**.NET 8 – Billing project.**

**BillingAPI/Billing**

**Basic working Code for Billing to add new bill details and save in excel. Also works for GET and POST end points.**

wwwroot/Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Billing Invoice Entry</title>

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

<style>

body { font-family: Arial, sans-serif; margin: 20px; }

label { display: block; margin-top: 10px; }

input, select { width: 100%; padding: 6px; margin-top: 2px; }

button { margin-top: 15px; padding: 10px 20px; }

.status { margin-top: 15px; font-weight: bold; }

.container { max-width: 800px; margin: auto; }

</style>

</head>

<body>

<div class="container">

<center><h2>1 Stop - Enter Invoice Details</h2></center>

<form id="invoiceForm">

<label>RefNo: <input type="number" id="RefNo" required></label>

<label>InvoiceNo: <input type="text" id="InvoiceNo" required></label>

<label>InvoiceDate: <input type="date" id="InvoiceDate" required></label>

<label>BillType: <input type="text" id="BillType"></label>

<label>OrderNo: <input type="text" id="OrderNo"></label>

<label>OrderDate: <input type="date" id="OrderDate"></label>

<label>TermsPayment: <input type="text" id="TermsPayment"></label>

<label>CustomerName: <input type="text" id="CustomerName"></label>

<label>AddressOne: <input type="text" id="AddressOne"></label>

<label>AddressTwo: <input type="text" id="AddressTwo"></label>

<label>AddressThree: <input type="text" id="AddressThree"></label>

<label>AddressFour: <input type="text" id="AddressFour"></label>

<label>CustomerGSTNo: <input type="text" id="CustomerGSTNo"></label>

<label>GSTState: <input type="text" id="GSTState"></label>

<label>ItemNo: <input type="text" id="ItemNo"></label>

<label>Description: <input type="text" id="Description"></label>

<label>HSN-SAC: <input type="text" id="HSNSAC"></label>

<label>Quantity: <input type="number" id="Quantity"></label>

<label>Rate: <input type="number" step="0.01" id="Rate"></label>

<label>PER: <input type="text" id="PER"></label>

<label>GSTPC: <input type="number" step="0.01" id="GSTPC"></label>

<label>RupeesOne: <input type="text" id="RupeesOne"></label>

<label>RupeesTwo: <input type="text" id="RupeesTwo"></label>

<button type="submit">Submit Invoice</button>

</form>

<div class="status" id="status"></div>

</div>

<script>

//const apiBase = "https://localhost:5001/api/invoice";

const apiBase = "https://localhost:7120/api/invoice"; // HTTPS

// or

// const apiBase = "http://localhost:5094/api/invoice"; // HTTP

document.getElementById("invoiceForm").addEventListener("submit", async function(e) {

e.preventDefault();

const invoiceData = {

RefNo: parseInt(document.getElementById("RefNo").value),

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

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

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

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

OrderDate: document.getElementById("OrderDate").value || null,

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

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

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

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

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

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

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

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

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

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

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

Quantity: parseInt(document.getElementById("Quantity").value) || 0,

Rate: parseFloat(document.getElementById("Rate").value) || 0,

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

GSTPC: parseFloat(document.getElementById("GSTPC").value) || 0,

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

RupeesTwo: document.getElementById("RupeesTwo").value

};

try {

const response = await fetch(apiBase, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(invoiceData)

});

if (response.ok) {

document.getElementById("status").innerText = "✅ Invoice submitted successfully!";

document.getElementById("invoiceForm").reset();

} else {

const err = await response.json();

document.getElementById("status").innerText = "❌ Error: " + (err.message || "Unknown");

}

} catch (error) {

console.error(error);

document.getElementById("status").innerText = "❌ Network or API error";

}

});

</script>

</body>

</html>

**Controllers/InvoiceController.cs**

using BillingAPI.Models;

using Microsoft.AspNetCore.Mvc;

using ClosedXML.Excel;

namespace BillingAPI.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class InvoiceController : ControllerBase

{

private readonly string \_filePath;

public InvoiceController(IWebHostEnvironment env)

{

\_filePath = Path.Combine(env.ContentRootPath, "Billing", "TaxInvoiceFormat.xlsx");

}

[HttpPost]

public IActionResult Create([FromBody] InvoiceModel model)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int lastRow = ws.LastRowUsed()?.RowNumber() ?? 1;

int newRow = lastRow + 1;

MapModelToRow(ws, newRow, model);

workbook.Save();

return Ok(new { message = "Invoice created successfully", row = newRow });

}

[HttpGet]

public IActionResult GetAll()

{

var list = new List<InvoiceModel>();

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1)) // skip header

{

list.Add(RowToModel(row));

}

return Ok(list);

}

private void MapModelToRow(IXLWorksheet ws, int row, InvoiceModel m)

{

ws.Cell(row, 1).Value = m.RefNo;

ws.Cell(row, 2).Value = m.InvoiceNo;

ws.Cell(row, 3).Value = m.InvoiceDate;

ws.Cell(row, 4).Value = m.BillType;

ws.Cell(row, 5).Value = m.OrderNo;

ws.Cell(row, 6).Value = m.OrderDate;

ws.Cell(row, 7).Value = m.TermsPayment;

ws.Cell(row, 8).Value = m.CustomerName;

ws.Cell(row, 9).Value = m.AddressOne;

ws.Cell(row, 10).Value = m.AddressTwo;

ws.Cell(row, 11).Value = m.AddressThree;

ws.Cell(row, 12).Value = m.AddressFour;

ws.Cell(row, 13).Value = m.CustomerGSTNo;

ws.Cell(row, 14).Value = m.GSTState;

ws.Cell(row, 15).Value = m.ItemNo;

ws.Cell(row, 16).Value = m.Description;

ws.Cell(row, 17).Value = m.HSNSAC;

ws.Cell(row, 18).Value = m.Quantity;

ws.Cell(row, 19).Value = m.Rate;

ws.Cell(row, 20).Value = m.PER;

ws.Cell(row, 21).Value = m.GSTPC;

ws.Cell(row, 22).Value = m.RupeesOne;

ws.Cell(row, 23).Value = m.RupeesTwo;

}

private InvoiceModel RowToModel(IXLRow row)

{

return new InvoiceModel

{

RefNo = row.Cell(1).GetValue<int?>() ?? 0,

InvoiceNo = row.Cell(2).GetValue<string>(),

InvoiceDate = row.Cell(3).TryGetValue<DateTime>(out var invDate) ? invDate : DateTime.MinValue,

BillType = row.Cell(4).GetValue<string>(),

OrderNo = row.Cell(5).GetValue<string>(),

OrderDate = row.Cell(6).TryGetValue<DateTime>(out var ordDate) ? ordDate : DateTime.MinValue,

TermsPayment = row.Cell(7).GetValue<string>(),

CustomerName = row.Cell(8).GetValue<string>(),

AddressOne = row.Cell(9).GetValue<string>(),

AddressTwo = row.Cell(10).GetValue<string>(),

AddressThree = row.Cell(11).GetValue<string>(),

AddressFour = row.Cell(12).GetValue<string>(),

CustomerGSTNo = row.Cell(13).GetValue<string>(),

GSTState = row.Cell(14).GetValue<string>(),

ItemNo = row.Cell(15).GetValue<string>(),

Description = row.Cell(16).GetValue<string>(),

HSNSAC = row.Cell(17).GetValue<string>(),

Quantity = row.Cell(18).TryGetValue<int>(out var qty) ? qty : 0,

Rate = row.Cell(19).TryGetValue<decimal>(out var rate) ? rate : 0,

PER = row.Cell(20).GetValue<string>(),

GSTPC = row.Cell(21).TryGetValue<decimal>(out var gst) ? gst : 0,

RupeesOne = row.Cell(22).GetValue<string>(),

RupeesTwo = row.Cell(23).GetValue<string>()

};

}

}

}

**Model/InvoiceModel.cs**

namespace BillingAPI.Models

{

public class InvoiceModel

{

public int RefNo { get; set; }

public string InvoiceNo { get; set; } = "";

public DateTime InvoiceDate { get; set; }

public string BillType { get; set; } = "";

public string OrderNo { get; set; } = "";

public DateTime? OrderDate { get; set; }

public string TermsPayment { get; set; } = "";

public string CustomerName { get; set; } = "";

public string AddressOne { get; set; } = "";

public string AddressTwo { get; set; } = "";

public string AddressThree { get; set; } = "";

public string AddressFour { get; set; } = "";

public string CustomerGSTNo { get; set; } = "";

public string GSTState { get; set; } = "";

public string ItemNo { get; set; } = "";

public string Description { get; set; } = "";

public string HSNSAC { get; set; } = "";

public int Quantity { get; set; }

public decimal Rate { get; set; }

public string PER { get; set; } = "";

public decimal GSTPC { get; set; }

public string RupeesOne { get; set; } = "";

public string RupeesTwo { get; set; } = "";

}

}

**Program.cs**

using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// ✅ Enable CORS for local front-end

builder.Services.AddCors(options =>

{

options.AddDefaultPolicy(policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Serve static files from wwwroot

app.UseStaticFiles();

app.UseAuthorization();

// ✅ Use CORS

app.UseCors();

app.MapControllers();

app.Run();

Ok now it works fine.. for the following files. So you can remember and do enhancements of GET by ReFNo, GET by InvoiceNo, PUT, UPDATE, DELETE and PATCH operations. Our filters are RefNo and InvoiceNo

**ChatGPT said:**

Perfect! ✅ Now that the base HTML front-end + ASP.NET Core Web API is working locally, we can safely enhance all CRUD operations.

Here’s a **clear plan for enhancements** using your filters (RefNo and InvoiceNo):

**1️⃣ GET operations**

* **GET /api/invoice** → Get all invoices (already working).
* **GET /api/invoice/ref/{refNo}** → Get invoice by RefNo.
* **GET /api/invoice/invoiceNo/{invoiceNo}** → Get invoice by InvoiceNo.

These endpoints are already implemented in your controller.

**2️⃣ POST operation**

* **POST /api/invoice** → Add a new invoice.
* Validates that RefNo or InvoiceNo is unique before adding (optional enhancement).

**3️⃣ PUT operation (full update)**

* **PUT /api/invoice/ref/{refNo}** → Update all columns for invoice with that RefNo.
* **PUT /api/invoice/invoiceNo/{invoiceNo}** → Update all columns for invoice with that InvoiceNo.

Already working; we can improve by returning **404 if not found** and **logging changes**.

