SSN COLLEGE OF ENGINEERING

KALAVAKKAM - 603 110

Department of Computer Science and Engineering

UCS1601 Internet Programming Lab

Mini Project

Implementation of InternTrack Full Stack Web Application

TABLE OF CONTENTS

Question:

Students gain practical experience by doing internship in industries. It bridges the gap between academic knowledge and real-world applications. In the organization, there needs to be a tracking mechanism that maintains the details of the information related to internships.

The objective of *InternTrack* Full Stack Web Application is to have credentials for Student and Coordinator with necessary authorization. It has the following basic functionalities:

- Load the Details with the necessary proofs
- View the list of the internship
- Retrieve the information
- Edit the details

Build the *InterTrack* Full stack Web Application using MERN stack by implementing all the above-mentioned functionalities. (CO3, K6)

Description of *Retrieve* Functionality

Design a suitable back-end database (Mongo, Firebase or Google Drive) to maintain the uploaded documents.

Develop necessary end points to serve the requests for the following functionalities:

- List the students pursuing internship via CDC or through Department
- List the students who pursue internship in Abroad or India
- List the students based on individual company
- List of Students with Title & Domain:
- List of students with duration of Internship

Add a functionality to ensure maintaining the unique information in the backend about the details of the students across the usage of the form.

Add a functionality to avoid SQL injection vulnerability for the Application.

Description of Edit functionality

During Edit, the form needs to be visible to the user for the necessary updates.

INTRODUCTION:

The **InternTrack** Full Stack Web Application is designed to streamline internship record management by providing a secure and structured platform for both students and coordinators.

InternTrack ensures that internship details are properly loaded, viewed, retrieved, and edited with authorization-based access. The application integrates **Google Drive** for document storage and **Google Sheets** for data retrieval and validation.

Core Functionalities of InternTrack:

- **Loading Internship Details with Proof Documents:** Students can upload internship details along with offer letters and other supporting documents. These documents are validated using **OCR (Tesseract.js)** to extract and verify details.

- **Viewing Internship Records:** Users can access a list of students undergoing internships based on multiple criteria.

- **Retrieving Internship Information:** The system provides advanced filtering options to categorize students based on location (India/Abroad), company, internship title, and duration.

- **Editing Internship Details:** Students and coordinators can update records through an interactive form, ensuring accurate and up-to-date information.

Additionally, InternTrack ensures data integrity by maintaining unique student records and implementing **security measures** to prevent SQL injection vulnerabilities.

FUNCTIONALITIES:

1. Functionality to ensure maintaining unique information in the backend about the details of the students across the usage of the form:

A **login system with authentication** is used to authenticate and maintain unique information of the students across the usage of the form.

- Each student is uniquely identified by **a registration number or email**.
- The system **associates internship details with a specific user**, avoiding duplicate or conflicting entries.
- Users **cannot accidentally modify other students' records** because access is restricted.

2. Functionality to avoid SQL injection vulnerability for the Application:

a. **Indirect Protection Through Backend API**

Your login page sends the user's input to an API (https://script.google.com/macros/s/AKfycbyBnI7D_4jOouMy6Eq3HtMLrWww5lCPwDCh2xeAl7y5uQ6LCcOsMlQBQefQqcoIuYeJ/exec?action=search&registerNo=${password}), which then fetches data from Google Sheets.

If the backend API is secure and does not execute raw SQL queries, then SQL injection risks are minimized. To ensure complete protection, the backend should use parameterized queries or prepared statements instead of raw SQL.

b. Input Validation on the Frontend

The login page ensures that **both the username and register number fields are required**, preventing empty or malformed inputs from reaching the backend.

c. Limited User Input Fields

Since the form **only accepts a username and register number**, it restricts users from injecting malicious SQL commands. Unlike traditional SQL login forms (which accept raw SQL statements), **this form sends structured API requests**.

d. Secure API Handling on Failure

The login function already checks if the API response is valid:

```
if (jsonData.status !== "success" || !jsonData.data)
        throw new Error("Invalid API response");
```

This prevents **unexpected SQL injection attempts through manipulated responses**.

3. Functionality for loading and Validation using Python Tesseract Library:

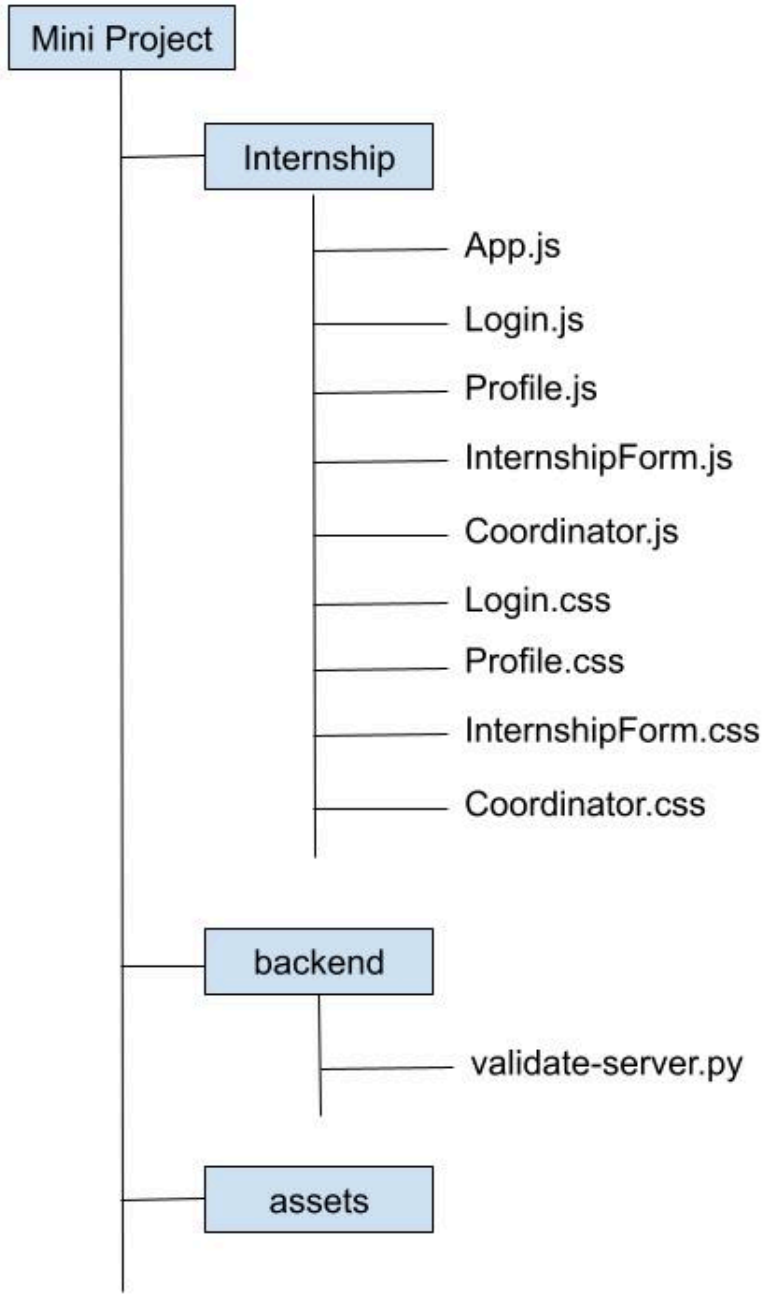
InternTrack incorporates the Tesseract OCR (Optical Character Recognition) library through a Python backend to ensure that uploaded offer letters are authentic and correctly filled. When a student submits their internship details through the form, they are required to upload a PDF of their offer letter. This document is sent to a Python-based validation server that uses the Tesseract OCR engine to extract textual data from the uploaded file. The extracted information, such as the student's name, registration number, company name, and internship duration, is then compared with the values entered in the form. This cross-verification ensures that students cannot submit incorrect or mismatched data, thereby improving the reliability and trustworthiness of the application.

The validation is performed in real-time, and only if the extracted details match the input values, the form is successfully submitted and the document is stored in the designated "Proof" folder on Google Drive. This adds an essential layer of security and verification to the system by automating what would otherwise be a manual, error-prone process.

IMPLEMENTATION:

- **Frontend:** React.js
- **Backend:** Python Tesseract for validating
- **External Services:** Google Drive API (document storage), Google Sheets API (data validation)

Folder Structure:



Files used:

- **Login.js** - Used by both student and coordinator for login purpose. Student login is verified using data available in the Google Sheets. Coordinator login consists of a username and password.
- **Profile.js** - If student logins using their name and register number, the details from the Google sheets are loaded and displayed in the page.
- **InternshipForm.js** - There is an Edit button in profile page, which will lead to a form, where the student can edit their details and on clicking submit button, the uploaded offer letter is first validated. Only if its correctly validated, the form is submitted and the offer letter will be uploaded in the Proof folder in the drive.
- **Coordinator.js** - It has 11 buttons, which act like various filters. Upon clicking each button, the corresponding filter is applied on the data in Google sheets and the data is displayed.
- **validate-server.py** - Its a python file with Tessaract Library function which is used to validate the contents in the offer letter with the ones that are entered in the form (InternshipForm.js). To start this "uvicorn validate-server:app --reload" command is used.
- **App.js** - contains the routes to all the pages

CODE:

App.js

```
JavaScript
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import Login from "./Login";
import Profile from "./Profile";
import Coordinator from "./Coordinator";
import InternshipForm from "./InternshipForm";

const App = () => {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/profile" element={<Profile />} />
        <Route path="/coordinator" element={<Coordinator />} />
        <Route path="/edit" element={<InternshipForm />} />
      </Routes>
    </Router>
```

```
  );
};

export default App;
```

Login page:

```
JavaScript
import { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import "./login.css";
import logo from './assets/logo.png';
import girlImage from './assets/girl.webp';

const Login = () => {
  const [isCoordinatorLogin, setIsCoordinatorLogin] = useState(false);
  const [loading, setLoading] = useState(true);
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [toastMessage, setToastMessage] = useState("");
  const [isToastSuccess, setIsToastSuccess] = useState(false);
  const navigate = useNavigate();

  useEffect(() => {
    setTimeout(() => {
      setLoading(false);
    }, 2000);
  }, []);

  const showToast = (message, isSuccess) => {
    setToastMessage(message);
    setIsToastSuccess(isSuccess);
    setTimeout(() => {
      setToastMessage("");
    }, 2000);
  };
```

```
  const handleLogin = async (event) => {
    event.preventDefault();

    if (isCoordinatorLogin) {
        if (username === "npr" && password === "npr@703") {
            showToast("Login successful! Redirecting...", true);
            setTimeout(() => {
                navigate("/coordinator");
            }, 1500);
        } else {
            showToast("Invalid username or password!", false);
        }
    } else {
        try {
            let response = await fetch(

`https://script.google.com/macros/s/AKfycbyBnI7D_4jOouMy6Eq3HtMLrWww5lCPwDCh2x
eAl7y5uQ6LCcOsMlQBQefQqcoIuYeJ/exec?action=search&registerNo=${password}`
            );

            let jsonData = await response.json();

            // Ensure the API response is correct
            if (jsonData.status !== "success" || !jsonData.data) {
                throw new Error("Invalid API response");
            }

            let userDetails = jsonData.data; // Array of values

            if (!userDetails || userDetails.length < 2) {
                showToast("Invalid Register Number!", false);
                return;
            }

            // Convert array to object for better usability
            let userObject = {
                registerNo: userDetails[0],
                name: userDetails[1],
                title: userDetails[2],
                mobileNo: userDetails[3],
                section: userDetails[4],
                obtainedInternship: userDetails[5],
                period: userDetails[6],
```

```
                startDate: userDetails[7],
                endDate: userDetails[8],
                company: userDetails[9],
                placementThrough: userDetails[10],
                stipend: userDetails[11],
                researchOrIndustry: userDetails[12],
                abroadOrIndia: userDetails[13],
                fileUrl: userDetails[14]
            };

            console.log("Login", userObject);

            // Check if username matches
            if (userObject.name === username) {
                showToast("Login successful! Redirecting...", true);
                setTimeout(() => {
                    navigate("/profile", { state: { user: userObject } }); //
Passing user details
                }, 1500);
            } else {
                showToast("Invalid Name or Register Number!", false);
            }

        } catch (error) {
            console.error("Error fetching student data:", error);
            showToast("Server error, please try again!", false);
        }
    }
};


  return (
    <div>
      {loading ? (
        <div className="loading-screen">
          <img src={logo} alt="InternTrack Logo" style={{ width: "600px" }} />
        </div>
      ) : (
        <div className="f">
          <img src={girlImage} alt="Girl Illustration" style={{ width:
"200px", height: "300px" }} />
          <div className="container">
```

```
          <img src={logo} alt="InternTrack Logo" style={{ height: "100px",
width: "auto" }}/>
            <div className="tabs">
              <span className={`tab ${!isCoordinatorLogin ? "active" : ""}`}
onClick={() => setIsCoordinatorLogin(false)}>
                Student Login
              </span>
              <span className={`tab ${isCoordinatorLogin ? "active" : ""}`}
onClick={() => setIsCoordinatorLogin(true)}>
                Coordinator Login
              </span>
            </div>
            <form onSubmit={handleLogin}>
              <input type="text" placeholder="Name" required value={username}
onChange={(e) => setUsername(e.target.value)} />
              <input type="password" placeholder="Register Number" required
value={password} onChange={(e) => setPassword(e.target.value)} />
              <button type="submit" className="login-btn">Login</button>
            </form>
          </div>
        </div>
      )}
      {toastMessage && (
        <div className={`toast show ${isToastSuccess ? "success" :
"error"}`}>{toastMessage}</div>
      )}
    </div>
  );
};

export default Login;
```

