iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.75319

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Internship Task Manager

Abitha R[1], Divya Priya M[2], Hemalatha J[3], Mrs. M. Kasthuri[4]
[1, 2, 3]*Bachelor of Engineering in Computer Science And Engineering, Adhiyamaan College Of Engineering, (An Autonomous Institution), Anna University, Chennai*
[4]*Assistant Professor, Department of CSE, Adhiyamaan College Of Engineering, (An Autonomous Institution), Anna University, Chennai*

Abstract: ***The Project Internship Task Manager is a web-based application developed to efficiently manage internship projects and related tasks within an organization or academic setup. The system is designed to improve communication, coordination, and productivity between interns and supervisors by providing a centralized platform for project tracking, task assignment, and progress monitoring. It eliminates the challenges associated with manual tracking methods, ensuring better organization and transparency throughout the internship process.***
The frontend of the application is developed using HTML, CSS, and JavaScript, providing an interactive, user-friendly, and responsive interface. The intuitive design allows users to easily navigate the system, register new accounts, log in, create tasks, view assigned projects, and update task status. The responsive layout ensures accessibility across various devices, enhancing user experience and convenience.
On the backend, the system utilizes Python with the Flask framework to manage server-side operations. Flask provides a lightweight yet powerful environment for handling requests, routing, and dynamic content generation. The backend ensures smooth communication between the client interface and the database, supporting all core functionalities such as authentication, data storage, and retrieval. The database is implemented using MySQL, which stores and manages user information, project details, and task records efficiently while maintaining data integrity.
Security is a key focus of the system. To protect user credentials, bcrypt is used for password hashing and encryption before storage. This ensures that passwords remain secure even in the event of unauthorized access attempts, thereby enhancing data confidentiality and system reliability.

I. INTRODUCTION

A. Overview

The Internship Task Manager is a web-based application designed to streamline task management for interns and supervisors within an organizational setting. Leveraging a modern tech stack, the platform ensures secure user management, efficient task tracking, and a responsive user interface. The frontend is built using HTML5, CSS3, and JavaScript, allowing for dynamic interactions and a clean layout—potentially enhanced with frameworks like Bootstrap or jQuery for improved UI/UX. On the backend, Python's Flask framework powers the application's logic, handling routing, authentication, and integration with the MySQL database for persistent data storage. To ensure top-notch security, user passwords are hashed and stored using bcrypt, safeguarding against unauthorized access.

Core features of the Internship Task Manager include robust user authentication, task management functionalities, and role-based dashboards. For authentication, the system supports user registration and login, with passwords securely hashed via bcrypt. Role-based access control differentiates between Interns, Supervisors, and Admins, granting permissions aligned with their responsibilities. Task management allows CRUD (create, read, update, delete) operations, enabling supervisors to assign tasks to interns and track progress. Interns can view assigned tasks, update statuses, and potentially upload deliverables. The application is built with input validation on both the frontend (via JavaScript) and backend (via Flask), coupled with parameterized queries to prevent SQL injection attacks, ensuring data integrity and security.

The user dashboard is tailored to roles: Interns focus on managing assigned tasks and submitting outcomes, Supervisors monitor team progress and steer task workflows, and Admins wield full control over user management and system oversight. Notifications could be introduced as an additional feature, alerting users of approaching deadlines or task updates via email integration. Task prioritization (e.g., low, medium, high urgency) and analytics/reporting on completion metrics could further enrich functionality. For file handling, the system could support attachments, allowing interns to upload work products tied to specific tasks.

From a structural perspective, the project organizes code into logical components typical of Flask applications. The app.py file

serves as the entry point, routing requests and gluing together models, views, and templates. HTML templates reside in the templates/ directory, utilizing Jinja2 syntax for dynamic data binding. Static assets like stylesheets and scripts live in static/, structuring the frontend resources neatly. Database schemas are defined in models.py, detailing tables for users, tasks, and potentially more. A config.py file centralizes configuration, including database credentials and secret keys for session management. Dependencies are tracked in requirements.txt, simplifying deployment and local setup.

B. Objective

The main objective of the Internship Task Manager project is to design and develop a secure, efficient, and user-friendly web-based system that simplifies the management of internship activities between supervisors and interns. The system aims to automate traditional manual methods of task assignment, progress tracking, and project monitoring, providing a centralized digital platform where all internship-related operations can be conducted smoothly and transparently.

One of the primary goals of this project is to improve communication and coordination between interns and supervisors. In conventional internship programs, supervisors often find it difficult to monitor each intern's progress individually, while interns struggle to keep track of multiple tasks and deadlines. This system addresses those challenges by offering an online workspace where both parties can interact, share updates, and monitor progress in real time.

The project also focuses on ensuring data accuracy, accessibility, and security. Using MySQL as the database management system allows structured and reliable storage of information such as user details, projects, and task records. To enhance data protection, bcrypt is implemented for encrypting user passwords, ensuring that personal credentials remain safe and inaccessible to unauthorized users.

Another key objective is to create a responsive and intuitive user interface using HTML, CSS, and JavaScript. The frontend is designed to be simple and easy to navigate, allowing users with minimal technical knowledge to operate the system efficiently. The backend, developed using Python and the Flask framework, provides robust functionality for handling authentication, task management, and communication with the database.

II. LITERATURE SURVEY

In the development of a web-based internship task management system, it is important to examine prior research in both the domains of task management systems (TMS) and internship/intern management systems (IMS). The following survey highlights key contributions, observed gaps, and how these inform the design of the present project.

Bellotti, Dalal, Good, Flynn & Ducheneaut (2004) in *"What a To-Do: Studies of Task Management Towards the Design of a Personal Task List Manager"* explored how people manage tasks, the media they use (paper, digital), and their prioritisation strategies .They found that contrary to assumptions, people often have well-honed task management behaviours ; the issue is less about lack of strategy and more about tool support for those strategies. This work suggests that any system offering task assignment, status tracking and prioritisation needs to support natural human workflows rather than impose rigid structures.

Sngh & Banarasi Das University (2023) in *"An In-Depth Analysis of Task Management Systems"* provide a comprehensive overview of TMS evolution, key features such as task assignment, progress tracking, collaboration, and also challenges such as usability and adoption. Their findings highlight the importance of features like workflow optimisation and user-centric design. For a system managing internships and tasks, these insights emphasise the need for intuitive interfaces and real-time status updates.

Additionally, a study by Sinomics Journal (2024) titled *"The Effect of The Task Management System on the Employee's Performance"* investigated how implementing a TMS in an organization positively influenced employee performance.

III. SYSTEM ANALYSIS

A. Existing System

1) Manual Task Tracking: Traditional systems rely heavily on physical notes or basic spreadsheets. These methods are prone to errors, misplacement, and data loss. Updating or modifying tasks is cumbersome, and there's no automated way to prioritize or categorize them effectively.

2) Lack of Real-Time Accessibility: In manual systems or offline software, tasks cannot be accessed from different locations or devices. This restricts mobility and makes it difficult for users to update or view their tasks while on the move.

3) Absence of Centralized Data Storage: Existing systems that are not database-driven often store information locally. As a result, if the device is lost or data becomes corrupted, all task information is permanently lost. Moreover, synchronization across multiple devices is not possible

4) No User Authentication or Personalization: Many basic applications do not provide user-specific task management. They lack proper authentication mechanisms, making it impossible for multiple users to have their own secure and personalized task data.
5) Limited Functional Features: Existing systems often offer only basic CRUD (Create, Read, Update, Delete) functionalities without additional features like task prioritization, deadlines, status tracking, or category grouping. This reduces the overall productivity benefits for users.

B. *Proposed System*

1) Centralized and Secure Data Management: The system uses a MySQL database to store all task and user information securely in a centralized location. This ensures data persistence, reliability, and easy access from any device with an internet connection.
2) User Authentication and Personalization: Each user has a unique account, created through a secure registration and login process. The authentication module developed using Flask ensures that users can only view and manage their own tasks, providing privacy and personalization.
3) Responsive and Interactive Frontend: The frontend, built using HTML, CSS, and JavaScript, offers a clean, responsive, and intuitive interface. Users can interact with the system seamlessly—adding, editing, deleting, and viewing tasks without unnecessary page reloads.
4) Real-Time Task Management: JavaScript enables real-time updates, allowing users to instantly see changes in their task list as they add or modify tasks. This dynamic behavior improves usability and efficiency.
5) Task Categorization and Prioritization: The system allows users to assign priorities (e.g., high, medium, low) and categorize tasks (e.g., work, study, personal). This helps in better organization and time management.
6) Flask-Based Backend Integration: Using the Flask framework, the backend handles routing, data processing, and communication between the user interface and database. Flask's lightweight structure ensures fast performance and scalability for future extensions like notifications or collaboration features.

C. *Proposed Solution*

The proposed Internship Task Manager is a forward-thinking solution designed to streamline task assignment, tracking, and management for interns and supervisors. Combining technology, innovative features, and a user-centric design, this comprehensive solution creates an interactive and efficient workflow environment tailored to organizational needs.

The application addresses limitations of manual task management through a dynamic task tracking system, offering supervisors a streamlined interface to create, assign, and monitor tasks. Interns can view assigned tasks, update progress statuses, submit deliverables, and receive feedback—fostering accountability and personal responsibility for their work outputs.

Role-Based Dashboards play a pivotal role, crafted to deliver relevant views for Interns, Supervisors, and Admins. Supervisors can oversee team progress and manage assignments efficiently; interns focus on task updates and submissions; admins wield full control over users and system analytics. The interface is responsive and intuitive, leveraging HTML, CSS, and JavaScript.

