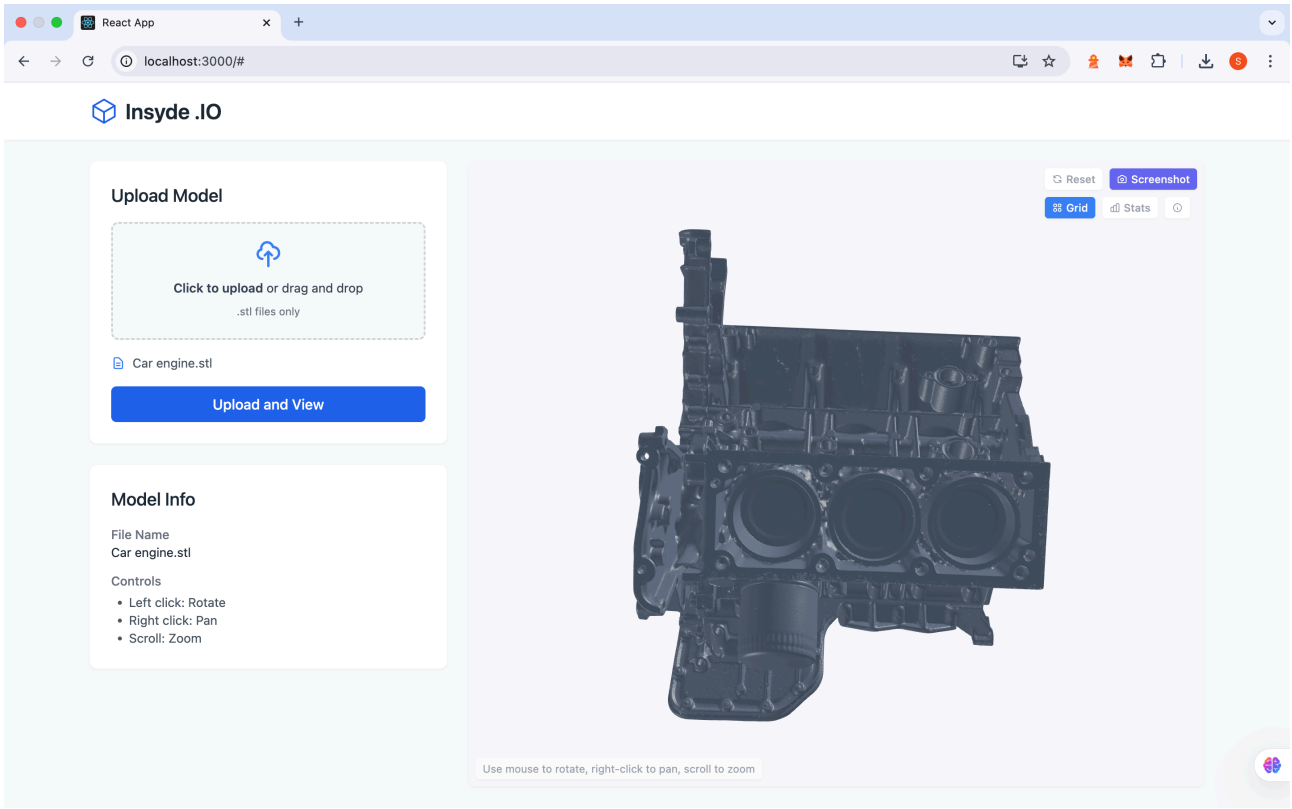
According to the full stack assignment used python and react in my project in below I given the demo of the project .Additionally added screenshot feature also for sty viewer. In below of demo given code of the project.



