

Daily Status Report

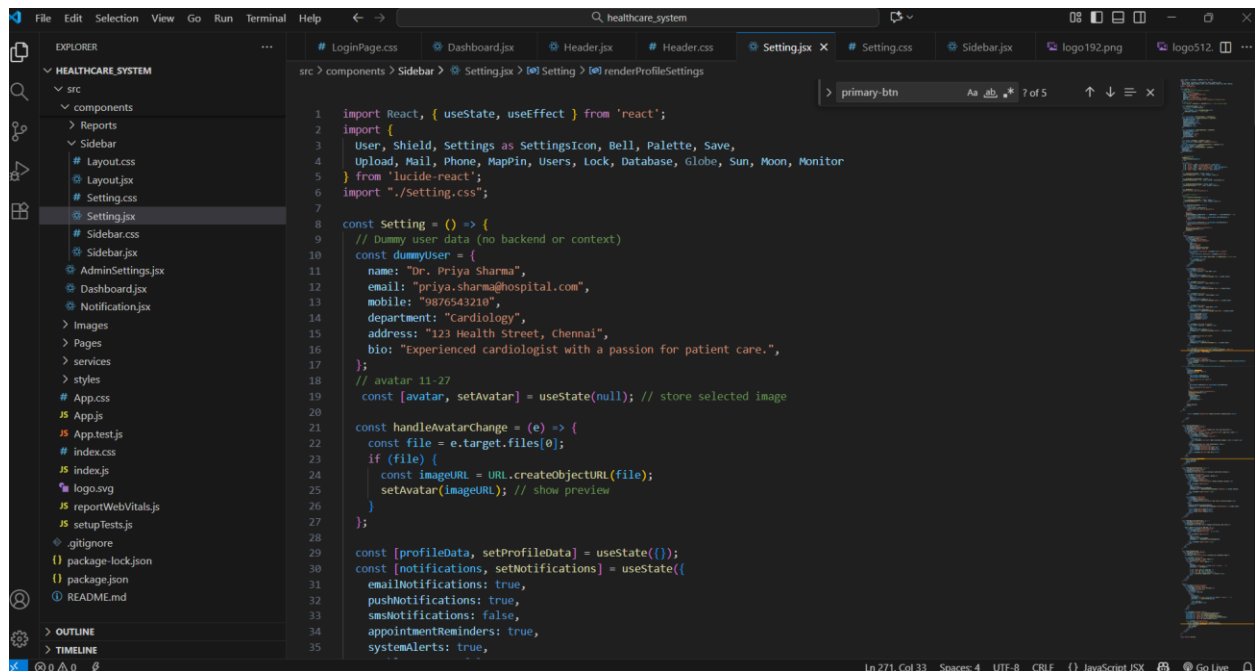
Name : M. Eswari

Date : 03-11-2025

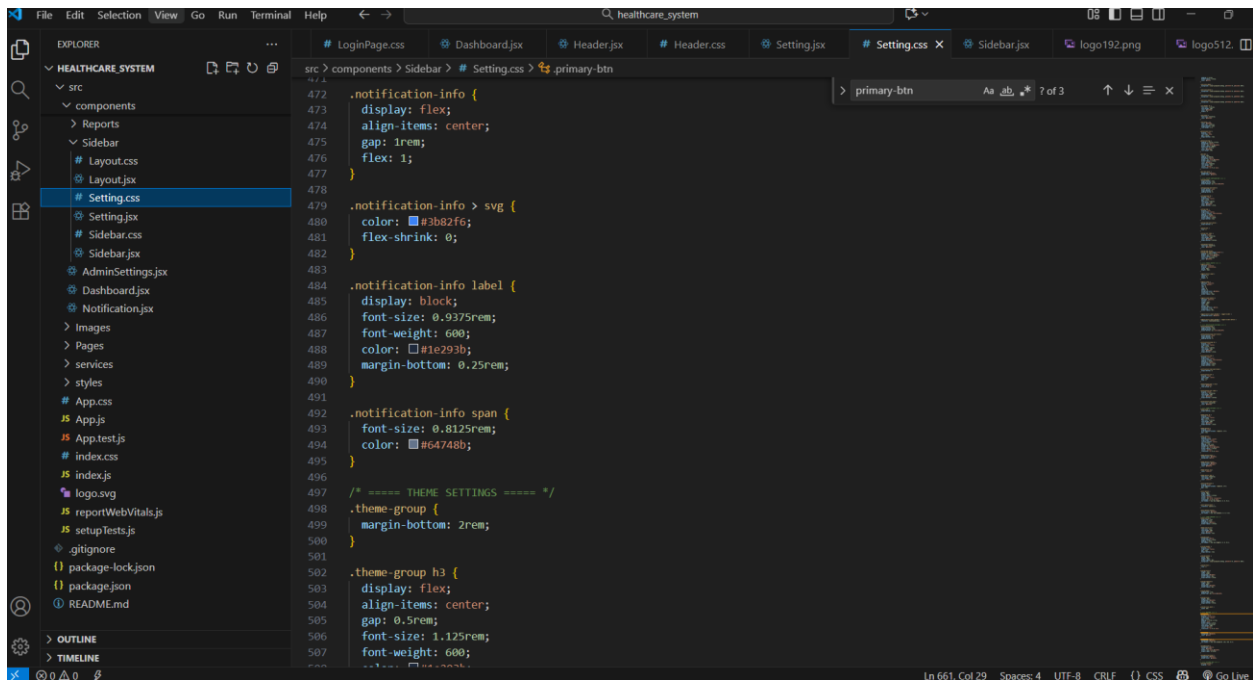
The **Settings page** lets users update personal details, preferences, and security options. The **FPS page** manages Fair Price Shop details like address, geo-location, and photos. Both pages ensure smooth data handling and user-friendly interaction within the React app.

