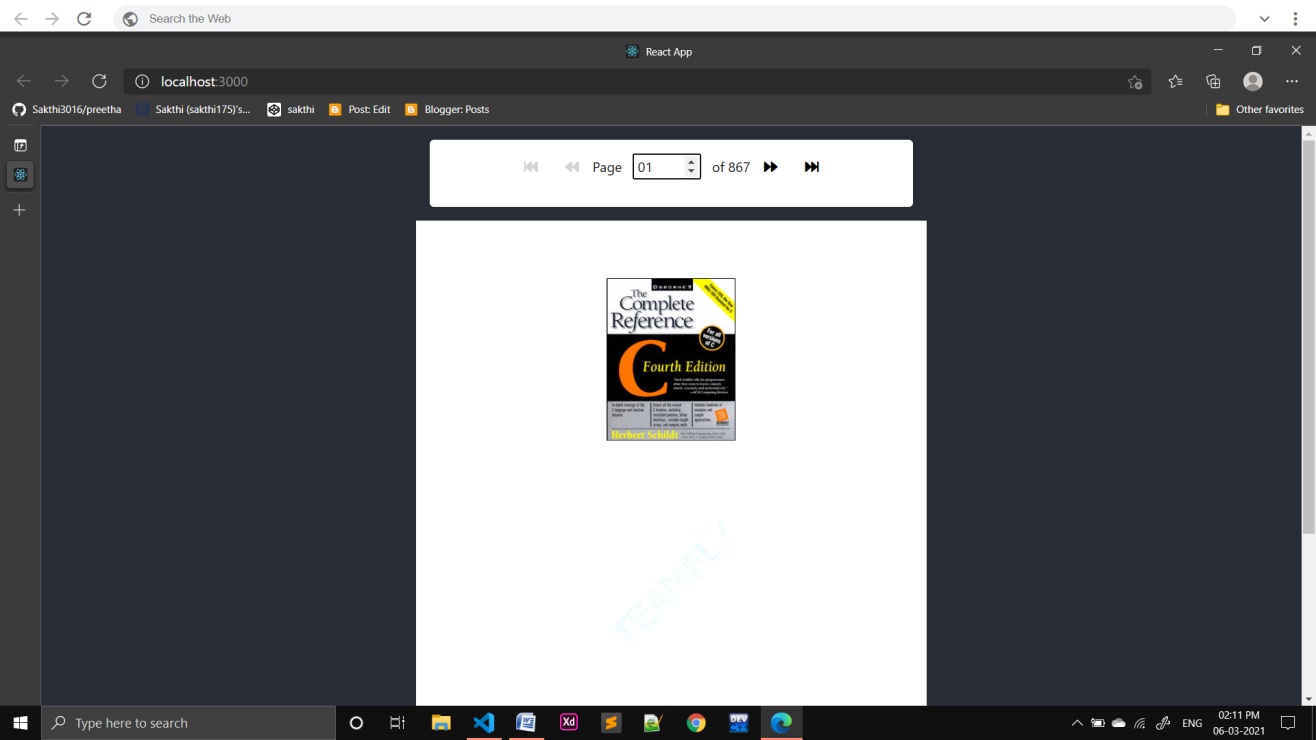
