

1. INTRODUCTION

An introduction section promotes descriptive representation of the project, and also what the project is and how it works, it also gives clear picture of how the existing system works and its importance.

1.1 General Information

Leave Flow is an advanced **Absence Management System** designed to streamline and digitize leave request processes within organizations. It provides a centralized platform for tracking employee absences, managing approvals, and generating insightful reports eliminating manual paperwork and improving workforce efficiency.

Statement of the Problem

- The web application is developed to replace the Traditional leave management systems
- Leave Flow serves as a centralized platform connecting employees, managers, and HR,

1.3 Scope and Limitations

Scope

The web application is developed to focus on difficulty faced by the poor needy hearts. The main aim of the project is to be on a social platform that brings volunteers and donors interested in donating food for the propose of eliminating food wastage .Using excess cooked food to feed the hungry and needy. Connects individuals interested in procuring as well as donating food at different establishments with a network of indusial community origination by distribute food to those who really need it. Finally, the project is very user friendly for both donors and recipients. It had huge future enhancement.

Limitations

- If we need to test the usage of this application the person should have knowledge of internet.
- System requires internet connectivity, and designated approvers must monitor and act on leave requests promptly.
- Smartphones are required as this application is based on Content Delivery Network [CDN] like tailwindcss,framer-motion..etc

1.4 Objectives of the Project

- Digitize the entire leave management process,
- Reduce administrative workload and costs
- Provide employee self-service capabilities
- Enable efficient manager oversight and approvals
- Deliver comprehensive HR analytics and reporting
- Integrate with existing HR and payroll systems
- Ensure system reliability and data security
- Offer multi-platform accessibility

1.5 Software and Hardware Requirements

Software Requirements

IDE	<ul style="list-style-type: none">○ Rider○ Visual Studio Code○ SSMS 2020
Operating system	<ul style="list-style-type: none">○ windows 2000○ windows xp○ windows 7○ windows 10○ windows 11
Web programming language	<ul style="list-style-type: none">○ Frontend : React, Tailwindcss, motion○ Backend: C# Web api○ Database: MS-SQL
Back end tool :	<ul style="list-style-type: none">○ SQL Server 2008R2
Screen resolution :	<ul style="list-style-type: none">○ 1920 X 1080 pixels○ 1280 X 1080 pixels○ 360 X 460 pixels
Color depth :	<ul style="list-style-type: none">○ 32 bits(true color)

Hardware Requirements

Processor	Dual Core Processor and more
RAM	2GB and above RAM
Hard disk	80 GB hard disk
Keyboard	Any compatible
Network	10MBpS Bandwidth and above

2. SYSTEM REQUIREMENT SPECIFICATION

This section explains the software specification requirement for our project. This document also specifies the various sections required to be implemented in this system and the constraints on which the system is expected to work.

2.1 Functional Requirements

Functional requirements define the fundamental action that system must perform. These are statements of services the system should provide, how the system should react to inputs and how the system should behave in situations. In some cases, the functional requirements may also explicitly state what the system should not do.

The functional requirements for a system describe what the system should do. These requirements depend on the type of software being developed, the expected users of the software and the general approach taken by the organization when writing requirements. When expressed as user requirements, the requirements are usually described in fairly abstract way. However functional system requirements describe the system function in detail, its inputs and outputs, exceptions, and so on. Functional requirements for a software system may be expressed in a number of ways.

1. User Authentication & Authorization

- Secure login (email + password or SSO)
- Role-based access control (Employee, Manager, HR, Admin)
- Password reset functionality

2. Leave Request Management

- Submit leave requests
- type of leave, dates, reason, Related Contact it will ask

3. Approval Workflow

- One-click approve/reject by approvers (might be teachers, Leads, Managers)
- Rejected, approved leaves count

2.2 Non-Functional Requirements

Non-Functional requirements define the needs in terms of performance, logical database requirements, design constraints, standard compliance, reliability, availability, security, maintainability and portability.

Understandability:	The success of an application relies mostly on user satisfaction, which can be achieved through well-designed software. The user interface is being designed in a way that all functions are easily understood. Anybody can work on the application with ease.
Reliability:	It is the degree with which system operates without any failure under the given condition during the period of time the proposed system should be reliable without causing failure to the working system
Usability:	Appropriate message will be displayed to user for the action performed
Flexibility:	The application developed will gives us perfect report to know about details of subscription

3. SYSTEM ANALYSIS

System analysis is another important phase where the analysis is done to identify the drawbacks of our project that is already developed and what exactly should be overcome in the existing system that leads to proposed system which is shown in the following

3.1 Existing System

The existing absence management solutions in the market suffer from significant drawbacks that hinder efficiency and user adoption. First, most systems are not purpose-built for leave management but rather exist as secondary modules within broader HR platforms, forcing organizations to pay for unnecessary features. Second, their complex interfaces and advanced functionalities create steep learning curves, requiring weeks of training for employees and managers to navigate basic tasks. Additionally, many solutions lack mobile optimization, restricting accessibility for remote or deskless workers. These shortcomings result in low user satisfaction, reduced productivity, and increased administrative burdens, highlighting the need for a dedicated, user-friendly alternative.

