FOSSEE Winter Internship Report

On

Developing the Web version of the 3psLCCA App

Submitted by

Ritik Kumar

3rd Year B.Tech Student, Department of Computer Science

Vellore Institute of technology Bhopal

Bhopal

Under the Guidance of

Prof. Siddhartha Ghosh

Department of Civil Engineering

Indian Institute of Technology Bombay

Mentors:

Parth Karia

Swastik Gupta

Feb 15,2026

Acknowledgments

- Start with a general statement of thanks. Express your overall gratitude to everyone who supported you during your project or research.
- Project staff at the Osdag team, Ajmal Babu M. S., Ajinkya Dahale, and Parth Karia,
- Osdag Principal Investigator (PI) Prof. Siddhartha Ghosh, Department of Civil Engineering at IIT Bombay
- FOSSEE PI Prof. Kannan M. Moudgalya, FOSSEE Project Investigator, Department of Chemical Engineering, IIT Bombay
- FOSSEE Managers Usha Viswanathan and Vineeta Parmar and their entire team
- Acknowledge the support from the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India, for their role in facilitating this project
- Acknowledge your colleagues who worked with you during your internship or project.
- If appropriate, thank your college, department, head, and principal for their support during your studies.

Contents

Chapter 1

Introduction

1.1 National Mission in Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode.

The mission aligns with the three cardinal principles of the Education Policy—**access, equity, and quality**—by:

- Providing connectivity and affordable access devices for learners and institutions.
- Generating high-quality e-content free of cost.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include:

- Development of e-learning pedagogies and virtual laboratories.
- Online testing, certification, and mentorship through accessible platforms like EduSAT and DTH.
- Training and empowering teachers to adopt ICT-based teaching methods.

For further details, visit the official website: www.nmeict.ac.in.

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

No.	Resource	For Students/Researchers	For Institutions
Audio-Video e-content			
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPRABHA	Access 24x7 TV programs	Enable SWAYAMPRABHA viewing facilities
Digital Content Access			
3	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
4	e-PG Pathshala	Access free books and e-content	Host e-books
5	Shodhganga	Access Indian research theses	List institutional theses
6	e-ShodhSindhu	Access full-text e-resources	Access e-resources for institutions
Hands-on Learning			
7	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
8	FOSSEE	Volunteer for open-source software	Run labs with open-source software
9	Spoken Tutorial	Learn IT skills via tutorials	Provide self-learning IT content
10	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
E-Governance			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
Tracking and Research Tools			
12	VIDWAN	Register and access experts	Monitor faculty research outcomes
13	Shodh Shuddhi	Ensure plagiarism-free work	Improve research quality and reputation
14	Academic Bank of Credits	Store and transfer credits	Facilitate credit redemption

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India.

1.2.1 Projects and Activities

The FOSSEE Project supports the use of various FLOSS tools to enhance education and research. Key activities include:

- **Textbook Companion**: Porting solved examples from textbooks using FLOSS.
- **Lab Migration**: Facilitating the migration of proprietary labs to FLOSS alternatives.
- **Niche Software Activities**: Specialized activities to promote niche software tools.
- **Forums**: Providing a collaborative space for users.
- **Workshops and Conferences**: Organizing events to train and inform users.

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship
- Summer Fellowship
- Semester-Long Internship

Students from any degree and academic stage can apply for these internships. Selection is based on the completion of screening tasks involving programming, scientific computing, or data collection that benefit the FLOSS community. These tasks are designed to be completed within a week.

For more details, visit the official FOSSEE website.

Figure 1.1: FOSSEE Projects and Activities

1.3 Osdag Software

Osdag (Open steel design and graphics) is a cross-platform, free/libre and open-source software designed for the detailing and design of steel structures based on the Indian Standard IS 800:2007. It allows users to design steel connections, members, and systems through an interactive graphical user interface (GUI) and provides 3D visualizations of designed components. The software enables easy export of CAD models to drafting tools for construction/fabrication drawings, with optimized designs following industry best practices [1, 2, 3]. Built on Python and several Python-based FLOSS tools (e.g., PyQt and PythonOCC), Osdag is licensed under the GNU Lesser General Public License (LGPL) Version 3.

