UCS2611 - Internet Programming Laboratory
Mini Project - ConsultansEase Full Stack Web Application

B Saathvik - CSE-C - 3122 22 5001 115
Shreedevi Seluka Balaji - CSE-C - 3122 22 5001 129
S Shwetha - CSE-C - 3122 22 5001 133

AIM:
To implement a consultancy full-stack web-application using the MERN stack and Google APIs.

REQUIREMENTS:

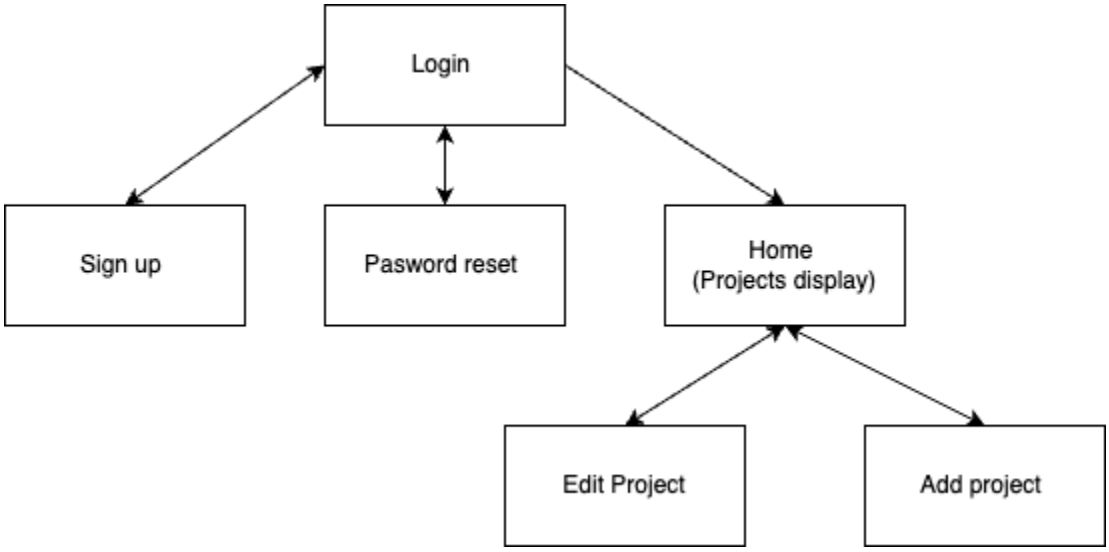
- The system must have a secure login for registered users using their college email ID.
- It should allow new users to register.
- After logging in, users must be able to enter and manage consultancy project details.
- The details to be entered include industry name, duration of the project, title of the project, principal investigator and co-principal investigator details, academic year, amount sanctioned, amount received, bill settlement details with proof uploads, signed agreement document upload, student details involved in the project, and a 100-word project summary.
- Users should be able to download project details based on filters like academic year, amount thresholds (e.g., above ₹50,000 or ₹1,00,000), faculty name, and industry name.
- The system should send notifications every 15 days to users about new consultancy projects and pending or completed bill settlements.
- All information should be stored and retrieved from Excel sheets, without using a database.

FILE STRUCTURE:

- ☐ consultansease
  - ☐ frontend
    - ☐ src
      - ☐ App.js
      - ☐ ConsultancyData.jsx
      - ☐ EditProjectForm.jsx
      - ☐ Forms.jsx
      - ☐ ConsultancyData.css
      - ☐ Forms.css
      - ☐ components
        - ☐ styles.css
        - ☐ ForgotPassword
          - ☐ ForgotPassword.jsx
        - ☐ Login
          - ☐ Login.jsx

- SignUp
  - SignUp.jsx
- backend
  - server.js
  - credentials.json
  - loginModel.js
  - uploads
  - routes
    - authRoutes.js
    - displayRoutes.js
    - formRoutes.js

FLOW DIAGRAM:



CODE:

FRONT-END:

App.js:

```
import React from "react";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import EditProjectForm from "./EditProjectForm";
import AddProjectForm from "./Forms";
import ConsultancyData from "./ConsultancyData";
import Login from "./components/Login/Login.jsx";
import Signup from "./components/SignUp/SignUp.jsx";
import ForgotPassword from "./components/ForgotPassword/ForgotPassword.jsx";
import PrivateRoute from "./PrivateRoute";
```

```
function App() {
 return (
   <Router>
     <Routes>
       <Route
         path="/edit/:id"
         element={
           <PrivateRoute>
             <EditProjectForm />
           </PrivateRoute>
         }
       />
       <Route
         path="/add-project"
         element={
           <PrivateRoute>
             <AddProjectForm />
           </PrivateRoute>
         }
       />
       <Route
         path="/home"
         element={
           <PrivateRoute>
             <ConsultancyData />
           </PrivateRoute>
         }
       />
       <Route path="/" element={<Login />} />
       <Route path="/signup" element={<Signup />} />
       <Route path="/forgotpassword" element={<ForgotPassword />} />
     </Routes>
   </Router>
 );
}

export default App;
```

ConsultancyData.jsx:

```
import React, { useEffect, useState, useCallback } from "react";
import { useNavigate } from "react-router-dom";
```

```jsx
import axios from "axios";
import "./ConsultancyData.css";

const ConsultancyData = () => {
 const navigate = useNavigate();
 const [data, setData] = useState([]);
 const [loading, setLoading] = useState(true);
 const [filters, setFilters] = useState({
   academicYear: "",
   pi: "",
   industry: "",
   minSanctioned: "",
 });

 // Wrap fetchData in useCallback to prevent unnecessary re-creation
 const fetchData = useCallback(async () => {
   setLoading(true);
   try {
     const response = await axios.get("http://localhost:5050/api", {
       params: filters,
     });
     setData(response.data || []);
   } catch (error) {
     console.error("Error fetching data:", error);
   } finally {
     setLoading(false);
   }
 }, [filters]); // Include filters as a dependency

 useEffect(() => {
   fetchData();
 }, [fetchData]);

 const handleInputChange = (e) => {
   const { name, value } = e.target;
   setFilters((prevFilters) => ({
     ...prevFilters,
     [name]: value,
   }));
 };

 const handleDelete = async (projectId) => {
```

```
  if (window.confirm("Are you sure you want to delete this entry?")) {
    try {
      await axios.delete(`http://localhost:5050/api/${projectId}`);
      // Refresh data after deletion
      fetchData();
    } catch (error) {
      console.error("Error deleting entry:", error);
      alert("Failed to delete entry. Please try again.");
    }
  }
};

const handleDownload = async () => {
  try {
    // Get filtered data
    const response = await axios.get("http://localhost:5050/api/download", {
      params: filters,
      responseType: "blob",
    });

    // Create download link
    const url = window.URL.createObjectURL(new Blob([response.data]));
    const link = document.createElement("a");
    link.href = url;
    link.setAttribute("download", "consultansease.xlsx");
    document.body.appendChild(link);
    link.click();

    // Clean up
    link.parentNode.removeChild(link);
  } catch (error) {
    console.error("Error downloading data:", error);
    alert("Failed to download data. Please try again.");
  }
};

const handleLogout = () => {
  localStorage.removeItem("isAuthenticated");
  navigate("/");
};

return (
```

```
<div className="container">
  <h4>ConsultansEase Data</h4>

  {/* Filter Inputs */}
  <div className="filter-bar">
    <input
      type="text"
      name="academicYear"
      placeholder="Academic Year"
      value={filters.academicYear}
      onChange={handleInputChange}
    />
    <input
      type="text"
      name="pi"
      placeholder="PI Name"
      value={filters.pi}
      onChange={handleInputChange}
    />
    <input
      type="text"
      name="industry"
      placeholder="Industry Name"
      value={filters.industry}
      onChange={handleInputChange}
    />
    <input
      type="text"
      name="minSanctioned"
      placeholder="Minimum Sanctioned Amount"
      value={filters.minSanctioned}
      onChange={handleInputChange}
    />
    <button
      onClick={() =>
        setFilters({
          academicYear: "",
          pi: "",
          industry: "",
          minSanctioned: "",
        })
      }
```

```
  >
    Reset Filters
  </button>

  <button
    onClick={() => navigate("../add-project")}
    className="add-project-btn"
  >
    {" "}
    Add New Project
  </button>

  {/* Download Button */}
  <button onClick={handleDownload} className="download-btn">
    Download Results
  </button>
</div>

{/* Table */}
{loading ? (
  <div className="loading-spinner">
    <div className="spinner-circle"></div>
    <p>Loading data...</p>
  </div>
) : (
  <div className="table-container">
    <table>
      <thead>
        <tr>
          <th>Project ID</th>
          <th>Industry</th>
          <th>Duration</th>
          <th>Title</th>
          <th>PI</th>
          <th>Co-PI</th>
          <th>Year</th>
          <th>Sanctioned</th>
          <th>Received</th>
          <th>Bill Proof</th>
          <th>Agreement</th>
          <th>Students</th>
          <th>Summary</th>
```

```
    <th>Actions</th>
  </tr>
</thead>
<tbody>
  {data.length === 0 ? (
    <tr>
      <td colSpan="14">No data available.</td>
    </tr>
  ) : (
    data.map((row, idx) => (
      <tr key={idx}>
        <td>{row.projectId}</td>
        <td>{row.industry}</td>
        <td>{row.duration}</td>
        <td>{row.title}</td>
        <td>{row.pi}</td>
        <td>{row.copi}</td>
        <td>{row.academicYear}</td>
        <td>{row.sanctionedAmount}</td>
        <td>{row.receivedAmount}</td>
        <td>{row.billProofLink}</td>
        <td>{row.agreementDocumentLink}</td>
        <td>{row.studentParticipants}</td>
        <td>{row.summary}</td>
        <td>
          <div className="action-buttons">
            <button
              className="edit-btn"
              onClick={() => navigate(`../edit/${row.projectId}`)}
            >
              Edit
            </button>
            <button
              className="delete-btn"
              onClick={() => handleDelete(row.projectId)}
            >
              Delete
            </button>
          </div>
        </td>
      </tr>
    ))
```

```
            )}
          </tbody>
        </table>
      </div>
    )}

    <a href="#!" onClick={handleLogout} className="logout-link">
      Logout &gt;
    </a>
  </div>
 );
};

export default ConsultancyData;
```

ConsultancyData.css:

```
* {
 margin: 0;
 padding: 0;
 box-sizing: border-box;
}

body {
 font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
 background-color: #fefae0;
 color: #333;
}

h4 {
 color: #8e24aa;
 font-size: 2rem;
 text-align: center;
 margin-bottom: 20px;
 margin-top: 0;
}

.container {
 max-width: 1000px;
 margin: auto;
 padding: 20px;
}
```

```css
.filter-bar {
 display: flex;
 flex-wrap: wrap;
 gap: 15px;
 margin-bottom: 20px;
}

.filter-bar select,
.filter-bar input {
 padding: 10px;
 border: 1px solid #ccc;
 border-radius: 8px;
 font-size: 1rem;
}

.project-card {
 background-color: #ffffff;
 padding: 20px;
 border-radius: 12px;
 box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
 margin-bottom: 15px;
}

button {
 padding: 12px;
 border-radius: 8px;
 border: none;
 background-color: #8e24aa;
 color: white;
 font-weight: 600;
 cursor: pointer;
 font-size: 1rem;
}

button:hover {
 background-color: #6a1b9a;
}

.loading-spinner {
 display: flex;
 flex-direction: column;
 align-items: center;
```

```
 justify-content: center;
 padding: 30px;
 color: #555;
}

.spinner-circle {
 border: 4px solid #f3f3f3;
 border-top: 4px solid #8e24aa;
 border-radius: 50%;
 width: 30px;
 height: 30px;
 animation: spin 1s linear infinite;
 margin-bottom: 10px;
}

@keyframes spin {
 to {
   transform: rotate(360deg);
 }
}

/* Table styles */
.table-container {
 overflow-x: auto;
 margin: 20px 0 20px;
}

table {
 width: 100%;
 border-collapse: collapse;
 background-color: white;
 box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
 border-radius: 8px;
 overflow: hidden;
}

thead {
 background-color: #8e24aa;
 color: white;
}

th,
```

```
td {
 padding: 12px 15px;
 text-align: left;
 border-bottom: 1px solid #ddd;
}

th {
 font-weight: bold;
}

tbody tr:hover {
 background-color: #f5f0ff;
}

tbody tr:last-child td {
 border-bottom: none;
}

.delete-btn:hover {
 background-color: #c62828;
}

.download-btn {
 background-color: #2e7d32;
 margin-left: auto;
}

.download-btn:hover {
 background-color: #1b5e20;
}

.edit-btn:hover {
 background-color: #45a049;
}
.action-buttons {
 display: flex;
 gap: 8px;
 justify-content: center;
}

.edit-btn,
.delete-btn {
```

```
 padding: 6px 12px;
 border: none;
 border-radius: 5px;
 font-weight: bold;
 cursor: pointer;
}

.edit-btn {
 background-color: #28a745; /* green */
 color: white;
}

.delete-btn {
 background-color: #dc3545; /* red */
 color: white;
}

.add-project-btn {
 background-color: #007bff;
}

.add-project-btn:hover {
 background-color: #0056b2;
}

a {
 text-decoration: none;
 color: #1976d2;
 text-align: center;
 margin-top: 10px;
 font-size: 0.95rem;
}

a:hover {
 text-decoration: underline;
 color: #0d47a1;
}
```

Forms.jsx:

```
import React, { useState, useEffect } from "react";
import "./Forms.css";
import { useNavigate } from "react-router-dom";
```

```
const AddProjectForm = () => {
 const navigate = useNavigate();
 const [formData, setFormData] = useState({
   industry: "",
   duration: "",
   title: "",
   pi: "",
   copi: "",
   year: "",
   sanctioned: "",
   received: "",
   billProof: null,
   agreementDoc: null,
   students: "",
   summary: "",
 });

 const [submitted, setSubmitted] = useState(false);

 useEffect(() => {
   if (submitted) {
     const timer = setTimeout(() => {
       setSubmitted(false);
       navigate("/home");
     }, 5000);
     return () => clearTimeout(timer); // Cleanup
   }
 }, [submitted, navigate]);

 const handleChange = (e) => {
   const { name, value, files } = e.target;
   setFormData({
     ...formData,
     [name]: files ? files[0] : value,
   });
 };

 const handleSubmit = async (e) => {
   e.preventDefault();

   const formDataToSend = new FormData();
```

```
  for (const key in formData) {
    formDataToSend.append(key, formData[key]);
  }

  try {
    const response = await fetch("http://localhost:5050/forms", {
      method: "POST",
      body: formDataToSend,
    });

    const contentType = response.headers.get("content-type");
    if (contentType && contentType.includes("application/json")) {
      const data = await response.json();
      console.log(data);
      setSubmitted(true);

      // Redirect to home page after submission
    } else {
      const text = await response.text();
      console.error("Non-JSON response:", text);
    }
  } catch (error) {
    console.error("Error submitting form:", error);
  }
};

return (
  <div className="form-container">
    <form onSubmit={handleSubmit}>
      <label>Industry:</label>
      <input
        type="text"
        name="industry"
        value={formData.industry}
        onChange={handleChange}
        required
      />

      <label>Duration:</label>
      <input
        type="number"
        name="duration"
```

```
  value={formData.duration}
  onChange={handleChange}
  required
/>

<label>Title of Project:</label>
<input
  type="text"
  name="title"
  value={formData.title}
  onChange={handleChange}
  required
/>

<label>Principal Investigator:</label>
<input
  type="text"
  name="pi"
  value={formData.pi}
  onChange={handleChange}
  required
/>

<label>Co-PI:</label>
<input
  type="text"
  name="copi"
  value={formData.copi}
  onChange={handleChange}
  required
/>

<label>Year:</label>
<input
  type="number"
  name="year"
  value={formData.year}
  onChange={handleChange}
  required
/>

<label>Sanctioned Amount:</label>
```

```
<input
  type="number"
  name="sanctioned"
  value={formData.sanctioned}
  onChange={handleChange}
  required
/>

<label>Amount Received:</label>
<input
  type="number"
  name="received"
  value={formData.received}
  onChange={handleChange}
  required
/>

<label>Bill Proof Document:</label>
<input
  type="file"
  name="billProof"
  accept=".pdf,.doc,.docx,.jpg,.jpeg,.png,.xlsx,.xls,.ppt,.pptx,.txt"
  onChange={handleChange}
  required
/>

<label>Agreement Document:</label>
<input
  type="file"
  name="agreementDoc"
  accept=".pdf,.doc,.docx,.jpg,.jpeg,.png,.xlsx,.xls,.ppt,.pptx,.txt"
  onChange={handleChange}
  required
/>

<label>Students Involved:</label>
<input
  type="text"
  name="students"
  value={formData.students}
  onChange={handleChange}
  required
```

```
        />

        <label>Summary of Work Done:</label>
        <textarea
          name="summary"
          value={formData.summary}
          onChange={handleChange}
          required
        ></textarea>

        <button type="submit">Submit</button>

        <a href="/home" id="backToHome">
          Back to Home {">"}
        </a>
      </form>

      {submitted && (
        <div className="success-message">
          Form submitted successfully! Redirecting...
        </div>
      )}
    </div>
  );
};

export default AddProjectForm;
```

Forms.css:

```
/* Base resets and font */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f7f7f7;
  height: 100vh;
}
```

```css
/* Center the form box */
.form-container {
 display: flex;
 justify-content: center;
 align-items: center;
 min-height: 100vh;
 padding: 20px;
 background-color: #fefae0;
}

/* White form box */
form {
 background-color: #fff;
 padding: 30px 40px;
 border-radius: 12px;
 box-shadow: 0 4px 25px rgba(0, 0, 0, 0.1);
 width: 100%;
 max-width: 500px;
 display: flex;
 flex-direction: column;
}

/* Label styling */
form label {
 margin-top: 10px;
 font-weight: 600;
 color: #333;
 font-size: 0.95rem;
}

/* Input and textarea */
form input[type="text"],
form input[type="email"],
form input[type="password"],
form input[type="number"],
form input[type="file"],
form textarea {
 padding: 10px 12px;
 margin-top: 6px;
 margin-bottom: 15px;
 border: 1px solid #ccc;
 border-radius: 8px;
```

```
 font-size: 0.95rem;
 width: 100%;
 transition: border-color 0.3s ease, box-shadow 0.3s ease;
}

form textarea {
 resize: vertical;
 min-height: 90px;
}

/* Focus styling */
form input:focus,
form textarea:focus {
 border-color: #8e24aa;
 box-shadow: 0 0 6px rgba(142, 36, 170, 0.25);
 outline: none;
}

/* Submit button */
form button {
 margin-top: 15px;
 padding: 10px;
 background-color: #8e24aa;
 color: white;
 font-weight: bold;
 border: none;
 border-radius: 8px;
 cursor: pointer;
 font-size: 1rem;
 transition: background-color 0.2s ease;
}

form button:hover {
 background-color: #6a1b9a;
}

/* Responsive tweaks */
@media (max-width: 600px) {
 form {
   padding: 25px 20px;
 }
```

```
 form input,
 form textarea {
   font-size: 0.9rem;
 }

 form button {
   font-size: 0.95rem;
 }
}

/* Prevent scroll on number input */
input[type="number"]::-webkit-outer-spin-button,
input[type="number"]::-webkit-inner-spin-button {
 -webkit-appearance: none;
 margin: 0;
}

/* Translucent green success box */
.success-message {
 position: fixed;
 bottom: 20px;
 left: 50%;
 transform: translateX(-50%);
 background-color: rgba(76, 175, 80, 0.85); /* green with some transparency */
 color: white;
 padding: 12px 24px;
 border-radius: 8px;
 box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
 font-weight: bold;
 animation: fadeIn 0.3s ease-in;
 z-index: 999;
}

#backToHome {
 margin-top: 20px;
}

/* Optional fade in animation */
@keyframes fadeIn {
 from {
   opacity: 0;
   transform: translateX(-50%) translateY(10px);
```

```
 }
 to {
   opacity: 1;
   transform: translateX(-50%) translateY(0);
 }
}
```

EditProjectForm.jsx:

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import "./Forms.css";
import { useNavigate } from "react-router-dom";

const EditProjectForm = () => {
 const navigate = useNavigate();
 const { id } = useParams();
 const [formData, setFormData] = useState({
   industry: "",
   duration: "",
   title: "",
   pi: "",
   copi: "",
   year: "",
   sanctioned: "",
   received: "",
   billProof: null,
   agreementDoc: null,
   students: "",
   summary: "",
 });

 const [submitted, setSubmitted] = useState(false);
 const [loading, setLoading] = useState(true);

 useEffect(() => {
   if (submitted) {
     const timer = setTimeout(() => {
       setSubmitted(false);
       navigate("/home");
     }, 5000);
     return () => clearTimeout(timer); // Cleanup
   }
```

