// code for pdf Dawnload with design

private void imgbtnPdf\_Click(object sender, EventArgs e)

{

// Create a SaveFileDialog to specify the location to save the PDF

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "PDF Files|\*.pdf",

Title = "Save as PDF",

FileName = "Visitor\_Block\_Report\_" + DateTime.Now.ToString("yyyy/MMdd/HHmmss") + ".pdf" // Auto-generate file name

};

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

// Create a new PDF document with increased page size

Document pdfDoc = new Document(PageSize.A4.Rotate(), 20, 20, 20, 20); // Wider page size with custom margins

PdfWriter writer = PdfWriter.GetInstance(pdfDoc, new FileStream(saveFileDialog.FileName, FileMode.Create));

pdfDoc.Open();

// Add a title and date to the PDF

var titleFont = FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 18, BaseColor.BLUE);

var normalFont = FontFactory.GetFont(FontFactory.HELVETICA, 12, BaseColor.BLACK);

// Title

pdfDoc.Add(new Paragraph("Visitor Block Report", titleFont));

// Add the current date and time

string currentDate = DateTime.Now.ToString("dd MMMM yyyy, HH:mm:ss");

pdfDoc.Add(new Paragraph($"Generated on: {currentDate}", normalFont));

pdfDoc.Add(new Paragraph(" ", normalFont)); // Add an empty line

// Create a PDF table with only the specified columns

string[] selectedColumns = { "Name", "Phone", "VisitorType", "BlockedDate", "BlockedTime", "BlockedBy", "TotalDays" };

PdfPTable pdfTable = new PdfPTable(selectedColumns.Length)

{

WidthPercentage = 100

};

// Set relative column widths

pdfTable.SetWidths(new float[] { 2f, 2f, 2f, 2f, 2f, 2f, 1.5f }); // Adjust column width as required

// Set header style

BaseColor headerBackgroundColor = new BaseColor(0, 102, 204); // Blue

BaseColor headerTextColor = BaseColor.WHITE;

// Add column headers to the PDF table

foreach (string columnName in selectedColumns)

{

PdfPCell headerCell = new PdfPCell(new Phrase(columnName, FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 12, headerTextColor)))

{

BackgroundColor = headerBackgroundColor,

Padding = 5,

HorizontalAlignment = Element.ALIGN\_CENTER

};

pdfTable.AddCell(headerCell);

}

// Add rows to the PDF table

bool isAlternateRow = false;

foreach (DataGridViewRow row in grdview.Rows)

{

if (row.IsNewRow) continue; // Skip the empty new row

foreach (string columnName in selectedColumns)

{

string cellValue;

if (columnName == "BlockedDate")

{

// Format the BlockedDate column to show only the date (dd/MM/yyyy)

if (DateTime.TryParse(row.Cells[columnName].Value?.ToString(), out DateTime parsedDate))

{

cellValue = parsedDate.ToString("dd/MM/yyyy"); // Format date

}

else

{

cellValue = string.Empty; // If the value is not a valid date

}

}

else

{

cellValue = row.Cells[columnName].Value?.ToString() ?? string.Empty;

}

PdfPCell cellData = new PdfPCell(new Phrase(cellValue, normalFont))

{

Padding = 5,

BackgroundColor = isAlternateRow ? BaseColor.LIGHT\_GRAY : BaseColor.WHITE,

HorizontalAlignment = Element.ALIGN\_LEFT

};

pdfTable.AddCell(cellData);

}

// Alternate row background color

isAlternateRow = !isAlternateRow;

}

// Add the table to the document

pdfDoc.Add(pdfTable);

// Close the document

pdfDoc.Close();

MessageBox.Show("PDF downloaded successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Error while generating PDF: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}