Staff Substitution and Duty Assigning System

RESHVANTH R R
VASANTH S
SATHISH M

BACHELOR OF ENGINEERING
in

COMPUTER SCIENCE AND ENGINEERING
ADHIYAMAAN COLLEGE OF ENGINEERING (AUTONOMOUS)DR.

M.G.R NAGAR, HOSUR-635130

ABSTRACT

The Staff Substitution & Duty Assigning System is a web-based solution developed to simplify staff availability and substitution management in colleges. Traditionally, when a faculty member is absent or engaged in other duties, administrators manually assign substitutes, which is time-consuming and often unfairly distributed. This system automates the process to ensure accuracy, fairness, and efficiency. Through the Staff Dashboard, teachers can log their daily status such as Available, Absent, On Duty (OD), Exam Duty, or Correction Duty. The system then uses the stored timetable and duty information to identify periods that need substitution. At the core of the application is the Substitution Engine, which checks the availability of other staff members, filters out those marked busy, and automatically assigns a substitute. Priority is given to staff who have the least workload for the day, ensuring fair and balanced responsibility sharing. The Admin Dashboard allows administrators to upload and manage the timetable, monitor staff availability, view automatically assigned substitutions, and generate reports. A Reports Module provides daily and weekly substitution logs along with workload distribution analytics. With a simple and modern interface supported by smooth animations, the system reduces manual effort, prevents confusion, ensures transparency, and guarantees uninterrupted class schedules.

1.1 OVERVIEW

1.2 CHAPTER 1 INTRODUCTION

1.1 OVERVIEW

Efficient staff management and duty allocation remain critical challenges in educational institutions, especially when handling sudden staff absences or timetable disruptions. Many schools and colleges still depend on manual registers or spreadsheets for managing teacher availability and substitution records, leading to confusion, workload imbalance, and loss of instructional time. To overcome these inefficiencies, the Staff Substitution and Duty Assigning System has been developed as a digital solution to automate and streamline staff allocation processes through a centralized, intelligent platform.

The Staff Substitution and Duty Assigning System is a full-stack web-based management platform designed to assist administrators in efficiently managing staff schedules, substitutions, and workload distribution. It bridges the communication gap between administrators, staff, and management by providing real-time data access and automated task assignment. The system ensures that whenever a staff member is absent, substitutes are automatically suggested and assigned based on availability, subject expertise, and workload constraints, minimizing classroom disruptions and ensuring smooth academic operations.

Technologically, the platform is built using a modern MERN-like architecture, featuring React and TypeScript for the frontend, Node.js and Express for the backend, and PostgreSQL as the primary database managed via Prisma ORM. This ensures scalability, security, and high system performance. Secure Firebase Authentication and JWT-based role access

guarantee that only authorized personnel—such as administrators and staff—can access or modify data. The responsive user interface allows seamless usage across desktops, tablets, and mobile devices, making it adaptable to institutional environments of all scales.
The system supports three primary user roles:

- **Administrators** can manage staff records, view absence requests, assign substitutes, and monitor real-time duty allocations through a centralized control panel.
- **Staff members** can submit leave requests, view assigned substitutions, and update class or duty statuses easily.
- **Non-teaching staff** can be assigned to administrative or event-based duties, ensuring efficient utilization of human resources across departments.

Key features include automated substitution assignment, real-time schedule updates, workload analytics, PDF/Excel report generation, and notification alerts for new assignments or changes. The system also maintains a detailed record of each staff member's teaching subjects, availability, and substitution history, allowing administrators to make informed and balanced decisions.
By implementing this system, institutions can significantly reduce the administrative burden, eliminate scheduling conflicts, and enhance overall staff coordination. It replaces cumbersome manual methods with an intelligent, data-driven approach, ensuring fair workload distribution and uninterrupted academic flow.
Ultimately, the Staff Substitution and Duty Assigning System represents a step toward smart institutional management. It enhances transparency, improves operational efficiency, and ensures that teaching and administrative responsibilities are handled seamlessly—even during unforeseen staff absences. Through automation and centralized data management, the system contributes to a more organized, efficient, and responsive educational environment.

1.3 OBJECTIVE

The main objectives of the project are:

• To develop a **centralized digital platform** that replaces traditional manual methods of staff management with an automated system for tracking attendance, scheduling, and substitution allocation. This ensures accuracy, transparency, and efficiency in managing teaching and non-teaching staff duties within educational institutions.

• To **enhance coordination among administrators and staff** through real-time data sharing, automated notifications, and secure access controls. This objective focuses on reducing communication delays, ensuring timely substitution assignments, and maintaining uninterrupted academic and administrative operations.

• To provide **administrators with analytical tools and reporting features** that support effective monitoring, performance evaluation, and workload management. The goal is to enable informed decision-making and balanced duty allocation based on accurate and up-to-date staff data.

CHAPTER 2 LITERATURE SURVEY

[1] Kavitha R., Nandhini S., and Meenakshi P. (2024). Automated Staff Allocation and Substitution Management System. IJERT | Volume 13, Issue 4 | ISSN: 2278-0181.
This study introduces an automated web-based system for managing staff allocation and substitution in educational institutions. The system automates substitution assignments based on teacher availability and subject expertise, reducing administrative workload and scheduling conflicts. It also integrates real-time notifications to keep staff informed of changes. However, the study lacks advanced analytics and workload visualization features that would further support data-driven decision-making for administrators.

[2] Anitha M. and Karthikeyan R. (2023). Smart Faculty Substitution and Attendance Management System. IJSART | Volume 9, Issue 6 | ISSN: 2395-1052.
This paper presents a smart substitution management system that automates teacher replacement during absences and manages attendance digitally. It emphasizes efficiency in academic scheduling through a centralized database and real-time updates. While the system efficiently handles substitution tasks, it does not include role-based access control or duty allocation for non-teaching staff, limiting its scope to classroom-level management.

[3] Patel, D., and Shah, A. (2022). Faculty Management and Duty Scheduling System Using Web Technologies. IJCSMC | Volume 11, Issue 9 | ISSN: 2320-088X.
The authors propose a web-based platform to streamline faculty duty scheduling and workload balancing. The system provides administrators with tools to manage classes, duties, and leave requests through an online interface. It enhances coordination by reducing manual errors in timetable adjustments. However, the work does not address mobile accessibility or notification mechanisms, which are essential for ensuring quick staff communication in dynamic institutional settings.

[4] Rahman, S., and Thomas, J. (2021). Design and Implementation of an Automated Staff Substitution System for Educational Institutions. IJARCCE | Volume 10, Issue 2 | ISSN: 2278-1021.
This research discusses an automated system designed to assign substitute teachers dynamically based on availability and workload distribution. The solution minimizes class disruptions and ensures fair duty assignment. It incorporates a digital attendance module but lacks reporting features such as PDF/Excel generation and analytics dashboards for performance evaluation.

CHAPTER 3 SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

In the current educational environment, particularly in schools and colleges, staff substitution and duty management are still largely handled through manual or paper-based processes. When a teacher or staff member is absent, administrators typically rely on handwritten registers, printed timetables, or verbal communication to find available substitutes. This traditional approach often leads to confusion, scheduling conflicts, and classroom disruptions.

The manual process of managing staff availability, assigning duties, and updating records consumes a significant amount of administrative time and effort. It is also prone to human errors, such as duplicate entries, missed substitutions, and inaccurate workload tracking. In most cases, there is no centralized system to verify which staff members are available or to automatically balance workloads among teaching and non-teaching personnel.

Additionally, communication between administrators and staff is often delayed or unclear, as updates are conveyed through notice boards, phone calls, or personal messages. This lack of real-time information sharing leads to inefficiencies, misunderstandings, and missed duties, especially during sudden absences or large institutional events.

The absence of automation and digital record-keeping also makes it difficult to generate analytical reports or track historical substitution data for performance evaluation. Administrators have limited visibility into workload distribution and overall staff utilization, which affects decision-making and planning.

Due to these shortcomings, institutions face challenges in maintaining academic continuity, ensuring fair duty allocation, and optimizing staff performance. The lack of a centralized, automated, and real-time system underscores the need for a technology-driven solution like the Staff Substitution and Duty Assigning *System*, which can streamline operations, reduce errors, and enhance communication within the institution.

3.2PROPOSED SYSTEM:

The proposed system is a comprehensive digital staff management platform designed to overcome the limitations of the existing manual method of handling substitutions and duty assignments. It introduces automation, centralized data management, and role-based accessibility to streamline administrative processes and ensure smooth institutional operations.

The proposed Staff Substitution and Duty Assigning System automates the process of managing staff attendance, leave requests, and substitution scheduling. It provides a centralized online platform where administrators can monitor staff availability, assign substitutes in real time, and generate analytical reports for better decision-making. Each user—administrator, teaching staff, and non-teaching staff—has a dedicated role-based dashboard with secure access, ensuring that sensitive institutional data remains protected through JWT-based authentication.

Key features such as automated substitution suggestions, real-time notifications, workload analysis, and report generation in PDF or Excel formats make the system efficient and reliable. When a staff member is absent, the system automatically identifies suitable substitutes based on subject expertise, availability, and workload balance, ensuring fair and timely allocations. Staff members receive instant notifications of their assigned duties, minimizing communication delays and confusion.

The system is built using React and TypeScript for the frontend, Node.js and Express for the backend, and PostgreSQL as the database, managed through Prisma ORM. This architecture ensures scalability, high performance, and a responsive user interface suitable for various institutional setups. Additionally, RESTful APIs enable smooth and secure communication between the client and server, while Firebase Authentication adds an additional layer of access control and security.

By automating the substitution and duty assignment process, the proposed system minimizes manual effort, reduces scheduling conflicts, and enhances transparency in workload distribution. It empowers administrators to make data-driven decisions, ensures fair task allocation, and maintains uninterrupted teaching and administrative operations. Ultimately, the Staff Substitution and Duty Assigning System aims to improve institutional efficiency, accuracy, and communication—creating a more organized and responsive academic environment.

3.3PROPOSED SOLUTION

The Staff Substitution and Duty Assigning System offers a digital, web-based solution to replace the existing manual process of managing staff duties and substitutions. It connects administrators and staff through role-based access, ensuring secure and efficient management of schedules, attendance, and leave requests.

Built with React, Node.js, Express, and PostgreSQL, the system automates substitution assignments, duty allocation, and workload tracking. Key features include real-time notifications, automated substitute suggestions, and report generation in PDF/Excel formats.

This solution improves accuracy, transparency, and coordination, reducing administrative workload and ensuring fair, efficient, and uninterrupted staff management across the institution.

3.4 IDEATION & BRAINSTORMING

The ideation and brainstorming phase played a crucial role in shaping the Staff Substitution and Duty Assigning System into an efficient and user-friendly staff management solution. The goal was to identify real institutional challenges and design a system that could automate substitutions, optimize duty allocation, and improve communication within educational environments.

1. Problem Identification

In the initial phase, the team analyzed the major issues faced by administrators and staff in schools and colleges. Key problems identified included:

• Manual substitution and duty assignment causing confusion and time delays.
• Lack of a centralized database for staff schedules, availability, and workload tracking.
• Difficulty in balancing teaching and non-teaching staff duties fairly.
• Poor communication during sudden absences or schedule changes.
• Absence of automated reporting tools and analytics for performance monitoring.

These challenges highlighted the need for a digital, automated, and centralized platform to manage institutional staff efficiently.

2. Idea Generation

Brainstorming sessions using mind mapping, “How Might We” questions, and workflow visualization led to the generation of several innovative ideas, such as:

• Centralized Staff Management System – a unified platform for administrators and staff.
• Automated Substitution Module – to instantly assign replacements based on availability and expertise.
• Real-Time Notification System – to alert staff about substitutions and duty changes.
• Workload Analytics Dashboard – to monitor and balance teaching loads fairly.
• Role-Based Access Control – separate interfaces for administrators, teachers, and non-teaching staff.
• Report Generation Tools – for substitution history and workload summaries in PDF/Excel formats.

