|  |  |  |
| --- | --- | --- |
|  | ROOT2019 Final Research Report |  |

Title:

Development of Shelter Navigation with Considering Three Cs in the Corona Age

Author:

Hiroto NAKADA, Himeji Nishi High School

Prepared as Final Research Report for the Advanced Stage in the GSC/ROOT Program.

Advisors:

Masahide NAKAMURA

Date: 2021.02.28

Abstract: Japan is a disaster-prone country, and tremendous damage occurs every year. Due to the global pandemic of COVID-19 in 2020, evacuation shelters have been forced to reduce the number of residents, in order to reduce the risk of infection. Some evacuation shelters have been unable to accommodate evacuators due to full capacity.

In this study, we propose a novel application, Shelter Navi. It aims to avoid the formation of dense crowd in evacuation shelters by self-efforts of citizens themselves.

Shelter Navi manages the location and the congestion status of every shelter within a local government, and visualizes the information on a map. When a disaster occurs, citizens check the information with mobile phones, and evacuate to vacant shelters. As a citizen “checks in” to a shelter, Shelter Navi updates the status at real time. Thus, the app allows crowd-aware evacuation without any special equipment in the shelter. In this paper, we first define use cases of Shelter Navi, and then conduct domain modeling and service API design. Finally, we implement a prototype of Shelter Navi as a Web application, using latest application frameworks Spring Boot and Bootstrap.

**1. Introduction**

Japan is a disaster-prone country, with large-scale natural disasters occurring every year. In recent years, torrential rains caused by so-called abnormal weather conditions have caused tremendous damage, and there have been many people who lost their lives. In July 2018, more than 200 people were killed in the torrential rains that occurred in western Japan [1]. Many of the areas where the victims were killed were in flood-prone areas or disaster warning areas, and information to evacuate the areas was issued in advance. In other words, especially in heavy rainfall disasters, there are many cases in which residents died due to delayed escape. The main reasons for the delay in evacuation are lack of awareness of evacuation due to psychological effects such as normality bias that leads people to believe that they are safe [2]~[4], lack of knowledge such as not knowing the appropriate actions to take during a disaster, and lack of confirmation of evacuation sites [5].

In addition, the epidemic of the new coronavirus infection that occurred in 2020 requires new shelter management that takes into account infection control [6]. As a part of such measures, the number of people to be admitted to evacuation shelters will be limited and controlled to avoid three densities, and a situation in which evacuated residents are unable to enter the shelters is expected. In fact, when Typhoon No. 10 hit Kyushu in September 2020 and evacuation shelters were opened, there was a case where evacuated residents were refused admission to two shelters in a row [7].

Some municipalities have taken their own measures to deal with such problems. In Nichinan City, Miyazaki Prefecture, VACAN, an application that distributes information on the crowdedness of restaurants, etc., is used to distribute information on the crowdedness of evacuation centers, and this has had a certain effect [8]. However, the congestion status of evacuation centers is measured and updated manually by local government officials. Therefore, new measures necessary for the corona era are left to each municipality, which leads to an increase in the workload of the staff.

In this context, we set the research question as follows. "In the event of a disaster in the corona age, can residents evacuate to appropriate shelters by themselves without relying on the local government?"

In this study, we propose Shelter Navi, a mobile application that helps residents to evacuate quickly and safely, avoiding the formation of 3Cs in shelters by self-help effort of citizens. Shelter Navi uses a cloud server to manage the locations and congestion of shelters in a municipality, visualizes them on a map, and provides information for residents to avoid dense situations and evacuate in a dispersed manner. When a resident "checks in" to a shelter, Shelter Navi updates the congestion level in real time. This makes it possible to evacuate in consideration of 3Cs without requiring special equipment at each shelter.

In this paper, we define the use case of Shelter Navi and design the domain model and service. We also implement the Shelter Navi prototype as a mobile Web application using Spring Boot [9] and Bootstrap. Shelter Navi is expected to improve the evacuation awareness of residents and the efficiency of shelter operations in the With/After Corona age.

**2. Prepare**

**2.1 Disaster Evacuation in Japan**

In recent years, Japan has frequently experienced large-scale torrential rainfall disasters that are said to occur once in several decades due to abnormal weather conditions, resulting in many victims every year. In the case of the torrential rains in western Japan in July 2018, it was reported that many human casualties occurred even though evacuation advisories had been issued. [1] Among the victims, there were many cases of "delayed evacuation".

In order to encourage residents to evacuate in the event of a disaster, the Japan Meteorological Agency issues weather warnings for heavy rains, storms, and floods, and local governments issue evacuation advisories and instructions. Since evacuation orders are not enforceable, it is up to the residents to decide whether to evacuate or not, taking into account the risks involved in doing so. However, in such a case of imminent danger to oneself, due to the strong stress of having to process a large amount of information accurately within a time constraint, people tend to make the same risk assessment as in normal times, i.e., to take things easy, in order to keep their cool.