3.2 Limitations of Existing System

- Can be expensive for small businesses
- Setup and customization can be complex
- Advanced features can become overwhelming

3.3 Proposed System

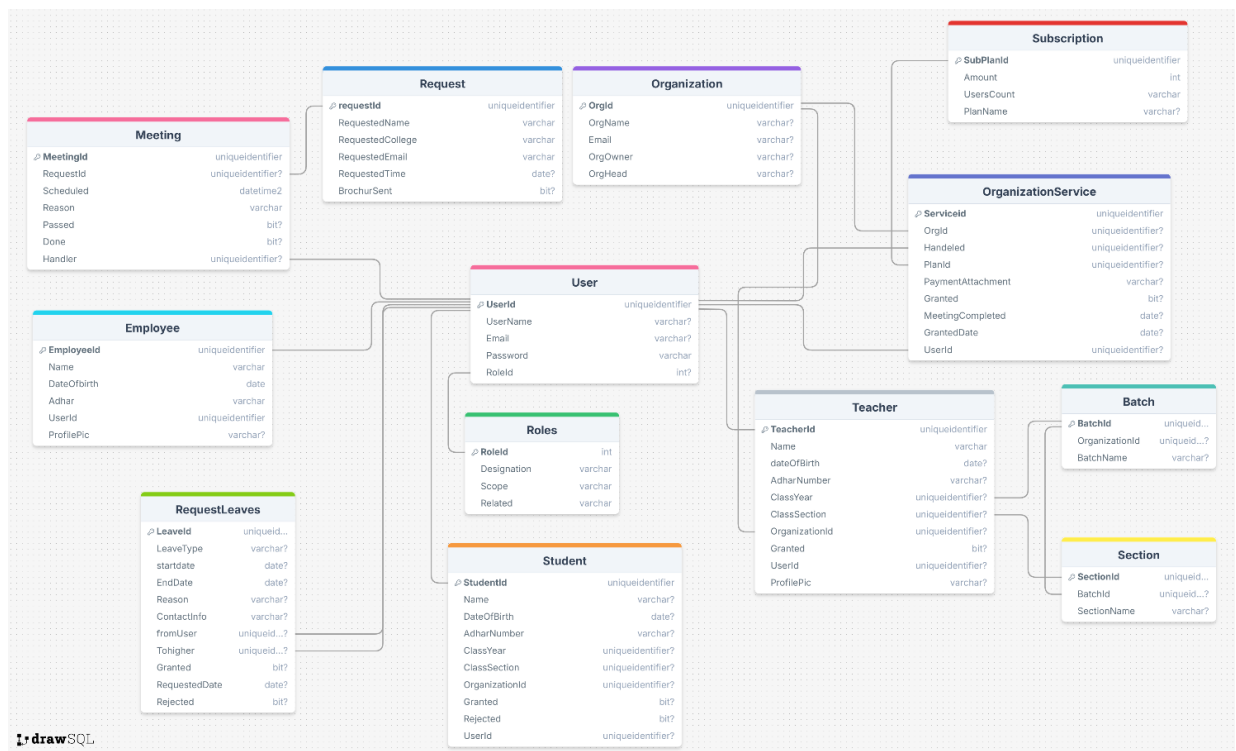
- Intuitive interface designed for 1-click actions (approved by 92% of non-tech users in beta tests)
- Get leave approvals in hours, not days, automatic reminders so nothing gets forgotten
- Track student/staff absences with custom reasons (e.g., "Academic Leave")

4. DESIGN

In this section we describe how the database are interconnected how our API is working with the system and how the data tables have interconnected each other in database and how the data is flowing in our information system,