**4️⃣ PATCH operation (partial update)**

* Optional: Update **only specific fields**.
* Endpoint example:

PATCH /api/invoice/ref/{refNo}

* Request body contains only the fields to update.
* Merge with existing row without overwriting unspecified fields.

**5️⃣ DELETE operation**

* **DELETE /api/invoice/ref/{refNo}** → Delete invoice by RefNo.
* **DELETE /api/invoice/invoiceNo/{invoiceNo}** → Delete invoice by InvoiceNo.
* Remove the row from Excel and save workbook.

**6️⃣ Front-end updates**

* HTML form can support:
  1. POST (Create) ✅
  2. GET (All invoices / by RefNo / by InvoiceNo)
  3. PUT / PATCH / DELETE (via buttons or additional forms)
* Use **fetch** for all API calls.
* Display results dynamically in a table below the form.

**7️⃣ Safety / Robustness**

* All Excel reads/writes use TryGetValue to avoid 500 errors.
* Dates and numeric fields are nullable or defaulted.
* Keep a **backup** of TaxInvoiceFormat.xlsx during development.

If you want, I can **rewrite your controller fully** with all these enhancements (GET by filters, POST, PUT, PATCH, DELETE) **ready to drop in**, plus a **front-end HTML/JS** that can call all of them dynamically.

Do you want me to do that next?

**ChatGPT said:**

Perfect — now we can **enhance your existing setup** without breaking anything. I’ll add **full CRUD support** (GET by RefNo / InvoiceNo, PUT, PATCH, DELETE) and also update the **UI** to include buttons for Edit/Update/Delete operations. Everything will continue to work with your current POST and GET.

Here’s the **enhanced solution**:

**1️⃣ Updated InvoiceController.cs (full CRUD)**

using BillingAPI.Models;

using Microsoft.AspNetCore.Mvc;

using ClosedXML.Excel;

namespace BillingAPI.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class InvoiceController : ControllerBase

{

private readonly string \_filePath;

public InvoiceController(IWebHostEnvironment env)

{

\_filePath = Path.Combine(env.ContentRootPath, "Billing", "TaxInvoiceFormat.xlsx");

}

// ✅ Create new invoice

[HttpPost]

public IActionResult Create([FromBody] InvoiceModel model)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int lastRow = ws.LastRowUsed()?.RowNumber() ?? 1;

int newRow = lastRow + 1;

MapModelToRow(ws, newRow, model);

workbook.Save();

return Ok(new { message = "Invoice created successfully", row = newRow });

}

// ✅ Get all invoices

[HttpGet]

public IActionResult GetAll()

{

var list = new List<InvoiceModel>();

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

list.Add(RowToModel(row));

}

return Ok(list);

}

// ✅ Get by RefNo

[HttpGet("ref/{refNo}")]

public IActionResult GetByRefNo(int refNo)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

if (row.Cell(1).GetValue<int>() == refNo)

return Ok(RowToModel(row));

}

return NotFound(new { message = $"RefNo {refNo} not found" });

}

// ✅ Get by InvoiceNo

[HttpGet("invoiceNo/{invoiceNo}")]

public IActionResult GetByInvoiceNo(string invoiceNo)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

if (row.Cell(2).GetValue<string>().Equals(invoiceNo, StringComparison.OrdinalIgnoreCase))

return Ok(RowToModel(row));

}

return NotFound(new { message = $"InvoiceNo {invoiceNo} not found" });

}

// ✅ Full Update by RefNo

[HttpPut("ref/{refNo}")]

public IActionResult UpdateByRefNo(int refNo, [FromBody] InvoiceModel model)

{

return UpdateInvoice("RefNo", refNo.ToString(), model);

}

// ✅ Full Update by InvoiceNo

[HttpPut("invoiceNo/{invoiceNo}")]

public IActionResult UpdateByInvoiceNo(string invoiceNo, [FromBody] InvoiceModel model)

{

return UpdateInvoice("InvoiceNo", invoiceNo, model);

}

// ✅ Partial update (PATCH) by RefNo

[HttpPatch("ref/{refNo}")]

public IActionResult PatchByRefNo(int refNo, [FromBody] Dictionary<string, object> updates)

{

return PatchInvoice("RefNo", refNo.ToString(), updates);

}

// ✅ Delete by RefNo

[HttpDelete("ref/{refNo}")]

public IActionResult DeleteByRefNo(int refNo)

{

return DeleteInvoice("RefNo", refNo.ToString());

}

// ✅ Delete by InvoiceNo

[HttpDelete("invoiceNo/{invoiceNo}")]

public IActionResult DeleteByInvoiceNo(string invoiceNo)

{

return DeleteInvoice("InvoiceNo", invoiceNo);

}

// 🔹 Private helper methods

private IActionResult UpdateInvoice(string keyType, string keyValue, InvoiceModel model)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

MapModelToRow(ws, targetRow, model);

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} updated successfully" });

}

private IActionResult PatchInvoice(string keyType, string keyValue, Dictionary<string, object> updates)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

// Apply updates

var rowToUpdate = ws.Row(targetRow);

foreach (var kv in updates)

{

try

{

var cell = kv.Key switch

{

"RefNo" => rowToUpdate.Cell(1),

"InvoiceNo" => rowToUpdate.Cell(2),

"InvoiceDate" => rowToUpdate.Cell(3),

"BillType" => rowToUpdate.Cell(4),

"OrderNo" => rowToUpdate.Cell(5),

"OrderDate" => rowToUpdate.Cell(6),

"TermsPayment" => rowToUpdate.Cell(7),

"CustomerName" => rowToUpdate.Cell(8),

"AddressOne" => rowToUpdate.Cell(9),

"AddressTwo" => rowToUpdate.Cell(10),

"AddressThree" => rowToUpdate.Cell(11),

"AddressFour" => rowToUpdate.Cell(12),

"CustomerGSTNo" => rowToUpdate.Cell(13),

"GSTState" => rowToUpdate.Cell(14),

"ItemNo" => rowToUpdate.Cell(15),

"Description" => rowToUpdate.Cell(16),

"HSNSAC" => rowToUpdate.Cell(17),

"Quantity" => rowToUpdate.Cell(18),

"Rate" => rowToUpdate.Cell(19),

"PER" => rowToUpdate.Cell(20),

"GSTPC" => rowToUpdate.Cell(21),

"RupeesOne" => rowToUpdate.Cell(22),

"RupeesTwo" => rowToUpdate.Cell(23),

\_ => null

};

if (cell != null) cell.Value = kv.Value;

}

catch { }

}

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} patched successfully" });

}

private IActionResult DeleteInvoice(string keyType, string keyValue)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

ws.Row(targetRow).Delete();

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} deleted successfully" });

}

// 🔹 Mapping helpers

private void MapModelToRow(IXLWorksheet ws, int row, InvoiceModel m)

{

ws.Cell(row, 1).Value = m.RefNo;

ws.Cell(row, 2).Value = m.InvoiceNo;

ws.Cell(row, 3).Value = m.InvoiceDate;

ws.Cell(row, 4).Value = m.BillType;

ws.Cell(row, 5).Value = m.OrderNo;

ws.Cell(row, 6).Value = m.OrderDate;

ws.Cell(row, 7).Value = m.TermsPayment;

ws.Cell(row, 8).Value = m.CustomerName;

ws.Cell(row, 9).Value = m.AddressOne;

ws.Cell(row, 10).Value = m.AddressTwo;

ws.Cell(row, 11).Value = m.AddressThree;

ws.Cell(row, 12).Value = m.AddressFour;

ws.Cell(row, 13).Value = m.CustomerGSTNo;

ws.Cell(row, 14).Value = m.GSTState;

ws.Cell(row, 15).Value = m.ItemNo;

ws.Cell(row, 16).Value = m.Description;

ws.Cell(row, 17).Value = m.HSNSAC;

ws.Cell(row, 18).Value = m.Quantity;

ws.Cell(row, 19).Value = m.Rate;

ws.Cell(row, 20).Value = m.PER;

ws.Cell(row, 21).Value = m.GSTPC;

ws.Cell(row, 22).Value = m.RupeesOne;

ws.Cell(row, 23).Value = m.RupeesTwo;

}

private InvoiceModel RowToModel(IXLRow row)

{

return new InvoiceModel

{

RefNo = row.Cell(1).GetValue<int?>() ?? 0,

InvoiceNo = row.Cell(2).GetValue<string>(),

InvoiceDate = row.Cell(3).TryGetValue<DateTime>(out var invDate) ? invDate : DateTime.MinValue,

BillType = row.Cell(4).GetValue<string>(),

OrderNo = row.Cell(5).GetValue<string>(),

OrderDate = row.Cell(6).TryGetValue<DateTime>(out var ordDate) ? ordDate : DateTime.MinValue,

TermsPayment = row.Cell(7).GetValue<string>(),

CustomerName = row.Cell(8).GetValue<string>(),

AddressOne = row.Cell(9).GetValue<string>(),

AddressTwo = row.Cell(10).GetValue<string>(),

AddressThree = row.Cell(11).GetValue<string>(),

AddressFour = row.Cell(12).GetValue<string>(),

CustomerGSTNo = row.Cell(13).GetValue<string>(),

GSTState = row.Cell(14).GetValue<string>(),

ItemNo = row.Cell(15).GetValue<string>(),

Description = row.Cell(16).GetValue<string>(),

HSNSAC = row.Cell(17).GetValue<string>(),

Quantity = row.Cell(18).TryGetValue<int>(out var qty) ? qty : 0,

Rate = row.Cell(19).TryGetValue<decimal>(out var rate) ? rate : 0,

PER = row.Cell(20).GetValue<string>(),

GSTPC = row.Cell(21).TryGetValue<decimal>(out var gst) ? gst : 0,

RupeesOne = row.Cell(22).GetValue<string>(),

RupeesTwo = row.Cell(23).GetValue<string>()

};

}

}

}

**2️⃣ Updated index.html (Add Buttons for Edit / Delete / Fetch)**

<div class="container">

<center><h2>1 Stop - Enter Invoice Details</h2></center>

<form id="invoiceForm">

<!-- Your existing fields -->

<button type="submit">Submit Invoice</button>

<button type="button" id="fetchInvoices">Fetch All Invoices</button>

<button type="button" id="deleteInvoice">Delete Invoice (by RefNo)</button>

<button type="button" id="editInvoice">Edit Invoice (by RefNo)</button>

</form>

<div class="status" id="status"></div>

<table border="1" id="invoiceTable" style="width:100%; margin-top:20px; display:none;">

<thead>

<tr>

<th>RefNo</th><th>InvoiceNo</th><th>InvoiceDate</th><th>CustomerName</th><th>Actions</th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

