using System;

using System.IO;

using System.Web;

using iTextSharp.text;

using iTextSharp.text.pdf;

using iTextSharp.text.pdf.draw;

namespace MedicalClaimFormGenerator

{

public partial class GenerateClaimForm : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

GenerateMedicalClaimForm();

}

}

private void GenerateMedicalClaimForm()

{

// Create document

Document document = new Document(PageSize.A4, 36, 36, 36, 36);

MemoryStream stream = new MemoryStream();

try

{

PdfWriter writer = PdfWriter.GetInstance(document, stream);

document.Open();

// Define fonts

Font titleFont = FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 16);

Font headerFont = FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 12);

Font normalFont = FontFactory.GetFont(FontFactory.HELVETICA, 10);

Font smallFont = FontFactory.GetFont(FontFactory.HELVETICA, 8);

Font boldFont = FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 10);

// Title

Paragraph title = new Paragraph("Medical Expenses Claim form", titleFont);

title.Alignment = Element.ALIGN\_CENTER;

title.SpacingAfter = 20f;

document.Add(title);

// Date field

PdfPTable dateTable = new PdfPTable(2);

dateTable.WidthPercentage = 100;

dateTable.SetWidths(new float[] { 1f, 4f });

PdfPCell dateLabel = new PdfPCell(new Phrase("Date :", normalFont));

dateLabel.Border = Rectangle.NO\_BORDER;

dateTable.AddCell(dateLabel);

PdfPCell dateValue = new PdfPCell(new Phrase("\_\_\_\_/\_\_\_\_/\_\_\_\_\_\_\_\_", normalFont));

dateValue.Border = Rectangle.NO\_BORDER;

dateTable.AddCell(dateValue);

document.Add(dateTable);

document.Add(new Paragraph(" ", normalFont)); // Spacing

// Clinic Name

PdfPTable clinicTable = new PdfPTable(2);

clinicTable.WidthPercentage = 100;

clinicTable.SetWidths(new float[] { 1f, 4f });

PdfPCell clinicLabel = new PdfPCell(new Phrase("Clinic Name", normalFont));

clinicLabel.Border = Rectangle.NO\_BORDER;

clinicTable.AddCell(clinicLabel);

PdfPCell clinicValue = new PdfPCell(new Phrase("Emirates \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont));

clinicValue.Border = Rectangle.NO\_BORDER;

clinicTable.AddCell(clinicValue);

document.Add(clinicTable);

document.Add(new Paragraph(" ", normalFont)); // Spacing

// Patient Details Table

PdfPTable patientTable = new PdfPTable(6);

patientTable.WidthPercentage = 100;

patientTable.SetWidths(new float[] { 2f, 3f, 1f, 1f, 1f, 1f });

// Row 1

patientTable.AddCell(new PdfPCell(new Phrase("Card Holder's Name", normalFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("Age", normalFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("Sex : M ☐ F ☐", normalFont)) { Border = Rectangle.NO\_BORDER, Colspan = 2 });

document.Add(patientTable);

document.Add(new Paragraph(" ", normalFont));

// Contact Details Table

PdfPTable contactTable = new PdfPTable(4);

contactTable.WidthPercentage = 100;

contactTable.SetWidths(new float[] { 2f, 3f, 2f, 3f });

contactTable.AddCell(new PdfPCell(new Phrase("Card Holder's Tel No", normalFont)) { Border = Rectangle.NO\_BORDER });

contactTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

contactTable.AddCell(new PdfPCell(new Phrase("Mobile No", normalFont)) { Border = Rectangle.NO\_BORDER });

contactTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

document.Add(contactTable);

document.Add(new Paragraph(" ", normalFont));

// Insurance Details Table

PdfPTable insTable = new PdfPTable(4);

insTable.WidthPercentage = 100;

insTable.SetWidths(new float[] { 2f, 3f, 2f, 3f });

insTable.AddCell(new PdfPCell(new Phrase("Ins. Card No", normalFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("Valid up to", normalFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_/\_\_\_\_/\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

document.Add(insTable);

document.Add(new Paragraph(" ", normalFont));

// Company Details Table

PdfPTable companyTable = new PdfPTable(6);

companyTable.WidthPercentage = 100;

companyTable.SetWidths(new float[] { 2f, 3f, 2f, 2f, 2f, 2f });

companyTable.AddCell(new PdfPCell(new Phrase("Company Name", normalFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("Employee No", normalFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("Nationality", normalFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

document.Add(companyTable);

document.Add(new Paragraph(" ", normalFont));

// Authorization text

Paragraph authText = new Paragraph(

"I hereby authorize the physician, Hospital or pharmacy to file a claim for medical services on my behalf and I confirm that the above-" +

"mentioned examination/Investigation/therapy is given to me by the doctor. I hereby authorize any Clinic, Physician, Pharmacy or any " +

"other person who has provided medical services to me to furnish any and all information with regard to any medical history, medical condition, " +

"or medical services and copies of all medical and Clinic records.", normalFont);

authText.SpacingAfter = 15f;

document.Add(authText);

// Signature Table

PdfPTable signatureTable = new PdfPTable(4);

signatureTable.WidthPercentage = 100;

signatureTable.SetWidths(new float[] { 1f, 2f, 2f, 3f });

signatureTable.AddCell(new PdfPCell(new Phrase("Date", normalFont)) { Border = Rectangle.NO\_BORDER });

signatureTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_/\_\_\_\_/\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

signatureTable.AddCell(new PdfPCell(new Phrase("Signature of the Patient", normalFont)) { Border = Rectangle.NO\_BORDER });

signatureTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont)) { Border = Rectangle.NO\_BORDER });

document.Add(signatureTable);

document.Add(new Paragraph(" ", normalFont));

// Service Type Checkboxes

PdfPTable serviceTable = new PdfPTable(4);

serviceTable.WidthPercentage = 100;

serviceTable.SetWidths(new float[] { 1f, 1f, 1f, 1f });

serviceTable.AddCell(new PdfPCell(new Phrase("☐ CLINIC", normalFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

serviceTable.AddCell(new PdfPCell(new Phrase("☐ PHARMACY", normalFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

serviceTable.AddCell(new PdfPCell(new Phrase("☐ DIAGNOSTIC CENTRE", normalFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

serviceTable.AddCell(new PdfPCell(new Phrase("☐ HOSPITAL OR OTHER", normalFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

document.Add(serviceTable);

Paragraph kindlyTick = new Paragraph("Kindly tick whichever is applicable", smallFont);

kindlyTick.Alignment = Element.ALIGN\_CENTER;

kindlyTick.SpacingAfter = 15f;

document.Add(kindlyTick);

// Pharmaceuticals Section

Paragraph pharmaTitle = new Paragraph("Pharmaceuticals (to be filled by treating doctor only)", boldFont);

pharmaTitle.SpacingAfter = 10f;

document.Add(pharmaTitle);

// Pharmaceuticals Table

PdfPTable pharmaTable = new PdfPTable(5);

pharmaTable.WidthPercentage = 100;

pharmaTable.SetWidths(new float[] { 3f, 2f, 2f, 2f, 2f });

// Header row

pharmaTable.AddCell(new PdfPCell(new Phrase("Trade Name", boldFont)) { HorizontalAlignment = Element.ALIGN\_CENTER });

pharmaTable.AddCell(new PdfPCell(new Phrase("Dose", boldFont)) { HorizontalAlignment = Element.ALIGN\_CENTER });

pharmaTable.AddCell(new PdfPCell(new Phrase("Total Duration", boldFont)) { HorizontalAlignment = Element.ALIGN\_CENTER });

pharmaTable.AddCell(new PdfPCell(new Phrase("Quantity", boldFont)) { HorizontalAlignment = Element.ALIGN\_CENTER });

pharmaTable.AddCell(new PdfPCell(new Phrase("Price", boldFont)) { HorizontalAlignment = Element.ALIGN\_CENTER });

// Add empty rows

for (int i = 1; i <= 4; i++)

{

pharmaTable.AddCell(new PdfPCell(new Phrase($"{i})", normalFont)) { MinimumHeight = 25f });

pharmaTable.AddCell(new PdfPCell(new Phrase("", normalFont)) { MinimumHeight = 25f });

pharmaTable.AddCell(new PdfPCell(new Phrase("", normalFont)) { MinimumHeight = 25f });

pharmaTable.AddCell(new PdfPCell(new Phrase("", normalFont)) { MinimumHeight = 25f });

pharmaTable.AddCell(new PdfPCell(new Phrase("", normalFont)) { MinimumHeight = 25f });

}

// Total row

pharmaTable.AddCell(new PdfPCell(new Phrase("Total", boldFont)) { Colspan = 4, HorizontalAlignment = Element.ALIGN\_RIGHT });

pharmaTable.AddCell(new PdfPCell(new Phrase("", normalFont)));

document.Add(pharmaTable);

Paragraph exclusions = new Paragraph("Please apply general exclusions", smallFont);

exclusions.SpacingAfter = 15f;

document.Add(exclusions);

// Clinical Details Section

Paragraph clinicalTitle = new Paragraph("Clinical Details:", boldFont);

clinicalTitle.SpacingAfter = 10f;

document.Add(clinicalTitle);

// Vital Signs Table

PdfPTable vitalsTable = new PdfPTable(6);

vitalsTable.WidthPercentage = 100;

vitalsTable.SetWidths(new float[] { 1f, 2f, 1f, 2f, 1f, 2f });

vitalsTable.AddCell(new PdfPCell(new Phrase("Temp", normalFont)) { Border = Rectangle.NO\_BORDER });

vitalsTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_°C", normalFont)) { Border = Rectangle.NO\_BORDER });

vitalsTable.AddCell(new PdfPCell(new Phrase("B.P.", normalFont)) { Border = Rectangle.NO\_BORDER });

vitalsTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_mmHg", normalFont)) { Border = Rectangle.NO\_BORDER });

vitalsTable.AddCell(new PdfPCell(new Phrase("Pulse", normalFont)) { Border = Rectangle.NO\_BORDER });

vitalsTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_/min", normalFont)) { Border = Rectangle.NO\_BORDER });

document.Add(vitalsTable);

document.Add(new Paragraph(" ", normalFont));

// Signs & Symptoms

Paragraph signsSymptoms = new Paragraph("Sign & Symptoms \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont);

signsSymptoms.SpacingAfter = 10f;

document.Add(signsSymptoms);

// Date of onset

Paragraph onsetDate = new Paragraph("Date of onset of illness:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont);

onsetDate.SpacingAfter = 10f;

document.Add(onsetDate);

// Visit Type Checkboxes

PdfPTable visitTable = new PdfPTable(4);

visitTable.WidthPercentage = 100;

visitTable.SetWidths(new float[] { 1f, 1f, 1f, 1f });

visitTable.AddCell(new PdfPCell(new Phrase("☐ Emergency", normalFont)) { Border = Rectangle.NO\_BORDER });

visitTable.AddCell(new PdfPCell(new Phrase("☐ Work related", normalFont)) { Border = Rectangle.NO\_BORDER });

visitTable.AddCell(new PdfPCell(new Phrase("☐ New visit", normalFont)) { Border = Rectangle.NO\_BORDER });

visitTable.AddCell(new PdfPCell(new Phrase("☐ Follow up visit", normalFont)) { Border = Rectangle.NO\_BORDER });

document.Add(visitTable);

document.Add(new Paragraph(" ", normalFont));

// Diagnosis

Paragraph diagnosis = new Paragraph("Diagnosis \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont);

diagnosis.SpacingAfter = 5f;

document.Add(diagnosis);

// Additional diagnosis lines

document.Add(new Paragraph("………………………..…………………………………………………………………………………", normalFont));

document.Add(new Paragraph("…………", normalFont));

document.Add(new Paragraph("……………………………………………………………………………………………..", normalFont));

document.Add(new Paragraph(" ", normalFont));

// Management Plan

Paragraph managementTitle = new Paragraph("Management plan (Services inside the clinic including injections and investigations)", boldFont);

managementTitle.SpacingAfter = 10f;

document.Add(managementTitle);

for (int i = 1; i <= 4; i++)

