Report on :

Implementing PDF editing/ manipulation feature in our e-courts project

To achieve the above given task, we firstly need to choose a suitable js library which supports pdf viewing,highlighting,editing etc.

Some libraries which are suitable for our use are:

1. Pdf.js
2. Pdf-lib
3. JsPDF

Features of PDF.js which makes it suitable for our use

- Viewing: PDF.js allows you to render PDF files in the browser with a customizable viewer. You can display the PDF files uploaded by the super admin using PDF.js's viewer or integrate it into your own Angular components.

- Highlighting: PDF.js supports text selection, which means you can enable users to select text in the PDF and apply highlighting programmatically. You would need to implement UI controls to allow users to select text and trigger the highlighting functionality.

- Writing: PDF.js also supports adding annotations to PDF documents. You can use this feature to enable users to write or draw on the PDF files. Annotations can include text, shapes, and freehand drawings.

Implementing PDF.js in our project:

1. Installation - "npm install pdfjs-dist"

2. Import PDF.js into our angular component (document upload) and we need to import packages - import * as pdfjsLib from 'pdfjs-dist/build/pdf'; import 'pdfjs-dist/web/pdf_viewer';

3. Load and render the pdf file - We can create a method to handle it.

4. Create a canvas element in our components .html file - <canvas id="pdfViewer"></canvas>

5. Now we need to call the method with the url of the pdf file we want to render

For implementing writing and highlighting features:

Text Selection: Use PDF.js's text selection API to detect when the user selects text in the PDF. We can listen for the textlayerrendered event on the PDF viewer and then use JavaScript to capture the selected text.

Highlighting: Once the text is selected, overlay a semi-transparent highlight color over the selected text. We can achieve this by drawing a rectangle with the highlight color on the canvas element at the coordinates of the selected text.

Writing: Implement a UI for allowing users to input text (e.g., through a textbox or drawing tool). Capture the user input and draw the text onto the canvas element at the desired position. We may need to handle font styles, sizes, and positioning to ensure the text appears correctly on the PDF.

Saving Changes: Provide a way for users to save their highlighted text and annotations. You can serialize the changes made to the PDF (e.g., highlighted text positions, written annotations) and store them as metadata associated with the PDF file or as a separate data structure.

Keep in mind that implementing these features will require additional JavaScript code to handle user interactions and manipulate the canvas element where the PDF is rendered.