MeetingMind AI: An AI-Powered System for Meeting Management, Transcription and Intelligent Analysis

Ramhari Sangale[1], Kartik Sangale[2], Payal Bhute[3], Ashwini Shidore[4]
[1,2,3,4]*Department of Artificial Intelligence and Data Science, Shri Chhatrapati Shivaji Maharaj College of Engineering, Nepti, Ahilyanagar, Affiliated to Savitribai Phule Pune University, Maharashtra, India*

***Abstract*—Meetings are central to how teams collaborate, yet the important details that come out of them, such as decisions, tasks, due dates and open issues, are often forgotten, and writing up minute's means listening back through long recordings. Commercial AI note-takers help with this, but they are usually cloud-based, sold on subscription, and require sending confidential audio to a third party. In this paper, we present MeetingMind AI, a web application that manages meetings, records them, transcribes speech and analyzes the content with AI.**

Our system includes user registration, meeting creation through a shareable link, WebRTC-based audio and video conferencing with chat over a FastAPI/WebSocket signaling server, and recording directly in the browser. Uploaded recordings are prepared with FFmpeg, transcribed with the Whisper speech-to-text model, and analyzed by a locally hosted language model (Qwen2.5 1.5B, served through Ollama). The model produces a summary, key takeaways, action items, decisions, deadlines, questions and announcements. Results are stored in a SQLite database and can be reopened through a Past Meetings module or queried through a chatbot. The application is publicly available online under a custom domain over HTTPS.

We surveyed related work and grouped existing solutions into manual note-taking, conferencing tools, cloud AI note-takers, transcription services and research summarization models. Their shortcomings, including fragmented workflows, privacy and cost concerns, and no conversational access to past meetings, motivated us to build a system that brings all the necessary components together while remaining open source. In this paper, we describe the architecture of our solution, its current implementation status, and our plans for verifying the deployed application.

***Index Terms*—Meeting Intelligence, WebRTC, Speech Recognition, Whisper, Large Language Models, Ollama, Meeting Summarization, Action Item Extraction, FastAPI, Chatbot.**

I. INTRODUCTION

Meetings are where team members share ideas, make decisions and assign work to one another. A typical hour-long meeting involves a lot of questions, decisions, task assignments and deadlines. Much of what is said is spoken rather than written down, so it is easy to forget. Meeting minutes are usually produced by replaying recordings by hand, which takes time and gives inconsistent results.

Recent advances in automatic speech recognition (ASR) and large language models (LLMs) make it possible to automate this work. ASR models such as Whisper can transcribe accented speech, noisy audio and multiple languages without task-specific fine-tuning [1]. Compact LLMs such as Qwen2.5 can run locally and pull structured information out of text [2]. Together, they make it possible to build a pipeline that transcribes a recording and turns it into well-organized notes.

Existing tools only solve parts of the problem. Conferencing platforms handle communication and recording but offer little analysis, or lock analysis behind a paid cloud service. Note-taking products require uploading recordings to third-party servers, which can be a problem for confidential meetings. Research systems for meeting summarization are evaluated on benchmark corpora [6][7] but are rarely released as ready-to-use applications.

A team that wants an end-to-end workflow therefore has to stitch together several products and accept their costs and privacy trade-offs. This work presents MeetingMind AI, a web-based system that covers the whole life cycle of a meeting.

The contributions of this paper are:

- An integrated approach that combines WebRTC meetings, browser recording, Whisper transcription and LLM analysis in a single software solution.
- A structured analysis (summary, highlights, action items, decisions, deadlines, questions and announcements) produced by a locally hosted model accessed through Ollama.
- A Past Meetings module and a chatbot that give chat-based access to information from previous meetings.
- An online deployment (FastAPI signaling backend, custom domain, HTTPS), together with an honest account of the project's current state and what still needs to be verified.

The rest of the paper is organized as follows. Section II covers existing approaches, Section III reviews the literature, Section IV presents a comparative analysis, Section V identifies gaps, Section VI outlines the methodology, Section VII describes the proposed approach and its current status, and Section VIII concludes the paper.

