

Daily Status Report

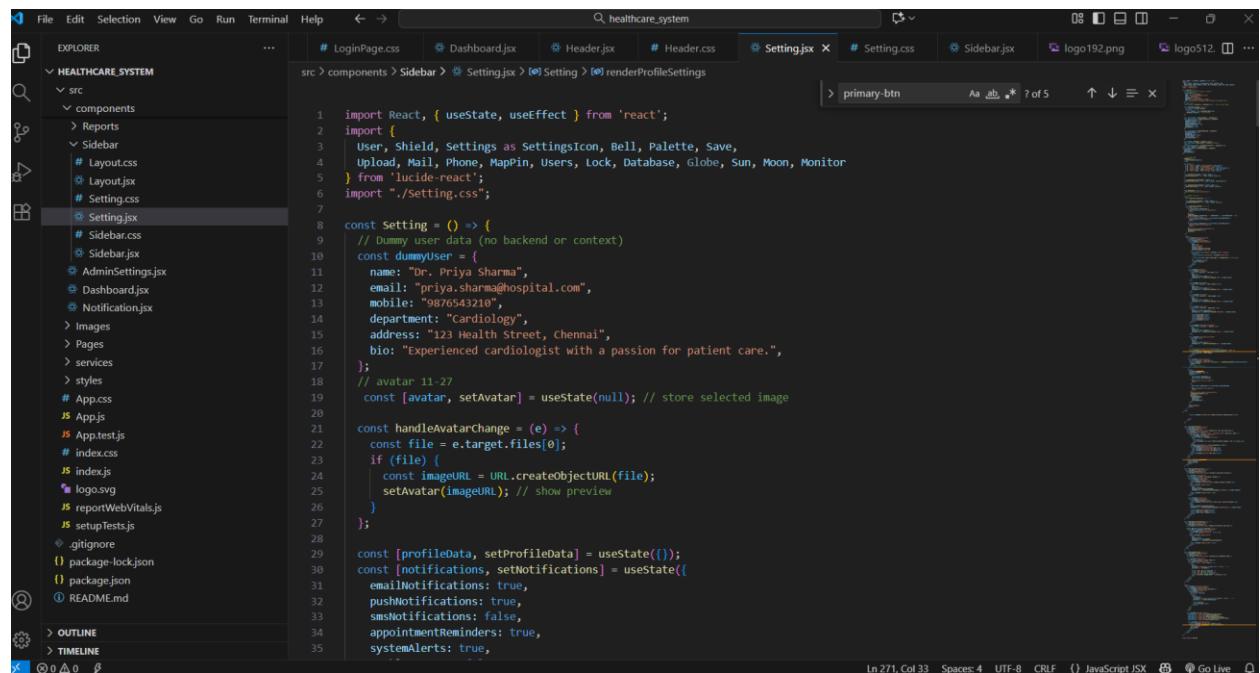
Name : M. Eswari

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**

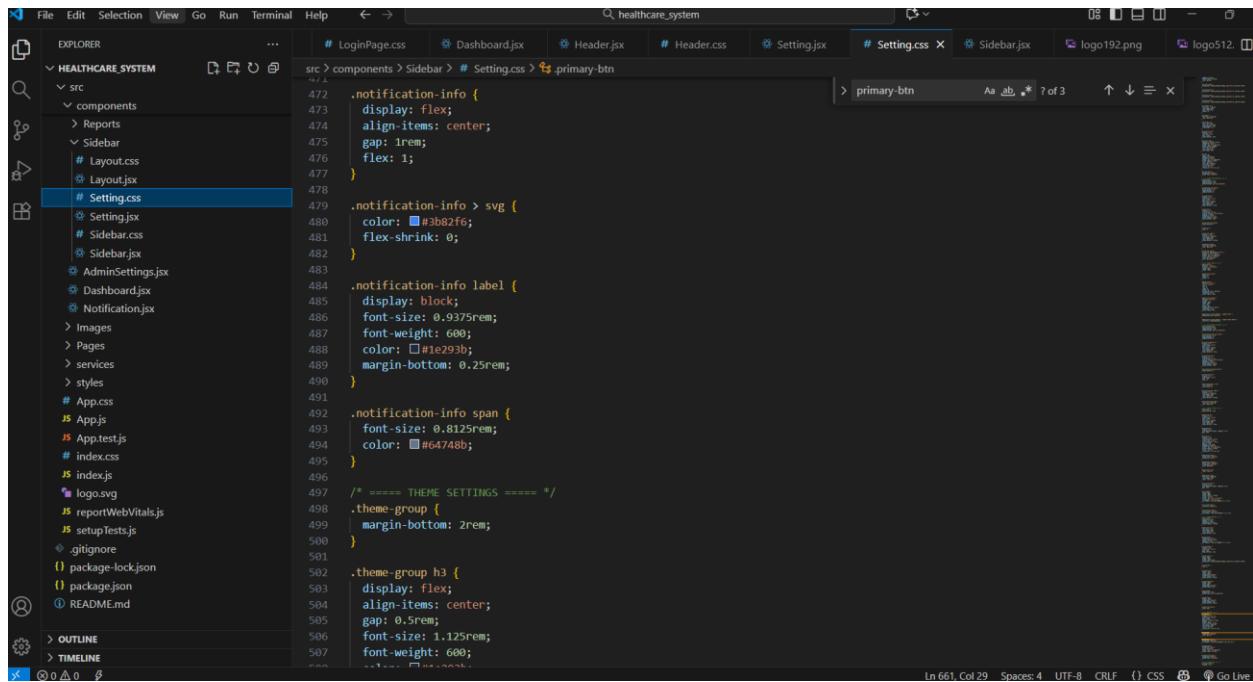


```
File Edit Selection View Go Run Terminal Help ← → Q healthcare_system
EXPLORER ... # LoginPage.css Dashboard.jsx Header.jsx # Header.css Setting.jsx # Setting.css Sidebar.jsx logo192.png logo512.png ...
HEALTHCARE_SYSTEM
src
  components
    Reports
    Sidebar
      # Layout.css
      # Layout.jsx
      # Setting.css
      # Setting.jsx
      # Sidebar.css
      # Sidebar.jsx
      # Sidebar.jsx
    AdminSettings.jsx
    Dashboard.jsx
    Notification.jsx
  Images
  Pages
  services
  styles
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
OUTLINE
TIMELINE
X 0 △ 0 ⌂
```

```
src > components > Sidebar > Setting.jsx > Setting > renderProfileSettings

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,
36  });
37
```

