J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr
Copyright ⓒ 2021 KSII

Design and Implementation of Disinfection Service Platform based on MVC Pattern Using Web/App☆

Ye-jin Jang[1] Yu-min Jo[1] Ji-in Shin[2] Yun-jeong Jang[2] Da-woon Jeong[3] Jong-ho Paik[2*]

ABSTRACT

Due to the COVID-19 pandemic, issues related to quarantine are emerging. There are various infection prevention methods, but among them, it is certain to frequently disinfect the surface or the entire space of an object that has come into contact with the virus. However, the reality is that the number of times such disinfection is legally designated and the required number of personnel are very different depending on the building and facility. For this reason, there are no companies and systems that can professionally receive orders for disinfection work. In order to solve the aforementioned problems, this paper presents a method to design a disinfection service platform based on the MVC pattern, and implements the required functions based on this. Through this, it is possible to build a more systematic system, and it is hoped that it will be of great help to quarantine with an orderly process for disinfection work.

☞ keyword : Covid-19, Disinfection, Service Platform, MVC pattern, Node.js, MySQL

1. Introduction

Due to the COVID-19 pandemic, disinfection has become a part of life. Accordingly, various methods for effective and efficient disinfection work are being sought, and new technologies are being applied. One of the ways a virus that causes COVID-19, such as SARS-CoV-2, can be transmitted to humans is infection by surface contact. The virus that causes COVID-19 can stick to surfaces. People can become infected when they touch the surface with the virus and then touch their nose, mouth, or eyes [1]. There are several ways to prevent surface infection: Personally, there are ways to wash your hands regularly or use hand sanitizer. Moreover, frequent disinfection of the surface of an object or the entire space is a surefire way. For this reason, the importance and demand for disinfection work are increasing. However, there is a need to improve methods and technologies that can quickly and conveniently quarantine a wide range of places.

The current situation in the Republic of Korea is that the number of times of disinfection legally designated varies depending on the building and facility. Also, there is a big difference in the amount and the intensity of disinfection work depending on the facility. Therefore, reflecting this situation, it is realistically impossible to set up a company specializing in only one disinfection task. In addition, it is difficult to recruit a disinfectant who specializes in only the disinfection work desired by the facility. In addition, each facility has a variety of tasks, and it isn't easy to hire multiple disinfectants accordingly.

Disinfection jobs are usually recruited through tenders. In the bidding process for disinfection work, the most important thing about disinfectants' hiring conditions is that they must have completed legally certified training. However, there is no organized system that manages those who have completed education or supports employment. Currently, disinfectants mainly apply for jobs through acquaintance or internet community sites[2].

In this paper, to solve the aforementioned problems, we present a method to design a disinfection service platform

1 Major in Software Fusion, Seoul Women's University, Seoul, 01797, Republic of Korea
2 Dept. of Software Convergence, Seoul Women's University, Seoul, 01797, Republic of Korea
3 ADR Labs, Seoul, 08378, Republic of Korea
* Corresponding author : jhpaik@swu.ac.kr
[Received 8 October 2021, Reviewed 14 October 2021, Accepted 4 November 2021]
☆ A prelimimary version of this paper was presented at APIC 2021.

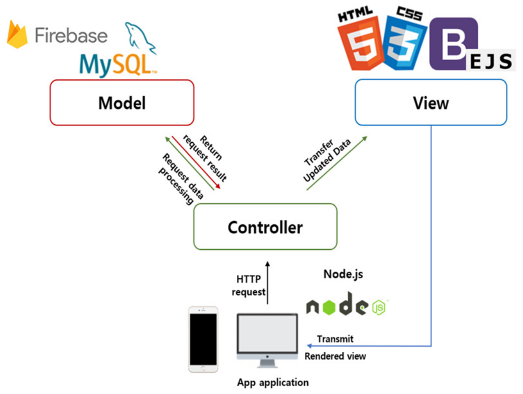
based on the MVC(Model-View-Controller) pattern and implement the platform for each required function accordingly. Since the functions required for each user are different, disinfection businesses can use the platform through the Web and disinfectants through the App. The proposed service platform is only for everyone related to disinfection work. Therefore, only disinfectants who have completed legal training and companies registered as a business can join. The platform proposed in this paper manages the overall process of disinfection work. The platform is an intermediary between the disinfection Business and the disinfectant in each cycle, from recruitment of disinfectants to hiring and reporting disinfection work.

