1. INTRODUCTION

" **ONLINE FURNITURE SHOP** " is web-application project which aims to provide customers with a seamless shopping experience from browsing to purchase. The website will have a responsive design, meaning it can be accessed from desktop and mobile devices. The web application will provide customers with a convenient platform to browse, select, and purchase a wide range of furniture items from the comfort of their homes. The application will be designed to cater to the needs of both residential and commercial customers, offering a diverse selection of furniture pieces to suit various styles and preferences.

1.1 Project Description :

Our Online Furniture Shop is a user-friendly MERN (MongoDB, Express.js, React.js, Node.js) full-stack project that offers customers a hassle-free browsing experience without the need for registration. Users can explore various furniture categories, add products to their cart, and create a personalized shopping list. During the checkout process, users are required to register and create an account for secure payment transactions.

Registered users gain access to additional features such as order tracking, order history, and profile management. They can update their personal information, including name, address, contact number, and email.

The project also provides comprehensive administrative capabilities. Admin users can manage furniture categories, add, modify, or delete products, and efficiently process customer orders. The system ensures an up-to-date catalog, responsive user interface, secure payment processing, and efficient inventory and order management.

1.2 Problem Statement :

The existing furniture shopping systems often require users to register before browsing products, which can be a barrier for potential customersThere may be limitations in terms of user profile management, order tracking, and administrative functionalities. Therefore, there is a need for an improved online furniture shop that provides a seamless user experience and efficient management capabilities.

1.3 Existing System :

The current online furniture shopping systems often require users to create an account before they can browse and make purchases. This can discourage potential customers who prefer to explore products without the hassle of registration. The existing systems may also lack comprehensive user profile management and administrative features, making it challenging to track orders and efficiently manage inventory and categories.

1.4 Limitation of Existing System :

The limitations of the existing system include mandatory user registration, limited user profile management options, lack of order tracking features for customers, and limited administrative capabilities for managing furniture categories, products, and orders. These limitations can hinder user engagement, decrease customer satisfaction, and make the administrative tasks more time-consuming.

1.5 Proposed System :

The proposed system is an enhanced online furniture shop that allows users to browse products without mandatory registration, improving the user experience. It incorporates comprehensive user profile management options, including the ability to update personal information such as name, address, contact number, and email. The system also includes order tracking features for customers and extensive administrative functionalities for managing furniture categories, products, and orders.

1.6 Objectives :

The objectives of the proposed system include:

- Allowing users to browse products without mandatory registration
- Providing comprehensive user profile management options
- Enabling customers to track their orders easily
- Facilitating efficient management of furniture categories, products, and orders for administrators

1.7 Benefits of Proposed System :

The proposed system offers several benefits, including:

- Enhanced user experience by allowing browsing without registration
- Improved customer satisfaction through comprehensive profile management and order tracking features
- Streamlined administrative tasks with efficient management of categories, products, and orders
- Increased user engagement and conversion rates due to a user-friendly interface
- Potential for higher sales and revenue as customers have a seamless shopping experience and access to personalized features.

2. SYSTEM REQUIREMENTS

2.1 Hardware Specifications :

- **Server:** You will need a reliable server to host your web application. The server should have sufficient processing power, memory, and storage capacity to handle the expected traffic and data storage requirements. Consider using a cloud hosting service like AWS, Google Cloud, or Azure, or a dedicated server if you prefer.
- **Database Server:** You will need a database server to store and retrieve your application's data. You can use MongoDB as the database for your MERN stack project. Ensure that the database server meets the recommended system requirements for MongoDB.
- **Network Infrastructure:** A stable and secure network infrastructure is essential to ensure smooth communication between the server and the client's devices. Make sure you have a reliable internet connection and proper network configurations in place.

2.2 Software Specifications :

- **Operating System:** Choose an operating system compatible with the MERN stack. You can use any popular operating system such as Linux (e.g., Ubuntu, CentOS), Windows, or macOS, depending on your familiarity and preferences.
- **Node.js:** Install the latest stable version of Node.js on the server. Node.js is the runtime environment for executing JavaScript on the server-side. Ensure that the Node.js version is compatible with the MERN stack framework you'll be using.
- **MongoDB:** Install and configure MongoDB on the server. MongoDB is a NoSQL database that works seamlessly with Node.js and is commonly used in MERN stack applications.
- **Web Browser:** Ensure that your web application is compatible with popular web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. Consider testing your application's compatibility with different browser versions to provide a consistent user experience.

2.3 Tools Survey :

- **Text Editors/IDEs:** Choose a text editor or integrated development environment (IDE) for writing your code. Some popular options include Visual Studio Code, Sublime Text, Atom, or WebStorm. Select the one that suits your preferences and offers useful features for web development.
- **Express.js:** Express is a popular web application framework for Node.js. It provides a robust set of features for building web APIs and handling HTTP requests and responses.
- **React.js:** React is a JavaScript library for building user interfaces. It allows you to create reusable UI components and efficiently update the UI when data changes. React is a key component of the MERN stack, responsible for rendering the frontend of your application.
- **Package Manager:** NPM (Node Package Manager) comes bundled with Node.js and is used to install and manage packages and dependencies for your MERN project. It allows you to easily integrate third-party libraries or frameworks into your application.
- **Axios:** Axios is a widely used HTTP client for making asynchronous requests from the frontend to the backend. It simplifies the process of sending and handling API requests and responses.
- **React Router:** React Router is a library for handling routing in React applications. It allows you to define routes and navigation between different pages or components within your application.
- **Bootstrap or Material-UI:** These are popular frontend frameworks that provide pre-designed UI components and CSS styles. They can help you build responsive and visually appealing user interfaces more efficiently.
- **JWT (JSON Web Tokens):** JWT is a standard for securely transmitting information between parties as a JSON object. It is commonly used for authentication and authorization purposes in web applications.

3. SYSTEM DESIGN

3.1 Modules

3.1.1 User Module:

- **View Furniture:** Users can browse through the available furniture items on the website without registering or logging in.
- **Add to Cart:** Users can add furniture items to their shopping cart for future purchase.
- **Register/Login:** To proceed with the checkout process, users need to register an account or log in if they already have one.
- **Checkout:** After logging in, users can proceed to the checkout process, enter their shipping information, and complete the payment for their selected furniture items.
- **Order Status:** Users can log in to their accounts and check the status of their previous orders.
- **Update Profile:** Users have the ability to update their personal information, such as name, address, contact number, and email address.

3.1.2 Admin Module:

- **Manage Furniture Categories:** The admin has access to the backend system to create, update, and delete furniture categories, such as "Sofas," "Tables," "Chairs," etc.
- **Manage Furniture Items:** The admin can add, update, and delete furniture items within each category, including details like name, description, price, and images.
- **Manage Orders:** The admin can view all the orders placed by users, change their status (e.g., pending, shipped, delivered), and manage inventory accordingly.

3.1.3 Public Visitors Module:

- **View Furniture:** Visitors who have not registered or logged in can browse the furniture items available on the website.
- **Cart Preview:** Visitors can add furniture items to their cart, but they won't be able to proceed with the checkout process unless they register or log in.
- **Contact Information:** Visitors can find contact information, such as email or phone number, for inquiries or assistance.

3.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data in a system. It is used to visualize and analyze the flow of information within a system and identify potential bottlenecks, inefficiencies, or security risks. A DFD typically consists of a set of symbols and shapes that represent various components in a system and the relationships between them.

DFDs are useful for a wide range of applications, including systems analysis, software design, process improvement, and project management. They are often used in the early stages of software development to define the requirements for a system and to identify potential problems before they occur.

3.2.1 Advantages of Data Flow Diagrams:

- **Clarity:** DFDs provide a clear visual representation of the flow of data in a system, making it easier to understand and communicate to others.
- **Systematic Approach:** DFDs help to identify the different components of a system and the relationships between them, making it easier to analyze and improve the system.
- **Improved Accuracy:** DFDs help to ensure that data is accurately represented in a system, reducing the risk of errors or discrepancies.
- **Early Identification of Problems:** DFDs can be used to identify potential bottlenecks, inefficiencies, or security risks in a system before they occur, allowing for early resolution.

- **Better Understanding of System Requirements:** DFDs help to define the requirements for a system and ensure that all stakeholders have a clear understanding of what the system is expected to do.

3.2.2 Disadvantages of Data Flow Diagrams:

- **Complexity:** For complex systems, creating a detailed and accurate DFD can be time consuming and require a high level of expertise.
- **Limited Flexibility:** Once a DFD has been created, it can be difficult to change or modify, making it less flexible than other design tools.
- **Limited Representation of Real-World Situations:** DFDs are a simplified representation of a system and may not accurately reflect real-world situations.
- **Limited Representation of Dynamic Systems:** DFDs are designed to represent static systems and may not accurately represent dynamic systems that change over time.
- **Limited Representation of Non-Data Aspects:** DFDs are focused on the flow of data and may not accurately represent other important aspects of a system, such as user interfaces or security requirements.

3.2.3 Components of Data Flow Diagrams:

	Process	A process shows a transformation or manipulation of data flows within the system.
	External Entity	An external entity is a source or destination of a data flow which is outside the area of study. Only those entities which originate or receive data are represented on a business process diagram.
	Data Store	A data store is a holding place for information within the system. It is represented by an open-ended narrow rectangle.
	Data Flow	A data flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow.

Fig. No. 3.1

3.2.4 Data Flow Diagrams Levels

Context Level (Zero level) DFD:

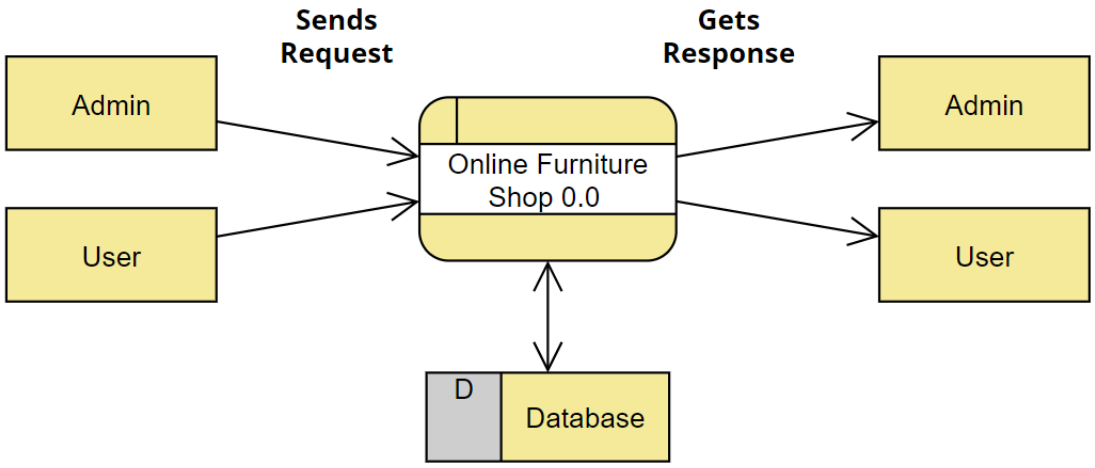

Fig. No. 3.2

First Level DFD:

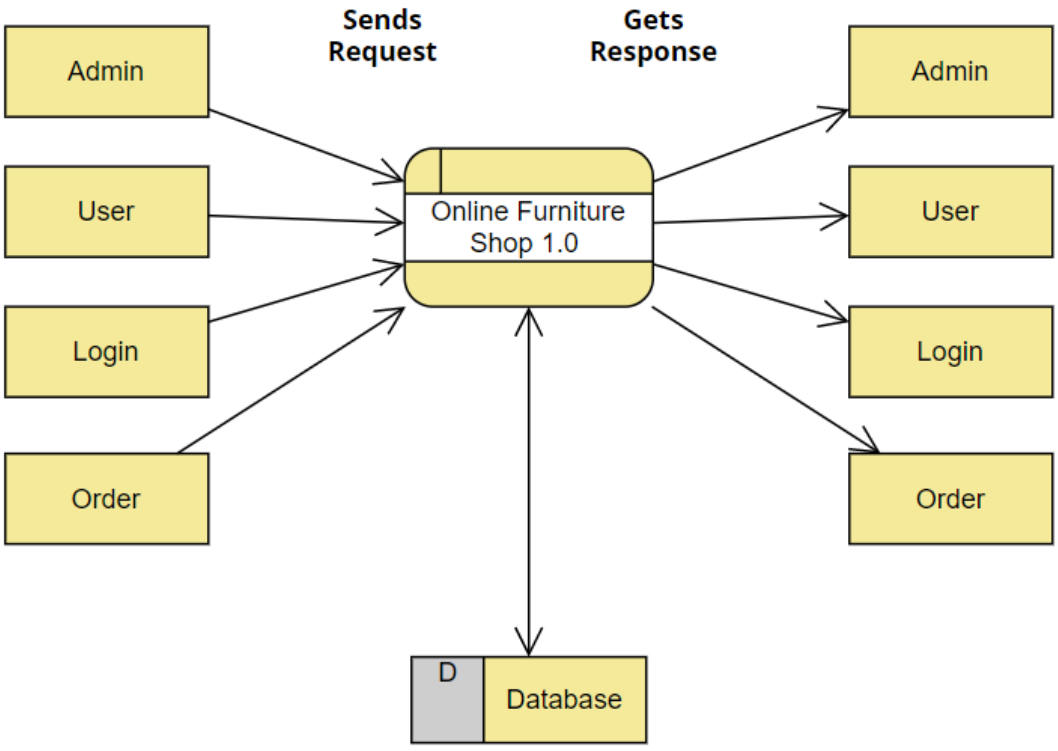

Fig. No. 3.3

Second Level DFD:

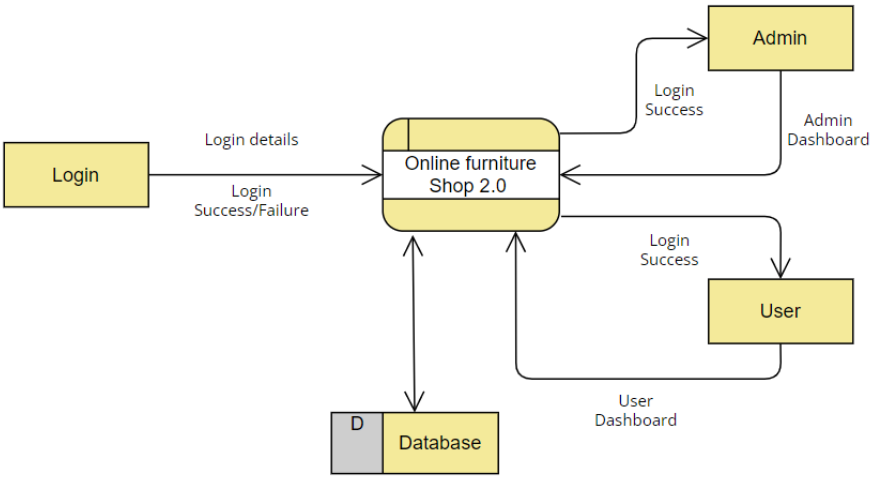

Fig. No. 3.4

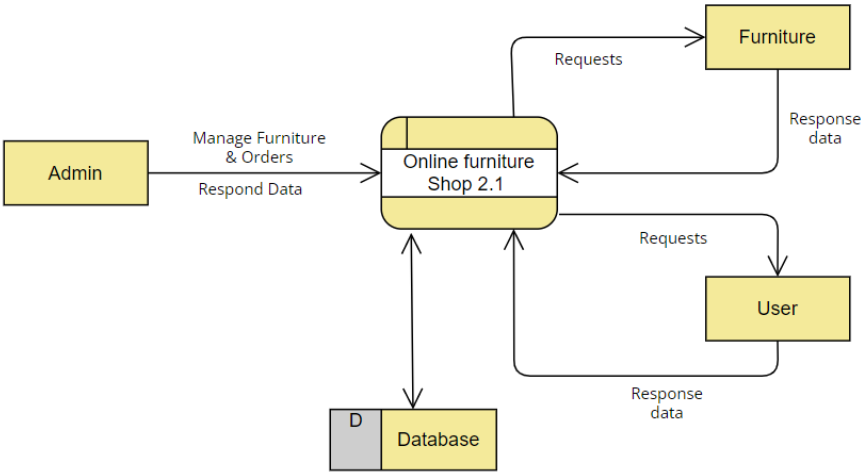

Fig. No. 3.5

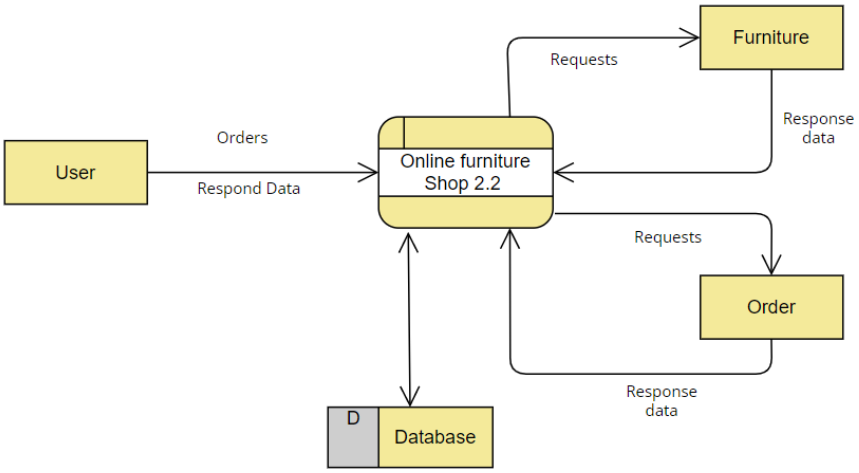

Fig. No. 3.6

3.3 ER-Diagram

An Entity Relationship Diagram (ERD) is a graphical representation of entities and their relationships to each other, used in database design. An entity can be a real-world object, such as a customer or an order, with attributes described by columns in a database table. Relationships between entities are depicted as lines connecting them, with a diamond shape representing the relationship itself and identifying the cardinality (e.g. one-to-one, one-to-many, many-to-many). ERDs help visualize the structure of data and communicate design decisions to stakeholders.

Entities: The representation of real-world objects or concepts as graphical symbols, with their properties defined as attributes.

Relationships: The connections between entities, depicted as lines connecting the entities, with a diamond shape representing the relationship itself and identifying the cardinality.

3.3.1 Advantages of Entity Relationship Diagram (ERD):

- **Clear visualization:** ERDs provide a clear and concise visual representation of entities and their relationships, making it easier for stakeholders to understand the data structure and design decisions.
- **Database design aid:** ERDs help database designers to identify entities, attributes, and relationships before creating tables, thus reducing the risk of design errors.
- **Improved data integrity:** ERDs help ensure data integrity by highlighting relationships between entities and enforcing rules, such as referential integrity, through the use of primary and foreign keys.
- **Facilitated communication:** ERDs can be used to communicate design decisions to stakeholders and facilitate discussion about the data structure.
- **Ease of maintenance:** ERDs make it easier to maintain the database as it grows and evolves, as changes to the data structure can be easily communicated and made through updating the diagram.

3.3.2 Disadvantages of Entity Relationship Diagram (ERD):

- **Complexity:** ERDs can become complex, especially for large and sophisticated databases, making it difficult to understand the relationships and design decisions.
- **Limited functionality:** ERDs only provide a high-level overview of the data structure and do not include detailed information such as data types and constraints.
- **Time-consuming:** Creating an ERD can be time-consuming, especially for large and complex databases, as it requires a thorough understanding of the data requirements and design decisions.
- **Limitations of visual representation:** ERDs are limited by the visual representation of data structures and relationships, which may not accurately reflect the complexity of the underlying data.
- **Difficulty in making changes:** Making changes to the data structure once the ERD is complete can be challenging, as it requires updates to the diagram and the corresponding database design.

3.3.3 Entity Relationship Diagram (ERD):

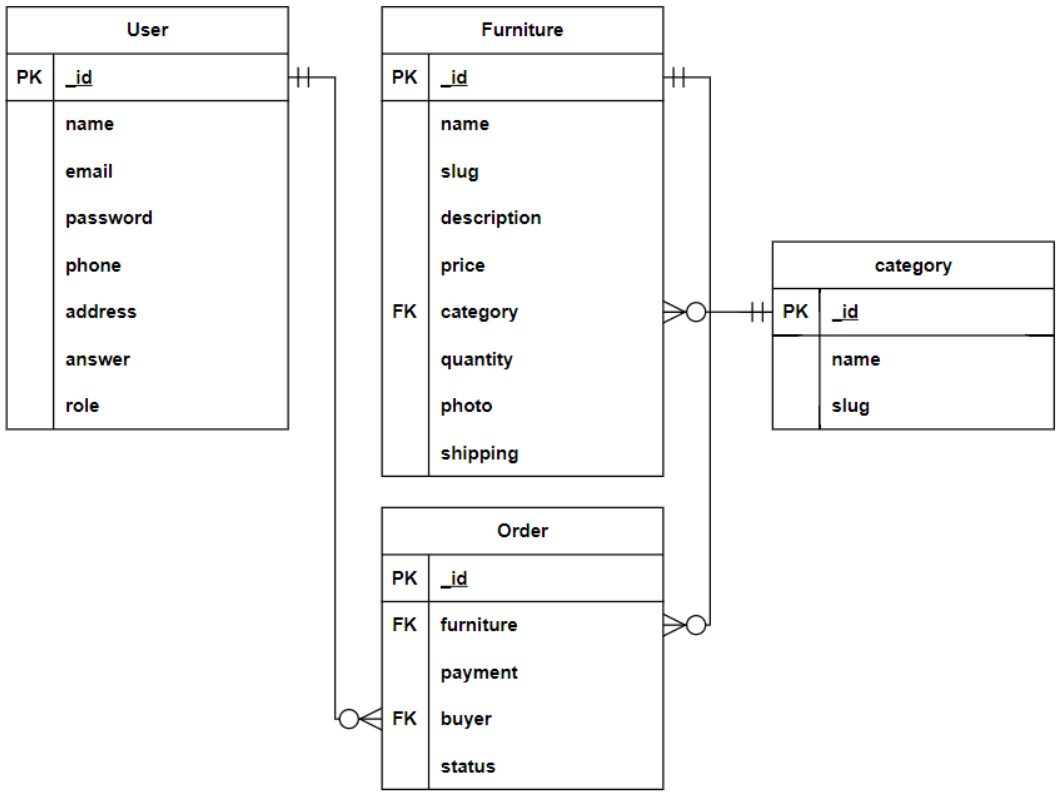

Fig. No. 3.7

3.4 Database Design

Database Name - Furnitureshop

1. **Users Collection**

	Field name	Datatype	Null
PK	_id	ObjectId	No
	name	string	No
	email	string	No
	password	string	No
	phone	string	No
	address	string	No
	answer	string	No
	role	Boolean	No

Table. No. 3.1

2. **Furnitures Collection**

	Field name	Datatype	Null
PK	_id	ObjectId	No
FK	category	ObjectId	No
	name	string	No
	slug	string	No
	description	String	No
	price	String	No
	category	String	No
	quantity	String	No
	specialization	String	No
	ratings	String	No
	photo	object	No
	shipping	Boolean	No

Table. No. 3.2

3. Orders Collection

	Field name	Datatype	Null
PK	_id	ObjectId	No
FK	furnitures	Array	No
	payment	object	No
FK	buyer	ObjectId	No
	status	string	No
	createdAt	Date	No

Table. No. 3.3

4. Categories Collection

	Field name	Datatype	Null
PK	_id	ObjectId	No
	name	string	No
	slug	string	No

Table. No. 3.4

4. CODING

4.1 Front-End

4.1.1 Client/src/components/form – Folder

❖ **CategoryForm.js**

```
import React from "react";
const CategoryForm = ({ handleSubmit, value, setValue }) => {
  return (
    <>
      <form onSubmit={handleSubmit}>
        <div className="mb-3">
          <input
            type="text"
            className="form-control"
            placeholder="Enter Your New Category"
            value={value}
            onChange={(e) => setValue(e.target.value)}
            required
          />
        </div>
        <button type="submit" className="btn btn-primary">
          Submit
        </button>
      </form>
    </>
  );
};
export default CategoryForm;
```

❖ **SearchInput.js**

```
import React from "react";
import { useSearch } from "../../context/Search";
import { useNavigate } from "react-router-dom";
```

```
import axios from "axios";
const SearchInput = () => {
  const [values, setValues] = useSearch({});
  const navigate = useNavigate();
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const { data } = await axios.get(
        `/api/v1/furniture/search/${values.keyword}`
      );
      setValues({ ...values, results: data });
      navigate("/search");
    } catch (error) {
      console.log(error);
    }
  };
  return (
    <>
      <form className="d-flex" role="search" onSubmit={handleSubmit}>
        <input
          className="form-control me-2"
          type="search"
          placeholder="Search"
          aria-label="Search"
          // value={values.keyword}
          onChange={(e) => setValues({ ...values, keyword: e.target.value })}
        />
        <button className="btn btn-primary" type="submit">
          Search
        </button>
      </form>
    </>
  );
};
```

```
export default SearchInput;
```

4.1.2 Client/src/components/layout – Folder

❖ **AdminMenu.js**

```
import React from "react";
import { NavLink } from "react-router-dom";
const AdminMenu = () => {
  return (
    <>
      <div className="list-group">
        <h4>Admin Panel</h4>
        <NavLink
          to="/dashboard/admin/create-category"
          className="list-group-item list-group-item-action"
        >
          Manage Category
        </NavLink>
        <NavLink
          to="/dashboard/admin/Add-furniture"
          className="list-group-item list-group-item-action"
        >
          Add Furniture
        </NavLink>
        <NavLink
          to="/dashboard/admin/furnitures"
          className="list-group-item list-group-item-action"
        >
          View All Furnitures
        </NavLink>
        <NavLink
          to="/dashboard/admin/orders"
          className="list-group-item list-group-item-action"
        >
```

```
          View All Orders
        </NavLink>
        {/* <NavLink
          to="/dashboard/admin/users"
          className="list-group-item list-group-item-action"
        >
          Users
        </NavLink> */}
      </div>
    </>
  );
};
export default AdminMenu;
```

❖ **Header.js**

```
import React from "react";
import { NavLink, Link } from "react-router-dom";
import logo from "./bird.svg";
import { useAuth } from "../../context/auth";
import toast from "react-hot-toast";
import SearchInput from "../Form/SearchInput";
import useCategory from "../../hooks/useCategory";
import { useCart } from "../../context/Cart";
import { Badge } from "antd";
function Header() {
  const [auth, setAuth] = useAuth();
  const [cart] = useCart();
  const categories = useCategory();
  const handlelogout = () => {
    setAuth({
      ...auth,
      user: null,
      token: "",
    });
```

```
  localStorage.removeItem("auth");
  setTimeout(() => {
    toast.success("Logout Successfull");
  }, 100);
 };
 return (
  <>
   <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
    <button
     className="navbar-toggler"
     type="button"
     data-bs-toggle="collapse"
     data-bs-target="#navbarTogglerDemo01"
     aria-controls="navbarTogglerDemo01"
     aria-expanded="false"
     aria-label="Toggle navigation"
    >
     <span className="navbar-toggler-icon" />
    </button>
    <div className="collapse navbar-collapse" id="navbarTogglerDemo01">
     <Link to="/" className="navbar-brand p-3">
      <img src={logo} alt="logo" style={{ width: "auto", height: 40 }} />{" "}
      ONLINE FURNITURE SHOP
     </Link>
     <SearchInput />
     <ul className="navbar-nav ms-auto mt-2 mt-lg-0">
      <li className="nav-item">
       <NavLink to="/" className="nav-link">
        Home
       </NavLink>
      </li>
      <li className="nav-item dropdown">
       <Link
        className="nav-link dropdown-toggle"
```

```
    to="/category"
    data-bs-toggle="dropdown"
   >
    Categories
   </Link>
   <ul className="dropdown-menu">
    <li>
     <Link className="dropdown-item" to="/categories">
      All categories
     </Link>
    </li>
    {categories?.map((c) => (
     <li key={c._id}>
      <Link className="dropdown-item" to={`/category/${c.slug}`}>
       {c.name}
      </Link>
     </li>
    ))}
   </ul>
  </li>
  {!auth.user ? (
   <>
    <li className="nav-item">
     <NavLink to="/register" className="nav-link">
      Register
     </NavLink>
    </li>
    <li className="nav-item">
     <NavLink to="/login" className="nav-link">
      Login
     </NavLink>
    </li>
   </>
  ) : (
```

```
<>
 <li className="nav-item dropdown">
  <NavLink
   className="nav-link dropdown-toggle"
   href="#"
   role="button"
   data-bs-toggle="dropdown"
   aria-expanded="false"
  >
   {auth?.user?.name}
  </NavLink>
  <ul className="dropdown-menu">
   <li>
    <NavLink
     to={`/dashboard/${
      auth?.user.role === 1 ? "admin" : "user"
     }`}
     className="dropdown-item"
    >
     Dashboard
    </NavLink>
   </li>
   <li>
    <NavLink
     to="/login"
     className="dropdown-item"
     onClick={handlelogout}
    >
     Logout
    </NavLink>
   </li>
  </ul>
 </li>
</>
```

```
          )}
          <li className="nav-item">
            <NavLink to="/cart" className="nav-link">
              🛒 cart  
              <Badge count={cart?.length} showZero></Badge>
            </NavLink>
          </li>
        </ul>
      </div>
    </nav>
  </>
 );
}
export default Header;
```

❖ **Footer.js**

```
import React from "react";
import { Link } from "react-router-dom";
function Footer() {
 return (
  <div className="footer">
   <p className="text-center mt3">
    <Link to="/about">About Us</Link>|<Link to="/contact">Contact Us</Link>|
    <Link to="/policy">Privacy & Policy</Link>|
    <Link to="/terms">Terms & Conditions</Link>
   </p>
   <h6 className="text-center">
    Copyrights &copy;
    {new Date().getFullYear()} <span className="text-danger">|</span>{" "}
    ONLINE FURNITURE SHOP <span className="text-danger">|</span> All Right
Reserved
   </h6>
  </div>
```

```
  );
}
export default Footer;
```

❖ **Layout.js**

```
import React from "react";
import Header from "./Header";
import Footer from "./Footer";
import { Helmet } from "react-helmet";
import { Toaster } from "react-hot-toast";
const Layout = ({ children, title, author, keywords, description }) => {
  return (
    <div>
      <Helmet>
        <meta charSet="utf-8" />
        <meta name="description" content={description} />
        <meta name="keywords" content={keywords} />
        <meta name="author" content={author} />
        <title>{title}</title>
      </Helmet>{" "}
      <Header />
      <main style={{ minHeight: "76vh" }}>{children}</main>
      <Toaster />
      <Footer />
    </div>
  );
};
Layout.defaultProps = {
  title: "Online Furniture Shop",
  description: "MERN Proect by CHETHAN N",
  keywords: "MERN,Furniture shop, node,react,mongodb,express",
  author: "CHETHAN N",
};
export default Layout;
```

