Deliverable D2.1 System Design and Specification

Deliverable type	R — Document, report
Dissemination level	PU – Public
Due date (month)	M4
Delivery submission date	31.05.2023
Work package number	2
Lead beneficiary	IOTIQ

This project has received funding from the Horizon Europe Framework Programme of the European Union under grant agreement No. 101094428

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Document Information

Project number	101094428	Acronym	CULTURATI
Project name	Customized Games and Routes For Cultural Heritage and Arts		
Call	HORIZON-CL2-2022-HERITAGE-01		
Topic	HORIZON-CL2-2022-HERITAGE-01-02		
Type of action	HORIZON-RIA		
Project starting date	1 February 2023	Project duration	36 months
Project URL	http://www.culturati.eu		
Document URL	https://culturati.eu/deliverables/		

Deliverable number	D2.1			
Deliverable name	System Design and Specification			
Work package number	2			
Work package name	WP2 – System Development and Evaluation			
Date of delivery	Contractual	M4	Actual	M4
Version	Version 1.0			
Lead beneficiary	IOTIQ			
Responsible author(s)	Metin Tekkalmaz, IOTIQ, metin@iotiq.de			
Reviewer(s)	Angel Lagares, NIMBEO alagares@nimbeo.com Eda Gürel, Bilkent Üniversitesi Vakıf, eda@tourism.bilkent.edu.tr Arzu Sibel İkinci,Bilkent Üniversitesi Vakıf, aikinci@bilkent.edu.tr			

Short Description	This deliverable is intended to be the reference document for the development of the system of CULTURATI. It centralizes the requirement that this module will satisfy and the decision processes that have led to these requirements. To remain useful, it is meant to be a live document, firstly released early in the project as a draft, and updated over the project's course.

History of Changes

Date	Version	Author	Remarks
24 May 2023	Draft 0.1	Metin Tekkalmaz	First version
06 June 2023	Final 1.0	Metin Tekkalmaz	Revised after review

Executive Summary

This deliverable outlines the requirements, architecture, and specifications of the software system of CULTURATI. It serves as a blueprint for developers, stakeholders, and the project team to understand and implement the system effectively.

The document starts with a short introduction, providing an overview of the system's purpose, goals, and key stakeholders. It also defines the scope and boundaries of the system, clarifying what is included and what is not.

The document covers the functional requirements describing the system's desired behavior and capabilities. In particular, it includes use cases, scenarios, and user stories to illustrate the expected functionalities and workflows. The document also consists of design patterns and approaches.

The system architecture section describes the high-level structure and components of the system. It identifies the major subsystems, their relationships, and the overall flow of data and control. Accordingly, it includes diagrams, such as block diagrams, flowcharts, and UML diagrams, to visualize the system's structure and interactions.

Table of Contents

1. Purpose of the Document

This document aims to outline the design and functionality of the system of CULTURATI whose primary objective is to provide customized information and navigation to visitors with an easier and more efficient way of experiencing and navigating a museum/heritage site. CULTURATI is also a global content platform for CCIs (Cultural and Creative Industries) to co-create content. This document serves as a comprehensive guide for the development team, stakeholders, and relevant parties involved in the project. It defines the goals, requirements, architecture, and components of the application, ensuring a clear understanding of the system's design principles and capabilities.

The goals of the system can be outlined as;

➔ Enhance the Museum and Cultural Heritage Site Experience: The system provides entertaining features such as customized games, Q&A games, and treasure hunt-like games, along with curated routes to engage and attract users to cultural heritage and arts.

➔ Serve as a Global Content Platform: The system functions as a global content platform for CCIs, enabling users to share information about items in museums and cultural heritage sites. It can be utilized by venue or site-based CCIs including museums, art galleries, art fairs, biennial events, palaces, castles, historic town centers, cities, etc. CCI professionals like artists, authors, handcraft designers, freelancers, painters, film and animation producers, etc., as well as citizens, can share knowledge related to cultural and artistic merit.

➔ Enable Personalization: The software should allow visitors to customize their museum experience based on their interests, preferences, and accessibility needs. It should offer personalized recommendations, guided tours, and the ability to bookmark favorite exhibits or create personalized collections.

➔ Provide Contextual Information: The application should deliver comprehensive and engaging content related to the museum exhibits, including multimedia presentations, historical background, interesting facts, and interactive elements, enabling visitors to gain a deeper understanding and appreciation of the artwork or artifacts.

➔ Streamline Navigation: The application/system should facilitate efficient navigation within the museum by offering intuitive maps, directions, and personalized itineraries tailored to visitors' preferences and interests. It should provide real-time updates on exhibit availability, events, and crowd levels to optimize visitor flow.

➔ Enhance Visitor Experience: The application aims to enrich the site visiting experience by providing users with a user-friendly and intuitive interface that allows them to explore the exhibits, access relevant information, and engage with interactive multimedia content.

➔ Facilitate Accessibility: The application should be designed with inclusivity in mind, ensuring accessibility for individuals with disabilities. It should provide features such as options for adjusting the user interface based on specific accessibility requirements.

The intended users of the software application include:

➔ Site Visitors: Individuals of all ages and backgrounds who visit the site and seek an enhanced experience through the use of technology.

➔ Museum/Heritage Site Staff: Employees responsible for managing exhibits, events, and visitor services, who will utilize the application to provide real-time information, updates, and support to visitors.

➔ Content Creators: Both institutions and individuals involved in Cultural and Creative Industries (CCIs) as well as citizens can all contribute and share their knowledge related to cultural and artistic experiences.

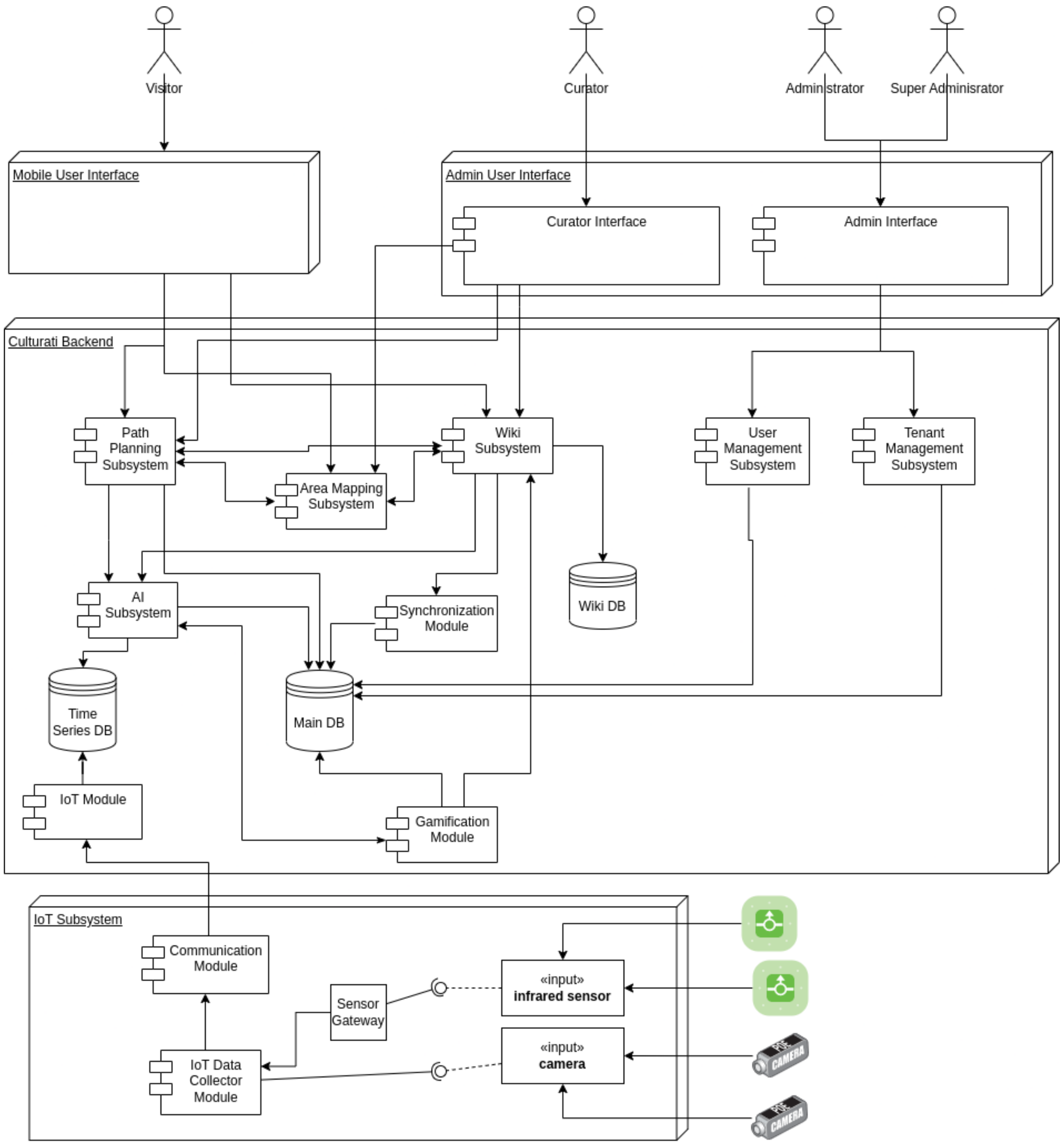
The scope of this document encompasses the high-level architecture, system components, data flow, user interface design, and key functionality of the software application. It outlines the system requirements, integrations with external systems, and any external dependencies necessary for its operation.

By providing a comprehensive overview of the system design, this document aims to ensure a common understanding of the goals, functionality, and expectations for the software application that will ultimately result in an improved and more enjoyable site-visiting experience for all users.

1.1 Browser Limitations and Design Decisions Disclaimer

The purpose of this document is to provide guidance on the design and specifications of our application, including its features, functionalities, and visual elements. However, it is essential to acknowledge that the actual implementation of these design decisions may be influenced by constraints imposed by web browsers. Browser capabilities and standards can vary, potentially impacting the precise realization of specific design elements. Consequently, certain design features or interactions may require adjustments, modifications, or even reimagining to ensure compatibility across diverse platforms and browsers. These limitations may arise from variations in CSS support, JavaScript functionality, rendering engines, mobile device permissions, or evolving web standards.

2. System Architecture



2.1 Navigational Subsystem (Path Planning Subsystem)

2.1.1 General Information

The path planning subsystem is responsible for generating efficient paths for the Culturati application users to navigate the area. It utilizes the following subsystems; the content management subsystem, the AI subsystem, and the IOT subsystem. This way the path planning subsystem takes into account the user interest data and sensor data of the area.

2.1.2 Use Cases

2.1.2.1 Start tour

Actors	Visitor	
Goals	To initiate a customized tour.	
Related use cases	**Generalization of:** ➔ Start tour, user created or system generated	
Steps	*Actor actions*	*System responses*
	1. Click on the 'Start Tour' option.	2. Display the visual representation of the route on the map view that is created based on the selection criteria of the user and crowd information. 3. Display the first stop of the route in the route stops view.
Post conditions	The system is in the *active route* mode.	

2.1.2.2 Travel along on a route

Actors	Visitor	
Goals	To provide guidance to the user as they navigate through the area and follow their active route.	
Summary	The system takes the user through a route while informing them of their location and their progress on the route.	
Related use cases	**Extension of:** ➔ Start tour	
Steps	*Actor actions*	*System responses*
	1. Move towards the stop shown on the route view. 2. Mark the stop as *seen*.	
		3. Scroll the stops so that the stop that has been marked as *seen* disappears. 4. Update the location of the user in the map view to show movement. 5. Recalculate the next stop depending on the crowd information.
	6. Go to #1 and loop until there are no more stops left.	
		7. Notify the user that the route is finished. 8. End *active route* mode.
Post conditions	The system exits *active route* mode and clears route stops from *route view*.	

2.1.2.3 View ‘seen’ stops

Actors	Visitor	
Goals	To display stops that have been marked as *seen.*	
Preconditions	The user is on a route, and the system is in *active route* mode.	
Summary	The system keeps the log of where the user is along the route.	
Related use cases	**Extension of:** ➔ Travel along on a route, extension point: step #6	
Steps	*Actor actions*	*System responses*
	1. Open the route stops list	
		2. Scroll to display stops that have been marked as *seen*.
Post conditions	Seen stops are updated.	

2.1.2.4 Create own route

Actors	Visitor	
Goals	To provide the user the ability to create their own route.	
Summary	The user selects the stop they want to visit and the system generates a route.	
Related use cases	**Includes:** ➔ Browse items, exhibits, establishments or facilities ➔ Start tour	
Steps	*Actor actions*	*System responses*
	1. Click on *Create your own route* option.	
		2. Display entities in a list view.
	3. Include use case: Browse items, exhibits, establishments or facilities	
	4. Select an item.	
		5. Add the selected item to the selected stops list.
	6. Loop #4	
	7. Included use case: Start tour	
Post conditions	Post conditions of Start tour use case	

2.1.2.5 Create system generated tour

Actors	Visitor	
Goals	To create a personalized tour for the user.	
Summary	The user answers a questionnaire for the system to learn their interests(categories) and level of knowledge. The system generates a tour that would appeal to the user.	
Related use cases	**Includes:** ➔ Start tour, ➔ Indicate Selection Criteria	
Steps	*Actor actions*	*System responses*
	1. Click *Suggest Tour* option. 3. Included use case: Indicate Selection Criteria 5. Included use case: Start tour	2. Display questionnaire. 4. Notify the user that the system is generating a tour.
Post conditions	Post conditions of Start tour use case.	

2.1.2.6 Indicate Selection Criteria

Actors	Visitor	
Goals	To collect the interest of the user.	
Summary	The user answers questions about their interests.	
Related use cases	**Extended by:** ➔ Exit Indicating Selection Criteria	
Steps	*Actor actions*	*System responses*
	2. Select an answer to the question. 3. Click the *next* option.	1. Display a question with multiple choices. (i.e. interested categories, level of knowledge, how much time the user has) 4. Update the progress 5. Go back to step #1 until there are no questions left. 6. Save the interests(categories) and level of knowledge of the user.
Post conditions	The system has the user's interest information.	

2.1.2.7 Exit Indicating Selection Criteria

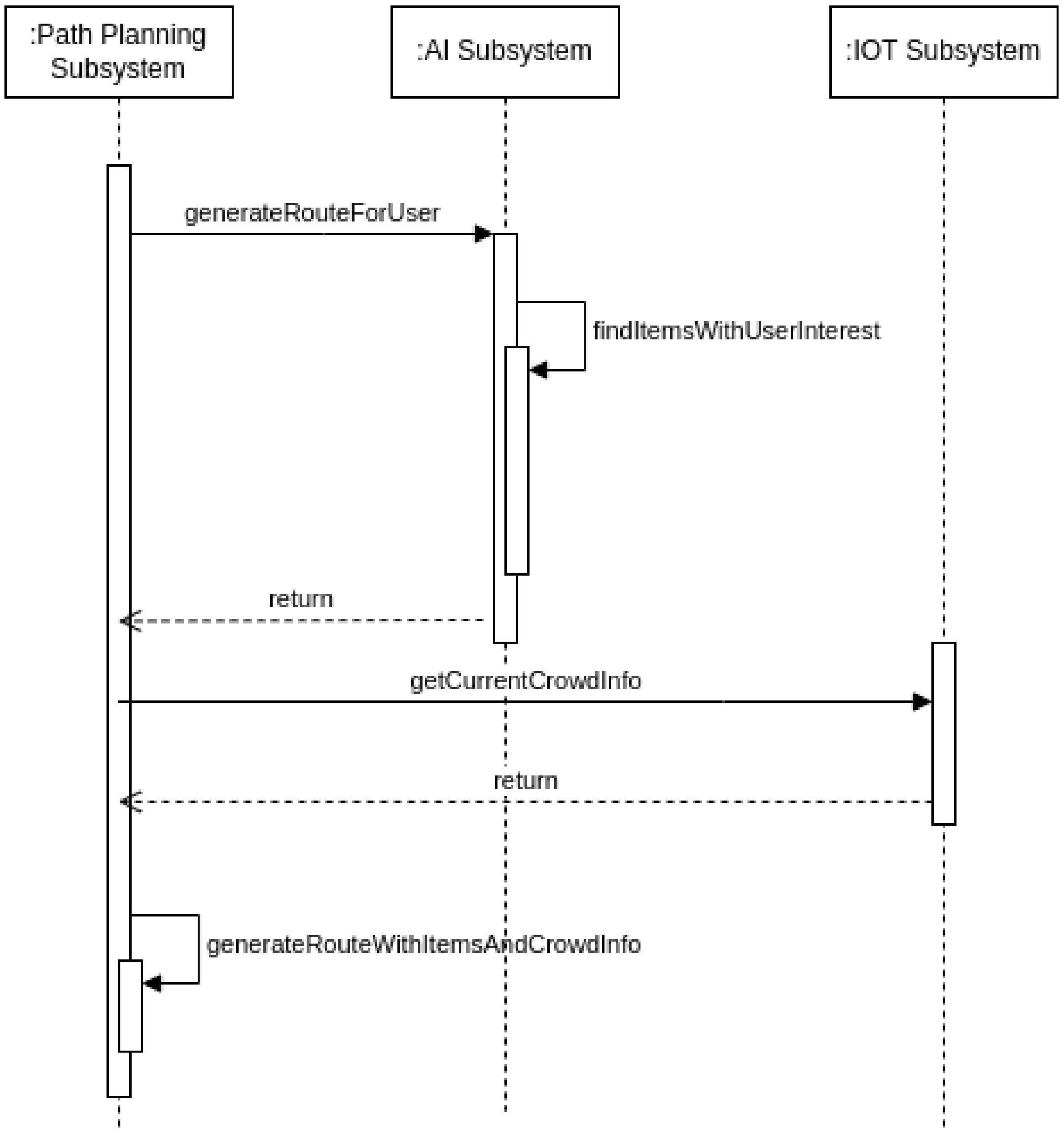
Actors	Visitor	
Goals	To let the user cancel out of the questionnaire.	
Preconditions	Questionnaire is in progress.	
Related use cases	**Extension of:** ➔ Indicate Selection Criteria, extension point: step #2-5	
Steps	*Actor actions*	*System responses*
	1. Click the *back* option.	2. Discard any answers. 3. Redirect to previous screen.

2.1.2.8 Add an item to the active route

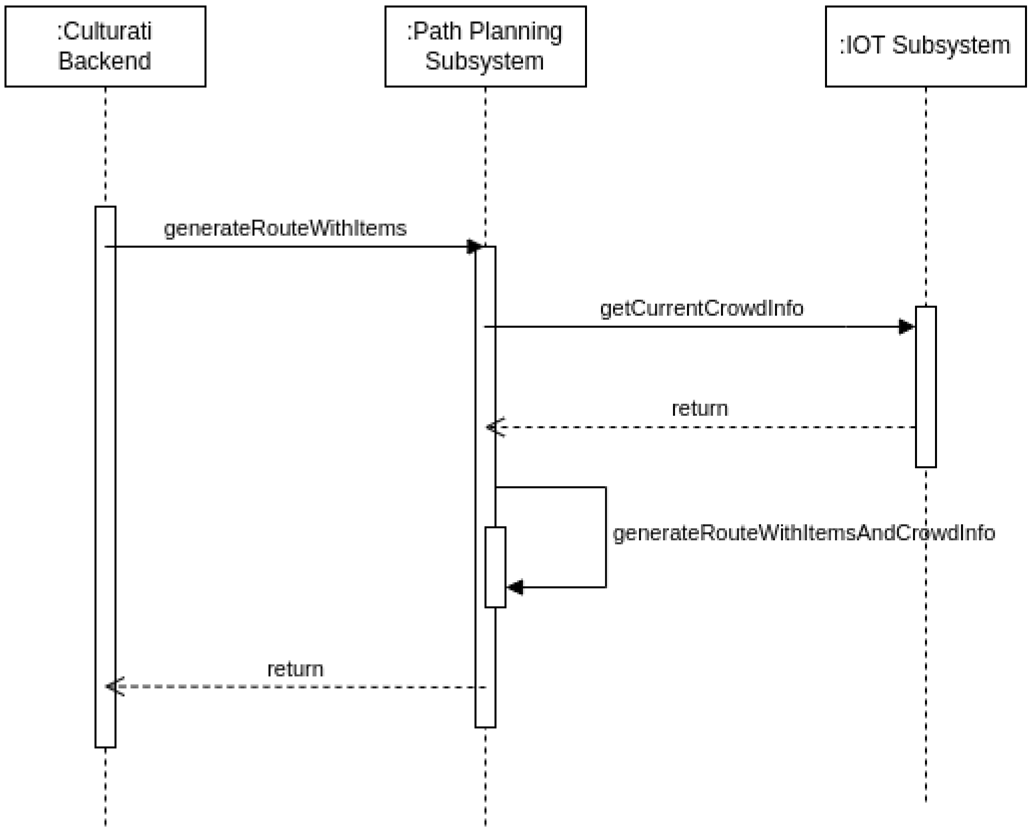
Actors	Visitor	
Goals	To update the active route by adding a new stop.	
Preconditions	The system is in *active route* mode.	
Summary	The user adds a new stop and the system updates the *active route*.	
Related use cases	**Includes:** ➔ View item details	
Steps	*Actor actions*	*System responses*
	1. Included use case: View item details 2. Click *add to route*	3. Add the selected item to the active route. 4. Recalculate the route to incorporate the new stop. 5. Display visual representation of the route in the map view. 6. Display the updated route in the route stops view.
Post conditions	The active route is updated.	

2.1.3 Sequences

2.1.3.1 Generate Route for a User



2.1.3.2 Generate Route for a User With Items



2.2 Area Mapping Subsystem

2.2.1 General Information

The Area Mapping Subsystem is a component of the system that facilitates the creation, management and display of interactive maps. The subsystem allows users to create interactive maps, providing them with a canvas to design and outline the areas of the map. Users have the flexibility to define the boundaries and dimensions of the map and the components of the map to ensure accurate representation. Users can easily place items on the created maps, accurately positioning them within the defined areas. These maps are also used to determine the carrying capacities of each area.

2.2.2 Use Cases

2.2.2.1 View map

Actors	Visitor, Curator	
Goals	To view the site or museum grounds as a map view.	
Steps	*Actor actions*	*System responses*
		1. Display an interactive map of the tenant highlighting items, exhibits, establishments and facilities.

2.2.2.2 Zoom in/out on the map

Actors	Visitor, Curator	
Goals	To view specific areas on the map view.	
Related use cases	**Extension of:** ➔ View map, extension point: step #1	
Steps	*Actor actions*	*System responses*
	1. Pinch the map view.	2. Zoom in or out depending on the direction of the pinch.

2.2.2.3 Pan the map

Actors	Visitor, Curator	
Goals	To navigate and explore the map.	
Related use cases	**Extension of:** ➔ View map, extension point: step #1	
Steps	*Actor actions*	*System responses*
	1. Drag the map view.	
		2. Pan to the direction of the gesture.

2.2.2.4 Create an area map

Actors	Curator	
Goals	To provide a way to the user to generate a visual representation of the area.	
Steps	*Actor actions*	
	1. Select *Create New Area Map* option. 4. Specify the dimensions of the area.	2. Display a blank canvas. 3. Display drawing tools, such as lines, shapes and text annotations.

2.2.2.5 Add a floor plan to an area map

Actors	Curator	
Preconditions	The user has the necessary permissions and role to manage area maps and floor plans.	
Related use cases	**Includes:** ➔ Create an area map	
Steps	*Actor actions*	*System responses*
	1. Included use case: Create an area map 2. Click the *add floor plan* option.	3. Associate the floor plan with the specified area map. 4. Confirm the successful addition of the floor plan and display a success message to the user.
Post conditions	➔ The floor plan is associated with the relevant area map. ➔ The floor plan is visible and accessible to the users with appropriate permissions.	

2.2.2.6 Draw a floor plan

Actors	Curator	
Goals	To visually represent the physical properties of the area map.	
Preconditions	An area map is available to the user.	
Steps	*Actor actions*	*System responses*
	1. Navigate to an area map. 2. Click the *edit* option.	
		3. Display a blank canvas or grid representing the floor plan.
	4. Enter the dimensions of the canvas. 5. Select a drawing tool from the application's toolbar, such as a wall tool, room tool, or text tool. 6. Use the tools to create a visual representation of the floor plan. 7. Save the floor plan assigning a floor name or identifier to it.	
		8. Save the floor plan in the system's storage. 9. Create a relation between the area map entity and the floor plan.
Post conditions	➔ The floor plan is saved and available for further editing or sharing. ➔ The floor plan can be visualized by the users with the appropriate roles.	

2.2.2.7 Add a room to a floor plan

Actors	Curator	
Steps	*Actor actions*	*System responses*
	1. Select the room tool from the drawing tools.	2. Start the room drawing mode.
	3. Click the left mouse button on a corner of a room.	4. Save the clicked point as the first corner of the room.
	5. Click all the corners of a room.	6. Draw a line from the previous corner to the newly clicked corner. 7. Loop #2 as the user adds new corners.
	8. Click the first corner of the room.	9. Save the corner points of the room in the order they were clicked. 10. Show a form for room properties like name, description, room capacity.
	11. Fill in the form. 12. Click the *save* option.	
Post conditions	➔ The room is saved and available for further editing or sharing. ➔ The room can be visualized by the users with the appropriate roles.	

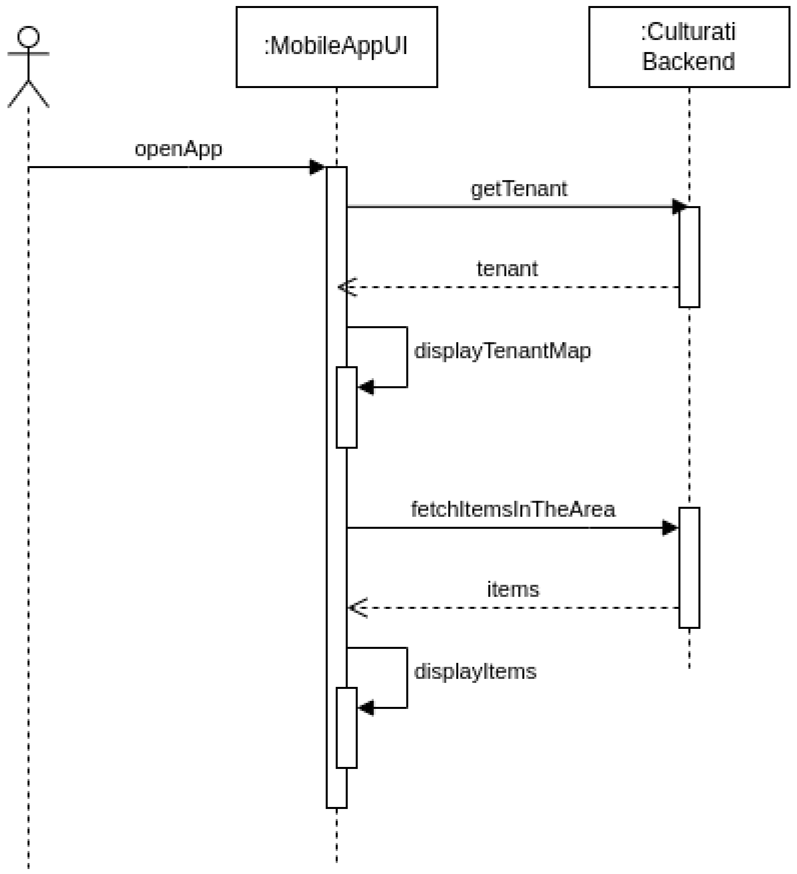
2.2.2.8 Add an item(object) to a floor or room

Actors	Curator	
Steps	*Actor actions*	*System responses*
	1. Select the *add item* option in the drawing tools. 2. Click the location of the item on the floor plan.	
		3. Save the location. 4. Display a search area where the user can search for items in the system.
	5. Search for an item.	
		6. Retrieve and present a list of matching items based on the user's search criteria.
	7. Select an item.	
		8. Validate the selected item and check if it is available for placement on the floor plan.
	9. Select a placement location.	
		10. Verify the placement location and check if it aligns with any restrictions or constraints. 11. Represent the selected item visually on the floor plan at the designated click location.
Post conditions	A new item is placed in the specified floor plan.	