These ideas were evaluated for technical feasibility within the chosen stack (React, TypeScript, Node.js, Express, and PostgreSQL).

3. Evaluation and Selection

The ideas were assessed based on feasibility, impact, scalability, and ease of use. The most practical and high-impact features—automated substitution, real-time alerts, and workload analytics—were selected for implementation in the initial version. The modular architecture ensures easy future expansion, such as mobile app integration or AI-based workload prediction.

4. **Concept Development**

Once core ideas were finalized, the team created:

• System flow diagrams outlining user roles and interactions.
• Database schema designs for storing staff data, schedules, and leave records.
• Wireframes and mockups for intuitive, responsive user interfaces.
• Feature prioritization lists to guide phased development.

Tools such as Miro, Figma, and Draw.io were used for collaborative brainstorming and prototype creation. Feedback from mentors and users helped refine the final concept, ensuring it met both administrative and functional requirements effectively.

3.5 PROBLEM–SOLUTION FIT

The Staff Substitution and Duty Assigning System was conceptualized to address the major challenges faced in manual staff management processes within educational institutions. The problem–solution fit ensures that the developed system directly resolves real-world administrative inefficiencies through a user-focused and technically efficient approach.

Identified Problems

1. Manual scheduling and substitution processes causing confusion and time delays.
2. Lack of a centralized platform for managing staff availability and workload.
3. Poor communication between administrators and staff during sudden absences.
4. Difficulty in tracking leave requests and substitution history.
5. Absence of analytical tools and automated reporting for performance monitoring.

Proposed Solutions

1. Centralized Digital Management: The system replaces manual records with a structured PostgreSQL database managed through Prisma ORM for secure and reliable data handling.
2. Automated Substitution Allocation: Automatically assigns substitutes based on subject expertise, workload, and availability, reducing administrative effort.
3. Real-Time Notifications: Alerts staff about duty assignments, substitutions, and schedule changes instantly.
4. Role-Based Access System: Separate dashboards for administrators, teaching staff, and non-teaching staff ensure secure and organized access.
5. Analytical Reports and Workload Tracking: Generates performance summaries and workload distribution reports in PDF/Excel formats for better decision-making.

Fit Analysis

The Staff Substitution and Duty Assigning System demonstrates a strong alignment between identified issues and implemented solutions. Each feature directly addresses a specific challenge, improving coordination, transparency, and operational efficiency within institutions. The system's automation capabilities significantly reduce manual errors and administrative burden, while real-time communication ensures seamless workflow continuity.

Moreover, the solution is scalable and adaptable, suitable for deployment across multiple departments or campuses. Its modular design allows easy integration of future enhancements such as AI-based workload prediction or mobile accessibility, ensuring long-term sustainability and improved institutional management.

3.6 ARCHITECTURE DESIGN:

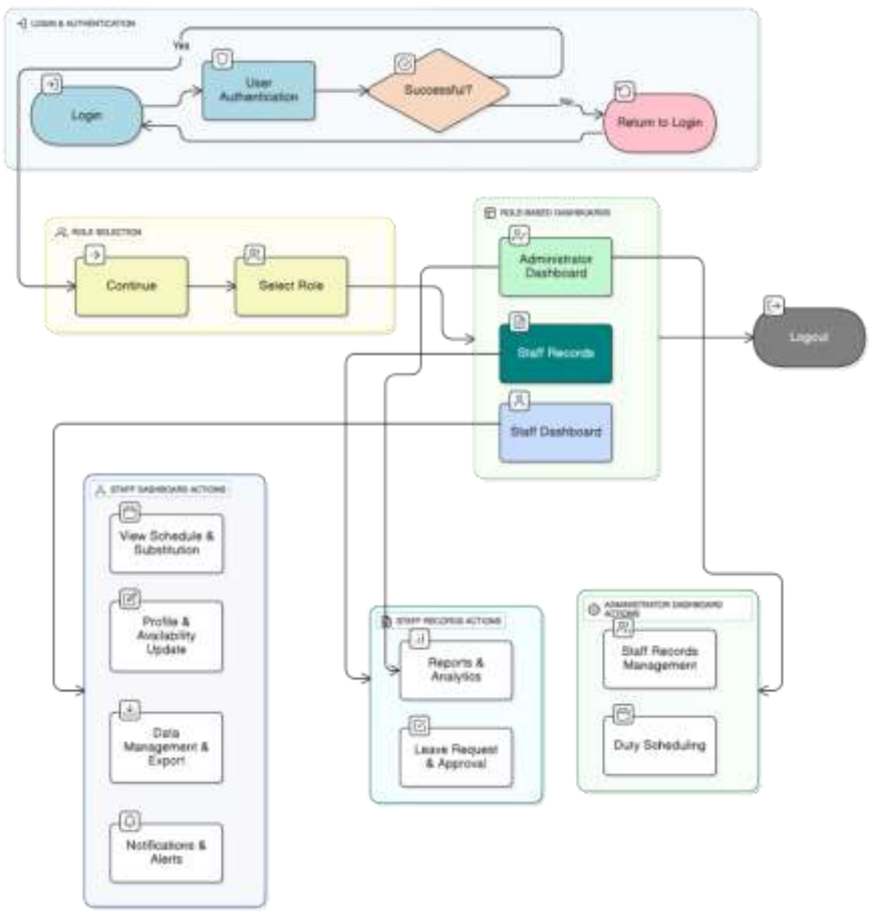


Figure 1: Model Architecture

The figure shown above represents the Solution Architecture that we made use of in our Project.

Project Workflow Diagram

The Project Workflow Diagram represents the complete structure of the Staff Substitution and Duty Assigning System, illustrating how different components interact to ensure efficient staff management. The process begins with the user logging into the system through the frontend interface, which communicates with the backend database to authenticate and manage data securely. The system uses Firebase Authentication or JWT tokens for secure login and access control. After successful authentication, role-based routing directs users—such as administrators, teaching staff, or non-teaching staff—to their respective dashboards. Each role is granted access to specific modules relevant to their responsibilities, ensuring data privacy and operational efficiency.

Administrators can manage staff records, handle leave requests, assign substitute staff, and generate reports for analysis and planning. Teaching staff can view duty schedules, submit leave requests, and acknowledge substitutions in real time.

Non-teaching staff can check assigned tasks and update work status through their dashboards.
This modular and automated workflow promotes centralized control, transparency, and timely communication between staff and administrators. By integrating real-time updates, notifications, and analytics, the system ensures seamless substitution management, balanced workload distribution, and uninterrupted institutional operations.

3.7. DESCRIPTION OF MODULES

3.7.1 ADMIN MODULE

The Admin Module serves as the central control unit of the Staff Substitution and Duty Assigning System. It allows administrators to manage user accounts, assign roles, and maintain department-wise staff data. Through this module, the admin can schedule duties, assign substitute staff during absences, and monitor overall work distribution. The system provides automated suggestions for substitution based on staff availability, workload, and skill compatibility. Additionally, the admin can generate duty rosters, leave reports, and performance summaries in PDF or Excel formats. This module ensures transparency, balanced workload allocation, and seamless coordination among staff members, enhancing institutional efficiency.

3.7.2 STAFF MODULE

The Staff Module enables employees to view their assigned duties, manage their schedules, and request substitutions or leaves through a digital interface. Staff members can receive notifications about new duty assignments, substitution approvals, or changes in scheduling. The module allows them to update their availability status, making it easier for the system to find suitable replacements when needed. This module also provides an overview of daily, weekly, or monthly duty plans and includes a feedback option for reporting issues or updates to the admin. It promotes effective communication, reduces scheduling conflicts, and enhances work transparency.

3.7.3 SUBSTITUTE MANAGEMENT MODULE

The Substitute Management Module handles the automatic or manual allocation of replacement staff when someone applies for leave or becomes unavailable. The module cross-checks staff availability, role suitability, and workload to suggest appropriate substitutes. It also allows administrators to approve or modify these suggestions before finalizing assignments.
The system keeps a record of all substitution activities, ensuring accountability and easy reference for future planning. This module minimizes disruption in workflow and ensures that all duties are covered efficiently, even during unexpected absences.

3.7.4 REPORT AND ANALYTICS MODULE

The Report and Analytics Module provides administrators and management with data-driven insights on staff attendance, duty coverage, and substitution patterns.
It generates detailed visual reports and charts that highlight key statistics such as the number of substitutions per month, staff utilization rates, and department-wise duty completion.
These reports can be exported in various formats (PDF, Excel) for administrative use or audits. The analytics component helps identify workload imbalances, frequently substituted roles, and overall efficiency in duty management, enabling informed decision-making.

3.8 DATAFLOW DIAGRAM:

The Data Flow Diagram (DFD) illustrates how data moves within the **Staff Substitution and Duty Assigning System**, showing the flow of information between users, processes, and the database. It provides a structured overview of how staff duties are assigned, substitutions are handled, and reports are generated. The DFD ensures a clear understanding of the interactions among the **Admin**, **Staff**, **Substitute Module**, and the **Database**, helping in efficient system design and implementation.

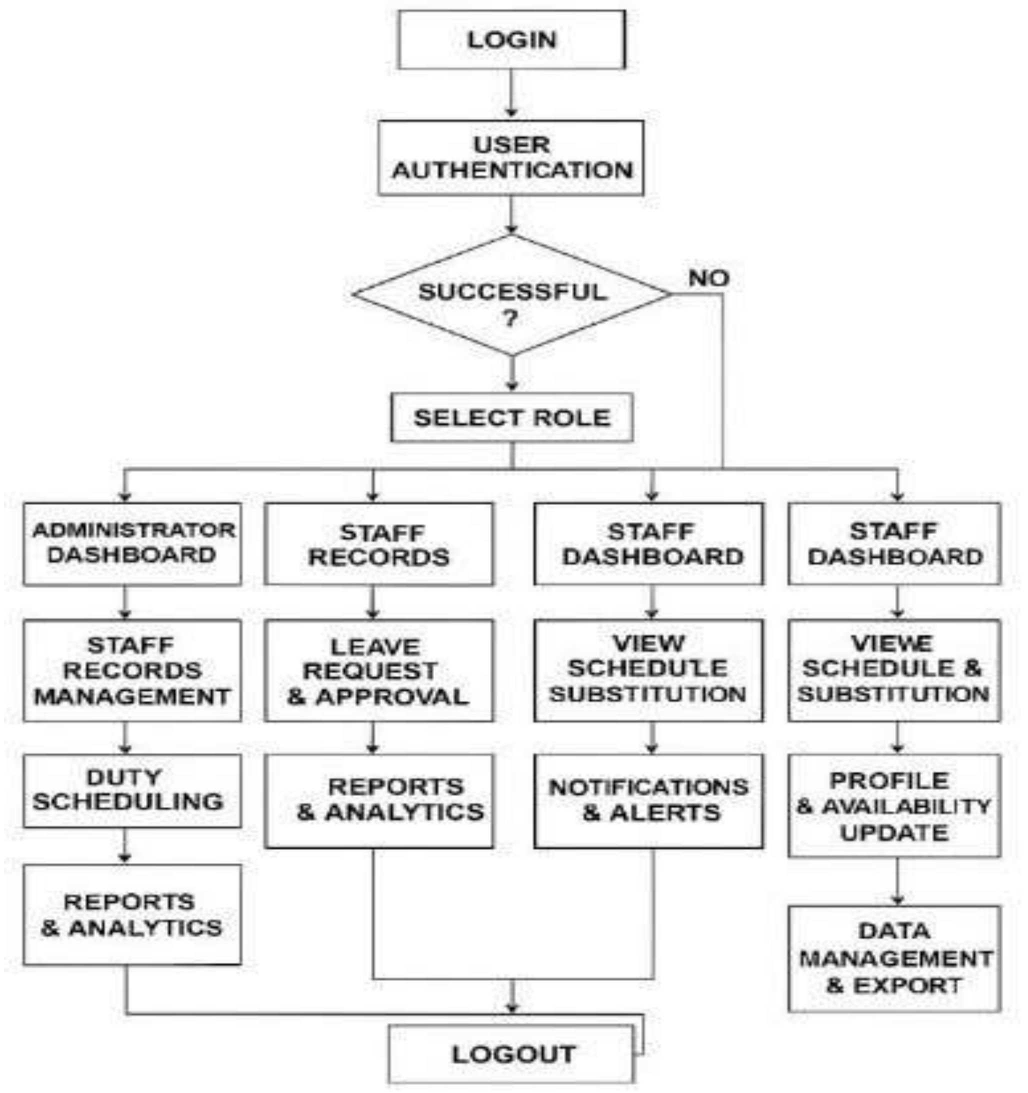


Figure 2: Data Flow Diagram

CHAPTER 4 SYSTEM REQUIREMENTS

4.1HARDWARE REQUIREMENTS

- **SYSTEM:** PC/Laptop with a minimum Intel i3 processor, 4 GB RAM, and at least 20 GB free storage space.
- **DISPLAY:** Minimum resolution of 1366×768 for proper user interface visualization.
- **INTERNET CONNECTION:** Stable internet connectivity required for accessing the web application, REST APIs, and online data synchronization.
- **SERVER (Optional for Deployment):** Hosting environment with Intel i5 or higher processor, 8 GB RAM, and 500 GB storage for production setup.
- **CLIENT DEVICES:** Compatible with desktop, laptop, tablet, or mobile devices supporting modern web browsers such as Chrome, Firefox, or Edge.

