

LABORATORY REPORT
Application Development Lab
(CS33002)

B.Tech Program in CSE

Submitted By

Name:- Rishi Banerjee

Roll No: 2205151



Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Exp No.	Title	Date of Experiment	Date of Submission	Remarks
1.	Build a Resume using HTML/CSS	16/01/2025	23/01/2025	
2.	Machine Learning for Cat and Dog Classification	23/01/2025	30/01/2025	
3.	Pneumonia Detection using CNN			
4.	Regression Analysis for Stock Prediction	30/01/2025	06/02/2025	
5.	Conversational Chatbot with Any Files	06/02/2025	20/02/2025	
6.				
7.				
8.				
9.	Open Ended 1			
10.	Open Ended 2			

Lab Number	6 & 7
Experiment Number	5
Experiment Title	Conversational Chatbot with Any Files
Date of Experiment	06/02/2025
Date of Submission	20/02/2025

1. Objective:-

2. To build a chatbot capable of answering queries from an uploaded PDF document.

3. Procedure:- (Steps Followed)

1. Integrate open-source LLMs such as LLama or Gemma from Ollama

2. Develop a backend to process the PDF content.

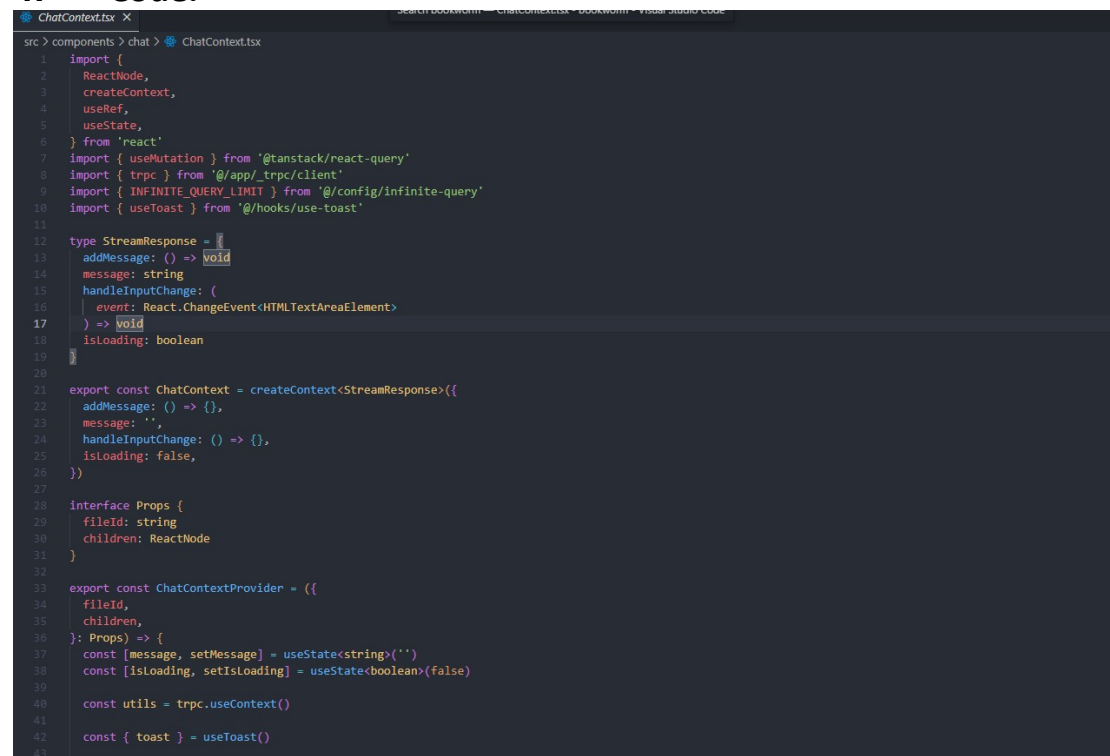
3. Implement Natural Language Processing (NLP) to allow queries. You can use LlamaIndex or Langchain