Backend code

Python app.py

```
from flask import Flask, request, jsonify, send_from_directory
from flask_cors import CORS
import os

app = Flask(__name__)
CORS(app)

UPLOAD_FOLDER = "uploads"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)


@app.route("/upload", methods=["POST"])
def upload_file():
    if "file" not in request.files:
        return jsonify({"error": "No file uploaded"}), 400

    file = request.files["file"]
    file_path = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(file_path)
    return jsonify({"message": "File uploaded", "filename": file.filename}), 200


@app.route("/files/<path:filename>")
def get_file(filename):
    file_path = os.path.join(UPLOAD_FOLDER, filename)

    if not os.path.exists(file_path):
        return jsonify({"error": "File not found"}), 404

    return send_from_directory(UPLOAD_FOLDER, filename, as_attachment=False)


if __name__ == "__main__":
    app.run(debug=True)
```

Frontend

ModelView.jsx

```
import { Canvas } from "@react-three/fiber";
import { OrbitControls, Grid, Stars, Stats } from "@react-three/drei";
import React, { Suspense, useEffect, useState } from "react";
import { STLLoader } from "three/examples/jsm/loaders/STLLoader";
import { useLoader, useThree } from "@react-three/fiber";
import * as THREE from "three";

const LoadingSpinner = () => (
  <div className="absolute inset-0 flex items-center justify-center">
    <div className="flex flex-col items-center">
      <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mb-2"></
div>
      <p className="text-sm text-gray-400">Loading model...</p>
    </div>
  </div>
);

// New screenshot button component
const ScreenshotButton = () => {
  const takeScreenshot = () => {
    const canvas = document.querySelector('canvas');
    if (!canvas) return;

    // Create a temporary link element
    const link = document.createElement('a');
    link.download = `cad-model-${Date.now()}.png`;

    // Convert canvas content to data URL
    canvas.toBlob((blob) => {
      link.href = URL.createObjectURL(blob);
      link.click();

      // Clean up
      URL.revokeObjectURL(link.href);
    }, 'image/png');
  };

  return (
    <button
      onClick={takeScreenshot}
      className="bg-indigo-500 hover:bg-indigo-600 text-white font-medium py-1 px-2 rounded
text-xs flex items-center shadow-sm transition"
    >
      <svg
        xmlns="http://www.w3.org/2000/svg"
        className="h-3 w-3 mr-1"
        fill="none"
        viewBox="0 0 24 24"
        stroke="currentColor"
      >
        <path
          strokeLinecap="round"
          strokeLinejoin="round"
          strokeWidth={2}
          d="M3 9a2 2 0 012-2h.93a2 2 0 001.664-.89l.812-1.22A2 2 0 0110.07 4h3.86a2 2 0
011.664.89l.812 1.22A2 2 0 0018.07 7H19a2 2 0 012 2v9a2 2 0 01-2 2H5a2 2 0 01-2-2V9z"
```

```
        />
        <path
          strokeLinecap="round"
          strokeLinejoin="round"
          strokeWidth={2}
          d="M15 13a3 3 0 11-6 0 3 3 0 016 0z"
        />
      </svg>
      Screenshot
    </button>
  );
};

const Model = ({ fileUrl }) => {
  const geometry = useLoader(STLLoader, fileUrl);
  const meshRef = React.useRef();
  const { camera, scene } = useThree();
  const [isScaled, setIsScaled] = useState(false);

  useEffect(() => {
    if (meshRef.current && geometry && !isScaled) {
      // Center the geometry
      geometry.center();

      // Create a bounding box
      const box = new THREE.Box3().setFromObject(meshRef.current);
      const size = box.getSize(new THREE.Vector3());
      const center = box.getCenter(new THREE.Vector3());

      // Find the max dimension
      const maxDim = Math.max(size.x, size.y, size.z);

      // Scale model to a reasonable size
      const scaleFactor = 5 / Math.max(0.1, maxDim);
      meshRef.current.scale.setScalar(scaleFactor);

      // Position camera to fit the object
      const cameraPosition = new THREE.Vector3(1, 0.5, 1);
      cameraPosition.normalize();
      cameraPosition.multiplyScalar(maxDim * 2.5);
      camera.position.copy(cameraPosition);
      camera.lookAt(center);

      // Reset orbit controls target to model center
