Indian Journal of Economics and Business
Vol. 20 No. 2 (July-December, 2021)

http://www.ashwinanokha.com/IJEB.php

Scalable Building Management System for Offices and Co-Working Spaces

Nicholas Dwiarto Wirasbawa, M. Dzulfiqar Ramadhan Wibawanto, Albert Kosasi, & Seng Hansun

Faculty of Engineering and Informatics, Department of Informatics, Universitas Multimedia Nusantara, Indonesia

[*] *Corresponding author:* seng.hansun@lecturer.umn.ac.id

Received: 30th May 2021
Revised: 15th July 2021
Accepted: 30th July 2021

Abstract: Co-working spaces and offices are two of the most growing and essential businesses in Asian countries. Several factors contribute to the development of this business, such as the offered interiors, facilities, events, etc. However, the most important factor for the development of this business is the community itself, as it can generally result in a collaborative environment where one could collaborate or assist each other. In this research, we implement a highly scalable and flexible information system for offices and co-working spaces building management. This research also encompasses creating a friendly user interface that conforms to the Eight Golden Rules. Several cutting-edge technologies, such as Next.js, Chakra UI, Express.js, Mongo DB, Vercel, and Heroku, are used in the development phase of this system. The prototype built then is tested by using Technology Acceptance Model. We got a satisfactory result, where most respondents claimed that the system could be used in production phase due to its ease of use, performance, and simplicity to manage the business process. It is also proven that scalability will not be a problem when using this system because the technologies used are always ready to be scaled.

Keywords: Building Management System, co-working spaces, cutting-edge technologies, offices, Technology Acceptance Model.

1. Introduction

The co-working space community is one of the most important factors for the continuity of this new business model. People like co-working spaces that have a welcoming community and open communication. Other factors that make people attracted to the business model are facilities, such as the interior, prices, and the events that are held in the co-working space. From those factors, one of the most important factors to be considered when developing a co-working space is the community, and in extension, the communication process (Seo et al., 2017).

The existence of co-working spaces makes people feel connected with each other. It is possible to create a collaborative environment where one could collaborate or assist each other (Bouncken et al., 2018). With these things in mind, solidarity and camaraderie are formed well with each other. A co-working space can be a good place where one could find social support, extend their personal networks, and could even possibly form business partnerships, a win-win solution for both parties (Bianchi et al., 2018).

The co-working space is one of the emerging business models that have gained a lot of exposure in Asian countries, and the business itself has gained a lot of profits and advantages. The main focus of this business model is to build bonds with one another (Bouncken et al., 2016). Another factor that would predict whether the business model will prevail or not is the literal location of the co-working space. If it is close to the center of the city and supported by the right marketing tactics, there is a high possibility that the co-working space will be successful.

Nowadays, people interact with each other in the world of e-commerce. In the world of e-commerce, factors that influence the trust element of e-commerce are branding, customer's trust, privacy concerns, and security concerns. The future of e-commerce is depended on the reputation and the systems or technologies used. If one of them fails to keep up with the current trend, then e-commerce has a high possibility of losing to its competitors (Mittal, 2013).

In Indonesia, there is little to no previous researches about the implementation of a scalable building management system. Therefore, we are interested to develop a highly scalable and flexible building management information system for offices and co-working spaces in the Indonesian context and community. This research also encompasses creating a friendly user interface to create a simple personal brand. The definition of flexible in this study is that a building could consist of n floors, where n is the maximum value of an integer in 32-bit systems (2.147.483.647), and in each floor, there can be a maximum of n rooms. The system is built with both performance and scalability in mind, and the system focuses on these two factors. We have decided to implement a horizontally scalable system, which means the scalability process will be much easier rather than a vertically scalable system.

2. Method

We start the discussion of this section with the research methodologies adopted in this study. Furthermore, general directives, use case and database design, and the scalability aspect also briefly discuss here.