4. Create a frontend to upload document files and interact with the chatbot, just like OpenAI interface

5. Provide an option to choose the LLM model from a dropdown list.

6. Display the chatbot responses on the webpage.

4. Code:-



```

src > components > chat > ChatContext.tsx
1  import {
2    ReactNode,
3    createContext,
4    useRef,
5    useState,
6  } from 'react'
7  import { useMutation } from '@tanstack/react-query'
8  import { trpc } from '@app/_trpc/client'
9  import { INFINITE_QUERY_LIMIT } from '@config/infinite-query'
10 import { useToast } from '@hooks/use-toast'
11
12 type StreamResponse = {
13   addMessage: () => void
14   message: string
15   handleInputChange: (
16     event: React.ChangeEvent<HTMLTextAreaElement>
17   ) => void
18   isLoading: boolean
19 }
20
21 export const ChatContext = createContext<StreamResponse>({
22   addMessage: () => {},
23   message: '',
24   handleInputChange: () => {},
25   isLoading: false,
26 })
27
28 interface Props {
29   fileId: string
30   children: ReactNode
31 }
32
33 export const ChatContextProvider = ({
34   fileId,
35   children,
36 }: Props) => {
37   const [message, setMessage] = useState<string>('')
38   const [isLoading, setIsLoading] = useState<boolean>(false)
39
40   const utils = trpc.useContext()
41
42   const { toast } = useToast()
43