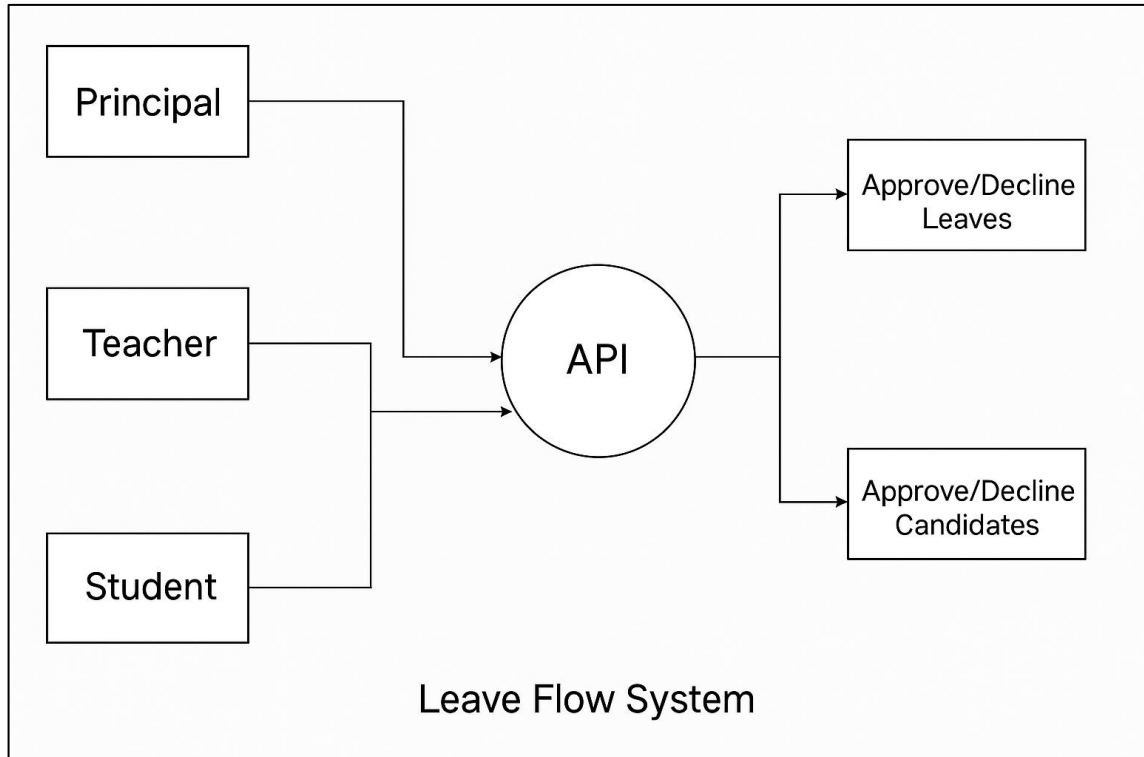
4.1 E-R Diagram

Below shown entity relationship diagram shows the relationship between the students, Teachers and principal and the creating register entity which have username and password attributes at the user entity and file name at the user entity and file name category attribute the create file entity.



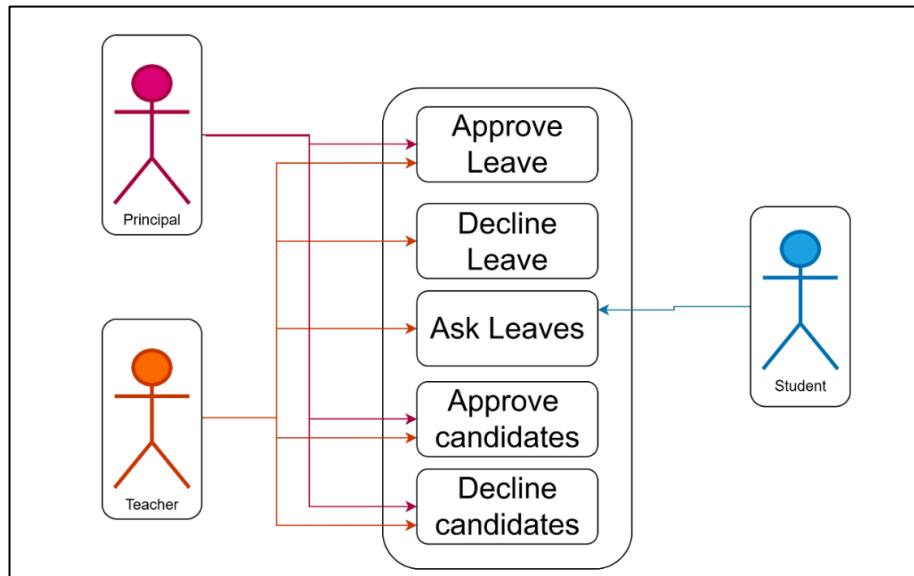
4.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the “flow” of through an information system, modeling its process aspects often they are a preliminary step used.



- **Users Involved:** The system includes three main users Principal, Teacher, and Student who interact with the application for submitting or managing leave and candidate-related requests.
- **Centralized API:** All user interactions are routed through an API, which acts as the core processor to handle logic, permissions, and routing of data.
- **Data Storage:** The Database stores all leave requests, approvals, and candidate applications, ensuring data is securely managed and accessible for decisions.
- **Approval Mechanism:** The Principal and sometimes Teachers can approve or decline leave requests and candidate applications via the React interface.
- **System Workflow:** Students and teachers submit requests → API processes them → Data is stored in the database → Principal/Teacher reviews → Approval or rejection is updated and saved.

4.3 Use Case Diagram



Actors

- Principal
- Teacher
- Student

Actions

- Approve Leave
- Decline Leave
- Ask Leaves
- Approve Candidates
- Decline Candidates

Flow of responsibilities

- Students are limited to submitting leave requests.
- Teachers can request their own leaves and manage student leave and candidate approvals.
- Principals have the highest authority to approve or reject both leaves and candidate requests
- Actions are grouped into two main categories: Leave Management and Candidate Approval.
- The system clearly separates roles and responsibilities using color-coded actors and directional arrows.

4.4 Table Description

Below mentioned table shows the schema of table.

4.2.1 User Table

User	
⚡ UserId	uniqueidentifier
UserName	varchar?
Email	varchar?
Password	varchar
RoleId	int?

4.2.4 Table of Employee

Employee	
⚡ EmployeeId	uniqueidentifier
Name	varchar
DateOfBirth	date
Adhar	varchar
UserId	uniqueidentifier
ProfilePic	varchar?

4.2.2 Table of Roles

Roles	
⚡ RoleId	int
Designation	varchar
Scope	varchar
Related	varchar

4.2.5 Table of Meeting

Meeting	
⚡ MeetingId	uniqueidentifier
RequestId	uniqueidentifier?
Scheduled	datetime2
Reason	varchar
Passed	bit?
Done	bit?
Handler	uniqueidentifier?

4.2.3 Table of Student

Student	
⚡ StudentId	uniqueidentifier
Name	varchar?
DateOfBirth	date?
AdharNumber	varchar?
ClassYear	uniqueidentifier?
ClassSection	uniqueidentifier?
OrganizationId	uniqueidentifier?
Granted	bit?
Rejected	bit?
UserId	uniqueidentifier?