2.2.3 Sequences

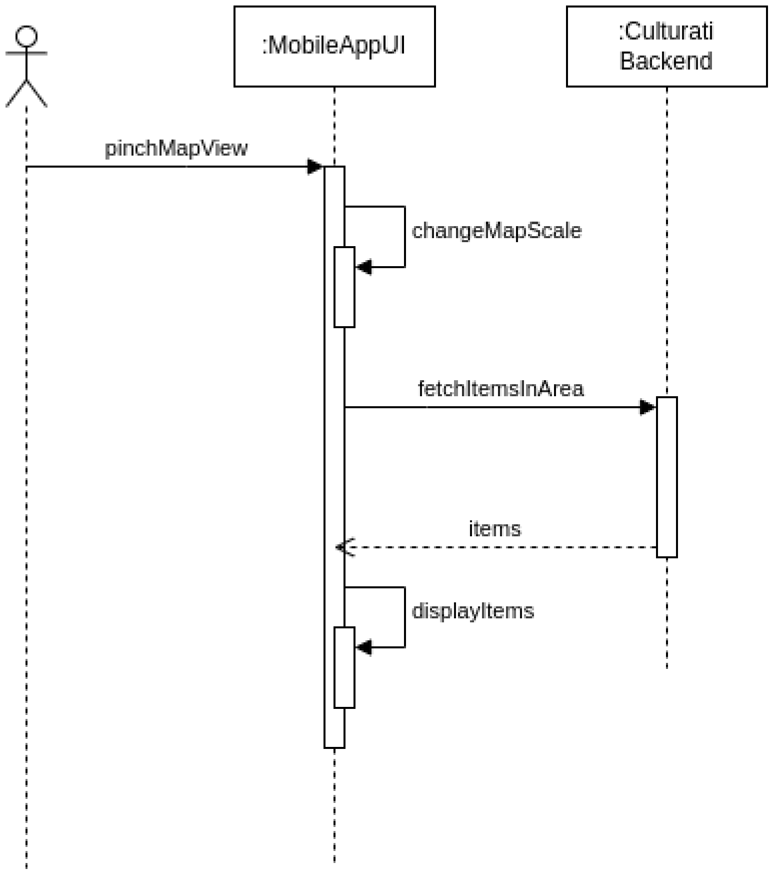
2.2.3.1 View Area Map

Role: Visitor, Curator



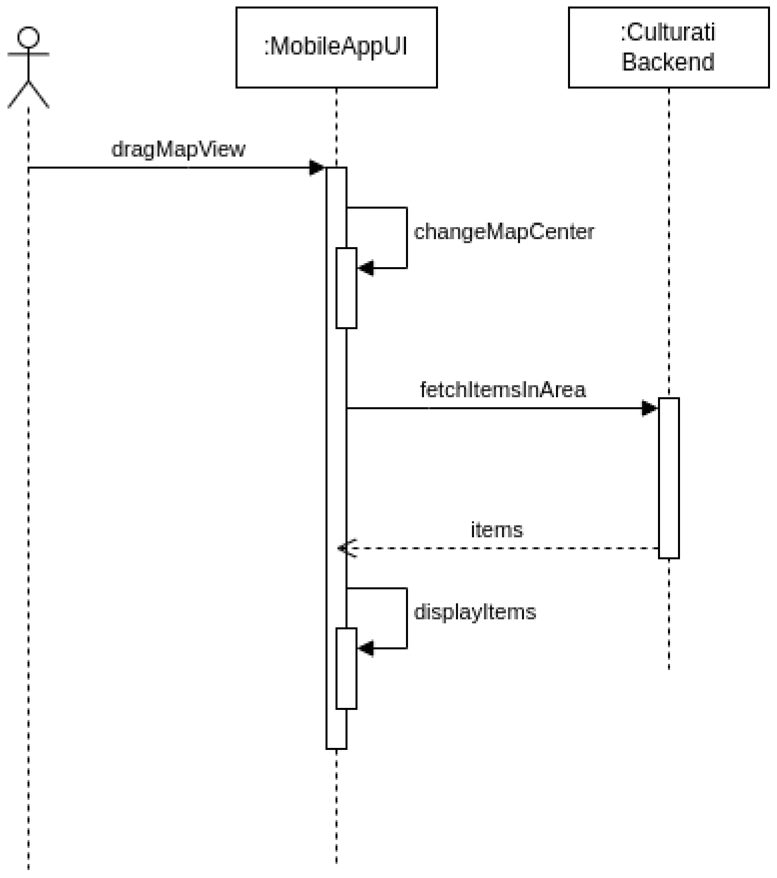
2.2.3.2 Zoom in/out on the map

Role: Visitor, Curator



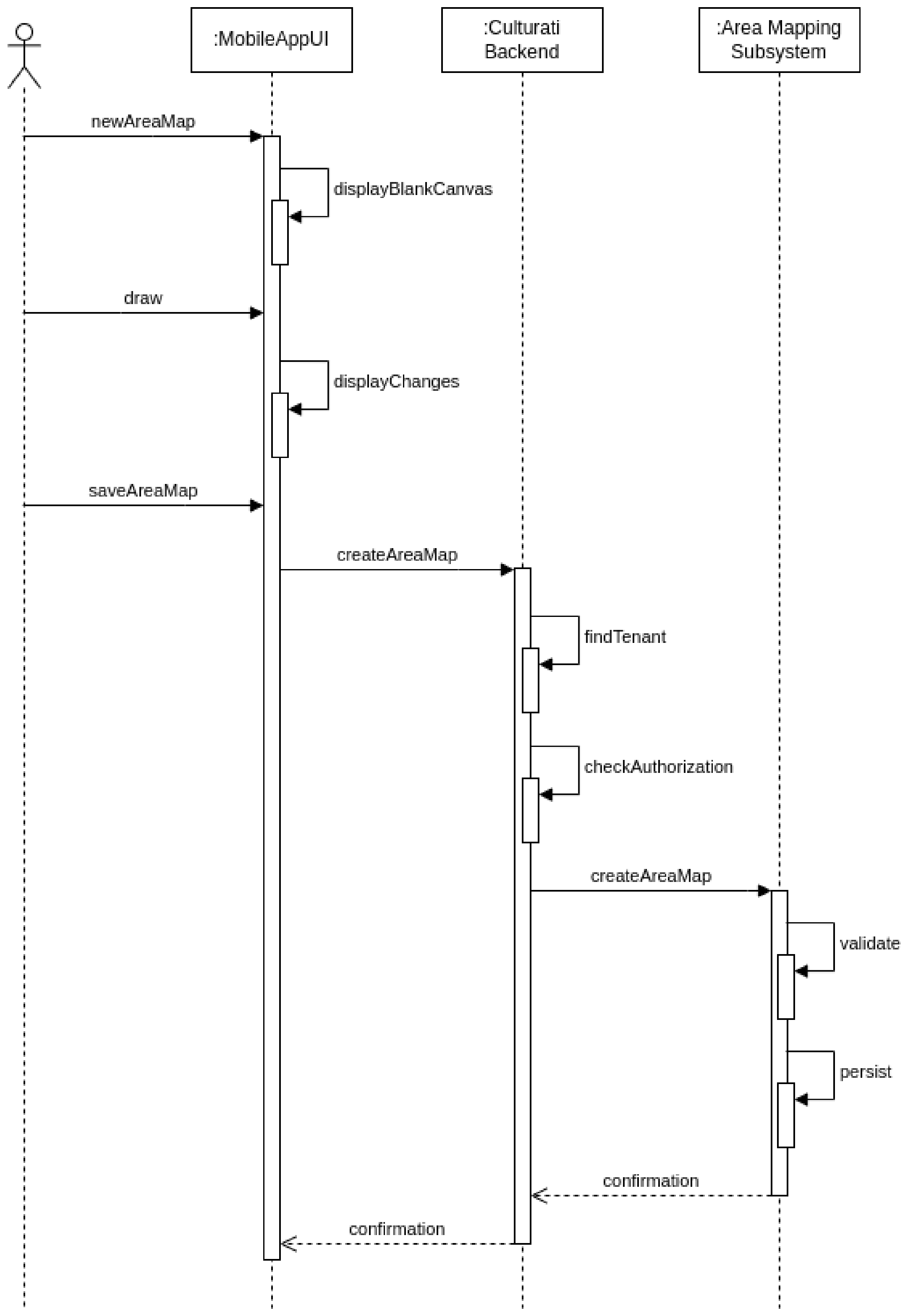
2.2.3.3 Pan the map

Role: Visitor, Curator



2.2.3.4 Create an area map

Role: Curator



2.3 Content Management Subsystem (Wiki)

2.3.1 General Information

The content management subsystem acts as a central repository for all relevant information about the museum, its exhibits, and its layout. It stores data such as exhibit locations, descriptions, multimedia content, accessibility information, and any other relevant details.

2.3.2 Use Cases

2.3.2.1 Browse items, exhibits, establishments or facilities

Actors	Curator, Administrator	
Goals	To provide a way for the user to explore the tenants' catalog and discover the establishments and facilities in the area.	
Summary	The user views system entities (items, exhibits, establishments, and facilities).	
Steps	*Actor actions*	*System responses*
	1. Navigate to the item, exhibit, establishment and facility browsing screen.	
		2. Display items, exhibits, establishments, and facilities in a list view, grouped by a criterion a user with administrator privileges has set.

2.3.2.2 Search for an item

Actors	Visitor, Curator, Administrator
Goals	To provide the user an efficient way of finding specific items or content within the system.
Summary	The user searches system entities with their preferred search method.
Related use cases	**Generalization of:** ➔ Search via query input ➔ Search via unique item identifier

2.3.2.3 Search via query input

Actors	Visitor, Curator, Administrator	
Goals	To provide the user an efficient way of finding specific items or content within the system.	
Summary	The user uses a text area to enter their query and searches system entities.	
Related use cases	**Generalization:** ➔ Search for an item **Includes:** ➔ Browse items, exhibits, establishments or facilities	
Steps	*Actor actions*	*System responses*
	1. Included use case: Browse items, exhibits, establishments or facilities 2. Type the search query in the *search bar.*	3. Search designated fields of the entities with the user query. 4. Display search results in the list view.

2.3.2.4 Search via unique item identifier

Actors	Visitor, Curator, Administrator	
Goals	To provide the user an efficient way of finding specific items or content within the system.	
Summary	The user uses the keypad to search for an item.	
Related use cases	**Generalization:** ➔ Search for an item	
Steps	*Actor actions*	*System responses*
	1. Navigate to the identifier input screen. 3. Type item number.	2. Display a keypad. 4. Search unique item numbers for the user input. 5. Display found item.

2.3.2.5 Filter search results

Actors	Visitor, Curator, Administrator	
Goals	To provide the user a way to refine their searches.	
Summary	The user uses predefined filters to narrow down their search.	
Related use cases	**Includes:** ➔ Search for an item	
Steps	*Actor actions*	*System responses*
	1. (Optional) Included use case: Search for an item 2. Select filter. 4. (Optional) Loop step #2	3. Narrow down items with filters.

2.3.2.6 View item details

Actors	Visitor, Curator, Administrator	
Goals	To provide in depth information about an item.	
Steps	*Actor actions*	*System responses*
	1. Click on item.	2. Display item detail page.

2.3.2.7 View media related to an item

Actors	Visitor, Curator, Administrator	
Goals	To enhance the user's experience by providing visual representation and context.	
Related use cases	**Includes:** ➔ View item details	
Steps	*Actor actions*	*System responses*
	1. Included use case: View item details	2. Display item media.
	3. Navigate media using UI controls (swiping etc.) or I/O.	
	4. Click on the image.	5. Open the image in full-screen mode.
	6. Click *back* option.	7. Close full-screen mode

2.3.2.8 Listen to audio related to an item

Actors	Visitor, Curator, Administrator	
Goals	To provide the users with a richer and more engaging multimedia experience, catering to different preferences and enhancing the overall usability and enjoyment of the application.	
Related use cases	**Includes:** ➔ View item details	
Steps	*Actor actions*	*System responses*
	1. Included use case: View item details 2. Click the *play audio* button.	3. Display *play audio* button. 1. Display audio control buttons such as play, pause, seek, adjust volume etc. 2. Play the audio file designated for the item.
Post conditions	The system utilizes either the device's built-in speakers or any preferred external listening device to play the audio.	

2.3.2.9 Create an item

Actors	Curator, Administrator	
Steps	*Actor actions*	*System responses*
	1. Click the *create item* option. 3. Fill in the fields in the form. 4. Click the *save* option.	2. Display item creation form. 5. Validate the entered information and check for any missing or erroneous data. 6. Save the item in the application's storage system. 7. Associate any uploaded media files with the item, if applicable. 8. Confirm the successful creation of the item and provide visual feedback or notification to the user.
Post conditions	The item is saved in the system and can be viewed by the users with the appropriate authority.	

2.3.2.10 Add item details

Actors	Curator, Administrator	
Steps	*Actor actions*	*System responses*
	1. Select an item from the application's interface or search results.	
		2. Display the details page or form for the selected item.
	3. Click the *edit* option.	
		4. Display the item edit form.
	5. Update information.	
	6. Assign tags of categories.	
		7. Validate the entered information.
		8. Save the updated item details in the application's database or storage system.
		9. Confirm the successful update of the item details and provide visual feedback or notification to the user.
Post conditions	➔ The item details are saved in the system and can be viewed by the users with the appropriate authority.	

2.3.2.11 Add item details in another language

Actors	Curator, Administrator	
Steps	*Actor actions*	*System responses*
	1. Select an item. 4. Select the target language for the item details. 5. Enter the item details into the target language. 6. Submit the translated item details in the target language.	2. Display the details page or form for the selected item. 3. Display a language selector or dropdown menu for the user to choose the target language. 7. Validate the entered information. 8. Save the translated item details in the application's database or storage system, associating them with the selected target language.
Post conditions	➔ The item details are saved in the system and can be viewed by the users with the appropriate authority.	

2.3.2.12 Create an exhibit

Actors	Curator, Administrator	
Steps	*Actor actions*	*System responses*
	1. Select the create exhibit option.	
	3. Fill in the required information for the exhibit, such as name, description, duration, and any other relevant attributes. 4. (Optional) Upload images, videos, or other multimedia content related to the exhibit. 5. Select items the exhibit will contain. 6. Submit the exhibit creation request.	2. Display exhibit creation form.
		7. Validate the entered information and check for any missing or erroneous data. 8. Save the item in the application's storage system. 9. Associate any uploaded media files with the item, if applicable. 10. Confirm the successful creation of the item and provide visual feedback or notification to the user.
Post conditions	➔ The exhibit is saved in the system and can be viewed by the users with the appropriate authority. ➔ The exhibit is associated with the specified items.	

2.3.2.13 Create a category

Actors	Administrator	
Goals	To allow users to add categories for item content.	
Summary	The user creates a new category in their tenant's scope.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the category creation screen. 2. Click the *create category* option. 4. Fill in the form. 5. Click the *save* option.	3. Display a form with input areas for a new category. 6. Validate the input and check for potential conflicts. 7. Create a new category with the information the user has provided. 8. Create a relation with the newly created category and the user's tenant. 9. Confirm the successful creation of the category and provide feedback to the user.
Post conditions	➔ The new category is successfully created and added to the system. ➔ The user can now assign items to the newly created category or further configure its attributes.	

2.3.2.14 Create a level

Actors	Administrator	
Goals	To allow users to add levels for item content.	
Summary	The user creates a new level in their tenant's scope.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the level creation screen. 2. Click the *create level* option. 4. Fill in the form. 5. Click the *save* option.	3. Display a form with input areas for a new level. 6. Validate the input and check for potential conflicts. 7. Create a new level with the information the user has provided. 8. Create a relation with the newly created level and the user's tenant. 9. Confirm the successful creation of the level and provide feedback to the user.
Post conditions	➔ The new level is successfully created and added to the system. ➔ The user can now assign items to the newly created level or further configure its attributes.	

2.3.2.15 Edit a category

Actors	Administrator	
Goals	To allow users to edit categories.	
Summary	The user edits a category.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the categories screen. 2. Locate the category to be edited. 3. Click the *edit category* option. 5. Change the details in the form. 6. Click the *save* option.	4. Display a form with input fields pre-filled with the properties of the selected category. 7. Validate the input and check for potential conflicts. 8. Save the category with the information the user has provided. 9. Confirm the successful update of the category and provide feedback to the user.
Post conditions	➔ The category is successfully updated. ➔ The changes made to the category will be reflected where it is used in the system.	

2.3.2.16 Edit a level

Actors	Administrator	
Goals	To allow users to edit levels.	
Summary	The user edits a level.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the levels screen. 2. Locate the level to be edited. 3. Click the *edit level* option. 5. Change the details in the form. 6. Click the *save* option.	4. Display a form with input fields pre-filled with the properties of the selected level. 7. Validate the input and check for potential conflicts. 8. Save the level with the information the user has provided. 9. Confirm the successful update of the level and provide feedback to the user.
Post conditions	➔ The level is successfully updated. ➔ The changes made to the level will be reflected where it is used in the system.	

2.3.2.17 Delete a category

Actors	Administrator	
Goals	To allow users to delete categories.	
Summary	The user deletes a category.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the categories screen. 2. Locate the category to be deleted. 3. Click the *delete category* option. 5. Confirm the action.	4. Check if the conditions are met for the category to be deleted. 5. Prompt to confirm the user action. 6. Remove category information from the application storage. 7. Confirm the successful removal of the category and provide visual feedback or notification to the user.
Post conditions	➔ The category is successfully removed from the application storage. ➔ The changes made to the category will be reflected where it is used in the system.	

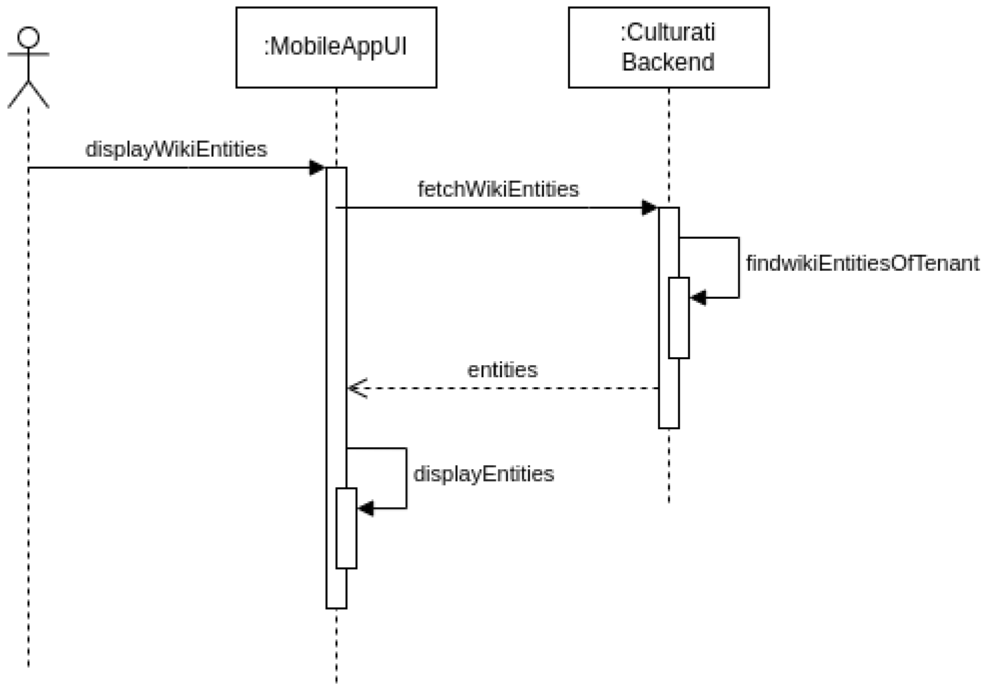
2.3.2.18 Delete a level

Actors	Administrator	
Goals	To allow users to delete levels.	
Summary	The user deletes a level.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the levels screen. 2. Locate the level to be deleted. 3. Click the *delete level* option. 6. Confirm the action.	4. Check if the conditions are met for the level to be deleted. 5. Prompt to confirm the user action. 7. Remove the level information from the application storage. 8. Confirm the successful removal of the level and provide visual feedback or notification to the user.
Post conditions	➔ The level is successfully removed from the application storage. ➔ The changes made to the level will be reflected where it is used in the system.	

2.3.3. Sequences

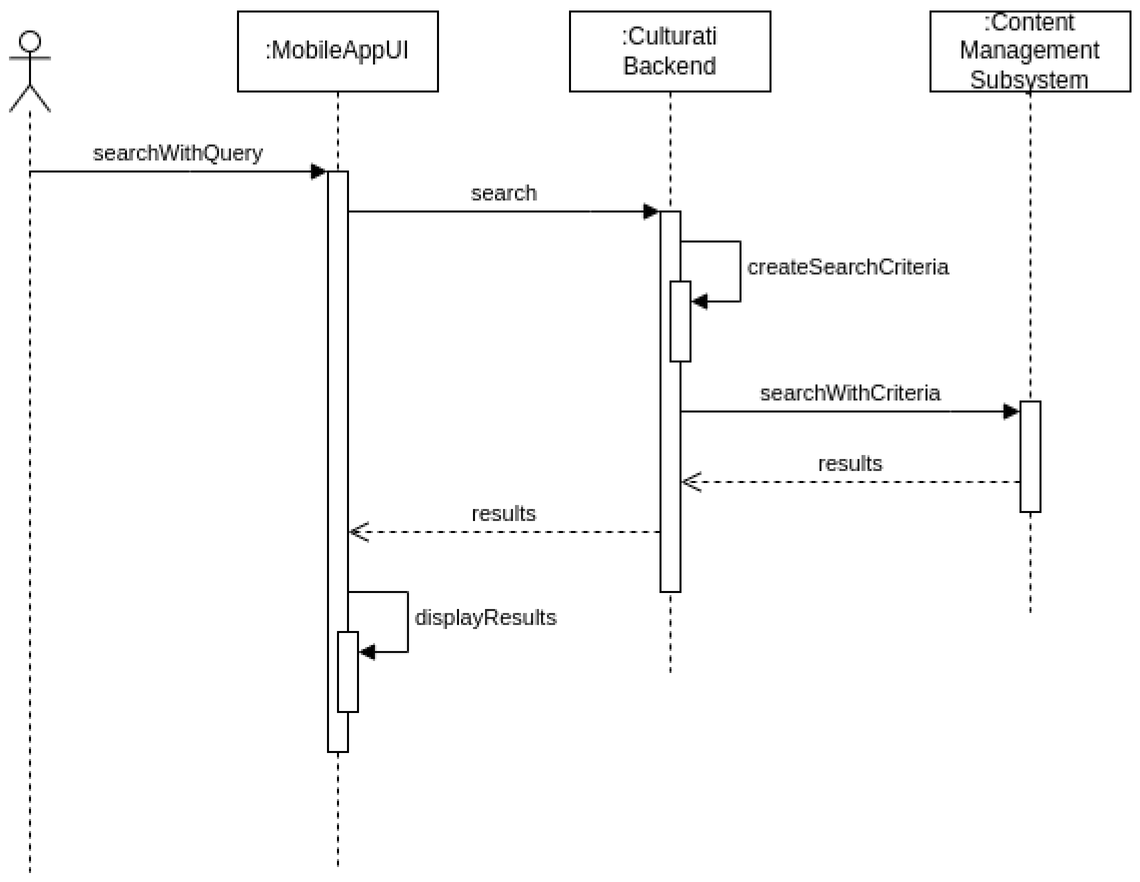
2.3.3.1 Browse items, exhibits, establishments or facilities

