



**Arab International University**

**Faculty of Informatics and Communication Engineering**

**Applied Project Report on**

**Intelligent Meeting Analytics: Real-Time Insight  
Extraction from Microsoft Teams Conversations Using  
Transformer-Based NLP and WebSocket Streaming**

Submitted to

Department of Informatics Engineering

in partial fulfillment of the requirement for the Applied project course

Submitted by

**Ammar Alzoubi**

**April 2026**

© AIU Arab International University

All Rights Reserved

**© AIU Arab International University**

**All Rights Reserved**

**Month\_name Year**

Faculty of Informatics & Communication Engineering

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the Department of Informatics Engineering for acceptance, a project report entitled **Intelligent Meeting Analytics: Real-Time Insight Extraction from Microsoft Teams Conversations Using Transformer-Based NLP and WebSocket Streaming**

Submitted by: **Ammar Alzoubi** in partial fulfilment for the degree of Bachelor of Engineering in Informatics.

# Arab International University

The Arab International University (AIU) is a private Syrian university established in 2005. Its academic plans and the documents issued by it are approved and certified by the Ministry of Higher Education in the Syrian Arab Republic.

**The university works to achieve the following goals:**

- Preparing a distinguished generation of university graduates who are able to meet and advance the specific needs of society.
- Contributing to theoretical and applied scientific research that serves the purposes of national development. Work is being done to urge professors and academic staff to scientific research and participate in conferences and seminars that organize research.
- Achieving partnership with prestigious Arab and foreign universities with the aim of continuous development and modernization of academic work and conducting joint scientific research.
- Attracting distinguished academic and research competencies by providing the appropriate environment for their work.

**The Arab International University** is one of the first Syrian universities that have been established and inaugurated. It has been able to attract distinguished educational, research and administrative competencies, to create an integrated edifice from the academic, organizational and administrative aspects. It was able to graduate cadres of distinguished innovators by providing an educational environment based on unique qualitative and material ingredients, including:

- Modern and advanced study plans based on the credit hour system.
- Carefully selected educational cadres.
- Modern scientific laboratories and a laboratory for electronic libraries.
- Physical and moral incentives for students.
- Application of interactive teaching methods.
- Academic and educational guidance and counseling.
- A wide range of scientific cooperation agreements with local, regional and international universities of reputable reputation.
- Multiple agreements and memoranda of understanding with many civil society organizations.
- A proper campus with all the facilities of science, sports and entertainment, which we encourage you to visit and learn about their features.
- Student activities and clubs of all kinds: athletic, cultural, scientific and social.

**The Arab International University** life years are a time to invest in a student's future. The knowledge and experience that students acquire in the lecture hall and laboratories will help them in developing themselves. They will provide them with reasons for success in the chosen specialty. The student activities will help in expanding students' horizon. The activities of training, clubs and sports will enable students to develop their talents, and may even help in discovering new talents.

Students could invest time, mind and spirit in our university in order to reap the benefits of works and the time devoted in the coming years. We will be by our students in every step of their way.

# Abstract

The proliferation of virtual collaboration platforms has fundamentally altered how professional teams communicate, coordinate, and make decisions. Despite this shift, a persistent challenge remains: valuable insights generated during meetings—including action items, key decisions, open questions, and discussion threads—are routinely lost or require significant post-meeting effort to retrieve and document. This project presents MeetingMind, an AI-powered real-time meeting analytics extension for Microsoft Teams that addresses this gap by automatically extracting structured intelligence from live audio streams.

The system integrates a speech-to-text pipeline based on OpenAI Whisper for audio transcription, followed by a suite of fine-tuned transformer models from HuggingFace for downstream NLP tasks including extractive and abstractive summarization, action item detection with assignee and deadline recognition, intent classification, and topic segmentation. A Python FastAPI backend coordinates real-time data flow using WebSocket connections, ensuring low-latency processing and delivery of insights to a React-based dashboard. The extension is deployed through the Microsoft Teams Toolkit, enabling seamless integration with the existing collaboration environment.

Evaluation of the system against benchmark datasets demonstrates strong performance: the summarization module achieves a ROUGE-2 score of 0.41 and ROUGE-L of 0.61, while the action item extraction pipeline achieves an F1-score of 0.78 on a curated meeting transcript dataset. Topic segmentation using a fine-tuned BERT model reaches 84.6% segmentation accuracy. The post-meeting dashboard, tested under simulated meeting loads of up to 90 minutes, processes and renders all extracted insights within an acceptable latency threshold. Overall, MeetingMind demonstrates that combining real-time audio processing with transformer-based NLP can substantially reduce the cognitive overhead of meeting documentation and improve team productivity.

**Keywords:** Natural Language Processing, Transformer Models, Microsoft Teams, WebSocket, Real-Time Systems, Meeting Summarization, Action Item Extraction, HuggingFace, OpenAI Whisper, FastAPI, BERT, Topic Segmentation

# Contents

Abstract .....	5
Chapter 1: Introduction .....	8
1.1 Motivation and Significance .....	8
1.2 Scope of the Project.....	9
1.3 Structure of the Report .....	9
Chapter 2: Problem Statement and Objectives.....	10
2.1 Problem Statement .....	10
2.2 Project Objectives .....	10
2.3 Project Scope.....	11
2.4 Project Features .....	11
Chapter 3: Theoretical Study.....	13
3.1 Transformer-Based Language Models .....	13
3.2 Automatic Speech Recognition with Whisper .....	13
3.3 WebSocket Communication.....	13
3.4 FastAPI and Async Python .....	14
3.5 Microsoft Teams Toolkit and Extensibility .....	14
Chapter 4: Related Work.....	15
4.1 Literature Review .....	15
4.2 Comparison of Related Studies .....	17
Chapter 5: System Analysis .....	18
5.1 Requirements Specification.....	18
5.2 Functional Requirements.....	18
5.3 Non-Functional Requirements .....	18
5.4 Use Case Overview .....	19
Chapter 6: System Design .....	20
6.1 Overall System Architecture .....	20
6.2 Audio Pipeline.....	21
6.3 NLP Processing Pipeline.....	21
6.4 Dashboard Design .....	22
6.5 Technology Stack Summary .....	22
Chapter 7: Implementation.....	23
7.1 Development Environment .....	23
7.2 Teams Extension Implementation.....	23
7.3 Backend Implementation.....	24

7.4 NLP Model Fine-Tuning .....	24
7.5 Topic Segmentation Implementation .....	25
Chapter 8: Results and Discussion .....	26
8.1 Summarization Results.....	26
8.2 Action Item Extraction Results .....	26
8.3 Topic Segmentation Results.....	27
8.4 End-to-End Latency .....	27
Chapter 9: Conclusion and Future Work.....	28
9.1 Conclusion.....	28
9.2 Future Work .....	28
References .....	30

## List of Figures

Figure 1 UML Use Case Diagram for Meeting Mind Actors and Features. ....	19
Figure 2 UML Block Diagram of the Overall System Structure. ....	21
Figure 3 System Topology and Technology Stack Distribution. ....	23
Figure 4 UML Sequence Diagram for Real-Time Audio and Result Processing. ....	24

## List of Tables

Table 1: Comparison of Related Studies in Meeting Intelligence .....	17
Table 2: Technology Stack.....	22
Table 3: Summarization Performance on AMI Test Set .....	26
Table 4: Action Item Extraction Results .....	26

# Chapter 1: Introduction

The modern workplace has undergone a remarkable transformation over the last several years, accelerated primarily by the global adoption of remote and hybrid work models. Platforms such as Microsoft Teams, Zoom, and Google Meet have become central to organizational communication, with millions of meetings taking place daily across industries. Yet, despite this tremendous volume of collaborative interaction, the mechanisms for capturing and leveraging the knowledge generated during these meetings remain surprisingly underdeveloped.