❖ **UserMenu.js**

```
import React from "react";
import { NavLink } from "react-router-dom";
const UserMenu = () => {
  return (
    <>
      <div className="list-group">
        <h4>Dashboard</h4>
        <NavLink
          to="/dashboard/user/profile"
          className="list-group-item list-group-item-action"
        >
          Profile
        </NavLink>
        <NavLink
          to="/dashboard/user/orders"
          className="list-group-item list-group-item-action"
        >
          Orders
        </NavLink>
        {/* <NavLink
          to="/dashboard/admin/users"
          className="list-group-item list-group-item-action"
        >
          Users
        </NavLink> */}
      </div>
    </>
  );
};
export default UserMenu;
```

4.1.3 Client/src/components/routes – Folder

❖ **AdminRoute.js**

```
import { useState, useEffect } from "react";
import { useAuth } from "../../context/auth";
import { Outlet } from "react-router-dom";
import axios from "axios";
import Spinner from "../Spinner";
export default function AdminRoute() {
  const [ok, setOk] = useState(false);
  const [auth, setAuth] = useAuth();
  useEffect(() => {
    const authCheck = async () => {
      const res = await axios.get("/api/v1/auth/admin-auth");
      if (res.data.ok) {
        setOk(true);
      } else {
        setOk(false);
      }
    };
    if (auth?.token) authCheck();
  }, [auth?.token]);
  return ok ? <Outlet /> : <Spinner path="" />;
}
```

❖ **Private.js**

```
import { useState, useEffect } from "react";
import { useAuth } from "../../context/auth";
import { Outlet } from "react-router-dom";
import axios from "axios";
import Spinner from "../Spinner";
export default function PrivateRoute() {
  const [ok, setOk] = useState(false);
```

```
  const [auth, setAuth] = useAuth();
  useEffect(() => {
    const authCheck = async () => {
      const res = await axios.get("/api/v1/auth/user-auth");
      if (res.data.ok) {
        setOk(true);
      } else {
        setOk(false);
      }
    };
    if (auth?.token) authCheck();
  }, [auth?.token]);
  return ok ? <Outlet /> : <Spinner />;
}
```

❖ **Prices.js**

```
export const Prices = [
  {
    _id: 0,
    name: "Show All",
    array: [1000, 9999],
  },
  {
    _id: 1,
    name: "₹10000 to ₹19999",
    array: [10000, 19999],
  },
  {
    _id: 2,
    name: "₹20000 to ₹29999",
    array: [20000, 29999],
  },
  {
    _id: 3,
```

```
    name: "₹30000 to ₹39999",
    array: [30000, 39999],
  },
  {
    _id: 4,
    name: "₹40000 to ₹49999",
    array: [40000, 49999],
  },
  {
    _id: 5,
    name: "₹50000 and More",
    array: [50000, 89999],
  },
];
```

❖ **Spinner.js**

```
import React, { useState, useEffect } from "react";
import { useNavigate, useLocation } from "react-router-dom";
import load from "./loading.json";
import Lottie from "lottie-react";
const Spinner = ({ path = "login" }) => {
  const [count, setCount] = useState(3);
  const navigate = useNavigate();
  const location = useLocation();
  useEffect(() => {
    const interval = setInterval(() => {
      setCount((prevValue) => --prevValue);
    }, 1000);
    count === 0 &&
      navigate(`/${path}`, {
        state: location.pathname,
        path,
      });
    return () => clearInterval(interval);
```

```
  }, [count, navigate]);
  return (
    <>
      <div
        className="d-flex fles-column justify-content-center align-items-center"
        style={{ height: "100vh" }}
      >
        <div style={{ width: "20%" }}>
          <Lottie animationData={load} loop={true} />
        </div>
      </div>
    </>
  );
};
export default Spinner;
```

4.1.4 Client/src/context – Folder

❖ **auth.js**

```
import { useState, useEffect, useContext, createContext } from "react";
import axios from "axios";
const AuthContext = createContext();
const AuthProvider = ({ children }) => {
  const [auth, setAuth] = useState({
    user: null,
    token: "",
  });
  //default axios
  axios.defaults.headers.common["Authorization"] = auth?.token;
  useEffect(() => {
    const data = localStorage.getItem("auth");
    if (data) {
      const parsedata = JSON.parse(data);
      setAuth({
```

```
        ...auth,
        user: parsedata.user,
        token: parsedata.token,
      });
    }
  }, []);
  return (
    <AuthContext.Provider value={[auth, setAuth]}>
      {children}
    </AuthContext.Provider>
  );
};
//Custom hook
const useAuth = () => useContext(AuthContext);
export { useAuth, AuthProvider };
```

❖ **Cart.js**

```
import { useState, useContext, createContext, useEffect } from "react";
const CartContext = createContext();
const CartProvider = ({ children }) => {
  const [cart, setCart] = useState([]);
  useEffect(() => {
    let existingCartitems = localStorage.getItem("cart");
    if (existingCartitems) setCart(JSON.parse(existingCartitems));
  }, []);
  return (
    <CartContext.Provider value={[cart, setCart]}>
      {children}
    </CartContext.Provider>
  );
};
//Custom hook
const useCart = () => useContext(CartContext);
export { useCart, CartProvider };
```

❖ **Search.js**

```
import { useState, useContext, createContext } from "react";
const SearchContext = createContext();
const SearchProvider = ({ children }) => {
  const [auth, setAuth] = useState({
    keyword: null,
    results: [],
  });
  return (
    <SearchContext.Provider value={[auth, setAuth]}>
      {children}
    </SearchContext.Provider>
  );
};
//Custom hook
const useSearch = () => useContext(SearchContext);
export { useSearch, SearchProvider };
```

4.1.5 Client/src/hooks – Folder

❖ **useCategory.js**

```
import { useState, useEffect } from "react";
import axios from "axios";
export default function useCategory() {
  const [categories, setCategories] = useState([]);
  //Get Categories
  const getCategories = async () => {
    try {
      const { data } = await axios.get(`/api/v1/category/get-category`);
      setCategories(data?.category);
    } catch (error) {
      console.log(error);
    }
```

```
  };
  useEffect(() => {
    getCategories();
  }, []);
  return categories;
}
```

4.1.6 Client/src/pages/admin – Folder

❖ **AddFurniture.js**

```
import React, { useState, useEffect } from "react";
import Layout from "../../components/Loyout/Layout";
import AdminMenu from "../../components/Loyout/AdminMenu";
import toast from "react-hot-toast";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import { Select } from "antd";
const { Option } = Select;
const AddFurniture = () => {
  const navigate = useNavigate();
  const [categories, setCategories] = useState([]);
  const [category, setCategory] = useState([]);
  const [photo, setPhoto] = useState("");
  const [name, setName] = useState("");
  const [description, setDescription] = useState("");
  const [price, setPrice] = useState("");
  const [quantity, setQuantity] = useState("");
  const [ratings, setRatings] = useState("");
  const [shipping, setShipping] = useState("");
  //handleCreate
  const handleCreate = async (e) => {
    e.preventDefault();
    try {
      const furnitureData = new FormData();
```

```
    furnitureData.append("name", name);
    furnitureData.append("photo", photo);
    furnitureData.append("description", description);
    furnitureData.append("price", price);
    furnitureData.append("quantity", quantity);
    furnitureData.append("ratings", ratings);
    furnitureData.append("shipping", shipping);
    furnitureData.append("category", category);
    const { data } = await axios.post(
      "/api/v1/furniture/create-furniture",
      furnitureData
    );
    if (data?.success) {
      setTimeout(() => {
        toast.success("Furniture Added Successfully");
      }, 100);
      navigate("/dashboard/admin/furnitures");
    } else {
      toast.error(data?.message);
    }
  } catch (error) {
    console.log(error);
    toast.error("Something went wrong");
  }
};
//Get All categories
const getAllcategory = async () => {
  try {
    const { data } = await axios.get("/api/v1/category/get-category");
    if (data?.success) {
      setCategories(data?.category);
    }
  } catch (error) {
    console.log(error);
```

```
    toast.error("Something went wrong while getting categories");
  }
};
useEffect(() => {
  getAllcategory();
}, []);
return (
  <Layout title={"Add Furniture"}>
    <div className="container-fluid m-3 p-3 dashboard">
      <div className="row">
        <div className="col-md-3">
          <AdminMenu />
        </div>
        <div className="col-md-9">
          <h3>Add Furniture</h3>
          <div className="m-1 w-75">
            <Select
              bordered={false}
              placeholder="Select the category"
              size="large"
              showSearch
              className="form-select mb-3"
              onChange={(value) => {
                setCategory(value);
              }}
            >
              {categories?.map((c) => (
                <Option key={c._id} value={c._id}>
                  {c.name}
                </Option>
              ))}
            </Select>
            <div className="mb-3">
              <label className="btn btn-outline-secondary col-md-12">
```

```
    {photo ? photo.name : "Upload Photo"}
    <input
     type="file"
     name="photo"
     accept="image/*"
     onChange={(e) => setPhoto(e.target.files[0])}
     hidden
     required
    />
  </label>
</div>
<div className="mb-3">
  {photo && (
    <div className="text-center">
      <img
        src={URL.createObjectURL(photo)}
        alt="Furniture-photo"
        height={"200px"}
        className="img img-responsive"
      />
    </div>
  )}
</div>
<div className="mb-3">
  <input
    required
    type="text"
    value={name}
    placeholder="Enter Furniture Name"
    className="form-control"
    onChange={(e) => setName(e.target.value)}
  />
</div>
<div className="mb-3">
```

```
  <textarea
    required
    type="text"
    value={description}
    placeholder="write a description"
    className="form-control"
    onChange={(e) => setDescription(e.target.value)}
  />
</div>
<div className="mb-3">
  <input
    required
    type="number"
    value={price}
    placeholder="write a Price"
    className="form-control"
    onChange={(e) => setPrice(e.target.value)}
  />
</div>
<div className="mb-3">
  <input
    required
    type="number"
    value={quantity}
    placeholder="write a quantity"
    className="form-control"
    onChange={(e) => setQuantity(e.target.value)}
  />
</div>{" "}
<div className="mb-3">
  <input
    required
    type="number"
    value={ratings}
```

```
            placeholder="Give an ratings"
            className="form-control"
            onChange={(e) => setRatings(e.target.value)}
           />
          </div>
          <div className="mb-3">
           <Select
            required
            bordered={false}
            placeholder="Select Shipping "
            size="large"
            showSearch
            className="form-select mb-3"
            onChange={(value) => {
             setShipping(value);
            }}
           >
            <Option value="0">No</Option>
            <Option value="1">Yes</Option>
           </Select>
          </div>
          <div className="mb-3">
           <button className="btn btn-primary" onClick={handleCreate}>
            Add Furniture
           </button>
          </div>
         </div>
        </div>
       </div>
      </div>
    </Layout>
  );
};
export default AddFurniture;
```

❖ **AdminDashboard.js**

```
import React from "react";
import Layout from "../../components/Loyout/Layout";
import AdminMenu from "../../components/Loyout/AdminMenu";
import { useAuth } from "../../context/auth";
const AdminDashboard = () => {
  const [auth] = useAuth();
  return (
    <Layout title={"Admin Dashboard"}>
      <div className="container-fluid m-3 p-3">
        <div className="row">
          <div className="col-md-3">
            <AdminMenu />
          </div>
          <div className="col-md-9">
            <div className="card w-75 p-3">
              <h3>Admin Name: {auth?.user?.name}</h3>
              <h3>Admin Email: {auth?.user?.email}</h3>
              <h3>Admin Number: {auth?.user?.phone}</h3>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
};
export default AdminDashboard;
```

❖ **AdminOrders.js**

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import toast from "react-hot-toast";
import AdminMenu from "../../components/Loyout/AdminMenu";
```

```
import Layout from "../../components/Loyout/Layout";
import { useAuth } from "../../context/auth";
import moment from "moment";
import { Select } from "antd";
const { Option } = Select;
const AdminOrders = () => {
  const [status, setStatus] = useState([
    "Not Process",
    "Processing",
    "Shipped",
    "deliverd",
    "cancel",
  ]);
  const [orders, setOrders] = useState([]);
  const [auth, setAuth] = useAuth();
  const getOrders = async () => {
    try {
      const { data } = await axios.get("/api/v1/auth/all-orders");
      setOrders(data);
    } catch (error) {
      console.log(error);
    }
  };
  useEffect(() => {
    if (auth?.token) getOrders();
  }, [auth?.token]);
  const handleChange = async (orderId, value) => {
    try {
      const { data } = await axios.put(`/api/v1/auth/order-status/${orderId}`, {
        status: value,
      });
      getOrders();
      toast.success("Order Status Updated");
    } catch (error) {
```

```
    console.log(error);
  }
};
return (
  <Layout title={"All Orders Data"}>
    <div className="container-fluid m-3 p-3">
      <div className="row dashboard">
        <div className="col-md-3">
          <AdminMenu />
        </div>
        <div className="col-md-9">
          <h1 className="text-center">All Orders</h1>
          {orders?.map((o, i) => {
            return (
              <div className="border shadow" key={o._id}>
                <table className="table">
                  <thead>
                    <tr>
                      <th scope="col">#</th>
                      <th scope="col">Status</th>
                      <th scope="col">Buyer</th>
                      <th scope="col">Order's Date</th>
                      <th scope="col">Payment</th>
                      <th scope="col">Quantity</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr>
                      <td>{i + 1}</td>
                      <td>
                        <Select
                          bordered={false}
                          onChange={(value) => handleChange(o._id, value)}
                          defaultValue={o?.status}
```

```
          >
            {status.map((s, i) => (
              <Option key={i} value={s}>
                {s}
              </Option>
            ))}
          </Select>
        </td>
        <td>{o?.buyer?.name}</td>
        <td>{moment(o?.createdAt).fromNow()}</td>
        <td>{o?.payment.success ? "Success" : "Failed"}</td>
        <td>{o?.furnitures?.length}</td>
      </tr>
    </tbody>
  </table>
  <div className="container">
    {o?.furnitures?.map((f) => (
      <div className="row mb-2 p-3 card flex-row" key={f._id}>
        <div className="col-md-4">
          <img
            src={`/api/v1/furniture/furniture-photo/${f._id}`}
            className="card-img-top"
            alt={f.name}
            //   width="100px"
            //   height={"100px"}
          />
        </div>
        <div className="col-md-8">
          <p>{f.name}</p>
          <p>{f.description.substring(0, 30)}</p>
          <p>Price : {f.price}</p>
        </div>
      </div>
    ))}
```

```
            </div>
          </div>
        );
      })}
    </div>
  </div>
 </div>
 </Layout>
 );
};
export default AdminOrders;
```

❖ **Furnitures.js**

```
import React, { useState, useEffect } from "react";
import AdminMenu from "../../components/Loyout/AdminMenu";
import Layout from "../../components/Loyout/Layout";
import axios from "axios";
import toast from "react-hot-toast";
import { Link } from "react-router-dom";
const Furnitures = () => {
 const [furnitures, setFurniture] = useState([]);
 //Get all furnitures
 const getAllFurnitures = async () => {
  try {
   const { data } = await axios.get("/api/v1/furniture/get-furniture");
   setFurniture(data.furnitures);
  } catch (error) {
   console.log(error);
   toast.error("Something Went wrong");
  }
 };
 //lifecycle method
 useEffect(() => {
  getAllFurnitures();
```

```
}, []);
return (
  <Layout>
    <div className="container-fluid m-3 p-3">
      <div className="row">
        <div className="col-md-3">
          <AdminMenu />
        </div>
        <div className="col-md-9">
          <h1 className="text-center">All Furnitures</h1>
          <div className="d-flex flex-wrap">
            {furnitures?.map((f) => (
              <Link
                to={`/dashboard/admin/update-furniture/${f.slug}`}
                className="furniture"
              >
                <div
                  className="card m-2"
                  style={{ width: "18rem" }}
                  key={f._id}
                >
                  <img
                    width={"auto"}
                    height={300}
                    src={`/api/v1/furniture/furniture-photo/${f._id}`}
                    className="card-img-top"
                    alt={f.name}
                  />
                  <div className="card-body">
                    <h5 className="card-title">{f.name}</h5>
                    <p className="card-text">{f.description}</p>
                    <p className="card-text">₹ {f.price}</p>
                    <h6
                      className={
```

```
                f.quantity > 0
                  ? `text-success text-end`
                  : `text-danger text-end`
                }
              >
                {f.quantity > 0
                  ? `${f.quantity} Stocks Available`
                  : `Out of Stock`}
              </h6>
            </div>
          </div>
        </Link>
      ))}
    </div>
  </div>
 </div>
</Layout>
);
};
export default Furnitures;
```

4.1.7 Client/src/pages/auth – Folder

❖ **ForgetPassword.js**

```
import React, { useState } from "react";
import Layout from "../../components/Loyout/Layout";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import toast from "react-hot-toast";
import "../../styles/AuthStyles.css";
import { useAuth } from "../../context/auth";
const ForgetPassword = () => {
  const [auth, setAuth] = useAuth();
```

```
const [email, setEmail] = useState("");
const [newPassword, setnewPassword] = useState("");
const [answer, setAnswer] = useState("");
const [phone, setPhone] = useState("");
const navigate = useNavigate();
// form function
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const res = await axios.post("/api/v1/auth/forget-password", {
      email,
      newPassword,
      answer,
      phone,
    });
    if (res && res.data.success) {
      setAuth({ ...auth, user: res.data.user, token: res.data.token });
      localStorage.setItem("auth", JSON.stringify(res.data));
      setTimeout(() => {
        toast.success(res.data && res.data.message);
      }, 100);
      navigate("/login");
    } else {
      setTimeout(() => {
        toast.error(res.data.message);
      }, 100);
    }
  } catch (error) {
    console.log(error);
    toast.error("Something went wrong");
  }
};
return (
  <Layout title={"Login Form"}>
```

```
<div className="form-container" style={{ minHeight: "76vh" }}>
  <form onSubmit={handleSubmit}>
    <h4 className="title">Forget Password</h4>
    <div className="mb-3">
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        className="form-control"
        id="email"
        placeholder="Enter Your Email "
        required
      />
    </div>
    <div className="mb-3">
      <input
        type="text"
        value={phone}
        onChange={(e) => setPhone(e.target.value)}
        className="form-control"
        id="phone"
        placeholder="Enter Your Correct Phone No."
        required
        autoFocus
      />
    </div>
    <div className="mb-3">
      <input
        type="text"
        value={answer}
        onChange={(e) => setAnswer(e.target.value)}
        className="form-control"
        id="answer"
        placeholder="What is Your Favorite sports"
```

```
            required
            autoFocus
          />
        </div>
        <div className="mb-3">
          <input
            type="password"
            value={newPassword}
            onChange={(e) => setnewPassword(e.target.value)}
            className="form-control"
            id="newpassword"
            placeholder="Enter Your New Password"
            required
          />
        </div>
        <button type="submit" className="btn btn-primary">
          Reset
        </button>
      </form>
    </div>
  </Layout>
 );
};
export default ForgetPassword;
```

❖ **Login.js**

```
import React, { useState } from "react";
import Layout from "../../components/Loyout/Layout";
import axios from "axios";
import { useNavigate, useLocation, Link } from "react-router-dom";
import toast from "react-hot-toast";
import "../../styles/AuthStyles.css";
import { useAuth } from "../../context/auth";
const Login = () => {
```

```
const [auth, setAuth] = useAuth();
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");

const navigate = useNavigate();
const location = useLocation();
// form function
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const res = await axios.post("/api/v1/auth/login", {
      email,
      password,
    });
    if (res && res.data.success) {
      setAuth({ ...auth, user: res.data.user, token: res.data.token });
      localStorage.setItem("auth", JSON.stringify(res.data));
      setTimeout(() => {
        toast.success(res.data && res.data.message);
      }, 100);
      const dashboard = JSON.parse(localStorage.getItem("auth"));
      navigate(
        location.state ||
          `/dashboard/${dashboard?.user.role === 1 ? "admin" : "user"}`
      );
    } else {
      setTimeout(() => {
        toast.error(res.data.message);
      }, 100);
    }
  } catch (error) {
    console.log(error);
    toast.error("Something went wrong");
  }
```

```
};
return (
  <Layout title={"Login Form"}>
    <div className="form-container" style={{ minHeight: "76vh" }}>
      <form onSubmit={handleSubmit}>
        <h4 className="title">LOGIN FORM</h4>
        <div className="mb-3">
          <input
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            className="form-control"
            id="email"
            placeholder="Enter Your Email "
            required
          />
        </div>
        <div className="mb-3">
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            className="form-control"
            id="password"
            placeholder="Enter Your Password"
            required
          />
        </div>
        <span style={{ fontSize: "15px", textAlign: "right" }}>
          Forget Password ? <Link to={"/forget-password"}>Click here</Link>
        </span>
        <br />
        <span style={{ fontSize: "15px", textAlign: "right" }}>
          Create new account ? <Link to={"/register"}>Register</Link>
```

```
        </span>
        <br />
        <br />
        <button type="submit" className="btn btn-primary">
          LOGIN
        </button>
       </form>
     </div>{" "}
   </Layout>
  );
};
export default Login;
```

❖ **Register.js**

```
import React, { useState } from "react";
import Layout from "../../components/Loyout/Layout";
import axios from "axios";
import { useNavigate, Link } from "react-router-dom";
import toast from "react-hot-toast";
import "../../styles/AuthStyles.css";
const Register = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [phone, setPhone] = useState("");
  const [address, setAddress] = useState("");
  const [answer, setAnswer] = useState("");
  const navigate = useNavigate();
  // form function
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await axios.post("/api/v1/auth/register", {
        name,
```

```
      email,
      password,
      phone,
      address,
      answer,
    });
    if (res && res.data.success) {
      setTimeout(() => {
        toast.success(res.data && res.data.message);
      }, 100);
      navigate("/login");
    } else {
      setTimeout(() => {
        toast.error(res.data.message);
      }, 100);
    }
  } catch (error) {
    console.log(error);
    toast.error("Something went wrong");
  }
};
return (
  <Layout title={"Register Form"}>
    <div className="form-container" style={{ minHeight: "76vh" }}>
      <form onSubmit={handleSubmit}>
        <h4 className="title">REGISTER FORM</h4>
        <div className="mb-3">
          <input
            type="text"
            value={name}
            onChange={(e) => setName(e.target.value)}
            className="form-control"
            id="name"
            placeholder="Enter Your Name"
```

```
            required
            autoFocus
          />
        </div>
        <div className="mb-3">
          <input
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            className="form-control"
            id="email"
            placeholder="Enter Your Email "
            required
          />
        </div>
        <div className="mb-3">
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            className="form-control"
            id="password"
            placeholder="Enter Your Password"
            required
          />
        </div>
        <div className="mb-3">
          <input
            type="text"
            value={phone}
            onChange={(e) => setPhone(e.target.value)}
            className="form-control"
            id="phone"
            placeholder="Enter Your Phone"
```

```
          required
        />
      </div>
      <div className="mb-3">
        <input
          type="text"
          value={address}
          onChange={(e) => setAddress(e.target.value)}
          className="form-control"
          id="address"
          placeholder="Enter Your Address"
          required
        />
      </div>
      <div className="mb-3">
        <input
          type="text"
          value={answer}
          onChange={(e) => setAnswer(e.target.value)}
          className="form-control"
          id="answer"
          placeholder="What is Your Favorite sports"
          required
        />
      </div>
      <span style={{ fontSize: "15px", textAlign: "right" }}>
        Already have Account ? <Link to={"/login"}>Login</Link>
      </span>
      <br />
      <br />
      <button type="submit" className="btn btn-primary">
        REGISTER
      </button>
    </form>
```

```
    </div>
   </Layout>
  );
};
export default Register;
```

4.1.8 Client/src/pages/user – Folder

❖ **Dashboard.js**

```
import React from "react";
import Layout from "../../components/Loyout/Layout";
import UserMenu from "../../components/Loyout/UserMenu";
import { useAuth } from "../../context/auth";
const Dashboard = () => {
  const [auth] = useAuth();
  return (
    <Layout title={"User Dashboard"}>
      <div className="container-fluid m-3 p-3">
        <div className="row">
          <div className="col-md-3">
            <UserMenu />
          </div>
          <div className="col-md-9">
            <div className="card w-75 p-3">
              <h3>User Name: {auth?.user?.name}</h3>
              <h3>User Email: {auth?.user?.email}</h3>
              <h3>User Number: {auth?.user?.phone}</h3>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
};
```

```
export default Dashboard;
```

❖ **Orders.js**

```
import React, { useState, useEffect } from "react";
import UserMenu from "../../components/Loyout/UserMenu";
import Layout from "./../../components/Loyout/Layout";
import axios from "axios";
import { useAuth } from "../../context/auth";
import moment from "moment";
const Orders = () => {
  const [orders, setOrders] = useState([]);
  const [auth, setAuth] = useAuth();
  const getOrders = async () => {
    try {
      const { data } = await axios.get("/api/v1/auth/orders");
      setOrders(data);
    } catch (error) {
      console.log(error);
    }
  };
  useEffect(() => {
    if (auth?.token) getOrders();
  }, [auth?.token]);
  return (
    <Layout title={"Your Orders"}>
      <div className="container-flui p-3 m-3 dashboard">
        <div className="row">
          <div className="col-md-3">
            <UserMenu />
          </div>
          <div className="col-md-9">
            <h1 className="text-center">All Orders</h1>
            {orders?.map((o, i) => {
              return (
```

```
<div className="border shadow">
  <table className="table">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Status</th>
        <th scope="col">Buyer</th>
        <th scope="col">Date</th>
        <th scope="col">Payment</th>
        <th scope="col">Quantity</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>{i + 1}</td>
        <td>{o?.status}</td>
        <td>{o?.buyer?.name}</td>
        <td>{moment(o?.createdAt).fromNow()}</td>
        <td>{o?.payment.success ? "Success" : "Failed"}</td>
        <td>{o?.furnitures?.length}</td>
      </tr>
    </tbody>
  </table>
  <div className="container">
    {o?.furnitures?.map((f) => (
      <div className="row mb-2 p-3 card flex-row" key={f._id}>
        <div className="col-md-4">
          <img
            src={`/api/v1/furniture/furniture-photo/${f._id}`}
            className="card-img-top"
            alt={f.name}
            // width="100px"
            // height={"100px"}
          />
```

```
                </div>
                <div className="col-md-8">
                  <p>{f.name}</p>
                  <p>{f.description.substring(0, 30)}</p>
                  <p>Price : {f.price}</p>
                </div>
              </div>
            ))}
          </div>
        </div>
      );
    })}
  </div>
 </div>
</div>
</Layout>
);
};
export default Orders;
```

❖ **Profile.js**

```
import Layout from "../../components/Loyout/Layout";
import UserMenu from "../../components/Loyout/UserMenu";
import React, { useEffect, useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import toast from "react-hot-toast";
import { useAuth } from "../../context/auth";
const Profile = () => {
  const [auth, setAuth] = useAuth();
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [phone, setPhone] = useState("");
```

```
const [address, setAddress] = useState("");
//Get User Data
useEffect(() => {
  const { email, name, phone, address, password } = auth.user;
  setName(name);
  setEmail(email);
  setPhone(phone);
  setAddress(address);
  setPassword();
}, [auth?.user]);
//Upadte Profile
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const { data } = await axios.put("/api/v1/auth/profile", {
      name,
      email,
      password,
      phone,
      address,
    });
    if (data?.error) {
      toast.error(data?.error);
    } else {
      setAuth({ ...auth, user: data?.updatedUser });
      let ls = localStorage.getItem("auth");
      ls = JSON.parse(ls);
      ls.user = data.updatedUser;
      localStorage.setItem("auth", JSON.stringify(ls));
      toast.success("Profile Updated Successfully");
    }
  } catch (error) {
    console.log(error);
    toast.error("Something went wrong");
```

```
  }
 };
 return (
  <>
   <Layout title={"Users"}>
    <div className="container-fluid m-3 p-3">
     <div className="row">
      <div className="col-md-3">
       <UserMenu />
      </div>
      <div className="col-md-9">
       <div className="card w-75 p-3">
        <h3>User's Profile</h3>
        <div className="form-container" style={{ minHeight: "76vh" }}>
         <form onSubmit={handleSubmit}>
          <div className="mb-3">
           <input
            type="text"
            value={name}
            onChange={(e) => setName(e.target.value)}
            className="form-control"
            id="name"
            placeholder="Enter Your Name"
            autoFocus
           />
          </div>
          <div className="mb-3">
           <input
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            className="form-control"
            id="email"
            placeholder="Enter Your Email "
```

```
    disabled
   />
  </div>
  <div className="mb-3">
   <input
    type="password"
    value={password}
    onChange={(e) => setPassword(e.target.value)}
    className="form-control"
    id="password"
    placeholder="Enter Your Password"
   />
  </div>
  <div className="mb-3">
   <input
    type="text"
    value={phone}
    onChange={(e) => setPhone(e.target.value)}
    className="form-control"
    id="phone"
    placeholder="Enter Your Phone"
   />
  </div>
  <div className="mb-3">
   <input
    type="text"
    value={address}
    onChange={(e) => setAddress(e.target.value)}
    className="form-control"
    id="address"
    placeholder="Enter Your Address"
   />
  </div>
  <button type="submit" className="btn btn-primary">
```

```
                UPDATE
              </button>
            </form>
          </div>
        </div>
      </div>
    </div>
  </Layout>
 </>
 );
};
export default Profile;
```

4.1.10 Client/ – Folder

❖ **App.js**

```
import { Routes, Route } from "react-router-dom";
import HomePage from "./pages/HomePage";
import About from "./pages/About";
import Contact from "./pages/Contact";
import Policy from "./pages/Policy";
import Pagenotfound from "./pages/Pagenotfound";
import Terms from "./pages/Terms";
import Register from "./pages/Auth/Register";
import Login from "./pages/Auth/Login";
import Dashboard from "./pages/User/Dashboard";
import PrivateRoute from "./components/Routes/Private";
import ForgetPassword from "./pages/Auth/ForgetPassword";
import AdminRoute from "./components/Routes/AdminRoute";
import AdminDashboard from "./pages/Admin/AdminDashboard";
import ManageCategory from "./pages/Admin/ManageCategory";
import AddFurniture from "./pages/Admin/AddFurniture";
import Users from "./pages/Admin/Users";
```