```
}, [submitted, navigate]);

// Fetch existing data
useEffect(() => {
  const fetchData = async () => {
    try {
      const res = await fetch(`http://localhost:5050/edit/${id}`);
      const data = await res.json();

      setFormData({
        industry: data[1],
        duration: data[2] || "",
        title: data[3] || "",
        pi: data[4] || "",
        copi: data[5] || "",
        year: data[6] || "",
        sanctioned: data[7] || "",
        received: data[8] || "",
        billProof: null,
        agreementDoc: null,
        students: data[11] || "",
        summary: data[12] || "",
      });

      setLoading(false);
    } catch (err) {
      console.error("Error fetching data:", err);
      setLoading(false);
    }
  };

  fetchData();
}, [id]);

const handleChange = (e) => {
  const { name, value, files } = e.target;
  setFormData({
    ...formData,
    [name]: files ? files[0] : value,
  });
};
```

```
const handleSubmit = async (e) => {
  e.preventDefault();

  const updatedFormData = new FormData();
  for (const key in formData) {
    updatedFormData.append(key, formData[key]);
  }

  try {
    const res = await fetch(`http://localhost:5050/edit/${id}`, {
      method: "PUT", // or PATCH depending on backend
      body: updatedFormData,
    });

    const contentType = res.headers.get("content-type");
    if (contentType && contentType.includes("application/json")) {
      const data = await res.json();
      console.log(data);
      setSubmitted(true);
    } else {
      const text = await res.text();
      console.error("Non-JSON response:", text);
    }
  } catch (err) {
    console.error("Error updating form:", err);
  }
};

if (loading) return <div>Loading...</div>;

return (
  <div className="form-container">
    <form onSubmit={handleSubmit}>
      <label>Industry:</label>
      <input
        type="text"
        name="industry"
        value={formData.industry}
        onChange={handleChange}
        required
      />
```

```
<label>Duration:</label>
<input
  type="text"
  name="duration"
  value={formData.duration}
  onChange={handleChange}
  required
/>

<label>Title of Project:</label>
<input
  type="text"
  name="title"
  value={formData.title}
  onChange={handleChange}
  required
/>

<label>Principal Investigator:</label>
<input
  type="text"
  name="pi"
  value={formData.pi}
  onChange={handleChange}
  required
/>

<label>Co-PI:</label>
<input
  type="text"
  name="copi"
  value={formData.copi}
  onChange={handleChange}
  required
/>

<label>Year:</label>
<input
  type="number"
  name="year"
  value={formData.year}
  onChange={handleChange}
```

```
  required
/>

<label>Sanctioned Amount:</label>
<input
  type="number"
  name="sanctioned"
  value={formData.sanctioned}
  onChange={handleChange}
  required
/>

<label>Amount Received:</label>
<input
  type="number"
  name="received"
  value={formData.received}
  onChange={handleChange}
  required
/>

<label>Add New Bill Proof Document:</label>
<input
  type="file"
  name="billProof"
  accept=".pdf,.doc,.docx,.jpg,.jpeg,.png,.xlsx,.xls,.ppt,.pptx,.txt"
  onChange={handleChange}
/>

<label>Add New Agreement Document:</label>
<input
  type="file"
  name="agreementDoc"
  accept=".pdf,.doc,.docx,.jpg,.jpeg,.png,.xlsx,.xls,.ppt,.pptx,.txt"
  onChange={handleChange}
/>

<label>Students Involved:</label>
<input
  type="text"
  name="students"
  value={formData.students}
```

```
          onChange={handleChange}
          required
        />

        <label>Summary of Work Done:</label>
        <textarea
          name="summary"
          value={formData.summary}
          onChange={handleChange}
          required
        ></textarea>

        <button type="submit">Update</button>
      </form>

      {submitted && (
        <div className="success-message">
          Form updated successfully! Redirecting...
        </div>
      )}

      <a href="/home" id="backToHome">
        Back to Home {">"}
      </a>
    </div>
  );
};

export default EditProjectForm;
```

Login.jsx:

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import "../styles.css";

function Login() {
  const navigate = useNavigate();

  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [message, setMessage] = useState("");
  const [loginSuccess, setLoginSuccess] = useState(false);
```

```
const handleSubmit = async (e) => {
  e.preventDefault();

  try {
    const res = await fetch("http://localhost:5050/validate-user", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ email, password }),
    });

    const data = await res.json();

    if (res.ok) {
      localStorage.setItem("isAuthenticated", "true");
      setLoginSuccess(true);
      setMessage(data.message);
      setTimeout(() => {
        navigate("/home");
      }, 1000);
    } else {
      setLoginSuccess(false);
      setMessage(data.message);
    }
  } catch (error) {
    console.error("Login error:", error);
    setLoginSuccess(false);
    setMessage("Something went wrong. Please try again.");
  }
};

return (
  <div id="loginPage">
    <div id="loginBox">
      <h4>ConsultansEase</h4>

      <form id="loginForm" onSubmit={handleSubmit}>
        <input
          type="email"
          id="userEmail"
          name="userEmail"
          placeholder="Email"
```

```
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />

          <input
            type="password"
            id="userPassword"
            name="userPassword"
            placeholder="Password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />

          {message && (
            <p className={loginSuccess ? "success-msg" : "error-msg"}>
              {message}
            </p>
          )}

          <div id="submitDiv">
            <button type="submit">Login</button>
            <a href="/forgotpassword" id="forgetPassword">
              Forgot Password? {">"}
            </a>
            <a href="/signup" id="signUp">
              Don't have an account? Sign up {">"}
            </a>
          </div>
        </form>
      </div>
    </div>
  );
}

export default Login;
```

SignUp.jsx:

```
import React, { useState } from "react";
import "../styles.css";
```

```
function Signup() {
 const [formData, setFormData] = useState({
   name: "",
   email: "",
   password: "",
 });

 const [message, setMessage] = useState("");
 const [isSuccess, setIsSuccess] = useState(null);

 const handleChange = (e) => {
   const { name, value } = e.target;
   setFormData((prev) => ({ ...prev, [name]: value }));
 };

 const handleSubmit = async (e) => {
   e.preventDefault();
   setMessage("");

   try {
     const res = await fetch("http://localhost:5050/create-new-user", {
       method: "POST",
       headers: { "Content-Type": "application/json" },
       body: JSON.stringify(formData),
     });

     const data = await res.json();

     if (res.status === 201) {
       setMessage(data.message);
       setIsSuccess(true);
       setFormData({ name: "", email: "", password: "" });
     } else {
       setMessage(data.message);
       setIsSuccess(false);
     }
   } catch (error) {
     console.error("Signup error:", error);
     setMessage("Something went wrong. Please try again.");
     setIsSuccess(false);
   }
 };
```

```
return (
  <div id="signUpPage">
    <form id="signUpForm" onSubmit={handleSubmit}>
      <h4>Welcome to ConsultansEase!</h4>

      <input
        type="text"
        id="userName"
        name="name"
        placeholder="Full Name"
        value={formData.name}
        onChange={handleChange}
        required
      />

      <input
        type="email"
        id="userEmail"
        name="email"
        placeholder="Email"
        value={formData.email}
        onChange={handleChange}
        required
      />

      <input
        type="password"
        id="userPassword"
        name="password"
        placeholder="Password"
        value={formData.password}
        onChange={handleChange}
        required
      />

      {message && (
        <p className={isSuccess ? "success-msg" : "error-msg"}>{message}</p>
      )}

      <button type="submit">Submit</button>
```

```
        <a href="/" id="backToLogin">
          Back to Login {">"}
        </a>
      </form>
    </div>
  );
}

export default Signup;
```

ForgotPassword.jsx:

```
import React, { useState } from "react";
import "../styles.css";

function ForgotPassword() {
  const [email, setEmail] = useState("");
  const [message, setMessage] = useState("");
  const [otpError, setOtpError] = useState("");
  const [showOtpInput, setShowOtpInput] = useState(false);
  const [otp, setOtp] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [otpVerified, setOtpVerified] = useState(false);
  const [loading, setLoading] = useState(false);
  const [passwordResetDone, setPasswordResetDone] = useState(false);
  const [disableAll, setDisableAll] = useState(false);

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!email) {
      setMessage("Please enter a valid email.");
      return;
    }

    setLoading(true);
    try {
      const response = await fetch("http://localhost:5050/request-otp", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ email }),
      });

      const data = await response.json();
```

```
    if (response.status === 200) {
      setMessage("OTP sent to your email! Please check your inbox/spam.");
      setShowOtpInput(true);
    } else {
      setMessage(data.message || "Something went wrong. Please try again.");
      setShowOtpInput(false);
    }
  } catch (error) {
    console.error("Error sending OTP:", error);
    setMessage("Error sending OTP. Please try again later.");
    setShowOtpInput(false);
  } finally {
    setLoading(false);
  }
};

const handleOtpSubmit = async (e) => {
  e.preventDefault();
  setLoading(true);
  setOtpError("");

  try {
    const response = await fetch("http://localhost:5050/verify-otp", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ email, otp }),
    });

    const data = await response.json();
    if (response.status === 200) {
      setOtpVerified(true);
      setMessage("OTP verified! You may now reset your password.");
    } else {
      setOtpError(
        data.message || "OTP verification failed. Please try again."
      );
    }
  } catch (error) {
    console.error("Error verifying OTP:", error);
    setOtpError("Server error. Please try again.");
  } finally {
    setLoading(false);
```

```
  }
};

const handleResetPassword = async (e) => {
  e.preventDefault();

  if (!newPassword) {
    setMessage("Please enter a new password.");
    return;
  }

  setLoading(true);
  try {
    const response = await fetch("http://localhost:5050/reset-password", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ email, password: newPassword }),
    });

    const data = await response.json();
    if (response.status === 200) {
      setMessage("Password updated successfully!");
      setPasswordResetDone(true);
      setDisableAll(true);
    } else {
      setMessage(data.message || "Failed to update password.");
    }
  } catch (error) {
    console.error("Error resetting password:", error);
    setMessage("Internal server error.");
  } finally {
    setLoading(false);
  }
};

return (
  <div id="forgotPasswordPage">
    {loading ? (
      <div className="loading-spinner">
        <div className="spinner-circle"></div>
        <p>Loading, please wait...</p>
      </div>
```

```
) : otpVerified ? (
  <form id="resetPasswordForm" onSubmit={handleResetPassword}>
    <input
      type="password"
      id="newPassword"
      placeholder="Enter New Password"
      value={newPassword}
      onChange={(e) => setNewPassword(e.target.value)}
      required
      disabled={disableAll}
    />
    {message && (
      <p
        className={
          passwordResetDone || otpVerified ? "success-msg" : "error-msg"
        }
      >
        {message}
      </p>
    )}
    {!passwordResetDone && (
      <button type="submit" disabled={disableAll}>
        Submit
      </button>
    )}
    <a
      href="/"
      id="backToLogin"
      style={passwordResetDone ? { marginTop: 0 } : {}}
    >
      Back to Login &gt;
    </a>
  </form>
) : (
  <form
    id="emailForm"
    onSubmit={showOtpInput ? handleOtpSubmit : handleSubmit}
  >
    <h4>Forgot your password? No worries.</h4>

    <input
      type="email"
```

```
  id="userEmail"
  name="userEmail"
  placeholder="Email"
  required
  value={email}
  disabled={disableAll || showOtpInput}
  onChange={(e) => setEmail(e.target.value)}
/>

{message && !otpVerified && (
  <p className={showOtpInput ? "success-msg" : "error-msg"}>
    {message}
  </p>
)}

{showOtpInput && (
  <>
    <input
      type="text"
      id="otpInput"
      placeholder="Enter OTP"
      value={otp}
      onChange={(e) => setOtp(e.target.value)}
      required
      disabled={disableAll}
    />
    {otpError && <p className="error-msg">{otpError}</p>}
    <button type="submit" disabled={disableAll}>
      Verify OTP
    </button>
  </>
)}

{!showOtpInput && (
  <button type="submit" disabled={disableAll}>
    Submit
  </button>
)}

<a href="/" id="backToLogin">
  Back to Login &gt;
</a>
```

```
        </form>
      )}
    </div>
  );
}

export default ForgotPassword;
```

styles.css:

```
* {
 margin: 0;
 padding: 0;
 box-sizing: border-box;
}

body {
 font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
 background-color: #fefae0;
 color: #333;
}

#loginPage,
#signUpPage,
#forgotPasswordPage {
 display: flex;
 justify-content: center;
 align-items: center;
 min-height: 100vh;
 padding: 20px;
 background-color: #fff9db;
}

#loginBox,
#signUpForm,
#emailForm,
#resetPasswordForm {
 width: 100%;
 max-width: 400px;
 background-color: #ffffff;
 padding: 30px;
 border-radius: 12px;
 box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
```

```
}

h4 {
 color: #8e24aa;
 font-size: 2rem;
 text-align: center;
 margin-bottom: 30px;
}

form {
 display: flex;
 flex-direction: column;
}

input[type="text"],
input[type="email"],
input[type="password"] {
 padding: 12px 15px;
 margin-bottom: 20px;
 width: 100%;
 border: 1px solid #ccc;
 border-radius: 8px;
 font-size: 1rem;
 transition: all 0.3s ease;
}

input:focus {
 border-color: #8e24aa;
 box-shadow: 0 0 5px rgba(142, 36, 170, 0.3);
 outline: none;
}

button {
 padding: 12px;
 border-radius: 8px;
 border: none;
 width: 100%;
 background-color: #8e24aa;
 color: white;
 font-weight: 600;
 cursor: pointer;
 transition: background-color 0.2s ease;
```

```
 font-size: 1rem;
}

button:hover {
 background-color: #6a1b9a;
}

a {
 text-decoration: none;
 color: #1976d2;
 text-align: center;
 margin-top: 10px;
 font-size: 0.95rem;
}

a:hover {
 text-decoration: underline;
 color: #0d47a1;
}

#submitDiv {
 display: flex;
 flex-direction: column;
 align-items: center;
 gap: 12px;
 margin-top: 10px;
}

#backToLogin {
 margin-top: 20px;
}

.success-msg {
 color: #4caf50;
 background-color: #e8f5e9;
 font-size: 0.9rem;
 margin-bottom: 20px;
 text-align: center;
 padding: 8px;
 border-radius: 5px;
}
```

```css
.error-msg {
 color: #f44336;
 background-color: #ffebee;
 font-size: 0.9rem;
 margin-bottom: 20px;
 text-align: center;
 padding: 8px;
 border-radius: 5px;
}

.loading-spinner {
 display: flex;
 flex-direction: column;
 align-items: center;
 justify-content: center;
 background-color: white;
 padding: 30px;
 border-radius: 10px;
 box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
 color: #333;
 margin: 20px auto;
 max-width: 400px;
 text-align: center;
}

.loading-spinner p {
 margin-top: 15px;
 font-size: 1rem;
 color: #555;
}

.spinner-circle {
 border: 4px solid #f3f3f3;
 border-top: 4px solid #8e24aa;
 border-radius: 50%;
 width: 30px;
 height: 30px;
 animation: spin 1s linear infinite;
 margin-bottom: 10px;
}

@keyframes spin {
```

```
 0% {
   transform: rotate(0deg);
 }
 100% {
   transform: rotate(360deg);
 }
}

@media (max-width: 480px) {
 h4 {
   font-size: 1.5rem;
 }

 #loginBox,
 #signUpForm,
 #emailForm {
   padding: 20px;
 }

 button {
   font-size: 0.95rem;
   padding: 10px;
 }

 input {
   font-size: 0.95rem;
 }
}
```

BACK-END:

server.js:

```
import express from "express";
import cors from "cors";
import mongoose from "mongoose";

import formRoutes from "./routes/formRoutes.js";
import displayRoutes from "./routes/displayRoutes.js";
import authRoutes from "./routes/authRoutes.js";

const app = express();

app.use(cors());
```

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

const PORT = 5050;
const MONGO_URL = "mongodb://localhost:27017/consultansease";

const startServer = async () => {
 try {
   await mongoose.connect(MONGO_URL);
   console.log("Database connected successfully!");

   app.listen(PORT, () => {
     console.log(`Server running on PORT ${PORT}`);
   });
 } catch (error) {
   console.error("Failed to connect to MongoDB:", error.message);
 }
};

app.use("/", formRoutes);
app.use("/", displayRoutes);
app.use("/", authRoutes);

startServer();
```

loginModel.js:

```
import mongoose from "mongoose";

const loginSchema = new mongoose.Schema(
 {
   name: {
     type: String,
     required: true,
     trim: true,
   },
   email: {
     type: String,
     required: true,
     unique: true,
     lowercase: true,
     trim: true,
   },
```

```
    password: {
      type: String,
      required: true,
    },
  },
  {
    collection: "login",
  }
);

const LoginModel = mongoose.model("login", loginSchema);
export default LoginModel;
```

authRoutes.js:

```
import express from "express";
import argon2 from "argon2";
import nodemailer from "nodemailer";
import LoginModel from "../loginModel.js";

const router = express.Router();

router.post("/get-user", async (req, res) => {
  try {
    const user = await LoginModel.findOne({ email: req.body.email });

    if (!user) {
      return res.status(404).json({ message: "No user found with this email" });
    }

    const validPassword = await argon2.verify(user.password, req.body.password);

    if (!validPassword) {
      return res.status(401).json({ message: "Invalid password" });
    }

    res.status(200).json({ message: "Login successful", user });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Internal server error" });
  }
});
```

```
router.post("/create-new-user", async (req, res) => {
 try {
   const payload = req.body;

   const userExists = await LoginModel.findOne({ email: payload.email });
   if (userExists) {
     return res.status(400).json({
       message: "Email already exists in our database! Try 'Forgot Password'?",
     });
   }

   const hashedPassword = await argon2.hash(payload.password);

   const newUser = new LoginModel({
     name: payload.name,
     email: payload.email,
     password: hashedPassword,
   });

   await newUser.save();

   res
     .status(201)
     .json({ message: "User created successfully", user: newUser });
 } catch (error) {
   console.error(error);
   res.status(500).json({ message: "Internal server error" });
 }
});

router.post("/validate-user", async (req, res) => {
 try {
   const { email, password } = req.body;

   const user = await LoginModel.findOne({ email });

   if (!user) {
     return res.status(404).json({ message: "User not found" });
   }

   const isPasswordValid = await argon2.verify(user.password, password);
```

```
    if (!isPasswordValid) {
      return res.status(401).json({ message: "Invalid password" });
    }

    res.status(200).json({ message: "Login successful", user: user });
  } catch (error) {
    console.error("Error validating user:", error);
    res.status(500).json({ message: "Internal server error" });
  }
});

const otpStore = new Map();

router.post("/request-otp", async (req, res) => {
  const email = req.body.email;

  const user = await LoginModel.findOne({ email });
  if (!user) {
    return res.status(404).json({ message: "User not found! Try signing up?" });
  }

  const otp = Math.floor(100000 + Math.random() * 900000).toString();
  const expiresAt = Date.now() + 5 * 60 * 1000;

  otpStore.set(email, { otp, expiresAt });

  const transporter = nodemailer.createTransport({
    service: "gmail",
    auth: {
      user: "consultansease@gmail.com",
      pass: "tlbl djnc qmtt kqit",
    },
  });

  const mailOptions = {
    from: "ConsultansEase <consultansease@gmail.com>",
    to: email,
    subject: "OTP for Password Reset - ConsultansEase",
    html: `
    <p>Dear User,</p>
    <p>Your OTP for Password Reset is <strong>${otp}</strong>. It is valid for 5
minutes.</p>
```

```
    <p>Thanks and Regards,<br>Team ConsultansEase</p>
  `,
  };

  try {
    await transporter.sendMail(mailOptions);
    res.status(200).json({ message: "OTP sent to your email!" });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Failed to send OTP" });
  }
});

router.post("/verify-otp", (req, res) => {
  const email = req.body.email;
  const otp = req.body.otp;
  const stored = otpStore.get(email);

  if (!stored) {
    return res.status(400).json({ message: "No OTP requested!" });
  }

  if (Date.now() > stored.expiresAt) {
    otpStore.delete(email);
    return res.status(400).json({ message: "OTP expired!" });
  }

  if (stored.otp !== otp) {
    return res.status(400).json({ message: "Incorrect OTP!" });
  }

  res.status(200).json({ message: "OTP verified!" });
});

router.post("/reset-password", async (req, res) => {
  try {
    const { email, password } = req.body;
    if (!email || !password) {
      return res
        .status(400)
        .json({ message: "Email and password are required." });
    }
```

