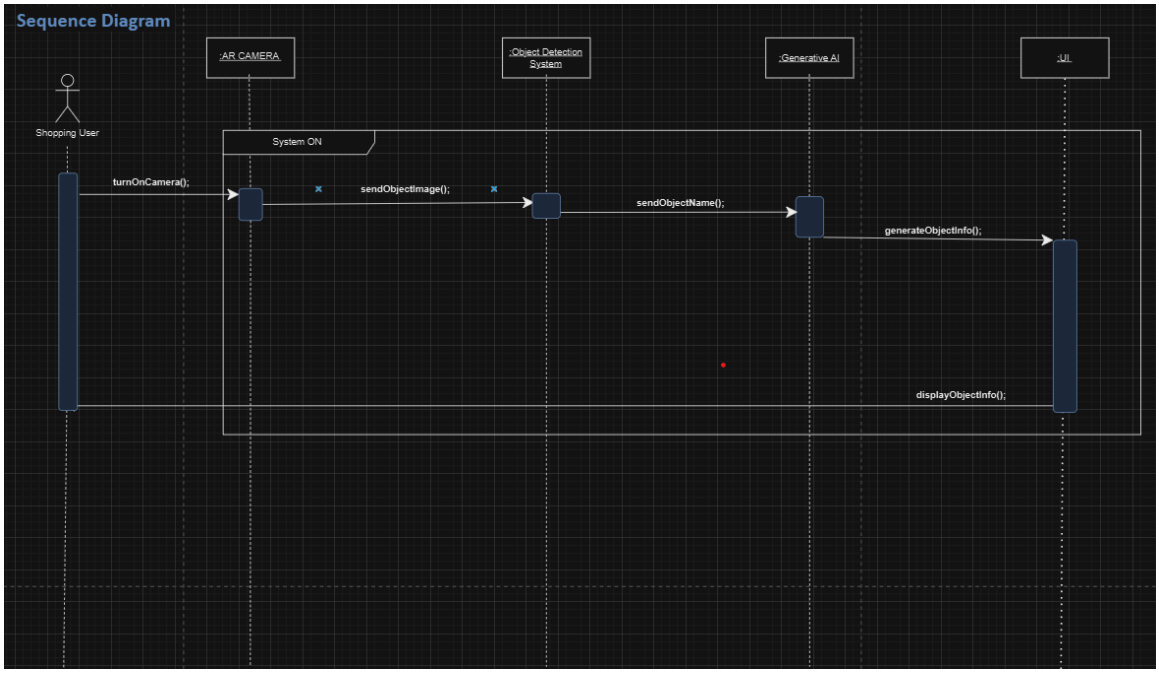


Figure 6.3 AR UI



6.4. Behavioral Design



The sequence diagram depicts how the user interacts with the system. First it is up to the user to start up the system and turn on the camera. This brings us to the main loop, where the AR camera when detecting an actual item sends the item image to the object detection system. Through this system, the object name is parsed and fed into our generative AI model in which detailed information about the object is displayed to the user via the user interface.

