Implementation Plan: Roolz Core (Document Upload, View, Edit)

Date: March 24, 2025
Platform: Replit (utilizing Replit AI for automation and optimization)
Focus: Core document upload, viewing, and editing capabilities

---

Objective

Build a basic version of Roolz where users can upload a PDF, view it, and add simple annotations (e.g., text or signatures), styled with beNext.io branding, using Replit AI to simplify the process.

Step-by-Step Plan

Step 1: Set Up the Replit Project

- **What to Do:**
  - Create a new Replit project using the "React + Node.js" template.
  - Name it "Roolz-Core".
- **Replit AI Task:**
  - Ask Replit AI: "Set up a React project with Node.js backend and install Tailwind CSS for styling."
  - AI will generate the base structure (`src/` for frontend, `server/` for backend) and configure Tailwind.
- **Time:** ~5 minutes.

Step 2: Add beNext.io Branding

- **What to Do:**
  - Style the app with beNext.io's colors: deep blue (#1E3A8A) background, teal (#2DD4BF) buttons, white (#FFFFFF) text.
  - Use a clean font like Roboto (available via Google Fonts).
- **Replit AI Task:**

    - Ask Replit AI: "Generate Tailwind CSS styles with a deep blue background, teal buttons, and Roboto font for a React app."
    - Copy the generated CSS into `src/index.css` or a new `styles.css` file.
    - Update `App.js` to include a simple header like `<h1 className="text-white">Roolz</h1>`.
- **Time:** ~10 minutes.

Step 3: Enable Document Upload

- **What to Do:**
    - Add a drag-and-drop area to upload PDFs and store them temporarily in memory (we'll add AWS S3 later).
    - Application must be responsive, working in mobile, tablet, desktop.
- **Replit AI Task:**
    - Ask Replit AI: "Create a React component with a drag-and-drop file upload for PDFs."
    - AI will generate a component (e.g., `UploadArea.js`) using HTML5 File API.
    - Add it to `App.js` with a state to hold the uploaded file:
    - `jsx`
    - const [pdfFile, setPdfFile] = useState(null);
    - Connect the upload: `<UploadArea onFileUpload={setPdfFile} />`.
- **Time:** ~15 minutes.

Step 4: Set Up PDF Viewing with PDF.js

- **What to Do:**
    - Display the uploaded PDF in the browser using PDF.js.
- **Replit AI Task:**
    - Ask Replit AI: "Add PDF.js to a React app to view a PDF file from state."
    - Install PDF.js: Run `npm install pdfjs-dist` in Replit's terminal.
    - AI will generate a `PDFViewer.js` component. Add the worker file:
        - Download `pdf.worker.min.js` from a CDN (e.g., unpkg.com) and place it in `public/`.
        - Set `pdfjs.GlobalWorkerOptions.workerSrc = '/pdf.worker.min.js';`.
    - Use it in `App.js`:
    - `jsx`
    - {pdfFile && <PDFViewer file={pdfFile} />}
- **Time:** ~20 minutes.

Step 5: Add Basic Editing with Fabric.js

- **What to Do:**
    - Overlay a canvas on the PDF to add text or a signature.
- **Replit AI Task:**
    - Ask Replit AI: "Create a Fabric.js canvas over a PDF.js viewer in React for text annotations."
    - Install Fabric.js: Run `npm install fabric` in the terminal.
    - AI will generate a `CanvasEditor.js` component. Combine it with `PDFViewer`:
    - `jsx`

```
const PDFEditor = ({ file }) => {
  const canvasRef = useRef(null);
  useEffect(() => {
    const canvas = new fabric.Canvas(canvasRef.current);
    canvas.add(new fabric.Textbox("Type here", { left: 100, top: 100, fill: "#2DD4BF" }));
  }, []);
  return (
    <div className="relative">
      <PDFViewer file={file} />
      <canvas ref={canvasRef} className="absolute top-0 left-0" />
    </div>
  );
```

    - };
    - Update `App.js`: Replace `<PDFViewer>` with `<PDFEditor file={pdfFile} />`.
- **Time:** ~25 minutes.

Step 6: Test and Polish

- **What to Do:**
    - Upload a PDF, view it, and add a teal-colored text annotation to confirm it works.
    - Ensure the UI looks clean with beNext.io styling.
- **Replit AI Task:**
    - Ask Replit AI: "Fix any layout issues with a PDF viewer and Fabric.js canvas in React."
    - AI will adjust CSS (e.g., `position: absolute`) to align the canvas over the PDF.
- **Time:** ~15 minutes.

Tools & Tech

- **Frontend:** React.js, Tailwind CSS (beNext.io styling).
- **Libraries:** PDF.js (viewing), Fabric.js (editing).
- **Replit Features:** AI code generation, built-in terminal, preview pane.