INTRODUCTION

1.1 INTRODUCTION

The **EDUEASE-Interactive OD Approval Application** Is A Next-Generation Digital Platform Created To Revolutionize The Way Educational Institutions Manage **On-Duty (OD) Requests** For Students. Recognizing The Challenges And Inefficiencies Inherent In Traditional, Paper-Based OD Request Processes, EDUEASE Offers An **All-In-One, Automated Solution** Tailored To Meet The Needs Of Both **Students And Faculty**. By Transforming An Often Cumbersome Process Into A **Fast, Transparent, And User-Friendly** Experience, EDUEASE Simplifies Leave Management And Strengthens **Communication Within Academic Environments**.

The Platform Is Designed With **Accessibility And Ease Of Use** At Its Core, Enabling Students To Submit OD Requests Efficiently And Accurately. Through An **Integrated Calendar**, Students Can Select Specific Dates For Their Leave, While The Reason For The Absence Can Be Provided Within Designated Fields. The Platform Also Allows For The Attachment Of **Supporting Documentation**, Ensuring Requests Are Complete And Can Be Processed Quickly. This Feature Simplifies The Submission Process For Students, **Eliminating The Need For Printed Forms** And Reducing The Possibility Of Lost Or Misplaced Requests.

Faculty And Administrative Staff Benefit From A Well-Organized **Dashboard** That Centralizes All Incoming OD Requests, Allowing For **Immediate Review And Action**. Faculty Members Can Accept, Decline, Or Request Additional Information With Just One Click, Streamlining **Decision-Making**. EDUEASE's Automated System Maintains A Record Of All Submitted And Processed Requests, Adding A Layer Of **Transparency And Accountability**. Faculty Are Thus Relieved Of The **Administrative Burden** Of Managing Physical Paperwork And Can Focus More On Effective Decision-Making.

Beyond Basic Request And Approval Features, EDUEASE Fosters Enhanced **Communication Between Students And Staff** Through **Real-Time Notifications**. Students Receive **Instant Updates On The Status** Of Their Applications, Reducing Uncertainties And Providing Clarity Regarding Their Leave Requests. This **Instant Feedback System** Minimizes Delays And Potential Misunderstandings. For Instance, If A Request Is Pending Additional Information Or

Review, EDUEASE Provides **Timely Notifications** That Encourage Prompt Follow-Up. **Security** Is A Critical Focus Of The EDUEASE Application. By Integrating **Robust Data Encryption** And Storage Protocols, The Platform Ensures That **Sensitive Student Information Remains Confidential** And Adheres To Data Privacy Regulations. This Commitment To Security Builds Trust Among Users, As They Are Assured That Their Personal And Attendance-Related Information Is Managed Securely Within The Platform. Additionally, EDUEASE's **Cross-Platform Compatibility** Allows Seamless Access From Any Device—Desktop, Tablet, Or Smartphone—Making It Easy For Users To Interact With The Application From Any Location.

The Platform's **Efficiency** And **Eco-Friendly Digital Approach** Align With Modern Campus Goals Of **Sustainability** And **Operational Efficiency**. By Reducing Paperwork, EDUEASE Not Only Supports **Environmental Objectives** But Also Reduces **Administrative Costs** Associated With Processing Physical Requests. Furthermore, The Platform's Built-In **Analytics** Provide Valuable Insights, Allowing Educational Institutions To **Track Attendance Trends**, **Forecast Needs**, And Make Informed Decisions To Improve Campus Management Practices.

1.2 OBJECTIVE

The primary objective of the EDUEASE-Interactive OD Approval Application is to revolutionize the leave management system in educational institutions by creating a seamless, efficient, and user-friendly platform for OD (On-Duty) requests and approvals. Key objectives include:

- **ADD NEW OD APPLICATION**: Enables students to submit a new OD (On Duty) request, including details like reason, date, and proof, ensuring requests are correctly recorded in the system.
- **VIEW OD APPLICATIONS**: Allows students and staff to view all submitted OD requests, including their status, date, and any associated feedback, for transparency and easy tracking.
- **APPROVE/REJECT OD APPLICATIONS**: Empowers staff to review, approve, or reject OD requests efficiently, keeping a clear record of each decision for accountability.

- **UPDATE OD APPLICATION STATUS**: Provides real-time updates on the status of OD applications, notifying students of any changes and reducing the need for manual follow-up.
- **MANAGE NOTIFICATIONS**: Sends timely notifications to students and staff about application status changes, helping streamline communication and ensure prompt information relay.

1.3 MODULES

- **STUDENT APPLICATION MODULE:** Enables Students To Submit OD Requests With Date Selection, Reason Entry, And Proof Upload.

- **STAFF DASHBOARD MODULE:** Provides Staff A Dashboard For Reviewing And Deciding On OD Requests.

- **NOTIFICATION AND COMMUNICATION MODULE:** Sends Real-Time Updates To Students On The Status Of Their OD Applications.

- **AUTHENTICATION AND ROLE MANAGEMENT MODULE:** Ensures Secure Logins And Role-Based Access For Students And Staff.

- **REPORT GENERATION AND ANALYTICS MODULE:** Generates Reports And Insights On OD Request Trends And Approval Rates.

- **ADMIN SETTINGS MODULE:** Allows Configuration Of System Settings, User Permissions, And Notification Preferences.

- **PROOF MANAGEMENT MODULE:** Facilitates Document Uploads By Students And Easy Access For Staff Review.2.1 Software Description

SURVEY OF TECHNOLOGY

2.1 SOFTWARE DESCRIPTION

VISUAL STUDIO CODE

Visual Studio Code, commonly referred to as VS Code, is a powerful, open-source code editor developed by Microsoft. Designed with flexibility and productivity in mind, VS Code combines the simplicity of a text editor with advanced developer tooling, making it suitable for a wide range of programming tasks across different languages and frameworks. It has gained immense popularity in the developer community due to its user-friendly interface, extensibility, and efficient features that streamline the development process.

KEY FEATURES AND BENEFITS

1. **INTELLISENSE CODE COMPLETION:**
   VS Code's IntelliSense offers intelligent autocompletion for variables, methods, and functions, speeding up the coding process and reducing errors by suggesting the correct syntax and function names.
2. **BUILT-IN DEBUGGING:**
   VS Code includes integrated debugging tools, allowing developers to set breakpoints and inspect code execution, making bug fixing faster and more efficient without switching between tools.
3. **CUSTOMIZATION AND EXTENSIONS:**
   With a vast marketplace of extensions, VS Code can be tailored to specific needs, enhancing functionality with support for new languages, Git integration, and tools for testing and deployment.
4. **VERSION CONTROL WITH GIT INTEGRATION:**
   VS Code’s built-in Git support enables easy version control, allowing developers to manage code, commit changes, and push updates directly within the editor.

LANGUAGES

2.2.1 HTML (HYPERTEXT MARKUP LANGUAGE) – FRONT END

HTML, or HyperText Markup Language, is the standard language used to structure and layout web pages. It forms the backbone of all web content, defining the organization of text, images, links, and multimedia on a webpage. Through various HTML tags, developers can create different types of content—such as headings, paragraphs, lists, forms, tables, and hyperlinks—that define the basic structure of any website. HTML is integral in web development, as it allows browsers to render and display content in a structured, readable format for users.

PURPOSE IN EDUEASE:

In the EDUEASE application, HTML is essential for creating the layout and structure that users interact with. This includes:

- **FORMS FOR OD REQUESTS:**
  HTML enables the creation of interactive forms where students and staff can submit On-Duty (OD) requests directly through the application. These forms collect important information, providing a streamlined process for users to request permission and add details about their absence.
- **DISPLAYING APPLICATION DETAILS:**
  HTML tags are used to present various application details in an organized format, such as displaying approved or pending OD requests. This structure ensures clarity and ease of navigation.
- **CONTENT ORGANIZATION FOR USERS:**
  HTML structures content in a way that is intuitive for both students and staff. It organizes essential information, such as user options, notifications, and links, making the application more user-friendly and accessible.

By establishing a clear and organized foundation, HTML ensures that the EDUEASE platform meets its purpose of providing a well-structured, reliable interface, ultimately enhancing the user experience for students and staff.

2.2.2 CSS (CASCADING STYLE SHEETS) – FRONT END

CSS, or Cascading Style Sheets, is the primary styling language used to define the visual presentation of web applications. It brings HTML elements to life, dictating the layout, colors, fonts, spacing, and responsiveness that collectively create an engaging and accessible interface. CSS is essential for transforming a basic HTML structure into a polished, visually appealing, and user-friendly application. By managing style elements, CSS helps create a cohesive, consistent design across devices, ensuring that applications remain visually consistent and accessible on various platforms.

PURPOSE OF CSS IN THE EDUEASE APPLICATION :

In the EDUEASE application, CSS plays a crucial role in establishing a user-friendly, visually engaging interface. It helps create a seamless and unified experience, where components such as dashboards, forms, and buttons maintain a consistent and professional design throughout the application.

1. **STYLING INTERFACE COMPONENTS:**

   CSS provides cohesive design for dashboards, buttons, forms, and notifications, ensuring EDUEASE has a unified and polished look.
2. **ENHANCED USABILITY & ACCESSIBILITY:**

   CSS enables clear visual hierarchy and accessible design elements, helping users navigate intuitively and accommodating diverse needs, such as high-contrast views.
3. **RESPONSIVE & ADAPTIVE LAYOUT:**

   CSS ensures EDUEASE adapts seamlessly across devices, providing consistent experiences on mobile, tablet, and desktop.
4. **INTERACTIVE FEEDBACK:**

   CSS animations and transitions offer dynamic feedback, enriching user interactions by guiding actions, like button hovers or field focus.

2.2.3 REACT (JAVASCRIPT LIBRARY) – FRONT END

React is a powerful JavaScript library designed for building fast, interactive, and scalable user interfaces. Known for its efficiency in handling single-page applications, React enables developers to create reusable, modular components that can be easily managed, reused, and tested. Its component-based architecture streamlines the process of building complex user interfaces, while also making applications more maintainable and scalable over time. By using a virtual DOM, React updates only the specific components that change, resulting in smoother, faster user interactions without unnecessary page reloads.

PURPOSE OF REACT IN THE EDUEASE APPLICATION :

React plays a central role in the EDUEASE application, enabling a responsive and interactive experience tailored for both students and staff. Through its component-based approach, React empowers EDUEASE to deliver a highly efficient, dynamic user experience with real-time updates.

1. **DYNAMIC & RESPONSIVE INTERFACE:**
   React enables EDUEASE to build interactive elements like forms, dashboards, and notifications, ensuring consistent functionality and layout across devices.
2. **REUSABLE COMPONENTS:**
   React allows for standardized UI elements (e.g., buttons, forms) that reduce redundant code, speeding up development and enhancing maintenance.
3. **REAL-TIME UPDATES:**
   React manages application state efficiently, ensuring dynamic updates without page reloads, so data like application statuses and notifications refresh seamlessly.
4. **ENHANCED INTERACTIVITY:**
   React’s event-handling ensures immediate user feedback, making interactions on EDUEASE intuitive and engaging.
5. **OPTIMIZED PERFORMANCE:**
   Using a virtual DOM, React syncs only necessary changes to the DOM, providing a smoother experience with faster performance.

2.2.4 MONOGODB (NOSQL DATABASE) - DATABASE

MongoDB is a powerful NoSQL database known for its flexibility and high performance. Unlike traditional SQL databases, MongoDB stores data in a JSON-like, document-oriented format, allowing for the storage of semi-structured data without a fixed schema. This structure enables MongoDB to handle large datasets with ease and adapt as the data needs of applications evolve. MongoDB’s document-based model supports complex data types, which makes it ideal for applications that require a highly flexible data structure. Additionally, its scalability ensures that MongoDB can handle growing amounts of data and users without compromising performance, making it suitable for both small and large-scale applications.

