
FAMILY
REUNION
نحنُ نسعى لغدٍ مُشرِق .. اليوم

Faculty of Computer Science and artificial intelligent

Information System Department

Supervised by:

Dr. Mohamed Marie

Team Members

Mahmoud Maged Mohamed

Mahmoud Gamal Mohamaden

Mahmoud Anwar Mahmoud

Mahmoud Ahmed Sayed

Mostafa Khaled Omran

Acknowledgement

First, we would express our deepest gratitude and appreciation to our advisor Dr. Mohamed Marie for his support, outstanding guidance and encouragement throughout our graduation project and his continuous support and help as he was always there for us providing more than the needed from him since the very beginning of our project.

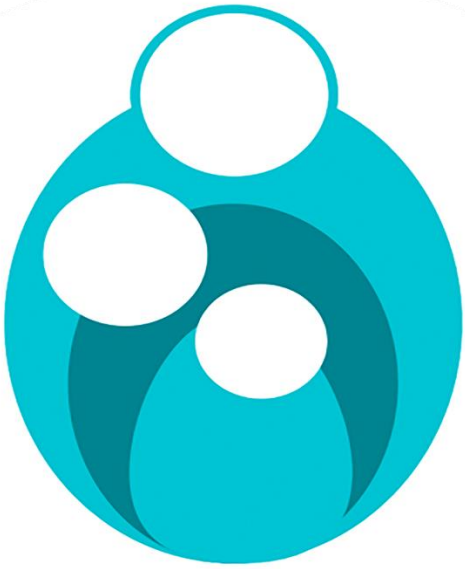
We would also like to thank Eng. Samar Samir for helping us to organize our idea, recommend solutions , we thank her for the effort she did with us from the beginning to end of the project

Finally, for our faculty and all FCAI-H doctors for teaching us and providing the suitable environment that leaded us to represent the best image that computer science graduates of Helwan University are supposed to represent

Table of content

ABSTRACT

These are chapters about REUNION project , this documentation provides the scope and context of the project to be undertaken. It details the intended user group and the value that the system provides to them. It also provides details about phases of implementing the project, schedule for the completion of the project, including a list of all the deliverables and presentations required. The intended audience of this document is all the companies which are interested in web applications and systems development and members of the faculty of computers and information Helwan University.



1- Introduction

In this chapter, we will discuss the project's objectives, purpose, scope of the project, and work needed to build the project.

1.1 Overview:

Our idea revolves around helping families and our society who lost someone they love kids or grandparents. We are trying to find missing people. Which help in reconnecting family member by forwarding relative to their current whereabouts

1.2 Project Motivation:

1-The reasons behind our choice to develop this project are:

- **We want to reunion lost peoples to their families.**
- **Finding lost people as soon as possible.**
- **We recognized that in the prior period many families search for their kids and old people.**

2-Our project is important because it help family to find their missing.

3-the new idea that have been proposed by this project is using face verification to match lost people faces and comparing data.

1.3 Problem Statement:

The issue that have been addressed by this project is finding lost people and bring them back to their families.

The steps to our solutions:

- **Entering all possible data about the lost person.**
- **The mainly point is to enter a photo of him**
- **System will use the photo to compare between faces using face verification technology.**

1.4 Project Aim and Objectives

- **Write about the overall purposes of this project, should be clearly and concisely defined. In this section we should answer the following questions:**
- **What is the goal that this project wants to achieve?**
- **How can this project achieve this goal?**
- **First user enters his missing member data then search in missing people data then system return result if the system managed to find the missing then provide user by enough information about the missing location and guide him by GPS if not, then system allow user to enter a report in waiting list and after 48-hours the system automatically change report status to Missing-reported list.**
- **System compares homeless with Missing-reported list and if he fined matched report then should notify missing relevant throw Gmail or Phone-Call if not matched then go on and add him in unknown-relevant table.**

1.5 Project Scope:

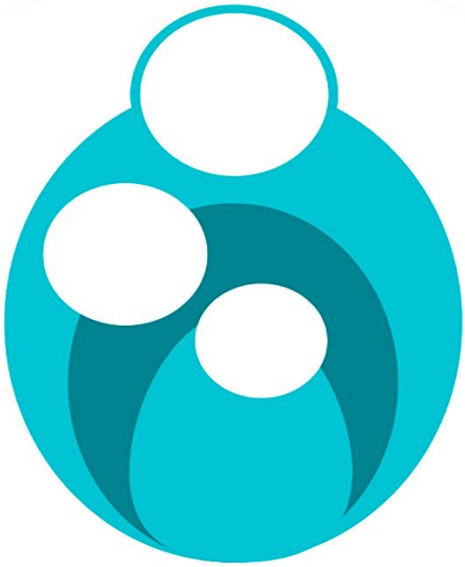
The system has a sociable scope which helps government and users to make it easier to find missing, if the system finds the right person, it will redirect the user to shelter or nursing home.

1.6 Project Limitations:

-We just play role of the bridge between user and governmental organization, with no responsibility about confidentiality of user or intentional miss use of website.
-We don't support communications between users.

1.7 Project Expected Output:

The system should have only 2 outputs:
-First case if the system finds the lost person in the data: the output will be the place where he stays in.
- Else: unfortunately, the system will output that there is no match data.



2- Planning and Study

In this chapter, we will discuss how we are planning to build the application, presenting many approaches like feasibility study, estimated cost plan, conducting analysis of existing systems and analysis for new system.

2.1 Project planning

2.1.1 Feasibility study:

In order to determine the achievability of this project we need to dig more in these aspects:

The problems that the project solves:

- **The increasing number of missing reports in the last few years and the dispersed families exposed in this situation.**
- **The main problem that our project solve is providing an easy way to go through arranged steps which will help in finding the missing.**
- **Helping those people will save a lot of time, effort, and a better quality of life.**
- **Long term follow-up is existing to have a good control on finding processes by admins, shelters and police Stations.**
- **Making a strong and secured data consistency between government organization and charity organization.**

Whom will the project benefit?

- **Shelters' admins**
- **Missing's relatives**
- **Police stations' admins**
- **Lost people**
- **All society**

Market analysis:

- Main competitors:
    - **There are no competitors in Egypt.**
    - **but international there is:**
        - **https://www.fbi.gov/wanted/kidnap**
        - **https://www.missingpeople.org.uk/**
        - **https://www.missingkids.org/search**

- Unique points in our project:
    - **Use face verification.**
    - **secure data**
    - **guide location**
    - **Provide process instructions.**

- Target Market:
    - **Age: 18 – 80 years.**
    - **Gender: Both.**
    - **Place: we'll start from Egypt then will expand world widely.**
    - **Income Level: nonprofit organization**

Organizational and operational analysis:

- Number of team members this project will need:
    - **4 developers.**
    - **1 tester.**
    - **2 designers.**
    - **2 analysts.**

- Technical analysis:

  1-Hardware:

    - **Computer device like pc, laptop , mobile or tablet**
    - **Internet connection**
    - **Server for hosting**

  2-Main technologies and programs we're associated with:

    - **browser**
    - **html5**
    - **css3**
    - **bootstrap 5**
    - **angular11**
    - **typescript**
    - **sass**

- node.js
- express.js framework
- mongo Database
- Microsoft Visual studio code
- Microsoft office
- trello
- git&GitHub
- Adobe XD – Adobe Photoshop
- Diagrams.net
- lucidchart.com
- visual diagram
- google maps API
- rapidapi/ Face Verification

Resource and Time analysis:

1-Resources needed:

- Devices for coding, running, and testing – Laptops, tablets and mobile phones.
- Coding programs.
- Internet service.
- Design programs.
- Found missing data set for their records.
- Police stations and shelters data set
- Maps API – packages
- Node.js and angular libraries and packages

2- Time schedule:

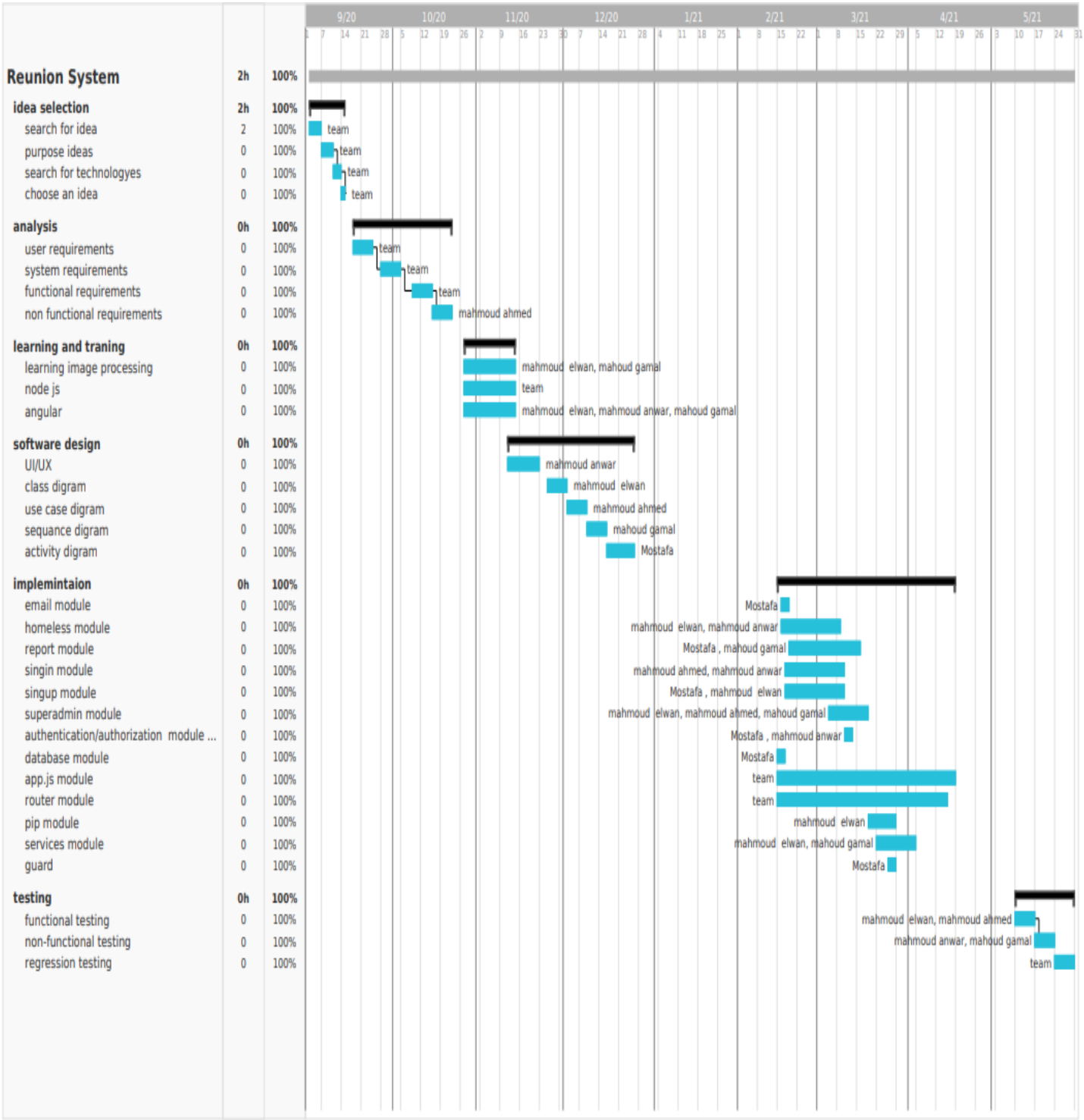
- 1 month for planning.
- 4 months for training.
- 5 months for developing.
- 2 months for testing.

2.1.2 Estimated cost

The budget of the application as programming and its process includes:

- Purchasing full stack web development course = 220$/member
- Publishing the website= 100$/year
- Meeting in a public workspace. =6.36$/h

2.1.3 Gantt chart



2.2 Analysis For New System:

2.2.1 System Requirements:

windows & Android & IOS

Since our app's code would be written in responsive web design way it'll work on any screen with operating system that runs browsers.

Browser

Because we would develop our code with html and node.js so it'll work on any browser like (chrome, opera, Firefox)

Internet connection

Since we are developing a website we need an internet connection to store and retrieve the data.

GPS

The application will need to know the guest place in order to direct them to shelters or police stations.

2.2.2 Functional Requirements

Use case	*Register*
Actors	Shelter/Police Station
Precondition	No precondition
Basic flow	1-fill all required data. 2-click signup
Post condition	The request be under review by Admin

Use case	*Login*
Actors	*SuperAdmin/Shelter/Police Station*
Precondition	*Registered,confirmEmail,SuperAdminAprove*
Basic flow	*1-enter username and password.* *2-click login*
Post condition	*If matched then admin or Moderator go to his dashboard with his privilege's function* *Else return to login page*

Use case	*Logout*
Actors	SuperAdmin/Shelter/PoliceStation
Precondition	Login
Basic flow	1-open profile page 2-click logout
Post condition	The logout will redirect to admin login page

Use case	*Forget Password*
Actors	SuperAdmin/Shelter/Police Station
Precondition	Register
Basic flow	1-fill all required data 2-click send
Post condition	Return to login Page

Use case	*Update Password*
Actors	*SuperAdmin/Shelter/Police Station*
Precondition	*Login*
Basic flow	*1-fill all required data* *2-click Update*
Post condition	*Return to login Page*

Use case	*Update Profile*
Actors	*SuperAdmin/Shelter/Police Station*
Precondition	*Login*
Basic flow	*1-fill all required data* *2-click update*
Post condition	*Return message updated Successfully*

Use case	*Add homeless people*
Actors	SuperAdmin /Police Station
Precondition	login
Basic flow	1-open add missing person form 2-fill all required data of the new missing person 3- click submit
Post condition	Will show a message data has been added successfully

Use case	*Communicate With Relative*
Actors	SuperAdmin /PoliceStation
Precondition	login
Basic flow	1-get matched report data 2-send email or call report maker
Post condition	Will show a message email has sent successfully

Use case	*Change Report Status*
Actors	SuperAdmin /PoliceStation
Precondition	login
Basic flow	1-detect report 2-chosse report statues 3-Click Update
Post condition	Will show a message data has Updated successfully

Use case	*Administration Search*
Actors	SuperAdmin/Shelter/PoliceStation
Precondition	login
Basic flow	1-fill in all required inputs 2-click search beside the form
Post condition	If data is matched with stored data: all person data and his place viewed. Else: unfortunately the system will output that there is no match data Then ask user to enter it in missing database

Use case	*Search Homeless Resident*
Actors	SuperAdmin/Shelter/PoliceStation
Precondition	login
Basic flow	1-fill in all required inputs 2-click search beside the form
Post condition	If data is matched with stored data: all person data and his place viewed. Else: unfortunately the system will output that there is no match data

Use case	*Display Homeless Resident*
Actors	SuperAdmin/Shelter/PoliceStation
Precondition	login
Basic flow	1-select specific shelter or select all 2-click search beside the form
Post condition	If data is matched with stored data: all person data and his place viewed. Else: unfortunately the system will output that there is no match data

Use case	*View Administration*
Actors	SuperAdmin
Precondition	login
Basic flow	1-choose to get all moderator or enter specific one details 2-click view
Post condition	Return list of moderator

Use case	*Change Privileges*
Actors	SuperAdmin
Precondition	login
Basic flow	1-select police Station or Shelter 2-pick new privilege 3- submit
Post condition	Return updated successfully

Use case	*Delete Administration*
Actors	SuperAdmin
Precondition	login
Basic flow	1-Select PoliceStation or Shelter profile 2-Click Delete
Post condition	Will show a message moderator has Deleted successfully

Use case	*ADD Administration*
Actors	SuperAdmin
Precondition	login
Basic flow	1-fill all required inputs 2-click add
Post condition	Will show a message moderator has added successfully

Use case	*check Request To Join*
Actors	SuperAdmin
Precondition	login
Basic flow	1-check requests 2- click approve or reject request
Post condition	Will show a message action has done successfully

Use case	*User Search*
Actors	Guest
Precondition	No precondition
Basic flow	1-fill in all required inputs 2-click search beside the form
Post condition	If data is matched with stored data: all person data and his place viewed. Else: unfortunately the system will output that there is no match data Then ask user to enter an report

Use case	*View Report*
Actors	SuperAdmin /Police Station
Precondition	Login
Basic flow	1-choose to get all reports or enter specific one details. 2-click view
Post condition	Return list of report

Use case	*Submit Police report*
Actors	Guest
Precondition	No precondition
Basic flow	1-fill all required inputs 2-click submit button
Post condition	Will show a message that inform the user that data has been added to waiting list

Use case	*Edit report*
Actors	Guest
Precondition	No precondition
Basic flow	1-select report 2-fill all required inputs 3-click update button
Post condition	Will show a message updated successfully

Use case	*Change report status*
Actors	system
Precondition	Make report with 24 hours in waiting list
Basic flow	After 24 hours system will change report status to active
Post condition	

Use case	*Close homeless status*
Actors	Super Admin/PoliceStation
Precondition	login
Basic flow	1-select homeless 2-pick new status 3-click change button
Post condition	Return message homeless closed successfully

2.2.3 Non-Functional Requirements

- **Availability**
  - **The guest will be able to use the website 24 hours/day 7 days/week**
  - **There is no data can be showed or updated without an internet connection.**
  - **The locating cannot work without an internet connection and GPS.**
  - **Under any bad circumstances, the website should notify user with message contain that the system is unavailable.**

- **Maintainability**
  - **Any new feature must not impact any of the dashboard functions.**

- The system shall not be shut down for maintenance more than once in a 48-hour period.

- **Security**

  - Any unauthorized access to the website is denied, no one can use any ungiven privilege without creating an authorized account.
  - All the admins are verified.
  - Authentication and authorization made for every role of registered users.
  - Shelters' data cannot be shown to other shelters, just super admin have authorization.
  - Police stations' data cannot be shown to other police station, just super admin have authorization.
  - Passwords will be hashed in database.
  - National ID will be hashed also.
  - The access permissions for any data may only be changed by the website developers.
  - Only super admin can edit any data on system.

- **Usability**

  - The guest will have an instructions page to see how to use the website.
  - The guest only fill the required data and submit it or search in few clicks, which will take no time.
  - The interface is very important to be user friendly since old people will use the website.
  - The website should be easy to use by adult members.
  - We followed comfortable material design, colors, and fonts.

- **Flexibility**

  - User will interact with the Egyptian national language (Arabic) initially (English will be added later).

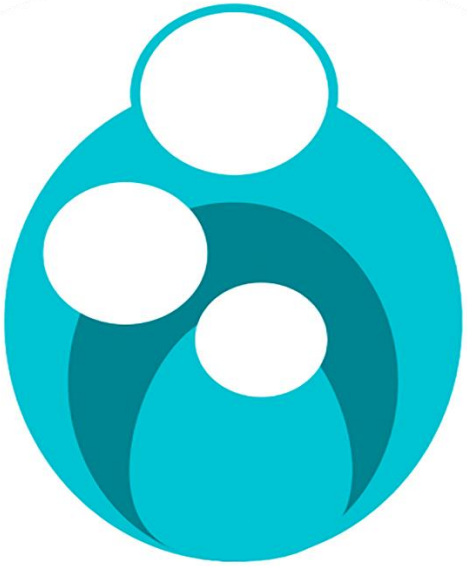
- No piece of text that might be displayed to a user shall reside in program without changing source code.
- Admins can edit or modify their data any time.
- Report status will be changed automatically after 48 hours from submitting.

• Scalability

- The website will be updated so it can serve more users (that increases gradually), process more data and do more transactions.
- The website will work on any device (laptops, mobile phones, tablets)
- The website can be updated gradually to fit working in other countries.
- Time to calculate any data mustn't increase with users' number increasing.
- Functions and roles shall be scaled for the number-growth of the users.

• Performance

- The application needs at most 1% percent of the battery usage and at most 80 MB used storage space should be available in the phone, which shall be all-unused at high load.
- Routine maintenance that is executed while users are active shall not cause any delay of more than 10% from the normal for the users.

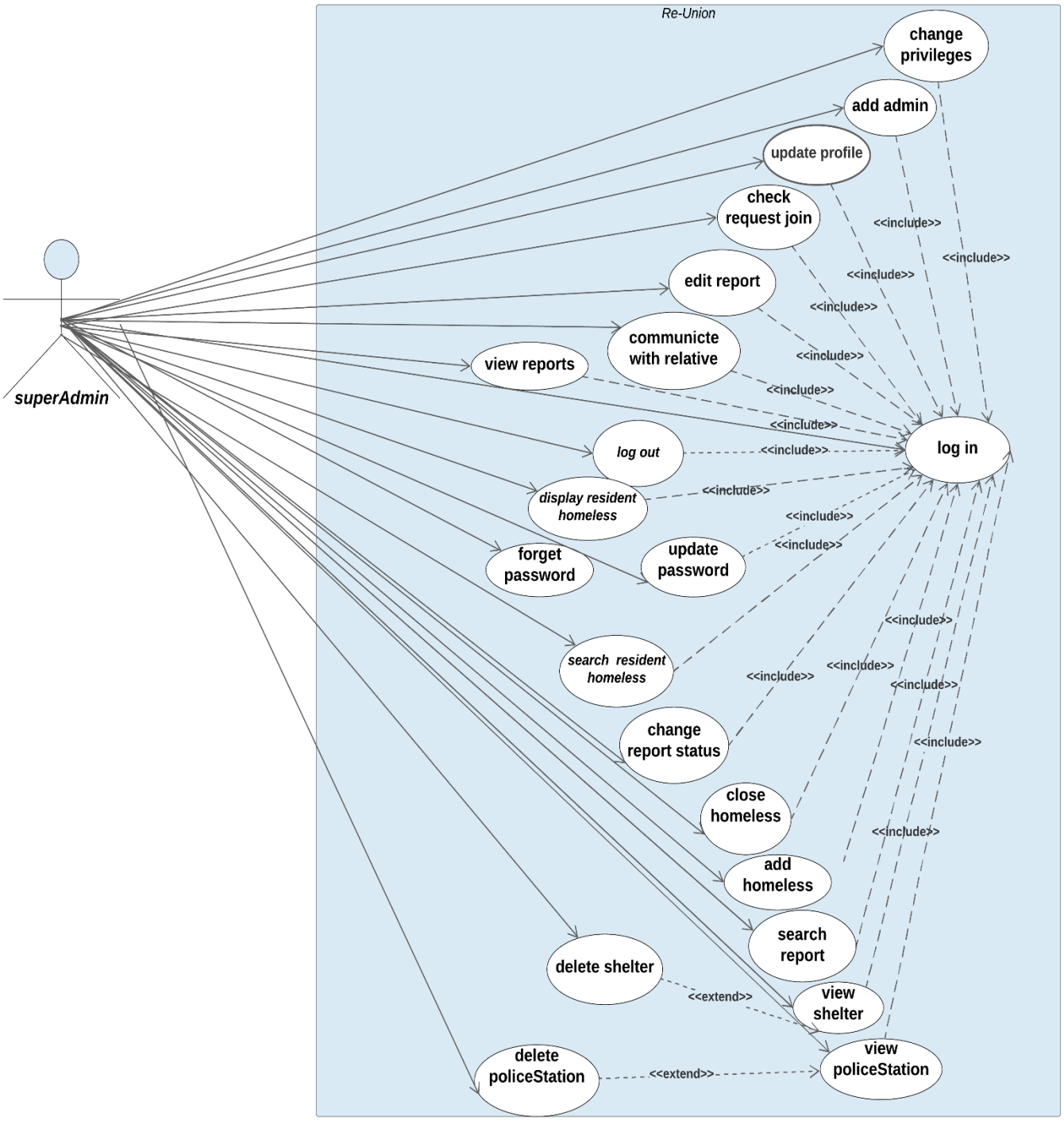


3-Software Design

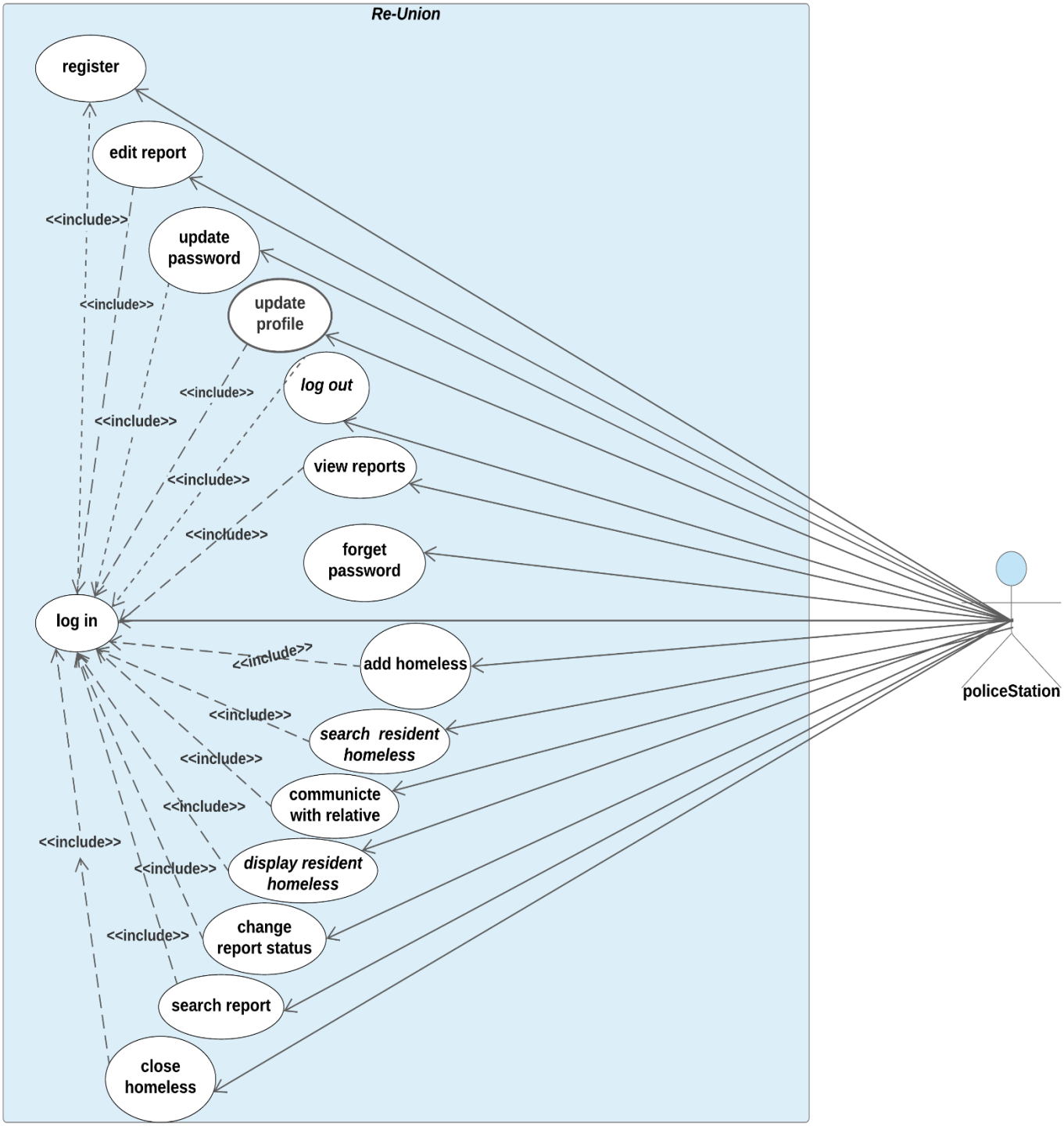
In this chapter we will go deeper in Reunion software design, and present its UML diagrams

