**Niagara College Toronto**

**Student:**

Ronaldo Adriano Ferreira **Student ID:** 4524286

**Final Project: Database Integration.**

**Contemporary Storage Systems Programming (PROG1870)**

**Toronto**

**2023**

**Step by Step - final project Contemporary Storage**

Project for a store.

* **1 -** Record agents (customers/vendors/visitors/employees);
* **2 -** Search for agents recorded;
* **3** Record products;
* **4** Search products;

**1 – Record agents**

**Register Customers, Vendors, Visitors and employees.**

**HTML e JavaScript and Server configuration NodeJS**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <title>TechStore - Record Agents</title>

  <link rel="stylesheet" href="./css/style.css" />

</head>

<body>

  <h1 class="title">Record Agents</h1>

  <nav>

    <div class="nav\_links" id="navLinks">

      <ul>

        <li><a href="index.html">Home</a></li>

        <li><a href="recordAgent.html">Record Agents</a></li>

        <li><a href="searchAgent.html">Search Agents</a></li>

        <li><a href="recordProduct.html">Record Products</a></li>

        <li><a href="searchProduct.html">Search Products</a></li>

      </ul>

    </div>

    <br />

    <a href="index.html"><img src="//" alt="" /></a>

  </nav>

  <!-- Record Form -->

  <h2>Record The Agent</h2>

  <form id="registerForm">

    <label for="registerFirstName">First Name:</label>

    <input type="text" id="registerFirstName" name="registerFirstName" required /><br /><br />

    <label for="registerLastName">Last Name:</label>

    <input type="text" id="registerLastName" name="registerLastName" required /><br /><br />

    <label for="registerEmail">Email:</label>

    <input type="email" id="registerEmail" name="registerEmail" required /><br /><br />

    <!--PASSWORD:

      minimum 6 characters with letters, numbers and special characters

      In this code, the pattern attribute contains the regular expression ^(?=.\*[a-zA-Z])(?=.\*\d)(?=.\*[\W\_]).{6,}$, which imposes the following rules:

      ^: Start of string.

      (?=.\*[a-zA-Z]): Must contain at least one letter (upper or lower case).

      (?=.\*\d): Must contain at least one digit.

      (?=.\*[\W\_]): Must contain at least one special (non-alphanumeric) character.

      .{6,}: The password must have at least 6 characters of any type.

      $: End of string. -->

    <label for="registerPassword">Password:</label>

    <input type="password" id="registerPassword" name="registerPassword"

      pattern="^(?=.\*[a-zA-Z])(?=.\*\d)(?=.\*[\W\_]).{6,}$" required /><br /><span class="password-requirements">Minimum 6

      characters with letters, numbers and special characters.</span><br /><br>

    <label for="confirmPassword">Retype passoword:</label>

    <input type="password" id="confirmPassword" name="confirmPassword" required /><br /><br />

    <label for="userType">User Type:</label>

    <select id="userType" name="userType" required>

      <option value=""></option>

      <option value="customer">Customer</option>

      <option value="vendor">Vendor</option>

      <option value="visitor">Visitor</option>

      <option value="employee">Employee</option>

    </select><br /><br />

    <div id="employeeFields" style="display: none">

      <label for="employeeRole">Role:</label>

      <select id="employeeRole" name="employeeRole">

        <option value=""></option>

        <option value="sales">Sales</option>

        <option value="management">Management</option>

        <option value="other">Other</option>

      </select><br /><br />

    </div>

    <label for="phoneNumber">Phone Number:</label>

    <input type="text" id="phoneNumber" name="phoneNumber" pattern="[0-9]{10,12}" /><br /><br />

    <label for="countryCode">Country Code:</label>

    <input type="text" id="countryCode" name="countryCode" pattern="^\d{1,3}$" maxlength="3" /><br />

    <span class="password-requirements">Enter your international telephone code (numbers only).</span><br /><br>

    <label for="address">Address:</label>

    <input type="text" id="address" name="address" /><br /><br />

    <label for="zipcode">ZipCode:</label>

    <input type="text" id="zipcode" name="zipcode" pattern="[A-Za-z0-9]{6}" maxlength="6" /><br /><br />

