NEU Study

Meeting Room Reservation System

Group2: Ying Tuo Xi Shen Yuqin Luo Huixin Huang

1 Overview

PROJECT DESCRIPTION

NEU Study Room Reservation System(running on port 3000)

The design for the system is mainly for Northeastern University students to reserve a meeting room in the library. The basic flow of reservation will be as follows:

Students log in to the system
Students choose the campus(Boston, Seattle, Silicon Valley, Portland)
Students choose the building
Students choose the time slot for the reservation
Students select a meeting room
Students submit the reservation ticket
The system confirms the reservation and shows confirmation

2 Demo

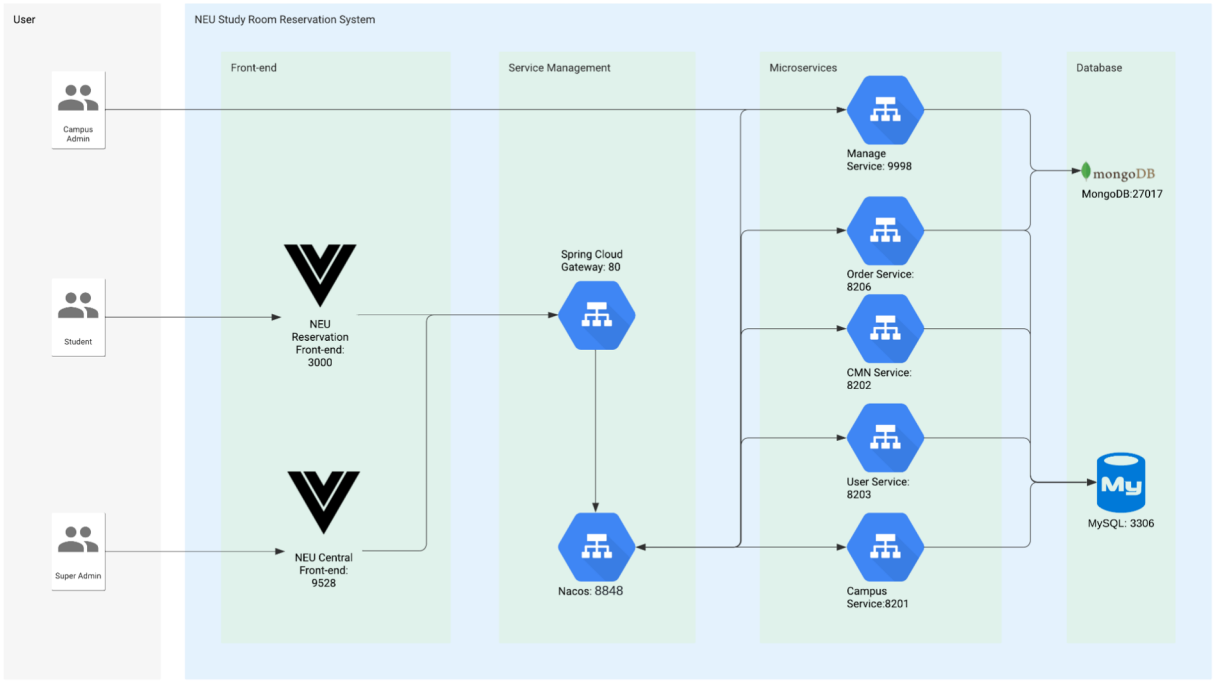
NEU Study Room Reservation System Demo

The video demonstrates how the three systems work.

Link: Link: https://drive.google.com/file/d/1pmWcS6g3PeoOeIPtaRn_BFvAaVsoGhGz/view?usp=sharing

3 Architecture

Architecture Overview Diagram (AOD)



User layer: Three types of user: Campus admin, Students, and Super admin.