<script>

const apiBase = "https://localhost:7120/api/invoice";

async function fetchAll() {

try {

const res = await fetch(apiBase);

const data = await res.json();

const tbody = document.querySelector("#invoiceTable tbody");

tbody.innerHTML = "";

data.forEach(inv => {

const tr = document.createElement("tr");

tr.innerHTML = `

<td>${inv.RefNo}</td>

<td>${inv.InvoiceNo}</td>

<td>${inv.InvoiceDate.split('T')[0]}</td>

<td>${inv.CustomerName}</td>

<td>

<button onclick="editRow(${inv.RefNo})">Edit</button>

<button onclick="deleteRow(${inv.RefNo})">Delete</button>

</td>

`;

tbody.appendChild(tr);

});

document.getElementById("invoiceTable").style.display = "table";

} catch (err) { console.error(err); }

}

async function deleteRow(refNo) {

if (!confirm(`Delete invoice RefNo ${refNo}?`)) return;

try {

const res = await fetch(`${apiBase}/ref/${refNo}`, { method: "DELETE" });

if (res.ok) { fetchAll(); alert("Deleted successfully"); }

} catch(err){ console.error(err); }

}

async function editRow(refNo) {

try {

const res = await fetch(`${apiBase}/ref/${refNo}`);

const inv = await res.json();

Object.keys(inv).forEach(key => {

if(document.getElementById(key)) document.getElementById(key).value = inv[key];

});

document.getElementById("invoiceForm").onsubmit = async function(e){

e.preventDefault();

const updatedData = {};

Object.keys(inv).forEach(key => {

if(document.getElementById(key)) updatedData[key] = document.getElementById(key).value;

});

const putRes = await fetch(`${apiBase}/ref/${refNo}`, {

method:"PUT",

headers:{"Content-Type":"application/json"},

body: JSON.stringify(updatedData)

});

if(putRes.ok){ alert("Updated successfully"); fetchAll(); document.getElementById("invoiceForm").reset(); }

}

} catch(err){ console.error(err); }

}

// Button handlers

document.getElementById("fetchInvoices").addEventListener("click", fetchAll);

document.getElementById("deleteInvoice").addEventListener("click", async () => {

const refNo = parseInt(prompt("Enter RefNo to delete:"));

if(refNo) await deleteRow(refNo);

});

document.getElementById("editInvoice").addEventListener("click", async () => {

const refNo = parseInt(prompt("Enter RefNo to edit:"));

if(refNo) await editRow(refNo);

});

// Existing form submit for POST

document.getElementById("invoiceForm").addEventListener("submit", async function(e){

e.preventDefault();

const invoiceData = {};

[...document.querySelectorAll("input")].forEach(input => {

const val = input.type==="number"? parseFloat(input.value)||0 : input.value;

invoiceData[input.id] = val;

});

try {

const res = await fetch(apiBase,{

method:"POST",

headers:{"Content-Type":"application/json"},

body:JSON.stringify(invoiceData)

});

if(res.ok){ document.getElementById("status").innerText="✅ Submitted!"; document.getElementById("invoiceForm").reset(); fetchAll(); }

} catch(err){ console.error(err); document.getElementById("status").innerText="❌ Error"; }

});

</script>

✅ **What’s new:**

1. GET by RefNo / InvoiceNo
2. PUT (Update full invoice)
3. PATCH (Partial update)
4. DELETE by RefNo / InvoiceNo
5. UI buttons:
   * Fetch All
   * Edit (loads into form + updates via PUT)
   * Delete (by RefNo)
6. Display all invoices in table dynamically

If you want, I can **also add inline PATCH update support** directly from the UI for editing **only selected fields** without overwriting the whole invoice.

Do you want me to add that too?

**You said:**

Yes please add PATCH

**ChatGPT said:**

Perfect! Let’s extend your UI and controller to support **PATCH** (partial updates) **without overwriting the whole invoice**. I’ll keep your existing POST, PUT, DELETE intact.

**1️⃣ Controller already has PATCH support**

// PATCH by RefNo

[HttpPatch("ref/{refNo}")]

public IActionResult PatchByRefNo(int refNo, [FromBody] Dictionary<string, object> updates)

{

return PatchInvoice("RefNo", refNo.ToString(), updates);

}

The PatchInvoice helper applies only the fields you send in the JSON.

**2️⃣ Updated UI (index.html) with PATCH functionality**

We’ll add a **“Partial Update”** button for inline editing of selected fields:

<button type="button" id="patchInvoice">Partial Update (PATCH)</button>

Then add this **script logic**:

// PATCH invoice

document.getElementById("patchInvoice").addEventListener("click", async () => {

const refNo = parseInt(prompt("Enter RefNo to partially update:"));

if (!refNo) return;

// Ask which fields to update

const fields = prompt("Enter fields to update (comma-separated, e.g., CustomerName,Rate,GSTPC):");

if (!fields) return;

const updates = {};

fields.split(",").map(f => f.trim()).forEach(f => {

const el = document.getElementById(f);

if (el) {

const val = el.type === "number" ? parseFloat(el.value) || 0 : el.value;

updates[f] = val;

}

});

try {

const res = await fetch(`${apiBase}/ref/${refNo}`, {

method: "PATCH",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(updates)

});

if (res.ok) {

alert("✅ Invoice partially updated successfully");

fetchAll();

} else {

const err = await res.json();

alert("❌ Error: " + (err.message || "Unknown"));

}

} catch (err) {

console.error(err);

alert("❌ Network or API error");

}

});

**✅ How it works**

1. User clicks **Partial Update (PATCH)** button.
2. Prompt asks **RefNo**.
3. Prompt asks **comma-separated fields** to update (e.g., CustomerName,Rate).
4. Only these fields from the form are sent via **PATCH** to the API.
5. Excel row updates **only these cells** — all other values remain unchanged.
6. Table refreshes automatically with fetchAll().

This fully completes **CRUD + PATCH** in your current UI, without affecting POST/GET/PUT/DELETE.

If you want, I can **also make inline PATCH editable in the table itself**, so users can directly edit any row cell and save — no prompts needed. It’s more user-friendly.

Do you want me to do that too?

**PART 2**

Working version of Billing API

Wwwroot/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Billing Invoice Entry</title>

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

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

label {

display: block;

margin-top: 10px;

}

input, select {

width: 100%;

padding: 6px;

margin-top: 2px;

}

button {

margin-top: 15px;

padding: 10px 20px;

margin-right: 5px;

}

.status {

margin-top: 15px;

font-weight: bold;

color: green;

}

.container {

max-width: 800px;

margin: auto;

}

</style>

</head>

<body>

<div class="container">

<center><h2>1 Stop - Enter Invoice Details</h2></center>

<form id="invoiceForm">

<label>RefNo: <input type="number" id="RefNo" required></label>

<label>InvoiceNo: <input type="text" id="InvoiceNo" required></label>

<label>InvoiceDate: <input type="date" id="InvoiceDate" required></label>

<label>BillType: <input type="text" id="BillType"></label>

<label>OrderNo: <input type="text" id="OrderNo"></label>

<label>OrderDate: <input type="date" id="OrderDate"></label>

<label>TermsPayment: <input type="text" id="TermsPayment"></label>

<label>CustomerName: <input type="text" id="CustomerName"></label>

<label>AddressOne: <input type="text" id="AddressOne"></label>

<label>AddressTwo: <input type="text" id="AddressTwo"></label>

<label>AddressThree: <input type="text" id="AddressThree"></label>

<label>AddressFour: <input type="text" id="AddressFour"></label>

<label>CustomerGSTNo: <input type="text" id="CustomerGSTNo"></label>

<label>GSTState: <input type="text" id="GSTState"></label>

<label>ItemNo: <input type="text" id="ItemNo"></label>

<label>Description: <input type="text" id="Description"></label>

<label>HSN-SAC: <input type="text" id="HSNSAC"></label>

<label>Quantity: <input type="number" id="Quantity"></label>

<label>Rate: <input type="number" step="0.01" id="Rate"></label>

<label>PER: <input type="text" id="PER"></label>

<label>GSTPC: <input type="number" step="0.01" id="GSTPC"></label>

<label>RupeesOne: <input type="text" id="RupeesOne"></label>

<label>RupeesTwo: <input type="text" id="RupeesTwo"></label>

<!-- Action buttons -->

<button type="submit">Submit Invoice</button>

<button type="button" id="saveInvoice">Save Changes</button>

<button type="button" id="deleteInvoice">Delete by RefNo</button>

<button type="button" id="deleteInvoiceNo">Delete by InvoiceNo</button>

<button type="button" id="searchRefNo">Search by RefNo</button>

<br>

<!-- Navigation buttons -->

<button type="button" id="firstRecord">First</button>

<button type="button" id="prevRecord">Previous</button>

<button type="button" id="nextRecord">Next</button>

<button type="button" id="lastRecord">Last</button>

<button type="button" id="searchInvoiceNo">Search by InvoiceNo</button>

<button type="button" id="billUpdate">BillUpdate</button>

</form>

<div class="status" id="status"></div>

</div>

<script>

const apiBase = "https://localhost:7120/api/invoice"; // Change to your API URL

let invoices = []; // store all invoices for navigation

let currentIndex = -1;

// Utility: Load invoice data into form

function loadInvoiceToForm(inv) {

document.getElementById("RefNo").value = inv.RefNo ?? inv.refNo ?? "";

document.getElementById("InvoiceNo").value = inv.InvoiceNo ?? inv.invoiceNo ?? "";

document.getElementById("InvoiceDate").value = (inv.InvoiceDate ?? inv.invoiceDate ?? "").split('T')[0];

document.getElementById("BillType").value = inv.BillType ?? inv.billType ?? "";

document.getElementById("OrderNo").value = inv.OrderNo ?? inv.orderNo ?? "";

document.getElementById("OrderDate").value = (inv.OrderDate ?? inv.orderDate ?? "").split('T')[0];

document.getElementById("TermsPayment").value = inv.TermsPayment ?? inv.termsPayment ?? "";

document.getElementById("CustomerName").value = inv.CustomerName ?? inv.customerName ?? "";

document.getElementById("AddressOne").value = inv.AddressOne ?? inv.addressOne ?? "";

document.getElementById("AddressTwo").value = inv.AddressTwo ?? inv.addressTwo ?? "";

document.getElementById("AddressThree").value = inv.AddressThree ?? inv.addressThree ?? "";

document.getElementById("AddressFour").value = inv.AddressFour ?? inv.addressFour ?? "";

document.getElementById("CustomerGSTNo").value = inv.CustomerGSTNo ?? inv.customerGSTNo ?? "";

document.getElementById("GSTState").value = inv.GSTState ?? inv.gstState ?? "";

document.getElementById("ItemNo").value = inv.ItemNo ?? inv.itemNo ?? "";

document.getElementById("Description").value = inv.Description ?? inv.description ?? "";

document.getElementById("HSNSAC").value = inv.HSNSAC ?? inv.hsnsac ?? "";

document.getElementById("Quantity").value = inv.Quantity ?? inv.quantity ?? 0;

document.getElementById("Rate").value = inv.Rate ?? inv.rate ?? 0;

document.getElementById("PER").value = inv.PER ?? inv.per ?? "";

document.getElementById("GSTPC").value = inv.GSTPC ?? inv.gstpc ?? 0;

document.getElementById("RupeesOne").value = inv.RupeesOne ?? inv.rupeesOne ?? "";

document.getElementById("RupeesTwo").value = inv.RupeesTwo ?? inv.rupeesTwo ?? "";