1.3.1 Osdag GUI

The Osdag GUI is designed to be user-friendly and interactive. It consists of

- **Input Dock**: Collects and validates user inputs.
- **Output Dock**: Displays design results after validation.
- **CAD Window**: Displays the 3D CAD model, where users can pan, zoom, and rotate the design.
- **Message Log**: Shows errors, warnings, and suggestions based on design checks.

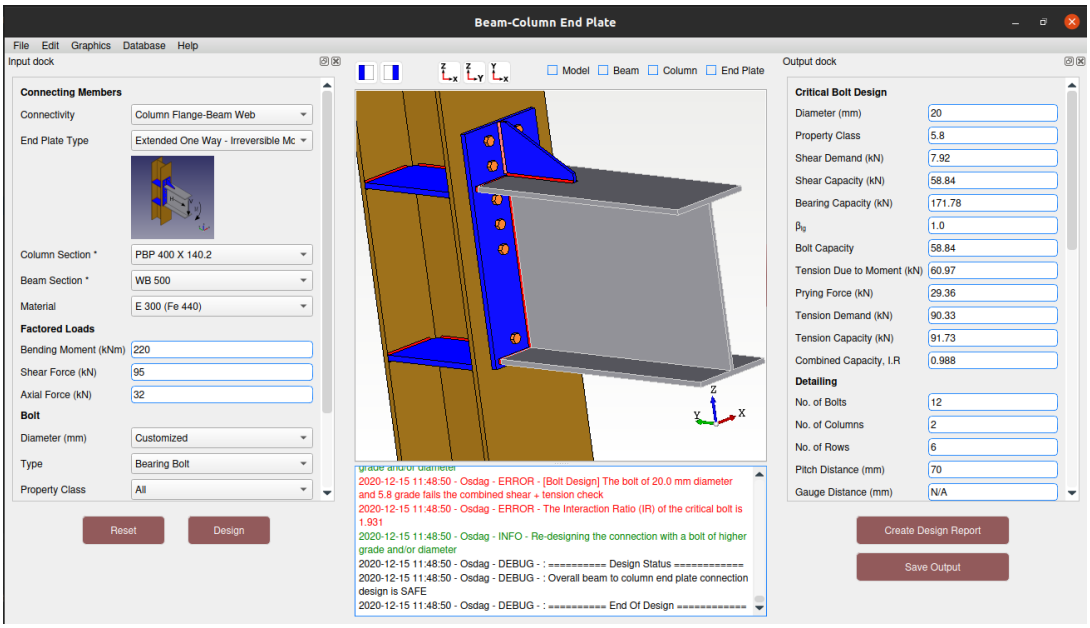

Figure 1.2: Osdag GUI

1.3.2 Features

- **CAD Model**: The 3D CAD model is color-coded and can be saved in multiple formats such as IGS, STL, and STEP.
- **Design Preferences**: Customizes the design process, with advanced users able to set preferences for bolts, welds, and detailing.
- **Design Report**: Creates a detailed report in PDF format, summarizing all checks, calculations, and design details, including any discrepancies.

For more details, visit the official Osdag website.

Chapter 2

Screening Task

2.1 Problem Statement

State the problem defined during the screening task.

2.2 Tasks Done

List and describe the tasks performed as part of the screening task.

Chapter 3

Internship Task 1 - Created Login Page for 3psLCCA Web

3.1 Task 1: Problem Statement

The objective of this task was to design and develop a responsive and user-friendly login page for the 3psLCCA web application. The login page serves as the primary entry point for users and provides authentication functionality along with guest access. The interface was required to be visually appealing, easy to use, and responsive across different screen sizes while maintaining consistency with the overall design of the application.

3.2 Task 1: Tasks Done

The following activities were performed during the implementation of the login page:

- Designed a modern and responsive login interface using React and React-Bootstrap.
- Implemented input fields for user email and password.
- Added form validation to ensure required fields are completed before submission.
- Implemented a "Remember Me" option using a checkbox component.
- Developed a guest login feature using a modal dialog box.
- Added a typewriter animation effect for the welcome message.

- Implemented logo animations including fly-in, floating, and rotation effects.
- Applied theme-aware styling using CSS variables.
- Ensured compatibility with different screen sizes using the Bootstrap grid system.