Front-end layer: NEU Reservation Front-end and NEU Central Front-end

Service Management layer: Spring Cloud Gateway for solving cross origin problem, and Nacos for calling the microservices

Microservices layer: Five microservices

Database layer: MongoDB and MySQL

PROJECT DESCRIPTION

NEU-Silicon Valley Campus(running on port 9998)

This system enables a campus administrator to manage their own campus.
The SV campus is taken as an example to illustrate how the system works.

- Campus set: It can edit the basic set of this campus.
- Campus management: It can add detailed information about the campus.
- Department list: It can add or delete the study rooms.
- Schedule list: It can add or delete the available study room schedules.

PROJECT DESCRIPTION

NEU Study Central Server Management(running on port 9528)

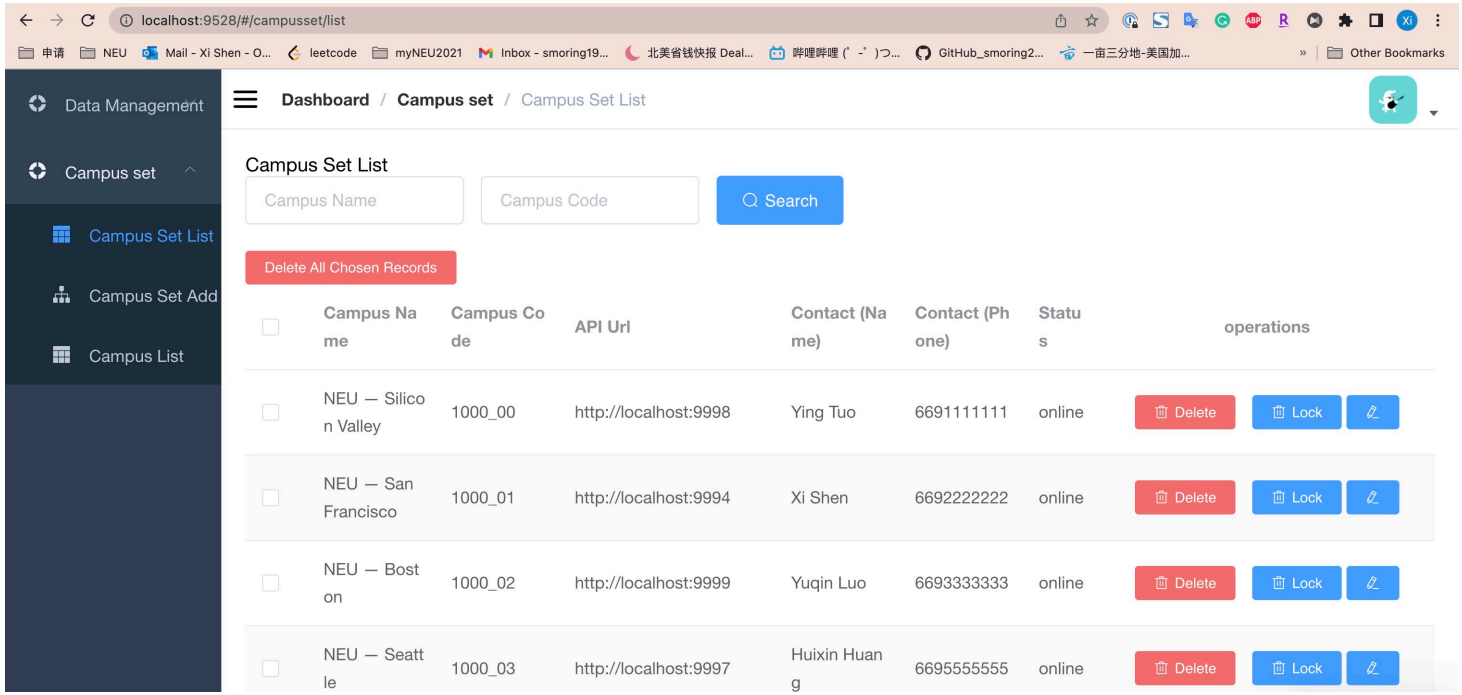
The design for the system also provides a website named NEU Study Central Server Management, which enables the super administrator to manage all the campuses. It can

- Data Management: Data Dict(Data dictionary) is the classified data or some fixed data commonly used in the management system. The relationships of the location data will show here.
- Campus Set: The super administrator can manage all campuses. It can delete, lock, edit or add any campus. It can also see the details of the campus through Campus List.