{

Paragraph managementItem = new Paragraph($"{i}) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont);

document.Add(managementItem);

}

document.Add(new Paragraph(" ", normalFont));

// Doctor's signature

Paragraph doctorSignature = new Paragraph("Doctor's Name and signature with seal: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", normalFont);

doctorSignature.SpacingAfter = 15f;

document.Add(doctorSignature);

// Diagnostic Procedures

Paragraph diagnosticProcedures = new Paragraph("Diagnostic Procedures referred outside:", normalFont);

diagnosticProcedures.SpacingAfter = 15f;

document.Add(diagnosticProcedures);

// Footer

Paragraph annexure = new Paragraph("ANNEXURE V", boldFont);

annexure.Alignment = Element.ALIGN\_CENTER;

annexure.SpacingAfter = 10f;

document.Add(annexure);

Paragraph insuranceCard = new Paragraph("Affix copy of front side of Insurance card", normalFont);

insuranceCard.Alignment = Element.ALIGN\_CENTER;

insuranceCard.SpacingAfter = 20f;

document.Add(insuranceCard);

// Company Footer

Paragraph companyFooter = new Paragraph("F M C NETWORK UAE", boldFont);

companyFooter.Alignment = Element.ALIGN\_CENTER;

companyFooter.SpacingAfter = 5f;

document.Add(companyFooter);

Paragraph contactInfo = new Paragraph(

"P. O. BOX: 50430, DUBAI, Tel – 04 3871900, Fax – 04 3977842\n" +

"Email – approval@fmchealthcare.ae Helpline Number: 600-565691", smallFont);

contactInfo.Alignment = Element.ALIGN\_CENTER;

document.Add(contactInfo);

document.Close();

// Send PDF to browser

Response.ContentType = "application/pdf";

Response.AddHeader("content-disposition", "attachment;filename=MedicalClaimForm.pdf");

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.BinaryWrite(stream.ToArray());

Response.End();

}

catch (Exception ex)

{

// Handle exceptions

Response.Write("Error generating PDF: " + ex.Message);

}

finally

{

if (document.IsOpen())

document.Close();

stream.Close();

}

}

}

// Model class for form data (if you want to populate from database or form inputs)

public class MedicalClaimData

{

public string Date { get; set; }

public string ClinicName { get; set; }

public string CardHolderName { get; set; }

public int Age { get; set; }

public string Sex { get; set; }

public string TelNo { get; set; }

public string MobileNo { get; set; }

public string InsCardNo { get; set; }

public string ValidUpTo { get; set; }

public string CompanyName { get; set; }

public string EmployeeNo { get; set; }

public string Nationality { get; set; }

public string SignatureDate { get; set; }

public string ServiceType { get; set; }

public string Temperature { get; set; }

public string BloodPressure { get; set; }

public string Pulse { get; set; }

public string SignsSymptoms { get; set; }

public string OnsetDate { get; set; }

public string VisitType { get; set; }

public string Diagnosis { get; set; }

public string ManagementPlan1 { get; set; }

public string ManagementPlan2 { get; set; }

public string ManagementPlan3 { get; set; }

public string ManagementPlan4 { get; set; }

public string DoctorName { get; set; }

public string DiagnosticProcedures { get; set; }

// Pharmaceutical items

public List<PharmaceuticalItem> PharmaceuticalItems { get; set; } = new List<PharmaceuticalItem>();

}

public class PharmaceuticalItem

{

public string TradeName { get; set; }

public string Dose { get; set; }

public string Duration { get; set; }

public string Quantity { get; set; }

public decimal Price { get; set; }

}

}

using System;

using System.IO;

using System.Web;

using System.Collections.Generic;

using iTextSharp.text;

using iTextSharp.text.pdf;

namespace MedicalClaimFormGenerator

{

public partial class GenerateClaimForm : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

GenerateMedicalClaimForm();

}

}

private void GenerateMedicalClaimForm()

{

Document document = new Document(PageSize.A4, 25, 25, 25, 25);

MemoryStream stream = new MemoryStream();

try

{

PdfWriter writer = PdfWriter.GetInstance(document, stream);

document.Open();

// Define exact fonts to match original

Font titleFont = FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 14, BaseColor.BLACK);

Font labelFont = FontFactory.GetFont(FontFactory.HELVETICA, 9, BaseColor.BLACK);

Font boldFont = FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 9, BaseColor.BLACK);

Font smallFont = FontFactory.GetFont(FontFactory.HELVETICA, 8, BaseColor.BLACK);

Font tinyFont = FontFactory.GetFont(FontFactory.HELVETICA, 7, BaseColor.BLACK);

// Title

Paragraph title = new Paragraph("Medical Expenses Claim form", titleFont);

title.Alignment = Element.ALIGN\_LEFT;

title.SpacingAfter = 8f;

document.Add(title);

// Date section

PdfPTable dateTable = new PdfPTable(new float[] { 60f, 40f, 200f, 200f });

dateTable.WidthPercentage = 100;

dateTable.DefaultCell.Border = Rectangle.NO\_BORDER;

dateTable.DefaultCell.PaddingBottom = 2f;

dateTable.AddCell(new PdfPCell(new Phrase("Date :", labelFont)) { Border = Rectangle.NO\_BORDER });

dateTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

dateTable.AddCell(new PdfPCell(new Phrase("/ \_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

dateTable.AddCell(new PdfPCell(new Phrase("/ \_\_\_\_\_\_\_\_\_\_.", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(dateTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 4))); // Small spacing

// Clinic Name section

PdfPTable clinicTable = new PdfPTable(new float[] { 100f, 100f, 300f });

clinicTable.WidthPercentage = 100;

clinicTable.DefaultCell.Border = Rectangle.NO\_BORDER;

clinicTable.DefaultCell.PaddingBottom = 2f;

clinicTable.AddCell(new PdfPCell(new Phrase("Clinic Name", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicTable.AddCell(new PdfPCell(new Phrase("Emirates", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(clinicTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 4)));

// Patient details section

PdfPTable patientTable = new PdfPTable(new float[] { 120f, 220f, 35f, 60f, 40f, 25f, 25f, 25f });

patientTable.WidthPercentage = 100;

patientTable.DefaultCell.Border = Rectangle.NO\_BORDER;

patientTable.DefaultCell.PaddingBottom = 2f;

patientTable.AddCell(new PdfPCell(new Phrase("Card Holder's Name", labelFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("Age", labelFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("Sex :", labelFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("M", labelFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER });

patientTable.AddCell(new PdfPCell(new Phrase("F", labelFont)) { Border = Rectangle.NO\_BORDER });

// Add checkbox for F

PdfPCell femaleCheckbox = new PdfPCell(new Phrase("☐", labelFont));

femaleCheckbox.Border = Rectangle.NO\_BORDER;

patientTable.AddCell(femaleCheckbox);

document.Add(patientTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 4)));

// Contact details section

PdfPTable contactTable = new PdfPTable(new float[] { 120f, 180f, 80f, 170f });

contactTable.WidthPercentage = 100;

contactTable.DefaultCell.Border = Rectangle.NO\_BORDER;

contactTable.DefaultCell.PaddingBottom = 2f;

contactTable.AddCell(new PdfPCell(new Phrase("Card Holder's Tel No", labelFont)) { Border = Rectangle.NO\_BORDER });

contactTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

contactTable.AddCell(new PdfPCell(new Phrase("Mobile No", labelFont)) { Border = Rectangle.NO\_BORDER });

contactTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(contactTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 4)));

// Insurance details section

PdfPTable insTable = new PdfPTable(new float[] { 80f, 200f, 80f, 40f, 40f, 110f });

insTable.WidthPercentage = 100;

insTable.DefaultCell.Border = Rectangle.NO\_BORDER;

insTable.DefaultCell.PaddingBottom = 2f;

insTable.AddCell(new PdfPCell(new Phrase("Ins. Card No", labelFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("Valid up to", labelFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("/ \_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

insTable.AddCell(new PdfPCell(new Phrase("/ \_\_\_\_\_\_\_\_\_\_.", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(insTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 4)));

// Company details section

PdfPTable companyTable = new PdfPTable(new float[] { 100f, 180f, 80f, 80f, 80f, 80f });

companyTable.WidthPercentage = 100;

companyTable.DefaultCell.Border = Rectangle.NO\_BORDER;

companyTable.DefaultCell.PaddingBottom = 2f;

companyTable.AddCell(new PdfPCell(new Phrase("Company Name", labelFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("Employee No", labelFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("Nationality", labelFont)) { Border = Rectangle.NO\_BORDER });

companyTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_.", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(companyTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 8)));

// Authorization paragraph

Paragraph authPara = new Paragraph();

authPara.Add(new Phrase("I hereby authorize the physician, Hospital or pharmacy to file a claim for medical services on my behalf and I confirm that the above-", labelFont));

authPara.Add(new Phrase("\nmentioned examination/Investigation/therapy is given to me by the doctor. I hereby authorize any Clinic, Physician, Pharmacy or any", labelFont));

authPara.Add(new Phrase("\nother person who has provided medical services to me to furnish any and all information with regard to any medical history, medical condition,", labelFont));

authPara.Add(new Phrase("\nor medical services and copies of all medical and Clinic records.", labelFont));

authPara.SpacingAfter = 8f;

document.Add(authPara);

// Signature section

PdfPTable sigTable = new PdfPTable(new float[] { 40f, 40f, 40f, 150f, 200f });

sigTable.WidthPercentage = 100;

sigTable.DefaultCell.Border = Rectangle.NO\_BORDER;

sigTable.DefaultCell.PaddingBottom = 2f;

sigTable.AddCell(new PdfPCell(new Phrase("Date", labelFont)) { Border = Rectangle.NO\_BORDER });

sigTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

sigTable.AddCell(new PdfPCell(new Phrase("/ \_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

sigTable.AddCell(new PdfPCell(new Phrase("/ \_\_\_\_ Signature of the Patient", labelFont)) { Border = Rectangle.NO\_BORDER });

sigTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(sigTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 8)));

// Service type checkboxes with exact spacing

PdfPTable serviceTypeTable = new PdfPTable(new float[] { 20f, 80f, 20f, 80f, 20f, 120f, 20f, 160f });

serviceTypeTable.WidthPercentage = 100;

serviceTypeTable.DefaultCell.Border = Rectangle.NO\_BORDER;

serviceTypeTable.DefaultCell.PaddingBottom = 2f;

serviceTypeTable.DefaultCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

serviceTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

serviceTypeTable.AddCell(new PdfPCell(new Phrase("CLINIC", labelFont)) { Border = Rectangle.NO\_BORDER });

serviceTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

serviceTypeTable.AddCell(new PdfPCell(new Phrase("PHARMACY", labelFont)) { Border = Rectangle.NO\_BORDER });

serviceTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

serviceTypeTable.AddCell(new PdfPCell(new Phrase("DIAGNOSTIC CENTRE", labelFont)) { Border = Rectangle.NO\_BORDER });

serviceTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

serviceTypeTable.AddCell(new PdfPCell(new Phrase("HOSPITAL OR OTHER", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(serviceTypeTable);

// "Kindly tick" text

Paragraph kindlyTick = new Paragraph("Kindly tick whichever is applicable", smallFont);

kindlyTick.Alignment = Element.ALIGN\_CENTER;

kindlyTick.SpacingAfter = 8f;

document.Add(kindlyTick);

// Pharmaceuticals section header

Paragraph pharmaHeader = new Paragraph("Pharmaceuticals (to be filled by treating doctor only)", boldFont);

pharmaHeader.SpacingAfter = 6f;

document.Add(pharmaHeader);

// Pharmaceuticals table with exact structure

PdfPTable pharmaMainTable = new PdfPTable(1);

pharmaMainTable.WidthPercentage = 100;

// Header section

PdfPTable pharmaHeaderTable = new PdfPTable(new float[] { 200f, 120f, 120f, 80f, 80f });

pharmaHeaderTable.WidthPercentage = 100;

// Add header cells with borders

PdfPCell tradeNameHeader = new PdfPCell(new Phrase("Trade Name", boldFont));

tradeNameHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

tradeNameHeader.BackgroundColor = BaseColor.LIGHT\_GRAY;

pharmaHeaderTable.AddCell(tradeNameHeader);

