1. COMP90082-2023-SM1-Digital-Health . . . 3
1.1 Requirements . . . 6
1.1.1 Background . . . 7
1.1.2 Project Scope . . . 10
1.1.3 Functional Requirements . . . 11
1.1.4 Non-Functional Requirements . . . 13
1.1.5 Motivational Model . . . 16
1.1.6 Personas . . . 18
1.1.7 User Stories and Acceptance Criteria . . . 26
1.1.8 Use Cases . . . 35
1.1.9 Product Backlog . . . 44
1.1.10 Milestones . . . 48
1.2 Timeline . . . 49
1.2.1 Sprint 1 - Design Sprint . . . 50
1.2.1.1 Sprint 1 - Plan . . . 51
1.2.1.2 Sprint 1 - Review . . . 52
1.2.2 Sprint 2 - Development . . . 55
1.2.2.1 Sprint 2 - Plan . . . 56
1.2.2.2 Sprint 2 - Backlog . . . 57
1.2.2.3 Sprint 2 - Review . . . 60
1.2.3 Sprint 3 - Development . . . 62
1.2.3.1 Sprint 3 - Plan . . . 63
1.2.3.2 Sprint 3 - Backlog . . . 64
1.2.3.3 Sprint 3 - Review . . . 69
1.2.4 Sprint 4 - Product . . . 71
1.2.4.1 Sprint 4 - Plan . . . 72
1.3 Handover . . . 73
1.3.1 Development Overview . . . 74
1.3.2 Release Notes . . . 78
1.3.3 User Guide . . . 84
1.4 Development . . . 90
1.4.1 APIs . . . 91
1.4.1.1 Auth Service . . . 93
1.4.1.2 Center Service . . . 100
1.4.1.3 Gateway Service . . . 115
1.4.1.4 Map Service . . . 116
1.4.2 Digital Prototype . . . 119
1.5 Architecture . . . 125
1.5.1 Technique Detail . . . 126
1.5.2 4+1 Architecture Models . . . 129
1.5.2.1 Container Diagram . . . 130
1.5.2.2 Database Model . . . 131
1.5.2.3 Sequence Diagrams . . . 133
1.5.2.4 State Diagram . . . 138
1.5.2.5 System Diagram . . . 140
1.6 Quality . . . 144
1.6.1 Test Plan . . . 145
1.6.1.1 Behaviour Driven Development . . . 147
1.6.1.2 Test Driven Development . . . 151
1.6.2 Test Cases . . . 152
1.6.2.1 TC0001 - Authentication & Authorization . . . 153
1.6.2.2 TC0002 - Upload & Download files . . . 154
1.6.2.3 TC0003 - Mapping Clinical Raw Text . . . 155
1.6.2.4 TC0004 - Visualise Mapping Result . . . 156
1.6.2.5 TC0005 - Curate Mapping Result . . . 157
1.6.2.6 TC0006 - Rollback to Default System . . . 158
1.6.2.7 TC0007 - Team Management . . . 159
1.6.2.8 TC0008 - Security . . . 160
1.6.3 Test Reports . . . 161
1.6.3.1 2023-04-28 Functional Test . . . 162
1.6.3.2 2023-05-25 Functional Test . . . 163
1.6.3.3 Performance Test Report . . . 164
1.6.3.4 Acceptance Test . . . 166
1.6.4 Code Reviews . . . 168
1.6.4.1 2023-04-27 Sprint 2 peer-to-peer Code Review Report - BackEnd . . . 169
1.6.4.2 2023-04-27 Sprint 2 peer-to-peer Code Review Report - FrontEnd . . . 170
1.6.4.3 2023-05-31 Sprint 3 ChatGPT-assisted Code Review Report - BackEnd . . . 171
1.6.4.4 2023-05-31 Sprint 3 ChatGPT-assisted Code Review Report - FrontEnd . . . 176
1.6.5 Issue Tracking . . . 180
1.6.5.1 ISSUE00010: Login incorrect error message . . . 181
1.6.5.2 ISSUE00012: MedCAT: Fail to load the model because of ValidationError . . . 182
1.6.5.3 ISSUE00015: Github Actions: checkout repository takes too long and sometimes fail #15 . . . 183
1.6.5.4 ISSUE00021: MedCAT: Cannot use cat.multi_processing to map a file of raw texts . . . 184
1.6.5.5 ISSUE00022: Dashboard nickname is not updated in real time . . . 185
1.6.5.6 ISSUE00023: Forced to refresh page to login in successfully after cookie removed . . . 186
1.6.5.7 ISSUE00025: Avatar is obscured when entering edit mode . . . 187
1.6.5.8 ISSUE00026: The numbers in the chart are inconsistent with the Numbers on the card . . . 188
1.6.5.9 ISSUE00027: New users cannot be invited if this user does not have an account in login status. . . . 189
1.6.5.10 ISSUE00028: New member register showing missing cookie . . . 190

1.6.5.11 ISSUE00029: Cannot display the board with deleted member's task . . . 191
1.6.6 CI/CD . . . 192
1.6.7 User Experience Evaluation . . . 193
1.6.8 Pair programming . . . 198
1.7 Decision . . . 199
1.7.1 Combine Ontoserver with MedCAT . . . 200
1.7.2 MedCAT - Use Single Processing Instead of Multiprocessing . . . 202
1.7.3 Github Branch Workflow . . . 203
1.7.4 No Continuous Deployment . . . 204
1.7.5 Use Apifox API Test to Replace Automatic Unit Test . . . 205
1.7.6 Use Apifox Instead of Postman . . . 206
1.7.7 Use Github Actions Instead of Ansible Script . . . 207
1.7.8 Role Assignment . . . 208
1.7.9 Technique Stack . . . 209
1.8 Others . . . 210
1.8.1 Demonstration . . . 211
1.8.2 Presentation . . . 212
1.8.3 Tool Uses . . . 213
1.8.4 Server Configuration . . . 220
1.8.5 Code Review Checklist . . . 221
1.8.6 Mapping Tools Survey . . . 222
1.8.7 Docker command . . . 223

COMP90082-2023-SM1-Digital-Health

Project Description

Our team is working on the development of a cutting-edge platform designed to enhance the analysis of clinical documentation in the digital health area. The primary objective of this platform is to determine whether medications prescribed to patients are appropriate by normalizing free-text clinical notes and mapping them to canonical clinical terms.

The platform's primary function is to simplify the process of associating free-text descriptions, which generally explain the reasoning behind prescribing specific medications, onto a Universal Indication List (UIL), which serves as a subset of the broader standardized knowledge base of clinical terms called SNOMED CT.

The platform features the integration of a human-in-the-loop system, which allows for manual review and correction of the mapping results. This feedback will be used to continuously enhance the platform's accuracy and performance.

This curation feature will further streamline the mapping process, ensuring that the most relevant and commonly used clinical terms are easily accessible for healthcare professionals and researchers.

Overview

Agile Methodology	Scrum
Status	Sprint 4 (Product)
Team Location	Melbourne, Australia
Team representative's Email	doncd@student.unimelb.edu.au

Important Pages

Github	Backlog	Timeline
Figma	Architecture	Meetings
Trello (Invitation link)	Development	Demonstration

Motivation, mission and vision

Due to the fact that extracting information from free-text clinical notes could be very difficult, our platform could be used to help medical researchers

- map the short texts into a certain category and view them
- manually curate the incorrect or unrecognised mapping result
- feed corrections back to improve the system
- download the mapping results

We hope that our final product could be officially deployed and used by our client to help real-world medical researchers.

Client Goals

- Developing a platform that will assess if a medication has been appropriately prescribed to a patient.
- Set up and customise a platform that will allow a human-in-the-loop to manually review the results of the mapping, make corrections, and feed these back to re-train the system.

Client

Daniel Capurro

Email: dcapurro@unimelb.edu.au

Supervisor

Mauro Mello Jr

Email: mauro.mellojr@unimelb.edu.au

Daniel is a Medical Doctor, trained in Internal Medicine, and hold a PhD in Biomedical and Health Informatics from the University of Washington in Seattle. He is the Deputy Director of the Centre for Digital Transformation of health where he co-leads the Digital health Validitron (a pipeline to validate digital health innovations in a way similar to what happens with drugs and vaccines) and the Data Science stream.

Extensive experience in Information Systems and Information Technology (since 1978) and tertiary education (since 1982). The fields are involed with Business and management, Consulting activities, Project management activities and IT and systems development methodologies.

Meet the team

Kunxi (Quincy) Sun

Email: kunxis@student.unimelb.edu.au

Role: Product Owner

Responsibility:

- Ensures the team delivers the most value to Daniel
- Backend, Design, Devops

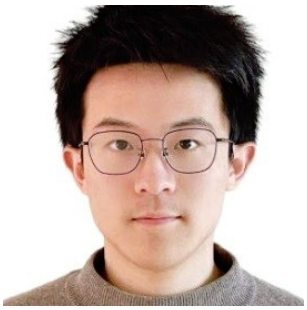
Chenyang (Peter) Dong

Email: doncd@student.unimelb.edu.au

Role: Scrum Master

Responsibility:

- Responsible for ensuring that the Scrum framework is followed
- Frontend, Backend



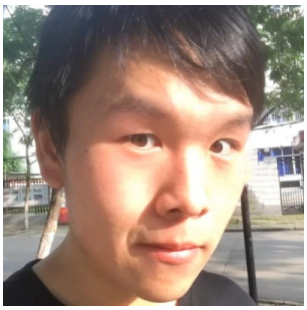
Hanyi (Henry) Gao

Email: hanyig1@student.unimelb.edu.au

Role: Development Team Member

Responsibility:

- Implements the system
- Frontend, Testing, Design



Yulai (Ricardo) Luo

Email: yulail1@student.unimelb.edu.au

Role: Development Team Member

Responsibility:

- Implements the system
- Frontend, Backend, UX

Yue (Molly) Fei

Email: yffei@student.unimelb.edu.au

Role: Development Team Member

Responsibility:

- Implements the system
- Frontend, Backend, UX

Recent space activity

Chenyang Dong

Handover updated a minute ago • view change

File Lists updated 3 minutes ago • view change

Hanyi Gao

Release Notes updated 9 minutes ago • view change

Space contributors

- Chenyang Dong (a minute ago)
- Hanyi Gao (9 minutes ago)
- Yue Fei (14 minutes ago)
- KUNXI SUN (9 hours ago)
- Ricardo Luo (11 hours ago)
- ...

Yue Fei

Release Notes updated 14 minutes ago • view change

Development Overview updated 17 minutes ago • view change

Requirements

Table of Content

Title	Creator	Modified
Background	Chenyang Dong	01 Jun, 2023
Product Backlog	Chenyang Dong	02 May, 2023
User Stories and Acceptance Criteria	KUNXI SUN	30 Apr, 2023
Use Cases	Chenyang Dong	29 Apr, 2023
Personas	Chenyang Dong	29 Apr, 2023
Motivational Model	Chenyang Dong	29 Apr, 2023
Non-Functional Requirements	Chenyang Dong	29 Apr, 2023
Milestones	Ricardo Luo	28 Apr, 2023
Functional Requirements	Chenyang Dong	28 Apr, 2023
Project Scope	KUNXI SUN	28 Apr, 2023
Digital Prototype	Chenyang Dong	02 Apr, 2023

Background

1 Clinical Documentation Analysis

1.1 Importance of Clinical Documentation

Clinical documentation is a critical component of healthcare, as it serves as a record of a patient's medical history, diagnosis, treatment, and care. Accurate and comprehensive documentation enables healthcare providers to make informed decisions and facilitate effective communication among team members. Additionally, clinical documentation plays a crucial role in various areas such as medical research, quality improvement initiatives, and billing or reimbursement processes.

1.2 Challenges in Extracting Information from Free-Text Clinical Notes

Free-text clinical notes are widely used in medical practice, as they allow clinicians to describe complex cases efficiently. However, extracting structured and actionable information from these notes is a challenging task. Natural language processing (NLP) techniques can be used to analyze and process free-text clinical notes. However, variations in terminology, abbreviations, and linguistic structures can lead to inaccuracies and inconsistencies when mapping the extracted information to standardized medical terms.

2 SNOMED CT

2.1 A Brief Overview of SNOMED CT

SNOMED CT (Systematized Nomenclature of Medicine - Clinical Terms) is a comprehensive, multilingual clinical terminology system designed for the global healthcare community. It enables the consistent representation of clinical information across different systems and facilitates interoperability between healthcare providers. SNOMED CT includes over 350,000 clinical concepts, each with a unique identifier, and supports mapping to other coding systems like ICD-10.

2.2 Benefits of Mapping Clinical Terms to SNOMED CT

Mapping clinical terms to SNOMED CT offers several benefits:

1. Enhanced data consistency and accuracy, as it enables a standardized representation of clinical concepts.
2. Improved communication and understanding among healthcare professionals.
3. Facilitation of medical research and data analytics by providing a common language for data sharing and comparison.
4. Streamlined billing and reimbursement processes through standardized coding.

2.3 SNOMED CT Latest Release

Use this link to download the last six SNOMED CT-AU monthly releases in their native RF2 distribution format.

3 Universal Indication List (UIL)

3.1 A Brief Overview of UIL:

The Universal Indications List (UIL) is a SNOMED CT coded list that standardizes indications for antimicrobial drug use and infections.

3.2 Development of UIL:

The UIL was developed and is maintained by the National Centre for Antimicrobial Stewardship - Guidance Group (Melbourne Health, Australia) in line with the Implementation Plan for Australia's National Antimicrobial Resistance Strategy.

3.3 Applications of UIL:

The UIL can be used in various clinical systems, including electronic medical records (EMR), electronic medication management (EMM) systems, auditing tools (e.g. NAPS), and antimicrobial stewardship approval systems (e.g. Guidance).

3.4 Customization of UIL:

While primarily designed for the Australian healthcare setting, the UIL can be customized for use in other countries.

3.5 Benefits of UIL:

The UIL promotes interoperability, seamless data flow between clinical systems, and standardization of local and national reporting. This facilitates audits, clinical research, benchmarking, and surveillance.

4 Ontoserver

4.1 A Brief Overview of Ontoserver

Ontoserver is a terminology server developed by CSIRO that provides a wide range of features to support the use of clinical terminologies, such as SNOMED CT, in healthcare applications. It allows for the storage, management, and querying of clinical terms and supports mapping between different terminologies. Ontoserver can be deployed locally and tuned to specific data sets, offering flexibility and adaptability to various healthcare contexts.

4.2 Features and Capabilities

Ontoserver provides several features and capabilities, including:

1. Terminology storage and management: Efficiently store and manage large terminologies such as SNOMED CT.
2. Mapping: Perform mapping between different terminologies and code systems.
3. Querying: Retrieve and search clinical terms using advanced query capabilities.
4. Customization: Customize the server to suit specific healthcare applications and requirements.
5. FHIR Standard and Syndication: Ontoserver implements the FHIR (Fast Healthcare Interoperability Resources) standard, ensuring seamless integration with other FHIR-compliant clients and systems.

4.3 Licensing

- Within Australia, email help@digitalhealth.gov.au to request a (free) Ontoserver licence. ADHA will then arrange authorisation for your quay.io account
- Elsewhere, email ontoserver-support@csiro.au to discuss licensing terms (both evaluation and production licences are available for single and multiple instances with no limit on the number of users). Once the licence is established, CSIRO will register your quay.io account name to enable access to their repository
- Note: The license is provided for either individual or organisation. University of Melbourne holds an active license, if you would like to use the license from Unimelb, please contact who is responsible for the Ontoserver license at Unimelb.
- To use the ontoserver image:
    1. (Once you have a license) Ensure your dockerhub account has been registered with ontoserver-support@csiro.au
    2. Log in to docker docker login
    3. Create a docker-compose.yml file for ontoserver (example provided in the repository), making sure to supply your syndication credentials (for information on how to retrieve these credentials, please consult the Ontoserver documentation).
    4. Start the compose docker-compose up -d
    5. To get the latest version of SNOMED CT-AU into the Ontoserver index, run the following: docker exec ontoserver /index.sh -s sctau Or, to get a specific version of SNOMED CT-AU into the Ontoserver index, run the following: docker exec ontoserver /index.sh -s sctau -v 20230131

4.4 Deployment and Customization

Ontoserver can be deployed locally, providing the flexibility to tailor the server to a specific use case. It can be deployed using Docker, enabling easy integration with existing virtual machines and making it a reusable component in a text analytics pipeline. Customization options include tuning the server to a specific dataset, enhancing performance, and adapting the server to support additional terminologies or features.

For technical documentation: https://ontoserver.csiro.au/site/technical-documentation/ontoserver-technical-documentation/

5 MedCAT

5.1 A Brief Overview of MedCAT

MedCAT (Medical Concept Annotation Tool) is an open-source NLP tool developed by the University of Cambridge that can be used to extract a wide range of medical concepts, such as diseases, symptoms, procedures, and medications, from clinical notes and other healthcare-related documents.

Paper on arXiv. Official Docs here.

5.2 How MedCAT Works:

MedCAT uses a combination of rule-based and machine learning-based approaches to identify and classify medical concepts. It first applies rule-based techniques to identify known medical terms, then uses machine learning to identify and classify new terms based on their context.

5.3 Downloading the Model

To download any of public models, please follow this link and sign into your NIH profile / UMLS license. You will then be redirected to the MedCAT model download form. Please complete this form and you will be provided a download link.

6 Security and Privacy in Healthcare Data

6.1 Data Protection and Compliance

When working with sensitive healthcare data, it is crucial to ensure that the platform adheres to strict data protection and privacy regulations such as HIPAA (Health Insurance Portability and Accountability Act) in the United States or GDPR (General Data Protection Regulation) in the European Union. These regulations are designed to protect patient's personal and medical information from unauthorized access and misuse.

6.2 Secure Data Storage and Transmission

To maintain the integrity and confidentiality of healthcare data, it is essential to implement robust security measures for data storage and transmission. This includes encrypting data at rest and in transit, using secure authentication methods, and regularly monitoring and auditing system activities. By incorporating best practices for data security, the platform can help protect sensitive information while still enabling valuable insights and analysis.

Project Scope

1 SNOMED CT knowledge base

- Understand the hierarchical structure, concepts, relationships, and terminology used in SNOMED CT.
- Learn how to query the SNOMED CT knowledge base and navigate its structure.
- Keep up to date with changes, updates, and new releases of the SNOMED CT knowledge base.

2 Ontoserver integration

- Research and select an appropriate Ontoserver implementation that fits your requirements.
- Set up and configure Ontoserver, including necessary hardware and software components.
- Understand and utilise the Ontoserver API for querying and integrating SNOMED CT data into your system.

3 Symptom to UIL mapping

- Develop or use existing algorithms and tools to accurately match symptom data to UIL terms.
- Handle possible ambiguities or variations in symptom descriptions.
- Develop a process for extracting and presenting disease names based on matched SNOMED CT terms.
- Allow user to download the mapping result.

4 Result curating and system improving

- Establish metrics for evaluating the mapping results.
- Conduct testing with real-world data and compare results against a gold standard or expert-reviewed data.
- Allow user to accurate the mapping result to continuously optimise the mapping tool to improve performance and address any identified issues.

5 User interface and experience

- Design a user-friendly web interface for entering symptom data and displaying terms of UIL

6 Documentation and training

- Create comprehensive documentation outlining the system's architecture, features, and API usage.
- Develop user guides, tutorials, or other training materials to assist users in understanding and utilising the system effectively.

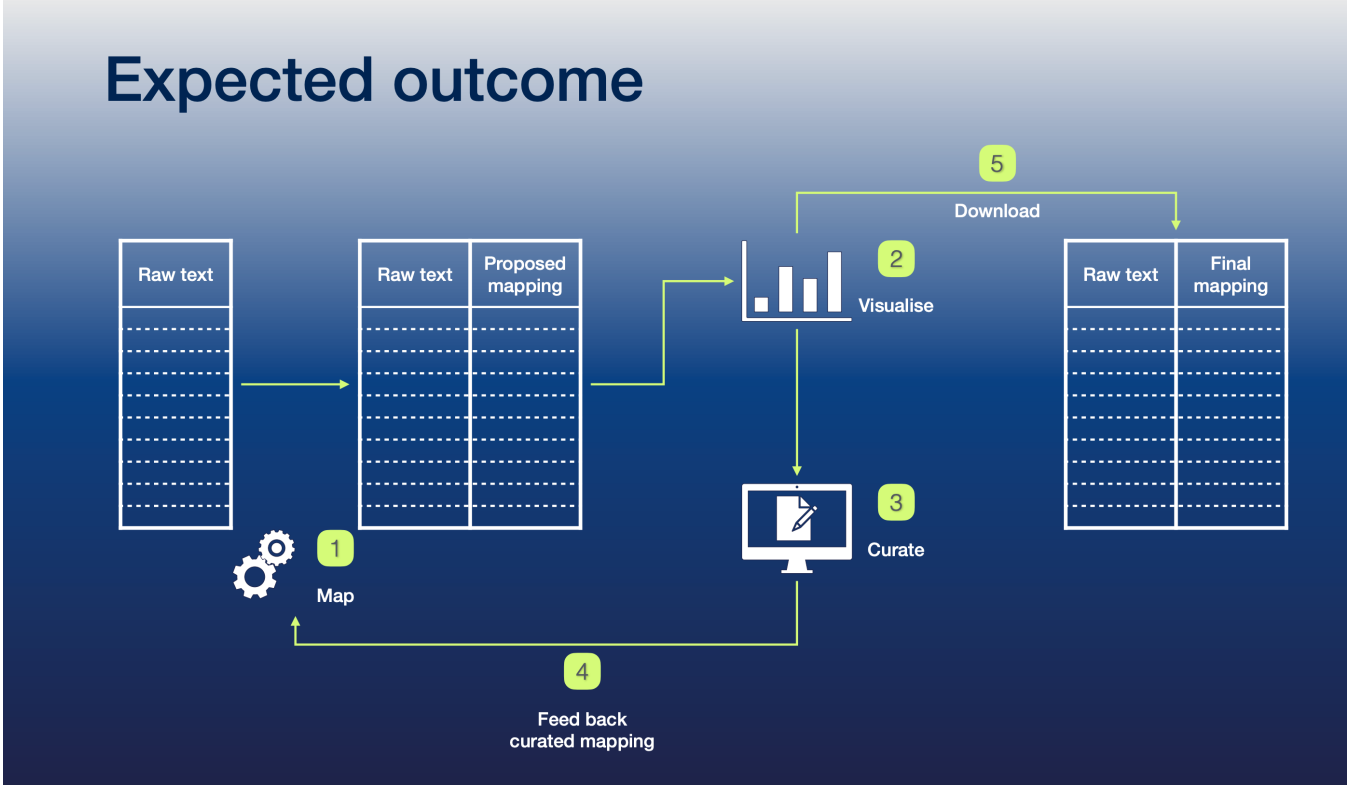
7 Project maintenance and updates

- Update the system as necessary to accommodate new versions or updates of the UIL and SNOMED CT knowledge base.

Functional Requirements

1 Expected Outcome

- The expected outcome of the system with five core functions are presented by client



2 Text Input and Processing

- The system must accept free-text clinical text input from medical researchers.
- The system must process and analyze the input text or file to identify relevant terms and phrases.

3 Mapping Terms to Categories

- The system must map identified terms to one or multiple categories based on a universal indication list.
- The system must allow users to review and adjust the selected category.

4 Integration with Mapping Tool

- The system must integrate with a mapping tool such as Ontoserver.
- The system must be able to send input text to the mapping tool and receive mapped categories and confidence levels.

5 Downloading Mapping Results

- The system must provide an option for users to download the mapping results.
- The system must generate a report containing history mapping and mapped categories for future reference.

6 Visualization

- The system must provide a user-friendly interface for curating.
- The system must display the mapping performance.

- The system must display the performance history of mapping.

7 Versioning

- The system must provide a user-friendly interface for the admin user to select the previous version to restore.
- The system should allow the admin user to view the restored version and the terms that were restored, to ensure that the correct version was rolled back.
- The system should also update the mapping history to reflect the rollback and the restored version.

8 User Authentication and Authorization

- The system must provide user authentication to ensure secure access to the mapping tool.
- The system must identify the user who curates the mapping results.
- The system may support role-based access control to manage user permissions and access to specific features.

Non-Functional Requirements

Version	Description	Date
2.0.0	1. Add more detail into security concern 2. Add QoS non functional requirements	25 Apr 2023
1.0.0	1. Basic non-functional requirements	23 Mar 2023

Version 2.0.0

1 QoS Non-Functional Requirements:

1.1 Responsiveness

- The platform should respond to user interactions within 2 seconds under normal load conditions, ensuring a smooth user experience.
- For computationally intensive tasks like mapping tasks, the platform should provide progress updates and complete tasks within a reasonable time frame.
- The platform should have a monitoring system to track performance metrics and detect any potential issues early.

1.2 Accuracy

- The system should incorporate advanced algorithms to increase the accuracy of mapping input data to categories and curating them.
- User curation should lead to further improvement in accuracy, with a target of 99% accuracy after multiple rounds of curation.

1.3 Scalability

- The system should be designed to handle a growing number of users, teams, and mapping tasks without significant performance degradation.
- The infrastructure should be easily expandable to accommodate the increasing workloads and storage needs.
- The platform should utilize cloud-based solutions to ensure flexibility in scaling up or down as needed.

1.4 Usability

- The user interface should be designed with a focus on ease of use, minimizing the learning curve for healthcare professionals and researchers.
- The system should provide clear instructions, tooltips, and context-sensitive help to guide users through complex tasks.
- The platform should offer customization options to accommodate the preferences and requirements of different user groups.

1.5 Security

As we embark on the development of our medication assessment system, it is essential to recognize the importance of addressing security concerns from the project's inception. Ensuring the confidentiality, integrity, and availability of sensitive medical data is critical to the success of our platform and the trust of our users. By thoroughly examining potential security risks and implementing appropriate countermeasures, we can establish a robust and secure foundation for our system, adhering to industry standards and regulatory requirements.

1.5.1 Authentication and authorisation

- Ensure that only authorized users have access to the system and its features by implementing robust authentication mechanisms (e.g., strong passwords, multi-factor authentication).
- Implement role-based access control to grant appropriate permissions to different user roles (e.g., researchers, administrators).

1.5.2 Secure APIs

- Ensure that any APIs are secure and follow best practices for authentication, data privacy, and data protection.

1.5.3 Data Protection and Privacy

- Safeguard sensitive data of patients and medical professionals, such as medical histories, diagnoses, and prescription information, in compliance with relevant privacy regulations (e.g., HIPAA, GDPR).
- Follow the principle of data minimization by collecting only the data needed for the project.

1.5.4 Secure Software Development Practices

- Implement security best practices throughout the development lifecycle, including code reviews, security testing, and vulnerability assessments

1.5.5 Backup and disaster recovery

- Establish regular backups and a disaster recovery plan to minimize data loss and system downtime in the event of a security incident or system failure.

2 Compliance Requirements (Standards):

2.1 Data Privacy Regulations

- The platform must implement robust access controls, data encryption, and secure data storage practices to comply with HIPAA or GDPR requirements.
- Regular security audits and vulnerability assessments should be conducted to identify and address potential threats.
- The platform should have a robust backup and recovery strategy to protect against data loss or corruption.

3 Architectural Constraints (SOA Principles):

3.1 Modularity

- The system should be divided into distinct modules (e.g., user management, mapping tasks, curation) that can be developed, tested, and deployed independently.

3.2 Reusability

- Common functionalities (e.g., authentication, logging, data validation) should be implemented as shared services or libraries that can be used across the system.
- APIs should be created to facilitate secure access and communication between different components and services.

3.3 Interoperability:

- The system should use standard data formats (e.g., JSON, XML) and communication protocols (e.g., REST, GraphQL) to facilitate integration with external systems and data sources.
- The system should support various data formats, such as CSV, JSON, and XML, to facilitate data import and export.

4 Development Constraints (Process):

4.1 Agile Development

- The project should follow an Agile development process, such as Scrum or Kanban, to support iterative development, continuous improvement, and adaptability to changing requirements.
- Regular feedback from stakeholders and users should be incorporated into the development process to ensure the platform meets their needs and expectations.

4.2 Continuous Integration & Deployment

- The development process should include automated build, test, and deployment pipelines to ensure consistent quality and faster delivery of new features and improvements.
- Code should be frequently merged into the main branch to minimize integration issues.

4.3 Code Quality

- The development team should follow coding best practices and conventions to maintain high-quality code.
- Test-driven development should be adopted to ensure comprehensive test coverage and reduce the likelihood of defects.
- Regular code reviews should be conducted to identify and address potential issues and improve the overall code quality.

4.4 Documentation

- The development team should provide clear and comprehensive documentation for the system, including technical specifications, user guides, and API documentation.
- Documentation should be regularly updated to reflect changes and improvements in the platform, ensuring that it remains a reliable source of information for stakeholders and users.

4.5 Version Control

- The development team should use a version control system (e.g., Git) to track code changes, manage branching and merging, and maintain a history of project development.

Version 1.0.0

Usability:

- The system must provide an intuitive and user-friendly interface for medical researchers.
- The system must include clear documentation, tooltips, and help resources for users to understand its mapping features and functionalities.

Performance:

- The system must process and map clinical text within a reasonable time frame to support efficient research workflows.
- The system should maintain its performance even when dealing with complex or large input data sets from diverse clinical sources in the future.

Scalability:

- The system should accommodate an increasing number of users and data without degrading performance.
- The system may be designed with a modular architecture to support future enhancements, such as integration with additional mapping tools or clinical databases.

Availability:

- The system must be highly available to minimize downtime and support user access at all times.
- The system must include monitoring and alerting mechanisms to detect and address potential issues.

Reliability:

- The system must ensure data integrity and consistency in processing and storing clinical text and mapping results.
- The system must provide error handling and recovery mechanisms to minimize disruptions in case of failures.

Security:

- The system must follow industry-standard security practices to protect sensitive data and user information.
- The system must undergo regular security audits and vulnerability assessments to identify and address potential risks.

Maintainability:

- The system must be easy to maintain, with well-organized code, documentation, and version control.
- The system must provide mechanisms for easy troubleshooting, bug fixes, and feature updates.

Compliance:

- The system must comply with all relevant industry standards, guidelines, and data privacy and security regulations.
- The system must maintain proper records and documentation to demonstrate compliance with regulatory requirements.

Motivational Model

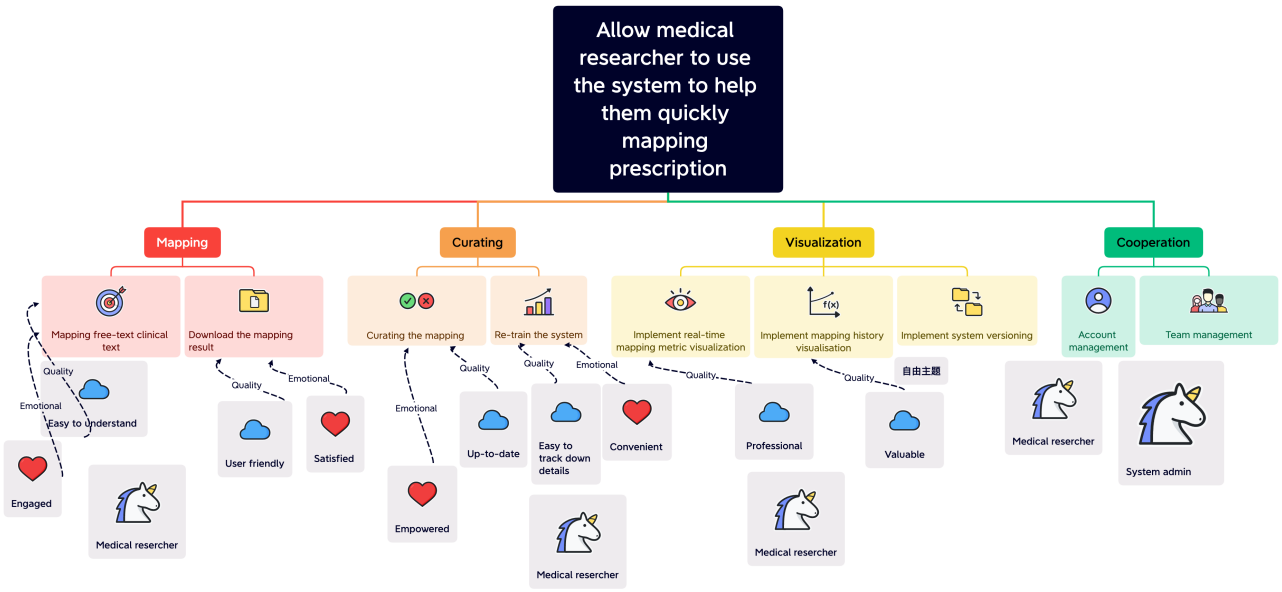
Version	Description	Date
2.0.0	1. Fix some tempo in goal model 2. Add Do-Be-Feel-Who list	23 Mar 2023
1.0.0	1. A basic goal model	15 Mar 2023

Version 2.0.0

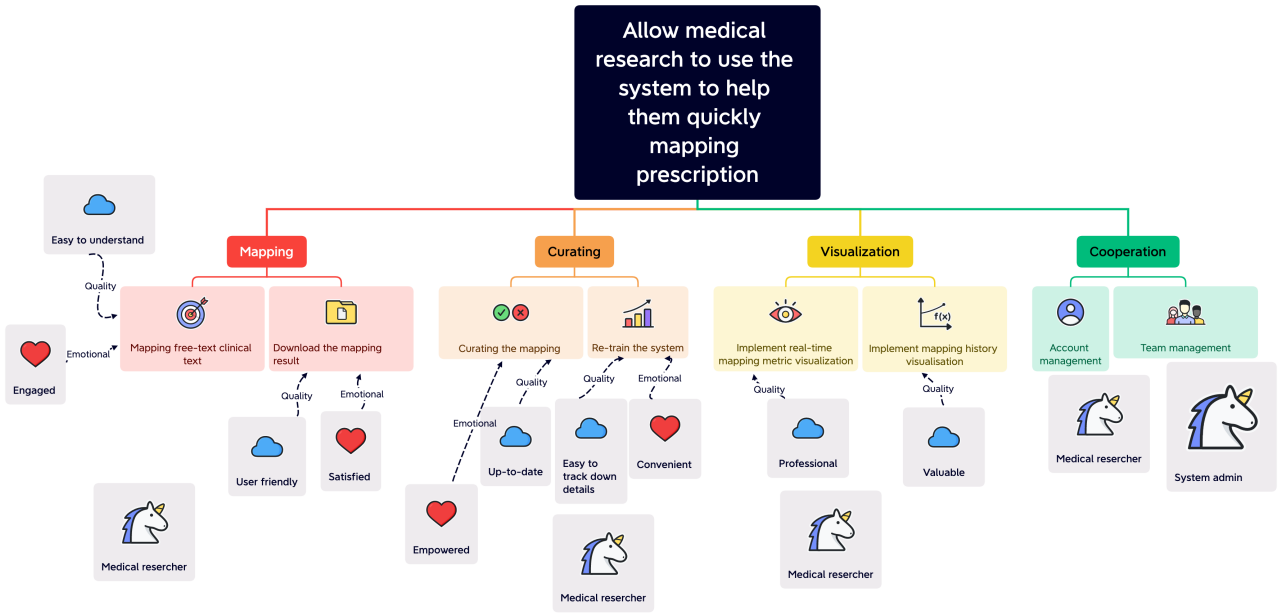
Do-Be-Feel-Who List

Do(Functional Goal)	Be (Quality Goal)	Feel(Emotional Goal)	Who(Roles)
Map the free-text clinical text	Easy to understand	Engaged	Medical researcher
Download the mapping results	User friendly	Satisfied	System Admin
Curate the incorrect mapping results	Up-to-date	Empowered	
Re-train the system use the curated data	Easy to track down details	Convenient	
Implement real-time mapping metric visualisation	Professional		
Implement mapping history visualisation	Mapping results should be valuable		
Implement system versioning			
Account management			
Team management			

Custom Goal Model


Allow medical researcher to use the system to help them quickly mapping prescription
Mapping
Curating
Visualization
Cooperation
Mapping free-text clinical text
Download the mapping result
Curating the mapping
Re-train the system
Implement real-time mapping metric visualization
Implement mapping history visualisation
Implement system versioning
Account management
Team management
Quality
Emotional
Easy to understand
Engaged
Medical researcher
User friendly
Satisfied
Empowered
Up-to-date
Easy to track down details
Convenient
Professional
Valuable
自由主题
System admin
Presented with xmind

Version 1.0.0

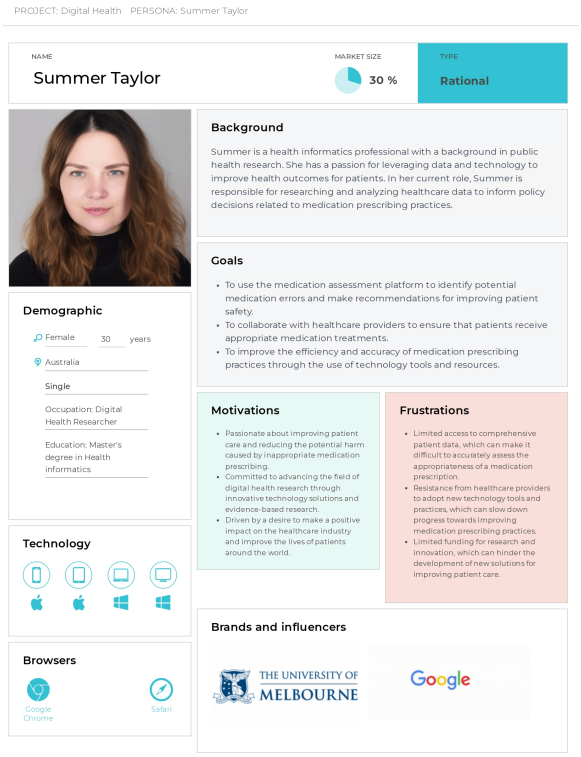

Allow medical research to use the system to help them quickly mapping prescription
Easy to understand
Mapping
Curating
Visualization
Cooperation
Quality
Emotional
Engaged
Mapping free-text clinical text
Download the mapping result
Curating the mapping
Re-train the system
Implement real-time mapping metric visualization
Implement mapping history visualisation
Account management
Team management
User friendly
Satisfied
Empowered
Up-to-date
Easy to track down details
Convenient
Professional
Valuable
Medical researcher
System admin
Presented with xmind

Personas

Versions

Version	Description	Date
2.0.0	1. Add persona analysis as justification for personas	23 Apr 2023
1.0.0	1. Three basic Personas for the sprint 1 submission	23 Mar 2023

Version 2.0.0



Persona Analysis

Summer Taylor

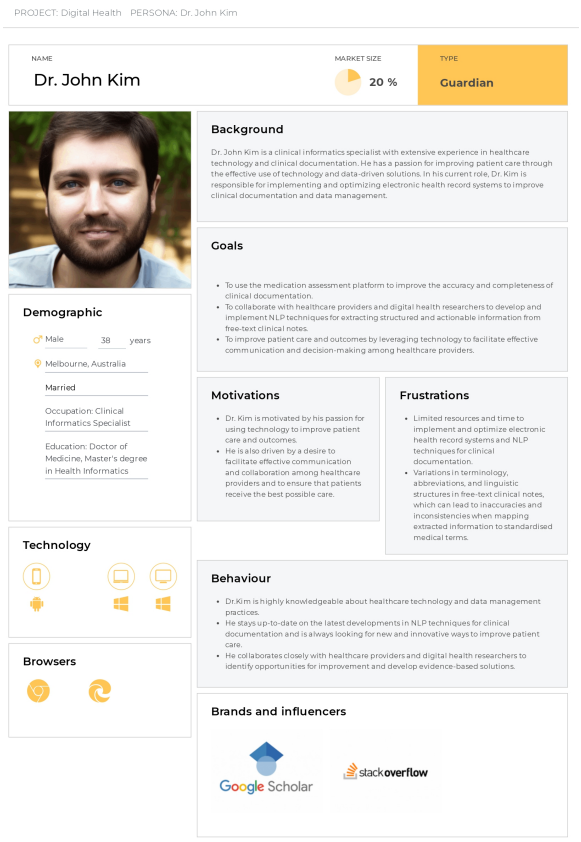
Summer's goals involve using the medication assessment platform to identify potential medication errors, collaborating with healthcare providers, and improving efficiency and accuracy in medication prescribing practices through technology. By focusing on these goals in the system design, the development team can create a tool that enhances patient safety, facilitates collaboration, and streamlines the medication prescribing process.

Summer's motivations, including passion for improving patient care, commitment to advancing digital health research, and a desire to make a positive impact on the healthcare industry, guide the development team in building a system that meets the needs of medical researchers and healthcare providers while driving innovation and enhancing patient care.

Addressing Summer's frustrations, the system design should prioritize providing comprehensive access to patient data, encouraging the adoption of technology tools among healthcare providers, and exploring ways to secure funding for research and innovation. By tackling these challenges, the development team can create a system that overcomes barriers to improved patient care and supports the advancement of the healthcare industry.

In summary, by considering Summer Taylor's goals, motivations, and frustrations, the development team can validate the system design and create a tool that meets the unique needs of medical researchers and healthcare providers, ultimately leading to better patient care and enhanced healthcare outcomes.

Persona Analysis



Dr. John Kim

By considering Dr. John Kim's goals, motivations, and frustrations, the development team can validate the system design and create a tool that caters to the needs of clinical informatics specialists, improving patient care and outcomes.

Dr. Kim's goals include using the medication assessment platform to enhance the accuracy and completeness of clinical documentation, collaborating with healthcare providers and digital health researchers to develop and implement NLP techniques, and leveraging technology to improve patient care and decision-making. By addressing these goals in the system design, the development team can create a tool that streamlines clinical documentation, fosters collaboration, and facilitates effective communication and decision-making among healthcare providers.

Dr. Kim's motivations, such as his passion for using technology to improve patient care and his desire to facilitate effective communication and collaboration among healthcare providers, guide the development team in creating a system that not only meets the needs of clinical informatics specialists but also promotes better patient care and outcomes.

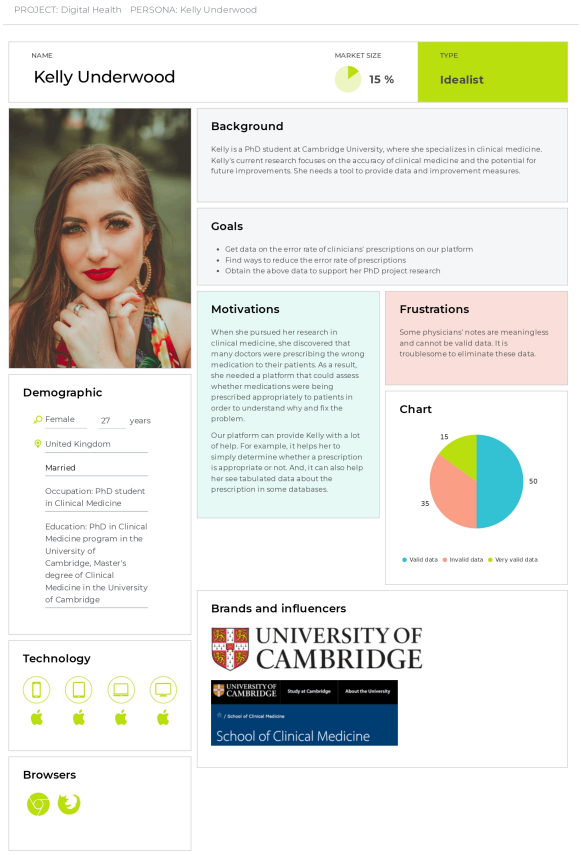
By addressing Dr. Kim's frustrations, the system design should prioritize optimizing electronic health record systems and NLP techniques, even with limited resources and time, and tackle variations in terminology, abbreviations, and linguistic structures to minimize inaccuracies and inconsistencies. By overcoming these challenges, the development team can create a system that supports clinical informatics specialists in their pursuit of improved patient care and enhanced healthcare outcomes.

In summary, by considering Dr. John Kim's goals, motivations, and frustrations, the development team can validate the system design and create a tool that meets the unique needs of clinical informatics specialists, ultimately leading to better patient care and improved healthcare outcomes.

Persona Analysis

Kelly Underwood

By considering Kelly Underwood's goals, motivations, and frustrations, the development team can validate the system design and create a tool that caters to the needs of PhD students in clinical medicine, ultimately helping them advance their research and improve patient care.



Kelly's goals include obtaining data on the error rate of clinicians' prescriptions, finding ways to reduce prescription error rates, and using this data to support her PhD research project. By addressing these goals in the system design, the development team can create a tool that provides valuable insights into prescription error rates and assists in identifying potential solutions.

Kelly's motivations, such as her passion for understanding and addressing the issue of inappropriate medication prescribing, guide the development team in creating a system that meets her needs and supports her research efforts. By offering features that help her easily assess prescription appropriateness and access relevant data, the platform can significantly contribute to her research.

Addressing Kelly's frustrations, the system design should focus on filtering out meaningless physician notes and ensuring that the data used in her research is valid and reliable. By developing features that help identify and eliminate such data, the development team can create a system that streamlines Kelly's research process and enhances the quality of her findings.

In summary, by considering Kelly Underwood's goals, motivations, and frustrations, the development team can validate the system design and create a tool that meets the unique needs of PhD students in clinical medicine, ultimately contributing to improved patient care and the advancement of clinical medicine research.

Version 1.0.0

NAME

Summer Taylor

MARKET SIZE

30 %

TYPE

Rational

Demographic

Female 30 years

Australia

Single

Occupation: Digital Health Researcher

Education: Master's degree in Health informatics

Technology

Browsers

Google Chrome

Safari

Background

Summer is a health informatics professional with a background in public health research. She has a passion for leveraging data and technology to improve health outcomes for patients. In her current role, Summer is responsible for researching and analyzing healthcare data to inform policy decisions related to medication prescribing practices.

Goals

- To use the medication assessment platform to identify potential medication errors and make recommendations for improving patient safety.
- To collaborate with healthcare providers to ensure that patients receive appropriate medication treatments.
- To improve the efficiency and accuracy of medication prescribing practices through the use of technology tools and resources.

Motivations

- Passionate about improving patient care and reducing the potential harm caused by inappropriate medication prescribing.
- Committed to advancing the field of digital health research through innovative technology solutions and evidence-based research.
- Driven by a desire to make a positive impact on the healthcare industry and improve the lives of patients around the world.

Frustrations

- Limited access to comprehensive patient data, which can make it difficult to accurately assess the appropriateness of a medication prescription.
- Resistance from healthcare providers to adopt new technology tools and practices, which can slow down progress towards improving medication prescribing practices.
- Limited funding for research and innovation, which can hinder the development of new solutions for improving patient care.

Brands and influencers

THE UNIVERSITY OF MELBOURNE

Google

NAME

Dr. John Kim

MARKET SIZE

20 %

TYPE

Guardian

Demographic

Male 38 years

Melbourne, Australia

Married

Occupation: Clinical Informatics Specialist

Education: Doctor of Medicine, Master's degree in Health Informatics

Technology

Browsers

Background

Dr. John Kim is a clinical informatics specialist with extensive experience in healthcare technology and clinical documentation. He has a passion for improving patient care through the effective use of technology and data-driven solutions. In his current role, Dr. Kim is responsible for implementing and optimizing electronic health record systems to improve clinical documentation and data management.

Goals

- To use the medication assessment platform to improve the accuracy and completeness of clinical documentation.
- To collaborate with healthcare providers and digital health researchers to develop and implement NLP techniques for extracting structured and actionable information from free-text clinical notes.
- To improve patient care and outcomes by leveraging technology to facilitate effective communication and decision-making among healthcare providers.

Motivations

- Dr. Kim is motivated by his passion for using technology to improve patient care and outcomes.
- He is also driven by a desire to facilitate effective communication and collaboration among healthcare providers and to ensure that patients receive the best possible care.

Frustrations

- Limited resources and time to implement and optimize electronic health record systems and NLP techniques for clinical documentation.
- Variations in terminology, abbreviations, and linguistic structures in free-text clinical notes, which can lead to inaccuracies and inconsistencies when mapping extracted information to standardised medical terms.

Behaviour

- Dr.Kim is highly knowledgeable about healthcare technology and data management practices.
- He stays up-to-date on the latest developments in NLP techniques for clinical documentation and is always looking for new and innovative ways to improve patient care.
- He collaborates closely with healthcare providers and digital health researchers to identify opportunities for improvement and develop evidence-based solutions.

Brands and influencers

Google Scholar

stackoverflow

NAME

Kelly Underwood

MARKET SIZE

15 %

TYPE

Idealist

Background

Kelly is a PhD student at Cambridge University, where she specializes in clinical medicine. Kelly's current research focuses on the accuracy of clinical medicine and the potential for future improvements. She needs a tool to provide data and improvement measures.

Goals

- Get data on the error rate of clinicians' prescriptions on our platform
- Find ways to reduce the error rate of prescriptions
- Obtain the above data to support her PhD project research

Motivations

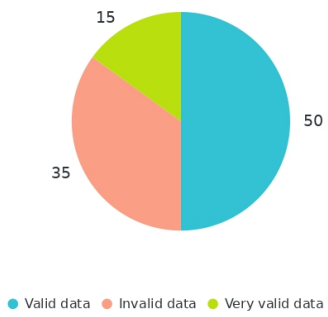
When she pursued her research in clinical medicine, she discovered that many doctors were prescribing the wrong medication to their patients. As a result, she needed a platform that could assess whether medications were being prescribed appropriately to patients in order to understand why and fix the problem.

Our platform can provide Kelly with a lot of help. For example, it helps her to simply determine whether a prescription is appropriate or not. And, it can also help her see tabulated data about the prescription in some databases.

Frustrations

Some physicians' notes are meaningless and cannot be valid data. It is troublesome to eliminate these data.

Chart



Demographic

Female 27 years

United Kingdom

Married

Occupation: PhD student in Clinical Medicine

Education: PhD in Clinical Medicine program in the University of Cambridge, Master's degree of Clinical Medicine in the University of Cambridge

Technology

Browsers

Brands and influencers

UNIVERSITY OF CAMBRIDGE

UNIVERSITY OF CAMBRIDGE | Study at Cambridge | About the University

/ School of Clinical Medicine

School of Clinical Medicine

User Stories and Acceptance Criteria

mapping task Note: All of these user stories are not generated by ChatGPT.

Version ID	Description	Date
2.0.0	1. Add ID to each user story 2. Split all user stories 3. Add specific priority 4. Add acceptance criteria 5. Add note to the each user story to explain the priority 6. Update some user stories with better descriptions 7. Change '*view other users' previous mapping task*' into '*US0003: View mapping task history*' under '*Epic 1 - Map free-text clinical text*' which includes viewing tasks of their owns and others in the team. 8. Delete '*filter the mapping history by date* range' and '*filter the mapping history by user*', instead keep them into decomposed tasks in *US0003: View mapping task history*' 9. Merge "*view the possible categories on each mapped item*" and "*see a list of category options while curating the data*" into "*US0005: View Category Options for Mapped Items*" 10. Merge "*re-train the system using the curated data' and 'curate the incorrect mapping results by correcting the data' into ' US0006: Curate Mapping Result*' 11. According to new requirement, add "*US0007: Update the Version of UIL*"	21 Apr 2023
1.0.0	Basic user stories with epic, stakeholders, size, priority and story point	23 Mar 2023

Version 2.0.0 - User Stories and Acceptance Criteria

Introduction

The user stories is listed as cards, with user story id, user story title, acceptance criteria, estimated story point, size estimation, priority rank, MoSCow priority, and a brief description about the priority.

User story card example

The following table provide an example about the user story which will be present in the following sections.

Note that the acceptance criteria is demonstrated in the Given-When-Then (GWT) format which is is a structure used in Behavior-Driven Development (BDD) to describe and test software features. It consists of three parts: setting the context (Given), specifying the action or event (When), defining the expected outcome (Then) and additional description of the previous three parts (And). GWT helps ensure clear communication and shared understanding among team members.

<User story ID>: <User Story Title>	
User story	**As** <stakeholder> **I want to** <> **So that** <>
Acceptance criteria	Given <> AND...(optional) When <> AND...(optional) Then <> AND...(optional)
Story point	<Fibonacci story point: 0, 1, 2, 3, 5, 8, 13>
Size estimation	<User story size: S, M, L, XL>
Priority	<Priority rank: 1, 2, 3, ..., No. of user stories>
MoSCow Priority	One of (Could have, Should have, Must have)

Note	<Priority description>

Epic 1 - Map free-text clinical text

US0001: Map short text into the terms of UIL(universal indication list)	
User story	**As** a medical researcher **I want to** map the short text into certain or multiple categories in a universal indication list **So that** I do not need to map the text manually
Acceptance criteria	Given there is an input box or a file uploader on the mapping webpage When I access the mapping webpage as a medical researcher Then I should see a user-friendly interface that allows me to enter a short text or upload a file of short free text describing a symptom or indication, and a button or option to submit the text for mapping processing.
Story point	8
Size estimation	XL
Priority	1
MoSCoW Priority	Must have
Note	This is the core functionality of the system, which enables users to automate the mapping process, saving time and effort.

US0002: Download the mapping result	
User story	**As a** medical researcher **I want to** download the mapping results **So that** I can save the mapping history and view it in the future
Acceptance criteria	Given I am a medical researcher who has completed a mapping process on the webpage When I review the displayed mapping results, Then I should see a "Download" button or option on the webpage, And when I click the "Download" button or option Then the system should generate a downloadable file containing the mapping results And the file should be in a commonly used format, such as CSV or Excel, And the download process should initiate, allowing me to save the file to my local storage or device.
Story point	3
Size estimation	S
Priority	2
MoSCow Priority	Must have
Note	Providing users with the ability to download and save their mapping results helps them maintain a historical record and facilitates future reference.