    <label for="observations">Observations:</label><br>

    <textarea id="observations" name="observations"></textarea><br />

    <input type="submit" id="submitButton" value="Record" />

  </form>

  <script src="script/recordAgent.js"></script>

</body>

</html>

**Script - Creating a JavaScript file to handle data (client side with HTML): recorAgent.js**

document.addEventListener("DOMContentLoaded", function () {

  const form = document.getElementById("registerForm");

  const employeeFields = document.getElementById("employeeFields");

  const employeeRoleSelect = document.getElementById("employeeRole");

  // When user selects "Employee", display job title fields

  document.getElementById("userType").addEventListener("change", function () {

    if (this.value === "employee") {

      employeeFields.style.display = "block";

    } else {

      // Set the value of employeeRole to "other" when the userType is not "employee"

      employeeRoleSelect.value = "other";

      employeeFields.style.display = "none";

    }

  });

  form.addEventListener("submit", function (event) {

    event.preventDefault(); // Prevents default form submission

    // Collect form values

    const data = {

      registerFirstName: document.getElementById("registerFirstName").value,

      registerLastName: document.getElementById("registerLastName").value,

      registerEmail: document.getElementById("registerEmail").value,

      registerPassword: document.getElementById("registerPassword").value,

      confirmPassword: document.getElementById("confirmPassword").value,

      userType: document.getElementById("userType").value,

      role: document.getElementById("employeeRole").value,

      phoneNumber: document.getElementById("phoneNumber").value,

      countryCode: document.getElementById("countryCode").value,

      address: document.getElementById("address").value,

      zipcode: document.getElementById("zipcode").value,

      observations: document.getElementById("observations").value,

    };

    // Validate that passwords match

    if (data.registerPassword !== data.confirmPassword) {

      alert("Passwords do not match. Please try again.");

      return;

    }

    //Send data to server

    fetch("http://localhost:3000/register", {

      method: "POST",

      body: JSON.stringify(data),

      headers: {

        "Content-Type": "application/json",

      },

    })

      .then((response) => {

        if (response.status === 200) {

          alert("Successfully Recorded.");

          // Redirect or perform other actions after registration

          form.reset();

        } else {

          alert("Error when registering. Please, try again.");

        }

      })

      .catch((error) => {

        console.error("Error sending data:", error);

        alert("Error when registering. Please, try again.");

      });

  });

});

**Creating DB in data stax Cassandra: (DB: techstore )**

CREATE KEYSPACE IF NOT EXISTS “**techstore\_tabular**”

WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1}; (can be created using the graphic interface on Datastax Cassandra)

USE **techstore\_tabular**;

CREATE TABLE IF NOT EXISTS agents (

agent\_id UUID PRIMARY KEY,

first\_name TEXT,

last\_name TEXT,

email TEXT,

phone\_number TEXT,

country\_code TEXT,

address TEXT,

zipcode TEXT,

observations TEXT,

password TEXT,

user\_type TEXT,

role TEXT,

registration\_date TIMESTAMP

);

**INSERT INTO : (manual model)**

INSERT INTO agents (agent\_id, first\_name, last\_name, email, phone\_number, country\_code, address, zipcode, observations, password, user\_type, role, registration\_date)

VALUES (uuid(), 'John', 'Doe', 'johndoe@example.com', '123-456-7890', 'US', '123 Main St', '12345', 'Some observations', 'password123', 'user', 'role', toTimestamp(now()));

INSERT INTO agents (agent\_id, first\_name, last\_name, email, phone\_number, country\_code, address, zipcode, observations, password, user\_type, role, registration\_date)

VALUES (uuid(), 'Ronaldo', 'Ferreira', 'ronaldo@radriano.com', '437-459-7878', '+1', '410 Duff St', 'm6k0p1', 'Some observations', 'radmin', 'admin', 'admin', toTimestamp(now()));

**For communication using JavaScript is necessary:**

* install the NodeJS. (<https://nodejs.org/en/download> )
* Install Cassandra driver on VSCode (on terminal in VSCode use: *npm install cassandra-driver*).
* Install CORS (on terminal in VSCode use: *npm install cors*).
* Install module Http from Node.js (on terminal in VSCode use: *npm install http url body-parser*).