CSS:

Unset

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}
```

```
body {
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-color: white;
    overflow: hidden; /* Prevent scrollbars during animation */
}

/* Loading Screen Styles */
.loading-screen {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: white;
    display: flex;
    align-items: center;
    justify-content: center;
    z-index: 1000;
    transition: opacity 0.5s ease;
}

.loading-screen img {
    width: 200px;
    height: auto;
}

.f {
    display: flex;
    align-items: center; /* Aligns items (image + container) vertically */
    justify-content: center; /* Centers them horizontally */
    gap: 20px;
    width: 100%; /* Ensure it spans full width */
    height: 100vh; /* Full viewport height */
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    opacity: 1;
    transition: opacity 0.5s ease;
```

```
}

/* Animation for the girl image (slides in from the left) */
.f img {
    height: 300px;
    width: auto;
    transform: translateX(-100%); /* Start off-screen to the left */
    animation: slideInLeft 1s ease-out forwards; /* Animation */
}

@keyframes slideInLeft {
    to {
        transform: translateX(0); /* Move to original position */
    }
}

/* Animation for the container (slides in from the right) */
.container {
    width: 500px;
    height: 400px;
    padding: 30px;
    background: #f4f4f4;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    text-align: center;
    transform: translateX(100%); /* Start off-screen to the right */
    animation: slideInRight 1s ease-out forwards; /* Animation */
}

@keyframes slideInRight {
    to {
        transform: translateX(0); /* Move to original position */
    }
}

.container img {
    width: 200px;
    margin-bottom: 20px;
}

.tabs {
    display: flex;
    justify-content: center;
```

```
    gap: 20px;
    margin-bottom: 20px;
}

.tab {
    font-weight: bold;
    cursor: pointer;
    padding: 10px;
    border-bottom: 3px solid transparent;
}

.tab.active {
    border-bottom: 3px solid #003b5b;
}

input {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
}

.login-btn {
    width: 100%;
    background: #2a2828;
    color: white;
    padding: 12px;
    border: none;
    border-radius: 5px;
    font-size: 18px;
    cursor: pointer;
    transition: background 0.3s;
}

.login-btn:hover {
    background:blue;
}

/* Toast Notification Styles */
.toast {
    position: fixed;
```

```
    bottom: 10%;
    left: 50%;
    transform: translateX(-50%);
    background-color: #333;
    color: white;
    padding: 12px 24px;
    border-radius: 5px;
    font-size: 16px;
    z-index: 1000;
    opacity: 0;
    visibility: hidden;
    transition: opacity 0.3s ease, visibility 0.3s ease;
}

.toast.show {
    opacity: 1;
    visibility: visible;
}

.toast.success {
    background-color: black;
}

.toast.error {
    background-color: #a03741;
}
```

Profile page:

```
JavaScript
import React, { useEffect, useState } from "react";
import { useLocation } from "react-router-dom";
import "./profile.css";
import boy from './assets/boy.webp';
import userImage from './assets/User.png';
import FormComponent from "./InternshipForm"; // Import the form
import { useNavigate } from "react-router-dom";
```

```
const Profile = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const [profileData, setProfileData] = useState(null);
  const [isProfileOpen, setIsProfileOpen] = useState(false);
  const [isEditing, setIsEditing] = useState(false);

  useEffect(() => {
    if (location.state && location.state.user) {
      setProfileData(location.state.user);
    }
  }, [location]);


  const handleEditClick = () => {
    navigate("/edit", { state: { user: profileData } }); // Navigate with data
  };

  const handleCloseForm = () => {
    setIsEditing(false); // Close form and return to profile
  };

  return (
    <div className="top">
      <img src={boy} alt="Boy" className="animated-img" />
      <div className={`container ${isProfileOpen ? "slide-left" : ""}`}>

        {!isEditing ? (
          <>
            <div className="profile-card">
              <img src={userImage} alt="Profile"/>
              <h1>{profileData ? profileData.name : "Loading..."}</h1>
              <p>{profileData ? `Reg No: ${profileData.registerNo}` :
"Loading..."}</p>
              <div className="menu">
                <button onClick={() => setIsProfileOpen(true)}>My
Profile</button>
              </div>
            </div>

            {isProfileOpen && profileData && (
              <div className="profile-details">
```

```
                <span className="close-btn" onClick={() =>
setIsProfileOpen(false)}>&times;</span>
                <h2>Profile Details</h2>

                <div className="details-container">
                  <div className="detail"><strong>Register Number</strong>
<span>{profileData.registerNo}</span></div>
                  <div className="detail"><strong>Name</strong>
<span>{profileData.name}</span></div>
                  <div className="detail"><strong>Title</strong>
<span>{profileData.title}</span></div>
                  <div className="detail"><strong>Mobile No.</strong>
<span>{profileData.mobileNo}</span></div>
                  <div className="detail"><strong>Section</strong>
<span>{profileData.section}</span></div>
                  <div className="detail"><strong>Internship Status</strong>
<span>{profileData.obtainedInternship}</span></div>
                  <div className="detail"><strong>Period</strong>
<span>{profileData.period}</span></div>
                  <div className="detail"><strong>Start Date</strong>
<span>{profileData.startDate}</span></div>
                  <div className="detail"><strong>End Date</strong>
<span>{profileData.endDate}</span></div>
                  <div className="detail"><strong>Company Name</strong>
<span>{profileData.company}</span></div>
                  <div className="detail"><strong>Placement Through</strong>
<span>{profileData.placementThrough}</span></div>
                  <div className="detail"><strong>Stipend</strong>
<span>{profileData.stipend}</span></div>
                  <div className="detail"><strong>Research or
Industry</strong> <span>{profileData.researchOrIndustry}</span></div>
                  <div className="detail"><strong>Abroad or India</strong>
<span>{profileData.abroadOrIndia}</span></div>
                  {profileData.fileUrl && (
                    <div className="detail">
                      <strong>File</strong>
                      <a href={profileData.fileUrl} target="_blank"
rel="noopener noreferrer">View Document</a>
                    </div>
                  )}
                </div>

                {/* Edit Button to Open Form */}
```

```
                <div className="edit-button-container">
                  <button
                    onClick={handleEditClick}
                    className="edit-btn">
                    Edit
                  </button>
                </div>
              </div>
            )}
          </>
        ) : (
          // Render FormComponent when Editing
          <FormComponent initialData={profileData} onClose={handleCloseForm}
/>
        )}
      </div>
    </div>
  );
};

export default Profile;
```

CSS:

```
Unset
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

body {
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-color: #f4f4f4;
    overflow: hidden;
}
```

```
.top {
    display: flex;
    align-items: center;
    gap: 40px;
}

.top img {
    height: 350px;
    transform: translateX(-100%);
    animation: slideInLeft 1s ease-out forwards;
}

@keyframes slideInLeft {
    to {
        transform: translateX(0);
    }
}

.container {
    width: 450px;
    background: white;
    border-radius: 20px;
    border: 1px solid black;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    position: relative;
    transform: translateX(100%);
    animation: slideInRight 1s ease-out forwards;
}

@keyframes slideInRight {
    to {
        transform: translateX(0);
    }
}

.profile-card {
    width: 100%;
    padding: 30px;
    text-align: center;
    transition: transform 0.5s ease-in-out, opacity 0.3s ease-in-out;
}
```

```
.profile-card img {
    width: 120px;
    height: 120px;
    border-radius: 50%;
    margin-bottom: 20px;
}

.profile-card h1 {
    font-size: 24px;
    margin-bottom: 10px;
    color: #2a2828;
}

.profile-card p {
    font-size: 16px;
    color: #666;
    margin-bottom: 20px;
}

.menu {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.menu button {
    padding: 12px;
    border: none;
    border-radius: 10px;
    background: #2a2828;
    color: white;
    font-size: 16px;
    cursor: pointer;
    transition: background 0.3s, transform 0.3s;
}

.menu button:hover {
    background: black;
    transform: translateY(-3px);
}

.profile-details {
    width: 100%;
```

```
    height: 100vh;
    padding: 30px;
    background: white;
    position: absolute;
    top: 0;
    left: 100%;
    transition: transform 0.5s ease-in-out;
}

.profile-details h2 {
    font-size: 22px;
    color: #2a2828;
    margin-bottom: 20px;
}

.details-container {
    display: flex;
    flex-direction: column;
    gap: 15px;
}

.detail {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
    background: #f9f9f9;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.detail strong {
    color: #2a2828;
}

.detail span {
    color: #666;
}

.container.slide-left .profile-card {
    transform: translateX(-100%);
    opacity: 0;
}
```

```
.container.slide-left .profile-details {
    transform: translateX(-100%);
}

.close-btn {
    position: absolute;
    top: 20px;
    right: 20px;
    font-size: 24px;
    cursor: pointer;
    color: #2a2828;
    transition: color 0.3s;
}

.close-btn:hover {
    color: black;
}

.profile-details {
    max-height: 50vh; /* Increase max height */
  overflow-y: auto; /* Enable scrolling */
  padding: 15px; /* Add some padding */
    border-radius: 10px;
    background: white;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  }

  .details-container {
    display: flex;
    flex-direction: column;
    gap: 10px;
  }
```

Student edit page:

JavaScript

```
import React, { useState, useEffect } from 'react';
```

```
import "./InternshipForm.css";
import { useLocation } from "react-router-dom";

const InternshipForm = () => {
    const location = useLocation();
    const initialData = location.state?.user || {};
    const scriptURL =
'https://script.google.com/macros/s/AKfycbwv8wus6zQYuCp60DZ-DvUWWeSCgqvyRyJf9s
XNDwGHVYYQkjyXIYW4QJKOo7U7cxJA/exec';
    const validationURL = 'http://localhost:8000/validate/';

    const [formData, setFormData] = useState({
      registerNo: '', name: '', mobileNo: '', section: '', obtainedInternship:
'', title: '',
      period: '', startDate: '', endDate: '', company: '', placementThrough:
'Through College',
      stipend: '', researchOrIndustry: 'Research', abroadOrIndia: 'India',
offerLetter: null,
      ...initialData  // ✅ Pre-fill form with existing user data
  });

    const [validationMessage, setValidationMessage] = useState('');
    const [loading, setLoading] = useState(false);

    useEffect(() => {
        if (initialData) {
          setFormData(initialData);
        }
      }, [initialData]);

    useEffect(() => {
      if (formData?.registerNo && formData.registerNo.length === 13) {
            fetch(`${scriptURL}?registerNo=${formData.registerNo}`)
                .then(response => response.json())
                .then(data => {
                    if (data.status === 'success' && data.rows.length > 0) {
                        const values = data.rows[0];
                        setFormData(prev => ({
                            ...prev,
                            name: values[1] || '',
                            mobileNo: values[3] || '',
                            section: values[4] || '',
                            obtainedInternship: values[5] || '',
```

```
                            title: values[2] || '',
                            period: values[6] || '',
                            startDate: values[7] || '',
                            endDate: values[8] || '',
                            company: values[9] || '',
                            placementThrough: values[10] || 'Through College',
                            stipend: values[11] || '',
                            researchOrIndustry: values[12] || 'Research',
                            abroadOrIndia: values[13] || 'India'
                        }));
                    }
                })
                .catch(error => console.error('Error fetching data:', error));
        }
    }, [formData.registerNo]);

    const handleChange = (e) => {
        const { name, type, files, value } = e.target;

        setFormData((prev) => ({
            ...prev,
            [name]: type === 'file' ? files[0] : value
        }));
    };



    const handleSubmit = async (e) => {
        console.log("Form Data Before Submit:", formData);

        e.preventDefault();
        setLoading(true);

        if (formData.offerLetter && formData.offerLetter.size > 10 * 1024 *
1024) {
            alert('File size must be less than 10MB');
            setLoading(false);
            return;
        }

        try {
            const validateFormData = new FormData();
            validateFormData.append("register_number", formData.registerNo);
```

```
            validateFormData.append("name", formData.name);
            validateFormData.append("mobile_number", formData.mobileNo);
            validateFormData.append("internship", formData.title);
            validateFormData.append("internship_obtained",
formData.obtainedInternship);
            validateFormData.append("internship_place",
formData.abroadOrIndia);
            validateFormData.append("start_date", formData.startDate);
            validateFormData.append("end_date", formData.endDate);
            validateFormData.append("company_name", formData.company);
            validateFormData.append("stipend", formData.stipend || "");
            validateFormData.append("permission_letter",
formData.offerLetter); // FILE UPLOAD

            const validationResponse = await fetch(validationURL, {
                method: "POST",
                body: validateFormData
            });


            const validationResult = await validationResponse.json();
            setValidationMessage(validationResult.message);
            if (validationResult.status === 'error') {
                setLoading(false);
                return;
            }

            const googleFormData = new FormData();
            Object.entries(formData).forEach(([key, value]) => {
                googleFormData.append(key, value);
            });

            // Convert File to Base64 ONLY when sending to Google Sheets
            if (formData.offerLetter) {
                const base64File = await new Promise((resolve) => {
                    const reader = new FileReader();
                    reader.onloadend = () =>
resolve(reader.result.split(',')[1]);  // Remove data URL prefix
                    reader.readAsDataURL(formData.offerLetter);
                });

                googleFormData.append('fileName', formData.offerLetter.name);
                googleFormData.append('mimeType', formData.offerLetter.type);
```