The "normalcy bias" is the tendency to make the same risk assessment as in normal times, that is, to be optimistic about the situation. It is believed that this bias leads to a delay in decision-making and evacuation actions, due to the tendency to interpret the situation in one's own favor and think, "I will be fine. [2] In addition, the lack of prior preparation and awareness among residents, such as not being able to confirm the evacuation site or not knowing the evacuation route, may also reduce the evacuation rate.

**2.2 Management of evacuation centers in the corona age**

In 2020, the global epidemic of a new coronavirus infection will require a change in style in all aspects of society. In terms of disaster prevention, countermeasures against infection in shelters are urgently needed, and each local government has issued a new manual for the operation of shelters that includes countermeasures against infection. For example, examples of the layout of evacuation spaces (number of people to be accommodated, spacing, etc.) and the implementation of temperature and medical examinations before accepting evacuees into shelters were newly indicated. In addition, in order to avoid crowding, the number of people per shelter had to be greatly limited.

Due to these infection control measures, there are concerns that some evacuees who do not know the situation may not be allowed to stay in shelters. In September 2020, when Typhoon No. 10 hit Kyushu and evacuation shelters were opened, there was a case where evacuees were refused admission to two shelters in a row [7].

Some municipalities have taken their own measures to deal with such problems. In Nichinan City, Miyamae Prefecture, the crowded status of shelters is distributed using VACAN, an application that distributes information such as the crowded status of restaurants. However, the congestion status of shelters is measured and updated manually by the local government staff.

In this way, new measures for the management of shelters in the corona age have been implemented in a trial-and-error manner, but they have basically been left to the local governments, which has led to an increase in the workload of their staff. In addition, there is a concern that residents may become dependent on such measures taken by local governments, and that self-help effort by residents may be suppressed.

**2.3 Research Questions**

Based on the above, we set the following research questions and study to answer them.

RQ "In the event of a disaster in the corona age, can residents evacuate to appropriate shelters by themselves without relying on the local government?"

**3 Method**

**3.1 System Overview**

We propose Shelter Navi, an application to support safe and smooth evacuation by self-help actions of residents as much as possible in times of disaster, while avoiding the formation of dense situations. Shelter Navi is composed of a cloud server and a smartphone owned by each resident. In order to solve the research question, we implement Shelter Navi using the following A1~A3 approach. Figure 1 shows the schematic diagram.

A1: Management of shelter information by cloud server

Shelter Navi manages the information of shelters managed by local governments on a cloud server. The information of shelters is divided into master information and status information. The master information consists of the name, address, latitude and longitude of the shelter, the number of people who can be accommodated, the manager, etc., and is registered in advance by the staff of the local government. The status information consists of the current occupants of each shelter, the number of occupants, the degree of congestion, etc., and is updated by the check-in described later in A3. In addition, residents who use Shelter Navi need to register their user information with the cloud server. The user information includes name, e-mail address, password, number of household members, and home address.

