PROJECT REPORT ON

AI CLASSROOM ASSISTANT USING INTEL OPENVINO AND MICROSOFT PHI-2

ClarifAI

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Technology in Computer Science and Engineering

Submitted by:
Aryan Naithani
Enrollment No: 2415800020
B.Tech (Hons.) CSE, 2024–2028

Atharv Golas
Enrollment No: 2415800022
B.Tech (Hons.) CSE, 2024–2028

Gauri Khandelwal
Enrollment No: 2415800033
B.Tech (Hons.) CSE, 2024–2028

Aryan Singh
Enrollment No: 2415800021
B.Tech (Hons.) CSE, 2024–2028

Ayushi Bansal
Enrollment No: 2415800025
B.Tech (Hons.) CSE, 2024–2028

Date of Submission: 5th July 2025

ABSTRACT

ClarifAI is a real-time intelligent tool designed to support interactive learning during online and offline classes. This assistant helps students resolve doubts instantly, while enabling teachers to stay focused on teaching. The primary goal of this project is to optimize a large language model using **Intel OpenVINO** to ensure faster, more efficient inference on resource-constrained devices.

The assistant is powered by **Microsoft's Phi-2** model—an advanced small language model optimized to deliver strong performance on academic queries. The optimized version of this model using OpenVINO significantly improves inference speed and lightens the load on the hardware without compromising accuracy.

The assistant can be accessed via a feature-rich **Chrome extension** that runs on top of Google Meet or any web-based learning platform.

Key features include:

- Natural language doubt resolution for high school-level queries.
- Support for **voice input** using Web Speech API.
- Easy integration into existing classroom environments through a browser extension.
- Enhanced inference speed and responsiveness due to OpenVINO optimization.

This project demonstrates the practical benefits of AI model optimization and deployment on edge devices, emphasizing speed, usability, and accessibility in real-world education settings.

INTRODUCTION

In modern classrooms, student engagement and timely doubt resolution are critical to effective learning. However, due to time constraints and large class sizes, students often hesitate or miss the opportunity to clarify their questions during live lectures. To address this issue, we have developed an AI-powered real-time doubt resolution assistant that provides instant, personalized responses to student queries.

ClarifAI leverages the Microsoft Phi-2 language model optimized using Intel's OpenVINO toolkit to deliver fast and efficient inference on local machines. The assistant is accessible through a lightweight, user-friendly Chrome Extension that integrates seamlessly into the learning environment, especially during online classes like those on Google Meet. This ensures students can interact with the assistant without interrupting the flow of the lecture.

The system is designed to act as a supportive educational tool that explains concepts in simple language, offers real-world examples, and encourages active learning by including short follow-up questions or exercises.

OBJECTIVES

The primary objective of this project is to develop a smart, real-time AI assistant that can help high school and college students resolve academic doubts during live classes with minimal disruption. The assistant should be accessible, fast, and capable of providing helpful and understandable responses.

The specific goals of the project are as follows:

1. **Real-Time Doubt Resolution**
   To allow students to ask questions and receive detailed, easy-to-understand answers instantly during lectures.

2. **Optimized Local Inference**
   To integrate and optimize the Microsoft Phi-2 language model using Intel OpenVINO, enabling fast response times even on resource-constrained local machines.

3. **Browser Extension for Accessibility**
   To design a Chrome Extension that brings the assistant directly into virtual classrooms (e.g., Google Meet) without the need for switching tabs or devices.

4. **Voice Input Support**
   To enhance usability and inclusivity by enabling speech-to-text functionality, allowing students to ask questions verbally.

5. **Enhanced Educational Experience**
   To provide not just direct answers but also real-world examples and short follow-up exercises that promote better conceptual understanding.

Team Contributions

---

1. **Lead Developer – Aryan Naithani**
    - Responsible for the overall system design and core architecture.
    - Handled model integration using Microsoft Phi-2 optimized with Intel OpenVINO.
    - Developed the inference pipeline, testing logic, and real-time communication between frontend and backend.
    - Implemented context-awareness via Google Meet transcript capture and prompt conditioning.
    - Designed and implemented voice input functionality using Web Speech API.
    - Led performance benchmarking, optimization testing, and ensured smooth model deployment.
2. **UI/UX Design and Frontend Integration – Ayushi Bansal**
    - Structured and styled the Chrome Extension interface using HTML, CSS, and JavaScript.
    - Contributed to the user experience design, including visual layout and chat flow.
    - Designed toggles and buttons for voice and context input features.
    - Ensured responsive styling and user-friendly interactions.
3. **Chrome Extension Engineering – Atharv Golas**
    - Developed and refined the extension logic, including communication with content scripts.
    - Implemented Chrome APIs for permissions, tab querying, and runtime messaging.
    - Configured the extension’s manifest and ensured compatibility with browser security policies.

