Component Documentation

This file contains documentation for the React components used in this project.

`DraggableSignature.tsx`

Description

This component provides a draggable and resizable signature element that can be placed on a document. It also supports rotation. The component displays an image of the signature and allows the user to interact with it by dragging, resizing, and rotating.

Props

Prop	Type	Description
`data`	`SignatureAnnotation`	An object containing the signature's data, such as its ID, position, dimensions, and image.
`onUpdate`	`(id: string, patch: Partial<SignatureAnnotation>) => void`	A function to call when the signature's data is updated.
`onDelete`	`(id: string) => void`	A function to call when the signature is deleted.
`onSelect`	`(id: string) => void`	A function to call when the signature is selected.
`scale`	`number`	The current scale of the document viewer.
`isSelected`	`boolean`	A boolean indicating whether the signature is currently selected.

Example

```
<DraggableSignature
  data={{
    id: 'signature-1',
    pageIndex: 0,
    image: '...',
    x: 100,
    y: 100,
    width: 150,
    height: 60,
    rotation: 0,
  }}
  onUpdate={(id, patch) => console.log('Update signature:', id, patch)}
  onDelete={(id) => console.log('Delete signature:', id)}
  onSelect={(id) => console.log('Select signature:', id)}
  scale={1.5}
  isSelected={true}
/>
```

`DraggableTextPro.tsx`

Description

This component provides a professional-looking, draggable, resizable, and rotatable text box that can be placed on a document. It allows for in-place editing of the text and supports various formatting options, such as font size, color, bold, italic, underline, and alignment.

Props

Prop	Type	Description
`data`	`PlacedText`	An object containing the text's data, such as its ID, content, position, and formatting.

Prop	Type	Description
onUpdate	(id: string, patch: Partial<PlacedText>) => void	A function to call when the text's data is updated.
onDelete	(id: string) => void	A function to call when the text is deleted.
onSelect	(id: string) => void	A function to call when the text is selected.
scale	number	The current scale of the document viewer.
isSelected	boolean	A boolean indicating whether the text is currently selected.

Example

```
<DraggableTextPro
  data={{
    id: 'text-1',
    text: 'Hello, world!',
    x: 100,
    y: 100,
    fontSize: 14,
    pageIndex: 0,
    color: '#000000',
  }}
  onUpdate={(id, patch) => console.log('Update text:', id, patch)}
  onDelete={(id) => console.log('Delete text:', id)}
  onSelect={(id) => console.log('Select text:', id)}
  scale={1.5}
  isSelected={true}
/>
```

SelectionHandles.tsx

Description

This component provides the selection handles (resize and rotate) for a draggable element. It displays 8 resize handles around the element and a rotation handle at the top. It also includes a delete button.

Props

Prop	Type	Description
rotation	number	The current rotation of the element in degrees.
onResize	(direction: string) => (e: React.MouseEvent) => void	A function to call when a resize handle is dragged.
onRotate	(e: React.MouseEvent) => void	A function to call when the rotation handle is dragged.
onDelete	() => void	A function to call when the delete button is clicked.

Example

```
{isSelected && (
  <SelectionHandles
    rotation={rotation}
    onResize={handleResize}
    onRotate={handleRotate}
    onDelete={() => onDelete(data.id)}
  />
)}
```

PDFViewer.tsx

Description

This component is a full-featured PDF viewer that allows for rendering PDF documents, navigating between pages, and adding annotations such as text, signatures, and shapes. It also supports zooming and fitting the document to the screen.

Props

Prop	Type	Description
`activeTool`	`'text' | 'signature' | 'shapes' | null`	The currently active tool for adding annotations.
`inputText`	`string`	The text to be inserted when the text tool is active.
`setInputText`	`(text: string) => void`	A function to update the `inputText`.
`signature`	`string | null`	The data URL of the signature image to be inserted.
`isSignatureFloating`	`boolean`	A boolean indicating whether the signature is floating.
`setIsSignatureFloating`	`(v: boolean) => void`	A function to update the `isSignatureFloating` state.
`textFormat`	`TextFormat`	An object containing the formatting for the text tool.
`selectedShape`	`ShapeType | null`	The currently selected shape to be inserted.
`isShapeFloating`	`boolean`	A boolean indicating whether the shape is floating.
`setIsShapeFloating`	`(v: boolean) => void`	A function to update the `isShapeFloating` state.
`onExitAddText`	`() => void`	A function to call when the text/signature/shape adding mode is exited.
`selectedAnnotationId`	`string | null`	The ID of the currently selected annotation.
`onSelectAnnotation`	`(id: string | null) => void`	A function to call when an annotation is selected.

Example

```
<PDFViewer
  activeTool="text"
  inputText="Hello"
  setInputText={setInputText}
  signature={null}
  isSignatureFloating={false}
  setIsSignatureFloating={setIsSignatureFloating}
  selectedAnnotationId={selectedAnnotationId}
  onSelectAnnotation={setSelectedAnnotationId}
/>
```

`CropTool.tsx`

Description

This component provides a tool for cropping and rotating an image. It displays the image with a crop box that can be resized and moved. It also includes a rotation handle to rotate the image. The component can auto-detect the bounds of a signature within the image.

Props

Prop	Type	Description
`currentImage`	`string`	The data URL of the image to be cropped.
`onApplyCrop`	`(croppedDataUrl: string) => void`	A function to call when the crop is applied.

