Implementation of InternTrack Full Stack Web Application

UCS2611 – Internet Programming Lab

A PROJECT REPORT

Submitted By

MUGILKRISHNA D U	3122 22 5001 073
NIKILESH JAYAGUPTHA	3122 22 5001 081
NIRANJAN B	3122 22 5001 082

Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Kalavakkam – 603110

November 2023

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

TABLE OF CONTENTS:

1. AIM

To design and implement a full-stack internship management web application—InternTrack—that automates the verification, storage, and viewing of internship details and documents for students and faculty. It leverages React.js for the frontend, Node.js + Express.js for the main backend, and Flask + Tesseract OCR for document processing, with integration of Google Sheets, and Google Drive APIs.

2. PROBLEM STATEMENT

In academic institutions like colleges, internships play a vital role in providing students with great exposure to industry environments. It allows them to work on various projects and interact with like-minded individuals. However, there is a lack of a centralized and systematic platform needed to track, manage, and verify the internship details. This often leads to inefficiencies, inconsistency in the data and huge communication gaps between students and coordinators

The current systems for maintaining internship records are prone to human error and they do not offer flexible retrieval options. There is also a growing need to handle document storage securely and most importantly prevent data duplication, and ensure protection against common web vulnerabilities such as SQL injection.

To address all these challenges, we developed a full-stack web application named InternTrack. The application developed supports user authentication for both students and coordinators, while also enabling the uploading of internship details and proofs. It also provides proper retrieval of data with various parameters, thereby allowing secure editing of records while maintaining data uniqueness and integrity in the backend.

3. REQUIREMENTS

1. Development Stack & Frameworks:

Frontend:

- React.js – For building dynamic and responsive user interfaces.
- Axios – For sending HTTP requests to the backend.
- Redux – For centralized state management (e.g., user session, role-based navigation).

Backend:

- Node.js – JavaScript runtime environment for executing server-side code.
- Express.js – Web framework for handling API routes and business logic.
- Flask (Python) – Used for document text extraction and OCR processing.

Database:

- MongoDB – Stores and validates user login credentials.

2. Tools & Utilities:

- Postman – For testing and verifying API endpoints (GET, POST, etc.).
- Visual Studio Code – Code editor used for frontend and backend development.
- Git & GitHub – For version control and collaboration.

3. Document Handling & OCR:

- Tesseract OCR – To extract structured text from PDF/image-based offer letters.
- pdf2image / PyMuPDF / OpenCV – Used for converting PDFs to images and preprocessing them before OCR.

4. Authentication & Security:

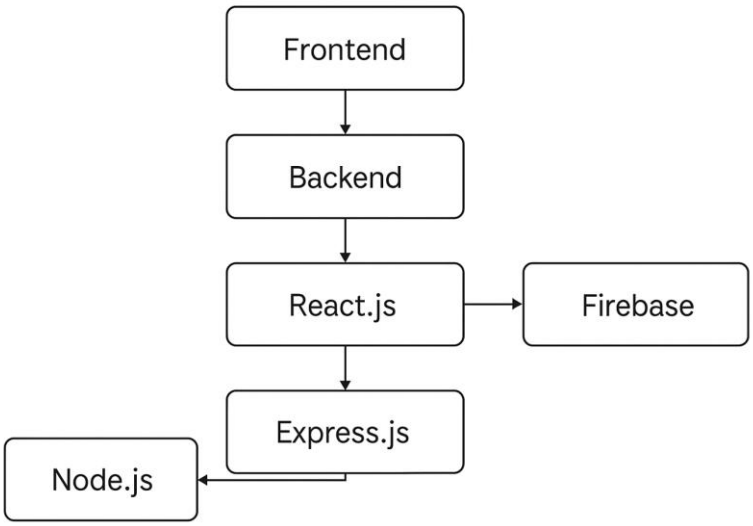
- JSON Web Tokens (JWT) – For securely maintaining authenticated sessions.
- CORS – Enables Cross-Origin Resource Sharing between React, Node.js, and Flask servers.

5. Validation & Injection Prevention:

- Input Validation – Basic checks in both frontend and backend to prevent invalid or malicious input.
- Google Sheets Verification – Cross-checks extracted data with verified institutional records.

4. FLOW DIAGRAMS

4.1) GENERAL RELATIONSHIP BETWEEN COMPONENTS:



Frontend Layer

- This represents the user interface, built using HTML, CSS, and JavaScript.
- It communicates with the backend to perform actions like form submission, file upload, etc.

Backend Layer

- Acts as the middle layer that processes requests from the frontend.
- It involves logic for routing, APIs, or handling data.

React.js as Part of the Frontend

- React.js is the framework used to build dynamic frontend components.
- It sends and receives data using Axios or fetch to interact with backend APIs.

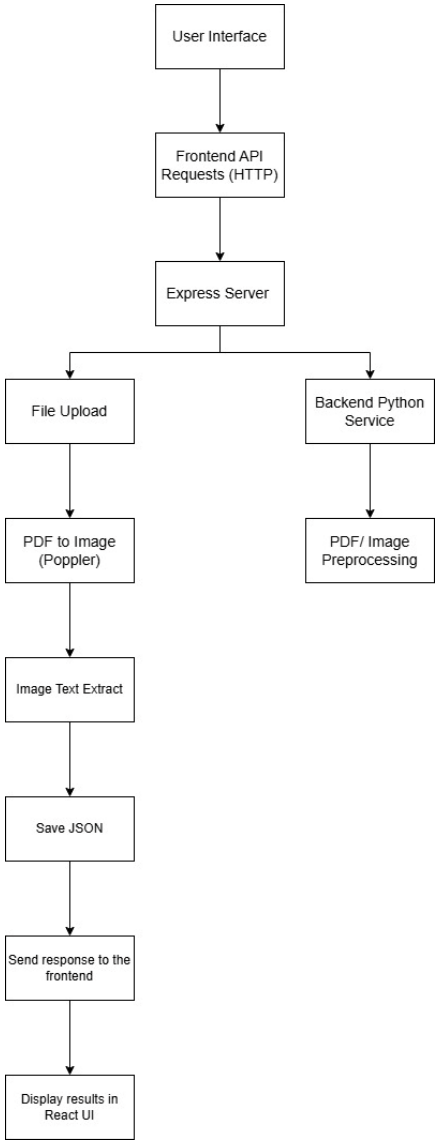
Express.js as a Backend Framework

- Express.js runs on top of Node.js and handles HTTP requests/responses.
- It connects React to the database and defines API routes for file uploads, data extraction, etc.

Node.js as the Runtime Environment

- Node.js runs the backend server and executes the Express.js code.
- It handles asynchronous operations and integrates with external libraries or services.

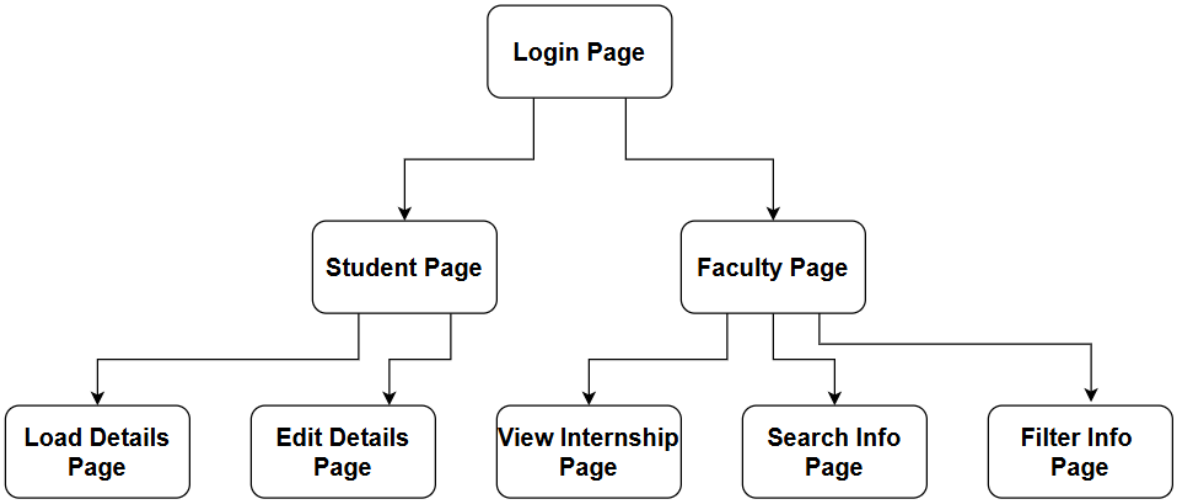
4.2) VALIDATING OFFER LETTERS – FLOW DIAGRAM



1. Frontend (React):
    - User uploads a file (.pdf or .png).
    - Axios sends the file to the backend via an HTTP request.
2. Backend (Node.js with Express):
    - Saves it in the uploads/ folder.
    - Triggers a Python script to process the file.

3. Python Service:
    - If it’s a PDF, it's converted to images (one per page) using pdftoppm or PyMuPDF.
    - Each image is cleaned and passed to Tesseract OCR.
    - Extracted text is cleaned/formatted and returned.
4. Response Flow:
    - The processed text is sent back to the Express server.
    - Express sends the final response to the React frontend.

4.3) COMPONENT DIAGRAM:



EXPLANATION OF THE COMPONENTS:

index.html: This is the login page where users can sign in using their Google account to access the system.

home.html: This serves as the dashboard for users, allowing navigation to different sections like loading details, viewing internships, retrieving information, and editing details.

faculty.html: Faculty members can view, search, and filter internship records of students. It also provides an option to sign out.

student.html: Students can load and edit their internship details from this page.

loaddetails.html: A form where students can enter and submit their internship details, including company name, duration, stipend, and document uploads.

view.html: Displays a table of internship records with details like student names, titles, company names, and document statuses.

search.html: Allows users to search for specific internships based on various criteria.

filter.html: Provides dropdown filters for users to sort internships based on parameters like location, placement type, duration, etc.

edit.html: Likely a page where students or faculty can modify previously submitted internship records.

5. FUNCTIONALITIES USED IN THE PROJECT

5.1) BACKEND

1. **PDF Text Extraction (Flask + Tesseract)**
    - Converts uploaded PDFs into images using pdf2image or PyMuPDF.
    - Preprocesses images using OpenCV (grayscale, thresholding).
    - Extracts structured text using pytesseract.
    - Uses regex to extract key fields like Name, Register Number, Mobile Number, Company, etc.
2. **Flask API Endpoints**
    - POST /upload – Accepts a file and triggers the OCR pipeline.
    - GET /extracteddata – Returns the latest extracted data from OCR.
3. **Node.js + Express Backend**
    - Handles API routes for login, verification, data fetch, and update.
    - Applies CORS middleware to support communication across ports.
4. **Google Sheets Integration**
    - Fetches all records from the Google Sheet (/get, /getall).
    - Matches Register Number and Mobile Number from extracted data.
    - Updates student internship details in the sheet if a match is found (/update).

5. **Google Drive Integration**
    - Checks if a folder exists for the student using Drive API.
    - If not present, creates a folder and uploads the verified document into it.
6. **Multer File Upload Handling**
    - Multer is used in the Node backend to receive uploaded files (/verify endpoint).
7. **Google Auth (Service Account)**
    - Authenticates access to both Google Sheets and Google Drive APIs using a service account (credentials.json).
8. **Verification Logic**
    - Compares extracted fields (Register Number, Mobile Number, Company Name) with Google Sheets.
    - If a valid match is found, uploads the file to the respective Drive folder.
    - Returns status message (yes or no) based on success or failure.
    - **SQL injection risk is mitigated**, as the project uses **NoSQL (MongoDB)** and **Google APIs**, both of which avoid traditional SQL queries entirely.
9. **MongoDB-Based Login Authentication**
    - Authenticates users (students/faculty) by checking login credentials stored in MongoDB.

5.2) FRONTEND

1. **File Upload UI (React + Axios)**
    - Users can upload PDFs for verification.
    - Sends HTTP requests to:
        - /upload → triggers OCR extraction.
        - /verify → uploads verified files to Drive.

2. **React State Management (Redux + useState/useEffect)**
    - Redux: Manages user session (name, register number, role).
    - useState & useEffect: Handles local state for data extraction, loading, and form inputs.
3. **Display Extracted Data**
    - Shows fields like Name, Register Number, Mobile Number, Company Name, etc., after OCR extraction.
4. **View All Student Data**
    - Fetches internship records using /getall.
    - Displays records in a scrollable and filterable table for faculty review.
5. **Verification & Upload Flow**
    - After file extraction and optional field updates, user uploads the file to Drive with a single click.
    - Displays response status based on verification result.
6. **Manual Data Update Functionality**
    - Students can manually edit internship fields.
    - Sends updated data to the backend (/update) to reflect in Google Sheets.
7. **UI Feedback & Input Validation**
    - Displays relevant success/failure/status messages to users.
    - Performs basic input checks (e.g., non-empty fields, valid email formats).
8. **CORS-Compatible Multi-Port Integration**
    - React (port 3000) communicates with:
        - Node backend (port 5000)
        - Flask OCR backend (port 5001)
    - CORS configuration allows smooth cross-origin requests.

5.3) UNIQUE FUNCTIONALITIES USED

1. **Hybrid Backend Architecture (Node.js + Flask)**
    - Node.js manages routing, database, and Google API logic.
    - Flask Python backend handles OCR and regex-based field parsing.
    - Clean separation between data processing (Python) and system integration (Node).
2. **Automated Document Verification**
    - Uses OCR to extract structured fields from offer letters.
    - Cross-validates with existing institutional records on Google Sheets.
    - Uploads verified files directly to Google Drive in designated folders.
3. **Google Workspace API Integration**
    - **Google Drive API**:
        - Auto-creates and manages student-specific folders.
        - Uploads verified documents post-validation.
    - **Google Sheets API**:
        - Maintains internship records.
        - Supports live update and search via backend.
4. **Redux-Powered Role-Based Access & UI Flow**
    - Redux stores login state and user roles.
    - Dynamically loads pages and navbars based on student/faculty access level.
5. **Real-Time Data Communication**
    - Backend and frontend exchange data in real time.
    - All endpoints work seamlessly across multiple backend ports using CORS.

OVERCOMING THE SQL INJECTION PROBLEM:

```
let result = await collection.findOne({ email: String(email) });
return result;
```

Why this overcomes SQL Injection

- MongoDB is NoSQL: It doesn't use raw SQL queries — so traditional injection attacks (' OR 1=1 --) simply won't work.
- findOne with object-based querying:
  Using an object like { email: String(email) } prevents the query from being interpreted as a string that can be manipulated.
- No string concatenation involved:
  Injection attacks often exploit code that concatenates user inputs directly into a query string (like in SQL). This code does not do that.

6. IMPLEMENTATION – CODE SNIPPETS

FRONTEND:

- index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
```

```
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico"
will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run
build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

- App.js

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Login from "./Login";
import FacultyPage from "./pages/FacultyPage";
import StudentPage from "./pages/StudentPage";
import Update from "./pages/Update";
import Upload from "./pages/Upload";
import View from "./pages/View";
import FilteredPage from "./pages/FilteredPage";
```

```
function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/faculty" element={<FacultyPage />} />
        <Route path="/student" element={<StudentPage />} />
        <Route path="/update" element={<Update />} />
        <Route path="/upload" element={<Upload />} />
        <Route path="/view" element={<View />} />
        <Route path="/filter" element={<FilteredPage />} />
      </Routes>
    </Router>
  );
}

export default App;
```

- App.test.js

```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

- index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import {Provider} from "react-redux"
import { store } from './store';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
```

```
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

- Login.js

```
import axios from 'axios';
import React, { useState } from 'react';
import { useDispatch } from "react-redux"
import {signin,setdetails} from "./userSlice"
import { useNavigate } from 'react-router-dom';
import "./login.css"

import logo from "./assets/nobglogo.png";
const Login = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    await axios.post("http://localhost:5000/login",{
      email:email,
      password:password
    }).then((response)=>{
      if(response.data.success){
        dispatch(signin());
        dispatch(setdetails(response.data));
        if (response.data.UserType==="Student"){
          navigate("/student");
        }
        else{
          navigate("/faculty");
        }
      }
```

```
  })
 };

 return (
  <div className="main_container">
   <div className='LoginSide'>
    <div >
     <form onSubmit={handleSubmit} className='loginform' >
      <h2>Login To Your Account</h2>
      <input
       type="email"
       placeholder="Email"
       value={email}
       onChange={(e) => setEmail(e.target.value)}
       required
      />
      <input
       type="password"
       placeholder="Password"
       value={password}
       onChange={(e) => setPassword(e.target.value)}
       required
      />
      <button type="submit" className='SubmitBtn' >Login</button>
     </form>
    </div>
    <div className='otherside'>
     <img className='logoimg' src={logo} alt='SSN LOGO' />
     <h1>InternTrack</h1>
     <h2>SSN's official platform designed to manage and monitor
internships for both students and faculty, providing a centralized system for
tracking internship activities and approvals.</h2>
    </div>
   </div>
  </div>
 );
};

export default Login;
```

- reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) 
=> {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

- store.js

```
import {configureStore} from "@reduxjs/toolkit"
import userSlice from "./userSlice"
export const store =  configureStore({
    reducer:{
        user:userSlice
    }
})
```