4. **Documentation and Reporting – Gauri Khandelwal**
    - Drafted the README file and updated it to reflect major changes.
    - Created and formatted the project report including technical summaries and architectural flow.
    - Organized installation instructions and showcased usage of features like voice and context.
    - Handled visual content like screenshots and diagrams of performance metrics.
5. **Testing, Feedback & Presentation – Aryan Singh**
    - Assisted with local deployment and testing of different components.
    - Collected feedback for iterative improvement of UI and response quality.
    - Created sample queries and helped validate model responses in context-enabled mode.
    - Prepared slides and presentation content for the final project submission.

SYSTEM OVERVIEW

ClarifAI is designed as a lightweight, efficient, and user-friendly tool to assist students during online classes. The system is divided into two main components:

1. **Backend (Inference Engine):**
    - Built using Python and Flask, the backend handles user queries, processes them through an optimized language model, and returns relevant responses.
    - The core AI model used is **Microsoft Phi-2**, optimized using **Intel OpenVINO** to ensure faster and more efficient inference on local hardware.
    - The inference engine formats student questions into structured prompts and retrieves detailed, context-aware answers.
2. **Frontend (User Interface):**
    - The primary interface is a **Chrome extension** that loads as a popup and connects with the local Flask server.
    - The extension includes a text-based chat system and supports **voice input** using the Web Speech API.

Workflow Summary:

- A student opens the extension and types or speaks a question.
- The question is sent to the local Flask server.
- The backend reformats the question into a structured prompt and sends it to the optimized Phi-2 model.
- The model generates an answer, which is sent back to the extension and displayed to the student.

TECHNOLOGY STACK

ClarifAI leverages a carefully selected set of technologies to ensure optimal performance, ease of development, and user accessibility. The stack is divided into three major layers: Back-end, Front-end, and Optimization Framework.

---

1. Backend Technologies

- **Python:**
  The core programming language used for server logic and model integration.

- **Flask:**
  A lightweight web framework used to create the backend API that communicates with the Chrome extension and processes user queries.

- **Hugging Face Transformers:**
  Initially used to load pre-trained models like **FLAN-T5**(during initial development) and **Phi-2** from Hugging Face's model hub.

- **Optimum + OpenVINO Toolkit:**
  Used to convert and accelerate the Phi-2 model for faster inference. OpenVINO enables running large models efficiently on local CPUs by optimizing computation graphs.

---

2. Model Optimization Tools

- **Intel OpenVINO:**
  This framework converts standard transformer models into a form that runs more efficiently on Intel hardware. It reduces memory footprint and speeds up inference without losing output quality.

- **Optimum-Intel:**
  A Hugging Face and Intel collaboration that bridges transformer models with OpenVINO, allowing seamless model export and inference integration.

---

3. Frontend Technologies

- **HTML/CSS/JavaScript:**
  Used to create the user interface of the Chrome extension, enabling text and voice interactions.

- **Chrome Extension API:**
  Provides the infrastructure to deploy the assistant as a browser extension with a popup interface.

- **Web Speech API:**
  Enables voice input functionality in supported browsers by integrating microphone-based speech-to-text.

MODEL OPTIMIZATION

One of the central goals of this project was to improve the performance and responsiveness of a large language model by optimizing it for local use, specifically on devices with limited computational resources.

Base Model: Microsoft Phi-2

- Phi-2 is a compact yet powerful language model released by Microsoft, known for its strong performance on reasoning and instruction-following tasks.
- However, running Phi-2 in its raw PyTorch form requires significant hardware, including a high-end GPU, which is not feasible for all users or classrooms.

Optimization Objective

To make the assistant usable on typical student or teacher devices (e.g., laptops with CPUs), the Phi-2 model was optimized using the Intel OpenVINO Toolkit.

Steps Followed for Optimization:

1. **Model Conversion:**
    - The pre-trained Phi-2 model was first downloaded and saved in ONNX (Open Neural Network Exchange) format.
    - ONNX provided a hardware-agnostic intermediate representation, compatible with OpenVINO.

2. **OpenVINO Model Conversion:**
    - The ONNX model was converted into OpenVINO Intermediate Representation (`.xml` and `.bin`) using the Model Optimizer.

    - This reduced model size and allowed inference through OpenVINO's Inference Engine.

3. **Integration with Optimum-Intel:**

    - The `optimum-intel` library was used to load and serve the optimized model within the Python backend.

    - This provided a seamless interface between the OpenVINO-optimized model and the Flask-based chatbot server.

Benefits Achieved:

- **Reduced Inference Latency:**
  Responses were generated significantly faster compared to the original PyTorch-based version.

