# [TECHNICIAN ASSIGNMENT MANAGER]

**A Project Report for Industrial Training and Internship**

**submitted by**

**[SHANKAR KUMAR DHAR]**
**[SOMNATH GHOSH]**
**[ARPITA BHOWMICK]**
**[VIVEK SHAW]**

***In the partial fulfillment of the award of the degree of***

## BCA

in the

**[BCA(CSE)]**

**Of**

## [JIS UNIVERSITY]

## Ardent Computech Pvt. Ltd.

Ardent Computech Pvt. Ltd.
*Drives you to the Industry*
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

**CERTIFICATE FROM SUPERVISOR**

This is to certify that **"<Shankar Kumar Dhar, Somnath Ghosh, Arpita Bhowmick, Vivek Shaw>, <23CS2061135,23CS2061151,23CS2061036,23CS2061189>**" have completed the project titled "<Technician Assignment Manager>" under my supervision during the period from "<16/06/2025>" to "<05/07/2026>" which is in partial fulfillment of requirements for the award of the **BCA** degree and submitted to the Department of "**<BCA(CSE)>**" of **"<JIS University >"**.

_____

**Signature of the Supervisor**

**Date:** 05/07/2025

**Name of the Project Supervisor: Subhojit Santra**

Ardent Computech Pvt. Ltd.
*Drives you to the Industry*
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

# BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

***"<<Technician Assignment Manager>>"*** is the bonafide work of

***Name of the student: Shankar Kumar Dhar*** ***Signature:***

***Name of the student: Somnath Ghosh*** ***Signature:***

***Name of the student: Arpita Bhowmick*** ***Signature:***

***Name of the student: Vivek Shaw*** ***Signature:***

**SIGNATURE**

**Name:**

**PROJECT MENTOR**

**SIGNATURE**

**Name:**

**EXAMINERS**

**Ardent Original Seal**

## ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, ***[Subhojit Santra]*** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

# CONTENT PAGE

# 1. COMPANY PROFILE

ARDENT (Ardent Computech Pvt. Ltd.), formerly known as Ardent Computech Private Limited, is an ISO 9001:2015 certified Software Development and Training Company based in India. Operating independently since 2003, the organization has recently undergone a strategic merger with ARDENT Technologies, enhancing its global outreach and service offerings.

**ARDENT Technologies**

ARDENT Technologies delivers high-end IT services across the UK, USA, Canada, and India. Its core competencies lie in the development of customized application software, encompassing end-to-end solutions including system analysis, design, development, implementation, and training. The company also provides expert consultancy and electronic security solutions. Its clientele spans educational institutions, entertainment companies, resorts, theme parks, the service industry, telecom operators, media, and diverse business sectors.

**ARDENT Collaborations**

ARDENT Collaborations, the Research, Training, and Development division of ARDENT (Ardent Computech Pvt. Ltd.), offers professional IT-enabled services and industrial training programs. These are tailored for freshers and professionals from B.Tech, M.Tech, MBA, MCA, BCA, and MSc backgrounds. ARDENT (Ardent Computech Pvt. Ltd.) provides Summer Training, Winter Training, and Industrial Training to eligible candidates. High-performing students may qualify for stipends, scholarships, and additional benefits based on performance and mentor recommendations.

**Associations and Accreditations**

ARDENT (Ardent Computech Pvt. Ltd.) is affiliated with the National Council of Vocational Training (NCVT) under the Directorate General of Employment & Training (DGET), Ministry of Labour & Employment, Government of India. The institution upholds strict quality standards under ISO 9001:2015 certification and is dedicated to bridging the gap between academic knowledge and industry skills through innovative training programs.

# 2. INTRODUCTION

In any organization that relies on technical assist or discipline services, correctly coping with technician assignments is critical for making sure easy operations and welltimed provider shipping. A Technician Assignment Manager is a software device or machine designed to streamline the system of assigning obligations to technicians based on their availability, place, talents, and workload.

This machine ambitions to optimize resource allocation, lessen response times, and enhance customer pleasure. It helps supervisors or managers to music ongoing obligations, reveal technician performance, and reassign tasks as needed. By automating assignment distribution and retaining actual-time records, the Technician Assignment Manager enhances operational performance and reduces the danger of human error.

Whether utilized in IT assist, home services, manufacturing maintenance, or telecom sectors, this answer plays a critical role in boosting productivity and ensuring that the right technician is assigned to the right process on the proper time.

## 2A.OBJECTIVE

The essential goal of the Technician Assignment Manager is to successfully manipulate and automate the technique of assigning service or protection tasks to technicians based on their capabilities, availability, and place. This machine ambitions to :-Ensure short and correct technician venture.

Improve undertaking scheduling and time management.

Reduce guide mistakes in venture allocation.

Track technician repute and overall performance in real-time.

Optimize the use of available technical sources.

Provide managers with useful records and reviews for better choice-making.

Overall, the Technician Assignment Manager facilitates streamline operations, improve productivity, and deliver faster, smarter, and greater prepared technical assist or offerings.

## 2B.SCOPE

The Technician Assignment Manager gadget is designed to streamline and automate the system of assigning carrier obligations to technicians across diverse industries inclusive of IT support, telecommunications, home offerings, and device maintenance. The scope of this gadget consists of:

**User Management**: Allows directors to manipulate profiles of technicians, supervisors, and clients.

**Task Assignment:** Enables automatic or manual task of tasks based on technician availability, area, and ability set.

**Real-time Tracking:** Provides GPS tracking and stay fame updates of technician pastime and process development.

**Mobile Access:** Supports cellular interfaces for technicians to receive, replace, and close duties remotely.

**Notification System:** Sends signals and updates to technicians and clients thru SMS or electronic mail.

**Performance Monitoring:** Tracks key metrics inclusive of reaction time, mission finishing touch time, and technician efficiency.

**Data Integration:** Integrates with CRM, ERP, or inventory structures for easy facts flow and coordination.

**Reporting and Analytics:** Generates distinctive reports on technician performance, venture distribution, and carrier trends.

# 3. SYSTEM ANALYSIS

# 3A.IDENTIFICATION OF NEED

In now days's service-orientated industries, handling technical aid and subject services effectively is a primary venture. Many organizations nonetheless depend upon manual challenge assignments, smartphone calls, or paper-based schedules, which can result in delays, miscommunication, unbalanced workloads, and customer dissatisfaction.

The need for a Technician Assignment Manager arises due to the following key issues:

- **Inefficient Task Allocation:** Leads to delayed service response.
- **No Real-Time Monitoring:** Managers cannot track technician status or task progress.
- **Skill Mismatch:** Difficulty assigning tasks based on technician skills and availability.
- **High Customer Complaints:** Due to late service or missed appointments.
- **Low Manager Visibility:** Limited insight into technician workload and performance.
- **High Operational Costs:** Caused by unorganized and manual task management.

## 3B.FEASIBILITY STUDY

The Technician Assignment Manager is a possible undertaking across key regions:-

**Technical Feasibility:** Can be advanced the usage of current internet and mobile technologies with GPS and database aid.

**Operational Feasibility:** Streamlines task assignments, improves technician productivity, and complements provider first-class.

**Economic Feasibility:** Offers lengthy-time period value savings and high go back on funding via green useful resource use.

**Legal Feasibility:** Complies with facts privateness laws if security features are in location.

**Schedule Feasibility:** Can be advanced and deployed inside an inexpensive timeframe (3–6 months).

## 3C.WORKFLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirements of the system. It is meant for use by the developers and will be the basic during the testing phase. Any changes made to the requirements in the future will have to go through a formal change approval process.

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In the waterfall model phases do not overlap.

### Waterfall Model Design:

The waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure the success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In the Waterfall model, typically, the Outcome of one phase acts as the input for the next phase sequentially.

### Iterative Waterfall Design:

**Definition:** The Iterative Waterfall Model is a variation of the traditional Waterfall model, which is a linear and sequential software development methodology. In the Iterative Waterfall Model, the development process is divided into small, manageable cycles, allowing for the revisiting and refinement of phases before progressing to the next stage. It combines the systematic structure of the Waterfall model with the flexibility of iterative development.

The sequential phases in Iterative Waterfall model are:

- **Requirement Gathering and Analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

- **System Design:** The requirement specifications from the first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

➢ **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing each unit. Post integration the entire system is tested for any faults and failures.

➢ **Deployment of the system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

➢ **Maintenance:** Some issues come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in progress and are seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Iterative Waterfall Model". In this model, phases do not overlap.

- **Advantages:**

**1 . Flexibility:** Iterations permit adjustments based on feedback.

**2 . Early Delivery:** Partial systems can be delivered incrementally.

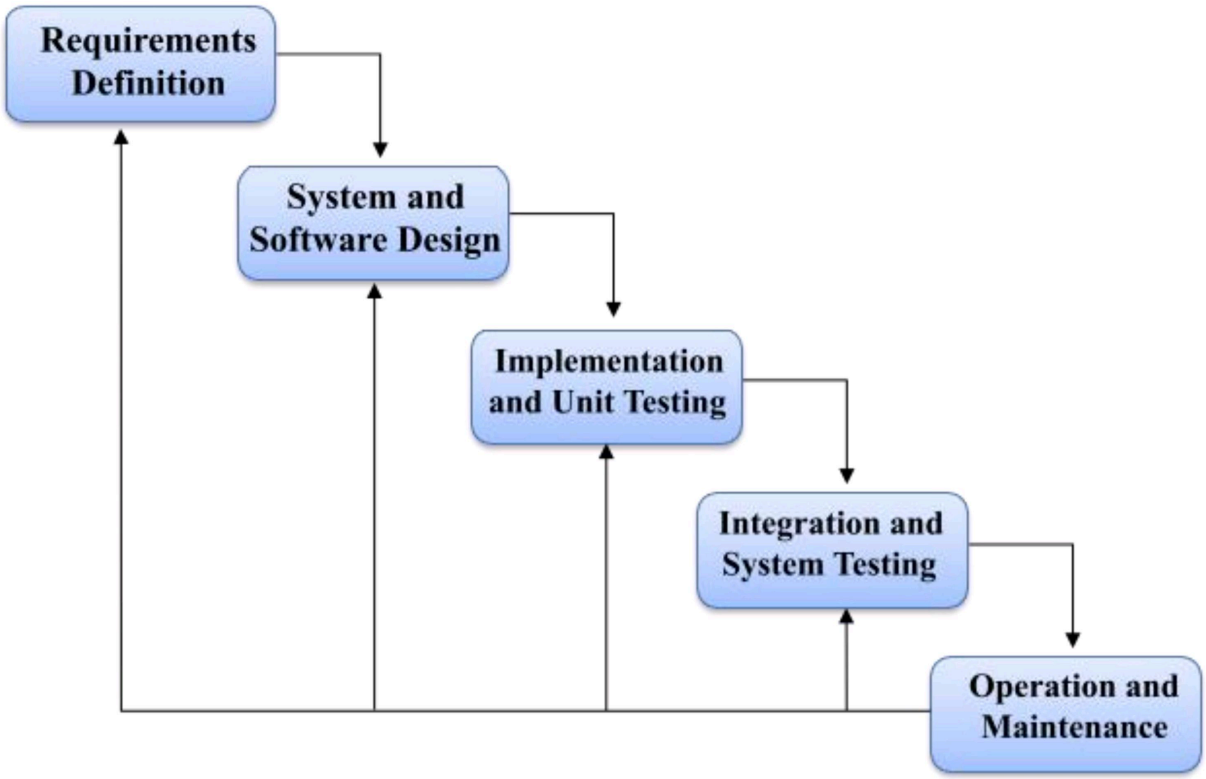
**3 . Risk Management:** Identifying and addressing issues early in the process.

- **Disadvantages:**

**1. Increased Complexity:** The iterative nature can make the process more complex.

**2. Potential for Scope Creep:** Frequent iterations may lead to scope changes.

**3. Resource Intensive:** Continuous revisiting of phases may demand more resources.




- **Applications:**

The Iterative Waterfall Model is suitable for projects with evolving or unclear requirements. It is commonly used in software development projects where regular feedback and refinement are essential.

Additionally, it is applicable in scenarios where partial system delivery is beneficial, allowing stakeholders to assess progress and make adjustments.

## 3D . STUDY OF THE SYSTEM – TECHNICIAN ASSIGNMENT MANAGER

Modules: The modules used in this software are as follows –

● Technician & Admin Registration:

1. **Technician Registration:** Technicians register by providing personal details, contact info, and skills.
2. **Admin Registration**: Admin registers to manage technician data, assignments, and system settings.

● Account Verification:

An OTP is sent to the registered email of the technician or admin during login to verify their identity and ensure secure access.

● Login Module:

1. Technician Login: Technicians log in to access their assigned tasks, update task status, and manage availability.
2. Admin Login: Admin logs in to assign tasks, manage users, and monitor system activities.

● Dashboard (Home):

Displays an overview of current tasks, technician performance, feedback received, and system statistics for Admin and Technicians.

● Task Management Module:

1. Technician Interface:
   - View list of assigned tasks with details like job type, location, deadline, and status.
   - Update task progress: Pending, In Progress, or Completed.
   - Search or filter tasks.
2. Admin Interface:
   - Assign tasks manually or based on skill/availability.
   - View, update, or delete tasks.

● Skill & Availability Module:

1. Technician Interface:
   - Update availability calendar and skills.
2. Admin Interface:
   - View and manage technician skillsets and availability for effective assignment.

● About Module:

Shows information about the Technician Assignment Manager system, purpose, developers, and background.

● Account Module:

1. Technician Interface:
    - My Profile: View personal, contact, and skill details.
    - Task Dashboard: Track active, pending, and completed tasks.
2. Admin Interface:
    - Admin Profile: View and edit admin account settings.
    - Admin Dashboard: Manage technicians, tasks, roles, and feedback.

## 3E. INPUT AND OUTPUT – TECHNICIAN ASSIGNMENT MANAGER

1. Main Inputs, Outputs, and Major Functional Details:

**INPUT:**

2. Technicians and Admins log in by entering their valid credentials (email/username and password) on the login page.

3. Admin inputs technician details such as skills, availability, and contact information while registering or updating technician profiles.

4. Admin inputs task details including job description, location, priority, and deadline to assign to a technician.

5. Technicians update task status (e.g., Pending, In Progress, Completed) as they work on their assigned tasks.

6. Technicians update their availability and skill information when needed.

**OUTPUT:**

7. Technicians can view their assigned tasks along with complete task details (task name, status, location, deadline).

8. Technicians can update task progress and receive confirmation/output messages for each update.

9. Admin can view and manage a centralized system dashboard that includes technician records, task status, and assignment history.

10. Admin can generate reports based on technician performance, task completion rates, and other system data.

11. Notifications or alerts may be shown for new task assignments, pending deadlines, or feedback messages.

# 3F.SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and non-functional requirements. The software requirements specification document enlists enough necessary requirements that are required for the project development. To derive the requirements we need to have a clear and thorough understanding of the project to be developed. This is prepared after detailed communication with the project team and the customer.

**The developer is responsible for:**

- Developing the system, which meets the SRS and solves all the requirements of the system.
- Demonstrating the system and installing the system at the client's location after acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

**A. Technician Registration and Authentication:**

1. Technicians and Admin users should be able to create accounts securely.
2. The system should authenticate users and manage secure login sessions with role-based access (Admin/Technician).

**B. Task Browse and Search:**

1. Admin should be able to browse, filter, and search technicians based on skills, availability, or location.
2. Technicians should be able to view a list of assigned tasks and search through them using relevant filters.

**C. Task Assignment and Display:**

1. Admin should be able to assign tasks to technicians manually or based on skill and availability.
2. Each technician should be able to view detailed task information (job type, location, deadline, status).
3. Technicians should be able to update task progress (e.g., Pending, In Progress, Completed).
4. Admin should have access to a dashboard displaying real-time status of all assignments.