2.1 Research methodologies

In this research, we adopted one of the Agile Development methodologies, namely the Scrum method (Alsaqqa et al., 2020). It is a popular framework for product and software development (Gonçalves, 2018; Srivastava et al., 2017). It has been applied in various fields and projects as reported in Khalid et al. (Khalid et al., 2020) and Hidalgo (Hidalgo, 2019). Scrum releases the features in an incremental method (called a sprint) progressively towards the full release of a system or application. This way, people can easily change the requirements with minimum changes to the code and minimizes the amount of effort required for

changing the system. In each sprint, there are several steps used, such as requirements engineering, design, development, testing, deployment, and review.

In order to simplify the development phase, we also use a GitHub repository to keep track of issues, bugs, and errors. Currently, GitHub is the largest repository hosting service that provides an ecosystem for developers to work together (Sharma et al., 2017). Moreover, Heroku has also been used to deliver continuous integration and delivery of the system. Interested readers may find detailed information about Heroku on the official website (heroku.com, n.d.).

2.2 General directives

In this research, the system created will have the following important entities:

1) User/ customers. This entity is capable of doing several operations in the system, such as registering, logging in, seeing the list of all available rooms, ordering a room, editing their profile, and checking their transaction history.

2) Admin is the derivative of the 'user' entity. An admin can do everything that a user can do, plus s/he able to perform several operations that require admin privileges, such as seeing the list of orders, changing order status, accepting visitors, and performing several data modifications.

3) Owner is the derivative of the 'user' entity. This entity is capable of doing anything that a normal user and an admin could do, with the additional feature of being able to perform CRUD (Create-Read-Update-Delete) operations on essential components that a building have (e.g., floors, admin, rooms), and able to see the earnings of the company.

4) Floors and rooms are the main components of this system. They can be CRUD-ed by an owner. Rooms are divided into two types, namely the co-working space and the office.

2.3 Use case and database design

This system is built to conform with the following use case diagram, which can also be interpreted as the business process, as shown in Fig. 1.