3 Architecture

PROJECT DESCRIPTION

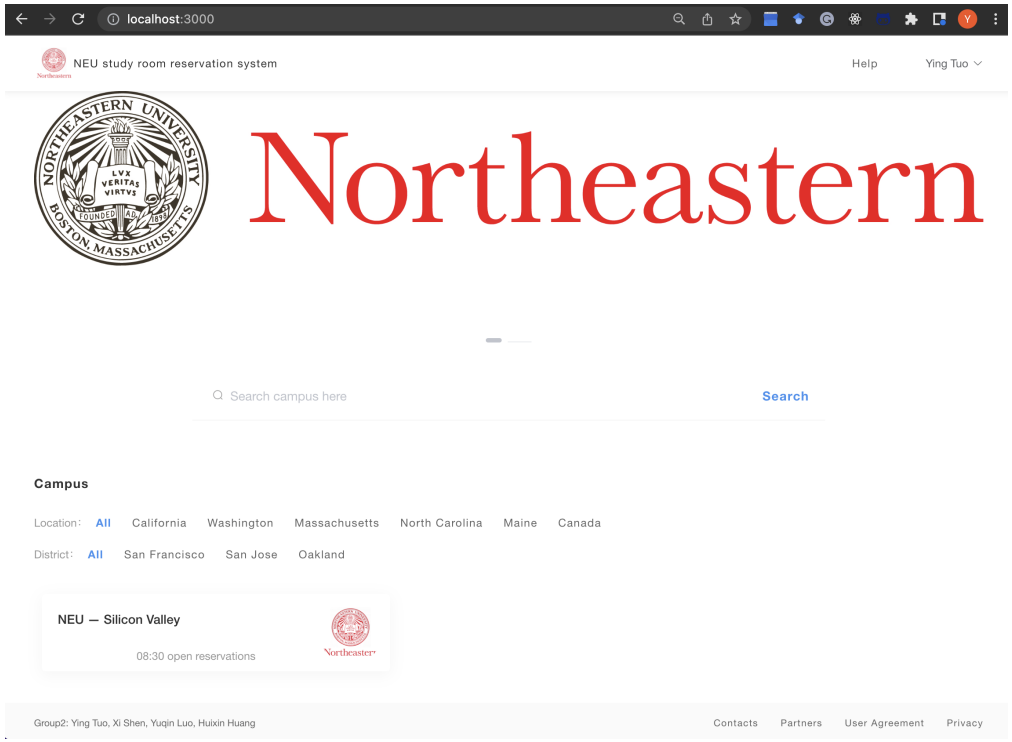
NEU Study – Super Admin Server (running on port 9528)