In 271, Col 33 Spaces: 4 UTF-8 CRLF {} JavaScript JSX ⌂ Go Live ⌂



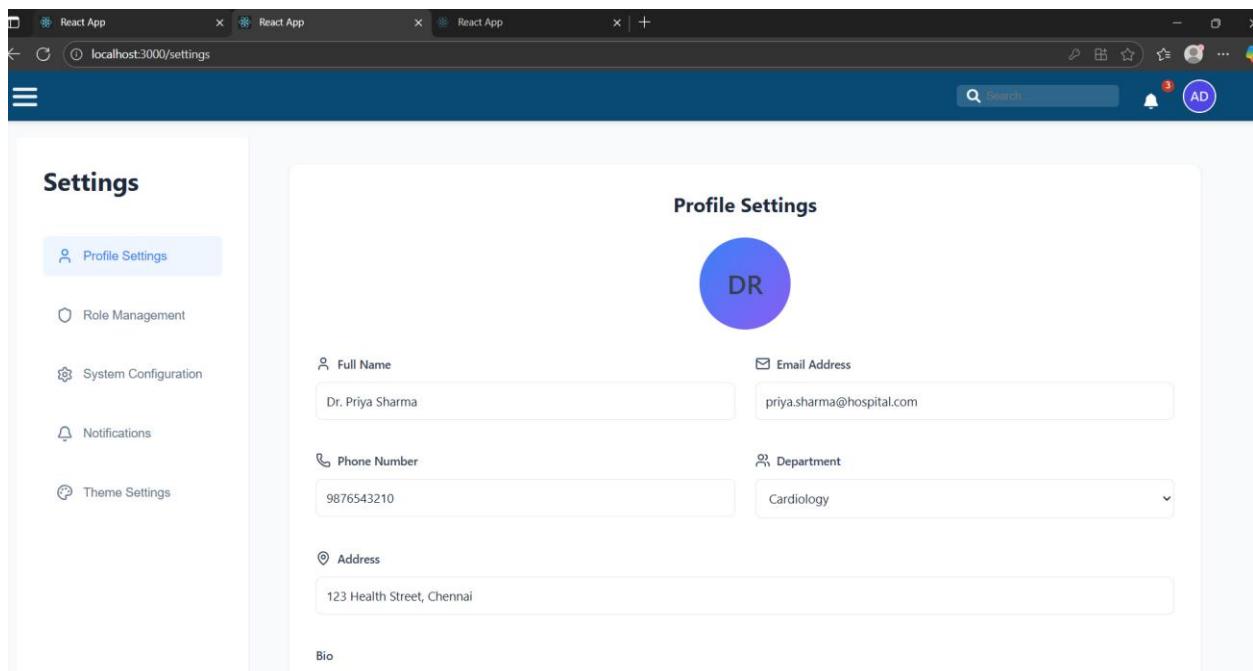
```

File Edit Selection View Go Run Terminal Help < -> healthcare_system
EXPLORER ... # LoginPage.css # Dashboard.jsx # Header.jsx # Header.css # Setting.jsx # Setting.css Sidebar.jsx logo192.png logo512.d
HEALTHCARE_SYSTEM ...
src ...
  components ...
    Reports ...
    Sidebar ...
      Layout.css ...
      Layout.jsx ...
    # Setting.css ...
      Setting.jsx ...
      # Sidebar.css ...
      Sidebar.jsx ...
    AdminSettings.jsx ...
    Dashboard.jsx ...
    Notification.jsx ...
    Images ...
    Pages ...
    services ...
    styles ...
    # App.css ...
    JS App.js ...
    JS App.test.js ...
