CHAPTER 1

INTRODUCTION

In the context of educational institutions, efficient administrative processes are paramount for fostering an environment conducive to learning and growth. Utrackk emerges as a pioneering solution, meticulously engineered to streamline and automate key administrative functions within educational institutions. With a primary focus on enhancing communication, efficiency, and data security, Utrackk offers a comprehensive suite of features tailored to meet the diverse needs of students, staff, and administrators.

At its core, Utrackk prioritizes staff availability tracking, recognizing the pivotal role that faculty and staff members play in the seamless operation of educational institutions. Through real-time tracking mechanisms, Utrackk empowers users to monitor the availability of staff members, facilitating efficient scheduling of appointments, meetings, and consultations. By providing instant visibility into staff availability, Utrackk enables stakeholders to optimize their time and resources effectively.

Furthermore, Utrackk introduces a user-friendly web-based complaint system, designed to foster transparent communication and accountability within educational institutions. Through this platform, users can submit feedback and complaints anonymously, ensuring that concerns are addressed promptly and efficiently. By centralizing the complaint management process, Utrackk promotes a culture of openness and responsiveness, thereby strengthening the bond between stakeholders and the institution.

In addition to its communication-enhancing features, Utrackk incorporates a secure student mark entry system, aimed at safeguarding the integrity and confidentiality of academic records. Through this system, teachers can securely manage and enter student marks, ensuring that sensitive information remains protected at all times. By leveraging robust encryption and access control mechanisms, Utrackk mitigates the risk of unauthorized access or tampering, instilling confidence in both educators and students alike.

By integrating modern web technologies and best practices in data security, Utrackk sets a new standard for educational management systems. Its intuitive interface and comprehensive feature set empower users to navigate complex administrative tasks with ease, fostering a culture of efficiency and innovation within educational institutions. As educational landscapes continue to evolve, Utrackk stands poised to redefine the way institutions manage their administrative processes, ushering in a new era of efficiency, transparency, and excellence.

CHAPTER 2

LITERATURE SURVEY

[1] Technology Integration for Educational Management by Anderson & Dexter (2005): Explores the role of modern web technologies in streamlining administrative processes within educational institutions.

[2] Communication and Transparency in Educational Institutions by Hargie et al. (2019): Discusses effective communication channels, like complaint systems, to enhance trust and collaboration in educational settings.

[3] Designing Web Interfaces for User Experience by Nielsen (2000): Offers practical guidance on creating user-friendly interfaces for web-based applications, including insights into usability testing and design principles.

[4] Data Security and Privacy in Educational Settings by Solove (2006): Addresses legal and ethical considerations in data handling practices, particularly concerning sensitive information like student records.

[5] Organizational Effectiveness in Educational Settings by Leithwood et al. (2004): Explores strategies for optimizing administrative processes to improve resource utilization and achieve institutional goals.

[6] Diffusion of Innovations in Organizations by Rogers (2003): Analyzes the factors influencing the adoption and implementation of new technologies within organizational contexts, offering insights into managing organizational change.

CHAPTER 3

MODEL ARCHITECTURE

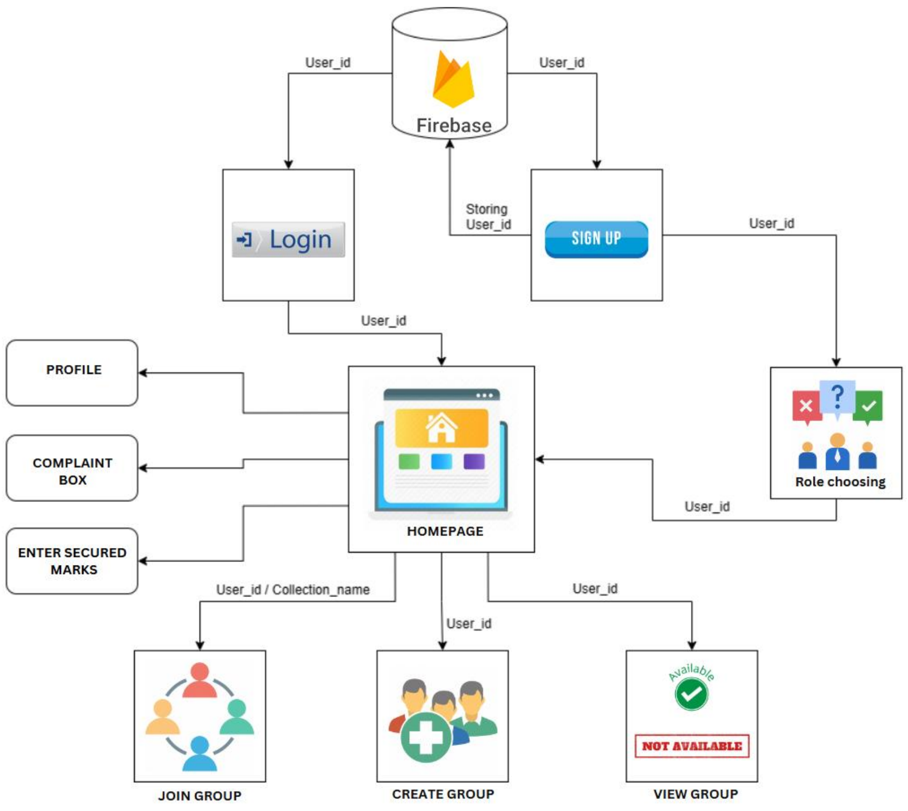


Fig.1: The Utrackk flowchart illustrates the process from staff and student registration and login to availability tracking within the institution.

The Utrackk system begins with staff and student login and signup, authenticated by the system and managed by administrators. Users can choose their role as staff or student. Staff can create and view groups, while students can join groups by typing the group name but have only read access. Additionally, there is a complaint box for posting complaints and a secured mark system where students can enter their marks for staff to review, reducing student insecurity about their marks. The architecture ensures efficient operations, user-friendly interactions, and robust administrative control.

CHAPTER 4
IMPLEMENTATION

Implementation of Utrackk

The implementation of Utrackk involves a systematic approach, focusing on key functionalities such as staff availability tracking, a web-based complaint box, and a secure student mark entering system. Here's how each aspect is addressed in the implementation.

Passengers provide personal details to create an account through a secure form, ensuring data completeness and correctness. User data is stored securely in Firebase, with each user assigned a unique identifier. Passwords are hashed using bcrypt for enhanced security. The front-end, developed with HTML, CSS, and JavaScript, ensures a responsive and user-friendly interface, while Flask handles form submissions and Firebase interactions on the back-end.

Real-time availability of staff is tracked using Firebase's real-time database capabilities. The system provides an intuitive interface for staff to update their availability status, which is then reflected in real-time for administrators and users.