**Creating a JavaScript file to handle server HTTP (server side) to connect with Datastax Cassandra: “app.js”**

**ATTENTION: run app.js before to use and disable VPNs on your local machine**

// JavaScript (app.js)

const express = require("express");

const cors = require('cors');

const bodyParser = require("body-parser");

const { Client } = require("cassandra-driver");

const app = express();

const port = 3000;

app.use(cors());

app.use(bodyParser.json());

// Configure connection to Cassandra database

const client = new Client({

  cloud: {

    secureConnectBundle: "C:/Users/radri/OneDrive - GUSCanada/001 - NCT - Niagara College Toronto/02\_Term/01\_Contemporary Storage Systems Programming/week12\_final project/Project\_r2/nodejsProject/secure-connect-techstore.zip",

  },

  credentials: {

    username: "dzAwICWmslTGgSqFpZTSCQmm",

    password: "Xb4Y4eRNkmZgrr3iSIm0aJ98WtCnjQrDF6wK\_Sl-wEqwWY6mP8hewXX,9no0pYNv6MClnjQNgfOu68BWfZJP\_kx24ZaLeiJakn3pB4ftE.koJrhvPi7XTqKABfu9UZM\_",

    //keyspace: "techstore\_tabular", // Name of your keyspace

  },

});

client.connect();

app.post("/register", async (req, res) => {

  try {

    // await client.connect();

    const { registerFirstName, registerLastName, registerEmail,  phoneNumber,  countryCode, address, zipcode, observations, registerPassword, userType, role, confirmPassword } = req.body;

    console.log('req.body: ', req.body)

    // Inserting data on Datastax Cassandra DB

    const query = `INSERT INTO techstore\_tabular.agents (

        agent\_id,

        first\_name,

        last\_name,

        email,

        phone\_number,

        country\_code,

        address,

        zipcode,

        observations,

        password,

        user\_type,

        role,

        registration\_date)

        VALUES (uuid(),?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, toTimestamp(now()))`;

    await client.execute(query, [registerFirstName, registerLastName, registerEmail,  phoneNumber,  countryCode, address, zipcode, observations, registerPassword, userType, role]);

    res.status(200).json({ message: "Successfully registered." });

  } catch (error) {

    console.error("Error when registering:", error);

    res.status(500).json({ error: "Error when registering." });

  } finally {

    //await client.shutdown();

  }

});

app.get("/search", (req, res) => {

  const searchText = req.query.query;

  // Query to search records using Email

  const query = "SELECT agent\_id, first\_name, last\_name, email, phone\_number, country\_code, address, zipcode, observations, user\_type, role, registration\_date FROM techstore\_tabular.agents WHERE email = ? ALLOW FILTERING";

  const params = [searchText];

  client.execute(query, params, { prepare: true }, (err, result) => {

      if (err) {

          console.error("Error on search: ", err);

          res.status(500).json([]);

      } else {

          res.json(result.rows);

      }

  });

});

app.listen(port, () => {

  console.log(`Server running on port ${port}`);

});

**2 - Search for agents recorded**

Made Update in the server file (server side) “**app.js**” to include “search” tool:

app.get("/search", (req, res) => {

  const searchText = req.query.query;

  // Consulta para pesquisar registros com base no First Name, Email ou Telefone

  const query = "SELECT agent\_id, first\_name, last\_name, email, phone\_number, country\_code, address, zipcode, observations, user\_type, role, registration\_date FROM techstore\_tabular.agents WHERE email = ? ALLOW FILTERING";

  const params = [searchText];

  client.execute(query, params, { prepare: true }, (err, result) => {

      if (err) {

          console.error("Erro na pesquisa: ", err);

          res.status(500).json([]);

      } else {

          res.json(result.rows);

      }

  });

});

An HTML file “**searchAgent.html**” separately was included to handle Search Agent:

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>TechStore - Search Agent</title>

