sandwico

User Manual: Real-time Fire Escape Route

by ERP

Mathilda Bresler
u16313382@tuks.co.za

Pieter Braak
u16313382@tuks.co.za

Kateryna Reva
u17035989@tuks.co.za

Jason Louw
u16313382@tuks.co.za

Xiao Jian Li
u16099860@tuks.co.za

12 May 2019

University Of Pretoria, Hatfield
Engineering, Built environment and Information Technology

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Contents

1 System Overview

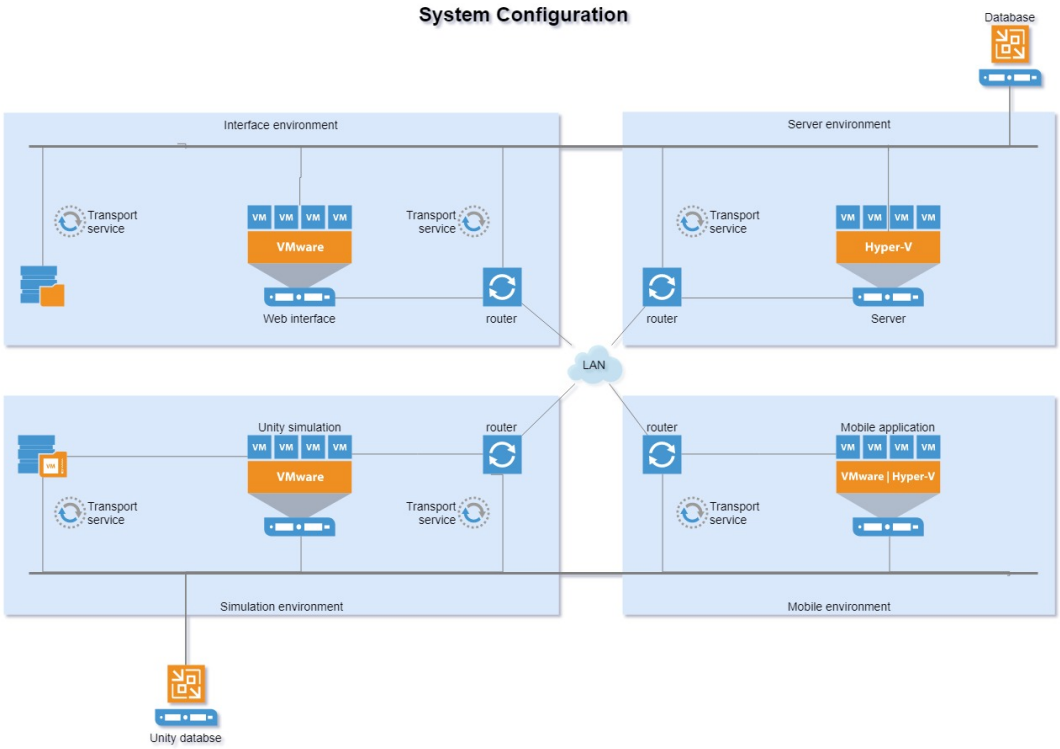
The FireWatch System is a new approach to solve an age old problem. The interface is in the form of mobile application as well as desktop application. The main goal of FireWatch is to indicate to the agent using the application what the most efficient route would be to take in the case of an emergency. The system will perform this functions by anonymously tracking the building populations to be aware of the building's current state. It will be done with the use of a heatmap, which is generated using the Bluetooth sensors installed throughout the monitored location. This will show the quantity and distribution of the population throughout the building.

The escape routes are pre-planned by the building designers, since there are special considerations that need to be made, including which walls have fire proofing, and which areas should be avoided for safety reasons. The heatmap is then to be consumed by an AI algorithm to assign a fire escape route to each agent in the building. The building's population distribution and available routes must be taken into consideration when assigning a route to each individual.

With the building heatmap and time sensitive fire escape route in place, each agent in the building will be sent a push notification to his/her phone indicating what escape path to follow in case of emergency. A real time fire escape route system:

- A software system based on a windows server system.
- Android application and web application interface.
- Bluetooth sensor locating.
- System name: *FireWatch System*
- System category:
  - *Major application*: performs clearly defined functions to achieve the system goal of providing evacuation from building at real time.
  - *General support system*: provides network support for a variety of users and applications on multiple platforms.
- Operational status:
  - Partially Operational
  - Under development
- Types of users:
  - Back-end users: Users responsible for maintenance, upkeep and general operation of the system.
  - Administrative users: Users responsible for managing the users of the system and the information access of various types of users.
  - Agents: Users that interact with the main interface of the application and are the main intended demographic of the application. These are the users that will be using the main functionality of the system.

2 System Configuration



2.1 Sensor configuration

(Unknown) Currently we do not have access to any sensors and can not explain how configurations of the sensors will take place.

2.2 Changing simulation parameters

(Planned) You can change which building will be used by the system in the simulation by changing settings in the browser. You will also be also be able to change how many people are in the building.

3 Installation

3.1 Installing the Application

Once a user is registered on the system they will be emailed details of their login details and a link to download the app. Alternatively for development and testing purposes the app can also be downloaded from our Git Page using this link: https://github.com/cos301-2019-se/Real-time-Fire-Escape-Routes.git

3.2 Installing the system locally

The system can be installed on a local machine by downloading and executing the jar file availible from this link https://github.com/cos301-2019-se/Real-time-Fire-Escape-Routes.git. If using this method you can run the system on any operating system.

3.2.1 Setting Up Unity during development stage

- To run the simulation in unity you will need to download the unity engine from: https://unity3d.com/get-unity/download
- Please ensure you download the Unity 2019 version.
- Once that is complete you can you can create a git repository from our our GITHUB branch Unity: https://github.com/cos301-2019-se/Real-time-Fire-Escape-Routes.git
- Once you have unity installed and have cloned the github unity branch you will open unity and choose open project and choose the branch you just downloaded.
- You can then press the play button and it will trigger and api call to the server setup in 3.2 and the simulation will play out.

3.2.2 Setting Up Unity as a final product

- The final unity simulation will be exported as a functional executable that will not require prerequisites or game engines to run.A Simple double click will open the application and it will run the simulation, This will be part of future demos.

4 Getting Started

Before a user can access the system they must be registered by an admin in the browser the system will them email them their credentials along with a link to download the latest version of the application.

Take note if no admin is registered on the system yet a developer will need to add that user to the system so that he/she may add more users on their own.

Once a user has been added to the system they can download the application and log in to the application. This gives them access to the main functionality of the system. When an alarm is triggered the system responds by performing the necessary calculations and then pushing the relevant information to the applications installed and holding active user accounts.
Once the user receives the information from the application they respond accordingly.

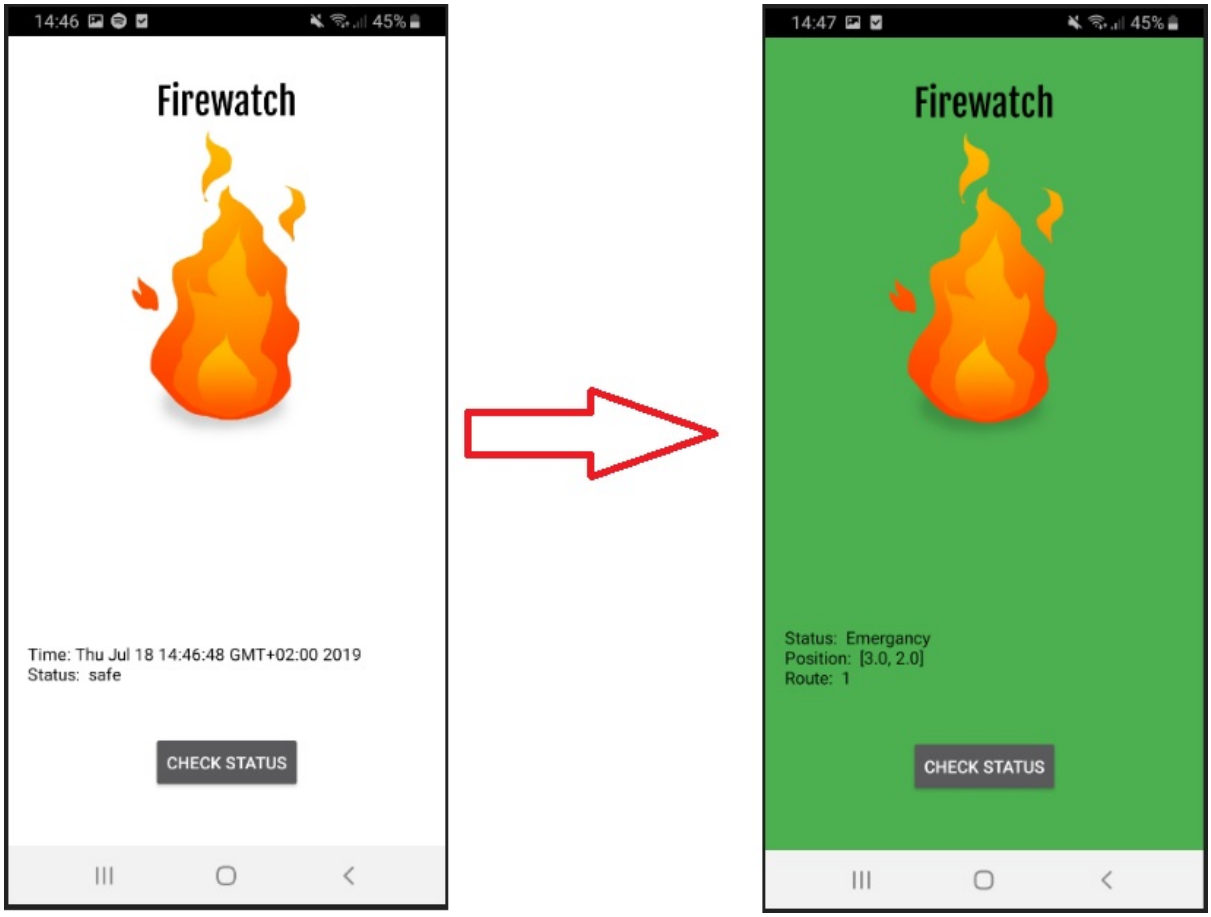
After the system has fully executed the evacuation procedure the data is backed up and the system is re-set, clearing the data on the application and reverting it to it's original state.

The application will exit once the user is removed from the database, this functionality is still to be implemented.

5 Using the system

5.1 UC1: Pushing notifications to the application

- The server is alerted to an emergency: This consists of either a sensory trigger or a manual trigger. This uses the function assignPeople.
- Processing of the data: The system performs a set of tasks to react to the emergency using the data from the sensors which are in the form of *X, Y coordinates*.
  - The sensors send the location data of each *Agent* to the system in the form of *X, Y coordinates*.
  - There is a function *addPerson* that uses the sensory data to find the room that each person is located in.
  - The rooms are then recursively traversed and each person in the room is assigned a route based on a heuristic calculation.
  - The system then checks to see if the building has been evacuated at certain time intervals.



5.2 UC2: Building a map

- Sending the data to the server: A post request is made to the HTTPServer containing the data as a Json string in the format

```
{                                                                    1
        "type":"unity",                                              2
        "floors":[                                                   3
                {                                                    4
                        "floor":0,                                   5
                        "corners":[                                  6
                                [0,0],[0,5.8],[4,10],                7
                                [24,10],[24,1.7],                    8
```

```
                                [17,1.7],[17,0]
                        ]
                }
        ],
        "halls":[
                {
                        "floor":0,
                        "corners":[
                                [0,4.5],[0,5.8],[24,5.8],
                                [24,4.5],[13.3,4.5],
                                [13.3,0],[12,0],[12,4.5]
                        ]
                }
        ],
        "rooms":[
                {
                        "floor":0,
                        "corners":[
                                [0,0],[0,4.5],[4,4.5],[4,0]
                        ]
                }
        ],

        "doors":[
                {
                        "floor": 0,
                        "type": "buildingExit",
                        "position": [12.6,0]
                }
        ],
        "people":[
                {
                        "floor": 0,
                        "id": 0,
                        "position": [2,2]
                },{
                        "floor": 0,
                        "id": 1,
                        "position": [2,2]
                }
        ]
}
```

- Processing the data: The server accepts the Json data and performs the following functions:
  - Data gets delagated to the BuildingGenerationAPI using the function "HandleReqeust(JSONObject request)"
  - Then the API makes use of the Builder Design Pattern to create the various parts of the building and add them into the final building object which contains the various functions for making escapes possible
  - The Builder Pattern makes use of a Director called "BuildingManager" that creates the appropriate concrete builders for each part of building's data that was sent throught the request.
- After the map has been constructed and saved the server can use this data to perform the necessary calculations.
- An appropriate success message is then returned to the calling function.

```
{
```

```
    "status": true,                                    2
    "msg": "Building built successfully"               3
}                                                      4
```

5.3 UC3: Sensor emergency Trigger

This Use case is currently being done manually through the back-end and will be explained when it occurs automatically or through the front end.

5.4 UC4: User (Agent) logging in to the mobile application

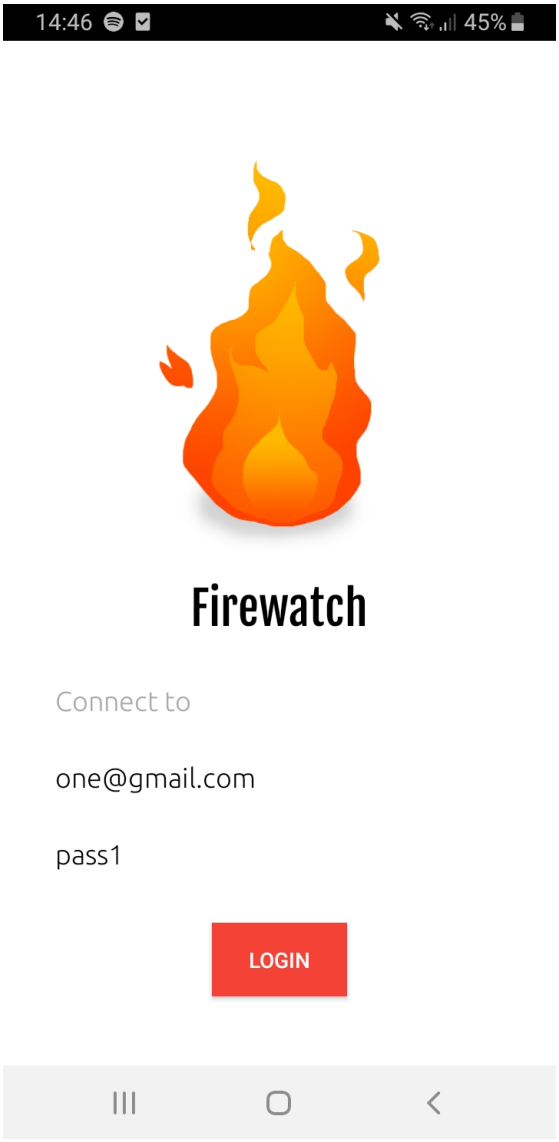
- Accessing the application: for that agent requires to have the FireWatch application to be installed on the device.
- Inputting user details: The user inputs their personal login information into the appropriate fields on the login page.
- Authenticating the user: The information is pushed from the application to the system using a post request which sends the data in the format:

```
{                                                      1
        "type": "login",                               2
        "email": "<email>",                            3
    "password": "<password>"                           4
}                                                      5
```

, where the information is then compared to the information stored on the database using the function *login(name,pass)* . If the user exists in the database and their provided information is correct they logged in, and an appropriate success message is then returned to the mobile application in the format.