- userSlice.js

```
import {createSlice} from "@reduxjs/toolkit"
export const userSlice = createSlice({
    initialState:{
        name:"",
        email:"",
        loggedIn:false,
        registerNumber:"",
        UserType:""
    },
    name:"User",
    reducers:{
        signin:(state)=>{
            state.loggedIn=true;
        },
```

```
        setdetails:(state,action)=>{
            state.name=action.payload.name,
            state.email=action.payload.email,
            state.registerNumber=action.payload.registerNumber,
            state.UserType=action.payload.UserType
        },
        logout:(state)=>{
            state.name="",
            state.email="",
            state.registerNumber="",
            state.UserType="",
            state.loggedIn=false
        }

    }
})

export const {signin,setdetails,logout} = userSlice.actions;
export default userSlice.reducer;
```

- FacultyPage.js

```
import React from 'react';
import Navbar from './Navbar';
import { useNavigate } from 'react-router-dom';
import { logout } from "../userSlice";
import { useSelector, useDispatch } from "react-redux";
import "./studentpage.css"; // Import the new CSS file

export default function StudentPage() {
  const dispatch = useDispatch();
  const navigate = useNavigate();

  const handleSignOut = () => {
    dispatch(logout());
    navigate("/");
  };

  const User = useSelector(state => state.user);

  return (
    <div className="StudentPageContainer">
      <div className="NavbarWrapper">
```

```
      <Navbar />
    </div>
    <div className="StudentContent">
      <div className='Studenttext'>
        <h1 className='Welcome'>Welcome</h1>
        <h1 className='Name'>{User.name}</h1>
        <h1 className='dashboardmsg'>to the Faculty Dashboard </h1>
        <p >You can update your internship details or upload your permission
form using the navbar above.</p>
        <button onClick={handleSignOut}>Sign Out</button>
      </div>
    </div>
  </div>
 );
}
```

- FilteredPage.js

```
import React, { useEffect, useState } from "react";
import Navbar from "./Navbar";
import axios from "axios";
import "./filtered.css"; // New layout file

export default function FilteredPage() {
 const [elements, setElements] = useState([]);
 const [filtered, setFiltered] = useState([]);
 const [filters, setFilters] = useState({
  company: "",
  placementType: "",
  researchIndustry: "",
  location: "",
 });

 useEffect(() => {
  const fetchData = async () => {
   try {
    const response = await axios.get("http://localhost:5000/getall");
    const padded = padArraysToLength(response.data.data.values);
    setElements(padded);
    setFiltered(padded);
   } catch (error) {
    console.log(error);
   }
```

```
    };
    fetchData();
  }, []);

  function padArraysToLength(arrays, targetLength = 15) {
    return arrays.map(arr => arr.concat(Array(targetLength -
arr.length).fill("")));
  }

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

  const applyFilters = () => {
    const [headers, ...rows] = elements;
    const filteredRows = rows.filter(row => {
      const company = row[10];
      const placement = row[11];
      const research = row[13];
      const location = row[14];
      return (
        (filters.company === "" ||
company.toLowerCase().includes(filters.company.toLowerCase())) &&
        (filters.placementType === "" || placement === filters.placementType)
&&
        (filters.researchIndustry === "" || research === filters.researchIndustry)
&&
        (filters.location === "" || location === filters.location)
      );
    });
    setFiltered([headers, ...filteredRows]);
  };

  return (
    <div className="FilteredPageContainer">
      <div className="NavbarWrapper">
        <Navbar />
      </div>
      <div className="FilteredContent">
        <div className="filter-section">
          <input
            type="text"
```

```
          name="company"
          placeholder="Company Name"
          value={filters.company}
          onChange={handleFilterChange}
        />
        <select name="placementType" value={filters.placementType}
onChange={handleFilterChange}>
          <option value="">Placement Type</option>
          <option value="College">College</option>
          <option value="Outside">Outside</option>
        </select>
        <select name="researchIndustry" value={filters.researchIndustry}
onChange={handleFilterChange}>
          <option value="">Research/Industry</option>
          <option value="Research">Research</option>
          <option value="Industry">Industry</option>
        </select>
        <select name="location" value={filters.location}
onChange={handleFilterChange}>
          <option value="">India/Abroad</option>
          <option value="India">India</option>
          <option value="Abroad">Abroad</option>
        </select>
        <button onClick={applyFilters}>Apply Filters</button>
      </div>

      <div className="table-container">
        {filtered.length > 0 ? (
          <table className="table">
            <thead>
              <tr>{filtered[0].map((head, i) => <th key={i}>{head}</th>)}</tr>
            </thead>
            <tbody>
              {filtered.slice(1).map((row, i) => (
                <tr key={i}>{row.map((val, j) => <td key={j}>{val}</td>)}</tr>
              ))}
            </tbody>
          </table>
        ) : (
          <p>Loading data...</p>
        )}
      </div>
    </div>
```

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

- Home.js

```
import React, { useEffect, useState } from "react";
import Navbar from "./Navbar";
import axios from "axios";
import "./home.css";

export default function Home() {
  const [elements, setElements] = useState([]);

  useEffect(() => {
    const fetchdata = async () => {
      try {
        const response = await axios.get("http://localhost:5000/get");
        const padded = padSubarrays(response.data.data.values)
        setElements(padded);
      } catch (error) {
        console.log(error);
      }
    };
    fetchdata();
  }, []);

  function padSubarrays(arr) {
    return arr.map(sub => {
        if (sub.length === 3 || sub[3] === "No") {
            return [...sub.slice(0, 3), "No internship"];
        }
        return sub;
    });
}

  return (
    <div>
      <Navbar />
      <div>
        {elements.length > 0 ? (
          <table className="table">
```

```
          <thead>
            <tr>
              {elements[0].map((header, index) => (
                <th key={index}>{header}</th>
              ))}
            </tr>
          </thead>
          <tbody>
            {elements.slice(1).map((row, rowIndex) => (
              <tr key={rowIndex} className="border">
                {row.map((value, colIndex) => (
                  <td key={colIndex}>{value}</td>
                ))}
              </tr>
            ))}
          </tbody>
        </table>
      ) : (
        <p>Loading data...</p>
      )}
    </div>
  </div>
 );
}
```

- Navbar.js

```
import React from 'react';
import "./layout.css";
import { Link } from "react-router-dom";
import logo from "../assets/nobglogo.png";
import { useSelector } from "react-redux"
export default function Navbar() {
 const User = useSelector(state=>state.user)
 const role=User.UserType;
 return (

  <div className='Navbar'>
   <Link to={role === "Teacher" ? "/faculty" : "/student"}>
    <img className='logoimag' src={logo} alt='SSN LOGO' />
   </Link>
   <div className='NavtextSection'>
    {role === "Teacher" ? (
```

```
        <>
          <Link to="/faculty" className='Navtext'>Home</Link>
          <Link to="/view" className='Navtext'>View</Link>
          <Link to="/filter" className='Navtext'>Filter</Link>
        </>
      ) : (
        <>
          <Link to="/student" className='Navtext'>Home</Link>
          <Link to="/update" className='Navtext'>Update</Link>
          <Link to="/upload" className='Navtext'>Upload</Link>
        </>
      )}
    </div>
  </div>
 );
}
```

- StudentPage.js

```
import React from 'react';
import Navbar from './Navbar';
import { useNavigate } from 'react-router-dom';
import { logout } from "../userSlice";
import { useSelector, useDispatch } from "react-redux";
import "./studentpage.css"; // Import the new CSS file

export default function StudentPage() {
 const dispatch = useDispatch();
 const navigate = useNavigate();

 const handleSignOut = () => {
  dispatch(logout());
  navigate("/");
 };

 const User = useSelector(state => state.user);

 return (
  <div className="StudentPageContainer">
   <div className="NavbarWrapper">
    <Navbar />
   </div>
   <div className="StudentContent">
```

```
        <div className='Studenttext'>
          <h1 className='Welcome'>Welcome</h1>
          <h1 className='Name'>{User.name}</h1>
          <h1 className='dashboardmsg'>to the Student Dashboard </h1>
          <p >You can update your internship details or upload your permission
form using the navbar above.</p>
          <button onClick={handleSignOut}>Sign Out</button>
        </div>
      </div>
    </div>
  );
}
```

- Update.js

```
import React, { useState } from 'react';
import Navbar from "./Navbar";
import axios from 'axios';
import { useSelector } from "react-redux";
import "./update.css"; // Import the new CSS file

export default function Update() {
  const User = useSelector(state => state.user);
  const [formData, setFormData] = useState({
    registerNumber: User.registerNumber, name: User.name, title: '', mobileNo:
'',
    startDate: '', endDate: '', companyName: '', placementType: '',
    stipend: '', researchIndustry: ''
  });

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

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await axios.post("http://localhost:5000/update", formData);
      if (res.data.message === "yes") {
        alert("Yes: Internship details updated successfully");
      } else {
        alert("No: Failed to update details");
      }
```

```
    } catch (error) {
      console.error(error);
      alert("No: Something went wrong");
    }
  };

  return (
    <div className="UpdatePageContainer">
      <div className="NavbarWrapper">
        <Navbar />
      </div>
      <div className="UpdateContent">
        <div className="UpdateFormWrapper">
          <h2>Update Internship Details</h2>
          <form onSubmit={handleSubmit}>
            {Object.entries(formData).map(([key, value]) => (
              key !== "placementType" && key !== "researchIndustry" ? (
                <div key={key}>
                  <label>{key.replace(/([A-Z])/g, ' $1')}</label>
                  <input
                    type={key.toLowerCase().includes("date") ? "date" : key ===
"stipend" ? "number" : "text"}
                    name={key}
                    value={value}
                    onChange={handleChange}
                    readOnly={key === "registerNumber" || key === "name"}
                    required
                  />
                </div>
              ) : null
            ))}
            <div>
              <label>Placement thru college / outside</label>
              <select name="placementType" value={formData.placementType}
onChange={handleChange} required>
                <option value="">Select</option>
                <option value="College">College</option>
                <option value="Outside">Outside</option>
              </select>
            </div>
            <div>
              <label>Research / Industry</label>
```

```
            <select name="researchIndustry"
value={formData.researchIndustry} onChange={handleChange} required>
              <option value="">Select</option>
              <option value="Research">Research</option>
              <option value="Industry">Industry</option>
            </select>
          </div>
          <button type="submit">Submit</button>
        </form>
      </div>
    </div>
  </div>
 );
}
```

- Upload.js

```
import { useState } from "react";
import "./upload.css"; // Import the new CSS file
import Navbar from "./Navbar";
import axios from "axios";

const Upload = () => {
 const [file, setFile] = useState(null);
 const [extractedData, setExtractedData] = useState(null);
 const [uploading, setUploading] = useState(false);

 const handleFileChange = (e) => setFile(e.target.files[0]);

 const handleUpload = async () => {
  if (!file) return alert("Please select a file");
  const formData = new FormData();
  formData.append("file", file);
  setUploading(true);
  try {
   const res = await fetch("http://127.0.0.1:5001/upload", {
    method: "POST",
    body: formData,
   });
   if (!res.ok) throw new Error("Upload failed");
   const data = await res.json();
   setExtractedData(data);
  } catch (err) {
```

```
      alert("Upload failed");
    } finally {
      setUploading(false);
    }
  };

  const handleVerify = async () => {
    if (!file) return alert("Upload a file first");
    const formData = new FormData();
    formData.append("file", file);
    try {
      const res = await axios.post("http://localhost:5000/verify", formData);
      alert(res.data.message);
    } catch {
      alert("Verification failed");
    }
  };

  return (
    <div className="UploadPageContainer">
      <div className="NavbarWrapper">
        <Navbar />
      </div>
      <div className="UploadContent">
        <div className="UploadFormWrapper">
          <h2>Upload Permission Form</h2>
          <label className="choose-button">
            Choose File
            <input type="file" accept="application/pdf"
onChange={handleFileChange} className="file-input" />
          </label>
          {file && <p className="file-name">{file.name}</p>}
          <button onClick={handleUpload} disabled={uploading}
className="upload-button">
            {uploading ? "Uploading..." : "Upload"}
          </button>

          {extractedData && (
            <div className="extracted-data">
              <h3>Extracted Information</h3>
              <table className="table">
                <thead>
                  <tr><th>Field</th><th>Value</th></tr>
```

```
            </thead>
            <tbody>
              {Object.entries(extractedData).map(([key, val]) => (
                <tr key={key}><td>{key}</td><td>{String(val)}</td></tr>
              ))}
            </tbody>
          </table>
          <button className="upload-drive-button"
onClick={handleVerify}>Upload to Drive</button>
        </div>
      )}
    </div>
  </div>
 );
};

export default Upload;
```

- View.js

```
import React, { useState, useEffect } from 'react';
import React, { useState, useEffect } from 'react';
import Navbar from "./Navbar";
import axios from 'axios';
import "./view.css";

export default function View() {
 const [registerNumber, setRegisterNumber] = useState('');
 const [person, setPerson] = useState([]);
 const [submitted, setSubmitted] = useState(false);
 const [trigger, setTrigger] = useState(0);

 useEffect(() => {
  axios.get("http://localhost:5000/getall")
   .then(res => {
    const allPersons = res.data.data.values;
    const match = allPersons.find(i => i[1] === registerNumber);
    setPerson(match || []);
   })
   .catch(console.error);
 }, [trigger]);
```

```
  const handleSubmit = (e) => {
    e.preventDefault();
    setSubmitted(true);
    setTrigger(prev => prev + 1);
  };

  return (
    <div className="ViewPageContainer">
      <div className="NavbarWrapper">
        <Navbar role="faculty" />
      </div>
      <div className="ViewContent">
        {!submitted ? (
          <div className="container">
            <h2>Enter Details</h2>
            <form className="form" onSubmit={handleSubmit}>
              <input
                type="text"
                placeholder="Register Number"
                value={registerNumber}
                onChange={e => setRegisterNumber(e.target.value)}
                required
                className="input-field"
              />
              <button type="submit" className="submit-
button">Submit</button>
            </form>
          </div>
        ) : (
          <div className="display">
            {person.length ? (
              <div className="table-container">
                <h2>Internship Details</h2>
                <table className="display-table">
                  <tbody>
                    <tr><td >Name</td><td>{person[2]}</td></tr>
                    <tr><td>Title</td><td>{person[3]}</td></tr>
                    <tr><td>Mobile No.</td><td>{person[4]}</td></tr>
                    <tr><td>Company</td><td>{person[10]}</td></tr>
                    <tr><td>Placement through</td><td>{person[11]}</td></tr>
                    <tr><td>Stipend</td><td>{person[12]}</td></tr>
                    <tr><td>Research/Industry</td><td>{person[13]}</td></tr>
                  </tbody>
```

```
            </table>
          </div>
        ) : (
          <p className="no-internship">No internship found</p>
        )}
      </div>
    )}
  </div>
 </div>
 );
}
```

- App.css

```
a {
  text-decoration: none;  /* Removes underline */
  color: inherit;         /* Uses the default text color */
 }

 a:hover {
  color: #000;  /* Optional: Set a custom color on hover */
 }


 .main-nav ul {
  display: flex;
  list-style: none;
  gap: 20px;
  padding: 10px;
  background-color: #f0f0f0;
 }
 .main-nav li a {
  text-decoration: none;
  font-weight: bold;
 }
```

- index.css

```
body {
 margin: 0;
 font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
```

```
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

- login.css

```
.main_container {
    display: flex;
    height: 100vh; /* Make sure it fills the full screen */
  }

.LoginSide {
  display: flex;
  flex: 1;
}

.LoginSide > div {
  flex: 1;
  display: flex;
  align-items: center;
  justify-content: center;
}
.loginform{
  display: flex;
  flex-direction: column;
}
.loginform > input{
  border: none;
  background-color: rgba(185, 237, 250,0.5);
  height:30px;
  width: 500px;
  margin: 10px;
  border-radius: 50px;
  padding:10px
}
.loginform > h2{
```

```
  align-self: center;
  font-size: xxx-large;
}
.SubmitBtn{
  background-color: #8BC6EC;
  background-image: linear-gradient(135deg, #8BC6EC 0%, #9599E2 100%);
  height:50px;
  width:100px;
  border: none;
  border-radius: 50px;
  color:white;
  font-weight: bold;
  align-self: center;
}
.otherside {
  color: white;
  background-color: #8BC6EC;
  background-image: linear-gradient(135deg, #8BC6EC 0%, #9599E2
100%);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  text-align: center;
  padding: 20px;

  flex: 1;
}
.otherside>h2{
  color: white;
  font-weight: lighter;
}
.otherside>h1{
  color: white;
  font-weight: bold;
  font-size: xxx-large;
}
.logoimg{
height: 200px;
width: 500px;
}
```

- filtered.css

```
.FilteredPageContainer {
  display: flex;
  height: 100vh;
  background: linear-gradient(to bottom right, #5f9ea0, #4f8a8b); /* Same
blue-green gradient */
}

.NavbarWrapper {
  flex: 0.05;
  height: 100%;
}

.FilteredContent {
  flex: 0.95;
  width: 100%;
  padding: 20px;
  overflow-y: auto;
  background: linear-gradient(135deg, #e0eafc, #cfdef3);
}

.filter-section {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
  margin-bottom: 20px;
}

.filter-section input,
.filter-section select {
  padding: 8px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 6px;
  min-width: 150px;
}

.filter-section button {
  padding: 10px 20px;
  background-color: #6495ed;
  border: none;
  color: white;
```

```
  font-size: 16px;
  border-radius: 6px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.filter-section button:hover {
  background-color: #4169e1;
}

.table {
  width: 100%;
  border-collapse: collapse;
  word-break: break-word;
}

.table th, .table td {
  padding: 8px;
  border: 1px solid #ddd;
  text-align: left;
  white-space: normal;
  word-wrap: break-word;
  max-width: 150px;
  font-size: 14px;
}