3.1 Use case Diagram

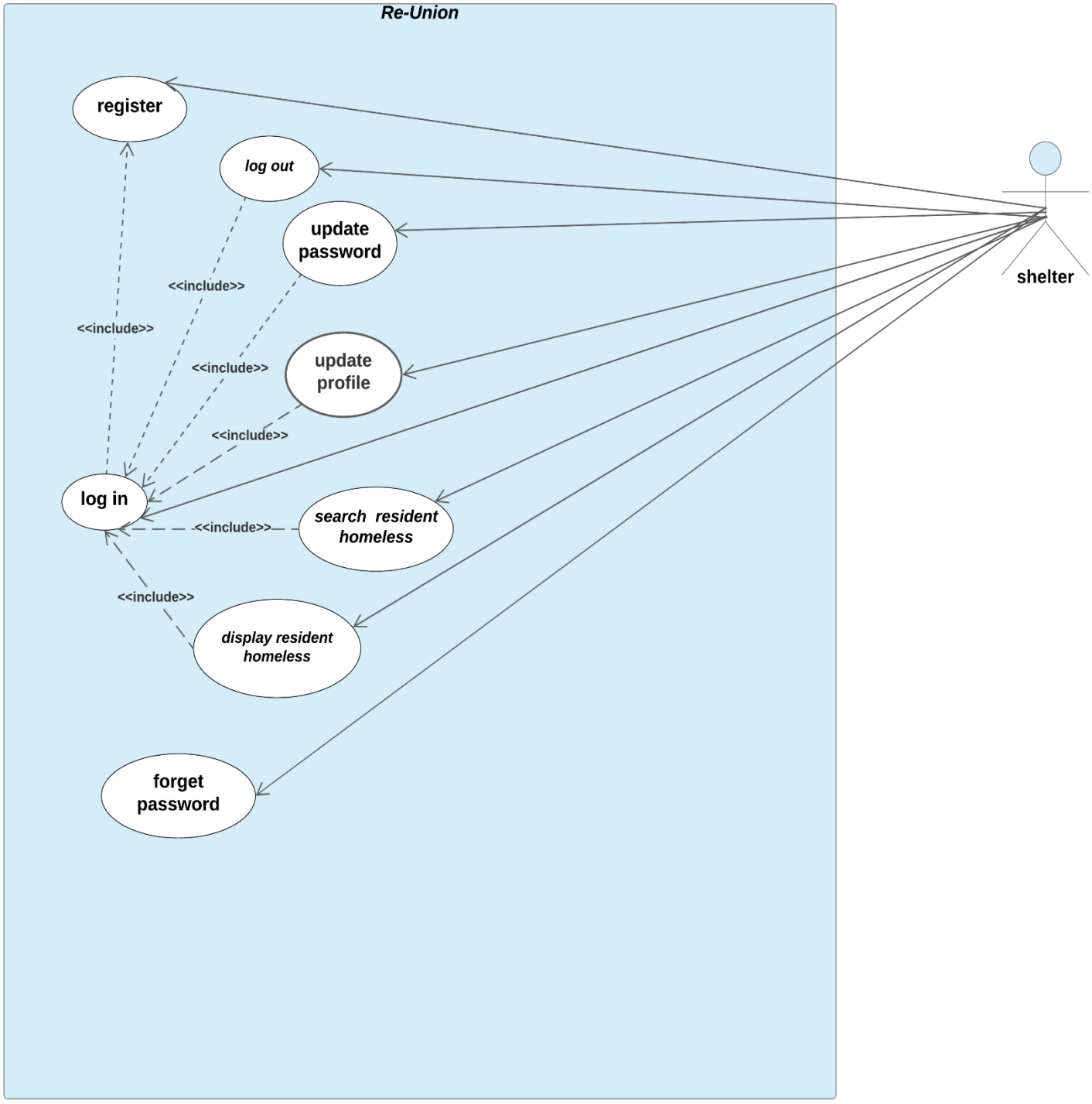
3.1.1 Super admin



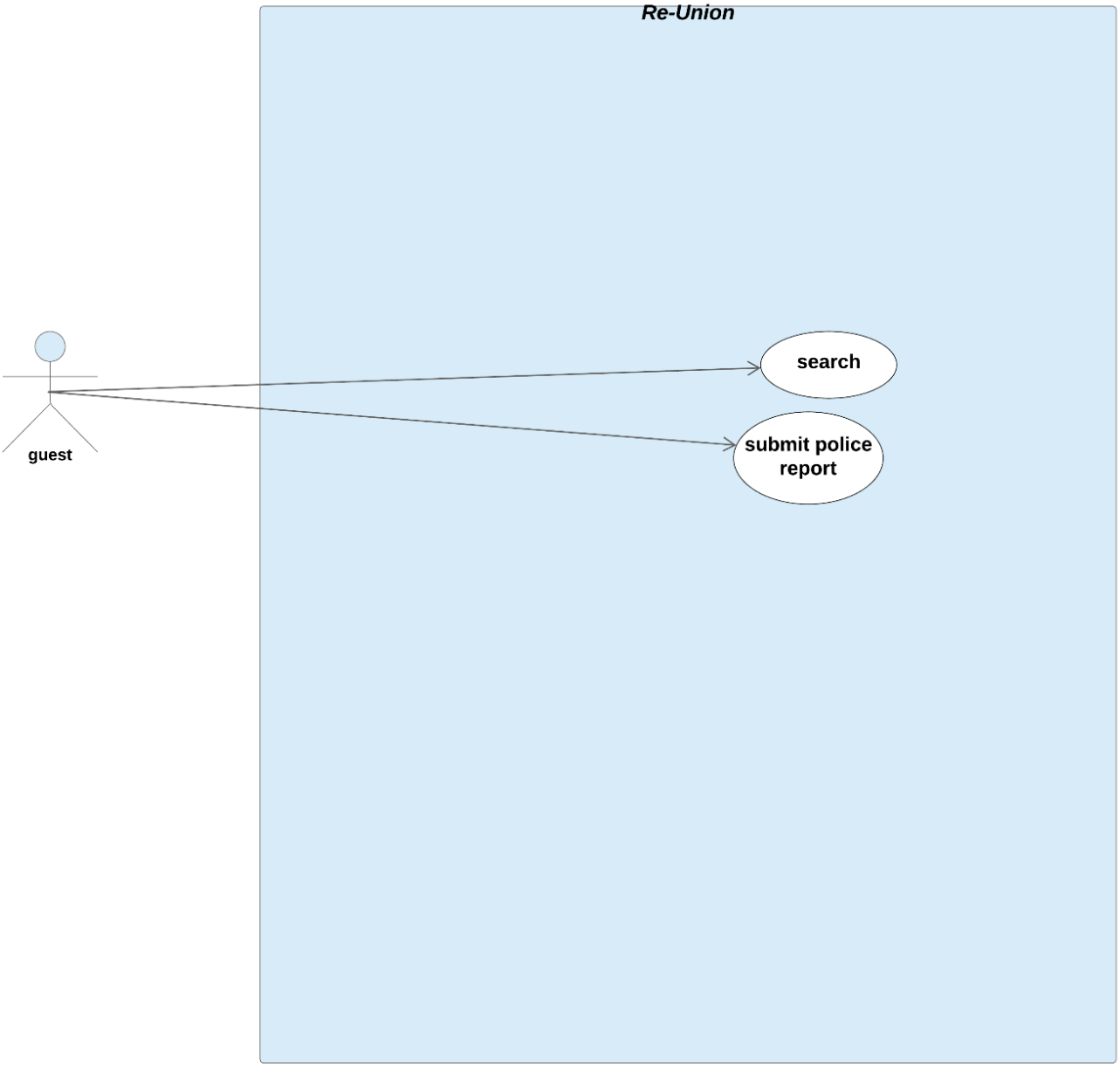
3.1.2 police station admin



3.1.3 Shelter admin

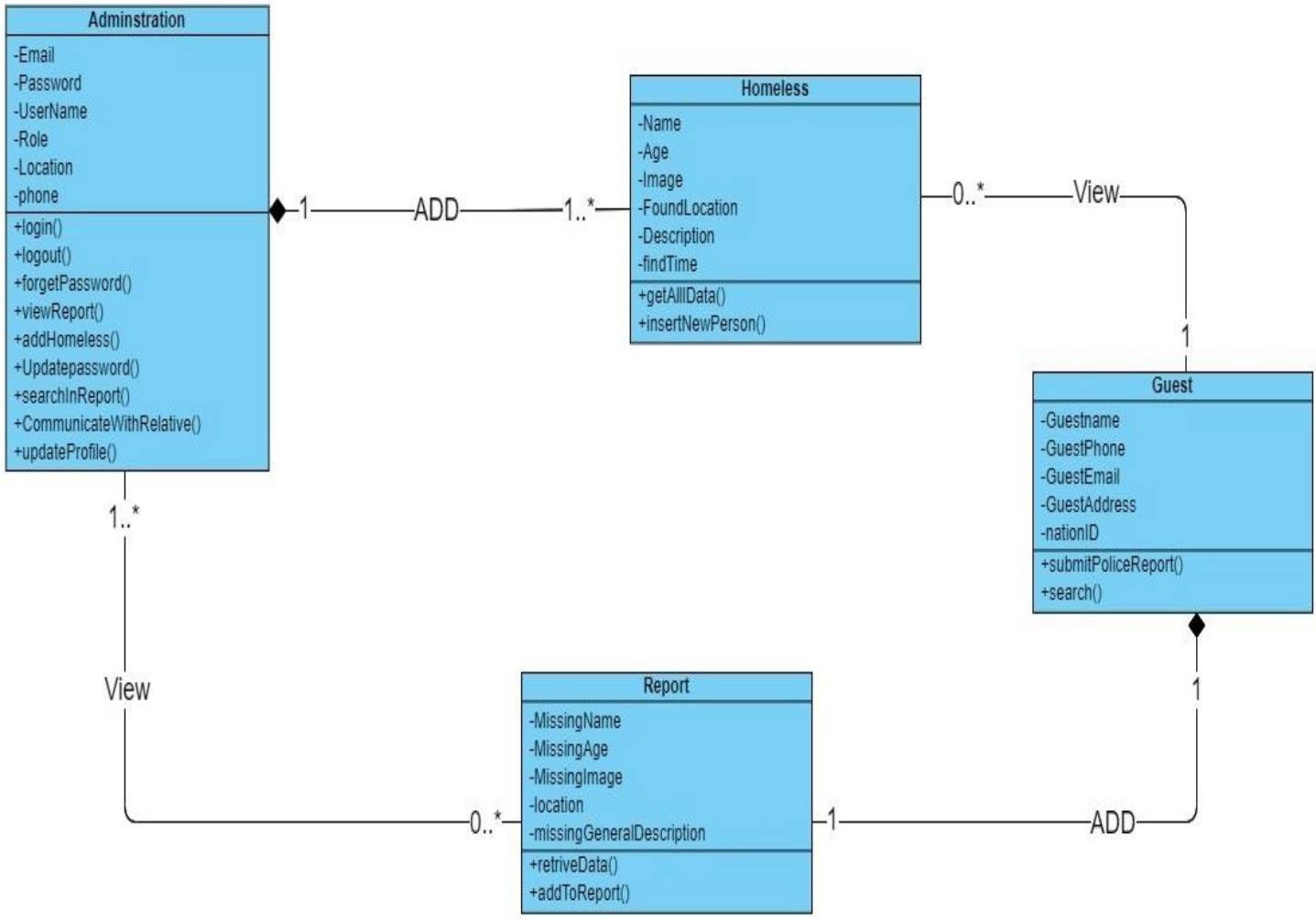


3.1.3 Guest

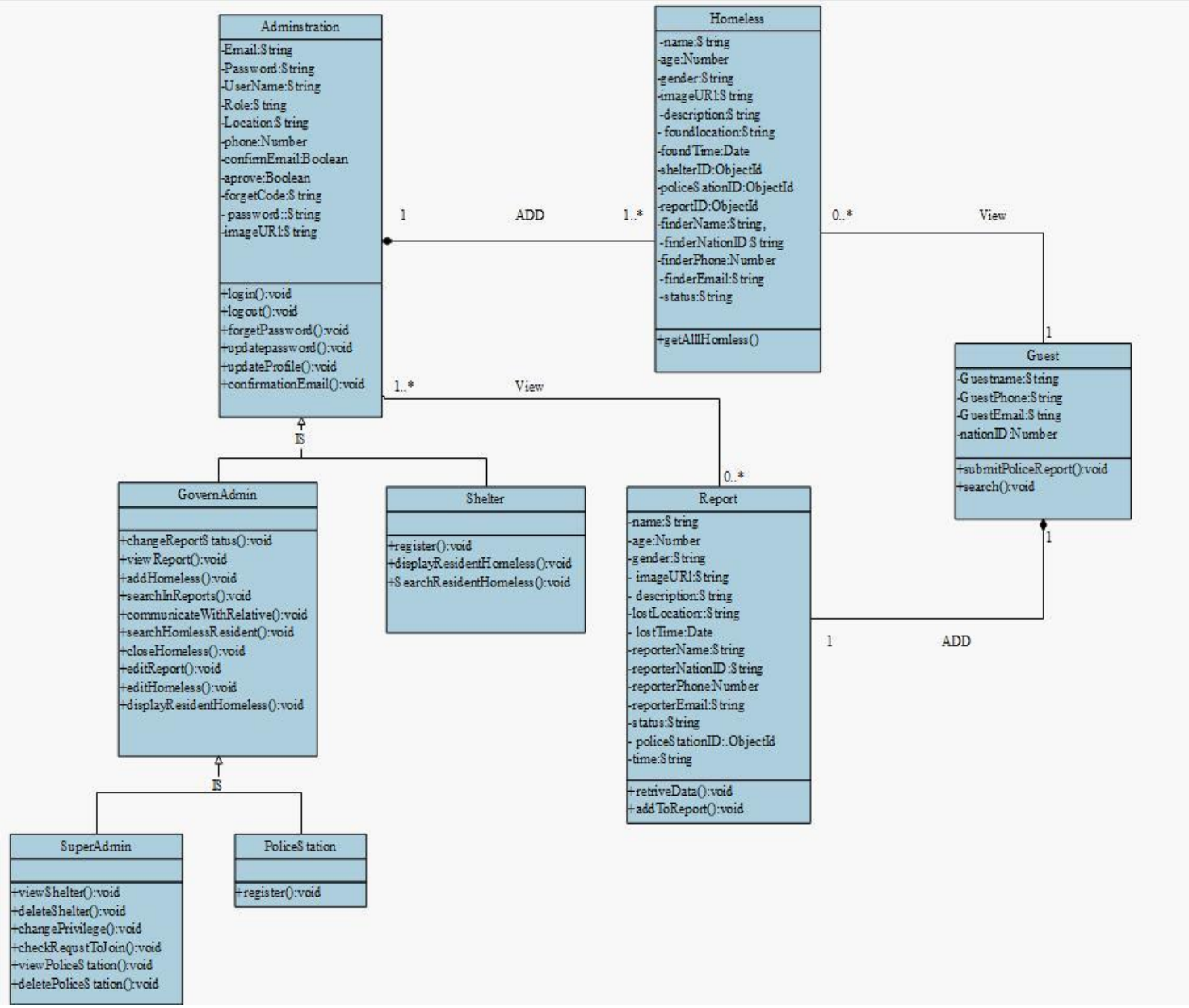


3.2 Class Diagram

3.2.1 class diagram V1

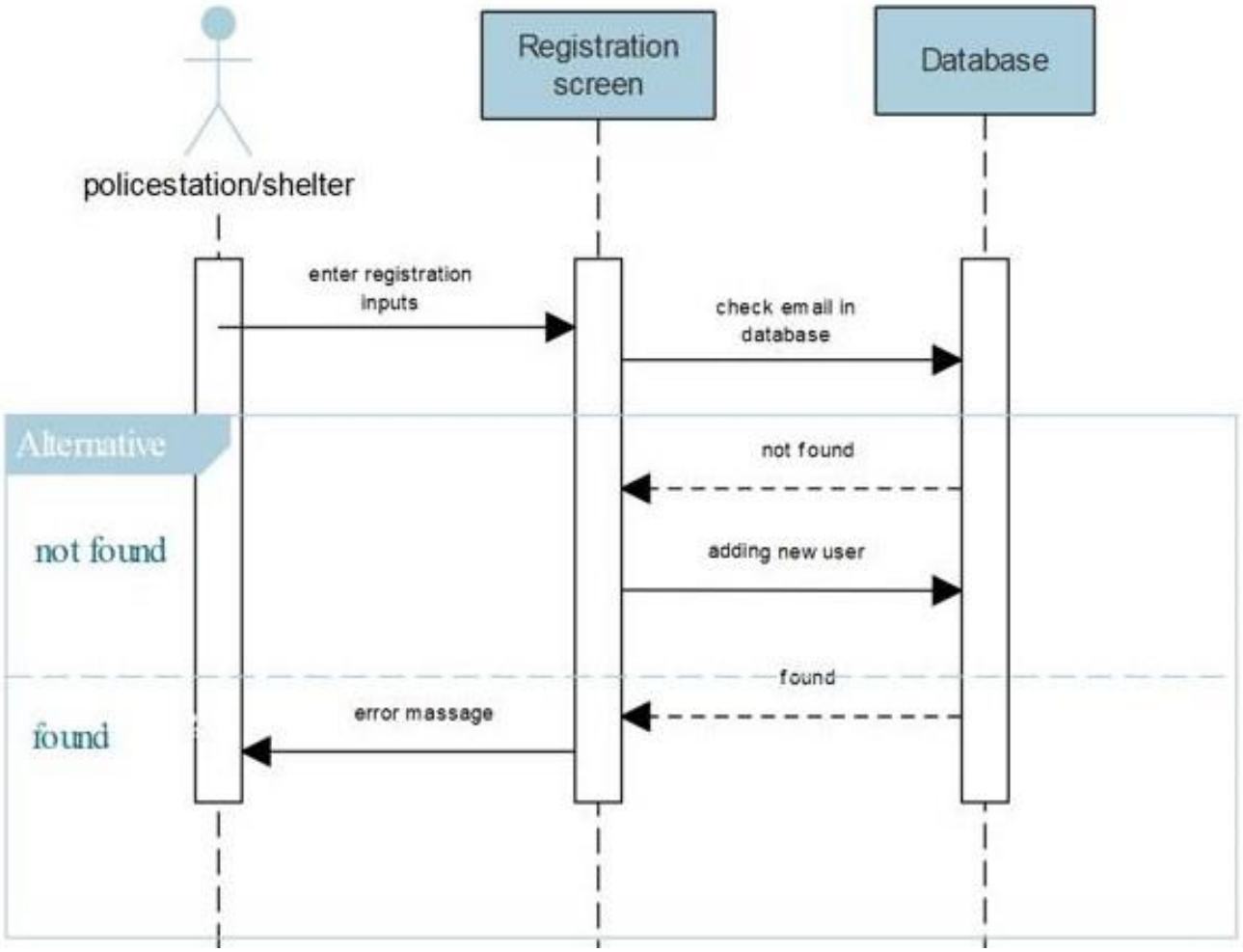


3.2.1 class diagram V2

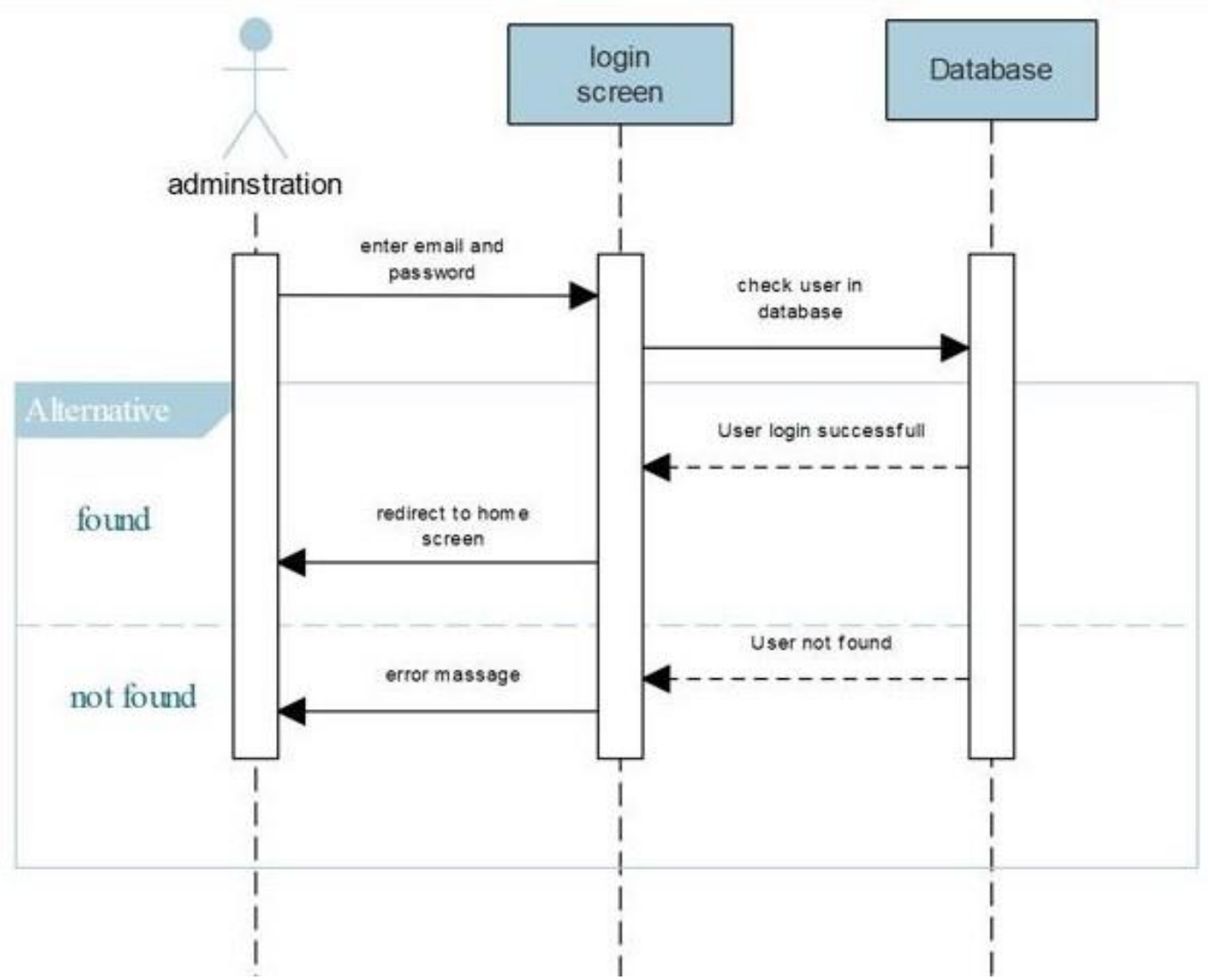


3.3 Sequence Diagram

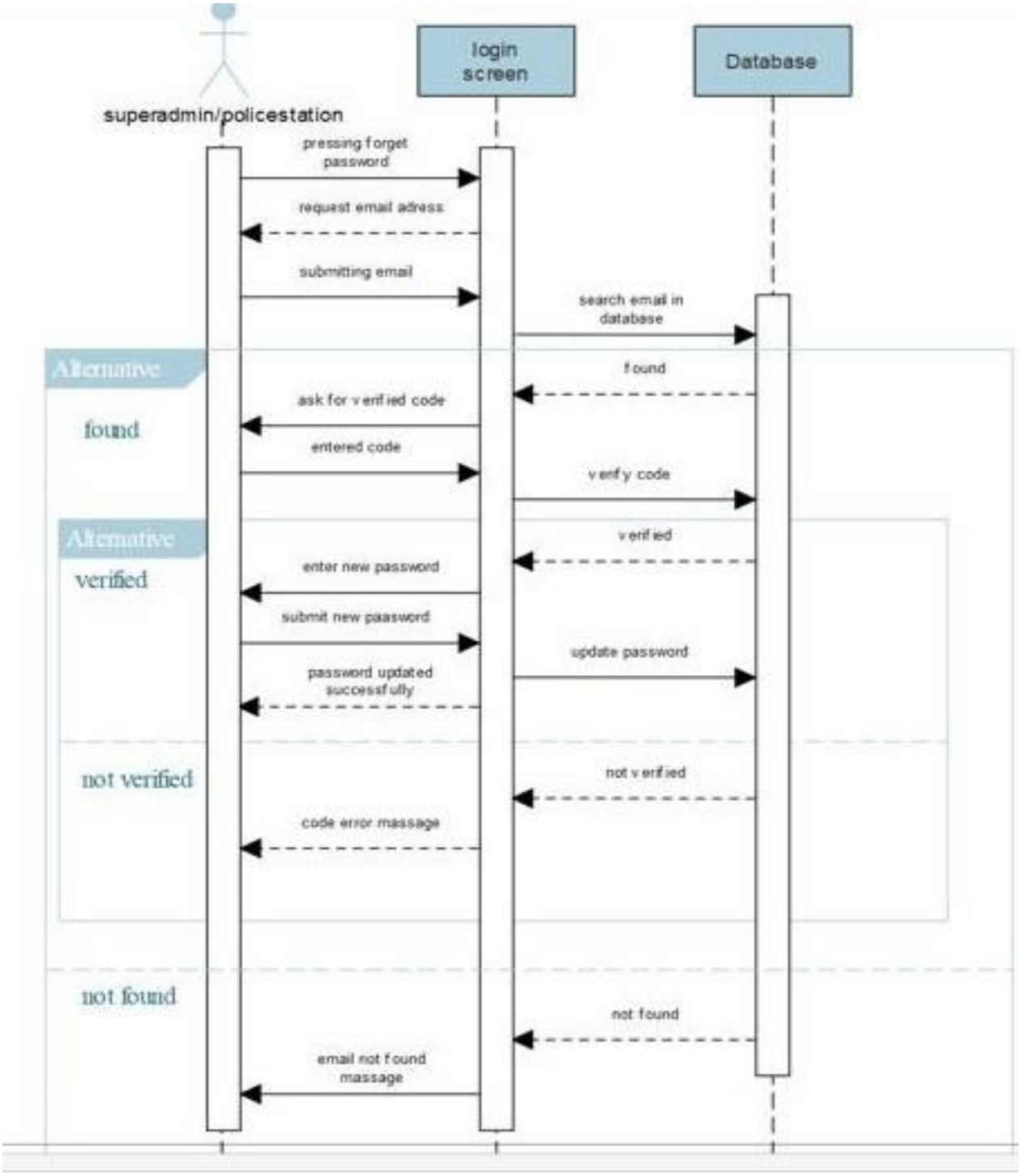
3.3.1 Registration



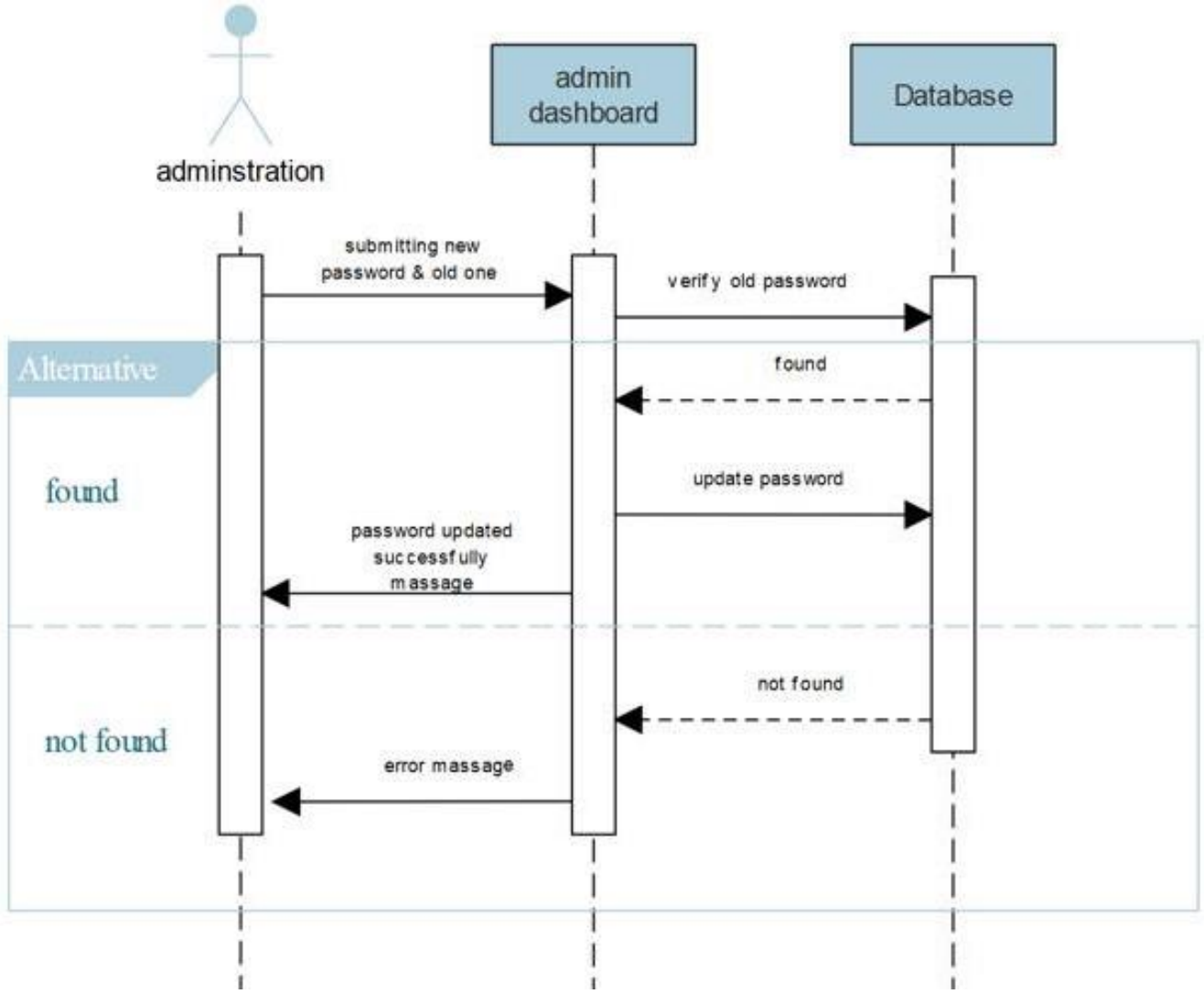
3.3.2 Login


adminstration
login screen
Database
enter email and password
check user in database
Alternative
User login successfull
found
redirect to home screen
User not found
not found
error massage