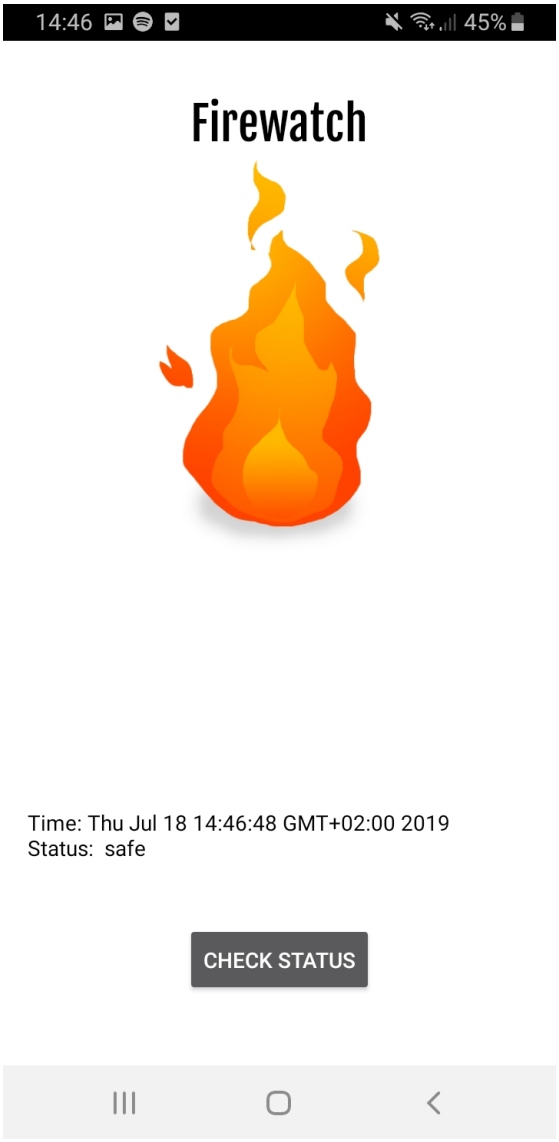
```
{                                                      1
    "status": true,                                    2
    "msg": "Login success"                             3
}                                                      4
```

The application then responds to this information by displaying welcome page and will now be on stand-by-mode (running on the background).



5.5 UC5: Viewing status of system on app

- Once Logged in the Application Queries the server on an interval basis. The interval is currently changing due to still being in development
- On each interval the application post the time of the last successful query time in case the user is unsure if they are getting updates.
- If the system has an emergency the Application's background will change to the color of the route assigned to the person. as shown in UC1: pushing notification to the app.



5.6 UC6: Administrative login to the system

- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Inputting user details: The user inputs their personal login information into the appropriate fields.
- Authenticating the user: The information is pushed from the web browser application to the system using the function *login(String name, String pass)* this then uses a post request, sending the data as a Json string in the format:

```
{                                                    1
    "type": "login",                                 2
        "email": "<email>",                          3
        "password": "<password>"                     4
}                                                    5
```

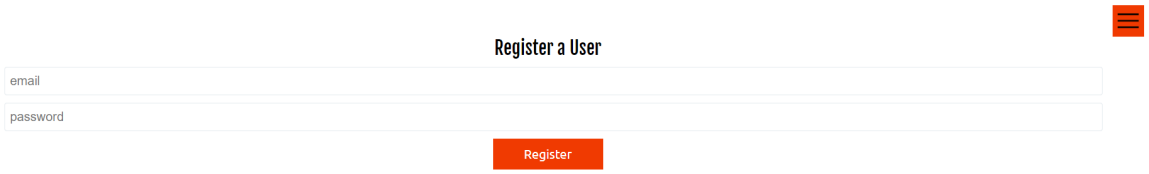
, where this information is then compared to the information stored on the database using the function *search(name,pass)* contained on the HTTPServer. If the user exists in the database and their provided information is correct the user is logged in and a success message is returned from the server in the format

```
{                                                    1
    "status": true,                                  2
```

```
    "msg": "Login success"
}
```

5.7 UC7: Registering a new user on the system

- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Administrative login: The user goes through *Use case 6* to gain necessary access to the system.
- Inputting the new user information: The administrative user navigates to the page where they input the data for the new user (Agent) who is to have access to the mobile application system in



- Registering the new user on the database: The web application makes a post request to the HTTPServer containing the relevant user information as a Json string in the format:

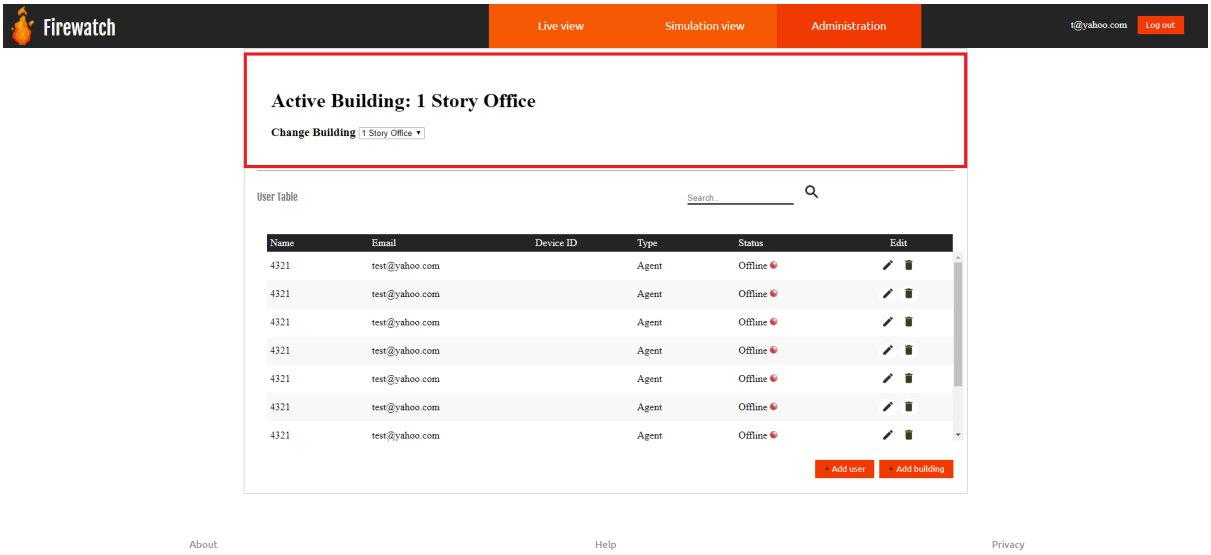
```
{
        "type": "register",
        "name": "<name>",
        "email": "<email>",
        "pass": "<password>",
        "userType": "<userType>"
}
```

. The information is then processed through the function *write(name,pass)* and stored in the database. An appropriate success message is then returned to the web application in the format.

```
{
    "status": true,
    "msg": "User Successfully created"
}
```

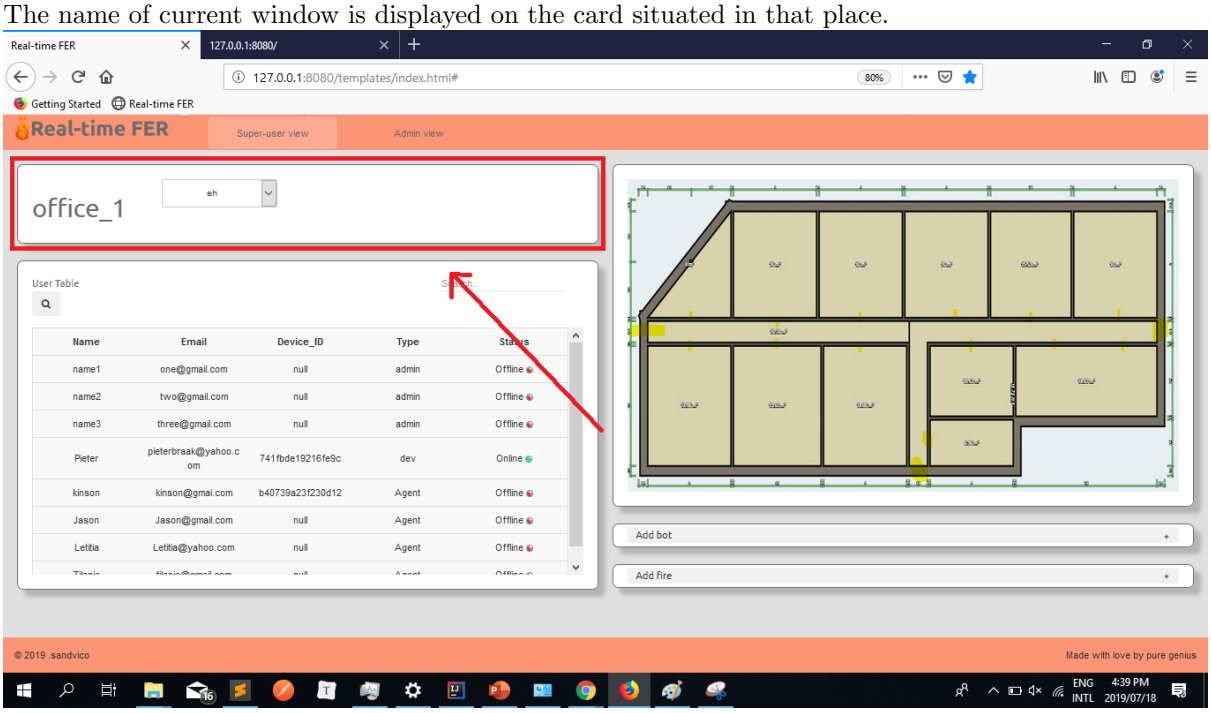
5.8 UC8: Change building selected

- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Administrative login: The user goes through *Use case 6* to gain necessary access to the system through the WebUI.
- On the top left corner of the default page you can find the card with currently chosen building name on it. On the right side of the name there is a dropdown list that contains all the buildings that were saved in the system.
- To choose another building press on the dropdown list and find and click on the building name that you would like to change to.

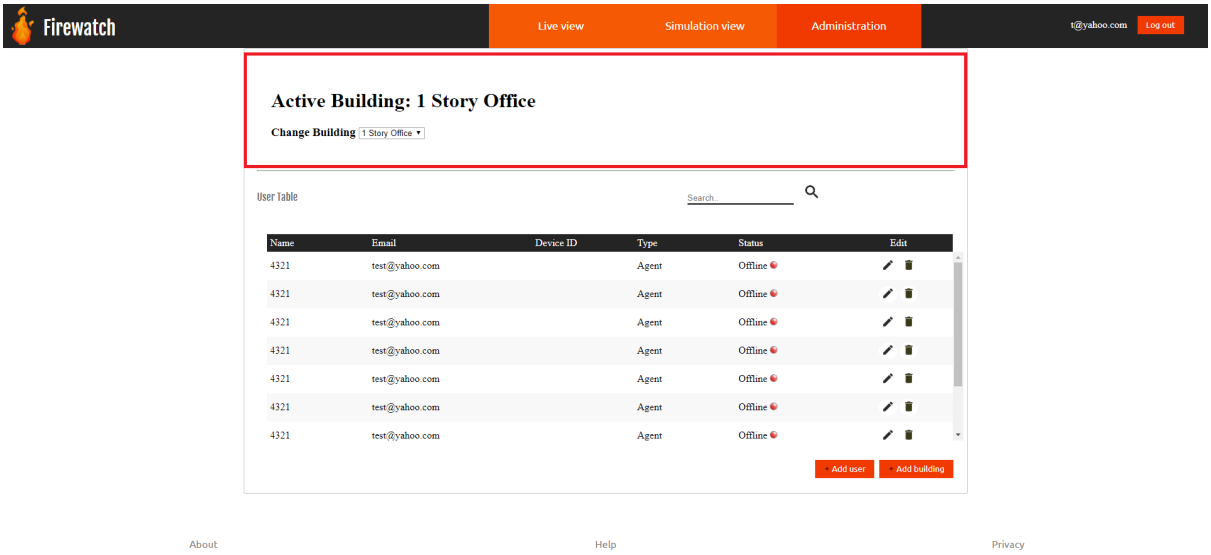


5.9 UC9: View building statistics

- The system fetches the information of the currently active building and displays on the Super-user view.
- To view current building name: on the default page observe the top left corner of the default page. The name of current window is displayed on the card situated in that place.

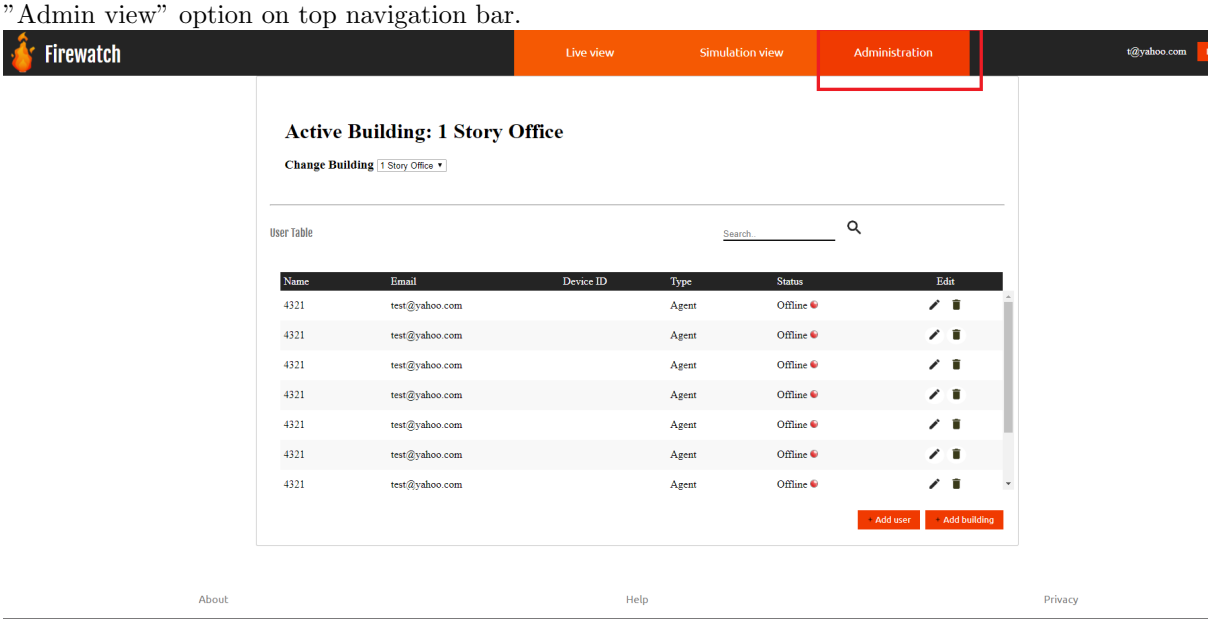


- To view building plan: on the default page observe the right side big window. The window displays the image of the plan of currently selected building.