II. EXISTING SYSTEMS

Traditionally, meetings have been transcribed by hand. One participant is asked to record decisions and action items during the meeting, or the whole session is recorded and reviewed afterwards. This needs no technology, but it distracts the note-taker and takes roughly as long as the meeting itself.

Video-conferencing tools such as Zoom, Microsoft Teams and Google Meet provide audio, video, chat and recording, and some also offer AI assistants that transcribe and summarize meetings. These services are usually delivered through cloud-based subscription models, so the recordings and transcriptions are stored on the provider's infrastructure, and the quality of the output depends on that provider.

Several specialised AI services can transcribe meetings and produce transcriptions, summaries and action items. They reduce the effort people have to put in, but they sit outside the conferencing tool, usually cost a recurring subscription, and require the meeting audio to leave the organisation.

Academic work on meeting understanding has concentrated on summarisation and question answering over transcripts, using corpora such as AMI [6] and QMSum [7]. These papers improve the models themselves, but they do not go as far as building an end-to-end solution that hosts meetings, captures their audio and keeps the data for future use [8]. The typical architecture of existing solutions is shown in Fig. 1.

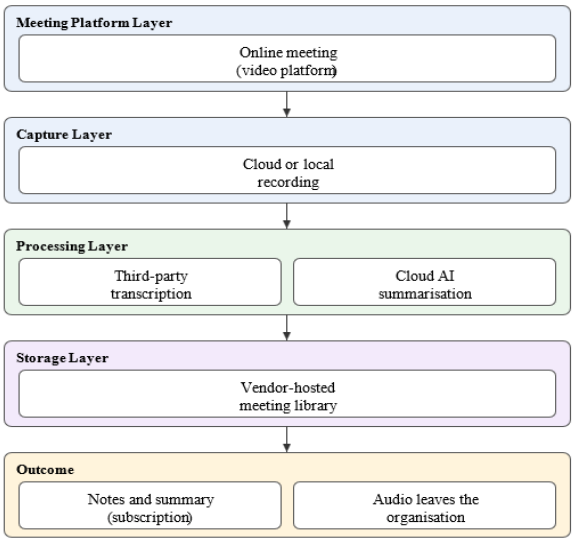

Fig. 1: Existing Meeting-Assistant Architecture

In this architecture, each stage is handled by a separate component, sometimes from different vendors. As a result, information crosses organisational boundaries at the processing layer, and the user receives only the final notes, with no control over how they were produced.

III. LITERATURE REVIEW

Meeting intelligence draws on speech recognition, language modelling, real-time communication and information retrieval. This section covers the works most relevant to the design of MeetingMind AI.

A. Robust Speech Recognition via Large-Scale Weak Supervision [1]

Authors: Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey and Ilya Sutskever Whisper is a Transformer-based encoder–decoder model trained on roughly 680,000 hours of multilingual and multitask supervised data collected from the web. The authors show that this diversity allows robust transcription without fine-tuning on any particular dataset. MeetingMind AI uses Whisper as its transcription engine. Accuracy, however, still depends on audio quality, language and overlapping speech, all of which are common in real meetings.

B. Qwen2.5 Technical Report [2]

Authors: A Yang et al. (Qwen Team)

The Qwen2.5 family consists of open-weight language models ranging from 0.5 to 72 billion parameters, with instruction-tuned variants that follow structured prompts. The 1.5B variant runs on modest hardware, which is why it was chosen for local analysis through Ollama. Its drawback is that it lacks the reasoning and long-context abilities of larger models, so long transcripts may need to be split.

C. Mistral 7B [3]

Authors: Albert Q. Jiang et al.

Mistral 7B uses grouped-query attention and sliding-window attention, which give strong performance at 7 billion parameters with efficient inference. It is a natural next step for improving analysis capability, provided the hardware is available, since it needs more memory and compute.

D. Retrieval-Augmented Generation for Knowledge-IntensiveNLP Tasks [4]

Authors: Patrick Lewis et al.

Retrieval-augmented generation pairs a parametric language model with a non-parametric memory that is consulted at query time. This grounds answers in retrieved documents and allows knowledge to be updated without retraining. The idea is directly relevant to the MeetingMind AI chatbot, which has to answer from stored meeting records rather than from general model knowledge.

E. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks [5]

Authors: Nils Reimers and Iryna Gurevych

Sentence-BERT adapts BERT with siamese networks to produce sentence embeddings that can be compared by cosine similarity, which makes semantic search over large numbers of sentences practical. Embeddings of transcript segments would let the chatbot retrieve relevant passages from long meetings, and this is planned as an extension.

F. The AMI Meeting Corpus [6]

Authors: Jean Carletta et al.

The AMI corpus provides about 100 hours of recorded meetings with synchronised audio and video and rich annotations, making it a standard resource in meeting research. It is a useful benchmark for evaluating transcription and summarisation, although many of its meetings are scenario-driven and differ from everyday workplace discussions.

G. QMSum: Query-based Multi-domain Meeting Summarization [7]

Authors: Ming Zhong, Da Yin, Tao Yu et al.

QMSum introduces query-based meeting summarisation: given a transcript and a query, a system must summarise the relevant part of the meeting. The benchmark contains 1,808 query–summary pairs over 232 meetings from several domains. It shows that long transcripts and specific user questions are hard to handle, which motivates combining structured notes with a question-answering chatbot.

H. A Survey on Dialogue Summarization [8]

Authors: Xiachong Feng, Xiaocheng Feng and Bing Qin

This survey organises research on dialogue summarisation and discusses the challenges specific to spoken dialogue: multiple speakers, topic shifts, informal language, and errors carried forward from speech recognition.

These challenges apply directly to MeetingMind AI, where Whisper errors flow straight into the language model's analysis.

I. WebRTC Requirements and the WebSocket Protocol [9][10]

Authors: C. Holmberg, S. Hakansson and G. Eriksson; I. Fette and A. Melnikov

RFC 7478 describes use cases and requirements for browser-to-browser real-time communication without plug-ins, while RFC 6455 defines WebSocket, a persistent two-way channel over a single TCP connection. WebRTC does not specify how peers find each other, so a signalling channel is needed. In MeetingMind AI, the FastAPI/WebSocket backend fills this role and also carries chat messages.

IV. COMPARATIVE ANALYSIS

Manual note-taking is flexible and needs no special infrastructure, but it is slow, and the quality of the notes depends heavily on who is taking them. Conferencing platforms keep live communication and recording in one place, yet someone still has to review the recording unless a paid AI add-on is available.

Cloud AI note-takers produce quick summaries and action items, and they are easy to adopt. The downsides are the subscription cost, the need to send confidential audio to external servers, and the fact that they sit apart from the tool where the meeting actually happens. Transcription-only services give accurate, searchable text, but they leave the reader to work out the decisions, owners and deadlines.

Research summarisation models achieve strong benchmark results [6][7], but they are rarely packaged for end users and are limited by input length. A locally hosted pipeline built from open components, as in MeetingMind AI, gives control over data and cost, but it depends on the capability of a small model and on the compute available [1][2]. A comparative summary is given in Table 1.

Table 1: Comparative Analysis of Meeting Documentation Approaches

Approach	*Advantages*	*Limitations*
Manual note-taking	No technology needed; flexible	Slow; inconsistent; distracts the note-taker
Platform + recording	Live meeting and recording together	Little or no analysis; replay needed
Cloud AI note-takers	Fast summaries and action items	Subscription; data leaves the organisation; separate tool
Transcription -only	Accurate, searchable text	No decisions or tasks extracted
Research models	Strong benchmark results	Not packaged as applications; input-length limits
MeetingMind AI (proposed)	Integrated; local analysis; chatbot	Small-model limits; full testing pending

V. GAP ANALYSIS

Although strong components already exist, several gaps remain. The first is fragmentation: users hold meetings in one tool, record in another and analyse in a third, which adds manual steps and loses context [1][8]. The second is privacy and control. The most capable services process recordings on external servers, which is unsuitable for confidential discussions and gives users little say over retention. The third is cost, since subscription pricing keeps students, small teams and institutions from adopting these tools.

The fourth gap is structure. Transcripts are long, and users need decisions, owners and deadlines presented in a consistent format [7]. The fifth is access to history: after a meeting, people want to ask questions of past meetings rather than reread their notes [4]. Fig. 2 compares tool categories by how many of eight workflow stages they cover: live meeting, recording, transcription, structured summary, action-item and decision extraction, meeting history, conversational question answering, and a self-hosted option.