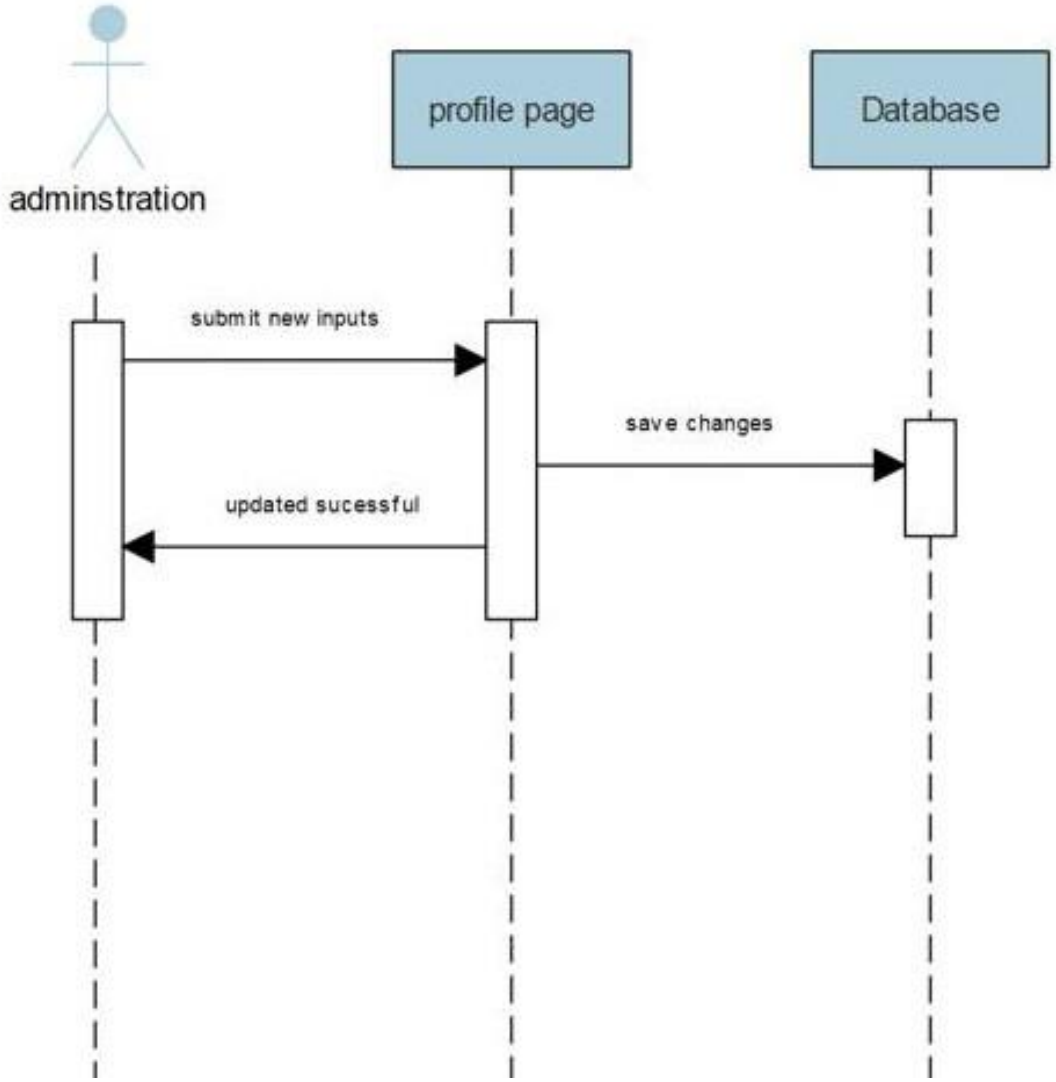
3.3.3 Forget password



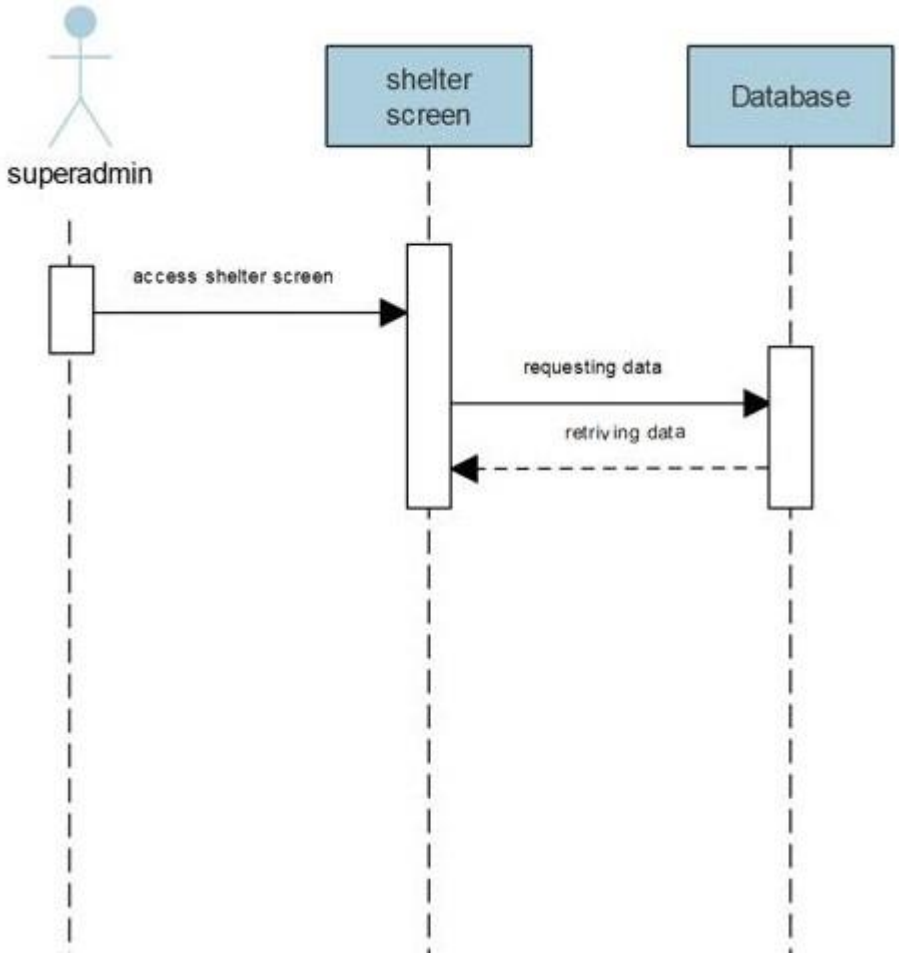
3.3.4 Update password



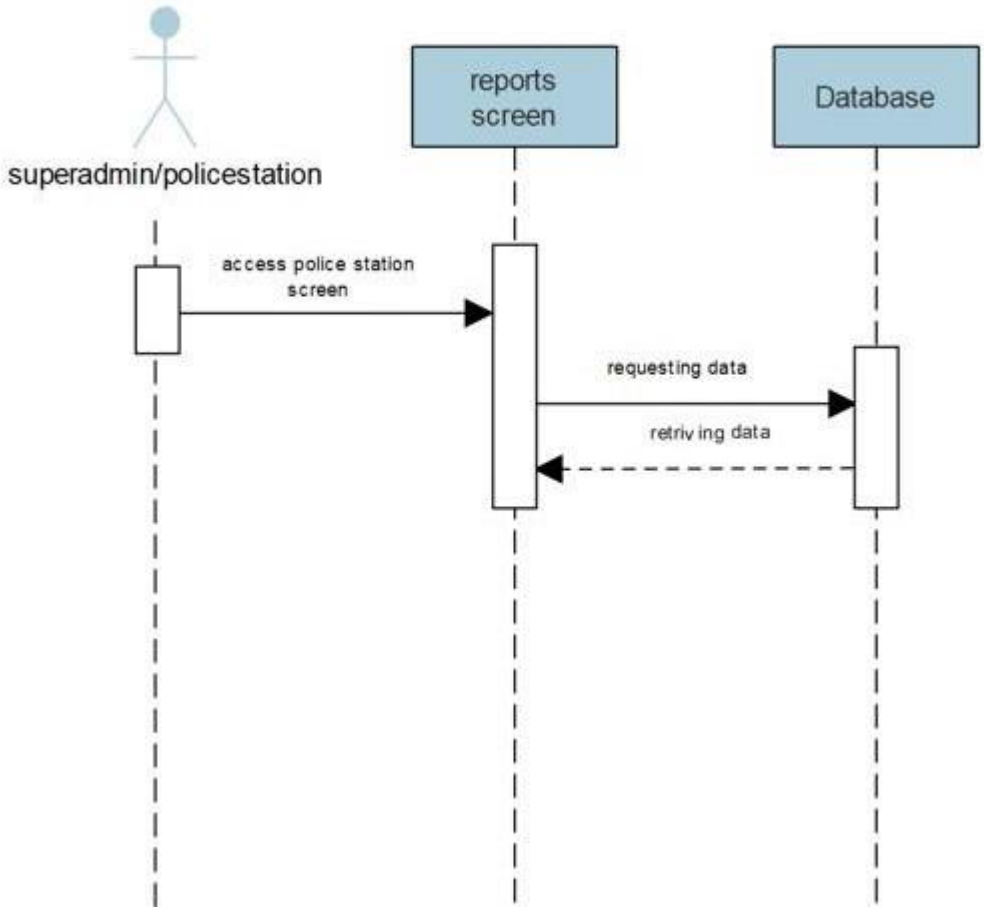
3.3.5 Update profile



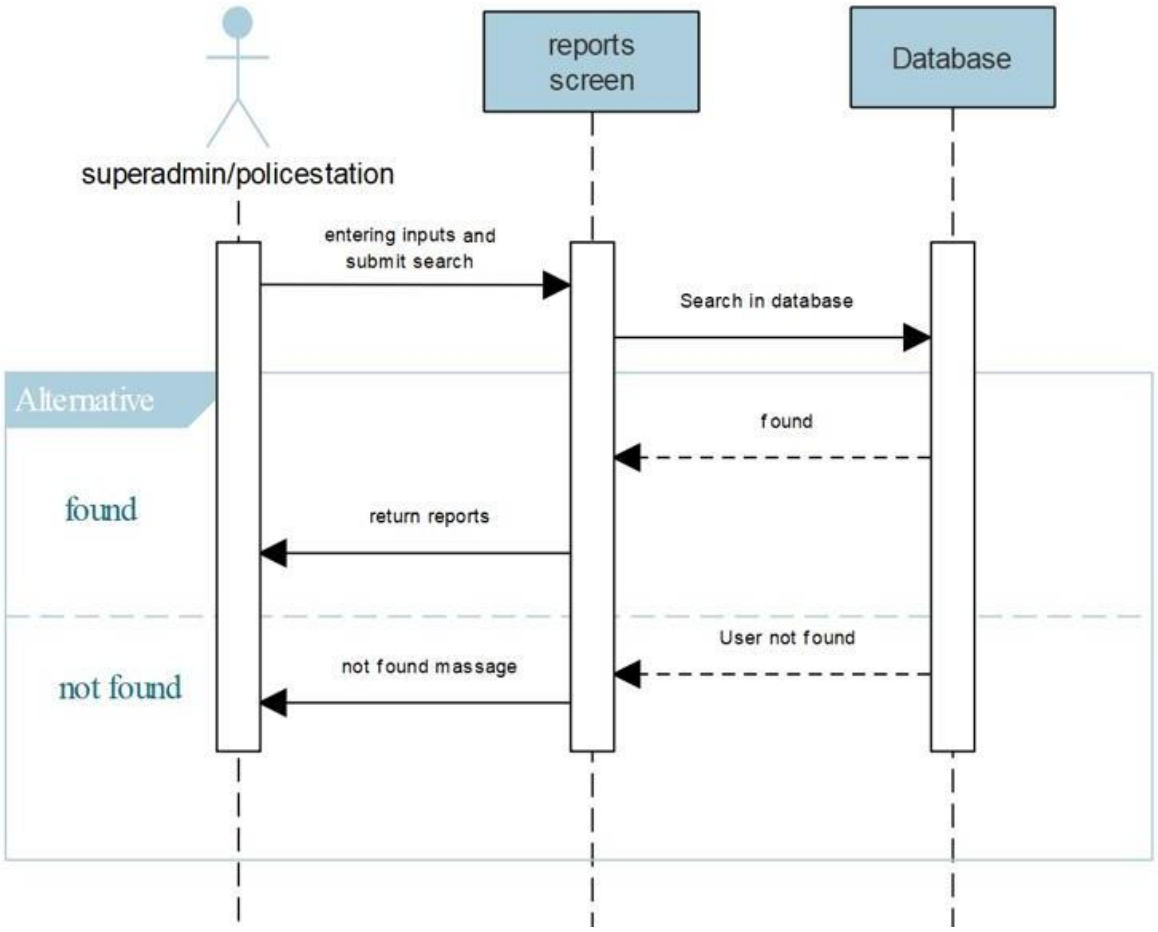
3.3.6 View shelter



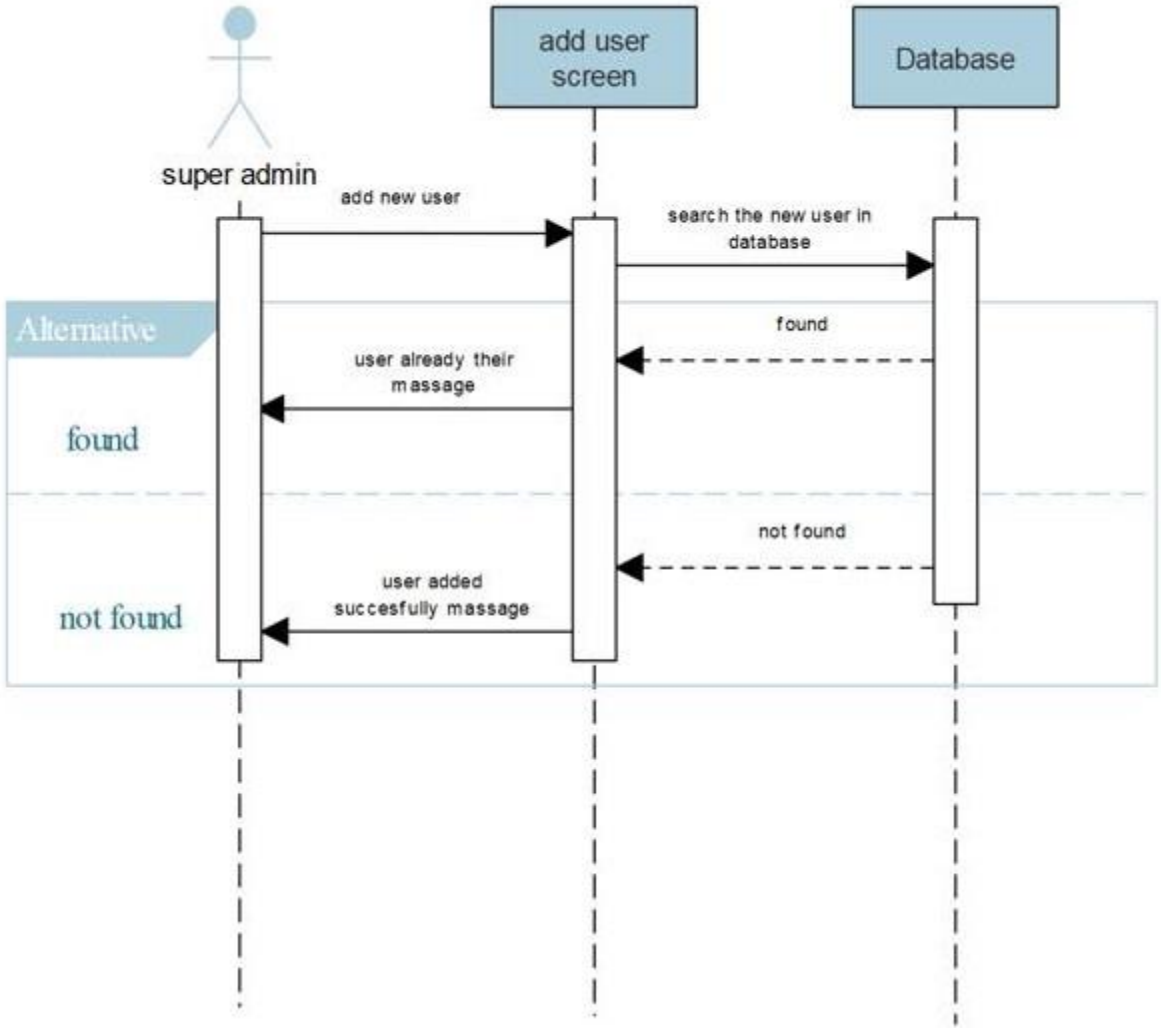
3.3.7 View report



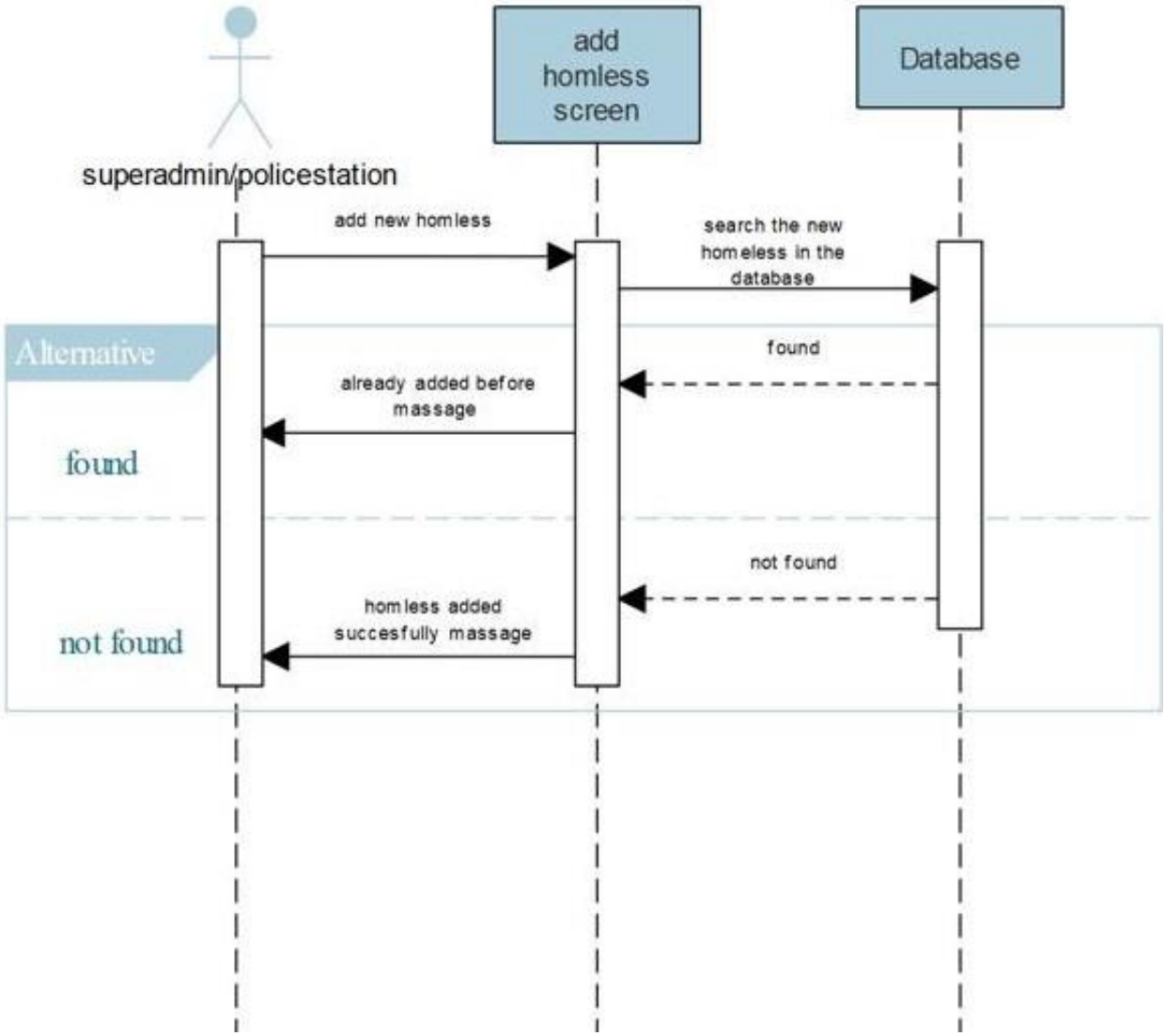
3.3.8 Search report



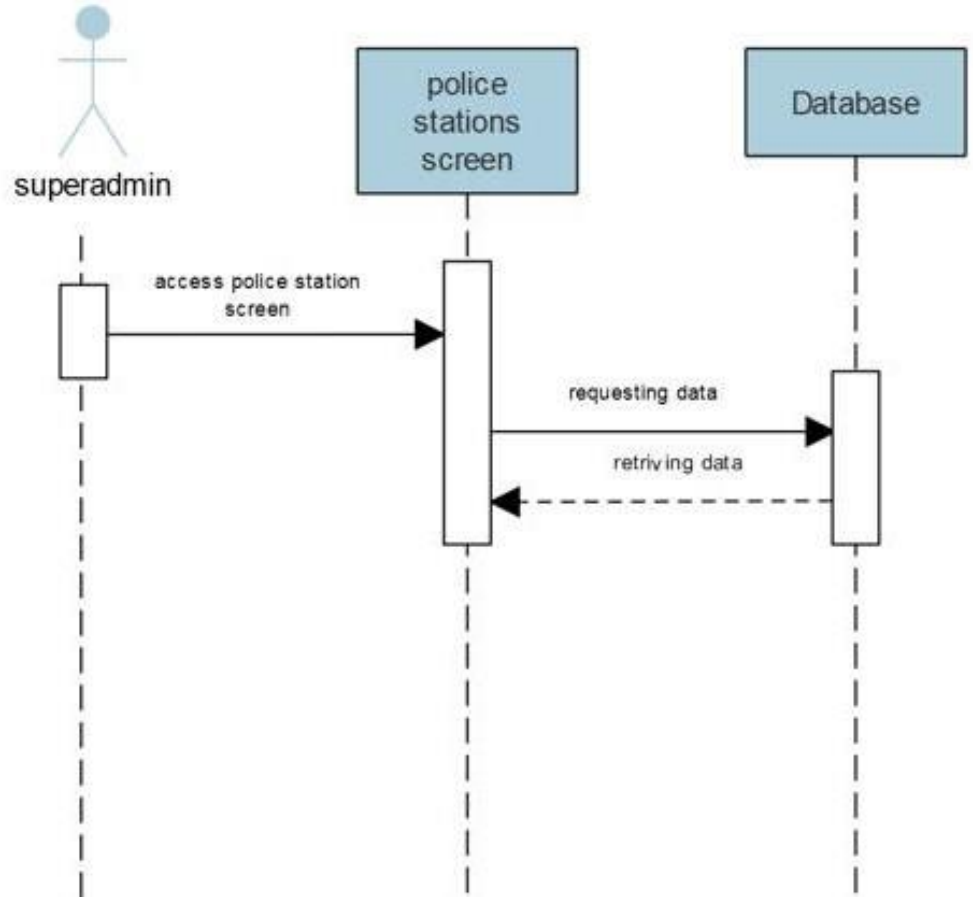
3.3.9 Add admin



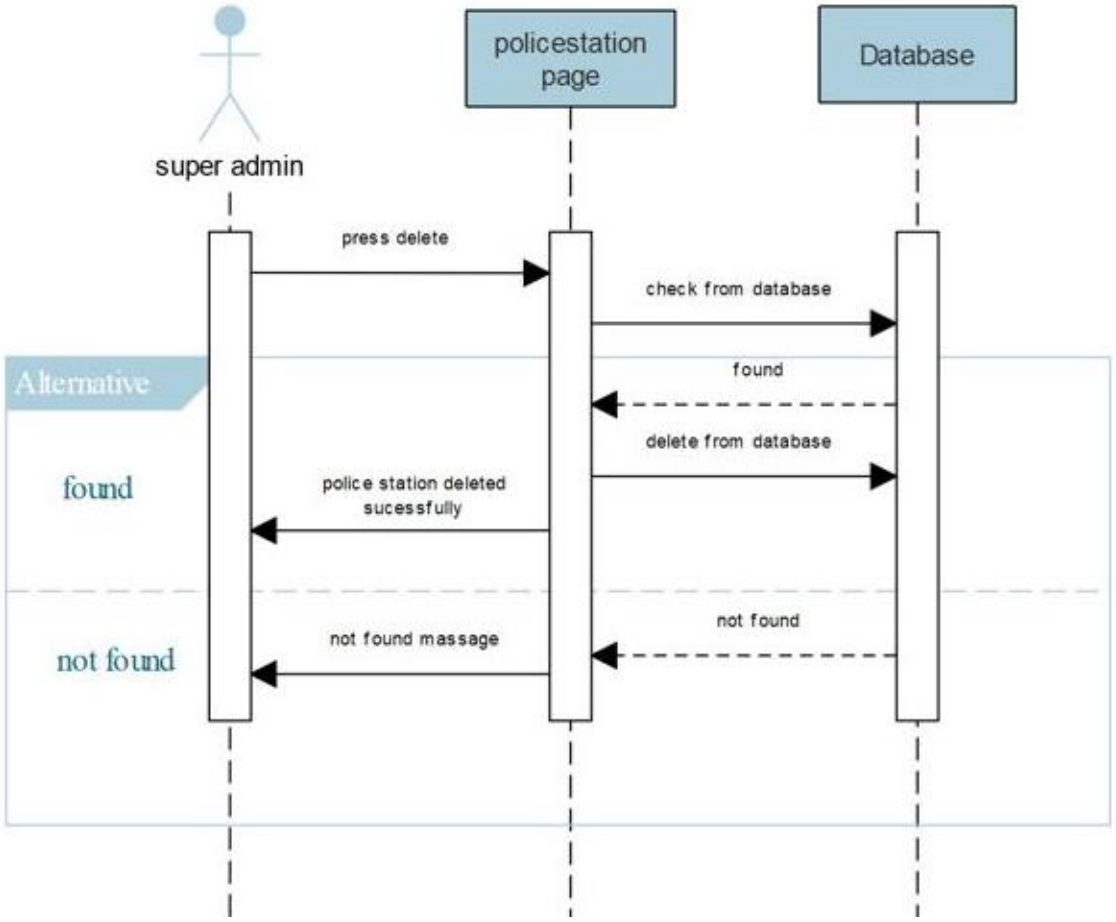
3.3.10 Add homeless



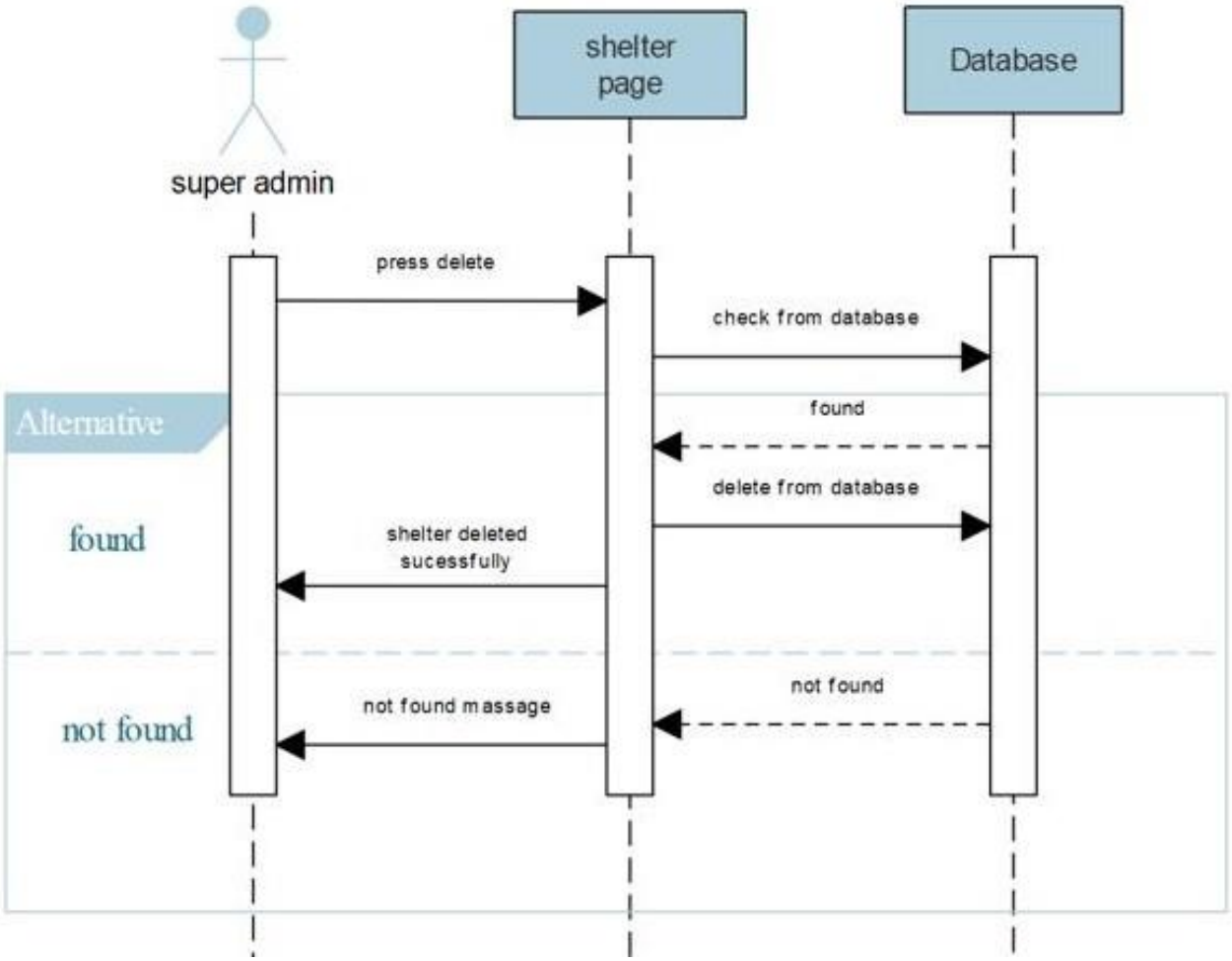
3.3.11 View police station



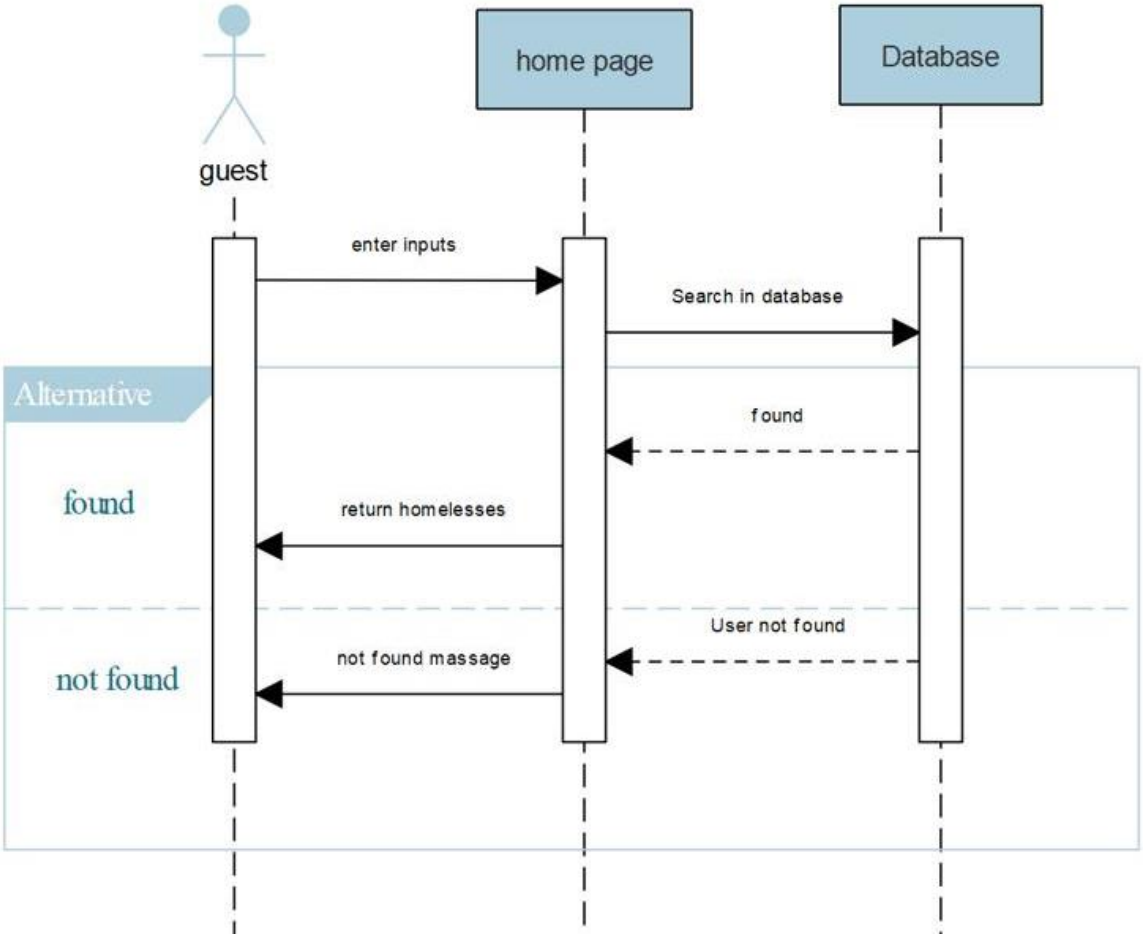
3.3.12 Delete police station



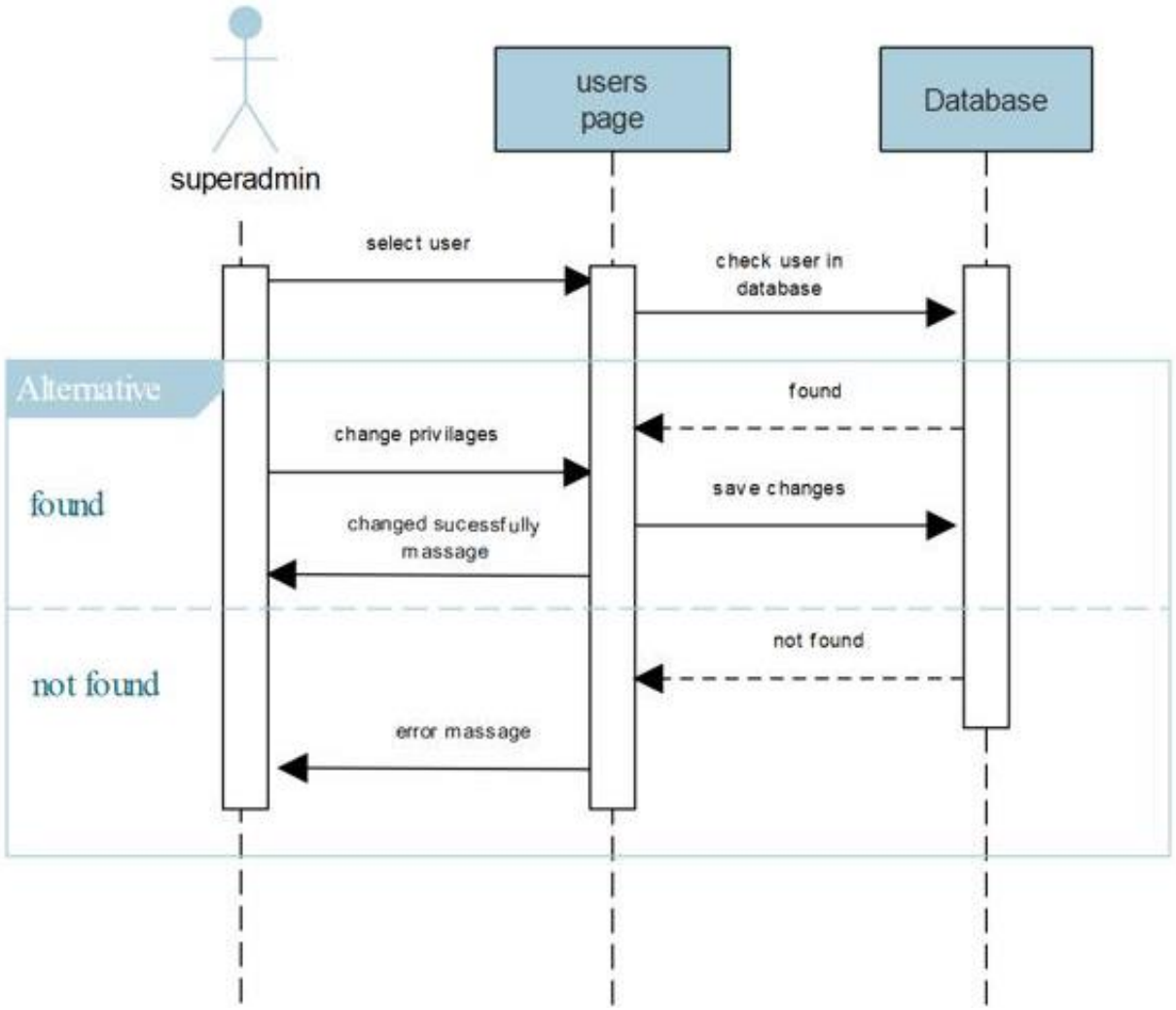
3.3.13 Delete shelter



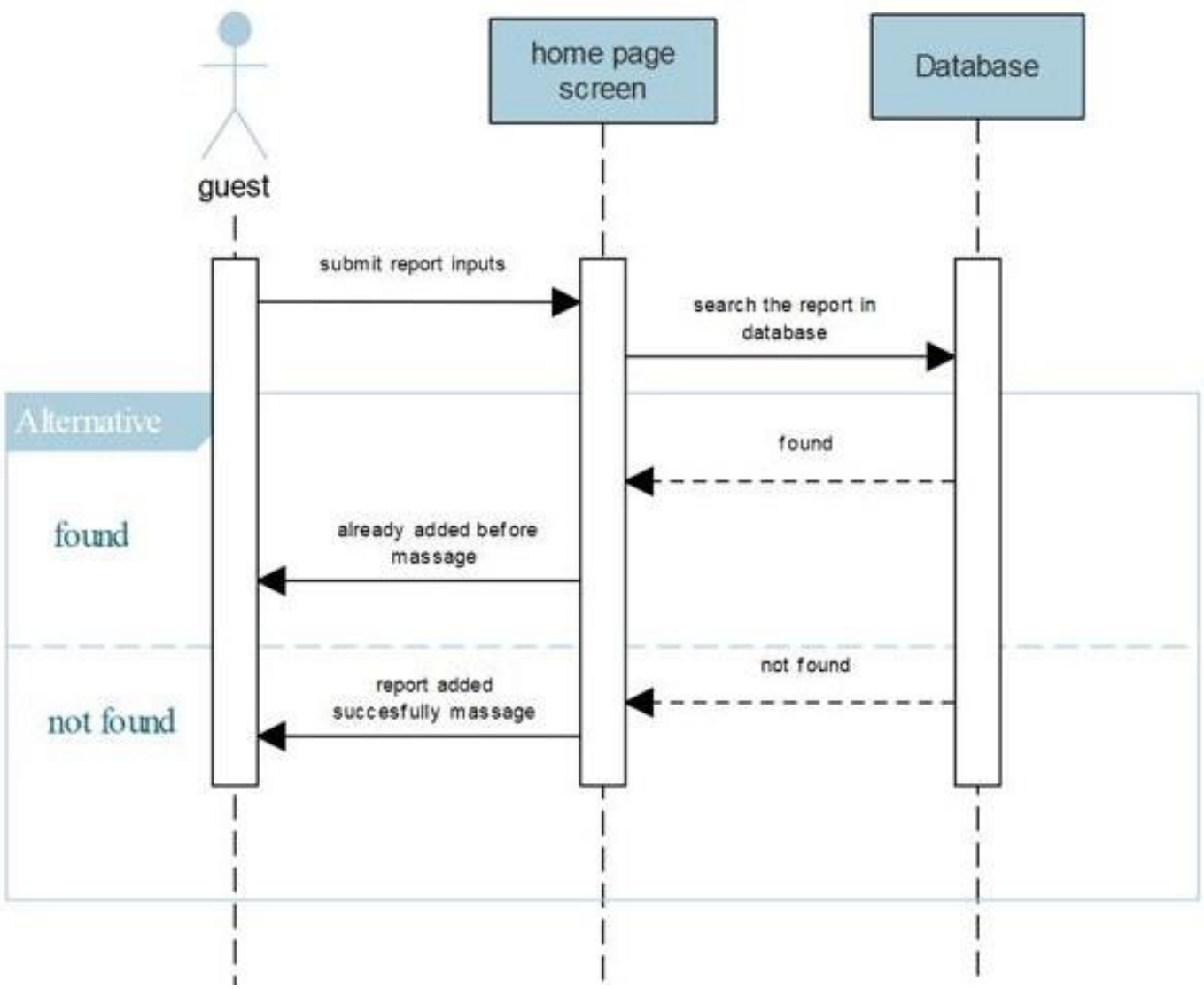
3.3.14 Search


guest
home page
Database
enter inputs
Search in database
Alternative
found
found
return homelesses
User not found
not found
not found massage

