PDF Template Editor Requirements/Milestones

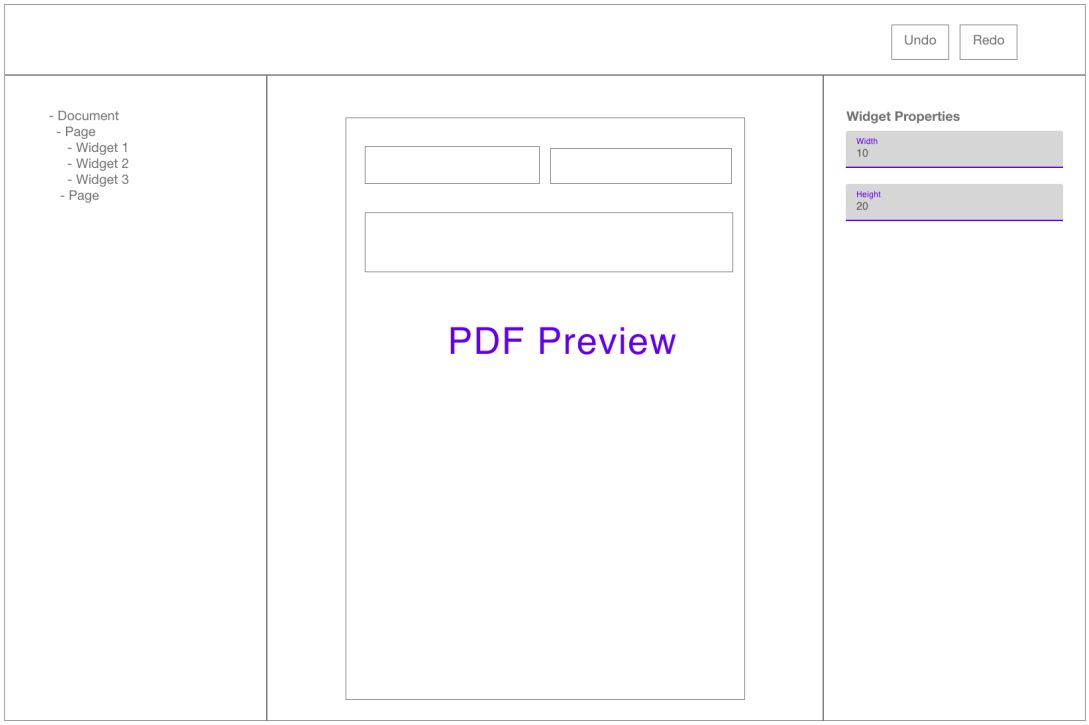
Visual Systems Corporation
2022-08-25

Requirements

These are the ultimate set of requirements for the PDF Template Editor. However, we will start with a basic editor and then build-up to meet these requirements.

- UX - basically a document/widget tree on the left, a preview in the middle, and widget properties on the right. See Rough Wireframe below.
- Initially usable for a designer or developer. It does not need to be usable for a non-technical end-user.
- The editor must be developed as a Flutter widget which could be embedded in another application.
- The PDF template will be stored in an abstract JSON format which can be transformed to PDF using the pdf package (https://pub.dev/packages/pdf) using its PDF Widgets library.
- The PDF transformer should be Dart compatible (no Flutter dependencies) so a server can generate a PDF.
- Images can be URLs or embedded in the template.
- The preview pane is generated with the https://pub.dev/packages/printing package. Sample JSON data can be supplied with the template. For simplicity, editor mode will always preview/refresh the rendered PDF. The user will only add/remove widgets in the tree view.
- Can add explicit pages to document, or use multi-page to handle overflow.
- Must support undo/redo
- In-line text styling (via tree of PDF widgets)
- Repeating group (iterates over JSON data list items)
- Literals or expressions in property values. The user can toggle between whether a property value is a literal (string, number, etc.), or an expression.
- Usable as a web or Linux desktop app.
- Will be open source. VSC will create a Github repo where the development will take place. Code is expected to be committed frequently - at least once a day, but more often is preferable.
- Font selection for text will ultimately be based on `PdfGoogleFonts` in the “printing” package.

Rough Wireframe



Milestones

Milestone 1

Goal: Get a basic editor widget working with a few basic PDF widgets and an example app created. You will only be able to add widgets to the document tree view. There will be no drag and drop to the visual layout. It should have a separate Dart-compatible transformer which converts the JSON template and JSON data to a PDF. At this phase, it only needs to support the canned set of JSON invoice data provided.

We need to support basic layout and basic content widgets so we can demonstrate that the editor is working.