      scene.userData.controls?.target.copy(center);

      setIsScaled(true);
    }
  }, [geometry, camera, scene, isScaled]);

  return (
    <mesh ref={meshRef} castShadow receiveShadow>
      <primitive object={geometry} attach="geometry" />
      <meshPhongMaterial
        color="#94a3b8" // Changed to a silver color (slate-400)
        specular="#aaaaaa"
        shininess={40}
        flatShading={false}
        side={THREE.DoubleSide}
      />
```

```
    </mesh>
  );
};

const CameraController = () => {
  const { camera } = useThree();

  useEffect(() => {
    camera.near = 0.01;
    camera.far = 10000;
    camera.updateProjectionMatrix();
  }, [camera]);

  return null;
};

const ViewerControls = ({
  controlsRef,
  showGrid,
  setShowGrid,
  showStats,
  setShowStats,
}) => {
  const [showHelp, setShowHelp] = useState(false);

  return (
    <>
      <div className="absolute top-2 right-2 flex flex-col space-y-2">
        <div className="flex space-x-2">
          <button
            onClick={() => controlsRef.current?.reset()}
            className="bg-white/90 hover:bg-white text-gray-400 font-medium py-1 px-2 rounded
text-xs flex items-center shadow-sm transition"
          >
            <svg
              xmlns="http://www.w3.org/2000/svg"
              className="h-3 w-3 mr-1"
              fill="none"
              viewBox="0 0 24 24"
              stroke="currentColor"
            >
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M4 4v5h.582m15.356 2A8.001 8.001 0 004.582 9m0 0H9m11 11v-5h-.581m0
0a8.003 8.003 0 01-15.357-2m15.357 2H15"
              />
            </svg>
            Reset
          </button>

          <ScreenshotButton />
        </div>

        <div className="flex space-x-2">
          <button
            onClick={() => setShowGrid(!showGrid)}
            className={`${
              showGrid ? "bg-blue-500 text-white" : "bg-white/90 text-gray-400"
```

```
          } hover:bg-blue-600 hover:text-white font-medium py-1 px-2 rounded text-xs flex items-
center shadow-sm transition`}
        >
          <svg
            xmlns="http://www.w3.org/2000/svg"
            className="h-3 w-3 mr-1"
            fill="none"
            viewBox="0 0 24 24"
            stroke="currentColor"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M4 6a2 2 0 012-2h2a2 2 0 012 2v2a2 2 0 01-2 2H6a2 2 0 01-2-2V6zM14 6a2 2 0
012-2h2a2 2 0 012 2v2a2 2 0 01-2 2h-2a2 2 0 01-2-2V6zM4 16a2 2 0 012-2h2a2 2 0 012 2v2a2 2
0 01-2 2H6a2 2 0 01-2-2v-2zM14 16a2 2 0 012-2h2a2 2 0 012 2v2a2 2 0 01-2 2h-2a2 2 0
01-2-2v-2z"
            />
          </svg>
          Grid
        </button>

        <button
          onClick={() => setShowStats(!showStats)}
          className={`${
            showStats ? "bg-blue-500 text-white" : "bg-white/90 text-gray-400"
          } hover:bg-blue-600 hover:text-white font-medium py-1 px-2 rounded text-xs flex items-
center shadow-sm transition`}
        >
          <svg
            xmlns="http://www.w3.org/2000/svg"
            className="h-3 w-3 mr-1"
            fill="none"
            viewBox="0 0 24 24"
            stroke="currentColor"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0 002-2zm0 0V9a2 2 0
012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2 2 0 012-2h2a2 2 0 012
2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z"
            />
          </svg>
          Stats
        </button>

        <button
          onClick={() => setShowHelp(!showHelp)}
          className="bg-white/90 hover:bg-white text-gray-400 font-medium py-1 px-2 rounded
text-xs flex items-center shadow-sm transition"
        >
          <svg
            xmlns="http://www.w3.org/2000/svg"
            className="h-3 w-3"
            fill="none"
            viewBox="0 0 24 24"
            stroke="currentColor"
```

```
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M13 16h-1v-4h-1m1-4h.01M21 12a9 9 0 11-18 0 9 9 0 0118 0z"
            />
          </svg>
        </button>
      </div>
    </div>

    {showHelp && (
      <div className="absolute bottom-10 right-2 bg-white/90 p-3 rounded shadow-sm text-xs