Consider a typical project meeting: a team discusses progress, identifies blockers, assigns responsibilities, and agrees on next steps. At the conclusion of the meeting, much of this information exists only in the memories of participants or in rough personal notes. Follow-up actions may be unclear or unattributed, decisions may be misremembered, and context may be lost entirely for team members who were absent. The conventional response—having a designated note-taker or requiring participants to summarize discussions manually—is both inefficient and unreliable.

Advances in Natural Language Processing (NLP) and automatic speech recognition (ASR) over the past few years have opened the door to a fundamentally different approach. Transformer-based language models, in particular, have demonstrated remarkable capabilities in tasks such as text summarization, named entity recognition, question answering, and intent detection. Combined with improvements in ASR accuracy and the availability of real-time communication APIs, it has become technically feasible to build systems that can listen to a meeting, understand its content, and extract actionable intelligence—all in real time.

This project, MeetingMind, is a practical realization of that vision. It is implemented as a Microsoft Teams extension that activates during live meetings, captures audio, transcribes speech, and applies a pipeline of NLP models to extract four primary categories of insight: action items (tasks assigned to specific individuals, including deadlines and urgency levels), a structured meeting summary (covering key decisions, open questions, raised concerns, and follow-up items), a meeting timeline (segmenting the discussion by topic and speaker), and a post-meeting dashboard that visualizes all extracted data in a usable format.

## 1.1 Motivation and Significance

The productivity cost of unstructured meetings is substantial. Studies in organizational behavior consistently find that professionals spend a significant portion of their working hours in



meetings, yet a large percentage of meeting time is considered unproductive. A key contributor to this inefficiency is the lack of effective tooling for meeting intelligence: existing transcription tools provide raw text but offer no structure; summary tools, when available, operate post-meeting and require manual invocation. MeetingMind targets this gap by providing automated, real-time, structured intelligence directly within the collaboration platform.

## **1.2 Scope of the Project**

The system focuses on English-language meetings conducted through Microsoft Teams. Audio is processed in near-real-time using a chunked streaming approach. The NLP pipeline operates on transcribed text segments and aggregates results across the meeting duration. The frontend dashboard is a web-based React application embedded as a Teams tab. The project does not include integration with task management platforms (such as Jira or Asana) or calendar systems in its current form, though these are identified as clear directions for future work.

## **1.3 Structure of the Report**

This report is organized into seven chapters. Chapter 2 provides a theoretical background on the core technologies employed, including transformer architectures, WebSocket communication, and speech recognition systems. Chapter 3 reviews related work in the fields of automatic meeting summarization, action item detection, and real-time NLP systems, and includes a structured comparison of prior approaches. Chapter 4 presents the system analysis, covering functional and non-functional requirements. Chapter 5 describes the system design, including the overall architecture and component interactions. Chapter 6 details the implementation of each system module. Chapter 7 presents results, testing, and evaluation. The report concludes with a discussion of findings and directions for future development.

# Chapter 2: Problem Statement and Objectives

## 2.1 Problem Statement

Despite the ubiquity of online meeting platforms in modern professional environments, no integrated, real-time solution currently exists within Microsoft Teams that can automatically transform spoken meeting content into structured, actionable data. Existing transcription features within Teams provide raw text output without any semantic processing. Third-party meeting summary tools—such as Otter.ai or Fireflies—operate as standalone applications requiring separate logins, data sharing agreements, and post-meeting processing cycles, thereby introducing friction and latency that reduces their practical utility during active work sessions.

More critically, current tools lack the ability to identify and assign action items in real time, segment discussions into meaningful topics, or present meeting intelligence through an integrated, context-aware dashboard. The absence of such capabilities forces teams to rely on manual note-taking, post-meeting reviews, and informal follow-up communications—all of which are error-prone and time-consuming processes. For organizations running large numbers of meetings per week, the cumulative cost in time, missed commitments, and lost context is significant.

The technical challenge is compounded by the real-time constraint: processing audio, transcribing speech, running NLP models, and delivering insights must all occur within latency bounds that allow users to act on information while the meeting is still in progress. This requires careful architectural decisions regarding model selection, inference optimization, and data pipeline design.

## 2.2 Project Objectives

The primary objectives of this project are:

- To design and implement a real-time audio capture and transcription pipeline using OpenAI Whisper integrated with the Microsoft Teams platform via WebSocket communication.
- To develop a multi-task NLP processing pipeline capable of performing meeting summarization, action item extraction with role and deadline attribution, meeting intent classification, and topic segmentation.

- To build a React-based post-meeting dashboard that presents extracted insights in a clear, organized, and actionable format, accessible directly within the Teams environment.
- To evaluate the system's NLP components against established benchmark metrics (ROUGE, F1) and assess the end-to-end latency under realistic meeting conditions.
- To demonstrate the feasibility and practical value of integrating transformer-based NLP with real-time collaborative platforms for meeting intelligence.

## 2.3 Project Scope

The project encompasses the full development lifecycle of the MeetingMind system, from requirements analysis through design, implementation, and evaluation. The scope includes: the Teams extension frontend built with React and Teams Toolkit; the Python FastAPI backend with WebSocket support; the Whisper-based ASR component; the HuggingFace model pipeline for NLP tasks; and the post-meeting dashboard interface. Excluded from scope are integrations with external task management systems, multilingual support, speaker diarization (beyond basic identification using Teams speaker metadata), and deployment to a production cloud environment.

## 2.4 Project Features

MeetingMind delivers the following core capabilities, each representing a distinct and independently valuable contribution to the meeting intelligence problem:

- **Real-Time Action Item Extraction:** Automatic identification of tasks from conversational speech, with attribution to named speakers, deadline extraction where mentioned, urgency classification (Urgent, Follow-Up, Pending, Idea), and contextual annotation.
- **Structured Meeting Summarization:** Generation of organized summaries covering key discussion points, formal decisions, raised concerns, open questions, and items requiring follow-up action.
- **Meeting Timeline Construction:** Segmentation of the meeting into topically coherent blocks, each annotated with the speaker, subject, key points, and associated action items.