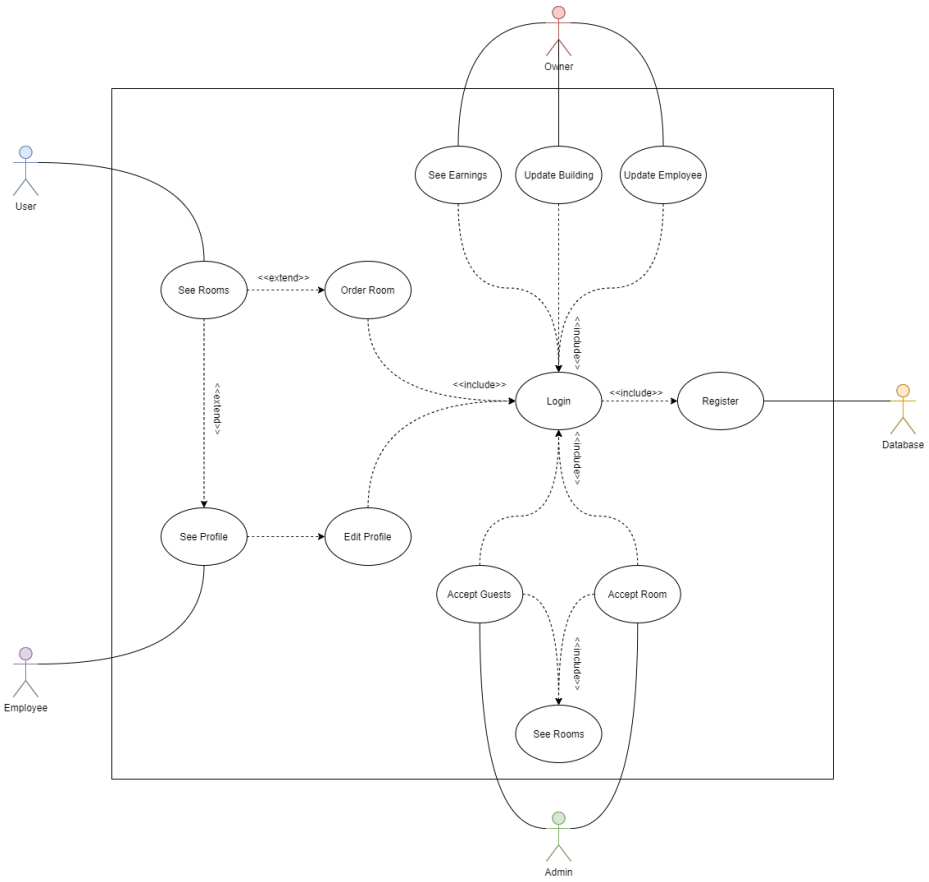

Figure 1:Use case diagram

The system is built on the MongoDB database, a document-based NoSQL. Considering this fact, the database has been modified or slightly denormalized to maximize the performance of the NoSQL database. The document schemas available for the database are:

• Users, where this particular schema will store the first name, last name, address, email, username, photo, password, role, activation status, and creation date.

• Employees, where this particular schema will store the salary, job description, join date, and the user that references to a user in the 'Users' schema.

• Floors, where this particular schema will store the floor number, name, the creation date, and the last modification date.

• Rooms, where this particular schema will store the name, description, room features, thumbnail, photos, price, type, slug (for SEO and URL-friendliness purposes), creation and modification date, and floor which references to a floor in the 'Floors' schema.

• Orders, where this particular schema will store the user which references to a user in the 'Users' schema, the room which references to a room in the 'Rooms' schema, the employee which references to an

employee in the 'Employees' schema, the start and end date, the order status, the total price, and the creation and the modification date.

• Visitors, where this particular schema will store the first name, last name, address, email, purpose, creation date, and a room id which references to a room in the 'Rooms' schema.

• Vouchers, where this particular schema will store the voucher code, the discount, and a Boolean value whether this voucher is valid or not.

2.4 Scalability

Because of the chosen technology stack and the programming paradigm, we believe that this system can be scaled horizontally. This means that the system can be scaled by simply buying more machines, instead of upgrading the current ones being used.

We use asynchronous programming to delegate processes into the thread pool so they can be executed separately from the main thread. Node.js is naturally asynchronous, so it is easy and even encouraged to implement the async-await pattern. Most of the methods in Node.js implement 'callback' functions or 'Promise-based callback' functions. In other words, the methods are executed asynchronously without blocking the main thread.

Using an API is essential as it provides our back-end to be completely separated from the front-end. Modifying one of them will not be difficult if we separate the concerns between the front-end and the back-end. Of course, this also helps with scalability issues as we do not need to change a lot of the source code if the separation of concerns is done well.

As the database used in this system is a document-based MongoDB, we believe that the database schema itself can be very flexible. For example, we can add another attribute to the schema without affecting other, already existing data, unlike the SQL-based databases.

3. Findings and Discussions

3.1 Technologies used

The implementation of this system is in the form of a web application, and it is done using several tools and technologies with MERN stack, such as:

• Next.js for the front-end;

• ChakraUI for the UI framework;

• Express.js for the back-end;

• MongoDB Atlas for the database;

- Git and GitHub for version control; and
- Heroku and Vercel for CI/CD

Basically, this system is maximizing the asynchronous, single-threaded part of JavaScript and Node.js to deliver high-performance, low latency responses to the client. The usage of MongoDB is to ensure that the data can be fetched with high-speed, low-query style. It is also used because of its flexibility with its database schemas.

We are using Next.js and ChakraUI to build high-performance, hybrid-rendered pages with a simple, but intuitive UI. According to Vercel, Next.js applications are production-ready and ready to scale. It is most often used for websites that require high traffic, such as Hulu, Netflix, GitHub, and many others. More information can be found on its official website (Vercel, 2021).

