Software Requirements Specification: Realm Software Engineering

Team #13, ARC
Avanish Ahluwalia
Russell Davidson
Rafey Malik
Abdul Zulfiqar

Contents

Revision History

Table 1: Revision History

Date	Version	Notes
2024-10-11	1.0	Initial SRS
2024-10-29	1.1	Reorganization and addition of NFRs, Completed missing sections

1 Introduction

1.1 Document Purpose

The purpose of this document is to outline key functional and non-functional requirements and constraints for the AR project, Realm. This will help plan and guide the developers throughout the development and testing of the app. The intended audience is for the developers and testers of this project. Additionally, it can serve as a reference to the stakeholders who require an understanding of the capabilities and expectations of the project.

1.2 Product Scope

Realm (version 1.4) is an AR platform that aims to create a cohesive augmented reality experience that enables users to engage in social, educational and entertaining activities through the use of AR content. The app will allow users to place AR objects, react to them, interact with friends, and participate in educational tours.

The key objectives of this platform include fostering social interactions through shared and customizable AR experiences and promoting educational content via AR tours.

1.3 Definitions, Acronyms and Abbreviations

- *AR object*: A 2D/3D model representing an object that can be presented in AR
  - *Artifact*: a 3D AR Object
  - *Sticker*: a 2D AR Object
- *AR object instance*: An instance of an AR object placed in the real world, defined by an AR object, and a location
- *AR object cluster*: A group of AR objects present within a confined area.
- *Users*: A term for anyone who uses the app.
- *Organization users*: Users who belong to an organization that have the ability to create tours. They are affiliated with a particular approved organization who have the ability to modify tours within their organization's domain.
- *General users*: Users that can access most of the app functionalities except creating tours.
- *Admins*: People who have access to nothing else but the admin interface within the app.
- *Sub-Realm*: A construct that specifies a set of users that are it's members, and a set of AR objects that are shared between its members
- *Encryption standard*: A set of algorithms used to encode data to ensure that it can be viewed by authorized users only.
- *Restricted Area*: A space that should not be accessible to the general public.

1.4 References

References

[1] O. of the Privacy Commissioner of Canada, "Pipeda in brief." https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/pipeda_brief/. Accessed: 2024-10-07.

[2] S. . S. Hogg, "Business tax records." https://www.hss-ca.com/wp-content/uploads/2013/07/hss-business-tax-records.pdf. Accessed: 2024-10-07.

[3] Google, "Google play: Developer content policy." https://play.google.com/about/developer-content-policy/. Accessed: 2024-10-07.

[4] Apple, "App review guidelines." https://developer.apple.com/app-store/review/guidelines/. Accessed: 2024-10-07.

2 Product Overview

2.1 Product Perspective

Realm, as specified in this SRS document, is a self-contained product designed to provide users with an immersive social, educational, and entertainment experience with the help of AR. This product is not part of an existing product family and does not replace any current systems. Instead, it is developed as a standalone system that allows users to create, interact with, and share virtual objects, such as 2D stickers and 3D artifacts, in real-world environments.

The app operates independently but may collaborate with third-party content providers, such as educational institutions or businesses, to offer //customized AR experiences. If needed, Realm could integrate with larger systems in the future, such as educational platforms or entertainment networks, by establishing appropriate interfaces for data sharing and interaction.

The app is to be compatible with most smartphone devices, as long as they have a camera.

2.2 Product Functions

- Tutorial
  - The user will be able to complete a walkthrough tutorial of all major app features.
- Tour Management

  - *Organization users* will be able to create/modify/delete AR tours owned by their organization.

- Touring
  - *General users* will be able to experience AR tours made by approved organizations.
- User Profile Management
  - Users can create, edit, and manage user profiles as well as manage privacy settings.
- AR Object Interaction
  - Users will be able to place, edit, and interact with 2D stickers and 3D artifacts within the sub-realm.
  - Users can react to and engage with objects placed by other users.
- Social Interaction
  - Users will be able to create, edit, and add/remove friends to a sub-realm, where they can each share AR objects within the sub-realm.
- Object Creation
  - Allow users to create their own AR objects and share them with others.
  - Objects can be created via a 'Scan and Upload' option or generate one using a user prompt.
- Reporting and Moderation
  - Allows users to report inappropriate objects which can go for review and moderation.
- Managing Settings
  - Manage a variety of settings including accessibility settings, display settings, privacy settings, profile settings, and sub-realm settings.

2.3 Product Constraints

- **Interfaces to Users**: The platform must be compatible with both Android and iOS devices, meaning the design must accommodate different screen sizes and operating system guidelines. Additionally, the app requires user-friendly interfaces for interacting with AR objects and managing sub-realms.

- **Hardware Constraints**: The app depends on mobile devices with AR capabilities (e.g., cameras, GPS, accelerometers). Users with older or less powerful devices might experience limited functionality or performance issues.
- **Quality of Service**: Realm needs to have a smooth interaction, with little to no lag. It must also handle multiple users within a shared AR space (sub-realm).
- **Standards Compliance**: The app has to comply with relevant privacy laws and app store requirements for both Android and iOS.
- **Design and Implementation Constraints**: The app needs to provide cloud storage for AR objects and user data, and its design should allow for easy scalability as the user base grows. Additionally, the AR features and general app use should be intuitive and easy to use for users.
- **Budget Constraints**: The project has a maximum budget of $750.

2.4 User Characteristics

There are three types of users.

General Users:

- Frequency of Use: Occasional to frequent.
- Functions Used: Creating, interacting with, and sharing AR objects; joining sub-realms; participating in AR tours.
- Technical Expertise: Moderate; basic understanding of mobile apps and social platforms.
- Security/Privilege Level: Standard user access with limited permissions (e.g., cannot create tours or manage content).
- Importance: Most important user class, as they represent the primary user base for the platform.

Organizational Users:

- Frequency of Use: Frequent.
- Functions Used: Creating and managing AR tours, educational content, and sub-realms for public or private use.
- Technical Expertise: Moderate to advanced; comfortable with content creation tools and AR platforms.
- Security/Privilege Level: Higher-level access to manage and publish content for users.

Table 2: Apportioning of Requirements

Function	Priority	Dates
Object Creation	1	2024-11-11
AR Object Interaction	1	
Tour Management	2	2025-12-01
Touring	2	
Social Interaction	2	
Reporting and Moderation	2	
User Profile Management	3	2025-01-15
Managing Settings	3	
Tutorial	3	

- Importance: Highly important, as they provide some of the platform's key content.

Administrators/Moderators:

- Frequency of Use: Regular.
- Functions Used: Managing user permissions, moderating content, handling reports of inappropriate AR objects.
- Technical Expertise: Advanced; familiar with platform management and moderation tools.
- Security/Privilege Level: Full access to the platform's management functions, including user and content control.
- Importance: Less important for day-to-day use, but crucial for ensuring platform integrity and moderation.

2.5 Assumptions and Dependencies

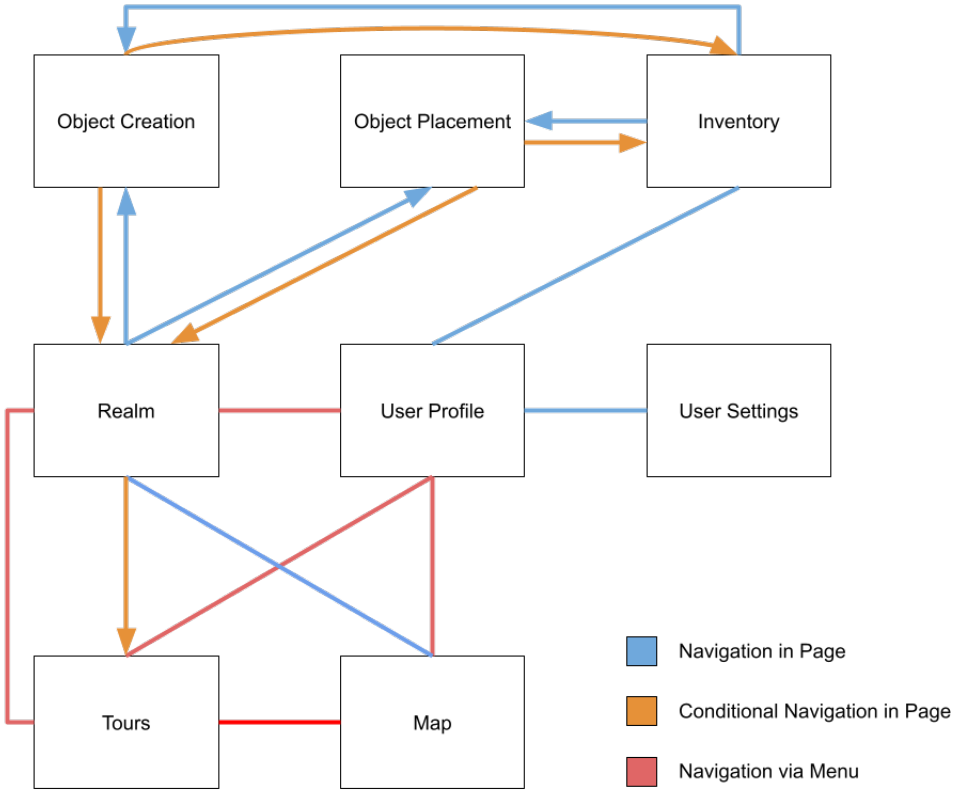
- User has used a mobile app before.
- User has internet access.
- User device can provide GPS location services.
- User device has a camera.
- User device utilizes either Android or iOS operating systems.