Role: Visitor, Curator, Administrator



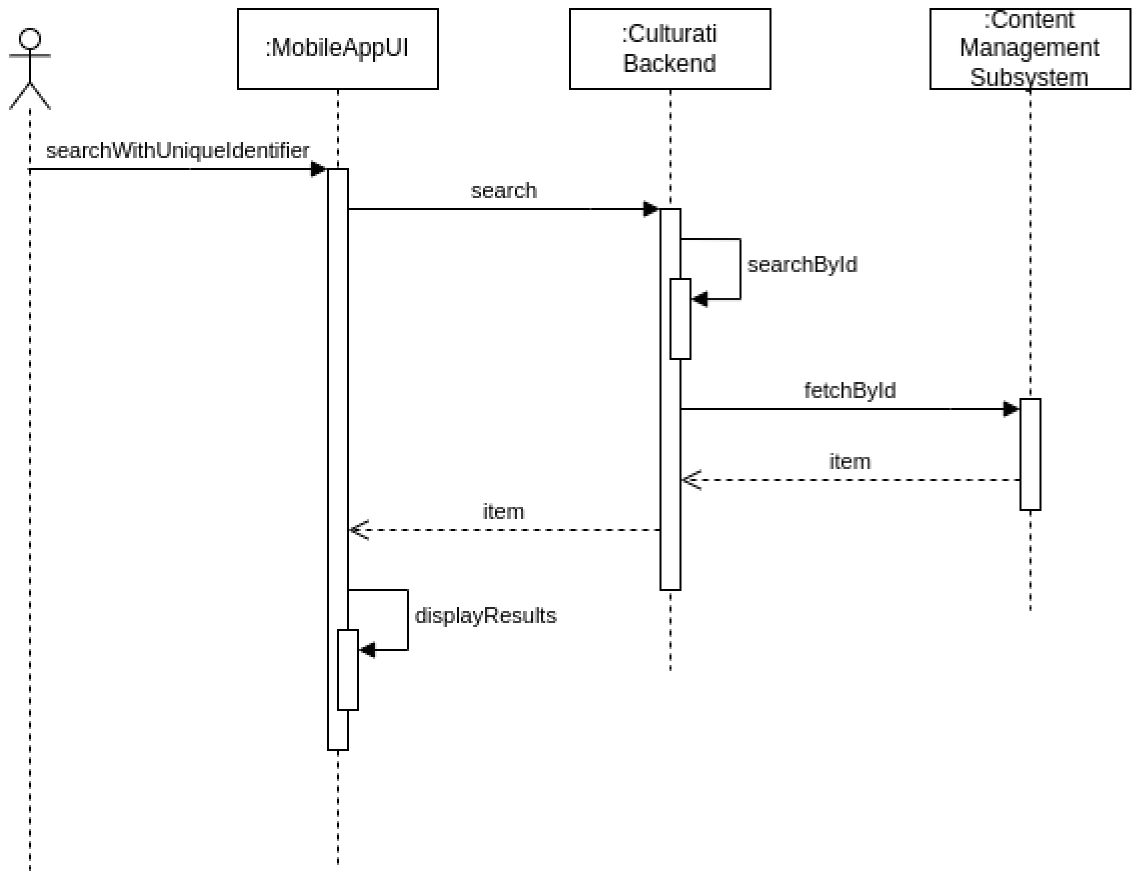
2.3.3.2 Search for an item via query

Role: Visitor, Curator, Administrator



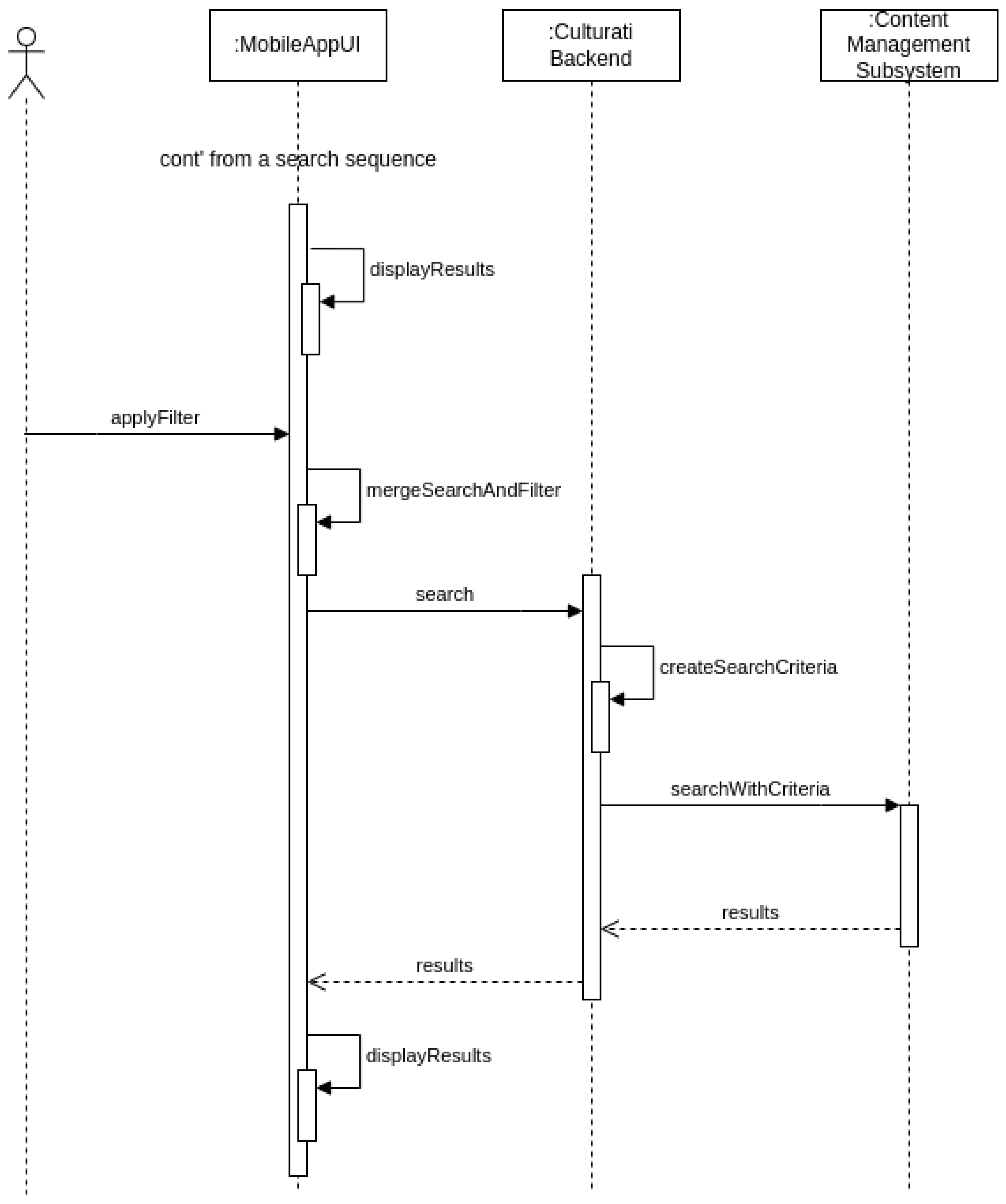
2.3.3.3 Search for an item via unique item identifier

Role: Visitor, Curator, Administrator



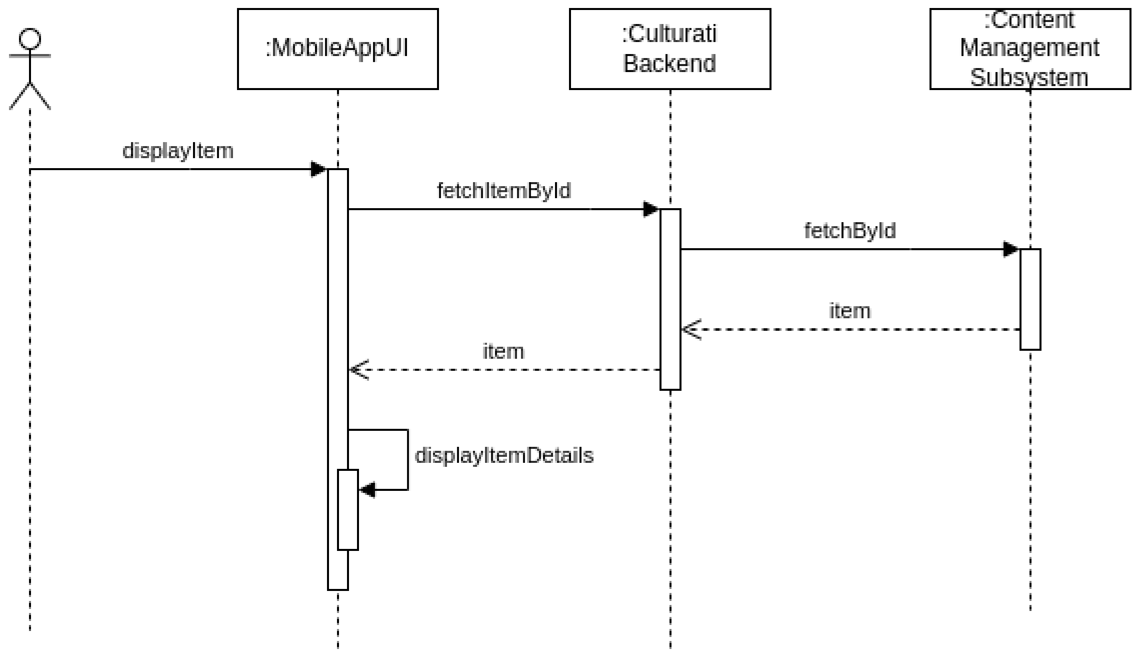
2.3.3.4 Filter search results

Role: Visitor, Curator, Administrator



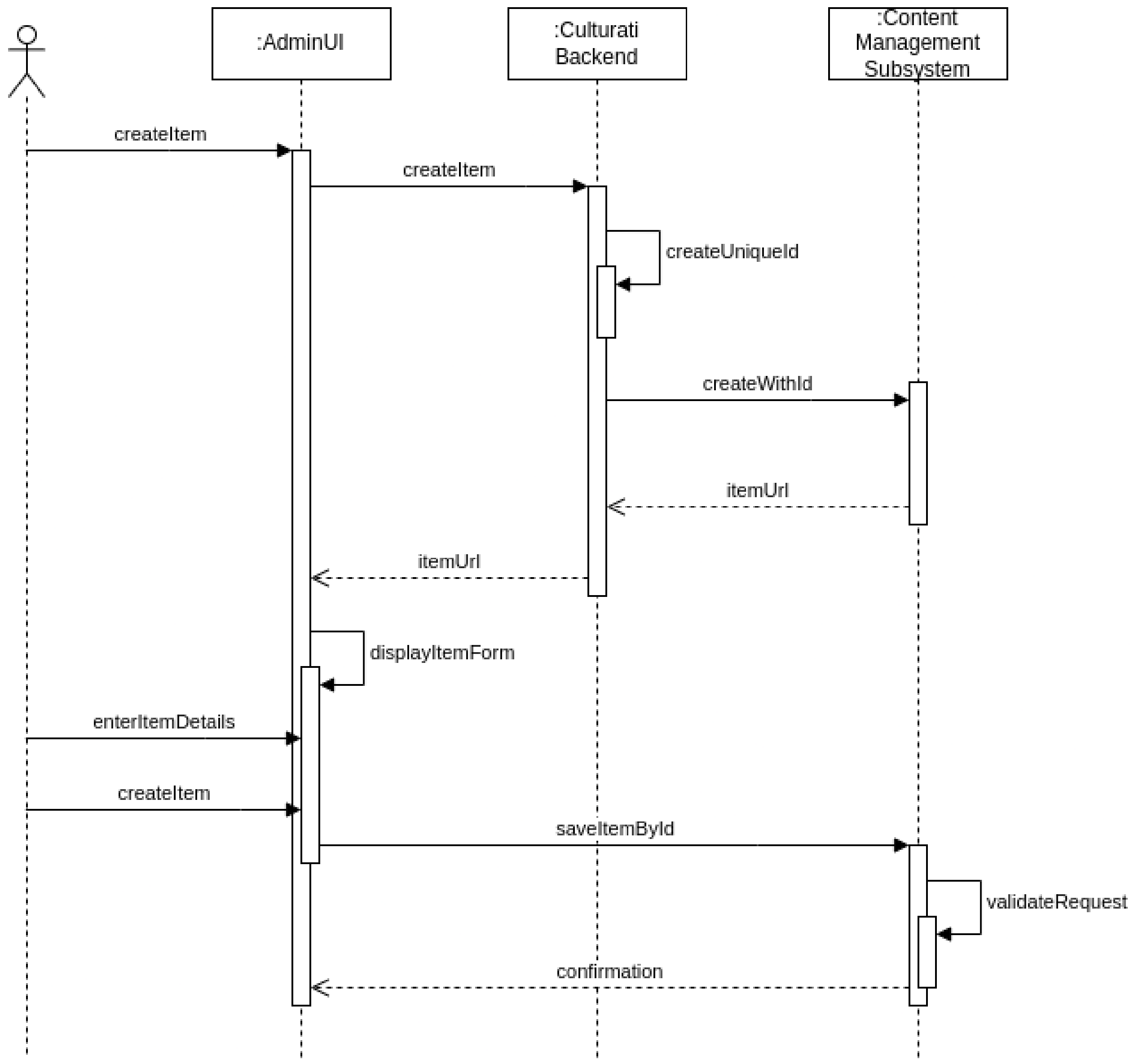
2.3.3.5 View item details

Role: Visitor, Curator, Administrator



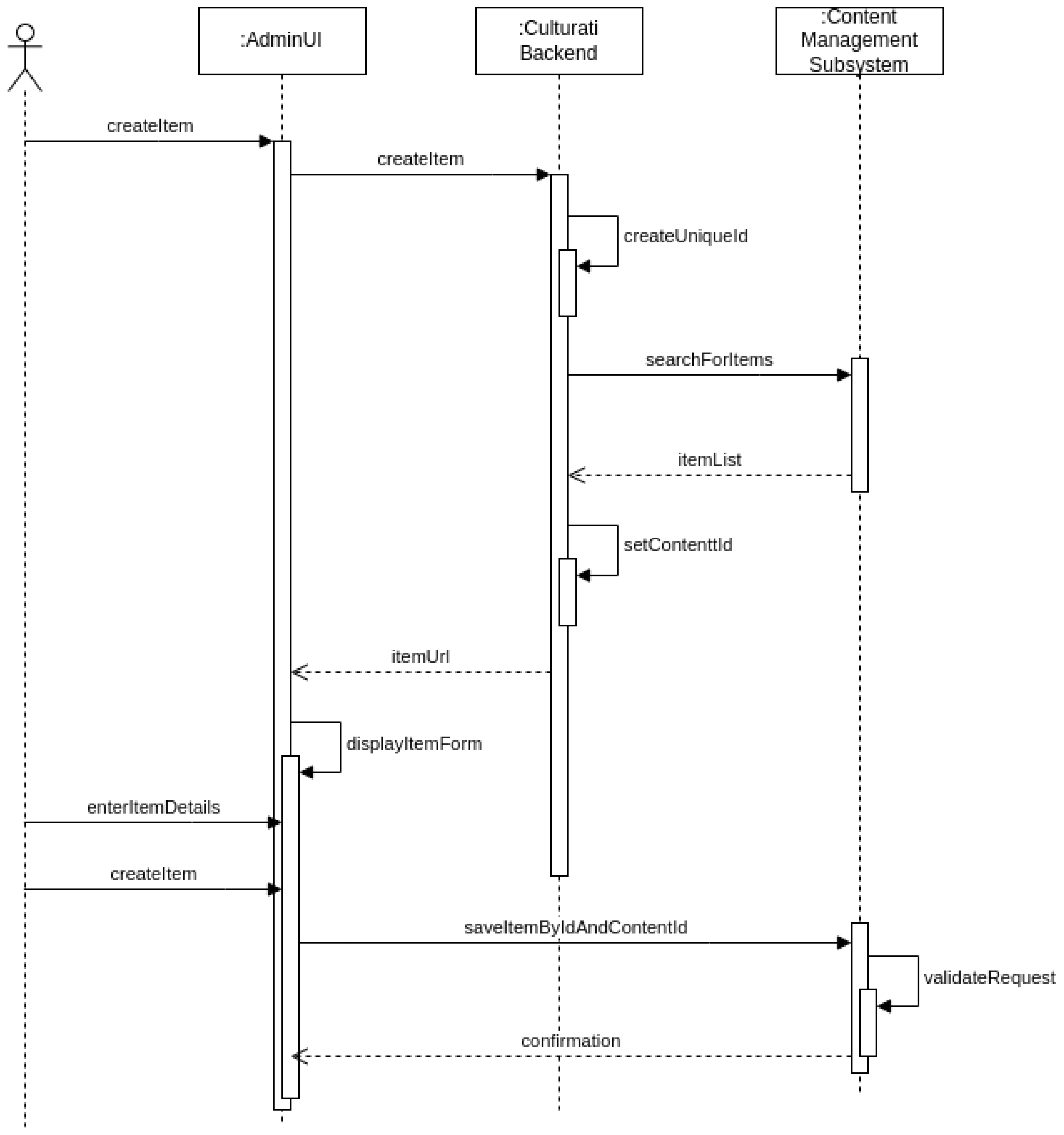
2.3.3.6 Create an item

Role: Admin, Curator



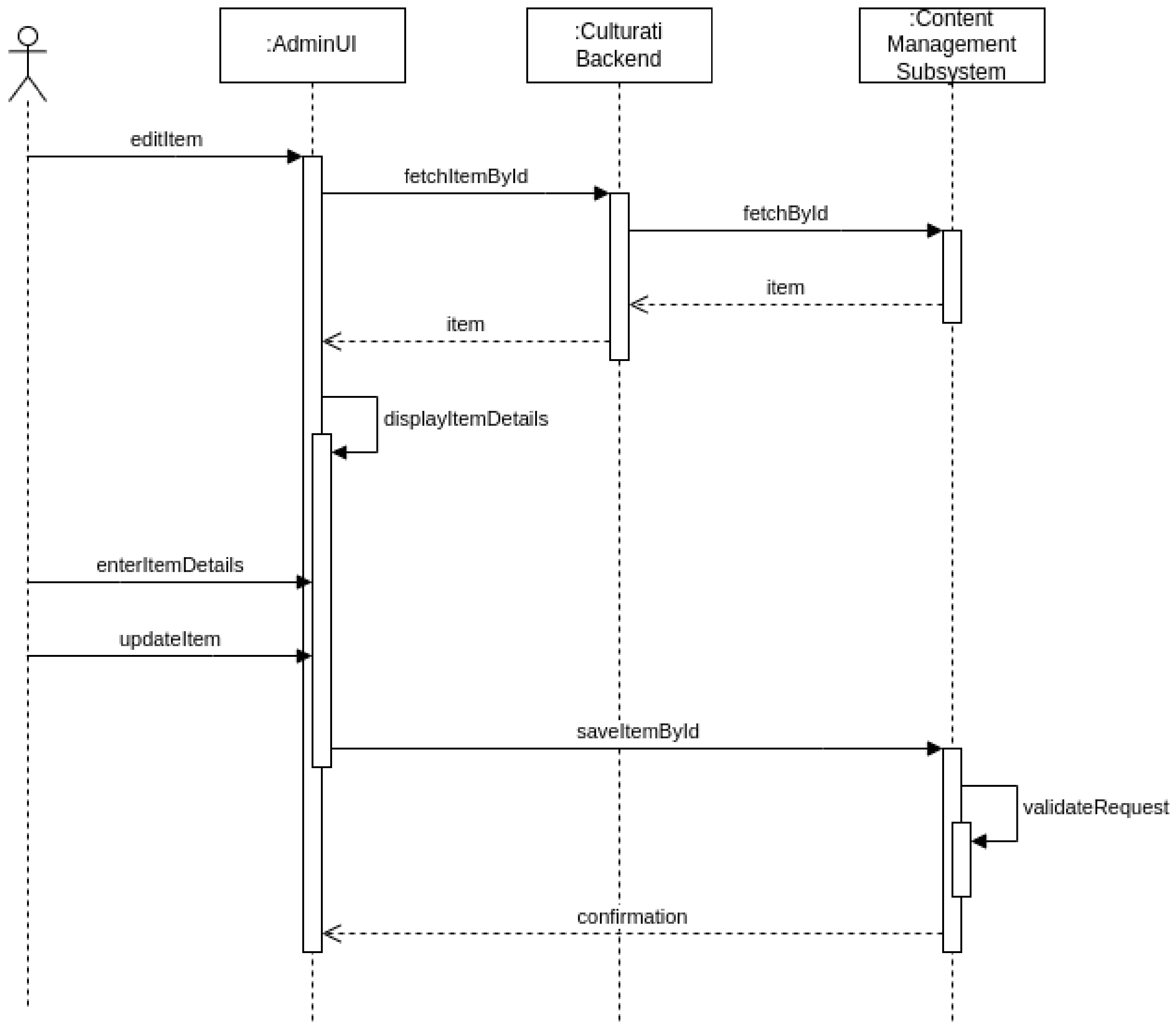
2.3.3.7 Create an item with existing content