PURPOSE OF MONGODB IN EDUEASE:

In the EDUEASE application, MongoDB is used as the primary storage solution for data related to student and staff interactions, such as Out of Duty (OD) requests, application statuses, and proof documents. MongoDB’s flexibility allows EDUEASE to accommodate various types of data without predefined schema constraints, enabling efficient updates and expansions as new requirements arise.

For instance, student OD requests can be stored with different attributes, such as request details, submission dates, and approval statuses, all within a single document. The document-oriented model also simplifies the organization of proof documents, allowing attachments or links to be stored alongside relevant application data, ensuring that everything related to a request is consolidated in one place.

Moreover, MongoDB supports efficient querying and indexing, which speeds up access to specific data, such as pending requests or recent approvals, enhancing the platform's responsiveness. This efficiency is especially beneficial as it enables quick retrieval of real-time information for students and staff without delays. By using MongoDB, EDUEASE can maintain a robust, scalable, and high-performing database that ensures data integrity and supports rapid access to critical information, ultimately contributing to a smooth and effective user experience.

2.2.5 JAVASCRIPT (PROGRAMMING LANGUAGE) - BACKEND

JavaScript is a highly versatile programming language widely used for creating dynamic, interactive web experiences. Originally developed for front-end functionality, it now also powers back-end systems through Node.js, allowing developers to build full-stack applications using a single language. JavaScript enables interactive elements, handles complex data processing, and facilitates real-time communication across various parts of an application. On the back end, it is particularly valuable for managing asynchronous operations, such as handling requests and performing background tasks, which are essential in modern, responsive web applications.

PURPOSE IN EDUEASE:

In EDUEASE, JavaScript (via Node.js) forms the core of the application's back-end logic. It is responsible for processing Out of Duty (OD) requests, handling form submissions, and managing real-time data updates. By utilizing JavaScript, EDUEASE can effectively interact with the MongoDB database to perform actions like storing user information, updating application statuses, and retrieving data in response to specific queries. JavaScript's asynchronous capabilities allow EDUEASE to handle multiple requests concurrently, ensuring that students and staff can interact with the application smoothly without experiencing slowdowns or delays.

Additionally, JavaScript enables EDUEASE to manage user sessions and application state efficiently. By tracking user sessions, EDUEASE can customize the user experience for both students and staff, maintaining relevant data across different interactions. This state management capability allows the application to provide a seamless experience as users navigate between forms, dashboards, and notifications. Furthermore, JavaScript facilitates secure, efficient communication between the front end and back end through API calls, ensuring data integrity while maintaining an interactive user experience. Overall, JavaScript's flexibility, efficiency, and real-time functionality make it an ideal choice for the back-end logic in EDUEASE.

REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 FUNCTIONAL REQUIREMENTS:

1. **OD REQUEST SUBMISSION:**
   Students can submit On-Duty (OD) requests through an intuitive form developed using HTML and React. The form allows students to select dates using an integrated calendar, input the reason for their leave, and submit their request. This feature ensures data is collected accurately and submission is smooth for the students.

2. **APPROVAL MANAGEMENT:**
   Faculty members can review and manage OD requests via a streamlined approval panel, built with React. This panel allows faculty to approve or decline requests with a single click, ensuring a fast and efficient approval workflow, thus reducing the time spent on administrative tasks.

3. **NOTIFICATION SYSTEM:**
   The system incorporates real-time notifications, powered by JavaScript and React, to instantly notify both students and faculty about the status of the OD requests. This functionality ensures transparent and timely communication regarding approval, rejection, or any status change of the requests.

4. **DASHBOARD AND RECORD MANAGEMENT:**
   A React-Based User Interface (UI) Serves As The Dashboard For Both Students And Faculty, Allowing Them To View, Track, And Manage Their OD Requests. The System Stores All Requests And Decisions In A Mongodb Database, Maintaining An Organized And Searchable Record For Easy Retrieval, Thus Ensuring That Application Data Is Efficiently Managed And Accessible At All Times.

3.1.2 NON-FUNCTIONAL REQUIREMENTS:

1. **HIGH ACCESSIBILITY:**

   The EDUEASE Platform, Built Using HTML, CSS, And React, Is Designed To Be Fully Responsive, Ensuring That The System Is Easily Accessible And Functional Across A Wide Range Of Devices, Including Desktops, Laptops, Tablets, And Mobile Phones. The Responsive Design Adjusts Seamlessly To Different Screen Sizes And Orientations, Providing An Optimal User Experience Regardless Of The Device Used. This Ensures That Students And Faculty Can Access The Platform Anytime, Anywhere, Enhancing Usability And Convenience.

2. **SECURE DATABASE:**

   The Platform's Database, Powered By Mongodb, Provides A Secure And Encrypted Solution For Storing Sensitive Student And Faculty Data. Data Privacy Is A Top Priority, And The System Ensures That All Personal Information, OD Request Details, And Approval Records Are Safely Stored And Protected From Unauthorized Access. Mongodb’s Robust Security Features, Including Encryption At Rest And Secure Data Access Protocols, Help Maintain Confidentiality And Integrity, Fostering Trust And Compliance With Data Protection Regulations.

3. **SCALABILITY:**

   The EDUEASE System Is Designed With Scalability In Mind, Capable Of Handling A Large Volume Of Requests And User Data Without Compromising Performance. As The Platform Grows To Accommodate An Increasing Number Of Students, Faculty Members, And Departments, The Infrastructure Can Scale Efficiently To Meet The Demands. Mongodb’s Flexible Schema And Horizontal Scaling Capabilities Ensure That The System Can Support Growing Data And Traffic, Enabling EDUEASE To Adapt To Future Growth And Increased Usage Over Time. This Scalability Ensures That The System Remains Responsive And Reliable As Its User Base Expands.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

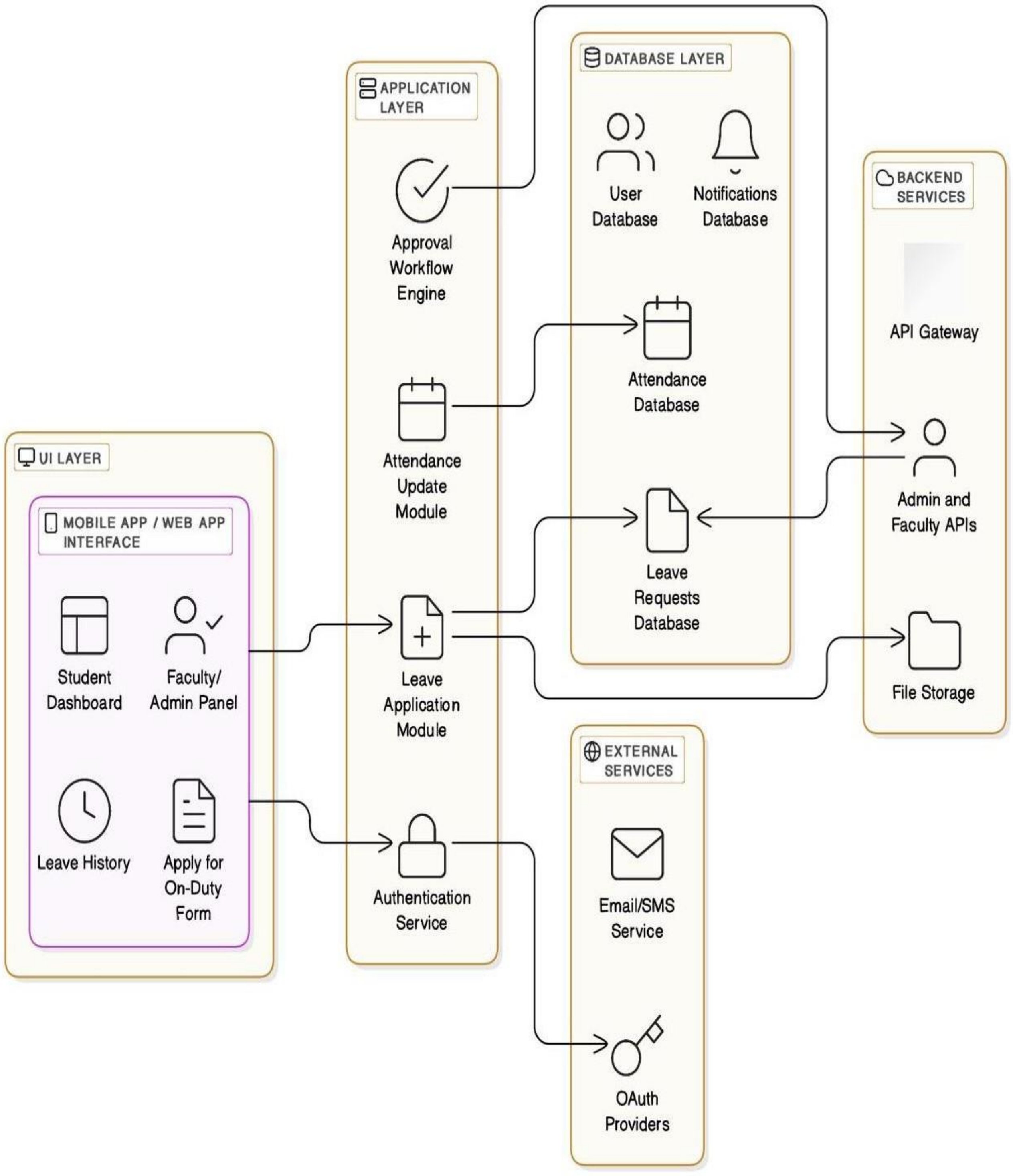
3.2.1 HARDWARE REQUIREMENTS:

1. **COMPUTER WITH WINDOWS 10/LINUX OPERATING SYSTEM**:
   Required For Both Development And Server Deployment.
2. **WEB SERVER OR CLOUD-BASED SERVER**:
   For Hosting The Application And Providing Real-Time Access To All Users.
3. **MINIMUM 4 GB RAM AND 500 GB STORAGE**:
   Ensures That The Application Operates Efficiently And Can Handle Data Storage And High-Performance Requirements