.table-container {
  width: 100%;
  overflow-x: hidden;
}
```

- home.css

```
.table {
    width: 100%;
    border-collapse: collapse;
    table-layout: auto; /* Adjust column sizes automatically */
  }

  .table th, .table td {
    padding: 8px;
    border: 1px solid #ddd;
    text-align: left;
```

```
  white-space: nowrap; /* Prevents text from breaking into multiple lines */
 }

 .table-container {
  width: 100%;
  overflow-x: auto; /* Enables horizontal scrolling if necessary */
 }
```

- layout.css

```
.Navbar {
 background-color: #2f3b3e; /* Dark teal for a more prominent sidebar */
 width: 220px; /* Adjusted width for sidebar appearance */
 height: 100vh; /* Full viewport height */
 display: flex;
 flex-direction: column;
 align-items: center;
 padding: 20px 10px;
 box-sizing: border-box;
}

.logoimag {
 height: 60px;
 margin-bottom: 40px;
}

.NavtextSection {
 display: flex;
 flex-direction: column;
 gap: 20px;
 width: 100%;
 align-items: center;
}

.Navtext {
 font-size: 20px;
 color: #b0c4de; /* Light steel blue for text */
 text-decoration: none;
 transition: color 0.3s ease;
}

.Navtext:hover {
 color: #ffffff; /* White for hover effect */
```

```
}

.home-content {
  margin-left: 220px; /* Push content to the right of navbar */
  padding: 20px;
}
```

- navbar.css

```
.Navbar{
    max-width: 100vw;
}
.logoimag{
    height: 10px;
    width: 10px;
}
```

- studentpage.css

```
.StudentPageContainer {
    display: flex;
    height: 100vh;
    background: linear-gradient(to bottom right, #5f9ea0, #4f8a8b); /* Blue-
green gradient */
}

.NavbarWrapper {
    flex: 0.05;
    height: 100%;
}

.StudentContent {
    flex: 0.95;
    width: 100%;
    padding: 20px;
    overflow-y: auto;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #8BC6EC;
    background-image: linear-gradient(135deg, #8BC6EC 0%, #9599E2
100%);
}
```

```
.Studenttext {
    display: flex;
    flex-direction: column;
    align-items: center;
    text-align: center;
    padding: 20px;
}

.Welcome {
    font-size: 60px;
    margin: 0;
    padding: 0px;
    color: #333333; /* Dark grey for main title text */
}

.Name {
    margin: 0px;
    padding: 0px;
    font-size: 45px;
    color: #2e3d49; /* Dark slate grey for name */
}

.dashboardmsg {
    margin: 0px;
    padding: 0px;
    font-size: 45px;
    color: #4a4a4a; /* Medium grey for subtitle */
}

.Studenttext > p {
    font-size: 30px;
    color: #333333; /* Dark grey for paragraph text */
}

.Studenttext > button {
    border: none;
    height: 50px;
    width: 100px;
    background-color: #6495ed; /* Cornflower blue for button */
    font-size: 20px;
    border-radius: 50px;
    color: white;
    cursor: pointer;
```

```
    transition: background-color 0.3s ease;
}

.Studenttext > button:hover {
    background-color: #4169e1; /* Royal blue for hover effect */
}
```

- update.css

```
.UpdatePageContainer {
    display: flex;
    height: 100vh;
    background: linear-gradient(to bottom right, #5f9ea0, #4f8a8b); /* Blue-
green gradient */

  }

  .NavbarWrapper {
    flex: 0.05;
    height: 100%;
  }

  .UpdateContent {
    flex: 0.95;
    width: 100%;
    padding: 20px;
    overflow-y: auto;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #8BC6EC;
    background-image: linear-gradient(135deg, #8BC6EC 0%, #9599E2
100%);

  }

  .UpdateFormWrapper {
    background-color: #ffffff;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    width: 100%;
    max-width: 800px;
```

```
  margin-top:300px;
}

.UpdateFormWrapper h2 {
  text-align: center;
  font-size: 36px;
  color: #333333;
}

.UpdateFormWrapper form {
  display: flex;
  flex-direction: column;
  gap: 20px;
  align-items: center;
  align-self:center ;
}

.UpdateFormWrapper input,
.UpdateFormWrapper select {
  padding: 12px;
  font-size: 18px;
  border: 1px solid #ccc;
  border-radius: 5px;
  width:500px
}

.UpdateFormWrapper button {
  padding: 15px;
  font-size: 20px;
  background-color: #6495ed;
  color: white;
  border: none;
  border-radius: 50px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.UpdateFormWrapper button:hover {
  background-color: #4169e1;
}
```

- upload.css

```
/* Upload container styling */
.UploadPageContainer {
  display: flex;
  height: 100vh;
  background: linear-gradient(to bottom right, #5f9ea0, #4f8a8b); /* Blue-green gradient */
}

.NavbarWrapper {
  flex: 0.05;
  height: 100%;
}

.UploadContent {
  flex: 0.95;
  width: 100%;
  padding: 20px;
  overflow-y: auto;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #8BC6EC;
  background-image: linear-gradient(135deg, #8BC6EC 0%, #9599E2 100%);
}

.UploadFormWrapper {
  background-color: #ffffff;
  padding: 40px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  width: 100%;
  max-width: 800px;
  text-align: center;
}

.UploadFormWrapper h2 {
  text-align: center;
  font-size: 36px;
  color: #333333;
}
```

```
.UploadFormWrapper .file-input,
.UploadFormWrapper .upload-button {
  width: 100%;
}

.UploadFormWrapper .upload-button {
  padding: 15px;
  font-size: 20px;
  background-color: #6495ed;
  color: white;
  border: none;
  border-radius: 50px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.UploadFormWrapper .upload-button:hover {
  background-color: #4169e1;
}

.UploadFormWrapper .file-name {
  font-size: 16px;
  color: #555;
}

.UploadFormWrapper .extracted-data {
  margin-top: 20px;
  text-align: left;
  background: #ffffff;
  padding: 15px;
  border-radius: 5px;
  box-shadow: 1px 1px 5px rgba(0, 0, 0, 0.1);
  max-height: 300px;
  overflow-y: auto;
  border: 1px solid #ddd;
}

.UploadFormWrapper .extracted-data button {
  display: block;
  margin: 10px auto;
  background-color: #6495ed;
  color: white;
  border: none;
```

```
  padding: 10px 15px;
  cursor: pointer;
  border-radius: 5px;
  font-size: 16px;
}

.UploadFormWrapper .extracted-data button:hover {
  background-color: #4169e1;
}
.UploadFormWrapper .table thead {
  background-color: #6495ed;
  color: white;
}

.UploadFormWrapper .table th {
  padding: 10px;
  text-align: left;
}

.UploadFormWrapper .choose-button {
  display: inline-block;
  padding: 15px;
  font-size: 20px;
  background-color: #6495ed;
  color: white;
  text-align: center;
  border: none;
  border-radius: 50px;
  cursor: pointer;
  transition: background-color 0.3s ease;
  margin-bottom: 15px;
  position: relative;
  overflow: hidden;


}

.UploadFormWrapper .choose-button:hover {
  background-color: #4169e1;
}

.UploadFormWrapper .file-input {
  position: absolute;
```

```
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  opacity: 0;
  cursor: pointer;
}
```

- view.css

```
.ViewPageContainer {
  display: flex;
  height: 100vh;
  background: linear-gradient(to bottom right, #5f9ea0, #4f8a8b); /* Same
blue-green gradient */
}

.NavbarWrapper {
  flex: 0.05;
  height: 100%;
}

.ViewContent {
  flex: 0.95;
  width: 100%;
  padding: 20px;
  overflow-y: auto;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #8BC6EC;
  background-image: linear-gradient(135deg, #8BC6EC 0%, #9599E2 100%);
}


  .container {
    max-width: 400px;
    margin: 50px auto;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
```

```
}

h2 {
  color: #333;
}


.form-group {
  margin-bottom: 15px;
  text-align: left;
}

label {
  display: block;
  margin-bottom: 5px;
  font-weight: bold;
}

.input-field {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 16px;
}

.submit-button {
  width: 100%;
  padding: 10px;
  background-color: #6495ed;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 16px;
}

.submit-button:hover {
  background-color: #4169e1;
}
.form{
  padding: 20px;
```

```
}

.display {
  max-width: 600px;
  margin: 50px auto;
  padding: 20px;
  background: #ffffff;
  border-radius: 10px;
  box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);
  text-align: center;
}

.table-container {
  width: 100%;
  overflow-x: auto;
}

.display-table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 10px;
}

.display-table th, .display-table td {
  border: 1px solid #ddd;
  padding: 10px;
  text-align: left;
}

.display-table th {
  background-color: #6495ed;
  color: white;
}

.display-table tr:nth-child(even) {
  background: #f2f2f2;
}

.display-table tr:hover {
  background: #ddd;
}

/* Styling for 'No internship obtained' message */
```

```
  .no-internship {
    font-size: 20px;
    font-weight: bold;
    color: #007bff; /* Same blue as table header */
    padding: 10px;
    border-radius: 5px;
  }
  .form-group input,
.form-group select {
  width: 100%;
  padding: 8px;
  box-sizing: border-box;
}
```

BACKEND:

- backend.js

```
import express from "express";
import cors from "cors";
import routes from "./routes/routes.js";
import mongoose from "mongoose";
const app = express();
app.use(cors({ origin: 'http://localhost:3000' })); // ✅ CORS applied first
app.use(express.json());
app.use("/", routes);

app.listen(5000,()=>{
    console.log("app is running at port 5000");
})
```

- abc.js

```
import express from "express";
import {google} from "googleapis"
import multer from "multer"
const routes = express.Router();
import path from "path";
import fs from "fs"
import axios from "axios"

const auth = new google.auth.GoogleAuth({
    keyFile:"credentials.json",
```

```
    scopes: [
      "https://www.googleapis.com/auth/drive",
      "https://www.googleapis.com/auth/drive.file",
      "https://www.googleapis.com/auth/drive.metadata.readonly",
      "https://www.googleapis.com/auth/drive.appdata"
    ]
});

const client = await auth.getClient();
const drive = google.drive({ version: "v3", auth: client });

async function getOrCreateFolder(parentFolderId, folderName) {
    try {
      // Step 1: Check if folder exists
      const response = await drive.files.list({
        q: `'${parentFolderId}' in parents and
mimeType='application/vnd.google-apps.folder' and name='${folderName}'
and trashed=false`,
        fields: "files(id, name)",
      });

      if (response.data.files.length > 0) {
        console.log(`Folder "${folderName}" exists with ID:
${response.data.files[0].id}`);
        return response.data.files[0].id;
      }

      // Step 2: Create folder if not found
      const fileMetadata = {
        name: folderName,
        mimeType: "application/vnd.google-apps.folder",
        parents: [parentFolderId],
      };

      const folder = await drive.files.create({
        resource: fileMetadata,
        fields: "id",
      });

      console.log(`Folder "${folderName}" created with ID:
${folder.data.id}`);
      return folder.data.id;
    } catch (error) {
```

```
        console.error("Error in getOrCreateFolder:", error);
        throw error;
    }
}

// Example usage
const parentFolderId = "14NNAgpMkqYSsMad2MXhhSOI2LZPIdWx_";
getOrCreateFolder(parentFolderId, "def").then((folderId) => {
    console.log("Final Folder ID:", folderId);
});
```

- credentials.json

```
{
  "type": "service_account",
  "project_id": "stellar-day-455414-f3",
  "private_key_id": "45d7113e2e9c52877549032ea8c26c7b2f631a39",
  "private_key": "-----BEGIN PRIVATE KEYY\n-----END PRIVATE KEY-----\n",
  "client_email": "testkk@stellar-day-455414-f3.iam.gserviceaccount.com",
  "client_id": "117196829601489056863",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/testkk%40stellar-day-455414-f3.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```

- UserModel.js

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
```

```
    required: true,
    unique: true
  },
  registerNumber: {
    type: String,
    required: true,
    unique: true
  },
  UserType: {
    type: String,
    required:true
  },
  password: {
    type: String,
    required: true
  }
}, {
  collection: 'Users'
});

export default mongoose.model('User', userSchema, 'Users');
```

- routes.js

```
import express from "express";
import {google} from "googleapis"
import multer from "multer"
const routes = express.Router();
import bcrypt from "bcrypt"
import fs from "fs"
import axios from "axios"
import {MongoClient} from "mongodb"


const spreadsheetId =
"1EIeDEO0kbJd36p3Xk0hhLNPjuC1j2Iv0Wey4qTwyrCg";
const auth = new google.auth.GoogleAuth({
    keyFile:"credentials.json",
    scopes: [
        "https://www.googleapis.com/auth/spreadsheets",
        "https://www.googleapis.com/auth/drive",
        "https://www.googleapis.com/auth/drive.file",
        "https://www.googleapis.com/auth/drive.metadata.readonly",
```

```
        "https://www.googleapis.com/auth/drive.appdata"
    ]
});

const client = await auth.getClient();

const sheets = google.sheets({version:"v4",auth:client});
const drive = google.drive({ version: "v3", auth: client });

async function getOrCreateFolder(parentFolderId, folderName) {
    try {
        const response = await drive.files.list({
            q: `'${parentFolderId}' in parents and
mimeType='application/vnd.google-apps.folder' and name='${folderName}'
and trashed=false`,
            fields: "files(id, name)",
        });

        if (response.data.files.length > 0) {
            console.log(`Folder "${folderName}" exists with ID:
${response.data.files[0].id}`);
            return response.data.files[0].id;
        }

        const fileMetadata = {
            name: folderName,
            mimeType: "application/vnd.google-apps.folder",
            parents: [parentFolderId],
        };

        const folder = await drive.files.create({
            resource: fileMetadata,
            fields: "id",
        });

        console.log(`Folder "${folderName}" created with ID:
${folder.data.id}`);
        return folder.data.id;
    } catch (error) {
        console.error("Error in getOrCreateFolder:", error);
        throw error;
    }
}
```

```
// Example usage
const parentFolderId = "14NNAgpMkqYSsMad2MXhhSOI2LZPIdWx_";

routes.get("/get",async (req,res)=>{
    const rows =await sheets.spreadsheets.values.get({
        auth,
        spreadsheetId,
        range:"s1!A:D"
    })
    res.status(200).json(rows)
})
routes.get("/getall",async (req,res)=>{
    const rows =await sheets.spreadsheets.values.get({
        auth,
        spreadsheetId,
        range:"s1!A:O"
    })
    res.status(200).json(rows)
})

routes.post("/update",async (req,res)=>{
    let db=[];
    const rows =await sheets.spreadsheets.values.get({
        auth,
        spreadsheetId,
        range:"s1!A:B"
    })
    let rowval = rows.data.values
    for(let i of rowval){
        if(i[1]===req.body.registerNumber){
            db=[...db,i[0]];
            break
        }
    }
    db=[...db,req.body.registerNumber,req.body.name,req.body.title,req.body.m
obileNo,"","","",req.body.startDate,req.body.endDate,req.body.companyName
,req.body.placementType,req.body.stipend,req.body.researchIndustry];

        await sheets.spreadsheets.values.update({
        spreadsheetId,
        range:`s1!A${parseInt(db[0])+1}:N${parseInt(db[0])+1}`,
        valueInputOption:"RAW",
```

```
      resource:{values:[db]}
    })
    res.status(200).json({ message: "yes" });


})
// Set up multer for   file upload
const upload = multer({ dest: "uploads/" }); // Store uploaded files
temporarily

routes.post("/verify", upload.single("file"), async (req, res) => {
  try {
    if (!req.file) {
      return res.status(400).json({ error: "No file uploaded" });
    }

    // Step 1: Get extracted data from Flask server
    const resp = await axios.get("http://127.0.0.1:5001/extracteddata");

    // Step 2: Check data in the spreadsheet
    const rows = await sheets.spreadsheets.values.get({
      auth,
      spreadsheetId,
      range: "s1!A:T",
    });

    let found = false;
    let folderId = null;
    for (let i of rows.data.values) {
      if (
        i[1] === resp.data["Register Number"] &&
        i[4] === resp.data["Mobile Number"] &&
        resp.data["Company"].includes(i[10])
      ) {
        found = true;

        // Step 3: Get or create the folder
        const folderName = `${resp.data["Register
Number"]}_${resp.data.Name}`;
        folderId = await getOrCreateFolder(parentFolderId, folderName);

        break;
      }
```

```
    }

    if (!found) {
      return res.json({ message: "no" });
    }

    // Step 4: Upload file to Google Drive folder
    const fileMetadata = {
      name: `${resp.data["Register Number"].slice(9)}-Permission Letter.pdf`,
      parents: [folderId],
    };
    const media = {
      mimeType: req.file.mimetype,
      body: fs.createReadStream(req.file.path),
    };

    const fileUpload = await drive.files.create({
      resource: fileMetadata,
      media: media,
      fields: "id",
    });

    fs.unlinkSync(req.file.path);

    return res.json({
      message: "yes",
      folderId: folderId,
      fileId: fileUpload.data.id,
    });

  } catch (error) {
    console.error("Error:", error.message);
    res.status(500).json({ error: "Failed to process verification and file upload"
});
  }
});


async function fetchData(email) {
  const url =
'mongodb+srv://nikilesh:nikilesh@projcluster.78ch9xt.mongodb.net/?retryWri
tes=true&w=majority&appName=Projcluster';  // MongoDB URL
  const dbName = 'Ip';        // Your database name
```