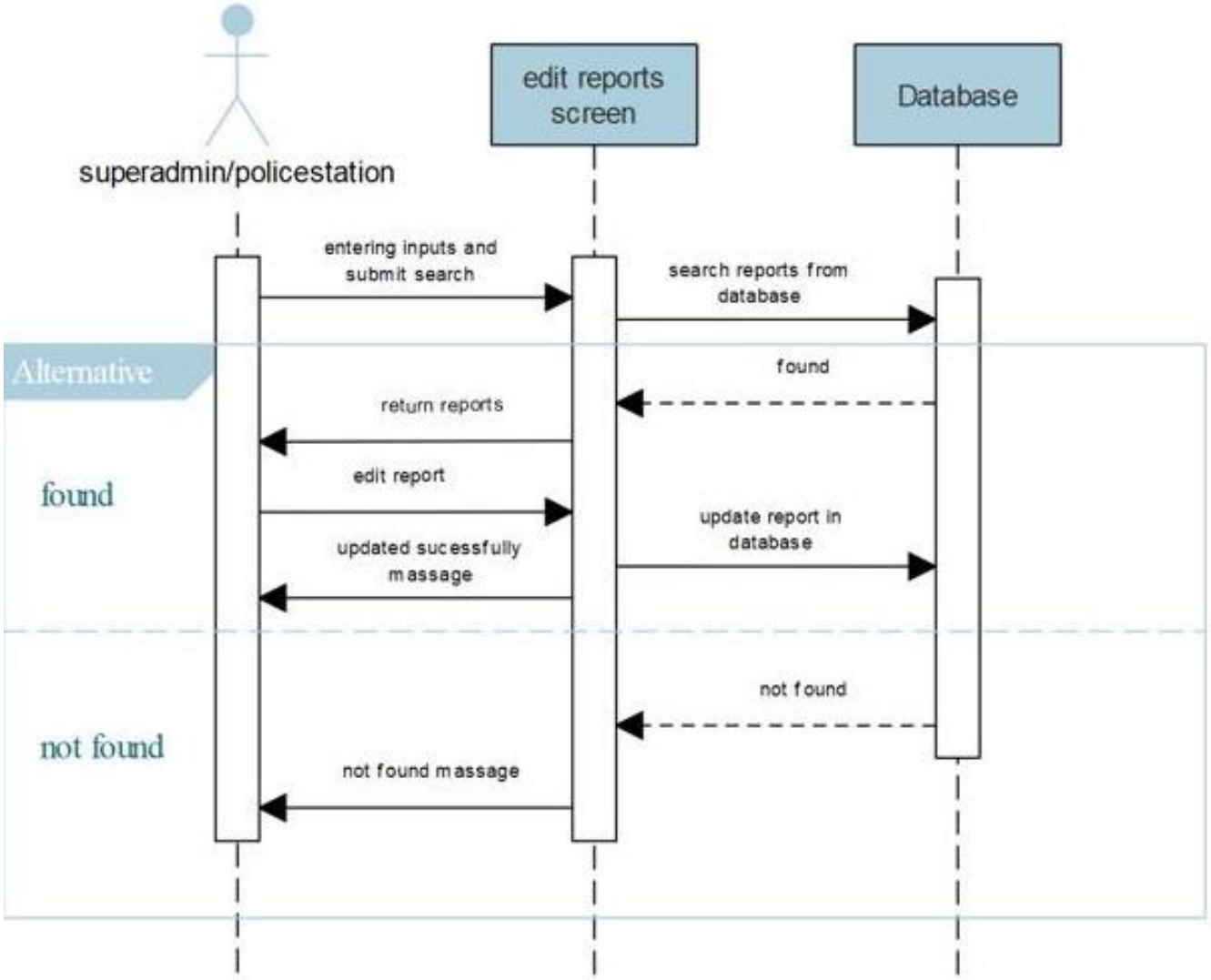
3.3.15 Change privileges



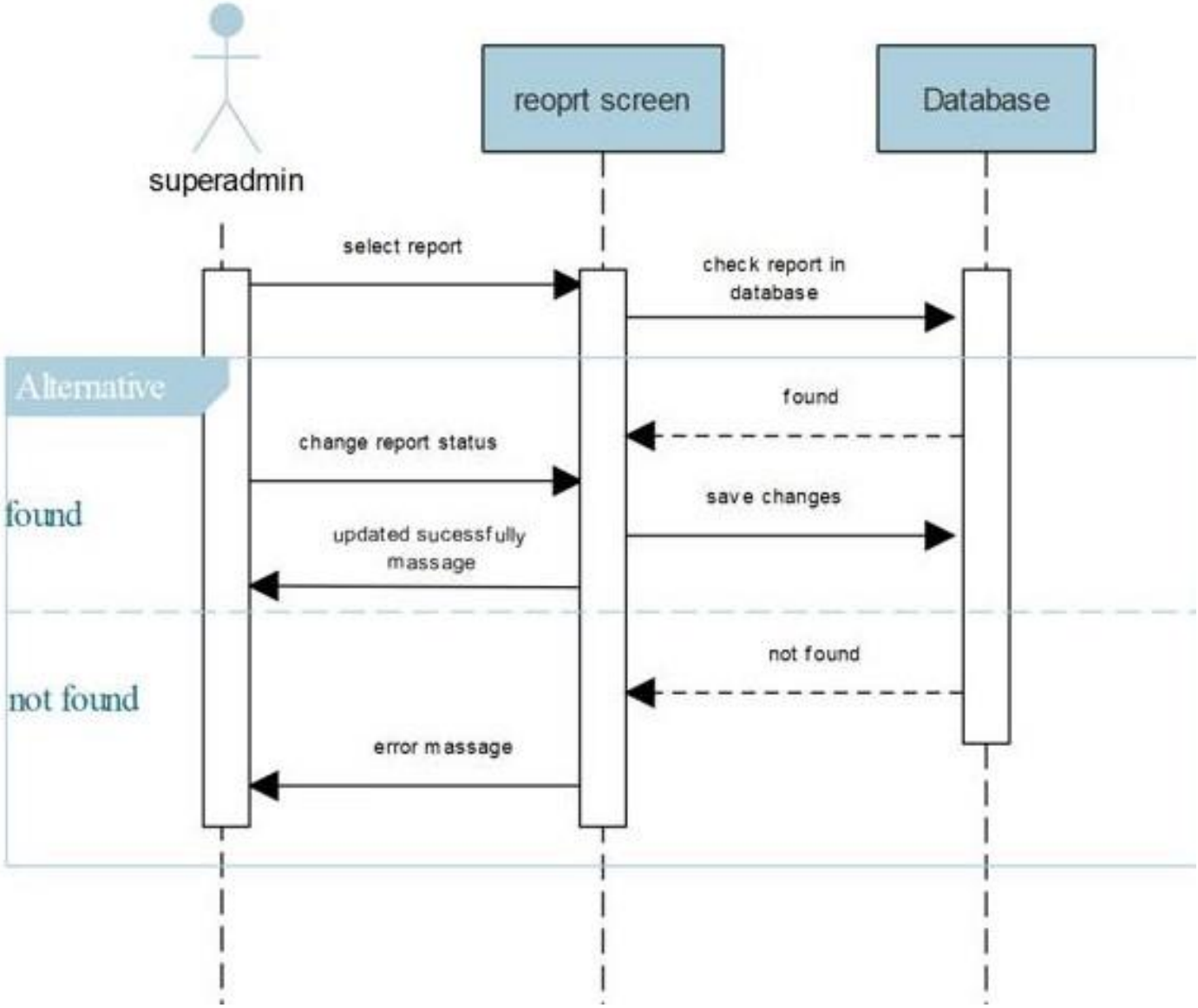
3.3.16 Submit police report



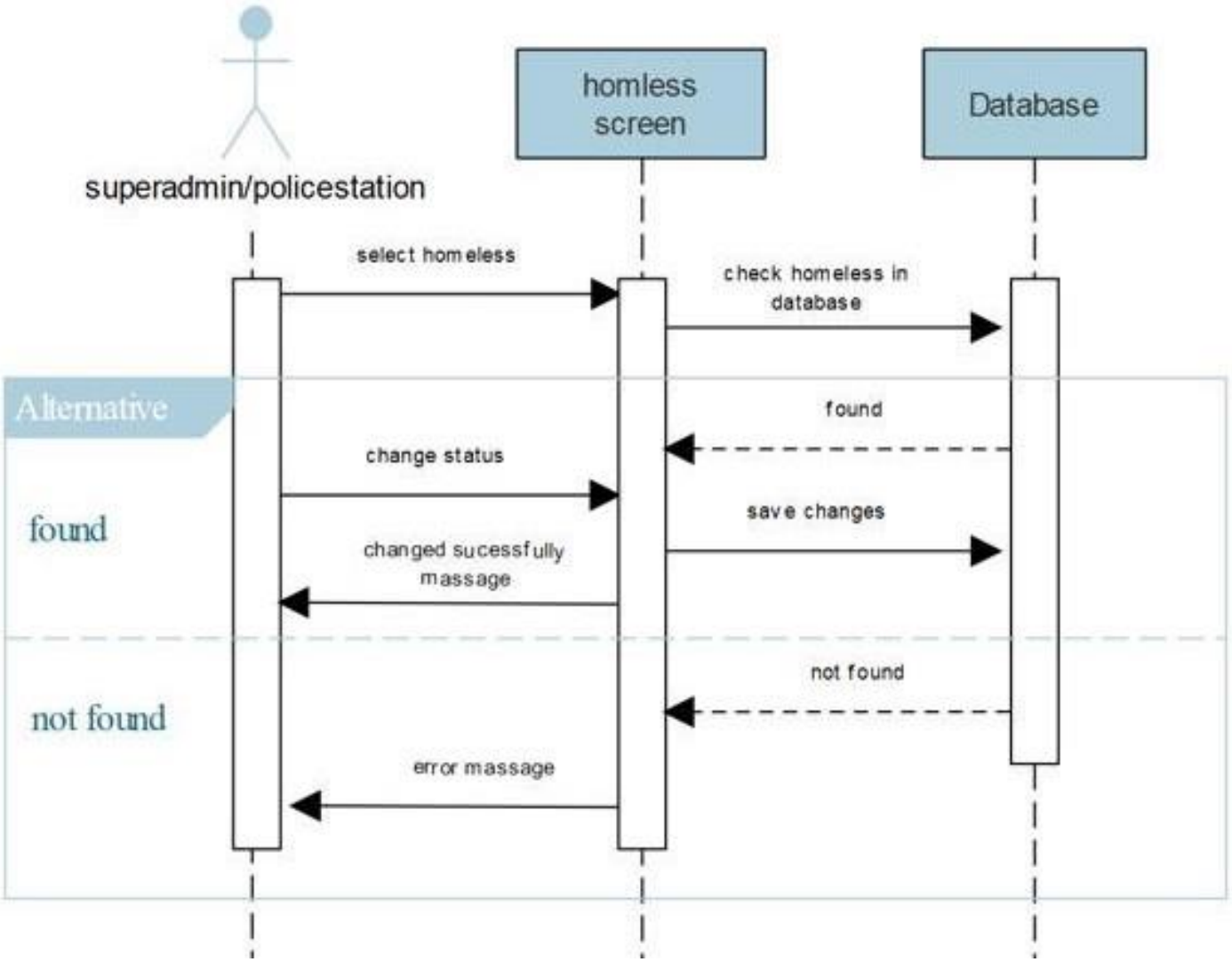
3.3.17 Edit report



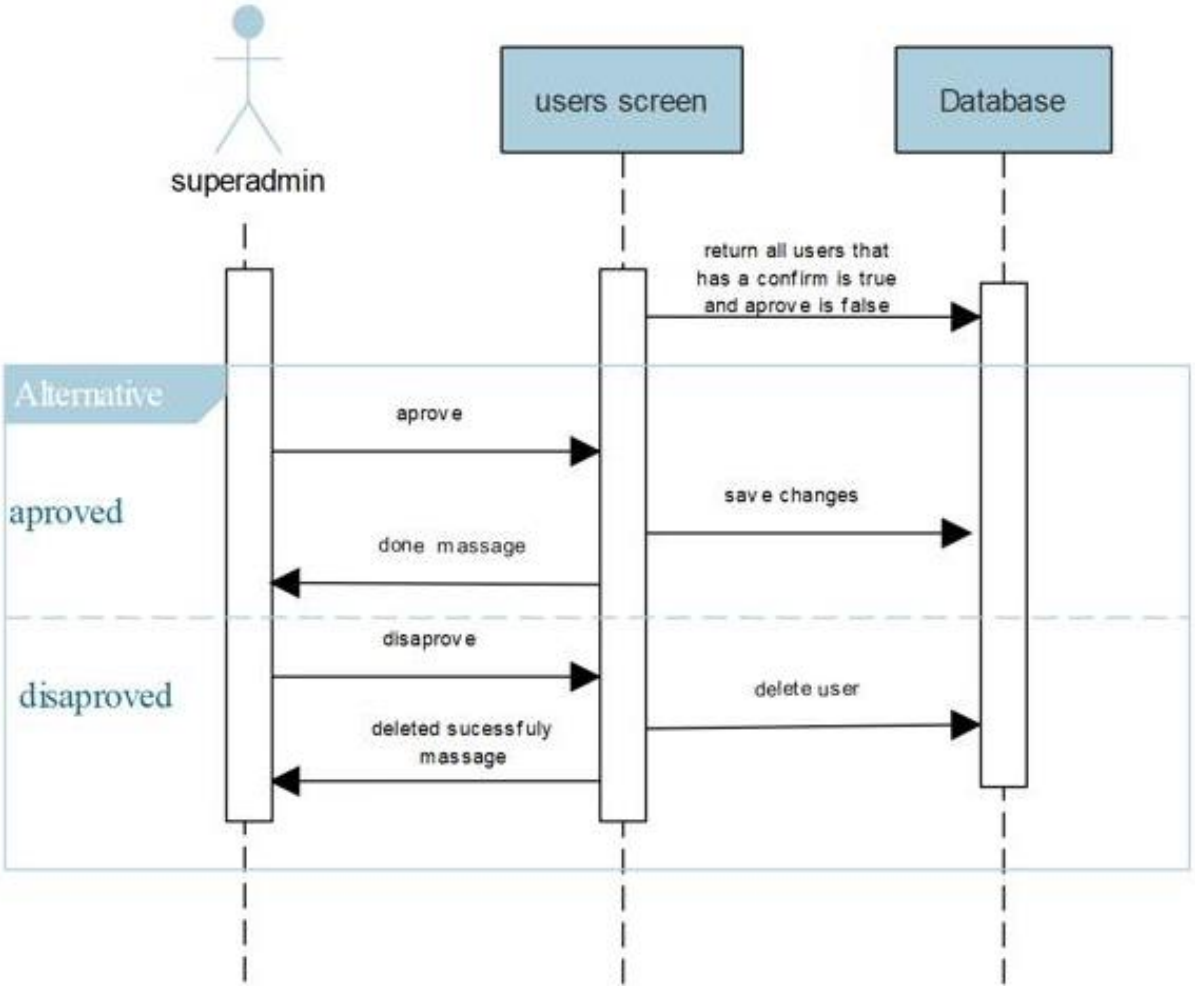
3.3.18 Change report status



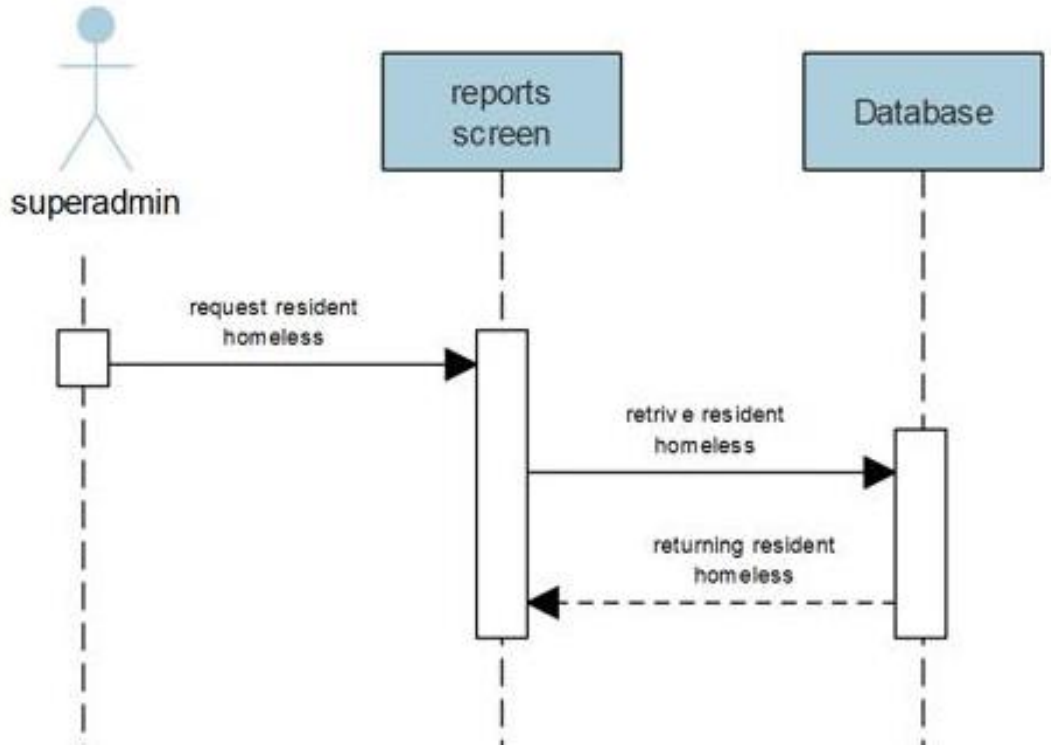
3.3.19 Close homeless



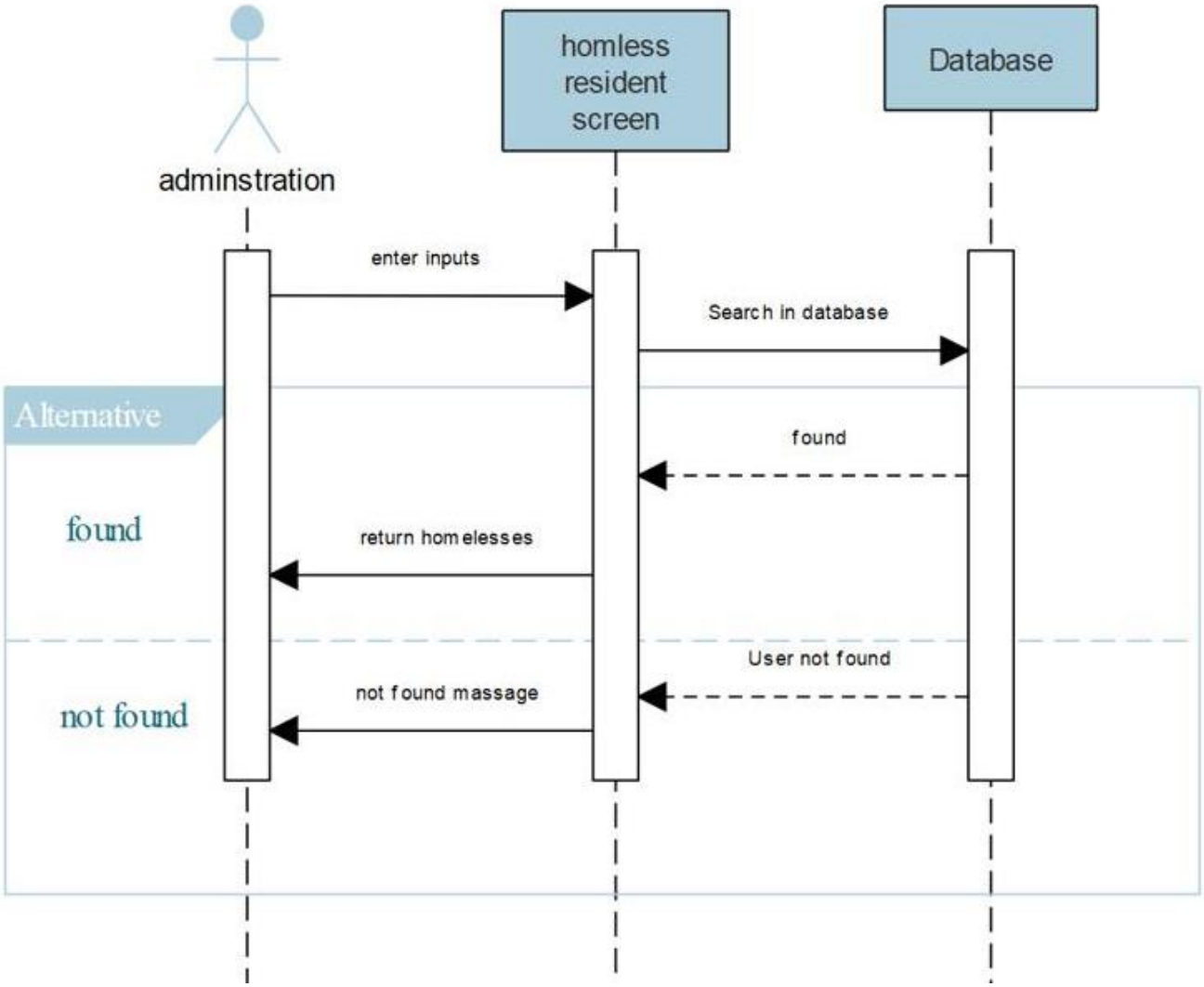
3.3.20 Check request



3.3.21 Display resident homeless

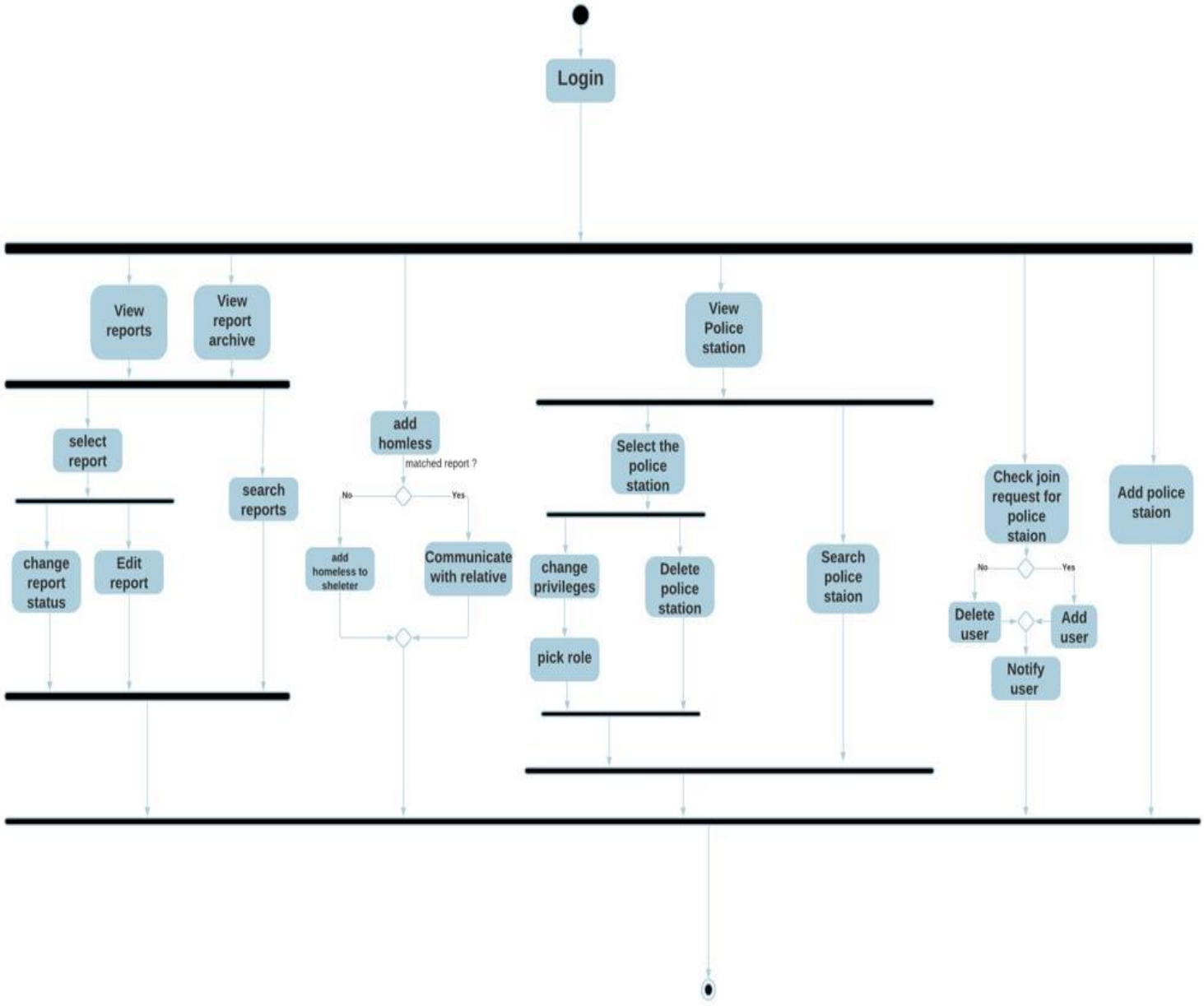


3.3.22 Search resident homeless

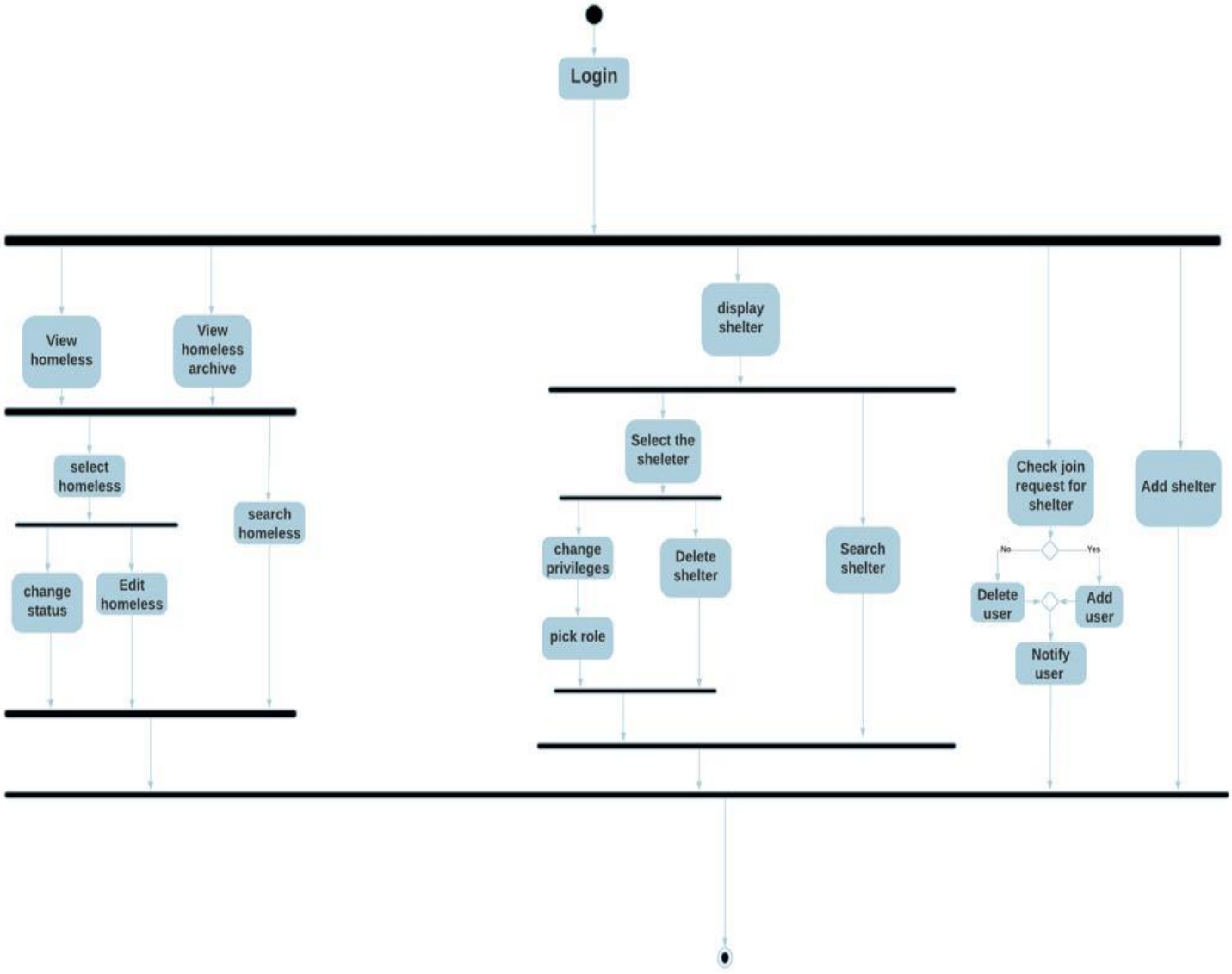


3.4 Activity Diagram

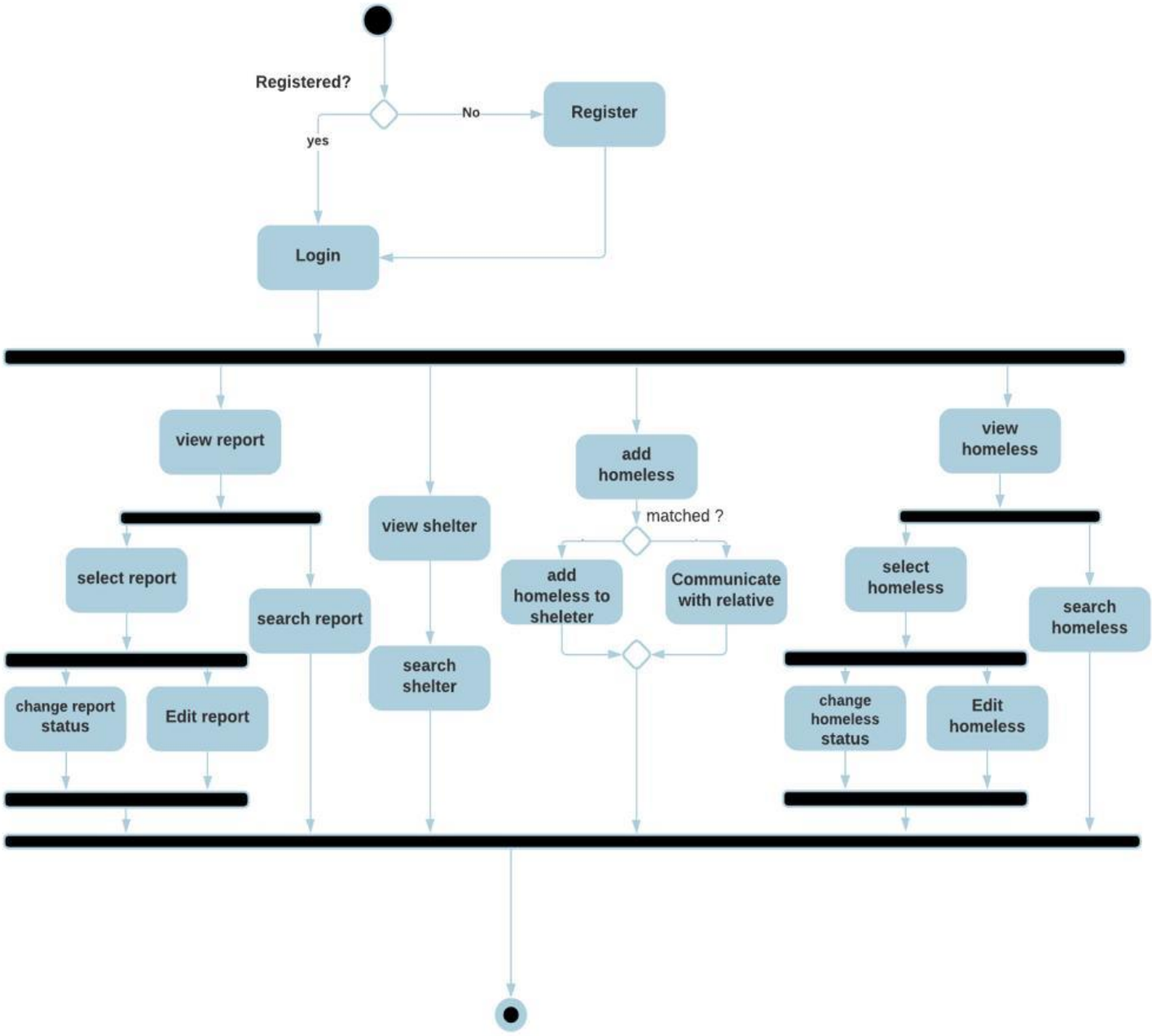
3.4.1 Super admin (1)



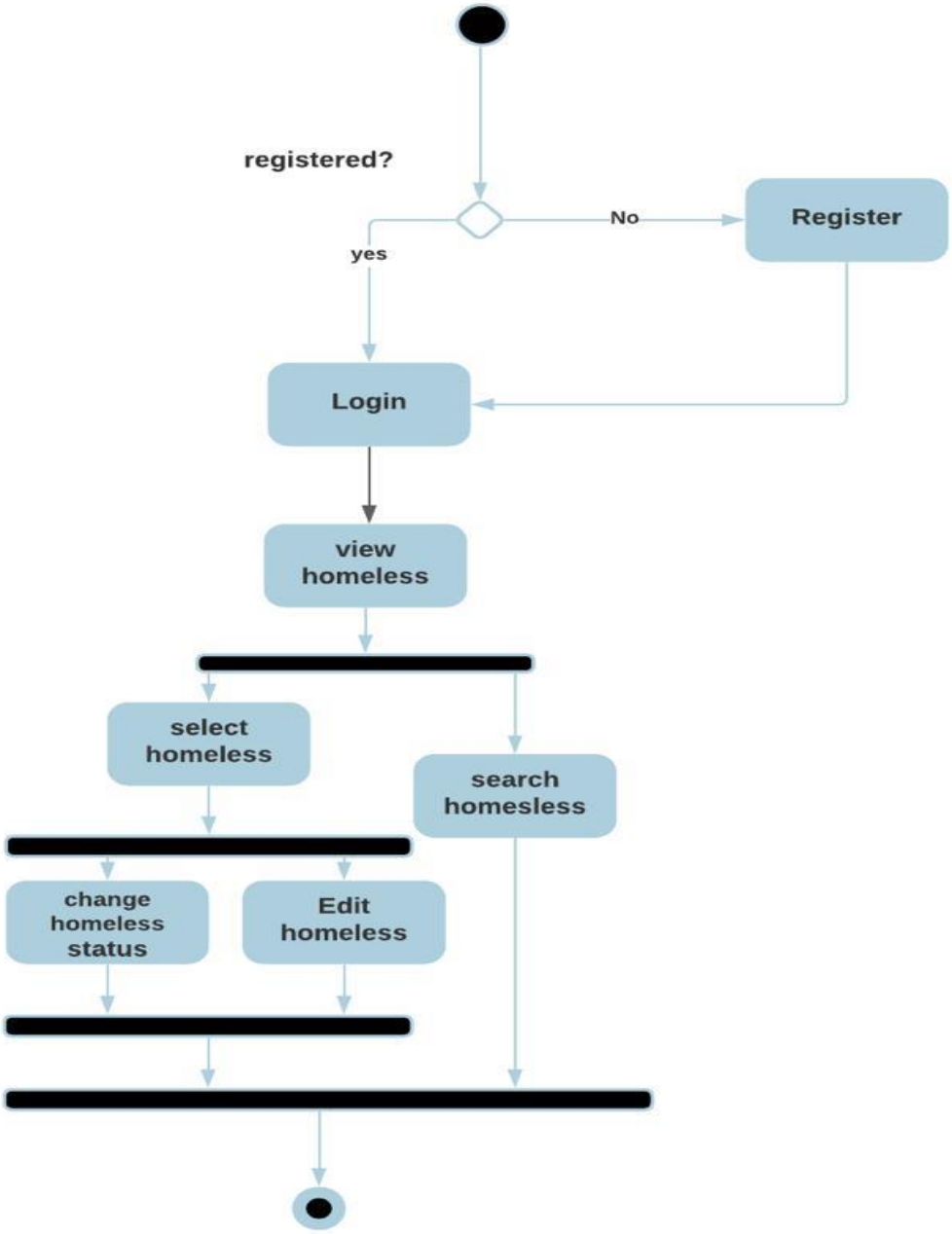
3.4.1 Super admin (2)



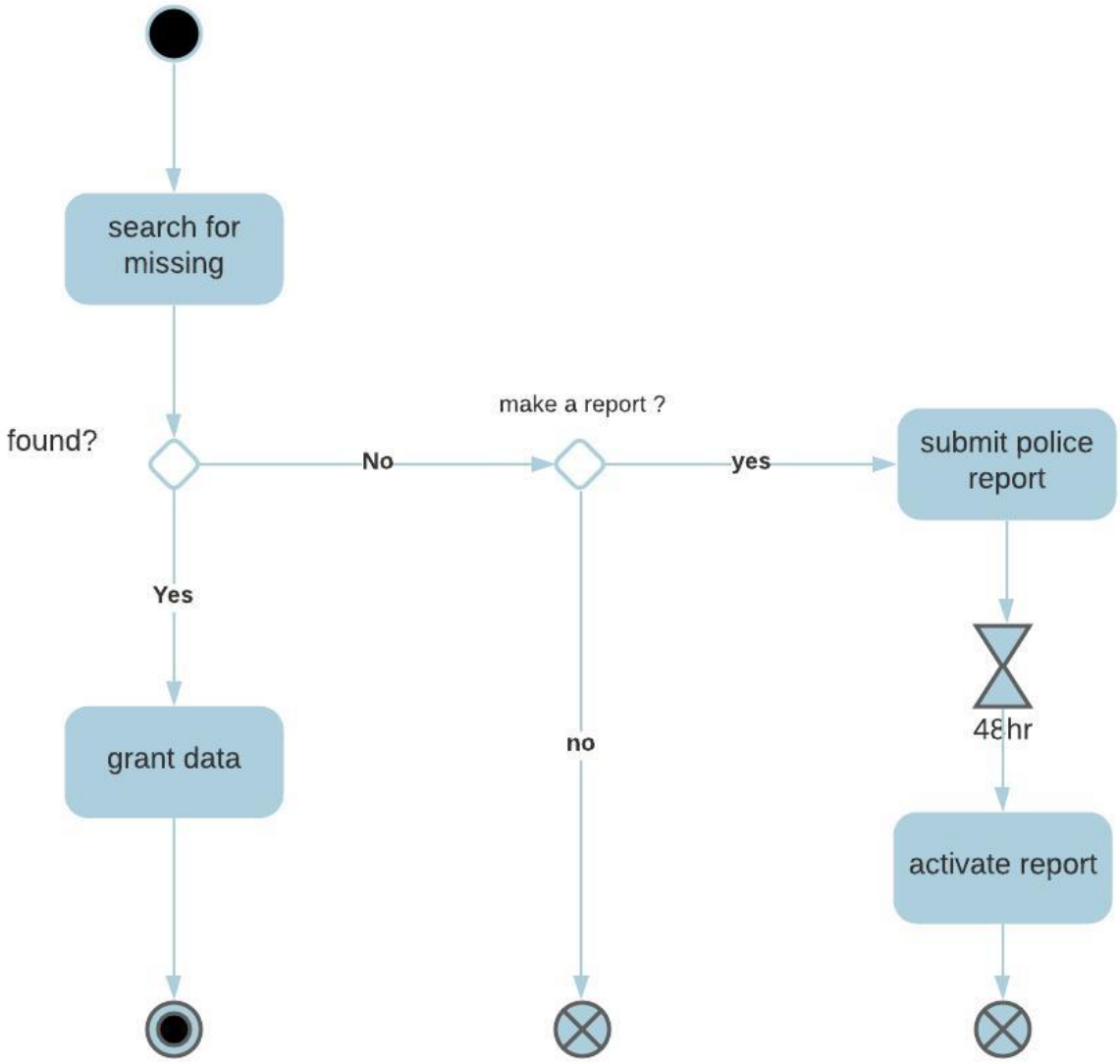
3.4.3 Police station admin

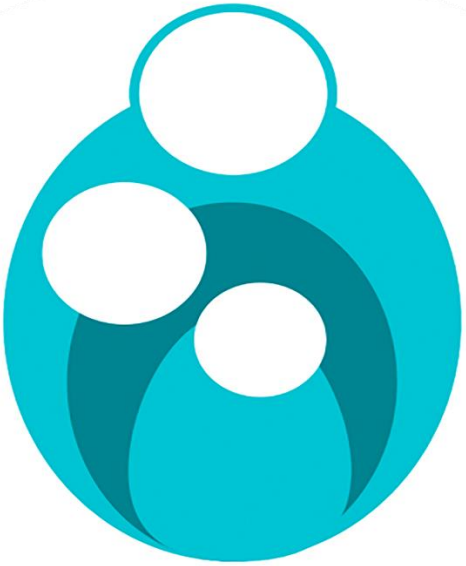


3.4.4 Shelter admin



3.4.5 Guest


search for missing
found?
No
make a report ?
yes
submit police report
Yes
grant data
no
48hr
activate report



4- Implementation

In this chapter, we will discuss application's implementation, and present its code, the UI and the algorithms and tools used to build it.

4.1 Software architecture

Sign up

Sign up(user model)

```
const mongoose = require('mongoose');
const userSchema = mongoose.Schema({
    userName:{type:String},
    email:{type:String},
    phone:{type:String},
    location:{type:String},
    role:{type:String , default:'user'},
    confirmEmail:{type:Boolean , default:false},
    aprove:{type:Boolean , default:false},
    forgetCode:{type:String},
    password:{type:String},
    imageURl:{type:String}

})
const userModel = mongoose.model("user", userSchema);
module.exports = userModel;
```

Sign up(validation)

```
const { body} = require('express-validator');

module.exports =[
    body("userName").matches(/[A-Z][a-zA-Z][^#&<>\"~;$^%{}?]{1,20}$/),
    body("email").isEmail(),
    body("phone").matches(/^01[0125][0-9]{8}$/),
    body("location").contains('https://www.google.com/maps'),
    body("role").matches(/[a-zA-Z][^#&<>\"~;$^%{}?]{1,20}$/),
    body("password").matches(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/),
    Complexity is 4 Everything is cool!
    body('Cpassword').custom((value, { req }) => {
        if (value !== req.body.password) {
          throw new Error('Password confirmation does not match password');
        }
        return true;
      })
]
```

Sign up(controller)

```
const {validationResult } = require('express-validator');
const userModel = require("../../model/user.model")
const bcrypt = require('bcrypt')
var jwt = require('jsonwebtoken');
const sendEmail = require("../email/senemail.contrroler")
Complexity is 10 It's time to do something...
module.exports = async (req, res) => {
    const { userName, email, phone, location, role, password, Cpassword } = req.body;
    try {
        const errorValidation = validationResult(req);
        if (errorValidation.isEmpty()) {
            const user = await userModel.findOne({ email });
            if (user) {
                res.json({
                    message: "email Exist",
                    oldInputs: { userName, email, phone, location, role, password, Cpassword }
                })
            } else {
                //insert User
                Complexity is 3 Everything is cool!
                await bcrypt.hash(password, 8, async (err, hash) => {
                    if (err) {
                        res.json({
                            message: "hash error",
                            oldInputs: { userName, email, phone, location, role, password, Cpassword }
                        })

                    } else {
                        await userModel.insertMany({ userName, email, phone, location, role, password: hash })
                        // send Confirm mail

                        const token = jwt.sign({ email }, 'shhhhh');
                        let message=` <a href="http://localhost:3000/confirmMessage/${token}"> confirm </a>`;
                        await sendEmail(req.body.email,message)
                        res.json({ message: 'user  registerd successfully' });
                    }
                });
            }

        } else {

            res.json({
                message: "Validation Error",
                validationError: errorValidation.array(),
                oldInputs: { userName, email, phone, location, role, password, Cpassword }
            })
        }
    } catch (error) {
        res.json({
            message: "esignUp catch",
            oldInputs: { userName, email, phone, location, role, password, Cpassword }
        })
    }

}
```

Confirm email (controller)

```
const userModel = require("../../model/user.model");
var jwt = require('jsonwebtoken');


module.exports = async (req, res) => {

    const { token } = req.params;
    console.log(token);
    try {

        if (token && token != null && token != undefined) {

            jwt.verify(token, 'shhhhh', async (err, decoded) => {
                if (err) {
                    res.json({ message: "can not  verify  token" })
                } else {
                    await userModel.updateOne({ email: decoded.email }, { confirmEmail: true });
                    console.log("confirmed");
                    res.redirect("http://localhost:4200/#/");
                }
            });
        } else {
            res.json({ message: "invalid token" })

        }

    } catch (error) {
        res.json({ message: "token error" })
    }
}
```

Login

Login(controller)

```
const { validationResult } = require('express-validator');
const userModel = require("../../model/user.model");
const bcrypt = require('bcrypt');
var jwt = require('jsonwebtoken');
Complexity is 13 You must be kidding
module.exports = async (req, res) => {
    try {
        const { email, password } = req.body;
        const loginErrors = validationResult(req);
        if (!loginErrors.isEmpty()) {
            res.json({ message: "invalid data", oldIputs: req.body, errorMessage: loginErrors.errors })
        } else {
            const user = await userModel.findOne({ email });
            if (user) {
                if (user.aprove) {
                    if (user.confirmEmail == true) {

                        const match = await bcrypt.compare(password, user.password);
                        if (match) {

                            var token = jwt.sign({
                                userRole: user.role, userName: user.userName,
                                userID: user._id, isLoggedIn: true, imag: user.imageURl,
                                location:user.location
                            }, 'shhhhh');
                            res.json({ message: "loginSucess", token });
                        } else {
                            res.json({ message: "invalid Password", oldIputs: req.body })

                        }
                    } else {
                        res.json({ message: "u have  to confirm u email First", oldIputs: req.body })
                    }
                } else {
                    res.json({ message: "pinding for  admin Aprove", oldIputs: req.body })

                }
            } else {
                res.json({ message: "not register user", oldIputs: req.body })
            }

        }
    } catch (error) {
        console.log("login catch");
        res.json({ message: "catch error" })
    }

}
```

Login (validation)

```
const { body} = require('express-validator');

module.exports =[
    body("email").isEmail(),
    body("password").matches(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/)
]
```

Forget password

Forget password (controller)

```
const userModel = require("../../model/user.model");
const sendEmail = require("../email/senemail.contrroler")
const { validationResult } = require("express-validator")
const bcrypt = require('bcrypt');
Complexity is 10 It's time to do something...
module.exports = async (req, res) => {
    const { email } = req.body;
    try {
        const errors = validationResult(req);
        if (errors.isEmpty()) {
            const user = await userModel.findOne({ email });
            if (user) {
                let randomCode = Math.random(5);
                randomCode = randomCode.toString();
                randomCode = randomCode.slice(2, 5);
                randomCode = parseInt(randomCode);

                let message = `<p> u activation  code is : ${randomCode}</p>`

                randomCode = randomCode.toString();
                Complexity is 3 Everything is cool!
                bcrypt.hash(randomCode, 6, async (err, hash) =>{
                    if (err) {
                        console.log(err);
                        res.json({message:"hash error" , err})
                    }else{
                        await userModel.updateOne({ _id: user._id }, { forgetCode: hash });
                        await sendEmail(user.email, message);
                        res.json({ message: 'done' })
                    }
                });


            } else {
                res.json({ message: 'in-valid user' });

            }
        } else {
            res.json({ message: 'validation error', oldInputs: { email }, errorMessage: errors.array() });
        }

    } catch (error) {
        res.json({ message: 'catch error',error });

    }
}
```

Forget password (validation)

```
const { body} = require('express-validator');
module.exports =[
    body("email").isEmail()
]
```

Check confirm code

Check confirm code (controller)

```
1
2   const { validationResult } = require('express-validator');
3   const userModel = require('../../model/user.model');
4   const bcrypt = require('bcrypt');
    Complexity is 9 It's time to do something...
5   module.exports = async (req, res) => {
6       const { email, code } = req.body;
7       try {
8           const errors = validationResult(req);
9           if (errors.isEmpty()) {
10              const user = await userModel.findOne({ email });
11              if (user) {
12
13                  const match = await bcrypt.compare(code, user.forgetCode);
14
15                  if (match) {
16                      res.json({
17                          message: "matched code"
18                      })
19                  } else {
20                      res.json({
21                          message: "invalid code"
22                      })
23                  }
24              } else {
25                  res.json({
26                      message: "invalid user"
27                  })
28              }
29          } else {
30              res.json({
31                  message: "validation error",
32                  oldInputs: { email, code },
33                  errorMessage: errors.array()
34              })
35          }
36      } catch (error) {
37          res.json({
38              message: "catch error"
39          })
40      }
41  }
```

Check confirm code (validation)

```
const { body} = require('express-validator');

module.exports =[
    body("email").isEmail(),
    body("code").isNumeric(),
]
```

Change password

Change password (controller)

```
const { validationResult } = require('express-validator');
const userModel = require('../../model/user.model');
const bcrypt = require('bcrypt');
Complexity is 13 You must be kidding
module.exports = async (req, res) => {
    const { email, code, password, cPassword } = req.body;
    try {
        const errors = validationResult(req);
        if (errors.isEmpty()) {
            const user = await userModel.findOne({ email });
