Trackly : A Lead Management System

By

Sanchala Ritesh – 92200133001
Arayan Langhanoja - 922200133030

Under the guidance of

Prof. Shamsaagazarzoo Alam

A Project Submitted to

Marwadi University in Partial Fulfillment of the Requirements for the **Bachelor of Technology** in **Information and Communication Technology**

April 2025

MARWADI UNIVERSITY
Rajkot-Morbi Road, At & Po. Gauridad,
Rajkot-360003, Gujarat, India.

Information and Communication Technology

2024-25

CERTIFICATE

This is to certify that the project entitled **Trackly : A Lead Management System** has been carried out by **Sanchala Ritesh (92200133001)** and **Aryan Langhnoja (92200133030)** under my guidance in partial fulfillment of the degree of Bachelor Engineering in Information and Communication Technology (6th Semester) of Marwadi University, Rajkot during the academic year 2024-25.

Date : 16-05-2025

Internal Guide	**Head of the Department**
Mr. Shamsagazarzoo Alam Subject Co-ordinator C.T.O. Ally Soft Solutions	Head of Department ICT Engineering

TABLE OF CONTENTS

ABSTRACT

The Lead Management Web Application is a client workflow automation system designed to help companies efficiently manage and convert potential customers (leads) received from marketing agencies. This solution addresses key challenges in lead handling, including data organization, follow-up continuity, employee accountability, and performance tracking, ultimately improving the chances of converting leads into long-term clients.

At its core, the application supports both manual and bulk upload of lead data through .xlsx files. This feature allows companies to onboard large volumes of lead information with minimal effort. Once uploaded, leads can be systematically assigned to employees by the company owner through an intuitive lead assignment module. This ensures that each lead is handled by a responsible team member, facilitating organized and personalized outreach.

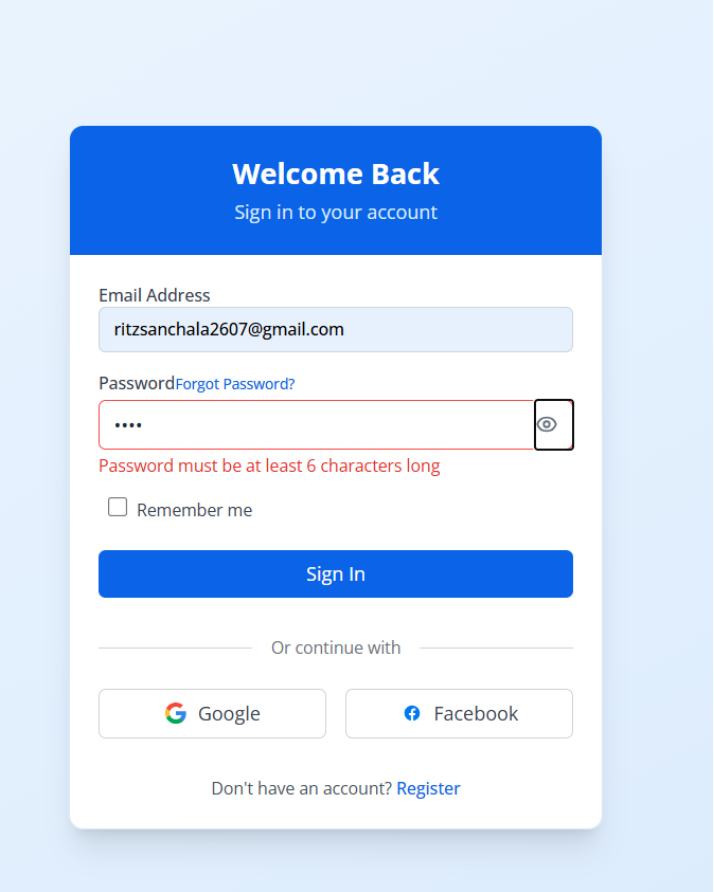
To maintain continuity in communication, the system includes a follow-up tracking feature where employees can log conversation summaries after each interaction. They can also schedule the next follow-up date, helping avoid missed opportunities and ensuring timely responses. This approach builds a structured timeline of engagement for each lead, which is essential for nurturing and converting prospects into clients.

A performance insights dashboard is provided for company owners, offering a clear overview of lead conversion rates and employee effectiveness. This dashboard includes visual and statistical data that helps management make informed decisions, identify top performers, and optimize the lead conversion strategy. It also highlights bottlenecks in the follow-up process, enabling timely interventions.

The entire system is built using a modern web development stack: React.js for the dynamic and responsive frontend, Node.js with Express.js for the backend API services, and a SQL database for structured data storage and retrieval. The architecture ensures scalability, performance, and maintainability, making the application suitable for small to medium-sized businesses looking to streamline their client acquisition workflows. Overall, this Lead Management Web Application offers a practical and powerful solution for companies aiming to improve client conversion rates through structured lead handling, proactive follow-up mechanisms, and actionable performance analytics.

Features:

1) Login Page:



A user-friendly login page with real-time input validation to ensure secure access. Includes seamless Google Single Sign-On (SSO) integration for quick and easy authentication.