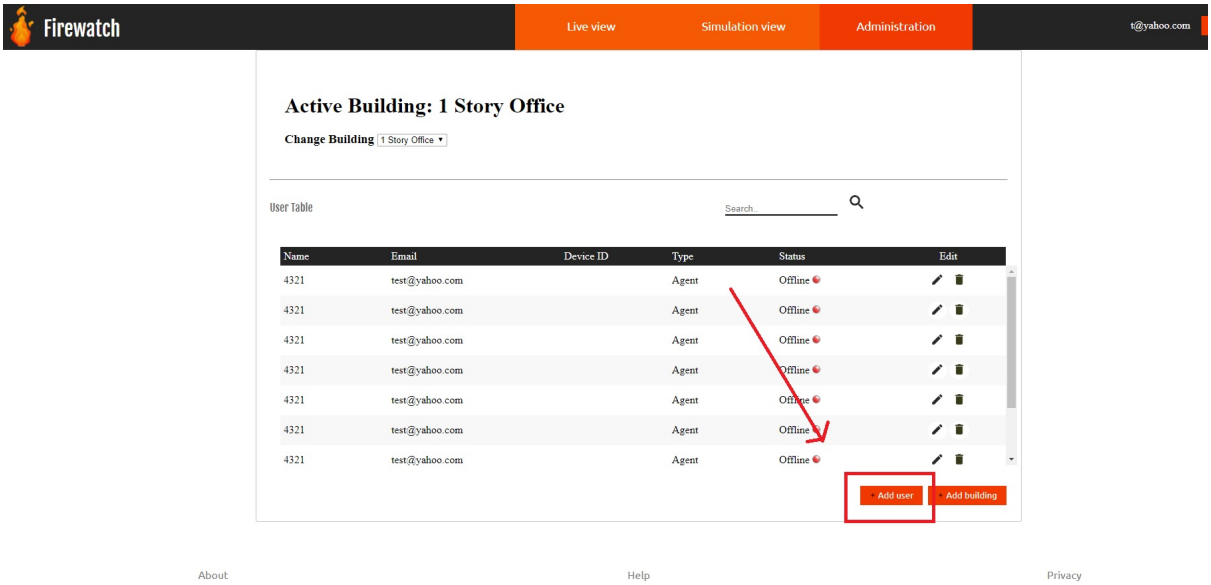


5.10 UC10: Adding users to building

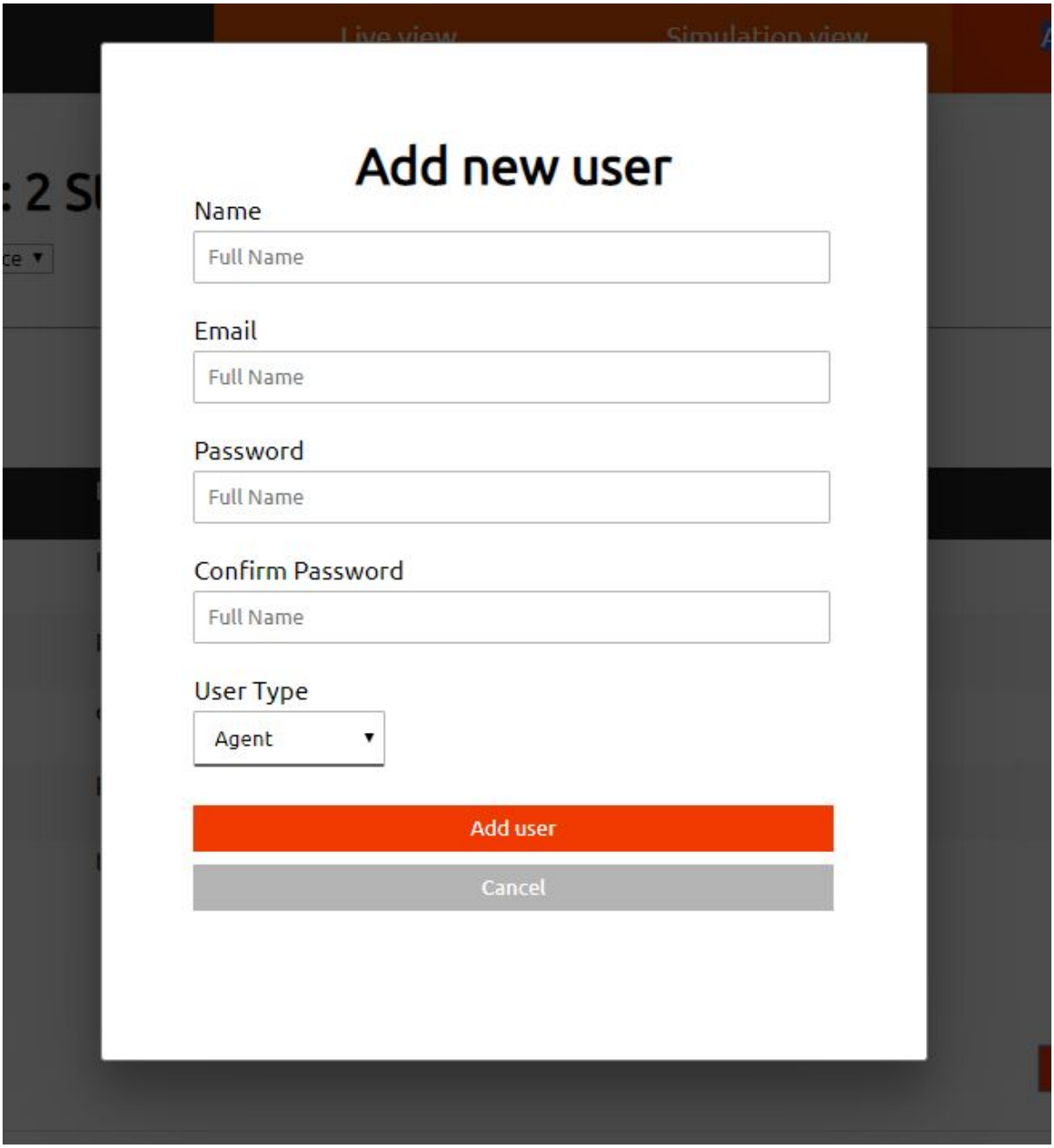
- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Administrative login: The user goes through *Use case 6* to gain necessary access to the system through the WebUI.
- In order to add new agent/user to the building one must switch to the correct view: for that choose "Admin view" option on top navigation bar.



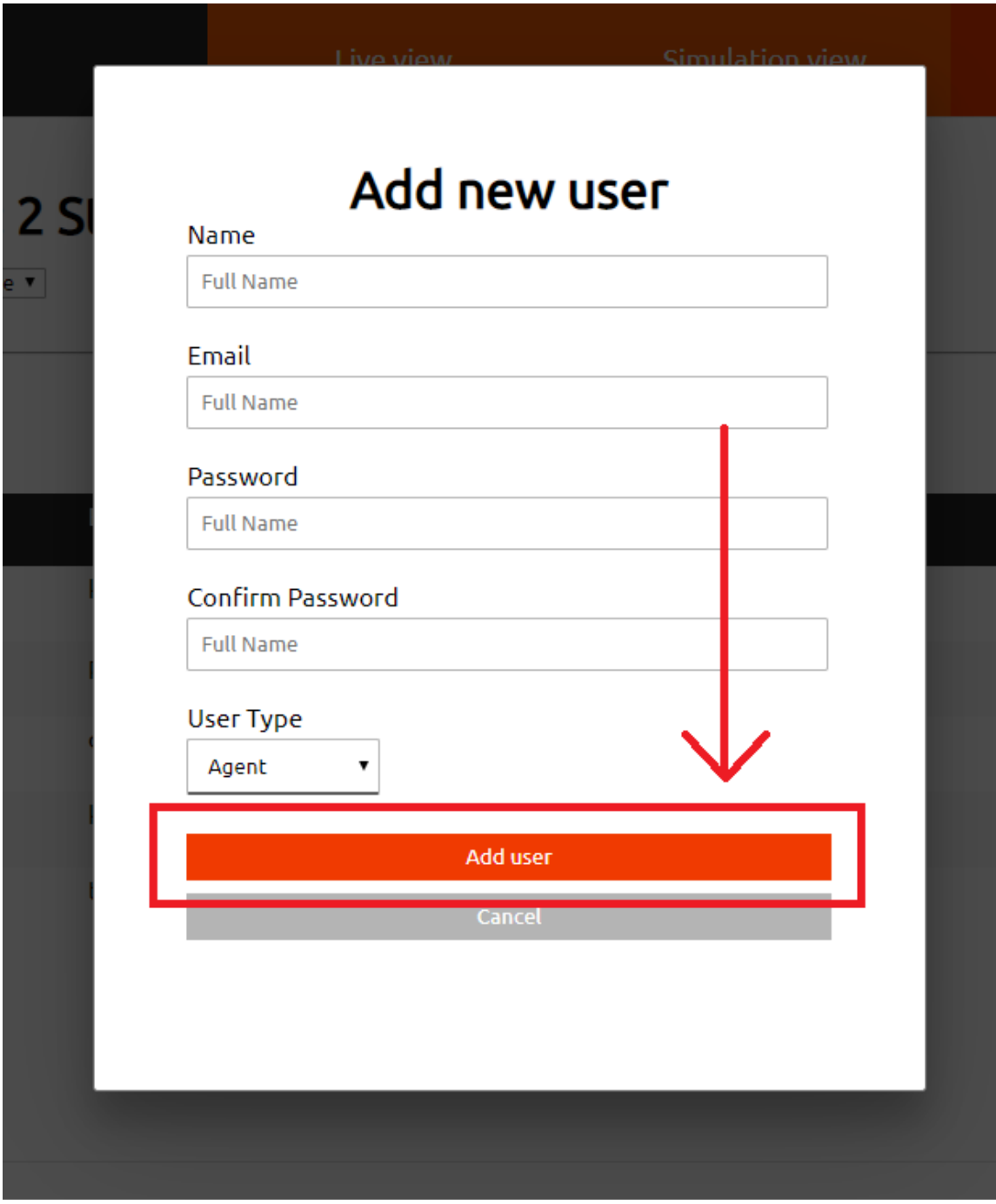
- Open window to fill in information about new user: for that click on the "Add user" button on the bottom of "Admin view" page.



- Fill in required information: for that input all open fields on the displayed view.

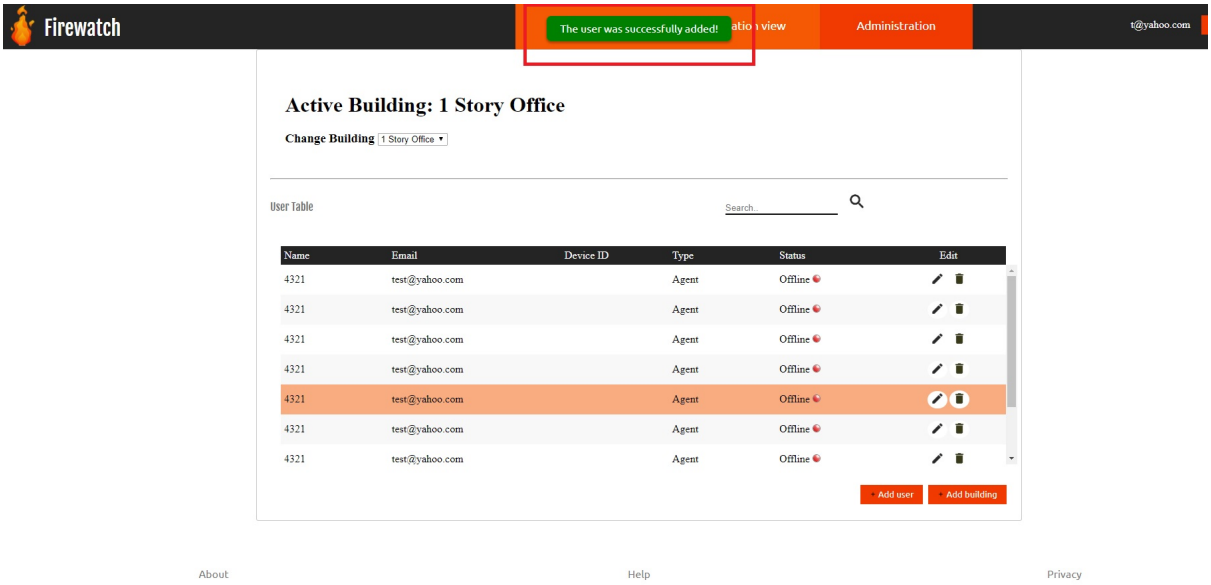


- Save new user to the system: after completing all the steps mentioned above you should press on the "Save" button situated in the bottom-right corner of the currently displayed "Add new user" window.
- In case if you changed your mind and you do not want to add new user to the system anymore you can cancel it by pressing on "Cancel" button that is situated on the right side of "Send Request" button.



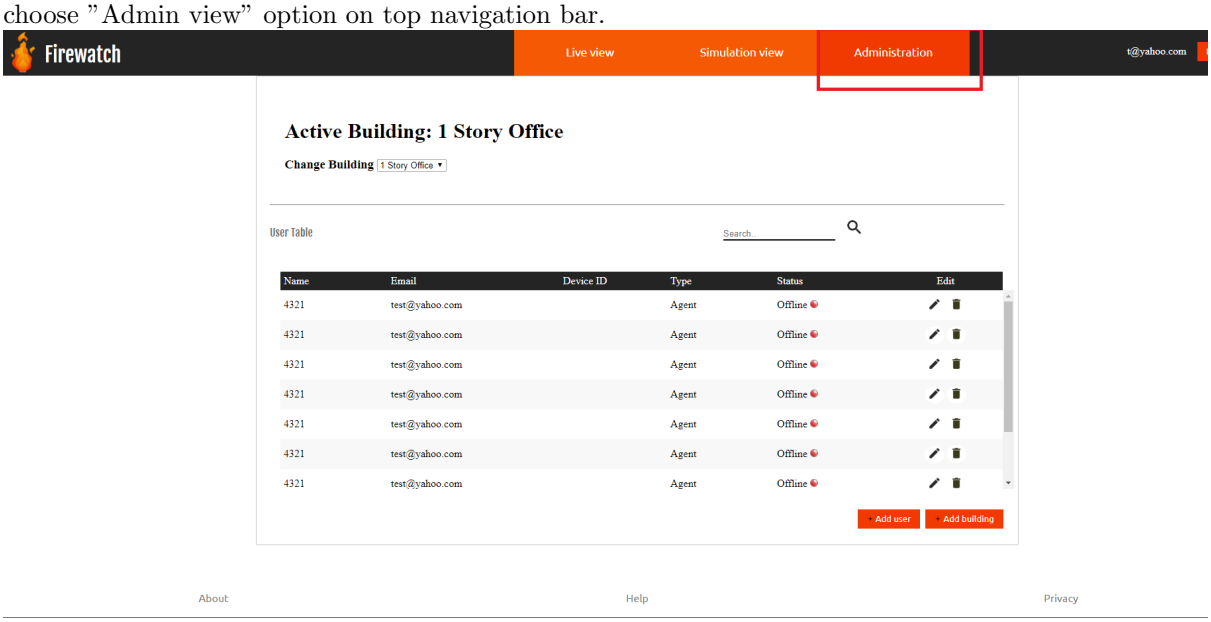
user button.png

- System feedback: the moment you will press on "Send Request" button, "Add new user" window will close automatically and you will see notification from the system that will notify you if the user was added or not added to the system.

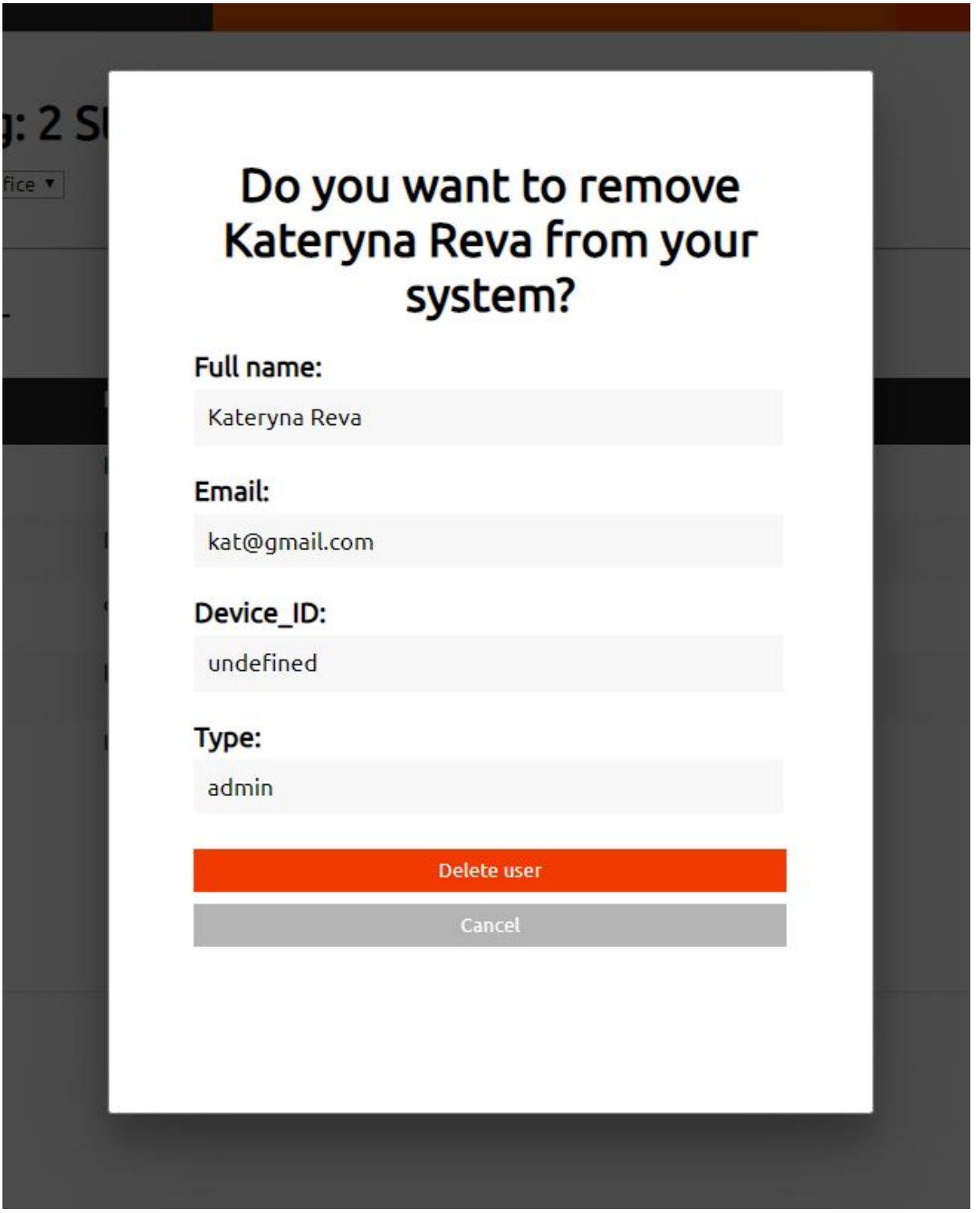


5.11 UC11: Removing users from building

- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Administrative login: The user goes through *Use case 6* to gain necessary access to the system through the WebUI.
- In order to remove agent/user from the building one must switch to the correct view: for that choose "Admin view" option on top navigation bar.