```
import Profile from "./pages/User/Profile";
import Orders from "./pages/User/Orders";
import Furnitures from "./pages/Admin/Furnitures";
import UpdateFurniture from "./pages/Admin/UpdateFurniture";
import Search from "./pages/Search";
import FurnitureDetails from "./pages/FurnitureDetails";
import Categories from "./pages/Categories";
import CategoryFurniture from "./pages/CategoryFurniture";
import CartPage from "./pages/CartPage";
import AdminOrders from "./pages/Admin/AdminOrders";
function App() {
 return (
  <>
   <Routes>
    <Route path="/" element={<HomePage />} />
    <Route path="/furniture/:slug" element={<FurnitureDetails />} />
    <Route path="/Categories" element={<Categories />} />
    <Route path="/Category/:slug" element={<CategoryFurniture />} />
    <Route path="/search" element={<Search />} />
    <Route path="/cart" element={<CartPage />} />
    <Route path="/dashboard">
     <Route path="user" element={<Dashboard />} />
     <Route path="user/profile" element={<Profile />} />
     <Route path="user/orders" element={<Orders />} />
    </Route>
    <Route path="/dashboard">
     <Route path="admin" element={<AdminDashboard />} />
     <Route path="admin/create-category" element={<ManageCategory />} />
     <Route path="admin/furnitures" element={<Furnitures />} />
     <Route path="admin/add-furniture" element={<AddFurniture />} />
     <Route
      path="admin/update-furniture/:slug"
      element={<UpdateFurniture />}
     />
```

```
        <Route path="admin/users" element={<Users />} />
        <Route path="admin/orders" element={<AdminOrders />} />
      </Route>
      <Route path="/register" element={<Register />} />
      <Route path="/forget-password" element={<ForgetPassword />} />
      <Route path="/login" element={<Login />} />
      <Route path="/about" element={<About />} />
      <Route path="/contact" element={<Contact />} />
      <Route path="/policy" element={<Policy />} />
      <Route path="/terms" element={<Terms />} />
      <Route path="*" element={<Pagenotfound />} />
    </Routes>
  </>
 );
}
export default App;
```

4.2 Back-End

4.2.1 /config/db.js

```
import mongoose from "mongoose";
const connectDB = async () => {
 try {
  const conn = await mongoose.connect(process.env.MONGODB_URL);
  console.log(`Connceted to MongoDB database ${conn.connection.host}`);
 } catch (error) {
  console.log(`Error in Mongodb ${error}`);
 }
};
export default connectDB;
```

4.2.2 /controller - Folder

❖ **authController.js**

```
import { comparePassword, hashPassword } from "../helpers/authHelper.js";
import userModel from "../models/userModel.js";
import orderModel from "../models/orderModel.js";
import JWT from "jsonwebtoken";
export const registerController = async (req, res) => {
  try {
    const { name, email, password, phone, address, answer } = req.body;
    //validation
    if (!name) {
      return res.send({ message: "Name is required" });
    }
    if (!email) {
      return res.send({ message: "Email is required" });
    }
    if (!password) {
      return res.send({ message: "Password is required" });
    }
    if (!phone) {
      return res.send({ message: "Phone Number is required" });
    }
    if (!address) {
      return res.send({ message: "Address is required" });
    }
    if (!answer) {
      return res.send({ message: "Address is required" });
    }
    function validatePassword(pass) {
      const uppercaseRegex = /[A-Z]/;
      const lowercaseRegex = /[a-z]/;
      const numberRegex = /[0-9]/;
      const symbolRegex = /[!@#$%^&*()_+\-=[\]{};':"\\|,.<>/?]/;
```

```
  const hasUppercase = uppercaseRegex.test(pass);
  const hasLowercase = lowercaseRegex.test(pass);
  const hasNumber = numberRegex.test(pass);
  const hasSymbol = symbolRegex.test(pass);
  return hasUppercase && hasLowercase && hasNumber && hasSymbol;
}
const isValid = validatePassword(password);
if (!isValid) {
  return res.send({ message: "Password is invalid." });
}
//Check user is unique or not
const existinguser = await userModel.findOne({ email });
//Existing user
if (existinguser) {
  return res.status(200).send({
    success: false,
    message: "Already Registed with this email, Please login",
  });
}
//Register New user
//Hash password
const hashedPassword = await hashPassword(password);
//save user data
const user = await new userModel({
  name,
  email,
  phone,
  address,
  answer,
  password: hashedPassword,
}).save();
res.status(201).send({
  success: true,
  message: "User Registeration is Successfull",
```

```
      user,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error in Registeration",
    });
  }
};
//POST LOGIN
export const loginController = async (req, res) => {
  try {
    const { email, password } = req.body;
    //Validation of inputs
    if (!email || !password) {
      return res
        .status(200)
        .send({ success: false, message: "Invalid Email or Password" });
    }
    //check user
    const user = await userModel.findOne({ email });
    if (!user) {
      return res
        .status(200)
        .send({ success: false, message: "Email is not registered" });
    }
    const match = await comparePassword(password, user.password);
    if (!match) {
      return res
        .status(200)
        .send({ success: false, message: "Invalid password" });
    }
    //token
```

```
    const token = await JWT.sign({ _id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "7d",
    });
    res.status(200).send({
      success: true,
      message: "Login Successfull",
      user: {
        name: user.name,
        email: user.email,
        phone: user.phone,
        address: user.address,
        role: user.role,
      },
      token,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error in Login",
      error,
    });
  }
};
//Forget-Password Controller
export const forgetpasswordController = async (req, res) => {
  try {
    const { email, newPassword, answer, phone } = req.body;
    //Validation of inputs
    if (!email) {
      return res.status(400).send({ message: "Email is required" });
    } else if (!newPassword) {
      return res.status(400).send({ message: "New Password is required" });
    } else if (!answer) {
```

```
      return res.status(400).send({ message: "Answer is required" });
    } else if (!phone) {
      return res.status(400).send({ message: "Phone is required" });
    }
    //check user
    const user = await userModel.findOne({ email });
    if (!user) {
      return res
        .status(200)
        .send({ success: false, message: "Entered Email is not Registered" });
    }
    if (!(answer === user.answer)) {
      return res.status(200).send({ success: false, message: "Wrong Answer" });
    } else if (!(phone === user.phone)) {
      return res
        .status(200)
        .send({ success: false, message: "Wrong Phone Number" });
    }
    const hashedpassword = await hashPassword(newPassword);
    await userModel.findByIdAndUpdate(user._id, { password: hashedpassword });
    res.status(200).send({
      success: true,
      message: "Password reset successfull",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Something went wrong",
      error,
    });
  }
};
//test Controller
```

```
export const testController = (req, res) => {
  try {
    res.send({
      name: "chethan",
    });
  } catch (error) {
    console.log(error);
    res.send({ error });
  }
};
//Profile Upadte Controller
export const upadteProfileController = async (req, res) => {
  try {
    const { name, email, password, address, phone } = req.body;
    const user = await userModel.findById(req.user._id);
    //password
    if (password && password.length < 6) {
      return res.json({ error: "Passsword is required and 6 character long" });
    }
    const hashedPassword = password ? await hashPassword(password) : undefined;
    const updatedUser = await userModel.findByIdAndUpdate(
      req.user._id,
      {
        name: name || user.name,
        password: hashedPassword || user.password,
        phone: phone || user.phone,
        address: address || user.address,
      },
      { new: true }
    );
    res.status(200).send({
      success: true,
      message: "Profile Updated Successfully",
      updatedUser,
```

```
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({ error });
  }
};
//Get Orders
export const getOrdersController = async (req, res) => {
  try {
    const orders = await orderModel
      .find({ buyer: req.user._id })
      .populate("furnitures", "-photo")
      .populate("buyer", "name");
    res.json(orders);
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error While getting Orders",
      error,
    });
  }
};
//Get All Orders for Admin
export const getAllOrdersController = async (req, res) => {
  try {
    const orders = await orderModel
      .find({})
      .populate("furnitures", "-photo")
      .populate("buyer", "name")
      .sort({ createdAt: "-1" });
    res.json(orders);
  } catch (error) {
    console.log(error);
```

```
    res.status(500).send({
      success: false,
      message: "Error While getting All Orders",
      error,
    });
  }
};
//Update Order
export const orderUpdateController = async (req, res) => {
  try {
    const { orderId } = req.params;
    const { status } = req.body;
    const orders = await orderModel.findByIdAndUpdate(
      orderId,
      { status },
      { new: true }
    );
    res.json(orders);
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error While updating order",
      error,
    });
  }
};
```

❖ **createCategoryController.js**

```
import categoryModel from "../models/categoryModel.js";
import slugify from "slugify";
export const createCategoryController = async (req, res) => {
  try {
    const { name } = req.body;
```

```
    if (!name) {
      return res.send({ message: "Name is required" });
    }
    //check category
    const existingcategory = await categoryModel.findOne({ name });
    if (existingcategory) {
      return res.status(200).send({
        success: true,
        message: "Category Aleady exist",
      });
    }
    const category = await new categoryModel({
      name,
      slug: slugify(name),
    }).save();
    res.status(200).send({
      success: true,
      message: "Category Created Successfull",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error in Category",
    });
  }
};
//Update category
export const updatecategoryController = async (req, res) => {
  try {
    const { name } = req.body;
    const { id } = req.params;
    const category = await categoryModel.findByIdAndUpdate(
```

```
      id,
      { name, slug: slugify(name) },
      { new: true }
    );
    res.status(200).send({
      success: true,
      message: "Category Updated",
      category,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error While Updating Category",
    });
  }
};
export const categoryController = async (req, res) => {
  try {
    const category = await categoryModel.find({});
    res.status(200).send({
      success: true,
      message: "All Category list",
      category,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error While Getting Category",
    });
  }
```

```
};
//singleCategoryController
export const singleCategoryController = async (req, res) => {
  try {
    const category = await categoryModel.findOne({ slug: req.params.slug });
    res.status(200).send({
      success: true,
      message: "All Category list",
      category,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error While Getting Single Category",
    });
  }
};
//Delete category
export const deleteCategoryController = async (req, res) => {
  try {
    const { id } = req.params;
    await categoryModel.findByIdAndDelete(id);
    res.status(200).send({
      success: true,
      message: "Successfully deleted Category",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error While Deleting Category",
```

```
    });
  }
};
```

❖ **furnitureController.js**

```
import slugify from "slugify";
import categoryModel from "../models/categoryModel.js";
import furnitureModel from "../models/furnitureModel.js";
import fs from "fs";
import braintree from "braintree";
import orderModel from "../models/orderModel.js";
import dotenv from "dotenv";
dotenv.config();
//Payment gateway
var gateway = new braintree.BraintreeGateway({
  environment: braintree.Environment.Sandbox,
  merchantId: process.env.BRAINTREE_MERCHANT_ID,
  publicKey: process.env.BRAINTREE_PUBLIC_KEY,
  privateKey: process.env.BRAINTREE_PRIVATE_KEY,
});
export const createFurnitureController = async (req, res) => {
  try {
    const { name, description, price, category, quantity, ratings } =
      req.fields;
    const { photo } = req.files;
    //Validation
    switch (true) {
      case !name:
        return res.status(500).send({ error: "Name is Required" });
      case !description:
        return res.status(500).send({ error: "Description is Required" });
      case !price:
        return res.status(500).send({ error: "Price is Required" });
      case !category:
```

```
      return res.status(500).send({ error: "Category is Required" });
    case !quantity:
      return res.status(500).send({ error: "Quantity is Required" });
    case photo && photo.size > 1000000:
      return res
        .status(500)
        .send({ error: "photo is Required and Should be less then  1mb" });
  }
  const furnitures = new furnitureModel({
    ...req.fields,
    slug: slugify(name),
  });
  if (photo) {
    furnitures.photo.data = fs.readFileSync(photo.path);
    furnitures.photo.contentType = photo.type;
  }
  await furnitures.save();
  res.status(200).send({
    success: true,
    message: "Furniture Created Successfull",
    furnitures,
  });
 } catch (error) {
  console.log(error);
  res.status(500).send({
    success: false,
    message: "Error while creating Furniture",
    error,
  });
 }
};
export const updateFurnitureController = async (req, res) => {
 try {
  const { name, description, price, category, quantity, ratings } =
```

```
  req.fields;
 const { photo } = req.files;
 //Validation
 switch (true) {
  case !name:
   return res.status(500).send({ error: "Name is Required" });
  case !description:
   return res.status(500).send({ error: "Description is Required" });
  case !price:
   return res.status(500).send({ error: "Price is Required" });
  case !category:
   return res.status(500).send({ error: "Category is Required" });
  case !quantity:
   return res.status(500).send({ error: "Quantity is Required" });
  case photo && photo.size > 1000000:
   return res
    .status(500)
    .send({ error: "photo is Required and Should be less then  1mb" });
 }
 const furnitures = await furnitureModel.findByIdAndUpdate(
  req.params.fid,
  { ...req.fields, slug: slugify(name) },
  { new: true }
 );
 if (photo) {
  furnitures.photo.data = fs.readFileSync(photo.path);
  furnitures.photo.contentType = photo.type;
 }
 await furnitures.save();
 res.status(200).send({
  success: true,
  message: "Furniture Updated Successfull",
  furnitures,
 });
```

```
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error while Updating Furniture",
      error,
    });
  }
};
export const getFurnitureController = async (req, res) => {
  try {
    const furnitures = await furnitureModel
      .find({})
      .select("-photo")
      .populate("category")
      .limit(12)
      .sort({ createdAt: -1 });
    res.status(200).send({
      success: true,
      total: furnitures.length,
      message: "All Furniture",
      furnitures,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error while Getting Furniture",
      error,
    });
  }
};
export const getSingleFurnitureController = async (req, res) => {
  try {
```

```
    const furniture = await furnitureModel
      .findOne({ slug: req.params.slug })
      .select("-photo")
      .populate("category");
    res.status(200).send({
      success: true,
      message: "Single Furniture",
      furniture,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error while Getting Single Furniture",
      error,
    });
  }
};
export const furniturePhotoController = async (req, res) => {
  try {
    const furniture = await furnitureModel
      .findById(req.params.fid)
      .select("photo");
    if (furniture.photo.data) {
      res.set("Content-type", furniture.photo.contentType);
      return res.status(200).send(furniture.photo.data);
    }
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error while Getting Furniture Photo",
      error,
    });
```

```
  }
};
export const deleteFurnitureController = async (req, res) => {
  try {
    await furnitureModel.findByIdAndDelete(req.params.fid).select("-photo");
    res.status(200).send({
      success: true,
      message: "Furniture Deleted",
    });
  } catch (error) {
    res.status(500).send({
      success: false,
      message: "Error while deleteing Furniture",
      error,
    });
  }
};
export const filtersFurnitureController = async (req, res) => {
  try {
    const { checked, radio } = req.body;
    let args = {};
    if (checked.length > 0) args.category = checked;
    if (radio.length) args.price = { $gte: radio[0], $lte: radio[1] };
    const furnitures = await furnitureModel.find(args);
    res.status(200).send({
      success: true,
      total: furnitures.length,
      furnitures,
    });
  } catch (error) {
    res.status(500).send({
      success: false,
      message: "Error while Filtering Furnitures",
      error,
```

```
    });
  }
};
//Furniture count
export const furnitureCountController = async (req, res) => {
  try {
    const total = await furnitureModel.find({}).estimatedDocumentCount();
    res.status(200).send({
      success: true,
      total,
    });
  } catch (error) {
    console.log(error),
      res.status(500).send({
        success: false,
        error,
        message: "Something went wrong in Furniture count",
      });
  }
};
// Get furniture list based on page
export const furnitureListController = async (req, res) => {
  try {
    const perPage = 3;
    const page = req.params.page ? req.params.page : 1;
    const furnitures = await furnitureModel
      .find({})
      .select("-photo")
      .skip((page - 1) * perPage)
      .limit(perPage)
      .sort({ createdAt: -1 });
    res.status(200).send({
      success: true,
      furnitures,
```

```
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Something went wrong in Furniture list",
    });
  }
};
//Search Furnitures
export const furnitureSearchController = async (req, res) => {
  try {
    const { keywords } = req.params;
    const result = await furnitureModel
      .find({
        $or: [
          { name: { $regex: keywords, $options: "i" } },
          { descripton: { $regex: keywords, $options: "i" } },
        ],
      })
      .select("-photo");
    res.json(result);
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Something went wrong in Search Furniture",
    });
  }
};
//Related Furniture Category
export const relatedFurnitureController = async (req, res) => {
```

```
  try {
    const { fid, cid } = req.params;
    const furnitures = await furnitureModel
      .find({
        category: cid,
        _id: { $ne: fid },
      })
      .select("-photo")
      .limit(3)
      .populate("category");
    res.status(200).send({
      success: true,
      furnitures,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Something went wrong in related Furniture",
    });
  }
};
//Furnitures by category
export const furnitureCategoryController = async (req, res) => {
  try {
    const category = await categoryModel.findOne({ slug: req.params.slug });
    const furnitures = await furnitureModel
      .find({ category })
      .populate("category");
    res.status(200).send({
      success: true,
      category,
      furnitures,
```

```
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Something went wrong in  Furniture category",
    });
  }
};
//Payment Gateway API
//Token
export const braintreeTokenController = async (req, res) => {
  try {
    gateway.clientToken.generate({}, function (err, response) {
      if (err) {
        res.status(500).send(err);
      } else {
        res.send(response);
      }
    });
  } catch (error) {
    console.log(error);
  }
};
//Payment API
export const braintreePaymentController = async (req, res) => {
  try {
    const { cart, nonce } = req.body;
    let total = 0;
    cart.map((i) => {
      total += i.price;
    });
    let newTransaction = gateway.transaction.sale(
```

```
      {
        amount: total,
        paymentMethodNonce: nonce,
        options: {
          submitForSettlement: true,
        },
      },
      function (error, result) {
        if (result) {
          const order = new orderModel({
            furnitures: cart,
            payment: result,
            buyer: req.user._id,
          }).save();
          res.json({ ok: true });
        } else {
          res.status(500).send(error);
        }
      }
    );
  } catch (error) {
    console.log(error);
  }
};
export const quantityController = async (req, res) => {
  try {
    const { cart } = req.body;
    const updatedFurniture = [];
    for (const item of cart) {
      const { _id, quantity } = item;
      const updatedQuantity = quantity - 1;
      const furniture = await furnitureModel.findByIdAndUpdate(
        _id,
        { quantity: updatedQuantity },
```

```
      { new: true }
    );
    updatedFurniture.push(furniture);
  }
  res.status(200).send({
    success: true,
    message: "Furniture Updated Successfully",
    // updatedFurniture,
  });
 } catch (error) {
  console.log(error);
  res.status(500).send({
    success: false,
    message: "Failed to update furniture quantity",
    error: error.message,
  });
 }
};
```

4.2.3 /Middlewares/authMiddleware.js

```
import JWT from "jsonwebtoken";
import userModel from "../models/userModel.js";
//Protected routes token base
export const requiresignin = async (req, res, next) => {
 try {
  const decode = JWT.verify(
    req.headers.authorization,
    process.env.JWT_SECRET
  );
  req.user = decode;
  next();
 } catch (error) {
  console.log(error);
 }
```

```
};
//Admin access
export const isAdmin = async (req, res, next) => {
  try {
    //check User is Admin or not
    const user = await userModel.findById(req.user._id);
    if (user.role !== 1) {
      return res.status(200).send({
        success: false,
        message: "UnAuthorized Access",
      });
    } else {
      next();
    }
  } catch (error) {}
};
```

4.2.4 /Models - Folder

❖ **categoryModel.js**

```
import mongoose from "mongoose";
const categoryschema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    unique: true,
  },
  slug: {
    type: String,
    lowercase: true,
  },
});
export default mongoose.model("Category", categoryschema);
```

❖ **furnitureModel.js**

```
import mongoose from "mongoose";
const furnitureschema = new mongoose.Schema(
 {
   name: {
     type: String,
     required: true,
   },
   slug: {
     type: String,
     required: true,
   },
   description: {
     type: String,
     required: true,
   },
   price: {
     type: String,
     required: true,
   },
   category: {
     type: mongoose.ObjectId,
     ref: "Category",
     required: true,
   },
   quantity: {
     type: String,
     required: true,
   },
   ratings: {
     type: String,
     required: true,
   },
```

```
    photo: {
      data: Buffer,
      contentType: String,
    },
    shipping: {
      type: Boolean,
    },
  },
  { timestamps: true }
);
export default mongoose.model("furnitures", furnitureschema);
```

❖ **orderModel.js**

```
import mongoose from "mongoose";
const orderschema = new mongoose.Schema(
  {
    furnitures: [
      {
        type: mongoose.ObjectId,
        ref: "furnitures",
      },
    ],
    payment: {},
    buyer: {
      type: mongoose.ObjectId,
      ref: "users",
    },
    status: {
      type: String,
      default: "Not Process",
      enum: ["Not Process", "Processing", "Shipped", "deliverd", "cancel"],
    },
  },
  { timestamps: true }
```

```
);
export default mongoose.model("Order", orderschema);
```

❖ **userModel.js**

```
import mongoose from "mongoose";
const userschema = new mongoose.Schema(
 {
   name: {
     type: String,
     required: true,
     trim: true,
   },
   email: {
     type: String,
     required: true,
     unique: true,
     trim: true,
   },
   password: {
     type: String,
     required: true,
     trim: true,
   },
   phone: {
     type: String,
     required: true,
     trim: true,
   },
   address: {
     type: {},
     required: true,
   },
   answer: {
     type: String,
```

```
    required: true,
   },
   role: {
    type: Number,
    default: 0,
   },
 },
 { timestamps: true }
);
export default mongoose.model("users", userschema);
```

4.2.5 /Routes - Folder

❖ **authRoute.js**

```
import Express from "express";
import {
 registerController,
 loginController,
 testController,
 forgetpasswordController,
 upadteProfileController,
 getOrdersController,
 getAllOrdersController,
 orderUpdateController,
} from "../controllers/authController.js";
import { isAdmin, requiresignin } from "../middlewares/authMiddleware.js";
//router object
const router = Express.Router();
//routing
//REGISTER || METHOD POST
router.post("/register", registerController);
//LOGIN || POST
router.post("/login", loginController);
//FORGET-PASSWORD || POST
router.post("/forget-password", forgetpasswordController);
```

```
//test routes
router.get("/test", requiresignin, isAdmin, testController);
//Protected User route auth
router.get("/user-auth", requiresignin, (req, res) => {
  res.status(200).send({ ok: true });
});
//Protected Admin route auth
router.get("/admin-auth", requiresignin, isAdmin, (req, res) => {
  res.status(200).send({ ok: true });
});
//Update profile
router.put("/profile", requiresignin, upadteProfileController);
//Orders
router.get("/orders", requiresignin, getOrdersController);
//All Orders
router.get("/all-orders", requiresignin, isAdmin, getAllOrdersController);
//Order Status Update
router.put(
  "/order-status/:orderId",
  requiresignin,
  isAdmin,
  orderUpdateController
);
export default router;
```

❖ **categoryRoutes.js**

```
import Express from "express";
import { isAdmin, requiresignin } from "../middlewares/authMiddleware.js";
import {
  categoryController,
  createCategoryController,
  deleteCategoryController,
  singleCategoryController,
  updatecategoryController,
```

```
} from "../controllers/createCategoryController.js";
const router = Express.Router();
//routes
//Create category
router.post(
  "/create-category",
  requiresignin,
  isAdmin,
  createCategoryController
);
//Upadte Category
router.put(
  "/update-category/:id",
  requiresignin,
  isAdmin,
  updatecategoryController
);
//Get All Category
router.get("/get-category", categoryController);
//Get single Category
router.get("/single-category/:slug", singleCategoryController);
//Delete Category
router.delete(
  "/delete-category/:id",
  requiresignin,
  isAdmin,
  deleteCategoryController
);
export default router;
```

- **furnitureRoutes.js**

```
import Express from "express";
import { isAdmin, requiresignin } from "../middlewares/authMiddleware.js";
import {
```

```
  createFurnitureController,
  deleteFurnitureController,
  filtersFurnitureController,
  furnitureCountController,
  furnitureListController,
  furniturePhotoController,
  getFurnitureController,
  getSingleFurnitureController,
  updateFurnitureController,
  furnitureSearchController,
  relatedFurnitureController,
  furnitureCategoryController,
  braintreeTokenController,
  braintreePaymentController,
  quantityController,
} from "../controllers/furnitureController.js";
import formidable from "express-formidable";
const router = Express.Router();
//Furniture Routes
router.post(
  "/create-furniture",
  requiresignin,
  isAdmin,
  formidable(),
  createFurnitureController
);
router.put(
  "/update-furniture/:fid",
  requiresignin,
  isAdmin,
  formidable(),
  updateFurnitureController
);
//Get Furniture
```

```
router.get("/get-furniture", getFurnitureController);
//Single Furniture
router.get("/single-furniture/:slug", getSingleFurnitureController);
//Furniture Photo
router.get("/furniture-photo/:fid", furniturePhotoController);
//Delete Furniture
router.delete("/delete-furniture/:fid", deleteFurnitureController);
//Filter Furnitures
router.post("/filters-furniture", filtersFurnitureController);
//Furnitures list count
router.get("/furniture-count", furnitureCountController);
//furniture per page
router.get("/furniture-list/:page", furnitureListController);
//Seacrch Furniture
router.get("/search/:keywords", furnitureSearchController);
//Similar furnitures
router.get("/related-furniture/:fid/:cid", relatedFurnitureController);
//category wise furnitures
router.get("/furniture-category/:slug", furnitureCategoryController);
//payments routes
//token
router.get("/braintree/token", braintreeTokenController);
//New Order payment
router.post(
  "/braintree/payment",
  requiresignin,
  braintreePaymentController,
  quantityController
);
// router.put("/quantity", requiresignin, quantityController);
export default router;
```

❖ **server.js**