**Hardware Requirements:**

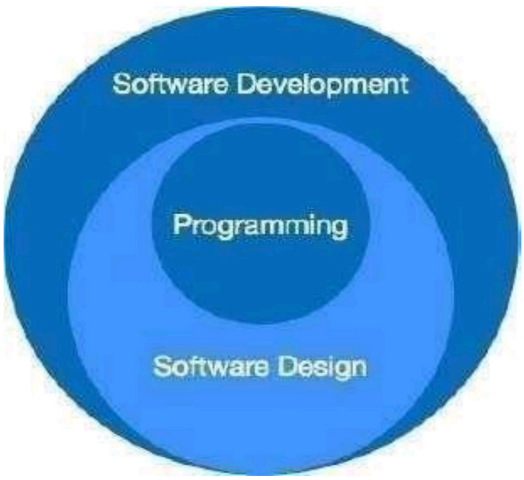
1 . Computer has Intel I3 Processor 2 .
8 GB RAM
3. SSD-ROM Drive

**Software Requirements:**

1. Windows 11 OS
2. Visual Studio Code
3. Mongo DB Atlas

## 3G.SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in software engineering these paradigms stand. These can be combined into various categories, though each of them is containedin one another.




The programming paradigm is a subset of Software design paradigm which is further a subset of the Software development paradigm.

There are two levels of reliability. The first is meeting the right requirements. A careful and thorough systems study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual work delivered to the user. At this level, the system's reliability is interwoven with software engineering and development.

There are three approaches to reliability.

1. **Error avoidance:** Prevents errors from occurring in software.
2. **Error detection and correction:** In this approach, errors are recognized whenever they are encountered, and correcting the error by the effect of the error of the system does not fail.
3. **Error to lerance:** In this approach, errors are recognized whenever they occur, but enables the system to keep running through degraded performance or Applying values that instruct the system to continue process.

# 4. SYSTEM DESIGN

## 4A. DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

DFD can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

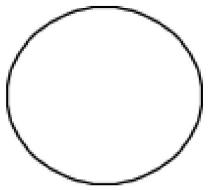
This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data-flow diagrams can be drawn up and compared with.

How any system is developed can be determined through a data flow diagram model. In the course of developing a set of leveled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow diagrams can be

used in both the Analysis and Design phase of the SDLC. There are different notations to draw data flow diagrams. Defining different visual representations for processes, data stores, data flow, and external entities.

## DFD Notation:



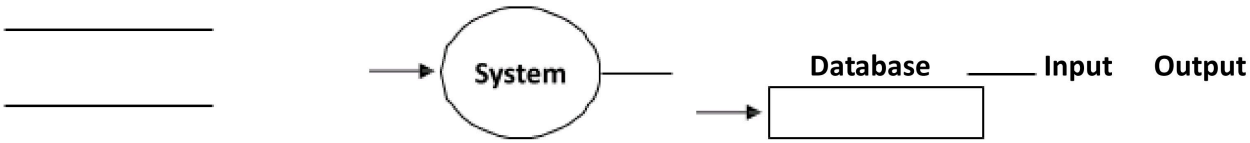
**Function**

**File / Database**

**Input / Output**

**Flow**

## DFD Example:




**Steps to Construct Data Flow Diagram:**

Four Steps are generally used to construct a DFD.

Process should be named and referred for easy reference. Each name should be representative of the reference.

The destination of flow is from top to bottom and from left to right.

When a process is distributed into lower-level details they are numbered.

The names of data stores, sources, and destinations are written in capital letters.

**Rules for constructing a Data Flow Diagram:**

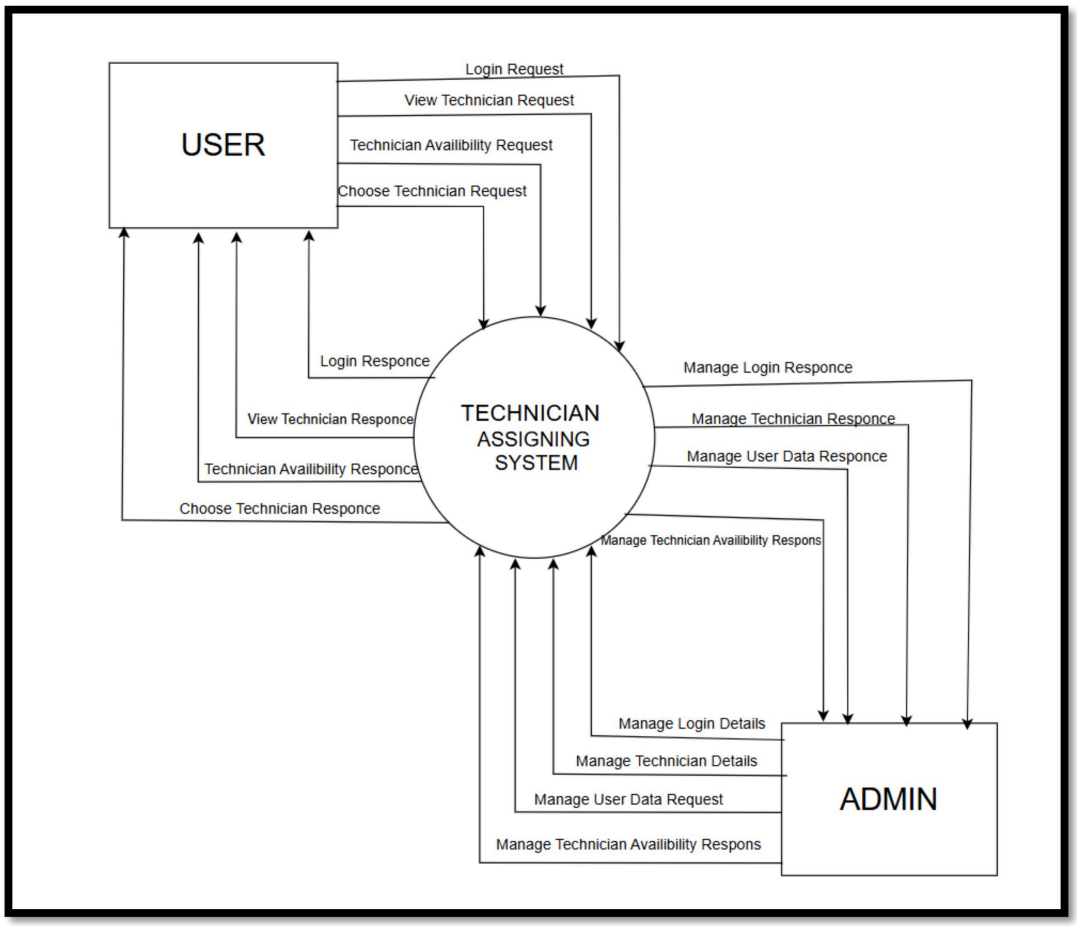
Arrows should not cross each other.

Squares, Circles, and Files must bear a name.

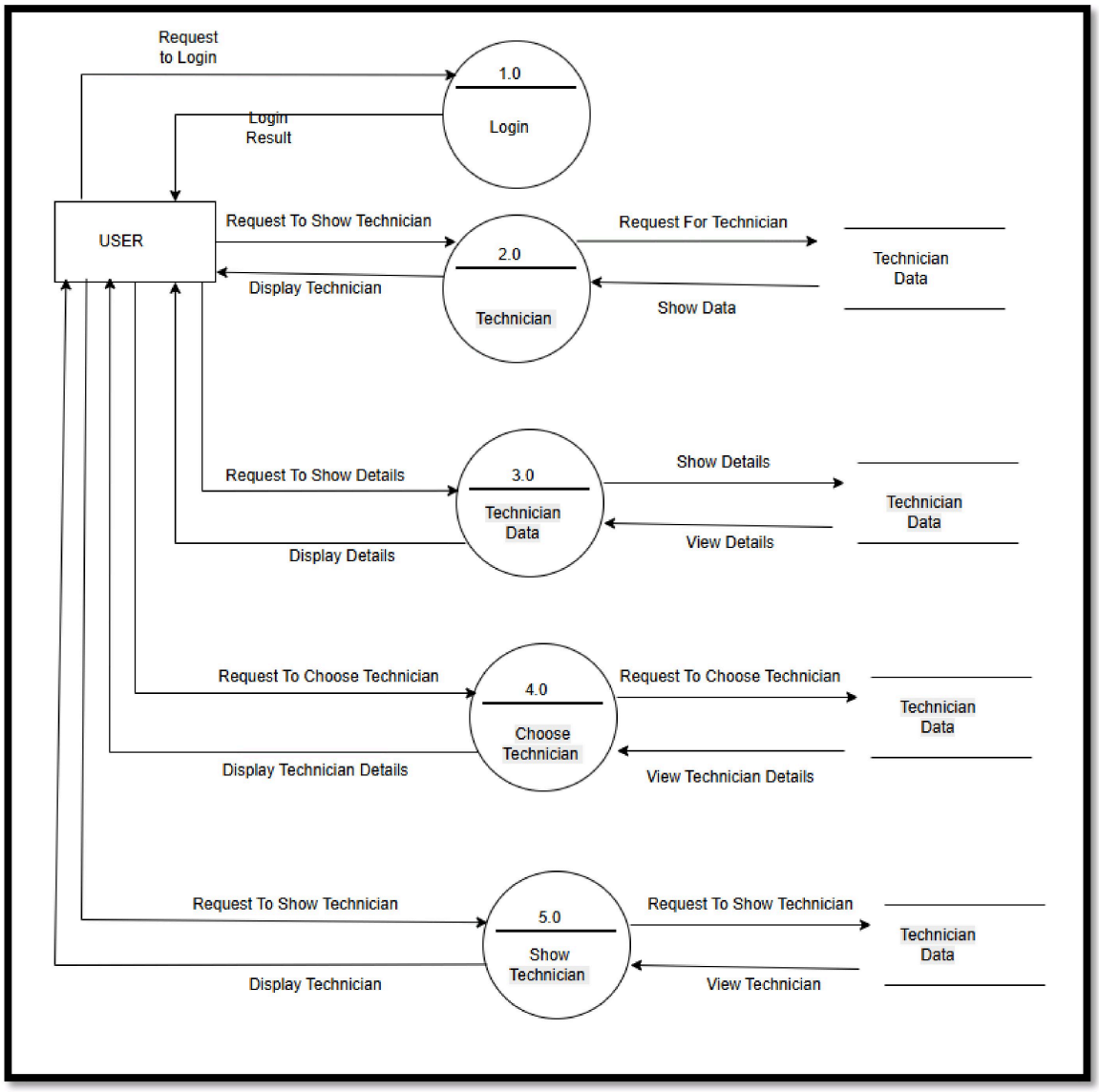
Decomposed data flow squares and circles can have the same names.

Draw all data flow around the outside of the diagram.

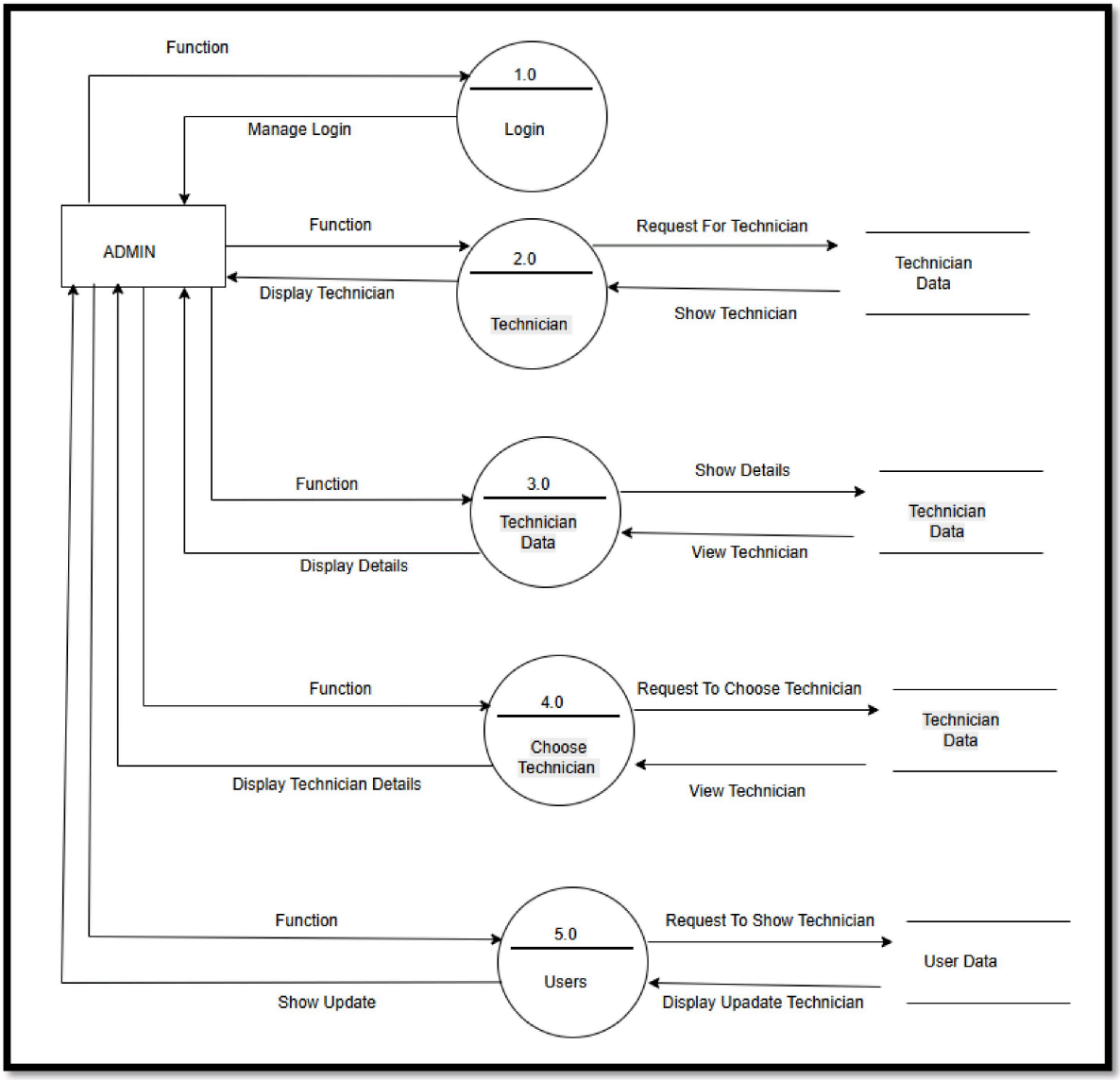
- **LEVEL 0 DFD OR CONTEXT DIAGRAM:**



● **LEVEL 1 DFD:**


Request
to Login
1.0
Login
Login
Result
USER
Request To Show Technician
Display Technician
2.0
Technician
Request For Technician
Show Data
Technician
Data
Request To Show Details
Display Details
3.0
Technician
Data
Show Details
View Details
Technician
Data
Request To Choose Technician
Display Technician Details
4.0
Choose
Technician
Request To Choose Technician
View Technician Details
Technician
Data
Request To Show Technician
Display Technician
5.0
Show
Technician
Request To Show Technician
View Technician
Technician
Data

**LEVEL 1 DFD:**


Function
1.0
Login
Manage Login
ADMIN
Function
2.0
Technician
Display Technician
Request For Technician
Show Technician
Technician Data
Function
3.0
Technician Data
Display Details
Show Details
View Technician
Technician Data
Function
4.0
Choose Technician
Display Technician Details
Request To Choose Technician
View Technician
Technician Data
Function
5.0
Users
Show Update
Request To Show Technician
Display Upadate Technician
User Data

## 4B.SEQUENCE DIAGRAM

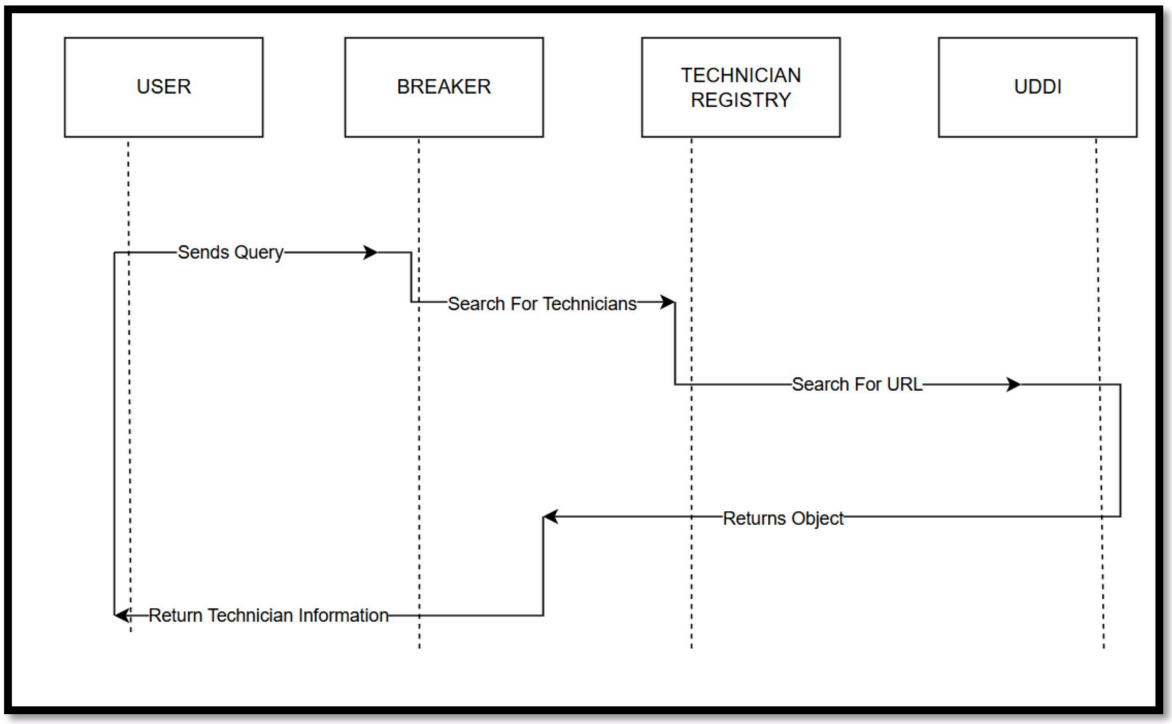
A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between several life lines .A sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, fragment, interaction, state invariant, continuation, and destruction occurrence.



A Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the use rand the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the system/subsystem of an application. A single-use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So, we will look into some specific purpose that will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modeled to present the outside view. So, in brief, the purposes of use case diagrams can be as follows:

Used to gather requirements of a system.

Used to get an outside view of a system.

Identify external and internal factors influencing the system.

**How to draw Use Case Diagram?**

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases.

So, we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So, in a brief when we are planning to draw use case diagram, we should have the following items identified.

Functionalities to be represented as a use case

Actors

Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So, after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

The name of a use case is very important. So, the name should be chosen in such a way so that it can identify the functionalities performed.

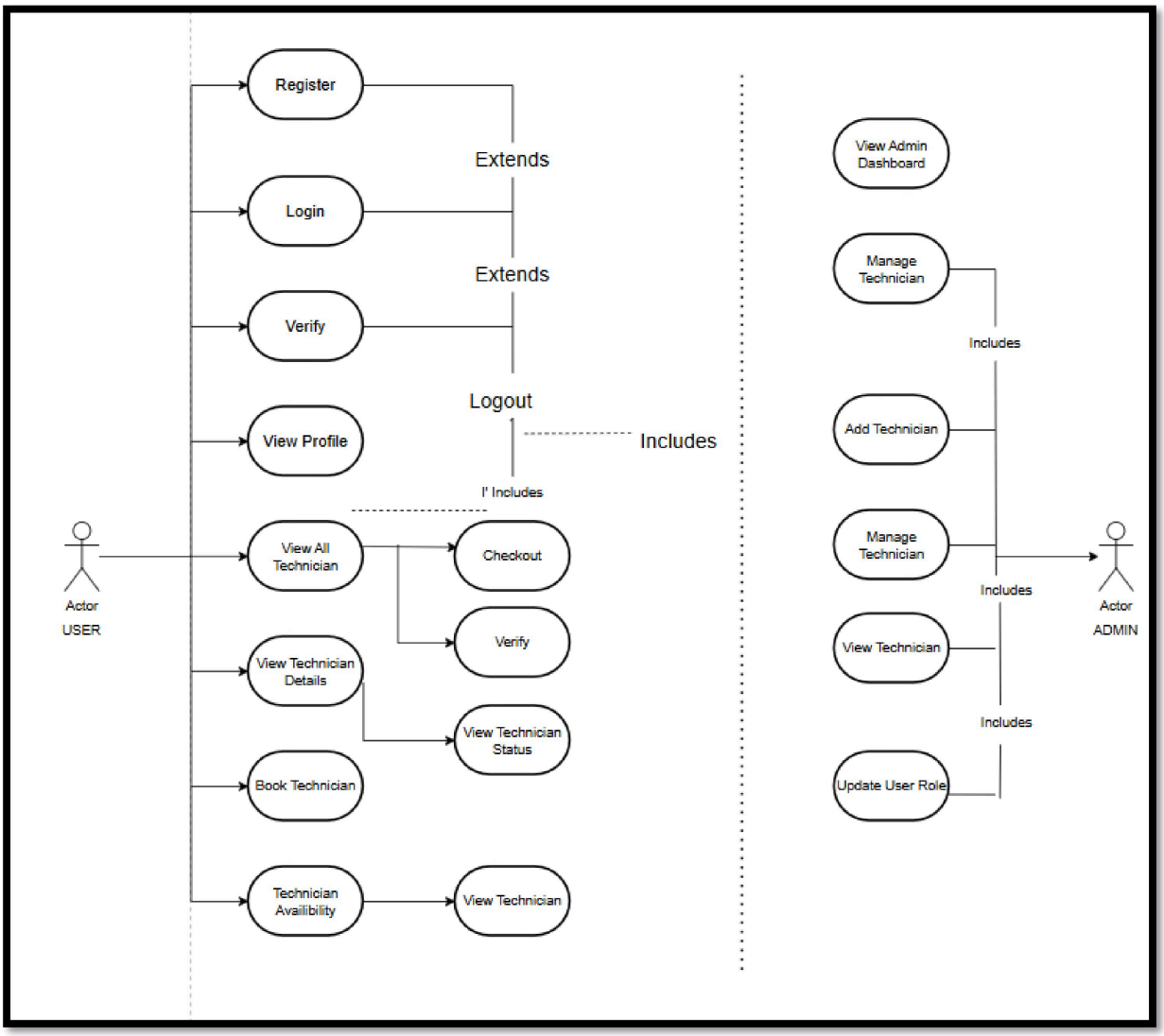
Give a suitable name for actors.

Show relationships and dependencies clearly in the diagram.

Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.

Use note whenever required to clarify some important point

● **USE CASE DIAGRAM:**

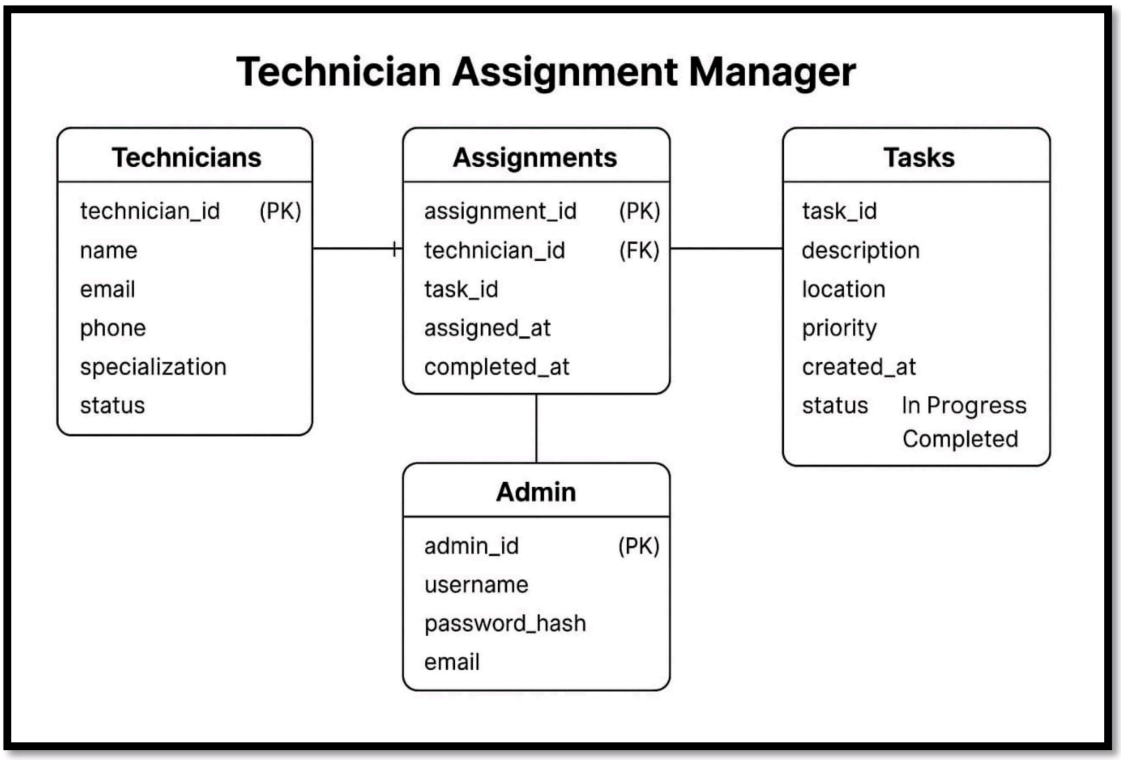


## 4D.SCHEMA DIAGRAM

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

DB schema design organizes data into separate entities, determines how to create relationships between organized entities, and influences the applications of constraints on data. Designers create database schema to give other database users, such as programmers and analysts, a logical understanding of data.

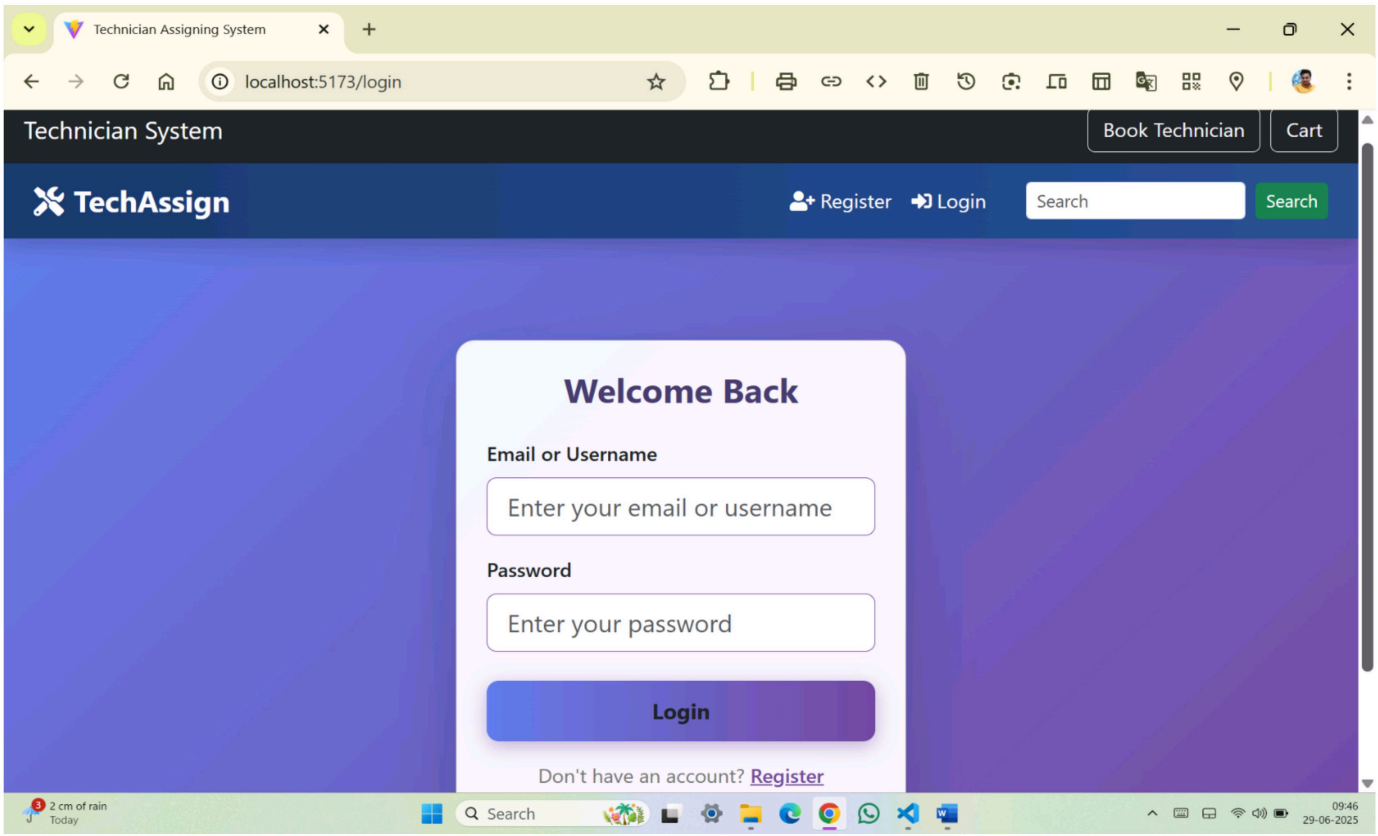
- **SCHEMA DESIGN:**



# 5. UI SNAPSHOT

**FRONTEND : -**

**1) Login Page:**




**CODE**

```
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();
```

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

    if (email === 'admin' && password === 'admin') {
      localStorage.setItem('isAdmin', 'true');
      localStorage.setItem('user', JSON.stringify({ name: 'Admin' }));
      navigate('/admin-dashboard');
      return;
    }

    try {
      const res = await axios.post('http://localhost:6500/api/auth/login', { email,
password });

      if (res.data.token) {
        localStorage.setItem('token', res.data.token);
        localStorage.setItem('user', JSON.stringify(res.data.user));
        localStorage.setItem('isAdmin', 'false');
        navigate('/home');
      } else {
        alert('Login failed. Please try again.');
      }
    } catch (err) {
      alert('Login failed. Invalid credentials or server error.');
      console.error(err);
    }
  };