    <link rel="stylesheet" href="./css/style.css" />

  </head>

  <body>

    <h1 class="title">Search Agents</h1>

    <nav>

      <div class="nav\_links" id="navLinks">

          <ul>

              <li><a href="index.html">Home</a></li>

              <li><a href="recordAgent.html">Record Agents</a></li>

              <li><a href="searchAgent.html">Search Agents</a></li>

              <li><a href="recordProduct.html">Record Products</a></li>

              <li><a href="searchProduct.html">Search Products</a></li>

          </ul>

      </div>

      <br />

      <a href="index.html"><img src="//" alt="" /></a>

  </nav>

    <input type="text" id="searchInput"

      placeholder="Search for Email"

    /> <br><br>

    <button id="searchButton">Search</button>

    <br>

    <br>

    <div id="searchResults">

      <!-- Show results -->

    </div>

    <script src="script/searchAgents.js"></script>

  </body>

</html>

Including a JS file “**searchAgents.js**” to get data to be searched and retrieve results from database back to HTML.

document.addEventListener("DOMContentLoaded", function () {

  const searchInput = document.getElementById("searchInput");

  const searchButton = document.getElementById("searchButton");

  const searchResults = document.getElementById("searchResults");

  searchButton.addEventListener("click", function () {

    const searchText = searchInput.value.trim();

    // Clean previous results

    searchResults.innerHTML = "";

    // Execute the search

    fetch(`http://localhost:3000/search?query=${searchText}`)

      .then((response) => response.json())

      .then((data) => {

        if (data.length === 0) {

          searchResults.innerHTML = "No results found.";

        } else {

          data.forEach((result) => {

            const resultElement = document.createElement("div");

            resultElement.innerText = `Name: ${result.first\_name} ${result.last\_name}

            Email: ${result.email}

            Phone Number: +${result.country\_code} ${result.phone\_number}

            Address: ${result.address} - ZipCode: ${result.zipcode}

            Observations: ${result.Observations}

            `;

            searchResults.appendChild(resultElement);

          });

        }

      })

      .catch((error) => {

        console.error("Error performing the search: ", error);

      });

  });

});

**3 – Record Products**

* Create in DataStax Cassandra a new keyspace called “**techstore\_keyvalue**” within “techstore” database.
* **Create a table** in “**techstore\_keyvalue**” named as “Products”.

mutation {

createTable(

keyspaceName: "techstore\_keyvalue",

tableName: "products",

partitionKeys: [{ name: "product\_code", type: { basic: TEXT } }]

values: [{ name: "product\_name", type: { basic: TEXT } }]

)

}

If you need **drop the table** :

mutation {

dropTable(

keyspaceName: "techstore\_keyvalue",

tableName: "products",

)

}

**Inserting data** to test:

mutation {

insertproducts (

value: {product\_code: "ABC001", product\_name: "Apple Iphone 15 - test" }

) {

value {

product\_code,

product\_name

}

}

}

* Create an html form to record “Product Code” and “Product Name”.
* Create a JS file “recordProducts.js” to handle data to send to server.
* Install the library axios using: “**npm install axios**” on terminal.
* Add code on “app.js” (file server side) to handle the new form.
* Install the library axios using: “**npm install express-graphql**” on terminal.
* Add on html axios library:
* <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

Creating HTML form:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>TechStore - Record Products</title>

    <link rel="stylesheet" href="./css/style.css" />

</head>

<body>

    <h1 class="title">Record Products</h1>

    <nav>

        <div class="nav\_links" id="navLinks">

            <ul>

                <li><a href="index.html">Home</a></li>

                <li><a href="recordAgent.html">Record Agents</a></li>

                <li><a href="searchAgent.html">Search Agents</a></li>

                <li><a href="recordProduct.html">Record Products</a></li>

                <li><a href="searchProduct.html">Search Products</a></li>

            </ul>

        </div>

        <br />

        <a href="index.html"><img src="//" alt="" /></a>

    </nav>

    <div class="container">

        <h1 class="title">Record Products</h1>

    </div>

    <form id="productForm">

        <label for="productCode">Product Code:</label>

        <input type="text" id="productCode" name="productCode" pattern="[A-Za-z0-9]{6}" maxlength="6" required><br><span class="password-requirements">6 characters</span><br /><br>