Healthcare project

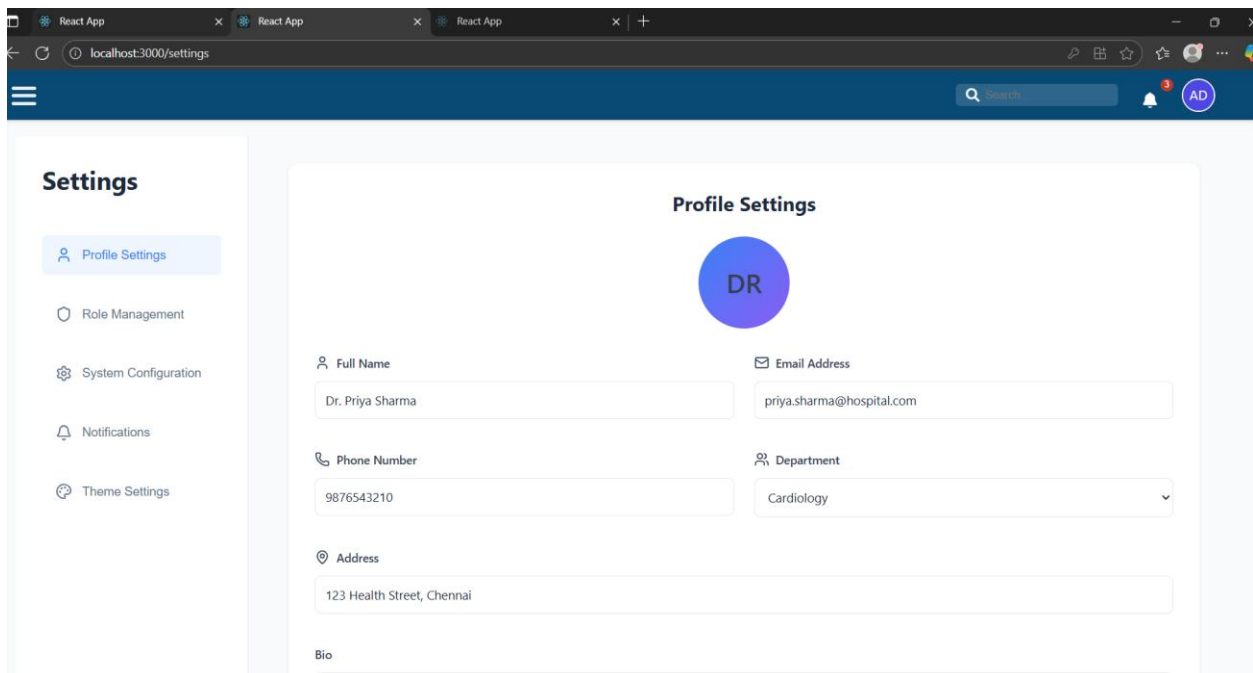
The **Save Changes** button is now placed inside the form on the right side. Clicking **Change Password** opens a popup asking for the old password first. After submitting, it asks for the new and confirm passwords. Once updated, a popup appears showing “**Password updated successfully**”



```
1  import React, { useState, useEffect } from 'react';
2  import {
3    User, Shield, Settings as Settingsicon, Bell, Palette, Save,
4    Upload, Mail, Phone, MapPin, Users, Lock, Database, Globe, Sun, Moon, Monitor
5  } from 'lucide-react';
6  import './Setting.css';
7
8  const Setting = () => {
9    // Dummy user data (no backend or context)
10   const dummyUser = {
11     name: "Dr. Priya Sharma",
12     email: "priya.sharma@hospital.com",
13     mobile: "9876543210",
14     department: "Cardiology",
15     address: "123 Health Street, Chennai",
16     bio: "Experienced cardiologist with a passion for patient care.",
17   };
18   // avatar 11-27
19   const [avatar, setAvatar] = useState(null); // store selected image
20
21   const handleAvatarChange = (e) => {
22     const file = e.target.files[0];
23     if (file) {
24       const imageURL = URL.createObjectURL(file);
25       setAvatar(imageURL); // show preview
26     }
27   };
28
29   const [profileData, setProfileData] = useState({});
30   const [notifications, setNotifications] = useState({
31     emailNotifications: true,
32     pushNotifications: true,
33     smsNotifications: false,
34     appointmentReminders: true,
35     systemAlerts: true,
```

