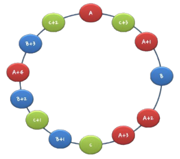
Lim, Hyeongseop

Backend Developer

Development Projects Overview



Consistent Hashing

Backend

Backend development project focusing on shard management and data redistribution systems.

2024.8 ~2024.9

STUDYCLUB

Backend

Web App for Study Recruitment and Activities: Unified platform for study groups.

2024.6 ~2024.7

Doggiverse

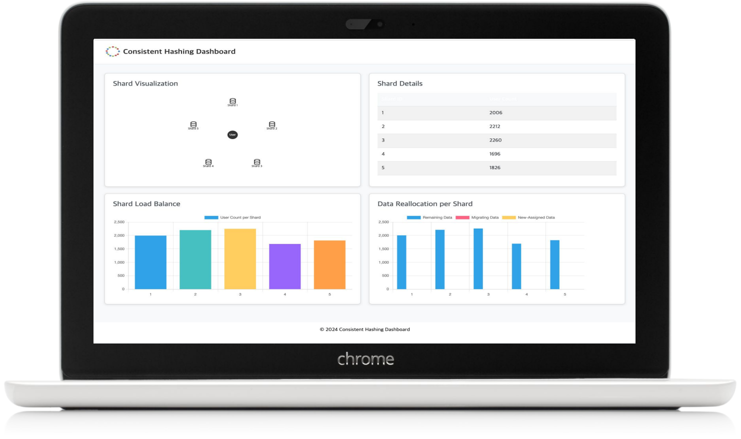
Backend

Web App for Pet Owners: Content-based community service for pet owners and their pets.

2024.5 ~2024.6

Shard Management and Efficient Data Redistribution System

Development Period	2024.8 ~2024.9
Platform	Web
Team Size	1



Consistent Hashing Dashboard

Shard Details

Shard ID	User Count
1	2006
2	2212
3	2260
4	1696
5	1826

© 2024 Consistent Hashing Dashboard

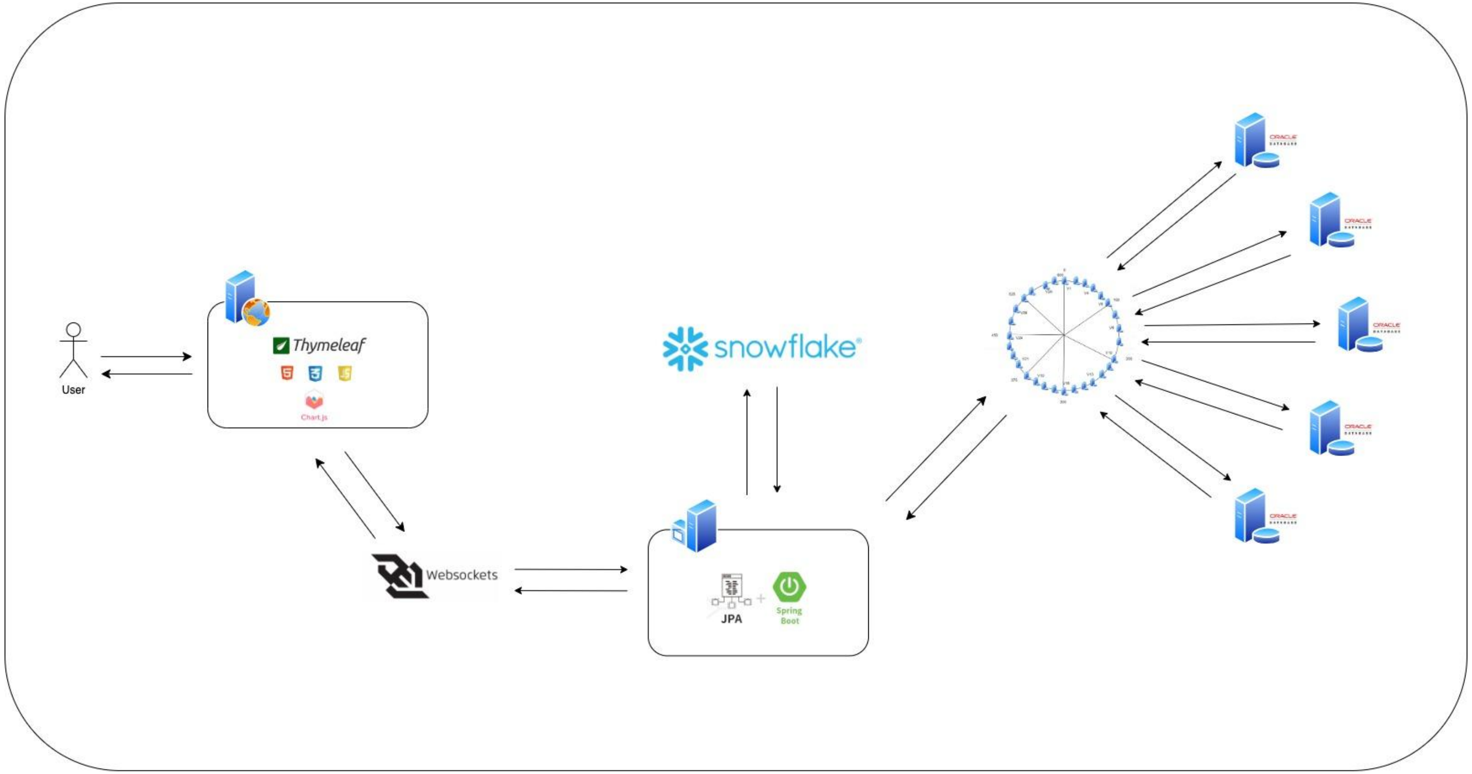
GitHub https://github.com/h3lim/Consistent-hashing

Development Environment

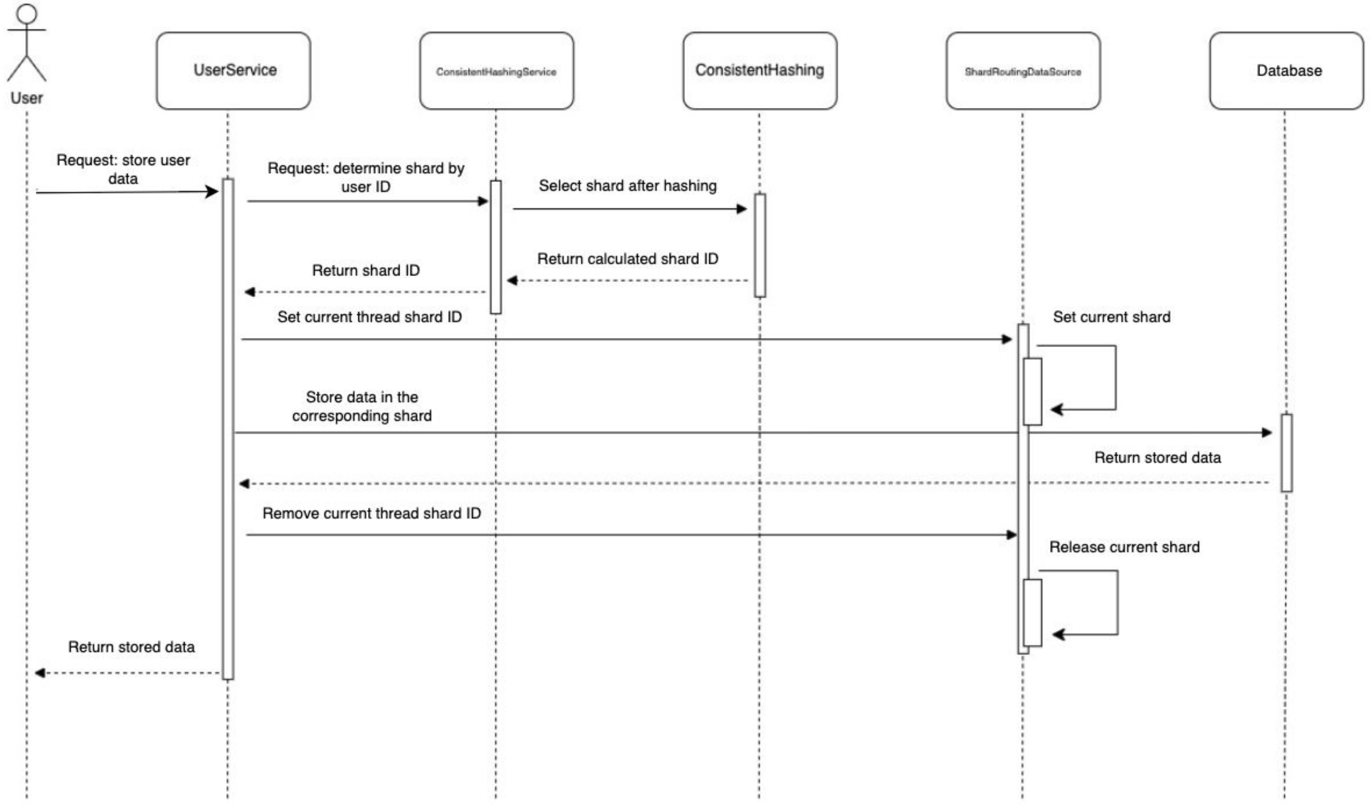
Languages	Java(JDK 17), HTML/CSS, JavaScript
Server	Spring Boot embedded Tomcat server
Frameworks	Spring Boot, JPA,Spring WebSocket
Database	Oracle DB
IDE	IntelliJ,SQL Developer,Postman
API, Libraries	WebSocket API, Thymeleaf, Guava

