Lab Exercise: Search, Edit and Delete Features

Objective:
- Implement a search feature to find contacts by name.
- Implement an edit function to allow users to edit and update contact details.
- Implement a delete function to allow users to delete contact details.

Prerequisites:
- Basic understanding of Express.js (for routing and handling HTTP requests).
- Familiarity with Pug (a template engine for rendering HTML).
- Knowledge of SQL queries (to interact with a database)

Part 1 – Search Function

1. Add a search form to the current pug file (addressbook.pug)

```
form(action="/addressBook", method="post")
    input(type="text", name="searchkey", placeholder="Search by name")
    button(type="submit") Search
```

2. Add a new route to handle the search request in the 'addressbook' Express router

```
router.post('/addressBook', (req, res) => {
    const searchKey = req.body.searchkey;
    const sql = 'SELECT * FROM contacts WHERE name LIKE ?';
    db.query(sql, [`%${searchKey}%`], (err, results) => {
        if (err) throw err;
        res.render('addressBook', { contacts: results });
    });
});
```

i) `router.post('/addressBook', (req, res) => {...})`: Sets up a POST route at the `/addressBook` endpoint. When a POST request is made to this endpoint, the function provided as the second argument is called. This function takes two arguments: `req` (the request object) and `res` (the response object).

ii) `const searchKey = req.body.searchkey;`: Extracts the `searchkey` from the body of the request and assigns it to the `searchKey` constant.

iii) `const sql = 'SELECT * FROM contacts WHERE name LIKE ?';`: Defines an SQL query that selects all records from the `contacts` table where the `name` is like the provided search key.

iv) `db.query(sql, ['%${searchKey}%'], (err, results) => {...})`: Executes the SQL query. The `?` in the SQL query is replaced by the value of `searchKey`, surrounded by `%` symbols to allow for a substring match. If there's an error executing the query (`err` is not `null`), an exception is thrown. Otherwise, the `results` of the query are passed to the next function.

v) `res.render('addressBook', { contacts: results });`: Sends a response to the client. The response is a rendered view called `addressBook`, and the `results` of the SQL query are passed to the view as `contacts`.

3. Test the search function on the browser.

Part 2 – Edit Function

1. In addressbook.pug, add a hyperlink within the `div(class="box")`

```
a(class='editBtn', href='edit/' + contact.id) Edit
```

2. Create edit.pug page and include this form

```
doctype html
html
  head
    title Edit Contact
    meta(name="viewport" content="width=device-width, initial-scale=1.0")
    link(href="/stylesheets/box.css", rel="stylesheet")
  body
    form(action=`/edit/${contact.id}`, method="post")
        input(type="text", name="name", placeholder="Name", value=contact.name)
        input(type="email", name="email", placeholder="Email",
value=contact.email)
        input(type="text", name="phone", placeholder="Phone",
value=contact.phone)
        button(type="submit") Edit Contact
```

3. Create a router file 'editContact.js' and add this code:

```
const express = require('express');
const router = express.Router();

module.exports = (db) => {

    router.get('/edit/:id', (req, res) => {
        const contactId = req.params.id;
        const query = 'SELECT * FROM contacts WHERE id = ?';

        db.query(query, [contactId], (err, result) => {
            if (err) throw err;
            if (result.length === 0) {
                res.redirect('/');
            } else {
                res.render('edit', { contact: result[0] });
            }
        });
    });
    router.post('/edit/:id', (req, res) => {
        const contactId = req.params.id;
        const { name, email, phone } = req.body;
        const query = 'UPDATE contacts SET name = ?, email = ?, phone = ? WHERE
id = ?';

        db.query(query, [name, email, phone, contactId], (err, result) => {
            if (err) throw err;
            res.redirect(`/addressBook`);
        });
    });
    return router;
};
```

i) `router.get('/edit/:id', (req, res) => { ... });`: Sets up a route for GET requests to '/edit/:id'. The ':id' is a route parameter that will match any string. Inside the callback function, it queries the database for a contact with the given ID. If a contact is found, it renders an 'edit' view with the contact's data. If not, it redirects to the home page.

ii) `router.post('/edit/:id', (req, res) => { ... });`: Sets up a route for POST requests to '/edit/:id'. It extracts the new contact data from the request body and updates the contact in the database. After the update, it redirects to the address book page.

4. Test the edit function by choosing the contact to edit.

Part 3 – Delete Function

1. In your addressbook.pug, add a hyperlink (similar to the edit link above) to trigger the delete action.

2. Create a route to handle the delete request. Retrieve the contact ID from the URL parameters and execute the deletion query.

3. Verify that the delete functionality works by selecting a contact and triggering the delete action.