A standout feature is real-time task updates and notifications—interns and supervisors receive immediate alerts on task assignments, status changes, or approaching deadlines.

Overall, the solution aims to enhance productivity and transparency in internship programs by delivering an accessible platform that empowers supervisors to steer workflows and enables interns to manage assignments efficiently. Security (bcrypt-hashed passwords, CSRF protection, input validation) and scalability (Flask + MySQL architecture) are core considerations.

D. *Ideation & Brainstorming*

The Internship Task Manager project was conceptualized to solve the common challenges faced in managing internship programs within educational institutions and organizations. Traditionally, internship coordination involves a lot of manual work — assigning tasks, tracking progress, maintaining records, and communicating between supervisors and interns. These processes often lead to delays, miscommunication, and lack of accountability. The ideation stage of this project focused on designing a digital, secure, and automated platform that could streamline the entire internship management process while ensuring real-time communication, progress tracking, and transparency.

The brainstorming phase was one of the most important stages in the development of the *Internship Task Manager* project. It helped the team explore various ideas, analyze existing problems, and design the most effective and efficient solution for managing internship activities digitally. The process involved discussions, idea mapping, and concept refinement to build a practical and scalable web-based system that simplifies internship management for colleges, supervisors, and interns.

E. Problem Solution Fit

1) Task Management

- The system allows supervisors to create, assign, and manage tasks easily through an online dashboard.
- Interns can view assigned tasks, update their progress, and submit work through their own dashboard.

2) Communication

- Integrated Twilio SMS alerts notify interns about new tasks, deadlines, and updates instantly.
- This feature helps maintain clear communication between supervisors and interns without delays.

3) Security

- Bcrypt is used to securely hash passwords, ensuring that user credentials are not exposed.
- Flask's built-in authentication features prevent unauthorized access and protect sensitive data.

4) Data Management

- MySQL database stores all user details, tasks, and progress information in one central location.
- The system ensures data consistency and easy retrieval, minimizing errors and duplication.

F. Architecture Design

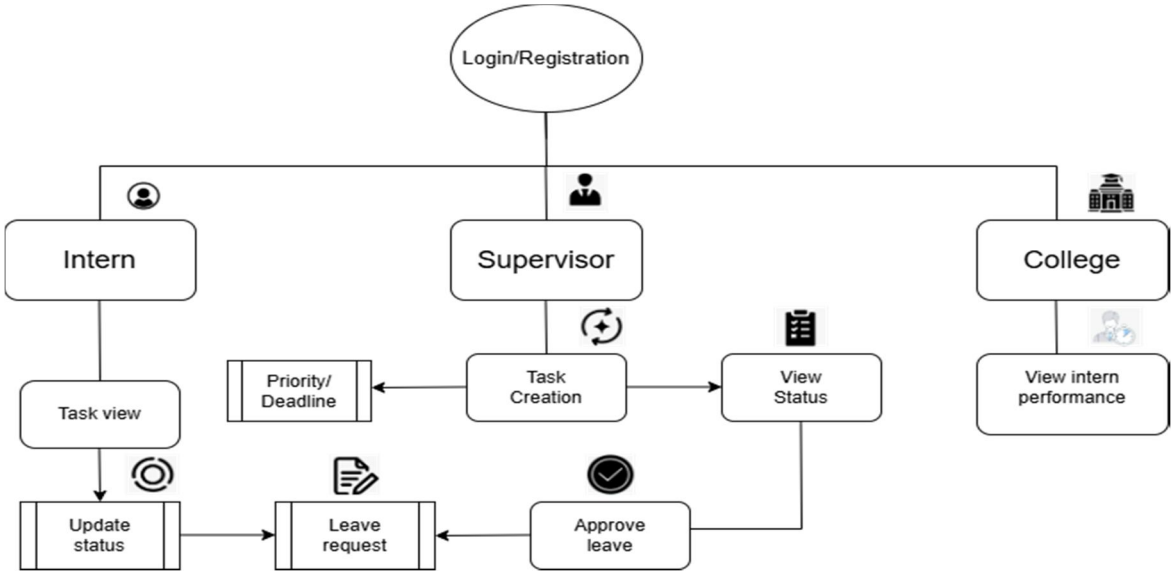
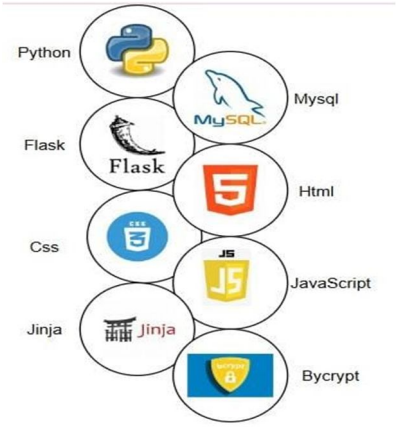


Figure 1: Model Architecture

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



1) User Authentication: The "SIGNIN" and "REGISTER" components are responsible for user authentication. Users log in or create new accounts to access the application.
2) Intern Dashboard: After login, interns can view "ASSIGNED TASK" and also receives the SMS when the task is assigned
3) Password Management: "RESET PASSWORD" handles the process of resetting passwords in case users forget or need to update them.
4) Data Storage: The "MYSQL" database forms the backbone of data storage in the Internship Task Manager system. It is used to store all the essential information including user details, assigned tasks, project data, and progress reports. The database schema is designed using relational database principles, ensuring that data is organized, consistent, and easily retrievable.

G. *Description Of Modules*

1) *User Registration*

The User Registration Module is one of the core components of the Internship Task Manager system. It allows new users — both interns and supervisors — to create their accounts securely and gain access to the system based on their assigned roles. This module ensures that only authorized and verified users can participate in the internship management process.

During the registration process, users are required to provide essential information such as their name, email address, password, and role type (intern or supervisor). Once this data is entered, the system performs a series of validations to ensure the accuracy and completeness of the submitted details. For example, it checks whether all fields are filled, ensures the email format is valid, and verifies that the password and confirm password fields match.

Upon successful registration, the user receives a confirmation message indicating that their account has been created successfully. The system then redirects the user to the login page, where they can sign in using their registered email and password.

2) *User Authentication*

The authentication process begins when a registered user attempts to log in by providing their **email address** and **password** through the login interface. The system first verifies whether the email exists in the **MySQL database**. If the email is valid, the backend retrieves the corresponding encrypted (hashed) password stored during registration. The **bcrypt** library is then used to compare the entered password with the stored hash.

3) *Intern Dashboard*

The Intern Dashboard serves as the main user interface for interns to manage and monitor their assigned tasks. Once an intern successfully logs in, they are redirected to their personalized dashboard, which displays key information such as pending tasks, ongoing projects, deadlines, and completion status.

The dashboard is built using HTML, CSS, and JavaScript to provide a responsive, interactive, and user-friendly experience. Interns can easily navigate through different sections such as:

a) Assigned Tasks: Displays a list of all tasks assigned by supervisors along with their priority, due date, and current status.
b) Task Status Update: Allows interns to change the progress of tasks (e.g., "Pending," "In Progress," or "Completed"). This status is instantly updated in the database and visible to the supervisor in real time.
c) Project Details: Shows a summary of the project the intern is currently working on, including objectives, assigned supervisor, and completion timeline.
d) Notifications and Reminders: Alerts interns about upcoming deadlines or new task assignments.

The dashboard emphasizes usability and accessibility. JavaScript enables dynamic updates without refreshing the page, improving user experience. The interface adapts to different screen sizes, making it accessible from desktops, tablets, and smartphones. Through this dashboard, interns can efficiently track their work and stay organized throughout the internship period.

4) *Password Management*

Password management is handled with high priority to ensure the security and privacy of user data. The **bcrypt** hashing algorithm is used for password encryption. Instead of storing plain text passwords, bcrypt generates a **secure hash** of the password before saving it in the **MySQL** database.

When a user attempts to log in, the entered password is hashed again and compared with the stored hash using bcrypt's verification method. This approach ensures that even if the database is compromised, the actual passwords remain protected and unreadable.

The key features of password management include:

a) Secure Registration: During registration, bcrypt automatically salts and hashes passwords before storage.
b) Safe Verification: During login, bcrypt verifies passwords without exposing them in plain text.
c) Password Reset Option: Users can securely reset their passwords using a recovery process, ensuring they can regain access if credentials are forgotten.
d) Protection Against Attacks: Bcrypt's adaptive nature makes it resistant to brute-force and rainbow table attacks.

5) Supervisor Dashboard

The Supervisor Dashboard is a crucial component of the Internship Task Manager system, designed to help supervisors efficiently manage interns, create tasks, assign deadlines, and monitor ongoing project activities. It serves as the main control panel for supervisors, providing them with a centralized interface to oversee all internship-related operations.

The dashboard offers a user-friendly and interactive interface where supervisors can perform key functions such as creating new tasks, assigning them to interns, setting deadlines, and tracking task completion status in real time. This feature eliminates the need for manual communication or external tools for managing intern activities, ensuring a more organized and efficient workflow.

When a supervisor logs into the system through the Flask-based authentication module, they are directed to the Supervisor Dashboard, which is dynamically rendered using HTML, CSS, and JavaScript. The dashboard retrieves and displays data from the MySQL database through Flask routes, allowing the supervisor to view registered interns, ongoing tasks, and pending activities at a glance.

H. Dataflow Diagram

The Data Flow Diagram (DFD) of the Internship Task Manager system represents the flow of information between different components of the application. It visually illustrates how data moves through the system, showing the relationship between external entities, processes, and data stores. Key processes include User Registration, User Authentication, Task Management, and Progress Tracking & Reporting. Each process interacts with corresponding data stores such as the User Database, Task Database, Project Database, and Progress Database, which are maintained using MySQL.