        <label for="productName">Product Name:</label>

        <input type="text" id="productName" name="productName" required><br><br>

        <input type="submit" value="Record">

    </form>

    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

    <script src="script/recordProducts.js"></script>

</body>

</html>

recorProducts.js

document.addEventListener("DOMContentLoaded", function () {

    const productForm = document.getElementById("productForm");

    productForm.addEventListener("submit", function (event) {

      event.preventDefault();

      const productCode = document.getElementById("productCode").value;

      const productName = document.getElementById("productName").value;

      // Build the product data

      const productData = {

        product\_code: productCode,

        product\_name: productName,

      };

      // Send the data to the server - "http://localhost:3000/graphql"

      axios.post("http://localhost:3000/registerProduct", {

        product\_code: productCode,

        product\_name: productName,

      })

      .then(response => {

        // const result = response.data.data.insert\_products.applied;

        if (response.status === 200) {

          console.log("Product registered successfully.");

          alert("Product registered successfully!");

        } else {

          console.error("Error when registering product.");

          alert("Error registering product. Try again.");

        }

      })

      .catch(error => {

        console.error("Error sending data: ", error);

        alert("Error registering product. Try again.");

      });

    });

  });

**4 – Search for Products**

* Create a new html file: “**searchProduct.html**”
* Create a new js file to handle the research: “**searchProduct.js**”
* Make changes on server side “**app.js**”;

“**searchProduct.html**”

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>TechStore - Product Search</title>

    <link rel="stylesheet" href="./css/style.css" />

</head>

<body>

    <h1 class="title">Product Search</h1>

    <nav>

        <div class="nav\_links" id="navLinks">

            <ul>

                <li><a href="index.html">Home</a></li>

                <li><a href="recordAgent.html">Record Agents</a></li>

                <li><a href="searchAgent.html">Search Agents</a></li>

                <li><a href="recordProduct.html">Record Products</a></li>

                <li><a href="searchProduct.html">Search Products</a></li>

            </ul>

        </div>

        <br />

        <a href="index.html"><img src="//" alt="" /></a>

    </nav>

    <!-- <div class="container">

        <h1 class="title">Product Search</h1>

    </div> -->

    <input type="text" id="searchInput"

      placeholder="Search for Product Code"

    /> <br><br>

    <button id="searchButton">Search</button>

    <br>

    <br>

    <div id="searchResults">

      <!-- Show results -->

    </div>

    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

    <script src="script/searchProducts.js"></script>

</body>

</html>

“**searchProduct.js**”

document.addEventListener("DOMContentLoaded", function () {

    const searchInput = document.getElementById("searchInput");

    const searchButton = document.getElementById("searchButton");

    const searchResults = document.getElementById("searchResults");

    searchButton.addEventListener("click", function () {

        const productCode = searchInput.value.trim();

        // Clear previous results

        searchResults.innerHTML = "";

        // Run the search

        axios.get(`http://localhost:3000/searchProduct?productCode=${productCode}`)

            .then((response) => {

                const data = response.data;

                if (data.length === 0) {

                    searchResults.innerHTML = "No results found.";

                } else {

                    data.forEach((result) => {

                        const resultElement = document.createElement("div");

                        resultElement.innerText = `Product Code: ${result.product\_code}, Product Name: ${result.product\_name}`;

                        searchResults.appendChild(resultElement);

                    });

                }

            })

            .catch((error) => {

                console.error("Error performing the search: ", error);

            });

    });

});

“**app.js**” – updated

// JavaScript (app.js)

const express = require("express");

const cors = require('cors');

const bodyParser = require("body-parser");

const { Client } = require("cassandra-driver");

const { buildSchema } = require('graphql');

const { graphqlHTTP } = require('express-graphql');

const axios = require('axios');

const app = express();

const port = 3000;

app.use(cors());

app.use(bodyParser.json());

// Configure connection to Cassandra database

const client = new Client({

  cloud: {

    secureConnectBundle: "C:/Users/radri/OneDrive - GUSCanada/001 - NCT - Niagara College Toronto/02\_Term/01\_Contemporary Storage Systems Programming/week12\_final project/Project\_r2/nodejsProject/secure-connect-techstore.zip",