3 Architecture

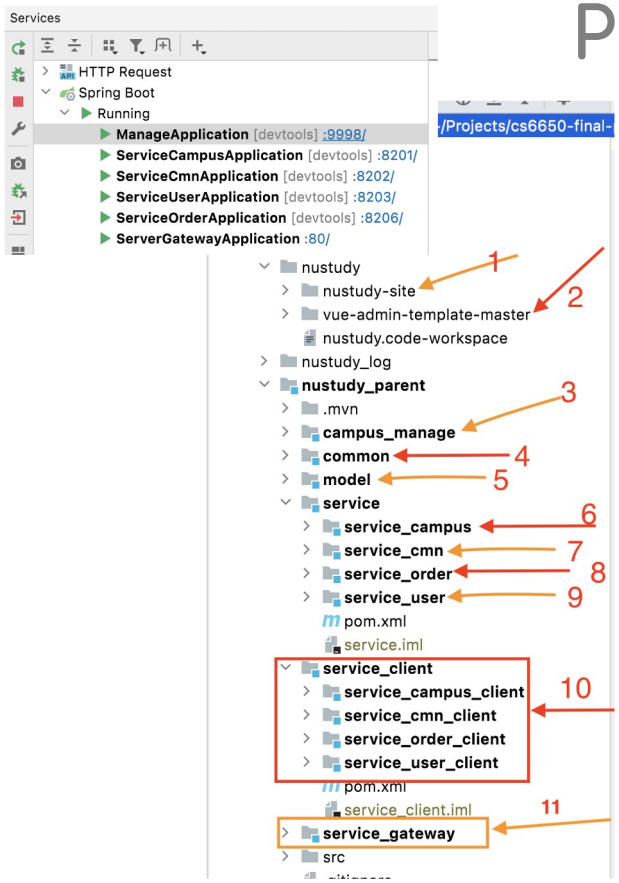
PROJECT DESCRIPTION

NEU Study Student (running on port 3000)



3 Architecture

Project Code Structure



1. **Front-end UI:** NEU Study Room Reservation System(running on port 3000)

2. **Front-end UI:** NEU Study Central Server Management(running on port 9528)

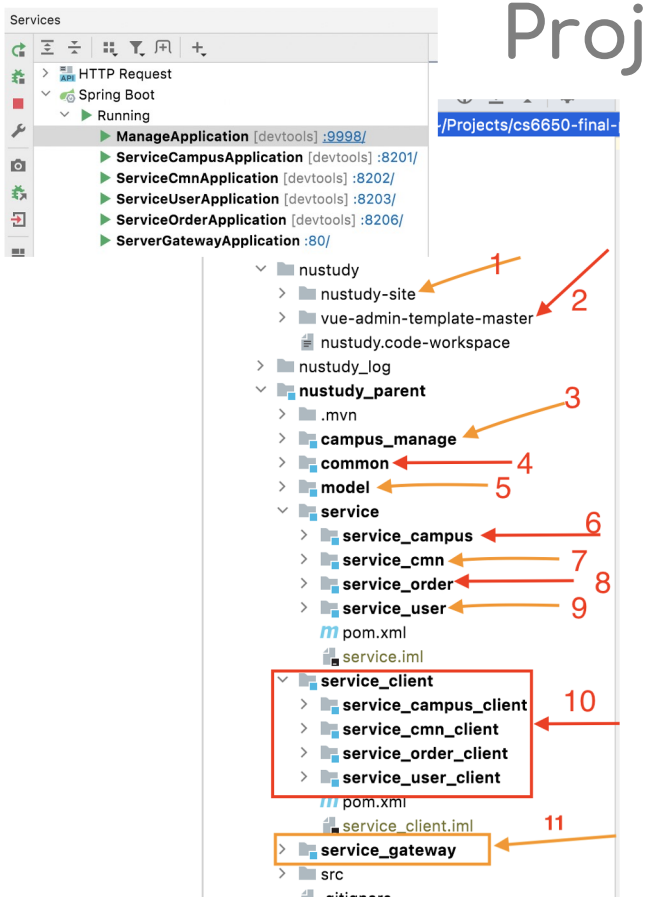
3. campus_manage(port 9998): NEU-Silicon Valley Campus, with front-end and back-end together. Connected with MongoDB(port 27017).

4. common: It provides the utilities for this project, including:
   common_util:Tool module
   rabbit_util: RabbitMQ encapsulation
   service_util: A toolkit for the microservices, including the public configuration class of the microservices, on which all microservices modules can import as dependency.

5. model: Entity class module.

3 Architecture

Project Code Structure



6. service_campus(port 8201): Campus API microservice including campus set, department, and schedule. Connected with MySQL(port 3306).

7. service_cmn(port 8202): Microservice of data dictionary. Connected with MySQL(port 3306).

8. service_order(port 8206): Microservice of order, which will call all other three microservices. Connected with MySQL(port 3306) and MongoDB(port 27017).