US0003: View mapping task history	
User story	**As a** medical researcher **I want to** view my own and team members' previous mapping tasks **So that** I can review the mapping results and performance
Acceptance criteria	Given I am a medical researcher who wants to view the mapping history of myself and other team members When I am logged in and visit history section of the mapping tool's interface Then I should see a list of mapping tasks which I can review mapping history and performance And the option or button should be easy to locate and use.
Story point	5
Size estimation	L
Priority	7
MoSCow Priority	Must have
Note	Viewing previous tasks of own and others in the team fosters collaboration, knowledge sharing, and helps maintain quality standards.

Epic 2 - Curate the mapping and re-train the system

US0004: Identify Results Status in Mapping Process

US0005: View Category Options for Mapped Items

US0006: Curate Mapping Result

User story	**As a** medical researcher (curator) **I want to** easily identify the status of the mapping process for each raw text input **So that** I can quickly determine if further curation or review is needed
Acceptance criteria	Given I am a medical researcher (curator) who has completed a mapping task When I review the displayed mapping results for different scenarios (e.g., no mapping, single or multiple mappings, previously curated mappings) Then I can see the relevant information (e.g., concept, similarity /confidence score, status)
Story point	5
Size estimation	M
Priority	3
MoSCoW Priority	Must have
Note	By providing an easy-to-use interface for reviewing mapping results under various scenarios, medical data curators can quickly identify inaccuracies and make necessary corrections.

User story	**As a** medical researcher (curator) **I want to** see a list of category options while curating the data **So that** I can choose the correct category for the incorrect mapping or unmapped items
Acceptance criteria	Given I am a medical researcher (curator) who has completed a mapping process on the webpage When I review the displayed mapping results and identify an incorrect mapping or unmapped items Then I should see an option to view a list of category options for the them, such as a dropdown menu, search bar, or button And when I interact with the option to view category options, Then the system should display a list of available categories from the universal indication list that I can choose from.
Story point	3
Size estimation	M
Priority	4
MoSCoW Priority	Must have
Note	Providing category options during curation improves the user experience and facilitates more accurate mapping corrections.

User story	**As a** medical researcher (curator) **I want to** review and curate the failed-mapping result or mapping results of raw text to SNOMED-CT into UIL **So that** I can have more accurate mapping result and give feedback to the system and improve it
Acceptance criteria	Given I am a medical researcher (curator) reviewing a failed-mapping result or successful mapping result on SNOMED-CT When I identify an appropriate mapping to a UIL concept Then I can manually curate the result using a drop-down menu with UIL entries and provide feedback to the system for improvement.
Story point	5
Size estimation	M
Priority	5
MoSCoW Priority	Must have
Note	Allowing users to curate data and provide feedback is essential for refining the system's performance and improving its mapping capabilities.

US0007: Retrain the system from curating	
User story	**As a** medical researcher (curator) **I want to** re-train the system using the curated data **So that** I can make the system have a better mapping performance in the future

US0008: Update the Version of UIL	
User story	**As an** admin user **I want to** update the UIL to the latest version **So that** I can access the most up-to-date term for curation

Acceptance criteria	Given the system has received curated data from one or more medical researchers (curators) When the system undergoes a re-training process Then it should incorporate the curated data into the re-training process to improve the mapping performance And the system should update its algorithms and knowledge base to better align with the curated data, And the improved mapping performance should be reflected in future mapping processes.
Story point	5
Size estimation	L
Priority	6
MoSCow Prioity	Must have
Note	Continuously refining the system based on user feedback is key to ensuring it remains relevant and useful over time.

Acceptance criteria	Given the system is running and the user has the necessary permissions, and the new UIL file is in a compatible format with the existing system When the user initiates the UIL update process, and the system verifies the integrity of the new UIL and ensures that it is not corrupted during the update process, and the system provides a backup option for the old UIL, and the system verifies the compatibility of the updated UIL Then the system should prompt the user to provide the new version of the UIL, display a confirmation message to the user indicating that the update process has been completed successfully
Story point	5
Size estimation	M
Priority	8
MoSCow Priority	Must have
Note	It is considered high-priority to make sure the user of the system access the most up-to-date information contained in the UIL they would like to identify for the mapping.

Epic 3 - Implement mapping metric visualization

US0009: Have a dashboard to display the mapping metrics	**US0010: View the successful mapping rate and the number of mapped items**	**US0011: View the overall confidence**

User story	**As a** medical researcher **I want to** have a dashboard to display the mapping metrics for a mapping task **So that** I can review and analyze mapping results
Acceptance criteria	Given I am a medical researcher who wants to review and analyze mapping results When I review the mapping metric dashboard on the mapping result page Then I should see an button to display a metric dashboard And when I click the button to access the dashboard, then the dashboard should be displayed, showing the mapping metrics for current task.
Story point	8
Size estimation	L
Priority	9
MoSCow Priority	Must have
Note	Offering metric analytics allows users to make informed decisions and monitor the system's performance.

User story	**As a** medical researcher **I want to** view the successful mapping rate and the number of mapped items in a dashboard **So that** I can assess the performance of the system on the current mapping task
Acceptance criteria	Given I am a medical researcher who has accessed the dashboard on the mapping result page When I review the mapping metric dashboard Then I should see the successful mapping rate, represented as a percentage or ratio of successful mappings to total mappings And I should see the total number of mapped items, indicating the volume of data that has been processed
Story point	5
Size estimation	S
Priority	14
MoSCow Priority	Should have
Note	Providing performance metrics enables users to evaluate the system's effectiveness and make informed decisions about its usage.

User story	**As a** medical researcher (curator) **I want to** view the overall confidence in a dashboard **So that** I can assess the performance of the system on the current mapping task
Acceptance criteria	Given I am a medical researcher who has accessed the dashboard on the mapping result page When I review the mapping metric dashboard Then I should see the overall confidence, represented as an average or median confidence score across all successful mappings And the overall confidence should be displayed in a clear and understandable format, such as a text summary or graphical representation
Story point	3
Size estimation	S
Priority	15
MoSCow Priority	Should have
Note	Providing an overall confidence metric helps users understand the system's performance and trust in the mapping results.

US0012: View the Specific Performance On Each Category	
User story	**As a** medical researcher **I want to** view the specific performance of each category **So that** I can know the system performance among different categories
Acceptance criteria	Given I am a medical researcher who has accessed the mapping history When I review the displayed mapping results Then I should see specific performance metrics for each category, such as successful mapping rate, average confidence score, and number of mapped items

US0013: View the Proportion of Each Category	
User story	**As a** medical researcher **I want to** view the proportion of each category **So that** I can know the distribution of different categories
Acceptance criteria	Given I am a medical researcher who has accessed the mapping history When I review the displayed mapping results Then I should see the proportion of each category, represented as a percentage or ratio of mapped items within that category to the total number of mapped items

Story point	5
Size estimation	M
Priority	16
MoSCoW Priority	Could have
Note	Offering category-specific performance data helps users identify areas for improvement and make targeted adjustments.

Story point	5
Size estimation	S
Priority	17
MoSCoW Priority	Could have
Note	Displaying category distribution information helps users identify trends and understand the data they are working with.

Epic 4 - Implement mapping history visualization

US0014: Visualize Mapping Performance History	
User story	**As a** medical researcher **I want to** visualize the history of the mapping performance **So that** I can measure the mapping quality
Acceptance criteria	Given I am a medical researcher who wants to visualize the history of the mapping performance When I access a dedicated section or page on the mapping tool's interface Then I should see a visual representation of the mapping performance over time, such as a line chart, bar chart, or other appropriate visualization.
Story point	8
Size estimation	XL
Priority	11
MoSCoW Priority	Must have

US0015: Rollback to Earlier System Version	
User story	**As an** admin user **I want to** roll back to the earlier (default) version of the mapping system **So that** I can restore the default behaviour of the system
Acceptance criteria	Given I am an admin user who wants to rollback to an earlier version (default) of the system When I access the mapping history performance page Then I should see the available version restore button And the version restoring should be easy to locate and use.
Story point	5
Size estimation	M
Priority	12
MoSCoW Priority	Should have

Note	Visualizing the performance history allows users to track progress and identify trends over time.

Note	Providing version rollback capabilities ensures admins can maintain system stability and performance in case of problematic updates.

Epic 5 - Manage account

US0016: Account Login	
User story	**As a** medical researcher **I want to** log in to my account **So that** I can be identified by the system for curating.
Acceptance criteria	Given I am a medical researcher who wants to log in to my account When I visit the mapping tool's website or application Then I should see a login page or button allowing me to access the login interface And the login page or button should be easy to locate and use.
Story point	3
Size estimation	S
Priority	10
MoSCow Priority	Must have
Note	User authentication ensures a secure environment and allows for personalized experiences.

US0017: Edit Personal Information	
User story	**As a** medical researcher **I want to** edit my personal information **So that** I can update my roles and responsibilities.
Acceptance criteria	Given I am a medical researcher who wants to edit my personal information When I am logged in to my account and visit the user profile or settings section of the mapping tool's interface Then I should see an option or button to edit my personal information And the option or button should be easy to locate and use.
Story point	2
Size estimation	S
Priority	18
MoSCow Priority	Could have
Note	Allowing users to edit their personal information helps maintain up-to-date user profiles and improves collaboration.

US0018: Add New User Account	
User story	**As an** admin user **I want to** add an account to the system **So that** I can give other people access to the system.
Acceptance criteria	Given I am a medical researcher who wants to add an account to the system And I have the necessary permissions to manage user accounts When I visit the account management section or page on the mapping tool's interface Then I should see an option or button to add a new account And the option or button should be easy to locate and use.
Story point	5
Size estimation	M
Priority	13
MoSCow Priority	Should have
Note	Adding new accounts enables collaboration and helps scale the system to accommodate more users.

Epic 6 - Manage team

US0019: Add Member to Team	
User story	**As an** admin user **I want to** add the account to a team **So that** I can let people on the same team work together.

US0020: Remove Member from Team	
User story	**As an** admin user **I want to** delete a member from a team **So that** I can remove a member's privileges in a team

Acceptance criteria	Given I am a medical researcher who wants to add an account to a team And I have the necessary permissions to manage team membership When I visit the team management section or page on the mapping tool's interface Then I should see an option or button to add a new member to the team
Story point	1
Size estimation	S
Priority	19
MoSCow Priority	Could have
Note	Team-based collaboration features help organize users and facilitate more efficient collaboration.

Acceptance criteria	Given I am a medical researcher who wants to delete a member from a team And I have the necessary permissions to manage team membership When I visit the team management section or page on the mapping tool's interface Then I should see an option or button to remove an existing member from the team And the option or button should be easy to locate and use.
Story point	1
Size estimation	S
Priority	20
MoS Cow Priority	Could have
Note	Enabling user management within teams ensures proper access control and maintains the integrity of the system.

Version 1.0.0 - User Story

Epic	As a	I want to	So That	Size Estimation	MoSCow Priority	Story point
Map free-text clinical text	medical researcher	map the short text into certain or multiple categories in a universal indication list	I do not need to map the text manually.	L	must have	5
		download the mapping results	I can save the mapping history and view it in the future.	S	should have	3
		view the possible categories on each mapped item	I can make decisions about the most appropriate category for the term.	M	must have	5
Curate the mapping and re-train the system	medical researcher (curator)	identify the unrecognized result	I can ensure that the mapping result is accurate and complete.	M	must have	5
		filter the possible incorrect mapping result by the confidence range	I can judge the possible incorrect mappings by myself.	M	must have	8
		curate the incorrect mapping results by correcting the data	I can give feedback to the system and retrain it.	M	must have	3
		see a list of category options while curating the data	I can choose the correct category for the incorrect mapping.	S	should have	1
		re-train the system using the curated data	I can make the system have a better mapping performance in the future.	L	must have	5
		test the updated system	I can ensure that the mapping is improved.	M	could have	5
Implement real-time mapping metric visualization	medical researcher	have a real-time dashboard to display the mapping metrics	I can review and analyze mapping results.	L	must have	8

		view the successful mapping rate and number of mapped items	I can assess the performance of the mapping tool accurately.	S	should have	5
		view the overall confidence	I can know the performance of the system.	S	should have	3
		view the specific performance of each category	I can know the system performance among different categories.	M	could have	5
		view the proportion of each category	I can know the distribution of different categories.	S	could have	5
Implement mapping history visualization	medical researcher	visualize the history of the mapping performance	I can measure the mapping quality	XL	must have	8
		filter the mapping history by date range	I can quickly find the mapping history	S	could have	1
		filter the mapping history by user	I can quickly find the mapping history of a specific team member	S	could have	1
	admin user	rollback to earlier version	I can choose the system version with better performance	M	could have	5
Manage account	medical researcher	login to my account	I can be identified by the system for curating.	S	must have	3
		edit my personal information	I can update my roles and responsibilities.	S	could have	2
	admin user	add an account to the system	I can give other people access to the system.	S	should have	1
Manage team	medical researcher	view other users' previous mapping task	I can check their mapping results and performance.	M	should have	3
	admin user	add the account to a team	I can let people on the same team work together.	S	could have	1
		delete a member from a team	I can remove a member's privileges in a team.	S	could have	1

Use Cases

Version	Description	Date
2.0.0	1. Based on the MRFF NCAS mapping tool requirements provided by Vlada on April 27, 2023.	24 Apr 2023
1.0.0	1. Basic use cases	23 Mar 2023

Version 2.0.0

Use Case 1: Map Free-Text clinical text to UIL category if no SNOMED CT concepts are mapped

Rationale:

A medical researcher wants to map the short clinical text into a universal indication category, automating the process and reducing manual effort for their research.

Preconditions:

1. User has available account.
2. A universal indication list is available and integrated with the system.
3. A tool to map raw text into SNOMED CT concept is available and integrated with the system.

Steps:

1. The user inputs the clinical text or upload a file with multiple clinical text into the system, and there will be three cases
    a. The map tool match the clinical text into SNOMED CT concepts, but no concepts are matched.
        i. Output of the mapping tool:
            1. Matched term: NULL
            2. Score: NULL
            3. Status: Fail
            4. Source: NULL
    b. The map tool match the clinical text into SNOMED CT concepts, and there is only one concept is matched.
        i. Output of the mapping tool:
            1. Matched term: if the SNOMED CT concept ID is in UIL, then set term to *UIL category*, otherwise, set source to *SNOMED-CT* concept
            2. Score: <confidence score>
            3. Status: <Success>
            4. Source: if the SNOMED CT concept ID is in UIL, then set source to *UIL*, otherwise, set source to *SNOMED-CT*
    c. The map tool match the clinical text into SNOMED CT concepts, and there are multiple concepts are matched.
        i. Select the SNOMED CT concept with the highest score.
        ii. Output of the mapping tool:
            1. Matched term: if the SNOMED CT concept ID is in UIL, then set term to *UIL category*, otherwise, set source to *SNOMED-CT* concept
            2. Score: <confidence score>
            3. Status: <Success>
            4. Source: if the SNOMED CT concept ID is in UIL, then set source to *UIL*, otherwise, set source to *SNOMED-CT*

Use Case 2: Curate the mapping result into UIL category

Rationale

Based on the use case 1, the mapping system may incorrectly classify the clinical text into wrong UIL category(even only classified into SNOMED CT concept but no UIL category), so the mechanism is designed to provide a way for the user to review and curate the unrecognised and possible incorrect mappings so that the system can learn from mapping result and improve its accuracy over time.

Preconditions

1. User has available account.
2. The system has successfully process a map task and generate a set of preliminary mappings.

Steps

1. The user successfully login
2. The user click into the map task to view its details.
3. The module displays the text that was mapped, along with the mapping term(could be none), confidence score(could be none), status(reviewed /success/fail), and source(SNOMED-CT/UIL)
4. The user can select any clinical text(no matter what ststatus: mapped success, reviewed, or fail) to change its mapped term into other UIL category from a dropdown menu
5. Then the status will change to reviewed.
6. When the user is finished reviewing and curating the mappings, they can submit their changes to the system, or they can save it to continue review next time.

Postconditions

1. The system uses the curated data to retrain its model, in order to improve the accuracy of future mappings.

Use Case 3: Visualise the history of the mapping performance

Rationale:

This is one of the key functional requirements of the project. Medical researchers need the system to show the historical mapping performance so that they can measure if the system is improving its accuracy.

Preconditions:

1. User has available account.
2. The system have the mapping short text into terms implemented
3. The user has at least one mapping result
4. (Optional)The system can be retrained by curating the mapping results manually.

Steps:

1. The user successfully login
2. The user clicks the mapping history button on the side navigation bar

Alternative Paths:

- If the user has no mapping result before, the page will show empty
- If the user does not have permission to watch the mapping history, then the user cannot see the performance

Postconditions:

- The user can see the overall training time
- The user can see the current curate rate and mapping failure rate
- The user can see the performance change over time

Use Case 4: Filter the mapping history/task by user and date range

Rationale:

This is a simple filter use case that allows the user to retrieve the mapping task of a specific person so that the team members.

Preconditions:

1. User has available account.
2. The user is in a workspace
3. The user has access to see other user's mapping
4. Other user has mapping result

Steps:

1. The user successfully login
2. User clicks the page go to the mapping tasks, so the user can see a list of mapping task
3. The user can filter those mapping task by selecting the date and users

Alternative Paths:

- If the workspace does not have any mapping history, the mapping history page will be a blank page

Postconditions:

- The user can filter the mapping history in the order of date
- The user can filter the mapping history of specific users

Use Case 5: A medical research wants to view team members mapping tasks.

Rationale:

A medical researcher wants to view team members mapping tasks so that they can help each other to curate the mapping.

Preconditions:

1. User has available account.
2. User is in a workspace
3. The username and password entered are accurate and matched.
4. Other team members have mapping history.

Step:

1. The user successfully login
2. The system prompts the user to log in successfully and jumps to the main page.
3. The user can view all members' mapping task in workspace mapping task page.

Alternative Path:

1. If there are no mapping tasks for other users, the system will display an empty screen.

Postconditions:

- After the user has viewed other users' previous mapping tasks, the user can make corrections to other users' mapping results.

Use Case 6: Perform account profile edit

Rationale:

A medical researcher wants to edit personal information

Preconditions:

1. User has available account.
2. The username and password entered are accurate and matched.
3. The system holds the user's personal information.

Steps:

1. The user successfully login
2. The system prompts the user to log in successfully and jumps to the main page.
3. The user enters the profile page.
4. The system displays the user's personal information.
5. The user modifies personal information and submits it.

Alternative Paths:

1. If the user enters an incorrect or mismatched username and password, the system will ask the user to re-enter it.
2. If the user's modified personal information does not match the rules, the system will ask the user to re-enter it.

Postconditions:

- After the user has viewed other users' previous mapping tasks, the user can make corrections to other users' mapping results.

Use Case 7: Adding team member to workspace by admin

Rationale:

An admin user wants to add an account to the current workspace, as also deleting a member from a workspace.

Preconditions:

1. User has available account.
2. The system shows that the admin user has logged in successfully.
3. The system provides functions for the workspace.

Steps:

1. The user successfully login
2. The admin user clicks the add user button in the workspace to invite member.
3. The admin user enters the user's email and submits it, and the user's status shows *Pending*.
4. The system send a invitation email to the email with a invitation link.
5. The invited user click the invitation link in email, and then the invited user will be redirected to a page to sign up with personal information.
6. By clicking the submit button on sign up page, the invited person will be invited to the workspace.
7. The invited person status will shows *Active.*

Postconditions:

- A user will be invited to the workspace

Use Case 8: Admin can allow the system to restore the default behaviour of the mapping tool

Rationale:

The medical researchers wants to restore the default behaviour of the mapping tool, but this operation should only be done by admin account.

Preconditions:

1. User has available account.

Steps:

1. The admin successfully login
2. The admin user create a new workspace.

Postcondition:

The admin user successfully restore a default version of the mapping tool, and can switch back to previous version anytime.

Use Case 9: Download mapping result

Rationale:

The medical researchers wants to download the mapping result from a mapping tasks.

Preconditions:

1. User has available account.
2. The workspace of the user has a successfully mapped task.

Steps:

1. The user successfully login.
2. The user go to the mapping task page.
3. The user click any successfully mapped task to go to the task detail.
4. The user click download button to download result.

Postcondition:

The user will receive an excel file with mapping result will be downloaded.

Version 1.0.0

Use Case 1: Map Free-Text Clinical Text to Categories

Rationale:

A medical researcher wants to map the short clinical text into certain or multiple categories in a universal indication list, automating the process and reducing manual effort.

Preconditions:

1. The user has access to the clinical text mapping system.
2. A universal indication list is available and integrated with the system.
3. The clinical text to be mapped is provided in a suitable format.

Steps:

1. The user inputs the free-text clinical text into the system.
2. The system processes the text and identifies possible categories from the universal indication list.
3. The system presents the suggested categories to the user.
4. The system maps the clinical text to the selected categories.
5. The user has the option to download the mapping results to save the mapping history and view it in the future.
6. The user can view the possible categories for each mapped item, allowing them to make decisions about the most appropriate category for the term.

Alternative Paths:

1. If the system is unable to suggest appropriate categories, the user can manually search for and select categories from the universal indication list.
2. If the user needs to review or change the mapped categories later, they can access the mapping history and make updates.

Postcondition:

The clinical text is successfully mapped to one or multiple categories in the universal indication list, and the user can download the results and view possible categories for each mapped item.

Use Case 2: Curate the Mapping Results

Rationale

The purpose of this module is to provide a mechanism for the user to correct any misclassifications made by the system when mapping free-text clinical data to different categories. The module is designed to provide a way for the user to review and curate the unrecognized and possible incorrect mappings so that the system can learn from this feedback and improve its accuracy over time.

Preconditions

1. The user has access to the clinical text that needs to be mapped.
2. The system has already processed the clinical text and generated a set of preliminary mappings.
3. The system has identified at least one original text that fails to be mapped to a certain category or the user has identified at least one mapping that they believe to be incorrect.

Steps

1. The module displays a list of the mappings that the users believe to be incorrect by filtering the confidence range or the mapping status.
2. The module displays the text that was mapped, along with the current mapping and the confidence score for that mapping.
3. The user can review the text and the current mapping, and can then choose to either correct the mapping or leave it unchanged.
4. If the user chooses to correct the mapping, the module displays a list of categories that the text could be mapped to.
5. The user selects the correct category from the list, and the module updates the mapping accordingly.
6. The user can repeat steps 2-7 for any other mappings that they believe to be incorrect.
7. When the user is finished reviewing and curating the mappings, they can submit their changes to the system.
8. The system uses the curated data to retrain its model, in order to improve the accuracy of future mappings.

Alternative Paths

1. If the user does not identify any mappings that they believe to be incorrect, they can simply close the module without making any changes.
2. If the user encounters any technical issues or errors while using the module, they can contact technical support for assistance.

Postconditions

1. The mappings have been reviewed and curated by the user, with any incorrect mappings corrected and any missing mappings added.
2. The curated data has been submitted to the system for retraining the mapping model.
3. The system has been improved with the curated data, leading to increased accuracy in future mappings.

Use Case 3: Real-Time Dashboard for Clinical Text Mapping Metrics

Rationale:

The real-time dashboard module is designed to provide stakeholders with a visual representation of the mapping performance in real time. It allows users to monitor the accuracy and efficiency of the mapping process, and make any necessary adjustments to improve the system's performance.

Preconditions:

- The clinical text mapping system is up and running.
- The real-time dashboard module is installed and connected to the mapping system.
- There is a database containing the mapping metrics data.

Steps:

1. Open the real-time dashboard module and log in.
2. The dashboard displays the overall mapping metrics, including the successful mapping rate and the number of mapped items.
3. The dashboard also displays the overall confidence level of the system in mapping the text.
4. Users can click on a specific category to view the performance of the mapping tool for that category.
5. The dashboard displays the proportion of each category in the mapped items.
6. Users can apply filters to view the mapping history by user or date range.
7. If the mapping performance is unsatisfactory, users can click on the specific category to view the individual items that were mapped incorrectly.
8. Users can manually correct the mapping for any incorrect items and update the mapping system with the new data.
9. The mapping system will re-train the machine learning model based on the corrected mapping data.
10. The user can download the mapping metrics data for further analysis.

Alternative Paths:

- If the mapping system is not running or the real-time dashboard module is not connected, an error message will be displayed.
- If the user does not have the necessary permissions, they will not be able to access the real-time dashboard module.
- If there are no incorrectly mapped items, the correction step (steps 7 and 8) will not be necessary.

Postconditions:

- The real-time dashboard module displays the mapping metrics data, allowing users to monitor the accuracy and efficiency of the mapping system.
- Users can manually correct any incorrectly mapped items, and feed them back into the system to improve the accuracy of the mapping tool.
- The mapping system re-trains the machine learning model based on the corrected mapping data.
- The mapping metrics data can be downloaded for further analysis.

Use Case 4: Visualise the history of the mapping performance

Rationale:

This is one of the key functional requirements of the project. Medical researchers need the system to show the historical mapping performance so that they can measure if the system is improving its accuracy.

Preconditions:

- Medical researchers have accounts of the system
- The system have the mapping short text into terms implemented
- The user has at least one mapping result
- (Optional)The system can be retrained by curating the mapping results manually.

Steps:

1. The user successfully logins the system
2. The user clicks the mapping history button on the side navigation bar

Alternative Paths:

- If the user has no mapping result before, the page will show empty
- If the user does not have permission to watch the mapping history, then the user cannot see the performance

Postconditions:

- The user can see the overall training time
- The user can see the current curate rate and mapping failure rate
- The user can see the performance change over time

Use Case 5: Filter the mapping history by user and date range

Rationale:

This is a simple filter use case that allows the user to retrieve the mapping history of a specific person so that the team members can follow up with other team members' mapping

Preconditions:

- The User successfully login
- The user is in a team
- The user has access to see other user's mapping
- Other user has mapping result

Steps:

1. User login
2. User clicks the mapping history button on the side navigation bar, so the user can see a list of mapping history
3. The list shows the latest mapping history on the top
4. The user can filter those mapping histories by selecting the date and other users

Alternative Paths:

- If the team does not have any mapping history, the mapping history page will be a blank page
- If the user does not allow to see the mapping history, then the user cannot see any mapping history

Postconditions:

- The user can filter the mapping history in the order of date
- The user can filter the mapping history of specific users

Use Case 6: Perform account operations for a medical researcher

Rationale:

A medical researcher wants to log in to the account, edit personal information, and then view other users' previous mapping tasks.

Preconditions:

1. Users have available accounts.
2. The username and password entered are accurate and matched.
3. The system holds the user's personal information.
4. Other users have mapping history.

Steps:

1. The user enters the login interface of the system.
2. The user enters a username and password.
3. The user clicks the login button.
4. The system prompts the user to log in successfully and jumps to the main interface.
5. The user enters the account interface.
6. The system displays the user's personal information.
7. The user modifies personal information and submits it.
8. The user enters the History Stats screen.
9. The system displays other users' previous mapping tasks.

Alternative Paths:

1. If the user enters an incorrect or mismatched username and password, the system will ask the user to re-enter it.
2. If the user's modified personal information does not match the rules, the system will ask the user to re-enter it.
3. If there are no mapping tasks for other users, the system will display an empty screen.

Postconditions:

- After the user has viewed other users' previous mapping tasks, the user can make corrections to other users' mapping results.

Use Case 7: Perform account operations for an Admin User

Rationale:

An admin user wants to add an account to the system, add the account to a team, and then delete a member from a team.

Preconditions:

1. The admin user enters the correct and matching username and password on the login screen.
2. The system shows that the admin user has logged in successfully.
3. The system provides functions for the team.

Steps:

1. The admin user clicks the Create User button.
2. The system displays the Create User interface.
3. The admin user enters the user's information and submits it.
4. The system shows that the user has been created successfully.
5. The admin user enters the user interface of the created user.
6. The admin user adds the user to a team.
7. The system displays that the user joined the team successfully.
8. The admin user enters the team interface and selects a user.
9. The admin user clicks the Remove User button.
10. The system shows that the user was successfully removed from the team.

Alternative Paths:

1. If the user information entered does not match the rules, the admin user needs to re-enter the information.
2. If the user wants to join a team that does not exist, the admin user needs to create a team first.
3. If the team that the user wants to join is full, the user will not be able to join.

Postconditions:

- Users on the same team will be able to start collaborating on mapping.

Use Case 8: Admin Rollback Version of Mapped Categories

Rationale:

The admin user wants to rollback the versioning of mapped categories to a previous version in case of errors or inconsistencies in the current mapping.

Preconditions:

1. The admin user has access to the clinical text mapping system.
2. The mapping system has previous versions of the mapped categories available.

Steps:

1. The admin user logs into the mapping system.
2. The admin user accesses the mapping history and identifies the previous version of the mapped categories that they wish to restore.
3. The admin user selects the previous version to restore.
4. The system restores the selected version and replaces the current mapping with the restored version.
5. The system updates the mapping history to reflect the rollback and the restored version.

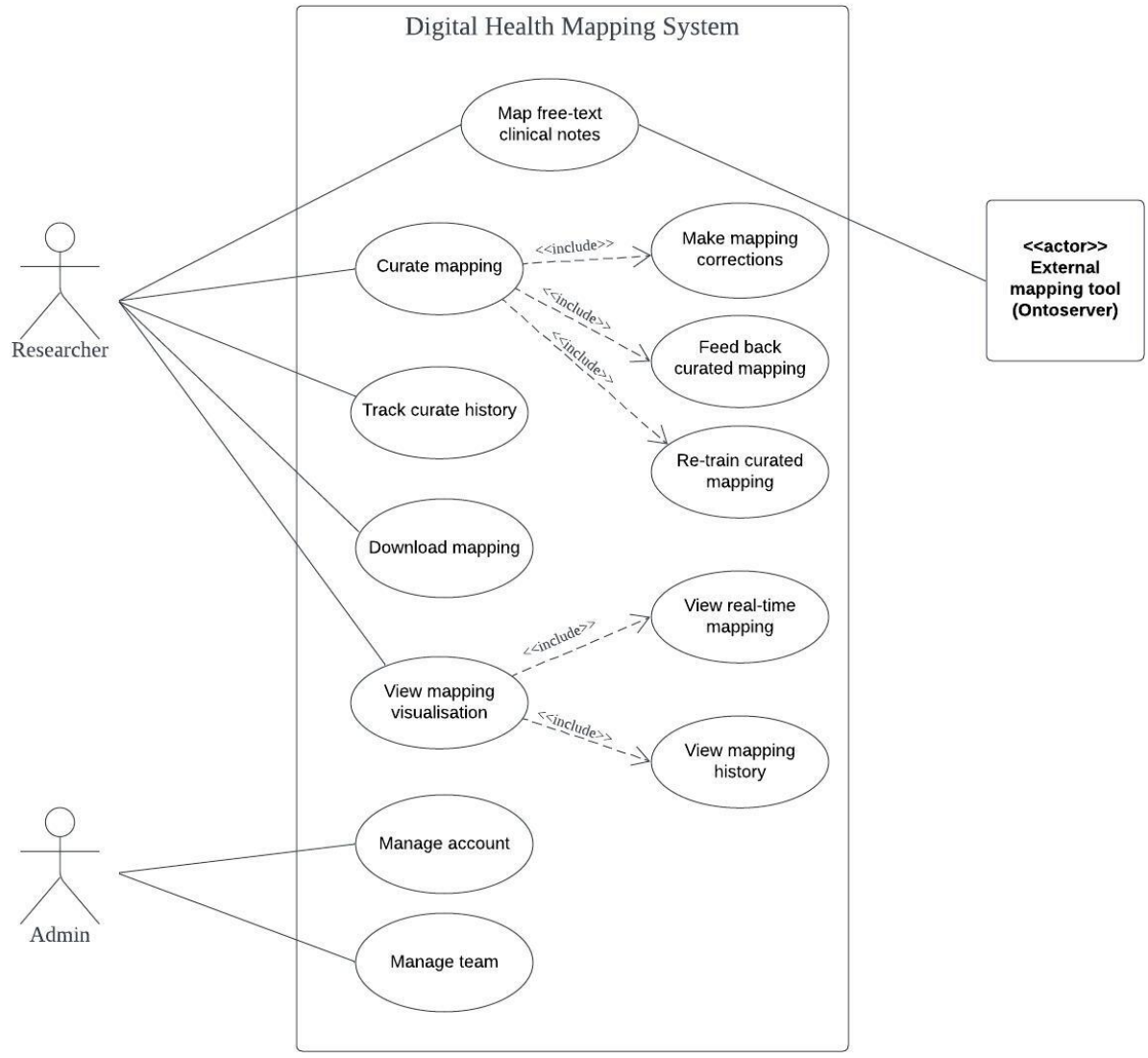
Alternative Paths:

1. If the admin user is unsure about which version to restore, they can preview the previous versions and their mapping results before selecting one to restore.
2. If the admin user encounters errors or inconsistencies in the restored version, they can repeat the rollback process and select a different version to restore.

Postcondition:

The admin user successfully rolls back the version of mapped categories to a previous version in the mapping system, and the system updates the mapping history to reflect the rollback and the restored version.

Overall Use Case Diagram



Product Backlog

Version ID	Description	Date
2.0.0	1. Add table to clearly show the product backlog 2. Align with User Stories and Acceptance Criteria modified by new requirements	21 Apr 2023
1.0.0	Basic product backlog using user story map framework	23 Mar 2023

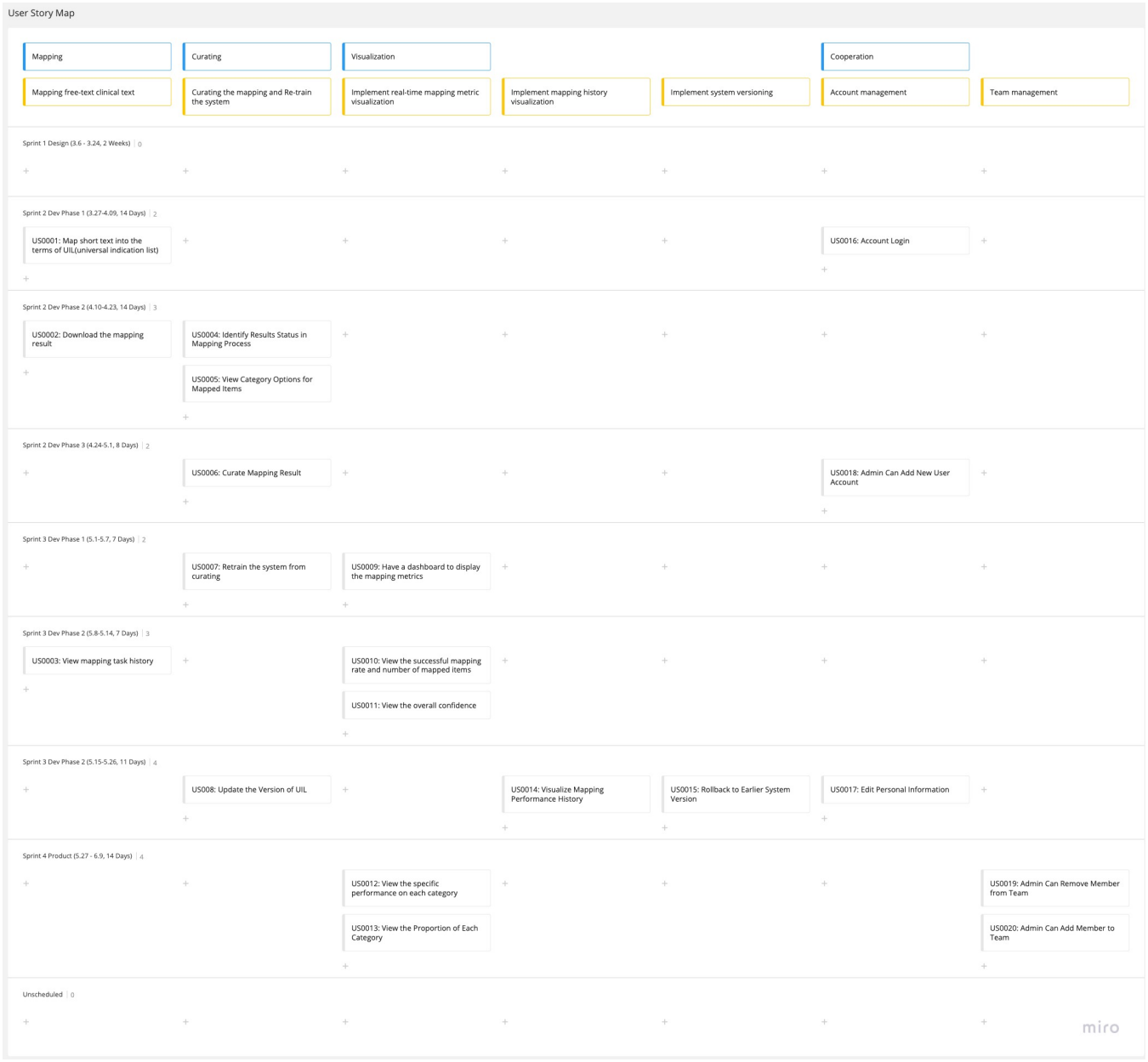
Version 2.0.0

Table View

Epic	Feature	StoryID	Priority	User	Story/Scenario	Estimate	MoSCoW Priority	Story Point
Epic 1 - Map free-text clinical text	Map short text into the terms of UIL(universal indication list)	US0001	1	Medical Researcher	**As** a medical researcher **I want to** map the short text into certain or multiple categories in a universal indication list **So that** I do not need to map the text manually	XL	Must Have	8
	Download Mapping Results	US0002	2	Medical Researcher	**As a** medical researcher **I want to** download the mapping results **So that** I can save the mapping history and view it in the future	S	Must Have	3
	View mapping task history	US0003	7	Medical Researcher	**As a** medical researcher **I want to** view my own and team members' previous mapping task **So that** I can review the mapping results and performance	L	Must	5
Epic 2 - Curate the mapping and re-train the system	Identify Results Status in Mapping Process	US0004	3	Medical Researcher (Curator)	**As a** medical researcher (curator) **I want to** easily identify the status of the mapping process for each raw text input **So that** I can quickly determine if further curation or review is needed	M	Must Have	5
	View Category Options for Mapped Items	US0005	4	Medical Researcher (Curator)	**As a** medical researcher (curator) **I want to** see a list of category options while curating the data **So that** I can choose the correct category for the incorrect mapping or unmapped items	M	Must Have	3
	Curate Mapping Result	US0006	5	Medical Researcher (Curator)	**As a** medical researcher (curator) **I want to** review and curate the failed-mapping result or mapping results of raw text to SNOMED-CT into UIL **So that** I can give feedback to the system and improve it	M	Must Have	5
	Retrain the system from curating	US0007	6	Medical Researcher (Curator)	**As a** medical researcher (curator) **I want to** re-train the system using the curated data **So that** I can make the system have a better mapping performance in the future	L	Must Have	5
	Update the Version of UIL	US0008	8	Admin user	**As an** admin user **I want to** update the UIL to the latest version **So that** I can access the most up-to-date term for curation	M	Must have	5

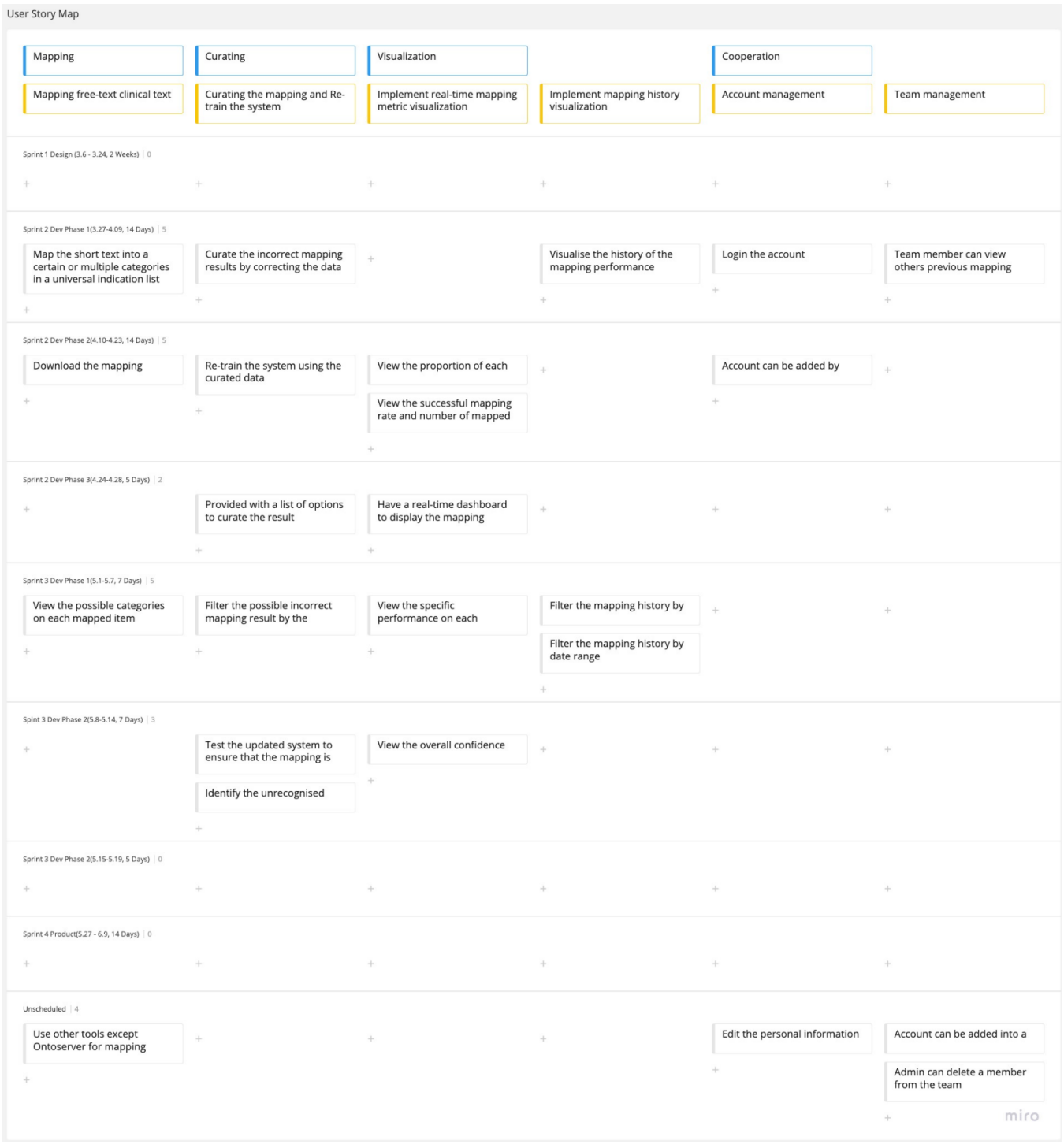
Epic 3 - Implement mapping metric visualization	Have a dashboard to display the mapping metrics	US0009	9	Medical Researcher	**As a** medical researcher **I want to** have a dashboard to display the mapping metrics for a mapping task **So that** I can review and analyze mapping results	L	Must Have	8
	View the successful mapping rate and number of mapped items	US0010	14	Medical Researcher	**As a** medical researcher **I want to** view the successful mapping rate and number of mapped items in a dashboard **So that** I can assess the performance of the system on current mapping task	S	Should Have	5
	View Overall Confidence	US0011	15	Medical Researcher	**As a** medical researcher (curator) **I want to** view the overall confidence in a dashboard **So that** I can assess the performance of the system on current mapping task	S	Should Have	3
	View Specific Performance of Each Category	US0012	16	Medical Researcher	**As a** medical researcher **I want to** view the specific performance of each category **So that** I can know the system performance among different categories	M	Could Have	5
	View Proportion of Each Category	US0013	17	Medical Researcher	**As a** medical researcher **I want to** view the proportion of each category **So that** I can know the distribution of different categories	S	Could Have	5
Epic 3 - Implement mapping history visualization	Visualize Mapping Performance History	US0014	11	Medical Researcher	**As a** medical researcher **I want to** visualize the history of the mapping performance **So that** I can measure the mapping quality	XL	Should Have	8
	Rollback to Earlier System Version	US0015	12	Admin User	**As a** admin user **I want to** rollback to earlier (default) version of the mapping system **So that** I can restore the default behaviour of the system	M	Should Have	5
Epic 4 - Manage account	Account Login	US0016	10	Medical Researcher	**As a** medical researcher **I want to** login my account **So that** I can be identified by the system for curating.	S	Must Have	3
	Edit Personal Information	US0017	18	Medical Researcher	**As a** medical researcher **I want to** edit my personal information **So that** I can update my roles and responsibilities.	S	Could Have	2
	Add New User Account	US0018	13	Admin User	**As a** medical researcher **I want to** add an account to the system **So that** I can give other people access to the system.	M	Should Have	5
Epic 5 - Manage team	Add Member to Team	US0019	19	Admin User	**As a** medical researcher **I want to** add the account to a team **So that** I can let people on the same team work together.	S	Could Have	1
	Delete Member from Team	US0020	20	Admin User	**As a** medical researcher **I want to** delete a member from a team **So that** I can remove a member's privileges in a team	S	Could Have	1

User Story Map Framework



Version 1.0.0

User Story Map Framework


User Story Map
Mapping
Curating
Visualization
Cooperation
Mapping free-text clinical text
Curating the mapping and Re-train the system
Implement real-time mapping metric visualization
Implement mapping history visualization
Account management
Team management
Sprint 1 Design (3.6 - 3.24, 2 Weeks) | 0
Sprint 2 Dev Phase 1(3.27-4.09, 14 Days) | 5
Map the short text into a certain or multiple categories in a universal indication list
Curate the incorrect mapping results by correcting the data
Visualise the history of the mapping performance
Login the account
Team member can view others previous mapping
Sprint 2 Dev Phase 2(4.10-4.23, 14 Days) | 5
Download the mapping
Re-train the system using the curated data
View the proportion of each
View the successful mapping rate and number of mapped
Account can be added by
Sprint 2 Dev Phase 3(4.24-4.28, 5 Days) | 2
Provided with a list of options to curate the result
Have a real-time dashboard to display the mapping
Sprint 3 Dev Phase 1(5.1-5.7, 7 Days) | 5
View the possible categories on each mapped item
Filter the possible incorrect mapping result by the
View the specific performance on each
Filter the mapping history by
Filter the mapping history by date range
Spint 3 Dev Phase 2(5.8-5.14, 7 Days) | 3
Test the updated system to ensure that the mapping is
View the overall confidence
Identify the unrecognised
Sprint 3 Dev Phase 2(5.15-5.19, 5 Days) | 0
Sprint 4 Product(5.27 - 6.9, 14 Days) | 0
Unscheduled | 4
Use other tools except Ontoserver for mapping
Edit the personal information
Account can be added into a
Admin can delete a member from the team
miro

Milestones

Sprint 1 - Design Sprint

There are four milestones during the development of our sprint 1, each of which took us a week to accomplish.

Sprint 1 is the design sprint, which means that the design and plan of the project are the core of this period.

Milestones	Date	Completed artifacts
Milestone1	From February 27th to March 5th	• Role assignments during the inception • Technologies being considered include deployment platform • Contact Client
Milestone2	From March 6th to March 12th	• User Stories • Initial system architecture • Functional & Non-functional Requirements
Milestone3	From March 13th to March 19th	• Product Backlog on Trello • The motivational model • Development Environment Plan • Github initialization
Milestone4	From March 20th to March 26th	• Digital Prototype • Product Backlog • Reviews of Tasks • Personas • Sprint 2 and sprint3 Roles Assignment

Sprint 2 - Development

There are four milestones during the development of our sprint 2, each of which took us a week to accomplish.

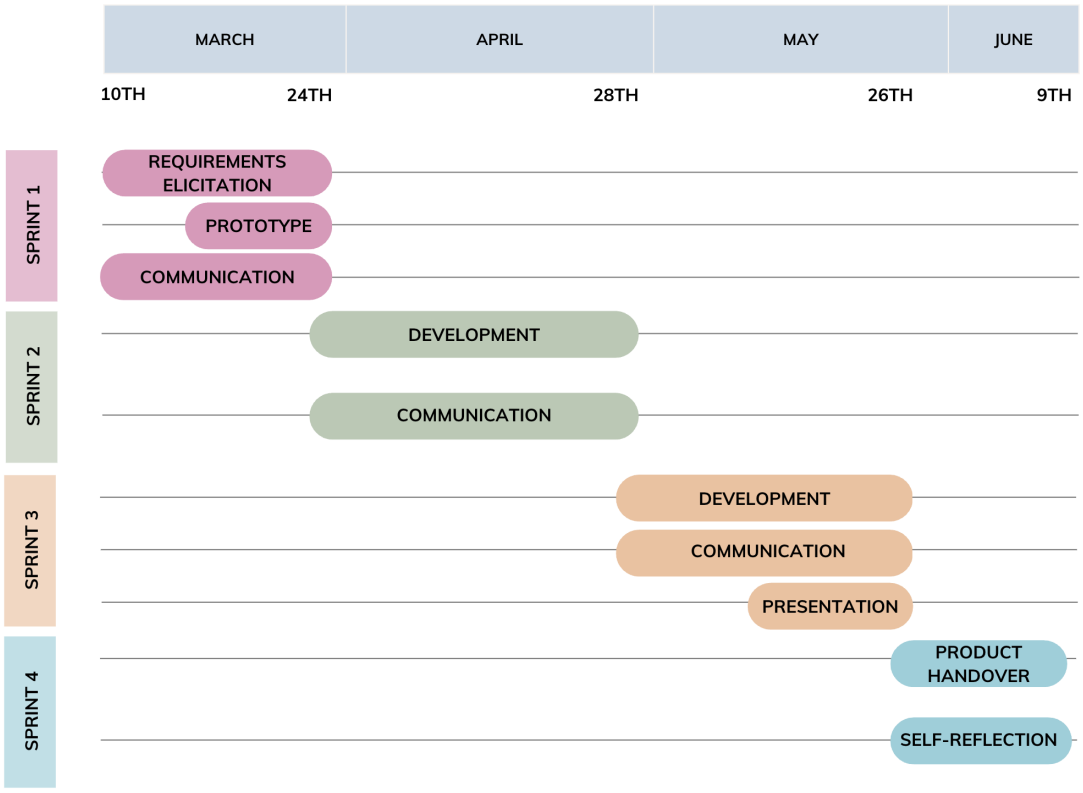
The days from April 10 to April 16 are during the Non-teaching period of the university.

Sprint 1 is the development sprint, which means that the implementation of the user stories is the core of this period.

Milestones	Date	Completed artifacts
Milestone1	From March 27 to April 2	• Account Login • Modify the sprint 1 confluence documentation • Auto deployment using scripts
Milestone2	From April 3th to April 9th	• Map short text into the terms of UIL(universal indication list) • Add on Ontoserver Licensing information on the Confluence page
Milestone3	From April 17th to April 23rd	• Identify Unrecognized Results in Mapping Process
Milestone4	From April 24th to May 1st	• Download Mapping Results • Write and organize all documents on the Confluence • Successful submission

Timeline

Title	Creator	Modified
Sprint 4 - Product	Chenyang Dong	26 May, 2023
Sprint 3 - Development	Ricardo Luo	30 Apr, 2023
Sprint 2 - Development	Ricardo Luo	30 Apr, 2023
Sprint 1 - Design Sprint	Chenyang Dong	30 Apr, 2023



Sprint 1 - Design Sprint

Title	Creator	Modified
Sprint 1 - Review	Chenyang Dong	30 Apr, 2023
Sprint 1 - Plan	Chenyang Dong	30 Apr, 2023

Sprint 1 - Plan

1 Sprint goals

- To build a substantial foundation for the project by producing critical design including user stories, product backlog, and digital prototype.
- Compose the functional and non-functional requirements of the project.
- Decompose the requirements into user stories.
- Provide a digital prototypes to client.
- Assign roles for team members.

2 Roles and Responsibility

- Role Assignment

3 Task Planning

- Contact Client
- User Stories
- Initial system architecture
- Development Environment Plan
- Summarise Functional & Non-functional Requirements
- The motivational model
- Role assignments during the inception
- Meeting minutes all the time
- Technologies being considered include deployment platform
- Personas
- Product Backlog on Trello
- Github initialization
- Reviews of Tasks on Trello
- Digital Prototype
- Sprint 2/3 Role Assignment

Sprint 1 - Review

1 Introduction

- Date: from 27 Feb 2023 to 24 Mar 2023
- Sprint Name/Number: Sprint 1 - Design Sprint
- Attendees (Team members, Product Owner, Scrum Master, stakeholders):
    - Clients: Daniel Capurro, Vlada Rozova, Mike Conway
    - Supervisor: Mauro Mello Jr
    - Team members: Kunxi (Quincy) Sun, Chenyang (Peter) Dong, Hanyi (Henry) Gao, Yue (Molly) Fei, Yulai (Ricardo) Luo

2 Sprint Goals

- Compose the functional and non-functional requirements of the project.
- Decompose the requirements into user stories and acceptance criteria.
- Provide a digital prototypes to client.
- Assign roles for team members.

3 Completed Work

Task	Complete?	Comment
Contact Client	☑	• Communication with Clients
User Stories	☑	• User Stories and Acceptance Criteria
Initial system architecture	☑	• Architecture
Development Environment Plan	☑	• Server configuration
Summarise Functional & Non-functional Requirements	☑	• Functional Requirements • Non-Functional Requirements
The motivational model	☑	• Motivational Model
Role assignments during the inception	☑	• Sprint 1 - Plan
Meeting minutes all the time	☑	• Meetings
Technologies being considered include deployment platform	☑	• Technique Detail
Personas	☑	• Personas
Product Backlog on Trello	☑	• Trello (Join by Invitation Link)

Github initialization	☑	• Github
Reviews of Tasks on Trello	☑	• Trello (Join by Invitation Link)
Digital Prototype	☑	• Digital Prototype
Sprint 2/3 Role Assignment	☑	• Sprint 2 - Plan • Sprint 3 - Plan

4 Incomplete Work

- User stories or tasks not completed during the sprint:

  None

- Explain the reasons for not completing the work and any related challenges:

  None

5 Metrics

- Total number of story points completed:

  None

- Total number of story points planned:

  None

- Velocity (completed story points / planned story points):

  None

- Burndown chart (if applicable):

  None

- There are no story points completed in the design sprint.

6 Stakeholder Feedback

6.1 Stakeholder's feedback and suggestions for improvement:

- Our client suggested that we could develop more mapping ideas and methods.
- Our supervisor suggested that we should focus more on the documentation for the first sprint instead of the code.