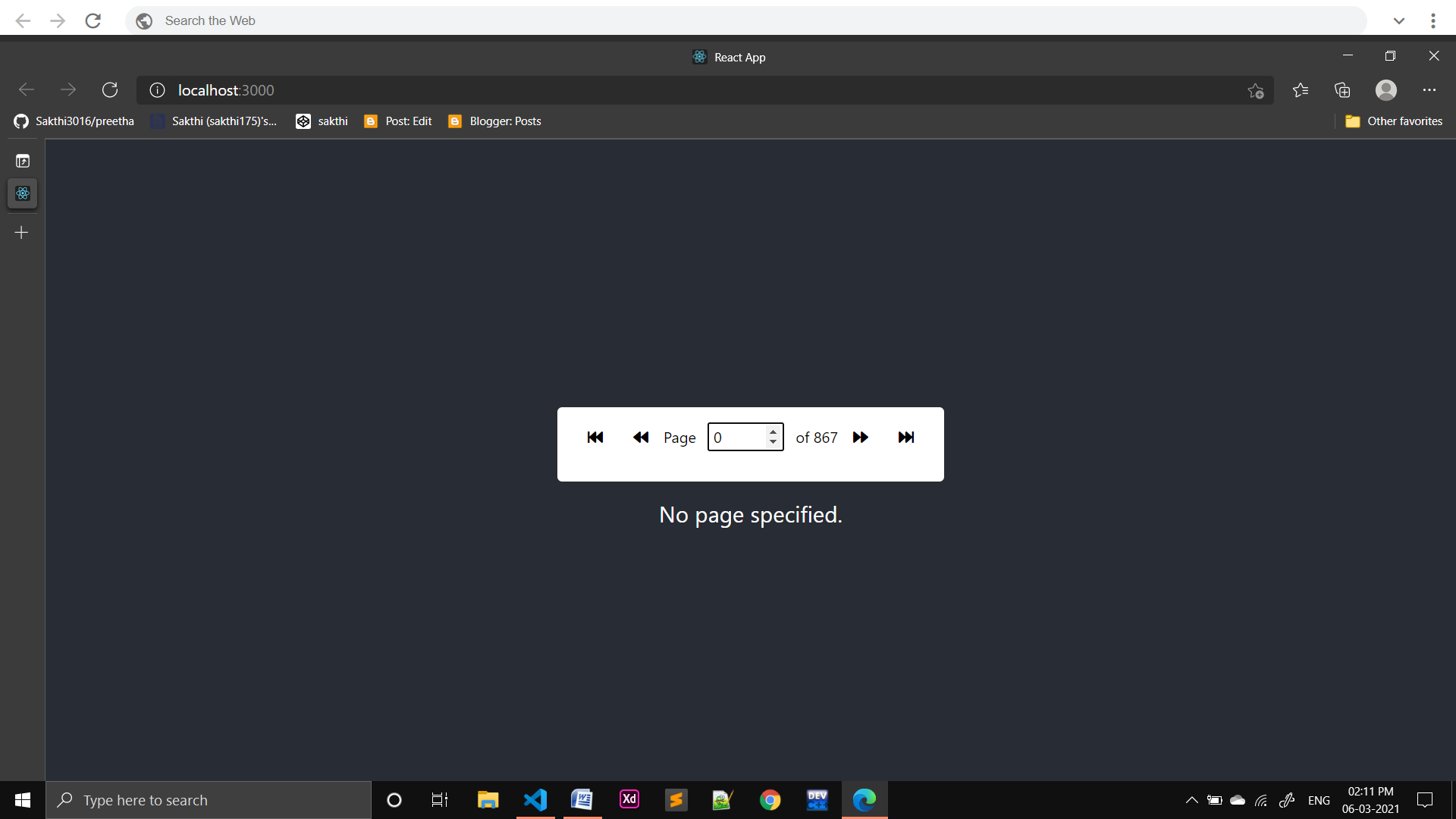
**PROBLEM STATEMENT-01**