Role: Admin, Curator



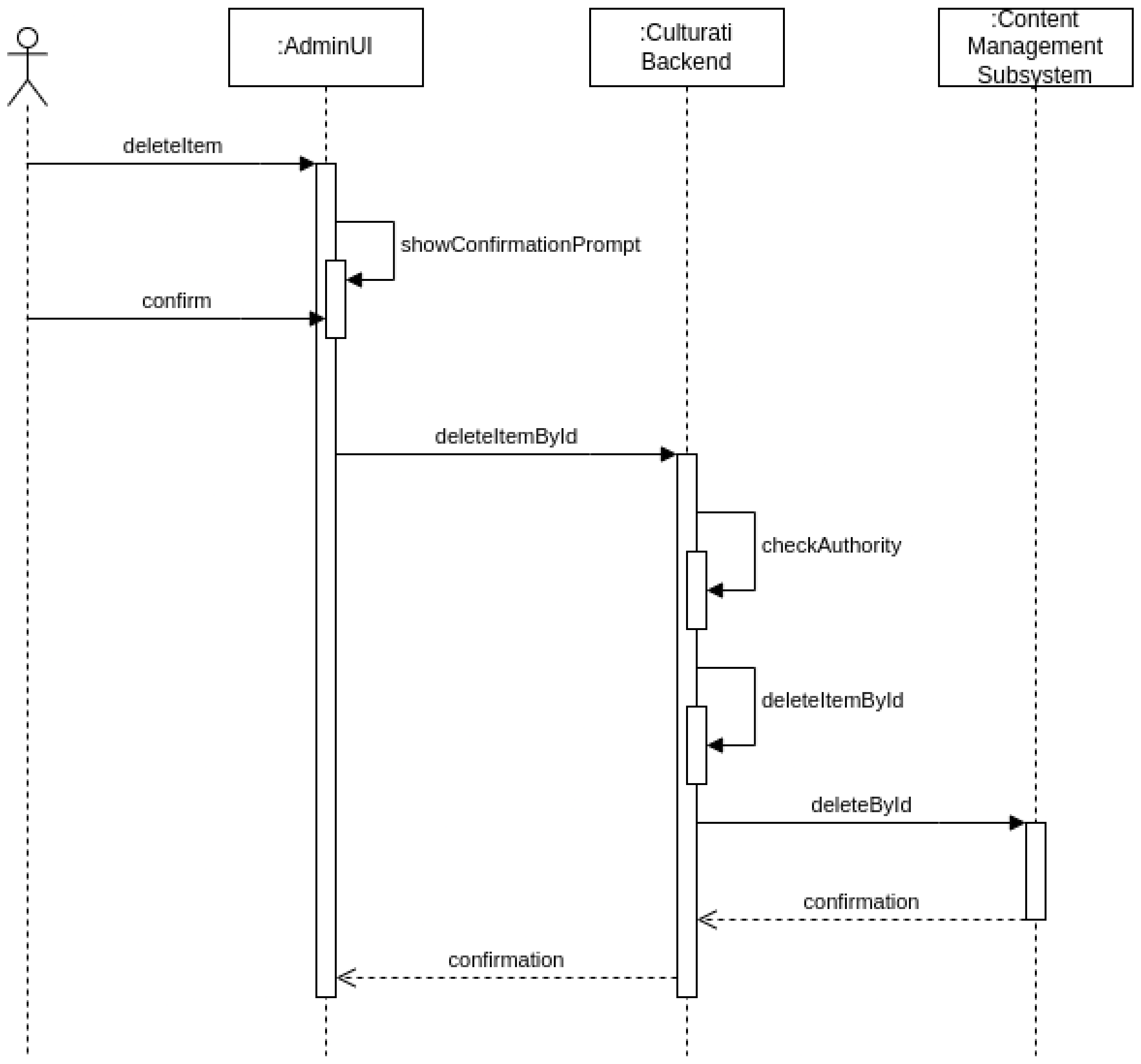
2.3.38 Edit an item

Role: Admin, Curator

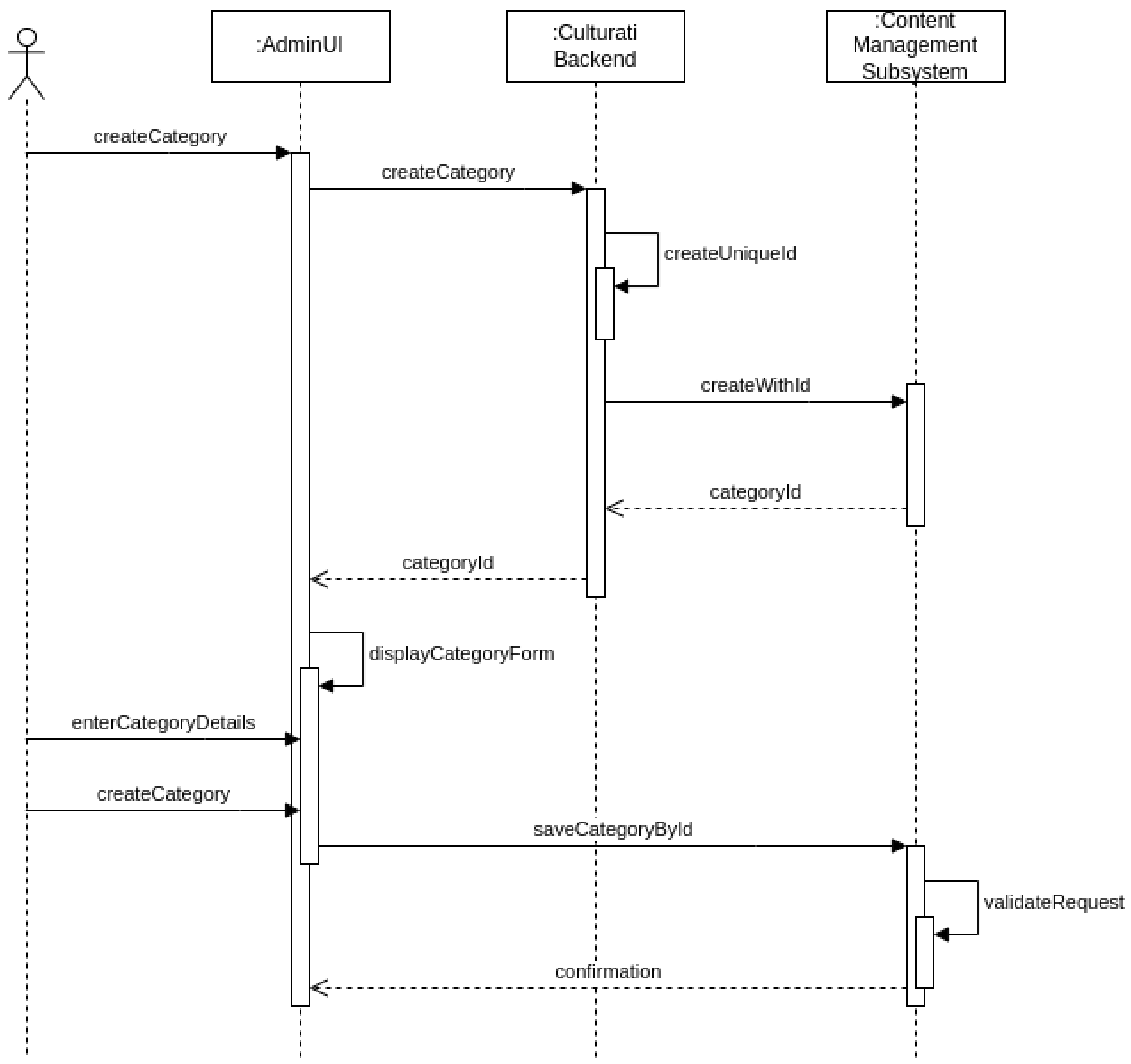


2.3.3.9 Delete an item

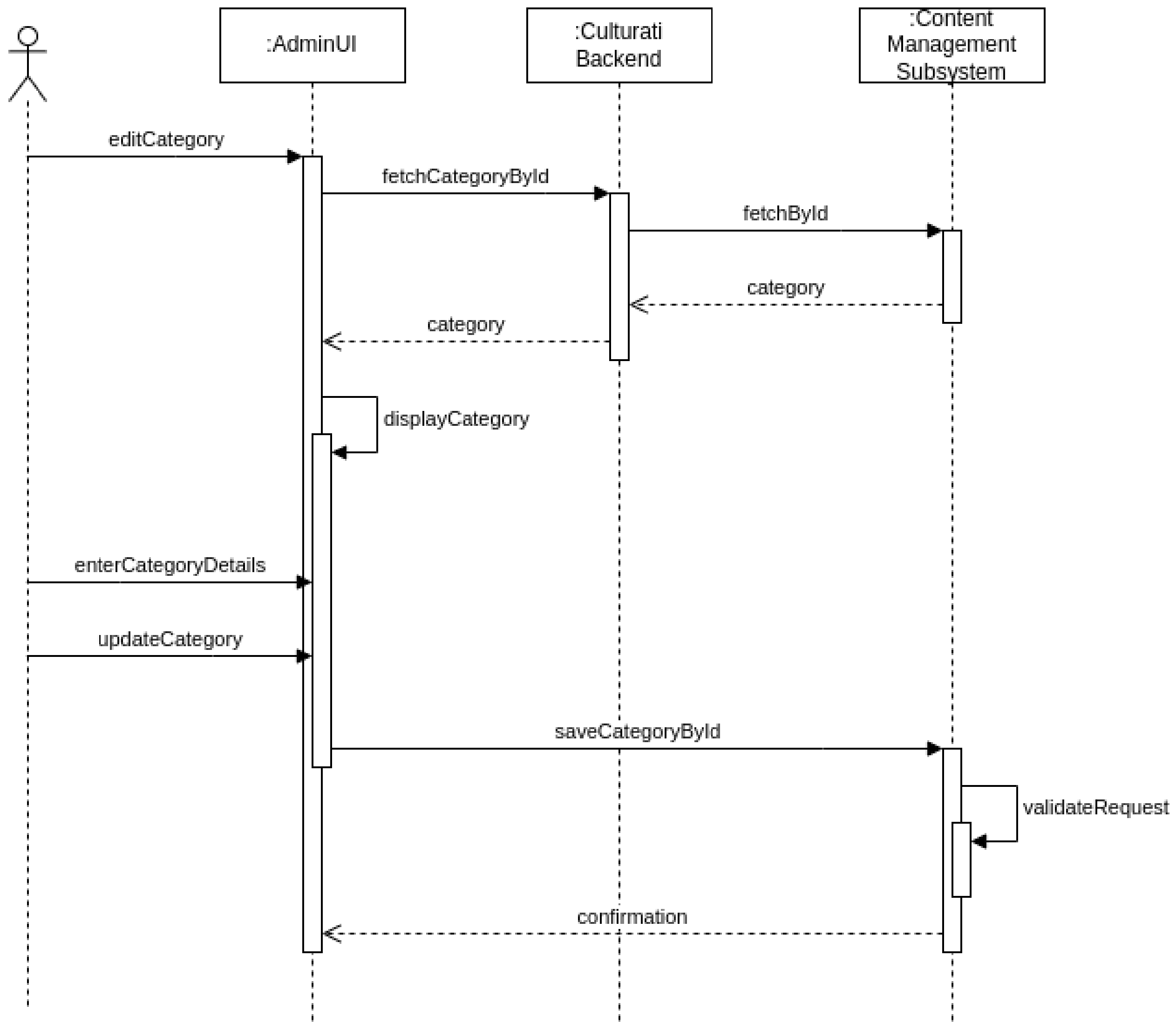
Role: Admin, Curator



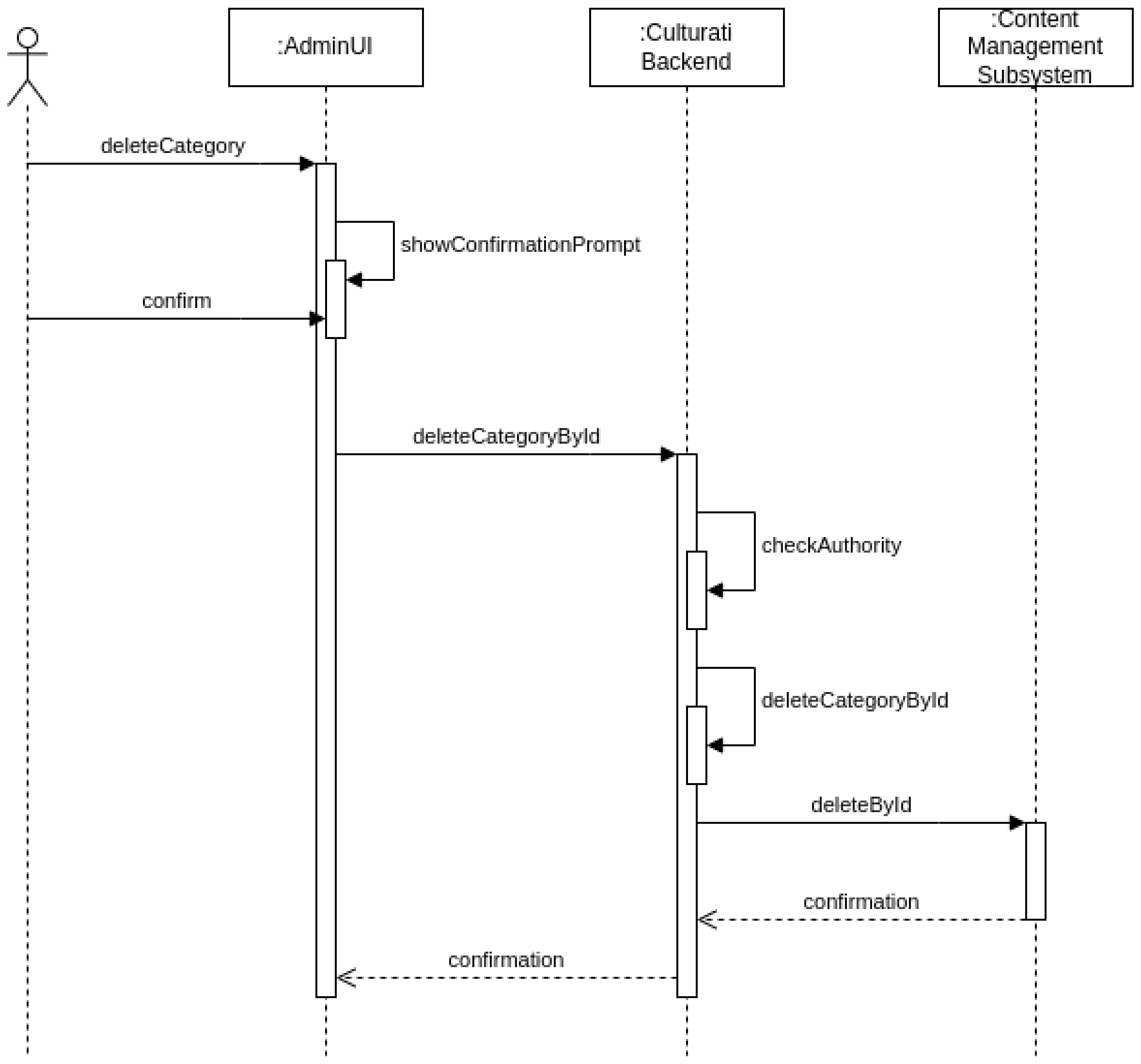
2.3.3.10 Create a category



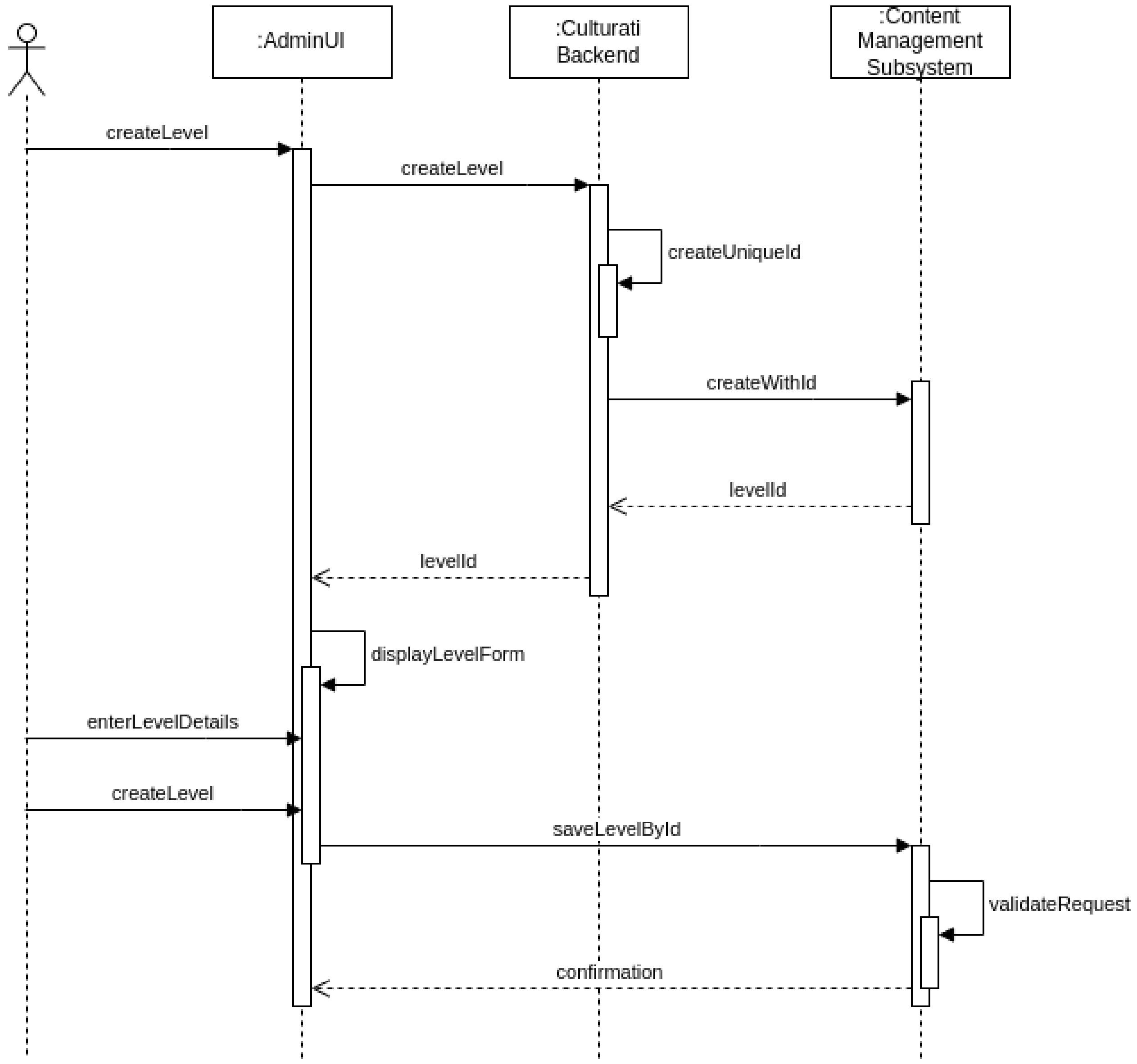
2.3.3.11 Edit a category



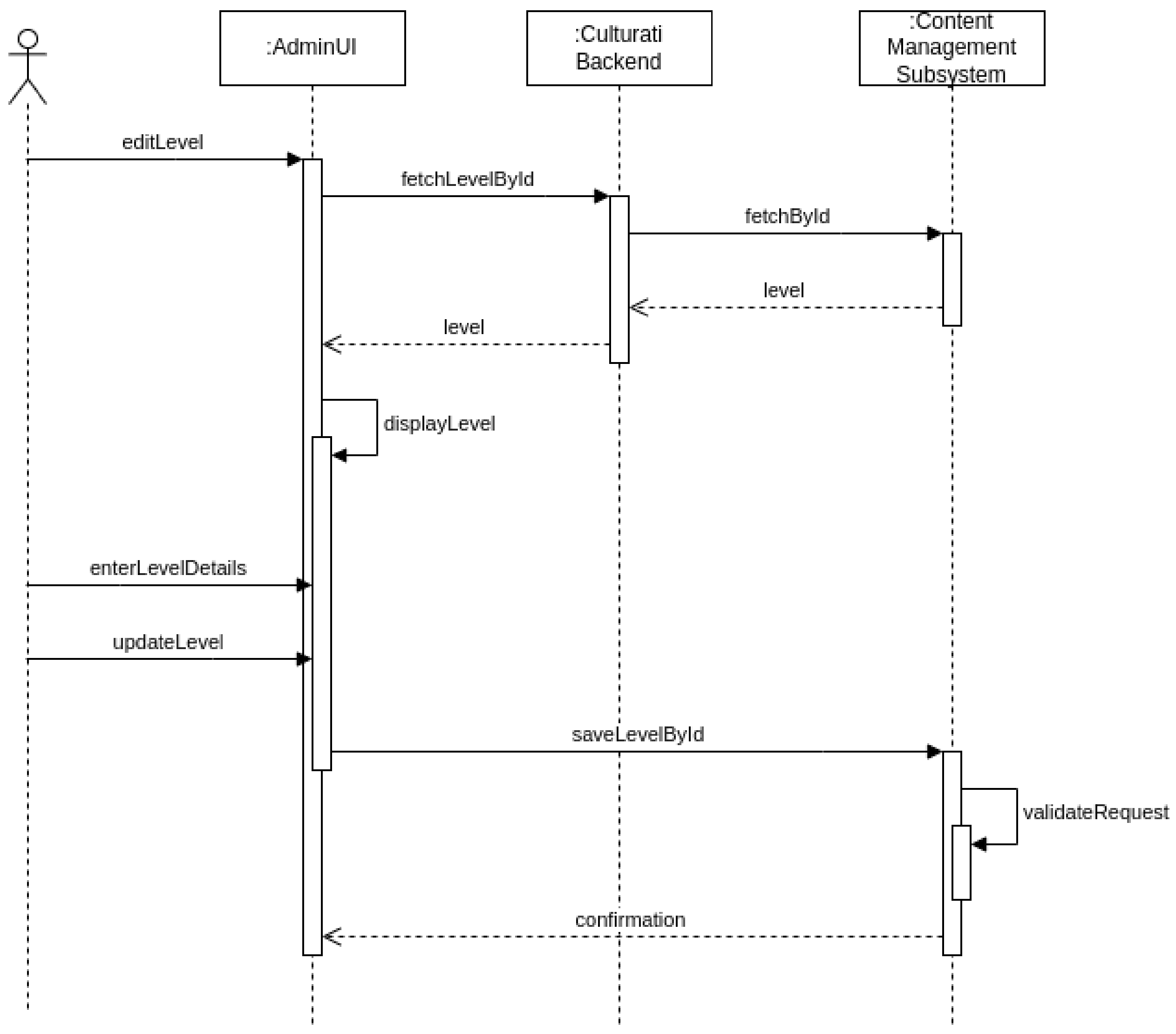
2.3.3.12 Delete a category



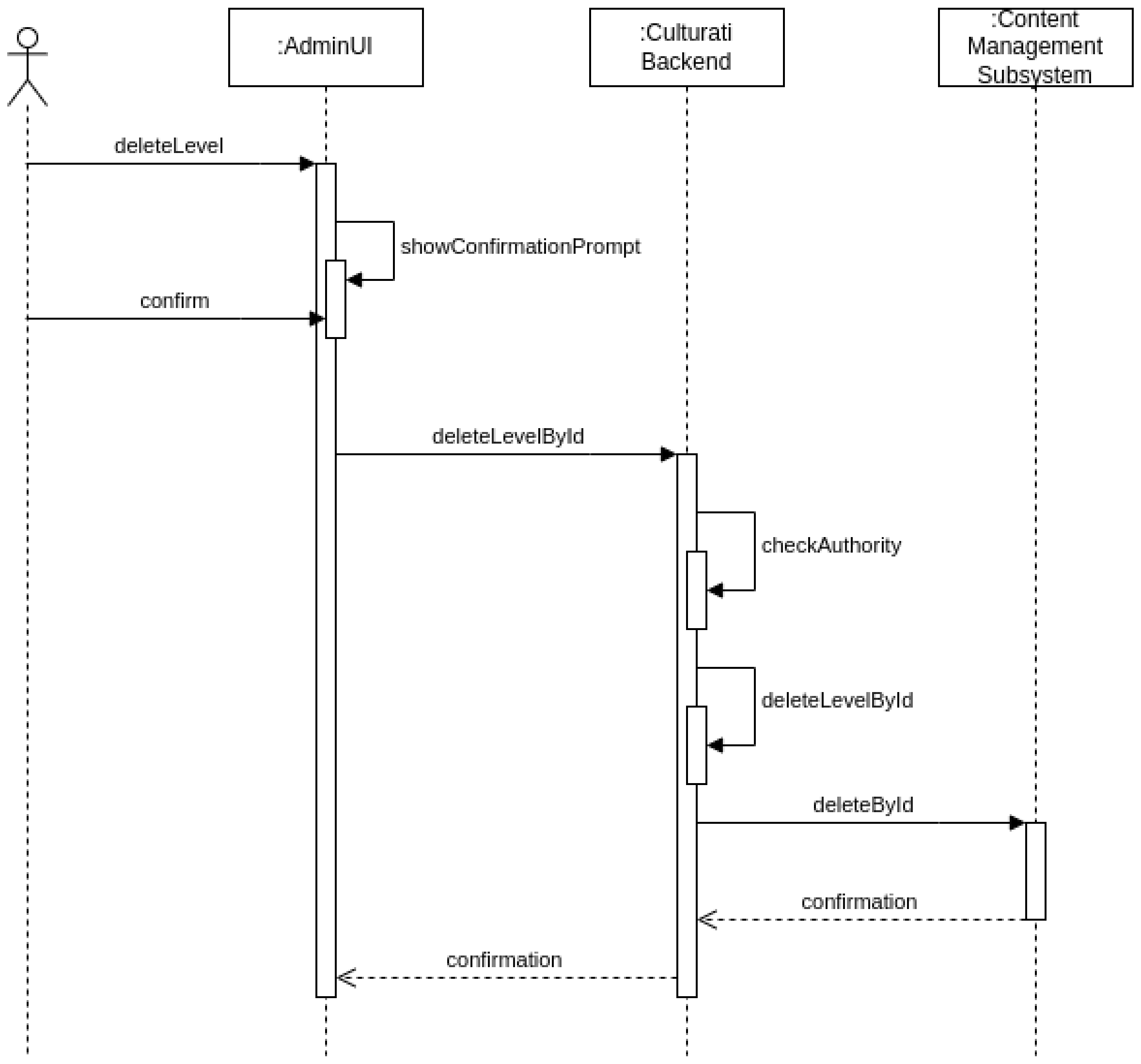
2.3.3.13 Create a level



2.3.3.14 Edit a level



2.3.3.15 Delete a level



2.4 User Management Subsystem

2.4.1 General Information

The user management subsystem is responsible for managing user-related functionalities and activities. It handles the management and administration of user accounts, authentication, authorization, and user profile information. The primary purpose of the user management subsystem is to facilitate secure and efficient management of user identities, access privileges, and personalized user experiences within the application.

2.4.2 Use Cases

2.4.2.1 Sign in

Actors	Visitor, Curator, Administrator, Super Administrator	
Goals	To authenticate the user.	
Preconditions	The user is not signed in.	
Steps	*Actor actions*	*System responses*
	1. Click on sign in. 3. Fill the form. 4. Click *Confirm.*	2. Show sign in form. 5. Authenticate the user. 6. Show success message.
Post conditions	The system identifies the user associated with the ongoing session.	

2.4.2.2 Register

This use case is optional for users. In the event that users choose not to register, a random ID and username are automatically assigned to them. The system then identifies them based on their random ID, which persists as long as their device or browser retains it.

Actors	Visitor	
Goals	To create an account for the user.	
Preconditions	The user is not signed in.	
Steps	*Actor actions*	*System responses*
	1. Click on register.	
		2. Show registration form that consists of username, email and password fields. 3. Fill the username with a randomly generated username.
	4. Fill in the form. 5. Change the randomly generated username. 6. Click confirm.	
		7. Save the user with a randomly generated id. 8. Sign the user in. 9. Redirect to the main page.
Post conditions	A new user is created in the system and authorized.	

2.4.2.3 Add a user to own tenant

Actors	Administrator	
Goals	To give access to other users of the admin's tenant.	
Steps	*Actor actions*	*System responses*
	1. Click *add user* option. 3. Fill in the form. 4. Click submit.	2. Display the user creation form. 5. Validate the entered information and check for any missing or erroneous data. 6. Save the user in the storage system. 7. Associate the user with the current tenant. 8. Confirm the successful creation of the item and provide visual feedback or notification to the user.

2.4.2.4 Delete a user of own tenant

Actors	Administrator	
Goals	To remove a user from the system and revoke their access to system resources.	
Steps	*Actor actions*	*System responses*
	1. Click *remove user* option. 3. Confirm action.	2. Display a prompt to confirm user action. 4. Remove user information from the application storage. 5. Change references to the user in the storage with a placeholder. 6. Confirm the successful removal of the user and provide visual feedback or notification to the user.

2.4.2.5 Change the password of a user in own tenant

Actors	Administrator	
Goals	To reset the password of a user.	
Steps	*Actor actions*	*System responses*
	1. Select a user. 3. Click the *reset password* option. 5. Enter the new password. 6. Click submit.	2. Display user actions. 4. Display a text field for the new password. 7. Update the user information in the application storage. 8. Confirm the successful change of the user password and provide visual feedback or notification to the user.
Post conditions		

2.4.2.6 Add a user to a tenant

Actors	Super Administrator	
Goals	To give access to a user of a tenant.	
Steps	*Actor actions*	*System responses*
	1. Click *add user* option. 3. Fill in the form. 4. Click submit.	2. Display the user creation form. 5. Validate the entered information and check for any missing or erroneous data. 6. Save the user in the storage system. 7. Associate the user with the current tenant. 8. Confirm the successful creation of the item and provide visual feedback or a notification to the user.
Post conditions	➔ A new user is successfully added to the tenant. ➔ The system sends an invitation or activation email to the new user. ➔ The new user can access the system using their provided credentials and the assigned role.	

2.4.2.7 Edit a user

Actors	Super Administrator, Administrator	
Goals	To change the information related to a user.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the user. 2. Click the *edit user* option. 4. Change the fields in the form. 5. Click submit.	3. Display the user edit form. 6. Validate the entered information and check for any missing or erroneous data. 7. Save the user in the storage system. 8. Associate the user with the current tenant. 9. Confirm the successful creation of the item and provide visual feedback or notification to the user.
Post conditions	➔ A new user is successfully added to the tenant. ➔ The system sends an invitation or activation email to the new user. ➔ The new user can access the system using their provided credentials and the assigned role.	

2.4.2.8 Delete a user

Actors	Super Administrator, Administrator	
Goals	To remove a user from the system and revoke their access to system resources.	
Steps	*Actor actions*	*System responses*
	1. Click *remove user* option. 3. Confirm action.	2. Display a prompt to confirm user action. 4. Remove user information from the application storage. 5. Change references to the user in the storage with a placeholder. 6. Confirm the successful removal of the user and provide visual feedback or notification to the user.
Post conditions	➔ The user's access and privileges within the tenant management system are revoked. ➔ Any related data or dependencies affected by the user's removal are appropriately updated.	

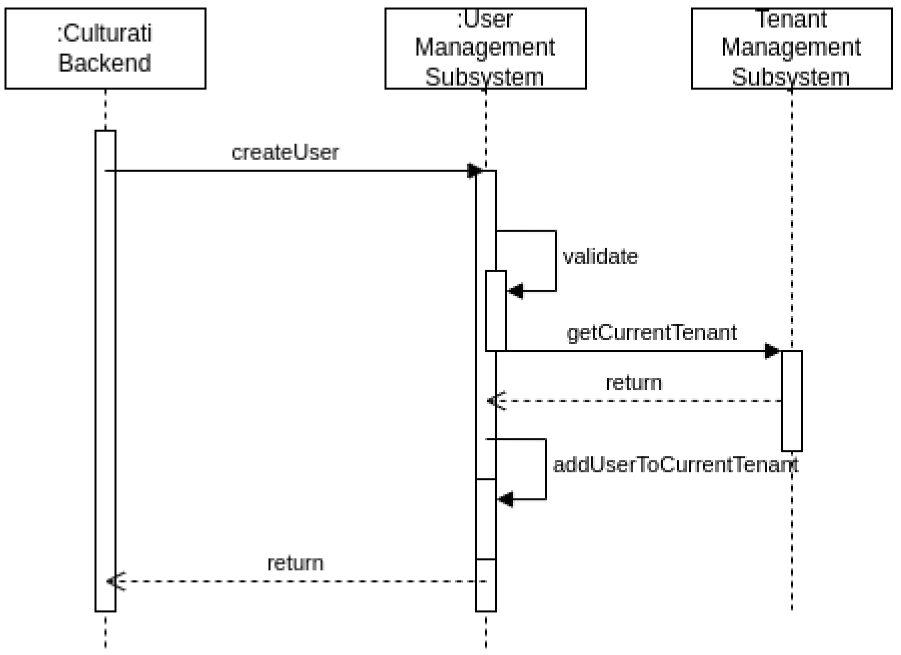
2.4.3 Design Decisions

No demographic data will be collected about the users using the mobile application. Users will be assigned a randomized username and a randomly generated ID. To access the administration application, users will need to log in using their email address and a password. The super-administrator alone will have the privilege to create admin users for tenants, while admin users will have the authority to create users specifically for their respective tenants.