```
  const client = new MongoClient(url);

  try {
     await client.connect();
     console.log('Connected to database');

     const db = client.db(dbName);
     const collection = db.collection('Users'); // Your collection name


     let result = await collection.findOne({ email: String(email) });
     return result;
  } catch (err) {
     console.error(err);
     return null;
  } finally {
     await client.close();
  }
}

routes.post("/login", async (req, res) => {

  const { email, password } = req.body;
  fetchData(email).then((result)=>{
    bcrypt.compare(password,result.password).then((match)=>{
      if (match){
        return
res.status(200).json({name:result.name,registerNumber:result.registerNumber
,UserType:result.UserType,success:true})
      }
      else{
        res.status(200).json({success:false})
      }
    })

  })

});


export default routes
```

PYTHON FILE FOR EXTRACTING TEXT FROM PDF

- backend.py

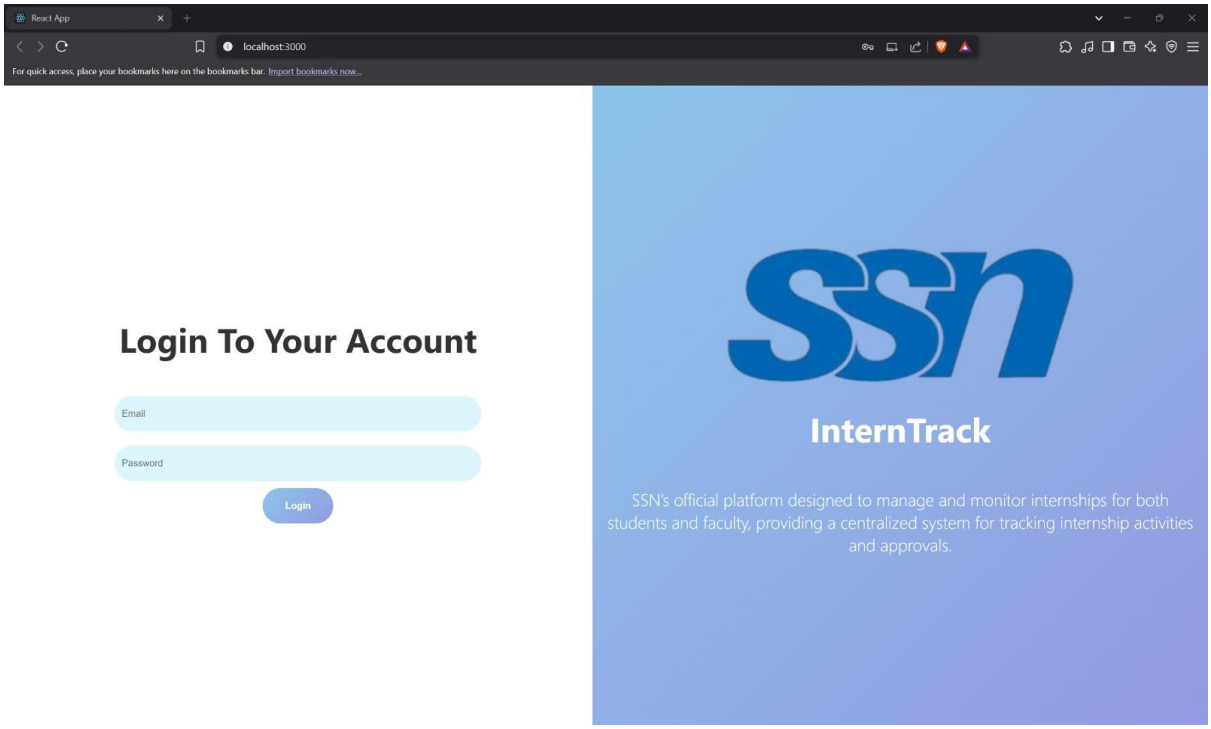
```
import pytesseract
import re
import cv2
import numpy as np
from flask import Flask, request, jsonify
from pdf2image import convert_from_bytes
from PIL import Image
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

def preprocess_image(img):
    """Convert image to grayscale, apply thresholding, and denoise."""
    img = cv2.cvtColor(np.array(img), cv2.COLOR_RGB2GRAY)
    img = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)[1]
    return Image.fromarray(img)

extracted_info={}

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

    file = request.files["file"]
    if file.filename == "":
        return jsonify({"error": "No selected file"}), 400

    # Read the file from memory (without saving)
    file_bytes = file.read()

    # Convert PDF bytes to images
    images = convert_from_bytes(file_bytes, dpi=300)

    # Extract text from images
    extracted_text = []
    for img in images:
        img = preprocess_image(img)  # Apply preprocessing
```

```
        text = pytesseract.image_to_string(img, config="--psm 6")
        extracted_text.append(text)

    # Combine text from all pages
    full_text = "\n".join(extracted_text)

    # Define patterns to extract required fields
    patterns = {
        "Name": r"Name\s*:\s*(.+)",
        "Register Number": r"Register Number\s*:\s*(\d+)",
        "CGPA": r"CGPA\s*:\s*([\d.]+)",
        "Mobile Number": r"Mobile Number\s*:\s*(\d{10})",
        "Email ID": r"Email ID\s*:\s*([\w\.-]+@[\w\.-]+)",
        "Company": r"Name of the Company / Institution\s*:\s*(.+)",
        "Internship Start Date": r"Internship start date\s*:\s*([\d./-]+)",
        "Internship End Date": r"Internship end date\s*:\s*([\d./-]+)"
    }

    for key, pattern in patterns.items():
        match = re.search(pattern, full_text, re.IGNORECASE)
        extracted_info[key] = match.group(1).strip() if match else "Not found"

    print(extracted_info)

    return jsonify(extracted_info)

@app.route("/extracteddata", methods=["GET"])
def dispatchdata():
    print(extracted_info)
    if extracted_info=={}:
        return jsonify({"message":"no data"})
    else:
        return jsonify(extracted_info)