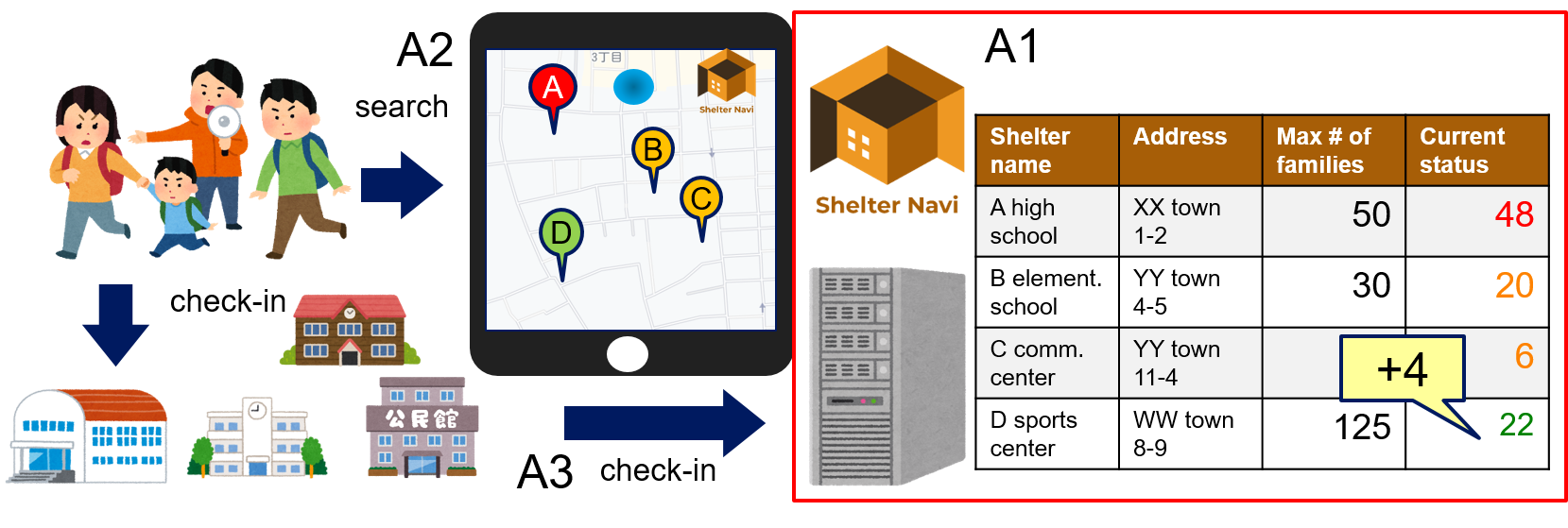


Figure 1 the schematic diagram of Shelter Navi

A2. Real-time distribution and visualization of crowded shelters

When a disaster occurs, Shelter Navi visualizes the shelter information in A1 on a map. Residents can check their current location (obtained by GPS) and nearby evacuation centers (indicated by pins on the map) with their smartphones. The level of crowdedness of each shelter can be checked by the color of the pin and detailed information, so residents can find the most suitable shelter based on their own situation and the crowdedness.

A3. Check-in at the shelter.

Shelter Navi records the check-in time of the resident in the cloud server, adds the number of residents in the household to the capacity of the shelter, and updates the crowded status. With the check-in function, the number of residents in each shelter can be automatically counted without the help of staff. For residents who are not familiar with smart phones, the system also provides a function that allows a staff member to check in residents on their behalf.

**3.2 Use cases**

To clarify the functions to be realized in Shelter Navi, we define use cases. The users of Shelter Navi are residents and local government employees, and we define use cases for each user role. The users of Shelter Navi are residents and local government employees, and the use cases are defined for each user role. The user registration of residents is done by the residents themselves, while the user registration of local government employees is done by the administrator of Shelter Navi. This is for the municipal staff users to refer to the personal information of residents in case of a disaster, and is a security consideration. Figure 2 shows the use case diagram of Shelter Navi.