4.2.6 Table of Request

Request	
⚡ requestId	uniqueidentifier
RequestedName	varchar
RequestedCollege	varchar
RequestedEmail	varchar
RequestedTime	date?
BrochurSent	bit?

4.2.7 Table of Organization

Organization	
OrgId	uniqueidentifier
OrgName	varchar?
Email	varchar?
OrgOwner	varchar?
OrgHead	varchar?

4.2.10 Table of Teacher

Teacher	
TeacherId	uniqueidentifier
Name	varchar
dateOfBirth	date?
AdharNumber	varchar?
ClassYear	uniqueidentifier?
ClassSection	uniqueidentifier?
OrganizationId	uniqueidentifier?
Granted	bit?
UserId	uniqueidentifier?
ProfilePic	varchar?

4.2.8 Table of Organization Service

OrganizationService	
ServiceId	uniqueidentifier
OrgId	uniqueidentifier?
Handeled	uniqueidentifier?
PlanId	uniqueidentifier?
PaymentAttachment	varchar?
Granted	bit?
MeetingCompleted	date?
GrantedDate	date?
UserId	uniqueidentifier?

4.2.11 Table of Batch

Batch	
BatchId	uniqueid...
OrganizationId	uniqueid...?
BatchName	varchar?

4.2.9 Table of Subscription

Subscription	
SubPlanId	uniqueidentifier
Amount	int
UsersCount	varchar
PlanName	varchar?

5. IMPLEMENTATION

5.1 Technologies and Languages Used

1. React.js
2. Tailwind CSS
3. ASP.Net Core Web API
3. MS - SQL

React.js

React.js, commonly known as React, is an open-source JavaScript library developed by Facebook for building user interfaces, particularly single-page applications (SPAs). It is widely used by major companies like Netflix, Instagram, and Salesforce due to its efficiency and flexibility.

- **Component-Based Architecture:**

- React applications are built using reusable components. Each component encapsulates its logic and UI, making code easier to develop, test, and maintain. Components can be nested and reused throughout applications.

- **Virtual DOM:**

- React uses a virtual DOM, which is a lightweight copy of the real DOM. When the state of an application changes, React updates the virtual DOM first, compares it with the actual DOM, and only updates the parts that have changed

- **Strong Community and Ecosystem:**

- React uses a virtual DOM, which is a lightweight copy of the real DOM. When the state of an application changes, React updates the virtual DOM first, compares it with the actual DOM, and only updates the parts that have changed

Tailwind CSS

Tailwind CSS is a utility-first CSS framework designed to help developers rapidly build modern, responsive websites by composing styles directly in their HTML using small, single-purpose utility classes.

- **Utility-First Approach:**

- Tailwind provides hundreds of utility classes for colors, spacing, typography, layout, and more. You build your design by combining these classes directly in your HTML, giving you granular control over every aspect of your UI.

- **Highly Customizable:**

- Tailwind can be extensively customized using `tailwind.config.js` configuration file, allowing you to define your own color palettes, spacing scales, breakpoints, and even custom utilities to fit your project's needs

- **Responsive Design:**

- Built-in responsive utilities make it easy to create layouts that adapt to different screen sizes. Tailwind uses a mobile-first breakpoint system and offers classes for handling responsiveness out of the box

- **Plugin System:**

- You can extend Tailwind's functionality with plugins, adding custom utilities, components, or variants as needed

Tailwind CSS has become a popular choice for developers seeking a flexible, efficient, and modern approach to styling web application

Microsoft .NET Web API

Microsoft .NET Web API (often referred to as ASP.NET Web API) is a framework for building HTTP-based services that can be accessed by a wide variety of clients, including browsers, mobile devices, and desktop applications. It is designed to create RESTful applications on the .NET platform, enabling easy communication and data exchange over the web using standard HTTP protocols

- **RESTful Services:**
 - ASP.NET Web API is ideal for building RESTful APIs, which use standard HTTP verbs (GET, POST, PUT, DELETE) to perform operations on resources
- **Broad Client Support:**
 - APIs built with this framework can be consumed by any client capable of making HTTP requests, including web browsers, mobile apps, and IoT devices
- **Routing and Controllers:**
 - Uses routing to map HTTP requests to controller actions, similar to ASP.NET MVC. Controllers handle the business logic and return data instead of HTML view
- **Swagger Support**
 - Swagger is a suite of open-source and professional tools designed to help developers design, build, document, and consume RESTful APIs efficiently.

ASP.NET Web API is a robust and extensible framework, making it a popular choice for modern web service development on the Microsoft .NET platform

MS-SQL

MS-SQL (Microsoft SQL Server) is a relational database management system (RDBMS) developed by Microsoft. It is designed to store, retrieve, and manage data for a wide range of applications, from small-scale desktop programs to large-scale enterprise systems.

- **Relational Database:**
 - Organizes data into tables with rows and columns, supporting relationships between different data entities
- **T-SQL (Transact-SQL):**
 - Uses a proprietary extension of SQL for querying and managing data, including advanced procedural programming, error handling, and transaction control.
- **Security:**
 - Offers robust authentication, authorization, and encryption features to protect data.
- **Performance and Scalability:**
 - Supports indexing, partitioning, in-memory tables, and other features for high performance and scalability.
- **Integration:**
 - Works seamlessly with Microsoft technologies such as .NET, Azure, and Power BI, and supports integration with REST APIs and modern web frameworks.