In this paper from this point, the company that orders for disinfection work is called Disinfection Business, and the worker who performs disinfection work is called Disinfectant.

2. Design of Disinfection Service Platform based on MVC Pattern

In this paper, the disinfection service platform is designed and implemented using the MVC pattern. MVC (Model-View-Controller) is one of the most extended architectural patterns due to the advantages it provides through its separation of concerns of applications characterized by their user interface and data management. This pattern decomposes an application into three main components: model, view, and controller[3]. The model plays a role in processing the object or data received from the controller according to the business logic of the application, the view presents this data in different ways depending on the user's needs[4], and the controller is the part responsible for the flow of objects or data[5].

The platform proposed in this paper uses MySQL and Firebase for Model, Html/CSS for View, Bootstrap and EJS, and Node.js for Controller. Figure 1 is an MVC pattern plot diagram of the proposed service platform.



(Figure 1) MVC pattern Plot of the Proposed Platform

2.1 Model

The model is responsible for defining what the application will do and handling the internal logic. Data, constants, variables, and databases to be processed correspond to Models. There is no information about the Model's View or Controller, only information can be returned or set. The database of the proposed service platform is composed as shown in Figure 2 and Table 1 below.

First, we need a table to store information about the disinfection business that uses this platform. Basic information such as name, ID, password, business registration number, phone number, fax number, e-mail address, and a unique number is stored for each disinfection table. When registering as a member, the disinfection business with a business registration certificate is only possible, so the administrator's approval is absolutely necessary. Immediately after the disinfection company submits the membership application, the default value of the bApproval column is set to 0. After administrator approval, the value is changed to 1 to complete membership registration.

Similarly, we need a table to store information about the disinfectant. Basic information such as name, ID, password, legal authentication number for disinfection education, phone number, e-mail, date of birth, gender, and a unique number is stored for each disinfectant table. As a disinfectant business, only disinfectants who have completed legal education on disinfection work can join in. Therefore, since the value of wApproval is 0 before admin approval, the value of wApproval is changed to 1 after approval, and the membership registration is finally completed. In addition, in the process of registering, we receive the desired area from the disinfectant and store it in DB so that in the future, recommendations between disinfectant work and disinfectants can be made.

After the disinfectants perform their disinfection work, each disinfectant can attain the experience of the disinfection work. Career experience is one of the essential factors in recommending disinfectant and disinfection business in the future. The career table stores information such as category, duration, year, and grade for disinfection jobs.

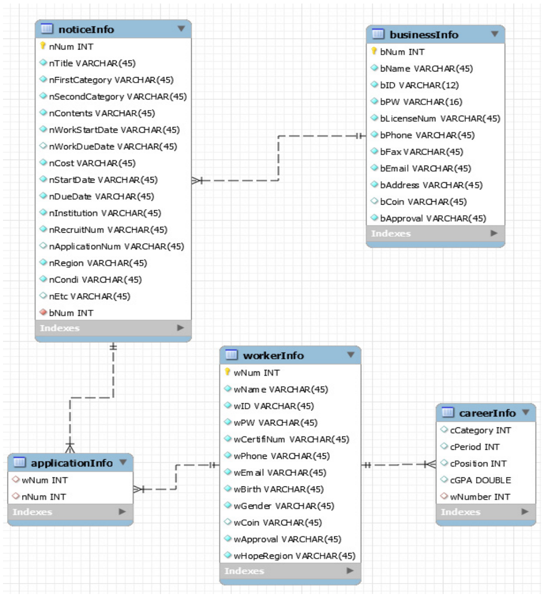
The disinfection business notifies the recruitment of disinfectants after membership registration. In other words, you need a table to store each notice. The notice table contains the unique number of each notice, notice title, major and sub-category of notice, notice content, notice start date, notice deadline, salary, notice start date and deadline, notice date, institution that applied for notice, number of recruitments, number of recruits, regions and optional preferential treatment are listed in the notification application form. Since multiple notices can be posted for each disinfection business a 1:N relationship is required between the disinfection business table and the notice table.