  return (
    <div
      className="d-flex align-items-center justify-content-center vh-100"
      style={{
```

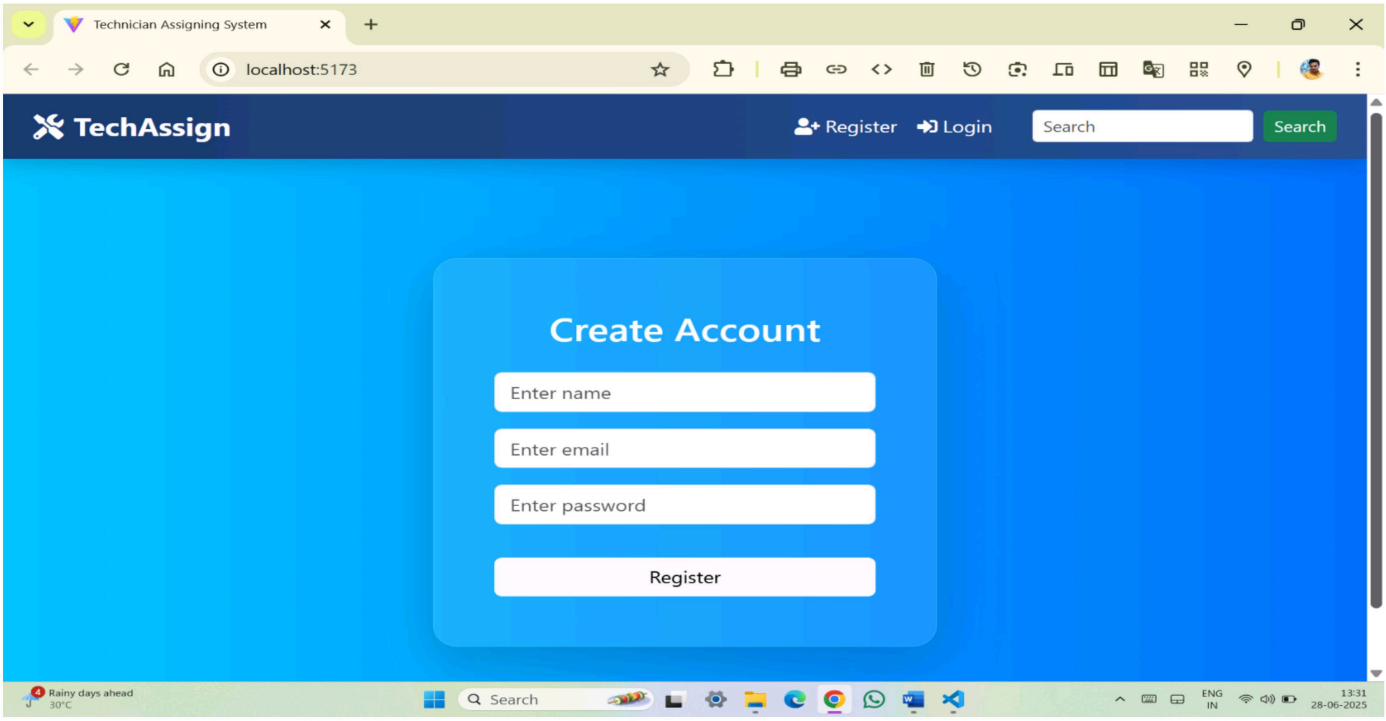
```
        background: 'linear-gradient(135deg, #667eea 0%, #764ba2 100%)',
        fontFamily: "'Segoe UI', Tahoma, Geneva, Verdana, sans-serif",
      }}
    >
      <div
        className="card p-4 shadow-lg"
        style={{
          width: '360px',
          borderRadius: '15px',
          backgroundColor: 'rgba(255, 255, 255, 0.95)',
        }}
      >
        <h3 className="text-center mb-4" style={{ color: '#4b3f72', fontWeight:
'700' }}>
          Welcome Back
        </h3>
        <form onSubmit={handleLogin}>
          <div className="form-group mb-3">
            <label htmlFor="email" className="form-label" style={{ fontWeight: '600'
}}>
              Email or Username
            </label>
            <input
              id="email"
              type="text"
              className="form-control form-control-lg"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              placeholder="Enter your email or username"
              required
              style={{
                borderRadius: '8px',
                borderColor: '#764ba2',
```

```
                transition: 'border-color 0.3s',
              }}
              onFocus={(e) => (e.target.style.borderColor = '#667eea')}
              onBlur={(e) => (e.target.style.borderColor = '#764ba2')}
            />
          </div>
          <div className="form-group mb-4">
            <label htmlFor="password" className="form-label" style={{ fontWeight:
'600' }}>
              Password
            </label>
            <input
              id="password"
              type="password"
              className="form-control form-control-lg"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              placeholder="Enter your password"
              required
              style={{
                borderRadius: '8px',
                borderColor: '#764ba2',
                transition: 'border-color 0.3s',
              }}
              onFocus={(e) => (e.target.style.borderColor = '#667eea')}
              onBlur={(e) => (e.target.style.borderColor = '#764ba2')}
            />
          </div>
          <button
            type="submit"
            className="btn btn-gradient w-100 fw-bold"
            style={{
```

```
              background:
                'linear-gradient(90deg, #667eea 0%, #764ba2 100%)',
              border: 'none',
              borderRadius: '10px',
              padding: '12px',
              fontSize: '1.1rem',
              boxShadow: '0 4px 15px rgba(118, 75, 162, 0.4)',
              transition: 'background 0.3s',
            }}
            onMouseEnter={(e) =>
              (e.target.style.background =
                'linear-gradient(90deg, #764ba2 0%, #667eea 100%)')
            }
            onMouseLeave={(e) =>
              (e.target.style.background =
                'linear-gradient(90deg, #667eea 0%, #764ba2 100%)')
            }
          >
            Login
          </button>
        </form>
        <p className="text-center mt-3 mb-0" style={{ color: '#777' }}>
          Don't have an account?{' '}
          <a href="/register" style={{ color: '#764ba2', fontWeight: '600' }}>
            Register
          </a>
        </p>
      </div>
    </div>
  );
};
```

```
export default Login;
```

**Register Page:**




**CODE**

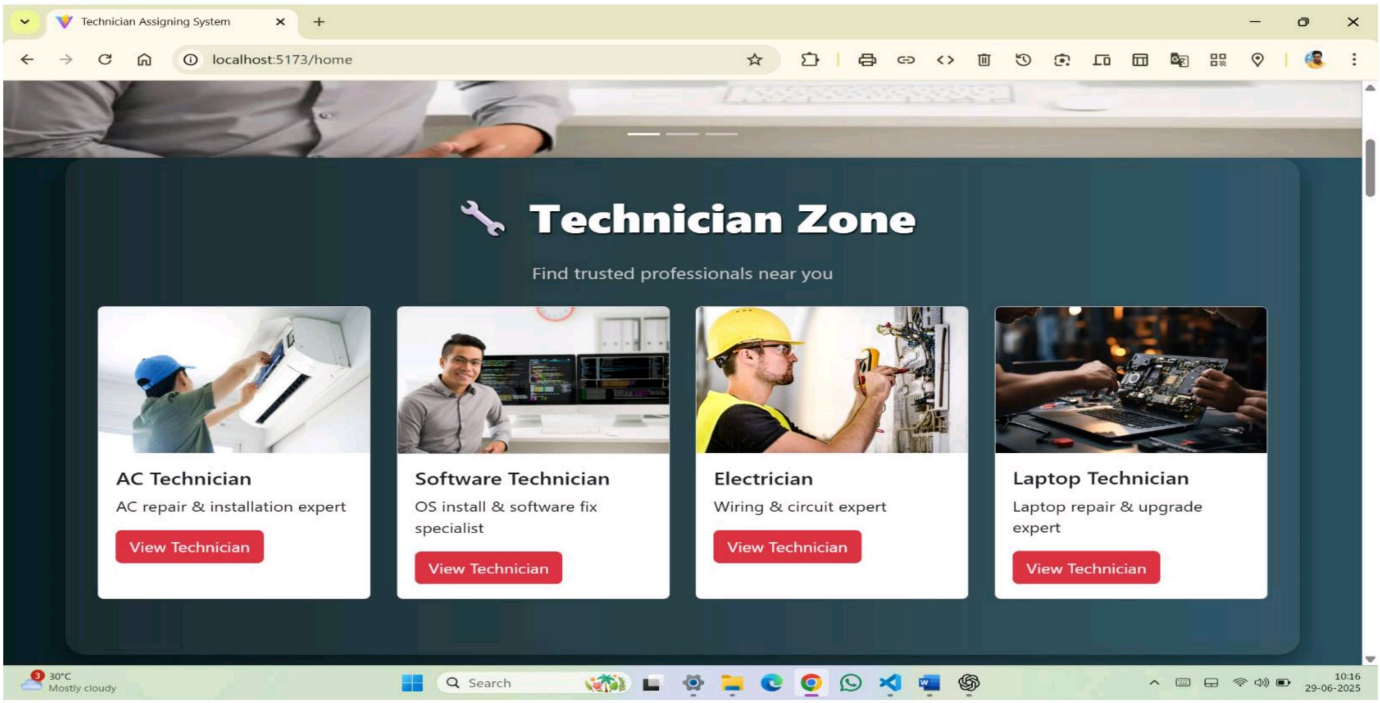
```
import React, { useState } from 'react';
import axios from 'axios';

const Register = () => {
  const [form, setForm] = useState({ name: "", email: "", password: "" });

  const hc = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

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

```
    await axios.post('http://localhost:6500/api/auth/register', form);
    alert('Registration Successful');
  } catch (error) {
    alert('Registration Failed');
  }
};

return (
  <div
    className="d-flex align-items-center justify-content-center vh-100"
    style={{
      background: 'linear-gradient(to right, #00c6ff, #0072ff)',
      fontFamily: "'Segoe UI', sans-serif"
    }}
  >
    <form
      onSubmit={hs}
      className="p-5 shadow-lg"
      style={{
        background: 'rgba(255, 255, 255, 0.1)',
        backdropFilter: 'blur(10px)',
        borderRadius: '20px',
        color: '#fff',
        width: '100%',
        maxWidth: '400px',
        border: '1px solid rgba(255, 255, 255, 0.2)'
      }}
    >
      <h2 className="text-center mb-4">Create Account</h2>

      <div className="form-group">
        <input
```

```
    type="text"
    name="name"
    placeholder="Enter name"
    onChange={hc}
    required
    className="form-control"
  />
</div>

<div className="form-group">
  <input
    type="email"
    name="email"
    placeholder="Enter email"
    onChange={hc}
    required
    className="form-control"
  />
</div>

<div className="form-group">
  <input
    type="password"
    name="password"
    placeholder="Enter password"
    onChange={hc}
    required
    className="form-control"
  />
</div>

<button type="submit" className="btn btn-light btn-block mt-3">
```

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

export default Register;
```

### 3) Technician Main Page(user):




- **components-pages-Cardlist.jsx**

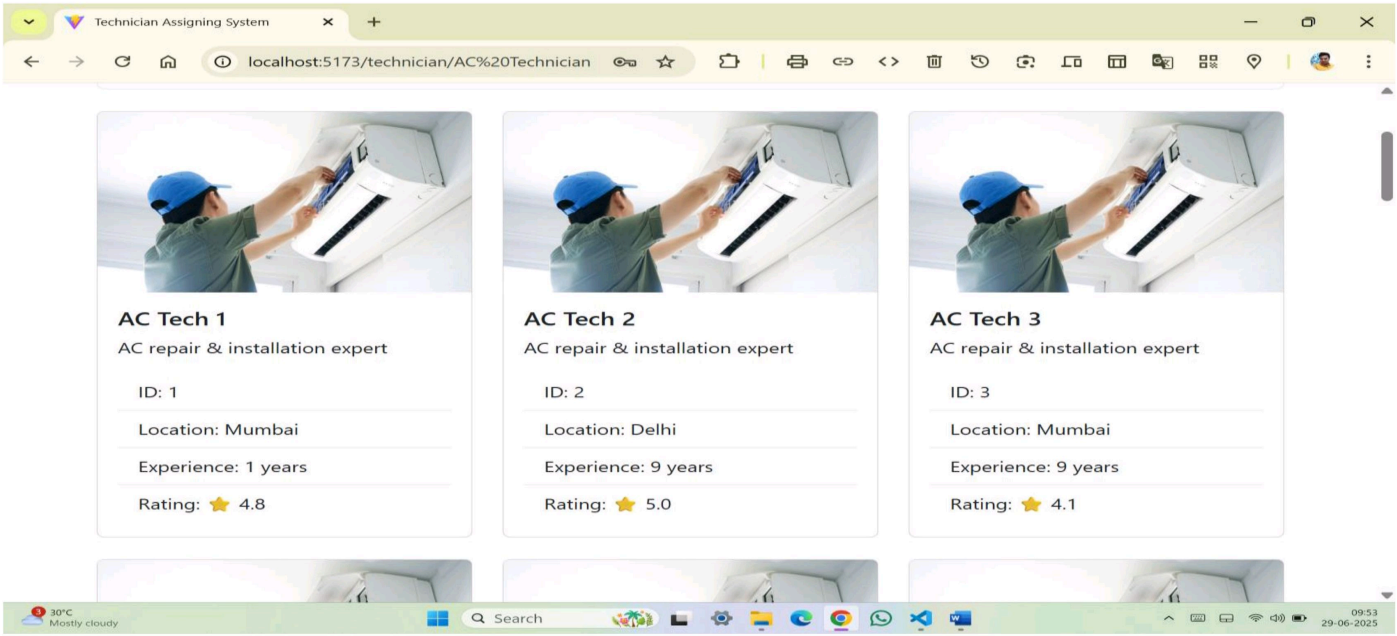
```
 import React from 'react';
import { useNavigate } from 'react-
  router-dom';

const Cardlist = ({ data }) => {
  const navigate = useNavigate();
    const uniqueTypes = [...new
  Set(data.map((tech) => tech.type))];
```

```
  return (
    <div className="row">
       {uniqueTypes.map((type, index)
  => {
        const tech = data.find((t) =>
  t.type === type);
        return (
          <div className="col-md-3 mb-
  4" key={index}>
              <div className="card h-
  100">
                <img src={tech.image}
  className="card-img-top"
  alt={tech.type} />
                <div className="card-
  body">
                  <h5 className="card-
  title">{tech.type}</h5>
                    <p className="card-
  text">{tech.desc}</p>
                <button
                   className="btn btn-
  danger"
                        onClick={() =>
  navigate(`/technician/${encodeURICo
  mponent(type)}`)}
                >
                  View Technician
                </button>
              </div>
            </div>
          </div>
        );
      })}
    </div>
  );
};

export default Cardlist;
```

- **components – Carouselslider.jsx :-**

```
import React from 'react';

const Carouselslider = () => {
  return (
    <div id="carouselExampleCaptions" className="carousel slide" data-
ride="carousel">
      <ol className="carousel-indicators">
        <li data-target="#carouselExampleCaptions" data-slide-to="0"
className="active"></li>
        <li data-target="#carouselExampleCaptions" data-slide-to="1"></li>
        <li data-target="#carouselExampleCaptions" data-slide-to="2"></li>
      </ol>
      <div className="carousel-inner">
        <div className="carousel-item active">
          <img src="./src/assets/img2.jpg" className="d-block w-100"
width={1560} height={450} alt="Slide 1" />
        </div>
        <div className="carousel-item">
          <img src="./src/assets/img1.jpg" className="d-block w-100"
width={1560} height={450} alt="Slide 2" />
        </div>
        <div className="carousel-item">
          <img src="./src/assets/img4.jpg" className="d-block w-100"
width={1560} height={450} alt="Slide 3" />
        </div>
      </div>
      <button className="carousel-control-prev" type="button" data-
target="#carouselExampleCaptions" data-slide="prev">
        <span className="carousel-control-prev-icon" aria-
hidden="true"></span>
        <span className="sr-only">Previous</span>
      </button>
      <button className="carousel-control-next" type="button" data-
target="#carouselExampleCaptions" data-slide="next">
        <span className="carousel-control-next-icon" aria-
hidden="true"></span>
        <span className="sr-only">Next</span>
      </button>
    </div>
  );
};
```

## 4) TECHNICIAN DESCRIPTION PAGE AND SYSTEM(for user);



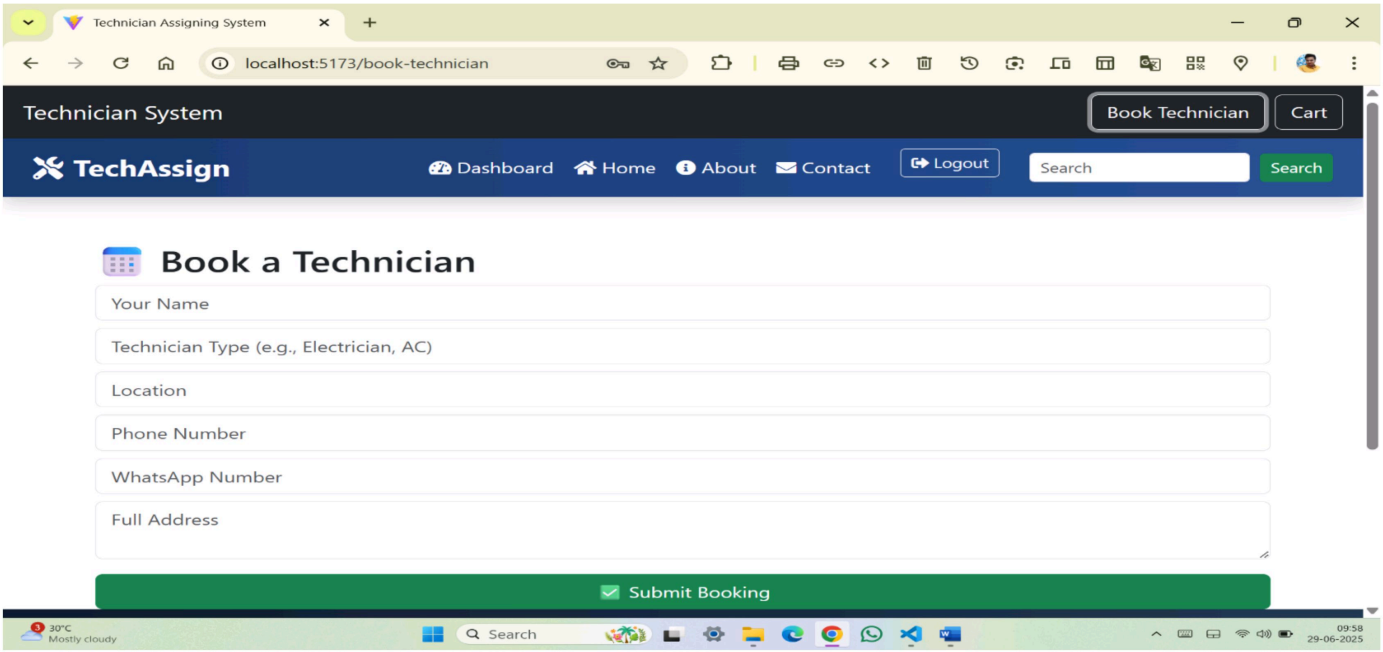

**CODE**