}

// Utility: Collect form data

function getFormData() {

const data = {};

[...document.querySelectorAll("#invoiceForm input")].forEach(input => {

data[input.id] = input.type === "number" ? parseFloat(input.value) || 0 : input.value;

});

return data;

}

// Fetch all invoices

async function fetchAllInvoices() {

try {

const res = await fetch(apiBase);

if (!res.ok) throw new Error("API returned " + res.status);

invoices = await res.json();

if (invoices.length > 0) {

currentIndex = 0;

loadInvoiceToForm(invoices[currentIndex]);

document.getElementById("status").innerText = "Fetched " + invoices.length + " invoices";

} else {

currentIndex = -1;

document.getElementById("status").innerText = "No invoices found";

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Error Fetching Invoices: " + err.message;

}

}

// Navigation buttons

document.getElementById("firstRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = 0;

loadInvoiceToForm(invoices[currentIndex]);

});

document.getElementById("lastRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = invoices.length - 1;

loadInvoiceToForm(invoices[currentIndex]);

});

document.getElementById("nextRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = Math.min(currentIndex + 1, invoices.length - 1);

loadInvoiceToForm(invoices[currentIndex]);

});

document.getElementById("prevRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = Math.max(currentIndex - 1, 0);

loadInvoiceToForm(invoices[currentIndex]);

});

// Submit new invoice (POST)

document.getElementById("invoiceForm").addEventListener("submit", async function (e) {

e.preventDefault();

const data = getFormData();

try {

const res = await fetch(apiBase, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

document.getElementById("status").innerText = "✅ Invoice submitted successfully!";

await fetchAllInvoices(); // refresh local list

} else {

const err = await res.json();

document.getElementById("status").innerText = "❌ Error: " + (err.message || "Unknown");

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Network or API error";

}

});

// Save changes (PUT by RefNo)

document.getElementById("saveInvoice").addEventListener("click", async () => {

const refNo = document.getElementById("RefNo").value;

if (!refNo) { alert("No RefNo present."); return; }

const data = getFormData();

try {

const res = await fetch(`${apiBase}/ref/${refNo}`, {

method: "PUT",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

document.getElementById("status").innerText = "✅ Saved successfully (by RefNo)";

await fetchAllInvoices();

} else {

const err = await res.json();

document.getElementById("status").innerText = "❌ Error: " + (err.message || "Unknown");

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Network or API error";

}

});

// Delete by RefNo

document.getElementById("deleteInvoice").addEventListener("click", async () => {

const refNo = document.getElementById("RefNo").value || prompt("Enter RefNo to delete:");

if (!refNo) return;

if (!confirm("Delete invoice RefNo " + refNo + "?")) return;

try {

const res = await fetch(`${apiBase}/ref/${refNo}`, { method: "DELETE" });

if (res.ok) {

document.getElementById("status").innerText = "✅ Deleted successfully (by RefNo)";

await fetchAllInvoices();

}

} catch (err) { console.error(err); }

});

// Delete by InvoiceNo

document.getElementById("deleteInvoiceNo").addEventListener("click", async () => {

const invoiceNo = prompt("Enter InvoiceNo to delete:");

if (!invoiceNo) return;

if (!confirm("Delete invoice InvoiceNo " + invoiceNo + "?")) return;

try {

const res = await fetch(`${apiBase}/invoiceNo/${invoiceNo}`, { method: "DELETE" });

if (res.ok) {

document.getElementById("status").innerText = "✅ Deleted successfully (by InvoiceNo)";

await fetchAllInvoices();

}

} catch (err) { console.error(err); }

});

// 🔹 Search by RefNo

document.getElementById("searchRefNo").addEventListener("click", async () => {

// Prompt user to enter RefNo

const refNo = prompt("Enter RefNo to search:");

if (!refNo) return;

try {

// Fetch invoice by RefNo

const res = await fetch(`${apiBase}/ref/${refNo}`);

if (!res.ok) {

document.getElementById("status").innerText = "❌ RefNo not found";

return;

}

const inv = await res.json();

// Load the invoice into the form

loadInvoiceToForm(inv);

// Update status message

document.getElementById("status").innerText = `✅ Invoice loaded for editing (RefNo: ${refNo})`;

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Error fetching by RefNo";

}

});

// Search by InvoiceNo

document.getElementById("searchInvoiceNo").addEventListener("click", async () => {

const invoiceNo = prompt("Enter InvoiceNo to search:");

if (!invoiceNo) return;

try {

const encodedInvoiceNo = encodeURIComponent(invoiceNo); //handles special chars or / etc

const res = await fetch(`${apiBase}/invoiceNo/${encodedInvoiceNo}`); //handles special chars or / etc

if (!res.ok) { document.getElementById("status").innerText = "❌ Not found"; return; }

const inv = await res.json();

loadInvoiceToForm(inv);

document.getElementById("status").innerText = "✅ Invoice loaded for editing (by InvoiceNo)";

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Error fetching by InvoiceNo";

}

});

// PUT by InvoiceNo (update)

async function saveByInvoiceNo(invoiceNo) {

const data = getFormData();

try {

const res = await fetch(`${apiBase}/invoiceNo/${invoiceNo}`, {

method: "PUT",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

document.getElementById("status").innerText = "✅ Saved successfully (by InvoiceNo)";

await fetchAllInvoices();

} else {

const err = await res.json();

document.getElementById("status").innerText = "❌ Error: " + (err.message || "Unknown");

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Network or API error";

}

}

/\*

🔹 NEW BILL UPDATE HANDLER

Calls POST /api/invoice/billupdate

Pushes current form values into "SingleBillSheet" in Excel in the new sheet first row

\*/

// BillUpdate button (send current form data to API)

document.getElementById("billUpdate").addEventListener("click", async () => {

const data = getFormData(); // Collects all inputs from form

try {

const res = await fetch(`${apiBase}/billupdate`, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

const result = await res.json();

document.getElementById("status").innerText = result.message;

} else {

const err = await res.text();

document.getElementById("status").innerText = "❌ Error in BillUpdate: " + err;

}

} catch (err) {

console.error("BillUpdate Error:", err);

document.getElementById("status").innerText = "❌ Network or API error";

}

});

// Update the new sheet first row is over

// Initial fetch

fetchAllInvoices();

</script>

</body>

</html>

**Controller/ InvoiceController.CS**

using BillingAPI.Models;

using Microsoft.AspNetCore.Mvc;

using ClosedXML.Excel;

namespace BillingAPI.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class InvoiceController : ControllerBase

{

private readonly string \_filePath;

public InvoiceController(IWebHostEnvironment env)

{

\_filePath = Path.Combine(env.ContentRootPath, "Billing", "TaxInvoiceFormat.xlsx");

}

// ✅ Create new invoice

[HttpPost]

public IActionResult Create([FromBody] InvoiceModel model)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int lastRow = ws.LastRowUsed()?.RowNumber() ?? 1;

int newRow = lastRow + 1;

MapModelToRow(ws, newRow, model);

workbook.Save();

return Ok(new { message = "Invoice created successfully", row = newRow });

}

// ✅ Get all invoices

[HttpGet]

public IActionResult GetAll()

{

var list = new List<InvoiceModel>();

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

list.Add(RowToModel(row));

}

return Ok(list);

}

// ✅ Get by RefNo

[HttpGet("ref/{refNo}")]

public IActionResult GetByRefNo(int refNo)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

if (row.Cell(1).GetValue<int>() == refNo)

return Ok(RowToModel(row));

}

return NotFound(new { message = $"RefNo {refNo} not found" });

}

// ✅ Get by InvoiceNo

[HttpGet("invoiceNo/{invoiceNo}")]

public IActionResult GetByInvoiceNo(string invoiceNo)

{

string decodedInvoiceNo = Uri.UnescapeDataString(invoiceNo); //to handle "/" character in invoiceNo

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

if (row.Cell(2).GetValue<string>().Equals(decodedInvoiceNo, StringComparison.OrdinalIgnoreCase))

return Ok(RowToModel(row));

}

return NotFound(new { message = $"InvoiceNo {invoiceNo} not found" });

}

// ✅ Full Update by RefNo

[HttpPut("ref/{refNo}")]

public IActionResult UpdateByRefNo(int refNo, [FromBody] InvoiceModel model)

{

return UpdateInvoice("RefNo", refNo.ToString(), model);

}

// ✅ Full Update by InvoiceNo

[HttpPut("invoiceNo/{invoiceNo}")]

public IActionResult UpdateByInvoiceNo(string invoiceNo, [FromBody] InvoiceModel model)

{

string decodedInvoiceNo = Uri.UnescapeDataString(invoiceNo);

return UpdateInvoice("InvoiceNo", decodedInvoiceNo, model);

}

// ✅ Partial update (PATCH) by RefNo

[HttpPatch("ref/{refNo}")]

public IActionResult PatchByRefNo(int refNo, [FromBody] Dictionary<string, object> updates)

{

return PatchInvoice("RefNo", refNo.ToString(), updates);

}

// ✅ Delete by RefNo

[HttpDelete("ref/{refNo}")]

public IActionResult DeleteByRefNo(int refNo)

{

return DeleteInvoice("RefNo", refNo.ToString());

}

// ✅ Delete by InvoiceNo

[HttpDelete("invoiceNo/{invoiceNo}")]

public IActionResult DeleteByInvoiceNo(string invoiceNo)

{

string decodedInvoiceNo = Uri.UnescapeDataString(invoiceNo);

return DeleteInvoice("InvoiceNo", decodedInvoiceNo);

}

// 🔹 NEW ADDITION: BillUpdate - Insert last submitted invoice into SingleBillSheet for new sheet of billing to update first row in new sheet