Prop	Type	Description
`onToggleTilt`	`() => void`	A function to call when the tilt tool is toggled.
`onExit`	`() => void`	A function to call when the crop tool is exited.

Example

```
<CropTool
  currentImage="..."
  onApplyCrop={(croppedImage) => console.log('Cropped image:', croppedImage)}
  onToggleTilt={() => console.log('Toggle tilt')}
  onExit={() => console.log('Exit crop tool')}
/>
```

`SignaturePreview.tsx`

Description

This component displays a preview of the original and processed signature images. It allows the user to download the final signature, reset the changes, and access the crop and tilt tools.

Props

Prop	Type	Description
`originalImage`	`string`	The data URL of the original signature image.
`processedImage`	`string`	The data URL of the processed signature image.
`onDownload`	`() => void`	A function to call when the download button is clicked.
`onReset`	`() => void`	A function to call when the reset button is clicked.

Example

```
<SignaturePreview
  originalImage="..."
  processedImage="..."
  onDownload={() => console.log('Download signature')}
  onReset={() => console.log('Reset signature')}
/>
```

`PDFEditor.tsx`

Description

This is the main component for the PDF editor. It integrates the `PDFViewer`, `PDFThumbnails`, and various toolbars to provide a complete PDF editing experience. It manages the active tool, annotations, and other state related to the editor.

Props

This component does not accept any props.

Example

```
<PDFEditor />
```

`ImageFilterEditor.tsx`

Description

This component provides an editor for applying filters to an image, specifically for enhancing a signature. It includes controls for brightness and contrast, and it automatically processes the image to create a clean, transparent signature.

Props

Prop	Type	Description
`imageSrc`	`string`	The data URL of the image to be edited.
`onDone`	`(dataURL: string) => void`	A function to call when the editing is complete.
`onCancel`	`() => void`	A function to call when the editing is canceled.

Example

```
<ImageFilterEditor
  imageSrc="..."
  onDone={(editedImage) => console.log('Edited image:', editedImage)}
  onCancel={() => console.log('Cancel editing')}
/>
```

`TiltCorrectionTool.tsx`

Description

This component provides a tool for correcting the perspective tilt of an image in 3D. It allows the user to adjust the horizontal and vertical tilt of the image using sliders.

Props

Prop	Type	Description
`processedImage`	`string`	The data URL of the image to be corrected.
`onApply`	`(correctedImage: string) => void`	A function to call when the correction is applied.
`onCancel`	`() => void`	A function to call when the correction is canceled.

Example

```
<TiltCorrectionTool
  processedImage="..."
  onApply={(correctedImage) => console.log('Corrected image:', correctedImage)}
  onCancel={() => console.log('Cancel correction')}
/>
```

`PDFThumbnails.tsx`

Description

This component displays thumbnails of the pages in a PDF document. It allows the user to navigate between pages by clicking on the thumbnails.

Props

This component does not accept any props.

Example

```
<PDFThumbnails />
```

TextSignature.tsx

Description

This component allows users to create a text-based signature. It provides a text input for the user to type their name and a selection of fonts to choose from. The component then generates an image of the signature.

Props

Prop	Type	Description
`onGenerate`	`(dataURL: string) => void`	A function to call when the signature is generated.
`closeModal`	`() => void`	A function to call to close the modal.

Example

```
<TextSignature
  onGenerate={(dataURL) => console.log('Signature generated:', dataURL)}
  closeModal={() => console.log('Close modal')}
/>
```

shapes-tool.tsx

Description

This file contains two components: `ShapesTool` and `ShapeRenderer`.

`ShapesTool` provides a UI for selecting a shape to add to the document.

`ShapeRenderer` is responsible for rendering the selected shape on the PDF.

Props

ShapesTool

Prop	Type	Description
`onShapeSelect`	`(shape: ShapeType) => void`	A function to call when a shape is selected.

ShapeRenderer

Prop	Type	Description
`shape`	`ShapeAnnotation`	The shape annotation object to render.
`scale`	`number`	The current scale of the document viewer.

Example

```
// ShapesTool
<ShapesTool onShapeSelect={(shape) => console.log('Shape selected:', shape)} />

// ShapeRenderer
<ShapeRenderer shape={shapeAnnotation} scale={1.5} />
```

`IslandCard.tsx`

Description

This component displays a card for a step in the IR pathway roadmap. It shows the step number, title, subtitle, and badges for the responsible person and timeline. The card's appearance changes based on whether the step is completed.

Props

Prop	Type	Description
`step`	`Step`	The step data to display.
`isCompleted`	`boolean`	A boolean indicating if the step is completed.
`onClick`	`() => void`	A function to call when the card is clicked.

Example

```
<IslandCard
  step={step}
  isCompleted={true}
  onClick={() => console.log('Card clicked')}
/>
```

`CompletionDialog.tsx`

Description

This component displays a dialog to congratulate the user upon completing all the steps in the IR pathway roadmap. It shows the user's progress and some achievement badges.

Props

Prop	Type	Description
`open`	`boolean`	A boolean to control the visibility of the dialog.
`onOpenChange`	`(open: boolean) => void`	A function to call when the dialog's open state changes.
`progress`	`number`	The user's progress percentage.

Example

```
<CompletionDialog
  open={true}
  onOpenChange={setOpen}
  progress={100}
/>
```

`compress-pdf.tsx`

Description

This component provides a UI for compressing a PDF file. It allows the user to select a PDF file, sends it to a server for compression, and then provides a download link for the compressed file.

Props

This component does not accept any props.

Example

```
<Compress />
```