```
                googleFormData.append('fileData', base64File);
            }

            // Debugging: Check if fileData is correctly added
            console.log("Sending Google FormData:");
            for (let pair of googleFormData.entries()) {
                console.log(pair[0], pair[1]);
            }

            const response = await fetch(scriptURL, {
                method: 'POST',
                body: googleFormData
            });



            const result = await response.json();
            setLoading(false);
            if (result.status === 'success') {
                alert('Form submitted successfully!');
                setFormData({
                    registerNo: '', name: '', mobileNo: '', section: '',
obtainedInternship: '', title: '',
                    period: '', startDate: '', endDate: '', company: '',
placementThrough: 'Through College',
                    stipend: '', researchOrIndustry: 'Research',
abroadOrIndia: 'India', offerLetter: null
                });
                setValidationMessage('');
            } else {
                alert('Submission failed');
            }
        } catch (error) {
            console.error('Error:', error);
            setLoading(false);
            alert('Error submitting form');
        }
    };

    return (
        <div className="container" style={{ maxHeight: '80vh', overflowY:
'auto', padding: '20px', border: '1px solid #ccc' }}>
          <h2>Internship Details Form</h2>
```

```
          <form onSubmit={handleSubmit}>
            <div className="form-group">
              <label>Register Number*:</label>
              <input type="text" name="registerNo" value={formData.registerNo}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>Name*:</label>
              <input type="text" name="name" value={formData.name}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>Mobile Number*:</label>
              <input type="text" name="mobileNo" value={formData.mobileNo}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>Section*:</label>
              <input type="text" name="section" value={formData.section}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>Obtained Internship*:</label>
              <div className="radio-group">
                <label>
                  <input type="radio" name="obtainedInternship" value="Yes"
                    checked={formData.obtainedInternship === "Yes"}
onChange={handleChange} required/>
                Yes</label>
                <label>
                  <input type="radio" name="obtainedInternship" value="No"
                    checked={formData.obtainedInternship === "No"}
onChange={handleChange}/>
                No</label>
              </div>
            </div>
            <div className="form-group">
              <label>Title*:</label>
              <input type="text" name="title" value={formData.title}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>Start Date*:</label>
```

```
              <input type="date" name="startDate" value={formData.startDate}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>End Date*:</label>
              <input type="date" name="endDate" value={formData.endDate}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>Company Name*:</label>
              <input type="text" name="company" value={formData.company}
onChange={handleChange} required />
            </div>
            <div className="form-group">
              <label>Placement through*:</label>
              <select name="placementThrough"
value={formData.placementThrough} onChange={handleChange} required>
                <option value="Through College">Through College</option>
                <option value="Off-Campus">Off-Campus</option>
              </select>
            </div>
            <div className="form-group">
              <label>Stipend (in Rs.):</label>
              <input type="number" name="stipend" value={formData.stipend}
onChange={handleChange} min="0" />
            </div>
            <div className="form-group">
              <label>Upload Offer Letter (PDF)*:</label>
              <input type="file" name="offerLetter" accept="application/pdf"
onChange={handleChange} required />
            </div>
            {validationMessage && <p>{validationMessage}</p>}
            {loading && <p>Validating document... Please wait.</p>}
            <button type="submit">Submit</button>
          </form>
        </div>
      );
};

export default InternshipForm;
```

CSS:

Unset

```
/* Container Styling */
.container {
    width: 95%;
    max-width: 800px; /* Prevents it from being too wide */
    margin: 20px auto;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    font-family: Arial, sans-serif;
    height: auto; /* Adjust height dynamically */
    overflow-y: hidden; /* Prevent scrolling */
}

/* Heading */
h2 {
    text-align: center;
    color: #333;
    margin-bottom: 20px;
}

/* Form Styling */
form {
    display: flex;
    flex-direction: column;
    gap: 15px; /* Add spacing between elements */
}

/* Form Group */
.form-group {
    display: flex;
    align-items: center;
    justify-content: space-between;
}

/* Labels */
.form-group label {
    width: 30%;
    font-size: 14px;
    font-weight: bold;
    color: #333;
}
```

```
/* Input Fields */
.form-group input,
.form-group select {
    width: 65%;
    padding: 10px;
    font-size: 14px;
    border: 1px solid #ccc;
    border-radius: 5px;
    transition: border 0.3s ease-in-out;
}

/* Radio Button Styling */
.form-group label input[type="radio"] {
    margin-right: 5px;
}

/* Hover & Focus Effects */
.form-group input:focus,
.form-group select:focus {
    outline: none;
    border-color: #007bff;
}

/* File Input */
.form-group input[type="file"] {
    border: none;
}
.radio-group {
    display: flex;
    gap: 20px; /* Adjust spacing */
    align-items: center;
  }

  .radio-group label {
    display: flex;
    align-items: center;
    gap: 5px;
    font-size: 16px;
    font-weight: 500;
    color: #333;
    cursor: pointer;
  }
```

```
  .radio-group input[type="radio"] {
    width: 18px;
    height: 18px;
    cursor: pointer;
    accent-color: #007bff; /* Make radio buttons blue */
    margin: 0;
    padding: 0;
  }



/* Submit Button */
button[type="submit"] {
    width: 100%;
    padding: 12px;
    font-size: 16px;
    font-weight: bold;
    color: white;
    background-color: black;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background 0.3s ease-in-out;
}

button[type="submit"]:hover {
    background-color: #0056b3;
    transform: scale(1.02);
}

/* Validation Message */
p {
    text-align: center;
    font-size: 14px;
    color: red;
}

/* Responsive Design */
@media (max-width: 768px) {
    .container {
        width: 98%;
        padding: 15px;
    }
```

```
    .form-group {
        flex-direction: column;
        align-items: flex-start;
        gap: 8px;
    }

    .form-group label {
        width: 100%;
    }

    .form-group input,
    .form-group select {
        width: 100%;
    }
}
```

Validating input data : validate-server.py

Python

```python
from fastapi import FastAPI, Form, UploadFile, File, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from typing import Optional
import pytesseract
from pdf2image import convert_from_bytes
import re
from dateparser import parse
import io

app = FastAPI()
pytesseract.pytesseract.tesseract_cmd = "/opt/homebrew/bin/tesseract"

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Allow all origins (You can restrict it later)
    allow_credentials=True,
    allow_methods=["*"],  # Allow all methods (GET, POST, etc.)
    allow_headers=["*"],  # Allow all headers
)

def extract_text_from_pdf(pdf_file: bytes):
```

```
    """Extract text from a PDF file using OCR."""
    try:
        images = convert_from_bytes(pdf_file,
poppler_path="/opt/homebrew/bin")
        extracted_text = ""

        for image in images:
            text = pytesseract.image_to_string(image)
            extracted_text += text + "\n"

        return extracted_text
    except Exception as e:
        print(f"Error extracting text from PDF: {str(e)}")
        return ""

def extract_dates(text):
    """Extract and normalize dates from OCR text."""
    date_patterns = [
        r"\b\d{1,2} [A-Za-z]+ \d{4}\b",  # Example: "16 June 2025"
        r"\b[A-Za-z]+ \d{1,2},? \d{4}\b", # Example: "June 16, 2025"
        r"\b\d{1,2}/\d{1,2}/\d{4}\b",    # Example: "16/06/2025"
        r"\b\d{4}-\d{2}-\d{2}\b"         # Example: "2025-10-06"
    ]

    found_dates = []

    for pattern in date_patterns:
        matches = re.findall(pattern, text)
        for match in matches:
            parsed_date = parse(match)  # Convert any format to datetime
object
            if parsed_date:
                found_dates.append(parsed_date.strftime("%Y-%m-%d"))  #
Convert to "YYYY-MM-DD"

    return found_dates

@app.post("/validate/")
async def validate_document(
    register_number: str = Form(...),
    name: str = Form(...),
    mobile_number: str = Form(...),
    internship: str = Form(...),
```

```
    internship_obtained: str = Form(...),
    internship_place: str = Form(...),
    start_date: Optional[str] = Form(None),
    end_date: Optional[str] = Form(None),
    company_name: Optional[str] = Form(None),
    stipend: Optional[int] = Form(None),
    permission_letter: UploadFile = File(...)
):
    try:
        # Extract PDF text
        pdf_bytes = await permission_letter.read()
        extracted_text = extract_text_from_pdf(pdf_bytes)

        if not extracted_text:
            return {"status": "error", "message": "Could not extract text from
the uploaded PDF file"}

        # Extract dates from OCR text
        extracted_dates = extract_dates(extracted_text)

        # Validate extracted text against form inputs
        validation_errors = []
        if name and name.lower() not in extracted_text.lower():
            validation_errors.append("Name in the form doesn't match what's in
the document")

        if company_name and company_name.lower() not in
extracted_text.lower():
            validation_errors.append("Company name in the form doesn't match
what's in the document")

        # More lenient date matching - check if any extracted date is close to
the provided dates
        date_matched = False
        if start_date:
            start_date_obj = parse(start_date)
            for extracted_date in extracted_dates:
                extracted_date_obj = parse(extracted_date)
                if extracted_date_obj and start_date_obj:
                    # Check if dates are within 2 days of each other
                    if abs((extracted_date_obj - start_date_obj).days) <= 2:
                        date_matched = True
                        break
```

```
            if not date_matched and extracted_dates:
                validation_errors.append(f"Start date {start_date} not found
in the document")

        # Validate end date if provided
        date_matched = False
        if end_date:
            end_date_obj = parse(end_date)
            for extracted_date in extracted_dates:
                extracted_date_obj = parse(extracted_date)
                if extracted_date_obj and end_date_obj:
                    # Check if dates are within 2 days of each other
                    if abs((extracted_date_obj - end_date_obj).days) <= 2:
                        date_matched = True
                        break

            if not date_matched and extracted_dates:
                validation_errors.append(f"End date {end_date} not found in
the document")

        # Check if internship terms are mentioned
        internship_terms = ["intern", "internship", "training", "trainee"]
        has_internship_term = any(term in extracted_text.lower() for term in
internship_terms)
        if not has_internship_term:
            validation_errors.append("Document doesn't appear to be an
internship offer letter")

        if validation_errors:
            return {
                "status": "error",
                "message": "Document validation failed: " + ",
".join(validation_errors),
                "errors": validation_errors
            }

        return {"status": "success", "message": "Document validated
successfully!"}

    except Exception as e:
        return {"status": "error", "message": f"Validation error: {str(e)}"}
```

```
# Add a simple health check endpoint
@app.get("/")
def read_root():
    return {"status": "online", "message": "Validation service is running"}
```

Coordinator page:

Coordinator.js

```
JavaScript
import React, { useState } from "react";
import "./coordinator.css";
import document from './assets/document.png';

const SHEET_API_URL =
"https://script.google.com/macros/s/AKfycbzUUFKAP0cDxUBOnoiyJFvZHksY3DXOMM9sOU
rbuclGDfEi7QhfacPGLkB38EE5SRa6/exec";

const CoordinatorPage = () => {
  const [data, setData] = useState(null);
  const [isModalOpen, setIsModalOpen] = useState(false);

  const fetchData = async () => {
    try {
      const response = await fetch(SHEET_API_URL);
      const jsonData = await response.json();
      setData(jsonData);
      setIsModalOpen(true);
    } catch (error) {
      console.error("Error fetching data:", error);
      setData([]);
    }
  };

  const CDC = async () => {
    try {
      const response = await fetch(SHEET_API_URL);
      const jsonData = await response.json();
      displayCDC(jsonData);
```

```
    } catch (error) {
      console.error("Error fetching CDC data:", error);
      setData([]);
    }
  };

  const displayCDC = (data) => {
    if (!data || data.length === 0) {
      setData([]);
      return;
    }

    let headers = data[0];
    let records = data.slice(1);
    let internshipTypeIndex = headers.indexOf("placement-source");

    if (internshipTypeIndex === -1) {
      setData([]);
      return;
    }

    let filteredRecords = records.filter(row => row[internshipTypeIndex] ===
"Through College");

    if (filteredRecords.length === 0) {
      setData([]);
      return;
    }

    setData([headers, ...filteredRecords]);
    setIsModalOpen(true);
  };

  const noncdc = async () => {
    try {
      const response = await fetch(SHEET_API_URL);
      const jsonData = await response.json();
      displayNonCDC(jsonData);
    } catch (error) {
      console.error("Error fetching Non-CDC data:", error);
      setData([]);
    }
  };
```

```
  const displayNonCDC = (data) => {
    if (!data || data.length === 0) {
      setData([]);
      return;
    }

    let headers = data[0];
    let records = data.slice(1);
    let internshipTypeIndex = headers.indexOf("placement-source");

    if (internshipTypeIndex === -1) {
      setData([]);
      return;
    }

    let filteredRecords = records.filter(row => row[internshipTypeIndex] ===
"Off-Campus");

    if (filteredRecords.length === 0) {
      setData([]);
      return;
    }

    setData([headers, ...filteredRecords]);
    setIsModalOpen(true);
  };

  const noInternship = async () => {
    try {
      const response = await fetch(SHEET_API_URL);
      const jsonData = await response.json();
      displayNoInternship(jsonData);
    } catch (error) {
      console.error("Error fetching No Internship data:", error);
      setData([]);
    }
  };

  const displayNoInternship = (data) => {
    if (!data || data.length === 0) {
      setData([]);
      return;
```

```
    }

    let headers = data[0];
    let records = data.slice(1);
    let internshipTypeIndex = headers.indexOf("obtained-internship");

    if (internshipTypeIndex === -1) {
      setData([]);
      return;
    }

    let filteredRecords = records.filter(row => row[internshipTypeIndex] ===
"No");

    if (filteredRecords.length === 0) {
      setData([]);
      return;
    }

    setData([headers, ...filteredRecords]);
    setIsModalOpen(true);
  };

  const applyFilter = async (filterType) => {
    try {
      const response = await fetch(SHEET_API_URL);
      const jsonData = await response.json();
      filterData(jsonData, filterType);
    } catch (error) {
      console.error("Error fetching filtered data:", error);
      setData([]);
    }
  };

  const filterData = (data, filterType) => {
    if (!data || data.length === 0) {
      setData([]);
      return;
    }

    let headers = data[0];
    let records = data.slice(1);
    let filterIndex = -1;
```

