NATIONAL ECONOMICS UNIVERSITY
FACULTY OF DATA SCIENCE AND ARTIFICIAL INTELLIGENCE
DATA SCIENCE PROGRAM

REPORT: WEB INTERVIEW RECORDER

Implementing Reliability and AI-Powered Transcription

Student:

Nguyen Viet Tien (ID: 11247233)
Pham Thu Ha (ID: 11247164)
Hoang Thanh Nhan (ID: 11274212)

Supervisor:

Dr. Tran Hung

December 10, 2025

TEAM CONTRIBUTION AND PRESENTATION DETAILS

CONTRIBUTION

Member Name	**Student ID**	**Contribution (%)**
Nguyen Viet Tien	11247233	33.34%
Pham Thu Ha	11247164	33.33%
Hoang Thanh Nhan	11247212	33.33%
	Total Contribution:	**100%**

DEMO VIDEO AND PRESENTATION LINK

- **YouTube Link:** `https://www.youtube.com/watch?v=YourVideoLinkHere`
- **Description:** End-to-end demonstration of token verification, sequential recording, upload reliability, and final review.

Contents

Abstract

This report details the implementation of a client-server web application for sequential video interviewing, emphasizing network reliability and advanced data processing. The core feature is the **Per-Question Upload** model, which is secured by server-side token validation and network resilience provided by the **Exponential Backoff and Retry Policy**. Furthermore, the project successfully integrates the **Gemini 2.5 Flash API** to perform **Speech-to-Text (STT)** transcription immediately after each video upload, providing granular data analysis (transcripts) and fulfilling the project's technical and bonus objectives.

1 Introduction and Project Objectives

1.1 Project Overview

The **Web Interview Recorder (HanTie)** project implements a client-server web application designed for structured, sequential video interviewing. The primary technical challenge involved implementing the core **Per-Question Upload** model to ensure data integrity and network resilience.

1.2 Learning Objectives and Connection to Computer Networking

The project serves as a practical demonstration of several key CNM concepts:

Networking Concept	**Implementation in HanTie**
Client-Server Communication	Node.js/Express (Server) handles requests; Pure JavaScript/HTML (Client) manages UI and media streams.
Transport Reliability	**Exponential Backoff and Retry Mechanism** on the Client for the upload API to prevent data loss over unstable networks.
API Contract	Four distinct API endpoints control the session lifecycle.
Secure Transport Requirement	Project mandates **HTTPS** for public deployment, essential for accessing media devices (`getUserMedia`).

Table 1: Connection of Project Features to Computer Networking Concepts.

2 System Design and Architecture

2.1 Client-Server Architecture and Flow

The application follows a standard Client-Server model. The Client's state machine progresses based only on successful (HTTP 200 OK) API responses.

