Assignment 2 - CSE5006 (T4 2025)

Jack (22122100)

31/08/2025

CSE5006 (T4 2025)

Introduction
This document outlines the work completed for **Assignment 2** of the CSE5006 course. The task required the development of a cloud-based web application with a PostgreSQL backend, a Node.js API, and a React frontend. The main tasks include UI modifications, API command demonstrations, database modelling, and the creation of new features such as managing company records.

Build and Start the Services:

docker-compose up --build

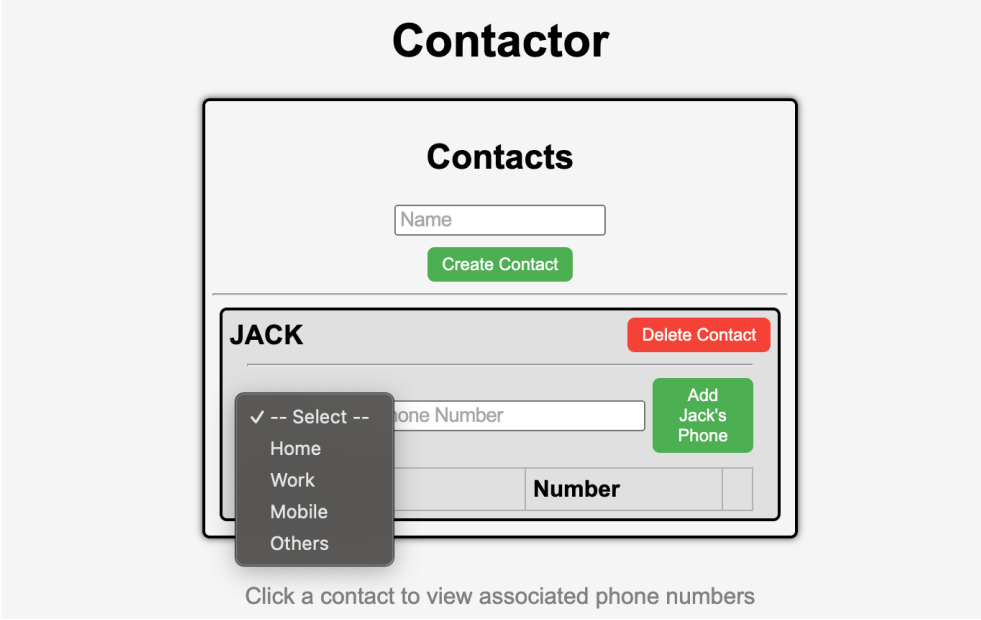
This command will build the Docker images for the frontend and backend services, pull the PostgreSQL image, and set up all the services. After the process completes, the application will be running locally.

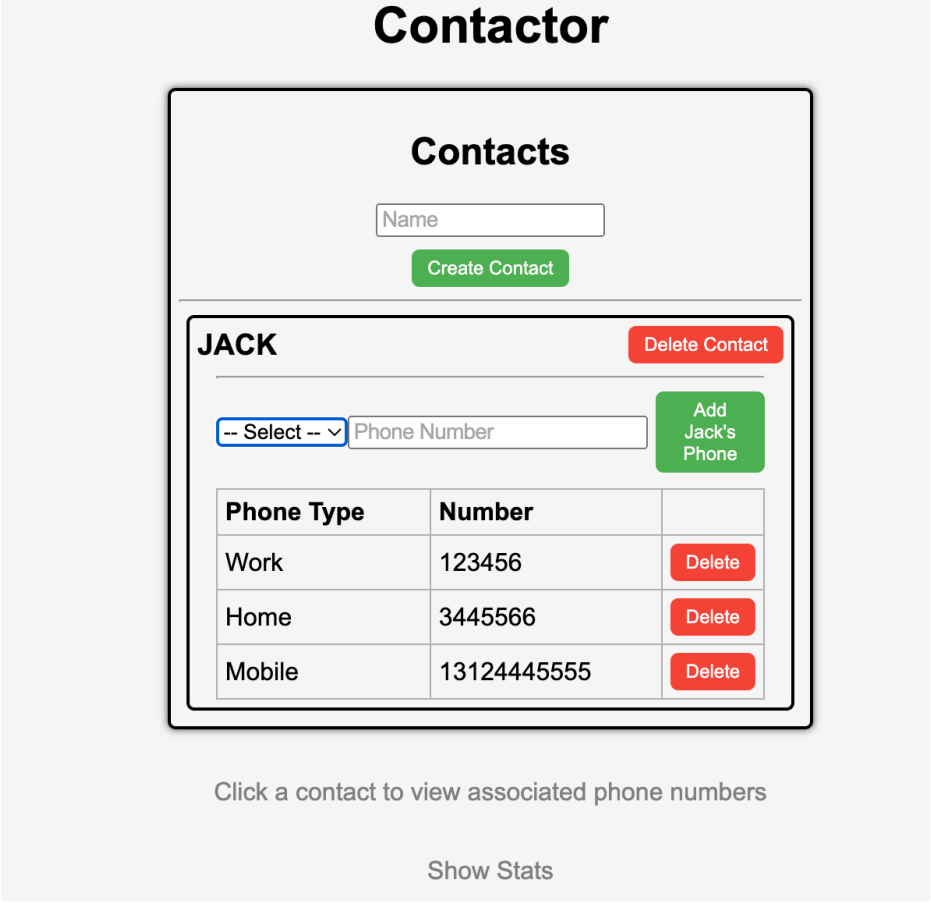
Then go to http://localhost to check the front end.

1. Task 1 – UI Changes

1.1 Changed "Delete" → "Delete Contact".
1.2 Changed "Add" → "Add <Name>'s Phone".
1.3 Changed "Name" input field → Dropdown with 4 options.
1.4 Changed table label: "Name" → "Phone Type".

Screenshot**:**


Contactor
Contacts
Name
Create Contact
JACK
Delete Contact
-- Select --
Home
Work
Mobile
Others
Add Jack's Phone
Number
Click a contact to view associated phone numbers


Contactor
Contacts
Name
Create Contact
JACK
Delete Contact
-- Select --
Phone Number
Add Jack's Phone
Phone Type
Number
Work
123456
Delete
Home
3445566
Delete
Mobile
13124445555
Delete
Click a contact to view associated phone numbers
Show Stats

2. TASK 2 – API Command Demonstrations

2.1 Show the API command for "Show Contact" and provide a screenshot of the output.

This command is used to fetch all contacts from the backend.
Command:

http GET http://localhost/api/contacts

Screenshot: (List is empty at the beginning)

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http GET http://localhost/api/contacts
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 2
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 06:59:58 GMT
ETag: W/"2-19Fw4VUO7kr8CvBlt4zaMCqXZ0w"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

[]
```

2.2 Show the API command for "Add Contact" and provide a screenshot of the output.

Command:

http POST http://localhost/api/contacts name="Jack"

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http POST http://localhost/api/contacts name="J
ack"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 101
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 07:02:49 GMT
ETag: W/"65-cEijmOOoeL4zR9ZxV/0wrZE0Ivc"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "createdAt": "2025-08-21T07:02:49.892Z",
    "id": 22,
    "name": "Jack",
    "updatedAt": "2025-08-21T07:02:49.892Z"
}
```

2.3 Show the API command for "Delete Contact" and provide a screenshot of the output.
Command:
http DELETE http://localhost/api/contacts/22

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http DELETE http://localhost/api/contacts/22
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 47
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 07:06:06 GMT
ETag: W/"2f-i0D5Qo4IGfH+OpTTITmyTnSzFvU"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Contact was deleted successfully!"
}
```

2.4 Show the API command for “Update Contact” and provide a screenshot of the output.

In order to update a contact, we need to add a Contact first. Then update it from “Jack” to “Jason”. Will use the Post first then ‘Put’.

Command:

http PUT http://localhost/api/contacts/23 name="Jason"

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http PUT http://localhost/api/contacts/23 name=
"Jason"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 47
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 07:09:24 GMT
ETag: W/"2f-9DEigpdI8FmatdY6qgJYc7CM5hQ"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Contact was updated successfully."
}
```

2.5 Show the API command for “Show Phone” and provide a screenshot of the output.

Now the phone list is empty so return [].