4.2 SOFTWARE REQUIREMENTS

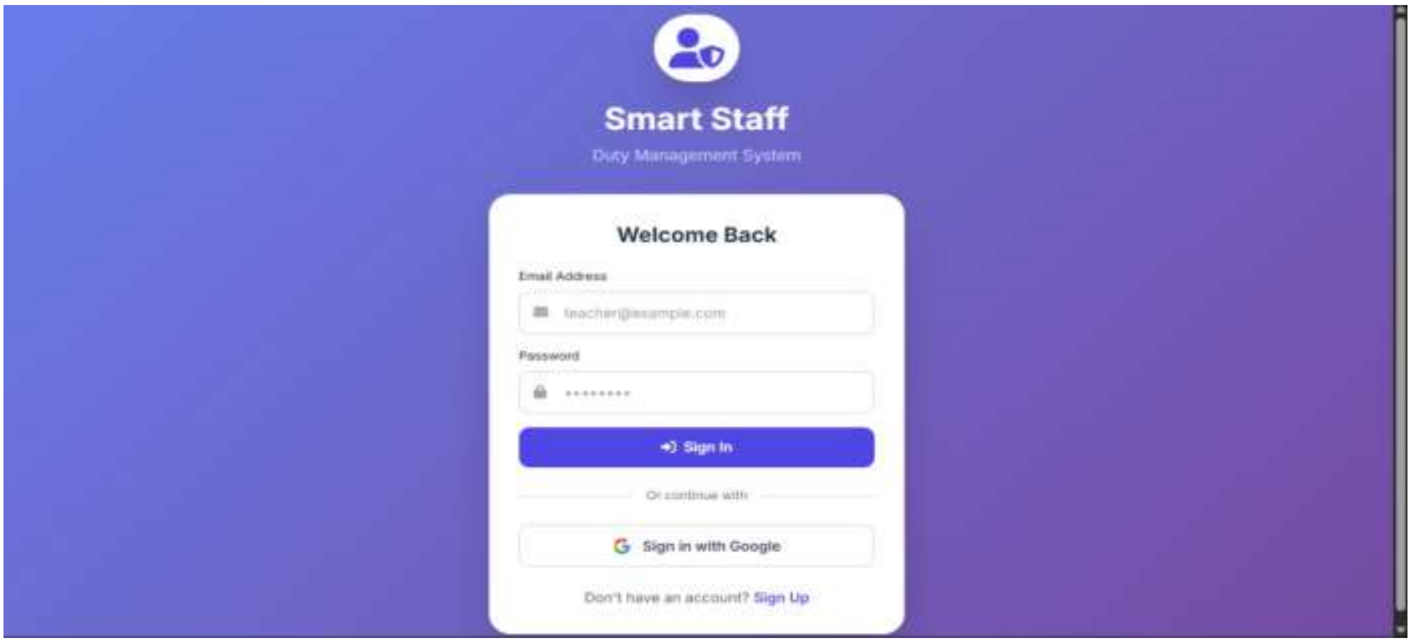
- **OPERATING SYSTEM:** Windows 10 or higher / Linux / macOS.
- **IDE / CODE EDITOR:** Visual Studio Code, Eclipse, or IntelliJ IDEA for development and debugging.
- **FRONTEND TECHNOLOGY:** React.js for building a responsive and interactive user interface.
- **BACKEND TECHNOLOGY:** Java Spring Boot for server-side logic and RESTful API development.
- **DATABASE:** MySQL for managing staff details, duty schedules, and substitution records.
- **WEB BROWSER:** Latest versions of Google Chrome, Mozilla Firefox, or Microsoft Edge for accessing the system.
- **VERSION CONTROL:** Git and GitHub for source code management and collaboration.
- **PACKAGE MANAGEMENT:** npm or Maven for handling project dependencies.
- **UI DESIGN TOOLS:** Figma or Adobe XD for creating wireframes and user interface mockups.
- **REPORTING LIBRARIES:** Jasper Reports or Apache POI for generating duty schedules and substitution reports in PDF/Excel format.
- **NOTIFICATION SERVICE:** Integration with Email or SMS API for sending duty alerts and substitution notifications.
- **TESTING TOOLS:** Postman for API testing and JUnit for backend testing.

CHAPTER 5 IMPLEMENTATION

5.1 LOGIN :

The **Login Page** of the **Staff Substitution and Duty Assigning System** serves as a secure entry point for both administrators and staff members. It allows users to log in using a **registered email and password**, with authentication handled through **JWT** or **Spring Security** for data security.

Upon successful login, users are redirected to their respective dashboards -**Admin Dashboard** for managing duties and substitutions, and **Staff Dashboard** for viewing schedules or applying for leave.The page includes essential features such as **input validation**, **forgot password option**, and **role-based redirection**, ensuring a smooth, secure, and user-friendly login experience.

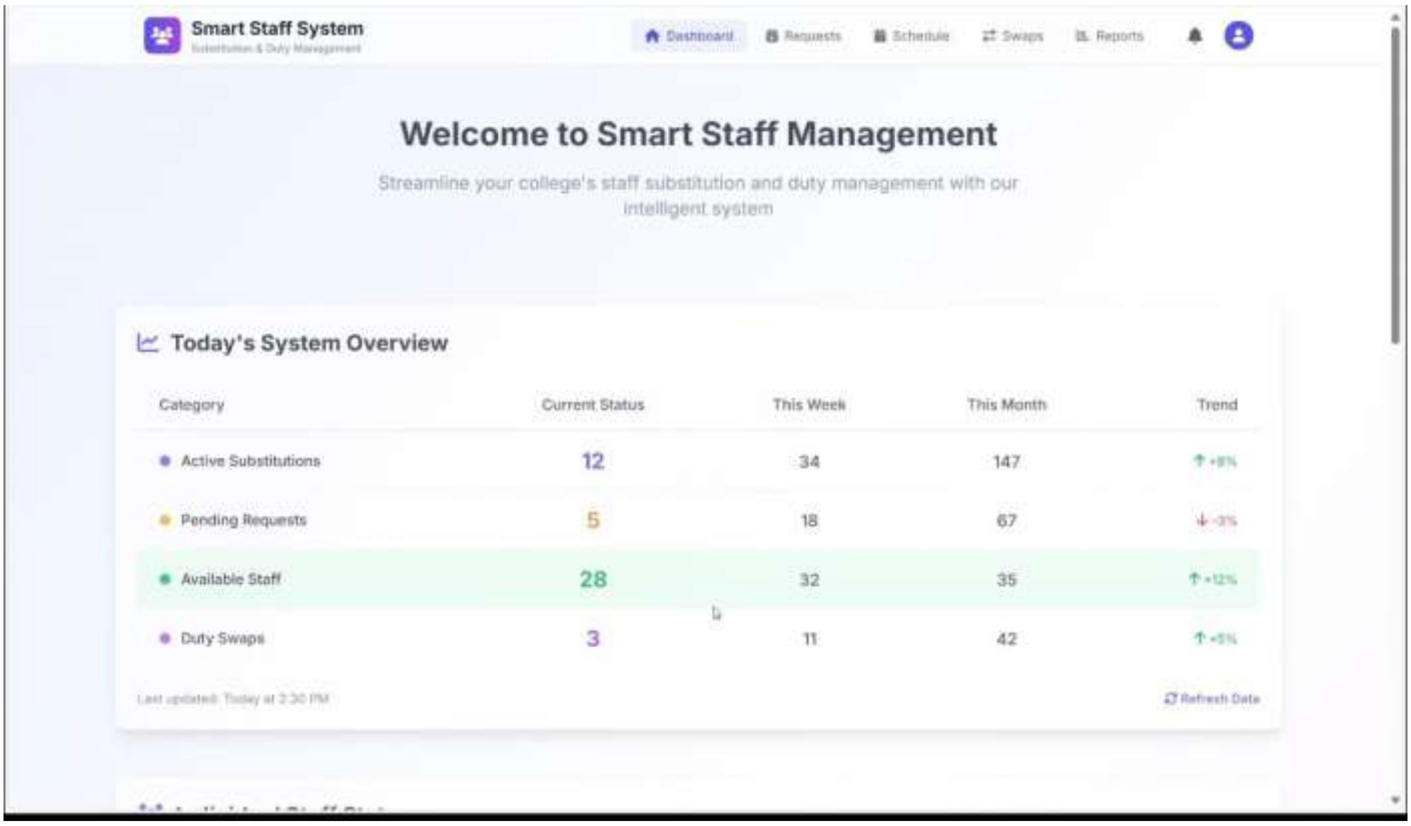


5.2. STAFF MODULE:

The **Staff Module** is a key component of the **Staff Substitution and Duty Assigning System**, designed to help employees manage their duties and substitution requests efficiently. This module provides each staff member with a **secure, role-based dashboard** where they can view assigned duties, submit leave applications, and check substitution status in real time.

Staff can update their **availability**, receive **notifications** for new duty assignments or approved substitutions, and track their schedules through a simple and interactive interface. The system ensures transparency and timely communication by allowing staff to stay informed about changes in their duty roster.

Additional features include access to **personal duty history**, **substitution logs**, and **automated alerts** for upcoming shifts or pending approvals. This module improves coordination, reduces scheduling conflicts, and enhances overall staff management efficiency.