Project Planning and Direction (1/4)

System Architecture

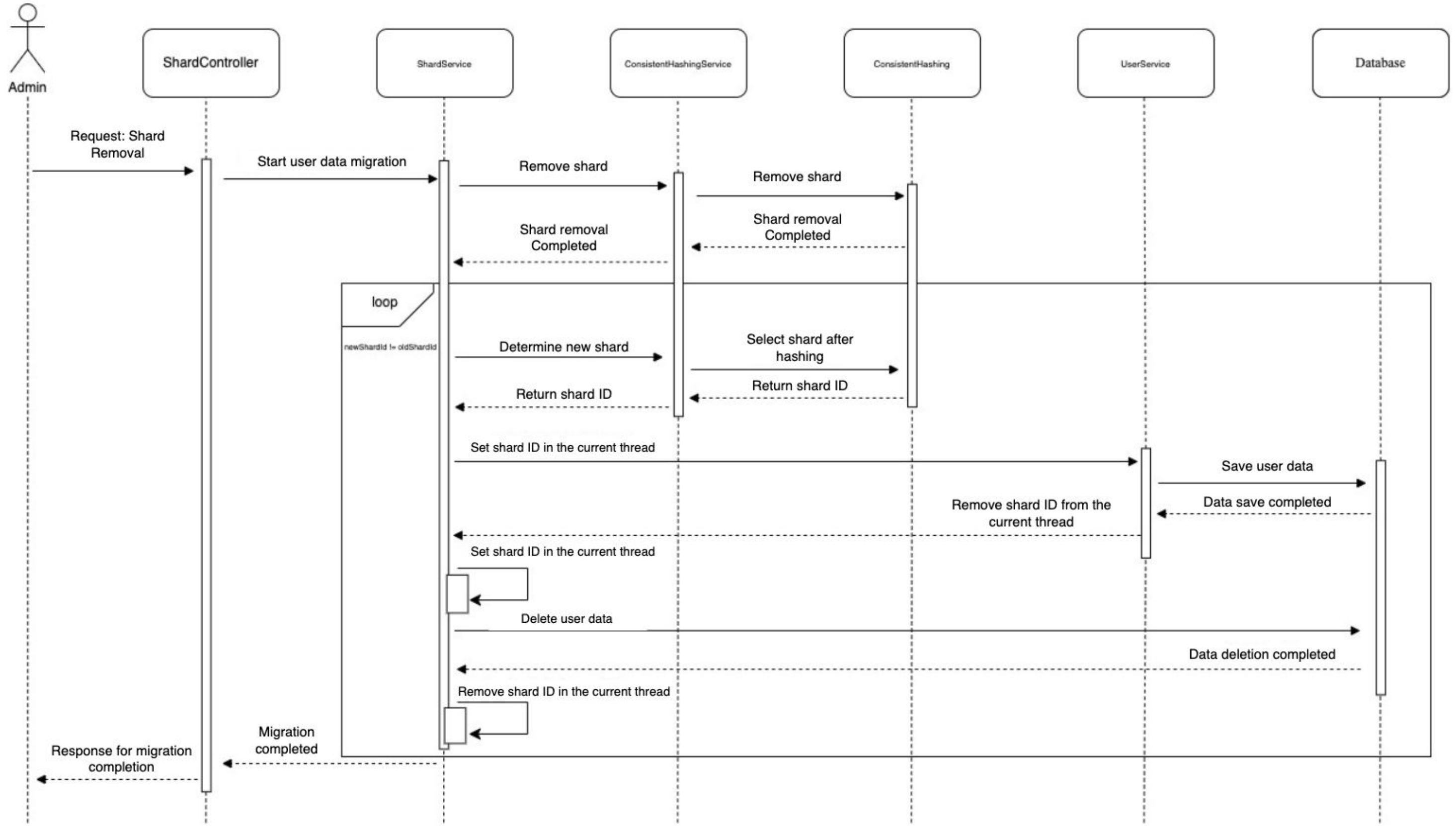


Sequence Diagram – Sharding Business Logic



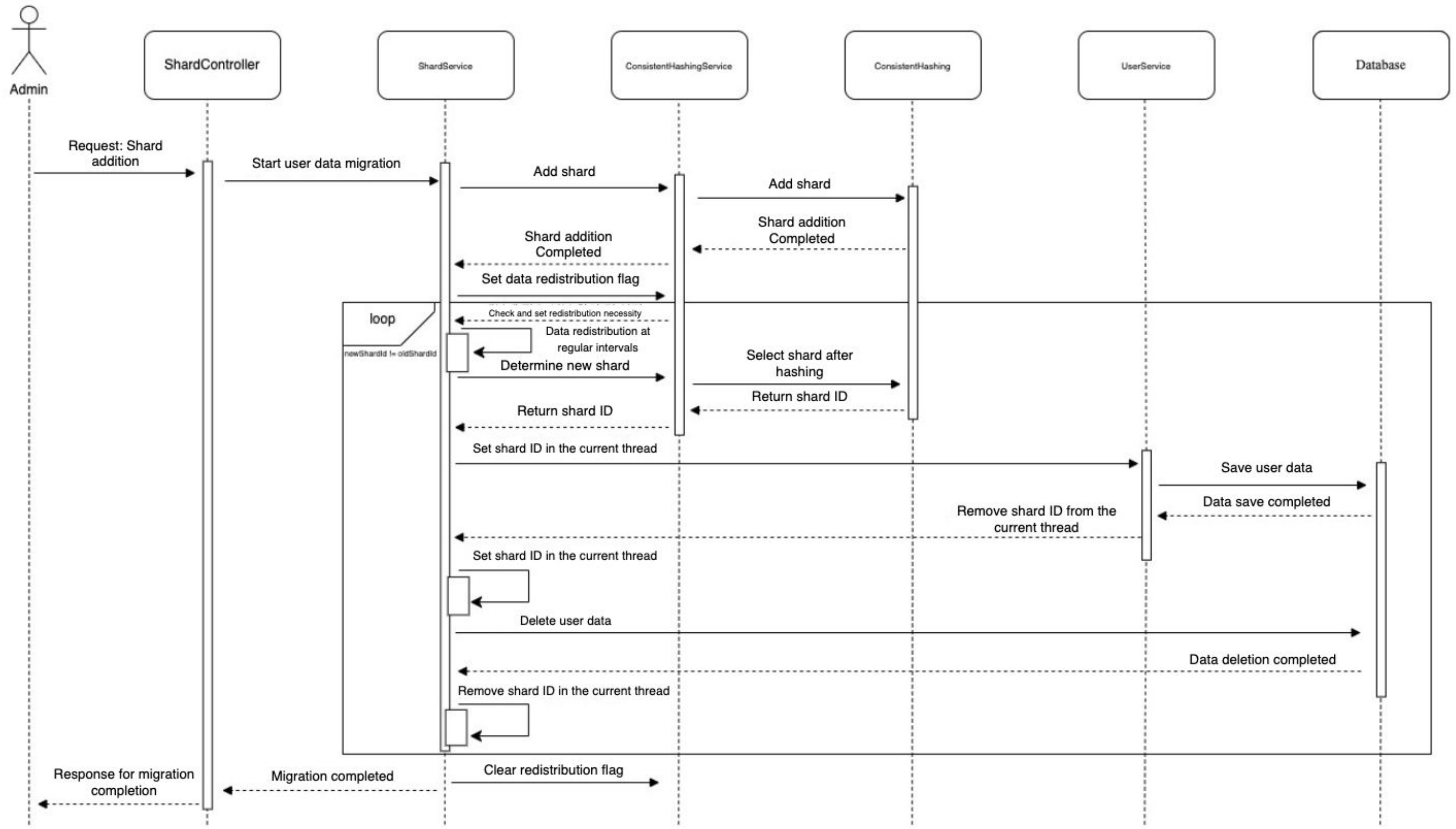
Project Planning and Direction (3/4)