2.4.4 Sequences

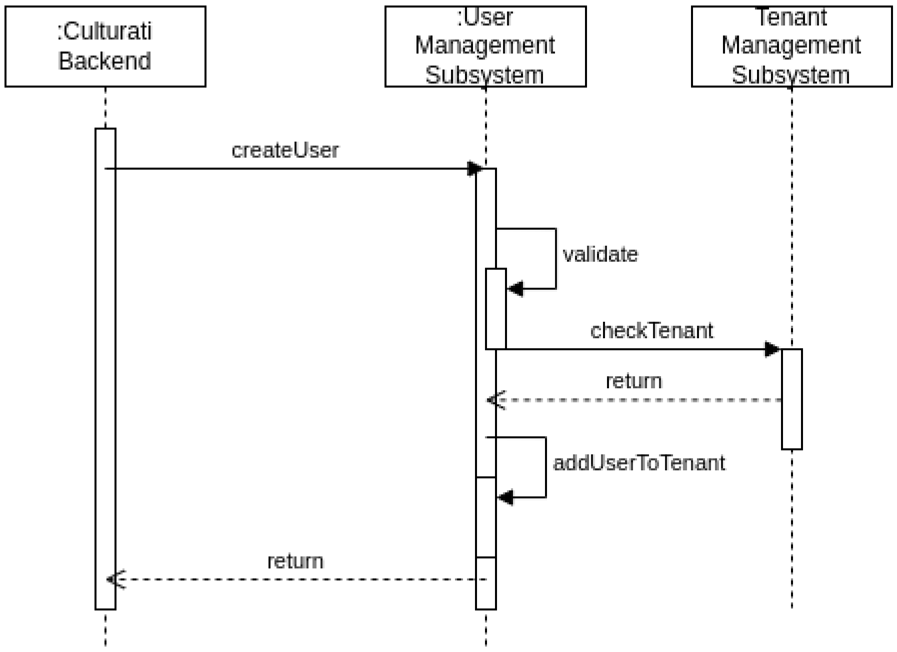
2.4.4.1 Create user

Role : Admin



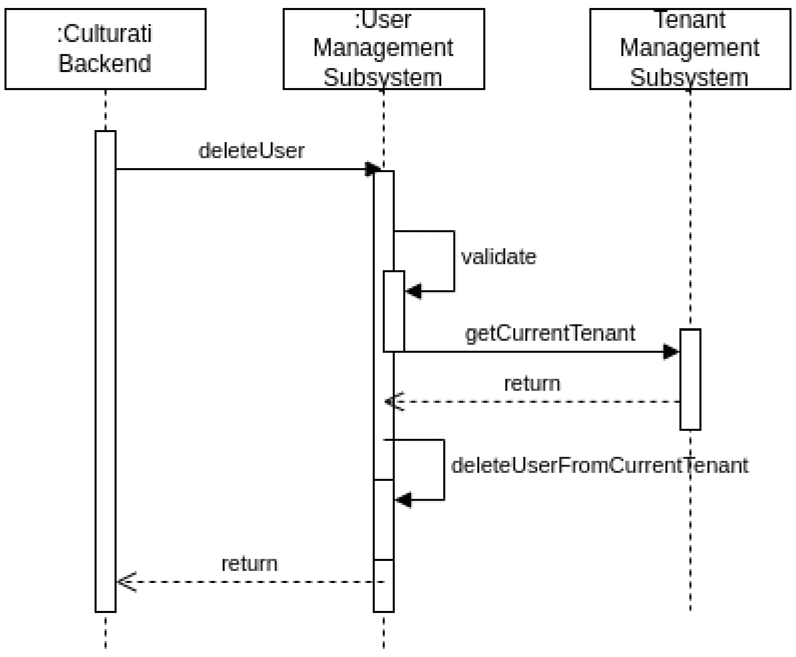
2.4.4.2 Create user on a tenant

Role: super admin



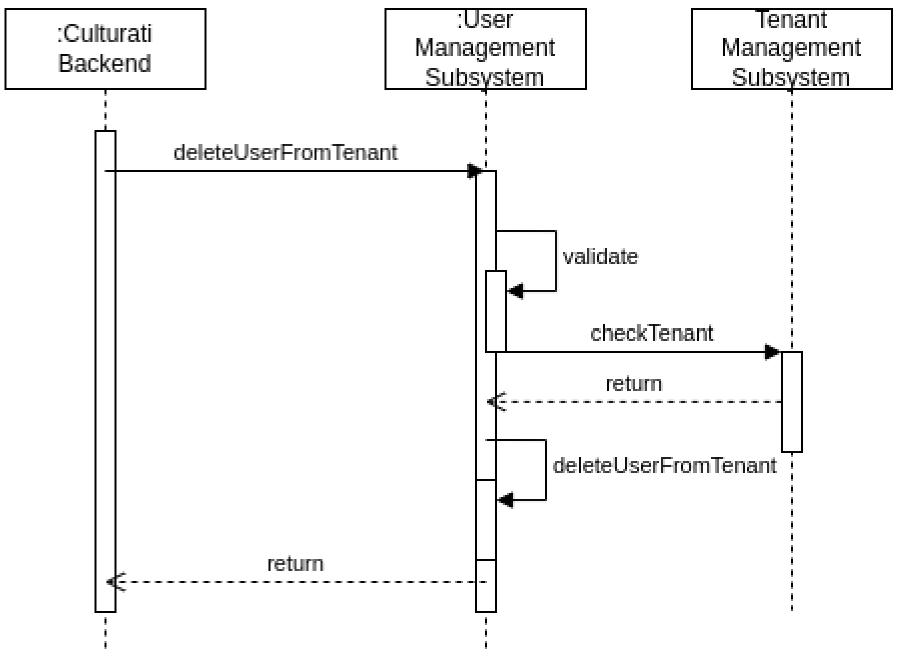
2.4.4.3 Deactivate user

Role: Admin


:Culturati Backend
:User Management Subsystem
Tenant Management Subsystem
deleteUser
validate
getCurrentTenant
return
deleteUserFromCurrentTenant
return

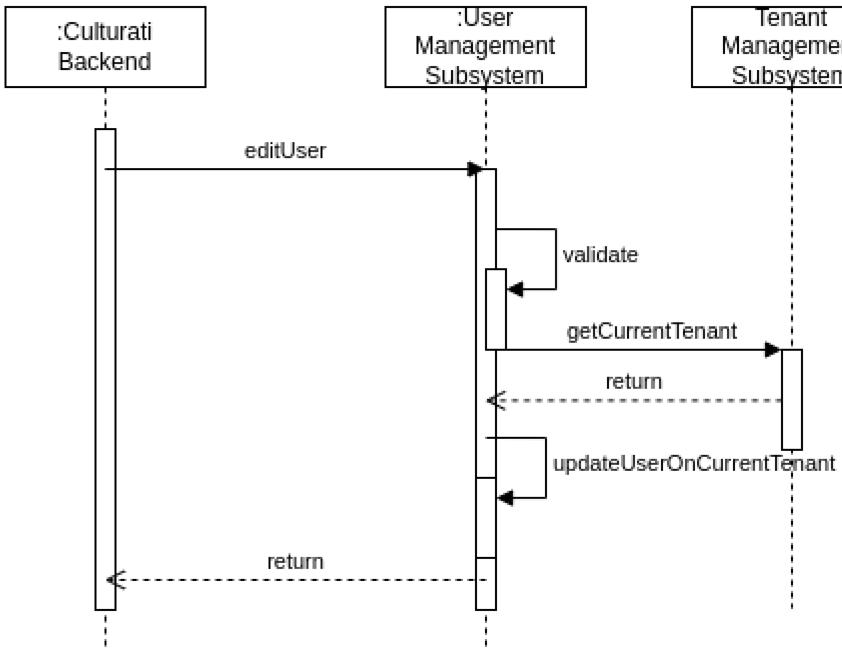
2.4.4.4 Delete user from a tenant

Role: Super Admin



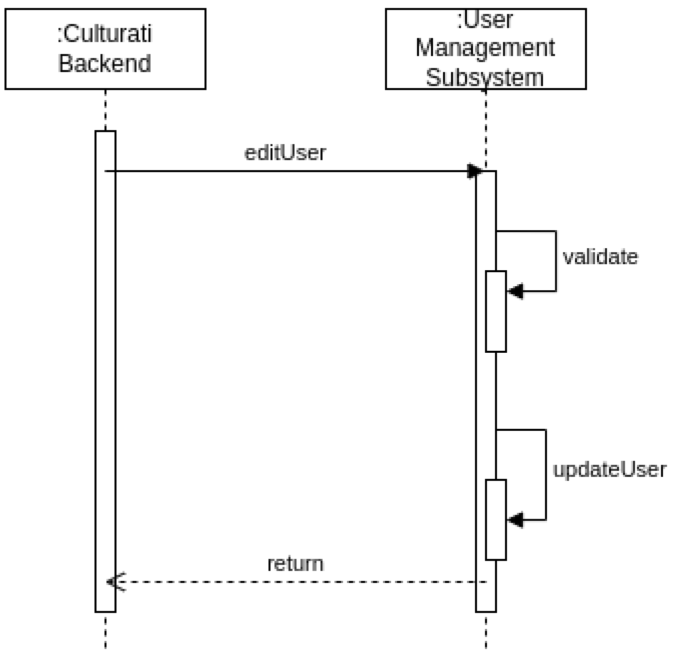
2.4.4.5 Edit user

Role: Admin


:Culturati Backend
:User Management Subsystem
Tenant Management Subsystem
editUser
validate
getCurrentTenant
return
updateUserOnCurrentTenant
return

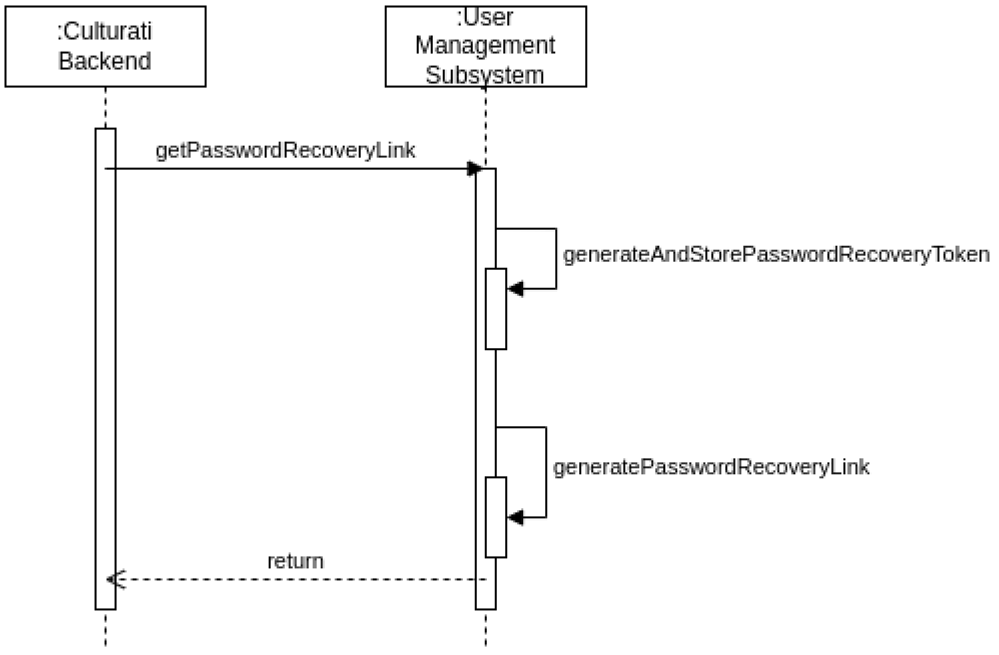
2.4.4.6 Edit user

Role: Super Admin



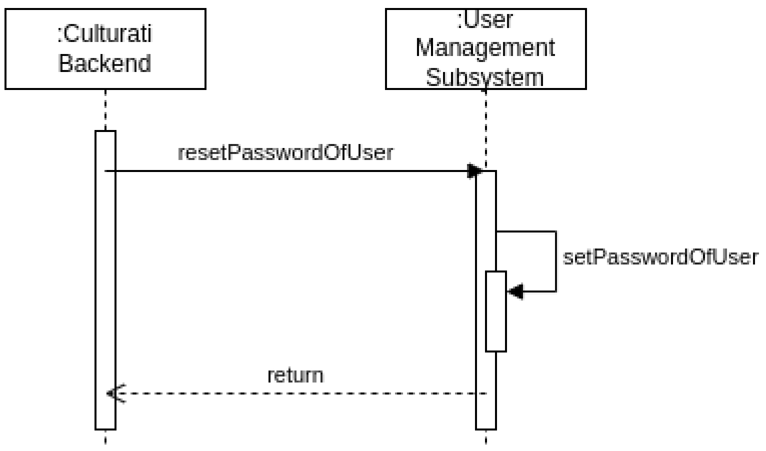
2.4.4.7 Get Password Recovery Link

Role: User


:Culturati Backend
:User Management Subsystem
getPasswordRecoveryLink
generateAndStorePasswordRecoveryToken
generatePasswordRecoveryLink
return

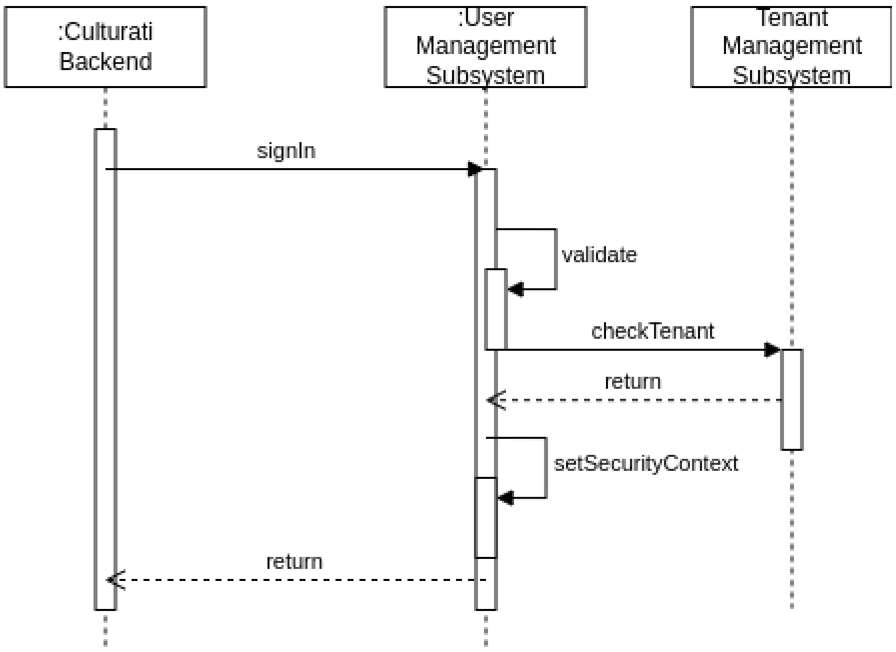
2.4.4.8 Reset Password Of User

Role: Admin



2.4.4.9 Sign in

Role: All roles



2.5 Tenant Management Subsystem

2.5.1 General Information

The tenant management submodule facilitates the administration and control of tenant-specific settings, configurations, and resources within the multi-tenant system. Its primary purpose is to provide features and functionalities that enable efficient management of multiple tenants while maintaining data isolation, security, and customization for each tenant.

2.5.2 Use Cases

2.5.2.1 Edit a tenant

Actors	Super Administrator, Administrator	
Goals	To allow the user with the authority to change the properties of a tenant.	
Preconditions	The user has the necessary permissions to edit their own tenant profile.	
Steps	*Actor actions*	*System responses*
	1. Navigate to the tenant screen. 2. Click the *change tenant configuration* option. 4. Update the fields. 5. Click the *confirm* option.	3. Display a form with fields of the tenant. 6. Validate the entered information and check for any missing or erroneous data. 7. Save the tenant in the storage system. 8. Confirm the successful update operation and provide visual feedback or notification to the user.
Post conditions	➔ The user's tenant profile is successfully updated with the edited information. ➔ The system reflects the changes made in the user's tenant profile across relevant modules or features.	

2.5.2.2 Add a tenant

Actors	Super Administrator	
Steps	*Actor actions*	*System responses*
	1. Navigate to the tenant management screen. 2. Click the *add tenant* option. 4. Fill in the form. 5. Submit the form.	3. Display the tenant creation form. 6. Validate the new tenant. 7. Generate a unique identifier for the new tenant. 8. Store the tenant in the application storage. 9. Confirm the successful creation of the tenant and display a success message to the user.
Post conditions	➔ A new tenant is successfully added to the system. ➔ The newly created tenant can be accessed and managed by the tenants' users.	

2.5.2.3 Delete a tenant

Actors	Super Administrator	
Steps	*Actor actions*	*System responses*
	1. Navigate to the tenant management screen. 2. Click the *delete tenant* option. 4. Confirm the action.	 3. Display a confirmation prompt. 5. Mark the tenant as deleted. 6. Revoke access of the users belonging to the tenant. 7. End any scheduled jobs of the tenant. 8. Confirm the successful deletion of the tenant and display a success message to the user.
Post conditions	➔ The tenant is successfully deleted from the system. ➔ Any related modules or features affected by the tenant's deletion are appropriately updated.	

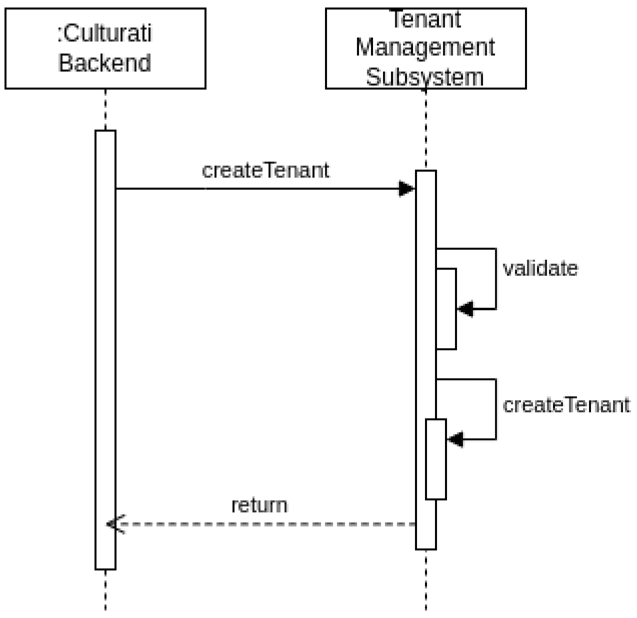
2.5.3 Design Decisions

Nothing written here

2.5.4 Sequences

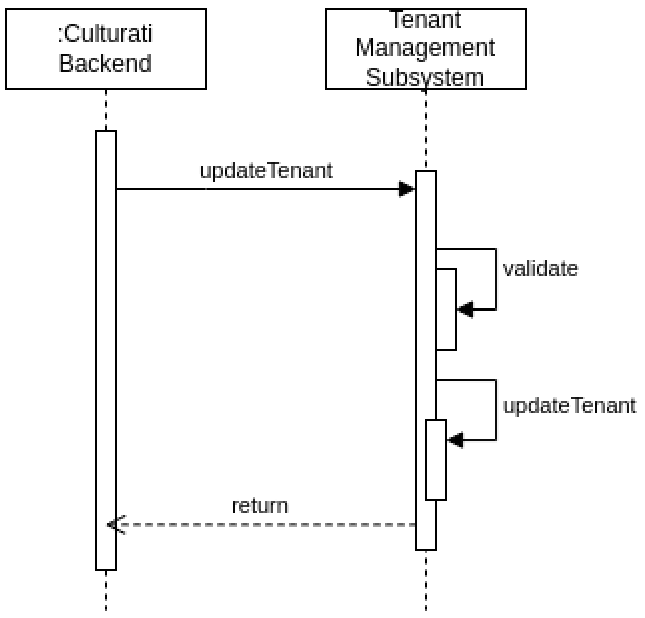
2.5.4.1 Create Tenant

Role: Super Admin



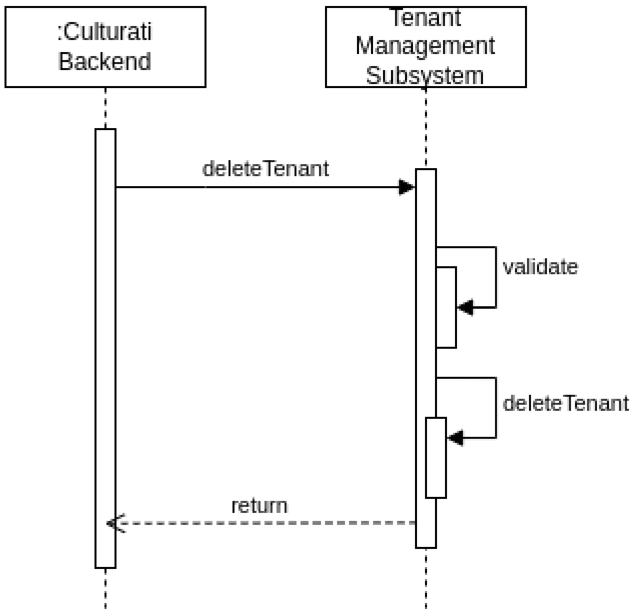
2.5.4.2 Edit Tenant

Role: Super Admin



2.5.4.3 Delete Tenant

Role: Super Admin



2.6 Artificial Intelligence Subsystem

2.6.1 General Information

An Artificial Intelligence Subsystem is a module within a software application that manages the AI capabilities according to the specific task that is going to make use of AI. These capabilities may be generative (text generation, image generation, audio generation, etc.) or non-generative (classification, forecasting, computer vision, anomaly detection, etc.). The primary purpose of the AI Subsystem is to control the interaction between AI models and requests and serve model outputs according to the task a given request specifies.

2.6.2 Use Cases

2.6.2.1 Path Planning and Load Optimization

The AI Subsystem together with IoT capabilities (coming from the IoT Subsystem) will be utilized to balance the number of people in various locations according to area limitations and user interests to be able to enhance the overall user experience. While doing this, the AI Subsystem will take the user interests into account to generate optimal paths for users.

Actors	Visitor	
Goals	To get a visit path recommendation from the system that matches the interests of the actor.	
Steps	*Actor actions*	*System responses*
	1. Request a path recommendation. 3. Choose the relevant (sub)categories.	2. Ask for current user interests by displaying categories and sub-categories. 4. Get data from AI Subsystem on the current visitor load at each site. 5. Get data from MainDB on the categorized content. 6. Apply the path recommendation algorithm to choose the best path (it can

	7. Accept the recommendation. If it is rejected the process starts again until the user stops asking for a new one.	be one of the previously used and voted paths) for the user.
Post conditions	➔ The recommended path is stored to be voted on by the user. ➔ The path is added to the system's previously used path set.	

2.6.2.2 Content Creation and Management

The AI Subsystem will utilize models to help manage internal archives in a structured manner to be able to deliver customized information to end-users based on their interests. The structured management of internal archives will also help with the content creation process as end users will be able to see the exact categories their content belongs to.

Actors	Curator	
Goals	To automate the content creation process and make it as efficient as possible.	
Steps	*Actor actions*	*System responses*
	1. Add new content (in text, image, audio, or video form). 4. The suggested categories relations, and labels are accepted by the curator.	2. The content is transcripted if required (image, audio, and video). 3. The content is a. automatically categorized and labeled b. associated to the existing content in MainDB by using previously set relation types.
Post conditions	➔ New content is added to the DB along with the transcriptions of each item. ➔ The accepted labels, categories, and relations are stored in the database.	

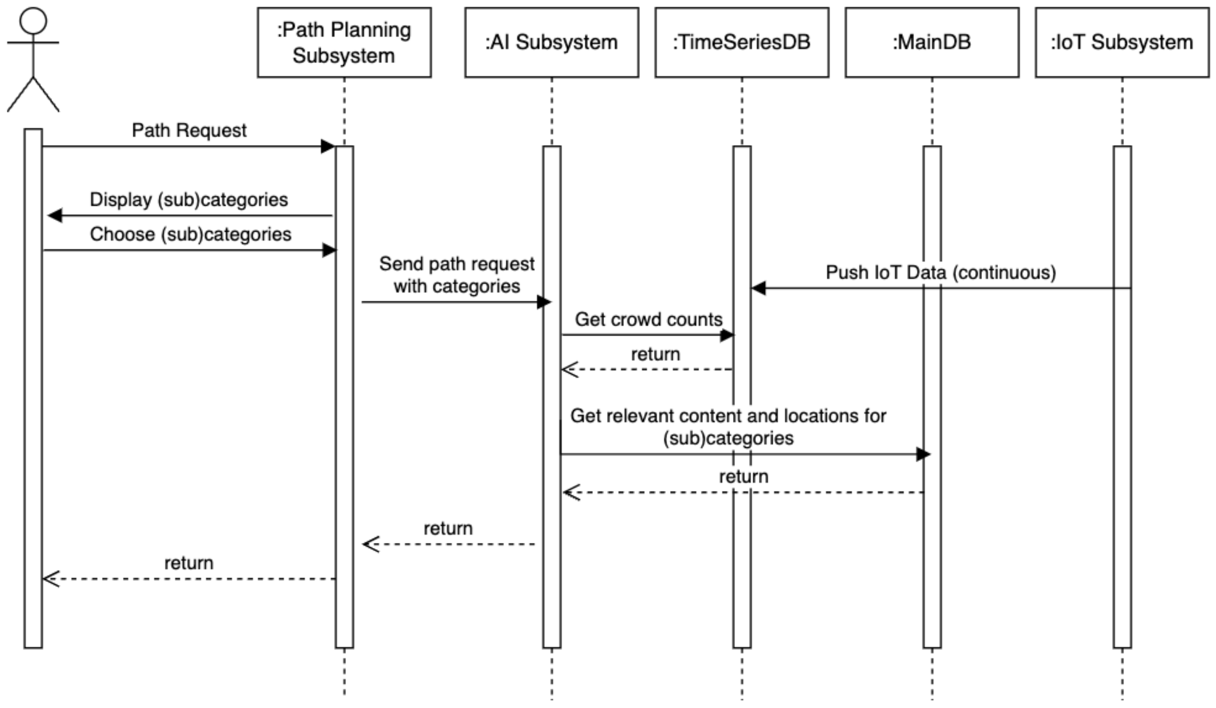
2.6.3 Design Decisions

For the first use case, we assumed that no user preference data is stored within the system unless requested. Hence, if the users allow us to do so, the path generation process can be further automated by using the previously visited and/or voted paths of the user.

2.6.4 Sequences

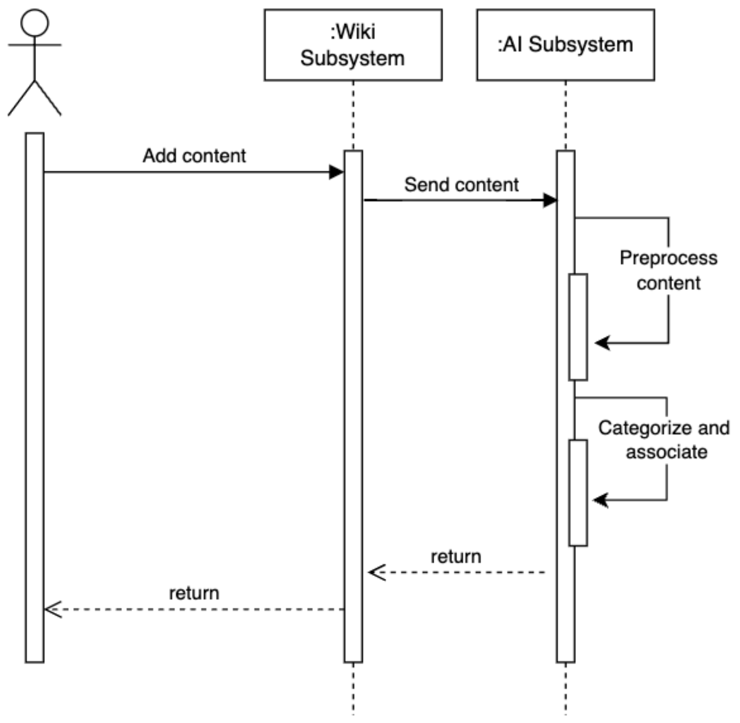
2.6.4.1 Recommend Path

Role: Visitor



2.6.4.2 Content Management Helper

Role: Curator



2.7 Educational Subsystem (Gamification Module)

2.7.1 General Information

The primary function of the Gamification Module is to generate quiz questions related to the items and exhibits to provide a more interactive and enjoyable experience for the visitors of the area allowing them to engage with the items and exhibits in a meaningful and educational manner.

The Gamification Module interacts with the AI Subsystem, which is responsible for question generation. The AI Subsystem dynamically creates quiz questions, which are then reviewed and approved by curators. These questions are specifically tailored to the items and exhibits within the space, ensuring a relevant and immersive learning experience for the visitors.

2.7.2 Use Cases

2.7.2.1 Add a question to an item in a category

Actors	Curator	
Goals	To enable the curator to add new questions to a specific category within the system.	
Preconditions	➔ The curator has identified a category to which the new questions will be added. ➔ The identified category exists in the system.	
Summary	The user adds a new question to a category within the system by finding the specific category and creating a new question.	
Related use cases	**Includes:** ➔ View item details	
Steps	*Actor actions*	*System responses*
	1. Included use case: View item details 2. Click the *add a question* option. 4. Select the specific category the question will be added about.	3. Display the available categories of the tenant. 5. Display a question form for the

	6. Write a question. 7. Add answers to the question. 8. Indicate one of the answers as the right answer. 9. Set a point when the question is answered correctly. 10. Click the *save* option.	user to write a question and add answers to the question. 11. Validate the entered information and check for any missing or erroneous data. 12. Save the question in the storage system. 13. Create relations between the saved questions and the selected item or exhibit. 14. Confirm the successful update operation and provide visual feedback or notification to the user.
Post conditions	The system has updated its storage to include the newly added question in relation to the specific item and category.	

2.7.2.2 Add categories to a question

Actors	Curator	
Goals	To enable the user to add categories to an existing question.	
Preconditions	→ The curator has identified a question to which the category will be added. → The identified question and category exist in the system.	
Summary	The user creates a relationship between a category and a question.	
Related use cases	**Includes:** → View questions related to an item	
Steps	*Actor actions*	*System responses*
	1. Included use case: View questions related to an item 2. Select the specific question to add the category. 4. Select a category. 5. Click the *save* option.	3. Display the categories of the tenant. 6. Validate the entered information and check for any missing or erroneous data. 7. Create relations between the saved questions and the category. 8. Confirm the successful update operation and provide visual feedback or notification to the user.
Post conditions	The system updates its database or storage to associate the selected categories with the question.	

2.7.2.3 Generate questions

Actors	Curator	
Goals	To provide the user with an assistant that suggests questions about certain items and/or exhibits.	
Summary	A curator generates questions using an AI-powered system to enhance visitor engagement and learning experiences.	
Related use cases	**Extension of:** ➔ Content Management Helper	
Steps	*Actor actions*	*System responses*
	1. Select a particular item or exhibit. 2. Click the *generate question* option.	
		3. Generate a set of questions. 4. Display the questions to the curator.
	5. Review the questions. 6. Select approved questions.	
		7. Validate the entered information and check for any missing or erroneous data. 8. Save the question in the storage system. 9. Create relations between the saved questions and the selected item or exhibit. 10. Confirm the successful update operation and provide visual feedback or notification to the user.

Post conditions	➔ The generated questions are stored within the system for future reference. ➔ The curator's selected item or exhibit is associated with the generated questions in the system's database.