Welcome to Smart Staff Management

Streamline your college's staff substitution and duty management with our intelligent system

Today's System Overview

Category	Current Status	This Week	This Month	Trend
● Active Substitutions	12	34	147	↑ +8%
● Pending Requests	5	18	67	↓ -3%
● Available Staff	28	32	35	↑ +12%
● Duty Swaps	3	11	42	↑ +5%

Last updated: Today at 2:30 PM

Refresh Data

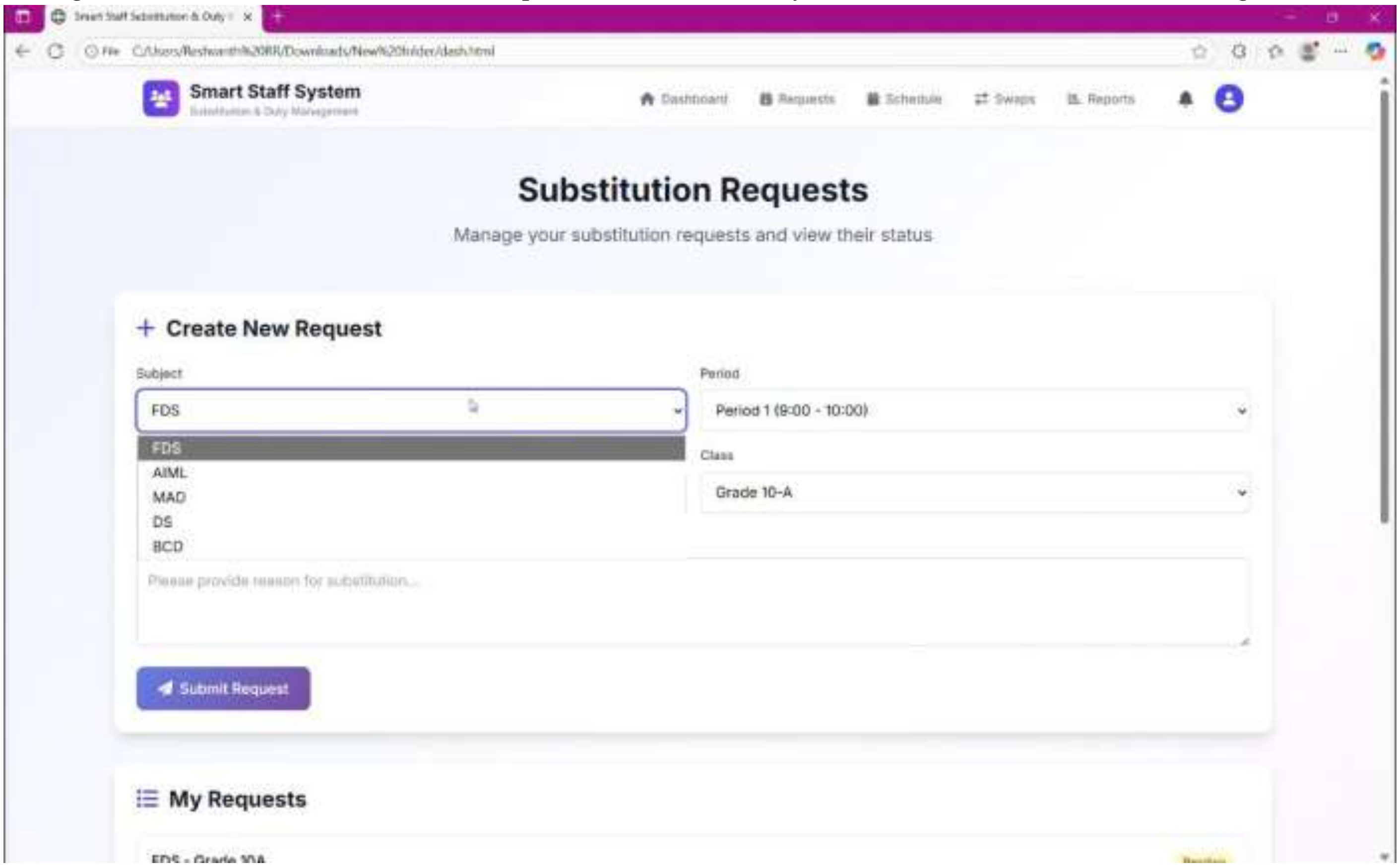
5.3. SUBSTITUTE MANAGEMENT MODULE:

The **Substitute Management Module** is designed to ensure smooth workflow continuity by automatically handling staff substitutions during absences or leaves. Through this module, administrators can view leave requests, check staff availability, and assign suitable substitutes based on department, role, and workload.

The system intelligently matches available staff with pending duties, ensuring that all responsibilities are covered without disruption. Staff members receive **automated notifications** about substitution assignments or approvals, keeping them informed in real time.

This module also maintains a **record of all substitution activities**, allowing the admin to review past substitutions and ensure fair workload distribution. By providing efficient duty reallocation and instant communication, the Substitute

Management Module enhances operational efficiency and minimizes scheduling conflicts.


Smart Staff System
Substitution & Duty Management

Substitution Requests

Manage your substitution requests and view their status

Create New Request

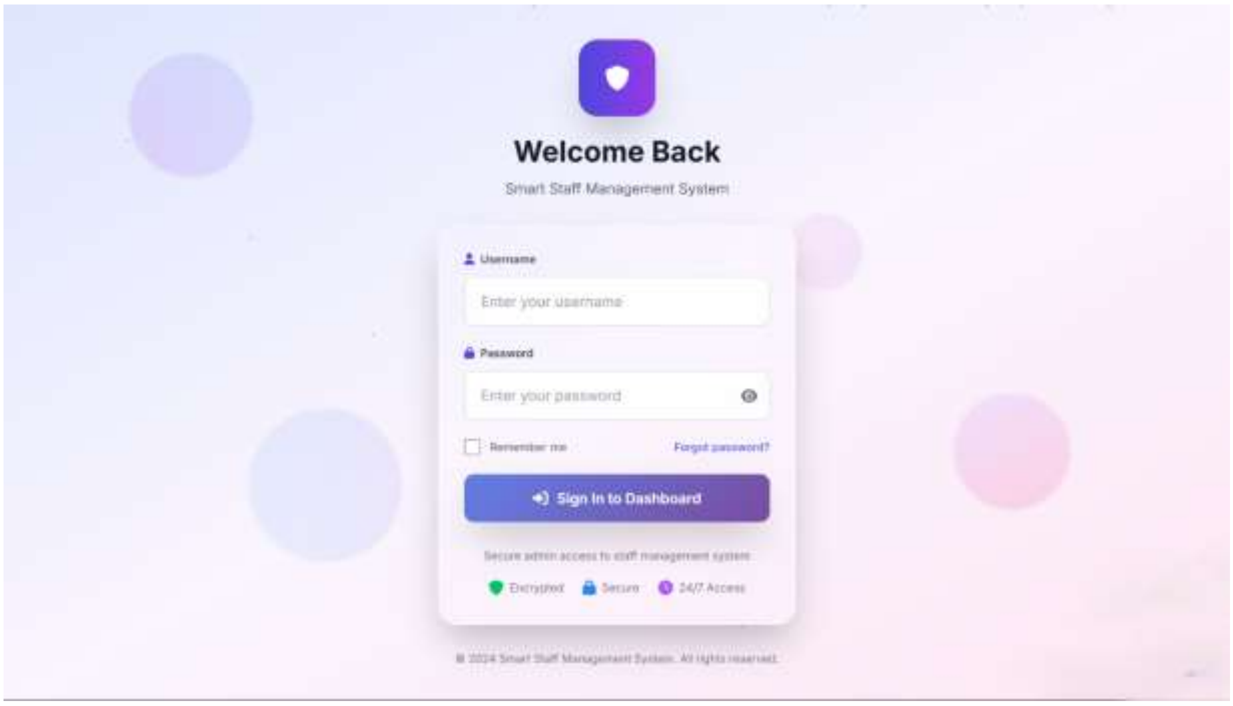
Subject: FDS
Period: Period 1 (9:00 - 10:00)
Class: Grade 10-A

5.4. ADMIN MODULE

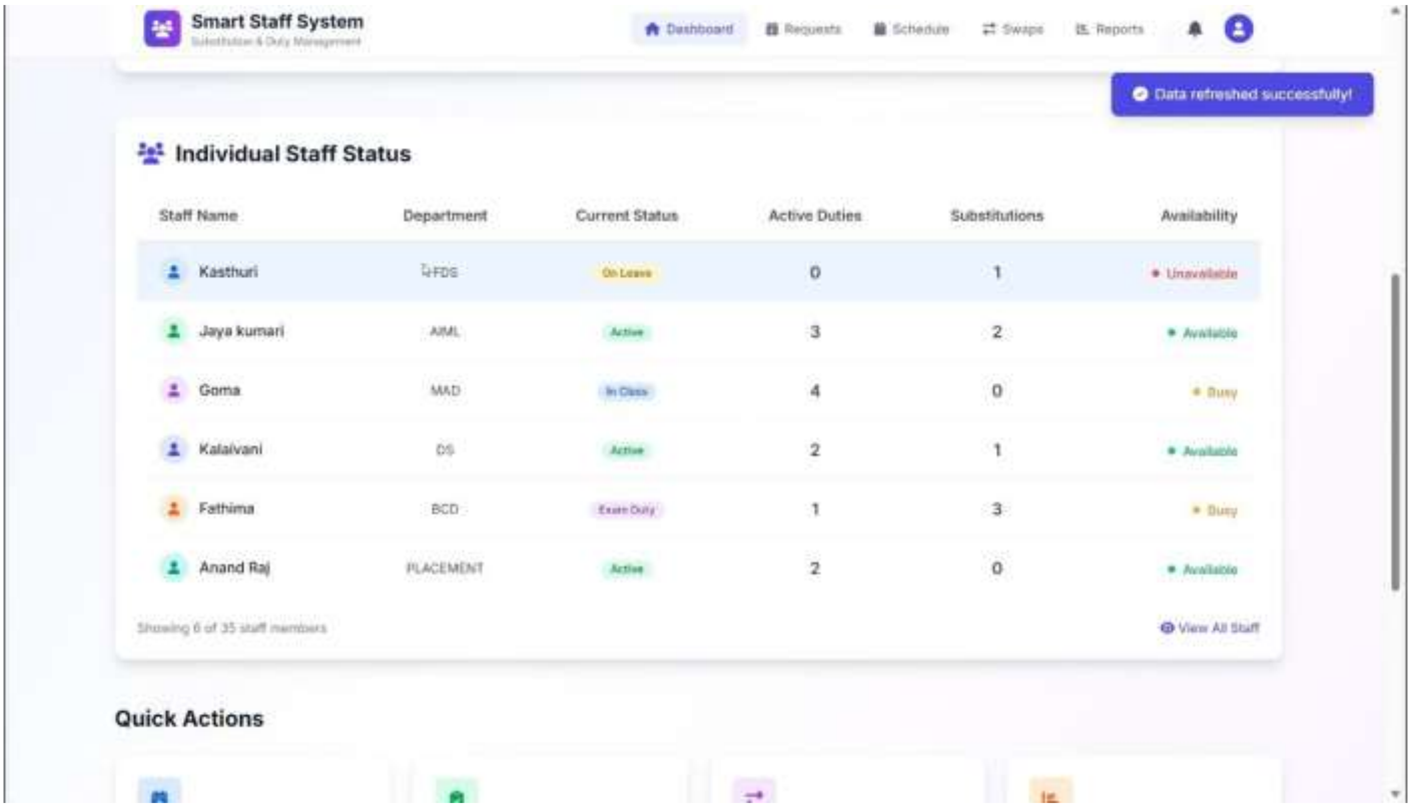
The **Admin Module** is the central component of the **Staff Substitution and Duty Assigning System**, designed to help administrators efficiently manage staff duties, leaves, and substitutions. Through this module, the admin can **assign daily or weekly duties**, **approve or reject leave requests**, and **allocate substitutes** based on staff availability and workload.

The module provides a **dashboard view** displaying duty schedules, pending approvals, and substitution summaries for better monitoring. Admins can also **generate reports** on staff attendance, duty performance, and substitution history in PDF or Excel format.