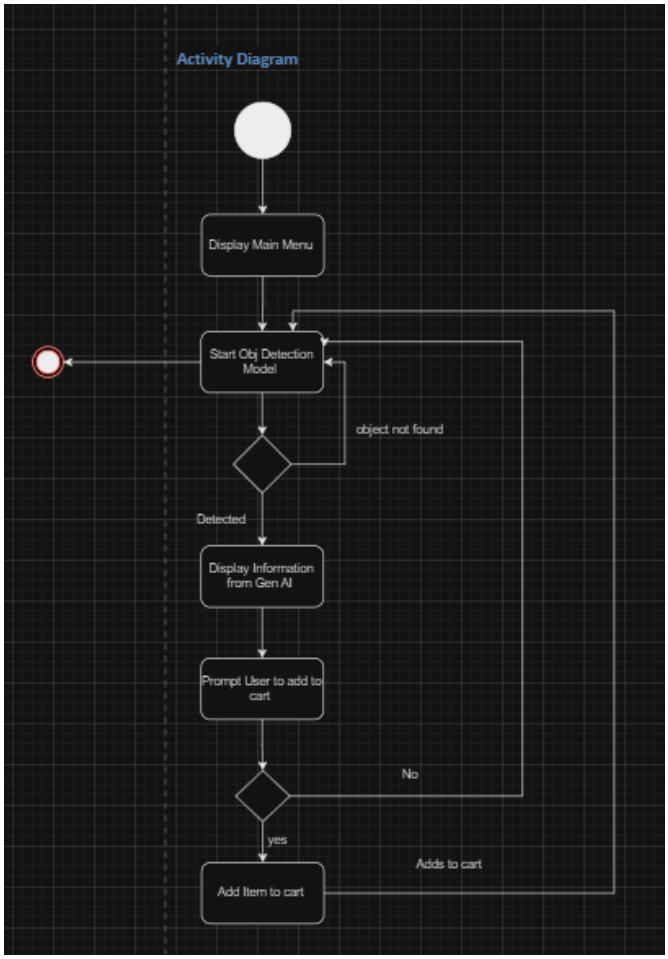
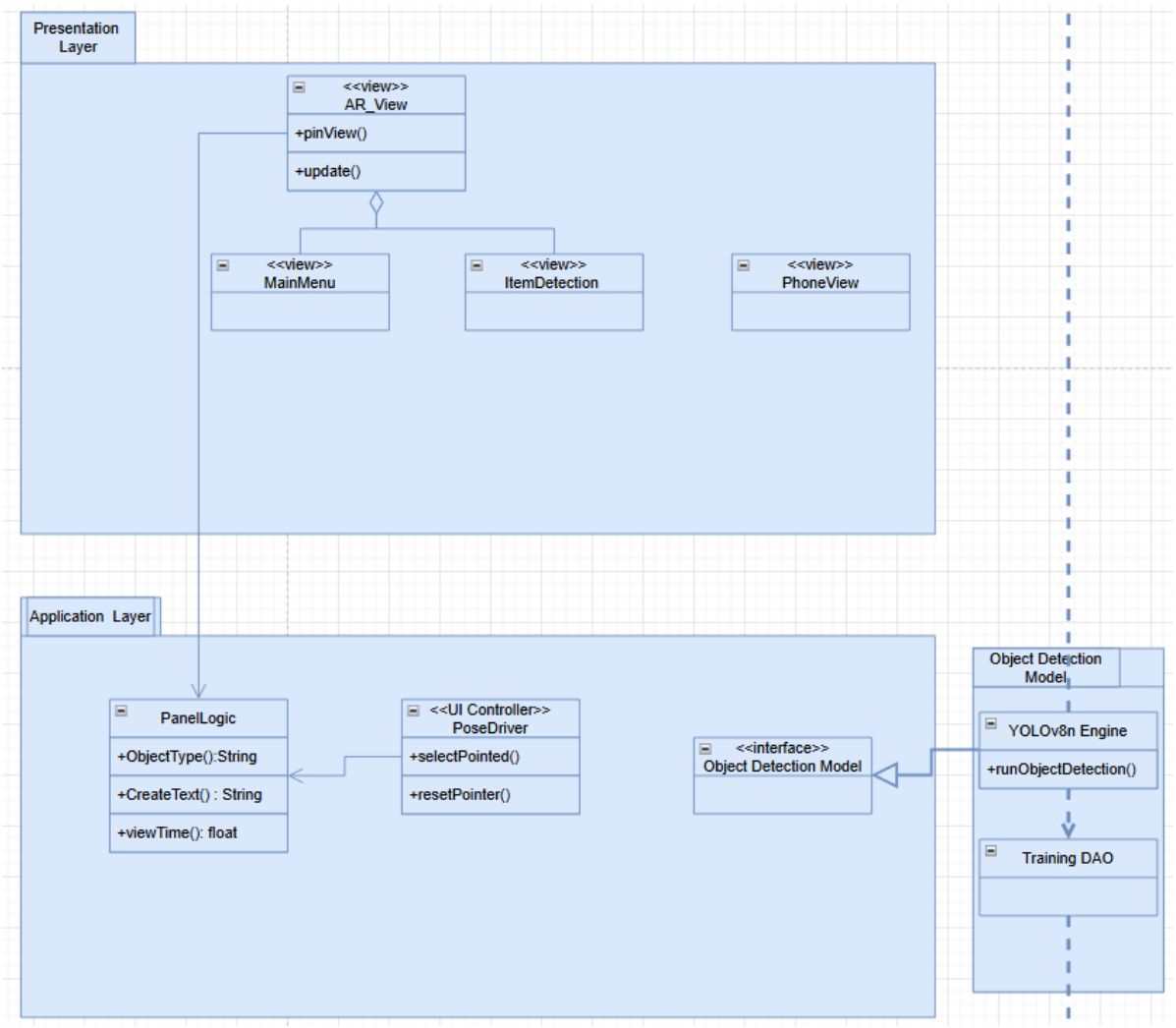


Figure 6.4 Sequence Diagram

On the activity diagram, the first scene that appears to the user within the Lenovo HMD is the main menu screen. From there, after the user selects start, they are brought to the main object detection model scene, where the camera sends images of objects the user looks at to the object detection system. If there is no object detected, then this process repeats until an actual object is detected. After the object is successfully detected, the AI should generate specific information about the detected object. The user is then prompted if they want to add the object to the cart. If they choose not, then they return to the main object detection scene. While saying yes, you then add the item to the cart and then still return to the object detection scene. The user may exit the application from here at any time.