2.6 Apportioning of Requirements

3 Requirements

3.1 External Interfaces

3.1.1 User interfaces



Interface Names

- **Realm:** the interface in the app that provides a view of AR objects imposed on a feed of the user's camera.
- **Tours:**

  - **Tour List:** Shows a list of the available tours and an option to start them.
  - **Tour Editor:** Allows organization users to edit tours and publish changes.

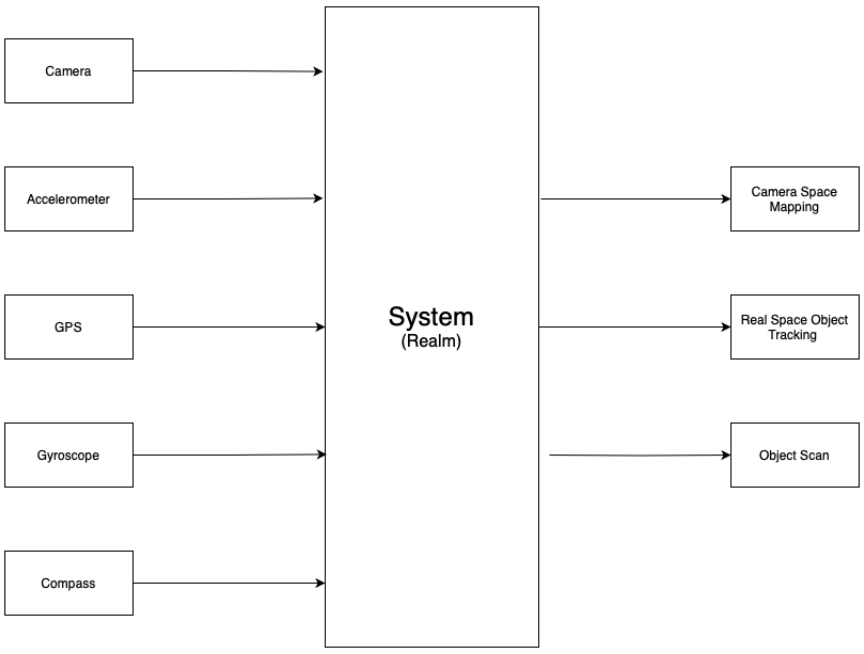
- **Maps:** Map view displaying objects placed around you
- **Inventory:** A display of available AR objects (personal objects, objects mapped to a sub-realm and application preset objects)
- **Profile:**
  - **Sub-Realms:** Ability to add, edit and interact with sub-realm settings
  - **Friends:** Managing friends
  - **Settings:** Accessibility settings, Display settings, Privacy settings, Profile Settings, Sub-Realm settings
  - **Help:** FAQ and request org account
- **Placement Editor:** Able to move, rotate, resize the object prior to confirming placement.

There must be a navigation menu that can be accessed from any page that allows the user to navigate to any of the four main pages: Realm, User Profile, Tours, and Map

3.1.2 Look and Feel Requirements

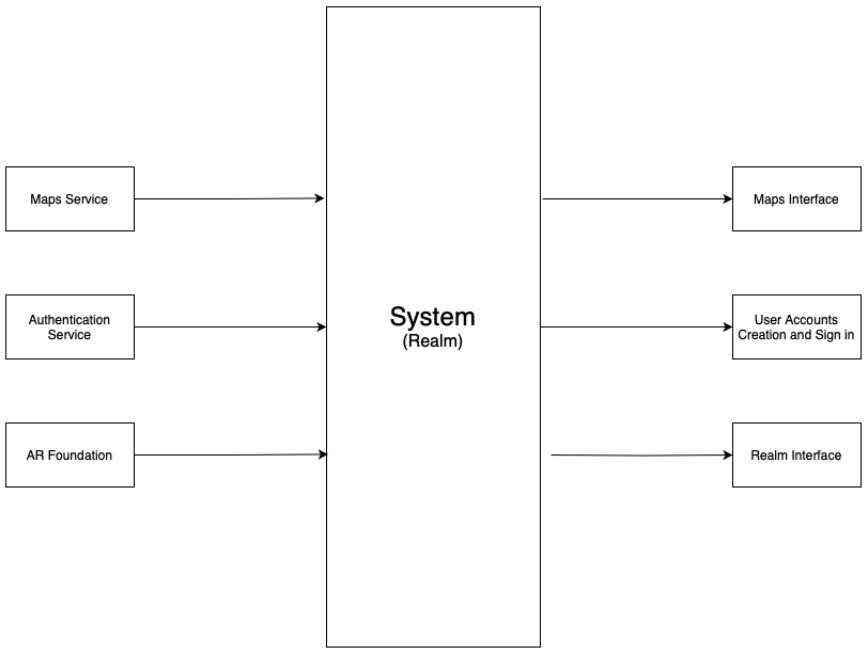
EI-LF1: The Realm Interface must provide an immersive view of AR objects imposed on the real world, and must be minimally interrupted by other UI elements.
Fit Criteria: Permanent UI elements besides the camera feed should take no more than 10% of the space on the screen.

3.1.3 Hardware interfaces



The above figure shows the hardware inputs of the system and the outputs of the system that directly follow from the hardware inputs.

3.1.4 Software interfaces



The above figure shows the external systems our system will interact with, and the outputs of the system that directly follow from the external software inputs

3.2 Functional

The functional requirements of the system broken down by the features they specify

3.2.1 Maps

MP-FR1: The map should provide the user's location

MP-FR2: The map should display location markers for AR object clusters

MP-FR3: The marker should display the count of objects present in a tight cluster

MP-FR4: The map should display location markers for AR object cluster

MP-FR5: The map should display indications of objects from all sub-realms that the user is a part of

MP-FR6: When a marker is selected, the option to provide directions should be available

MP-FR7: The map should give directions to the user for a selected marker

MP-FR8: The objects shown on the map should be grouped to avoid clutter

MP-FR9: The map should allow the user to terminate navigation.

MP-FR10: The map should be able to identify restricted areas and disallow navigation.

3.2.2 Inventory

IV-FR1: The user should be able to delete objects from their inventory

IV-FR2: The user should be able to add objects to their inventory

IV-FR3: The inventory should at least have the application-provided objects

IV-FR4: The inventory should contain a maximum of 100 personal objects excluding the application-provided ones

IV-FR5: The personal objects present in the inventory should either be generated by the user or shared by other users

IV-FR6: The inventory should provide the count of the total objects

IV-FR7: The inventory should store both 2D and 3D AR objects

IV-FR8: The user should be able to add their objects to a "favourite group"

IV-FR9: The user should be able to sort their objects according to the number of uses, favourites and storage size

IV-FR10: The inventory should display the 3D objects in a continuous rotating state

IV-FR11: The user can change the inventory background color

3.2.3 Object Scan

OS-FR1: The user should be able to scan their surroundings to create objects.

OS-FR2: The user should select whether a 2D or 3D object is being scanned.

OS-FR3: While scanning, a real-time render of the scanned portion will be shown to the user.

OS-FR4: The scanning state should not be longer than 120 seconds.

OS-FR5: When the user has completed scanning, they should be able to confirm that the scanning process is finished.

3.2.4 Object Upload to Inventory

OUI-FR1: The object render provided by the scan would be displayed to the user.

OUI-FR2: The user should be able to remove unneeded features of the render after the target is done being scanned.

OUI-FR3: For 2D objects, the user can crop and resize the object in the edit interface.

OUI-FR4: Once the user is satisfied with the scanned object, they should be able to confirm the creation of the AR object.

OUI-FR5: The user should be able to name their created object.

OUI-FR6: The name should only contain ASCII characters.

OUI-FR7: The user-created name of the object should be stored along with the AR object.

OUI-FR8: The date and time of creation should be stored with the AR object.

OUI-FR9: The user information should be stored with the AR object.

OUI-FR10: The storage size should be stored with the AR object.

OUI-FR11: The type (2D or 3D) of the object should be stored with the AR object.

OUI-FR12: The user should be able to select portions of the object and change its color.

3.2.5 Custom AR Object Generation via prompt

POG-FR1: The user should be able to enter a prompt.

POG-FR2: The user prompt should not be longer than 200 characters (including spaces and punctuation).

POG-FR3: The current character count for the prompt should be displayed to the user.

POG-FR4: The user prompt should not contain profanity.

POG-FR5: The user should select whether they want a 2D or 3D object.

POG-FR6: The user should be able to confirm that they have finished writing their prompt and would like to generate the AR object.

POG-FR7: The application should provide multiple AR objects for the entered prompt.

POG-FR8: After objects are generated, the user should be able to select an object from one of the provided options.

POG-FR9: Once the user has chosen an object, they should be able to add it to their personal inventory.

POG-FR10: The objects can be previewed by rotating them.

3.2.6 Tutorial

TU-FR1: The app shall have a step-by-step interactive guide of how to use all major app features.

TU-FR2: The app shall prompt the user to complete the app tutorial after they create an account.

TU-FR3: The app shall allow the user to leave the tutorial at any time.

TU-FR4: The tutorial shall be available at any time through the app's help page in the **Settings screen**.

TU-FR5: The tutorial will involve user participation to directly use functionality in a sandbox environment.

TU-NFR1: Each interaction step should not take the user more than 15 seconds to figure out.

TU-NFR2: The entire tutorial shall not take longer than 5 minutes to complete for 80% of users.

3.2.7 Tour Management

TM-FR1: The tour management functionality within the app shall only be available to Organization users.

TM-FR2: Tours within the app can be created as a "draft" to make it available to other Organization users but not released to the public.

TM-FR3: Tours within the app can be published to the public from a "draft" or from directly after creation.