```
import express from "express";
```

```
import dotenv from "dotenv";
import morgan from "morgan";
import connectDB from "./config/db.js";
import authRoutes from "./routes/authRoute.js";
import categoryRoutes from "./routes/categoryRoutes.js";
import furnitureRoutes from "./routes/furnitureRoutes.js";
import cors from "cors";
//configure env
dotenv.config();
//Database config
connectDB();
//rest object
const app = express();
//middlewares
app.use(cors());
app.use(express.json());
app.use(morgan("dev"));
//routes
app.use("/api/v1/auth", authRoutes);
app.use("/api/v1/category", categoryRoutes);
app.use("/api/v1/furniture", furnitureRoutes);
//rest API
app.get("/", (req, res) => {
  res.send("WORKING");
});
//Back-end Port Number
const PORT = process.env.PORT || 8080;
//run listen
app.listen(PORT, () => {
  console.log(`Backend server is running on port ${PORT}`);
});
```

5. IMPLEMENTATION

5.1 Screenshots

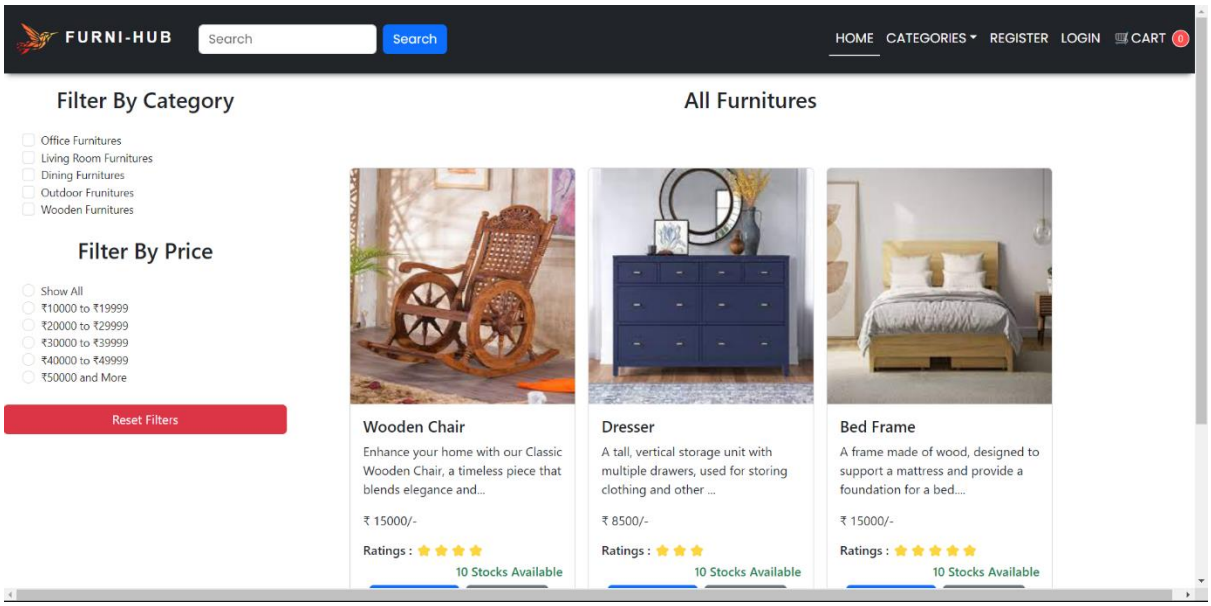

Fig. No. 5.1: Home Page Before Login

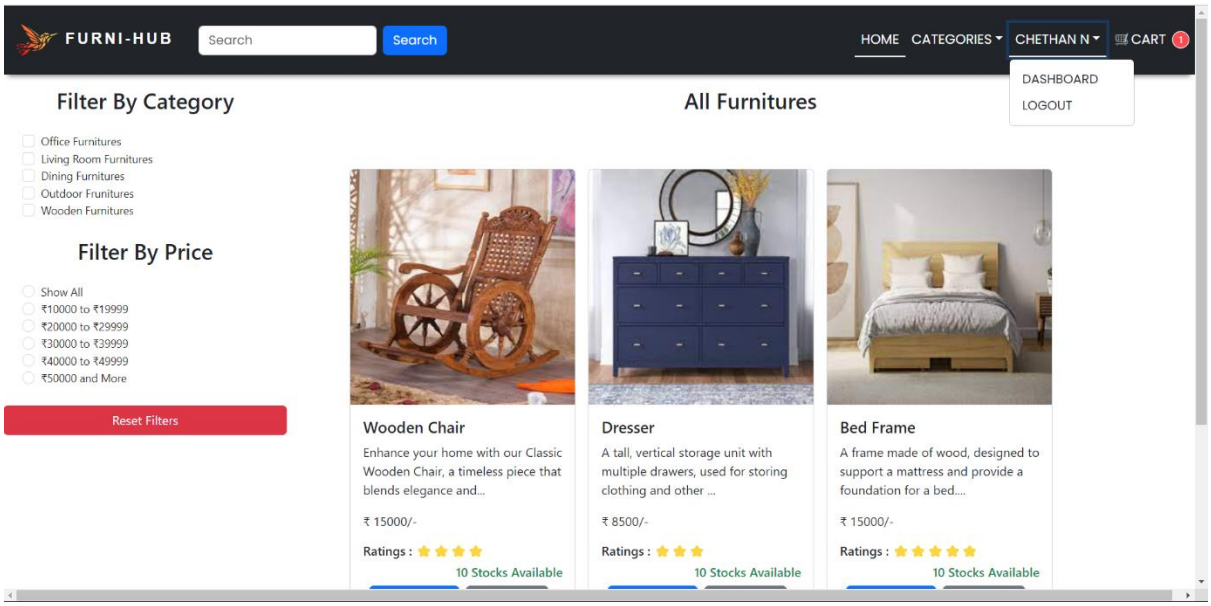

Fig. No. 5.2: Home Page After Login

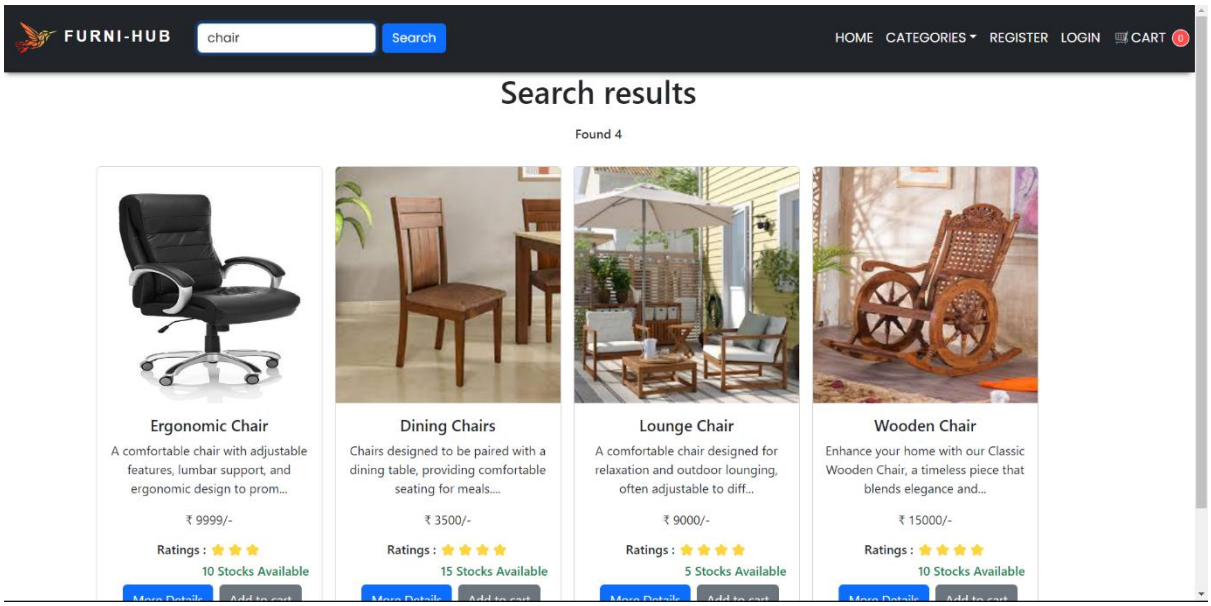

Fig. No. 5.3: Search Page

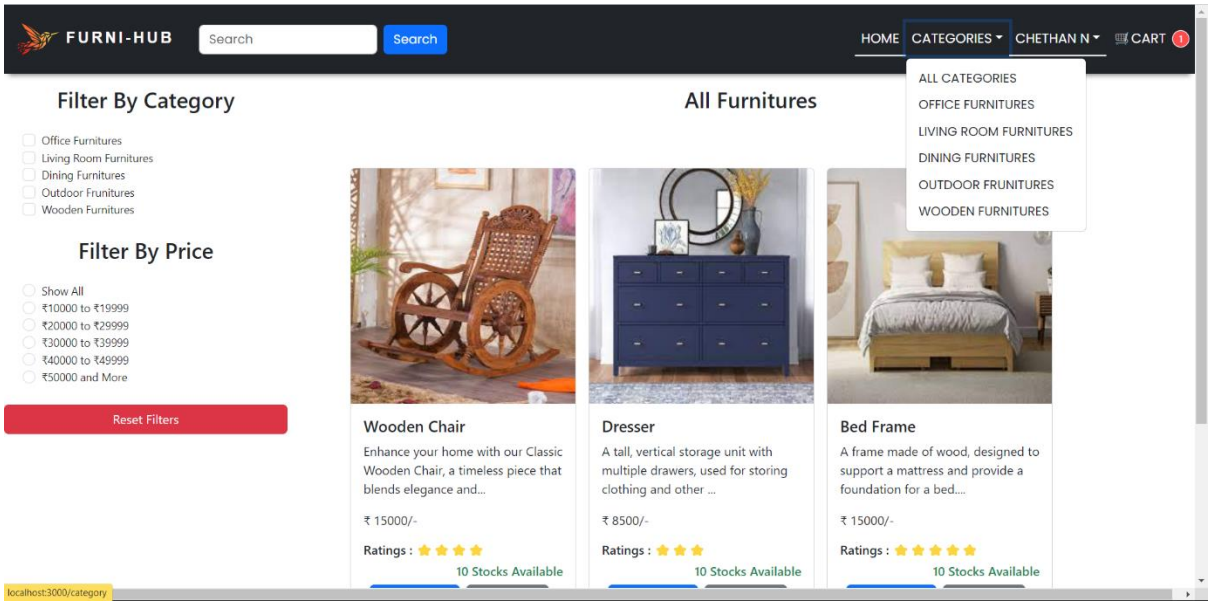

Fig. No. 5.4: Home Categories Page

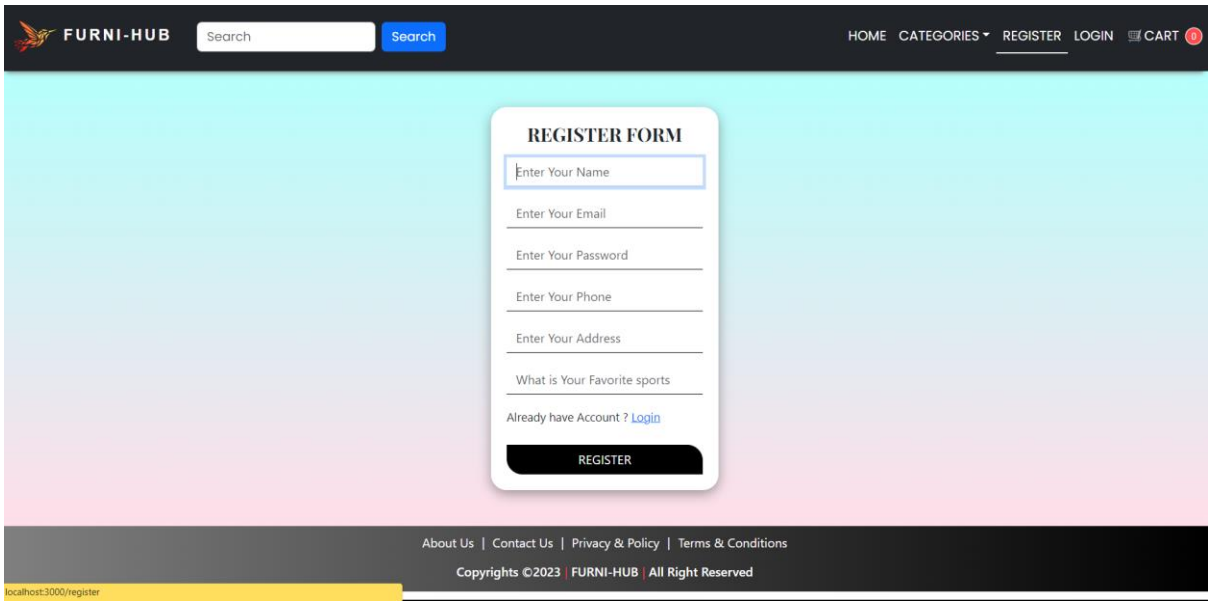

Fig. No. 5.5: Register Page

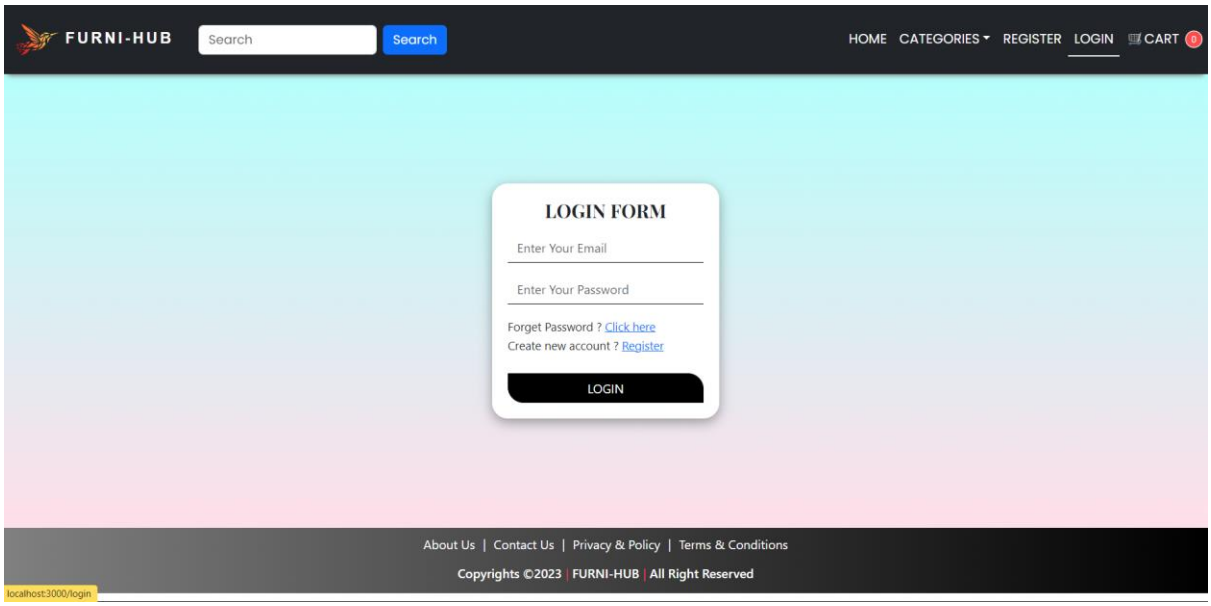

Fig. No. 5.6: Login Page

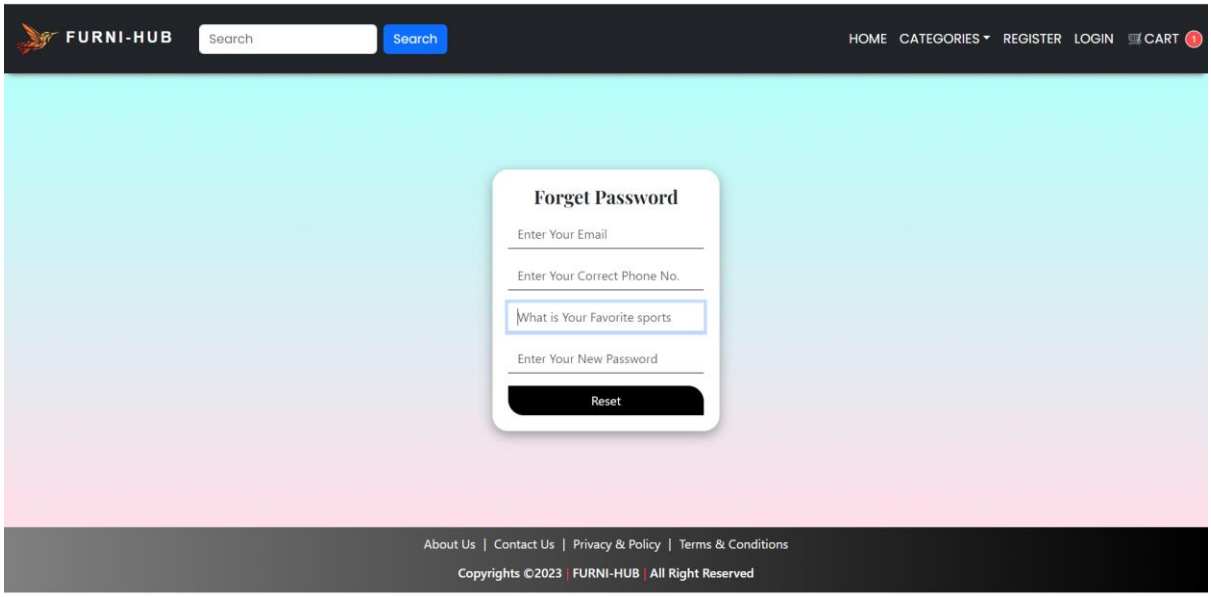

Fig. No. 5.7: Forget Password Page

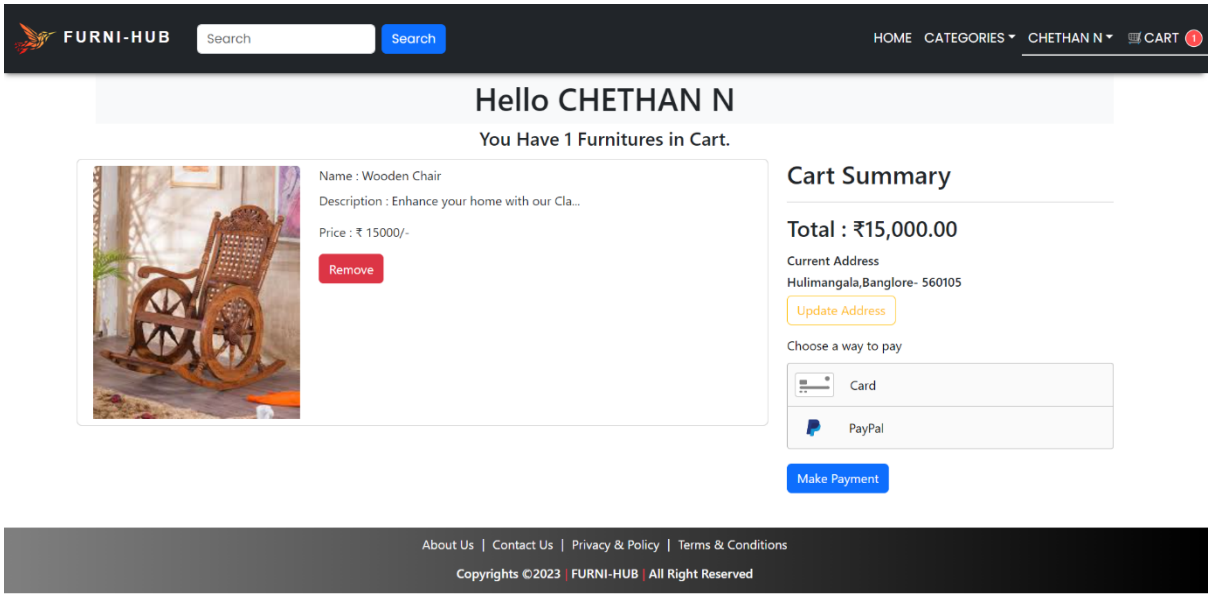

Fig. No. 5.8: Cart Page

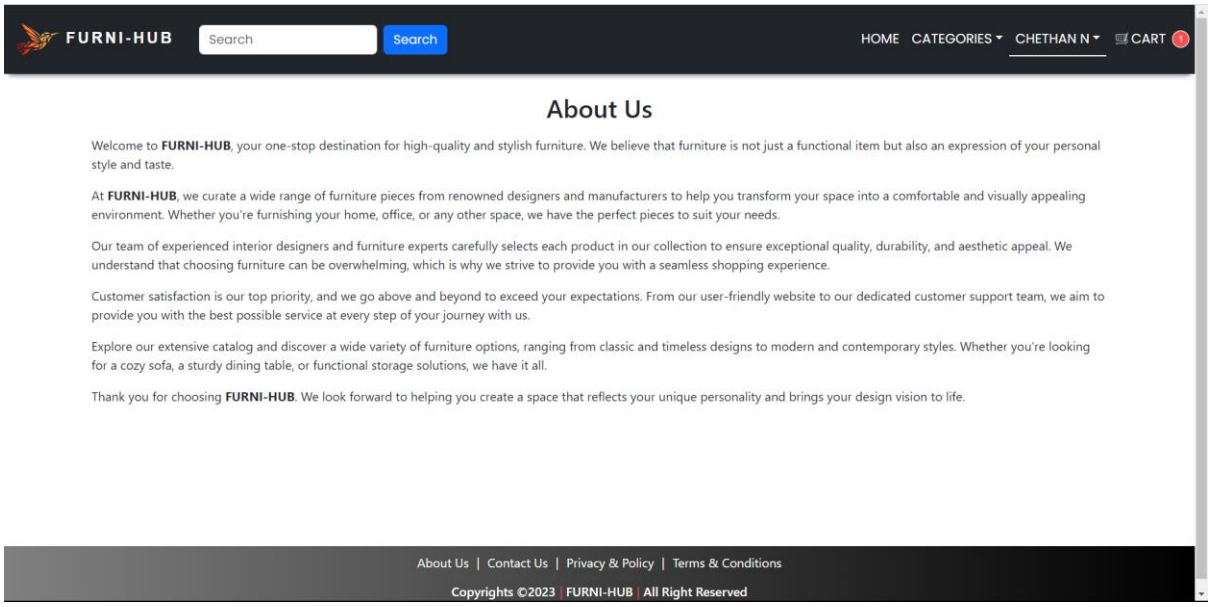

Fig. No. 5.9: About us Page

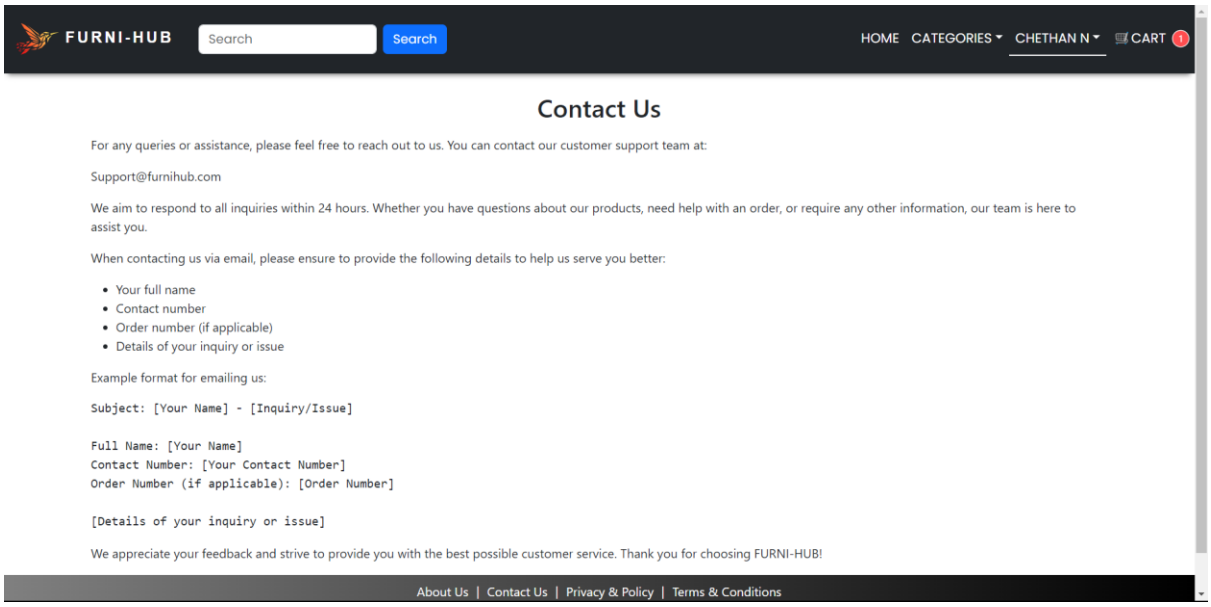

Fig. No. 5.10: Contact us Page

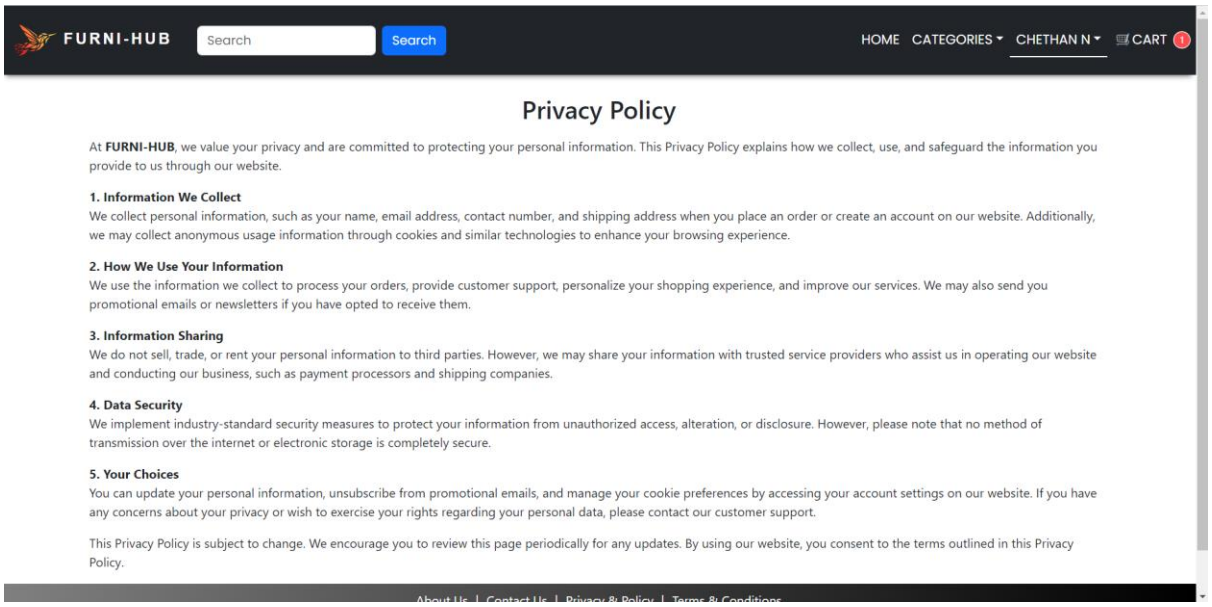

Fig. No. 5.11: Privacy Policy Page

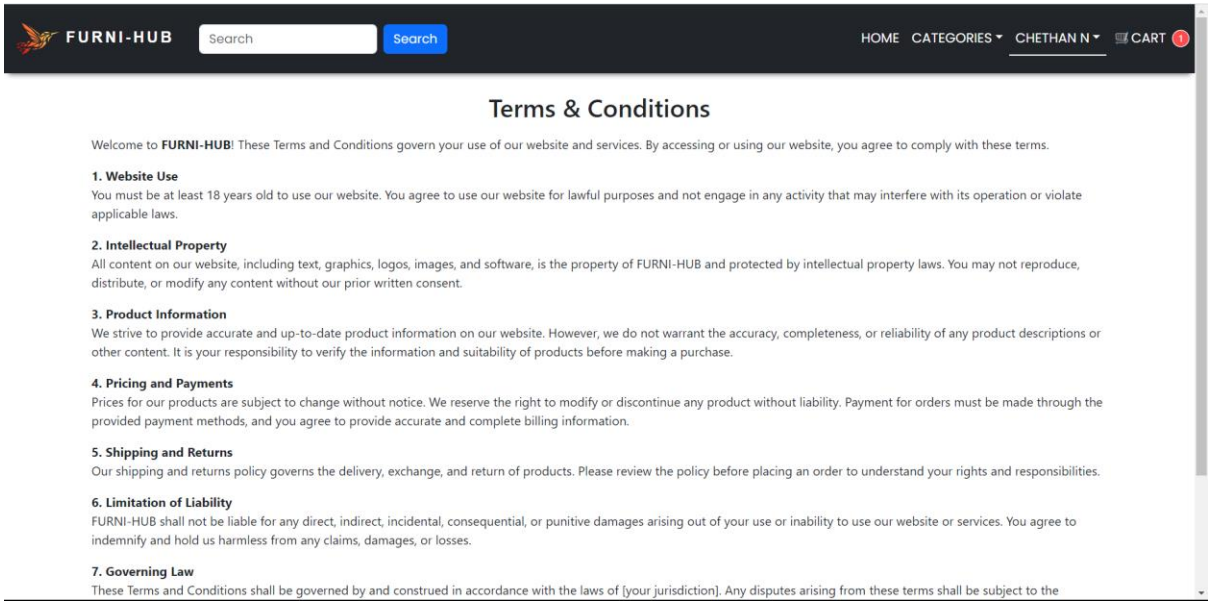

Fig. No. 5.12: Terms and Conditions Page

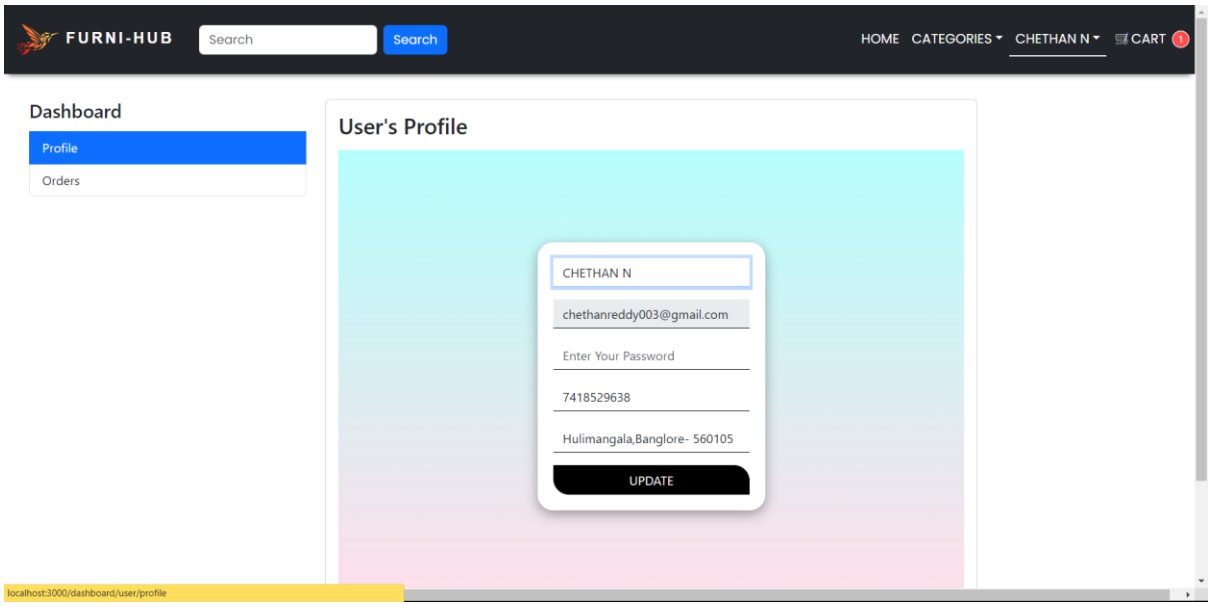

Fig. No. 5.13: User's Profile Page

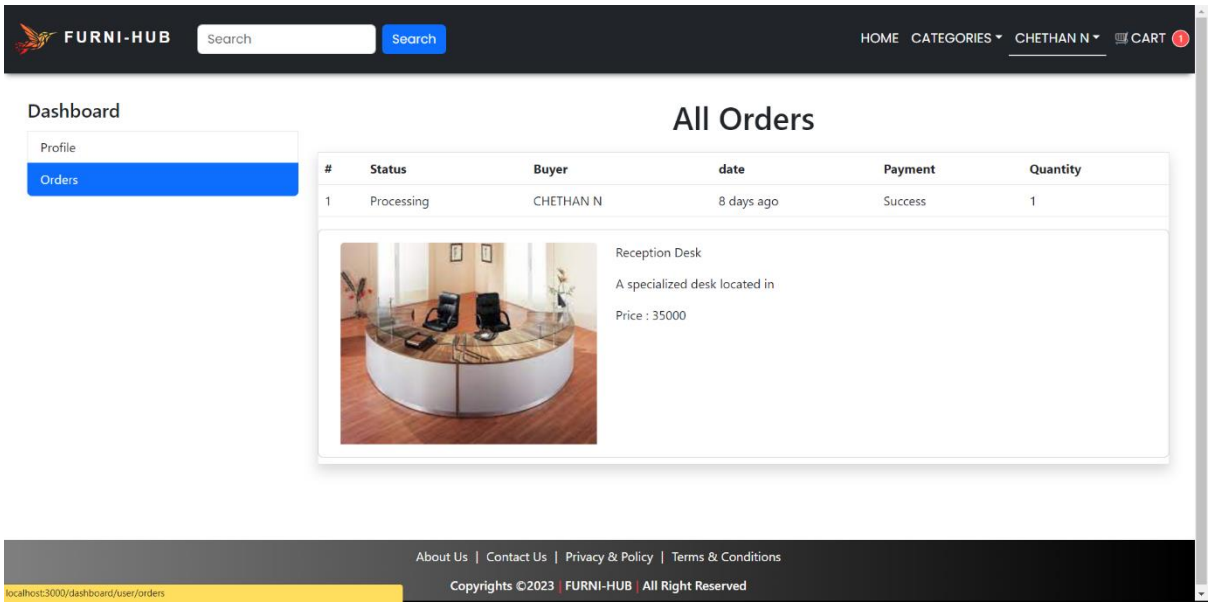

Fig. No. 5.14: User's Recent Orders Page

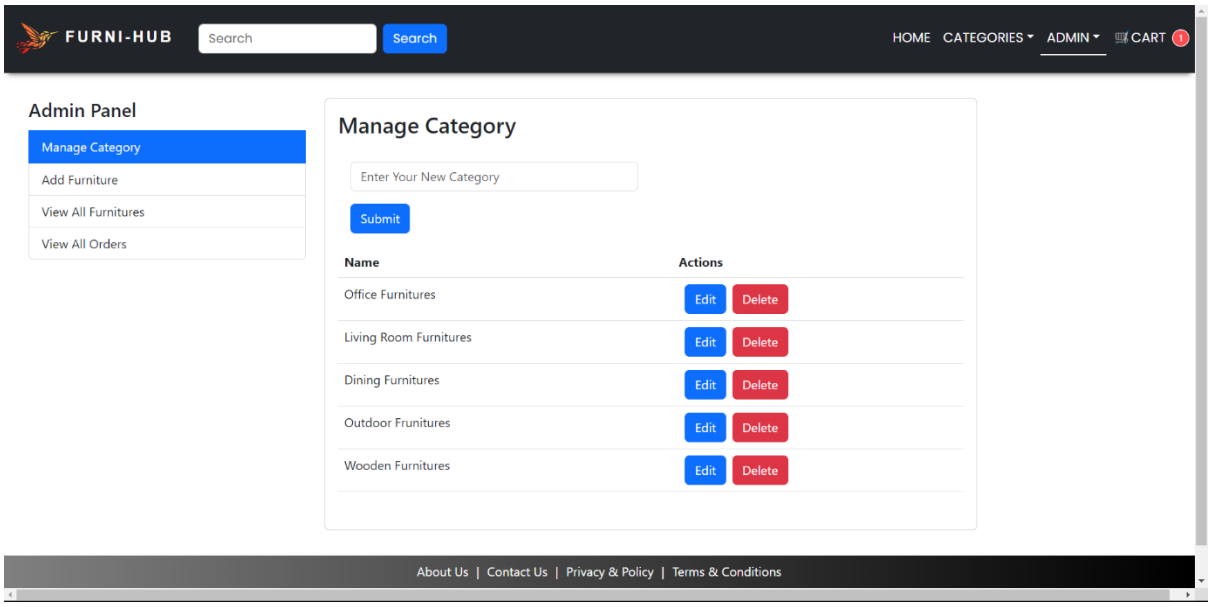

Fig. No. 5.15: Admin's Manage Category Page

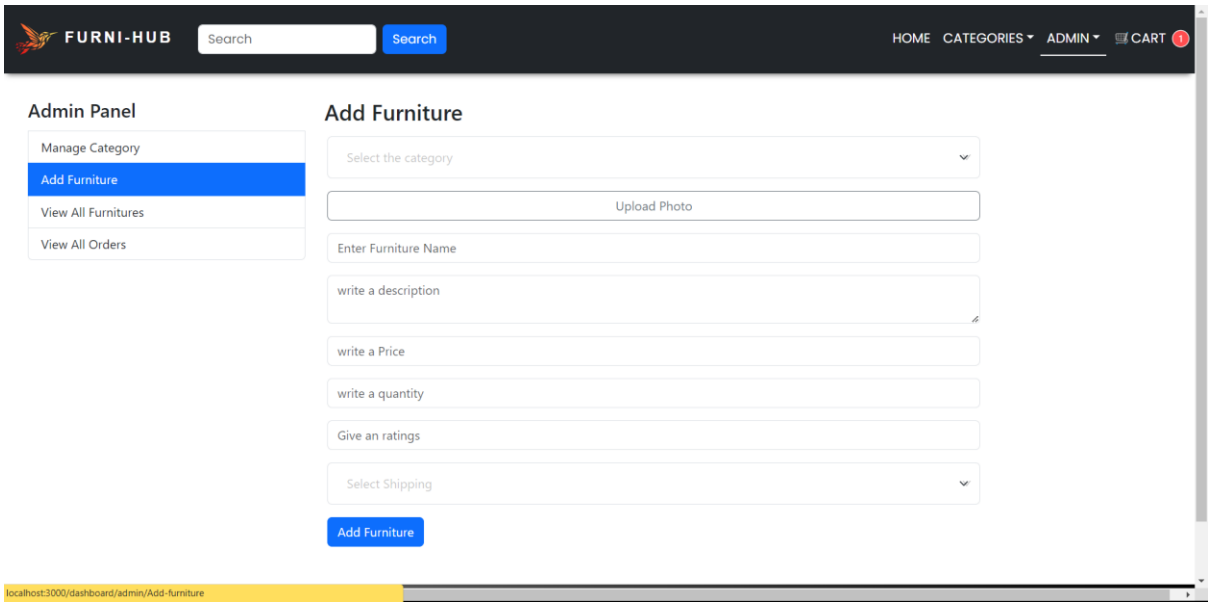

Fig. No. 5.16: Admin's Add Furniture Page

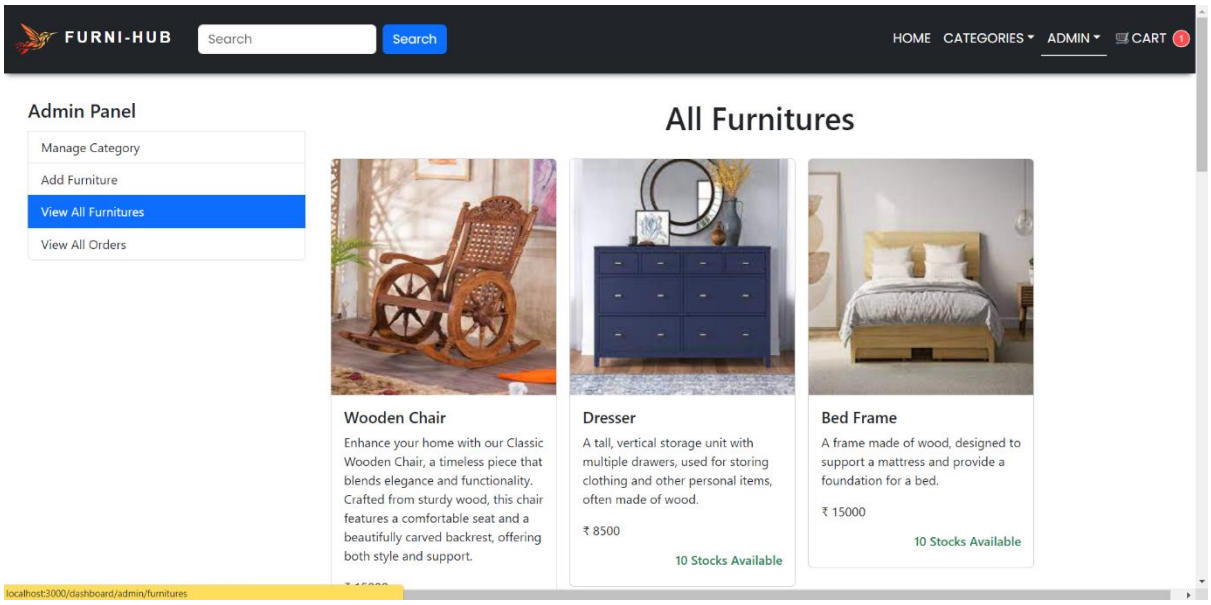

Fig. No. 5.17: Admin's Manage Furnitures Page

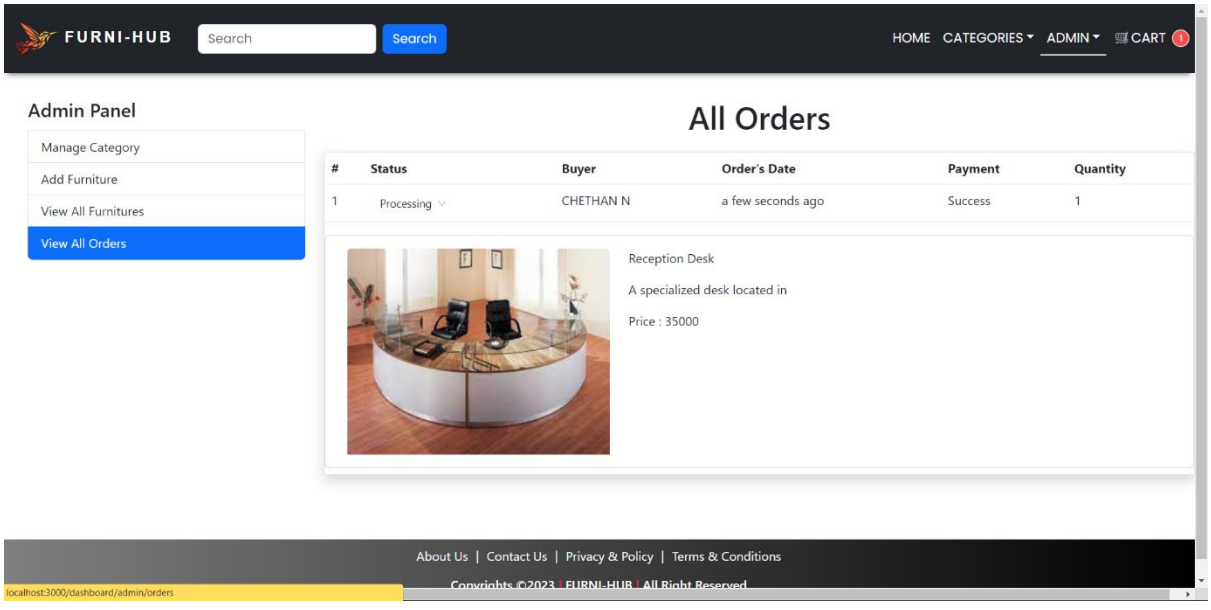

Fig. No. 5.18: Admin's All Orders Page

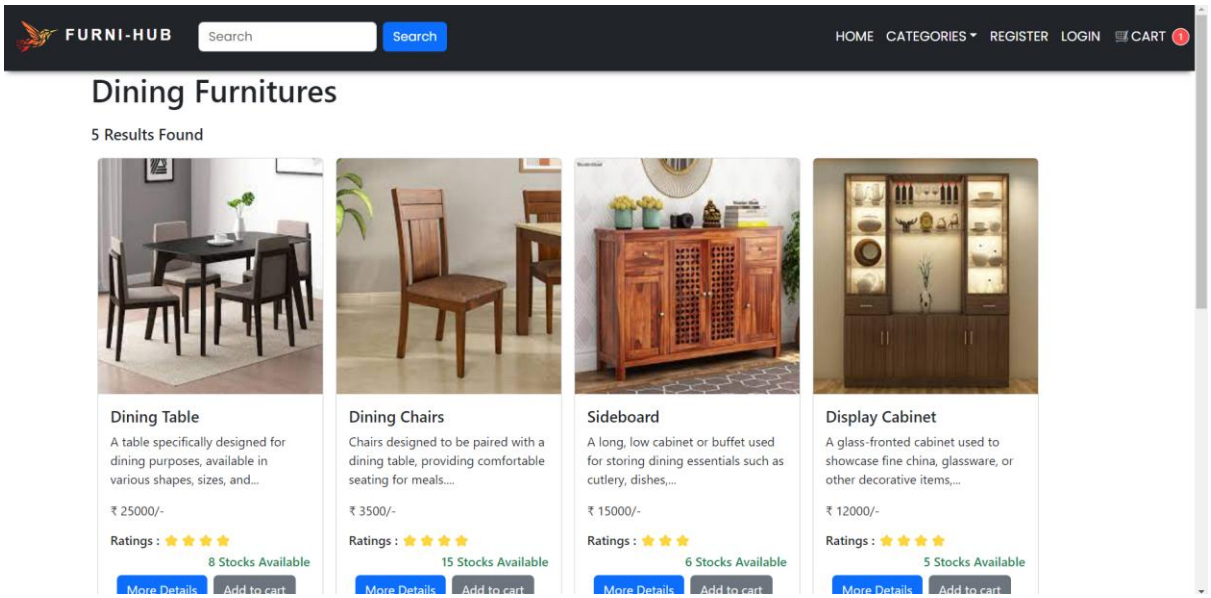

Fig. No. 5.19: Category Page

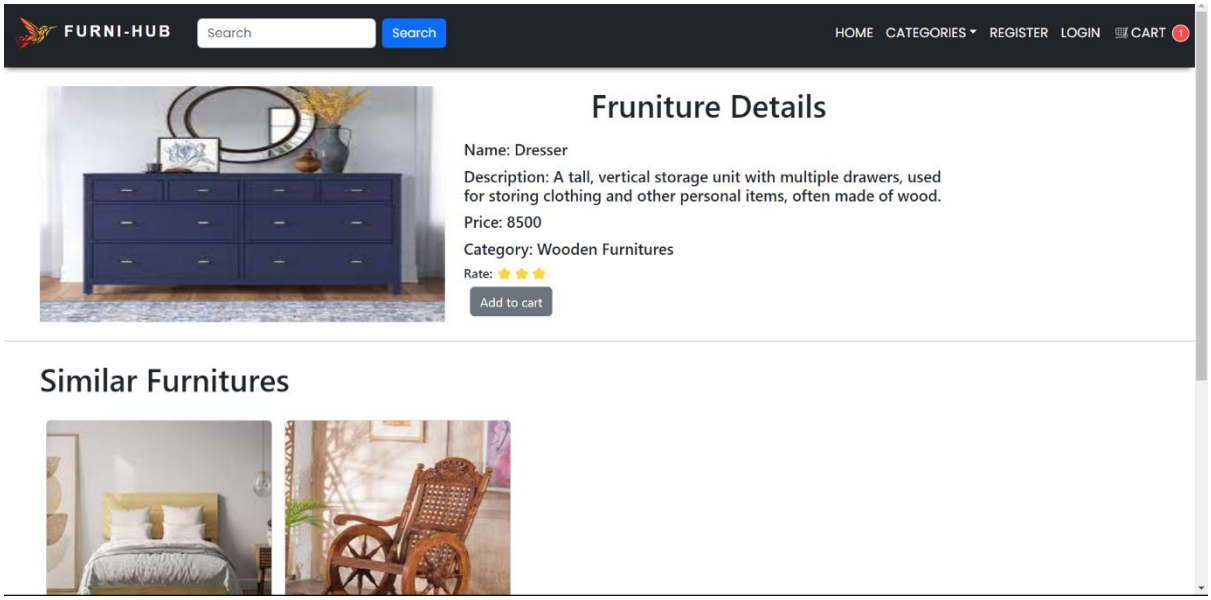

Fig. No. 5.20: More Details Page

6. TESTING

In general, testing is finding out how well something works. In terms of human beings, testing tells what level of knowledge or skill has been acquired. In computer hardware and software development, testing is used at key checkpoints in the overall process to determine whether objectives are being met.

6.1 System Testing