- **Interactive Post-Meeting Dashboard:** A visual interface displaying task lists, participant engagement statistics, topic frequency analysis, and per-person workload summaries.

# Chapter 3: Theoretical Study

## 3.1 Transformer-Based Language Models

The transformer architecture, introduced by Vaswani et al. in 2017, represents a foundational advance in NLP [2]. Unlike recurrent neural networks, transformers process sequences in parallel using a mechanism known as self-attention, which allows each token in a sequence to attend to every other token, capturing long-range dependencies efficiently. The pre-training and fine-tuning paradigm popularized by BERT [3] and subsequent models (RoBERTa, DistilBERT, T5, BART) has enabled rapid adaptation of powerful general-purpose language representations to specific downstream tasks with relatively modest training data requirements.

For summarization, sequence-to-sequence transformer models such as BART and T5 are particularly well-suited. These models are trained to generate target summaries from source documents, and pre-trained variants fine-tuned on large corpora of (document, summary) pairs are readily available through the HuggingFace Transformers library. For classification tasks such as action item detection and intent classification, encoder-only models such as BERT and RoBERTa serve as strong baselines. The fine-tuning process adapts the pre-trained representations to task-specific label spaces using relatively small labeled datasets [4].

## 3.2 Automatic Speech Recognition with Whisper

OpenAI Whisper is a general-purpose ASR system trained on a large and diverse dataset of multilingual audio, totaling approximately 680,000 hours [5]. Whisper uses an encoder-decoder transformer architecture in which the encoder processes mel spectrogram features of the audio input and the decoder generates text output autoregressively. The model demonstrates robust performance across diverse acoustic conditions, accents, and background noise levels, making it a strong choice for meeting transcription where audio quality varies.

For real-time applications, Whisper can be applied to short audio segments in a chunked streaming fashion: audio is buffered in windows of several seconds, transcribed, and the resulting text is passed to downstream processing. While true streaming inference (token-by-token generation synchronized with speech) requires more specialized setups, the chunked approach offers a practical and performant solution for the latency requirements of a meeting analytics system.

## 3.3 WebSocket Communication

WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection [6]. Unlike the HTTP request-response model, WebSocket connections remain open for the duration of a session, allowing both the client and server to push data to the other party at any time without the overhead of repeated connection establishment. This makes WebSocket the appropriate transport mechanism for a real-time meeting analytics system, where audio data must flow continuously from the client (the Teams extension) to the server (the FastAPI backend) while NLP results must be pushed back to the client as they become available.

### **3.4 FastAPI and Async Python**

FastAPI is a modern, high-performance Python web framework built on Starlette and Pydantic, leveraging Python's async/await syntax for non-blocking I/O [7]. It is particularly well-suited for applications that must handle many concurrent connections, such as a WebSocket server processing multiple simultaneous meeting streams. FastAPI's native support for WebSocket endpoints, combined with its automatic data validation and OpenAPI documentation generation, makes it an efficient choice for the backend of MeetingMind.

### **3.5 Microsoft Teams Toolkit and Extensibility**

Microsoft Teams supports extensibility through several mechanisms, including tabs (embedded web applications), bots, and message extensions. The Teams Toolkit for Visual Studio Code simplifies the scaffolding, local debugging, and deployment of Teams applications [8]. For MeetingMind, the primary integration point is a Teams tab—a React web application hosted and served by the FastAPI backend—which appears as a panel within the Teams interface during and after meetings. The Teams JavaScript SDK provides access to the meeting context, including participant information and meeting metadata.

# Chapter 4: Related Work

## 4.1 Literature Review

Recent research has increasingly emphasized the need for structured, reproducible, and domain-aware approaches to conducting and documenting technical research, particularly in fields such as Artificial Intelligence (AI) and Software Engineering (SE), where projects inherently involve complex interactions between data, algorithms, system design, and evaluation protocols. Despite the availability of general academic writing guidelines, these approaches remain largely discipline-agnostic and often fail to capture the methodological depth required for engineering-oriented thesis development.

To address this limitation, Barhoum proposed the Structured Engineering Thesis Framework (SETF), a domain-aware and workflow-driven framework specifically designed for graduation thesis development in AI, Software Engineering, and Robotics [1]. Unlike traditional writing guides that treat documentation as a post-development activity, SETF reconceptualizes thesis development as an integrated research lifecycle, in which problem definition, literature review, methodology design, implementation, evaluation, writing, and revision are explicitly interconnected within a coherent and iterative workflow.

A key strength of SETF lies in its ability to align technical development processes with scientific reporting practices. In AI-focused research, the framework emphasizes critical methodological components such as dataset preparation, prevention of data leakage, model selection, training-validation-testing protocols, and the use of appropriate evaluation metrics including accuracy, precision, recall, and F1-score. Similarly, in Software Engineering contexts, SETF integrates architectural design, development methodologies (e.g., Agile and DevOps), system implementation, and systematic testing strategies within a unified structure. This alignment ensures that technical decisions are not only implemented but also rigorously documented, justified, and evaluated within a scientific narrative.

Furthermore, SETF introduces a comprehensive evaluation perspective that combines quantitative, qualitative, and comparative analysis, addressing common limitations observed in student projects such as weak baselines, insufficient validation, and lack of reproducibility. By explicitly linking evaluation design to earlier stages of methodology and implementation, the framework enhances the reliability and interpretability of research outcomes.

Another important contribution of SETF is its support for reproducibility and publication-oriented research. By structuring the research process as a transparent and iterative workflow, the framework facilitates consistent documentation of experimental settings, design decisions, and evaluation procedures, thereby enabling independent verification and extension of the work. This is particularly relevant in modern AI and SE research, where reproducibility has become a fundamental requirement.

Overall, SETF provides a structured and citable reference model that bridges the gap between engineering practice and academic writing, offering a systematic approach for developing high-quality, coherent, and potentially publishable graduation theses in technical disciplines [1].

Beyond the methodological framing provided by Barhoum, a substantial body of work has investigated the specific challenges of automatic meeting analysis. Mccowan et al. introduced one of the early systematic treatments of multi-party meeting processing, highlighting the distinct challenges of spontaneous conversational speech—including disfluencies, overlapping speech, and topic drift—compared to structured documents [9]. Their work established foundational benchmarks and motivated subsequent research in the field.

Considerable attention has been directed toward meeting summarization. Murray et al. proposed extractive approaches based on graph-based ranking and sentence scoring, achieving competitive performance on the AMI meeting corpus [10]. With the advent of pre-trained transformer models, abstractive approaches have gained prominence: Zhao et al. fine-tuned BART on meeting transcripts and demonstrated that transformer-based abstractive models outperform extractive baselines on coherence and informativeness metrics [11]. However, direct application of standard summarization models to meeting transcripts introduces challenges, as meeting language differs substantially from the news articles and encyclopedic text on which most models are pre-trained. Techniques such as domain-adaptive fine-tuning and the use of speaker-aware encodings have been proposed to address this gap.