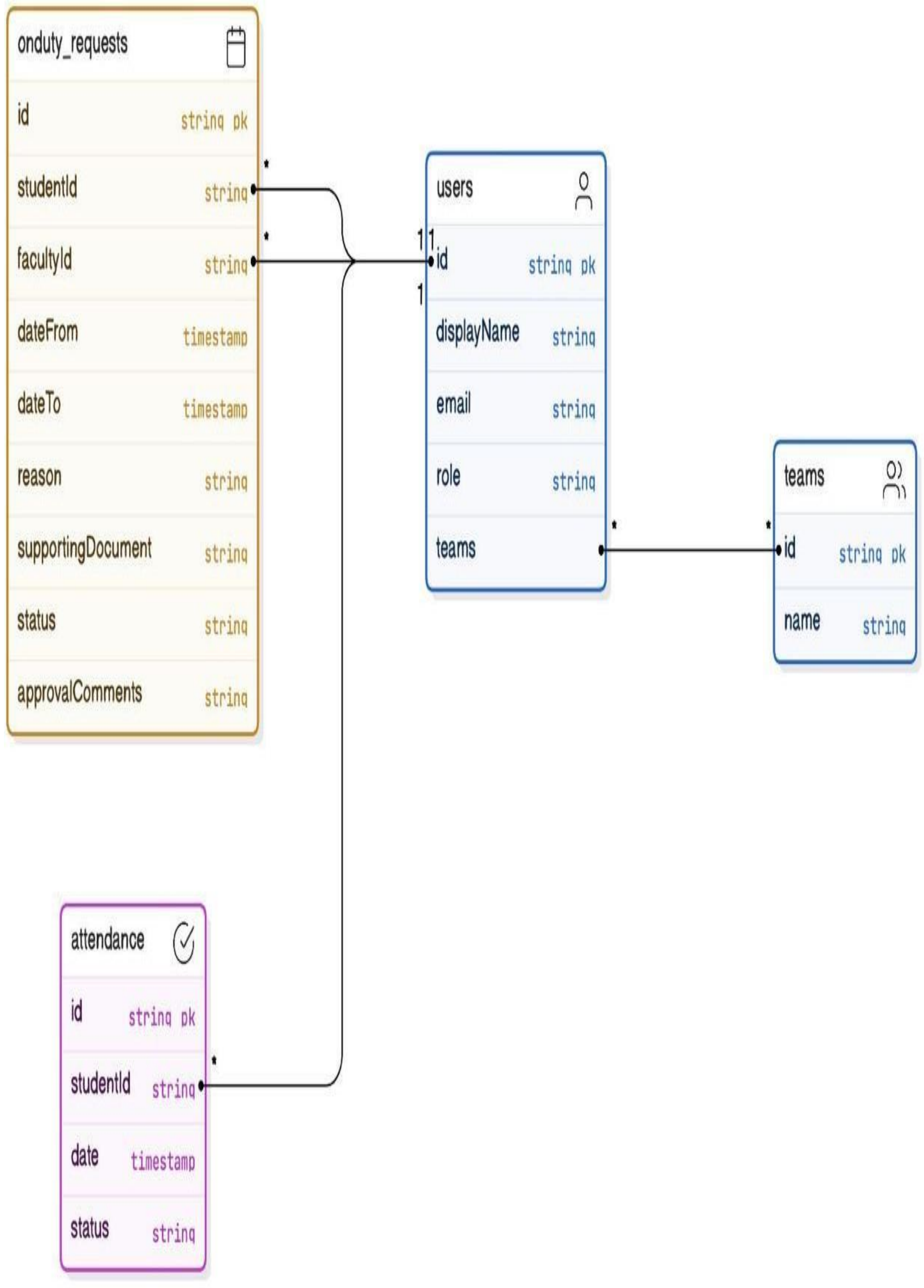
3.2.2 SOFTWARE REQUIREMENTS:

1. **HTML, CSS, AND REACT (FRONT-END)**:
   Used For Creating A Responsive And Interactive Interface, Forming The Core Front-End Technologies For User Experience.
2. **MONGODB (BACK-END)**:
   A Nosql Database Chosen For Its Flexibility And Scalability, Suitable For Storing Student Requests, Faculty Decisions, And All Record Data.
3. **JAVASCRIPT (BACK-END)**:
   Powers Interactive And Dynamic Functionalities, Including Real-Time Notifications, Event Handling, And Additional Server-Side Logic When Used With **Node.Js**.
4. **ADDITIONAL LIBRARIES AND TOOLS**:
   REACT ROUTER:
   For Navigating Between Different Pages And Components Within The Application.
   EXPRESS.JS:
   Server Framework For Handling Api Requests And Integrating With **Mongodb**.
   MONGOOSE:
   An Object Data Modeling (Odm) Library For Mongodb, Used For Managing Data Structure And Database Interactions.
   NODEMAILER (If Email Notifications Are Needed):
   For Sending Automated Email Updates On Od Request Status.

ARCHITECTURE DIAGRAM OF EDUEASE APPLICATION



ER DIAGRAM



onduty_requests
id string pk
studentId string
facultyId string
dateFrom timestamp
dateTo timestamp
reason string
supportingDocument string
status string
approvalComments string
users
id string pk
displayName string
email string
role string
teams
teams
id string pk
name string
attendance
id string pk
studentId string
date timestamp
status string
*
*
1
1
1
*
*
*

PROGRAM CODE

ARENA FOR CODING : VISUAL STUDIO CODE

CODE FOR LOGIN PAGE

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { User } from 'lucide-react';

const Login: React.FC = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const navigate = useNavigate();

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setError('');

    try {
      const response = await fetch('/api/login', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ username, password }),
      });

      if (response.ok) {
        const data = await response.json();
        localStorage.setItem('user', JSON.stringify(data.user));
navigate(data.user.role === 'student' ? '/student' : '/staff');
      } else {
        const errorData = await response.json();
        setError(errorData.message || 'Login failed. Please try again.');
      }
    } catch (err) {
      setError('An error occurred. Please try again later.');
    }
  };

  return (
    <div className="flex items-center justify-center min-h-screen bg-gray-100">
      <div className="px-8 py-6 mt-4 text-left bg-white shadow-lg rounded-lg">
        <h3 className="text-2xl font-bold text-center flex items-center justify-center">
          <User className="mr-2" /> Login to Your Account
        </h3>
        {error && <p className="text-red-500 text-center mt-2">{error}</p>}
        <form onSubmit={handleSubmit}>
          <div className="mt-4">
            <div>
              <label className="block" htmlFor="username">Username</label>
              <input
                type="text"
                placeholder="Username"
                className="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-
1 focus:ring-blue-600"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
                required
              />
            </div>
            <div className="mt-4">
              <label className="block" htmlFor="password">Password</label>
```

```
<input
               type="password"
               placeholder="Password"
               className="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-
1 focus:ring-blue-600"
               value={password}
               onChange={(e) => setPassword(e.target.value)}
               required
             />
           </div>
           <div className="flex items-baseline justify-between">
             <button className="px-6 py-2 mt-4 text-white bg-blue-600 rounded-lg hover:bg-blue-
900" type="submit">Login</button>
           </div>
         </div>
       </form>
     </div>
   </div>
  );
};

export default Login;
```

CODE FOR STUDENT/STAFF LOGIN PAGE

```
import React from 'react';
import { Navigate } from 'react-router-dom';

interface PrivateRouteProps {
  children: React.ReactNode;
  role: 'student' | 'staff';
}

const PrivateRoute: React.FC<PrivateRouteProps> = ({ children, role }) => {
  const user = JSON.parse(localStorage.getItem('user') || '{}');

  if (!user || user.role !== role) {
    return <Navigate to="/" replace />;
  }

  return <>{children}</>;
};

export default PrivateRoute;
```

CODE FOR STAFF DASHBOARD INTERFACE

```
import React, { useState, useEffect } from 'react';
import { CheckCircle, XCircle, FileText, LogOut } from 'lucide-react';
import { useNavigate } from 'react-router-dom';

interface OnDutyApplication {
  id: string;
  studentName: string;

reason: string;
  date: string;
  proofFile: string | null;
  status: 'pending' | 'approved' | 'rejected';
  feedback?: string;
}

const StaffDashboard: React.FC = () => {
  const [applications, setApplications] = useState<OnDutyApplication[]>([]);
  const navigate = useNavigate();

  useEffect(() => {
    // In a real application, fetch applications from an API
    const mockApplications: OnDutyApplication[] = [
      { id: '1', studentName: 'John Doe', reason: 'Conference', date: '2024-03-15', proofFile:
'conference_invite.pdf', status: 'pending' },
      { id: '2', studentName: 'Jane Smith', reason: 'Workshop', date: '2024-03-20', proofFile:
'workshop_details.pdf', status: 'pending' },
    ];
    setApplications(mockApplications);
  }, []);

  const handleApprove = (id: string) => {
    setApplications(applications.map(app =>
      app.id === id ? { ...app, status: 'approved', feedback: 'Application approved.' } : app
    ));
  };

  const handleReject = (id: string, feedback: string) => {
    setApplications(applications.map(app =>
      app.id === id ? { ...app, status: 'rejected', feedback } : app
    ));
  };
  const handleLogout = () => {
    localStorage.removeItem('user');
    navigate('/');
  };

  return (
    <div className="container mx-auto p-4">
      <div className="flex justify-between items-center mb-6">
```

```
        <h1 className="text-3xl font-bold">Staff Dashboard</h1>
        <button
          onClick={handleLogout}
          className="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded flex
items-center"
        >
          <LogOut className="mr-2" /> Logout
        </button>
      </div>

      <div className="bg-white p-6 rounded-lg shadow-md">
        <h2 className="text-2xl font-semibold mb-4">Pending Applications</h2>
        {applications.filter(app => app.status === 'pending').length === 0 ? (
          <p>No pending applications.</p>
        ) : (
          <ul className="divide-y divide-gray-200">
            {applications.filter(app => app.status === 'pending').map((app) => (
              <li key={app.id} className="py-4">
                <div className="flex justify-between items-start">
                  <div>
                    <p className="font-semibold">{app.studentName}</p>
                    <p className="text-sm text-gray-500">{app.reason} - {app.date}</p>
                    {app.proofFile && (
                      <a href="#" className="text-blue-600 hover:underline flex items-center mt-1">
                        <FileText className="w-4 h-4 mr-1" /> View Proof
                      </a>
                    )}
                  </div>
                  <div className="flex space-x-2">
                    <button
                      onClick={() => handleApprove(app.id)}
                      className="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4
rounded flex items-center"
```

```
                  >
                    <CheckCircle className="w-4 h-4 mr-1" /> Approve
                  </button>
                  <button
                    onClick={() => {
                      const feedback = prompt('Enter rejection reason:');
                      if (feedback) handleReject(app.id, feedback);
                    }}
                    className="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded
flex items-center"
                  >
                    <XCircle className="w-4 h-4 mr-1" /> Reject
                  </button>
                </div>
              </div>
            </li>
          ))}
        </ul>
      )}
    </div>

    <div className="mt-6 bg-white p-6 rounded-lg shadow-md">
      <h2 className="text-2xl font-semibold mb-4">Recent Decisions</h2>
      <ul className="divide-y divide-gray-200">

{applications.filter(app => app.status !== 'pending').map((app) => (
          <li key={app.id} className="py-4">
            <div className="flex justify-between">
              <div>
                <p className="font-semibold">{app.studentName}</p>
                <p className="text-sm text-gray-500">{app.reason} - {app.date}</p>
              </div>
              <span className={`px-2 inline-flex text-xs leading-5 font-semibold rounded-full ${
```

```
              app.status === 'approved' ? 'bg-green-100 text-green-800' : 'bg-red-100 text-red-800'
            }`}>
              {app.status.charAt(0).toUpperCase() + app.status.slice(1)}
            </span>
          </div>
          {app.feedback && (
            <p className="mt-1 text-sm text-gray-600">Feedback: {app.feedback}</p>
          )}
        </li>
      ))}
    </ul>
  </div>
 </div>
 );
};

export default StaffDashboard;
```

CODE FOR STUDENT STAFF DASHBOARD INTERFACE

```
import React, { useState, useEffect } from 'react';
import { FileUp, Send, LogOut } from 'lucide-react';
import { useNavigate } from 'react-router-dom';

interface OnDutyApplication {
  id: string;
  reason: string;
  date: string;
  proofFile: File | null;
  status: 'pending' | 'approved' | 'rejected';
  feedback?: string;
```

```
}
const StudentDashboard: React.FC = () => {
  const [applications, setApplications] = useState<OnDutyApplication[]>([]);
  const [reason, setReason] = useState('');
  const [date, setDate] = useState('');
  const [proofFile, setProofFile] = useState<File | null>(null);
  const navigate = useNavigate();

  useEffect(() => {
    // In a real application, fetch applications from an API
    const mockApplications: OnDutyApplication[] = [
      { id: '1', reason: 'Conference', date: '2024-03-15', proofFile: null, status: 'approved',
feedback: 'Approved. Enjoy the conference!' },
      { id: '2', reason: 'Workshop', date: '2024-03-20', proofFile: null, status: 'pending' },
    ];
    setApplications(mockApplications);
  }, []);