Disinfection business must select applicants directly or through a matching algorithm after application for notice. The applicant table required to store some applicants temporarily exists in a 1:N relationship between the notice table and the disinfectant table, respectively.

We chose MySQL as DBMS because we need these relationships between all tables. And all query statements are executed using Node.js SQL statements. In other words, the database server which is Node.js handles the queries by SQL[6].

2.2 View

In this paper, the view is designed after a process of deriving a function definition and users' requirements. And we visualize the concept and interface by sketching a wireframe based on the determined function definition[7]. Especially, the interface is designed by using Adobe XD, a design and prototyping tool that can identify and produce



(Figure 2) E-R Diagram designed of the Purposed Platform

(Table 1) Function Definition of the Proposed Platform

DB table	Function
businessInfo	Save disinfection business information (name, ID, password, business registration certificate number, phone number, fax number, email, address, bApproval)
workerInfo	Save disinfectants information (name, ID, password, legal certificaion number about disinfection education, phone number, e-mail, date of birth, and gender, wApproval, desired-regions)
careerInfo	Store category, period, year, and rating about disinfection work
noticeInfo	Store the unique number of each notice, notice title, major-classification of notice and sub-classification of notice, notice content, notice start date, notice deadline, salary, notice- posting start date and deadline, name of agency that applied notice, recruitment number, number of applicants, regions, optional preferential treatments, and other matters
applicationInfo	Store corresponding unique Number of disinfectants and notice, respectively

user experiences in advance. In addition, we design the web and app separately.

The screen composition of the web is divided mainly into three types according to the function of the platform: the main screen, the notice-posting screen, and the report-management screen. On the main screen, the advertisement banners of disinfection businesses are designed in the middle. In addition, a button linked to the login and member registration screen is located at the top. The notice-posting screen is designed as a general notice board screen. Disinfection businesses can check the title, schedule, and the number of recruits of each notice. If disinfection businesses click on the title of the notice, they will be routed to the notice detail view page. On the report-management screen, the disinfection business can read all of the reports written by disinfectants when they finish their work. This is also designed in the form of a notice board.

The app screen configuration is different from the web screen configuration. The app is designed the list of notice is displayed directly without the main screen. When Disinfectants click the several notice, it is routed to the detail view page, which is similar to the screen of the web. Disinfectants can check the list of their applied notice and selected notice on My-Page screen. For convenience, the navigation menu is set at the bottom.

2.3 Controller

The implemented platform designed Node.js as a server. Node.js is a server that supports asynchronous methods, which is easy when implementing the server with a lot of input/output (I/O) tasks. Even if hundreds of I/O requests are received, the main thread delegates the request to the background and receives more processing of the request. When the background informs that each request has been processed, it processes the callback function[8]. The advantage of Node.js is that it is extremely fast because it does not care whether background tasks are completed or not.

In the part of Controller, there were three most important parts when implementing the platform. Firstly, it is the request-response processing without clogging. In other words, when the disinfection company using the Web and the disinfection worker using the App responded to the server, it was essential to handle the data without error for routing processing. When using Node.js, it is particularly important to separate modules or use them in appropriate place. Therefore, it is important to respond to the server according to the request pass by creating a routing function in advance.

The second is to access the database using the correct SQL query in the right place is also important because the database stores many kinds of data, including user information. In addition, it is important to properly render data from a database using SQL query to a web browser. The router middleware in the Express module and EJS(Embedded JavaScript) templates were rendering to a web browser. Basically, the routing module branches the URI or HTTP request through the user’s web browser, with the help of the "Express" module[9]. The router middleware can solve the hassle of checking the request URL one by one because router middleware distinguishes based on the path what the client requested function is. EJS is a type of Server Side Template Engine, which allows servers to create and float HTML of data from a DB or API into a template. In other words, on implemented platforms, EJS renders HTML designed via Bootstrap and delivers it to clients at high speed through a web browser. The App does not use the template engine but renders it to XML after routing processing in the same way.