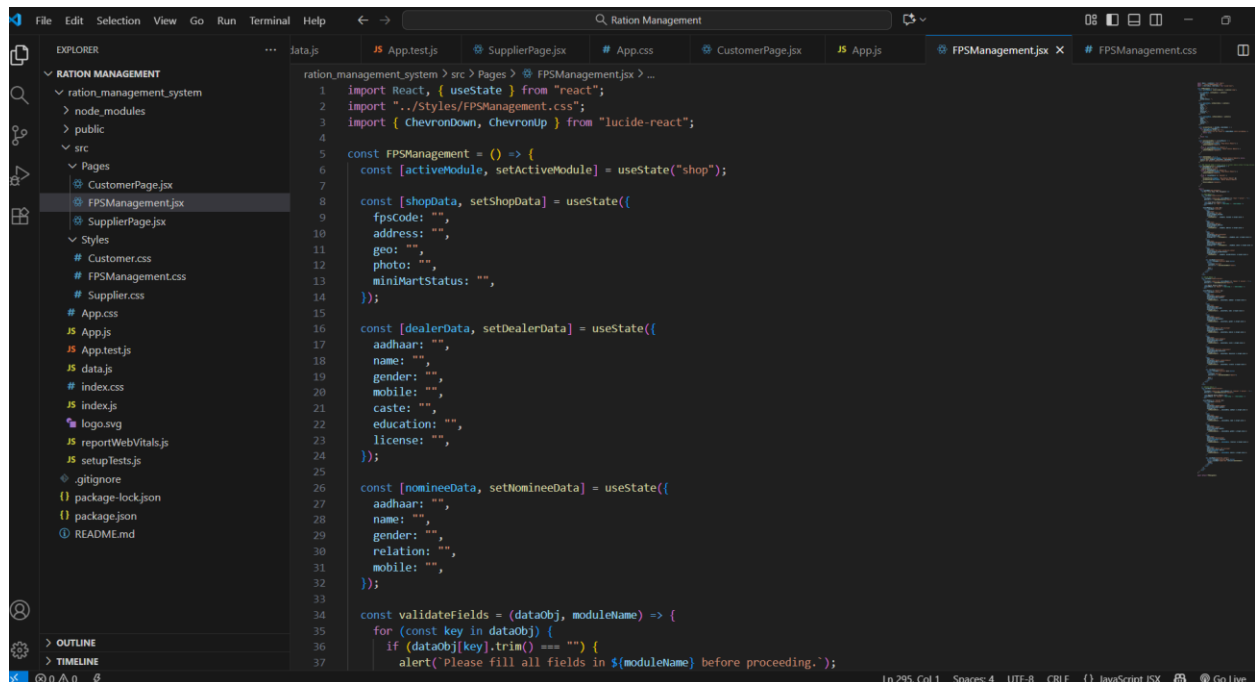


The form now has a right-aligned **Save Changes** button, and the **Change Password** option opens a two-step popup