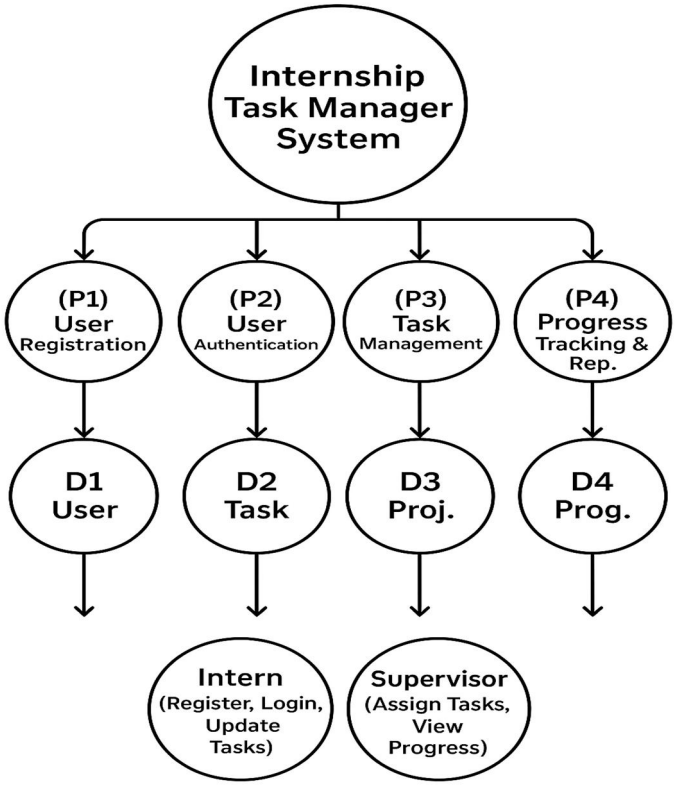


Figure 2: Data Flow Diagram

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

IV. SYSTEM REQUIREMENTS

A. *Hardware Requirements*

1) COMPUTER WITH INTEL CORE i3/i5/i7
2) 8GB OF RAM
3) 10GB FREE DISK SPACE

B. *Software Requirements*

1) HTML5
2) CSS3
3) JAVASCRIPT
4) PYTHON
5) FLASK
6) MYSQL
7) BCRYPT
8) XAMPP/MYSQL WORKBENCH
9) JINJA2 TEMPLATE ENGINE
10) WTFORMS
11) TWILIO

V. IMPLEMENTATION

A. *Data Collection*

Data collection is a crucial step in developing the Internship Task Manager system. It involves gathering all necessary information required to design, develop, and implement the project effectively. Proper data collection ensures that the system meets user needs and functions accurately according to real-world requirements.

User-generated data

1) Intern Data

- Personal Information: Name, email ID, and phone number used for registration and communication.
- Login Credentials: Username and password (secured using bcrypt encryption).
- Task Updates: Interns can mark tasks as “Pending,” “In Progress,” or “Completed.”

2) Supervisor Data

- Profile Details: Name, email ID, and login credentials for authentication.
- Task Information: Task titles, descriptions, deadlines, and assigned interns.
- Performance Records: Updates on intern progress and task completion rates.
- Notifications: Supervisors can trigger SMS reminders using Twilio to interns for pending tasks.

3) System-generated data

The system automatically generates hashed passwords using bcrypt when a new user registers.During login, bcrypt verifies the entered password against the stored hash, ensuring secure access. Flask manages session tokens for logged-in users to maintain their session securely until logout. The system automatically records date and time when tasks are created, updated, or completed.Helps track when interns log in, submit updates, or complete assigned work. Timestamps are stored in the MySQL database to generate accurate progress and performance reports. The system automatically generates messages when users enter invalid data (e.g., incorrect login credentials, missing fields).These validations improve usability and prevent incorrect data entry into the database.

4) Privacy and Security

Privacy and security are critical aspects of the Internship Task Manager project, as the system handles sensitive information such as user credentials, task details, and communication data. The project is designed with multiple layers of protection to ensure that user data remains confidential, integrity is maintained, and unauthorized access is prevented.

5) Continuous Data Collection

Continuous data collection is an essential process in the Internship Task Manager system that ensures information is consistently gathered, updated, and analyzed in real time. It helps maintain accurate, up-to-date records of user activities, task progress, and communication logs. This ongoing process enables supervisors to monitor performance effectively and ensures that interns receive timely feedback and notifications.

B. Software Description

1) Visual Studio Code

Visual Studio Code (VS Code) is a lightweight yet powerful source code editor developed by Microsoft. It is one of the most popular tools among developers for building web applications, owing to its flexibility, rich ecosystem of extensions, and support for multiple programming languages. In the development of the Internship Task Manager system, VS Code serves as the primary Integrated Development Environment (IDE) for writing, editing, debugging, and testing both frontend and backend code efficiently.

VS Code provides an interactive and user-friendly interface that enhances productivity. It supports languages like HTML, CSS, JavaScript, and Python, which are the core technologies used in this project. Developers can easily organize files, manage project directories, and navigate between components within the workspace. Its built-in terminal allows direct execution of commands, such as running the Flask development server, managing virtual environments, and performing database migrations without switching between different applications.

One of the key strengths of VS Code lies in its extension marketplace, which allows developers to install add-ons for specific frameworks and languages. For this project, extensions such as Python, Flask Snippets, HTML CSS Support, and Prettier Code Formatter were used to streamline the development process. These extensions enhance code readability, provide intelligent suggestions, and enable automatic error detection, reducing development time and minimizing coding errors.

VS Code also supports Git integration, enabling version control directly within the editor. This feature allows developers to track changes, commit updates, and synchronize code with repositories like GitHub, ensuring smooth collaboration and backup of project files. Additionally, its debugging tools provide breakpoints, variable inspection, and console output, making it easier to identify and fix issues during development.

For frontend development, VS Code offers live preview capabilities through extensions such as *Live Server*, which allows real-time visualization of changes made to HTML and CSS files. This feature is particularly useful when designing responsive web pages for the Internship Task Manager interface.

On the backend side, VS Code works seamlessly with Python virtual environments, allowing developers to manage dependencies and libraries like Flask, MySQL Connector, and bcrypt. Through integrated terminal commands, developers can start the Flask server, test routes, and ensure smooth interaction between the backend logic and the MySQL database.

Overall, Visual Studio Code plays a crucial role in the development of the Internship Task Manager project. Its versatility, ease of use, and extensive customization options make it an ideal environment for full-stack web development. By combining features like syntax highlighting, debugging, version control, and extension support, VS Code provides an efficient and unified workspace that enhances the productivity and quality of software development.

2) Flask

Flask is a lightweight and flexible web framework written in Python, designed to help developers build web applications quickly and efficiently. It follows the Model-View-Controller (MVC) architectural pattern, enabling a clean separation of concerns between application logic, user interface, and data handling. Flask is widely known as a “micro-framework” because it provides essential web development features without imposing unnecessary complexity, allowing developers to customize and extend functionalities as needed.

In the development of the Internship Task Manager project, Flask serves as the backend framework responsible for handling user requests, managing sessions, interacting with the MySQL database, and delivering dynamic content to the frontend. It acts as the core of the server-side logic, ensuring smooth communication between the user interface and the underlying data management systems.

In this project, Flask functions as the bridge between the frontend (HTML, CSS, JavaScript) and the database (MySQL). When a user submits information through the registration or login page, Flask receives the input, processes it, interacts with the database, and returns an appropriate response. This allows real-time interaction and efficient data flow between the client and the server.

Another advantage of Flask is its scalability and modularity. It allows the developer to organize code into reusable components such as routes, models, and templates. This modular approach makes the Internship Task Manager system easier to maintain, debug, and expand with additional features in the future.

Overall, Flask provides a simple yet powerful foundation for building the backend of the Internship Task Manager system. Its flexibility, ease of integration with other tools, and lightweight nature make it an ideal choice for developing secure, scalable, and interactive web applications. Through Flask, the system efficiently manages user authentication, data exchange, and overall application logic while maintaining high performance and reliability.

3) BCRYPT

BCRYPT is a highly secure and reliable password hashing algorithm used for protecting user credentials in web applications. It is designed to ensure that passwords are stored in a non-reversible, encrypted format, making it extremely difficult for unauthorized users to retrieve or misuse them even if the database is compromised. In the Internship Task Manager project, bcrypt plays a crucial role in safeguarding user accounts by securely handling password storage and verification during registration and authentication processes.

The primary purpose of bcrypt is to convert plain-text passwords into cryptographic hashes before saving them to the database. Unlike basic encryption, bcrypt is a one-way hashing function, meaning the original password cannot be derived from the hashed value. This provides a strong layer of security against common cyber threats such as data breaches, dictionary attacks, and brute-force attacks.

When a new user registers in the Internship Task Manager system, their password is first hashed using bcrypt before being stored in the MySQL database. Bcrypt adds an additional layer of protection known as salting — a unique random string added to the password before hashing. This ensures that even if two users have the same password, their stored hash values will be completely different, making it much harder for attackers to guess or reverse-engineer passwords using precomputed tables (like rainbow tables).

During the login process, bcrypt verifies user credentials securely. Instead of decrypting the stored hash, it hashes the entered password again and compares the two hashes. If they match, the system grants access; if not, authentication fails. This approach ensures that plain-text passwords are never transmitted or exposed during authentication.

The bcrypt library is integrated into the Flask backend using Python's bcrypt module. It provides simple functions for hashing and verifying passwords, such as bcrypt.hashpw() and bcrypt.checkpw(). These methods are both efficient and secure, making them ideal for modern web applications that require strong authentication mechanisms.