System testing is a type of software testing that evaluates the complete and fully integrated system to verify that it meets the specified requirements and works as intended. System testing is performed after integration testing and focuses on verifying the system's behavior in a real-world scenario, including performance, security, and other non-functional requirements. The objective of system testing is to validate that the system meets the business and user requirements, and to identify and fix any defects or issues before the system is released to the end-users. System testing can be performed manually or with the help of automated tools, and typically involves testing the system with realistic data and user scenarios to simulate real-world usage. System testing is an important step in the software development process as it ensures that the system meets the quality standards and is ready for deployment.

6.2 Integration Testing

Integration testing is a type of software testing where individual units or components of the software are combined and tested as a group to verify the interactions between them. The purpose of integration testing is to validate that the interfaces between the components work correctly, and that the components function as intended when integrated into the larger system. Integration testing is performed after unit testing and focuses on verifying the interaction and communication between components, as well as the overall functionality of the system as a whole. Integration testing can be performed manually or with the help of automated tools and typically involves testing the system with realistic data and user scenarios to simulate real-world usage. Integration testing is an important step in the software development process as it helps to identify and resolve any issues with the integration of components before the system is tested at the system level.

6.3 Unit Testing

Unit testing is a type of software testing where individual units or components of the software are tested in isolation from the rest of the system. The purpose of unit testing is to validate that each unit of the system performs as intended and meets its functional and design specifications. Unit tests are typically automated and are run frequently during the development process to catch any issues early on. This type of testing is performed by developers and focuses on small, isolated pieces of code, such as individual functions or methods. Unit testing helps to identify and fix bugs early in the development process, which can save time and resources later on in the software development cycle. Additionally, unit tests serve as a documentation of the expected behavior of the system, making it easier for developers to maintain and update the code over time.

6.4 Test Cases

Test Case – 01 (Register Form)

SI NO.	Description	Expected Result	Actual Result	Status