```
    const user = await LoginModel.findOne({ email });
    if (!user) {
      return res.status(404).json({ message: "User not found!" });
    }

    const newPassword = await argon2.hash(password);
    user.password = newPassword;
    await user.save();

    otpStore.delete(email);

    res.status(200).json({ message: "Password updated successfully!" });
  } catch (error) {
    console.error("Error resetting password:", error);
    res.status(500).json({ message: "Internal server error." });
  }
});

export default router;
```

displayRoutes.js:

```
import express from "express";
import { google } from "googleapis";
import ExcelJS from "exceljs";
import fs from "fs";
import path from "path";

// Create Express app
const router = express.Router();

// Main API endpoint
router.get("/api", async (req, res) => {
  try {
    const auth = new google.auth.GoogleAuth({
      keyFile: "credentials.json",
      scopes: "https://www.googleapis.com/auth/spreadsheets",
    });

    const client = await auth.getClient();
    const sheets = google.sheets({ version: "v4", auth: client });
```

```
const spreadsheetId = "1_T8zggpboigzMz9pG29FHoo8kN_gA2a_S12bz4ijdZs";

const getRows = await sheets.spreadsheets.values.get({
  spreadsheetId,
  range: "Sheet1",
});

const [headers, ...rows] = getRows.data.values;

const mappedHeaders = {
  projectID: "projectId",
  industry: "industry",
  duration: "duration",
  title: "title",
  pi: "pi",
  copi: "copi",
  year: "academicYear",
  sanctioned: "sanctionedAmount",
  received: "receivedAmount",
  billProof: "billProofLink",
  agreementDoc: "agreementDocumentLink",
  students: "studentParticipants",
  summary: "summary",
};

// Convert each row to an object using mapped headers
const formattedRows = rows.map((row) => {
  const entry = {};
  headers.forEach((header, i) => {
    const key = mappedHeaders[header.trim()] || header;
    entry[key] = row[i] || "";
  });
  return entry;
});

// Apply query param filtering
const filters = req.query;
const filteredRows = formattedRows.filter((row) => {
  if (
    filters.minSanctioned &&
    Number(row.sanctionedAmount) < Number(filters.minSanctioned)
  ) {
```

```
        return false;
      }

      // Handle other filters
      return Object.entries(filters).every(([key, value]) => {
        if (key === "minSanctioned") return true; // Already handled above
        if (!value) return true; // Skip empty filters
        return row[key]?.toString().toLowerCase().includes(value.toLowerCase());
      });
    });

    res.json(filteredRows);
  } catch (error) {
    console.error("Error fetching data:", error);
    res.status(500).json({ error: "Something went wrong" });
  }
});

// Delete entry by project ID
router.delete("/api/:projectId", async (req, res) => {
  try {
    const { projectId } = req.params;

    const auth = new google.auth.GoogleAuth({
      keyFile: "credentials.json",
      scopes: "https://www.googleapis.com/auth/spreadsheets",
    });

    const client = await auth.getClient();
    const sheets = google.sheets({ version: "v4", auth: client });

    const spreadsheetId = "1_T8zggpboigzMz9pG29FHoo8kN_gA2a_S12bz4ijdZs";

    // Step 1: Get all rows from the sheet
    const getRows = await sheets.spreadsheets.values.get({
      spreadsheetId,
      range: "Sheet1",
    });

    const values = getRows.data.values;
    const headers = values[0];
```

```
const projectIdIndex = headers.findIndex(
  (header) => header === "projectID"
);

if (projectIdIndex === -1) {
  return res.status(400).json({ error: "Project ID column not found" });
}

// Step 2: Find the row with matching projectId
const rowIndexToDelete = values.findIndex((row, index) => {
  if (index === 0) return false;
  return row[projectIdIndex] === projectId;
});

if (rowIndexToDelete === -1) {
  return res.status(404).json({ error: "Project not found" });
}

const row = values[rowIndexToDelete];
const billProofPath = row[9]; // 10th column
const agreementDocPath = row[10]; // 11th column

// Step 3: Delete the associated files
const deleteFileIfExists = (filePath) => {
  if (filePath && fs.existsSync(filePath)) {
    try {
      fs.unlinkSync(filePath);
    } catch (err) {
      console.error(`Failed to delete file: ${filePath}`, err);
    }
  }
};

deleteFileIfExists(billProofPath);
deleteFileIfExists(agreementDocPath);

// Step 4: Delete the row in Google Sheets
await sheets.spreadsheets.batchUpdate({
  spreadsheetId,
  resource: {
    requests: [
      {
```

```
            deleteDimension: {
              range: {
                sheetId: 0, // Assuming Sheet1 has ID 0
                dimension: "ROWS",
                startIndex: rowIndexToDelete,
                endIndex: rowIndexToDelete + 1,
              },
            },
          },
        ],
      },
    });

    res.json({
      success: true,
      message: "Entry and files deleted successfully",
    });
  } catch (error) {
    console.error("Error deleting entry:", error);
    res.status(500).json({ error: "Something went wrong" });
  }
});

// Download endpoint for filtered data
router.get("/api/download", async (req, res) => {
  try {
    const auth = new google.auth.GoogleAuth({
      keyFile: "credentials.json",
      scopes: "https://www.googleapis.com/auth/spreadsheets",
    });

    const client = await auth.getClient();
    const sheets = google.sheets({ version: "v4", auth: client });

    const spreadsheetId = "1_T8zggpboigzMz9pG29FHoo8kN_gA2a_S12bz4ijdZs";

    const getRows = await sheets.spreadsheets.values.get({
      spreadsheetId,
      range: "Sheet1",
    });

    const [headers, ...rows] = getRows.data.values;
```

```
const mappedHeaders = {
  "project id": "projectId",
  industry: "industry",
  duration: "duration",
  title: "title",
  pi: "pi",
  copi: "copi",
  year: "academicYear",
  sanctioned: "sanctionedAmount",
  received: "receivedAmount",
  billProof: "billProofLink",
  agreementDoc: "agreementDocumentLink",
  students: "studentParticipants",
  summary: "summary",
};

// Convert each row to an object using mapped headers
const formattedRows = rows.map((row) => {
  const entry = {};
  headers.forEach((header, i) => {
    const key = mappedHeaders[header.trim()] || header;
    entry[key] = row[i] || "";
  });
  return entry;
});

// Apply query param filtering
const filters = req.query;
const filteredRows = formattedRows.filter((row) => {
  if (
    filters.minSanctioned &&
    Number(row.sanctionedAmount) < Number(filters.minSanctioned)
  ) {
    return false;
  }

  return Object.entries(filters).every(([key, value]) => {
    if (key === "minSanctioned") return true;
    if (!value) return true;
    return row[key]?.toString().toLowerCase().includes(value.toLowerCase());
  });
```

```
});

// Create an Excel workbook using ExcelJS
const workbook = new ExcelJS.Workbook();
const worksheet = workbook.addWorksheet("Filtered Data");

// Add headers
worksheet.addRow(headers);

// Add data rows
filteredRows.forEach((row) => {
  const rowData = headers.map((header) => {
    const key = mappedHeaders[header.trim()] || header;
    return row[key] || "";
  });
  worksheet.addRow(rowData);
});

// Set column widths to auto
worksheet.columns.forEach((column) => {
  column.width = 15;
});

// Style the header row
worksheet.getRow(1).font = { bold: true };
worksheet.getRow(1).fill = {
  type: "pattern",
  pattern: "solid",
  fgColor: { argb: "FFE0E0E0" },
};

// Write to buffer
const buffer = await workbook.xlsx.writeBuffer();

// Send the response
res.setHeader(
  "Content-Type",
  "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
);
res.setHeader(
  "Content-Disposition",
  "attachment; filename=consultancy_data.xlsx"
```

```
    );
    res.send(buffer);
  } catch (error) {
    console.error("Error downloading data:", error);
    res.status(500).json({ error: "Something went wrong" });
  }
});

export default router;
```

formRoutes.js:

```
import express from "express";
import { google } from "googleapis";
import multer from "multer";
import path from "path";
import fs from "fs";
import { v4 as uuidv4 } from "uuid";

const router = express.Router();

const __filename = new URL(import.meta.url).pathname;
const __dirname = path.dirname(__filename);

// Ensure uploads directory exists
const uploadDir = path.join(__dirname, "..", "uploads");
if (!fs.existsSync(uploadDir)) {
  fs.mkdirSync(uploadDir);
}

// Multer setup
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, uploadDir);
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + path.extname(file.originalname));
  },
});
const upload = multer({ storage: storage });

// POST /forms
router.post(
```

```
"/forms",
upload.fields([{ name: "billProof" }, { name: "agreementDoc" }]),
async (req, res) => {
  try {
    const auth = new google.auth.GoogleAuth({
      keyFile: "credentials.json",
      scopes: "https://www.googleapis.com/auth/spreadsheets",
    });

    const client = await auth.getClient();
    const googleSheets = google.sheets({ version: "v4", auth: client });
    const spreadsheetId = "1_T8zggpboigzMz9pG29FHoo8kN_gA2a_S12bz4ijdZs";

    const formData = req.body;
    formData.billProof = req.files.billProof
      ? req.files.billProof[0].path
      : "";
    formData.agreementDoc = req.files.agreementDoc
      ? req.files.agreementDoc[0].path
      : "";

    const projectId = uuidv4();

    const rowData = [
      [
        projectId,
        formData.industry,
        formData.duration,
        formData.title,
        formData.pi,
        formData.copi,
        formData.year,
        formData.sanctioned,
        formData.received,
        formData.billProof,
        formData.agreementDoc,
        formData.students,
        formData.summary,
      ],
    ];

    await googleSheets.spreadsheets.values.append({
```

```
        spreadsheetId,
        range: "Sheet1!A:M",
        valueInputOption: "USER_ENTERED",
        resource: { values: rowData },
      });

      res.status(200).json({ message: "Data written to spreadsheet" });
    } catch (error) {
      console.error("Error writing to spreadsheet:", error.stack || error);
      res.status(500).json({ error: "Error writing to spreadsheet" });
    }
  }
);

// GET /edit/:id
router.get("/edit/:id", async (req, res) => {
 const { id } = req.params;
 try {
   const auth = new google.auth.GoogleAuth({
     keyFile: "credentials.json",
     scopes: "https://www.googleapis.com/auth/spreadsheets",
   });

   const client = await auth.getClient();
   const googleSheets = google.sheets({ version: "v4", auth: client });
   const spreadsheetId = "1_T8zggpboigzMz9pG29FHoo8kN_gA2a_S12bz4ijdZs";

   const response = await googleSheets.spreadsheets.values.get({
     spreadsheetId,
     range: "Sheet1!A:M",
   });

   const rows = response.data.values;
   const rowIndex = rows.findIndex((row) => row[0] === id);

   if (rowIndex === -1) {
     return res.status(404).json({ error: "Page not found" });
   }

   res.status(200).json(rows[rowIndex]);
 } catch (error) {
   console.error("Error fetching data:", error);
```

```
    res.status(500).json({ error: "Error fetching data" });
  }
});
// PUT /edit/:id
router.put(
  "/edit/:id",
  upload.fields([{ name: "billProof" }, { name: "agreementDoc" }]),
  async (req, res) => {
    const { id } = req.params;
    try {
      const auth = new google.auth.GoogleAuth({
        keyFile: "credentials.json",
        scopes: "https://www.googleapis.com/auth/spreadsheets",
      });

      const client = await auth.getClient();
      const googleSheets = google.sheets({ version: "v4", auth: client });
      const spreadsheetId = "1_T8zggpboigzMz9pG29FHoo8kN_gA2a_S12bz4ijdZs";

      // Fetch existing project data to find the old files
      const response = await googleSheets.spreadsheets.values.get({
        spreadsheetId,
        range: `Sheet1!A:M`,
      });

      const rows = response.data.values;
      const rowIndex = rows.findIndex((row) => row[0] === id);

      if (rowIndex === -1) {
        return res.status(404).json({ error: "Page not found" });
      }

      // Get the old files' paths
      const oldBillProof = rows[rowIndex][9]; // Assuming billProof is in the 10th
column (index 9)
      const oldAgreementDoc = rows[rowIndex][10]; // Assuming agreementDoc is in the
11th column (index 10)

      // Process the new form data
      const formData = req.body;
      let newBillProof = formData.billProof;
      let newAgreementDoc = formData.agreementDoc;
```

```
    // Only overwrite file paths if new files are uploaded
    if (req.files.billProof) {
      newBillProof = req.files.billProof[0].path;
      // Delete the old file if a new one is uploaded
      if (oldBillProof && fs.existsSync(oldBillProof)) {
        fs.unlinkSync(oldBillProof); // Delete the old billProof file
      }
    } else {
      newBillProof = oldBillProof; // Keep the old file if no new one is uploaded
    }

    if (req.files.agreementDoc) {
      newAgreementDoc = req.files.agreementDoc[0].path;
      // Delete the old file if a new one is uploaded
      if (oldAgreementDoc && fs.existsSync(oldAgreementDoc)) {
        fs.unlinkSync(oldAgreementDoc); // Delete the old agreementDoc file
      }
    } else {
      newAgreementDoc = oldAgreementDoc; // Keep the old file if no new one is
uploaded
    }

    // Update the row with the new data
    rows[rowIndex] = [
      id,
      formData.industry,
      formData.duration,
      formData.title,
      formData.pi,
      formData.copi,
      formData.year,
      formData.sanctioned,
      formData.received,
      newBillProof,
      newAgreementDoc,
      formData.students,
      formData.summary,
    ];

    // Update the spreadsheet with the modified row
    await googleSheets.spreadsheets.values.update({
```

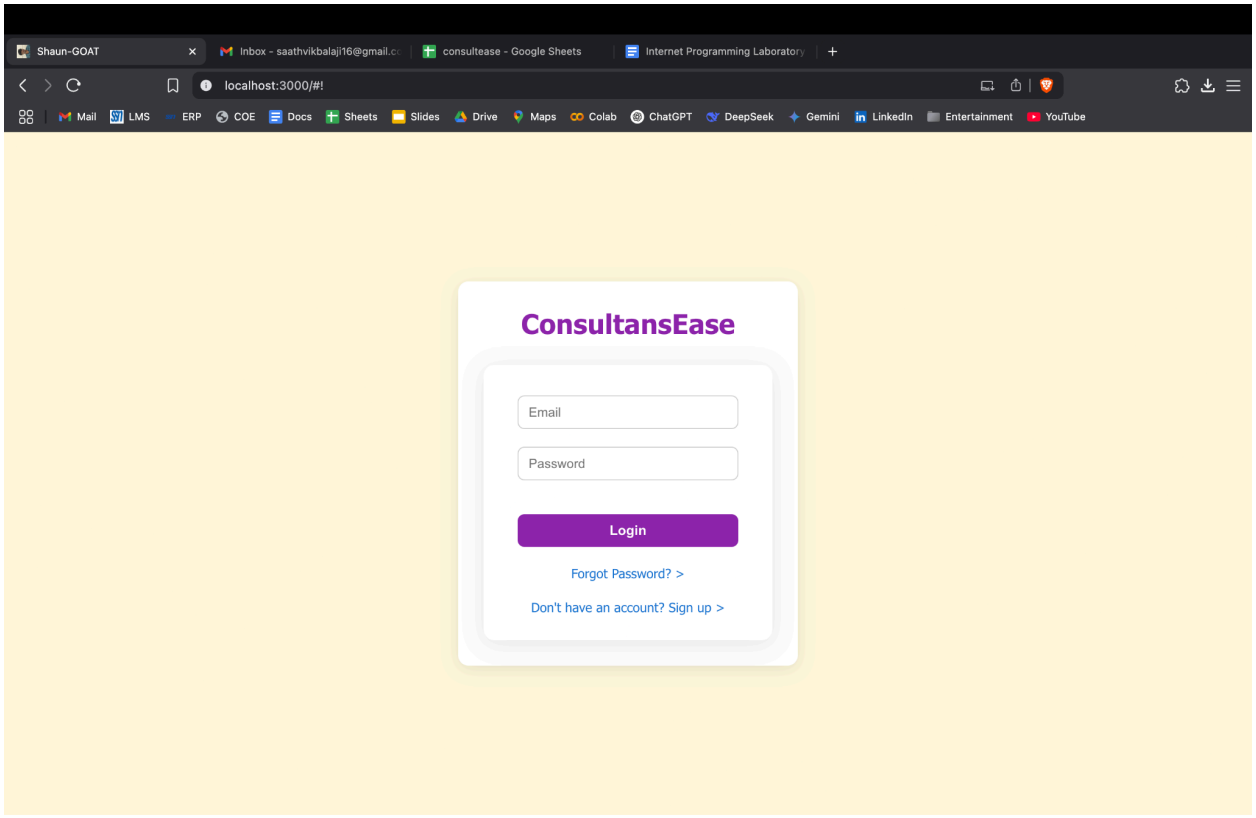
```
        spreadsheetId,
        range: `Sheet1!A:M`,
        valueInputOption: "USER_ENTERED",
        resource: { values: rows },
      });

      res.status(200).json({ message: "Data updated successfully" });
    } catch (error) {
      console.error("Error updating data:", error);
      res.status(500).json({ error: "Error updating data" });
    }
  }
);

export default router;
```

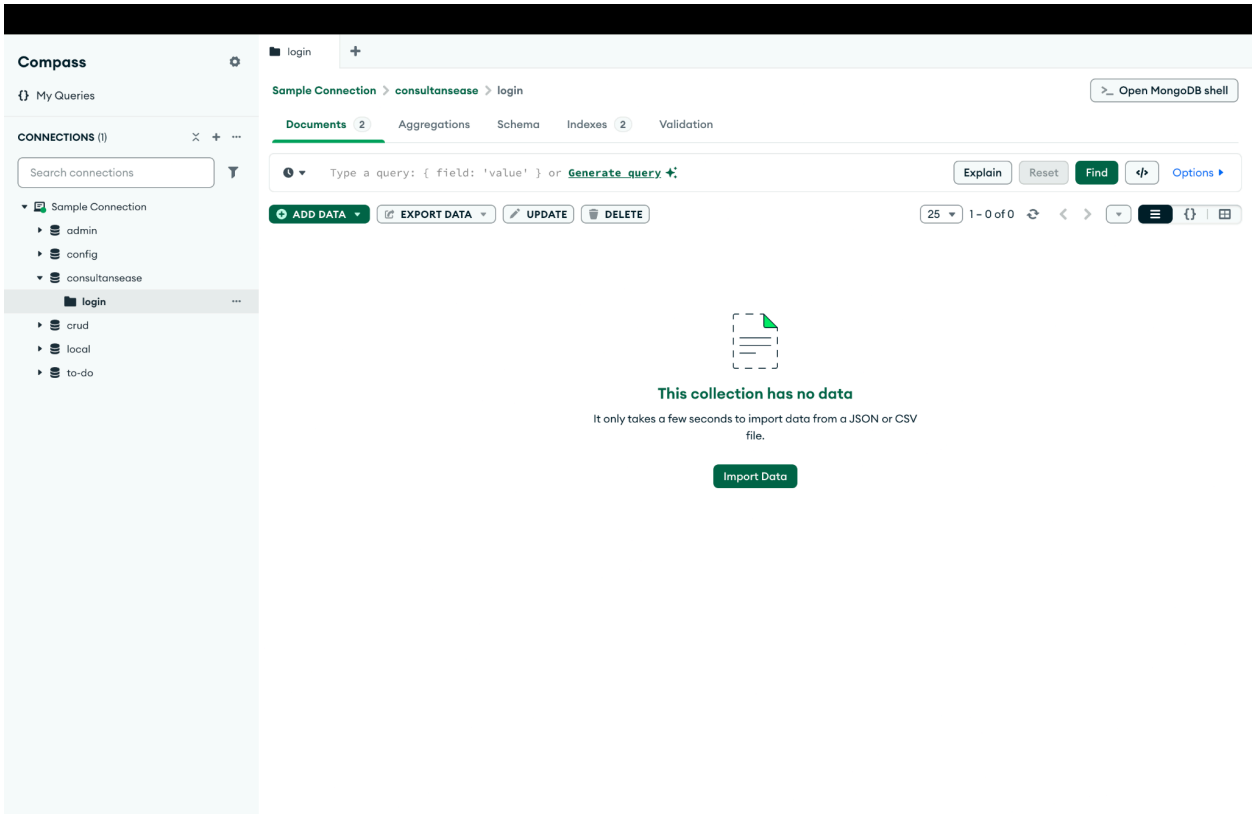
OUTPUT:

Initial State:

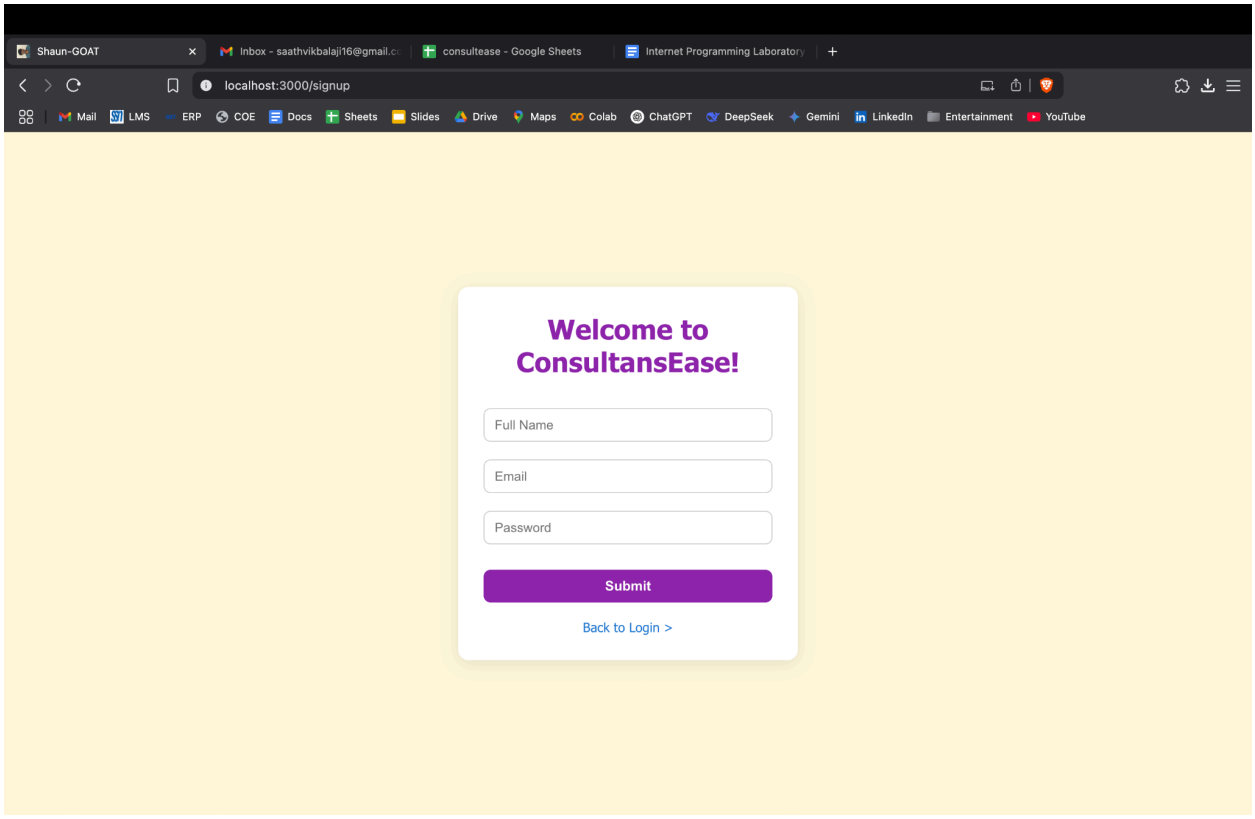
```
Compiled successfully!