2.7.2.4 View questions related to an item

Actors	Curator	
Steps	*Actor actions*	*System responses*
	1. Select a particular item or exhibit to view. 2. Click the *display questions* option.	3. Query and retrieve questions related to the specified item or exhibit. 4. Display the questions associated with the particular item or exhibit to the user.

2.7.2.5 Revise a question

Actors	Curator	
Goals	To provide the user an option to revise previously generated questions.	
Preconditions	A previously generated question exists that requires revision.	
Summary	The user modifies the question's wording, content, or structure to their liking.	
Related use cases	**Includes:** ➔ View questions related to an item	
Steps	*Actor actions*	*System responses*
	1. Included use case: View questions related to an item 2. Select a question to revise. 4. Change the question by changing the fields in the form. 5. Click the *save* option.	3. Display a question revision form. 6. Validate the entered information and check for any missing or erroneous data. 7. Save the question in the storage system. 8. Confirm the successful update operation and provide visual feedback or notification to the user.
Post conditions	The revised question is saved within the system or platform, replacing the previous version.	

2.7.2.6 Delete a question

Actors	Curator	
Goals	To remove the question from the system so that the users are not presented with the question.	
Preconditions	A question exists that needs to be deleted.	
Steps	*Actor actions*	*System responses*
	1. Included use case: View questions related to an item 2. Select a question to delete. 4. Approve the deletion process.	3. Request for final approval. 5. Delete the question from the system.
Post conditions	The selected question is deleted from the system or platform.	

2.7.2.7 Visitor answers questions

Actors	Visitor	
Goals	To make the visitor answer the question on the item/exhibit s/he is visiting and show the correct answer.	
Preconditions	A question exists for the item being visited.	
Steps	*Actor actions*	*System responses*
	1. Visitor requests a question for a specific item. 3. Visitor answers the question. 5. Visitor requests another question or stops the game.	2. System chooses and shows one of the questions for the item. 4. System shows the correct answer.
Post conditions	The daily and global score of the visitor is updated.	

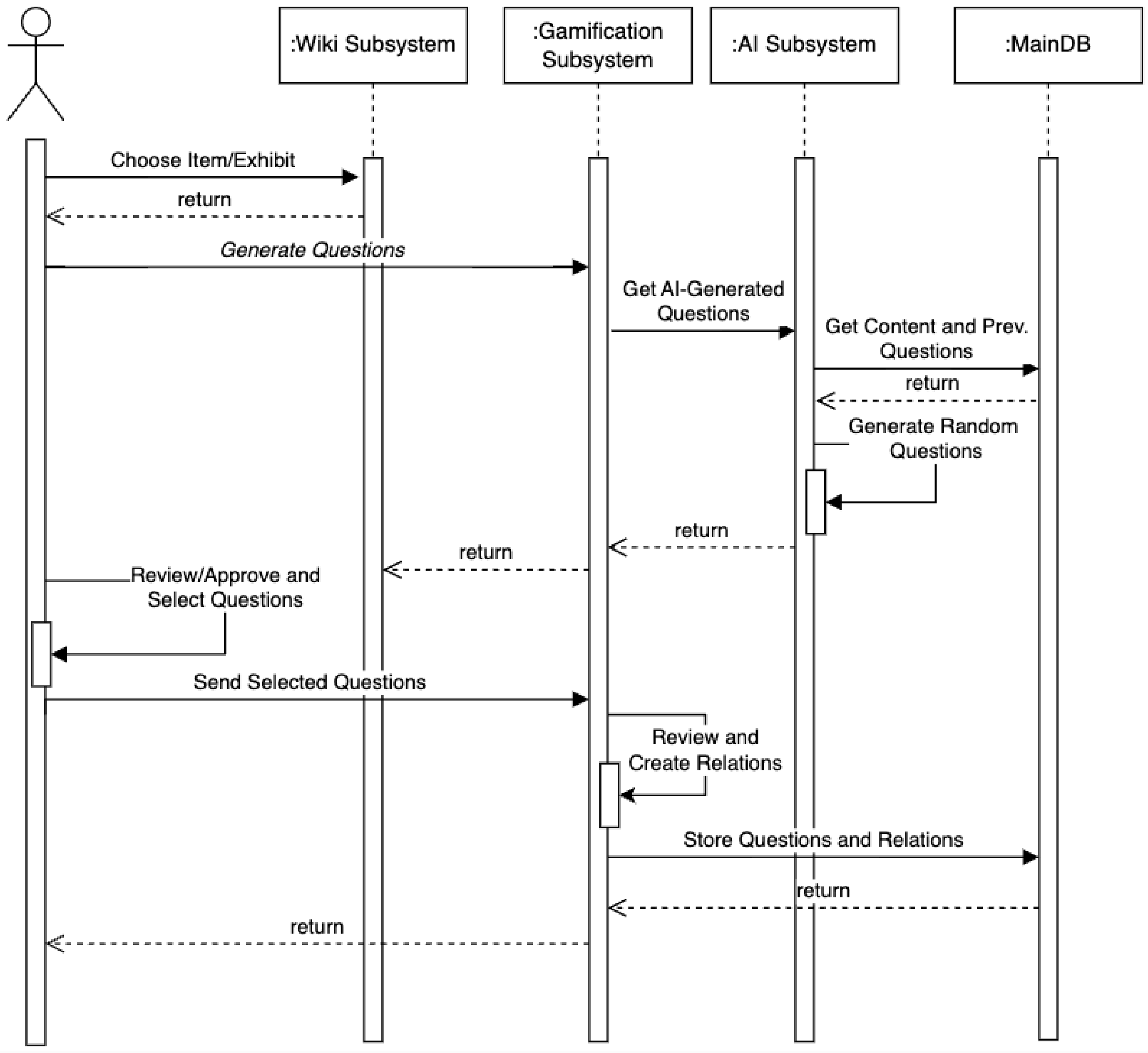
2.7.2.8 Gameplay

Actors	Visitor	
Goals	To play a game	
Preconditions	At least one question exists for the items. The system knows the user's interested category and level.	
Steps	*Actor actions*	*System responses*
	1. Visitor requests to play a game	
		2. System chooses a question according to the user's input and displays it on the screen 3. System displays the path to the related item/work.
	4. Visitor follows the route and finds the information about the related item 5. Visitor answers the question.	
		6. If the answer is, a. correct, the system awards the user with points/badges etc. b. not correct, the system shows a hint to the visitor and goes back to step 5.
	7. Visitor requests another question (back to step 2) or stops the game.	
Post conditions	The daily and global score of the visitor is updated.	

2.7.3 Sequences

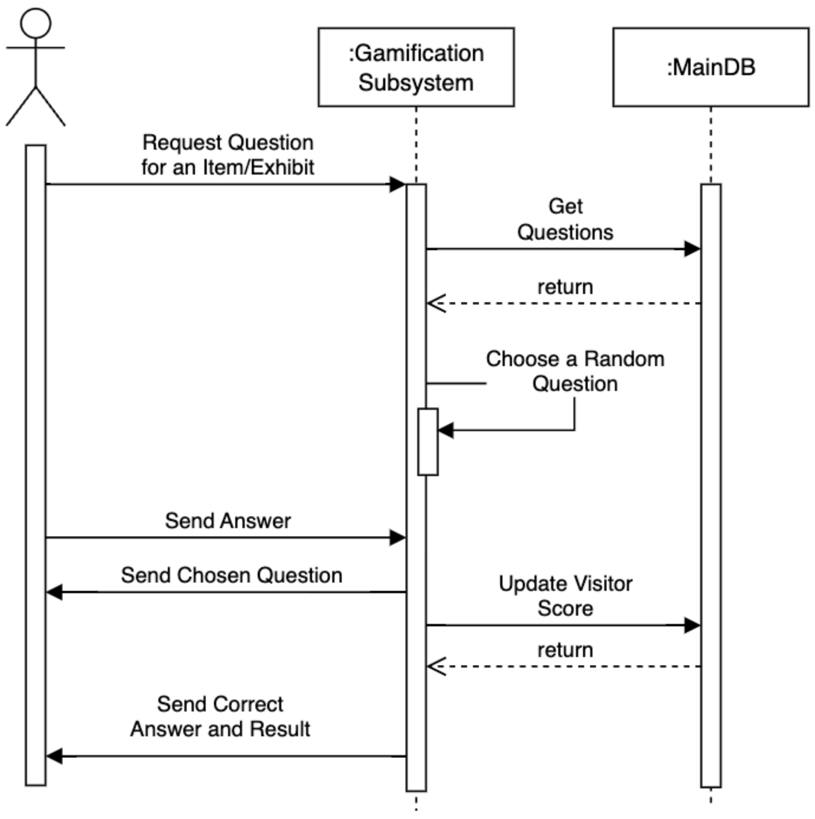
2.7.3.1 Curator generates questions

Role: Curator


:Wiki Subsystem
:Gamification Subsystem
:AI Subsystem
:MainDB
Choose Item/Exhibit
return
Generate Questions
Get AI-Generated Questions
Get Content and Prev. Questions
return
Generate Random Questions
return
return
Review/Approve and Select Questions
Send Selected Questions
Review and Create Relations
Store Questions and Relations
return
return

2.7.3.2 Visitor answers questions

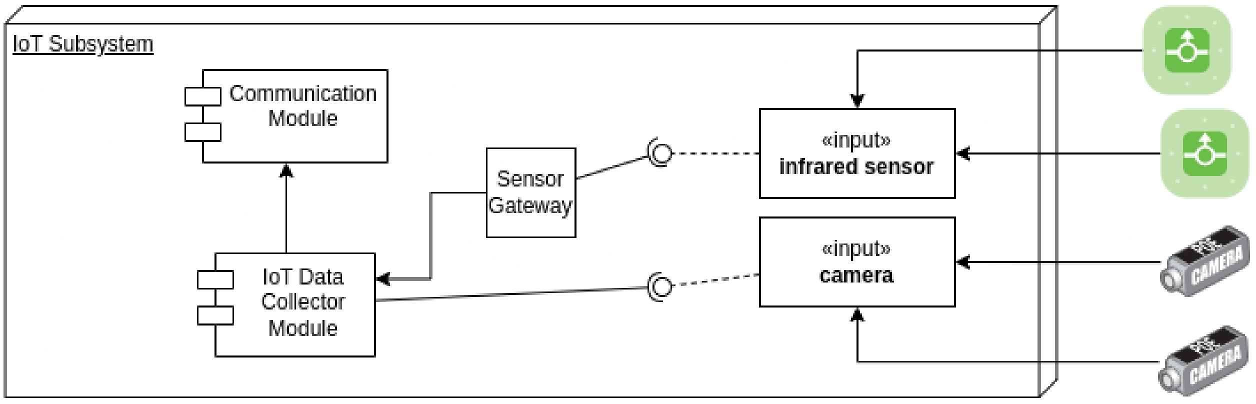
Role: Visitor



2.8 IoT Subsystem

2.8.1 General Information

The overall architecture of the IoT subsystem is shown below. The main goal of the IoT subsystem is to collect real-time data and send the information extracted from this data to the IoT module in the main backend.



The sensors that will be used in the project are infrared sensors and cameras to get the crowd data. In addition to these, the system will be flexible enough to have other types of sensors that measure temperature and CO2. All these sensor data will be first forwarded to an IoT data Collector Module (e.g., OPCRouter) then it will be forwarded to the IoT Module by the Communication Module to be ingested to Time Series DB.

2.8.2 Use Cases

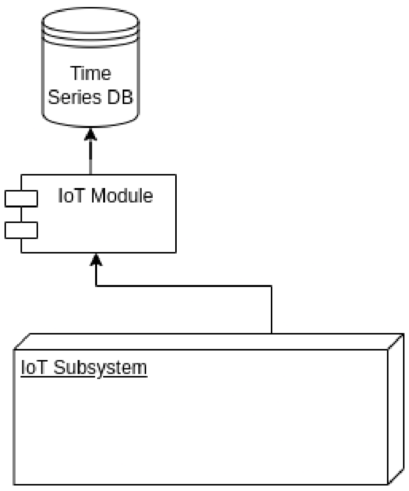
2.8.2.1 Individuals entering the area equipped with crowd sensors

Actors	Physical Sensors	
Goals	To save the sensor data	
Preconditions	Crowd sensors are appropriately installed and calibrated in the designated area.	
Summary	A curator generates questions using an AI-powered system to enhance visitor engagement and learning experiences.	
Steps	*Actor actions*	*System responses*
	1. Data are sent from sensors	

	2. The crowd sensors detect the presence and movement of individuals within their range. 3. As people move, the sensors capture their motion data and transmit it to the IoT subsystem.	4. The IoT subsystem stores the raw data. 5. The IoT subsystem analyzes the sensor data, identifies patterns, and determines the presence of a crowd. 6. If the crowd presence is confirmed this information will be sent to the main backend through the webhook provided by the IoT module.
Post conditions	1. The crowd motion detection system successfully detects and confirms the presence of people entering the monitored area. 2. Crowd data is successfully stored.	

2.9 IoT Module

2.9.1 General Information



The modules/entities with which the IoT module interacts with the IoT subsystem are shown below. The IoT module receives all the data from the IoT Subsystem. As mentioned above the communication will be performed via the Communication Module with an appropriate protocol (e.g., Line Protocol). The data ingested to the streaming database, Time Series DB, (e.g., InfluxDB) will be available to Admin via an interactive visualization tool (e.g., Grafana) that supports geo mapping to show the current and past state of the site with respect to a type of telemetry data (e.g., number of people and temperature). In addition, the Time Series DB will serve the momentary and past data to the AI Subsystem to be used in its tasks such as path recommendation.

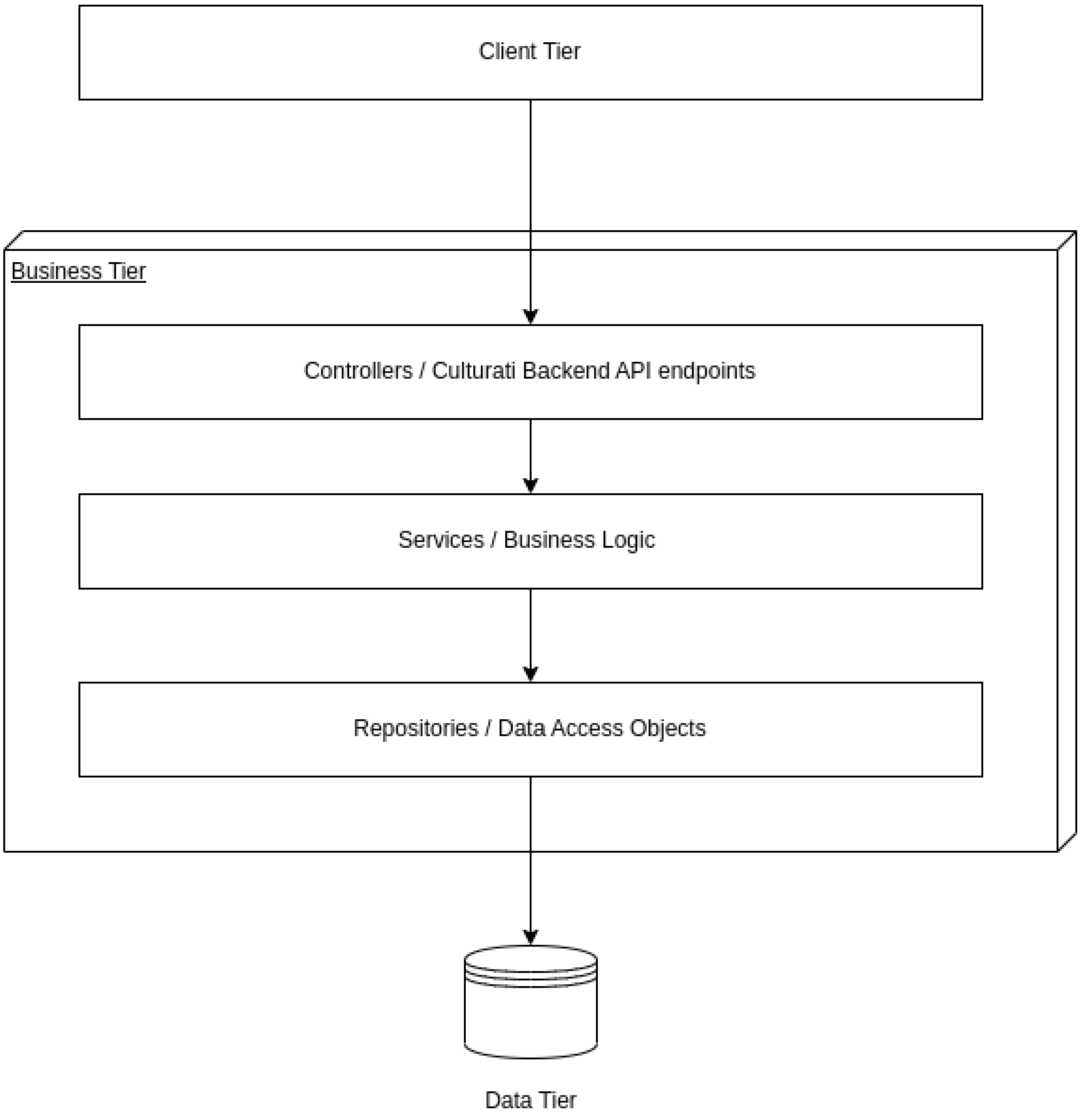
2.9.1.1 Individuals entering the area equipped with crowd sensors

Actors	Physical Sensors	
Goals	To save the sensor data	
Preconditions	Crowd sensors are appropriately installed and calibrated in the designated area.	
Summary	A curator generates questions using an AI-powered system to enhance visitor engagement and learning experiences.	
Steps	*Actor actions*	*System responses*
	1. Data are sent from sensors 2. The crowd sensors detect the presence and movement of individuals within their range. 3. As people move, the sensors capture their motion data and	

	transmit it to the IoT subsystem.	4. The IoT subsystem stores the raw data. 5. The IoT subsystem analyzes the sensor data, identifies patterns, and determines the presence of a crowd. 6. If the crowd presence is confirmed this information will be sent to the main backend through the webhook provided by the IoT module.
Post conditions		

3.Design Patterns and Approaches

The application employs the Spring MVC architecture with a client-server model, where a Spring Boot application serves as the backend and a React application serves as the frontend for both the mobile application and the administrator application.


Client Tier
Business Tier
Controllers / Culturati Backend API endpoints
Services / Business Logic
Repositories / Data Access Objects
Data Tier

3.1 Backend

The backend will be developed using the Spring Boot framework. The backend architecture will include the following components:

Controllers: Controllers in Spring Boot handle incoming HTTP requests from both the mobile application and administration application. The requests are received through controller endpoints where the controller executes the corresponding business logic.

Webhooks: In addition to its core functionality, the application offers webhooks as a means of integrating with external modules, such as the AI subsystem. Webhooks are endpoints exposed by the application that allow external systems or modules to receive real-time notifications or callbacks from the application. The application may make requests to an external module to perform certain tasks or computations. After processing the requests, the external module can utilize the webhooks provided by the application to notify or send the results back to the application.

Services: Services encapsulate the business logic of the application. They handle complex operations, interact with data repositories, and perform necessary validations or calculations. Services are responsible for implementing the application's rules and workflows.

Data Access Layer: The data access layer consists of repositories or DAOs (Data Access Objects) that interact with the database or other data storage systems. They provide CRUD (Create, Read, Update, Delete) operations and abstract the underlying data access logic.

3.1.1 Data synchronization between different data sources

To address synchronization challenges, the system will employ an **eventual consistency** approach. The system's replica nodes will originate from the duplicate entity representations in the content management database and the main database. By adopting eventual consistency, the system aims to ensure that all replicas eventually reach a consistent state, despite potential temporary inconsistencies.

To achieve this, the system will implement periodic synchronization of created or updated nodes on a daily basis. This synchronization process will reconcile any differences between the replica nodes, ensuring they converge towards a consistent state over time. Additionally, the system will provide users with the option to manually trigger synchronization if immediate consistency is required, granting them control over the synchronization process.

By utilizing eventual consistency, the system strikes a balance between synchronization efficiency and maintaining data integrity. It acknowledges that temporary inconsistencies may arise but ensures that over time, all replica nodes will align and reach a consistent state.

3.1.2 Crosscutting concerns

Aspect-Oriented Programming will be used as a technique or approach used to address crosscutting concerns in the application's architecture. AOP provides a way to encapsulate and separate crosscutting concerns from the core business logic, improving modularity and maintainability.

3.1.2.1 Logging

The system will integrate with a logging framework, such as Logback or Log4j, to provide robust and customizable logging capabilities. The logging configuration will be defined to capture appropriate log levels, including debug, info, warning, and error. Logs will be generated for critical operations, exceptional conditions, and important events. The logs will include relevant contextual information, such as timestamps, log levels, source components, and descriptive messages. Additionally, a log rotation strategy will be implemented to manage log files and prevent them from consuming excessive disk space.

3.1.2.2 Security

Security is a vital concern to protect sensitive data and ensure authorized access to the application. The application will employ Spring Security, a widely adopted security framework, to implement authentication, authorization, and access control mechanisms. User authentication will be implemented using standard techniques such as username/password-based authentication or token-based authentication (e.g., JWT). Role-based access control will be enforced to restrict access to specific resources or functionalities based on user roles. Security configurations, including secure communication over HTTPS, session management, and protection against common security vulnerabilities (e.g., cross-site scripting, cross-site request forgery), will be implemented to mitigate potential security risks.

3.1.2.3 Auditing

The application will incorporate auditing capabilities for the administration application to track and log important activities, ensuring accountability and compliance. Audit logs will capture critical events such as user actions, system configuration changes, and data modifications. The audit logs will include relevant information such as user identity, timestamp, action performed, and affected resources.

3.1.2.4 Transaction Management

The application will leverage Spring's transaction management capabilities, which provide declarative transaction demarcation and coordination. Appropriate transaction boundaries will be defined, allowing operations to be executed atomically. Transaction isolation levels and propagation rules will be configured based on the specific requirements of each operation. Error handling and rollback mechanisms will be implemented to handle exceptional conditions and ensure data consistency in case of failures.

3.1.2.5 Caching

To improve application performance and reduce the load on data sources especially when generating tour paths, caching will be implemented for frequently accessed data. The application will integrate with caching frameworks such as Ehcache, Caffeine, or Redis. Caching strategies, such as cache eviction policies and cache update strategies, will be defined based on the characteristics of the data. Caching annotations will be used to mark methods or data sources that can be cached.

3.2 Frontend

The frontend of the application will be developed using React, a widely adopted JavaScript library renowned for its ability to create dynamic and interactive user interfaces. Complementing React, Next.js, a robust React framework, will be utilized to build a scalable and production-ready web application.

3.2.1 Design Patterns and Approaches

3.2.1.1 Component Based Architecture

The user interface will be structured by breaking it down into reusable components that encapsulate specific functionality. This approach promotes code reusability, modularity, and easier maintenance. Components will be composed to build complex UIs, facilitating flexibility and scalability.

3.2.2.4 Modular CSS

The implementation of modular CSS serves as an extension of the component-based architecture, presenting a systematic methodology for styling. Custom styling will be implemented in dedicated CSS files, prioritizing improved maintainability, scalability, and modularity.

3.2.1.2 Modular CSS

The implementation of modular CSS serves as an extension of the component-based architecture, presenting a systematic methodology for styling. Custom styling will be implemented in dedicated CSS files, prioritizing improved maintainability, scalability, and modularity.

3.2.1.3 Externalized Configuration

The system will utilize externalized configurations to allow the system to be deployed using the same codebase with different configurations resulting in system-wide changes to be made without changing the codebase. Furthermore, it simplifies the process of altering visual aspects, such as styling, within the system. This decoupling of configurations from the codebase streamlines the deployment process and minimizes the risk of introducing errors.

3.2.1.4 Higher-Order Components (HOCs)

Higher-Order Components (HOCs) provide a powerful mechanism for encapsulating common functionalities, reusing behaviors across components, extracting cross-cutting concerns, composing multiple behaviors, and decoupling component logic from presentation. These capabilities enhance code reusability, modularity, maintainability, and flexibility within frontend applications.

- ➔ HOCs enable the system to encapsulate and abstract common functionalities or behaviors, which can then be applied to multiple components. By wrapping components with HOCs, these behaviors can be reused throughout different parts of the application, eliminating the need for redundant code.
- ➔ Furthermore, HOCs are valuable in extracting cross-cutting concerns, such as authentication, data fetching, or state management. This approach helps keep the component code focused, improving its readability and maintainability.
- ➔ The composability of HOCs is another advantage they offer. Multiple HOCs can be stacked together, with each adding a distinct aspect to the component's behavior. This compositional nature allows for the creation of complex components by combining smaller, reusable HOCs. As a result, modularity and flexibility are promoted within the application's architecture.
- ➔ One significant benefit of HOCs is their ability to facilitate the decoupling of concerns. They achieve this by separating the logic of a component from its

presentation. By abstracting shared functionalities into HOCs, components can focus on rendering UI and handling user interactions while relying on the HOCs to manage underlying behaviors.

3.2.2 State Management

Redux will be utilized for efficient state management in the frontend. It will provide a centralized store to manage application state, enabling a predictable state container. Actions and reducers will be implemented to handle state changes, ensuring a consistent data flow and facilitating easier debugging and testing.

3.2.3 API Communication

Asynchronous data fetching will be handled using Axios, a popular JavaScript library for making HTTP requests. Axios seamlessly integrates with the project, providing a simple and efficient way to retrieve and update data from the backend APIs. It will enable smooth communication and data synchronization between the frontend and backend.

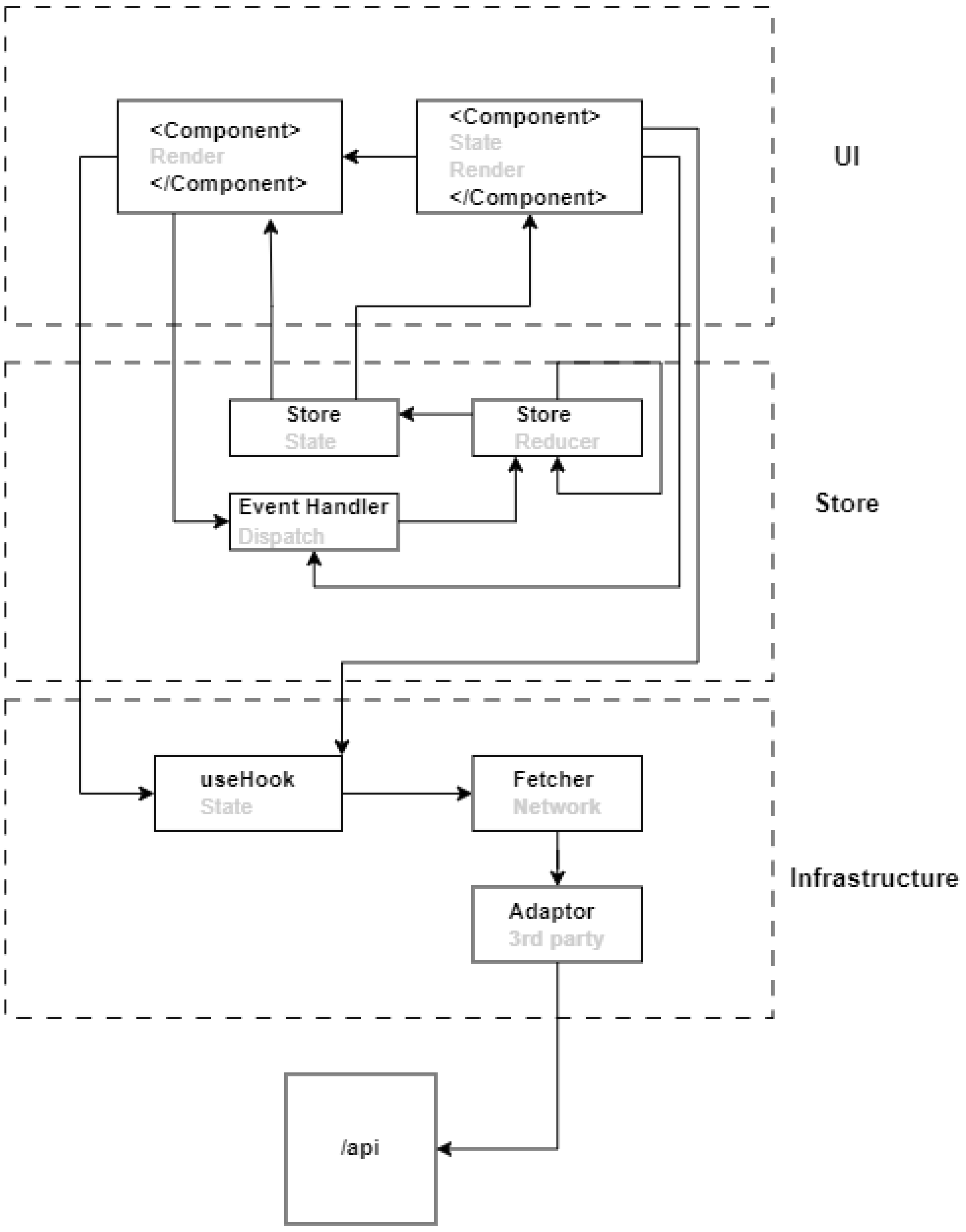
3.2.3.1 Routing

To handle navigation and different views within the frontend, the project will incorporate a client-side routing pattern. Client-side routing ensures that the user interface dynamically updates based on the current URL, allowing for smooth navigation between different sections of the application.
React Router will be used as the routing library to implement the client-side routing pattern. It provides a declarative way to define routes and associate them with specific components or views. React Router enables the application to maintain a single-page experience, where navigation occurs within the browser without requiring a full page refresh.