    # index.css ...
    JS index.js ...
    logo.svg ...
    JS reportWebVitals.js ...
    JS setupTests.js ...
    .gitignore ...
    package-lock.json ...
    package.json ...
    README.md ...
  OUTLINE ...
  TIMELINE ...
  0 0 0 0 ...
  primary-btn ...
  Aa ab ...
  7 of 3 ...
  Go Live ...
  In 661 Col 29 Spaces: 4 UTF-8 CRLF {} CSS ...

```

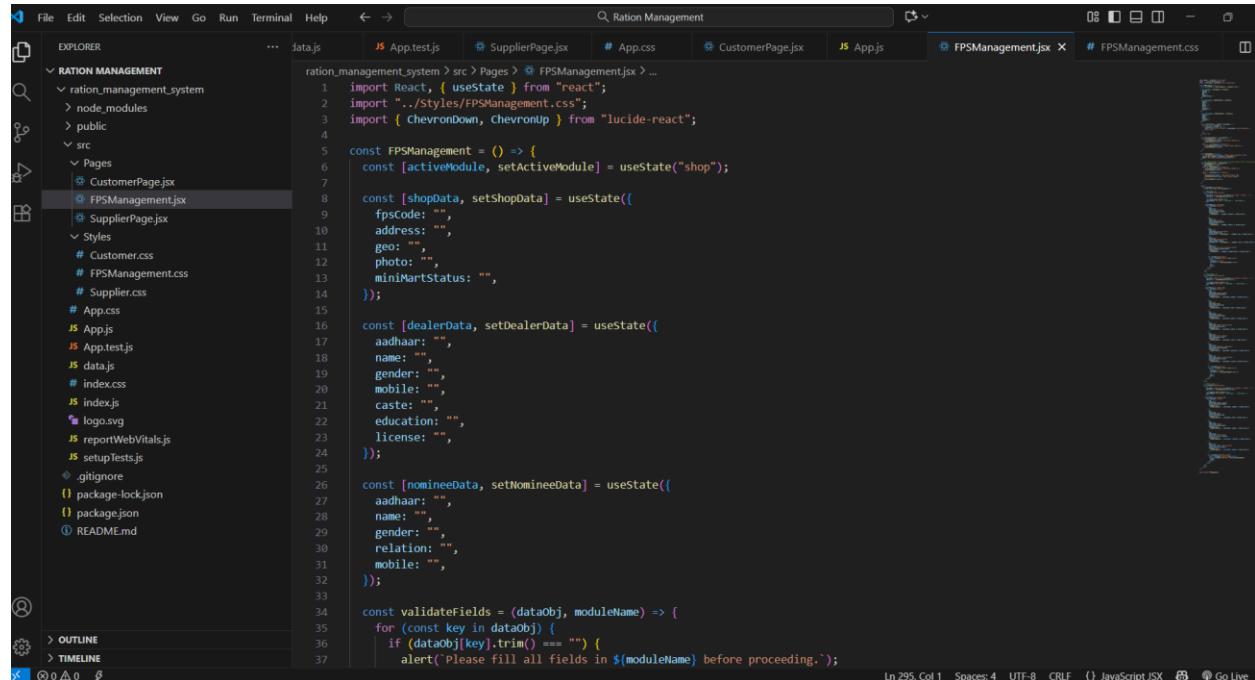
The screenshot shows the VS Code interface with the 'Setting.css' file selected in the Explorer sidebar. The code editor displays CSS for a 'primary-btn' class, which includes styling for a flex container, an SVG icon, and text labels. The status bar at the bottom indicates the file has 661 lines, 29 columns, and is in UTF-8 format.