6.5. Design Alternatives & Decision Rationale

In terms of design alternatives, there were many different concepts that were considered during the conception of the UI. The first decision that needs to be made was which UI elements would be world-level in comparison with screen-level. The former involves interactable objects placed in the world via Augmented Reality and the latter is attached to the user's field of view no matter what direction the user looks towards. Some of the alternatives included having the item description on the world-level, however that option was quickly ruled out knowing that if the user was mobile, then the description quickly falls out of the user's field of view. This is an integral factor to consider since the system is meant to be applied in a real-time shopping environment, so the user is likely mobile the majority of the time.

That being said, we then settled to the current UI design where the relevant information about a product is displayed for the user at the screen-level, making the design simpler and easier to access at all times.

7. System Implementation

7.1. Programming Languages & Tools

We are developing the entire application in Unity using C#. To ensure compatibility across devices, we are leveraging Snapdragon Spaces in Unity, enabling us to test the app on both smartphones and AR HMD.

7.2. Coding Conventions

We make sure we consider using good naming conventions, good folder structure in Unity (e.g. Scripts, Prefabs, Animations, etc.), proper commenting and documentation, and other common coding conventions that help us be more organized and productive.

7.3. Code Version Control

We use Git for tracking changes, collaborating with team members, and maintaining the history of the project.

7.4. Implementation Alternatives & Decision Rationale

We originally planned to utilize Niantic Lightship for object detection, but as we attempted to build our project onto the associated android phone, we realized that there was a compatibility issue between Lightship and Snapdragon Spaces OpenXR, which is a mandatory requirement to utilize within our project. After realizing this, we pivoted our implementation to utilizing YOLOv8, a separate object detection API that happens to be compatible with OpenXR. This solidified our decision to move forward using YOLOv8 for object detection of our system.

8. System Testing

8.1. Test Automation Framework

NUnit: This tool is used to run unit tests on our C# scripts.

Unity test framework: Unit testing for AR portion of system.

The goal of these tests was to evaluate the integration of software and hardware components in a real-use scenario. For example, we validated that the HMD camera correctly streams video to the application, prompts are accurately processed by the Gemini API, and the interface responds smoothly to user interaction. Expected outcomes were defined in advance, and the system's responses were compared against them to confirm functionality.

8.2. Test Case Design

Test Case TC-003 – Start Button Transitions to Object Detection Scene

Table 8.2.4. Test Case TC-003

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-001: UI_TESTS	
Test Case ID	TC-003 (Unit Test)	
What To Test	Start button transitions to Object Detection scene.	
Test Data Input	User clicks Start	
Expected Result	Scene should transition to Object Detection scene.	
Traceability	Relevant User Req.(s)	UP-01,UP-03
	Relevant System Req.(s)	SP-01-01
	Relevant Use Case(s)	

Acknowledgment: Generated from the CapStone process management system ©2025

This test verifies that the system responds appropriately when the user clicks the "Start" button on the main menu. The test ensures that the interface transitions to the Object Detection scene, confirming that the navigation logic and UI responsiveness are functioning as intended. This is critical for a smooth user experience and aligns with the simplicity and usability expectations (UP-01, UP-03).

Test Case TC-007 – Prompt Sent to Google Gemini API

Table 8.2.5. Test Case TC-007

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-003: GenAI_TESTS	
Test Case ID	TC-007 (Unit Test)	
What To Test	Prompt goes through via request to Google Gemini API	
Test Data Input	Prompt, requestURL	
Expected Result	True	
Traceability	Relevant User Req.(s)	UF-A
	Relevant System Req.(s)	
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2025		

This unit test ensures that the application correctly forwards a user's prompt and request URL to the Google Gemini API. It validates the prompt handling function and confirms that the system can initiate an external API call based on user input. A successful result means the GenAI pipeline is connected and correctly accepting input (UF-A, UC-001).