6.2 How the feedback will be addressed in future sprints:

- In response to our client's suggestion, we decided to research more technical papers on mapping to get inspiration
- In response to our supervisor's suggestion, we decided to add more details to the Confluence pages.

7 Next Steps

7.1 Plan for the next sprint:

- The next sprint is the development sprint, which means that we will focus more on coding.
- The planned user stories which will be implemented in sprint 2 are:

US0001, US0002, US0004, US0006, US0007, US0019 and US0021

7.2 Adjustments needed based on the sprint review feedback:

- We planned to read more papers for finding a more accurate and efficient mapping approach.
- And we will continue to do detailed documentation.

7.3 Schedule the next sprint planning meeting:

- We plan the sprint 2 planning meeting on 24 Mar 2023.

8 Closing Remarks

8.1 Final thoughts or comments on the sprint and the review from team members and stakeholders:

- **Team members:**

  Teamwork is vital and we need more meetings to thoroughly understand each other's perspectives. At the moment, we are working well together as a team and we hope that the sprint will be the same afterward!

Sprint 2 - Development

Title	Creator	Modified
Sprint 2 - Review	Chenyang Dong	20 May, 2023
Sprint 2 - Plan	Chenyang Dong	30 Apr, 2023
Sprint 2 - Backlog	Chenyang Dong	30 Apr, 2023

Sprint 2 - Plan

1 Sprint Goals

- Enable efficient and accurate mapping of short text into the terms of the Universal Indication List (UIL) by developing a mapping system, allowing for account login and management, and providing category options for mapped items

2 Roles and Responsibility

- Role Assignment

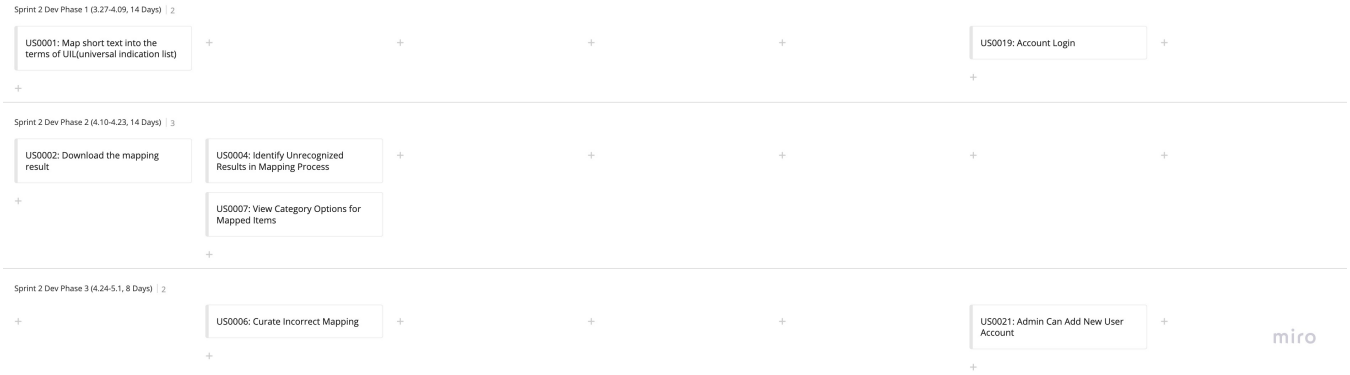
3 Planning

3.1 Techniques to use

- Backend: Python Flask, Nginx, Gunicorn, Docker, Ansible
- Frontend: React, HTML, JS, CSS, Ant Design
- Database: MongoDB
- Mapping tool: Ontoserver and other potential alternatives
- Technique Detail

3.2 Backlog with Phase Plan

> *Table view of backlog in Sprint 3 - Review*



4 Infrastructure for Continuous Integration

- **Team Development Cloud Server configuration**

Sprint 2 - Backlog

Table View

Feature	StoryID	Story/Scenario	MoSCoW Priority	Story Point	Subtask [Estimate]	Status	Assigned To	Due Date	Comment
Map short text into the terms of UIL(universal indication list)	US0001	**As** a medical researcher **I want to** map the short text into certain or multiple categories in a universal indication list **So that** I do not need to map the text manually	Must Have	8	☑ Implement the single text mapping user interface [1 hour]	DONE	Hanyi Gao	09 Apr 2023	Due to complexity of this function, change of mapping requirements and and recent notice that the deployment environment does not allow constant use of external endpoints, our progress was impacted. New approach is going to be experimented to combine with existing mapping tool to fulfil the requirements in the next sprint.
					☑ Implement the CSV file mapping user interface with a browse button [1 hour]	DONE	Hanyi Gao	09 Apr 2023	
					☑ Create a function for submitting short text and retrieving mapped categories using the Ontoserver API [1 hour]	DONE	Hanyi Gao	09 Apr 2023	
					☐ Map the short text into certain or multiple categories in SNOMED CT [3 hours]	IN PROGRESS	Chenyang Dong	28 Apr 2023	
					☐ Further match the mapping result from SNOMED CT into UIL [2 hours]	IN PROGRESS	Chenyang Dong	28 Apr 2023	
					☐ Preprocess the raw text [1 hour]	TO DO	Yue Fei	TBD	The preprocessing of the raw text was pending since many approaches contain such strategies. Unique preprocessing guide for clinical terms
Download Mapping Results	US0002	**As a** medical researcher **I want to** download the mapping results **So that** I can save the mapping history and view it in the future	Must Have	3	☑ Develop the 'Download' button user interface for the input CSV file [1 hour]	DONE	Hanyi Gao	23 Apr 2023	
					☐ Develop a secure API endpoint to retrieve the mapping results for the input CSV file [3 hours]	TO DO	Chenyang Dong	23 Apr 2023	Delayed to next sprint due to being blocked by mapping result.
					☐ Create API endpoint for retrieving mapping results [2 hours]	TO DO	Chenyang Dong	23 Apr 2023	

					☐ Connect the 'Download' button to the API endpoint to trigger the file download [1 hour]	TO DO	Hanyi Gao	23 Apr 2023	
Identify Results Status in Mapping Process	US0004	**As a** medical researcher (curator) **I want to** easily identify the status of the mapping process for each raw text input **So that** I can quickly determine if further curation or review is needed	Must Have	5	☑ Implement the unrecognized results user interface by adding mapping status for the result [2 hours]	DONE	Hanyi Gao	23 Apr 2023	Modified after change of output requirements of four different scenarios. 2023-04-14 Change of Mapping Requirements
Curate Mapping Result	US0006	**As a** medical researcher (curator) **I want to** review and curate the failed-mapping result or mapping results of raw text to SNOMED-CT into UIL **So that** I can give feedback to the system and improve it	Must Have	5	☑ Implement the curation user interface [1 hour]	DONE	Hanyi Gao	28 Apr 2023	
					☐ Create API endpoint for submitting curated data [1 hour]	TO DO	Chenyang Dong	28 Apr 2023	Approach to make curation improve the system TBD depends on the method finally used
View Category Options for Mapped Items	US0005	**As a** medical researcher (curator) **I want to** see a list of category options while curating the data **So that** I can choose the correct category for the incorrect mapping or unmapped items	Must Have	3	☑ Develop the category options dropdown user interface [1 hour]	DONE	Hanyi Gao	23 Apr 2023	Different level of category options according to UIL.
					☑ Add functionality to handle medical researchers' category selections [2 hours]	DONE	Hanyi Gao	23 Apr 2023	After Selecting from the UIL, the mapping status will become 'Reviewed'
Account Login	US0016	**As a** medical researcher **I want to** login my account **So that** I can be identified by the system for curating.	Must Have	3	☑ Develop the login user interface based on the approved design [1 hour]	DONE	Hanyi Gao	09 Apr 2023	With email and password
					☑ Develop a secure API endpoint for user authentication [1 hour]	DONE	KUNXI SUN	09 Apr 2023	
					☑ Implement input validation on the front-end [1 hour]	DONE	Hanyi Gao	09 Apr 2023	
					☑ Connect front-end login to the authentication API [2 hours]	DONE	Hanyi Gao	09 Apr 2023	
Add New User Account	US0018	**As an** admin user **I want to** add an account to the system **So that** I can give other people access to the system	Should Have	5	☐ Develop the 'Add New User' user interface [1 hour]	IN PROGRESS	Ricardo Luo	28 Apr 2023	
					☐ Create API endpoint for sending email invitations [1 hour]	IN PROGRESS	Ricardo Luo	28 Apr 2023	

					☐ Connect the 'Add New User' UI to the email invitation API [1 hour]	TO DO	Ricardo Luo	28 Apr 2023	
					☐ Implement email-sending functionality [2 hours]	IN PROGRESS	Ricardo Luo	28 Apr 2023	

Sprint 2 - Review

This sprint review note is the discussion result based on the Sprint 2 Review Meeting.

1 Introduction

- Date: from 27 Mar 2023 to 01 May 2023
- Sprint Name/Number: Sprint 2 - Develop sprint
- Attendees (Team members, Product Owner, Scrum Master, stakeholders):
    - Clients: Daniel Capurro, Vlada Rozova, Mike Conway
    - Supervisor: Mauro Mello Jr
    - Team members: Kunxi (Quincy) Sun, Chenyang (Peter) Dong, Hanyi (Henry) Gao, Yue (Molly) Fei, Yulai (Ricardo) Luo

2 Sprint Goals

- Enable efficient and accurate mapping of short text into the terms of the Universal Indication List (UIL) by developing a mapping system, allowing for account login and management, and providing category options for mapped items

3 Completed Work

- Finished user stories:
    - US0004: Identify Results Status in Mapping Process
    - US0005: View Category Options for Mapped Items
    - US0016: Account login
    - US0018: Add New User Account
- CI/CD: Script to auto-deploy services: gateway, map, auth, web, MongoDB and the ChatGPT code review.
- System testing for all finished features

4 Incomplete Work

- Incomplete user stories:
    - US0001: Map short text into the terms of UIL(universal indication list)

      Reason: Due to the complexity of this function, change of mapping requirements and recent notice that the deployment environment does not allow constant use of external endpoints, our progress was impacted. A new approach is going to be experimented with to combine with the existing mapping tool to fulfill the requirements in the next sprint. Also, the deployment on SRE is not possible at this stage.

    - US0002: Download the mapping results

      Reason: Delayed to the next sprint due to being blocked by the mapping approach.

    - US0006: Curate Mapping Result

      Reason: Delayed to the next sprint as well due to being blocked by the mapping approach.

5 Metrics

- Total number of story points completed: 16
- Total number of story points planned: 30
- Velocity (completed story points / planned story points): 0.53

6 Stakeholder Feedback

- Stakeholder's feedback and suggestions for improvement:
    - Clients suggest that we can develop our own algorithm as well as look into MedCAT since the requirements change.
    - Clients suggest that we could catch up on the current sprint and adjust to the next sprint as soon as possible.
    - The supervisor suggest that we could create a more organised timeline and urge each member to complete the corresponding tasks on time.
- How the feedback will be addressed in future sprints:
    - We accept the client's advice and plan to develop our own algorithm while also continuing to develop MedCAT.
    - We aim to complete our delayed work as soon as possible and will incorporate risk management into the upcoming sprints.

7 Reflection

- Over-relying on one approach and wasting too much time on Ontoserver(The mapping tool) cause the following issues:
    - Deploying Ontoserver in Australia and out of Australia is different, but the production environment will be in Australia so figuring out how to deploy Ontoserver in out of Australia is quite a waste of effort.
    - One of the team members is in China which makes him unable to deploy Ontoserver locally.
    - Team didn't realise the risk of disability by using third-party tools such as Ontoserver.
- Lose communication with clients about the SRE(production environment).
- Could have done a better job with risk management, such as tracking the production environment, and managing risk of using Ontoserver as the only mapping tool.
- Tasks on Trello should be assigned to the person with a due date ---- Scrum master could manage the project progress by tracking the team member's tasks.
- Team members unfamiliar with the techniques used in this project ---- can be solved to have more pair programming on Zoom.
- Lack of regular weekly team meetings ---- can be solved to hold regular team meetings.

8 Decision and Next Steps

- Better risk management:
    - Currently we only specify the risk with description and progress update on Trello with a due date
    - In the following sprint, risk response strategy could be documented to help with responding the risk
- Regular internal team meeting, everyday on 9pm
- Regular pair programming: can be held by all team members, and can be held by 2-3 people
- Better task management: task assign to a person with due dates and scrum master track team member's task
- Plan for the next sprint:
    - Finish the core user stories by using MedCAT or a self-implemented algorithm at the beginning of 15 days.
    - Implement GitHub Actions to do CI/CD.
    - Decide to use MedCAT or a self-implemented mapping algorithm.
- Adjustments needed based on the sprint review feedback:
    - The team organises a meeting for about one to two hours every day to implement tasks together
    - Every team member spends more time on the project
- Move on sprint 3 planning meeting after sprint 2 review meeting

9 Closing Remarks

9.1 Final thoughts or comments on the sprint and the review from team members and stakeholders:

- **Team members:**

  Our team believes that we urgently need to accelerate our progress. Due to some technical difficulties and unexpected changes, our sprint 2 is not progressing as expected. In our next sprint, we need to keep the lessons of sprint 2 in mind. We need to prepare multiple alternatives. If a development solution fails, we need to start a new one soon to save time.

- **Stakeholders:**

  Although our team spent a lot of time trying out the mapping approach in sprint 2, our current product is still good and we are not behind in our development.

Sprint 3 - Development

Title	Creator	Modified
Sprint 3 - Backlog	Chenyang Dong	30 May, 2023
Sprint 3 - Plan	Chenyang Dong	30 Apr, 2023

Sprint 3 - Plan

1 Sprint Goal

- Complete the remaining user stories from sprint 2
- Finish major user stories in the first 15 days (mapping, curate, visualise and download)
- Additionally, if possible, implement optional features in product backlogs.

2 Roles and Responsibility

- Role Assignment

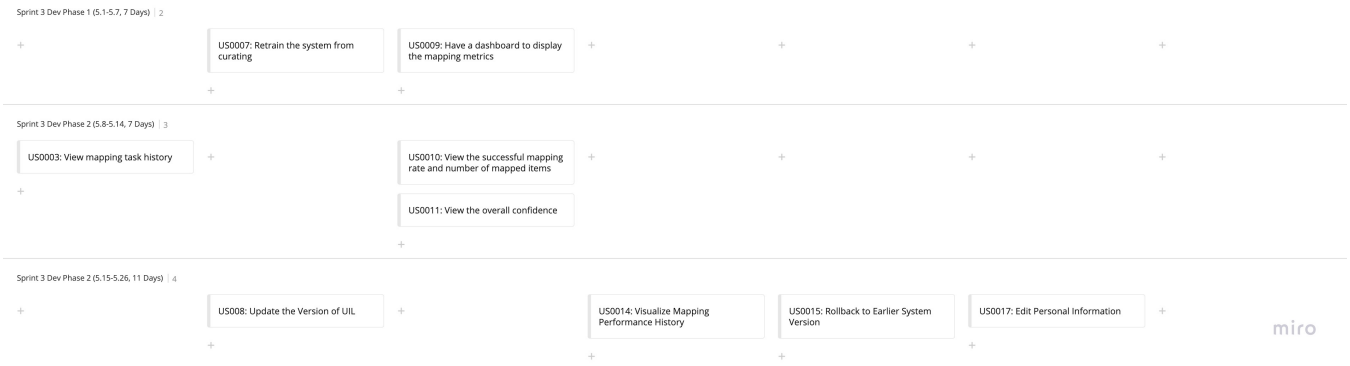
3 Planning

3.1 Tasks details

- Implement user stories: Use the first 15 days of the next sprint to implement the following user stories
    - Mapping function: Finish these core user stories by using MedCAT or a self-implemented algorithm.
    - Visualisation: Follow the requirements from the client to implement the front-end pages.
    - Curate:
- Testing: Use the rest of the sprint to do testing and implement additional trivial user stories.
- CI/CD: implement GitHub Actions to do CI/CD (combine with the current Ansible script which is used for handover-deploy)

3.2 User stories map

> *Table view of backlog in Sprint 3 - Backlog*



3.3 Addition tasks/rules

- Better risk management:
    - Currently, we only specify the risk with a description and progress update on Trello with a due date
    - In the following sprint, the risk response strategy could be documented to help with responding to the risk
- Regular internal team meeting, every day at 9 pm
- Regular pair programming: can be held by all team members, and can be held by 2-3 people
- Better task management: task assigned to a person with due dates and scrum master track team member's task

Sprint 3 - Backlog

Table View

Feature	StoryID	Story/Scenario	MoSCoW Priority	Story Point	Subtask [Estimate]	Status	Assigned To	Due Date	Comment
Map short text into the terms of UIL(universal indication list) [From Sprint 2]	US0001	**As** a medical researcher **I want to** map the short text into certain or multiple categories in a universal indication list **So that** I do not need to map the text manually	Must Have	8	☑ Preprocess the raw text [1 hour]	DONE	Yue Fei	07 May 2023	[Not required since MedCAT is used] Unique preprocessing guide for clinical terms
					☑ Map the short text into certain or multiple categories in SNOMED CT [3 hours]	DONE	Chenyang Dong	07 May 2023	Custom algorithm and MedCAT implemented at the same time
					☑ Further match the mapping result from SNOMED CT into UIL [2 hours]	DONE	KUNXI SUN	14 May 2023	Based on the current assumption from the client, this is easy to implement. However, by observing the UIL, the assumption might change in the future when it is confirmed.
Download Mapping Results [From Sprint 2]	US0002	**As a** medical researcher **I want to** download the mapping results **So that** I can save the mapping history and view it in the future	Must Have	3	☑ Develop a secure API endpoint to retrieve the mapping results for the input CSV file [3 hours]	DONE	KUNXI SUN	14 May 2023	
					☑ Create API endpoint for retrieving mapping results [2 hours]	DONE	Chenyang Dong	14 May 2023	
					☑ Connect the 'Download' button to the API endpoint to trigger the file download [1 hour]	DONE	Hanyi Gao	14 May 2023	
View mapping task history	US0003	**As a** medical researcher **I want to** view my own and team members' previous mapping task **So that** I can review the mapping results and performance	Must	5	☑ Create a backend API for fetching mapping task history [3 hours]	DONE	KUNXI SUN	14 May 2023	
					☑ Develop a frontend component for displaying mapping task history (card style) [1 hour]	DONE	Hanyi Gao	14 May 2023	
					☑ Integrate the frontend component with the backend API [2 hours]	DONE	Hanyi Gao	14 May 2023	

					☑ Implement search and filtering functionalities (by date range or user) for mapping task history [3 hours]	DONE	KUNXI SUN	26 May 2023	
Curate Mapping Result [From Sprint 2]	US0006	**As a** medical researcher (curator) **I want to** review and curate the failed-mapping result or mapping results of raw text to SNOMED-CT into UIL **So that** I can give feedback to the system and improve it	Must Have	5	☑ Create a backend API endpoint to gather and store curated data for submitting [1 hour]	DONE	Chenyang Dong	14 May 2023	
					☑ Create a function to update the downloadable data with the curated mapping results [1 hours]	DONE	Chenyang Dong	22 May 2023	
Retrain the system from curating	US0007	**As a** medical researcher (curator) **I want to** re-train the system using the curated data **So that** I can make the system have a better mapping performance in the future	Must Have	5	☑ Collect curated data via the API created in US0006 [1 hour]	DONE	KUNXI SUN	14 May 2023	
					☑ Create a backend API for retraining the system [2 hours]	DONE	KUNXI SUN	19 May 2023	Approach to make curation improve the system TBD depends on the method finally used
					☑ Adapt the system's training algorithm to incorporate the curated data [3 hours]	DONE	KUNXI SUN	19 Apr 2023	
Update the Version of UIL	US0008	**As an** admin user **I want to** update the UIL to the latest version **So that** I can access the most up-to-date term for curation	Must Have	5	☑ Create API points to create a UIL list, group, and category [3 hours]	DONE	KUNXI SUN	07 May 2023	
					☑ Develop the interface to allow uploading a format-compatible file of the new UIL version [1 hour]	DONE	Hanyi Gao	14 May 2023	
					☑ Develop a version comparison tool [3 hours]	DONE	Chenyang Dong	26 May 2023	Keep all versions for user to choose
					☑ Communicate the UIL version update or display the version of UIL on the interface [2 hours]	DONE	Chenyang Dong	19 May 2023	To notify the user version update on UIL (Will be a version display)
Have a dashboard to display the mapping metrics	US0009	**As a** medical researcher **I want to** have a dashboard to display the mapping metrics for a mapping task **So that** I can review and analyze mapping results	Must Have	8	☑ Develop the interface of data visualization components [2 hours]	DONE	Ricardo Luo	07 May 2023	

					☑ Integrate the dashboard with the mapping system [3 hours]	DONE	Chenyang Dong	14 May 2023	
View the successful mapping rate and number of mapped items	US0010	**As a** medical researcher **I want to** view the successful mapping rate and number of mapped items in a dashboard **So that** I can assess the performance of the system on the current mapping task	Should Have	3	☑ Determine the calculations for the successful mapping rate and the number of mapped items [1 hour]	DONE	KUNXI SUN	14 May 2023	Component of the dashboard in US0010
					☑ Develop the data visualization component for successful mapping rate and number of mapped items [2 hours]	DONE	Ricardo Luo	14 May 2023	
~~View the overall confidence~~	~~US0011~~	~~**As a** medical researcher (curator)~~ ~~**I want to** view the overall confidence in a dashboard~~ ~~**So that** I can assess the performance of the system on current mapping task~~	~~Should Have~~	~~5~~	☐ ~~Determine the calculation or aggregation method for the overall confidence [1 hour]~~	TO DO	~~TBD~~	14 May 2023	~~Component of the dashboard in US0010~~ ~~(Optional, now considering to view mapping results on SCT or UIL)~~
					☐ ~~Develop the data visualization component for the overall confidence metric [2 hours]~~	TO DO	~~TBD~~	14 May 2023	
Visualize Mapping Performance History	US0014	**As a** medical researcher **I want to** visualize the history of the mapping performance **So that** I can measure the mapping quality	Must Have	8	☑ Create a backend API for fetching overall mapping performance data [2 hours]	DONE	KUNXI SUN	19 May 2023	Component required to be further confirmed
					☑ Develop a data visualization component for overall mapping performance [2 hours]	DONE	Yue Fei	19 May 2023	
					☑ Connect the overall mapping performance visualization to the backend API [2 hours]	DONE	Yue Fei	26 May 2023	
~~Rollback to Earlier System Version~~	~~US0015~~	~~**As a** admin user~~ ~~**I want to** rollback to earlier (default) version of the mapping system~~ ~~**So that** I can restore the default behaviour of the system~~	~~Should Have~~	~~5~~	☐ ~~Identify and store the default version of the mapping system [2 hours]~~	TO DO	~~TBD~~	19 May 2023	~~From requirement of clients, the system should be able to restore to default version.~~
					☐ ~~Create a backend API for triggering the rollback process [2 hours]~~	TO DO	~~TBD~~	26 May 2023	

					☐ ~~Develop a frontend component for initiating the rollback process [1 hour]~~	TO DO	~~TBD~~	19 May 2023	
					☐ ~~Implement user authentication and authorization for admin users [2 hours]~~	TO DO	~~TBD~~	26 May 2023	
Edit Personal Information	US0017	**As a** medical researcher **I want to** edit my personal information **So that** I can update my roles and responsibilities.	Could Have	2	☑ Develop the profile interface based on the approved design [2 hours]	DONE	Ricardo Luo	22 May 2023	
					☑ Develop a secure API endpoint for user profile information [1 hour]	DONE	KUNXI SUN	22 May 2023	
					☑ Develop the edit mode interface for profile update [1 hour]	DONE	Ricardo Luo	22 May 2023	
					☑ Develop a secure API endpoint for updating profile information [1 hour]	DONE	KUNXI SUN	22 May 2023	
Add New User Account [From Sprint 2]	US0018	**As a** medical researcher **I want to** add an account to the system **So that** I can give other people access to the system	Should Have	5	☑ Develop the 'Member' page user interface [1 hour]	DONE	Ricardo Luo	07 May 2023	Changed to copying an invitation link on 'Member' page for members to register.
					☑ Create API endpoint for registering [1 hour]	DONE	Chenyang Dong	22 May 2023	
					☑ Implement registering functionality [3 hours]	DONE	Ricardo Luo	22 May 2023	
Add Member to Team	US0019	**As a** medical researcher **I want to** add the account to a team **So that** I can let people on the same team work together.	Could Have	1	☑ Develop the 'Add Member' button on the member page [1 hour]	DONE	Hanyi Gao	22 May 2023	
					☑ Create API endpoint for adding new members [1 hour]	DONE	Chenyang Dong	22 May 2023	
					☑ Create the new member page [1 hour]	DONE	Ricardo Luo	22 May 2023	
					☑ Implement adding member functionality [1 hour]	DONE	Ricardo Luo	22 May 2023	

Delete Member from Team	US0020	**As a** medical researcher **I want to** delete a member from a team **So that** I can remove a member's privileges in a team	Could Have	1	☑ Develop the 'Delete Member' button on the member page [1 hour]	DONE	Hanyi Gao	24 May 2023	
					☑ Develop a secure API endpoint for deleting a member from a team [1 hour]	DONE	KUNXI SUN	23 May 2023	

Sprint 3 - Review

This sprint review note is the discussion result based on the Sprint 3 Review Meeting.

1 Introduction

- Date: from 02 May 2023 to 01 Jun 2023
- Sprint Name/Number: Sprint 3 - Develop sprint
- Attendees (Team members, Product Owner, Scrum Master, stakeholders):
    - Clients: Daniel Capurro, Vlada Rozova, Mike Conway
    - Supervisor: Mauro Mello Jr
    - Team members: Kunxi (Quincy) Sun, Chenyang (Peter) Dong, Hanyi (Henry) Gao, Yue (Molly) Fei, Yulai (Ricardo) Luo

2 Sprint Goals

- Produce a fully functional website platform
- Implement all user stories
- Meet all the requirements of our clients

3 Completed Work

- Finished user stories:
    - US0001: Map short text into the terms of UIL(universal indication list)
    - US0002: Download Mapping Results
    - US0003: View mapping task history
    - US0006: Curate Mapping Result
    - US0007: Retrain the system from curating
    - US0008: Update the Version of UIL
    - US0009: Have a dashboard to display the mapping metrics
    - US0010: View the successful mapping rate and number of mapped items
    - US0014: Visualize Mapping Performance History
    - US0017: Edit Personal Information
    - US0018: Add New User Account
    - US0019: Add Member to Team
    - US0020: Delete Member from Team
- User-friendly web interface
- Highly scalable microservices architecture
- Fast and accurate mapping based on MedCAT
- Continuous integration via GitHub Actions
- Follow the SOLID software principles
- APIs including Authentication Service, Center Service and Mapper Service
- Tests including Unit Test, Functional Test, Integration Test and System Test
- Discard user stories: US0011, US0012, US0013 and US0015. Reason: through communication with our clients, those user stories are out of scope and do not meet the needs of our clients. Therefore, they were carefully discarded by our team.
- Product presentation to the clients.

4 Incomplete Work

- Blocked user stories:
    - US0011: View the overall confidence
    - US0012: View Specific Performance of Each Category
    - US0013: View the proportion of each category
    - US0015: Rollback to Earlier System Version
- Unit testing

  We use API tools such as Apifox to complete the unit testing. If we have time later, we will implement unit testing using code.

5 Metrics

- Total number of story points completed: 59
- Total number of story points planned: 59
- Velocity (completed story points / planned story points): 1

6 Stakeholder Feedback

- Stakeholder's feedback and suggestions for improvement:
    - Our clients felt that our product met most of the expectations and requirements. They will test and compare the performance of our system later. (Further acceptance testing required in Sprint 4)
    - Our supervisor felt that our team was making a lot of effort and progress, which satisfied him.
- How the feedback will be addressed in future sprints:
    - We will review the system to see if there is any issue to be fixed and any room for improvement.

7 Reflection

- Could have done a better job on automatic testing via GitHub Actions while releasing the product.
- Could have done a better job on task management.
- The agile process could be improved – continuously deploy a complete system, but the client did not offer a production server, more detail is available at No Continuous Deployment.

8 Decision and Next Steps

- Therefore, the system will not be deployed. Due to the complexities and challenges in advancing the deployment of the Secure Research Environment, and the intricate approval processes involved, we have mutually agreed with our clients to abstain from final deployment at this stage.
- Format code and release the branch
- Release documentation
- Write Docker compose scripts for the next sprint

9 Closing Remarks

9.1 Final thoughts or comments on the sprint and the review from team members and stakeholders:

- **Team members:**

  Our team believes that we made significant progress in this sprint. We achieved most of our user stories and made a product that satisfied our customers. The next thing to improve is adding more tests and how to manage with deployment when it is impossible at this stage.

- **Stakeholders:**

  The stakeholders felt that our product met most of their needs.

Sprint 4 - Product

Title	Creator	Modified
Sprint 4 - Plan	KUNXI SUN	01 Jun, 2023

Sprint 4 - Plan

1 Sprint Goal

- Finish all the testing
- Deploy the final product on the client's server
- Do handover with clients and the next team

2 Roles and Responsibility

- Role Assignment

3 Planning

3.1 Tasks details

- Testing: Finish acceptance testing of our product and the final system testing on the client's server
- Handover:
    - write detailed release documents
    - write a docker-compose file to set up the system automatically
    - write detailed development documentation for the next team
    - write help documents for users
    - add detailed comments to the code

3.2 User Stories Map

- No user stories will be implemented in Sprint 4(Deploy/Handover Sprint)

3.3 Addition Tasks/Rules

- Fix bugs if found while deploying the product.
- Improve code based on code review.

4 Future work

- Redesign and implement cancelled user stories in previous sprints. These user stories are beyond the scope of the project, but they still have value for implementation.
    - US0012: View Specific Performance of Each Category
    - US0013: View the proportion of each category
    - US0015: Rollback to Earlier System Version
- Regarding the mapping algorithm, we will look at ways to improve its accuracy, such as how to train Medcat's model with the latest sct.

Handover

- Development Overview

- Release Notes
- User Guide

Development Overview

About

This document is a handover report for another development team to take over the project if needed.

Features

The main goal of the system we implemented is following below:

- Map the clinical text to standardised clinical terms (SNOMED CT or UIL).
- Visualising the system performance and task results, thereby allowing users to understand and optimise their workflows.
- Users can curate and retain the system for better efficiency and reliability.
- Download the mapping results for further analysis and documentation.
- User authentication and authorization to ensure the security of the system and role-based control.

User stories

Complete User Stories

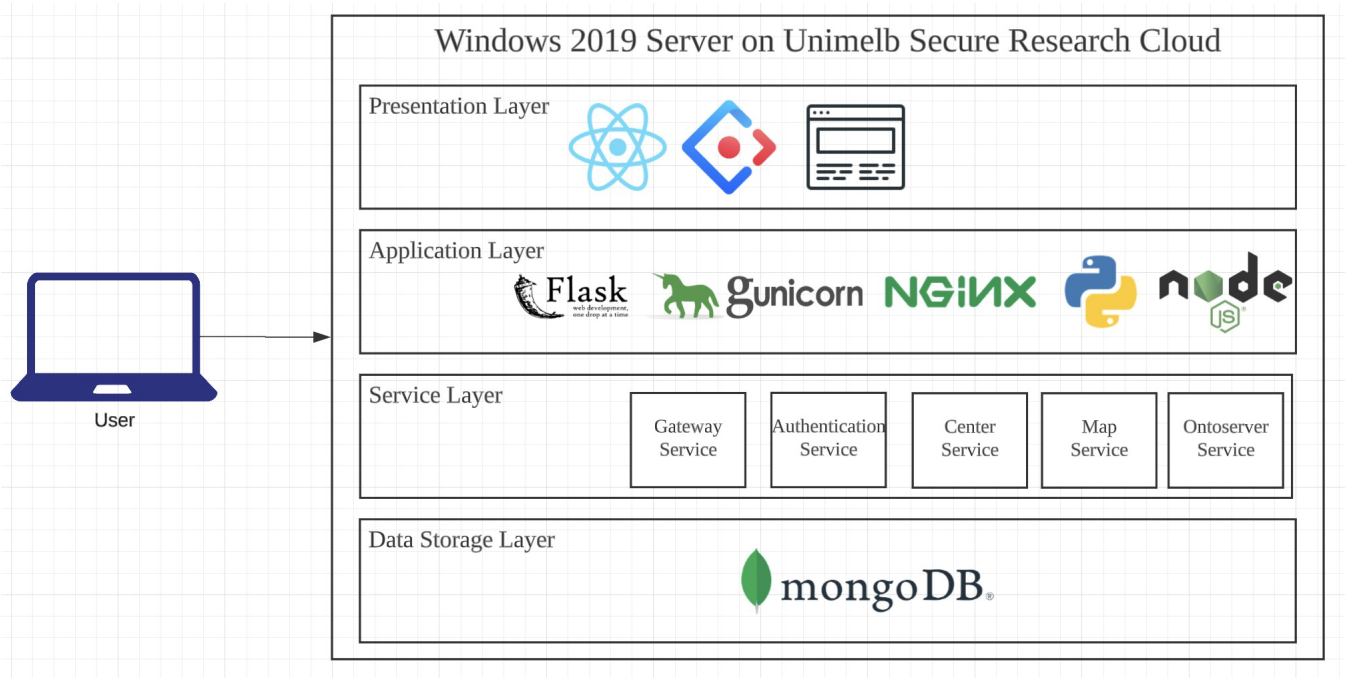
User Story ID	User Story	Story Point
US0001	Map short text into the terms of UIL(universal indication list)	8
US0002	Download the mapping result	3
US0003	View mapping task history	5
US0004	Identify Results Status in Mapping Process	5
US0005	View Category Options for Mapped Items	3
US0006	Curate Mapping Result	5
US0007	Retrain the system from curating	5
US0008	Update the Version of UIL	5
US0009	Have a dashboard to display the mapping metrics	8
US0010	View the successful mapping rate and the number of mapped items	5
US0014	Visualize Mapping Performance History	8
US0016	Account Login	3
US0017	Edit Personal Information	2
US0019	Add Member to Team	1
US0020	Remove Member from Team	1

Incomplete User Stories

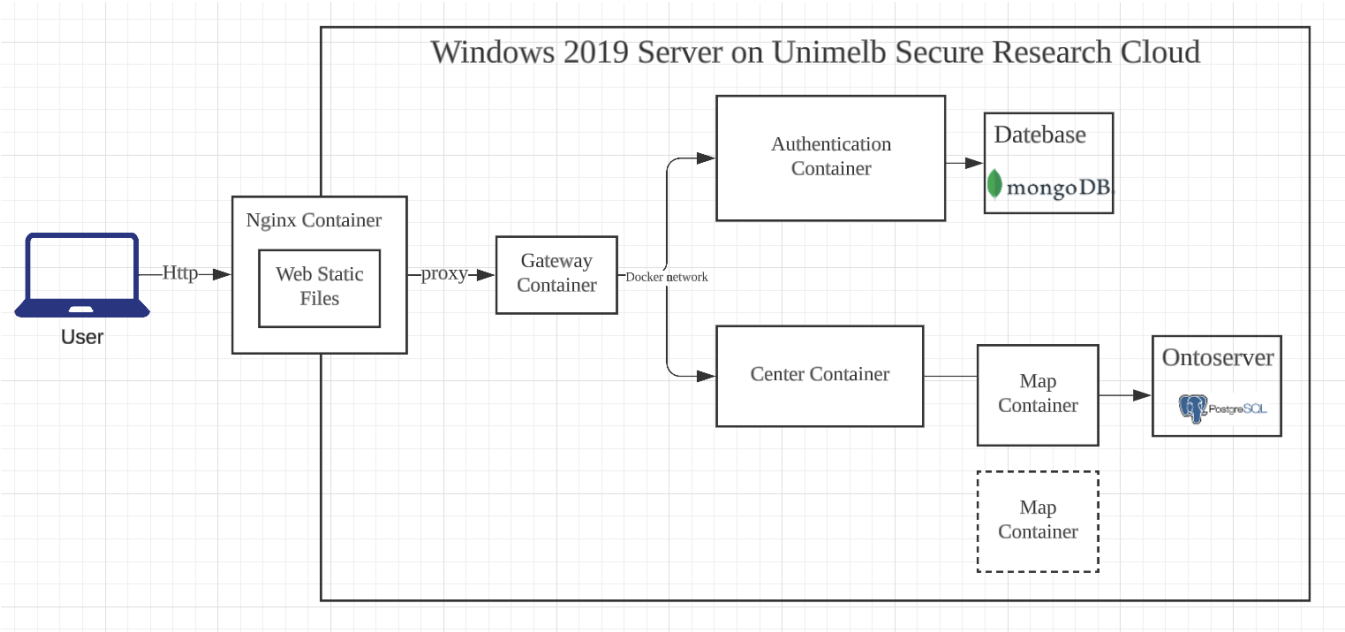
User Story ID	User Story	Story Point
US0011	View the overall confidence	3
US0012	View the Specific Performance On Each Category	5
US0013	View the Proportion of Each Category	5
US0015	Rollback to Earlier System Version	5
US0018	Add New User Account	5

System Architecture

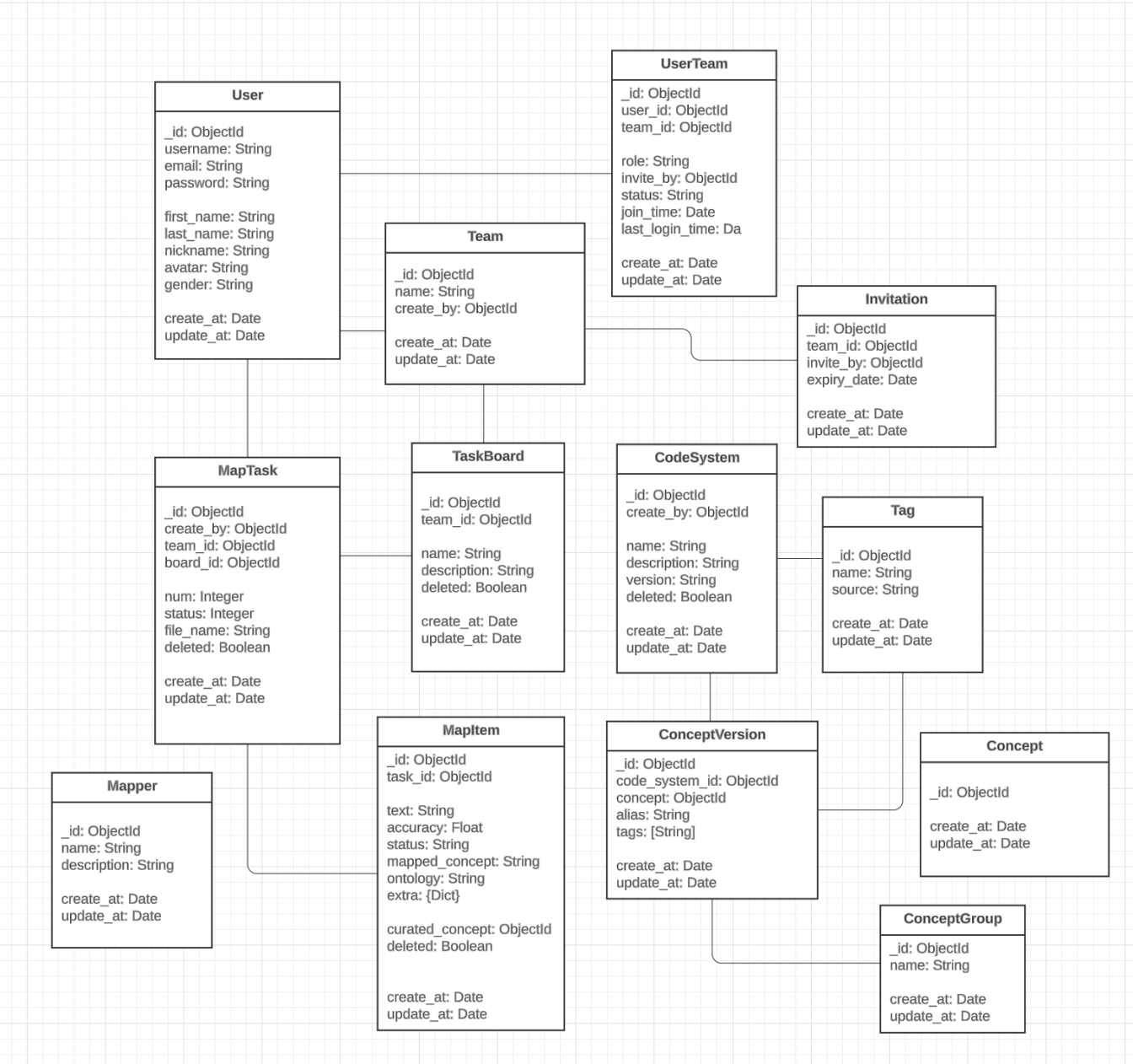
System Diagram



Container Diagram



Data models



Code Structure

Frontend Code Structure

Frontend code follows a structured organization under the 'src/di-web/src' directory. Here is a brief description of each folder:

- components: contains reusable UI components used throughout the project. These components encapsulate specific functionalities and can be easily integrated into different pages or sections.
- mocks: stores mock data files used for testing or simulating API responses during development. These mock files aid in frontend development when the backend APIs are not yet available or fully implemented. **Mock Service Worker** is used as the tool to mock API
- modules: stores all the pages
    - CodeSystem
    - Dashboard
    - Login
    - Main
    - MappingResult
    - Profile
    - Register
    - TeamProfile

- store: stores the state management code. It contains the files responsible for defining and managing the application state using the **Zustand** librar y. Zustand is used as the state management tool in this project to efficiently handle application-level states.
- utils: contains utility functions and helper modules that provide common functionalities used throughout the project. These utilities are used to abstract common operations and promote code reuse.

The following libraries and tools are crucial for the development and maintenance of the frontend project:

- Tailwind CSS: the chosen CSS framework for this project. It is highly recommended to install **Tailwind CSS IntelliSense** extension in VSCode if VSCode is used as the IDE
- axios: a JavaScript library used for making HTTP requests
- ahooks: provides various hooks to simplify development. In this project, in particular, **useRequest** is used to handle HTTP responses more easily
- Zustand: a state management tool
- msw: an API mocking library that uses Service Worker API to intercept actual requests.

Backend Code Structure

Backend code follows a structured organized under a **microservice sturcture**. It has the following four directories, and each of them is deployed as a Docker container in the production environment:

- 'src/di-gateway': It acts as a routing of incoming requests to the appropriate microservice based on the request path. This provides a single entry point into the system, which can simplify client-side communication logic and provides an extra layer of abstraction and isolation for the microservices.
- 'src/di-auth': It manages user credentials and handles authentication and authorization. It is responsible for generating JWT tokens, validating user credentials, managing user profiles, manage team members' operations.
- 'src/di-center': This directory contains the functions of code system management, task management, visualizations and communication with the mapper.
- 'src/di-map': This directory contains the functions of mapping and curating.

The following libraries and tools are crucial for the development and maintenance of the backend project:

- gunicorn: Python WSGI HTTP server.
- flask: Micro web framework for defining API endpoints.
- Flask-RESTful: Flask extension for building REST APIs.
- requests: Python library for making HTTP requests.
- Flask-Pymongo: Flask extension that provides PyMongo support.
- marshmallow: Library for complex data serialization.
- mongoengine: Object-Document Mapper for MongoDB.
- python-dateutil: Extension to the Python datetime module.
- pandas: Library for data analysis and manipulation.
- xlsxwriter: Python module for writing Excel 2007+ XLSX files.
- openpyxl: Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files.
- medcat: Medical Concept Annotation Tool, a powerful tool for Named Entity Recognition (NER) in the medical domain.
- python-Levenshtein: Python extension for computing string distances and similarities

Potential Improvements

1. Service Registration: Backend code follows a Microservice architecture. The system uses a static service map in the 'src/di-gateway/app_config. py' file to simulate. It is simple but lacks flexibility. A good solution is to use Consul. It is a robust, open-source tool developed by HashiCorp that allows developers to build, deploy, and scale services across multiple platforms and architectures. It provides key features such as service discovery, configuration, and coordination, making it a one-stop-shop for microservices architecture management.
2. Task Status Notification: To check the mapping task status, users have to refresh the page to check. There should be a pop-up window to let the user know if the task is finished/failed. A good solution is to implement a web socket function to track the status of task. This is not implemented due to being out of scope.
3. Reset/Forget Password: We did not implement the reset and forget password function, due to being out of scope.
4. Mapper controller: One of the initial goals from the client is that they would like to use different tools to do the mapping. Therefore, when we design the system architecture, we isolated the map function into the Map container. We assume there could be multiple Map containers but with different functions of mapping, curating, and resetting default setting functions. When a map task is created, the user can select the mapper to do the mapping task. However, this is not implemented due to the time limit.
5. A file system: there could be a file storage system like Redis. Specifically, the only file the system stores is the avatar. The system is storing the avatar in the local directory. If there are more files needed to save, a Redis container will be a good choice.
6. Task queue: The system has a weakness in task management. In detail, the system has the risk if there are multiple task processing. There could be a mechanism to allow the system to have a task queue so that the new tasks will wait for the old tasks to finish.

Release Notes

Product name	Digital Health Platform
Release date	09 Jun 2023
Release notes version	Version 1.0.0
Release notes date	08 Jun 2023

Background

The primary objective of this platform is to determine whether medications prescribed to patients are appropriate by normalizing free-text clinical notes and mapping them to canonical clinical terms.

The platform's primary function is to simplify the process of associating brief free-text descriptions, which generally explain the reasoning behind prescribing specific medications, onto a Universal Indication List (UIL), which serves as a subset of the broader standardized knowledge base of clinical terms known as SNOMED CT.

The platform features the integration of a human-in-the-loop system, which allows for manual review and correction of the mapping results. This feedback will be used to continuously enhance the platform's accuracy and performance.

This curation feature will further streamline the mapping process, ensuring that the most relevant and commonly used clinical terms are easily accessible for healthcare professionals and researchers.

Repository Structure

Here we list the top-level directory of this repository

More details about the sub-directories can be found in docs directory repository structure.

```
.
 .github       # CI/CD Github Action scripts
 data samples  # Sample input for the prototype
 docs          # Documentation files
 prototypes    # Designed user interface - prototypes
 src           # Source code
 tests         # Code pieces and tests of source code
 LICENSE       # <Not included for now>
 docker-compose.yml  # deploy other services exclude ontoserver
 ontoserver-docker-compose.yml   # deploy ontoserver
 README.md
 .gitignore
```

Features

- Map: Translate clinical texts to Universal Indication List and SNOMED-CT
- Curate: Manually map result category to continuously improve mapping performance
- Visulization: Mapping result visulization
- Download: Export the mapping result
- Dashboard: System performance visulization
- Team: Member management
- Code system: Update code system version

Requirements

System requirments

Resource	Minmum	Recommended
CPUs or Cores	4	8
RAM	8G	16G
Storage/Disk	20G	>=40G

Prerequisites (for Windows/Windows Server)

Ensure that the following software is installed:

1. Windows Subsystem for Linux (WSL): Follow the official Microsoft WSL Installation Guide.
2. Docker: Download Docker for Windows from the Docker website and install it.
3. Docker Compose: Docker Compose is included in the Docker Desktop installation package.
4. Node.js and Yarn: Download and install Node.js from the official Node.js website. Please ensure you select a version that is 18.x or above. After installing Node.js, install Yarn by following the instructions on the Yarn website.

Ensure that you select the option to use WSL 2 as the default when installing Docker.

Note: If using Docker Desktop for Windows, Select the Use WSL 2 based engine check box from Settings. And then select Apply & Restart.

Prerequisites (for Ubuntu Linux)

Ensure that the following software is installed:

1. Docker: Follow the official Docker installation guide for Ubuntu to install it.
2. Docker Compose: After Docker is installed, follow the Docker Compose installation guide.
3. Node.js and Yarn: Download and install Node.js and Yarn by running the following commands in the terminal:

```
sudo apt update
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install nodejs
sudo apt install npm
sudo npm install --global yarn
```

Prerequisites (for macOS)

Ensure that the following software is installed:

1. Docker: Download Docker Desktop for Mac from the Docker website and install it.
2. Docker Compose: Docker Compose is included in the Docker Desktop installation package.
3. Node.js and Yarn: Download and install Node.js from the official Node.js website. Please ensure you select a version that is 18.x or above. After installing Node.js, install Yarn by following the instructions on the Yarn website.

Ontoserver Licensing and Setup

- Root permission required
- Ontoserver license:
- Within Australia, email help@digitalhealth.gov.au to request a (free) Ontoserver licence. ADHA will then arrange authorisation for your quay.io account
- Elsewhere, email ontoserver-support@csiro.au to discuss licensing terms (both evaluation and production licences are available for single and multiple instances with no limit on the number of users). Once the licence is established, CSIRO will register your quay.io account name to enable access to their repository
- Note: The license is provided for either individual or organisation. University of Melbourne holds an active license, if you would like to use the license from Unimelb, please contact who is responsible for the Ontoserver license at Unimelb.
- Once you have a license and have access to the Ontoserver image on quay.io, remember to change the `CLIENT_ID` and `CLIENT_SECRET` in the `docker-compose.yml`. For information on how to retrieve these credentials, please consult the Ontoserver documentation.
- Ensure your dockerhub account has been registered with ontoserver-support@csiro.au
- Docker login to quay.io required

MedCAT Model Setup

1. Download the model by following this link and sign into your NIH profile / UMLS license.
2. You will then be redirected to the MedCAT model download form. Please complete the form and you will be provided a download link.
3. Unzip the downloaded model, rename the folder as `medcat_model` and move it under `DI-Boxjelly/src/di-map` folder.

Ensure that you select the option to use model of SNOMED International (Full SNOMED modelpack trained on MIMIC-III).

Environment Variables Setup

Environment variables are used in this project to manage different configurations for different environments (development, production, etc.). In particular, REACT_APP_PROD_BASE_URL and REACT_APP_PROD_DOMAIN_URL are required.

1. In the directory of the `DI-Boxjelly/src/di-web`, create a new file named .env.
2. Open the .env file and define the environment variables in the following format:

   ```
   REACT_APP_PROD_BASE_URL=http://localhost:8000
   REACT_APP_PROD_DOMAIN_URL=http://localhost
   ```

   Replace http://localhost:8000 and http://localhost with the appropriate URL of your production server and domain, especially if you're running on a remote instance.

Installation and deployment

1. Clone the repository:

```
git clone https://github.com/COMP90082-2023-SM1/DI-Boxjelly.git
```

Or unzip the provided package to a suitable location on your system.

2. (For Windows users) Open the WSL terminal. You can do this by opening the Start menu, typing "WSL", and selecting the WSL application. Alternatively, open a command prompt and type `wsl` to switch to the Linux environment.
3. Navigate into the project repository:

   ```
   cd DI-Boxjelly
   ```

   (For Windows users) The Windows file system can be accessed under `/mnt/` in WSL. For example, if you unzipped the package to `C:\myproject\DI-Boxjelly` on your Windows file system, you can access it in WSL with the following command:

   ```
   cd /mnt/c/myproject/DI-Boxjelly
   ```

4. Docker login to quay.io following instruction in Ontoserver Licensing and Setup.

   ```
   sudo docker login -u=[Username] -p=[CLI Password] quay.io
   ```

5. Run the setup script:

   ```
   sudo ./setup.sh
   ```

   (For Windows users) Run the setup script:

   ```
   sudo apt-get install dos2unix
   dos2unix setup.sh
   sudo bash ./setup.sh
   ```

   - The setup script automates the installation and deployment process.
   - This allows you to have 8 containers: mongo, di-gateway, di-auth, di-center, di-map, nginx, ontoserver and ontoserver-db.

   Note: `sudo` runs commands with administrative privileges, which may ask for your system password.

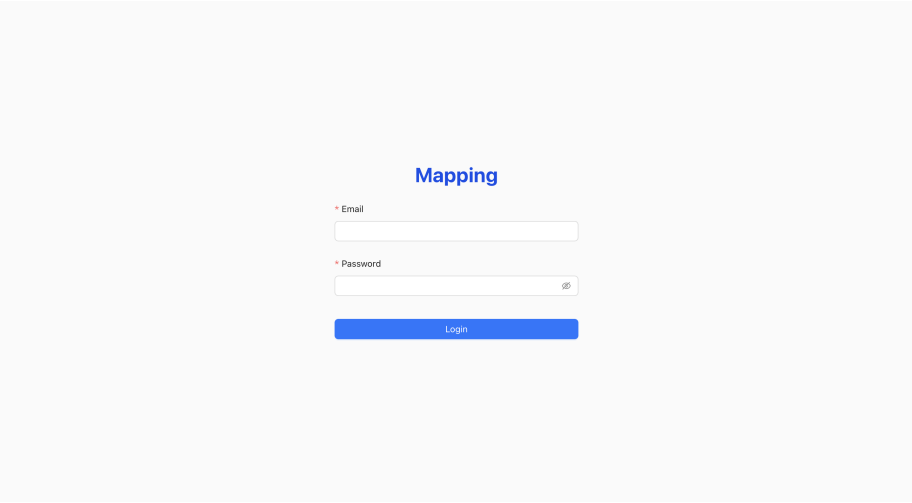
Website Demo

The production environment can be accessed at http://172.26.131.202. Please note that access is only available via the University of Melbourne's Wi-Fi network or through a Cisco VPN. This is due to the fact that the website is hosted on an internal network for security reasons. This instance will not be available in the near future.