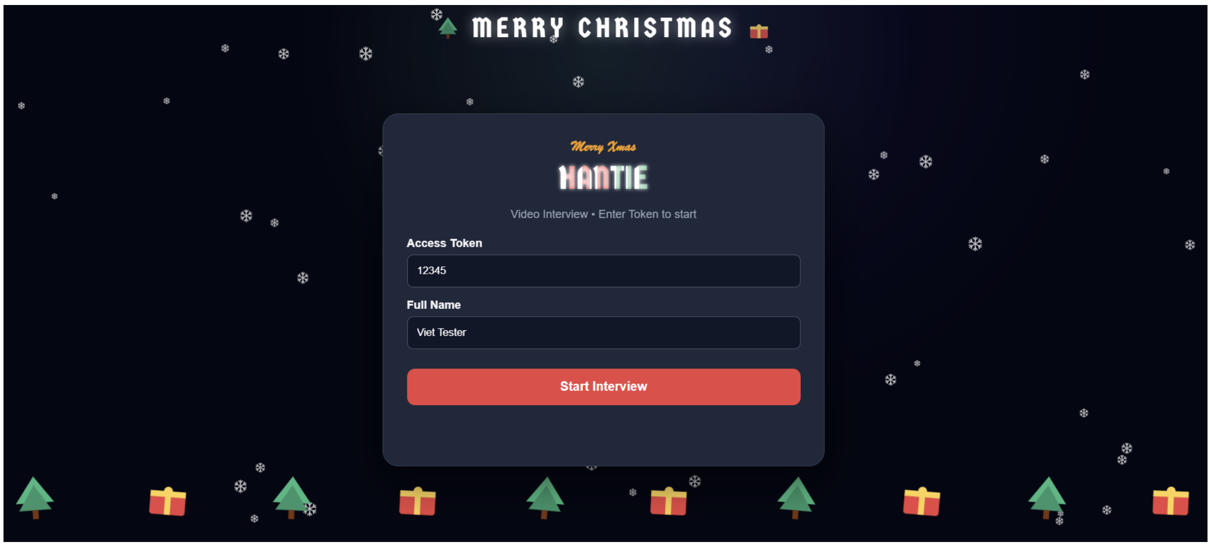


Figure 1: Start Screen (Token and Name Input).

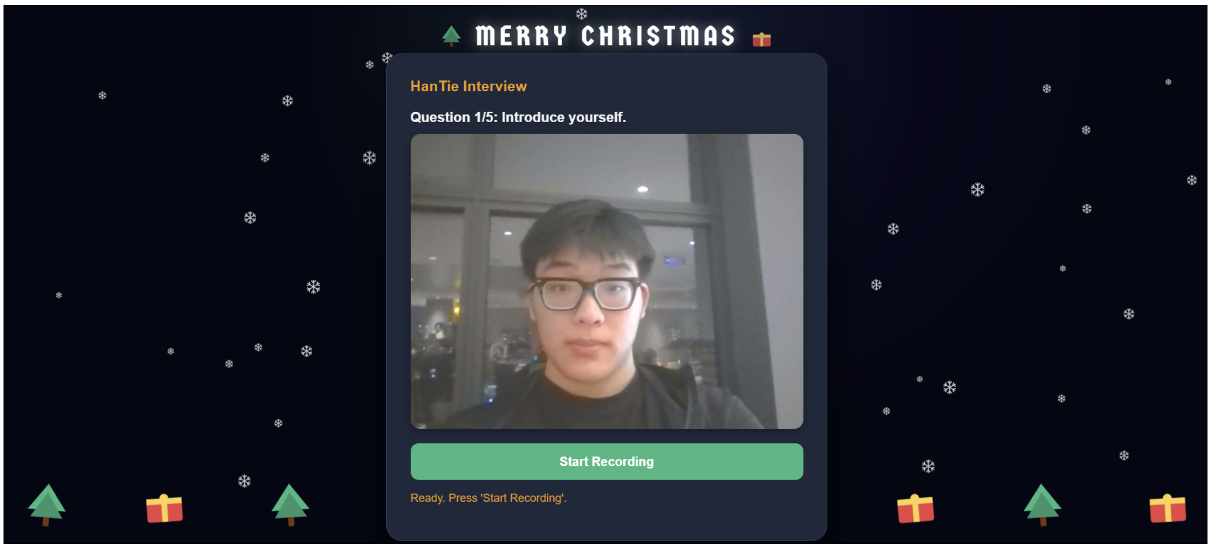

MERRY CHRISTMAS
HanTie Interview
Question 1/5: Introduce yourself.
Start Recording
Ready. Press 'Start Recording'.

Figure 2: Recording State (Camera Preview).