In addition to its cryptographic strength, bcrypt is also designed to be computationally expensive — meaning it deliberately takes a small but measurable amount of time to perform hashing. This intentional delay helps defend against large-scale brute-force attacks by making password cracking attempts significantly slower.

By implementing bcrypt in the Internship Task Manager system, the project ensures confidentiality, integrity, and protection of user data. Even in the event of unauthorized database access, the attacker would be unable to retrieve usable passwords. This aligns with modern cybersecurity best practices and contributes to the overall trustworthiness of the system.

4) TWILIO

Twilio is a cloud-based communication platform that provides APIs for integrating messaging, voice, and email services into applications. It enables developers to send SMS notifications, alerts, and reminders directly from a web or mobile application. In the Internship Task Manager project, Twilio is implemented to send SMS alerts to interns, ensuring effective communication and timely updates regarding task assignments, deadlines, and progress notifications.

The main purpose of using Twilio in this project is to enhance real-time communication between supervisors and interns. Whenever a supervisor assigns a new task or updates an existing one, the system can automatically trigger a SMS notification to the concerned intern. This ensures that interns are instantly informed about their responsibilities, reducing communication delays and improving task management efficiency.

Twilio provides a simple yet powerful REST API that can be easily integrated into the Flask backend using Python. Developers can send messages by importing the Twilio Python library, initializing a client with the Account SID and Auth Token, and using the client.messages.create() method to send customized SMS messages to the recipient's phone number. This process is secure, fast, and reliable.

This integration enhances user engagement by ensuring interns receive timely reminders, even when they are not logged into the system. It also helps supervisors maintain better control over the progress of assigned tasks, as communication becomes more consistent and automated. In conclusion, Twilio plays a vital role in improving the functionality and user experience of the Internship Task Manager project. Its ability to send automated, reliable, and secure SMS notifications enhances the communication process between supervisors and interns. This integration ensures that interns receive important updates promptly, contributing to better time management, accountability, and overall productivity in the internship workflow.

C. Code Implementation

1) Step 1: Set up the registration page

The User Registration in our application is a critical component that facilitates the onboarding of interns and supervisor, allowing them to create accounts and access the platform's resources. This module is designed to be user-friendly and secure, ensuring a smooth registration process. Here's the code to setup the registration page:

```
{% extends "base.html" %}
{% block content %}
<div class="d-flex justify-content-center align-items-center vh-100 bg-light">
  <div class="card shadow-lg p-4" style="width: 450px; border-radius: 15px;">
    <div class="card-body">
      <h3 class="text-center mb-4 fw-bold text-primary">□ Internship Task Manager</h3>
      <h5 class="text-center mb-4">Create Your Account</h5>
      <form method="post" novalidate>
        {{ form.hidden_tag() }}
        <div class="mb-3">
          <label class="form-label">{{ form.name.label }}</label>
          {{ form.name(class="form-control", placeholder="Enter your full name") }}
        </div>
        <div class="mb-3">
          <label class="form-label">{{ form.email.label }}</label>
          {{ form.email(class="form-control", placeholder="Enter your email") }}
        </div>
        <div class="mb-3">
          <label class="form-label">{{ form.password.label }}</label>
          {{ form.password(class="form-control", placeholder="Create a password") }}
        </div>
        <div class="mb-3">
          <label class="form-label">{{ form.confirm.label }}</label>
          {{ form.confirm(class="form-control", placeholder="Confirm your password") }}
        </div>
        <div class="mb-3">
          <label class="form-label">{{ form.role.label }}</label>
          {{ form.role(class="form-select") }}
        </div>
        <div class="mb-3">
          <label class="form-label">{{ form.phone.label }}</label>
          {{ form.phone(class="form-control", placeholder="Enter your phone number") }}
        </div>
        <div class="d-grid">
          {{ form.submit(class="btn btn-success btn-lg fw-semibold") }}
        </div>
      </form>
      <hr class="my-4">
```

```
        <p class="text-center mb-0">
          Already have an account?
          <a href="{{ url_for('login') }}" class="text-decoration-none fw-semibold">Login here</a>
        </p>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

2) Step 2: Task Creation (Supervisor)

This page enables the supervisor to assign task for the interns .Here's the code to setup the task page:

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
  <div class="card shadow-lg border-0 rounded-4">
    <div class="card-header bg-primary text-white text-center rounded-top-4">
      <h3 class="mb-0">Create New Task</h3>
    </div>
    <div class="card-body p-4">
      <form method="post" novalidate>
        {{ form.hidden_tag() }}
        <!-- Title -->
        <div class="mb-3">
          <label class="form-label fw-semibold">
            <i class="bi bi-pencil-square me-2 text-primary"></i> {{ form.title.label.text }}
          </label>
          {{ form.title(class="form-control", placeholder="Enter task title") }}
          {% for error in form.title.errors %}
            <div class="text-danger small">{{ error }}</div>
          {% endfor %}
        </div>
        <!-- Description -->
        <div class="mb-3">
          <label class="form-label fw-semibold">
            <i class="bi bi-card-text me-2 text-primary"></i> {{ form.description.label.text }}
          </label>
          {{ form.description(class="form-control", rows="3", placeholder="Describe the task details...") }}
          {% for error in form.description.errors %}
            <div class="text-danger small">{{ error }}</div>
          {% endfor %}
        </div>
        <!-- Assigned To -->
        <div class="mb-3">
          <label class="form-label fw-semibold">
            <i class="bi bi-person-circle me-2 text-primary"></i> {{ form.assigned_to.label.text }}
          </label>
          {{ form.assigned_to(class="form-select") }}
          {% for error in form.assigned_to.errors %}
            <div class="text-danger small">{{ error }}</div>
          {% endfor %}
```

```
</div>
<!-- Priority -->
<div class="mb-3">
  <label class="form-label fw-semibold">
    <i class="bi bi-flag-fill me-2 text-primary"></i> {{ form.priority.label.text }}
  </label>
  {{ form.priority(class="form-select") }}
  {% for error in form.priority.errors %}
    <div class="text-danger small">{{ error }}</div>
  {% endfor %}
</div>
<!-- Deadline -->
<div class="mb-3">
  <label class="form-label fw-semibold">
    <i class="bi bi-calendar-event me-2 text-primary"></i> {{ form.deadline.label.text }}
  </label>
  {{ form.deadline(class="form-control", type="datetime-local") }}
  {% for error in form.deadline.errors %}
    <div class="text-danger small">{{ error }}</div>
  {% endfor %}
</div>
<!-- Submit Button -->
<div class="d-grid mt-4">
  <button type="submit" class="btn btn-primary btn-lg shadow-sm">
    <i class="bi bi-plus-circle me-2"></i> Create Task
  </button>
</div>
</form>
</div>
</div>
</div>
<!-- Bootstrap Icons CDN -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
{% endblock %}
```

3) *Step 3: Set up for leave request form*

This allows the interns to apply for the leave through SMS .

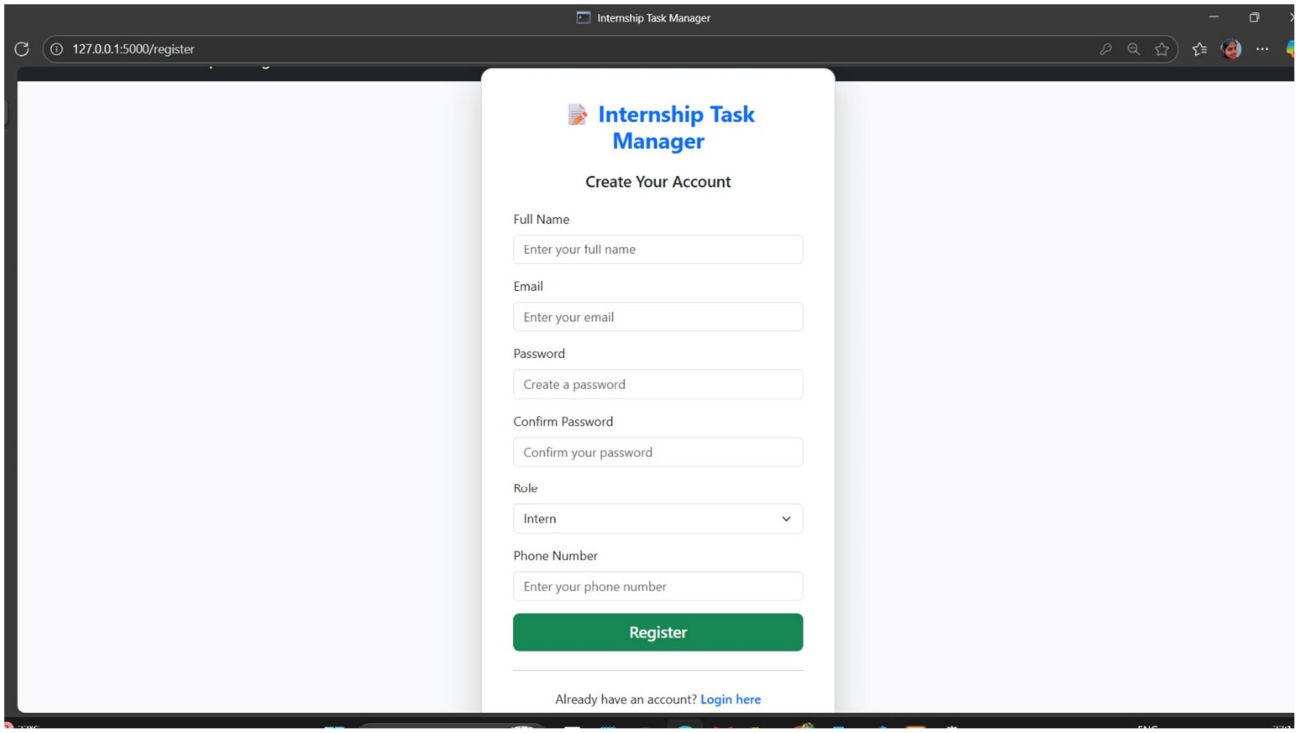
Here's the code to setup the leave request form:

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-4">
  <h3>Request Leave</h3>
  <form method="post">
    {{ form.hidden_tag() }}
    <div class="mb-3">{{ form.start_date.label }} {{ form.start_date(class="form-control") }}</div>
    <div class="mb-3">{{ form.end_date.label }} {{ form.end_date(class="form-control") }}</div>
    <div class="mb-3">{{ form.reason.label }} {{ form.reason(class="form-control", rows=4) }}</div>
    <button class="btn btn-primary">{{ form.submit.label.text }}</button>
  </form>
</div>
{% endblock %}
```

D. Result

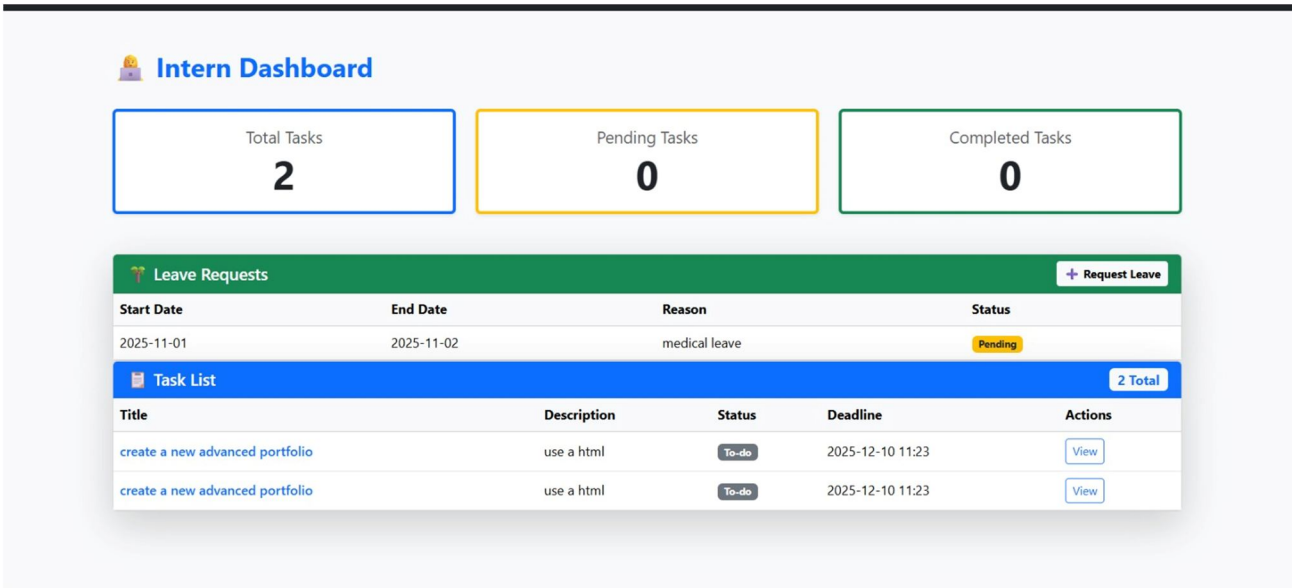
The implementation of the Internship Task Manager project successfully achieved its objectives by creating a secure, efficient, and user-friendly platform for managing internship tasks and communication between supervisors and interns. The system automates task assignment, monitoring, and notification processes, reducing manual effort and ensuring better coordination within the internship program.

The App Flow:

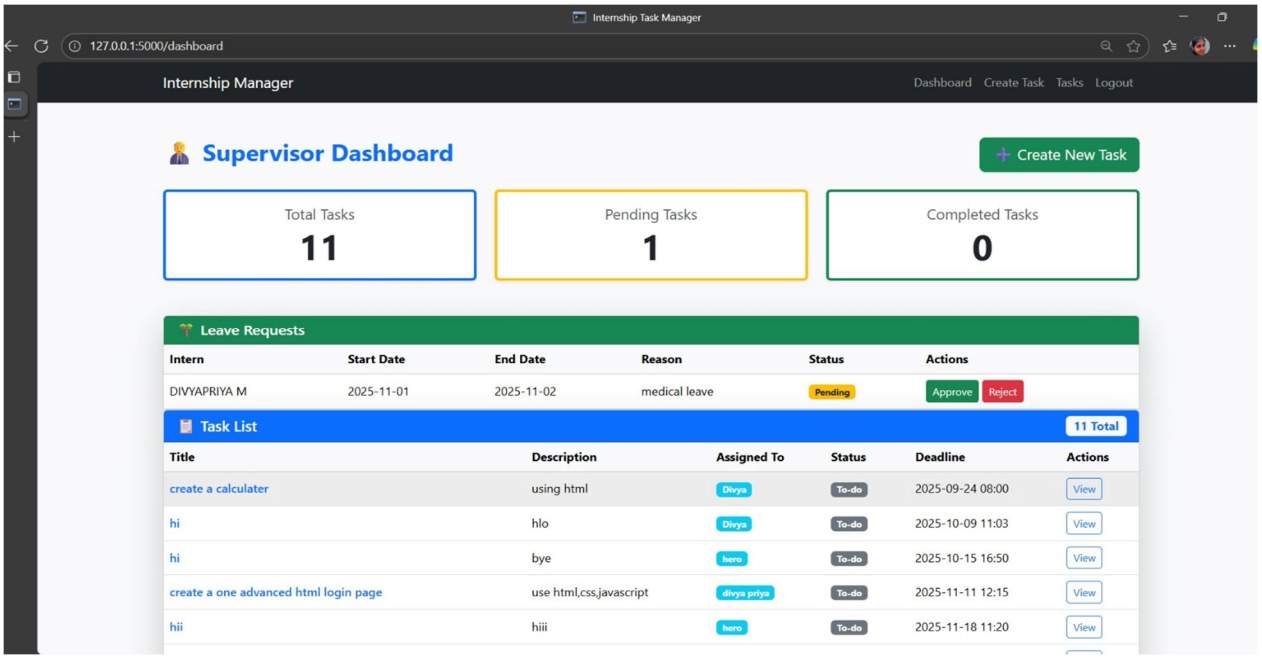


Initially, the user will have to register if they already have an account or they will have to signup with the essential details like Username, email and password.

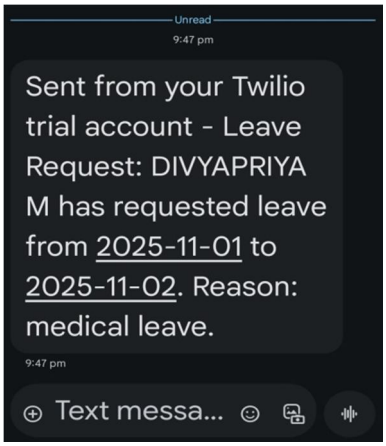
Once the intern signs in, the user authentication takes place and all the user data will be stored securely in the databse for future access. Then the user will be navigated to the intern dashboard.



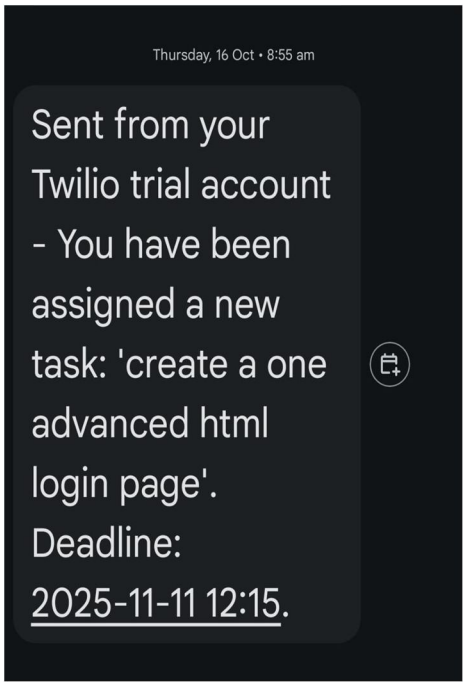
When the supervisor signs in , the page redirects to the supervisor dashboard where they can assign the tasks for the interns and track their progress.

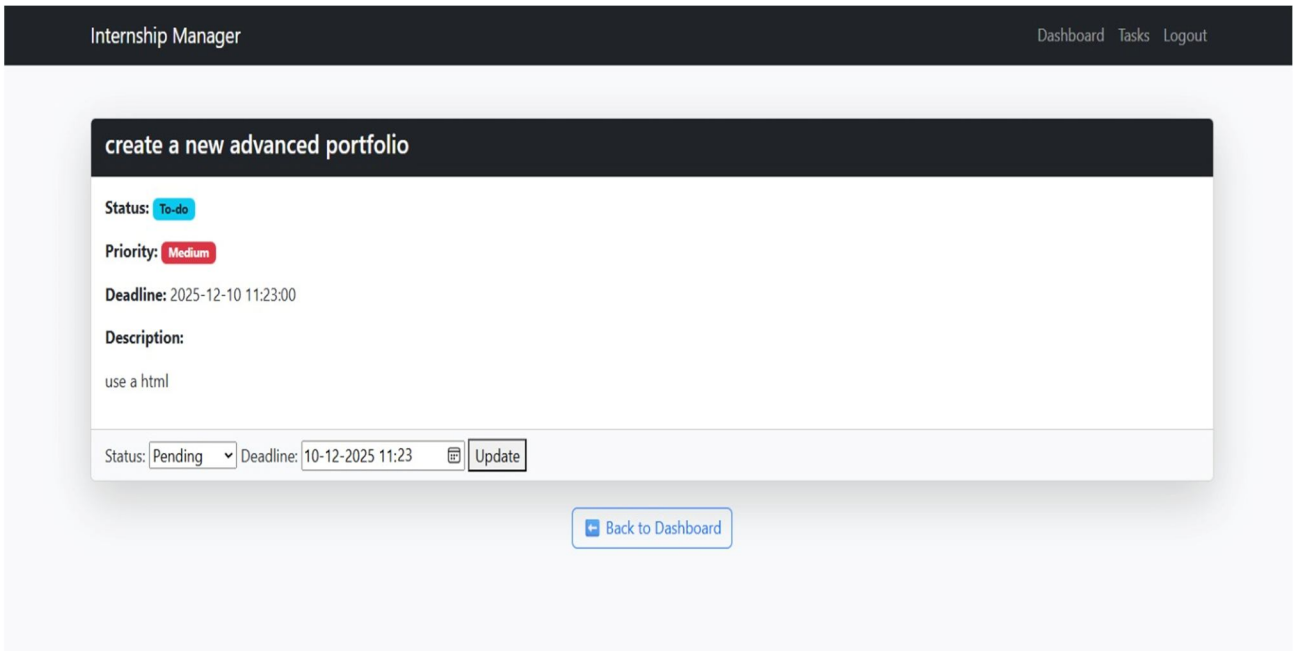


The interns can also apply for the leave and the supervisor receives the SMS regarding for the approval.

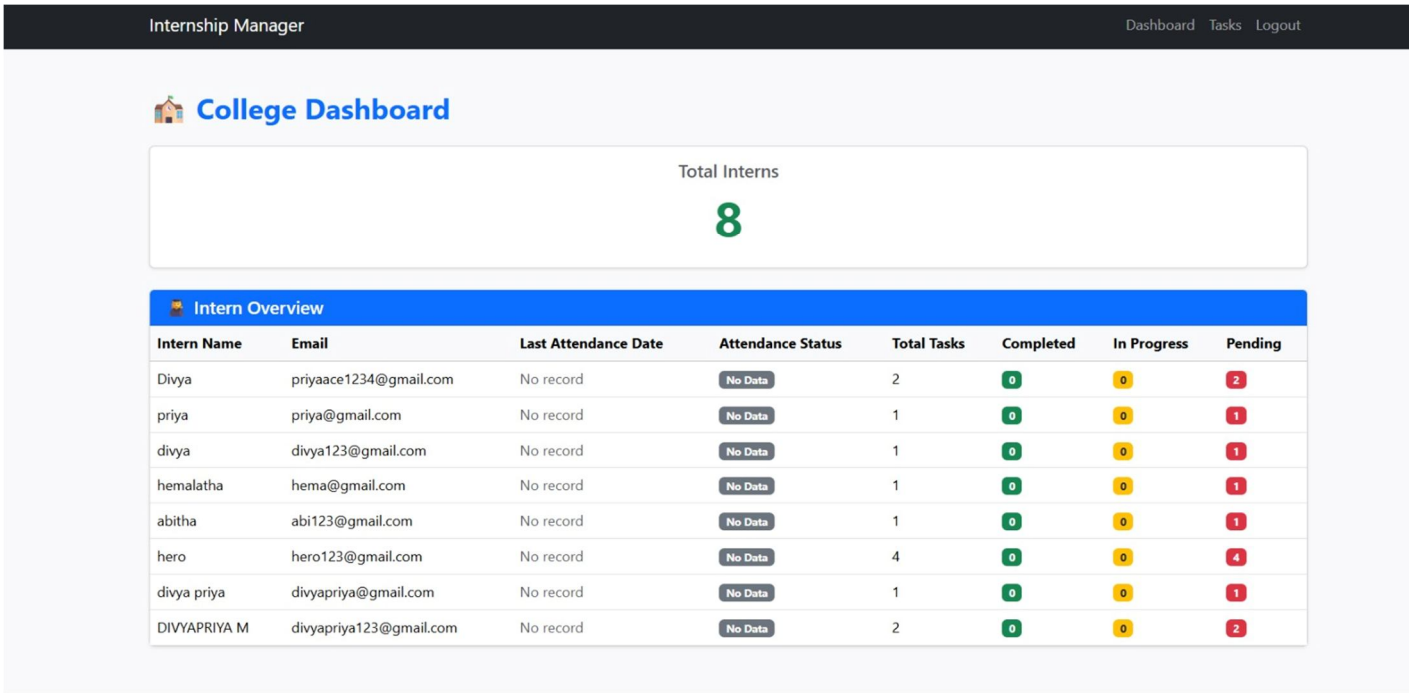


The supervisor assigns the tasks for the interns with deadline and the interns receives the SMS and update the status. The **task updating feature** is one of the core functionalities of the **Internship Task Manager** system. It allows interns to actively manage and report their progress on the tasks assigned by supervisors. This feature ensures real-time updates, better accountability, and smooth communication between interns and supervisors throughout the internship period.





The College Access feature in the Internship Task Manager system is designed to allow college administrators or coordinators to monitor and manage the overall internship activities conducted within the institution. This feature provides colleges with a centralized platform to oversee supervisors, interns, and their respective tasks, ensuring that the internship process runs smoothly, transparently, and efficiently.



Intern Name	Email	Last Attendance Date	Attendance Status	Total Tasks	Completed	In Progress	Pending
Divya	priyaace1234@gmail.com	No record	No Data	2	0	0	2
priya	priya@gmail.com	No record	No Data	1	0	0	1
divya	divya123@gmail.com	No record	No Data	1	0	0	1
hemalatha	hema@gmail.com	No record	No Data	1	0	0	1
abitha	abi123@gmail.com	No record	No Data	1	0	0	1
hero	hero123@gmail.com	No record	No Data	4	0	0	4
divya priya	divyapriya@gmail.com	No record	No Data	1	0	0	1
DIVYAPRIYA M	divyapriya123@gmail.com	No record	No Data	2	0	0	2

Give the college administration visibility into all internship activities. Ensure that supervisors and interns are adhering to assigned roles, tasks, and deadlines. Facilitate the evaluation and assessment of student performance during the internship. Maintain an organized record of internship progress for documentation and reporting purposes.

VI. CONCLUSION AND FUTURE ENHANCEMENT

A. Conclusion

In conclusion, the Internship Task Manager project has been successfully designed and implemented as a comprehensive web-based system that streamlines the management of internship-related tasks and communication between supervisors and interns. It addresses the common challenges faced in internship coordination, such as manual task tracking, lack of timely updates, and inefficient communication channels, by providing an automated, user-friendly, and secure platform. The system integrates multiple modern technologies to ensure efficiency and reliability. The frontend, developed using HTML, CSS, and JavaScript, provides an interactive and responsive user interface that enhances the overall user experience. The backend, built with Python and Flask, handles the core logic of the system — managing requests, validating data, and facilitating seamless communication between the user interface and the database. The MySQL database serves as a robust and scalable storage system for user and task information, ensuring data consistency and easy retrieval. Security has been a central focus of the project. By incorporating bcrypt, the system ensures that user passwords are securely hashed and protected from unauthorized access. This approach aligns with modern cybersecurity standards and promotes trust among users. Furthermore, the inclusion of Twilio enhances the system's communication efficiency by sending real-time SMS notifications to interns, keeping them informed about new tasks, updates, and approaching deadlines.

B. Future Scope

The Internship Task Manager project has been designed with scalability and flexibility in mind, providing a strong foundation for further improvements and expansions in the future. While the current version efficiently manages task assignments, communication, and tracking between supervisors and interns, there are several potential enhancements that can be implemented to make the system more intelligent, interactive, and feature-rich. One of the primary areas of future development is the integration of real-time analytics and reporting tools. By incorporating data visualization techniques using libraries such as Chart.js or Plotly, supervisors can gain insights into intern performance, task completion rates, and productivity trends. These analytics can assist in evaluating individual and team performance, helping organizations make data-driven decisions regarding intern assessments and project outcomes.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

The system can also be extended to support multi-level user roles and permissions. In addition to supervisors and interns, roles such as coordinators or administrators could be introduced to manage larger internship programs across departments. This hierarchical structure would enhance organizational control and allow better distribution of responsibilities.

In future versions, the inclusion of cloud deployment and scalability features will be essential. Hosting the application on cloud platforms such as AWS, Google Cloud, or Microsoft Azure would ensure greater accessibility, improved performance, and seamless scalability as the number of users increases. Cloud integration can also facilitate automated backups, ensuring data safety and reliability.

APPENDICES

A. Source Code

App.py:

```
from flask import Flask, render_template, redirect, url_for, flash, request
from flask_login import LoginManager, login_user, logout_user, login_required, current_user
from config import Config
from forms import LoginForm, RegisterForm, TaskForm, LeaveForm
from models import db, User, Task, Leave
from datetime import date
from apscheduler.schedulers.background import BackgroundScheduler
from datetime import datetime, timedelta
from twilio.rest import Client
import bcrypt
def send_sms(to_number, message):
    account_sid = 'ACb0cc546a645139c97458767213c22c72'
    auth_token = '7bb1817c0f5fa36b723eb292861da551'
    from_number = '+16282576861'
    client = Client(account_sid, auth_token)
    client.messages.create(
        body=message,
        from_=from_number,
        to=to_number
    )
def notify_upcoming_deadlines():
    now = datetime.now()
    soon = now + timedelta(hours=24)
    tasks = Task.query.filter(Task.deadline != None, Task.deadline > now, Task.deadline <= soon).all()
    for task in tasks:
        intern = User.query.get(task.assigned_to)
        if intern and intern.phone:
            message = f"Reminder: Your task '{task.title}' is due on {task.deadline.strftime('%Y-%m-%d %H:%M')}."
            send_sms(intern.phone, message)
scheduler = BackgroundScheduler()
scheduler.add_job(func=notify_upcoming_deadlines, trigger="interval", hours=1)
scheduler.start()
app = Flask(__name__)
app.config.from_object(Config)
db.init_app(app)
login_manager = LoginManager(app)
login_manager.login_view = "login"
def _scheduled_job_wrapper():
    with app.app_context():
```