PDF widgets and classes supported for this milestone:

- Border - value
- BorderRadius - value
- BorderSide - value
- BorderStyle - value
- BoxBorder - value
- BoxConstraints - value
- BoxDecoration - value - It might be ok to just have some canned selectable set of these. Like a single-pixel width border, and maybe a fixed rounded corner one
- BoxShadow - value
- Center - layout
- Column - layout
- Container - layout
- Divider - content

- EdgeInsets - value
- Expanded - layout
- FixedColumnWidth - value
- Font - value
- FullPage - layout/container
- Header - content - part of page declaration or in-line and indexed by TableOfContent
- MultiPage - should be used to construct PDF, but not used directly in the editor.
- NewPage - content widget causes a page break (I think)
- Padding - container/layout
- Repeater - not a PDF widget class, but our container “widget” (might not be an actual widget?) - loops over JSON list data and builds repeated child widget trees for each item in the list. This should feed widgets which support multiple children, such as column or row.
- Placeholder - content widget - paints an X filling the space
- Row - layout
- SizedBox - layout
- Spacer - content
- Text - content
- TextDecoration - value
- TtfFont - value

For this milestone, we will need to support some property value expressions for things like Repeater (you need to get a hold of the JSON list) and to get values for text. This can be supported by the https://pub.dev/packages/expressions package which allows for Javascript-like expressions so that lists or values can be extracted from the JSON data. We should do the simplest thing that will make this work.

It would be great if the widgets could be defined in some kind of metadata which would make adding new widgets easier. In other words, it would be nice not to have to write editor code that is specific to each widget. An initial task for this milestone would be to come up with a proposal for how this would work. If it seems like it will introduce a lot of work, we may not do this immediately.

Milestone 2

Support expressions formatting functions in property values. These functions would support formatting of dates/times and numeric values.

We will also support the next set of PDF widgets and classes which support additional layout and content:

- Align - Layout
- Alignment - Value
- AspectRatio - Layout
- BoxDecoration - value - complete this
- Checkbox
- ConstrainedBox
- DecoratedBox - widget/container
- DecorationGraphic
- DecorationImage

- DecorationSvgImage
- FittedBox - layout
- FittedSizes - value
- FlatButton - widget
- Flex - layout
- Flexible - wrapper widget
- Footer - widget, but is attached to a page
- FractionalOffset - value
- FractionColumnWidth - value -table column
- FlexColumnWidth - value
- IntrinsicColumnWidth - value for tables
- Gradient - value
- LinearGradient - value
- GridView
- Image - content widget
- ImageImage - provider - wrapper for flutter image
- ImageProvider - superclass provider
- ImageProxy - proxy to a PdfImage
- MemoryImage - in-memory image provider - used when a JSON template contains an embedded image, or data contains a base64 encoded image.
- RawImage - in-memory image provider - used when a JSON template contains an embedded image. Figure out difference between this and MemoryImage
- SvgImage - image provider. SVG should be embedded or URL.
- LimitedBox - layout
- Link
- ListView
- Paragraph
- Positioned - layout ala Flutter
- RadialGradient - value
- Radius - value
- Stack - layout
- Table
- TableBorder
- TableColumnWidth - value
- TableRow
- TextStyle
- UrlLink
- VerticalDivider
- Watermark
- Wrap - layout

Milestone 3

Support extended widgets and charting:

Extended widget support

- Bullet - Widget (renders a bullet list item)
- Icon - widget
- IconData - value

- Opacity - layout/container
- PageTheme - set at page level
- Partition
- Partitions
- RichText
- InlineSpan - span of text for rich text (See TextSpan and WidgetSpan). Widgets can be embedded in text...
- TableOfContent - content widget
- TextField
- TextSpan
- WidgetSpan

Charting

- BarDataSet<T extends PointChartValue> - data for charts
- CartesianGrid - Widget - a chart widget contained in a Chart
- Chart
- ChartGrid
- ChartLegend
- ChartValue
- Dataset
- FixedAxis<T extends num>
- LineChartValue
- LineDataSet<T extends PointChartValue>
- PieDataSet
- PieGrid
- PointChartValue
- PointDataSet<T extends PointChartValue>
- RadialGrid - chart

Milestone 4

Support graphics and barcodes:
Graphics

- Circle - Widget renders a circle
- ClipOval
- ClipRect
- ClipRRect
- GridAxis - widget
- GridPaper - widget
- Polygon
- Rectangle
- Shape
- Transform

Barcode and misc. support

- Barcode - Widget for barcodes
- BarcodeBar - Widget for barcodes
- BarcodeEan - Widget for barcodes
- BarcodeElement - Widget for barcodes

- BarcodeText - Widget for barcodes
- BarcodeWidget - Widget for barcodes
- DataMatrixEncoder
- Signature - this is a PDF signature, not a human signature
- Theme

Development Process

The development process will be informal. We are a small company and you will work directly with me (Dan). We'll host a Trello board where we can track tasks, bugs, and milestones. We will follow a Scrum two week sprint. We'll roughly plan out the tasks for the sprint at the start of the sprint. However, we probably don't need daily stand-ups. We can just use chat and email throughout the day, so I should always have an idea of the status.

Commits to the repo will be done at least daily on a branch you create. When a set of functionality is ready, you will submit a PR for my review.

Timeline

We would like at least Milestone 1 completed within the first month of the contract. From there, we'll estimate timelines for the subsequent milestones.