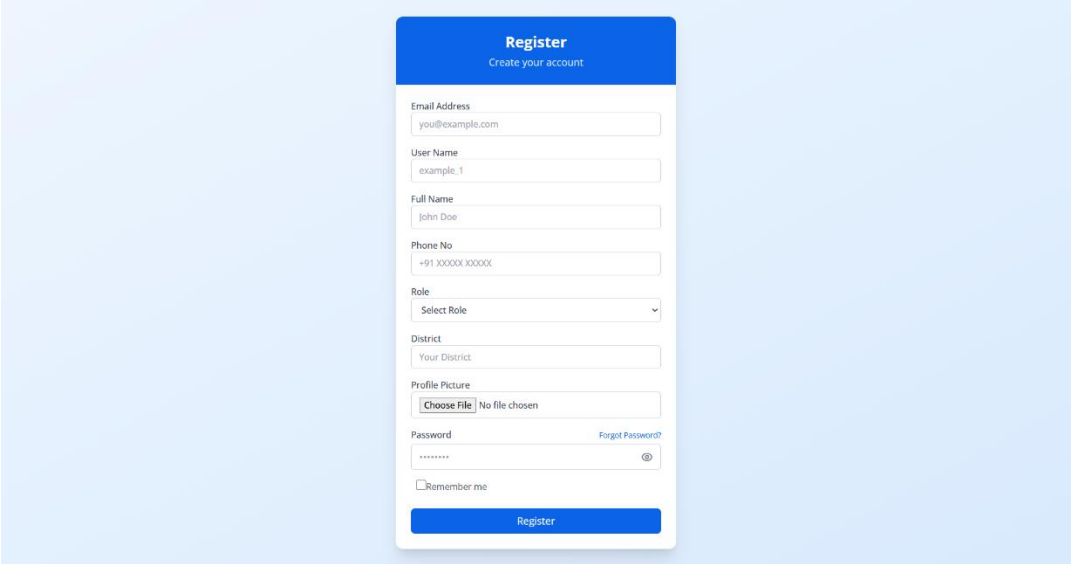
Code Snippet:

```
const navigate = useNavigate();
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [rememberMe, setRememberMe] = useState(false);
  const [showPassword, setShowPassword] = useState(false);
  const [token, setToken] = useState('No Token');
  const [role, setRole] = useState('No Role');

  const fetchSession = async () => {
    try {
      const res = await
axios.get(${process.env.REACT_APP_BASE_URL}/api/user/get-token, {
        withCredentials: true,
      });
```

```
      const freshToken = res.data.token;
      setToken(freshToken);

      const userData = await
axios.get(${process.env.REACT_APP_BASE_URL}/api/user/get-user, {
        headers: { Authorization: Bearer ${freshToken} },
        withCredentials: true,
      });

      setRole(userData.data.user.role); // NOTE: use data.user.role, not just
user.role
    } catch (error) {
      setToken('Error');
    }
  };

  useEffect(() => {
    if (role && role !== 'No Role') {
      if (role === 'Admin') navigate('/admindashboard');
      else if (role === 'Employee') navigate('/empdashboard');
      else if (role === 'Marketing Agency') navigate('/madashboard');
      else navigate('/login');
    }
  }, [role, navigate]);

  const handleSubmit = (e) => {
    e.preventDefault();
    const payload = {
      email,
      password,
    };

    axios
      .post(${process.env.REACT_APP_BASE_URL}/api/user/login, payload, {
        headers: {
          'Content-Type': 'application/json',
        },
        withCredentials: true,
      })
      .then((res) => {
        // eslint-disable-next-line no-alert
        alert(res.data.message);
        setEmail('');
        setPassword('');
        fetchSession();
      })
      .catch((err) => {
        // eslint-disable-next-line no-console
        console.error(err);
        // eslint-disable-next-line no-alert
        alert('Something went wrong during registration.');
```

```
    });
  // Handle authentication logic here
};

const togglePasswordVisibility = () => {
  setShowPassword(!showPassword);
};
```

2) Register Page:



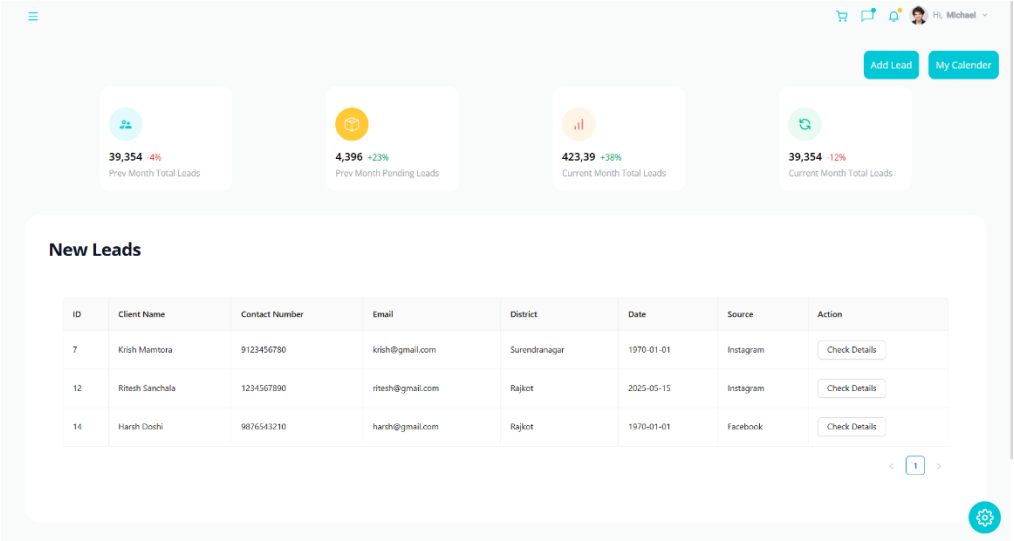
A secure registration page with field-level validation to ensure all inputs are required. Validates email format, password strength, and other essential fields for accurate user data.

Code Snippet:

```
const navigate = useNavigate();
const [formData, setFormData] = useState({
  user_name: '',
  name: '',
  email: '',
  phone: '',
  role: '',
  district: '',
  password: '',
});

const [rememberMe, setRememberMe] = useState(false);
const [showPassword, setShowPassword] = useState(false);
const [selectedFile, setSelectedFile] = useState(null);

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

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

const handleFileChange = (e) => {
  setSelectedFile(e.target.files[0]);
};

const togglePasswordVisibility = () => {
  setShowPassword(!showPassword);
};

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

  const data = new FormData();
  Object.keys(formData).forEach((key) => {
    data.append(key, formData[key]);
  });

  if (selectedFile) {
    data.append('image', selectedFile);
  }

  axios
    .post(`${process.env.REACT_APP_BASE_URL}/api/user/register`, data, {
      headers: { 'Content-Type': 'multipart/form-data' },
    })
    .then((res) => {
      // eslint-disable-next-line no-alert
      alert(res.data.message);
      setFormData({
        user_name: '',
        name: '',
        email: '',
        phone: '',
        role: '',
        district: '',
        password: '',
      });
      setSelectedFile(null);
      navigate('/login');
    })
    .catch((err) => {
      // eslint-disable-next-line no-console
      console.log(err);
      // eslint-disable-next-line no-alert
      alert('Something went wrong during registration.');
    });
};
```

3) Employee Dashboard:



- Interactive employee dashboard with features to add new leads and view assigned lead details.
- Includes a personalized calendar for each employee to track and manage their tasks efficiently.

