INT202_Assignment1 จัดทำโดย 65130500013 นายชนกานต์ เครือหงษ์

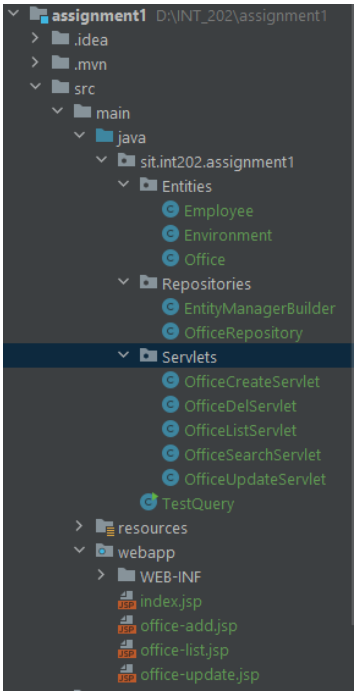
ใช้หลักการ MVC ในการออกแบบ Application ตัวนี้ซึ่งแบ่งออกเป็น 3 ส่วนหลักๆ มี

1. Model

2. View (JPA)

3. Controller (Servlets)

โครงสร้าง Project



1. Model จะมีอยู่ 1 คลาสก็คือ Office.java

```java
@Entity
@Getter
@Setter
@Table(name = "offices")
@NamedQueries({
        @NamedQuery(name = "OFFICE.FIND_ALL", query = "select o from Office o"),
        @NamedQuery(name = "OFFICE.FIND_BY_COUNTRY", query = "select o from Office o where o.country like :countryParam"),  //countryParam ตั้งชื่ออะไรก็ได้ ซึ่งการใส่ : คือเรียกใช้ parameter
        @NamedQuery(name = "OFFICE.FIND_BY_CITY_OR_COUNTRY", query = "select o from Office o where lower(o.city) like :city or lower(o.country) like :country")
})
public class Office {
    @Id
    private String officeCode;
    private String addressLine1;
    private String addressLine2;
    private String city;
    private String state;
    private String country;
    private String postalCode;
    @Column(name = "phone") // ChangeColumName
    private String phoneNumber;
    private String territory;
}
```

@Entity: บอกว่าคลาสนี้เป็น Entity ที่สามารถถูกจัดเก็บในฐานข้อมูลได้

@Getter และ @Setter: โค้ดจะถูก Lombok ใช้เพื่อสร้าง getter และ setter โดยอัตโนมัติ

@Table(name = "offices"): บอกว่า Entity นี้จะถูกเก็บในตารางที่มีชื่อว่า "offices" ในฐานข้อมูล

@NamedQueries: ใช้เพื่อกำหนดคิวรีที่ใช้บ่อยในรูปแบบของ Named Query เพื่อให้เรียกใช้งานได้ง่าย

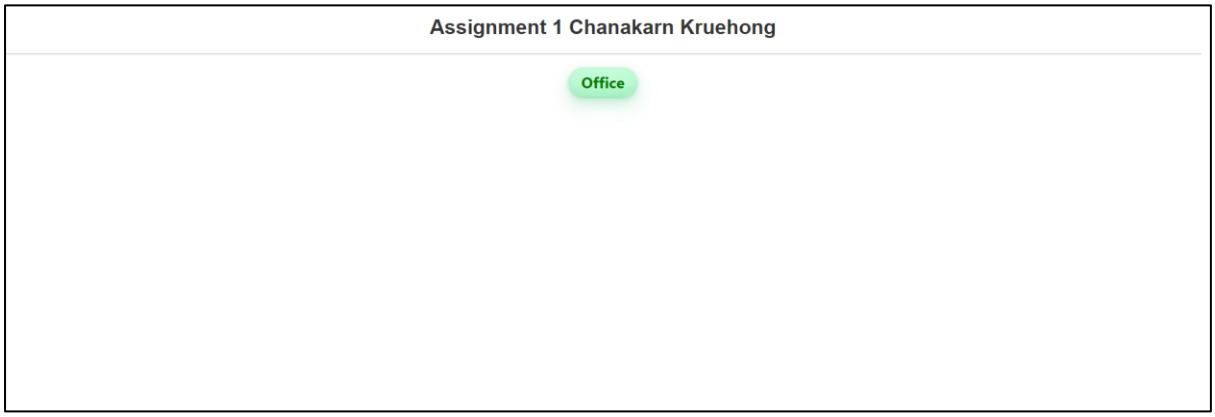
@Id: บอกว่า officeCode เป็น Primary Key ของ Entity

@Column (name = "phone"): บอกว่า attribute phoneNumber จะถูกเก็บในคอลัมน์ที่มีชื่อว่า "phone" ในฐานข้อมูล

2. View จะมีอยู่ 4 ไฟล์

2.1 index.jsp

```
<body>
<h1>Assignment 1 Chanakarn Kruehong</h1>
<hr>
<div class="center">
  <h2><a href="OfficeListServlet">Office</a></h2>
</div>
</body>
</html>
```



เป็นหน้าแรกเมื่อ User เปิดมาก็จะเห็นเมื่อกดปุ่ม office ก็จะ mapping ไปหา OfficeListServlet

2.2 office-list.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Kanit&display=swap" rel="stylesheet">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
<%--  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLf
  <meta charset="UTF-8">
  <title>Office-List</title>
```

ไฟล์ JSP นี้ใช้ Bootstrap CSS เพื่อจัดรูปแบบหน้าเว็บและสร้างสีของปุ่มที่สวยงาม

```jsp
<!-- ส่วนของ Navbar -->
<nav class="navbar bg-body-tertiary">
  <div class="container-fluid">
    <a class="navbar-brand">Offices List</a>
    <form class="d-flex" role="search" action="OfficeSearchServlet" method="post">
      <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search" name="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </div>
</nav>
```

มี Navbar ที่มีฟอร์มค้นหา (Search) เพื่อให้ผู้ใช้ค้นหา Offices ได้

```jsp
<c:if test="${not empty requestScope.selectedOffice}">
  <h3>Selected Office Details:</h3>
  <table>
    <tr>
      <th>Office Code</th>
      <th>City</th>
      <th>Phone</th>
      <th>Address Line 1</th>
      <th>Address Line 2</th>
      <th>State</th>
      <th>Country</th>
      <th>Postal Code</th>
      <th>Territory</th>
    </tr>
    <tr>
      <td>${requestScope.selectedOffice.officeCode}</td>
      <td>${requestScope.selectedOffice.city}</td>
      <td>${requestScope.selectedOffice.phoneNumber}</td>
      <td>${requestScope.selectedOffice.addressLine1}</td>
      <td>${requestScope.selectedOffice.addressLine2}</td>
      <td>${requestScope.selectedOffice.state}</td>
      <td>${requestScope.selectedOffice.country}</td>
      <td>${requestScope.selectedOffice.postalCode}</td>
      <td>${requestScope.selectedOffice.territory}</td>
    </tr>
  </table>
</c:if>
```

มี Navbar ที่มีฟอร์มค้นหา (Search) เพื่อให้ผู้ใช้ค้นหา Offices ได้

```jsp
<!-- แสดงรายการ Offices ทั้งหมด -->
<h3>List of Offices:</h3>
<table>
  <tr>
    <th>Office Code</th>
    <th>City</th>
    <th>Country</th>
    <th>Action</th>
  </tr>
  <c:if test="${messageErr == null}">
    <c:forEach var="office" items="${offices}">
      <tr>
        <td>${office.officeCode}</td>
        <td>${office.city}</td>
        <td>${office.country}</td>
        <td>
          <a class="btn btn-outline-info" href="${pageContext.request.contextPath}/OfficeListServlet?officeCode=${office.officeCode}">View Details </a>
          <a class="btn btn-outline-warning" href="${pageContext.request.contextPath}/OfficeUpdateServlet?officeCode=${office.officeCode}"> Update</a>
          <a class="btn btn-outline-danger" href="${pageContext.request.contextPath}/OfficeDelServlet?officeCode=${office.officeCode}" onclick="return confirm('Are you sure you want to delete this office?')"> Delete </a>
        </td>
      </tr>
    </c:forEach>
  </c:if>
</table>
```

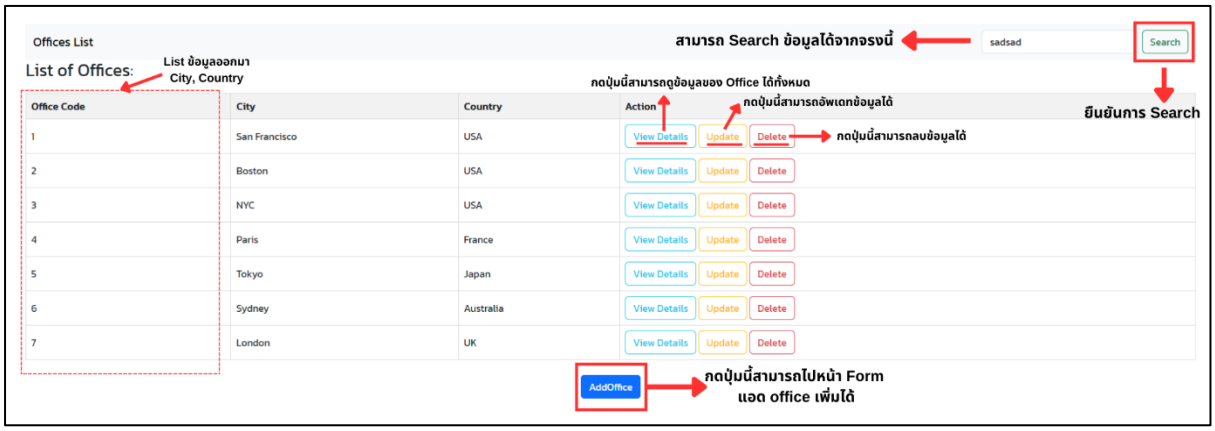
ตรวจสอบว่าตัวแปร messageErr มีค่าเป็น null หรือไม่ ถ้าเป็น null แสดงว่าไม่มีข้อผิดพลาด จึงทำการแสดงข้อมูล Offices ที่ได้จากลูป forEach ใช้ forEach เพื่อวนลูปข้อมูล Offices ที่ได้จาก ${offices} แสดง ${office.officeCode}, ${office.city}, ${office.country}: แสดงข้อมูลของแต่ละ Office ในแต่ละคอลัมน์ของตาราง มีปุ่ม "ViewDetails", "Update", และ "Delete" ให้ผู้ใช้ทำการดูรายละเอียด, แก้ไข, หรือลบข้อมูลของ Office นั้น ๆ ตามลำดับ การใช้ onclick="return confirm('Are you sure you want to delete this office?')" เพื่อให้มีการยืนยันจากผู้ใช้ก่อนที่จะทำการลบข้อมูล Office นั้น ๆ