TM-FR4: Organization users will be able to customize the following for each of their tours:

(a) Name

(b) Description

(c) Route

i. Will be editable on a map of the tour area

ii. Intended direction of travel can be set

iii. Estimated time of completion that will be automatically determined by the distance if not set

(d) Objects

i. AR object can be placed along the route at specific geographic locations along with description text for each of the AR object

ii. Historical information text popups can be placed along the route

iii. All text will have an audio playback option with text-to-speech or pre-recorded audio (if available)

(e) Price

i. Can be free or

ii. a price under $10

(f) Relevant Web Link(s)

TM-FR5: Organization users will have the ability to preview the tours that belong to their organization.

TM-FR6: Organization users will have the ability to edit tours that belong to their organization.

3.2.8 Touring

TR-FR1: The touring functionality within the app shall only be available to General users.

TR-FR2: The app will have three avenues for a user to find tours:

TR-FR2.1: The app will allow General users to see a list of available tours through the **Tour List Interface**.

- The tours can be grouped in this page by organization or by location

TR-FR2.2: The app will also have tours show up in a push notification (if configured by a user) when in close proximity to a tour area.

TR-FR2.3: Locations can place QR codes at the starting location of the tour which can be scanned by a mobile camera that will open the tour preview in the app.

TR-FR3: Users will be able to preview a tour to see the following information:

(a) Name

(b) Description

(c) Relevant web link(s)

(d) Tour distance (auto-calculated from route)

(e) Estimated time of completion

(f) A map that will show the route and locations of AR objects

(g) Price

TR-FR4: Once a user starts a tour they will have two main views they can switch between:

TR-FR4.1: One of the tour views is a map

- The designated tour area will be outlined
- The user's current location will be shown
- Intended route and direction will be overlaid
- Location of AR objects will be marked

TR-FR4.2: One of the tour views is an AR view

- Will be very similar to the interface of the **Realm Interface**
- Have an added indicator of the intended direction
- Popups for historical information

3.2.9 Object Placement

OP-FR1: The system must have the capability to store object instances that specify the following details:

- The AR object that was placed.
- The sub-realm(s) in which the object instance will be shared.
- All data required to precisely reproduce the position and orientation of the object in real space.

OP-FR2: The system must provide a way for a user to place AR objects in the space around them, creating object instances.

OP-FR2.1: The system must provide a way for a user to select an object either from their inventory or by generating a new object.

OP-FR2.2: The system must provide a way for the user to select one or more sub-realms of which they are a member.

OP-FR2.3: The system must provide a way for the user to position an AR object in real space.

OP-FR3: The system must limit the number of objects that a user can place

OP-FR3.1: The user must be limited in how many objects they can place in a given area

OP-FR3.2: The user must be limited in how many objects they can place over a given period

OP-FR4: The system must have an automated mechanism to retry the upload and storage of object instances when an initial attempt fails.

3.2.10 Realm Interface

RI-FR1: The Realm interface must provide a view of the AR object instances in the area around the user overlaid on their camera feed positioned in real space.

- **RI-FR1.1:** AR object instances presented on the Realm interface must always appear to change perspective correctly with respect to the user's camera position and angle.
- **RI-FR1.2:** AR object instances that overlap or clutter a given area in real space will not be presented simultaneously.
- **RI-FR1.3:** AR object instances reproduced on the Realm interface must be in the correct position and orientation as defined by the object placement.

RI-FR2: The Realm interface must keep track of a selected sub-realm which determines which object instances are displayed.

- **RI-FR2.1:** The Realm interface must provide an indication of what sub-realm is currently selected.
- **RI-FR2.2:** The Realm interface must provide a way for the user to change the selected sub-realm.

RI-FR3: The Realm interface must provide a control to allow the user to enter the object placement workflow.

RI-FR4: The Realm interface must provide a control to allow the user to enter the object scanning workflow.

RI-FR5: When the user is near the starting point of a tour, the Realm interface must provide an indication that there is a tour nearby as well as a link to the tour preview.

RI-FR6: The interface must display warnings to alert users of going outside during dangerous weather conditions.

RI-FR7: The interface must provide ample warning to users if they are close to a real-world hazard and on track to collide with it.

RI-FR8: The interface must provide an offline view without location syncing for interactive components.

3.2.11 Admin Interface

AI-FR1: The admin interface within the app shall only be available to Admins.

AI-FR2: The app shall provide an interface for Admins to carry-out their two special roles:

- **AI-FR2.1:** The app shall permit Admins to act on user reports of an AR object by either keeping or removing the AR object.
- **AI-FR2.2:** The app shall permit Admins to review requests for Organization users accounts and either accept or deny the request.

3.2.12 Profile Screen

PS-FR1: The user must enter valid credentials to be authenticated and logged into their account.

PS-FR2: The system shall provide a multi-factor authentication option for user accounts to enhance security.

PS-FR3: Allow the user to change their password.

PS-FR4: The user must be able to view their profile information including username, password, profile picture, and status.

PS-FR5: The user must be able to view all sub-realms that they are part of.

PS-FR6: The user shall be able to view a help page with frequently asked questions and additional help.

3.2.13 Sub-Realms

G-FR1: The user must be able to create a new sub-realm with a sub-realm name, description, and also allow inviting members.

G-FR2: Allow users to add or remove members from a sub-realm.

G-FR3: The user must be able to edit sub-realm settings, including sub-realm name and description.

G-FR4: The user must be able to delete a sub-realm, and all associated data should be removed.

G-FR5: All users within a sub-realm must be able to interact with sub-realm-specific AR content, including shared AR objects and experiences.

3.2.14 Friends screen

FS-FR1: The user shall be able to send friend requests to other users within the platform.

FS-FR2: The user must be able to accept or reject incoming friend requests.

FS-FR3: The user shall be able to view their list of friends.

FS-FR4: The user must be able to remove friends.

3.2.15 Settings

S-FR1: The user shall be able to access and modify Accessibility Settings, including text size, enabling/disabling viewing of object names, and changing language.

S-FR2: The user shall be able to adjust Display Settings, including light/dark mode, AR object visibility.

S-FR3: The user must be able to manage Privacy Settings, including controlling who can see their profile, friends list, and AR interactions.

S-FR4: Allow the user to manage Profile Settings, such as changing their username, password, profile picture, and status.

S-FR5: The user must be able to access sub-realm Settings, as mentioned in the 3.2.13 section of the requirements.

S-FR6: The system must provide a setting to set the app theme to **Dark** or **Light**, with the option to automatically match the system theme

3.2.16 Databases

DB-FR1: All system databases must have an automated mechanism to periodically backup data

DB-FR2: The system must encrypt all user data stored using an encryption standard.

3.3 Use Cases

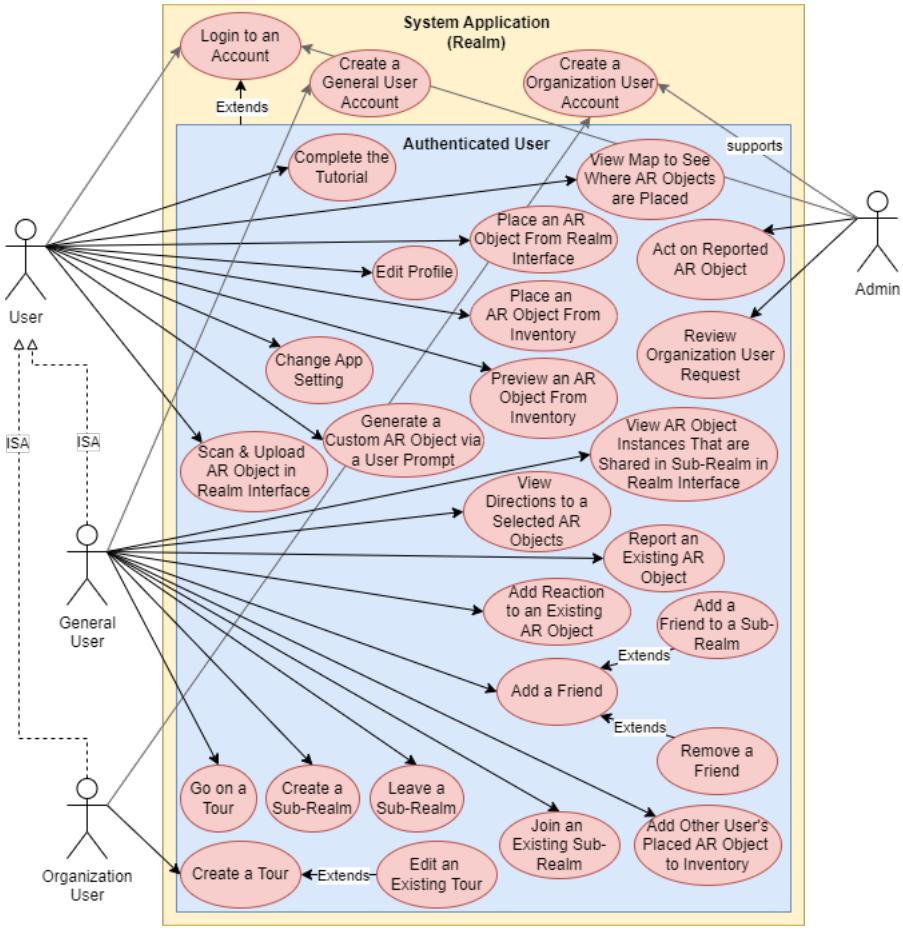


Figure 1: Use Cases

UC1 Complete the Tutorial
Actor: Users
Pre-condition: User has opened the app

Main Success Scenario:

1. User creates an account (**UC23**)
2. System prompts user to complete the tutorial (TU-FR2)
3. User indicates they would like to do the tutorial
4. System opens the **Tutorial Interface**
5. System provides directions to use a feature
6. User tries the feature (TU-FR5)

7. User goes to next feature when satisfied
8. Repeat steps **5-7** for all major app features (TU-FR1)
9. System prompts user to leave and end tutorial
10. User ends tutorial (TU-FR3)
11. System redirects to **Home Interface**

Secondary Scenarios:

1.1: User already has an account
1. User logs into account (**UC24**)
2. User navigates to **Help Interface** (TU-FR4)
3. Go to step **2**

10.1: User chooses to stay in sandbox environment
1. System closes prompt and allows user to use sandbox environment
2. User indicates they want to leave tutorial
3. Go to step **11**

Success Postcondition: The user has a general idea of how all the app's major features work.