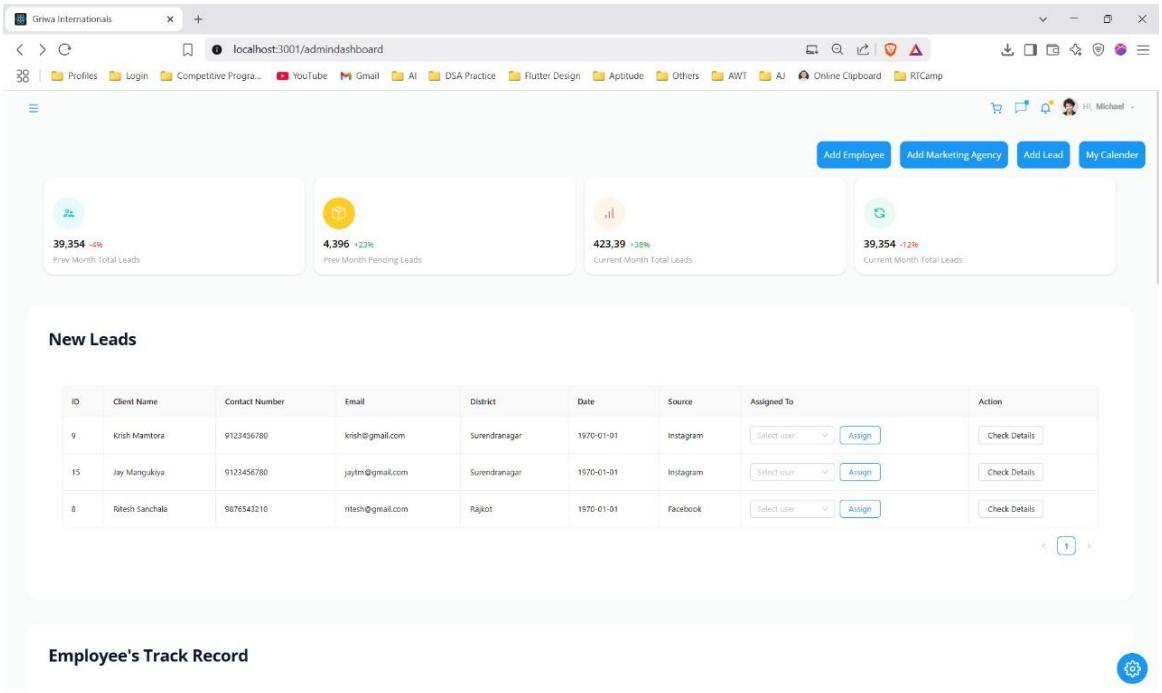
Code Snippet :-

```
  const { currentColor } = useStateContext();
  const toolbarOptions = ['Search'];

  const editing = { allowDeleting: true, allowEditing: true };

  const [token, setToken] = useState('');
  const [empId, setEmpID] = useState(null);
  const [data, setData] = useState([]);
  const navigate = useNavigate();

  useEffect(() => {
    const fetchSession = async () => {
      try {
        const res = await
axios.get(`${process.env.REACT_APP_BASE_URL}/api/user/get-token`, {
          withCredentials: true,
        });

        const freshToken = res.data.token;
        setToken(freshToken);

        const userData = await
axios.get(`${process.env.REACT_APP_BASE_URL}/api/user/get-user`, {
```

```
          headers: { Authorization: `Bearer ${freshToken}` }, // use
freshToken here
          withCredentials: true,
        });

        setEmpID(userData.data.user.user_id);
      } catch (error) {
        console.error('Error fetching session:', error);
      }
    };

    fetchSession();
  }, []);

  useEffect(() => {
    if (!empId) return; // wait until empId is set

    const fetchLeads = async () => {
      try {
        const res = await
axios.get(`${process.env.REACT_APP_BASE_URL}/api/lead/emp/${empId}`);
        setData(res.data);
      } catch (err) {
        console.error('Error fetching leads:', err);
      }
    };

    fetchLeads();
  }, [empId]); // <== This triggers once empId is updated

  const columns = [
    {
      title: 'ID',
      dataIndex: 'lead_id',
      key: 'lead_id',
    },
    {
      title: 'Client Name',
      dataIndex: 'client',
      key: 'client',
    },
    {
      title: 'Contact Number',
      dataIndex: 'contact_number',
      key: 'contact_number',
    },
    {
      title: 'Email',
      dataIndex: 'email',
      key: 'email',
    },
```

```
    {
      title: 'District',
      dataIndex: 'district',
      key: 'district',
    },
    {
      title: 'Date',
      dataIndex: 'date',
      key: 'date',
    },
    {
      title: 'Source',
      dataIndex: 'source',
      key: 'source',
    },
    {
      title: 'Action',
      key: 'action',
      render: (_, record) => (
        <Button
          type="default"
          onClick={() => navigate('/leaddetails', { state: { leadData: record
} })}
        >
          Check Details
        </Button>
      ),
    },
  ];
```

4) Admin Dashboard:



- Admin dashboard with functionality to add new leads, assign them to employees, and view lead details.
- Also includes options to add new employees and register marketing agencies for better management.