Command:

http GET http://localhost/api/contacts/23/phones

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http GET http://localhost/api/contacts/23/phone
s
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 2
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 07:11:38 GMT
ETag: W/"2-l9Fw4VUO7kr8CvBlt4zaMCqXZ0w"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

[]
```

2.6 Show the API command for “Add Phone” and provide a screenshot of the output.
Command:
http POST http://localhost/api/contacts/23/phones name="Work" number="10086"

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http POST http://localhost/api/contacts/23/phon
es name="Work" number="10086"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 133
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 07:16:11 GMT
ETag: W/"85-jQckus0dVSG5YlAF2KurVMZChOY"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "contactId": 23,
    "createdAt": "2025-08-21T07:16:11.815Z",
    "id": 11,
    "name": "Work",
    "number": "10086",
    "updatedAt": "2025-08-21T07:16:11.815Z"
}
```

2.7 Show the API command for “Update Phone” and provide a screenshot of the output.

I have changed the order of Update and Delete, update first then delete, so no need to add a phone again.
Command:
http PUT http://localhost/api/contacts/23/phones/11 name="Work" number="10000"

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http PUT http://localhost/api/contacts/23/phones/11 name="Work" number="10000"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 45
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 07:18:07 GMT
ETag: W/"2d-p9Lx2PQGimApZ9nkrVa0opZVZlQ"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Phone was updated successfully."
}
```

2.8 Show the API command for “Delete Phone” and provide a screenshot of the output.
Command:
http DELETE http://localhost/api/contacts/23/phones/11

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http DELETE http://localhost/api/contacts/23/phones/11
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 45
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 07:19:34 GMT
ETag: W/"2d-FdOer7L1Hk5YcQlrlpn01BrNJmA"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Phone was deleted successfully!"
}
```

3. Task 3 – Database Modeling and API Testing

In this section, I will describe the changes made to the database, the front-end adjustments required to reflect those changes, and the testing of the APIs related to the modified database.

3.1 Modify the Contacts Table
The contacts table was modified to include a new address field, which stores the physical address of each contact.

Changes Made:

In the Sequelize model for the contacts table (contact.model.js), I added the address attribute. Here’s the modification:

```
the address: {
        type: Sequelize.STRING,
        allowNull: true,   // Allow null for address
    },
```

This new field stores the contact's address as a string, and it is marked as nullable to allow for situations where no address is provided for a contact.

3.2 Modify the Phones Table

The phones table was modified to better represent the phone information associated with each contact. The original field name was renamed to phone_type to represent the type of phone (e.g., Mobile, Home, Work), and the number field was renamed to phone_number to better reflect the stored data.

Changes Made:

In the Sequelize model for the phones table (phone.model.js), I made the following updates:

```
},
phone_type: {
    type: Sequelize.STRING
},
phone_number: {
    type: Sequelize.STRING
},
```

These changes ensure that phone numbers are categorized by type and that both phone_type and phone_number are required fields.

3.3 Front-End Adjustments

After modifying the database schema, I updated the front-end to reflect these changes. Specifically, I updated the forms and components to handle the new address field for contacts and the updated phone information with the phone_type and phone_number fields.

Changes Made:

NewContacts.js

```
addconst [address, setAddress] = useState(""); // Added address state
```

```
body: JSON.stringify({
    name,
    address // Include address in the request body
})
```

```
orm className='new-contact' onSubmit={createContact}>
  <input type='text' placeholder='Name' onChange={(e) => setName(e.target.value)} value={name}/>
  <input type="text" placeholder="Address" onChange={(e) => setAddress(e.target.value)} value={address}
  <button className='button green' type='submit'>Create Contact</button>
form>
```

Contact.js

```
return (
    <div key={contact.id} className='contact' onClick={(e) => setExpanded(!expanded)}>
        <div className='title'>
            <h3>{contact.name}</h3>
            <h3>{contact.address}</h3>
            <button className='button red' onClick={doDelete}>Delete Contact</button>
        </div>
```

Besides, I add the “Contact summary” as shown in the demo.

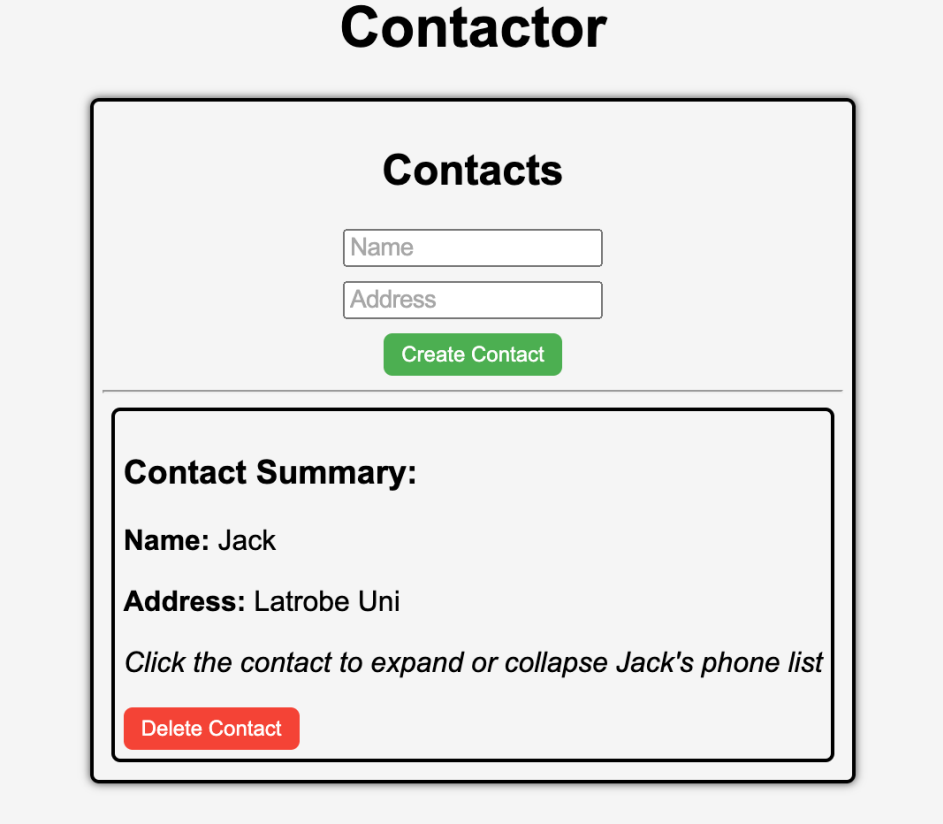
Modify the contacts.js and the contact.controller.js.

```
<div className='summary'>
          <h3>Contact Summary:</h3>
          <p><strong>Name:</strong> {contact.name}</p>
          <p><strong>Address:</strong> {contact.address}</p>
          <p><em>Click the contact to expand or collapse {contact.name}'s phone
list</em></p>
          <button className='button red' onClick={doDelete}>Delete Contact</button>
        </div>
```

```
const contact = {
    name: req.body.name,
    address: req.body.address
  };
```

When test, need to execute: docker compose down -v first. To make the database table structure consistent with the Sequelize model (the contacts table should include the address field.