if __name__ == "__main__":
    app.run(debug=True,port=5001)
```

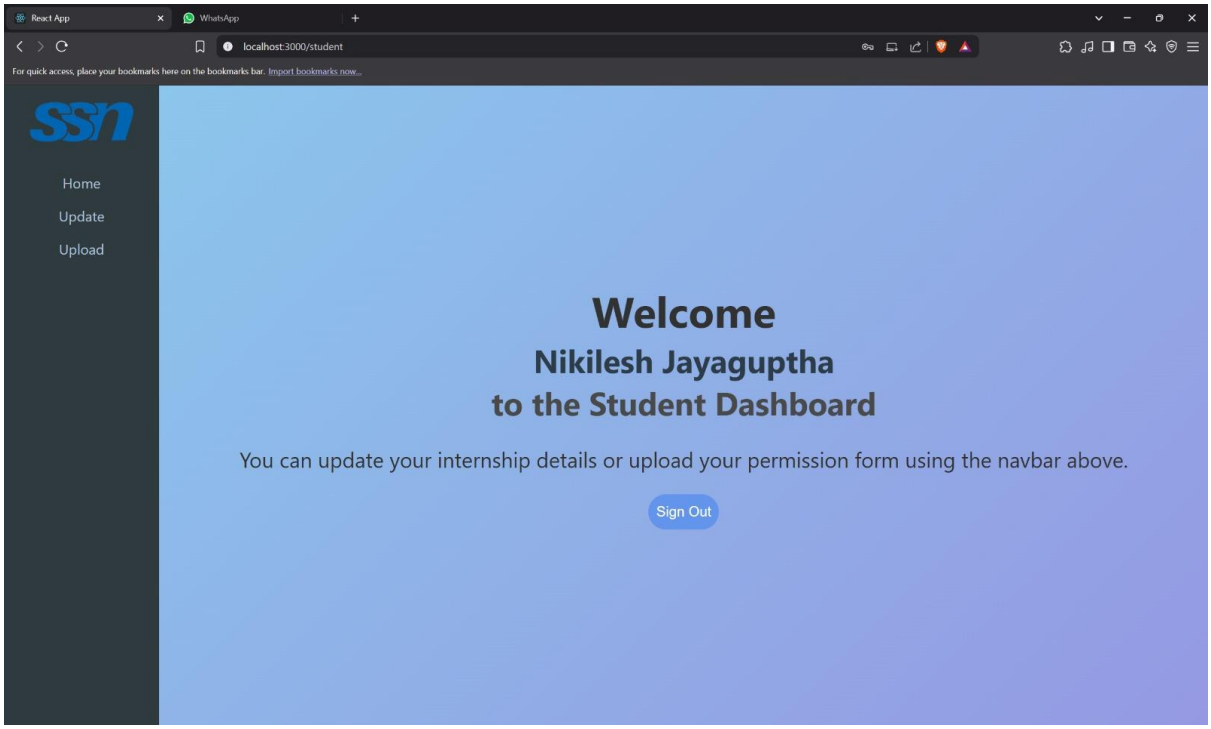
7. OUTPUT SCREENSHOTS

STUDENT SIDE:

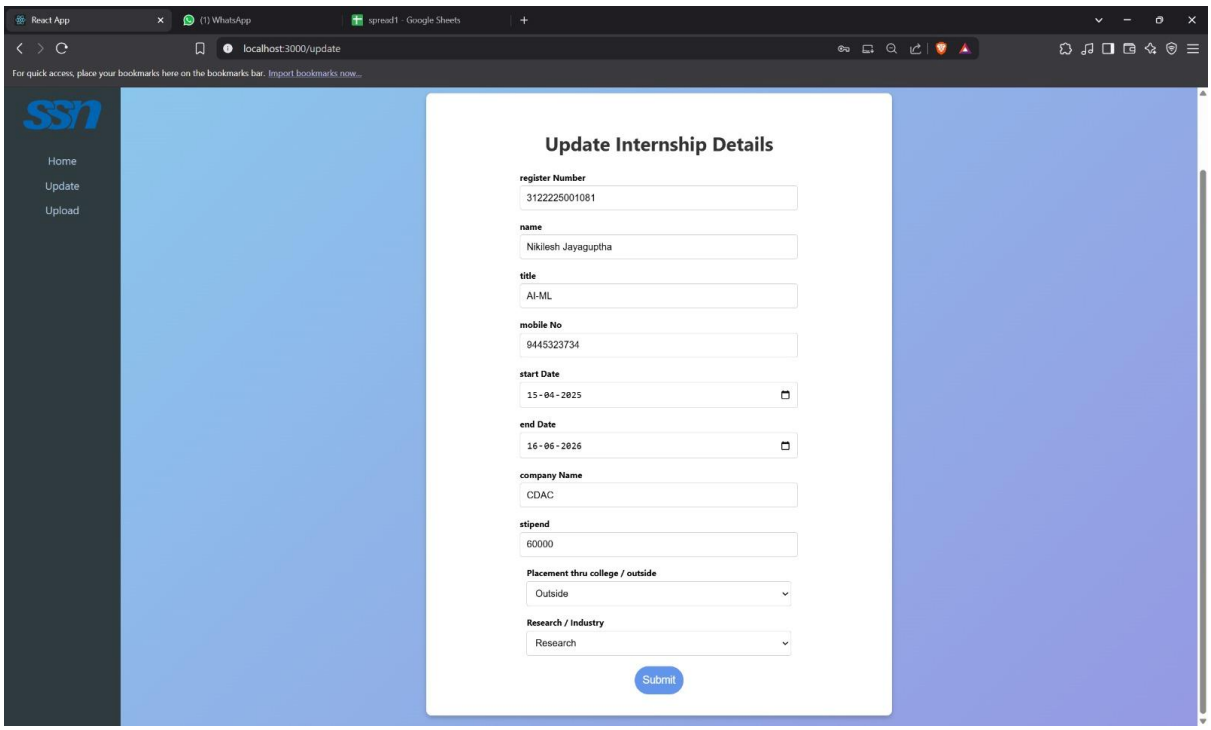
1) SSN Portal Login Page



2) Login into the student dashboard



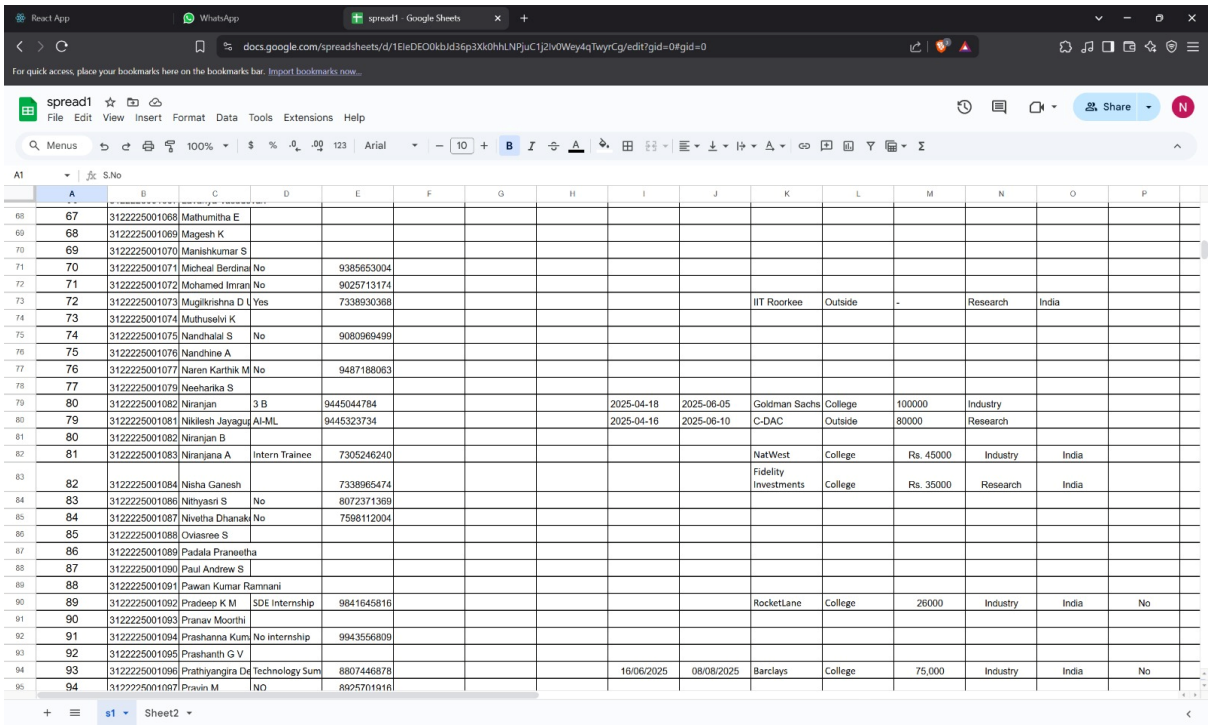
3) Updating the Internship Details; The Internship details available in the spreadsheet are updated



After clicking the Submit button, all the details in the spreadsheet corresponding to the particular student are updated

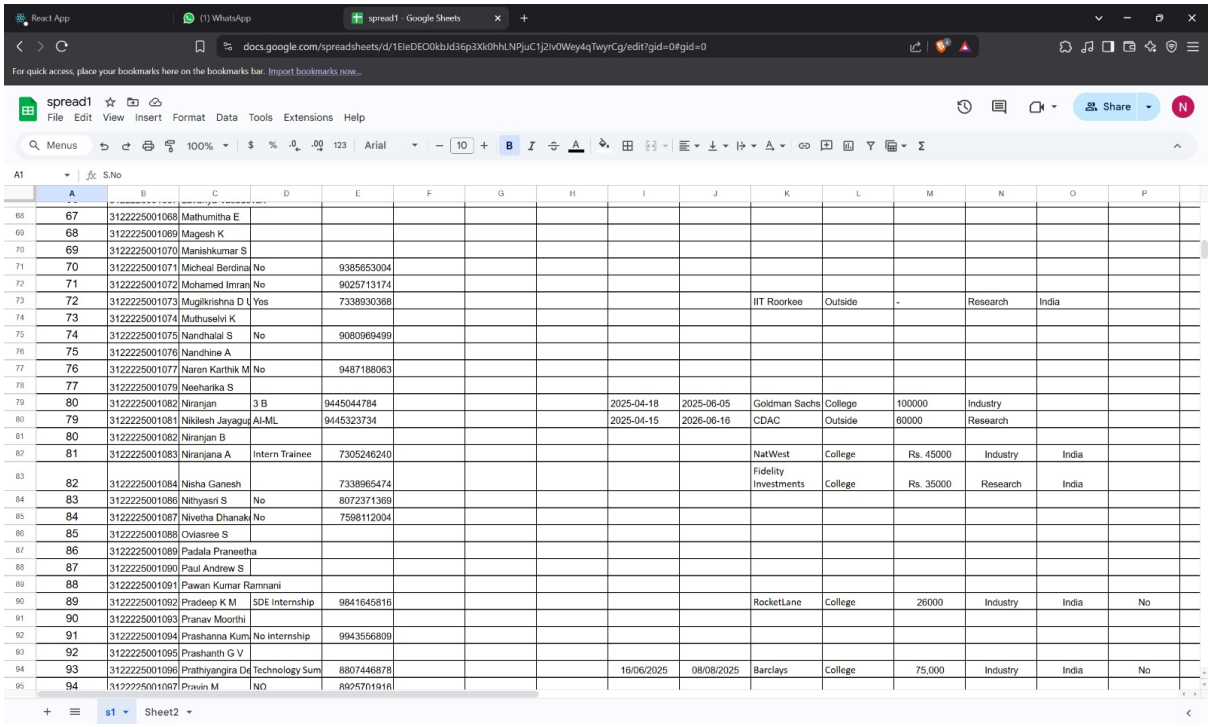
Changes in the spreadsheet before and after update:

<u>Before:</u>



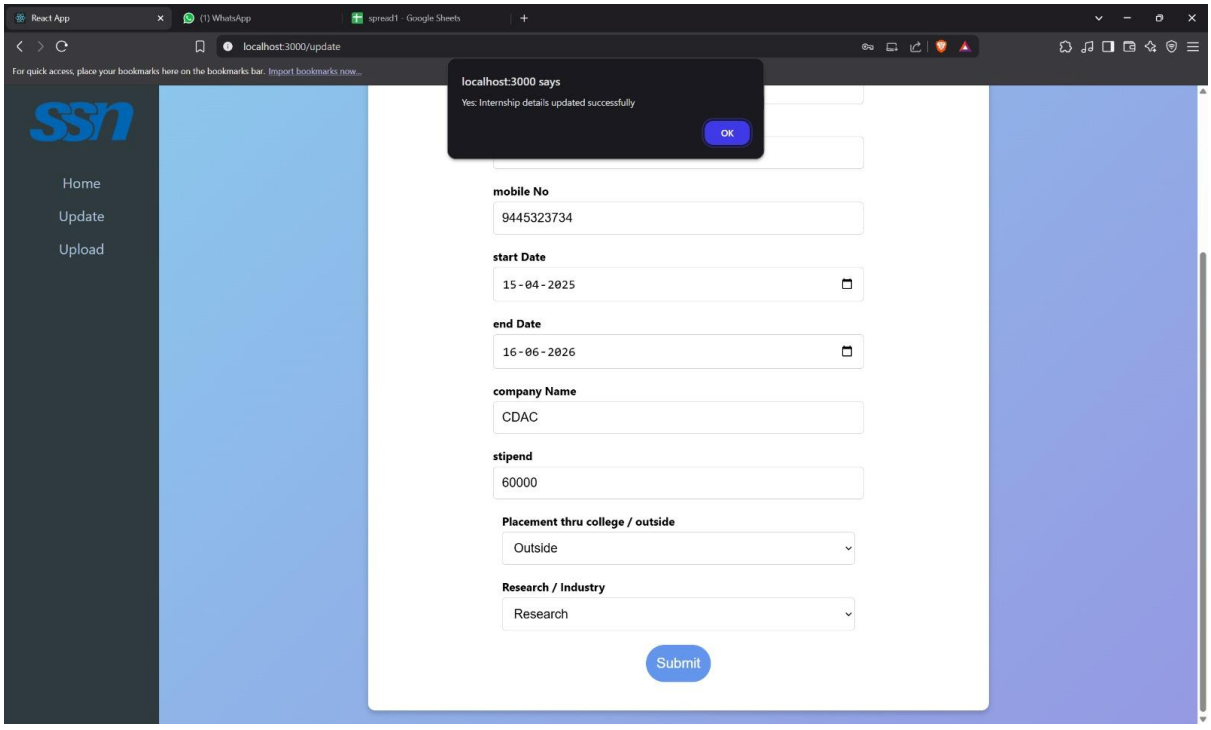
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
68	67	3122225001068	Mathumitha E													
69	68	3122225001069	Magesh K													
70	69	3122225001070	Manishkumar S													
71	70	3122225001071	Micheal Berdina	No	9385653004											
72	71	3122225001072	Mohamed Imran	No	9025713174											
73	72	3122225001073	Mugilkrishna D L	Yes	7338930368						IIT Roorkee	Outside	-	Research	India	
74	73	3122225001074	Muthuselvi K													
75	74	3122225001075	Nandhalal S	No	9080969499											
76	75	3122225001076	Nandhine A													
77	76	3122225001077	Naren Karthik M	No	9487188083											
78	77	3122225001079	Neeharika S													
79	80	3122225001082	Niranjan	3 B	9445044784				2025-04-18	2025-06-05	Goldman Sachs	College	100000	Industry		
80	79	3122225001081	Nikilesh Jayagu	AI-ML	9445323734				2025-04-16	2025-06-10	C-DAC	Outside	80000	Research		
81	80	3122225001082	Niranjan B													
82	81	3122225001083	Niranjana A	Intern Trainee	7305246240						NatWest	College	Rs. 45000	Industry	India	
83	82	3122225001084	Nisha Ganesh		7338965474						Fidelity Investments	College	Rs. 35000	Research	India	
84	83	3122225001086	Nithyasri S	No	8072371369											
85	84	3122225001087	Nivetha Dhanak	No	7598112004											
86	85	3122225001088	Oviasree S													
87	86	3122225001089	Padala Praneetha													
88	87	3122225001090	Paul Andrew S													
89	88	3122225001091	Pawan Kumar Ramnani													
90	89	3122225001092	Pradeep K M	SDE Internship	9841645816						RocketLane	College	26000	Industry	India	No
91	90	3122225001093	Pranav Moorthi													
92	91	3122225001094	Prashanna Kum	No internship	9943556809											
93	92	3122225001095	Prashanth G V													
94	93	3122225001096	Prathiyangira De	Technology Sum	8807446878				16/06/2025	08/08/2025	Barclays	College	75,000	Industry	India	No
95	94	3122225001097	Pravin M	NO	8925701916											

After:

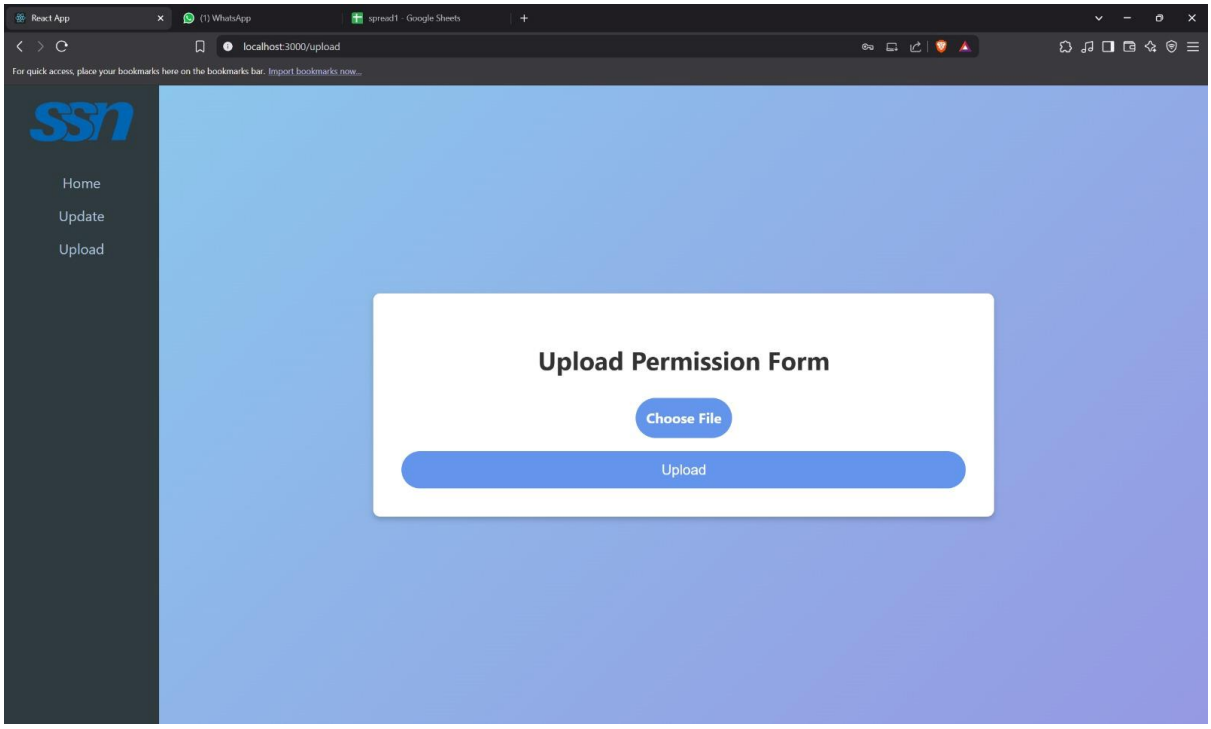


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
68	67	3122225001068	Mathumitha E													
69	68	3122225001069	Magesh K													
70	69	3122225001070	Manishkumar S													
71	70	3122225001071	Micheal Berdina	No	9385653004											
72	71	3122225001072	Mohamed Imran	No	9025713174											
73	72	3122225001073	Mugilkrishna D L	Yes	7338930368						IIT Roorkee	Outside	-	Research	India	
74	73	3122225001074	Muthuselvi K													
75	74	3122225001075	Nandhalal S	No	9080969499											
76	75	3122225001076	Nandhine A													
77	76	3122225001077	Naren Karthik M	No	9487188063											
78	77	3122225001079	Neeharika S													
79	80	3122225001082	Niranjan	3 B	9445044784				2025-04-18	2025-06-05	Goldman Sachs	College	100000	Industry		
80	79	3122225001081	Nikilesh Jayagug	AI-ML	9445323734				2025-04-15	2026-06-16	CDAC	Outside	60000	Research		
81	80	3122225001082	Niranjan B													
82	81	3122225001083	Niranjana A	Intern Trainee	7305246240						NatWest	College	Rs. 45000	Industry	India	
83	82	3122225001084	Nisha Ganesh		7338965474						Fidelity Investments	College	Rs. 35000	Research	India	
84	83	3122225001086	Nithyasri S	No	8072371369											
85	84	3122225001087	Nivetha Dhanak	No	7598112004											
86	85	3122225001088	Oviasree S													
87	86	3122225001089	Padala Praneetha													
88	87	3122225001090	Paul Andrew S													
89	88	3122225001091	Pawan Kumar Ramnani													
