CircleSpace : University Club Archive for All Students

SeokHyun Bae[1], Jiwoo Chung[2], Guho Kim[2], and Jinu Park[3]

[1] Department of Computer Science and Engineering
[2] Department of Natural Science
[3] College of Engineering

Abstract. This report provides an in-depth exploration of the development process, design rationale, and challenges encountered while building CircleSpace, a university club platform. This platform was designed to centralize club-related information, ensuring ease of access and streamlined interactions for users. By leveraging advanced web crawling, real-time updates, and intuitive UI/UX design principles, CircleSpace addresses the core needs of university students seeking club-related information and facilitates a comprehensive and efficient browsing experience.

1 Introduction

University clubs play a vital role in fostering student engagement and personal growth, but students often face challenges in finding comprehensive and reliable information about these clubs. Existing methods such as "Everytime," official university websites, and "CampusPick" lack systematic organization, leading to fragmented and inefficient processes for students searching for relevant club details. To address these issues, we developed CircleSpace, an integrated platform designed to revolutionize how students discover and join university clubs. CircleSpace categorizes clubs systematically by type and field of activity, enhancing navigation and accessibility. It standardizes essential club details, providing clear information about membership fees, recruitment schedules, and contact methods. The platform offers real-time updates by integrating with official club websites and social media platforms, ensuring the latest news and announcements are always available. To build trust and credibility, CircleSpace includes a secure login system using university accounts and features personalized tools like favorite club notifications and a review system to help students make informed decisions.

Our platform leverages Selenium for web crawling and AWS-based deployment to ensure stability and scalability while employing advanced data retrieval techniques to deliver accurate and timely updates. By combining real-time updates, user-friendly design, and sophisticated web crawling techniques, CircleSpace transforms fragmented club information into an organized, accessible, and interactive experience. This approach not only simplifies the process of finding clubs but also deepens engagement within student communities, making university life more connected and enriching.

CircleSpace provides users with a comprehensive club information system by integrating and categorizing all club data by type and affiliated university. This allows students to quickly find clubs that interest them and helps address information imbalances between club administrators, current members, and prospective applicants. Through daily web crawling, the platform ensures users always have access to the most up-to-date club details. Additionally, CircleSpace integrates various club management processes, such as viewing club information, applying for membership, and managing members, into a single service. This offers club administrators a more streamlined and efficient management experience while enhancing accessibility and engagement for students.

2 Design for the Proposed Service

2.1 Overall Architecture

The CircleSpace platform is built on a full-stack architecture designed to provide high performance, scalability, and an intuitive user experience. The frontend was designed using the Vite-based React framework to improve build performance and enhance SEO for future website operations. Various libraries within the React framework were utilized to implement service logic, with a particular focus on properly managing authentication tokens stored as cookies in the browser. Our platform relies on real-time data reception and efficient HTTP communication with the database. To achieve this, we utilized the Axios library and designed a separate AxiosInstance to handle token-based requests effectively.

Providing a user-friendly environment was a top priority for our platform. Figma was extensively used for UI/UX design, and DevMode was employed to assist in writing CSS. Beyond design, Figma significantly contributed to team collaboration. Given our project follows a Single Page Application structure, we focused on optimizing website loading speed by minimizing image file sizes and improving image quality during the creation of design assets.

The backend is powered by Spring Boot, which seamlessly integrates with the frontend and provides a robust foundation for enforcing business logic and handling API operations. We actively utilized Spring Boot's annotation-based libraries. Since we were not highly familiar with SQL queries, we relied on Spring's JPA methods for most operations, except for cases where JPQL was necessary. Authentication and security are ensured through a JWT-based login and role management system, safeguarding user data and providing secure access.

The database, hosted on AWS RDS using MySQL, is structured to efficiently handle large volumes of user queries while maintaining data consistency. To ensure fast response times for data requests, we designed the system to use DTOs to provide the most efficient responses for all queries.

For web crawling, Selenium played a pivotal role, enabling the platform to retrieve and update real-time promotional posts from external sources like Everytime. This aspect is further elaborated in the "General Challenges" section.

For deployment, the platform leverages AWS EC2 instances running Ubuntu Linux. This approach provides precise control over dependencies, such as the

installation of a Chrome browser on the Linux instances to support web crawling operations.