Code Snippet :-

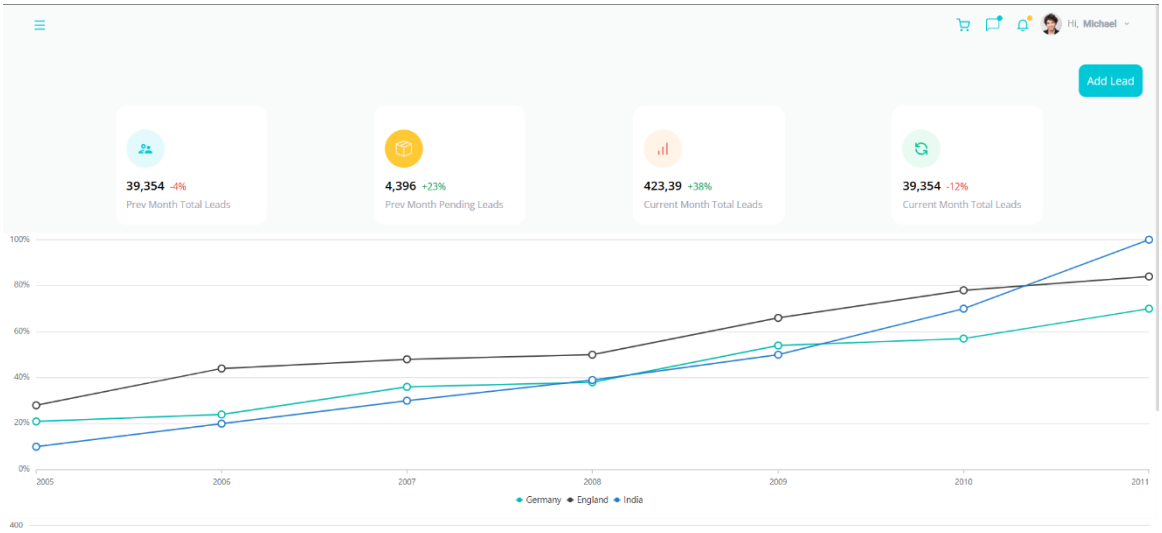
```
  const [assignments, setAssignments] = useState({});
  const [hoveredKey, setHoveredKey] = useState(null);
  const [employee, setEmployee] = useState([]);
  const [availableUsers, setAvailableUsers] = useState([{}]);
  const [data, setData] = useState([]);
  const navigate = useNavigate();

  const handleAssign = (key) => {
    const lead_id = key;
    const emp_id = assignments[key];

    console.log(`Assigning employee ${emp_id} to lead with ID ${lead_id}`);

    axios
      .put(
        `${process.env.REACT_APP_BASE_URL}/api/lead/${lead_id}`,
        {
          user_id: emp_id, // or use the correct field name expected by your
backend
        },
```

```
      {
        headers: {
          'Content-Type': 'application/json',
        },
        withCredentials: true,
      },
    )
    .then((res) => {
      alert(`Lead ${lead_id} assigned successfully`);
    })
    .catch((err) => {
      console.error('Error assigning lead:', err);
      alert('Failed to assign employee to lead.');
    });
};

useEffect(() => {
  axios
    .get(`${process.env.REACT_APP_BASE_URL}/api/user/get_employees`)
    .then((res) => {
      setEmployee(res.data.employees);

      // Extract only names
      const names = res.data.employees.map((emp) => ({
        name: emp.name,
        id: emp.user_id,
      }));
      setAvailableUsers(names);
    })
    .catch((err) => {
      console.error('Error fetching employees', err);
    });

  axios
    .get(`${process.env.REACT_APP_BASE_URL}/api/lead/unassigned`)
    .then((res) => {
      setData(res.data.leads);
    })
    .catch((err) => {
      console.error('Error fetching employees', err);
    });
}, []);

const handleChange = (value, key) => {
  setAssignments({ ...assignments, [key]: value });
};

const columns = [
  {
    title: 'ID',
    dataIndex: 'lead_id',
```

```
    key: 'lead_id',
  },
  {
    title: 'Client Name',
    dataIndex: 'client',
    key: 'client',
  },
  {
    title: 'Contact Number',
    dataIndex: 'contact_number',
    key: 'contact_number',
  },
  {
    title: 'Email',
    dataIndex: 'email',
    key: 'emial',
  },
  {
    title: 'District',
    dataIndex: 'district',
    key: 'district',
  },
  {
    title: 'Date',
    dataIndex: 'date',
    key: 'date',
  },
  {
    title: 'Source',
    dataIndex: 'source',
    key: 'source',
  },
  {
    title: 'Assigned To',
    key: 'assignedTo',
    render: (_, record) => (
      <div style={{ display: 'flex', gap: '8px', alignItems: 'center' }}>
        {record.user_id === null ? (
          <>
            <Select
              defaultValue={assignments[record.lead_id] || null}
              style={{ width: 150 }}
              placeholder="Select user"
              onChange={(value) => handleChange(value, record.lead_id)}
            >
              {availableUsers.map((user) => (
                <Option key={user.id} value={user.id}>
                  {user.name}
                </Option>
              ))}
            </Select>
```

```
              <Button
                onMouseEnter={() => setHoveredKey(record.lead_id)}
                onMouseLeave={() => setHoveredKey(null)}
                onClick={() => handleAssign(record.lead_id)}
                style={{
                  border: '1px solid #1890ff',
                  color: hoveredKey === record.lead_id ? '#fff' : '#1890ff',
                  backgroundColor: hoveredKey === record.lead_id ? '#1890ff' :
'#fff',
                  transition: 'all 0.3s',
                }}
              >
                Assign
              </Button>
            </>
          ) : (
            <span>
              {availableUsers.find((user) => user.id === record.user_id)?.name
|| record.user_id}
            </span>
          )}
        </div>
      ),
    },

    {
      title: 'Action',
      key: 'action',
      render: (_, record) => (
        <Button
          type="default"
          onClick={() => navigate('/leaddetails', { state: { leadData: record
} })}
        >
          Check Details
        </Button>
      ),
    },
  ];
```

5) Market Agency:



- Marketing agency dashboard with options to manually add leads or upload them via XLSX files.
- Includes interactive graphs showing monthly and yearly lead generation statistics for performance tracking