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    const newApplication: OnDutyApplication = {
      id: Date.now().toString(),
      reason,
      date,
      proofFile,
      status: 'pending'
    };
    setApplications([...applications, newApplication]);
    setReason('');
    setDate('');
    setProofFile(null);
    alert('Application submitted successfully!');
  };
  const handleFileChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    if (e.target.files) {
      setProofFile(e.target.files[0]);
    }
  };

  const handleLogout = () => {
    localStorage.removeItem('user');
    navigate('/');
  };

  return (
    <div className="container mx-auto p-4">
      <div className="flex justify-between items-center mb-6">
        <h1 className="text-3xl font-bold">Student Dashboard</h1>
        <button

onClick={handleLogout}
          className="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded flex
items-center"
        >
          <LogOut className="mr-2" /> Logout
        </button>
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        <div className="bg-white p-6 rounded-lg shadow-md">
          <h2 className="text-2xl font-semibold mb-4">Apply for On-Duty Leave</h2>
          <form onSubmit={handleSubmit}>
            <div className="mb-4">
              <label htmlFor="reason" className="block text-sm font-medium text-gray-
700">Reason</label>
```

```
            <input
              type="text"
              id="reason"
              value={reason}
              onChange={(e) => setReason(e.target.value)}
              className="mt-1 block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50"
              required
            />
          </div>
          <div className="mb-4">
            <label htmlFor="date" className="block text-sm font-medium text-gray-
700">Date</label>
            <input
              type="date"
              id="date"
              value={date}

onChange={(e) => setDate(e.target.value)}
              className="mt-1 block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50"
              required
            />
          </div>
          <div className="mb-4">
            <label htmlFor="proof" className="block text-sm font-medium text-gray-
700">Upload Proof</label>
            <div className="mt-1 flex items-center">
              <input
                type="file"
                id="proof"
                onChange={handleFileChange}
                className="hidden"
                accept=".pdf,.png,.jpg,.jpeg,.zip"
              />
              <label
                htmlFor="proof"
                className="cursor-pointer bg-white py-2 px-3 border border-gray-300 rounded-md
shadow-sm text-sm leading-4 font-medium text-gray-700 hover:bg-gray-50 focus:outline-none
focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"
              >
                <FileUp className="w-5 h-5 inline-block mr-2" />
                Choose file
              </label>
              <span className="ml-3">{proofFile ? proofFile.name : 'No file chosen'}</span>
            </div>
          </div>
          <button
            type="submit"
            className="w-full flex justify-center py-2 px-4 border border-transparent rounded-
md shadow-sm text-sm font-medium text-white bg-indigo-600 hover:bg-indigo-700
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"
          >
            <Send className="w-5 h-5 mr-2" /> Submit Application
          </button>
        </form>
      </div>

      <div className="bg-white p-6 rounded-lg shadow-md">
        <h2 className="text-2xl font-semibold mb-4">Your Applications</h2>
        {applications.length === 0 ? (
          <p>No applications submitted yet.</p>
        ) : (
          <ul className="divide-y divide-gray-200">
            {applications.map((app) => (
              <li key={app.id} className="py-4">
```

```
                <div className="flex justify-between">
                  <div>
                    <p className="font-semibold">{app.reason}</p>
                    <p className="text-sm text-gray-500">{app.date}</p>
                  </div>
                  <span className={`px-2 inline-flex text-xs leading-5 font-semibold rounded-full
${
                    app.status === 'approved' ? 'bg-green-100 text-green-800' :
                    app.status === 'rejected' ? 'bg-red-100 text-red-800' :
                    'bg-yellow-100 text-yellow-800'
                  }`}>
                    {app.status.charAt(0).toUpperCase() + app.status.slice(1)}
                  </span>
                </div>
                {app.feedback && (
                  <p className="mt-1 text-sm text-gray-600">Feedback: {app.feedback}</p>
                )}

</li>
            ))}
          </ul>
        )}
      </div>
    </div>
  </div>
  );
};

export default StudentDashboard;
```

CODE FOR CONNECTING MONGODB SERVER

TO THE FRONT-END

```
import express from 'express';
import { MongoClient } from 'mongodb';
import bcrypt from 'bcryptjs';
import dotenv from 'dotenv';

dotenv.config();

const app = express();
app.use(express.json());

const mongoURI = process.env.MONGODB_URI;
const client = new MongoClient(mongoURI);

async function connectToDatabase() {
  try {
    await client.connect();
    console.log('Connected to MongoDB');
  } catch (error) {
    console.error('Error connecting to MongoDB:', error);
    process.exit(1);
  }
}

connectToDatabase();

app.post('/api/login', async (req, res) => {
  const { username, password } = req.body;

  try {
    const db = client.db('on-duty-app');
    const usersCollection = db.collection('users');

    const user = await usersCollection.findOne({ username });

    if (!user) {
      return res.status(401).json({ message: 'Invalid username or password' });
    }

    const isPasswordValid = await bcrypt.compare(password, user.password);

    if (!isPasswordValid) {
      return res.status(401).json({ message: 'Invalid username or password' });
    }
    // Remove the password from the user object before sending it to the client
    const { password: _, ...userWithoutPassword } = user;

    res.json({ user: userWithoutPassword });
  } catch (error) {
    console.error('Login error:', error);
    res.status(500).json({ message: 'An error occurred during login' });
  }
});

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

CODE FOR APP BUILDING

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Login from './components/Login';
import StudentDashboard from './components/StudentDashboard';

import StaffDashboard from './components/StaffDashboard';
import PrivateRoute from './components/PrivateRoute';

function App() {
  return (
    <Router>
      <div className="min-h-screen bg-gray-100">
        <Routes>
          <Route path="/" element={<Login />} />
          <Route
            path="/student"
            element={
              <PrivateRoute role="student">
                <StudentDashboard />
              </PrivateRoute>
            }
          />
          <Route
            path="/staff"
            element={
              <PrivateRoute role="staff">
                <StaffDashboard />
              </PrivateRoute>
            }
          />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

CODE FOR DESIGNING INDEX INTERFACE

```
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';

/* Base font and background */
body {
  font-family: 'Inter', sans-serif;
  background-color: #0011ff; /* White background */
  color: #430081; /* Purple text */
}

/* Container styling */
.container {
    max-width: 1200px;
}

/* Class to hide elements */
.hidden {
    display: none;
}

/* Icon size and positioning */
.icon {
    width: 1em;
    height: 1em;
    vertical-align: middle;
}

/* New styles for buttons, links, etc. to match theme */
.bg-primary {
    background-color: #7affe2; /* Purple background for buttons */

color: #a0ff94; /* White text on buttons */
}

.bg-primary:hover {
    background-color: #35ccf1; /* Slightly lighter purple on hover */
}
```

CODE FOR VIEWING MAIN FILE

```
import { StrictMode } from 'react';
import { createRoot } from 'react-dom/client';
import App from './App.tsx';
import './index.css';
```

```
createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

CODE FOR STORING DATAS IN APPLICATION

```
// Mock data for applications
let applications = [
    { id: '1', studentName: 'John Doe', reason: 'Conference', date: '2024-03-15', proofFile:
'conference_invite.pdf', status: 'pending' },
    { id: '2', studentName: 'Jane Smith', reason: 'Workshop', date: '2024-03-20', proofFile:
'workshop_details.pdf', status: 'pending' },
];

// Current user
let currentUser = null;

// DOM Elements
const app = document.getElementById('app');

// Render functions
function renderLogin() {
    app.innerHTML = `
        <div class="flex items-center justify-center min-h-screen">
            <div class="px-8 py-6 mt-4 text-left bg-white shadow-lg rounded-lg">
                <h3 class="text-2xl font-bold text-center flex items-center justify-center">
                    <i data-lucide="user" class="mr-2"></i> Login to Your Account
                </h3>
                <form id="loginForm">
                <div class="mt-4">
                    <div>
                        <label class="block" for="username">Username</label>
                        <input type="text" placeholder="Username" id="username"
                            class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none
focus:ring-1 focus:ring-blue-600"
                            required>
                    </div>
                    <div class="mt-4">
                        <label class="block" for="password">Password</label>
                        <input type="password" placeholder="Password" id="password"
                            class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none
focus:ring-1 focus:ring-blue-600"
                            required>
                    </div>
                    <div class="mt-4">
                        <label class="block">Role</label>
                        <select id="role"
                            class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none
focus:ring-1 focus:ring-blue-600">
                            <option value="student">Student</option>
                            <option value="staff">Staff</option>
                        </select>
                    </div>
                    <div class="flex items-baseline justify-between">
                        <button class="px-6 py-2 mt-4 text-white bg-blue-600 rounded-lg hover:bg-
blue-900" type="submit">Login</button>
                    </div>
                </div>
            </form>
        </div>
    </div>
  `;
```

```
    // Add event listener to the login form
    document.getElementById('loginForm').addEventListener('submit', handleLogin);
    lucide.createIcons();
}

function renderStudentDashboard() {
    app.innerHTML = `
        <div class="container mx-auto p-4">
            <div class="flex justify-between items-center mb-6">
                <h1 class="text-3xl font-bold">Student Dashboard</h1>
                <button id="logoutBtn" class="bg-red-500 hover:bg-red-700 text-white font-bold py-
2 px-4 rounded flex items-center">
                    <i data-lucide="log-out" class="mr-2"></i> Logout
                </button>
            </div>



<div class="grid grid-cols-1 md:grid-cols-2 gap-6">
                <div class="bg-white p-6 rounded-lg shadow-md">
                    <h2 class="text-2xl font-semibold mb-4">Apply for On-Duty Leave</h2>
                    <form id="applicationForm">
                        <div class="mb-4">
                            <label for="reason" class="block text-sm font-medium text-gray-
700">Reason</label>
                            <input type="text" id="reason" class="mt-1 block w-full rounded-md border-
gray-300 shadow-sm focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-
opacity-50" required>
                        </div>
                        <div class="mb-4">
                            <label for="date" class="block text-sm font-medium text-gray-
700">Date</label>
```

```
              <input type="date" id="date" class="mt-1 block w-full rounded-md border-
gray-300 shadow-sm focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-
opacity-50" required>
            </div>
            <div class="mb-4">
              <label for="proof" class="block text-sm font-medium text-gray-700">Upload
Proof</label>
              <div class="mt-1 flex items-center">
                <input type="file" id="proof" class="hidden"
accept=".pdf,.png,.jpg,.jpeg,.zip">
                <label for="proof" class="cursor-pointer bg-white py-2 px-3 border
border-gray-300 rounded-md shadow-sm text-sm leading-4 font-medium text-gray-700
hover:bg-gray-50 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500">
                  <i data-lucide="file-up" class="w-5 h-5 inline-block mr-2"></i>
                  Choose file
                </label>
                <span id="fileName" class="ml-3">No file chosen</span>
              </div>

</div>
            <button type="submit" class="w-full flex justify-center py-2 px-4 border
border-transparent rounded-md shadow-sm text-sm font-medium text-white bg-indigo-600
hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-
500">
              <i data-lucide="send" class="w-5 h-5 mr-2"></i> Submit Application
            </button>
          </form>
        </div>

        <div class="bg-white p-6 rounded-lg shadow-md">
          <h2 class="text-2xl font-semibold mb-4">Your Applications</h2>
          <ul id="applicationsList" class="divide-y divide-gray-200"></ul>
        </div>
```

```
          </div>
        </div>
    `;