The last is to encrypt the data to be stored in the DB securely. Data encryption such as application-level encryption is one of the solutions to protect data confidentiality and face security threats. Application-level encryption is a process in which data is encrypted before being sent to the database[10]. The last is to encrypt the data to be stored in the DB securely. Application level Data Encryption has several benefits. Among them, we focused on Move data Secure. Move data Secure means the transfer and use of data will be in secure condition[11].

As a result, Node.js provides many modules and packages. Many kinds of modules and packages on node.js made it more convenient to implement the server.

2.4 Using AWS and Github

When implementing the service platform proposed in this paper, if the cloud service is not used, testing is possible only in the local test environment. These limitations can be addressed using cloud computing. Cloud computing is a network-based environment that focus on sharing computation resources. Cloud resources are provided to users as a service on as needed basis[12]. In addition, cloud services have the advantages of being able to quickly establish the environment required for implementation, easy maintenance, and excellent security. For these reasons, we implemented the service platform using a cloud service called AWS. And all the code was stored in a private repository on Github, and the EC2 instance and the corresponding Github were linked.

3. Implementation of Service Platform by Functions

The main functions of the platform proposed in this paper are member management, notice management, and disinfection work management. All of these functions are implemented based on the previously designed MVC pattern. Before the full-scale implementation, the requirements for each function were summarized, as shown in Table 2 below.

(Table 2) Requirements by Function

Function	Specifics	Subject
Member Management	Profile Management	Disinfection Business, Disinfectant
	Career Management	
Notice Management	Notice Registration	Disinfection Business
	Notice Application	Disinfectant
Disinfection Work Management	Disinfection Work Result Report Management	Disinfectant

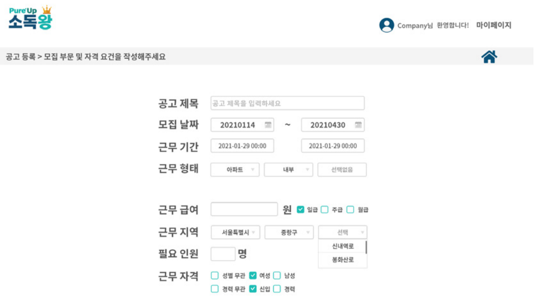
3.1 Member Management

The proposed platform can only be used by approved users. Approval conditions vary depending on the subject. Disinfection business must submit a business registration certificate, and disinfectants must submit a legal education certificate at the time of membership registration. The platform administrator confirms each of the two types of files submitted by the user and grants approval. Only authorized users can finally use the platform freely. For user authentication, the Passport module, one of the Node.js modules, is used. The Passport module provides hundreds of authentication methods. Among them, user authentication is implemented using a local strategy that can compare user information stored in the database. When a client requests authentication, the Passport module authenticates the user and stores the user information in the session. Through this session information, the platform continuously identifies whether a user is logged in or not, thereby limiting the functionality of the platform[13].

In addition, the platform manages the career details of each disinfectant. When the disinfectants complete their work, this is automatically saved as the career history of the disinfectant, and the disinfection business can evaluate the disinfectants that have completed their work.

3.2 Notice Management

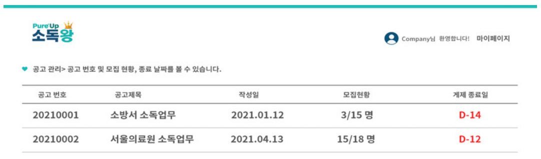
The notice management function is the most core function of the proposed platform. The disinfection business operator publishes the notice on the platform including the overall contents of disinfection work and the requirements for recruitment of disinfectants. The form for the disinfection operator to fill out the notice is as shown in Figure 3. Also, the screen where the disinfection business can see the list of posted notices is shown in Figure 4.