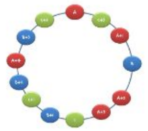
Sequence Diagram - Shard Removal and Data Redistribution



Project Planning and Direction (4/4)

Sequence Diagram - Shard Addition and Data Redistribution


Admin
ShardController
ShardService
ConsistentHashingService
ConsistentHashing
UserService
Database
Request: Shard addition
Start user data migration
Add shard
Add shard
Shard addition Completed
Shard addition Completed
Set data redistribution flag
loop
newShardId != oldShardId
Check and set redistribution necessity
Data redistribution at regular intervals
Determine new shard
Select shard after hashing
Return shard ID
Return shard ID
Set shard ID in the current thread
Save user data
Data save completed
Remove shard ID from the current thread
Set shard ID in the current thread
Delete user data
Data deletion completed
Remove shard ID in the current thread
Response for migration completion
Migration completed
Clear redistribution flag



Key Features (1/3)

Core Sharding Logic

- Data is distributed using consistent hashing and stored across multiple database shards. This method allows for handling shard additions and removals with minimal data movement, avoiding the need to redistribute all data. This enables dynamic shard management while ensuring system scalability and stability.

Implementation Details

- ConsistentHashing Class:
  - The ConsistentHashing class manages the hash ring and calculates hashes using the given number of nodes (shards) and virtual nodes. It maps user IDs or data keys onto the hash ring using the murmur3_32 hashing algorithm.
  - addNode(int shard): A method that adds a shard to the hash ring based on virtual nodes.
  - removeNode(int shard): A method that removes a shard from the hash ring by deleting the virtual nodes associated with the removed shard.
  - getNode(long userId): A method that maps a given user ID or data key onto the hash ring and returns the corresponding shard.
- ConsistentHashingService Class::
  - This service class determines which shard a user's data belongs to when storing or retrieving data from actual shards.
  - getShardForUser(Long userId): Takes a user ID as input, determines the shard the user belongs to, and manages data storage or retrieval in that shard.
  - addShard(int shardId): Adds a new shard to the hash ring and updates the shard distribution.

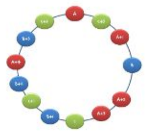
Key Features (2/3)

Unique Key Generation

- To generate unique identifiers for each user's ID or data record, a custom ID generator was implemented. This generator was inspired by the Snowflake algorithm and designed to create conflict-free identifiers in distributed systems by combining a unique node ID, a timestamp, and a sequence value.

Implementation Details

- CustomIdGenerator Class:
    - The node ID is a unique value assigned to each server or shard, ensuring that all IDs generated on that node are unique.
    - The timestamp is based on the current time, allowing the generation of multiple unique IDs even within the same time frame.
    - The sequence value prevents duplicate ID generation for the same timestamp.
- IDs are created by combining the millisecond-based timestamp, node ID, and sequence value, allowing the generation of millions of unique IDs in a short period.



Key Features (3/3)

Data Movement (Shard Migration)

- When shards are added or removed, data is redistributed to new shards. This redistribution is done using consistent hashing, which minimizes data movement and ensures asynchronous processing to prevent service interruptions. Additionally, real-time updates on data movement can be monitored through a dashboard using WebSocket.

Implementation Details

- ShardService Class:
  - migrateUsersFromShard(int removedShardId): A method that migrates data from a removed shard to newly allocated shards.
  - migrateUser(User user, int oldShardId): The logic for moving an individual user's data. When a user's shard is removed, this method locates a new shard and saves the user's data there.
- WebSocket Integration:
  - sendShardUpdates(): Periodically sends real-time updates on data distribution and movement via WebSocket, allowing administrators to monitor the dashboard.

Troubleshooting (1/2)

Static Database Management Issues

During testing, repeated difficulties were encountered in registering and configuring data for each shard. Adding or removing shards required manual registration, which led to inconsistencies and prolonged test preparation times.

Symptoms

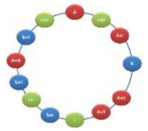
- Data registration was required for each database (shard) during every test.
- The more databases (shards) there were, the more time-consuming the manual setup became, leading to frequent errors during test environment setup.
- Manual configuration changes were needed each time a new shard was added or removed.

Cause Analysis

- Database settings were hard-coded, necessitating code modifications whenever a shard was changed, which was inefficient.
- The application lacked a feature for automatic shard registration or removal at runtime, making manual adjustments necessary for test environments.

Solutions

- Modified the system to use Spring’s @ConfigurationProperties to dynamically load shard information from a configuration file, allowing automatic registration and management of shards as defined in the configuration during application startup.
- Used the ShardDataSourceProperties class to dynamically load and manage all shards.



Troubleshooting (2/2)

Data Consistency Issues (During Shard Addition/Removal)

Issues were encountered when user data did not migrate correctly or data was lost during shard addition or removal, especially when handling large data volumes. This led to inconsistent data movement.

Symptoms

- Data disappearing instead of being transferred when a shard was removed.
- Existing data not being redistributed to new shards after a shard was added.
- Data inconsistencies, such as duplicated data across shards or data remaining unassigned.

Cause Analysis

- Inadequate transaction management during data movement, resulting in incomplete processing or failures that compromised data consistency.
- Attempting to move large amounts of data in real-time caused system load, leading to incomplete data migration.

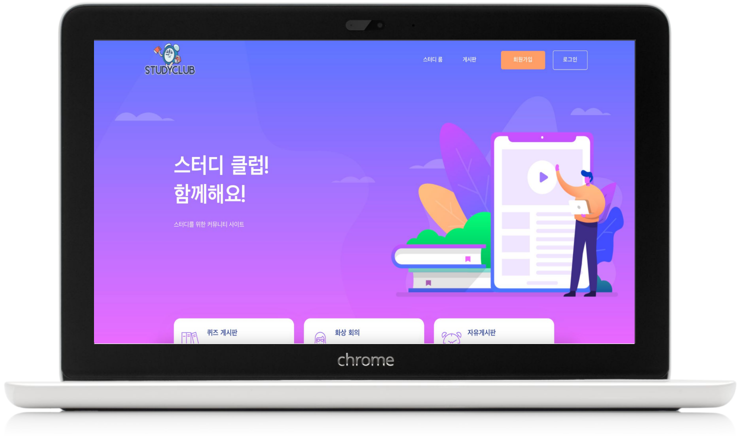
Solutions

- Used transactions to ensure that data migration was only marked as successful when fully completed on a shard, with a rollback mechanism for failed transactions to maintain data consistency.
- Designed the migration process to move user data sequentially instead of all at once, reducing system load and ensuring stable data migration.

STUDYCLUB

Web App for Study Recruitment and Activities

Development Period	2024.7 ~2024.8
Platform	Web
Team Size	4
Responsibilities	Implemented 1:1 video interviews using WebRTC, developed the quiz feature, and contributed to project planning and direction setting.