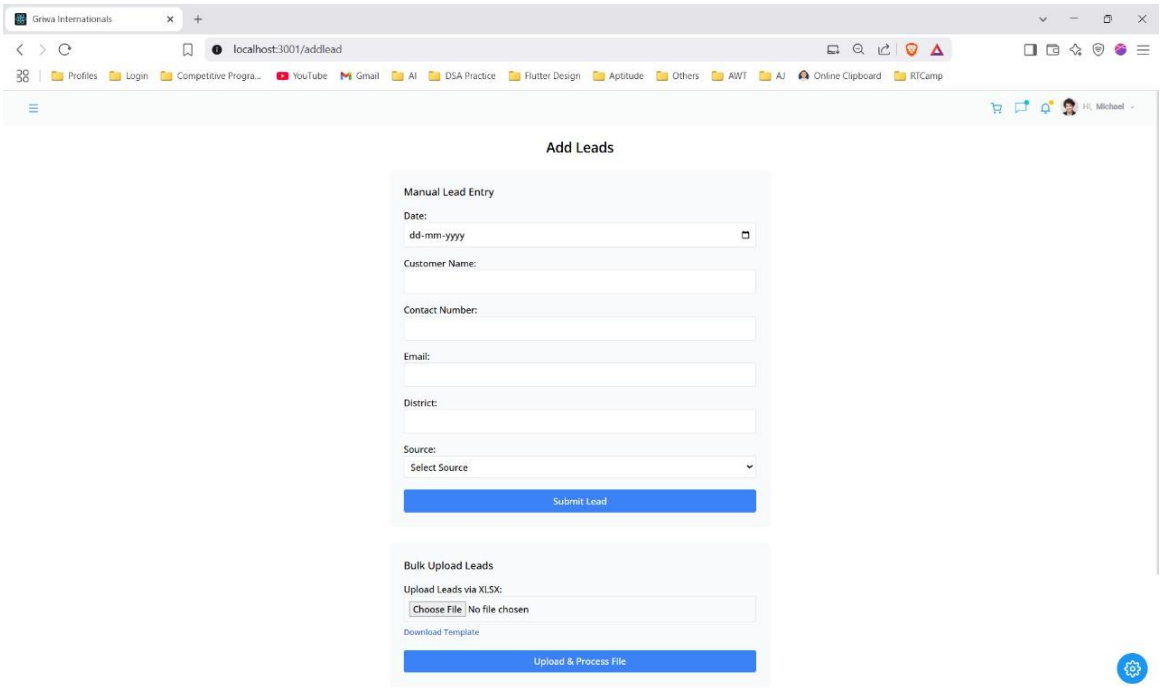
Code Snippet :-

```
import React from 'react';
// import { GoPrimitiveDot } from 'react-icons/go';
// import { IoIosMore } from 'react-icons/io';
// import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import { Link } from 'react-router-dom';
import {
  GridComponent,
  Inject,
  ColumnsDirective,
  ColumnDirective,
  Search,
  Page,
} from '@syncfusion/ej2-react-grids';
import { Button, Header } from '../components';
import {
  earningData,
  // medicalproBranding,
  // recentTransactions,
  // weeklyStats,
  // dropdownData,
  // SparklineAreaData,
  // ecomPieChartData,
  employeesData,
  employeesGrid,
```

```
} from '../data/dummy';

import { useStateContext } from '../contexts/ContextProvider';

// const DropDown = ({ currentMode }) => (
//   <div className="w-28 border-1 border-color px-2 py-1 rounded-md">
//     <DropDownListComponent
//       id="time"
//       fields={{ text: 'Time', value: 'Id' }}
//       style={{ border: 'none', color: currentMode === 'Dark' && 'white' }}
//       value="1"
//       dataSource={dropdownData}
//       popupHeight="220px"
//       popupWidth="120px"
//     />
//   </div>
// );

const MADashboard = () => {
  const { currentColor } = useStateContext();
  const toolbarOptions = ['Search'];

  const editing = { allowDeleting: true, allowEditing: true };
  return (
    <div className="mt-7">
      <div className="flex flex-wrap lg:flex-nowrap justify-end gap-x-4 mr-5">
        <Link to="/addlead">
          <Button color="white" bgColor={currentColor} text="Add Lead"
borderRadius="10px" />
        </Link>
        <Link to="/calendar">
          <Button color="white" bgColor={currentColor} text="My Calender"
borderRadius="10px" />
        </Link>
      </div>
      <div className="flex m-3 flex-wrap justify-evenly gap-1 items-center">
        {earningData
          .map((item) => (
            <div
              key={item.title}
              className="bg-white h-44 dark:text-gray-200 dark:bg-secondary-
dark-bg md:w-56  p-4 pt-9 rounded-2xl "
            >
              <button
                type="button"
                style={{ color: item.iconColor, backgroundColor: item.iconBg
}}
                
                className="text-2xl opacity-0.9 rounded-full  p-4 hover:drop-
shadow-xl"
              >
                {item.icon}
```

```
              </button>
              <p className="mt-3">
                <span className="text-lg font-semibold">{item.amount}</span>
                <span className={`text-sm text-${item.pcColor} ml-
2`}>{item.percentage}</span>
              </p>
              <p className="text-sm text-gray-400  mt-1">{item.title}</p>
            </div>
          ))
          .slice(-4)}
      </div>

      {/* New Leads Table */}
      <div className="m-2 md:m-10 mt-24 p-2 md:p-10 bg-white rounded-3xl">
        <Header title="New Leads" />
        <GridComponent
          dataSource={employeesData}
          width="auto"
          allowPaging
          allowSorting
          pageSettings={{ pageCount: 5 }}
          editSettings={editing}
          toolbar={toolbarOptions}
        >
          <ColumnsDirective>
            {/* eslint-disable-next-line react/jsx-props-no-spreading */}
            {employeesGrid.map((item, index) => (
              <ColumnDirective key={index} {...item} />
            ))}
          </ColumnsDirective>
          <Inject services={[Search, Page]} />
        </GridComponent>
      </div>
    </div>
  );
};

export default MADashboard;
```

6) Add Leads Page:



Add New Leads with options to manually add leads or upload them via XLSX files.