**1.UI design**

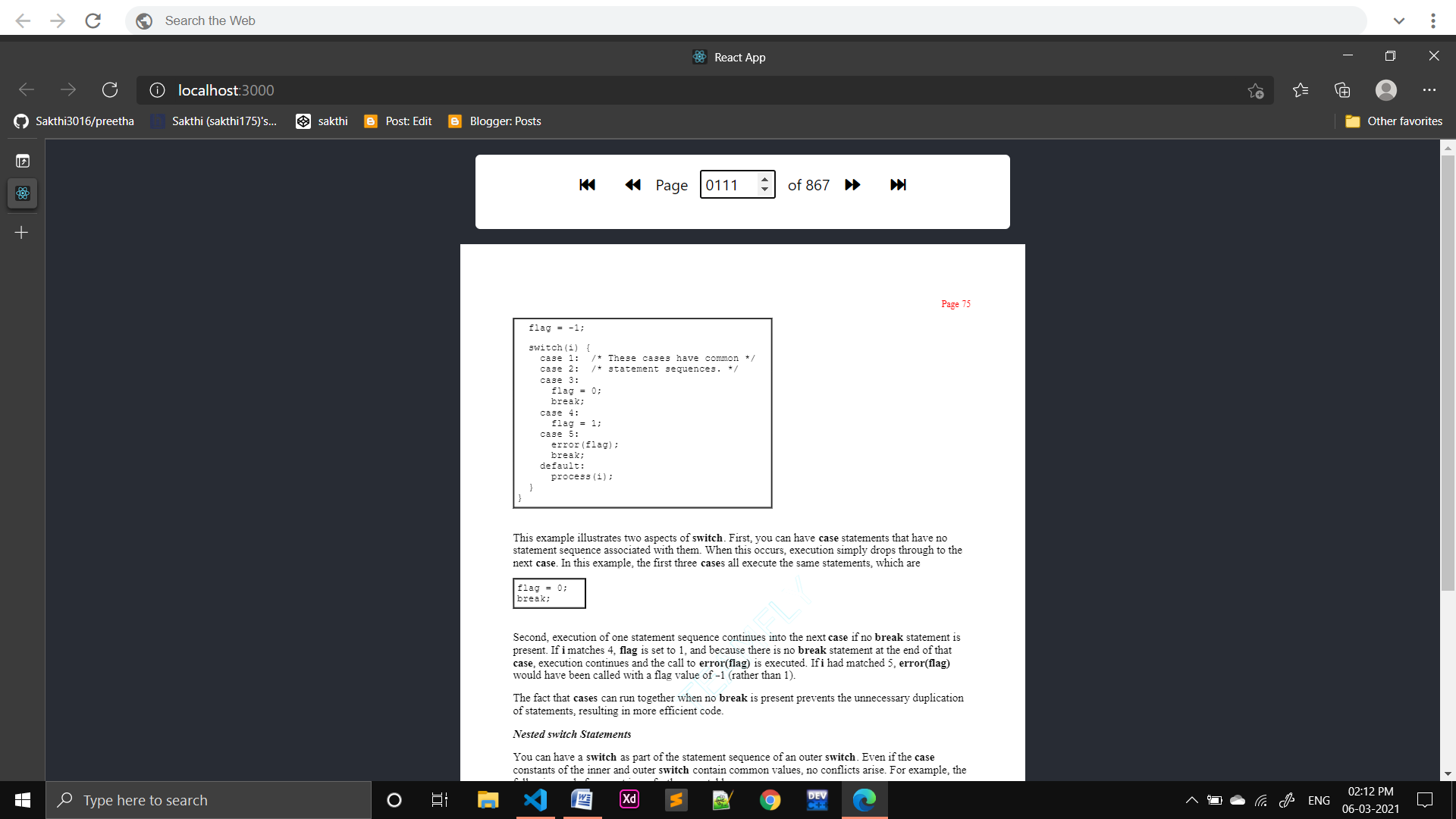
**Step1:Default value for page number is 1 and the previous menu is kept as default option.if given 0 page will be minimized will be shown in step-2**

****

**Step 2:If I gave pagenumber as 0**

****

**Step 3:Previous page next page first page and lastpage options are given and also if you give number it will redirect to that page.**

****

**SOURCE CODE:**

**PDFReader.jsx**

import React,{useState} from 'react'

import Loader from './Loader';

import {Document,Page,pdfjs} from 'react-pdf';

import Controlpanel from './Controlpanel';

pdfjs.GlobalWorkerOptions.workerSrc = `//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjs.version}/pdf.worker.min.js`;

const PDFReader=()=> {

  const [numPages, setNumPages] = useState(null);

  const [pageNumber, setPageNumber] = useState(1);

  const [isLoading,setIsLoading]=useState(true);

  function onDocumentLoadSuccess({ numPages }) {

    setNumPages(numPages);

    setIsLoading(false);

  }

  return (

    <div>

        <Loader isLoading={isLoading}/>

        <section id="pdf-section">

        <Controlpanel numPages={numPages} pageNumber={pageNumber} setPageNumber={setPageNumber}/>

    <Document

      file="/assets/docs/boot.pdf"

      onLoadSuccess={onDocumentLoadSuccess}

    >

      <Page pageNumber={pageNumber} />

    </Document>

    </section>

  </div>

  );

}

export default PDFReader;