UC2 Create a Tour
Actor: Organization users
Pre-condition: User is logged into the app **AND** User has navigated to the **Tour Management Interface**

Main Success Scenario:

1. User selects the option to create a new tour
2. System opens up a form for the user to add the following information about the tour (TM-FR4[1,2,5,6]):
   i. Name
   ii. Description
   iii. Price
   iv. Relevant web link(s)
3. User fills out the information
4. System identifies no blank fields
5. System gives the option to move on to the route configuration
6. User goes to the route configuration
7. System presents tour route configuration
8. User configures tour area, route, and direction (TM-FR4[3])

9. System gives the option to move on to the inventory setup
10. User goes to the inventory setup (TM-FR4[4])
11. System opens the **Inventory Interface**
12. User scans and uploads a new AR object (**UC5**)
13. User repeats step **12** until all desired AR objects are uploaded
14. User indicates they would like to move on to object placement
15. System opens the **Realm Interface** and allows the user to place AR objects and historical information popups in an isolated environment.
16. User places down AR object using the **Realm Interface** (**UC7**)
17. User repeats step **16** until all desired AR objects are placed
18. User indicates they would like to finish tour creation
19. System gives the user the option create the tour as a draft or directly publish it (TM-FR2/TM-FR3)
20. User selects direct publish option
21. System uploads the tour data
22. System makes the tour available to General users

Secondary Scenarios:

4.1: System identifies some blank fields
1. System validates input to ensure all required fields are not empty
2. System determines input passes validation
3. Go to step **5**

4.1.2: System determines input fails validation
1. Go to step **3**

12.1: User does not have any AR objects to scan
1. Go to step **14**

20.1: User selects draft option
1. System uploads the tour data but does not make it available to General users
2. User decides at a later time to publish the draft tour
3. Go to step **22**

Success Postcondition: One of the Organization users has a new tour linked to their organization

UC3 Edit an existing Tour
Actor: Organization users
Pre-condition: User is logged into the app **AND** User has at least one tour connected to their organization **AND** User has navigated to the **Tour Management Interface**

Main Success Scenario:

1. System provides a list of all tours connected to the user's organization
2. User selects one of the tours
3. System shows the preview of the tour (TM-FR5)
4. System provides the option to edit the tour (TM-FR6)
5. User decides to edit the tour
6. System opens up a form for the user to with the following information about the tour that may or not be populated (TM-FR4[1,2,5,6]):
    i. Name
    ii. Description
    iii. Price
    iv. Relevant web link(s)
7. User edits information as desired
8. System identifies no blank fields
9. System gives the option to move on to the route configuration
10. User goes to the route configuration
11. System presents tour route configuration
12. User edits configuration of tour area, route, and direction as desired (TM-FR4[3])
13. System gives the option to move on to the inventory setup
14. User goes to the inventory setup (TM-FR4[4])
15. System opens the **Inventory Interface**
16. User scans and uploads a new AR object (**UC5**)
17. User repeats step **12** until all desired AR objects are uploaded
18. User indicates they would like to move on to object placement
19. System opens the **Realm Interface** and allows the user to place/edit AR objects and historical information popups in an isolated environment.
20. User edits AR object using the **Realm Interface** (**UC7**)
21. User repeats step **16** until all desired AR objects are edited
22. User indicates they would like to finish tour editing
23. System gives the user the option to save the edited tour as a draft or directly publish it (TM-FR2/TM-FR3)
24. User selects direct publish option
25. System uploads the updated tour data
26. System makes the updated tour available to General users

Secondary Scenarios:

5.1: User decides not to edit the tour

1. System goes back to the **Tour Management Interface**

8.1: System identifies some blank fields

1. System validates input to ensure all required fields are not empty
2. System determines input passes validation
3. Go to step **9**

16.1: User decides not to scan and upload any new AR objects

1. Go to step **18**

20.1: User places down new AR object using the **Realm Interface**

1. User repeats step **20.1** until all desired AR objects are placed

24.1: User selects draft option

1. System uploads the updated tour data but does not make it available to General users
2. User decides at a later time to publish the draft of the updated tour
3. Go to step **26**

Success Postcondition: One of the Organization users has edited an existing tour linked to their organization

UC4 Go on a tour
Actor: General users
Pre-condition: User is logged into the app

Main Success Scenario:

1. User navigates to the **Tour Interface**
2. System displays a list of all tours
3. User selects a tour (TR-FR2.1)
4. System opens up the preview of the selected tour (TR-FR3)
5. System also provides the option to take the tour
6. User reviews details of the tour
7. User decides to go on the tour
8. System opens a modified version of the **Realm Interface** in an isolated environment that does not allow them to place any objects or modify existing objects (TR-FR4.2)
9. System shows the intended direction of travel
10. User walks to the highlighted AR object of interest
11. System acknowledges when the user makes it to the AR object
12. System provides historical information regarding AR object

13. System updates the target to be the next AR object
14. Repeat steps **10-13** for every AR object
15. System indicates that the tour has been completed
16. System provides metrics for time taken and distance traveled
17. System prompts user to leave tour
18. User leaves tour
19. System goes back to the tour list

Secondary Scenarios:

1.1: System pushes a notification to the user when they are in the proximity of a tour (TR-FR2.2)

1. System opens up the **Tour Interface**
2. Go to step **4**

1.2: User scans a tour QR code in camera app (TR-FR2.3)

1. System opens up the **Tour Interface**
2. Go to step **4**

7.1: User decides not to go on the tour

1. Go to step **19**

10.1: User indicates they want to open the map

1. System opens up the **Map** view of the tour (TR-FR4.2)
2. System shows the user's current location
3. System shows the indented route, direction and boundaries of the tour
4. System marks the location of AR objects
5. User views the map
6. User closes the map
7. Go to step **9**

14.1: User does not want to go to all AR objects

1. User indicates they wish to exit the tour
2. Go to step **19**

18.1: User does not leave tour

1. System removes target object and allows user to explore AR objects in the isolated environment
2. User eventually decides to exit the tour
3. Go to step **19**

Success Postcondition: One of the General users has completed a tour

UC5 Scan and upload AR object to the user inventory
Actor: General users
Pre-condition: User has an account on the application **AND** is logged into the application **AND** navigates to the Realm display

Main Success Scenario:

1. System displays Realm screen to user
2. User selects control to scan and create an AR object
3. System presents user with the option to scan a 2D or 3D AR object
4. User selects one of the provided options
   (a) User selects 2D AR object option
   (b) User selects 3D AR object option
5. System displays a scanning interface
6. User performs motions to scan the targeted object
7. System provides a render for the scanned portion (happens concurrently with step **6**)
8. User selects the option to confirm that the scanning process is complete
9. System navigates user to an editor with the complete render of the target entity
10. User performs actions to edit the render to meet their needs
    (a) User removes unneeded parts of the render
    (b) User changes the colour of selected portions
11. System applies the changes to the AR object intended by the user
12. User confirms that the editing process is completed
13. System receives the user confirmation and provides a preview of the final AR object with the options to discard or add it to their inventory
14. User selects what they would want to do with the new AR object
    (a) User returns to the editing interface for further customization (Go to step **10**)
    (b) User selects control to add the object to their inventory
    (c) User wants to discard the object (Go to Secondary Scenario 2)
15. System adds the new AR object to the User's inventory and notifies the User about adding the object to their inventory
16. System returns to the Realm screen

Secondary Scenarios:

(3.1, 5.1) User cancels scanning process

1. Main scenario steps *1-3 or 1-5*
2. User selects option to cancel the scanning process
3. System receives User's request to cancel scanning process and returns to the Realm screen

9.1 User discards the scanned AR object

1. Main scenario steps *1-9*
2. User selects option to discard the scanned object
3. System receives User's request and presents confirmation prompt for discarding object
4. User interacts with confirmation prompt
   (a) User selects to discard object
   (b) User selects to not discard object (Go back to Main scenario step **13**)
5. System receives User response and deletes object and returns to the Realm screen

Success Postcondition: The User can view the new object in their inventory

UC6 Generate a custom AR object via a user prompt
Actor: General users
Pre-condition: User has an account on the application **AND** is logged in to the app **AND** has navigated to the **Realm Interface**

Main Success Scenario:

1. User is present in the Realm Interface and performs actions to navigate to the object prompt generation interface
2. System presents the object prompt generation interface
3. User enters their desired prompt and confirms that the prompt is complete
4. System checks the prompt for profanity
   (a) If the prompt contains profanity, system notifies user that the prompt is invalid due to profanity and requests the user to reenter their prompt (Go to step **2**)
   (b) If the prompt does not contain profanity, system accepts the prompt and generates objects
5. User selects from an array of objects provided by the system
6. System provides the options to **preview the object**, **add it to user inventory** and **place object in the Realm Interface**
7. User selects one of the options
   (a) User selects the options to preview the object

