```php
<?php

namespace App\Reports\Support;

use Fpdf\Fpdf;

class LogicsBuilderPdf extends Fpdf
{

    const COL_HDR_COLOR = 100;
    const ROW_FILL_COLOR = 235;
    const LINE_HEIGHT = 5;
    const HEADING_SIZE = 13;
    const BODY_SIZE = 10;
    const CAPTION_SIZE = 7;

    protected $pageHeaderRepeat;
    protected $pageHeaderAdded;
    protected $reportTitle;
    protected $fromDate;
    protected $toDate;
    protected $isFooterRequired;
    protected $widths;
    protected $aligns;
    protected $tblHdrWidths;
    protected $tblHdrAligns;
    protected $colsWithoutSpan1;
    protected $colsWithSpan1;
    protected $colsWithoutSpan2;
    protected $colsWithSpan2;
    protected $colsWithoutSpan3;
    protected $colsWithSpan3;
    protected $colsWithoutSpan4;
    protected $colsWithSpan4;
    protected $colsWithoutSpan5;
    protected $colsWithSpan5;
    protected $subTitle;

    protected $accountRepeatingTableCaption;

    public function __construct($orientation = 'P', $pageHeaderRepeat = false, $reportTitle = 'Report',
$fromDate = '', $toDate = '', $isFooterRequired = true, $subTitle = '', $isMonthlyBill = 0)
    {
        parent::__construct($orientation, 'mm', 'A4');
        $this->pageHeaderRepeat = $pageHeaderRepeat;
        $this->reportTitle = $reportTitle;
        $this->fromDate = $fromDate;
        $this->toDate = $toDate;
        $this->isFooterRequired = $isFooterRequired;
```

```
    $this->resetTableHeaders();

    $this->subTitle = $subTitle;
    $this->isMonthlyBill = $isMonthlyBill;

    $this->SetFont('Times', '', self::BODY_SIZE);
    $this->SetLineWidth(0.3);
    $this->SetDrawColor(0);

    $this->accountRepeatingTableCaption = '';
    $this->vendorName = '';
    $this->vendorContact = '';
}

public function resetTableHeaders()
{
    $this->colsWithoutSpan1 = array();
    $this->colsWithSpan1 = array();
    $this->colsWithoutSpan2 = array();
    $this->colsWithSpan2 = array();
    $this->colsWithoutSpan3 = array();
    $this->colsWithSpan3 = array();
    $this->colsWithoutSpan4 = array();
    $this->colsWithSpan4 = array();
    $this->colsWithoutSpan5 = array();
    $this->colsWithSpan5 = array();
}

public function AddTableCaption($tableCaption)
{
    $this->resetTableHeaders();
    $this->CheckPageBreak(self::CAPTION_SIZE + self::LINE_HEIGHT * 3);
    $this->SetFont('', 'B', self::BODY_SIZE + 1);
    $this->Cell(0, self::CAPTION_SIZE, $tableCaption, 0, 0, 'L');
    $this->SetFont('', '', self::BODY_SIZE);
    $this->Ln();
}

public function AddTableHeader(
    $colsWithoutSpan1,
    $colsWithSpan1 = array(),
    $colsWithoutSpan2 = array(),
    $colsWithSpan2 = array(),
    $colsWithoutSpan3 = array(),
    $colsWithSpan3 = array(),
    $colsWithoutSpan4 = array(),
    $colsWithSpan4 = array(),
    $colsWithoutSpan5 = array(),
    $colsWithSpan5 = array()
```

```
    ) {

        if ($this->accountRepeatingTableCaption != '') {
            $this->AddTableCaption($this->accountRepeatingTableCaption);
        }

        $this->CheckPageBreak(self::LINE_HEIGHT * 3);

        $this->colsWithoutSpan1 = $colsWithoutSpan1;
        $this->colsWithSpan1 = $colsWithSpan1;
        $this->colsWithoutSpan2 = $colsWithoutSpan2;
        $this->colsWithSpan2 = $colsWithSpan2;
        $this->colsWithoutSpan3 = $colsWithoutSpan3;
