Deliverable 2.2 – System Architecture Design and Specification Report

Deliverable type	R- Document, Report
Dissemination level	PU- Public
Due date (month)	M7
Delivery submission date	31.08.2023
Work package number	2
Lead beneficiary	IOTIQ GMBH

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

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	D2.2			
Deliverable name	System Architecture Design and Specification Report			
Work package number	WP2			
Work package name	System Development and Evaluation			
Date of delivery		M7		M7
Version	Version 1.0			
Lead beneficiary	IOTIQ GMBH			
Responsible author(s)	Metin Tekkalmaz, IOTIQ, metin@iotiq.de			
Reviewer(s)	Angel Lagares, NIMBEO, alagares@nimbeo.com Santiago Rondon, NIMBEO, srondon@nimbeo.com Neşe Şahin Özçelik, Bilkent Universitesi Vakıf, nozcelik@bilkent.edu.tr Arzu Sibel İkinci, Bilkent Universitesi Vakıf, aikinci@bilkent.edu.tr Eda Gürel, Bilkent Universitesi Vakıf, eda@tourism.bilkent.edu.tr			

Short Description	This deliverable is intended to be the reference document for developing the system architecture of CULTURATI. This document aims to provide information on the modular reference architecture specification, the modules' interfaces, and the data flow between them.

History of Changes			
Date	Version	Author	Remarks
01.07.2023	Draft 0.1	Metin Tekkalmaz	First Version

31.07.2023	Final 1.0	Metin Tekkalmaz	Revised After Review

Executive Summary

The System Architecture Design and Specification document outlines the design and implementation of an innovative application that seamlessly combines Java Spring Boot, React.JS, and Artificial Intelligence (AI) algorithms to create a personalized visitor experience. The proposed system aims to optimize visitor flow in indoor and outdoor spaces while providing guided tours and detailed information through a connected Wiki.JS library. The system also has a platform for content creators to create customized content.

The application will be developed with a micro-services architecture to ensure modularity and scalability. The back end will utilize Java Spring Boot, offering a robust and flexible foundation for handling various functionalities, including user management, AI algorithms, and database interactions. The micro-services approach allows for independent development and deployment of specific components, enhancing maintainability and reducing dependencies.

The front-end will be developed using React.JS, providing a modern and responsive user interface for visitors to access personalized routes and guided tours. React's component-based architecture promotes re-usability and enhances the overall user experience, accommodating visitors from different devices and platforms.

The core feature of the system involves the integration of AI algorithms to analyze real-time data and identify visitor traffic on each site. By leveraging AI, the application can optimize visitor flows, minimize congestion, and provide an enhanced visiting experience. The AI algorithms will also enable personalized routes for visitors, offering tailored tours based on their interests and preferences.

To support the site’s Wiki module, the system will seamlessly connect to a Wiki.JS library. This Wiki module will contain comprehensive information about registered sites, allowing visitors to access general details, historical insights, and exhibit information. Visitors will also have the ability to post comments, share experiences, and interact with the community. The Wiki module will also offer an administration role to manage and update information about the site.

The system architecture emphasizes security and data privacy. Robust authentication and authorization mechanisms will be implemented to safeguard sensitive information, ensuring that

only authorized users have access to specific features and data. Data encryption will be applied to protect user information and maintain compliance with data protection regulations.

The system will utilize RESTful APIs to facilitate seamless communication between different modules. This approach allows for the smooth integration of data and functionalities across the entire system, enabling real-time updates and ensuring data consistency.

Table of Contents

Index of Figures

1. Introduction

CULTURATI aims to foster and enhance visitor learning and experiences at indoor and outdoor attractions, museums, national parks, gardens, and various sites of cultural and artistic significance.

At its core, CULTURATI represents a trans-formative method of delivering information to visitors, enriching their understanding and appreciation of cultural heritage and artistic wonders. By seamlessly integrating education and navigation subsystems, this innovative system offers customized Q&A games and personalized routes, ensuring that each visitor embarks on a unique journey tailored to their interests and preferences.

CULTURATI serves a diverse user base, catering to both administrators of attractions, creative professionals, and citizens with cultural expertise (content creators), as well as visitors seeking knowledge, information, and direction (end-users and customers of cultural heritage and arts). By bridging the gap between content creators and enthusiasts, the platform fosters an inclusive and collaborative environment, encouraging the exchange of insights and enriching the collective cultural experience.

The Education Subsystem of CULTURATI engages visitors through captivating Q&A games, delivering information in an interactive and enjoyable manner. By customizing the content to suit the interests of each individual, the system promotes a deeper understanding of artworks, historical events, and cultural nuances. The Education Subsystem not only enhances visitors' knowledge but also nurtures their curiosity, inspiring a lifelong love for cultural exploration.

Complementing the Education Subsystem is the Navigation Subsystem, which offers personalized routes through attractions and sites. Powered by a sophisticated algorithm, CULTURATI dynamically curates routes that align with visitors' preferences, allowing them to experience the cultural heritage and arts at their own pace and according to their unique interests. This navigation feature optimizes visitor flow, ensuring a seamless and fulfilling experience for all.

The central premise of CULTURATI revolves around delivering the right information to the right person at the right time. As visitors engage with the platform, they unlock a wealth of insights into the works of art, history, and cultural significance presented in a manner that resonates with their individual inclinations. This tailored approach empowers users to forge meaningful connections with the cultural and artistic aspects that matter most to them.

2. CULTURATI General Architecture

The first architecture shows a core vision of the project, covering the aspects of the Artificial Intelligence sub-system to provide the right information to the right person at the right time, these algorithms will be in charge of the selection of that person and that time based on the data from the Internet of Things module. Subsequently, it includes using sensors that will obtain the necessary data so that the algorithms that apply Artificial Intelligence can provide accurate information to each visitor.

The second architecture shows the architecture of the part of the project that will be carried out by NIMBEO and has all the details regarding the Wiki, this is responsible for storing all the information of the places and visitors can interact with it.

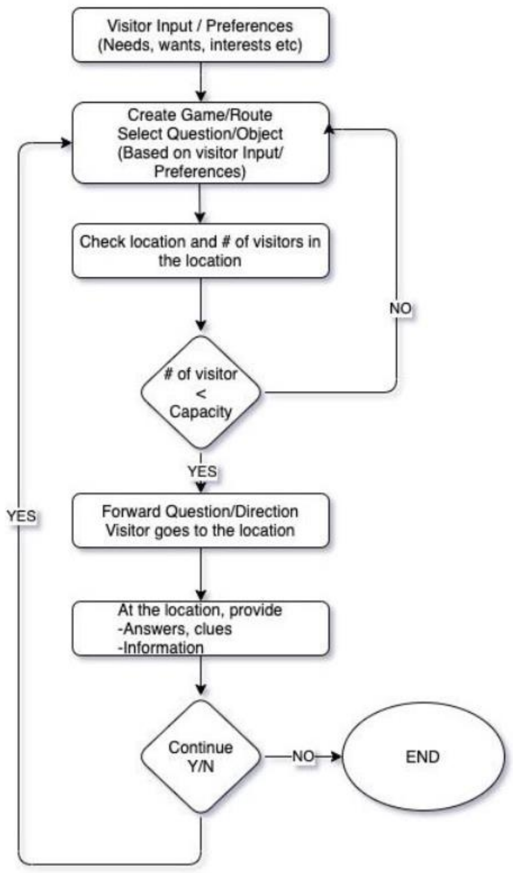
Visitor Input / Preferences (Needs, wants, interests etc)
Create Game/Route Select Question/Object (Based on visitor Input/ Preferences)
Check location and # of visitors in the location
# of visitor < Capacity
NO
YES
Forward Question/Direction Visitor goes to the location
At the location, provide -Answers, clues -Information
Continue Y/N
NO
END
YES

Figure 1: System Flow Chart

2.1. General Architecture of the Project

The architecture of the project is designed to deliver a cutting-edge and interconnected ecosystem that harmoniously integrates two distinct but synergistic platforms: The core feature of the system involves the integration of AI algorithms to analyze real-time data and identify visitor traffic on each site. By leveraging AI, the application can optimize visitor flows, minimize congestion, and provide an enhanced visiting experience. The AI algorithms will also enable personalized routes for visitors, offering tailored tours based on their interests and preferences. and the Wiki module will contain comprehensive information about registered sites, allowing visitors to access general details, historical insights, and information. The project's overarching objective is to provide users an immersive and enriching experience engaging with cultural heritage and arts while fostering collaboration and knowledge-sharing.

Real time data and location: CULTURATI is an innovative platform that re-imagines cultural heritage and arts experiences using real-time data and location-driven technologies. The platform dynamically curates personalized games and routes for each user, optimizing their interactions at venues and site-based cultural institutions. By detecting user movements through sensors and GPS, CULTURATI delivers relevant information to visitors.

This dynamic approach ensures a seamless and immersive cultural exploration, fostering a deeper connection between users and the rich cultural heritage they encounter. Additionally, CULTURATI efficiently manages visitor flow, enhancing crowd management and creating an enjoyable experience for all.

Artificial Intelligence Integration: The AI Subsystem and IoT capabilities (coming from the IoT Subsystem) will be utilized to optimizebalance the number of people in various locations according to carrying capacities and user interests 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.

Gamification Module: 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.

Content Management Sub-system (wiki): The content management subsystem acts as a central repository for all relevant information about the sites, its exhibits, and its layout. It stores data such as exhibit locations, descriptions, multimedia content, accessibility information, and any other relevant details.