// ✅ BillUpdate endpoint

[HttpPost("billupdate")]

public IActionResult BillUpdate([FromBody] InvoiceModel model)

{

try

{

//string filePath = Path.Combine(Directory.GetCurrentDirectory(), "TaxInvoiceFormat.xlsx");

//String filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Billing", "TaxInvoiceFormat.xlsx");

string projectRoot = Directory.GetParent(AppDomain.CurrentDomain.BaseDirectory).Parent.Parent.Parent.FullName;

string filePath = Path.Combine(projectRoot, "Billing", "TaxInvoiceFormat.xlsx");

Console.WriteLine($"Looking for Excel file at: {filePath}");

using (var workbook = new XLWorkbook(filePath))

{

var singleBillSheet = workbook.Worksheets.FirstOrDefault(ws => ws.Name == "SingleBillSheet");

if (singleBillSheet == null)

{

// ✅ Create sheet with headers if it doesn't exist

singleBillSheet = workbook.AddWorksheet("SingleBillSheet");

singleBillSheet.Cell(1, 1).Value = "RefNo";

singleBillSheet.Cell(1, 2).Value = "InvoiceNo";

singleBillSheet.Cell(1, 3).Value = "InvoiceDate";

singleBillSheet.Cell(1, 4).Value = "BillType";

singleBillSheet.Cell(1, 5).Value = "OrderNo";

singleBillSheet.Cell(1, 6).Value = "OrderDate";

singleBillSheet.Cell(1, 7).Value = "TermsPayment";

singleBillSheet.Cell(1, 8).Value = "CustomerName";

singleBillSheet.Cell(1, 9).Value = "AddressOne";

singleBillSheet.Cell(1, 10).Value = "AddressTwo";

singleBillSheet.Cell(1, 11).Value = "AddressThree";

singleBillSheet.Cell(1, 12).Value = "AddressFour";

singleBillSheet.Cell(1, 13).Value = "CustomerGSTNo";

singleBillSheet.Cell(1, 14).Value = "GSTState";

singleBillSheet.Cell(1, 15).Value = "ItemNo";

singleBillSheet.Cell(1, 16).Value = "Description";

singleBillSheet.Cell(1, 17).Value = "HSNSAC";

singleBillSheet.Cell(1, 18).Value = "Quantity";

singleBillSheet.Cell(1, 19).Value = "Rate";

singleBillSheet.Cell(1, 20).Value = "PER";

singleBillSheet.Cell(1, 21).Value = "GSTPC";

singleBillSheet.Cell(1, 22).Value = "RupeesOne";

singleBillSheet.Cell(1, 23).Value = "RupeesTwo";

}

// ✅ Convert invoice model to array

var rowData = new object[]

{

model.RefNo,

model.InvoiceNo,

model.InvoiceDate,

model.BillType,

model.OrderNo,

model.OrderDate,

model.TermsPayment,

model.CustomerName,

model.AddressOne,

model.AddressTwo,

model.AddressThree,

model.AddressFour,

model.CustomerGSTNo,

model.GSTState,

model.ItemNo,

model.Description,

model.HSNSAC,

model.Quantity,

model.Rate,

model.PER,

model.GSTPC,

model.RupeesOne,

model.RupeesTwo

};

// ✅ Insert as row 2 (push down older data)

singleBillSheet.Row(2).InsertRowsAbove(1);

for (int i = 0; i < rowData.Length; i++)

{

singleBillSheet.Cell(2, i + 1).Value = rowData[i]?.ToString() ?? string.Empty;

}

workbook.Save();

}

return Ok(new { message = "✅ Invoice added to SingleBillSheet (as first row)" });

}

catch (Exception ex)

{

return StatusCode(500, $"Error in BillUpdate: {ex.Message}");

}

}

// 🔹 END OF BillUpdate addition - for new sheet of billing to update first row in new sheet

private IActionResult UpdateInvoice(string keyType, string keyValue, InvoiceModel model)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

MapModelToRow(ws, targetRow, model);

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} updated successfully" });

}

private IActionResult PatchInvoice(string keyType, string keyValue, Dictionary<string, object> updates)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

// Apply updates

var rowToUpdate = ws.Row(targetRow);

foreach (var kv in updates)

{

var cell = kv.Key switch

{

"RefNo" => rowToUpdate.Cell(1),

"InvoiceNo" => rowToUpdate.Cell(2),

"InvoiceDate" => rowToUpdate.Cell(3),

"BillType" => rowToUpdate.Cell(4),

"OrderNo" => rowToUpdate.Cell(5),

"OrderDate" => rowToUpdate.Cell(6),

"TermsPayment" => rowToUpdate.Cell(7),

"CustomerName" => rowToUpdate.Cell(8),

"AddressOne" => rowToUpdate.Cell(9),

"AddressTwo" => rowToUpdate.Cell(10),

"AddressThree" => rowToUpdate.Cell(11),

"AddressFour" => rowToUpdate.Cell(12),

"CustomerGSTNo" => rowToUpdate.Cell(13),

"GSTState" => rowToUpdate.Cell(14),

"ItemNo" => rowToUpdate.Cell(15),

"Description" => rowToUpdate.Cell(16),

"HSNSAC" => rowToUpdate.Cell(17),

"Quantity" => rowToUpdate.Cell(18),

"Rate" => rowToUpdate.Cell(19),

"PER" => rowToUpdate.Cell(20),

"GSTPC" => rowToUpdate.Cell(21),

"RupeesOne" => rowToUpdate.Cell(22),

"RupeesTwo" => rowToUpdate.Cell(23),

\_ => null

};

if (cell != null)

{

object value = kv.Value;

if (value == null)

cell.Clear();

else if (value is string s)

cell.Value = s;

else if (value is int i)

cell.Value = i;

else if (value is long l)

cell.Value = l;

else if (value is double d)

cell.Value = d;

else if (value is decimal dec)

cell.Value = dec;

else if (value is DateTime dt)

cell.Value = dt;

else

cell.Value = value.ToString(); // fallback

}

}

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} patched successfully" });

}

private IActionResult DeleteInvoice(string keyType, string keyValue)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

ws.Row(targetRow).Delete();

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} deleted successfully" });

}

// 🔹 Mapping helpers

private void MapModelToRow(IXLWorksheet ws, int row, InvoiceModel m)

{

ws.Cell(row, 1).Value = m.RefNo;

ws.Cell(row, 2).Value = m.InvoiceNo;

ws.Cell(row, 3).Value = m.InvoiceDate;

ws.Cell(row, 4).Value = m.BillType;

ws.Cell(row, 5).Value = m.OrderNo;

ws.Cell(row, 6).Value = m.OrderDate;

ws.Cell(row, 7).Value = m.TermsPayment;

ws.Cell(row, 8).Value = m.CustomerName;

ws.Cell(row, 9).Value = m.AddressOne;

ws.Cell(row, 10).Value = m.AddressTwo;

ws.Cell(row, 11).Value = m.AddressThree;

ws.Cell(row, 12).Value = m.AddressFour;

ws.Cell(row, 13).Value = m.CustomerGSTNo;

ws.Cell(row, 14).Value = m.GSTState;

ws.Cell(row, 15).Value = m.ItemNo;

ws.Cell(row, 16).Value = m.Description;

ws.Cell(row, 17).Value = m.HSNSAC;

ws.Cell(row, 18).Value = m.Quantity;

ws.Cell(row, 19).Value = m.Rate;

ws.Cell(row, 20).Value = m.PER;

ws.Cell(row, 21).Value = m.GSTPC;

ws.Cell(row, 22).Value = m.RupeesOne;

ws.Cell(row, 23).Value = m.RupeesTwo;

}

private InvoiceModel RowToModel(IXLRow row)

{

return new InvoiceModel

{

RefNo = row.Cell(1).GetValue<int?>() ?? 0,

InvoiceNo = row.Cell(2).GetValue<string>(),

InvoiceDate = row.Cell(3).TryGetValue<DateTime>(out var invDate) ? invDate : DateTime.MinValue,

BillType = row.Cell(4).GetValue<string>(),

OrderNo = row.Cell(5).GetValue<string>(),

OrderDate = row.Cell(6).TryGetValue<DateTime>(out var ordDate) ? ordDate : DateTime.MinValue,

TermsPayment = row.Cell(7).GetValue<string>(),

CustomerName = row.Cell(8).GetValue<string>(),

AddressOne = row.Cell(9).GetValue<string>(),

AddressTwo = row.Cell(10).GetValue<string>(),

AddressThree = row.Cell(11).GetValue<string>(),

AddressFour = row.Cell(12).GetValue<string>(),

CustomerGSTNo = row.Cell(13).GetValue<string>(),

GSTState = row.Cell(14).GetValue<string>(),

ItemNo = row.Cell(15).GetValue<string>(),

Description = row.Cell(16).GetValue<string>(),

HSNSAC = row.Cell(17).GetValue<string>(),

Quantity = row.Cell(18).TryGetValue<int>(out var qty) ? qty : 0,

Rate = row.Cell(19).TryGetValue<decimal>(out var rate) ? rate : 0,

PER = row.Cell(20).GetValue<string>(),

GSTPC = row.Cell(21).TryGetValue<decimal>(out var gst) ? gst : 0,

RupeesOne = row.Cell(22).GetValue<string>(),

RupeesTwo = row.Cell(23).GetValue<string>()

};

}

}

}

**Model/ InvoiceModel.cs**

namespace BillingAPI.Models

{

public class InvoiceModel

{

public int RefNo { get; set; }

public string InvoiceNo { get; set; } = "";

public DateTime InvoiceDate { get; set; }

public string BillType { get; set; } = "";

public string OrderNo { get; set; } = "";

public DateTime? OrderDate { get; set; }

public string TermsPayment { get; set; } = "";

public string CustomerName { get; set; } = "";

public string AddressOne { get; set; } = "";

public string AddressTwo { get; set; } = "";

public string AddressThree { get; set; } = "";

public string AddressFour { get; set; } = "";

public string CustomerGSTNo { get; set; } = "";

public string GSTState { get; set; } = "";

public string ItemNo { get; set; } = "";

public string Description { get; set; } = "";

public string HSNSAC { get; set; } = "";

public int Quantity { get; set; }

public decimal Rate { get; set; }

public string PER { get; set; } = "";

public decimal GSTPC { get; set; }

public string RupeesOne { get; set; } = "";

public string RupeesTwo { get; set; } = "";

}

}

**Program.cs**

using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// ✅ Enable CORS for local front-end

builder.Services.AddCors(options =>

{

options.AddDefaultPolicy(policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Serve static files from wwwroot

app.UseStaticFiles();

app.UseAuthorization();