2.2 General Challenges

To implement one of the most critical features of our platform—real-time information retrieval—web crawling became a key challenge. During this process, Selenium was required to automatically launch a Chrome browser and successfully log in to Everytime during its execution. To achieve this, two components were essential: the Everytime authentication token and a pre-installed Chrome browser. In the early stages of development, we initially planned to deploy using AWS Elastic Beanstalk and tailored our plans accordingly. However, meeting Selenium's crawling requirements posed a problem, as Elastic Beanstalk's method of dynamically creating and terminating instances was not suitable. As a result, we had to completely revise our plans and switch to using fixed instances on AWS EC2. Fortunately, we were able to successfully implement our goals, but the process of changing the deployment method introduced several challenges. These included manually issuing SSL certificates and configuring Nginx directly, which added unexpected variables to the deployment process and proved to be quite demanding.

2.3 Reasoning

The selection of technologies and architectural design was carefully considered to meet the platform's functional and performance requirements. React and Spring Boot were chosen for their scalability, reliability, and strong community support, making them ideal for building enterprise-grade applications. Selenium's ability to handle dynamic web content made it the optimal choice for web crawling, while AWS EC2 with Ubuntu Linux provided a stable and customizable deployment environment essential for browser-based operations. The architecture was designed to balance maintainability with user experience, ensuring that students could efficiently access comprehensive and reliable club information.

3 Implementation

3.1 Main Page

When users access the website, they can quickly navigate to various categories, communities, and the Daily Board through the navigation bar at the top of the main page. On the top-right corner, buttons allow users to proceed to the login or sign-up screens. Below the banner, the "Recommended Clubs" section displays clubs based on the user's interest categories set during registration, sorted by the latest additions. If the user is logged in, the server makes an additional request based on the user's token information stored as a cookie in the browser to retrieve data for the user's interest categories. If the user is not logged in, the section simply shows the clubs with the highest views.

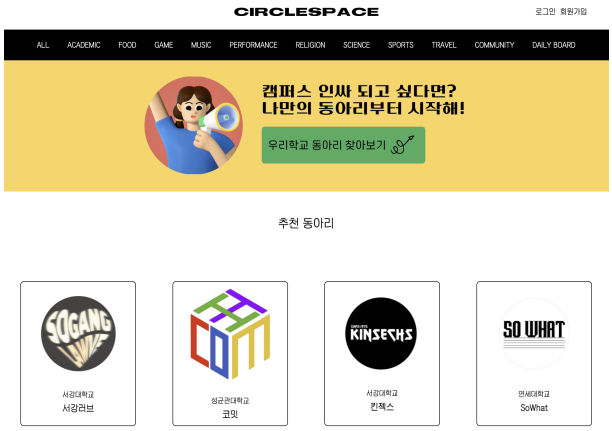

Fig. 1. Main Page

3.2 Club Features

The club browsing feature prioritizes making club information visually accessible and intuitively sorted. Considering this objective, controllers were designed to respond to various requests according to the user's needs. On the backend, response methods were implemented to handle all possible combinations of university affiliation, category, and sorting criteria (e.g., latest, most viewed). JPA played a critical role in this process, with features such as pagination, Specification, and Derived Query Methods being utilized extensively. Additionally, fields specific to sorting criteria were added to the Club, Member, and University entities, and careful consideration was given to structuring these fields using appropriate data structures like List and Set. Each entity was mapped correctly using Spring's mapping annotations. This approach allowed most of the database logic to be implemented entirely in Java code without the need for raw SQL queries.

On the club detail page, users can click on a club's icon to access detailed information, including key details such as recruitment schedules, membership fees, target audience, main activities, and contact information. Reviews written by current members of the club are also displayed. Users can mark a club as a favorite, enabling quick access via their favorites list, and they can apply to join the club by clicking the "Apply" button. During the application process, users submit a short self-introduction, which is made available for the club administrator to review on their My Page. On the backend, a Membership entity is created, containing essential details about the applicant, including their submitted self-introduction. Club managers can review applications and either approve or reject them during the Club Management process.