```
  import React, { useState } from 'react';
import { useParams } from 'react-router-dom';
import citydata from '../data/citydata';

const TechnicianDetail = () => {
  const { type } = useParams();
  const decodedType = decodeURIComponent(type);
  const [searchTerm, setSearchTerm] = useState('');

  // Filter by selected type (Electrician, AC, etc.)
  const filteredByType = citydata.filter((x) => x.type === decodedType);

  // Further filter by search input (case insensitive)
  const filtered = filteredByType.filter((tech) =>
    tech.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
    tech.location.toLowerCase().includes(searchTerm.toLowerCase())
  );
```

```
  return (
    <div className="container mt-5">
      <h2 className="text-center mb-4">{decodedType} Technicians</h2>

      {/* ✅ Search Bar */}
      <input
        type="text"
        className="form-control mb-4"
        placeholder="🔍 Search by name or location..."
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
      />

      {filtered.length === 0 ? (
        <h4 className="text-center text-danger">No Technicians Found</h4>
      ) : (
        <div className="row">
          {filtered.map((tech) => (
            <div className="col-md-4 mb-4" key={tech.id}>
              <div className="card h-100">
                <img src={tech.image} className="card-img-top" alt={tech.name} />
                <div className="card-body">
                  <h5 className="card-title">{tech.name}</h5>
                  <p className="card-text">{tech.desc}</p>
                  <ul className="list-group list-group-flush">
                    <li className="list-group-item">ID: {tech.id}</li>
                    <li className="list-group-item">Location: {tech.location}</li>
                    <li className="list-group-item">Experience: {tech.experience}
years</li>
                    <li className="list-group-item">Rating: ☆ {tech.rating}</li>
                  </ul>
```

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

export default TechnicianDetail;
```

### 5) BOOK TECHNICIAN PAGE:



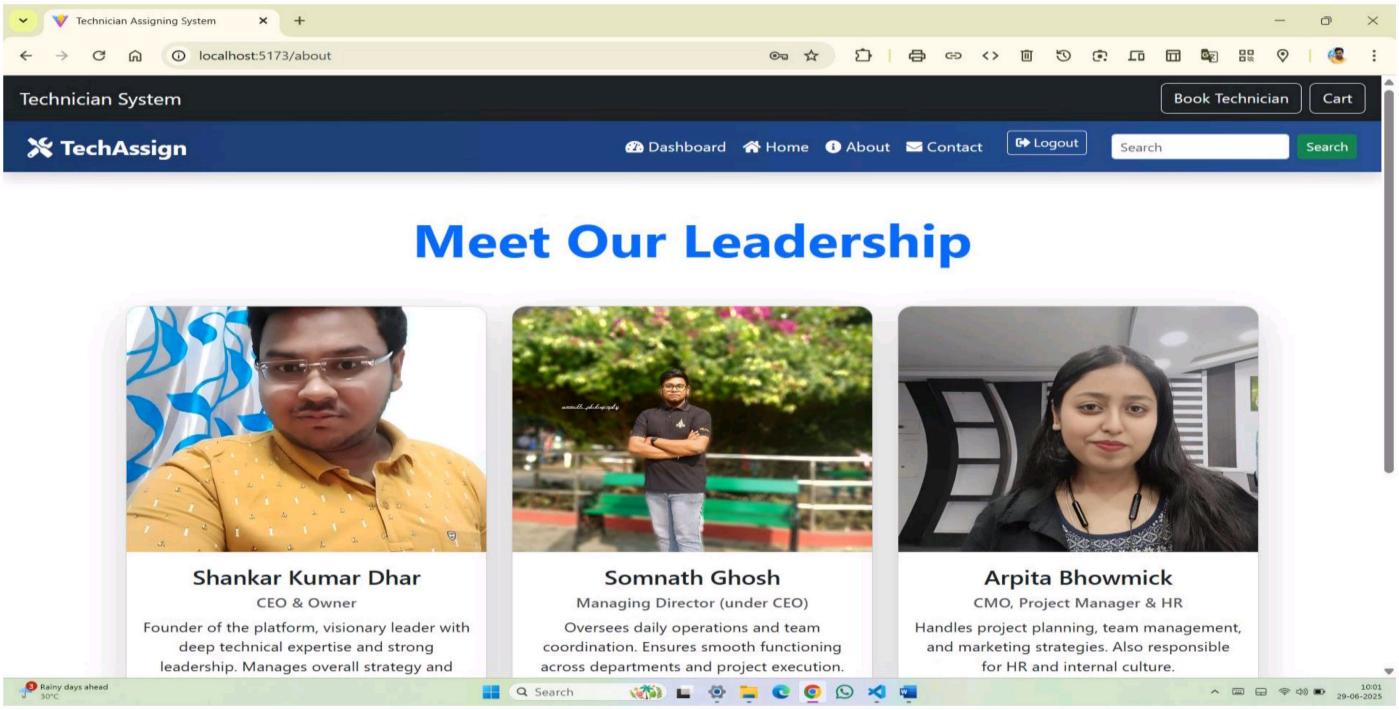

### CODE

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

const BookTechnician = () => {
  const [form, setForm] = useState({
```

```
  name: '',
  technicianType: '',
  location: '',
  phone: '',
  wp: '',
  address: '',
});

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

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

  const user = localStorage.getItem('user');
  if (!user) {
    alert('⚠️ Please login to book a technician.');
    return;
  }

  try {
    const response = await axios.post('http://localhost:6500/api/bookings', form);
    if (response.status === 200 || response.status === 201) {
      alert('✅ Booking successful!');
      setForm({
        name: '',
        technicianType: '',
        location: '',
        phone: '',
        wp: '',
        address: '',
      });
    }
  } catch (err) {
    console.error(err);
    alert('❌ Error booking technician. Please try again.');
  }
};

return (
  <div className="container mt-5">
    <h2>📅 Book a Technician</h2>
    <form onSubmit={handleSubmit}>
      <input
        className="form-control mb-2"
        name="name"
        value={form.name}
        placeholder="Your Name"
        onChange={handleChange}
```

```
          required
        />
        <input
          className="form-control mb-2"
          name="technicianType"
          value={form.technicianType}
          placeholder="Technician Type (e.g., Electrician, AC)"
          onChange={handleChange}
          required
        />
        <input
          className="form-control mb-2"
          name="location"
          value={form.location}
          placeholder="Location"
          onChange={handleChange}
          required
        />
        <input
          className="form-control mb-2"
          name="phone"
          value={form.phone}
          placeholder="Phone Number"
          onChange={handleChange}
          required
        />
        <input
          className="form-control mb-2"
          name="wp"
          value={form.wp}
          placeholder="WhatsApp Number"
          onChange={handleChange}
          required
        />
        <textarea
          className="form-control mb-3"
          name="address"
          value={form.address}
          placeholder="Full Address"
          onChange={handleChange}
          required
        />
        <button type="submit" className="btn btn-success w-100">✅ Submit
Booking</button>
      </form>
    </div>
  );
};

export default BookTechnician;
```

## 6) About Page:



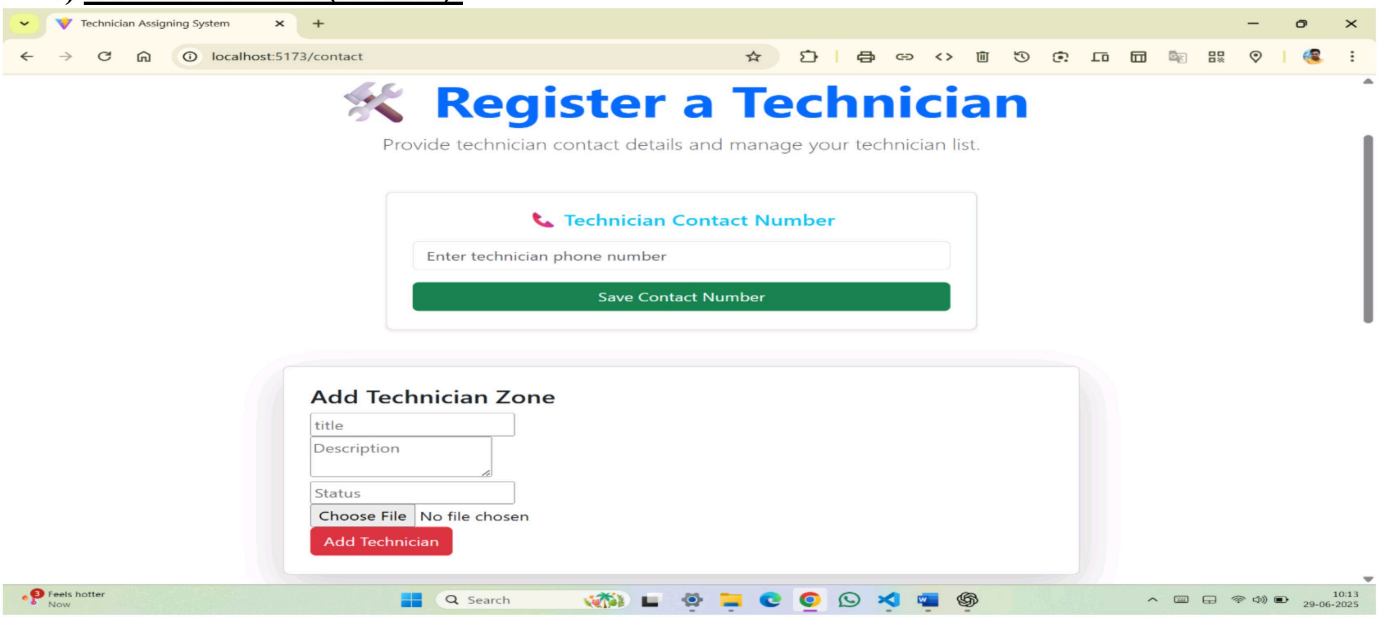

### CODE

```
    import React from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import ceoImg from '../assets/ceo.jpg';          // Add your image in src/assets
import mdImg from '../assets/md.jpg';
import cmoImg from '../assets/cmo.jpg';

const About = () => {
  const team = [
    {
      name: 'Shankar Kumar Dhar',
      title: 'CEO & Owner',
      image: ceoImg,
      description:
        'Founder of the platform, visionary leader with deep technical expertise and
strong leadership. Manages overall strategy and growth.',
    },
    {
```

```
      name: 'Somnath Ghosh',
      title: 'Managing Director (under CEO)',
      image: mdImg,
      description:
        'Oversees daily operations and team coordination. Ensures smooth functioning
across departments and project execution.',
    },
    {
      name: 'Arpita Bhowmick',
      title: 'CMO, Project Manager & HR',
      image: cmoImg,
      description:
        'Handles project planning, team management, and marketing strategies. Also
responsible for HR and internal culture.',
    },
  ];

  return (
    <div className="container py-5">
      <h1 className="text-center mb-5 display-4 fw-bold text-primary">Meet Our
Leadership</h1>
      <div className="row justify-content-center">
        {team.map((member, index) => (
          <div className="col-md-4 mb-4" key={index}>
            <div
              className="card shadow-lg h-100"
              style={{
                borderRadius: '15px',
                overflow: 'hidden',
              }}
            >
              <img
                src={member.image}
                className="card-img-top"
                alt={member.name}
                style={{ height: '300px', objectFit: 'cover' }}
              />
              <div className="card-body text-center">
                <h4 className="card-title">{member.name}</h4>
```

```
                <h6 className="text-muted">{member.title}</h6>
                <p className="card-text mt-2">{member.description}</p>
              </div>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

export default About;
```

### 7) CONTACT PAGE(for user):



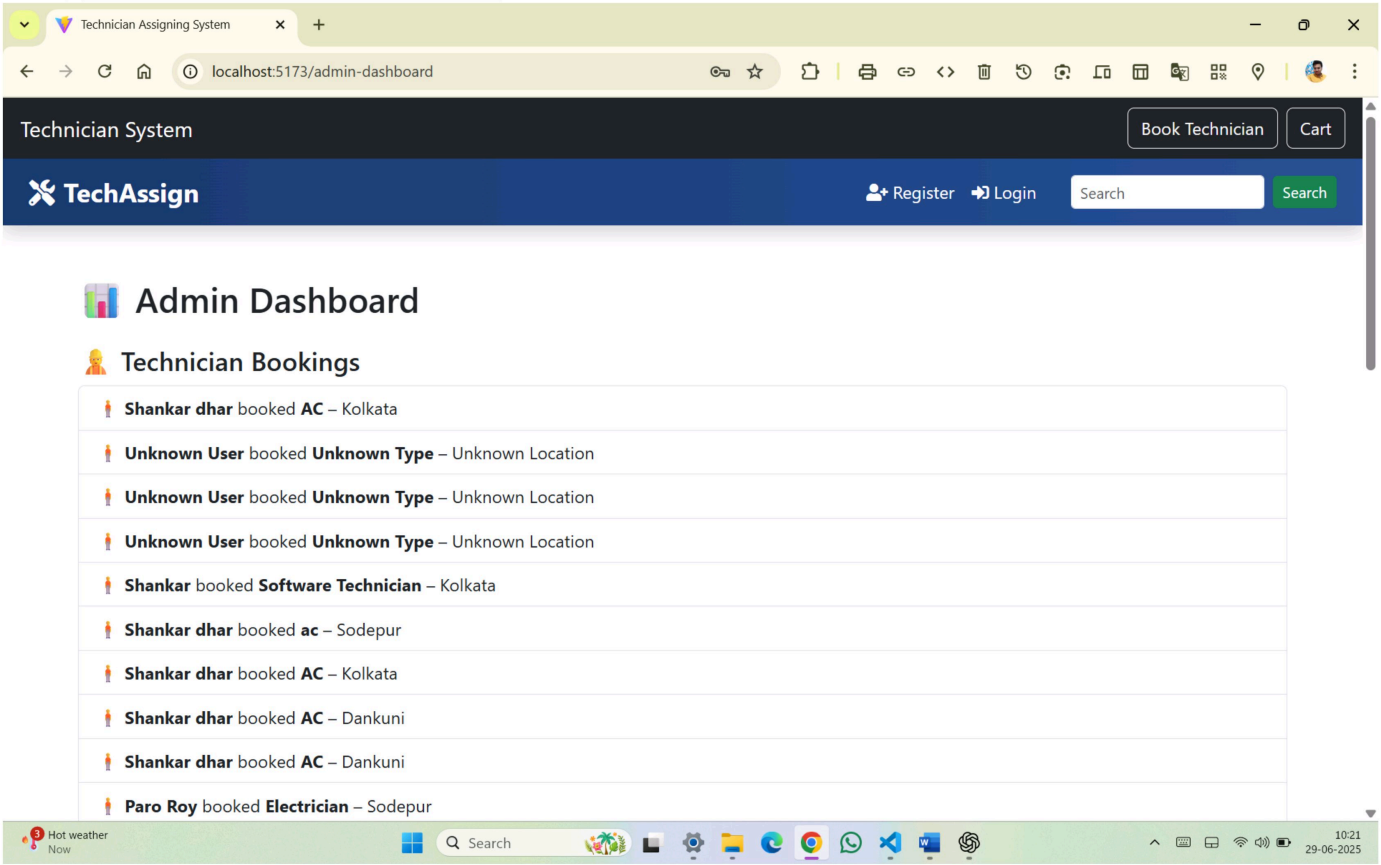

✓ **CODE**