text-gray-400 w-48">
        <div className="font-medium mb-2">Controls</div>
        <ul className="space-y-1">
          <li className="flex">
            <span className="font-medium w-24">Left click + drag:</span>{" "}
            Rotate
          </li>
          <li className="flex">
            <span className="font-medium w-24">Right click + drag:</span> Pan
          </li>
          <li className="flex">
            <span className="font-medium w-24">Scroll:</span> Zoom
          </li>
          <li className="flex">
            <span className="font-medium w-24">Reset button:</span> Reset view
          </li>
        </ul>
        <button
          className="mt-2 text-blue-600 hover:text-blue-800"
          onClick={() => setShowHelp(false)}
        >
          Close
        </button>
      </div>
    )}
  </>
 );
};

const ModelViewer = ({ fileUrl }) => {
 const controlsRef = React.useRef();
 const [showGrid, setShowGrid] = useState(true);
 const [showStats, setShowStats] = useState(false);

 const handleCreated = (state) => {
  if (controlsRef.current) {
   state.scene.userData.controls = controlsRef.current;
  }
 };

 return (
  <div className="relative w-full h-full">
   <Canvas
    camera={{
     position: [20, 20, 20],
     fov: 45,
```

```
    near: 0.01,
    far: 10000,
  }}
  className="w-full h-full"
  shadows
  onCreated={handleCreated}
  dpr={[1, 2]}
>
  {/* Scene lighting */}
  <color attach="background" args={["#f8f9fa"]} />
  <ambientLight intensity={0.5} />
  <directionalLight position={[10, 10, 5]} intensity={0.7} castShadow />
  <directionalLight position={[-10, -10, -5]} intensity={0.3} />
  <hemisphereLight
    color="#ffffff"
    groundColor="#b9b9b9"
    intensity={0.3}
  />
  <Stars radius={100} depth={50} count={1000} factor={4} fade />

  <CameraController />

  {/* Reference grid */}
  {showGrid && (
    <Grid
      renderOrder={-1}
      position={[0, -0.5, 0]}
      infiniteGrid
      cellSize={0.6}
      cellThickness={0.6}
      sectionSize={3}
      sectionThickness={1}
      sectionColor="#6b7280"
      fadeDistance={50}
    />
  )}

  <Suspense fallback={null}>
    <Model fileUrl={fileUrl} />
  </Suspense>

  <OrbitControls
    ref={controlsRef}
    enablePan={true}
    enableZoom={true}
    enableRotate={true}
    minDistance={0.1}
    maxDistance={1000}
    target={[0, 0, 0]}
    makeDefault
    zoomSpeed={1}
    enableDamping={true}
    dampingFactor={0.05}
  />

  {showStats && <Stats />}
</Canvas>

<Suspense fallback={<LoadingSpinner />} />
```

```
      <ViewerControls
        controlsRef={controlsRef}
        showGrid={showGrid}
        setShowGrid={setShowGrid}
        showStats={showStats}
        setShowStats={setShowStats}
      />

      <div className="absolute bottom-2 left-2 bg-white/80 px-2 py-1 rounded text-xs text-
gray-400 shadow-sm">
        Use mouse to rotate, right-click to pan, scroll to zoom
      </div>
    </div>
  );
};

export default ModelViewer;
```

File upload.jsx

```
import { useState } from "react";
import axios from "axios";

const FileUpload = ({ onUpload }) => {
  const [file, setFile] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const [isDragging, setIsDragging] = useState(false);

  const handleUpload = async () => {
    if (!file) {
      setError("Please select a file first");
      return;
    }

    try {
      setLoading(true);
      setError(null);
      const formData = new FormData();
      formData.append("file", file);

      const response = await axios.post(
        "http://127.0.0.1:5000/upload",
        formData,
        {
          headers: {
            "Content-Type": "multipart/form-data",
          },
        }
      );
      onUpload(response.data.filename, file.name);
    } catch (err) {
      setError("Error uploading file. Please try again.");
      console.error(err);
    } finally {
      setLoading(false);
    }
  };

  const handleDragOver = (e) => {
```

```
    e.preventDefault();
    setIsDragging(true);
  };

  const handleDragLeave = (e) => {
    e.preventDefault();
    setIsDragging(false);
  };

  const handleDrop = (e) => {