        $this->colsWithSpan3 = $colsWithSpan3;
        $this->colsWithoutSpan4 = $colsWithoutSpan4;
        $this->colsWithSpan4 = $colsWithSpan4;
        $this->colsWithoutSpan5 = $colsWithoutSpan5;
        $this->colsWithSpan5 = $colsWithSpan5;

        $this->SetFillColor(self::COL_HDR_COLOR);
        $this->SetTextColor(255);
        $this->SetFont('', 'B');

        if (count($this->colsWithSpan1) == 0) {
            if (isset($this->colsWithoutSpan1) && count($this->colsWithoutSpan1) > 0) {
                $this->AddRow($this->colsWithoutSpan1, true, false, true);
            }
        } else {
            $colIndex = 0;
            $this->AddTableHeaderCells($colIndex, $colsWithoutSpan1, $colsWithSpan1);
            $colIndex += count($colsWithoutSpan1) + (count($colsWithSpan1) == 0 ? 0 :
count($colsWithSpan1) - 1);

            $this->AddTableHeaderCells($colIndex, $colsWithoutSpan2, $colsWithSpan2);
            $colIndex += count($colsWithoutSpan2) + (count($colsWithSpan2) == 0 ? 0 :
count($colsWithSpan2) - 1);

            $this->AddTableHeaderCells($colIndex, $colsWithoutSpan3, $colsWithSpan3);
            $colIndex += count($colsWithoutSpan3) + (count($colsWithSpan3) == 0 ? 0 :
count($colsWithSpan3) - 1);

            $this->AddTableHeaderCells($colIndex, $colsWithoutSpan4, $colsWithSpan4);
            $colIndex += count($colsWithoutSpan4) + (count($colsWithSpan4) == 0 ? 0 :
count($colsWithSpan4) - 1);

            $this->AddTableHeaderCells($colIndex, $colsWithoutSpan5, $colsWithSpan5);

            $this->Ln(self::LINE_HEIGHT * 2);
        }
```

```
    $this->SetFont('', '');
    $this->SetTextColor(0);
    $this->tblHdrWidths = $this->widths;
    $this->tblHdrAligns = $this->aligns;
}

public function AddRow($data, $drawBorder = true, $fillRow = false, $isHeaderRow = false)
{
    $nb = 0;
    for ($i = 0; $i < count($data); $i++) {
        $nb = max($nb, $this->NbLines($this->widths[$i], $data[$i]));
    }
    $h = self::LINE_HEIGHT * $nb;

    $this->CheckPageBreak($h);
    for ($i = 0; $i < count($data); $i++) {

        $w = $this->widths[$i];
        $a = isset($this->aligns[$i]) ? $this->aligns[$i] : 'L';
        $x = $this->GetX();
        $y = $this->GetY();
        if (!$isHeaderRow) {
            $this->SetFillColor($fillRow ? self::COL_HDR_COLOR : 255);
        }
        $this->Rect($x, $y, $w, $h, $drawBorder ? 'DF' : 'F');
        if ($fillRow) {
            $this->SetTextColor(255);
            $this->MultiCell($w, self::LINE_HEIGHT, $data[$i], 0, $a);
            $this->SetTextColor(0);
        } else {
            $this->MultiCell($w, self::LINE_HEIGHT, $data[$i], 0, $a);
        }

        $this->SetXY($x + $w, $y);
    }
    $this->Ln($h);
}

public function addLineSeparator()
{
    $this->Ln(2);
    $this->Cell(0, 0.5, '', 'B', 1);
    $this->Ln(2);
}

public function Header()
{
    if ($this->pageHeaderRepeat || (!$this->pageHeaderRepeat && !$this->pageHeaderAdded)) {
```

```
        $path = public_path('assets/dist/img');

        $image = $path . "/avatar5.png";

        $image = str_replace("\\", "/", $image);
        $this->Image($image, $this->GetX(), $this->GetY(), 26);

        $this->SetFont('', 'B', 14);
        $this->Cell(0, 7, 'Patient Management System', 0, 0, 'C');
        $this->Ln();