```
import React, { useState } from 'react';
import TechnicianForm from './CityForm';  // renamed from Cityform
import TechnicianList from './CityList';  // renamed from Citylist

const Contact = () => {
  const [phone, setPhone] = useState('');

  const handlePhoneSubmit = (e) => {
    e.preventDefault();
    if (!phone.trim()) return alert("Please enter a valid phone number.");
    alert(`Technician phone number submitted: ${phone}`);
```

```
    setPhone('');
  };

  return (
    <div className="container my-5">
      <div className="text-center mb-5">
        <h2 className="display-4 font-weight-bold text-primary">🛠 Register a
Technician</h2>
        <p className="lead text-muted">Provide technician contact details and manage
your technician list.</p>
      </div>

      {/* Contact Number Form */}
      <div className="row justify-content-center mb-5">
        <div className="col-md-6">
          <div className="card p-4 shadow-sm">
            <h5 className="mb-3 text-center text-info">📞 Technician Contact
Number</h5>
            <form onSubmit={handlePhoneSubmit}>
              <div className="form-group">
                <input
                  type="tel"
                  className="form-control"
                  placeholder="Enter technician phone number"
                  value={phone}
                  onChange={(e) => setPhone(e.target.value)}
                  required
                />
              </div>
              <button type="submit" className="btn btn-success btn-block">
                Save Contact Number
              </button>
            </form>
          </div>
        </div>
      </div>

      {/* Technician Form */}
      <div className="row justify-content-center">
        <div className="col-md-8">
          <div className="card shadow-lg p-4">
            <TechnicianForm />
          </div>
        </div>
      </div>

      <hr className="my-5" />

      {/* Technician List */}
      <div className="text-center mb-4">
```

```
        <h3 className="font-weight-bold text-success">👷 Technician List</h3>
        <p className="text-muted">View and manage all available technicians
below.</p>
      </div>

      <div className="row justify-content-center">
        <div className="col-md-10">
          <TechnicianList />
        </div>
      </div>
    </div>
  );
};

export default Contact;
```

## 8) ADMIN DASHBOARD PAGE:



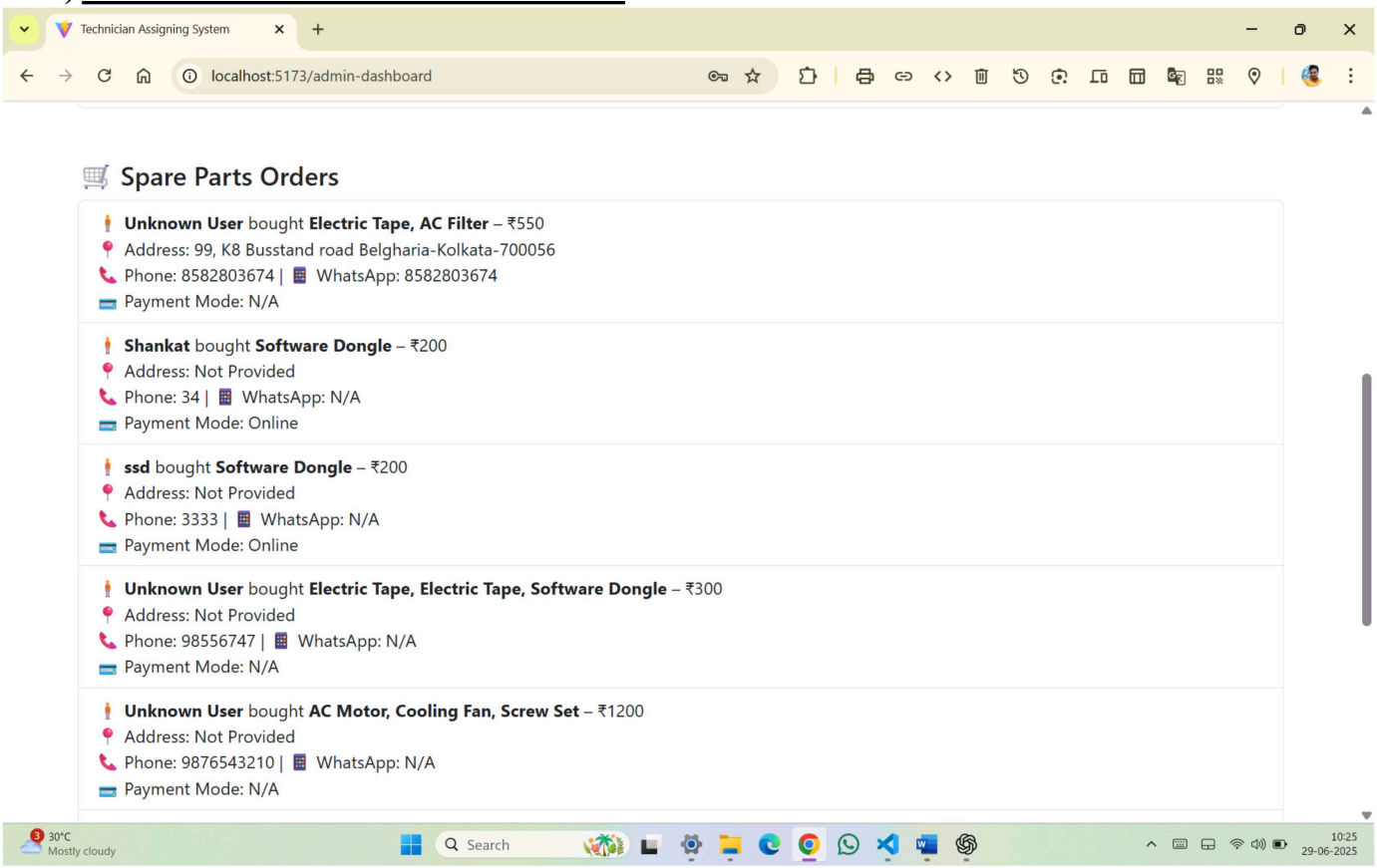

### CODE:

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

const AdminDashboard = () => {
  const [bookings, setBookings] = useState([]);
```

```
  const [orders, setOrders] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchDashboardData = async () => {
      try {
        const [bookingRes, orderRes] = await Promise.all([
          axios.get('http://localhost:6500/api/bookings'),
          axios.get('http://localhost:6500/api/orders'),
        ]);
        setBookings(bookingRes.data);
        setOrders(orderRes.data);
      } catch (error) {
        console.error('❌ Error fetching admin dashboard data:', error);
      } finally {
        setLoading(false);
      }
    };

    fetchDashboardData();
  }, []);

  if (loading) {
    return <div className="container mt-5"><h5>Loading admin
dashboard...</h5></div>;
  }

  return (
    <div className="container mt-5">
      <h2 className="mb-4">📊 Admin Dashboard</h2>

      {/* Technician Bookings Section */}
      <div className="mb-5">
        <h4>👷 Technician Bookings</h4>
        <ul className="list-group">
          {bookings.length > 0 ? (
            bookings.map((b, i) => (
              <li className="list-group-item" key={i}>
                🔧 <strong>{b.name || 'Unknown User'}</strong> booked
<strong>{b.technicianType || 'Unknown Type'}</strong> – {b.location || 'Unknown
Location'}
```

```
            </li>
          ))
        ) : (
          <li className="list-group-item text-muted">No bookings yet.</li>
        )}
      </ul>
    </div>

    {/* Spare Parts Orders Section */}
    <div>
      <h4>🛒 Spare Parts Orders</h4>
      <ul className="list-group">
        {orders.length > 0 ? (
          orders.map((o, i) => (
            <li className="list-group-item" key={i}>
              👤 <strong>{o.name || 'Unknown User'}</strong> bought
<strong>{Array.isArray(o.items) ? o.items.join(', ') : 'No items'}</strong> –
₹{o.total || 0}
              <br />
              📍 Address: {o.address || 'Not Provided'}
              <br />
              📞 Phone: {o.phone || 'N/A'} | 📱 WhatsApp: {o.wp || 'N/A'}
              <br />
              💳 Payment Mode: {o.mode || 'N/A'}
            </li>
          ))
        ) : (
          <li className="list-group-item text-muted">No spare part orders
yet.</li>
        )}
      </ul>
    </div>
  </div>
 );
};

export default AdminDashboard;
```

## 9) ADMIN SPARE PARTS ORDER PAGE:




## CODE:

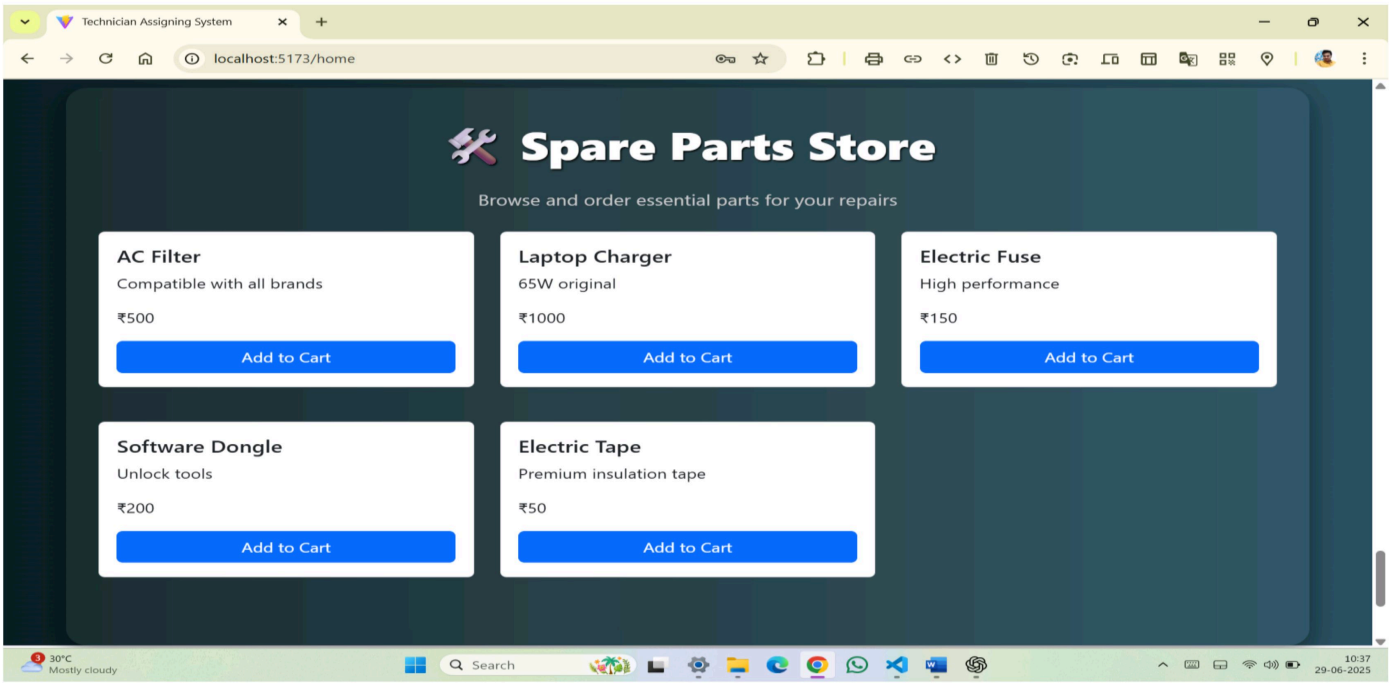
```
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const AdminDashboard = () => {
  const [bookings, setBookings] = useState([]);
  const [orders, setOrders] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchDashboardData = async () => {
      try {
        const [bookingRes, orderRes] = await Promise.all([
```

```
        axios.get('http://localhost:6500/api/bookings'),
        axios.get('http://localhost:6500/api/orders'),
      ]);
      setBookings(bookingRes.data);
      setOrders(orderRes.data);
    } catch (error) {
      console.error('❌ Error fetching admin dashboard data:', error);
    } finally {
      setLoading(false);
    }
  };

  fetchDashboardData();
}, []);

if (loading) {
  return <div className="container mt-5"><h5>Loading admin
dashboard...</h5></div>;
}

return (
  <div className="container mt-5">
    <h2 className="mb-4">📊 Admin Dashboard</h2>

    {/* Technician Bookings Section */}
    <div className="mb-5">
      <h4>👨‍🔧 Technician Bookings</h4>
      <ul className="list-group">
        {bookings.length > 0 ? (
```

```
            bookings.map((b, i) => (
              <li className="list-group-item" key={i}>
                 <strong>{b.name || 'Unknown User'}</strong> booked
<strong>{b.technicianType || 'Unknown Type'}</strong> – {b.location || 'Unknown
Location'}
              </li>
            ))
          ) : (
            <li className="list-group-item text-muted">No bookings yet.</li>
          )}
        </ul>
      </div>

      {/* Spare Parts Orders Section */}
      <div>
        <h4>🛒 Spare Parts Orders</h4>
        <ul className="list-group">
          {orders.length > 0 ? (
            orders.map((o, i) => (
              <li className="list-group-item" key={i}>
                 <strong>{o.name || 'Unknown User'}</strong> bought
<strong>{Array.isArray(o.items) ? o.items.join(', ') : 'No items'}</strong> –
₹{o.total || 0}
                <br />
                📍 Address: {o.address || 'Not Provided'}
                <br />
                📞 Phone: {o.phone || 'N/A'} | 📱 WhatsApp: {o.wp || 'N/A'}
                <br />
                💳 Payment Mode: {o.mode || 'N/A'}
              </li>
```

```
            ))
          ) : (
            <li className="list-group-item text-muted">No spare part orders
yet.</li>
          )}
        </ul>
      </div>
    </div>
  );
};

export default AdminDashboard;
```