```

```

Dashboard.tsx X
src > components > Dashboard.tsx
1  'use client'
2
3  import { Ghost, Loader2, Plus, Trash } from "lucide-react"
4  import UploadButton from "../UploadButton"
5  import { trpc } from "@/_trpc/client";
6  import Skeleton from "react-loading-skeleton"
7  import Link from "next/link";
8  import { format } from "date-fns"
9  import { Button } from "../ui/button";
10 import { useState } from "react";
11 import { getUserSubscriptionPlan } from "@/_lib/stripe";
12
13 const getGradientByIndex = (index: number) => {
14   const colors = [
15     'from-orange-500 to-yellow-400',
16     'from-blue-500 to-purple-300',
17     'from-red-500 to-pink-300',
18     'from-green-500 to-teal-300',
19     'from-indigo-500 to-blue-300',
20     'from-purple-500 to-pink-300',
21     'from-teal-500 to-green-300',
22   ];
23   return colors[index % colors.length]; // Use modulo to cycle through colors
24 };
25
26 interface PageProps {
27   subscriptionPlan: Awaited<ReturnType<typeof getUserSubscriptionPlan>>
28 }
29
30 const Dashboard = ({subscriptionPlan}:PageProps) => {
31   const [currentlyDeletingFile, setCurrentlyDeletingFile] = useState<string | null>(null); // Defined without value
32
33   const utils = trpc.useUtils();
34
35   const { data: files, isLoading } = trpc.getUserFiles.useQuery();
36
37   const { mutate: deleteFile } = trpc.deleteFile.useMutation({
38     onSuccess: () => {
39       utils.getUserFiles.invalidate();
40     },
41     onMutate: ({ id }) => {
42       setCurrentlyDeletingFile(id);
43     },
44     onSettled: () => {
45       setCurrentlyDeletingFile(null);
46     }
47   });
48
49   return (

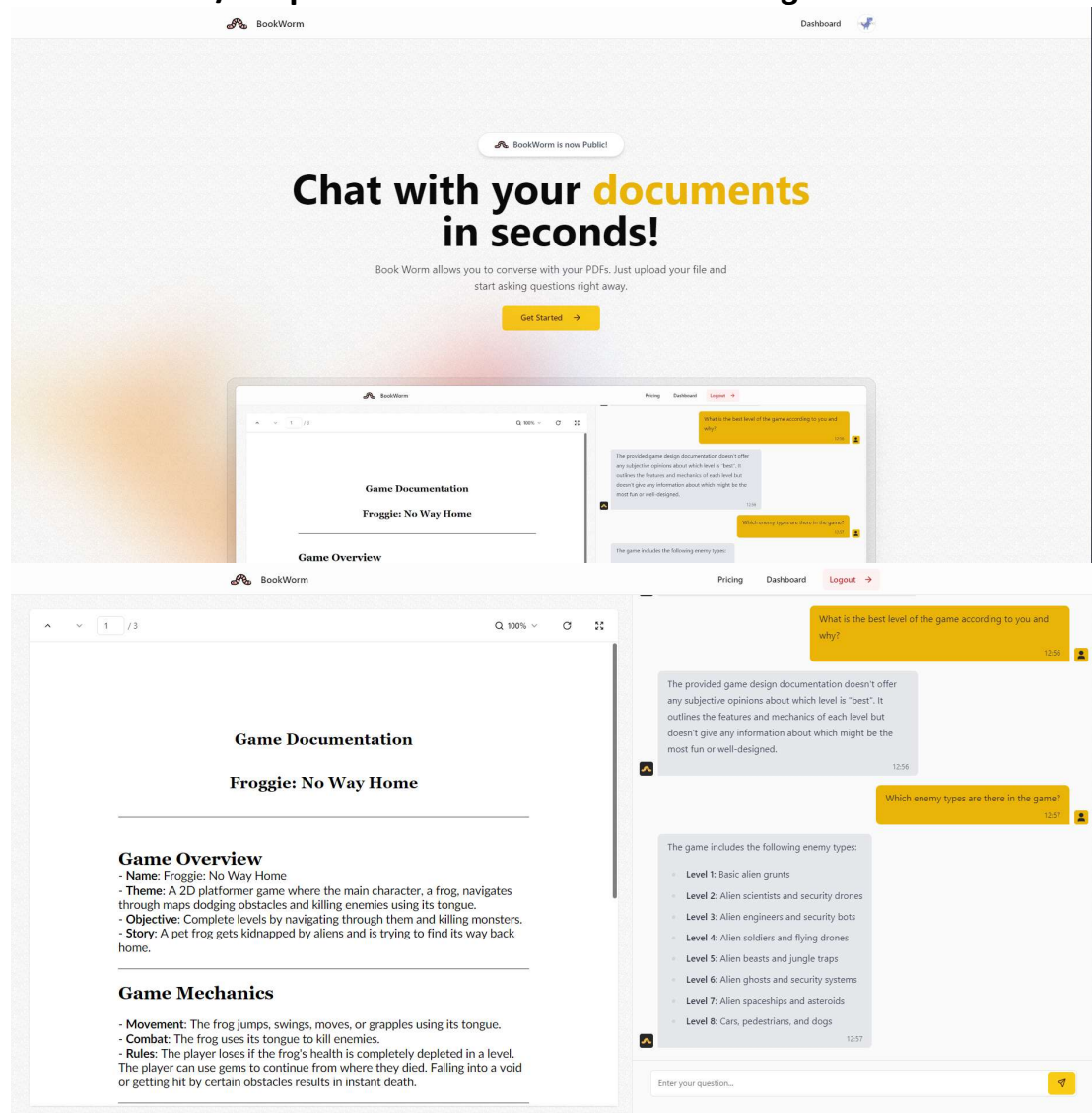
```

```

page.tsx X
src > app > auth-callback > page.tsx
1  "use client";
2
3  import { Suspense } from "react";
4  import { useRouter, useSearchParams } from "next/navigation";
5  import { trpc } from "@/_trpc/client";
6  import { Loader2 } from "lucide-react";
7
8  export const dynamic = "force-dynamic";
9
10 const AuthCallbackHandler = () => {
11   console.log("auth callback");
12
13   const router = useRouter();
14   const searchParams = useSearchParams();
15   const origin = searchParams.get("origin");
16
17   // Use the query hook correctly
18   const { data, error } = trpc.authCallback.useQuery(undefined, {
19     retry: true,
20     retryDelay: 500, // checking every 0.5s
21   });
22
23   // Handle success and error states
24   if (data && data.success) {
25     // User is synced to db
26     router.push(origin ? `/${origin}` : "/dashboard");
27   } else if (error) {
28     if (error.data?.code === "UNAUTHORIZED") {
29       router.push("/sign-in");
30     }
31   }
32
33   return (
34     <div className="w-full mt-24 flex justify-center">
35       <div className="flex flex-col items-center gap-2">
36         <Loader2 className="h-8 w-8 animate-spin text-zinc-800" />
37         <h3 className="font-semibold text-xl">Setting up your account...</h3>
38         <p>You will be redirected automatically.</p>

```

5. Results/Output:- Entire Screen Shot including Date & Time



6. Remarks:-

Signature of the Student

(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)