90	89	3122225001092	Pradeep K M	SDE Internship	9841645816						RocketLane	College	26000	Industry	India	No
91	90	3122225001093	Pranav Moorthi													
92	91	3122225001094	Prashanna Kum	No internship	9943556809											
93	92	3122225001095	Prashanth G V													
94	93	3122225001096	Prathiyangira De	Technology Sum	8807446878				16/06/2025	08/08/2025	Barclays	College	75,000	Industry	India	No
95	94	3122225001097	Pravin M	NO	8925701916											

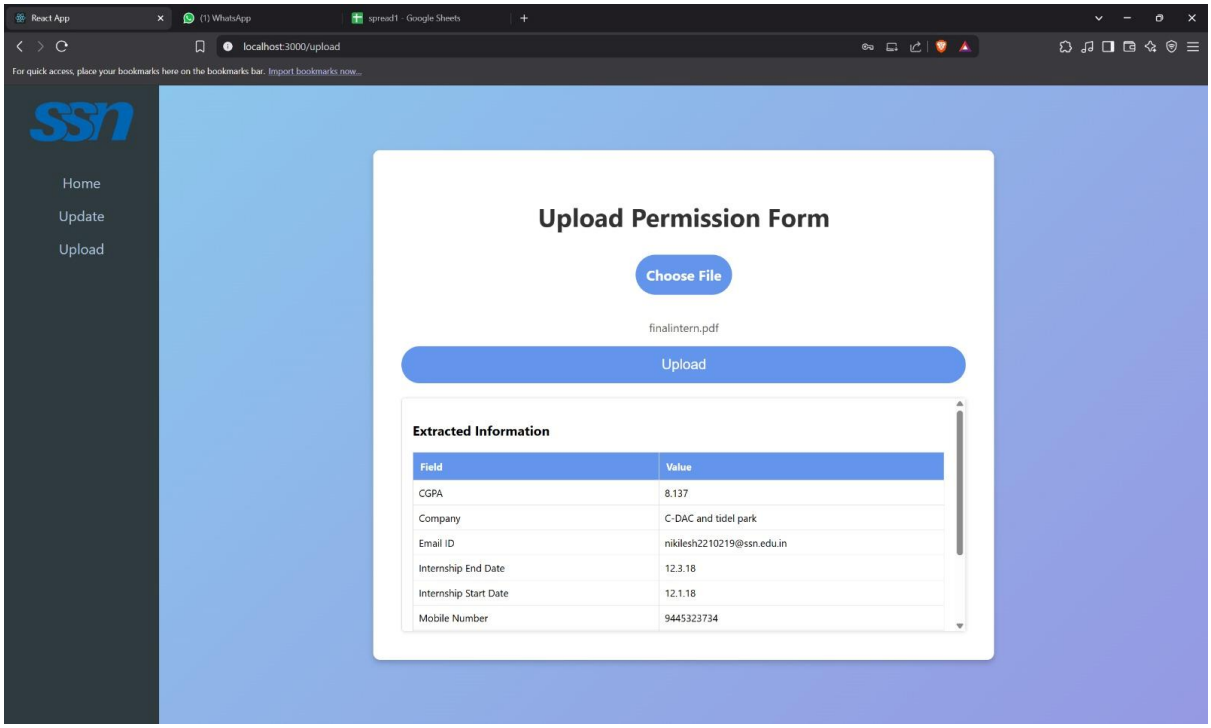
Once the details are successfully updated in the spreadsheet, it displays an aler box saying “Internship Details updated successfully”



4) Uploading the offer letter



Uploading finalintern.pdf to the app by clicking on “Choose File” and then upload

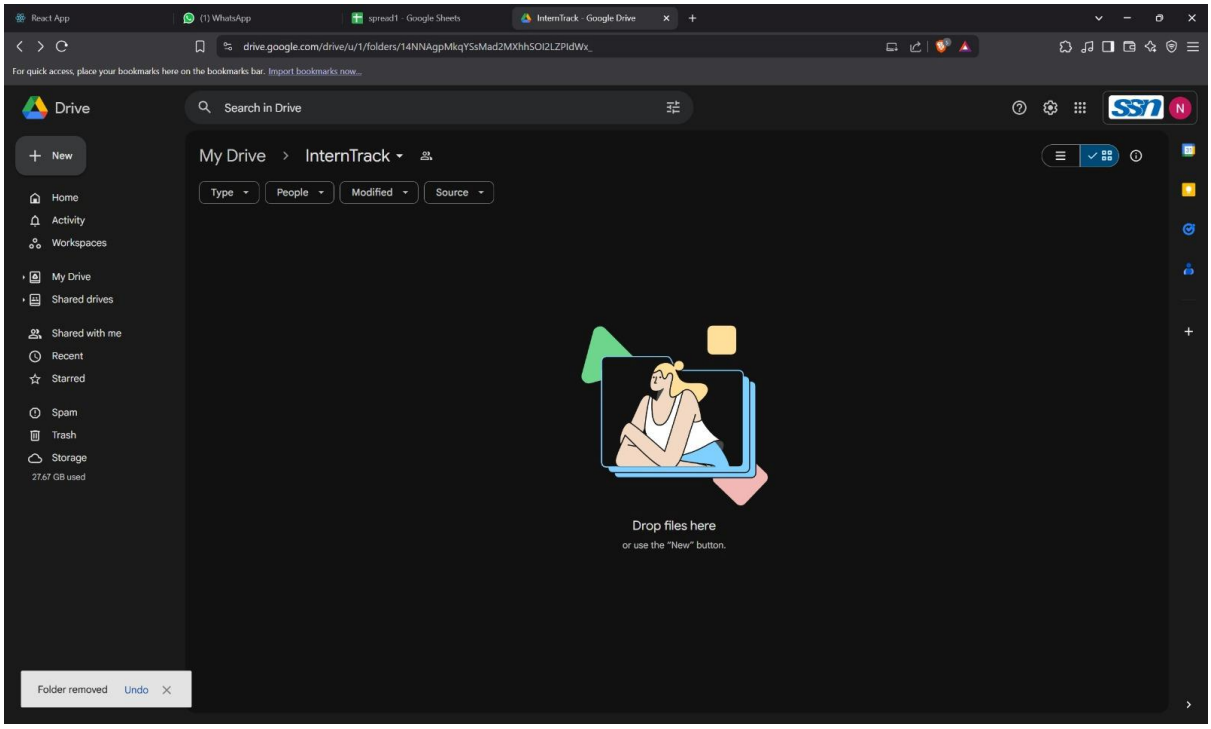


Once the uploading is done, the extracted information from the pdf is displayed.

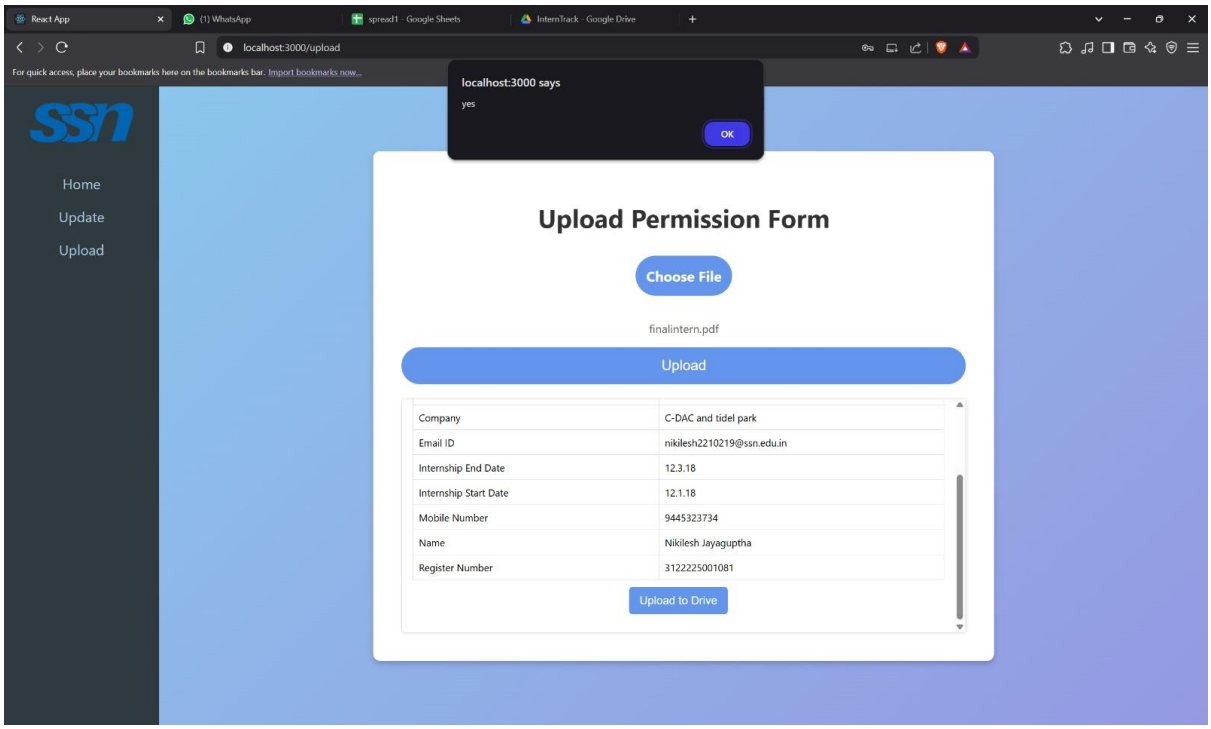
5) Changes in Google Drive (which stores the uploaded offer letter PDFs)

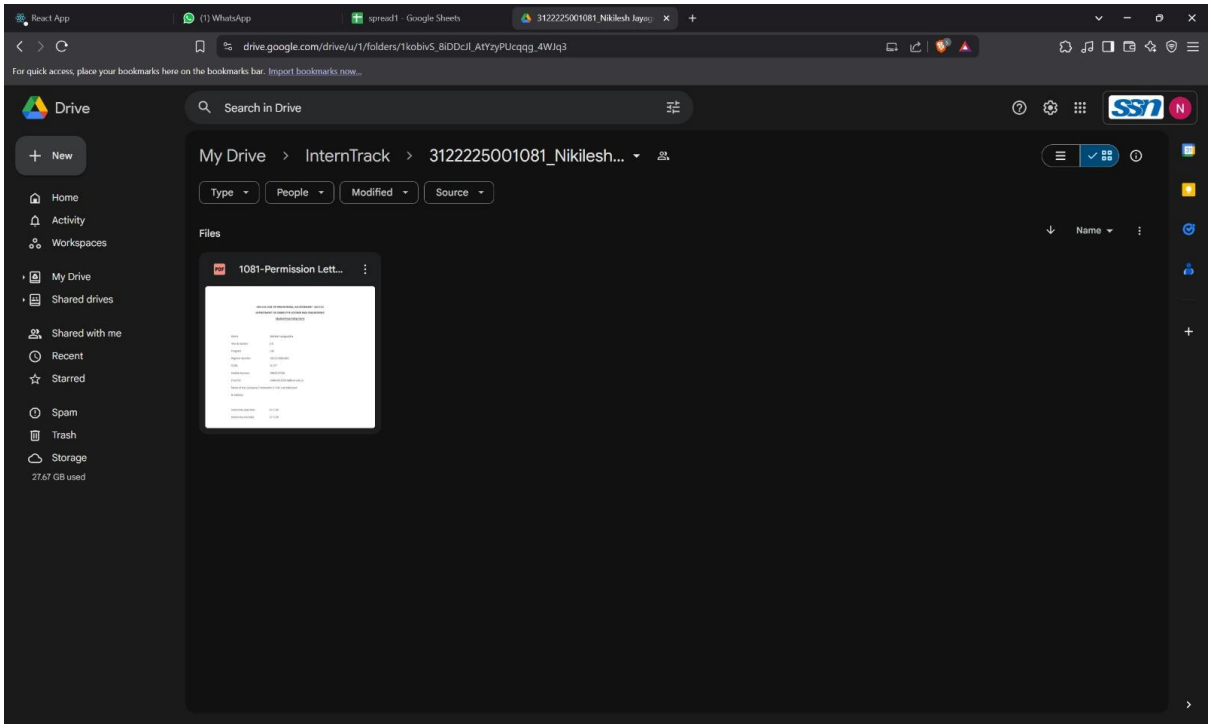
Choose the file you want to upload and then click the upload button; the offer letter gets stored in the Google Drive

Before Uploading: (Google Drive is Empty)



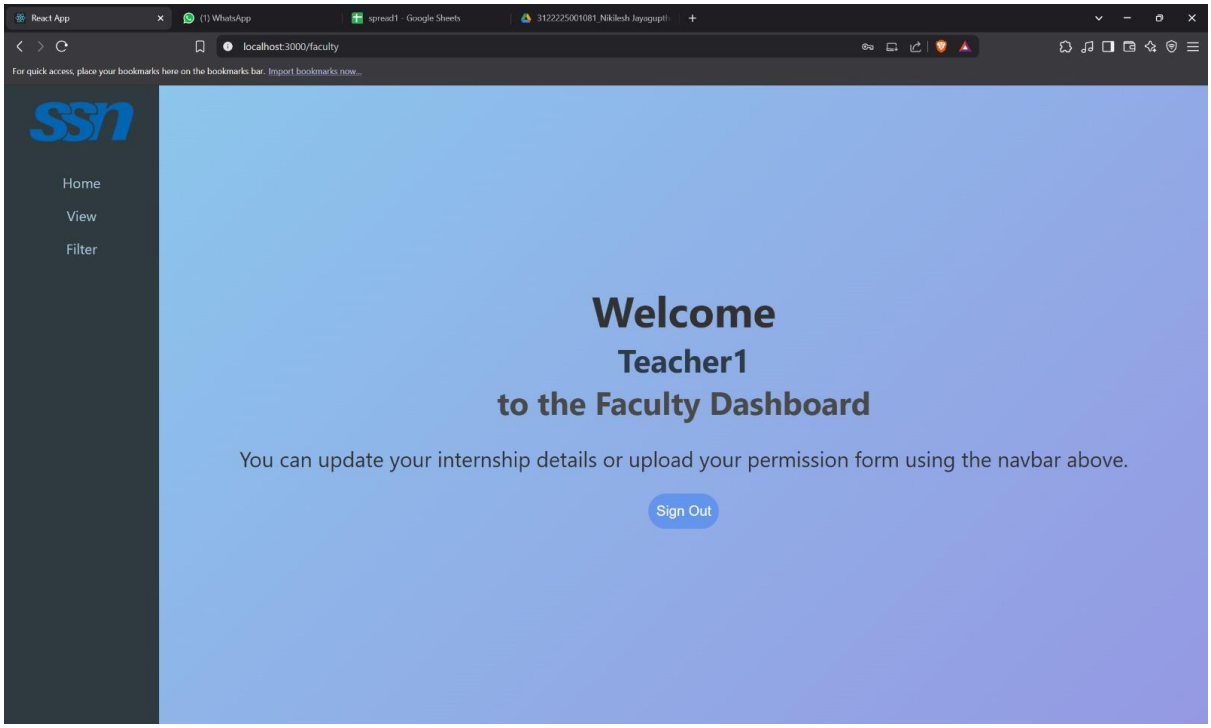
After Uploading:



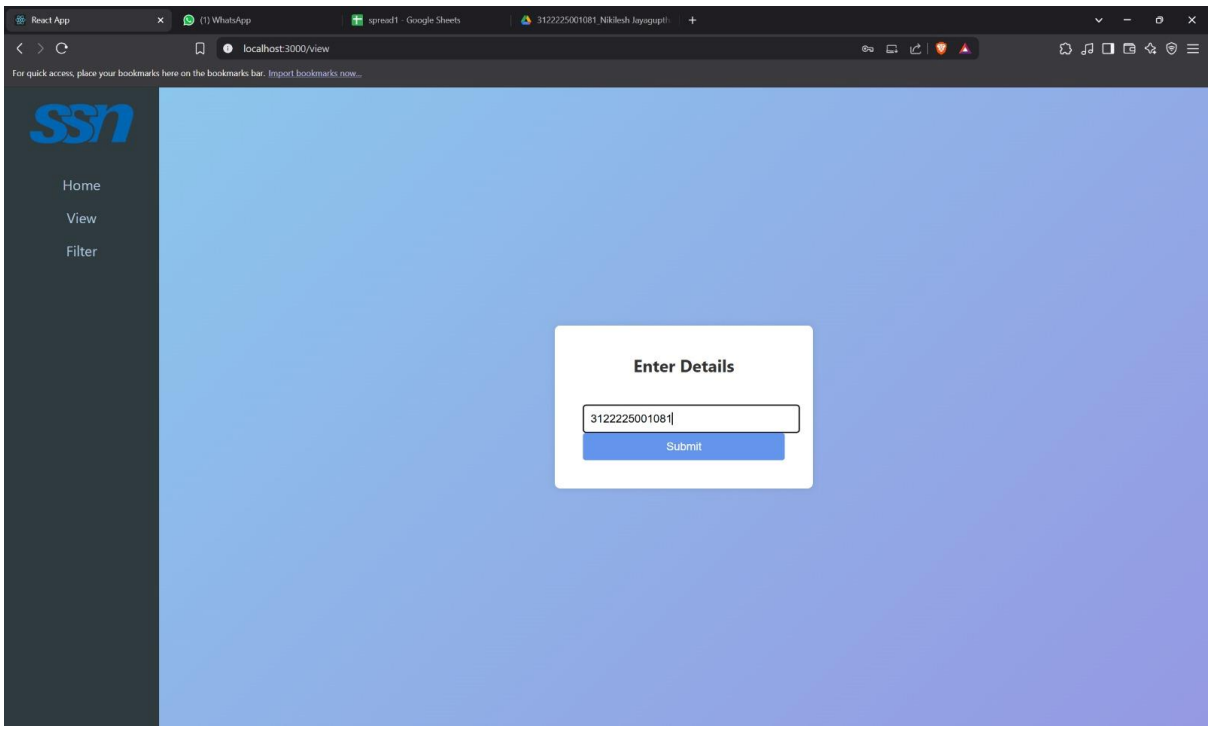


FACULTY SIDE:

6) Faculty Dashboard (Login with faculty Email)

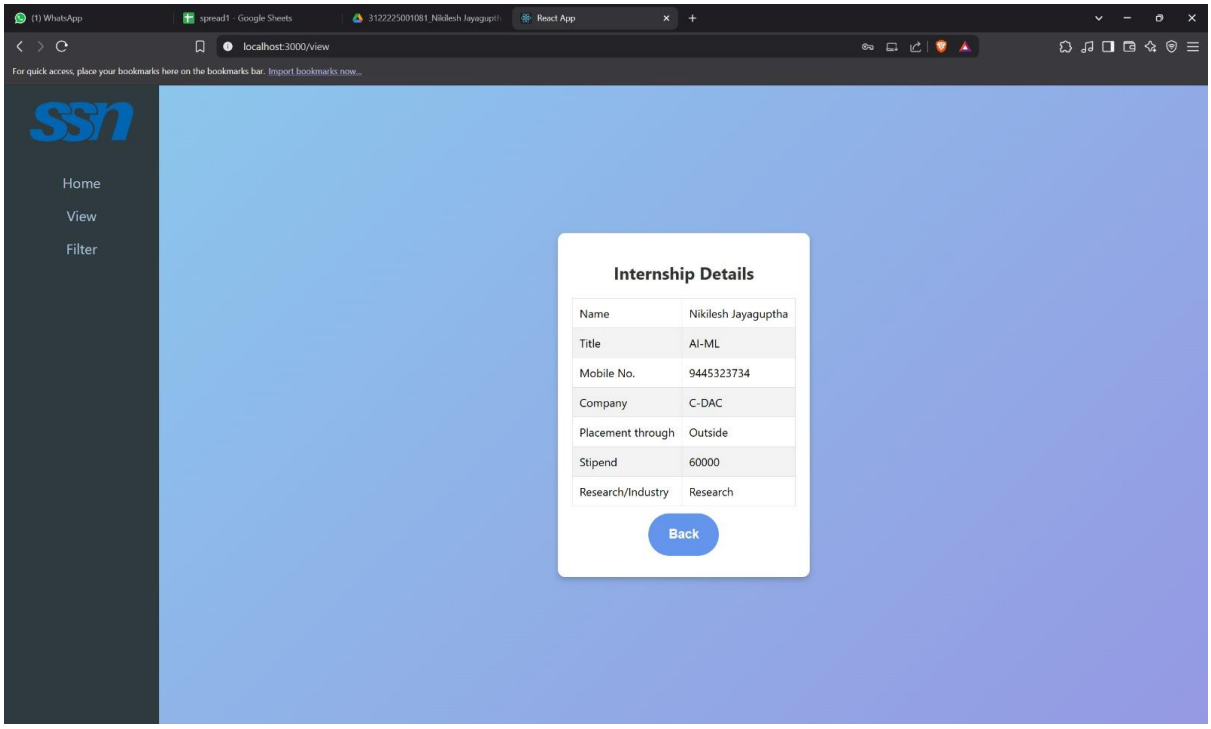


7) Check for internship details of a student based on His/Her's unique registration number

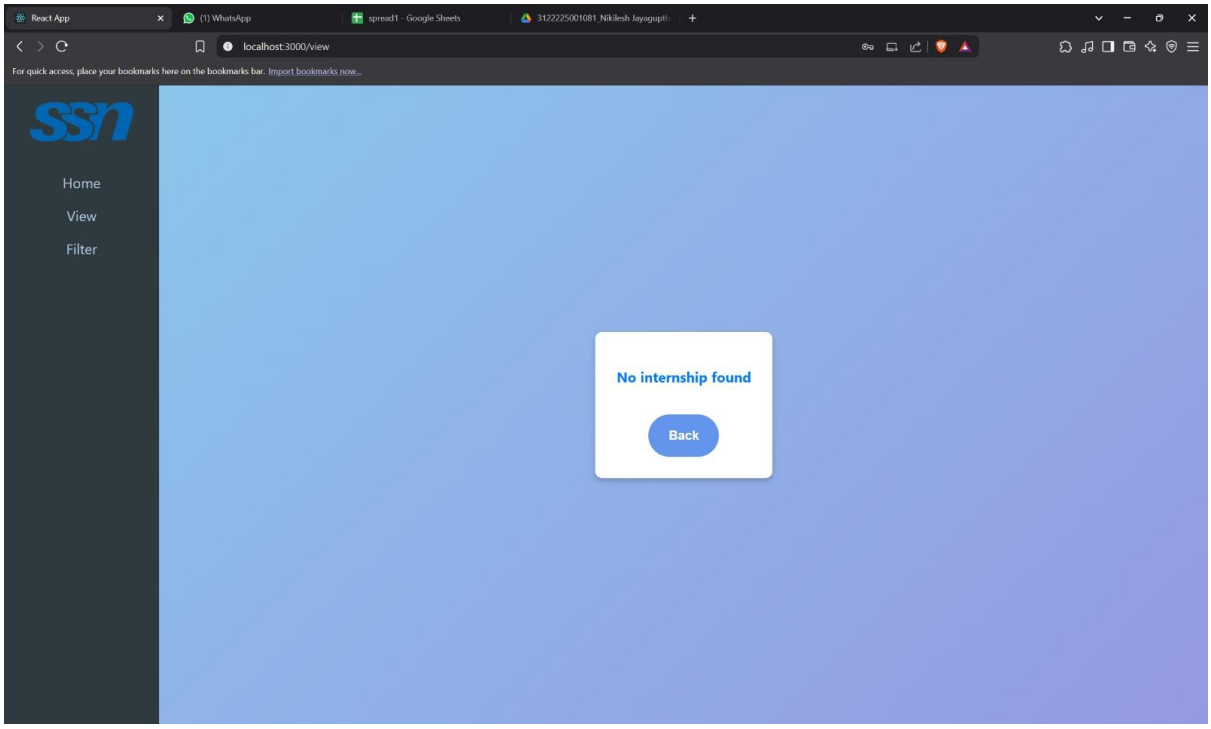


8) Displaying the internship details

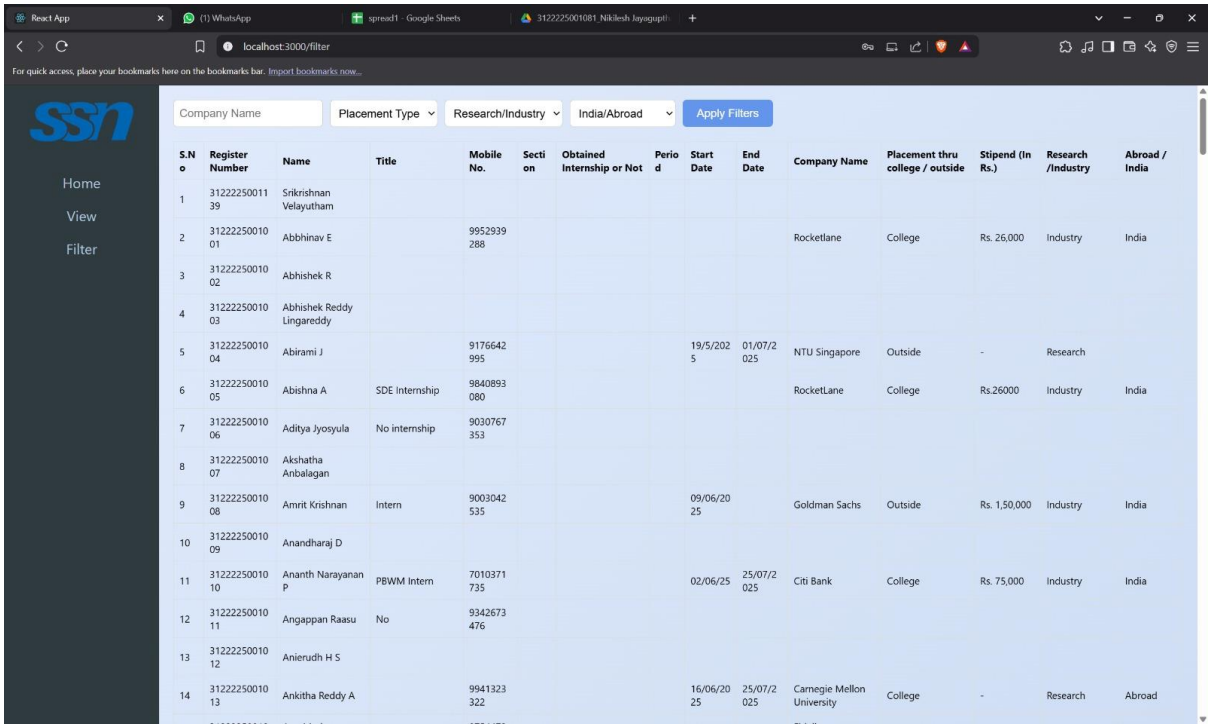
(Filled entry – Got Internship)



(Not filled entry – No Internship yet)

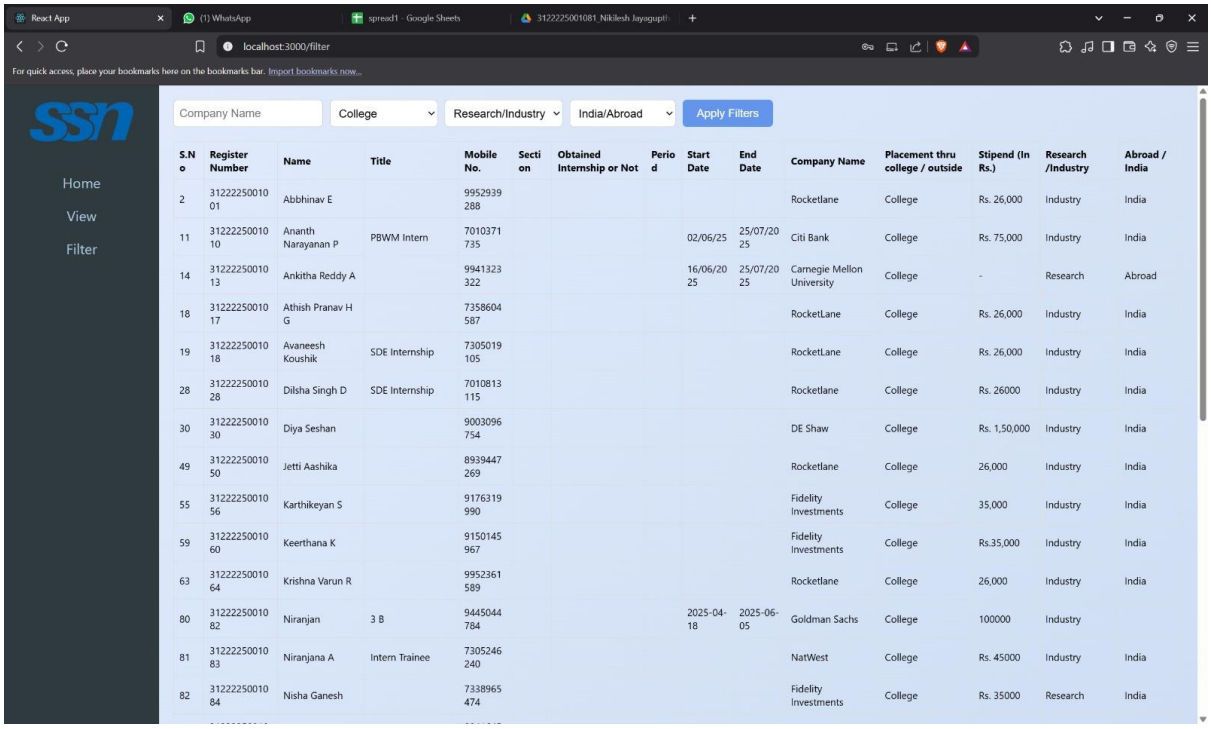


9) Viewing the internship details of all students



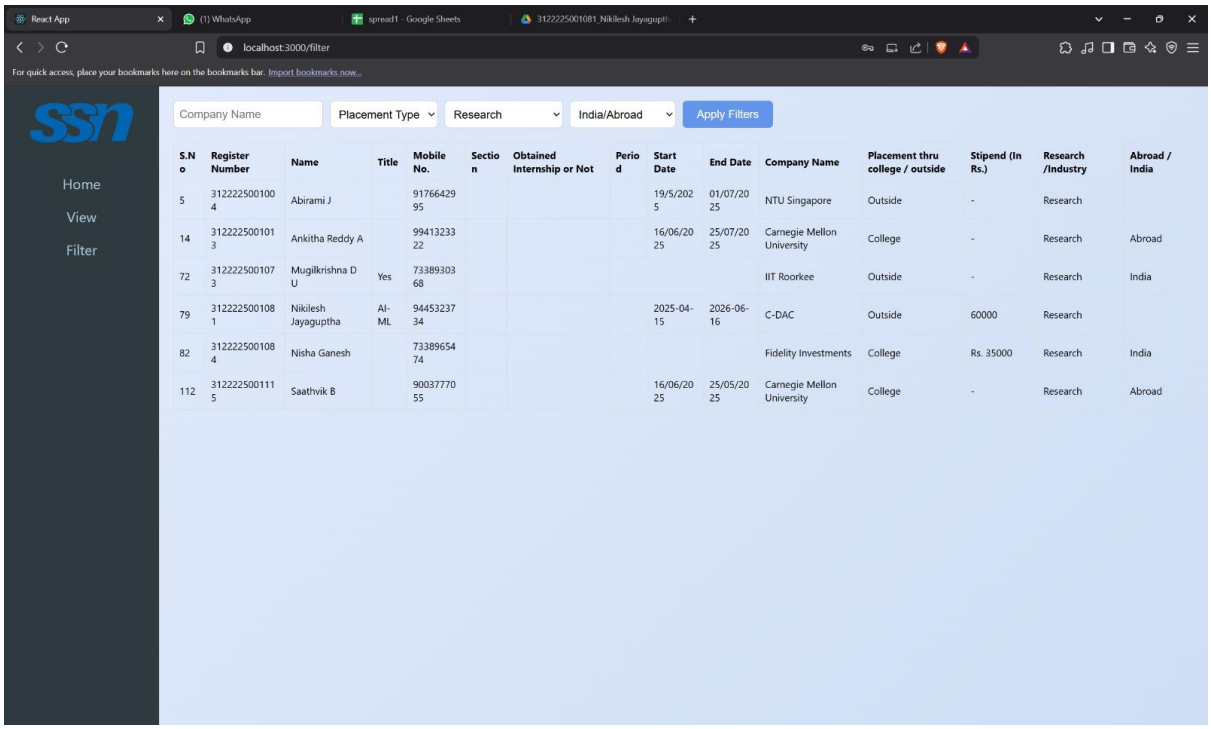
S.No	Register Number	Name	Title	Mobile No.	Section	Obtained Internship or Not	Period	Start Date	End Date	Company Name	Placement thru college / outside	Stipend (In Rs.)	Research /Industry	Abroad / India
1	3122225001139	Srikrishnan Velayutham												
2	3122225001001	Abbhinav E		9952939288						Rocketlane	College	Rs. 26,000	Industry	India
3	3122225001002	Abhishek R												
4	3122225001003	Abhishek Reddy Lingareddy												
5	3122225001004	Abirami J		9176642995				19/5/2025	01/07/2025	NTU Singapore	Outside	-	Research	
6	3122225001005	Abishna A	SDE Internship	9840893080						RocketLane	College	Rs.26000	Industry	India
7	3122225001006	Aditya Jyosyula	No internship	9030767353										
8	3122225001007	Akshatha Anbalagan												
9	3122225001008	Amrit Krishnan	Intern	9003042535				09/06/2025		Goldman Sachs	Outside	Rs. 1,50,000	Industry	India
10	3122225001009	Anandharaj D												
11	3122225001010	Ananth Narayanan P	PBWM Intern	7010371735				02/06/25	25/07/2025	Citi Bank	College	Rs. 75,000	Industry	India
12	3122225001011	Angappan Raasu	No	9342673476										
13	3122225001012	Anierudh H S												
14	3122225001013	Ankitha Reddy A		9941323322				16/06/2025	25/07/2025	Carnegie Mellon University	College	-	Research	Abroad

10) Filtering based on “Internship through college”



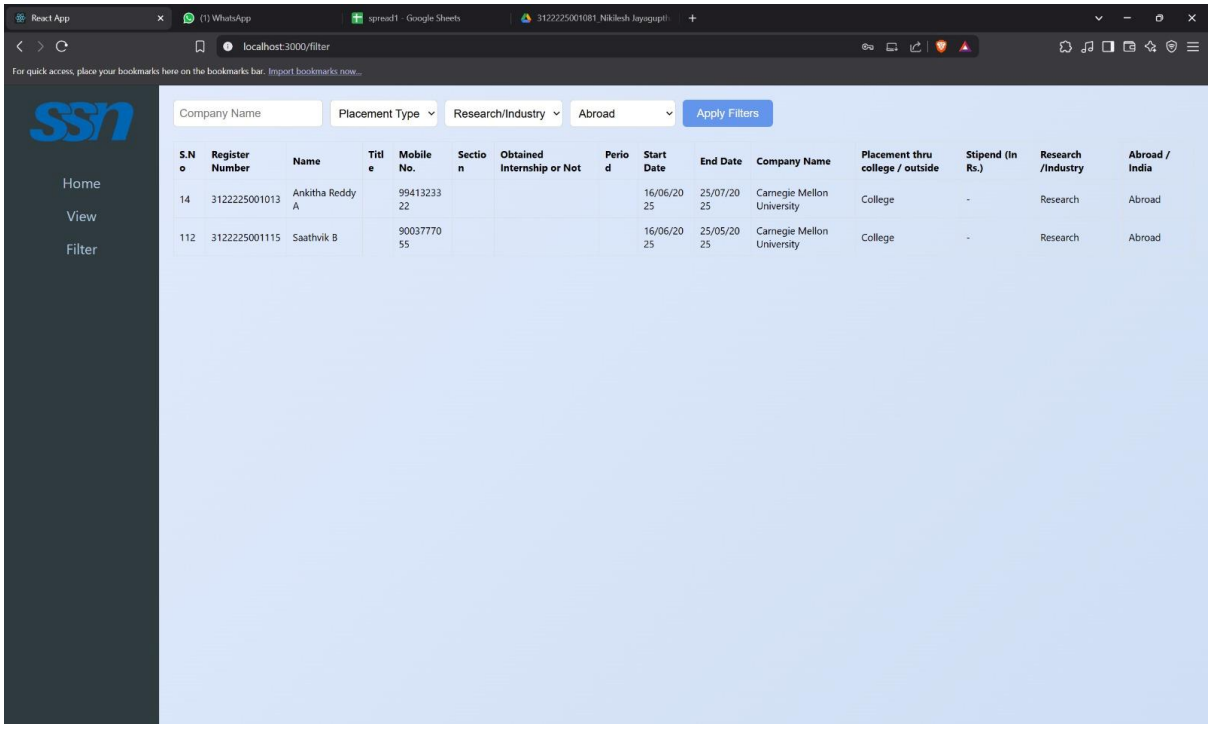
S.No	Register Number	Name	Title	Mobile No.	Section	Obtained Internship or Not	Period	Start Date	End Date	Company Name	Placement thru college / outside	Stipend (In Rs.)	Research /Industry	Abroad / India
2	3122225001001	Abbhinav E		9952939288						Rocketlane	College	Rs. 26,000	Industry	India
11	3122225001010	Ananth Narayanan P	PBWM Intern	7010371735				02/06/25	25/07/2025	Citi Bank	College	Rs. 75,000	Industry	India
14	3122225001013	Ankitha Reddy A		9941323322				16/06/2025	25/07/2025	Carnegie Mellon University	College	-	Research	Abroad
18	3122225001017	Athish Pranav H G		7358604587						RocketLane	College	Rs. 26,000	Industry	India
19	3122225001018	Avaneesh Koushik	SDE Internship	7305019105						RocketLane	College	Rs. 26,000	Industry	India
28	3122225001028	Dilsha Singh D	SDE Internship	7010813115						Rocketlane	College	Rs. 26000	Industry	India