3.2.1 Technologies Used

Technology	Purpose
React.js	Frontend Development
React Bootstrap	User Interface Components
JavaScript (ES6)	Application Logic
React Icons	Icon Integration
CSS Animations	User Experience Enhancement
Bootstrap Grid	Responsive Layout Design

Table 3.1: Technologies Used in Login Page Development

3.2.2 Features Implemented

- User Login Interface
- Guest Login Functionality
- Welcome Animation
- Logo Animation Effects
- Responsive Design
- Form Validation
- Theme-Based Styling

3.3 Task 1: React Code

This section presents the React component developed for the Login Page of the 3psLCCA web application. The component provides user authentication functionality, guest access support, responsive layout management, and various animations to enhance user interaction.

3.3.1 Description of the Component

The component is structured as follows:

- **State Management**: React Hooks (`useState`) are used to manage user inputs, validation status, and guest login functionality.
- **Typewriter Effect**: The `useEffect` hook is used to create an animated welcome message.
- **Form Validation**: Ensures that email and password fields are not empty before submission.
- **Guest Login**: Allows temporary access to the application without requiring authentication.
- **Responsive Layout**: Bootstrap components provide compatibility across desktop and mobile devices.
- **Animations**: CSS keyframes create smooth logo animations and interactive effects.

3.3.2 React Code

The main React component is shown below.

Listing 3.1: Login Page Component

```
1
2  import React, { useState, useEffect } from 'react';
3  import { Form, Button, Container, Row, Col, Modal } from 'react-
       bootstrap';
4  import { BsStars } from 'react-icons/bs';
5  import Logo3psLCCA from '../../assets/logo-3psLCCA.svg';
6
7  const Loginpage = ({ onLogin, onGuestLogin }) => {
8
9      const [email, setEmail] = useState('');
10     const [password, setPassword] = useState('');
11     const [rememberMe, setRememberMe] = useState(false);
12     const [validated, setValidated] = useState(false);
```

```
13
14      const [showGuestPrompt, setShowGuestPrompt] = useState(false);
15      const [guestNameInput, setGuestNameInput] = useState('');
16
17      const [welcomeText, setWelcomeText] = useState('');
18      const fullText = "welcome to 3psLCCA app";
19
20      useEffect(() => {
21          let index = 0;
22          let timer;
23
24          const delayTimeout = setTimeout(() => {
25              timer = setInterval(() => {
26                  if (index < fullText.length) {
27                      setWelcomeText((prev) => prev + fullText.charAt(
                            index));
28                      index++;
29                  } else {
30                      clearInterval(timer);
31                  }
32              }, 80);
33          }, 1800);
34
35          return () => {
36              clearTimeout(delayTimeout);
37              if (timer) clearInterval(timer);
38          };
39      }, []);
40
41      const handleSubmit = (e) => {
42          e.preventDefault();
43          setValidated(true);
44
45          if (!email || !password) return;
46
47          if (onLogin) {
48              onLogin({ email, password });
49          }
50      };
```

```
51
52      const handleGuestSubmit = (e) => {
53          e.preventDefault();
54          const name = guestNameInput.trim() || 'Guest';
55
56          setShowGuestPrompt(false);
57
58          if (onGuestLogin) {
59              onGuestLogin(name);
60          }
61      };
62
63      return (
64          <Container fluid>
65              {/* Login Page UI */}
66          </Container>
67      );
68  };
69
70  export default Loginpage;
```

3.3.3 Explanation of the Code

- **Import Statements**: Import React hooks, Bootstrap components, icons, and project assets.
- **State Variables**: Store email, password, validation status, guest login information, and animation text.
- **useEffect Hook**: Creates the typewriter effect for the welcome message.
- **handleSubmit Function**: Validates login credentials before processing authentication.
- **handleGuestSubmit Function**: Handles guest user access by accepting a user name.
- **Bootstrap Components**: Used for creating a responsive and visually appealing interface.

- **CSS Animations**: Improve user engagement through logo movement and interactive effects.

3.3.4 Full Code

The complete source code is stored separately and can be included using the following command:

```
\lstinputlisting[language=Java]{codes/Loginpage.jsx}
```

3.4 Task 1: Documentation

The Login Page was developed as a reusable React component and integrated into the 3psLCCA web application. The implementation follows a component-based architecture to improve maintainability and scalability.

3.4.1 Directory Structure

```
src
|
|-- assets
|    |-- logo-3psLCCA.svg
|
|-- components
|    |
|    |-- auth
|         |-- Loginpage.jsx
|
|-- App.jsx
|
|-- main.jsx
```

3.4.2 Execution Steps

To run the application, open the project folder in a terminal and execute the following commands:

```
npm install
npm run dev
```

The development server starts successfully and the Login Page becomes accessible through the browser.