Then docker compose up –build



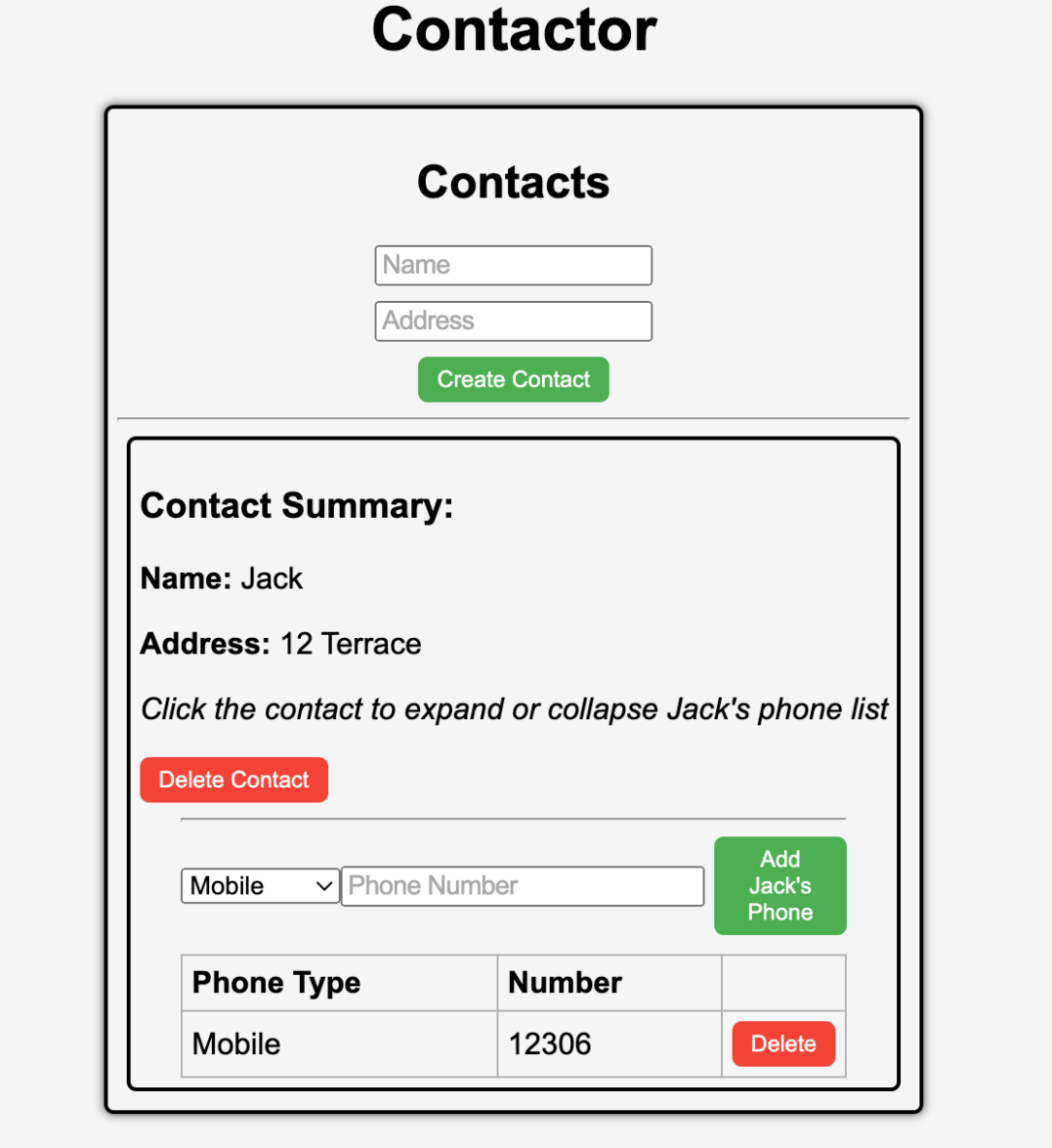
Similarly, modify to NewPhone.js and Phone.js and phone.controller.js.

```
    body: JSON.stringify({
        phone_type,
        phone_number
    })
```

```
const {contact, phones, setPhones} = props;
//const [number, setNumber] = useState('');
const [phone_number, setPhoneNumber] = useState('');
//const [name, setName] = useState('');
const [phone_type, setPhoneType] = useState("Mobile");
async function createPhone(e) {
    e.preventDefault();

    const response = await fetch('http://localhost/api/contacts/' + contact.id + '/phones', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            phone_type,
            phone_number
        })
    });

    const data = await response.json();

    if (data.id) {
        setPhones([...phones, data]);
    }

    setPhoneNumber('');
    setPhoneType('');
}

return (
    <form onSubmit={createPhone} onClick={(e) => e.stopPropagation()} className='new-phone'>

        <select value={phone_type} onChange={(e) => setPhoneType(e.target.value)}>
            <option value="">-- Select --</option>
            <option value="Home">Home</option>
            <option value="Work">Work</option>
```

```
JS NewContact.js  M
JS NewPhone.js    M
JS Phone.js
18      <td>{ phone.phone_type}</td>
19      <td>{ phone.phone_number}</td>
20      <td style={
21          {
```

```
// create phone
exports.create = (req, res) => {
    const phone = {
        phone_type: req.body.phone_type,
        phone_number: req.body.phone_number,
        contactId: parseInt(req.params.contactId)
    };
```



3.4 API Testing After Modifications

Once the database schema and the front-end were modified, I thoroughly tested the relevant APIs to ensure they returned the correct data and handled the new fields properly.

3.4.1 Show the API command for “Show Contact” and provide a screenshot of the output.

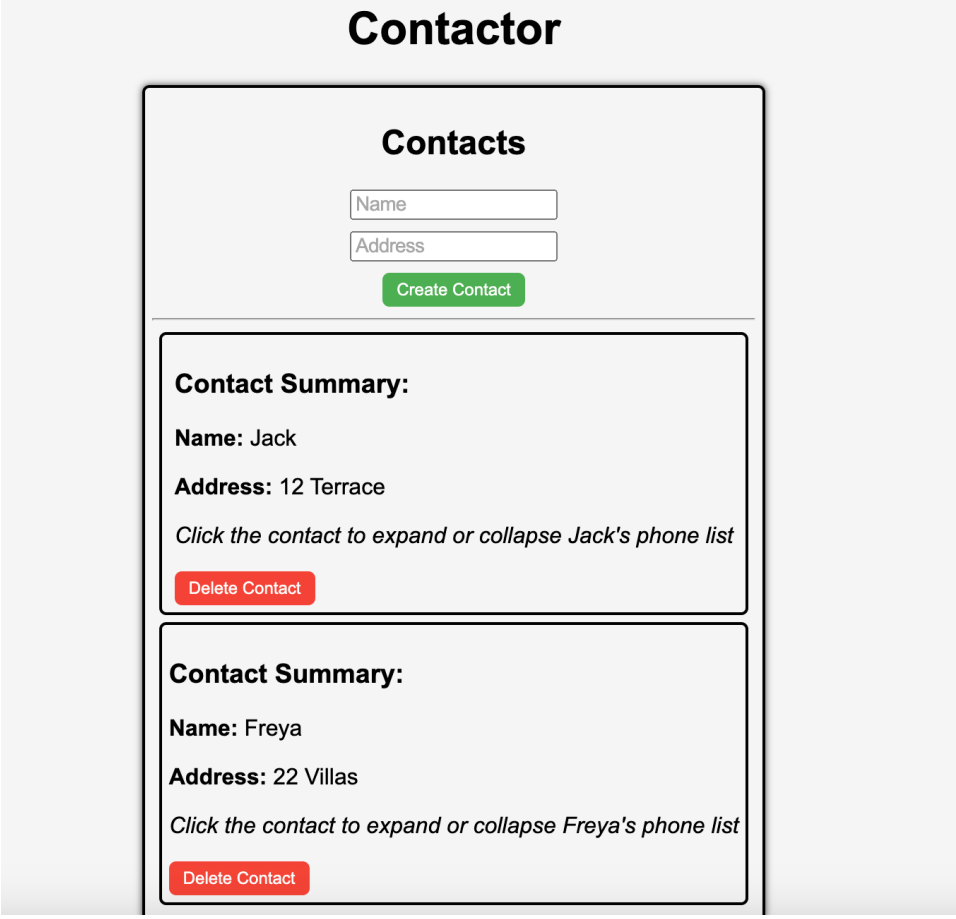
Command:

http GET http://localhost/api/contacts

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http GET http://localhost/api/contacts
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 125
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 09:42:27 GMT
ETag: W/"7d-G5nbs6/X4+wrIQcevUGXUbRg+iY"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