공고 등록 > 모집 부문 및 자격 요건을 작성해주세요

공고 제목	공고 제목을 입력하세요
모집 날짜	20210114 ~ 20210430
근무 기간	2021-01-29 00:00 2021-01-29 00:00
근무 형태	아파트 / 내부 / 선택없음
근무 급여	원 [✓ 일급] 주급 월급
근무 지역	서울특별시 / 중랑구 / 선택 (신내역로, 봉화산로)
필요 인원	명
근무 자격	성별 무관 [✓ 여성] 남성
	경력 무관 [✓ 신입] 경력

(Figure 3) Web Screen of Notice Registration

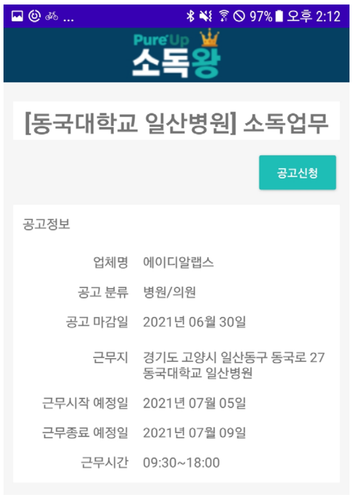


공고 관리> 공고 번호 및 모집 현황, 종료 날짜를 볼 수 있습니다.

공고 번호	공고제목	작성일	모집현황	게제 종료일
20210001	소방서 소독업무	2021.01.12	3/15 명	D-14
20210002	서울의료원 소독업무	2021.04.13	15/18 명	D-12

(Figure 4) Web Screen about List of Notice Administration

The data written in the form is put in the router object, which was considered important in the design process, and delivered to the server in the post method. The post method has high security and has the advantage of being able to transmit large object values such as query string data and radio buttons. Disinfectants check the recruitment requirements of the notices by reviewing the various notices posted. After that, they apply for notices that are deemed appropriate for them. Figure 5 shows the app screen where disinfectants can apply for the notice they want.



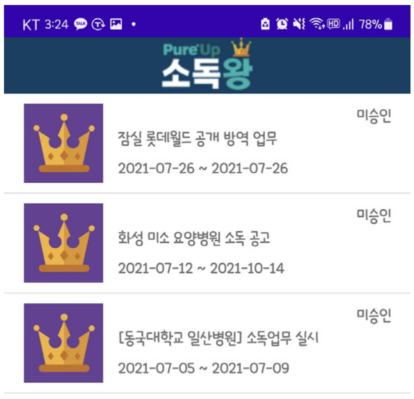
[동국대학교 일산병원] 소독업무

공고정보

업체명	에이디알랩스
공고 분류	병원/의원
공고 마감일	2021년 06월 30일
근무지	경기도 고양시 일산동구 동국로 27 동국대학교 일산병원
근무시작 예정일	2021년 07월 05일
근무종료 예정일	2021년 07월 09일
근무시간	09:30~18:00

(Figure 5) App Screen of Applicaion

At this time, disinfection businesses can check the status of application for notice posted from time to time. When the notice's deadline is ended, disinfection businesses can check the list of applied disinfectants and their profiles, and finally hire some disinfectants to work with. Once the selection of the disinfectants is completed, the disinfectant can check the application results individually. The related screen is shown in Figure 6.

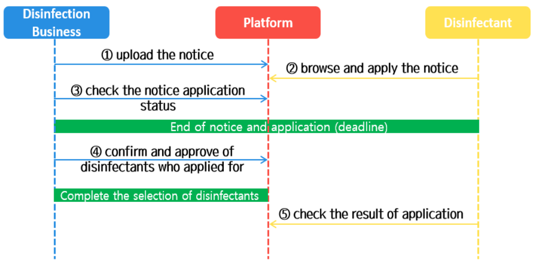


잠실 롯데월드 공개 방역 업무 2021-07-26 ~ 2021-07-26	미승인
화성 미소 요양병원 소독 공고 2021-07-12 ~ 2021-10-14	미승인
[동국대학교 일산병원] 소독업무 실시 2021-07-05 ~ 2021-07-09	미승인