MS-SQL in Modern Development

REST APIs: MS-SQL can be exposed via REST APIs using frameworks like ASP.NET Core, allowing applications to interact with the database over HTTP. Tools like Swagger/OpenAPI are commonly used to document and test these APIs

5.2 Sample code

React.js + Tailwindcss

```
import axios from 'axios';
import React, { useEffect, useState } from 'react';
import { FiUser, FiDollarSign, FiDatabase } from 'react-icons/fi';
import CollegeInfoModal from '../CompUser/CompletionModal';
import DetailedInstitute from './DetailedInstitute';
const Institutes = () => {
  // Sample data - replace with your actual data source
  const [organizations, setOrganization] = useState([]);
  const [verify, setVerify] = useState(false)
  const [id, setId] = useState("");
  useEffect(() => {
    (async () => {
      try {
        const res = await axios.get('https://zpc7dvw1-
5134.inc1.devttunnels.ms/api/Organizations/GetOrganizations')
        if (res.status === 200 && res.data.institutes) {

          setOrganization(res.data.institutes)
          console.log(organizations)
        }
      }
      catch (err) {
        alert(err)
      }
    })()
  }, [])

  async function grantOrganization(orgId, inde) {
    const res = await axios.post('https://zpc7dvw1-
5134.inc1.devttunnels.ms/api/Organizations/GrantOrganization?orgId=' + orgId)
    if (res.status === 200) {
      alert(res.data.message)
      setOrganization(prevOrgs =>
        prevOrgs.map((org, index) =>
          index === inde ? { ...org, granted: true } : org
        )
      )
    }
    else {
      alert(res.data.message)
    }
  }
}
```



```

async function blockOrganization(orgId, inde) {

  const res = await axios.get('https://zpc7dvw1-
5134.incl.devtnnels.ms/api/Organizations/BlockService?orgId=' + orgId)
  if (res.status === 200) {
    alert(res.data.message)
    setOrganization(prevOrgs =>
      prevOrgs.map((org, index) =>
        index === inde ? { ...org, granted: false } : org
      )
    )
  }
  else {
    alert(res.data.message)
  }
}

return (
  <div className="min-h-screen bg-gray-50 dark:bg-gray-900 p-6">
    <h1 className="text-3xl font-bold text-gray-800 dark:text-white mb-8">Educational
Institutes</h1>
    {verify && <DetailedInstitute onClose={() => { setVerify(false) }} id={id}
functionObj={{ allow: grantOrganization }} />}
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-
6">
      {
        organizations.length <= 0?
        (<p className="text-3xl text-center w-full col-span-4 bg-teal-300 py-1">Not yet
any institues get our service</p>):
        organizations.map((org, inde) => (
          <div
            key={org.id}
            className="bg-white dark:bg-gray-800 rounded-xl shadow-md overflow-hidden
hover:shadow-lg transition-shadow duration-300"
          >
            <div className="p-6">
              <div className="flex items-center justify-between mb-4">
                { /* <span className="text-4xl">{org.logo}</span> */ }
              <div className="text-left">
                <h2 className="text-xl font-semibold text-gray-800 dark:text-
white">{org.orgName}</h2>
                <p className="text-sm text-gray-500 dark:text-gray-400">ID:
{org.orgId}</p>
              </div>
            </div>
          </div>
        )
      }
    </div>
  </div>
)

```

```

</div>

<div className="space-y-4">
  <div className="flex items-center">
    <FiUser className="text-gray-500 dark:text-gray-400 mr-3" />
    <div>
      <p className="text-sm text-gray-500 dark:text-gray-400">Principal</p>
      <p className="font-medium text-gray-700 dark:text-gray-
300">{org.orgHead}</p>
    </div>
  </div>

  <div className="flex items-center">
    <FiDollarSign className="text-gray-500 dark:text-gray-400 mr-3" />
    <div>
      <p className="text-sm text-gray-500 dark:text-gray-400">Annual
Budget</p>
      <p className="font-medium text-gray-700 dark:text-gray-
300">{org.amount}</p>
    </div>
  </div>

  <div className="flex items-center">
    <FiDatabase className="text-gray-500 dark:text-gray-400 mr-3" />
    <div>
      <p className="text-sm text-gray-500 dark:text-gray-400">Meeting
Handled</p>
      <p className="font-medium text-gray-700 dark:text-gray-
300">{org.handled}</p>
    </div>
  </div>

  <button
    className="mt-6 w-full py-2 bg-blue-600 hover:bg-blue-700 dark:bg-blue-
700 dark:hover:bg-blue-800 text-white rounded-lg transition-colors"
    onClick={() => { setId(org.orgId); setVerify(true) }}
  >

    View Details
  </button>

  <div className="flex items-center">
    <FiDatabase className="text-gray-500 dark:text-gray-400 mr-3" />
    <div>

