CVDL: CV Description Language

Plan

The current plan is to start by creating a command-line tool that takes a **document definition** and generates a PDF as the result.

The **document definition** and related concepts are provided below.

For the first phase, we will start with a set of implicitly encoded schemas and refine the code for relying on explicitly provided schemas.

Concepts

Document Definition

A document is defined by combining some set of schemas, some data conforming to the given schema, and the required layouts.

Section Data Schema

A section schema defines elements and their types for a section.

Example:

```
{
    "schema-name": "Education",
    "element-schema": [
        { "name": "School", "type": "String" },
        { "name": "Degree", "type": "String" },
        { "name": "Department", "type": "String" },
        { "name": "Date-Started", "type": "Date" },
        { "name": "Date-Finished", "types": ["Date", "String"] },
        { "name": "Location", "type": "String" },
        { "name": "Text", "type": "MarkdownString" },
    ]
}
```

Each section will have conform to a schema. `data-schemas.json` file has some example schemas.

Section Layout Schema

A section layout schema defines a mapping from a section into a box. A box is a theoretical 2d space, constructed of a composition of smaller boxes. The language for expressing layouts is WIP at the moment. Instead, I will provide an informal description written in Latex style below.

```
{Degree} - {Department} \hfill {Date-Started} - {Date-Finished} \newline
{School} \hfill {Location} \newline
{Text}
```

You will also be able to provide rendering directives.

```
{Degree: Italic} - {Department: Bold} \hfill {Date-Started: Year-Only} -
{Date-Finished: Year-Only} \newline
{School} \hfill {Location: Cut-First} \newline
{Text}
```

Rendering Directive

A rendering directive is a function from a datatype T into a box. This is the reason the Latex like model above does not capture our full intuitions. All elements are rendered into boxes, and the layout schema composes a box. All types have a default rendering directive that is used when no other directive is provided.

There are more complex rendering directives available that composes several datatypes into a box, which is implicitly encoded in the latex above.

```
{Degree: Italic} - {Department: Bold}
```

We see the dash(-) between two elements, which we know are actually boxes. So, what does dash mean there? If we desugared the expression we would get a function call similar to below.

```
dashBetween(italic(degree), bold(department))
```

Element Types

An element type must have a name and a default rendering directive. Optionally, element type may contain validators and addtional rendering directives. All elements are represented as strings, and it is up to the rendering directives to do the parsing/deserialization/decoding. As an example, String will be wrapped at box bounds by the default rendering directive, MarkdownString will be rendered by respecting italic/link/bold annotations.

Section Definitions

A section is defined by its name, the schema it conforms to, and its elements.

Example:

```
{
    "section-name": "Education",
    "schema": "Education",
    "elements": [
        {
            "School": "University of Maryland, College Park",
            "Degree": "Doctorate of Philosophy",
            "Department": "Computer Science",
            "Date-Started": "2021",
            "Date-Finished": "2026(Expected)",
            "Location": "Maryland, USA"
        },
        {
            "School": "Middle East Technical University",
            "Degree": "Bachelor of Engineering",
            "Department": "Computer Engineering",
            "Date-Started": "2017",
            "Date-Finished": "2021",
            "Location": "Ankara, Turkey",
            "Text": "GPA: 3.66/4.0 ***(top 5% in class of 229)***"
        }
    ]
}
```

Document Layout Schema

A document will have several sections, and it will need to compute its layout by putting those sections together. We know that a section has a name and a list of elements defined by the `element-schema`. We also know that each element can be rendered into a standalone box. We can vertically compose elements into a larger box, add section name on top, which will give us the section box. Now, all the document has to do is specify how the layers are composed. The language for this layout specification is also WIP, but the idea informally looks something like below.

```
| Profile |
| Education |
| Hobbies : 50% | %10 | Talents : 40% |
```

Each line above is a layer of a stack, and when there is more than one section in a line, that means we have multiple columns. We can have empty spaces between columns, and we can define the length of columns.