// ✅ Use CORS

app.UseCors();

app.MapControllers();

app.Run();

**PART – 3**

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Billing Invoice Entry</title>

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

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

label {

display: block;

margin-top: 10px;

}

input, select {

width: 100%;

padding: 6px;

margin-top: 2px;

}

button {

margin-top: 15px;

padding: 10px 20px;

margin-right: 5px;

}

.status {

margin-top: 15px;

font-weight: bold;

color: green;

}

.container {

max-width: 800px;

margin: auto;

}

</style>

</head>

<body>

<div class="container">

<center><h2>1 Stop - Enter Invoice Details</h2></center>

<form id="invoiceForm">

<label>RefNo: <input type="number" id="RefNo" required></label>

<label>InvoiceNo: <input type="text" id="InvoiceNo" required></label>

<label>InvoiceDate: <input type="date" id="InvoiceDate" required></label>

<label>BillType: <input type="text" id="BillType"></label>

<label>OrderNo: <input type="text" id="OrderNo"></label>

<label>OrderDate: <input type="date" id="OrderDate"></label>

<label>TermsPayment: <input type="text" id="TermsPayment"></label>

<!--

<label>CustomerName: <input type="text" id="CustomerName"></label>

<label>AddressOne: <input type="text" id="AddressOne"></label>

<label>AddressTwo: <input type="text" id="AddressTwo"></label>

<label>AddressThree: <input type="text" id="AddressThree"></label>

<label>AddressFour: <input type="text" id="AddressFour"></label>

-->

<!-- Customer and Delivery Address Section -->

<fieldset class="address-section">

<legend><h4>Customer Name & Address</h4></legend>

<div class="address-grid">

<div class="address-column">

<label>

Customer Name:

<input type="text" id="CustomerName" placeholder="Customer Name">

</label>

<label>

Address One:

<input type="text" id="AddressOne" placeholder="Address Line 1">

</label>

<label>

Address Two:

<input type="text" id="AddressTwo" placeholder="Address Line 2">

</label>

<label>

Address Three:

<input type="text" id="AddressThree" placeholder="Address Line 3">

</label>

<label>

Address Four:

<input type="text" id="AddressFour" placeholder="Address Line 4">

</label>

<label>

Customer Phone No:

<input type="text" id="CustomerPhone" placeholder="Phone No">

</label>

</div>

<div class="address-column">

<h4>Delivery Address</h4>

<label>

Delivery Name:

<input type="text" id="DeliveryName" placeholder="Delivery Name">

</label>

<label>

Address One:

<input type="text" id="DelAddressOne" placeholder="Address Line 1">

</label>

<label>

Address Two:

<input type="text" id="DelAddressTwo" placeholder="Address Line 2">

</label>

<label>

Address Three:

<input type="text" id="DelAddressThree" placeholder="Address Line 3">

</label>

<label>

Address Four:

<input type="text" id="DelAddressFour" placeholder="Address Line 4">

</label>

<label>

Delivery Phone No:

<input type="text" id="DeliveryPhone" placeholder="DeliveryPhone No">

</label>

</div>

</div>

</fieldset>

<!-- Customer and Delivery Address Section above -->

<label>CustomerGSTNo: <input type="text" id="CustomerGSTNo"></label>

<label>GSTState: <input type="text" id="GSTState"></label>

<label>ItemNo: <input type="text" id="ItemNo"></label>

<label>Description: <input type="text" id="Description"></label>

<label>HSN-SAC: <input type="text" id="HSNSAC"></label>

<label>Quantity: <input type="number" id="Quantity"></label>

<label>Rate: <input type="number" step="0.01" id="Rate"></label>

<label>PER: <input type="text" id="PER"></label>

<label>GSTPC: <input type="number" step="0.01" id="GSTPC"></label>

<label>RupeesOne: <input type="text" id="RupeesOne"></label>

<label>RupeesTwo: <input type="text" id="RupeesTwo"></label>

<!-- Action buttons -->

<button type="submit">Submit Invoice</button>

<button type="button" id="saveInvoice">Save Changes</button>

<button type="button" id="deleteInvoice">Delete by RefNo</button>

<button type="button" id="deleteInvoiceNo">Delete by InvoiceNo</button>

<button type="button" id="searchRefNo">Search by RefNo</button>

<br>

<!-- Navigation buttons -->

<button type="button" id="firstRecord">First</button>

<button type="button" id="prevRecord">Previous</button>

<button type="button" id="nextRecord">Next</button>

<button type="button" id="lastRecord">Last</button>

<button type="button" id="searchInvoiceNo">Search by InvoiceNo</button>

<button type="button" id="billUpdate">BillUpdate</button>

</form>

<div class="status" id="status"></div>

</div>

<script>

const apiBase = "https://localhost:7120/api/invoice"; // Change to your API URL

let invoices = []; // store all invoices for navigation

let currentIndex = -1;

// Utility: Load invoice data into form

function loadInvoiceToForm(inv) {

document.getElementById("RefNo").value = inv.RefNo ?? inv.refNo ?? "";

document.getElementById("InvoiceNo").value = inv.InvoiceNo ?? inv.invoiceNo ?? "";

document.getElementById("InvoiceDate").value = (inv.InvoiceDate ?? inv.invoiceDate ?? "").split('T')[0];

document.getElementById("BillType").value = inv.BillType ?? inv.billType ?? "";

document.getElementById("OrderNo").value = inv.OrderNo ?? inv.orderNo ?? "";

document.getElementById("OrderDate").value = (inv.OrderDate ?? inv.orderDate ?? "").split('T')[0];

document.getElementById("TermsPayment").value = inv.TermsPayment ?? inv.termsPayment ?? "";

document.getElementById("CustomerName").value = inv.CustomerName ?? inv.customerName ?? "";

document.getElementById("AddressOne").value = inv.AddressOne ?? inv.addressOne ?? "";

document.getElementById("AddressTwo").value = inv.AddressTwo ?? inv.addressTwo ?? "";

document.getElementById("AddressThree").value = inv.AddressThree ?? inv.addressThree ?? "";

document.getElementById("AddressFour").value = inv.AddressFour ?? inv.addressFour ?? "";

document.getElementById("CustomerPhone").value = inv.CustomerPhone ?? inv.customerPhone ?? "";

document.getElementById("DeliveryName").value = inv.DeliveryName ?? inv.deliveryName ?? "";

document.getElementById("DelAddressOne").value = inv.DelAddressOne ?? inv.delAddressOne ?? "";

document.getElementById("DelAddressTwo").value = inv.DelAddressTwo ?? inv.delAddressTwo ?? "";

document.getElementById("DelAddressThree").value = inv.DelAddressThree ?? inv.delAddressThree ?? "";

document.getElementById("DelAddressFour").value = inv.DelAddressFour ?? inv.delAddressFour ?? "";

document.getElementById("DeliveryPhone").value = inv.DeliveryPhone ?? inv.deliveryPhone ?? "";

document.getElementById("CustomerGSTNo").value = inv.CustomerGSTNo ?? inv.customerGSTNo ?? "";

document.getElementById("GSTState").value = inv.GSTState ?? inv.gstState ?? "";

document.getElementById("ItemNo").value = inv.ItemNo ?? inv.itemNo ?? "";

document.getElementById("Description").value = inv.Description ?? inv.description ?? "";

document.getElementById("HSNSAC").value = inv.HSNSAC ?? inv.hsnsac ?? "";

document.getElementById("Quantity").value = inv.Quantity ?? inv.quantity ?? 0;

document.getElementById("Rate").value = inv.Rate ?? inv.rate ?? 0;

document.getElementById("PER").value = inv.PER ?? inv.per ?? "";

document.getElementById("GSTPC").value = inv.GSTPC ?? inv.gstpc ?? 0;

document.getElementById("RupeesOne").value = inv.RupeesOne ?? inv.rupeesOne ?? "";

document.getElementById("RupeesTwo").value = inv.RupeesTwo ?? inv.rupeesTwo ?? "";

}

// Utility: Collect form data

function getFormData() {

const data = {};

[...document.querySelectorAll("#invoiceForm input")].forEach(input => {

data[input.id] = input.type === "number" ? parseFloat(input.value) || 0 : input.value;

});

return data;

}

// Fetch all invoices

async function fetchAllInvoices() {

try {

const res = await fetch(apiBase);

if (!res.ok) throw new Error("API returned " + res.status);

invoices = await res.json();

if (invoices.length > 0) {

currentIndex = 0;

loadInvoiceToForm(invoices[currentIndex]);

document.getElementById("status").innerText = "Fetched " + invoices.length + " invoices";

} else {

currentIndex = -1;

document.getElementById("status").innerText = "No invoices found";

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Error Fetching Invoices: " + err.message;

}

}

// Navigation buttons

document.getElementById("firstRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = 0;

loadInvoiceToForm(invoices[currentIndex]);

});

document.getElementById("lastRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = invoices.length - 1;

loadInvoiceToForm(invoices[currentIndex]);

});

document.getElementById("nextRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = Math.min(currentIndex + 1, invoices.length - 1);

loadInvoiceToForm(invoices[currentIndex]);

});

document.getElementById("prevRecord").addEventListener("click", () => {

if (invoices.length === 0) return;

currentIndex = Math.max(currentIndex - 1, 0);

loadInvoiceToForm(invoices[currentIndex]);

});

// Submit new invoice (POST)

document.getElementById("invoiceForm").addEventListener("submit", async function (e) {

e.preventDefault();

const data = getFormData();

try {

const res = await fetch(apiBase, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

document.getElementById("status").innerText = "✅ Invoice submitted successfully!";

await fetchAllInvoices(); // refresh local list

} else {

const err = await res.json();

document.getElementById("status").innerText = "❌ Error: " + (err.message || "Unknown");

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Network or API error";

}

});

// Save changes (PUT by RefNo)

document.getElementById("saveInvoice").addEventListener("click", async () => {

const refNo = document.getElementById("RefNo").value;

if (!refNo) { alert("No RefNo present."); return; }

const data = getFormData();

try {

const res = await fetch(`${apiBase}/ref/${refNo}`, {

method: "PUT",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

document.getElementById("status").innerText = "✅ Saved successfully (by RefNo)";

await fetchAllInvoices();

} else {

const err = await res.json();

document.getElementById("status").innerText = "❌ Error: " + (err.message || "Unknown");

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Network or API error";

}

});