```
    let filterValue = "";

    switch (filterType) {
      case 'research':
        filterIndex = headers.indexOf("internship-type");
        filterValue = "Research";
        break;
      case 'industry':
        filterIndex = headers.indexOf("internship-type");
        filterValue = "Industrial";
        break;
      case 'abroad':
        filterIndex = headers.indexOf("location");
        filterValue = "Abroad";
        break;
      case 'india':
        filterIndex = headers.indexOf("location");
        filterValue = "India";
        break;
      default:
        setData([]);
        return;
    }

    if (filterIndex === -1) {
      setData([]);
      return;
    }

    let filteredRecords = records.filter(row => row[filterIndex] ===
filterValue);

    if (filteredRecords.length === 0) {
      setData([]);
      return;
    }

    setData([headers, ...filteredRecords]);
    setIsModalOpen(true);
  };

  const filterByPeriod = async () => {
    try {
```

```
      const response = await fetch(SHEET_API_URL);
      const jsonData = await response.json();
      if (!jsonData || jsonData.length === 0) {
        setData([]);
        return;
      }

      let headers = jsonData[0];
      let records = jsonData.slice(1);
      let periodIndex = headers.indexOf("period");

      if (periodIndex === -1) {
        setData([]);
        return;
      }

      let grouped = { "1 Month": [], "2 Months": [], "6 Months": [] };
      records.forEach(row => {
        let duration = parseInt(row[periodIndex]);
        if (duration === 1) grouped["1 Month"].push(row);
        else if (duration === 2) grouped["2 Months"].push(row);
        else if (duration === 6) grouped["6 Months"].push(row);
      });

      let groupedData = [];
      for (let period in grouped) {
        if (grouped[period].length > 0) {
          groupedData.push([period]);
          groupedData.push(...grouped[period]);
        }
      }

      setData([headers, ...groupedData]);
      setIsModalOpen(true);
    } catch (error) {
      console.error("Error fetching period data:", error);
      setData([]);
    }
  };

  const filterByCompany = async () => {
    try {
      const response = await fetch(SHEET_API_URL);
```

```
      const jsonData = await response.json();
      if (!jsonData || jsonData.length === 0) {
        setData([]);
        return;
      }

      let headers = jsonData[0];
      let records = jsonData.slice(1);
      let companyIndex = headers.indexOf("company-name");

      if (companyIndex === -1) {
        setData([]);
        return;
      }

      let grouped = {};
      records.forEach(row => {
        let company = row[companyIndex];
        if (!company) return;
        if (!grouped[company]) grouped[company] = [];
        grouped[company].push(row);
      });

      let groupedData = [];
      for (let company in grouped) {
        groupedData.push([company]);
        groupedData.push(...grouped[company]);
      }

      setData([headers, ...groupedData]);
      setIsModalOpen(true);
    } catch (error) {
      console.error("Error fetching company data:", error);
      setData([]);
    }
  };

  const filterByStipend = async () => {
    try {
      const response = await fetch(SHEET_API_URL);
      const jsonData = await response.json();
      if (!jsonData || jsonData.length === 0) {
        setData([]);
```

```
        return;
      }

      let headers = jsonData[0];
      let records = jsonData.slice(1);
      let stipendIndex = headers.indexOf("stipend");

      if (stipendIndex === -1) {
        setData([]);
        return;
      }

      let filteredRecords = records.filter(row => {
        if (!row[stipendIndex]) return false;
        let stipendValue = String(row[stipendIndex]).replace(/[^0-9.]/g, "");
        let stipend = parseFloat(stipendValue) || 0;
        return stipend > 100000;
      });

      setData([headers, ...filteredRecords]);
      setIsModalOpen(true);
    } catch (error) {
      console.error("Error fetching stipend data:", error);
      setData([]);
    }
  };

  const closeModal = () => {
    setIsModalOpen(false);
  };

  const renderTable = () => {
    if (!data || data.length === 0) {
      return <p>No data available.</p>;
    }

    return (
      <table border="1" cellSpacing="0" cellPadding="5">
        <thead>
          <tr>
            {data[0].map((header, index) => (
              <th key={index}>{header}</th>
            ))}
```

```
          </tr>
        </thead>
        <tbody>
          {data.slice(1).map((row, rowIndex) => (
            <tr key={rowIndex}>
              {row.map((cell, cellIndex) => (
                <td key={cellIndex}>{cell}</td>
              ))}
            </tr>
          ))}
        </tbody>
      </table>
    );
  };

  return (
    <div>
      <div className="top">
      <div className="container">
        <h2>INTERNSHIP RECORDS</h2><br></br><br></br>

        <div className="filter-buttons">
            <div className="column">
                <button onClick={CDC}>Students Got Internship Through
CDC</button>
                <button onClick={noncdc}>Students Got Internship Not Through
CDC</button>
                <button onClick={noInternship}>Students who didn't get
internship</button>
                <button onClick={() => applyFilter('research')}>Research
Internships</button>
                <button onClick={() => applyFilter('industry')}>Industrial
Internships</button>
            </div>
            <div className="column">
                <button onClick={filterByPeriod}>Internships by
Period</button>
                <button onClick={filterByCompany}>Company-wise List</button>
                <button onClick={filterByStipend}>Stipend More Than 1
Lakh</button>
                <button onClick={() => applyFilter('abroad')}>Internships
Completed Abroad</button>
```

```
                <button onClick={() => applyFilter('india')}>Internships
Completed in India</button>
            </div>
        </div>

        {/* Centered Display All Records button */}
        <div className="centered-button">
            <button onClick={fetchData}>
                <strong>Display All Records</strong>
            </button>
        </div>
    </div>

    <img src={document} alt="Document" />
      </div>

      {isModalOpen && (
        <div className="modal">
          <span className="close" onClick={closeModal}>
            &times;
          </span>
          <h3>Internship Records</h3>
          <div id="jsonData">{renderTable()}</div>
        </div>
      )}
    </div>
  );
};

export default CoordinatorPage;
```

CSS:

```
Unset
#jsonData {
    width: 100%;
    height: 400px;
    overflow: auto;
    border: 1px solid #ccc;
    padding: 10px;
    background-color: white;
```

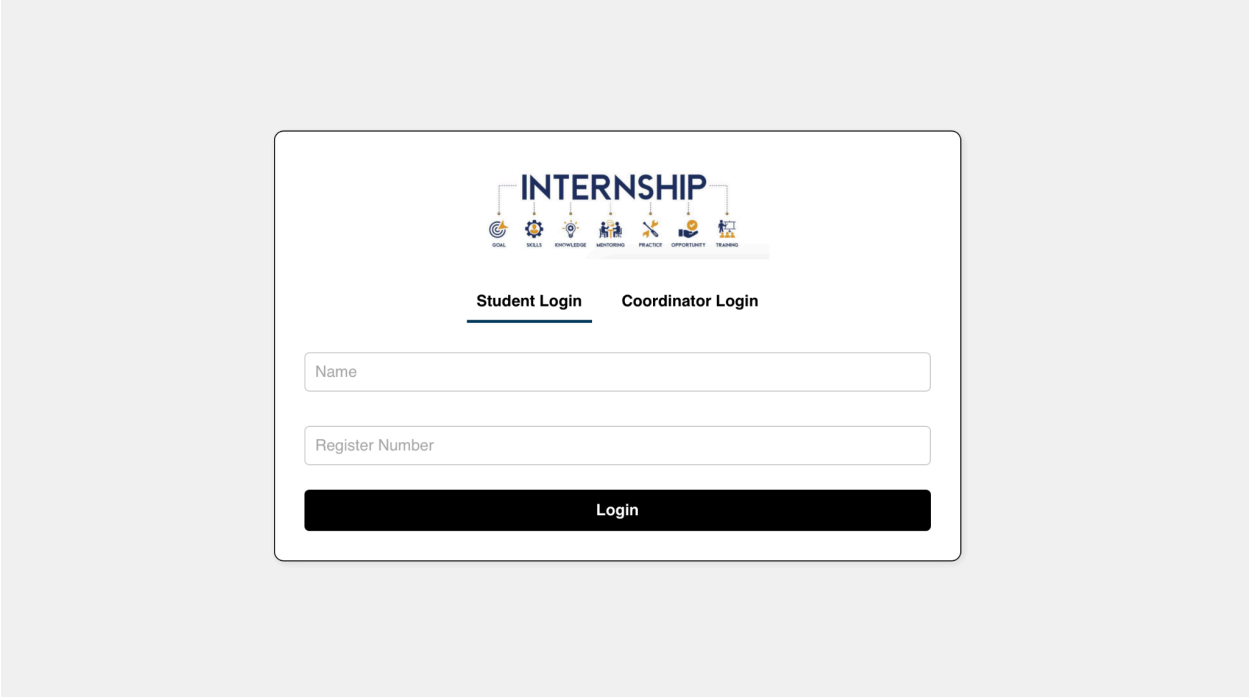
```
}


#jsonData table {
    width: 100%;
    border-collapse: collapse;
}


#jsonData th, #jsonData td {
    border: 1px solid black;
    padding: 8px;
    text-align: center;
}

#jsonData th {
    background-color: #f2f2f2;
}

.top {
    display: flex;
    align-items: center;
    gap: 40px;
}
body {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    background-color: #f4f4f4;
}
.container {
    width: 700px;
    padding: 30px;
    border: 1px solid black;
    background-color: white;
    text-align: center;
    box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
    border-radius: 10px;
}
.filter-buttons {
```

```
    display: flex;
    justify-content: space-between;
    margin-bottom: 20px;
}
.column {
    display: flex;
    flex-direction: column;
    gap: 10px;
    width: 48%;
}
button {
    padding: 10px;
    border: none;
    background-color: #2a2828;
    color: white;
    cursor: pointer;
    border-radius: 5px;
    transition: transform 0.3s ease, background-color 0.3s ease, box-shadow
0.3s ease;
}
button:hover {
    background-color: black;
    transform: scale(1.1);
}
.modal-overlay {
    display: block;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    z-index: 999;
}
.modal {
    display: block;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: white;
    padding: 20px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.5);
```

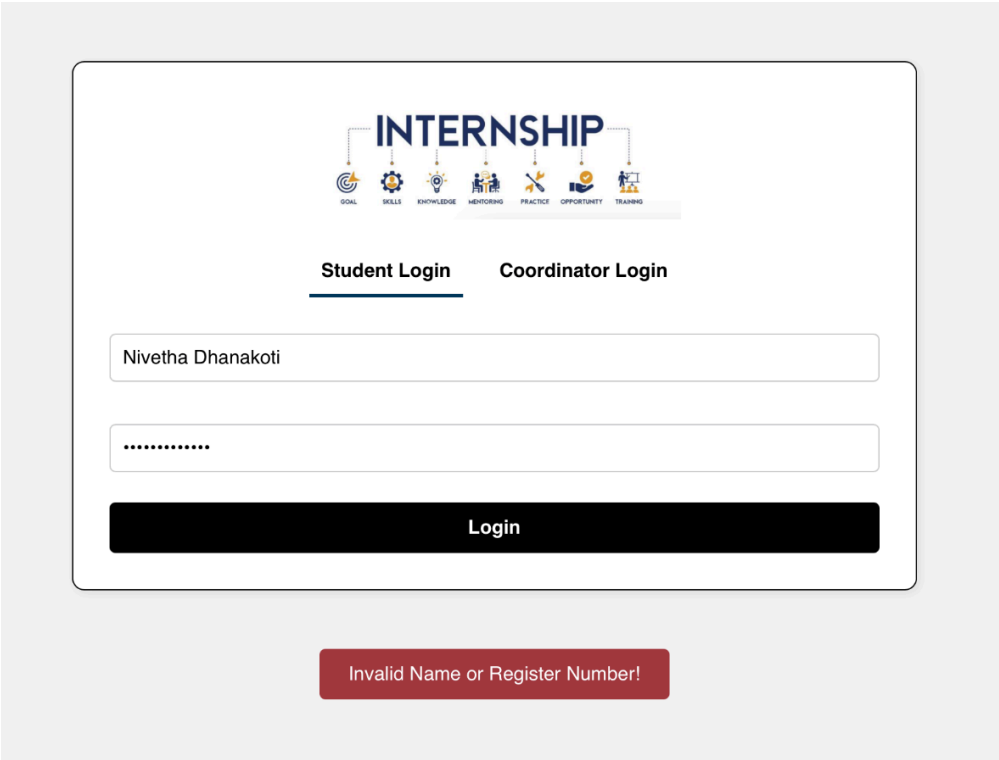
```
    z-index: 1000;
    max-width: 95%;
    max-height: 80%;
    overflow-y: auto;
}