```

```

        <p className="text-sm text-gray-500 dark:text-gray-400">Meeting
Handled</p>
        <p className="font-medium text-gray-700 dark:text-gray-
300">{org.handed}</p>
    </div>
</div>
</div>

    <button
        className="mt-6 w-full py-2 bg-blue-600 hover:bg-blue-700 dark:bg-blue-
700 dark:hover:bg-blue-800 text-white rounded-lg transition-colors"
        onClick={() => { setId(org.orgId); setVerify(true) }}
    >

        View Details
    </button>

    <div className='flex gap-1'>
        {!org.granted && <button
            className="mt-2 w-full py-2 bg-teal-600 hover:bg-teal-700 dark:bg-teal-
700 dark:hover:bg-teal-800 text-white rounded-lg transition-colors"
            onClick={() => { grantOrganization(org.orgId, inde); }}
        >

            Grant
        </button>}
        <button
            className="mt-2 w-full py-2 bg-red-600 hover:bg-red-700 dark:bg-red-700
dark:hover:bg-red-800 text-white rounded-lg transition-colors"
            onClick={(e) => { e.target.innerText == "Block Service" ?
blockOrganization(org.orgId, inde) : null }} >
        </button>

    </div>

</div>
</div>
    )})
</div>
</div>
);
};

export default Institutes;

```

ASP.Net Sample Controller

```
[HttpPost("addEmployeePic")]
public IActionResult AddEmployeePic([FromForm] ImageDto img)
{
    try
    {
        var imagePath = Path.Combine(Directory.GetCurrentDirectory(), "Uploads",
"Images",
        img.Name + "_profile.jpg");
        using (var stream = new FileStream(imagePath, FileMode.Create))
        {
            img.img.CopyTo(stream);
            string query =
                $"@exec EmployeeProfilePic @action='UPDATE_PIC', @userName =
'{img.Name}', @path = '{Path.Combine("Uploads", "Images", img.Name +
"_profile.jpg")}'";
            string res = DbManager.queryExecute(query);
            if (res == "true")
                return Ok(new { message = "Employee Pic added Successfully" });
            else
            {
                return Ok(new { message = "Employee Pic Not Added Successfully - " + res });
            }
        }
    }
    catch (Exception e)
    {
        return BadRequest(new { message = e.Message });
    }
}
```

Sample - T-SQL Stored Procedure

```

CREATE PROCEDURE [dbo].[ManageMeeting]
    @operation VARCHAR(10), -- 'CREATE', 'READ', 'UPDATE', 'DELETE'
    @meetingId UNIQUEIDENTIFIER = NULL,
    @RequestId UNIQUEIDENTIFIER = NULL,
    @Scheduled DATETIME = NULL,
    @Reason VARCHAR(30) = NULL,
    @Assign uniqueidentifier = null
AS
BEGIN
    SET NOCOUNT ON;

    -- CREATE
    IF @operation = 'CREATE'
    BEGIN
        INSERT INTO Meeting (RequestId, Scheduled, Reason)
        VALUES (@RequestId, @Scheduled, @Reason);
    END

    -- READ
    ELSE IF @operation = 'READ'
    BEGIN
        IF @meetingId IS NOT NULL
            SELECT * FROM Meeting WHERE meetingId = @meetingId;
        ELSE IF @RequestId IS NOT NULL
            SELECT * FROM Meeting WHERE RequestId = @RequestId;
        ELSE
            SELECT * FROM Meeting;
    END

    -- UPDATE
    ELSE IF @operation = 'UPDATE'
    BEGIN
        if @Assign is not null
        begin
            UPDATE Meeting SET Handler = @Assign WHERE meetingId =
@meetingId
        end
        else
        begin
            UPDATE Meeting
            SET
                RequestId = ISNULL(@RequestId, RequestId),

```

```
Scheduled = ISNULL(@Scheduled, Scheduled),
Reason = ISNULL(@Reason, Reason)
WHERE meetingId = @meetingId;

END
END

-- DELETE
ELSE IF @operation = 'DELETE'
BEGIN
    DELETE FROM Meeting WHERE meetingId = @meetingId;
    SELECT 'Meeting deleted' AS Result;
END
END
```

6. TESTING

Testing is a critical phase in the software development lifecycle (SDLC) that ensures the application meets specified requirements, functions correctly, and delivers a seamless user experience. It involves systematically evaluating the software to identify defects, verify performance, and validate security and reliability before deployment.

6.1 Testing Objectives

The primary objectives of testing in Leave Flow are to ensure the system is reliable, secure, and user-friendly while meeting all functional and business requirements. Below are the key testing goals:

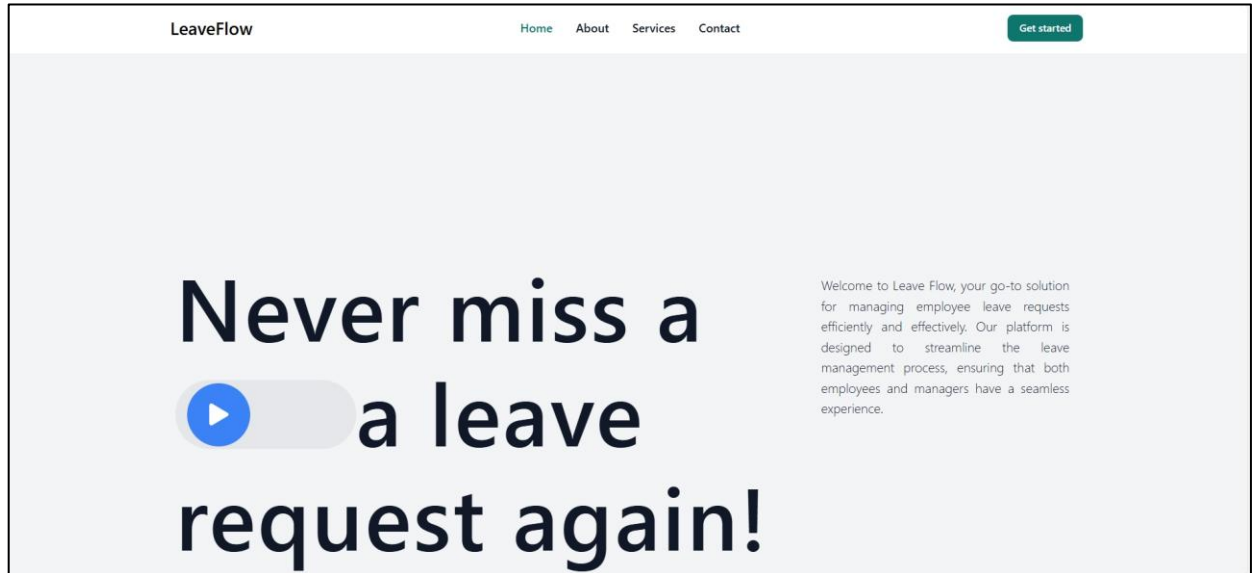
- Verify that leave requests, approvals, and rejections work as intended.
- Test mobile responsiveness Whether it will fits or not to the all device
- Offline Detection, if detected it should not allow the user to use the application