PdfPCell doseHeader = new PdfPCell(new Phrase("Dose", boldFont));

doseHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

doseHeader.BackgroundColor = BaseColor.LIGHT\_GRAY;

pharmaHeaderTable.AddCell(doseHeader);

PdfPCell durationHeader = new PdfPCell(new Phrase("Total Duration", boldFont));

durationHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

durationHeader.BackgroundColor = BaseColor.LIGHT\_GRAY;

pharmaHeaderTable.AddCell(durationHeader);

// Second header row for pharmacy section

PdfPTable pharmacyHeaderTable = new PdfPTable(new float[] { 440f, 80f, 80f });

pharmacyHeaderTable.WidthPercentage = 100;

PdfPCell pharmacyLabel = new PdfPCell(new Phrase("(To be filled by the pharmacy)", tinyFont));

pharmacyLabel.HorizontalAlignment = Element.ALIGN\_CENTER;

pharmacyLabel.BackgroundColor = BaseColor.LIGHT\_GRAY;

pharmacyHeaderTable.AddCell(pharmacyLabel);

PdfPCell quantityHeader = new PdfPCell(new Phrase("Quantity", boldFont));

quantityHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

quantityHeader.BackgroundColor = BaseColor.LIGHT\_GRAY;

pharmacyHeaderTable.AddCell(quantityHeader);

PdfPCell priceHeader = new PdfPCell(new Phrase("Price", boldFont));

priceHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

priceHeader.BackgroundColor = BaseColor.LIGHT\_GRAY;

pharmacyHeaderTable.AddCell(priceHeader);

document.Add(pharmaHeaderTable);

document.Add(pharmacyHeaderTable);

// Pharmaceutical rows

PdfPTable pharmaDataTable = new PdfPTable(new float[] { 30f, 170f, 120f, 120f, 80f, 80f });

pharmaDataTable.WidthPercentage = 100;

for (int i = 1; i <= 4; i++)

{

pharmaDataTable.AddCell(new PdfPCell(new Phrase($"{i})", labelFont)) { MinimumHeight = 20f, VerticalAlignment = Element.ALIGN\_MIDDLE });

pharmaDataTable.AddCell(new PdfPCell(new Phrase("", labelFont)) { MinimumHeight = 20f });

pharmaDataTable.AddCell(new PdfPCell(new Phrase("", labelFont)) { MinimumHeight = 20f });

pharmaDataTable.AddCell(new PdfPCell(new Phrase("", labelFont)) { MinimumHeight = 20f });

pharmaDataTable.AddCell(new PdfPCell(new Phrase("", labelFont)) { MinimumHeight = 20f });

pharmaDataTable.AddCell(new PdfPCell(new Phrase("", labelFont)) { MinimumHeight = 20f });

}

document.Add(pharmaDataTable);

// Total row and exclusions

PdfPTable totalTable = new PdfPTable(new float[] { 520f, 80f });

totalTable.WidthPercentage = 100;

PdfPCell totalLabel = new PdfPCell(new Phrase("Please apply general exclusions Total", labelFont));

totalLabel.HorizontalAlignment = Element.ALIGN\_RIGHT;

totalLabel.VerticalAlignment = Element.ALIGN\_MIDDLE;

totalLabel.Border = Rectangle.NO\_BORDER;

totalTable.AddCell(totalLabel);

totalTable.AddCell(new PdfPCell(new Phrase("", labelFont)) { MinimumHeight = 20f });

document.Add(totalTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 8)));

// Footer company info

PdfPTable footerTable = new PdfPTable(1);

footerTable.WidthPercentage = 100;

PdfPCell footerCell = new PdfPCell();

footerCell.Border = Rectangle.BOX;

footerCell.Padding = 5f;

Paragraph fmcNetwork = new Paragraph("F M C NETWORK UAE", boldFont);

fmcNetwork.Alignment = Element.ALIGN\_CENTER;

fmcNetwork.SpacingAfter = 4f;

Paragraph contactDetails = new Paragraph();

contactDetails.Add(new Phrase("P. O. BOX: 50430, DUBAI, Tel – 04 3871900, Fax – 04 3977842", smallFont));

contactDetails.Add(new Phrase("\nEmail – approval@fmchealthcare.ae Helpline Number: 600-565691", smallFont));

contactDetails.Alignment = Element.ALIGN\_CENTER;

footerCell.AddElement(fmcNetwork);

footerCell.AddElement(contactDetails);

footerTable.AddCell(footerCell);

document.Add(footerTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 8)));

// Clinical Details Section - Page 2 content

document.NewPage();

// Clinical Details Header

PdfPTable clinicalHeaderTable = new PdfPTable(new float[] { 120f, 60f, 80f, 80f, 80f, 60f, 120f });

clinicalHeaderTable.WidthPercentage = 100;

clinicalHeaderTable.DefaultCell.Border = Rectangle.NO\_BORDER;

clinicalHeaderTable.DefaultCell.PaddingBottom = 3f;

clinicalHeaderTable.AddCell(new PdfPCell(new Phrase("Clinical Details: Temp", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeaderTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_°", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeaderTable.AddCell(new PdfPCell(new Phrase("C B.P.", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeaderTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeaderTable.AddCell(new PdfPCell(new Phrase("mmHg Pulse", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeaderTable.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeaderTable.AddCell(new PdfPCell(new Phrase("/ min", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(clinicalHeaderTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 6)));

// Signs & Symptoms

Paragraph signsSymptoms = new Paragraph("Sign & Symptoms \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", labelFont);

signsSymptoms.SpacingAfter = 6f;

document.Add(signsSymptoms);

// Date of onset

Paragraph onsetDate = new Paragraph("Date of onset of illness:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", labelFont);

onsetDate.SpacingAfter = 6f;

document.Add(onsetDate);

// Visit type checkboxes

PdfPTable visitTypeTable = new PdfPTable(new float[] { 20f, 80f, 20f, 100f, 20f, 80f, 20f, 140f });

visitTypeTable.WidthPercentage = 100;

visitTypeTable.DefaultCell.Border = Rectangle.NO\_BORDER;

visitTypeTable.DefaultCell.PaddingBottom = 2f;

visitTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Emergency", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Work related", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("New visit", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("☐", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Follow up visit", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(visitTypeTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 6)));

// Diagnosis

Paragraph diagnosis1 = new Paragraph("Diagnosis \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", labelFont);

document.Add(diagnosis1);

Paragraph diagnosis2 = new Paragraph("………………………..…………………………………………………………………………………", labelFont);

document.Add(diagnosis2);

Paragraph diagnosis3 = new Paragraph("…………", labelFont);

document.Add(diagnosis3);

Paragraph diagnosis4 = new Paragraph("…………………………………………………………………………………………….", labelFont);

diagnosis4.SpacingAfter = 8f;

document.Add(diagnosis4);

// Management plan

Paragraph managementTitle = new Paragraph("Management plan (Services inside the clinic including injections and investigations)", labelFont);

managementTitle.SpacingAfter = 6f;

document.Add(managementTitle);

for (int i = 1; i <= 4; i++)

{

Paragraph managementItem = new Paragraph($"{i}) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", labelFont);

document.Add(managementItem);

}

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 8)));

// Doctor's signature

Paragraph doctorSig = new Paragraph("Doctor's Name and signature with seal: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", labelFont);

doctorSig.SpacingAfter = 8f;

document.Add(doctorSig);

// Diagnostic Procedures

Paragraph diagnosticProc = new Paragraph("Diagnostic Procedures referred outside:", labelFont);

diagnosticProc.SpacingAfter = 20f;

document.Add(diagnosticProc);

// ANNEXURE V

Paragraph annexure = new Paragraph("ANNEXURE V", boldFont);

annexure.Alignment = Element.ALIGN\_CENTER;

annexure.SpacingAfter = 8f;

document.Add(annexure);

// Insurance card instruction

Paragraph insuranceInstruction = new Paragraph("Affix copy of front side of Insurance card", labelFont);

insuranceInstruction.Alignment = Element.ALIGN\_CENTER;

document.Add(insuranceInstruction);

document.Close();

// Send PDF to browser

Response.ContentType = "application/pdf";

Response.AddHeader("content-disposition", "attachment;filename=MedicalExpensesClaimForm.pdf");

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.BinaryWrite(stream.ToArray());

Response.End();

}

catch (Exception ex)

{

Response.Write("Error generating PDF: " + ex.Message);

}

finally

{

if (document.IsOpen())

document.Close();

stream?.Close();

}

}

}

// Enhanced data models for form population

public class MedicalClaimFormData

{

public string Date { get; set; }

public string ClinicName { get; set; } = "Emirates";

public string CardHolderName { get; set; }

public int? Age { get; set; }

public string Sex { get; set; } // "M" or "F"

public string TelNo { get; set; }

public string MobileNo { get; set; }

public string InsCardNo { get; set; }

public string ValidUpTo { get; set; }

public string CompanyName { get; set; }

public string EmployeeNo { get; set; }

public string Nationality { get; set; }

public string SignatureDate { get; set; }

public List<string> ServiceTypes { get; set; } = new List<string>(); // CLINIC, PHARMACY, DIAGNOSTIC CENTRE, HOSPITAL OR OTHER

public List<PharmaceuticalItem> Pharmaceuticals { get; set; } = new List<PharmaceuticalItem>();

public string Temperature { get; set; }

public string BloodPressure { get; set; }

public string Pulse { get; set; }

public string SignsSymptoms { get; set; }

public string OnsetDate { get; set; }

public List<string> VisitTypes { get; set; } = new List<string>(); // Emergency, Work related, New visit, Follow up visit

public string Diagnosis { get; set; }

public List<string> ManagementPlan { get; set; } = new List<string>();

public string DoctorNameAndSignature { get; set; }

public string DiagnosticProcedures { get; set; }

}

public class PharmaceuticalItem

{

public string TradeName { get; set; }

public string Dose { get; set; }

public string TotalDuration { get; set; }

public string Quantity { get; set; }

public decimal? Price { get; set; }

}

}

==========================================Database version of report =====================================================================================

using System;

using System.IO;

using System.Web;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

using iTextSharp.text;

using iTextSharp.text.pdf;

using System.Linq;

namespace MedicalClaimFormGenerator

{

public partial class GenerateClaimForm : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

// Example: Get form data from database or form submission

var formData = GetFormDataFromDatabase(Request.QueryString["claimId"]);

GenerateMedicalClaimForm(formData);

}

}

private MedicalClaimFormData GetFormDataFromDatabase(string claimId)

{

var formData = new MedicalClaimFormData();

// Sample database integration

if (!string.IsNullOrEmpty(claimId))

{

string connectionString = ConfigurationManager.ConnectionStrings["MedicalDB"].ConnectionString;

using (SqlConnection conn = new SqlConnection(connectionString))

{

string sql = @"

SELECT c.\*, p.Name, p.Age, p.Sex, p.TelNo, p.MobileNo, p.InsCardNo,

p.ValidUpTo, p.CompanyName, p.EmployeeNo, p.Nationality

FROM Claims c

INNER JOIN Patients p ON c.PatientId = p.Id

WHERE c.ClaimId = @ClaimId";

using (SqlCommand cmd = new SqlCommand(sql, conn))

{

cmd.Parameters.AddWithValue("@ClaimId", claimId);

conn.Open();

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

formData.Date = reader["ClaimDate"].ToString();

formData.CardHolderName = reader["Name"].ToString();

formData.Age = Convert.ToInt32(reader["Age"]);

formData.Sex = reader["Sex"].ToString();

formData.TelNo = reader["TelNo"].ToString();

formData.MobileNo = reader["MobileNo"].ToString();

formData.InsCardNo = reader["InsCardNo"].ToString();

formData.ValidUpTo = reader["ValidUpTo"].ToString();

formData.CompanyName = reader["CompanyName"].ToString();

formData.EmployeeNo = reader["EmployeeNo"].ToString();

formData.Nationality = reader["Nationality"].ToString();