Anonymous complaints and feedback are submitted through a secure web-based form. The system ensures transparency and accountability by routing complaints to appropriate authorities for resolution. Complaint data is securely stored in Firebase, with administrators having access to manage and resolve complaints efficiently.

Teachers securely manage and enter student marks through a dedicated interface. Each teacher has access only to the marks of their respective students, ensuring confidentiality. Firebase Firestore is used to store student mark data securely, with encryption techniques employed to protect sensitive information.

Administrators have access to a dashboard where they can view and manage staff availability, complaints, and student marks. The dashboard provides tools for updating staff schedules, resolving complaints, and generating reports. Flask and Firebase Authentication are utilized to ensure secure access and data management.

The front-end is built with HTML, CSS, and JavaScript for a responsive and user-friendly interface. Flask is used as the backend framework, handling server-side logic and database interactions. Firebase is employed for data storage, offering real-time database capabilities and secure data handling. Firebase Authentication ensures secure user authentication and access control.

Data security is prioritized throughout the implementation. Passwords are hashed using bcrypt, and sensitive information is encrypted using Firebase's security features. Firebase Authentication and HTTPS ensure secure data transmission and user authentication. Regular security audits are conducted to address potential vulnerabilities and ensure compliance with best practices.

Extensive testing, including unit tests, integration tests, and end-to-end tests, is conducted to ensure reliability and performance. User acceptance testing (UAT) gathers feedback from real users to make necessary adjustments.

In conclusion, Utrackk represents a robust platform that integrates modern web technologies and secure data handling practices to streamline administrative processes and enhance educational management.

By following this structured approach and leveraging modern web technologies and secure data handling practices, Utrackk aims to provide a comprehensive solution for educational management, enhancing efficiency and user experience for students, staff, and administrators.

CHAPTER 5
RESULTS AND DISCUSSIONS

The implementation of the Utrackk website emphasize its transformative impact on educational institutions. Through its key features such as staff availability tracking, a web-based complaint box, and a secure student mark entering system, Utrackk has significantly enhanced administrative efficiency and communication within these institutions. The staff availability tracking feature, for instance, has streamlined scheduling processes by providing real-time insights into staff availability, minimizing conflicts, and optimizing resource allocation. This has led to smoother coordination among staff members and improved productivity across various departments.

Additionally, the web-based complaint box has revolutionized institutional communication by offering stakeholders a platform to voice their feedback and concerns anonymously. This has fostered transparency and accountability, enabling administrators to address issues promptly and effectively. As a result, students, faculty, and staff feel more empowered to contribute to the improvement of institutional processes, leading to a more collaborative and supportive environment. Furthermore, the secure student mark entering system has modernized academic record management, ensuring data integrity and confidentiality while simplifying the mark entry process for teachers. This has not only reduced administrative burden but also enhanced the overall academic workflow, allowing educators to focus more on student engagement and learning outcomes.

APPENDIX

Sample code

Home page:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Ubuntu:ital,wght@0,300;0,400
;0,500;0,700;1,300;1,400;1,500;1,700&display=swap" rel="stylesheet">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Homepage</title>
<style>
  body {
    font-family: "Ubuntu", sans-serif;
    background: rgb(47,180,191);
    background: linear-gradient(90deg, rgba(47,180,191,1) 0%,
rgba(20,66,102,1) 100%);
    margin: 0;
    padding: 0;
  }

  /* Navigation Bar */
  .navbar {
    background-color: #203647;
    color: #fff;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px 20px;
    box-shadow: 6px 4px 5px #203647;
  }

  .navbar-brand {
    font-size: 24px;
    font-weight: bold;
  }

  .nav-links {
    display: none;
    flex-direction: column;
    position: absolute;
    top: 50px;
    right: 20px;
    background-color: #203647;
    padding: 10px;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    border: 2px solid white;
}

.nav-links a {
    color: #fff;
    text-decoration: none;
    padding: 5px 0;
    transition: all 0.3s ease;
}

.nav-links a:hover {
    background-color: #4da8da;
    padding: 3px;
}

/* Main Content */
.main-content {
    padding: 20px;
    margin-bottom: 15%;
    margin-top: 4%;
    color: #203647;
}
h2{
    text-align: center;
}
.block {
    background-color: #fff;
    padding: 15px;
    margin-bottom: 20px;
    border-radius: 5px;
    width: 60%;
    margin:0 auto 20px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    transition: all 0.3s ease;
}
.block:hover{
    width: 61%;
    }
    @media screen and (max-width: 480px) {
    .main-content {
        padding: 20px;
        margin-bottom: 15%;
        margin-top: 10%;
        color: #203647;
    }
    .block {
        background-color: #fff;
        padding: 2px;
        margin-bottom: 20px;
        border-radius: 5px;
        width: 70%;
        margin:0 auto 20px;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
        transition: all 0.3s ease;
    }
    .block:hover{
        width: 71%;
    }
    }
    /* Footer */
    footer {
        background-color: #203647;
        color: #fff;
        text-align: center;
        padding: 10px 0;
        position: fixed;
        bottom: 0;
        width: 100%;
    }
</style>
</head>
<body>

<!-- Navigation Bar -->
<div class="navbar">
    <div class="navbar-brand">
        <span>UTRACKK.com</span>
    </div>
    <div class="dropdown">
        <span onclick="toggleNavLinks()">Menu &#9662;</span>
        <div class="nav-links" id="navLinks">
            <a href="#">Availability Check</a>
            <a href="#">Complaint Box</a>
            <a href="#">Enter Marks</a>
            <a href="#">Contact Us</a>
            <a href="#">Profile</a>
        </div>
    </div>
</div>

<!-- Main Content -->
<div class="main-content">
    <h1 style="color: #fff;text-align: center;">Welcome
{{utrackk_name}}!</h1>
    <a href="/group_joined/{{user_id}}" style="text-decoration: none;color:
#203647 "><div class="block">
        <h2>Your Group</h2>
    </div></a>
    <a href="/join_group/{{user_id}}" style="text-decoration: none;color:
#203647 "><div class="block">
        <h2>Join Group</h2>
    </div></a>
    <a href="/create_group/{{user_id}}" style="text-decoration: none;color:
#203647 "><div id="dynamic-content"></div></a>
</div>
</div>

<!-- Footer -->
<footer>
    <p>Terms and Conditions | Contact Us | All Rights Reserved</p>
</footer>