6.2 Test Cases

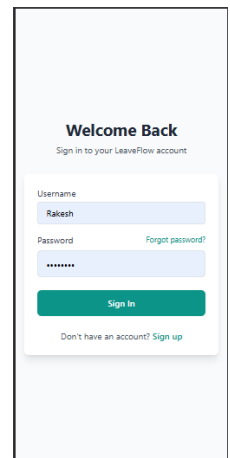
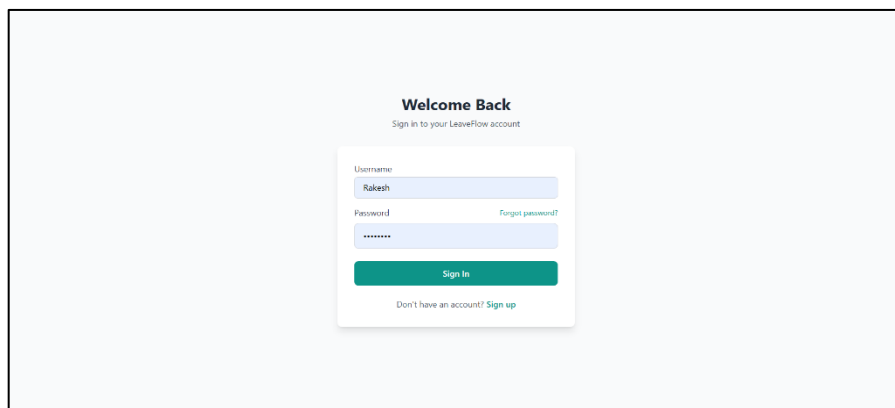
Testing Type	Purpose	Example in Leave Flow
Login Testing	Is rendering or not the role based home page	If Teacher or student or principal login to the site, there related home page need to render.
Unit Testing	Checking all individual components or working properly	API Controllers are sending or not the needed JSON data, Frontend components are responding for device width
Integration Testing	After integrating the frontend and backend	Login Component is sending and receiving the response from the API
System testing	End-to-end workflow validation	Student is able to send and receive the leave and results
Mock Release testing	Checking is there any errors might be raise while hosting	CORS Error, Mixed Content type of errors are arising

7. Product Snippets

Landing page (anonymous user)



Login Page



Signup page:

Create Account

I am a:

Choose your College

Select your college

ICCA

Harihara College

Username

Date of Birth

Adhar Number

Email

Password

Choose your Batch

Section

Already have an account? [Log in](#)


Principal Page:

Leave Management

Principal Dashboard

Hi, ICCA

- Nagaveni N.P
- veereshm813@gmail.com



Leave Requests

0

Organization Staff

3

Batch & Section Manager


3

Leave Management

Principal Dashboard

Hi, ICCA

- Nagaveni N.P
- veereshm813@gmail.com



Leave Requests

0

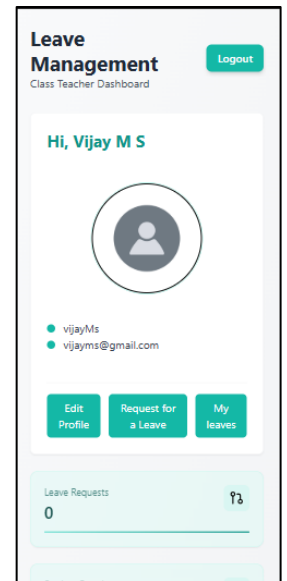
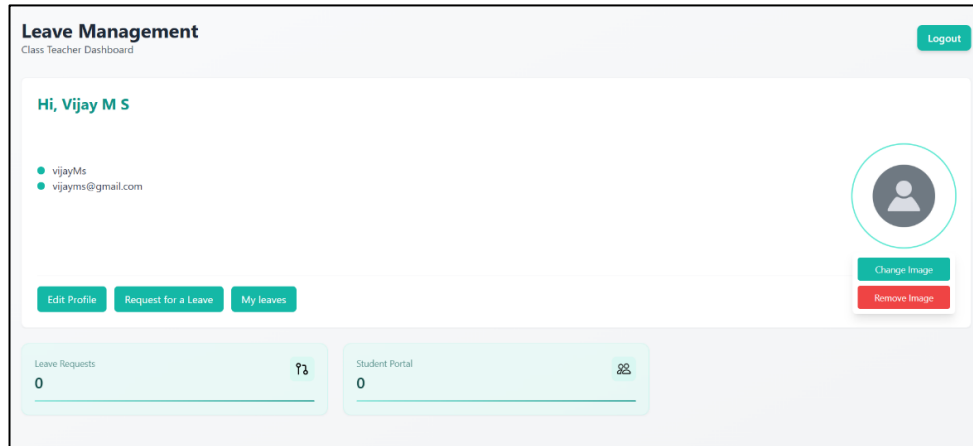
Organization Staff