// Delete by RefNo

document.getElementById("deleteInvoice").addEventListener("click", async () => {

const refNo = document.getElementById("RefNo").value || prompt("Enter RefNo to delete:");

if (!refNo) return;

if (!confirm("Delete invoice RefNo " + refNo + "?")) return;

try {

const res = await fetch(`${apiBase}/ref/${refNo}`, { method: "DELETE" });

if (res.ok) {

document.getElementById("status").innerText = "✅ Deleted successfully (by RefNo)";

await fetchAllInvoices();

}

} catch (err) { console.error(err); }

});

// Delete by InvoiceNo

document.getElementById("deleteInvoiceNo").addEventListener("click", async () => {

const invoiceNo = prompt("Enter InvoiceNo to delete:");

if (!invoiceNo) return;

if (!confirm("Delete invoice InvoiceNo " + invoiceNo + "?")) return;

try {

const res = await fetch(`${apiBase}/invoiceNo/${invoiceNo}`, { method: "DELETE" });

if (res.ok) {

document.getElementById("status").innerText = "✅ Deleted successfully (by InvoiceNo)";

await fetchAllInvoices();

}

} catch (err) { console.error(err); }

});

// 🔹 Search by RefNo

document.getElementById("searchRefNo").addEventListener("click", async () => {

// Prompt user to enter RefNo

const refNo = prompt("Enter RefNo to search:");

if (!refNo) return;

try {

// Fetch invoice by RefNo

const res = await fetch(`${apiBase}/ref/${refNo}`);

if (!res.ok) {

document.getElementById("status").innerText = "❌ RefNo not found";

return;

}

const inv = await res.json();

// Load the invoice into the form

loadInvoiceToForm(inv);

// Update status message

document.getElementById("status").innerText = `✅ Invoice loaded for editing (RefNo: ${refNo})`;

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Error fetching by RefNo";

}

});

// Search by InvoiceNo

document.getElementById("searchInvoiceNo").addEventListener("click", async () => {

const invoiceNo = prompt("Enter InvoiceNo to search:");

if (!invoiceNo) return;

try {

const encodedInvoiceNo = encodeURIComponent(invoiceNo); //handles special chars or / etc

const res = await fetch(`${apiBase}/invoiceNo/${encodedInvoiceNo}`); //handles special chars or / etc

if (!res.ok) { document.getElementById("status").innerText = "❌ Not found"; return; }

const inv = await res.json();

loadInvoiceToForm(inv);

document.getElementById("status").innerText = "✅ Invoice loaded for editing (by InvoiceNo)";

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Error fetching by InvoiceNo";

}

});

// PUT by InvoiceNo (update)

async function saveByInvoiceNo(invoiceNo) {

const data = getFormData();

try {

const res = await fetch(`${apiBase}/invoiceNo/${invoiceNo}`, {

method: "PUT",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

document.getElementById("status").innerText = "✅ Saved successfully (by InvoiceNo)";

await fetchAllInvoices();

} else {

const err = await res.json();

document.getElementById("status").innerText = "❌ Error: " + (err.message || "Unknown");

}

} catch (err) {

console.error(err);

document.getElementById("status").innerText = "❌ Network or API error";

}

}

/\*

🔹 NEW BILL UPDATE HANDLER

Calls POST /api/invoice/billupdate

Pushes current form values into "SingleBillSheet" in Excel in the new sheet first row

\*/

// BillUpdate button (send current form data to API)

document.getElementById("billUpdate").addEventListener("click", async () => {

const data = getFormData(); // Collects all inputs from form

try {

const res = await fetch(`${apiBase}/billupdate`, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(data)

});

if (res.ok) {

const result = await res.json();

document.getElementById("status").innerText = result.message;

} else {

const err = await res.text();

document.getElementById("status").innerText = "❌ Error in BillUpdate: " + err;

}

} catch (err) {

console.error("BillUpdate Error:", err);

document.getElementById("status").innerText = "❌ Network or API error";

}

});

// Update the new sheet first row is over

// Initial fetch

fetchAllInvoices();

</script>

</body>

</html>

**InvoiceController.CS**

using BillingAPI.Models;

using Microsoft.AspNetCore.Mvc;

using ClosedXML.Excel;

namespace BillingAPI.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class InvoiceController : ControllerBase

{

private readonly string \_filePath;

public InvoiceController(IWebHostEnvironment env)

{

\_filePath = Path.Combine(env.ContentRootPath, "Billing", "TaxInvoiceFormat.xlsx");

}

// ✅ Create new invoice

[HttpPost]

public IActionResult Create([FromBody] InvoiceModel model)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int lastRow = ws.LastRowUsed()?.RowNumber() ?? 1;

int newRow = lastRow + 1;

MapModelToRow(ws, newRow, model);

workbook.Save();

return Ok(new { message = "Invoice created successfully", row = newRow });

}

// ✅ Get all invoices

[HttpGet]

public IActionResult GetAll()

{

var list = new List<InvoiceModel>();

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

list.Add(RowToModel(row));

}

return Ok(list);

}

// ✅ Get by RefNo

[HttpGet("ref/{refNo}")]

public IActionResult GetByRefNo(int refNo)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

if (row.Cell(1).GetValue<int>() == refNo)

return Ok(RowToModel(row));

}

return NotFound(new { message = $"RefNo {refNo} not found" });

}

// ✅ Get by InvoiceNo

[HttpGet("invoiceNo/{invoiceNo}")]

public IActionResult GetByInvoiceNo(string invoiceNo)

{

string decodedInvoiceNo = Uri.UnescapeDataString(invoiceNo); //to handle "/" character in invoiceNo

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

foreach (var row in ws.RowsUsed().Skip(1))

{

if (row.Cell(2).GetValue<string>().Equals(decodedInvoiceNo, StringComparison.OrdinalIgnoreCase))

return Ok(RowToModel(row));

}

return NotFound(new { message = $"InvoiceNo {invoiceNo} not found" });

}

// ✅ Full Update by RefNo

[HttpPut("ref/{refNo}")]

public IActionResult UpdateByRefNo(int refNo, [FromBody] InvoiceModel model)

{

return UpdateInvoice("RefNo", refNo.ToString(), model);

}

// ✅ Full Update by InvoiceNo

[HttpPut("invoiceNo/{invoiceNo}")]

public IActionResult UpdateByInvoiceNo(string invoiceNo, [FromBody] InvoiceModel model)

{

string decodedInvoiceNo = Uri.UnescapeDataString(invoiceNo);

return UpdateInvoice("InvoiceNo", decodedInvoiceNo, model);

}

// ✅ Partial update (PATCH) by RefNo

[HttpPatch("ref/{refNo}")]

public IActionResult PatchByRefNo(int refNo, [FromBody] Dictionary<string, object> updates)

{

return PatchInvoice("RefNo", refNo.ToString(), updates);

}

// ✅ Delete by RefNo

[HttpDelete("ref/{refNo}")]

public IActionResult DeleteByRefNo(int refNo)

{

return DeleteInvoice("RefNo", refNo.ToString());

}

// ✅ Delete by InvoiceNo

[HttpDelete("invoiceNo/{invoiceNo}")]

public IActionResult DeleteByInvoiceNo(string invoiceNo)

{

string decodedInvoiceNo = Uri.UnescapeDataString(invoiceNo);

return DeleteInvoice("InvoiceNo", decodedInvoiceNo);

}

// 🔹 NEW ADDITION: BillUpdate - Insert last submitted invoice into SingleBillSheet for new sheet of billing to update first row in new sheet

// ✅ BillUpdate endpoint

[HttpPost("billupdate")]

public IActionResult BillUpdate([FromBody] InvoiceModel model)

{

try

{

//string filePath = Path.Combine(Directory.GetCurrentDirectory(), "TaxInvoiceFormat.xlsx");

//String filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Billing", "TaxInvoiceFormat.xlsx");

string projectRoot = Directory.GetParent(AppDomain.CurrentDomain.BaseDirectory).Parent.Parent.Parent.FullName;

string filePath = Path.Combine(projectRoot, "Billing", "TaxInvoiceFormat.xlsx");

Console.WriteLine($"Looking for Excel file at: {filePath}");

using (var workbook = new XLWorkbook(filePath))

{

var singleBillSheet = workbook.Worksheets.FirstOrDefault(ws => ws.Name == "SingleBillSheet");

if (singleBillSheet == null)

{

// ✅ Create sheet with headers if it doesn't exist

singleBillSheet = workbook.AddWorksheet("SingleBillSheet");

singleBillSheet.Cell(1, 1).Value = "RefNo";

singleBillSheet.Cell(1, 2).Value = "InvoiceNo";

singleBillSheet.Cell(1, 3).Value = "InvoiceDate";

singleBillSheet.Cell(1, 4).Value = "BillType";

singleBillSheet.Cell(1, 5).Value = "OrderNo";

singleBillSheet.Cell(1, 6).Value = "OrderDate";

singleBillSheet.Cell(1, 7).Value = "TermsPayment";

singleBillSheet.Cell(1, 8).Value = "CustomerName";

singleBillSheet.Cell(1, 9).Value = "AddressOne";

singleBillSheet.Cell(1, 10).Value = "AddressTwo";

singleBillSheet.Cell(1, 11).Value = "AddressThree";

singleBillSheet.Cell(1, 12).Value = "AddressFour";

singleBillSheet.Cell(1, 13).Value = "CustomerPhone";

singleBillSheet.Cell(1, 14).Value = "DeliveryName";

singleBillSheet.Cell(1, 15).Value = "DelAddressOne";

singleBillSheet.Cell(1, 16).Value = "DelAddressTwo";

singleBillSheet.Cell(1, 17).Value = "DelAddressThree";

singleBillSheet.Cell(1, 18).Value = "DelAddressFour";

singleBillSheet.Cell(1, 19).Value = "DeliveryPhone";

singleBillSheet.Cell(1, 20).Value = "CustomerGSTNo";

singleBillSheet.Cell(1, 21).Value = "GSTState";

singleBillSheet.Cell(1, 22).Value = "ItemNo";

singleBillSheet.Cell(1, 23).Value = "Description";

singleBillSheet.Cell(1, 24).Value = "HSNSAC";

singleBillSheet.Cell(1, 25).Value = "Quantity";

singleBillSheet.Cell(1, 26).Value = "Rate";

singleBillSheet.Cell(1, 27).Value = "PER";

singleBillSheet.Cell(1, 28).Value = "GSTPC";

singleBillSheet.Cell(1, 29).Value = "RupeesOne";

singleBillSheet.Cell(1, 30).Value = "RupeesTwo";

}

// ✅ Convert invoice model to array

var rowData = new object[]