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.



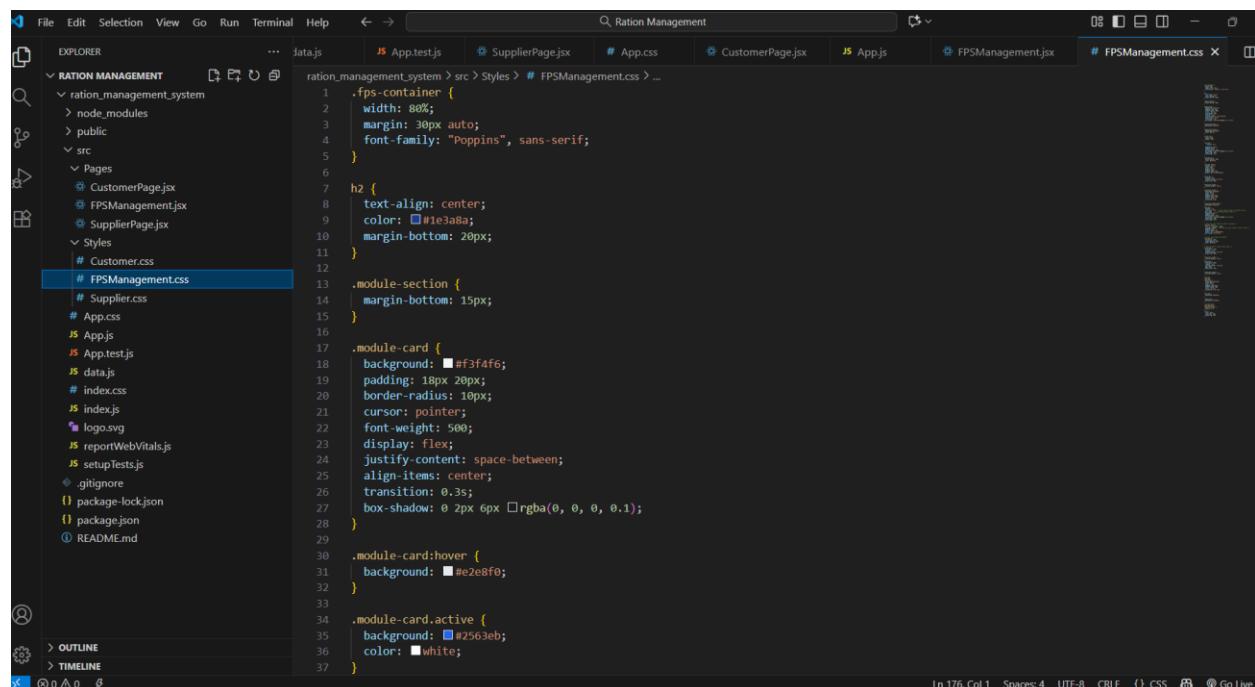
Ration Management System:

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



```
File Edit Selection View Go Run Terminal Help < > Ration Management
EXPLORER ... /data.js JS App.test.js SupplierPage.jsx # App.css CustomerPage.jsx JS App.js # FPSManagement.jsx # FPSManagement.css
RATION MANAGEMENT
ration_management_system
node_modules
public
src
Pages
CustomerPage.jsx
# FPSManagement.jsx
SupplierPage.jsx
Styles
# Customer.css
# FPSManagement.css
# Supplier.css
App.css
JS App.js
JS App.test.js
JS data.js
# index.css
JS index.jsx
logo.svg
JS reportWebVitals.js
JS setupTests.js
.gitignore
package-lock.json
package.json
README.md
OUTLINE > TIMELINE
Ln 295, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript JSX Go Live
```

```
import React, { useState } from "react";
import "./styles/FPSManagement.css";
import { ChevronDown, ChevronUp } from "lucide-react";
const FPSManagement = () => {
  const [activeModule, setActiveModule] = useState("shop");
  const [shopData, setShopData] = useState({
    fpicode: "",
    address: "",
    geo: "",
    photo: "",
    miniMartStatus: ""
  });
  const [dealerData, setDealerData] = useState({
    aadhaar: "",
    name: "",
    gender: "",
    mobile: "",
    caste: "",
    education: "",
    license: ""
  });
  const [nomineeData, setNomineeData] = useState({
    aadhaar: "",
    name: "",
    gender: "",
    relation: "",
    mobile: ""
  });
  const validateFields = (dataObj, moduleName) => {
    for (const key in dataObj) {
      if (dataObj[key].trim() === "") {
        alert(`Please fill all fields in ${moduleName} before proceeding.`);
      }
    }
  };
}
const validateFields = (dataObj, moduleName) => {
  for (const key in dataObj) {
    if (dataObj[key].trim() === "") {
      alert(`Please fill all fields in ${moduleName} before proceeding.`);
    }
  }
};
```



```
File Edit Selection View Go Run Terminal Help < > Ration Management
EXPLORER ... /data.js JS App.test.js SupplierPage.jsx # App.css CustomerPage.jsx JS App.js # FPSManagement.jsx # FPSManagement.css
RATION MANAGEMENT
ration_management_system
node_modules
public
src
Pages
CustomerPage.jsx
# FPSManagement.jsx
SupplierPage.jsx
Styles
# Customer.css
# Supplier.css
App.css
JS App.js
JS App.test.js
JS data.js
# index.css
JS index.jsx
logo.svg
JS reportWebVitals.js
JS setupTests.js
.gitignore
package-lock.json
package.json
README.md
OUTLINE > TIMELINE
Ln 176, Col 1 Spaces: 4 UTF-8 CRLF {} CSS Go Live
```

```
.fps-container {
  width: 80px;
  margin: 30px auto;
  font-family: "Poppins", sans-serif;
}

h2 {
  text-align: center;
  color: #1e3a8a;
  margin-bottom: 20px;
}

.module-section {
  margin-bottom: 15px;
}

.module-card {
  background: #f3f4f6;
  padding: 18px 20px;
  border-radius: 10px;
  cursor: pointer;
  font-weight: 500;
  display: flex;
  justify-content: space-between;
  align-items: center;
  transition: 0.3s;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
}

.module-card:hover {
  background: #e2e8f0;
}

.module-card.active {
  background: #2563eb;
  color: white;
}
```

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.

A screenshot of a web browser displaying the "Fair Price Shops (FPS) Management" application. The page title is "Fair Price Shops (FPS) Management". Below it, a blue header bar contains the text "Shop Details Module". The main content area contains four input fields: "FPS Code" (empty), "Address" (empty), "Geo-coordinates" (empty), and "Photograph URL" (empty). Below these fields is a text input for "Mini Mart suitability status" which also appears to be empty. At the bottom right of the content area are two buttons: "Save" and "Next →".

A screenshot of the same web browser displaying the "Fair Price Shops (FPS) Management" application. The page title is "Fair Price Shops (FPS) Management". Below it, a blue header bar contains the text "Dealer Details Module". The main content area contains four input fields: "Aadhaar Number" (empty), "Name" (empty), "Gender" (empty), and "Mobile (OTP verified)" (empty). Below these fields is a text input for "Caste Category" (empty) and another for "Education (Digilocker)" (empty). At the bottom right of the content area are two buttons: "Save" and "Next →".