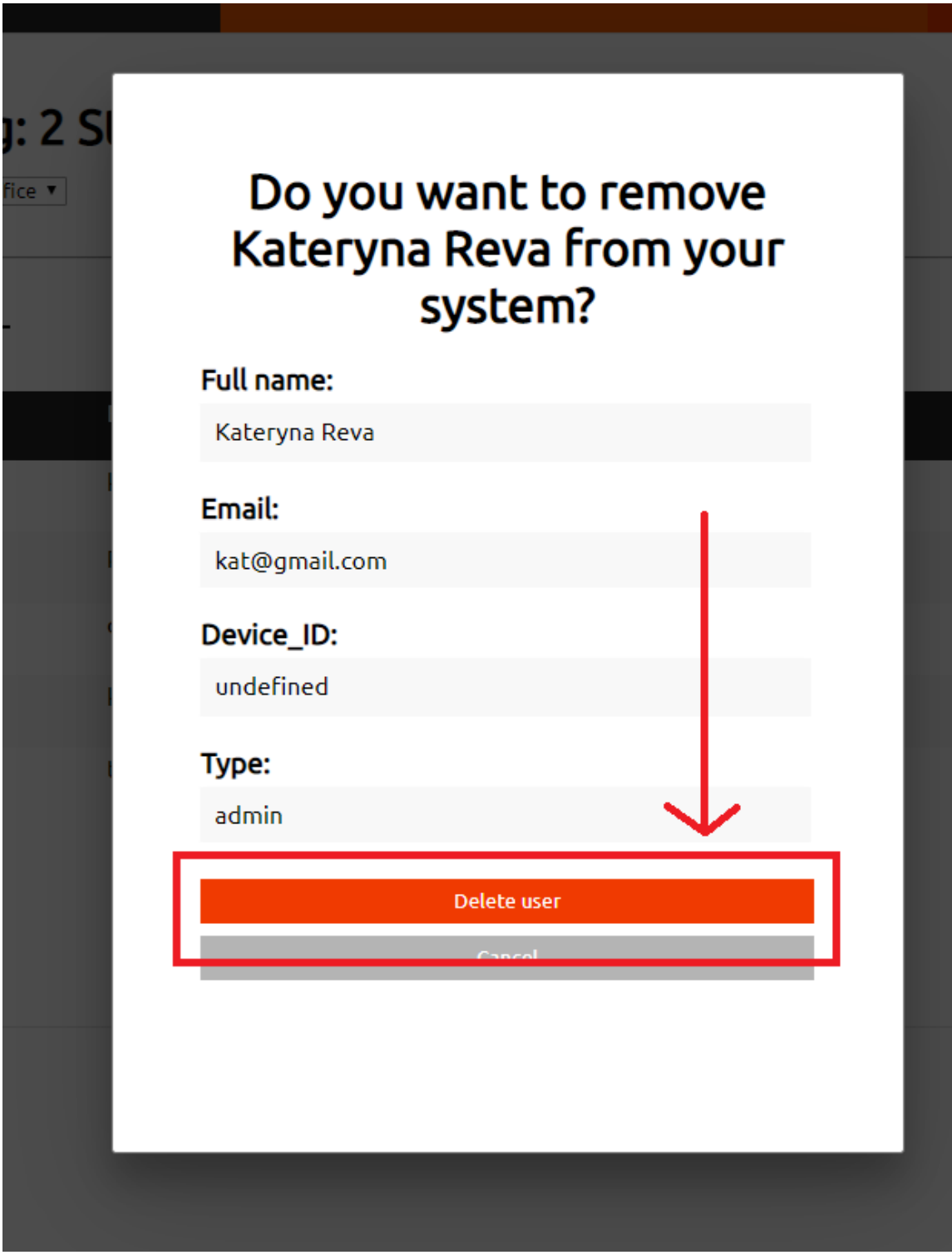


- Find the user you want to delete: for that search the user in "User table" that is displayed in the middle of the page.



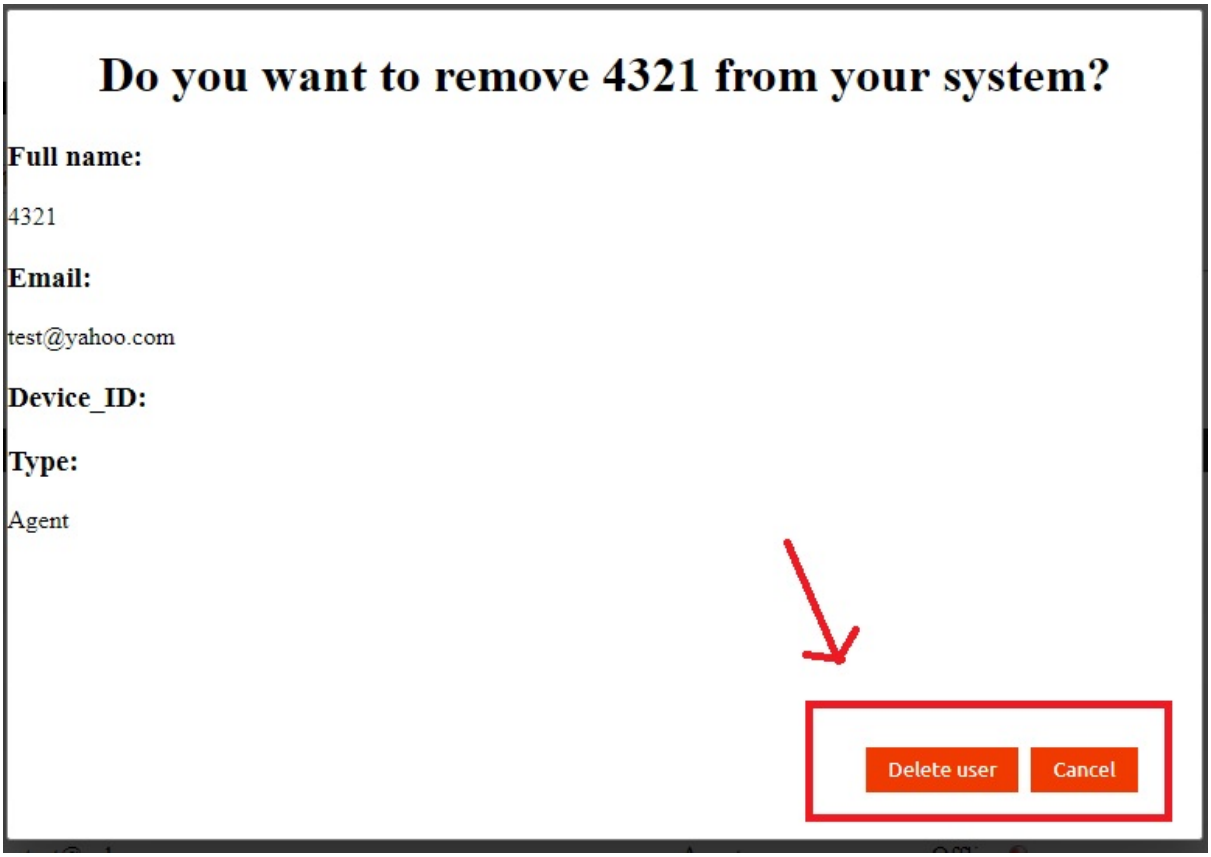
user.JPG

- Press on the delete icon(in the form of dustpan) that is situated on the same row where user that you are willing to delete is displayed. You will get a window that asking you if you would really want to delete that user.

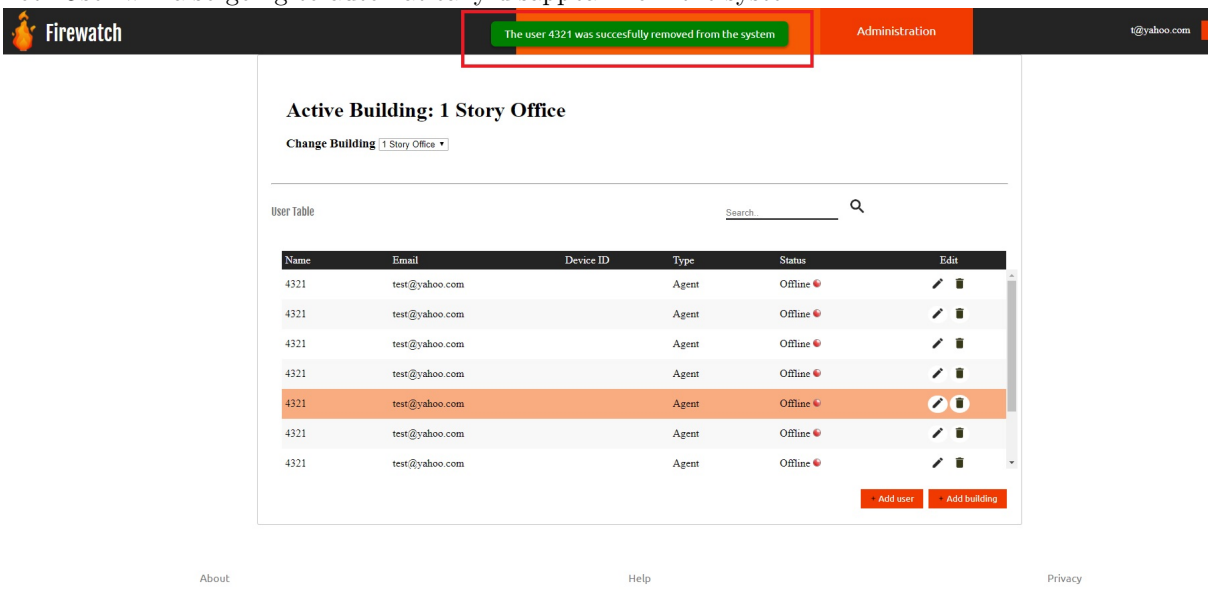


user button.png

- If you sure you want to delete that user press "Delete user" butter, if you do not want to delete user press on "Cancel" button.



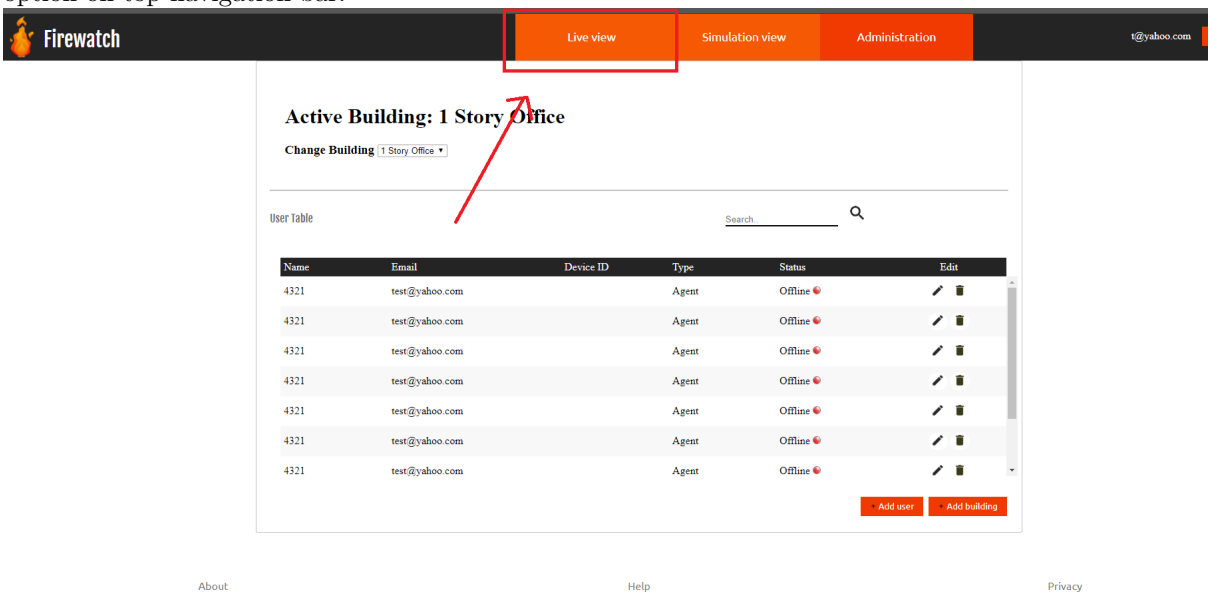
- System feedback: the moment you will press on "Delete user" button, you will see notification on top of the screen from the system that will notify you if the user was deleted from the system or not. User will also going to automatically disappear from the system.



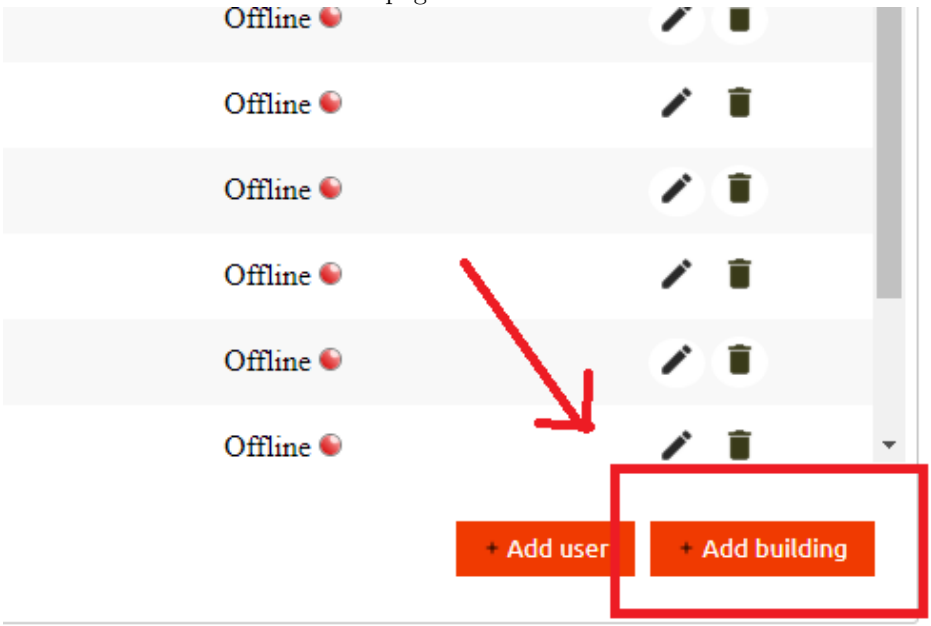
5.12 UC12: Upload building

- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Administrative login: The user goes through *Use case 6* to gain necessary access to the system through the WebUI.
- In order to add new building one must switch to the correct view: for that choose "Admin view"

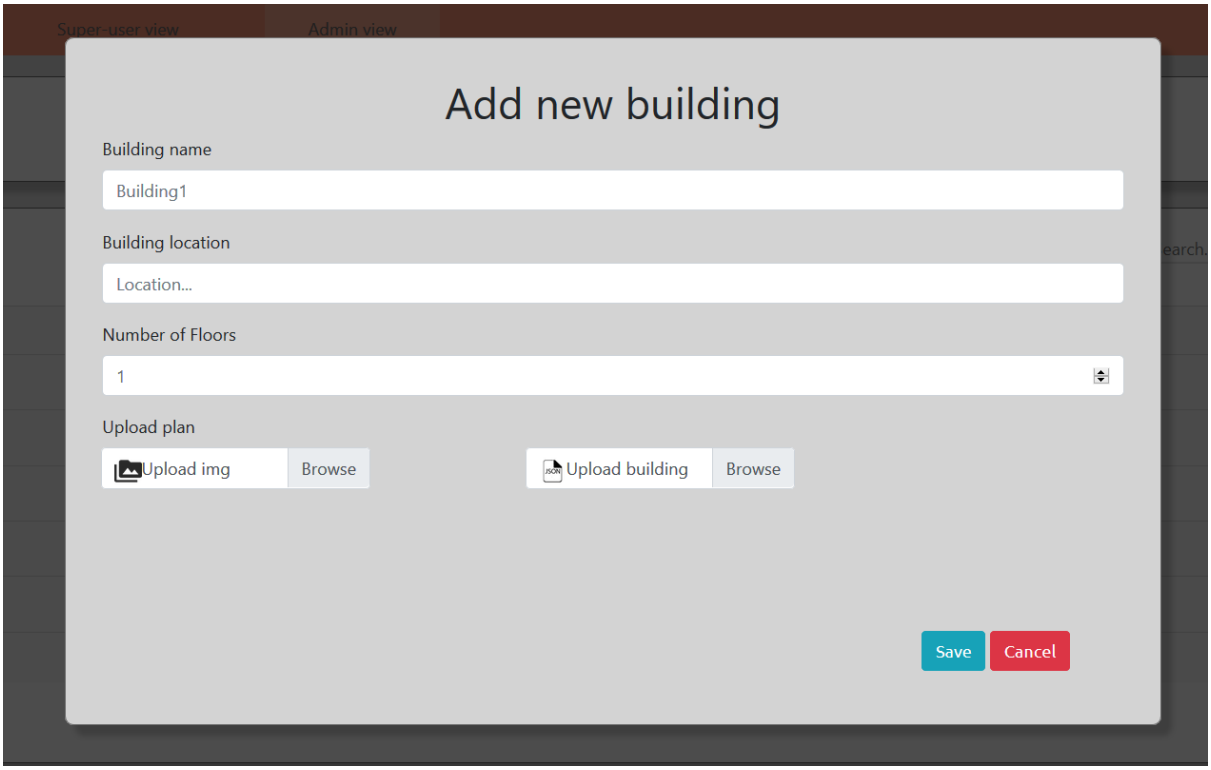
option on top navigation bar.



