Architecture

4+1 Architecture Models

To make the system clear to the user and other developers, we include many diagrams to introduce this product in the following views. Most of the techniques used in this product are listed in detail in the technique details page.

Logical View

- Database Model Diagram
- State Diagram

Process View

- *Planned in Sprint 3*

Development View

- System Diagram
- Container Diagram

Physical View

- Deployment Pipeline (On Plan)

Scenario / Use Case View

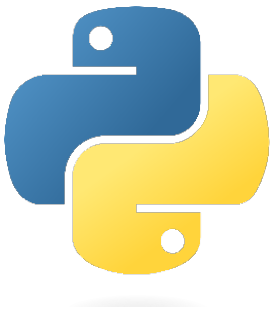
- Sequence diagrams

Reference

https://cis-projects.github.io/project_based_course_notes/topics/architecture.html?highlight=diagram

Technique Detail

Programming language



Architecture Overview

Front-end and back-end separation

From the perspective independently, we separate the frontend and backend, so that each part can be developed, maintained, and scaled independently. From the perspective of software future extensibility, as we aim to handover several docker images including backends docker images and frontend docker images for our client. This separation of concerns simplifies the development process, so that we can implement the code by separated the groups into frontend and backend. For the project to be further extent in the future, this architecture is easier to allow it to be developed as a complex applications. On the other hand, if the client decide to change the backend technology or migrate to a different frontend framework in the future implementation of the project, having a clear separation between the two will make the transition smoother and less disruptive.

Microservice

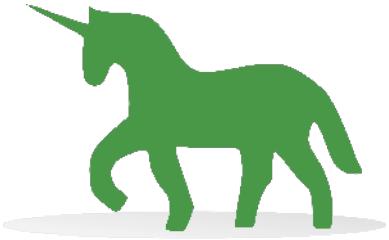
Microservice is an popular architecture in todays software, as it offers various benefits in terms of scalability, maintainability, and flexibility. Microservices can be independently scaled based on their specific resource requirements or load, allowing the project to handle increased demand more efficiently. This can lead to more cost-effective use of resources and better performance under varying workloads. In addition, microservices can be implemented using different programming languages, frameworks, or technologies, based on the requirements of each service. This enables the future development teams who aim to improve this project to choose the best tools for their specific needs and avoid being locked into our current technology stack. Furthermore, microservices are smaller and more focused than monolithic applications, making them easier to understand, maintain, and update. This can lead to improved code quality, reduced technical debt, and a lower likelihood of bugs or issues. Lastly, Microservice architecture promotes modular design and separation of concerns, making it easier to manage complexity and maintain a clear focus on individual components, and each service can be on different infrastructure components, allowing for more efficient use of resources and better distribution of workloads.

Backend

Flask, Gunicorn, and Nginx are popular choices for web development projects due to their ease of use, flexibility, and high-performance. These choices allow the team to quickly develop and deploy a robust, scalable and efficient web application that meets the client's requirement.

Flask provides the freedom to design the application's structure and architecture as you see fit, making it suitable for a wide variety of projects with different requirements and constraints. As the project is to intergrate the functions from Mapping tools such as Ontoserver.

More about Flask



Gunicorn is usually applied as a combination with Flask appication. By using Gunicorn as the WSGI server, the application can efficiently manage multiple concurrent requests, leading to better performance and a more responsive user experience.

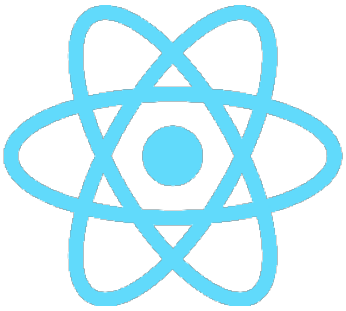
More about Gunicore

Nginx is widely used which is known for its high-performance capabilities and ability to handle a large number of simultaneous connections. It also provides load balancing, caching, and SSL termination, which further improves your application's performance, security, and reliability.

More about Nginx

Frontend

We use React and Ant Design for the frontend, the project can benefit from a modular, maintainable, and scalable architecture that enables efficient development and collaboration. This approach also allows for greater flexibility in adapting to future changes or integrating with external services.



React is a popular, powerful, and efficient JavaScript library for building user interfaces. It utilizes a component-based architecture, making it easy to create modular, reusable UI components. React's virtual DOM ensures high performance and efficient updates, leading to a smoother user experience.

More about React

Ant Design is a comprehensive UI design framework for React applications. It provides a wide range of pre-built, customizable components that follow a consistent design language, which accelerates the development process and ensures a professional-looking application.

More about Ant design

Database

MongoDB is a popular NoSQL database that stores data in a flexible BSON format. It is chosen for this project because it offers schema flexibility, allowing data storage without a fixed structure, which is useful for diverse data types and evolving data models. Additionally, it provides horizontal scalability, making it suitable for handling large data volumes and high-traffic loads. MongoDB's document-based storage model results in faster and more efficient queries compared to traditional relational databases, and it supports indexing and caching for improved query performance. The database also features a rich query language, enabling developers to build complex and efficient queries. Furthermore, MongoDB has extensive support for various programming languages, including JavaScript, Python, Java, and C#.

More about mongoDB

Deployment

We use Docker and Ansible for deployment which can make the project benifitted from the consistency, scalability, and automation. It can streamline the deployment process and improve infrastructure's reliability. However, Ansible rquire a learning curve, which might be a challenge for future teams.

Docker enables us to package the project and its dependencies in a container, ensuring consistency across development, testing, and production environments. This eliminates the "it works on my machine" problem and streamlines deployment. In addition, as we decided to use stucture our project as microservice architecture, docker is the most prevelent tools to helps us to fulfill this goal. Furthermore, D ocker containers run in isolation, reducing the risk of conflicts between applications or dependencies. This isolation improves the security and reliability of the services. Lastly, Docker makes it easy to scale applications horizontally by deploying additional containers. This scalability allows future teams to add more functions("components") into the system.

More about Docker

Ansible uses a declarative language, implemented by python, to define the project infrastructure and configurations, which means the infrastructure is treated as code. This enables version control, easier collaboration, and improved maintainability. It automates the deployment, configuration, and management of infrastructure by reducing manual effort and the potential for human error. Unlike some other configuration management tools, Ansible is easy to install, because it is a python package, and can be easy install by 'pip install ansbile' which does not require an agent to be installed on the production system.

Ansible's playbooks are idempotent, meaning they can be run multiple times without causing unintended side effects. This ensures consistent and predictable results when applying configuration changes.