The session lifecycle follows a synchronous progression:

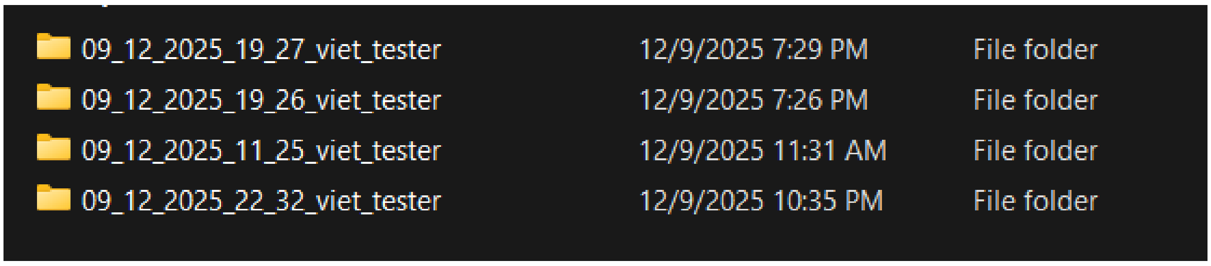
1. **Start Phase (→ `/api/session/start`):** The Server validates the token and creates a unique, timestamped session folder (`DD_MM_YYYY_HH_mm_ten_user/`).

2. **Interview Loop (→ `/api/upload-one`):** Record → Stop → **POST `/api/upload-one`** → Server saves video, runs STT, and updates metadata → Proceed to next question upon success.

3. **Finish Phase (→ `/api/session/finish`):** The Server finalizes the session metadata.

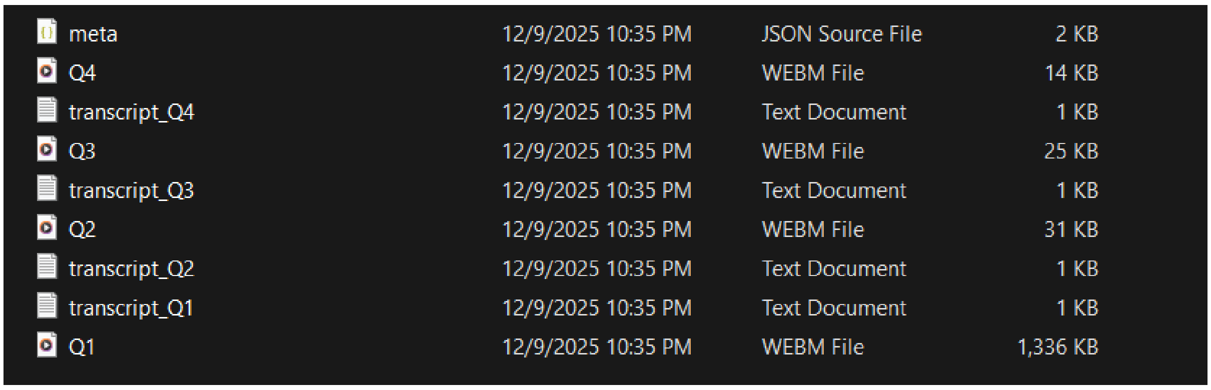
2.2 Storage and Metadata Management

Storage organization strictly adheres to the project requirements.



09_12_2025_19_27_viet_tester	12/9/2025 7:29 PM	File folder
09_12_2025_19_26_viet_tester	12/9/2025 7:26 PM	File folder
09_12_2025_11_25_viet_tester	12/9/2025 11:31 AM	File folder
09_12_2025_22_32_viet_tester	12/9/2025 10:35 PM	File folder

Figure 3: Server Storage Proof: Session Folder Naming Convention.



meta	12/9/2025 10:35 PM	JSON Source File	2 KB
Q4	12/9/2025 10:35 PM	WEBM File	14 KB
transcript_Q4	12/9/2025 10:35 PM	Text Document	1 KB
Q3	12/9/2025 10:35 PM	WEBM File	25 KB
transcript_Q3	12/9/2025 10:35 PM	Text Document	1 KB
Q2	12/9/2025 10:35 PM	WEBM File	31 KB
transcript_Q2	12/9/2025 10:35 PM	Text Document	1 KB
transcript_Q1	12/9/2025 10:35 PM	Text Document	1 KB
Q1	12/9/2025 10:35 PM	WEBM File	1,336 KB

Figure 4: Server Storage Proof: Folder Contents (Q*.webm and transcript_Q*.txt files).