  },

  credentials: {

    username: "tzTiiKiiQbIIOwOhacvKiYgZ",

    password: "6TokGUlXEHZ9mbB\_2z\_Z-8TWo8.Zn3I7AASAa7AWvGgkx5PBDpjswMwjooQptWyCNLd-L5ZWP+5sx9kfP+RojZkUJNmgmsIrYQ4Z503wUd8sGyiYApv4UAlgh.q87A83",

    keyspace: "techstore\_keyvalue", // Name of your keyspace

  },

});

client.connect();

app.post("/register", async (req, res) => {

  try {

    // await client.connect();

    const { registerFirstName, registerLastName, registerEmail,  phoneNumber,  countryCode, address, zipcode, observations, registerPassword, userType, role, confirmPassword } = req.body;

    console.log('req.body: ', req.body)

    // Inserting data on Datastax Cassandra DB

    const query = `INSERT INTO techstore\_tabular.agents (

        agent\_id,

        first\_name,

        last\_name,

        email,

        phone\_number,

        country\_code,

        address,

        zipcode,

        observations,

        password,

        user\_type,

        role,

        registration\_date)

        VALUES (uuid(),?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, toTimestamp(now()))`;

    await client.execute(query, [registerFirstName, registerLastName, registerEmail,  phoneNumber,  countryCode, address, zipcode, observations, registerPassword, userType, role]);

    res.status(200).json({ message: "Successfully registered." });

  } catch (error) {

    console.error("Error when registering:", error);

    res.status(500).json({ error: "Error when registering." });

  } finally {

    //await client.shutdown();

  }

});

app.get("/search", (req, res) => {

  const searchText = req.query.query;

  // Query to search records using Email

  const query = "SELECT agent\_id, first\_name, last\_name, email, phone\_number, country\_code, address, zipcode, observations, user\_type, role, registration\_date FROM techstore\_tabular.agents WHERE email = ? ALLOW FILTERING";

  const params = [searchText];

  client.execute(query, params, { prepare: true }, (err, result) => {

      if (err) {

          console.error("Error on search: ", err);

          res.status(500).json([]);

      } else {

          res.json(result.rows);

      }

  });

});

// GraphQL connection configuration

const schema = buildSchema(`

  type Mutation {

    insert\_products(value: ProductInput): AppliedResult

  }

  input ProductInput {

    product\_code: String

    product\_name: String

    keyspace: String

  }

  type AppliedResult {

    value: Product

  }

  type Product {

    product\_code: String

    product\_name: String

  }

`);

const root = {

    insert\_products: async ({ value }) => {

        const query = `INSERT INTO ${value.keyspace}.products JSON ?`; // Incorporating the keyspace in the query

        const params = [JSON.stringify(value)];

        try {

            await client.execute(query, params, { prepare: true });

            return { value };

        } catch (error) {

            console.error('Error inserting data into Cassandra:', error);

            return { value: null };

        }

    },

};

// Middleware to parse the request body as JSON

app.use(bodyParser.json());

// Route to handle GraphQL query

app.use('/graphql', graphqlHTTP({

    schema: schema,

    rootValue: root,

    graphiql: true,

}));

// Route to handle product registration

app.post('/registerProduct', async (req, res) => {

  console.log("AQUII ", req.body)

  const { product\_code, product\_name } = req.body;

  const keyspace = 'techstore\_keyvalue';

  const query = `INSERT INTO ${keyspace}.products JSON ?`;

  const params = [JSON.stringify({ product\_code, product\_name })];

  try {

      await client.execute(query, params, { prepare: true });

      res.status(200).json({ success: true, message: 'Product registered successfully!' });

  } catch (error) {

      console.error('Error registering product:', error);

      res.status(500).json({ success: false, message: 'Error registering product. Try again.' });

  }

});

//executing the Product search

app.get("/searchProduct", (req, res) => {

  const productCode = req.query.productCode;

  // Query to search records using Product Code

  const query = "SELECT product\_code, product\_name FROM techstore\_keyvalue.products WHERE product\_code = ?";

  const params = [productCode];

  client.execute(query, params, { prepare: true }, (err, result) => {

      if (err) {

          console.error("Search error: ", err);

          res.status(500).json([]);

      } else {

          res.json(result.rows);

      }

  });

});

app.listen(port, () => {

  console.log(`Server running on port ${port}`);

});