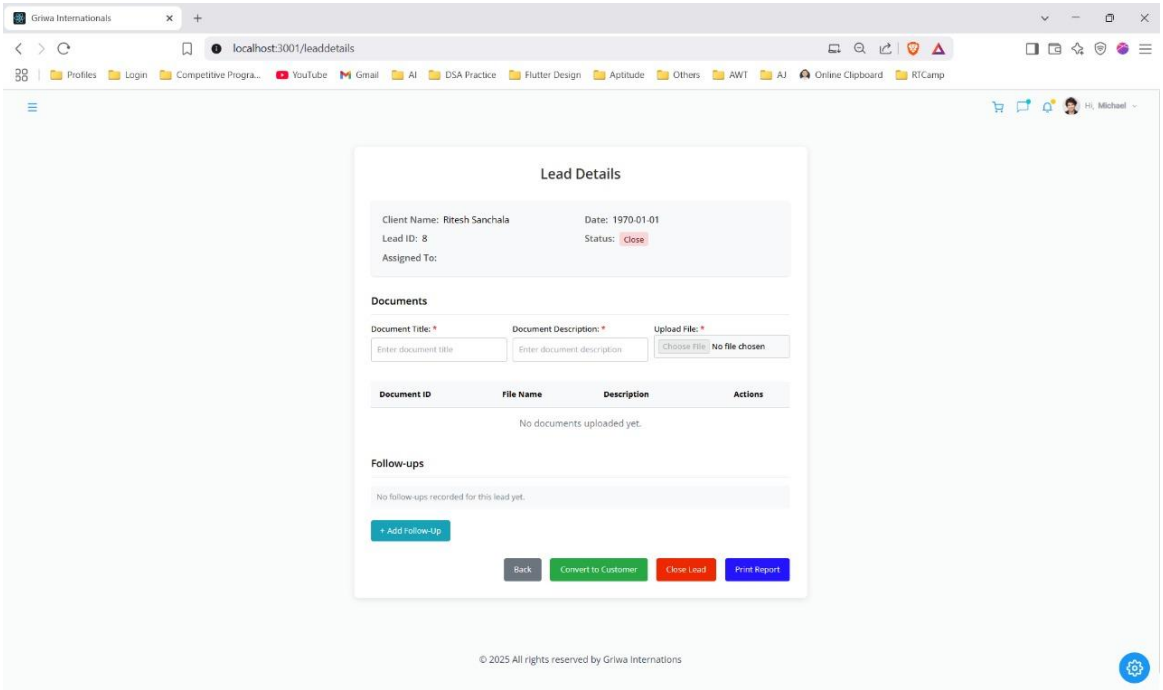
Code Snippet :-

```
  const [formData, setFormData] = useState({
    client: '',
    date: '',
    contact_number: '',
    email: '',
    district: '',
    source: '',
  });

  const [errors, setErrors] = useState({});

  const validate = () => {
    const newErrors = {};
    if (!formData.client.trim()) newErrors.client = 'Customer name is
required';
    if (!formData.date) newErrors.date = 'Date is required';
    if (!formData.contact_number.trim()) {
      newErrors.contact_number = 'Contact number is required';
    } else if (!/^\d{10}$/.test(formData.contact_number)) {
      newErrors.contact_number = 'Enter a valid 10-digit number';
    }
    if (!formData.email.trim()) {
      newErrors.email = 'Email is required';
    } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
```

```
      newErrors.email = 'Invalid email address';
    }
    if (!formData.district.trim()) newErrors.district = 'District is
required';
    if (!formData.source) newErrors.source = 'Source is required';

    setErrors(newErrors);
    return Object.keys(newErrors).length === 0;
  };

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
    setErrors({ ...errors, [e.target.name]: '' }); // Clear error on change
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!validate()) return;

    axios
      .post(`${process.env.REACT_APP_BASE_URL}/api/lead/`, formData)
      .then((res) => {
        alert(res.data.message);
        setFormData({
          client: '',
          date: '',
          contact_number: '',
          email: '',
          district: '',
          source: '',
        });
        setErrors({});
      })
      .catch((err) => {
        console.error(err);
        alert('Something went wrong during registration.');
      });
  };

  const handleFileUpload = (e) => {
    e.preventDefault();
    const file = e.target.files[0];
    if (!file) return;

    if (!file.name.endsWith('.xlsx') && !file.name.endsWith('.xls')) {
      alert('Please upload a valid Excel file (.xlsx or .xls)');
      return;
    }

    const reader = new FileReader();
```

```
  reader.onload = (evt) => {
    try {
      const bstr = evt.target.result;
      const workbook = XLSX.read(bstr, { type: 'binary' });

      const sheetName = workbook.SheetNames[0];
      const worksheet = workbook.Sheets[sheetName];
      const jsonData = XLSX.utils.sheet_to_json(worksheet, { defval: '' });

      axios
        .post(`${process.env.REACT_APP_BASE_URL}/api/lead/excel`, jsonData)
        .then((res) => {
          alert(res.data.message);
        })
        .catch((err) => {
          console.error(err);
          alert('Something went wrong during registration.');
        });
    } catch (error) {
      alert('Failed to read the Excel file.');
    }
  };

  reader.readAsBinaryString(file);
};
```

7) Lead Details Page:



- Lead details page displaying all follow-ups, with options to add new follow-ups and upload documents like quotations and electricity bills.

- Shows complete lead information including person name, lead number, and Lead generation date for easy reference.

Code Snippet :-

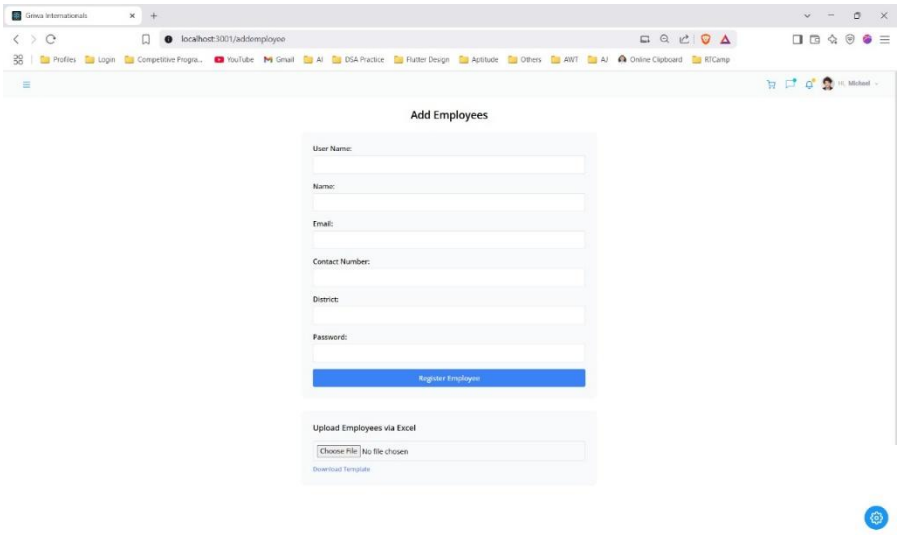
```
  const location = useLocation();
  const navigate = useNavigate();
  const leadData = location.state?.leadData;

  const [followUps, setFollowUps] = useState([]);
  const [documents, setDocuments] = useState([]);
  const [isUploading, setIsUploading] = useState(false);
  const [documentTitle, setDocumentTitle] = useState('');
  const [documentDescription, setDocumentDescription] = useState('');

  const fetchDocuments = () => {
    axios
      .get(`${process.env.REACT_APP_BASE_URL}/api/document/lead/${leadData.lea
d_id}`, {
        withCredentials: true,
      })
      .then((res) => {
        setDocuments(res.data.data || []);
      })
      .catch((err) => {
        console.error('Failed to fetch documents:', err);
        setDocuments([]);
      });
  };
  // Fetch existing follow-ups
  useEffect(() => {
    if (leadData?.lead_id) {
      axios
        .get(`${process.env.REACT_APP_BASE_URL}/api/followup/lead/${leadData.l
ead_id}`, {
          withCredentials: true,
        })
        .then((res) => {
          const formatted = res.data.follow_ups.map((f) => ({
            note: f.conclusion || '',
            date: f.next_followup_date?.slice(0, 10) || '',
          }));
          setFollowUps(formatted);
        })
        .catch((err) => {
          console.error('Failed to fetch follow-ups:', err);
          setFollowUps([]);
```

```
      });

    // Fetch documents related to this lead
    fetchDocuments();
  }
}, [leadData?.lead_id]);

const handleFollowUpChange = (index, field, value) => {
  const updated = [...followUps];
  updated[index] = { ...updated[index], [field]: value };
  setFollowUps(updated);
};

const handleAddFollowUp = () => {
  setFollowUps([...followUps, { note: '', date: '' }]);
};

const changeStatus = (new_staus) => {
  axios
    .put(`${process.env.REACT_APP_BASE_URL}/api/lead/${leadData.lead_id}`, {
      status: new_staus, // Send the status in the request body
    })
    .then((response) => {
      console.log('Status updated successfully:');
      // Optionally update UI or state here
    })
    .catch((error) => {
      console.error('Error updating status:', error);
    });
};

const handleSaveFollowUp = (index) => {
  const followUp = followUps[index];
  const payload = {
    user_id: leadData.user_id,
    no_of_followup: index + 1,
    lead_id: leadData.lead_id,
    conclusion: followUp.note,
    next_followup_date: followUp.date,
  };

  axios
    .post(`${process.env.REACT_APP_BASE_URL}/api/followup`, payload, {
      headers: {
        'Content-Type': 'application/json',
      },
      withCredentials: true,
    })
    .then((res) => {
      alert('Follow-up added successfully');
    })
```

```
      .catch((err) => {
        console.error(err);
        alert('Something went wrong while saving the follow-up.');
      });
  };

  const uploadDocument = (file) => {
    setIsUploading(true);

    const formData = new FormData();
    formData.append('document', file);
    formData.append('lead_id', leadData.lead_id);
    formData.append('doc_name', documentTitle);
    formData.append('doc_desc', documentDescription);
    // formData.append('uploaded_by', leadData.user_id);

    axios
      .post(`${process.env.REACT_APP_BASE_URL}/api/document/`, formData, {
        headers: {
          'Content-Type': 'multipart/form-data',
        },
        withCredentials: true,
      })
      .then((res) => {
        alert('Document uploaded successfully');
        setDocumentTitle('');
        setDocumentDescription('');
        fetchDocuments(); // Refresh the documents list
      })
      .catch((err) => {
        console.error('Error uploading document:', err);
        alert('Failed to upload document');
      })
      .finally(() => {
        setIsUploading(false);
      });
  };

  const handleFileChange = (e) => {
    if (!e.target.files[0]) return;

    const file = e.target.files[0];
    if (!documentTitle.trim()) {
      alert('Please enter a document title before uploading');
      return;
    }

    uploadDocument(file);
  };

  const handleDeleteDocument = (documentId) => {
```

```
    if (window.confirm('Are you sure you want to delete this document?')) {
      axios
        .delete(`${process.env.REACT_APP_BASE_URL}/api/document/${documentId}`
, {
          withCredentials: true,
        })
        .then(() => {
          alert('Document deleted successfully');
          fetchDocuments(); // Refresh the documents list
        })
        .catch((err) => {
          console.error('Error deleting document:', err);
          alert('Failed to delete document');
        });
    }
  };

  const handleBack = () => {
    navigate(-1);
  };
```