.modal .close {
    position: absolute;
    top: 10px;
    right: 15px;
    font-size: 20px;
    cursor: pointer;
}

img {
    mix-blend-mode: multiply;
    background-color: #f4f4f4;
    height: 310px;
}
```

OUTPUT:

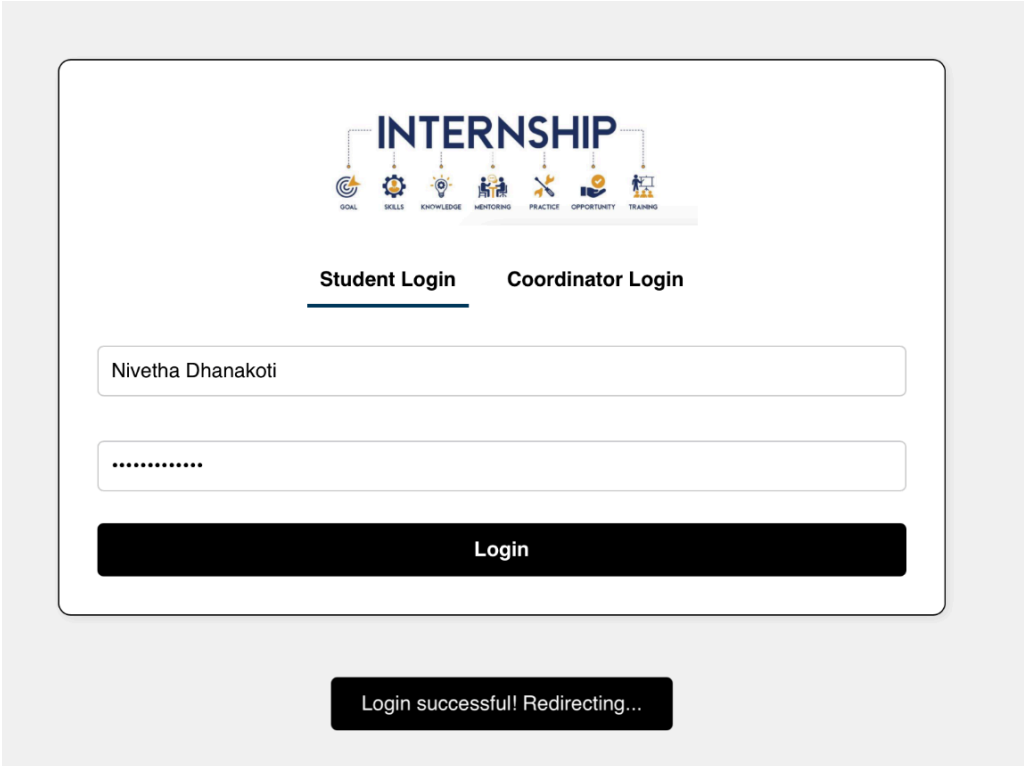
Initially, on running the React App with npm start, the following screen is displayed for 3 seconds. Then, the login page is loaded which has 2 options: Student login and Coordinator login.

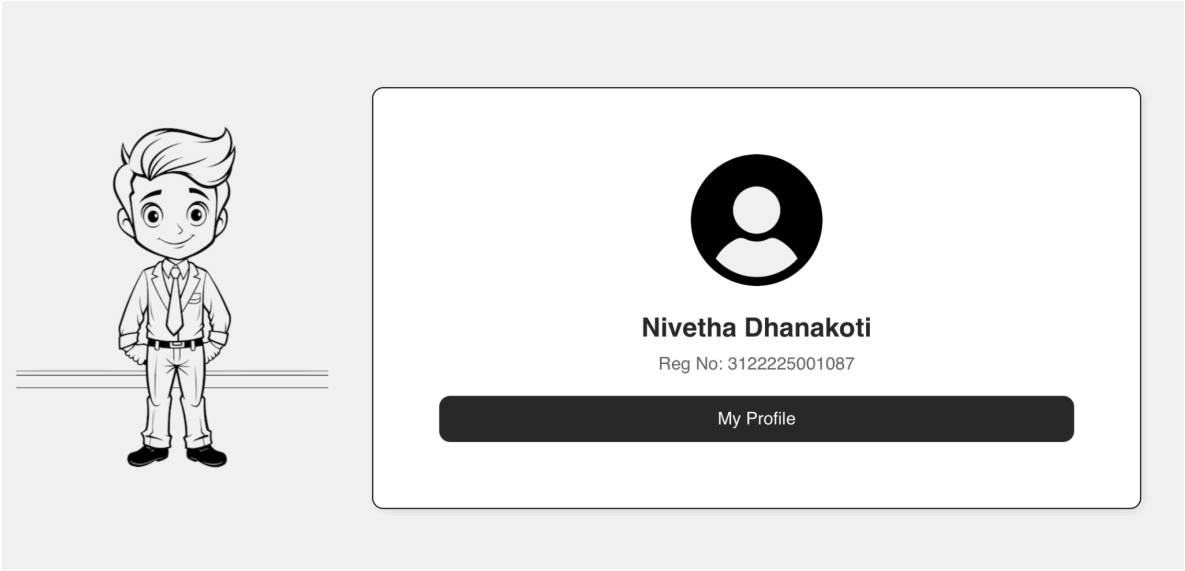


On entering the Student name and register number, user will be redirected to Profile page. On entering the wrong details (mismatch with the one stored in Google Sheets), it will show error message.

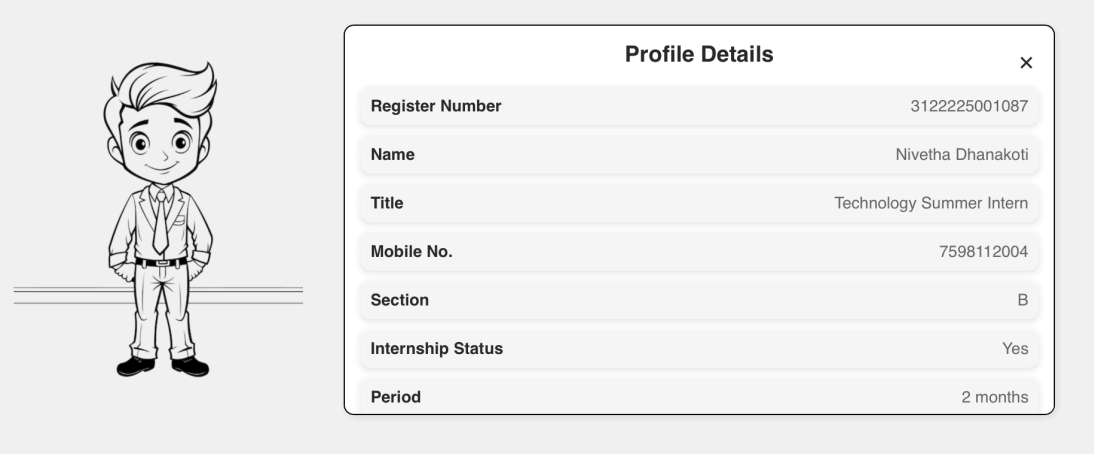


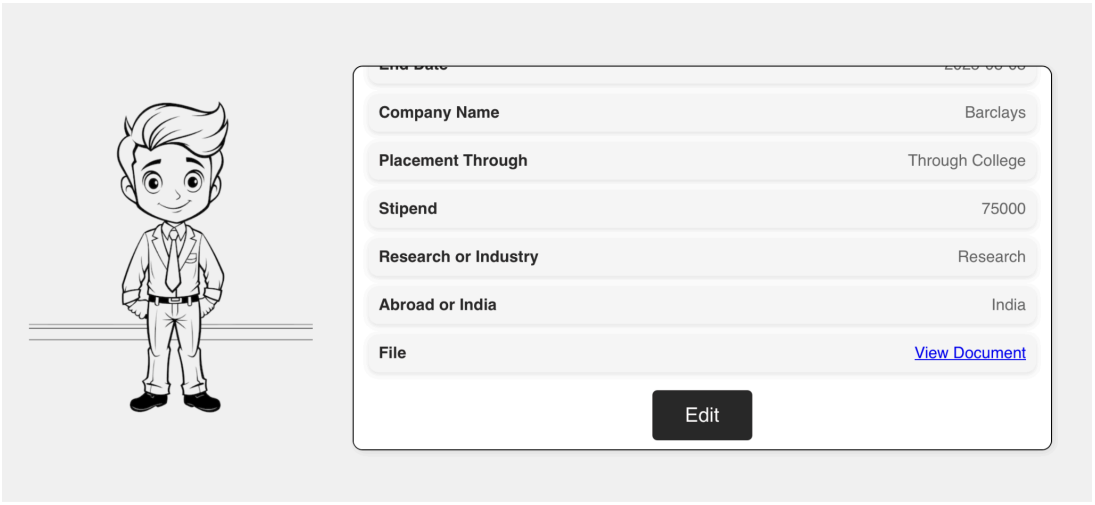
If the details match with the ones from Google sheets, then it will display a redirecting message and will lead to Profile page.





On clicking the My Profile button on the Profile page, it will display all the details associated with the student. If there is no details, then the field will be left blank.

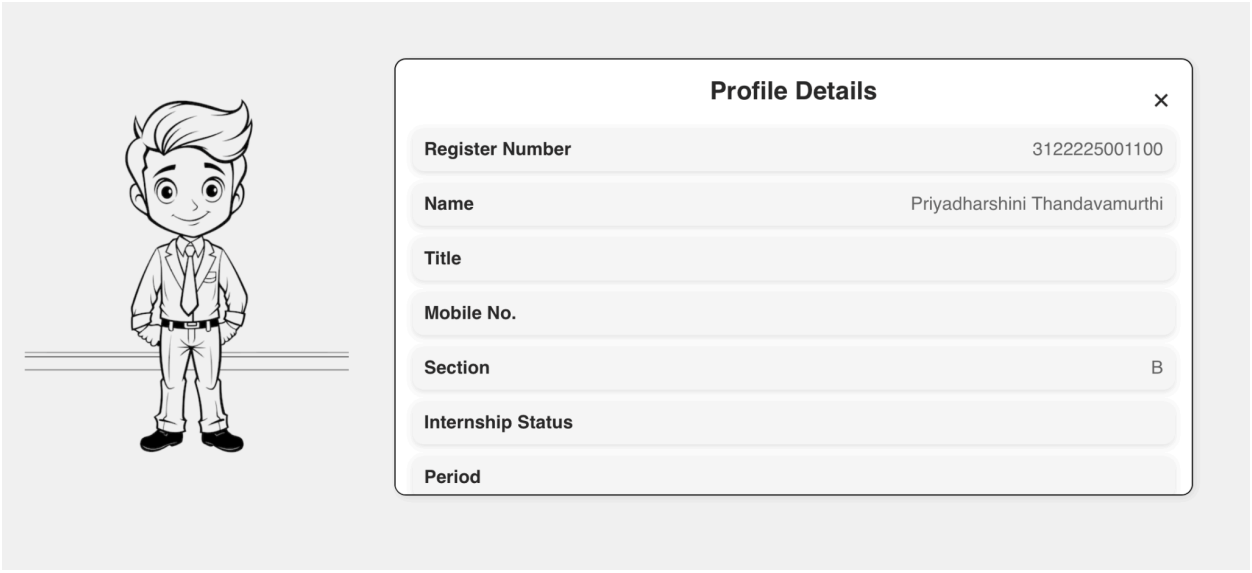


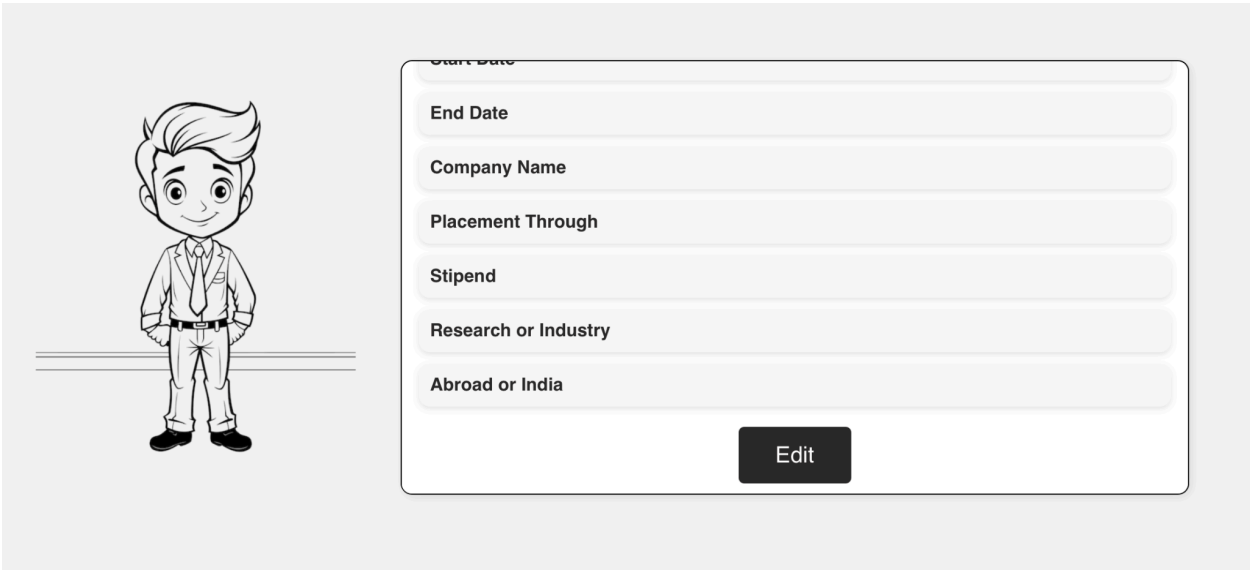


Since all the fields for the particular student is already filled in Google sheets, it reflects those details. Below shows the example of a student profile who didnt fill any field in Google sheets.

The contents for the register number : 3122225001100 in the Google sheets is empty initially.

3122225001098	Preethi Prative			B										
3122225001099	Prithivi Rajan D	No Internship	6374588076	B	No									
3122225001100	Priyadharshini Thandavamurthi			B										
3122225001101	Rajesh G	No Internship	8220133879	B	No									





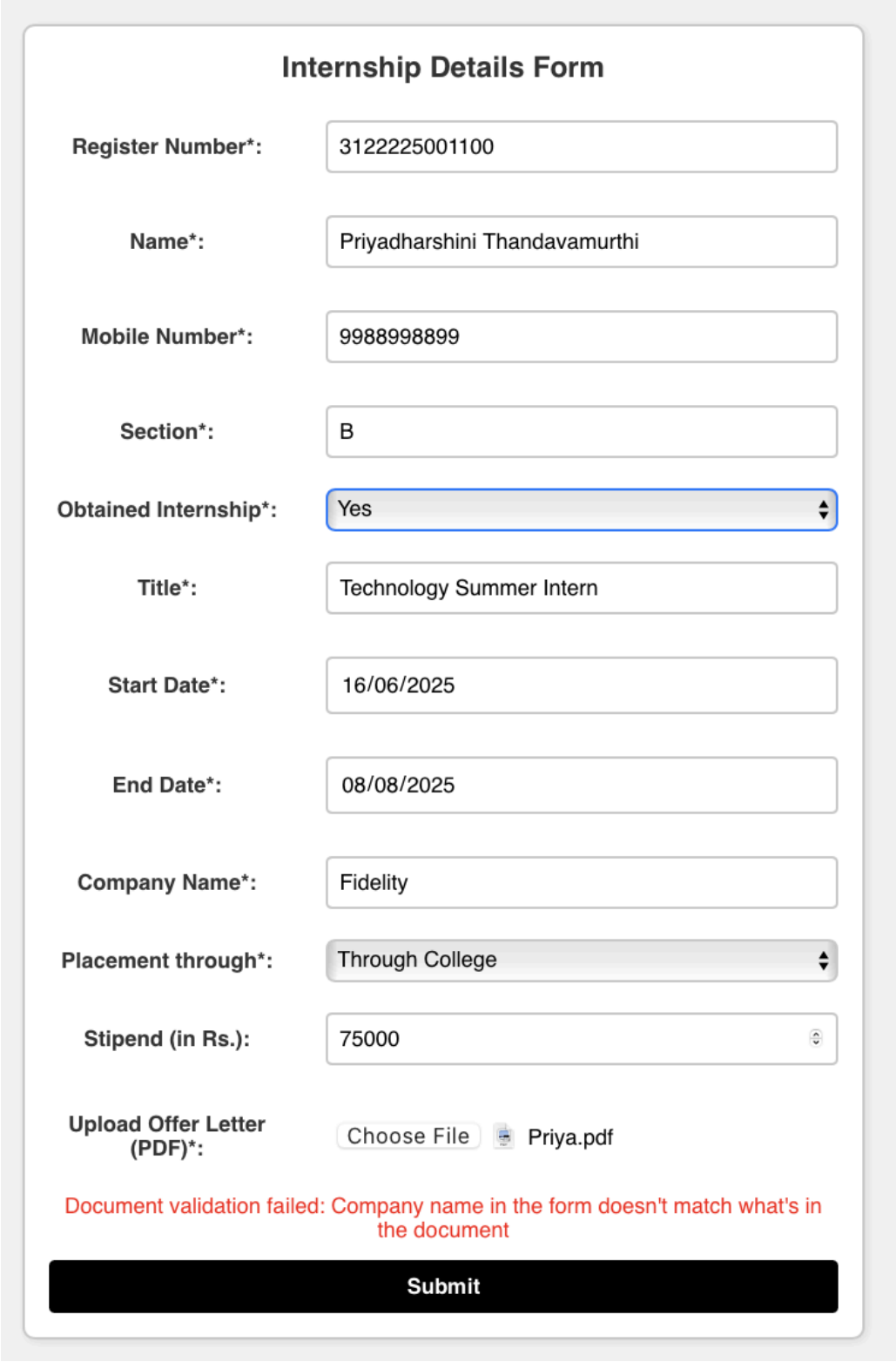
Since the content for the particular student is not filled in google sheets, it displays nothing. Clicking on the Edit button, it will lead to InternshipForm page, where the student can enter the details and will have to upload the offer letter which then will be validated with the details entered by the student.

Internship Details Form

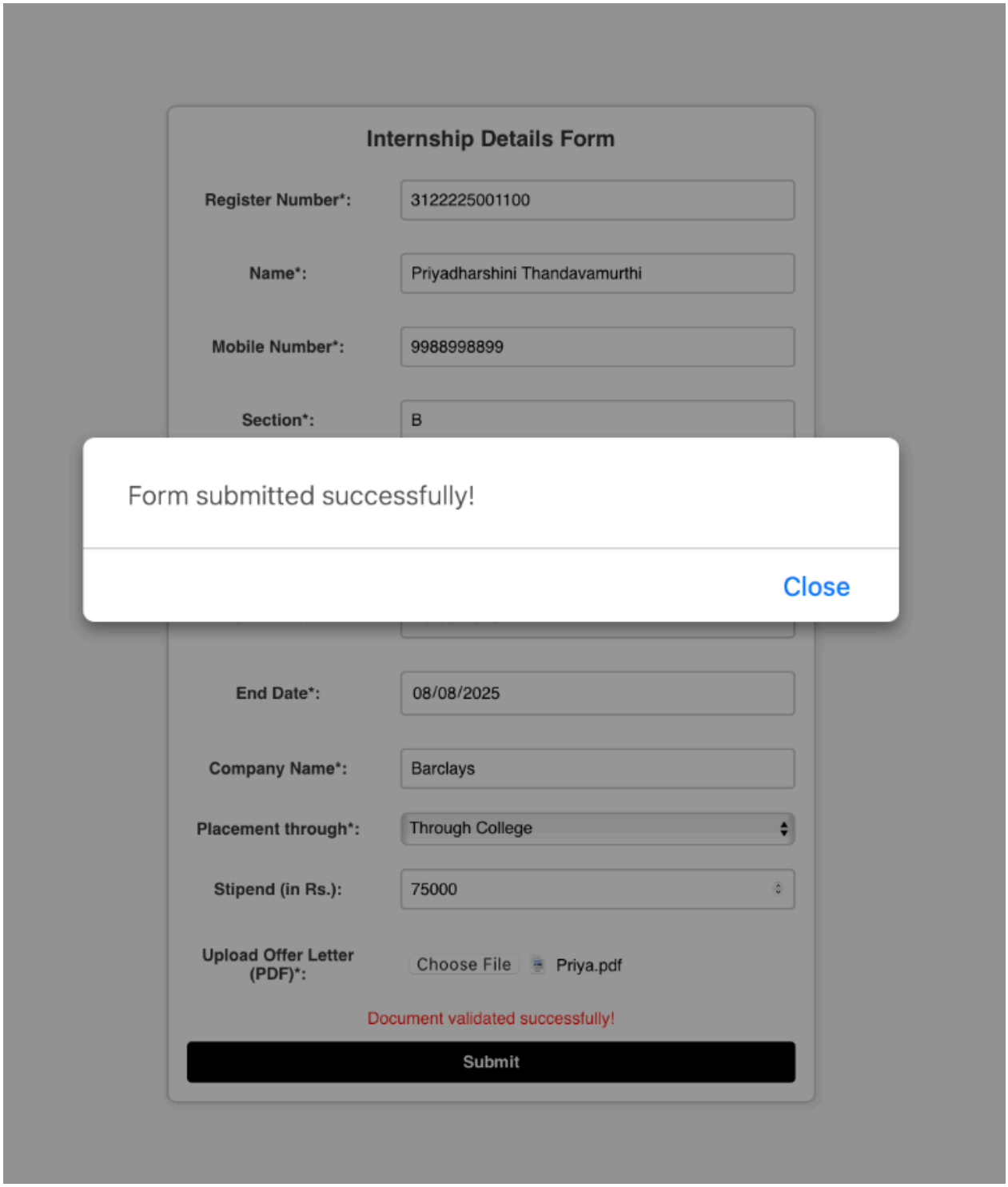
Register Number*:	3122225001100
Name*:	Priyadharshini Thandavamurthi
Mobile Number*:	
Section*:	B
Obtained Internship*:	Yes
Title*:	
Start Date*:	04/04/2025
End Date*:	04/04/2025
Company Name*:	
Placement through*:	Through College
Stipend (in Rs.):	
Upload Offer Letter (PDF)*:	Choose File no file selected

Submit

On entering the wrong details, the error message will be displayed.

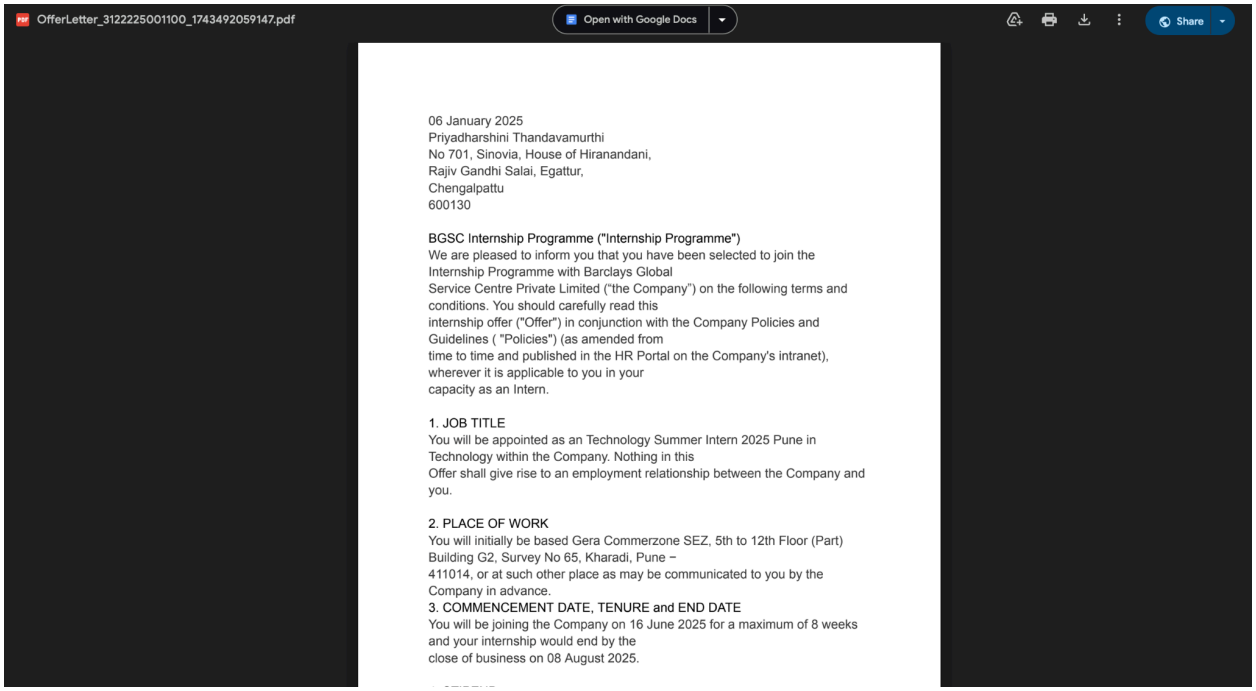


Once the student enters the right details, it will submit and the details will be reflected in the Google sheets and the offer letter will be stored in the Proof folder connected to Google drive.

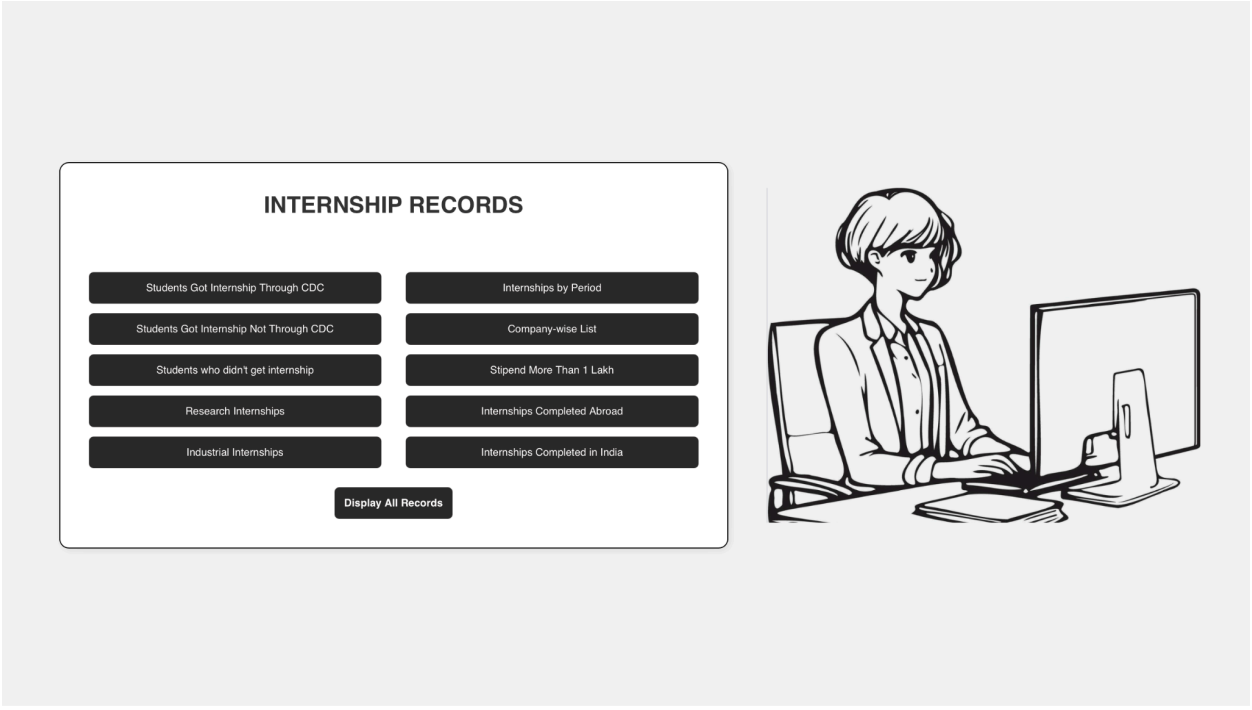


3122225001099	Prithivi Rajan D	No Internship	6374588076	B	No									
3122225001100	Priyadharshini Thandavamurthi	Technology Sum	9988998899	B			2025-06-16	2025-08-08	Barclays		75,000			https://drive.google.com/file/d/1fCHkzbPlAsyrALJ0W__SufP7jcuu434C/view?usp=drivesdk