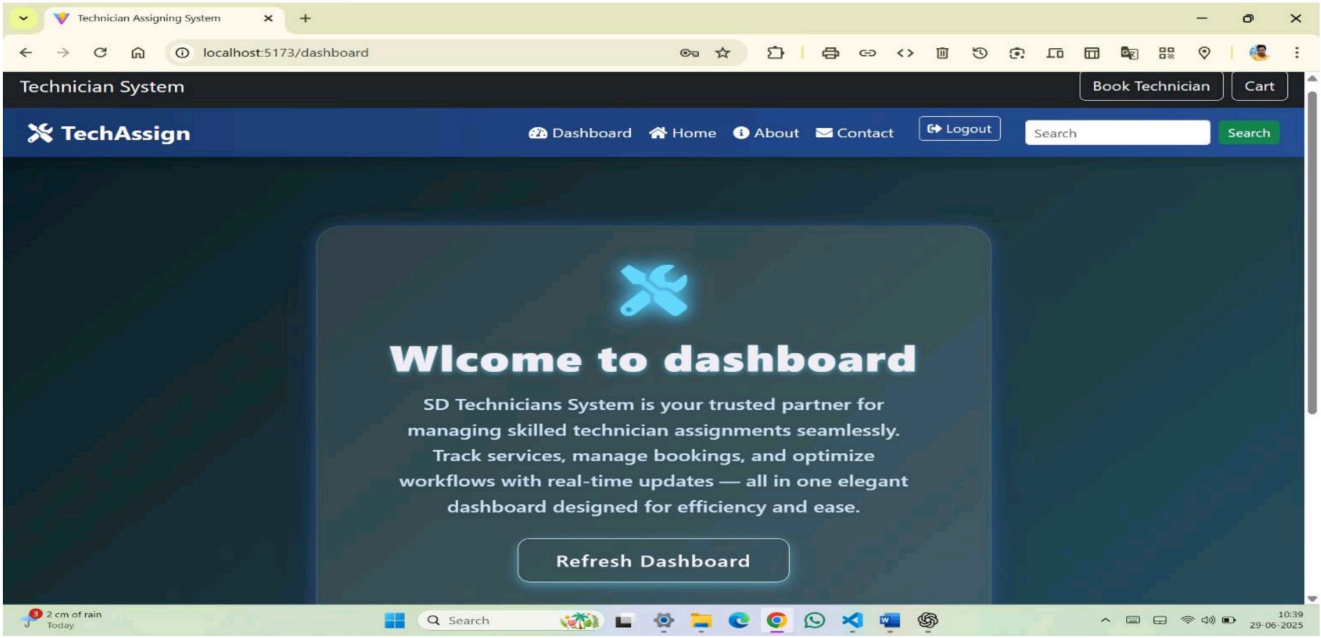
## 10) USER SPARE PARTS CARD PAGE:



**CODE**

```
import React from 'react';
import { useCart } from '../context/CartContext';

const SparePartCard = ({ part }) => {
  const { cart, setCart } = useCart();

  const handleAdd = () => {
    setCart([...cart, part]);
    alert(`${part.name} added to cart`);
  };

  return (
    <div className="card p-3 mb-3">
      <h5>{part.name}</h5>
      <p>{part.desc}</p>
      <p>₹{part.price}</p>
      <button className="btn btn-primary" onClick={handleAdd}>Add to
Cart</button>
    </div>
  );
};

export default SparePartCard;
```

**USER DASHBOARD PAGE:**




**CODE:**

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

const Dashboard = () => {
  const [msg, setMsg] = useState('');

  useEffect(() => {
    axios
      .get("http://localhost:6500/api/auth/dashboard", {
        headers: { Authorization: `Bearer ${localStorage.getItem('token')}` }
      })
      .then(res => setMsg(res.data.message))
      .catch(() => setMsg('Welcome to SD Technicians System'));
  }, []);
```

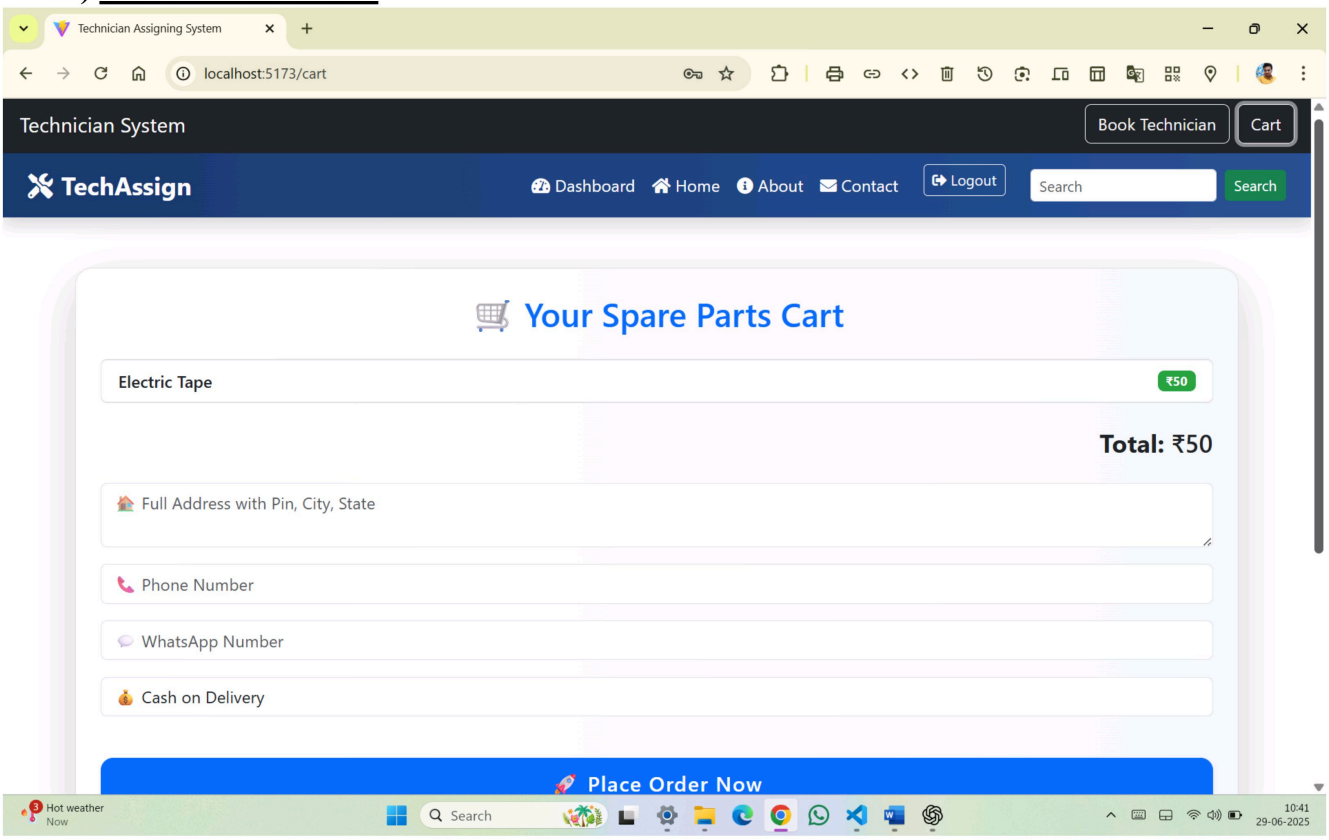
```
  return (
    <div
      className="d-flex align-items-center justify-content-center vh-100 px-3"
      style={{
        background: 'linear-gradient(135deg, #0f2027, #203a43, #2c5364)',
        fontFamily: "'Segoe UI', Tahoma, Geneva, Verdana, sans-serif",
      }}
    >
      <div
        className="glass-card text-center p-5"
        style={{
          background: 'rgba(255, 255, 255, 0.12)',
          backdropFilter: 'blur(15px)',
          WebkitBackdropFilter: 'blur(15px)',
          borderRadius: '25px',
          color: '#f0f0f3',
          maxWidth: '650px',
          width: '100%',
          boxShadow:
            '0 8px 32px 0 rgba(31, 38, 135, 0.37), 0 0 15px 2px rgba(100, 180,
255, 0.25)',
          border: '1px solid rgba(255, 255, 255, 0.18)',
          transition: 'transform 0.3s ease',
        }}
      >
        <div className="mb-4">
          <i
```

```
    className="fas fa-tools fa-4x"
    style={{
      color: '#67d9ff',
      textShadow: '0 0 8px #67d9ff',
      animation: 'pulseGlow 2.5s infinite alternate',
    }}
  ></i>
</div>

<h1
  className="mb-3"
  style={{
    fontWeight: '900',
    fontSize: '2.8rem',
    letterSpacing: '1.3px',
    textShadow: '0 2px 6px rgba(103, 217, 255, 0.6)',
  }}
>
  {msg}
</h1>

<p
  className="lead mb-4"
  style={{
    fontSize: '1.25rem',
    lineHeight: '1.6',
    color: '#cde6f7',
    maxWidth: '520px',
```

```
            margin: '0 auto',
            fontWeight: '500',
          }}
        >
          SD Technicians System is your trusted partner for managing skilled technician assignments seamlessly. Track services, manage bookings, and optimize workflows with real-time updates — all in one elegant dashboard designed for efficiency and ease.
        </p>

        <button
          onClick={() => window.location.reload()}
          className="btn btn-outline-light btn-lg"
          style={{
            borderRadius: '12px',
            padding: '12px 36px',
            fontWeight: '600',
            letterSpacing: '0.05em',
            boxShadow: '0 4px 12px rgba(103, 217, 255, 0.5)',
            transition: 'all 0.3s ease',
          }}
          onMouseEnter={(e) => {
            e.target.style.backgroundColor = '#67d9ff';
            e.target.style.color = '#02273a';
            e.target.style.boxShadow = '0 6px 20px rgba(103, 217, 255, 0.8)';
          }}
          onMouseLeave={(e) => {
            e.target.style.backgroundColor = 'transparent';
            e.target.style.color = '#f0f0f3';
```

```
            e.target.style.boxShadow = '0 4px 12px rgba(103, 217, 255, 0.5)';
          }}
        >
          Refresh Dashboard
        </button>

      <style>{`
        @keyframes pulseGlow {
          0% {
            text-shadow: 0 0 8px #67d9ff;
          }
          100% {
            text-shadow: 0 0 20px #67d9ff, 0 0 30px #3bb9ff;
          }
        }
      `}</style>
      </div>
    </div>
  );
};

export default Dashboard;
```

## 12) USER CART PAGE:




### CODE:

```
import React, { useState } from 'react';
import { useCart } from '../context/CartContext';
import axios from 'axios';

const Cart = () => {
  const { cart, setCart } = useCart();
  const [name, setName] = useState('');
  const [address, setAddress] = useState('');
  const [phone, setPhone] = useState('');
  const [wp, setWp] = useState('');
  const [mode, setMode] = useState('COD');

  const total = cart.reduce((sum, item) => sum + item.price, 0);

  const placeOrder = async () => {
```

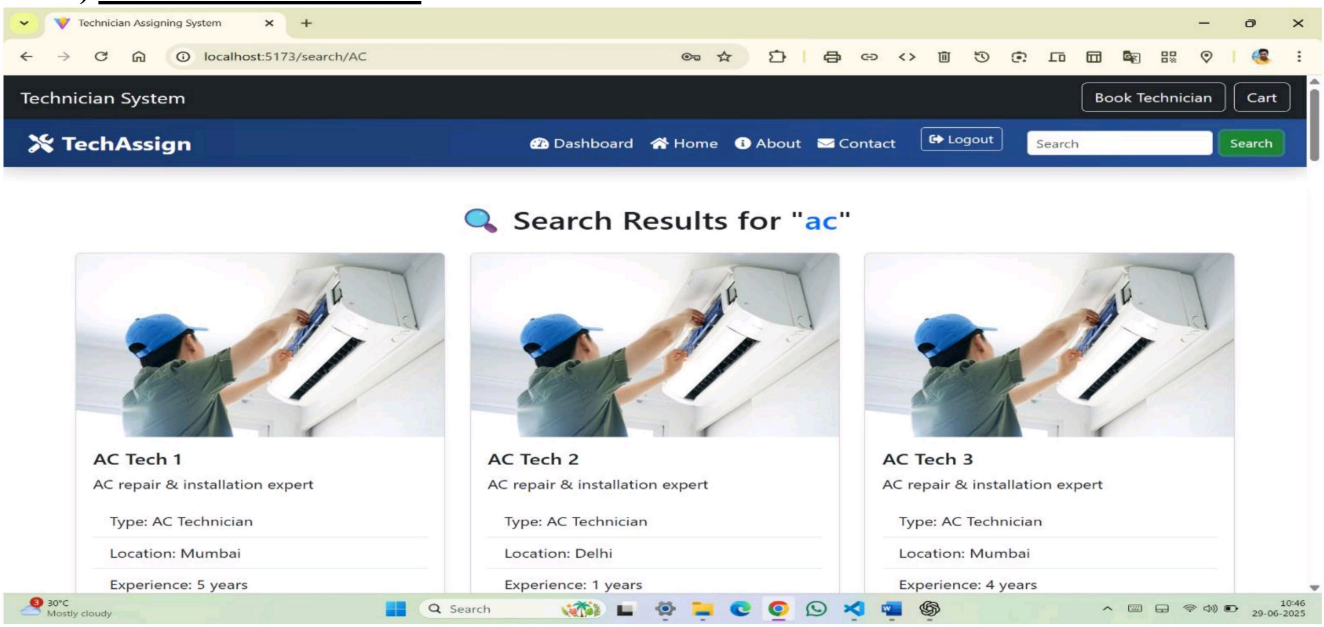
```
  const user = JSON.parse(localStorage.getItem('user'));
  if (!user) {
    alert('⚠ Please login to place an order.');
    return;
  }

  const order = {
    items: cart.map(item => item.name),
    total,
    name: user.name || name,
    email: user.email || 'unknown',
    address,
    phone,
    wp,
    mode,
  };

  try {
    await axios.post('http://localhost:6500/api/orders', order);
    alert(`✅ Order placed using ${mode}!\n📞 Owner: 9876543210`);
    setCart([]);
    setName('');
    setAddress('');
    setPhone('');
    setWp('');
  } catch (err) {
    alert('❌ Failed to place order');
    console.error(err);
  }
};

return (
```

```
    <div className="container mt-5 mb-5">
      <div
        className="card p-4 shadow-lg"
        style={{
          background: 'linear-gradient(to right, #ffffff, #f7f8fa)',
          borderRadius: '16px',
          border: '1px solid #e0e0e0',
        }}
      >
        <h2 className="text-center mb-4 text-primary">🛒 Your Spare Parts Cart</h2>

        {cart.length === 0 ? (
          <p className="text-center text-muted">🔘 Your cart is empty.</p>
        ) : (
          <>
            <ul className="list-group mb-4">
              {cart.map((item, index) => (
                <li
                  key={index}
                  className="list-group-item d-flex justify-content-between align-
items-center"
                  style={{ fontWeight: '500' }}
                >
                  {item.name}
                  <span className="badge badge-success badge-
pill">₹{item.price}</span>
                </li>
              ))}
            </ul>

            <h4 className="text-right text-dark mb-4">
              <strong>Total:</strong> ₹{total}
            </h4>
```

```
<div className="form-group">
  <textarea
    className="form-control mb-3"
    placeholder="🏠 Full Address with Pin, City, State"
    value={address}
    onChange={(e) => setAddress(e.target.value)}
    required
  />
  <input
    className="form-control mb-3"
    placeholder="📞 Phone Number"
    value={phone}
    onChange={(e) => setPhone(e.target.value)}
    required
  />
  <input
    className="form-control mb-3"
    placeholder="💬 WhatsApp Number"
    value={wp}
    onChange={(e) => setWp(e.target.value)}
    required
  />
  <select
    className="form-control mb-4"
    value={mode}
    onChange={(e) => setMode(e.target.value)}
  >
    <option value="COD">💰 Cash on Delivery</option>
    <option value="Online">💳 Online Payment</option>
  </select>
</div>
```