More about Ansible

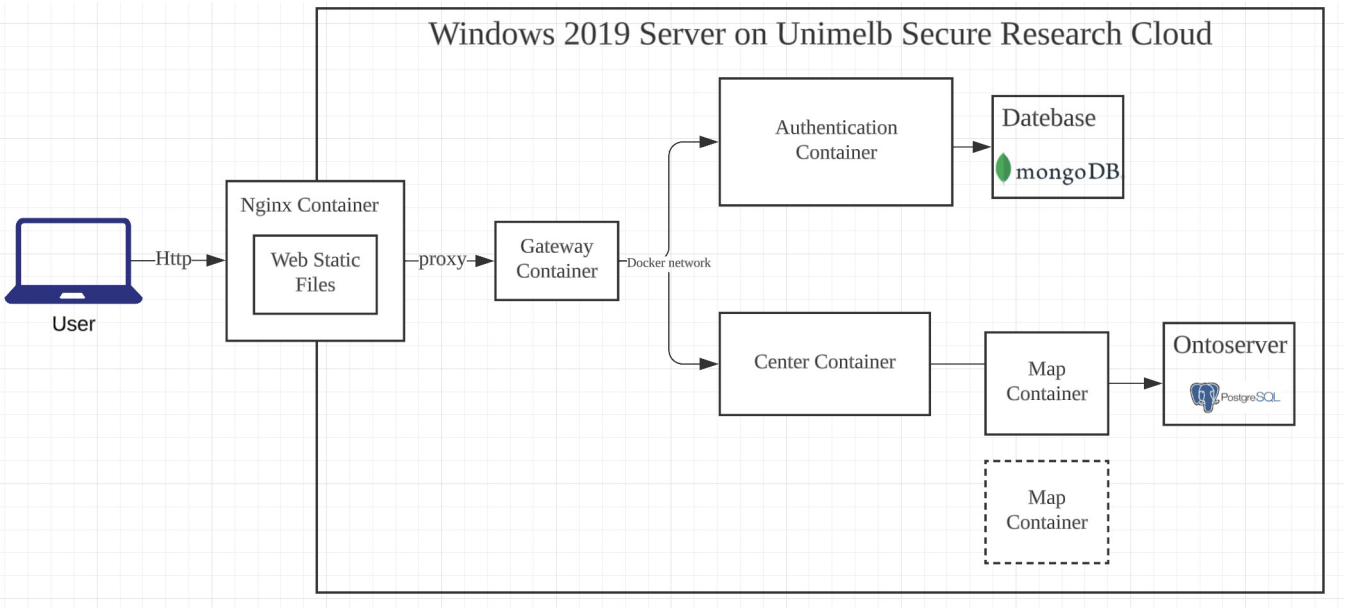
4+1 Architecture Models

Title	Creator	Modified
Database Model	KUNXI SUN	31 May, 2023
Container Diagram	KUNXI SUN	30 May, 2023
System Diagram	KUNXI SUN	30 May, 2023
State Diagram	KUNXI SUN	30 May, 2023
Sequence Diagrams	KUNXI SUN	30 May, 2023

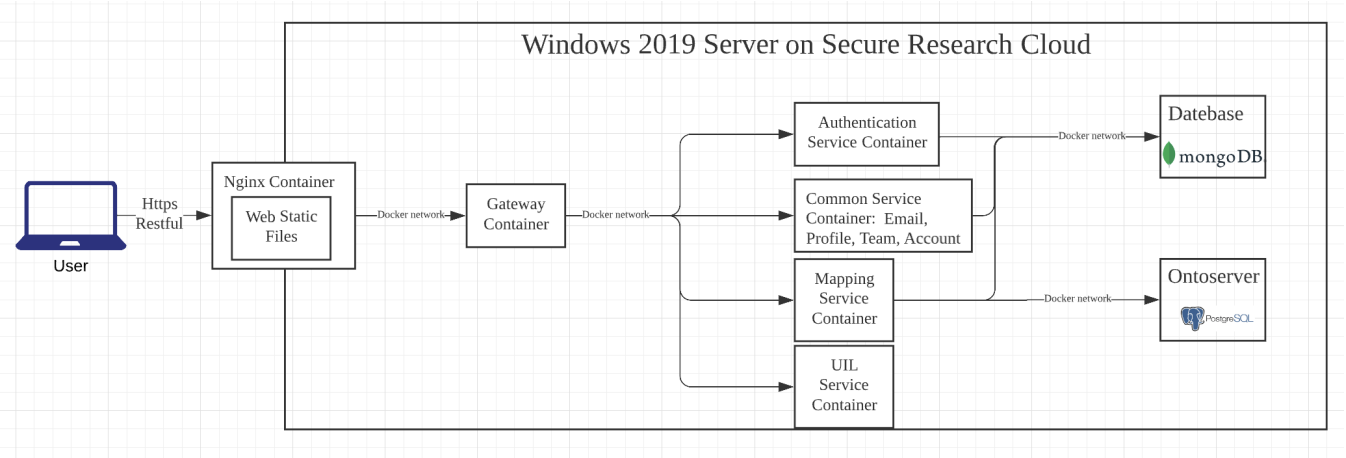
Container Diagram

Version	Description	Date
2.0.0	Change the container diagram aligning with the latest structure	30 May 2023
1.0.0	Basic container diagram	28 Apr 2023