(b) User selects to add object to their inventory

(c) User selects option to place object in the Realm Interface

8. System performs the action selected by the user

(a) User selects option to preview object

A. System opens the AR object in the Preview Interface

B. User performs actions on the AR object

- User rotates the object to view different angles
- User zooms in and out to view object details

C. System provides the view of the AR object according to the User's performed actions

(b) User selects option to add object to inventory

A. System adds object to user inventory

(c) User selects option to place object in the Realm space (Go to **UC7** Main Success Scenario)

Secondary Scenarios:

(2.1, 4(b).1) User cancels the AR object generation

1. Main scenario steps *1-2 or 1-4(b)*

2. User selects control to cancel object generation via prompt

3. System receives user request and returns to the Realm interface

(2.1, 4(a).1) User reenters prompt due to presence of profanity

1. Main scenario steps *1-2 or 1-4(a)*

2. User is notified of profanity in the prompt and reenters their prompt

3. Go to Main scenario step 4

4. Continues Main Scenario from step 5

(2.1, 4(b).1) User reenters prompt for a different variation

1. Main scenario steps *1-2 or 1-4(b)*

2. User reenters their prompt

3. System validates user prompt (Go to Main scenario step **4**)

4. Continues Main Scenario from step **5**

Success Postcondition: The User can view the new object in their inventory

UC7 Place an AR object from the **Realm Interface** (RI-FR1)
Actor: General User
Pre-condition: User is logged in to the app **AND** has navigated to the **Realm Interface** (RI-FR1)

Main Success Scenario:

1. System presents **Realm interface**.
2. User performs control to initiate object placement.
3. System presents the **Object Selection Menu** (OP-FR2.1).
4. User selects an existing AR object for placement.
5. System presents the **Sub-Realm Selection Menu** (OP-FR2.2).
6. User selects the sub-realm(s) in which they wish for their object instance to be shared.
7. System presents the **Object Positioning Interface** (OP-FR2.3).
8. User positions the AR object using given controls and confirms when done.
9. System returns to Realm screen.

Secondary Scenarios:

3.1: User creates new object with a prompt:

1. Main scenario steps 1-3.
2. User selects the option to generate an object via prompt.
3. System presents the **Prompt Object Generation Interface**.
4. User completes the prompt object generation workflow.
5. Main scenario steps 5-9.

3.2: User creates a new object with object scanning:

1. Main scenario steps 1-3.
2. User selects the option to generate an object via object scan.
3. System presents the **Object Scanning Interface**.
4. User completes object scanning workflow.
5. Main scenario steps 5-9.

3.3, 5.1, 7.1: User cancels object placement:

1. Main scenario steps 1-3, or 1-5, or 1-7.
2. User selects option to cancel object placement.
3. System returns to the **Realm Interface**.

5.1: User reselects object:

1. Main scenario steps 1-5.
2. User selects option to return to **Object Selection Menu** (OP-FR2.1).
3. Main scenario resumes from step 3.

7.2: User reselects sub-realms:

1. Main scenario steps 1-7.
2. User selects option to return to **Sub-Realm Selection Menu** (OP-FR2.2).

3. Main scenario resumes from step 5.

Success Postcondition: Users that are members of the sub-realm in which the object instance has been shared can see the object instance from the **Realm interface**.

UC8 Place an AR object from the Inventory
Actor: General User
Pre-condition: User is logged in to the app **AND** has navigated to the **Inventory Interface**

Main Success Scenario:

1. System presents Inventory interface.
2. User selects an object they wish to place.
3. System presents object details.
4. User selects option to initiate object placement.
5. System presents the **Sub-Realm Selection Menu** (OP-FR2.1).
6. User selects the sub-realm(s) in which they wish for their object instance to be shared.
7. System presents the **Object Positioning Interface** (OP-FR2.3).
8. User positions the AR object using given controls and confirms when done.
9. System returns to Realm screen.

Secondary Scenarios:

5.1, 7.1: User cancels object placement:

1. Main scenario steps 1-5, or 1-7.
2. User selects option to cancel object placement.
3. System returns to the Inventory screen.

7.2: User reselects sub-realm:

1. Main scenario steps 1-7.
2. User selects option to return to **Sub-Realm Selection** (OP-FR2.2).
3. Main scenario resumes from step 5.

Success Postcondition: Users that are members of the sub-realm in which the object instance has been shared can see the object instance from the **Realm Interface**.

UC9 Preview AR objects from the Inventory
Actor: General users
Pre-condition: User has an account on the system **AND** is logged into the system

Main Success Scenario:

1. User navigates to their profile view
2. System presents the User's profile and inventory
3. User selects controls to view their entire inventory
4. System provides complete view of the inventory
5. User selects an AR object they would like to view
6. System provides the option to preview the AR object
7. User selects the option to preview the AR object
8. System opens the AR object in the Preview Interface
9. User performs actions on the AR object
   (a) User rotates the object to view different angles
   (b) User zooms in and out to view object details
10. System provides the view of the AR object according to the User's performed actions

Secondary Scenarios: N/A

Success Postcondition: User is able to view an AR object in their inventory from different angles

UC10 View object instances that are shared in a sub-realm in the **Realm interface**
Actor: General User
Pre-condition: User is logged in to the app **AND** has navigated to the **Realm Interface** (RI-FR1)

Main Success Scenario:

1. System presents the **Realm Interface**.
2. User selects a sub-realm to see objects from.
3. System only presents object instances in the selected sub-realm.
4. User moves their device to see any angle of the object instance.
5. System continuously presents the object instance with the correct perspective relative to the user's camera.

Secondary Scenarios:

10.1: User is not a member of any sub-realms:

1. System presents **Realm Interface** (RI-FR1).
2. System presents only public object instances.
3. User moves their device to see any angle of the object instance.
4. System continuously presents the object instance with the correct perspective relative to the user's camera.

Success Postcondition: The user can see any AR object instances that are in any sub-realm that they are a part of correctly positioned in real space.

UC11 Check the map to see where objects are placed in the world
Actor: General users
Pre-condition: User has an account on the application **AND** is logged in to the app

Main Success Scenario:

1. User performs actions to navigate to the Map view
2. System presents the Maps Interface
3. User observes location markers with the count of the objects and selects a marker
4. System shows information and options for the marker
   (a) System displays number of visits to the marker, the number of objects available and the area covered by the marker
   (b) System provides the options to receive directions for the selected marker (Go to **UC12**)

Secondary Scenarios: N/A

Success Postcondition: User is able to view cluster of objects in a confined area on the Maps Interface

UC12 Provide directions to a selected group of objects
Actor: General users
Pre-condition: User has an account on the application **AND** is logged in to the app **AND** is present on the Maps Interface

Main Success Scenario:

1. User selects a location marker on the Map Interface
2. System shows information about the marker (see **UC11** Main scenario step *4(a)*) and the option to provide directions for the marker
3. User selects the option to get directions for the location markers
4. System provides a set of instructions to direct the user to the marker
5. User views the instructions and follows the directions provided

6. System tracks User location to check if they have arrived at the marker location (happens concurrently with step 5)
7. User arrives at the selected marker
8. System is informed that the user has arrived at the marker location (based on device location) and notifies the User of their arrival

Secondary Scenarios:

6.1 User ends navigation to marker before reaching the marker

1. Main Sceanrio step *1-6*
2. User selects option to end navigation to the marker
3. System provides a confirmation prompt for ending the navigation
4. User responds to the confirmation prompt
   (a) User selects option to confirm terminating navigation
   (b) User selects option to deny terminating the navigation and continues navigation (Go to Main scenario step *5*)
5. System ends navigation and returns to the Map Interface

Success Postcondition: The User is provided the directions to their chosen marker and have arrived at the marker location

UC13 Change App Settings
Actor: General users
Precondition: User is logged into the app.

Main Success Scenario:

1. User navigates to the settings menu from the app's main interface.
2. System presents the Settings Menu with various categories (e.g., Accessibility, Display, Privacy).
3. User selects a specific settings category to modify.
4. System presents the options within the selected category.
5. User modifies the desired settings (e.g., adjusts brightness, changes privacy preferences).
6. User confirms the changes.
7. System saves the changes locally and within the cloud, and updates the settings.
8. System returns to the main settings screen.

Secondary Scenarios:

User cancels changes

1. Main scenario 1-5
2. User selects the option to cancel changes before confirming.

3. System discards the changes and returns to the main settings screen.

User reselects settings category:

1. Main scenario 1-4
2. User selects the option to return to the Settings Menu and choose another category.
3. Main scenario resumes from step 3.

User has no network connection:

1. Main scenario 1-7
2. System detects that the network is unavailable.
3. System queues the changes for synchronization once the network is reconnected.
4. Main scenario resumes from step 8 when the network is restored.

Success Postcondition: The user's settings are successfully updated locally and, when connected, synchronized to the cloud.

UC14 Report an existing AR object
Actor: General User
Pre-condition: User is logged in to the app **AND** has navigated to the **Realm Interface** (RI-FR1) **AND** there is another user's AR object placed in the vicinity of the user

Main Success Scenario:

1. System presents **Realm Interface** with another user's object instance present on screen.
2. User selects the object they wish to report from the screen.
3. System presents **Object Selection Context Action Menu**.
4. User selects option to report object.
5. System presents **Object Reporting Interface**.
6. User provides required details for report and submits report.
7. System presents message thanking user for report.

Secondary Scenarios:

5.1: User cancels report:

1. User closes **Object Reporting Interface**.
2. System returns to **Realm Interface**.

3.1: User cancels object selection:

1. User closes **Object Selection Context Action Menu**.
2. Main scenario resumes from step 2.

Success Postcondition: The user is informed that their report will be investigated. The system initiates a report review process.