01	Register Form	All Valid Details	Empty/ Invalid Name	Fail
		All Valid Details	Empty / Invalid Number	Fail
		All Valid Details	Empty / Invalid Email	Fail
		All Valid Details	Empty / Invalid Password	Fail
		All Valid Details	Empty Address	Fail
		All Valid Details	Empty Answer	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.1

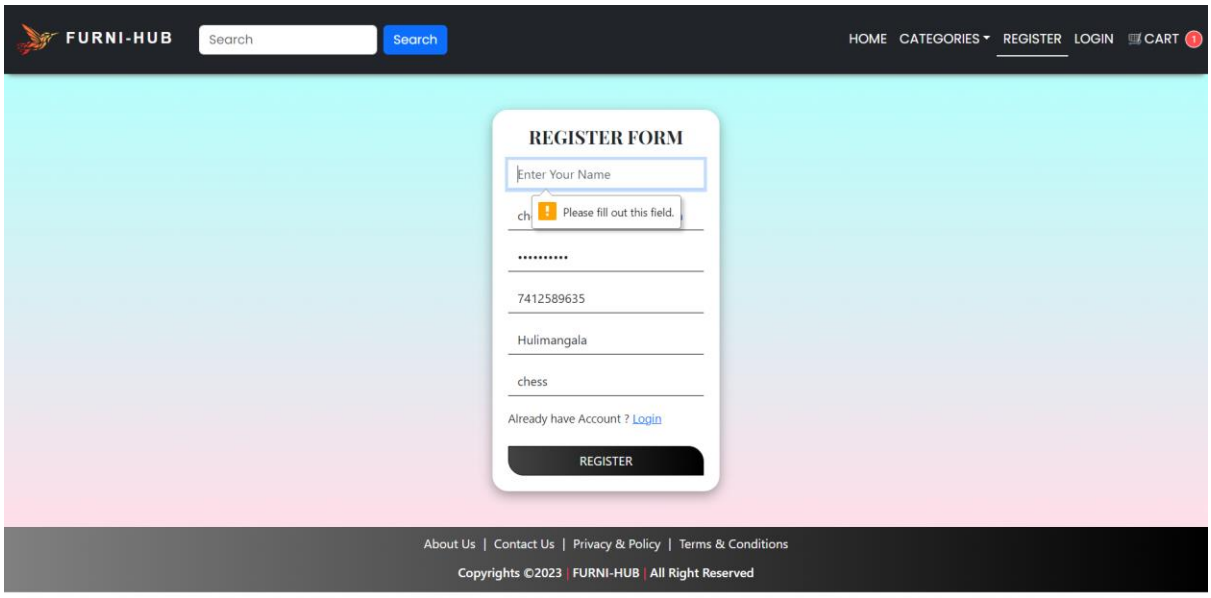

Fig. No. 6.1: Empty Name Field

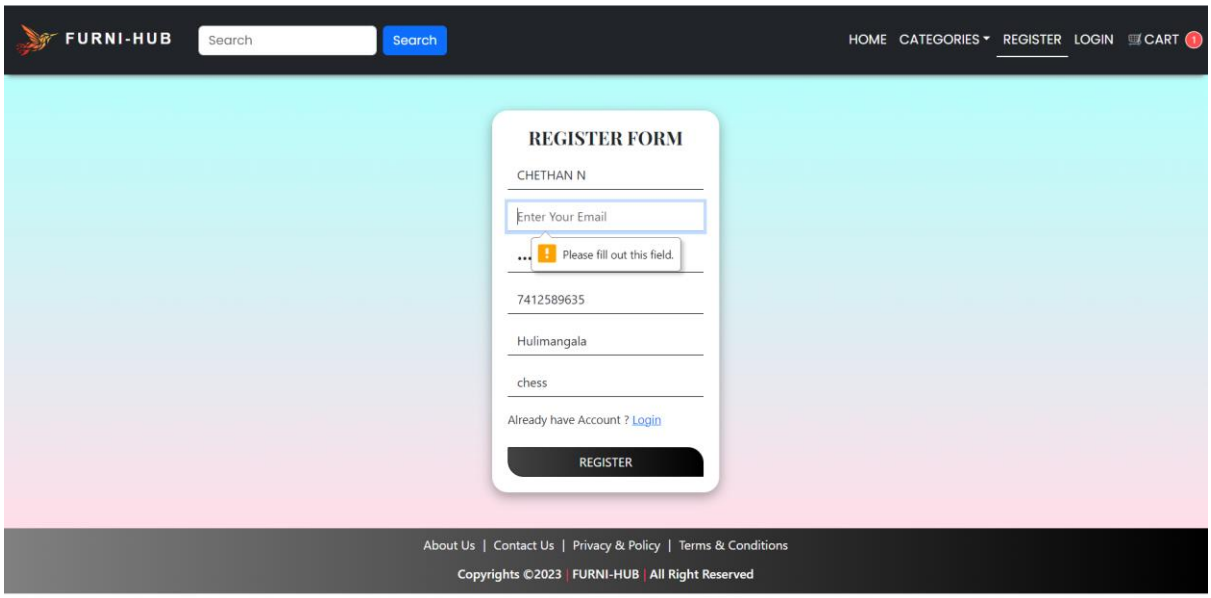

Fig. No. 6.2:Empty Email Field

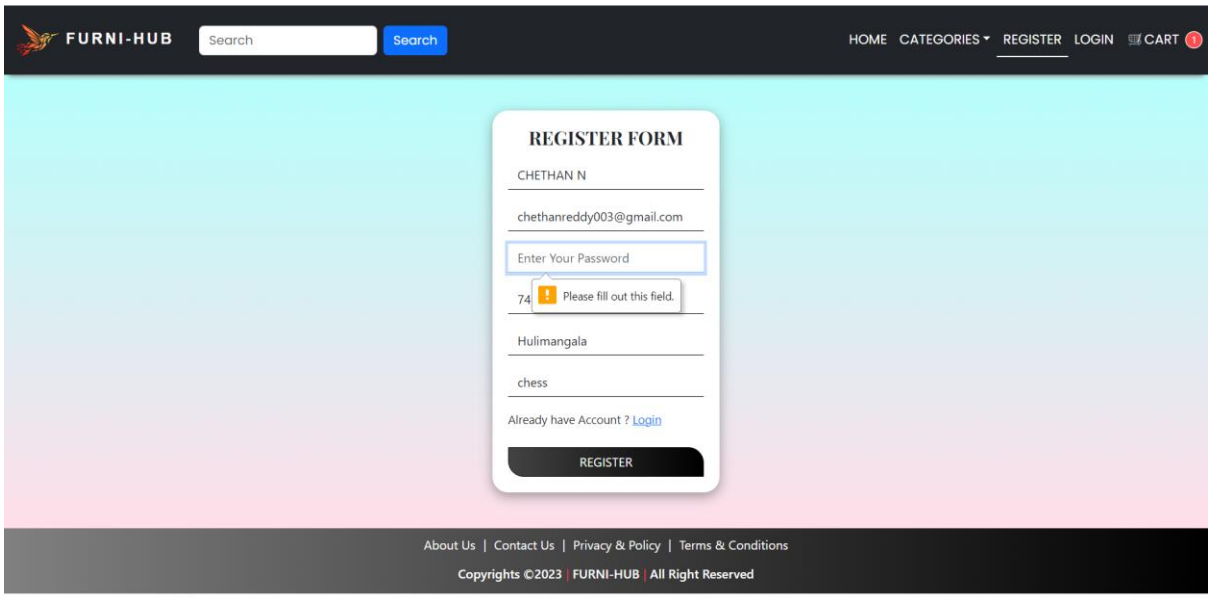

Fig. No. 6.3:Empty Password Field

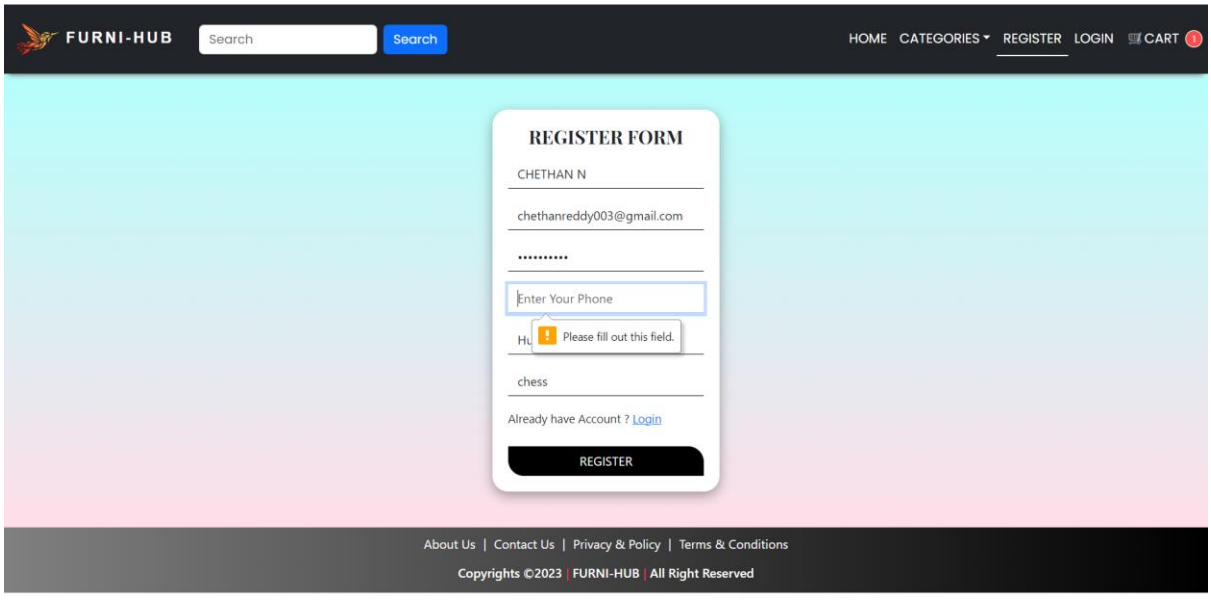

Fig. No. 6.4:Empty Number Field

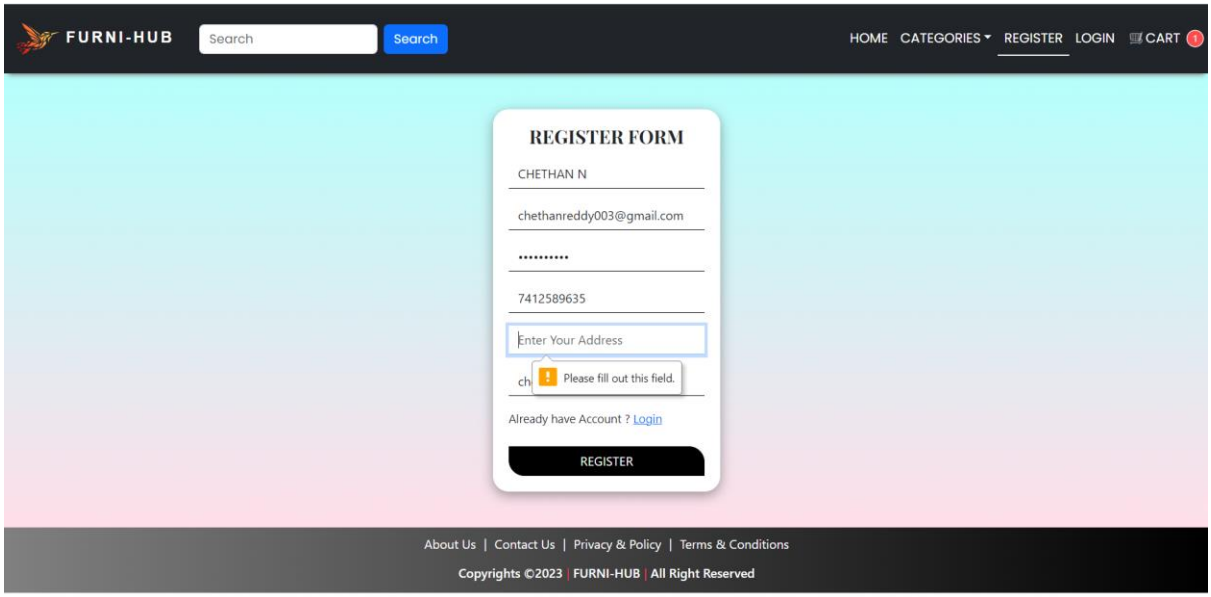

Fig. No. 6.5: Empty Address Field

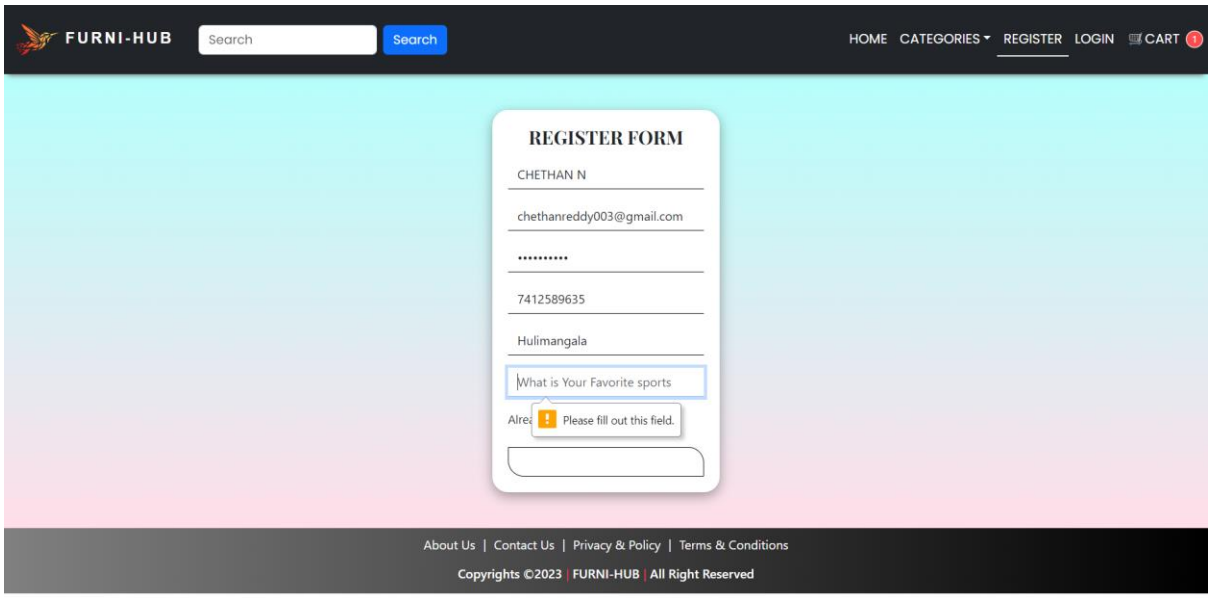

Fig. No. 6.6: Empty Answer Field

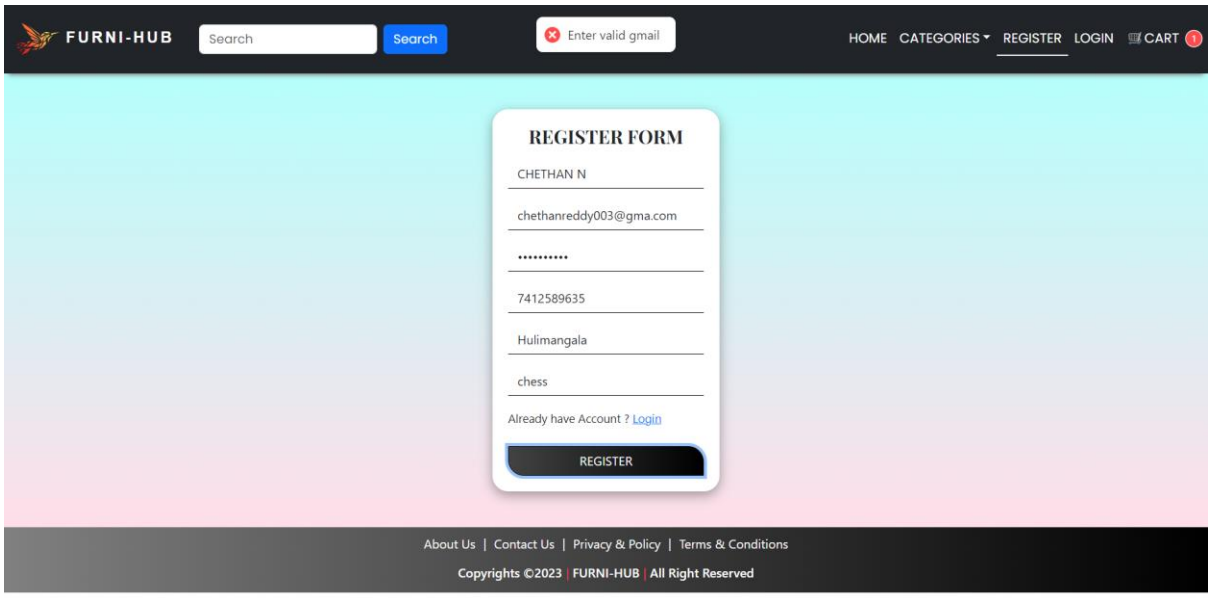

Fig. No. 6.7: Invalid Email

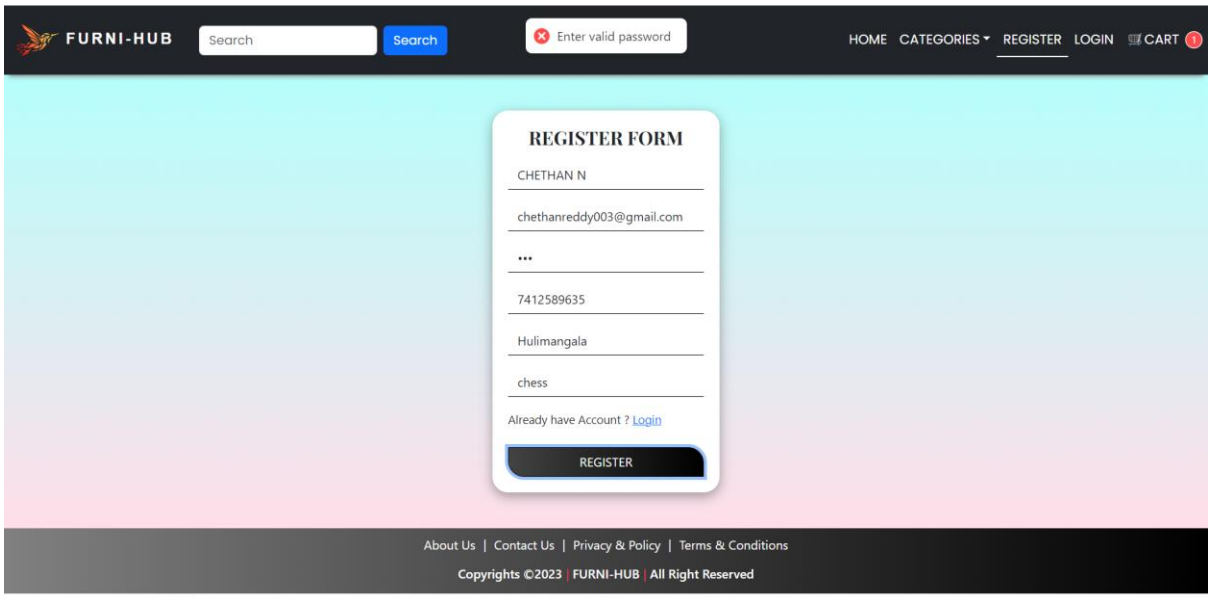

Fig. No. 6.8: Invalid Password

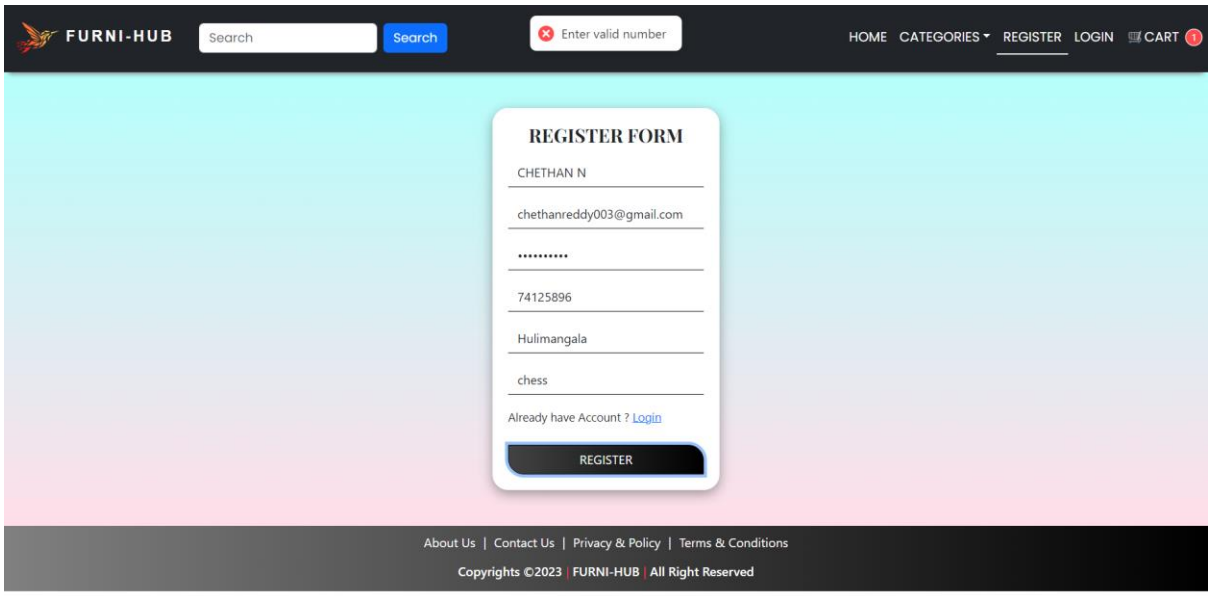

Fig. No. 6.9: Invalid Number

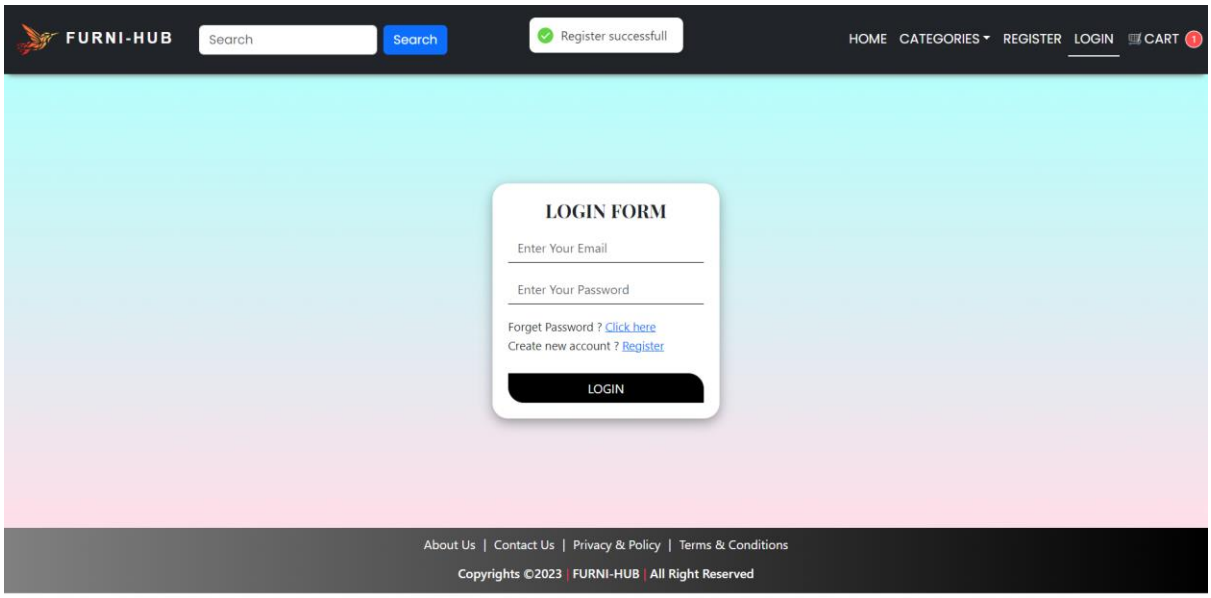

Fig. No. 6.10: Registration successful

Test Case – 02 (Login Form)

SI NO.	Description	Expected Result	Actual Result	Status
02	Login Form	All Valid Details	Empty / Invalid Email	Fail
		All Valid Details	Empty / Invalid Password	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.2

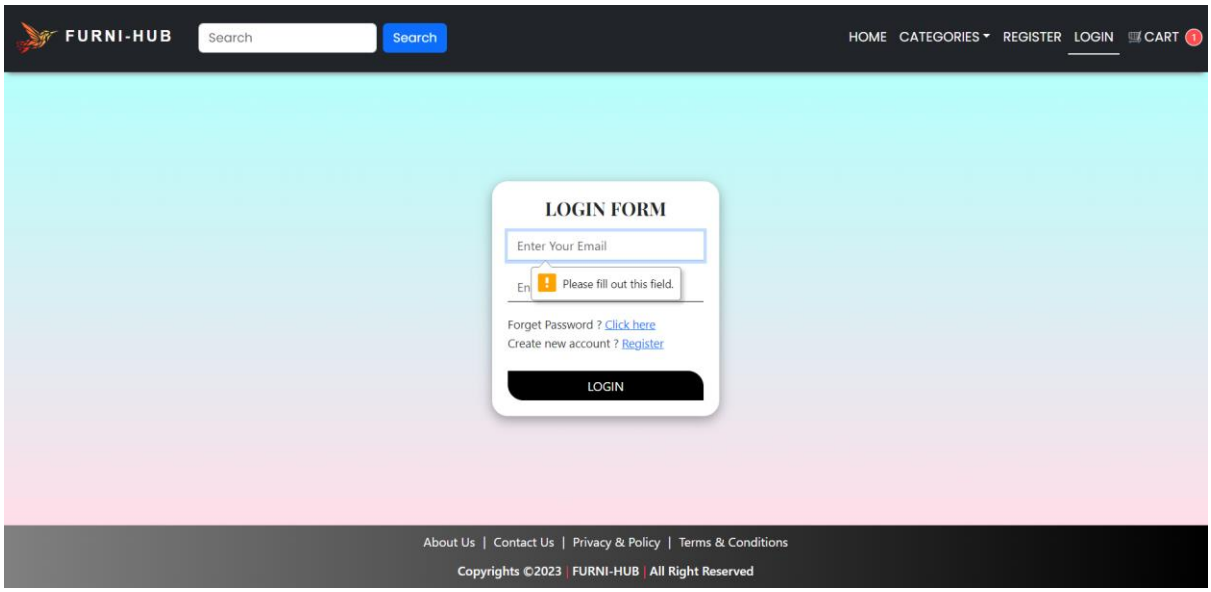

Fig. No. 6.11: Empty Email Field

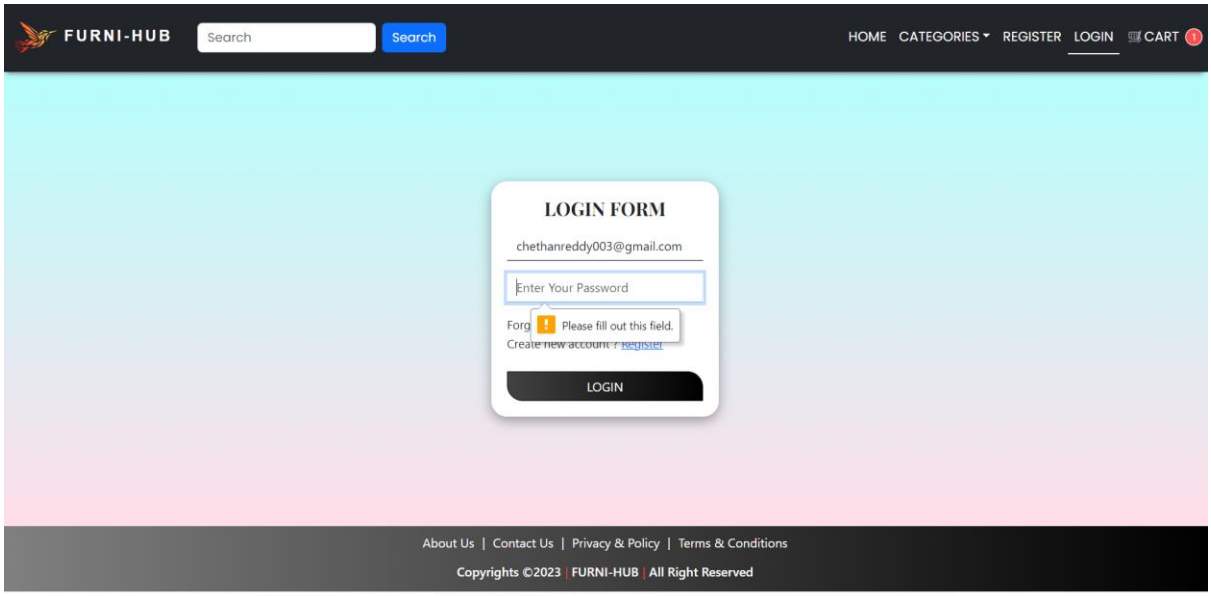

Fig. No. 6.12: Empty Password Field

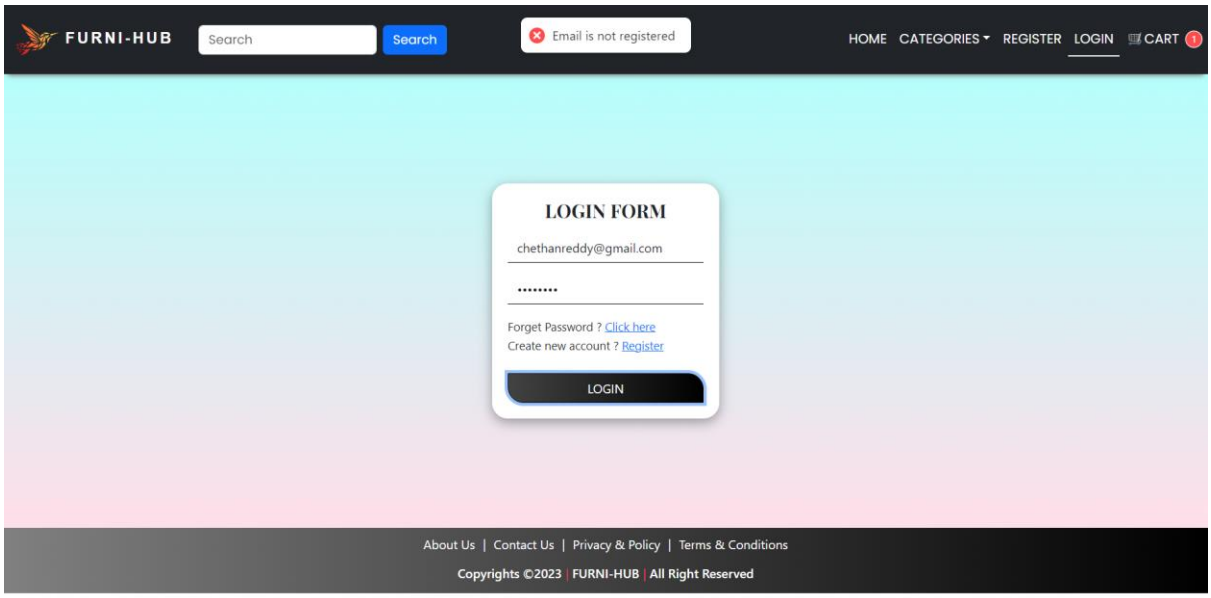

Fig. No. 6.13: Wrong Email

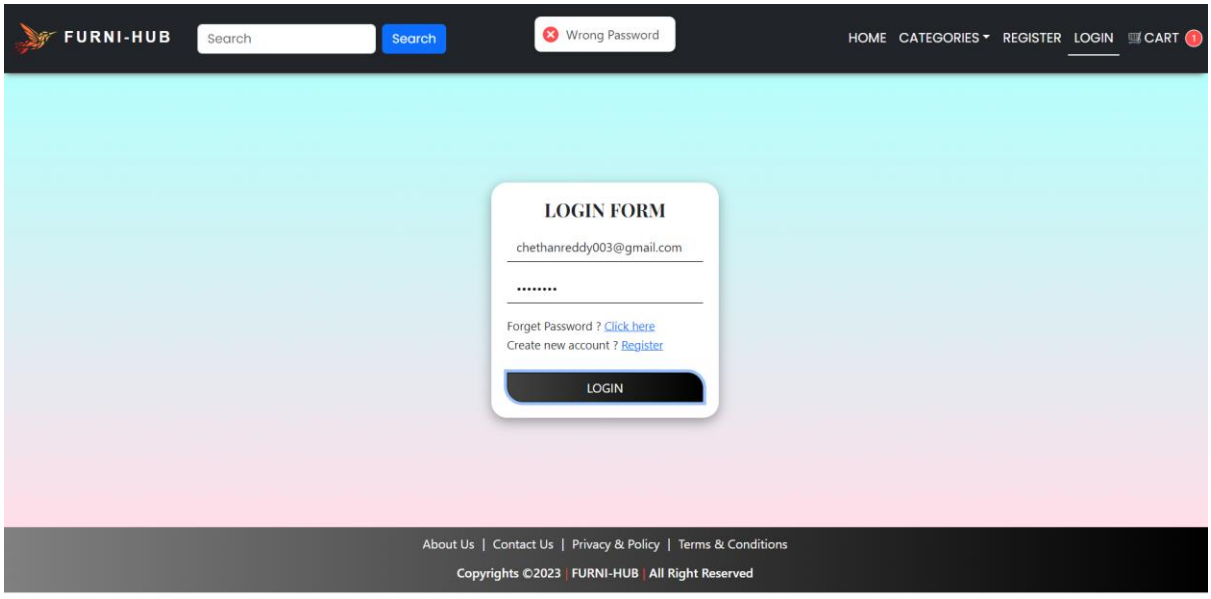

Fig. No. 6.14: Wrong Password

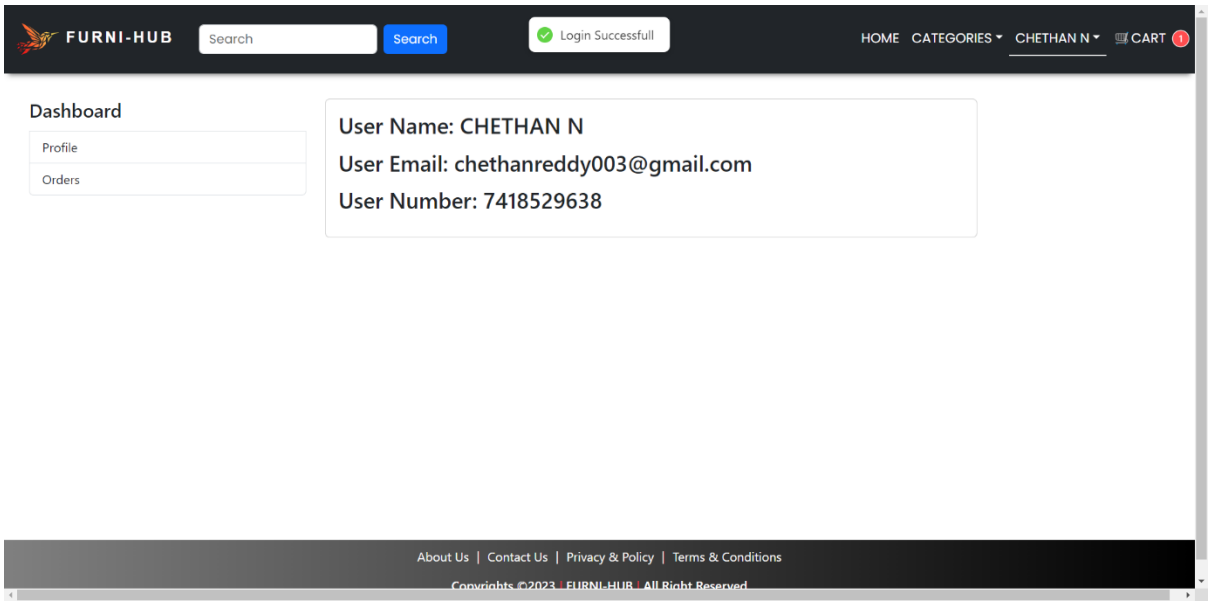

Fig. No. 6.15: User's Login Successful

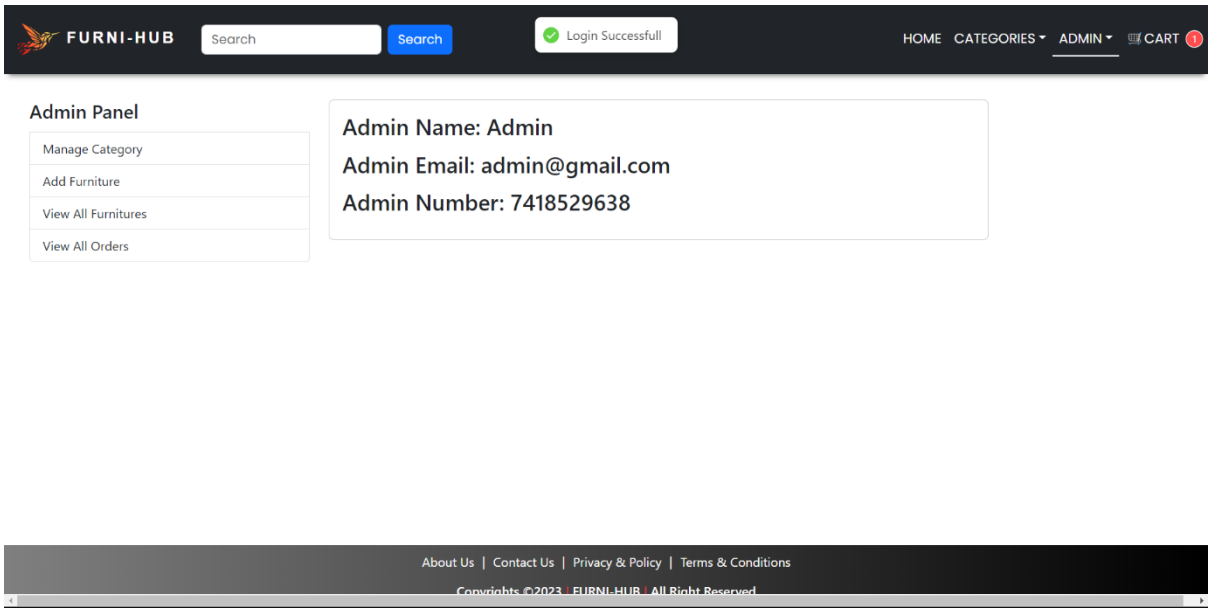

Fig. No. 6.16: Admin's Login Successful

Test Case – 03 (Manage category)

SI NO.	Description	Expected Result	Actual Result	Status
03	Manage Category	All Valid Details	Empty Name	Fail
		All Valid Details	All Valid Details	Pass
		Deleted category	Deleted category	Pass

Table No. 6.3

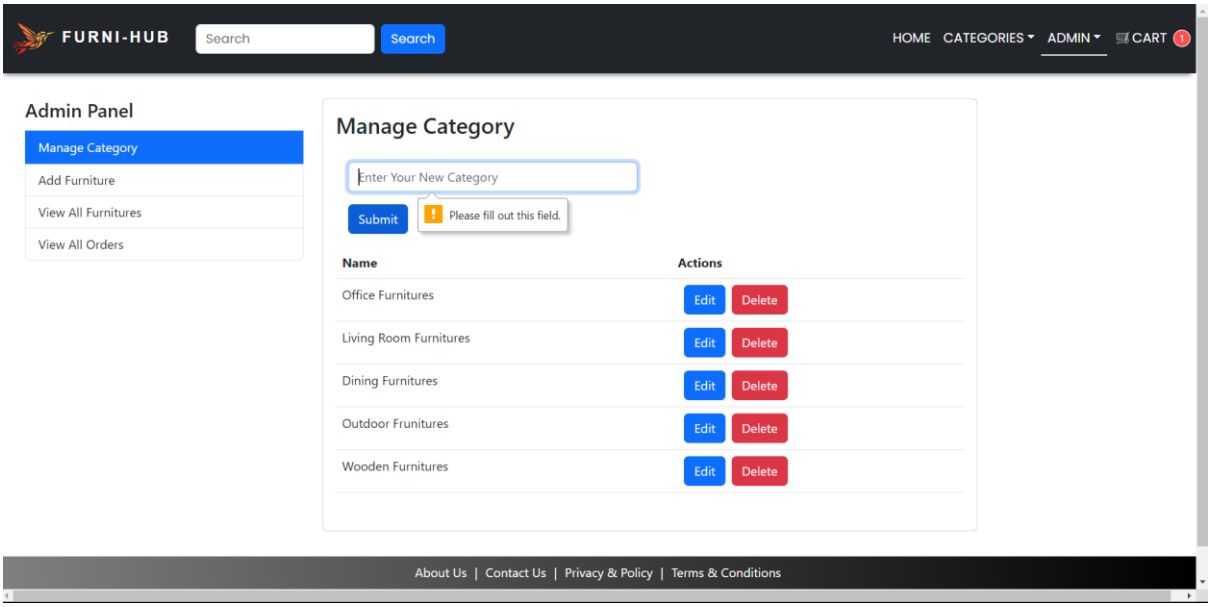

Fig. No. 6.17: Empty Category Name

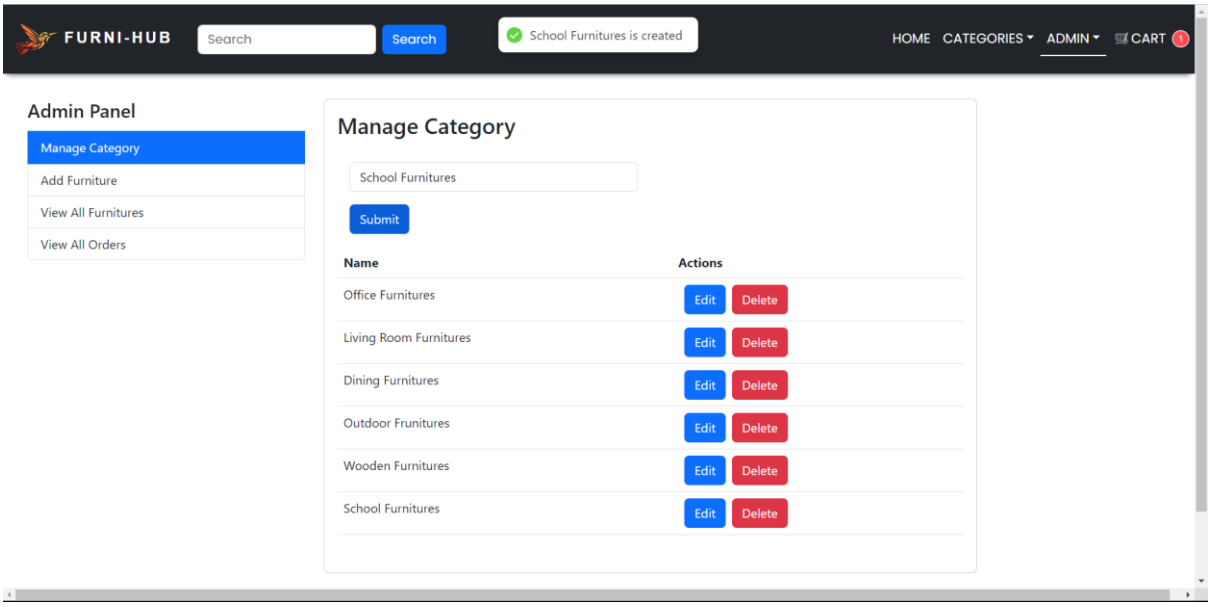

Fig. No. 6.18: Category Created Successful

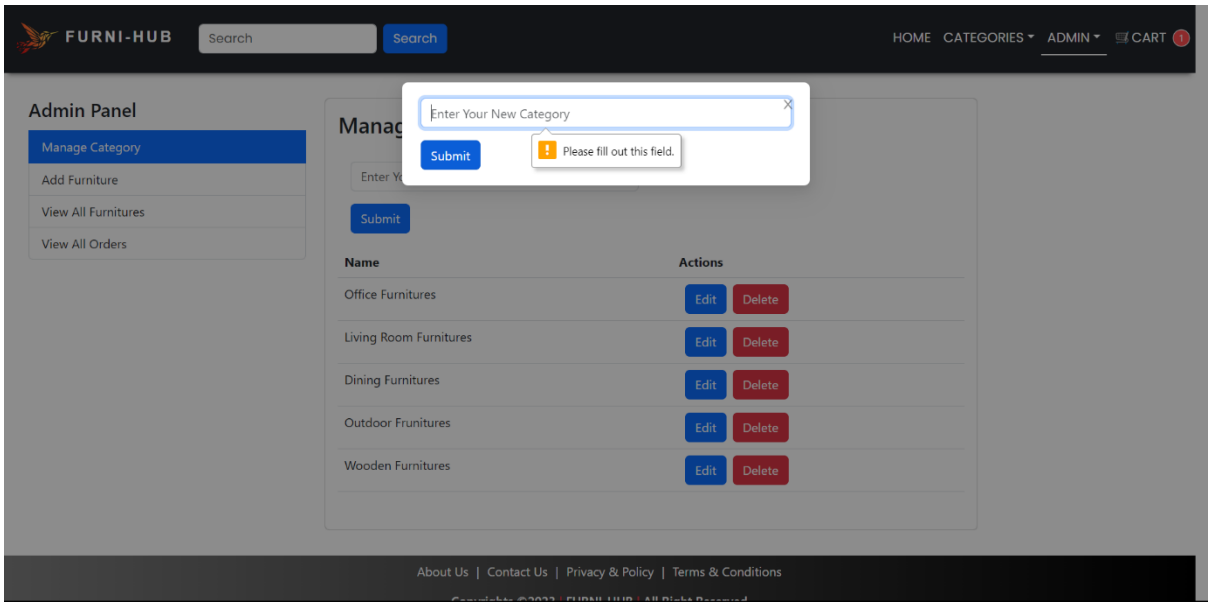

Fig. No. 6.19: Edit Empty Name

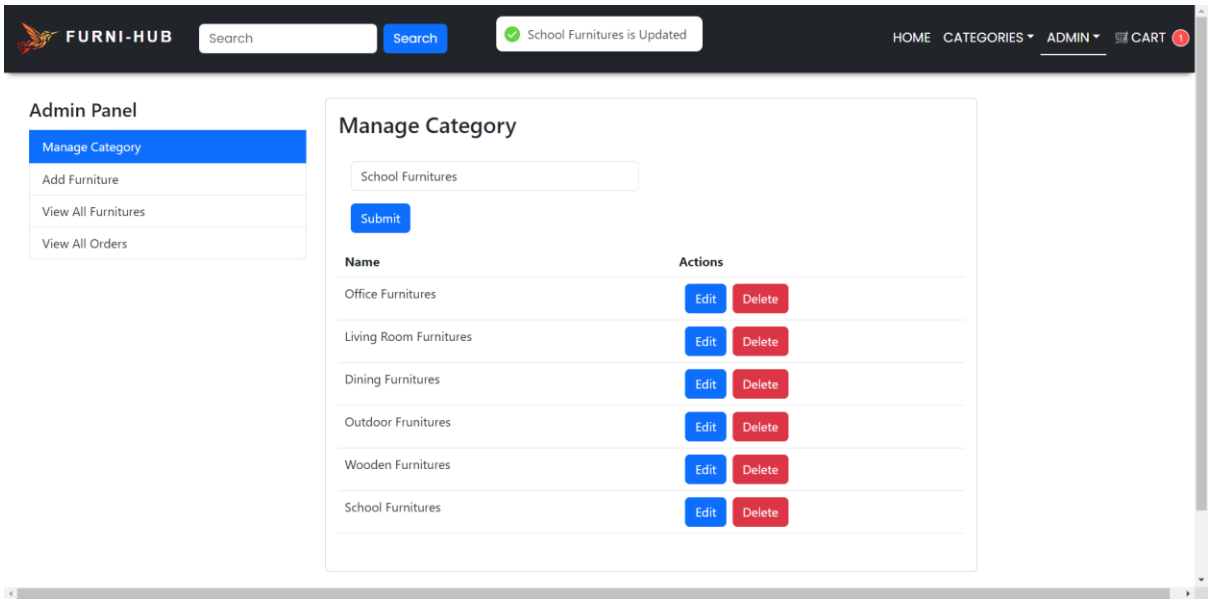

Fig. No. 6.20: Category Updated Successful

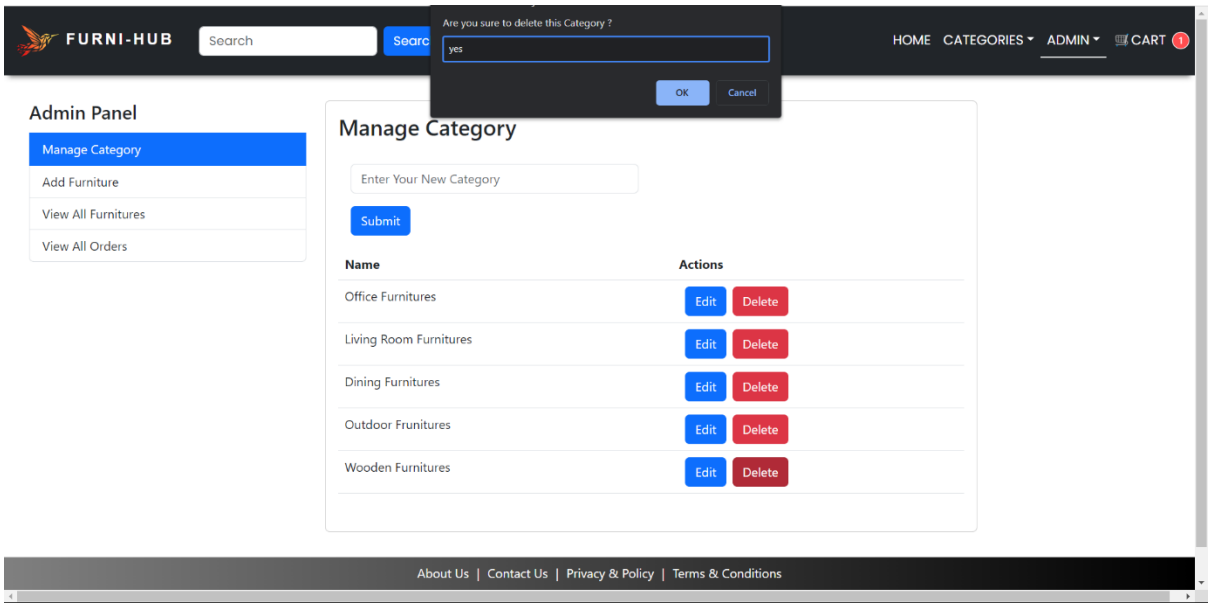

Fig. No. 6.21: Delete Category Confirmation

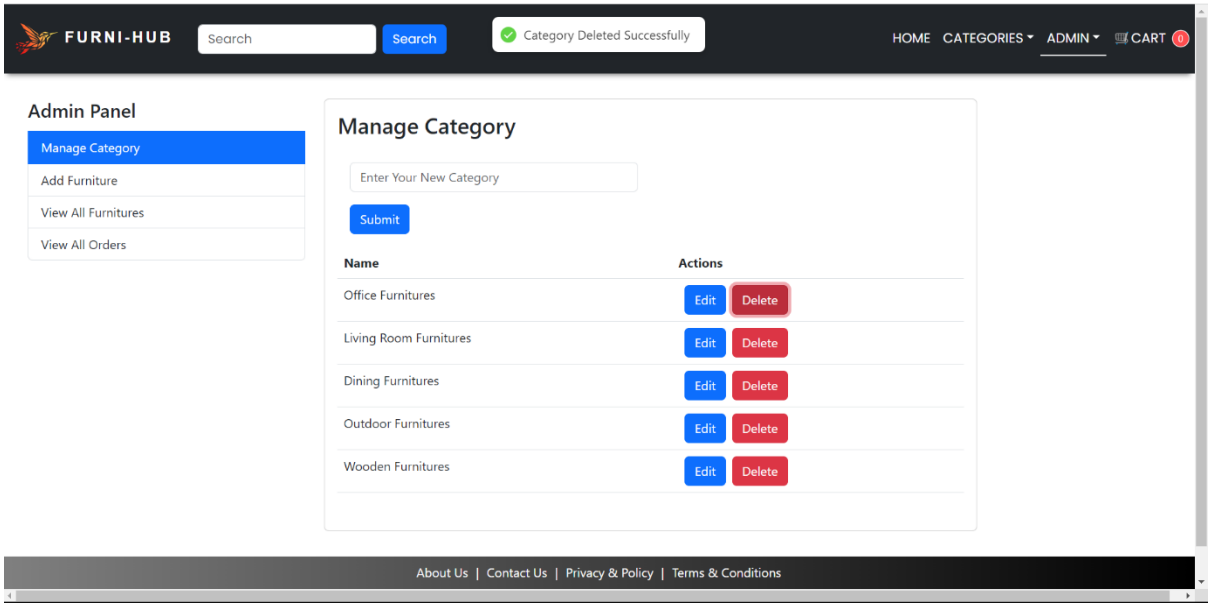

Fig. No. 6.22: Category Deleted Successful

Test Case – 04 (Add Furniture)

SI NO.	Description	Expected Result	Actual Result	Status
04	Add Furniture	All Valid Details	Empty Category	Fail
		All Valid Details	Empty Name	Fail
		All Valid Details	Empty Photo	Fail
		All Valid Details	Empty Description	Fail
		All Valid Details	Empty Price	Fail