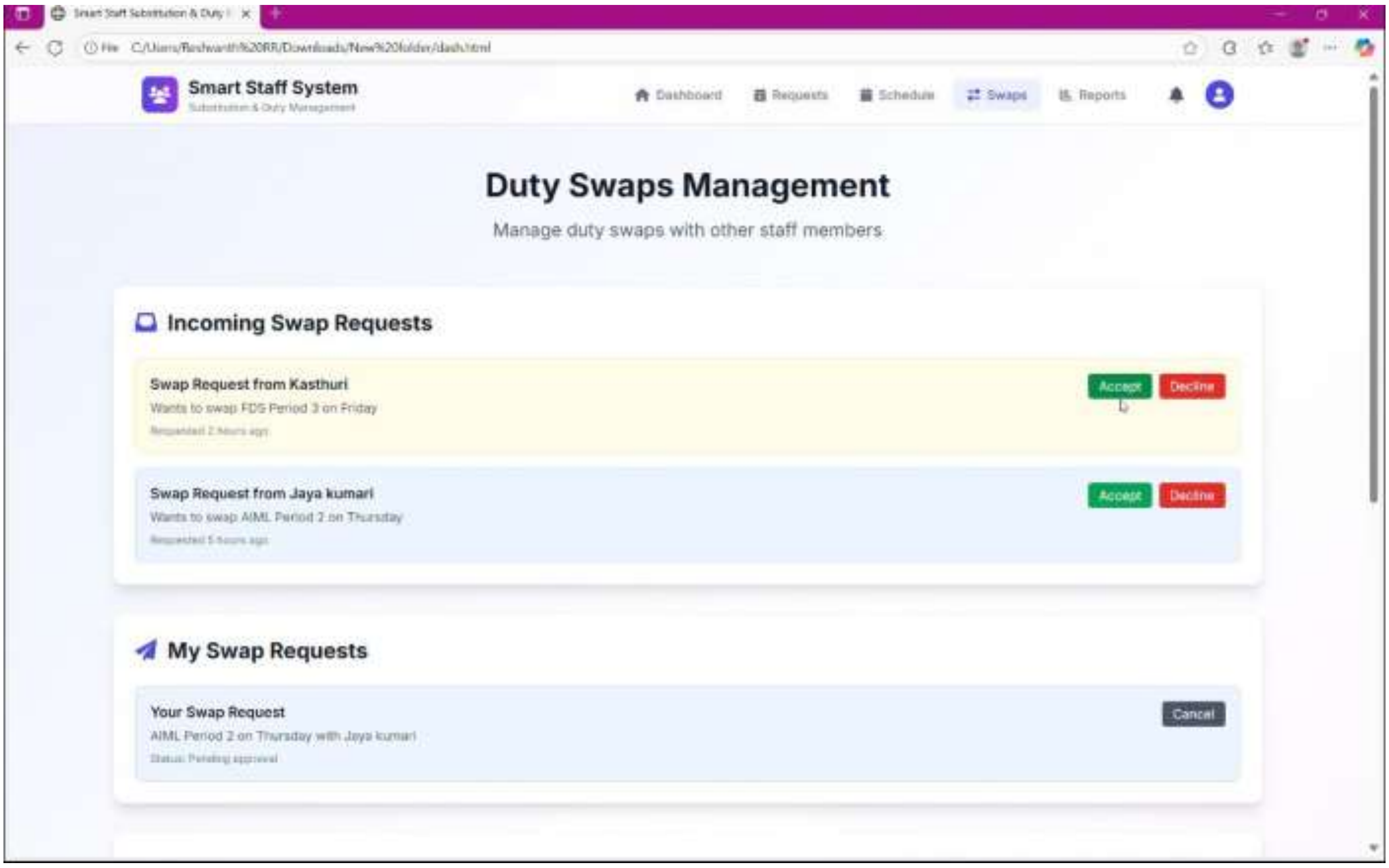
In addition, the system sends **automated notifications** to staff regarding duty assignments, substitutions, and updates. This module ensures transparency, balanced workload distribution, and smooth coordination across departments, improving overall operational efficiency.



ADMIN LOGIN



Staff Name	Department	Current Status	Active Duties	Substitutions	Availability
Kasthuri	FDS	On Leave	0	1	Unavailable
Jaya kumari	AIML	Active	3	2	Available
Goma	MAD	In Class	4	0	Busy
Kalaivani	DS	Active	2	1	Available
Fathima	BCD	Exam Duty	1	3	Busy
Anand Raj	PLACEMENT	Active	2	0	Available


Smart Staff Substitution & Duty
C:/Users/Reshwanth%20RR/Downloads/New%20folder/dash.html
Smart Staff System
Substitution & Duty Management
Dashboard
Requests
Schedule
Swaps
Reports
Accept
Decline
Accept
Decline
Cancel

Duty Swaps Management

Manage duty swaps with other staff members

Incoming Swap Requests

Swap Request from Kasthuri
Wants to swap FDS Period 3 on Friday
Requested 2 hours ago

Swap Request from Jaya kumari
Wants to swap AIML Period 2 on Thursday
Requested 5 hours ago

My Swap Requests

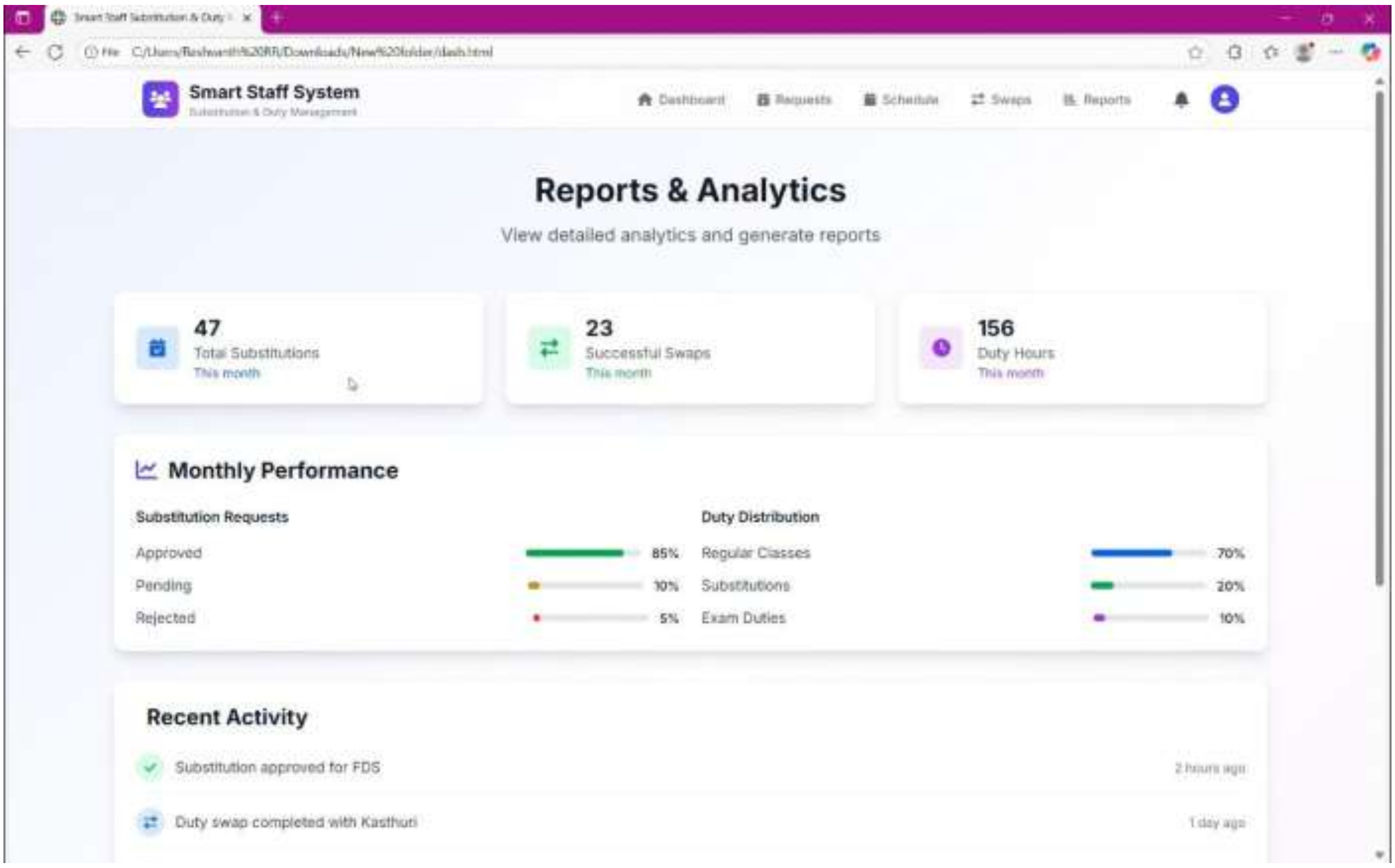
Your Swap Request
AIML Period 2 on Thursday with Jaya kumari
Status: Pending approval

5.5. REPORT AND ANALYTICS MODULE:

The **Report and Analytics Module** provides comprehensive insights into staff performance, duty allocation, attendance, and workload distribution. It helps administrators monitor staff activity, track leave patterns, and evaluate duty efficiency through interactive charts and dashboards.

The module enables automated generation of detailed analytical reports in **PDF or Excel** formats, which can be used for internal reviews or official submissions. It visualizes key metrics such as duty coverage, substitution frequency, shift utilization, and overall workforce performance, allowing management to make data-driven decisions.

By integrating advanced data analytics and visualization tools, this module enhances transparency, optimizes resource allocation, and improves operational efficiency. It ensures that duty assignments are balanced, performance is measurable, and staff productivity trends are easily trackable over time.



Reports & Analytics

View detailed analytics and generate reports

47
Total Substitutions
This month

23
Successful Swaps
This month

156
Duty Hours
This month

Monthly Performance

Substitution Requests		Duty Distribution	
Approved	85%	Regular Classes	70%
Pending	10%	Substitutions	20%
Rejected	5%	Exam Duties	10%

Recent Activity

Substitution approved for FDS 2 hours ago

Duty swap completed with Kasthuri 1 day ago

SOFTWARE DESCRIPTION:

1. VS CODE:

Visual Studio Code (VS Code) serves as the primary Integrated Development Environment (IDE) for the development of the *Staff Substitution and Duty Assigning System*. It is a lightweight, open-source, and highly efficient code editor that supports multiple programming languages and frameworks. VS Code enables seamless development of both frontend and backend components in a single environment, offering features like syntax highlighting, IntelliSense, debugging tools, and intelligent code completion to enhance productivity and minimize errors.

The IDE's integrated debugging tools allow developers to test, trace, and fix errors efficiently, making it ideal for full-stack projects. Extensions like **Prettier**, **ESLint**, and **Live Server** help maintain code consistency, style, and provide instant frontend previews. With **Git integration**, developers can easily track code changes, collaborate, and manage version control directly within the editor. VS Code's support for Docker, Remote Development, and cloud extensions allows the system to be tested in multiple environments, ensuring smooth deployment and scalability. Its flexibility, extensibility, and ease of use make it ideal for continuous improvement and maintenance of the system.

2. FRONTEND:

The frontend of the *Staff Substitution and Duty Assigning System* is developed using **React.js**, providing a responsive, dynamic, and user-friendly interface. React's component-based architecture allows modular development, ensuring faster updates and improved maintainability. The interface provides different dashboards for administrators and PHC staff, displaying duty schedules, substitution requests, staff availability, and notifications.

Interactive components like **calendars, alert boxes, and charts** provide real-time insights into staff assignments and substitution activities. The frontend is fully responsive, allowing access from desktops, tablets, and smartphones—making it convenient for on-site PHC personnel. Emphasis is placed on a **clean and intuitive UI**, ensuring users can easily navigate, view duties, request substitutions, and manage schedules without technical difficulties.

Secure communication with the backend is achieved through **RESTful APIs**, ensuring real-time updates and data synchronization. The frontend's design enhances accessibility, usability, and performance, supporting smooth and efficient management of staff duties in healthcare center.

3. BACKEND:

The backend of the system is built using **Node.js** with the **Express** framework, ensuring high performance, scalability, and security. Node.js handles multiple requests efficiently, making it suitable for real-time duty allocation and substitution processes. Express provides robust routing and middleware support, forming the backbone for API development and secure data exchange between frontend and database.

The backend implements **JWT-based authentication** and **role-based access control**, ensuring that administrators and staff access only their authorized functions. It manages all operations related to duty assignment, leave requests, substitution approval, notifications, and report generation. Advanced error handling, input validation, and secure session management help maintain data integrity and system reliability.