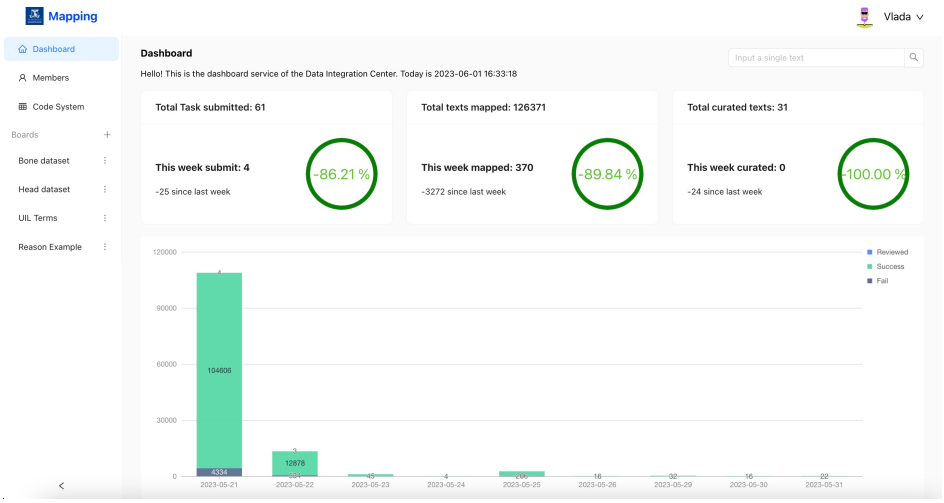
A demo video is available:

Website Preview

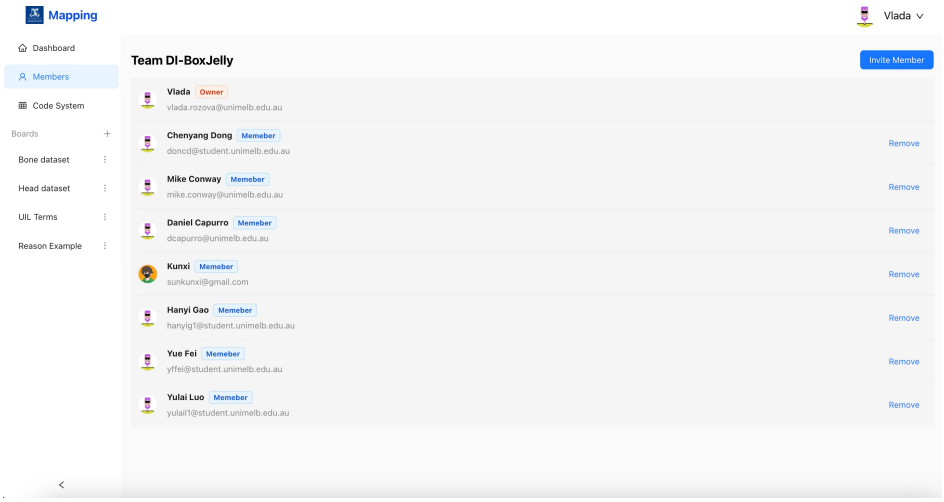
Login:



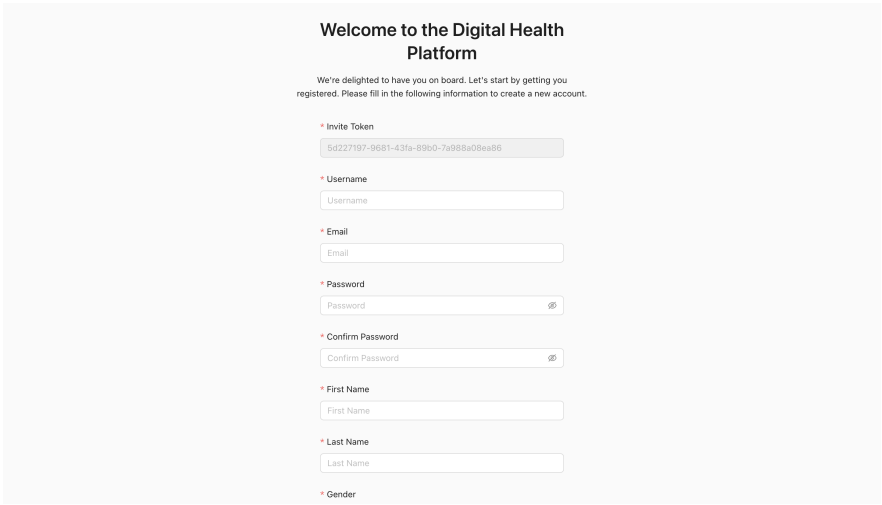
Dashboard:



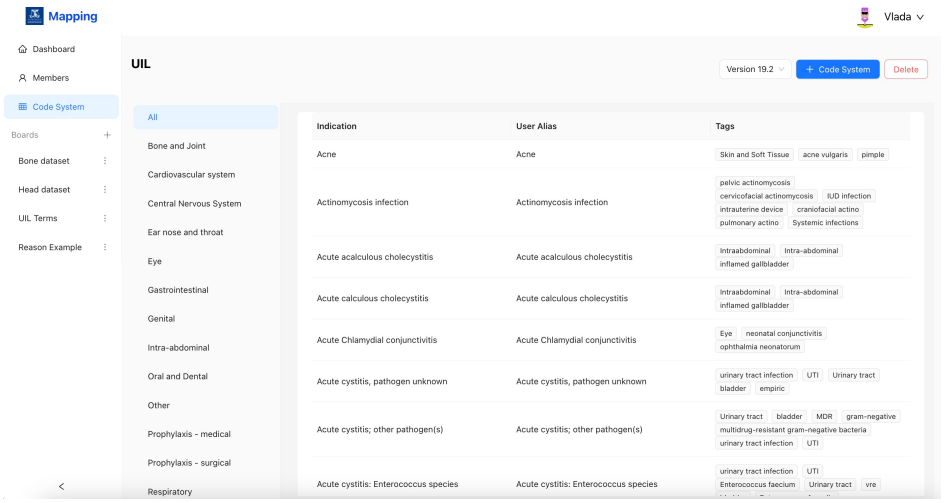
Team Management



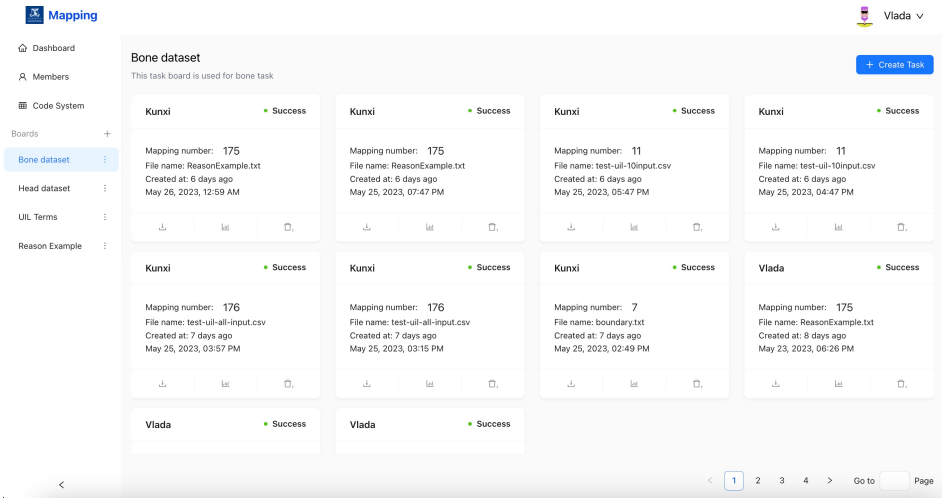
Invite Members



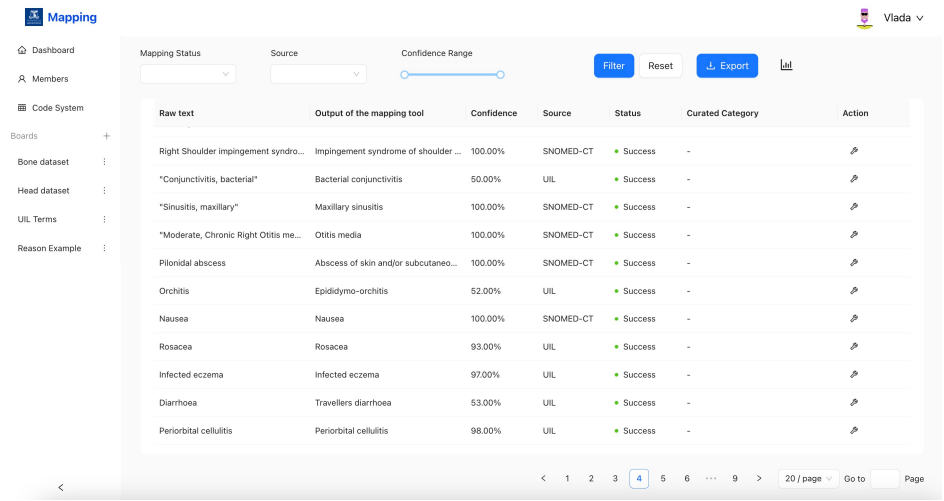
Code System



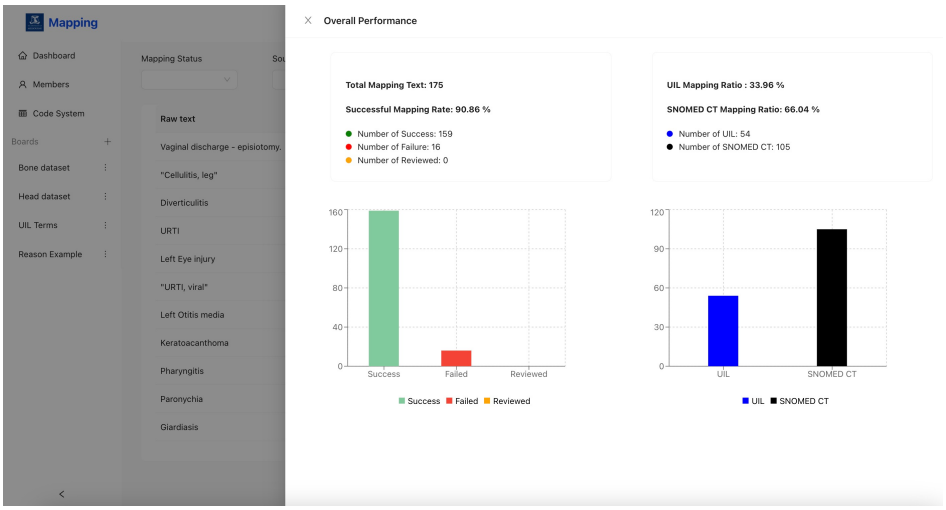
Map Task



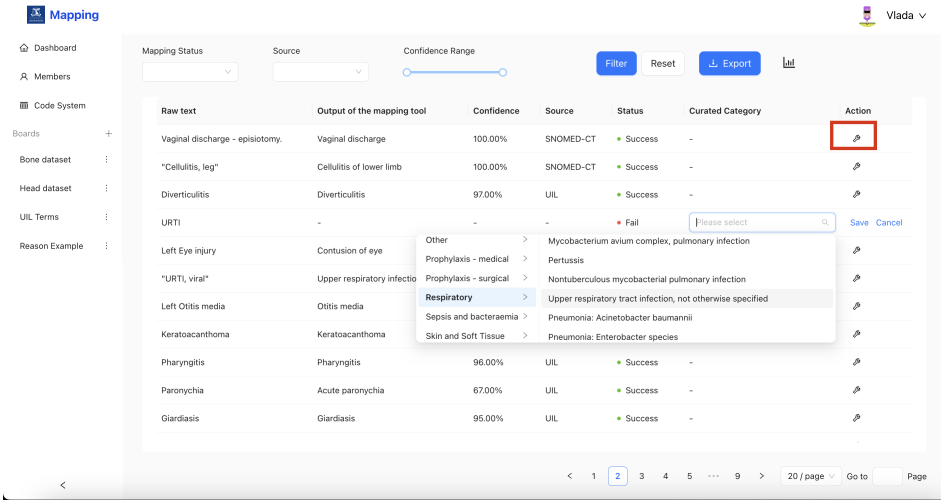
Map Result



Map Result Visualisation

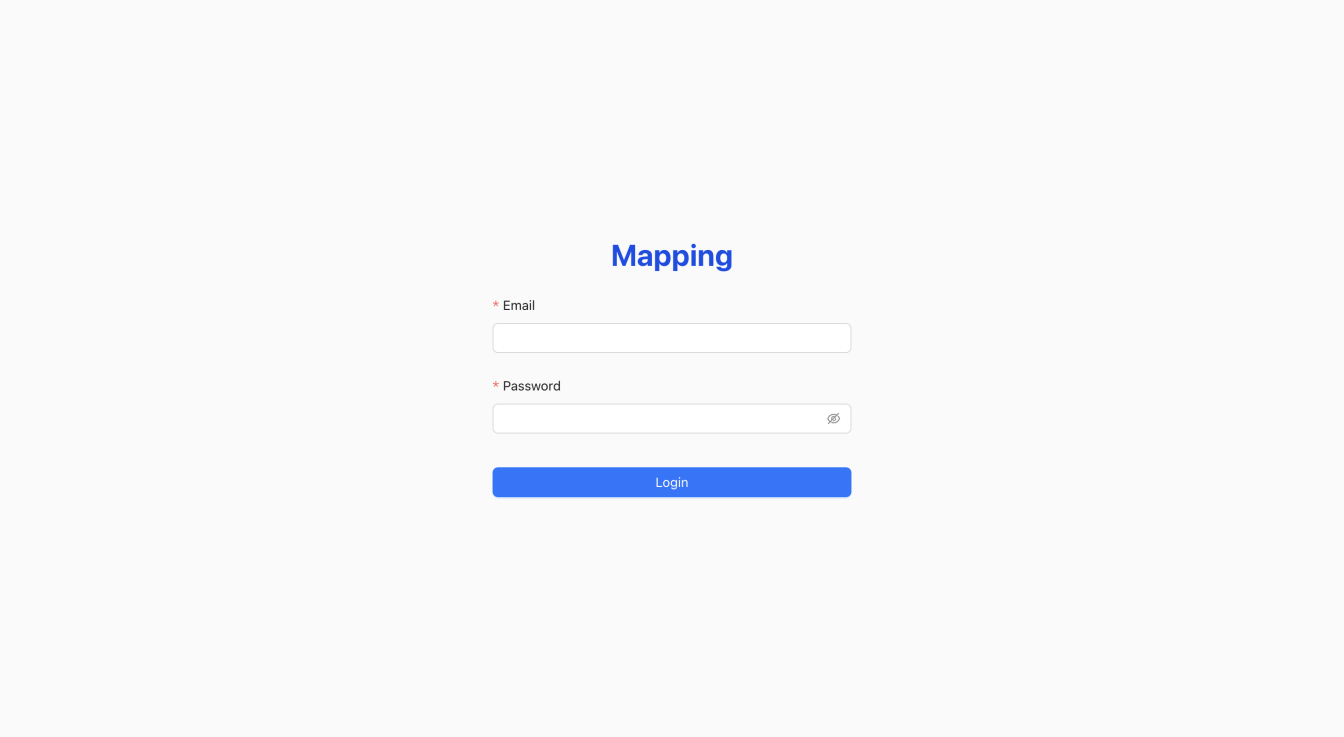

Mapping
Dashboard
Members
Code System
Boards
Bone dataset
Head dataset
UIL Terms
Reason Example
Mapping Status
Raw text
Vaginal discharge - episiotomy.
"Cellulitis, leg"
Diverticulitis
URTI
Left Eye injury
"URTI, viral*
Left Otitis media
Keratoacanthoma
Pharyngitis
Paronychia
Giardiasis
Overall Performance
Total Mapping Text: 175
Successful Mapping Rate: 90.86 %
Number of Success: 159
Number of Failure: 16
Number of Reviewed: 0
UIL Mapping Ratio : 33.96 %
SNOMED CT Mapping Ratio: 66.04 %
Number of UIL: 54
Number of SNOMED CT: 105
160
120
80
40
0
Success
Failed
Reviewed
Success Failed Reviewed
120
90
60
30
0
UIL
SNOMED CT
UIL SNOMED CT

Curation


Mapping
Vlada
Dashboard
Members
Code System
Boards
Bone dataset
Head dataset
UIL Terms
Reason Example
Mapping Status
Source
Confidence Range
Filter
Reset
Export
Raw text
Output of the mapping tool
Confidence
Source
Status
Curated Category
Action
Vaginal discharge - episiotomy.
Vaginal discharge
100.00%
SNOMED-CT
Success
"Cellulitis, leg"
Cellulitis of lower limb
100.00%
SNOMED-CT
Success
Diverticulitis
Diverticulitis
97.00%
UIL
Success
URTI
Fail
Please select
Save
Cancel
Left Eye injury
Contusion of eye
"URTI, viral*
Left Otitis media
Otitis media
Keratoacanthoma
Keratoacanthoma
Pharyngitis
Pharyngitis
96.00%
UIL
Success
Paronychia
Acute paronychia
67.00%
UIL
Success
Giardiasis
Giardiasis
95.00%
UIL
Success
Other
Prophylaxis - medical
Prophylaxis - surgical
Respiratory
Sepsis and bacteraemia
Skin and Soft Tissue
Mycobacterium avium complex, pulmonary infection
Pertussis
Nontuberculous mycobacterial pulmonary infection
Upper respiratory tract infection, not otherwise specified
Pneumonia: Acinetobacter baumannii
Pneumonia: Enterobacter species
20 / page
Go to
Page

User Guide

User Login

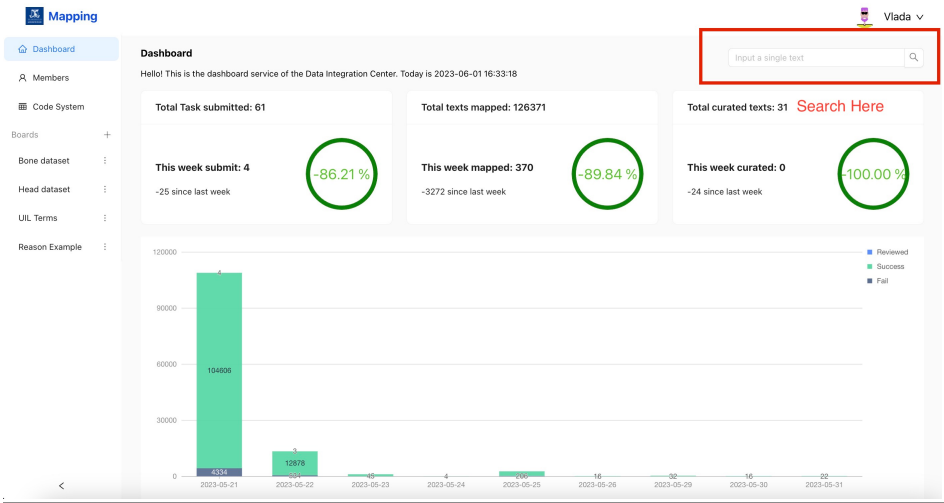


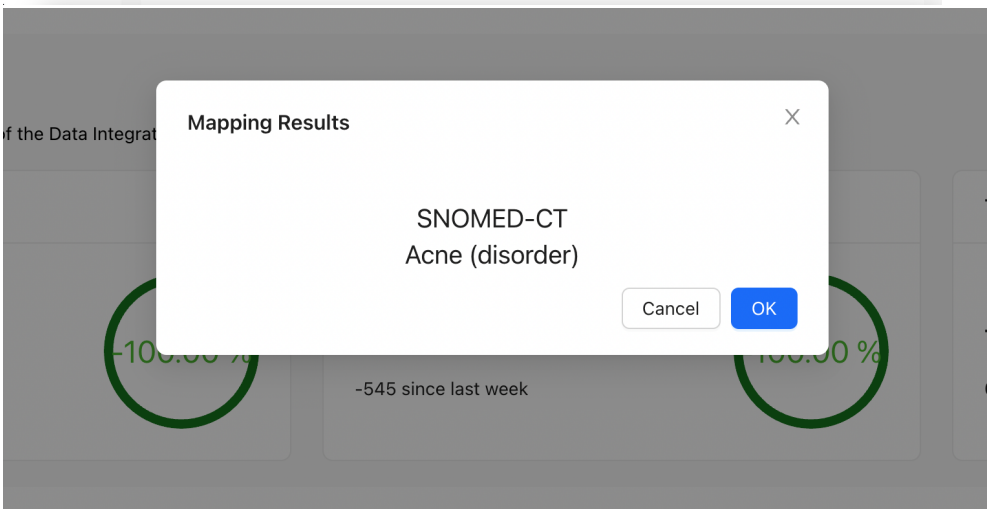
Step 1: Input username and password

Step 2: Click Sign in

(Note: Make sure you have accepted the link invitation and registered)

Single-text Mapping





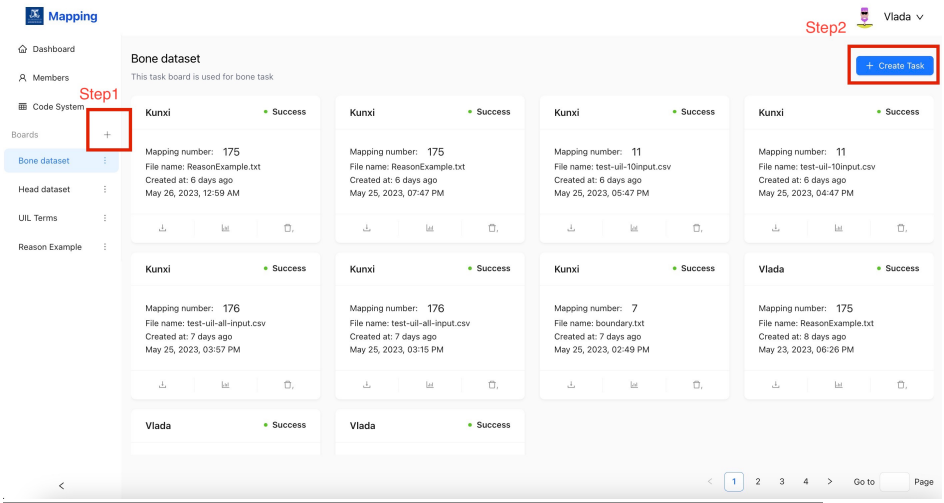
Step 1: After logining the system, click on the search box in the top-right corner.

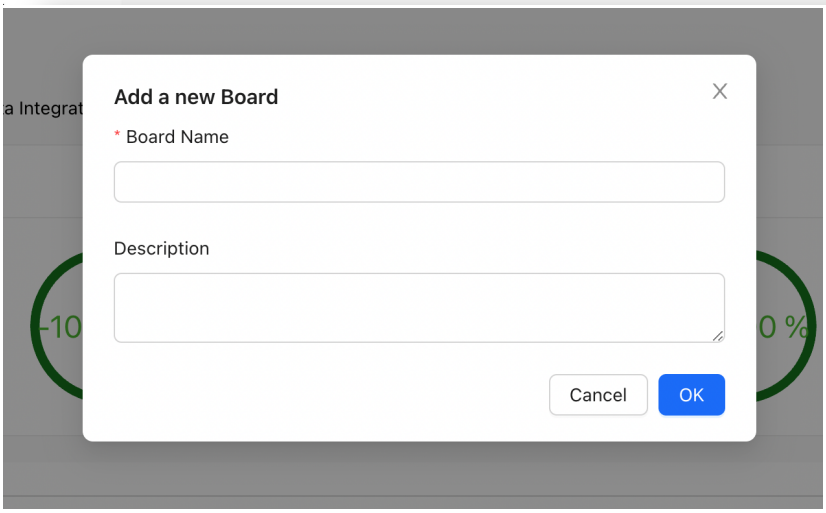
Step 2: Click the search button, and then the mapping results will show.

Create Map Task

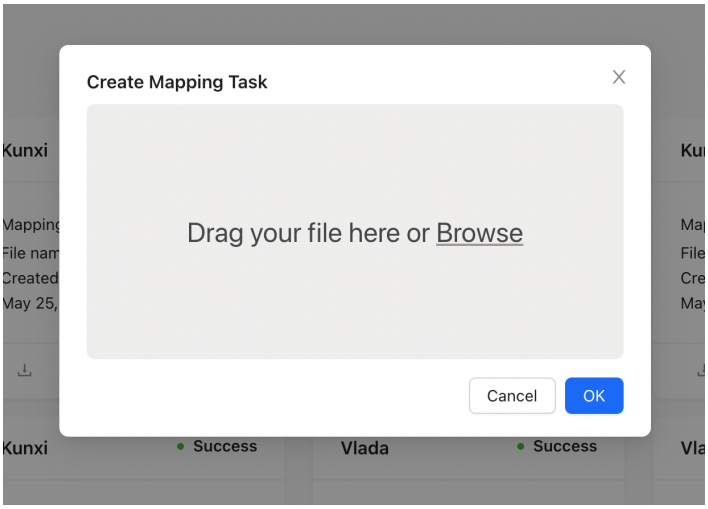
Step1: Create a Board and enter the board name on the pop-up window

Step2: Create a Task on the board

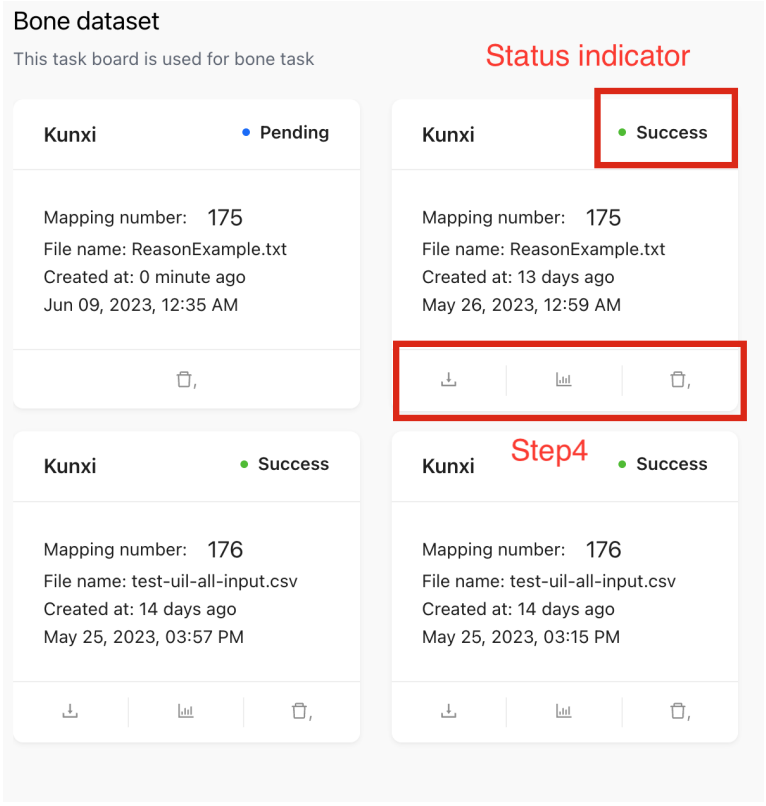




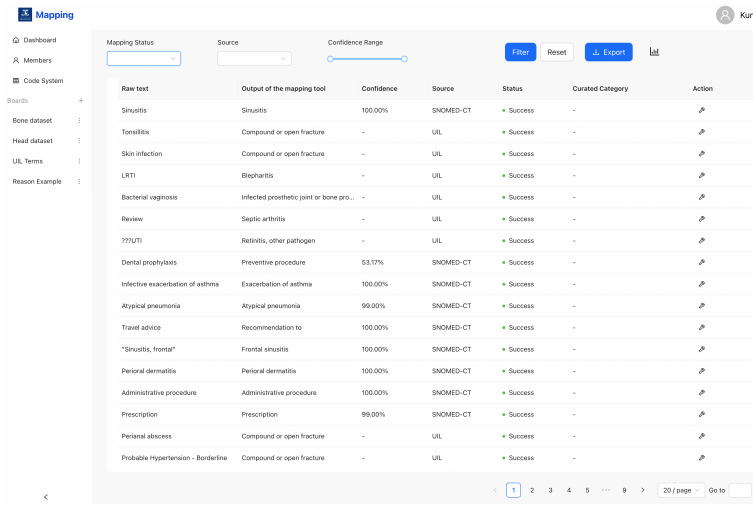
Step3: Drag or click Browse to choose the file on your computer, then click OK



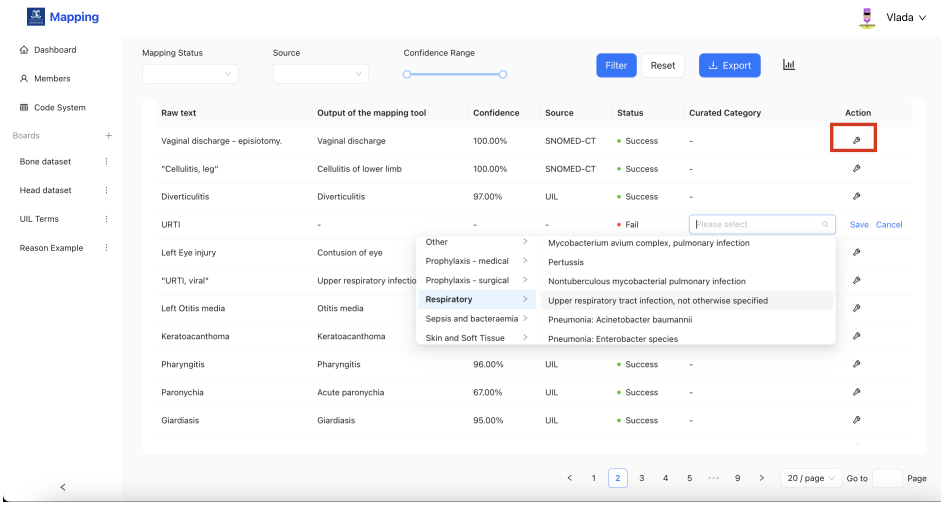
Step 4: After the status indicator shows success, the user can click the three under the icon to download, visualise and delete the task. Or they can click the rest of the card to view task details.



Step5: Now you can get the mapping result of the document



Curate and Retrain

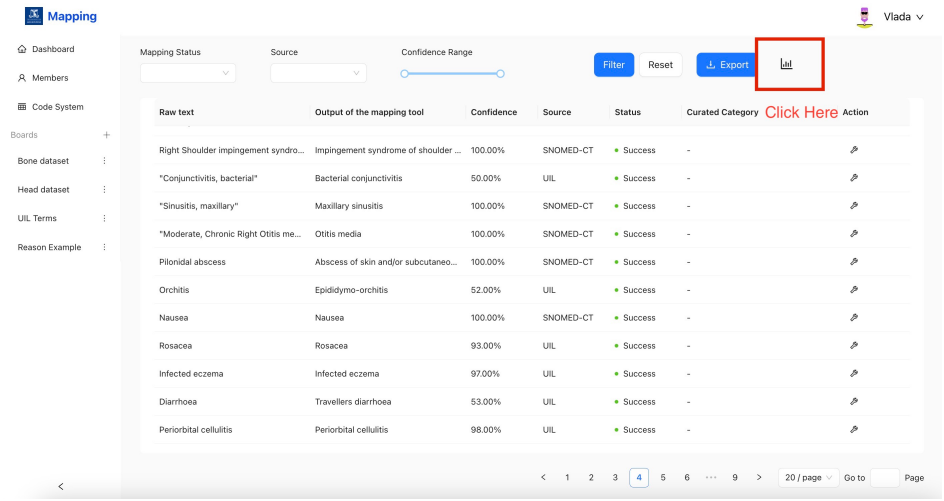


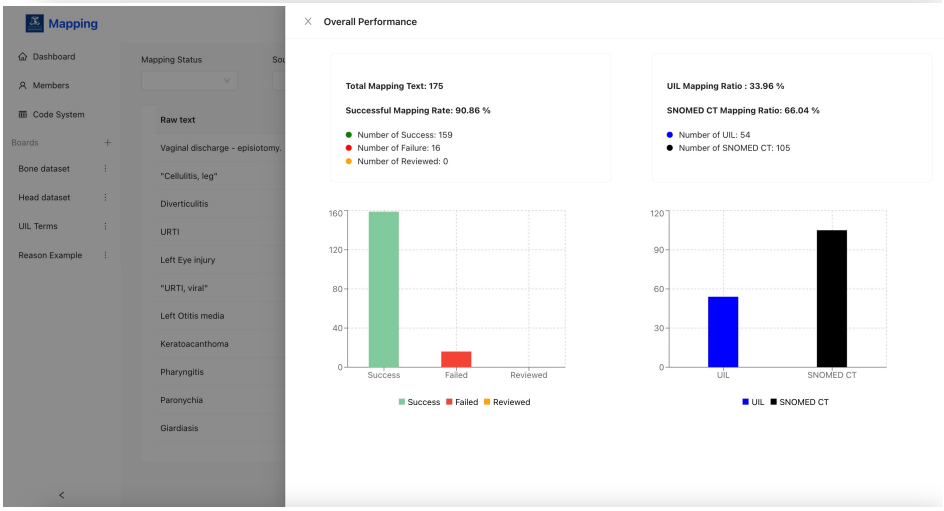
Step1: Click on the action icon on the term that you want to curate

Step2:Choose the category that you want to curate to

Step3: Click save, then the status of the curated term will be changed to "Reviewed"

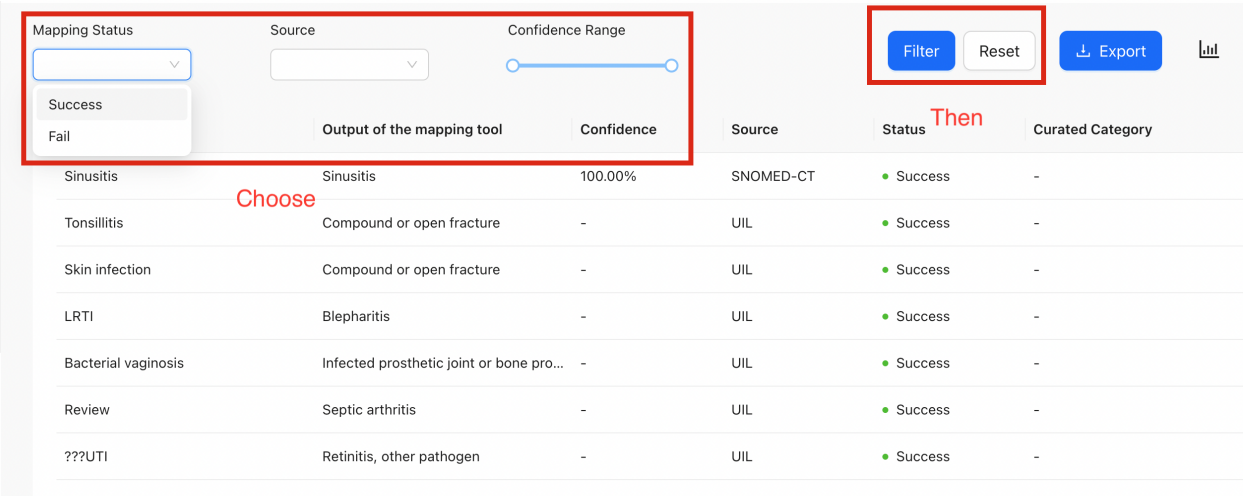
Visualisation





Step1:Click on the visualisation icon, then the visualisation drawer shows

Filter the results



Step1: Choose the filter condition

Step2: Click "Filter" to get the result

Development

Title	Creator	Modified
APIs	KUNXI SUN	31 May, 2023
Digital Prototype	Chenyang Dong	02 Apr, 2023

APIs

Version ID	Description	Date
2.0.0	1. Change the API to the latest	May 30,2023
1.0.0	1. A basic API plan	May 01,2023

Version 2.0.0

Modules	Sub Modules	APIs	Description
Authentication Service	-	1. User Login 2. User Logout 3. User Register <FOR DEVELOPER TEST ONLY> 4. Get user profile data 5. Update user profile 6. Get team members and info 7. Invite a team member to the team 8. Accept an invitation 9. Get user's avatar 10. Update user's avatar	Team management, User information, Authentication
Center Service	Dashboard	1. Predict single clinical text 2. Get the item status ratio	CodeSystem, Task, Visualization, Dashboard
	CodeSystem	1. Create a new code system 2. Get code system concepts of a team 3. Delete a code system 4. Get all versions of code system	
	Boards	1. Get board lists 2. Modify a board detail 3. Delete a board 4. Create a new task board 5. Get Task Meta Data 6. Get all map tasks 7. Get map task details 8. Create a map task 9. download map task result 10. Curate the map item	
Mapper Service	-	1. Map Clinical Text to SNOMED CT 2. Predict the texts 3. Retrain the model 4. Reset the model to original	Map, Curation, Retrain
Gateway Service	-	1. Redirect requests	Control and manage the request and response

---

Based on the user stories, to implement maintainable and extendable system architecture, we split the system into five modules. Each of them represents a Flask application in Docker container.

Version 1.0.0

Module	APIs	Description

Auth Service	1. User Login 2. User Logout 3. User forget password 4. User reset password 5. User Register <FOR DEVELOPER TEST ONLY>	Authentication, team management, user information
Center Service	1. Create a map task 2. Get detail of a map task 3. Delete a map task 4. Get mapped items of a map task 5. Update(Curate) the category of a mapped item in map task	
Map Service	1. Create a new UIL category 2. Get details of a category 3. Update a category 4. Delete a category 5. Get the UIL list of all version 6. Get the UIL list of a version	
Common Service	1. Create a Team 2. Add team member into a team 3. Delete a team member from a team 4. Change owner of a team 5. Update permission of a team member 6. Send a email 7. Upload a file 8. Update SNOMED CT version 9. Get Dictionary	
Gateway Service	1. Redirect requests	

Auth Service

DigitalHealth

> *v1.0.0*

Base URLs: http://localhost:8001

auth

POST Login

POST /auth/login/email

Authenticates a user by their username and password, and if successful, returns an access token (e.g., JWT).

> *Body Parameters*

```
email: sunkunxi@gmail.com
password: "12345678"
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none
» email	body	string	no	User email
» password	body	string	no	User password

> *Response Examples*

> *200 Response*

```
{
  "code": 200,
  "msg": "ok"
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline
400	Bad Request	Invalid Email and Password	Inline
401	Unauthorized	Incorrect password	Inline
404	Not Found	User not found	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» msg	string	true	none		none

HTTP Status Code **400**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» err	string	true	none		none

HTTP Status Code **401**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» err	string	true	none		none

HTTP Status Code **404**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» err	string	true	none		none

POST Logout

POST /auth/logout

Logs out the user by invalidating their access token.

> *Response Examples*

> *200 Response*

```
{
  "code": 200,
  "msg": "ok"
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline
400	Bad Request	Not log in	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» msg	string	true	none		none

HTTP Status Code **400**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» err	string	true	none		none

POST Invite a team member to a team

GET /auth/team/invite

This system allows users to add a new team member to the existing team by providing the team ID and the new member's user ID along with their role.

Params

Name	Location	Type	Required	Description
team_id	query	string	no	The unique identifier of the team.

> *Response Examples*

> *200 Response*

```
{
  "team_id": "string"
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline
400	Bad Request	Bad Request	Inline
404	Not Found	Record Not Found	Inline
500	Internal Server Error	Server Error	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» team_id	string	true	none		none

HTTP Status Code **400**

Name	Type	Required	Restrictions	Title	description
» error	string	true	none		none
» message	string	true	none		none

HTTP Status Code **404**

Name	Type	Required	Restrictions	Title	description
» error	string	true	none		none
» message	string	true	none		none

HTTP Status Code **500**

Name	Type	Required	Restrictions	Title	description
» error	string	true	none		none
» message	string	true	none		none

GET Get team members and info

GET /auth/team

Get the team members and team informations

Params

Name	Location	Type	Required	Description

team_id	query	string	no	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

GET Get User profile data

GET /auth/user

Get user profile information, including name, email, nickname, gender

Params

Name	Location	Type	Required	Description
user_id	query	string	no	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

PUT Update user profile

PUT /auth/user

Update the user profile

> *Body Parameters*

```
{
  "first_name": "DI",
  "last_name": "Boxjelly",
  "nickname": "di-boxjelly",
  "gender": "male"
}
```

Params

Name	Location	Type	Required	Description

body	body	object	no	none
» first_name	body	string	no	none
» last_name	body	string	no	none
» nickname	body	string	no	none
» gender	body	string	no	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

POST Accept a invitation

POST /auth/team/accept

The invitee can accept the invitation by registering using invitation link.

> *Body Parameters*

```
invite_token: string
username: string
email: string
password: string
first_name: string
last_name: string
gender: string
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none
» invite_token	body	string	no	none
» username	body	string	no	none
» email	body	string	no	none
» password	body	string	no	none
» first_name	body	string	no	none
» last_name	body	string	no	none
» gender	body	string	no	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	Success	Inline

Responses Data Schema

GET Get avatar of a user

GET /auth/user/avatar

Get avatar of a user

Params

Name	Location	Type	Required	Description
avatar	query	string	no	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

POST Update the avatar file

POST /auth/user/avatar

> *Body Parameters*

```
avatar: file:///Users/sunkunxi/Pictures/IMG_4391.JPG
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none
» avatar	body	string(binary)	yes	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

Data Schema

Center Service

DigitalHealth

> *v1.0.0*

Base URLs: http://localhost:8002

center

POST Create a Map Task

POST /center/boards/tasks

Create a new map task for a user in a team with the specified UIL version.

> *Body Parameters*

```
file: file:///Users/sunkunxi/Desktop/test.csv
team_id: 60c879e72cb0e6f96d6b0f65
board_id: 60c879e72cb0e6f96d6b0f65
```

Params

Name	Location	Type	Required	Description
Content-Type	header	string	yes	none
body	body	object	no	none
» file	body	string(binary)	no	none
» team_id	body	string	no	none
» board_id	body	string	no	none

> *Response Examples*

> *200 Response*

```
{
  "data": {
    "id": "645328311abd2039faa81f3a",
    "num": 0,
    "status": "pending"
  },
  "code": 0,
  "msg": "string"
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» data	object	true	none		none

»» id	string	true	none		none
»» num	integer	true	none		none
»» status	string	true	none		none
» code	number	true	none		none
» msg	string	true	none		none

GET Get All Map Tasks

GET /center/boards/tasks

Params

Name	Location	Type	Required	Description
page	query	integer	no	none
size	query	integer	no	none
team_id	query	string	no	none
board_id	query	string	no	none

> *Response Examples*

> *success*


```
{
  "code": "9",
  "msg": "success",
  "data": {
    "page": 2,
    "size": 10,
    "page_num": 2,
    "tasks": [
      {
        "id": "47",
        "status": "pending",
        "num": 8,
        "create_by": "dolor",
        "create_at": "2003-12-15 15:17:17",
        "update_at": "2006-07-07 11:30:59"
      },
      {
        "id": "52",
        "status": "success",
        "num": 25,
        "create_by": "Duis et nisi dolore ad",
        "create_at": "1981-04-26 06:12:29",
        "update_at": "2007-09-20 23:51:33"
      },
      {
        "id": "35",
        "status": "fail",
        "num": 58,
        "create_by": "exercitation in esse",
        "create_at": "1978-03-07 15:46:58",
        "update_at": "2020-02-01 13:32:27"
      },
      {
        "id": "25",
        "status": "success",
        "num": 10,
        "create_by": "nisi",
        "create_at": "2015-12-17 00:21:47",
        "update_at": "1995-01-16 13:10:39"
      }
    ]
  }
}
```


Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» code	number	true	none		none
» msg	string	true	none		none
» data	object	true	none		none
»» page	integer	true	none		current page number
»» size	integer	true	none		page size
»» page_num	integer	true	none		total page number
»» tasks	[object]	true	none		none
»»» id	string	true	none		none
»»» status	string	true	none		none
»»» num	integer	true	none		number of mapped items in the task
»»» create_by	string	true	none		none
»»» create_at	string	true	none		none
»»» update_at	string	true	none		none

GET Get Map Task Detail

GET /center/boards/task/detail

Get detail of a map task

> *Body Parameters*

```
{}
```

Params

Name	Location	Type	Required	Description
page	query	integer	no	none
size	query	integer	no	none
team_id	query	string	no	none
board_id	query	string	no	none
task_id	query	string	no	none
body	body	object	no	none

> *Response Examples*

> *200 Response*

```
{
  "data": {
    "id": "string",
    "status": "pending",
    "create_at": "string",
    "update_at": null,
    "items": [
      {
        "id": "string",
        "mapping_status": 0,
        "original_text": "string",
        "mapped_text": null,
        "confidence": null,
        "source": null,
        "curate": null
      }
    ],
    "num": 0,
    "num_success": 0,
    "num_fail": 0
  },
  "code": "200",
  "msg": "ok"
}
```


Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» data	object	true	none		none
»» id	string	true	none		none
»» status	string	true	none		0: fail 1: pending 2: success
»» create_at	string	true	none		none
»» update_at	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»» *anonymous*	string	false	none		none

continued

Name	Type	Required	Restrictions	Title	description
»» items	[object]	true	none		none
»»» id	string	true	none		none
»»» mapping_status	integer	true	none		0: fail 1:success 2: reviewed
»»» original_text	string	true	none		none
»»» mapped_text	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description

»»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»»» *anonymous*	string	false	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»»» confidence	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»»» *anonymous*	null	false	none		if map fail

or

Name	Type	Required	Restrictions	Title	description
»»»»» *anonymous*	number	false	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»»» source	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»»» *anonymous*	object	false	none		none
»»»»»» status	integer	true	none		0: snomed, 1: uil
»»»»»» snomed	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»»»»» *anonymous*	object	false	none		none
»»»»»»»» code	string	true	none		none
»»»»»»»» name	string	true	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»»»»» uil	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»»»»» *anonymous*	object	false	none		none
»»»»»»»» id	string	true	none		none
»»»»»»»» snomed_code	string	true	none		none
»»»»»»»» name	string	true	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»» curate	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	object	false	none		none
»»»»» category_id	string	true	none		none
»»»»» category_name	string	true	none		none
»»»»» update_at	string	true	none		none

continued

Name	Type	Required	Restrictions	Title	description
»» num	integer	true	none		total number of items
»» num_success	integer	true	none		none
»» num_fail	integer	true	none		none
» code	string	true	none		none
» msg	string	true	none		none

DELETE Delete a Map Task

DELETE /uil/boards/tasks

Params

Name	Location	Type	Required	Description
task_id	query	string	no	none
board_id	query	string	no	none
team_id	query	string	no	none

> *Response Examples*

> *200 Response*

```
{
  "success": true,
  "code": "string",
  "msg": "string"
}
```


Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» success	boolean	true	none		none
» code	string	true	none		none
» msg	string	true	none		none

GET get all mapped items of a task

GET /map/items

Params

Name	Location	Type	Required	Description
id	query	string	no	taskId

> *Response Examples*

> *200 Response*


```
{
  "data": {
    "items": [
      {
        "itemId": "string",
        "mappingSuccess": true,
        "mappingStatus": 0,
        "originalText": "string",
        "mappedText": null,
        "confidence": null,
        "curate": null,
        "uil": null,
        "snomed": null,
        "source": 0
      }
    ],
    "totalNumber": "string",
    "numberOfSuccess": 0,
    "numberOfFail": 0
  },
  "code": "string",
  "msg": "string"
}
```


Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» data	object	true	none		none
»» items	[object]	true	none		none
»»» itemId	string	true	none		none
»»» mappingSuccess	boolean	true	none		none
»»» mappingStatus	integer	true	none		0: success 1:fail 2: reviewed
»»» originalText	string	true	none		none
»»» mappedText	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	string	false	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»» confidence	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	null	false	none		if mapping fail

or

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	number	false	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»» curate	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	object	false	none		none
»»»»» categoryName	string	true	none		none
»»»»» categoryId	string	true	none		none
»»»»» updateAt	string	true	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»» uil	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	object	false	none		none
»»»»» id	string	true	none		none
»»»»» snomedCode	string	true	none		none
»»»»» name	string	true	none		uil category name, should be same as the mapped Text

continued

Name	Type	Required	Restrictions	Title	description
»»» snomed	any	true	none		none

anyOf

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	null	false	none		none

or

Name	Type	Required	Restrictions	Title	description
»»»» *anonymous*	object	false	none		none
»»»»» code	string	true	none		none
»»»»» name	string	true	none		none

continued

Name	Type	Required	Restrictions	Title	description
»»» source	number	true	none		0: fail, 1: snomed, 2: uil
»» totalNumber	string	true	none		none
»» numberOfSuccess	number	true	none		none
»» numberOfFail	number	true	none		none
» code	string	true	none		none
» msg	string	true	none		none

POST Curate The Map Item

POST /center/boards/task/curate

Update(Curate) a category of a mapped item in map task

> *Body Parameters*

```
{
  "board_id": "string",
  "team_id": "string",
  "map_item_id": "string",
  "concept_id": "string"
}
```

Params

Name	Location	Type	Required	Description
id	query	string	no	taskId
body	body	object	no	none
» board_id	body	string	yes	none
» team_id	body	string	yes	none
» map_item_id	body	string	yes	the id of the item
» concept_id	body	string	yes	the id (code) of the curated uil

> *Response Examples*

> *200 Response*

```
{
  "code": "string",
  "msg": "string",
  "success": true
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» code	string	true	none		none
» msg	string	true	none		none
» success	boolean	true	none		none

GET Get Task Meta Data

GET /uil/boards/task/meta

Num of success, number of fail, number of reviewed....

Params

Name	Location	Type	Required	Description
task_id	query	string	no	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema

200	OK	success	Inline

Responses Data Schema

GET Get task board list

GET /uil/boards

Get the task board of a team

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

POST Create a new task board

POST /uil/boards

Get the task board of a team

> *Body Parameters*

```
{
  "team_id": "string",
  "name": "string",
  "description": "string"
}
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none
» team_id	body	string	yes	none
» name	body	string	yes	none
» description	body	string	no	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

PUT Modify a board detail

PUT /uil/boards

Modify a board's name and description

> *Body Parameters*

```
{}
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none

> *Response Examples*

> *200 Response*

```
{
  "team_id": "string",
  "board_id": "string",
  "new_name": "string",
  "new_description": "string"
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» team_id	string	true	none		none
» board_id	string	true	none		none
» new_name	string	true	none		none
» new_description	string	true	none		none

GET /center/dashboard/item-status-ratio

GET /center/dashboard/item-status-ratio

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema

200	OK	success	Inline

Responses Data Schema

center/dashboard

POST Predict single clinical text

POST /center/dashboard/predict

> *Body Parameters*

```
{
  "text": "string"
}
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none
» text	body	string	yes	none

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

center/codesystem

GET Get Code System Concepts of a Team

GET /center/codesystem

Retrieves the UIL category list detail for a specific version.

Params

Name	Location	Type	Required	Description
version	query	string	no	none

> *Response Examples*

> *200 Response*

```
{
  "code": "string",
  "msg": "string",
  "data": {
    "version": "string",
    "categories": [
      {
        "id": "string",
        "name": "string"
      }
    ]
  }
}
```


Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» code	string	true	none		none
» msg	string	true	none		none
» data	object	true	none		none
»» version	string	true	none		The UIL list version.
»» categories	[object]	true	none		An array of categories for the specified version.
»»» id	string	true	none		none
»»» name	string	true	none		none

POST Create a New Code System

POST /center/codesystem

When a new team is created, create a new code sytem

> *Response Examples*

> *200 Response*


```
{}
```


Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

DELETE Delete a code system

DELETE /center/codesystem

Delete the code system given a unique version

Params

Name	Location	Type	Required	Description
version	query	string	no	Version of code system

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

GET Get all versions of codesystem

GET /center/codesystem/versions

Get all version of code system from database

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

Data Schema

Gateway Service

Gateway service redirect all APIs to its sub-services.