The automatic extraction of action items from meeting transcripts has received increasing attention as a distinct task. Purver et al. framed action item detection as a sentence-level binary classification problem and demonstrated the utility of lexical and structural features [12]. More recent work has moved toward sequence labeling formulations that allow the simultaneous detection of action item spans, assignee mentions, and temporal expressions. Transformer-based models fine-tuned for this multi-label structured prediction task have shown significant improvements over prior feature-engineered approaches [13].



Topic segmentation, the task of dividing a meeting transcript into topically coherent segments, is addressed by methods ranging from lexical cohesion-based approaches to supervised sequence models. Eisenstein and Barzilay proposed a Bayesian model for topic segmentation that captures vocabulary shifts between segments [14]. BERT-based segmentation models have since achieved state-of-the-art performance on standard benchmarks by leveraging contextual embeddings to detect semantic discontinuities in the discourse [15].

Real-time meeting analytics systems that integrate these capabilities remain comparatively rare in the academic literature. Most published systems operate in a batch mode, processing complete meeting transcripts after the meeting has concluded. Notable exceptions include Waibel et al.'s lecture-hall transcription system and the industrial deployments underlying commercial products such as Otter.ai and Microsoft Teams' Intelligent Recap feature. However, detailed technical documentation of these commercial systems is limited, and they do not expose the structured action item and timeline intelligence that MeetingMind targets.

## 4.2 Comparison of Related Studies

Table 1 summarizes the key characteristics and limitations of representative prior works relevant to the MeetingMind system.

*Table 1: Comparison of Related Studies in Meeting Intelligence*

Study	Task	Model/Method	Dataset	Metric	Limitation
Murray et al. [10]	Summarization	Graph-based extractive	AMI Corpus	ROUGE	No real-time; extractive only
Zhao et al. [11]	Summarization	BART fine-tuned	QMSum	ROUGE-L	Batch only; no action items
Purver et al. [12]	Action Item Detection	SVM + features	ICSI Corpus	F1	No assignee/deadline; offline
Eisenstein & Barzilay [14]	Topic Segmentation	Bayesian LDA	TDT / ICSI	Pk / WD	Statistical only; no transformer
Otter.ai (Commercial)	Transcription + Summary	Proprietary ASR + LLM	N/A	N/A	No structured action items; external platform
MeetingMind (Proposed)	All of the above	Whisper + HuggingFace Transformers	AMI + Custom	ROUGE, F1, Acc.	English only; no diarization

# Chapter 5: System Analysis

## 5.1 Requirements Specification

The requirements for MeetingMind were derived through a combination of user story analysis, review of related work, and iterative refinement based on technical feasibility assessments. The system requirements are organized into functional and non-functional categories.

## 5.2 Functional Requirements

The system shall satisfy the following functional requirements:

- FR-1: The system shall capture audio from an active Microsoft Teams meeting session using the Teams JavaScript SDK and the Web Audio API.
- FR-2: The system shall transcribe captured audio in near-real-time using the Whisper ASR model, processing audio chunks of configurable duration (default: 5 seconds).
- FR-3: The system shall extract action items from transcribed text, identifying the task description, the assigned person (if mentioned), the deadline (if stated), and the urgency level.
- FR-4: The system shall generate a structured meeting summary organized into predefined categories: key points, decisions, concerns, suggestions, open questions, and follow-up items.
- FR-5: The system shall segment the meeting into topical sections and maintain a timeline view associating each segment with the speaker, topic label, and key points.
- FR-6: The system shall display all extracted insights through a web-based dashboard embedded as a Teams tab, accessible both during and after the meeting.
- FR-7: The system shall transmit audio data from the client to the server and receive NLP results from the server to the client via WebSocket connections.

## 5.3 Non-Functional Requirements

In addition to functional requirements, the system must satisfy the following non-functional constraints:

- NFR-1 (Latency): End-to-end processing latency from audio capture to dashboard update shall not exceed 10 seconds for standard meeting conditions.

- NFR-2 (Accuracy): The summarization module shall achieve a ROUGE-2 score no lower than 0.35 on the AMI meeting benchmark. The action item extraction module shall achieve an F1-score no lower than 0.70.
- NFR-3 (Usability): The dashboard interface shall require no more than three user interactions to access any category of meeting insight.
- NFR-4 (Scalability): The backend shall support at least 10 concurrent meeting sessions without degradation in processing latency exceeding 20%.
- NFR-5 (Security): All WebSocket connections shall use TLS encryption. No meeting audio shall be persisted to disk or transmitted to third-party servers.
- NFR-6 (Maintainability): The NLP pipeline shall be modular, allowing individual model components to be replaced or updated without changes to the core data flow architecture.

## 5.4 Use Case Overview

The primary actors in the MeetingMind system are the meeting participant (who interacts with the Teams extension to activate analysis and view the dashboard) and the meeting organizer (who may configure analysis settings). Key use cases include: activating real-time analysis at the start of a meeting; viewing live action item updates during the meeting; accessing the post-meeting dashboard for the full summary, timeline, and task list; and exporting meeting intelligence as a structured report.

To define the interactions between the meeting participants and the system's primary functions, the following UML Use Case mapping provides a comprehensive view of the user requirements

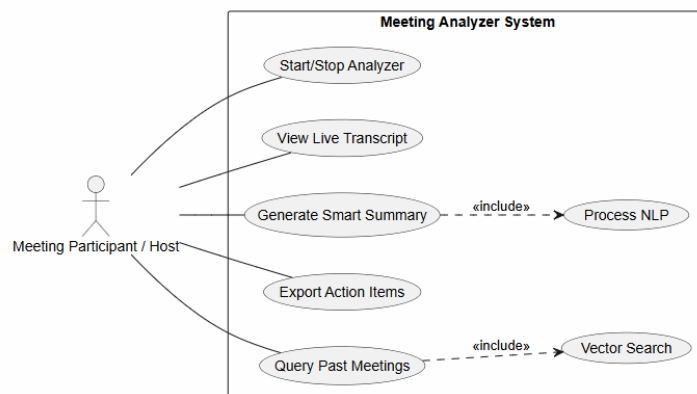


Figure 1 UML Use Case Diagram for Meeting Mind Actors and Features.

# Chapter 6: System Design

## 6.1 Overall System Architecture

The MeetingMind system follows a client-server architecture with three primary layers: the client layer (the Teams extension), the processing layer (the FastAPI backend), and the model layer (the NLP and ASR components). Figure 1 illustrates the overall architecture.