Furthermore, the backend is designed to be modular, allowing easy integration of future enhancements like AI-based staff allocation, workload analytics, or cloud synchronization. It ensures smooth communication with the database and efficient handling of concurrent requests, providing a reliable foundation for the entire system.

4. DATABASE:

The system uses **PostgreSQL** as its primary database, managed through **Prisma ORM** for efficient and secure data handling. PostgreSQL ensures reliable storage of critical data such as staff details, duty rosters, leave records, and substitution logs. Its relational structure maintains data integrity, supports complex queries, and reduces redundancy through normalization.

Prisma ORM simplifies database operations, offering an abstraction layer that allows developers to perform CRUD operations with ease and safety. It ensures smooth interaction between the backend and database, enabling quick retrieval and updates of staff records, schedules, and reports.

The database supports analytics and reporting, allowing administrators to generate PDF or Excel summaries of duty schedules, staff attendance, and substitution history. Security features like **role-based access, encryption, and periodic backups** protect sensitive personnel data. Scalable and robust, the PostgreSQL database ensures efficient data management and high system performance, forming the backbone of the *Staff Substitution and Duty Assigning System*.

5.6. CODE IMPLEMENTATION

Step 1: Set up the Authentication System

The Authentication System in the Staff Substitution and Duty Assigning System ensures secure Login for Administrators and Staff using JWT authentication and bcrypt password hashing. It validates user credentials and provides authorized access to system features like duty assignment and substitution.

Code Snippet:

```
const [email, setEmail] = useState('');
const [password, setPassword] = useState('');
const [showPassword, setShowPassword] = useState(false);

const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    await login(email, password);
    console.log('Login Successful');
  } catch {
    alert('Login Failed');
  }
};

<Input type="email" placeholder="Email" value={email}
  onChange={(e) => setEmail(e.target.value)} required />

<Input type={showPassword ? "text" : "password"}
                    placeholder="Password"
  value={password} onChange={(e) =>
                    setPassword(e.target.value)} required />

<button onClick={() => setShowPassword(!showPassword)}>
  {showPassword ? <EyeOff /> : <Eye />}
</button>
```

Step 2: Set up the Role-Based Dashboards

```
const [data, setData] = useState({ duties: 0, staff: 0 });

useEffect(() => {
  fetch(`${api}/api/dashboard`, {
    headers: { Authorization: `Bearer ${token}` }
  })
    .then(res => res.json())
    .then(res => setData(res.data));
}, []);

<DashboardCard title="Total Staff" value={data.staff} />
<DashboardCard title="Assigned Duties" value={data.duties} />  onClick={()
=> navigate('/admin/substitutions')}
/>
```

The **Role-Based Dashboard** in the *Staff Substitution and Duty Assigning System* displays key details like **assigned duties**, **staff count**, and **pending substitutions** based on the user's role. It helps administrators and staff monitor and manage tasks efficiently.

Step 3: Set up the Duty and Substitution Tracking System

The **Duty and Substitution Tracking Module** in the *Staff Substitution and Duty Assigning System* allows administrators and staff to view assigned duties, track substitutions, and monitor schedule updates in real time. Administrators can easily see who is on duty, identify pending substitution requests, and approve replacements when necessary. Staff can view their daily duties, upcoming shifts, and substitution status.

Code Snippet:

```
  const [duties, setDuties] = useState([]);

useEffect(() => {
 const fetchDuties = async () => {
  const response = await fetch(`${api}/api/staff/duties`, {
   headers: { Authorization: `Bearer ${token}` }
  });
  const { data } = await response.json();
  setDuties(data);
 };
 fetchDuties();
}, []);

{duties.map((duty) => (
 <div key={duty.id}>
  <h3>{duty.staffName}</h3>
  <p>Duty: {duty.task}</p>
  <p>Date: {new Date(duty.date).toLocaleDateString()}</p>
  <Badge>{duty.status}</Badge>
 </div>
))}
```

Step 4: Set up the Duty Tracking System

The **Duty Tracking Module** in the *Staff Substitution and Duty Assigning System* allows staff to view their assigned duties, completed tasks, and upcoming schedules. Administrators can track duty progress, monitor staff attendance, and review substitution details.

Code Snippet:

```
const [visits, setVisits] = useState([]);

useEffect(() => {
  const fetchVisits = async () => {
    const response = await fetch(`${api}/api/mother/visits`, {
      headers: { 'Authorization': `Bearer ${token}` }
    });
    const { data } = await response.json();
    setVisits(data);
  };
  fetchVisits();
}, []);

{visits.map((visit) => (
  <Card key={visit.id}>
    <p>Visit {visit.visitNumber} - Week {visit.week}</p>
    <p>Date: {new Date(visit.date).toLocaleDateString()}</p>
    <p>BP: {visit.bp}</p>
    <p>Weight: {visit.weight} kg</p>
    <p>Hemoglobin: {visit.hemoglobin} g/dL</p>
  </Card>
))}
```

Step 5 Set up the Duty Assignment Management System

The **Duty Assignment Management Module** in the *Staff Substitution and Duty Assigning System* allows administrators to assign duties, set dates, and manage staff schedules. Staff members can view assigned duties and substitutions in real time.

Code Snippet:

```
const [dutyData, setDutyData] = useState({
  staffName: '', duty: '', date: '', shift: ''
});

const handleAssignDuty = async (e) => {
  e.preventDefault();
  await fetch(`${api}/api/admin/assign-duty`, {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(dutyData)
  });
};

<Input
  placeholder="Staff Name"
  value={dutyData.staffName}
  onChange={(e) => setDutyData({...dutyData, staffName: e.target.value})}
/>

<Button onClick={handleAssignDuty}>Assign Duty</Button>
```

5.7. RESULT:

The **Staff Substitution and Duty Assigning System** successfully demonstrates a fully functional web-based platform for managing staff duties and substitutions in healthcare or institutional environments. The system, developed using **React, Node.js, Express, and MongoDB**, achieves its objectives by automating **duty scheduling**, **substitution handling**, and **attendance tracking**. It provides **role-based dashboards** for **Administrators** and **Staff Members**, ensuring seamless coordination, reduced manual errors, and efficient management of staff availability and assignments.

Functional Results:

- **Administrator Dashboard:** Administrators can create and manage staff profiles, assign duties, approve substitution requests, and monitor attendance. The dashboard provides real-time updates, duty statistics, and reports, ensuring smooth staff coordination and minimal scheduling conflicts.

- **Staff Dashboard:** Staff members can view their assigned duties, request substitutions when unavailable, and track duty history. They receive instant notifications for new assignments or approved substitutions, improving communication and task clarity.

- **Substitution Management:** The system enables quick and transparent substitution handling, where admins can approve or reject requests based on staff availability. Automated updates ensure accurate duty allocation without manual intervention.

- **Reports and Analytics:** The reporting module generates daily and monthly summaries of duty assignments, substitutions, and attendance patterns. Graphical insights assist management in optimizing staff utilization and ensuring operational efficiency.

Performance and Usability Results:

• The system is fully responsive, ensuring smooth access on desktops, tablets, and mobile devices, enabling easy usage by administrators and staff across different work environments.

• Secure authentication using JWT guarantees that staff data and duty records remain protected, with restricted access based on user roles to maintain confidentiality and integrity.

• Real-time notifications alert staff about new duty assignments, substitutions, and schedule updates, improving communication and reducing manual coordination delays.

• The centralized database efficiently manages staff information, duty rosters, and substitution history, ensuring accurate and quick data retrieval for reporting and analysis.

• Interactive dashboards and visual analytics assist management in tracking duty distribution, workload balance, and staff performance for better decision-making.

• Overall, the system enhances operational efficiency, minimizes scheduling conflicts, and ensures seamless staff coordination, achieving the project’s goal of streamlining substitution and duty management.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION:

The *Staff Substitution and Duty Assigning System* successfully demonstrates the development of an efficient and automated platform for managing staff schedules, substitutions, and workload distribution within an organization or institution. By leveraging modern web technologies such as React, Node.js, Express, and MongoDB, the system ensures secure, responsive, and user-friendly operation for administrators and staff members alike.

The system simplifies the traditionally manual and time-consuming process of assigning duties, managing substitutions, and maintaining attendance records. With features like real-time notifications, role-based dashboards, substitution request handling, and automated reporting, it enhances coordination, reduces scheduling conflicts, and ensures smooth day-to-day operations.

Through centralized data management and analytics, administrators can monitor staff availability, duty allocation efficiency, and substitution frequency, leading to better planning and improved productivity. The platform also ensures transparency by allowing staff to view their assigned duties, request changes, and track approvals in real time.

In conclusion, the *Staff Substitution and Duty Assigning System* serves as a reliable and scalable solution that promotes digital transformation in workforce management. It enhances operational efficiency, improves communication between administrators and staff, and ensures seamless substitution handling, making it a valuable tool for educational institutions, healthcare centers, and other organizations.

6.2 FUTURE SCOPE

The *Staff Substitution and Duty Assigning System* holds strong potential for future development and scalability to enhance workforce management efficiency across various sectors such as healthcare, education, and government institutions. In the future, the system can be extended into **mobile applications** for Android and iOS platforms, enabling real-time duty notifications, substitution alerts, and attendance updates, even in remote or on-field environments.

Integration with **biometric systems** and **IoT-enabled attendance devices** can automate staff check-ins and ensure real-time synchronization of duty assignments with actual availability. The inclusion of **AI and machine learning algorithms** can enable intelligent duty scheduling, predicting potential shortages, and automatically suggesting suitable substitutes based on experience, workload, and previous duty patterns.

Further advancements may include **cloud-based deployment** for scalability and multi-branch access, **multi-language support** for broader usability, and **data analytics dashboards** that visualize duty trends, performance metrics, and substitution frequency for better decision-making. Additionally, integration with **HR and payroll systems** can streamline staff compensation and performance evaluation.

In the long term, the platform can evolve into a **comprehensive workforce management ecosystem**, supporting large organizations with seamless coordination, predictive analytics, and improved transparency. These enhancements will make the *Staff Substitution and Duty Assigning System* a robust, intelligent, and adaptive solution for modern institutional management and operational efficiency.

APPENDICES

SOURCE CODE

dash.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Smart Staff Substitution & Duty Management System</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <script src="https://unpkg.com/framer-motion@10.16.4/dist/framer-motion.js"></script>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" rel="stylesheet">
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap');
    body { font-family: 'Inter', sans-serif; }

    .fade-in {
```

```
  animation: fadeIn 0.6s ease-out forwards;
  opacity: 0;
}

.slide-up {
  animation: slideUp 0.8s ease-out forwards;
  opacity: 0;
  transform: translateY(30px);
}

.scale-in {
  animation: scaleIn 0.5s ease-out forwards;
  opacity: 0;
  transform: scale(0.9);
}

@keyframes fadeIn {
  to { opacity: 1; }
}

@keyframes slideUp {
  to { opacity: 1; transform: translateY(0); }
}

@keyframes scaleIn {
  to { opacity: 1; transform: scale(1); }
}

.stagger-1 { animation-delay: 0.1s; }
.stagger-2 { animation-delay: 0.2s; }
.stagger-3 { animation-delay: 0.3s; }
.stagger-4 { animation-delay: 0.4s; }

.card-hover {
  transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
}

.card-hover:hover {
  transform: translateY(-8px);
  box-shadow: 0 25px 50px -12px rgba(0, 0, 0, 0.25);
}

.btn-primary {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  transition: all 0.3s ease;
}

.btn-primary:hover {
  transform: translateY(-2px);
  box-shadow: 0 10px 25px rgba(102, 126, 234, 0.4);
}
```

```
        .glass-effect {
          background: rgba(255, 255, 255, 0.1);
          backdrop-filter: blur(10px);
          border: 1px solid rgba(255, 255, 255, 0.2);
        }

        .nav-tab {
          color: #6b7280;
        }

        .nav-tab:hover {
          color: #4f46e5;
          background-color: #f3f4f6;
        }