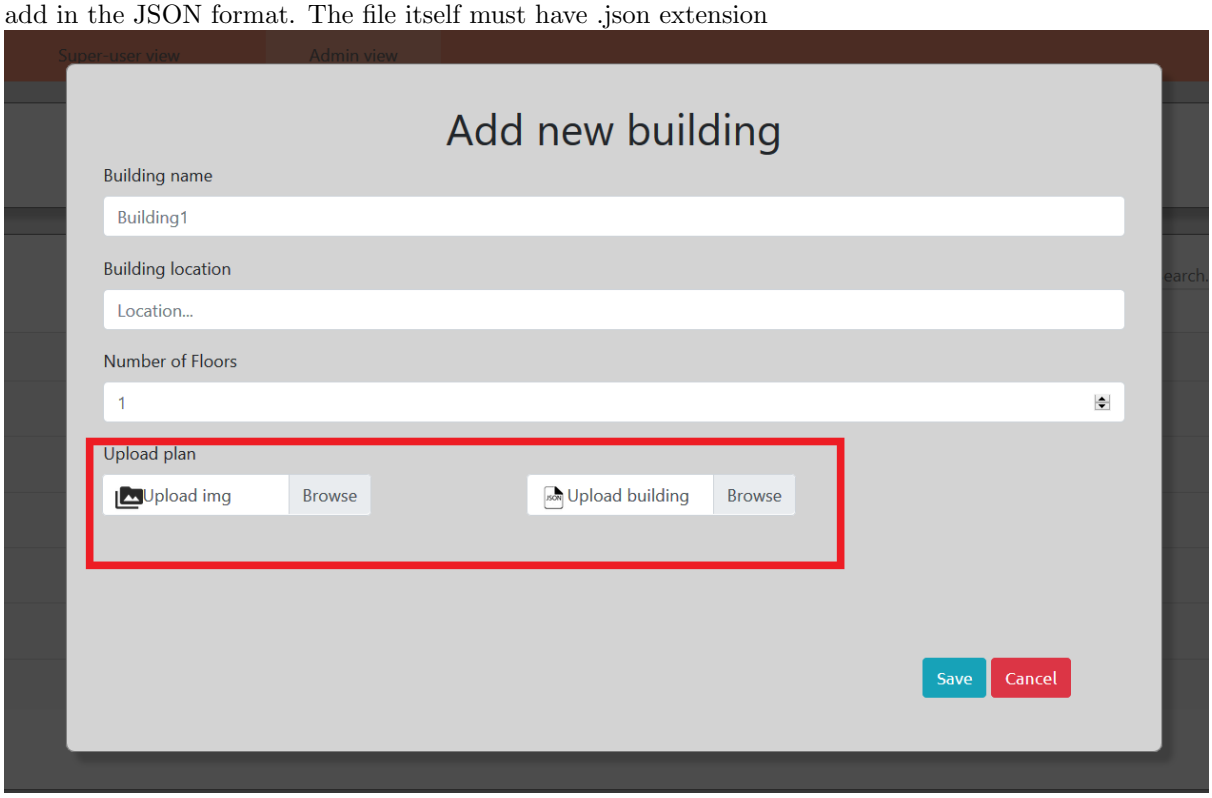
- Open window to fill in information about new building: for that click on the "Add building" button on the bottom of "Admin view" page.

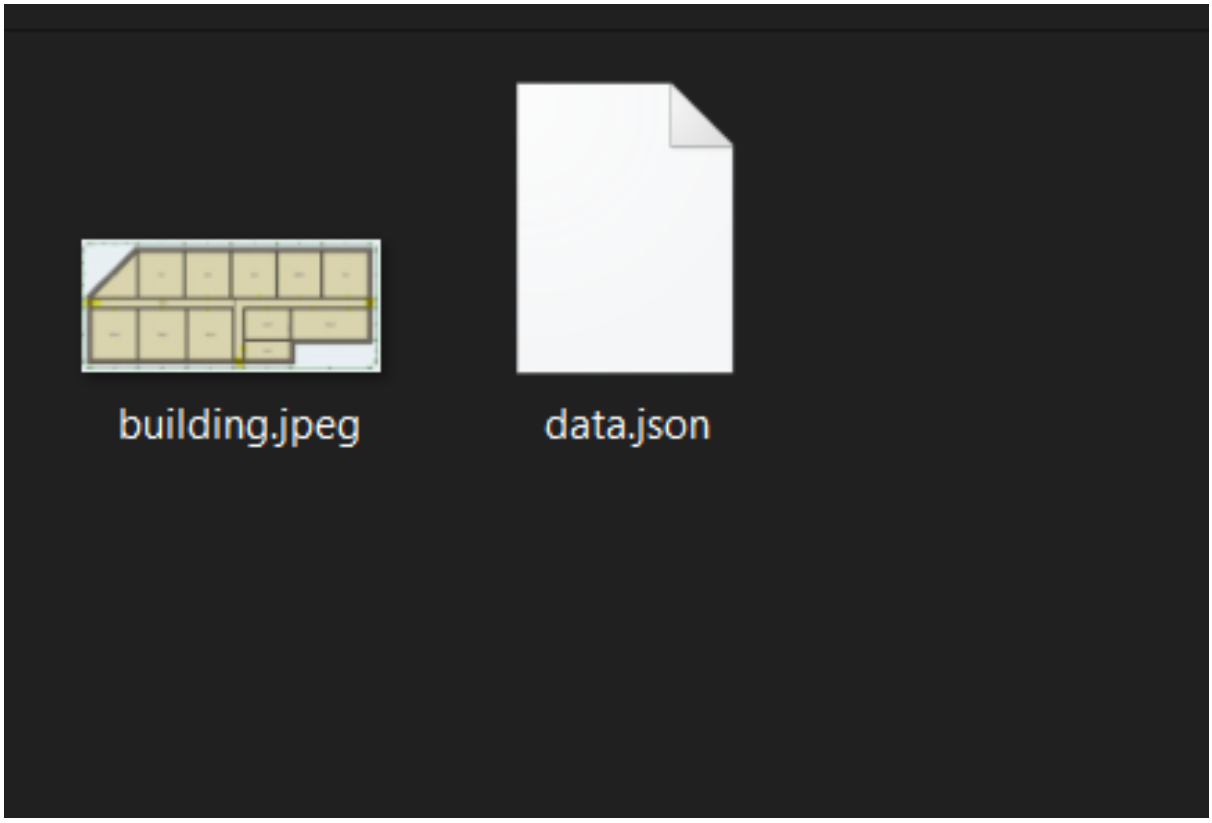


- Fill in required information: for that input all open fields on the displayed view.

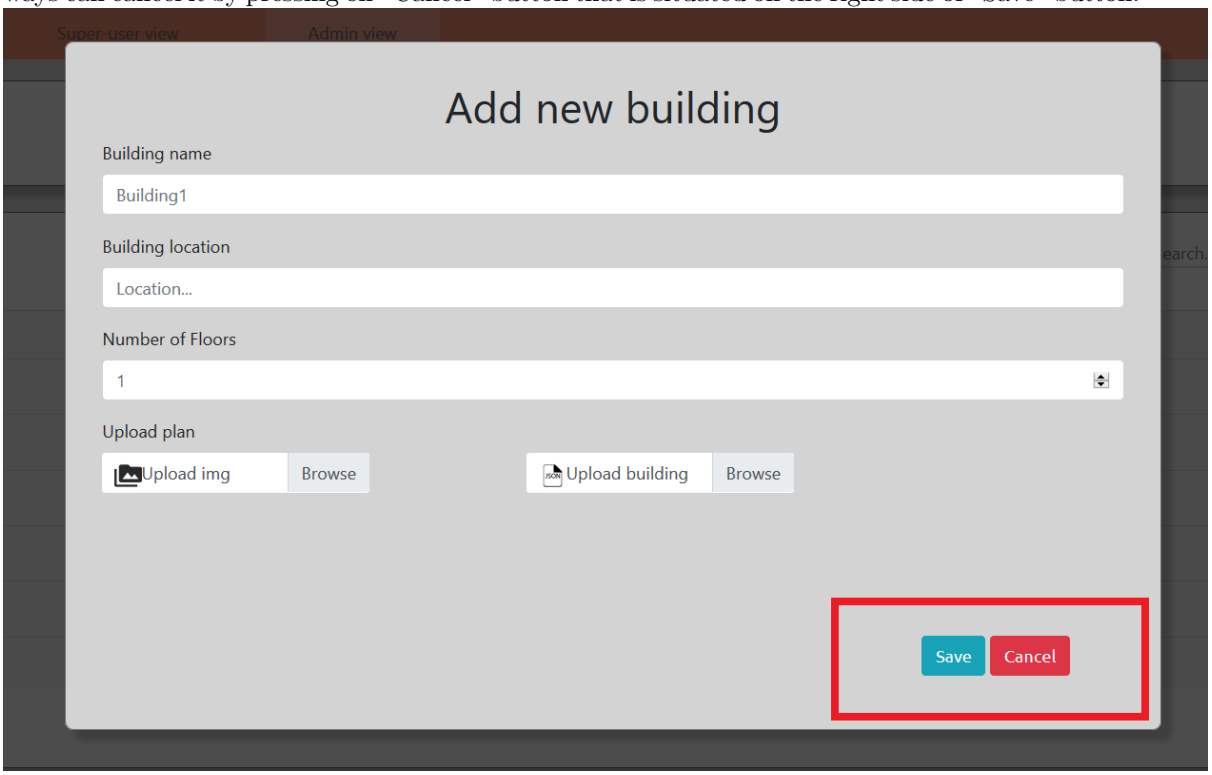


– Make sure to upload required files: for that, under "Upload plan" section, click on "Upload img" field and choose the image of the type "jpg" or "jpeg". The image must contain the plan of the building you want to add. The image name should be "building.jpg" or "building.jpeg". Than click on the "Upload building" field and choose building information in the JSON format. For that it is required for you to have information about the building you would like to add in the JSON format. The file itself must have .json extension

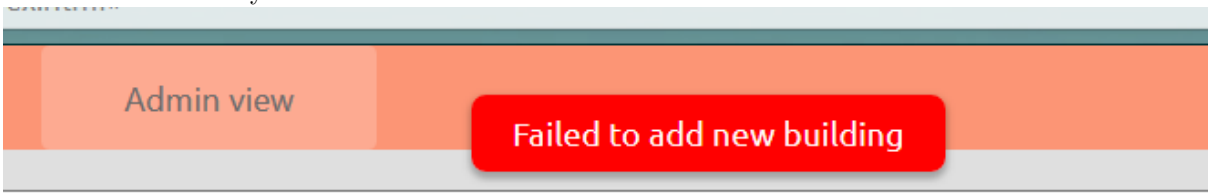




- Save new building to the system: after completing all the steps mentioned above you should press on the "Save" button situated in the bottom-right corner of the currently displayed "Add new building" window.
- In case if you changed your mind and you do not want to add new building to the system you always can cancel it by pressing on "Cancel" button that is situated on the right side of "Save" button.

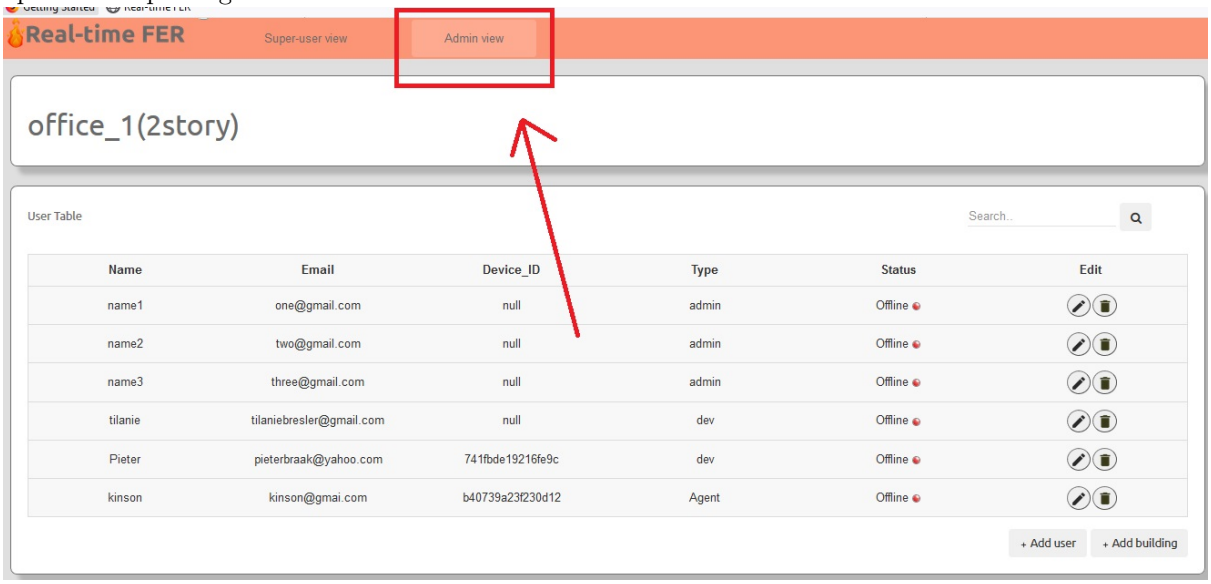


- System feedback: the moment you will press on "Save" button, "Add new building" window will close and you will see notification from the system that will notify you if the building was added or not added to the system.