<script>
    var role = '{{role}}';
    function toggleNavLinks() {
        var navLinks = document.getElementById("navLinks");
        if (navLinks.style.display === "flex") {
            navLinks.style.display = "none";
        } else {
            navLinks.style.display = "flex";
        }
    }
    function createDynamicBlock() {
        var dynamicContent = document.getElementById("dynamic-content");
        var block = document.createElement("div");
        block.className = "block";
        block.innerHTML = "<h2>Create Group</h2>";
        dynamicContent.appendChild(block);
    }

    if (role === 'staff') {
        createDynamicBlock();
    }
</script>

</body>
</html>
```

Join Group :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Ubuntu:ital,wght@0,300;0,400
;0,500;0,700;1,300;1,400;1,500;1,700&display=swap" rel="stylesheet">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@10"></script>
<title>Join Group</title>
<style>
    body {
        font-family: "Ubuntu", sans-serif;
        background: rgb(47,180,191);
        background: linear-gradient(90deg, rgba(47,180,191,1) 0%,
rgba(20,66,102,1) 100%);
        margin: 0;
        padding: 0;
        display: flex;
        flex-direction: column;
        align-items: center;
        min-height: 100vh;
        text-align: center;
        color: #203647;
    }

    .container {
        background-color: rgba(255, 255, 255, 0.8);
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        padding: 20px;
        width: 90%;
        max-width: 600px;
        margin-top: 20px;
        animation: fadein 1s;
    }

    @keyframes fadein {
        from { opacity: 0; }
        to   { opacity: 1; }
    }

    h2 {
        margin-bottom: 20px;
        color: #203647;
    }

    .search-bar {
        display: flex;
        margin-bottom: 20px;
    }

    .search-bar input[type="text"] {
        width: 80%;
        padding: 10px;
        border: 1px solid #ccc;
        border-radius: 5px 0 0 5px;
        font-size: 16px;
    }

    .search-bar button {
        width: 20%;
        padding: 10px;
        border: none;
        background-color: #4da8da;
        color: #fff;
        border-radius: 0 5px 5px 0;
        cursor: pointer;
        transition: background-color 0.3s;
    }

    .search-bar button:hover {
        background-color: #4da8da;
    }

    .results {
        margin-top: 20px;
    }

    .result-block {
        background-color: #fff;
        padding: 15px;
        margin-bottom: 10px;
        border-radius: 5px;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        display: flex;
        justify-content: space-between;
        align-items: center;
    }

    .result-block:hover {
        background-color: #f0f8ff;
    }

    .join-btn {
        background-color: green;
        color: #fff;
        border: none;
        padding: 10px;
        width: 80px;
        border-radius: 5px;
        cursor: pointer;
        transition: background-color 0.3s;
    }

    .join-btn:hover {
        background-color: darkgreen;
    }

    @media (max-width: 480px) {
        .container {
            padding: 10px;
        }
        .join-btn {
        background-color: green;
        color: #fff;
        border: none;
        padding: 10px;
        width: 60px;
        border-radius: 5px;
        cursor: pointer;
        transition: background-color 0.3s;
    }
        .search-bar input[type="text"] {
            width: 70%;
        }

        .search-bar button {
            width: 30%;
        }
    }
    .loading-screen {
        display: none;
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background-color: rgba(0, 0, 0, 0.7); /* Semi-transparent background */
        justify-content: center;
        align-items: center;
        animation: fadeOut 2s ease forwards; /* Fade out animation */
    }

    .loader {
        width: 100px;
        height: 100px;
        display: flex;
        justify-content: space-between;
        align-items: center;
    }

    .loader div {
        width: 20px;
        height: 20px;
        background-color: #fff; /* Dot color */
        border-radius: 50%;
        animation: bounce 0.6s infinite ease-in-out alternate; /* Dot animation */
    }

    .loader div:nth-child(1) {
        animation-delay: 0.2s;
    }

    .loader div:nth-child(2) {
        animation-delay: 0.4s;
    }

    .loader div:nth-child(3) {
        animation-delay: 0.6s;
    }

    @keyframes bounce {
        to {
            transform: translateY(-30px);
        }
    }

</style>
</head>
<body>

<div class="container">
    <h2>Join Group</h2>
    <div class="search-bar">
        <input type="text" id="searchInput" placeholder="Search group name...">
        <button onclick="searchGroup()" id = "search">Search</button>
    </div>
    <div class="results" id="results"></div>
</div>

<div id = "loading-screen" class="loading-screen">
    <div class="loader">
        <div></div>
        <div></div>
        <div></div>
    </div>
</div>

<script>
    function searchGroup() {
        document.getElementById('loading-screen').style.display = "flex";
        const query = document.getElementById('searchInput').value;
        fetch('/search-group', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ query }),
        })
        .then(response => response.json())
        .then(data => {
            const resultsDiv = document.getElementById('results');
            resultsDiv.innerHTML = ''; // Clear previous results
            if (data.length === 0) {
                const notFoundMessage = document.createElement('p');
                notFoundMessage.textContent = 'No group found!';
                document.getElementById('loading-screen').style.display = "none";
                resultsDiv.appendChild(notFoundMessage);
            } else {data.forEach(group => {
                const block = document.createElement('div');
                block.className = 'result-block';
                block.innerHTML = `
                    <span>${group.name} (${group.count} members)</span>
                    <button class="join-btn" id = "join"
onclick="joinGroup('${group.name}')">Join</button>
                `;
                resultsDiv.appendChild(block);
            });
            document.getElementById('loading-screen').style.display = "none";
        }
        })
        .catch(error => {console.error('Error:',
error);document.getElementById('loading-screen').style.display = "none";});

    }

    function joinGroup(groupName) {
        document.getElementById('loading-screen').style.display = "flex";
        fetch('/add_group/{{user_id}}', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ groupName , }),
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                document.getElementById('loading-screen').style.display = "none";
                Swal.fire({
                icon: 'success',
                title: 'Successfully Joined',
                text: 'You can view the group in Your group section'
                });
            } else {
```

```
            document.getElementById('loading-screen').style.display = "none";
            Swal.fire({
            icon: 'error',
            title: 'Failed to Join!',
            text: 'You are already Joined in this group'
            });
          }
        })
        .catch(error => {console.error('Error:',
error);document.getElementById('loading-screen').style.display = "none";});

    }
</script>

</body>
</html>
```

View Group:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Ubuntu:ital,wght@0,300;0,400
;0,500;0,700;1,300;1,400;1,500;1,700&display=swap" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@10"></script>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Group Details</title>
<style>
    body {
        font-family: "Ubuntu", sans-serif;
        background: rgb(47,180,191);
        background: linear-gradient(90deg, rgba(47,180,191,1) 0%,
rgba(20,66,102,1) 100%);
        margin: 0;
        padding: 0;
        color: #203647;
    }
    .container {
        max-width: 800px;
        margin: 20px auto;
        padding: 20px;
        border-radius: 10px;
```