Test Case TC-008 – Invalid Object Should Not Generate Anything

Table 8.2.6. Test Case TC-008

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-003: GenAI_TESTS	
Test Case ID	TC-008 (Unit Test)	
What To Test	Unapplicable Object passed for Generative AI	
Test Data Input	Object label	
Expected Result	Fail, should not generate anything	
Traceability	Relevant User Req.(s)	
	Relevant System Req.(s)	
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2025		

This test checks whether the system properly rejects an unrecognized object label sent to the generative AI component. If the input does not match any supported categories, the system should fail gracefully and not generate an output. This prevents irrelevant or misleading responses and ensures input validity checks are in place.

Test Case TC-009 – API Call Limit Handling

Table 8.2.7. Test Case TC-009

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-003: GenAI_TESTS	
Test Case ID	TC-009 (Unit Test)	
What To Test	Too many API calls in a short period of time	
Test Data Input	integer representing number of API calls	
Expected Result	Fail, should not exceed set limit of API calls in minute timeframe	
Traceability	Relevant User Req.(s)	
	Relevant System Req.(s)	
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2025		

This test simulates too many API requests within a short period. The system is expected to reject excess calls, ensuring adherence to rate-limiting constraints and avoiding server overload or bans. This helps maintain system performance and aligns with best practices in API usage.

Test Case TC-010 – Response Matches Detected Object from Gemini API

Table 8.2.8. Test Case TC-010

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-003: GenAI_TESTS	
Test Case ID	TC-010 (Unit Test)	
What To Test	Response should be relevant to the detected object	
Test Data Input	Generated AI Response from Google Gemini API	
Expected Result	Pass	
Traceability	Relevant User Req.(s)	
	Relevant System Req.(s)	
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2025		

This test validates that the response generated by the Gemini API corresponds directly to the detected object. It ensures that the system understands context and produces meaningful, object-specific information. This is key for delivering a useful and intelligent AR experience.

Test Case TC-004 – HMD Connects to Phone Application

Table 8.2.9. Test Case TC-004

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-002: HMD_Interactions	
Test Case ID	TC-004 (Integration Test)	
What To Test	HMD connects to application on phone	
Test Data Input	Application launched while HMD is connected to phone.	
Expected Result	Application log shows headset is connected.	
Traceability	Relevant User Req.(s)	UF-B,UP-01,UP-03
	Relevant System Req.(s)	
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2025		

This integration test ensures that the HMD correctly connects to the mobile application upon launch. It validates that the system can detect the headset and enable its features, which is fundamental for initializing the user session and supporting HMD-dependent interactions.

Test Case TC-005 – Camera Connection to Unity Feed

Table 8.2.10. Test Case TC-005

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-002: HMD_Interactions	
Test Case ID	TC-005 (Integration Test)	
What To Test	HMD camera is connected	
Test Data Input	Camera footage	
Expected Result	Video passed to Unity	
Traceability	Relevant User Req.(s)	UF-B
	Relevant System Req.(s)	SF-A-01
	Relevant Use Case(s)	UC-001
Acknowledgment: Generated from the CapStone process management system ©2025		

This test confirms that the HMD's camera is active and successfully streams video into the Unity environment. It validates camera setup and Unity video pipeline integration, which is essential for object detection and user feedback within the AR system.

Test Case TC-001 – Main Menu Loads at Startup

Table 8.2.11. Test Case TC-001

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-001: UI_TESTS	
Test Case ID	TC-001 (System Test)	
What To Test	Main menu loads on start	
Test Data Input	User launches application	
Expected Result	Main menu scene appears	
Traceability	Relevant User Req.(s)	UP-01
	Relevant System Req.(s)	
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2025		

This system test confirms that the main menu automatically loads after the application is launched. It validates the initial boot sequence and UI rendering, ensuring users are presented with an operational interface without delay.

Test Case TC-002 – Dual Render Fusion of HMD and Phone Display

Table 8.2.12. Test Case TC-002

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-001: UI_TESTS	
Test Case ID	TC-002 (System Test)	
What To Test	Dual Render Fusion: Separate phone and HMD displays	
Test Data Input	User launches application	
Expected Result	HMD and Phone render separate views.	
Traceability	Relevant User Req.(s)	UP-02
	Relevant System Req.(s)	SP-01-01
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2025		

This test checks the functionality of dual rendering—verifying that the HMD and phone screen both render their own views without interference. This is vital for applications where both users and bystanders may need access to different but simultaneous views.

Test Case TC-006 – Real-Time HMD Camera Feed Displayed on Screen

Table 8.2.13. Test Case TC-006

Project Name:	Intuitive user interface with AR+Gen AI	
Test Suite	TS-002: HMD_Interactions	
Test Case ID	TC-006 (System Test)	
What To Test	Camera frame access works	
Test Data Input	HMD Camera Video Feed	
Expected Result	Video feed will display on screen in real time	
Traceability	Relevant User Req.(s)	UF-A,UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-001
Acknowledgment: Generated from the CapStone process management system ©2025		

This system test verifies that the HMD camera feed displays in real time on the screen. It ensures low-latency video processing and confirms the integration between hardware (HMD) and software (Unity AR interface) for a responsive user experience.