All APIs requires JWT token, except three API from authentication service, login, logout and register

Redirect request

POST/GET/PUT/DELETE /<path:path>

<Redirect all APIs>

Map Service

DigitalHealth

> *v1.0.0*

Base URLs: http://localhost:8003

map

POST Map Clinical Text To Snomed CT

POST /map/translate

> *Body Parameters*

```
{
  "texts": [
    "Pharyngitis, streptococcal",
    "Abscess - left lower conjunctiva"
  ]
}
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none
» texts	body	[string]	yes	none

> *Response Examples*

> *200 Response*

```
{
  "code": 0,
  "data": {},
  "msg": "string"
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» data	object	true	none		none
» msg	string	true	none		none

POST Predict the texts

POST /map/predict

Predict the texts

> *Body Parameters*

```
{
  "texts": [
    "string"
  ]
}
```

Params

Name	Location	Type	Required	Description
body	body	object	no	none
» texts	body	[string]	yes	none

> *Response Examples*

> *200 Response*

```
{
  "code": 0,
  "msg": "string",
  "data": {
    "name": "string",
    "predictions": {}
  }
}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
» code	integer	true	none		none
» msg	string	true	none		none
» data	object	true	none		none
»» name	string	true	none		none
»» predictions	object	true	none		none

POST Retrain the model

POST /map/retrain

> *Response Examples*

> *200 Response*

```
{}
```

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

POST Reset the model to original

POST /map/reset

> *Response Examples*

> *200 Response*