```
}
h2 {
   margin-top: 50px;
   text-align: center;
   font-weight: bold;
   color: #fff;
}
.member-block {
   margin-top: 20px;
   padding: 10px;
   background-color: rgba(255, 255, 255, 0.6);
   border-radius: 5px;
   box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
   text-align: center;
}
.member-name {
   font-weight: bold;
}
.availability {
   color: green;
}
.availability.unavailable {
   color: red;
}
.change-btn {
   margin-top: 10px;
   padding: 5px 10px;
   background-color: #203647;
   color: #fff;
   border: none;
   border-radius: 5px;
   cursor: pointer;
   display: block;
   width: 100%;
   max-width: 200px;
   margin: 10px auto;
}
.leave-btn {
   display: block;
   margin: 20px auto;
   padding: 10px 20px;
   background-color: #ff3333;
   color: #fff;
   border: none;
   border-radius: 5px;
```

```
  cursor: pointer;
}
@media (max-width: 600px) {
  .container {
    padding: 10px;
  }
  .change-btn, .leave-btn {
    width: 100%;
    max-width: 100%;
  }
}
.loading-screen {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.7); /* Semi-transparent background */
  justify-content: center;
  align-items: center;
  animation: fadeOut 2s ease forwards; /* Fade out animation */
}

.loader {
  width: 100px;
  height: 100px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.loader div {
  width: 20px;
  height: 20px;
  background-color: #fff; /* Dot color */
  border-radius: 50%;
  animation: bounce 0.6s infinite ease-in-out alternate; /* Dot animation */
}

.loader div:nth-child(1) {
  animation-delay: 0.2s;
}

.loader div:nth-child(2) {
        animation-delay: 0.4s;
    }

    .loader div:nth-child(3) {
        animation-delay: 0.6s;
    }

    @keyframes bounce {
        to {
            transform: translateY(-30px);
        }
    }
</style>
</head>
<body>
    <h2>{{grp_name}}</h2>
<div class="container">
    <!-- Member Blocks will be added dynamically here -->
</div>
<button class="leave-btn" onclick="leaveGroup()">Leave Group</button>

<div id = "loading-screen" class="loading-screen">
    <div class="loader">
        <div></div>
        <div></div>
        <div></div>
    </div>
</div>

<script>
    const collection_name = '{{grp_name}}';
    const userEmail = '{{user_email}}';
    document.getElementById('loading-screen').style.display = "flex";
    function createMemberBlock(member, isCurrentUser) {
        const container = document.querySelector('.container');
        const memberBlock = document.createElement('div');
        memberBlock.classList.add('member-block');

        if (isCurrentUser) {
            const youLabel = document.createElement('div');
            youLabel.textContent = 'You';
            youLabel.style.fontWeight = 'bold';
            memberBlock.appendChild(youLabel);
        }
        const nameElement = document.createElement('div');
        nameElement.classList.add('member-name');
        nameElement.textContent = member.utrackk_name;

        const availabilityElement = document.createElement('div');
        availabilityElement.textContent = member.availability ? '(Available)' :
'(Not Available)';
        availabilityElement.classList.add('availability');
        if (!member.availability) {
            availabilityElement.classList.add('unavailable');
        }

        const venueElement = document.createElement('div');
        venueElement.textContent = 'Venue: ' + member.venue;

        memberBlock.appendChild(nameElement);
        memberBlock.appendChild(document.createElement('br'));
        memberBlock.appendChild(availabilityElement);
        memberBlock.appendChild(document.createElement('br'));
        memberBlock.appendChild(venueElement);

        if (isCurrentUser) {
            const changeAvailabilityBtn = document.createElement('button');
            changeAvailabilityBtn.textContent = 'Change Availability';
            changeAvailabilityBtn.classList.add('change-btn');
            changeAvailabilityBtn.onclick = () =>
updateAvailability(member.email);

            const changeVenueBtn = document.createElement('button');
            changeVenueBtn.textContent = 'Change Venue';
            changeVenueBtn.classList.add('change-btn');
            changeVenueBtn.onclick = () => updateVenue(member.email);


            memberBlock.appendChild(changeAvailabilityBtn);

            memberBlock.appendChild(changeVenueBtn);
        }

        if (isCurrentUser) {
            container.insertBefore(memberBlock, container.firstChild);
        } else {
            container.appendChild(memberBlock);
        }
        document.getElementById('loading-screen').style.display = "none";
```

```
    }

    function updateAvailability(email) {
        document.getElementById('loading-screen').style.display = "flex";
        fetch(`/update-availability/${collection_name}`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ email }),
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                document.getElementById('loading-screen').style.display = "none";
                location.reload();
            } else {
                document.getElementById('loading-screen').style.display = "none";
                alert('Error updating availability.');
            }
        })
        .catch(error => {console.error('Error:',
error);document.getElementById('loading-screen').style.display = "none";});
    }

    function updateVenue(email) {
        const newVenue = prompt('Enter new venue:');
        document.getElementById('loading-screen').style.display = "flex";
        if (newVenue) {
            fetch(`/update-venue/${collection_name}`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ email, venue: newVenue }),
            })
            .then(response => response.json())
            .then(data => {
                if (data.success) {
                    document.getElementById('loading-screen').style.display = "none";
                    location.reload();
                } else {
                    document.getElementById('loading-screen').style.display = "none";
                    alert('Error updating venue.');
                }
            })
            .catch(error => {console.error('Error:',
error);document.getElementById('loading-screen').style.display = "none";});
        }
        document.getElementById('loading-screen').style.display = "none";
    }

    function leaveGroup() {
        document.getElementById('loading-screen').style.display = "flex";
        fetch(`/leave-group/${collection_name}`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ email: userEmail }),
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                document.getElementById('loading-screen').style.display = "none";
                alert('You have left the group.');
                window.location.href = '/';
            } else {
                document.getElementById('loading-screen').style.display = "none";
                alert('Error leaving group.');
            }
        })
        .catch(error => {console.error('Error:',
error);document.getElementById('loading-screen').style.display = "none";});
    }

    fetch(`/view-group-details/${collection_name}/${userEmail}`)
        .then(response => response.json())
        .then(data => {
            data.forEach(member => {
                const isCurrentUser = member.email === userEmail;
                createMemberBlock(member, isCurrentUser);
            });
        })
        .catch(error => console.error('Error:', error));
</script>