8) Add Employee Page:



Add New Employee with options to add manually or upload them via XLSX files.

Code Snippet :-

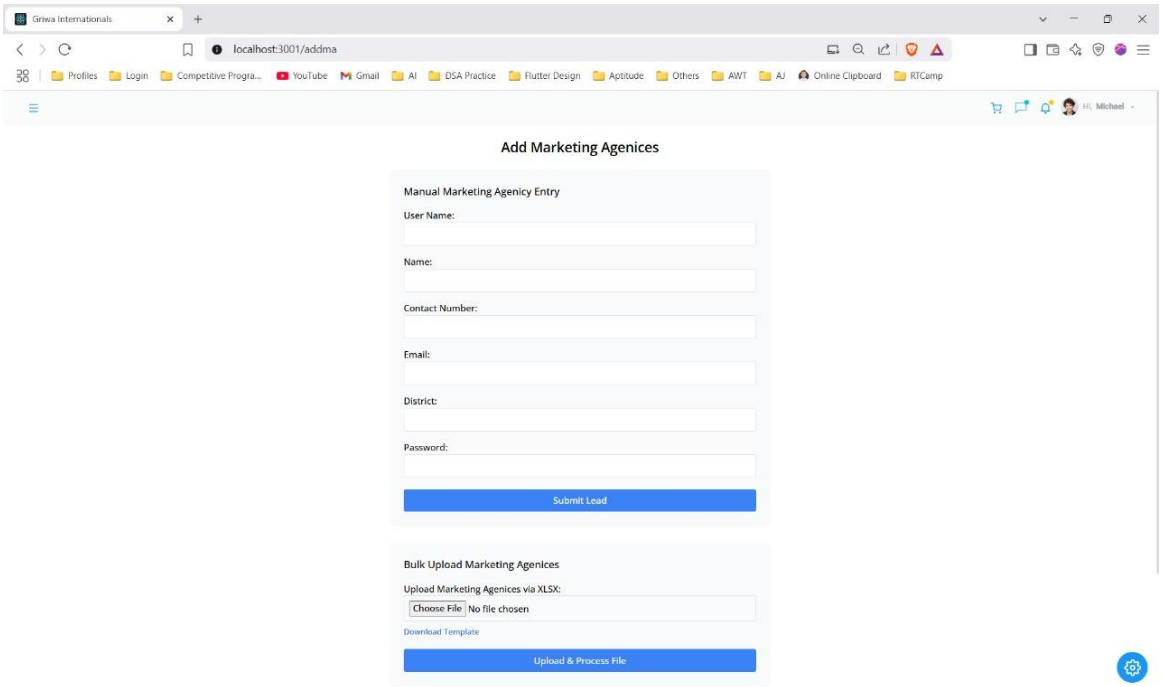
```
  const [formData, setFormData] = useState({
    user_name: '',
    name: '',
    email: '',
    phone: '',
```

```
    role: 'Employee',
    district: '',
    password: '',
  });

  const [errors, setErrors] = useState({});

  const validate = () => {
    const newErrors = {};
    if (!formData.user_name.trim()) newErrors.user_name = 'Username is
required';
    if (!formData.name.trim()) newErrors.name = 'Name is required';

    if (!formData.email.trim()) {
      newErrors.email = 'Email is required';
    } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
      newErrors.email = 'Invalid email format';
    }

    if (!formData.password || formData.password.length < 6) {
      newErrors.password = 'Password must be at least 6 characters';
    }

    if (!formData.phone.trim()) {
      newErrors.phone = 'Phone number is required';
    } else if (!/^\d{10}$/.test(formData.phone)) {
      newErrors.phone = 'Phone number must be 10 digits';
    }

    if (!formData.district.trim()) newErrors.district = 'District is
required';

    setErrors(newErrors);
    return Object.keys(newErrors).length === 0;
  };

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

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!validate()) return;

    axios
      .post(`${process.env.REACT_APP_BASE_URL}/api/user/`, formData)
      .then((res) => {
        alert(res.data.message);
        setFormData({
```

```
          user_name: '',
          name: '',
          email: '',
          phone: '',
          role: 'Employee',
          district: '',
          password: '',
        });
        setErrors({});
      })
      .catch((err) => {
        console.error(err);
        alert('Something went wrong during registration.');
      });
  };

  const handleFileUpload = (e) => {
    const file = e.target.files[0];
    if (!file) return;

    if (!file.name.endsWith('.xlsx') && !file.name.endsWith('.xls')) {
      alert('Please upload a valid Excel file (.xlsx or .xls)');
      return;
    }

    const reader = new FileReader();
    reader.onload = (evt) => {
      try {
        const workbook = XLSX.read(evt.target.result, { type: 'binary' });
        const sheetName = workbook.SheetNames[0];
        const worksheet = workbook.Sheets[sheetName];
        const jsonData = XLSX.utils.sheet_to_json(worksheet, { defval: '' });

        const updatedData = jsonData.map((row) => ({
          ...row,
          role: 'Employee',
          password: String(row.password), // convert password to string
        }));

        console.log(updatedData);

        axios
          .post(`${process.env.REACT_APP_BASE_URL}/api/user/excel`,
updatedData)
          .then((res) => alert(res.data.message))
          .catch((err) => {
            console.log(err);
            alert('Something went wrong during Excel upload.');
          });
      } catch (error) {
        console.error('Error reading file:', error);
```

```
    }
  };

  reader.readAsBinaryString(file);
};
```