// Add more fields as needed

}

}

}

}

}

else

{

// Sample dummy data for demonstration

formData = new MedicalClaimFormData

{

Date = DateTime.Now.ToString("dd/MM/yyyy"),

ClinicName = "Emirates Medical Center",

CardHolderName = "AHMED AL MAHMOUD",

Age = 35,

Sex = "M",

TelNo = "04-3456789",

MobileNo = "050-1234567",

InsCardNo = "EMC123456789",

ValidUpTo = "31/12/2025",

CompanyName = "ABC TRADING LLC",

EmployeeNo = "EMP001",

Nationality = "UAE",

Temperature = "37.2",

BloodPressure = "120/80",

Pulse = "72",

SignsSymptoms = "Fever, headache, body ache",

OnsetDate = "15/01/2025",

Diagnosis = "Viral fever with associated symptoms",

DoctorNameAndSignature = "Dr. Sarah Ahmed"

};

formData.ServiceTypes.Add("CLINIC");

formData.VisitTypes.Add("New visit");

formData.Pharmaceuticals.Add(new PharmaceuticalItem

{

TradeName = "Paracetamol 500mg",

Dose = "500mg",

TotalDuration = "5 days",

Quantity = "15 tablets",

Price = 25.50m

});

formData.ManagementPlan.Add("Complete blood count test");

formData.ManagementPlan.Add("Paracetamol 500mg TID");

formData.ManagementPlan.Add("Rest and adequate fluid intake");

}

return formData;

}

private void GenerateMedicalClaimForm(MedicalClaimFormData formData)

{

Document document = new Document(PageSize.A4, 20, 20, 20, 20);

MemoryStream stream = new MemoryStream();

try

{

PdfWriter writer = PdfWriter.GetInstance(document, stream);

document.Open();

// Create precise fonts matching the original document

BaseFont helvetica = BaseFont.CreateFont(BaseFont.HELVETICA, BaseFont.CP1252, BaseFont.NOT\_EMBEDDED);

BaseFont helveticaBold = BaseFont.CreateFont(BaseFont.HELVETICA\_BOLD, BaseFont.CP1252, BaseFont.NOT\_EMBEDDED);

BaseFont times = BaseFont.CreateFont(BaseFont.TIMES\_ROMAN, BaseFont.CP1252, BaseFont.NOT\_EMBEDDED);

Font titleFont = new Font(helveticaBold, 14, Font.BOLD, BaseColor.BLACK);

Font headerFont = new Font(helveticaBold, 11, Font.BOLD, BaseColor.BLACK);

Font labelFont = new Font(helvetica, 9, Font.NORMAL, BaseColor.BLACK);

Font boldLabelFont = new Font(helveticaBold, 9, Font.BOLD, BaseColor.BLACK);

Font smallFont = new Font(helvetica, 8, Font.NORMAL, BaseColor.BLACK);

Font tinyFont = new Font(helvetica, 7, Font.NORMAL, BaseColor.BLACK);

// Add company logo (placeholder)

AddCompanyLogo(document, writer);

// Title section

Paragraph title = new Paragraph("Medical Expenses Claim form", titleFont);

title.Alignment = Element.ALIGN\_LEFT;

title.SpacingAfter = 10f;

document.Add(title);

// Create main form table to match exact layout

PdfPTable mainTable = new PdfPTable(1);

mainTable.WidthPercentage = 100;

mainTable.DefaultCell.Border = Rectangle.NO\_BORDER;

// Date row

PdfPTable dateRow = CreateDateRow(formData, labelFont);

document.Add(dateRow);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Clinic name row

PdfPTable clinicRow = CreateClinicRow(formData, labelFont);

document.Add(clinicRow);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Patient details section

PdfPTable patientSection = CreatePatientSection(formData, labelFont);

document.Add(patientSection);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Contact details section

PdfPTable contactSection = CreateContactSection(formData, labelFont);

document.Add(contactSection);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Insurance details section

PdfPTable insuranceSection = CreateInsuranceSection(formData, labelFont);

document.Add(insuranceSection);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Company details section

PdfPTable companySection = CreateCompanySection(formData, labelFont);

document.Add(companySection);

document.Add(new Paragraph(" ", new Font(helvetica, 8)));

// Authorization text - exact formatting

Paragraph authText = new Paragraph();

authText.Add(new Chunk("I hereby authorize the physician, Hospital or pharmacy to file a claim for medical services on my behalf and I confirm that the above-", labelFont));

authText.Add(new Chunk("\nmentioned examination/Investigation/therapy is given to me by the doctor. I hereby authorize any Clinic, Physician, Pharmacy or any", labelFont));

authText.Add(new Chunk("\nother person who has provided medical services to me to furnish any and all information with regard to any medical history, medical condition,", labelFont));

authText.Add(new Chunk("\nor medical services and copies of all medical and Clinic records.", labelFont));

authText.SpacingAfter = 10f;

document.Add(authText);

// Signature section

PdfPTable signatureSection = CreateSignatureSection(formData, labelFont);

document.Add(signatureSection);

document.Add(new Paragraph(" ", new Font(helvetica, 8)));

// Service type checkboxes with exact positioning

PdfPTable serviceTypeSection = CreateServiceTypeSection(formData, labelFont);

document.Add(serviceTypeSection);

// "Kindly tick" instruction

Paragraph kindlyTick = new Paragraph("Kindly tick whichever is applicable", smallFont);

kindlyTick.Alignment = Element.ALIGN\_CENTER;

kindlyTick.SpacingAfter = 10f;

document.Add(kindlyTick);

// Pharmaceuticals section with exact table structure

AddPharmaceuticalsSection(document, formData, headerFont, labelFont, smallFont, tinyFont);

// Add spacing before footer

document.Add(new Paragraph(" ", new Font(helvetica, 10)));

// Company footer with exact formatting

AddCompanyFooter(document, headerFont, smallFont);

// Start new page for clinical details

document.NewPage();

AddClinicalDetailsPage(document, formData, headerFont, labelFont, smallFont);

document.Close();

// Send PDF to browser

Response.ContentType = "application/pdf";

Response.AddHeader("content-disposition", "attachment;filename=Medical\_Expenses\_Claim\_Form.pdf");

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.BinaryWrite(stream.ToArray());

Response.Flush();

//Response.End();

}

catch (Exception ex)

{

Response.Write($"Error generating PDF: {ex.Message}");

}

finally

{

if (document.IsOpen())

document.Close();

stream?.Close();

}

}

private void AddCompanyLogo(Document document, PdfWriter writer)

{

try

{

// Create a simple placeholder logo using shapes

PdfContentByte cb = writer.DirectContent;

// Draw placeholder logo box

cb.Rectangle(document.PageSize.Width - 100, document.PageSize.Height - 80, 80, 60);

cb.SetColorStroke(BaseColor.BLUE);

cb.SetLineWidth(2f);

cb.Stroke();

// Add logo text

cb.BeginText();

cb.SetFontAndSize(BaseFont.CreateFont(BaseFont.HELVETICA\_BOLD, BaseFont.CP1252, BaseFont.NOT\_EMBEDDED), 10);

cb.SetTextMatrix(document.PageSize.Width - 95, document.PageSize.Height - 50);

cb.ShowText("FMC");

cb.SetTextMatrix(document.PageSize.Width - 95, document.PageSize.Height - 65);

cb.ShowText("LOGO");

cb.EndText();

}

catch (Exception)

{

// Logo creation failed, continue without logo

}

}

private PdfPTable CreateDateRow(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 50f, 30f, 20f, 30f, 20f, 350f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Date :", font)) { Border = Rectangle.NO\_BORDER });

if (!string.IsNullOrEmpty(formData.Date))

{

var dateParts = formData.Date.Split('/');

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 0 ? dateParts[0] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 1 ? dateParts[1] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase((dateParts.Length > 2 ? dateParts[2] : "\_\_\_\_\_\_\_\_") + ".", font)) { Border = Rectangle.NO\_BORDER });

}

else

{

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_.", font)) { Border = Rectangle.NO\_BORDER });

}

return table;

}

private PdfPTable CreateClinicRow(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 100f, 100f, 400f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.AddCell(new PdfPCell(new Phrase("Clinic Name", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Emirates", font)) { Border = Rectangle.NO\_BORDER });

string clinicName = !string.IsNullOrEmpty(formData.ClinicName) ?

formData.ClinicName.Replace("Emirates", "").Trim() :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

table.AddCell(new PdfPCell(new Phrase(clinicName + ".", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreatePatientSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 120f, 250f, 35f, 50f, 40f, 15f, 15f, 15f, 15f, 35f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Card Holder's Name", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.CardHolderName) ? formData.CardHolderName : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Age", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(formData.Age.HasValue ? formData.Age.Value.ToString() : "\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Sex :", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("M", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(formData.Sex == "M" ? "☑" : "☐", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(" F", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(formData.Sex == "F" ? "☑" : "☐", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("", font)) { Border = Rectangle.NO\_BORDER }); // Empty cell for spacing

return table;

}

private PdfPTable CreateContactSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 120f, 200f, 80f, 200f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Card Holder's Tel No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.TelNo) ? formData.TelNo : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Mobile No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase((!string.IsNullOrEmpty(formData.MobileNo) ? formData.MobileNo : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_") + ".", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreateInsuranceSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 80f, 200f, 80f, 240f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Ins. Card No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.InsCardNo) ? formData.InsCardNo : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Valid up to", font)) { Border = Rectangle.NO\_BORDER });

if (!string.IsNullOrEmpty(formData.ValidUpTo))

{

var validParts = formData.ValidUpTo.Split('/');

string validText = (validParts.Length > 0 ? validParts[0] : "\_\_\_\_") + " / " +

(validParts.Length > 1 ? validParts[1] : "\_\_\_\_") + " / " +

(validParts.Length > 2 ? validParts[2] : "\_\_\_\_\_\_\_\_\_\_") + ".";

table.AddCell(new PdfPCell(new Phrase(validText, font)) { Border = Rectangle.NO\_BORDER });

}

else

{

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_ / \_\_\_\_ / \_\_\_\_\_\_\_\_\_\_.", font)) { Border = Rectangle.NO\_BORDER });

}

return table;

}

private PdfPTable CreateCompanySection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 100f, 180f, 80f, 80f, 80f, 80f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Company Name", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.CompanyName) ? formData.CompanyName : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Employee No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.EmployeeNo) ? formData.EmployeeNo : "\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Nationality", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase((!string.IsNullOrEmpty(formData.Nationality) ? formData.Nationality : "\_\_\_\_\_\_\_\_\_\_") + ".", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreateSignatureSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 40f, 30f, 20f, 30f, 20f, 150f, 210f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Date", font)) { Border = Rectangle.NO\_BORDER });

if (!string.IsNullOrEmpty(formData.SignatureDate))

{

var dateParts = formData.SignatureDate.Split('/');

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 0 ? dateParts[0] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 1 ? dateParts[1] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

}

else

{

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

}

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_ Signature of the Patient", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreateServiceTypeSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 30f, 100f, 30f, 100f, 30f, 150f, 30f, 130f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 5f;

table.DefaultCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

string clinicCheck = formData.ServiceTypes.Contains("CLINIC") ? "☑" : "☐";

string pharmacyCheck = formData.ServiceTypes.Contains("PHARMACY") ? "☑" : "☐";

string diagnosticCheck = formData.ServiceTypes.Contains("DIAGNOSTIC CENTRE") ? "☑" : "☐";

string hospitalCheck = formData.ServiceTypes.Contains("HOSPITAL OR OTHER") ? "☑" : "☐";

table.AddCell(new PdfPCell(new Phrase(clinicCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("CLINIC", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(pharmacyCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("PHARMACY", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(diagnosticCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("DIAGNOSTIC CENTRE", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(hospitalCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("HOSPITAL", font)) { Border = Rectangle.NO\_BORDER });

// Second row for "OR OTHER"

for (int i = 0; i < 6; i++)

{

table.AddCell(new PdfPCell(new Phrase("", font)) { Border = Rectangle.NO\_BORDER });

}

table.AddCell(new PdfPCell(new Phrase("", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("OR OTHER", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private void AddPharmaceuticalsSection(Document document, MedicalClaimFormData formData, Font headerFont, Font labelFont, Font smallFont, Font tinyFont)