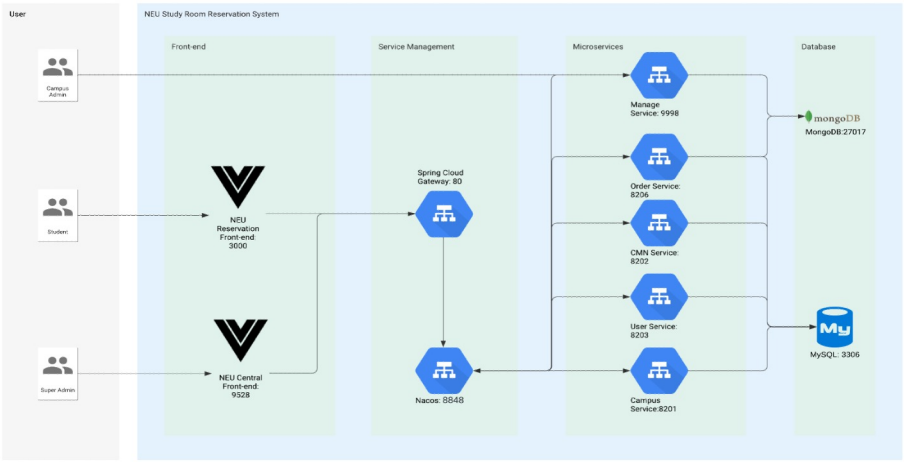
9 .service_user(port 8203): Microservice of user. Connected with MySQL(port 3306).

10. service_client: The parent node of feign service calls. Its children include cmn api interface, campus api interface, order api interface, and user api interface.

11. service_gateway(port 80): Spring Cloud Gateway provides a simple, effective and unified API routing management method for microservice architecture.

4 Algorithm

Microservice



Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features.

We designed five microservices for this system.

Manage Service (port: 9998) is developed for the campus admin.
Campus Service (port: 8201) is a central campus information management service.
CMN Service (port: 8202) is like a dictionary service that maintains the codes and their corresponding information.
User Service (port: 8203) is responsible for managing the users'(student) information.
Order Service (port: 8206) is responsible for maintaining the schedule-order records.
Details about these services is described in Report Section 3.1.

4 Algorithm

Nacos and RAFT VS PAXOS

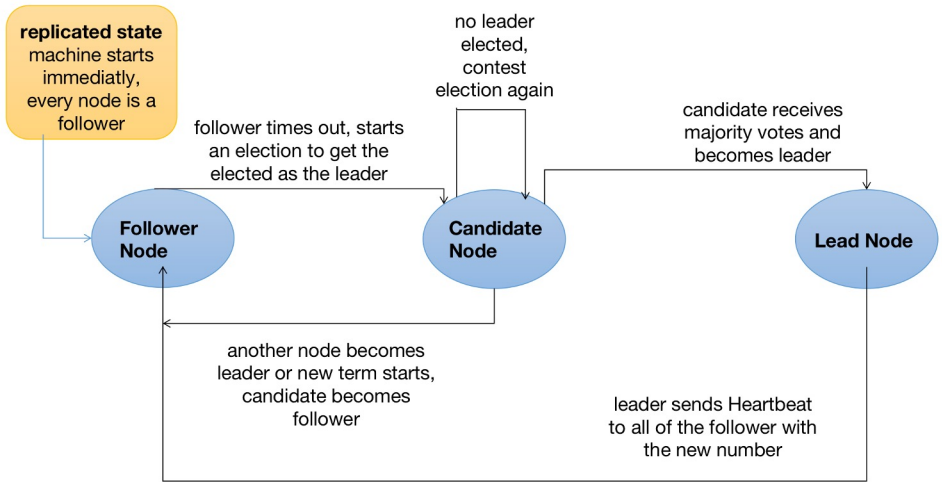
	Raft	Multi-Paxos
Leader	Strong leader	Week leader
Voting rights for the leader	Have a replica of the latest committed log	Arbitrary replica
Log replication	Guaranteed continuity	Allow voids
Log submission	Push forward the commit index	Asynchronous commit messages

The Raft algorithm divides the Server into 3 states/roles:

Leader: Responsible for Client interaction and log replication

Follower: passively responds to request RPC

Candidate: a temporary role that only exists in the leader election phase.