            if (user) {

                const match = await bcrypt.compare(code, user.forgetCode);

                if (match) {

                    Complexity is 4 Everything is cool!
                    bcrypt.hash(password, 8, async (err, hash) => {
                        if (err) {
                            res.json({ message: "hash error" });
                        } else {
                            let randomCode = Math.random(5);
                            randomCode = randomCode.toString();
                            randomCode = randomCode.slice(2, 5);
                            randomCode = parseInt(randomCode);
                            randomCode = randomCode.toString();
                            bcrypt.hash(randomCode, 6, async (err, hashCode) => {
                                await userModel.updateOne({ _id: user._id }, { password: hash, forgetCode: hashCode })
                                res.json({ message: "password updated Successfully" });
                            });
                        }
                    });
                } else {
                    res.json({
                        message: "invalid code"
                    })
                }
            } else {
                res.json({
                    message: "invalid user"
                })
            }
        } else {
            res.json({
                message: "validation error",
                oldInputs: { email, code, password, cPassword },
                errorMessage: errors.array()
            })
        }
    } catch (error) {
        res.json({
            message: "catch error"
        })
    }
}
```

Change password (validation)

```
const { body} = require('express-validator');

module.exports =[
    body("email").isEmail(),
    body("code").isNumeric(),
    body("password").matches(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/),
    Complexity is 4 Everything is cool!
    body('cPassword').custom((value, { req }) => {
        if (value !== req.body.password) {
          throw new Error('Password confirmation does not match password');
        }
        return true;
      })
]
```

Update password

Update password (controller)

```
const { validationResult } = require("express-validator");
const userModel = require("../../model/user.model");
const bcrypt = require("bcrypt");
Complexity is 12 You must be kidding
module.exports = async (req, res) => {
    const { oldPassword , password, cPassword } = req.body
    try {
        const error = validationResult(req);
        if (error.isEmpty()) {
            const user = await userModel.findOne({ _id: req.userID });
            if (user) {

                const match = await bcrypt.compare(oldPassword,user.password);
                if (match) {
                 Complexity is 3 Everything is cool!
                 bcrypt.hash(password, 8, async (err, hash) => {
                    if (err) {
                        res.json({ message: "hash error" });
                    } else {
                        await userModel.updateOne({ _id: user._id }, { password: hash })
                        res.json({ message: "password updated Successfully" });
                    }
                 });
                }else{
                 res.json({ message: "wrong old Password" })
                }
            } else {
                res.json({ message: "in-valid user" })
            }
        } else {
            res.json({
                message: "validation-error",
                oldInputs: {oldPassword, password, cPassword },
                errorMessage: error.array()
            })
        }
    } catch (error) {
        res.json({ message: "catch errorddddd" , error})
    }
}
```

Update password (validation)

```
const { body} = require('express-validator');

module.exports =[
  body("oldPassword").matches(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/),
    body("password").matches(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/),

    body('cPassword').custom((value, { req }) => {
      if (value !== req.body.password) {
        throw new Error('Password confirmation does not match password');
      }
      return true;
    })
]
```

Update profile image

Update profile image (controller)

```
controller > signin > JS uploadImag.controller.js > ...
const { validationResult } = require("express-validator");
const userModel = require("../../model/user.model");
const bcrypt = require("bcrypt");
var jwt = require('jsonwebtoken');


module.exports = async (req, res) => {
    try {
        let imageURl;
        if (req.file == undefined) {
            res.json({ message: "in-valid image" })
        } else {
            imageURl = req.file.path;
        }
        const user = await userModel.findOne({ _id: req.userID });
        if (user) {


            await userModel.updateOne({ _id: user._id }, {imageURl})
            var token = jwt.sign({
                userRole: user.role, userName: user.userName,
                userID: user._id, isLoggedIn: true, imag:imageURl,
                location:user.location
            }, 'shhhhh');
            res.json({ message: "password updated Successfully"  ,token, imageURl});



        } else {
            res.json({ message: "in-valid user" })

        }

    } catch (error) {
        res.json({ message: "catch errorddddd", error })
    }
}
```

Update username and profile

Update username and profile(controller)

```
const { validationResult } = require("express-validator");
const userModel = require("../../model/user.model");
var jwt = require('jsonwebtoken');
Complexity is 7 It's time to do something...
module.exports = async (req, res) => {
    const {userName,location}= req.body;
    try {
        const user = await userModel.findOne({ _id: req.userID });
        const validation =  validationResult(req);
        if (validation.isEmpty()) {
            if (user) {


                await userModel.findByIdAndUpdate({ _id: user._id }, {userName,location})
                var token = jwt.sign({
                    userRole: user.role, userName:userName,
                    userID: user._id, isLoggedIn: true, imag: user.imageURl,
                    location:location
                }, 'shhhhh');
                res.json({ message: "Updated successfully"  ,token, info:{userName,location}});



            } else {
                res.json({ message: "in-valid user" })

            }
        } else {
            res.json({message:"validation error" , errMessage:validation.array()});
        }


    } catch (error) {
        res.json({ message: "catch error", error })
    }
}
```

Update username and profile(validation)

```
const { body} = require('express-validator');

module.exports =[
    body("userName").matches(/[A-Z][a-zA-Z][^#&<>\"~;$^%{}?]{1,20}$/),
    body("location").contains('https://www.google.com/maps')
]
```

Homeless model

```
const mongoose = require('mongoose');

const homelessSchema = mongoose.Schema({
    // homeless info
    name:{type:String},
    age:{type:Number},
    gender:{type:String},
    imageURl:{type:String},
    description:{type:String},
    foundlocation:{type:String},
    foundTime:{type:Date},
    //shelter info
    shelterID:{type:mongoose.Schema.Types.ObjectId , ref:"user"},
    policeSationID:{type:mongoose.Schema.Types.ObjectId , ref:"user" },
    reportID:{type:mongoose.Schema.Types.ObjectId , ref:"report"},

    //founder info
    finderName:{type:String},
    finderNationID:{type:String},
    finderPhone:{type:String},
    finderEmail:{type:String},
    //status
    status:{type:String, default:"undefined"}

})


const homelessModel = mongoose.model("homeless", homelessSchema);
module.exports = homelessModel;
```

Add homeless

Add homeless (controller)

```
const homelessModel = require("../../model/homeless.model");
const { validationResult } = require('express-validator');
const bcrypt = require('bcrypt');
const CryptoJS = require("crypto-js");

const userModel = require("../../model/user.model");
Complexity is 11 You must be kidding
module.exports = async (req, res) => {
    let imageURl;
    if (req.file == undefined) {
        res.json({ message: "in-valid image" })
    } else {
        imageURl = req.file.path;
    }
    const { name, age, gender, description, foundlocation, foundTime, shelterName,
        finderName, finderNationID, finderPhone, finderEmail } = req.body;
    let policeSationID = req.userID;
    try {
        const validationError = validationResult(req);
        if (validationError.isEmpty()) {
            const homeless = await homelessModel.findOne({ name });
            if (homeless) {
                res.json({ message: "already exist", homeless })
            } else {
                let shelter = await userModel.findOne({ userName: shelterName })
                if (shelter) {
                    let ciphertext =  CryptoJS.AES.encrypt(finderNationID, 'secret key 123').toString();

                            await homelessModel.insertMany({
                                name, age, gender, imageURl, description,
                                foundlocation, foundTime, shelterID: shelter._id, policeSationID,
                                finderName, finderNationID: ciphertext, finderPhone, finderEmail
                            });
                            res.json({ message: "added successfully" });
                } else {
                    res.json({
                        message: "in-valid shelter id", oldInputs: {
                            name, age, gender, imageURl, description,
                            foundlocation, foundTime, shelterName, policeSationID,
                            finderName, finderNationID, finderPhone, finderEmail
                        }
                    })
                }

            }
        } else {
            res.json({…
            });
        }

    } catch (error) {
        res.json({ message: "catch err", error });
    }

}
```

Add homeless (validation)

```
const { body} = require('express-validator');
module.exports =[
    body("name").matches(/^[\u0621-\u064A\-\ ][^#&<>\"~;$^%{}?]{2,20}$/),
    body("age").isNumeric(),
    body("gender").isString(),
    // body("imageURl").isString(),
    body("description").isString(),
    body("foundlocation").isString(),
    body("foundTime").isDate(),
    body("shelterName").notEmpty(),
    // body("policeSationID").notEmpty(),
    body("finderName").matches(/^[\u0621-\u064A\-\ ][^#&<>\"~;$^%{}?]{2,20}$/),
    body("finderNationID").matches(/(2|3)[0-9][1-9][0-1][1-9][0-3][1-9](01|02|03|04|11|12|13|14|15|16|17|18|19|21|22|23|24|25|26|27|28|29|31|32|33|34|35|88)\d\d\d\d\d/),
    body("finderPhone").matches(/^01[0125][0-9]{8}$/),
    body("finderEmail").isEmail()
]
```

Search report

Search report (model)

```
const mongoose = require('mongoose');

const reportSchema = mongoose.Schema({
    //missing info
    name:{type:String},
    age:{type:Number},
    gender:{type:String},
    imageURl:{type:String},
    description:{type:String},
    lostLocation:{type:String},
    lostTime:{type:Date},
    //reporter info
    reporterName:{type:String},
    reporterNationID:{type:String},
    reporterPhone:{type:String},
    reporterEmail:{type:String},
    status:{type:String , default:"hold"},
    // policeStation info
    policeStationID:{type:mongoose.Types.ObjectId , ref:'user'},
    time:{type:String}

})


const reportModel = mongoose.model("report", reportSchema);
module.exports = reportModel;
```