- **Lower Resource Consumption:**
  The model could run efficiently on standard CPUs without requiring a GPU.

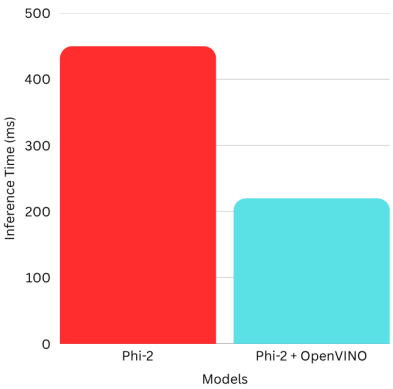
- **Increased Portability:**
  The solution became deployable on a wider range of systems, making it accessible for everyday classroom use.

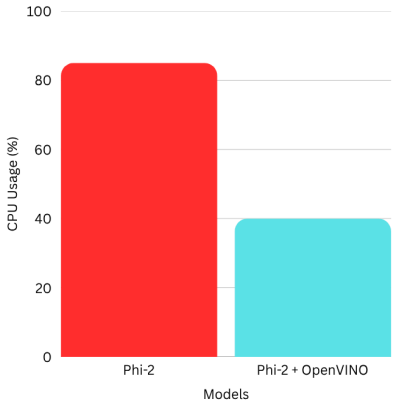
PERFORMANCE EVALUATION

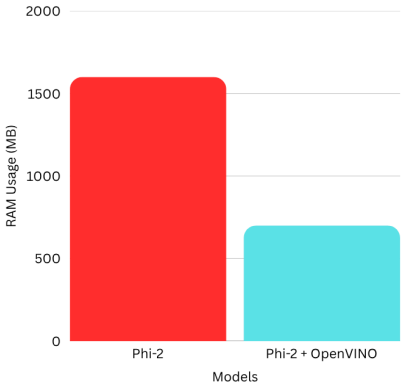
To assess the effectiveness of OpenVINO optimization, we compared the performance of the original Microsoft Phi-2 model (PyTorch version) and the optimized OpenVINO version across multiple metrics.

Evaluation Setup

- **Device Used:** Intel Core i5 (11th Gen), 8 GB RAM, no dedicated GPU
- **Test Environment:** Local Flask server using Python
- **Test Data:** A set of 10 common educational prompts across science, math, and general knowledge
- **Frameworks Compared:**
    - Baseline: PyTorch + Transformers
    - Optimized: OpenVINO + Optimum-Intel







Observations:

- The OpenVINO model was nearly **2x faster** on average in generating responses.
- The system remained responsive and smooth even under continuous usage.
- Memory and CPU utilization dropped significantly, allowing multitasking on standard laptops.

Conclusion:

OpenVINO optimization clearly enhanced the deployability of the model in resource-constrained environments like classrooms. It allowed high-quality AI interaction without the need for expensive GPUs or cloud infrastructure.

Chrome Extension – Interface and Features

The Chrome extension is the primary interface for interacting with the AI Classroom Assistant. It was designed to be lightweight, responsive, and easy to use within any live online classroom setting, such as Google Meet or Zoom.

Key Features:

1. **Popup Chat Interface:**
    - The extension appears as a simple popup when clicked from the toolbar.
    - It supports a familiar chat-like interface where users can type questions and receive answers in real time.
2. **Voice Input Support:**
    - Users can click on the mic icon to ask questions using voice input.
    - The extension uses the browser's Web Speech API to transcribe the question before sending it to the backend.
3. **Typing Indicator and Timestamps:**
    - Messages show timestamps to improve context.
    - A typing indicator is displayed while the AI is generating a response.
4. **Persistent Conversation:**
    - The extension retrieves previous messages from the Flask server and maintains chat history during the session.
5. **Visual Feedback and Error Handling:**
    - Clear visual indicators are used for loading, error messages, and speech recognition status.
    - Network issues or lack of microphone access are handled gracefully.

6. **Security and Permissions:**
    - The extension requests minimal permissions (microphone, activeTab).
    - It runs entirely on `localhost`, ensuring privacy and offline support (if the model is run locally).