At the bottom of the detail page, a review section displays reviews submitted by current members. Reviews may include images and can only be deleted by

the reviewer or a site administrator. When a review is submitted, an entity is created in the database, which is mapped to both the Club and User entities.

Fig. 2. Club Application

Fig. 3. Club Detail Page

Fig. 4. Review Section

3.3 Member Features

On the register page, users register by entering personal information. To ensure that each person creates only one account, phone number and email verification have been implemented. Users are required to provide their university email address, which the platform uses to identify their affiliated university. This affiliation is later utilized for features such as the "View clubs from my university only" option on the main page and within categorized club listings. For phone number verification, we utilized an external service called "Nurigo," integrating its API key into our code to send authentication codes through a pre-configured message format and sender number. Email verification was implemented seamlessly using Spring Boot. By registering the sender's email address and secret key,

a controller inheriting from Java Mail Sender was used to send authentication codes. On the backend, verified email domains are matched to the corresponding university, which is then mapped automatically to the user entity through the University entity, where domain names are pre-configured for each university.

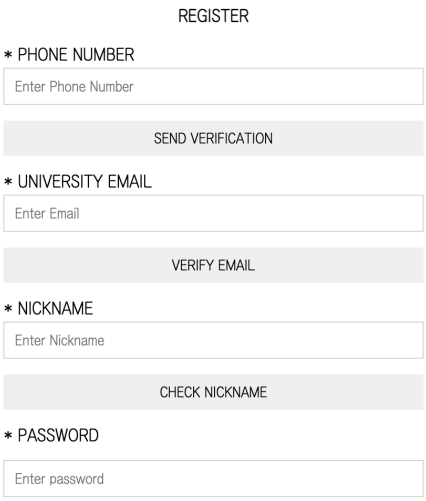

Fig. 5. Register Page

The first main feature on the My Page is the ability to register clubs. Users can provide essential information and set details such as the affiliated university and category. They can also upload a representative image and additional detailed images. Image uploads were implemented using a Presigned URL method, where the server generates a URL for image registration approval, and the client directly uploads the image to AWS S3 Cloud. This approach reduces the server's performance burden.

The second main feature is the ability to view and manage clubs the user has registered or joined. On the management page of a user-registered club, users can edit information, add detailed images, and review membership applications. They can approve or reject applicants. When an application is approved, the Approved field in the Membership entity is updated to True, officially making the user a club member. If an application is rejected, the Membership entity is deleted from the database.

3.4 Web Crawling Implementation (Daily Board)

By clicking the Daily Board button on the nav bar, users can view posts from the "Everytime" club board. These posts are synced with "Everytime" at 3 AM

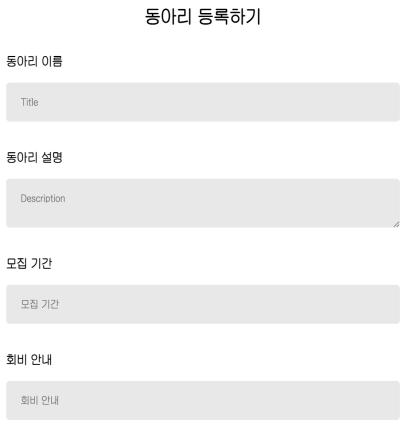

Fig. 6. Club Register Page

Fig. 7. Club Management

Fig. 8. Daily Board

every day. Through this feature, users can access information about clubs and academic societies that are not registered on the site.

3.4.1 Automate Retrieval of Club Promotion Posts Using Selenium

The retrieval of club promotion posts from Everytime was automated using Selenium to perform daily scraping tasks at 3 AM. The crawling logic was meticulously designed to navigate multiple pages and retrieve post content efficiently, minimizing resource usage and ensuring operational robustness.

3.4.2 WebDriver Initialization

To optimize performance and reduce detection by anti-bot mechanisms, ChromeDriver was configured in headless mode with tailored options. These included disabling GPU usage, managing window dimensions, and simulating legitimate user-agent headers. A mechanism to reuse authentication cookies was implemented, enabling stable login sessions and minimizing the need for repetitive manual login, thereby enhancing the efficiency of the crawling process.