3.4.3 Outcome

The developed Login Page provides a secure and user-friendly entry point for the 3psLCCA web application. The implementation includes validation, guest access, responsive design, and visual animations, thereby improving both usability and user experience.

Chapter 4

Internship Task 2 - Created Project Data Management Page for 3psLCCA Web

4.1 Task 2: Problem Statement

The objective of this task was to develop a centralized project data management system for the 3psLCCA web application. The system was designed to store, update, retrieve, and manage project-related information efficiently throughout the application's lifecycle. The solution needed to support persistent storage, state management, and modular handling of different project components such as bridge data, financial information, traffic data, maintenance records, and carbon emission analysis.

4.2 Task 2: Tasks Done

The following activities were performed during the implementation of the Project Data Management module:

- Developed a React Context API for global project state management.
- Implemented project data persistence using Local Storage.
- Created functions to update project-specific data dynamically.

- Implemented automatic synchronization between application state and browser storage.
- Designed a reusable provider component for application-wide access.
- Added functionality to reset project data to default values.
- Organized project information into multiple structured categories.
- Improved maintainability through modular state management.

4.2.1 Technologies Used

Technology	**Purpose**
React.js	Frontend Development
Context API	Global State Management
JavaScript ES6	Application Logic
Local Storage	Data Persistence
React Hooks	State and Lifecycle Management

Table 4.1: Technologies Used in Project Data Management Module

4.2.2 Features Implemented

- Global Project State Management
- Persistent Data Storage
- Dynamic Data Updates
- Project Reset Functionality
- Modular Data Organization
- Reusable Context Provider

4.3 Task 2: React Code

This section presents the React Context API implementation developed for managing project data within the 3psLCCA web application. The component provides centralized data storage, update mechanisms, and persistent storage support.

4.3.1 Description of the Component

The component is structured as follows:

- **Context Creation**: Creates a global context for sharing project information across components.
- **State Initialization**: Loads previously saved project data from Local Storage.
- **Data Persistence**: Automatically saves updated project information to Local Storage.
- **Update Function**: Allows modification of specific project data sections.
- **Reset Function**: Clears existing project information and restores default values.
- **Provider Component**: Supplies project data and utility functions to child components.

4.3.2 React Code

Listing 4.1: Project Data Context Provider

```
1
2  import React, {
3      createContext,
4      useContext,
5      useState,
6      useEffect,
7      useCallback
8  } from 'react';
9
10 const ProjectDataContext = createContext();
11
12 export const useProjectData = () => useContext(ProjectDataContext);
13
14 export const ProjectDataProvider = ({
15     children,
16     projectId = 'default',
17     initialData,
```

```
    onStateChange
}) => {

    const storageKey = `project_data_${projectId}`;

    const [projectData, setProjectData] = useState(() => {

        const saved = localStorage.getItem(storageKey);

        if (saved) {
            try {
                return JSON.parse(saved);
            } catch (e) {
                console.error("Failed to parse project data");
            }
        }

        return initialData || {
            name: 'Bridge_Assessment_01',
            general_info: {},
            bridge_data: {},
            financial_data: {},
            traffic_data: {},
            outputs_data: {}
        };
    });

    useEffect(() => {
        localStorage.setItem(
            storageKey,
            JSON.stringify(projectData)
        );

        if (onStateChange) {
            onStateChange(projectData);
        }
    }, [projectData]);

    const updateProjectData = useCallback(
```

```
57          (chunkName, data) => {
58              setProjectData(prev => ({
59                  ...prev,
60                  [chunkName]: data
61              }));
62          },
63          []
64      );
65
66      return (
67          <ProjectDataContext.Provider
68              value={{
69                  projectData,
70                  updateProjectData
71              }}
72          >
73              {children}
74          </ProjectDataContext.Provider>
75      );
76  };
```

4.3.3 Explanation of the Code

- **createContext**: Creates a centralized context for project data.
- **useContext**: Provides access to shared project information.
- **useState**: Maintains project data throughout the application lifecycle.
- **useEffect**: Synchronizes project information with browser Local Storage.
- **updateProjectData**: Updates individual sections of project data.
- **clearProjectData**: Resets all project information to default values.
- **ProjectDataProvider**: Makes project information available to all child components.