```
<%--แสดง messageErr เมื่อ Search หาข้อมูลไม่เจอ--%>
<c:if test="${messageErr != null}">
  <!-- ถ้า messageErr ไม่เท่ากับ null -->
  <h2 class="center" style="color: red">${messageErr}<br>
    <a href="OfficeListServlet">
      <!-- ปุ่มที่ให้ผู้ใช้คลิกเพื่อกลับไปยังหน้ารายการ Offices -->
      <button type="button" class="btn btn-outline-secondary">Back To List of Offices</button>
    </a>
  </h2>
</c:if>
```

ตัวแปร messageErr ถ้ามีค่าไม่เท่ากับ null จะแสดงข้อความที่ติดตัวแปรนี้พร้อมกับปุ่มที่ผู้ใช้สามารถคลิกเพื่อกลับไปยังหน้ารายการ Offices ได้

```
<%--เป็นปุ่มที่สามารถเข้าไปหน้า Form เพื่อ Add office เพิ่มไปได้ --%>
<div class="center">
  <a href="OfficeCreateServlet" style="color: white"> <button type="button" class="btn btn-primary" > Add Office + </button></a>
</div>
```

มีปุ่มเพิ่ม Offices ที่เชื่อมไปยัง OfficeCreateServlet สำหรับการเพิ่ม Office ใหม่



ภาพรวม UI ของ office-list.jsp

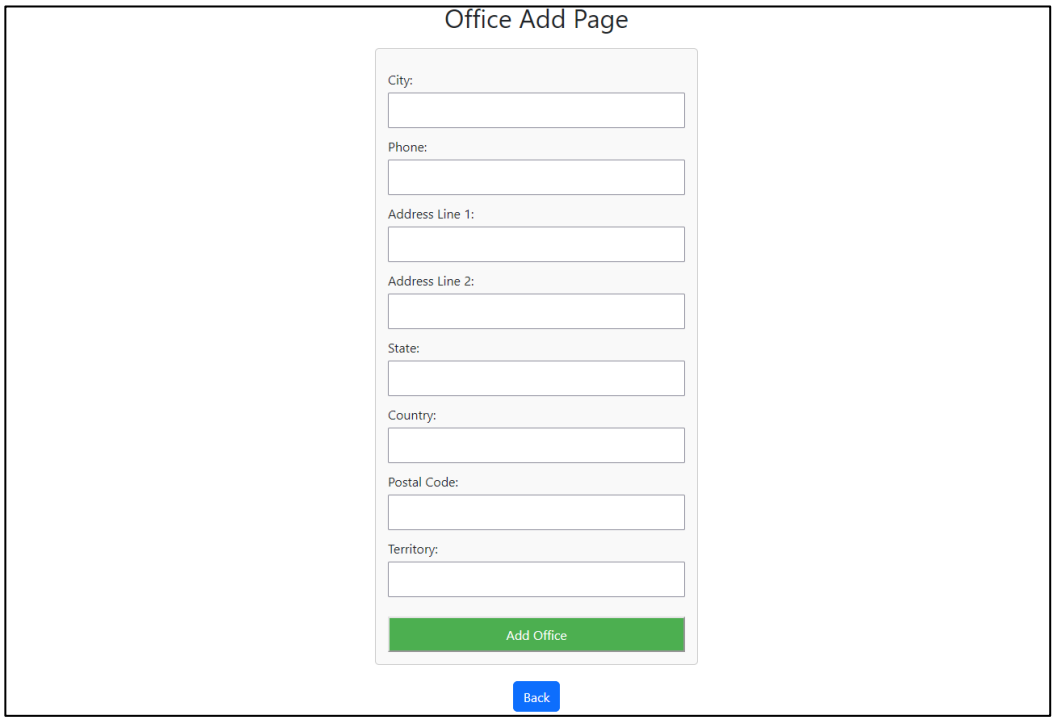
2.3 office-add.jsp

```jsp
<body>
<h2>Office Add Page</h2>

<%-- ฟอร์มสำหรับเพิ่มข้อมูล Office --%>
<form action="OfficeCreateServlet" method="post">
    <label for="city">City:</label>
    <input type="text" id="city" name="city" required><br>

    <label for="phone">Phone:</label>
    <input type="text" id="phone" name="phone" required><br>

    <label for="addressLine1">Address Line 1:</label>
    <input type="text" id="addressLine1" name="addressLine1" required><br>

    <label for="addressLine2">Address Line 2:</label>
    <input type="text" id="addressLine2" name="addressLine2"><br>

    <label for="state">State:</label>
    <input type="text" id="state" name="state"><br>

    <label for="country">Country:</label>
    <input type="text" id="country" name="country" required><br>

    <label for="postalCode">Postal Code:</label>
    <input type="text" id="postalCode" name="postalCode" required><br>

    <label for="territory">Territory:</label>
    <input type="text" id="territory" name="territory" required><br><br>

    <input type="submit" value="Add Office">
</form>
```

ระบุว่าข้อมูลจะถูกส่งไปยัง OfficeCreateServlet ผ่าน POST ในแต่ละ <label> และ <input>: รับข้อมูลจากผู้ใช้ เช่น City, Phone, Address เป็นต้น ปุ่มที่ให้ผู้ใช้คลิกเพื่อทำการส่งข้อมูลไปยัง OfficeCreateServlet เพื่อเพิ่ม Office ใหม่ ใส่ require ไปใน input เพื่อ เพื่อบังคับข้อให้กรอกข้อมูลที่จำเป็นต้องกรอกไม่งั้นไม่สามารถแอดข้อมูลไปได้