Version 2.0.0



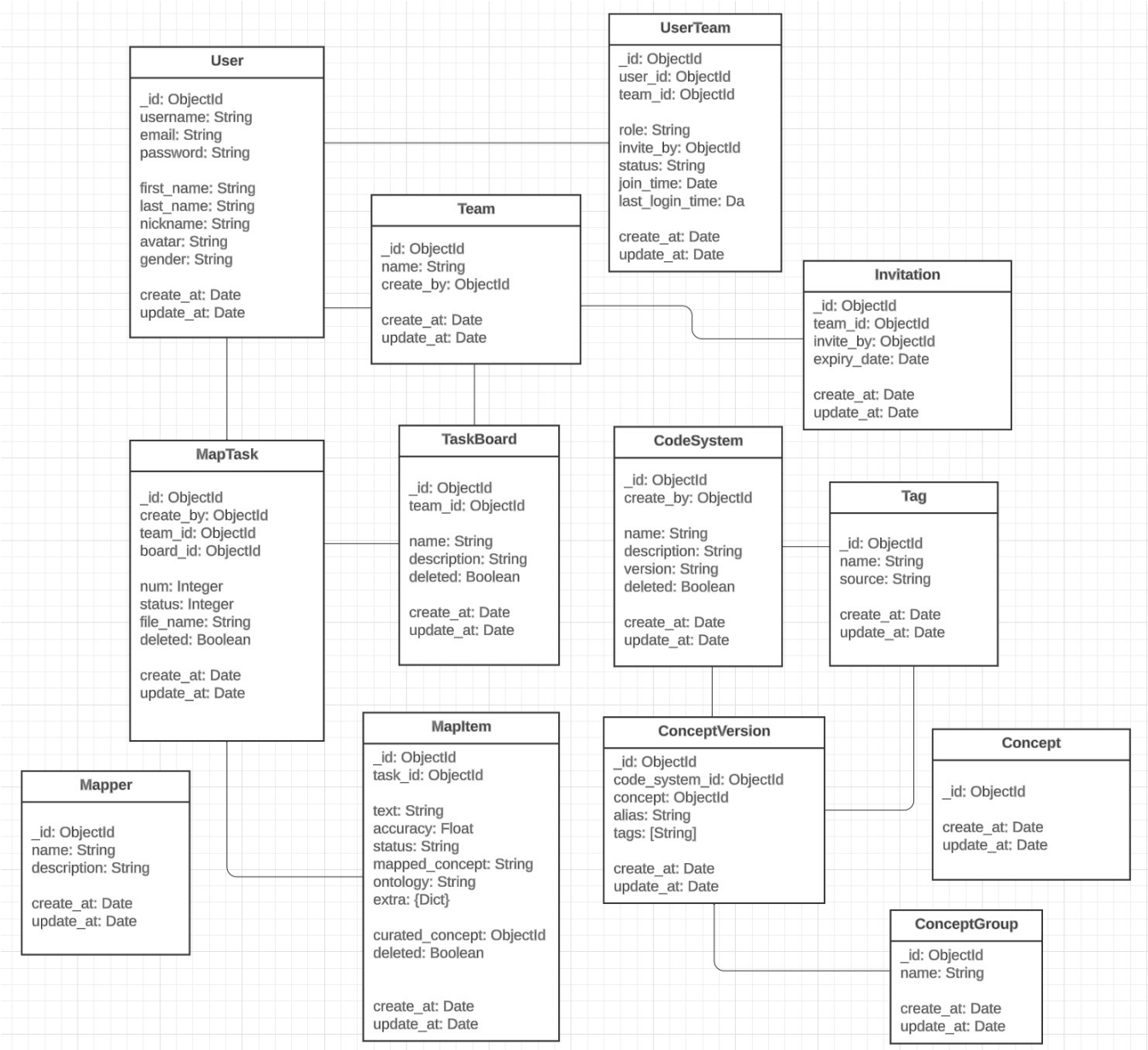
Version 1.0.0



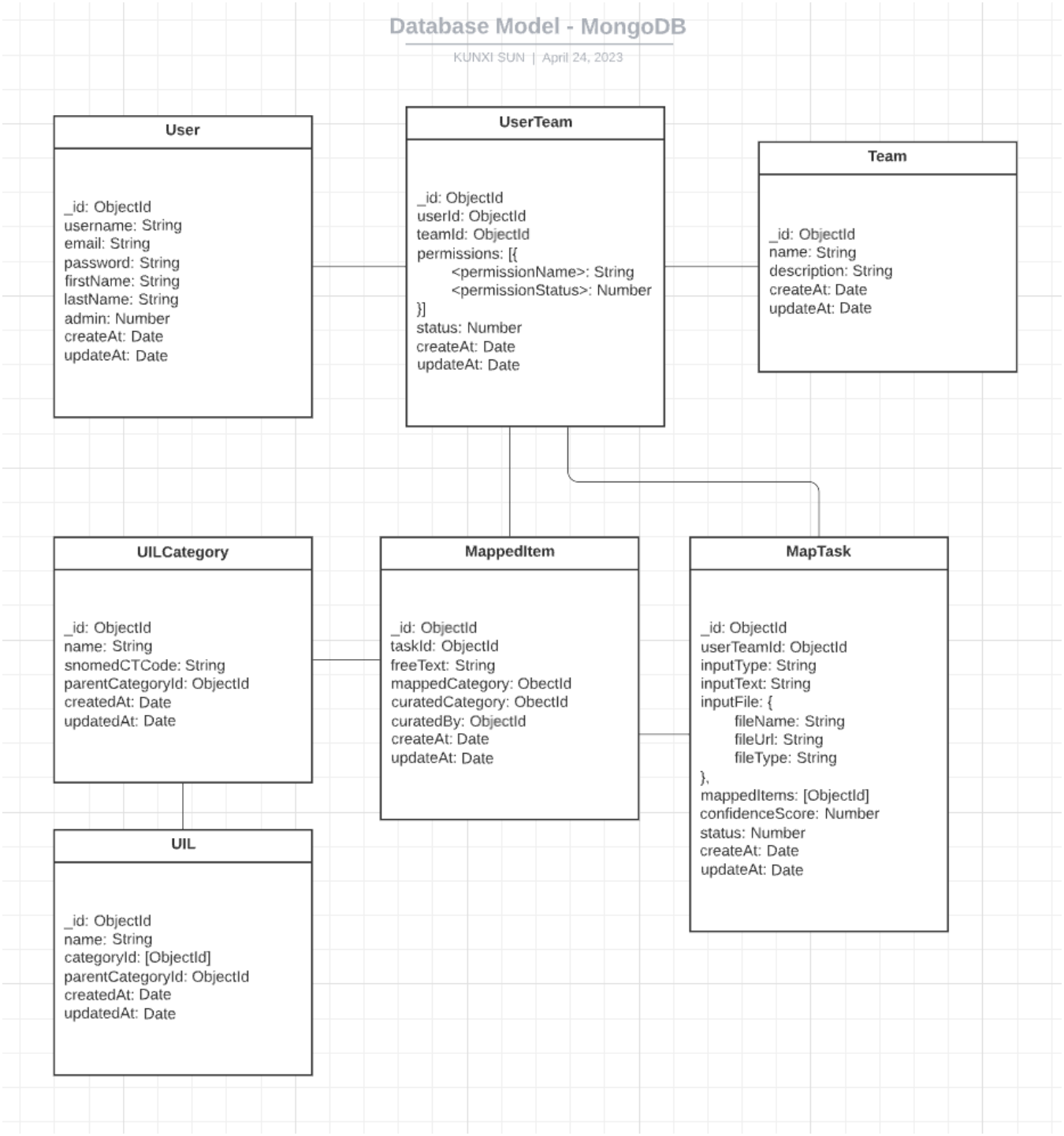
Database Model

Version	Description	Date
Version 2.0.0	1. Updated MongoDB data models with latest system	30 May 2023
Version 1.0.0	1. Basic MongoDB data models with basic relations	24 Apr 2023