    e.preventDefault();
    setIsDragging(false);

    if (e.dataTransfer.files && e.dataTransfer.files[0]) {
      const droppedFile = e.dataTransfer.files[0];
      if (droppedFile.name.toLowerCase().endsWith(".stl")) {
        setFile(droppedFile);
        setError(null);
      } else {
        setError("Please upload only .stl files");
      }
    }
  };

  return (
    <div className="space-y-4">
      <div
        className={`border-2 ${
          isDragging ? "border-blue-500 bg-blue-50" : "border-gray-300"
        } border-dashed rounded-lg cursor-pointer ${file ? "bg-gray-50" : ""}`}
        onDragOver={handleDragOver}
        onDragLeave={handleDragLeave}
        onDrop={handleDrop}
      >
        <label className="flex flex-col items-center justify-center h-32 w-full cursor-pointer">
          <div className="flex flex-col items-center justify-center pt-5 pb-6">
            <svg
              className={`w-8 h-8 mb-3 ${
                file ? "text-blue-500" : "text-gray-500"
              }`}
              xmlns="http://www.w3.org/2000/svg"
              fill="none"
              viewBox="0 0 24 24"
              stroke="currentColor"
            >
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M7 16a4 4 0 01-.88-7.903A5 5 0 1115.9 6L16 6a5 5 0 011 9.9M15 13l-3-3m0 0l-3
3m3-3v12"
              />
            </svg>
            <p className="mb-2 text-sm text-gray-700">
              <span className="font-semibold">Click to upload</span> or drag and
              drop
            </p>
            <p className="text-xs text-gray-500">.stl files only</p>
          </div>
          <input
```

```
          type="file"
          className="hidden"
          accept=".stl"
          onChange={(e) => {
            if (e.target.files.length > 0) {
              setFile(e.target.files[0]);
              setError(null);
            }
          }}
        />
      </label>
    </div>

    {error && (
      <div className="text-red-500 text-sm bg-red-50 px-3 py-2 rounded-md flex items-center">
        <svg
          xmlns="http://www.w3.org/2000/svg"
          className="h-4 w-4 mr-1.5"
          fill="none"
          viewBox="0 0 24 24"
          stroke="currentColor"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            d="M12 8v4m0 4h.01M21 12a9 9 0 11-18 0 9 9 0 0118 0z"
          />
        </svg>
        {error}
      </div>
    )}

    {file && !error && (
      <div className="flex items-center space-x-2 text-sm text-gray-600">
        <svg
          xmlns="http://www.w3.org/2000/svg"
          className="h-4 w-4 text-blue-500"
          fill="none"
          viewBox="0 0 24 24"
          stroke="currentColor"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            d="M9 12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0 01.707.293l5.414
5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"
          />
        </svg>
        <span className="truncate max-w-[200px]">{file.name}</span>
      </div>
    )}

    <button
      onClick={handleUpload}
      disabled={loading || !file}
      className={`w-full flex justify-center items-center px-4 py-2 rounded-md text-white font-
medium transition ${
        loading || !file
```

```
          ? "bg-gray-400 cursor-not-allowed"
          : "bg-blue-600 hover:bg-blue-700"
      }`}
    >
      {loading ? (
        <>
          <svg
            className="animate-spin -ml-1 mr-2 h-4 w-4 text-white"
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
          >
            <circle
              className="opacity-25"
              cx="12"
              cy="12"
              r="10"
              stroke="currentColor"
              strokeWidth="4"
            ></circle>
            <path
              className="opacity-75"
              fill="currentColor"
              d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014
12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"
            ></path>
          </svg>
          Uploading...
        </>
      ) : (
        "Upload and View"
      )}
    </button>
  </div>
 );
};

export default FileUpload;
```