3122225001101	Rajesh G	No Internship	8220133879	B	No									

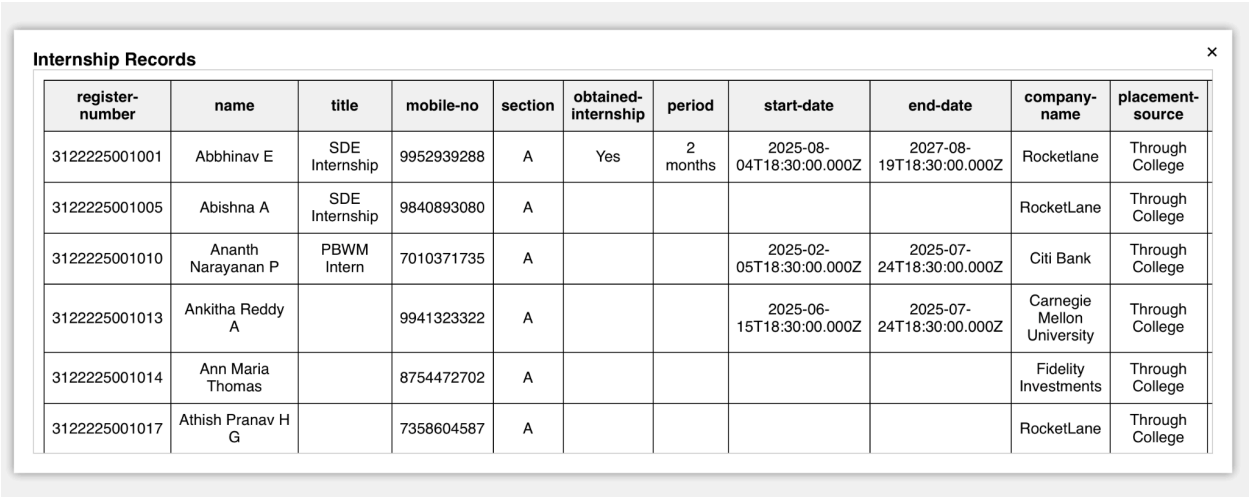
Stored in google drive:



If the login is done by the coordinator with username and password, then the following page will be loaded. It has 11 buttons which works like filters and upon clicking each the filtered content will be displayed.



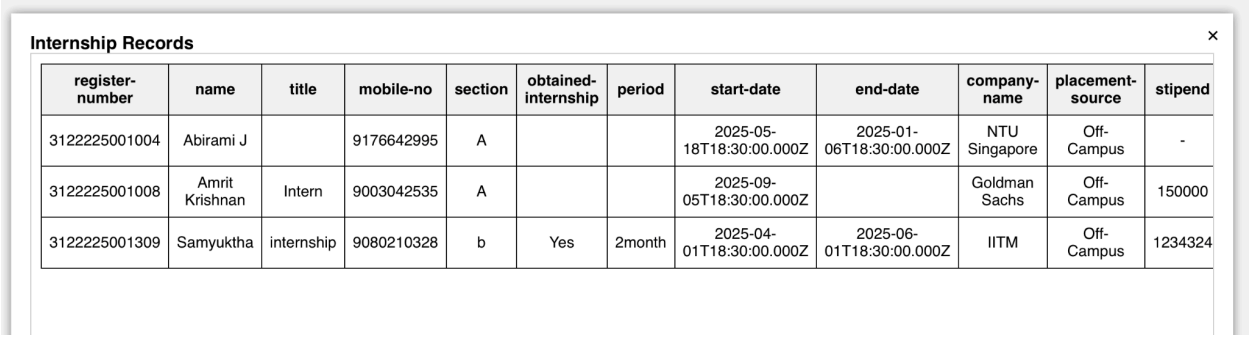
Through College (CDC)



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source
3122225001001	Abbhinav E	SDE Internship	9952939288	A	Yes	2 months	2025-08-04T18:30:00.000Z	2027-08-19T18:30:00.000Z	Rocketlane	Through College
3122225001005	Abishna A	SDE Internship	9840893080	A					RocketLane	Through College
3122225001010	Ananth Narayanan P	PBWM Intern	7010371735	A			2025-02-05T18:30:00.000Z	2025-07-24T18:30:00.000Z	Citi Bank	Through College
3122225001013	Ankitha Reddy A		9941323322	A			2025-06-15T18:30:00.000Z	2025-07-24T18:30:00.000Z	Carnegie Mellon University	Through College
3122225001014	Ann Maria Thomas		8754472702	A					Fidelity Investments	Through College
3122225001017	Athish Pranav H G		7358604587	A					RocketLane	Through College

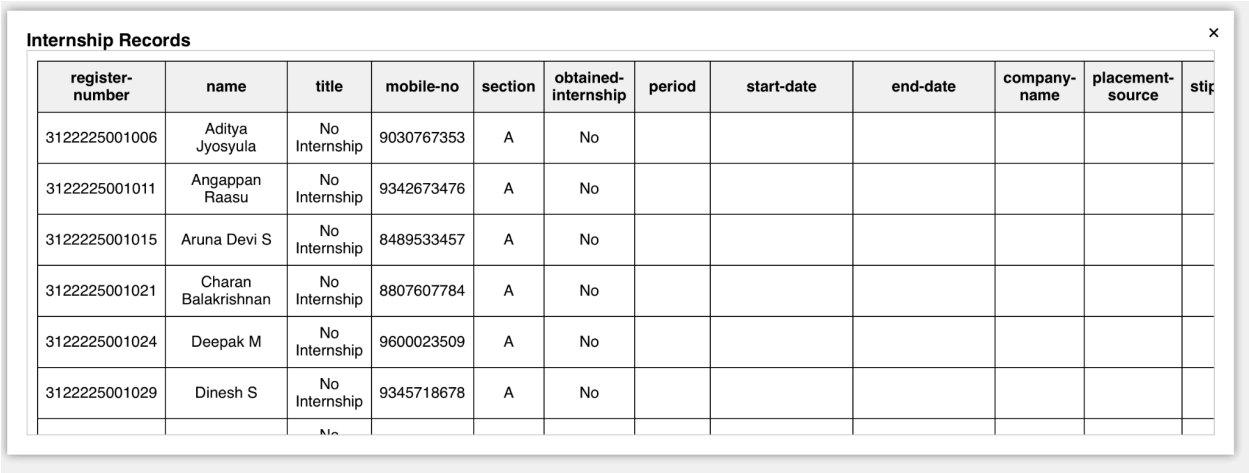
Not Through CDC



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend
3122225001004	Abirami J		9176642995	A			2025-05-18T18:30:00.000Z	2025-01-06T18:30:00.000Z	NTU Singapore	Off-Campus	-
3122225001008	Amrit Krishnan	Intern	9003042535	A			2025-09-05T18:30:00.000Z		Goldman Sachs	Off-Campus	150000
3122225001309	Samyuktha	internship	9080210328	b	Yes	2month	2025-04-01T18:30:00.000Z	2025-06-01T18:30:00.000Z	IITM	Off-Campus	1234324

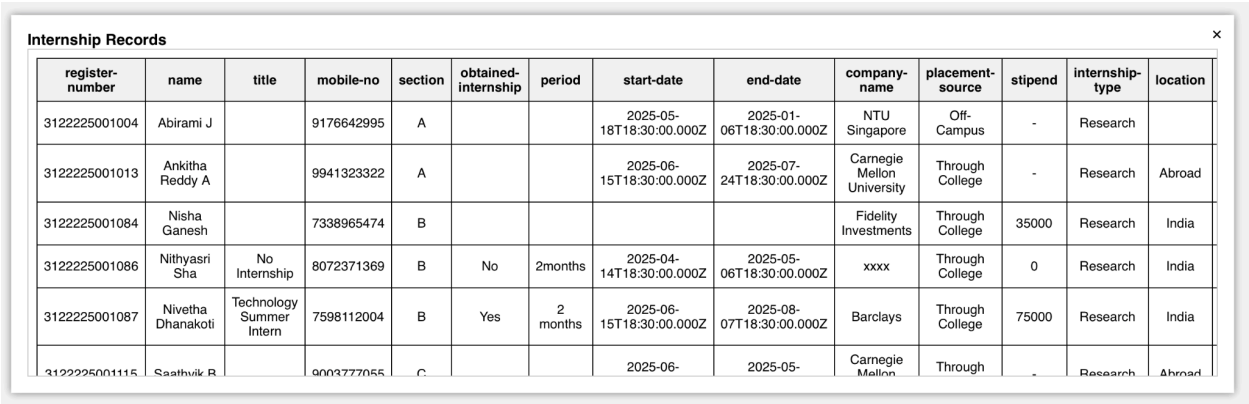
Students who didnt get internship



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stip
3122225001006	Aditya Jyosyula	No Internship	9030767353	A	No						
3122225001011	Angappan Raasu	No Internship	9342673476	A	No						
3122225001015	Aruna Devi S	No Internship	8489533457	A	No						
3122225001021	Charan Balakrishnan	No Internship	8807607784	A	No						
3122225001024	Deepak M	No Internship	9600023509	A	No						
3122225001029	Dinesh S	No Internship	9345718678	A	No						

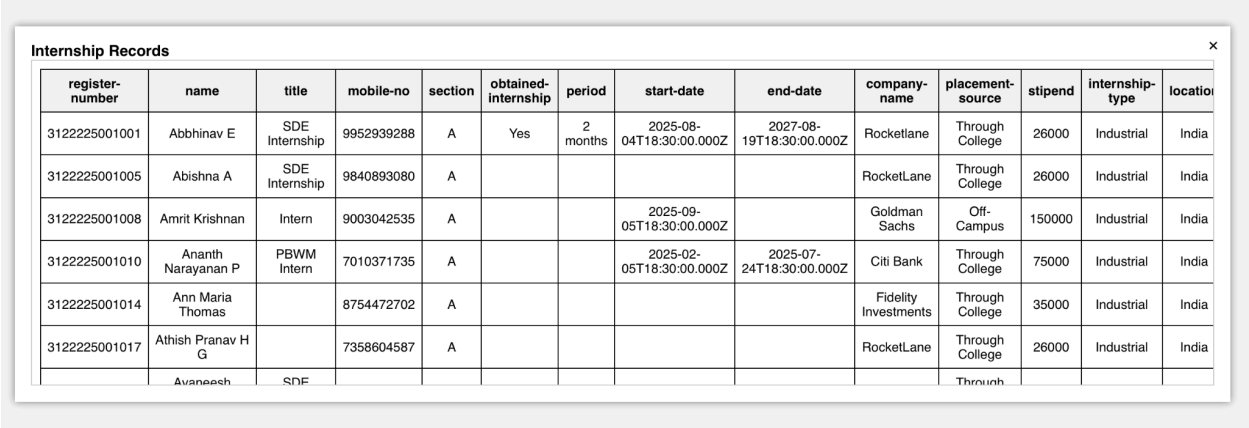
Research Internships



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	location
3122225001004	Abirami J		9176642995	A			2025-05-18T18:30:00.000Z	2025-01-06T18:30:00.000Z	NTU Singapore	Off-Campus	-	Research	
3122225001013	Ankitha Reddy A		9941323322	A			2025-06-15T18:30:00.000Z	2025-07-24T18:30:00.000Z	Carnegie Mellon University	Through College	-	Research	Abroad
3122225001084	Nisha Ganesh		7338965474	B					Fidelity Investments	Through College	35000	Research	India
3122225001086	Nithyasri Sha	No Internship	8072371369	B	No	2months	2025-04-14T18:30:00.000Z	2025-05-06T18:30:00.000Z	xxxx	Through College	0	Research	India
3122225001087	Nivetha Dhanakoti	Technology Summer Intern	7598112004	B	Yes	2 months	2025-06-15T18:30:00.000Z	2025-08-07T18:30:00.000Z	Barclays	Through College	75000	Research	India