It is also proven that through empirical observations that most web pages coded with Next.js are optimized and therefore, much faster. Creating and using an API with Express.js is also much faster than using other languages (except for some compiled languages) due to its asynchronous and single-threaded, non-blocking nature.

3.2 Highlights and core features

This system is coded with several highlights as follows:

- Simple, colorful, but intuitive UI.
- Ability to create an infinite number of rooms and floors.
- Users can book a room, edit their profile, and see their transactions.
- Multiple roles support - user, admin, owner.
- Admins can perform CRUD operations on floors, rooms, employees.
- Admins can manage available orders from users directly.
- Admins can see the lifetime earnings of the system.
- Admins can create visitors via an interface in the admin panel.
- Admins can create vouchers via API endpoints.
- Dark mode support for the front-end.
- High-performance support with pre-loaded pages, image lazy-loading, and caching.
- Server-side rendering support for pages requiring authentication.
- API-proxy via Next.js serverless functions to provide extended security.

• Personal and secure authentication utilizing http Only or same Site cookies and JWT (stateless but secure).

• Accessibility support (a11y).

3.3Implementation results

Some system core features that conform to the scope of the research project are described in the following pages. Firstly, we developed the main page, complete with the Search Engine Optimization so that the Internet could crawl the website faster, resulting in faster indexing. The main page of the website is developed with UI/UX in mind, which conforms to Shneiderman's Eight Golden Rules (Aottiwerch & Kokaew, 2017), and is created to emphasize the 'user-friendliness' and 'branding' in this research project. Fig. 2 shows the Main page of the system.

The next part is the 'Rooms' page, where we display the data of all of the available rooms. We prepared pagination (per floor) to save memory. Each of the room cards could be clicked to access the corresponding room page directly. Fig. 3 shows the Rooms page of the system.

Each room on the 'Room' page can be clicked. The user can view all of the details of the room and can also order a room if they are logged in. If the user has successfully ordered a room, they can see their orders on their profile page. The user can only order if their order duration or time does not overlap or conflict with other people. This was done to prevent double orders at the same time, for the same room. Fig. 4 shows the Order page of the system.

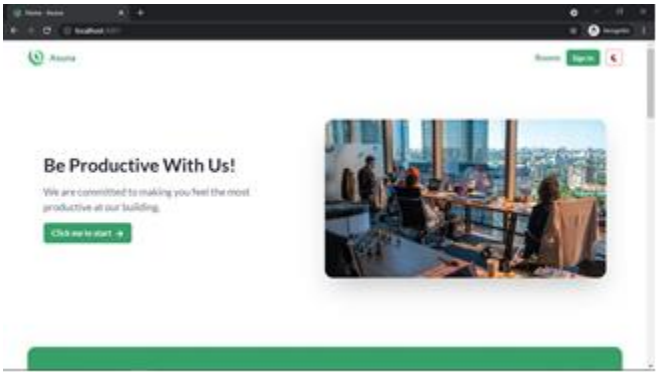

Figure 2:Main page

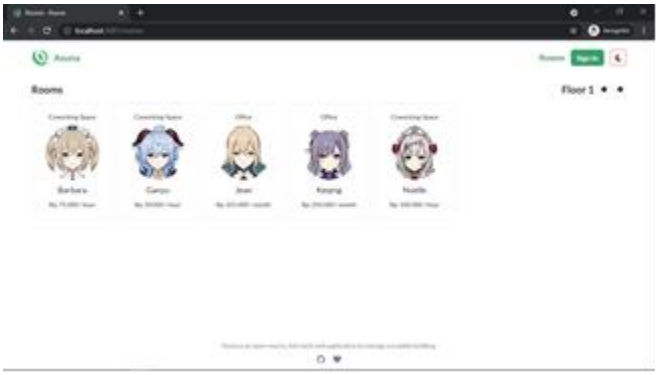

Figure 3: Room page

Figure 4: Order page

Furthermore, an owner could perform CRUD operations on the entities of the system, and they can also see earnings. An administrator could register a visitor to a room directly, and they can also manage orders. An order is assigned to an admin to prevent rare conditions of an admin accepting two same orders at once. Fig. 5 shows the Admin panel page of the system. Lastly, this system is also packed with error pages. If an unauthorized user attempts to access a protected page, the system will give an error message. The same thing will happen if a user tries to access a non-existent route in the system.