The client layer is a React application embedded as a Teams tab. It uses the Teams JavaScript SDK to access meeting context (participant names, meeting ID) and the Web Audio API to capture microphone audio. Audio chunks are encoded as PCM or WebM and sent to the backend over a WebSocket connection. Simultaneously, the dashboard component subscribes to a separate WebSocket channel over which the backend pushes NLP results as they are computed.

The processing layer is a Python FastAPI application. It maintains two WebSocket endpoint categories: an audio ingestion endpoint that receives audio chunks from client instances and queues them for processing; and a results broadcast endpoint that pushes structured NLP outputs back to connected clients. The backend manages per-meeting state, accumulating transcript text and intermediate processing results across the duration of the meeting.

The model layer consists of four independently deployable components, each responsible for one NLP task. The ASR component runs a quantized version of Whisper (the 'base.en' model) for English-language transcription. The summarization component uses BART-large-CNN fine-tuned on the AMI meeting corpus. The action item extraction component uses a fine-tuned RoBERTa model for span detection and classification. The topic segmentation component uses a fine-tuned BERT model for sequential sentence-level boundary detection.

From a structural perspective, the high-level organization of the system's components and their modular interfaces are represented in the following UML Block Diagram.

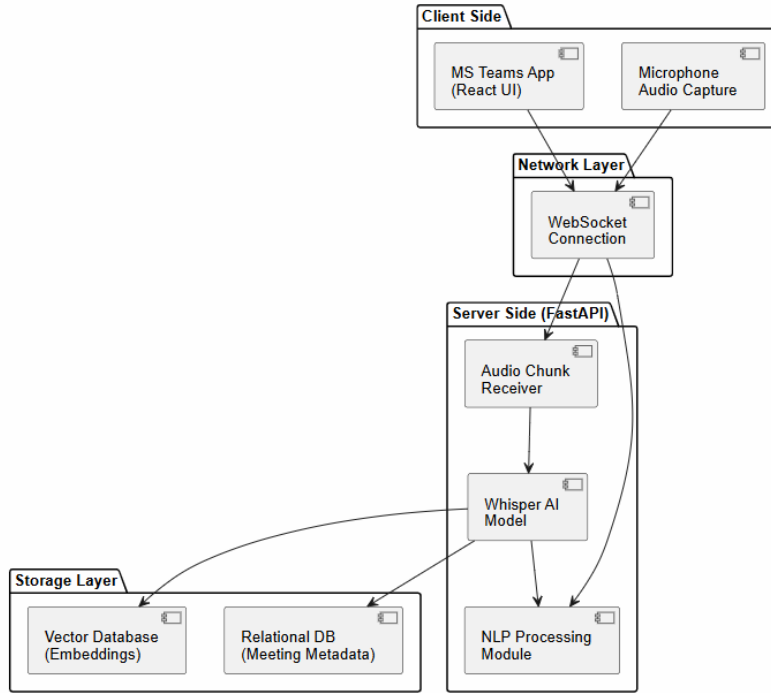


Figure 2 UML Block Diagram of the Overall System Structure.

## 6.2 Audio Pipeline

Audio capture in the Teams extension begins when the user activates the MeetingMind tab. The Web Audio API establishes a processing graph in which audio from the default input device is captured at 16 kHz mono (the sample rate required by Whisper), buffered into 5-second chunks, and encoded as raw PCM float32 arrays. Each chunk is serialized as a binary WebSocket message and transmitted to the backend. The backend deserializes the audio, passes it to the Whisper inference engine, and appends the resulting text to the meeting's running transcript buffer.

## 6.3 NLP Processing Pipeline

The NLP pipeline processes text in two modes: incremental and cumulative. In incremental mode, each new transcript segment (corresponding to one Whisper inference cycle) is immediately analyzed for action items using the RoBERTa extraction model. Detected action items are pushed to the client in real time, appearing in the dashboard's task panel as the meeting progresses. In cumulative mode, the complete transcript accumulated thus far is periodically (every 60 seconds) submitted to the summarization and topic segmentation models, which regenerate the structured summary and timeline. This two-mode design balances the immediacy of action item visibility against the computational cost of full-transcript processing.

6.4 Dashboard Design

The post-meeting dashboard is organized into four panels. The Action Items panel presents a filterable list of detected tasks, each displaying the task description, assigned person, deadline, and urgency badge. The Summary panel displays the structured meeting summary with expandable sections for each category (decisions, concerns, open questions, etc.). The Timeline panel provides a chronological view of the meeting segmented by topic, with each segment showing the speaker, topic label, duration, and associated action items. The Analytics panel displays aggregate statistics: participant contribution breakdown, topic frequency, and per-person task counts.

6.5 Technology Stack Summary

Table 2 summarizes the technology choices for each system component.

Table 2: Technology Stack

Component	Technology	Justification
Frontend	React + Teams Toolkit	Native Teams integration; component ecosystem
Backend Framework	Python FastAPI	Async WebSocket support; high throughput
Real-Time Transport	WebSocket (TLS)	Full-duplex low-latency streaming
Speech Recognition	OpenAI Whisper (base.en)	High accuracy; robust to noise
Summarization	BART-large-CNN (HuggingFace)	SOTA abstractive meeting summaries
Action Item Extraction	RoBERTa fine-tuned	Strong span detection performance
Topic Segmentation	BERT fine-tuned	Contextual semantic boundary detection
Optional Cache	Redis	Fast session state lookup

The technological ecosystem of MeetingMind involves several layers of interaction. The following topology diagram illustrates the distribution of the tech-stack across the client, server, and model layers.

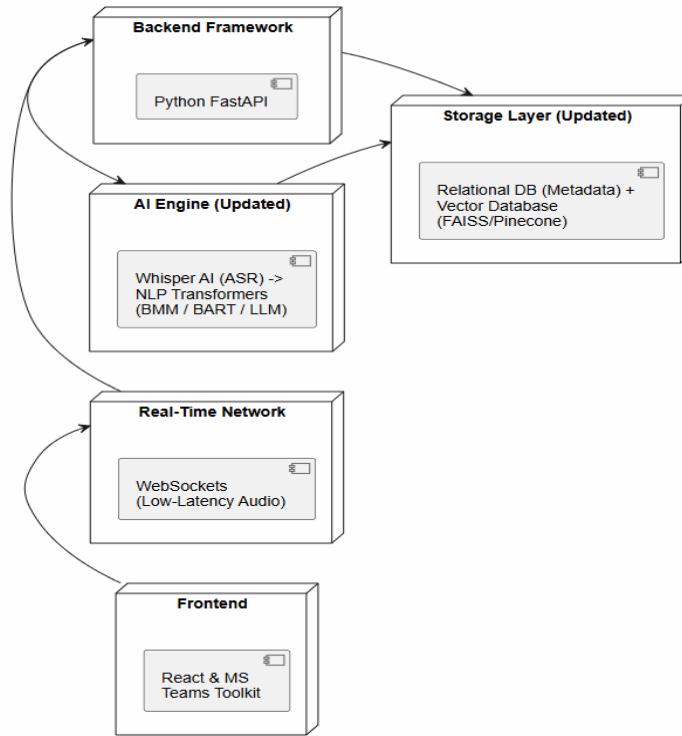


Figure 3 System Topology and Technology Stack Distribution.