{

model.RefNo,

model.InvoiceNo,

model.InvoiceDate,

model.BillType,

model.OrderNo,

model.OrderDate,

model.TermsPayment,

model.CustomerName,

model.AddressOne,

model.AddressTwo,

model.AddressThree,

model.AddressFour,

model.CustomerPhone,

model.DeliveryName,

model.DelAddressOne,

model.DelAddressTwo,

model.DelAddressThree,

model.DelAddressFour,

model.DeliveryPhone,

model.CustomerGSTNo,

model.GSTState,

model.ItemNo,

model.Description,

model.HSNSAC,

model.Quantity,

model.Rate,

model.PER,

model.GSTPC,

model.RupeesOne,

model.RupeesTwo

};

// ✅ Insert as row 2 (push down older data)

singleBillSheet.Row(2).InsertRowsAbove(1);

for (int i = 0; i < rowData.Length; i++)

{

singleBillSheet.Cell(2, i + 1).Value = rowData[i]?.ToString() ?? string.Empty;

}

workbook.Save();

}

return Ok(new { message = "✅ Invoice added to SingleBillSheet (as first row)" });

}

catch (Exception ex)

{

return StatusCode(500, $"Error in BillUpdate: {ex.Message}");

}

}

// 🔹 END OF BillUpdate addition - for new sheet of billing to update first row in new sheet

private IActionResult UpdateInvoice(string keyType, string keyValue, InvoiceModel model)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

MapModelToRow(ws, targetRow, model);

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} updated successfully" });

}

private IActionResult PatchInvoice(string keyType, string keyValue, Dictionary<string, object> updates)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

// Apply updates

var rowToUpdate = ws.Row(targetRow);

foreach (var kv in updates)

{

var cell = kv.Key switch

{

"RefNo" => rowToUpdate.Cell(1),

"InvoiceNo" => rowToUpdate.Cell(2),

"InvoiceDate" => rowToUpdate.Cell(3),

"BillType" => rowToUpdate.Cell(4),

"OrderNo" => rowToUpdate.Cell(5),

"OrderDate" => rowToUpdate.Cell(6),

"TermsPayment" => rowToUpdate.Cell(7),

"CustomerName" => rowToUpdate.Cell(8),

"AddressOne" => rowToUpdate.Cell(9),

"AddressTwo" => rowToUpdate.Cell(10),

"AddressThree" => rowToUpdate.Cell(11),

"AddressFour" => rowToUpdate.Cell(12),

"CustomerPhone" => rowToUpdate.Cell(13),

"DeliveryName" => rowToUpdate.Cell(14),

"DelAddressOne" => rowToUpdate.Cell(15),

"DelAddressTwo" => rowToUpdate.Cell(16),

"DelAddressThree" => rowToUpdate.Cell(17),

"DelAddressFour" => rowToUpdate.Cell(18),

"DeliveryPhone" => rowToUpdate.Cell(19),

"CustomerGSTNo" => rowToUpdate.Cell(20),

"GSTState" => rowToUpdate.Cell(21),

"ItemNo" => rowToUpdate.Cell(22),

"Description" => rowToUpdate.Cell(23),

"HSNSAC" => rowToUpdate.Cell(24),

"Quantity" => rowToUpdate.Cell(25),

"Rate" => rowToUpdate.Cell(26),

"PER" => rowToUpdate.Cell(27),

"GSTPC" => rowToUpdate.Cell(28),

"RupeesOne" => rowToUpdate.Cell(29),

"RupeesTwo" => rowToUpdate.Cell(30),

\_ => null

};

if (cell != null)

{

object value = kv.Value;

if (value == null)

cell.Clear();

else if (value is string s)

cell.Value = s;

else if (value is int i)

cell.Value = i;

else if (value is long l)

cell.Value = l;

else if (value is double d)

cell.Value = d;

else if (value is decimal dec)

cell.Value = dec;

else if (value is DateTime dt)

cell.Value = dt;

else

cell.Value = value.ToString(); // fallback

}

}

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} patched successfully" });

}

private IActionResult DeleteInvoice(string keyType, string keyValue)

{

using var workbook = new XLWorkbook(\_filePath);

var ws = workbook.Worksheets.Worksheet(1);

int targetRow = -1;

foreach (var row in ws.RowsUsed().Skip(1))

{

string value = keyType == "RefNo" ? row.Cell(1).GetValue<int>().ToString() : row.Cell(2).GetValue<string>();

if (value.Equals(keyValue, StringComparison.OrdinalIgnoreCase))

{

targetRow = row.RowNumber();

break;

}

}

if (targetRow == -1)

return NotFound(new { message = $"{keyType} {keyValue} not found" });

ws.Row(targetRow).Delete();

workbook.Save();

return Ok(new { message = $"Invoice {keyType} {keyValue} deleted successfully" });

}

// 🔹 Mapping helpers

private void MapModelToRow(IXLWorksheet ws, int row, InvoiceModel m)

{

ws.Cell(row, 1).Value = m.RefNo;

ws.Cell(row, 2).Value = m.InvoiceNo;

ws.Cell(row, 3).Value = m.InvoiceDate;

ws.Cell(row, 4).Value = m.BillType;

ws.Cell(row, 5).Value = m.OrderNo;

ws.Cell(row, 6).Value = m.OrderDate;

ws.Cell(row, 7).Value = m.TermsPayment;

ws.Cell(row, 8).Value = m.CustomerName;

ws.Cell(row, 9).Value = m.AddressOne;

ws.Cell(row, 10).Value = m.AddressTwo;

ws.Cell(row, 11).Value = m.AddressThree;

ws.Cell(row, 12).Value = m.AddressFour;

ws.Cell(row, 13).Value = m.CustomerPhone;

ws.Cell(row, 14).Value = m.DeliveryName;

ws.Cell(row, 15).Value = m.DelAddressOne;

ws.Cell(row, 16).Value = m.DelAddressTwo;

ws.Cell(row, 17).Value = m.DelAddressThree;

ws.Cell(row, 18).Value = m.DelAddressFour;

ws.Cell(row, 19).Value = m.DeliveryPhone;

ws.Cell(row, 20).Value = m.CustomerGSTNo;

ws.Cell(row, 21).Value = m.GSTState;

ws.Cell(row, 22).Value = m.ItemNo;

ws.Cell(row, 23).Value = m.Description;

ws.Cell(row, 24).Value = m.HSNSAC;

ws.Cell(row, 25).Value = m.Quantity;

ws.Cell(row, 26).Value = m.Rate;

ws.Cell(row, 27).Value = m.PER;

ws.Cell(row, 28).Value = m.GSTPC;

ws.Cell(row, 29).Value = m.RupeesOne;

ws.Cell(row, 30).Value = m.RupeesTwo;

}

private InvoiceModel RowToModel(IXLRow row)

{

return new InvoiceModel

{

RefNo = row.Cell(1).GetValue<int?>() ?? 0,

InvoiceNo = row.Cell(2).GetValue<string>(),

InvoiceDate = row.Cell(3).TryGetValue<DateTime>(out var invDate) ? invDate : DateTime.MinValue,

BillType = row.Cell(4).GetValue<string>(),

OrderNo = row.Cell(5).GetValue<string>(),

OrderDate = row.Cell(6).TryGetValue<DateTime>(out var ordDate) ? ordDate : DateTime.MinValue,

TermsPayment = row.Cell(7).GetValue<string>(),

CustomerName = row.Cell(8).GetValue<string>(),

AddressOne = row.Cell(9).GetValue<string>(),

AddressTwo = row.Cell(10).GetValue<string>(),

AddressThree = row.Cell(11).GetValue<string>(),

AddressFour = row.Cell(12).GetValue<string>(),

CustomerPhone = row.Cell(13).GetValue<string>(),

DeliveryName = row.Cell(14).GetValue<string>(),

DelAddressOne = row.Cell(15).GetValue<string>(),

DelAddressTwo = row.Cell(16).GetValue<string>(),

DelAddressThree = row.Cell(17).GetValue<string>(),

DelAddressFour = row.Cell(18).GetValue<string>(),

DeliveryPhone = row.Cell(19).GetValue<string>(),

CustomerGSTNo = row.Cell(20).GetValue<string>(),

GSTState = row.Cell(21).GetValue<string>(),

ItemNo = row.Cell(22).GetValue<string>(),

Description = row.Cell(23).GetValue<string>(),

HSNSAC = row.Cell(24).GetValue<string>(),

Quantity = row.Cell(25).TryGetValue<int>(out var qty) ? qty : 0,

Rate = row.Cell(26).TryGetValue<decimal>(out var rate) ? rate : 0,

PER = row.Cell(27).GetValue<string>(),

GSTPC = row.Cell(28).TryGetValue<decimal>(out var gst) ? gst : 0,

RupeesOne = row.Cell(29).GetValue<string>(),

RupeesTwo = row.Cell(30).GetValue<string>()

};

}

}

}

**InvoiceModel.cs**

namespace BillingAPI.Models

{

public class InvoiceModel

{

public int RefNo { get; set; }

public string InvoiceNo { get; set; } = "";

public DateTime InvoiceDate { get; set; }

public string BillType { get; set; } = "";

public string OrderNo { get; set; } = "";

public DateTime? OrderDate { get; set; }

public string TermsPayment { get; set; } = "";

public string CustomerName { get; set; } = "";

public string AddressOne { get; set; } = "";

public string AddressTwo { get; set; } = "";

public string AddressThree { get; set; } = "";

public string AddressFour { get; set; } = "";

public string CustomerPhone { get; set; } = "";

public string DeliveryName { get; set; } = "";

public string DelAddressOne { get; set; } = "";

public string DelAddressTwo { get; set; } = "";

public string DelAddressThree { get; set; } = "";

public string DelAddressFour { get; set; } = "";

public string DeliveryPhone { get; set; } = "";

public string CustomerGSTNo { get; set; } = "";

public string GSTState { get; set; } = "";

public string ItemNo { get; set; } = "";

public string Description { get; set; } = "";

public string HSNSAC { get; set; } = "";

public int Quantity { get; set; }

public decimal Rate { get; set; }

public string PER { get; set; } = "";

public decimal GSTPC { get; set; }

public string RupeesOne { get; set; } = "";

public string RupeesTwo { get; set; } = "";

}

}

**Program.CS**

using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// ✅ Enable CORS for local front-end

builder.Services.AddCors(options =>

{

options.AddDefaultPolicy(policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Serve static files from wwwroot

app.UseStaticFiles();

app.UseAuthorization();

// ✅ Use CORS

app.UseCors();

app.MapControllers();

app.Run();

**TaxInvoiceFormat.xlsx** in **Billing** Folder inside **BillingAPI**.