You can now view consultansease in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://10.121.34.213:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

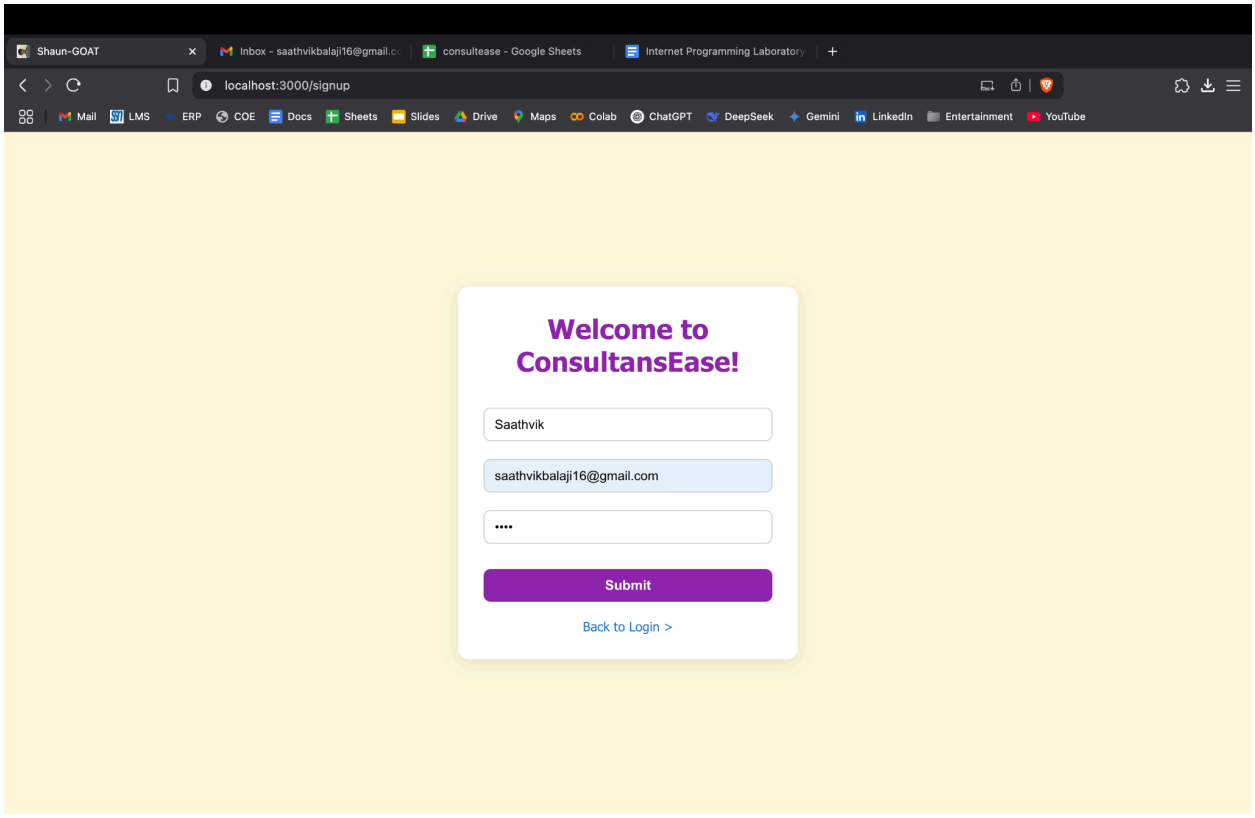
```
(base) saathvik@Saathviks-MacBook-Air backend % npm start

> backend@1.0.0 start
> nodemon server.js

[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully!
Server running on PORT 5050
```

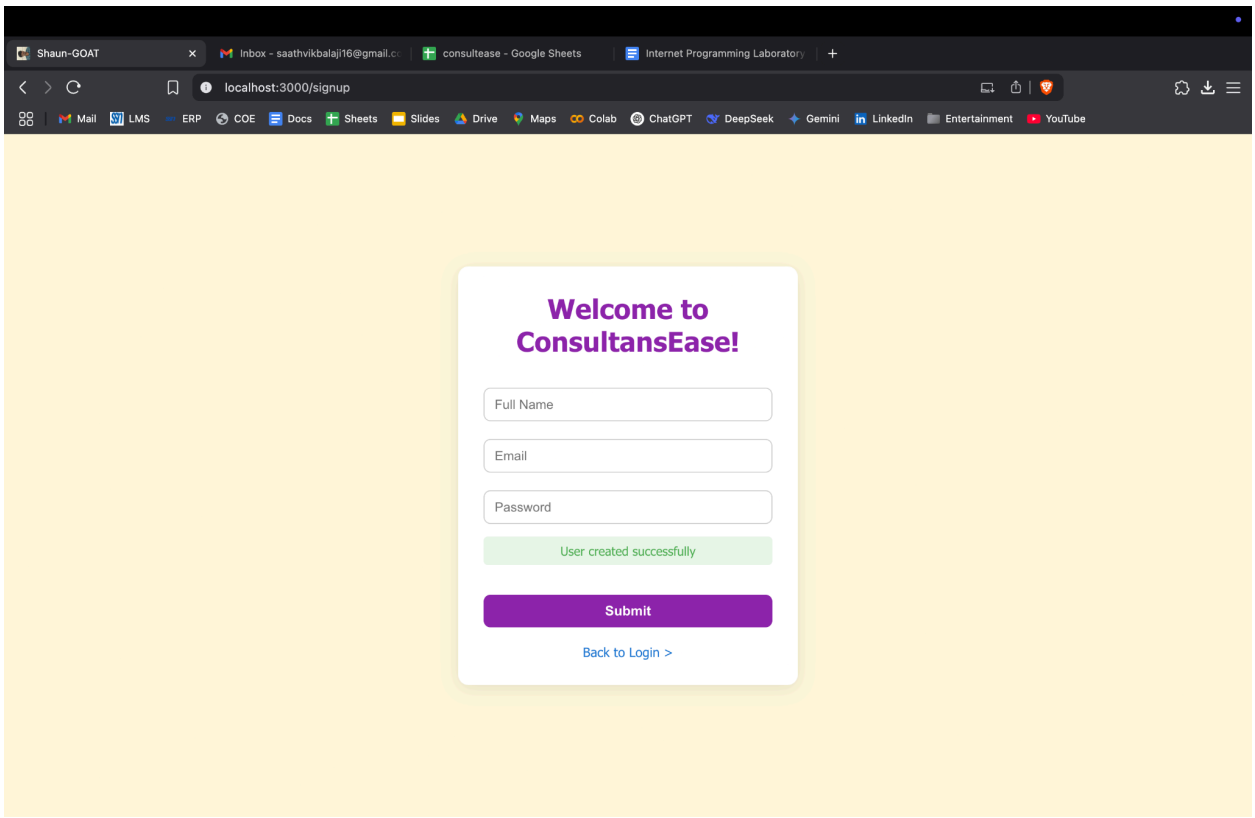


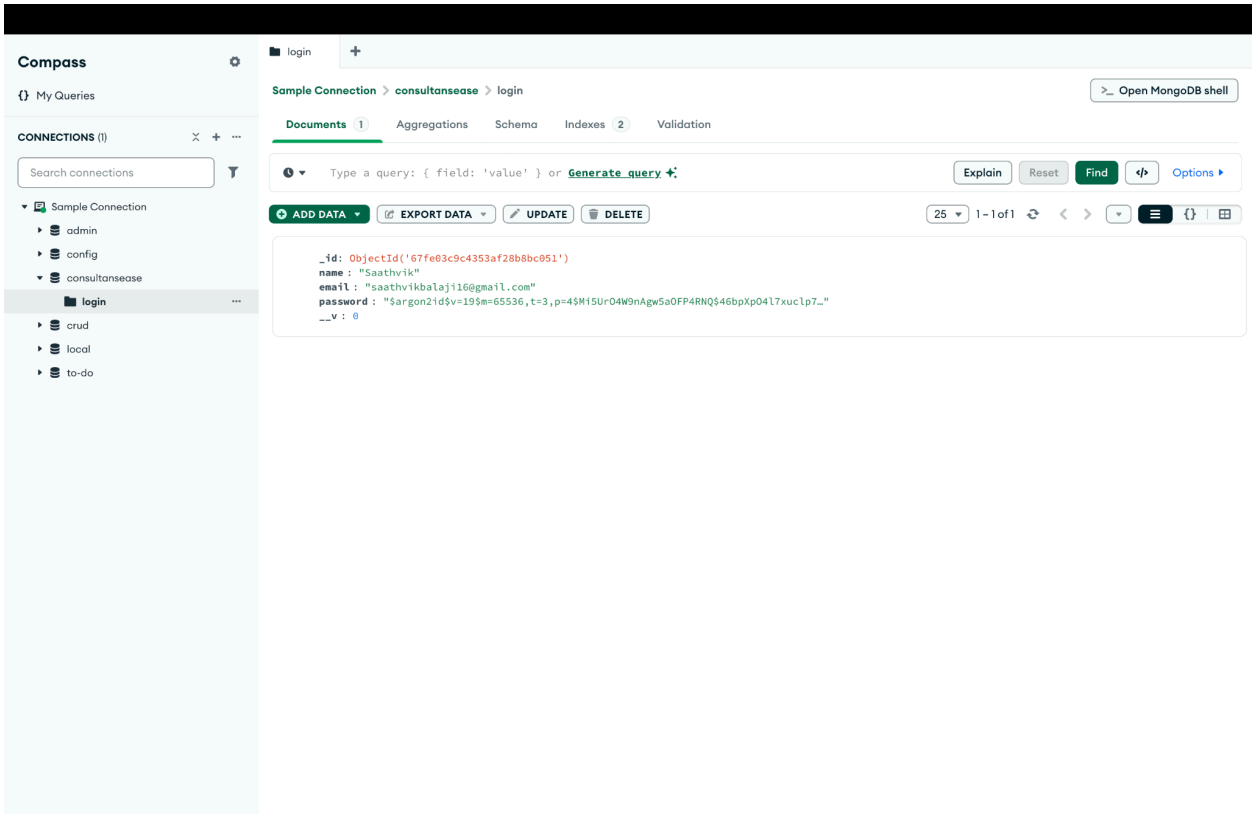

Compass
My Queries
CONNECTIONS (1)
Search connections
Sample Connection
admin
config
consultansease
login
crud
local
to-do
login
Sample Connection > consultansease > login
Open MongoDB shell
Documents 2
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 0 of 0
This collection has no data
It only takes a few seconds to import data from a JSON or CSV file.
Import Data

Sign-up:

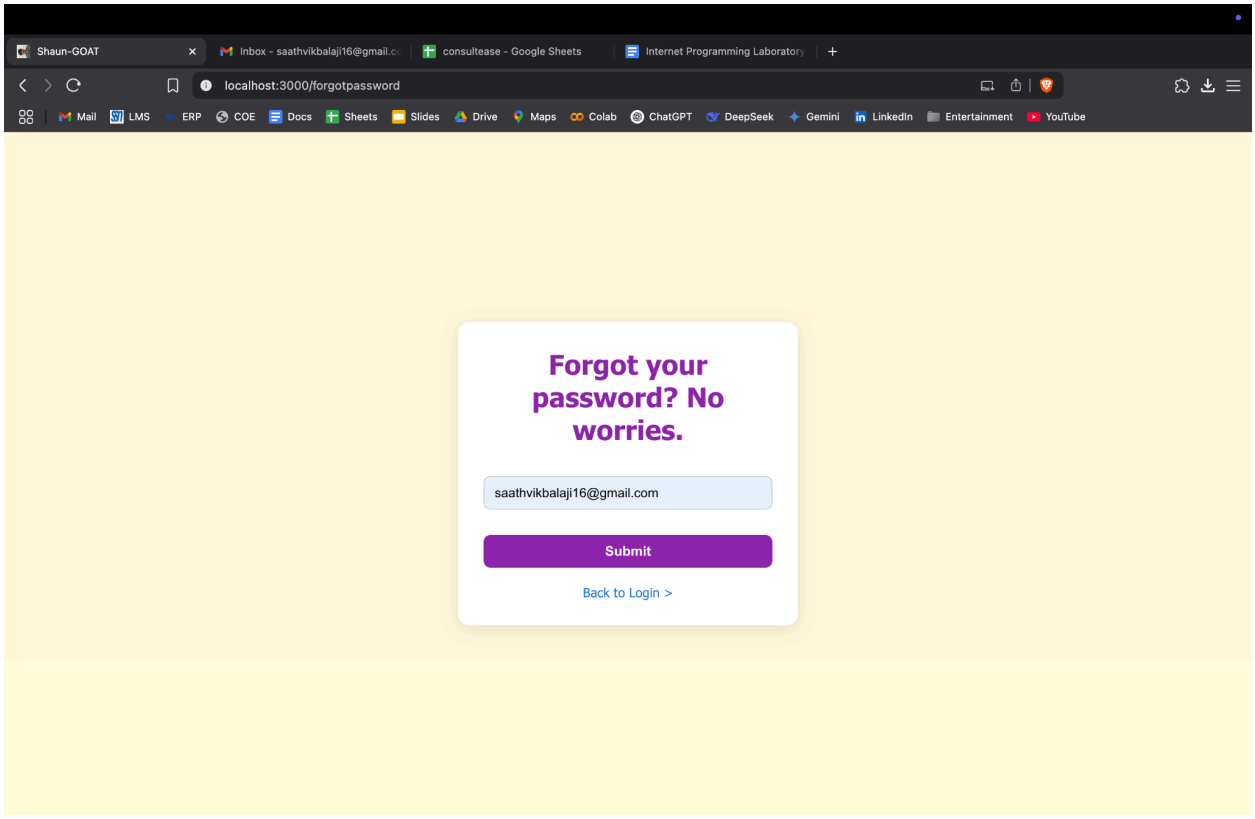

Shaun-GOAT
Inbox - saathvikbalaji16@gmail.c
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000/signup
Mail
LMS
ERP
COE
Docs
Sheets
Slides
Drive
Maps
Colab
ChatGPT
DeepSeek
Gemini
LinkedIn
Entertainment
YouTube
Welcome to ConsultansEase!
Full Name
Email
Password
Submit
Back to Login >

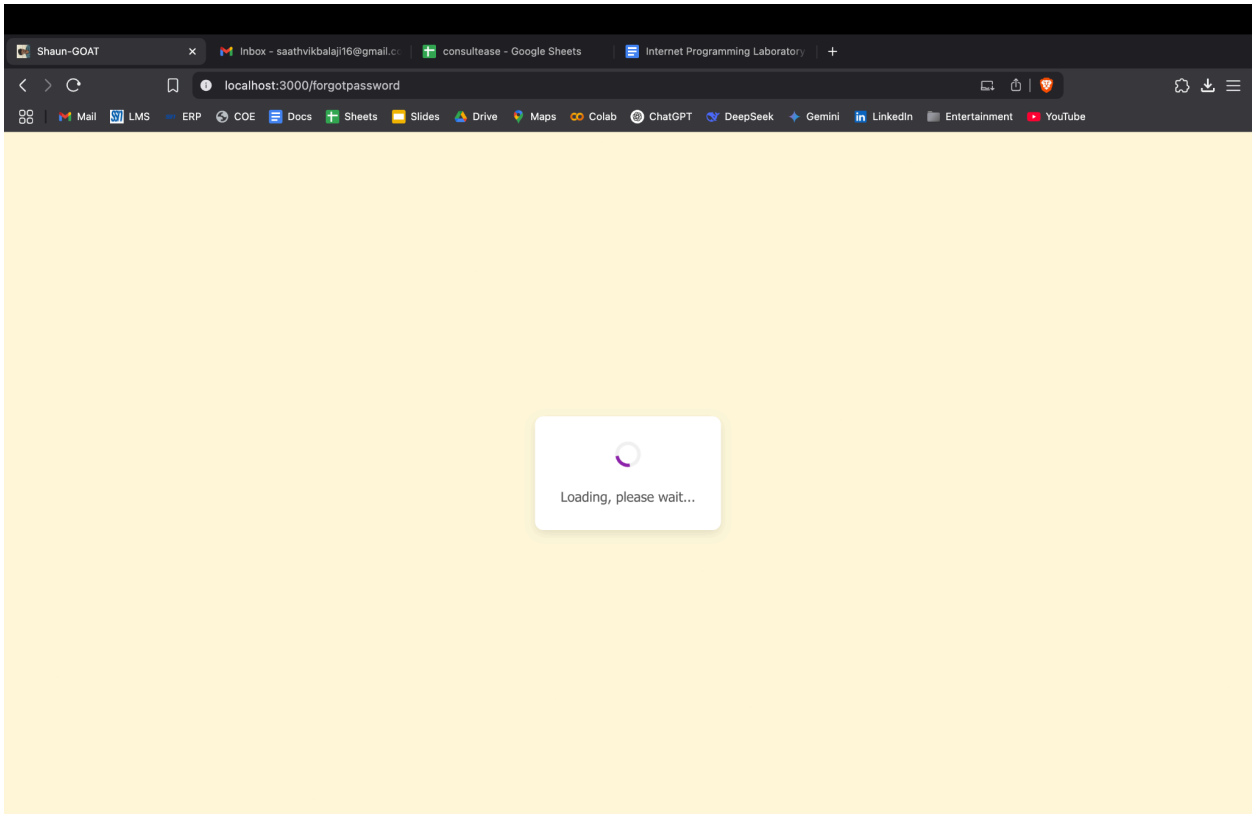

Shaun-GOAT
Inbox - saathvikbalaji16@gmail.cc
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000/signup
Mail
LMS
ERP
COE
Docs
Sheets
Slides
Drive
Maps
Colab
ChatGPT
DeepSeek
Gemini
LinkedIn
Entertainment
YouTube
Welcome to ConsultansEase!
Saathvik
saathvikbalaji16@gmail.com
Submit
Back to Login >