30	3122225001030	Diya Seshan		9003096754						DE Shaw	College	Rs. 1,50,000	Industry	India
49	3122225001050	Jetti Aashika		8939447269						Rocketlane	College	26,000	Industry	India
55	3122225001056	Karthikeyan S		9176319990						Fidelity Investments	College	35,000	Industry	India
59	3122225001060	Keerthana K		9150145967						Fidelity Investments	College	Rs.35,000	Industry	India
63	3122225001064	Krishna Varun R		9952361589						Rocketlane	College	26,000	Industry	India
80	3122225001082	Niranjan	3 B	9445044784				2025-04-18	2025-06-05	Goldman Sachs	College	100000	Industry	
81	3122225001083	Niranjana A	Intern Trainee	7305246240						NatWest	College	Rs. 45000	Industry	India
82	3122225001084	Nisha Ganesh		7338965474						Fidelity Investments	College	Rs. 35000	Research	India

11) Filtering based on “Research Internship”



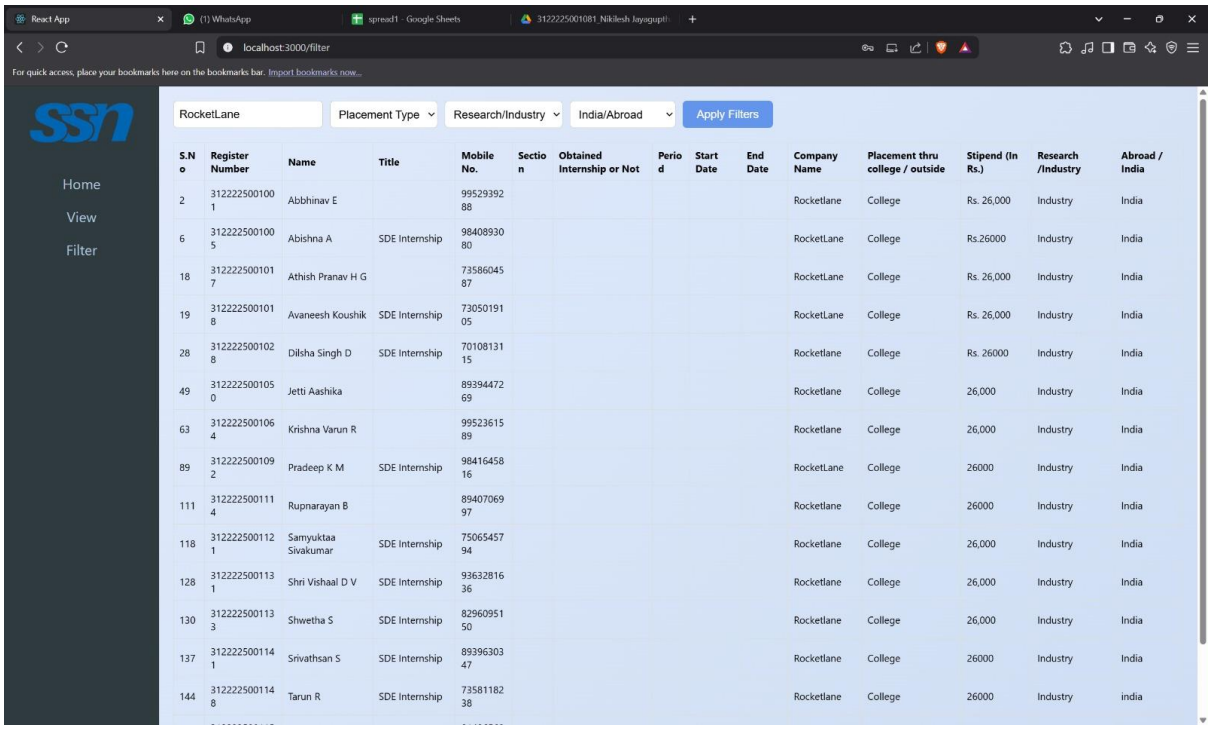
S.No	Register Number	Name	Title	Mobile No.	Section	Obtained Internship or Not	Period	Start Date	End Date	Company Name	Placement thru college / outside	Stipend (In Rs.)	Research /Industry	Abroad / India
5	3122225001004	Abirami J		9176642995				19/5/2025	01/07/2025	NTU Singapore	Outside	-	Research	
14	3122225001013	Ankitha Reddy A		9941323322				16/06/2025	25/07/2025	Carnegie Mellon University	College	-	Research	Abroad
72	3122225001073	Mugilkrishna D U	Yes	7338930368						IIT Roorkee	Outside	-	Research	India
79	3122225001081	Nikilesh Jayaguptha	AI-ML	9445323734				2025-04-15	2026-06-16	C-DAC	Outside	60000	Research	
82	3122225001084	Nisha Ganesh		7338965474						Fidelity Investments	College	Rs. 35000	Research	India
112	3122225001115	Saathvik B		9003777055				16/06/2025	25/05/2025	Carnegie Mellon University	College	-	Research	Abroad

12) Filtering based on “Internship Location – Abroad”



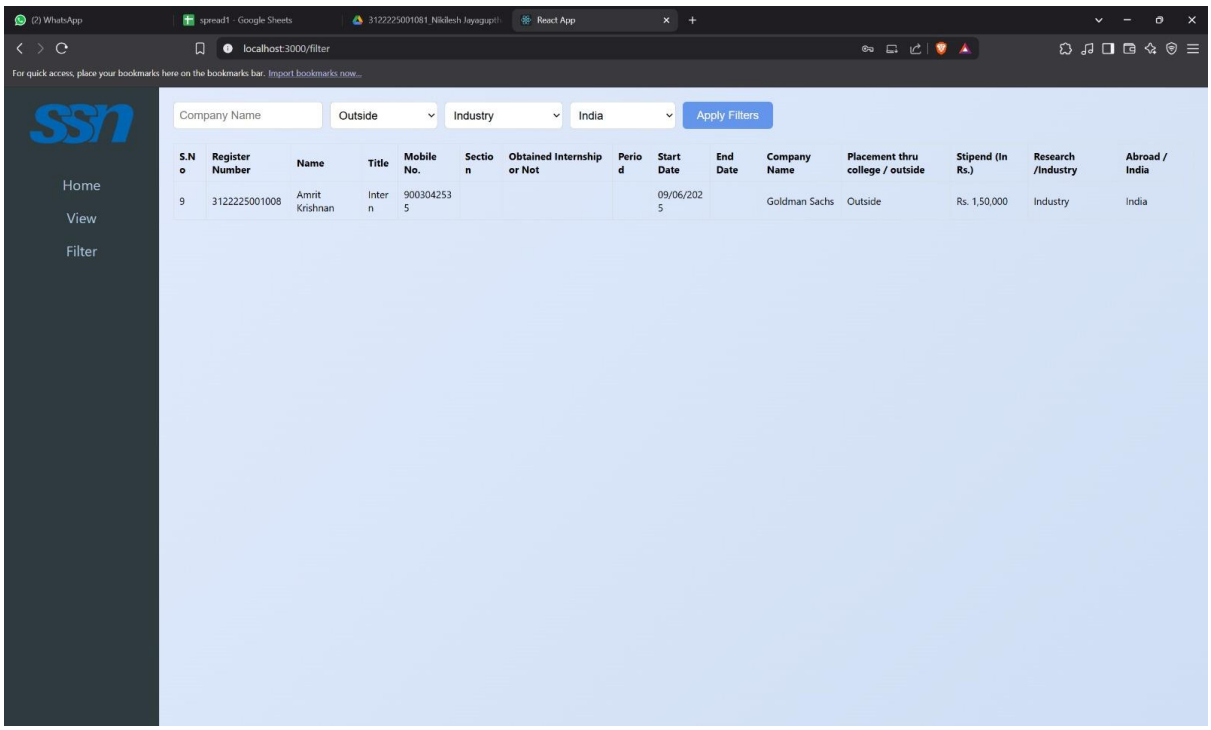
S.No	Register Number	Name	Title	Mobile No.	Section	Obtained Internship or Not	Period	Start Date	End Date	Company Name	Placement thru college / outside	Stipend (In Rs.)	Research /Industry	Abroad / India
14	3122225001013	Ankitha Reddy A		9941323322				16/06/2025	25/07/2025	Carnegie Mellon University	College	-	Research	Abroad
112	3122225001115	Saathvik B		9003777055				16/06/2025	25/05/2025	Carnegie Mellon University	College	-	Research	Abroad

13) Filtering based on Company Name (in this case, Rocketlane)

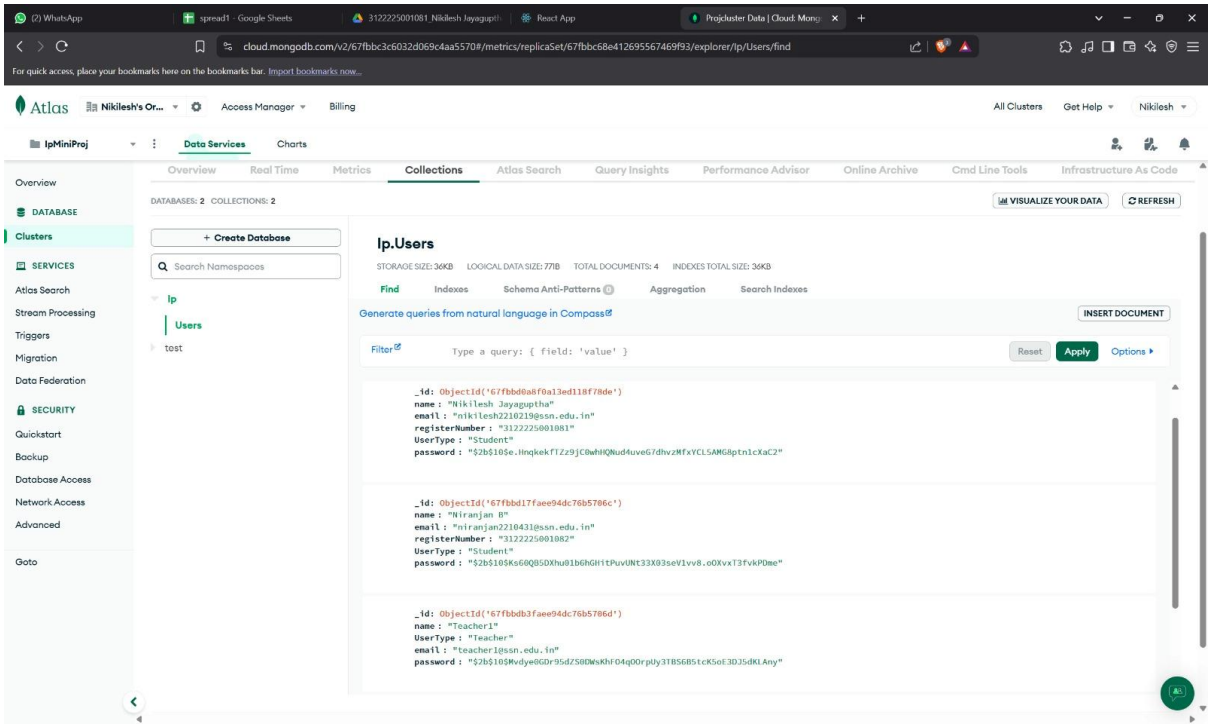


S.No	Register Number	Name	Title	Mobile No.	Section	Obtained Internship or Not	Period	Start Date	End Date	Company Name	Placement thru college / outside	Stipend (In Rs.)	Research /Industry	Abroad / India
2	3122225001001	Abbhinav E		9952939288						Rocketlane	College	Rs. 26,000	Industry	India
6	3122225001005	Abishna A	SDE Internship	9840893080						RocketLane	College	Rs.26000	Industry	India
18	3122225001017	Athish Pranav H G		7358604587						RocketLane	College	Rs. 26,000	Industry	India
19	3122225001018	Avaneesh Koushik	SDE Internship	7305019105						RocketLane	College	Rs. 26,000	Industry	India
28	3122225001028	Dilsha Singh D	SDE Internship	7010813115						Rocketlane	College	Rs. 26000	Industry	India
49	3122225001050	Jetti Aashika		8939447269						Rocketlane	College	26,000	Industry	India
63	3122225001064	Krishna Varun R		9952361589						Rocketlane	College	26,000	Industry	India
89	3122225001092	Pradeep K M	SDE Internship	9841645816						RocketLane	College	26000	Industry	India
111	3122225001114	Rupnarayan B		8940706997						Rocketlane	College	26000	Industry	India
118	3122225001121	Samyuktaa Sivakumar	SDE Internship	7506545794						Rocketlane	College	26,000	Industry	India
128	3122225001131	Shri Vishaal D V	SDE Internship	9363281636						Rocketlane	College	26,000	Industry	India
130	3122225001133	Shwetha S	SDE Internship	8296095150						Rocketlane	College	26,000	Industry	India
137	3122225001141	Srivathsan S	SDE Internship	8939630347						Rocketlane	College	26000	Industry	India
144	3122225001148	Tarun R	SDE Internship	7358118238						Rocketlane	College	26000	Industry	india

14) Filtering based on multiple parameters (Placement Through College/Outside, Research/Industry and India/Abroad)