        .nav-tab.active {
          color: #4f46e5;
          background-color: #eef2ff;
        }

        .tab-content {
          display: none;
        }

        .tab-content.active {
          display: block;
        }
    </style>
</head>
<body class="bg-gradient-to-br from-indigo-50 via-white to-purple-50 min-h-screen">
    <!-- Navigation -->
    <nav class="bg-white/80 backdrop-blur-md border-b border-gray-200 sticky top-0 z-50 fade-in">
        <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
            <div class="flex justify-between items-center h-16">
                <div class="flex items-center space-x-3">
                    <div class="w-10 h-10 bg-gradient-to-r from-indigo-600 to-purple-600 rounded-lg flex items-center justify-
center">
                        <i class="fas fa-users text-white text-lg"></i>
                    </div>
                    <div>
                        <h1 class="text-xl font-bold text-gray-900">Smart Staff System</h1>
                        <p class="text-xs text-gray-500">Substitution & Duty Management</p>
                    </div>
                </div>
                <div class="flex items-center space-x-6">
                    <div class="hidden md:flex space-x-1">
                        <button onclick="showTab('dashboard')" class="nav-tab active px-4 py-2 text-sm font-medium rounded-
lg transition-colors">
                            <i class="fas fa-home mr-2"></i>Dashboard
                        </button>
                        <button onclick="showTab('requests')" class="nav-tab px-4 py-2 text-sm font-medium rounded-lg
transition-colors">
                            <i class="fas fa-calendar-plus mr-2"></i>Requests
                        </button>
                        <button onclick="showTab('schedule')" class="nav-tab px-4 py-2 text-sm font-medium rounded-lg
transition-colors">
                            <i class="fas fa-calendar mr-2"></i>Schedule
                        </button>
                        <button onclick="showTab('swaps')" class="nav-tab px-4 py-2 text-sm font-medium rounded-lg
transition-colors">
                            <i class="fas fa-exchange-alt mr-2"></i>Swaps
                        </button>
                        <button onclick="showTab('reports')" class="nav-tab px-4 py-2 text-sm font-medium rounded-lg
transition-colors">
                            <i class="fas fa-chart-bar mr-2"></i>Reports
                        </button>
                    </div>
                    <div class="flex items-center space-x-4">
                        <button class="p-2 text-gray-600 hover:text-indigo-600 transition-colors">
                            <i class="fas fa-bell text-lg"></i>
                        </button>
                        <div class="w-8 h-8 bg-indigo-600 rounded-full flex items-center justify-center">
                            <i class="fas fa-user text-white text-sm"></i>
                        </div>
                    </div>
                </div>
          </div>
        </div>
    </nav>

    <!-- Dashboard Tab -->
    <div id="dashboard" class="tab-content active">
        <!-- Hero Section -->
        <section class="py-12 px-4 sm:px-6 lg:px-8">
          <div class="max-w-7xl mx-auto">
                <div class="text-center mb-12">
                    <h2 class="text-4xl font-bold text-gray-900 mb-4 slide-up">Welcome to Smart Staff Management</h2>
                    <p class="text-xl text-gray-600 max-w-3xl mx-auto slide-up stagger-1">Streamline your college's staff
substitution and duty management with our intelligent system</p>
                </div>
          </div>
        </section>
       function changeWeek(direction) {
          currentWeekIndex += direction;
          if (currentWeekIndex < 0) currentWeekIndex = weeks.length - 1;
          if (currentWeekIndex >= weeks.length) currentWeekIndex = 0;

          updateScheduleDisplay();
        }

        function updateScheduleDisplay() {
          const currentWeek = weeks[currentWeekIndex];
          document.getElementById('currentWeek').textContent = currentWeek.title;

          // Update dates in header
          const dateHeaders = document.querySelectorAll('#scheduleGrid').length > 0 ?
                document.querySelectorAll('.grid.grid-cols-8.gap-2.mb-2 > div') : [];

          if (dateHeaders.length > 1) {
                for (let i = 1; i < 8; i++) {
                    const dayNames = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'];
                    dateHeaders[i].innerHTML = ${dayNames[i-1]}<br><span class="text-xs font-
            normal">${currentWeek.dates[i-1]}</span>;
                }
          }

          // Update schedule grid
          const scheduleGrid = document.getElementById('scheduleGrid');
          const timeSlots = ['9:00 AM', '10:00 AM', '11:00 AM', '1:00 PM', '2:00 PM', '3:00 PM'];

          scheduleGrid.innerHTML = '';

          currentWeek.schedule.forEach((timeSlot, timeIndex) => {
                const rowDiv = document.createElement('div');
                rowDiv.className = 'grid grid-cols-8 gap-2';

                // Time label
                const timeLabel = document.createElement('div');
                timeLabel.className = 'bg-gray-100 p-2 rounded text-center text-sm font-medium text-gray-700';
                timeLabel.textContent = timeSlots[timeIndex];
                rowDiv.appendChild(timeLabel);

                // Add lunch break row after 11:00 AM
                if (timeIndex === 2) {
                    scheduleGrid.appendChild(rowDiv);

                    // Lunch break row
                    const lunchRow = document.createElement('div');
                    lunchRow.className = 'grid grid-cols-8 gap-2';

                    const lunchTimeLabel = document.createElement('div');
                    lunchTimeLabel.className = 'bg-gray-100 p-2 rounded text-center text-sm font-medium text-gray-700';
                    lunchTimeLabel.textContent = '12:00 PM';
                    lunchRow.appendChild(lunchTimeLabel);

                    for (let i = 0; i < 7; i++) {
                        const lunchCell = document.createElement('div');
                        if (i < 5) {
                          lunchCell.className = 'bg-gray-100 border border-gray-300 rounded p-2 min-h-[60px] flex items-
            center justify-center';
                          lunchCell.innerHTML = '<div class="text-xs text-gray-600 text-center">Lunch Break</div>';
                        } else {
                          lunchCell.className = 'bg-gray-50 border border-gray-200 rounded p-2 min-h-[60px]';
                        }
                        lunchRow.appendChild(lunchCell);
                    }

                    scheduleGrid.appendChild(lunchRow);

                    // Create new row for current time slot
                    rowDiv = document.createElement('div');
                    rowDiv.className = 'grid grid-cols-8 gap-2';

                    const newTimeLabel = document.createElement('div');
                    newTimeLabel.className = 'bg-gray-100 p-2 rounded text-center text-sm font-medium text-gray-700';
                    newTimeLabel.textContent = timeSlots[timeIndex];
                    rowDiv.appendChild(newTimeLabel);
                }

                // Day cells
                timeSlot.forEach(classInfo => {
                    const cellDiv = document.createElement('div');

                    if (classInfo) {
                        cellDiv.className = bg-${classInfo.color}-50 border border-${classInfo.color}-200 rounded p-2 min-h-
            [60px];
                        cellDiv.innerHTML = `
                          <div class="text-xs font-medium text-${classInfo.color}-800">${classInfo.class}</div>
                          <div class="text-xs text-${classInfo.color}-600">${classInfo.room}</div>
                          <div class="text-xs text-${classInfo.color}-500">${classInfo.grade}</div>
                        `;
                    } else {
                        cellDiv.className = 'bg-gray-50 border border-gray-200 rounded p-2 min-h-[60px]';
                    }

                    rowDiv.appendChild(cellDiv);
                });

                scheduleGrid.appendChild(rowDiv);
          });
            }

            // Navigation and functionality
            document.addEventListener('DOMContentLoaded', function() {
          // Button click handlers
          const buttons = document.querySelectorAll('button');
          buttons.forEach(button => {
                button.addEventListener('click', function(e) {
                    // Skip if it's a tab navigation button
                    if (this.classList.contains('nav-tab')) {
                        return;
                    }
                    e.preventDefault();

                    // Add click animation
                    this.style.transform = 'scale(0.95)';
                    setTimeout(() => {
                      this.style.transform = '';
                    }, 150);

                    const buttonText = this.textContent.trim();

                    // Handle different button actions
                    switch(buttonText) {
                      case 'Submit Request':
                      case 'Send Swap Request':
                        showNotification('Request submitted successfully!');
                        break;
                      case 'Accept':
                        showNotification('Swap request accepted!');
                        this.parentElement.parentElement.parentElement.style.display = 'none';
                        break;
                      case 'Decline':
                        showNotification('Swap request declined.');
                        this.parentElement.parentElement.parentElement.style.display = 'none';
                        break;
                      case 'Cancel':
                        showNotification('Request cancelled.');
                        this.parentElement.parentElement.style.display = 'none';
                        break;
                      case 'Refresh Data':
                        showNotification('Data refreshed successfully!');
                        break;
                      case 'View All Staff':
                        showNotification('Loading all staff members...');
                        break;
                      default:
                        break;
                    }
                  });
                });

                // Notification system
                function showNotification(message) {
                  const notification = document.createElement('div');
                  notification.className = 'fixed top-20 right-4 bg-indigo-600 text-white px-6 py-3 rounded-lg shadow-lg z-50
            transform translate-x-full transition-transform duration-300';
                  notification.innerHTML = <i class="fas fa-check-circle mr-2"></i>${message};
                  document.body.appendChild(notification);

                  setTimeout(() => {
                    notification.style.transform = 'translateX(0)';
                  }, 100);
```

```
                setTimeout(() => {
                    notification.style.transform = 'translateX(full)';
                    setTimeout(() => {
                        if (document.body.contains(notification)) {
                            document.body.removeChild(notification);
                        }
                    }, 300);
                }, 3000);
            }

            // Add hover effects to cards
            const cards = document.querySelectorAll('.card-hover');
            cards.forEach(card => {
                card.addEventListener('mouseenter', function() {
                    this.style.transform = 'translateY(-8px)';
                });

                card.addEventListener('mouseleave', function() {
                    this.style.transform = 'translateY(0)';
                });
            });