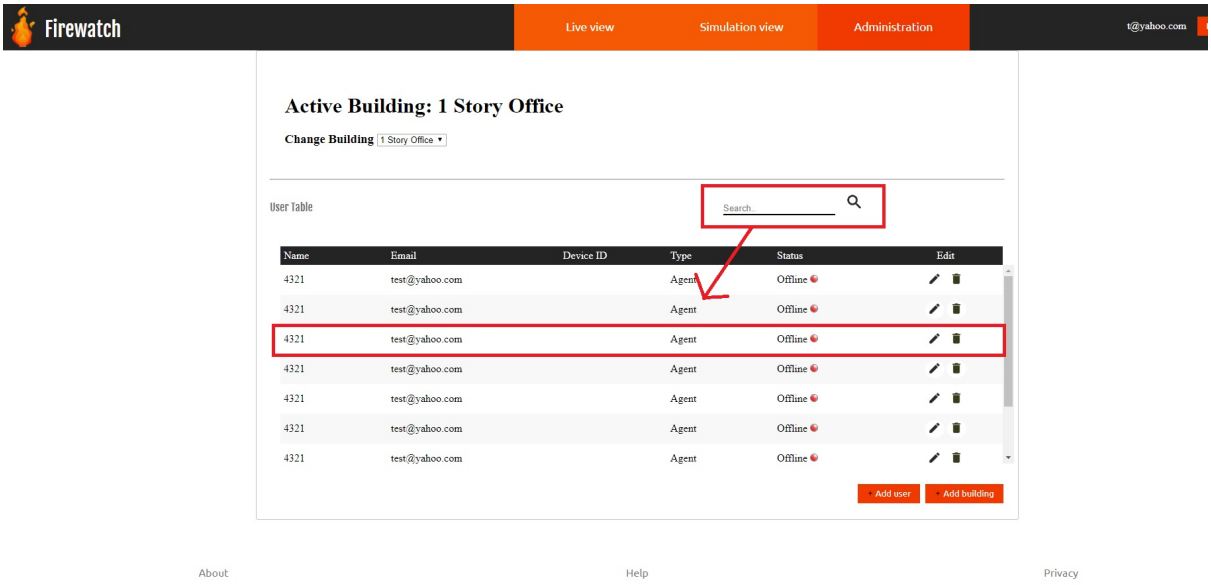


5.13 UC13: Promote user

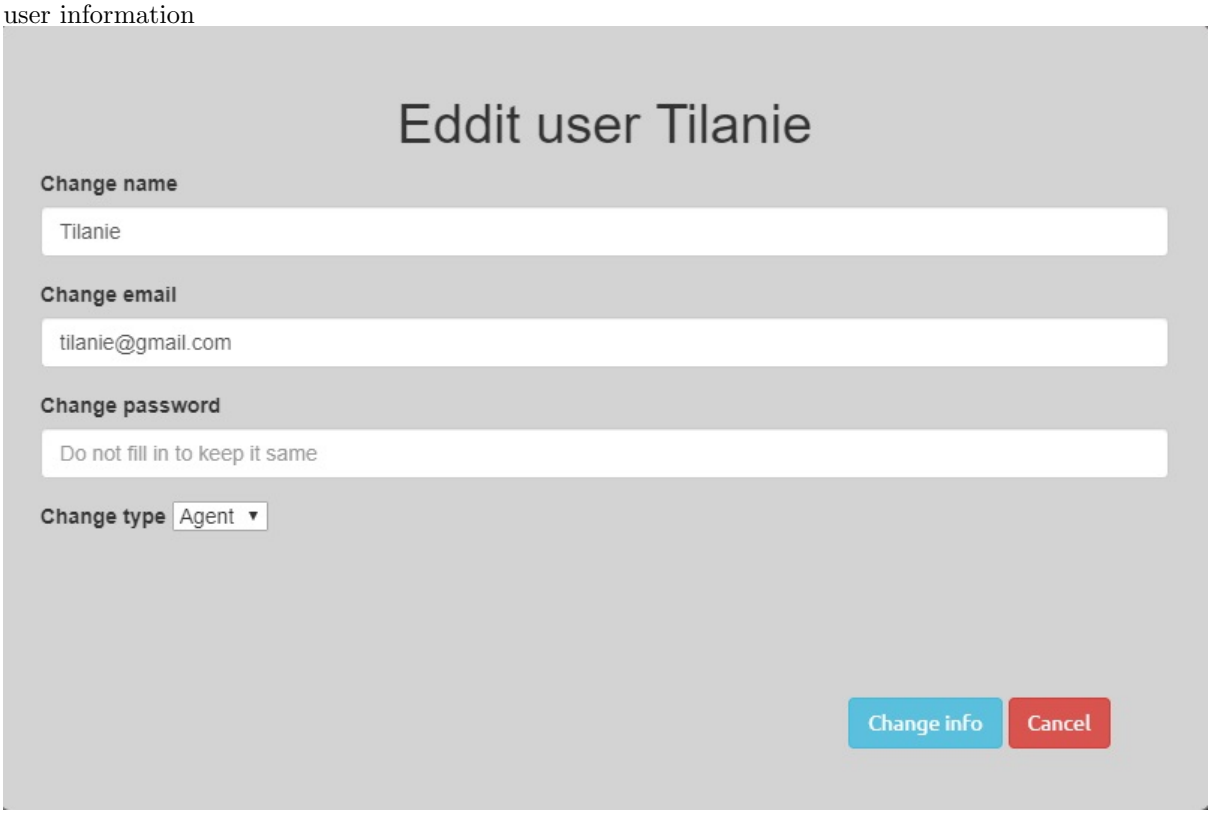
- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Administrative login: The user goes through *Use case 6* to gain necessary access to the system through the WebUI.
- In order to promote agent/user one must switch to the correct view: for that choose "Admin view" option on top navigation bar.



- Find the user you want to promote: for that search the user in "User table" that is displayed in the middle of the page.

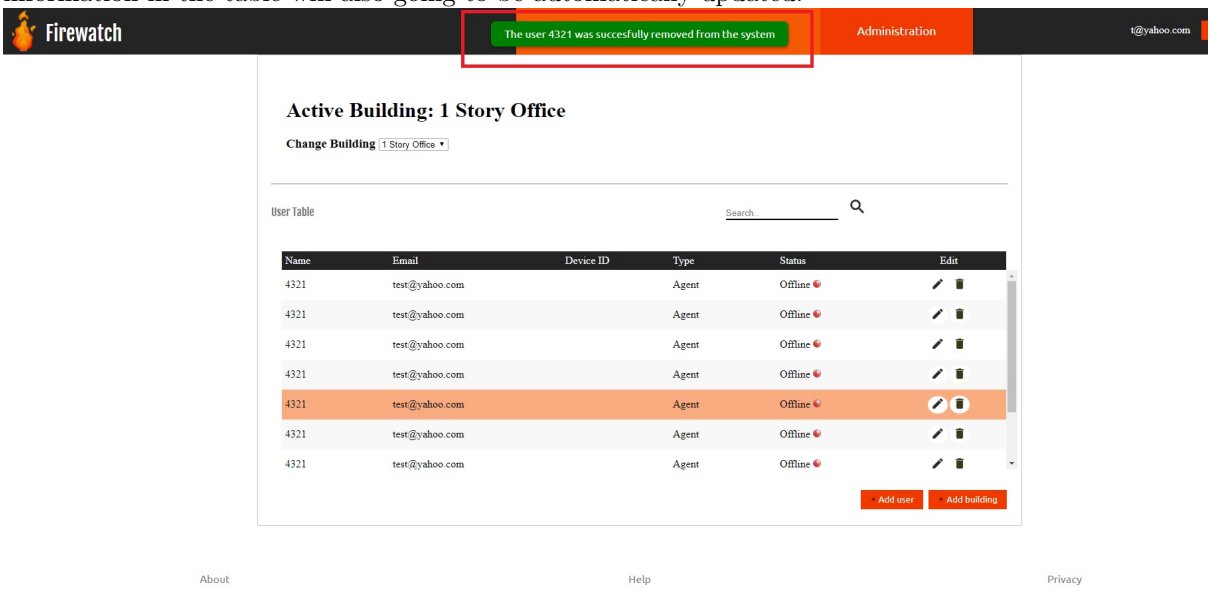


- Press on the edit icon(in the form of pan) that is situated on the same row where user that you are willing to promote is displayed. You will get a window with fields that are filled with current user information



- If you want to change current information about the user just delete what is written in the specific field you would like to change and type in information you want to add.
- To save changes: just press on the "Change info" button situated in the bottom-right corner of the currently displayed "Edit user" window.
- In case if you changed your mind and you do not want to change user's information you can cancel it by pressing on "Cancel" button that is situated on the right side of "Save" button
- System feedback: the moment you will press on "Save" button, you will see notification on top of the screen from the system that will notify you if the user's information was changed or not. User's

information in the table will also going to be automatically updated.



5.14 UC14: View all users

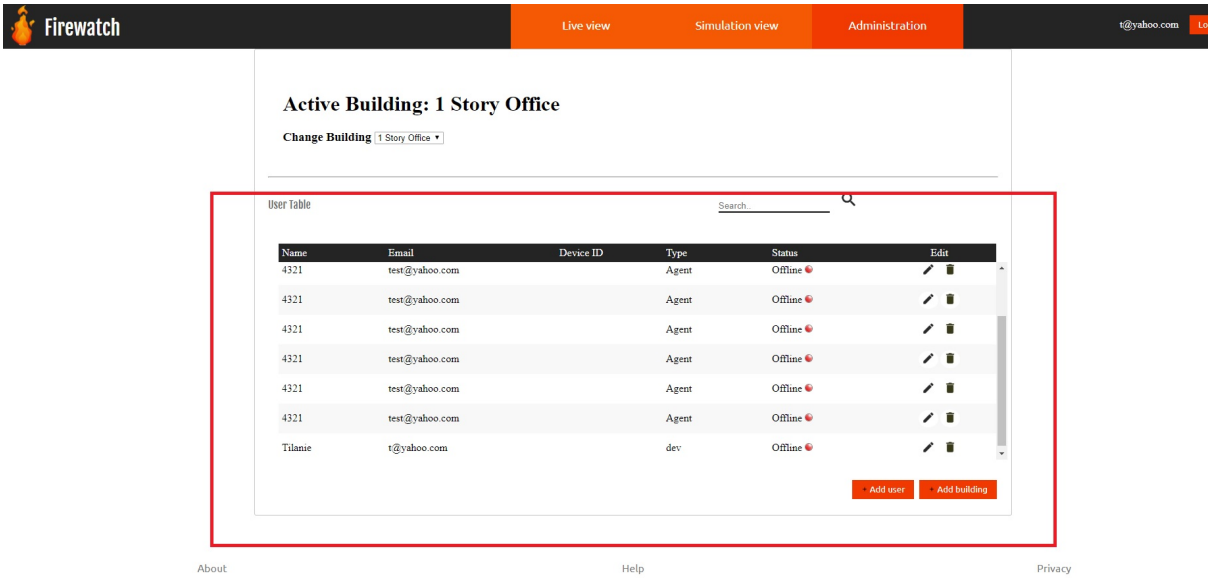
- Accessing the browser: Assuming the system is installed locally you can access the system via http://localhost:8080/ or enter the IP of the PC where the software is installed on.
- Administrative login: The user goes through *Use case 6* to gain necessary access to the system through the WebUI.
- Querying the information from the system: The web application makes a post request to the HTTPServer.

```
{                                                                    1
        "type": "getUsers"                                           2
}                                                                    3
```

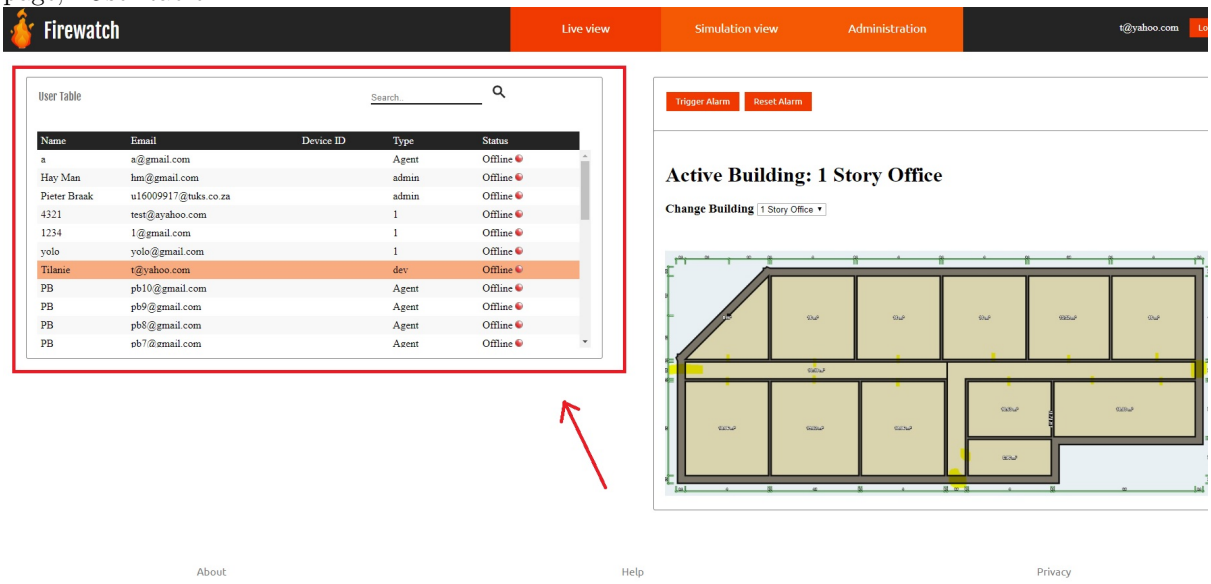
The information is then processed through the function *getUsers()* which in turn accesses the function *getUsers()* in the database subsystem. This function queries the database and selects the data from the database. The following Json structure is returned:

```
{                                                                    1
    "msg": "Users returned",                                         2
    "data": "[[0, one@gmail.com, name1, pass1, admin, null],         3
    [0, two@gmail.com, name2, pass2, admin, null]]",                 4
    "status": true                                                   5
}                                                                    6
```

- To observe all users with the use of the web site interface, one can do it by observing the left side of the page that was displayed by default; under "User table".

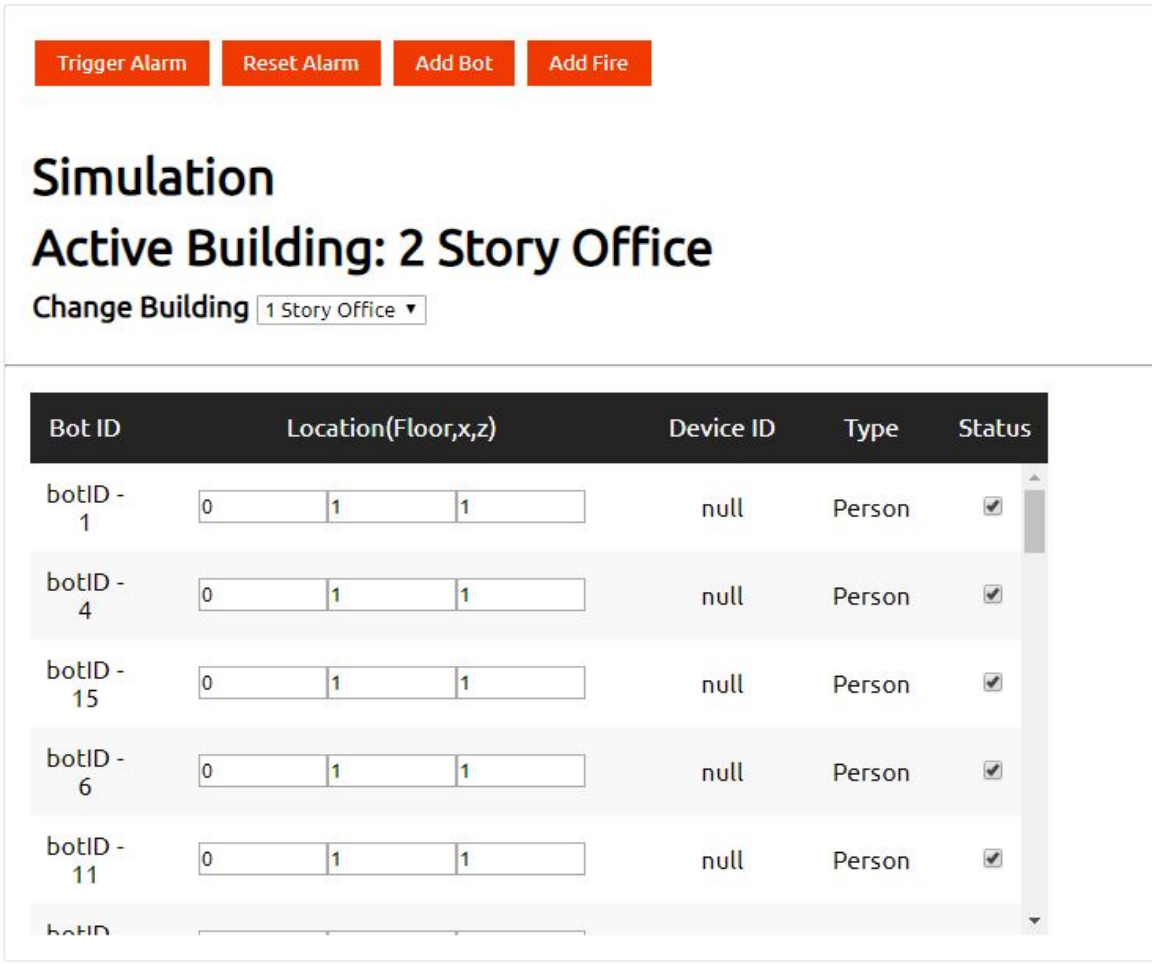


- One can also switch to "Admin view" and observe all users on the displayed, in the middle of the page, "User table".