8.3. Test Case Execution Report

Table 8.3.1 – Execution Report for Test Case TC-003

Table 8.3.1. Execution Report of Test Case TC-003

Project Name:	Intuitive user interface with AR+Gen AI					
Test Case ID:	TC-003					
Testing Tools Used:	Manual Test					
Testing Type:	Functional testing					
Execution Steps:	1	Connect HMD to Phone				
	2	Launch application				
	3	Hit "Next" on the main menu scene				
	4	Check if scene is transitioned				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Mason	01/29/2025	Scene is transitioned, but the content is not displayed in new scene	Fail	AR Session object was destroyed in transition	02/09/2025 by Aaron
2	Aaron	02/09/2025	Scene is transitioned into object detection scene	Pass		
Execution Summary:	By making AR Session not destroy on new Scene, Start button can transition into Object Detection Scene					
Acknowledgment: Generated from the CapStone process management system ©2025						

This test validated whether the application successfully transitioned to the Object Detection scene after clicking "Next." The first test failed due to the AR Session object being destroyed between scenes. After adjusting the Unity scene structure to preserve the AR Session, the test passed successfully, confirming the scene transition works under the corrected configuration.

Table 8.3.2 – Execution Report for Test Case TC-007

Table 8.3.2. Execution Report of Test Case TC-007

Project Name:	Intuitive user interface with AR+Gen AI					
Test Case ID:	TC-007					
Testing Tools Used:	Manual Test					
Testing Type:	Functional testing					
Execution Steps:	1	Turn on HMD and connect to Android Device				
	2	Detect object on HMD camera				
	3	Generate Information about the detected object				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Aaron	02/28/2025	Prompt did not go through and displayed html error page	Fail	The Google Gemini API was not updated	02/28/2025 by Aaron
2	Aaron	02/28/2025	Prompt went through via API request and AI generated information came back	Pass		
Execution Summary:	The prompt was not going through to the Google Gemini API because it was not updated, by testing and checking the versioning, the API request went through and relevant info was AI generated.					
Acknowledgment: Generated from the CapStone process management system ©2025						

This test confirmed that a user-generated prompt could be passed to the Google Gemini API. Initially, the API was not updated and caused a failed response. After correcting the versioning, the test passed, and the system returned the appropriate AI-generated output. This test verifies the GenAI integration and update management process.

Table 8.3.3 – Execution Report for Test Case TC-008

Table 8.3.3. Execution Report of Test Case TC-008

Project Name:	Intuitive user interface with AR+Gen AI					
Test Case ID:	TC-008					
Testing Tools Used:	Manually tested					
Testing Type:	Functional testing					
Execution Steps:	1	Run the HMD and connect it to the Android phone				
	2	Detect object that is not supposed to have info generated about it (Person)				
	3	Observe Generated AI Info				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Aaron	02/21/2025	Generated random information about a person when it should not	Fail	If the detected object is a certain label, such as person, we should not send the prompt.	02/21/2025 by Aaron
2	Aaron	02/21/2025	Tailored error message for labels that are not food related.	Pass		
Execution Summary:	Error message will detect if a label does not fall under a specific food category.					
Acknowledgment: Generated from the CapStone process management system ©2025						

This test ensured that the system would reject unrecognized object labels such as "person" and not generate a response. The initial failure showed the system attempted to generate info for irrelevant labels. The logic was updated to prevent sending prompts for unsupported objects, resulting in a successful retest.

Table 8.3.4 – Execution Report for Test Case TC-009

Table 8.3.4. Execution Report of Test Case TC-009

Project Name:	Intuitive user interface with AR+Gen AI						
Test Case ID:	TC-009						
Testing Tools Used:	Manually tested						
Testing Type:	Exception handling testing						
Execution Steps:	1	Connect HMD to Phone					
	2	Launch application					
	3	Make multiple API calls in quick succession					
	4	Observe results					
Test Execution Records:							
#	Tester	Test Date	Actual Result	Status	Defect	Correction	
1	Aaron	02/21/2025	API accepts all calls in a short amount of time(lags systems)	Fail	No cooldown in script	02/22/2025 by Mason	
2	Mason	02/22/2025	API accepts 2 call per minute	Pass			
Execution Summary:	We limited the number of API calls						
Acknowledgment: Generated from the CapStone process management system ©2025							

This test checked if the system enforces API rate limits. The system initially failed, allowing multiple API calls rapidly without cooldown. After adding a cooldown script, the test passed. This ensures proper rate-limiting behavior for the GenAI API.