UC15 Add reaction to an existing AR object instance
Actor: General User
Pre-condition: User is logged in to the app **AND** has navigated to the **Realm Interface** (RI-FR1) **AND** there is an AR object placed in the vicinity of the user

Main Success Scenario:

1. System presents **Realm Interface** with an object instance present on screen.
2. User selects the object they wish to react to from the screen.
3. System presents **Object Selection Context Action Menu**.
4. User selects option to react to object instance.
5. System presents **Reaction Selection Menu**.
6. User selects desired reaction.
7. System returns to **Object Selection Context Action Menu** with user reaction displayed.

Secondary Scenarios:

3.1: User cancels object selection:

1. User closes **Object Selection Context Action Menu**.
2. Main scenario resumes from step 2.

Success Postcondition: The user's reaction is visible to any user who selects the AR object instance.

UC16 Add other user's placed AR object to Inventory
Actor: General User
Pre-condition: User is logged in to the app **AND** has navigated to the **Realm Interface** (RI-FR1) **AND** there are other users' AR objects placed in the vicinity of the user

Main Success Scenario:

1. System presents **Realm Interface** with another user's object instance present on screen.
2. User selects the object they wish to add to their inventory from the screen.
3. System presents **Object Selection Context Action Menu**.
4. User selects option to add object to inventory.

5. System returns to **Realm Interface** with indication that object has been added to Inventory.

Secondary Scenarios:

4.1: User inventory is full:

1. System returns to **Realm Interface** with indication that Inventory is full.

3.1: User cancels object selection:

1. User exits **Object Selection Context Action Menu**.
2. System returns to **Realm Interface**.

Success Postcondition: The user's desired object is available in the user's inventory for preview and placement.

UC17 Create a Sub-Realm
Actor: General users
Precondition: User is logged into the app.

Main Success Scenario:

1. User navigates to the "Create Sub-Realm" page from the app's main interface.
2. System presents the Sub-Realm Creation Page.
3. User provides a name and description for the sub-realm.
4. System presents the option to "Add users to sub-realm."
5. User selects the option to add users.
6. System displays the user's friend list.
7. User selects the friends they want to add to the sub-realm.
8. User confirms that they are done adding friends.
9. User confirms creation of the sub-realm.
10. System saves the sub-realm with the provided information and friends list.
11. System returns to the Realm interface, displaying the newly created sub-realm.

Secondary Scenarios:

User cancels the creation before confirming:

1. Main scenario 1-8
2. User selects the option to cancel sub-realm creation.
3. System discards the sub-realm creation process and returns to the Realm interface.

User reselects friends:

1. Main scenario 1-7
2. User selects the option to go back to the friends list.
3. Main scenario resumes from step 6.

User does not add any friends:

1. Main scenario 1-5
2. User selects the option to skip adding friends.
3. Main scenario resumes from step 9.

Success Postcondition: A new sub-realm is created and linked to the user's account with the chosen settings and added friends.

UC18 Leave a Sub-Realm
Actor: General users
Precondition: User is logged into the app AND is a member of the sub-realm.

Main Success Scenario:

1. User navigates to the sub-realm and opens the Sub-Realm Settings.
2. System presents the Sub-Realm Settings Menu.
3. User selects the "Leave Sub-Realm" option.
4. System prompts the user to confirm the action.
5. User confirms that they wish to leave the sub-realm.
6. System removes the user from the sub-realm.
7. System returns the user to the Realm Interface, with the sub-realm no longer visible.

Secondary Scenarios:

User cancels leaving the sub-realm:

1. Main scenario 1-4
2. User selects the option to cancel leaving the sub-realm.
3. System returns to the Sub-Realm Settings without making any changes.

Success Postcondition: The user is no longer part of the sub-realm, and the sub-realm is no longer accessible or visible to the user.

UC19 Join an ExistingSub-Realm
Actor: General users
Precondition: User is logged into the app AND has received an invite to the sub-realm.

Main Success Scenario:

1. User receives a notification of a sub-realm invitation.

2. System presents the sub-realm invitation details.
3. User selects the sub-realm they wish to join.
4. System presents a confirmation prompt.
5. User confirms their request to join the sub-realm.
6. System adds the user to the selected sub-realm.
7. System returns to the Realm Interface, now displaying the newly joined sub-realm.

Secondary Scenarios:

Sub-realm is invite-only, and approval is required:

1. Main scenario 1-5
2. System notifies the user that their request to join is pending approval.
3. User must wait for an admin's approval before being added to the sub-realm.

User cancels the request before joining:

1. Main scenario 1-4
2. User selects the option to cancel the request to join.
3. System returns to the main interface, and no further action is taken.

Success Postcondition: The user is successfully added to the sub-realm and can now access it from within the Realm Interface.

UC20 Add a Friend

Actor: General users

Precondition: User is logged into the app AND the friend has created an account on the app.

Main Success Scenario:

1. User navigates to the "Add Friend" option in the app's interface.
2. System presents the "Add Friend" search interface.
3. User searches for the friend's username.
4. System displays matching search results.
5. User selects the friend's username from the results.
6. User sends a friend request.
7. System sends the friend request and notifies the friend.
8. System returns the user to the friend management screen.

Secondary Scenarios:

Friend accepts or rejects the request:

1. Main scenario 6-7
2. Friend receives the request and either accepts or rejects it.
3. System notifies the user if their request was accepted or rejected.

User cancels the friend request before sending:

1. Main scenario 1-5
2. User cancels the request, and no action is taken.
3. System returns the user to the previous screen.

User cancels the friend request after sending:

1. Main scenario 6-7
2. User cancels the sent request before the friend responds.
3. System withdraws the request.

Success Postcondition: The user has successfully added the friend as a connection, and the friend now appears in their friends list.

UC21 Add a Friend to aSub-Realm

Actor: General users

Precondition: User is logged into the app AND is a member of the sub-realm AND has the friend already added to their friend list.

Main Success Scenario:

1. User navigates to the desired sub-realm from the Realm Interface.
2. System presents the sub-realm view.
3. User opens theSub-Realm Settings.
4. System presents theSub-Realm Settings Menu.
5. User selects the option to "Add a Friend toSub-Realm."
6. System displays the user's friend list.
7. User selects the friend they wish to add to the sub-realm.
8. System presents a confirmation prompt.
9. User confirms adding the friend.
10. System adds the friend to the sub-realm.
11. System returns to theSub-Realm Interface, now displaying the updated member list.

Secondary Scenarios:

User cancels adding the friend:

1. Main scenario 1-8
2. User selects the option to cancel adding the friend.
3. System discards the action and returns to theSub-Realm Settings Menu.

User selects a friend who is already a sub-realm member:

1. Main scenario 1-7
2. System notifies the user that the selected friend is already a member of the sub-realm.
3. User can return to the friend list and select another friend.

User reselects a friend:

1. Main scenario 1-7
2. User decides to reselect a different friend from the list.
3. System returns to the friend selection interface.
4. Main scenario resumes from step 7.

Success Postcondition: The user has successfully added the friend to the sub-realm, and the updated member list is displayed in theSub-Realm Interface.

UC22 Remove a Friend from aSub-Realm

Actor: General users

Precondition: User is logged into the app AND is a member of the sub-realm AND has the friend added to the sub-realm.

Main Success Scenario:

1. User navigates to the desired sub-realm from the Realm Interface.
2. System presents the sub-realm view.
3. User opens theSub-Realm Settings.
4. System presents theSub-Realm Settings Menu.
5. User selects the option to "Remove a Friend fromSub-Realm."
6. System displays the current list of members in the sub-realm.
7. User selects the friend they wish to remove.
8. System presents a confirmation prompt.
9. User confirms the removal.
10. System removes the friend from the sub-realm.
11. System returns to theSub-Realm Interface, displaying the updated member list without the removed friend.

Secondary Scenarios:

User removes the friend from settings:

1. User navigates to the settings page in the app.
2. System presents the settings display.
3. User navigates to the sub-realm settings.
4. System displays a list of sub-realms.

5. User selects the desired sub-realm.
6. Main scenario 4-11

User cancels removing the friend:

1. Main scenario 1-8
2. User selects the option to cancel the removal.
3. System discards the action and returns to theSub-Realm Settings Menu.

User attempts to remove a friend who is not in the sub-realm:

1. Main scenario 1-7
2. System notifies the user that the selected friend is not a member of the sub-realm.
3. User returns to the member list to reselect another friend.

User reselects a friend to remove:

1. Main scenario 1-7
2. User decides to reselect a different friend from the list.
3. System returns to the member selection interface.
4. Main scenario resumes from step 7.

Success Postcondition: The selected friend is successfully removed from the sub-realm and no longer has access to the sub-realm content.

UC23 Create an Account
Actor: General users
Precondition: User has installed the app AND is on the login screen without an existing account.

Main Success Scenario:

1. User selects the "Sign Up" option from the login screen.
2. System presents the account creation form, asking for details such as email, username, and password.
3. User fills in the required information.
4. System performs validation to ensure the information is complete and valid (e.g., unique username, valid email format).
5. User submits the form.
6. System creates the account and stores the user's information securely.
7. System logs the user into the app and navigates to the home screen.

Secondary Scenarios:

User cancels account creation:

1. Main scenario 1-4

2. User cancels the sign-up process before submitting.
3. System discards the entered information and returns the user to the login screen.

Account creation fails due to existing email/username:

1. Main scenario 1-5
2. System detects that the email or username is already in use.
3. System presents an error message, and the user must provide new information.
4. Main scenario resumes from step 2-7.