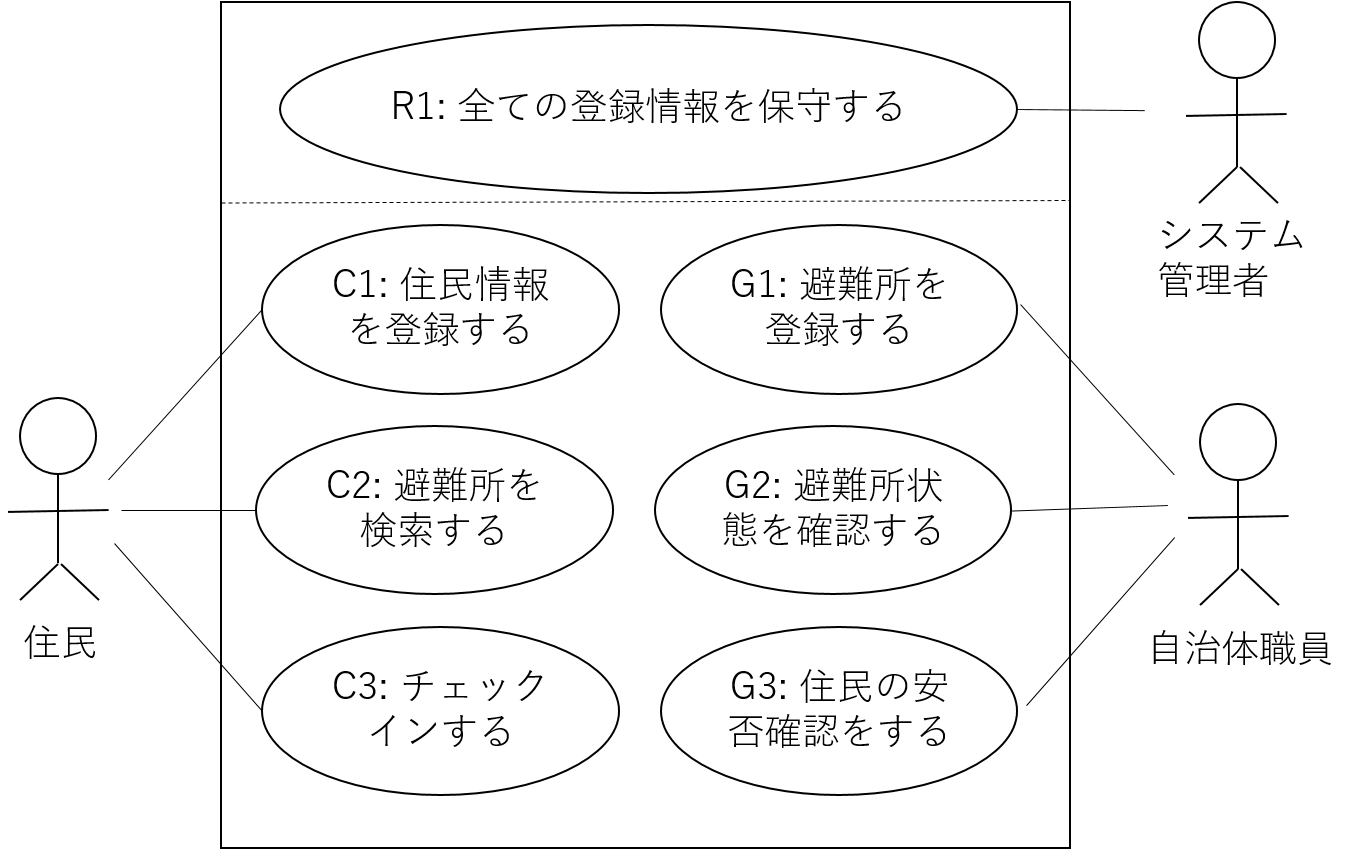


Figure 2 the use case of Shelter Navi

**3.2.1 Residents' Use Cases**

As use cases for residents, the following C1~C3 are shown.

C1: Registering resident information

In order to use Shelter Navi, residents need to register their resident information in advance. From the sign-up screen, they register their name, e-mail address, password, number of household members, address, and other information. After registration, residents log in to the application with their e-mail address and password.

C2: Searching for shelters.

When a disaster occurs, residents can search for nearby shelters on Shelter Navi. When a disaster occurs, residents can search for shelters in their vicinity on Shelter Navi. You can also click on a pin to see detailed information about that shelter. The detailed information includes master information such as the name and address of the shelter and the current congestion level.

C3. check-in

When a resident arrives at a shelter, he/she checks in on Shelter Navi. The system records the check-in of the resident. Furthermore, the system adds the number of household members of the resident to the capacity of the shelter and updates the crowded status of the shelter.

**3. 2. 2 Municipal use cases**

The following G1~G3 are shown as municipal use cases.

G1: Registering a shelter

The staff in charge of the municipality registers the master information of the shelter. G1. The staff member in charge of registering a shelter registers the master information of the shelter, including the shelter name, address, latitude and longitude, and the number of people who can be accommodated.

G2: Checking the status of shelters.

When a disaster occurs, the staff in charge of the local government can check the status of each shelter. The screen displays a list of all the municipalities in the region and the congestion status in real time. By selecting a specific shelter, the information and number of residents who have checked in can be checked. This eliminates the need for local government officials to go directly to shelters to check on crowded conditions, thereby preventing the spread of infection and reducing their workload.

G3: Confirming the safety of residents

In the event of a disaster, local government officials can confirm the safety of specific residents. The system searches for residents by name or e-mail address, and confirms information such as shelter which they have evacuated to and when they checked in. The safety confirmation is assumed to be performed by the local government staff based on requests from relatives, hospitals, and other related people and organizations.

**3.3 System architecture**

Figure 3 shows an image of the system architecture of Shelter Navi.

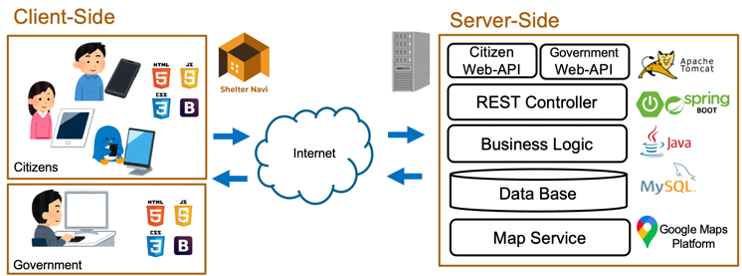


Figure 3 the system architecture of Shelter Navi

Shelter Navi is configured as a client-server Web application. The server side is equipped with cloud server functions and manages major data such as shelters, users, and check-ins. A typical layered architecture with a controller layer, domain layer, and persistent layer is adopted, and access to data and operations is done through REST-API. For the map function, external map cloud services are used in conjunction. The server-side technology used is Java as the implementation language, Spring Boot application framework, and Apache Tomcat web server. MySQL is used as the database, and Google Maps API is used as the map service.

On the other hand, on the client side, user interface functions are mainly arranged. To make it possible to run Shelter Navi use cases on a smartphone or PC web browser without being aware of the differences in OS and models, HTML, CSS, and JavaScript are the implementation languages. In addition, a responsive design using Bootstrap is incorporated for ease of use on smartphone screens.