Figure 5: Admin panel page

3.4 Testing results

For the testing part of the system, we used internal and external testing. At the internal testing, we used the white-box and black-box testing method, where we analyzed the source code and the live version of the system to find bugs, errors, unexpected behaviors, or other possible problems. From the results of our tests, we did find some bugs, such as 1) users should not be able to change their username, 2) on the 'All Orders' button, there should be a back/ return button for a good user experience, 3) several messages need to be changed when an admin manages orders in 'All Orders', 4) the 'Delete' and 'Add Rooms' features should only be able to be used by the owner, 5) sometimes, the voucher feature caused system glitch. After we discovered the bugs and glitches, we fixed them immediately.

For the external testing phase of the system, we applied Technology Acceptance Model (TAM) (Davis, 1989). TAM has been used to measure the acceptance level of a new or modified system to the user within different kinds of fields (Dhammayanti et al., 2019; Nugraha & Hansun, 2020). With this tool, we can get the perceived usefulness and perceived ease of use of the system that has been built for the intended user. We will measure the 'Usefulness', 'Ease of Use', and 'Acceptance' metrics with the said method, which was implemented using the Likert Scale with the rating score of one to seven. The results of the external testing show positive feedbacks from around 20 users. We found out that the system possesses a good user interface, easy to use, and proven to be useful, as we have successfully managed to make the business process to reserve and order offices and co-working spaces to be much easier, efficient, and simplified.

Lastly, the output of this research project could be accessed from several links, such as a GitHub repository at https://github.com/lauslim12/Asuna, web front-end at https://www.asuna.vercel.app/, and API back-end at https://asuna-api.herokuapp.com/. It should be noted that the API is not intended to be accessed by the public. The API is specifically used for handling GET, POST, PATCH, and DELETE requests from the web front-end. The CORS mechanism will prevent any unauthorized access to the back-end. Please inform the first or corresponding author for further contribution to this project.

4. Conclusion

In this research project, we have successfully built a building management system for offices and co-working spaces. From the results of the Technology Acceptance Model survey, our respondents said that the built system is easy to use, can be used to help people in ordering or reserving offices or co-working spaces, and can also be used to help managers to manage their buildings easily. The system has proven itself to be highly scalable due to the stack technology used. According to the respondents, the system does have high-performance capabilities, and their requests were handled in a short amount of time. Moreover, scalability should not be an issue when using this system.

In order to improve this research project, several suggestions are provided as follows: 1) implement Redis, a NoSQL database dedicated to caching purposes to increase the processing time of the system, 2) implement Docker to containerize the system with all its dependencies (databases, API, etc.) to help easing the deployment process, 3) implement a micro services architecture, such as Kubernetes, so that the system could replicate itself into multiple containers, 4) add internationalization (i18n) support by offering several languages in the system. Moreover, future research to add predictive functionality in the system may also be done by using several prediction methods, such as Weighted Exponential Moving Average (Hansun, 2013) or even LSTM and Bi-LSTM deep learning methods (Hansun & Suryadibrata, 2021).

Acknowledgment

The authors would like to thank Universitas Multimedia Nusantara for the support and facilities given for this research project. We would also like to credit mi HoYo and Unsplash for providing different public and free assets to be used in this research project.