(Figure 6) App Screen about List of Applied Notice

Firebase's ID Token method is used to display the application result as a notification on the disinfectant's personal application. When communicating between the Firebase client app and the Node.js server, the server needs to identify the user who currently logged in. To securely identify the user, the user's ID token is sent to the server using HTTPS after a successful login. The server then verifies the integrity and authenticity of the ID token and gets the uid from the ID token. The uid sent in this way can be used to securely identify the user currently logged into the server[14].

The overall sequence diagram of notice management function is shown in Figure ().


Disinfection Business
Platform
Disinfectant
① upload the notice
② browse and apply the notice
③ check the notice application status
End of notice and application (deadline)
④ confirm and approve of disinfectants who applied for
Complete the selection of disinfectants
⑤ check the result of application

(Figure 5) Sequence Diagram of Notice Management Process

3.3 Disinfection Work Management

Disinfection operators must register the list of tasks to be handled by disinfectants on the platform prior to disinfection work. Disinfectants check the list and proceed with their work. When the work is completed, the disinfectants write a report on the results of the disinfection work in the application. The created data is delivered to the Node.js server and converted into a file. At this time, the converted files using the fs module of Node.js are stored separately in Firebase. Disinfectants will no longer have to write reports manually, and disinfectants will be able to manage reports in online.

4. Conclusion

In this paper, we proposed a disinfection service platform that mediates disinfection work between disinfection business and disinfectants. The proposed platform was designed and implemented by applying the MVC (Model-View-Controller) pattern, which is one of the patterns used in software design.

In the case of the Model, conceptual design and logical design were performed based on the content that defined the requirements and functions of this service platform. Disinfectant information, disinfection business information, notice information, applicant information, and career information correspond to content of this service platform, and each table requires a relationship respectively, so the Model was implemented using MySQL as DBMS. Also, when dealing with unstructured data, Firebase was used.

For View, the screen was designed using Adobe XD, which is a prototyping tool. In particular, it is designed to use customized UX for each user by dividing the web and the app. This was later implemented using Bootstrap and EJS, which is a screen template of Node.js.

In the case of Controller, Node.js is used to enable data transfer fastly through asynchronous method. At this time, the focus is that fast request processing, proper use of various modules and packages, and data encryption. Among the various modules of Node.js, the server was built through the Express module, and user authentication was implemented using the Passport module.

The overall platform was implemented by functions according to user requirements which are Member Management, Notice Management, and Disinfection Work Management.

Until now, there was no system related to disinfection work. However, in this paper, a service platform that can systematically manage the overall process of disinfection was proposed. It is hoped that the disinfection work process will be in order through the proposed service platform, and disinfection workers will be able to work more comfortably than now. In addition, it is expected to contribute to daily life related to disinfection.

Acknowledgment

This work was supported by a research grant from Seoul Women's University(2021-0207).

References

[1] NCIRD, “Cleaning and Disinfecting Facilities,” CDC, 2021.
https://korean.cdc.gov/coronavirus/2019-ncov/community/disinfecting-building-facility.html

[2] Yumin Jo, Yejin Jang, Hajeong Yoo, Junhwan Kim, and Jongho Paik, “Design of Sterilization Service Platform based on O2O(Online to Offline),” Proceedings of the Fall Conference of the Korea Internet and Information Society(KSII), Vol 21. No.2, 2020.

[3] S. Burbeck, ‘‘Applications programming in smalltalk-80 (TM): How to use model-view-controller (MVC),’’ Smalltalk, vol. 5 pp. 1–11, 1992.
http://st-www. cs. uiuc. edu/users/smarch/st-docs/mvc. html.