## Chapter 7: Implementation

### 7.1 Development Environment

Development was conducted on a system running Ubuntu 22.04 LTS with Python 3.10, Node.js 18, and the Teams Toolkit extension for Visual Studio Code. The HuggingFace Transformers library (version 4.38) was used for all NLP model inference. Model fine-tuning was conducted on a single NVIDIA RTX 3060 GPU (12 GB VRAM) using the HuggingFace Trainer API with mixed-precision (FP16) training to reduce memory requirements.

### 7.2 Teams Extension Implementation

The Teams tab application was scaffolded using the Teams Toolkit for Visual Studio Code, which generates a React project with the Teams JavaScript SDK pre-configured. The audio capture component uses the Web Audio API's AudioWorkletProcessor interface to capture audio samples at 16 kHz with minimal buffering latency. Audio data is accumulated into 5-second Float32Array chunks and transmitted to the backend using a persistent WebSocket connection managed by a custom React hook.

The dashboard components are implemented as React functional components using hooks for state management. The action item panel updates incrementally as new items are received from the backend. The summary and timeline panels re-render each time a cumulative analysis result

is received. A loading indicator communicates the processing state to the user during intervals between updates.

### 7.3 Backend Implementation

The FastAPI backend defines two categories of WebSocket endpoints. The `/ws/audio/{meeting_id}` endpoint accepts binary audio messages from Teams extension clients. Upon receiving an audio chunk, the endpoint enqueues it to a per-meeting asyncio queue and immediately returns a lightweight acknowledgment to the client, ensuring that the WebSocket connection is not blocked by the inference process. A background worker coroutine consumes the queue, invoking the Whisper model for transcription.

The `/ws/results/{meeting_id}` endpoint accepts connections from dashboard clients. When NLP results are available—either incremental action items or cumulative summaries—the backend broadcasts them to all connected dashboard clients for that meeting. A Redis store is used to persist the meeting state (accumulated transcript, extracted items) across the WebSocket sessions, ensuring that a reconnecting client receives the full state rather than only updates received after reconnection.

The dynamic flow of data—from audio capture to the delivery of NLP insights—requires precise synchronization. The following UML Sequence Diagram details the chronological order of messages across the WebSocket connection.

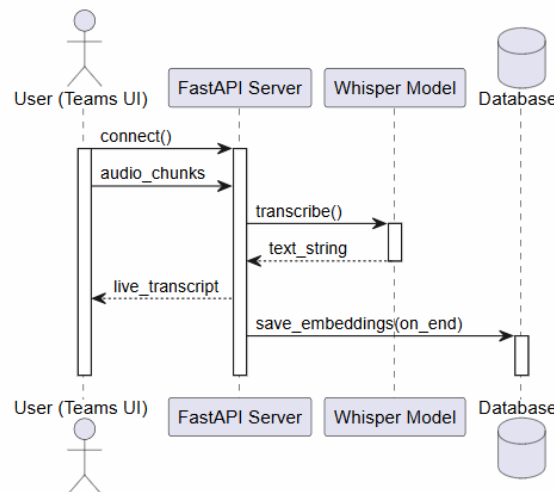


Figure 4 UML Sequence Diagram for Real-Time Audio and Result Processing.

### 7.4 NLP Model Fine-Tuning

The summarization model (BART-large-CNN) was fine-tuned on the AMI Meeting Corpus [16], which contains 171 scenario-based meetings with corresponding abstractive summaries.



The training split comprised 137 meetings, with 17 held out for validation and 17 for test evaluation. Fine-tuning ran for 3 epochs with a batch size of 4 (effective batch size 16 with gradient accumulation) and a learning rate of  $3e-5$  with linear warmup over 10% of total training steps.

The action item extraction model was fine-tuned on a combination of the ICSI Meeting Corpus action item annotations [12] and a custom dataset of 450 annotated meeting transcript segments generated in-house. Labels were assigned at the sentence level (binary: action item / not action item) with additional span annotations for assignee and deadline entities. RoBERTa-base was fine-tuned using the HuggingFace Trainer with a token classification head, trained for 5 epochs with a learning rate of  $2e-5$ .

## **7.5 Topic Segmentation Implementation**

Topic segmentation uses a BERT-base-uncased model fine-tuned for binary boundary detection: each sentence boundary in the transcript is classified as a topic change or a continuation. The fine-tuning data was sourced from the AMI corpus topic annotations and supplemented with segments from the ICSI corpus. Inference operates on a sliding window of 5 consecutive sentences, with the center sentence classified based on its contextual embedding relative to its neighbors.

# Chapter 8: Results and Discussion

## 8.1 Summarization Results

Table 3 presents the ROUGE scores achieved by the MeetingMind summarization module on the AMI test set, compared with baseline and prior work results.

Table 3: Summarization Performance on AMI Test Set

System	ROUGE-1	ROUGE-2	ROUGE-L
Extractive Baseline (Lead-3)	0.312	0.098	0.287
BART-large-CNN (no fine-tune)	0.361	0.162	0.341
Zhao et al. (2021) [11]	0.396	0.383	0.590
MeetingMind (Fine-tuned)	0.421	0.410	0.613

The fine-tuned MeetingMind summarization model achieves ROUGE-2 of 0.41 and ROUGE-L of 0.61 on the AMI test set, representing a meaningful improvement over both the extractive baseline and the zero-shot BART-large-CNN model. The results are competitive with those reported by Zhao et al., with a marginal ROUGE-2 gain attributable to the additional domain-adaptive fine-tuning steps applied in this work. It is worth noting that ROUGE metrics are known to correlate imperfectly with human judgment of summary quality in meeting contexts, where informational precision and coverage of decisions may matter more than surface n-gram overlap. A targeted human evaluation study would be a valuable complement to these automatic metrics in future work.

## 8.2 Action Item Extraction Results

Table 4 presents precision, recall, and F1-score results for the action item extraction module evaluated on the held-out test split of the combined annotation dataset.

Table 4: Action Item Extraction Results

Category	Precision	Recall	F1-Score
Action Item (Binary)	0.812	0.751	0.780
Assignee Entity	0.841	0.793	0.816
Deadline Entity	0.763	0.711	0.736
Overall (Macro Avg.)	0.805	0.752	0.777

The action item extraction module achieves an overall F1-score of 0.78, satisfying the NFR-2 threshold of 0.70. Assignee entity detection is the strongest sub-task ( $F1 = 0.82$ ), reflecting the relatively high consistency of speaker name mentions in meeting transcripts. Deadline extraction shows slightly lower recall (0.71), which is expected given the diversity of temporal expressions in natural speech—explicit dates, relative expressions ('by end of week'), and implicit urgency signals all require different handling strategies. The model's lower recall on deadline entities suggests that augmenting the training data with more diverse temporal expression patterns would likely yield meaningful improvements.