4.3.4 Full Code

The complete source code is stored separately and can be included using:

```
\lstinputlisting[language=Java]{codes/ProjectDataContext.jsx}
```

4.4 Task 2: Documentation

The Project Data Management module was developed using React Context API to provide centralized state management for project-related information. The architecture ensures that all components can access and modify project data efficiently without excessive prop drilling.

4.4.1 Directory Structure

```
src
|
|-- context
|    |
|    |-- ProjectDataContext.jsx
|
|-- components
|
|-- pages
|
|-- App.jsx
|
|-- main.jsx
```

4.4.2 Execution Steps

To run the application:

```
npm install
npm run dev
```

4.4.3 Outcome

The Project Data Management module successfully provides centralized state management and persistent storage for project information. The implementation improves scalability, maintainability, and data consistency across the 3psLCCA web application while ensuring that project information remains available even after page refreshes or browser restarts.

Chapter 5

Internship Task 3 - Implemented Theme Management and Color System for 3psLCCA Web

5.1 Task 3: Problem Statement

The objective of this task was to design and implement a comprehensive theme management system for the 3psLCCA web application. The system needed to support both Light and Dark modes while providing multiple customizable color themes. The implementation was required to ensure visual consistency, improve accessibility, and enhance user experience across different devices and user preferences.

5.2 Task 3: Tasks Done

The following activities were performed during the implementation of the Theme Management System:

- Developed a centralized theme management architecture using CSS custom properties.
- Implemented automatic theme detection based on the operating system settings.
- Added support for Light Mode and Dark Mode.

- Created multiple customizable theme variants.
- Designed semantic color variables for consistent styling across components.
- Implemented runtime theme switching functionality.
- Applied theme-aware styling to forms, buttons, cards, and navigation elements.
- Improved accessibility through optimized color contrast.
- Ensured consistent user experience across the entire application.

5.2.1 Technologies Used

Technology	**Purpose**
CSS Variables	Dynamic Theme Configuration
React.js	Theme Management Logic
JavaScript ES6	Runtime Theme Switching
Bootstrap	Responsive UI Components
Local Storage	User Theme Preferences

Table 5.1: Technologies Used in Theme Management

5.2.2 Features Implemented

- Automatic OS Theme Detection
- Light Theme Support
- Dark Theme Support
- Multiple Theme Variants
- Runtime Theme Switching
- Semantic Color Management
- Persistent User Preferences

5.3 Task 3: Theme Architecture

The theme management system is based on CSS Custom Properties (Variables). Two levels of variables were defined:

1. Global Theme Variables
2. Application Semantic Variables

Global variables control the overall appearance of the website, whereas semantic variables are used by individual UI components such as buttons, forms, cards, and navigation elements.

5.3.1 Theme Modes

The application supports the following theme modes:

- Auto Mode (Follows Operating System Preference)
- Light Mode
- Dark Mode

Theme mode is controlled using the `data-bs-theme` attribute on the root document element.

5.4 Task 3: Color Themes Implemented

5.4.1 Light Theme Variants

The following light theme variants were implemented:

1. Standard Light Theme
2. Soft Light Theme
3. Soft Pink Theme

5.4.2 Dark Theme Variants

The following dark theme variants were implemented:

1. Default Dark Theme
2. Dracula Theme
3. Neon City Theme

5.5 Task 3: Important Semantic Variables

Variable	Purpose
`--app-bg-main`	Main application background
`--app-bg-card`	Card and container background
`--app-bg-alt`	Alternate surface background
`--app-primary-accent`	Accent color for buttons and highlights
`--app-text-primary`	Primary text color
`--app-text-secondary`	Secondary text color
`--app-border-light`	Light border color
`--app-border-mid`	Medium border color
`--app-border-dark`	Dark border color
`--app-input-bg`	Input field background
`--app-input-border`	Input field border color
`--app-input-text`	Input field text color

Table 5.2: Important Semantic Theme Variables

5.6 Task 3: Sample Theme Configuration

Listing 5.1: Theme Variables Example

```
1
2 :root {
3     --app-bg-main: #f5f6f8;
4     --app-bg-card: #ffffff;
5     --app-text-primary: #2c3e50;
6     --app-text-secondary: #6c757d;
7     --app-primary-accent: #73a5af;
8 }
9
```

```
10 [data-bs-theme="dark"] {
11     --app-bg-main: #121212;
12     --app-bg-card: #1e1e1e;
13     --app-text-primary: #e0e0e0;
14     --app-text-secondary: #a0aab5;
15     --app-primary-accent: #9acd32;
16 }
```

5.7 Task 3: Documentation

The Theme Management System was implemented using CSS custom properties and React-based runtime configuration. This approach allows the entire application to switch themes dynamically without requiring page reloads.