5.15 UC15: Changing simulation parameters

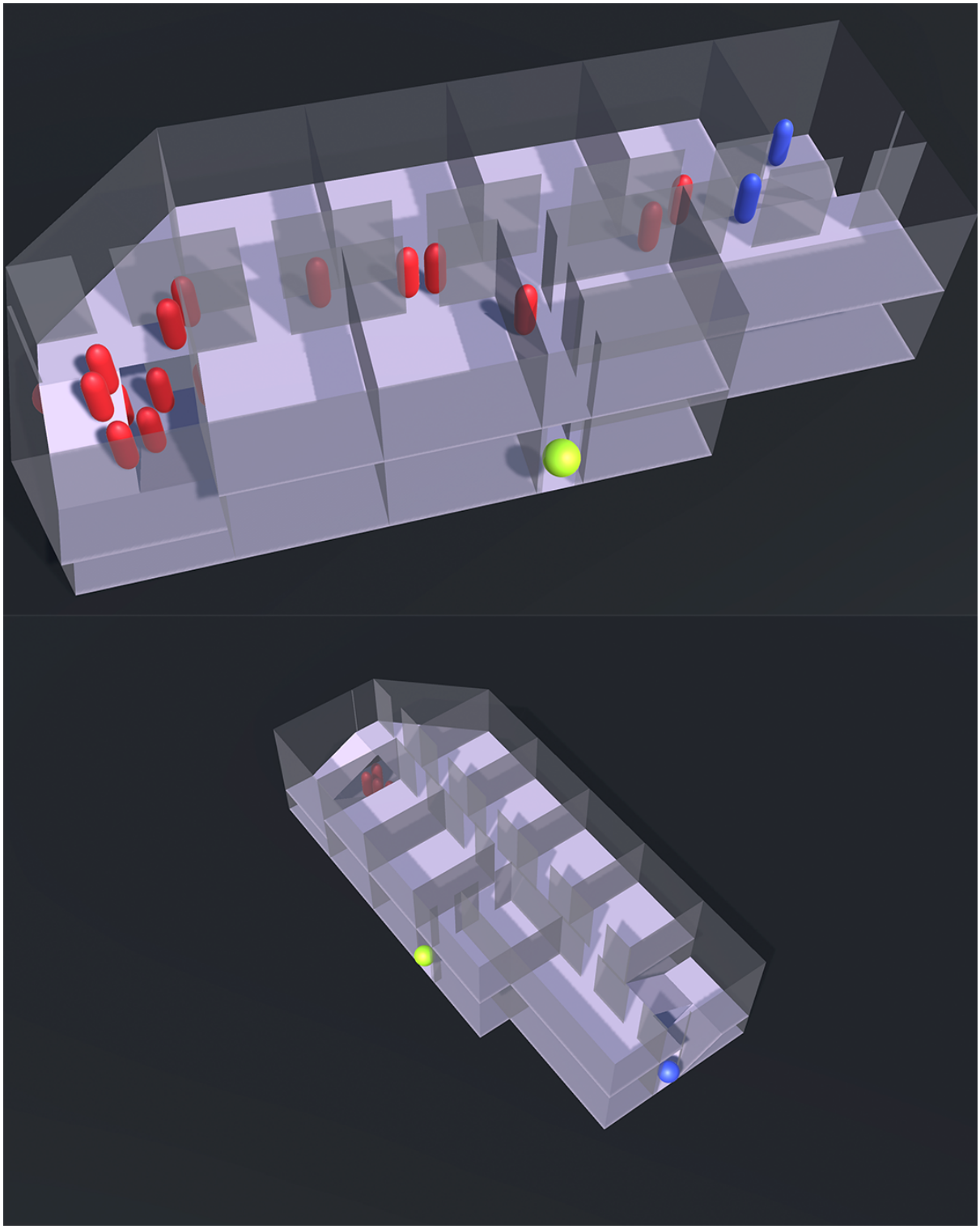
- When opening the simulation a the server will be called this will receive all the correct parameters, needed to run the simulation.
- Changing of simulation parameters will take place from the website by navigating to the Super-user view.
- Once on the correct page you can affect the simulation by changing the current active building (UC8) or by adding a fire (UC19).
- The system sends this information down the chain to the processing subsystem which then does the necessary changes to the simulation.
- A restart/refresh of the simulation is required to see the new changes.



params.JPG

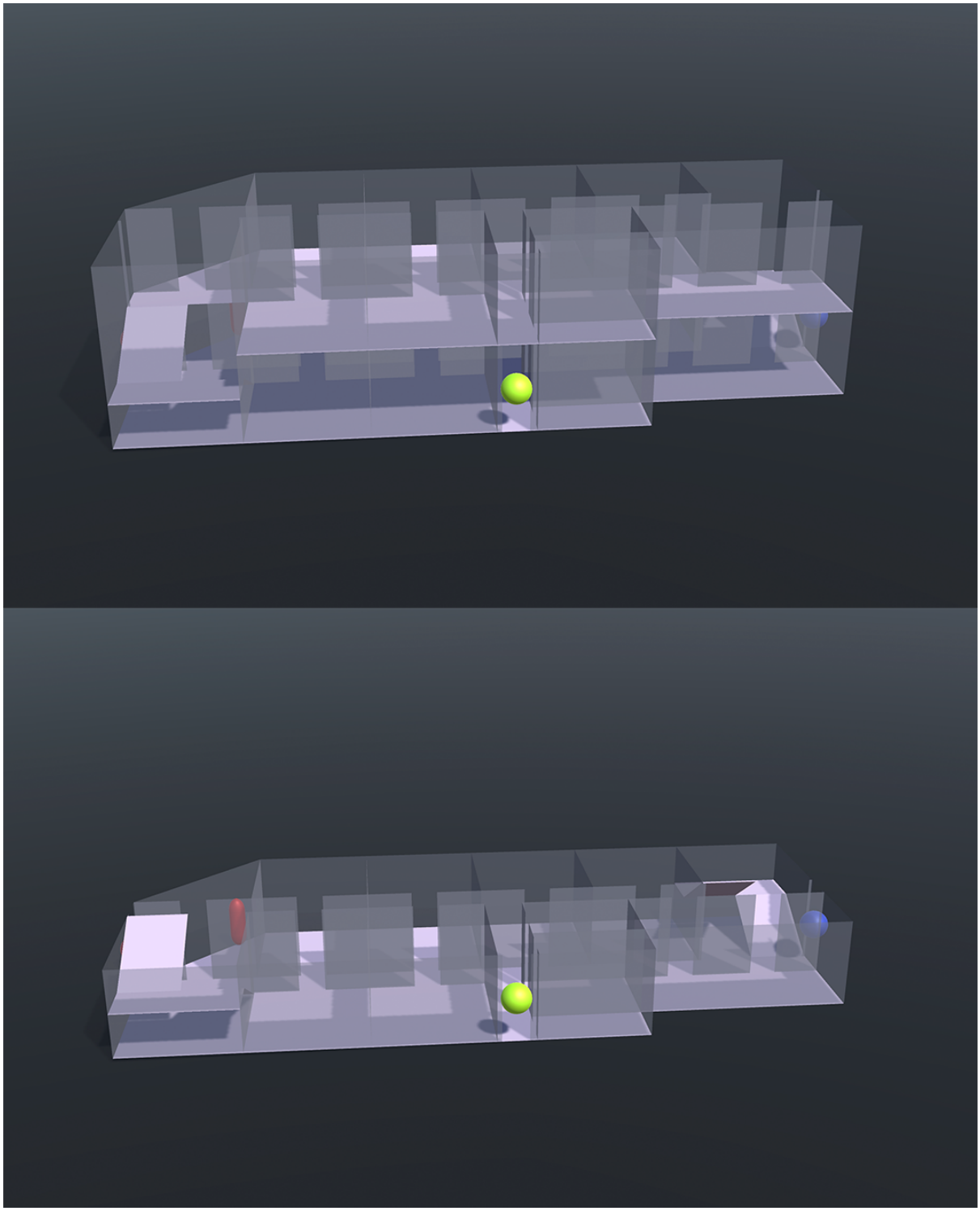
5.16 UC16: Camera movement

- Press L to unlock the camera in the simulation.
- Then use the mouse to move the camera angle in to the wanted position.
- To zoom in use the W key, and to zoom out use the S key.
- Use the A key to move left, and the D to move right.
- Press L to again lock the camera in the position so it will stay fixed



5.17 UC17: View selected floor

- Use the Down Arrow key, to hide floors/people below a certain floor
- Use the Up Arrow key, to show floors/people below a certain level



5.18 UC18: Run simulation

- Open the simulation and it will automatically call the server and start the simulation.
- Press the space button to restart simulation and restart server

5.19 UC19: Adding a fire

- Administrative login: The user goes through *Use case 6* to gain necessary access to the system through the WebUI.

- Querying the information from the system: The web application makes a post request to the HTTPServer indicating that a fire has occured.

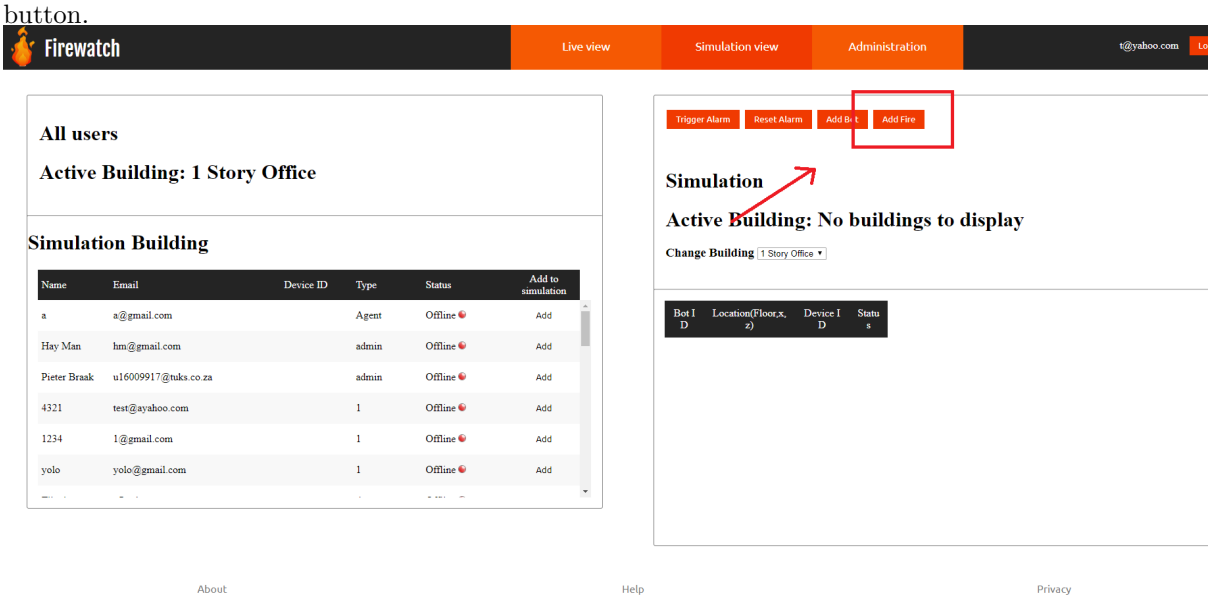
```
{
        "type":"fire",
    "position":[0,5.2],
        "radius": 4,
        "floor":0
}
```

The information is then processed through the function *addFire()* which in turn accesses the function *addFire()* in the processing subsystem. This function calculates the areas affected by the fire, updates the available routes, and redirects users affected to new routes. The following is returned if routes were effectively updated.

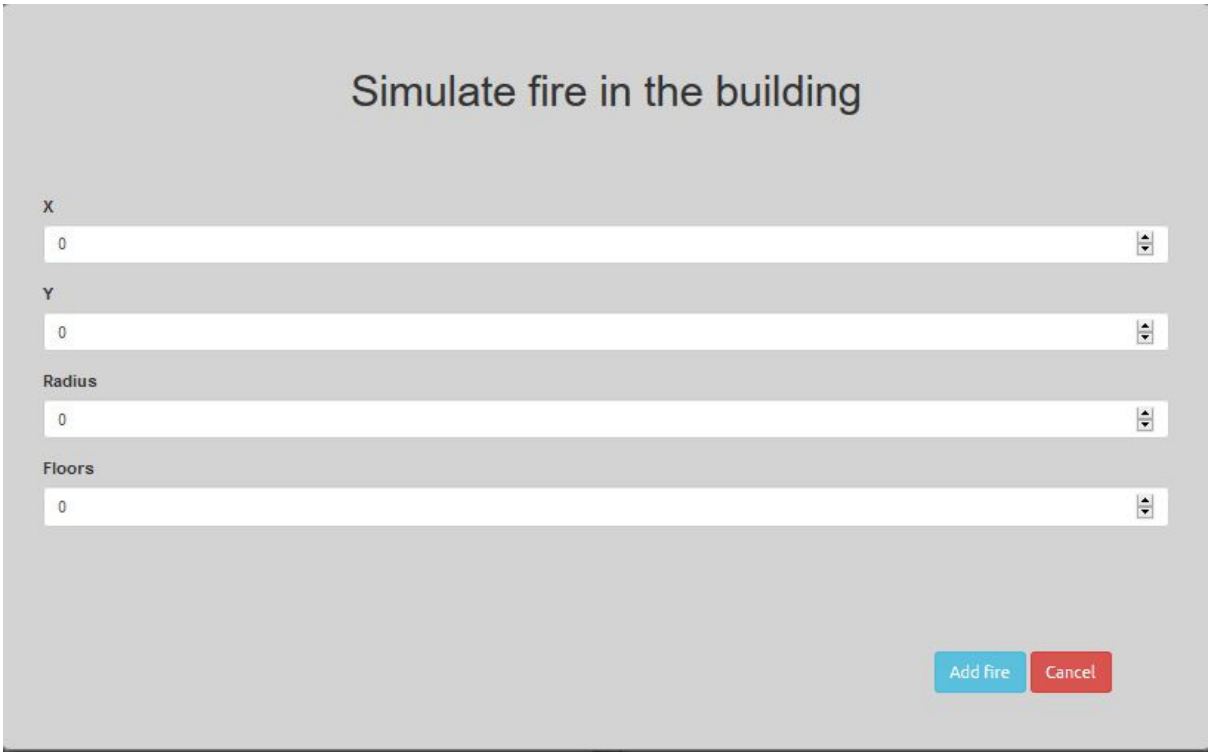
```
{
    "msg": "Fire Added. Some Routes were affected",
    "status", true
}
```

In order to add the fire to the building with the use of website interface:

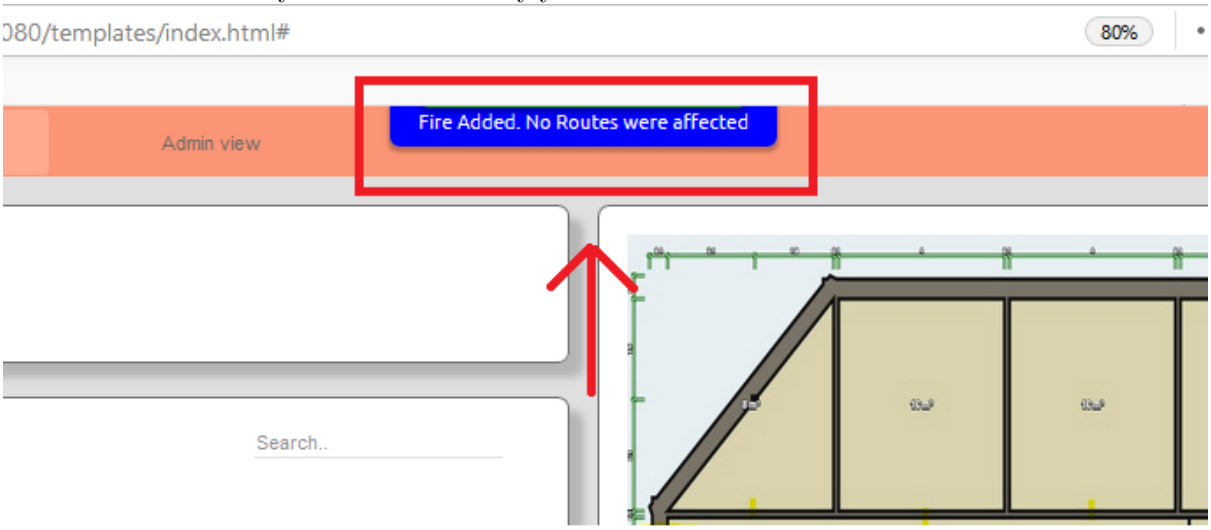
- To open fire parameter window: on the right corner of the default page click on the "Add fire" button.