        $this->SetFont('', 'B', 12);
        $this->Cell(0, 7, 'Peshawar, Pakistan. Tell: 091-00000000-00', 0, 0, 'C');
        $this->Ln();
        $this->Cell(0, 7, $this->reportTitle, 0, 0, 'C');
        $this->Ln();

        if ($this->subTitle != '') {
            $this->Cell(0, 7, $this->subTitle, 0, 0, 'C');
            $this->Ln();
        }

        if ($this->fromDate != '' && $this->toDate != '') {
            $this->SetFont('', 'B', 10);
            $this->Cell(0, 7, 'From ' . $this->fromDate . '  To  ' . $this->toDate, 0, 0, 'C');
        } else if ($this->fromDate != '' && $this->toDate == '') {
            $this->SetFont('', 'B', 10);
            $this->Cell(0, 7, $this->fromDate, 0, 0, 'C');
        }
        $this->Ln(10);
        $this->SetFont('', 'B', 10);
        $this->pageHeaderAdded = true;
    }
}

public function Footer()
{
    if ($this->isFooterRequired) {
        $this->SetY(-11);
        $this->SetFont('', 'B', self::BODY_SIZE);
        $this->Cell(0, 6, 'Page ' . $this->PageNo() . ' / {nb}', 0, 0, 'C');
    }
}

public function SetWidths($w)
{
    $this->widths = $w;
}
```

```
public function SetAligns($a)
{
    $this->aligns = $a;
}

private function CheckPageBreak($h)
{
    if ($this->GetY() + $h > $this->PageBreakTrigger) {
        $this->AddPage($this->CurOrientation);

        $currentWidths = $this->widths;
        $currentAligns = $this->aligns;
        $this->widths = $this->tblHdrWidths;
        $this->aligns = $this->tblHdrAligns;

        $this->AddTableHeader(
            $this->colsWithoutSpan1,
            $this->colsWithSpan1,
            $this->colsWithoutSpan2,
            $this->colsWithSpan2,
            $this->colsWithoutSpan3,
            $this->colsWithSpan3,
            $this->colsWithoutSpan4,
            $this->colsWithSpan4
        );
        $this->widths = $currentWidths;
        $this->aligns = $currentAligns;
    }
}

private function NbLines($w, $txt)
{
    $cw = &$this->CurrentFont['cw'];
    if ($w == 0) {
        $w = $this->w - $this->rMargin - $this->x;
    }
    $wmax = ($w - 2 * $this->cMargin) * 1000 / $this->FontSize;
    $s = str_replace("\r", '', $txt);
    $nb = strlen($s);
    if ($nb > 0 and $s[$nb - 1] == "\n") {
        $nb--;
    }

    $sep = -1;
    $i = 0;
    $j = 0;
    $l = 0;
    $nl = 1;
    while ($i < $nb) {
```

```
            $c = $s[$i];
            if ($c == "\n") {
                $i++;
                $sep = -1;
                $j = $i;
                $l = 0;
                $nl++;
                continue;
            }
            if ($c == ' ')
                $sep = $i;
            $l += $cw[$c];
            if ($l > $wmax) {

        if ($sep == -1) {
                    if ($i == $j) {
                        $i++;
                    }
                } else {
                    $i = $sep + 1;
                }
                $sep = -1;
                $j = $i;
                $l = 0;
                $nl++;
            } else {
                $i++;
            }
        }
        return $nl;
    }

    private function AddTableHeaderCells($colIndex, $cols, $spannedCols)
    {
        for ($i = 0; $i < count($cols); $i++) {
            $this->AddHdrCell($this->widths[$colIndex], self::LINE_HEIGHT * 2, $cols[$i], $this-
>aligns[$colIndex]);
            $colIndex++;