            // Make notification function globally available
            window.showNotification = showNotification;
        });
    </script>
<script>(function(){function c(){var b=a.contentDocument||a.contentWindow.document;if(b){var
d=b.createElement('script');d.innerHTML="window.__CF$cv$params={r:'974e52cb86ee9374',t:'MTc1NjE1ODgxOS4
wMDAwMDA='};var a=document.createElement('script');a.nonce='';a.src='/cdn-cgi/challenge-
platform/scripts/jsd/main.js';document.getElementsByTagName('head')[0].appendChild(a);";b.getElementsByTagName(
'head')[0].appendChild(d)}}if(document.body){var
a=document.createElement('iframe');a.height=1;a.width=1;a.style.position='absolute';a.style.top=0;a.style.left=0;a.style.
border='none';a.style.visibility='hidden';document.body.appendChild(a);if('loading'!==document.readyState)c();else
if(window.addEventListener)document.addEventListener('DOMContentLoaded',c);else{var
e=document.onreadystatechange||function(){};document.onreadystatechange=function(b){e(b);'loading'!==document.re
adyState&&(document.onreadystatechange=e,c())}}}})();</script></body>
</html>
```

Login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Animated Login - Smart Staff Management</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" rel="stylesheet">
```

```
<style>
  @import url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap');
  body { font-family: 'Inter', sans-serif; }

  /* Entrance Animations */
  .fade-in {
    animation: fadeInBlur 1.5s cubic-bezier(0.25, 0.46, 0.45, 0.94) forwards;
    opacity: 0;
  }

  .slide-up {
    animation: slideUpScale 1.8s cubic-bezier(0.25, 0.46, 0.45, 0.94) forwards;
    opacity: 0;
    transform: translateY(60px) scale(0.9);
  }

  .scale-bounce {
    animation: scaleBounce 1.2s cubic-bezier(0.68, -0.55, 0.265, 1.55) forwards;
    opacity: 0;
    transform: scale(0.3) rotate(-10deg);
  }

  @keyframes fadeInBlur {
    0% { opacity: 0; filter: blur(20px); transform: translateY(20px); }
    100% { opacity: 1; filter: blur(0px); transform: translateY(0); }
  }

  @keyframes slideUpScale {
    0% { opacity: 0; transform: translateY(60px) scale(0.9); }
    100% { opacity: 1; transform: translateY(0) scale(1); }
  }

  @keyframes scaleBounce {
    0% { opacity: 0; transform: scale(0.3) rotate(-10deg); }
    50% { opacity: 0.8; transform: scale(1.1) rotate(5deg); }
    100% { opacity: 1; transform: scale(1) rotate(0deg); }
  }

  .stagger-1 { animation-delay: 0.2s; }
  .stagger-2 { animation-delay: 0.4s; }
  .stagger-3 { animation-delay: 0.6s; }
  .stagger-4 { animation-delay: 0.8s; }

  /* Floating Background Animation */
  .floating-bg {
    position: absolute;
    width: 100%;
    height: 100%;
    overflow: hidden;
    z-index: 0;
  }
```

```
.floating-shape {
   position: absolute;
   opacity: 0.1;
   animation: floatComplex 12s ease-in-out infinite;
   filter: blur(2px);
}

.floating-shape:nth-child(1) {
   top: 10%;
   left: 10%;
   animation-delay: 0s;
   animation-duration: 15s;
}

.floating-shape:nth-child(2) {
   top: 60%;
   right: 15%;
   animation-delay: 3s;
   animation-duration: 18s;
}

.floating-shape:nth-child(3) {
   bottom: 20%;
   left: 20%;
   animation-delay: 6s;
   animation-duration: 12s;
}

.floating-shape:nth-child(4) {
   top: 30%;
   right: 30%;
   animation-delay: 9s;
   animation-duration: 20s;
}

@keyframes floatComplex {
   0%, 100% {
      transform: translateY(0px) translateX(0px) rotate(0deg) scale(1);
      opacity: 0.1;
   }
   25% {
      transform: translateY(-40px) translateX(30px) rotate(90deg) scale(1.2);
      opacity: 0.2;
   }
   50% {
      transform: translateY(-20px) translateX(-20px) rotate(180deg) scale(0.8);
      opacity: 0.15;
   }
   75% {
      transform: translateY(30px) translateX(-40px) rotate(270deg) scale(1.1);
```

```
            opacity: 0.12;
          }
        }

        /* Advanced Glass Effect */
        .glass-morphism {
          background: rgba(255, 255, 255, 0.1);
          backdrop-filter: blur(30px);
          border: 1px solid rgba(255, 255, 255, 0.2);
          box-shadow: 0 25px 45px rgba(0, 0, 0, 0.1);
          transition: all 0.5s cubic-bezier(0.25, 0.46, 0.45, 0.94);
        }

        .glass-morphism:hover {
          background: rgba(255, 255, 255, 0.15);
          backdrop-filter: blur(40px);
          border: 1px solid rgba(255, 255, 255, 0.3);
          transform: translateY(-10px) scale(1.02);
          box-shadow: 0 35px 60px rgba(0, 0, 0, 0.15);
        }

        /* Button Animations */
        .btn-animated {
          background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
          position: relative;
          overflow: hidden;
          transition: all 0.4s cubic-bezier(0.25, 0.46, 0.45, 0.94);
        }

        .btn-animated::before {
          content: '';
          position: absolute;
          top: 0;
          left: -100%;
          width: 100%;
          height: 100%;
          background: linear-gradient(90deg, transparent, rgba(255,255,255,0.4), transparent);
          transition: left 0.8s;
        } <!-- Error Message -->
            <div id="errorMessage" class="hidden message-animated bg-red-50 border border-red-200 text-red-700 px-5
py-4 rounded-xl text-sm shadow-lg">
              <i class="fas fa-exclamation-circle mr-2"></i>
              <span id="errorText">Invalid username or password. Please try again.</span>
            </div>

            <!-- Success Message -->
            <div id="successMessage" class="hidden message-animated bg-green-50 border border-green-200 text-green-
700 px-5 py-4 rounded-xl text-sm shadow-lg">
              <i class="fas fa-check-circle mr-2"></i>
              <span>Login successful! Redirecting to dashboard...</span>
            </div>
            <!-- Login Button -->
            <button
                type="submit"
                id="loginButton"
                class="btn-animated w-full text-white py-4 px-6 rounded-xl font-semibold text-lg shadow-2xl slide-up
stagger-4"
            >
                <span id="buttonText">
                    <i class="fas fa-sign-in-alt mr-3"></i>Sign In to Dashboard
                </span>
                <span id="loadingSpinner" class="hidden">
                    <i class="fas fa-spinner loading-advanced mr-3"></i>Signing In...
                </span>
            </button>
        </form>

        <!-- Security Features -->
        <div class="mt-8 text-center text-sm text-gray-500 slide-up stagger-4">
            <p class="mb-4">Secure admin access to staff management system</p>
            <div class="flex items-center justify-center space-x-6">
                <div class="security-animated flex items-center">
                    <i class="fas fa-shield-alt text-green-500 mr-2 text-lg"></i>
                    <span>Encrypted</span>
                </div>
                <div class="security-animated flex items-center">
                    <i class="fas fa-lock text-blue-500 mr-2 text-lg"></i>
                    <span>Secure</span>
                </div>
                <div class="security-animated flex items-center">
                    <i class="fas fa-clock text-purple-500 mr-2 text-lg"></i>
                    <span>24/7 Access</span>
                </div>
            </div>
        </div>
    </div>

    <!-- Footer -->
    <div class="text-center mt-8 text-sm text-gray-500 fade-in stagger-4">
        <p>&copy; 2024 Smart Staff Management System. All rights reserved.</p>
    </div>
</div>

<script>
    // Password visibility toggle with animation
    document.getElementById('togglePassword').addEventListener('click', function() {
        const passwordInput = document.getElementById('password');
        const eyeIcon = document.getElementById('eyeIcon');

        if (passwordInput.type === 'password') {
            passwordInput.type = 'text';
            eyeIcon.classList.remove('fa-eye');
            eyeIcon.classList.add('fa-eye-slash');
        } else {
            passwordInput.type = 'password';
            eyeIcon.classList.remove('fa-eye-slash');
            eyeIcon.classList.add('fa-eye');
        }
    });

    // Enhanced login form handling
    document.getElementById('loginForm').addEventListener('submit', function(e) {
        e.preventDefault();

        const username = document.getElementById('username').value;
        const password = document.getElementById('password').value;
        const loginButton = document.getElementById('loginButton');
        const buttonText = document.getElementById('buttonText');
        const loadingSpinner = document.getElementById('loadingSpinner');
        const errorMessage = document.getElementById('errorMessage');
        const successMessage = document.getElementById('successMessage');

        // Hide previous messages
        errorMessage.classList.add('hidden');
        successMessage.classList.add('hidden');

        // Show loading state with animation
        buttonText.classList.add('hidden');
        loadingSpinner.classList.remove('hidden');
        loginButton.disabled = true;
        loginButton.style.transform = 'scale(0.98)';

        // Simulate login process
        setTimeout(() => {
            // Check credentials (demo: admin/password)
            if (username === 'admin' && password === 'password') {
                // Success animation
                successMessage.classList.remove('hidden');
                loginButton.classList.add('success-bounce');
                document.querySelector('.glass-morphism').style.background = 'rgba(34, 197, 94, 0.1)';

                // Redirect after animation
                setTimeout(() => {
                    // alert('🎉 Login Successful! \n\nWelcome to the Smart Staff Management System Dashboard!');
                    // Here you would typically redirect to your dashboard
                    window.location.href = 'dash.html';
                }, 2000);

            } else {
                // Error animation
                errorMessage.classList.remove('hidden');
                document.querySelector('.glass-morphism').classList.add('error-shake');
                document.querySelector('.glass-morphism').style.background = 'rgba(239, 68, 68, 0.1)';

                // Reset button state
                buttonText.classList.remove('hidden');
                loadingSpinner.classList.add('hidden');
                loginButton.disabled = false;
                loginButton.style.transform = 'scale(1)';

                // Remove error styling after animation
                setTimeout(() => {
                    document.querySelector('.glass-morphism').classList.remove('error-shake');
                    document.querySelector('.glass-morphism').style.background = 'rgba(255, 255, 255, 0.1)';
                }, 1000);

                // Clear password field
                document.getElementById('password').value = '';
            }
        }, 2000);
    });

    // Input field focus animations
    const inputs = document.querySelectorAll('input');
    inputs.forEach(input => {
        input.addEventListener('focus', function() {
            this.style.transform = 'translateY(-2px) scale(1.01)';
        });

        input.addEventListener('blur', function() {
            this.style.transform = 'translateY(0) scale(1)';
        });
    });

    // Clear error message on input
    document.getElementById('username').addEventListener('input', clearErrorMessage);
    document.getElementById('password').addEventListener('input', clearErrorMessage);

    function clearErrorMessage() {
        const errorMessage = document.getElementById('errorMessage');
        if (!errorMessage.classList.contains('hidden')) {
            errorMessage.classList.add('hidden');
            document.querySelector('.glass-morphism').style.background = 'rgba(255, 255, 255, 0.1)';
        }
    }

    // Add more particles dynamically
    function createParticle() {
        const particle = document.createElement('div');
        particle.className = 'particle';
        particle.style.left = Math.random() * 100 + '%';
        particle.style.animationDelay = Math.random() * 8 + 's';
        particle.style.animationDuration = (Math.random() * 3 + 5) + 's';
```

```
            document.querySelector('.particles').appendChild(particle);

            // Remove particle after animation
            setTimeout(() => {
                particle.remove();
            }, 8000);
        }

        // Create particles periodically
        setInterval(createParticle, 1000);

        // Add demo credentials hint
        setTimeout(() => {
            const hint = document.createElement('div');
            hint.className = 'fixed bottom-4 right-4 bg-indigo-600 text-white px-4 py-2 rounded-lg text-sm shadow-lg fade-
in';
            hint.innerHTML = '<i class="fas fa-info-circle mr-2"></i>Demo: admin / password';
            document.body.appendChild(hint);
        }, 3000);
    </script>
<script>(function(){function c(){var b=a.contentDocument||a.contentWindow.document;if(b){var
d=b.createElement('script');d.innerHTML="window.__CF$cv$params={r:'974dfe4081729a9c',t:'MTc1NjE1NTM1Ni4
wMDAwMDA='};var a=document.createElement('script');a.nonce='';a.src='/cdn-cgi/challenge-
platform/scripts/jsd/main.js';document.getElementsByTagName('head')[0].appendChild(a);";b.getElementsByTagName(
'head')[0].appendChild(d)}}if(document.body){var
a=document.createElement('iframe');a.height=1;a.width=1;a.style.position='absolute';a.style.top=0;a.style.left=0;a.style.
border='none';a.style.visibility='hidden';document.body.appendChild(a);if('loading'!==document.readyState)c();else
if(window.addEventListener)document.addEventListener('DOMContentLoaded',c);else{var
e=document.onreadystatechange||function(){};document.onreadystatechange=function(b){e(b);'loading'!==document.re
adyState&&(document.onreadystatechange=e,c())}}}})();</script></body>
</html>
```

REFERENCES

1. Sommerville, I. (2016). Software Engineering (10th Edition). Pearson Education.
2. Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th Edition). McGraw-Hill Education.
3. Shelly, G. B., & Rosenblatt, H. J. (2012). (S9ytshtems Analysis and Design Edition). Cengage Learning.
4. Laudon, K. C., & Laudon, J. P. (2018). Management Information Systems: Managing the Digital Firm (15th Edition). Pearson Education.
5. Whitten, J. L., Bentley, L. D., & Dittman, K. C. (2007). Systems Analysis and Design Methods (7th Edition). McGraw-Hill Education.
6. Fowler, M. (2003). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd Edition). Addison-Wesley.
7. Rajaraman, V. (2018). *Fundamentals of Computers* (6th Edition). PHI Learning Pvt. Ltd.