[
    {
        "address": "12 Terrace",
        "createdAt": "2025-08-21T09:38:46.114Z",
        "id": 1,
        "name": "Jack",
        "updatedAt": "2025-08-21T09:38:46.114Z"
    }
]
```

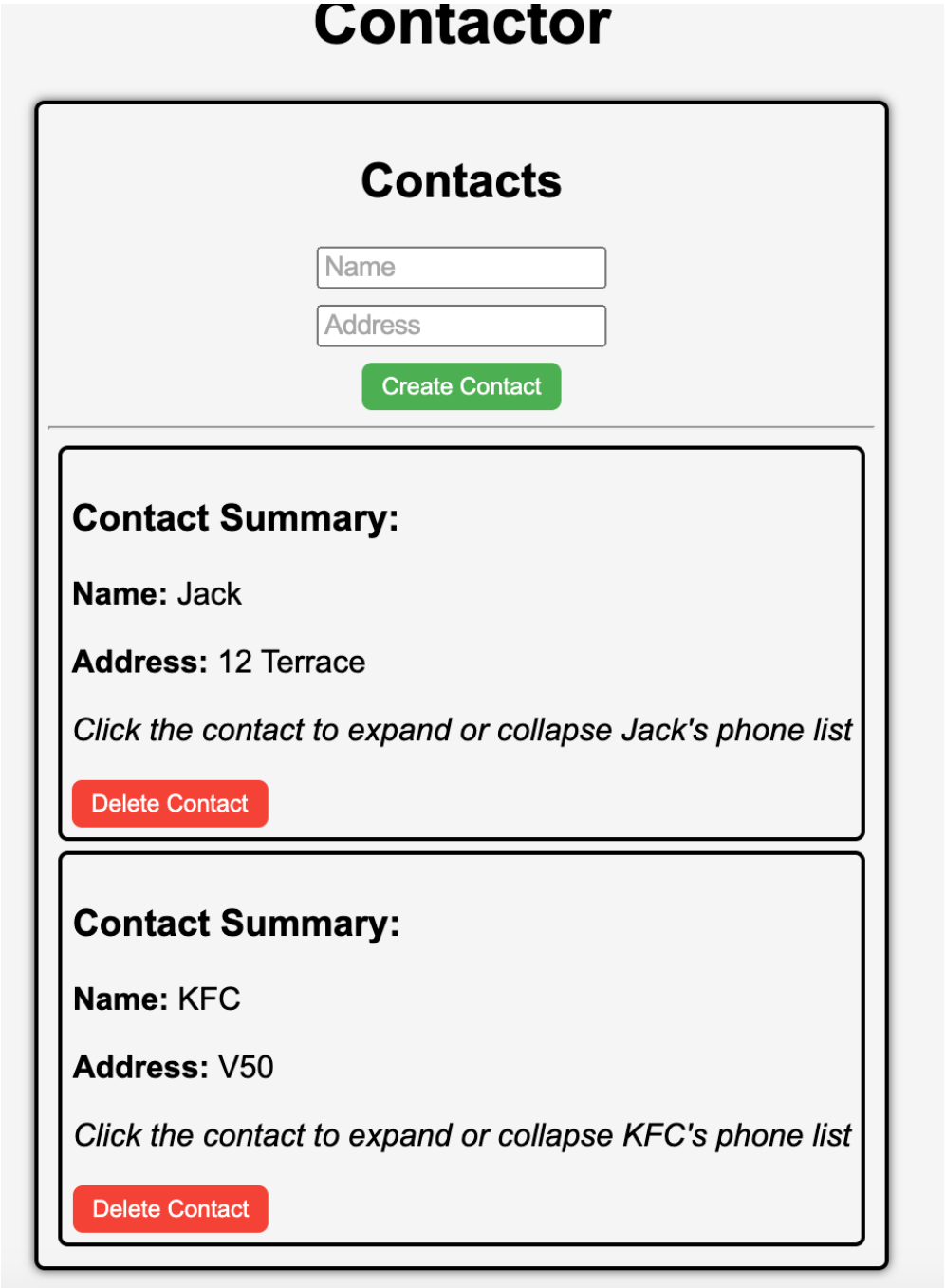
3.4.2 Show the API command for “Add Contact” and provide a screenshot of the output.
Command:
http POST http://localhost/api/contacts name="Freya" address="22 Villas"

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http POST http://localhost/api/contacts name="F
reya" address="22 Villas"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 123
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 09:48:25 GMT
ETag: W/"7b-idhc0cSz9c3RFFRZAxjXcuTsiHI"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "address": "22 Villas",
    "createdAt": "2025-08-21T09:48:25.843Z",
    "id": 4,
    "name": "Freya",
    "updatedAt": "2025-08-21T09:48:25.843Z"
}
```



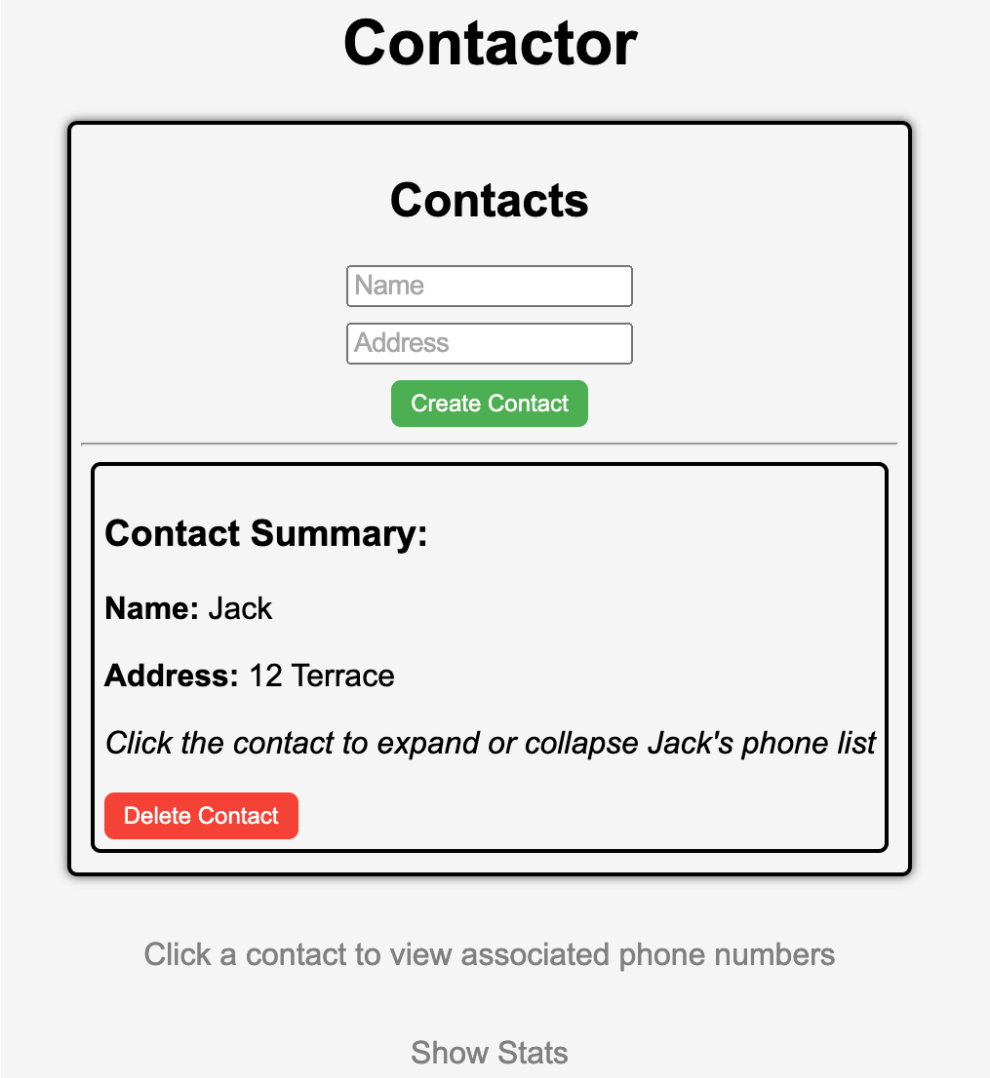
3.4.3 Show the API command for “Update Contact” and provide a screenshot of the output.
Command:
http PUT http://localhost/api/contacts/4 name="KFC" address="V50"