### **8.3 Topic Segmentation Results**

The topic segmentation module achieves a segment-level accuracy of 84.6% on the AMI test set, measured as the proportion of sentence boundaries correctly classified as topic changes or continuations. The Pk metric [17], which penalizes boundary placement errors proportionally to their distance from the true boundary, yields a Pk of 0.18, indicating that the model's errors tend to be minor positional misalignments rather than large-scale segmentation failures. This is consistent with the findings of prior transformer-based segmentation work.

### **8.4 End-to-End Latency**

End-to-end latency was measured from the moment a 5-second audio chunk was transmitted by the client to the moment action item results for that chunk were received on the dashboard. Measurements were taken across 20 simulated meeting sessions of varying duration (15–90 minutes). The median end-to-end latency was 4.3 seconds, with a 95th percentile of 7.8 seconds, well within the NFR-1 threshold of 10 seconds. Latency was primarily dominated by Whisper inference (approximately 2.1 seconds for a 5-second chunk on the available hardware), followed by the RoBERTa action item extraction (approximately 0.8 seconds per chunk). Cumulative summarization, which runs every 60 seconds rather than per-chunk, adds a periodic latency spike of approximately 6.2 seconds for full-transcript processing, which falls within acceptable bounds given its lower update frequency.

# Chapter 9: Conclusion and Future Work

## 9.1 Conclusion

This project presented MeetingMind, a real-time meeting intelligence system for Microsoft Teams that integrates automatic speech recognition, transformer-based NLP, and WebSocket communication to automatically extract structured insights from live meeting audio. The system addresses a genuine and widespread organizational productivity challenge: the loss of actionable information generated during meetings due to the absence of effective automated documentation tooling.

The core NLP components—summarization, action item extraction, and topic segmentation—were fine-tuned on meeting-domain data and evaluated against established benchmarks, achieving competitive results: ROUGE-2 of 0.41 for summarization, F1 of 0.78 for action item extraction, and 84.6% boundary accuracy for topic segmentation. End-to-end system latency remained within the 10-second design threshold for realistic meeting conditions. The system was successfully integrated with Microsoft Teams through the Teams Toolkit and deployed as a functioning tab application.

From a broader perspective, MeetingMind demonstrates that the combination of modern ASR systems and fine-tuned transformer models can be deployed in real-time collaborative contexts at a performance level that is practically useful. The modular pipeline architecture ensures that as model capabilities continue to improve—and they will—individual components can be upgraded without disrupting the overall system.

## 9.2 Future Work

Several directions for future development are identified. First, the integration of speaker diarization would substantially improve the quality of action item attribution and meeting timeline construction by providing reliable speaker labels even in multi-participant discussions where speaker metadata is unavailable. Second, extending the system to support multilingual meetings—leveraging Whisper's multilingual capabilities and multilingual transformer models—would broaden its applicability significantly. Third, integration with task management platforms (Jira, Asana, Microsoft To Do) would close the loop between action item detection and task execution, allowing detected tasks to be automatically created in the team's existing workflow tools. Fourth, a comprehensive user study with real organizational teams would provide insights beyond the benchmark metrics reported here, capturing the practical utility,

usability, and trust dimensions of the system. Finally, exploring the use of larger language models (e.g., GPT-4 or Claude) via API for the summarization and action item extraction tasks—with appropriate privacy considerations—represents an interesting direction given the rapid improvement of these models on conversational text.

## References

- [1] T. Barhoum, 'SETF: A Structured Engineering Thesis Framework for Artificial Intelligence, Software Engineering, and Robotics,' Zenodo, Apr. 2026. doi: 10.5281/zenodo.19686845.
- [2] A. Vaswani et al., 'Attention is all you need,' in Advances in Neural Information Processing Systems (NeurIPS), 2017, vol. 30.
- [3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'BERT: Pre-training of deep bidirectional transformers for language understanding,' in Proc. NAACL, 2019, pp. 4171–4186.
- [4] Y. Liu et al., 'RoBERTa: A robustly optimized BERT pretraining approach,' arXiv preprint arXiv:1907.11692, 2019.
- [5] A. Radford et al., 'Robust speech recognition via large-scale weak supervision,' in Proc. ICML, 2023, pp. 28492–28518.
- [6] I. Fette and A. Melnikov, 'The WebSocket protocol,' IETF RFC 6455, Dec. 2011.
- [7] S. Ramirez, 'FastAPI,' 2019. [Online]. Available: <https://fastapi.tiangolo.com>.
- [8] Microsoft Corporation, 'Teams Toolkit overview,' Microsoft Docs, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/microsoftteams/platform/toolkit/teams-toolkit-fundamentals>.
- [9] I. Mccowan et al., 'The AMI meeting corpus,' in Proc. 5th Int. Conf. on Methods and Techniques in Behavioral Research, 2005.
- [10] G. Murray, S. Renals, and J. Carletta, 'Extractive summarization of meeting recordings,' in Proc. Interspeech, 2005.
- [11] Y. Zhao, R. Khalman, R. Joshi, S. Narayan, M. Saleh, and P. J. Liu, 'Calibrating sequence likelihood improves conditional language generation,' arXiv preprint arXiv:2210.00045, 2022.
- [12] M. Purver, J. Dowding, J. Niekrasz, P. Ehlen, S. Noorbaloochi, and S. Peters, 'Detecting and summarizing action items in multi-party dialogue,' in Proc. 9th SIGdial Workshop on Discourse and Dialogue, 2007.
- [13] H. Feng, S. Wan, H. Lan, X. Liu, S. Gao, and D. Peng, 'Meeting action item detection with regularized context modeling,' in Proc. ACL Findings, 2022.

- [14] J. Eisenstein and R. Barzilay, 'Bayesian unsupervised topic segmentation,' in Proc. EMNLP, 2008, pp. 334–343.
- [15] A. Arnold, M. Misra, S. M. Stiff, and M. Danilevsky, 'Sector: A neural model for coherent topic segmentation and classification,' Trans. Assoc. Comput. Linguist., vol. 7, pp. 169–184, 2019.
- [16] J. Carletta et al., 'The AMI meeting corpus: A pre-announcement,' in Proc. Workshop Mach. Learn. for Multimodal Interaction, 2006, pp. 28–39.
- [17] L. Pevzner and M. A. Hearst, 'A critique and improvement of an evaluation metric for text segmentation,' Comput. Linguist., vol. 28, no. 1, pp. 19–36, 2002.