9) Add Marketing Agency Page:



Add Marketing Agency with options to add manually or upload them via XLSX files.

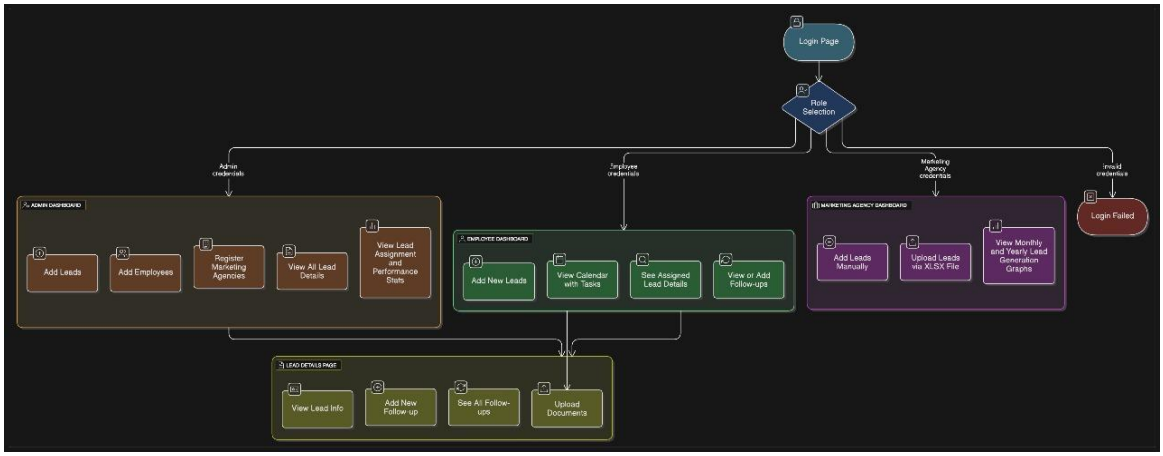
Code Snippet :-

```
const [formData, setFormData] = useState({
  user_name: '',
  name: '',
  email: '',
  phone: '',
  role: 'Marketing Agency',
  district: '',
  password: '',
});

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

const handleSubmit = (e) => {
  e.preventDefault();
  // console.log("Manually Submitted Lead:", formData);
```

```
    axios
      .post(`${process.env.REACT_APP_BASE_URL}/api/user/`, formData)
      .then((res) => {
        // eslint-disable-next-line no-alert
        alert(res.data.message);
        setFormData({
          client: '',
          date: '',
          contact_number: '',
          email: '',
          district: '',
          source: '',
        });
      })
      .catch((err) => {
        // eslint-disable-next-line no-console
        console.error(err);
        // eslint-disable-next-line no-alert
        alert('Something went wrong during registration.');
      });

    setFormData({
      user_name: '',
      name: '',
      email: '',
      phone: '',
      role: '',
      district: '',
      password: '',
    });
  };

  const handleFileUpload = (e) => {
    e.preventDefault();
    const file = e.target.files[0];
    if (!file) return;

    if (!file.name.endsWith('.xlsx') && !file.name.endsWith('.xls')) {
      alert('Please upload a valid Excel file (.xlsx or .xls)');
      return;
    }

    const reader = new FileReader();

    reader.onload = (evt) => {
      try {
        const bstr = evt.target.result;
        const workbook = XLSX.read(bstr, { type: 'binary' });

        const sheetName = workbook.SheetNames[0];
```

```
        const worksheet = workbook.Sheets[sheetName];
        const jsonData = XLSX.utils.sheet_to_json(worksheet, { defval: '' });
        const updatedData = jsonData.map((row) => ({
          ...row,
          role: 'Marketing Agency',
          password: String(row.password),
        }));

        console.log('Updated Data:', updatedData);

        axios
          .post(`${process.env.REACT_APP_BASE_URL}/api/user/excel`,
updatedData)
          .then((res) => {
            // eslint-disable-next-line no-alert
            alert(res.data.message);
          })
          .catch((err) => {
            // eslint-disable-next-line no-console
            console.error(err);
            // eslint-disable-next-line no-alert
            alert('Something went wrong during registration.');
          });
        // You can now send `updatedData` to your backend or use it as needed
      } catch (error) {
        console.error('Error reading file:', error);
      }
    };

    reader.readAsBinaryString(file);
  };
```

Flow Chart :-



- **GitHub**
    - **Frontend :-** https://github.com/ritzsanchala2607/Trackly
    - **Backend :-** https://github.com/Aryanlanghanoja/Trackly_Backend_Node

- **Deployed At**
    - **Frontend :-** https://trackly-five.vercel.app/
    - **Backend :-** https://trackly-backend-node.vercel.app/