        }
        if (count($spannedCols) > 0) {
            $spanningColWidth = 0;
            for ($i = 1, $j = $colIndex; $i < count($spannedCols); $i++, $j++) {
                $spanningColWidth = $spanningColWidth + $this->widths[$j];
            }
            for ($i = 0; $i < count($spannedCols); $i++) {
                if ($i == 0) {
                    $this->AddHdrCell($spanningColWidth, self::LINE_HEIGHT, $spannedCols[$i], 'C');
                    $this->SetXY($this->GetX() - $spanningColWidth, $this->GetY() +
self::LINE_HEIGHT);
```

```
            } else {
                $this->AddHdrCell($this->widths[$colIndex], self::LINE_HEIGHT, $spannedCols[$i],
$this->aligns[$colIndex]);
                $colIndex++;
            }
        }
        $this->SetXY($this->GetX(), $this->GetY() - self::LINE_HEIGHT);
    }
}

private function AddHdrCell($width, $height, $text, $align)
{
    $textHeight = self::LINE_HEIGHT;
    if ($height > self::LINE_HEIGHT && $this->NbLines($width, $text) == 1) {
        $textHeight = $height;
    }
    $x = $this->GetX();
    $y = $this->GetY();
    $this->Rect($x, $y, $width, $height, 'DF');
    $this->MultiCell($width, $textHeight, $text, 0, $align);
    $this->SetXY($x + $width, $y);
}

public function addVerticalLine($x1, $y1, $x2, $y2)
{
    $this->Line($x1, $y1, $x2, $y2);
}

/* alpha code starts here */

protected $extgstates = array();

/*
 * alpha: real value from 0 (transparent) to 1 (opaque)
 *  bm:    blend mode, one of the following:
 *          Normal, Multiply, Screen, Overlay, Darken, Lighten, ColorDodge, ColorBurn,
 *          HardLight, SoftLight, Difference, Exclusion, Hue, Saturation, Color, Luminosity
 */

function SetAlpha($alpha, $bm = 'Normal')
{
    // set alpha for stroking (CA) and non-stroking (ca) operations
    $gs = $this->AddExtGState(array('ca' => $alpha, 'CA' => $alpha, 'BM' => '/' . $bm));
    $this->SetExtGState($gs);
}

function AddExtGState($parms)
{
    $n = count($this->extgstates) + 1;
```

```
    $this->extgstates[$n]['parms'] = $parms;
    return $n;
}

function SetExtGState($gs)
{
    $this->_out(sprintf('/GS%d gs', $gs));
}

function _enddoc()
{
    if (!empty($this->extgstates) && $this->PDFVersion < '1.4')
        $this->PDFVersion = '1.4';
    parent::_enddoc();
}

function _putextgstates()
{
    for ($i = 1; $i <= count($this->extgstates); $i++) {
        $this->_newobj();
        $this->extgstates[$i]['n'] = $this->n;
        $this->_put('<</Type /ExtGState');
        $parms = $this->extgstates[$i]['parms'];
        $this->_put(sprintf('/ca %.3F', $parms['ca']));
        $this->_put(sprintf('/CA %.3F', $parms['CA']));
        $this->_put('/BM ' . $parms['BM']);
        $this->_put('>>');
        $this->_put('endobj');
    }
}

function _putresourcedict()
{
    parent::_putresourcedict();
    $this->_put('/ExtGState <<');
    foreach ($this->extgstates as $k => $extgstate)
        $this->_put('/GS' . $k . ' ' . $extgstate['n'] . ' 0 R');
    $this->_put('>>');
}

function _putresources()
{
    $this->_putextgstates();
    parent::_putresources();
}

/* alpha code ends here */
```

```
    public function setACRepeatingTableCaption($accountRepeatingTableCaption)
    {
        $this->accountRepeatingTableCaption = $accountRepeatingTableCaption;
    }

}
```