```jsp
<%-- แสดงข้อความสำเร็จหลังจากทำการเพิ่ม Office --%>
<c:if test="${not empty requestScope.successMessage}">
    <div class="success-message">${requestScope.successMessage}</div>
</c:if>
<%-- ปุ่ม "Back" เพื่อกลับไปยังหน้ารายการ ListOffices --%>
<div class="center">
    <a href="OfficeListServlet" style="color: white"><button type="button" class="btn btn-primary"> Back </button></a>
</div>
</body>
```

ใช้เพื่อตรวจสอบว่ามีข้อความ successMessage ที่ต้องการแสดงหลังจากการเพิ่ม Office หรือไม่



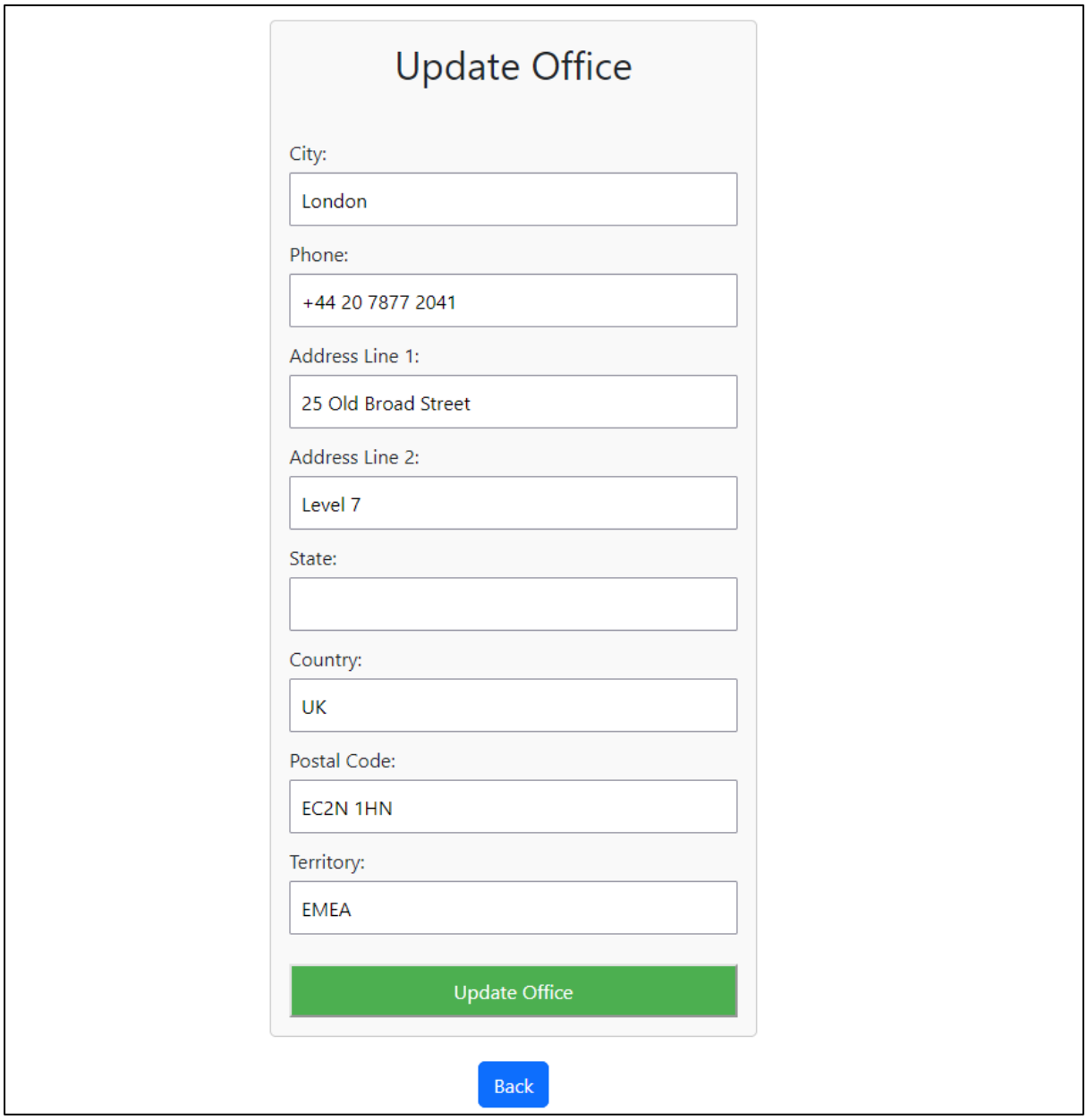
2.4 office-update.jsp

```jsp
<body>
<form action="OfficeUpdateServlet" method="post">
    <h2>Update Office</h2>
    <!-- ใช้ input type="hidden" เพื่อเก็บค่า officeCode และไม่ต้องแสดงให้ผู้ใช้เห็น -->
    <input type="hidden" name="officeCode" required value="${office.officeCode}"><br>
    <label for="city">City:</label>
    <!-- ใช้ value="${office.city}" เพื่อแสดงข้อมูลปัจจุบันของ city -->
    <input type="text" id="city" name="city" required value="${office.city}"><br>
    <label for="phone">Phone:</label>
    <!-- ใช้ value="${office.phoneNumber}" เพื่อแสดงข้อมูลปัจจุบันของ phoneNumber -->
    <input type="text" id="phone" name="phone" required value="${office.phoneNumber}"><br>
    <!-- ต่อไปนี้เช่นเดียวกันกับข้างบน ให้ใส่ value="${office.ชื่อ property}" เพื่อแสดงข้อมูลปัจจุบัน -->
    <label for="addressLine1">Address Line 1:</label>
    <input type="text" id="addressLine1" name="addressLine1" required value="${office.addressLine1}"><br>
    <label for="addressLine2">Address Line 2:</label>
    <input type="text" id="addressLine2" name="addressLine2" value="${office.addressLine2}"><br>
    <label for="state">State:</label>
    <input type="text" id="state" name="state" value="${office.state}"><br>
    <label for="country">Country:</label>
    <input type="text" id="country" name="country" required value="${office.country}"><br>
    <label for="postalCode">Postal Code:</label>
    <input type="text" id="postalCode" name="postalCode" required value="${office.postalCode}"><br>
    <label for="territory">Territory:</label>
    <input type="text" id="territory" name="territory" required value="${office.territory}"><br><br>
    <input type="submit" value="Update Office">
</form>
<!-- แสดงข้อความสำเร็จหลังจากทำการอัปเดต Office -->
<c:if test="${not empty requestScope.successMessage}">
    <div class="success-message">${requestScope.successMessage}</div>
</c:if>
<!-- ปุ่ม "Back" เพื่อกลับไปยังหน้ารายการ Offices -->
<div class="center">
    <a href="OfficeListServlet" style="color: white"><button type="button" class="btn btn-primary"> Back </button></a>
</div>
</body>
```

ใช้ input type="hidden" เพื่อเก็บค่า officeCode และไม่ต้องแสดงให้ผู้ใช้เห็น แต่จะถูกส่งไปยัง OfficeUpdateServlet เพื่อให้รู้ว่าจะต้องอัปเดต Office ที่มี officeCode ที่กำหนด

value="${office.propertyName}": ใช้ attribute value เพื่อกำหนดค่าที่ต้องการแสดงในฟิลด์ input ที่แต่ละตัว เช่น ${office.city}, ${office.phoneNumber} เป็นต้น

<c:if test="${not empty requestScope.successMessage}">: ใช้เพื่อตรวจสอบว่ามีข้อความ successMessage ที่ต้องการแสดงหลังจากการอัปเดต Office หรือไม่



ภาพรวม UI ของ office-update.jsp

3. Servlets มีอยู่ 5 ไฟล์

3.1 OfficeListServlet

```java
@WebServlet(name = "OfficeListServlet", value = "/OfficeListServlet")
public class OfficeListServlet extends HttpServlet {
    no usages  new *
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // ดึง session ที่เกี่ยวข้อง
        HttpSession session = request.getSession();

        // สร้างอ็อบเจ็กต์ OfficeRepository เพื่อดึงข้อมูล Offices จากฐานข้อมูล
        OfficeRepository officeRepository = new OfficeRepository();

        // เก็บข้อมูล Offices ทั้งหมดลงใน session เพื่อนำไปใช้ในหน้า JSP
        session.setAttribute( s: "offices", officeRepository.findAll());

        // ดึง officeCode จาก parameter ของ request
        String officeCode = request.getParameter( s: "officeCode");

        // ถ้ามีการระบุ officeCode, ให้เซ็ต attribute "selectedOffice" เพื่อนำไปใช้ในหน้า JSP
        if (officeCode != null) {
            request.setAttribute( s: "selectedOffice", officeRepository.find(officeCode));
        }

        // ส่ง request ไปยังหน้า JSP (/office-list.jsp) เพื่อแสดงผล
        getServletContext().getRequestDispatcher( s: "/office-list.jsp").forward(request, response);
    }

    no usages  new *
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // ไม่ได้ทำการจัดการในกรณีที่มีการส่ง request แบบ POST
    }
}
```

สร้าง session

สร้างอ็อบเจ็กต์ OfficeRepository เพื่อดึงข้อมูล Offices จากฐานข้อมูล.

เก็บข้อมูล Offices ทั้งหมดลงใน session.

ดึง officeCode จาก parameter ของ request.

ถ้ามีการระบุ officeCode, ให้เซ็ต attribute "selectedOffice" เพื่อนำไปใช้ในหน้า JSP.

ส่ง request ไปยังหน้า JSP (/office-list.jsp) เพื่อแสดงผล.

3.2 OfficeCreateServlet