		All Valid Details	Empty Quantity	Fail
		All Valid Details	Empty Ratings	Fail
		All Valid Details	Empty Shipping	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.4

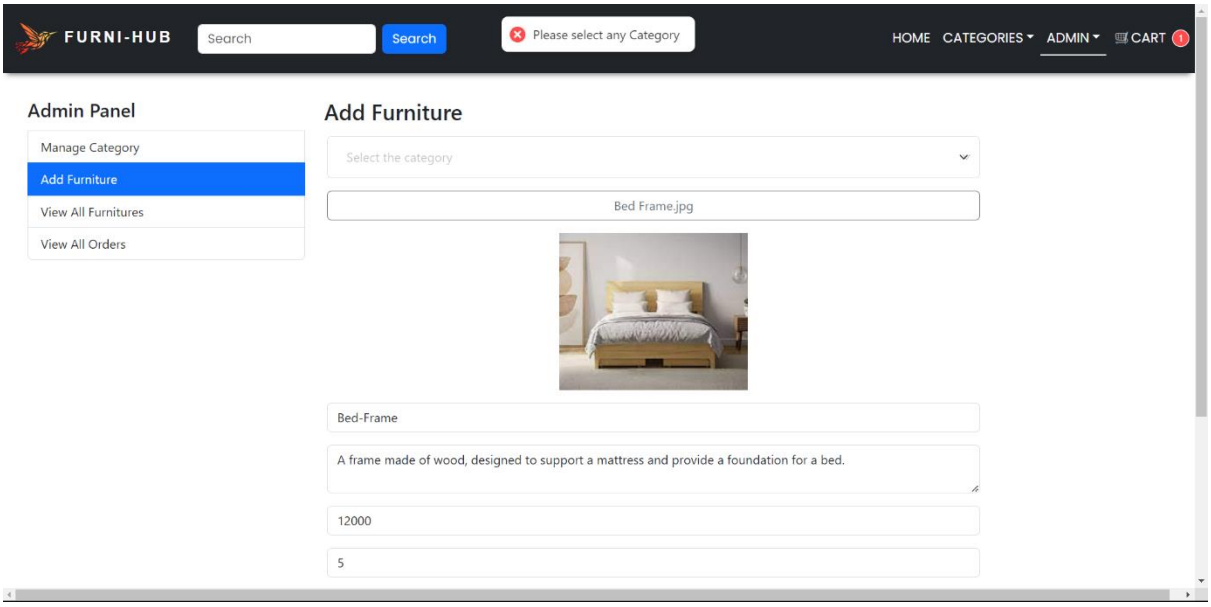

Fig. No. 6.23: Empty category

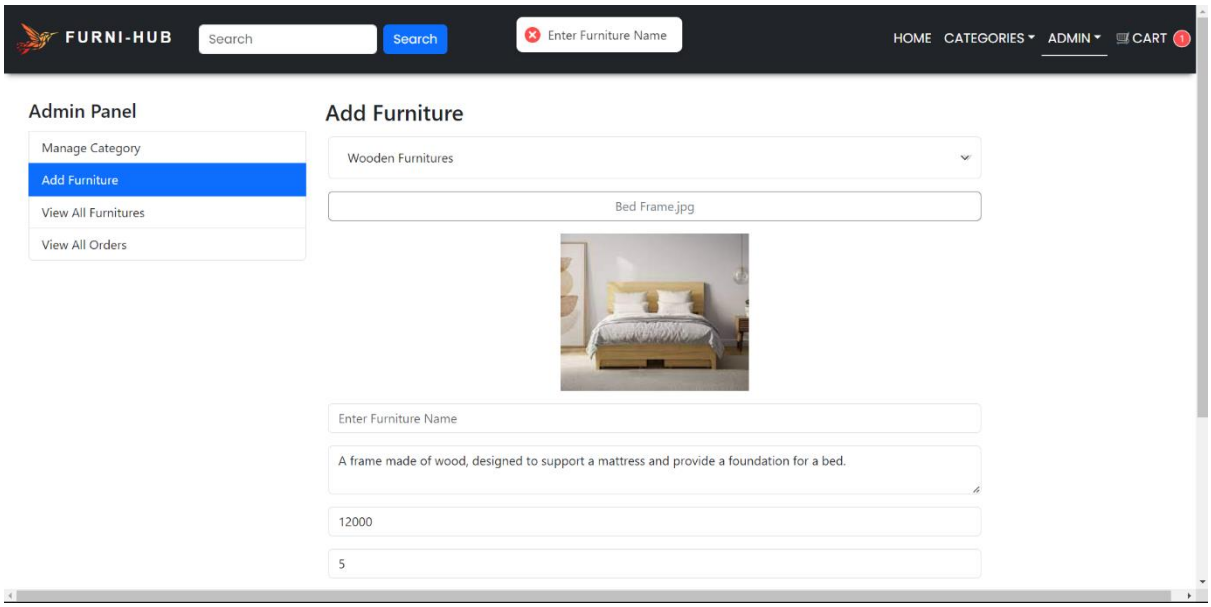

Fig. No. 6.24: Empty Name

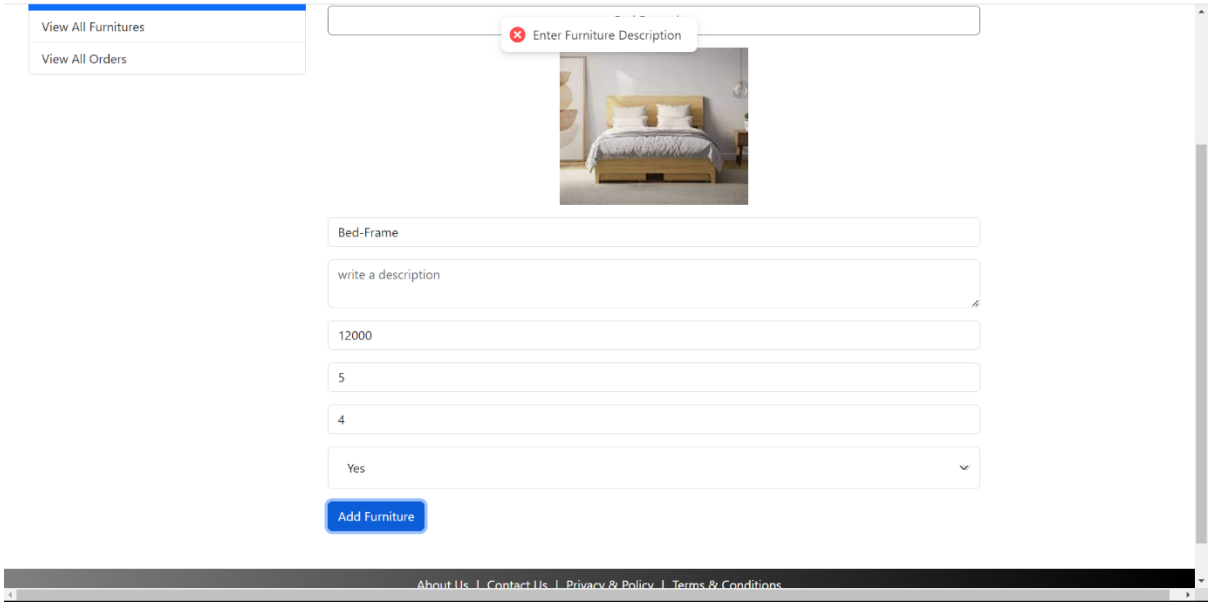

Fig. No. 6.25: Empty Description

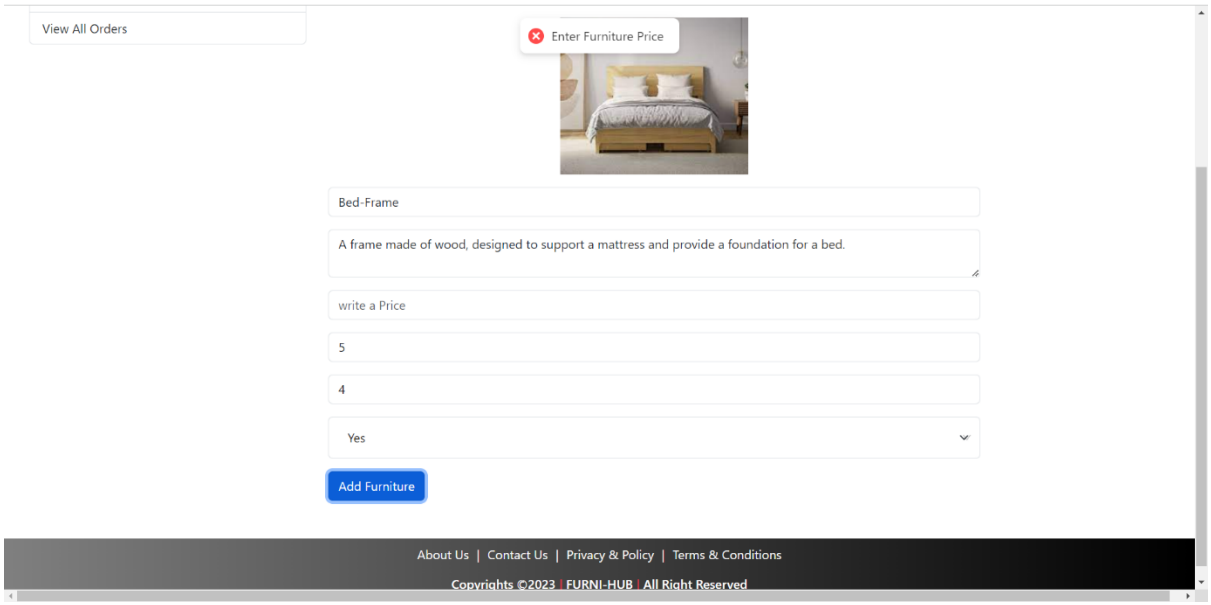

Fig. No. 6.26: Empty Price

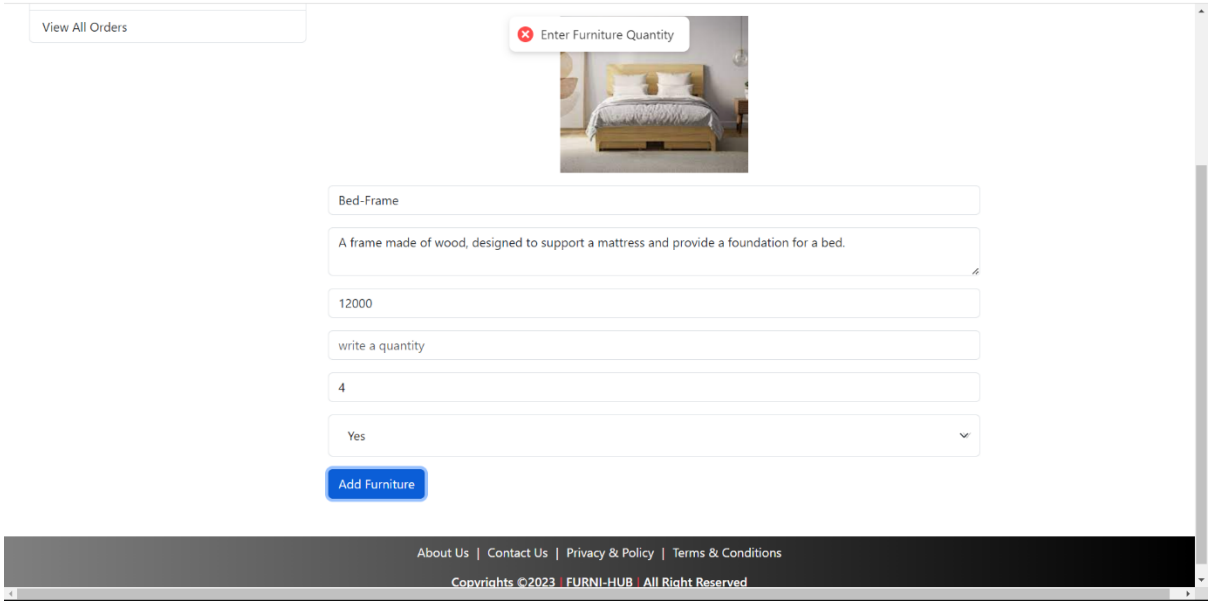

Fig. No. 6.27: Empty Quantity

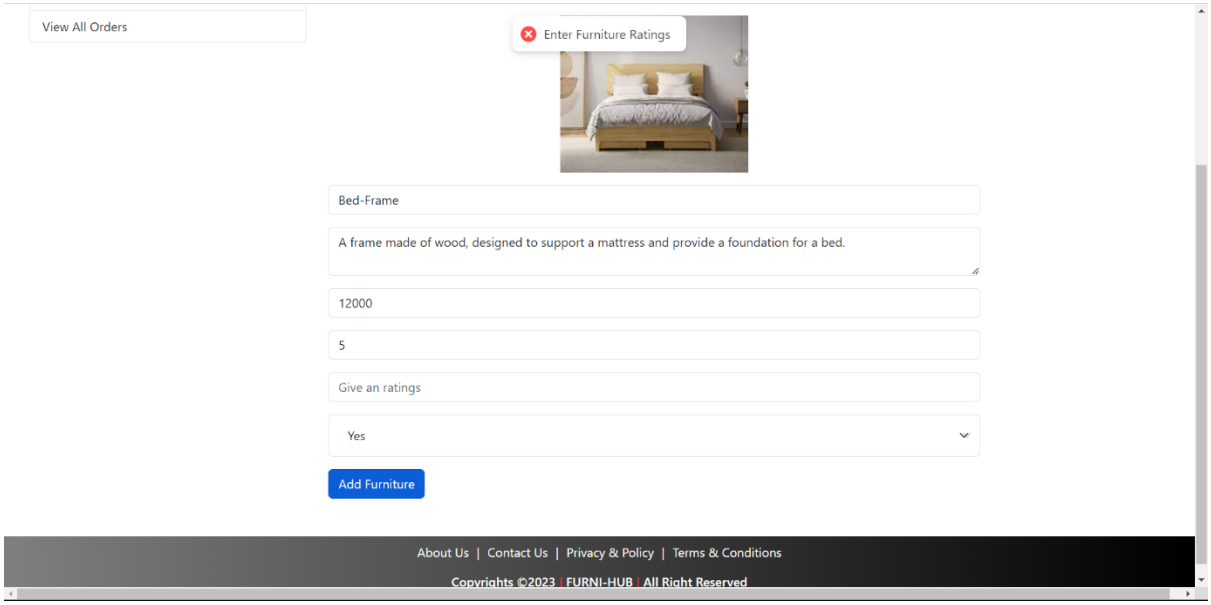

Fig. No. 6.28: Empty Ratings

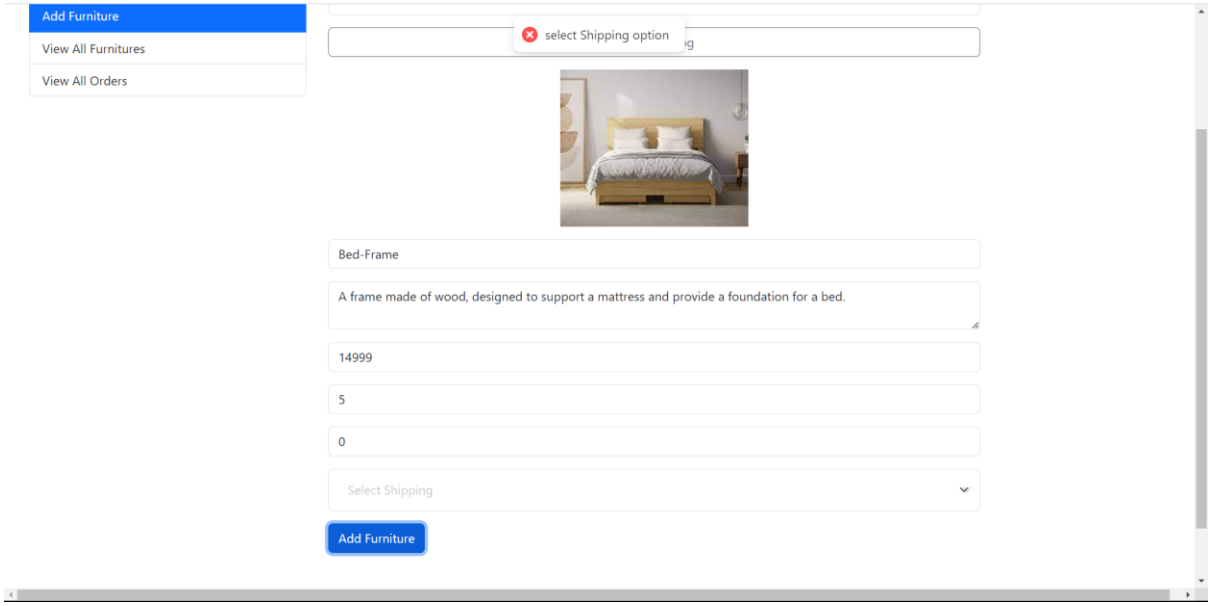

Fig. No. 6.29: Empty Shipping

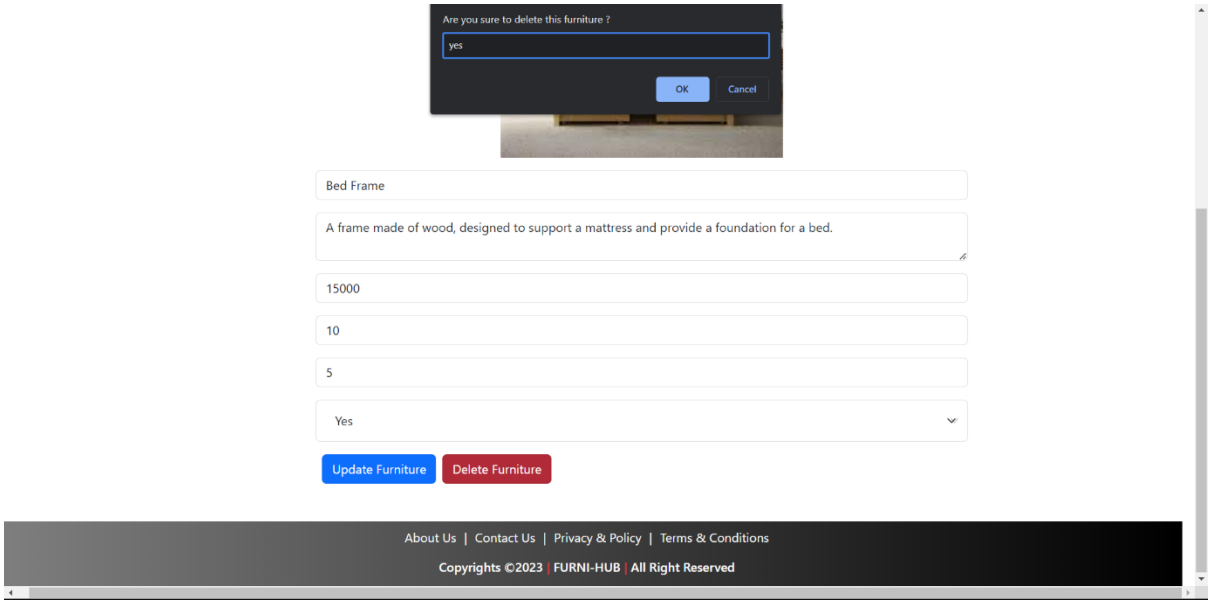

Fig. No. 6.30: Furniture deletion Conformation

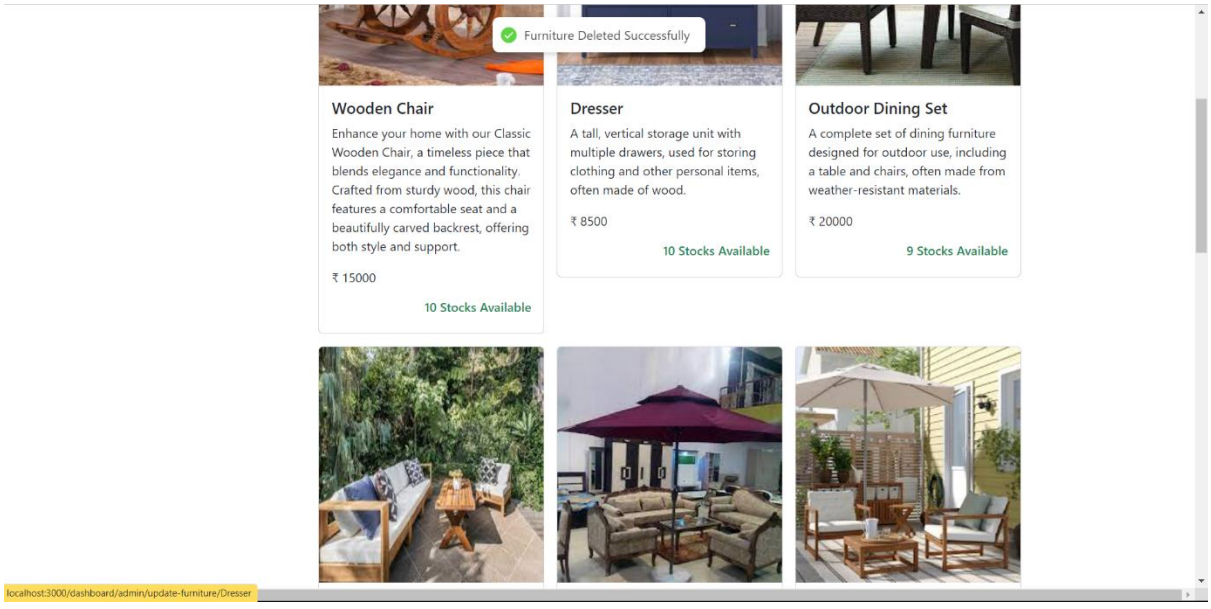

Fig. No. 6.31: Furniture Deleted Successful

Test Case – 05 (Update Furniture)

SI NO.	Description	Expected Result	Actual Result	Status
05	Update Furniture	All Valid Details	Empty Category	Fail
		All Valid Details	Empty Name	Fail
		All Valid Details	Empty Photo	Fail
		All Valid Details	Empty Description	Fail
		All Valid Details	Empty Price	Fail
		All Valid Details	Empty Quantity	Fail
		All Valid Details	Empty Ratings	Fail
		All Valid Details	Empty Shipping	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.5

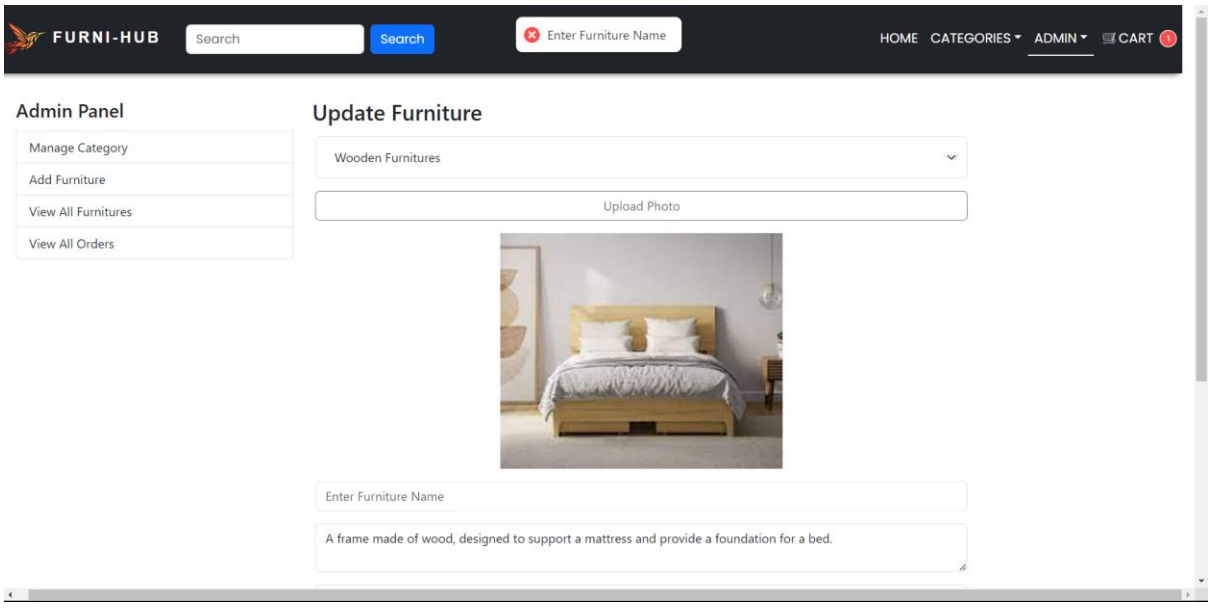

Fig. No. 6.32: Empty Name

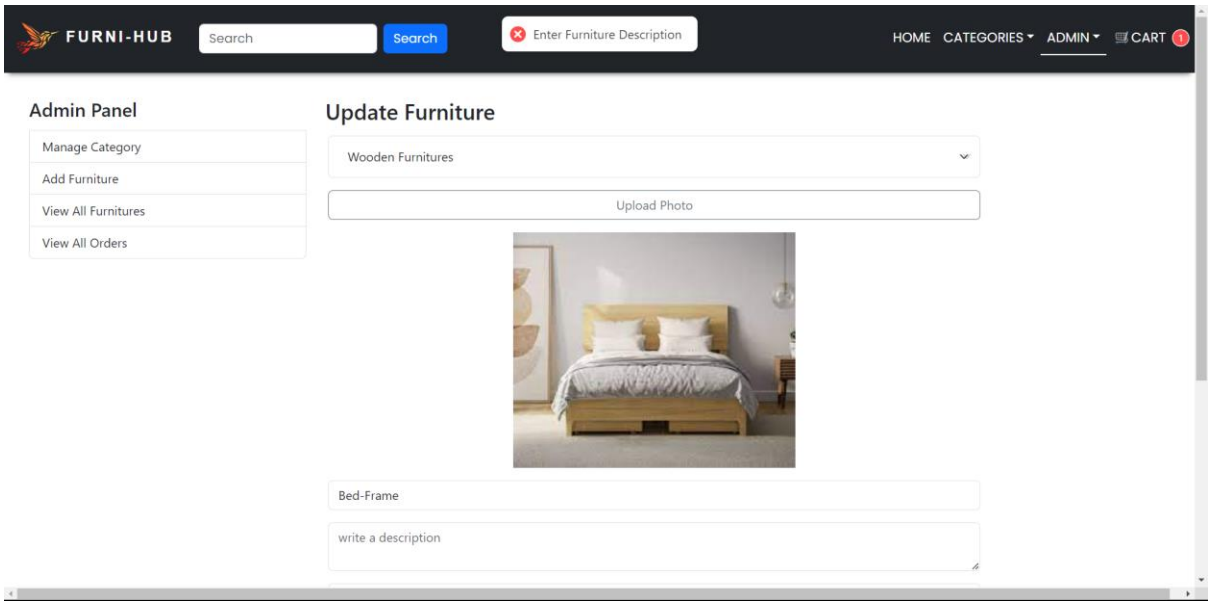

Fig. No. 6.33: Empty Description

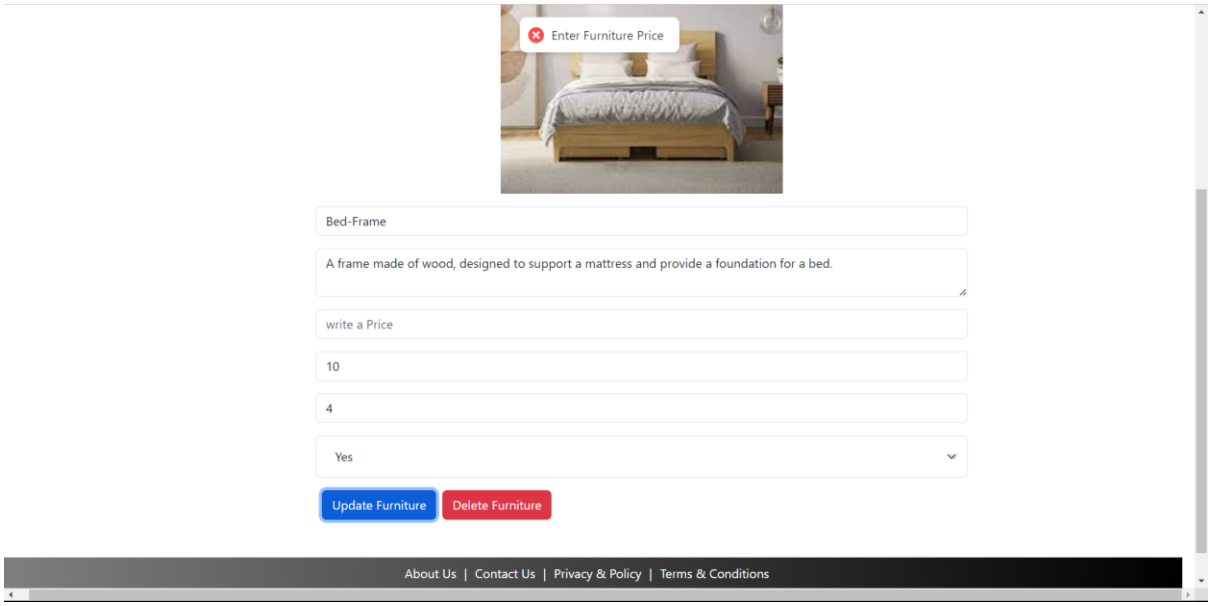

Fig. No. 6.34: Empty Price

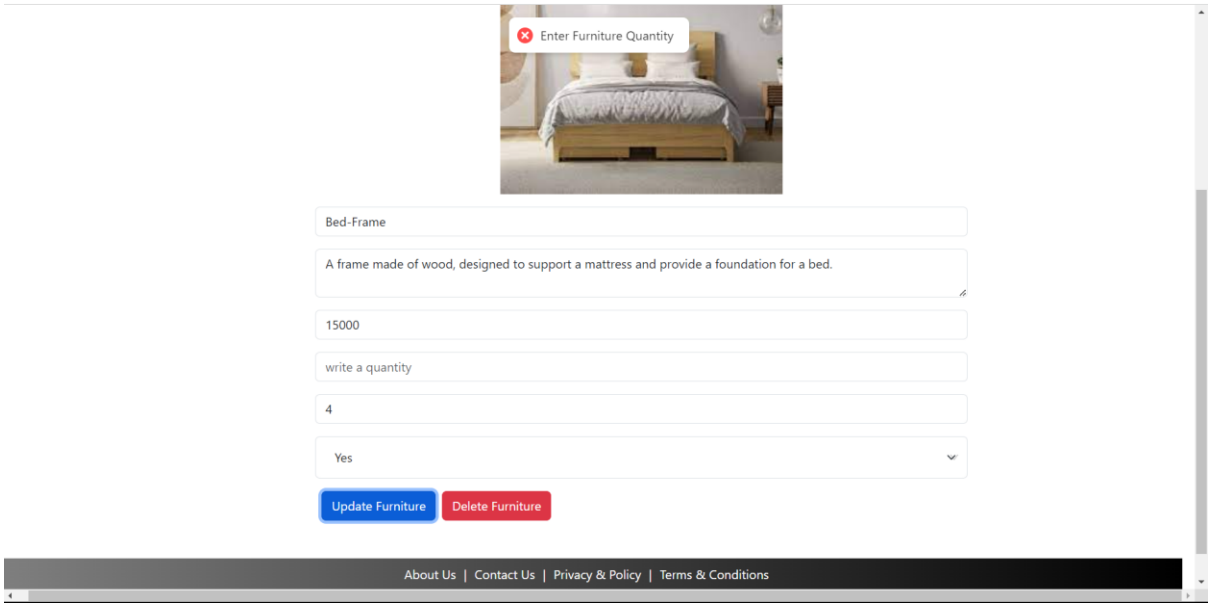

Fig. No. 6.35: Empty Quantity

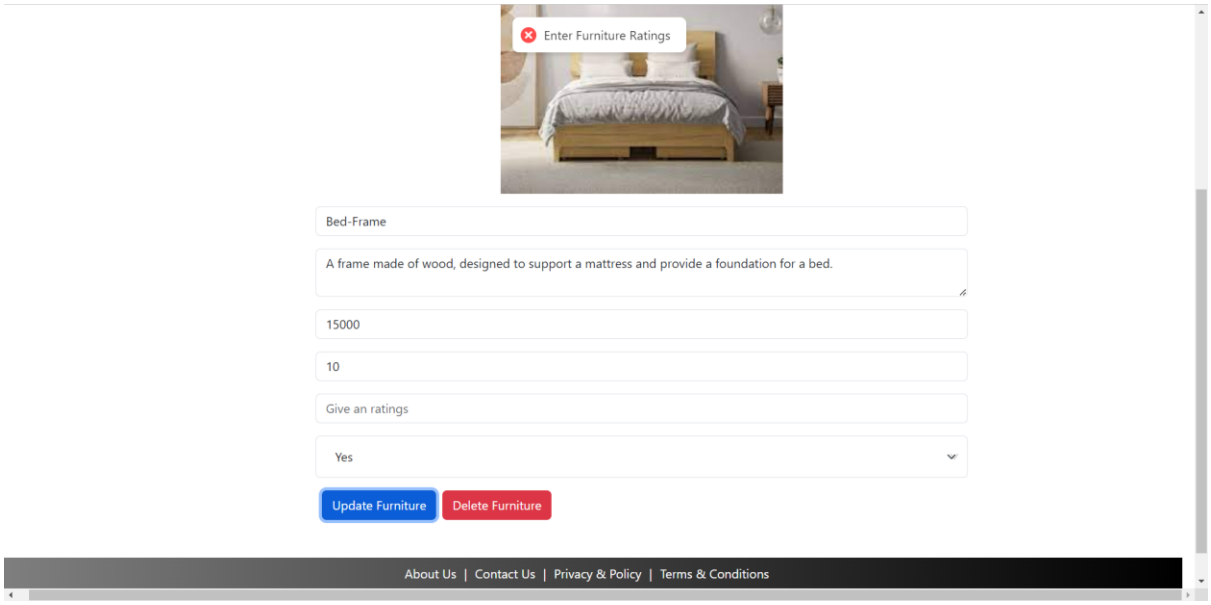

Fig. No. 6.36: Empty Ratings

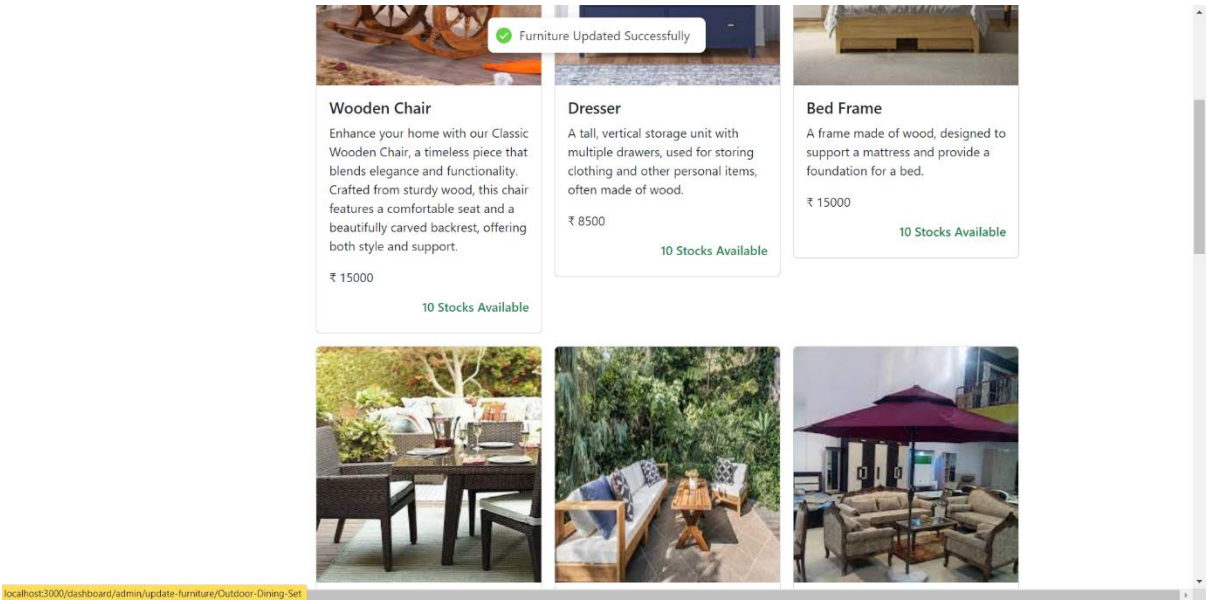

Fig. No. 6.37: Furniture Updated

Test Case – 06 (Checkout)

SI NO.	Description	Expected Result	Actual Result	Status
06	Checkout	All Valid Details	Without Login	Fail
		All Valid Details	Invalid Details	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.6

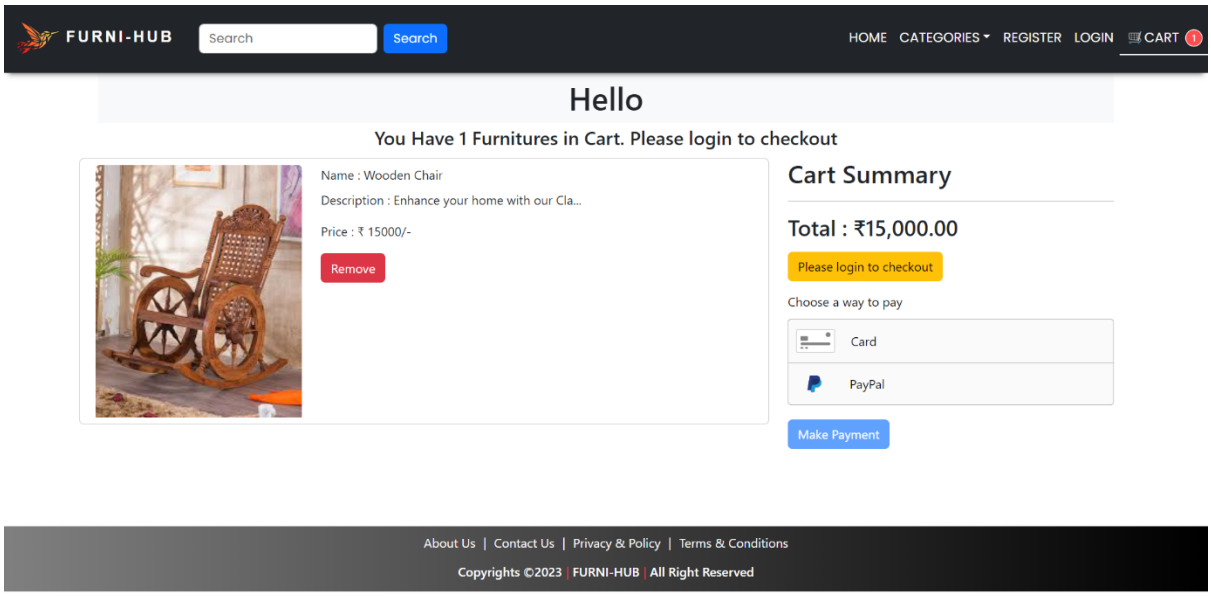

Fig. No. 6.38: Checkout without login

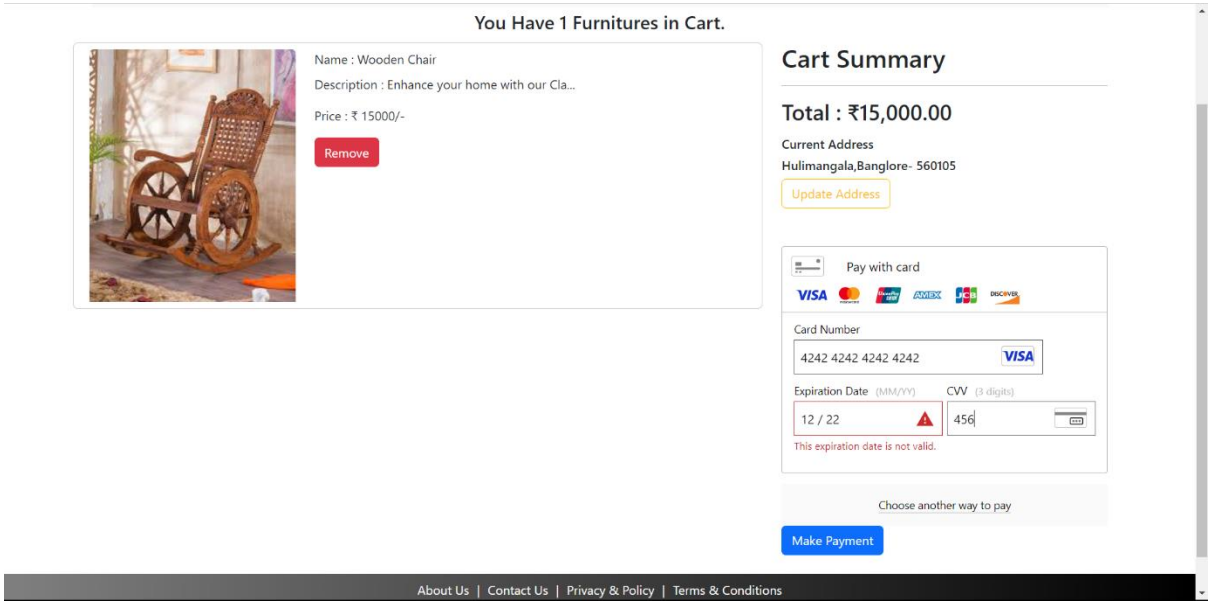

Fig. No. 6.39: Invalid Card Details

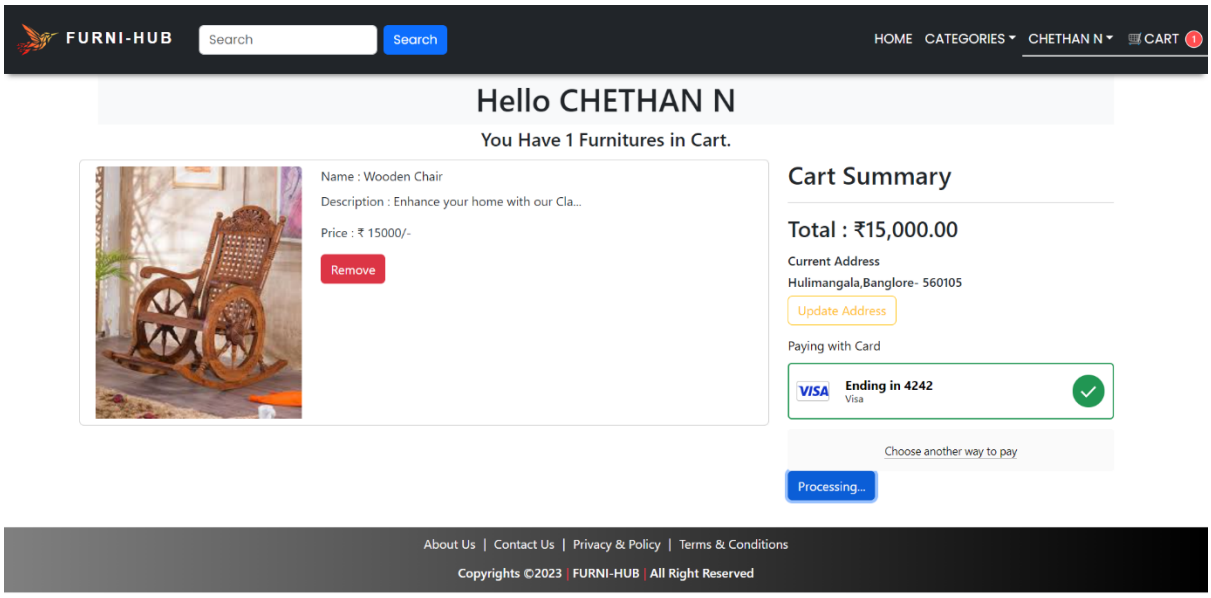

Fig. No. 6.40: Order Successful

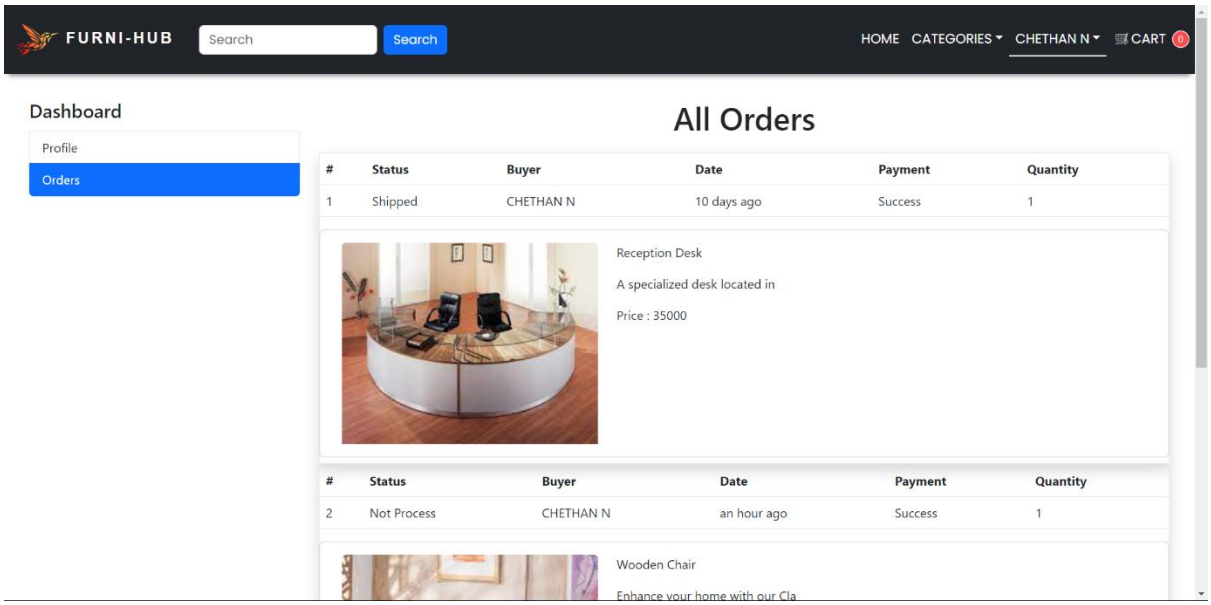

Fig. No. 6.41: User's Recent Order

Test Case – 07 (Admin's Manage Order)

SI NO.	Description	Expected Result	Actual Result	Status
07	Admin's Manage Order	All Valid Details	Invalid Details	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.7

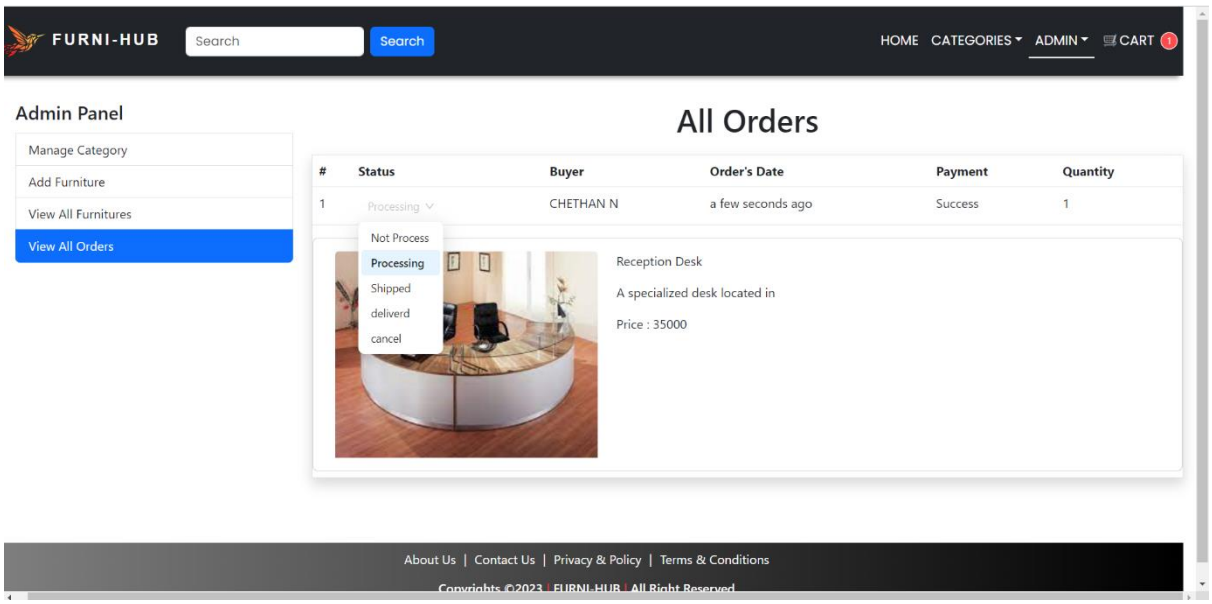

Fig. No. 6.42: Managing Order Status

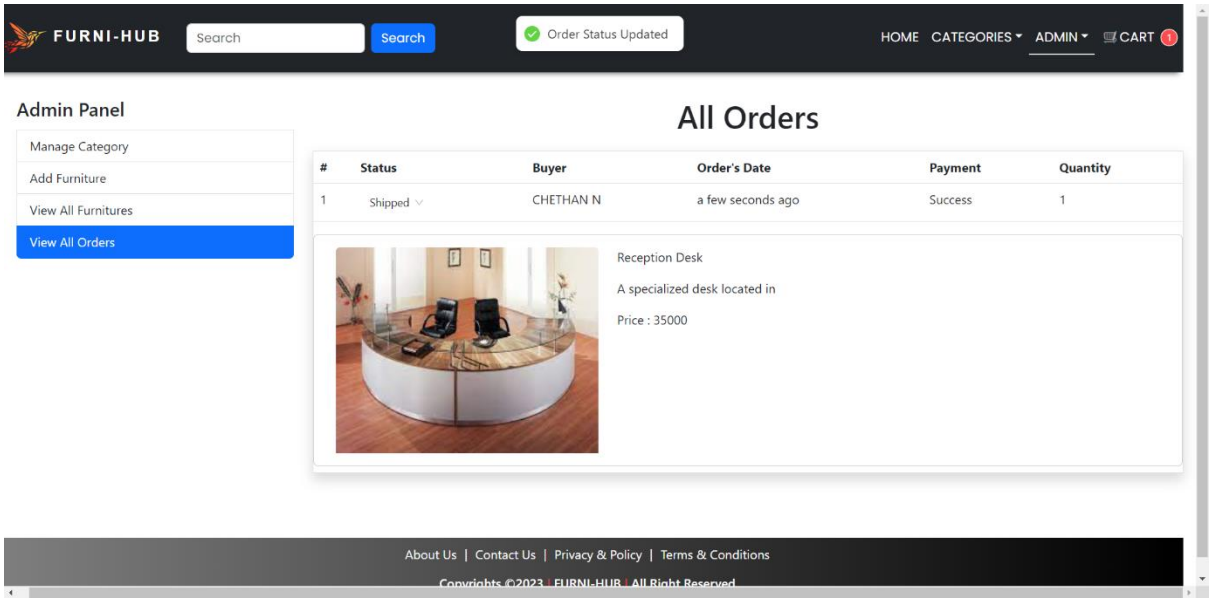

Fig. No. 6.43: Order status update successful

7. CONLUSION