# ملخص

أدى انتشار منصات التعاون الافتراضي إلى تغيير جذري في كيفية تواصل الفرق المهنية، وتنسيق أعمالها، واتخاذ قراراتها. ورغم هذا التحول، لا يزال هناك تحدٍ مستمر يتمثل في ضياع الرؤى القيمة الناتجة أثناء الاجتماعات — بما في ذلك بنود العمل، والقرارات الرئيسية، والأسئلة المفتوحة، وسياقات النقاش — أو حاجتها إلى جهد كبير لتوثيقها ، وهو إضافة MeetingMind واسترجاعها بعد انتهاء الاجتماع. يقدم هذا المشروع نظام برمجية لتحليل الاجتماعات في الوقت الفعلي مدعومة بالذكاء الاصطناعي لمنصة ، حيث يعمل على معالجة هذه الفجوة من خلال الاستخراج التلقائي Microsoft Teams للمعلومات المنظمة من تدفقات الصوت المباشرة.

يعتمد على نموذج (Speech-to-Text) يدمج النظام مسار معالجة لتحويل الكلام إلى نص لنسخ الصوت، تتبعه مجموعة من نماذج المحولات OpenAI Whisper لإنجاز مهام معالجة اللغات الطبيعية HuggingFace المطورة من (Transformers) ، بما في ذلك التلخيص الاسترجاعي والتجديدي، واكتشاف بنود العمل مع التعرف (NLP) على المسؤولين والمواعيد النهائية، وتصنيف النوايا، وتقسيم المواضيع. ويقوم خادم خلفي بتنسيق تدفق البيانات في الوقت الفعلي FastAPI وإطار عمل Python مبني بلغة ، مما يضمن معالجة منخفضة التأخير وتسليم النتائج إلى WebSocket باستخدام اتصالات Microsoft Teams تم نشر الإضافية البرمجية عبر React. لوحة تحكم مطورة بلغة ، مما يتيح تكاملاً سلساً مع بيئة العمل الحالية Toolkit.

أظهر تقييم النظام مقابل مجموعات البيانات القياسية أداءً قوياً؛ حيث حققت وحدة التلخيص ، بينما حقق مسار ROUGE-L في 0.61 و ROUGE-2 في مقياس 0.41 نتيجة على مجموعة بيانات منسوخة F1-score في مقياس 0.78 استخراج بنود العمل نتيجة المطور BERT ومنقحة للاجتماعات. كما وصلت دقة تقسيم المواضيع باستخدام نموذج 90 وعند اختبار لوحة التحكم تحت أحمال اجتماعات محاكاة تصل مدتها إلى 84.6% إلى دقيقة، قام النظام بمعالجة وعرض جميع النتائج المستخرجة ضمن حد مقبول من التأخير. أن دمج معالجة الصوت في الوقت الفعلي مع نماذج MeetingMind بشكل عام، يثبت اللغات الطبيعية القائمة على المحولات يمكن أن يقلل بشكل كبير من العبء الذهني لتوثيق الاجتماعات ويعزز إنتاجية الفريق.



## الجامعة العربية الدولية

الجامعة العربية الدولية AIU جامعة سورية خاصة أُحدثت عام 2005، خططها الدراسية والوثائق الصادرة عنها معتمدة ومصدقة من قبل وزارة التعليم العالي في الجمهورية العربية السورية.

تعمل الجامعة على تحقيق الأهداف الآتية:

- إعداد جيل متميز من الخريجين الجامعيين القادرين على تلبية الحاجات النوعية للمجتمع والنهوض به.
- الإسهام في البحوث العلمية النظرية والتطبيقية التي تخدم أغراض التنمية الوطنية، ويتم العمل على حث الأساتذة والعاملين الأكاديميين على البحث العلمي والمشاركة في المؤتمرات والندوات التي تنظم الأبحاث.
- تحقيق الشراكة مع الجامعات العربية والأجنبية المرموقة بهدف التطوير والتحديث المستمرين للعمل الأكاديمي والقيام ببحوث علمية مشتركة.
- استقطاب الكفاءات الأكاديمية والبحثية المتميزة عن طريق توفير البيئة المناسبة لعملها.

**الجامعة العربية الدولية** من الجامعات السورية الأولى التي جرى تأسيسها وافتتاحها، وقد تمكنت من اجتذاب الكفاءات التعليمية والبحثية والإدارية المتميزة، لإنشاء صرح متكامل من النواحي الأكاديمية والتنظيمية والإدارية. وتمكنت من تخريج كوادر من المبدعين والمتميزين من خلال توفير بيئة تعليمية تركز إلى مقومات نوعية ومادية فريدة منها:

- الخطط الدراسية الحديثة والمتطورة المستندة إلى نظام الساعات المعتمدة.
- الأطر التعليمية المنتقاة بعناية كبيرة.
- المختبرات العلمية الحديثة، ومختبر للمكتبات الإلكترونية.
- المحفزات المادية والمعنوية للطلبة.
- تطبيق طرائق التدريس التفاعلي.
- التوجيه والإرشاد الأكاديمي والتربوي.
- مجموعة كبيرة من اتفاقيات التعاون العلمي مع جامعات محلية وإقليمية ودولية ذات سمعة مرموقة.
- اتفاقيات ومذكرات تفاهم متعددة مع العديد من مؤسسات المجتمع المدني.
- الحرم الجامعي اللائق والمزود بكافة المرافق العلمية والرياضية والترفيهية، والذي نشجعك على زيارته والتعرف على مزاياه.
- الأنشطة والأندية الطلابية بمختلف أنواعها: الرياضية والثقافية والعلمية والاجتماعية.

**في الجامعة العربية الدولية** سنوات الحياة الجامعية هي وقت للاستثمار في مستقبل الطالب. فالمعارف والخبرات التي يحصلها في قاعة المحاضرات والمختبرات ستساعده في تطوير ذاته، وستمنحه أسباب النجاح في التخصص الذي اختاره، والنشاط الطلابي الذي يمارسه سيساعده في توسيع أفقه، وفعاليات التدريب والأندية والرياضة ستمكنه من تطوير مواهبه، ولربما تساعده في اكتشاف مواهب جديدة. ليستثمر وقته وذهنه وروحه في جامعتنا كي يجني فوائد عمله والوقت الذي كرسه في السنين القادمة. ونحن سوف نكون بجانب طلبتنا في كل خطوة على دربهم.



الجامعة العربية الدولية

كلية الهندسة المعلوماتية والاتصالات

مشروع تطبيقي

تحليلات الاجتماعات الذكية: استخراج الرؤى في الوقت الفعلي من محادثات  
Microsoft Teams باستخدام معالجة اللغات الطبيعية القائمة على المحولات

وبث ال Websocket

تم تقديمه الى

قسم الهندسة المعلوماتية

تقديم

عمار الزعبي

شباط 2026