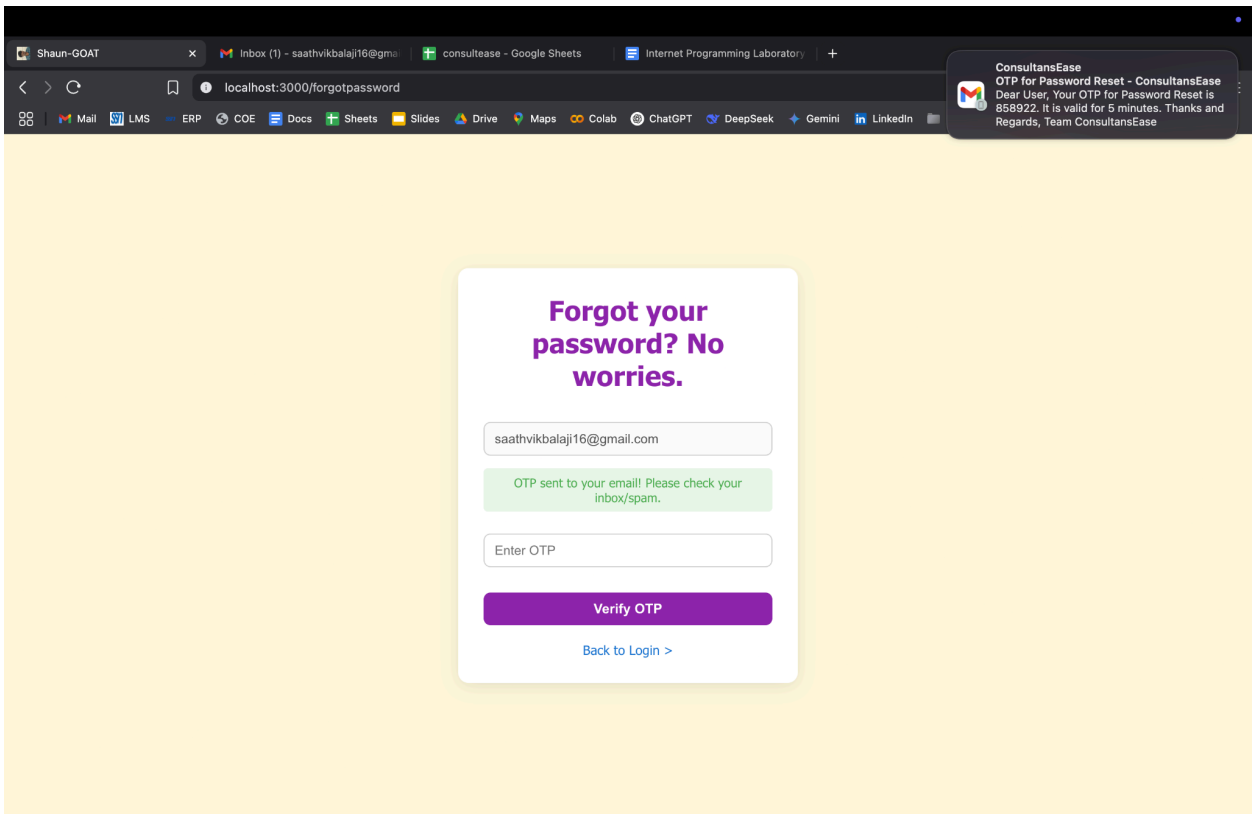

Shaun-GOAT
Inbox - saathvikbalaji16@gmail.cc
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000/signup
Mail
LMS
ERP
COE
Docs
Sheets
Slides
Drive
Maps
Colab
ChatGPT
DeepSeek
Gemini
LinkedIn
Entertainment
YouTube
Welcome to ConsultansEase!
Full Name
Email
Password
User created successfully
Submit
Back to Login >

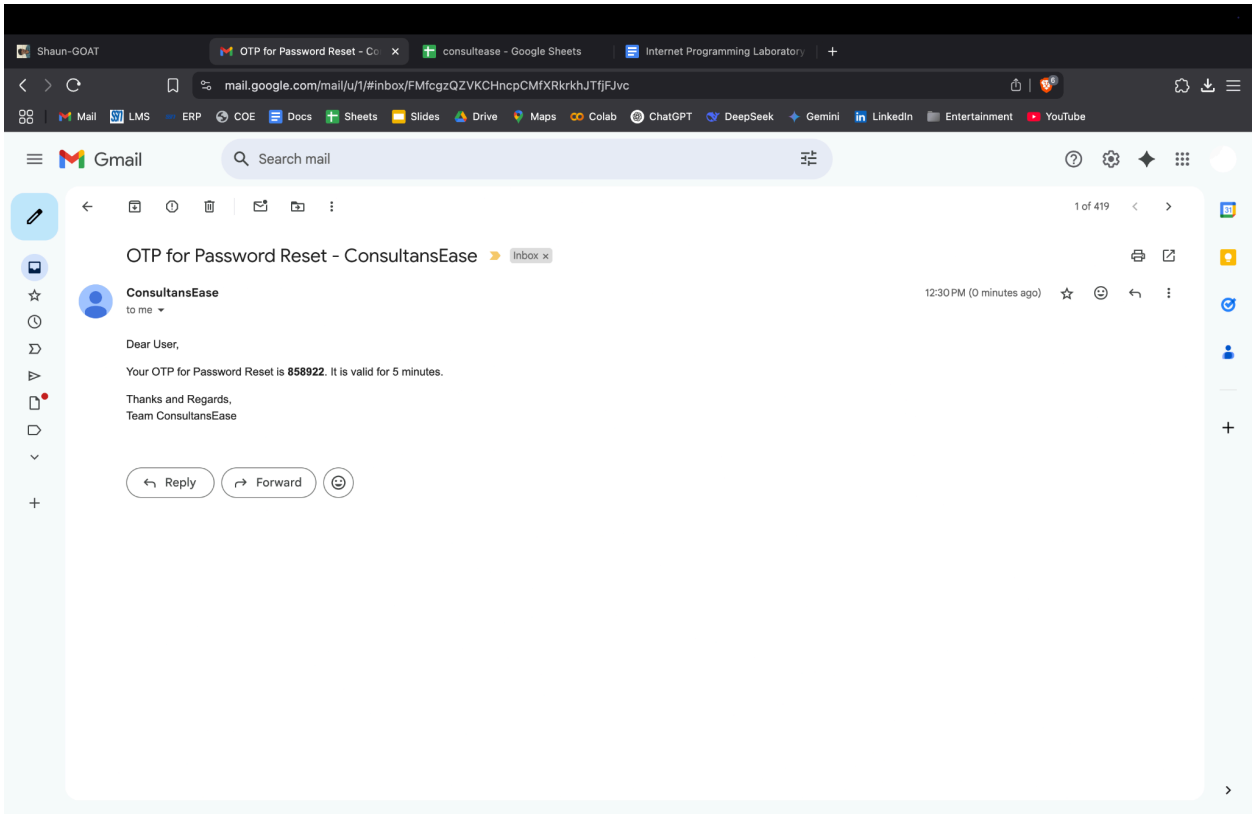


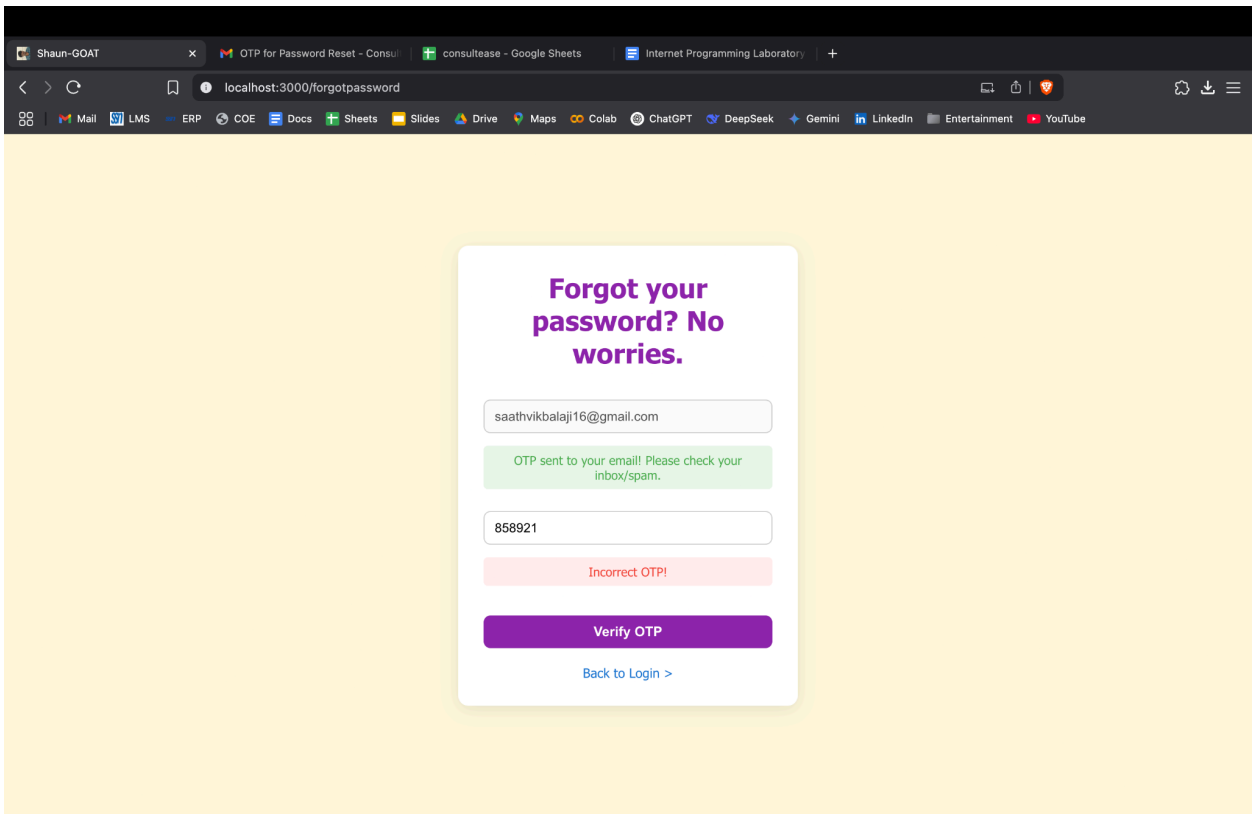
Forgot Password:

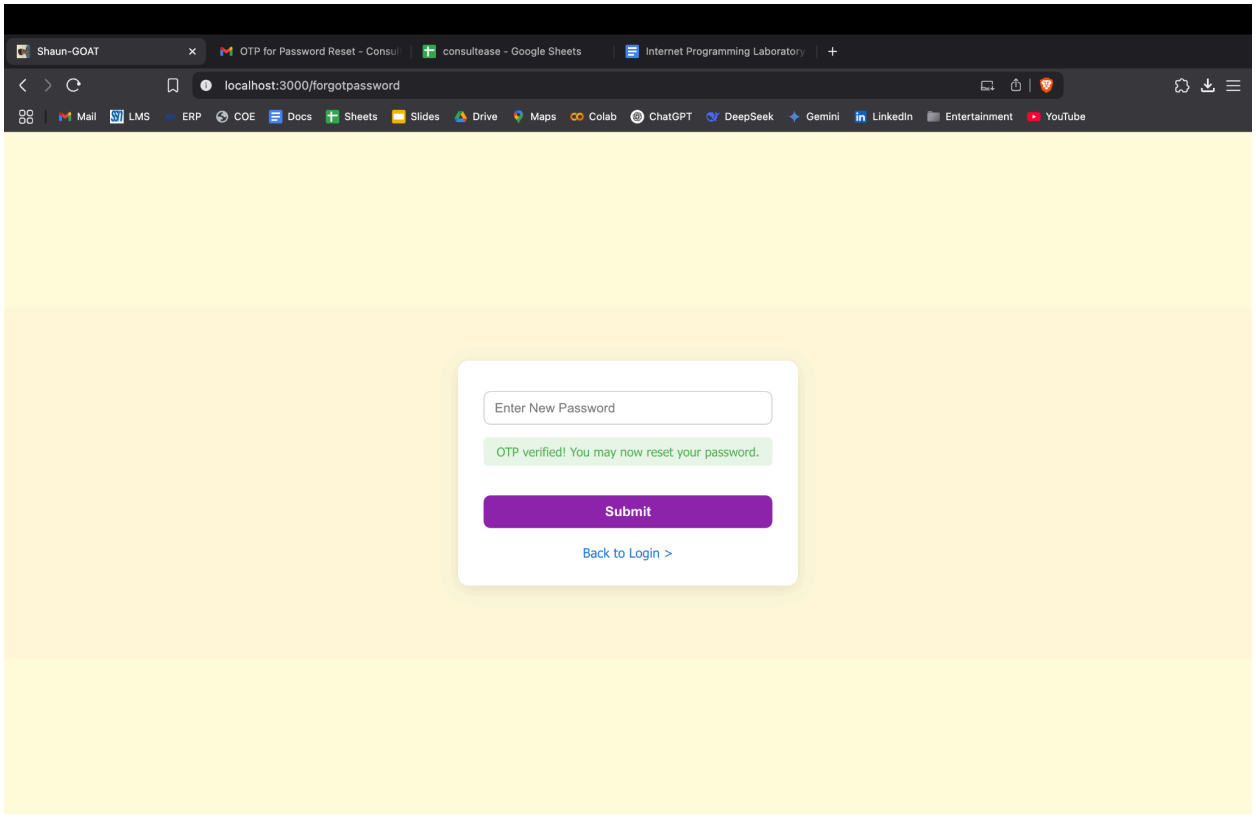


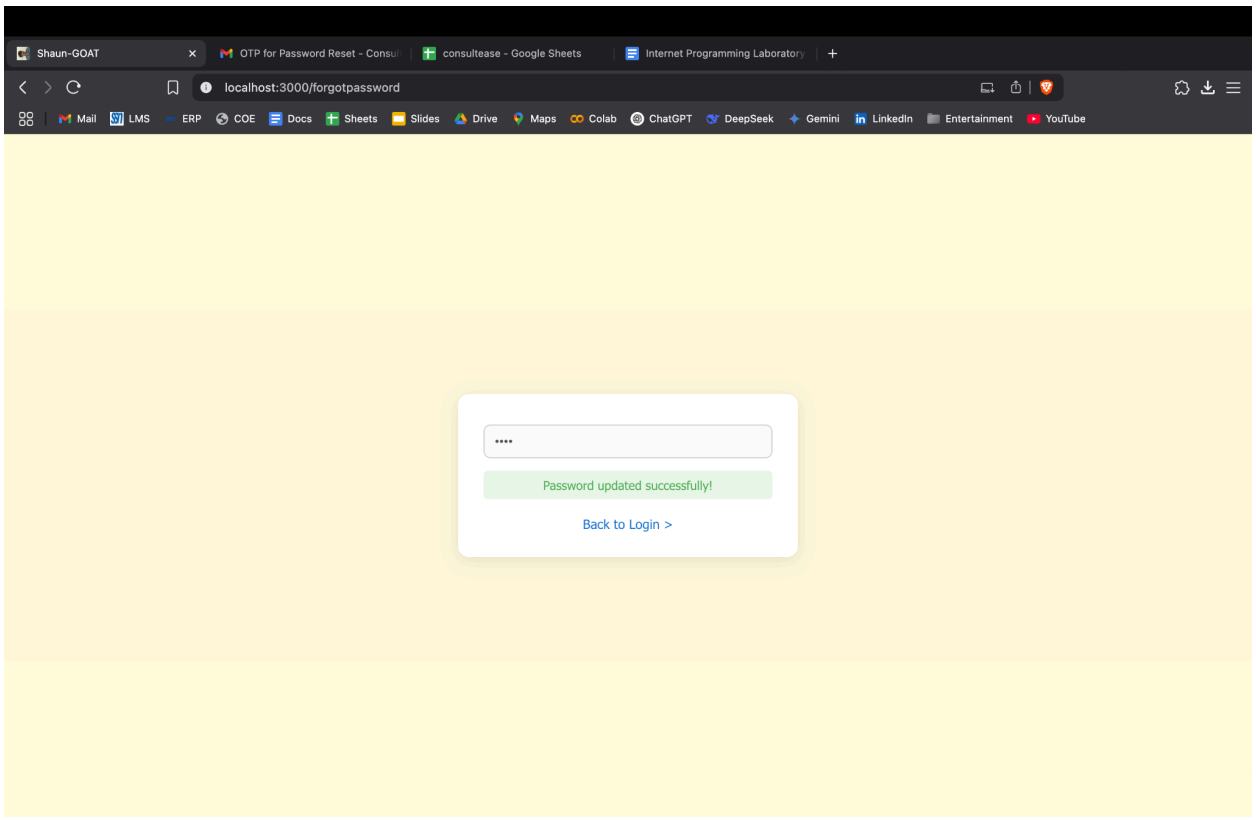

Shaun-GOAT
Inbox - saathvikbalaji16@gmail.cc
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000/forgotpassword
Mail LMS ERP COE Docs Sheets Slides Drive Maps Colab ChatGPT DeepSeek Gemini LinkedIn Entertainment YouTube
Loading, please wait...


Shaun-GOAT
Inbox (1) - saathvikbalaji16@gma
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000/forgotpassword
Mail LMS ERP COE Docs Sheets Slides Drive Maps Colab ChatGPT DeepSeek Gemini LinkedIn
ConsultansEase
OTP for Password Reset - ConsultansEase
Dear User, Your OTP for Password Reset is 858922. It is valid for 5 minutes. Thanks and Regards, Team ConsultansEase
Forgot your password? No worries.
saathvikbalaji16@gmail.com
OTP sent to your email! Please check your inbox/spam.
Enter OTP
Verify OTP
Back to Login >

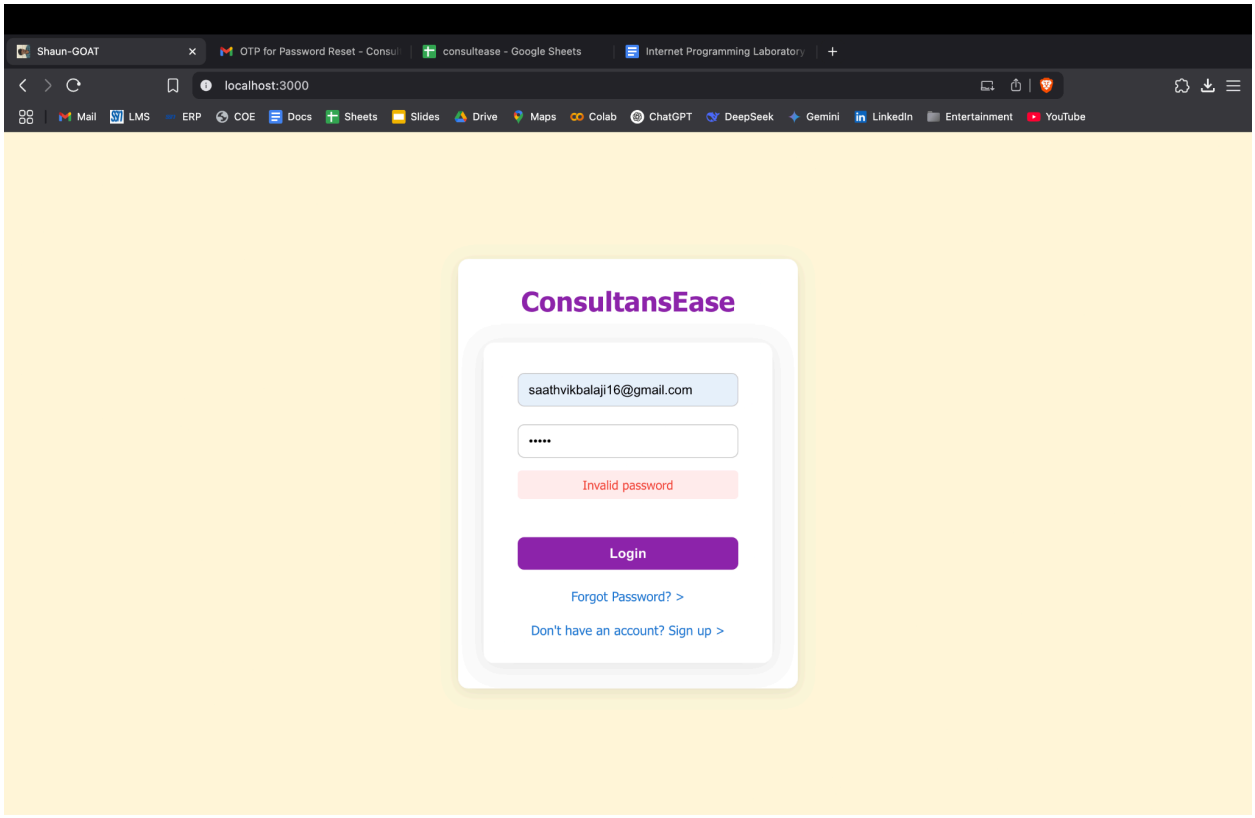

Shaun-GOAT
OTP for Password Reset - Co
consultease - Google Sheets
Internet Programming Laboratory
mail.google.com/mail/u/1/#inbox/FMfcgzQZVKCHncpCMfXRkrkhJTfjFJvc
Mail LMS ERP COE Docs Sheets Slides Drive Maps Colab ChatGPT DeepSeek Gemini LinkedIn Entertainment YouTube
Gmail
Search mail
1 of 419
OTP for Password Reset - ConsultansEase
Inbox
ConsultansEase
to me
12:30 PM (0 minutes ago)
Dear User,
Your OTP for Password Reset is 858922. It is valid for 5 minutes.
Thanks and Regards,
Team ConsultansEase
Reply
Forward