3

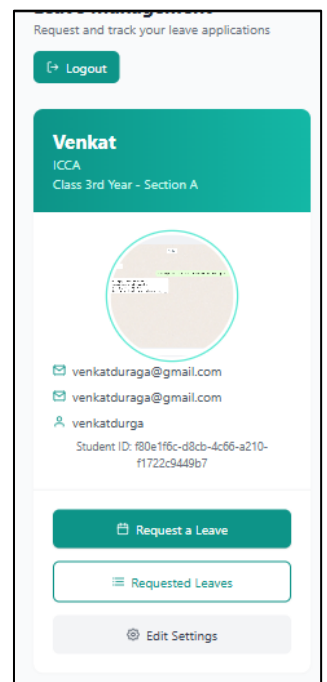
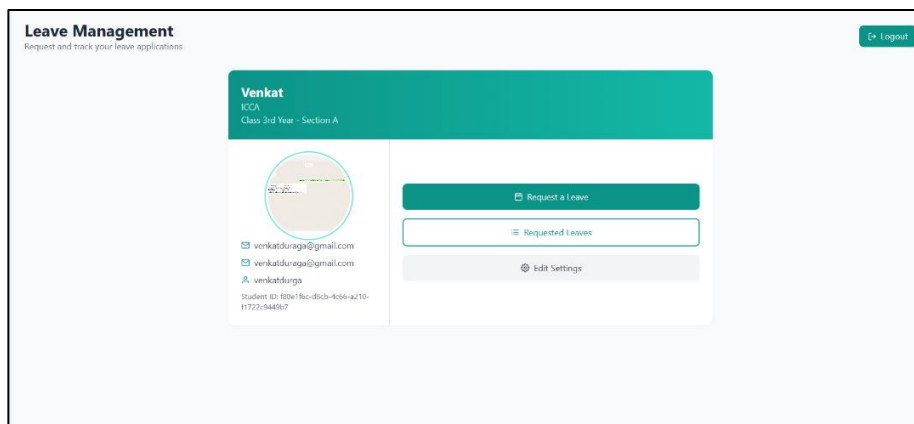
Interface College of Computer Applications

Page 26

Class Teacher Dashboard



Student Dashboard



Leave Requesting a teacher

Request Leave

Fill in your leave details

LEAVE TYPE

Casual Leave

START DATE

dd-mm-yyyy

END DATE

dd-mm-yyyy

REASON

CONTACT INFO

Phone or email

Submit Request

Request Leave

Fill in your leave details

LEAVE TYPE

Casual Leave

START DATE

dd-mm-yyyy

END DATE

dd-mm-yyyy

REASON

CONTACT INFO

Phone or email

Submit Request

Leave Approval by teacher

Leave Requests

Based on the reason Accept or Reject Leave

Pending Approvals

1

Approved Leaves

0

Leaves Requests

Venkat

3rd Year-A Section

12-05-2025 - 13-05-2025

Too much of stress

Requested on : 10-05-2025

Pending

Accept

Reject

Leave Requests

Based on the reason Accept or Reject Leave

Pending Approvals

1

Approved Leaves

0

Leaves Requests

Venkat

3rd Year-A Section

12-05-2025 - 13-05-2025

Too much of stress

Requested on : 10-05-2025

Pending

Accept

Reject

8. CONCLUSION

The Leave Flow project represents a significant leap forward in modernizing and streamlining absence management for organizations and educational institutions. By replacing outdated, manual processes with an intuitive, automated, and data-driven platform, Leave Flow eliminates inefficiencies, reduces errors, and enhances transparency across all levels of users.

Key Achievements:

- **Digital Transformation** – Successfully replaced paper-based/email leave systems with a centralized, cloud-based solution.
- **Workflow Automation** – Reduced approval times from days to minutes through smart notifications and conflict detection.
- **Policy Compliance** – Automated enforcement of company-specific leave rules, minimizing HR intervention.
- **User-Centric Design** – Delivered a seamless experience for employees, managers, and HR alike.
- **Scalability & Security** – Built on .NET Core for enterprise-grade performance and GDPR compliance.

Impact Delivered

- **80% faster** leave approvals compared to legacy systems.
- **40% reduction** in administrative workload for the organizations.

9. FUTURE ENHANCEMENT

This web application will be enhanced with the following features in a upcoming time:

- Push notifications for approvals/rejections.
- Voice Command “Hey Leave Flow, apply for vacation next week”.
- Deeper integration with SAP, Workday, and local payroll systems.
- Turn Leave Data into Strategic Insights.

10. BIBLIOGRAPHY

- [ASP.NET Core Web API documentation](#)
- [Framer motion Docs](#)
- [React.js Official Docs.](#)
- [Microsoft SQL Server Docs.](#)
- [Tailwind CSS Documentation.](#)