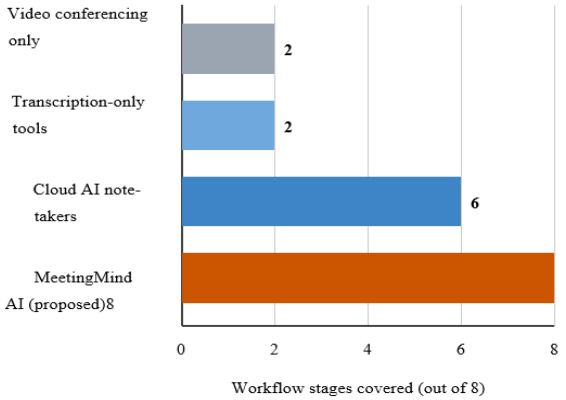

Fig. 2: Workflow Stage Coverage by Tool Category (indicative)

The counts in Fig. 2 are the authors' qualitative assessment of typical feature sets described in Section II, not a benchmark.

Locally hosted open models' close part of these gaps but bring problems of their own. Small models are less accurate and slower on CPU-only servers, long meetings can exceed context limits, and Whisper's accuracy drops when speakers overlap [1][2]. A practical system has to balance accuracy, cost, privacy and speed, and it has to be tested end to end in the environment where it will actually run.

What is needed, then, is an integrated platform that combines live meetings, recording, transcription, structured analysis, stored history and conversational access, built on open components.

VI. METHODOLOGY

MeetingMind AI follows a pipeline in which each meeting passes through the stages below. The processing pipeline is illustrated in Fig. 3.

A. Account Management

Users register and log in, and their accounts and meeting records are kept in SQLite. Meetings are retrieved per user (for example, through a *get_user_meetings()* function), so each user sees only the meetings that belong to them.

B. Meeting Creation and Joining

A host creates a meeting, and the system generates a meeting ID or room link that can be shared. Participants open the link to enter the meeting room.

C. Real-Time Communication

Audio and video are exchanged between browsers using WebRTC.

A FastAPI signalling server, communicating over WebSockets, helps participants establish their connections and relays chat messages [9][10]. The room offers camera and microphone controls, chat and a leave option.

D. Recording

The room can start and stop a recording, save it as a WebM file and upload it to the backend for processing. In the current implementation, the recording captures the local microphone/camera stream rather than a mix of all participants, so whether a combined recording is produced still needs to be verified.

E. Preprocessing and Transcription

FFmpeg prepares or converts the uploaded media, and Whisper converts speech to text [1]. Transcript quality is expressed as word error rate:

$$WER = (S + D + I) / N \quad (1)$$

where S, D and I are the numbers of substituted, deleted and inserted words, and N is the number of words in the reference transcript.

F. Structured Analysis

The transcript is passed to the Qwen2.5 1.5B model running under Ollama [2], with instructions to return seven sections: summary, key points, action items, decisions, deadlines, questions and announcements. For example, if a team says Ram will design the homepage, Kartik will build the login page and the work must be finished by Friday, the system should list two action items with their owners and the Friday deadline.

G. Storage, Past Meetings and Chatbot

The analysis and related meeting information are saved in the database so that users can reopen a previous meeting. The chatbot answers questions such as “What decisions were made?” or “What tasks were assigned to Ram?” by using the stored information as context, in the spirit of retrieval-augmented generation [4]. Embedding-based retrieval [5] is a planned extension. Here’s how the pipeline works. A meeting recording in WebM format goes first through FFmpeg, which prepares the audio. Whisper then turns that audio into a transcript. Next, the transcript is combined with analysis instructions to build a prompt, and a locally hosted language model, Qwen2.5 1.5 B running through Ollama, reads it and produces structured meeting notes. Those notes cover a summary, key points, action items, decisions, deadlines, open questions, and announcements. Finally, everything is saved to a SQLite database, where it can be revisited through a Past Meetings view or queried with a meeting chatbot