```
{}
```

Responses

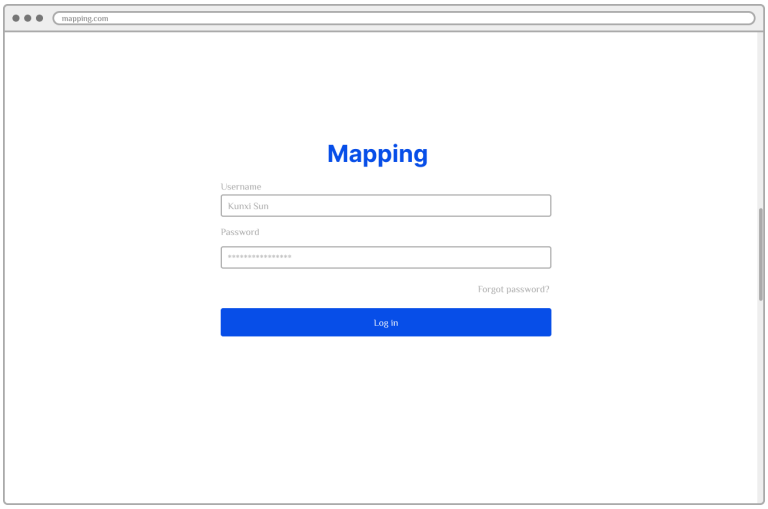
HTTP Status Code	Meaning	Description	Data schema
200	OK	success	Inline

Responses Data Schema

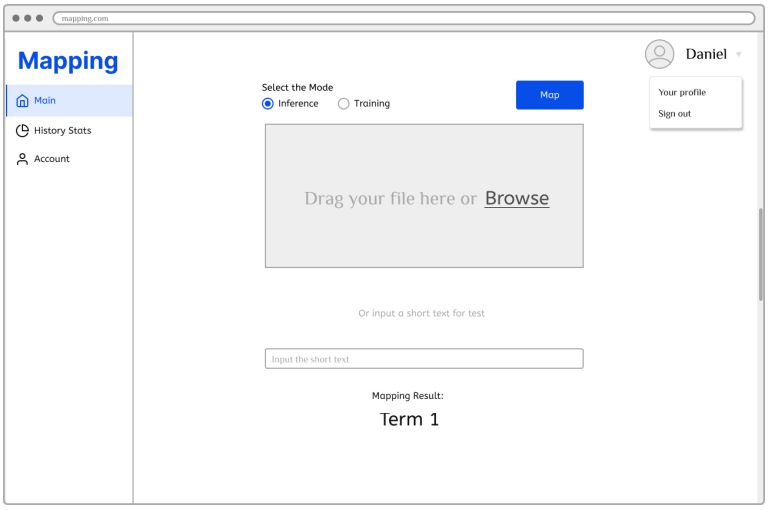
Data Schema

Digital Prototype

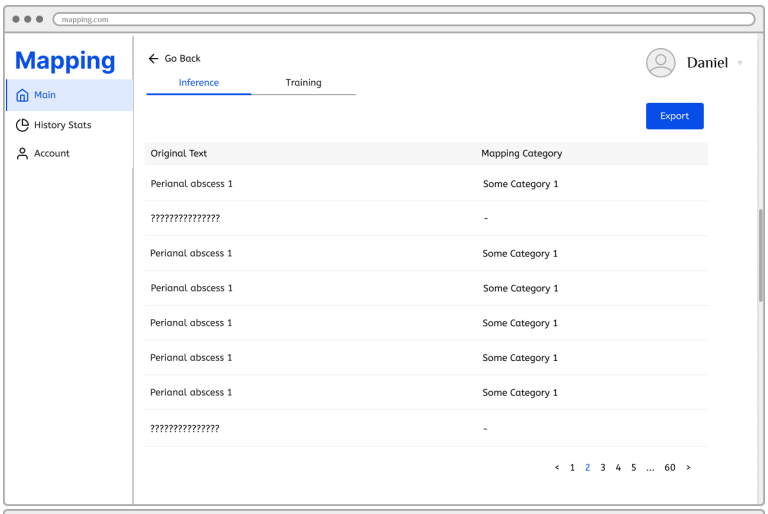
Log in

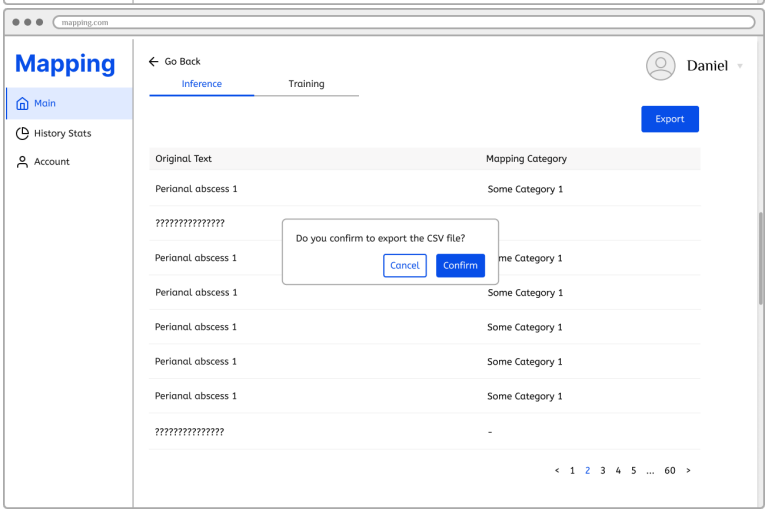


Main

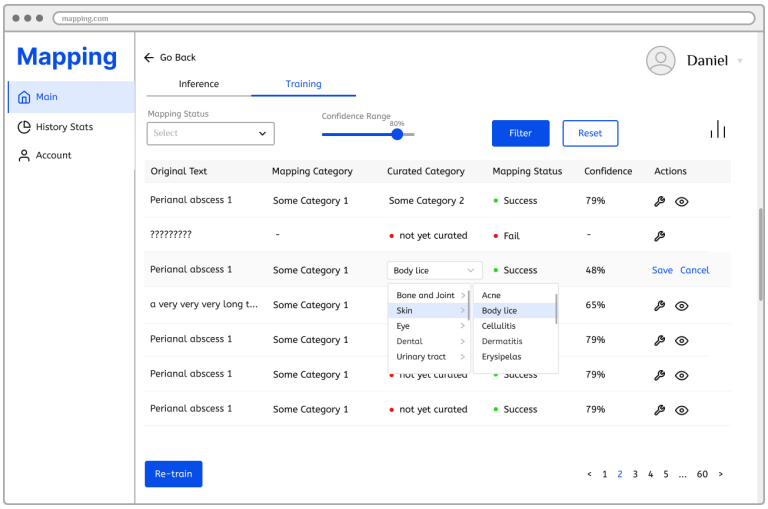


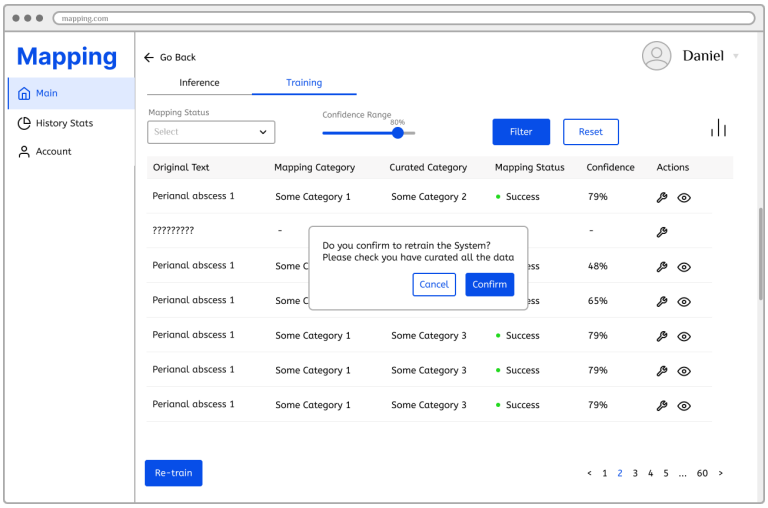
Inference Mode

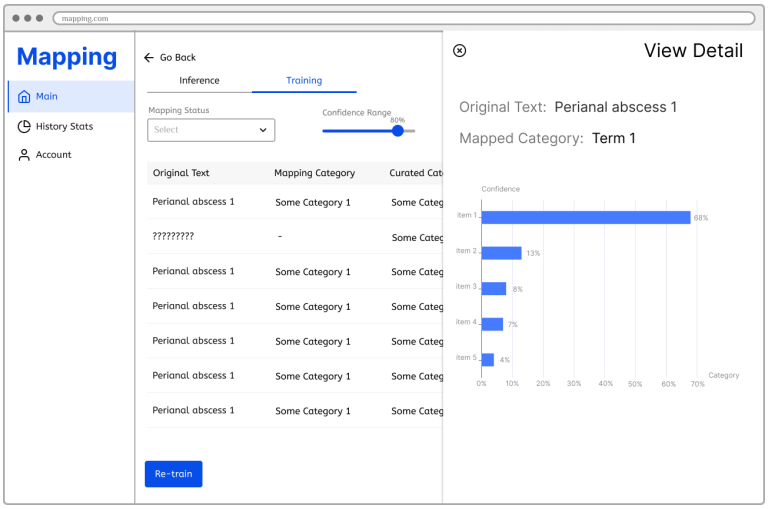


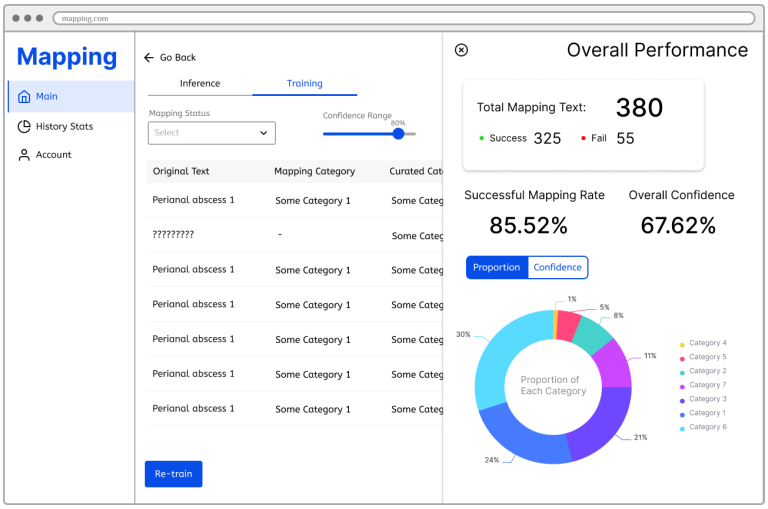


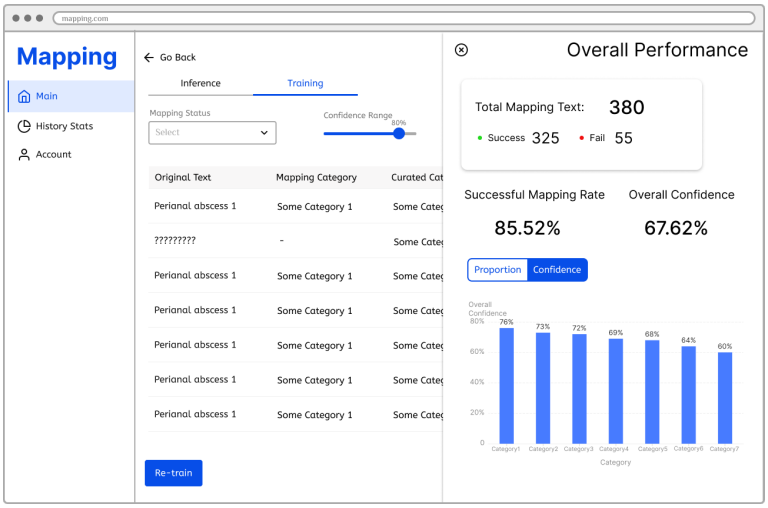
Training Mode



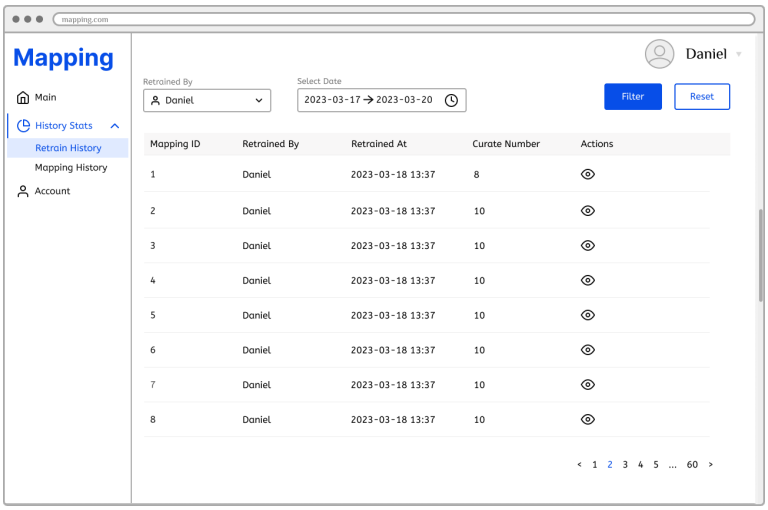

mapping.com
Mapping
Main
History Stats
Account
Go Back
Daniel
Inference
Training
Mapping Status
Select
Confidence Range
80%
Filter
Reset
Original Text
Mapping Category
Curated Category
Mapping Status
Confidence
Actions
Perianal abscess 1
Some Category 1
Some Category 2
Success
79%
?????????
48%
65%
Some Category 3
Do you confirm to retrain the System?
Please check you have curated all the data
Cancel
Confirm
Re-train
< 1 2 3 4 5 ... 60 >

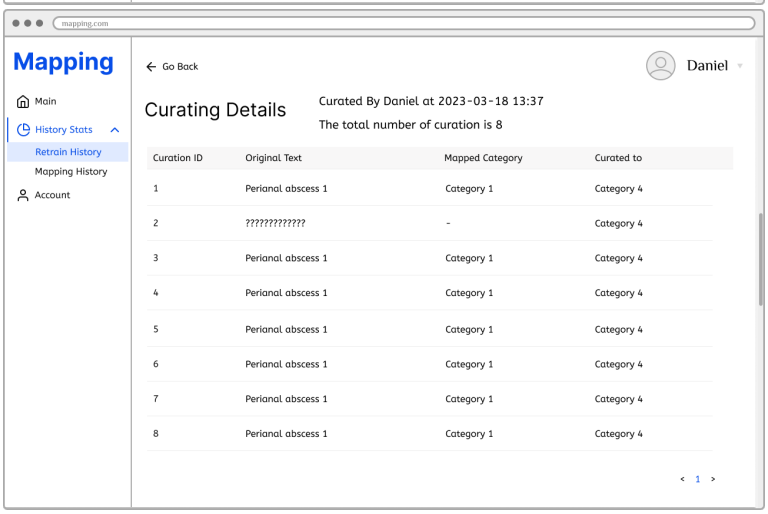

mapping.com
Mapping
Main
History Stats
Account
Go Back
Inference
Training
Mapping Status
Select
Confidence Range
80%
Original Text
Mapping Category
Perianal abscess 1
Some Category 1
?????????
Re-train
View Detail
Original Text: Perianal abscess 1
Mapped Category: Term 1
Confidence
item 1
item 2
item 3
item 4
item 5
68%
13%
8%
7%
4%
0%
10%
20%
30%
40%
50%
60%
70%
Category


mapping.com
Mapping
Main
History Stats
Account
Go Back
Inference
Training
Mapping Status
Select
Confidence Range
80%
Original Text
Mapping Category
Perianal abscess 1
Some Category 1
?????????
Re-train
Overall Performance
Total Mapping Text:
380
Success 325
Fail 55
Successful Mapping Rate
85.52%
Overall Confidence
67.62%
Proportion
Confidence
1%
5%
8%
11%
21%
24%
30%
Proportion of Each Category
Category 4
Category 5
Category 2
Category 7
Category 3
Category 1
Category 6

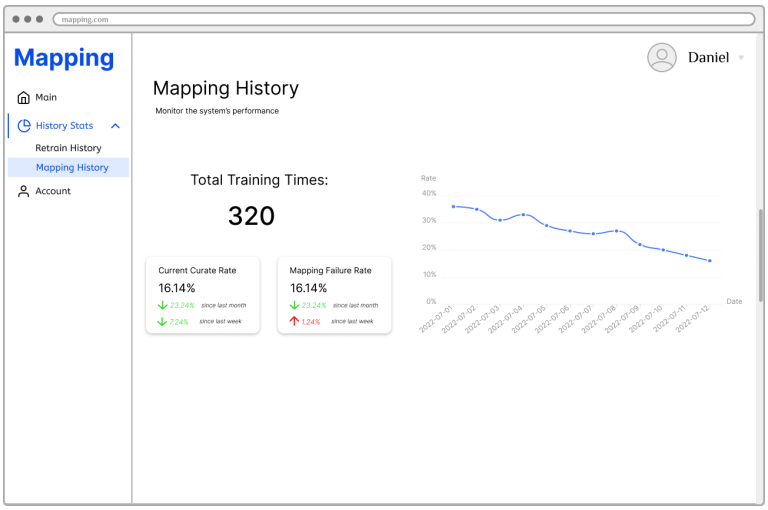


Retrain History

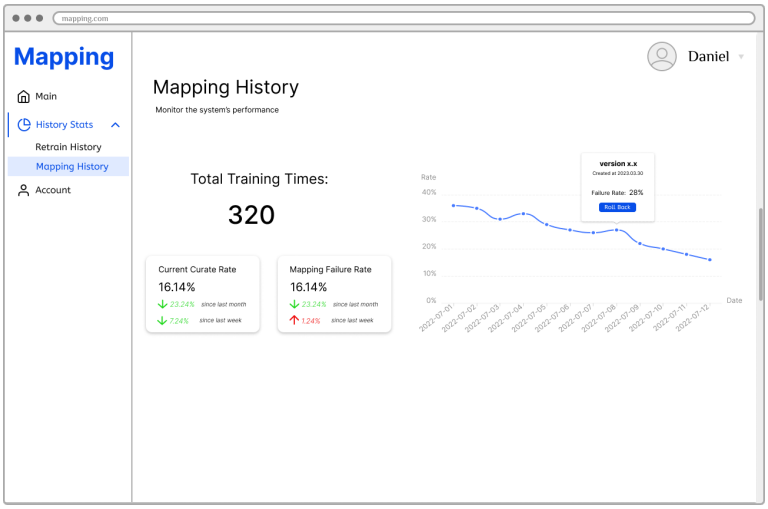




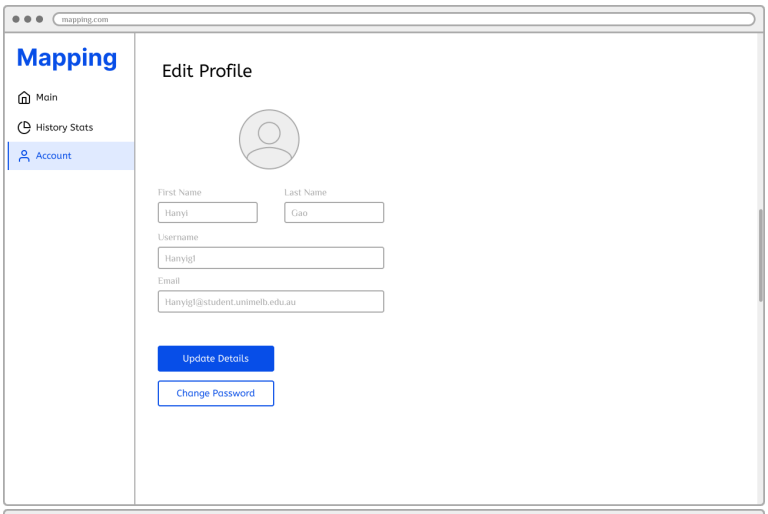
Mapping History

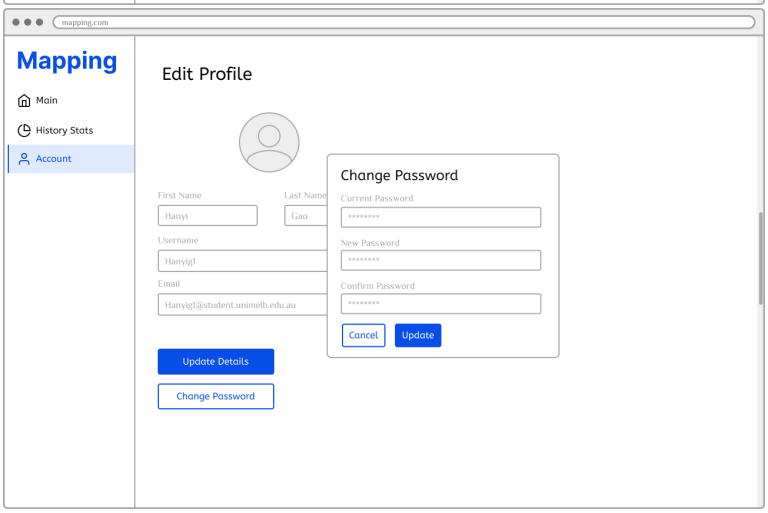


Mapping Versioning (Admin only)

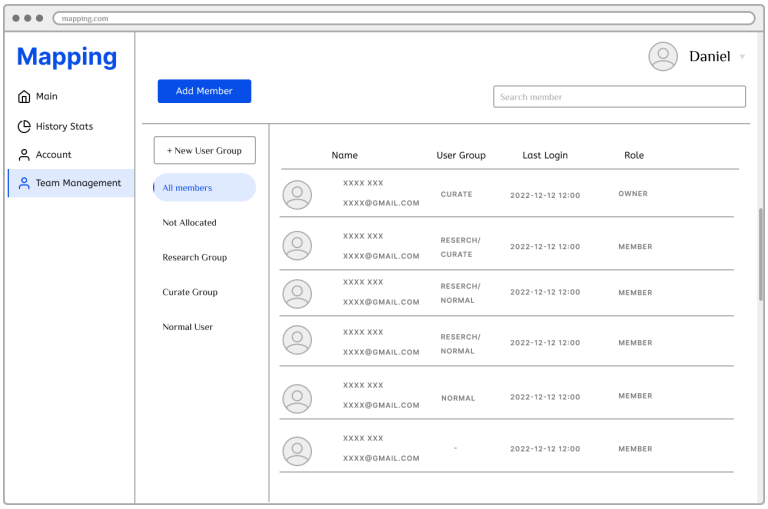


Profile





Team Management (Admin only)



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

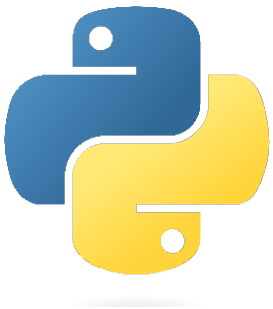
- Sequence diagrams

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

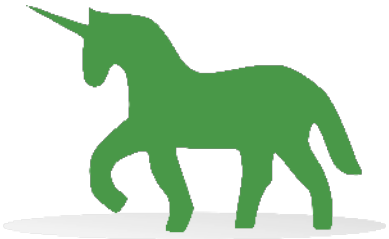
Microservice is an popular architecture in todays software, as it offers various benefits in terms of scalability, maintainability, and flexibility. Microservices can be independently scaled based on their specific resource requirements or load, allowing the project to handle increased demand more efficiently. 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.

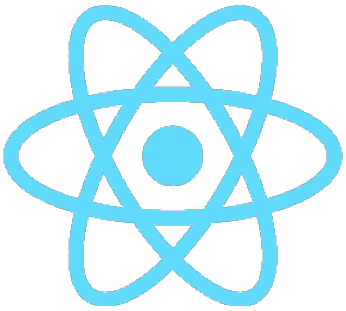
More about Gunicore

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

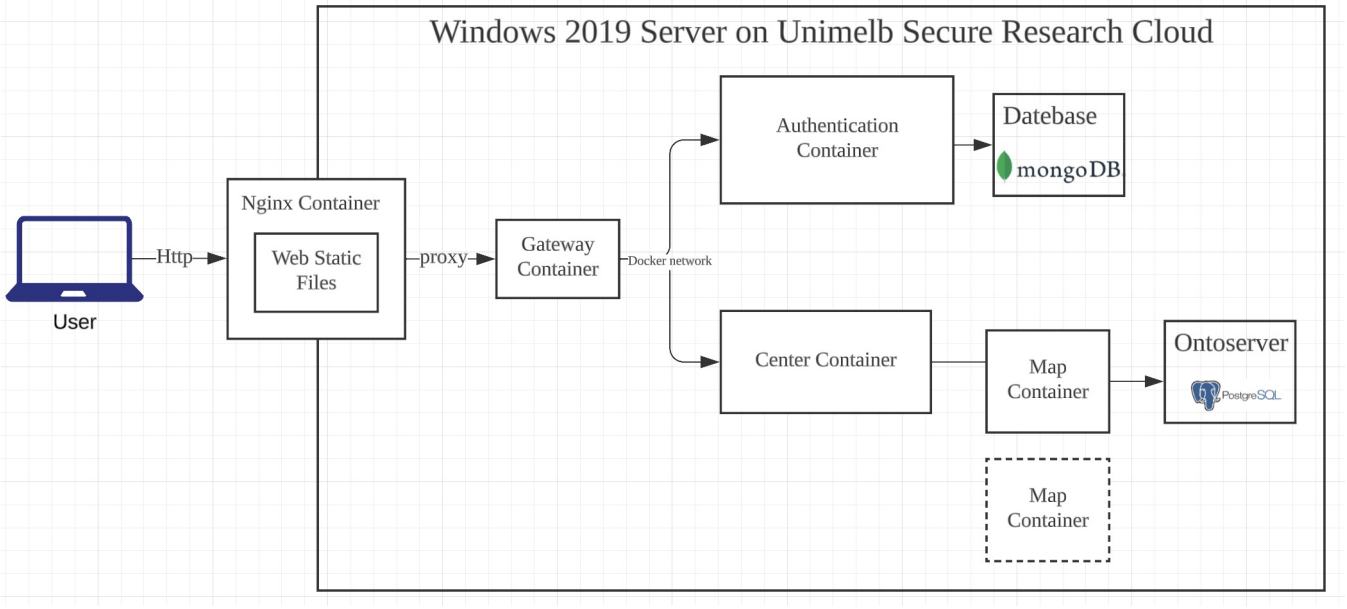
4+1 Architecture Models

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

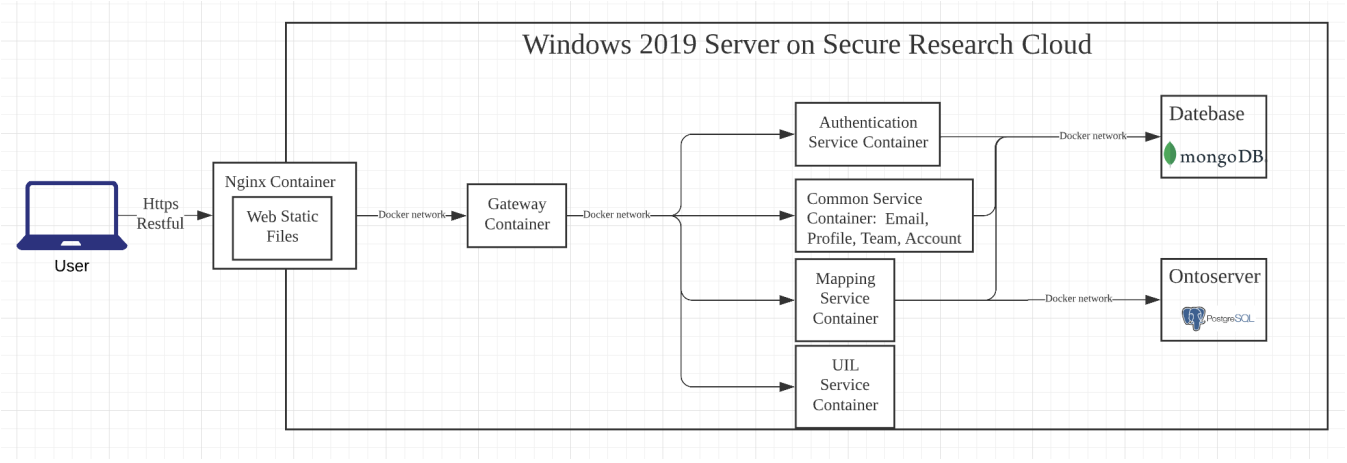
Container Diagram

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



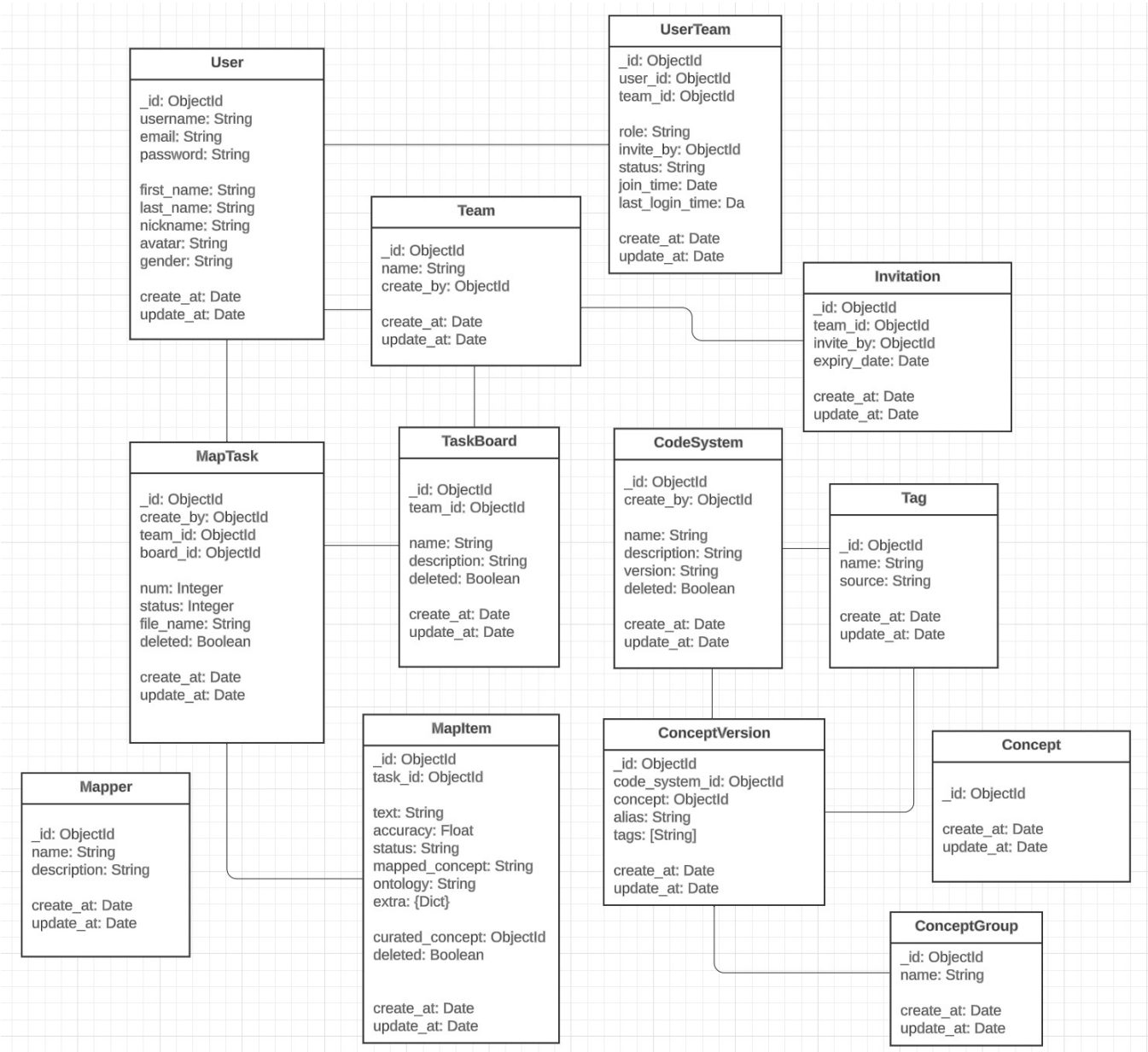
Version 1.0.0



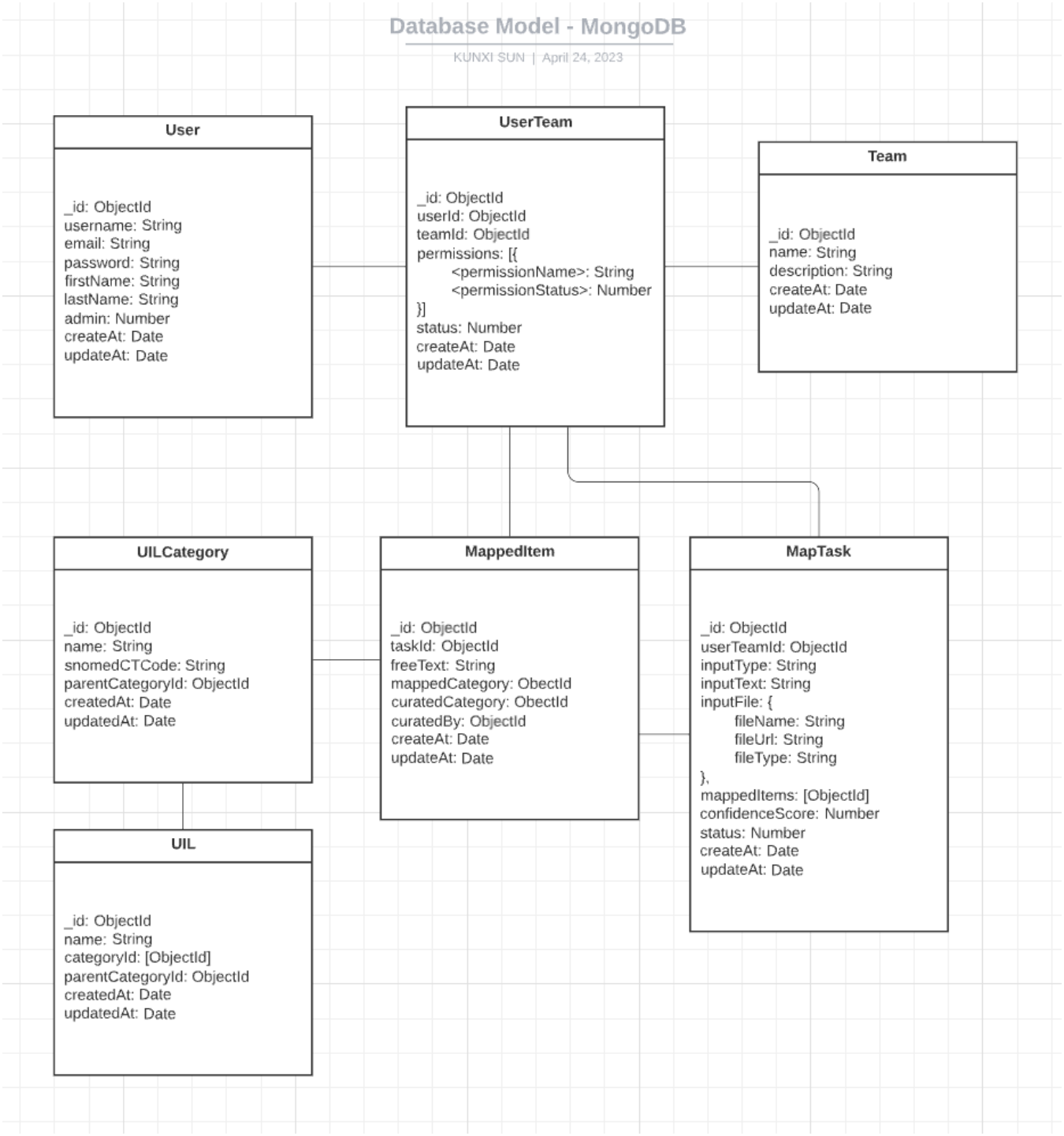
Database Model

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

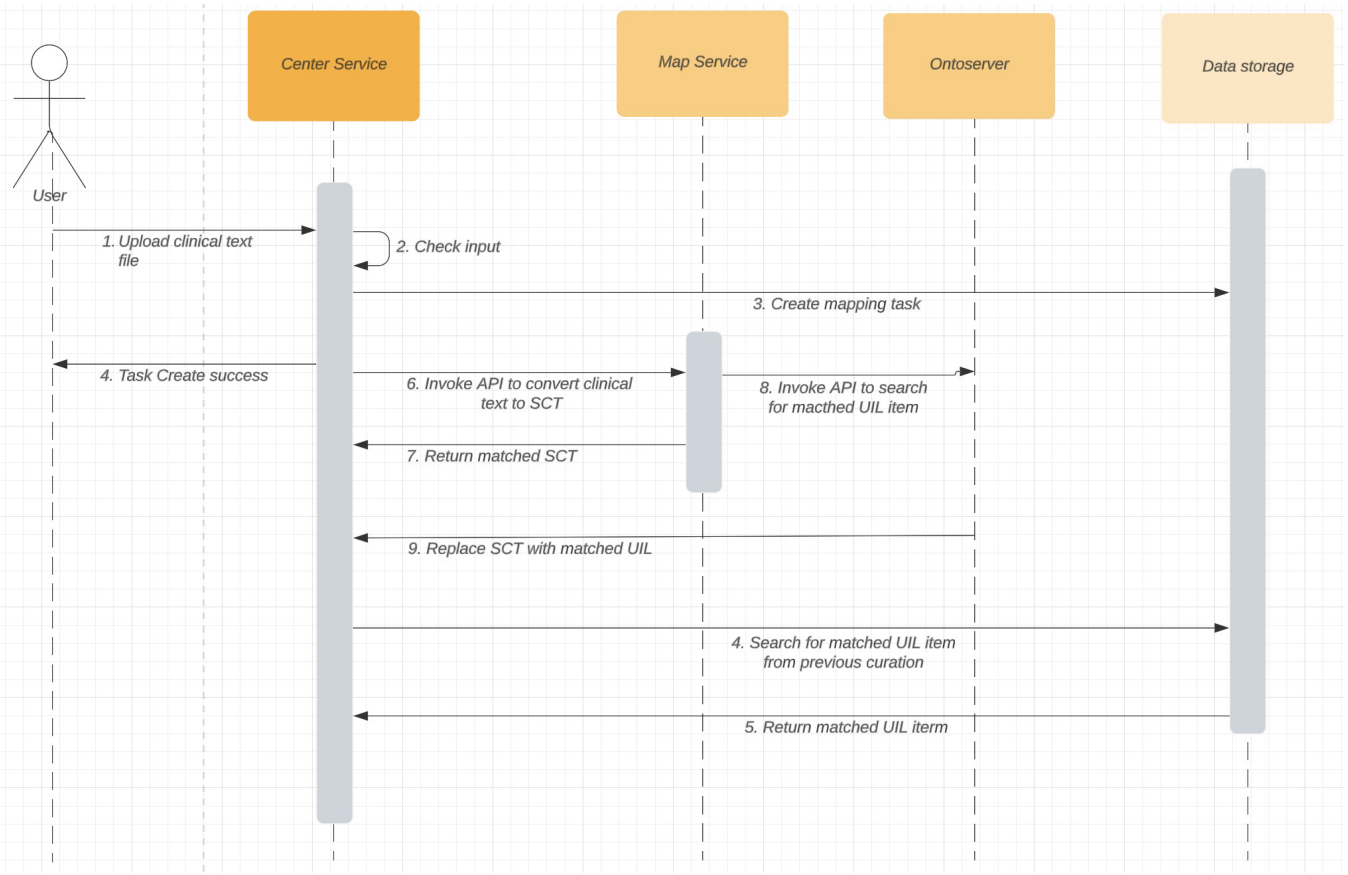

Database Model - MongoDB
KUNXI SUN | April 24, 2023
User
_id: ObjectId
username: String
email: String
password: String
firstName: String
lastName: String
admin: Number
createAt: Date
updateAt: Date
UserTeam
_id: ObjectId
userId: ObjectId
teamId: ObjectId
permissions: [{
<permissionName>: String
<permissionStatus>: 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

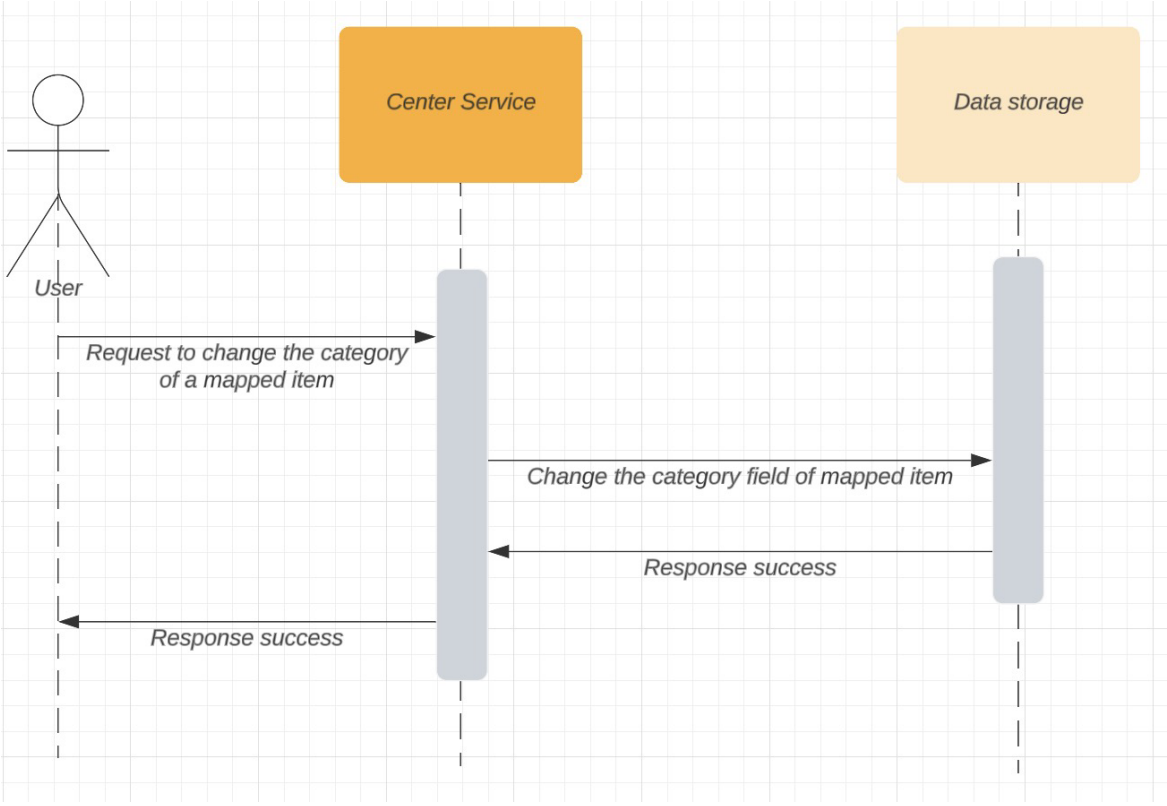
Version	Description	Date	Comment
2.0.0	Change the sequence diagram aligning with the latest system	30 May 2023	1. Update on mapping sequence with Ontoserver service 2. Update on system rollback sequence with new logic. However, this function is not yet implemented due to time limitation.
1.0.0	Basic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

Version 2.0.0

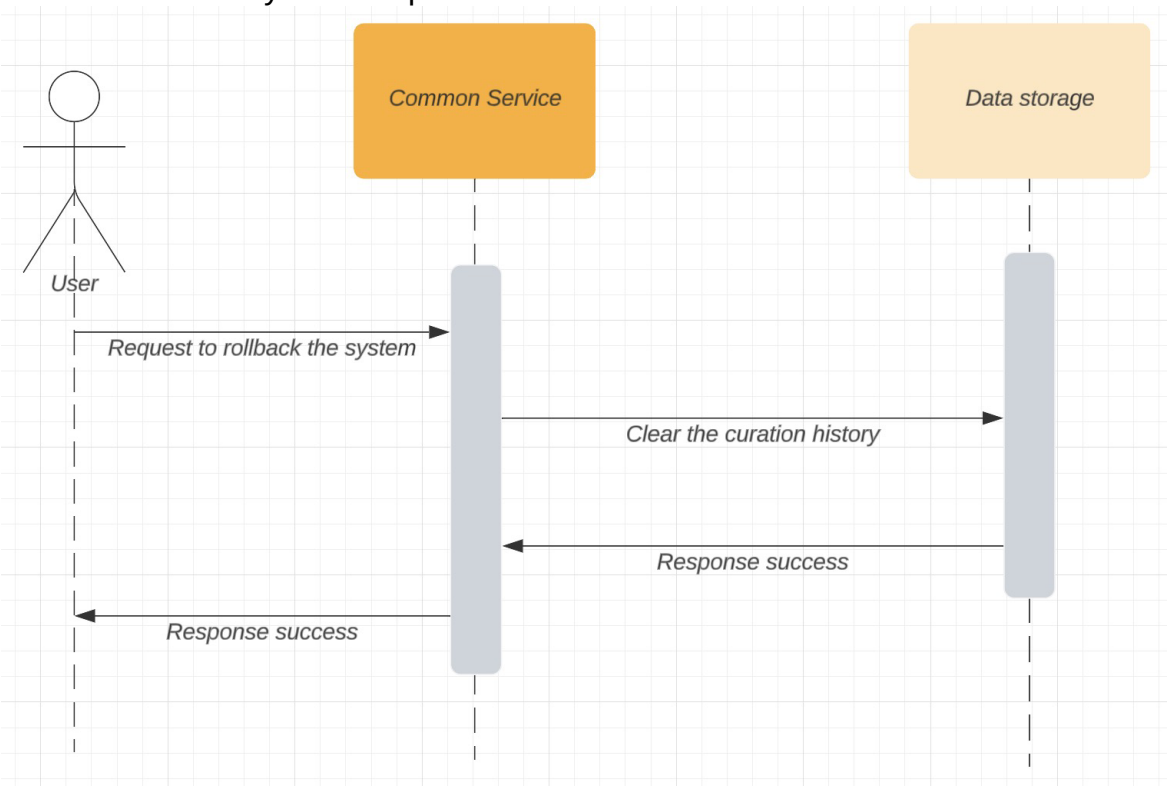
Mapping Sequence



Curate Sequence

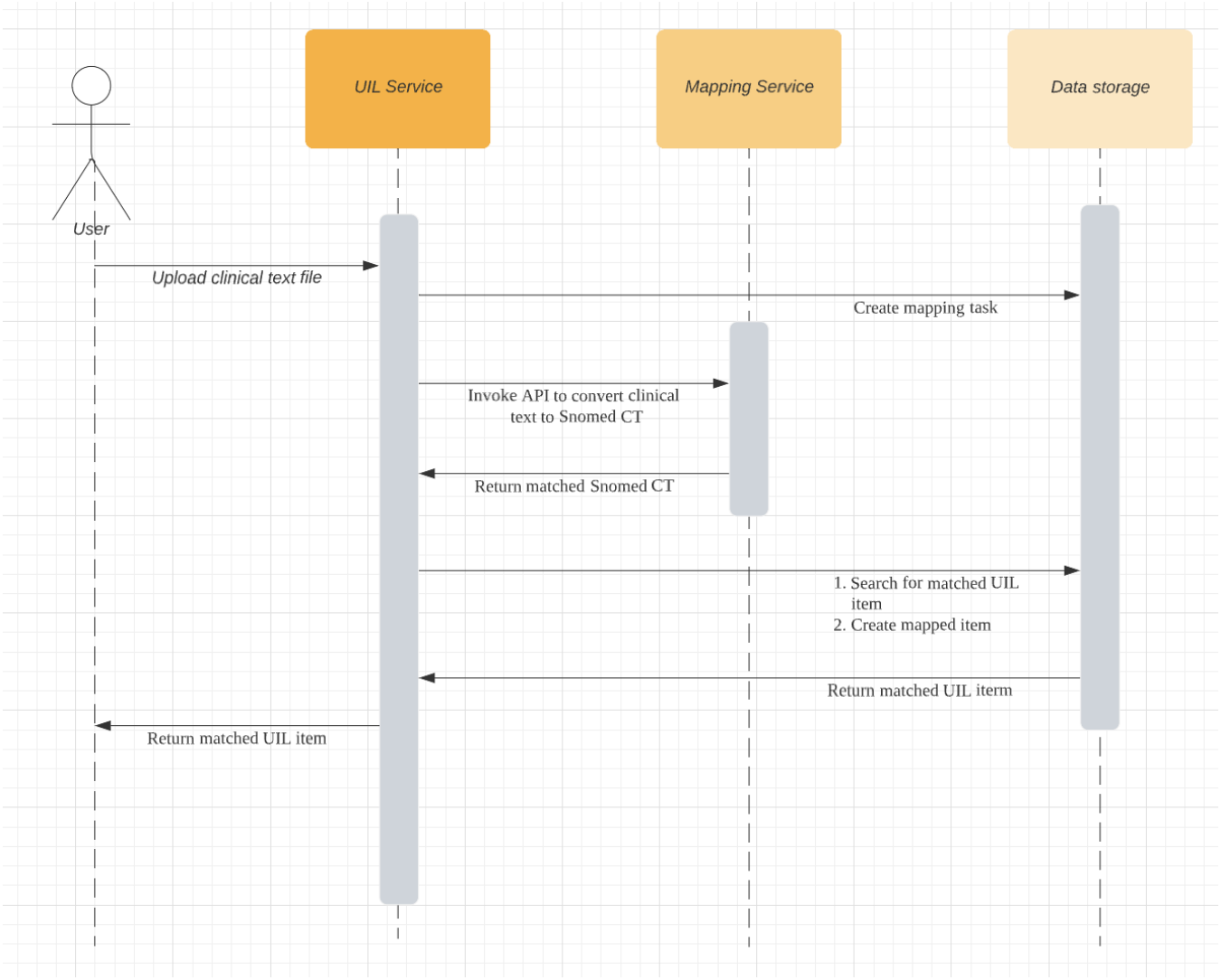


Rollback/Default System Sequence

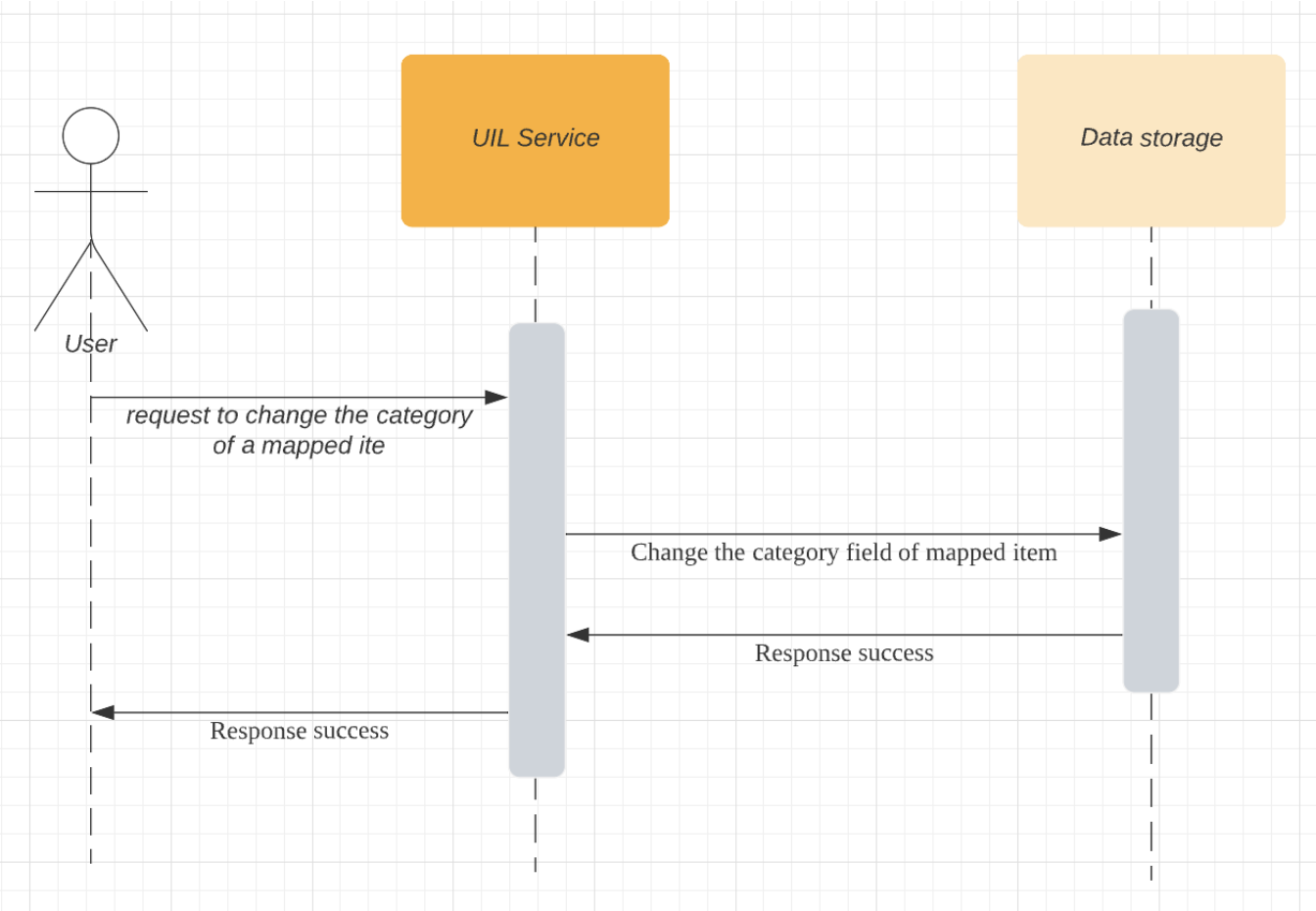


Version 1.0.0

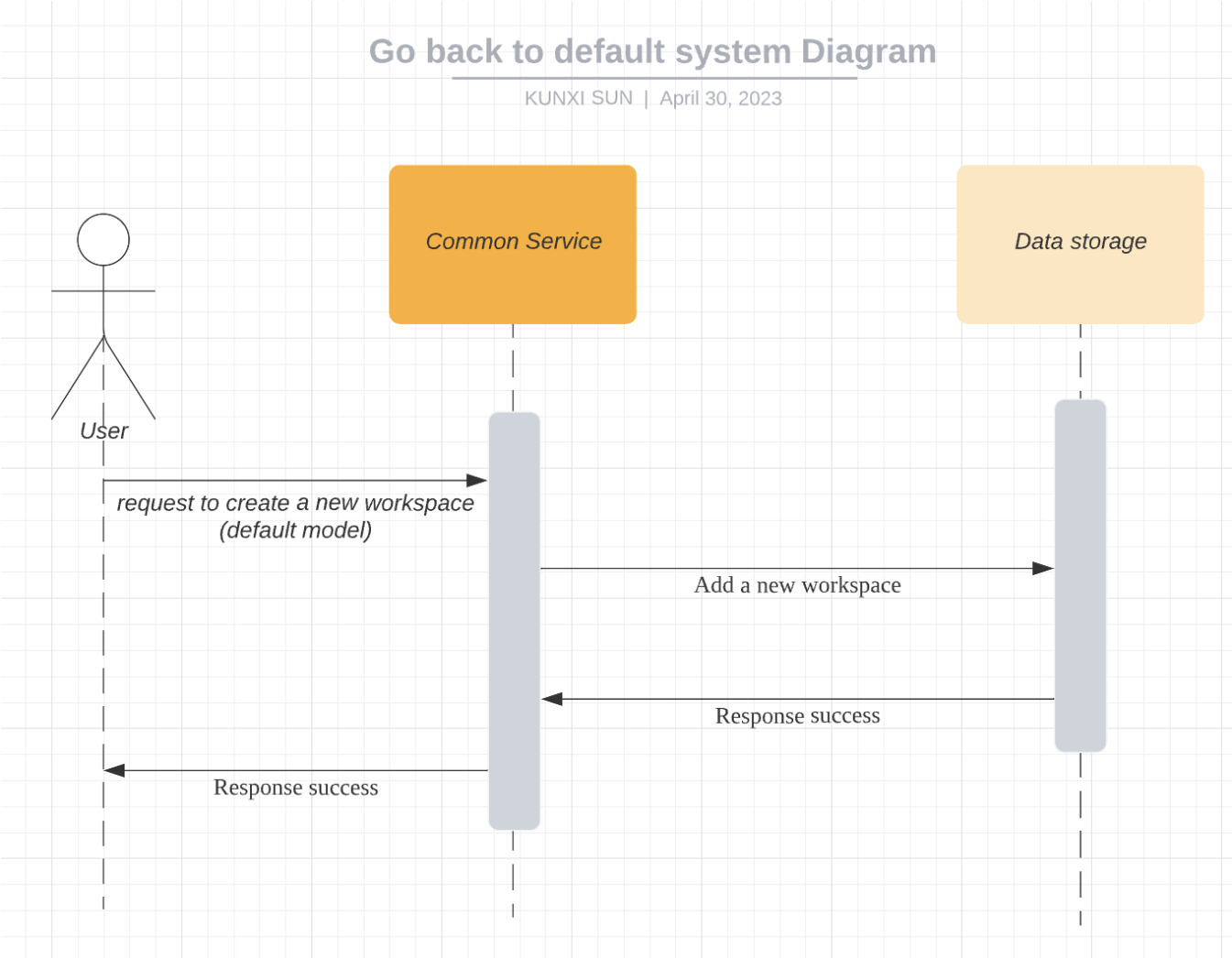
Mapping Sequence



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

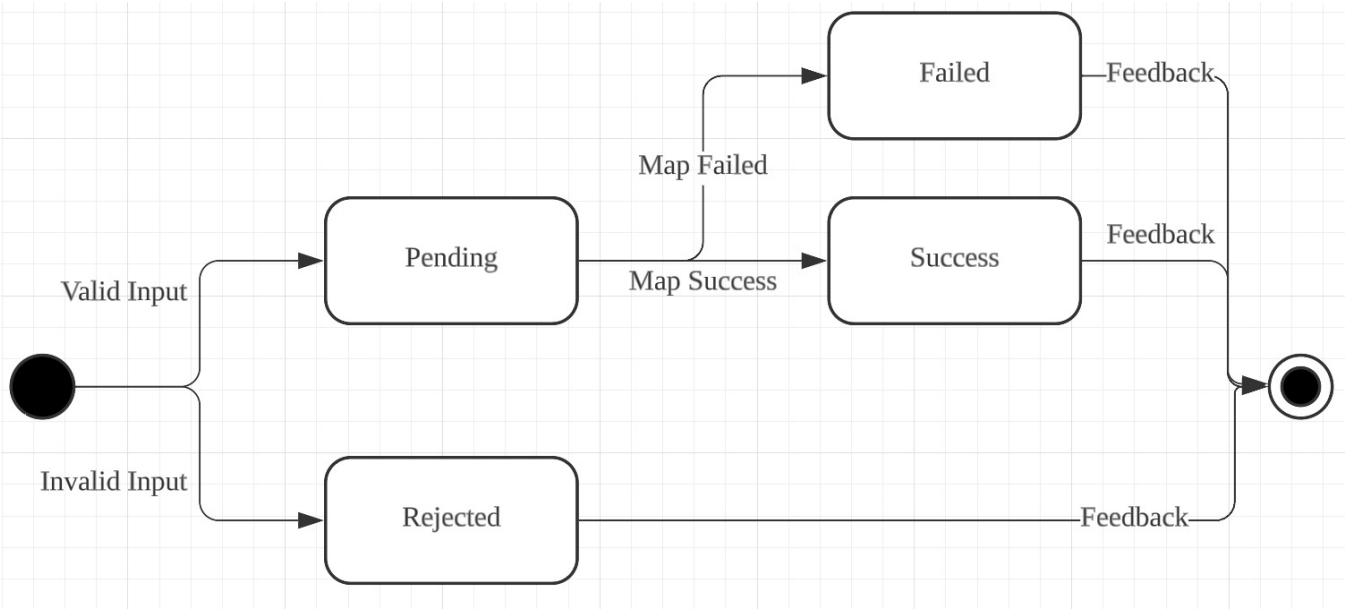
State Diagram

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

Version 1.0.0

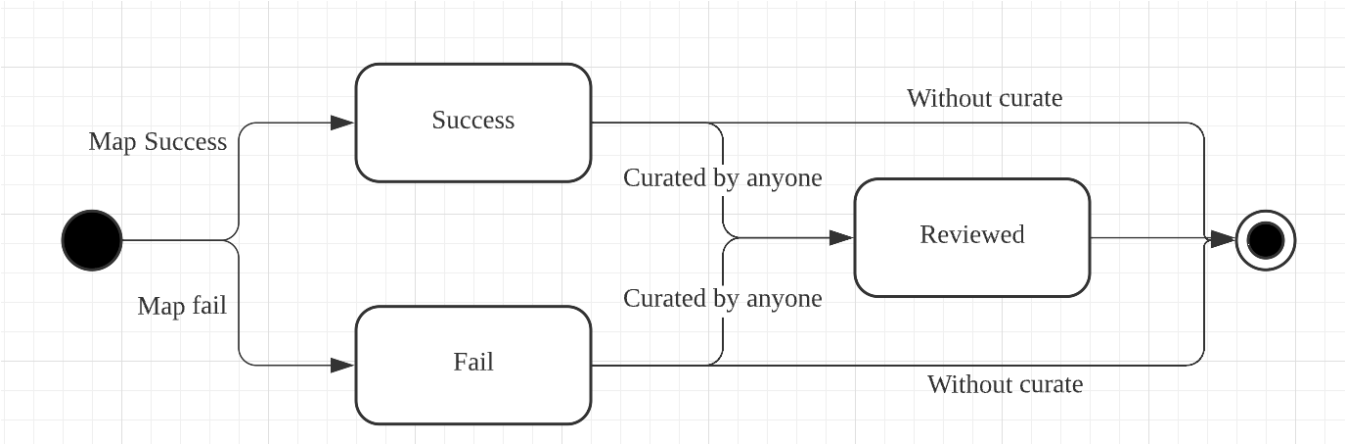
MapTask Status

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

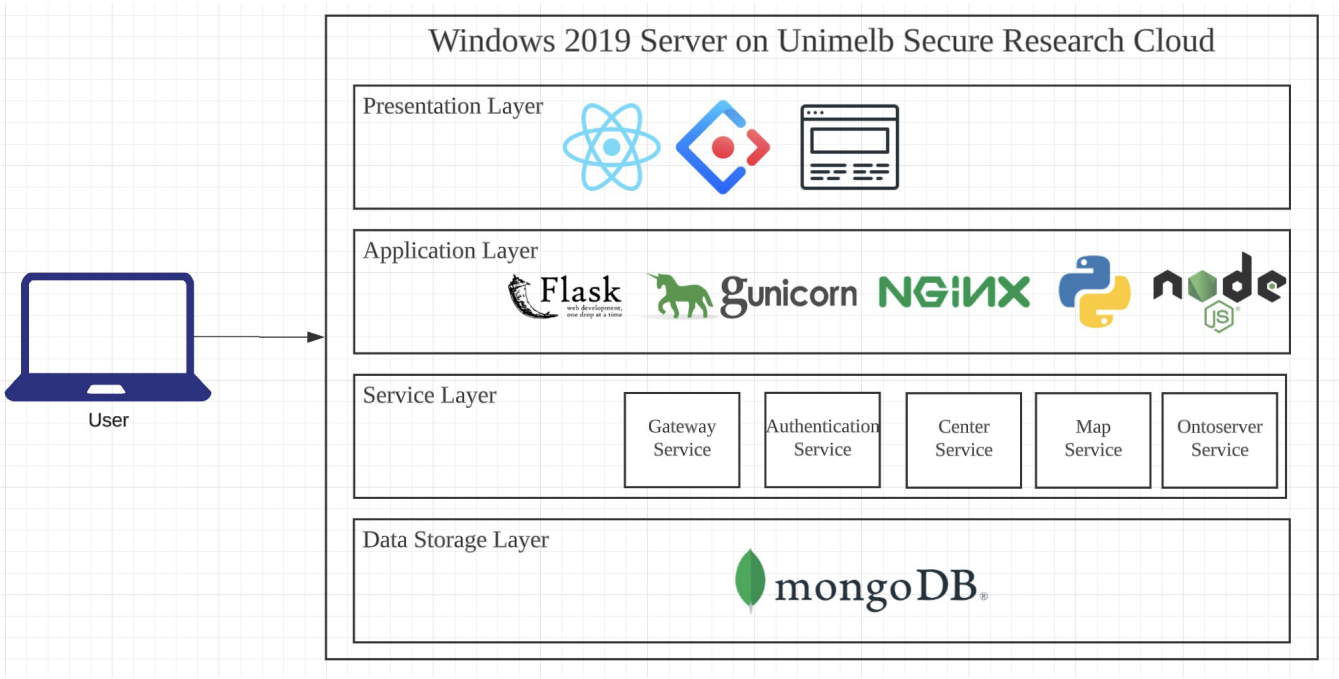
Status Name	Status Number	Description
Fail	0	Mapping system failed to map this raw text
Success	1	Mapping system map this item successfully
Reviewed	2	This raw text has been curated by user


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

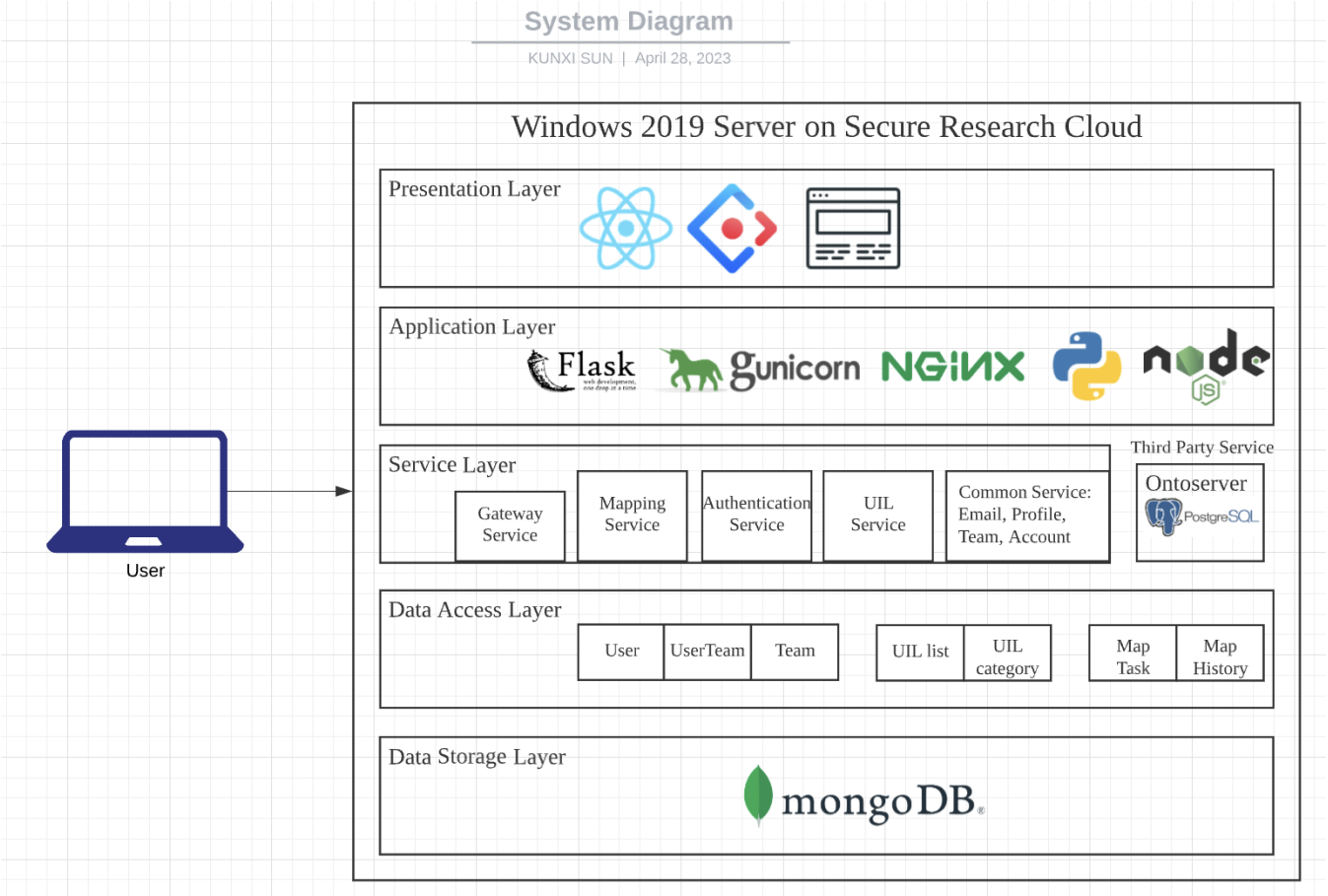
System Diagram

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

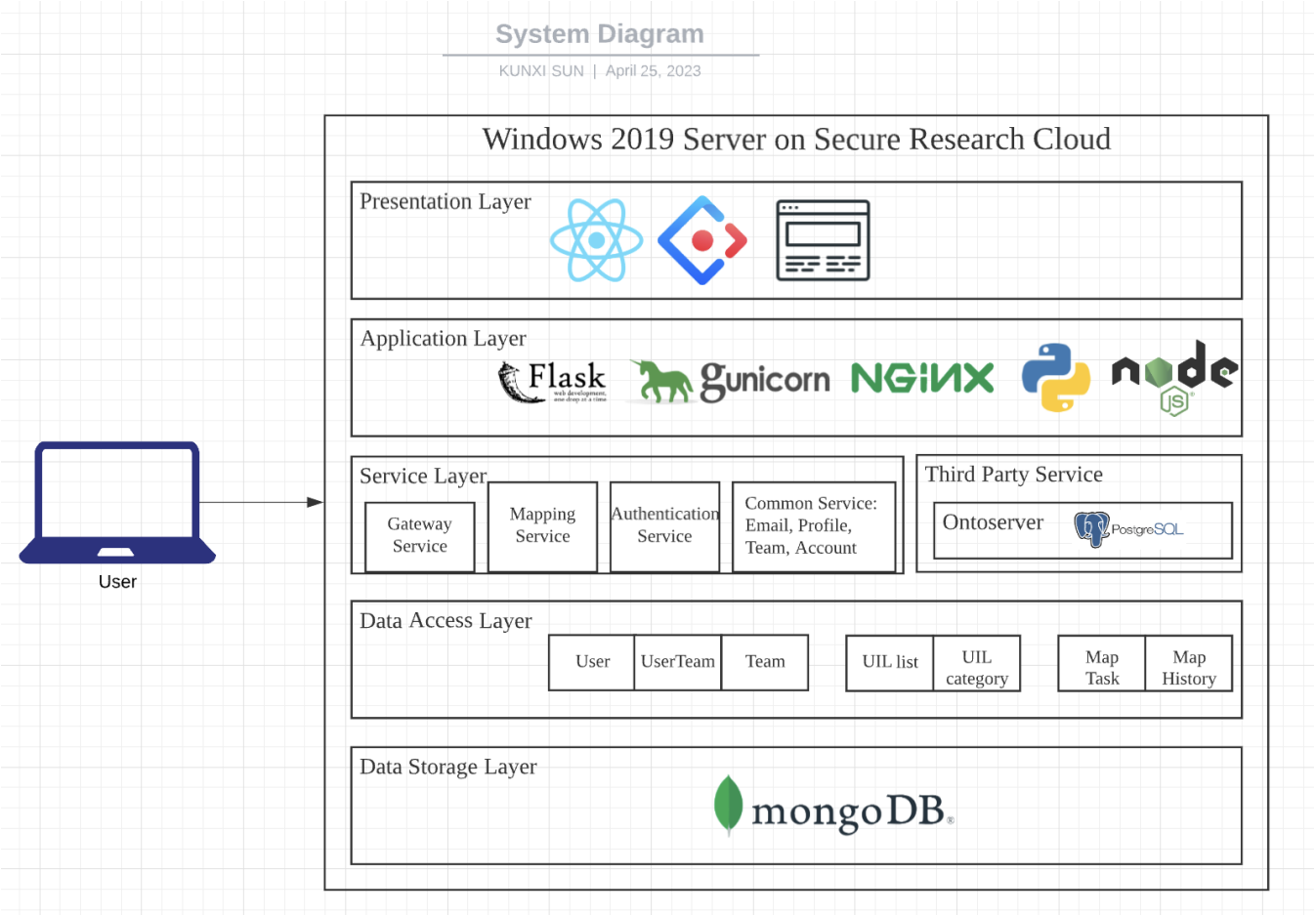
Version 3.0.0



Version 2.1.0

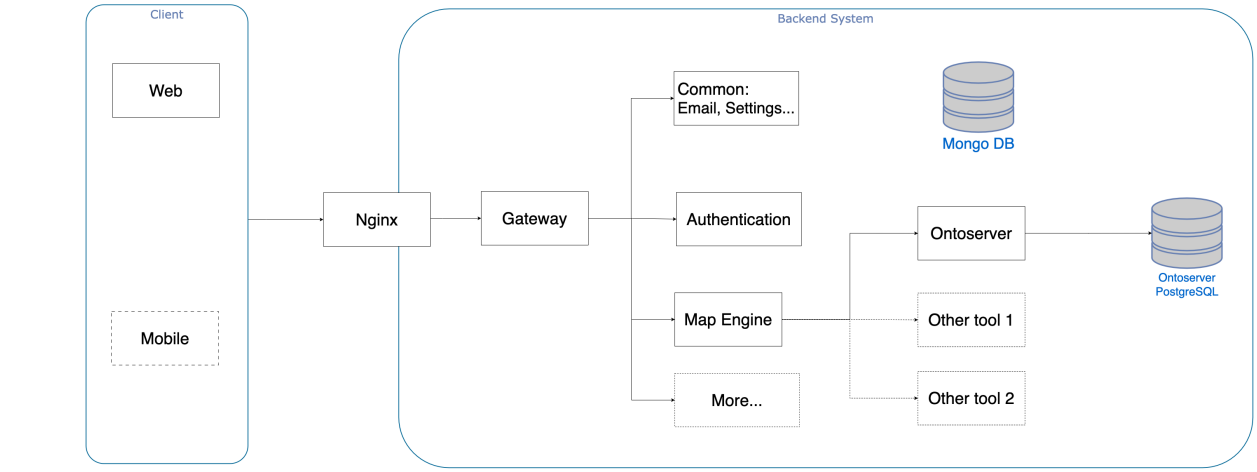

System Diagram
KUNXI SUN | April 28, 2023
Windows 2019 Server on Secure Research Cloud
Presentation Layer
Application Layer
Flask
gunicorn
NGINX
node
Service Layer
Gateway Service
Mapping Service
Authentication Service
UIL Service
Common Service: Email, Profile, Team, Account
Third Party Service
Ontoserver
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



Quality

This page contains documentation related to quality control.

Title	Creator	Modified
User Experience Evaluation	Yue Fei	01 Jun, 2023
Pair programming	KUNXI SUN	01 Jun, 2023
Test Plan	Ricardo Luo	30 Apr, 2023
CI/CD	KUNXI SUN	29 Apr, 2023
Issue Tracking	Hanyi Gao	29 Apr, 2023
Test Cases	KUNXI SUN	29 Apr, 2023
Test Reports	KUNXI SUN	29 Apr, 2023
Code Reviews	KUNXI SUN	29 Apr, 2023

Test Plan

1 Introduction

The objective of this test plan is to outline the testing strategy for the given project, which involves mapping short text into the terms of a Universal Indication List (UIL), curating and updating the mappings, and providing a real-time dashboard to monitor the mapping metrics. The plan will cover Unit Test, Functional Test, Integration Test, and System Test.

The project development will focus on two prevalent development strategy: Behaviour Driven Development(BDD) and Test Driven Development(TDD).

2 Test Objectives

- Ensure that all user stories are implemented correctly
- Verify that the project meets the functional and non-functional requirements
- Validate that the system performs as expected under different conditions
- Identify any defects or issues in the software

3 Test Strategy

3.1 Unit Test

Unit tests will focus on testing individual components, functions, and classes in isolation. These tests will help identify issues at the code level.

1. Map short text into the terms of UIL (US0001)
    - Test input validation for short texts
    - Test mapping of different types of short texts to UIL terms
2. Download the mapping result (US0002)
    - Test downloading of different file formats (e.g. CSV, JSON)
3. Identify unrecognized results in the mapping process (US0004)
    - Test the identification of unrecognized results in different situations
4. Filter incorrect mappings by confidence (US0005)
    - Test filtering functionality with different confidence thresholds
5. Curate incorrect mapping (US0006)
    - Test the editing and updating of incorrect mappings
6. View category options for mapped items (US0007)
    - Test the display and navigation of category options
7. Retrain the system from curating (US0008)
    - Test the retraining process after curating incorrect mappings
8. Update the version of UIL (US0025)
    - Test updating the UIL version and ensure backward compatibility
9. Account login (US0019)
    - Test user authentication and validation
10. Edit personal information (US0020)
    - Test the editing and updating of user information
11. Add new user account (US0021)
    - Test the creation of new user accounts
12. Add member to team (US0023)
    - Test adding members to the team
13. Remove member from team (US0024)
    - Test removing members from the team

3.2 Functional Test

Functional tests will focus on validating the functionality of the project according to the user stories.

1. Test mapping process (US0001, US0004)
    - Test the mapping of short texts to UIL terms with various inputs and scenarios
2. Test mapping results management (US0002, US0005, US0006)
    - Test downloading, filtering, and curating of mapping results
3. Test mapping dashboard (US0010-US0018)
    - Test the display of mapping metrics, filtering options, and performance history

4. Test user management (US0019-US0024)
    - Test user authentication, account creation, and team management

3.3 Integration Test

Integration tests will focus on testing the interaction between different components and modules of the project.

1. Test the interaction between mapping components (US0001, US0004, US0005, US0006)
    - Test the end-to-end process of mapping, identifying unrecognized results, filtering, and curating
2. Test the interaction between dashboard components (US0010-US0018)
    - Test the display and updating of mapping metrics and performance history
3. Test the interaction between user management components (US0019-US0024)
    - Test the integration of user authentication, account creation, and team management

3.4 System Test

System Test will test the complete system, including its performance, reliability, and security. We will focus on testing the complete system as a whole.

1. Test the system's performance under various load conditions
2. Test the system's reliability by simulating different failure scenarios
3. Test the system's security by simulating various attack scenarios
4. Test Environment and Resources

Behaviour Driven Development

Behaviour Driven Development (BDD) is a software development methodology that emphasises collaboration between developers, QA, and non-technical participants in a software project. BDD encourages teams to use conversation and concrete examples to formalise a shared understanding of how the application should behave. Acceptance Criteria are created based on BDD, and corresponding acceptance criteria tests will be held based on the system behaviour. Test plans in BDD use Given-When-Then statements to describe the scenarios and expected outcomes of the system. Our BDD test plan, as well as the acceptance test, is as follows:

1.

Feature: Text Mapping

Scenario: Map short text to UIL categories

Given a short text input

When the system processes the input

Then the short text is mapped to one or more UIL categories

2.

Feature: Download Mapping Results

Scenario: Download the mapping results for future reference

Given a completed mapping task

When the user requests to download the mapping results

Then the user receives a downloadable file containing the mapping results

3.

Feature: Mapping Task History

Scenario: View previous mapping tasks and results

Given a user with previous mapping tasks

When the user accesses their task history

Then the user can view their own and team members' previous mapping tasks and results

4.

Feature: Mapping Process Status

Scenario: Identify the status of the mapping process for raw text input

Given a list of raw text inputs

When the user views the mapping process status

Then the user can quickly determine if further curation or review is needed for each input

5.

Feature: Category Options

Scenario: View category options while curating data

Given a list of mapped items

When the user curates the data

Then the user can view and choose from a list of category options for each item

6.

Feature: Curate Mapping Results

Scenario: Review and curate failed or incomplete mapping results

Given a list of failed or incomplete mapping results

When the user curates the data

Then the user can review and modify the mapping results for accuracy

7.

Feature: Retrain System Scenario: Retrain the system using curated data

Given a set of curated data

When the user retrains to the system

Then the system's mapping performance is improved

8.

Feature: Update UIL Version

Scenario: Update the UIL to the latest version

Given an admin user

When the user updates the UIL version

Then the system has access to the most up-to-date UIL terms

9.

Feature: Dashboard and Metrics

Scenario: Display mapping metrics on a dashboard

Given a completed mapping task

When the user views the dashboard

Then the user can see various mapping metrics for analysis

10.

Feature: Successful Mapping Rate and Mapped Items

Scenario: View the successful mapping rate and the number of mapped items

Given a completed mapping task

When the user views the dashboard

Then the user can see the successful mapping rate and the number of mapped items

11.

Feature: Overall Confidence

Scenario: View the overall confidence in mapping results

Given a completed mapping task

When the user views the dashboard

Then the user can see the overall confidence in the mapping results

12.

Feature: Category Performance Metrics

Scenario: View specific performance metrics for each category

Given a completed mapping task

When the user views the dashboard

Then the user can see the specific performance of each category

13.

Feature: Proportion of Each Category

Scenario: View the proportion of each category in the dataset

Given a completed mapping task

When the user views the dashboard

Then the user can see the proportion of each category in the dataset

14.

Feature: Visualize Mapping Performance History

Scenario: Visualize the history of the mapping performance

Given a user with previous mapping tasks

When the user views the mapping performance history

Then the user can visualize the historical mapping performance for analysis

15.

Feature: Rollback System Version

Scenario: Roll back to an earlier system version

Given an admin user

When the user initiates a rollback to an earlier version

Then the system is restored to the specified earlier version

16.

Feature: Account Login

Scenario: Log in to a user account

Given a registered user

When the user logs in with their credentials

Then the user is granted access to their account

17.

Feature: Edit Personal Information

Scenario: Update personal information, roles, and responsibilities

Given a logged-in user

When the user edits their personal information

Then the system updates the user's roles and responsibilities accordingly

18.

Feature: Add New User Account

Scenario: Add a new user account to the system

Given an admin user

When the admin creates a new user account

Then the new user account is added to the system and the new user can log in

19.

Feature: Add Member to Team

Scenario: Add a user account to a team

Given an admin user and a user account

When the admin adds the user account to a team

Then the user account becomes a member of the specified team

20.

Feature: Remove Member from Team

Scenario: Remove a user account from a team

Given an admin user and a user account that is part of a team

When the admin removes the user account from the team

Then the user account is no longer part of the specified team and loses team privileges

Test Driven Development

Test Driven Development (TDD) is a software development methodology that involves writing tests before writing the code to be tested. The test plan for your project will outline the testing requirements and steps for each feature or user story. Our TDD test plan is as follows:

1. **Text Mapping:**
    - Test: Map a short text input to one or more UIL categories.
    - Requirement: Input validation, UIL mapping function, and mapping result.
2. **Download Mapping Results:**
    - Test: Download the mapping results as a file.
    - Requirement: Completed mapping task, download functionality, and file format.
3. **Mapping Task History:**
    - Test: Access and view previous mapping tasks and results.
    - Requirement: User authentication, task history storage, and display functionality.
4. **Mapping Process Status:**
    - Test: Identify the status of the mapping process for raw text input.
    - Requirement: Mapping status determination and display functionality.
5. **Category Options:**
    - Test: View and select category options while curating data.
    - Requirement: Category list, display functionality, and selection functionality.
6. **Curate Mapping Results:**
    - Test: Review and modify failed or incomplete mapping results.
    - Requirement: List of failed or incomplete mapping results, display functionality, and curation functionality.
7. **Retrain System:**
    - Test: Retrain the system using curated data to improve performance.
    - Requirement: Curated data, machine learning model, and retraining functionality.
8. **Update UIL Version:**
    - Test: Update the UIL to the latest version.
    - Requirement: Admin authentication, UIL version management, and update functionality.
9. **Dashboard and Metrics:**
    - Test: Display various mapping metrics on a dashboard.
    - Requirement: Completed mapping task, dashboard functionality, and metric calculation.
10. **Successful Mapping Rate and Mapped Items:**
    - Test: View the successful mapping rate and the number of mapped items.
    - Requirement: Completed mapping task, dashboard functionality, and metric calculation.
11. **Overall Confidence:**
    - Test: View the overall confidence in mapping results.
    - Requirement: Completed mapping task, dashboard functionality, and metric calculation.
12. **Category Performance Metrics:**
    - Test: View specific performance metrics for each category.
    - Requirement: Completed mapping task, dashboard functionality, and metric calculation.
13. **Proportion of Each Category:**
    - Test: View the proportion of each category in the dataset.
    - Requirement: Completed mapping task, dashboard functionality, and metric calculation.
14. **Visualize Mapping Performance History:**
    - Test: Visualize the history of mapping performance.
    - Requirement: Mapping task history, visualization functionality, and metric calculation.
15. **Rollback System Version:**
    - Test: Roll back to an earlier system version.
    - Requirement: Admin authentication, version management, and rollback functionality.
16. **Account Login:**
    - Test: Log in to a user account.
    - Requirement: User authentication, login functionality, and account management.
17. **Edit Personal Information:**
    - Test: Update personal information, roles, and responsibilities.
    - Requirement: User authentication, personal information management, and role management.
18. **Add New User Account:**
    - Test: Add a new user account to the system.
    - Requirement: Admin authentication, account creation functionality, and account management.
19. **Add Member to Team:**
    - Test: Add a user account to a team.
    - Requirement: Admin authentication, team management functionality, and account management.
20. **Remove Member from Team:**
    - Test: Remove a user account from a team.
    - Requirement: Admin authentication, team management functionality, and account management.

For each feature or user story, you should create tests that check for both successful and unsuccessful outcomes, including edge cases and unexpected inputs. These tests will help ensure that your application is robust, reliable, and meets the requirements of the project. Implementing these tests before writing the code will guide your development and help you focus on creating well-structured and maintainable code.

Test Cases

Title	Creator	Modified
TC0004 - Visualise Mapping Result	KUNXI SUN	25 May, 2023
TC0008 - Security	KUNXI SUN	30 Apr, 2023
TC0007 - Team Management	KUNXI SUN	30 Apr, 2023
TC0005 - Curate Mapping Result	KUNXI SUN	30 Apr, 2023
TC0003 - Mapping Clinical Raw Text	Hanyi Gao	30 Apr, 2023
TC0006 - Rollback to Default System	KUNXI SUN	29 Apr, 2023
TC0002 - Upload & Download files	Hanyi Gao	29 Apr, 2023
TC0001 - Authentication & Authorization	KUNXI SUN	29 Apr, 2023

TC0001 - Authentication & Authorization

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0001-1	Login with a correct email and password	Manage Account	1. Enter user's email 2. Enter user's password 3. Click 'Login' button	1. user is invited in the team	The user has logged into the system successfully. Navigate to the main page	Email: diboxjelly@gmail.com Password: 12345678
TC0001-2	Login with an email that does not exist	Manage Account	1. Enter a non-exsiting email 2. Enter any password 3. Click 'Login' button	-	The user should not log into the system. The system should give a notification telling the user that the email does not exist	Email: email@noexist.com Password: 12345678
TC0001-3	Login with an exist email but wrong password	Manage Account	1. Enter a exsiting email 2. Enter a wrong password 3. Click 'Login' button	1. user is invited in the team	The user should not log into the system. The system should give a notification telling the user that the password is wrong	Email: diboxjelly@gmail.com Password: wrongpassword
TC0001-4	Logout	Manage Account	1. Move mouse to the username on the top right 2. Click 'Sign out' in the dropdown menu	1. TC0001-1	The user has logged out successfully. Navigate to the Login page	

TC0002 - Upload & Download files

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0002-1	Upload a CSV file	Map free text	1. Click the `Browse` button in the upload area 2. Select a CSV file	1. TC0001-1	Only the CSV file can be selected. The selected file name should be rendered in the uploaded area	

TC0003 - Mapping Clinical Raw Text

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0003-1	Map the CSV file in inference mode	Map free text	1. Click the `Inference` radio button 2. Click `Map` button	1. TC0002-1 2. the uploaded CSV file is in a correct format	Navigate to the mapping result inference mode page The table at the mapping result page should show the relevant result	The header of the CSV file should be `Text`
TC0003-2	Map the CSV file in training mode	Map free text	1. Click the `Training` radio button 2. Click `Map` button	1. TC0002-1 2. the uploaded CSV file is in a correct format	Navigate to the mapping result training mode page The table at the mapping result page should show the relevant result	The header of the CSV file should be `Text`
TC0003-3	Map the CSV file that has a wrong format	Map free text	1. Click `Map` button	1. TC0002-1 2. the uploaded CSV file is in a wrong format	Mapping should fail. The system should give a notification that the CSV format is wrong	
TC0003-4	View mapping task history	Map free text	1. Click Retain History on the side bar	1. TC0001-1 2. The team or researcher has had a mapping task before.	History mapping task is shown on the webpage.	

TC0004 - Visualise Mapping Result

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0004-01	Have a dashboard to display the mapping metrics	Mapping Visualisation	1. Click on the dashboard icon	1. TC0004-2 2. Mapping is successful	Dashboard is shown on the right side drawer.	
TC0004-02	View the successful mapping rate and number of mapped items	Mapping Visualisation	1. Click on the dashboard icon	1. TC0004-2 2. Mapping is successful	Mapping rate and number of mapped items are shown on the right side drawer.	
~~TC0004-03~~	~~View the overall confidence~~	~~Mapping~~ ~~Visualisation~~	1. ~~Click on the dashboard icon~~	1. ~~TC0004-2~~ 2. ~~Mapping is successful~~	~~Overall confidence is shown on the right side drawer.~~	
~~TC0004-04~~	~~View the Specific Performance On Each Category~~	~~Mapping~~ ~~Visualisation~~	1. ~~Click on the dashboard icon~~ 2. ~~Then click the confidence button on the right side drawer.~~	1. ~~TC0004-2~~ 2. ~~Mapping is successful~~	~~Specific performance of each category on the right side drawer.~~	
~~TC0004-05~~	~~View the Proportion of Each Category~~	~~Mapping Visualisation~~	1. ~~Click on the dashboard icon~~	1. ~~TC0004-2~~ 2. ~~Mapping is successful~~	~~Ring Chart indicates proportion of each category is shown on the right side drawer.~~	
~~TC0004-06~~	~~Visualize Mapping Performance History~~	~~History Visualisation~~	1. ~~Click on History States on the left side bar~~ 2. ~~Click on the Mapping History below the History States section~~	1. ~~Have successful mapping results before~~	~~Mapping history dashboard is shown on the web page.~~	
~~TC0004-07~~	~~Filter Mapping History by Date Range~~	~~History Visualisation~~	1. ~~Select the date range on the webpage~~	1. ~~TC0004-06~~	~~Mapping History within the selected date range is shown on the web page.~~	
~~TC0004-08~~	~~Filter Mapping History by User~~	~~History Visualisation~~	1. ~~Select the user on the webpage~~	1. ~~TC0004-06~~	~~Mapping History within the selected user is shown on the web page.~~	
TC0004-09	Rollback to Earlier System Version	History Visualisation	1. Click on the roll back button on the mapping diagram	1. TC0004-06	Roll back to the selected version.	

TC0005 - Curate Mapping Result

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0005-1	View Category Options for Mapped Items and curate	Curate the mapping and re-train the system	1. Click `Curate` Icon in the action column 2. Select a category in the dropdown 3. Click `save` to curate the result	TC0003-2	1. Able to see all the UIL categories in the dropdown 2. The status becomes `reviewed` after saving the curated category	
TC0005-2	The system is retrained after curating the result	Curate the mapping and re-train the system	1. Curate a mapping 2. Click `Retrain` button 3. Do the mapping again	TC0005-1	The system is retrained and the new mapping should show the results that are curated last time	

TC0006 - Rollback to Default System

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0006-1	Rollback the mapping system	Implement mapping history visualization	1. Click 'Mapping Stats' section 2. Click 'Mapping History' subsection 3. Click 'Rollback to Default' button beyond mapping history monitoring timeline 4. Click 'Confirm' button	1. user is admin	The user has rollbacked the system successfully. Navigate to the main page.	Email: diboxjelly@gmail.com Password: 12345678

TC0007 - Team Management

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0008-1	Create a team	Manage team	1. Click the "Create Team" button 2. Enter the name of the team 3. Click the "Confirm" button	1. The admin user has successfully logged in	The system shows a new team page, which means that the admin user successfully creates a team.	The team name should not be empty.
TC0008-2	Add a team member to a team	Manage team	1. Select a user 2. Click the "Add to a team" button 3. View the list of all team names 4. Select a team from the team list 5. Click the "Confirm" button	1. The admin user has successfully logged in 2. A team exists 3. The team is not full	A new user appears on the team page, which means that the user has been successfully added to a team.	The admin can only add one user to a team in one operation.
TC0008-3	Remove a team member from a team	Manage team	1. Select a team 2. Select a team member from the team 3. Click the "Remove from the team" button 4. Click the "Confirm" button	1. The admin user has successfully logged in 2. A team exists 3. The team is not empty	The user no longer appears on the team page which means that the user has been successfully removed from the team.	The admin can remove multiple users from a team in one operation.

TC0008 - Security

ID	Description	Category	Test Steps	Prerequisites	Expected Results	Note
TC0008-1	API should be protected	-	1. Without using access token from login URL 2. Use all request method(POST/GET /UPDATE/DELETE) to all API with protected resource	1. Does not log in	1. The API does not return any data without access token	Sensitive data can only be seen after log in
TC0008-2	System can safely rollback /default system setting without losing data	-	1. System create a new workspace (empty mapping history). 2. Check the previous mapping history, and the history did not lose.	1. User log in	1. Check the previous workspace, and the mapping history will not lose.	Data will never lose

Test Reports

Title	Creator	Modified
Performance Test Report	KUNXI SUN	31 May, 2023
2023-05-25 Functional Test	Hanyi Gao	25 May, 2023
2023-04-28 Functional Test	KUNXI SUN	29 Apr, 2023

2023-04-28 Functional Test

Tester	**Hanyi Gao**
Date	27 Apr 2023

Test cases	Test Case ID	Status	Note
Test Cases - Authentication&Authorization	TC0001-1	PASS	
	TC0001-2	FAIL	see ISSUE00010: Login incorrect error message
	TC0001-3	FAIL	see ISSUE00010: Login incorrect error message
	TC0001-4	PASS	
TC0001 - Upload&Download files	TC0002-1	PASS	
TC0003 - Mapping Clinical Raw Text	TC0003-1	PASS	
	TC0003-2	PASS	
	TC0003-3	PASS	
TC0004 - Visualise Mapping Result		DEVELOPING	
TC0005 - Curate Mapping Result	TC0005-1	PASS	
	TC0005-2	DEVELOPING	
TC0007 - Team management		DEVELOPING	
TC0006 - Rollback to Default System	TC0006-1	DEVELOPING	
TC0008 - Security	TC0008-1	PASS	

2023-05-25 Functional Test

Tester	**Hanyi Gao**
Date	25 May 2023

Test cases	Test Case ID	Status	Note
Test Cases - Authentication&Authorization	TC0001-1	PASS	
	TC0001-2	PASS	
	TC0001-3	PASS	
	TC0001-4	PASS	
TC0001 - Upload&Download files	TC0002-1	PASS	
TC0003 - Mapping Clinical Raw Text	TC0003-1	PASS	
	TC0003-2	PASS	
	TC0003-3	PASS	
TC0004 - Visualise Mapping Result	TC0004-01	PASS	
	TC0004-02	PASS	
TC0005 - Curate Mapping Result	TC0005-1	PASS	
	TC0005-2	PASS	
TC0007 - Team management	TC0007-2	PASS	
	TC0007-3	PASS	
TC0006 - Rollback to Default System	TC0006-1	DEVELOPING	
TC0008 - Security	TC0008-1	PASS	

Performance Test Report

Tester	**Yue Fei**
Technology	Macbook Pro M1
Date	04 May 2023

1.Introduction

The purpose of this performance test is to assess the speed and efficiency of MedCAT's `get_entities` function in conjunction with multiprocessing. The `get_entities` function is a key component of MedCAT, responsible for extracting relevant entities and concepts from clinical text data. By evaluating the execution time of this function using both single-process and multi-process approaches, we aim to determine the potential performance gains achieved through multiprocessing.

The performance test will focus on measuring the time taken by MedCAT's `get_entities` function to process a given set of text data. The comparison between single-process and multi-process execution will enable us to assess the benefits of parallel processing and evaluate the scalability of MedCAT in handling larger workloads.

The outcomes of this performance test will provide valuable insights into the efficiency of MedCAT's `get_entities` function and the effectiveness of multiprocessing as a technique to improve processing speed. These findings will aid in optimizing the utilization of MedCAT for clinical text analysis, potentially leading to enhanced productivity and reduced processing times for healthcare-related applications.

2.Test Objectives

The primary objectives of the performance testing were to measure and compare the execution time of MedCAT's `get_entities` function using both single-process and multi-process approaches. The specific goals were as follows:

1. Execution Time Measurement:
    - Determine the time taken by MedCAT's `get_entities` function to process a given set of text data using a single-process approach.
    - Measure the execution time of the `get_entities` function accurately to assess the performance of MedCAT.
2. Single-Process Approach:
    - Evaluate the performance of MedCAT when processing the test data sequentially within a single process.
    - Measure and record the execution time of the `get_entities` function for the entire dataset to establish a baseline performance metric.
3. Multi-Process Approach:
    - Assess the performance improvement achieved by employing multiprocessing to parallelize the execution of the `get_entities` function.
    - Measure the execution time of the `get_entities` function when the workload is distributed across multiple processes.
4. Performance Comparison:
    - Compare the execution times of the single-process and multi-process approaches.
    - Identify any significant differences in execution time and assess the efficiency gains achieved through multiprocessing.

By achieving these objectives, the performance test aimed to provide insights into the efficiency and scalability of MedCAT's `get_entities` function under different processing paradigms. The results would inform decision-making regarding the optimal approach to leverage the function's capabilities while optimizing resource utilization and reducing processing time.

3.Test Environment

MedCAT Version	v1.7.0

4.Test Data

Two sets of test data were used for the performance testing:

1. 100 Texts:
    - This dataset consisted of 100 individual texts.
    - The texts were carefully selected to cover a range of scenarios and variations in text length.
2. 100,000 Texts:
    - This dataset comprised 100,000 individual texts.
    - The texts were designed to reflect the diversity and complexity of real-world text data.

By using these two datasets, we aimed to assess the performance and scalability of MedCAT's `get_entities` function across varying text volumes. The smaller dataset (100 texts) helped evaluate the performance of MedCAT on a smaller scale, while the larger dataset (100,000 texts) provided insights into the performance of MedCAT in handling more significant workloads.

5.Test Methodology

1. Single-Process Testing:
    - Implement the single-process approach by calling MedCAT's `get_entities` function on the test dataset within a single process.
    - Measure the execution time of the `get_entities` function using a timer or performance profiling tools.
    - Repeat the single-process testing multiple times to capture variations in execution time and establish a reliable baseline.
2. Multi-Process Testing:
    - Employ multiprocessing to parallelize the execution of the `get_entities` function across multiple processes using the multiprocessing command.
    - Measure the overall execution time of the multi-process approach, considering the time taken by each process and any synchronization overhead.

6. Test results

Based on the testing result, multiprocessing significantly speed up the processing for both large number of text and small number of text.

These results clearly indicate that multiprocessing significantly enhances the performance of MedCAT's `get_entities()` function, regardless of the number of texts being processed. The processing time for both small and large datasets is considerably reduced when using multiprocessing compared to the single-process approach.

By leveraging the capabilities of multiprocessing, the processing time for 100 texts was reduced by a factor of approximately 15 (from 0.75 seconds to 0.05 seconds), while for 100,000 texts, the processing time was reduced by a factor of approximately 120 (from 636.90 seconds to 5.30 seconds).

These findings demonstrate the effectiveness of multiprocessing in improving the efficiency and speed of MedCAT's text processing capabilities. Utilizing multiprocessing can significantly reduce the time required for analyzing clinical text data, leading to enhanced productivity and streamlined workflows in healthcare-related applications.

Text number	Function used	Time Used(s)
100	get_entities()	0.75
100000	get_entities()	636.90
100	multiprocessing	0.05
100000	multiprocessing	5.30

Table1 : Table of the processing time for get_entities and multiprocessing function

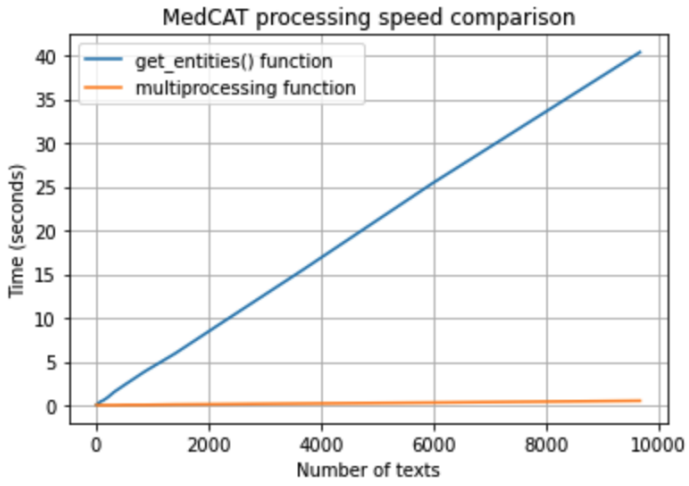


Figure 1: Plot of the processing time for get_entities and multiprocessing function

Acceptance Test

User Story ID	Acceptance Test	Step	Critical	Actor	Comment
US0001	Summer logs into the Digital Health website;	US0001-1	Yes	Summer Taylor (Assume Summer is the member role) Dr. John Kim (Assume Dr. John Kim is the 'owner' role) Kelly Underwood (Assume Dr. John Kim is the member role)	
	Summer opens her dashboard;	US0001-2	Yes		
	Summer searches for the short text they want to map.	US0001-3	Yes		
	Summer view the map results;	US0001-4	Yes		
US0002	Summer opens the task mapping results on the task card;	US0002-1	Yes		
	Summer click the Export button;	US0002-2	Yes		
	Summer get the CSV file downloaded.	US0002-3	Yes		
US0003	Summer opens the board;	US0003-1	Yes		
	Summer clicks on the task card on the board;	US0003-2	Yes		
	Summer views the historical task;	US0003-3	Yes		
US0004	Summer submits a map task;	US0004-1	Yes		
	Summer opens the task detail page;	US0004-2	Yes		
	Summer will see the status of each mapped item: success/fail/review	US0004-3	Yes		
US0005	Summer submits a map task;	US0005-1	Yes		
	Summer opens the task detail page, and clicks the curate dropdown box	US0005-2	Yes		
	Summer will see a dropdown list of the code system categories	US0005-3	Yes		
US0006	Summer submits a map task;	US0006-1	Yes		
	Summer opens the task detail page;	US0006-2	Yes		
	Summer click the curate dropdown box curate on the right of each mapped item;	US0006-3	Yes		
	Summer selects the mapped item into a UIL category;	US0006-4	Yes		
US0007	Summer submits a map task;	US0007-1	Yes		
	Summer opens the task detail page;	US0007-2	Yes		
	Summer clicks the curate dropdown box curate on the right of each mapped item;	US0007-3	Yes		
	Summer selects the mapped item into a UIL category;	US0007-4	Yes		
	Summer submits similar items again, then the system shows the curated category.	US0007-5	Yes		
US0008	Summer goes to the code system;	US0008-1	Yes		
	Summer adds Code System;	US0008-2	Yes		

	Summer enters new code system information;	US0008-3	Yes		
	Summer uploads the new code system file.	US0008-4	Yes		
US0009	Summer login to the system;	US0009-1	Yes		
	Summer opens her dashboard.	US0009-2	Yes		
US00010	Summer opens her dashboard;	US00010-1	Yes		
	Summer views the successful mapping rate and the number of mapped items.	US00010-2	Yes		
US00011	*Incomplete User story*				
US00012	Summer opens a board;	US00012-1	Yes		
	Summer chooses a historical mapping task;	US00012-2	Yes		
	Summer views the visualisation.	US00012-3	Yes		
US00013	*Incomplete User story*				
US00014	*Incomplete User story*				
US00015	*Incomplete User story*				
US00016	Summer at the login page;	US00016-1	Yes		Summer cannot go to any page without login.
	Summer inputs the correct email and password;	US00016-2	Yes		
	Summer login successfully.	US00016-3	Yes		
US00017	Summer is on the main page and clicks the avatar in the top right corner;	US00017-1	Yes		
	Summer clicks "Your profile" in the dropdown box;	US00017-2	Yes		
	Summer input new profile information after clicking "Edit Profile", and submitting.	US00017-3	Yes		
US00018	Summer clicks the "Members" on the left navigation bar;	US00018-1	Yes		
	Summer clicks the invited member;	US00018-2	Yes		
	Summer pastes the invitation link to the new user;	US00018-3	Yes		
	New user click the invitation link, and a new account will be added after the user submit personal information.	US00018-4	Yes		
US00019	Summer clicks the "Members" on the left navigation bar;	US00019-1	Yes		
	Summer clicks the "invite member";	US00019-2	Yes		
	Summer paste the invitation link to the new user;	US00019-3	Yes		
	New user click the invitation link, and a new account will be added after the user submit personal information.	US00019-4	Yes		
US0020	Dr. John Kim clicks the "Members" on the left navigation bar;	US00020-1	Yes	Dr. John Kim (Dr. John Kim is the 'owner' role)	
	Dr. John Kim clicks "Remove" button on the right of the member, then the member is removed	US00020-2	Yes		

Code Reviews

Title	Creator	Modified
2023-05-31 Sprint 3 ChatGPT-assisted Code Review Report - FrontEnd	Ricardo Luo	31 May, 2023
2023-05-31 Sprint 3 ChatGPT-assisted Code Review Report - BackEnd	Yue Fei	31 May, 2023
2023-04-27 Sprint 2 peer-to-peer Code Review Report - FrontEnd	Yue Fei	26 May, 2023
2023-04-27 Sprint 2 peer-to-peer Code Review Report - BackEnd	KUNXI SUN	26 May, 2023

2023-04-27 Sprint 2 peer-to-peer Code Review Report - BackEnd

Code review information

Date	27 Apr 2023
Reviewer	Yue Fei
Author	KUNXI SUN
Code Repository / Branch	DI-Boxjelly/src/

Code review spreadsheet

Issue ID	Artifact(on Github)	Location	Severity	Type	Defects Category	Description	Fixed by the author?	Verified by the moderator?