By adopting the client-side routing pattern, the frontend will deliver a responsive and interactive user experience, enabling users to seamlessly navigate between various sections or views of the application while preserving the state and context of the current page.

3.2.4 Styling

Styling will be implemented using Tailwind CSS, a utility-first CSS framework. Tailwind CSS provides a wide range of utility classes that facilitate rapid UI development and consistent styling. It simplifies the process of creating visually appealing and responsive user interfaces, ensuring a consistent and cohesive design across the application.


<Component>
Render
</Component>
<Component>
State
Render
</Component>
UI
Store
State
Store
Reducer
Event Handler
Dispatch
Store
useHook
State
Fetcher
Network
Adaptor
3rd party
Infrastructure
/api

3.3 Communication between Backend and Frontend

The communication between the backend and frontend occurs through RESTful APIs. The frontend sends HTTP requests to the backend API endpoints to retrieve data, submit forms, or perform other actions. The backend processes the requests, executes the necessary operations, and returns responses with the requested data or status codes.

3.4 Authentication and authorization

The system will utilize JWT for authentication and authorization purposes. JWT tokens are granted after the system authorizes the user and can be sent with subsequent requests to access protected resources. The system can then verify the authenticity of the token and extract user information from it.

Since tokens contain necessary information to prove authenticity and authorization, the backend of the system is stateless, meaning it does not need to maintain session information of the user. The statelessness of the system allows for horizontal scalability, where multiple instances of the system can handle requests without needing to share session data. Each instance can independently verify the JWTs, making it easier to distribute the application across multiple servers or use serverless architectures.

Additionally, JWTs can include expiration times, limiting their validity. These security features help protect against token forgery, tampering, and unauthorized access improving the system’s security.

4. User Experience Design

4.1 Design Decisions

4.1.1 Crowd Management

The UX design incorporates a map component to effectively address one of the critical aspects of the application: guiding users to less congested areas thereby managing the congestion. The map allows the user to navigate the area and find where crowds are located with ease. By making the map a central component of the application the design emphasizes the importance of crowd management.

4.1.2 Personalized Tours

The design incorporates prompts strategically placed throughout the application to encourage the user to try AI-generated, personalized tours. These prompts are displayed on the main screen when no active route is available to the user, ensuring maximum visibility and user engagement.

4.1.3 Accessibility

4.1.3.1 Text alternatives

Text alternatives enhance accessibility by providing textual information about visuals on the screen, enabling visually impaired users to engage with the visual content using screen readers on their devices. The system will enable content creators to add text alternatives to visual content and make this information accessible to screen readers.

4.1.3.2 Multi-language support

Multi-language support enables the application to be available in multiple languages, catering to users who speak different languages. The system allows users to select their preferred language from a list of languages. The language selection option is conveniently accessible from all screens of the application, ensuring access is one click away on all screens.

Multi-language support encapsulates both UI elements and the content serving the user the content in their preferred language. By providing a personalized language experience, the application enhances user satisfaction and enables effective communication and comprehension.

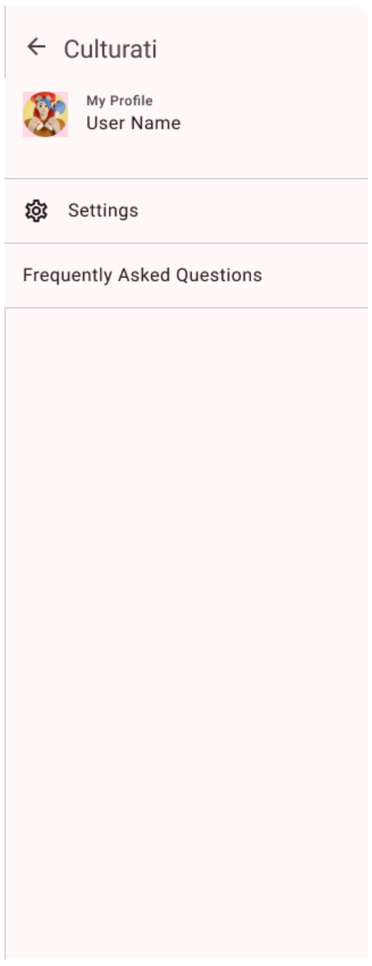
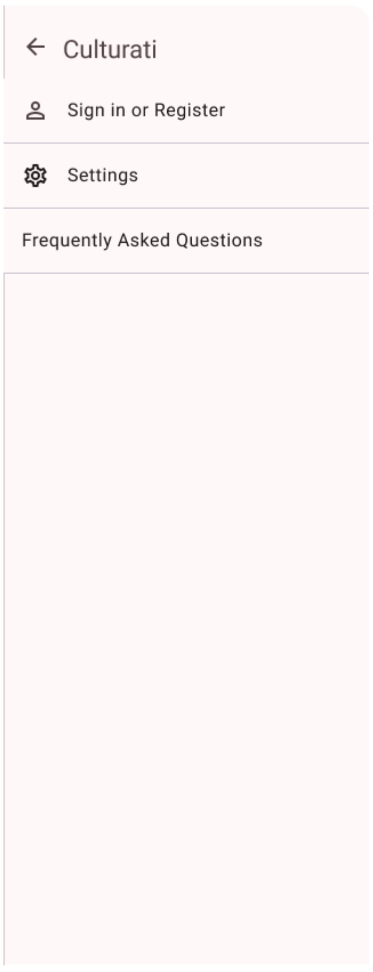
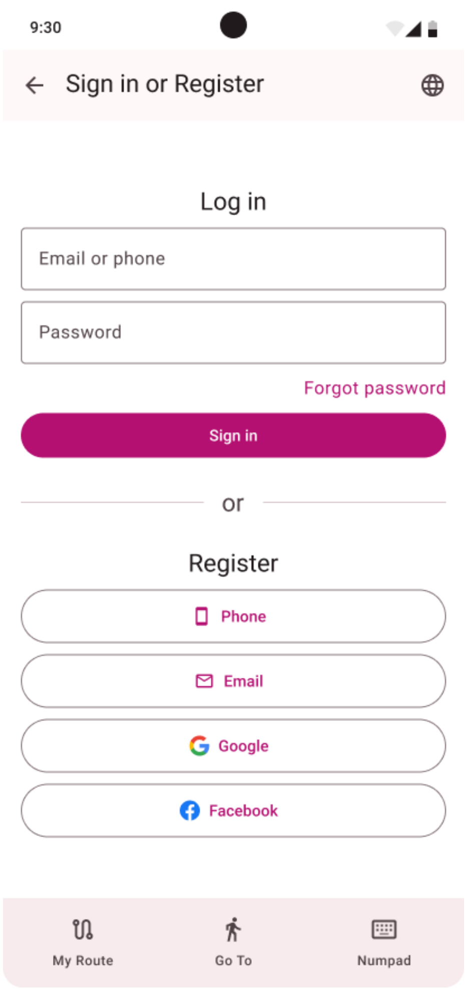
4.1.3.3 Color Contrast and Color Modes

The application utilizes color contrast and color modes to aid both visually impaired users and non-visually impaired users.

Color modes and contrast play a crucial role in making an application accessible to users with visual impairments or color vision deficiencies. By providing alternative color modes, such as high contrast or dark mode, users can customize the UI to their specific needs and preferences. Additionally, ensuring sufficient color contrast between text and background elements enhances readability for all users.

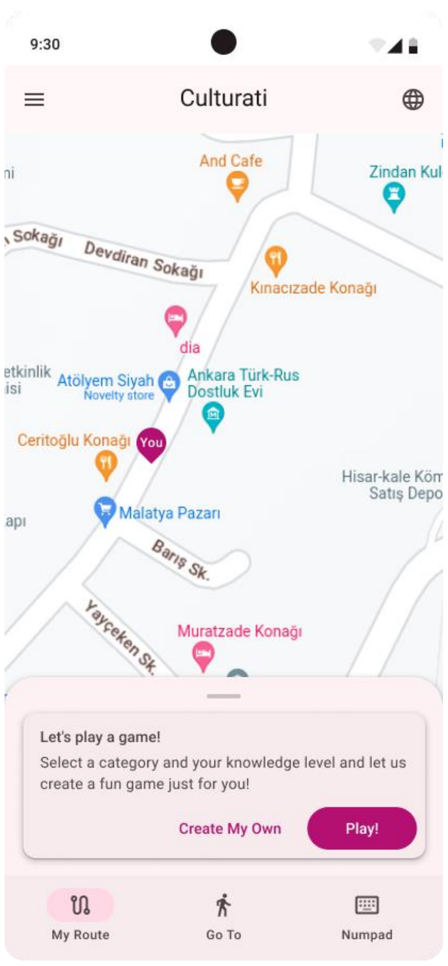
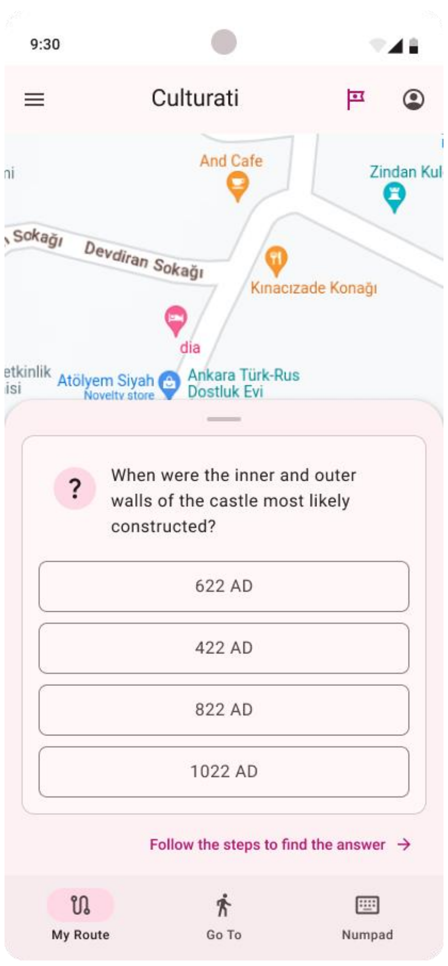
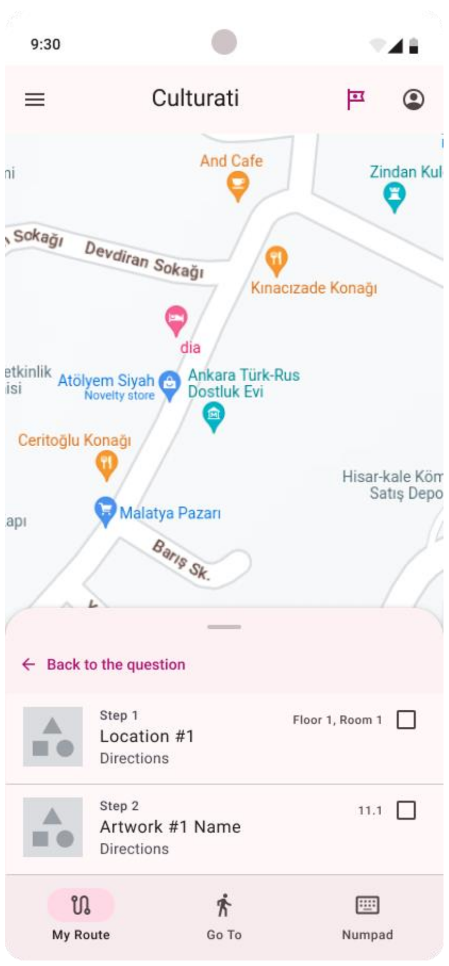
4.2 Mobile Application Mockups

4.2.1 User Management Components

 ← Culturati My Profile User Name Settings Frequently Asked Questions	 ← Culturati Sign in or Register Settings Frequently Asked Questions	 9:30 ← Sign in or Register Log in Email or phone Password Forgot password Sign in or Register Phone Email Google Facebook My Route Go To Numpad
The side sheet menu of a logged in user.	The side sheet menu of a logged out user.	A possible sign in and register screen.

The proposed system does not require the user to sign in with an email or phone number but for future consideration a sign in and register screen has been provided in this document.

4.2.2 Main screen and its modes

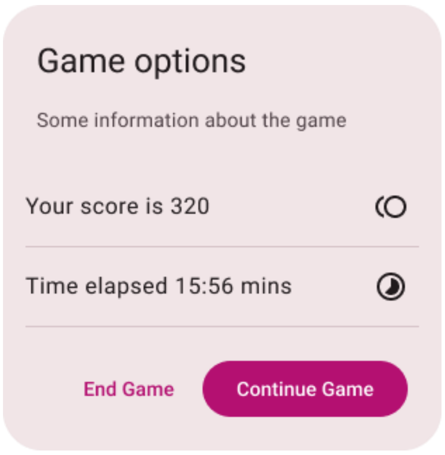
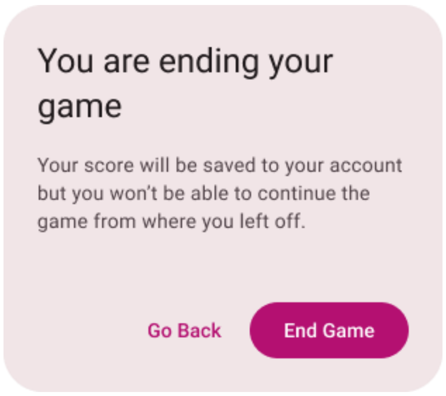
		
The main screen of the application when it’s first opened.	The application shows the quiz question on the bottom sheet in the main screen.	The application shows the route to an item on the bottom sheet in the main screen.

Let's play a game!
Select a category and your knowledge level and let us create a fun game just for you!

When were the inner and outer walls of the castle most likely constructed?

Step 1 Location #1 Directions	Floor 1, Room 1	☐
Step 2 Artwork #1 Name Directions	11.1	☐

Users can access the question and the steps to their next stop through the bottom sheet component. They can navigate between these views using slide gestures or utilize the labeled buttons: 'Follow the steps to find the answer' and 'Back to the question'. When the application is in *active route* mode, meaning the user is being guided through a route, the system utilizes a list view where the user can easily follow their previous and next stops.

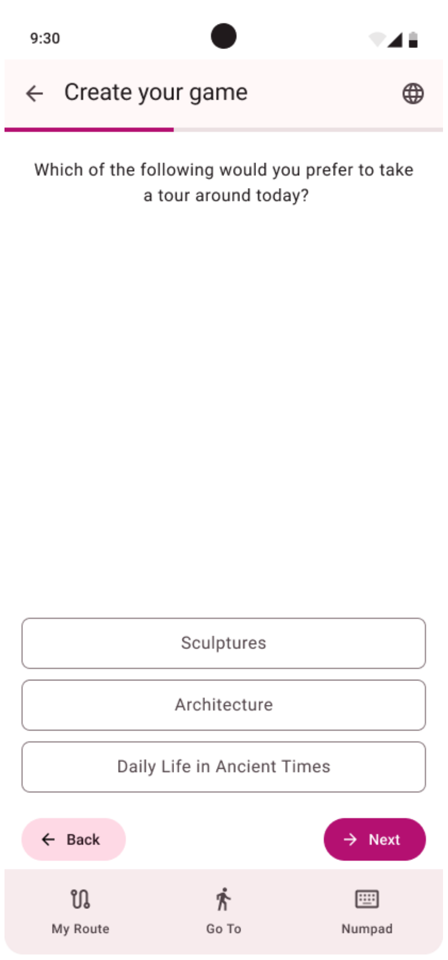
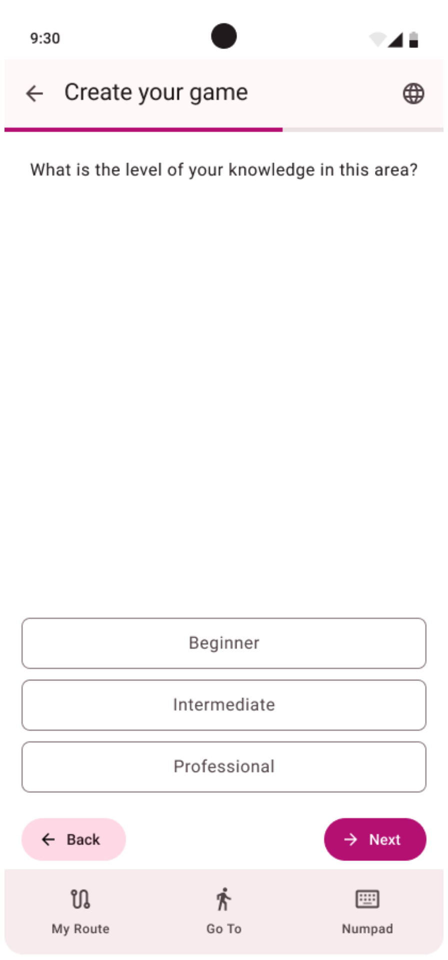
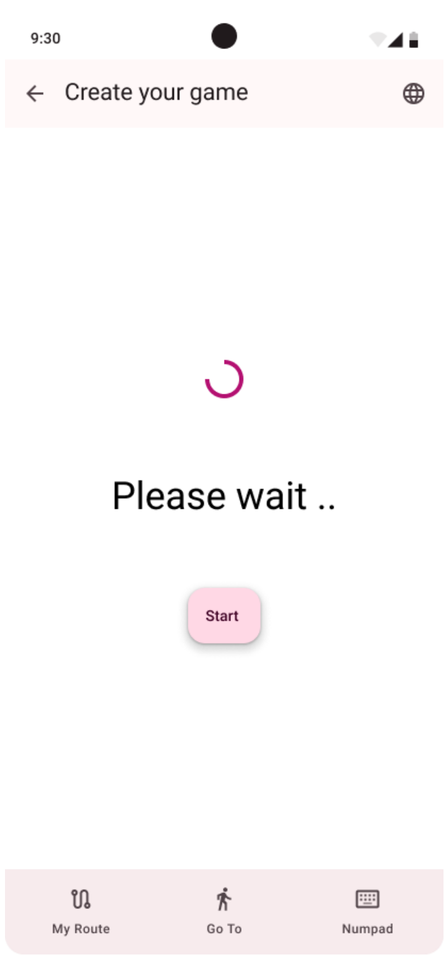
	
Game options modal showing the current score and elapsed time.	A prompt to confirm the user action of ending a game.

Game options: Some information about the game. Your score is 320. Time elapsed 15:56 mins. End Game. Continue Game.

You are ending your game: Your score will be saved to your account but you won't be able to continue the game from where you left off. Go Back. End Game.

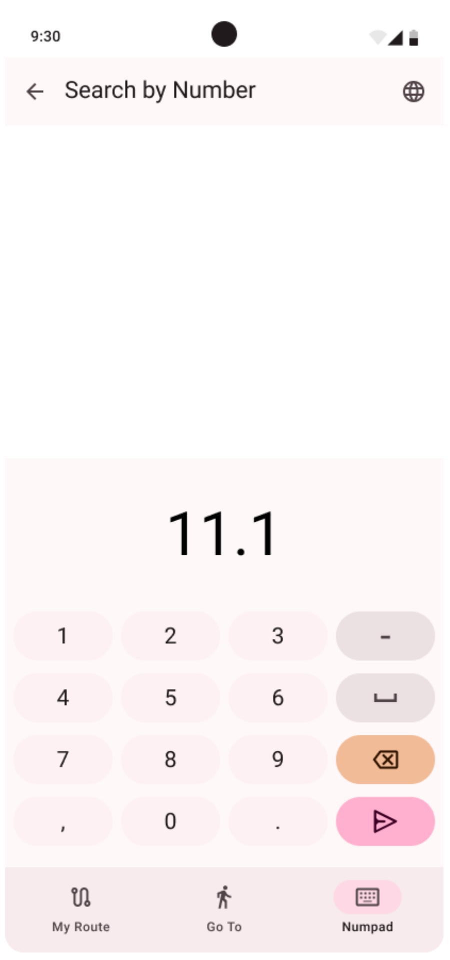
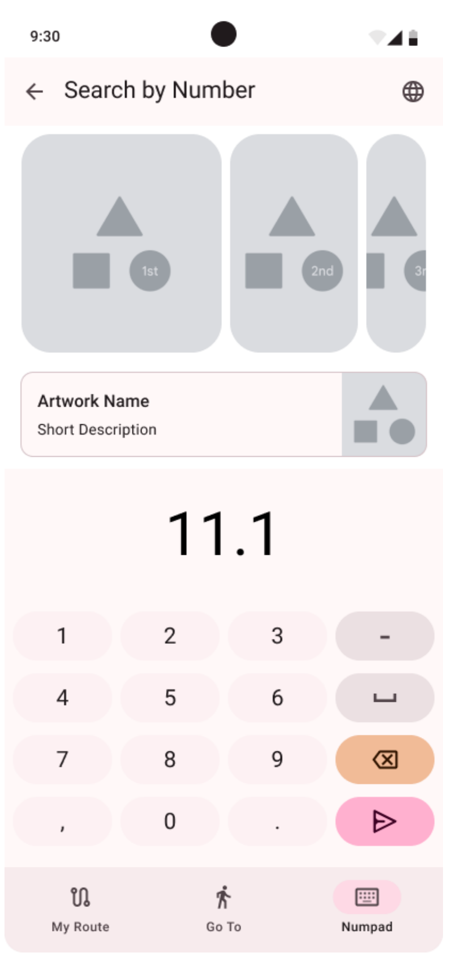
Game options modal is opened through the *game options button* that shows on the top navigation bar only when the user is playing a game.

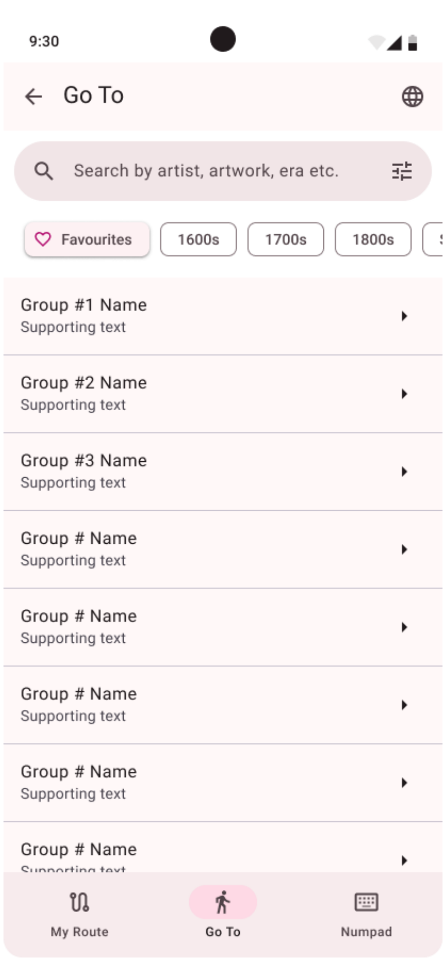
4.2.3 Game creation questions

 9:30 Create your game Sculptures Architecture Daily Life in Ancient Times Back Next My Route Go To Numpad Which of the following would you prefer to take a tour around today?	 9:30 Create your game Beginner Intermediate Professional Back Next My Route Go To Numpad What is the level of your knowledge in this area?	 9:30 Create your game Start My Route Go To Numpad Please wait ..
A question to learn the users interests. Categories are shown as answers.	A question to gauge the users knowledge level about the category they have selected.	Information screen that notifies the user while the AI subsystem is generating a game and a route.

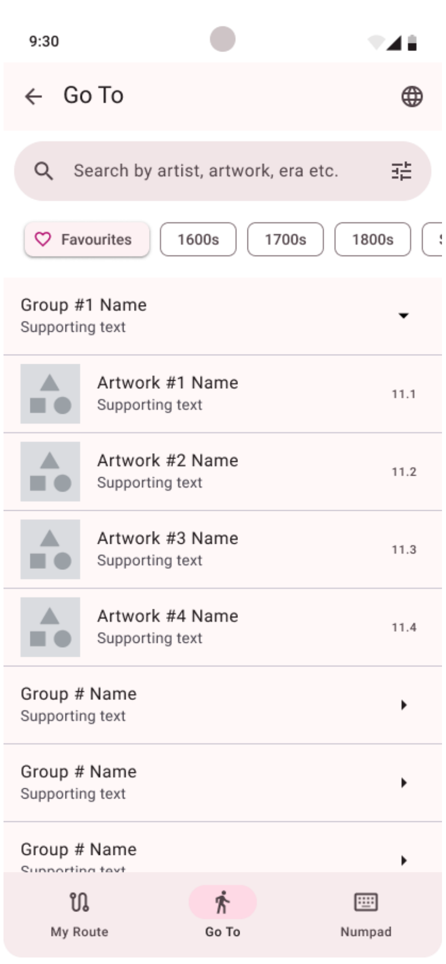
A progress bar is utilized in the question screens to display the user's ongoing progress.

4.2.4 Searching and filtering

 9:30 Search by Number 11.1	 9:30 Search by Number Artwork Name Short Description 11.1
The numpad screen is used to search the tenant catalog, utilizing distinct numbers assigned to items and exhibits.	The numpad screen shown with search results.