3.4.3 Crawling Workflow

The crawling workflow was designed to handle dynamic web elements effectively. FluentWait was employed to accommodate variable loading times of page elements, ensuring reliable data extraction. For each page, the crawler parsed titles, timestamps, and body content of articles. The process terminated when previously crawled data was encountered, preventing redundant data collection.

3.4.4 Data Processing

Data processing included normalizing and filtering the retrieved content to address varying timestamp formats. A rigorous duplication check was conducted using title, body content, and posting date to avoid redundant storage. These measures ensured the integrity and efficiency of the database.

3.4.5 Security Measures

Several security measures were implemented to protect sensitive information. Authentication cookies were securely stored and managed within the EC2 instance, reducing the risk of exposing login credentials. Hardcoded credentials were avoided by using environment variables or secure vault systems to obfuscate sensitive information. Additionally, network security was enhanced by restricting EC2 instance access through security groups, permitting only essential inbound connections.

3.4.6 Optimization Strategies

Efforts were made to optimize the crawling process for performance and reliability. The use of headless ChromeDriver conserved system resources while maintaining operational efficiency. The navigation through page content was optimized by reducing unnecessary retries and implementing streamlined DOM interaction. Additionally, error handling mechanisms were integrated to manage unexpected conditions such as format changes or network interruptions gracefully.

3.4.7 Results and Impact

The implementation successfully automated the retrieval of promotional posts on a daily basis. Operational efficiency was significantly improved, reducing the need for manual intervention and ensuring timely updates of content. This system facilitated the seamless integration of real-time promotional content into the website, thereby enhancing user engagement and supporting the overall functionality of the platform.

3.5 Depolyment

3.5.1 Deployment Challenges with Elastic Beanstalk

Initially, deployment was attempted using Elastic Beanstalk due to its simplicity and automated resource management. However, challenges arose from Elastic Beanstalk's dynamic instance creation and termination, which disrupted the stability of the crawling environment. These disruptions made it difficult to persist ChromeDriver configurations and session cookies, which were essential for the automated web crawling tasks.

3.5.2 Transition to EC2 for Stability

To address these issues, the deployment strategy was transitioned to an EC2 instance setup, allowing for a more stable and controlled environment. By using EC2, it became possible to maintain a consistent crawling configuration, ensuring seamless operations and reducing the risk of interruptions due to environmental changes. This setup enabled precise control over the installation and management of dependencies, such as ChromeDriver, and facilitated the secure storage of authentication cookies.

3.5.3 Implementation of CI/CD Pipeline

To streamline updates and deployments, a CI/CD pipeline was implemented using GitHub Actions. This pipeline automated the build and deployment processes, significantly reducing manual effort and the risk of human error. The pipeline operates by first fetching the latest code from the deploy branch whenever a push occurs. It then sets up the required environments, including Java

(Temurin 21) and Node.js (v18), ensuring compatibility for both backend and frontend builds. Vulnerability checks are also automated using npm audit fix, which addresses potential security issues prior to the build phase.

3.5.4 Deployment Workflow

The application is then built using Gradle, producing a .jar file that is ready for deployment. This build artifact is securely transferred to the EC2 instance via SCP, and any previously running instances of the application are terminated using SSH commands. The deployment script also incorporates log management by archiving existing logs with timestamped filenames to facilitate monitoring and debugging. Once the deployment is complete, the new application instance is started with timezone settings configured for Asia/Seoul, ensuring accurate time logging.

4 Evaluation

To evaluate CircleSpace, we focused on demonstrating how the platform meets its primary objectives, including systematic categorization, real-time updates, and enhanced user engagement. Empirical results were derived from both functionality testing and user feedback to assess the effectiveness of the platform in addressing the challenges faced by university students in discovering and managing club-related activities.

4.1 Systematic Categorization

CircleSpace successfully organizes clubs based on type, university affiliation, and field of activity. This structured approach allows users to intuitively navigate and explore clubs that match their interests. The implementation of dynamic filtering and sorting options—such as sorting by the latest updates or most viewed—ensures that users can easily find relevant clubs. Backend controllers were carefully designed to handle various combinations of these criteria using JPA features like pagination and Derived Query Methods. These efforts resulted in a seamless and efficient browsing experience, effectively solving the problem of fragmented and unorganized information.