```
    notify_upcoming_deadlines()
@login_manager.user_loader
def load_user(user_id):
  return db.session.get(User, int(user_id))

@app.route("/")
def home():
  return redirect(url_for("dashboard") if current_user.is_authenticated else url_for("login"))
@app.route("/register", methods=["GET","POST"])
def register():
  form = RegisterForm()
  if form.validate_on_submit():
    if User.query.filter_by(email=form.email.data).first():
      flash("Email already registered","danger")
    else:
     pw = bcrypt.hashpw(form.password.data.encode(), bcrypt.gensalt()).decode()
      u = User(
        name=form.name.data,
        email=form.email.data,
        phone=form.phone.data,  # <-- Add this line
        password_hash=pw,
        role=form.role.data
      )
      db.session.add(u)
      db.session.commit()
      flash("Registration successful. Login now.","success")
      return redirect(url_for("login"))
  return render_template("register.html", form=form)

@app.route("/login", methods=["GET","POST"])
def login():
  form = LoginForm()
  if form.validate_on_submit():
    user = User.query.filter_by(email=form.email.data).first()
 if user and bcrypt.checkpw(form.password.data.encode(), user.password_hash.encode()):
      login_user(user)
      return redirect(url_for("dashboard"))
    flash("Invalid credentials","danger")
  return render_template("login.html", form=form)
@app.rote("/logout")
@login_required
def logout():
  logout_user()
  return redirect(url_for("login"))
@app.route("/dashboard")
@login_required
def dashboard():
  if current_user.role == "supervisor":
   tasks = Task.query.order_by(Task.deadline.is_(None), Task.deadline.asc()).all()
    users = {u.id: u.name for u in User.query.all()}
```

```
        pending_count = sum(1 for t in tasks if t.status and t.status.lower() == "pending")
        completed_count = sum(1 for t in tasks if t.status and t.status.lower() == "completed")

        leaves = Leave.query.order_by(Leave.created_at.desc()).all()

        return render_template(
            "dashboard.html",
            tasks=tasks,
            leaves=leaves,
            supervisor=True,
            now=datetime.now,
            users=users,
            pending_count=pending_count,
            completed_count=completed_count
        )
    else:
        mytasks = Task.query.filter_by(assigned_to=current_user.id).all()
        pending_count = sum(1 for t in mytasks if t.status and t.status.lower() == "pending")
        completed_count = sum(1 for t in mytasks if t.status and t.status.lower() == "completed")
        my_leaves = Leave.query.filter_by(intern_id=current_user.id).order_by(Leave.created_at.desc()).all()
        return render_template(
            "dashboard.html",
            tasks=mytasks,
            leaves=my_leaves,
            supervisor=False,
            now=datetime.now,
            pending_count=pending_count,
            completed_count=completed_count
        )
@app.route("/tasks")
@login_required
def tasks():
    tasks = Task.query.all() if current_user.role=="supervisor" \
            else Task.query.filter_by(assigned_to=current_user.id).all()
    return render_template("tasks.html", tasks=tasks)
@app.route("/task/create", methods=["GET","POST"])
@login_required
def create_task():
    if current_user.role != "supervisor":
        flash("Only supervisor can create tasks","danger")
        return redirect(url_for("tasks"))
    form = TaskForm()
    interns = User.query.filter_by(role="intern").all()
    form.assigned_to.choices = [(0, "Unassigned")] + [(int(i.id), i.name) for i in interns]
    if form.validate_on_submit():
        deadline = None
        if form.deadline.data:
            try:
                deadline = datetime.strptime(form.deadline.data, "%Y-%m-%dT%H:%M")
            except:
```

```
            flash("Invalid date format","warning")
            return render_template("create_task.html", form=form)
        assignee = int(form.assigned_to.data) if form.assigned_to.data and int(form.assigned_to.data) != 0 else None
        t = Task(
            title=form.title.data,
            description=form.description.data,
            assigned_to=assignee,
            priority=form.priority.data,
            deadline=deadline,
            created_by=current_user.id
        )
        db.session.add(t)
        db.session.commit()
        # --- Add this block for immediate SMS ---
        if assignee:
            intern = User.query.get(assignee)
            if intern and intern.phone:
                message = f"You have been assigned a new task: '{form.title.data}'."
                if deadline:
                    message += f" Deadline: {deadline.strftime('%Y-%m-%d %H:%M')}."
                send_sms(intern.phone, message)
        # --- End SMS block ---
        flash("Task created","success")
        return redirect(url_for("tasks"))
    return render_template("create_task.html", form=form)
    if current_user.role != "supervisor":
        flash("Only supervisor can create tasks","danger")
        return redirect(url_for("tasks"))
    form = TaskForm()
    interns = User.query.filter_by(role="intern").all()
    form.assigned_to.choices = [(0, "Unassigned")] + [(int(i.id), i.name) for i in interns]
    if form.validate_on_submit():
        deadline = None
        if form.deadline.data:
            try:
                # Use the correct format for datetime-local input
                deadline = datetime.strptime(form.deadline.data, "%Y-%m-%dT%H:%M")
            except:
                flash("Invalid date format","warning")
                return render_template("create_task.html", form=form)
        assignee = int(form.assigned_to.data) if form.assigned_to.data and int(form.assigned_to.data) != 0 else None
        t = Task(
            title=form.title.data,
            description=form.description.data,
            assigned_to=assignee,
            priority=form.priority.data,
            deadline=deadline,
            created_by=current_user.id
        )
        db.session.add(t)
```

```
            db.session.commit()
            flash("Task created","success")
            return redirect(url_for("tasks"))
        return render_template("create_task.html", form=form)
    @app.route("/task/<int:tid>", methods=["GET", "POST"])
    @login_required
    def view_task(tid):
        task = Task.query.get_or_404(tid)
        if request.method == "POST":
            # Only allow assigned intern or supervisor to update
            if (current_user.role == "intern" and task.assigned_to == current_user.id) or current_user.role == "supervisor":
                # Update status
                new_status = request.form.get("status")
                if new_status:
                    task.status = new_status
                # Update deadline (optional, if you allow editing)
                new_deadline = request.form.get("deadline")
                if new_deadline:
                    try:
                        # Correct format for <input type="datetime-local">
                        task.deadline = datetime.strptime(new_deadline, "%Y-%m-%dT%H:%M")
                    except Exception:
                        flash("Invalid deadline format", "danger")
                db.session.commit()
                flash("Task updated", "success")
            return redirect(url_for("view_task", tid=tid))
        return render_template("view_task.html", task=task)
# Route: request leave (intern)
    @app.route("/leave/request", methods=["GET", "POST"])
    @login_required
    def request_leave():
        if current_user.role != "intern":
            flash("Only interns can request leave", "danger")
            return redirect(url_for("dashboard"))
        form = LeaveForm()
        if form.validate_on_submit():
            if form.end_date.data < form.start_date.data:
                flash("End date cannot be before start date", "danger")
            else:
                lv = Leave(
                    intern_id=current_user.id,
                    start_date=form.start_date.data,
                    end_date=form.end_date.data,
                    reason=form.reason.data,
                    status="Pending"
                )
                db.session.add(lv)
                db.session.commit()
                # ✅Notify supervisor(s) via SMS
                supervisors = User.query.filter_by(role="supervisor").all()
```

```
            message = (
                f"Leave Request: {current_user.name} has requested leave "
                f"from {form.start_date.data} to {form.end_date.data}. "
                f"Reason: {form.reason.data}."
            )
            for sup in supervisors:
                if sup.phone:
                    send_sms(sup.phone, message)
            flash("Leave requested successfully! Supervisor has been notified.", "success")
            return redirect(url_for("my_leaves"))
    return render_template("leave_request.html", form=form)
# Route: view leaves (intern sees own, supervisor sees all)
@app.route("/leaves")
@login_required
def my_leaves():
    if current_user.role == "supervisor":
        leaves = Leave.query.order_by(Leave.created_at.desc()).all()
    else:
        leaves = Leave.query.filter_by(intern_id=current_user.id).order_by(Leave.created_at.desc()).all()
    return render_template("leaves.html", leaves=leaves)
# Route: approve leave
@app.route("/leave/<int:lid>/approve", methods=["POST"])
@login_required
def approve_leave(lid):
    if current_user.role != "supervisor":
        flash("Unauthorized", "danger")
        return redirect(url_for("dashboard"))
    lv = Leave.query.get_or_404(lid)
    lv.status = "Approved"
    lv.reviewed_by = current_user.id
    db.session.commit()
    flash("Leave approved", "success")
    return redirect(url_for("my_leaves"))
# Route: reject leave
@app.route("/leave/<int:lid>/reject", methods=["POST"])
@login_required
def reject_leave(lid):
    if current_user.role != "supervisor":
        flash("Unauthorized", "danger")
        return redirect(url_for("dashboard"))
    lv = Leave.query.get_or_404(lid)
    lv.status = "Rejected"
    lv.reviewed_by = current_user.id
    db.session.commit()
    flash("Leave rejected", "info")
    return redirect(url_for("my_leaves"))
if __name__ == "__main__":
    with app.app_context():
        db.create_all()
        # schedule the job to run inside the app context
```