for updating passwords. A success popup confirms when the password is updated successfully.



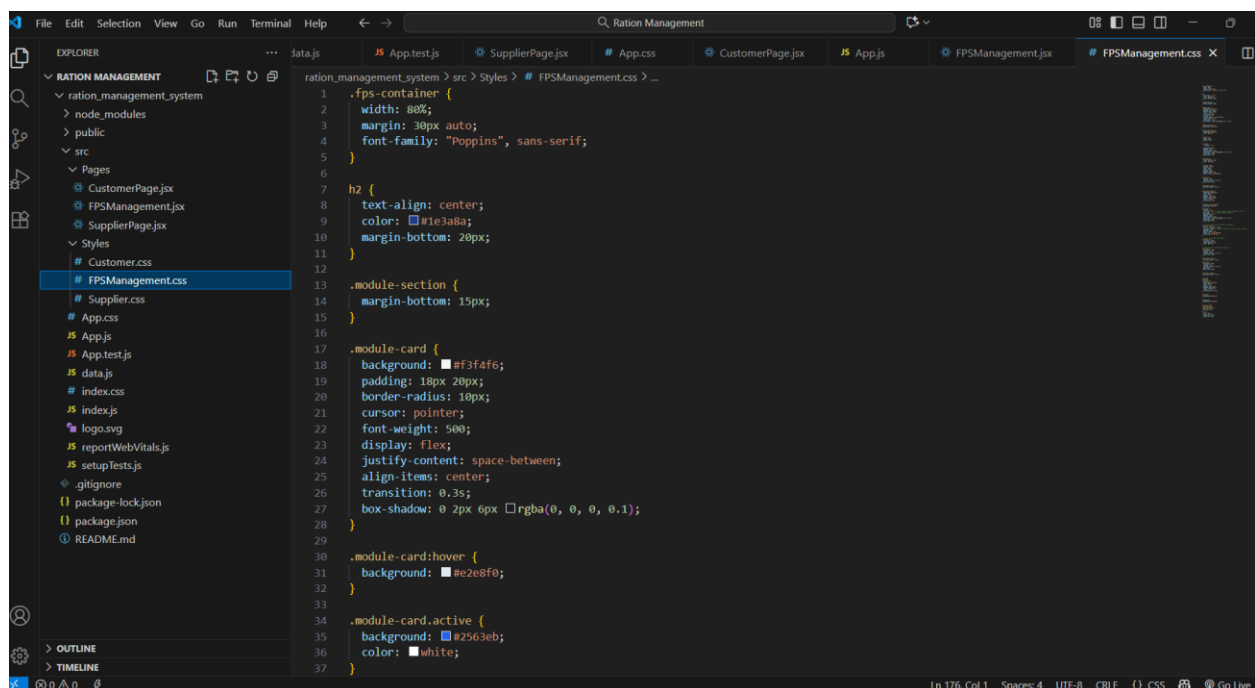
Ration Management System:

This update adds inline error handling for all FPS Management form fields. Each input now shows a specific red error message below it when left blank. The validateFields function tracks missing fields and sets error messages. Errors automatically disappear when the user starts typing in the field. The layout, logic, and class names remain unchanged while improving form validation



The screenshot shows the VS Code editor with the 'Ration Management' project open. The Explorer panel on the left shows the file structure, with 'FPSManagement.jsx' selected under the 'Pages' folder. The main editor displays the code for 'FPSManagement.jsx'. The code includes imports for React, useState, and CSS modules. It defines state for 'shopData', 'dealerData', and 'nomineeData'. A 'validateFields' function is implemented, which iterates through the data object and alerts the user if any field is empty. The status bar at the bottom indicates 'Ln 295, Col 1'.

```
1 import React, { useState } from "react";
2 import "../Styles/FPSManagement.css";
3 import { ChevronDown, ChevronUp } from "lucide-react";
4
5 const FPSManagement = () => {
6   const [activeModule, setActiveModule] = useState("shop");
7
8   const [shopData, setShopData] = useState({
9     fpsCode: "",
10    address: "",
11    geo: "",
12    photo: "",
13    miniMartStatus: "",
14  });
15
16   const [dealerData, setDealerData] = useState({
17     aadhaar: "",
18     name: "",
19     gender: "",
20     mobile: "",
21     caste: "",
22     education: "",
23     license: "",
24  });
25
26   const [nomineeData, setNomineeData] = useState({
27     aadhaar: "",
28     name: "",
29     gender: "",
30     relation: "",
31     mobile: "",
32  });
33
34   const validateFields = (dataObj, moduleName) => {
35     for (const key in dataObj) {
36       if (dataObj[key].trim() === "") {
37         alert("Please fill all fields in ${moduleName} before proceeding.");
```



The screenshot shows the VS Code editor with the 'Ration Management' project open. The Explorer panel on the left shows the file structure, with 'FPSManagement.css' selected under the 'Styles' folder. The main editor displays the CSS code for 'FPSManagement.css'. The code defines styles for the FPS container, module section, module card, and module card hover/active states. The status bar at the bottom indicates 'Ln 176, Col 1'.

```
1 .fps-container {
2   width: 80%;
3   margin: 30px auto;
4   font-family: "Poppins", sans-serif;
5 }
6
7 h2 {
8   text-align: center;
9   color: #1e3a8a;
10  margin-bottom: 20px;
11 }
12
13 .module-section {
14   margin-bottom: 15px;
15 }
16
17 .module-card {
18   background: #f3f4f6;
19   padding: 18px 20px;
20   border-radius: 10px;
21   cursor: pointer;
22   font-weight: 500;
23   display: flex;
24   justify-content: space-between;
25   align-items: center;
26   transition: 0.3s;
27   box-shadow: 0 2px 6px #0000000a;
28 }
29
30 .module-card:hover {
31   background: #e2e8f0;
32 }
33
34 .module-card.active {
35   background: #2563eb;
36   color: white;
37 }
```

The form now displays red error messages below any empty fields, preventing users from moving to the next module until all required inputs are filled. Once the user enters data, the error message automatically disappears, ensuring clear and smooth validation feedback.

localhost:3001

Fair Price Shops (FPS) Management

Shop Details Module

FPS Code

Address

Geo-coordinates

Photograph URL

Mini Mart suitability status

Save

Next →

Dealer Details Module

Nominee Details Module

Fair Price Shops (FPS) Management

Shop Details Module

Dealer Details Module

Aadhaar Number

Name

Gender

Mobile (OTP verified)

Caste Category

Education (Digilocker)

Dealer License Details

Save

Next →

Nominee Details Module