```
            <button
              className="btn btn-primary btn-lg btn-block shadow-sm"
              style={{
                fontWeight: '600',
                letterSpacing: '1px',
                borderRadius: '12px',
              }}
              onClick={placeOrder}
            >
              🚀 Place Order Now
            </button>
          </>
        )}
      </div>
    </div>
  );
};

export default Cart;
```

## 13) USER SEARCH PAGE:




### CODE:

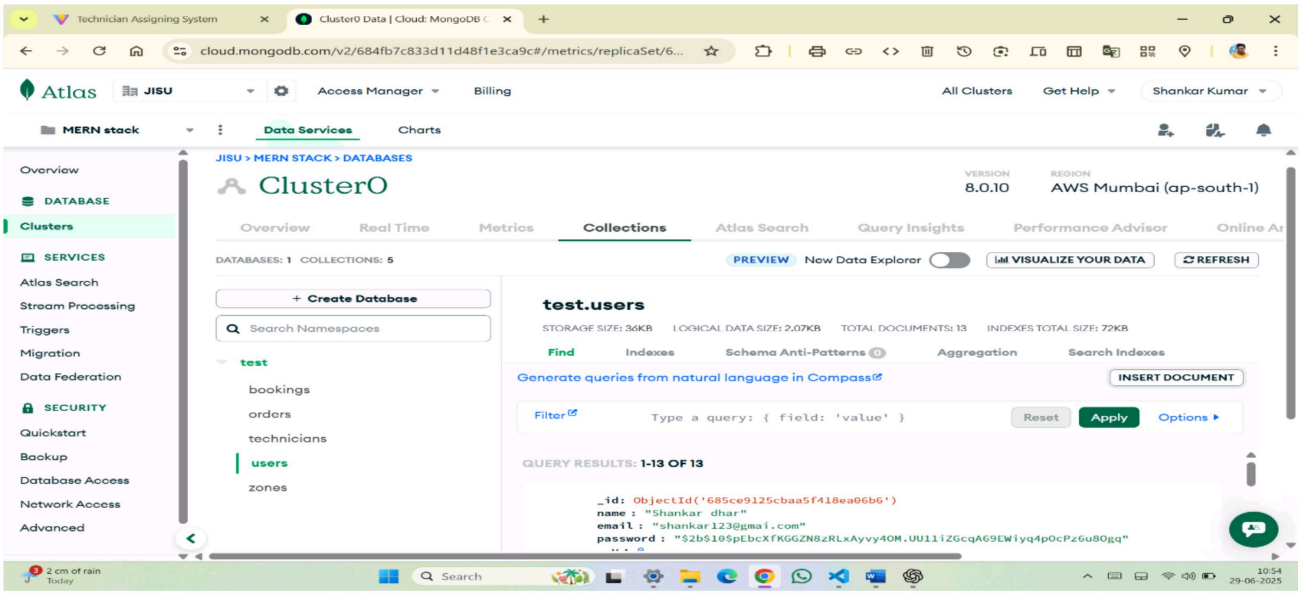
```
import React from 'react';
import { useParams, Link } from 'react-router-dom';
import citydata from '../data/citydata';

const SearchResults = () => {
  const { query } = useParams();
  const searchTerm = decodeURIComponent(query).toLowerCase();

  // 🔍 Filter technician data
  const results = citydata.filter((tech) =>
    tech.name.toLowerCase().includes(searchTerm) ||
    tech.type.toLowerCase().includes(searchTerm) ||
    tech.location.toLowerCase().includes(searchTerm)
  );

  return (
    <div className="container mt-5">
      <h2 className="text-center mb-4">
```

```
        🔍 Search Results for "<span className="text-primary">{searchTerm}</span>"
      </h2>

      {results.length === 0 ? (
        <h4 className="text-center text-danger">No technicians found.</h4>
      ) : (
        <div className="row">
          {results.map((tech) => (
            <div className="col-md-4 mb-4" key={tech.id}>
              <div className="card h-100 shadow-sm">
                <img src={tech.image} className="card-img-top" alt={tech.name} />
                <div className="card-body">
                  <h5 className="card-title">{tech.name}</h5>
                  <p className="card-text">{tech.desc}</p>
                  <ul className="list-group list-group-flush">
                    <li className="list-group-item">Type: {tech.type}</li>
                    <li className="list-group-item">Location: {tech.location}</li>
                    <li className="list-group-item">Experience: {tech.experience}
years</li>
                    <li className="list-group-item">Rating: ☆ {tech.rating}</li>
                  </ul>
                  <Link to={`/view/${encodeURIComponent(tech.type)}`} className="btn
btn-outline-primary mt-3 w-100">
                    View Technicians
                  </Link>
                </div>
              </div>
            </div>
          ))}
        </div>
      )}
    </div>
  );
```

```
};

export default SearchResults;
```

## ⍰ BACKEND:-

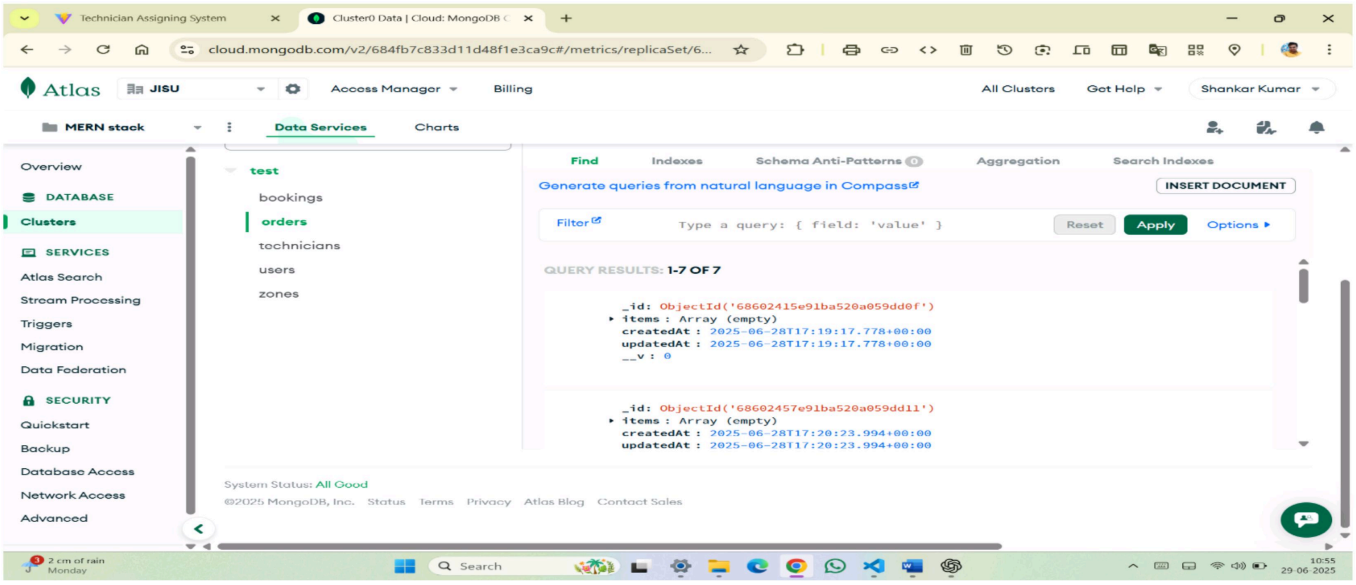
### 1) USER AND ADMIN DATA:




- **Database - db.js:**

```
const mongoose = require('mongoose');


const connectdb = async()=> {
    try {
        await mongoose.connect(process.env.MONGO_URI);
        console.log("mongodb connected");
    }
    catch(error) {
        console.error("error:",error);
    }
}

module.exports = connectdb;z
```

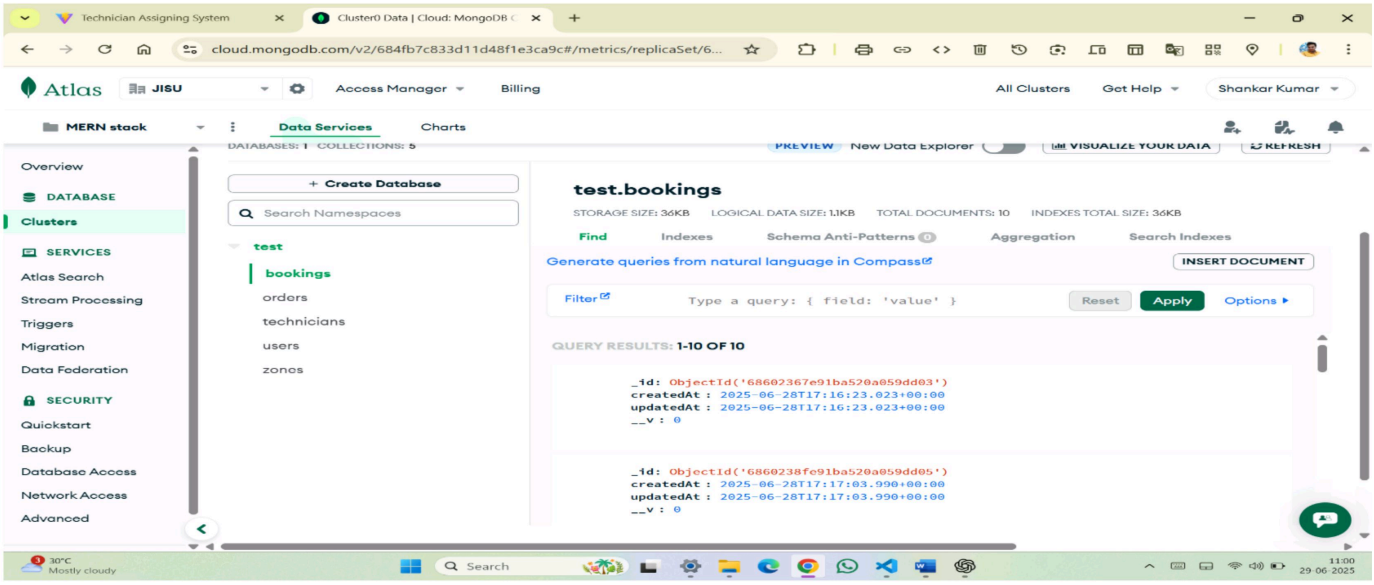
## 2) ORDER DATA:




- **models– Order.js :-**

```
const mongoose = require('mongoose');

const orderSchema = new mongoose.Schema({
  items: [String],
  total: Number,
  user: String,        // □ Name of user
  address: String,
  phone: String,
  wp: String,
  createdAt: {
    type: Date,
    default: Date.now,
  },
});

module.exports = mongoose.model('Order', orderSchema);
```

## 3) BOOKING DATA

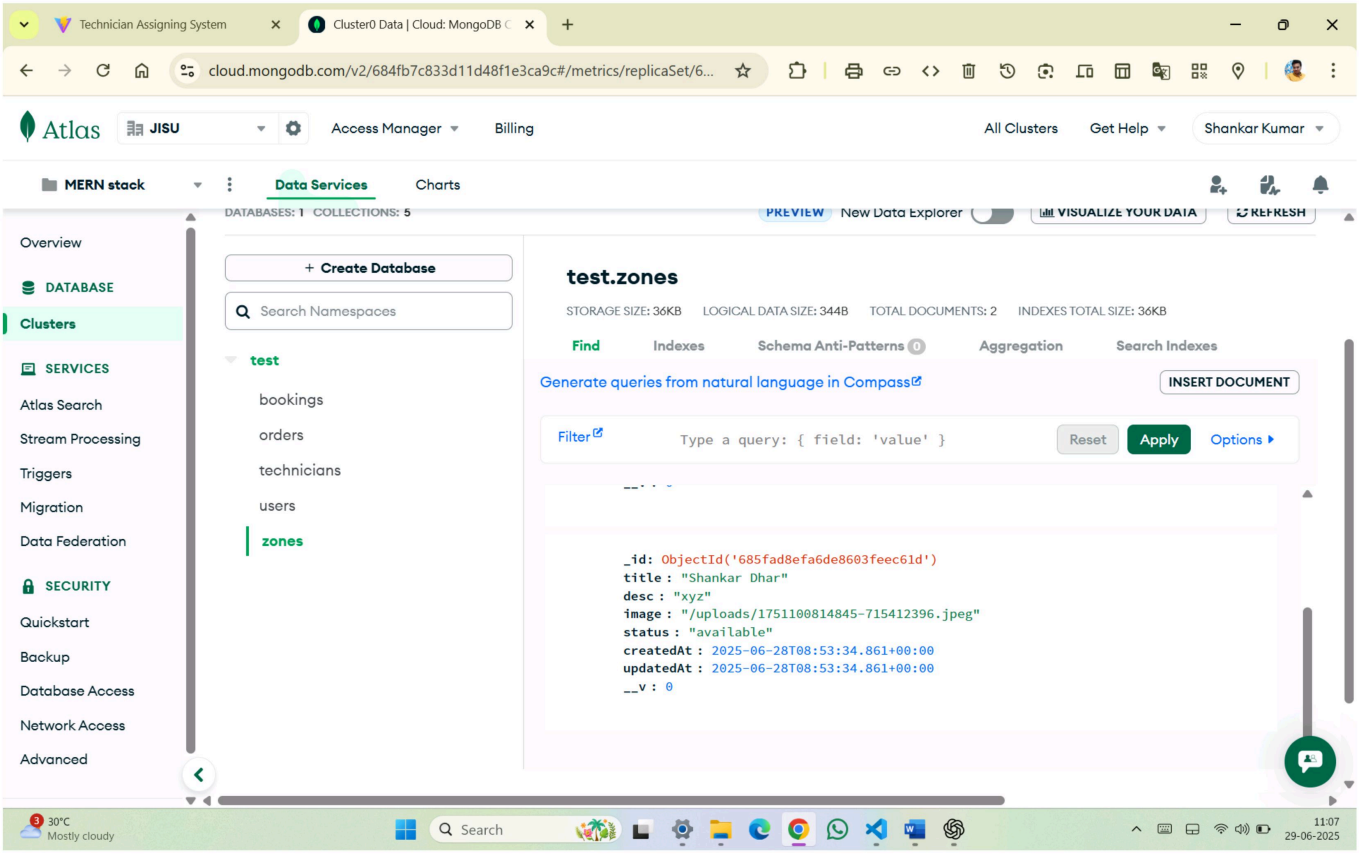



- **models – Booking.js :-**

```
const mongoose = require('mongoose');

const bookingSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  technicianType: {
    type: String,
    required: true,
  },
  location: {
    type: String,
    required: true,
  },
  createdAt: {
    type: Date,
```

```
    default: Date.now,
  },
});

module.exports = mongoose.model('Booking', bookingSchema);
```

### 4) ZONE DATA




- **models – Zone.js :-**

```
const mongoose = require('mongoose');

//create schema
const zoneschema = new mongoose.Schema({
   title:{type:String,required:true},
   desc:{type:String,required:true},
```

```
    image:{type:String},
    status:{type:String,default:'Pending'},

},{timestamps:true});
module.exports = mongoose.model('Zone',zoneschema)
```

# 6. CONCLUSION

The Technician Assignment Manager device plays a crucial position in streamlining service operations with the aid of ensuring efficient technician allocation, actual-time monitoring, and well timed resolution of customer service requests. By automating job task based on technician abilities, place, and availability, the machine reduces guide errors, delays, and mismatches. It additionally enables obvious communication among clients, technicians, and directors.

Key benefits of the machine encompass:

Improved carrier shipping and customer pleasure
Real-time status monitoring of tasks
Optimal usage of technician assets
Reduced response time and better accountability

# 7. FUTURE SCOPE & FURTHER ENHANCEMENTS

## ❖ Future scope:-

The Technician Assignment Manager device has the ability for vast destiny improvement to make it smarter, faster, and greater person-friendly. Below are some possible future scope regions and upgrades:--

**AI-Based Job Assignment:** Use Artificial Intelligence to routinely assign jobs primarily based on technician performance history, vicinity, and skill suit.

**Mobile Application Support:-** Launch mobile apps for technicians and customers to update fame, track development, and provide remarks in real-time.

**GPS & Route Optimization:-** Integrate GPS monitoring to locate technicians and use path optimization for quicker carrier delivery.

**Multi-Language Support:-** Add local language options to make the system greater handy across distinctive geographies.

**Integration with IoT Devices:-** Use IoT sensors to car-generate service requests from smart home or business devices.

## ❖ Further enhancement:-

**Real-Time Notification System:-** SMS, e mail, and push notifications for job updates, reminders, and technician **arrival signals.**

**Technician Rating & Review System:-** Let clients charge technicians, assisting improve service fine and trust.

**Analytics & Dashboards:-** Visual dashboards for admin to display key metrics like technician overall performance, common reaction time, and so on.

**Chatbot Support:-** Provide AI chat aid for clients to quickly get updates or record court cases.

**Cloud-Based Architecture:-** Move the device to the cloud for higher scalability, protection, and far flung get admission to.

# 8. BIBLIOGRAPHY


1) www.w3schools.com

2) www.youtube.com

3) www.pexels.com

4)www.codepen.io

5) www.google.com

6) www.googlefont.com

7) www.react.our