Version 2.0.0



Version 1.0.0

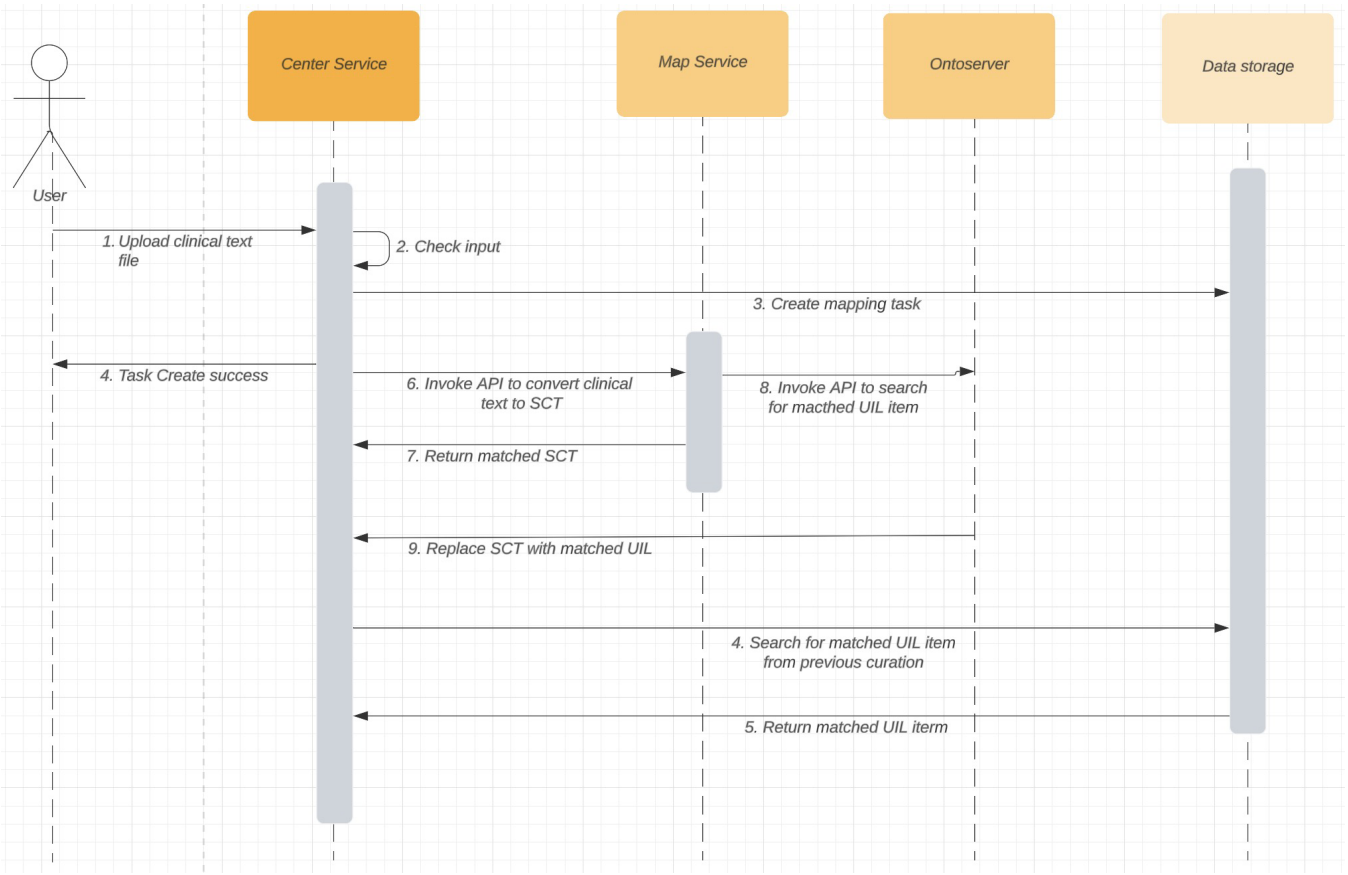

Database Model - MongoDB
KUNXI SUN | April 24, 2023
User
_id: ObjectId
username: String
email: String
password: String
firstName: String
lastName: String
admin: Number
createAt: Date
updateAt: Date
UserTeam
_id: ObjectId
userId: ObjectId
teamId: ObjectId
permissions: [{
<permissionName>: String
<permissionStatus>: Number
}]
status: Number
createAt: Date
updateAt: Date
Team
_id: ObjectId
name: String
description: String
createAt: Date
updateAt: Date
UILCategory
_id: ObjectId
name: String
snomedCTCode: String
parentCategoryId: ObjectId
createdAt: Date
updatedAt: Date
MappedItem
_id: ObjectId
taskId: ObjectId
freeText: String
mappedCategory: ObectId
curatedCategory: ObectId
curatedBy: ObjectId
createAt: Date
updateAt: Date
MapTask
_id: ObjectId
userTeamId: ObjectId
inputType: String
inputText: String
inputFile: {
fileName: String
fileUrl: String
fileType: String
},
mappedItems: [ObjectId]
confidenceScore: Number
status: Number
createAt: Date
updateAt: Date
UIL
_id: ObjectId
name: String
categoryId: [ObjectId]
parentCategoryId: ObjectId
createdAt: Date
updatedAt: Date

Sequence Diagrams

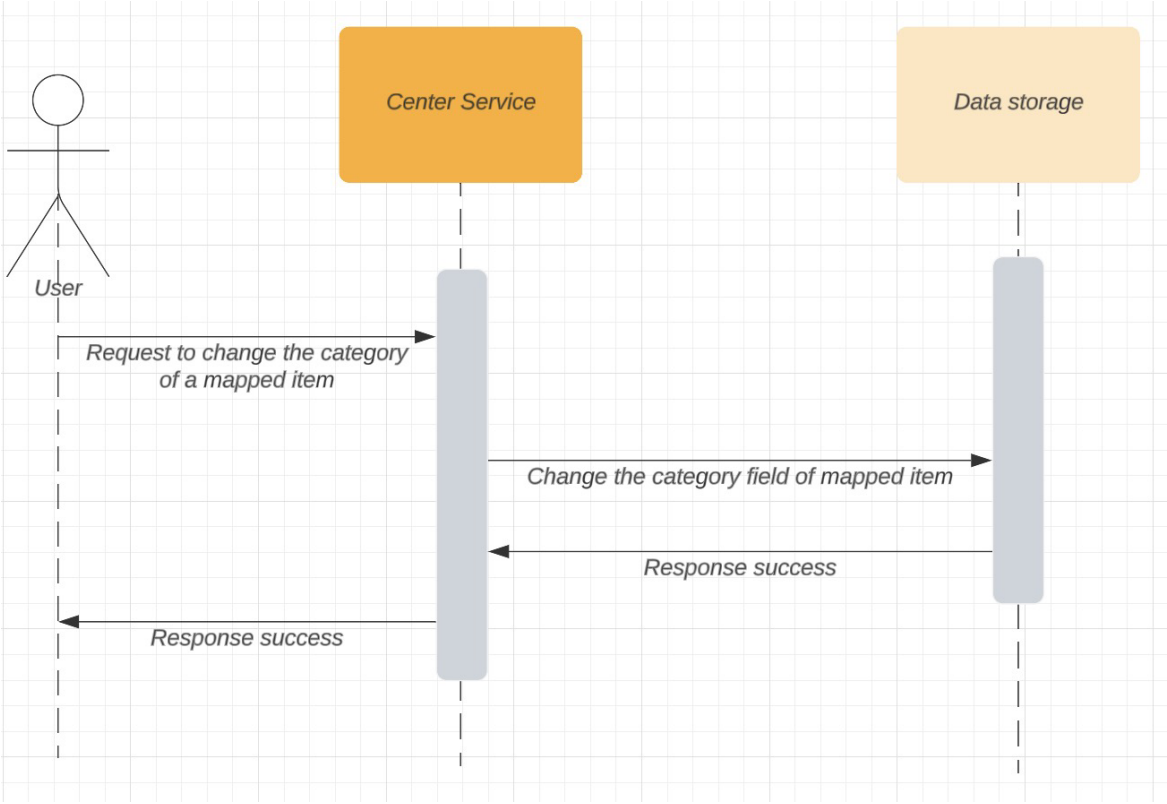
Version	Description	Date	Comment
2.0.0	Change the sequence diagram aligning with the latest system	30 May 2023	1. Update on mapping sequence with Ontoserver service 2. Update on system rollback sequence with new logic. However, this function is not yet implemented due to time limitation.
1.0.0	Basic sequence diagram	30 Apr 2023	