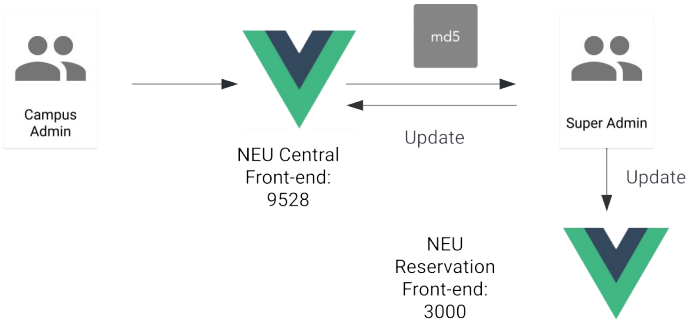


4 Algorithm

Security, Digital Signature and MD5

Strong digital signatures are an essential requirement for secure systems.

In the microservices of our project, we implement the MD5 for the *campus_management service* (campus admin). In the super admin's database, each campus has its own *sign_key*. If the campus admins would like to add, check, update or even delete (CRUD) the corresponding campus information, they need to encrypt its sign_key (private key) and then send it to the super admin (central campus information management system) for verification. If the encrypted sign_key is correct, the campus admin will get access to its corresponding campus information and the request will be processed.



4 Algorithm

Message Queue and RabbitMQ

Indirect Communication plays a role of great importance in a distributed system. Message queues (distributed message queues) are a further important category of indirect communication systems. Message queues provide a point-to-point service using the concept of a message queue as an indirection, achieving the desired properties of space and time uncoupling.

In our project, we applied message queues with RabbitMQ for the communication of each microservices. For example, one of our microservices, *service_order* sends messages to the database by RabbitMQ. Queues in RabbitMQ are ordered collections of messages. Messages are enqueued and dequeued (delivered to consumers) in the FIFO manner (FIFO ordering is not guaranteed for priority and shared queues).