{

// Section header

Paragraph pharmaHeader = new Paragraph("Pharmaceuticals (to be filled by treating doctor only)", headerFont);

pharmaHeader.SpacingAfter = 6f;

document.Add(pharmaHeader);

// Main table structure

PdfPTable mainPharmaTable = new PdfPTable(new float[] { 200f, 120f, 120f, 80f, 80f });

mainPharmaTable.WidthPercentage = 100;

// Headers with gray background

PdfPCell tradeNameHeader = new PdfPCell(new Phrase("Trade Name", headerFont));

tradeNameHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

tradeNameHeader.BackgroundColor = new BaseColor(220, 220, 220);

tradeNameHeader.Padding = 5f;

mainPharmaTable.AddCell(tradeNameHeader);

PdfPCell doseHeader = new PdfPCell(new Phrase("Dose", headerFont));

doseHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

doseHeader.BackgroundColor = new BaseColor(220, 220, 220);

doseHeader.Padding = 5f;

mainPharmaTable.AddCell(doseHeader);

PdfPCell durationHeader = new PdfPCell(new Phrase("Total Duration", headerFont));

durationHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

durationHeader.BackgroundColor = new BaseColor(220, 220, 220);

durationHeader.Padding = 5f;

mainPharmaTable.AddCell(durationHeader);

// Second header row for pharmacy section

PdfPCell pharmacySection = new PdfPCell(new Phrase("(To be filled by the pharmacy)", tinyFont));

pharmacySection.HorizontalAlignment = Element.ALIGN\_CENTER;

pharmacySection.BackgroundColor = new BaseColor(220, 220, 220);

pharmacySection.Padding = 3f;

pharmacySection.Colspan = 3;

mainPharmaTable.AddCell(pharmacySection);

PdfPCell quantityHeader = new PdfPCell(new Phrase("Quantity", headerFont));

quantityHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

quantityHeader.BackgroundColor = new BaseColor(220, 220, 220);

quantityHeader.Padding = 5f;

mainPharmaTable.AddCell(quantityHeader);

PdfPCell priceHeader = new PdfPCell(new Phrase("Price", headerFont));

priceHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

priceHeader.BackgroundColor = new BaseColor(220, 220, 220);

priceHeader.Padding = 5f;

mainPharmaTable.AddCell(priceHeader);

// Data rows

for (int i = 0; i < 1; i++)

{

PharmaceuticalItem item = i < formData.Pharmaceuticals.Count ? formData.Pharmaceuticals[i] : null;

mainPharmaTable.AddCell(new PdfPCell(new Phrase($"{i + 1}) " + (item?.TradeName ?? ""), labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

mainPharmaTable.AddCell(new PdfPCell(new Phrase(item?.Dose ?? "", labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

mainPharmaTable.AddCell(new PdfPCell(new Phrase(item?.TotalDuration ?? "", labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

mainPharmaTable.AddCell(new PdfPCell(new Phrase(item?.Quantity ?? "", labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

mainPharmaTable.AddCell(new PdfPCell(new Phrase((bool)(item?.Price.HasValue) ? item.Price.Value.ToString("0.00") : "", labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

}

// Total row

decimal totalPrice = formData.Pharmaceuticals.Where(p => p.Price.HasValue).Sum(p => p.Price.Value);

PdfPCell exclusionsCell = new PdfPCell(new Phrase("Please apply general exclusions", smallFont));

exclusionsCell.Colspan = 3;

exclusionsCell.HorizontalAlignment = Element.ALIGN\_LEFT;

exclusionsCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

exclusionsCell.Padding = 5f;

mainPharmaTable.AddCell(exclusionsCell);

PdfPCell totalLabelCell = new PdfPCell(new Phrase("Total", headerFont));

totalLabelCell.HorizontalAlignment = Element.ALIGN\_RIGHT;

totalLabelCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

totalLabelCell.Padding = 5f;

mainPharmaTable.AddCell(totalLabelCell);

PdfPCell totalValueCell = new PdfPCell(new Phrase(totalPrice > 0 ? totalPrice.ToString("0.00") : "", labelFont));

totalValueCell.HorizontalAlignment = Element.ALIGN\_CENTER;

totalValueCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

totalValueCell.Padding = 5f;

mainPharmaTable.AddCell(totalValueCell);

document.Add(mainPharmaTable);

}

private void AddCompanyFooter(Document document, Font headerFont, Font smallFont)

{

// Company footer in bordered box

PdfPTable footerTable = new PdfPTable(1);

footerTable.WidthPercentage = 100;

footerTable.SpacingBefore = 15f;

PdfPCell footerCell = new PdfPCell();

footerCell.BorderWidth = 1f;

footerCell.BorderColor = BaseColor.BLACK;

footerCell.Padding = 8f;

// Company name

Paragraph companyName = new Paragraph("F M C NETWORK UAE", headerFont);

companyName.Alignment = Element.ALIGN\_CENTER;

companyName.SpacingAfter = 5f;

// Contact details

Paragraph contactDetails = new Paragraph();

contactDetails.Add(new Chunk("P. O. BOX: 50430, DUBAI, Tel – 04 3871900, Fax – 04 3977842", smallFont));

contactDetails.Add(new Chunk("\nEmail – approval@fmchealthcare.ae Helpline Number: 600-565691", smallFont));

contactDetails.Alignment = Element.ALIGN\_CENTER;

footerCell.AddElement(companyName);

footerCell.AddElement(contactDetails);

footerTable.AddCell(footerCell);

document.Add(footerTable);

}

private void AddClinicalDetailsPage(Document document, MedicalClaimFormData formData, Font headerFont, Font labelFont, Font smallFont)

{

// Clinical Details header with vitals

PdfPTable clinicalHeader = new PdfPTable(new float[] { 120f, 60f, 30f, 60f, 80f, 60f, 80f, 110f });

clinicalHeader.WidthPercentage = 100;

clinicalHeader.DefaultCell.Border = Rectangle.NO\_BORDER;

clinicalHeader.DefaultCell.PaddingBottom = 3f;

clinicalHeader.AddCell(new PdfPCell(new Phrase("Clinical Details: Temp", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.Temperature) ? formData.Temperature : "\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("°C", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("B.P.", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.BloodPressure) ? formData.BloodPressure : "\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("mmHg", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("Pulse.", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase((!string.IsNullOrEmpty(formData.Pulse) ? formData.Pulse : "\_\_\_\_\_\_") + " / min", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(clinicalHeader);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 6)));

// Signs & Symptoms

string signsText = !string.IsNullOrEmpty(formData.SignsSymptoms) ?

formData.SignsSymptoms :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph signsSymptoms = new Paragraph("Sign & Symptoms " + signsText + ".", labelFont);

signsSymptoms.SpacingAfter = 8f;

document.Add(signsSymptoms);

// Date of onset

string onsetText = !string.IsNullOrEmpty(formData.OnsetDate) ?

formData.OnsetDate :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph onsetDate = new Paragraph("Date of onset of illness:" + onsetText, labelFont);

onsetDate.SpacingAfter = 8f;

document.Add(onsetDate);

// Visit type checkboxes

PdfPTable visitTypeTable = new PdfPTable(new float[] { 30f, 100f, 30f, 120f, 30f, 100f, 30f, 160f });

visitTypeTable.WidthPercentage = 100;

visitTypeTable.DefaultCell.Border = Rectangle.NO\_BORDER;

visitTypeTable.DefaultCell.PaddingBottom = 3f;

string emergencyCheck = formData.VisitTypes.Contains("Emergency") ? "☑" : "☐";

string workCheck = formData.VisitTypes.Contains("Work related") ? "☑" : "☐";

string newCheck = formData.VisitTypes.Contains("New visit") ? "☑" : "☐";

string followupCheck = formData.VisitTypes.Contains("Follow up visit") ? "☑" : "☐";

visitTypeTable.AddCell(new PdfPCell(new Phrase(emergencyCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Emergency", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase(workCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Work related", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase(newCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("New visit", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase(followupCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Follow up visit", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(visitTypeTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 8)));

// Diagnosis section

string diagnosisText = !string.IsNullOrEmpty(formData.Diagnosis) ?

formData.Diagnosis :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph diagnosis1 = new Paragraph("Diagnosis " + diagnosisText + ".", labelFont);

document.Add(diagnosis1);

// Additional diagnosis lines

Paragraph diagnosis2 = new Paragraph("………………………..…………………………………………………………………………………", labelFont);

document.Add(diagnosis2);

Paragraph diagnosis3 = new Paragraph("…………", labelFont);

document.Add(diagnosis3);

Paragraph diagnosis4 = new Paragraph("………………………………………………………………………………………….", labelFont);

diagnosis4.SpacingAfter = 10f;

document.Add(diagnosis4);

// Management plan

Paragraph managementTitle = new Paragraph("Management plan (Services inside the clinic including injections and investigations)", labelFont);

managementTitle.SpacingAfter = 8f;

document.Add(managementTitle);

for (int i = 0; i < 4; i++)

{

string planText = i < formData.ManagementPlan.Count && !string.IsNullOrEmpty(formData.ManagementPlan[i]) ?

formData.ManagementPlan[i] :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph managementItem = new Paragraph($"{i + 1}) " + planText, labelFont);

document.Add(managementItem);

}

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 10)));

// Doctor's signature

string doctorText = !string.IsNullOrEmpty(formData.DoctorNameAndSignature) ?

formData.DoctorNameAndSignature :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph doctorSig = new Paragraph("Doctor's Name and signature with seal: " + doctorText + ".", labelFont);

doctorSig.SpacingAfter = 12f;

document.Add(doctorSig);

// Diagnostic Procedures

string diagnosticText = !string.IsNullOrEmpty(formData.DiagnosticProcedures) ?

formData.DiagnosticProcedures :

"";

Paragraph diagnosticProc = new Paragraph("Diagnostic Procedures referred outside: " + diagnosticText, labelFont);

diagnosticProc.SpacingAfter = 25f;

document.Add(diagnosticProc);

// ANNEXURE V section

Paragraph annexure = new Paragraph("ANNEXURE V", headerFont);

annexure.Alignment = Element.ALIGN\_CENTER;

annexure.SpacingAfter = 10f;

document.Add(annexure);

// Insurance card instruction

Paragraph insuranceInstruction = new Paragraph("Affix copy of front side of Insurance card", labelFont);

insuranceInstruction.Alignment = Element.ALIGN\_CENTER;

document.Add(insuranceInstruction);

}

}

// Enhanced data models with database integration support

public class MedicalClaimFormData

{

public string Date { get; set; }

public string ClinicName { get; set; } = "Emirates";

public string CardHolderName { get; set; }

public int? Age { get; set; }

public string Sex { get; set; } // "M" or "F"

public string TelNo { get; set; }

public string MobileNo { get; set; }

public string InsCardNo { get; set; }

public string ValidUpTo { get; set; }

public string CompanyName { get; set; }

public string EmployeeNo { get; set; }

public string Nationality { get; set; }

public string SignatureDate { get; set; }

public List<string> ServiceTypes { get; set; } = new List<string>(); // CLINIC, PHARMACY, DIAGNOSTIC CENTRE, HOSPITAL OR OTHER

public List<PharmaceuticalItem> Pharmaceuticals { get; set; } = new List<PharmaceuticalItem>();

public string Temperature { get; set; }

public string BloodPressure { get; set; }

public string Pulse { get; set; }

public string SignsSymptoms { get; set; }

public string OnsetDate { get; set; }

public List<string> VisitTypes { get; set; } = new List<string>(); // Emergency, Work related, New visit, Follow up visit

public string Diagnosis { get; set; }

public List<string> ManagementPlan { get; set; } = new List<string>();

public string DoctorNameAndSignature { get; set; }

public string DiagnosticProcedures { get; set; }

}

public class PharmaceuticalItem

{

public string TradeName { get; set; }

public string Dose { get; set; }

public string TotalDuration { get; set; }

public string Quantity { get; set; }

public decimal? Price { get; set; }

}

// Database helper class for data access

public static class DatabaseHelper

{

public static MedicalClaimFormData GetClaimData(string claimId)