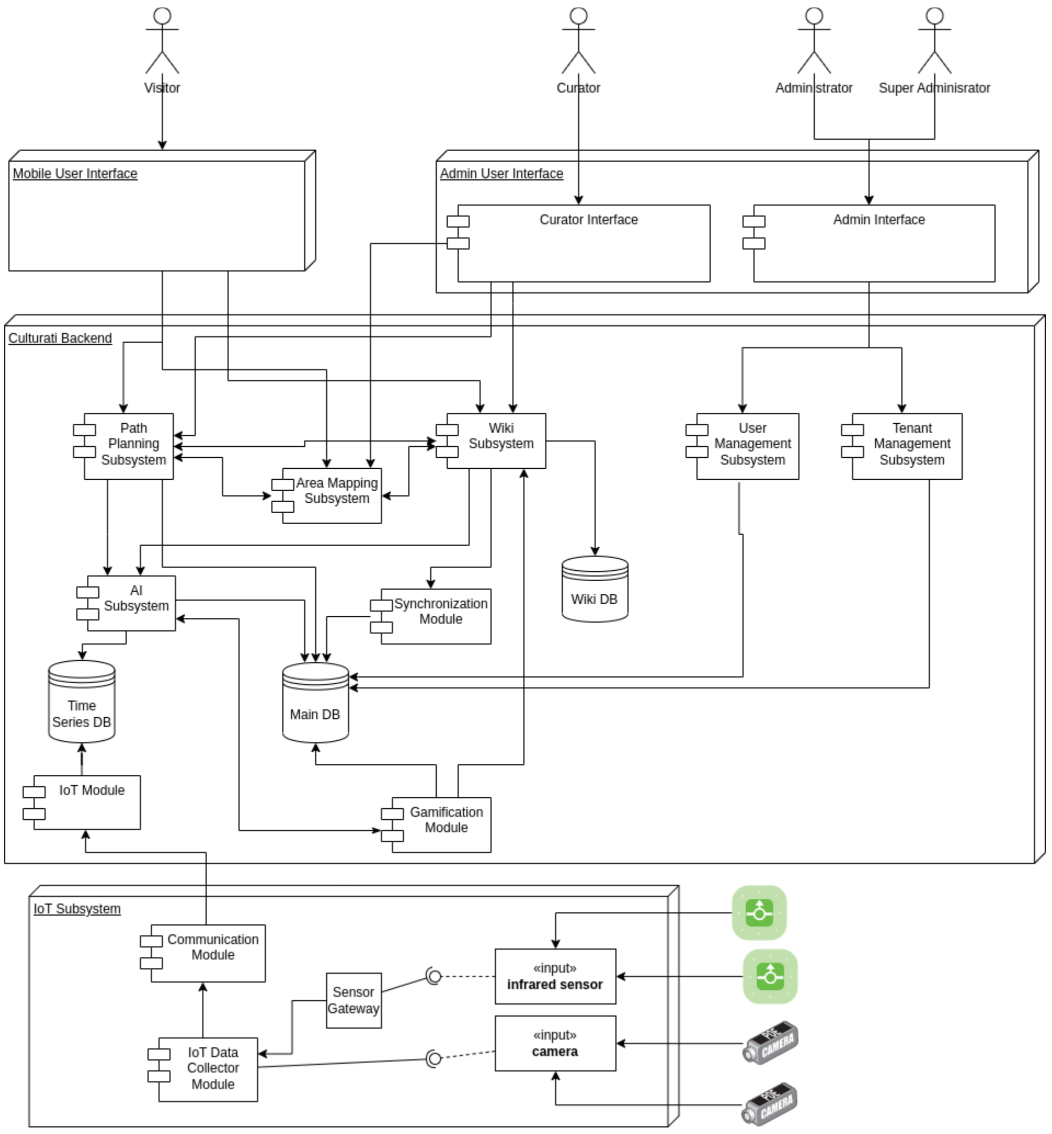

Figure 2: General System Architecture

3. CULTURATI Core Application

The CULTURATI Core Application consists of the Navigational Subsystem, the Area Mapping Subsystem, the AI Subsystem, the IoT Subsystem and Educational Subsystem. It consolidates these subsystems and orchestrates their functionalities while integrating with the Content Management Subsystem (Wiki) to provide the users of CULTURATI with a seamless experience.

This section of the document details the composition and communication of the subsystems mentioned above.

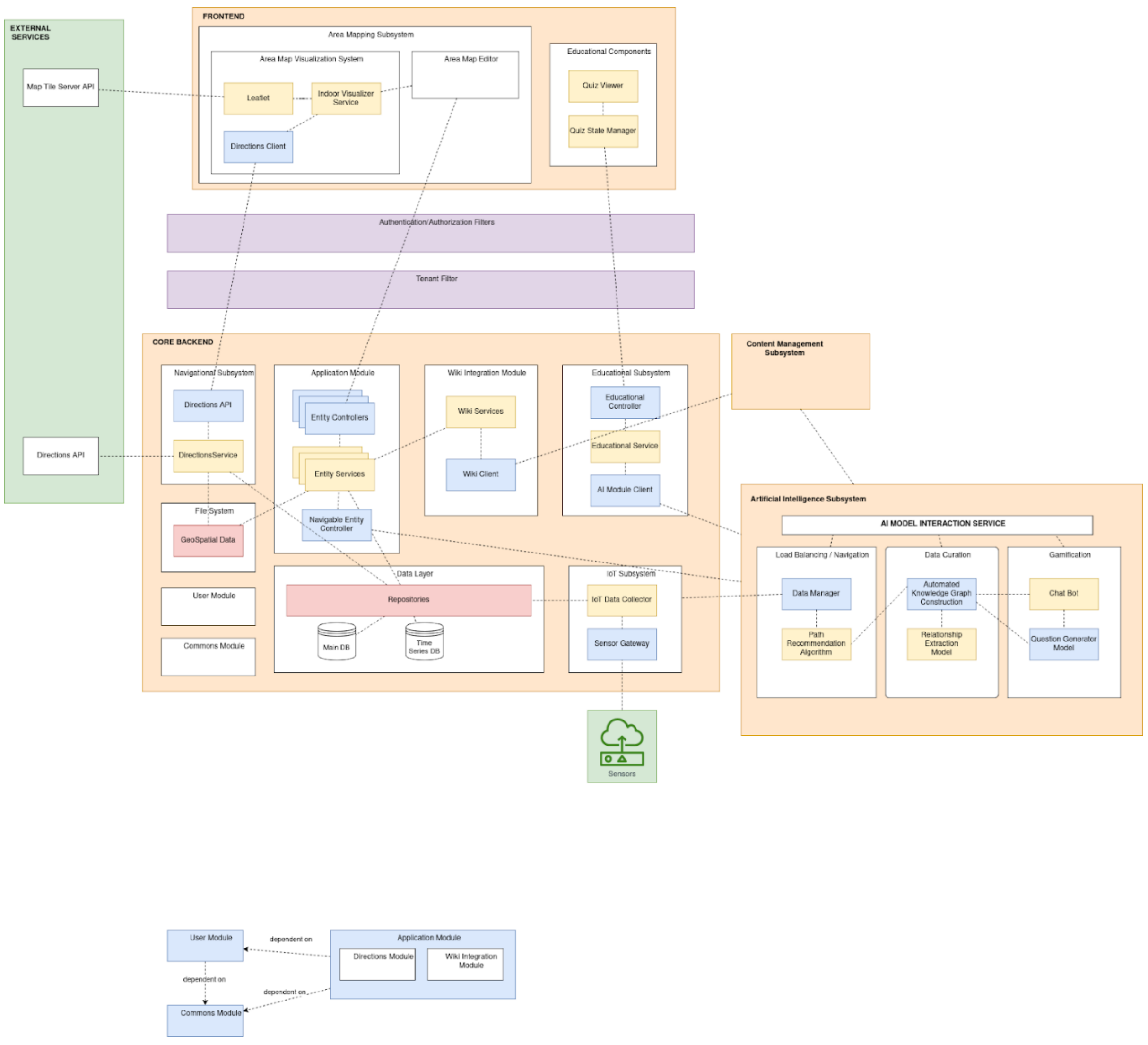

Figure 3: Modular Reference Architecture showing the overall core backend structure and relationships.

3.1. Navigational Subsystem (Path Planning Subsystem)

The path planning subsystem plays a crucial role in the CULTURATI application, ensuring that users can efficiently navigate the designated area while enjoying a personalized and enriching experience. To accomplish this, the path planning subsystem spans several modules, each contributing specific functionalities to enhance the overall user experience.

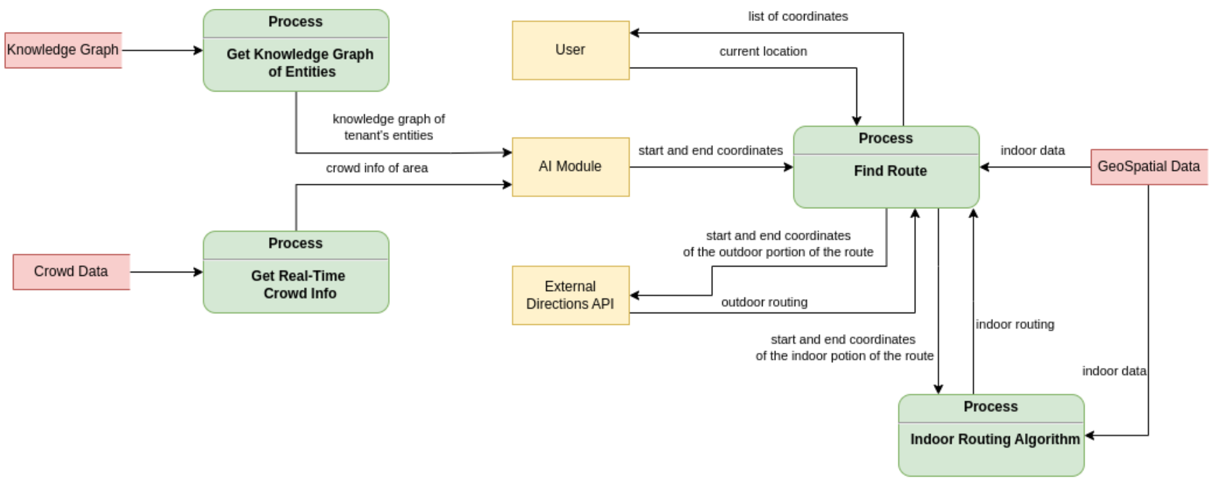

Figure 4: Data flow diagram of routing with the navigational subsystem.

Content Management Subsystem:

The content management subsystem is responsible for curating and managing a diverse range of cultural information, historical facts, points of interest, and relevant content about the area. It gathers data from various reliable sources, including museums, cultural institutions, historical records, and local experts. This data is then organized, updated, and made available to the path planning subsystem to consider while generating paths for users.