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http PUT http://localhost/api/contacts/4 name="
KFC" address="V50"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 47
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 09:51:26 GMT
ETag: W/"2f-9DEigpdI8FmatdY6qgJYc7CM5hQ"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Contact was updated successfully."
}
```



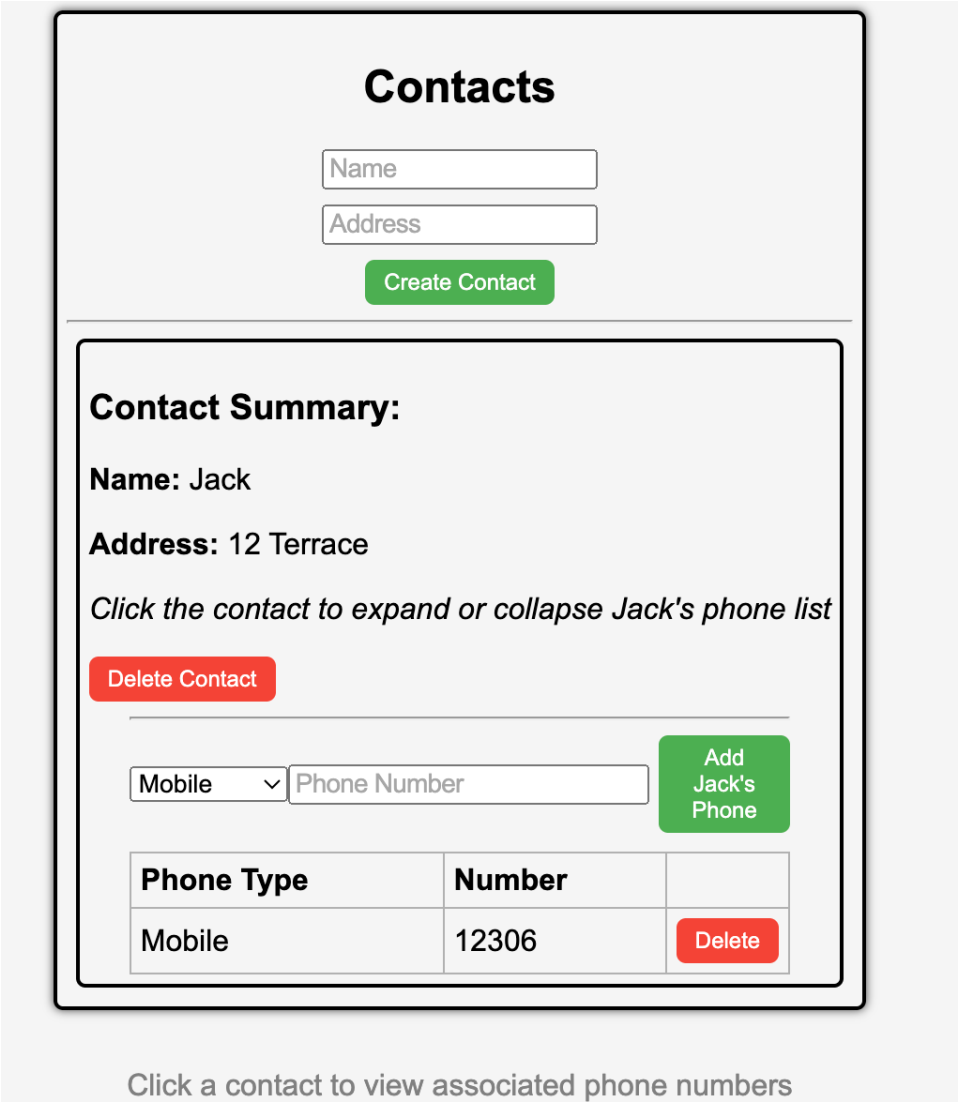
3.4.4 Show the API command for “Delete Contact” and provide a screenshot of the output.
Command:
http DELETE http://localhost/api/contacts/4

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http DELETE http://localhost/api/contacts/4
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 47
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 09:54:35 GMT
ETag: W/"2f-i0D5Qo4IGfH+OpTTITmyTnSzFvU"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Contact was deleted successfully!"
}
```



3.4.5 Show the API command for “Show Phone” and provide a screenshot of the output.
Command:
http GET http://localhost/api/contacts/1/phones

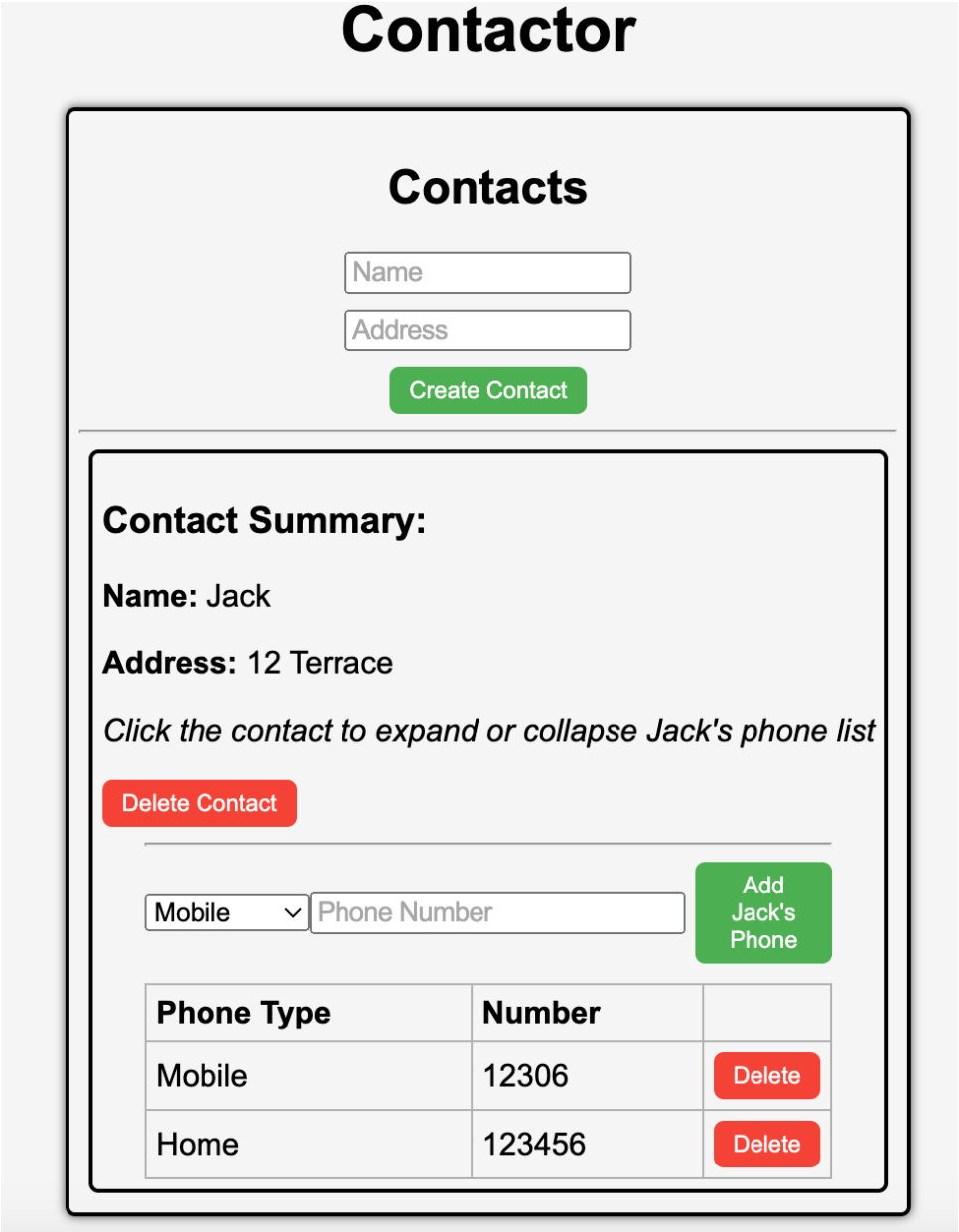
```
(base) jackhao@HaojiedeMacBook-Pro ~ % http GET http://localhost/api/contacts/1/phones

HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 147
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 12:07:19 GMT
ETag: W/"93-cOeHuR4clzTjfDPSiQopvjH2qTk"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

[
    {
        "contactId": 1,
        "createdAt": "2025-08-21T09:40:36.863Z",
        "id": 4,
        "phone_number": "12306",
        "phone_type": "Mobile",
        "updatedAt": "2025-08-21T09:40:36.863Z"
    }
]
```