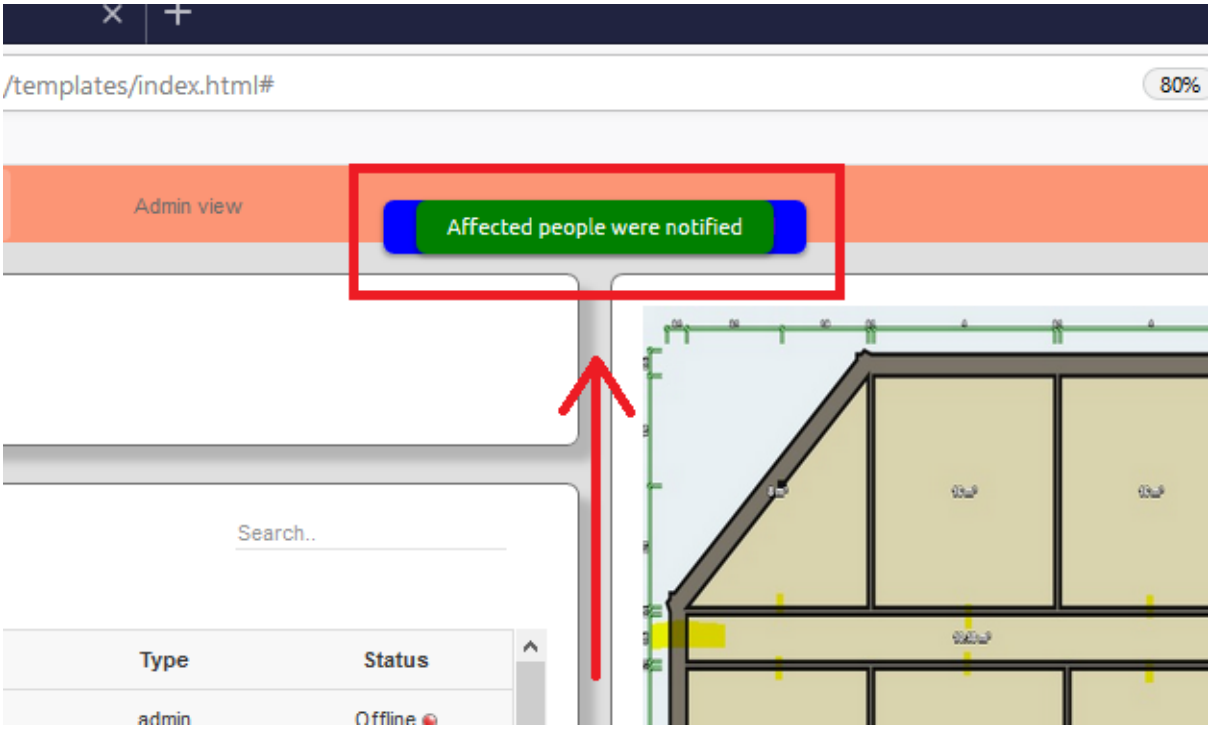


- Provide required information to add fire: for that fill in all the input fields displayed in the window.



- To add fire with specified coordinates: press "Add fire" button which is situated in the bottom-right corner of the currently displayed "Simulate fire in the building" window.
- In case if you change your mind and you do not want to simulate fire you can cancel it by pressing on "Cancel" button that is situated on the right side of "Add fire" button.
- System feedback: the moment you will press on "Add fire" button, you will see notification on top of the screen from the system that will notify you if the fire was added or not.





5.20 UC20: Constructing routes

- The system iterates through all the room within a building and connects all the doors.
- The system also keeps track of the distances between each door when connecting them.
- The system then goes through each floor locating the stairs on each floor and connects them to their counterparts located on other floors.

```
Main
Connecting doors: 0 <-> 2  // distance: 8.0
Connecting doors: 0 <-> 3  // distance: 8.54400374531753
Connecting doors: 0 <-> 4  // distance: 2.5
Connecting doors: 0 <-> 5  // distance: 6.18465843842649
Connecting doors: 0 <-> 6  // distance: 4.924428900898052
Connecting doors: 0 <-> 7  // distance: 7.5
Connecting doors: 1 <-> 2  // distance: 8.54400374531753
Connecting doors: 1 <-> 3  // distance: 8.0
Connecting doors: 1 <-> 4  // distance: 4.924428900898052
Connecting doors: 1 <-> 5  // distance: 7.5
Connecting doors: 1 <-> 6  // distance: 2.5
Connecting doors: 1 <-> 7  // distance: 6.18465843842649
Connecting doors: 2 <-> 3  // distance: 3.0
Connecting doors: 2 <-> 4  // distance: 6.18465843842649
Connecting doors: 2 <-> 5  // distance: 2.5
Connecting doors: 2 <-> 6  // distance: 7.5
Connecting doors: 2 <-> 7  // distance: 4.924428900898052
Connecting doors: 3 <-> 4  // distance: 7.5
Connecting doors: 3 <-> 5  // distance: 4.924428900898052
Connecting doors: 3 <-> 6  // distance: 6.18465843842649
Connecting doors: 3 <-> 7  // distance: 2.5
Connecting doors: 4 <-> 5  // distance: 4.0
Connecting doors: 4 <-> 6  // distance: 6.0
Connecting doors: 4 <-> 7  // distance: 7.211102550927978
Connecting doors: 5 <-> 6  // distance: 7.211102550927978
Connecting doors: 5 <-> 7  // distance: 6.0
Connecting doors: 6 <-> 7  // distance: 4.0
```

- After all the doors and possible exits are connected to each other the system starts with the construction of escape routes.

- The system iterates through all exits specified as building exists which is assumed to be valid according to the Fire escape regulations, and generates a graph for the exits.

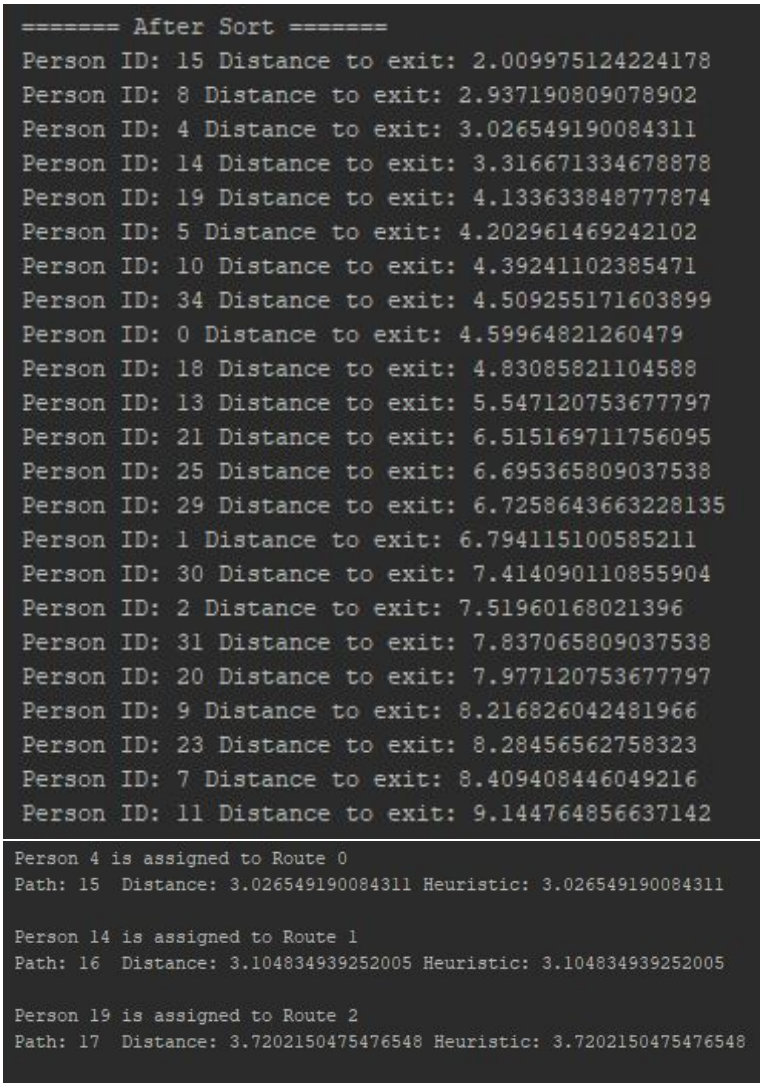
```
Building Complete
Building Routes
Building Routes Complete
Server -> Client:{"msg":"Building built successfully","status":true}
```

5.21 UC21: Assigning routes

- The system first fetches all the people in with their locations.
- The system then uses the graph constructed in *UC20* to determine their distances to each exit.

```
======= After Sort =======
Person ID: 15 Distance to exit: 2.009975124224178
Person ID: 8 Distance to exit: 2.937190809078902
Person ID: 4 Distance to exit: 3.026549190084311
Person ID: 14 Distance to exit: 3.316671334678878
Person ID: 19 Distance to exit: 4.133633848777874
Person ID: 5 Distance to exit: 4.202961469242102
Person ID: 10 Distance to exit: 4.39241102385471
Person ID: 34 Distance to exit: 4.509255171603899
Person ID: 0 Distance to exit: 4.59964821260479
Person ID: 18 Distance to exit: 4.83085821104588
Person ID: 13 Distance to exit: 5.547120753677797
Person ID: 21 Distance to exit: 6.515169711756095
Person ID: 25 Distance to exit: 6.695365809037538
Person ID: 29 Distance to exit: 6.7258643663228135
Person ID: 1 Distance to exit: 6.794115100585211
Person ID: 30 Distance to exit: 7.414090110855904
Person ID: 2 Distance to exit: 7.51960168021396
Person ID: 31 Distance to exit: 7.837065809037538
Person ID: 20 Distance to exit: 7.977120753677797
Person ID: 9 Distance to exit: 8.216826042481966
Person ID: 23 Distance to exit: 8.28456562758323
Person ID: 7 Distance to exit: 8.409408446049216
Person ID: 11 Distance to exit: 9.144764856637142
```

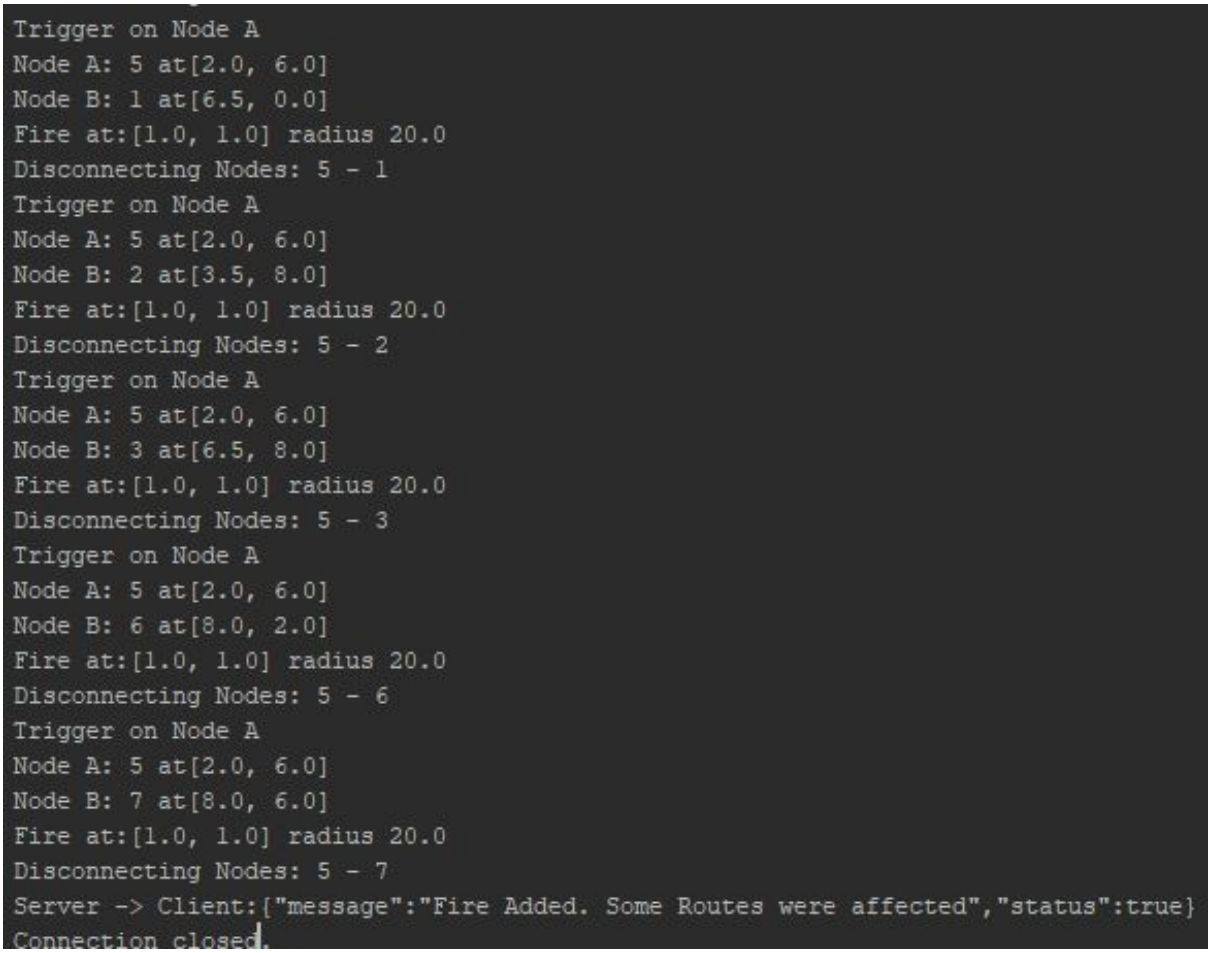
- The system then sorts all the people based on their distances to the exits and begins the route assignment. The system now also considers the amount of people assigned to the routes when assigning.



5.22 UC22: Changing routes

Changing of routes can only take place if there was a fire added within a building (*UC19*).

- The server takes the location of the fire and calculates if any of the paths in the graph intersect with it.
- If a path intersects with a fire it marks the path as invalid and removes it from the graph to ensure that people cant be routed through danger by mistake.



5.23 UC23: Binding device IDs

This Use case is currently being done manually through the back-end and will be explained when it occurs automatically or through the front end.

5.24 UC24: Unbinding device IDs

To be implemented

6 Troubleshooting

Errors that might occur:

- Invalid building coordinates: Upon constructing the building coordinates are provided that indicate the corners of the room. These corners are considered invalid if:
  - They do not create a full room with connecting walls
  - Corner coordinates aren't exact

  The system handles these invalid point by:
- The system calls a check function first to see if the room is valid, if it isn't valid it will print a message in the console and it won't be added to the building
- If a room is valid but its coordinates falls outside the floorplan it won't add it to the building.
- Invalid user placement: A user's position is considered invalid if:
  - They are colliding with objects in the building such as walls.

This will be handled once we have access to the sensors and can investigate the functionality of the sensors themselves.

- Network error: As of currently the servers are run locally. Once the servers are deployed online investigation will be done into the reliability of the system. Backup servers will be set up in case of primary server failure, with sufficient testing to ensure reliable service.
- Simulation Error: if the simulation is open and is not running the simulation, ensure the server is running and and has a building loaded. Once this is complete press the space bar when in the simulation it will restart the simulation.
- Application Error:
  - due to the fact that the current server can only run locally, the device the application is installed on must be connected to the same network which the local server is running on.
  - if the application is running and no activities seem to be occurring, pressing the "check status" button will initiate a new call to the server.
  - if the application is running and no activities seem to be occurring, tapping on the screen 5 times will activate the developer terminal which will display all calls made from the application to the server.
  - if the application is running and no activities seem to be occurring then close the application and restart it
- Website Error:
  - Most of the errors that may occur on the website might be due to user's incorrect inputs. In this case the system will notify user with the red notification that would usually occur on top of the screen. Such notifications will provide all the needed information in order to correct the error.
  - In case if user has added new user or building to the system, but the system does not display new information, user should just refresh the page. For that one can use "f5" key, "f5+fn" keys, "R+Shift" keys, or simply use refresh button of your browser.
  - In case if the layout of the website will suddenly break or nothing will be displayed on the table, check that the server, on which you running the entire system, actually running.