    // Add event listeners
    document.getElementById('logoutBtn').addEventListener('click', handleLogout);
    document.getElementById('applicationForm').addEventListener('submit',
handleApplicationSubmit);
    document.getElementById('proof').addEventListener('change', handleFileChange);

    // Render applications list
    renderApplicationsList();
    lucide.createIcons();
}

function renderStaffDashboard() {
    app.innerHTML = `
        <div class="container mx-auto p-4">

 <div class="flex justify-between items-center mb-6">
                <h1 class="text-3xl font-bold">Staff Dashboard</h1>
                <button id="logoutBtn" class="bg-red-500 hover:bg-red-700 text-white font-bold py-
2 px-4 rounded flex items-center">
                    <i data-lucide="log-out" class="mr-2"></i> Logout
                </button>
            </div>

            <div class="bg-white p-6 rounded-lg shadow-md">
                <h2 class="text-2xl font-semibold mb-4">Pending Applications</h2>
                <ul id="pendingApplicationsList" class="divide-y divide-gray-200"></ul>
            </div>

            <div class="mt-6 bg-white p-6 rounded-lg shadow-md">
            <h2 class="text-2xl font-semibold mb-4">Recent Decisions</h2>
            <ul id="recentDecisionsList" class="divide-y divide-gray-200"></ul>
          </div>
        </div>
    `;

    // Add event listener to logout button
    document.getElementById('logoutBtn').addEventListener('click', handleLogout);

    // Render applications lists
    renderPendingApplicationsList();
    renderRecentDecisionsList();
    lucide.createIcons();
}

function renderApplicationsList() {
    const applicationsList = document.getElementById('applicationsList');
    const userApplications = applications.filter(app => app.studentName ===
currentUser.username);
    if (userApplications.length === 0) {
        applicationsList.innerHTML = '<p>No applications submitted yet.</p>';
    } else {
        applicationsList.innerHTML = userApplications.map(app => `
            <li class="py-4">
                <div class="flex justify-between">
                    <div>
                        <p class="font-semibold">${app.reason}</p>
                        <p class="text-sm text-gray-500">${app.date}</p>
                    </div>
                    <span class="px-2 inline-flex text-xs leading-5 font-semibold rounded-full ${
                        app.status === 'approved' ? 'bg-green-100 text-green-800' :
                        app.status === 'rejected' ? 'bg-red-100 text-red-800' :
                        'bg-yellow-100 text-yellow-800'
                }">
                    ${app.status.charAt(0).toUpperCase() + app.status.slice(1)}
                </span>
            </div>
            ${app.feedback ? `<p class="mt-1 text-sm text-gray-600">Feedback:
${app.feedback}</p>` : ''}
        </li>
      `).join('');
    }
}

function renderPendingApplicationsList() {
    const pendingApplicationsList = document.getElementById('pendingApplicationsList');
    const pendingApplications = applications.filter(app => app.status === 'pending');

    if (pendingApplications.length === 0) {
        pendingApplicationsList.innerHTML = '<p>No pending applications.</p>';
    } else {

pendingApplicationsList.innerHTML = pendingApplications.map(app => `
        <li class="py-4">
            <div class="flex justify-between items-start">
                <div>
                    <p class="font-semibold">${app.studentName}</p>
                    <p class="text-sm text-gray-500">${app.reason} - ${app.date}</p>
                    ${app.proofFile ? `
                        <a href="#" class="text-blue-600 hover:underline flex items-center mt-1">
                            <i data-lucide="file-text" class="w-4 h-4 mr-1"></i> View Proof
                        </a>
                    ` : ''}
                </div>
                <div class="flex space-x-2">
                    <button onclick="handleApprove('${app.id}')" class="bg-green-500 hover:bg-
green-700 text-white font-bold py-2 px-4 rounded flex items-center">
                        <i data-lucide="check-circle" class="w-4 h-4 mr-1"></i> Approve
                    </button>
                    <button onclick="handleReject('${app.id}')" class="bg-red-500 hover:bg-red-
700 text-white font-bold py-2 px-4 rounded flex items-center">
                        <i data-lucide="x-circle" class="w-4 h-4 mr-1"></i> Reject
                    </button>
                </div>
            </div>
        </li>
        `).join('');
    }
    lucide.createIcons();
}

function renderRecentDecisionsList() {
    const recentDecisionsList = document.getElementById('recentDecisionsList');
    const recentDecisions = applications.filter(app => app.status !== 'pending');

    if (recentDecisions.length === 0) {
        recentDecisionsList.innerHTML = '<p>No recent decisions.</p>';
    } else {
        recentDecisionsList.innerHTML = recentDecisions.map(app => `
        <li class="py-4">
            <div class="flex justify-between">
                <div>
                    <p class="font-semibold">${app.studentName}</p>
                    <p class="text-sm text-gray-500">${app.reason} - ${app.date}</p>
                </div>
                <span class="px-2 inline-flex text-xs leading-5 font-semibold rounded-full ${
                    app.status === 'approved' ? 'bg-green-100 text-green-800' : 'bg-red-100 text-red-
800'
```

```
                }">
                    ${app.status.charAt(0).toUpperCase() + app.status.slice(1)}
                </span>
            </div>
            ${app.feedback ? `<p class="mt-1 text-sm text-gray-600">Feedback:
${app.feedback}</p>` : ''}
        </li>
    `).join('');
  }
}

// Event handlers
function handleLogin(e) {
  e.preventDefault();
  const username = document.getElementById('username').value;
  const password = document.getElementById('password').value;
  const role = document.getElementById('role').value;

  // In a real application, you would validate credentials here
  currentUser = { username, role };
  localStorage.setItem('user', JSON.stringify(currentUser));

  if (role === 'student') {
    renderStudentDashboard();
  } else {
    renderStaffDashboard();
  }
}

function handleLogout() {
  localStorage.removeItem('user');
  currentUser = null;
  renderLogin();
```

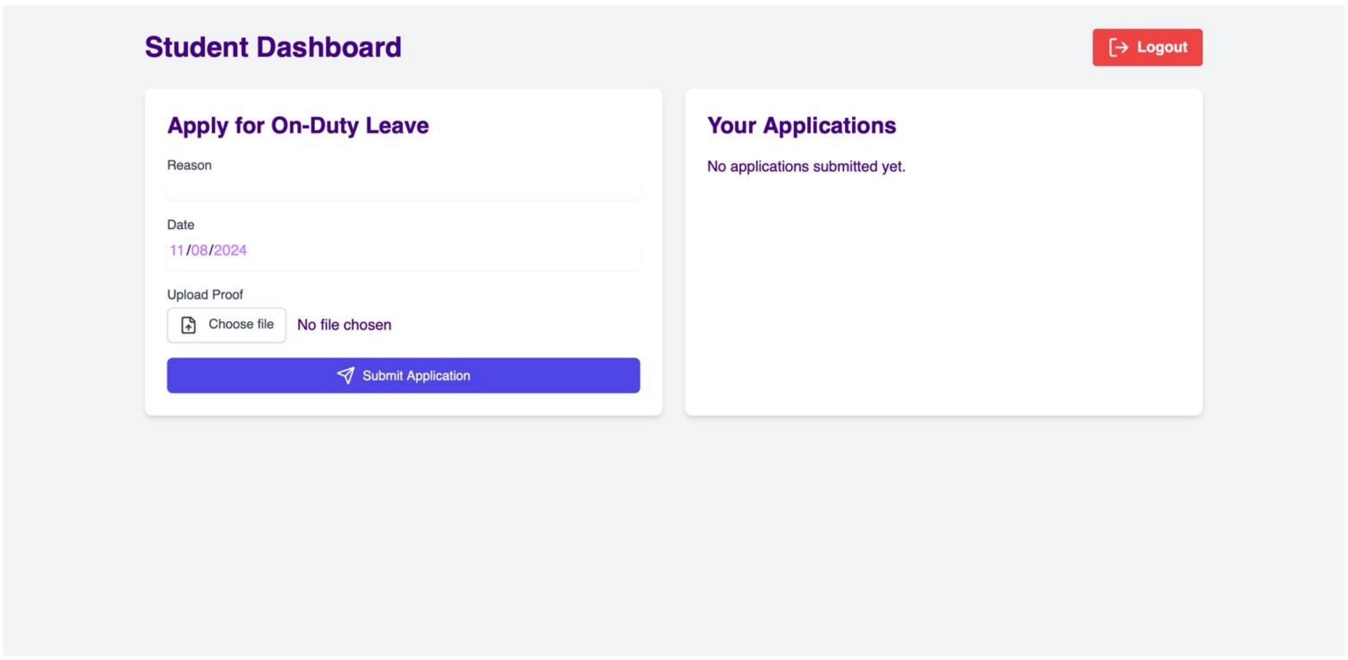
```
}

function handleApplicationSubmit(e) {
  e.preventDefault();
  const reason = document.getElementById('reason').value;
  const date = document.getElementById('date').value;
  const proofFile = document.getElementById('proof').files[0];

  const newApplication = {
    id: Date.now().toString(),
    studentName: currentUser.username,
    reason,
    date,
    proofFile: proofFile ? proofFile.name : null,
    status: 'pending'
  };

  applications.push(newApplication);
  renderApplicationsList();

  // Reset form
  e.target.reset();
  document.getElementById('fileName').textContent = 'No file chosen';

  alert('Application submitted successfully!');
}

function handleFileChange(e) {
  const fileName = e.target.files[0] ? e.target.files[0].name : 'No file chosen';
  document.getElementById('fileName').textContent = fileName;
}

function handleApprove(id) {
    applications = applications.map(app =>
        app.id === id ? { ...app, status: 'approved', feedback: 'Application approved.' } : app
    );
    renderPendingApplicationsList();
    renderRecentDecisionsList();
}

function handleReject(id) {
    const feedback = prompt('Enter rejection reason:');
    if (feedback) {
        applications = applications.map(app =>
            app.id === id ? { ...app, status: 'rejected', feedback } : app
        );
        renderPendingApplicationsList();
        renderRecentDecisionsList();
    }
}
// Initial render
function init() {
    const storedUser = localStorage.getItem('user');
    if (storedUser) {
        currentUser = JSON.parse(storedUser);
        if (currentUser.role === 'student') {
            renderStudentDashboard();
        } else {
            renderStaffDashboard();
        }
    } else {
        renderLogin();
    }
}
init();
```

CODE FOR CREATING INDEX USING HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>On-Duty Application</title>
    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
rel="stylesheet">
    <link rel="stylesheet" href="styles.css">
</head>
<body class="bg-gray-100 min-h-screen">
    <div id="app" class="container mx-auto p-4"></div>
    <script src="https://unpkg.com/lucide@latest"></script>
    <script src="app.js"></script>
</body>
</html>
```