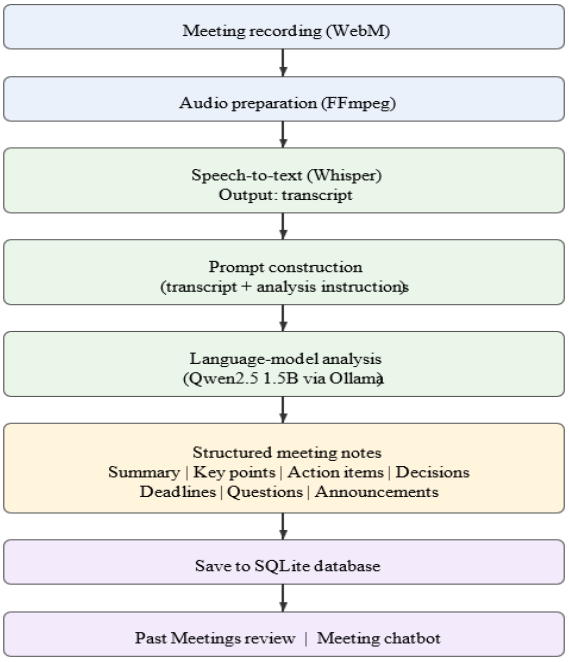

Fig. 3: Meeting Processing Pipeline

H. Evaluation Plan

For evaluation, we plan to run end-to-end tests on recorded test meetings, each paired with manually written reference notes, and supplement them with public corpora such as AMI. Action items, decisions, and deadlines will be scored using precision, recall, and F, calculated from true positives, false positives,

and 1 false negative. Summaries will be compared against reference summaries using ROUGE-N, which measures the overlap of n-grams with the reference, and the results will be checked with human judgement as well.

$$P = TP / (TP + FP),\ R = TP / (TP + FN),\ F = 2PR / (P + R) \quad (2)$$

$$ROUGE\text{-}N = (overlapping\ n\text{-}grams) / (n\text{-}grams\ in\ reference) \quad (3)$$

VII. PROPOSED SYSTEM

The system is organised in layers so that each part can be built, tested, and swapped out on its own. The presentation layer holds the Streamlit interface and the browser-based meeting room. The communication layer handles WebRTC audio and video, a FastAPI signalling service, and WebSocket chat. The media layer records meetings and prepares the audio. The AI processing layer runs Whisper and the Ollama-hosted model. The storage layer keeps users, meetings, transcripts, and analysis in SQLite, and the service layer exposes Past Meetings review and the chatbot to the user.

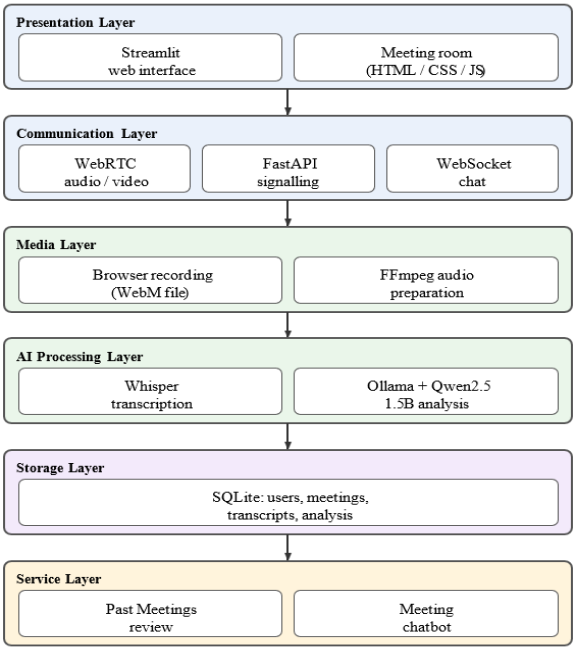

Fig. 4: Proposed MeetingMind AI System Architecture

A. Technologies Used

The main technologies are Python for the core application and AI integration, Streamlit for the web interface, FastAPI and WebRTC/WebSocket for signalling and real-time communication, Whisper for speech-to-text, Ollama with Qwen2.5 1.5B for local analysis, SQLite for storage, FFmpeg for media conversion, and HTML, CSS, and JavaScript for the meeting room. The application is hosted on Render under the domain meetingmindai.site with HTTPS enabled.