Success Postcondition: A new account is created, and the user is logged into the app with their new credentials.

UC24 Login to an Account
Actor: General users
Precondition: User has an existing account on the app.

Main Success Scenario:

1. User navigates to the "Login" page.
2. System presents the login form, asking for email or username, and password.
3. User enters their email or username, and password.
4. System performs validation to check if the credentials match an existing account.
5. User submits the form.
6. System logs the user into the app and navigates to the home screen.

Secondary Scenarios:

Invalid credentials entered:

1. Main scenario 1-5
2. System detects that the credentials are invalid (e.g., incorrect password).
3. System presents an error message, allowing the user to try again.

User resets password:

1. Main scenario 1-4
2. User selects the "Forgot Password" option.
3. System initiates the password recovery workflow, and the user must follow the steps to reset their password.

User cancels login process:

1. Main scenario 1-4
2. User cancels the login process before submitting.

3. System returns to the initial screen without logging in.

Success Postcondition: The user is successfully logged into the app and gains access to their account and personalized settings.

UC25 Edit Profile
Actor: General users
Precondition: User is logged into the app AND has an existing profile.

Main Success Scenario:

1. User navigates to the "Settings" page from the app's main interface.
2. System presents the Settings Menu with the list of settings categories, including "Profile Settings."
3. User selects the "Profile Settings" option.
4. System displays editable fields, such as username, bio, password, and other personal details.
5. User modifies the desired fields.
6. User confirms the changes.
7. System saves the updated profile information.
8. System returns the user to the Settings page with a confirmation that the profile has been updated.

Secondary Scenarios:

User cancels profile editing:

1. Main scenario 1-5
2. User selects the option to cancel editing before confirming.
3. System discards the changes and returns the user to the Settings page without applying any updates.

Invalid information provided:

1. Main scenario 1-6
2. System detects that the updated information (e.g., username) is invalid or already taken.
3. System presents an error message and prompts the user to enter valid information.

Success Postcondition: The user's profile is successfully updated and reflects the new information.

UC26 Act on reported AR object
Actor: Admins
Pre-condition: Admin is logged into the app **AND** Admin has navigated to the **Admin Interface**

Main Success Scenario:

1. Admin navigates to the "AR Object Reports" panel (AI-FR2.1)
2. System shows a list of app AR objects that have been reported
3. Admin selects an object
4. System gives more detailed information of AR object and allows Admin to open an AR object inventory preview (IV-FR6)
5. System also provides decision options on whether to keep the object or remove it
6. Admin opens AR object preview
7. System shows preview of AR object
8. Admin views AR object preview
9. Admin closes AR object preview
10. System once again shows the object detailed information and decision options
11. Admin reviews the information and decides to keep the AR object
12. System keeps AR object
13. System removes user report of AR object
14. System brings admin back to "AR Object Reports" panel

Secondary Scenarios:

5.1: Admin does not open AR object preview
1. Go to step **11**

11.1: Admin reviews the information and decides to remove the object
1. System removes object
2. Go to step **13**

Success Postcondition: An AR object report is resolved (AR object is kept or removed)

UC27 Act on reported AR object
Actor: Admins
Pre-condition: Admin is logged into the app **AND** Admin has navigated to the **Admin Interface**

Main Success Scenario:

1. Admin navigates to the "Organization User Requests" panel (AI-FR2.2)
2. System shows a list of Organization users requests
3. Admin selects a request
4. System gives more details about a request

5. System also provides decision options on whether to accept or decline a request
6. Admin reviews information and approves the request
7. System activates the user as one of the Organization users
8. System notifies the user that their request has been approved
9. System removes organization user request
10. System brings admin back to "Organization User Requests" panel

Secondary Scenarios:

6.1: Admin reviews information and denies the request

1. System notifies the user that their request has been denied and gives a reason why
2. Go to step **9**

Success Postcondition: An Organization users account request is resolved (approved or denied).

UC28 Unfriend a Friend
Actor: General users
Precondition: User is logged into the app AND the friend is currently on the user's friend list.

Main Success Scenario:

1. User navigates to friend list from the app's main interface.
2. System presents the list of friends.
3. User selects a friend.
4. System shows list of actions with the friend.
5. User selects the "Unfriend" option.
6. System presents confirmation prompt.
7. User confirms to unfriend the person.
8. System removes the friend from the user's friend list.
9. System returns the user to the friend list.

Secondary Scenarios:

User cancels unfriending the friend:

1. Main scenario 1-7
2. User selects the option to cancel.
3. System discards action and returns to the friend list.

User reselects a friend to unfriend:

1. Main scenario 1-6

2. User decides to reselect a different friend from the list.
3. System returns to the friend list interface.
4. Main scenario resumes from step 3.

Success Postcondition: The selected friend is successfully unfriended and removed from the user's friend list.

3.4 Quality of Service

Non-functional requirements pertaining to the quality of the product

3.4.1 Performance

QS-P1. The map rendering should be completed quickly

QS-P2. The inventory should load quickly when viewing the user's profile

QS-P3. The real-time render should be delayed from the actual scan by at most 1 second.

QS-P4. The object generation should not take longer than 30 seconds.

QS-P5. The system must provide fallback modes for rendering AR objects on low-performance devices to ensure accessibility for all users.
Rationale: This allows for a wider user base as hardware is less of a restriction.
Fit Criteria: *All* tested devices with the required sensors report no performance issues.

3.4.2 Usability

QS-U1. The system should be localized in most common languages.
Rationale: The app should be usable by people in non-english speaking countries.
Fit Criteria: The app can be set to display all text in the 5 most spoken languages worldwide.

QS-U2. The interfaces and workflows in the app should be intuitive, easy to learn, and satisfying to use.
Rationale: The satisfaction the user feels when using the app is a large component of their experience.
Fit Criteria: Survey results show most users rate the app as highly intuitive and highly satisfying.

3.4.3 Security

QS-SC1. The system should follow an encryption standard for all communications
Rationale: This measure eliminates many security vulnerabilities
Fit Criteria: It is not possible to extract information from a man in the middle attack

QS-SC2. The system should avoid directly or indirectly leaking private user data to other users.
Rationale: Giving other users indication of the user's email, name, or location without the user's consent can be harmful to the user
Fit Criteria: All interfaces presenting user data including AR object instances present only consentually shared data

3.4.4 Safety

QS-SA1. The system should not be excessively distracting.
Rationale: Distracting users from their surroundings may cause lead to collisions or cause users to inadvertently enter restricted areas.
Fit Criteria: Survey results show users rate the app as not distracting.

QS-SA2. No system interfaces should present bright flashes or loud noises
Rationale: Bright flashes and loud noises can cause harm to users with epilepsy
Fit Criteria: No interfaces in the system present bright flashes or loud noises

3.4.5 Reliability

QS-R1. The system databases must be resilient to failure/corruption, able to recover, and return to normal operation easily.
Rationale: unrecoverable database failure would be fatal to the application.
Fit Criteria: fewer than 2% of users report data loss in the case of database failure and recovery.

3.4.6 Availability

QS-A1. The server of the system should be highly available.
Rationale: Downtime prevents users from using the app when they want to, creating a poor experience.
Fit Criteria: System unavailability does not generate complaints for the duration of user testing.

3.5 Compliance

CO1. The project shall comply with the *Personal Information and Electronic Documents Act* (PIPEDA).
Rationale: The Government of Canada requires all companies to follow certain rules regarding the collection, use, and dissemination of personal user information [1].

CO2. The project shall keep records of all in-app purchases and ad revenue for the purposes of yearly tax filing for the period of six years.
Rationale: Corporate taxes must be filed every year and both streams of app income will need to be reported. Businesses must keep records going back six years in the event of an audit [2].

CO3. The app shall comply with the *Google Play* developer policy.
Rationale: All apps published through *Google Play* must first be reviewed by Google for compliance with the developer policy published on their website [3].

CO4. The app shall comply with *App Store* review guidelines.
Rationale: For an app to be approved for dissemination on the *App Store*, the app must be reviewed and approved by Apple in accordance with the acceptance criteria published on their website [4].

CO5. The system shall comply with the *Web Content Accessibility Guidelines (WCAG) 2.1 Level AA* and the *Accessibility for Ontarians with Disabilities Act (AODA)*.
Rationale: Ensuring compliance with these accessibility standards allows users with disabilities to fully engage with the system and meets legal requirements for accessibility in Ontario.

3.5.1 Accessibility Compliance

QS-AC1. The system shall comply with established accessibility standards, including **WCAG 2.1 Level AA** and the **Accessibility for Ontarians with Disabilities Act (AODA)**.
Rationale: Ensuring compliance with accessibility standards allows users with disabilities to fully engage with the system.
Fit Criteria: The app shall pass accessibility testing for compliance with WCAG 2.1 Level AA and AODA regulations.

QS-AC2. The system shall offer a high-contrast mode and customizable text size for users with visual impairments.
Rationale: High contrast and adjustable text improve readability for users with low vision.
Fit Criteria: The app shall provide an accessibility setting to adjust contrast and font size.

QS-AC3. The system shall avoid flashing visuals and rapid animations that could trigger seizures, as specified in **QS-SA2**.
Rationale: Flashing content can cause seizures in users with photosensitive epilepsy.
Fit Criteria: The app shall pass seizure safety tests and restrict flashing effects.

QS-AC4. The system shall allow users to modify UI settings, including color themes, font scaling, and motion reduction.
Rationale: Users with cognitive disabilities or visual impairments may require a more adaptable interface.
Fit Criteria: The app shall provide an accessibility menu with UI customization options.