```java
@WebServlet(name = "OfficeCreateServlet", value = "/OfficeCreateServlet")
public class OfficeCreateServlet extends HttpServlet {
    no usages  new *
    @Override
    // เมื่อมีการเรียกดูหน้า /OfficeCreateServlet ผ่านเมธอด GET
    // จะทำการส่งผู้ใช้ไปยังหน้า JSP ที่ใช้สำหรับการเพิ่มข้อมูล Office
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.getRequestDispatcher( s: "/office-add.jsp").forward(request, response);
    }
    no usages  new *
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // ดึงข้อมูลจากฟอร์ม
        String city = request.getParameter( s: "city");
        String phone = request.getParameter( s: "phone");
        String addressLine1 = request.getParameter( s: "addressLine1");
        String addressLine2 = request.getParameter( s: "addressLine2");
        String state = request.getParameter( s: "state");
        String country = request.getParameter( s: "country");
        String postalCode = request.getParameter( s: "postalCode");
        String territory = request.getParameter( s: "territory");
        // สร้างอ็อบเจ็กต์ Office และกำหนดค่าข้อมูล
        OfficeRepository officeRepository = new OfficeRepository();;
        Office office = new Office();
        String officeCode = officeRepository.getNewOfficeCode();
        office.setOfficeCode(officeCode);
        office.setCity(city);
        office.setPhoneNumber(phone);
        office.setAddressLine1(addressLine1);
        office.setAddressLine2(addressLine2);
        office.setState(state);
        office.setCountry(country);
        office.setPostalCode(postalCode);
        office.setTerritory(territory);
        // เพิ่มข้อมูล Office ลงในฐานข้อมูล
        officeRepository.insert(office);
        // ตั้งค่าข้อความเมื่อ success และส่งผู้ใช้กลับไปยังหน้า JSP ที่ใช้สำหรับการเพิ่มข้อมูล Office
        request.setAttribute( s: "successMessage",  o: "Add office successfully");
        request.getRequestDispatcher( s: "/office-add.jsp").forward(request, response);
    }
}
```

เมื่อมีการเรียกดูหน้า /OfficeCreateServlet ผ่านเมธอด GET จะทำการส่งผู้ใช้ไปยังหน้า JSP ที่ใช้สำหรับการเพิ่มข้อมูล Office (/office-add.jsp). เมื่อมีการส่งข้อมูลมาผ่านเมธอด POST (เช่น ผ่านฟอร์ม), Servlet จะดึงข้อมูลจากฟอร์มและสร้างอ็อบเจ็กต์ Office จากข้อมูลที่ได้รับ. จากนั้นจะทำการเพิ่มข้อมูล Office ลงในฐานข้อมูลผ่าน OfficeRepository.

3.3 OfficeDelServlet

```java
@WebServlet(name = "OfficeDelServlet", value = "/OfficeDelServlet")
public class OfficeDelServlet extends HttpServlet {
    no usages  new *
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // สร้างอ็อบเจ็กต์ OfficeRepository เพื่อจัดการกับข้อมูล Offices
        OfficeRepository officeRepository = new OfficeRepository();

        // ดึง officeCode ที่ต้องการลบจาก parameter ของ request
        String officeCode = request.getParameter( s: "officeCode");

        // เรียกเมธอด delete ของ OfficeRepository เพื่อลบข้อมูล Office
        officeRepository.delete(officeCode);

        // ส่ง request ไปยัง "/OfficeListServlet" เพื่อแสดงรายการ Offices หลังจากที่มีการลบข้อมูล
        request.getRequestDispatcher( s: "/OfficeListServlet").forward(request, response);
    }

    no usages  new *
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

อธิบายได้เขียนไว้ในคอมเม้นของโค้ดเรียบร้อยแล้ว

3.4 OfficeSearchServlet

```java
@WebServlet(name = "OfficeSearchServlet", value = "/OfficeSearchServlet")
public class OfficeSearchServlet extends HttpServlet {
    no usages  new *
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // ไม่ได้ใช้ GET ในที่นี้
    }

    no usages  new *
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // สร้างอ็อบเจ็กต์ OfficeRepository เพื่อจัดการกับข้อมูล Offices
        OfficeRepository officeRepository = new OfficeRepository();

        // ดึงข้อมูลที่ผู้ใช้กรอกในฟอร์มค้นหา
        String searchCountry = request.getParameter( s: "Search");

        // ค้นหา Offices จากฐานข้อมูล โดยใช้เมธอด findByCityOrCountry
        List<Office> foundOffices = officeRepository.findByCityOrCountry(searchCountry);

        // ตรวจสอบว่ามีข้อมูลที่ค้นหาเจอหรือไม่
        if (foundOffices.isEmpty()) {
            // ถ้าไม่พบข้อมูล กำหนด attribute "messageErr" และส่งกลับไปที่หน้า JSP
            request.setAttribute( s: "messageErr",  o: "NOT FOUND DATA");
            request.getRequestDispatcher( s: "/office-list.jsp").forward(request, response);
        }

        // ถ้าพบข้อมูล กำหนด attribute "offices" และส่งกลับไปที่หน้า JSP
        request.setAttribute( s: "offices", foundOffices);
        request.getRequestDispatcher( s: "/office-list.jsp").forward(request, response);
    }
}
```

อธิบายได้เขียนไว้ในคอมเม้นของโค้ดเรียบร้อยแล้ว

3.5 OfficeUpdatesServlet

```java
@WebServlet(name = "OfficeUpdateServlet", value = "/OfficeUpdateServlet")
public class OfficeUpdateServlet extends HttpServlet {
    no usages  new *
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // ดึง officeCode จาก parameter ของ request
        String officeCode = request.getParameter( s: "officeCode");

        // สร้างอ็อบเจ็กต์ OfficeRepository เพื่อจัดการกับข้อมูล Offices
        OfficeRepository officeRepository = new OfficeRepository();

        // ดึงข้อมูล Office ที่ต้องการอัปเดตจากฐานข้อมูล
        Office office = officeRepository.find(officeCode);

        // เซ็ต attribute "office" เพื่อนำไปใช้ในหน้า JSP
        request.setAttribute( s: "office", office);

        // ส่ง request ไปยัง "office-update.jsp" เพื่อแสดงฟอร์มอัปเดต
        request.getRequestDispatcher( s: "office-update.jsp").forward(request, response);
    }
```

```java
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // สร้างอ็อบเจ็กต์ OfficeRepository เพื่อจัดการกับข้อมูล Offices
        OfficeRepository officeRepository = new OfficeRepository();

        // ดึงข้อมูลจากฟอร์มที่ผู้ใช้กรอก
        String city = request.getParameter( s: "city");
        String phone = request.getParameter( s: "phone");
        String addressLine1 = request.getParameter( s: "addressLine1");
        String addressLine2 = request.getParameter( s: "addressLine2");
        String state = request.getParameter( s: "state");
        String country = request.getParameter( s: "country");
        String postalCode = request.getParameter( s: "postalCode");
        String territory = request.getParameter( s: "territory");
        String officeCode = request.getParameter( s: "officeCode");

        // สร้างอ็อบเจ็กต์ Office และกำหนดค่า
        Office office = new Office();
        office.setOfficeCode(officeCode);
        office.setCity(city);
        office.setPhoneNumber(phone);
        office.setAddressLine1(addressLine1);
        office.setAddressLine2(addressLine2);
        office.setState(state);
        office.setCountry(country);
        office.setPostalCode(postalCode);
        office.setTerritory(territory);

        // เรียกเมธอด update ของ OfficeRepository เพื่ออัปเดตข้อมูลในฐานข้อมูล
        officeRepository.update(office);

        // เซ็ต attribute "successMessage" เพื่อแสดงข้อความสำเร็จในหน้า JSP
        request.setAttribute( s: "successMessage",  o: "Update office successfully");

        // ส่ง request ไปยัง "/office-update.jsp" เพื่อแสดงผล
        request.getRequestDispatcher( s: "/office-update.jsp").forward(request, response);
    }
}
```

คอมเม้นได้อธิบายโค้ดไว้เรียบร้อยแล้ว

นอกจาก MVC แล้วยังมีคลาสอื่นที่เป็นเครื่องมือเสริมทาให้การเขียนโค้ดมีประสิทธิภาพมากขึ้น ๆประกอบไปด้วยคลาสมีดังนี้