3.4.6 Show the API command for “Add Phone” and provide a screenshot of the output.

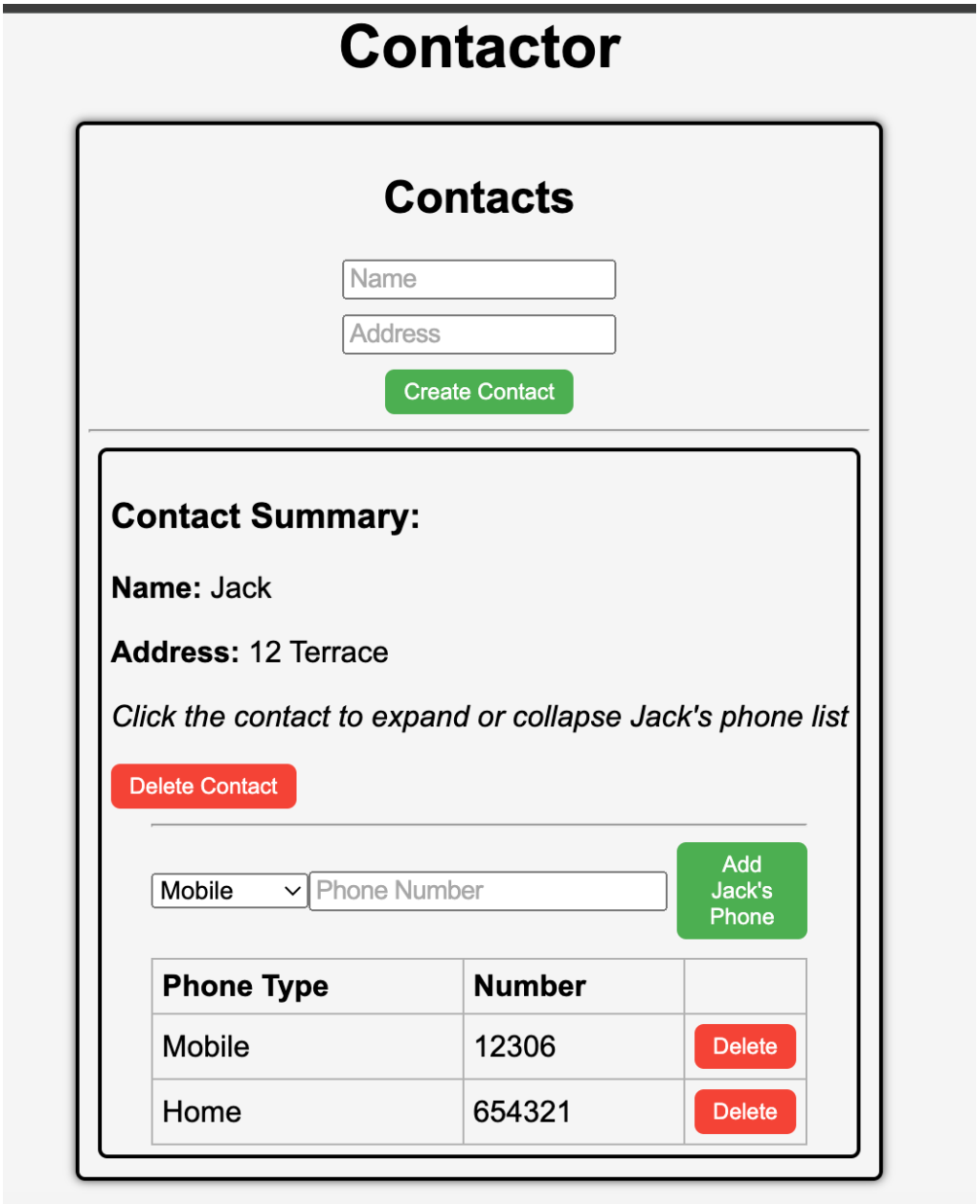
Command:
http POST http://localhost/api/contacts/1/phones phone_type="Home" phone_number="123456"

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http POST http://localhost/api/contacts/1/phone
s phone_type="Home" phone_number="123456"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 144
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 12:09:21 GMT
ETag: W/"90-V+QlDnu1R7fPZXp6R60y61N9pIQ"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "contactId": 1,
    "createdAt": "2025-08-21T12:09:21.393Z",
    "id": 5,
    "phone_number": "123456",
    "phone_type": "Home",
    "updatedAt": "2025-08-21T12:09:21.393Z"
}
```



3.4.7 Show the API command for “Update Phone” and provide a screenshot of the output.
Command:
http PUT http://localhost/api/contacts/1/phones/5 phone_type="Home" phone_number="654321"

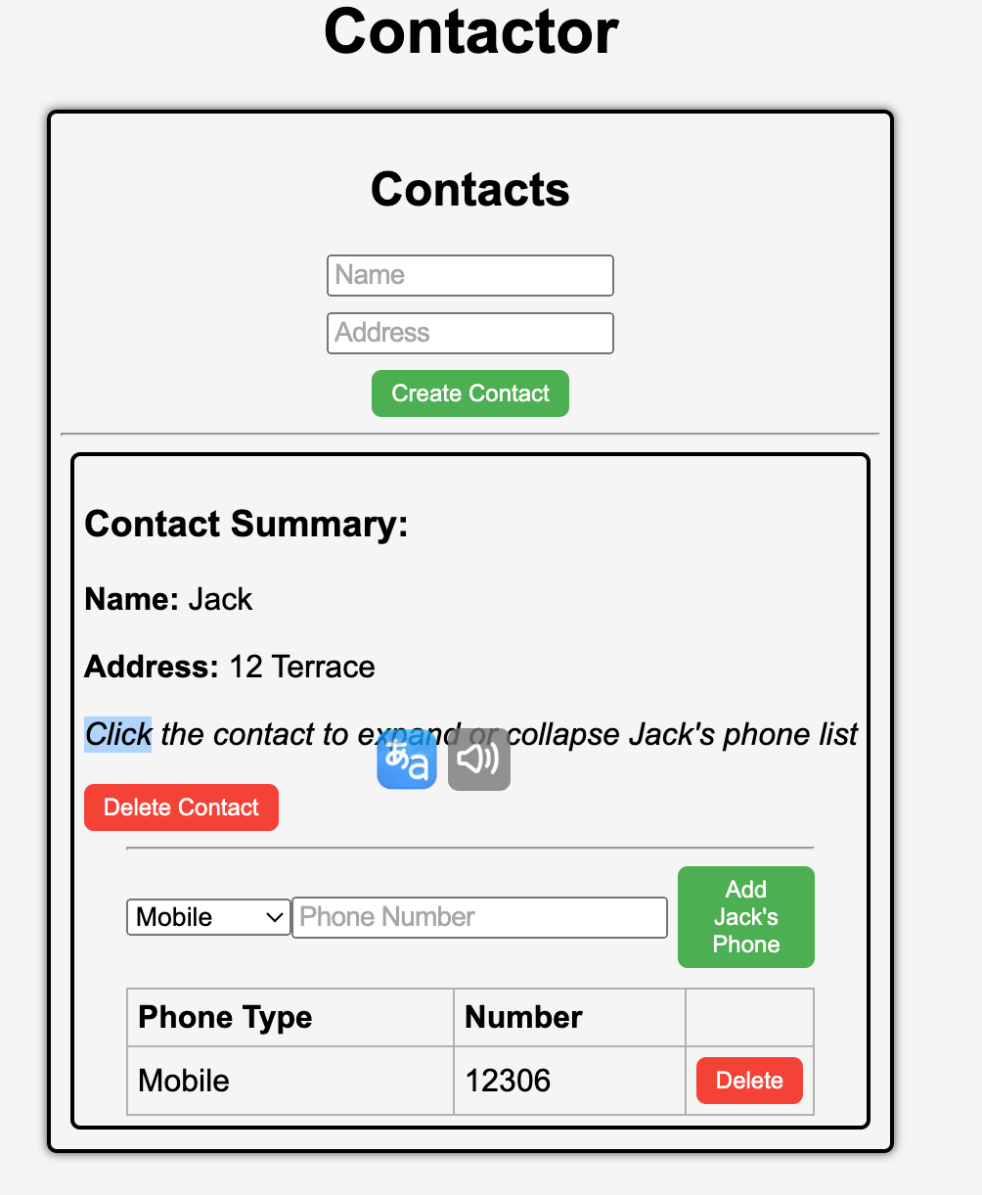
```
(base) jackhao@HaojiedeMacBook-Pro ~ % http PUT http://localhost/api/contacts/1/phones
/5 phone_type="Home" phone_number="654321"
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 45
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 12:13:23 GMT
ETag: W/"2d-p9Lx2PQGimApZ9nkrVa0opZVZlQ"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Phone was updated successfully."
}
```



3.4.8 Show the API command for “Delete Phone” and provide a screenshot of the output.

Command:
http DELETE http://localhost/api/contacts/1/phones/5

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http DELETE http://localhost/api/contacts/1/pho
nes/5
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 45
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 12:15:08 GMT
ETag: W/"2d-FdOer7L1Hk5YcQlrlpn01BrNJmA"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Phone was deleted successfully!"
}
```