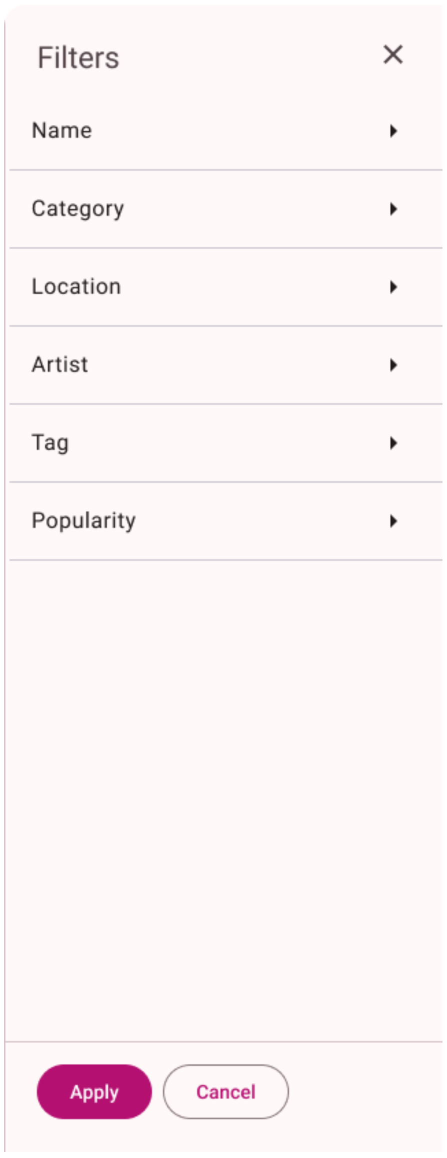
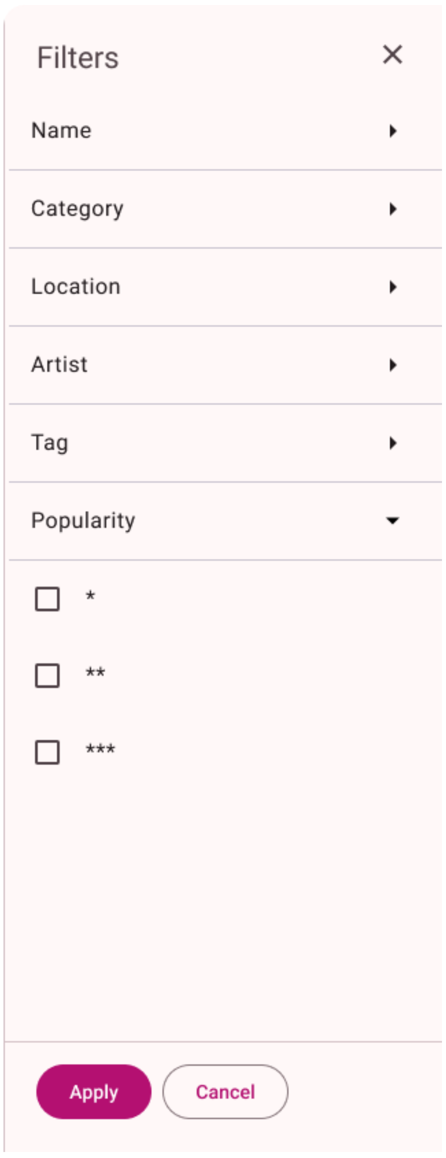


Go To screen with categories collapsed.

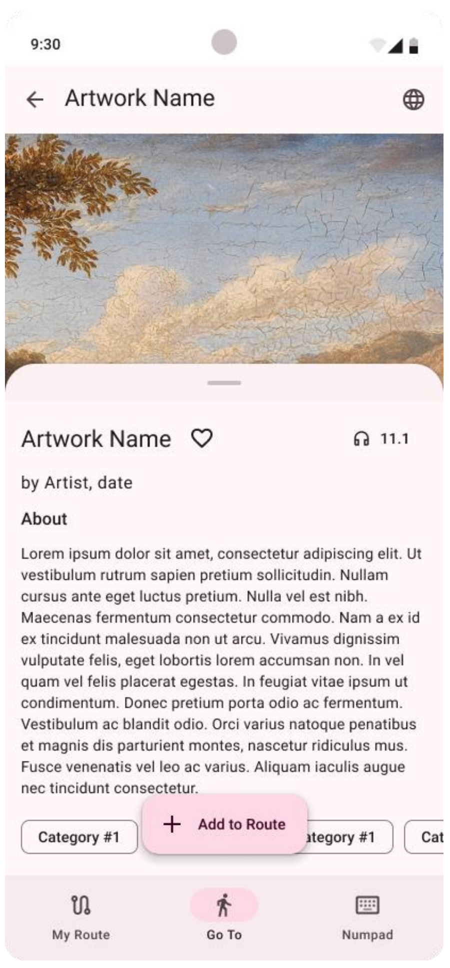
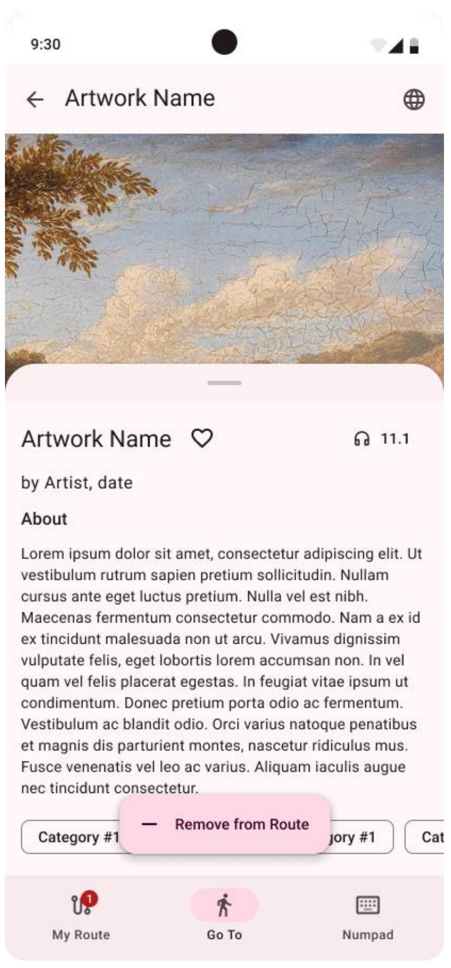


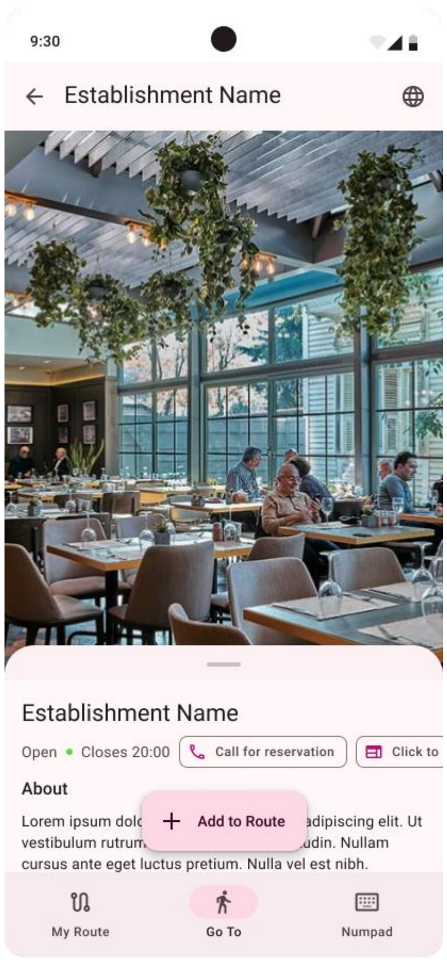
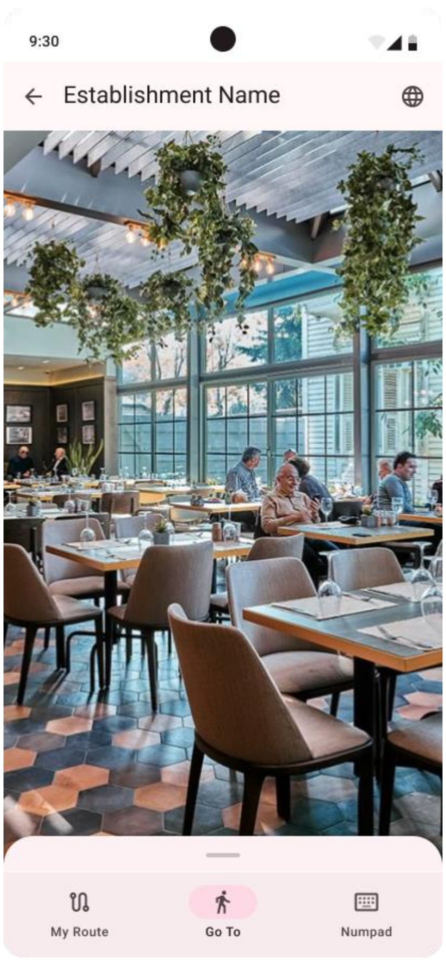
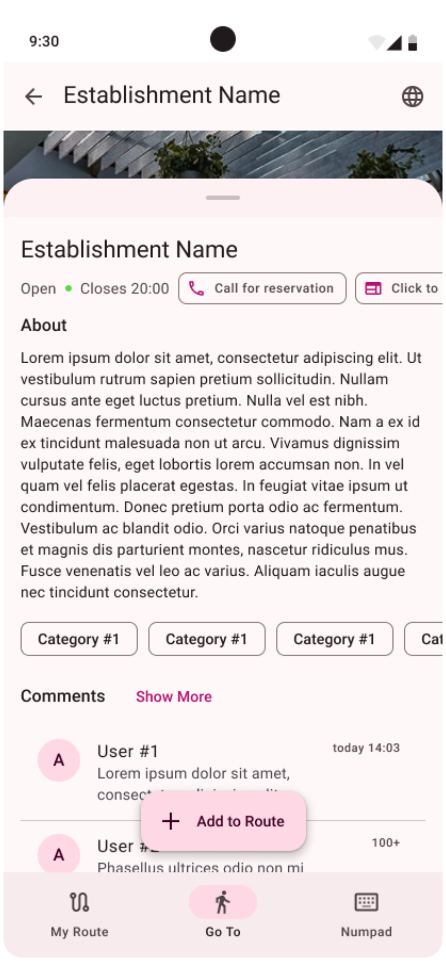
Go To screen with one category expanded.

The 'Go To' screen provides the user a way to browse, search and filter the catalog of the tenant.

 Filters Name Category Location Artist Tag Popularity 	 Filters Name Category Location Artist Tag Popularity ☐ * ☐ ** ☐ ***
Side sheet that shows the available filters.	Popularity filter expanded.

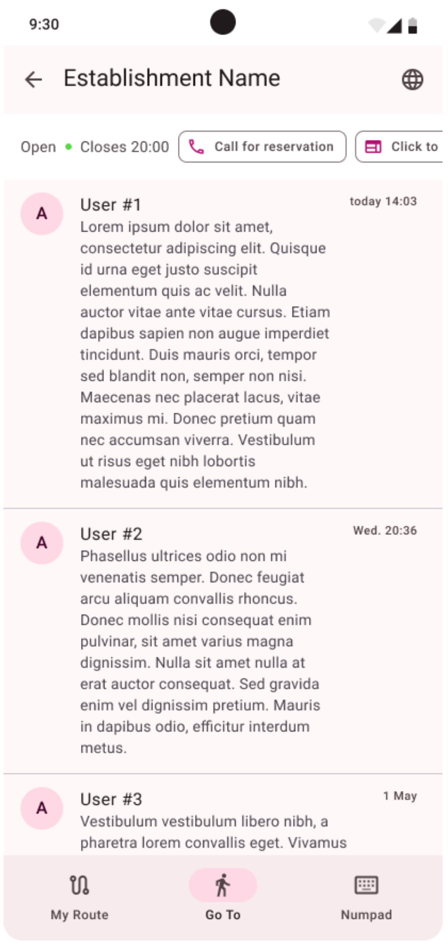
4.2.5 Entity detail screens

 9:30 Artwork Name Artwork Name 11.1 by Artist, date About Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vestibulum rutrum sapien pretium sollicitudin. Nullam cursus ante eget luctus pretium. Nulla vel est nibh. Maecenas fermentum consectetur commodo. Nam a ex id ex tincidunt malesuada non ut arcu. Vivamus dignissim vulputate felis, eget lobortis lorem accumsan non. In vel quam vel felis placerat egestas. In feugiat vitae ipsum ut condimentum. Donec pretium porta odio ac fermentum. Vestibulum ac blandit odio. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Fusce venenatis vel leo ac varius. Aliquam iaculis augue nec tincidunt consectetur. Category #1 + Add to Route ategory #1 Cat My Route Go To Numpad	 9:30 Artwork Name Artwork Name 11.1 by Artist, date About Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vestibulum rutrum sapien pretium sollicitudin. Nullam cursus ante eget luctus pretium. Nulla vel est nibh. Maecenas fermentum consectetur commodo. Nam a ex id ex tincidunt malesuada non ut arcu. Vivamus dignissim vulputate felis, eget lobortis lorem accumsan non. In vel quam vel felis placerat egestas. In feugiat vitae ipsum ut condimentum. Donec pretium porta odio ac fermentum. Vestibulum ac blandit odio. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Fusce venenatis vel leo ac varius. Aliquam iaculis augue nec tincidunt consectetur. Category #1 − Remove from Route jory #1 Cat 1 My Route Go To Numpad
Artwork screen showing the media, description and categories of an artwork.	The artwork added to the route of the user.

 9:30 Establishment Name Establishment Name Open • Closes 20:00 Call for reservation Click to About Lorem ipsum dolc adipiscing elit. Ut vestibulum rutrum udin. Nullam cursus ante eget luctus pretium. Nulla vel est nibh. Add to Route My Route Go To Numpad		 9:30 Establishment Name Establishment Name Open • Closes 20:00 Call for reservation Click to About Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vestibulum rutrum sapien pretium sollicitudin. Nullam cursus ante eget luctus pretium. Nulla vel est nibh. Maecenas fermentum consectetur commodo. Nam a ex id ex tincidunt malesuada non ut arcu. Vivamus dignissim vulputate felis, eget lobortis lorem accumsan non. In vel quam vel felis placerat egestas. In feugiat vitae ipsum ut condimentum. Donec pretium porta odio ac fermentum. Vestibulum ac blandit odio. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Fusce venenatis vel leo ac varius. Aliquam iaculis augue nec tincidunt consectetur. Category #1 Category #1 Category #1 Cat Comments Show More A User #1 today 14:03 Lorem ipsum dolor sit amet, consect Add to Route A User 100+ Phasellus ultrices odio non mi My Route Go To Numpad
A detail screen showing the half expanded bottom sheet that holds the information of an establishment.	Detail screen with a collapsed bottom sheet.	Detail screen with an expanded bottom sheet.

Detail screens for establishments can show communication information and opening hours of the establishment.

4.2.6 Comments


9:30

Establishment Name

Open • Closes 20:00 Call for reservation Click to

User #1 today 14:03
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque id urna eget justo suscipit elementum quis ac velit. Nulla auctor vitae ante vitae cursus. Etiam dapibus sapien non augue imperdiet tincidunt. Duis mauris orci, tempor sed blandit non, semper non nisi. Maecenas nec placerat lacus, vitae maximus mi. Donec pretium quam nec accumsan viverra. Vestibulum ut risus eget nibh lobortis malesuada quis elementum nibh.

User #2 Wed. 20:36
Phasellus ultrices odio non mi venenatis semper. Donec feugiat arcu aliquam convallis rhoncus. Donec mollis nisi consequat enim pulvinar, sit amet varius magna dignissim. Nulla sit amet nulla at erat auctor consequat. Sed gravida enim vel dignissim pretium. Mauris in dapibus odio, efficitur interdum metus.

User #3 1 May
Vestibulum vestibulum libero nibh, a pharetra lorem convallis eget. Vivamus

Comments screen of an establishment.

5. Appendix A

5.1. System actors

5.1.1. **Visitor:** a user of the mobile application. They are individuals who access and interact with the application to consume content and use the public features of the application.

5.1.2. **Curator:** a user of the administration application who is authorized to input data to the wiki module of the tenant they belong to. They can use the domain models to represent their tenant's floor plans, inventory, exhibits and facilities.

5.1.3. **Administrator:** a user of the administration application who is authorized to make changes to the tenant they belong to. The role of an administrator may involve tasks such as managing user accounts and configuring settings.

5.1.4. **Super administrator:** a user of the administration application who possesses elevated permissions, allowing them to perform tasks such as managing system-level configurations, modifying access controls, and controlling global settings that affect the entire tenant or multiple tenants.

5.2. Functional Requirements

5.2.1. Access management

5.2.1.1. The system enforces role-based access control to ensure that users can only perform actions allowed by their respective roles.

5.2.1.2. The administration application has the following roles (see System actors for more information):

5.2.1.2.1. Curator

5.2.1.2.2. Administrator (shortened as **admin** in this document)

5.2.1.2.3. Super administrator (shortened as **super admin** in this document)

5.2.1.3. Each user is granted only one role.

5.2.1.4. Each role is granted access to functionality and data allowed by their assigned role.

The functional capabilities and data accessibility of each role will be specified in this document through the use of subheadings categorizing each role.

5.2.2. User credential management

5.2.2.1. The system keeps an email address and an encrypted password string for each user.

5.2.2.2. The user uses their email and password to login.

5.2.2.3. The user uses the *forgot password* option to change their password without entering their current password.

5.2.2.4. The system emails a recovery link to the user for the user to reset their password.

5.2.2.5. When the recovery link is clicked the system redirects the user to a screen where they can reset their password.

5.2.2.6. The user can change their password by typing their old password.

5.2.2.7. For super admin and admin

5.2.2.7.1. The user can update the password of a user with a role of lesser privileges.

5.2.3. User management

5.2.3.1. The system provides a user search feature, allowing the user to search for other users within the system.

5.2.3.2. The system provides a filtering feature, allowing users to filter displayed users based on specific criteria.

5.2.3.3. The system provides a functionality where the user can create, update and delete a user.

5.2.3.4. The user can edit their own information.

5.2.3.5. The user has the capability to permanently delete their information from the system in a manner that is irrecoverable.

5.2.3.6. For super admin

5.2.3.6.1. The system displays all users of the system.

5.2.3.6.2. The user has the authority to create users with an admin role within a specific tenant.

5.2.3.6.3. The user has the authority to create users with a curator role within a specific tenant.

5.2.3.6.4. The user can remove users from tenants.

5.2.3.7. For admin

5.2.3.7.1. The system displays all users of the tenant the user belongs to.

5.2.3.7.2. The user has the authority to create users with a curator role within their own tenant.

5.2.3.7.3. The user can remove users from their own tenant.

5.2.4. Tenant management

5.2.4.1. For super admin

5.2.4.1.1. The user can create tenants.

5.2.4.1.2. Tenant entities contain details such as name, logo, user interface colors, contact person information, admin console URL and visitor application URL.

5.2.4.1.3. The user can edit the information of existing tenants.

5.2.4.1.4. The user can change user roles of users with lesser privilege.

5.2.4.2. For admin

5.2.4.2.1. The user can edit the information of their own tenant.

5.2.5. Area map

5.2.5.1. For curators

5.2.5.1.1. The system displays the map of the user's tenant.

File upload

5.2.5.1.2. The user can delete area maps.

Drawing interface

5.2.5.1.3. The system provides a drawing interface for the user to create floor plans.

5.2.5.1.4. The user can add architectural elements like walls and rooms to represent the floor plan of the area.

5.2.5.1.5. The user can add walkable areas to the area map.

5.2.5.1.6. The system provides drawing tools like lines, polygons and text annotations.

5.2.5.1.7. The user can add multiple floors to the area plan and designate names to the floors.

5.2.5.1.8. The user can view and edit different floors.

Map markers

5.2.5.1.9. The user can create, update and delete markers that show entities on the map.

5.2.5.1.10. The marker types are:

5.2.5.1.10.1. Item

5.2.5.1.10.2. Exhibit

5.2.5.1.10.3. Establishment

5.2.5.1.10.4. Facility

5.2.5.1.11. Each marker type has a distinct icon to visually represent the item type.

5.2.5.1.12. The user can create, update and delete areas that show items, exhibits, establishments and facilities on the map.

5.2.6. Item wiki

5.2.6.1. For curators

5.2.6.1.1. The user can create, update and delete items, exhibits, establishments and facilities.

5.2.6.1.2. The user can create, update and delete categories and tags.

5.2.6.1.3. The user can add and remove categories and tags from items, exhibits, establishments and facilities.

5.2.6.1.4. The user can enter detailed information about items, exhibits and establishments for each category.

5.2.6.1.5. The user can upload media related to the items and establishments.

5.2.7. Multi-language support

5.2.7.1. The system supports localization of user interface elements and provides a user-friendly experience for users in different language settings.

5.2.7.2. The system supports data entry in multiple languages for items, exhibits, establishments and facilities; and complementary entities like categories and tags.

5.2.8. Opening the application, login/sign up processes

5.2.8.1. The user scans a QR code to be redirected to the application.

5.2.8.2. The user can sign up using their

5.2.8.2.1. email address and a user defined password or

5.2.8.2.2. phone number

5.2.8.3. The system should not store any demographic information about the user like age, gender, marital status, ethnicity or race, religion, sexual orientation, disability status etc.

5.2.8.4. The user uses the forgot password option to send a recovery link to their mode of registration.

5.2.9. Location and crowd information

5.2.9.1. The system determines the user's location by

5.2.9.1.1. The QR code the user has scanned

5.2.9.1.2. The user's location services

5.2.9.2. The system shows the map/blueprint/plan of the tenant (ex. museum)

5.2.9.3. The system shows the user's location on the map view.

5.2.9.4. The system shows crowd density on the map view.

5.2.9.5. The system shows locations of items, exhibits, establishments (hotels, restaurants etc.), walkable areas, facilities etc.

5.2.9.6. The system displays crowd information of areas on a map view. The system conveys the density of the crowd by utilizing color, different shapes in different sizes.

5.2.10. User created tours

5.2.10.1. The user can view items or locations. (see Searching and filtering)

5.2.10.2. The user can add items or locations to a tour.

5.2.10.3. The user selects the items or any location they want to add to their route while browsing items and locations, in search and filter results or in item or location detail screens.

5.2.10.4. The system adds the item or location the user has selected to the active route.

5.2.10.5. If the user tries to add a step the active route already contains, the system notifies the user and does not add the step again.

5.2.11. System generated tours

5.2.11.1. The system will provide an option for auto-generating a tour.

5.2.11.2. The system notifies the user that in order to generate a tour the user has to answer a few questions and requires a consent form to be accepted

5.2.11.3. If the user has accepted the consent form the system displays a questionnaire. (see Questionnaire and interest collection)

5.2.11.4. According to the interests collected from the questionnaire the system generates a collection of items and exhibits that it deems the user might be interested in.

5.2.11.5. The system generates a route that consists of the collection of locations from the interest algorithm.

5.2.11.6. The generated route is aimed to be the shortest path(taking the crowd information into account) that visits all the items in the collection.

5.2.12. Questionnaire and interest collection

5.2.12.1. The system collects user interests through questionnaires.

5.2.12.2. Questionnaires can have multiple choice, single choice, yes/no and likert scale questions.

5.2.12.3. The system shows the progress of the questionnaire to the user by a progress indicator.

5.2.12.4. Demographic information is not collected by the system via questionnaires.

5.2.13. Routes and map navigation

5.2.13.1. Route view can be in two modes;

5.2.13.1.1. No route is selected: in which the user can use their location, the map view to navigate the area, view crowd information, view locations of exhibitions, objects, establishments, facilities etc.

5.2.13.1.2. A route is active: in which the user can do the same activities they could do in ‘no routes selected’ mode. Additionally, the system will show a visual representation of the route the user has started in map view, with an accompanying list of the steps of the route in a list view.

5.2.13.2. The system will provide a way to mark items on the route as ‘seen’.

5.2.13.3. The system will keep track of the user’s location by the user’s device’s location service if available.

5.2.13.4. If device location is not available, the system will determine the user's location by comparing the items the user has selected as 'seen' and the user's route.

5.2.13.5. The system shows steps of the active route in visiting order (meaning first stop first).

5.2.13.6. The user can click on a step of the route in the list view to view a detailed page about the item or location.

5.2.13.7. The system displays information about the item or location that fits the user's interests by computing which categories the user might be interested in and selects among item descriptions in these categories.

5.2.14. Games

5.2.14.1. The system will let the users experience their visit by playing a quiz game.

5.2.14.2. The system will ask the user whether he/she wants to play a quiz game.

5.2.14.3. The system will ask for the user's categories of interest and knowledge level.

5.2.14.4. The system will provide a user interface to start the game.

5.2.14.5. The system will display a question with multiple answers and lead the user to where he/she can find the correct answer.

5.2.14.6. The system will display a hint in case the user can't give the correct answer.

5.2.14.7. The system will award the user with some points for the correct answer.

5.2.14.8. The system will provide a UI to end the game whenever the user wants.

5.2.14.9. The system will display an end-game screen containing the results and resulting points, badges etc.

5.2.14.10. The system will provide a UI for the users with the "Curator" role to enter questions for items.

5.2.14.11. The system will provide an AI-based assistant to the Curators while entering the questions.

5.2.15. Searching, filtering and searching via keypad

5.2.15.1. The system provides a search functionality for the user to search

- 5.2.15.1.1. Items by name, description, type, category and tag;
- 5.2.15.1.2. Exhibits by name, description, category and tag;
- 5.2.15.1.3. establishments and facilities by name and type.

5.2.15.2. The user types their query in the search field. The system finds the entities that fit the user's query and displays the results in a list view.

5.2.15.3. The system provides a filter functionality to filter

5.2.15.3.1. Items and exhibits by name, category, location, artists, tag, popularity;

5.2.15.3.2. Establishments and facilities by type.

5.2.15.4. The system provides easy access to some predetermined filters. The user can click on them to apply filters.

5.2.15.5. The user can apply multiple filters at the same time.

5.2.15.6. The system shows the result of the filtering process in a list view.

5.2.15.7. Each search and filtering result is shown with its name, media and a short description.

5.2.15.8. The user selects a result by clicking on it. The system redirects the user to the result's detail page.

5.2.15.9. If there are no entities that fit the user's search query and filters the system shows a message to inform the user.

5.2.15.10. The system provides a keypad so that the user can enter unique item numbers. The user enters a number via keypad. If there is an entity related to the number the system shows the entity. If not, the system displays an error message.

5.2.16. Viewing exhibits, items, establishments and facilities

5.2.16.1. The system displays entities in the map view as markers.

5.2.16.2. The system displays a compact view for all entities (items, exhibits, establishments and facilities) in list views. The information being displayed can range from just the type of the facility to including name, image and a short description.

5.2.16.3. The user can click on compact views of entities to display more information. The system redirects the user to the corresponding entity page.

5.2.16.4. Item pages consist of item name, various media related to the item, description, the artist(s) if applicable, categories and tags.

5.2.16.5. Exhibit pages consist of exhibit name, media related to the exhibit, description, list of items included in the exhibit, the artist(s) if applicable, categories and tags.

5.2.16.6. Establishment pages consist of establishment name, media related to the establishment, description, type of the establishment, contact number if applicable, web page if applicable, other links, categories and tags.

5.2.16.7. Some facilities, like classrooms, can have detail pages. The pages consist of the name of the facility if applicable, type of the facility, media if applicable, crowd information if applicable.

5.2.17. Social networking

5.2.17.1. The users can enter comments about items, exhibits, and establishments.

5.2.17.2. The users can read comments left by other users.

5.2.17.3. The system displays comments in descending order by date.

5.2.17.4. The users can add items, exhibits and establishments to their favorites list.

5.2.18. Accessibility

5.2.18.1. The system provides an option to change the applications language.

5.2.18.2. The system employs color contrast, color palettes and typography to accommodate color blind users.

5.2.18.3. The system employs font options to accommodate users with dyslexia.

5.2.19. Multi-language support

5.2.19.1. The system supports localization of user interface elements and provides a user-friendly experience for users in different language settings.

5.2.19.2. The system provides language fallback mechanisms, displaying content in a default language if translations are missing or incomplete.

5.2.19.3. Users can search and filter content based on different language options, enabling browsing of information in the preferred language.