Other Tech Stacks in our project

Backend

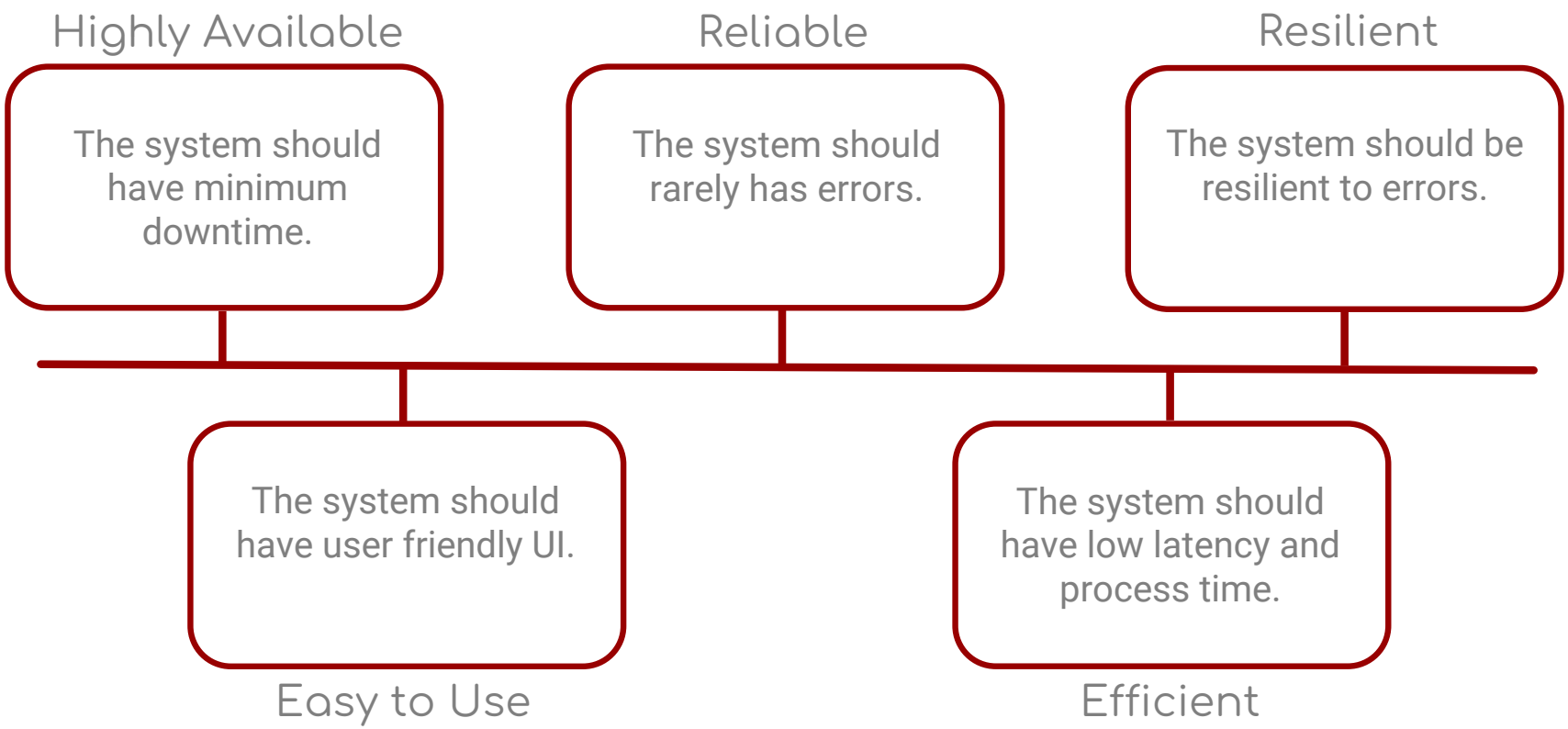
- MyBatis (MyBatis-Plus)
- Swagger2: we use this tool to test the APIs
- Lombok: is a java library that automatically plugs into the editor and builds tools, spicing up the java code
- MySQL: store the data (we may also use MongoDB as backup)
- SpringCloud：The libraries we will use in our project are: SpringCloudGateway, Spring Cloud Alibaba Sentinel, SpringCloud Task, etc.)

Frontend

- Vue.js: the framework to build our project UI
- Node.js: our project UI will be based on NodeJS and use JavaScript
- Axios: to handle the promise and asyn operations
- NPM, Babel, Webpack: manage the packages in Vue

5 Result

EXPECTED RESULTS



REFERENCES

[1] Coulouris, George, Jean Dollimore, and Tim Kindberg. "Distributed Systems: Concepts and Design Edition 5." System 2.11 (2012): 15.

[2] Lamport, Leslie, "Paxos made simple." ACM SIGACT News {Distributed Computing Column} 32, 4 (Whole Number 121, December 2001) (2001): 51 - 58

[3] Lamport, Leslie, Robert Shostak, and Marshall Pease. "The Byzantine general problem." Concurrency: the Works of Leslie Lamport,. 2019.203-226

[4]https://www.alibabacloud.com/blog/paxos-raft-epaxos-how-has-distributed-consensus-technology-evolved_597127

[5] Bakshi, Kapil. "Microservices-based software architecture and approaches." 2017 IEEE aerospace conference. IEEE, 2017

[6] Ongaro, Diego, and John Ousterhout. "The raft consensus algorithm." (2015): 54.

[7] Hong, Xian Jun, Hyun Sik Yang, and Young Han Kim. "Performance analysis of RESTful API and RabbitMQ for microservice web application." 2018 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2018.

[8] Rivest, Ronald. The MD5 message-digest algorithm. No. rfc1321. 1992.

Thank You!

NEU Study -

Meeting Room Reservation System