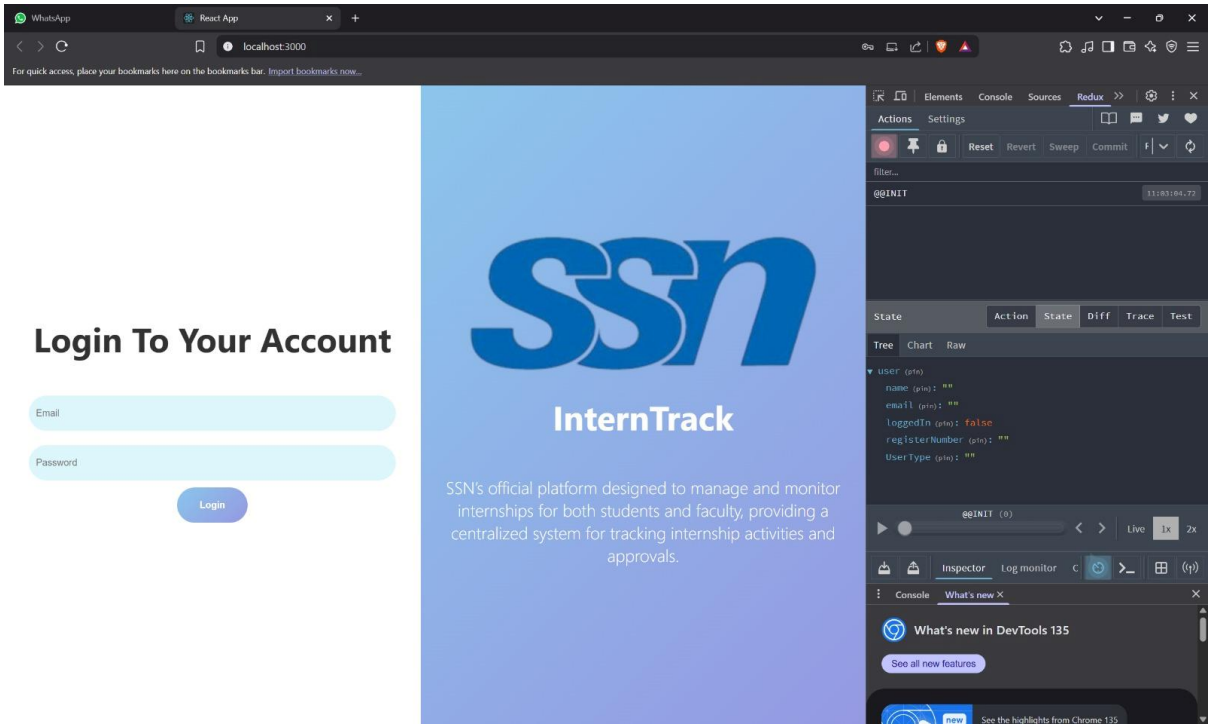


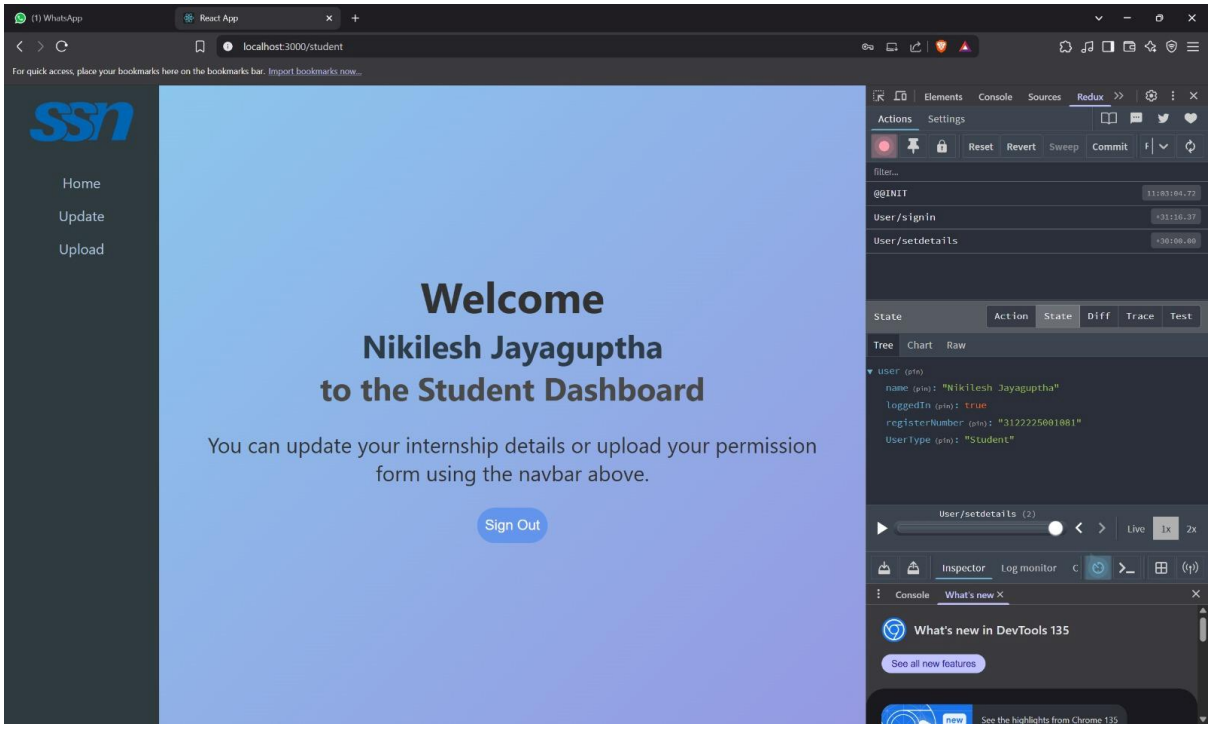
MONGODB SCREENSHOTS



We can observe that the password is hashed to protect the user identity and potential attacks.

USAGE OF REDUX





8. SCOPE AND LIMITATIONS

SCOPE:

1. Automated PDF Extraction and Data Verification
   Enables students to upload internship offer letters in PDF format, extract key fields using OCR, and verify them against institutional records.
2. Google Sheets and Drive Integration
   Dynamically updates verified data in Google Sheets and stores documents in structured student folders on Google Drive.
3. Manual Correction of Extracted Data
   Allows users to edit extracted information before final submission to ensure accuracy and prevent false verification.
4. Centralized Dashboard with Filtering and Search
   Faculty can view, filter, and search all student internship records from a unified, scrollable interface.
5. Robust State Management via Redux & React Hooks
   Tracks user login, file uploads, and extracted data across pages for a seamless experience.

LIMITATIONS:

1. OCR Accuracy Depends on PDF Quality
   Irregular layouts or low-resolution scans can lead to incorrect field extraction, requiring manual intervention.
2. Single-File Verification Workflow
   The system currently processes one PDF at a time; bulk uploads and automation are not yet supported.
3. Google API Quota Limits
   Drive and Sheets operations are bound by API usage quotas, which may cause delays in high-traffic scenarios.

9. FUTURE IMPROVEMENTS

1. User Authentication System
Add a secure login system with role-based access (e.g., Admin, Verifier, Student) to control actions like upload, verify, and update.

2. Bulk Verification & Upload
Support multi-PDF upload and batch verification to reduce manual overhead and speed up the processing of large datasets.

3. Analytics Dashboard
Introduce charts and statistics showing verification trends, number of successful uploads, and field mismatch rates using libraries like Chart.js or Recharts.

4. Email Notifications
Send automated confirmation emails after successful verification or notify users if a file fails verification.

5. Deployment & Scaling
Containerize using Docker and deploy on platforms like Heroku, AWS, or GCP with support for CI/CD pipelines and horizontal scaling.

10. LEARNING OUTCOMES

- Learned how to create a full data pipeline from upload to storage by integrating React (frontend), Flask/Node.js (backend), Google Sheets API, and Google Drive API.
- Developed endpoints, managed POST/GET flows with multipart/form-data and JSON, and handled file uploads with Axios.
- Gained expertise in managing dynamic user interfaces, file states, error handling, and live messages using useState, useEffect, and conditional rendering.
- An end-to-end document verification flow was constructed, which reinforced logic building and data accuracy by extracting fields from PDFs, validating them against Sheets, and pushing verified files to Drive.
- Comprehended CORS headers, cross-origin requests, and the safe and effective configuration of client-server communication across various ports.

11. REFERENCES

- https://github.com/nazir20/Internship-Tracking-Web-Application

A web app that tracks internship applications and status updates for students and admins using MERN stack.

- https://github.com/AppStateESS/InternshipInventory

A React-based platform for managing and viewing internship data with a clean interface and organized records.

- https://www.hackveda.in/intern-tracker/

An online tracker that assists organizations in monitoring intern progress and documentation throughout the internship period.

- https://github.com/OmkarPathak/pyresparser

An online tracker that assists organizations in monitoring intern progress and documentation throughout the internship period.

- https://github.com/theoephraim/node-google-spreadsheet

A Python library that extracts useful information like name, email, skills, and education from resumes using NLP.