**3.4 Domain Modeling**

**3.4.1 Overall structure and entity relationships**

In order to clarify the information to be managed in Shelter Navi, the structure of the entire domain and entity-related modeling are performed. Figure 4 shows the domain model diagram of Shelter Navi. As shown in Figure 4, there are five entities in the domain of Shelter Navi: Shelter (shelter), User (resident/user), Check-In (check-in or check-out to the shelter), ShelterState (shelter state), and UserState (user state). (shelter state), and UserState (user state).

Check-In associates a user with a shelter that the user has used, and one Check-In always associates one User with one Shelter. The ShelterState indicates the state of each shelter, and the UserState indicates the evacuation state of the user. Each of them is updated and created when a new Check-In object is created. In the following, the data items of each entity in Shelter Navi are described. 3.

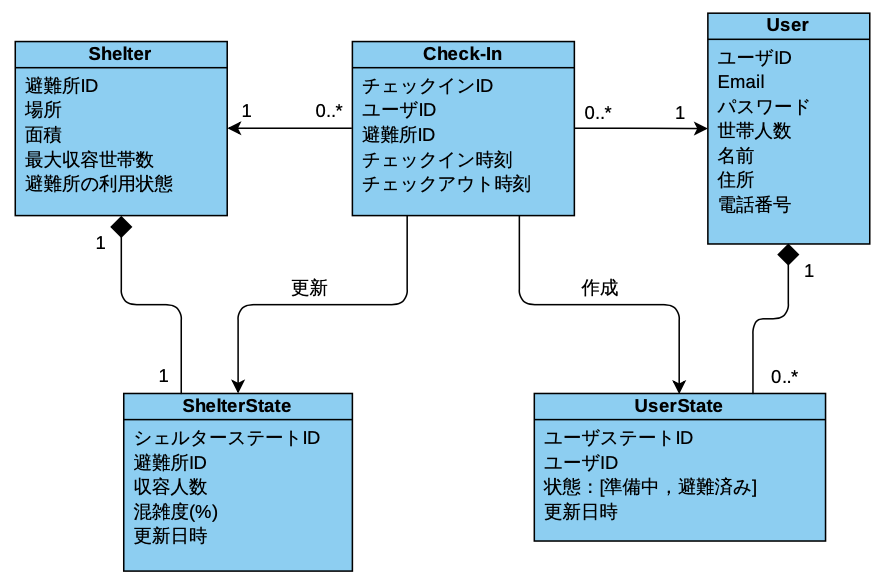


Figure 4 the domain model diagram of Shelter Navi

**3. 4. 2 Shelter:** shelter entity

The Shelter entity has an ID to identify which shelter the system is. It is also necessary to indicate the name and location of the shelter to the evacuating users. It also handles information about the opening of the shelter.

- **sid**: ID of the shelter

- **name**: Name of the shelter

- **address**: Address of the shelter

- **lat**: Latitude of the shelter

- **lng**: Longitude of the shelter

- **capacity**: Number of people the shelter can accommodate

- **isActive**: Whether the shelter is available

**3. 4. 3 User:** User entity

A user entity needs a user ID to uniquely identify a user. In addition, it handles the email address and password for registration to the service, the name and address required for safety confirmation, and the number of people in the household for calculating the degree of congestion.

- **uid**: User ID

- **email**: Email address

- **password**: Password

- **name**: User's name

- **address**: Address of the user

- **of familie**s: Number of families

- **Check-In**: Check-in entity

The Check-In entity needs a user ID and a shelter ID to manage which user has checked in to which shelter. In addition, the check-in time and the check-out time are also used to check whether there are multiple check-ins to a shelter.

- **cid**: Check-in ID

- **uid**: User ID

- **sid**: shelter ID

- **checkin-datetime**: check-in time

- **checkout-datetime**: checkout time

**3. 4. 5 ShelterState**: State of the shelter

The ShelterState entity represents the state of the shelter as it changes with the check-in (and check-out) of the residents. It has the following attributes: the ID of the target shelter, the current capacity, the current congestion level, and the update date. The capacity is calculated from the sum of the number of household members of all the users who have checked in. The congestion level is calculated by dividing the capacity by the number of people the shelter can accommodate.

- **id**: State ID of the shelter

- **sid**: ID of the shelter

- **num of people**: Number of people in the shelter

- **congestion**: Congestion level

- **changedAt**: Update date

**3. 4. 6 UserState**: User state

The UserState entity manages the evacuation state of a user and its history. To view the history of a specific user, a user ID is required. The UserState entity indicates the evacuation state of the user (home, preparing, evacuating, or evacuated). The date and time when the evacuation status changes are also recorded. In addition, the user status is recorded by adding new data, rather than updating a single data, so that the user can refer to the history data.

- **id**: User state ID

- **uid**: User ID

- **state**: Evacuation state [home, preparing, evacuating, evacuated]

- **changedAt**: Update date

**3.5 Main Services**

Shelter Navi is deployed as a RESTful Web-API that can be invoked by HTTP-REST in consideration of its scalability and interoperability with other applications. The details of each service are shown below.

**3. 5. 1 ShelterService**: Shelter service

This service performs CRUD (creation, reading, updating, and deletion) of shelters.

The following is a list of APIs. The following is a list of APIs.

- **createShelter( shelterForm )**: Creates shelter data based on the shelter ID, shelter name, and location information.

- **createShelter( shelterForm )**:Create and retrieve shelter data based on shelter ID, shelter name, and location information.

- **createShelter( shelterForm )**: creates and retrieves shelter data based on shelter ID, shelter name, and location information.

- **getShelter( sid )**: retrieves shelter data by specifying the shelter ID.

- **deleteShelter( sid )**:Deletes the corresponding shelter data by specifying the shelter ID.

- **clearAllShelters()**:Deletes all shelters data.

- **clearAllShelters()**:delete all shelters.

- **searchSheltersByDistance( lng, lat, distance )**:Specify longitude, latitude, and distance.

By specifying the longitude, latitude, and distance, the API retrieves all shelters within a radius distance[km] from the specified location coordinates (lng, lat). Our API adopts the method of calculating the global distance, and calculates the linear distance 𝑑 (𝑝, 𝑠) between the user's current position 𝑝 and the distance 𝑠 of the given shelter. Specifically, when the radius of the earth is 6371[km], the longitude coordinate of the user is 𝑙𝑛𝑔, the latitude coordinate is 𝑙𝑎𝑡, and the longitude coordinate of each shelter data is 𝑠.𝑙𝑛𝑔 and the latitude coordinate is 𝑠.𝑙𝑎𝑡, 𝑑(𝑝, 𝑠) is calculated by the following formula.

*𝑑(𝑝, 𝑠) = 6371 × arccos(*

*cos(𝑟𝑎𝑑(𝑙𝑎𝑡)) × cos(𝑟𝑎𝑑(𝑠.𝑙𝑎𝑡)) × cos(𝑟𝑎𝑑(𝑠.𝑙𝑛𝑔) - 𝑟𝑎𝑑(𝑙𝑛𝑔)) + sin(𝑟𝑎𝑑(𝑙𝑎𝑡)) × sin(𝑟𝑎𝑑(𝑠.𝑙𝑎𝑡))*

*)*

- **searchSheltersByKeyword( keyword )**: Search for shelters by specifying a string. It retrieves all shelter entities that partially match the shelter name or address.

**3.5.2 UserService**: User service

This is a service to perform user CRUD. The following is a list of APIs.

- **createUser( userForm )**: Creates a new user based on user information including e-mail address, password, name, address, and number of people in the household.

- **getUser( uid )**: Creates a new user based on user information including e-mail address, password, name, address, and number of people in the household.

- **createUser( userForm )**: Creates a new user based on user information including e-mail address, password, name, address, and number of household members.

- **updateUser(uid, userForm )**:Updates existing user information by inputting a user form.

- **updateUser(uid, userForm )**: Updates existing user information by entering a user form.

- **deleteUser( uid** ): Deletes the corresponding user entity by specifying a user ID.

**3.5.3 CheckInService**: Check-in service

This is a service that performs check-in and check-out operations for a shelter. A list of APIs is shown below.

- **checkIn( uid, sid )**: Performs the check-in process by specifying a user ID and shelter ID. Specifically, a new Check-In object is created and the creation date and time are recorded. At the same time, the number of people in the shelter is increased by the number of people in the user's household, the crowdedness of the shelter is recalculated, and the ShelterState object is updated. Furthermore, we create a new UserState object with the evacuation state set to "evacuated".

- **checkOut( uid, sid )**: Performs the checkout process by specifying the user ID and shelter ID. It obtains the latest check-in data uniquely determined by the user ID and shelter ID, and fills in the check-out time. At the same time, the number of people in the shelter is reduced by the number of people in the user's household, the crowdedness of the shelter is recalculated, and the ShelterState object is updated. Furthermore, we create a new UserState object with the evacuation state set to "evacuating".

**4. Implementation**

**4.1 Implementation of Shelter Navi Prototype**

Based on the design described above, we implemented a prototype of Shelter Navi. This prototype covers the resident use cases C1~C3. The data of 335 designated emergency shelters in Kobe City and 255 designated shelters in Himeji City were manually registered in the database. These shelters data were created based on the open data of shelters published by Kobe City [10] and Himeji City [11].

The technologies used in the prototype implementation are as follows.