In conclusion, the MERN Full stack project for the Online furniture shop provides a seamless user experience by allowing users to browse and add items to their cart without the need for registration. By implementing the account creation and registration process during checkout, the project ensures a secure and convenient payment process for users.

The ability for users to login and check their order status enhances the overall user experience, providing transparency and allowing users to stay informed about their purchases. Furthermore, the option to update their profile information such as name, address, contact number, and email address adds a personalized touch and flexibility for users.

On the admin side, the project offers comprehensive management capabilities for furniture categories, individual furniture items, and orders. This allows the admin to efficiently organize and maintain the online furniture shop, ensuring smooth operations.

8. FUTURE ENHANCEMENT

For future enhancements, several possibilities exist to further improve the project. Here are a few suggestions:

- **Implement a search functionality:** Allow users to search for specific furniture items based on keywords, categories, or other relevant criteria. This would enhance user convenience and help them find desired products more quickly.
- **Integrate social media sharing:** Enable users to share their favorite furniture items or completed purchases on social media platforms, expanding the shop's reach and potentially attracting more customers.
- **Offer personalized recommendations:** Utilize machine learning algorithms to provide personalized furniture recommendations to users based on their browsing history, purchase patterns, or preferences. This can help enhance the user experience and increase customer satisfaction.
- **Enable customer reviews and ratings:** Allow users to provide feedback and ratings for purchased furniture items, providing social proof and aiding potential buyers in their decision-making process.
- **Expand payment options:** Integrate additional payment gateways, such as PayPal or digital wallets, to offer users more flexibility and convenience during the checkout process.
- **Enhance the admin dashboard:** Improve the admin interface by providing advanced analytics and reporting features, enabling the admin to gain insights into sales performance, popular products, and customer behavior.

By incorporating these future enhancements, the Online furniture shop MERN Full stack project can continue to evolve and provide an even better user experience while empowering the admin to efficiently manage the shop's operations.

9. REFERENCES

1) https://code.visualstudio.com/download
2) https://github.com/techinfo-youtube/ecommerce-app-2023/tree/15-admin-orders-css
3) https://legacy.reactjs.org/docs/getting-started.html
4) https://nodejs.dev/en/
5) https://nodejs.org/en/docs
6) https://npmjs.com/
7) https://youtube.com/playlist?list=PLuHGmgpyHfRzhGkSUfY0vpi67X64g0mXB