</body>
</html>
```
Flask Framework:

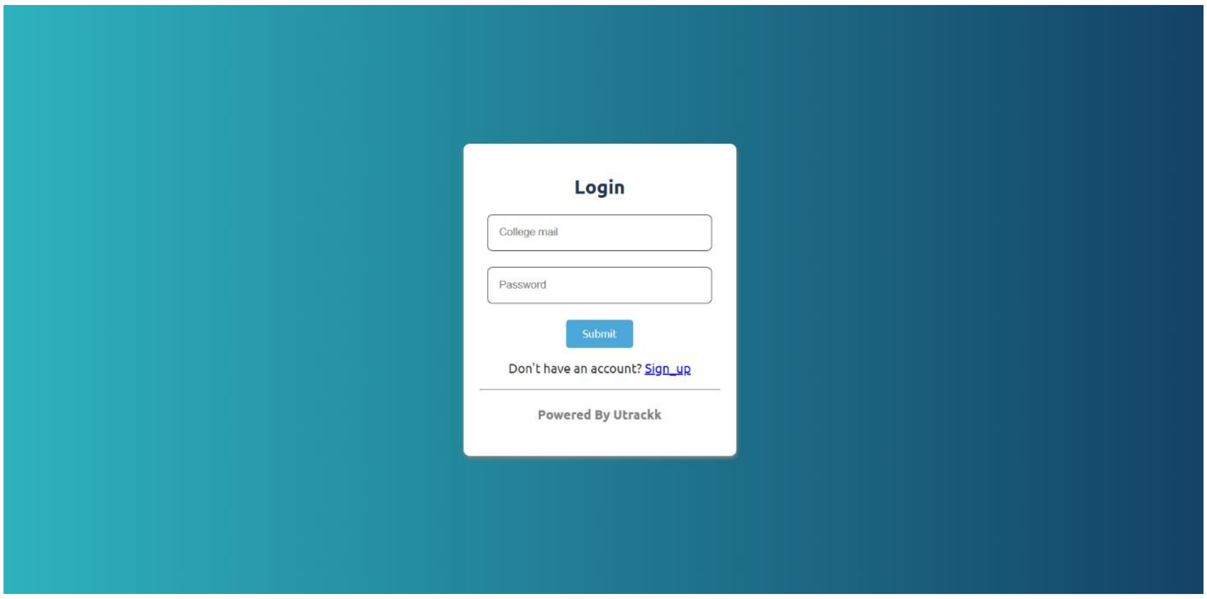
```
from flask import Flask,render_template,redirect,request,jsonify,url_for
import firebase_admin
from firebase_admin import credentials
from firebase_admin import auth
from firebase_admin import firestore

app = Flask(__name__)

cred = credentials.Certificate('C:/Users/SIVA/Utrackk/webpage/utrackk-ee8ab-
firebase-adminsdk-6n9l5-15d8827dcb.json')
firebase_admin.initialize_app(cred)
db = firestore.client()


# home page
@app.route("/")
def homescreen():
    return render_template('login.html')

# Login call
@app.route("/login", methods =["POST"])
def login():
    data = request.get_json()
    email = data.get("email")
    password = data.get("password")

    try:
        user = auth.get_user_by_email(email)
        user_ref = db.collection('users').document(user.uid)
        user_data = user_ref.get().to_dict()
        if user_data and user_data.get('password') == password:
            utrackk_name = user_data.get('utrackk_name', '')
            role = user_data.get('role', '')
            return jsonify({'success': True, 'redirect_url':
'/homepage/{}/{}/{}'.format(utrackk_name,role,user.uid)})
        else:
            return jsonify({'success': False, 'error': 'Invalid credentials'})

    except Exception as e:
        print("Error:", e)
        return jsonify({'success': False, 'error': 'An error occurred'})

# Logout call
@app.route("/logout")
def logout():
    return redirect('/')

# Sign_up page
@app.route("/sign_up")
def sign_up():
    return render_template('sign_up.html')

# Storing account
@app.route("/sign_up/account_added", methods=["GET", "POST"])
def account_added():
    if request.method == "POST":
        data = request.get_json()
        email = data.get("email")
        password = data.get("password")
        try:
            user = auth.create_user(
                email=email,
                password=password
            )

            user_ref = db.collection('users').document(user.uid)
            user_ref.set({
                'email': email,
                'password': password
            })
            return jsonify({"success": True, "user_id": user.uid})
        except Exception as e:
            return jsonify({"message": str(e)}), 500
    return render_template('sign_up.html')

# enter name page
@app.route('/enter_name/<string:user_id>')
def enter_name(user_id):
    return render_template('enter_name.html',user_id=user_id)

@app.route('/store_name/<string:user_id>', methods = ['GET','POST'])
def store_name(user_id):
    if request.method == "POST":
        data = request.get_json()
        utrackk_name = data.get("utrackk_name")
        role = data.get("role")
        try:
            user_ref = db.collection('users').document(user_id)
            user_ref.update({
                'role': role ,
            'utrackk_name' : utrackk_name
          })
          return jsonify({"success": True, "user_id": user_id, "utrackk_name":
utrackk_name ,"role" : role})
      except Exception as e:
          return jsonify({"message": str(e)}), 500
   return render_template('enter_name.html',user_id =  user_id)

# homepage
@app.route('/homepage/<string:utrackk_name>/<string:role>/<string:user_id>'
)
def homepage(utrackk_name,role,user_id):
   return render_template('homepage.html',user_id = user_id,utrackk_name =
utrackk_name,role = role)

# group_joined
@app.route("/group_joined/<string:user_id>")
def group_joined(user_id):
   return render_template('group_joined.html',user_id = user_id)

@app.route('/get-groups/<string:user_id>', methods=['GET'])
def get_groups(user_id):
   try:
      user_ref = db.collection('users').document(user_id)
      user_doc = user_ref.get()
      if user_doc.exists:
         user_data = user_doc.to_dict()
         user_email = user_data.get('email')
      else:
         return jsonify({'success': False, 'message': 'User not found'})
   except Exception as e:
      return jsonify({'success': False, 'message': str(e)})

   collections = db.collections()
   user_groups = []

   for collection in collections:
      if collection.id != 'users':
         docs = collection.stream()
         member_count = 0
         is_member = False

         for doc in docs:
            doc_data = doc.to_dict()
            member_count += 1
```

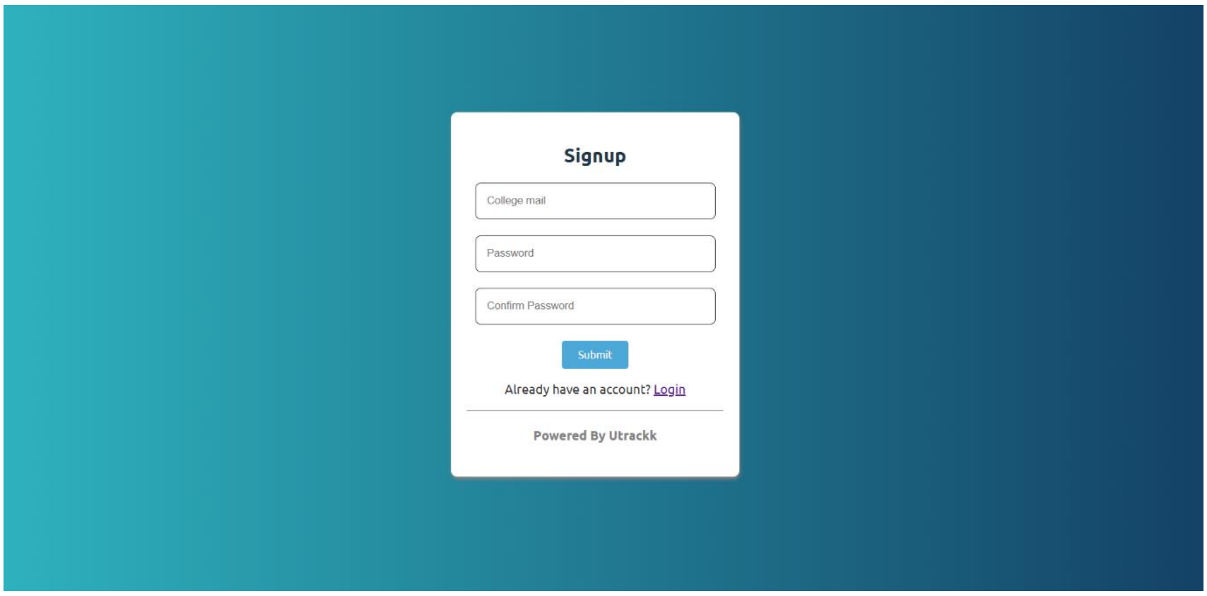
```
                if doc_data.get('email') == user_email:
                    is_member = True

            if is_member:
                user_groups.append({'name': collection.id, 'count': member_count})

    return jsonify(user_groups)

# group details
@app.route("/groups_joined/group_details/<string:grp_name>/<string:user_id>
")
def group_details(grp_name,user_id):
    try:
        user_ref = db.collection('users').document(user_id)
        user_doc = user_ref.get()
        if user_doc.exists:
            user_data = user_doc.to_dict()
            user_email = user_data.get('email')
            return render_template('group_details.html',grp_name = grp_name,
user_id = user_id, user_email = user_email)
        else:
            print("user_not_found")
    except Exception as e:
        print("error in groups details rendering")


@app.route('/view-group-details/<string:collection_name>/<string:user_id>',
methods=['GET'])
def view_group_details(collection_name,user_id):
    collection_ref = db.collection(collection_name)
    docs = collection_ref.stream()

    members = []
    for doc in docs:
        doc_data = doc.to_dict()
        if doc_data.get('role') == 'staff':
            member_info = {
                'utrackk_name': doc_data.get('utrackk_name'),
                'availability': doc_data.get('availability'),
                'venue': doc_data.get('venue'),
                'email': doc_data.get('email')
            }
            members.append(member_info)

    return jsonify(members)
```

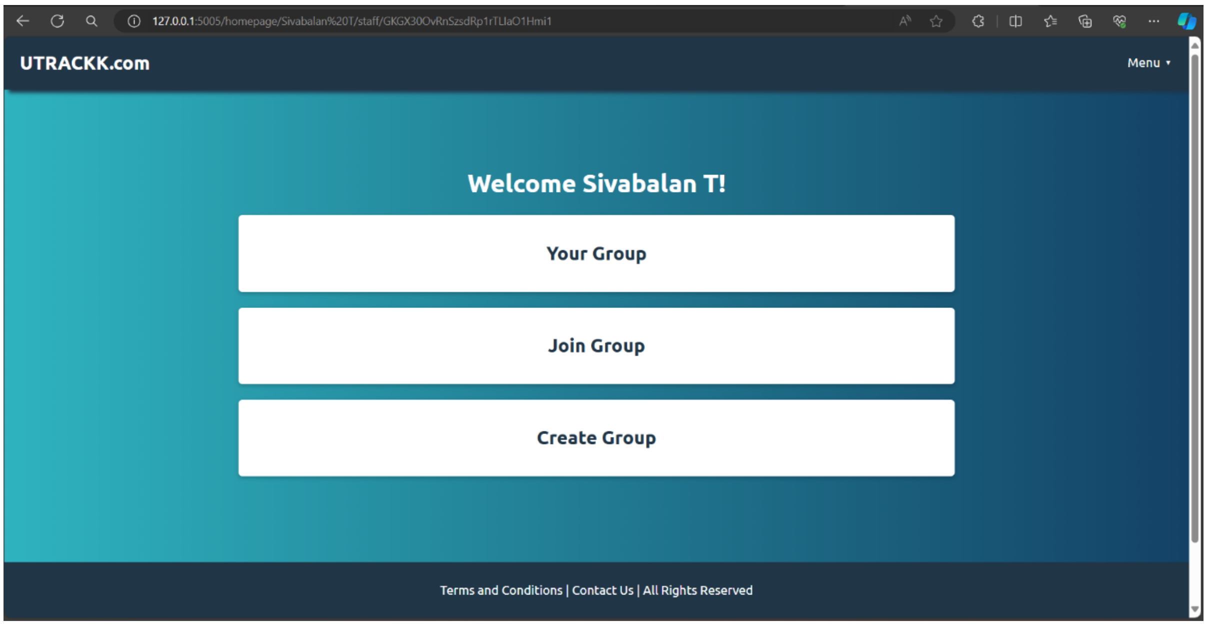
```
@app.route('/update-availability/<string:collection_name>', methods=['POST'])
def update_availability(collection_name):
    data = request.json
    email = data.get('email')

    collection_ref = db.collection(collection_name)
    docs = collection_ref.where('email', '==', email).stream()

    for doc in docs:
        doc.reference.update({
            'availability': not doc.to_dict().get('availability', False)
        })

    return jsonify({'success': True})

@app.route('/update-venue/<string:collection_name>', methods=['POST'])
def update_venue(collection_name):
    data = request.json
    email = data.get('email')
    new_venue = data.get('venue')

    collection_ref = db.collection(collection_name)
    docs = collection_ref.where('email', '==', email).stream()

    for doc in docs:
        doc.reference.update({
            'venue': new_venue
        })

    return jsonify({'success': True})

@app.route('/leave-group/<string:collection_name>', methods=['POST'])
def leave_group(collection_name):
    data = request.json
    user_email = data.get('email')

    collection_ref = db.collection(collection_name)
    docs = collection_ref.where('email', '==', user_email).stream()

    for doc in docs:
        doc.reference.delete()

    return jsonify({'success': True})
```

```
# join group
@app.route("/join_group/<string:user_id>")
def join_group(user_id):
    return render_template('join_group.html',user_id = user_id)

@app.route('/search-group', methods=['POST'])
def search_group():
    data = request.json
    query = data.get('query')

    if not query:
        return jsonify([])

    groups_ref = db.collections()
    groups = []

    for collection in groups_ref:
        if collection.id != 'users':
            if query.lower() in collection.id.lower():
                doc_count = len(list(collection.stream()))
                groups.append({'name': collection.id, 'count': doc_count})
    return jsonify(groups)

@app.route('/add_group/<string:user_id>', methods=['POST'])
def add_group(user_id):
    data = request.json
    group_name = data.get('groupName')

    if not group_name:
        return jsonify({'success': False, 'message': 'Group name is required'})

    # Fetch user details from Firestore using user_id
    try:
        user_ref = db.collection('users').document(user_id)
        user_doc = user_ref.get()
        if user_doc.exists:
            user_data = user_doc.to_dict()
            user_email = user_data.get('email')
            user_role = user_data.get('role')
            user_name = user_data.get('utrackk_name')
        else:
            return jsonify({'success': False, 'message': 'User not found'})
    except Exception as e:
        return jsonify({'success': False, 'message': str(e)})

    if not user_email or not user_role:
        return jsonify({'success': False, 'message': 'User email or role is missing'})
    try:
        group_ref = db.collection(group_name)
        query = group_ref.where('email', '==', user_email).stream()
        if any(query):
            return jsonify({'success': False, 'message': 'Email already exists in the
group'})
    except Exception as e:
        return jsonify({'success': False, 'message': str(e)})
    try:
        if user_role == "staff":
            group_ref.add({
                'email': user_email,
                'role': user_role,
                'availability': False,
                'venue': "none",
                'utrackk_name': user_name
            })
        else:
            group_ref.add({
                'email': user_email,
                'role': user_role,
                'utrackk_name': user_name
            })
    except Exception as e:
        return jsonify({'success': False, 'message': str(e)})

    return jsonify({'success': True})

# join group
@app.route("/create_group/<string:user_id>")
def create_group(user_id):
    return render_template('create_group.html',user_id = user_id)

@app.route("/create_group/store/<string:user_id>" , methods = ['GET','POST'])
def store(user_id):
    data = request.json
    group_name = data.get('group_name')
    description = data.get('description')
    user_id = data.get('user_id')
    group_ref = db.collection(group_name).get()
    if group_ref:
        return jsonify({"success": False, "message": "Group name already
exists"})

    # Fetch user data from Firestore using user_id
    user_ref = db.collection('users').document(user_id)
    user_data = user_ref.get().to_dict()

    if user_data:
        group_ref = db.collection(group_name)
        group_ref.document(user_id).set({
            'email': user_data.get('email'),
            'role': user_data.get('role'),
            'availability': False,
            'description': description,
            'group_name': group_name,
            'venue' : "none",
            'utrackk_name' : user_data.get('utrackk_name')
        })

        return jsonify({"success": True})
    else:
        return jsonify({"success": False, "message": "User not found"})


# complaint box page
@app.route("/complaint_box")
def complaint_box():
    return render_template('complaint_box.html')

# mark entering
@app.route("/enter_secured_mark")
def enter_secured_mark():
    return render_template('enter_mark.html')

#profile page
@app.route("/profile")
def profile():
    return "profile with edit icon interms of form (easy approach)"

if  __name__ == "__main__":
    app.run(host='0.0.0.0',debug=True,port=5005)
```

OUTPUT SCREENSHOTS:







127.0.0.1:5005/homepage/Sivabalan%20T/staff/GKGX30OvRnSzsdRp1rTLlaO1Hmi1

UTRACKK.com

Menu ▾

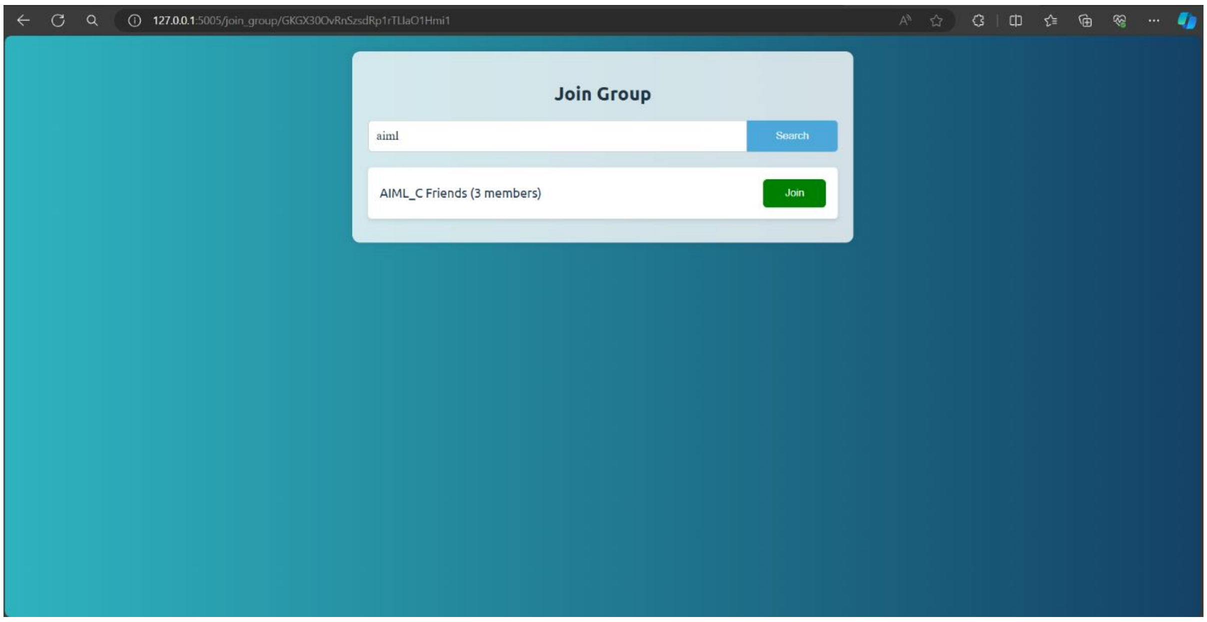
Welcome Sivabalan T!

Your Group

Join Group

Create Group

Terms and Conditions | Contact Us | All Rights Reserved



127.0.0.1:5005/join_group/GKGX30OvRnSzsdRp1rTLlaO1Hmi1

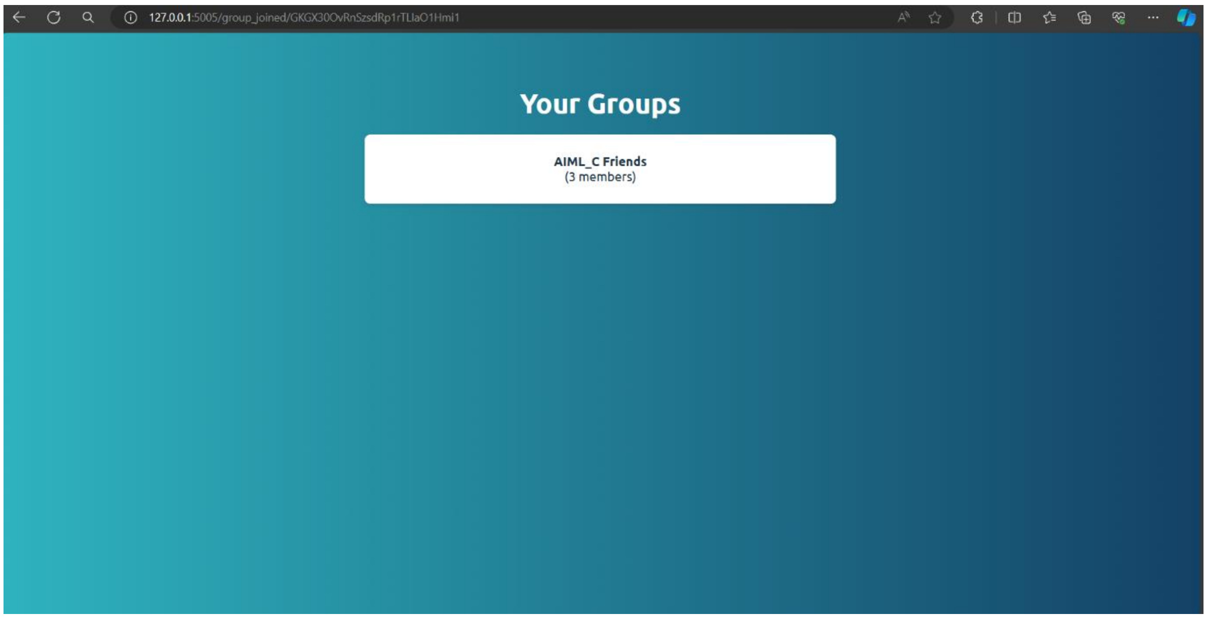
Join Group

aiml

Search

AIML_C Friends (3 members)

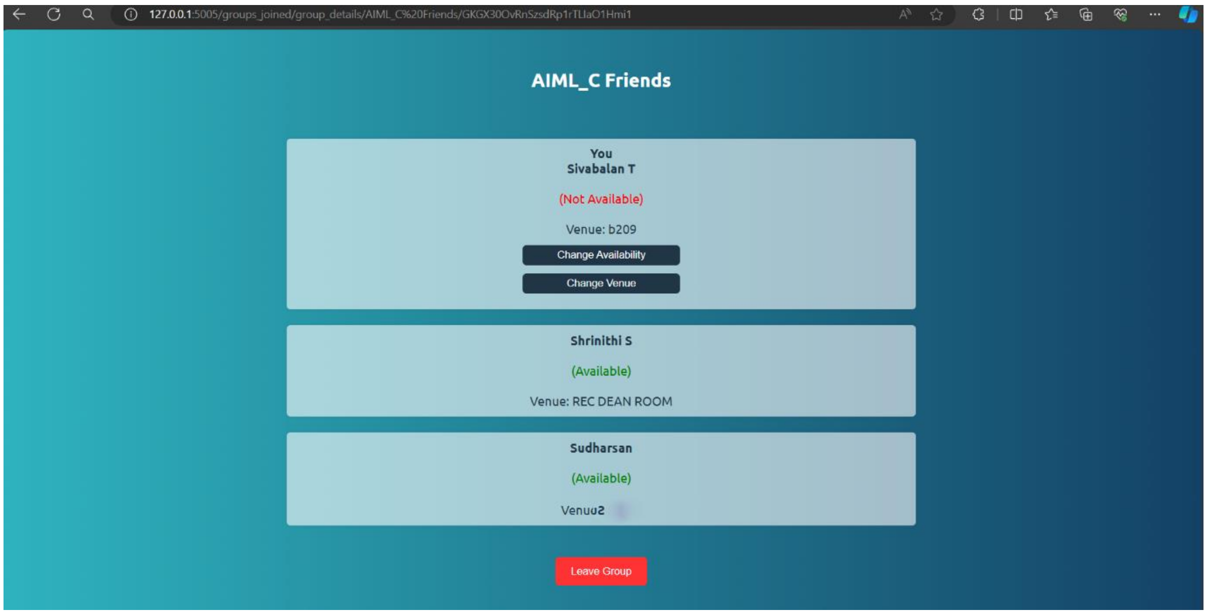
Join



127.0.0.1:5005/group_joined/GKGX30OvRnSzsdRp1rTLlaO1Hmi1

Your Groups

AIML_C Friends
(3 members)



127.0.0.1:5005/groups_joined/group_details/AIML_C%20Friends/GKGX30OvRnSzsdRp1rTLlaO1Hmi1

AIML_C Friends

You
Sivabalan T

(Not Available)

Venue: b209

Shrinithi S

(Available)

Venue: REC DEAN ROOM

Sudharsan

(Available)

Venuo2

CHAPTER 6

CONCLUSION

The Utrackk system has streamlined the process of tracking staff availability in educational institutions, enhancing the efficiency of managing schedules, group activities, and student interactions. Real-time data processing ensures accurate staff availability, aiding in better planning and coordination. Administrative functions effectively manage operations, while robust security measures protect sensitive data. Utrackk has successfully met its objectives, showcasing the transformative potential of modern technology in educational management.

The system’s modular architecture supports future enhancements, such as advanced analytics, mobile app integration, and additional collaboration tools, further improving user experience and operational efficiency.

REFERENCES

[1] Smith, J., & Johnson, A. "Enhancing Educational Management: The Role of Technology Integration." Educational Technology Research Journal, 25[3], 45-58.

[2] Brown, L., & Wilson, C. "Improving Institutional Communication: Insights from a Web-based Complaint Box Implementation." Journal of Educational Administration, 35[2], 112-125.

[3] Patel, R., & Jones, M. "Secure Data Handling Practices in Educational Settings: A Case Study of Student Information Systems." Journal of Information Security, 15[4], 267-280.

[4] Gonzalez, E., & Martinez, D. "Efficiency and Effectiveness in Educational Management: Lessons Learned from Implementing a Staff Availability Tracking System." International Journal of Educational Administration, 22[1], 89-102.

[5] White, S., & Lee, K. "User Experience Design Principles for Educational Management Systems." Journal of Human-Computer Interaction, 12[3], 176-189.PHP and MySQL And PHP: A Beginner's Guide: 3rd Edition , 2012 And 2008