3122225001115	Saathvik B		9003777055	C			2025-06-	2025-05-	Carnegie Mellon	Through	-	Research	Abroad

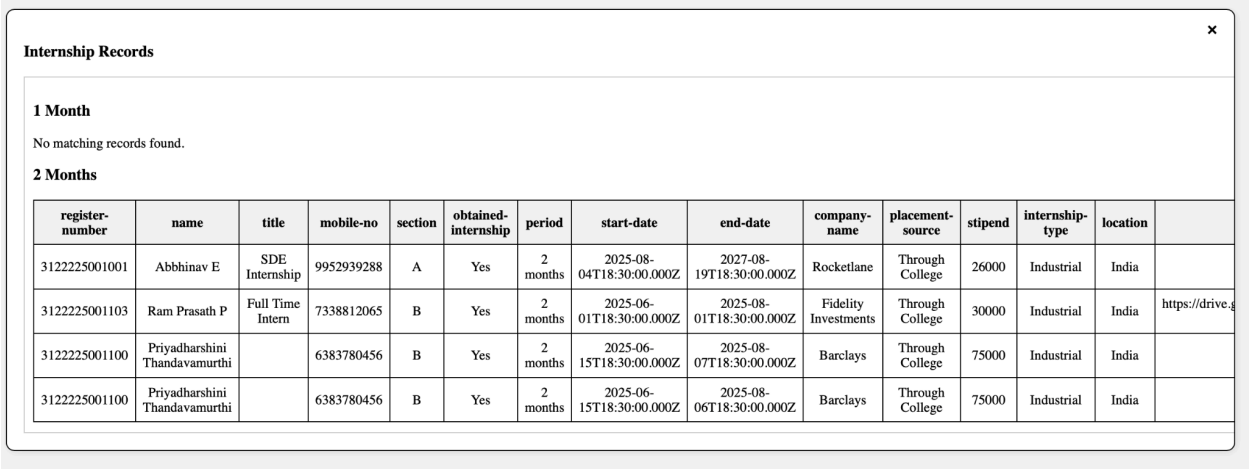
Industrial Internships



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	locatio
3122225001001	Abbhinav E	SDE Internship	9952939288	A	Yes	2 months	2025-08-04T18:30:00.000Z	2027-08-19T18:30:00.000Z	Rocketlane	Through College	26000	Industrial	India
3122225001005	Abishna A	SDE Internship	9840893080	A					RocketLane	Through College	26000	Industrial	India
3122225001008	Amrit Krishnan	Intern	9003042535	A			2025-09-05T18:30:00.000Z		Goldman Sachs	Off-Campus	150000	Industrial	India
3122225001010	Ananth Narayanan P	PBWM Intern	7010371735	A			2025-02-05T18:30:00.000Z	2025-07-24T18:30:00.000Z	Citi Bank	Through College	75000	Industrial	India
3122225001014	Ann Maria Thomas		8754472702	A					Fidelity Investments	Through College	35000	Industrial	India
3122225001017	Athish Pranav H G		7358604587	A					RocketLane	Through College	26000	Industrial	India
	Avaneesh	SDE								Through			

Internships by period



Internship Records

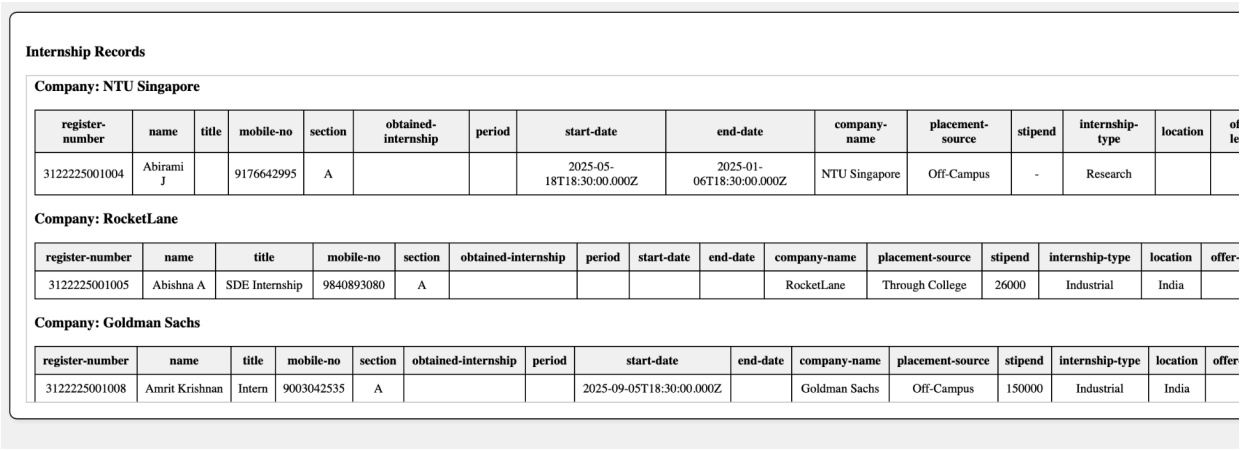
1 Month

No matching records found.

2 Months

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	location	
3122225001001	Abbhinav E	SDE Internship	9952939288	A	Yes	2 months	2025-08-04T18:30:00.000Z	2027-08-19T18:30:00.000Z	Rocketlane	Through College	26000	Industrial	India	
3122225001103	Ram Prasath P	Full Time Intern	7338812065	B	Yes	2 months	2025-06-01T18:30:00.000Z	2025-08-01T18:30:00.000Z	Fidelity Investments	Through College	30000	Industrial	India	https://drive.g
3122225001100	Priyadharshini Thandavamurthi		6383780456	B	Yes	2 months	2025-06-15T18:30:00.000Z	2025-08-07T18:30:00.000Z	Barclays	Through College	75000	Industrial	India	
3122225001100	Priyadharshini Thandavamurthi		6383780456	B	Yes	2 months	2025-06-15T18:30:00.000Z	2025-08-06T18:30:00.000Z	Barclays	Through College	75000	Industrial	India	

Company wise



Internship Records

Company: NTU Singapore

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	location	of le
3122225001004	Abirami J		9176642995	A			2025-05-18T18:30:00.000Z	2025-01-06T18:30:00.000Z	NTU Singapore	Off-Campus	-	Research		

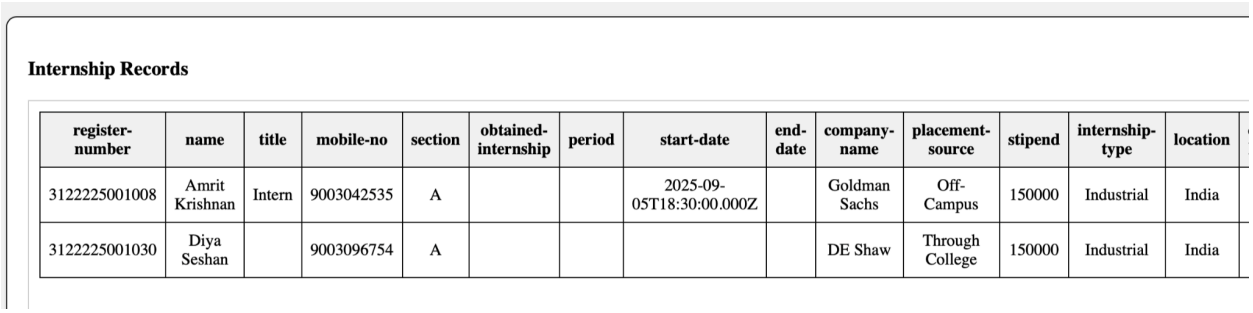
Company: RocketLane

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	location	offer-
3122225001005	Abishna A	SDE Internship	9840893080	A					RocketLane	Through College	26000	Industrial	India	

Company: Goldman Sachs

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	location	offer
3122225001008	Amrit Krishnan	Intern	9003042535	A			2025-09-05T18:30:00.000Z		Goldman Sachs	Off-Campus	150000	Industrial	India	

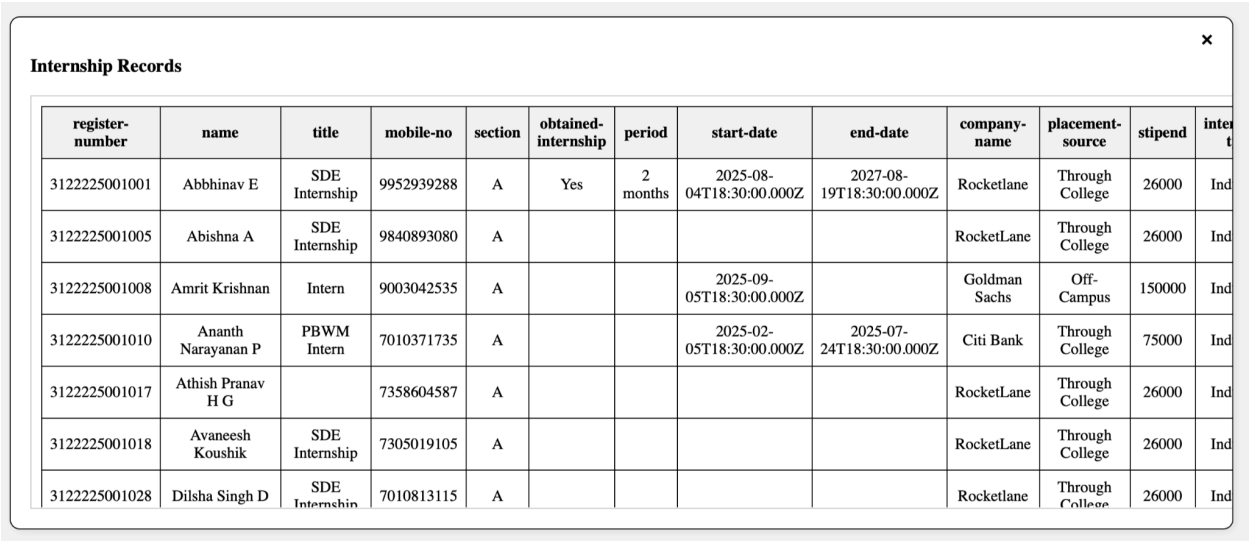
Stipend more than 1 lakh



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	location
3122225001008	Amrit Krishnan	Intern	9003042535	A			2025-09-05T18:30:00.000Z		Goldman Sachs	Off-Campus	150000	Industrial	India
3122225001030	Diya Seshan		9003096754	A					DE Shaw	Through College	150000	Industrial	India

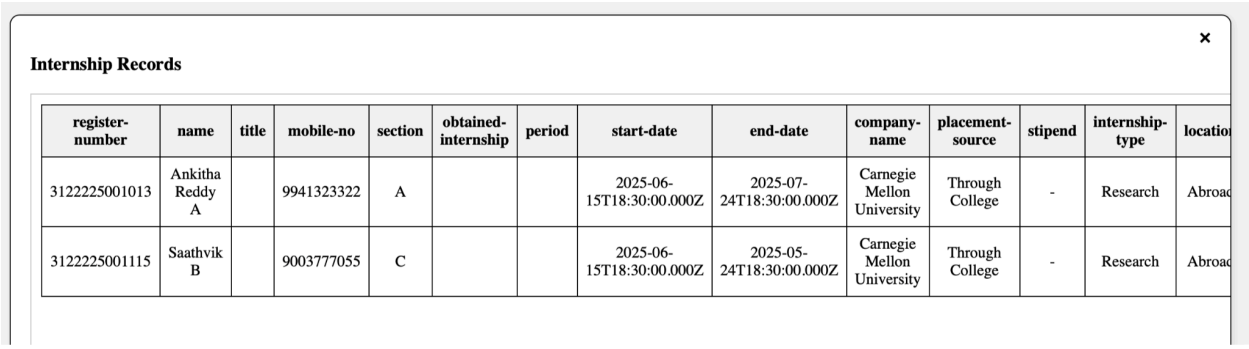