Version 2.0.0

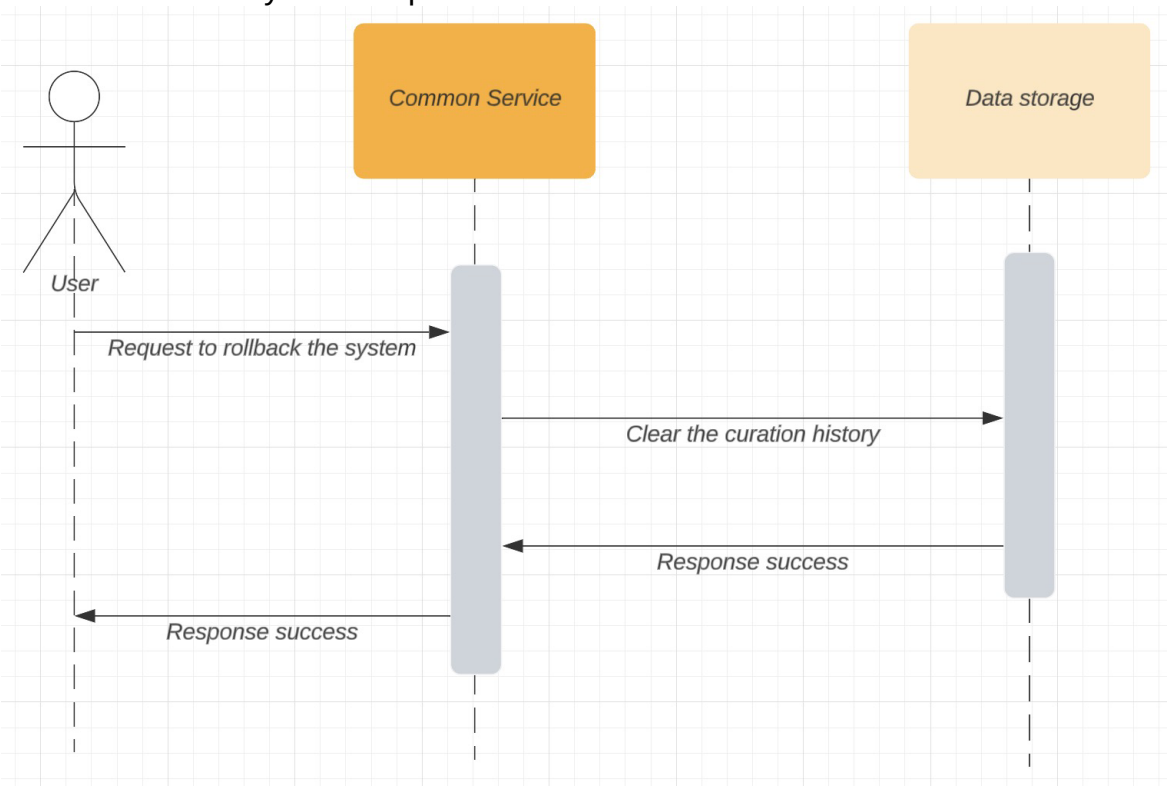
Mapping Sequence



Curate Sequence

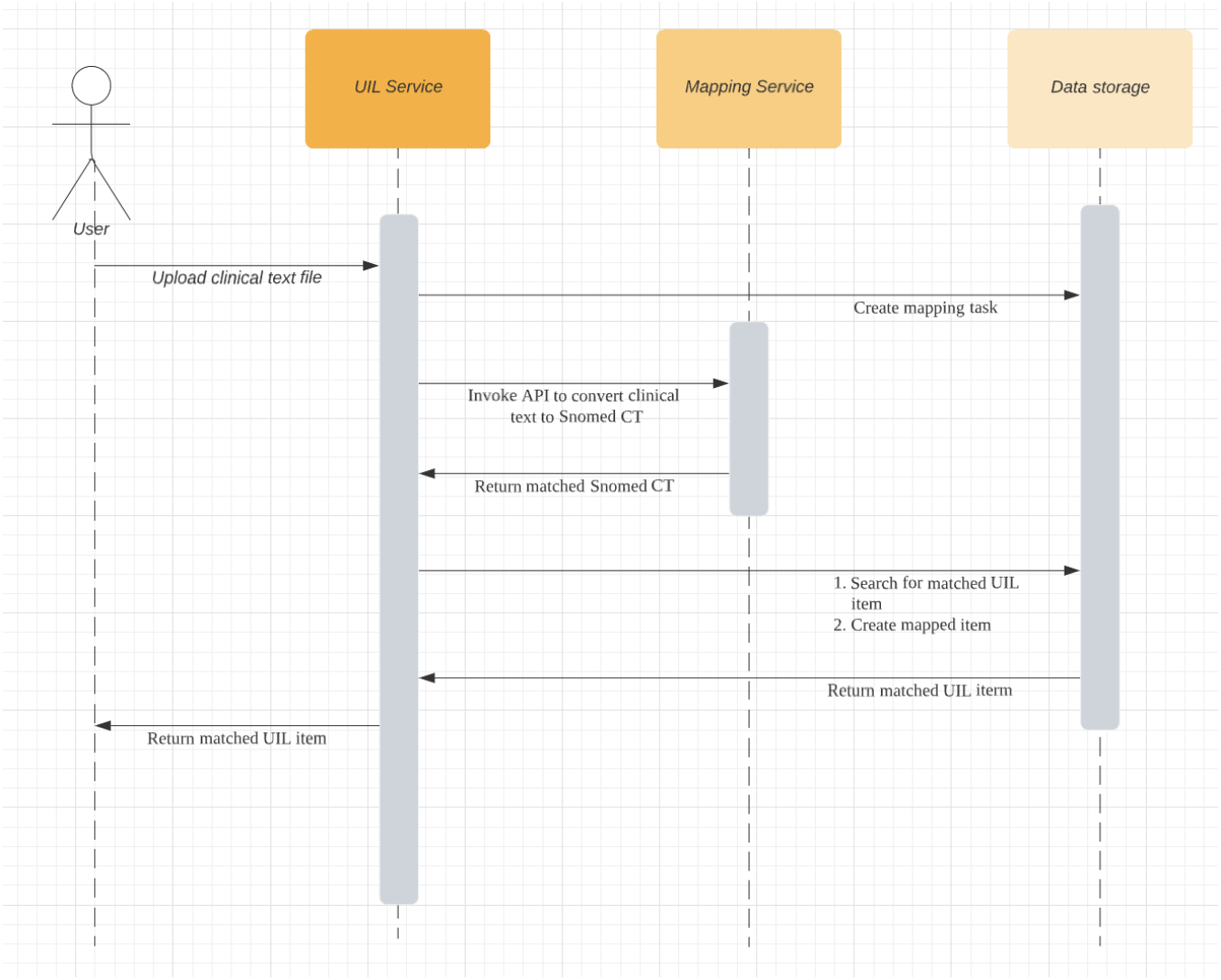


Rollback/Default System Sequence

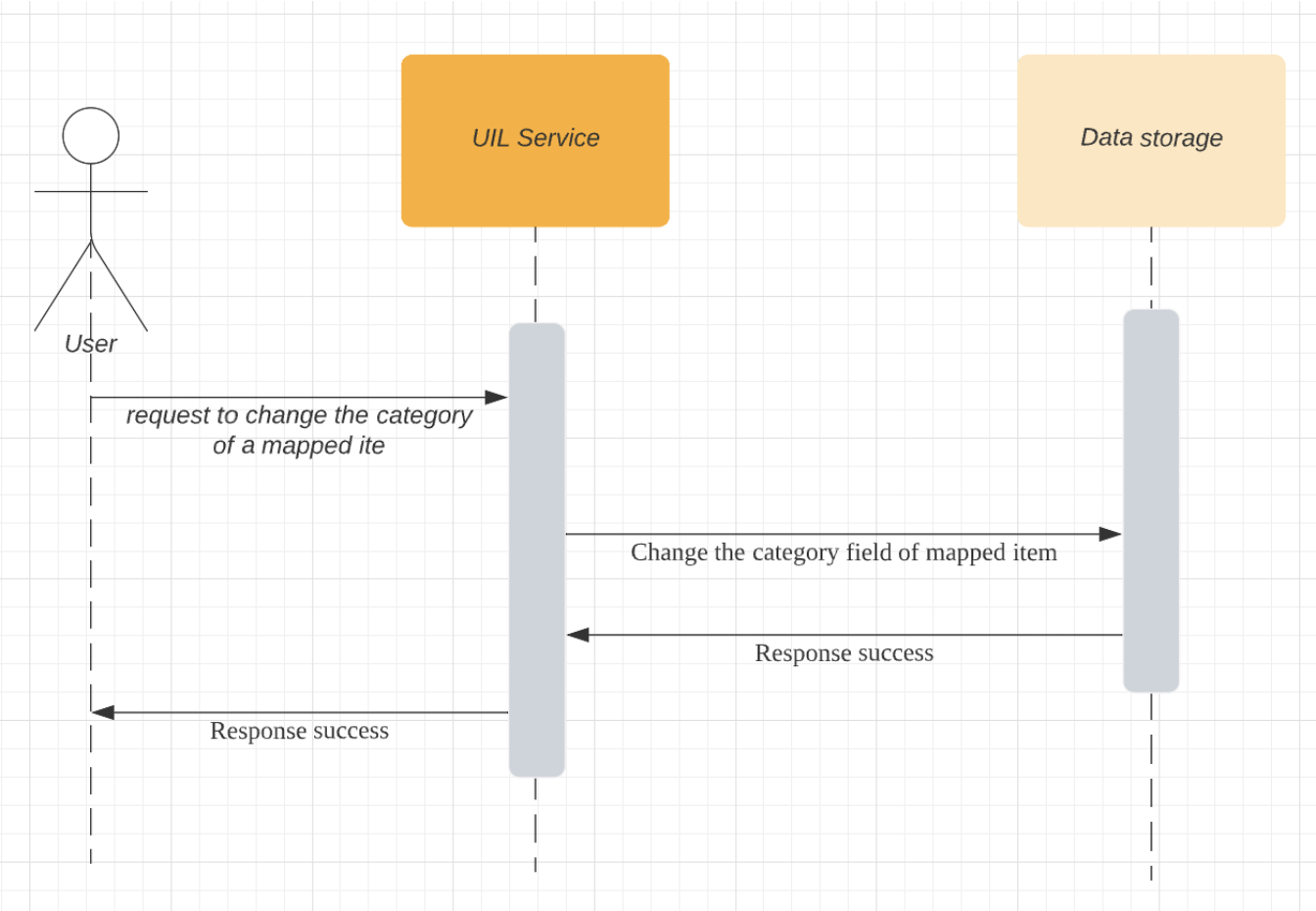


Version 1.0.0

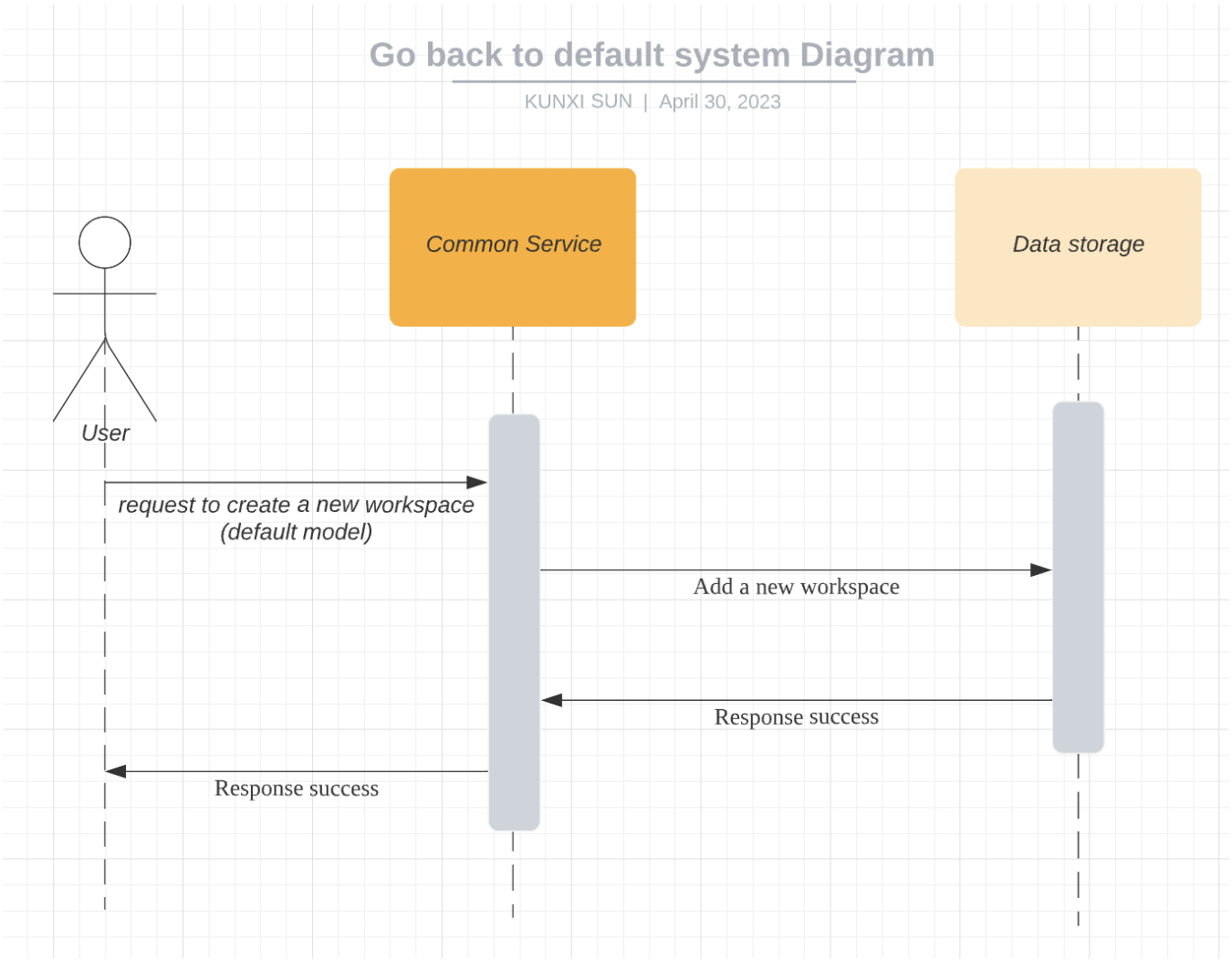
Mapping Sequence



Curate Sequence



Rollback/Default System Sequence


Go back to default system Diagram
KUNXI SUN | April 30, 2023
Common Service
Data storage
User
request to create a new workspace
(default model)
Add a new workspace
Response success
Response success