AI Subsystem: The AI subsystem is a critical component of the CULTURATI application, leveraging artificial intelligence and machine learning algorithms to analyze user behavior, preferences, and historical data. By employing AI, the path planning subsystem can adapt and improve its recommendations over time based on user feedback and interaction patterns.

The path recommendation system employed by the AI Subsystem leverages a combination of visitor interests and real-time foot traffic data. By seamlessly integrating these two critical elements, the system intelligently guides visitors through the site, suggesting optimal pathways based on their preferences while also accounting for current congestion levels. This dynamic synergy ensures a personalized and efficient exploration, enhancing visitor satisfaction and enabling site managers to maintain a harmonious and secure environment.

IoT Subsystem:

The IoT (Internet of Things) subsystem is responsible for collecting real-time sensor data from the designated area. These IoT sensors may include environmental sensors, crowd density detectors, temperature and weather sensors, and other relevant devices. By integrating sensor data with the path planning subsystem, the application can optimize paths based on the current conditions of the area. For example, during crowded times the path planning system can dynamically adjust routes to avoid congested areas, ensuring a smooth and pleasant experience for the users while managing the crowds in the area.

CULTURATI Directions API:

The CULTURATI Directions API consolidates the information coming from the AI Subsystem and real time geospatial data of the area to find routes the users can traverse. The Directions API uses both the geospatial data from external sources for outdoor navigation and indoor maps of the sites for indoor navigation. The indoor area maps will be tagged with Simple Indoor Tagging1 guidelines that enable tagging building levels; traversable areas like rooms and corridors; obstructions like walls; connections like doors and windows. Using these indoor tags, the CULTURATI Directions API provides wayfinding indoors. Outdoor wayfinding will be done by external Direction APIs and will be merged with indoor wayfinding results to provide a seamless experience.

External Directions API:

Leveraging the existing solutions for outdoor routing, the Navigational Subsystem integrates an external directions API to complement its own directions API. This integration capitalizes on the well-established expertise in outdoor routing of Open Route Service2, allowing the CULTURATI Directions Module to provide robust and accurate navigation services to users.

3.2. Area Mapping Subsystem

Area mapping is one of the core components of the CULTURATI application benefiting site administrators, curators and visitors. It enables administrators and curators to visually represent the

[1] https://wiki.openstreetmap.org/wiki/Simple_Indoor_Tagging

[2] https://openrouteservice.org/

areas and place objects and artworks. Meanwhile, mobile user interface users are facilitated with an intuitive means of navigating these areas through visual cues and route guidance.

The Area Mapping Subsystem functions as a pivotal component, catering to both user interactions and facilitating the AI subsystem. Its role encompasses user input management for area mapping and visualization, while also serving the AI module by generating navigable entity graphs. This subsystem acts as a bridge, harmonizing the needs of both users and the AI system.

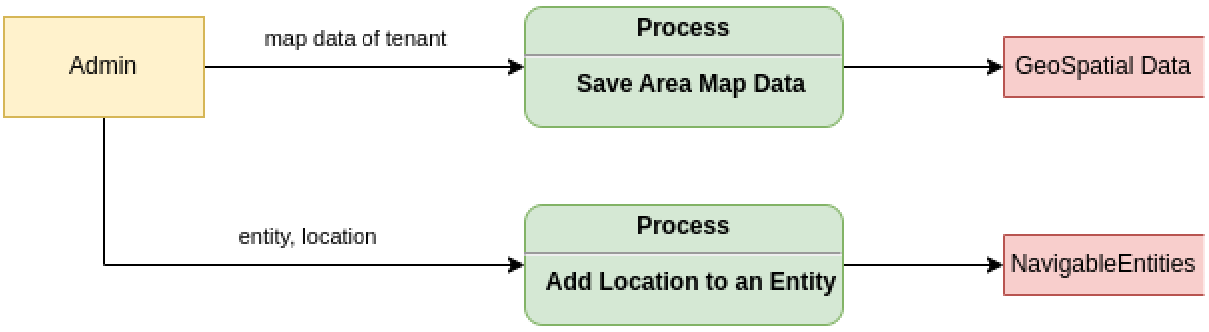

Figure 5: Data flow diagram of a user with the admin role, uploading the map area data of their tenant and adding the location of an entity

The Area Mapping Subsystem takes the user's map input and translates it into visual representations that users can interact with. This visualization aids users in understanding the layout and geography of the mapped area.

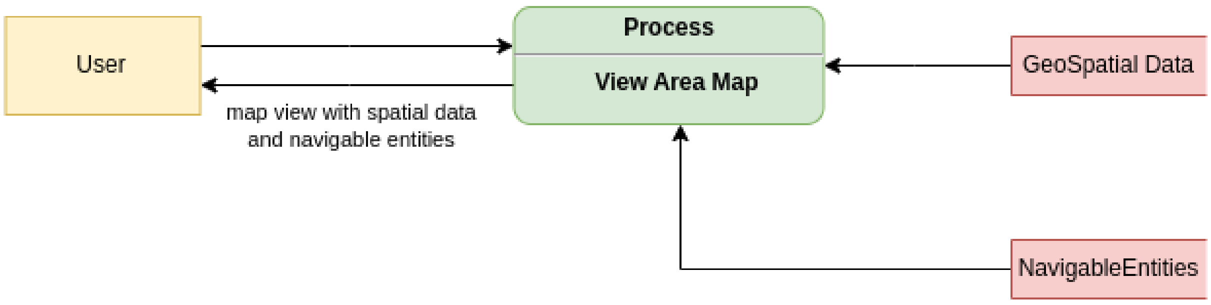

Figure 6: Data flow diagram of a user viewing the area map of their desired tenant

The Area Mapping Subsystem extends its functionality to support the AI subsystem, contributing crucial data and insights that enhance the AI's capabilities. To assist the AI module, the subsystem generates navigable entity graphs. These graphs represent the connections and relationships between different entities within the mapped area. This information is invaluable for the AI's navigation and decision-making processes.

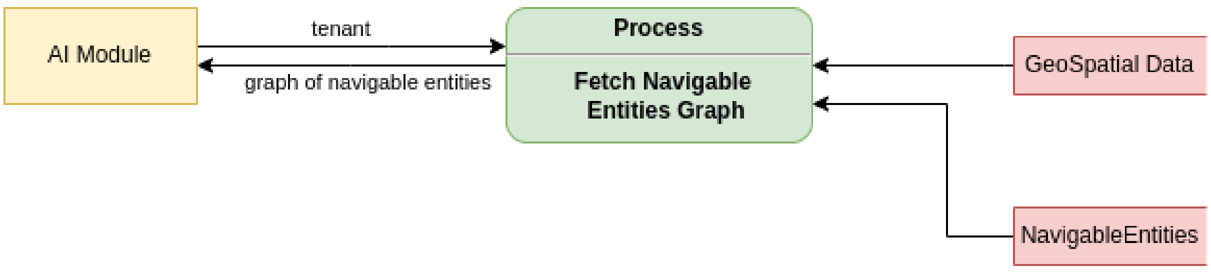

Figure 7: Data flow diagram of the AI module requesting the graph of the navigable entities belonging to a tenant

In order to comply with geospatial standards, the area maps will be drawn with external software like JOSM or OSMInEdit. However item, exhibit, facility or establishment placements on map areas will be managed in the CULTURATI admin application using the Area Mapping Subsystem. 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.

The CULTURATI application extends its navigation capabilities to indoor environments by implementing Indoor Mapping and Routing features. This functionality is made possible by integrating the standardized Simple Indoor Tagging schema, which has been developed to cater to indoor mapping.

The Simple Indoor Tagging schema serves as a guideline for annotating indoor spaces with relevant metadata, including details about rooms, corridors, points of interest, and other spatial entities within enclosed structures such as museums or cultural venues. This schema ensures a consistent approach to indoor mapping across diverse settings. Since indoor wayfinding is not as commonplace as outdoor wayfinding, the CULTURATI application will be employing its own wayfinding algorithm using these tags.

3.3. Artificial intelligence Subsystem

An Artificial Intelligence Subsystem is a module within a software application that manages the AI capabilities according to the specific task that will use 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.

Expanding upon this foundation, the AI Subsystem will take on an even more dynamic role within the software application. Its multifaceted responsibilities will extend beyond conventional AI capabilities to encompass experiential engagement and strategic guidance.

At its core, the AI Subsystem will serve as the orchestrator of gamification, infusing the visitor's journey with an interactive and educational dimension. Leveraging its generative prowess, the subsystem will collaborate with visitors to generate pertinent questions that align with their interests and curiosities. This personalized query generation will not only stimulate the visitor's intellectual engagement but also foster a deeper connection with the presented content. As the visitor embarks on their quest for answers, the AI Subsystem will stand as their guide, assisting in the navigation of the knowledge base and the site's diverse offerings.

The AI Subsystem plays a pivotal role in curating and enhancing accessibility to cultural heritage knowledge. By implementing knowledge graphs, it organizes and presents information in a structured manner, connecting various elements such as narratives, exhibits, and artworks. This approach not only preserves cultural heritage but also makes it more accessible to visitors. By visually mapping out relationships between different components, the subsystem simplifies complex information, offering a user-friendly and engaging experience that fosters a deeper understanding and appreciation of our cultural legacy.

Furthermore, the AI Subsystem's role will transcend the confines of individual curiosity, extending its reach to enhance the overall management of visitor traffic. Drawing upon its capacity for classification and forecasting, the subsystem will analyze visitor interests, preferences, and historical patterns. Armed with this insight, it will collaborate harmoniously with both visitors and curators, proposing optimal pathways tailored to individual preferences and occupancy information. This dynamic traffic management will not only optimize the visitor experience but also alleviate congestion and ensure a harmonious flow throughout the site.

While considering visitor interests is very important, foot traffic will also be taken into account for path recommendations. Harnessing the power of real-time data, the AI Subsystem seamlessly integrates sensor-generated information on visitor traffic. By analyzing metrics such as foot traffic within designated areas over 1-2 hour intervals, the subsystem takes on a pivotal role in estimating the number of people at any time of the day and balancing the crowd distribution to these areas.