4.2 Real-Time Updates

One of CircleSpace's core features is the ability to provide up-to-date club information. By leveraging Selenium for daily web crawling, the platform retrieves and updates promotional posts from sources like "Everytime." These updates are synchronized at 3 AM daily, ensuring users have access to the latest announcements and activities. The backend efficiently processes retrieved data, ensuring accuracy and eliminating duplication, while the deployment on AWS EC2 ensures the stability of crawling operations. This real-time update mechanism enhances the platform's reliability and keeps users informed about ongoing club activities.

4.3 Enhanced User Experience

CircleSpace prioritizes user engagement through intuitive UI/UX design and personalized features. The recommendation system on the main page, which suggests clubs based on users' selected interest categories, creates a tailored browsing experience. For logged-in users, authentication tokens stored as browser cookies enable personalized data retrieval, such as recommendations for their specific university or interests. For non-logged-in users, the platform provides a general list of the most popular clubs, ensuring accessibility for all.

The streamlined application process further improves the user experience. On the club detail page, users can view comprehensive club information, mark clubs as favorites, and submit applications with a personalized self-introduction. Applications are then managed efficiently by club administrators using tools provided in the My Page feature. These features simplify the club joining process and foster stronger connections between students and clubs.

5 Limitations

Despite its many achievements, CircleSpace faced several limitations. These challenges, however, provide valuable insights for future iterations. The current implementation assumes the availability of Everytime authentication tokens to automate updates for university clubs. This reliance on external tokens limits the scalability of the platform, as it cannot actively support universities without such credentials. Addressing these issues may involve securing authentication tokens for a broader range of universities, implementing automated moderation systems for verifying club information, and designating representatives at multiple institutions to ensure platform scalability.

One significant limitation is the reliance on Everytime authentication tokens, which restricts the platform's ability to actively update club information for universities other than Sungkyunkwan University. This dependency emphasizes the need for broader integration with other university platforms. Additionally, the platform requires moderators to verify the authenticity and relevance of registered club information, which can create bottlenecks and pose challenges to scalability. Another critical limitation was the inability to designate representatives for other universities, making it difficult to expand the service. Addressing this issue will likely require decentralized management or the implementation of automated solutions to support broader service expansion.

6 Related Work

There is a well-known club browsing platform for Sungkyunkwan University called SKKLUB. However, CircleSpace distinguishes itself from SKKLUB in several ways. The most significant difference is that SKKLUB is a platform exclusively for Sungkyunkwan University. CircleSpace, on the other hand, can integrate with all universities as long as appropriate administrators are assigned

for each university. This also provides the major advantage of being able to manage numerous inter-university clubs. Secondly, SKKLUB does not have member features, which limits activities based on user information that CircleSpace supports. These include sorting and categorizing clubs according to the user's affiliated university, as well as applying to and registering for clubs. Additionally, the ability to post and share content on a community board is a feature unique to CircleSpace. The third difference is that CircleSpace retrieves information daily through web crawling from platforms like Everytime, where club-related information is available. This makes accessing the latest information extremely convenient, eliminating the need to visit multiple sites to stay updated.

7 Conclusion

Through CircleSpace, students were provided with an intuitive and centralized platform to easily explore, join, and engage with university clubs. The platform tackled common challenges in the club registration process by offering well-organized categorization and sorting options, allowing users to efficiently discover clubs that matched their interests. The streamlined application process removed unnecessary complexity, enabling students to join clubs effortlessly, while the management tools empowered club administrators to oversee operations more effectively.

Additionally, the platform's personalized features, such as real-time updates and tailored club recommendations based on user preferences, further enhanced the user experience. By integrating tools like web crawling and automated updates, CircleSpace ensured that users always had access to the most accurate and up-to-date information.

The system also promoted stronger connections between students and clubs by addressing information gaps and providing clear, consistent details about membership, recruitment, and events. These features not only simplified the process for users but also fostered a more vibrant and active university club ecosystem. The platform's innovative approach and practical solutions set a strong foundation for future enhancements and wider adoption across universities.

References

1. Selenium Documentation: https://www.selenium.dev/documentation/
2. React and Vite Documentation: https://vitejs.dev/
3. Spring Boot Documentation: https://spring.io/projects/spring-boot