[4] D. Guamán, S. Delgado and J. Pérez, "Classifying Model-View-Controller Software Applications Using Self-Organizing Maps," in IEEE Access, vol. 9, pp. 45201-45229, 2021.
https://ieeexplore.ieee.org/document/9380344

[5] Sehwan Yeo and Jungsun Kim, "Android Application Development following the MVC Architecture," Proceedings of the Korean Society of Information Sciences, Vol 39(1D), pp.76-78, 2012.
http://www.riss.kr/link?id=A60223046

[6] Khaled Saleh Maabreh, “An Enhanced University

Registration Model Using Distributed Database Schema," TIIS, VOL. 13, NO. 7, pp. 3533-3549, 2019. https://doi.org/10.3837/tiis.2019.07.011
[7] Jisu Park and Hun Kim, "7 SECRETS OF UX DESIGN," Ahn graphics, 2013.
[8] HyunYoung Cho, "Node.js Textbook," Gilbut, 2019.
[9] SongYeon Lee and JongHo Paik, "An Asynchronous-Driven Node.js Based Intermediary-free Direct Deal Distribution Platform Converged with Cloud Service," TIIS, VOL. 13, NO. 8, pp.4212-4226, 2019. https://doi.org/10.3837/tiis.2019.08.022
[10] L. Bouganim and Y. Guo, "Database encryption," Encyclopedia of Cryptography 2nd Edition Springer US, pp. 307-312, 2011. https://doi.org/10.1007/978-1-4419-5906-5_677
[11] Shahrzad Azizi and Davud Mohammadpur, "Searchable Encrypted String for Query Support on Different Encrypted Data Types", TIIS, Vol. 14, No. 10, 2020. https://doi.org/10.3837/tiis.2020.10.015
[12] Muhammad Imran Tariq, "Agent Based Information Security Framework for Hybrid Cloud Computing," TIIS, VOL. 13, NO. 1, 2019. https://doi.org/10.3837/tiis.2019.01.023
[13] JaeGon Jung, "Do it! Node.js Programming," Easy-Publishing, 2018.
[14] Google, "Verify ID Tokens," 2021. https://firebase.google.com/docs/auth/admin/verify-id-tokens

◐ 저 자 소 개 ◑

Ye-jin Jang
2021 B.S in Software Convergence from Seoul Women's University
2021~present M.S in Software Fusion from Seoul Women's University
Research Interest : Server, Database, Intelligence Platform, Android Application, Programming
E-mail : yuvm02@swu.ac.kr

Yu-min Jo
2021 B.S in Mathematics from Seoul Women's University
2021 B.S in Software Convergence from Seoul Women's University
2021~present M.S in Software Fusion from Seoul Women's University
Research Interest : Database, Algorithm, Machine learning, BigData, Intelligence Platform, Recommendation System
E-mail : rosemin97@swu.ac.kr

Ji-in Shin
2018~present Undergraduate in Software Convergence from Seoul Women's University
Research Interest : Server, Intelligence Platform, Android Application, Programming
E-mail : sonia9.shin@gmail.com

◐ 저 자 소 개 ◑

Yun-jeong Jang
2019~present Undergraduate in Software Convergence from Seoul Women's University
Research Interest : Database, BigData, Intelligence Platform
E-mail : dbs920313@naver.com

Da-woon Jeong
2021 B.S in Software Convergence from Seoul Women's University
2021~present Researcher at ADR Labs
Research Interest : Front-End, Server, Programming
E-mail : ekdnsdl4@adrlabs.com

Jong-ho Paik
1994 B.S in Electrical and Electronic Engineering from Chung-Ang University, Seoul, Korea
1997 M.S in Electrical and Electronic Engineering from Chung-Ang University, Seoul, Korea
2007 Ph.D in Electrical and Electronic Engineering from Chung-Ang University, Seoul, Korea
1997~2011 Director, Mobile Terminal Convergence Research Center, Korea
2011~present Professor in Software Convergence, Seoul Women's University
Research Interest : web-based communication, intelligence platform, machine learning, software testing, wireless/wired communications system design, video communications system design and system architecture for realizing advanced digital communications system and for advanced mobile broadcasting networks
E-mail : jhpaik@swu.ac.kr