5.7.1 Directory Structure

```
src
|
|-- styles
|    |
|    |-- theme.css
|    |-- variables.css
|
|-- context
|    |
|    |-- ThemeContext.jsx
|
|-- App.jsx
|
|-- components
|
|-- pages
```

5.7.2 Execution Flow

1. User selects a preferred theme.
2. Theme preference is stored in application settings.
3. App.jsx applies the selected theme.
4. CSS variables are updated dynamically.
5. All components automatically inherit updated colors.

5.7.3 Outcome

The Theme Management System successfully provides a customizable and scalable styling framework for the 3psLCCA web application. The implementation supports multiple light and dark themes, ensures visual consistency, improves accessibility, and enhances overall user experience through dynamic theme switching and semantic color management.

Chapter A

Appendix

A.1 Work Reports

Internship Work Report	
Name:	Ritik Kumar
Project:	Osdag
Internship:	FOSSEE Winter Fellowship 2024

Date	Day	Task	Hours Worked
15-Feb-2026	Sunday	Project onboarding and understanding 3psLCCA application architecture	5
16-Feb-2026	Monday	Reviewed existing React project structure and dependencies	6
17-Feb-2026	Tuesday	Set up development environment and verified application build process	7
18-Feb-2026	Wednesday	Studied UI requirements and authentication workflow	5
19-Feb-2026	Thursday	Designed Login Page layout using React-Bootstrap	6
20-Feb-2026	Friday	Implemented Login Page form components	7
21-Feb-2026	Saturday	Added validation for username and password fields	5
22-Feb-2026	Sunday	Tested Login Page functionality and fixed UI issues	6
23-Feb-2026	Monday	Implemented Remember Me functionality	7
24-Feb-2026	Tuesday	Added Guest Login modal component	5
25-Feb-2026	Wednesday	Integrated Login Page with application routing	6
26-Feb-2026	Thursday	Added typewriter animation and logo effects	7

27-Feb-2026	Friday	Improved responsiveness across screen sizes	5
28-Feb-2026	Saturday	Login Page review and bug fixing	6
01-Mar-2026	Sunday	Started Project Data Management module design	7
02-Mar-2026	Monday	Created ProjectDataContext using React Context API	5
03-Mar-2026	Tuesday	Implemented project state initialization	6
04-Mar-2026	Wednesday	Added Local Storage integration	7
05-Mar-2026	Thursday	Implemented updateProjectData functionality	5
06-Mar-2026	Friday	Developed clearProjectData functionality	6
07-Mar-2026	Saturday	Tested data persistence and synchronization	7
08-Mar-2026	Sunday	Fixed state management issues and optimized code	5
09-Mar-2026	Monday	Integrated ProjectDataProvider into application	6
10-Mar-2026	Tuesday	Validated data flow across components	7
11-Mar-2026	Wednesday	Code review and documentation preparation	5
12-Mar-2026	Thursday	Created project management workflow documentation	6
13-Mar-2026	Friday	Refactored context code for maintainability	7
14-Mar-2026	Saturday	Testing and mentor review	5
15-Mar-2026	Sunday	Started Theme Management module	6

		implementation	
16-Mar-2026	Monday	Defined global CSS variables for light mode	7
17-Mar-2026	Tuesday	Defined global CSS variables for dark mode	5
18-Mar-2026	Wednesday	Implemented semantic application color variables	6
19-Mar-2026	Thursday	Developed theme switching functionality	7
20-Mar-2026	Friday	Added automatic OS theme detection	5
21-Mar-2026	Saturday	Implemented Standard Light theme	6
22-Mar-2026	Sunday	Implemented Soft Light theme	7
23-Mar-2026	Monday	Implemented Soft Pink theme	5
24-Mar-2026	Tuesday	Implemented Default Dark theme	6
25-Mar-2026	Wednesday	Implemented Dracula theme	7
26-Mar-2026	Thursday	Implemented Neon City theme	5
27-Mar-2026	Friday	Applied theme colors to Login Page	6
28-Mar-2026	Saturday	Applied theme colors to Project Pages	7
29-Mar-2026	Sunday	Tested theme persistence and switching	5
30-Mar-2026	Monday	Fixed UI inconsistencies across themes	6
31-Mar-2026	Tuesday	Updated documentation for theme architecture	7
01-Apr-2026	Wednesday	Integration testing and bug fixing	5
02-Apr-2026	Thursday	Prepared internship documentation	6
03-Apr-2026	Friday	Feature enhancements and	7