PROJECT SCREENSHOTS

LOGIN PAGE-STUDENT DASHBOARD



Student Dashboard

Logout

Apply for On-Duty Leave

Reason

Date

11/08/2024

Upload Proof

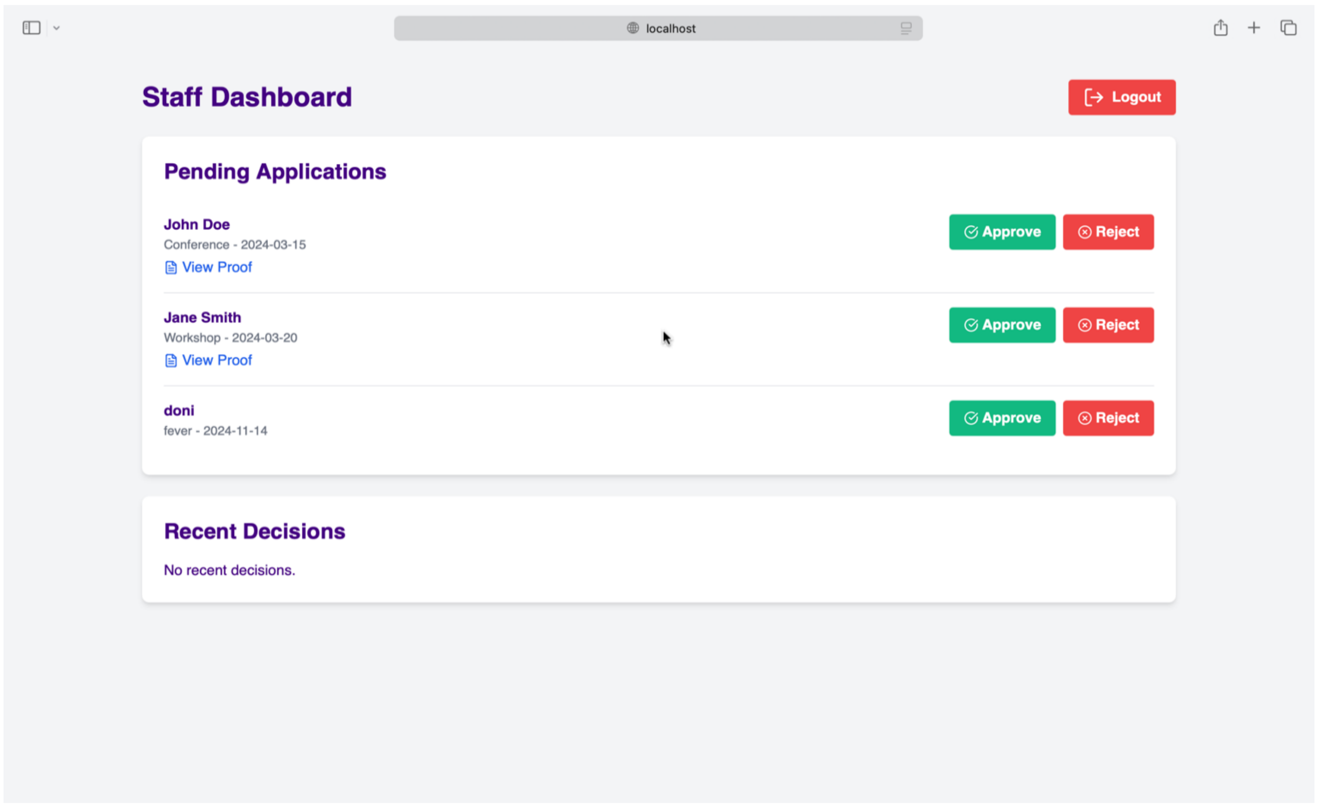
Choose file No file chosen

Submit Application

Your Applications

No applications submitted yet.

STAFF DASHBOARD



localhost

Staff Dashboard

Logout

Pending Applications

John Doe
Conference - 2024-03-15
View Proof
Approve Reject

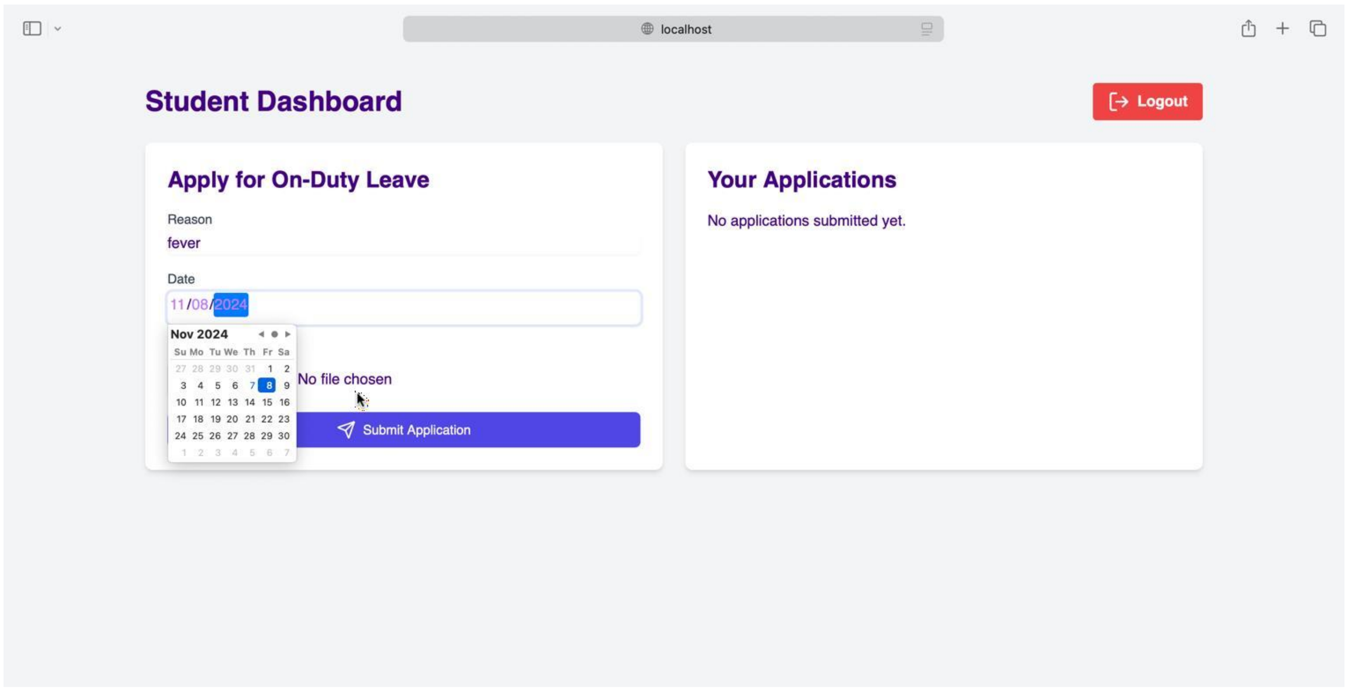
Jane Smith
Workshop - 2024-03-20
View Proof
Approve Reject

doni
fever - 2024-11-14
Approve Reject

Recent Decisions

No recent decisions.

CALENDAR USER INTERFACE


localhost

Student Dashboard

Logout

Apply for On-Duty Leave

Reason

fever

Date

11/08/2024

Nov 2024

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

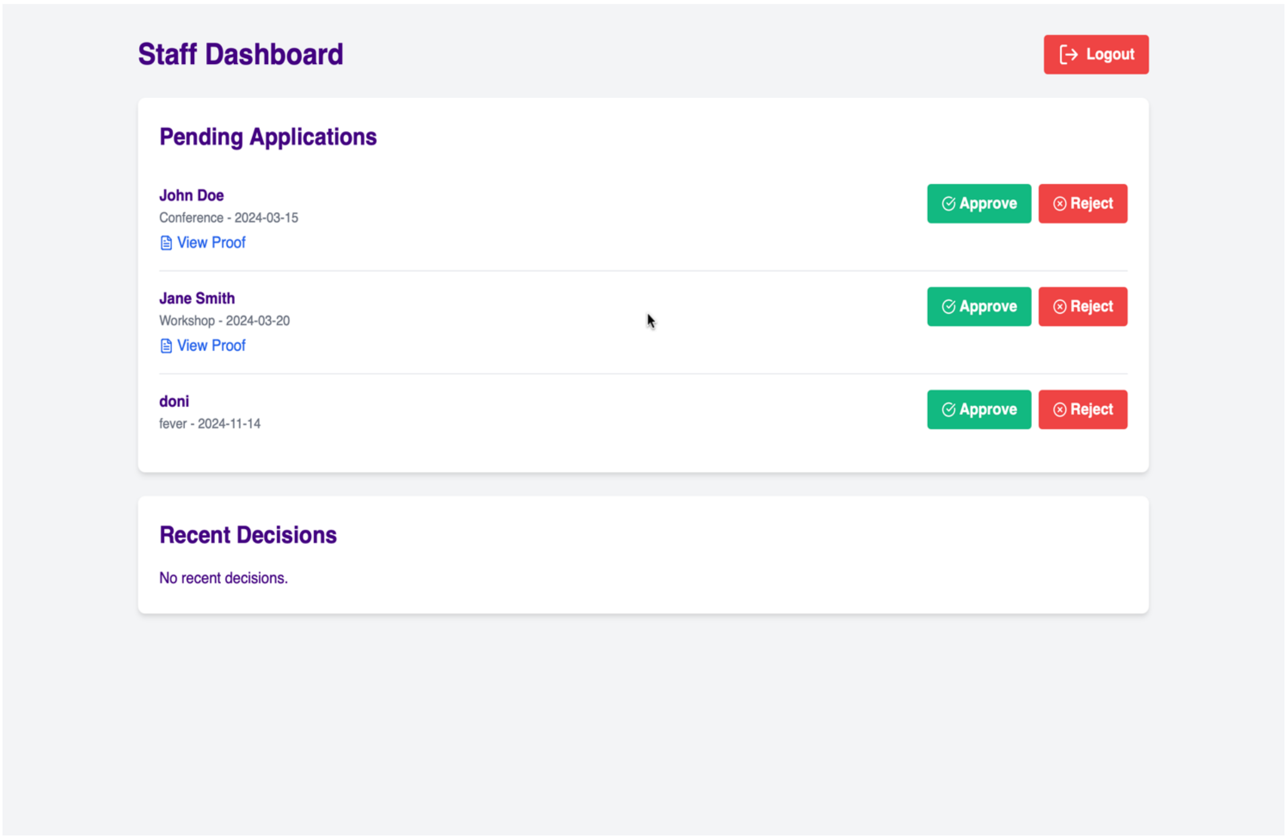
No file chosen

Submit Application

Your Applications

No applications submitted yet.