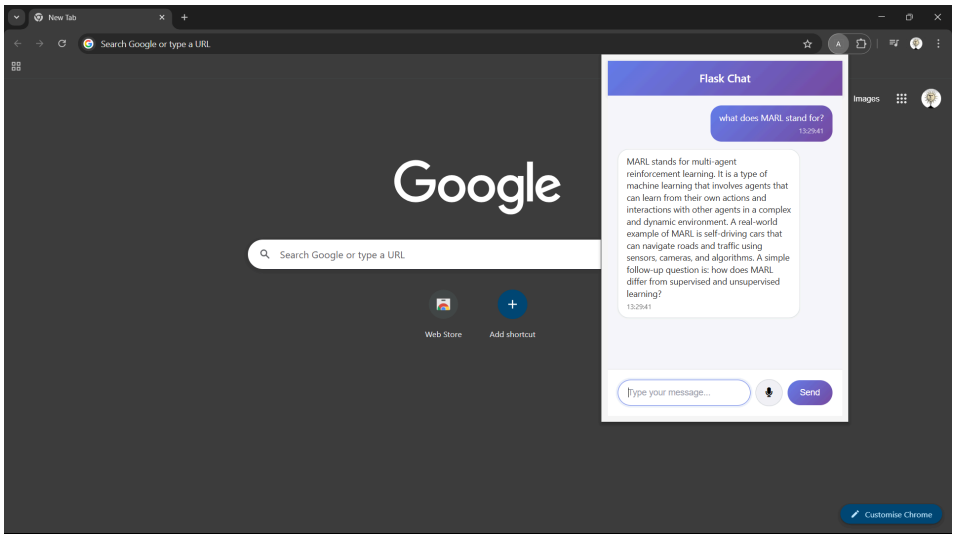
Technologies Used:

- HTML, CSS, and JavaScript for the frontend
- Manifest V3 format for extension configuration
- Web APIs like `SpeechRecognition` and `fetch()` for real-time interaction

Why the Extension?

- **Portability:** Works on any Chromium-based browser (Chrome, Edge, Brave).
- **Usability:** Students can interact with the assistant without switching tabs or opening separate apps.
- **Speed:** Optimized for fast, local inference to reduce classroom interruptions.

Visual Preview:



Context Awareness in ClarifAI

To make the assistant more effective and responsive during live online sessions, ClarifAI includes a **Context Awareness** feature that enhances the relevance of its answers by incorporating ongoing discussions into its responses.

This feature is particularly useful during **Google Meet** classes, where live captions are generated in real time. A custom-built content script in the Chrome extension captures these captions from the meeting interface at regular intervals. These live transcriptions are stored in a buffer and serve as the *context* for any query a user submits through the extension.

When the user enables the “Use Context” toggle inside the extension, the assistant fetches the buffered transcript and appends it to the user's input as part of the prompt. This allows the AI model to understand not only the standalone question but also the broader conversation in which the question was asked. As a result, answers become significantly more aligned with the current topic being discussed in class.

Technical Workflow:

1. A content script continuously extracts captions from the Meet interface (targeting the caption `<div>` element).

2. Captions are stored in a transcript buffer and made accessible via Chrome’s messaging system.

3. When the toggle is enabled, the extension fetches this transcript and sends it along with the user’s query to the Flask backend.

4. The backend appends the transcript to the prompt, providing the model with rich, relevant background.

5. The model generates a more context-aware and helpful response.

Benefits:

- Better understanding of in-class discussions

- Reduces repetitive or redundant explanations

- Makes ClarifAI behave more like a live classroom assistant, rather than a generic chatbot

This addition bridges the gap between standalone AI tools and true real-time classroom integration, providing a glimpse into the future of context-sensitive intelligent tutoring systems.

Testing & Evaluation

ClarifAI was tested for both functionality and performance before and after model optimization. This involved checking system stability, response times, memory usage, and accuracy of answers.

Functional Testing

Component	Test Description	Result
Flask Backend	API tested with various questions	✅ Pass
Chrome Extension	UI tested for responsiveness and behavior	✅ Pass
Voice Input	Tested using SpeechRecognition on Chrome	✅ Pass
Error Handling	Disconnected mic, no internet, invalid input	✅ Pass
Model Response Quality	Compared with original HuggingFace Phi-2	✅ Comparable

---

Conclusion & Future Scope

Conclusion

ClarifAI demonstrates the practical application of generative AI in real-time educational environments. By leveraging Microsoft's Phi-2 language model and optimizing it using Intel's OpenVINO toolkit, we were able to significantly improve performance in terms of inference speed and memory efficiency — both of which are critical for real-time usage in classroom settings.

The combination of a lightweight Flask backend and a polished Chrome extension provided a user-friendly interface for interaction. Students could ask questions, receive detailed AI-generated answers, and even use voice input for added convenience.

The success of this project shows how open-source tools, model optimization techniques, and simple web technologies can come together to create impactful educational tools.

Future Scope

While the current version serves as a solid foundation, the project has significant room for growth:

- **Learning Analytics:**
  Track types of questions asked, student engagement metrics, and topic-wise performance to assist teachers.
- **Teacher Dashboard:**
  Create an admin panel where teachers can view questions asked during class and student performance trends.