**controlpanel.jsx**

import React from 'react'

const Controlpanel=(props)=>{

    const {pageNumber,numPages,setPageNumber}=props;

    const isFirstPage=pageNumber===1;

    const isLastPage=pageNumber===numPages;

    const firstPageClass=isFirstPage?'disabled':'clickable';

    const LastPageClass=isLastPage?'disabled':'clickable';

    const gotofirstpage=()=>{

        if(!isFirstPage) setPageNumber(1);

    };

    const gotopreviouspage=()=>{

        if(!isFirstPage) setPageNumber(pageNumber-1);

    };

    const gotonextpage=()=>{

        if(!isLastPage) setPageNumber(pageNumber+1);

    };

    const gotolastpage=()=>{

        if(!isLastPage) setPageNumber(numPages);

    };

        const onpagechange=(e)=>{

            const {value}=e.target;

            setPageNumber(Number(value));

        }

    return(

        <div className="control-panel m-3 p-3 d-flex align-items-baseline justify-content-center">

            <i className={`fas fa-fast-backward mx-3 ${firstPageClass}`} onClick={gotofirstpage}/>

            <i className={`fas fa-backward mx-3 ${firstPageClass}`} onClick={gotopreviouspage}/>

            <span>

    <p>Page <input name="pageNumber" type="number" min={1} max={numPages|1} className="p-o pl-1 mx-2" value={pageNumber} onChange={onpagechange}/> of {numPages}</p></span>

            <i className={`fas fa-forward mx-3 ${LastPageClass}`} onClick={gotonextpage}/>

            <i className={`fas fa-fast-forward mx-3 ${LastPageClass}`} onClick={gotolastpage}/>

        </div>

    )

}

export default Controlpanel;

**Loader.jsx**

import React from 'react'

const Loader=({isLoading})=>{

    if(!isLoading) return null;

    return(

        <div id="loader" className="d-flex justify-content-center align-items-center flex-column">

            <img src="https://react-pdf.org/images/logo.png" alt="loader" className="mb-5 App-logo"/>

            <p>Loading...</p>

        </div>

    )

}

export default Loader;

**index.css**

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

}

.control-panel{

  font-size: 16px;

  color: black;

  background-color: white;

  border-radius: 5px;

}

.clickable{

  cursor: pointer;

}

.disabled{

  cursor: not-allowed;

  color: lightgray;

}

**App.css**

.App {

  text-align: center;

  background-color: #282c34;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  font-size: calc(10px + 2vmin);

  color: white;

}

.App-logo {

  height: 30vmin;

  pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

  .App-logo {

    animation: App-logo-spin infinite 5s linear;

  }

}

@keyframes App-logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}

#loader{

  position: fixed;

  top: 0;

  left: 0;

  width:100vw;

  height: 100vh;

  background-color: rgba(155,155,155,0.7);

}

**Index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import 'bootstrap/dist/css/bootstrap.css'

import App from './App';

import reportWebVitals from './reportWebVitals';

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**PROBLEM STATEMENT-02**

**COMPARISON:**

**CHENEY’S ALGORITHM:**

* + It is a stop and copy method for tracing garbage collection in software system.
  + In this,heap is divided into two equal halves,but only one can be used at a time.Garbage collection is performed by copying (from-space) to other(to-space),which becomed new heap.
  + Some actions is performed whenever we refer object in from-space,some actions are
    - If object is not moved to (to-space) then copy of that object is created in to-space then replacing from-space with a forwarding pointer to the to-space.
    - If object is moved already to to-space then you just update forwarding pointer in from-space.
  + An algorithm needs no stack and only two pointers outside of from and to-space:  a pointer to the beginning of free space in the to-space, and a pointer to the next word in to-space. The forwarding pointer is used only during the garbage collection process

**MARK-COMPACT ALGORITHM:**

* + Mark-compact algorithm is a type of garbage collection algorithm used to reclaim unreachable memory.Mark-compact algorithm can be regarded as combination of the mark-sweep algorithm and Cheney’s copying algorithm.
  + The goal of mark-compact algorithms is to shift the live objects in memory together so the fragmentation is eliminated. The challenge is to correctly update all pointers to the moved objects, most of which will have new memory addresses after the compaction. The issue of handling pointer updates is handled in different ways.
  + Table-based compaction
  + LISP2 algorithm