- Server development language: Java

- Application framework: Spring Boot

- Database: MySQL 8.0.20

- Web server: Apache Tomcat 9.0

- Client development language: HTML5, JavaScript

- CSS library: Bootstrap 4

In the following sections, we outline the implementation of the prototypes we have created, using the "Login", "Sign Up", "Shelter Search", and "Check In" screens of Shelter Navi as examples.

**4.2 Login**

The login screen of Shelter Navi is shown in Figure 5.

Resident users log in by entering their e-mail address and password in the form at the center of the screen. If the correct e-mail address and password are entered, the authentication is successful and the user moves to the shelter search screen. If the authentication is successful, the entity related to the user is granted access authorization. The authorization granted to a general resident user only authorizes access to his or her own entities, and thus his or her information cannot be viewed, updated, or deleted by other users.

If you have not registered your user information yet, click the "New Registration" button to go to the sign-up screen described in the next section. 4. 3 Sign-up

**4.3 Sign-up**

The sign-up screen is shown in Figure 6.

A resident enters his/her own user information in the form to register. An e-mail address is required to identify the user, and a password is required for authentication. They also enter their "personal information," which consists of their name, address, number of people in their household, and telephone number (optional). After entering the information, click the "Register" button at the bottom of Fig. 6 to complete the new user registration.

**4.4 Shelter Search**

The shelter search screen is shown in Figure 7.

In the evacuation center search screen, a map is displayed at the top of the screen with the blue marker as the current location. On the map, shelters in the vicinity of the current location (assumed to be within a radius of 1[km] in the prototype) are placed with numbered pins. The map is updated in real time using searchSheltersByDistance(), the API introduced in Section 3.5, to search for shelters in the vicinity. The colors of the pins are green (low), yellow (medium), and red (high), depending on the congestion level of the shelter. You can also click on the pin of the shelter to display detailed information about the shelter.

The list at the bottom of the screen shows the shelters on the map in order of distance from each other. The list consists of four items from left to right: the number of the marker on the map, the name of the shelter, the address of the shelter, and the congestion status. Finally, the search bar at the bottom of the screen allows the user to search for shelters by shelter ID, name, and address.

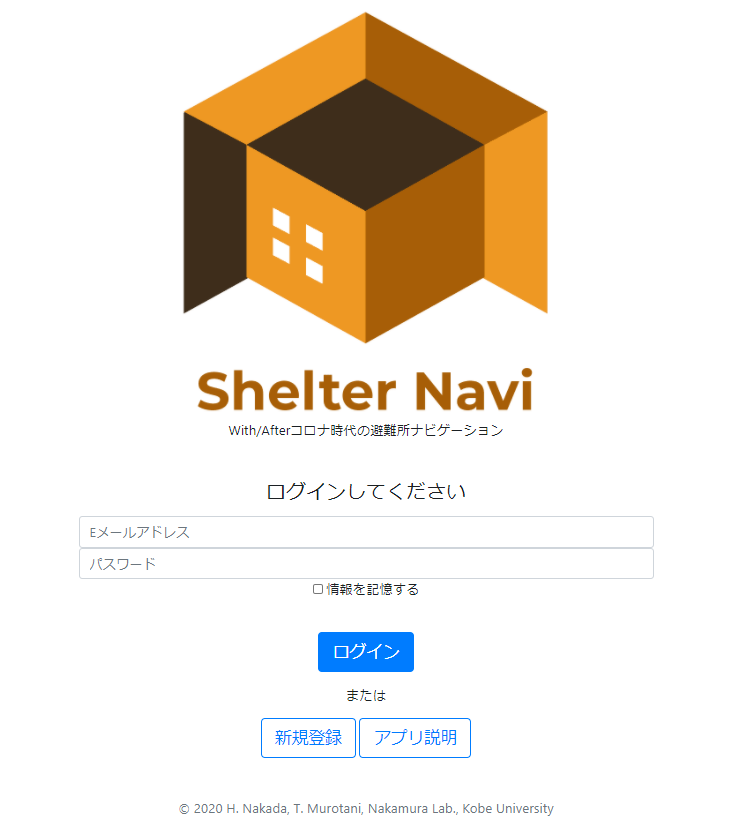


Figure 5 the login screen of Shelter Navi Figure 6 the sign-up screen of Shelter Navi



Figure 7 the shelter search screen of Shelter Navi Figure 8 the check-in screen of Shelter Navi



Figure 9 the check-out screen of Shelter Navi

**4.5 Check-in**

The check-in screen is shown in Figure 8.