Search report (controller)

```
const homelessModel = require("../../model/homeless.model");
const { validationResult } = require('express-validator');
const bcrypt = require('bcrypt')
const fs = require('fs');
const request = require("request-promise");
const reportModel = require("../../model/report.model");
Complexity is 16 You must be kidding
module.exports = async (req, res) => {
    const file =req.file;
    // console.log(file);
    let imageURl;
    if (!file) {
        res.json({ message: "in-valid image" })
    } else {
        imageURl = req.file.path;
    }
    const { name, startAge, endAge, gender } = req.body;
    try {
        let matchedResult = [];
        const errorvalidationResult = validationResult(req);
        if (errorvalidationResult.isEmpty()) {
            // look  for match in Report table
            const allUsers = await reportModel.find({ gender, age: { $gte: startAge, $lte: endAge } }).populate(["policeStationID"])
            if (allUsers) {
                for (let i = 0; i < allUsers.length; i++) {
                    const options = {
                        method: 'POST',
                        origin:'*',
                        url: 'https://face-verification2.p.rapidapi.com/FaceVerification',
                        headers: {
                            'content-type': 'multipart/form-data; boundary=---011000010111000001101001',
                            'x-rapidapi-key': '5834cb2847msh2c96ebb8f6b326ap1276d5jsn4ff377f89c79',
                            'x-rapidapi-host': 'face-verification2.p.rapidapi.com',
                            useQueryString: true
                        },
                        formData: {
                            photo1: {
                                value: fs.createReadStream(imageURl),
                                options: { filename: 'mg2.jpg', contentType: 'application/octet-stream' }
                            },
                            photo2: {
                                value: fs.createReadStream(allUsers[i].imageURl),
                                options: { filename: 'mg1.jpg', contentType: 'application/octet-stream' }
                            }
                        }
                    };
                    Complexity is 6 It's time to do something...
                    await request(options ,  (error, response, body)=> {
                        if (error) throw new Error(error);
                        let jsonVariable = JSON.parse(body)
                        if (jsonVariable['data'].similarPercent >= 75) {
                            matchedResult.push({
                                foundList: allUsers[i],
                                faceSimilarPercent: jsonVariable['data'].similarPercent
                            });

                        } else if (allUsers[i].name === name) {
                            matchedResult.push({
                                foundList: allUsers[i],
                                faceSimilarPercent: jsonVariable['data'].similarPercent
                            });
                        }
                    });
                }
                    res.json({message:"search Done" ,matchedResult});

            } else {
                res.json({ message: "gender not fount woulf u like   continue to add", matchedResult })
            }

        } else {
            //validation errror
            console.log(errorvalidationResult.array());
            res.json({
                message: "errorvalidationResult",
                errorvalidationResultArray: errorvalidationResult.array(),
                oldInputs: { name, startAge, endAge, gender, imageURl }
            })

        }
    } catch (error) {
        // console.log(error);
        // res.json({ message: "catch error"  , error})
    }
```

Search report (validation)

```
const { body} = require('express-validator');

module.exports =[
    body("name").matches(/^[\u0621-\u064A ][^#&<>\"~;$^%{}?]{2,20}$/),
    body("startAge").isNumeric(),
    body("endAge").isNumeric(),
    body("gender").isString()
]
```

Communicate with relative

Communicate with relative (controller)

```
const userModel = require('../../model/user.model');
const homelessModel = require('../..//model/homeless.model');
const senemail = require("../email/senemail.contrroler");
const { validationResult } = require("express-validator");
const CryptoJS = require("crypto-js");
Complexity is 13 You must be kiddling
module.exports = async (req, res) => {
    // const { shelterName } = req.body
    const id = req.params.id; // repoert id
    let imageURl;
    if (req.file == undefined) {
        res.json({ message: "in-valid image" })
    } else {
        imageURl = req.file.path;
    }
    const { shelterName, name, age, gender, description, foundlocation, foundTime,
        finderName, finderNationID, finderPhone, finderEmail } = req.body;
    let policeSationID = req.userID;
    try {
        const errors = validationResult(req);
        if (errors.isEmpty()) {
            const parent = await reportModel.findOne({ _id: id });
            if (parent) {
                if (parent.status == "hold") {
                    const shelter = await userModel.findOne({ userName: shelterName });
                    if (shelter) {
                        let ciphertext = CryptoJS.AES.encrypt(finderNationID, 'secret key 123').toString();
                        const data = await homelessModel.insertMany({
                            name, age, gender, imageURl, description, foundlocation, foundTime,
                            shelterID: shelter._id, policeSationID, reportID: parent._id,
                            finderName, finderNationID: ciphertext, finderPhone, finderEmail,
                            status: "inCommunicate"
                        }) // add  in temprory shelter
                        //updateReport
                        await reportModel.updateOne({ _id: parent._id },
                            { status: "inCommunicate" });
                        //email
                        let message = `
                                <p>reportID: ${parent._id}</p>
                                <p> <a href='http://localhost:3000/seachHomelessByID/${data._id}'>view  u missing </a></p>
                                <P>congratolatioin we have found u  missing please go to
                                <a href='${shelter.location}'> open location in google maps</a> </P> `;
                        await senemail(parent.reporterEmail, message)
                        res.json({ message: "Done" })
                    } else {
                        res.json("invalid shelter")
                    }
                } else {
                    res.json({ message: "already in coummunicate" });
                }
            } else {
                res.json({ message: "invalid report" });
            }
        } else {
            res.json({
                message: "in-valid input",
                messageError: errors.array(),
                oldInputs: {
                    name, age, gender, description, foundlocation, foundTime, policeSationID,
                    finderName, finderNationID, finderPhone, finderEmail, shelterName
                }
            })
        }
    } catch (error) {
        res.json({ message: "catch error", error })
    }


}
```

Communicate with relative (validation)

```
const { body} = require('express-validator');
module.exports =[
    body("name").matches(/^[\u0621-\u064A\-\ ][^#&<>\"~;$^%{}?]{2,20}$/),
    body("age").isNumeric(),
    body("gender").isString(),
    body("shelterName").matches(/[A-Z][a-zA-Z][^#&<>\"~;$^%{}?]{1,20}$/),
    body("description").isString(),
    body("foundlocation").isString(),
    body("foundTime").isDate(),
    // body("shelterID").notEmpty(),
    body("finderName").matches(/^[\u0621-\u064A\-\ ][^#&<>\"~;$^%{}?]{2,20}$/),
    body("finderNationID").matches(/(2|3)[0-9][1-9][0-1][1-9][0-3][1-9](01|02|03|04|11|12|13|14|15|16|17|18|19|21|22|23|24|25|26|27|28|29|31|32|33|34|35|88)\d\d\d\d\d/),
    body("finderPhone").matches(/^01[0125][0-9]{8}$/),
    body("finderEmail").isEmail()
]
```

Close homeless

Close homeless (controller)

```
const homelessModel = require("../../model/homeless.model");


module.exports =async (req, res) => {

    const id = req.params.id
    try {
        const homeless = await homelessModel.findOne({ _id: id });
        if (homeless) {
            if (homeless.status!="closed") {

            await homelessModel.updateOne({ _id: homeless._id }, { status: 'closed' });
            res.json({ message: "homless closed successfully" });
            } else {
            res.json({ message: "homless  alrady closed " });

            }
        } else {
            res.json({ message: "homless not found" });

        }
    } catch (error) {

        res.json({ message: "catch error" });
    }
}
```

Undefined homeless

Undefined homeless (controller)

```
const homelessModel = require("../../model/homeless.model");


module.exports =async (req, res) => {

    const id = req.params.id
    try {
        const homeless = await homelessModel.findOne({ _id: id });
        if (homeless) {
            if (homeless.status!="undefined") {

            await homelessModel.updateOne({ _id: homeless._id }, { status: 'undefined' });
            res.json({ message: "homless undefined successfully" });
            } else {
            res.json({ message: "homless  alrady undefined" });

            }
        } else {
            res.json({ message: "homless not found" });

        }
    } catch (error) {

        res.json({ message: "catch error" });
    }
}
```

Search homeless by id

search homeless by id (controller)

```
const homelessModel = require("../../model/homeless.model")

module.exports = async (req, res) => {

    const id = req.params.id;
    try {
        const homless = await homelessModel.findOne({ _id: id })

        if (homless) {
            res.json({ message: "found", homless });

        } else {
            res.json({ message: "invalid id" })
        }
    } catch (error) {
        res.json({ message: "catch error", error });

    }
}
```

Get all homeless

Get all homeless (controller)

```
const homelessModel = require("../../model/homeless.model")
var CryptoJS = require("crypto-js");

module.exports = async(req,res)=>{

    try {
        const homeLessList = await homelessModel.find({}).populate(["policeSationID" ,"shelterID","reportID"]);

        for (let i = 0; i < homeLessList.length; i++) {
            let bytes=  CryptoJS.AES.decrypt(homeLessList[i].finderNationID , 'secret key 123');
            homeLessList[i].finderNationID  =bytes.toString(CryptoJS.enc.Utf8);
        }
        console.log(homeLessList);
            res.json({ homeLessList , message:"Done" })
    } catch (error) {
        res.json({message:"error catch" , error})
    }
}
```

Get all shelter

get all shelter (controller)

```
const homelessModel = require("../../model/homeless.model");

Complexity is 5 Everything is cool!
module.exports = async (req,res)=>{

    try {
        const residentList = await homelessModel.find({shelterID: req.userID}).populate(['shelterID','policeSationID']);
        if (residentList) {
            res.json({message:"Done" , residentList});
        } else {
            res.json({message:"in-valid shelter id"});

        }
    } catch (error) {
        res.json({message:"catch error"});

    }
}
```

Update homeless

Update homeless (controller)

```
const homelessModel = require("../../model/homeless.model");
const { validationResult } = require('express-validator');
const bcrypt = require('bcrypt');
const CryptoJS = require("crypto-js");

const userModel = require("../../model/user.model");
Complexity is 11 You must be kidding
module.exports = async (req, res) => {
    const id= req.params.id;
    let imageURl;

if (req.file == undefined) {
    // res.json({ message: "in-valid image" });
    const homeless = await homelessModel.findOne({ _id:id });
    imageURl = homeless.imageURl;
} else {
    imageURl = req.file.path;
}
    const { name, age, gender, description, foundlocation, foundTime, shelterName,
        finderName, finderNationID, finderPhone, finderEmail  , reportID} = req.body;
    let policeSationID = req.userID;
    try {
        const validationError = validationResult(req);
        if (validationError.isEmpty()) {
            const homeless = await homelessModel.findOne({ name });
            if (!homeless) {
                res.json({ message: "invalid homless", homeless })
            } else {
                let shelter = await userModel.findOne({ userName: shelterName })
                if (shelter) {

                    let ciphertext =  CryptoJS.AES.encrypt(finderNationID, 'secret key 123').toString();

                        await homelessModel.updateOne({_id:id},{
                            name, age, gender, imageURl, description,
                            foundlocation, foundTime, shelterID: shelter._id, policeSationID,
                            finderName, finderNationID: ciphertext, finderPhone, finderEmail ,reportID
                        });
                        res.json({ message: "updated successfully" });

                } else {
                    res.json({
                        message: "in-valid shelter id", oldInputs: {
                            name, age, gender, imageURl, description,
                            foundlocation, foundTime, shelterName, policeSationID,
                            finderName, finderNationID, finderPhone, finderEmail,reportID
                        }
                    })
                }


            }
        } else {
            res.json({
                message: "please enter valid data err",
                errorMessage: validationError.array(),
                oldInputs: {
                    name, age, gender, imageURl, description,
                    foundlocation, foundTime, shelterName, policeSationID,
                    finderName, finderNationID, finderPhone, finderEmail
                }
            });
        }

    } catch (error) {
        res.json({ message: "catch err", error });
    }

}
```

Search homeless

Search homeless (controller)

```
const homelessModel = require("../../model/homeless.model");
const { validationResult } = require('express-validator');
const fs = require("fs")
const request = require("request-promise")
Complexity is 15 You must be kidding
module.exports = async (req, res) => {
    let imageUrl;
    if (req.file == undefined) {
        res.json({ message: "in-valid image" })
    } else {
        imageUrl = req.file.path;
    }
    const { name, startAge, endAge, gender } = req.body;

    try {
        const validationResultError = validationResult(req);
        if (validationResultError.isEmpty()) {
            let matchedResult = [];
            let homelessList = await homelessModel.find({ gender, age: { $gte: startAge, $lte: endAge } }).populate(['shelterID','policeSationID']);

            if (homelessList) {
                console.log(homelessList.length);
                for (let i = 0; i < homelessList.length; i++) {
                    console.log(homelessList[i].imageUrl);
                    console.log(typeof (homelessList[i].imageUrl));
                    const options = {
                        method: 'POST',
                        url: 'https://face-verification2.p.rapidapi.com/FaceVerification',
                        headers: {
                            'content-type': 'multipart/form-data; boundary=---011000010111000001101001',
                            'x-rapidapi-key': '5834cb2847msh2c96ebb8f6b326ap1276d5jsn4ff377f09c79',
                            'x-rapidapi-host': 'face-verification2.p.rapidapi.com',
                            useQueryString: true
                        },
                        formData: {
                            photo1: {···
                            },
                            photo2: {
                                value: fs.createReadStream(homelessList[i].imageUrl),
                                options: { filename: 'mg1.jpg', contentType: 'application/octet-stream' }
                            }
                        }
                    };
                    Complexity is 5 Everything is cool!
                    await request(options, async (error, response, body) => {
                        if (error) {
                            res.json(error)
                        }
                        let jsonVariable = await JSON.parse(body);
                        if (jsonVariable['data'].similarPercent >= 75) {
                            matchedResult.push({ foundList: homelessList[i], faceSimilarPercent: jsonVariable['data'].similarPercent });
                        } else if (name == homelessList[i].name) {
                            matchedResult.push({ foundList: homelessList[i], faceSimilarPercent: jsonVariable['data'].similarPercent });
                        }
                    });
                }
                res.json({message:"search Done" , matchedResult});
            } else {
                res.json({ message: "gender not fount woulf u like   continue to add", matchedResult })
            }
        } else {
            res.json({
                message: "pleas enter valid data ",
                messageError: validationResultError.array(),
                oldInputs: { name, startAge, endAge, gender }
            })
        }
    } catch (error) {
        res.json({ message: "catch err", error })
```

Search homeless (validation)

```
const { body} = require('express-validator');

module.exports =[
    body("name").matches(/^[\u0621-\u064A ][^#&<>\"~;$^%{}?]{2,20}$/),
    body("startAge").isNumeric(),
    body("endAge").isNumeric(),
    body("gender").isString()
]
```

Add report

Add report (controller)

```
const { validationResult } = require('express-validator');
const reportModel = require('../../model/report.model');
const sendMail = require('../email/senemail.contrroler');
const bcrypt = require('bcrypt');
const CryptoJS = require("crypto-js");
const userModel = require('../../model/user.model');
Complexity is 3 Everything is cool!
async function addReport(req, res, data, myemail, meassage) { ...
}
Complexity is 16 You must be kidding
module.exports = async (req, res) => {
    let imageURl;
    if (req.file == undefined) {
        res.json({ message: "in-valid image" })
    } else {
        imageURl = req.file.path;
    }
    const { name, age, description, gender, lostLocation, lostTime, reporterName,
        reporterNationID, reporterPhone, reporterEmail, policeStationName } = req.body;
    try {
        const errors = validationResult(req);
        if (errors.isEmpty()) {
            let message = `<p> u report submited successfully   and would be activated after 24h </p>`;
            const report = await reportModel.findOne({ name });
            const policeStation = await userModel.findOne({ userName: policeStationName })
            if (policeStation) {
                if (report) {
                    if (report.status === 'active' || report.status === 'inCommunicate' | report.status === 'hold') {
                        res.json({ message: "u report already submited once", report })
                    } else if (report.status === 'closed') {

                        let ciphertext = CryptoJS.AES.encrypt(reporterNationID, 'secret key 123').toString();
                        let myparadox = new Date();
                        myparadox.setTime(myparadox.getTime() + (2 * 60 * 60 * 1000))
                        let paradox = myparadox.toISOString();
                        await addReport(req, res, {
                            name, age, description, gender, imageURl, lostLocation, lostTime, reporterName,
                            reporterNationID: ciphertext, reporterPhone, reporterEmail, policeStationID: policeStation._id, time: paradox
                        }, reporterEmail, message);
                        res.json({ message: "Done" });

                    }
                } else {

                    let ciphertext = CryptoJS.AES.encrypt(reporterNationID, 'secret key 123').toString();
                    let myparadox = new Date();
                    myparadox.setTime(myparadox.getTime() + (2 * 60 * 60 * 1000))
                    let paradox = myparadox.toISOString();
                    await addReport(req, res, {
                        name, age, description, gender, imageURl, lostLocation, lostTime, reporterName,
                        reporterNationID: ciphertext, reporterPhone, reporterEmail, policeStationID: policeStation._id, time: paradox
                    }, reporterEmail, message);
                    res.json({ message: "Done" });

                }
            } else {
                res.json({
                    message: "in-valid policeStation",
                    oldInputs: {
                        imageURl, name, age, description, gender, lostLocation, lostTime, reporterName,
                        reporterNationID, reporterPhone, reporterEmail, policeStationName
                    }
                })
            }
```

```
            } else {
                res.json({
                    message: "in-valid policeStation",
                    oldInputs: {
                        imageURl, name, age, description, gender, lostLocation, lostTime, reporterName,
                        reporterNationID, reporterPhone, reporterEmail, policeStationName
                    }
                })
            }


        } else {
            res.json({
                message: 'invalidInput',
                errorMessage: errors.array(),
                oldInputs: {
                    name, age, imageURl, description, gender, lostLocation, lostTime, reporterName,
                    reporterNationID, reporterPhone, reporterEmail, policeStationName
                }
            })

        }

    } catch (error) {
        res.json({ message: 'catch error', error })
    }
}
```

Add report (validation)

```
const { body} = require('express-validator');
module.exports =[

    body("name").matches(/^[\u0600-\u06ff]+[\u0600-\u06ff-_\.][^#&<>\"~;$^%{}?]{1,20}$/),
    body("age").isNumeric(),
    body("gender").isString(),
    body("description").isString(),
    body("lostLocation").isString(),
    body("lostTime").isDate(),
    body("reporterName").matches(/^[\u0600-\u06ff]+[\u0600-\u06ff-_\.][^#&<>\"~;$^%{}?]{1,20}$/),
    body("reporterNationID").matches(/(2|3)[0-9][1-9][0-1][1-9][0-3][1-9](01|02|03|04|11|12|13|14|15|16|17|18|19|21|22|23|24|25|26|27|28|29|31|32|33|34|35|88)\d\d\d\d\d/),
    body("reporterPhone").matches(/^01[0125][0-9]{8}$/),
    body("reporterEmail").isEmail(),
    body("policeStationName").isString()
]
```

Close report

Close report (controller)

```
const homelessModel = require("../../model/homeless.model");
const reportModel = require("../../model/report.model")
Complexity is 5 Everything is cool!
module.exports =  async (req, res) => {

    const id = req.params.id
    try {
        const report = await reportModel.findOne({ _id: id });
        if (report) {
            await reportModel.updateOne({ _id: report._id }, { status: 'closed' });
            await homelessModel.updateOne({ reportID: report._id }, { status: 'closed' });
            res.json({ message: "report and homless closed successfully" });
        } else {
            res.json({ message: "report not found" });

        }
    } catch (error) {

        res.json({ message: "catch error" });
    }
}
```

View report

View report (controller)

```
const reportModel = require("../../model/report.model")
const CryptoJS = require("crypto-js");
Complexity is 4 Everything is cool!
module.exports = async (req,res)=>{
    try {
        const reportList = await reportModel.find({}).populate('policeStationID');
        for (let i = 0; i < reportList.length; i++) {
            let bytes=  CryptoJS.AES.decrypt(reportList[i].reporterNationID , 'secret key 123');
            reportList[i].reporterNationID  =bytes.toString(CryptoJS.enc.Utf8);
        }
        res.json({message:"Done",reportList})

    } catch (error) {
        res.json({message:"catch error",error})

    }
}
```

Close report

Close report (controller)

```
ontroller > report > JS closeReport.controller.js > ...
const reportModel = require("../../model/report.model");
Complexity is 5 Everything is cool!
module.exports = async (req,res)=>{
    try {
        const id = req.params.id;

        const report = await reportModel.findOne({_id:id});
        if (report) {
            await reportModel.updateOne({ _id: report._id }, { status: 'closed' });
            res.json({message:"Report Closed successfully"});
        }else{
            res.json({message:"invalid report id "});

        }

    } catch (error) {
        res.json({message:"Report Closed catch"});

    }
}
```

Activate report

Activate report (controller)

```
const reportModel = require("../../model/report.model");
Complexity is 5 Everything is cool!
module.exports = async (req,res)=>{
    try {
        const id = req.params.id;

        const report = await reportModel.findOne({_id:id});
        if (report) {
            await reportModel.updateOne({ _id: report._id }, { status: 'active' });
            res.json({message:"Report activeted successfully"});
        }else{
            res.json({message:"invalid report id "});

        }

    } catch (error) {
        res.json({message:"Report activeted catch"});

    }
}
```

Edit report

Edit report (controller)

```
const reportModel = require("../../model/report.model");
Complexity is 5 Everything is cool!
module.exports = async (req,res)=>{
    try {
        const id = req.params.id;

        const report = await reportModel.findOne({_id:id});
        if (report) {
            await reportModel.updateOne({ _id: report._id }, { status: 'active' });
            res.json({message:"Report activeted successfully"});
        }else{
            res.json({message:"invalid report id "});

        }

    } catch (error) {
        res.json({message:"Report activeted catch"});

    }
}
```

Display police station signup request

Display police station signup request (controller)

```
const userModel = require("../../model/user.model")

module.exports = async (req, res) => {
    try {
        const requestList = await userModel.find({ aprove: false, confirmEmail: true  , role:"policeStation"});
        res.json({message:'done',requestList});
    } catch (error) {
        res.json({ message: "can not get any users" });
    }
}
```

Display shelter signup request

Display shelter signup request (controller)

```
const userModel = require("../../model/user.model")

module.exports = async (req, res) => {
    try {
        const requestList = await userModel.find({
            aprove: false,
            confirmEmail: true, role: "shelter"
        });
        res.json({message:'done',requestList});

    } catch (error) {
        res.json({ message: "can not get any users" });
    }
}
```

Approve sign up request controller

Approve sign up request controller(controller)

```
const userModel = require("../../model/user.model")


module.exports = async (req, res) => {
    const id = req.params.id;
    try {
        const user = await userModel.findOne({ _id: id });
        if (user) {
            if (user.aprove == true) {
                res.json({ message: 'already  approved user' });

            } else {
                await userModel.updateOne({ _id: user._id }, { aprove: true });
                res.json({ message: 'aproved user' });
            }


        } else {
            res.json({ message: 'invalid  user id' });

        }

    } catch (error) {
        res.json({ message: "fail to  take action pleas try again" });
    }
}
```

Delete policestation or shelter

Delete policestation or shelter (controller)

```
const userModel = require("../../model/user.model");

module.exports = async(req,res)=>{
    const id = req.params.id;
    try {
        const user = await userModel.findOne({_id:id});
        if (user) {
            await userModel.deleteOne({_id:user._id});
            res.json({message:"user deleted successfully"})

        } else {
            res.json({meaasge:"user not found"})
        }
    } catch (error) {
        res.json({meaasge:"catch error"})

    }
}
```

Change privileges

Change privileges (controller)

```
const userModel = require("../../model/user.model")
Complexity is 3 Everything is cool!
module.exports = async(req,res)=>{
    try {
        const shelterList = await userModel.find({role:"shelter" , aprove:true});
       res.json({message:"done" , shelterList})

    } catch (error) {
       res.json({message:"catch error display shelterList"})
    }
}
```

Display shelter

Display shelter (controller)

```
const userModel = require("../../model/user.model")
Complexity is 3 Everything is cool!
module.exports = async(req,res)=>{
    try {
        const shelterList = await userModel.find({role:"shelter" , aprove:true});
       res.json({message:"done" , shelterList})

    } catch (error) {
       res.json({message:"catch error display shelterList"})
    }
}
```

Display police station

Display police station(controller)

```
const userModel = require("../../model/user.model")
Complexity is 3 Everything is cool!
module.exports = async(req,res)=>{
    try {
        const policeStationList = await userModel.find({role:"policeStation"  , aprove:true});
       res.json({message:"done" , policeStationList})

    } catch (error) {
       res.json({message:"catch error display policeStationList"})
    }
}
```

Schedule tasks

```
/*============================== Start  schedule part ================================== */
var moment = require('moment');
const homelessModel = require('../model/homeless.model');
const sendEmail = require('../controller/email/senemail.contrroler');
Complexity is 4 Everything is cool!
async function chagngeRepot() { ...
}


Complexity is 11 You must be kidding
async function faceCompare() {
    console.log("k");
    let homelessList = await homelessModel.find({ status: "undefined" }).populate('shelterID');
    const reportList = await reportModel.find({ status: "active" });

    for (let i = 0; i < homelessList.length; i++) {

        for (let j = 0; j < reportList.length; j++) {
            if ((reportList[j].age + 5 >= homelessList[i].age || reportList[j].age - 5 <= homelessList[i].age) &&
                reportList[j].gender == homelessList[i].gender) {
                const options = {...
                };
                Complexity is 5 Everything is cool!
                await request(options, async (error, response, body) => {
                    if (error) throw new Error(error);
                    let jsonVariable = JSON.parse(body)
                    if (jsonVariable['data'].similarPercent >= 75 || allUsers[i].name === homelessList[i].name) {

                        //commmunicate attach
                        console.log("matched" + homelessList[i]);

                        let message = `
                                <p>reportID: ${reportList[j]._id}</p>
                                <p> <a href='http://localhost:3000/seachHomelessByID/${homelessList[i]._id}'>view  your missing </a></p>
                                <P>congratolatioin we have found u  missing please go to
                                <a href='${homelessList[i].shelterID.location}'> open location in google maps</a> </P> `;
                        await homelessModel.insertMany({ reportID: reportList[j]._id, status: "inCommunicate" });
                        await reportModel.updateOne({ _id: reportList[j]._id },
                            { status: "inCommunicate" });
                        await sendEmail(reportList[j].reporterEmail, message)



                    }
                });
            }
        }


    }
}
Complexity is 3 Everything is cool!
schedule.scheduleJob(`0  * * * *`, async () => {
    try {
        console.log("run");
        await chagngeRepot();
        await faceCompare();
    } catch (error) {
        // console.log('');`
    }

});
```

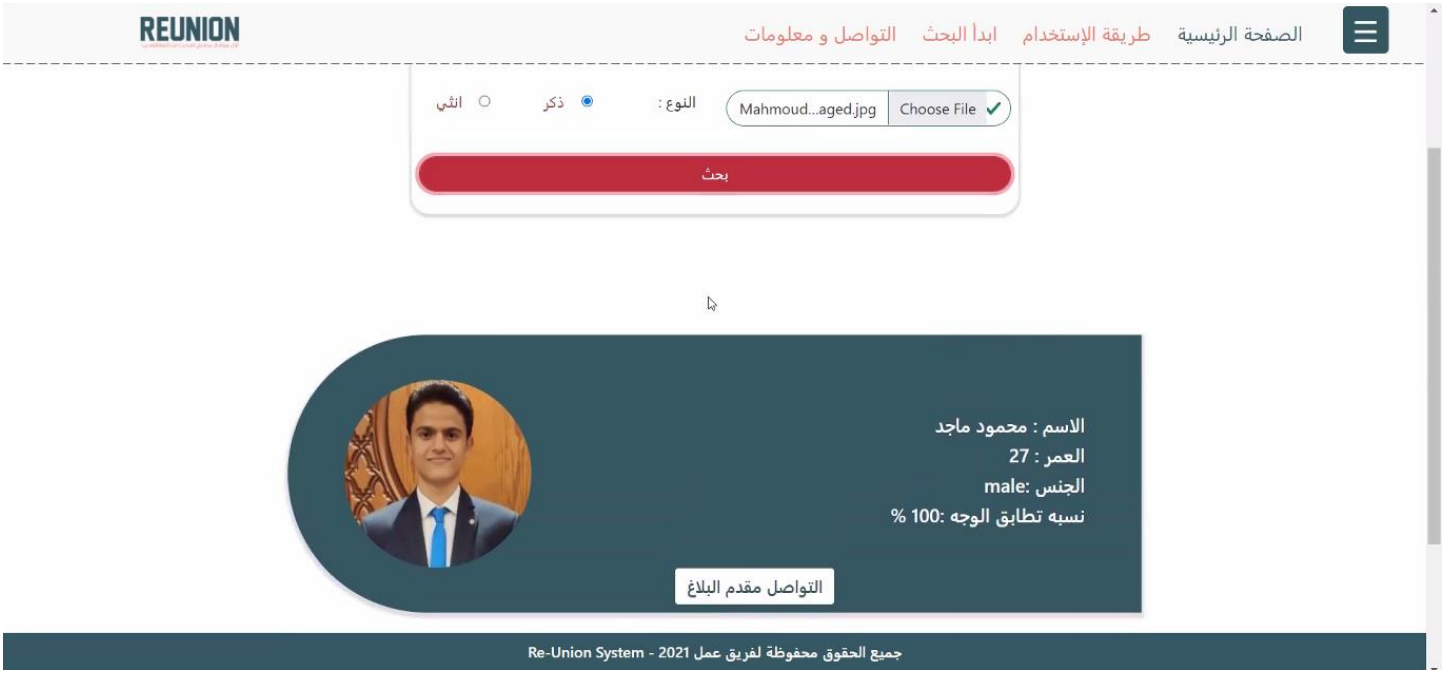
Authentication and authorization

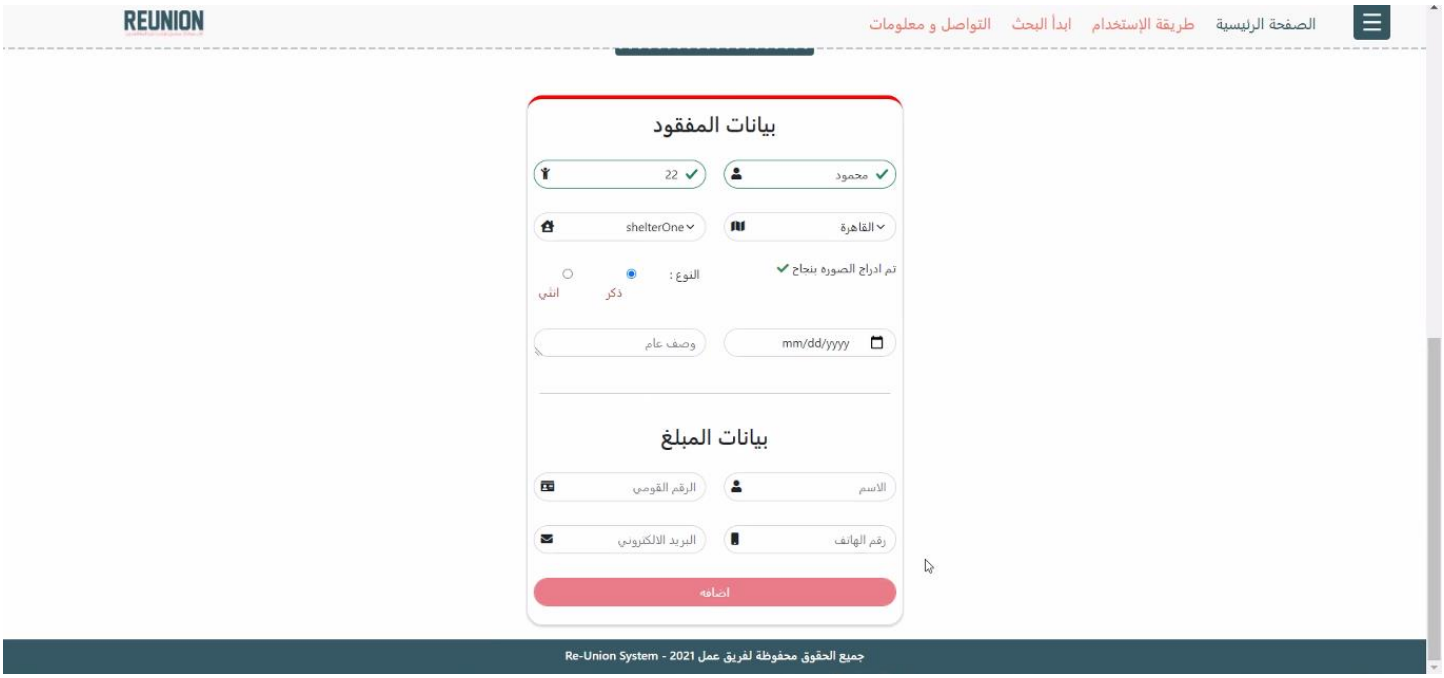
```
var jwt = require('jsonwebtoken');

function authentication(req, res, next) {

    const token = req.header("token");
    console.log(token);
    try {
        // console.log(token);
        if (token != undefined && token != null && token) {

            jwt.verify(token, 'shhhhh', async function (err, decoded) {
                if (err) {
                    res.json({ message: "fail token" })
                } else {

                    if (decoded.isLoggedIn) {
                        req.userID = decoded.userID;
                        req.userName = decoded.userName;
                        req.userRole =decoded.userRole;
                        next();

                    } else {
                        res.json({ message: "fail to login" })
                    }
                }
            });

        } else {
            res.json({ message: "invalidToken" })
        }

    } catch (error) {
        console.log("catch Error");
        res.json({ message: "auth catch Error", error })

    }
}

function authRole(role) {
try {

    return (req, res, next) => {
        if (role.includes(req.userRole)) {
            next()
        }else{

            res.status(401)
            return res.send('Not allowed')
        }

    }
} catch (error) {
    res.status(403)
    return error;
}
}

module.exports = {
    authentication,
    authRole
}
```