Internship at India



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	inter t
3122225001001	Abbhinav E	SDE Internship	9952939288	A	Yes	2 months	2025-08-04T18:30:00.000Z	2027-08-19T18:30:00.000Z	Rocketlane	Through College	26000	Ind
3122225001005	Abishna A	SDE Internship	9840893080	A					RocketLane	Through College	26000	Ind
3122225001008	Amrit Krishnan	Intern	9003042535	A			2025-09-05T18:30:00.000Z		Goldman Sachs	Off-Campus	150000	Ind
3122225001010	Ananth Narayanan P	PBWM Intern	7010371735	A			2025-02-05T18:30:00.000Z	2025-07-24T18:30:00.000Z	Citi Bank	Through College	75000	Ind
3122225001017	Athish Pranav H G		7358604587	A					RocketLane	Through College	26000	Ind
3122225001018	Avaneesh Koushik	SDE Internship	7305019105	A					RocketLane	Through College	26000	Ind
3122225001028	Dilsha Singh D	SDE Internship	7010813115	A					Rocketlane	Through College	26000	Ind

Abroad Internships



Internship Records

register-number	name	title	mobile-no	section	obtained-internship	period	start-date	end-date	company-name	placement-source	stipend	internship-type	locatio
3122225001013	Ankitha Reddy A		9941323322	A			2025-06-15T18:30:00.000Z	2025-07-24T18:30:00.000Z	Carnegie Mellon University	Through College	-	Research	Abroad
3122225001115	Saathvik B		9003777055	C			2025-06-15T18:30:00.000Z	2025-05-24T18:30:00.000Z	Carnegie Mellon University	Through College	-	Research	Abroad

INFERENCES:

- InternTrack provides a centralized platform for managing internship records with controlled access.
- The integration of Google Drive and Google Sheets ensures efficient data storage and retrieval while maintaining data integrity.
- A **login-based authentication system** prevents unauthorized access.
- The application **does not use direct SQL queries**, reducing the risk of SQL injection.
- The **Tesseract OCR library** is used to extract text from uploaded documents. This ensures that only **valid internship offer letters** are accepted, reducing manual verification errors.
- Students can **view and edit** only their own internship details. Coordinators have access to **filtered views and data management options**, ensuring proper administrative control.

LEARNING OUTCOMES:

- **Full-Stack Development:** Implemented a web application using **React.js** for the frontend and **Google Sheets API** for backend data storage, ensuring seamless integration.
- **OCR-Based Document Validation:** Utilized **Python Tesseract** for text extraction from proof documents, validating internship details against user inputs.
- **Secure Authentication & Data Integrity:** Designed a login system with unique user identification to maintain accurate and non-redundant student records.
- **API Integration & Data Handling:** Connected **Google Drive API** for document storage and implemented filtering mechanisms to efficiently retrieve and display internship data.
- **Web Security Best Practices:** Implemented **input validation** and followed security measures to prevent vulnerabilities such as SQL injection.