BE01-01	DI-Boxjelly/src/di-auth /app/api/register.py	class EmailRegister (Resource) Function post	Trivial	Improvement	Check Defects	Add password strength validation.	No	No
BE01-02	DI-Boxjelly/src/di-auth /app/api/register.py	class EmailRegister (Resource) Function post	Medium	Improvement	Logic Defects	Add failure model of MangoDB.	No	No
BE01-03	DI-Boxjelly/src/di-auth /app/api/register.py	class EmailRegister (Resource) Function post	Trivial	Improvement	Structure Defects	Consider defining a function or class method to break down to the smaller function for validating the length and format of input data.	No	No
BE01-04	DI-Boxjelly/src/di-auth /app/api/login.py	class EmailLogin (Resource) Function post	High	Issue	Check Defects	Consider changing the error code when the email or password is incorrect from 404 or 401 to 200.	No	No
BE01-05	DI-Boxjelly/src/di-common/app/api /email.py	class Mail (Resource)	Trivial	Improvement	Structure Defects	Please delete the dead code.	No	No
BE01-06	DI-Boxjelly/src/di-common/app/api /email.py	class Mail (Resource)	Medium	Improvement	Logic Defects	Add error handling code to catch any exceptions that might be raised while sending the email.	No	No
BE01-07	DI-Boxjelly/src/di-auth DI-Boxjelly/src/di-gateway DI-Boxjelly/src/di-map	All	Trivial	Improvement	Documentation Defects	Add comment to each file.	No	No
BE01-08	DI-Boxjelly/src/di-auth DI-Boxjelly/src/di-gateway DI-Boxjelly/src/di-map	All	Trivial	Improvement	Structure Defects	Please delete dead code.	No	No

Summary and Next Steps

In general, the backend code appears to meet the standards outlined in the checklist, but there are some issues that has been identified and listed above. It would be greatly appreciated if you could kindly take the time to remove any unused code and add comments to the code within the di-common, di-gateway, and di-map folders in order to improve the code's readability and comprehension. Please consider to add some more error handling cases and fix the error code bugs as well.

2023-04-27 Sprint 2 peer-to-peer Code Review Report - FrontEnd

Code review information

Date	27 Apr 2023
Reviewer	Ricardo Luo
Author	Hanyi Gao
Code Repository / Branch	DI-Boxjelly/src/di-web/src/

Code review spreadsheet

Issue ID	Artifact(on Github)	Location	Severity	Type	Defects Category	Description	Fixed by the author?	Verified by the moderator?
FE01-01	DI-Boxjelly/src/di-web/src/App.js	all	Trivial	Improvement	Documentation Defects	Need more comments to describe the aim of the file and functions	No	No
FE01-02	DI-Boxjelly/src/di-web/src/App.js	function App()	Trivial	Improvement	Visual Representation Defects	There are some instances of long lines which may make the code difficult to read.	No	No
FE01-03	DI-Boxjelly/src/di-web/src/modules/Login/index.jsx	all	Trivial	Improvement	Documentation Defects	Need more comments to describe the aim of the file and functions	No	No
FE01-04	DI-Boxjelly/src/di-web/src/modules/Login/index.jsx	return ()	Trivial	Improvement	New Functionality	Change the class attribute to className in the JSX div elements to fix the React-specific syntax.	No	No
FE01-05	DI-Boxjelly/src/di-web/src/modules/Dashboard/index.jsx	all	Trivial	Improvement	Documentation Defects	Need more comments to describe the aim of the file and functions	No	No
FE01-06	DI-Boxjelly/src/di-web/src/modules/Dashboard/index.jsx	all	Trivial	Improvement	New Functionality	Change the class attribute to className in the JSX div elements to fix the React-specific syntax.	No	No
FE01-07	DI-Boxjelly/src/di-web/src/modules/Dashboard/index.jsx	onProfileClick ()	Trivial	Improvement	New Functionality	Implement the profile page navigation in the onProfileClick function, replacing the console.log statement with the appropriate navigation code.	No	No
FE01-08	DI-Boxjelly/src/di-web/src/modules/Mapping/index.jsx	all	Trivial	Improvement	Documentation Defects	Need more comments to describe the aim of the file and functions	No	No
FE01-09	DI-Boxjelly/src/di-web/src/modules/MappingResult/index.js	all	Trivial	Improvement	Documentation Defects	Need more comments to describe the aim of the file and functions	No	No

Summary and Next Steps

Overall, the front-end code quality was excellent and met almost all the criteria and requirements in the checklist. Through the code review, we found some minor issues, but most and the code are easy to fix. No major structural or logical issues were found so far, so hopefully we can keep up the good work.

2023-05-31 Sprint 3 ChatGPT-assisted Code Review Report - BackEnd

Code review information

Date	31 May 2023
Reviewer	Yue Fei
Author	KUNXI SUN Chenyang Dong
Code Repository / Branch	DI-Boxjelly/src/di-auth/ DI-Boxjelly/src/di-center/ DI-Boxjelly/src/di-map/

Feedback

1. Login

a. Documentation Defects:
   a. Naming: The naming of software elements is good. The names are descriptive and follow Python naming conventions.
   b. Comment: The comments are well-written and provide useful information. They explain the purpose of the classes, methods, and arguments. The code examples in the comments are helpful.
b. Visual Representation Defects:
   a. Bracket Usage: The bracket usage is correct.
   b. Indentation: The indentation is consistent and follows Python's recommended style.
   c. Long Line: The code statements are not too long and are easy to read.
c. Structure Defects:
   a. Dead Code: There is no dead code.
   b. Duplication: There is no duplicate code.
d. New Functionality:
   a. Use Standard Method: The code already uses standard methods for the tasks it performs.
e. Resource Defects:
   a. Variable Initialization: All variables are initialized correctly.
   b. Memory Management: There are no apparent memory management issues.
f. Check Defects:
   a. Check User Input: The code validates user input and handles invalid input correctly.
g. Interface Defects:
   a. Parameter: The code correctly handles parameters when calling functions or libraries.
h. Logic Defects:
   a. Compute: The logic appears to be correct and there are no apparent issues.
   b. Performance: The code's performance appears to be good.

Recommended changes:

- None. The code is well-written and follows best practices.

2. Logout

a. Documentation Defects:
   a. Naming:
      - The naming of the class and method is appropriate and descriptive.
      - The naming of the attributes 'mongo' and 'jwt' could be more descriptive.

      b. Comment:
      - The comments are informative and provide examples.
      - The TODO comments should be elaborated upon to provide more information.
b. Visual Representation Defects:
   a. Bracket Usage:
      - The bracket usage is correct.

      b. Indentation:
      - The indentation is consistent and correct.

      c. Long Line:
      - There are no long lines in the code.
c. Structure Defects:
   a. Dead Code:

- There is no dead code in the code.
  b. Duplication:
- There is no duplication in the code.

d. New Functionality:
   a. Use Standard Method:

- There are no single-purpose code statements that require a standardized approach.

e. Resource Defects:
   a. Variable Initialization:

- All variables are correctly initialized.
  b. Memory Management:
- There are no memory management issues in the code.

f. Check Defects:
   a. Check User Input:

- There is no user input in the code.

g. Interface Defects:
   a. Parameter:

- There are no incorrect or missing parameters when calling functions or libraries.

h. Logic Defects:
   a. Compute:

- There are no incorrect logic during system execution.
  b. Performance:
- There are no performance issues in the code.

Recommended Changes:

- Improve the naming of the 'mongo' and 'jwt' attributes to make them more descriptive.
- Elaborate on the TODO comments to provide more information.
- Add more comments to explain the code flow and logic.
- Add error handling for cases where the header does not have the cookie.
- Consider adding more functionality to the code, such as blacklisting the tokens on the server side.

3. Codesystem

a. Documentation Defects:
   a. Naming: The naming of software elements is generally good, with clear and descriptive names used throughout the code.
   b. Comment: The code comments are generally good, with clear explanations provided for the purpose of each section of code.
b. Visual Representation Defects:
   a. Bracket Usage: The bracket usage is correct throughout the code.
   b. Indentation: The indentation is consistent and correct throughout the code.
   c. Long Line: There are no long code statements that hinder readability.
c. Structure Defects:
   a. Dead Code: There is no dead code in the provided code.
   b. Duplication: There is no duplicate code in the provided code.
d. New Functionality:
   a. Use Standard Method: There are no single-purpose code statements that require a standardized approach.
e. Resource Defects:
   a. Variable Initialization: All variables are correctly initialized.
   b. Memory Management: The program's memory usage and management are not evaluated in the provided code.
f. Check Defects:
   a. Check User Input: The validity of user input and its handling are not analyzed in the provided code.
g. Interface Defects:
   a. Parameter: There are no incorrect or missing parameters when calling functions or libraries.
h. Logic Defects:
   a. Compute: There are no incorrect logic during system execution.
   b. Performance: The efficiency of the algorithm used is not evaluated in the provided code.

Recommended changes to improve the quality of the code provided:

- Add more comments to explain the purpose of each section of code in detail.
- Evaluate the program's memory usage and management to ensure that it is efficient and optimized.
- Analyze the validity of user input and its handling to ensure that the code is secure and robust.
- Evaluate the efficiency of the algorithm used to ensure that it is optimized for performance.

4. Versioning

1. Documentation Defects:

- Naming: The names of the imported modules and classes are appropriate and descriptive. However, the variable name "user_id" could be more descriptive, such as "request_user_id".
- Comment: The code has a single comment that explains the purpose of the "get" method.

2. Visual Representation Defects:

- Bracket Usage: The code has correct bracket usage.

    - Indentation: The code has correct indentation.
    - Long Line: The code has no long lines.
3. Structure Defects:
    - Dead Code: The code has no dead code.
    - Duplication: The code has no duplicate code.
4. New Functionality:
    - Use Standard Method: The code does not have any single-purpose code statements that require a standardized approach.
5. Resource Defects:
    - Variable Initialization: The code initializes all the variables correctly.
    - Memory Management: The code does not have any memory management issues.
6. Check Defects:
    - Check User Input: The code does not have any user input validation or handling.
7. Interface Defects:
    - Parameter: The code does not have any incorrect or missing parameters when calling functions or libraries.
8. Logic Defects:
    - Compute: The code does not have any incorrect logic during system execution.
    - Performance: The code does not have any performance issues.

Recommended Changes:

1. Rename the "user_id" variable to a more descriptive name.
2. Add more comments to explain the purpose of the code and functions.
3. None. The code is well-structured and has no dead or duplicate code.
4. None. The code does not have any single-purpose code statements that require a standardized approach.
5. None. The code initializes all the variables correctly and does not have any memory management issues.
6. None. The code does not have any user input validation or handling.
7. None. The code does not have any incorrect or missing parameters when calling functions or libraries.
8. None. The code does not have any performance issues and does not require optimization.

5. Dashboard

1. Documentation Defects:
   a. Naming: The naming of software elements is generally good, with clear and concise names that accurately describe their purpose.
   b. Comment: The code comments are minimal, but they are accurate and helpful in understanding the purpose of each function and class.
2. Visual Representation Defects:
   a. Bracket Usage: The bracket usage is correct throughout the code.
   b. Indentation: The indentation is consistent and easy to read.
   c. Long Line: There are a few long code statements that could be broken up to improve readability.
3. Structure Defects:
   a. Dead Code: There is no dead code in the provided code.
   b. Duplication: There is no duplicate code in the provided code.
4. New Functionality:
   a. Use Standard Method: There are no single-purpose code statements that would benefit from a standardized approach.
5. Resource Defects:
   a. Variable Initialization: All variables are correctly initialized.
   b. Memory Management: The program's memory usage and management cannot be evaluated without more information.
6. Check Defects:
   a. Check User Input: User input is checked for validity, but the error message returned could be more informative.
7. Interface Defects:
   a. Parameter: There are no incorrect or missing parameters when calling functions or libraries.
8. Logic Defects:
   a. Compute: The logic appears to be correct.
   b. Performance: The efficiency of the algorithm used cannot be evaluated without more information.

Recommended changes:

- Improve code comments to provide more detail and context.
- Break up long code statements to improve readability.
- Improve error message returned when user input is invalid.
- Provide more information on memory usage and management to evaluate its effectiveness.

6. Mapping

1. Documentation Defects:
   a. Naming: The software element names are descriptive and follow PEP8 naming conventions.
   b. Comment: The code comments are sparse and could be improved to explain the purpose of each method and class.
2. Visual Representation Defects:
   a. Bracket Usage: No issues with bracket usage.
   b. Indentation: The code follows PEP8 indentation conventions.
   c. Long Line: Some lines exceed the recommended 79 character limit.

3. Structure Defects:
   a. Dead Code: There is some commented-out code that serves no purpose.
   b. Duplication: No duplicate code statements were found.
4. New Functionality:
   a. Use Standard Method: No single-purpose code statements were found that could benefit from a standardized approach.
5. Resource Defects:
   a. Variable Initialization: No uninitialized or incorrectly initialized variables were found.
   b. Memory Management: No issues with memory usage or management were found.
6. Check Defects:
   a. Check User Input: No analysis of user input or handling was found.
7. Interface Defects:
   a. Parameter: No missing or incorrect parameters when calling functions or libraries were found.
8. Logic Defects:
   a. Compute: No incorrect logic during system execution was found.
   b. Performance: No evaluation of the efficiency of the algorithm used was found.

Recommendations to improve the quality of the code:

- Add more descriptive comments to explain the purpose of each method and class.
- Remove commented-out code that serves no purpose.
- Break up long lines of code to improve readability.
- Consider evaluating the efficiency of the algorithm used to identify potential performance improvements.

7. Curate

1. Documentation Defects:
   a. Naming: The naming of software elements is mostly clear and follows Python naming conventions. However, the name of the `PostMapTaskCurateSchema` class could be improved to better reflect its purpose.
   b. Comment: The code comments are clear and concise, providing useful information about the purpose of the code.
2. Visual Representation Defects:
   a. Bracket Usage: There are no issues with incorrect or missing brackets.
   b. Indentation: The code is well-indented and easy to read.
   c. Long Line: There are no long code statements that hinder readability.
3. Structure Defects:
   a. Dead Code: There is some commented-out code that serves no meaningful purpose and should be removed.
   b. Duplication: There are no duplicate code statements that can be refactored.
4. New Functionality:
   a. Use Standard Method: There are no single-purpose code statements that could benefit from a standardized approach.
5. Resource Defects:
   a. Variable Initialization: All variables are correctly initialized.
   b. Memory Management: The code does not seem to have any memory management issues.
6. Check Defects:
   a. Check User Input: The code correctly validates user input and returns an error response if the input is invalid.
7. Interface Defects:
   a. Parameter: There are no incorrect or missing parameters when calling functions or libraries.
8. Logic Defects:
   a. Compute: The code logic seems correct and there are no issues during system execution.
   b. Performance: It is difficult to evaluate the efficiency of the algorithm used without more context about the overall system architecture.

Recommended changes:

- Improve the name of the `PostMapTaskCurateSchema` class to better reflect its purpose.
- Remove commented-out code that serves no meaningful purpose.
- Consider adding more detailed comments to explain the purpose of some of the code statements.
- Provide more context about the overall system architecture to better evaluate the performance of the algorithm used.

8. Visualisation

1. Documentation Defects:

   a. Naming: The names of the imported modules and classes are appropriate and descriptive. However, consider renaming the variable "user_id" to a more descriptive name, such as "request_user_id", to improve code clarity and understanding.
   b. Comment: The code has a single comment that explains the purpose of the "get" method. Consider adding more comments throughout the code to provide additional context and improve code readability.
2. Visual Representation Defects:

   a. Bracket Usage: The code has correct bracket usage, and no changes are required.
   b. Indentation: The code has correct indentation, and no changes are required.
   c. Long Line: The code does not have any long lines, which is commendable.
3. Structure Defects:

   a. Dead Code: The code does not contain any dead code, which is good.
   b. Duplication: The code does not have any duplicate code, which is well-maintained.
4. New Functionality:

   a. Use Standard Method: The code does not have any single-purpose code statements that require a standardized approach. No changes are needed in this regard.
5. Resource Defects:

    a. Variable Initialization: The code initializes all the variables correctly, and no changes are necessary.
    b. Memory Management: The code does not have any memory management issues, which is appropriate.
6. Check Defects:

    a. Check User Input: The code does not have any user input validation or handling. Consider implementing proper validation and error handling for user input to improve the robustness of the code.
7. Interface Defects:

    a. Parameter: The code does not have any incorrect or missing parameters when calling functions or libraries. No changes are required here.
8. Logic Defects:

    a. Compute: The code does not contain any incorrect logic during system execution, which is well-maintained.
    b. Performance: The code does not have any performance issues, which is commendable.

Recommended Changes:

- Rename the "user_id" variable to a more descriptive name, such as "request_user_id", to improve code clarity and maintainability.
- Add more comments throughout the code to explain the purpose and functionality of various sections and improve code understandability.
- Consider implementing user input validation and error handling mechanisms to ensure the code can handle invalid or unexpected user input gracefully.

Overall, the code appears to be well-structured and organized, with only minor recommendations for improvement.

9. Download

1. Documentation Defects:
   a. Naming: The naming of software elements is generally good, with clear and descriptive names used throughout the code.
   b. Comment: The code comments are generally good, with clear explanations of what the code is doing and why. However, there could be more comments explaining some of the more complex logic.
2. Visual Representation Defects:
   a. Bracket Usage: The bracket usage is correct throughout the code.
   b. Indentation: The indentation is consistent and easy to read.
   c. Long Line: There are no long lines of code that hinder readability.
3. Structure Defects:
   a. Dead Code: There is no dead code in the provided code.
   b. Duplication: There is no duplicate code in the provided code.
4. New Functionality:
   a. Use Standard Method: The code appears to use standard methods and approaches where appropriate.
5. Resource Defects:
   a. Variable Initialization: All variables are correctly initialized before use.
   b. Memory Management: The code does not appear to have any memory management issues.
6. Check Defects:
   a. Check User Input: The code checks for invalid input and returns an appropriate error message if necessary.
7. Interface Defects:
   a. Parameter: The code correctly handles parameters when calling functions and libraries.
8. Logic Defects:
   a. Compute: The logic in the code appears to be correct.
   b. Performance: The performance of the algorithm used is not evaluated in the provided code.

Recommended changes:

- Consider adding more comments to explain the more complex logic in the code.
- Evaluate the performance of the algorithm used to ensure it is efficient enough for the intended use case.

2023-05-31 Sprint 3 ChatGPT-assisted Code Review Report - FrontEnd

Code review information

Date	31 May 2023
Reviewer	Ricardo Luo
Author	Hanyi Gao Ricardo Luo Yue Fei
Code Repository / Branch	DI-Boxjelly/src/di-web/src/

Feedback

1. CodeSystem

1. Documentation Defects: a. Naming: The naming conventions used in this code are generally clear and descriptive. b. Comment: The code lacks comments, making it harder to understand the purpose of some variables or operations. It would be beneficial to add comments for clarity and maintainability.
2. Visual Representation Defects: a. Bracket Usage: Bracket usage seems to be correct throughout the code. b. Indentation: Indentation is consistently applied, improving code readability. c. Long Line: There are no overly long lines in the code.
3. Structure Defects: a. Dead Code: There's some commented out code regarding 'exportCodeSystem'. If this functionality is not needed, it should be removed. b. Duplication: No obvious code duplication detected.
4. New Functionality: a. Use Standard Method: The code seems to make good use of standard methods.
5. Resource Defects: a. Variable Initialization: Variables seem to be properly initialized. b. Memory Management: The code does not appear to have any memory leaks or improper memory management. React and the browser should handle memory management automatically.
6. Check Defects: a. Check User Input: User input seems to be validated in the form through 'rules' in Form.Item. It's not clear how file uploads are validated; the code might benefit from additional input validation on the uploaded file.
7. Interface Defects: a. Parameter: Function calls seem to include correct parameters.
8. Logic Defects: a. Compute: No obvious logic defects in the code. b. Performance: The performance of the code appears to be acceptable. However, the efficiency of the algorithm used depends on the actual functions in the 'api' file, which are not provided.

In conclusion, the code seems to be well-written overall with good use of React hooks and antd components. However, the code would benefit from the following improvements:

- Add comments throughout the code to improve readability and maintainability.
- Remove unused, commented-out code related to 'exportCodeSystem'.
- Add more extensive validation for file uploads.

2. Dashboard

1. Documentation Defects: a. Naming: The software element names are generally clear and describe their purpose. `inputRef` could be more explicit about its purpose, like `searchInputRef`. b. Comment: The code has no comments. Although the code is generally readable due to good variable naming, some complex parts like the `onSuccess` callback in `useRequest` might benefit from a brief explanation.
2. Visual Representation Defects: a. Bracket Usage: All brackets seem to be correctly used. There are no obvious syntax errors. b. Indentation: The code's indentation appears to be consistent and correct, improving readability. c. Long Line: There aren't any exceedingly long lines in the code that hinder readability.
3. Structure Defects: a. Dead Code: There doesn't appear to be any dead code. b. Duplication: No obvious duplication is present in the code.
4. New Functionality: a. Use Standard Method: The functions are generally following standard methods and using hooks effectively.
5. Resource Defects: a. Variable Initialization: All variables appear to be correctly initialized. b. Memory Management: Given that this is a JavaScript program running in a managed environment, it does not manually manage memory. No memory leaks are apparent from this code.
6. Check Defects: a. Check User Input: The function `onSingleTextSearch` correctly checks for an empty string before proceeding with the search.
7. Interface Defects: a. Parameter: All function calls and parameters appear to be correct. There don't seem to be any missing or incorrect parameters.
8. Logic Defects: a. Compute: The code's logic appears correct, though without knowing the exact specifications, this cannot be completely verified. b. Performance: The performance of this component should be sufficient for most use cases. However, a potential performance bottleneck could be the number of requests being made in parallel using `useRequest`. If the component needs to handle a large amount of data, this could cause problems.

Overall, the code is well written and adheres to many best practices. However, it could be improved by adding comments to explain complex or critical parts, renaming `inputRef` for more clarity, and considering potential performance issues if a large amount of data is being handled.

3. Login

1. Documentation Defects: a. Naming: Most function and variable names like `login`, `navigate`, `setLoggedIn` are clear and self-descriptive, which is good practice. b. Comment: The code has no comments, which makes it harder to understand its intent. Even though the code is fairly clear, some sections (such as storing data in localStorage and calling the `login` function) could benefit from explanations.
2. Visual Representation Defects: a. Bracket Usage: Brackets are used correctly and in line with JavaScript/React conventions. b. Indentation: The indentation of the code is consistent and helps readability. c. Long Line: There are no excessively long lines in the code.
3. Structure Defects: a. Dead Code: There is no dead code in the provided snippet. b. Duplication: There are no significant duplicate code statements that need to be refactored.
4. New Functionality: a. Use Standard Method: The code appears to use standard methods appropriately.
5. Resource Defects: a. Variable Initialization: All variables are correctly initialized. b. Memory Management: In JavaScript, memory management is handled by the JavaScript engine and doesn't require explicit handling in code.
6. Check Defects: a. Check User Input: User input for email and password is being accepted and seems to be handled correctly. However, the email input could benefit from additional validation (e.g., checking for proper email format).
7. Interface Defects: a. Parameter: The function calls with parameters appear to be correct.
8. Logic Defects: a. Compute: No computational errors are apparent in the code. b. Performance: No obvious performance issues in the given code. However, in real-world applications, the performance of the `login` function will be dependent on factors such as network latency and server performance, which aren't under control of this component.

In summary, the code appears to be well written, but could benefit from comments for better readability and maintainability, as well as additional validation checks for the email input field. Also, consider the implications of storing sensitive user information in localStorage, as it may pose security risks. If the data is sensitive, consider storing the data in HTTP-only cookies or using a more secure strategy for managing authentication.

4. Main

1. Documentation Defects: a. Naming: The names of variables, functions, and components are meaningful and descriptive, which is good practice. b. Comment: The code lacks comments, which makes it hard to understand the role of certain functions and components without a careful study. Adding comments to the code would significantly increase its readability and maintainability.
2. Visual Representation Defects: a. Bracket Usage: The bracket usage in the code seems correct, no issues were found. b. Indentation: Indentation is consistently applied, making the code blocks clear and easy to read. c. Long Line: There are a few long lines of code in the script. Breaking these down would improve readability.
3. Structure Defects: a. Dead Code: There is a commented line, `// const user = JSON.parse(localStorage.getItem ('userDetail'));` which appears to be unused and can be considered dead code. b. Duplication: There's no significant duplication in the code.
4. New Functionality: The code uses standard and modern React practices, such as functional components and hooks.
5. Resource Defects: a. Variable Initialization: All variables appear to be correctly initialized. b. Memory Management: The code doesn't seem to have any apparent issues with memory management.
6. Check Defects: a. Check User Input: User input is validated in the form fields.
7. Interface Defects: a. Parameter: The code does not seem to have incorrect or missing parameters when calling functions or libraries.
8. Logic Defects: a. Compute: The logic seems correct as far as can be determined from the code snippet. b. Performance: There is no explicit optimization done in the code, but also there are no obvious inefficiencies. Using the `React.memo` or `useMemo` hooks might improve performance in some cases, but it would be premature optimization without profiling.

In summary, the code is overall well-written and adheres to modern React standards. The main areas for improvement would be adding comments and perhaps breaking down some of the longer lines of code for readability. Also, the dead code should be removed if it is not needed for future reference. It would also be helpful to handle possible exceptions, especially for API calls to ensure the application doesn't crash unexpectedly when an error occurs.

5. Mapping

1. Documentation Defects: a. Naming: The naming convention is good and reflects the purpose of each function and variable. The function and variable names are clear and descriptive. b. Comment: The code has no comments at all. It would improve readability and maintainability if comments explaining the purpose and functioning of key blocks of code were included.
2. Visual Representation Defects: a. Bracket Usage: Brackets are used correctly throughout the code. b. Indentation: The indentation is consistent and enhances the readability of the code. c. Long Line: No excessively long lines of code are present.
3. Structure Defects: a. Dead Code: No dead code detected. b. Duplication: No duplicated code found.
4. New Functionality: a. Use Standard Method: The code appears to use standard methods appropriately.
5. Resource Defects: a. Variable Initialization: All variables are correctly initialized. b. Memory Management: As this is a React functional component, the memory management is handled by the React framework itself.
6. Check Defects: a. Check User Input: The code checks for an empty string in the `onSingleTextSearch` function. This can be considered appropriate validation for the given scenario.
7. Interface Defects: a. Parameter: No incorrect or missing parameters detected when calling functions or libraries.
8. Logic Defects: a. Compute: No incorrect logic found during the analysis. b. Performance: The efficiency of the algorithm cannot be fully assessed from this code. It largely depends on the implementations of the functions `mapSingleText` and `createMappingTask` which are imported from '. /api'.

To summarize, the overall quality of the code is good with appropriate use of React hooks and other conventions. However, there's a significant scope of adding comments to the code to make it more readable and maintainable. Comments can especially help explain the purpose of the functions, the useState and useRequest hooks, the useRef hook, and the logic inside the functions.

6. MappingHistory

1. Documentation Defects: a. Naming: The quality of software element names is good and generally self-descriptive. However, `team_id` and `board _id` might be better written in camelCase to follow JavaScript naming conventions. b. Comment: The code lacks comments. Key blocks of code and functions should be accompanied by comments to enhance understanding and maintainability.
2. Visual Representation Defects: a. Bracket Usage: The code appears to have all brackets correctly placed. b. Indentation: The indentation in this code is consistent and enhances readability. c. Long Line: No excessively long lines of code are found.
3. Structure Defects: a. Dead Code: No dead code is detected in the snippet. b. Duplication: There's no noticeable code duplication.

4. New Functionality: a. Use Standard Method: The code uses standard methods as per React and Ant Design norms.
5. Resource Defects: a. Variable Initialization: All variables are correctly initialized. b. Memory Management: As this is a React functional component, the memory management is handled by the React framework itself.
6. Check Defects: a. Check User Input: The user input checks are not fully visible in this component, likely they are implemented within the API calls.
7. Interface Defects: a. Parameter: No issues found in this area. Parameters are correctly passed to functions and libraries.
8. Logic Defects: a. Compute: No logical errors were identified. b. Performance: The efficiency of the algorithm cannot be fully assessed from this code. It largely depends on the implementations of the functions imported from './api'.

In conclusion, the code is written well, adheres to React norms, and uses hooks effectively. However, it lacks comments which makes it a bit hard to understand for someone new to the code. Adding descriptive comments would significantly improve its readability and maintainability. Variable names should also follow a consistent naming convention, preferably camelCase which is the standard in JavaScript.

7. MappingResult

1. Documentation Defects: a. Naming: The variable names are descriptive and easy to understand. However, like the previous example, `taskId`, `teamId`, and `boardId` might be better written in camelCase. b. Comment: There are no comments in the code. Adding comments would improve readability and maintainability, especially for other developers who might work on this code in the future.
2. Visual Representation Defects: a. Bracket Usage: No issues found with brackets. They're correctly used. b. Indentation: Indentation is consistent and enhances code readability. c. Long Line: There are no excessively long lines that hinder readability.
3. Structure Defects: a. Dead Code: No dead code is identified. b. Duplication: There's no noticeable code duplication.
4. New Functionality: a. Use Standard Method: Standard methods are being used according to React and JavaScript norms.
5. Resource Defects: a. Variable Initialization: Variables are initialized correctly. b. Memory Management: Memory management is handled by the React framework.
6. Check Defects: a. Check User Input: The user input checks are not fully visible in this component. It seems to be done in the API calls.
7. Interface Defects: a. Parameter: No issues were found with parameters. They're correctly passed to functions and libraries.
8. Logic Defects: a. Compute: No logic errors were identified in the code. b. Performance: The efficiency of the code can't be fully assessed from this snippet, as it depends on the implementations of the functions imported from '../Mapping/api'.

In summary, the code is well-written and follows the React hooks style. However, it lacks comments which makes it harder for others to understand what each part of the code is doing. I recommend adding comments to improve code readability. Additionally, consistent naming conventions (preferably camelCase) should be followed throughout the code.

8. Profile

1. Documentation Defects: a. Naming: The names for functions and variables are descriptive and properly illustrate their purpose, making the code self-documenting to an extent. b. Comment: No comments are provided in the code. While the function and variable names are quite descriptive, some lines or sections of code may still benefit from comments for easier understanding by other developers.
2. Visual Representation Defects: a. Bracket Usage: Brackets are correctly used throughout the code. b. Indentation: The indentation in this code is consistent and greatly improves readability. c. Long Line: No lines in this code appear to be excessively long.
3. Structure Defects: a. Dead Code: No dead code is apparent. b. Duplication: There doesn't appear to be any duplicated code.
4. New Functionality: a. Use Standard Method: The code uses standardized methods for its functions.
5. Resource Defects: a. Variable Initialization: All variables appear to be initialized correctly. b. Memory Management: As this is a React application, memory management is primarily handled by the framework itself.
6. Check Defects: a. Check User Input: It's not completely clear if there's validation happening for user input before it is processed or sent to the server, which is a potential issue.
7. Interface Defects: a. Parameter: All functions and methods seem to have the correct parameters when called.
8. Logic Defects: a. Compute: There doesn't seem to be any incorrect logic during the execution of the code. b. Performance: The code's performance seems adequate, but it's also dependent on how the API handles the requests.

To summarize, this code is generally well-written with clear variable and function names. However, there's a noticeable lack of comments, which would help others understand the code more quickly. Additionally, while there's no clear user input validation, the true status of input checking would depend on the API's implementation and it's worth ensuring that there is robust checking and validation there.

9. Register

1. Documentation Defects: a. Naming: The naming conventions are clear and descriptive. Variables, functions, and components all have names that accurately represent their functionality. b. Comment: The code lacks comments. Although the code is quite readable and follows standard conventions, including comments would provide better context and understanding for future developers.
2. Visual Representation Defects: a. Bracket Usage: The bracket usage is correct. All brackets are properly opened and closed. b. Indentation: The indentation is consistent and follows standard conventions, which enhances readability. c. Long Line: No extremely long lines were observed that could impair readability.
3. Structure Defects: a. Dead Code: No dead code was found in the script. b. Duplication: No significant code duplication was found.
4. New Functionality: a. Use Standard Method: Standard methods and libraries are being used correctly. The form validation is done by Ant Design's Form component which handles most of the low-level details.
5. Resource Defects: a. Variable Initialization: All variables are properly initialized. b. Memory Management: As it's a front-end JavaScript code and doesn't handle explicit memory allocation, it doesn't have any apparent memory management issues.
6. Check Defects: a. Check User Input: User input is properly validated using the rules prop provided by Ant Design's Form.Item component.
7. Interface Defects: a. Parameter: There don't appear to be any missing or incorrect parameters when calling functions or libraries. All necessary data seems to be correctly passed between functions and components.