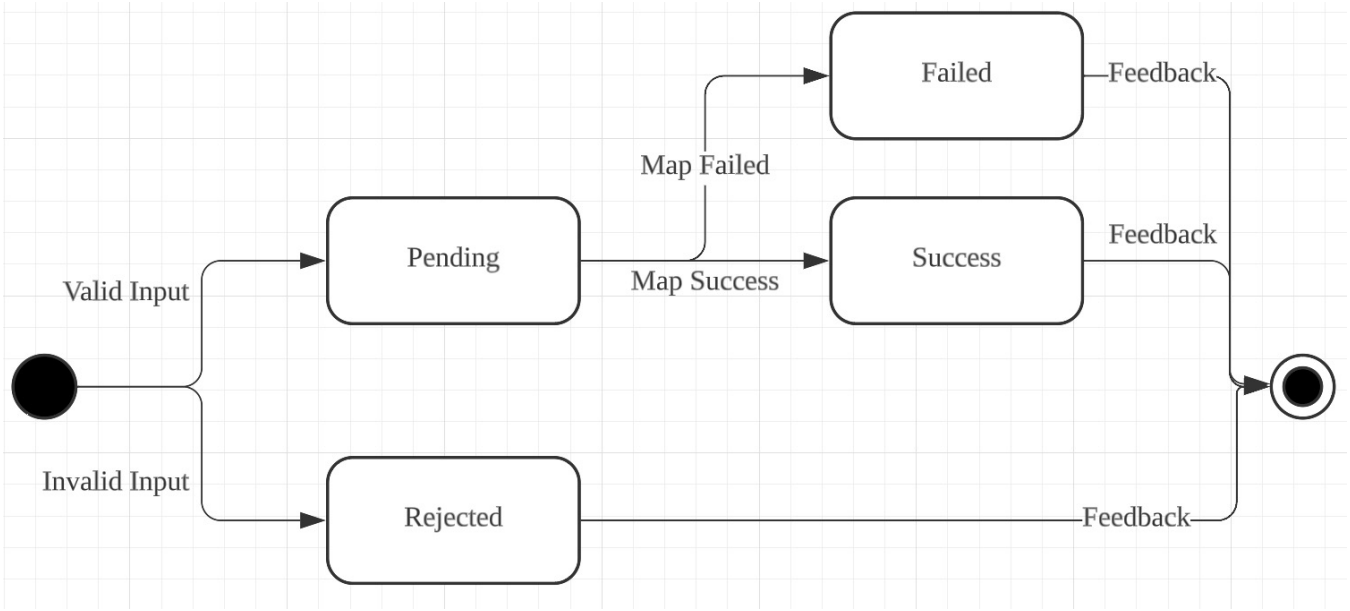
State Diagram

Version	Description	Date
1.0.0	1. Map Task State Diagram	April 24, 2023

Version 1.0.0

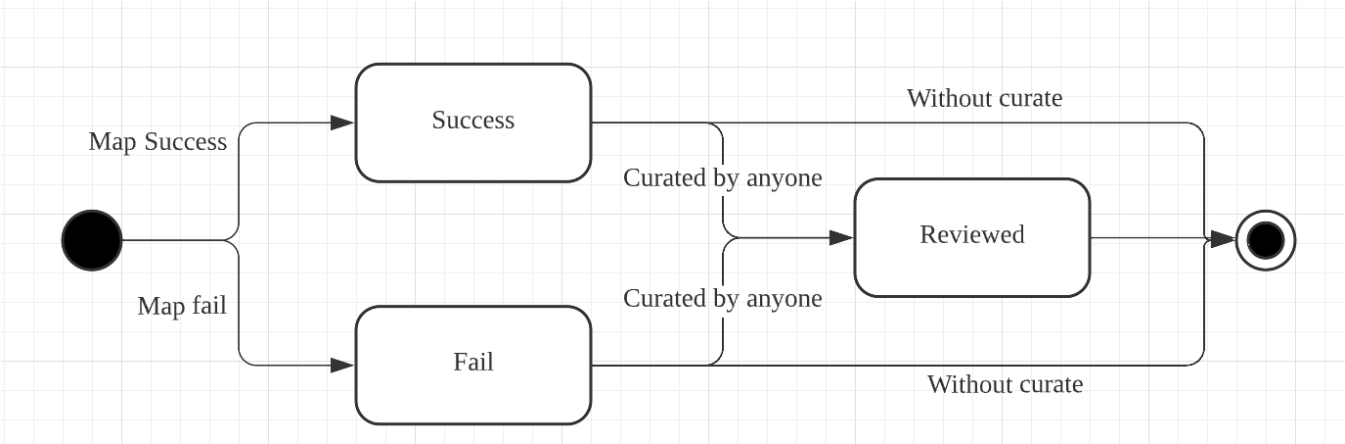
MapTask Status

Status Name	Status Number	Description
Pending	0	Waiting for task to finish
Success	1	Task success
Rejected	2	Task rejected due to invalid input or other reasons which cause the task did not start
Failed	3	ask failed during mapping for any reasons



Mapped Item Status

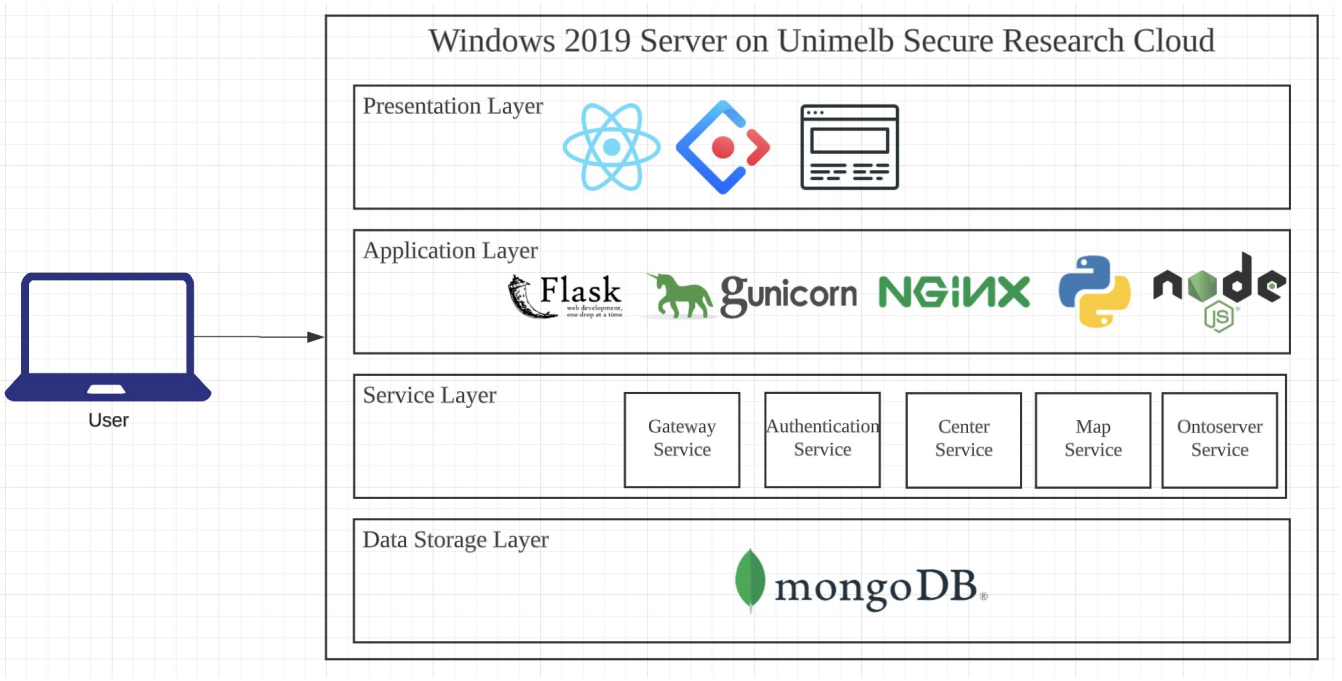
Status Name	Status Number	Description
Fail	0	Mapping system failed to map this raw text
Success	1	Mapping system map this item successfully
Reviewed	2	This raw text has been curated by user


Success
Without curate
Map Success
Curated by anyone
Reviewed
Map fail
Curated by anyone
Fail
Without curate