STAFF DASHBOARD FOR OD APPROVAL


Staff Dashboard

Logout

Pending Applications

John Doe
Conference - 2024-03-15
View Proof
Approve Reject

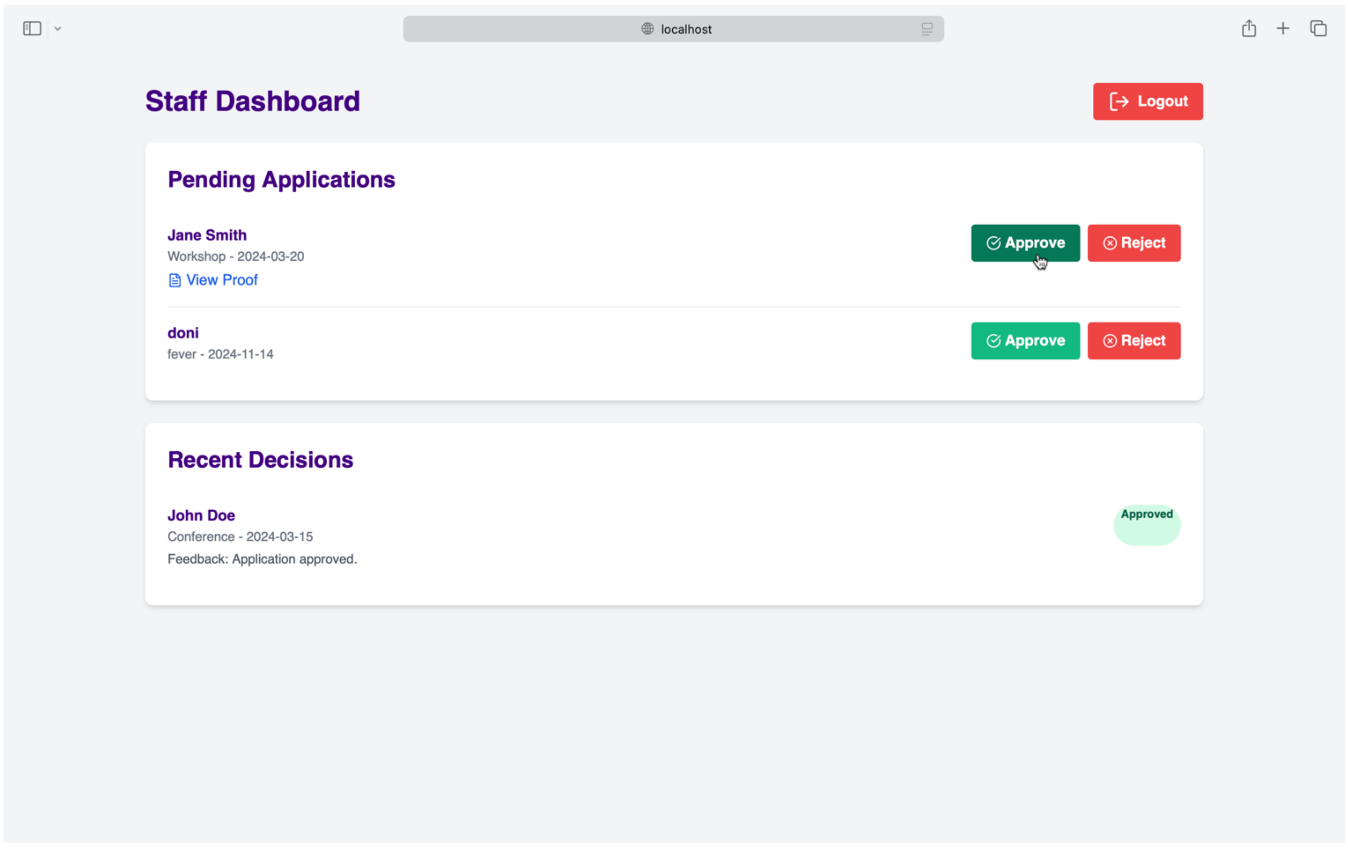
Jane Smith
Workshop - 2024-03-20
View Proof
Approve Reject

doni
fever - 2024-11-14
Approve Reject

Recent Decisions

No recent decisions.

OD APPROVAL INTERFACE


localhost

Staff Dashboard

Logout

Pending Applications

Jane Smith
Workshop - 2024-03-20
View Proof

Approve Reject

doni
fever - 2024-11-14

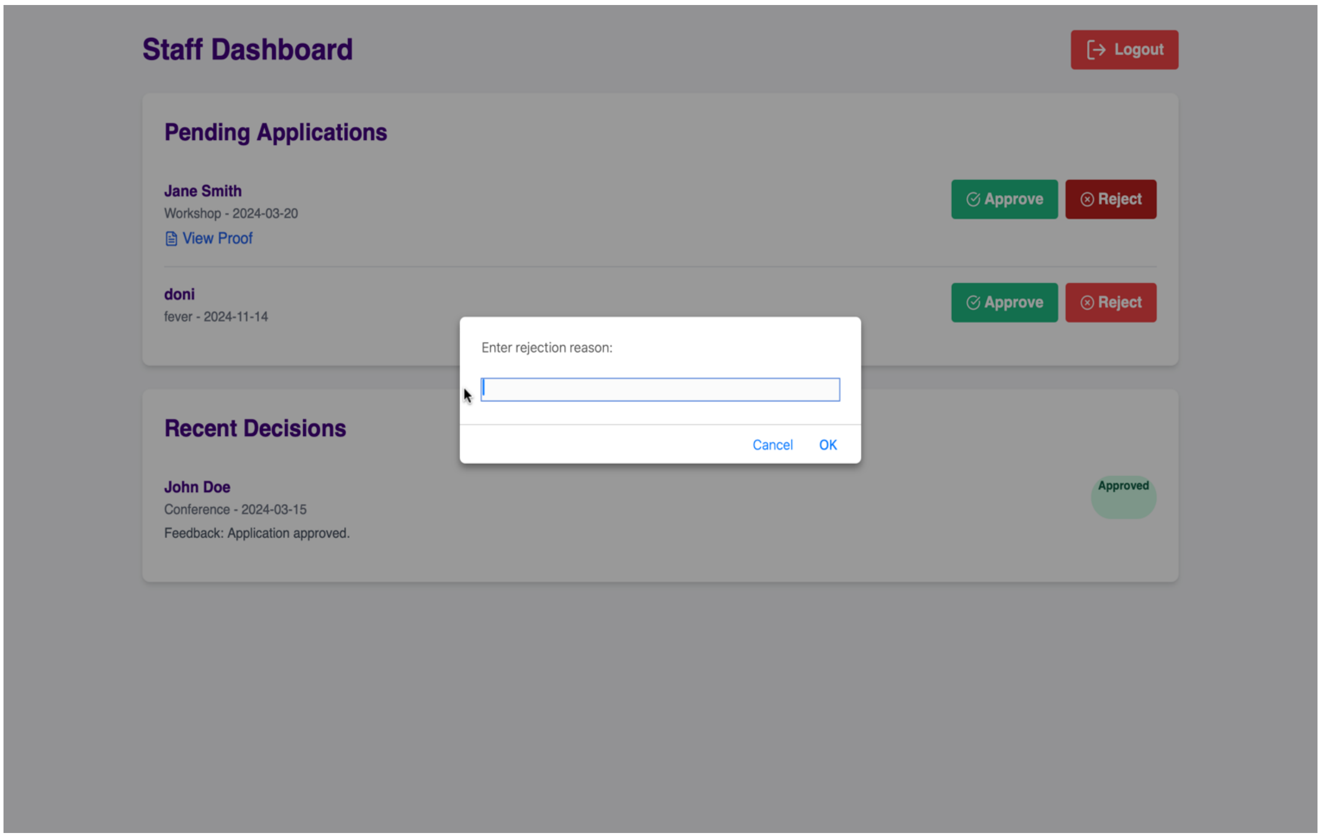
Approve Reject

Recent Decisions

John Doe
Conference - 2024-03-15
Feedback: Application approved.

Approved

OD REJECTION INTERFACE


Staff Dashboard

Logout

Pending Applications

Jane Smith
Workshop - 2024-03-20
View Proof

Approve Reject

doni
fever - 2024-11-14

Approve Reject

Enter rejection reason:

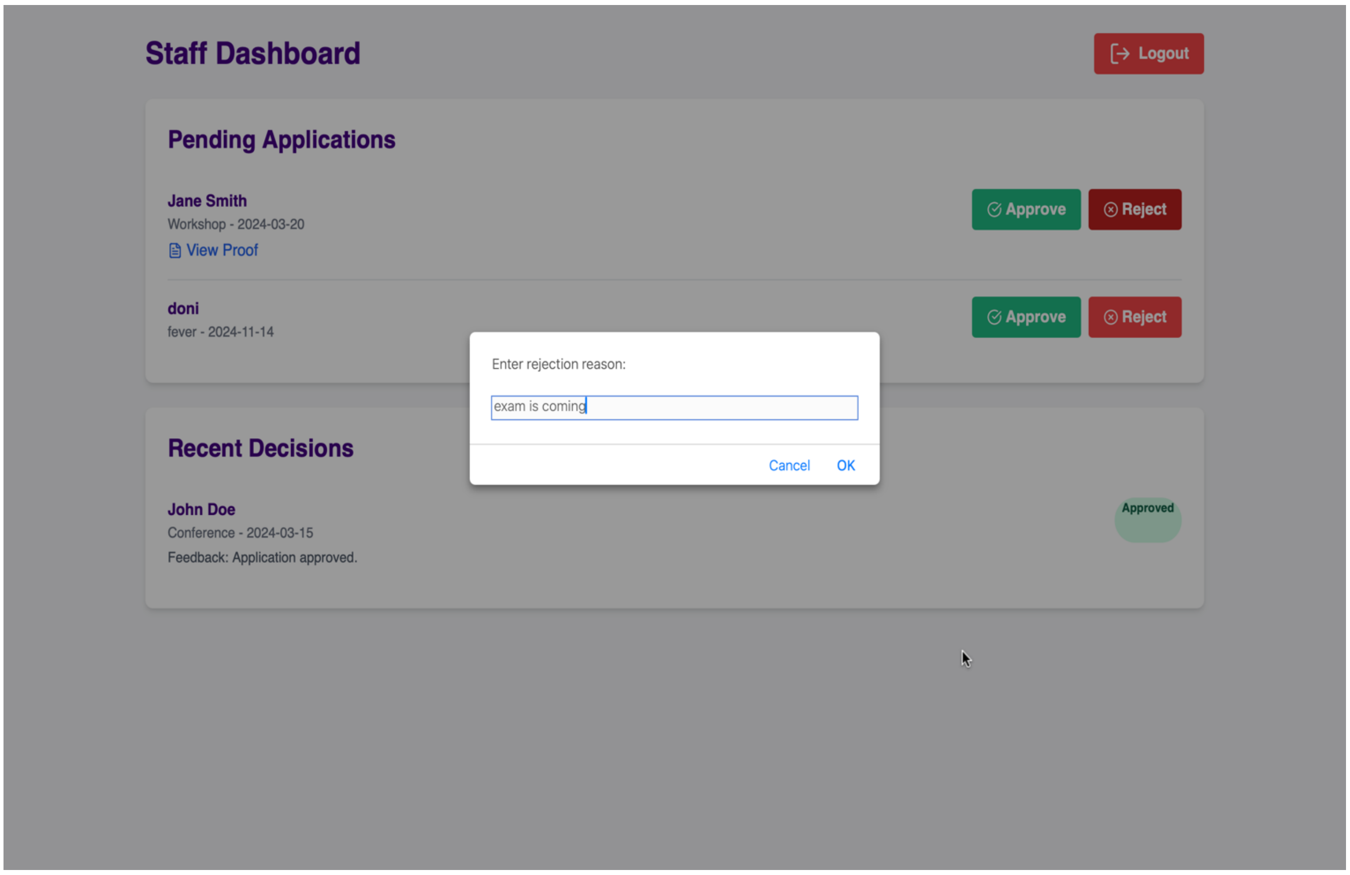
Cancel OK

Recent Decisions

John Doe
Conference - 2024-03-15
Feedback: Application approved.