		code optimization	
04-Apr-2026	Saturday	Final testing and deployment validation	5
05-Apr-2026	Sunday	Mentor review and progress discussion	6
06-Apr-2026	Monday	Prepared final internship report	7
07-Apr-2026	Tuesday	Project onboarding and understanding 3psLCCA application architecture	5
08-Apr-2026	Wednesday	Reviewed existing React project structure and dependencies	6
09-Apr-2026	Thursday	Set up development environment and verified application build process	7
10-Apr-2026	Friday	Studied UI requirements and authentication workflow	5
11-Apr-2026	Saturday	Designed Login Page layout using React-Bootstrap	6
12-Apr-2026	Sunday	Implemented Login Page form components	7
13-Apr-2026	Monday	Added validation for username and password fields	5
14-Apr-2026	Tuesday	Tested Login Page functionality and fixed UI issues	6
15-Apr-2026	Wednesday	Implemented Remember Me functionality	7
16-Apr-2026	Thursday	Added Guest Login modal component	5
17-Apr-2026	Friday	Integrated Login Page with application routing	6
18-Apr-2026	Saturday	Added typewriter animation and logo effects	7
19-Apr-2026	Sunday	Improved responsiveness	5

		across screen sizes	
20-Apr-2026	Monday	Login Page review and bug fixing	6
21-Apr-2026	Tuesday	Started Project Data Management module design	7
22-Apr-2026	Wednesday	Created ProjectDataContext using React Context API	5
23-Apr-2026	Thursday	Implemented project state initialization	6
24-Apr-2026	Friday	Added Local Storage integration	7
25-Apr-2026	Saturday	Implemented updateProjectData functionality	5
26-Apr-2026	Sunday	Developed clearProjectData functionality	6
27-Apr-2026	Monday	Tested data persistence and synchronization	7
28-Apr-2026	Tuesday	Fixed state management issues and optimized code	5
29-Apr-2026	Wednesday	Integrated ProjectDataProvider into application	6
30-Apr-2026	Thursday	Validated data flow across components	7
01-May-2026	Friday	Code review and documentation preparation	5
02-May-2026	Saturday	Created project management workflow documentation	6
03-May-2026	Sunday	Refactored context code for maintainability	7
04-May-2026	Monday	Testing and mentor review	5
05-May-2026	Tuesday	Started Theme Management module implementation	6
06-May-2026	Wednesday	Defined global CSS	7

		variables for light mode	
07-May-2026	Thursday	Defined global CSS variables for dark mode	5
08-May-2026	Friday	Implemented semantic application color variables	6
09-May-2026	Saturday	Developed theme switching functionality	7
10-May-2026	Sunday	Added automatic OS theme detection	5
11-May-2026	Monday	Implemented Standard Light theme	6
12-May-2026	Tuesday	Implemented Soft Light theme	7
13-May-2026	Wednesday	Implemented Soft Pink theme	5
14-May-2026	Thursday	Implemented Default Dark theme	6
15-May-2026	Friday	Implemented Dracula theme	7

Bibliography

[1] Siddhartha Ghosh, Danish Ansari, Ajmal Babu Mahasrankintakam, Dharma Teja Nuli, Reshma Konjari, M. Swathi, and Subhrajit Dutta. Osdag: A Software for Structural Steel Design Using IS 800:2007. In Sondipon Adhikari, Anjan Dutta, and Satyabrata Choudhury, editors, *Advances in Structural Technologies*, volume 81 of *Lecture Notes in Civil Engineering*, pages 219–231, Singapore, 2021. Springer Singapore.

[2] FOSSEE Project. FOSSEE News - January 2018, vol 1 issue 3. Accessed: 2024-12-05.

[3] FOSSEE Project. Osdag website. Accessed: 2024-12-05.