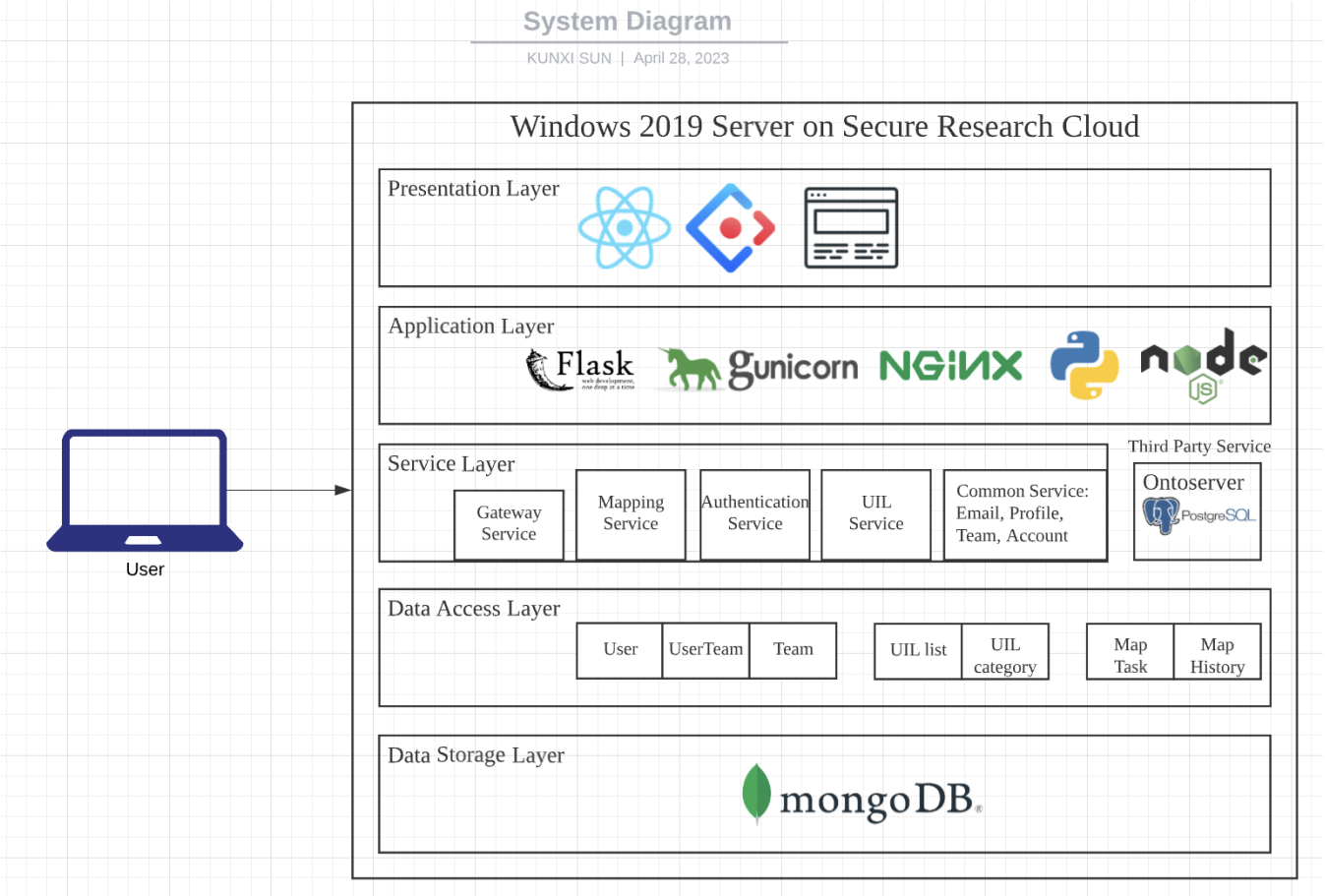
System Diagram

Version	Description	Date
3.0.0	1. Align with latest services	30 May 2023
2.1.0	1. Add UIL service	28 Apr 2023
2.0.0	1. Change the system diagram into a standard layers structure	24 Apr 2023
1.0.0	1. Basic system diagram with moduls	23 Mar 2023

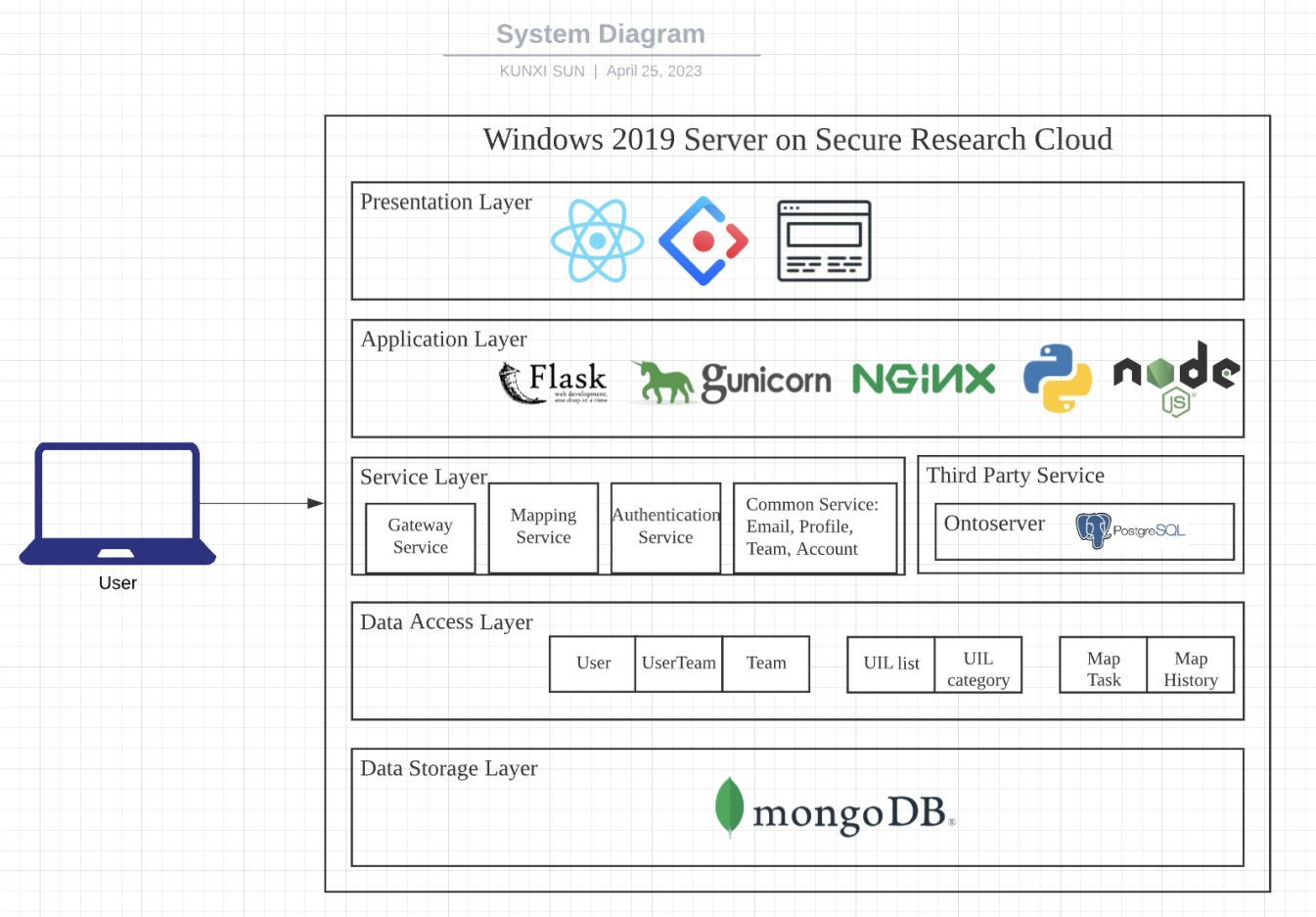
Version 3.0.0



Version 2.1.0


System Diagram
KUNXI SUN | April 28, 2023
Windows 2019 Server on Secure Research Cloud
Presentation Layer
Application Layer
Flask
gunicorn
NGINX
node
Service Layer
Gateway Service
Mapping Service
Authentication Service
UIL Service
Common Service: Email, Profile, Team, Account
Third Party Service
Ontoserver
PostgreSQL
Data Access Layer
User
UserTeam
Team
UIL list
UIL category
Map Task
Map History
Data Storage Layer
mongoDB.
User

Version 2.0.0


System Diagram
KUNXI SUN | April 25, 2023
Windows 2019 Server on Secure Research Cloud
Presentation Layer
Application Layer
Flask
gunicorn
NGINX
node
Service Layer
Gateway Service
Mapping Service
Authentication Service
Common Service: Email, Profile, Team, Account
Third Party Service
Ontoserver
PostgreSQL
Data Access Layer
User
UserTeam
Team
UIL list
UIL category
Map Task
Map History
Data Storage Layer
mongoDB.
User

Version 1.0.0

System Acrchitecture Diagram