1. **OfficeRepository**.java เป็นคลาสที่ใช้สำหรับจัดการกับข้อมูลในฐานข้อมูลของ Offices Ex CRUD Operation, อธิบายโค้ดไว้ในส่วนของคอมเม้นเรียบร้อยแล้ว

```java
public class OfficeRepository {
    7 usages
    private EntityManager entityManager;    //สร้างแล้วเก็บไว้ที่นี่

    8 usages  new *
    private EntityManager getEntityManager() {
        if (entityManager == null || !entityManager.isOpen()) {
            entityManager = EntityManagerBuilder.getEntityManager();
        }
        return entityManager;
    }

    1 usage  new *
    public List<Office> findAll() {
        return getEntityManager().createNamedQuery( s: "OFFICE.FIND_ALL").getResultList();
    } // return Offices ทั้งหมดจากฐานข้อมูล.

    4 usages  new *
    public Office find(String officeCode) {
        return getEntityManager().find(Office.class, officeCode);
    } //ค้นหาและ return ข้อมูลของ Office จากฐานข้อมูลโดยใช้ officeCode.

    no usages  new *
    public void close() {   //ปิดการใช้งาน entity manager
        if (entityManager != null && entityManager.isOpen()) {
            entityManager.close();
        }
    }
    //Create => insert
    1 usage  new *
    public boolean insert(Office office) {
        try {
            EntityManager entityManager = getEntityManager();
            entityManager.getTransaction().begin();
            entityManager.persist(office);
            entityManager.getTransaction().commit();
        } catch (Exception e) {
            return false;
        }
        return true;
    }
```

```java
//All Delete
no usages  new *
public boolean delete(Office office) {
    if (office != null) {
        EntityManager entityManager = getEntityManager();
        if (entityManager.contains(office)) {
            entityManager.getTransaction().begin();
            entityManager.remove(office);
            entityManager.getTransaction().commit();
            return true;
        } else {
            return delete(office.getOfficeCode());
        }
    }
    return false;
}
//Delete by Code ลบข้อมูล Office จากฐานข้อมูลโดยใช้ officeCode.
2 usages  new *
public boolean delete(String officeCode) {
    EntityManager entityManager = getEntityManager();
    Office office = find(officeCode);
    if (office != null) {
        entityManager.getTransaction().begin();
        entityManager.remove(office);
        entityManager.getTransaction().commit();
        return true;
    }
    return false;
}
```

```java
public boolean update(Office newOffice) {
    if (newOffice != null) { //newOffice ที่รับเข้ามาไม่ใช่ null
        EntityManager entityManager = getEntityManager();
        Office office = find(newOffice.getOfficeCode());
        if (office != null) {
            entityManager.getTransaction().begin(); // การเปิด Transaction เมื่อได้ Office
            office.setCity(newOffice.getCity());
            office.setPhoneNumber(newOffice.getPhoneNumber());
            office.setAddressLine1(newOffice.getAddressLine1());
            office.setAddressLine2(newOffice.getAddressLine2());
            office.setState(newOffice.getState());
            office.setCountry(newOffice.getCountry());
            office.setPostalCode(newOffice.getPostalCode());
            office.setTerritory(newOffice.getTerritory());
            entityManager.getTransaction().commit();
            return true;
        } // เมทอดจะคืนค่า true เพื่อแสดงว่าการอัปเดตสำเร็จ. ในกรณีที่ไม่พบข้อมูล Office ที่ต้องการอัปเดต, หรือเกิดข้อผิดพลาดในระหว่างการทำ Transaction, เมทอดจะคืนค่า false.
    }
    return false;
}
1 usage  new *
public String getNewOfficeCode(){
    //อยากได้ officeCode ใหม่ ต่อจาก officeCode ที่มีอยู่ในฐานข้อมูล
    //เช่น มี officeCode ที่มีอยู่ 100 แล้ว ต้องการ officeCode ใหม่ ให้ได้ 101
    //เราต้องหา officeCode ที่มีอยู่ แล้ว +1 แล้ว return ออกมา

    Query query = getEntityManager().createNamedQuery( s: "OFFICE.FIND_ALL");
    List<Office> officeList = query.getResultList();
    int max = 0;
    for (Office office : officeList) {
        int officeCode = Integer.parseInt(office.getOfficeCode());
        if (officeCode > max) {
            max = officeCode;
        }
    }
    return String.valueOf( i: max + 1);
}
```

```java
    public List<Office> findByCityOrCountry(String cityOrCountry) {    //  ค้นหา field city or country
        cityOrCountry = cityOrCountry.toLowerCase() + '%';
        Query query = getEntityManager().createNamedQuery( s: "OFFICE.FIND_BY_CITY_OR_COUNTRY");
        query.setParameter( s: "city", cityOrCountry);
        query.setParameter( s: "country", cityOrCountry);
        return query.getResultList();
    }
}
```

2. EntityManagerBuilder.java

```java
1 usage  new *
public class EntityManagerBuilder {
    1 usage
    private final static EntityManagerFactory emf = Persistence.createEntityManagerFactory(Environment.UNIT_NAME);

    1 usage  new *
    public static EntityManager getEntityManager() { return emf.createEntityManager(); }
}
```

EntityManagerBuilder ทำหน้าที่สร้างและจัดการ EntityManager เพื่อให้โปรแกรมสามารถทำงานกับข้อมูลในฐานข้อมูลได้ตามที่กำหนดใน Persistence Unit ที่ตั้งค่าใน persistence.xml

3.Environment,java

```java
2 usages  new *
public class Environment {
    1 usage
    public static final String UNIT_NAME = "classic-models";
}
```

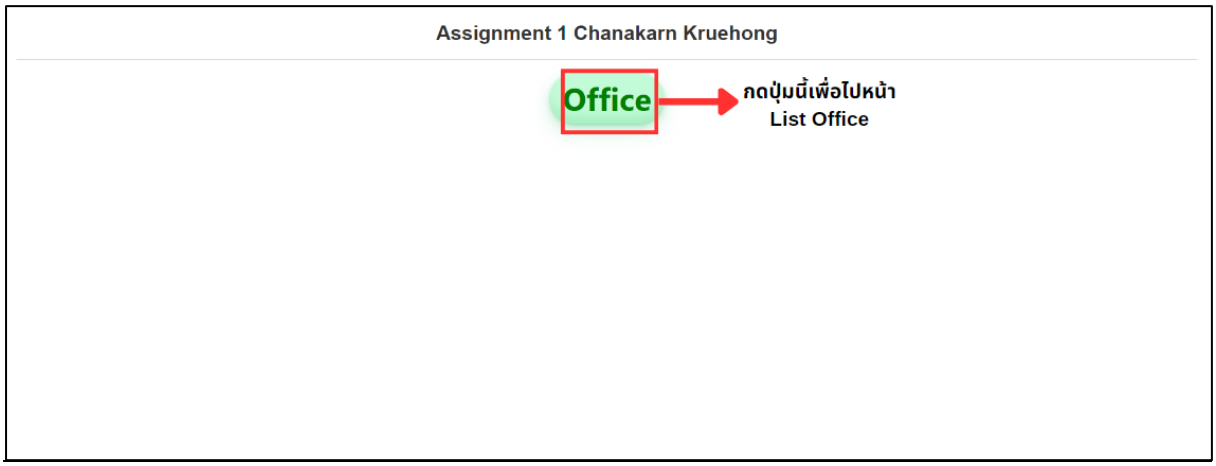
UNIT_NAME ซึ่งมีค่าเป็น "classic-models" นั่นคือชื่อของ Persistence Unit ที่ถูกกำหนดในไฟล์ persistence.xml0