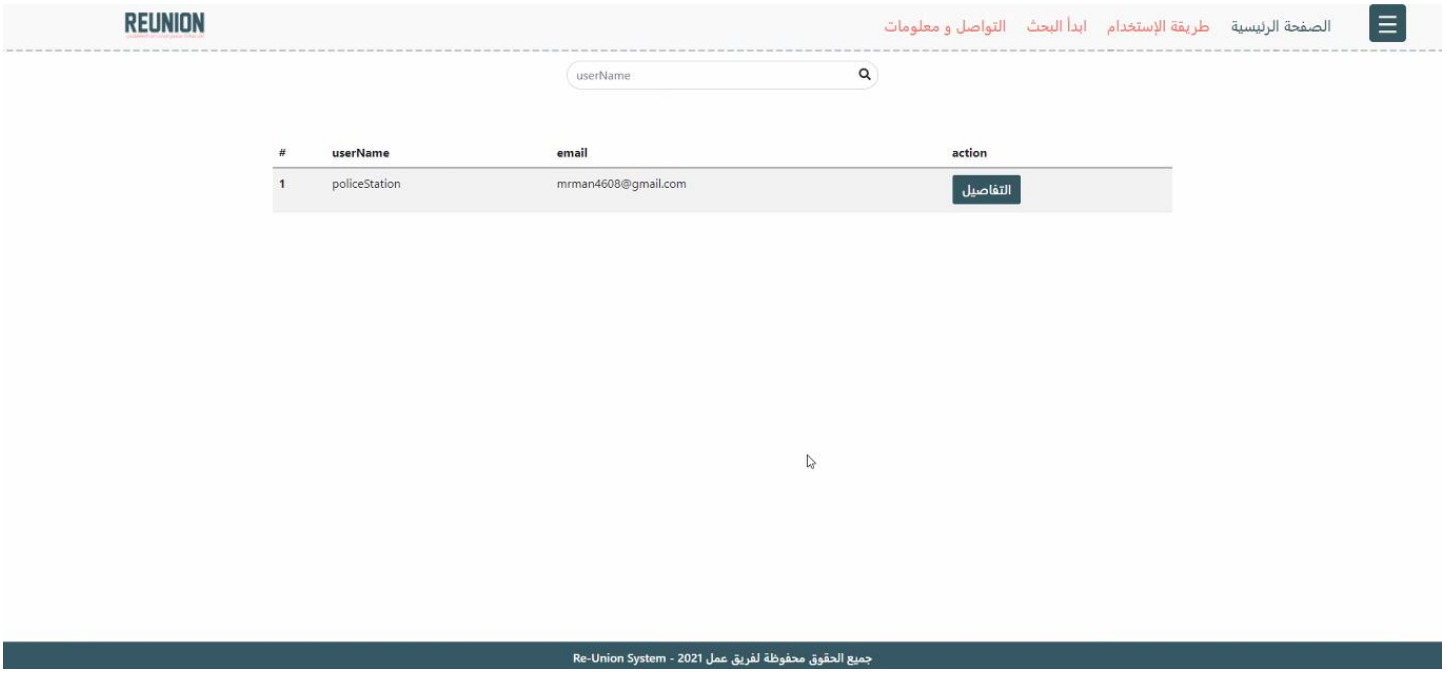
4.2 Screens

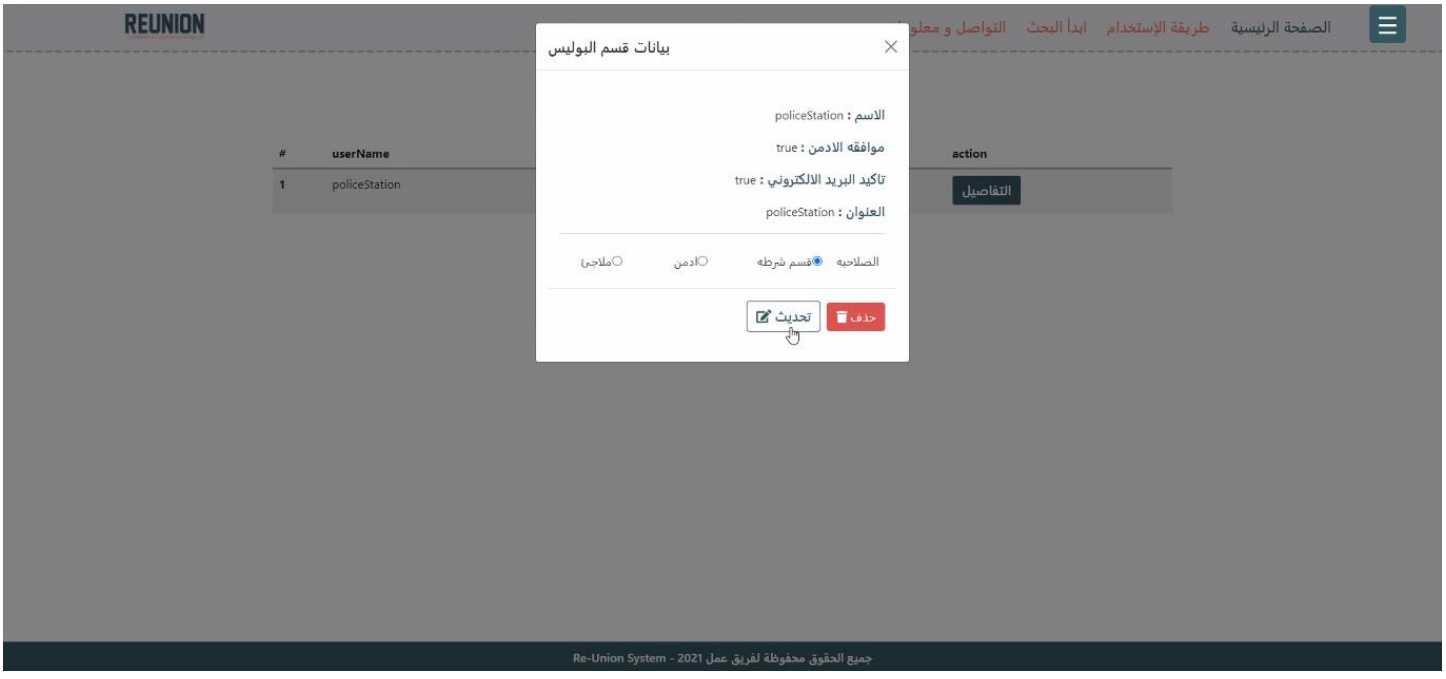

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
REUNION
أول موقع مصري للبحث عن المفقودين
تابعنا علي الفيسبوك
تنزيل تطبيقنا
إبدأ البحث
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021

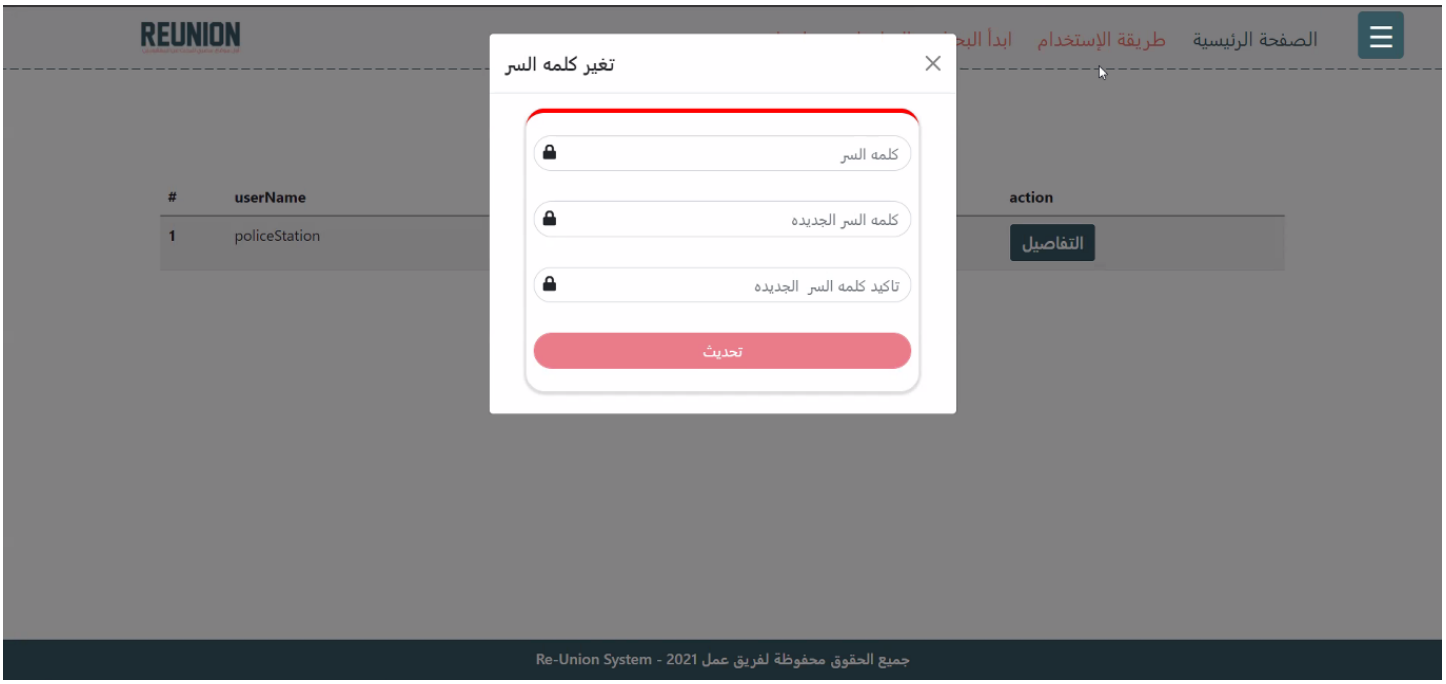

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
بيانات المفقود
الاسم
العمر
النوع :
ذكر
انثي
No file chosen
Choose File
بحث
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021


REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
النوع :
ذكر
انثي
Mahmoud...aged.jpg
Choose File
بحث
الاسم : محمود ماجد
العمر : 27
الجنس :male
نسبه تطابق الوجه :100 %
التواصل مقدم البلاغ
جميع الحقوق محفوظة لفريق عمل 2021 - Re-Union System

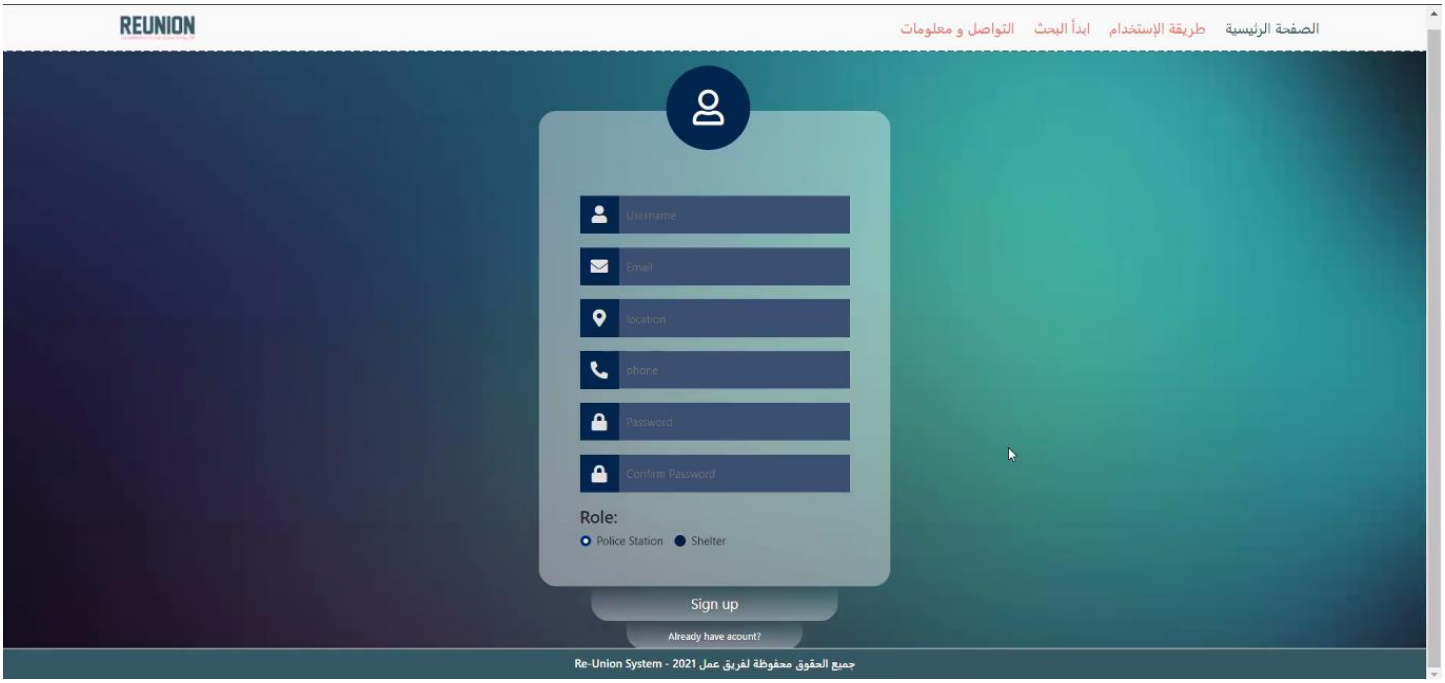

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
بيانات المفقود
محمود
22
القاهرة
shelterOne
تم ادراج الصوره بنجاح
النوع :
ذكر
انثي
mm/dd/yyyy
وصف عام
بيانات المبلغ
الاسم
الرقم القومي
رقم الهاتف
البريد الالكتروني
اضافه
جميع الحقوق محفوظة لفريق عمل 2021 - Re-Union System


REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
userName
userName email action
1 policeStation mrman4608@gmail.com التفاصيل
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021

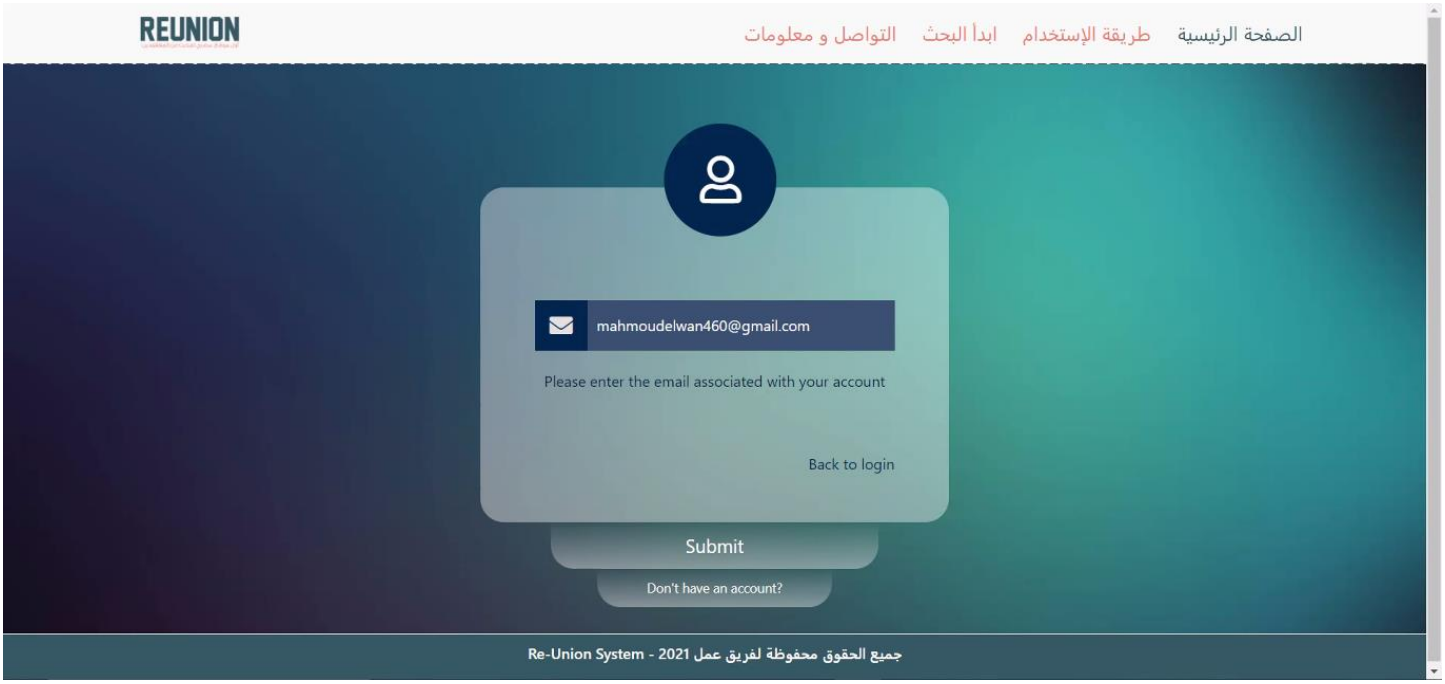

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
بيانات قسم البوليس
الاسم : policeStation
موافقه الادمن : true
تاكيد البريد الالكتروني : true
العنوان : policeStation
الصلاحيه قسم شرطه ادمن ملاجى
تحديث
حذف
userName action
1 policeStation التفاصيل
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021

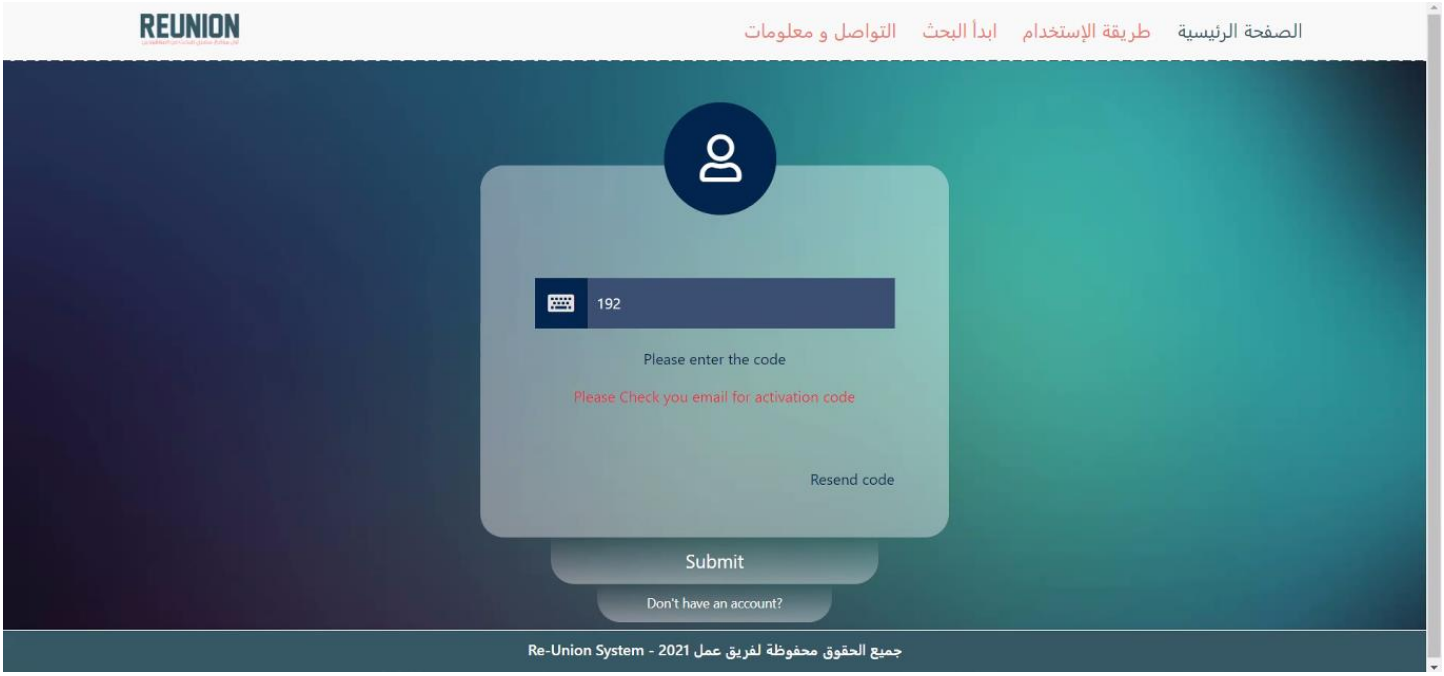

REUNION
الصفحة الرئيسية
طريقة الإستخدام
تغير كلمه السر
كلمه السر
كلمه السر الجديده
تاكيد كلمه السر الجديده
تحديث
userName
1 policeStation
action
التفاصيل
Re-Union System - 2021 جميع الحقوق محفوظة لفريق عمل

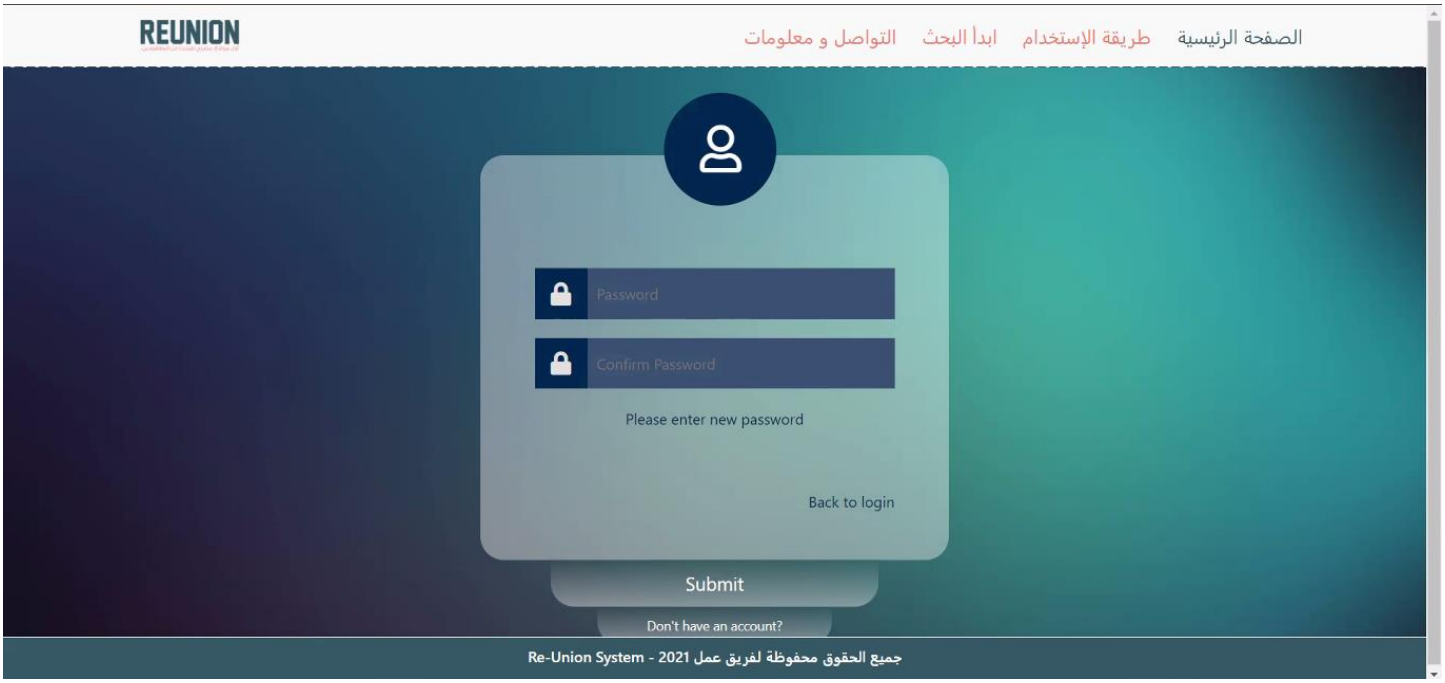

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
الأسئلة الأكثر شيوعًا
يوجد عدد من الإجابات على الأسئلة الأكثر شيوعًا. يمكنك العثور على إجابة سؤالك بداخلها ، أو يمكنك إرسال سؤال
من يمكنه استخدام خدمتنا؟
يجب أن يكون عمرك 18 عامًا على الأقل لاستخدام الموقع. يرجى التأكد من إخبارنا باسمك القانوني ورقمك القومي . في حالة إدخال معلومات خاطئة ، لن يتم النظر في البلاغ المقدم وقد تعرض نفسك لمشاكل قانونية
هل الخدمة خاصة / آمنة؟
كيف سيتم إخطاري عند العثور على نتيجه للبلاغ الذي قدمته؟
ما هو Runion System ؟
ما هو محتوى خدمات المقدمه؟
طرح سؤال؟
Name*
email*
Title*
Re-Union System - 2021 جميع الحقوق محفوظة لفريق عمل


REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
Username
Email
location
phone
Password
Confirm Password
Role:
Police Station
Shelter
Sign up
Already have acount?
جميع الحقوق محفوظة لفريق عمل 2021 - Re-Union System


REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
Email
Password
Forgort password?
Login
Don't have an account?
جميع الحقوق محفوظة لفريق عمل 2021 - Re-Union System

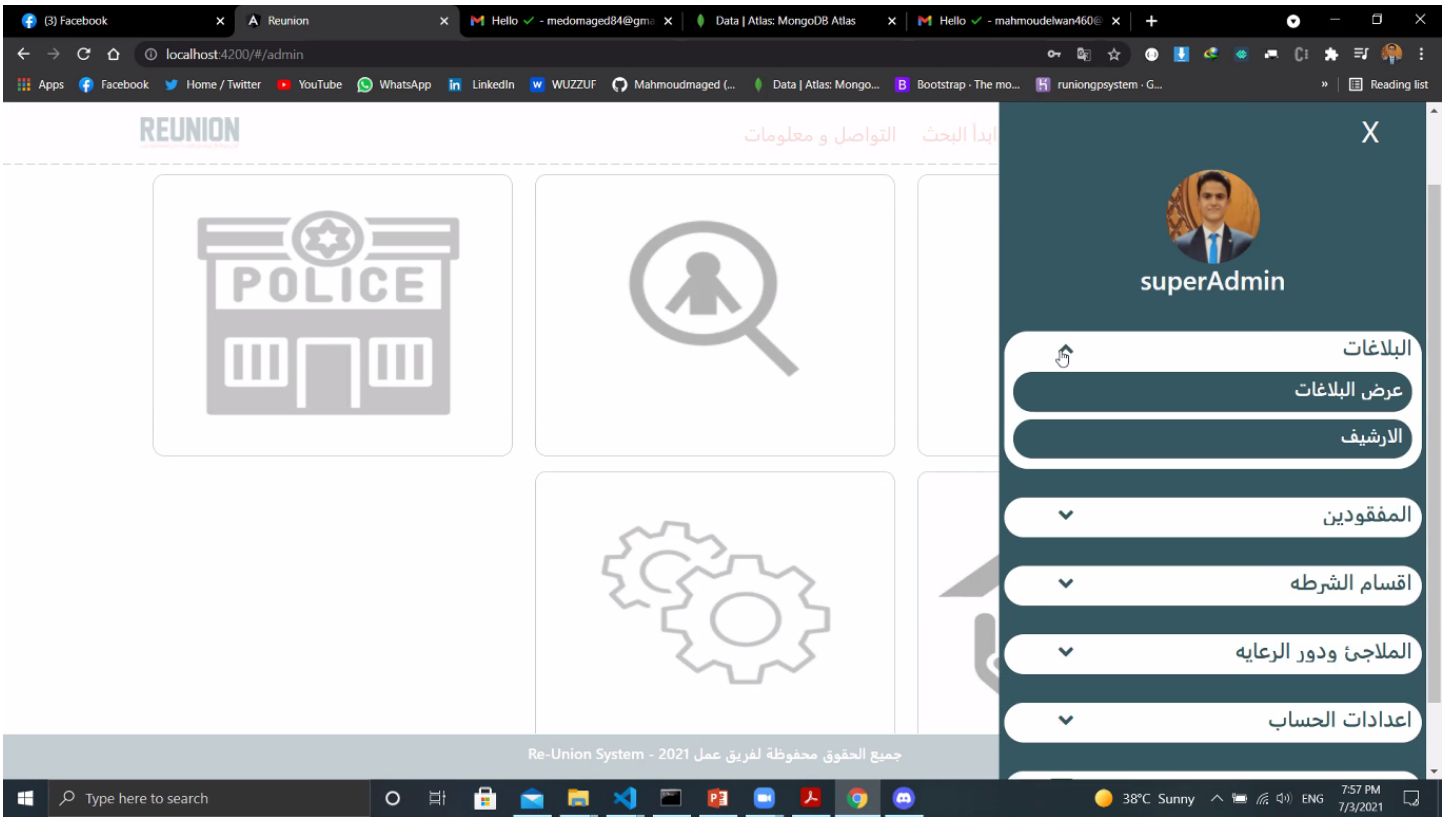

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
mahmoudelwan460@gmail.com
Please enter the email associated with your account
Back to login
Submit
Don't have an account?
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021