Table 8.3.5 – Execution Report for Test Case TC-010

Table 8.3.5. Execution Report of Test Case TC-010

Project Name:	Intuitive user interface with AR+Gen AI						
Test Case ID:	TC-010						
Testing Tools Used:	Manually tested						
Testing Type:	Functional testing						
Execution Steps:	1	Turn on HMD and connect to Android phone.					
	2	Run application					
	3	Present desired object to be detected					
	4	Call Google Gemini API					
Test Execution Records:							
#	Tester	Test Date	Actual Result	Status	Defect	Correction	
1	Aaron	02/21/2025	Information was not relevant to the object detected.	Fail	API was not updated	02/21/2025 by Aaron	
2	Aaron	02/21/2025	Information was relevant to the object detected	Pass			
Execution Summary:	It now works because the information is relevant to the object detected. Therefore, it passed.						
Acknowledgment: Generated from the CapStone process management system ©2025							

This test evaluated the accuracy of the AI's response in relation to the detected object. The initial failure resulted from an outdated API version. After updating, the correct object-related info was returned. This verifies prompt-to-response relevance in the AI pipeline.

Table 8.3.6 – Execution Report for Test Case TC-004

Table 8.3.6. Execution Report of Test Case TC-004

Project Name:	Intuitive user interface with AR+Gen AI					
Test Case ID:	TC-004					
Testing Tools Used:	Manual Test					
Testing Type:	Component interface testing					
Execution Steps:	1	Connect HMD to Phone				
	2	Launch application				
	3	Observe Results				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Kyle	11/11/2024	HMD stays blank while the application displays on the phone.	Fail	No AR Session in Unity	11/11/2024 by Kyle
2	Kyle	11/11/2024	HMD is blank again	Fail	AR Camera was disabled during runtime	11/11/2024 by Kyle
3	Kyle	11/11/2024	HMD is blank again	Fail	Wrong camera calibration and AR Camera Manager script enabled	11/16/2024 by Mason
4	Mason	11/16/2024	HMD displays Main Menu	Pass		
Execution Summary:	At first our hierarchy was set up incorrectly in Unity, and after we set it up correctly it finally displayed what it showed in Unity.					
Acknowledgment: Generated from the CapStone process management system ©2025						

This component integration test ensured the HMD could connect and interact with the mobile app. Initial failures were due to no AR session, camera disablement, and script issues. After fixing the Unity setup and calibration, the system performed as expected.

Table 8.3.7 – Execution Report for Test Case TC-005

Table 8.3.7. Execution Report of Test Case TC-005

Project Name:	**Intuitive user interface with AR+Gen AI**					
Test Case ID:	TC-005					
Testing Tools Used:	Manual Test					
Testing Type:	Component interface testing					
Execution Steps:	1	Connect HMD to Phone				
	2	Launch application				
	3	Check red light on HMD				
Test Execution Records:						
#	**Tester**	**Test Date**	**Actual Result**	**Status**	**Defect**	**Correction**
1	Mason	01/29/2025	Red light does not turn on	Fail	We did not have access to the camera permissions	02/02/2025 by Mason
2	Mason	02/02/2025	Red light turns on	Pass		
Execution Summary:	We just had to modify the Android Manifest to ask for permissions.					
Acknowledgment: Generated from the CapStone process management system ©2025						

This test verified HMD camera readiness. The red light indicator initially failed due to missing permissions. After updating the Android Manifest and enabling access, the test passed. This confirms that the camera setup and system permissions were properly handled.

Table 8.3.8 – Execution Report for Test Case TC-001

Table 8.3.8. Execution Report of Test Case TC-001

Project Name:	**Intuitive user interface with AR+Gen AI**					
Test Case ID:	TC-001					
Testing Tools Used:	Manual Test					
Testing Type:	Functional testing					
Execution Steps:	1	Connect HMD to Phone				
	2	Launch application				
	3	Observe results				
Test Execution Records:						
#	**Tester**	**Test Date**	**Actual Result**	**Status**	**Defect**	**Correction**
1	Mason	11/22/2024	Main menu does not load.	Fail	Wrong default scene	11/22/2024 by Mason
2	Mason	11/22/2024	Main menu loads but doesn't render.	Fail	Wrong camera calibration	11/23/2024 by Mason
3	Mason	11/23/2024	Main menu loads correctly.	Pass		
Execution Summary:	Main menu loads correctly on start.					
Acknowledgment: Generated from the CapStone process management system ©2025						

This test ensured the main menu loaded upon launch. It initially failed due to incorrect default scene and calibration. After correcting scene references and camera setup, the test passed. This confirms proper startup behavior.