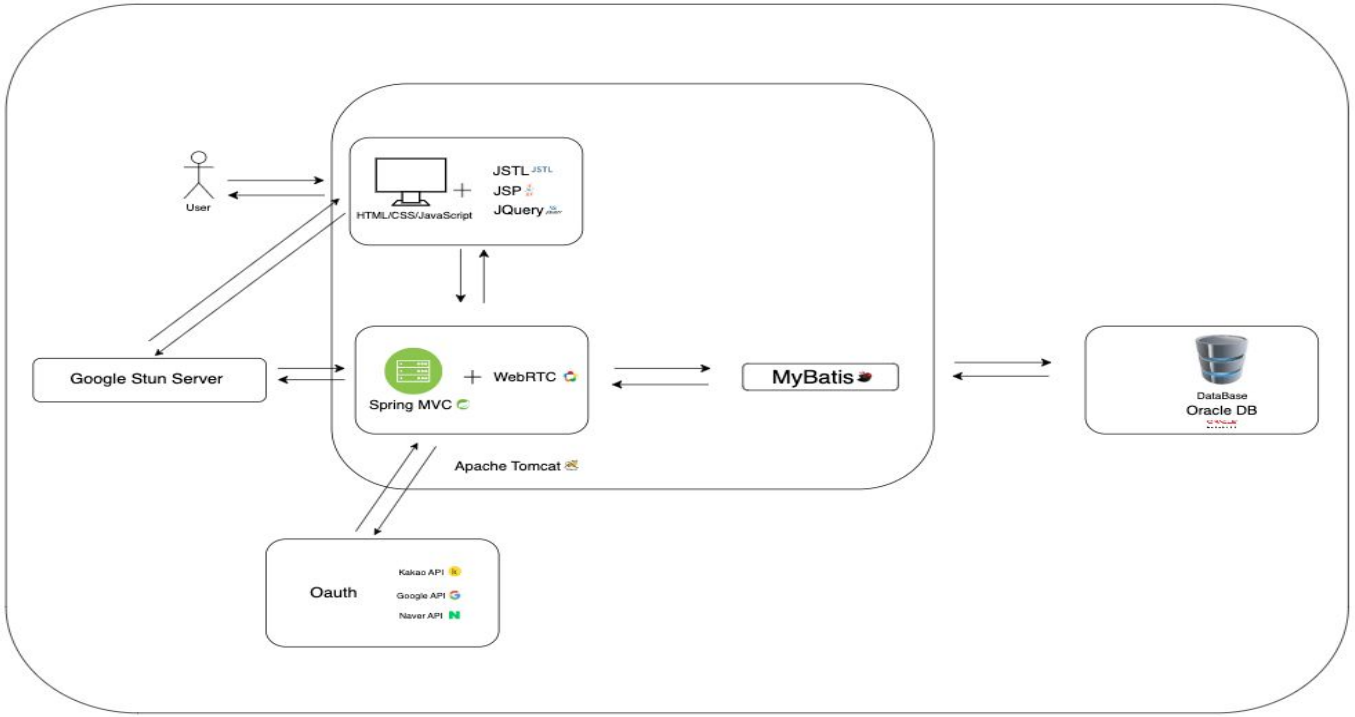
GitHub https://github.com/h3lim/StudyClub

Development Environment

Languages	Java(JDK 17), HTML/CSS, JavaScript
Server	Apache Tomcat 9.0, WebSocket, WebRTC
Frameworks	Spring MVC, Spring Boot, Spring Security, MyBatis 3.4.6
Database	Oracle DB
IDE	IntelliJ,SQL Developer
API, Libraries	KAKAO/NAVER/GOOGLE Login API, JQuery, WebRTC API, WebSocketAPI, Log4j, SLF4J,Validation

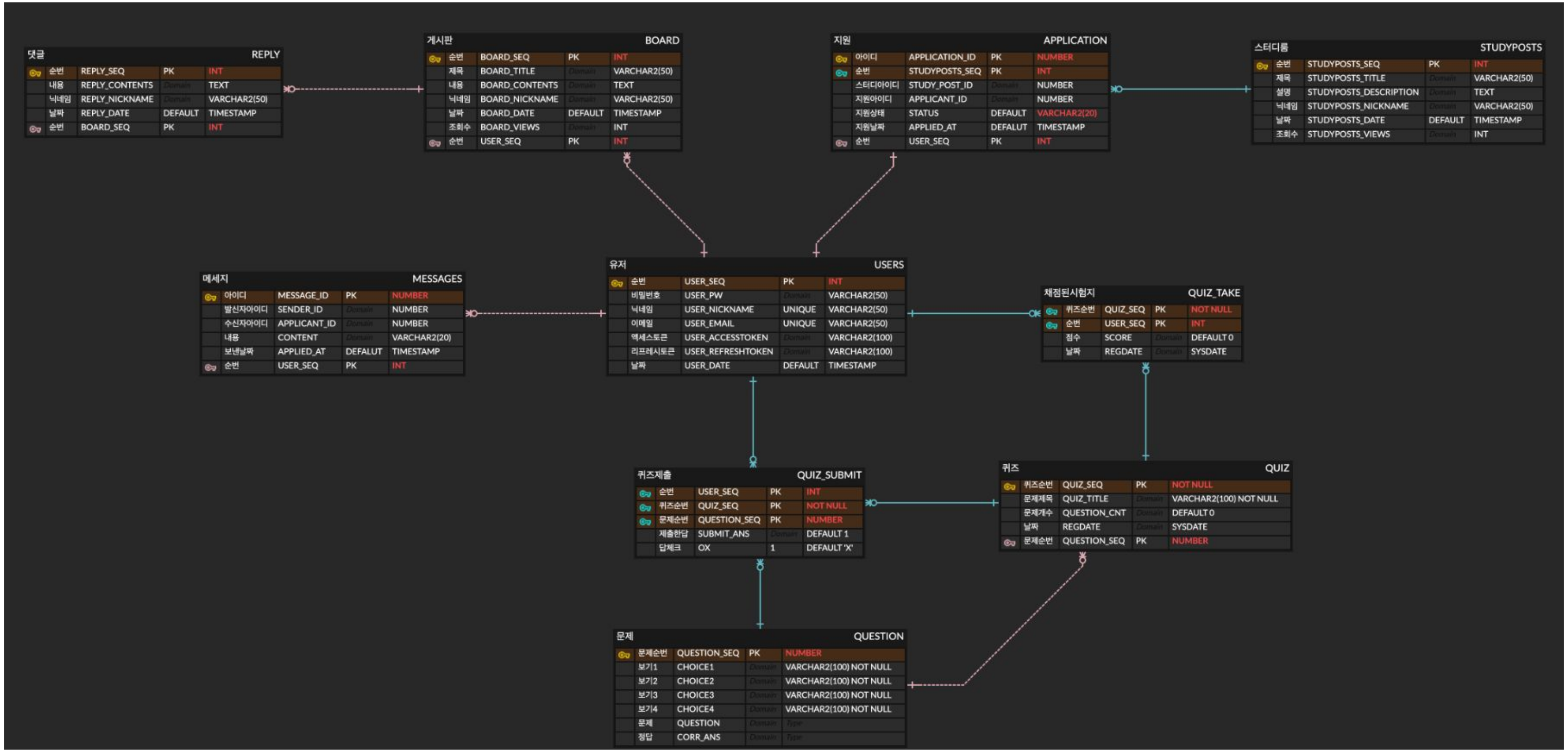
Project Planning and Direction (1/4)

System Architecture



Project Planning and Direction (2/4)

ERD Design



댓글			REPLY
순번	REPLY_SEQ	PK	INT
내용	REPLY_CONTENTS	Domain	TEXT
닉네임	REPLY_NICKNAME	Domain	VARCHAR2(50)
날짜	REPLY_DATE	DEFAULT	TIMESTAMP
순번	BOARD_SEQ	PK	INT

게시판			BOARD
순번	BOARD_SEQ	PK	INT
제목	BOARD_TITLE	Domain	VARCHAR2(50)
내용	BOARD_CONTENTS	Domain	TEXT
닉네임	BOARD_NICKNAME	Domain	VARCHAR2(50)
날짜	BOARD_DATE	DEFAULT	TIMESTAMP
조회수	BOARD_VIEWS	Domain	INT
순번	USER_SEQ	PK	INT

지원			APPLICATION
아이디	APPLICATION_ID	PK	NUMBER
순번	STUDYPOSTS_SEQ	PK	INT
스터디아이디	STUDY_POST_ID	Domain	NUMBER
지원아이디	APPLICANT_ID	Domain	NUMBER
지원상태	STATUS	DEFAULT	VARCHAR2(20)
지원날짜	APPLIED_AT	DEFALUT	TIMESTAMP
순번	USER_SEQ	PK	INT

스터디룸			STUDYPOSTS
순번	STUDYPOSTS_SEQ	PK	INT
제목	STUDYPOSTS_TITLE	Domain	VARCHAR2(50)
설명	STUDYPOSTS_DESCRIPTION	Domain	TEXT
닉네임	STUDYPOSTS_NICKNAME	Domain	VARCHAR2(50)
날짜	STUDYPOSTS_DATE	DEFAULT	TIMESTAMP
조회수	STUDYPOSTS_VIEWS	Domain	INT

메세지			MESSAGES
아이디	MESSAGE_ID	PK	NUMBER
발신자아이디	SENDER_ID	Domain	NUMBER
수신자아이디	APPLICANT_ID	Domain	NUMBER
내용	CONTENT	Domain	VARCHAR2(20)
보낸날짜	APPLIED_AT	DEFALUT	TIMESTAMP
순번	USER_SEQ	PK	INT

유저			USERS
순번	USER_SEQ	PK	INT
비밀번호	USER_PW	Domain	VARCHAR2(50)
닉네임	USER_NICKNAME	UNIQUE	VARCHAR2(50)
이메일	USER_EMAIL	UNIQUE	VARCHAR2(50)
액세스토큰	USER_ACCESSTOKEN	Domain	VARCHAR2(100)
리프레시토큰	USER_REFRESHTOKEN	Domain	VARCHAR2(100)
날짜	USER_DATE	DEFAULT	TIMESTAMP

채점된시험지			QUIZ_TAKE
퀴즈순번	QUIZ_SEQ	PK	NOT NULL
순번	USER_SEQ	PK	INT
점수	SCORE	Domain	DEFAULT 0
날짜	REGDATE	Domain	SYSDATE

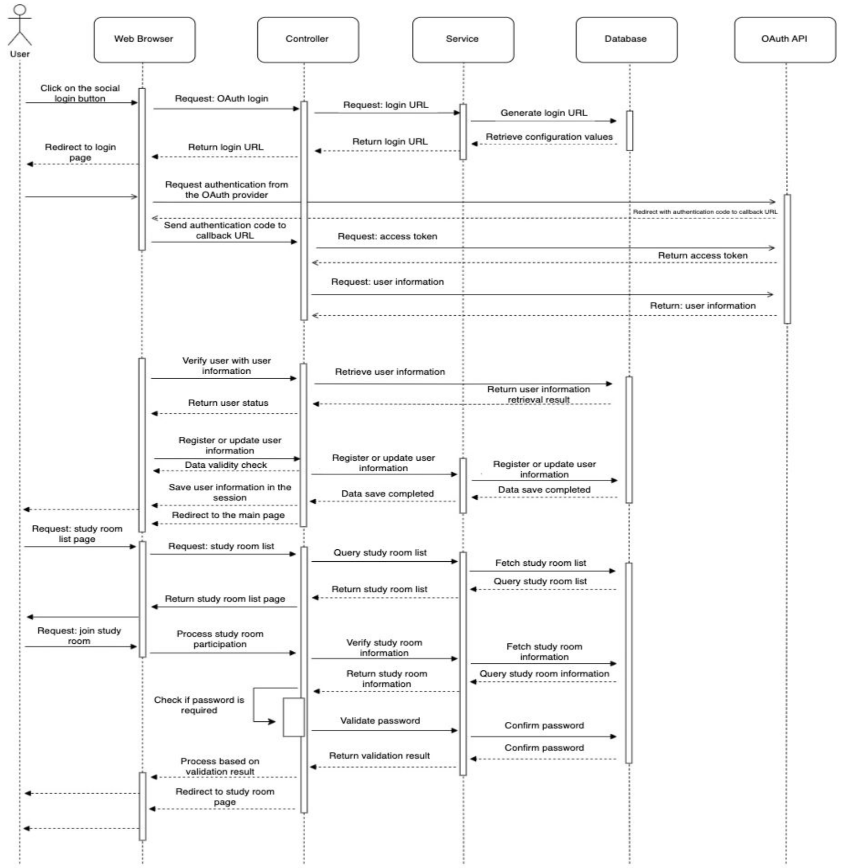
퀴즈제출			QUIZ_SUBMIT
순번	USER_SEQ	PK	INT
퀴즈순번	QUIZ_SEQ	PK	NOT NULL
문제순번	QUESTION_SEQ	PK	NUMBER
제출한답	SUBMIT_ANS	Domain	DEFAULT 1
답체크	OX	1	DEFAULT 'X'

퀴즈			QUIZ
퀴즈순번	QUIZ_SEQ	PK	NOT NULL
문제제목	QUIZ_TITLE	Domain	VARCHAR2(100) NOT NULL
문제개수	QUESTION_CNT	Domain	DEFAULT 0
날짜	REGDATE	Domain	SYSDATE
문제순번	QUESTION_SEQ	PK	NUMBER

문제			QUESTION
문제순번	QUESTION_SEQ	PK	NUMBER
보기1	CHOICE1	Domain	VARCHAR2(100) NOT NULL
보기2	CHOICE2	Domain	VARCHAR2(100) NOT NULL
보기3	CHOICE3	Domain	VARCHAR2(100) NOT NULL
보기4	CHOICE4	Domain	VARCHAR2(100) NOT NULL
문제	QUESTION	Domain	Type
정답	CORR_ANS	Domain	Type

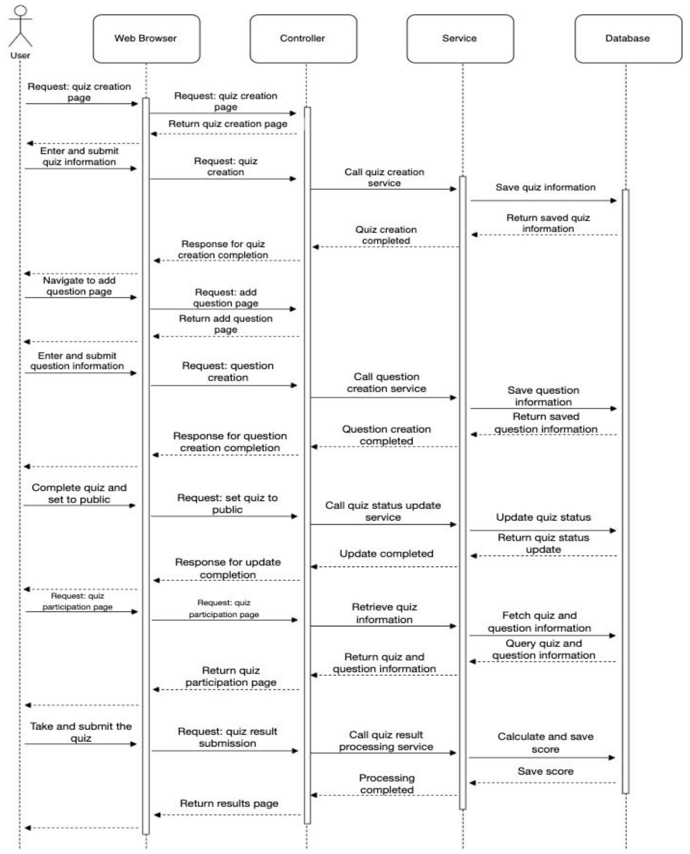
Project Planning and Direction (3/4)

Sequence Diagram – OAuth Login and Study Room Participation


User
Web Browser
Controller
Service
Database
OAuth API
Click on the social login button
Request: OAuth login
Request: login URL
Generate login URL
Retrieve configuration values
Return login URL
Return login URL
Redirect to login page
Request authentication from the OAuth provider
Redirect with authentication code to callback URL
Send authentication code to callback URL
Request: access token
Return access token
Request: user information
Return: user information
Verify user with user information
Retrieve user information
Return user information retrieval result
Return user status
Register or update user information
Data validity check
Register or update user information
Register or update user information
Save user information in the session
Data save completed
Data save completed
Redirect to the main page
Request: study room list page
Request: study room list
Query study room list
Fetch study room list
Query study room list
Return study room list
Return study room list page
Request: join study room
Process study room participation
Verify study room information
Fetch study room information
Query study room information
Return study room information
Check if password is required
Validate password
Confirm password
Confirm password
Return validation result
Process based on validation result
Redirect to study room page

Project Planning and Direction (4/4)

Sequence Diagram – Quiz Creation and Participation



User
Web Browser
Controller
Service
Database
Request: quiz creation page
Request: quiz creation page
Return quiz creation page
Enter and submit quiz information
Request: quiz creation
Call quiz creation service
Save quiz information
Return saved quiz information
Quiz creation completed
Response for quiz creation completion
Navigate to add question page
Request: add question page
Return add question page
Enter and submit question information
Request: question creation
Call question creation service
Save question information
Return saved question information
Question creation completed
Response for question creation completion
Complete quiz and set to public
Request: set quiz to public
Call quiz status update service
Update quiz status
Return quiz status update
Update completed
Response for update completion
Request: quiz participation page
Request: quiz participation page
Retrieve quiz information
Fetch quiz and question information
Query quiz and question information
Return quiz and question information
Return quiz participation page
Take and submit the quiz
Request: quiz result submission
Call quiz result processing service
Calculate and save score
Save score
Processing completed
Return results page

Real-Time Video Conference Functionality Using WebRTC

- In the STUDYCLUB project, WebRTC technology was used to enable real-time video meetings within study rooms, allowing remote users to communicate and collaborate as if they were in the same physical space.

Implementation Details

- Signaling Server Implementation: CamController was implemented to handle the signaling process necessary for establishing WebRTC connections.
- Use of STUN Server: A public STUN server was utilized to support connections between clients behind NAT firewalls.
- Communication via WebSocket: The WebSocket protocol was used for real-time message exchange between clients and the server.
- Client-Side Implementation:
  - Used JavaScript's RTCPeerConnection API to establish peer-to-peer connections.
  - Captured and transmitted user media streams (audio and video) to the counterpart.
  - Communicated with the server for signaling data exchange via WebSocket.

Key Features (2/2)

Quiz Creation and Participation Feature

- STUDYCLUB provides a feature for users to create quizzes and allows other users to participate in them to review study content, promoting knowledge sharing and learning motivation among users.

Implementation Details

- Quiz Creation Process:
  - Users input quiz titles and descriptions to create quizzes.
  - Multiple questions can be added to the created quiz.
  - Each question includes information such as the question text, choices, and the correct answer.
- Quiz Participation Process:
  - Other users can select and attempt public quizzes.
  - After answering and submitting the quiz, the score is calculated and displayed.
- Database Design:
  - Quiz Table: Stores basic information about quizzes.
  - Question Table: Stores questions related to each quiz.
  - UserGrade Table: Stores user quiz participation results.

Challenges in ERD Design for Quiz Feature

Significant time was spent designing the ERD (Entity-Relationship Diagram) for the quiz feature.

Challenges

- **Relationship Between Quizzes and Questions**: Each quiz contains multiple questions, and each question belongs to a specific quiz. How should this be represented efficiently?
- **Storing User Participation Results**: How to store and manage user participation data (scores, submitted answers, etc.) effectively?

Solutions

- Quiz-Question Relationship: Created Quiz and Question tables, with quiz_id in the Question table as a foreign key referencing quiz_id in the Quiz table to establish a clear 1:N relationship.
- Storing User Participation Results: Created a UserGrade table to store user quiz participation results, including columns like userNickname, quizSeq, score, and submitDate, allowing tracking of which user took which quiz and the score received.

Results

- A well-defined database structure facilitated efficient data processing for the quiz feature.
- Systematic management of user participation results enabled learning performance tracking and feedback.

Problem with Missing await in Asynchronous WebRTC Operations

Issues arose when the send() function was called without await, causing asynchronous task execution to become disordered. For example, send() was executed before setLocalDescription completed in the createOffer() function, leading to the transmission of incomplete signals.

Symptoms

- Offer and answer transmissions failed to function properly, causing unstable or failed connections.
- Audio or video streams were not sent, or real-time communication delays occurred.

Cause Analysis

- Calling send() without await led to signal transmission before setLocalDescription completed, resulting in unexpected signals on the receiver's side.
- Signal transmission timing became disordered, leading to unstable connections and missing signals.

Solutions

- Added await to ensure send() was executed only after the completion of asynchronous tasks, maintaining proper order.
- Adjusted createOffer() to send signals with await send() after setLocalDescription completed, ensuring correct signals were received.

Video Transmission Issues During 1:1 Interviews – HTTPS and SSL Certificate

Problems occurred with the WebRTC-based video call functionality during 1:1 interviews in STUDYCLUB, where the counterpart's video was not transmitted or frequently froze.

Symptoms

- Counterpart's video was not displayed, and WebRTC connections were blocked in the browser.
- WebRTC failure logs appeared in the console.

Cause Analysis

- WebRTC functions only over secure (HTTPS) connections. Without an SSL certificate or with a misconfigured certificate, WebRTC communication fails. Modern browsers block WebRTC over HTTP, causing video transmission failures.

Solutions

- Obtained and registered an SSL certificate in Tomcat and configured server.xml to enable HTTPS.
- Set up automatic redirection from HTTP to HTTPS.
- Configured the browser to access the application over HTTPS, ensuring secure WebRTC connections.

Video Transmission Issues During 1:1 Interviews – Port Forwarding and Firewall Problem

Problems were encountered with WebRTC video calls during 1:1 interviews in STUDYCLUB, where the video transmission was interrupted or frequently stopped.

Symptoms

- Counterpart's video was not transmitted or was interrupted intermittently.
- P2P connections were blocked due to certain ports being restricted by firewalls or routers.

Cause Analysis

- WebRTC communication relies on specific ports. Without proper port forwarding or if firewall rules block these ports, P2P connections fail, and video transmission is interrupted.
- Issues occurred when port forwarding on Tomcat server and Oracle DB was not properly configured or relevant ports (HTTP: 8087, HTTPS: 8447, Oracle DB: 1521) were blocked by firewalls.

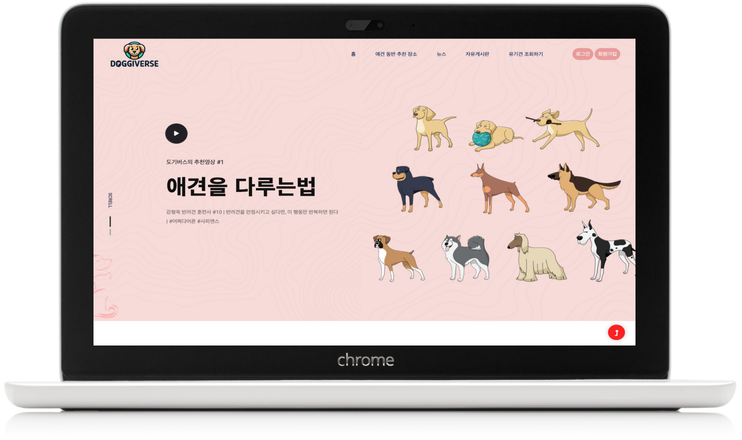
Solutions

- Allowed ports (8447, 8087, 1521) necessary for WebRTC communication in firewall and router settings.
- Verified firewall settings to ensure port allowances and added rules to open these ports.
- Added specific ports to inbound firewall rules and applied them to TCP.

DOGGIVERSE

Web App for Pet Owners: Content-based community service

Development Period	2024.5 ~2024.6
Platform	Web
Team Size	3
Responsibilities	Implemented user registration and My Page features, , and contributed to project planning and direction setting

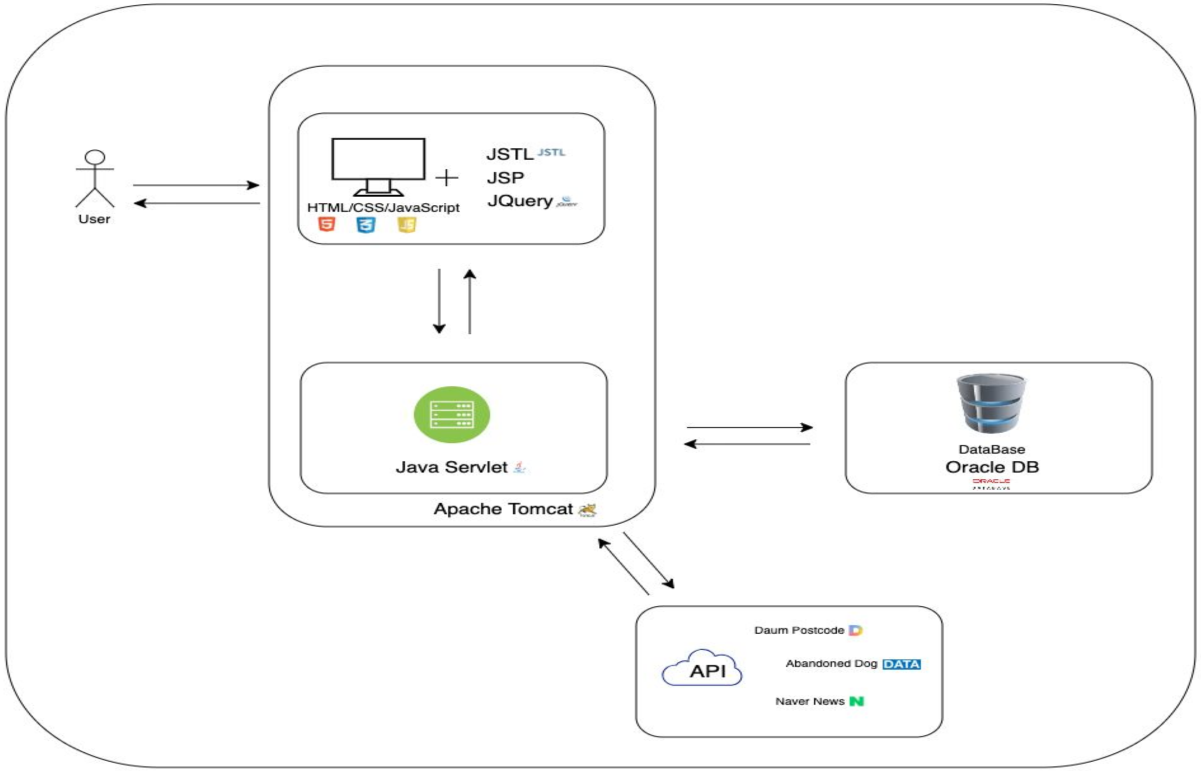


GitHub https://github.com/h3lim/Pet-Community-Web

Development Environment

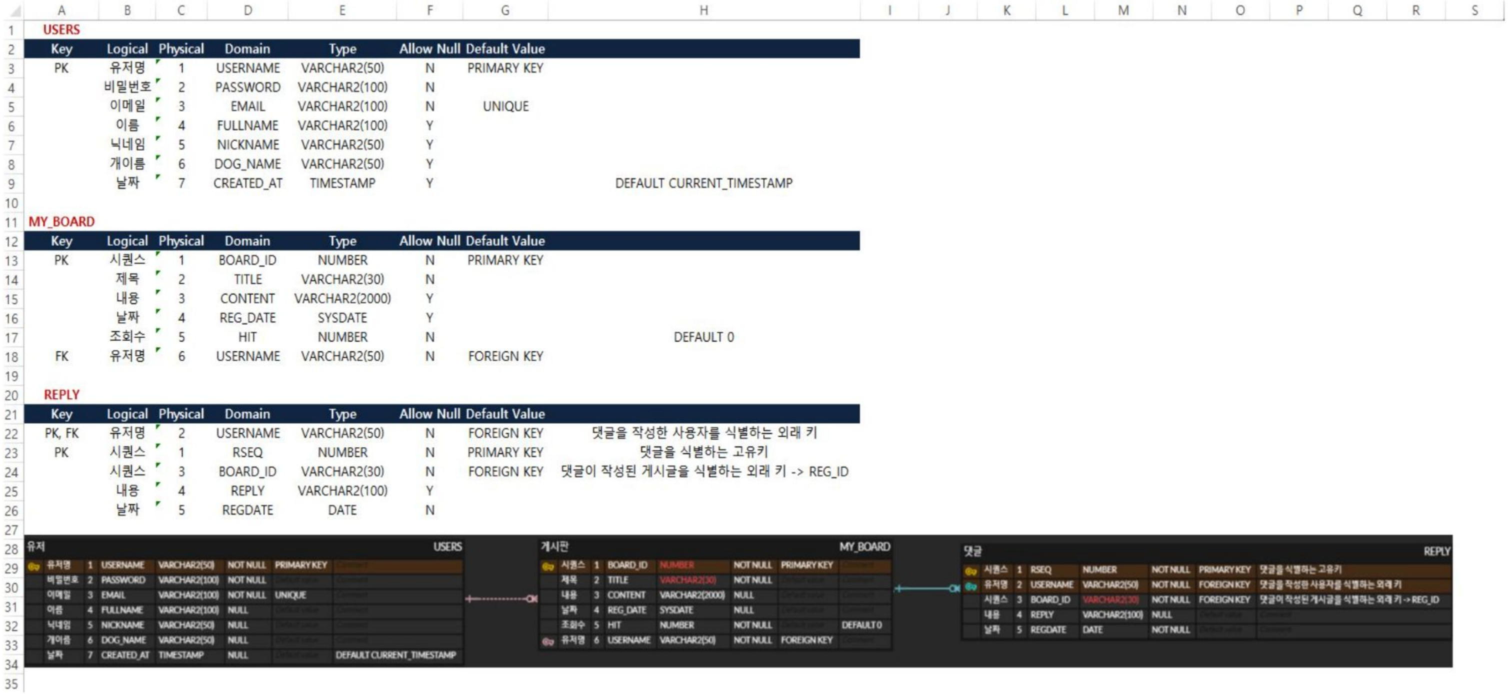
Languages	Java(JDK 17), HTML/CSS, JavaScript
Server	Apache Tomcat 9.0
Frameworks	Java Servlets & JSP
Database	Oracle DB
IDE	Eclipse,IntelliJ,SQL Developer
API, Libraries	Stray Dogs API, Daum Postcode API, Gson,JSoup, JSTL,JQuery

System Architecture



Project Planning and Direction (2/4)

ERD Design



USERS

Key	Logical	Physical	Domain	Type	Allow Null	Default Value	
PK	유저명	1	USERNAME	VARCHAR2(50)	N	PRIMARY KEY	
	비밀번호	2	PASSWORD	VARCHAR2(100)	N		
	이메일	3	EMAIL	VARCHAR2(100)	N	UNIQUE	
	이름	4	FULLNAME	VARCHAR2(100)	Y		
	닉네임	5	NICKNAME	VARCHAR2(50)	Y		
	개이름	6	DOG_NAME	VARCHAR2(50)	Y		
	날짜	7	CREATED_AT	TIMESTAMP	Y		DEFAULT CURRENT_TIMESTAMP

MY_BOARD

Key	Logical	Physical	Domain	Type	Allow Null	Default Value	
PK	시퀀스	1	BOARD_ID	NUMBER	N	PRIMARY KEY	
	제목	2	TITLE	VARCHAR2(30)	N		
	내용	3	CONTENT	VARCHAR2(2000)	Y		
	날짜	4	REG_DATE	SYSDATE	Y		
	조회수	5	HIT	NUMBER	N		DEFAULT 0
FK	유저명	6	USERNAME	VARCHAR2(50)	N	FOREIGN KEY	

REPLY

Key	Logical	Physical	Domain	Type	Allow Null	Default Value	
PK, FK	유저명	2	USERNAME	VARCHAR2(50)	N	FOREIGN KEY	댓글을 작성한 사용자를 식별하는 외래 키
PK	시퀀스	1	RSEQ	NUMBER	N	PRIMARY KEY	댓글을 식별하는 고유키
	시퀀스	3	BOARD_ID	VARCHAR2(30)	N	FOREIGN KEY	댓글이 작성된 게시글을 식별하는 외래 키 -> REG_ID
	내용	4	REPLY	VARCHAR2(100)	Y		
	날짜	5	REGDATE	DATE	N		

유저 USERS

유저명	1	USERNAME	VARCHAR2(50)	NOT NULL	PRIMARY KEY	
비밀번호	2	PASSWORD	VARCHAR2(100)	NOT NULL		
이메일	3	EMAIL	VARCHAR2(100)	NOT NULL	UNIQUE	
이름	4	FULLNAME	VARCHAR2(100)	NULL		
닉네임	5	NICKNAME	VARCHAR2(50)	NULL		
개이름	6	DOG_NAME	VARCHAR2(50)	NULL		
날짜	7	CREATED_AT	TIMESTAMP	NULL		DEFAULT CURRENT_TIMESTAMP

게시판 MY_BOARD

시퀀스	1	BOARD_ID	NUMBER	NOT NULL	PRIMARY KEY	
제목	2	TITLE	VARCHAR2(30)	NOT NULL		
내용	3	CONTENT	VARCHAR2(2000)	NULL		
날짜	4	REG_DATE	SYSDATE	NULL		
조회수	5	HIT	NUMBER	NOT NULL		DEFAULT 0
유저명	6	USERNAME	VARCHAR2(50)	NOT NULL	FOREIGN KEY	

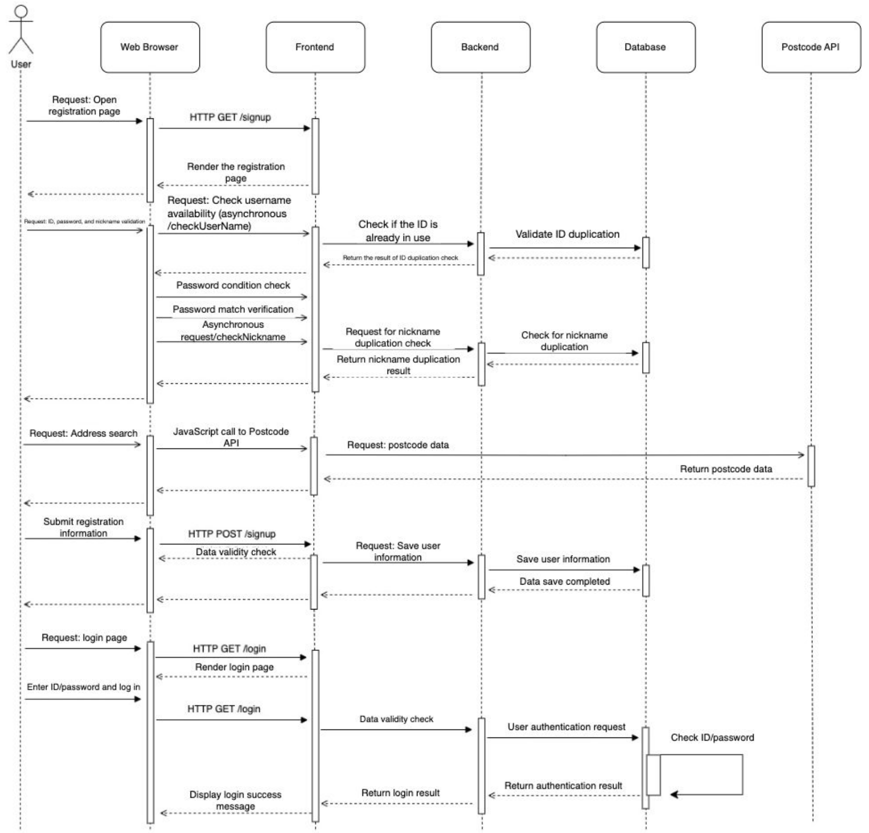
댓글 REPLY

시퀀스	1	RSEQ	NUMBER	NOT NULL	PRIMARY KEY	댓글을 식별하는 고유키
유저명	2	USERNAME	VARCHAR2(50)	NOT NULL	FOREIGN KEY	댓글을 작성한 사용자를 식별하는 외래 키
시퀀스	3	BOARD_ID	VARCHAR2(30)	NOT NULL	FOREIGN KEY	댓글이 작성된 게시글을 식별하는 외래 키 -> REG_ID
내용	4	REPLY	VARCHAR2(100)	NULL		
날짜	5	REGDATE	DATE	NOT NULL		

DOGGIVERSE

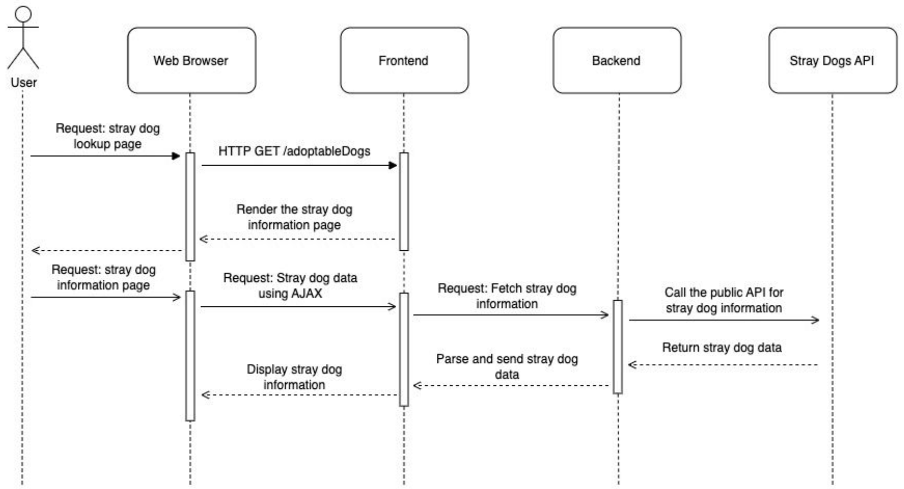
Project Planning and Direction (3/4)

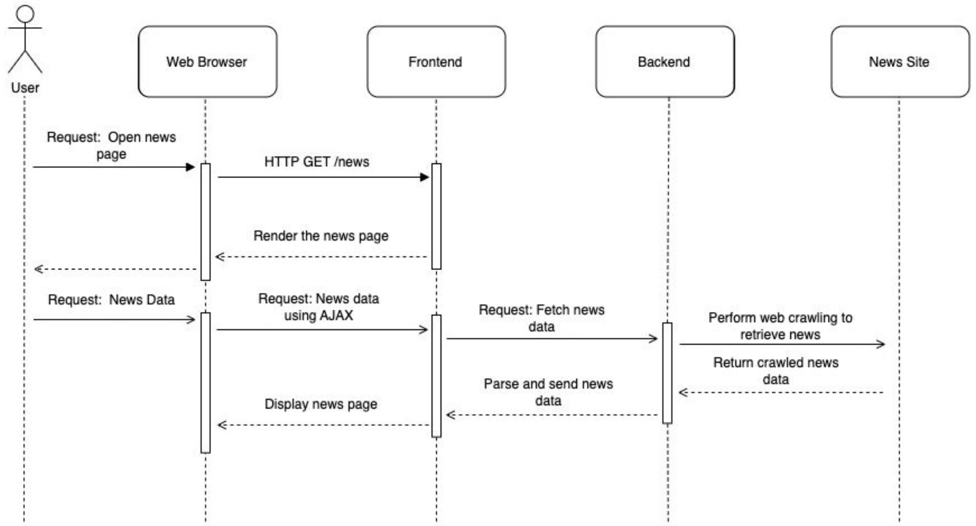
Sequence Diagram - User Registration and Login


User
Web Browser
Frontend
Backend
Database
Postcode API
Request: Open registration page
HTTP GET /signup
Render the registration page
Request: ID, password, and nickname validation
Request: Check username availability (asynchronous /checkUserName)
Check if the ID is already in use
Validate ID duplication
Return the result of ID duplication check
Password condition check
Password match verification
Asynchronous request/checkNickname
Request for nickname duplication check
Check for nickname duplication
Return nickname duplication result
Request: Address search
JavaScript call to Postcode API
Request: postcode data
Return postcode data
Submit registration information
HTTP POST /signup
Data validity check
Request: Save user information
Save user information
Data save completed
Request: login page
HTTP GET /login
Render login page
Enter ID/password and log in
HTTP GET /login
Data validity check
User authentication request
Check ID/password
Return authentication result
Return login result
Display login success message

DOGGIVERSE

Sequence Diagram - Stray Dog Lookup, Pet-Related News Lookup


User
Web Browser
Frontend
Backend
Stray Dogs API
Request: stray dog lookup page
HTTP GET /adoptableDogs
Render the stray dog information page
Request: stray dog information page
Request: Stray dog data using AJAX
Request: Fetch stray dog information
Call the public API for stray dog information
Return stray dog data
Parse and send stray dog data
Display stray dog information



Key features (1/2)

News Crawling Feature

- The system provides the latest pet-related news to users by crawling data from Naver News. To achieve this, the JSoup library is used to parse web pages and extract necessary information.

Implementation Details

- The NaverNewsCrawl class uses JSoup to fetch the HTML code of the Naver News page.
- Desired elements (e.g., article title, link, image, summary) are extracted using CSS selectors from the retrieved HTML document.
- Extracted data is stored in NewsVO objects and managed as a list.
- The CrawlServlet handles client requests for news and returns data in JSON format using Gson.
- The news.jsp page uses AJAX to fetch data from the servlet and dynamically display the news content.

Key features (2/2)

Stray Dog Information Lookup Feature

- The system provides real-time stray animal information from the Public Data Portal to users. The DogDataFetcher class is used to communicate with the external public API to collect data and present it to users..jsp page uses AJAX to fetch data from the servlet and dynamically display the news content.

Implementation Details

- The DogDataFetcher class retrieves stray animal information from the Public Data Portal via HTTP requests.
- The service key and request parameters are included in the URL during the API call.
- The response XML data is converted to JSON, and JSONObject is used for parsing.
- Extracted data is stored in HashMap objects and managed as a list.
- The dog.jsp page allows users to input a start date and end date to query data and implements pagination for displaying the results.

API Rate Limit Issue with Naver News Crawling

When crawling data from Naver News search pages, there is a potential risk of hitting the server's maximum request rate, leading to rate-limit blocks. This is especially likely when multiple consecutive requests are made, which may cause the server to restrict or block the IP.

Symptoms

- After making several calls within a short time, data may fail to load, or crawling attempts may fail.
- Jsoup.connect() may return an IOException, or network block notifications may occur.

Cause Analysis

- Frequent requests may be interpreted as spam by the server.
- Insufficient delay between requests can result in exceeding the API rate limit.

Solutions

- Use Thread.sleep(REQUEST_DELAY); in the code to apply a 1-second delay between requests to mitigate rate-limit issues.
- Introduce caching to prevent redundant calls within a specific time frame, allowing frequently requested data to be reused and reducing API call frequency.

Response Delay and Intermittent Errors in Stray Dog API

Delays or intermittent errors may occur when calling the public API, resulting in failure to retrieve data.

Symptoms

- Specific error messages frequently appeared on the page during intermittent errors, leading to repeated error displays if unresolved.
- Long response times from the API resulted in prolonged page loading times, with some requests eventually failing due to timeouts.

Cause Analysis

- Public APIs may have variable server status and response times based on usage, with potential abnormal states.

Solutions

- Implement retry logic with specified MAX_RETRIES and RETRY_DELAY_MS values to perform re-requests after a delay in case of a failure.
- Limit retry attempts to 3 to avoid excessive requests and set a delay time of 2 seconds to prevent server overload.