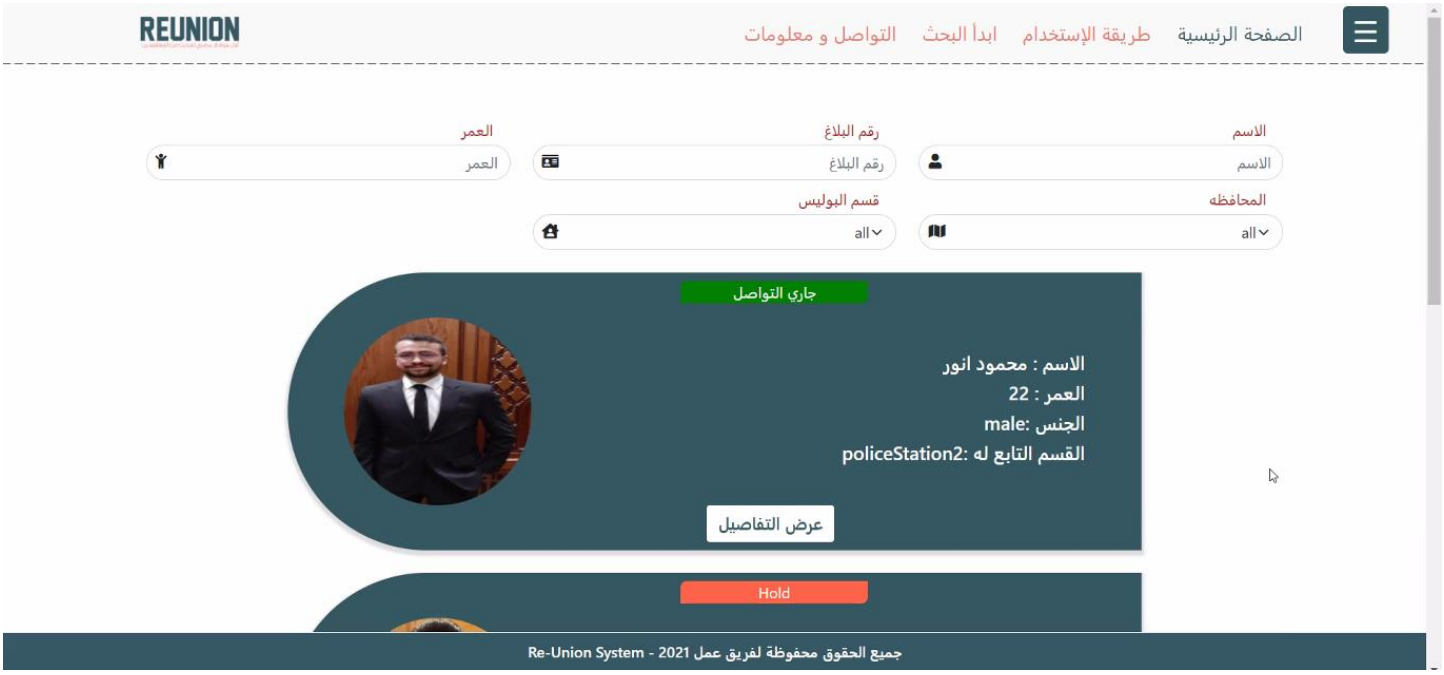

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
192
Please enter the code
Please Check you email for activation code
Resend code
Submit
Don't have an account?
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021

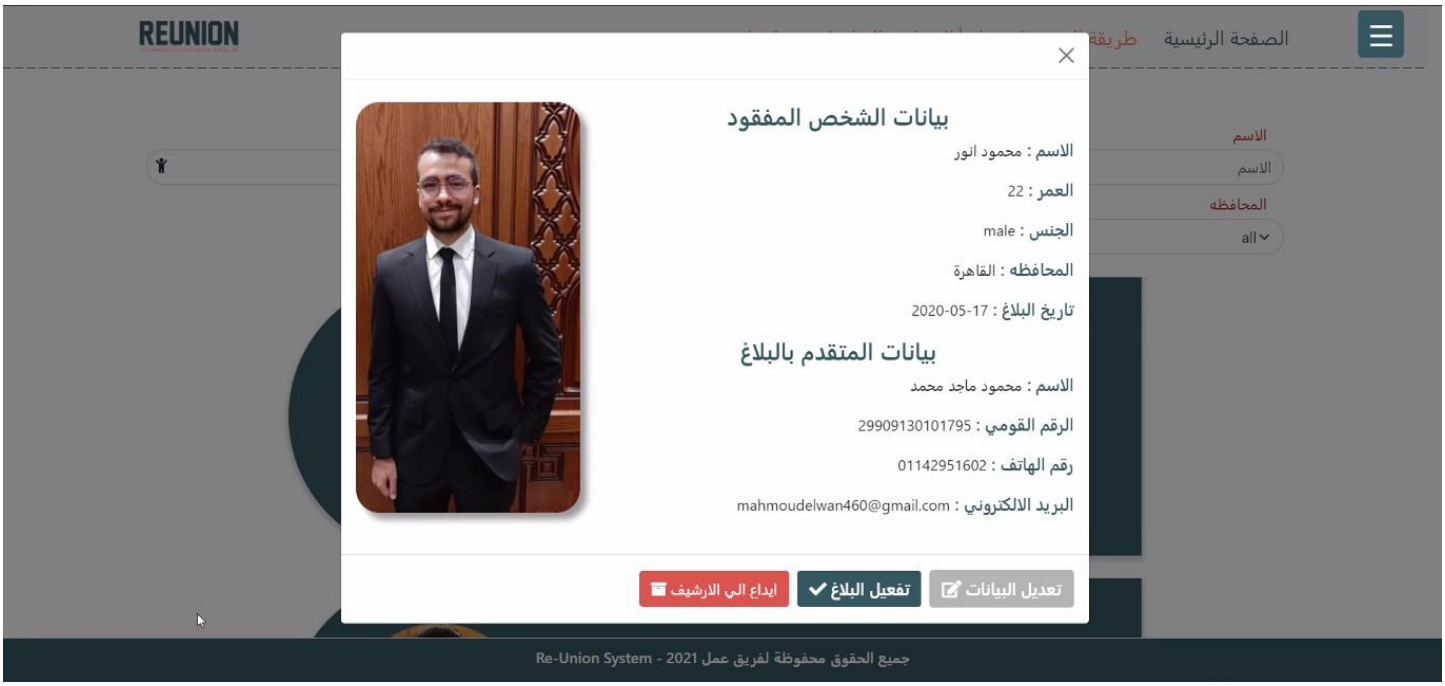

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
Password
Confirm Password
Please enter new password
Back to login
Submit
Don't have an account?
جميع الحقوق محفوظة لفريق عمل 2021 - Re-Union System

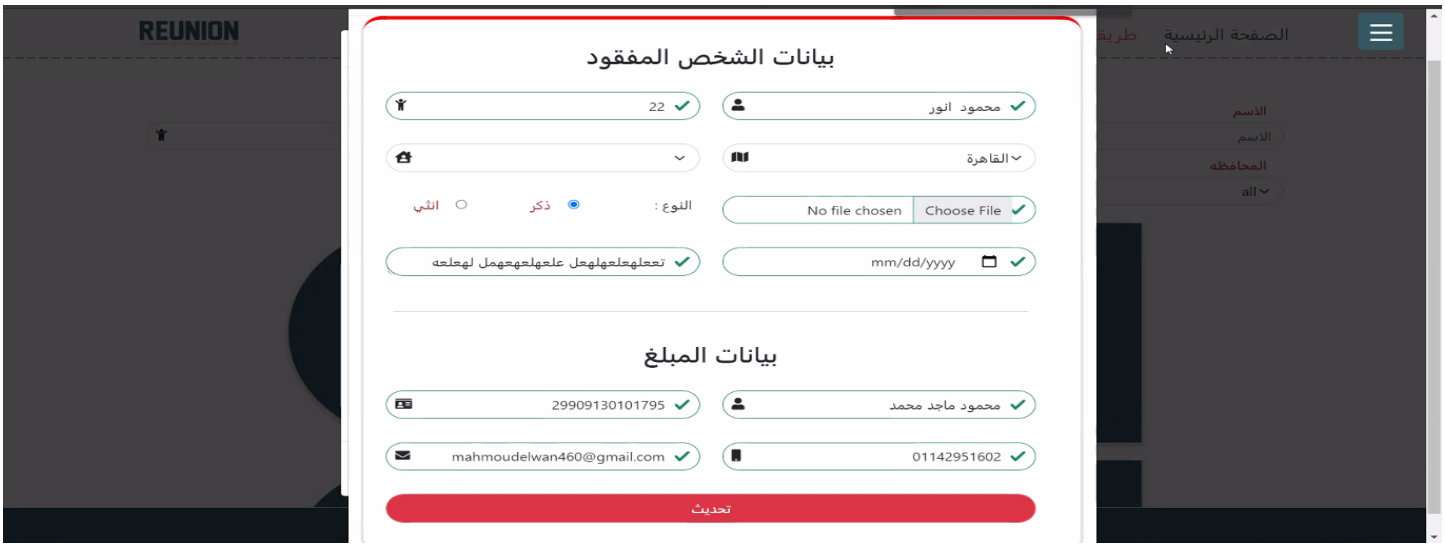

REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
POLICE
جميع الحقوق محفوظة لفريق عمل 2021 - Re-Union System


REUNION
التواصل و معلومات
ابدأ البحث
POLICE
X
superAdmin
البلاغات
المفقودين
اقسام الشرطه
الملاجئ ودور الرعايه
اعدادات الحساب
تسجيل الخروج
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021


(3) Facebook
Reunion
localhost:4200/#/admin
REUNION
التواصل و معلومات
ابدأ البحث
POLICE
X
superAdmin
البلاغات
عرض البلاغات
الارشيف
المفقودين
اقسام الشرطه
الملاجئ ودور الرعايه
اعدادات الحساب
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021
Type here to search
38°C Sunny
ENG
7:57 PM
7/3/2021


REUNION
الصفحة الرئيسية
طريقة الإستخدام
ابدأ البحث
التواصل و معلومات
الاسم
رقم البلاغ
العمر
المحافظه
قسم البوليس
all
جاري التواصل
الاسم : محمود انور
العمر : 22
الجنس :male
القسم التابع له :policeStation2
عرض التفاصيل
Hold
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021


REUNION
الصفحة الرئيسية
بيانات الشخص المفقود
الاسم : محمود انور
العمر : 22
الجنس : male
المحافظه : القاهرة
تاريخ البلاغ : 2020-05-17
بيانات المتقدم بالبلاغ
الاسم : محمود ماجد محمد
الرقم القومي : 29909130101795
رقم الهاتف : 01142951602
البريد الالكتروني : mahmoudelwan460@gmail.com
ايداع الي الارشيف
تفعيل البلاغ
تعديل البيانات
جميع الحقوق محفوظة لفريق عمل Re-Union System - 2021



4.3 Software tools

4.5.1 Node.js

- Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser

4.5.2 MongoDB

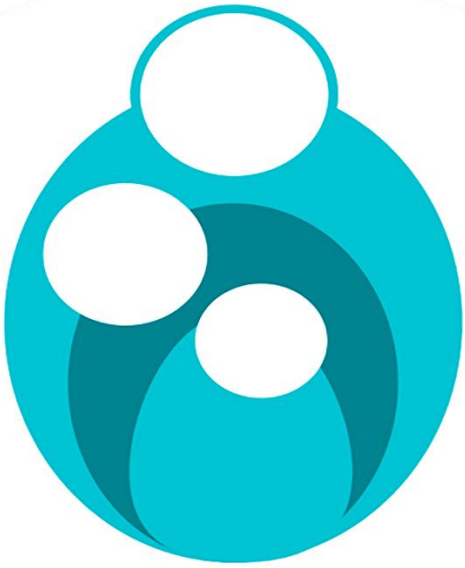
- MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License.

4.5.3 Visual Studio code IDE

- It is a free source-code editor made by Microsoft its Features include support for debugging, syntax highlighting, intelligent code completion, code refactoring and embedded Git, and it supports Node.js

4.5.4 Angular

- Angular is a Typescript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built Angular



5- Testing

In this chapter, we will present types of testing used after implementation and test cases document bugs of the system and solve it.

5.1 Functional testing:

5.1.1 Unit test

Testing of individual items as part of the implementation phase, not with integrated items and the system as a whole

5.1.2 Integration test

Testing in which software items are combined and tested as a group.

5.1.3 System testing

Testing a system behavior after it is fully integrated, when development is finished, hence, the system can be tested as complete entity.

5.1.4 Regression testing

Test older functionality to check after integrating new functionality.

5.2 Non-Functional Testing :

5.2.1 Performance Testing

Accomplished a designated function regarding processing time and through put rate.

5.2.2 Security Testing

Testing how system do with unauthorized access.

5.2.3 Usability Testing

Testing how the GUI is user-friendly and easy to use

5.3 Test cases :

5.3.1 Register test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
01	Check user register with valid data	1- Open login page 2. Click on don't have account 3.Fill required data	Username="policeStation2" Email=" mrman4608@gmail.com" Phone="01142951602" Location= "https://www.google.com/maps/@30.1446634,31.3961021,16z" Role="policeStation" Password="Mahmoud@123" Confirm password= "Mahmoud@123"	User will be redirected to login page	As expected	Pass
02	Check user register with invalid username	1- Open login page 2-Click on don't have account 3.Fill required data	Username="policestation2" Email=" mrman4608@gmail.com" Phone="01142951602" Location= "https://www.google.com/maps/@30.1446634,31.3961021,16z" Role="policeStation" Password="Mahmoud@123" Confirm password= "Mahmoud@123"	User alerted about invalid username	As expected	Pass
03	Check user register with invalid email	1- Open login page 2-Click on don't have account 3.Fill required data	Username="policeStation2" Email= "mrman4608@gmail" Phone="01142951602" Location= "https://www.google.com/maps/@30.1446634,31.3961021,16z" Role="policeStation" Password="Mahmoud@123" Confirm password= "Mahmoud@123"	User alerted about invalid email	As expected	Pass

04	Check user register with invalid phone number	1- Open login page 2-Click on don't have account 3.Fill required data	Username="policeStation2" Email=" mrman4608@gmail.com" Phone="01142951jk2" Location= "https://www.google.com/maps/@30.1446634,31.3961021,16z" Role="policeStation" Password="Mahmoud@123" Confirm password= "Mahmoud@123"	User alerted about invalid number	As expected	Pass
05	Check user register with invalid password	1- Open login page 2-Click on don't have account 3.Fill required data	Username="policeStation2" Email=" mrman4608@gmail.com" Phone="01142951602" Location= "https://www.google.com/maps/@30.1446634,31.3961021,16z" Role="policeStation" Password="Mahmoud@" Confirm password= "Mahmoud@	User alerted about invalid password	As expected	Pass
06	Check user register with wrong confirm password	1- Open login page 2-Click on don't have account 3.Fill required data	Username="policeStation2" Email=" mrman4608@gmail.com" Phone="01142951602" Location= "https://www.google.com/maps/@30.1446634,31.3961021,16z" Role="policeStation" Password="Mahmoud@123" Confirm password= "Mahmoud@1"	User alerted about invalid confirm password	As expected	Pass
07	Check user register with wrong confirm location	1- Open login page 2-Click on don't have account 3.Fill required data	Username="policeStation2" Email=" mrman4608@gmail.com" Phone="01142951602" Location=	User alerted about invalid location	As expected	Pass

		"https://www.google.com" Role="policeStation" Password="Mahmoud@123" Confirm password= "Mahmoud@1"			

5.3.2 Login test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
08	Check user login with valid data and super admin accepted user's registration.	1- Open login page 2-Enter email and password	Email= "mrman4608@gmail.com" Password="Mahmoud@123"	User will be redirect to profile page	As expected	Pass
09	Check user login with valid data and super admin Still not confirm user's registration	1- Open login page 2-Enter email and password	Email= "mrman4608@gmail.com" Password="Mahmoud@123"	A message said wait for admin approval	As expected	Pass
10	Check user login with super admin denied user's registration	1- Open login page 2-Enter email and password	Email= "mrman4608@gmail.com" Password="Mahmoud@123"	A message said invalid email or password	As expected	Pass
11	Check user login with invalid email	1- Open login page 2-Enter email and password	Email= "mrman4608@gmail.com" Password="Mahmoud@123"	A message said invalid email or password	As expected	Pass

12	Check user login with invalid password	1- Open login page 2-Enter email and password	Email= "mrman4608@gmail.com" Password="Mahmoud@123"	A message said invalid email or password	As expected	Pass

5.3.3 Forget password test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
13	Check forget password function With valid email	1- Open login page 2- Click on forget password link 3-Enter email 4-Click submit	Email= "mrman4608@gmail.com"	User will be sent a mail to change password and redirect to new form page	As expected	Pass
14	Check user with valid confirmation code after sent a mail to change password	1-Enter confirmation code 2-Enter new password and confirm password 3-Click submit	Confirmation code=568 New password= "Mahmoud@123" Confirm new password= "Mahmoud@123"	User Will be redirect to login page	As expected	Pass
15	Check user with invalid confirmation code after sent a mail to change password	1-Enter confirmation code 2-Enter new password and confirm password 3-Click submit	Confirmation code=56j New password= "Mahmoud@123" Confirm new password= "Mahmoud@123"	User alerted about invalid Confirmation code	As expected	Pass

16	Check forget password function With invalid email	1- Open login page 2- Click on forget password link 3-Enter email 4-Click submit	Email= "mrman4608@gmail.com"	A message said invalid email or password	As expected	Pass

5.3.4 Update password test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
17	Check user can update password with a valid new password	1-Logged in 2-Enter profile page 3-Click on update password 4-Enter new password 5-Enter confirm password 5-Click submit	New password= "Mahmoud@183" Confirm new password= "Mahmoud@183"	A message said your password updated successfully	As expected	Pass
18	Check user can update password with an invalid new password	1-Logged in 2-Enter profile page 3-Click on update password 4-Enter new password 5-Enter confirm password 5-Click submit	New password= "Mahmoud@" Confirm new password= "Mahmoud@"	A message said invalid email or password	As expected	Pass
19	Check user can update password with wrong confirm password	1-Logged in 2-Enter profile page 3-Click on update password 4-Enter new password 5-Enter confirm password 5-Click submit	New password= "Mahmoud@123" Confirm new password= "Mahmoud@1"	User alerted about invalid confirm password	As expected	Pass

5.3.5 Search report test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
20	Check super admin can search report with valid data with matched report	1-Logged in 2-Enter profile page 3-Click on add homeless 4-Fill required data 5-Click search	Name= "محمود" Age=22 Gender = “male" Image = “uploadsImages/mahmoud.jpg”	Matched report will be showed	As expected	Pass
21	Check super admin can search report with invalid data	1-Logged in 2-Enter profile page 3-Click on add homeless 4-Fill required data 5-Click search	Name= "haadn " Age=22 Gender = “male" Image = “uploadsImages/mahmoud.jpg”	User alerted about invalid username	As expected	Pass
22	Check super admin can search report with valid data with unmatched report	1-Logged in 2-Enter profile page 3-Click on add homeless 4-Fill required data 5-Click search	Name= "سيف" Age=22 Gender = “male" Image = “uploadsImages/mahmoud.jpg”	A message is shown that no report is matched	As expected	Pass
23	Check communicate with relative when a report is matched	1-Logged in 2-Enter profile page 3-Click on add homeless 4-Enter name, age, gender And photo 5-Click search (in case found) 6-Click on communicate with relative 7-Fill required data 8- Click submit	Name=” سيف” Age=22 Gender =“male” Image=”uploadsImages/mahmoud.jpg” Description=”شعر اسود و عينه بني بشرته بيضاء و ” Found time=7/5/2019 Found location=”Cairo” Shelter Name= “elmaady “ Finder name=” محمد علي سامي” Finder Nation ID= 29924890102285 Finder Phone= 01045876587 Finder Email= “Sasd@gmail.com”	Mail will be sent to relative	As expected	Pass

5.3.6 Add homeless test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
24	Check admin can add homeless with valid data	1-Logged in 2-Enter profile page 3-Click on add homeless 4-Enter required data 5-Click search(not matched) 6-Click on continue to Add homeless 7-Fill required data 8-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description="شعر اسود و عينه بني بشرته بيضاء و " Found time=7/5/2019 Found location="Cairo" Shelter Name= "elmaady " Finder name=" محمد علي سامي" Finder Nation ID= 29924890102285 Finder Phone= 01045876587 Finder Email= "Sasd@gmail.com"	A message is shown that homeless is added successfully	As expected	Pass
25	Check admin can add homeless with invalid data	1-Logged in 2-Enter profile page 3-Click on add homeless 4-Enter required data 5-Click search(not matched) 6-Click on continue to Add homeless 7-Fill required data 8-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description="شعر اسود و عينه بني بشرته بيضاء و " Found time=7/5/2019 Found location="Cairo" Shelter Name= "elmaady " Finder name=" محمد علي سامي" Finder Nation ID= 2992489010nacaja Finder Phone= 01045876587 Finder Email= "Sasd@gmail.com"	User alerted about invalid inputs	As expected	Pass

5.3.7 Edit homeless test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
26	Check admin can edit homeless with valid data	1-Logged in 2-Enter profile page 3-Click on homeless page 4-select homeless 5-Click edit homeless 6-Fill required data 7-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description="شعر اسود و عينه بني بشرته بيضاء و " Found time=7/5/2019 Found location="Cairo" Shelter Name= "elmaady " Finder name=" محمد علي سامي" Finder Nation ID= 29924890102285 Finder Phone= 01045876587 Finder Email= "Sasd@gmail.com"	A message is shown that homeless is updated successfully	As expected	Pass
27	Check admin can edit homeless with invalid data	1-Logged in 2-Enter profile page 3-Click on homeless page 4-select homeless 5-Click edit homeless 6-Fill required data 7-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description="شعر اسود و عينه بني بشرته بيضاء و " Found time=7/5/2019 Found location="Cairo" Shelter Name= "elmaady " Finder name=" محمد علي سامي" Finder Nation ID= 29924890102kljj Finder Phone= 01045876587 Finder Email= "Sasd"	User alerted about invalid data	As expected	Pass

5.3.8 Add report test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
28	Check admin can add report with valid data	1-Logged in 2-Enter profile page 3-Click on add report 4- Fill required data 5-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description=" شعر اسود و عينه بشرته بيضاء و بني " Lost time=7/5/2019 Lost location="Cairo" Police station Name= "elmaady " Reporter name=" محمد علي سامي" Reporter Nation ID= 29924890102285 Reporter Phone= 01045876587 Reporter Email= "Sasd@gmail.com"	A message is shown that report is added successfully	As expected	Pass
29	Check admin can add report with invalid data	1-Logged in 2-Enter profile page 3-Click on add report 4- Fill required data 5-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description=" شعر اسود و عينه بشرته بيضاء و بني " Lost time=7/5/2019 Lost location="Cairo" Police station Name= "elmaady " Reporter name=" محمد علي سامي" Reporter Nation ID="kjknk" Reporter Phone= 010458765 Reporter Email= "Sasd@"	User alerted about invalid data	As expected	Pass

5.3.9 Edit report test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
30	Check admin can edit report with valid data	1-Logged in 2-Enter profile page 3-Click on report page 4-select report 5-Click edit report 6-Fill required data 7-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description=" شعر اسود و عينه بشرته بيضاء و بني " Lost time=7/5/2019 Lost location="Cairo" Police station Name= "elmaady " Reporter name=" محمد علي سامي" Reporter Nation ID= 29924890102285 Reporter Phone= 01045876587 Reporter Email= "Sasd@gmail.com"	A message is shown that report is updated successfully	As expected	Pass
31	Check admin can edit report with invalid data	1-Logged in 2-Enter profile page 3-Click on homeless page 4-select homeless 5-Click edit homeless 6-Fill required data 7-Click submit	Name=" سيف" Age=22 Gender ="male" Image="uploadsImages/mahmoud.jpg" Description=" شعر اسود و عينه بشرته بيضاء و بني " Lost time=7/5/2019 Lost location="Cairo" Police station Name= "elmaady " Reporter name=" محمد علي سامي" Reporter Nation ID="kjknk" Reporter Phone= 010458765 Reporter Email= "Sasd@"	User alerted about invalid data	As expected	Pass

5.3.10 Search homeless test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
32	Check guest can search homeless with valid data with matched homeless	1-Enter home page 2-Fill required data 3-Click search	Name= "محمود" Age=22 Gender = “male" Image = “uploadsImages/mahmoud.jpg ”	Matched homeless will be showed	As expected	Pass
33	Check guest can search homeless with valid data with unmatched homeless	1-Enter home page 2-Fill required data 3-Click search	Name= "محمود" Age=22 Gender = “male" Image = “uploadsImages/mahmoud.jpg ”	A clickable button will showed to make a report	As expected	Pass
34	Check guest can search homeless with invalid data	1-Enter home page 2-Fill required data 3-Click search	Name= "1257 " Age=22 Gender = “male" Image = “uploadsImages/mahmoud.jpg ”	Guest alerted about invalid data	As expected	Pass

5.3.11 Change report status test cases

Test case ID	Test scenario	Test steps	Test data	Expected results	Actual result	Pass or fail
35	Check admin can change report status	1-Logged in 2-Enter profile page 3-Click on report page 4-Select report 5-Click on chosen status		A message is shown that report status is updated successfully	As expected	Pass

6- Results and Discussion

In this chapter, we will discuss the results; expected and actual, difficulties faced, experiences gained, results achieved, our recommendation and our final thoughts on the project.

6.1 Result

6.1.1 Expected Result

- **PoliceStation and shelter can register**
- **PoliceStation ,shelter and superAdmin can login ,forgetPassword ,update-password , update Profile, search Resident Homeless, display Resident Homeless and logout .**
- **SuperAdmin can change privileges, checkRequestToJoin, delete shelter, display-shelter, deletePoliceStation and displayPoliceStation.**
- **SuperAdmin PoliceStation can editReport, addHomeless, communicateWithRelative, changeReportStatus, searchReport, viewReport and closeHomeless.**
- **Guest can search and submit-Report**

6.1.2 Actual Result

- **PoliceStation and shelter can register**
- **PoliceStation ,shelter and superAdmin can login ,forgetPassword ,update-password , updateProfile, searchResidentHomeless, displayResidentHomeless, and logout .**
- **SuperAdmin can change privileges, checkRequestToJoin, delete shelter, display-shelter, deletePoliceStation and displayPoliceStation.**
- **SuperAdmin PoliceStation can editReport, addHomeless, communicateWithRelative, changeReportStatus, searchReport, viewReport and closeHomeless.**
- **Guest can search and submit-Report**

6.2 Discussion

We have managed to get the same expected results

6.3 the difficult we face

- **Find and integrating facial verification API.**
- **Integrate global positioning system (GPS).**
- **Upload images from Angular to node.js.**
- **Arranging time because it was our first time to deal with a complex project like this**

6.4 Experiences gained

- **Planning and arranging our project and avoid mistakes.**
- **Working with node.js, express and MongoDB.**
- **Working with 3rd party module like (face verifications and GPS).**
- **Working with angular.**
- **Integration backend-API with Angular.**

6.5 Results achieve

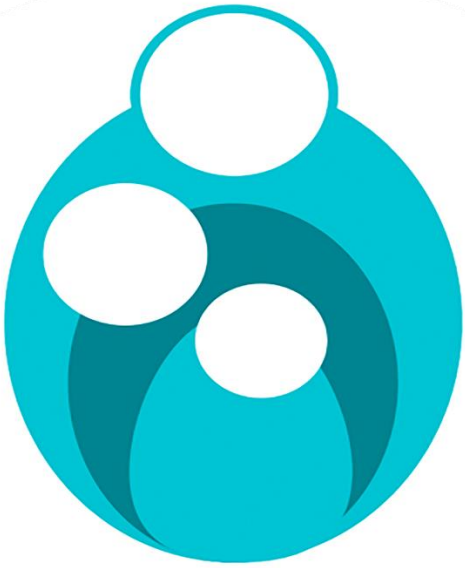
- **We achieve runnable website that works well and handles data inputs, make its calculations, and bring satisfying outputs and results.**
- **Combine all data under unified data source.**
- **Mange to reconnecting family.**
- **Make it more easy and secure way to find missing people.**
- **Avoid fraud and corruption.**

6.6 Final thoughts on the project

- **We need more cooperation with government organizations to provide us with the required data from police stations and shelters**
- **Obtaining support from licensed and supervised government source**
- **An expanded advertising campaign to inform all people of the site's services and its importance**

6.7 Our recommendation

- **Using more powerful and more durable equipment and the existence of the necessary time and human capabilities of software engineers, software developers, and security engineers, as well as the presence of test engineers and the existence of a budget appropriate for the work we can get more results and add new features.**
- **Because the equipment used in the implementation of this project can be considered as only prototypes as well as we did not have all knowledge about the best tools that needs learning and work on them.**
- **Therefore, the availability of experienced engineers in the field of software, testing, Security, mechatronics, and electronics, as well as providing the appropriate budget not only for the use of better tools and equipment but also for marketing the product better consider our main recommendations for this helpful project.**



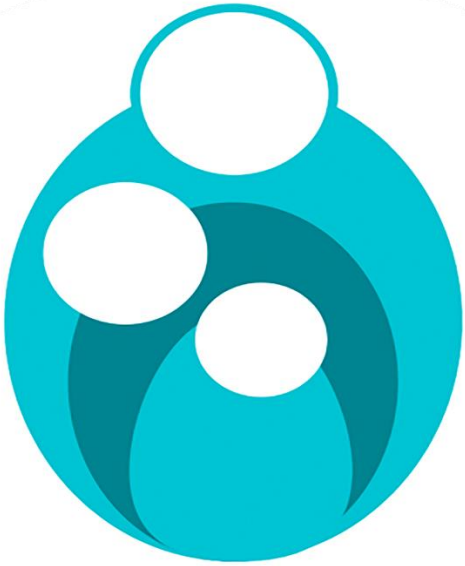
7- Future work

in this chapter, we will purpose what we are planning to do in the future

We are planning to develop more items to make the applications work the best afterwards...

7.1 REUNION Future Work:

1. **Integrate our back-end API with Android and iOS.**
2. **Handling multiple languages view .**
3. **Integrate with Closed-circuit television (CCTV) to make live scanning of missing.**
4. **Create social part to receive information from people about last place watched in**

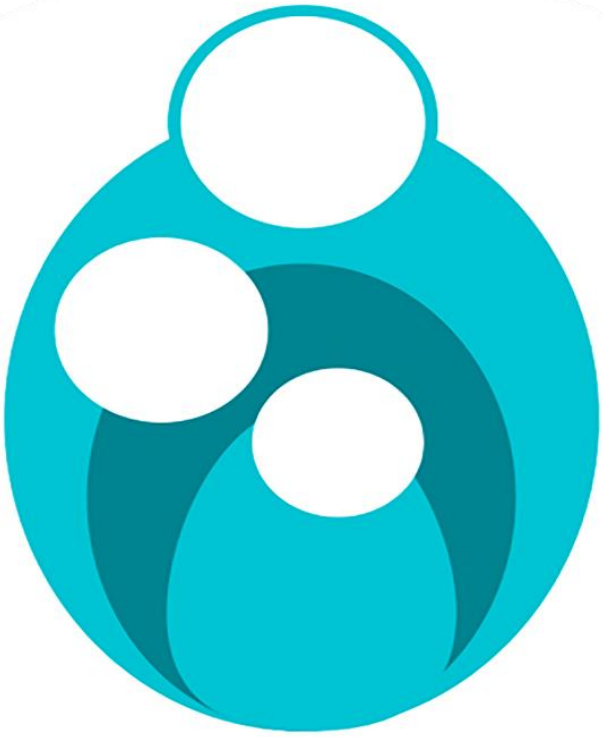


8- Conclusion

Eventually, in this project we tried to make the best we can to help families , kids, and government in our country. Consider it as a way of expressing our duty towards our society and every kid or man lives as a homeless ,we felt responsible to participate in trying to make life easy for them and their relatives by making a quick processing to find them in low time and low effort using powerful technologies.

In summary, REUNION is an application that help families in daily huge problem threatening future and life of thousands of people by following few steps in a smart and easy Way. It also helps him society and government by making a very useful connection between different management to save time instead of routine slow way . losing one of your family is a big problem for every one of us, people who suffer from that have the right to live their normal life..

Finally, realizing the fact that "a few clicks" can do a lot to a society, will definitely make us more aware of the technology role and become better versions of ourselves.



THANK YOU