```
    scheduler.add_job(func=_scheduled_job_wrapper, trigger="interval", hours=1)
    # start scheduler once (inside __main__)
      app.run(debug=True)
```

forms.py:

```
    from flask_wtf import FlaskForm
    from wtforms import StringField, PasswordField, SubmitField, SelectField, TextAreaField
    from wtforms.validators import Regexp
    from wtforms.validators import DataRequired, Email, EqualTo, Length, ValidationError, Optional
    from models import User
    from wtforms import DateField, TextAreaField
    from wtforms.validators import DataRequired, Length

    # -------------------- RegisterForm --------------------
    class RegisterForm(FlaskForm):
      name = StringField("Full Name", validators=[DataRequired(), Length(min=3, max=50)])
      email = StringField("Email", validators=[DataRequired(), Email()])
      phone = StringField(
      "Phone Number",
      validators=[
        DataRequired(),
        Regexp(r'^\+\d{10,15}$', message="Enter phone number in international format, e.g., +919876543210")
      ]
    )
      password = PasswordField("Password", validators=[
        DataRequired(),
        Length(min=8, max=20, message="Password must be between 8 and 20 characters.")
      ])
      confirm = PasswordField("Confirm Password", validators=[
        DataRequired(),
        EqualTo("password", message="Passwords must match.")
      ])
      role = SelectField("Role", choices=[("intern", "Intern"), ("supervisor", "Supervisor")])
      submit = SubmitField("Register")

      def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
          raise ValidationError("Email already registered. Please login instead.")

    # -------------------- LoginForm --------------------
    class LoginForm(FlaskForm):
      email = StringField("Email", validators=[DataRequired(), Email()])
      password = PasswordField("Password", validators=[DataRequired()])
      submit = SubmitField("Login")

    # -------------------- TaskForm --------------------
    class TaskForm(FlaskForm):
      title = StringField("Title", validators=[DataRequired(), Length(max=100)])
      description = TextAreaField("Description", validators=[DataRequired()])
```

```
        priority = SelectField("Priority", choices=[("Low","Low"),("Medium","Medium"),("High","High")],
validators=[DataRequired()])
    assigned_to = SelectField("Assign To", coerce=int, validators=[Optional()])
    deadline = StringField("Deadline (YYYY-MM-DD HH:MM)", validators=[Optional()])
    submit = SubmitField("Create Task")


class LeaveForm(FlaskForm):
    start_date = DateField('Start Date', format='%Y-%m-%d', validators=[DataRequired()])
    end_date = DateField('End Date', format='%Y-%m-%d', validators=[DataRequired()])
    reason = TextAreaField('Reason', validators=[Length(max=500), Optional()])
    submit = SubmitField('Request Leave')
```

models.py

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from datetime import datetime

db = SQLAlchemy()
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=False)
    email = db.Column(db.String(150), unique=True, nullable=False)
    password_hash = db.Column(db.String(200), nullable=False)
    role = db.Column(db.String(20), default="intern")  # intern/supervisor
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    # models.py
    phone = db.Column(db.String(20))
    tasks_assigned = db.relationship("Task", backref="assignee", lazy=True)
class Task(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(200), nullable=False)
    description = db.Column(db.Text)
    status = db.Column(db.String(30), default="To-do")
    priority = db.Column(db.String(20), default="Normal")
    assigned_to = db.Column(db.Integer, db.ForeignKey("user.id"))
    deadline = db.Column(db.DateTime)
    created_by = db.Column(db.Integer)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
class Leave(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    intern_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    start_date = db.Column(db.Date, nullable=False)
    end_date = db.Column(db.Date, nullable=False)
    reason = db.Column(db.Text, nullable=True)
    status = db.Column(db.String(20), nullable=False, default="Pending")  # Pending / Approved / Rejected
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    reviewed_by = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=True)
    intern = db.relationship("User", foreign_keys=[intern_id], backref="leaves")
    reviewer = db.relationship("User", foreign_keys=[reviewed_by])
```

VII. ACKNOWLEDGEMENT

It is one of the most efficient tasks in life to choose the appropriate words to express one's gratitude to the beneficiaries. We are very much grateful to God who helped us all the way through the project and how molded us into what we are today.

We are grateful to our beloved Principal Dr. R. RADHAKRISHNAN, M.E., Ph.D., Adhiyamaan College of Engineering (An Autonomous Institution), Hosur for providing the opportunity to do this work in premises.

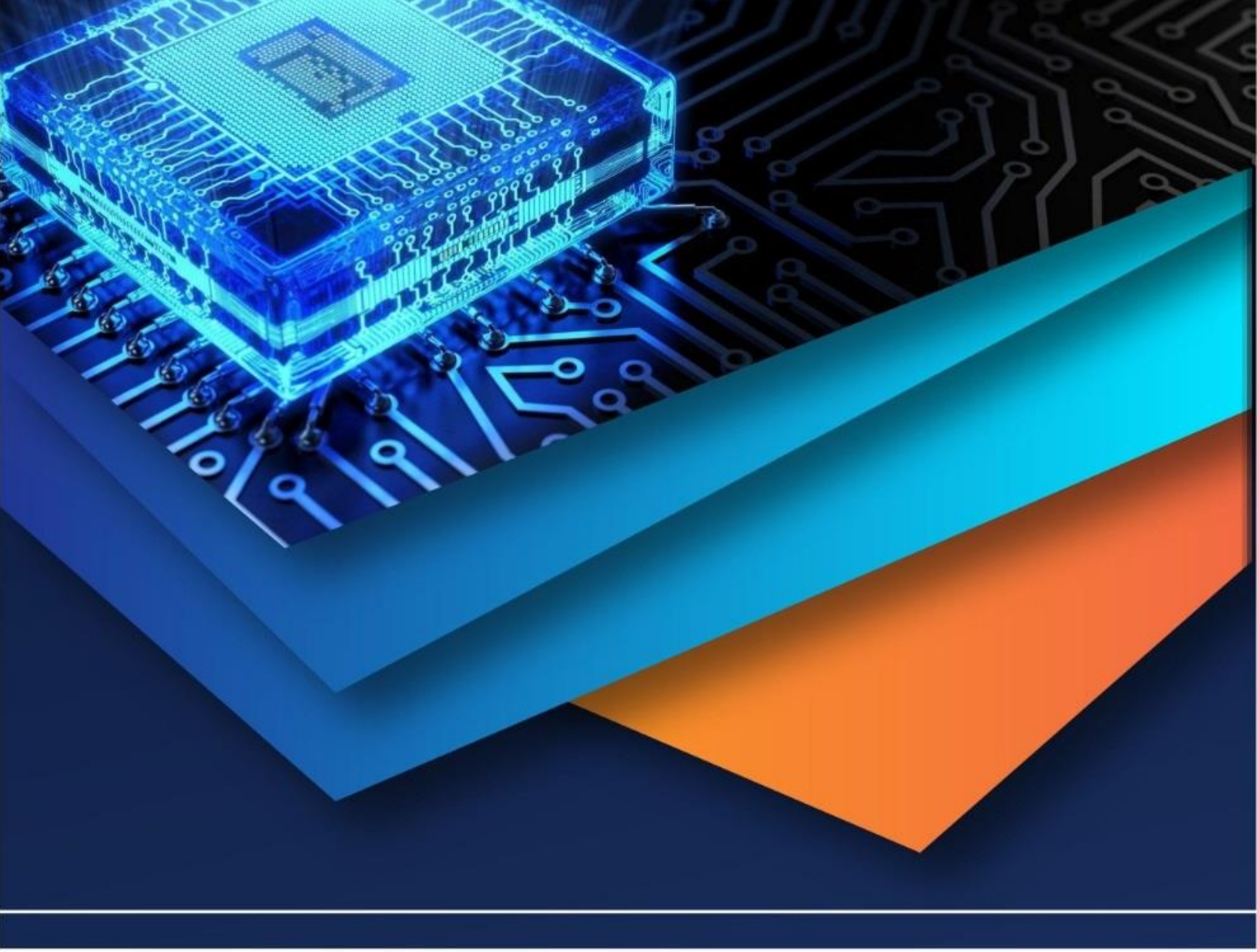
We acknowledge our heartful gratitude to Dr. G. FATHIMA, M.E., Ph.D., Professor and Head of the Department, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, for her guidance and valuable suggestions and encouragement throughout this project and made us to complete this project successfully.

We are highly indebted to Mrs. M. KASTHURI, M.E., (Ph.D.,) Supervisor, Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering(An Autonomous Institution), Hosur, whose immense support encouragement and valuable guidance were responsible to complete the project successfully.

We also extent our thanks to Project Coordinator and all Staff Members for their support in complete this project successfully. Finally, we would like to thank to our parents, without their motivational and support would not have been possible for us to complete this project successfully.

REFERENCES

[1] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python (2nd Edition). O'Reilly Media, Inc.
[2] Beazley, D., & Jones, B. K. (2013). Python Cookbook (3rd Edition). O'Reilly Media.
[3] Twilio Inc. (2024). Twilio SMS API Documentation. Retrieved from https://www.twilio.com/docs/sms
[4] The Flask Community. (2024). Flask Official Documentation. Retrieved from https://flask.palletsprojects.com/
[5] bcrypt Project Contributors. (2023). bcrypt Python Library Documentation. Retrieved from https://pypi.org/project/bcrypt/
[6] Real Python Team. (2024). Flask Tutorials and Web Development Guides. Retrieved from https://realpython.com/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY