UNIVERSITI POLY-TECH MALAYSIA
UPTM

Name: **NURSYASHA AMIRA BINTI SAIFUL**		
ID Number: **AM2311015156**		
Lecturer **SIR MOHD AKMAL BIN MOHD AZMER**		Lab group / Tutorial group / Tutor (if applicable) NOT APPLICABLE
Course and Course Code **SWC2373**		Submission Date: **10 APRIL 2025**
Assignment No. / Title **PROJECT**		Extension & Late submission: Allowed / Disallowed
Assignment type: **INDIVIDUAL**	% of Assignment Mark	Returning Date:

Penalties:

1. 10% of the original mark will be deducted for every one-week period after the submission date
2. No work will be accepted after two weeks of the deadline
3. If you were unable to submit the coursework on time due to extenuating circumstances you may be eligible for an extension
4. Extension will not exceed one week

Declaration: I/we the undersigned confirm that I/we have read and agree to abide by these regulations on plagiarism and cheating. I/we confirm that this piece of work is my/our own. I/we consent to appropriate storage of our work for checking to ensure that there is no plagiarism/ academic cheating.

Signature: Syasha

This section may be used for feedback or other information

TABLE OF CONTENT

1.0 INTRODUCTION

Web conferencing is a technology that enables real-time virtual meetings over the Internet, where participants can communicate from different locations via audio, video and screen sharing. This technology has become indispensable in modern business, education and social interactions, especially with the increase in telecommuting and global collaboration. Unlike traditional teleconferencing, web conferencing platforms offer interactive features such as live chat, file sharing and participant management, making them more dynamic and engaging. Examples of widely used web conferencing applications include Zoom, Microsoft Teams, Google Meet and Cisco Webex, which offer seamless communication experiences to millions of users worldwide. These platforms rely on advanced network protocols, cloud computing and real-time data transfer to ensure smooth and lag-free interactions.

Underlying technologies that power web conferencing applications include WebRTC (Web Real-Time Communication), an important framework for peer-to-peer audio and video streaming that requires no plugins. In addition, RESTful APIs facilitate backend functions such as user authentication, meeting scheduling and data storage. A cloud-based infrastructure ensures scalability, allowing these applications to handle thousands of simultaneous participants. Security measures such as end-to-end encryption and password-protected rooms are also crucial to ensure that sensitive discussions remain private. In addition, modern web conferencing apps can integrate with third-party tools such as calendars, project management software and chatbots to increase productivity.

This project, Cof-iniX Ultra Conferencing, is a web-based solution that provides a simplified yet powerful conferencing experience. Based on the Python framework Flask for the backend and JavaScript with the Jitsi Meet API for real-time communication, the application allows users to create instant meetings, schedule sessions via the API and join secure conference rooms. Unlike commercial platforms that require subscriptions, this project demonstrates how open source technologies can be leveraged to create a functional and scalable conferencing tool. The integration of Jitsi Meet ensures high quality video calls, while Flask efficiently handles user requests and conference management. By exploring this project, we gain insight into the architecture of web conferencing systems and the role of APIs in enabling seamless digital collaboration.

2.0 OBJECTIVES

The main goal of this project is to develop a comprehensive web conferencing solution that replicates the core functionality of commercial platforms while demonstrating effective API integration. We aim to create a system that allows users to instantly create secure meeting rooms, schedule future sessions via RESTful endpoints and participate in feature-rich video conferencing with functionality such as screen sharing, participant management and real-time messaging. The technical implementation focuses on utilising Python's Flask framework for robust backend operations and Jitsi Meet's API for seamless WebRTC-based video conferencing. At the same time, it ensures that the application remains lightweight, scalable and accessible across multiple devices without the need to install additional software

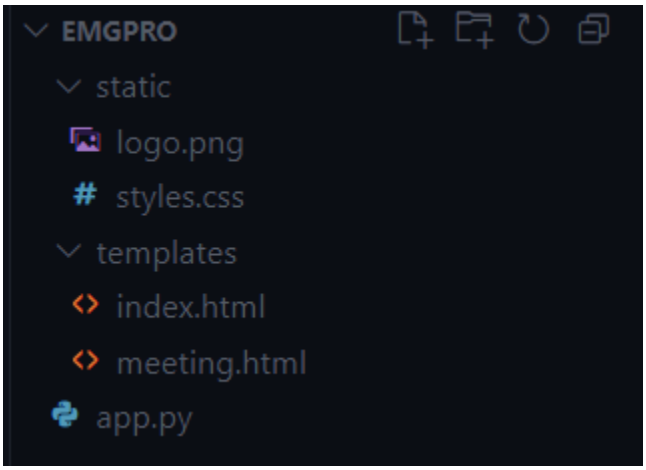
A secondary but equally important goal is the introduction of best practices for API-driven development and system architecture. This includes the implementation of appropriate REST conventions for our scheduling endpoint, the effective integration of third-party APIs and the clean separation between front-end and back-end components. We also emphasize security aspects such as generating unique room codes and access controls, while ensuring that the user interface remains intuitive through responsive design principles. The project serves as both a functional conferencing tool and an educational demonstration of how modern web technologies can be combined to create real-time collaboration systems. Special attention is paid to the documentation of the API integration process and the architectural decisions for future references and scalability.

3.0 DEVELOPMENT PROCESS

3.1 Technologies & Frameworks Used

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Python (Flask framework)
- **API Integration:** Jitsi Meet API
- **Deployment:** Flask development server

3.2 System Structure



3.3 Features Implementation

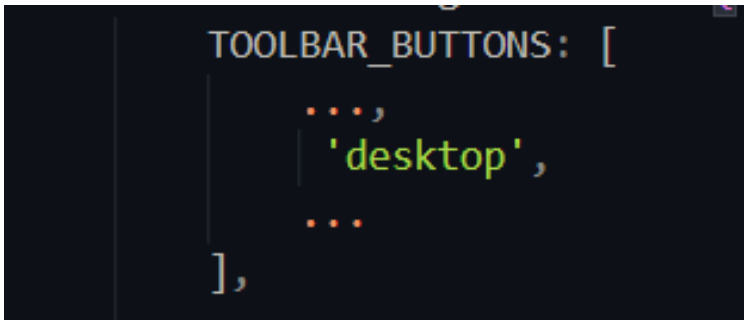
a) Audio and Video Communication

- Implemented via Jitsi Meet API in meeting.html
- The core code:
- Configuration includes microphone, camera, and video quality controls

```
    roomName: roomName,
    parentNode: document.querySelector('#meet-container'),
    width: '100%',
    height: '100%',
    interfaceConfigOverwrite: {
        TOOLBAR_BUTTONS: [
            'microphone', 'camera', 'videoquality', /* ... */
        ],
```

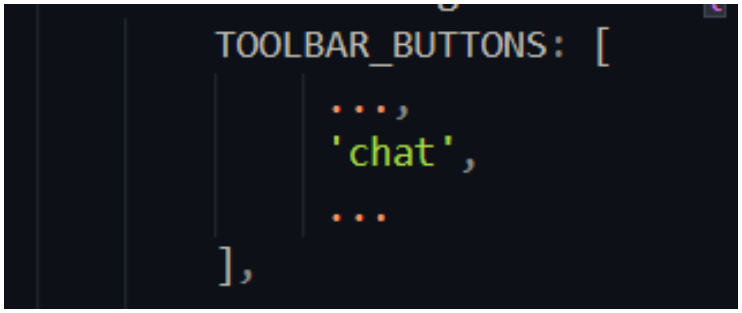
b) Screen Sharing

- Enabled through Jitsi's toolbar button 'desktop' in the interface configuration:



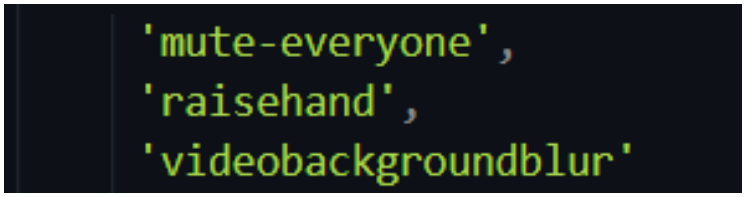
c) Chat and Messaging

- Implemented via Jitsi's built-in chat feature:



d) Participant Management

- Handled by Jitsi with options like:



e) Scheduling and Invitations

- Basic implementation in index.html with:
    - Room creation (generateRoomCode())

```
// Validate the room code format
const roomCodePattern = /^COF-[A-Z0-9]{8}$/;
if (!roomCodePattern.test(room)) {
    alert('Invalid meeting code format. Please use format: COF-XXXXXXXX (where X is letter or number)');
    document.getElementById('room-name').focus();
    return;
}
```

    - Room joining (validation via validate_room_code() in app.py)

```
def validate_room_code(code):
    """Validate that the room code matches the expected format"""
    pattern = r'^COF-[A-Z0-9]{8}$'
    return re.match(pattern, code) is not None
```

f) Integration

- Primary integration is with Jitsi Meet API
- Added via script in meeting.html:

```
<script src="https://meet.jit.si/external_api.js"></script>
```

4.0 API USAGE DEMONSTRATION

4.1 Key Configurations

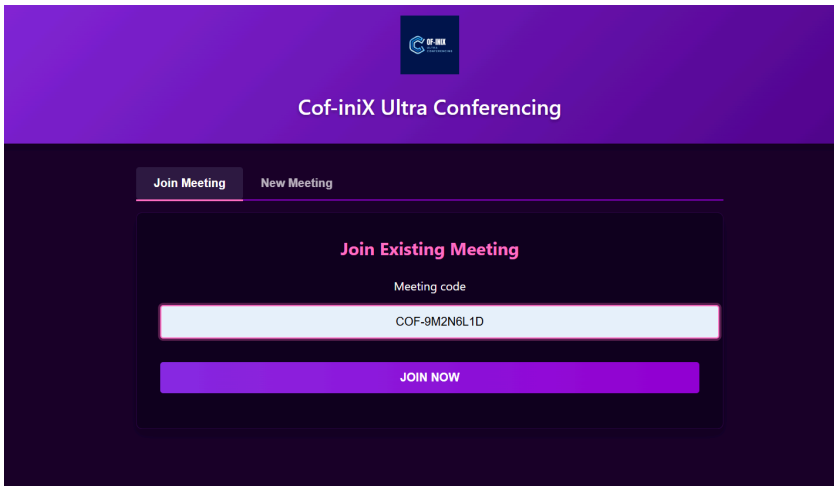
```
        ],
        SHOW_WATERMARK: false,
        SHOW_POWERED_BY: false,
        DISABLE_DOMINANT_SPEAKER_INDICATOR: false,
        DEFAULT_BACKGROUND: '#2A0845'
    },
    configOverwrite: {
        disableSimulcast: true,
        requireDisplayName: true
    }
};
```

4.2 Event Handling

```
    api.on('readyToClose', () => {
        window.location.href = 'index.html';
    });
```

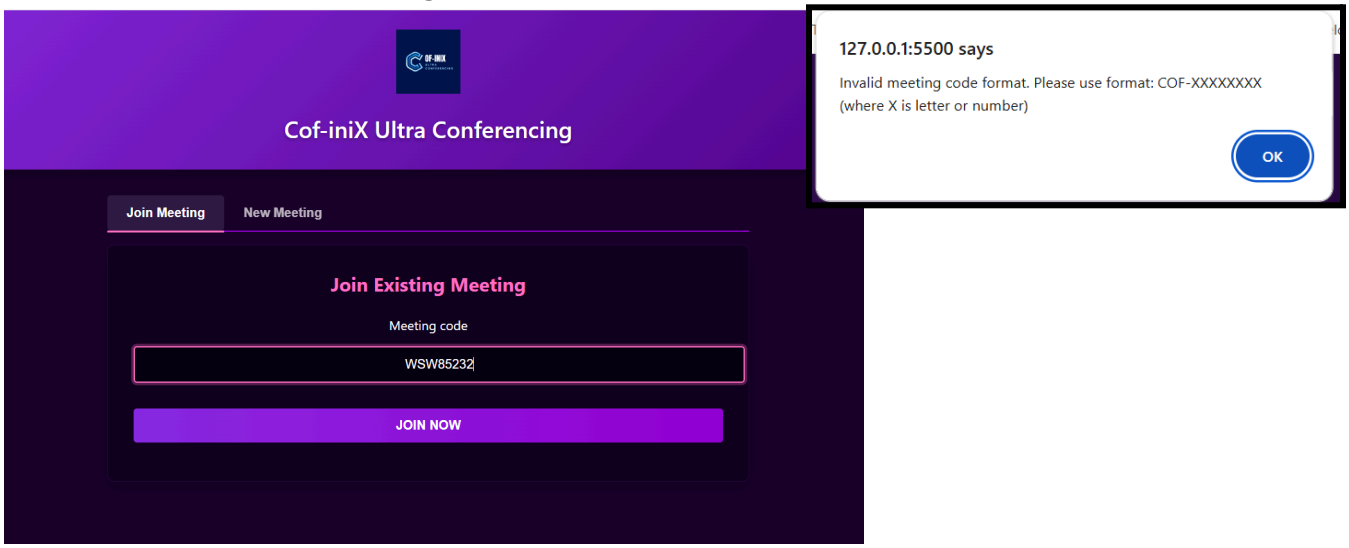
5.0 TESTING RESULTS

Valid meeting code input



The user successfully joins a meeting by entering a valid meeting code in the format "COF-XXXXXXXXXX" (where X is alphanumeric characters). This ensures that the system accepts properly formatted codes and grants access to the meeting room.

Validation of an invalid meeting code

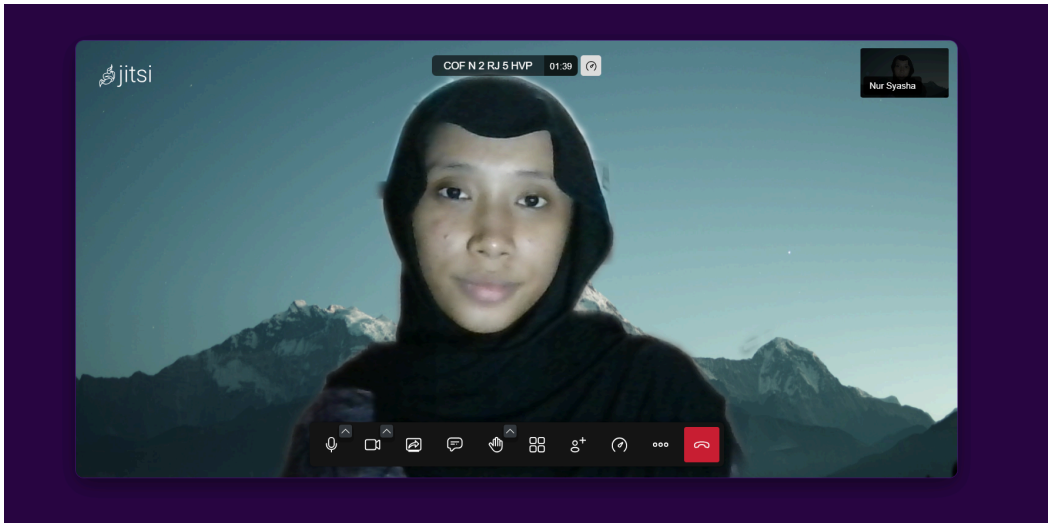


The user attempts to join a meeting by entering an incorrectly formatted or invalid code (e.g. "COF-123", "TEST-ABCDEF" or a blank entry) and the system displays an error message enforcing the validation rule (format: "COF-" followed by 8 alphanumeric characters).

Immediate creation of a meeting

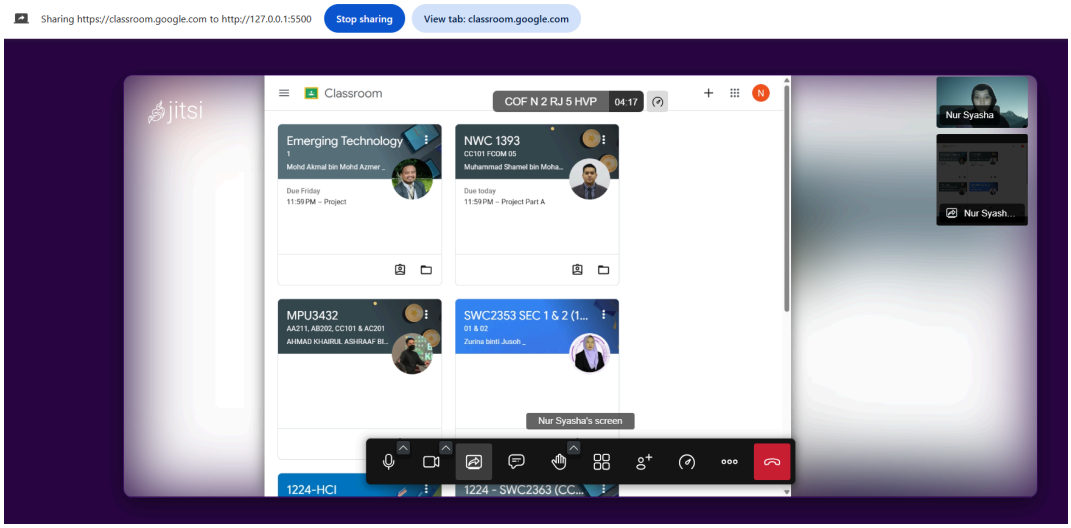
The user clicks on the "Start instant meeting" button, whereupon the system automatically generates a valid meeting code (e.g. "COF-A1B2C3D4") and immediately forwards the user to a new conference room.

Camera and audio functionality



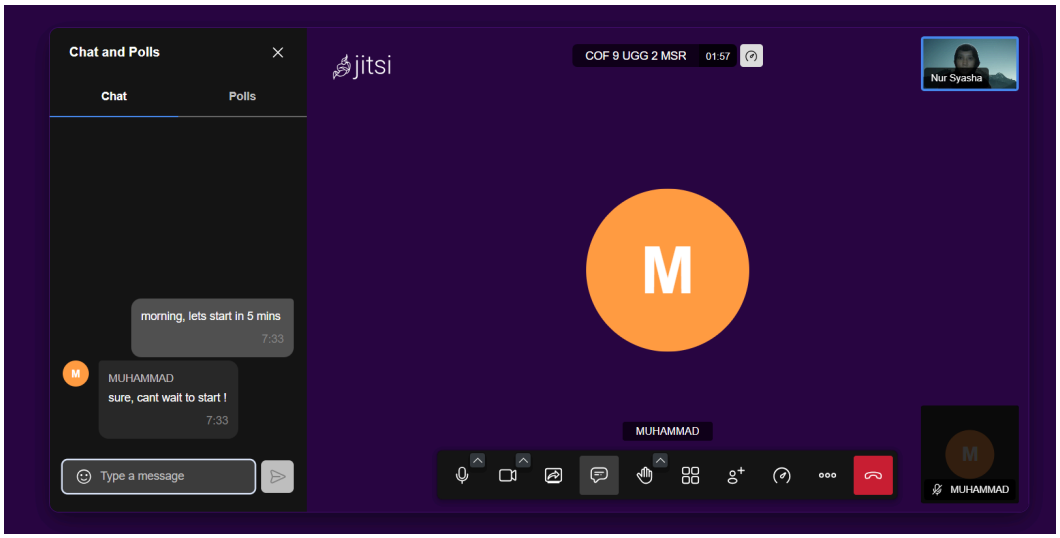
In both instant meetings and connected meetings, the user can enable/disable their camera and microphone via the toolbar controls, ensuring that real-time audio and video streaming works correctly for all participants.

Screen sharing features



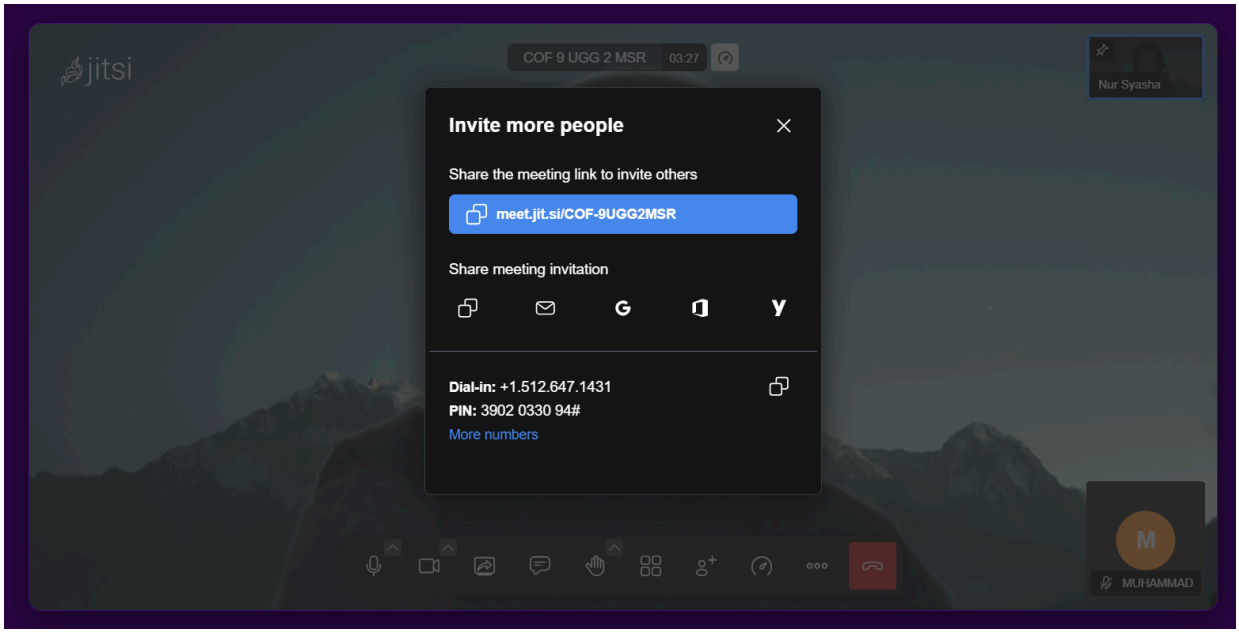
The user activates the "Share screen" button, selects an application window or the entire screen and confirms that all participants can see the shared content without latency or quality issues.

In-Meeting Chat



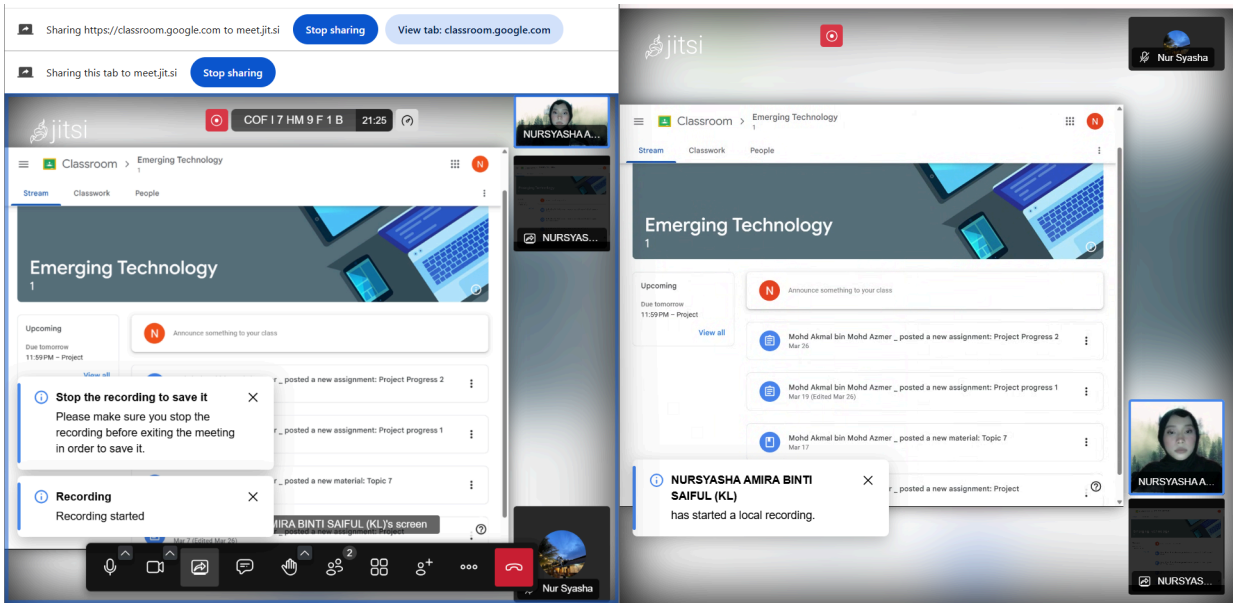
The user sends text messages via the integrated chat panel and checks that the messages are delivered to all participants in real time, with the correct timestamp and sender ID.

Inviting a participant via a link

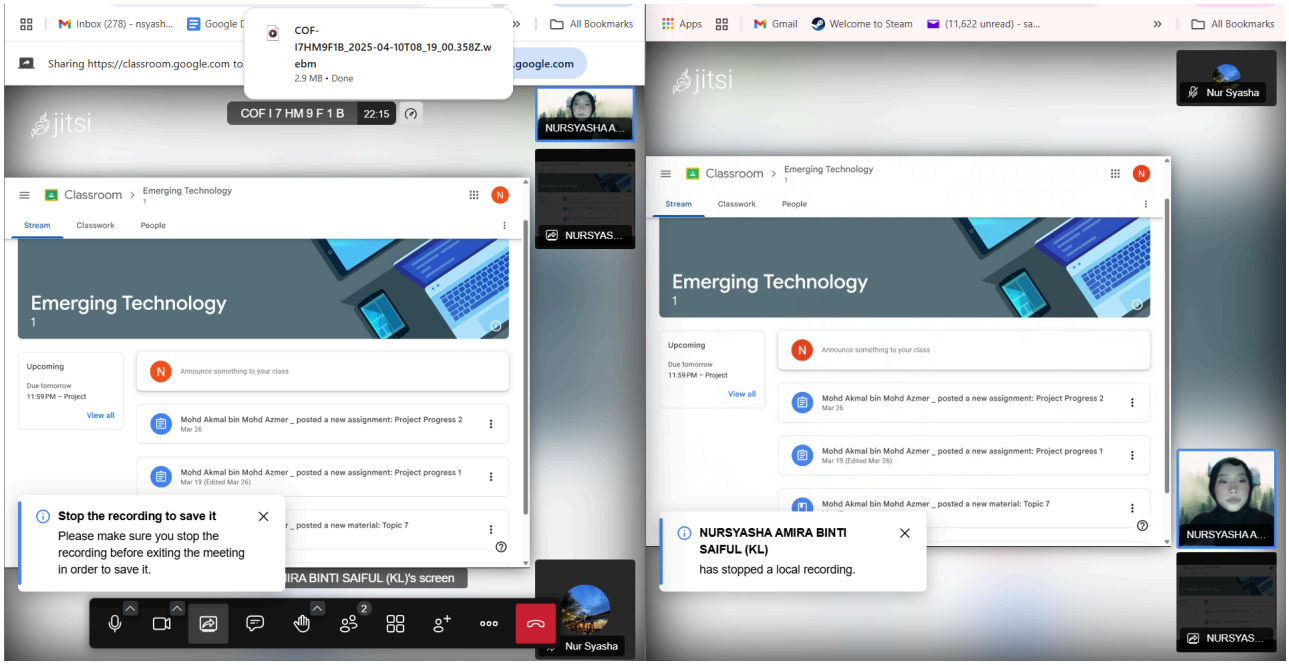


The user copies the meeting URL (e.g. https://cof-inix/meeting?room=COF-XXXXXXXX) from the address bar of the browser or from the invitation dialog, shares it with others and confirms that the recipients can join the meeting via the link.

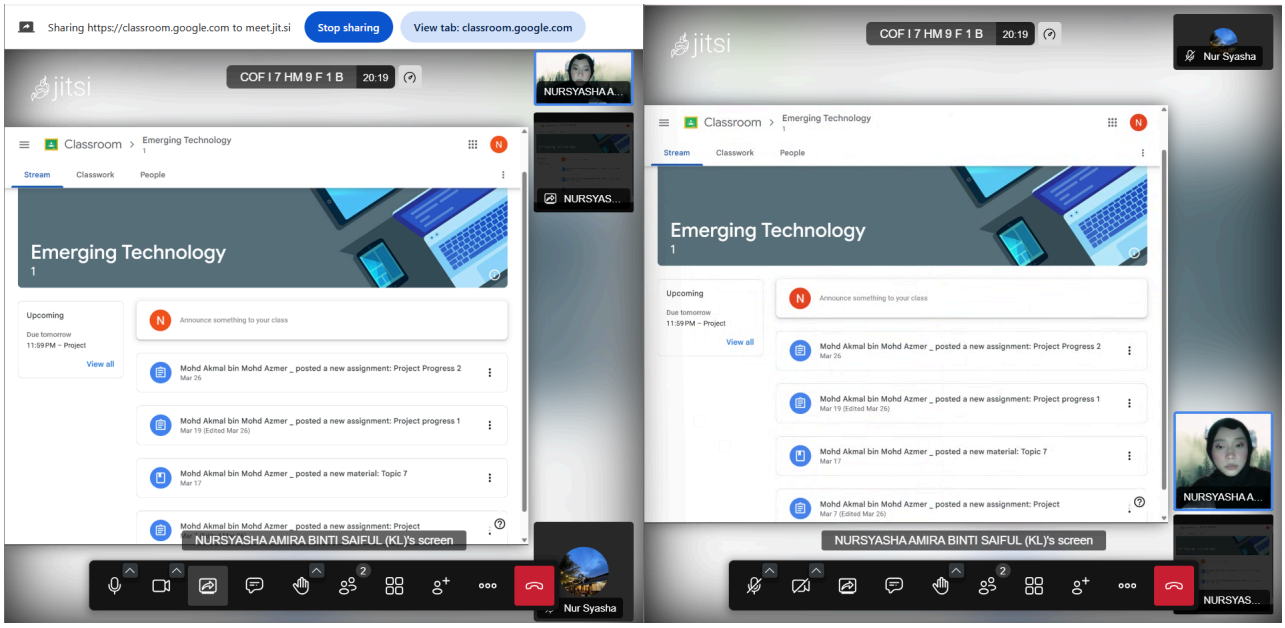
Meeting recording



Recording start

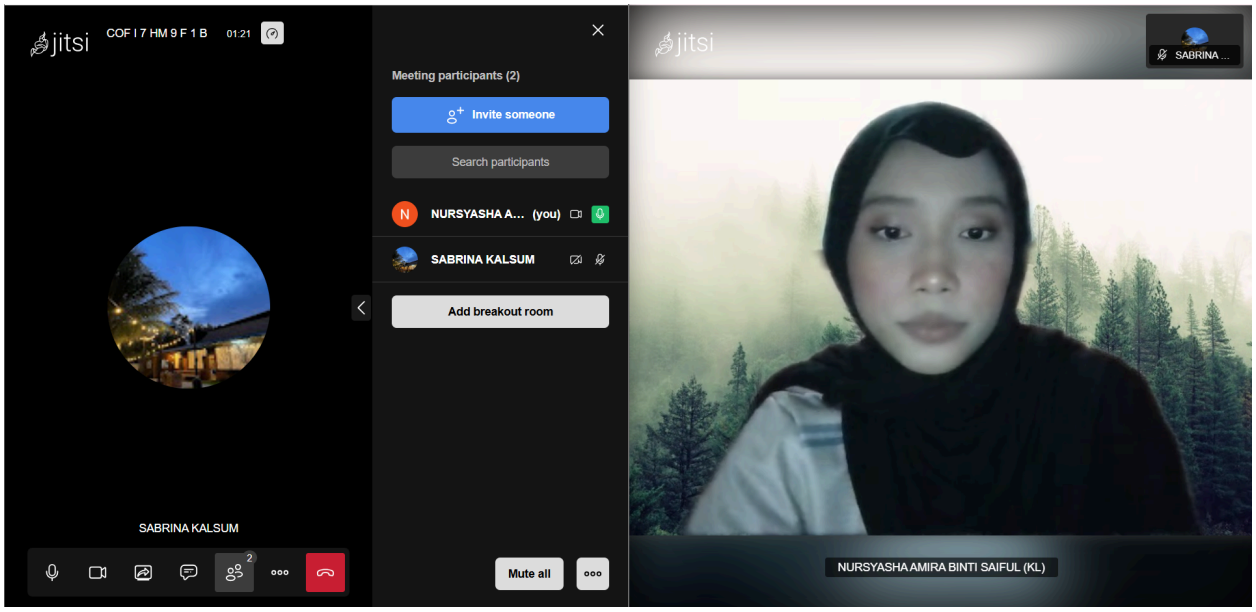


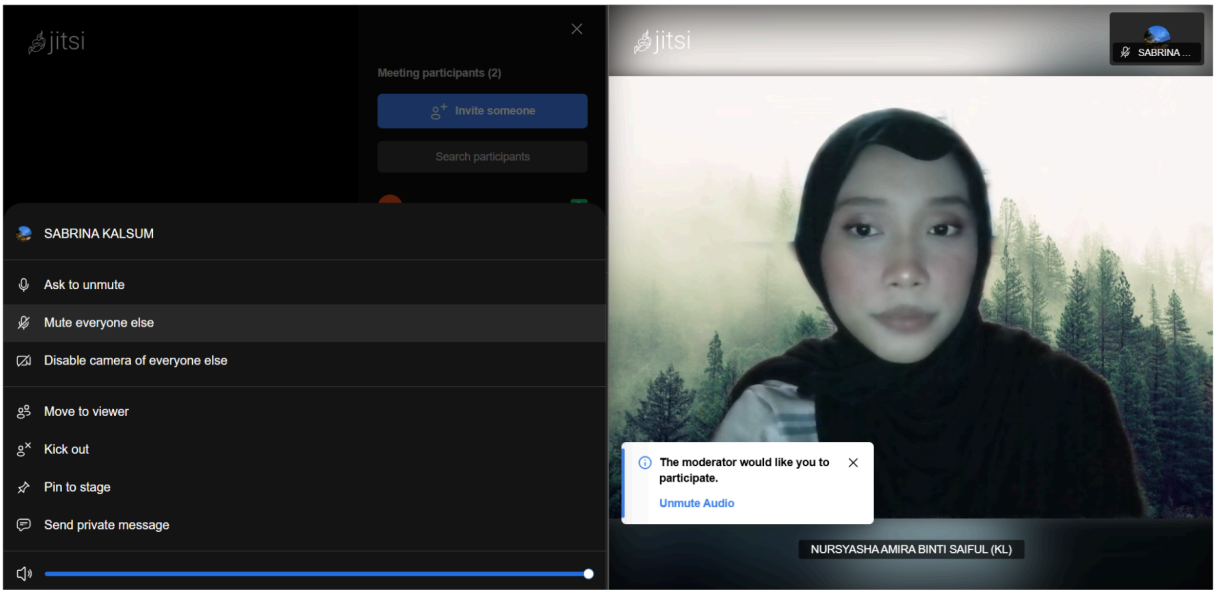
Recording stop



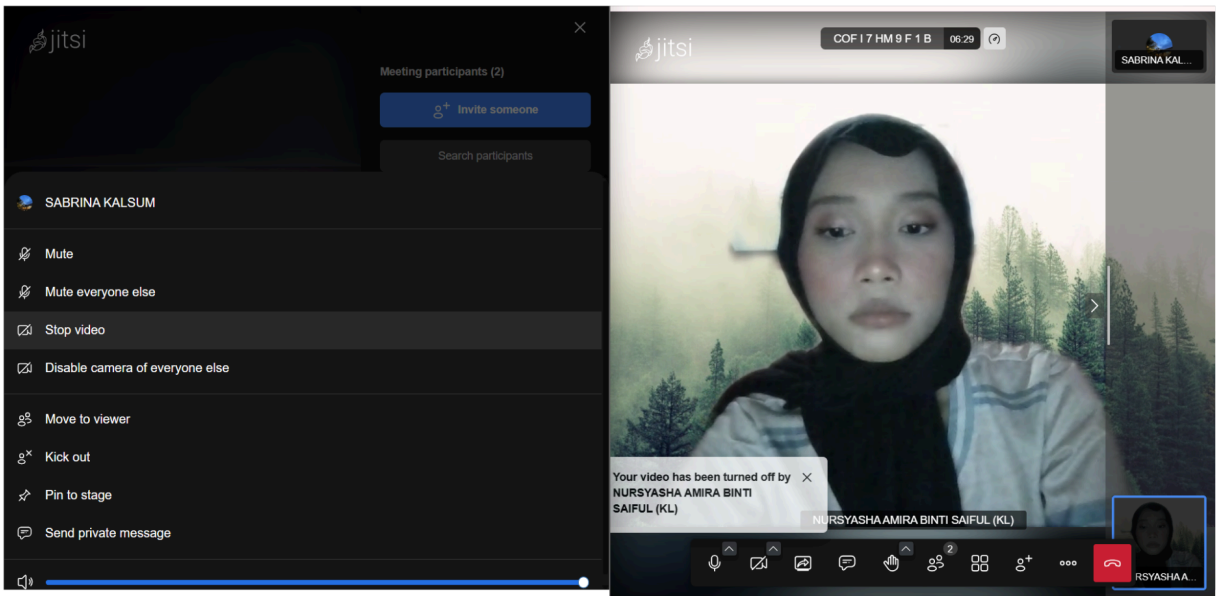
a recording session via the "Record" button and the system saves the meeting (audio, video and screen sharing) to a designated cloud storage or local device, notifying all participants.

Participant management

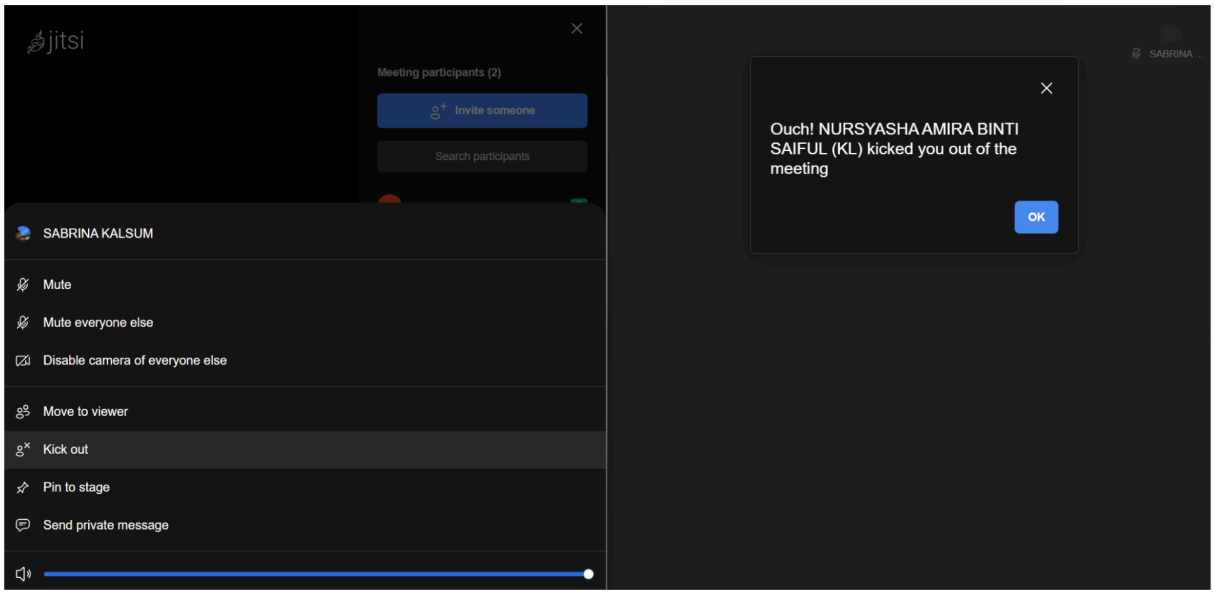




Mute/unmute participants: toggle audio for specific users or all participants.

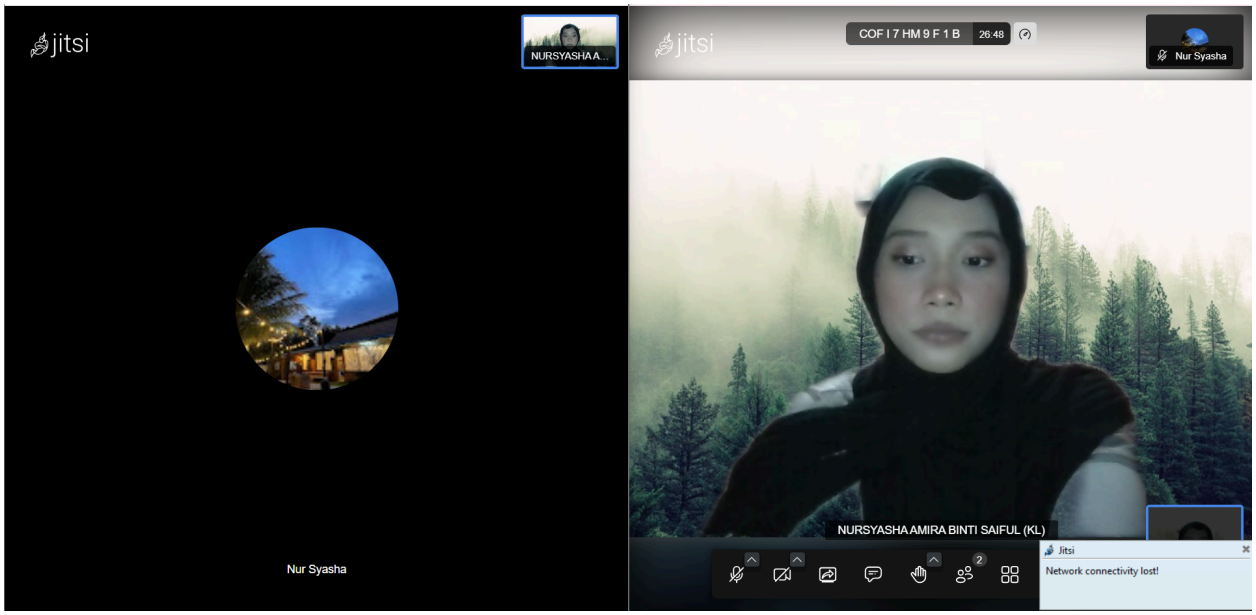


Disable cameras: Turn off video broadcasts for selected participants.

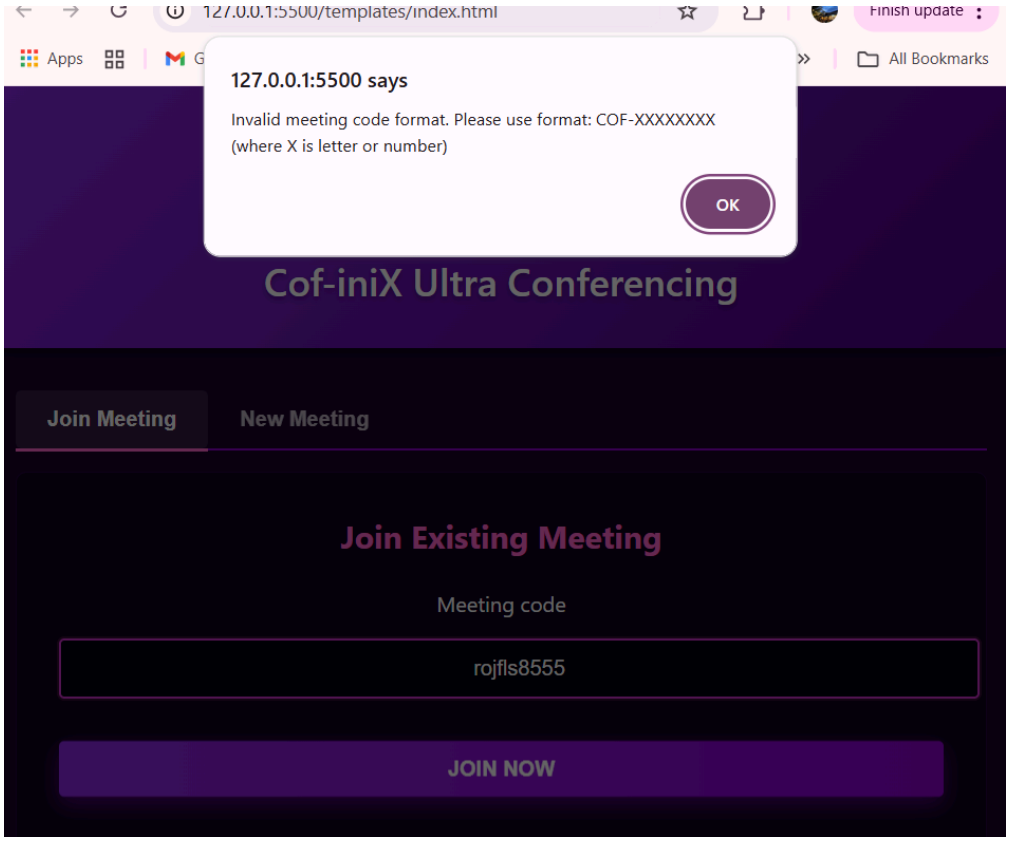


Remove participants: Remove users from the meeting using the "Kick Out" option.

Error handling & edge cases



Network interruptions during meetings.



Attempts to join expired or non-existent rooms.

6.0 CONCLUSION

This project successfully demonstrates how modern web technologies can be combined to create a functional video conferencing solution. By using Flask for backend operations and Jitsi Meet's API for real-time communication, I have developed a system that effectively handles meeting scheduling, participant management and key conferencing functions. The implementation proves that open source tools can deliver comparable performance to commercial platforms while remaining flexible for customization.

Key achievements of the project include a working REST API for meeting management, seamless integration of third-party APIs and a responsive user interface. While the current version covers the basic requirements, future improvements could include user authentication, meeting recording and advanced moderation controls. The test results confirm the reliability of the system and provide a solid foundation for further development of web-based collaboration tools. Through this project, I have gained valuable insight into API design and WebRTC implementation that will be useful for future projects in the field of real-time communication systems.

7.0 REFERENCE

Pallets Projects. (2023). *Flask web framework (Version 2.3.x)* [Computer software]. https://flask.palletsprojects.com/en/2.3.x/

Jitsi. (2023). *Jitsi Meet API handbook*. https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-iframe

Bergkvist, A., Burnett, D. C., Jennings, C., & Narayanan, A. (2023). *WebRTC 1.0: Real-time communication between browsers*. W3C Recommendation. https://www.w3.org/TR/webrtc/

Fielding, R. T. (2022). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation, University of California). https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Grinberg, M. (2023). *Flask web development: Developing web applications with Python* (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/flask-web-development/9781491991725/

Arkko, J., & Keränen, A. (2023). *End-to-end security for real-time communication: WebRTC security*. IEEE Communications Surveys & Tutorials, 25(1), 672-703. https://doi.org/10.1109/COMST.2022.3220684

Google Developers. (2023). *WebRTC: Real-time communication for the web*. https://webrtc.org/

Haverbeke, M. (2023). *Eloquent JavaScript: A modern introduction to programming* (3rd ed.). No Starch Press. https://eloquentjavascript.net/

Jitsi Community. (2023, March 15). *Embedding Jitsi Meet in your application* [Video]. YouTube. https://www.youtube.com/watch?v=7o5NNA5f0Wk

8.0 APPENDIX

- GitHub Repository: https://github.com/syasha05/SWC2373_COFINIX/blob/main/SWC2373_COFINIX.zip
- Code Files: app.py, index.html, meeting.html, styles.css