This dynamic orchestration ensures an optimal visitor experience by preventing overcrowding and minimizing exposure to high visitor traffic. In doing so, the subsystem creates an environment that prioritizes visitor satisfaction while simultaneously enabling site managers to uphold a seamless, secure, and organized experience. This amalgamation of data-driven insights and strategic management underscores the AI Subsystem's contribution to fostering a harmonious and enjoyable exploration of the site's rich cultural offerings.

In summary, the AI Subsystem's evolution from a mere controller of AI capabilities to a catalyst of immersive engagement and strategic guidance underscores its pivotal role in shaping the visitor experience. Through gamification, personalized guidance, and intelligent traffic management, the subsystem will emerge as a vital companion, empowering visitors to embark on enlightening journeys and curators to orchestrate seamless and enriching explorations of the cultural tapestry held within the site.

The AI Subsystem employs advanced models, including Rebel and IBM Knowgl, to extract intricate relationships from unstructured data sources. Through a sophisticated integration framework, the subsystem orchestrates a seamless dialogue between user requests and model-generated outputs. This orchestration is facilitated by a Flask service, complemented by the asynchronous task management capabilities of Celery. This synergy ensures efficient and real-time interactions, allowing users to effortlessly navigate the knowledge base and receive timely, relevant responses.

As the knowledge base matures and solidifies, the AI Subsystem leverages this stability to enhance its functionalities incrementally. In particular, the subsystem's capacity for question generation and gamification evolves in tandem with the growing robustness of the knowledge base. This synergy is pivotal in harnessing the potential of Language Models (LLMs) responsible for gamification. By exposing these LLMs to factual knowledge derived from the evolving knowledge base, their performance can be elevated, enabling the generation of more insightful and engaging queries. This virtuous cycle reinforces the system's ability to captivate users through gamified interactions, fostering a deeper integration of educational and entertaining elements into the visitor experience.

3.4. IoT Subsystem

The CULTURATI system extensively uses the Internet of Things (IoT) to optimize the management of crowds in venue-based and site-based cultural and creative industries (CCIs). To achieve this, the system employs various tracking technologies, such as bidirectional sensors (people counter sensors), radio frequency, RFID tags, Wi-Fi, Bluetooth, indoor positioning systems (IPS), local positioning systems (LPS), cell-tracking, camera, and GPS. These technologies allow the system to monitor the movement of people within the venues accurately.

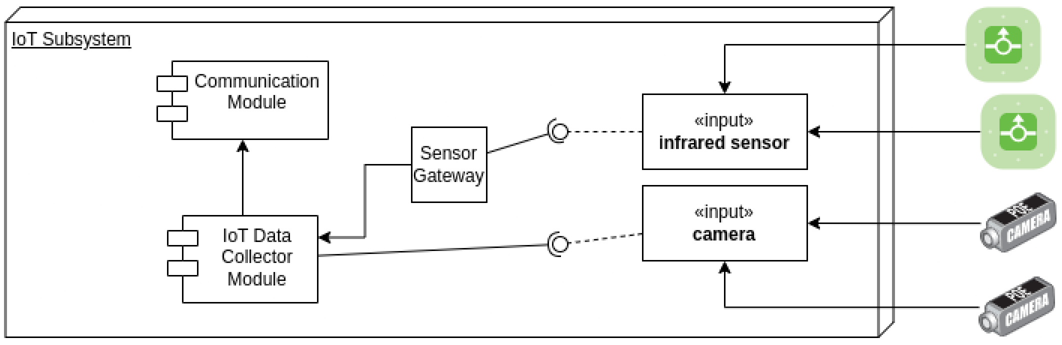

Figure 8: The general architecture of the IoT subsystem.

The IoT Subsystem plays a pivotal role in processing and preserving raw sensor data, which subsequently becomes accessible to the AI module for route-finding purposes. The IoT Subsystem serves as the initial processing hub for raw sensor data, ensuring its transformation into meaningful and usable information.

Raw sensor data is collected from various IoT devices deployed across the environment. The IoT Subsystem processes the collected data, performing necessary calculations, filtering, and data conditioning to extract valuable insights. This processed data may include real-time environmental conditions, object movement patterns, and other relevant information.

The processed data with the raw data are then saved within the IoT Subsystem's storage infrastructure. This organized repository retains historical and current sensor data, forming a valuable resource for the AI subsystem.

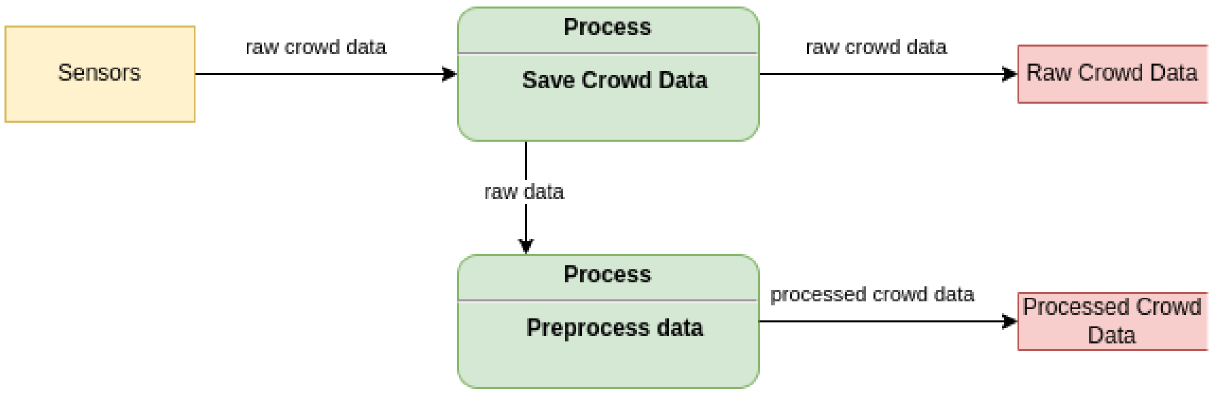

Figure 9: Data flow diagram of collection of crowd data.

3.5. Educational Subsystem (Gamification Module)

The Gamification Module is a crucial component designed to enhance the interactive and educational experience of visitors in the designated area. Its primary function is to generate quiz questions related to the items and exhibits, providing an engaging and enjoyable way for visitors to interact meaningfully with the content. By collaborating with the AI Subsystem, responsible for question generation, the module ensures that the quiz questions are relevant, immersive, and tailored specifically to the items and exhibits within the space. This personalized approach aims to deepen visitors' understanding and appreciation of the cultural and educational content present in the area.

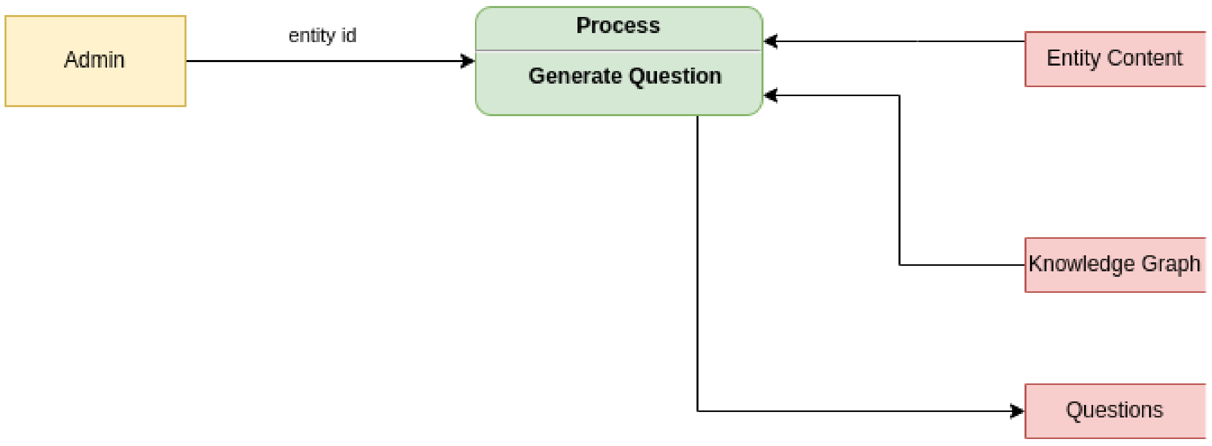

Figure 10: Data flow diagram of question generation

The Gamification Module's core functionality includes dynamic quiz question generation. It interacts with the AI Subsystem, which utilizes advanced machine learning algorithms to dynamically create quiz questions based on the available items and exhibits in the area. The AI Subsystem ensures that the questions are diverse, contextually relevant, and aligned with the educational goals of the content. Once generated, the quiz questions undergo a review process by curators to ensure accuracy, quality, and relevance. This ensures that the questions provide a meaningful and educational experience for the visitors.

The AI Subsystem employs transformer models to facilitate question generation. Initially, these transformers create a series of question-answer pairs by drawing context from an input text. Subsequently, leveraging this array of pairs alongside the original text, the subsystem employs its cognitive prowess to construct deceptive answers designed to challenge and engage the user. These crafted distractors not only enrich the user's interaction by presenting a rigorous test but also stimulate critical thinking and analytical skills, thus amplifying the overall learning experience in an immersive and thought-provoking manner.

Our approach involves utilizing pre-trained models enhanced with Knowledge Graph designed for question-answer pairs to generate both questions and corresponding answers. By leveraging the capabilities of these specialized models, we ensure the creation of coherent and contextually relevant questions, along with accurate answers that align with the given queries. Our strategy employs a multi-task model that generates both questions and answers as a unified output. This innovative approach ensures a cohesive and interconnected dialogue, where the generated questions seamlessly correlate with their respective answers. By employing this integrated framework, we create a more dynamic and engaging user experience, fostering a deeper understanding and interaction with the content presented on the site. In our ‘’wrong answer’’ generation process, we utilize a comprehensive input that includes the question, answer, and relevant text. Through this holistic approach, our system produces insightful deceptive answers obtained from the models and KG as the output. By drawing from the entirety of provided information, we ensure that the generated distractors are contextually relevant and intellectually stimulating.

Integrating seamlessly with the user interface, the Gamification Module presents the approved quiz questions to visitors in an intuitive and visually appealing manner. As visitors interact with the questions, their responses are tracked and analyzed, allowing the system to assess the effectiveness of the quiz questions and overall engagement level.

In summary, the Gamification Module plays a central role in creating an interactive and educational environment for visitors. By leveraging the capabilities of the AI Subsystem, it generates personalized quiz questions that enhance visitors' understanding and enjoyment of the cultural and educational content within the designated area. The dynamic and immersive quiz-based interactions provide a unique and engaging experience, fostering a deeper connection between visitors and the exhibits or items on display.

3.6. System Architecture

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 user interface and the administrator application.

On the other hand, the frontend of the application is developed using React, a JavaScript library for building user interfaces. React is known for its component-based approach and virtual DOM rendering, which enhances the application's performance and user experience. By employing React for both mobile and administrator applications, the development team can reuse components and ensure consistent functionality and user interfaces across platforms.

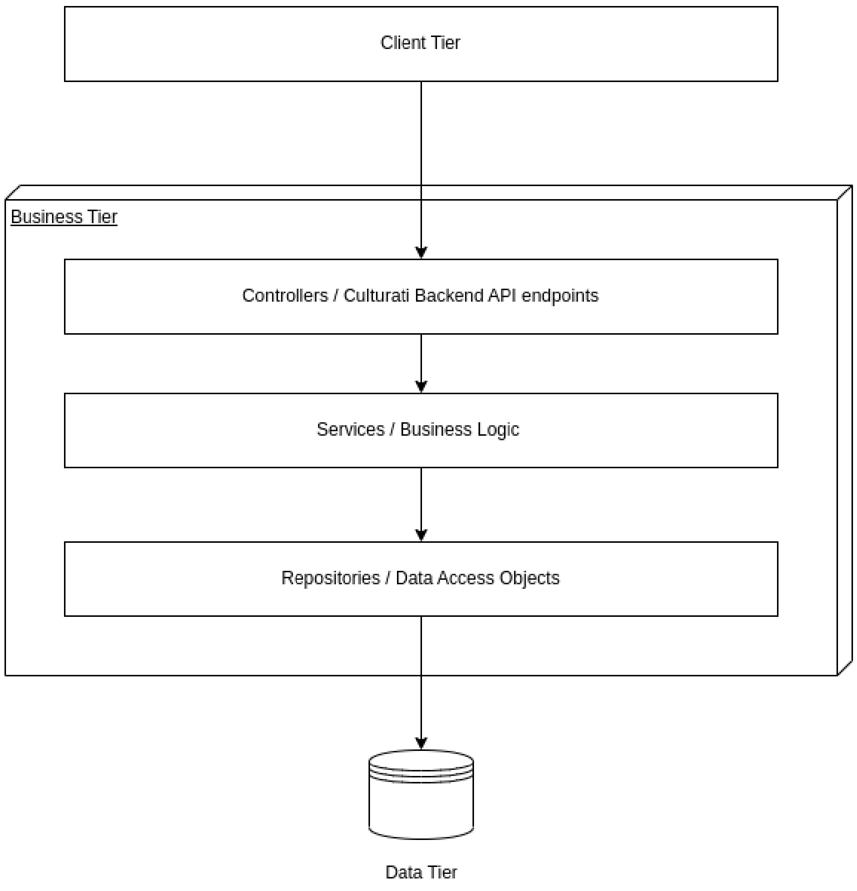
Client Tier
Business Tier
Controllers / Culturati Backend API endpoints
Services / Business Logic
Repositories / Data Access Objects
Data Tier

Figure 11: Core Back-end Architecture

3.6.1. Back-end Architecture

The back-end will be developed using the Spring Boot framework. The back-end architecture will include the following components:

Controllers: Controllers in Spring Boot handle incoming HTTP requests from both the mobile user interface and administration application. The requests are received through controller end-points 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 end-points 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.

3.6.1.1. Submodule Architecture

The architecture of the core back-end application utilizes submodules to achieve effective data separation. By employing this modular approach, we are able to compartmentalize distinct functional aspects of the application, thereby promoting organization, maintainability, and scalability. To implement this submodule approach CULTURATI's core back-end uses git submodules and java packages.

Submodules implemented using git submodules are beneficial to encapsulation by facilitating the organization and separation of different components. Each submodule retains its distinct version history and can be developed independently, resulting in isolation that safeguards the internal intricacies of the submodule's codebase within its dedicated repository. This isolation mitigates the risk of interference from other sections of the codebase.

Submodules also promote the separation of concerns, ensuring that each module focuses on a specific functionality. Therefore, they can be developed, tested, and maintained individually. This modular approach enhances codebase maintainability, as changes or updates to a specific module can be managed within its own repository, minimizing the impact on other parts of the codebase.

Encapsulating code in submodules makes it easier to version and reuse specific functionalities across projects. You can treat a submodule as a standalone component with a well-defined interface, allowing for more efficient reusability and sharing of code.

The core back-end codebase has been structured to encompass several key submodules, each serving a specific purpose. These submodules are designed to encapsulate discrete sets of functionalities, ensuring a clear and coherent separation of concerns. Here's a brief overview of the individual submodules:

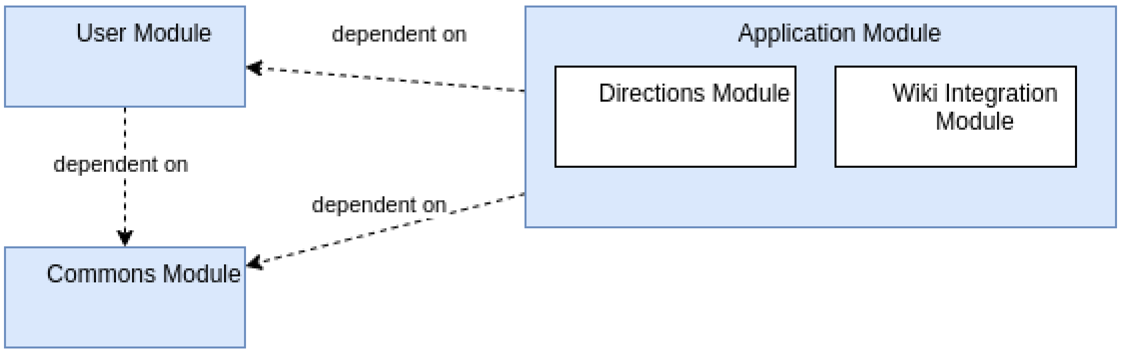

Figure 12: General architecture of the submodules and packages of the Core Back-end Application

commons: This submodule serves as a shared foundation, housing common utilities, resources, and components that are utilized across different parts of the application. It fosters reusability and consistency throughout the codebase.

user: The user submodule is dedicated to managing user-related functionalities, encompassing authentication, authorization, user profiles, and related operations. This focused approach streamlines user management while maintaining a clear boundary with other modules.

wiki-integration: The wiki-integration submodule establishes content management for users with the admin role. It facilitates creation, modification, and retrieval of content in the core application.

directions: The directions submodule tackles location-based services and route calculations. It enables the application to provide accurate and efficient directions, enhancing the user experience for geospatial interactions.

application: Lastly, the application submodule serves as the core engine that orchestrates the integration of the aforementioned submodules. It brings together their functionalities and orchestrates their interactions, ensuring a cohesive and harmonious overall application behavior.

By adopting this submodule-based architecture, we ensure not only a robust foundation for our backend application but also a flexible framework that can accommodate future expansions and enhancements.

3.6.1.2. 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.

3.6.1.3. 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.

3.6.1.4. 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.

3.6.1.5. 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.

3.6.1.6. 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.

3.6.1.7. 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.

3.6.1.8. 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.

3.6.2. Front-end Architecture

The front-end 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.

3.6.2.1. 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.

3.6.2.2. 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.

3.6.2.3. 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.

3.6.2.4. 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.

3.6.2.5. State Management

Redux will be utilized for efficient state management in the front-end. It will provide a centralized store to manage the 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.

3.6.2.6. 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 back-end APIs. It will enable smooth communication and data synchronization between the front-end and back-end.

3.6.2.7. Routing

To handle navigation and different views within the front-end, 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.

3.6.2.8. 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.

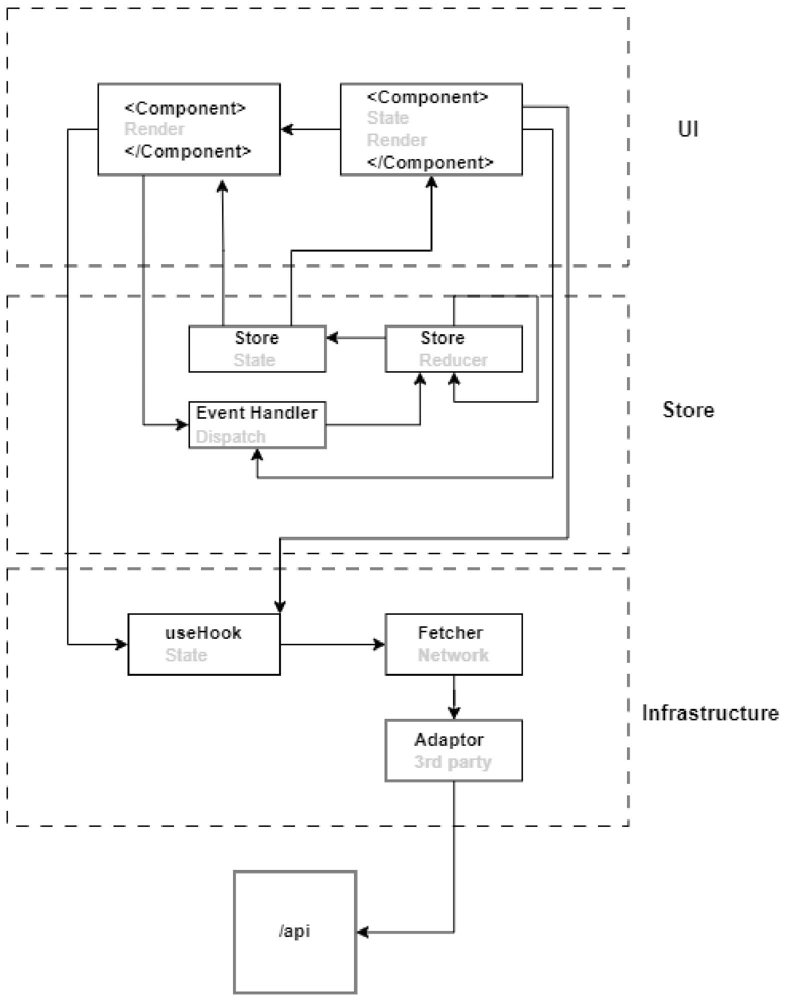

Figure 13: CULTURATI Front-end Component

3.6.3. 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 the 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.

3.7. Communication Between Software Components

3.7.1. Core Back-end and Front-end

The system architecture of the application involves seamless communication between the back-end and front-end through RESTful APIs. The front-end, developed using React and Next.js, interacts with the back-end by sending HTTP requests to the backend's API endpoints. These requests are responsible for tasks such as retrieving data, submitting forms, or performing various actions.

Upon receiving the HTTP requests, the back-end processes the incoming data and executes the required operations. This involves handling business logic, data manipulation, and any necessary interactions with databases or external services. The back-end then generates responses based on the processing results, which include requested data or appropriate status codes.

Through this communication mechanism, data flow between the front-end and back-end is facilitated, allowing for a dynamic and interactive user experience. The front-end can efficiently retrieve and update information, ensuring that the user interface remains up-to-date and responsive to user interactions. The RESTful API approach promotes a well-organized and standardized communication process, enabling different parts of the application to work harmoniously together.

3.7.2. Internal Communication of the Core Back-end

The core back-end of the CULTURATI application integrates git submodules and Java packages to establish encapsulation. This approach sets an organized and modular architecture.

Git submodules serve as the initial layer of this architectural strategy. They enable separating various components or features into discrete, standalone units. Each submodule represents a distinct and self-contained module of functionality.

Complementing the use of git submodules, Java packages provide a hierarchical structuring mechanism. These packages act as containers for organizing related classes and interfaces within each submodule. This internal structuring ensures that the encapsulation achieved at the submodule level is further upheld within the individual components. By assigning classes and interfaces to appropriate packages, a clear and intuitive structure emerges, promoting code readability and navigation.

The coordination between git submodules and Java packages is underscored by the establishment of interfaces. These interfaces act as communication gateways, defining the contracts and interactions between different submodules and their encapsulated functionalities. By adhering to these interface specifications, submodules can seamlessly collaborate without delving into one another's implementation details.

3.7.3. Core Back-end and the Wiki instance

The wiki instance employs a GraphQl API where it provides CRUD (Create, Read, Update, Delete) operations. These operations are made accessible through end-points, enabling the core back-end to manipulate the wiki data. The communication is established via GraphQl queries, initiated from the core back-end and directed towards the wiki instance.

When a user intends to engage with the wiki instance through the core application, first, the core back-end authorizes through an authorization request. The necessary credentials are created through the wiki instance’s UI and are preconfigured in the configuration files specific to the core backend user’s tenant. Using these credentials, the core back-end establishes authorization with the wiki and then interprets and sends the users request to the wiki instance.

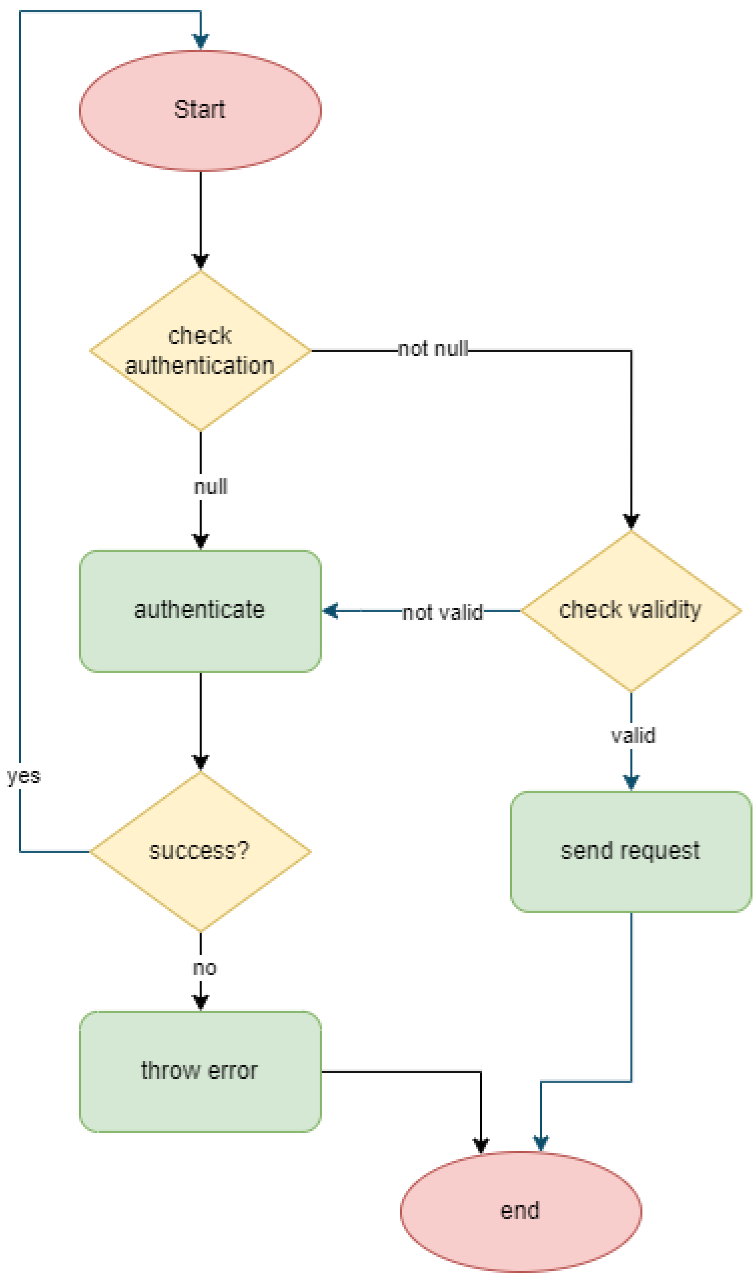

Figure 14: Diagram showing the authorization flow for requests made to the wiki instance

The core back-end facilitates CRUD operations of pages in the wiki. However, it is important to note that while the core back-end governs the Create, Read, Update, and Delete processes for pages, it does not directly engage in the creation or modification of the content within these pages.

When a user initiates the creation or modification of page content, the core back-end redirects the user to the user interface of the wiki instance. This redirection ensures that the user interfaces with the wiki instance's dedicated UI environment, where content creation and updates are managed. This architectural delineation streamlines the overall process, with the core back-end responsible for the synchronization and data integrity of the entities in the system, while the task of generating and updating content is delegated to the wiki instance's user interface.

3.7.4. Core Back-end and the AI Module

The interaction between the Core Backend and AI Module is facilitated via a REST API, offering a comprehensive range of operations. These operations are accessible through designated end-points, enabling both the Core Back-end and AI Module to exchange and retrieve data.

Whenever one module requires data from the other, the process begins with an authorization request. The essential credentials are preconfigured within the configuration files of their respective applications. These credentials serve as the means by which the modules authenticate with each other and subsequently send requests to fulfill their specific requirements.

3.8. Version Control Practices

All core applications projects and modules leverage the capabilities of Github repositories. This adoption of Github repositories provides a robust and centralized platform for overseeing the evolution and management of the application's source code. Github's version control mechanism operates through the Git system. Git records every alteration made to the source code, creating a chronological log of commits that can be revisited, compared, and analyzed.

The feature branching approach is employed for both the back-end and front-end repositories. Given the core application consists of distinct submodules, the component-based workflow is also utilized.

In the context of version control and collaboration in software development, the "feature branching approach" refers to the practice of creating separate branches for each new feature or bug fix. This segregation enables developers to work on specific changes independently without affecting the main codebase. Once a feature is complete, it can be merged back into the main branch after review and testing.

A "component-based workflow" revolves around managing different components or services of an application as separate entities. Each component or service has its own repository and development process, allowing teams to work on specific functionalities in isolation. This approach is particularly effective in complex projects where different components interact and need to be developed, tested, and deployed independently.

4. NIMBEO Architecture

NIMBEO's goal in the CULTURATI project is to design and create a Wiki. This Wiki will serve as a centralized repository, storing crucial information about museums and places, including exhibition details, multimedia content, and accessibility information. Wiki's user-friendly interface, customization options, content management flexibility, and version control ensure seamless content updates and data integrity. Additionally, its collaborative editing, security measures, and integrations empower Nimbeo to deliver a comprehensive and engaging cultural exploration platform, enriching visitors' experiences and fostering knowledge-sharing within the community.

4.1. Key aspects of the Wiki

The CULTURATI's Wiki is a user-friendly interface designed to enable efficient and collaborative knowledge-sharing within organizations. With a focus on simplicity and accessibility, The Wiki offers a feature-rich platform for creating, organizing, and maintaining internal documentation and knowledge bases.

At the core of the Wiki is its intuitive and modern user interface, which allows users to effortlessly create and edit pages, encouraging active contributions from team members. Real-time editing capabilities facilitate seamless collaboration, making it an ideal choice for distributed teams and organizations.

Key features of the Wiki include a flexible markdown editor, version control for pages, granular access controls, and a powerful search functionality. These features ensure that content remains up-to-date and accurate while allowing administrators to manage permissions effectively.

The flexible markdown editor further enhances collaboration by enabling users to format easily and structure content, adding headings, lists, images, and links with simplicity and efficiency. This intuitive editor reduces barriers to entry for non-technical contributors, encouraging a diverse range of team members to participate in content creation and updates actively.

Moreover, the Wiki's granular access controls empower administrators with the ability to define specific user roles and permissions, tailoring access to different sections and functionalities based on individual responsibilities and organizational hierarchies. This fine-grained control safeguards sensitive information while allowing the right people to access the appropriate resources. Administrators can effortlessly manage permissions, granting edit access to content creators, read-only access to stakeholders, and complete access to designated leaders or supervisors.

Security is a top priority in the Wiki’s workflow, and the code implements robust authentication and authorization mechanisms to protect sensitive information. Additionally, data backups and restoration features safeguard against accidental data loss, providing peace of mind to administrators.

Finally, the Wiki's powerful search functionality ensures that users can swiftly locate the information they need amid vast repositories of content. Advanced search filters and keyword highlighting streamline the process of finding relevant articles, documents, and data, saving valuable time.

4.1.1. CULTURATI wiki side overview

The Wiki is built on a modern and flexible architecture that facilitates efficient knowledge-sharing and documentation management. At its core, the CULTURATI Wiki follows a client-server architecture, enabling smooth communication between the front-end and back-end components.

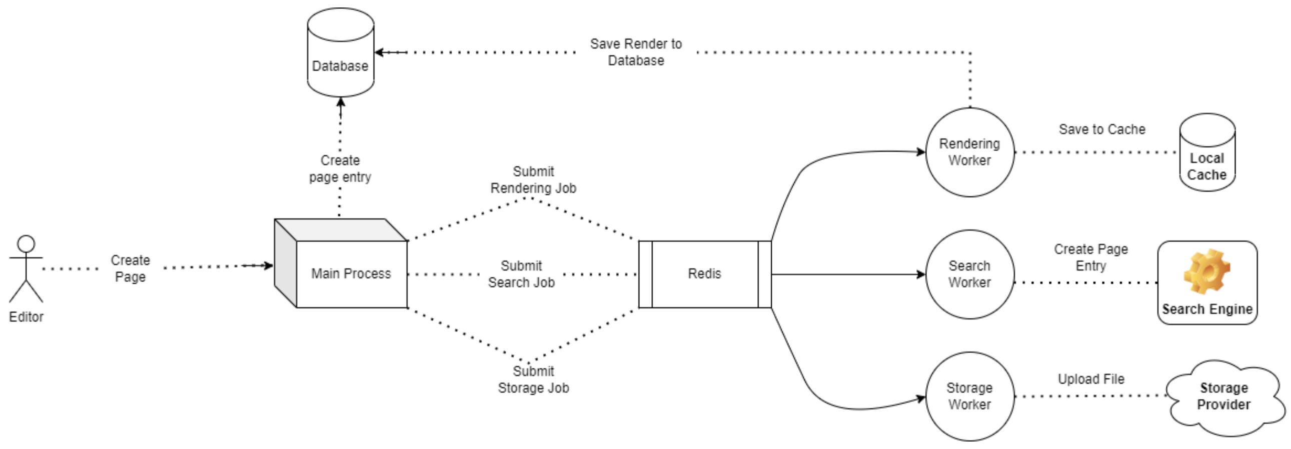

Figure 15: Wiki's General Process

Front-end Architecture:

The front-end of the Wiki is developed using modern web technologies, primarily built on JavaScript, HTML, and CSS. It embraces a component-based architecture, dividing the user interface into reusable and modular components. This approach promotes code maintainability and extensibility, making it easier for developers to add new features or customize existing ones.

By adopting a component-based architecture for the front-end, the Wiki gains a considerable advantage in terms of scalability and development efficiency. Each component is designed to be self-contained, encapsulating its unique functionality and appearance. This modularity allows developers to work on different sections of the application independently, fostering parallel development and reducing the risk of introducing unintended bugs. Moreover, the re-usability of components accelerates the development process as developers can leverage existing components to create new features, reducing redundant code and saving valuable time.

In addition to its modern web technologies, the Wiki prioritizes responsive design principles, ensuring that the user interface adapts seamlessly to various screen sizes and devices. This responsiveness is crucial in today's mobile-first world, where users access information from smartphones, tablets, laptops, and desktops. By providing a consistent and optimized experience across different platforms, the Wiki enhances user satisfaction and accessibility, enabling contributors and readers to access content from virtually anywhere.

Server-side Architecture:

On the server-side, the Wiki employs Node.js, a fast and scalable JavaScript runtime, to handle back-end operations. Node.js's event-driven, non-blocking I/O model ensures high performance and

responsiveness, making the CULTURATI’s Wiki suitable for concurrent user interactions.
One of the key advantages of Node.js is its event-driven architecture, which allows the server to handle multiple simultaneous operations without being blocked by any single request. This non-blocking I/O model ensures that the server can efficiently manage concurrent user interactions, making the Wiki highly responsive even during periods of high traffic and user activity. As a result, users can seamlessly navigate through the platform, edit content, and perform searches without experiencing delays or slowdowns, enhancing the overall user experience.

Additionally, Node.js is renowned for its scalability, making it well-suited for handling the demands of a rapidly growing user base or a large enterprise environment. With its ability to handle a significant number of concurrent connections, the Wiki can effortlessly accommodate an expanding community of contributors and readers without compromising on performance.

Data Storage and Database:
The CULTURATI’s Wiki stores its data in a lightweight and efficient database using PostgreSQL. The database is responsible for storing page content, user information, access control settings, and other relevant data.

The decision to utilize PostgreSQL as the database for the CULTURATI’s Wiki brings forth a host of advantages, making it an ideal choice for storing and managing crucial data. PostgreSQL is an open-source, object-relational database management system that has earned a reputation for its robustness, stability, and performance. Its mature and well-established architecture ensures data integrity and reliability, making it a dependable foundation for the Wiki's content and user information.

One of PostgreSQL's standout features is its support for complex data types and advanced querying capabilities. This flexibility allows the wiki to store a diverse range of content efficiently, including rich media, structured documents, and user-generated data. Moreover, the database's ability to handle large volumes of data and execute complex queries in a performant manner guarantees that the Wiki can manage a vast repository of articles, revisions, and user profiles without compromising on response times.

APIs and Communication:

The front-end and back-end of CULTURATI's Wiki communicate via RESTful APIs. These APIs enable seamless data exchange between the client and server, allowing real-time updates and interactions. RESTful APIs also facilitate integration with other applications and services.

REST (Representational State Transfer) is an architectural style that relies on standard HTTP methods, making it lightweight, simple, and widely supported across various platforms and programming languages. By adhering to RESTful principles, the wWiki's APIs ensure a standardized and intuitive interface for data exchange, facilitating smooth interactions between the client-side and server-side components.

Moreover, RESTful APIs promote modularity and decoupling between the front-end and back-end of the wiki. The well-defined API endpoints allow each component to evolve independently without affecting the other. This separation of concerns enables the front-end and back-end teams to work in parallel, streamlining the development process and reducing potential conflicts.

Authentication and Authorization:

Security is a fundamental aspect of the CULTURATI's Wiki architecture. The library implements robust authentication and authorization mechanisms to control user access and ensure data privacy. Administrators can define user roles with specific permissions, ensuring that only authorized users can create, edit, or delete content.

One of the cornerstones of the platform's security is the use of encryption for sensitive data both in transit and at rest. When data is transmitted between the front-end and back-end or exchanged with external services, it is encrypted using industry-standard protocols such as TLS (Transport Layer Security). Additionally, data stored in the database, including user credentials and other confidential information, is encrypted at rest, ensuring that even in the event of unauthorized access to the underlying storage, the data remains protected.

To safeguard against common security vulnerabilities, the CULTURATI's Wiki employs rigorous input validation and sanitation mechanisms. This proactive approach helps prevent malicious input, such as SQL injection or cross-site scripting (XSS) attacks, from compromising the platform's integrity or exposing sensitive information. By validating and sanitizing user inputs at every stage of data processing, the wiki significantly reduces the risk of data breaches and unauthorized access.

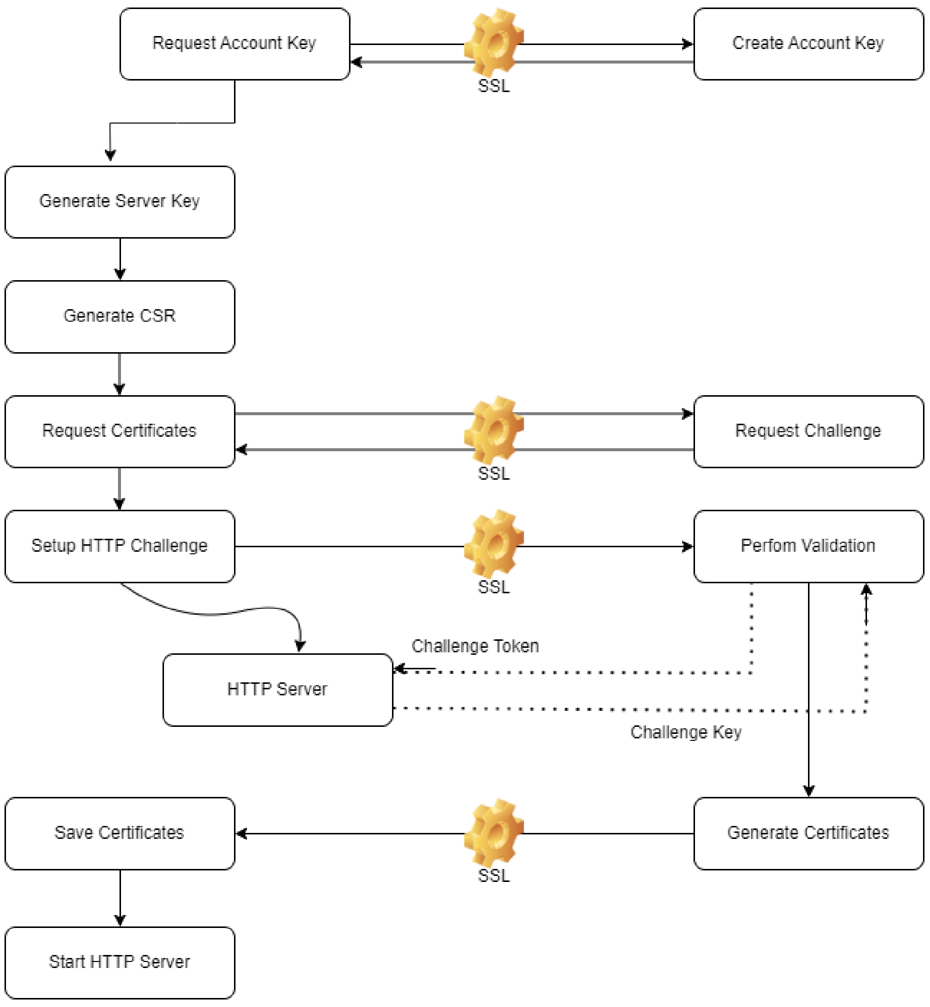

Figure 16: Wiki Authentication Process

4.1.2. Data Architecture

The data architecture of CULTURATI's Wiki plays a crucial role in efficiently managing and organizing the content and user-related information within the platform. The Wiki relies on a database to store various types of data, such as page content, user details, access control settings, and system configurations. The choice of database can be tailored to meet specific needs, but popular options include PostgreSQL.