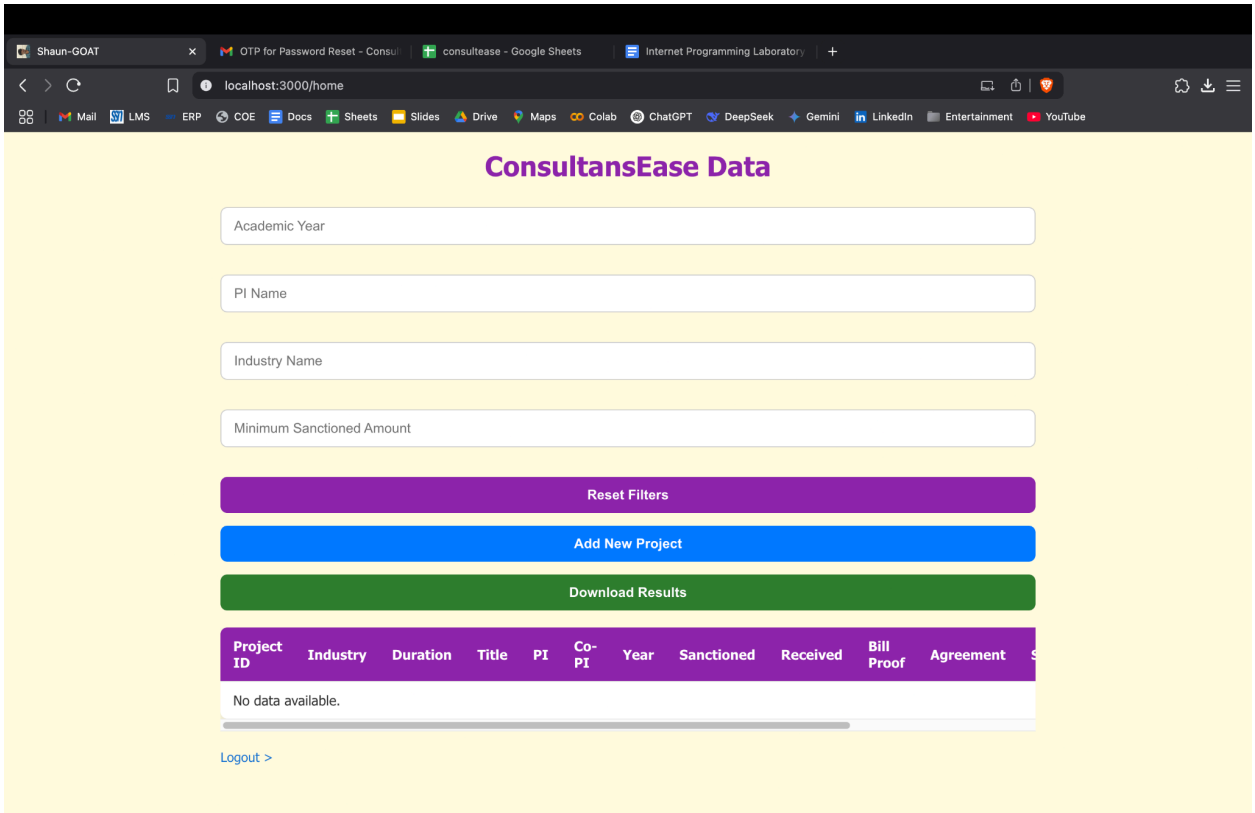

Shaun-GOAT
OTP for Password Reset - Consul
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000/forgotpassword
Mail LMS ERP COE Docs Sheets Slides Drive Maps Colab ChatGPT DeepSeek Gemini LinkedIn Entertainment YouTube
Forgot your password? No worries.
saathvikbalaji16@gmail.com
OTP sent to your email! Please check your inbox/spam.
858921
Incorrect OTP!
Verify OTP
Back to Login >



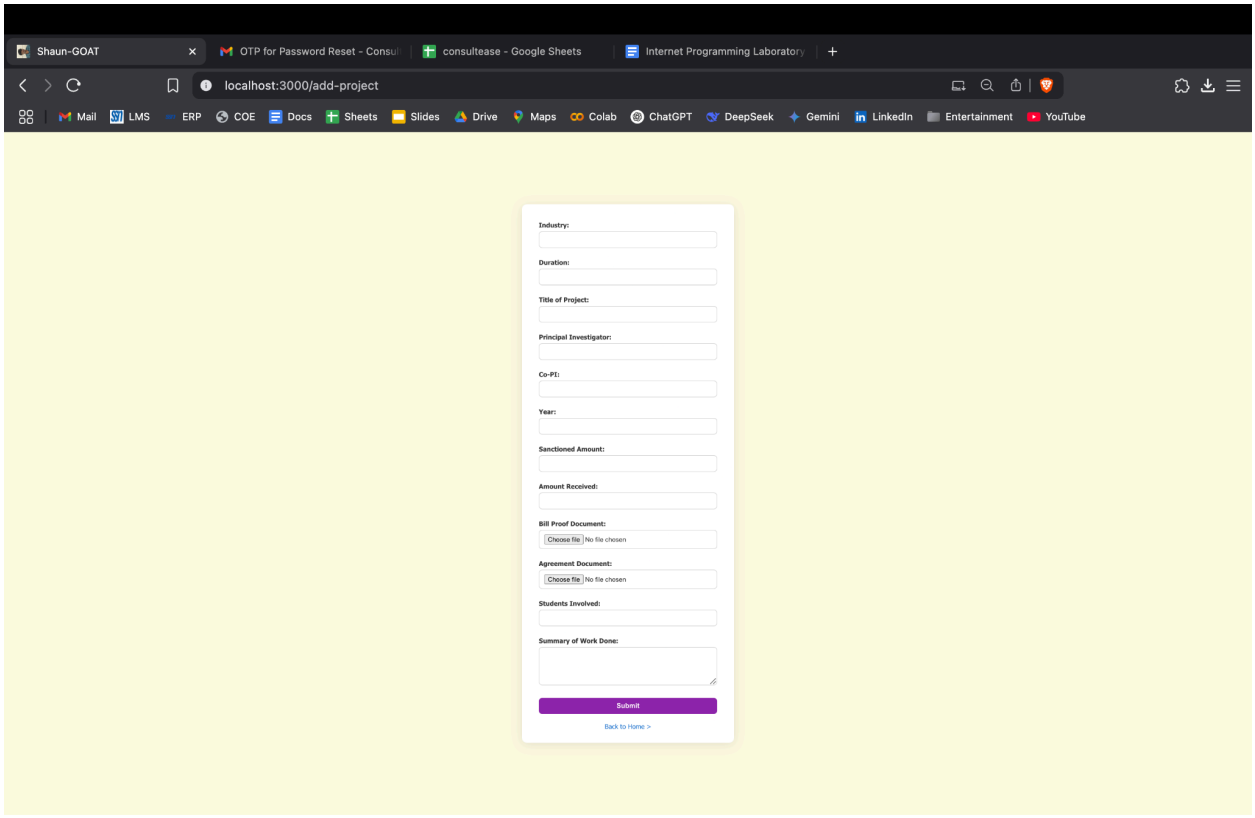


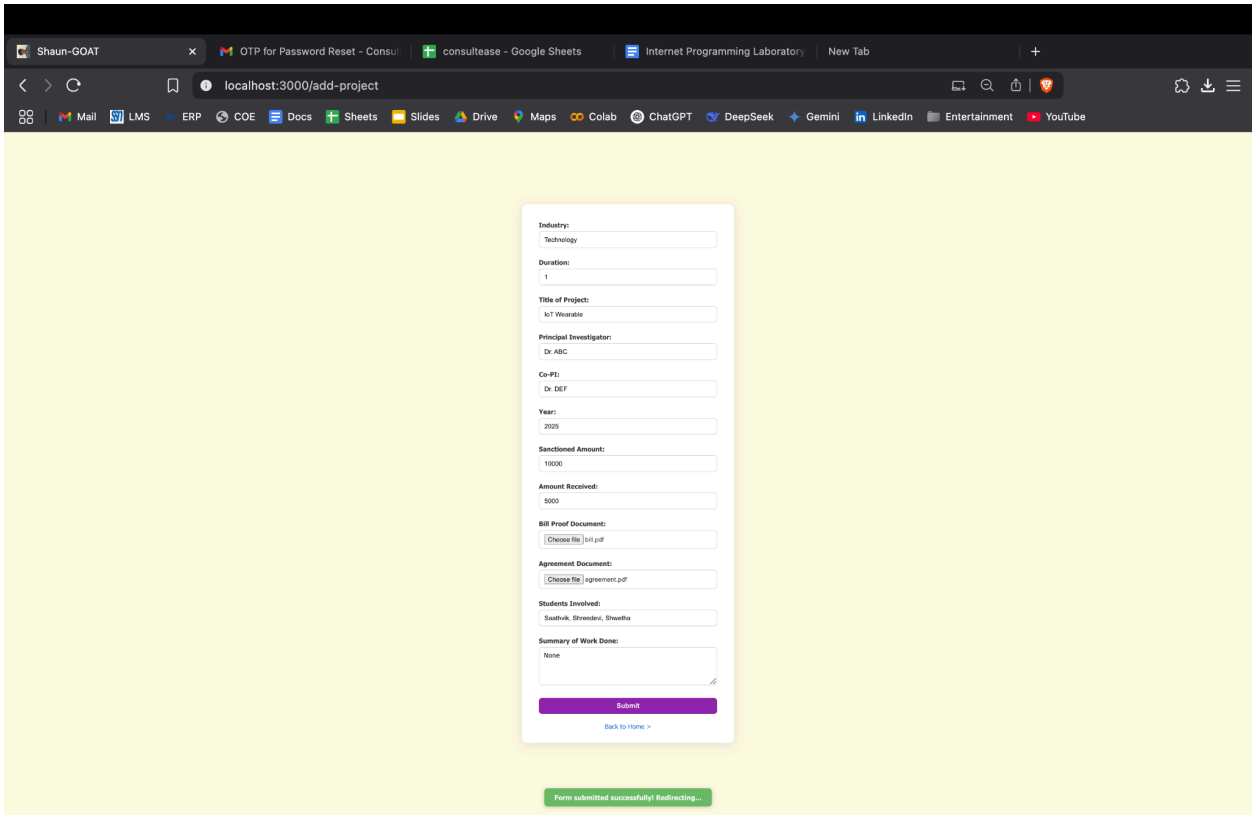
Login - Home Screen:

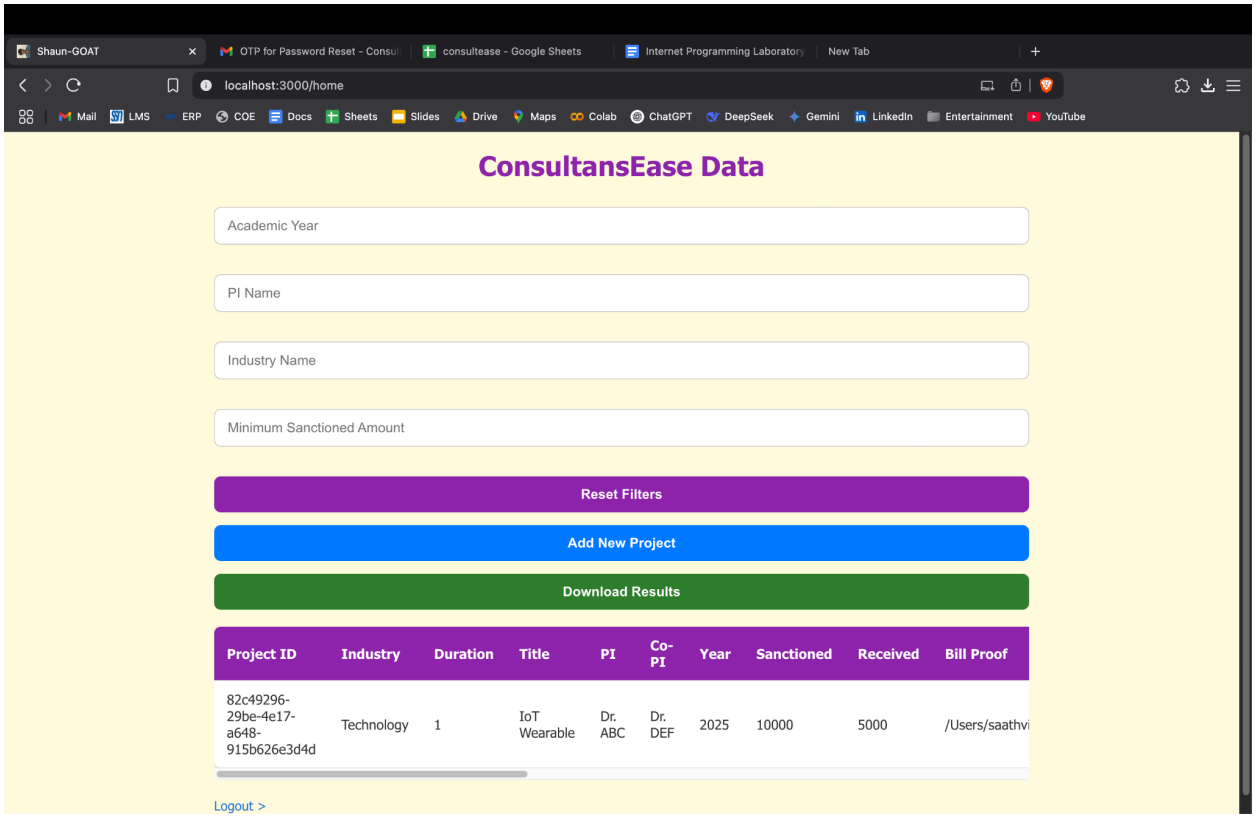

Shaun-GOAT
OTP for Password Reset - Consul
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000
Mail LMS ERP COE Docs Sheets Slides Drive Maps Colab ChatGPT DeepSeek Gemini LinkedIn Entertainment YouTube
ConsultansEase
saathvikbalaji16@gmail.com
Invalid password
Login
Forgot Password? >
Don't have an account? Sign up >


Shaun-GOAT
OTP for Password Reset - Consul
consultease - Google Sheets
Internet Programming Laboratory
localhost:3000/home
Mail LMS ERP COE Docs Sheets Slides Drive Maps Colab ChatGPT DeepSeek Gemini LinkedIn Entertainment YouTube
ConsultansEase Data
Academic Year
PI Name
Industry Name
Minimum Sanctioned Amount
Reset Filters
Add New Project
Download Results
Project ID
Industry
Duration
Title
PI
Co-PI
Year
Sanctioned
Received
Bill Proof
Agreement
No data available.
Logout >

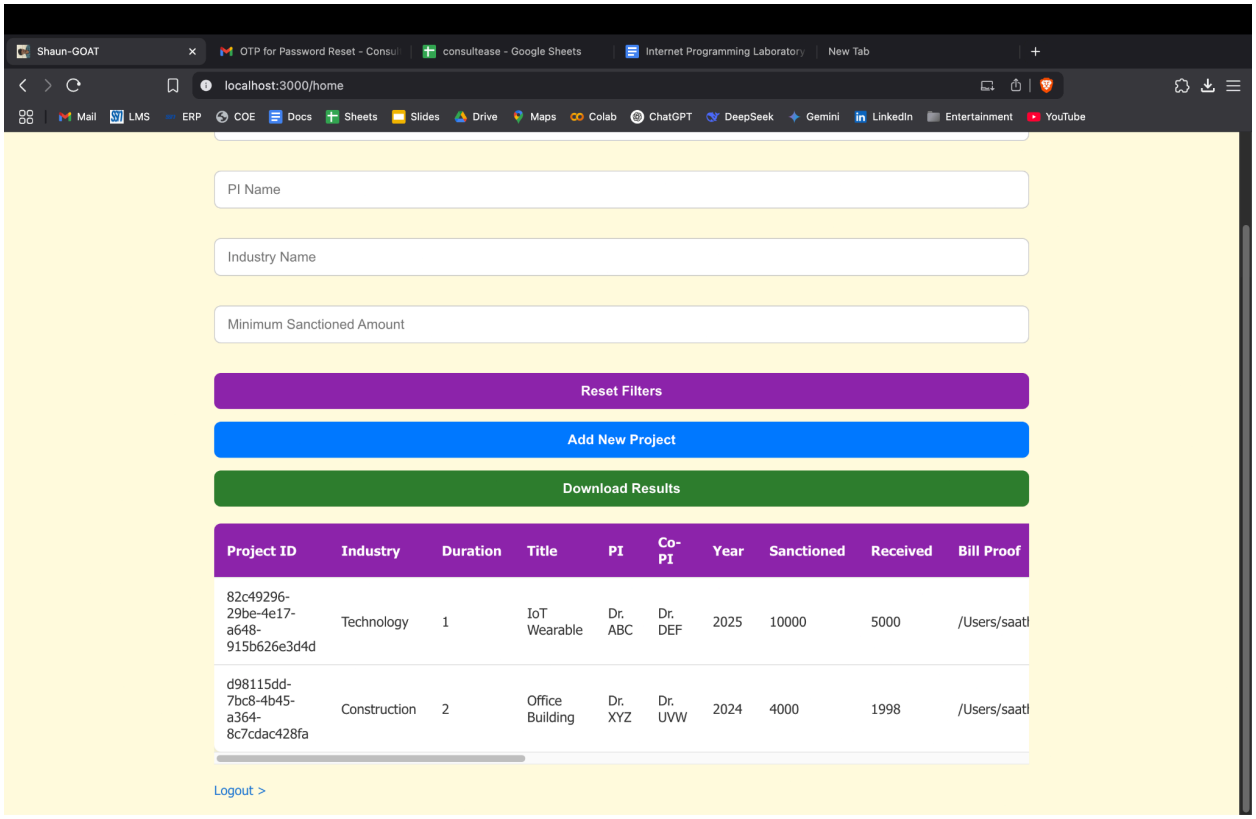
Add New Project:



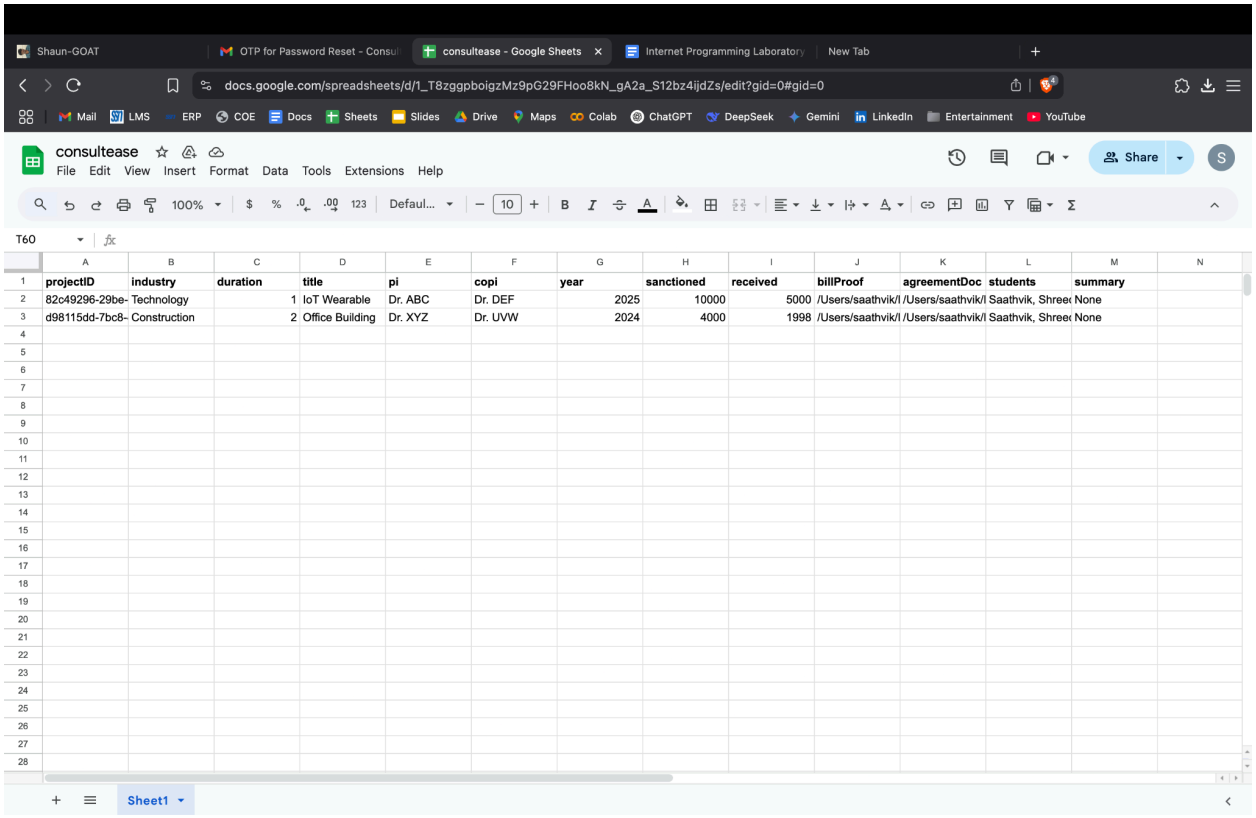




Adding More Projects:



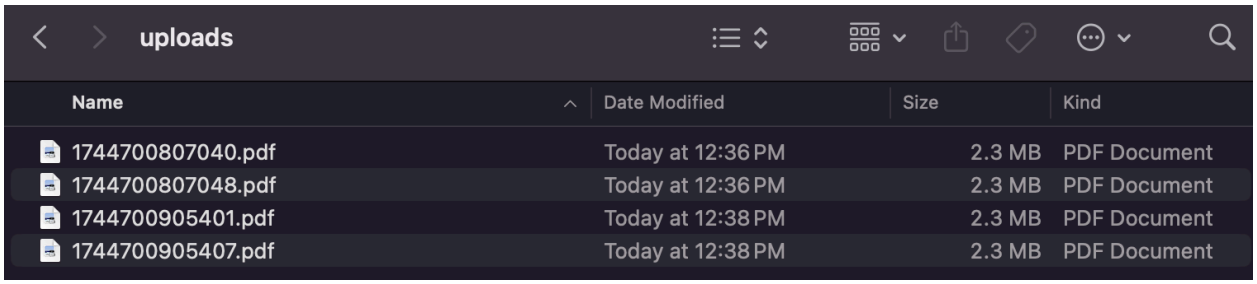
Sheets (updated):



consultease

projectID	industry	duration	title	pi	copi	year	sanctioned	received	billProof	agreementDoc	students	summary
82c49296-29be-	Technology	1	IoT Wearable	Dr. ABC	Dr. DEF	2025	10000	5000	/Users/saathvik/I	/Users/saathvik/I	Saathvik, Shreek	None
d98115dd-7bc8-	Construction	2	Office Building	Dr. XYZ	Dr. UVW	2024	4000	1998	/Users/saathvik/I	/Users/saathvik/I	Saathvik, Shreek	None

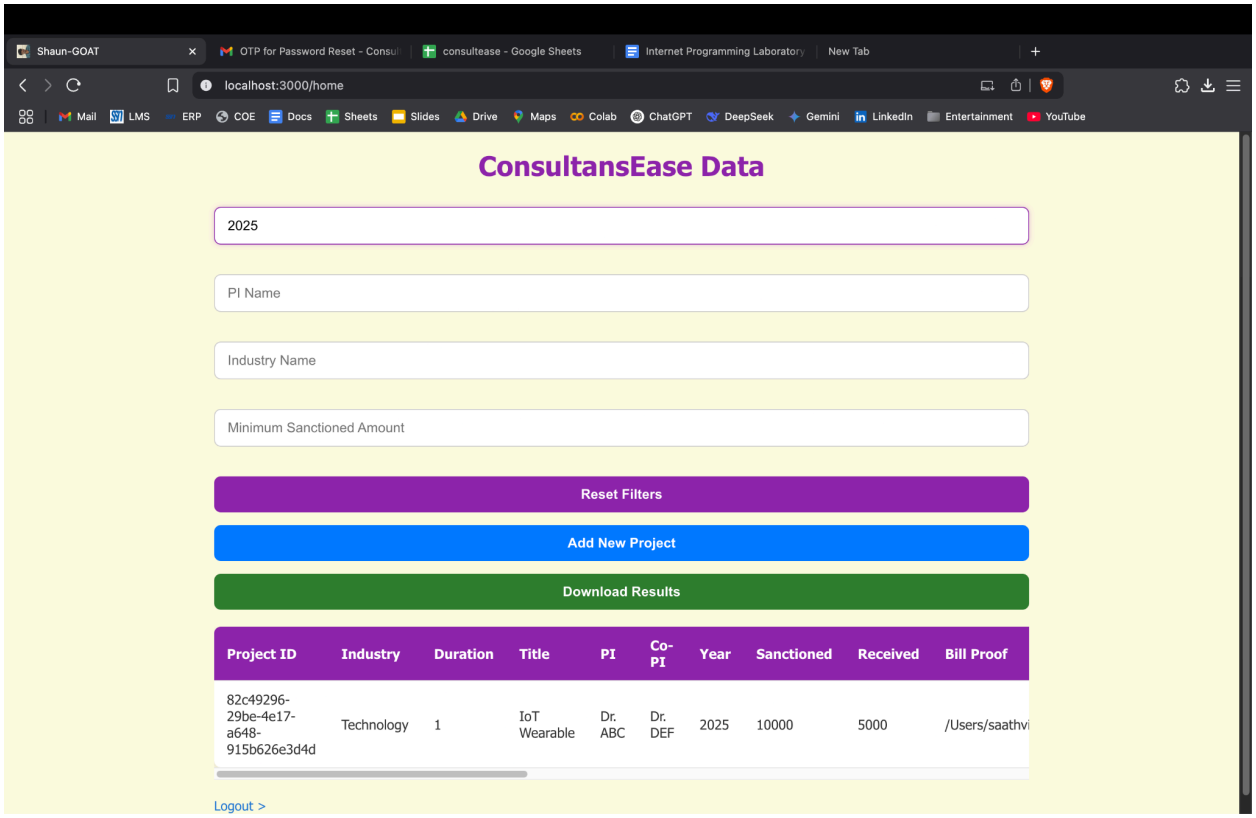
uploads:



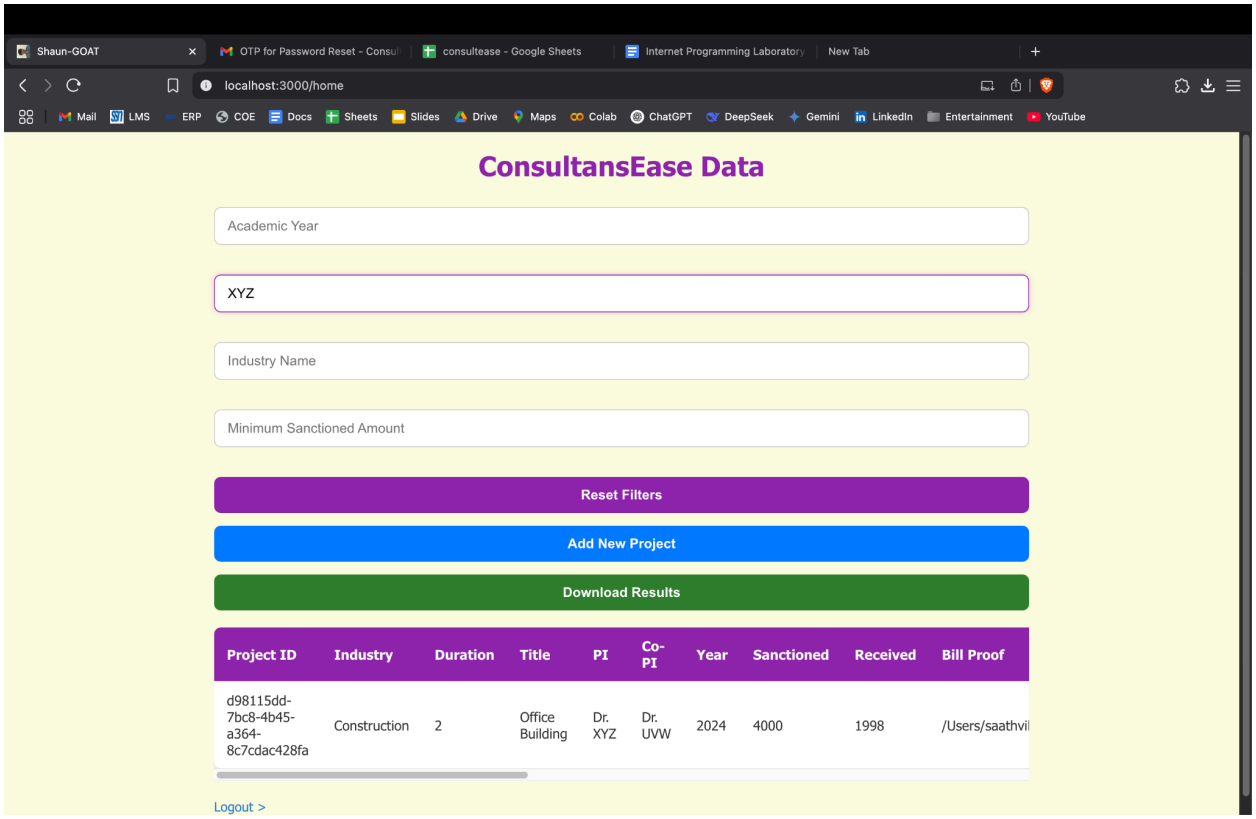
uploads

Name	Date Modified	Size	Kind
1744700807040.pdf	Today at 12:36 PM	2.3 MB	PDF Document
1744700807048.pdf	Today at 12:36 PM	2.3 MB	PDF Document
1744700905401.pdf	Today at 12:38 PM	2.3 MB	PDF Document
1744700905407.pdf	Today at 12:38 PM	2.3 MB	PDF Document

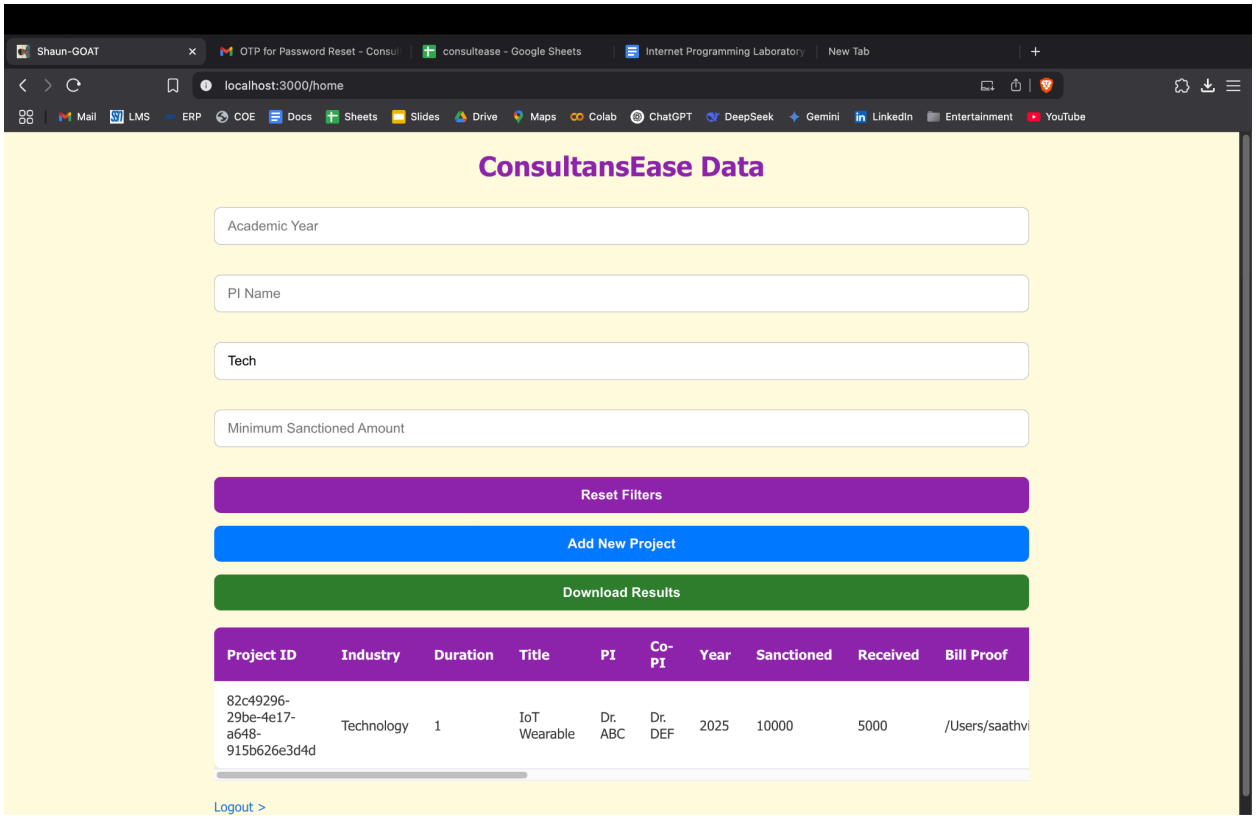
Search by Year:



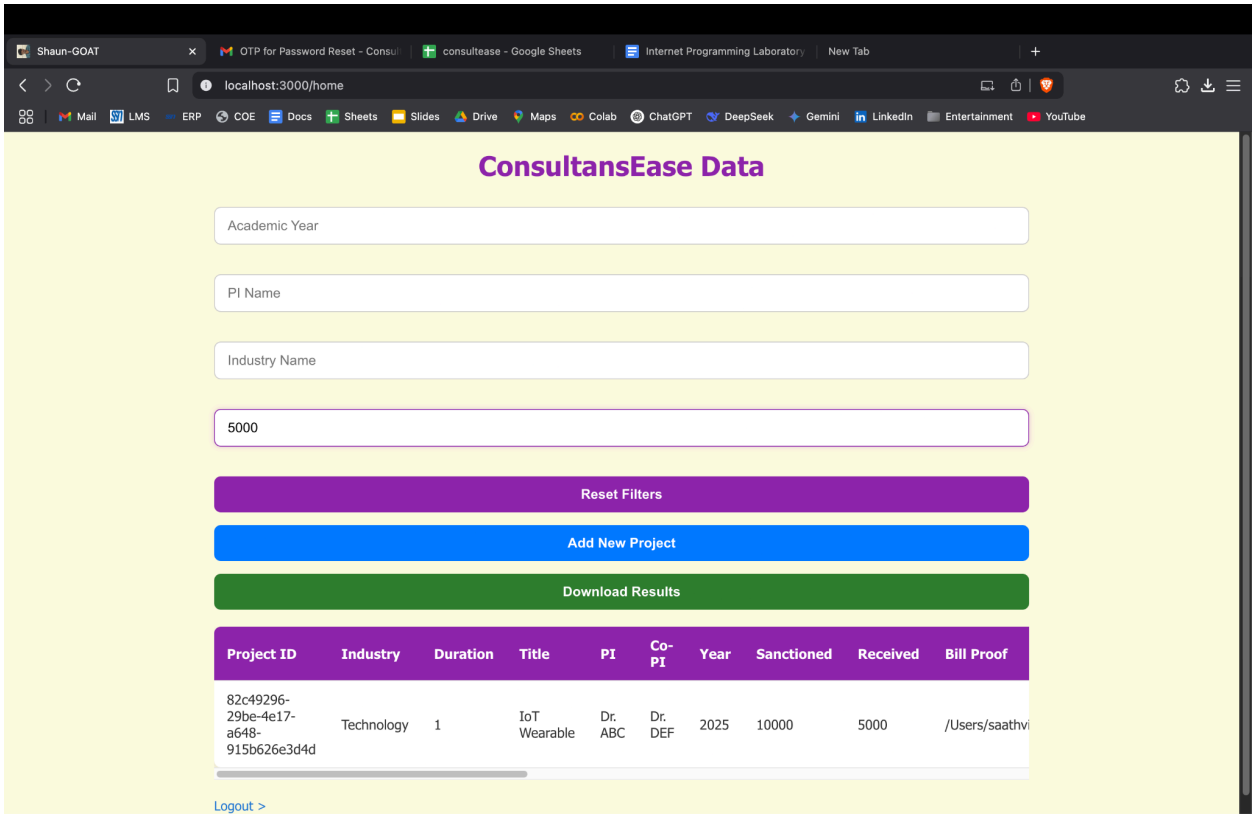
Search by PI Name:



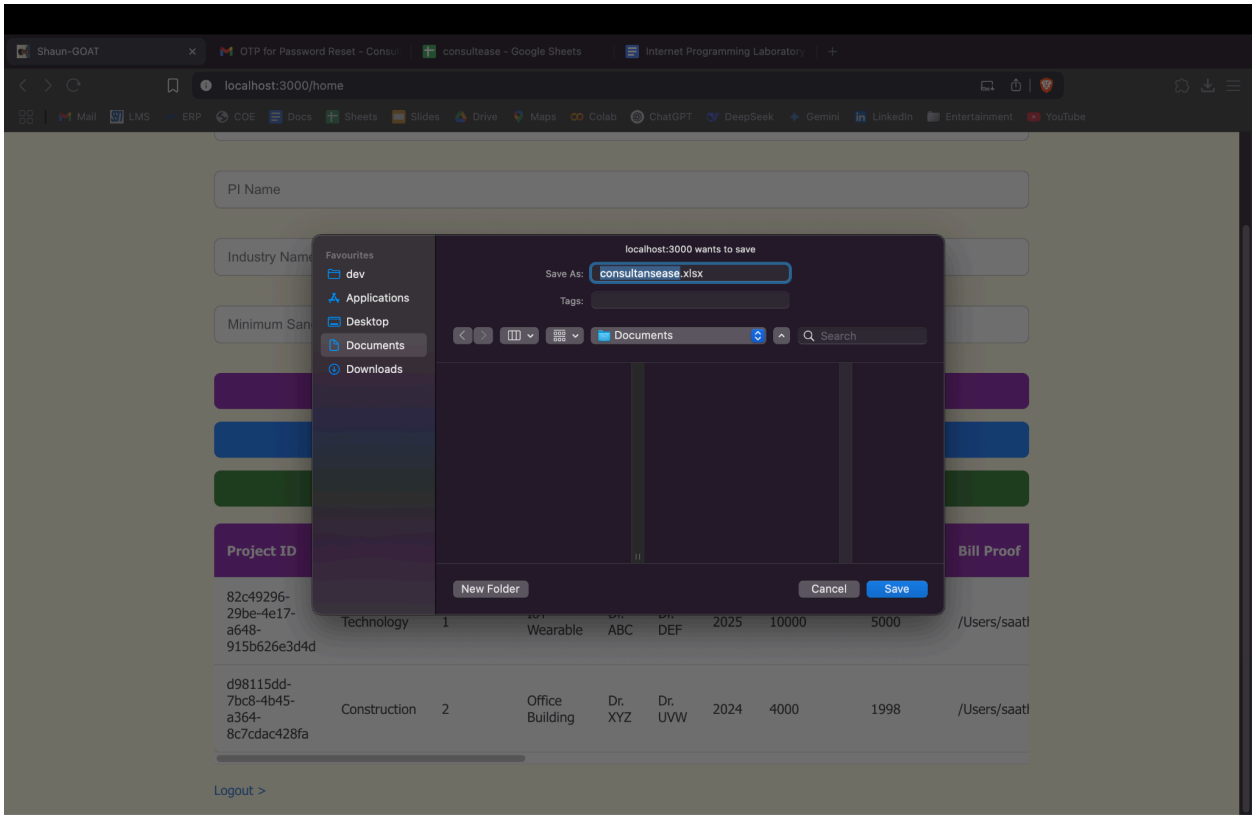
Search by Industry:

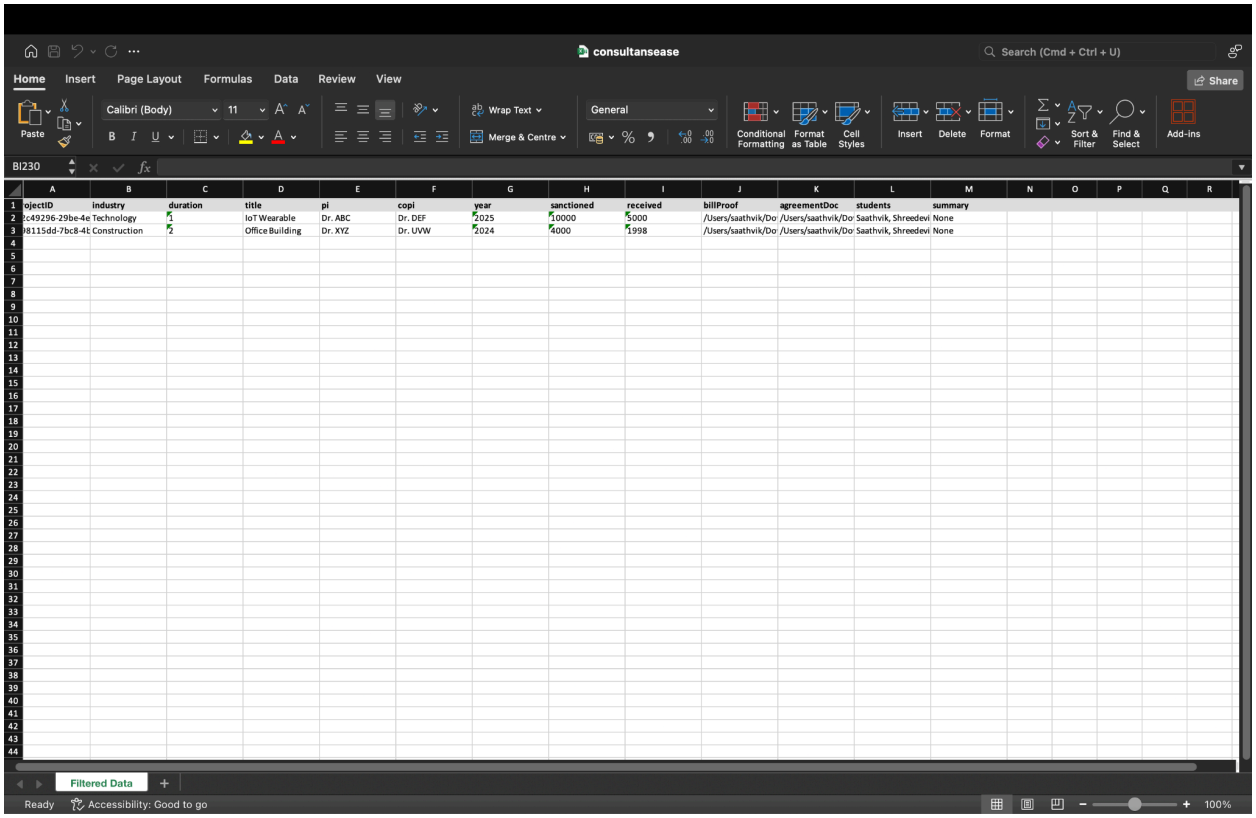


Search by Amount:

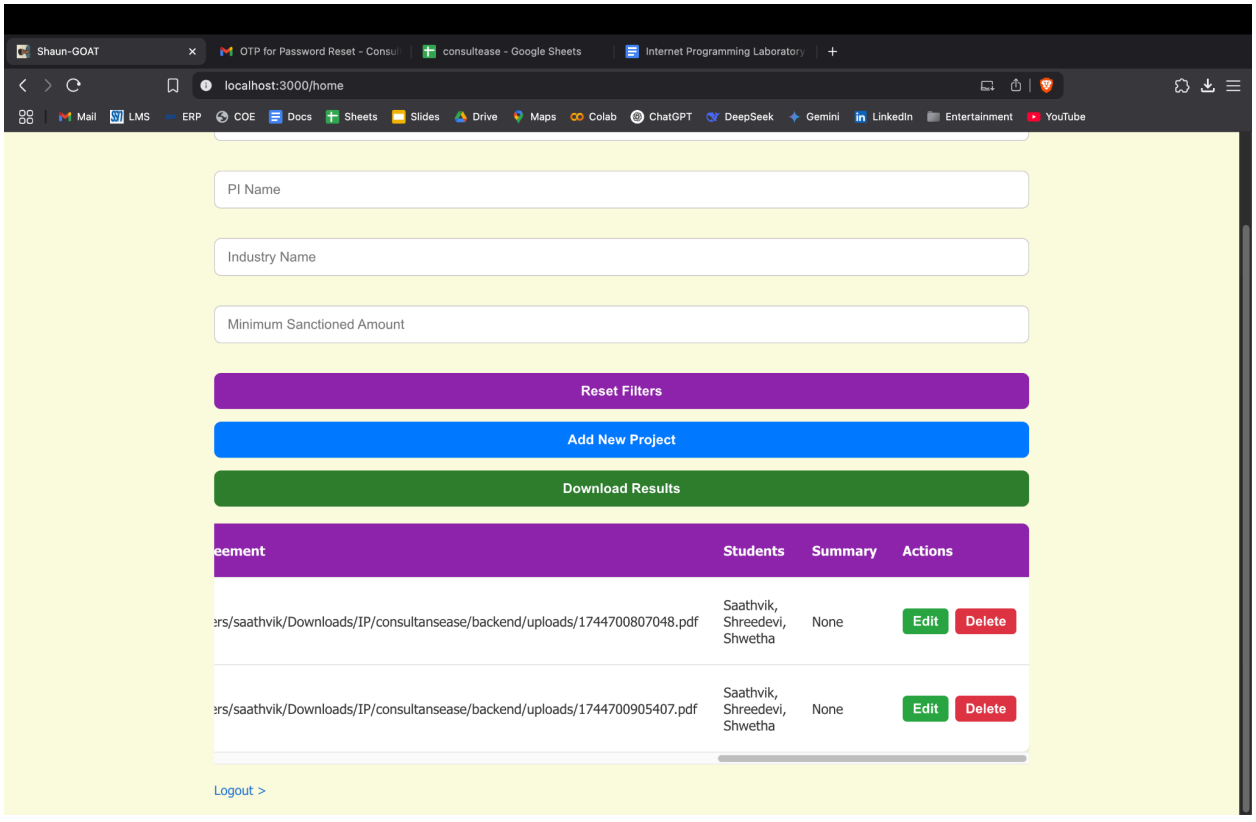


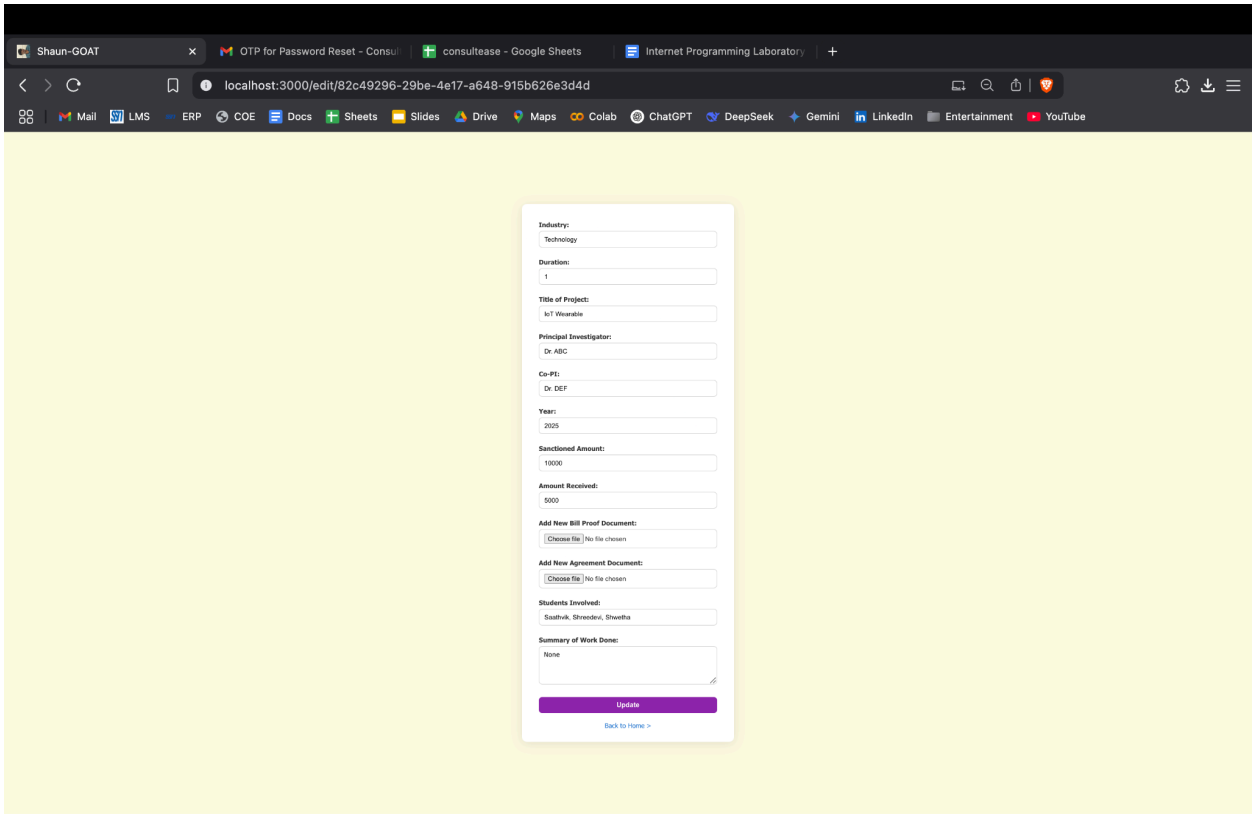
Export Details:



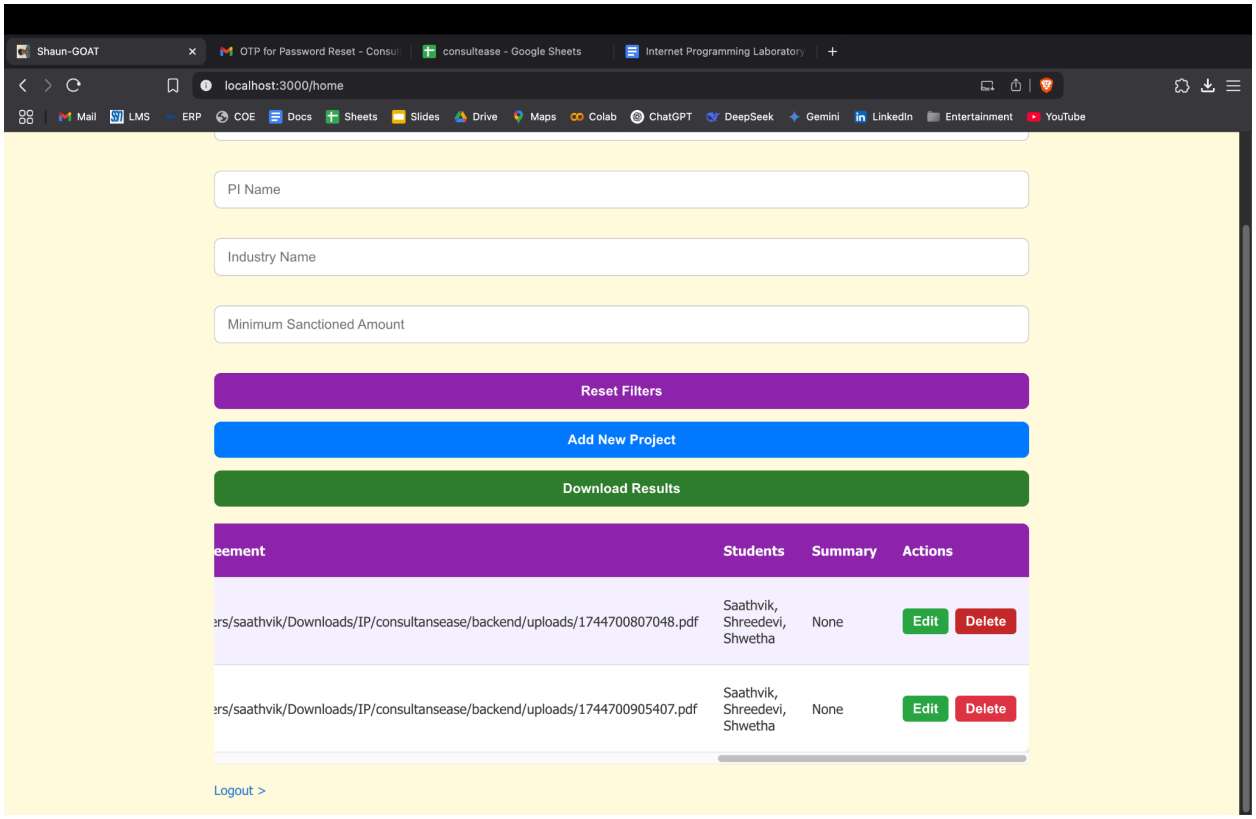


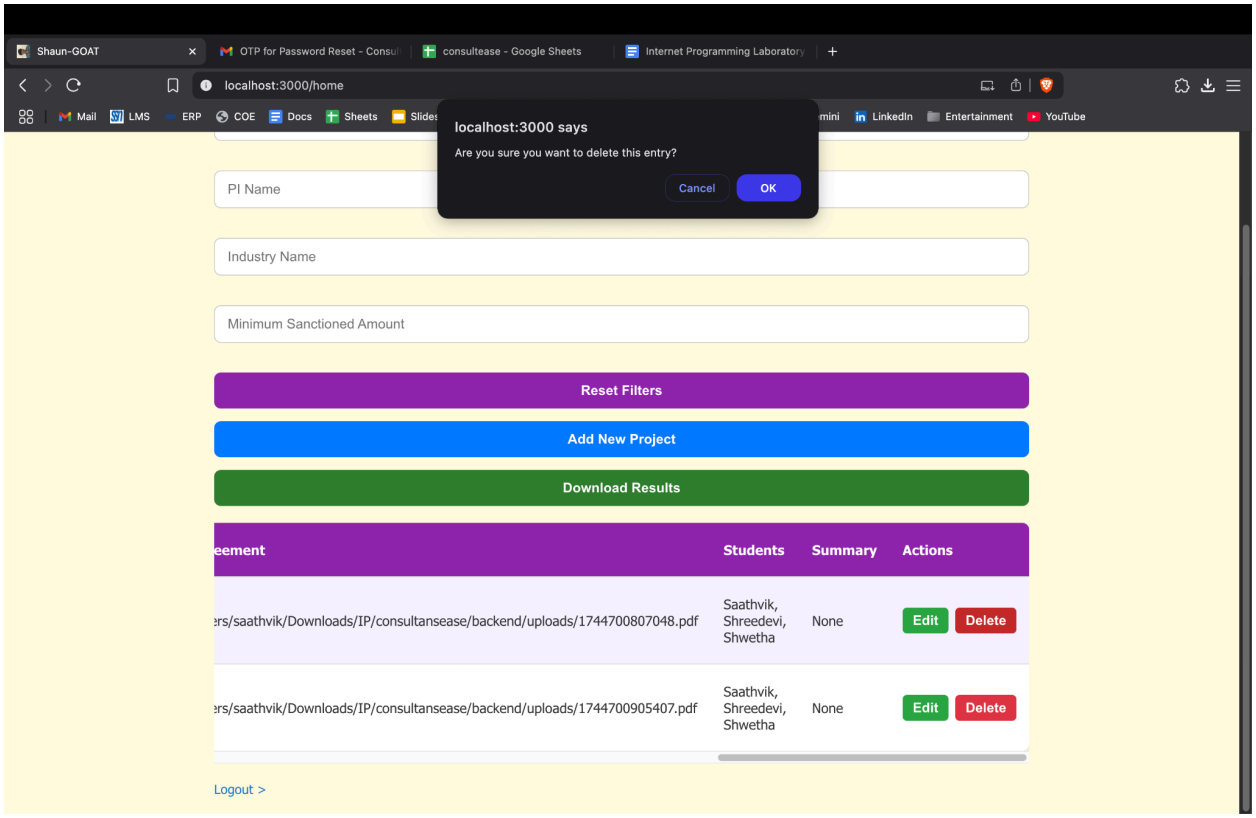
Edit Functionality:

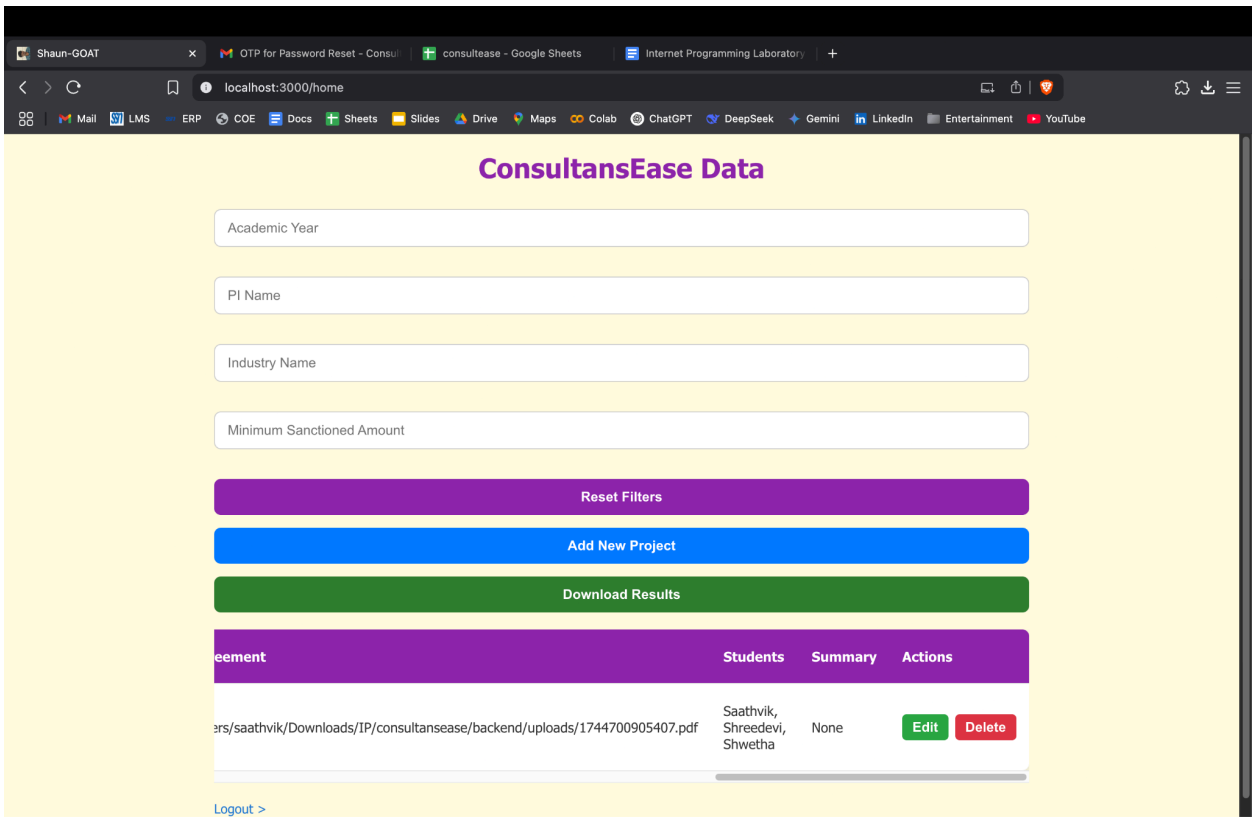




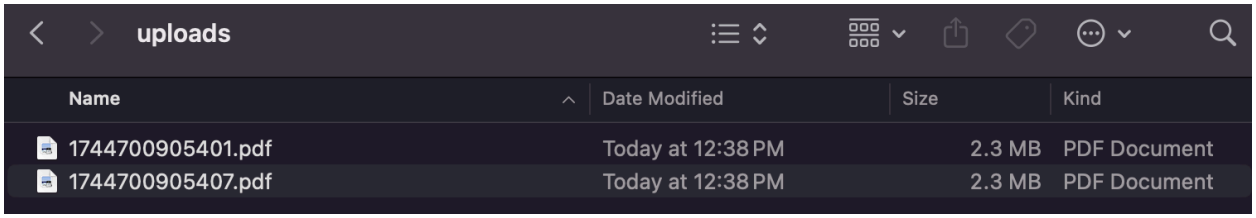
Delete Functionality:







Uploaded files are deleted as well:



LEARNING OUTCOME:

We have learnt to implement a complete full-stack web-application using the MERN stack and Google APIs.