{

string connectionString = ConfigurationManager.ConnectionStrings["MedicalDB"].ConnectionString;

using (SqlConnection conn = new SqlConnection(connectionString))

{

var formData = new MedicalClaimFormData();

// Get main claim data

string mainQuery = @"

SELECT c.\*, p.Name, p.Age, p.Sex, p.TelNo, p.MobileNo, p.InsCardNo,

p.ValidUpTo, p.CompanyName, p.EmployeeNo, p.Nationality,

cd.Temperature, cd.BloodPressure, cd.Pulse, cd.SignsSymptoms,

cd.OnsetDate, cd.Diagnosis, cd.DoctorNameAndSignature, cd.DiagnosticProcedures

FROM Claims c

INNER JOIN Patients p ON c.PatientId = p.Id

LEFT JOIN ClinicalDetails cd ON c.Id = cd.ClaimId

WHERE c.ClaimId = @ClaimId";

using (SqlCommand cmd = new SqlCommand(mainQuery, conn))

{

cmd.Parameters.AddWithValue("@ClaimId", claimId);

conn.Open();

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

formData.Date = Convert.ToDateTime(reader["ClaimDate"]).ToString("dd/MM/yyyy");

formData.CardHolderName = reader["Name"].ToString();

formData.Age = Convert.ToInt32(reader["Age"]);

formData.Sex = reader["Sex"].ToString();

formData.TelNo = reader["TelNo"].ToString();

formData.MobileNo = reader["MobileNo"].ToString();

formData.InsCardNo = reader["InsCardNo"].ToString();

formData.ValidUpTo = Convert.ToDateTime(reader["ValidUpTo"]).ToString("dd/MM/yyyy");

formData.CompanyName = reader["CompanyName"].ToString();

formData.EmployeeNo = reader["EmployeeNo"].ToString();

formData.Nationality = reader["Nationality"].ToString();

formData.Temperature = reader["Temperature"].ToString();

formData.BloodPressure = reader["BloodPressure"].ToString();

formData.Pulse = reader["Pulse"].ToString();

formData.SignsSymptoms = reader["SignsSymptoms"].ToString();

formData.OnsetDate = reader["OnsetDate"].ToString();

formData.Diagnosis = reader["Diagnosis"].ToString();

formData.DoctorNameAndSignature = reader["DoctorNameAndSignature"].ToString();

formData.DiagnosticProcedures = reader["DiagnosticProcedures"].ToString();

}

}

}

// Get service types

string serviceQuery = "SELECT ServiceType FROM ClaimServiceTypes WHERE ClaimId = (SELECT Id FROM Claims WHERE ClaimId = @ClaimId)";

using (SqlCommand cmd = new SqlCommand(serviceQuery, conn))

{

cmd.Parameters.AddWithValue("@ClaimId", claimId);

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

formData.ServiceTypes.Add(reader["ServiceType"].ToString());

}

}

}

// Get visit types

string visitQuery = "SELECT VisitType FROM ClaimVisitTypes WHERE ClaimId = (SELECT Id FROM Claims WHERE ClaimId = @ClaimId)";

using (SqlCommand cmd = new SqlCommand(visitQuery, conn))

{

cmd.Parameters.AddWithValue("@ClaimId", claimId);

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

formData.VisitTypes.Add(reader["VisitType"].ToString());

}

}

}

// Get pharmaceuticals

string pharmaQuery = @"

SELECT TradeName, Dose, TotalDuration, Quantity, Price

FROM PharmaceuticalItems

WHERE ClaimId = (SELECT Id FROM Claims WHERE ClaimId = @ClaimId)";

using (SqlCommand cmd = new SqlCommand(pharmaQuery, conn))

{

cmd.Parameters.AddWithValue("@ClaimId", claimId);

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

formData.Pharmaceuticals.Add(new PharmaceuticalItem

{

TradeName = reader["TradeName"].ToString(),

Dose = reader["Dose"].ToString(),

TotalDuration = reader["TotalDuration"].ToString(),

Quantity = reader["Quantity"].ToString(),

Price = reader["Price"] != DBNull.Value ? Convert.ToDecimal(reader["Price"]) : (decimal?)null

});

}

}

}

// Get management plan

string planQuery = "SELECT PlanItem FROM ManagementPlan WHERE ClaimId = (SELECT Id FROM Claims WHERE ClaimId = @ClaimId) ORDER BY ItemOrder";

using (SqlCommand cmd = new SqlCommand(planQuery, conn))

{

cmd.Parameters.AddWithValue("@ClaimId", claimId);

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

formData.ManagementPlan.Add(reader["PlanItem"].ToString());

}

}

}

return formData;

}

}

public static void SaveClaimData(MedicalClaimFormData formData)

{

string connectionString = ConfigurationManager.ConnectionStrings["MedicalDB"].ConnectionString;

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

using (SqlTransaction transaction = conn.BeginTransaction())

{

try

{

// Insert/Update main claim record

string claimSql = @"

INSERT INTO Claims (ClaimDate, PatientId, CreatedDate)

OUTPUT INSERTED.Id

VALUES (@ClaimDate, @PatientId, GETDATE())";

// Implementation for saving data would go here

// This is a sample structure for database operations

transaction.Commit();

}

catch (Exception)

{

transaction.Rollback();

throw;

}

}

}

}

}

}

using System;

using System.IO;

using System.Web;

using System.Collections.Generic;

using System.Linq;

using iTextSharp.text;

using iTextSharp.text.pdf;

namespace MedicalClaimFormGenerator

{

public partial class GenerateClaimForm : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

// Get form data from request parameters or use sample data

var formData = GetFormDataFromRequest();

GenerateMedicalClaimForm(formData);

}

}

private MedicalClaimFormData GetFormDataFromRequest()

{

// Create form data from query parameters or form submission

var formData = new MedicalClaimFormData();

// Try to get data from request parameters

formData.Date = Request.QueryString["date"] ?? DateTime.Now.ToString("dd/MM/yyyy");

formData.ClinicName = Request.QueryString["clinicName"] ?? "Emirates Medical Center";

formData.CardHolderName = Request.QueryString["cardHolderName"] ?? "AHMED AL MAHMOUD";

// Parse age safely

if (int.TryParse(Request.QueryString["age"], out int age))

formData.Age = age;

else

formData.Age = 35; // Default value

formData.Sex = Request.QueryString["sex"] ?? "M";

formData.TelNo = Request.QueryString["telNo"] ?? "04-3456789";

formData.MobileNo = Request.QueryString["mobileNo"] ?? "050-1234567";

formData.InsCardNo = Request.QueryString["insCardNo"] ?? "EMC123456789";

formData.ValidUpTo = Request.QueryString["validUpTo"] ?? "31/12/2025";

formData.CompanyName = Request.QueryString["companyName"] ?? "ABC TRADING LLC";

formData.EmployeeNo = Request.QueryString["employeeNo"] ?? "EMP001";

formData.Nationality = Request.QueryString["nationality"] ?? "UAE";

formData.SignatureDate = Request.QueryString["signatureDate"] ?? DateTime.Now.ToString("dd/MM/yyyy");

// Clinical details

formData.Temperature = Request.QueryString["temperature"] ?? "37.2";

formData.BloodPressure = Request.QueryString["bloodPressure"] ?? "120/80";

formData.Pulse = Request.QueryString["pulse"] ?? "72";

formData.SignsSymptoms = Request.QueryString["signsSymptoms"] ?? "Fever, headache, body ache";

formData.OnsetDate = Request.QueryString["onsetDate"] ?? "15/01/2025";

formData.Diagnosis = Request.QueryString["diagnosis"] ?? "Viral fever with associated symptoms";

formData.DoctorNameAndSignature = Request.QueryString["doctorName"] ?? "Dr. Sarah Ahmed";

formData.DiagnosticProcedures = Request.QueryString["diagnosticProcedures"] ?? "";

// Parse service types from comma-separated string

string serviceTypes = Request.QueryString["serviceTypes"] ?? "CLINIC";

formData.ServiceTypes = serviceTypes.Split(',').Where(s => !string.IsNullOrEmpty(s)).ToList();

// Parse visit types from comma-separated string

string visitTypes = Request.QueryString["visitTypes"] ?? "New visit";

formData.VisitTypes = visitTypes.Split(',').Where(v => !string.IsNullOrEmpty(v)).ToList();

// Parse pharmaceuticals from request (simple format)

for (int i = 1; i <= 1; i++)

{

string tradeName = Request.QueryString[$"pharma{i}TradeName"];

string dose = Request.QueryString[$"pharma{i}Dose"];

string duration = Request.QueryString[$"pharma{i}Duration"];

string quantity = Request.QueryString[$"pharma{i}Quantity"];

string priceStr = Request.QueryString[$"pharma{i}Price"];

if (!string.IsNullOrEmpty(tradeName))

{

var pharmaItem = new PharmaceuticalItem

{

TradeName = tradeName,

Dose = dose ?? "",

TotalDuration = duration ?? "",

Quantity = quantity ?? "",

Price = decimal.TryParse(priceStr, out decimal price) ? price : (decimal?)null

};

formData.Pharmaceuticals.Add(pharmaItem);

}

else if (i == 1) // Add default pharmaceutical for demo

{

formData.Pharmaceuticals.Add(new PharmaceuticalItem

{

TradeName = "Paracetamol 500mg",

Dose = "500mg",

TotalDuration = "5 days",

Quantity = "15 tablets",

Price = 25.50m

});

}

}

// Parse management plan from request

for (int i = 1; i <= 4; i++)

{

string planItem = Request.QueryString[$"plan{i}"];

if (!string.IsNullOrEmpty(planItem))

{

formData.ManagementPlan.Add(planItem);

}

else if (i <= 3) // Add default plans for demo

{

switch (i)

{

case 1: formData.ManagementPlan.Add("Complete blood count test"); break;

case 2: formData.ManagementPlan.Add("Paracetamol 500mg TID"); break;

case 3: formData.ManagementPlan.Add("Rest and adequate fluid intake"); break;

}

}

}

return formData;

}

private void GenerateMedicalClaimForm(MedicalClaimFormData formData)

{

Document document = new Document(PageSize.A4, 20, 20, 20, 20);

MemoryStream stream = new MemoryStream();

try

{

PdfWriter writer = PdfWriter.GetInstance(document, stream);

document.Open();

// Create precise fonts matching the original document

BaseFont helvetica = BaseFont.CreateFont(BaseFont.HELVETICA, BaseFont.CP1252, BaseFont.NOT\_EMBEDDED);

BaseFont helveticaBold = BaseFont.CreateFont(BaseFont.HELVETICA\_BOLD, BaseFont.CP1252, BaseFont.NOT\_EMBEDDED);

Font titleFont = new Font(helveticaBold, 14, Font.BOLD, BaseColor.BLACK);

Font headerFont = new Font(helveticaBold, 11, Font.BOLD, BaseColor.BLACK);

Font labelFont = new Font(helvetica, 9, Font.NORMAL, BaseColor.BLACK);

Font boldLabelFont = new Font(helveticaBold, 9, Font.BOLD, BaseColor.BLACK);

Font smallFont = new Font(helvetica, 8, Font.NORMAL, BaseColor.BLACK);

Font tinyFont = new Font(helvetica, 7, Font.NORMAL, BaseColor.BLACK);

// Add company logo (placeholder)

AddCompanyLogo(document, writer);

// Title section

Paragraph title = new Paragraph("Medical Expenses Claim form", titleFont);

title.Alignment = Element.ALIGN\_LEFT;

title.SpacingAfter = 10f;

document.Add(title);

// Date row

PdfPTable dateRow = CreateDateRow(formData, labelFont);

document.Add(dateRow);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Clinic name row

PdfPTable clinicRow = CreateClinicRow(formData, labelFont);

document.Add(clinicRow);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Patient details section

PdfPTable patientSection = CreatePatientSection(formData, labelFont);

document.Add(patientSection);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Contact details section

PdfPTable contactSection = CreateContactSection(formData, labelFont);

document.Add(contactSection);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Insurance details section