By following these steps, I successfully modified the database schema, adjusted the front-end accordingly, and thoroughly tested the APIs to ensure everything works as expected. All

changes were successfully reflected in the system, and the application is now capable of managing contacts with detailed address and phone information.

4. Task 4 Expanding the Existing Tables

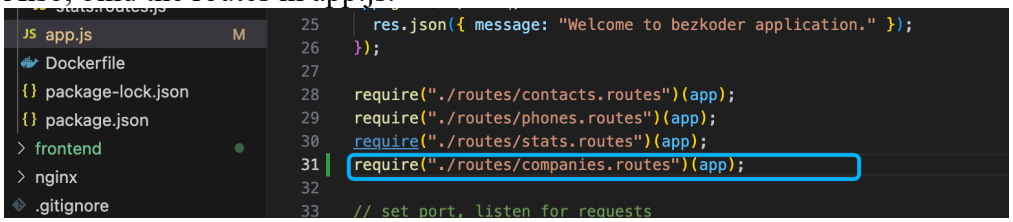
In this task, I was required to create a new table called companies in the database and establish relationships with the existing contacts table. This task also involves developing APIs to interact with the companies table, including CRUD operations (Create, Read, Update, Delete). Below is a step-by-step explanation of how I implemented this task.

4.1 Create the Companies Table

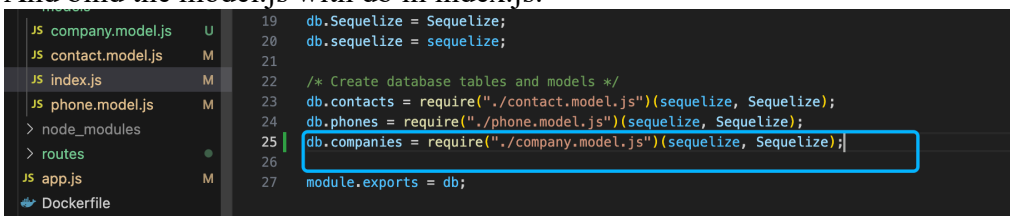
The goal of this part is to design a new table called companies, which will store information related to companies. Each company will be associated with a contact (through the contact_id foreign key).

I created the new companies table by adding a Sequelize model in the models folder. Check the code in company.models.js, company.controller.js and companies.routes.js for more details.

Also, bind the routes in app.js.



And bind the model.js with db in index.js.



Then rebuild the database to make the company table take effect.

4.2 Develop API Endpoints for Companies

Now that the companies table is created, I needed to implement API endpoints to allow users to interact with the table. These endpoints will provide CRUD operations for managing company records.

4.2.1 Create Company (POST /companies)

Command:

```
http POST http://localhost/api/companies \
  company_name="La Trobe Uni" \
  company_address="Bendigo" \
```

contact_id:=1

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http POST http://localhost/api/companies \
  company_name="La Trobe Uni" \
  company_address="Bendigo" \
  contact_id:=1

HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 167
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 23:47:44 GMT
ETag: W/"a7-e/kRkYu3EAes7wC9vKxiM992DXE"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "company_address": "Bendigo",
    "company_id": 2,
    "company_name": "La Trobe Uni",
    "contact_id": 1,
    "createdAt": "2025-08-21T23:47:43.963Z",
    "updatedAt": "2025-08-21T23:47:43.963Z"
}
```

4.2.2 Get All Companies (GET /companies)

Command:
http GET http://localhost/api/companies

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http GET http://localhost/api/companies
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 169
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 23:48:53 GMT
ETag: W/"a9-eBqnKN01EF8TT8Li78Ko1O+eQwI"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

[
    {
        "company_address": "Bendigo",
        "company_id": 2,
        "company_name": "La Trobe Uni",
        "contact_id": 1,
        "createdAt": "2025-08-21T23:47:43.963Z",
        "updatedAt": "2025-08-21T23:47:43.963Z"
    }
]
```

4.2.3 Update Company by ID

Command:
http PUT http://localhost/api/companies/2 \
  company_name="KFC" \
  company_address="V50" \
  contact_id:=1

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http PUT http://localhost/api/companies/2 \
  company_name="KFC" \
  company_address="V50" \
  contact_id:=1

HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 47
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 23:50:47 GMT
ETag: W/"2f-tSOkyn1aLnHg00JkjP0hv/QxH7Q"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Company was updated successfully."
}
```

4.2.4 Delete Company (DELETE /companies/:id)
Command:
http DELETE http://localhost/api/companies/2

```
(base) jackhao@HaojiedeMacBook-Pro ~ % http DELETE http://localhost/api/companies/2
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 47
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Aug 2025 23:51:56 GMT
ETag: W/"2f-goeWLYgQgcZh1o2QS0V4ovFdEa0"
Server: nginx/1.25.1
Vary: Origin
X-Powered-By: Express

{
    "message": "Company was deleted successfully!"
}
```

In this task, I expanded the existing tables by creating the companies table and implementing the necessary relationships with the contacts table. Additionally, I developed API endpoints for performing CRUD operations on the companies table. These APIs were thoroughly tested to ensure they functioned correctly, and the application is now capable of managing companies and their associated contacts.

5. Task 5 Front End

In this task, I implemented a front-end interface that allows users to manage companies associated with a specific contact. Instead of creating a separate page for companies, the companies list is displayed under each contact in the contact's details page. Each company is linked to the contact via the contact_id, and users can add, view, and delete companies associated with that contact.

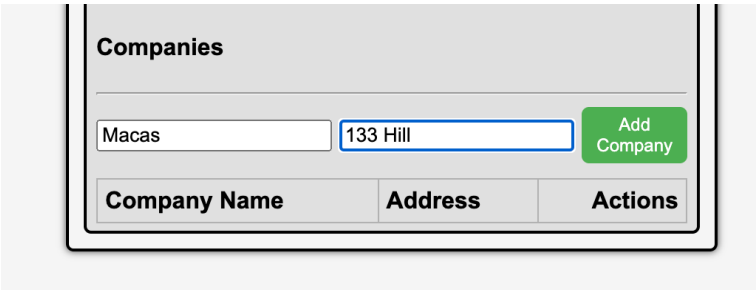
5.1 Front-End Requirements

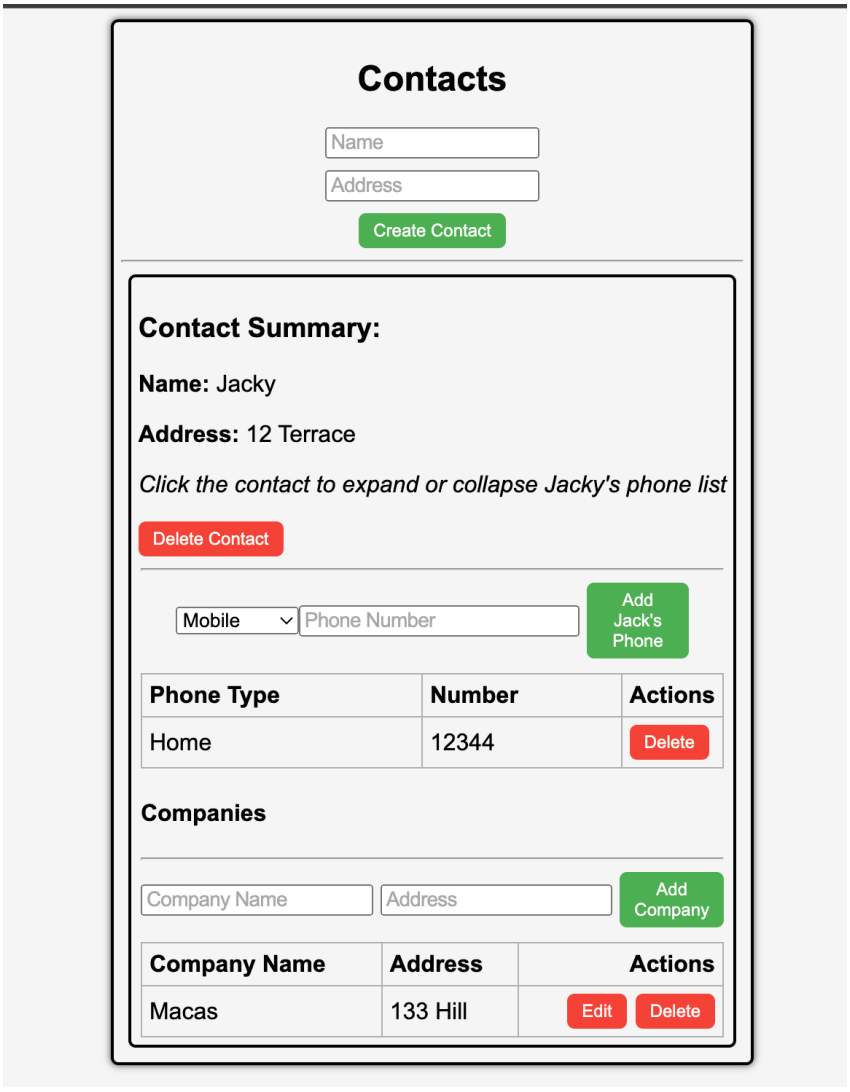
The requirements for managing companies were integrated into the Contact View. Specifically, the functionality includes:

1. Add Company under a Contact

Users can add a new company to a contact's details by filling in a form that includes the company name, address.The form appears as a collapsible section under the contact details.

Upon submission, the form sends a POST request to the backend (/contacts/:contactId/companies), adding the company to the specific contact.



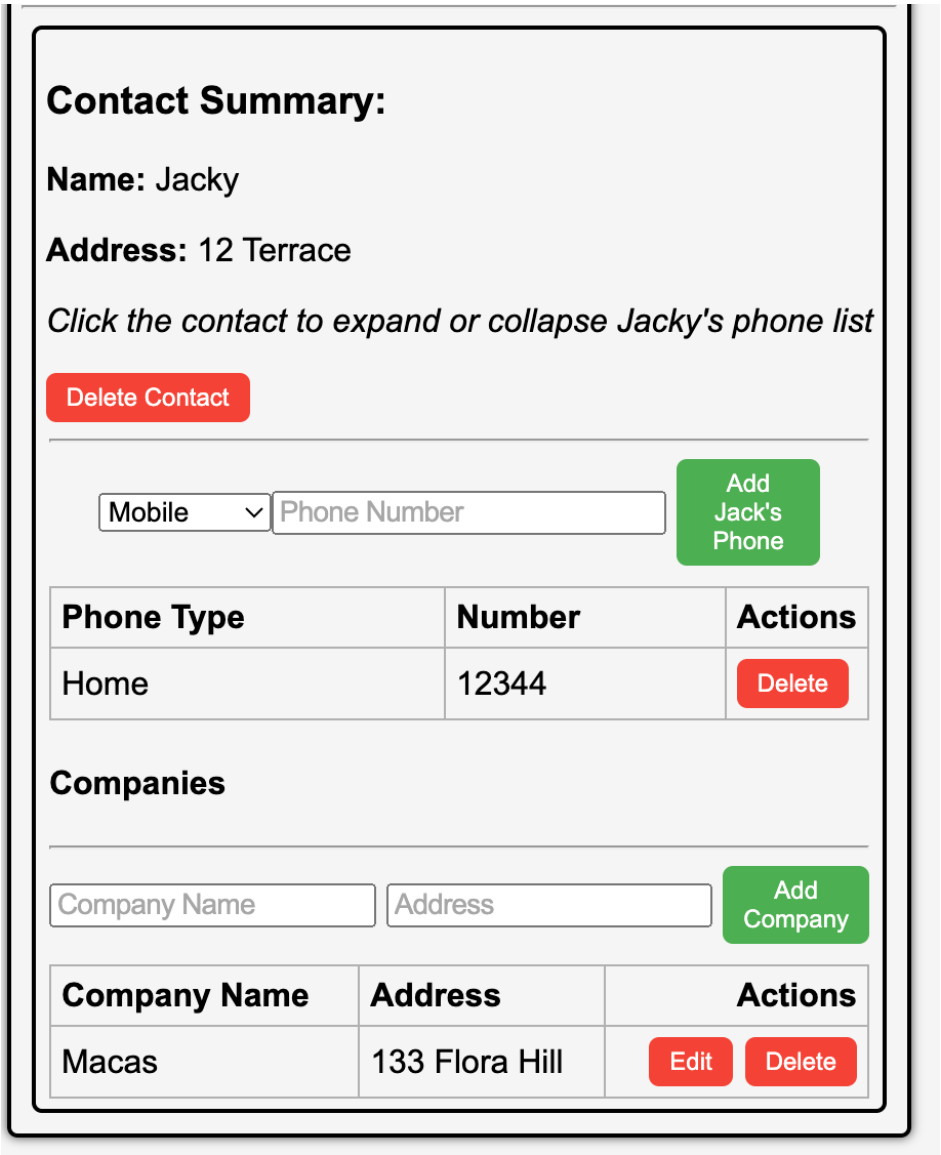


2. **View Companies for a Contact**

   When viewing a contact, users can expand a section that lists all companies associated with that specific contact. The companies are fetched from the database and displayed in this section.

3. **Update Companies for a Contact**

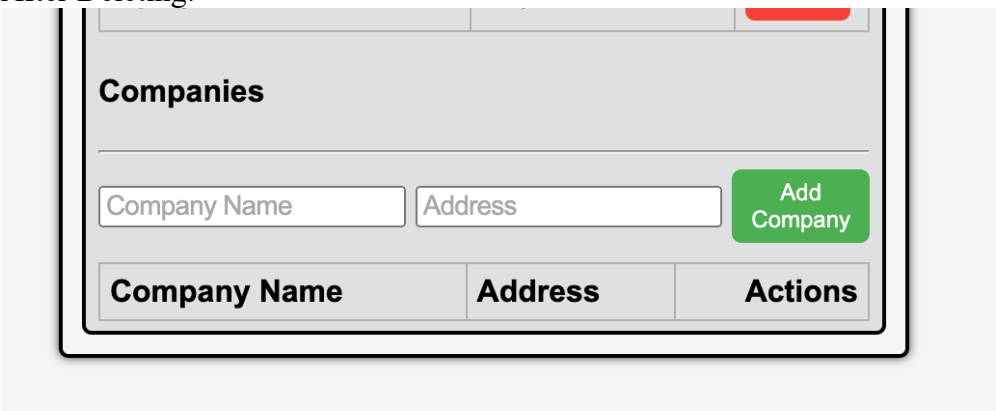
   When updating a company, user can click the “edit’ button then the company name and company address become editable. After finishing editing, click ‘save’ to update the information.



4. **Delete Company from a Contact**

Each company listed under a contact can be deleted by clicking a "Delete" button next to the company. This sends a DELETE request to the backend to remove the company from the database and the contact's company list.

After Deleting:



5.2 Implementation Details

Check the code for more details.

In Task 5, I successfully embedded company management features directly under each contact's details, instead of creating a standalone page. This design reflects the relationship between contacts and companies via contact_id and provides users with an intuitive and seamless way to manage related company records. The system now supports adding, viewing, editing and deleting companies, ensuring full CRUD functionality for this new entity within the web application.