On the check-in screen, you can select the shelter you want to check in from either "current location," "area," or "map. When selecting the "current location," click the blue "Get current location" button at the top of the screen to select a shelter in the current location. In the case of selecting from "Area", select a prefecture, municipality, or region from the pull-down menu in the "Select from Area" section in the center of the screen, and the list of shelters corresponding to the condition will appear in the pull-down menu. When selecting from the "Map", click the red marker on the map at the bottom of the screen to select an shelter. After selecting a shelter by either of these methods, click the "Check-in" button at the bottom right of the screen to complete the check-in.

The check-out screen is shown in Figure 9. If you have already checked in to one of the shelters, the lower left corner of the checkout screen displays the sentence "Do you want to check out from the current shelter: '\*Checked-in shelter name'? is displayed at the bottom left of the checkout screen. By clicking the "Check out" button at the bottom right of the screen, you can check out from the currently checked-in shelter.

**5 Results and Discussion**

**5.1 Effects and merits of Shelter Navi**

The following three effects and merits can be obtained by the realization of Shelter Navi.

The first is that residents can find shelters with enough capacity on their own. This will help to solve the research question of this paper, "Can residents evacuate to an appropriate shelter by themselves without relying on the local government?"

The second is that the congestion level of the evacuation centers is automatically calculated and updated by the system based on the residents' check-in. This eliminates the need to assign a full-time staff member to manage the crowdedness of each shelter.

Third, local government officials can check the status of all registered shelters in real time. Since the status of each evacuation center can be centrally and cross-sectionally managed by Shelter Navi, it reduces the workload of local government officials and enables them to take actions for evacuation center management that were difficult in the past, such as safety confirmation and dispersed evacuation instructions.

**5.2 Limitations of Shelter Navi**

There are three limitations of Shelter Navi that we are aware of at present.

The first is that accurate shelter information needs to be registered. Specifically, master information such as the latitude and longitude of evacuation centers, the number of people who can be accommodated, and the opening status of evacuation centers must be registered and maintained during normal times, and this requires the cooperation of local governments.

The second is that people who are not used to operating smartphones cannot operate Shelter Navi well. Since Shelter Navi is an application designed to be used with smartphones, it is expected to be difficult for elderly people who are not familiar with the operation of smartphones or who do not own smartphones to use this application.

Third, the application is dependent on the mobile communication network. If the mobile communication network is interrupted during a disaster, Shelter Navi will not work because the server cannot be accessed.

**6. Conclusions**

In this study, we proposed a mobile application, Shelter Navi, which helps residents to evacuate quickly and safely by avoiding the formation of crowded shelters through self-help, assuming a disaster in the corona age. By using a cloud server to manage and visualize the congestion status of shelters, and by allowing residents to search for and check in at shelters, it is possible to evacuate avoiding 3Cs without requiring special facilities at each shelter. In addition, we defined the use cases of Shelter Navi, designed the domain model and services, and implemented the prototype as a web application.

The future task is to evaluate the effectiveness of the system under the assumption of actual evacuation. For this purpose, we would like to ask for cooperation from neighboring municipalities to prepare data on shelters, and then have residents try and evaluate the system during evacuation drills and other occasions. We also plan to enhance the functions of Shelter Navi. First of all, we plan to add a function for users to register and manage their private shelters (relatives' or friends' homes). Furthermore, we would like to add functions to improve the quality of evacuation life after check-in, such as information sharing and communication among evacuees.

**7. Acknowledgements**

This research was cooperated by Toshiki MUROTANI, Graduate School of Kobe University.

This research was supported by JSPS KAKENHI Grant Numbers JP19H01138, JP18H03242, JP18H03342, JP19H04154, JP19K02973，JP20K11059, JP20H04014, JP20H05706

**8. References/Bibliography**

[1] 国土交通省 “平成30年7月豪雨災害の概要と被害の特徴”

[2] 菊池聡, “非常時の思い違いと批判的思考” 日本科学教育学会年

会論文集, Vol.35, pp.9-10, 2011.

[3] 国土交通省 “住民自らの行動に結びつく災害情報の提供へ ~危機感が伝わる、メディアとの連携策をとりまとめ~” 2018.12.11

[4] 皆川勝、中村遼太、高橋翔天 “極低頻度の災害に対する避難 行動の社会心理学的な考察” 土木学会論文集 F 6 Vol.72 No.2 I191 I198 2015.7.10

[5] 田中重好 “東日本大震災を踏まえた防災パラダイム転換” Vol.64 No.3 p.342-365 2013.

[6] 内閣府 “新型コロナウイルス感染症を踏まえた災害対応のポイント【第1版】について”

[7] 毎日新聞 “避難所「先着順で満員、2 回も振られる」台風 10 号、コロナ対策で収容人数の減少で” 2020.09.07

[8] 株式会社バカン “株式会社バカンと宮崎県日南市、災害発生時に IoT を活用して

避難所の混雑情報配信を支援する協定を締結” 2020.08.03

[9] Spring “Why Spring?” https://spring.io/why-spring