PdfPTable insuranceSection = CreateInsuranceSection(formData, labelFont);

document.Add(insuranceSection);

document.Add(new Paragraph(" ", new Font(helvetica, 3)));

// Company details section

PdfPTable companySection = CreateCompanySection(formData, labelFont);

document.Add(companySection);

document.Add(new Paragraph(" ", new Font(helvetica, 8)));

// Authorization text - exact formatting

Paragraph authText = new Paragraph();

authText.Add(new Chunk("I hereby authorize the physician, Hospital or pharmacy to file a claim for medical services on my behalf and I confirm that the above-", labelFont));

authText.Add(new Chunk("\nmentioned examination/Investigation/therapy is given to me by the doctor. I hereby authorize any Clinic, Physician, Pharmacy or any", labelFont));

authText.Add(new Chunk("\nother person who has provided medical services to me to furnish any and all information with regard to any medical history, medical condition,", labelFont));

authText.Add(new Chunk("\nor medical services and copies of all medical and Clinic records.", labelFont));

authText.SpacingAfter = 10f;

document.Add(authText);

// Signature section

PdfPTable signatureSection = CreateSignatureSection(formData, labelFont);

document.Add(signatureSection);

document.Add(new Paragraph(" ", new Font(helvetica, 8)));

// Service type checkboxes with exact positioning

PdfPTable serviceTypeSection = CreateServiceTypeSection(formData, labelFont);

document.Add(serviceTypeSection);

// "Kindly tick" instruction

Paragraph kindlyTick = new Paragraph("Kindly tick whichever is applicable", smallFont);

kindlyTick.Alignment = Element.ALIGN\_CENTER;

kindlyTick.SpacingAfter = 10f;

document.Add(kindlyTick);

// Pharmaceuticals section with exact table structure

AddPharmaceuticalsSection(document, formData, headerFont, labelFont, smallFont, tinyFont);

// Add spacing before footer

document.Add(new Paragraph(" ", new Font(helvetica, 10)));

// Company footer with exact formatting

AddCompanyFooter(document, headerFont, smallFont);

// Start new page for clinical details

document.NewPage();

AddClinicalDetailsPage(document, formData, headerFont, labelFont, smallFont);

document.Close();

// Send PDF to browser

Response.ContentType = "application/pdf";

Response.AddHeader("content-disposition", "attachment;filename=Medical\_Expenses\_Claim\_Form.pdf");

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.BinaryWrite(stream.ToArray());

Response.End();

}

catch (Exception ex)

{

Response.Write($"Error generating PDF: {ex.Message}");

}

finally

{

if (document.IsOpen())

document.Close();

stream?.Close();

}

}

private void AddCompanyLogo(Document document, PdfWriter writer)

{

try

{

// Create a simple placeholder logo using shapes

PdfContentByte cb = writer.DirectContent;

// Draw placeholder logo box

cb.Rectangle(document.PageSize.Width - 100, document.PageSize.Height - 80, 80, 60);

cb.SetColorStroke(BaseColor.BLUE);

cb.SetLineWidth(2f);

cb.Stroke();

// Add logo text

cb.BeginText();

cb.SetFontAndSize(BaseFont.CreateFont(BaseFont.HELVETICA\_BOLD, BaseFont.CP1252, BaseFont.NOT\_EMBEDDED), 10);

cb.SetTextMatrix(document.PageSize.Width - 95, document.PageSize.Height - 50);

cb.ShowText("FMC");

cb.SetTextMatrix(document.PageSize.Width - 95, document.PageSize.Height - 65);

cb.ShowText("LOGO");

cb.EndText();

}

catch (Exception)

{

// Logo creation failed, continue without logo

}

}

private PdfPTable CreateDateRow(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 50f, 30f, 20f, 30f, 20f, 350f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Date :", font)) { Border = Rectangle.NO\_BORDER });

if (!string.IsNullOrEmpty(formData.Date))

{

var dateParts = formData.Date.Split('/');

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 0 ? dateParts[0] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 1 ? dateParts[1] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase((dateParts.Length > 2 ? dateParts[2] : "\_\_\_\_\_\_\_\_") + ".", font)) { Border = Rectangle.NO\_BORDER });

}

else

{

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_.", font)) { Border = Rectangle.NO\_BORDER });

}

return table;

}

private PdfPTable CreateClinicRow(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 100f, 100f, 400f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.AddCell(new PdfPCell(new Phrase("Clinic Name", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Emirates", font)) { Border = Rectangle.NO\_BORDER });

string clinicName = !string.IsNullOrEmpty(formData.ClinicName) ?

formData.ClinicName.Replace("Emirates", "").Trim() :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

table.AddCell(new PdfPCell(new Phrase(clinicName + ".", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreatePatientSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 120f, 250f, 35f, 50f, 40f, 15f, 15f, 15f, 15f, 35f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Card Holder's Name", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.CardHolderName) ? formData.CardHolderName : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Age", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(formData.Age.HasValue ? formData.Age.Value.ToString() : "\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Sex :", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("M", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(formData.Sex == "M" ? "☑" : "☐", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(" F", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(formData.Sex == "F" ? "☑" : "☐", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("", font)) { Border = Rectangle.NO\_BORDER }); // Empty cell for spacing

return table;

}

private PdfPTable CreateContactSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 120f, 200f, 80f, 200f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Card Holder's Tel No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.TelNo) ? formData.TelNo : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Mobile No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase((!string.IsNullOrEmpty(formData.MobileNo) ? formData.MobileNo : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_") + ".", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreateInsuranceSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 80f, 200f, 80f, 240f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Ins. Card No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.InsCardNo) ? formData.InsCardNo : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Valid up to", font)) { Border = Rectangle.NO\_BORDER });

if (!string.IsNullOrEmpty(formData.ValidUpTo))

{

var validParts = formData.ValidUpTo.Split('/');

string validText = (validParts.Length > 0 ? validParts[0] : "\_\_\_\_") + " / " +

(validParts.Length > 1 ? validParts[1] : "\_\_\_\_") + " / " +

(validParts.Length > 2 ? validParts[2] : "\_\_\_\_\_\_\_\_\_\_") + ".";

table.AddCell(new PdfPCell(new Phrase(validText, font)) { Border = Rectangle.NO\_BORDER });

}

else

{

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_ / \_\_\_\_ / \_\_\_\_\_\_\_\_\_\_.", font)) { Border = Rectangle.NO\_BORDER });

}

return table;

}

private PdfPTable CreateCompanySection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 100f, 180f, 80f, 80f, 80f, 80f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Company Name", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.CompanyName) ? formData.CompanyName : "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Employee No", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.EmployeeNo) ? formData.EmployeeNo : "\_\_\_\_\_\_\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("Nationality", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase((!string.IsNullOrEmpty(formData.Nationality) ? formData.Nationality : "\_\_\_\_\_\_\_\_\_\_") + ".", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreateSignatureSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 40f, 30f, 20f, 30f, 20f, 150f, 210f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 2f;

table.AddCell(new PdfPCell(new Phrase("Date", font)) { Border = Rectangle.NO\_BORDER });

if (!string.IsNullOrEmpty(formData.SignatureDate))

{

var dateParts = formData.SignatureDate.Split('/');

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 0 ? dateParts[0] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(dateParts.Length > 1 ? dateParts[1] : "\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

}

else

{

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("/", font)) { Border = Rectangle.NO\_BORDER });

}

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_ Signature of the Patient", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private PdfPTable CreateServiceTypeSection(MedicalClaimFormData formData, Font font)

{

PdfPTable table = new PdfPTable(new float[] { 30f, 100f, 30f, 100f, 30f, 150f, 30f, 130f });

table.WidthPercentage = 100;

table.DefaultCell.Border = Rectangle.NO\_BORDER;

table.DefaultCell.PaddingBottom = 5f;

table.DefaultCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

string clinicCheck = formData.ServiceTypes.Contains("CLINIC") ? "☑" : "☐";

string pharmacyCheck = formData.ServiceTypes.Contains("PHARMACY") ? "☑" : "☐";

string diagnosticCheck = formData.ServiceTypes.Contains("DIAGNOSTIC CENTRE") ? "☑" : "☐";

string hospitalCheck = formData.ServiceTypes.Contains("HOSPITAL OR OTHER") ? "☑" : "☐";

table.AddCell(new PdfPCell(new Phrase(clinicCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("CLINIC", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(pharmacyCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("PHARMACY", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(diagnosticCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("DIAGNOSTIC CENTRE", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase(hospitalCheck, font)) { Border = Rectangle.NO\_BORDER, HorizontalAlignment = Element.ALIGN\_CENTER });

table.AddCell(new PdfPCell(new Phrase("HOSPITAL", font)) { Border = Rectangle.NO\_BORDER });

// Second row for "OR OTHER"

for (int i = 0; i < 4; i++)

{

table.AddCell(new PdfPCell(new Phrase("", font)) { Border = Rectangle.NO\_BORDER });

}

table.AddCell(new PdfPCell(new Phrase("", font)) { Border = Rectangle.NO\_BORDER });

table.AddCell(new PdfPCell(new Phrase("OR OTHER", font)) { Border = Rectangle.NO\_BORDER });

return table;

}

private void AddPharmaceuticalsSection(Document document, MedicalClaimFormData formData, Font headerFont, Font labelFont, Font smallFont, Font tinyFont)

{

// Section header

Paragraph pharmaHeader = new Paragraph("Pharmaceuticals (to be filled by treating doctor only)", headerFont);

pharmaHeader.SpacingAfter = 6f;

document.Add(pharmaHeader);

// Main table structure

PdfPTable mainPharmaTable = new PdfPTable(new float[] { 200f, 120f, 120f, 80f, 80f });

mainPharmaTable.WidthPercentage = 100;

// Headers with gray background

PdfPCell tradeNameHeader = new PdfPCell(new Phrase("Trade Name", headerFont));

tradeNameHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

tradeNameHeader.BackgroundColor = new BaseColor(220, 220, 220);

tradeNameHeader.Padding = 5f;

mainPharmaTable.AddCell(tradeNameHeader);

PdfPCell doseHeader = new PdfPCell(new Phrase("Dose", headerFont));

doseHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

doseHeader.BackgroundColor = new BaseColor(220, 220, 220);

doseHeader.Padding = 5f;

mainPharmaTable.AddCell(doseHeader);

PdfPCell durationHeader = new PdfPCell(new Phrase("Total Duration", headerFont));

durationHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

durationHeader.BackgroundColor = new BaseColor(220, 220, 220);

durationHeader.Padding = 5f;

mainPharmaTable.AddCell(durationHeader);

// Second header row for pharmacy section

PdfPCell pharmacySection = new PdfPCell(new Phrase("(To be filled by the pharmacy)", tinyFont));

pharmacySection.HorizontalAlignment = Element.ALIGN\_CENTER;

pharmacySection.BackgroundColor = new BaseColor(220, 220, 220);

pharmacySection.Padding = 3f;

pharmacySection.Colspan = 3;

mainPharmaTable.AddCell(pharmacySection);

PdfPCell quantityHeader = new PdfPCell(new Phrase("Quantity", headerFont));

quantityHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

quantityHeader.BackgroundColor = new BaseColor(220, 220, 220);

quantityHeader.Padding = 5f;

mainPharmaTable.AddCell(quantityHeader);

PdfPCell priceHeader = new PdfPCell(new Phrase("Price", headerFont));

priceHeader.HorizontalAlignment = Element.ALIGN\_CENTER;

priceHeader.BackgroundColor = new BaseColor(220, 220, 220);

priceHeader.Padding = 5f;

mainPharmaTable.AddCell(priceHeader);

// Data rows

for (int i = 0; i < 1; i++)

{

PharmaceuticalItem item = i < formData.Pharmaceuticals.Count ? formData.Pharmaceuticals[i] : null;

mainPharmaTable.AddCell(new PdfPCell(new Phrase($"{i + 1}) " + (item?.TradeName ?? ""), labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

mainPharmaTable.AddCell(new PdfPCell(new Phrase(item?.Quantity ?? "", labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

mainPharmaTable.AddCell(new PdfPCell(new Phrase((bool)item?.Price.HasValue ? item.Price.Value.ToString("0.00") : "", labelFont)) { MinimumHeight = 25f, VerticalAlignment = Element.ALIGN\_MIDDLE, Padding = 3f });