Table 8.3.9 – Execution Report for Test Case TC-00

Table 8.3.9. Execution Report of Test Case TC-002

Project Name:	**Intuitive user interface with AR+Gen AI**					
Test Case ID:	TC-002					
Testing Tools Used:	Manual Test					
Testing Type:	Usability testing					
Execution Steps:	1	Connect HMD to Phone				
	2	Launch application				
	3	Ensure that phone and HMD have different views				
Test Execution Records:						
#	**Tester**	**Test Date**	**Actual Result**	**Status**	**Defect**	**Correction**
1	Kyle	11/20/2024	HMD does not render anything	Fail	Dual Render Fusion disabled	11/22/2024 by Kyle
2	Kyle	11/22/2024	Phone and HMD render same	Fail	Second camera was disabled	11/22/2024 by Kyle
3	Kyle	11/22/2024	Both show different views but same scene	Fail	Need another scene connected	
Execution Summary:	Dual Render Fusion still does not fully work correctly.					
Acknowledgment: Generated from the CapStone process management system ©2025						

This usability test checked if dual render fusion was working. All three initial runs failed due to disabled rendering, single-camera setup, and scene configuration issues. Fixes were applied, but based on the results shown, you should confirm whether it eventually passed or if it needs to be re-tested.

Table 8.3.10 – Execution Report for Test Case TC-006

Table 8.3.10. Execution Report of Test Case TC-006

Project Name:	**Intuitive user interface with AR+Gen AI**					
Test Case ID:	TC-006					
Testing Tools Used:	Manual Test					
Testing Type:	Functional testing					
Execution Steps:	1	Connect HMD to Phone				
	2	Launch application				
	3	Enable permissions for camera access				
	4	Go to object detection scene				
Test Execution Records:						
#	**Tester**	**Test Date**	**Actual Result**	**Status**	**Defect**	**Correction**
1	Mason	02/13/2025	Camera frame access sample scene does not work	Fail	Camera frame access feature disabled in project settings	02/13/2025 by Mason
2	Mason	02/13/2025	Sample scene still doesn't work	Fail	Camera Frame Access Script disabled	02/14/2025 by Kyle
3	Kyle	02/14/2025	Sample scene still doesn't work	Fail	Camera permissions disabled	02/17/2025 by Mason
4	Mason	02/17/2025	Sample scene still doesn't work	Fail	Scene hierarchy set up incorrectly	02/17/2025 by Mason
5	Mason	02/17/2025	Sample scene still doesn't work	Fail	AR Camera Manager was disabled	02/22/2025 by Kyle
6	Kyle	02/22/2025	Sample scene still doesn't work	Fail	Debug script caused runtime errors	02/22/2025 by Kyle
7	Kyle	02/22/2025	Sample scene still doesn't work	Fail	We don't know	
Execution Summary:	We still do not know what is wrong with the sample scene.					

This test aimed to verify real-time video streaming from the HMD. Multiple failures occurred due to a combination of camera access issues, permission settings, incorrect scene hierarchy, and runtime errors. These were fixed iteratively. The summary reflects ongoing debugging efforts, and a successful pass should be confirmed before final delivery.

9. Challenges & Open Issues

9.1. Challenges Faced in Requirements Engineering

The biggest challenge we encountered in requirements engineering was dealing with the broad and loosely defined project scope. While this flexibility allowed for creativity, it also made it difficult to establish clear and specific requirements. Along with determining the size and scope of the project, we also had to choose which frameworks and tools to work with, as we were presented with multiple options. This meant deciding not only what features and functionality to include but also which technologies best aligned with our goals. Balancing ambition with practicality, we needed to ensure that the selected frameworks were feasible to implement within the given time frame and realistic in terms of development complexity. Ultimately, these decisions required thoughtful consideration to define a scope that was both achievable and effective.

9.2. Challenges Faced in System Development

During the development process, we encountered significant challenges that required us to adapt and troubleshoot extensively. One major hurdle was the incompatibility of the Niantic Lightship API with our project requirements. Initially, we planned to leverage its capabilities, but its limitations forced us to pivot to alternative solutions, consuming valuable time and resources. Additionally, implementing Dual Render Fusion presented a steep learning curve. At first, the Head-Mounted Display (HMD) failed to render any visuals, leaving us puzzled about the root cause. Once we resolved the initial setup issues, we faced another challenge with incorrect configuration settings, which caused the HMD and the phone to display identical views instead of the intended separate perspectives. This misconfiguration highlighted the importance of thoroughly understanding and fine-tuning the integration between software and hardware components. These obstacles not only delayed progress but also underscored the complexity of creating a seamless AR experience, pushing us to enhance our debugging and problem-solving strategies. We also had been having trouble with the camera frame access sample scene.