4. persistence.xml ไฟล์การกำหนดค่า **Persistence** ที่ใช้ในโปรเจคของเรา

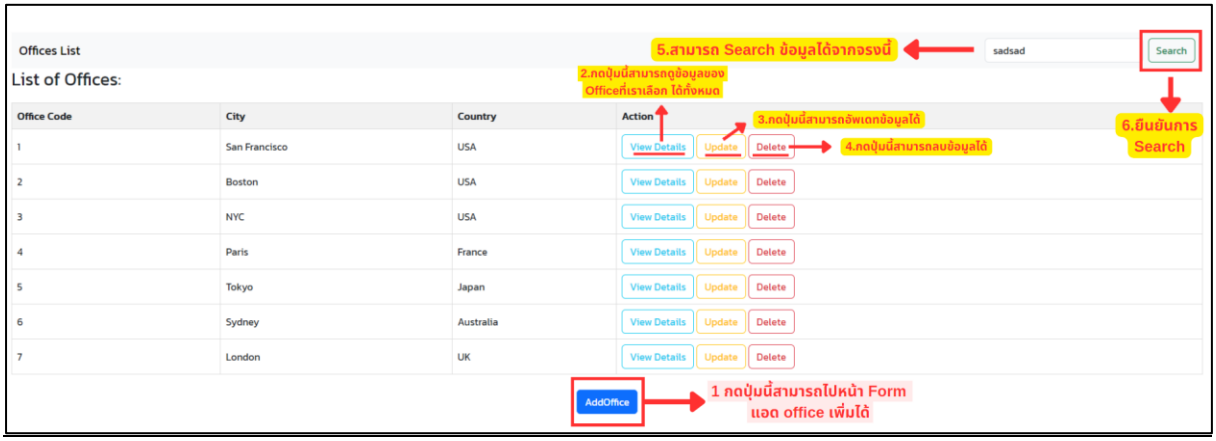
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="https://jakarta.ee/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence https://jakarta.ee/xml/ns/persistence/persistence_3_0.xsd"
             version="3.0">
<!--  กำหนด Persistence Unit ชื่อ "classic-models" ซึ่งเป็นชื่อที่จะใช้ในการอ้างอิง Persistence Unit นี้ในโปรเจค.-->
  <persistence-unit name="classic-models">
<!--    ระบุ JPA Provider ที่จะใช้ ซึ่งในที่นี้คือ Hibernate.-->
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
<!--    ระบุคลาสที่จะถูกสร้างเป็น Entity ใน Persistence Unit นี้, ซึ่งในที่นี้คือคลาส Office ใน package sit.int202.assignment1.Entities-->
    <class>sit.int202.assignment1.Entities.Office</class>
    <properties>
<!--      ระบุ driver ของฐานข้อมูล MySQL.-->
      <property name="jakarta.persistence.jdbc.driver"
                value="com.mysql.cj.jdbc.Driver"/>
<!--      ระบุ URL ของฐานข้อมูล MySQL ที่ต้องการเชื่อมต่อ.-->
      <property name="jakarta.persistence.jdbc.url"
                value="jdbc:mysql://localhost:3306/classicmodels"/>