4.1.2.1. Page Content:

The core element of the Wiki is the content of individual pages. Each page may contain rich text, images, links, and other multimedia elements. The wiki stores this page content in the database, ensuring that it remains persistent and accessible for users across sessions.

The rendering module within the Wiki enhances content processing capabilities by introducing additional features to the platform. As an integral part of the rendering pipeline, this module plays a vital role in transforming raw content into its final HTML form.

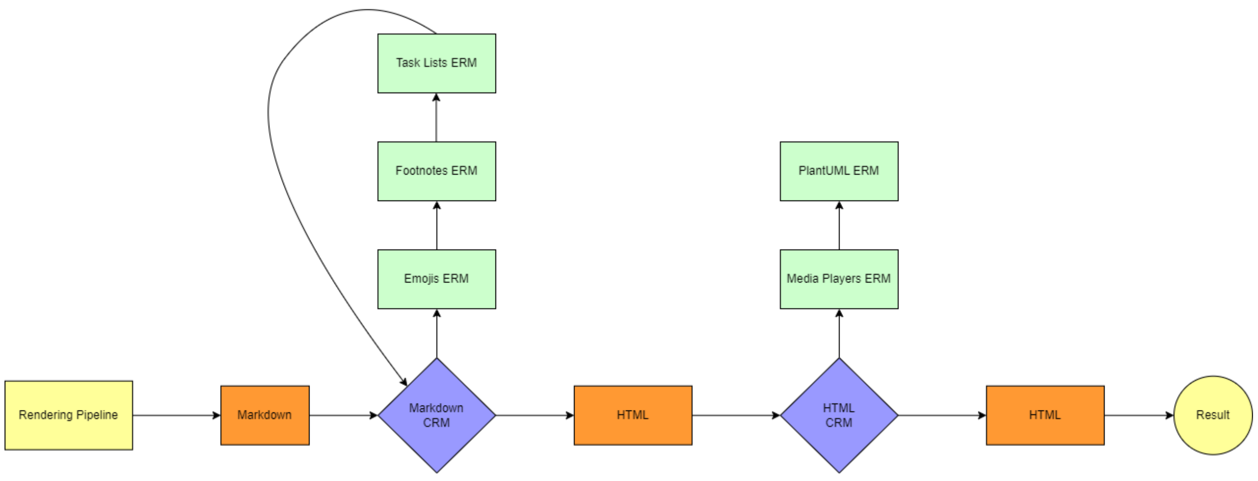

Figure 17: Wiki Extension Rendering Module Flow

In the rendering pipeline, the rendering module takes on the responsibility of parsing and processing the original content provided by users or administrators. It employs a series of sophisticated algorithms and transformations to convert the raw input into a visually appealing and well-structured HTML representation. By integrating the rendering module into the Wiki architecture, the platform gains the ability to handle diverse types of content efficiently. This Includes supporting various media formats, embedding rich media elements, handling code syntax highlighting, and managing the organization of text, images, and links. The rendering pipeline's meticulous parsing and transformation process ensure that the content presented to users is consistent, accessible, and easily digestible. It optimizes the user experience by delivering visually coherent and interactive pages, promoting effective knowledge-sharing and communication within the community.

4.1.2.2. User Information:

User management is essential for any collaborative platform, and the CULTURATI’s Wiki is no exception. The data architecture stores user information, including usernames, email addresses, and hashed passwords for authentication purposes. User roles and permissions are also managed, allowing administrators to define access levels for different users.

As users are a central aspect of the platform, the data architecture stores essential user information in a secure and organized manner. This includes user attributes such as usernames, email addresses, and hashed passwords. Hashing passwords before storage ensures that sensitive information remains protected even in the event of a data breach.

Beyond basic user information, the wiki's data architecture incorporates features for managing user roles and permissions. Administrators hold the authority to assign specific roles to users, granting them distinct levels of access and capabilities within the platform. User roles typically include categories like administrators, editors, contributors, and readers, each with varying degrees of privileges. For instance, administrators possess complete control over the platform's settings, user management, and content moderation, while editors can create, modify, and publish content.

4.1.2.3. Access Control Settings:

Access control settings govern the permissions granted to users, determining their ability to create, edit, or delete content. These settings are associated with user roles and are stored in the database to enforce proper data privacy and security measures.

4.1.2.4. Version Control:

The Wiki often implements version control for pages, allowing users to track changes made to content over time. This version history is stored in the database, enabling administrators to revert to previous versions if necessary and ensuring data integrity.

4.1.2.5. Configuration Settings:

The data architecture stores system configuration settings, which allow administrators to customize various aspects of the Wiki. This includes options for enabling or disabling features, setting up integrations with other systems, and configuring the appearance of the user interface.

One of the key functions of the system configuration settings is to enable or disable specific features of the Wiki. Administrators can toggle various functionalities on or off, depending on the organization's preferences or the specific use case. For instance, they can activate real-time collaborative editing, content versioning, or user commenting based on the collaborative workflow that best suits their team and community. The ability to enable or disable features provides a level of flexibility that ensures the Wiki remains adaptable to different contexts and user preferences.

4.1.2.6. Metadata and Search Indexing:

To support efficient search functionality, the CULTURATI’s Wiki may employ metadata and search indexing techniques. Metadata tags associated with pages are stored in the database, enabling faster and more accurate search results for users.

Metadata plays a crucial role in enriching the content of pages with additional information and context. Each page is associated with relevant metadata tags, which describe its content, category, author, creation date, and other pertinent details. These metadata tags are stored in the database alongside the page content, creating a comprehensive index that facilitates swift and targeted searches.

To further optimize search performance, the Wiki employs search indexing techniques. Search indexing involves creating a separate, optimized data structure that maps key terms and metadata to the corresponding pages in the database. This indexing process pre-computes the results of potential search queries, improving search response times significantly. When a user enters a search query, the search engine consults the search index rather than scanning the entire content of each page, leading to faster retrieval of matching pages.

4.1.3. Integration and API's

Wiki offers several integration options through its API (Application Programming Interface), which allows external applications to interact with the Wiki pro-grammatically. The API follows RESTful principles, providing standardized end-points for data exchange between the front-end and back-end components of the Wiki. This RESTful API architecture makes it easier for developers to connect the Wiki with other web services and applications.

With the Wiki API, developers can perform a variety of tasks, such as creating, updating, and deleting pages, managing user accounts and permissions, and retrieving content or metadata from the wiki. This API enables seamless integration of the wiki into custom applications or third-party tools, extending the platform's capabilities beyond traditional wiki functionalities.

Additionally, the Wiki supports various authentication and authorization mechanisms to ensure secure API access. Administrators can configure access controls and user permissions, enabling fine-grained control over API operations. This robust security infrastructure protects sensitive data and ensures that only authorized applications or users can interact with the Wiki through the API.

Conclusion

In conclusion, the presented System Architecture Design and Specification Report outlines the creation of an innovative application that seamlessly merges Java Spring Boot, React.JS, and AI algorithms to craft a unique visitor experience with the following main aspects;

Innovative Integration for Personalized Experiences

By harmonizing the technologies stated above, our proposed system optimizes movement within indoor and outdoor spaces, offering guided tours and comprehensive information through a connected Wiki.JS library. Content creators also play a role in enriching engagement through tailored content.

Agile Development with Micro-Services

The adoption of a micro-services architecture ensures adaptability and scalability, enabling independent evolution and deployment of distinct components. Java Spring Boot forms a robust backend foundation, capable of managing diverse functionalities, including user administration, AI algorithms, and database interactions. Meanwhile, the React.JS-powered frontend guarantees a modern, responsive, and reusable user interface.

AI-Powered Optimization and Personalization

Our pioneering integration of AI algorithms enables real-time data analysis and enhances visitor flows, offering personalized routes based on AI-driven insights. This feature-driven approach elevates the visitor experience by minimizing congestion and offering tailored experiences.

Rich Information Access through Wiki Integration

The seamless connection to a Wiki.JS library serves as a knowledge hub, offering visitors historical insights, exhibit details, and the ability to engage with the community. A well-defined administration role ensures the accuracy of site information and encourages user interaction.

Collaborative Content Creation: Fostering Engagement and Diversity

Expanding on the content creation platform, it's important to emphasize its pivotal role in cultivating collaboration among content creators. This platform will not only enrich the system with diverse and dynamic content but also acts as a catalyst for building a vibrant community of contributors who are deeply invested in the success of the system.

Security, Compliance, and Adaptability

With a focus on security and data privacy, robust authentication, authorization mechanisms, and data encryption protect sensitive user information and ensure compliance with data protection regulations. The system's adaptability extends to different environments, making it suitable for museums, galleries, and historical sites.

Continual Improvement and Integration

While partnerships and collaborations could enhance the system's value, our commitment to continuous improvement ensures its relevance over time. Leveraging RESTful APIs, the system enables seamless communication and integration between modules, promoting real-time updates and data consistency.

Setting a New Standard

In essence, our system architecture design not only promises an exceptional visitor experience through the convergence of cutting-edge technologies but also establishes a solid foundation for future expansion and innovation. With user-centric design, security measures, and a commitment to excellence, our proposed system redefines the integration of technology, visitor engagement, and operational efficiency.