9.3. Open Issues & Ideas for Solutions

The biggest challenge we encountered in requirements engineering was dealing with the broad and loosely defined project scope. While this flexibility allowed for creativity, it also made it difficult to establish clear and specific requirements. In addition to defining the project's scope, we are currently in the process of selecting the appropriate neural network model for object detection, weighing options such as YOLOv2 and YOLOv8 Each has its strengths, and we need to analyze and experiment with both before making a final decision. This added layer of decision-making has required us to balance technical feasibility, compatibility with our goals, and the time constraints of the project, ensuring that the chosen solution is effective and practical to implement within our timeline.

10. System Manuals

10.1. Instructions for System Development

1. Elicit Requirements
2. Map out Software Architecture
3. Implement Software System
4. Create Testing for Functions of Systems

10.2. Instructions for System Deployment

1. Save the project
2. File -> Build Settings -> Build
3. Save APK file to PC
4. Send APK file to android device
5. Connect Android to Snapdragon Spaces HMD
6. Open app and display Unity project to HMD

10.3. Instructions for System End Users

1. Wear the Snapdragon Spaces HMD while it is connected to the associated Android device
2. Navigate through the scene menu to the appropriate scene
3. Look at objects to display relevant information
4. Sort through information and manage cart functionalities

11. Conclusion

11.1. Achievement

- So far in Sprint 9, we were able to achieve some very important milestones such as designing UI for the HMD, creating a working Object Detection Scene, and properly setting up Dual Render Fusion. However, this is just the start of what we plan to achieve with this project.
- We have a working model of our system that can detect objects and display relevant information to the user. We plan to make this work as quickly as possible in a controlled environment, then we slowly expanded the scope of the project until the system is fully function and fulfills our more detailed requirements such as the shopping cart and checkout functionality.

11.2. Lessons Learned

- Our team has learned many valuable lessons that we have kept in mind allowing us to move forward and effectively make progress during our time working on our project.
- The first and most important lesson we learned is utilizing resources to learn the necessary technology stack. We were able to get our environment set up through the help of many different online resources such as the official Snapdragon Spaces documentation, a Snapdragon Spaces course, and some YouTube videos about AR and Unity. By relying on these resources, we were able to develop our own understanding of the necessary technologies as well.
- We also learned the importance of communication and collaboration. By centralizing all of our communications on Microsoft Teams, we had easy access to all of our files, progress updates, and meeting times. Collaboration came in the form of our time blocks for multiple team meetings throughout the week. By having these pre-scheduled meetings, our team was able to consistently make progress in our project.
- It is extremely valuable for each team member to work individually on the project in their free time.

11.3. Acknowledgment

- We want to thank Qualcomm for sponsoring a very valuable opportunity for us to work with their product in the form of their Snapdragon spaces AR technology, allowing us to grow through practical industry experience.
- We also want to thank Karen Weeks for the guidance and support throughout this project.
- Also thank you to our faculty advisor, Yongjie Zheng, who has helped us tremendously when it came to eliciting requirements and creating the objectives for this project.
- Finally, Simon Fan, who has taught us the importance of project management and Agile methodology so that these tools can help us progress further into the project.

12. References

[1] DeepLearning.AI, "Introduction to On-Device AI," [Online]. Available: https://www.deeplearning.ai/short-courses/introduction-to-on-device-ai/. [Accessed: Nov. 25, 2024].

[2] XR Bootcamp, "Snapdragon Spaces," [Online]. Available: https://xrbootcamp.com/snapdragon-spaces/. [Accessed: Nov. 25, 2024].

[3] Qualcomm Technologies, Inc., "Setup Guide for Unity," [Online]. Available: https://docs.spaces.qualcomm.com/unity/setup/setup-guide#import-the-package. [Accessed: Nov. 25, 2024].

[4] Unity Technologies, "Barracuda: Neural Networks for Unity," [Online]. Available: https://docs.unity3d.com/Packages/com.unity.barracuda%401.0/manual/index.html. [Accessed: Nov. 25, 2024].

[5] Ultralytics, "ONNX Integration with Ultralytics," [Online]. Available: https://docs.ultralytics.com/integrations/onnx/. [Accessed: Nov. 25, 2024].