<!--      ระบุ username สำหรับเข้าสู่ระบบฐานข้อมูล.-->
      <property name="jakarta.persistence.jdbc.user"
                value="root"/>
<!--      ระบุ password สำหรับเข้าสู่ระบบฐานข้อมูล.-->
      <property name="jakarta.persistence.jdbc.password"
                value="b1o2o3k4"/>
<!--      ระบุ dialect ของ Hibernate ที่ใช้กับ MySQL.-->
      <property name="hibernate.dialect"
                value="org.hibernate.dialect.MySQLDialect"/>
    </properties>
  </persistence-unit>
</persistence>
```

<u>อธิบาย Flow การทำของ Project</u>

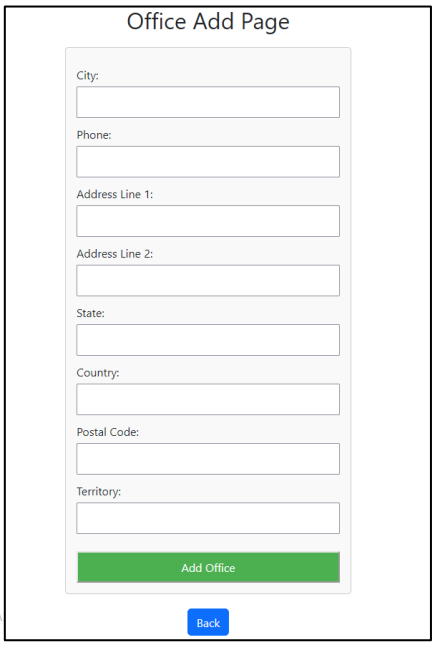
1. หน้าแรกที่ User ต้องเจอ

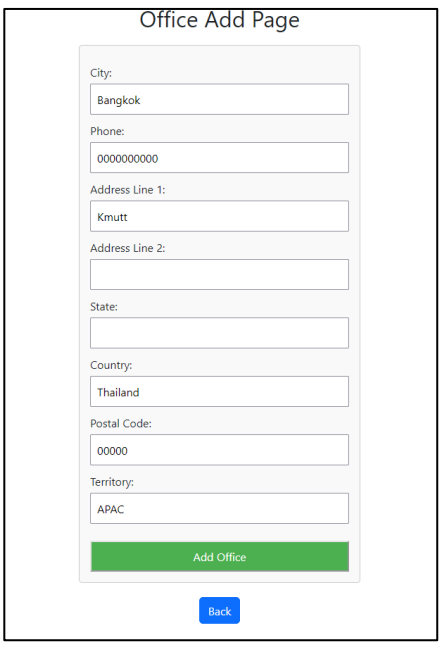


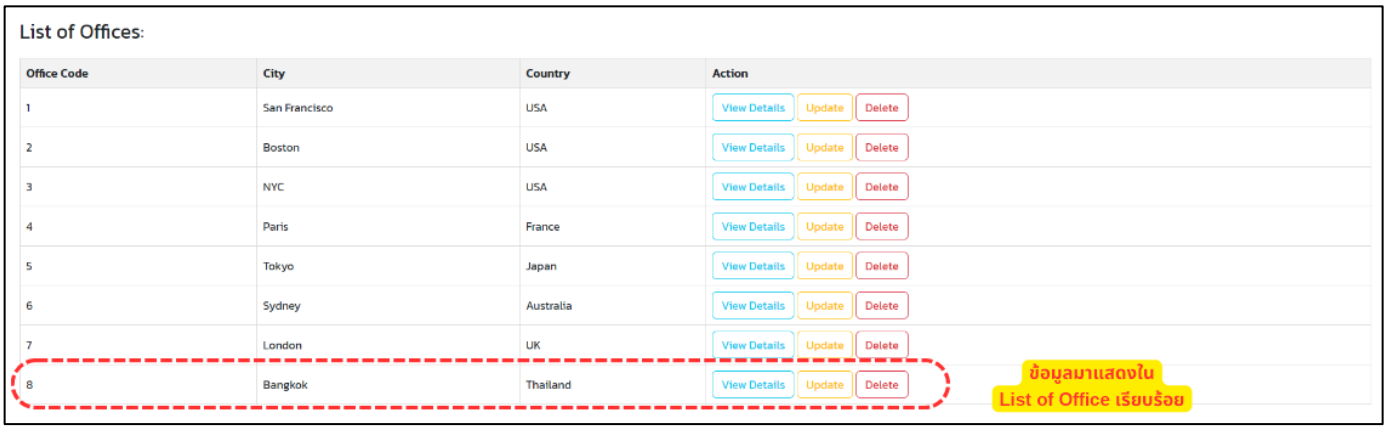
2. หน้า List of Office สามารถทำได้ 6 action



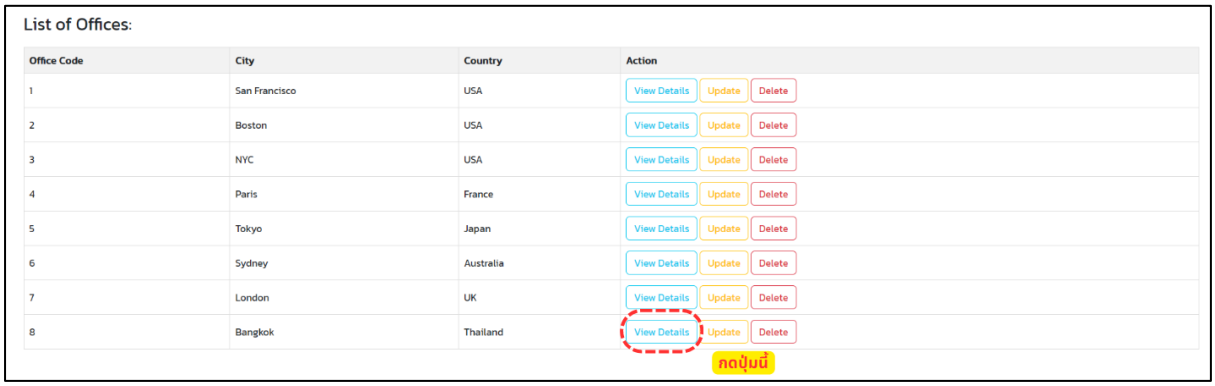
2.1 สามารถกดที่ปุ่ม AddOffice เพื่อไปที่หน้า form ที่สามารถแอด office ตัวใหม่เข้าไปในฐานข้อมูล แล้วกรอกข้อมูล แล้วกด add Office จะมี Message ขึ้นมาว่าแอดสำเร็จ แล้วก็กด Back เพื่อไปดูหน้า List of Offices ได้เลย

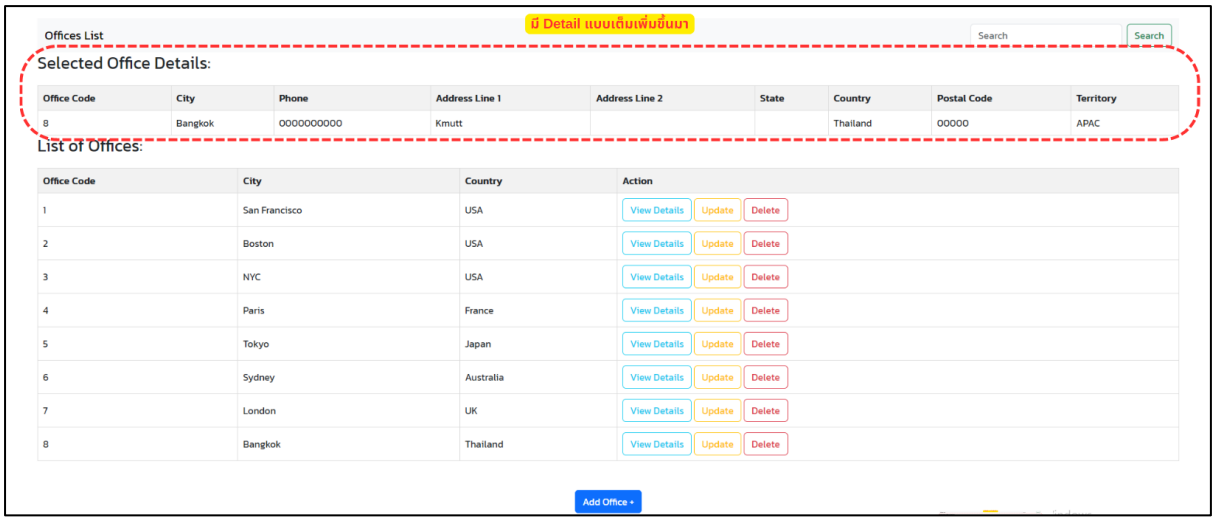




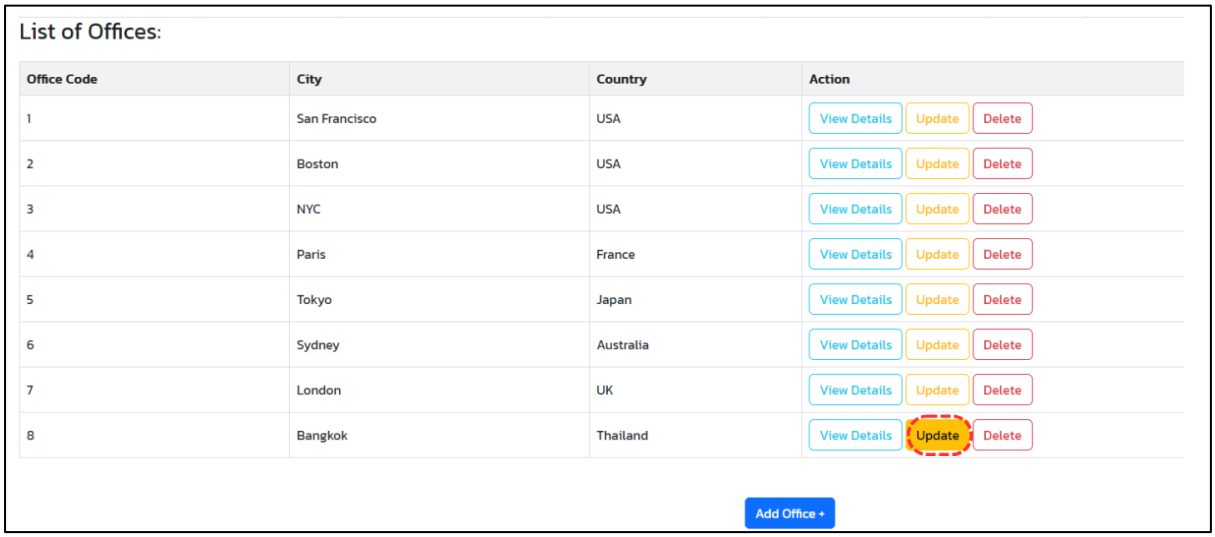


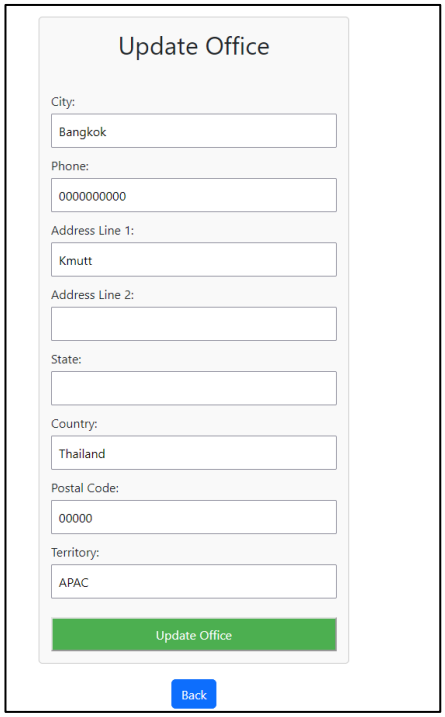
2.2 กดปุ่ม View Details เพื่อดูข้อมูลแบบเต็ม