Approved

OD REJECTION REASON UI



Staff Dashboard

Logout

Pending Applications

Jane Smith
Workshop - 2024-03-20
View Proof

Approve Reject

doni
fever - 2024-11-14

Approve Reject

Enter rejection reason:

exam is coming

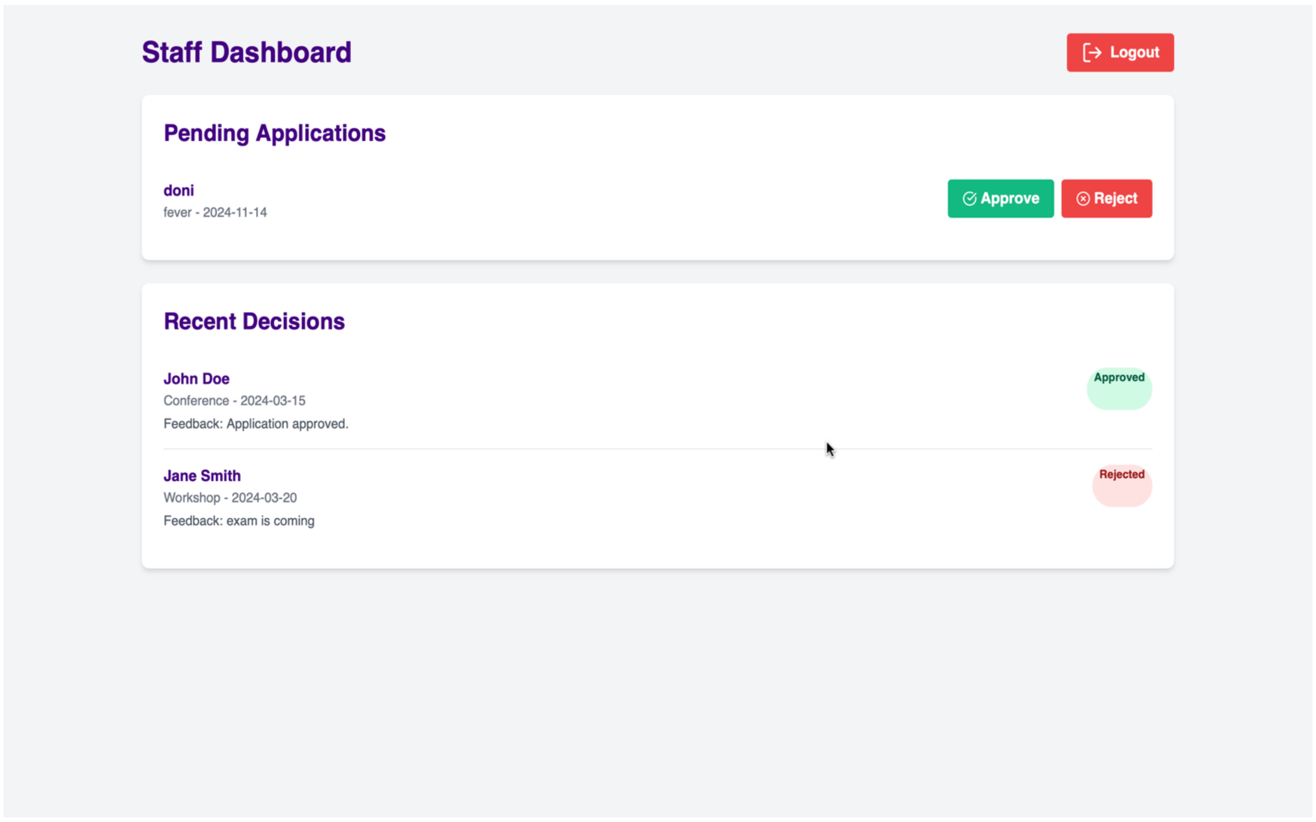
Cancel OK

Recent Decisions

John Doe
Conference - 2024-03-15
Feedback: Application approved.

Approved

RECENT APPROVED/REJECTED INTERFACE



Staff Dashboard

Logout

Pending Applications

doni
fever - 2024-11-14

Approve Reject

Recent Decisions

John Doe
Conference - 2024-03-15
Feedback: Application approved.

Approved

Jane Smith
Workshop - 2024-03-20
Feedback: exam is coming

Rejected

RESULTS AND DISCUSSION

5.1 OBSERVATIONS

- The System Successfully Allows Students To Submit On-Duty (Od) Requests And Faculty To Manage And Approve Them Through An Intuitive Interface.
- Real-Time Notifications And Status Tracking Improve Communication And Transparency Between Students And Faculty.
- The User-Friendly Gui Provides A Seamless Experience For Both Students And Faculty In Managing Od Requests.
- The Application Allows Students To Select Dates From An Integrated Calendar, Simplifying The Process And Reducing Errors.
- Faculty Can Review And Approve Or Decline Requests With Just A Click, Improving Efficiency In The Approval Process.

5.2 LIMITATIONS

- The Application Relies On Stable Internet Connectivity For Real-Time Notifications And The Smooth Functioning Of The Database.
- There Could Be Occasional Delays In Notifications Due To Server Or Network Issues.
- User Experience Can Vary Across Different Devices Or Browsers, Requiring Optimization For Maximum Compatibility.
- Depending On The Implementation, The System Might Face Minor Scalability Challenges During High-Demand Periods.

5.3 FUTURE IMPROVEMENTS

- **MOBILE APP INTEGRATION**: Extend The Eduease Platform To Mobile Devices, Providing Students And Faculty With On-The-Go Access To Od Requests, Approvals, And Notifications.
- **CLOUD DATABASE**: Implement A Cloud-Based Solution For Storing Data To Improve Scalability, Security, And Ease Of Access Across Different Locations And Platforms.

- **ADVANCED NOTIFICATION SYSTEM**: Integrate More Advanced Notification Features Such As Email Alerts, Sms, And Push Notifications To Ensure Timely Communication.
- **ADDITIONAL LEAVE CATEGORIES**: Extend The Leave Management System To Accommodate Various Leave Types Beyond Od Requests (E.G., Medical Leave, Personal Leave).
- **AI-POWERED ANALYTICS**: Implement Analytics Tools That Can Help Both Faculty And Students Track Leave Patterns And Improve Decision-Making Regarding Attendance And Leave Policies.
- **FACE MASK DETECTION**: Include Face Mask Detection Features To Ensure The System Remains Functional In Post-Pandemic Scenarios, Improving Security And Adaptability.

ADDITIONAL ANALYTICS

- Additionally, Feedback From Initial Users Indicates High Satisfaction With The Ease Of Use And Reliability Of The System. The Streamlined Workflow Reduces Administrative Burden And Allows Faculty To Focus More On Teaching And Less On Paperwork. Students Appreciate The Transparency And The Ability To Track Their Request Status In Real-Time, Which Reduces Anxiety And Uncertainty About Their Leave Requests.
- The Eduease Project Has Successfully Addressed Many Of The Challenges Associated With Traditional Leave Management Processes In Educational Settings. With Future Improvements, Eduease Has The Potential To Become An Indispensable Tool For Educational Institutions, Providing A Seamless, Efficient, And User-Friendly Solution For Managing Attendance And Leave Requests.

TESTING

6.1 UNIT TESTING:

Unit Testing In The Eduease Project Focuses On Verifying The Functionality Of Individual Components Of The Application, Such As The Student Request Submission Form, Faculty Approval Panel, Notification System, And Database Interactions. Each Module Is Tested Independently To Ensure That It Performs Its Specific Function Correctly. For Example, The Form Where Students Submit Their On-Duty (OD) Requests, The Logic For Date Selection, And The Validation Of Reason Input Are All Tested Separately To Confirm That They Work As Expected.

6.2 INTEGRATION TESTING:

Integration Testing For Eduease Involves Testing The Interaction Between Different Modules That Have Already Passed Unit Testing. This Includes Verifying That The Submission Of An OD Request Correctly Triggers The Faculty Approval Process And That The Notification System Sends Real-Time Updates To Both Students And Faculty. Integration Testing Ensures That Data Flows Seamlessly Between The Front-End (React-Based Interface), Back-End (Mongodb), And The Notification System, As Well As The Interactions Between Student And Faculty Accounts.

6.3 SYSTEM TESTING:

System Testing Involves Testing The Entire Eduease Application As A Whole. The Goal Is To Ensure That The System Meets The Functional And Non-Functional Requirements, Such As User Experience, Security, And Scalability. During System Testing, The Application Is Installed On Different Operating Systems And Environments To Ensure Compatibility And Stability. Potential Errors, Bugs, Or System Crashes Are Identified And Resolved, Ensuring The Platform Is Reliable And Performs As Expected Under Various Conditions.

6.4 ACCEPTANCE TESTING:

User Acceptance Testing (UAT) In Eduease Involves The Client Or Administrators (E.G., Educational Institution Representatives) Testing The Application To Verify That It Meets The Agreed-Upon Requirements. The UAT Process Focuses On Checking Whether The Features, Such As OD Request Submission, Approval Process, Notifications, And Record Management, Function According To The Specifications. Feedback From The Users During This Phase Helps To Ensure That The Final Product Aligns With User Needs And Is Ready For Deployment In The Real-World Environment Of The Educational Institution. This Testing Phase Occurs Before The Final Release And **Deployment Of Eduease.**

OVERALL PERFORMANCE

To Further Enhance The Robustness And Reliability Of The Eduease Application, It Is Recommended To Implement **Performance Testing** And **Security Testing**. Performance Testing Ensures That The Application Can Handle High Loads And Stress Conditions Without Degrading The User Experience. Security Testing Identifies Potential Vulnerabilities And Ensures That The Application Is Protected Against Common Security Threats Such As SQL Injection, Cross-Site Scripting (XSS), And Data Breaches. Implementing These Additional Testing Methodologies Will Contribute To A More Resilient And Secure Eduease Platform, Ultimately Leading To Higher User Satisfaction And Trust.

CONCLUSION

The Eduease Project Represents A Significant Advancement In Automating And Modernizing The Leave Management Process For Educational Institutions. By Leveraging A User-Friendly Interface And An Automated Workflow, Eduease Reduces Reliance On Traditional, Paper-Based Systems, Thereby Streamlining The Submission, Review, And Approval Of On-Duty (OD) Requests. This Platform Not Only Simplifies The Request Process For Students But Also Empowers Faculty With Efficient Tools For Timely Decision-Making And Record Management.

The Application's Real-Time Notification And Status-Tracking Features Enhance Transparency And Communication Between Students And Faculty, Reducing Delays And Misunderstandings. With A Scalable And Secure Backend, Eduease Is Designed To Handle The Demands Of Large Student Bodies And Ensure Data Privacy, Making It Well-Suited For Use In A Variety Of Educational Environments.

In The Future, Eduease Could Be Expanded With Additional Features Like Mobile App Compatibility, Analytics Dashboards, And AI-Driven Insights Into Leave Patterns. Overall, This Project Is A Valuable Tool That Addresses The Unique Challenges Of Attendance And Leave Management In Academic Settings, Fostering A More Organized And Responsive Campus Environment.

Seamless Integration With Existing Learning Management Systems (LMS) And Student Information Systems (SIS) Would Create A Unified Educational Ecosystem. Development Of Advanced Analytics Tools Could Provide Deeper Insights Into Student Attendance Patterns, Helping Institutions To Identify And Address Potential Issues Proactively. Creating A Mobile Application For Easier Access And On-The-Go Management Of OD Requests Would Ensure Convenience For Students And Faculty Alike. Offering Multi-Language Support Would Cater To Diverse Student Populations, Making The Application More Accessible And Inclusive. Establishing A Feedback Mechanism For Students And Faculty To Suggest Improvements And Report Issues Would Facilitate Continuous Enhancement Based On User Needs.

REFERENCES

REFERENCES TEXTBOOKS:

- Database System Concepts(6th Edition) By Abraham Silberschatz, Henry F. Korth,S. Sudarshan.

WEBSITES:

- Www.Geeksforgeeks.Com
- Www.W3schools.Com

REFERENCE VIDEOS :

- **HTML AND CSS TIPS AND TRICKS:**

  Https://Www.Youtube.Com/Watch?V=Aactxswxsro&List=PL4-

- **HTML AND CSS FOR BEGINNERS:**

  https://www.youtube.com/watch?v=FYErehuSuuw

- **REACT.JS FOR BEGINNERS :**

  https://www.youtube.com/watch?v=dz458ZkBMak

- **JAVASCRIPT :**

  https://www.youtube.com/watch?v=poo0BXryffI&t=4s

GITHUB PAGE:

https://github.com/Thambu2005/DBMS-Project.git

RESEARCH PAPERS LINKS:

- **"AUTOMATED LEAVE MANAGEMENT USING E-LEAVE"**

  https://www.researchgate.net/publication/335123703_Gottebenbildlichkeit_Christologie_und_Geborenwerden

- **IEEE EXPLORE DIGITAL LIBRARY-" A WEB BASED LEAVE MANAGEMENT SYSTEM FOR UNIVERSITY "**

  https://ieeexplore.ieee.org/Xplore/home.jsp