}

// Total row

decimal totalPrice = formData.Pharmaceuticals.Where(p => p.Price.HasValue).Sum(p => p.Price.Value);

PdfPCell exclusionsCell = new PdfPCell(new Phrase("Please apply general exclusions", smallFont));

exclusionsCell.Colspan = 3;

exclusionsCell.HorizontalAlignment = Element.ALIGN\_LEFT;

exclusionsCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

exclusionsCell.Padding = 5f;

mainPharmaTable.AddCell(exclusionsCell);

PdfPCell totalLabelCell = new PdfPCell(new Phrase("Total", headerFont));

totalLabelCell.HorizontalAlignment = Element.ALIGN\_RIGHT;

totalLabelCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

totalLabelCell.Padding = 5f;

mainPharmaTable.AddCell(totalLabelCell);

PdfPCell totalValueCell = new PdfPCell(new Phrase(totalPrice > 0 ? totalPrice.ToString("0.00") : "", labelFont));

totalValueCell.HorizontalAlignment = Element.ALIGN\_CENTER;

totalValueCell.VerticalAlignment = Element.ALIGN\_MIDDLE;

totalValueCell.Padding = 5f;

mainPharmaTable.AddCell(totalValueCell);

document.Add(mainPharmaTable);

}

private void AddCompanyFooter(Document document, Font headerFont, Font smallFont)

{

// Company footer in bordered box

PdfPTable footerTable = new PdfPTable(1);

footerTable.WidthPercentage = 100;

footerTable.SpacingBefore = 15f;

PdfPCell footerCell = new PdfPCell();

footerCell.BorderWidth = 1f;

footerCell.BorderColor = BaseColor.BLACK;

footerCell.Padding = 8f;

// Company name

Paragraph companyName = new Paragraph("F M C NETWORK UAE", headerFont);

companyName.Alignment = Element.ALIGN\_CENTER;

companyName.SpacingAfter = 5f;

// Contact details

Paragraph contactDetails = new Paragraph();

contactDetails.Add(new Chunk("P. O. BOX: 50430, DUBAI, Tel – 04 3871900, Fax – 04 3977842", smallFont));

contactDetails.Add(new Chunk("\nEmail – approval@fmchealthcare.ae Helpline Number: 600-565691", smallFont));

contactDetails.Alignment = Element.ALIGN\_CENTER;

footerCell.AddElement(companyName);

footerCell.AddElement(contactDetails);

footerTable.AddCell(footerCell);

document.Add(footerTable);

}

private void AddClinicalDetailsPage(Document document, MedicalClaimFormData formData, Font headerFont, Font labelFont, Font smallFont)

{

// Clinical Details header with vitals

PdfPTable clinicalHeader = new PdfPTable(new float[] { 120f, 60f, 30f, 60f, 80f, 60f, 80f, 110f });

clinicalHeader.WidthPercentage = 100;

clinicalHeader.DefaultCell.Border = Rectangle.NO\_BORDER;

clinicalHeader.DefaultCell.PaddingBottom = 3f;

clinicalHeader.AddCell(new PdfPCell(new Phrase("Clinical Details: Temp", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.Temperature) ? formData.Temperature : "\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("°C", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("B.P.", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase(!string.IsNullOrEmpty(formData.BloodPressure) ? formData.BloodPressure : "\_\_\_\_\_\_\_", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("mmHg", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase("Pulse.", labelFont)) { Border = Rectangle.NO\_BORDER });

clinicalHeader.AddCell(new PdfPCell(new Phrase((!string.IsNullOrEmpty(formData.Pulse) ? formData.Pulse : "\_\_\_\_\_\_") + " / min", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(clinicalHeader);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 6)));

// Signs & Symptoms

string signsText = !string.IsNullOrEmpty(formData.SignsSymptoms) ?

formData.SignsSymptoms :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph signsSymptoms = new Paragraph("Sign & Symptoms " + signsText + ".", labelFont);

signsSymptoms.SpacingAfter = 8f;

document.Add(signsSymptoms);

// Date of onset

string onsetText = !string.IsNullOrEmpty(formData.OnsetDate) ?

formData.OnsetDate :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph onsetDate = new Paragraph("Date of onset of illness:" + onsetText, labelFont);

onsetDate.SpacingAfter = 8f;

document.Add(onsetDate);

// Visit type checkboxes

PdfPTable visitTypeTable = new PdfPTable(new float[] { 30f, 100f, 30f, 120f, 30f, 100f, 30f, 160f });

visitTypeTable.WidthPercentage = 100;

visitTypeTable.DefaultCell.Border = Rectangle.NO\_BORDER;

visitTypeTable.DefaultCell.PaddingBottom = 3f;

string emergencyCheck = formData.VisitTypes.Contains("Emergency") ? "☑" : "☐";

string workCheck = formData.VisitTypes.Contains("Work related") ? "☑" : "☐";

string newCheck = formData.VisitTypes.Contains("New visit") ? "☑" : "☐";

string followupCheck = formData.VisitTypes.Contains("Follow up visit") ? "☑" : "☐";

visitTypeTable.AddCell(new PdfPCell(new Phrase(emergencyCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Emergency", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase(workCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Work related", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase(newCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("New visit", labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase(followupCheck, labelFont)) { Border = Rectangle.NO\_BORDER });

visitTypeTable.AddCell(new PdfPCell(new Phrase("Follow up visit", labelFont)) { Border = Rectangle.NO\_BORDER });

document.Add(visitTypeTable);

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 8)));

// Diagnosis section

string diagnosisText = !string.IsNullOrEmpty(formData.Diagnosis) ?

formData.Diagnosis :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph diagnosis1 = new Paragraph("Diagnosis " + diagnosisText + ".", labelFont);

document.Add(diagnosis1);

// Additional diagnosis lines

Paragraph diagnosis2 = new Paragraph("………………………..…………………………………………………………………………………", labelFont);

document.Add(diagnosis2);

Paragraph diagnosis3 = new Paragraph("…………", labelFont);

document.Add(diagnosis3);

Paragraph diagnosis4 = new Paragraph("………………………………………………………………………………………….", labelFont);

diagnosis4.SpacingAfter = 10f;

document.Add(diagnosis4);

// Management plan

Paragraph managementTitle = new Paragraph("Management plan (Services inside the clinic including injections and investigations)", labelFont);

managementTitle.SpacingAfter = 8f;

document.Add(managementTitle);

for (int i = 0; i < 1; i++)

{

string planText = i < formData.ManagementPlan.Count && !string.IsNullOrEmpty(formData.ManagementPlan[i]) ?

formData.ManagementPlan[i] :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph managementItem = new Paragraph($"{i + 1}) " + planText, labelFont);

document.Add(managementItem);

}

document.Add(new Paragraph(" ", new Font(Font.FontFamily.HELVETICA, 10)));

// Doctor's signature

string doctorText = !string.IsNullOrEmpty(formData.DoctorNameAndSignature) ?

formData.DoctorNameAndSignature :

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

Paragraph doctorSig = new Paragraph("Doctor's Name and signature with seal: " + doctorText + ".", labelFont);

doctorSig.SpacingAfter = 12f;

document.Add(doctorSig);

// Diagnostic Procedures

string diagnosticText = !string.IsNullOrEmpty(formData.DiagnosticProcedures) ?

formData.DiagnosticProcedures :

"";

Paragraph diagnosticProc = new Paragraph("Diagnostic Procedures referred outside: " + diagnosticText, labelFont);

diagnosticProc.SpacingAfter = 25f;

document.Add(diagnosticProc);

// ANNEXURE V section

Paragraph annexure = new Paragraph("ANNEXURE V", headerFont);

annexure.Alignment = Element.ALIGN\_CENTER;

annexure.SpacingAfter = 10f;

document.Add(annexure);

// Insurance card instruction

Paragraph insuranceInstruction = new Paragraph("Affix copy of front side of Insurance card", labelFont);

insuranceInstruction.Alignment = Element.ALIGN\_CENTER;

document.Add(insuranceInstruction);

}

}

// Data model classes - no database dependencies

public class MedicalClaimFormData

{

public string Date { get; set; }

public string ClinicName { get; set; } = "Emirates";

public string CardHolderName { get; set; }

public int? Age { get; set; }

public string Sex { get; set; } // "M" or "F"

public string TelNo { get; set; }

public string MobileNo { get; set; }

public string InsCardNo { get; set; }

public string ValidUpTo { get; set; }

public string CompanyName { get; set; }

public string EmployeeNo { get; set; }

public string Nationality { get; set; }

public string SignatureDate { get; set; }

public List<string> ServiceTypes { get; set; } = new List<string>(); // CLINIC, PHARMACY, DIAGNOSTIC CENTRE, HOSPITAL OR OTHER

public List<PharmaceuticalItem> Pharmaceuticals { get; set; } = new List<PharmaceuticalItem>();

public string Temperature { get; set; }

public string BloodPressure { get; set; }

public string Pulse { get; set; }

public string SignsSymptoms { get; set; }

public string OnsetDate { get; set; }

public List<string> VisitTypes { get; set; } = new List<string>(); // Emergency, Work related, New visit, Follow up visit

public string Diagnosis { get; set; }

public List<string> ManagementPlan { get; set; } = new List<string>();

public string DoctorNameAndSignature { get; set; }

public string DiagnosticProcedures { get; set; }

}

public class PharmaceuticalItem

{

public string TradeName { get; set; }

public string Dose { get; set; }

public string TotalDuration { get; set; }

public string Quantity { get; set; }

public decimal? Price { get; set; }

}

// Utility class for form data validation and formatting

public static class FormValidator

{

public static bool ValidateFormData(MedicalClaimFormData formData, out List<string> errors)

{

errors = new List<string>();

// Basic validations

if (string.IsNullOrEmpty(formData.CardHolderName))

errors.Add("Card Holder Name is required");

if (!formData.Age.HasValue || formData.Age < 0 || formData.Age > 120)

errors.Add("Valid age is required (0-120)");

if (string.IsNullOrEmpty(formData.Sex) || (formData.Sex != "M" && formData.Sex != "F"))

errors.Add("Sex must be M or F");

if (!string.IsNullOrEmpty(formData.Date))

{

if (!DateTime.TryParseExact(formData.Date, "dd/MM/yyyy", null, System.Globalization.DateTimeStyles.None, out \_))

errors.Add("Date must be in dd/MM/yyyy format");

}

// Validate pharmaceutical prices

foreach (var pharma in formData.Pharmaceuticals)

{

if (pharma.Price.HasValue && pharma.Price < 0)

errors.Add($"Price for {pharma.TradeName} cannot be negative");

}

return errors.Count == 0;

}

public static void FormatPhoneNumbers(MedicalClaimFormData formData)

{

// Format phone numbers to standard UAE format

if (!string.IsNullOrEmpty(formData.TelNo))

formData.TelNo = FormatUAEPhoneNumber(formData.TelNo);

if (!string.IsNullOrEmpty(formData.MobileNo))

formData.MobileNo = FormatUAEMobileNumber(formData.MobileNo);

}

private static string FormatUAEPhoneNumber(string phone)

{

// Remove all non-digit characters

string digits = new string(phone.Where(char.IsDigit).ToArray());

// Format as UAE landline (04-XXXXXXX)

if (digits.Length == 8 && digits.StartsWith("04"))

return $"{digits.Substring(0, 2)}-{digits.Substring(2)}";

else if (digits.Length == 7)

return $"04-{digits}";

return phone; // Return original if format doesn't match

}

private static string FormatUAEMobileNumber(string mobile)

{

// Remove all non-digit characters

string digits = new string(mobile.Where(char.IsDigit).ToArray());

// Format as UAE mobile (050-XXXXXXX)

if (digits.Length == 9 && (digits.StartsWith("050") || digits.StartsWith("055") ||

digits.StartsWith("056") || digits.StartsWith("052")))

return $"{digits.Substring(0, 3)}-{digits.Substring(3)}";

return mobile; // Return original if format doesn't match

}

}

}