8. Logic Defects: a. Compute: There do not seem to be any logical errors in the code. b. Performance: The script doesn't have any operations that would cause significant performance issues. It's a straightforward form submission.

In conclusion, the code is well-written and doesn't require any major changes. To improve the quality of the code, consider adding comments to explain the purpose and functionality of the components and functions. This will enhance readability and maintainability, especially for developers who are unfamiliar with the code.

10. TeamProfile

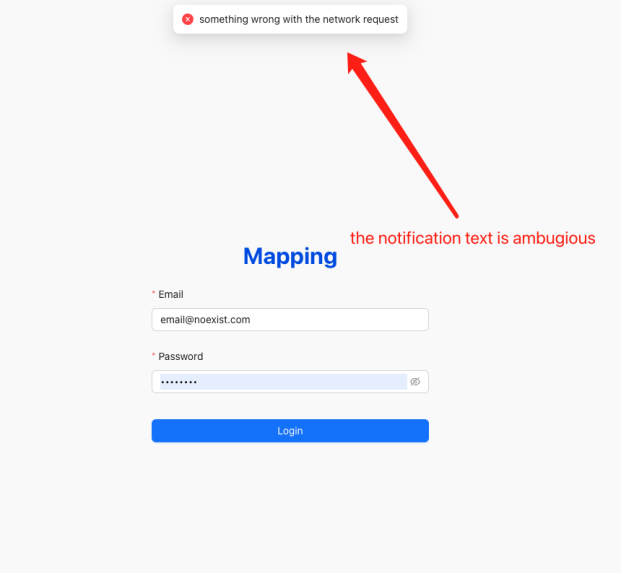
1. Documentation Defects: a. Naming: The variable and function names are mostly clear and meaningful. However, `msgApi` and `api` could be more descriptive. b. Comment: There are no comments in the code. Adding comments to explain more complex parts of the code or overall functionality would improve the maintainability of the code.
2. Visual Representation Defects: a. Bracket Usage: The code properly uses brackets, both for function declaration and JSX syntax. b. Indentation: The code's indentation seems to follow standard conventions and is consistent throughout, making it readable. c. Long Line: The line `const link = DOMAIN_URL + '/register?invite_token=' + result.data;` could be considered slightly long. It could be rewritten using a template literal for better readability.
3. Structure Defects: a. Dead Code: There doesn't appear to be any dead code in the provided script. b. Duplication: The code doesn't seem to contain duplicate blocks.
4. New Functionality: a. Use Standard Method: The code seems to be using standard practices for React and Ant Design libraries.
5. Resource Defects: a. Variable Initialization: All variables seem to be initialized correctly, considering they are either imported modules or state variables managed by React. b. Memory Management: There is no explicit memory management in this code as it's handled by the JavaScript engine and the React library.
6. Check Defects: a. Check User Input: This component doesn't handle direct user input, except for button clicks. The handling of user input from local storage or APIs should be more robust to prevent errors if the expected data is not present.
7. Interface Defects: a. Parameter: The function doesn't take any parameters. However, the code is making use of functions from other modules with parameters and it seems to pass the correct arguments.
8. Logic Defects: a. Compute: The logic seems to be correct in this function. b. Performance: There is nothing inherently inefficient with the algorithm used in this component. However, re-rendering could potentially be optimized based on how often state updates occur.

In conclusion, this code seems to be mostly well-written and follows best practices for a React component. However, to improve it, consider adding comments for better understandability and maintainability, and consider handling exceptions if the data retrieved from the local storage or APIs is not as expected. Also, as a small improvement, the `api` variable could be given a more descriptive name.

Issue Tracking

Title	Creator	Modified
ISSUE00029: Cannot display the board with deleted member's task	Chenyang Dong	01 Jun, 2023
ISSUE00010: Login incorrect error message	KUNXI SUN	01 Jun, 2023
ISSUE00012: MedCAT: Fail to load the model because of ValidationError	Yue Fei	01 Jun, 2023
ISSUE00015: Github Actions: checkout repository takes too long and sometimes fail #15	KUNXI SUN	01 Jun, 2023
ISSUE00021: MedCAT: Cannot use cat.multi_processing to map a file of raw texts	Yue Fei	01 Jun, 2023
ISSUE00022: Dashboard nickname is not updated in real time	Ricardo Luo	01 Jun, 2023
ISSUE00023: Forced to refresh page to login in successfully after cookie removed	KUNXI SUN	01 Jun, 2023
ISSUE00025: Avatar is obscured when entering edit mode	Ricardo Luo	01 Jun, 2023
ISSUE00026: The numbers in the chart are inconsistent with the Numbers on the card	Ricardo Luo	01 Jun, 2023
ISSUE00028: New member register showing missing cookie	KUNXI SUN	01 Jun, 2023
ISSUE00027: New users cannot be invited if this user does not have an account in login status.	Ricardo Luo	01 Jun, 2023

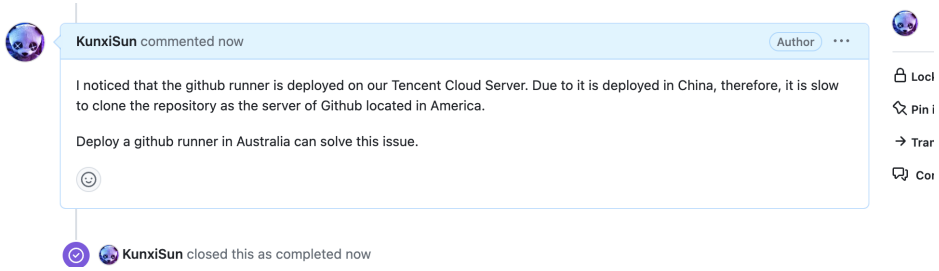
ISSUE00010: Login incorrect error message

ISSUE00010	☑ Login incorrect error message
Status	SOLVED
Assignee	KUNXI SUN
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/10
Description	Fail to give reasonable notification when the email or password is incorrect. 
Solution	Solved in eaf5d01 commit on branch feature/uil

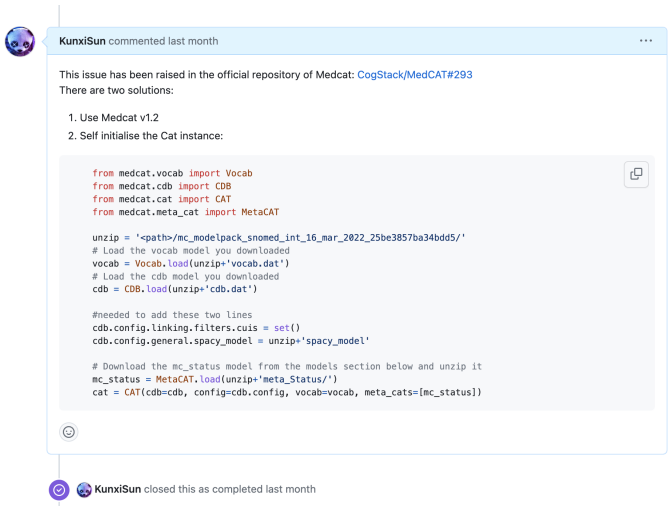
ISSUE00012: MedCAT: Fail to load the model because of ValidationError

ISSUE00012	☑ MedCAT: Fail to load the model because of ValidationError
Status	SOLVED
Assignee	Yue Fei
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/12
Description	When use the MedCAT demo code to load the model ValidationError raised. MedCAT demo code: ```python from medcat.cat import CAT # Download the model_pack from the models section in the github repo. cat = CAT.load_model_pack('<path to downloaded zip file>') # Test it text = "My simple document with kidney failure" entities = cat.get_entities(text) print(entities) # To run unsupervised training over documents data_iterator = <your iterator> cat.train(data_iterator) #Once done, save the whole model_pack cat.create_model_pack(<save path>) ``` ValidationError message: ``` File "pydantic/main.py", line 384, in pydantic.main.BaseModel.__setattr__ pydantic.error_wrappers.ValidationError: 1 validation error for Config linking -> filters -> cuis value is not a valid set (type=type_error.set) ```
Solution	Solution: This issue has been raised in the official repository of Medcat: CogStack/MedCAT#293 There are two solutions: 1. Use Medcat v1.2 2. Self initialise the Cat instance: `from medcat.vocab import Vocab` `from medcat.cdb import CDB` `from medcat.cat import CAT` `from medcat.meta_cat import MetaCAT` `unzip = '<path>/mc_modelpack_snomed_int_16_mar_2022_25be3857ba34bdd5/'` `# Load the vocab model you downloaded` `vocab = Vocab.load(unzip+'vocab.dat')` `# Load the cdb model you downloaded` `cdb = CDB.load(unzip+'cdb.dat')` `#needed to add these two lines` `cdb.config.linking.filters.cuis = set()` `cdb.config.general.spacy_model = unzip+'spacy_model'` `# Download the mc_status model from the models section below and unzip it` `mc_status = MetaCAT.load(unzip+'meta_Status/')` `cat = CAT(cdb=cdb, config=cdb.config, vocab=vocab, meta_cats=[mc_status])`

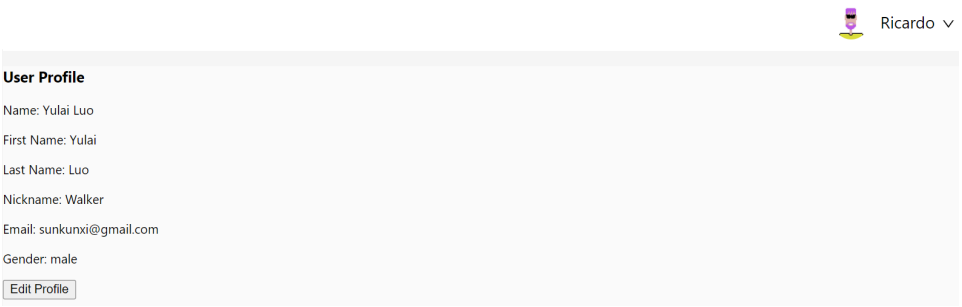
ISSUE00015: Github Actions: checkout repository takes too long and sometimes fail #15

ISSUE00015	☑ Github Actions: checkout repository takes too long and sometimes fail
Status	SOLVED
Assignee	KUNXI SUN
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/15
Description	actions/checkout@v2 can successfully checkout the repository on the runner, but sometimes it will fail and it takes too long to checkout. Not sure what is going on. `Error: fatal: unable to access 'https://github.com/COMP90082-2023-SM1/DI-Boxjelly/': GnuTLS recv error (-110): The TLS connection was non-properly terminated.` `The process '/usr/bin/git' failed with exit code 128` `Waiting 11 seconds before trying again`
Solution	

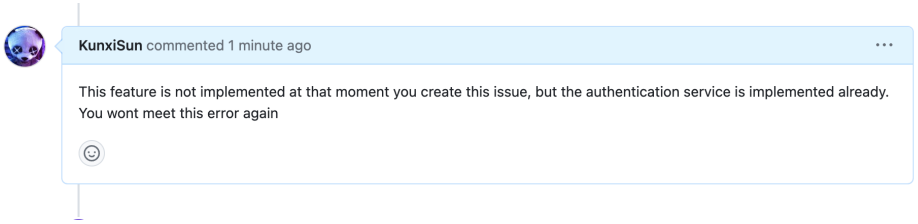
ISSUE00021: MedCAT: Cannot use cat.multi_processing to map a file of raw texts

ISSUE00021	☑ MedCAT: Cannot use cat.multi_processing to map a file of raw texts
Status	SOLVED
Assignee	Yue Fei
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/21
Description	When use cat.multi_processing to map a file of raw texts ``` --- AttributeError Traceback (most recent call last) /var/folders/7h/cjlhkqns6ql6trqrp2lr1thc0000gn/T/ipykernel_37206/989250044.py in <module> ----> 1 results = cat.multi_processing(texts, nproc=2, only_cui=True) AttributeError: 'CAT' object has no attribute 'multi_processing' ``` raise. Solution: Update python and use multiprocessing instead.
Solution	

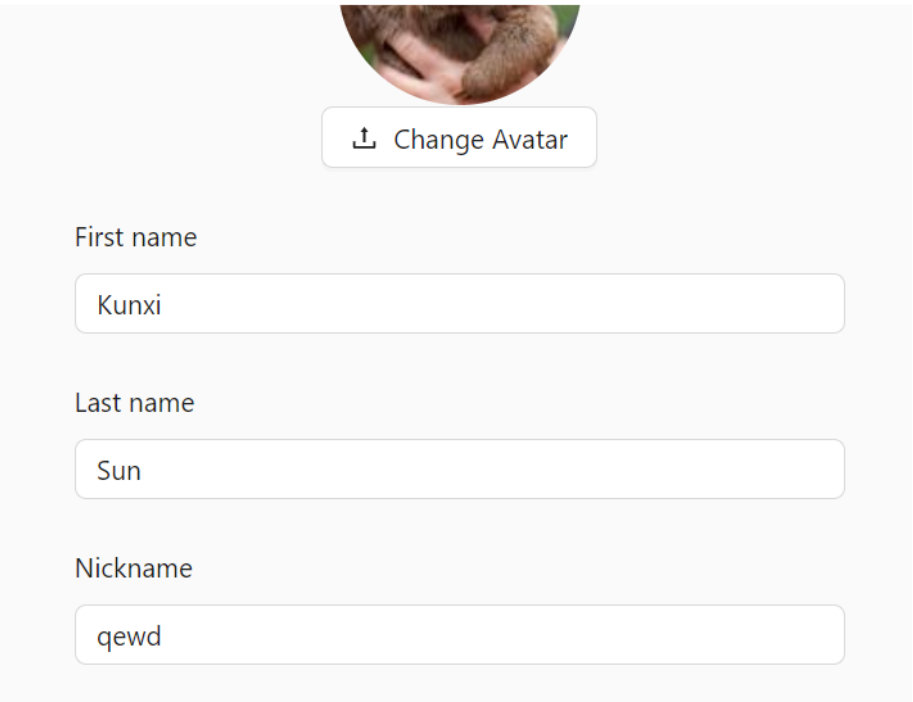
ISSUE00022: Dashboard nickname is not updated in real time

ISSUE00022	☑ Dashboard nickname is not updated in real time
Status	SOLVED
Assignee	Hanyi Gao
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/22
Description	After I edit the nickname on the profile page, the dashboard nickname does not update with the profile nickname. It does not update until I log in again. 
Solution	Solved in 9162ca9 commit on branch feature/uil

ISSUE00023: Forced to refresh page to login in successfully after cookie removed

ISSUE00023	☑ Forced to refresh page to login in successfully after cookie removed
Status	SOLVED
Assignee	Hanyi Gao
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/23
Description	Fail to give reasonable notification when the email or password is incorrect. If I clear the cookie when I'm on dashboard page or any page inside the system, after I refresh, I will be identified as not logged in, and redirects me to login page which is correct. After that, I can use email and password to log in, which shows that I login successfully and I can see cookies are set successfully but it does not redirect me to dashboard page. I have to refresh on the login page and then click on login to succeed logging in with redirecting.
Solution	

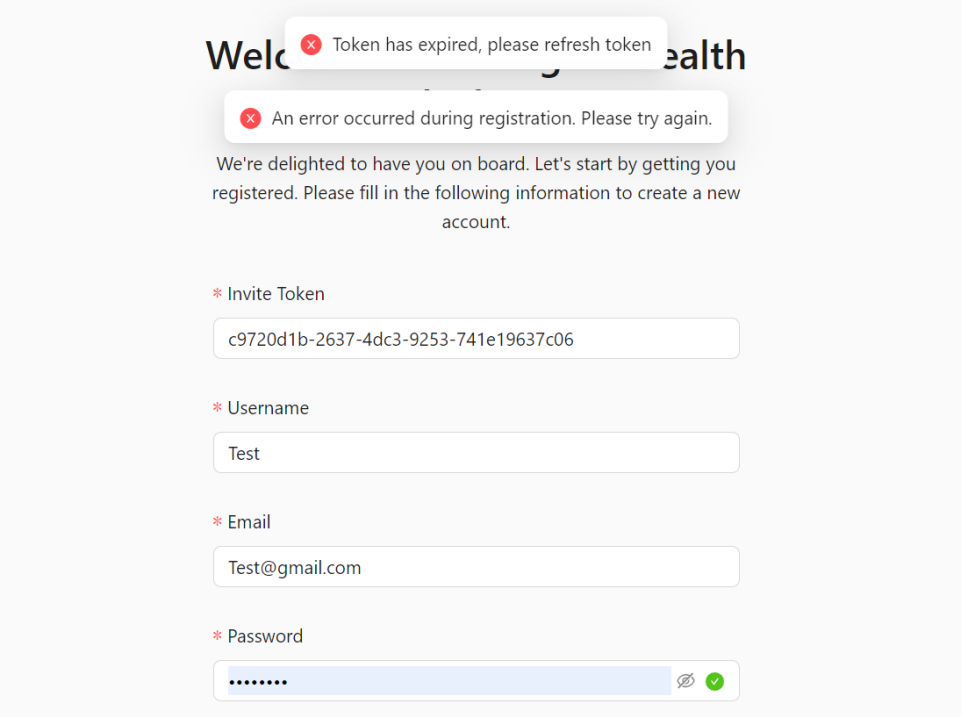
ISSUE00025: Avatar is obscured when entering edit mode

ISSUE000025	☑ Avatar is obscured when entering edit mode
Status	SOLVED
Assignee	Ricardo Luo
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/25
Description	When entering edit mode, the avatar on the profile page is obscured 
Solution	Solved in a640d14 commit on branch feature/uil

ISSUE00026: The numbers in the chart are inconsistent with the Numbers on the card

ISSUE000026	☑ The numbers in the chart are inconsistent with the Numbers on the card
Status	SOLVED
Assignee	Ricardo Luo
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/26
Description	In the visualization component, the numbers in the chart are inconsistent with the numbers on the card. 
Solution	Solved in b073037 commit on branch feature/uil

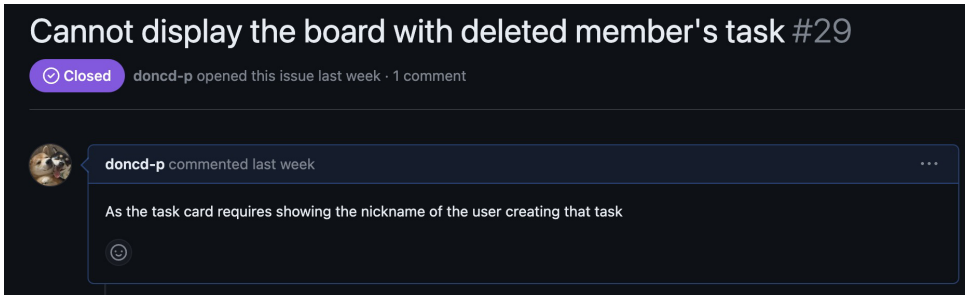
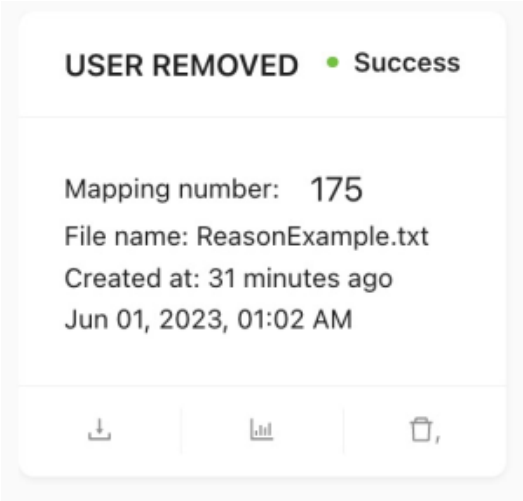
ISSUE00027: New users cannot be invited if this user does not have an account in login status.

ISSUE00027	☑ New users cannot be invited if this user does not have an account in login status.
Status	SOLVED
Assignee	Ricardo Luo
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/27
Description	We use "const headers = getCSRFTokenHeader();" in api.js for the Register page. But the new user does not have CSRFToken. If the new user does not have an account in login status, the user could not register. This is contradictory and unreasonable. 
Solution	Solved in 2d2dd64 commit on branch feature/uil

ISSUE00028: New member register showing missing cookie

ISSUE0028	☑ New member register showing missing cookie
Status	SOLVED
Assignee	KUNXI SUN Hanyi Gao
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/28
Description	
Solution	Refresh token is canceled, you will get error when the token is expired. When the access token is expired, you should re-login to get a new access token

ISSUE00029: Cannot display the board with deleted member's task

ISSUE0029	**Cannot display the board with deleted member's task**
Status	SOLVED
Assignee	Chenyang Dong
Github Issue link	https://github.com/COMP90082-2023-SM1/DI-Boxjelly/issues/29
Description	
Solution	Storing the nickname in the dictionary and use get method of dictionary to prevent possible KeyError in case there are tasks with removed users. It will default to "USER REMOVED" if not found. 

CI/CD

Version	Description	Date
Version 1.0.0	1. Basic CI/CD using Ansible, a short description about how team use Ansible to deploy product	28 Apr 2023

Version 1.0.0

1 Introduction

1.1 Ansible & Github Actions

A Continuous Integration/Continuous Deployment (CI/CD) pipeline is crucial for modern software development, as it helps to automate the process of building, testing, and deploying applications. While we were planing to develop our project, we plan to use Ansible for deploying the product on our development cloud server. The Ansible script has been successfully implemented and tested. In later Sprint, team will try to use Github Actions for automating deployment by merging branches.

- A short description about Ansible can be found in confluence page.
- More details about how to deploy our product on development server can be found on Github Ansible README.

2 Continuous Integration using Github Actions

This is planned in Sprint 3

3 Continuous Deployment for Development environment

Ansible is used to automate the deployment of the application to the development environment. It provisions the cloud server, configures the environment, installs dependencies, and deploys the application.

4 Continuous Deployment for Production environment

As team still did not get the instance from client, therefore, we cannot deploy project on the production environment

5 Monitoring and Feedback

Throughout the CI/CD process, the team monitors the build and deployment status using GitHub Actions logs and notifications. Any failures or issues encountered during the process are addressed and resolved to ensure a smooth and efficient pipeline.

User Experience Evaluation

In this project, our team decide to conduct an expert review instead of usability test due to the time and resource limit. Heuristic Evaluation and Cognitive Walkthrough will be conducted to identify general usability issues.

Website:	Digital health mapping platform
Technology:	Macboook Pro M1
Evaluator:	Yue Fei
Date of Evaluation:	16 May 2023

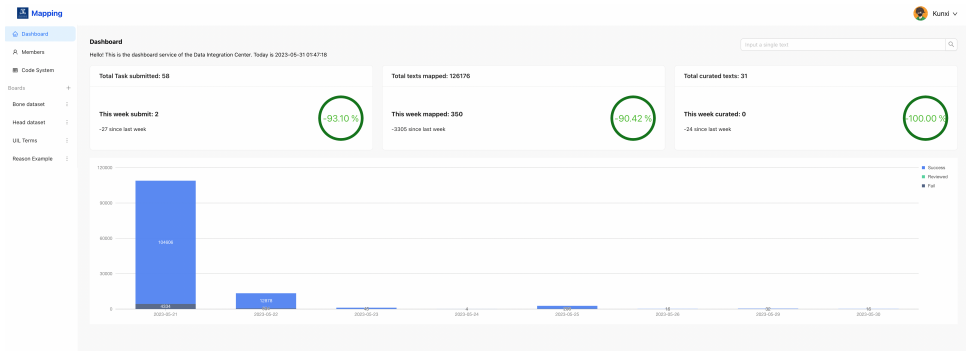
Heuristic Evaluation

Introduction:

Heuristic evaluation is a process where experts use rules of thumb to measure the usability of user interfaces in independent walkthroughs and report issues. In this report, I will use the Nielsen heuristic evaluation 10 principles to evaluate the website.

1. Visibility of system status;
2. Match between system and the real world;
3. User control and freedom;
4. Consistency and standards;
5. Error prevention;
6. Recognition rather than recall;
7. Flexibility and efficiency of use;
8. Aesthetic and minimalist design;
9. Help users recognize, diagnose, and recover from errors;
10. Help and documentation.

Positive Findings:

Heuristic **(Nielsen, 2000)**	**Comments**
Aesthetic and minimalist design	Overall speaking, the heath mapping platform has a very concise UI design, presenting information and interfaces in a visually appealing and uncluttered manner. This website has clear and concise interface, the main function of the website is clearly displayed on the left sidebar with a very clear structure. 
Help users recognize, diagnose, and recover from errors	When errors happens. This website some pop up message to help the user recover from unwanted state.
Match between system and the real world	This website has standard and clear design. Use the word, phrases and concept familiar to users. Users can leverage their existing knowledge and transfer it seamlessly to the system, reducing the need for extensive learning or cognitive effort.

Flexibility and efficiency of use	This website provide shortcut for expert user and hidden from novice user. The design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. For example, user can use download and visualise button on the task card to download and visualise data instead of click into the task download. And for novice user, they can download the mapping file inside the task page.

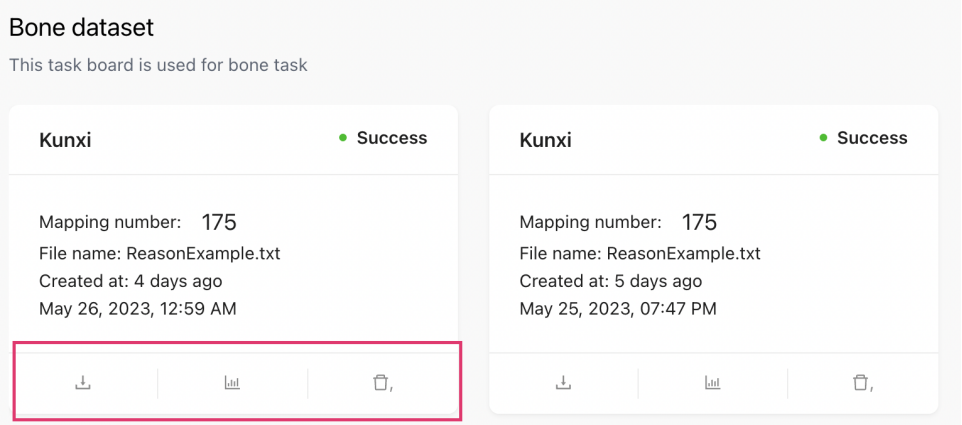


Figure1: Shortcut for expert user

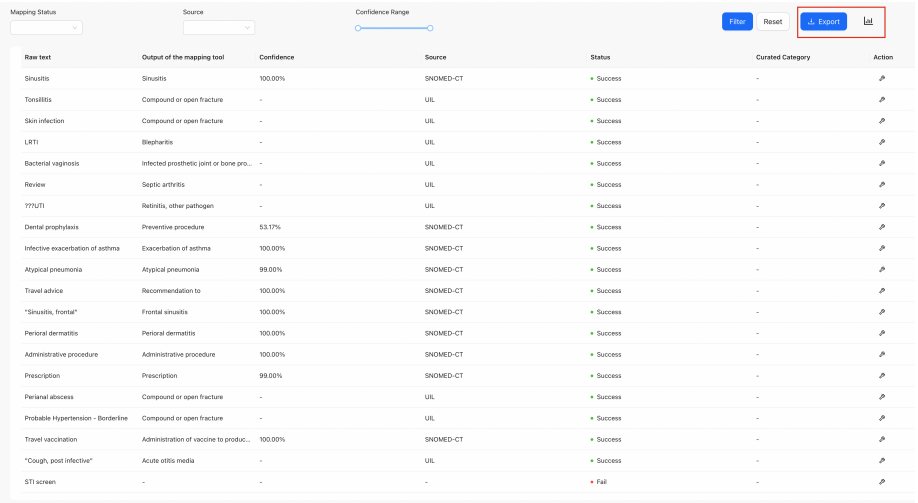
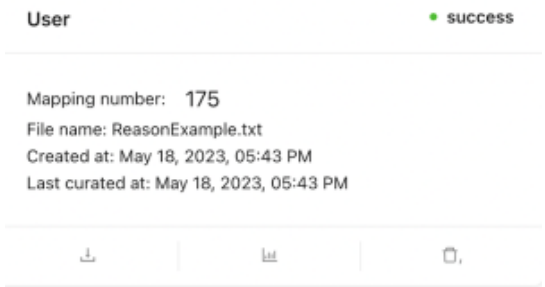
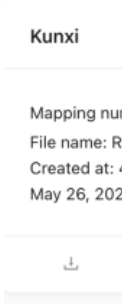


Figure2: Download and Visualisation for Novice User

Issues:

Heuristic (Nielsen, 2000)	Comments	Scope	Frequency	Impact	Severity	Recommendation	Status

Visibility of system status	There is no indication of the current location of the website	Medium Some users may be confused about their current state.	High	Low The website is straight forward	Low	Add the current path or highlight the current page on the side bar.	Already fixed. In 
User control and freedom	Overall is good, but can add option for user to select the file type.	Low Some users may want to download the file in a different format.	Low The download function is frequently used.	Low CSV file can meet the majority of needs.	Low	Add the file type option for user to select.	Discard
Consistency and standards	Overall is good, but the card information is not follow the standard website design. Usually the the create time is represented as "XXdays ago". 	Low Some users may calculate the time elapsed from a specific "create" time to the current time	Medium The task card is frequently used.	Low It do not lead to any confusion.	Low	Change the time format.	Already fixed. Cr 
Error prevention	Overall, it is good, but there is no confirmation dialog if the user clicks the log out button accidentally.	High All users need to use this function.	High It is a frequently used function for all users.	Low	Low	Add a confirmation dialog window to confirm log out.	Discard
Help and documentation	There is no help document.	Medium Some users may need documentation or guidance to use the website.	Medium Without any help documentation or contact information, users may find it difficult to get assistance.	Low The website is straight forward and have aesthetic and minimalist design.	Low	Add help document.	Will send to clien

Result:

This website has good usability based on the Nielsen 10 usability heuristics for user interface design. It demonstrates a match between the system and the real world, consistency and standards, good memorability, flexibility and efficiency of use, and an aesthetic and minimalist design. Despite some identified issues, the overall usability is satisfactory.

Cognitive Walkthrough

Cognitive walkthrough is a task-based usability-inspection method that involves a cross-functional team of reviewers walking through each step of a task flow and answering a set of prescribed questions, with the goal of identifying those aspects of the interface that could be challenging to new users. In this report, I will go through major tasks to test the functionalities.

Task1:

Mapping a single text to SNOMED-CT or UIL view the result.

	steps	Will users know what to do?	Will the users know how to do it?	Will users understand from feedback whether the actions correct or not?
Step1	Enter the free text that needs to be mapped.	Yes, usually there is a search bar to search.	Yes, it is shown on the dashboard	Yes, it will show the search bar.
Step2	Click the search button to map the text.	Yes, usually users need to click the search icon to search.	Yes, it is just next to the search box.	Yes, it will show the result.
Step3	View the mapping results.	Yes, usually it will automatically appear somewhere on the screen	Yes, it will show on the screen.	Yes, there is a pop-up window showing the result.

Task2:

Mapping a file to SNOMED-CT or UIL and viewing the result.

	steps	Will users know what to do?	Will the users know how to do it?	Will users understand from feedback whether the actions correct or not?
Step1	Open the board.	No, because mapping a single task is on the dashboard, users may assume that mapping a file is in the same location. However, it is actually located on the board.	No, there is no indication of its location, so users may take a while to find it.	Yes, once they click on the board, they will be able to find the location to create a task and upload the file.
Step2	Create a task.	Yes, there is usually a location to upload a file to create a new task. However, the naming of the button may be confusing for new users.	Yes, there is usually a button on the screen to create a task.	Yes, there is a pop-up window that requires the user to upload a file for mapping, with an indication text.
Step3	Upload the file for mapping	Yes, there is an indication text on the popup window asking the user to drag or select a file to upload.	Yes, usually users need to drag or select a file to upload.	Yes after the file is select, it will show on the screen.
Step4	View the mapping result.	Yes, the mapping task is clearly shown on the screen.	Yes, usually user can click on the mapping task to view the result.	Yes, it will redirect the user to the mapping result.

Recommendation: Add help documents to help the user find the right place.

Task3:

Curate the mapping result.

	steps	Will users know what to do?	Will the users know how to do it?	Will users understand from feedback whether the actions correct or not?
Step1	Click on the action button.	Yes, the mapping tool always has the curation function.	Yes, the action button is clearly shown on the mapped task. But new users may not know to click the button to curate the mapping.	Yes, the curated category will show a window for users to select the term that they want to curate the mapping.
Step2	Click on the selection box to curate the category.	Yes, usually users need to select the category.	Yes, there is a pop-up box under the curated category column for users to select the category they want to curate.	Yes, the categories that users want to curate will be displayed on the screen after being clicked.
Step3	Save the curated term to the task.	Yes, usually users need to confirm their curation and feedback to the system.	Yes, the save button is clearly shown on the right side of the screen.	Yes, the status will change after being reviewed after curation.

Task4:

Visualisation of the mapping task.

	steps	Will users know what to do?	Will the users know how to do it?	Will users understand from feedback whether the actions correct or not?

Step1	Click on the visualisation button on the task page.	Yes, usually visualisation may show directly on the page or have a place to enter.	Yes, the visualisation button is clearly shown on the top-right hand side of the task.	Yes, the visualisation drawer will show.

Result:

Overall speaking, I walked through the major task that users will use the website for. Most of them are easy and clear to use. But fresh users of the website may need time to find the right place to complete the task. But the memorability of the website is excellent, after they learn the right way to finish their task, they can use it efficiently.

Pair programming

What is pair programming?

> *Note: the following definition is generated by ChatGPT 4.0*

Pair programming is a software development technique where two programmers work together at one workstation. The technique involves two roles: the "driver" and the "navigator" (also known as the "observer").

1. **Driver:** The driver is the one who writes the code. This person focuses on the tactical aspects of completing the task, like syntax, coding conventions, and the specific implementation details.
2. **Navigator:** The navigator observes the work of the driver, focusing more on the strategic aspects of the task. They think about the design, the overall direction of the work, and what will come next. They also review the code written by the driver and can suggest changes or improvements.

The programmers switch roles frequently (maybe every few minutes or hours) to get the benefits of both aspects. This approach is believed to improve code quality by catching mistakes early (two sets of eyes instead of one), improve collaboration and communication, facilitate knowledge sharing and skill improvement, and generally lead to better design decisions.

However, it's also important to note that pair programming requires a good interpersonal rapport and can be exhausting, so it's usually not done 100% of the time. Some tasks may also not be well-suited to pair programming. The best usage of pair programming often depends on the team, the individuals involved, and the task at hand.

How should we do pair programming?

> *The following shows our pair programming plan to implement this project*

1. Front-end and Back-end Interface Debugging

- Start by clearly defining the interface between the front-end and back-end. This includes the structure of the request, response, error messages, and status codes.
- The driver writes the code for implementing or modifying the interface. The navigator reviews and provides suggestions for improvements, looks up relevant documentation, and thinks about edge cases.
- Switch roles and debug the interface. Test it by sending requests from the front-end and verifying the responses received from the back-end.

2. Microservice Module Integration

- Understand the interaction between different modules in the microservice architecture. Each service should be loosely coupled and should communicate with each other through well-defined APIs.
- The driver starts integrating two services while the navigator reviews the code, suggests improvements, and thinks about potential issues with the integration.
- Switch roles, refine the integration, and verify its correctness by writing and running integration tests.

3. Code Refactoring

- Review the code together to identify parts that can be made more efficient, readable, or maintainable.
- The driver starts refactoring the code while the navigator thinks about potential impacts on other parts of the codebase.
- Switch roles, continue the refactoring process, and make sure to run tests to ensure the refactoring did not break anything.

4. Documentation

- Document the interfaces, integrations, and important parts of the codebase for future reference.
- Both the driver and navigator contribute to the documentation, with one typing and the other reviewing and suggesting additions or changes.

Decision

Create decision

Decision	Status	Stakeholders	Outcome	Due date	Owner
MedCAT - Use Single Processing Instead of Multiprocessing	DECIDED	Clients	Use single processor to predict SNOMED CT codes	-	KUNXI SUN
Use Github Actions Instead of Ansible Script	DECIDED	Team members and the next team	Change the way for continuous integreation	-	KUNXI SUN
No Continuous Deployment	DECIDED	Clients and development team members	No continuous deployment	-	-
Github Branch Workflow	DECIDED	Team member and clients	Github branch workflow	-	KUNXI SUN
Combine Ontoserver with MedCAT	DECIDED	Clients	Combine Ontoserver and Medcat	-	Chenyang Dong
Use Apifox API Test to Replace Automatic Unit Test	DECIDED	Team member and clients	No automatic unit test	-	KUNXI SUN
Use Apifox Instead of Postman	DECIDED	Development Team	Use ApiFox to make API documentations	-	KUNXI SUN
Technique Stack	DECIDED	Team members	Technique stack	-	-
Role Assignment	DECIDED	Team members	Role assignment	-	-

Combine Ontoserver with MedCAT

Version ID	Description	Date
2.0.0	Further decision why team decide to combine Ontoserver and Medcat in the mapping algorithm	25 May 2023
1.0.0	A basic description about why team decide to use MedCat for the mapping algorithm	27 Apr 2023

Version 2.0.0

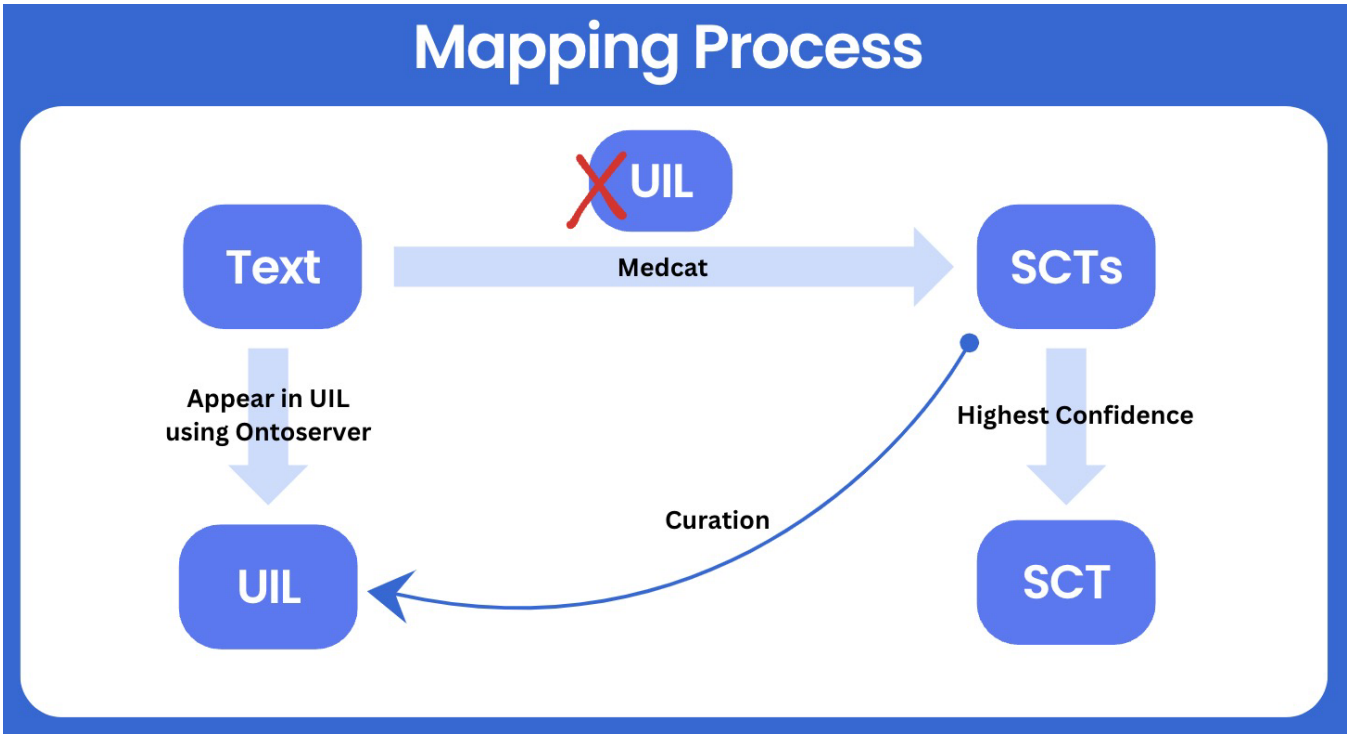
Status	DECIDED
Stakeholders	Clients
Outcome	Combine Ontoserver and Medcat
Due date	-
Owner	Chenyang Dong

Background

- Information about Ontoserver and Medcat including their usage can be found in Background.
- Using Medcat alone provide high performance on mapping speed and accuracy. It prodcide trustworthy result on mapping short text to SNOMED-CT code. However, the final goal for the product is to map the clinical texts into code in Universal Indication List(UIL). Therefore, team decided to use Ontoserver to map the SNOMED-CT into UIL classes.

Action items

- [x] Connect the current map service with Ontoserver



Version 1.0.0

- Due to change of mapping requirements and and recent notice that the deployment environment does not allow constant use of external endpoints , our previous mapping approach has to be deprecated.
- Based on 2023-04-28 Meeting notes - Client, the team is going to implement MedCAT to map raw text to SNOMED CT as one of the approaches.

MedCAT - Use Single Processing Instead of Multiprocessing

Status	DECIDED
Stakeholders	Clients
Outcome	Use single processor to predict SNOMED CT codes
Due date	-
Owner	KUNXI SUN

Background

Performance test of Medcat is applied in Performance Test Report.

Although the Medcat allows multiprocessing to predict, it should to be avoided due to that the product is deployed on a single node(a server/ a virtual machine instance), use multiprocessing technique have negative effect on other services' performance. However, if the Medcat is deployed on a single node with isolated resources(cores, memories), it is better to use the multiprocessing of the Medcat.

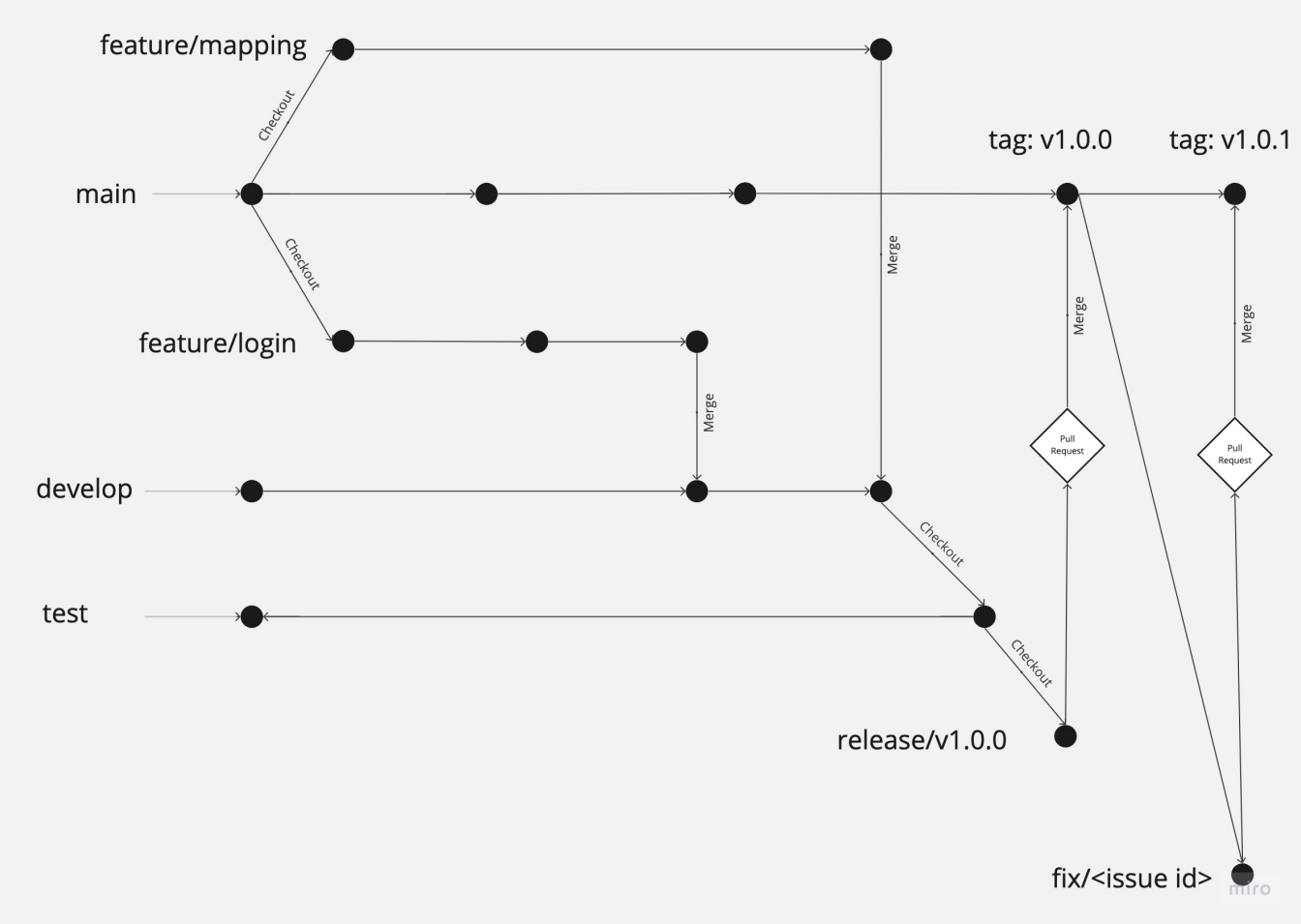
Github Branch Workflow

Status	DECIDED
Stakeholders	Team member and clients
Outcome	Github branch workflow
Due date	-
Owner	KUNXI SUN

Background

Team member should follow the following graph to make commit and create branches.

Action items



No Continuous Deployment

Status	DECIDED
Stakeholders	Clients and development team members
Outcome	No continuous deployment
Due date	-
Owner	-

Background

Based on communication with SRC and clients started at sprint 1, team has tried to get a production server. However, team did not get production server until sprint 3. Therefore, team member decided to get rid of Continuous Deployment(CD) in this project. Team will rent a server as development server for continuously integration.

Update: the production server is offered by MRC at the end of Sprint 3 (May 30, 2023)

Action items

- [x] Rent a development server from Cloud Service Provider
- [x] Get rid of continuously deployment

Use Apifox API Test to Replace Automatic Unit Test

Status	DECIDED
Stakeholders	Team member and clients
Outcome	No automatic unit test
Due date	-
Owner	KUNXI SUN

Background

Consider that team has derived further functional requirements(Code system management, team member management), and there are only few user stories therefore team is short of time in the third sprint. Therefore, in order to finish as much user stories as possible, team decided to use Apifox to write test cases for each API.

Action items

- [x] No automatic unit testing
- [x] Write test cases in ApiFox to cover as much code coverage as possible

Use Apifox Instead of Postman

Status	DECIDED
Stakeholders	Development Team
Outcome	Use ApiFox to make API documentations
Due date	-
Owner	KUNXI SUN

Background

The free plan of postman has limitation that it only allow three people in a team for the API documentations, therefore the team decided to use Apifox to do API documentation.

Action items

- [x] Copy and paste the APIs from postman to Apifox
- [x] Create team and invite members into Apifox.=

Use Github Actions Instead of Ansible Script

Status	DECIDED
Stakeholders	Team members and the next team
Outcome	Change the way for continuous integreation
Due date	-
Owner	KUNXI SUN

Background

Team has decided and implemented Ansible script to integrate service components on the development server. However, in the later sprint, team get notification to use Github Actions to do ChatGPT code review. Therefore, team decided to get rid of Ansible script for deployment, and pay extra effort on implementing Github Actions scripts. Not just for the ChatGPT automatic code review, but also for the automatic continuous integration and deployment.

Action items

- [x] Delete the ansible module on Github
- [x] Implement the Github Actions

Role Assignment

Status	DECIDED
Stakeholders	Team members
Outcome	Role assignment
Due date	-
Owner	-

Background

Name	Preferred Name	Role	Responsibility
Kunxi Sun	Quincy	Product Owner	• Manage the product backlog • Prototype Design • Github repository setup and management
Chenyang Dong	Peter	Scrum Master	• Responsible for ensuring that the Scrum framework is followed • Negotiation and communication with Client and Supervisor
Hanyi Gao	Henry	Development Team Member	• Prototype Design • Determine the architectural decisions
Yulai Luo	Ricardo	Development Team Member	• Verify that the documents meet the standards and guidelines • Meeting Minutes
Yue Fei	Molly	Development Team Member	• User experience on prototype • Review the task

Technique Stack

Status	DECIDED
Stakeholders	Team members
Outcome	Technique stack
Due date	-
Owner	-

Update: The technologies to use in this project has been decided based on the following technique stack.

Background

Rate the familiarity with the following technologies on a scale from 1 to 5.

Name	Kunxi Sun	Yulai Luo	Yue Fei	Chenyang Dong	Hanyi Gao
Major	AI	AI	AI	Distributed Computing	Computing
Python	5	4	5	5	3
Java	3	4	3	3	3
C	3	1	2	2	2
Golang	0	0	0	0	0
Javascript	2	2	3	4	5
Spring	0	3	0	0	0
Flask	4	0	0	0	0
Django	0	1	0	1	0
Pandas	3	3	5	5	3
Scikit-learn	3	1	5	5	1
Numpy	4	4	5	5	3
Matplotib	4	3	5	5	3
Tensorflow	4	1	3	3	0
Keras	3	2	3	3	0
Pytorch	4	2	4	3	0
React	0	3	0	3	5
Vue	0	3	0	0	1
Html	4	5	4	4	5
Css	2	5	4	4	5
Figma	5	0	0	0	5
Adobe XD	2	0	0	3	2
Sketch	0	0	0	0	0

Others

Title	Creator	Modified
Demonstration	Hanyi Gao	about 8 hours ago
Server Configuration	Ricardo Luo	01 Jun, 2023
Tool Uses	KUNXI SUN	31 May, 2023
Mapping Tools Survey	KUNXI SUN	30 Apr, 2023
Code Review Checklist	Yue Fei	30 Apr, 2023

Demonstration

> *Note: Due to the complexities and challenges in advancing the deployment of the Secure Research Environment, and the intricate approval processes involved, we have mutually agreed with our clients to abstain from final deployment at this stage. Therefore, the system you'll see in this demonstration is running on Melbourne Research Cloud instance or locally on our host machine.*
>
> *The production environment using Melbourne Research Cloud instance can be accessed at http://172.26.131.202. Please note that access is only available via the University of Melbourne's Wi-Fi network or through a Cisco VPN. This is due to the fact that the website is hosted on an internal network for security reasons. This instance will not be available in the near future.*

Sprint 4 video demo:

https://youtu.be/FCiu6JOqIio

Sprint 3 video demo:

https://youtu.be/BC8NPPdGJ6M

Sprint 2 video demo:

https://www.youtube.com/watch?v=zaSCr7h_XnI

Presentation

http://youtube.com/watch?v=lDwuwGa9t3g

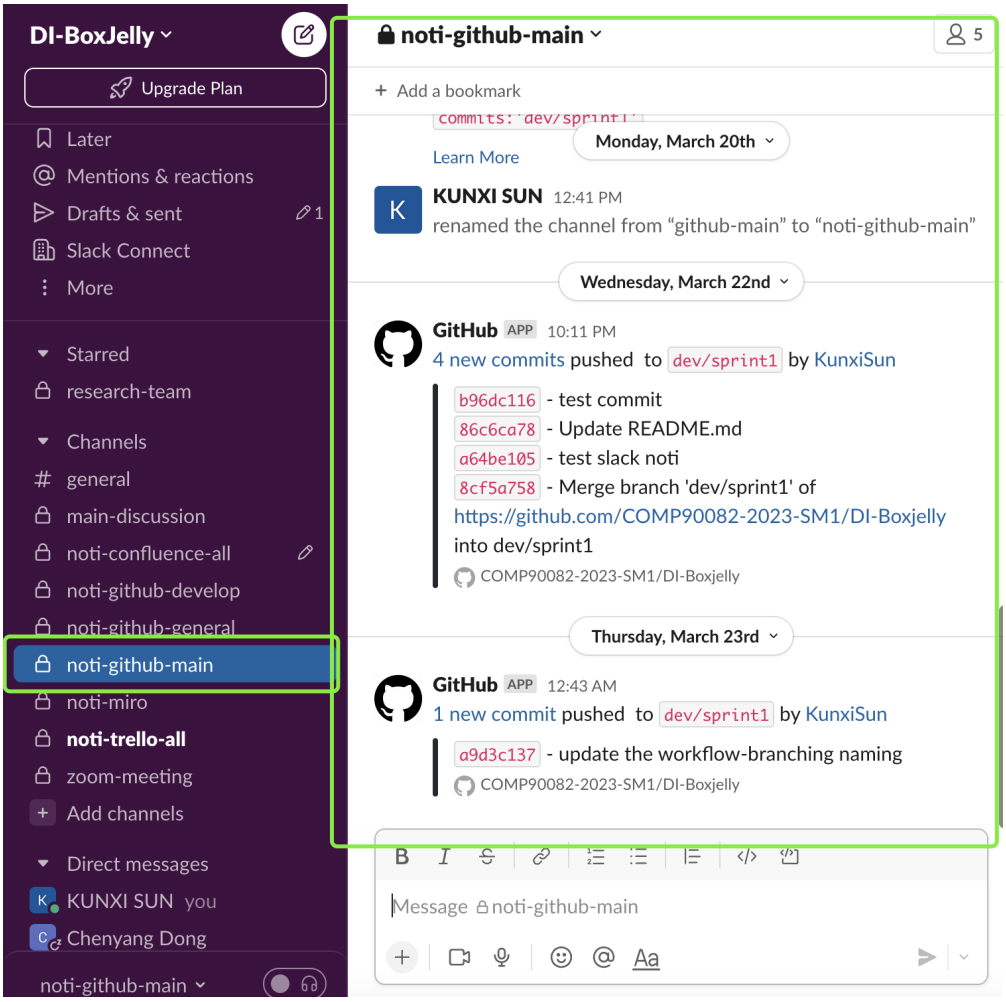
> *Below is the presentation slides.*

Tool Uses

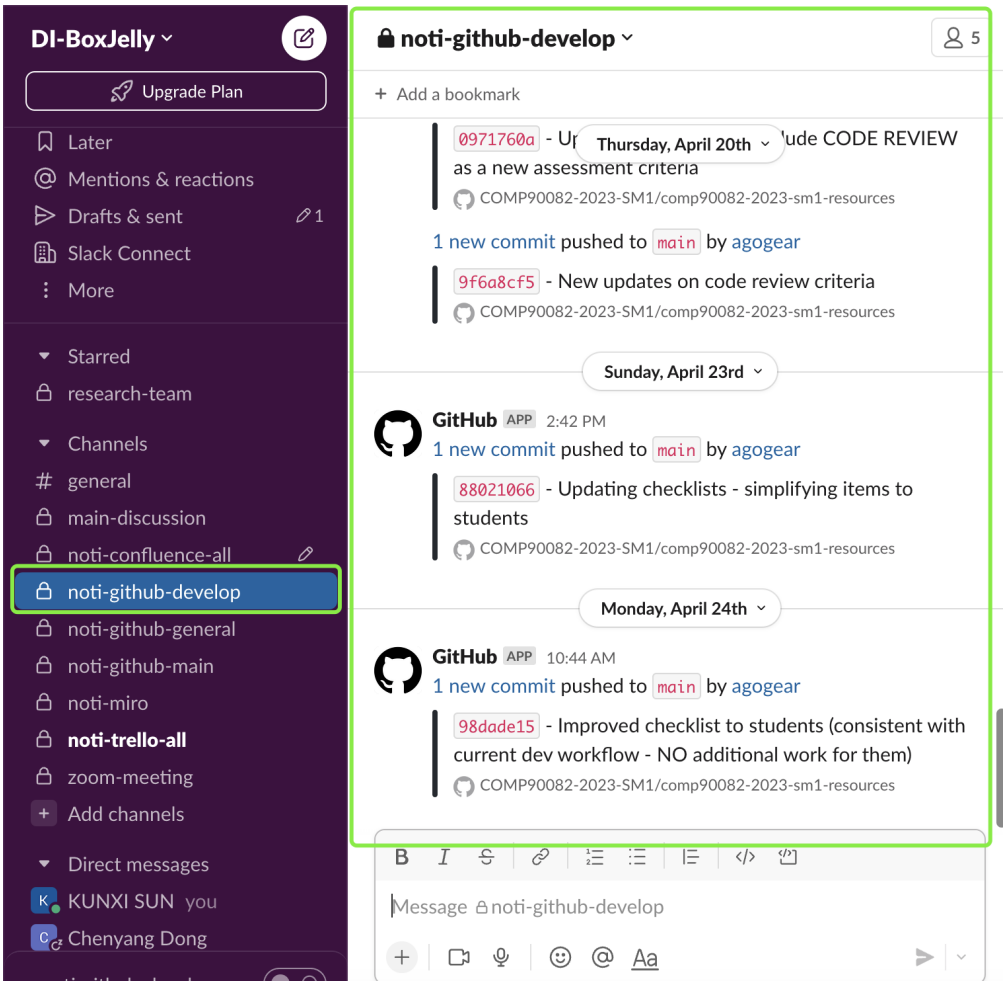
1 Integrate Github on Slack

Integrate GitHub on Slack helps the team track the commit on important branches: develop and main branch

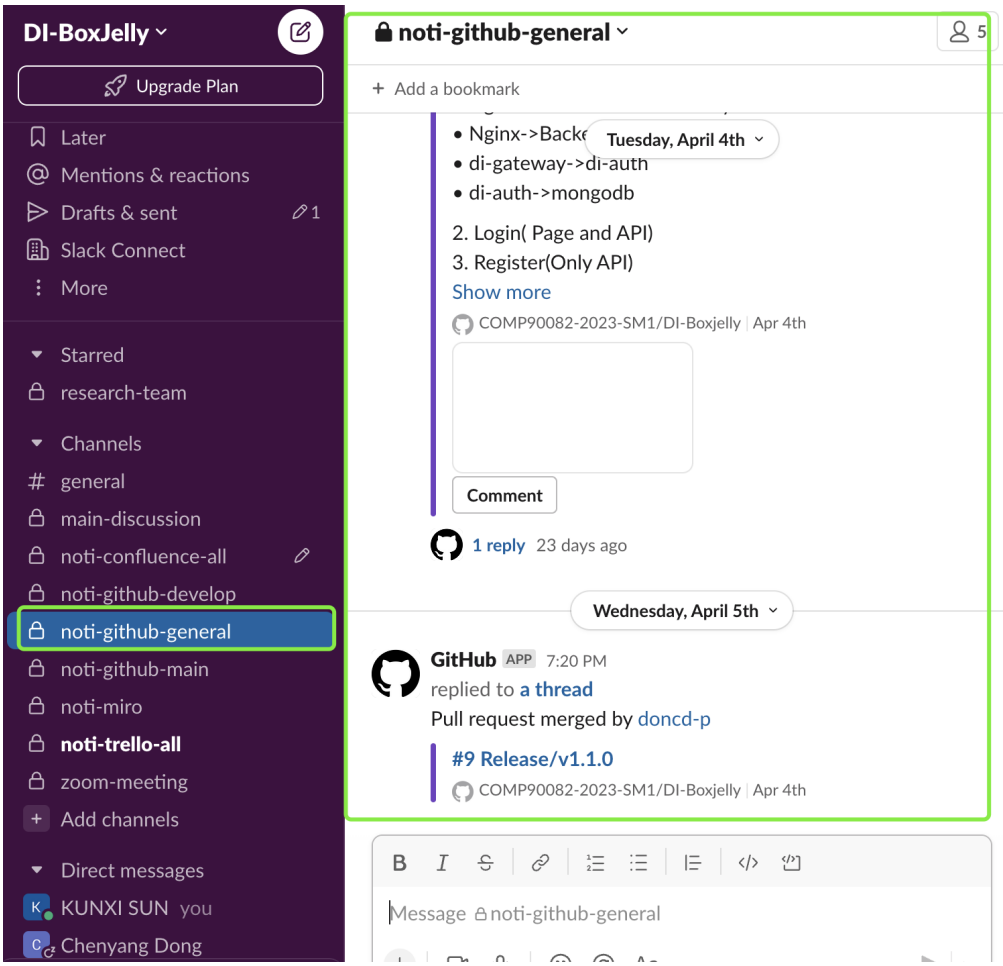
Example 1: Main branch notification on Slack



Example 2: Develop branch notifictaion on Slack



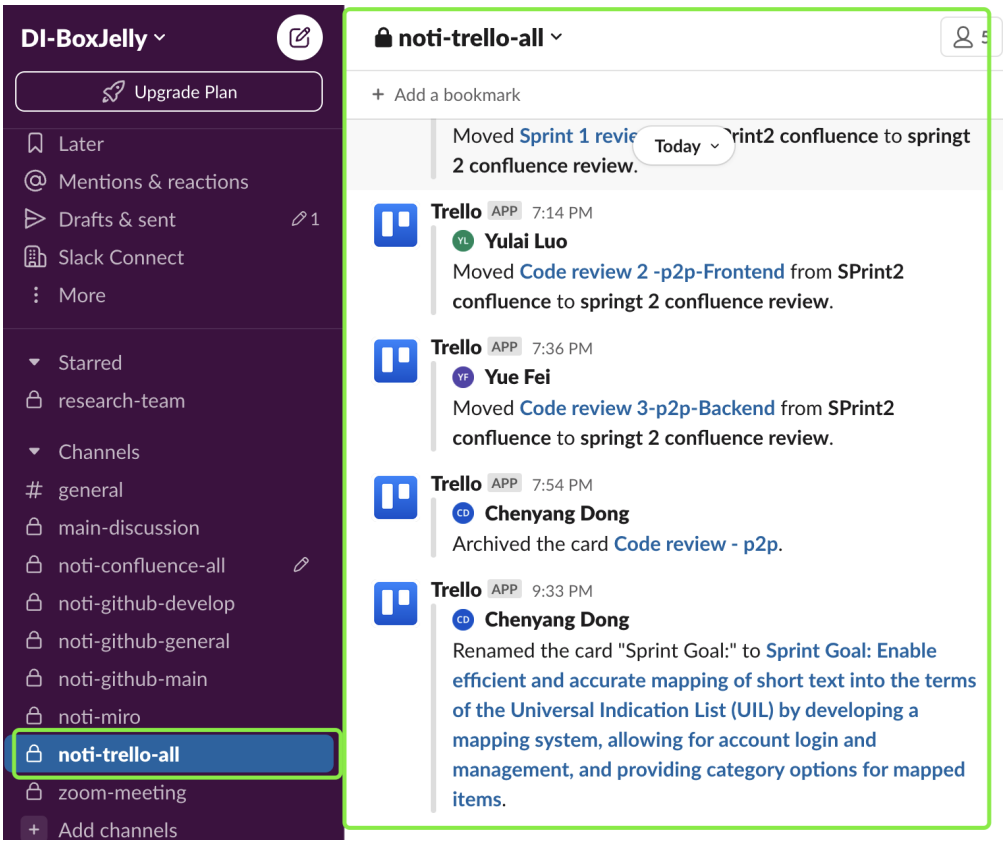
Example 3: General information on Slack



2 Integrate Trello on Slack

Integrate Trello on Slack helps team track the status of user story cards.

Example 1: Trello card notifications on Slack



3 Integrate Confluence on Slack

FAIL

Reason: Permission from unimelb Confluence administration required.

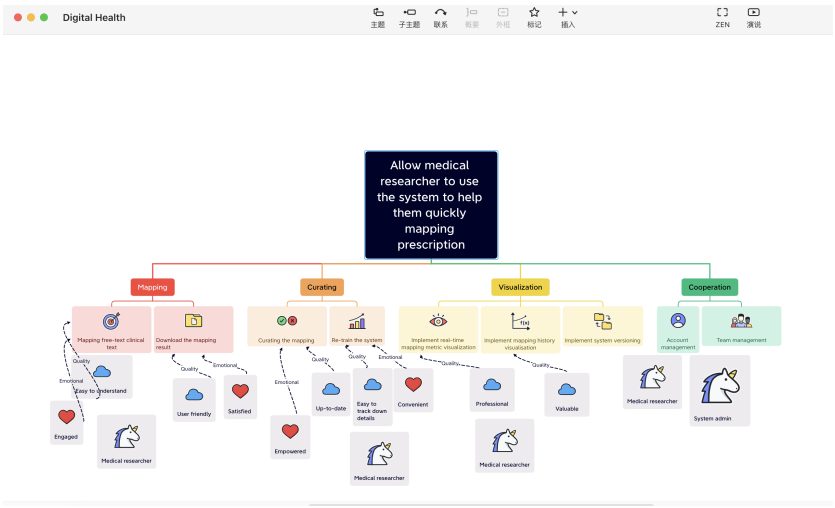
4 Integrate Zoom on Slack

FAIL

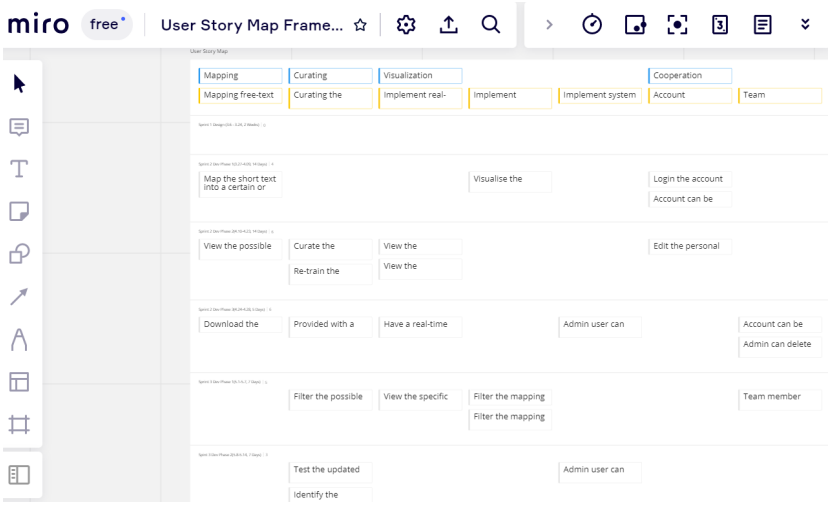
Reason: Permission from unimelb Zoom administration required.

5 Xmind

Xmind is a very powerful tool which helps team to draw the motivational model.

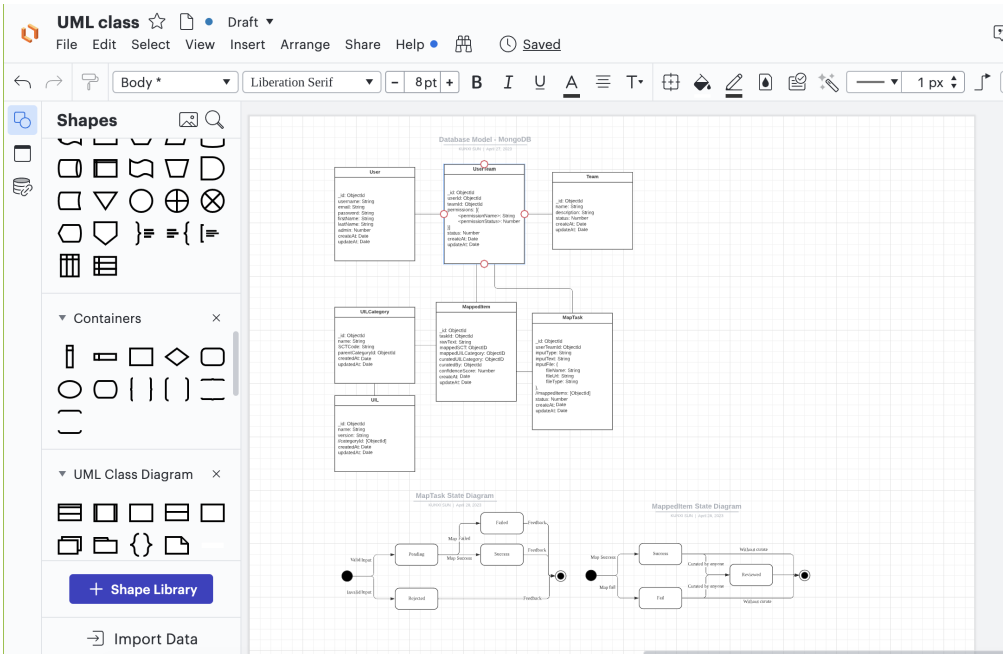


6 Miro



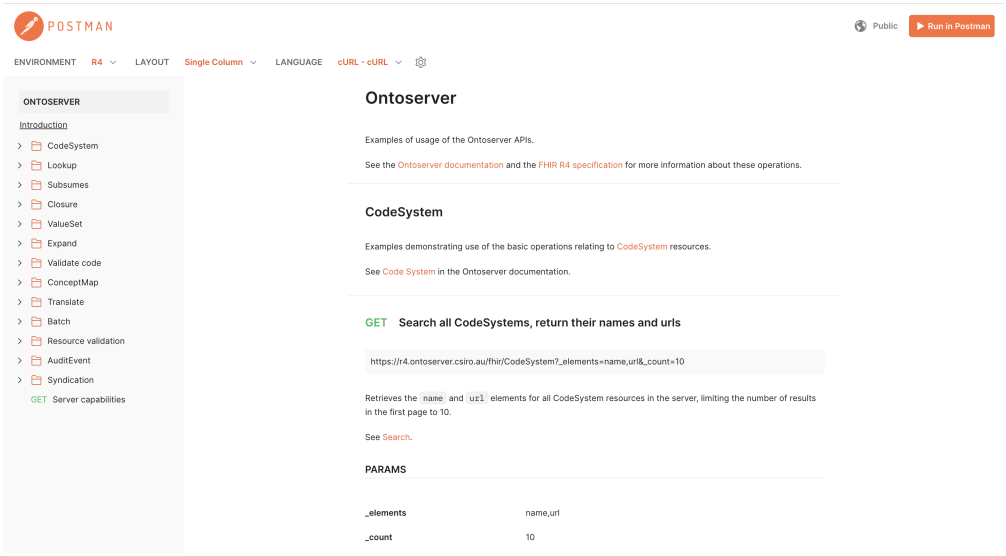
7 Lucidchart

Lucidchart is a very powerful tool to draw diagrams. It has been used to produce the 4+1 architecture diagrams.



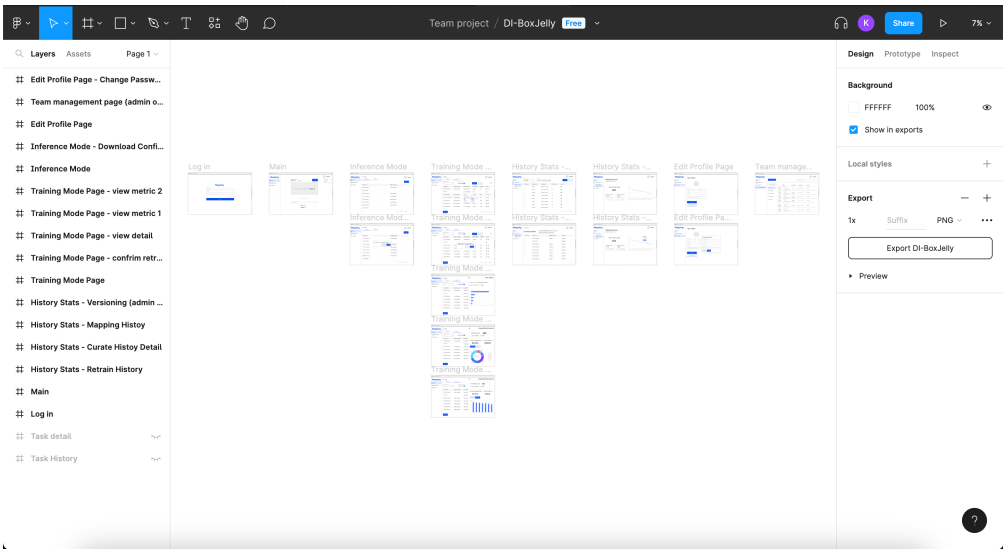
8 Postman

Ontoserver provides their APIs through Postman, so team members use postman very often to test the Ontoserver API, and test our own backend APIs.



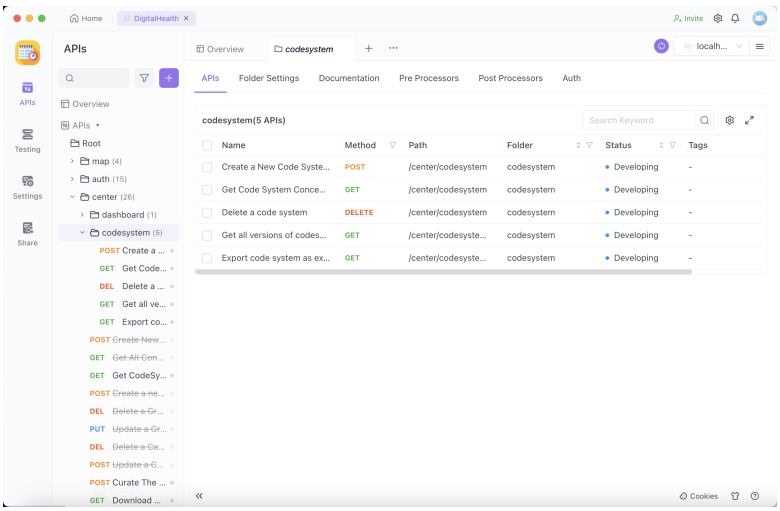
9 Figma

Figma is a collaborative ui design tools which has been used to produce the digital prototypes.



10 Apifox

Apifox is a free application for API documentation, testing tool. It has similar functions to Postman but with free plan.



Server Configuration

Dev Cloud Server

Public IP	101.43.110.249
Password	******
Instance ID	lhins-6y84pbke
username	ubuntu
Area	Shanghai
secret key	lhkp-j8q3flr4

Production MRC server (Obtatined at May 30, 2023)

> *To use this server, you need to connect to the Unimelb wifi, or use the Anyconnect VPN to connect to Unimelb network.*

Public IP	115.146.95.215
Password	******
Instance ID	-
username	ubuntu
Area	Melbourne(MRC)
secret key	-

Develop Database

Address	http://101.43.110.249:27017/
admin username	boxjelly
admin password	di_boxjelly90082
Connection URL	mongodb://boxjelly:di_boxjelly90082@101.43.110.249:27017/

Develop Web

Address	http://101.43.110.249/

Production Web

Address	http://115.146.95.215/

Code Review Checklist

The following checklist is used for our analysis:

1. Documentation Defects:
    a. Naming: Assess the quality of software element names.
    b. Comment: Analyze the quality and accuracy of code comments.
2. Visual Representation Defects:
    a. Bracket Usage: Identify any issues with incorrect or missing brackets.
    b. Indentation: Check for incorrect indentation that affects readability.
    c. Long Line: Point out any long code statements that hinder readability.
3. Structure Defects:
    a. Dead Code: Find any code statements that serve no meaningful purpose.
    b. Duplication: Identify duplicate code statements that can be refactored.
4. New Functionality:
    a. Use Standard Method: Determine if a standardized approach should be used for single-purpose code statements.
5. Resource Defects:
    a. Variable Initialization: Identify variables that are uninitialized or incorrectly initialized.
    b. Memory Management: Evaluate the program's memory usage and management.
6. Check Defects:
    a. Check User Input: Analyze the validity of user input and its handling.
7. Interface Defects:
    a. Parameter: Detect incorrect or missing parameters when calling functions or libraries.
8. Logic Defects:
    a. Compute: Identify incorrect logic during system execution.
    b. Performance: Evaluate the efficiency of the algorithm used.

Mapping Tools Survey

	Ontoserver	3M HDD	Apelon DTS	Clinical Architecture Symedical	HealthTerm
Support SNOMED CT	Yes	Yes	Yes	Yes	Yes
Mapping	Yes	Yes	Yes	Yes	Yes
Automapping	Yes	No	No	No	Yes
Search Engine	Yes	Yes	Yes	Yes	Yes
RESTful APIs	Yes	Yes	Yes	Yes	Yes
Custom Mappings	Yes	No	No	Yes	Yes
Technical Support	Yes	No	Yes	-	-
Company Location	AU	US	US	US	US
Pricing	Free License held	Starts at USD $27,500 per year for development use	Basic: USD $27,000 per year With software support: additional USD $15,000 per year	Waiting for response	Waiting for response

Docker command

This page is to share some common commands of docker

What is Docker?

Docker is an open-source platform that enables developers to automate the deployment, scaling, and management of applications within containers. Docker containers package an application along with its runtime environment, libraries, and system tools - essentially, everything that the application needs to run. This allows applications to run reliably and consistently across different computing environments.

Docker Common Commands

Below are some commonly used Docker commands:

1. `docker version`: This command is used to show the Docker version information.
2. `docker info`: This command provides system-wide information regarding the Docker installation. This includes details such as the number of containers and images, etc.
3. `docker pull [OPTIONS] NAME[:TAG|@DIGEST]`: This command is used to pull an image or a repository from a Docker registry server.
4. `docker push [OPTIONS] NAME[:TAG]`: This command is used to push an image or a repository to a Docker registry server.
5. `docker run [OPTIONS] IMAGE [COMMAND] [ARG...]`: This command is used to run a command in a new container. If the image does not exist locally, Docker will try to download it first.
6. `docker ps [OPTIONS]`: This command is used to list the currently running Docker containers.
7. `docker stop [OPTIONS] CONTAINER [CONTAINER...]`: This command is used to stop one or more running containers.
8. `docker rm [OPTIONS] CONTAINER [CONTAINER...]`: This command is used to remove one or more containers.
9. `docker images [OPTIONS] [REPOSITORY[:TAG]]`: This command is used to list the images.
10. `docker rmi [OPTIONS] IMAGE [IMAGE...]`: This command is used to remove one or more images.
11. `docker build [OPTIONS] PATH | URL | -`: This command is used to build an image from a Dockerfile.

Remember to replace `[OPTIONS]`, `NAME[:TAG|@DIGEST]`, `IMAGE [COMMAND] [ARG...]`, and `CONTAINER [CONTAINER...]` with your actual parameters when you use these commands.