REFERENCES

Alsaqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies (IJIM)*, *14*(11), 246–270. https://doi.org/10.3991/ijim.v14i11.13269

Aottiwerch, N., & Kokaew, U. (2017). Design Computer-assisted Learning in an Online Augmented Reality Environment based on Shneiderman's Eight Golden Rules. *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 1–5. https://doi.org/10.1109/JCSSE.2017.8025926

Bianchi, F., Casnici, N., & Squazzoni, F. (2018). Solidarity as a Byproduct of Professional Collaboration: Social Support and Trust in a Coworking Space. *Social Networks*, *54*, 61–72. https://doi.org/10.1016/j.socnet.2017.12.002

Bouncken, R. B., Clauss, T., & Reuschl, A. (2016). Coworking-spaces in Asia: A Business Model Design Perspective. *SMS Special Conference Hong Kong 2016*.

Bouncken, R. B., Laudien, S. M., Fredrich, V., & Görmar, L. (2018). Coopetition in Coworking-Spaces: Value Creation and Appropriation Tensions in an Entrepreneurial Space. *Review of Managerial Science*, *12*(2), 385–410. https://doi.org/10.1007/s11846-017-0267-7

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, *13*(3), 319–340. https://doi.org/10.2307/249008

Dhammayanti, K., Wicaksana, A., & Hansun, S. (2019). Position Placement DSS Using Profile Matching and Analytical Hierarchy Process. *International Journal of Scientific & Technology Research*, *8*(11), 204–207. http://www.ijstr.org/final-print/nov2019/-Position-Placement-Dss-Using-Profile-Matching-And-Analytical-Hierarchy-Process.pdf

Gonçalves, L. (2018). Scrum: The Methodology to Become More Agile. *Controlling & Management Review*, *62*(4), 40–42. https://doi.org/10.1007/s12176-018-0020-3

Hansun, S. (2013). A new approach of moving average method in time series analysis. *2013 Conference on New Media Studies (CoNMedia)*, 1–4. https://doi.org/10.1109/CoNMedia.2013.6708545

Hansun, S., & Suryadibrata, A. (2021). Gold Price Prediction in COVID-19 Era. *International Journal of Computational Intelligence in Control*, *13*(2). https://www.mukpublications.com/ijcic-v13-2-2021.php

heroku.com. (n.d.). *Heroku*. Retrieved June 22, 2021, from https://www.heroku.com/

Hidalgo, E. S. (2019). Adapting the Scrum Framework for Agile Project Management in Science: Case Study of a Distributed Research Initiative. *Heliyon*, *5*(3), e01447. https://doi.org/10.1016/j.heliyon.2019.e01447

Khalid, A., Butt, S. A., Jamal, T., & Gochhait, S. (2020). Agile Scrum Issues at Large-Scale Distributed Projects. *International Journal of Software Innovation*, *8*(2), 85–94. https://doi.org/10.4018/IJSI.2020040106

Mittal, A. (2013). E-commerce: It's Impact on Consumer Behavior. *Global Journal of Management and Business Studies*, *3*(2), 131–138. https://www.ripublication.com/gjmbs_spl/gjmbsv3n2spl_09.pdf

Nugraha, A., & Hansun, S. (2020). Fiturebot: CS Chatbot Application using Nazief-Adriani Algorithm. *International Journal of Emerging Trends in Engineering Research*, *8*(2), 350–354. https://doi.org/10.30534/ijeter/2020/18822020

Seo, J., Lysiankova, L., Ock, Y.-S., & Chun, D. (2017). Priorities of Coworking Space Operation Based on Comparison of the Hosts and Users' Perspectives. *Sustainability*, *9*(8), 1494. https://doi.org/10.3390/su9081494

Sharma, A., Thung, F., Kochhar, P. S., Sulistya, A., & Lo, D. (2017). Cataloging GitHub Repositories. *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 314–319. https://doi.org/10.1145/3084226.3084287

Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM Model for Agile Methodology. *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 864–869. https://doi.org/10.1109/CCAA.2017.8229928

Vercel. (2021). *Next.js on Vercel*. https://vercel.com/docs/next.js/overview