3.6 Design and Implementation

Non-functional requirements pertaining to the design and implementation of the product

3.6.1 Installation

DI-I1. The app shall be installable on *Android* and *iOS* devices from their respective app stores.
Rationale: Users are accustomed to downloading their apps from their device app store and should not be required to navigate to a 3rd party app store. They also should not have to change OS settings in order to download the app.

DI-I2. The app shall not require any additional installation steps beyond those required from within the target device app store.
Rationale: Users may be dissuaded from downloading the app if the installation process is too cumbersome compared to other apps.

3.6.2 Distribution

DI-D1. The app shall be distributed on any mobile devices running iOS 16.0+ or Android 12+
Rationale: The app should be available to as many people as possible while at the same time making the development easier by not having to keep old operating system versions supported. Versions should be supported for at least a couple years.

DI-D2. The app shall be available in Canada and the USA.
Rationale: Due to legal considerations in different countries, the focus for this app should be the country this project is based out of, Canada, and the USA since they have a larger population with similar laws.

DI-D3. The app shall have a recommended age requirement of 16+.
Rationale: Users may be exposed to content not suitable for really young kids so an age requirement should be recommended.

DI-D4. The system shall store all user data within North America.
Rationale: Data should be located in a jurisdiction close to home and in a reputable country to reduce privacy concerns of foreign state actors viewing user data.

3.6.3 Maintainability

DI-M1. All internal APIs of the system must provide useful error messages in the case of system failures.
Rationale: Helps with finding and fixing bugs in the system.
Fit Criteria: API error messages help identify error source in 90% of cases.

3.6.4 Reusability

DI-R1. Common elements in the user interfaces of the system will be created as reusable components.
Rationale: Keeps the look of the app consistent across interfaces, and speeds up development.
Fit Criteria: All UI elements appearing in more than one place are derived from common components.

3.6.5 Portability

DI-P1. The app shall be developed using a cross-platform mobile platform that can build iOS and Android applications.
Rationale: To reach as many potential users as possible, the app should be available on the two major mobile operating systems.

DI-P2. The app shall have a common codebase that only differs in configuration files for different target operating systems.
Rationale: For ease of development on the different mobile operating systems, there should be no extra consideration for the differences between native app implementations that are not handled by the cross-platform framework.

3.6.6 Cost

The Project is constrained to a budget of $700.

3.6.7 Deadline

Refer to Figure 2, the apportioning table, for the planned deadlines for the requirements outlined in this document. Overall project deadlines are outlined in section eight of the Development Plan.

3.6.8 Proof of Concept

As part of the proof of concept for this product, the following functional requirements will be implemented:

- RI-FR1
- RI-FR3
- RI-FR4
- OP-FR1
- OP-FR2
- OS-FR1
- OS-FR5

4 Verification

See the Verification and Validation Plan.

5 Appendixes

Supplemental documentation

5.1 Activity Diagrams

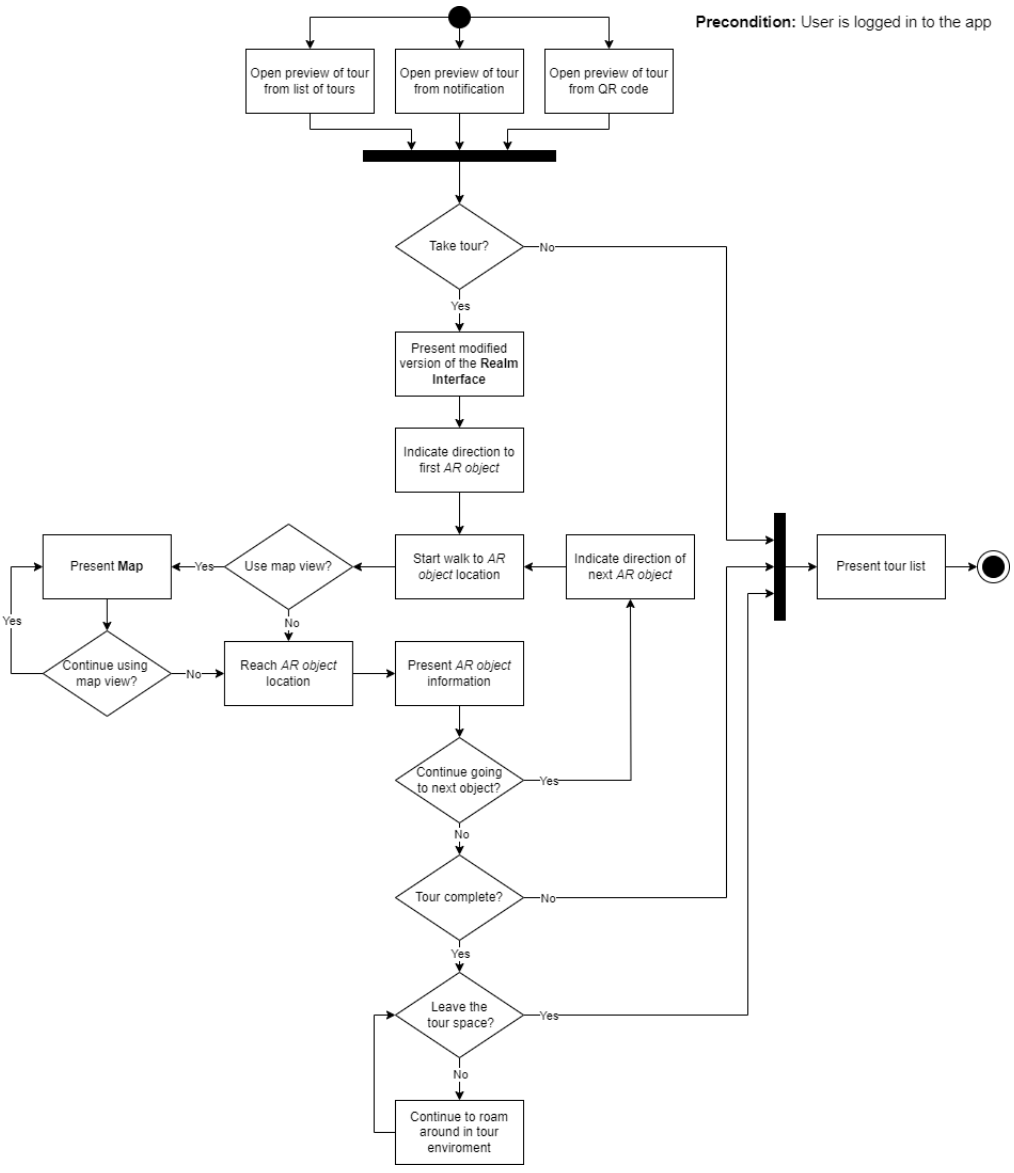


Figure 2: UC4

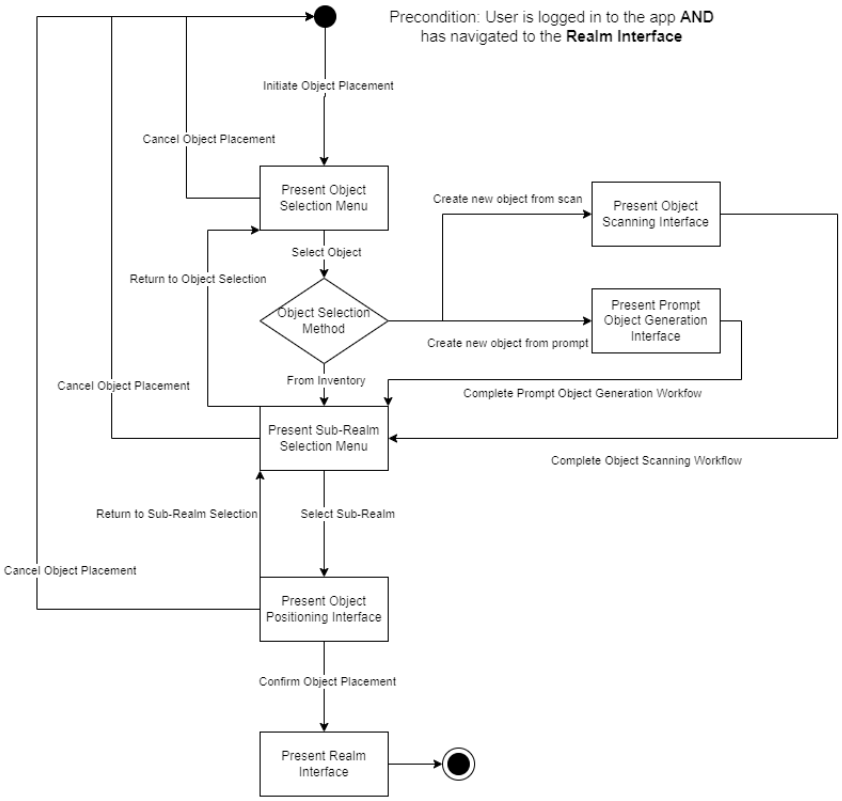

Precondition: User is logged in to the app AND has navigated to the Realm Interface
Initiate Object Placement
Cancel Object Placement
Present Object Selection Menu
Create new object from scan
Present Object Scanning Interface
Select Object
Return to Object Selection
Object Selection Method
Present Prompt Object Generation Interface
Create new object from prompt
Cancel Object Placement
From Inventory
Complete Prompt Object Generation Workfow
Present Sub-Realm Selection Menu
Complete Object Scanning Workflow
Return to Sub-Realm Selection
Select Sub-Realm
Cancel Object Placement
Present Object Positioning Interface
Confirm Object Placement
Present Realm Interface

Figure 3: UC7