Table 2: Technologies and Their Roles

Technology	*Role in the system*
Python	Main application and AI integration
Streamlit	Web application interface
FastAPI	Backend and signalling service
WebRTC / WebSocket	Browser audio/video; real-time signalling and chat
Whisper	Speech-to-text transcription
Ollama, Qwen2.5 1.5B	Local language-model analysis of transcripts
SQLite	Users, meetings and analysis storage
FFmpeg	Audio/video conversion and preparation
HTML, CSS, JavaScript	Meeting room interface
Render, custom domain	Online hosting; meetingmindai.site with HTTPS

B. Implementation Status

Several components are working: registration and login, meeting creation with room links, local recording, and AI analysis, which has generated output successfully. The site is online and the signalling endpoint reports its status correctly. Other parts still need checking in the deployed environment. Cross-network audio and video, multi-device chat, multi-participant recording, Whisper and Ollama on the server, per-user access controls, and chatbot accuracy across meetings all remain to be tested.

A site that loads only proves users can reach it. Uploads, AI processing, multi-user sessions, and data persistence each need separate verification, since features that run on a development machine can fail on a server because of configuration, network limits, storage constraints, or missing AI services.

Table 3: Implementation Status of Components

Component	*Status*
Registration and login	Confirmed working
Meeting creation and room link	Implemented
Audio / video (WebRTC)	Implemented; cross-network testing pending
Meeting chat	Implemented; multi-device testing pending
Recording (WebM)	Local save verified; multi-participant capture to be checked
Whisper transcription	Integrated; deployed-environment check pending
Ollama analysis	Generated successfully earlier; deployed check pending
Past Meetings	Implemented; per-user access checks pending
Chatbot	Implemented; validation across meetings pending
Online deployment	Site online; signalling endpoint reports online

C. Security and Privacy

Because recordings and transcripts are sensitive, several safeguards must be confirmed before wider use: secure password storage, access limited to each user's own meetings, protected API keys and configuration, secure storage of recordings, a recording-consent process, deletion and retention rules, and backups. Running the language model locally means transcripts never leave the system for an external AI provider, which was a main reason for this design.

VIII. CONCLUSION

MeetingMind AI brings these steps together into one system that turns online meetings into organised, reusable information. It combines WebRTC meetings and chat, recording, Whisper transcription, local LLM analysis, a stored meeting history, and a question-answering chatbot, and it is deployed online. A review of existing tools found that they tend to be fragmented, often depend on the cloud, and seldom allow conversational access to past meetings, which is the gap this design aims to fill.

The work is not yet fully validated. Multi-device meetings, deployed AI processing, chatbot accuracy, and security controls still need systematic testing, and a 1.5B-parameter model limits the quality of analysis. Future work includes a TURN fallback for difficult networks, a waiting room and participant list, screen sharing, downloadable transcripts and reports, searchable history, embedding-based retrieval for the chatbot, speaker identification, larger models where hardware allows, and load testing for concurrent meetings, uploads, and storage.

REFERENCES

[1] Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint* arXiv:2212.04356, 2022.

[2] A. Yang *et al*., "Qwen2.5 technical report," *arXiv preprint* arXiv:2412.15115, 2024.

[3] A. Q. Jiang *et al*., "Mistral 7B," *arXiv preprint* arXiv:2310.06825, 2023.

[4] P. Lewis *et al*., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[5] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods in Natural Language Processing and Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[6] J. Carletta *et al*., "The AMI meeting corpus: A pre-announcement," in *Proc. Machine Learning for Multimodal Interaction (MLMI)*, 2005.

[7] M. Zhong *et al*., "QMSum: A new benchmark for query-based multi-domain meeting summarization," in *Proc. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2021.

[8] X. Feng, X. Feng, and B. Qin, "A survey on dialogue summarization: Recent advances and new frontiers," in *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI), Survey Track*, 2022.

[9] C. Holmberg, S. Hakansson, and G. Eriksson, "Web Real-Time Communication use cases and requirements," IETF RFC 7478, 2015.

[10] I. Fette and A. Melnikov, "The WebSocket protocol," IETF RFC 6455, 2011.