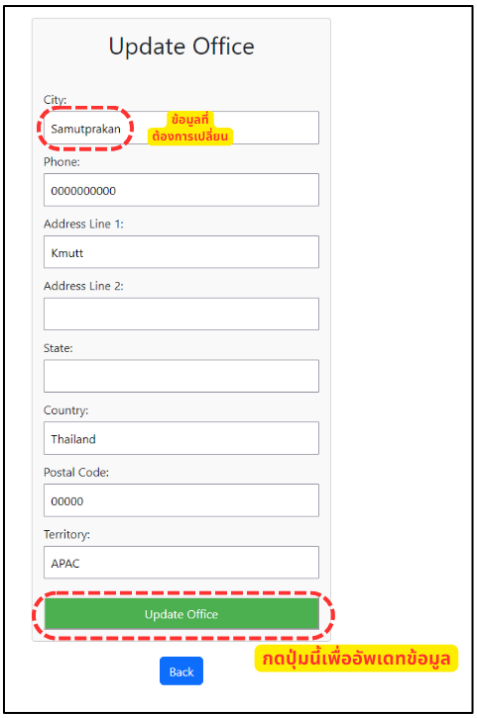


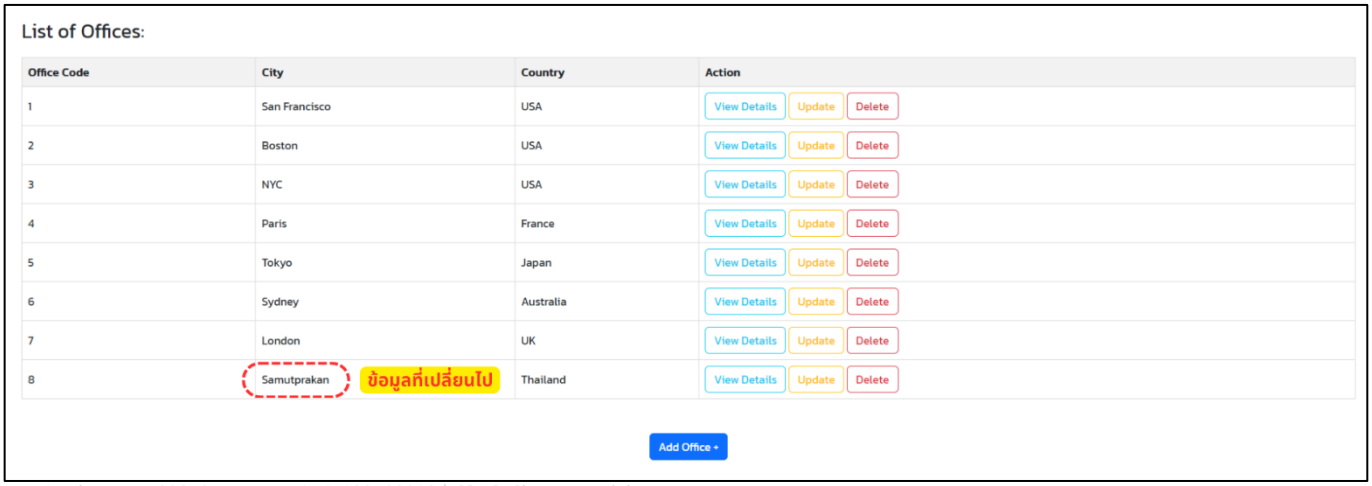


2.3 กดปุ่ม Update Details เพื่อเปลี่ยนข้อมูลที่ต้องการจะ Update เลือกข้อมูลที่ต้องการจะอัพเดทแล้วกดอัพเดทได้เลย จะมีข้อความบอกว่าอัพเดทเสร็จเรียบร้อย แล้วก็กด Back เพื่อไปดูหน้า List of Offices ได้เลย **จะสังเกตเห็นการเปลี่ยนแปลงของข้อมูลที่เรานั้นได้อัพเดทไป**

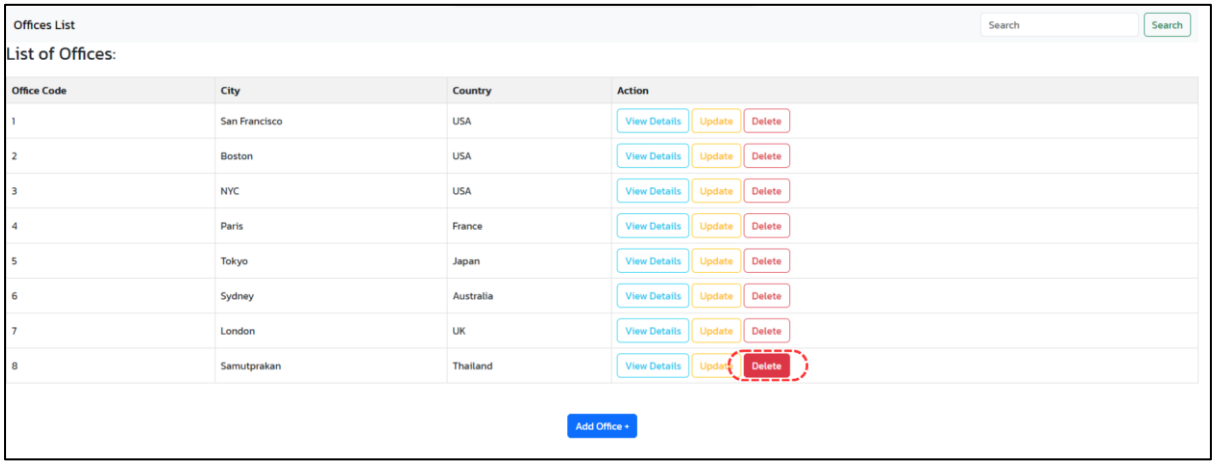


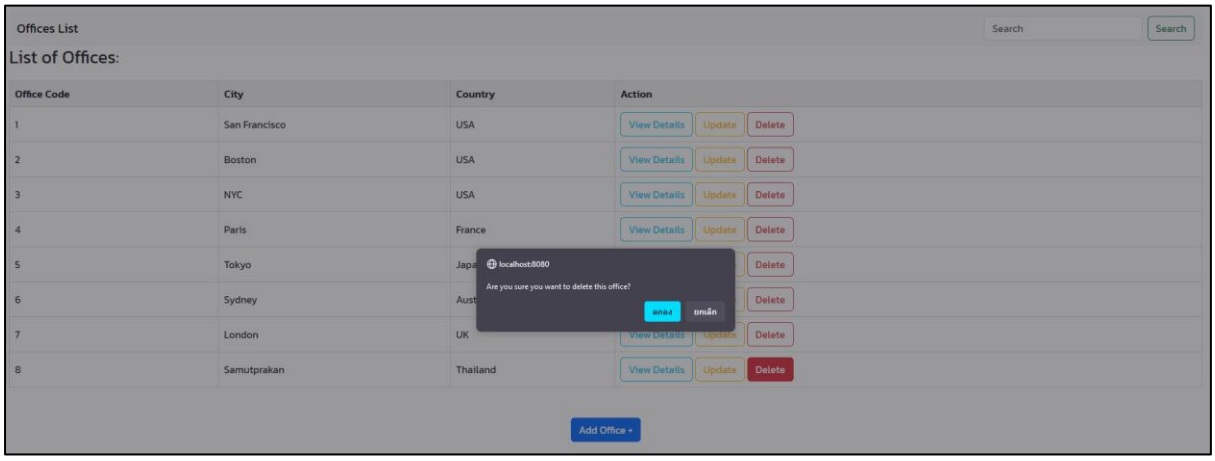


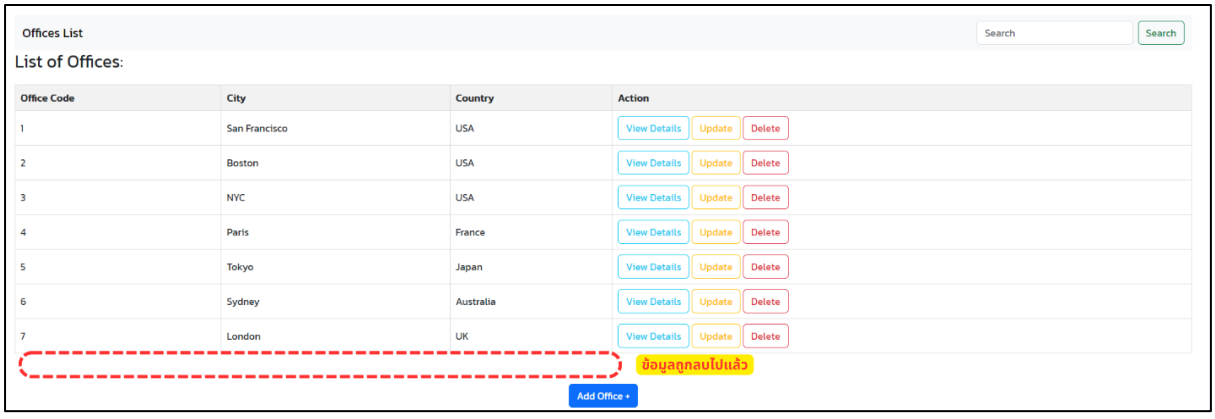




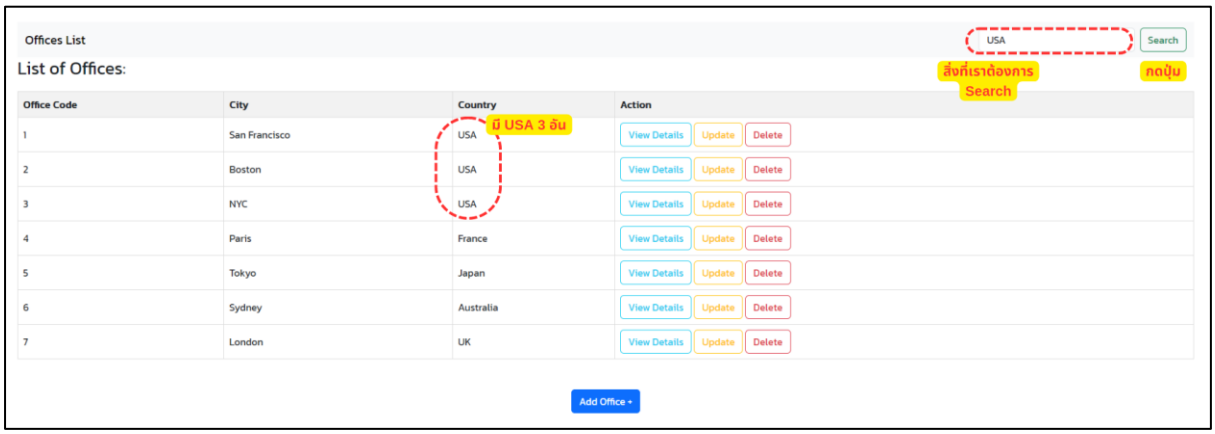
2.4 กดปุ่มนี้สามารถลบข้อมูลของ office Code ที่ต้องการได้ โดยเมื่อกดปุ่มจะมี Alert โดยใช้ JS confirsm มาแจ้งเตือนก่อนลบ เพื่อนกัน User เผลอลบไปโดยบังเอิญ โดยให้ User กดปุ่มตกลงเพื่อยืนยันการลบ แล้วข้อมูลก็จะหายไป

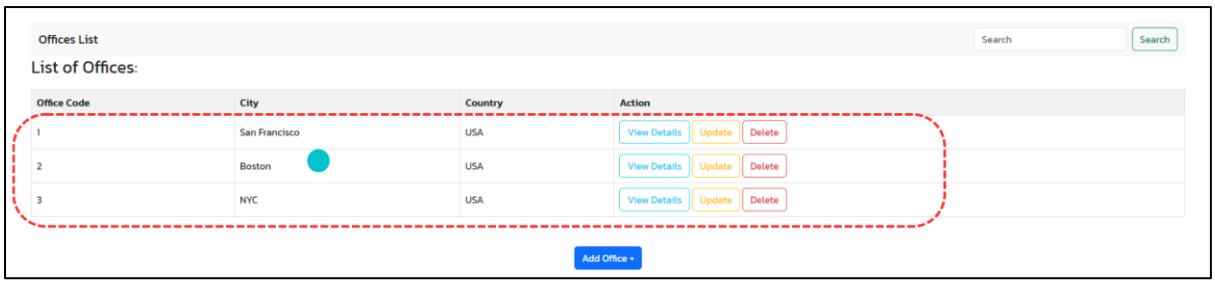




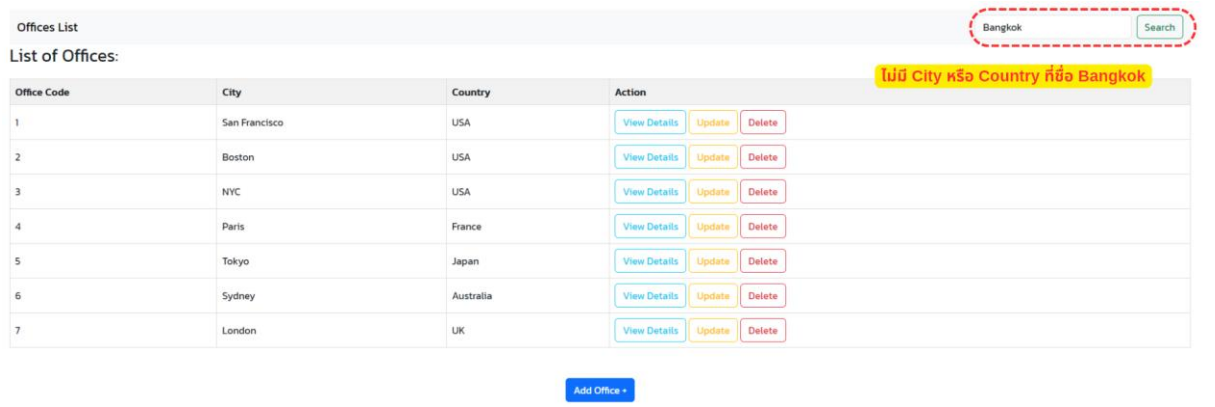


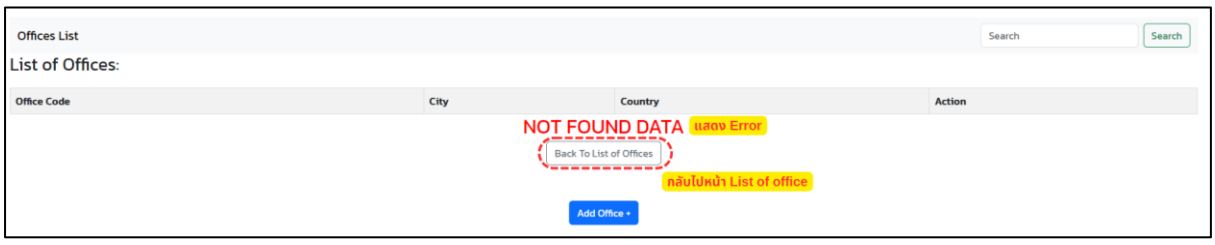
2.5 และ 2.6 ปุ่ม Search จะมีหน้าที่ค้นหาข้อมูลที่ตรงกับชื่อเมืองหรือประเทศที่ถูกส่งเข้ามาในรูปแบบของ Text ที่เราพิมพ์ ถ้าหากตรงก็จะมี List ที่เราค้นหามาให้ แต่ถ้าเกิดว่าไม่ตรงก็จะขึ้น NOT FOUND DATA แล้วมีปุ่มกดให้กลับไปหน้า List of Offices





กรณีหาไม่เจอ





จบการนำเสนอครับ