3 Reliability and Security

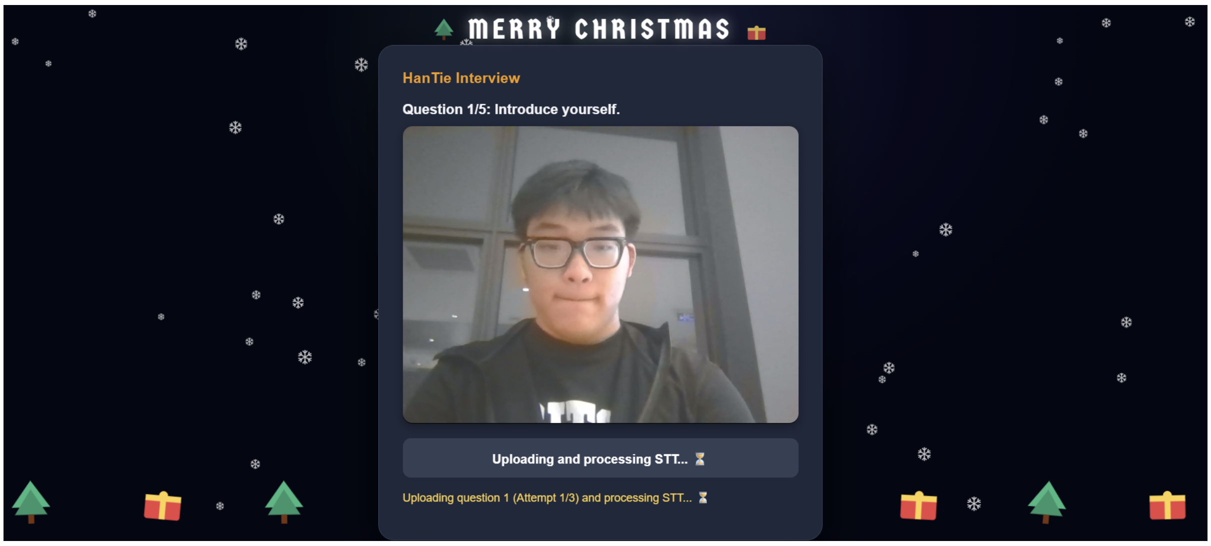
3.1 Per-Question Upload Integrity and Control Flow

The core reliability feature is the synchronous upload. The client is explicitly programmed to ensure that the user interface only transitions to the next question $(i+1)$ if the upload for question i is confirmed successful by the `/api/upload-one` response.

3.2 Network Resilience: Exponential Backoff

The implementation of the **Retry with Exponential Backoff** algorithm is critical for the stability of the upload endpoint.

- **Policy:** The client attempts up to three retries.
- **Delay Calculation:** The waiting time before each retry doubles: Delay $= 2^n \times 1$ second.
- **Control Flow Decision:** The next question is blocked until upload is successful. The upload status (Ref: Fig 5) clearly indicates the current attempt.



Question 1/5: Introduce yourself.

Uploading and processing STT... ⌛

Uploading question 1 (Attempt 1/3) and processing STT... ⌛

Figure 5: Retry Status Example: Showing upload status (Attempt 1/3) during the processing phase.

3.3 Security and Validation

- **Server-Side Token Validation:** The token is validated exclusively on the server.
- **Resource Control (Multer):** Input files are strictly filtered for `video/webm` and limited to 50MB.

- **Filesystem Safety:** The `userName` input is sanitized and truncated.
- **Logging:** All server events are logged with ISO 8601 timestamps in the Asia/Bangkok timezone.

3.4 Self-Check Questions

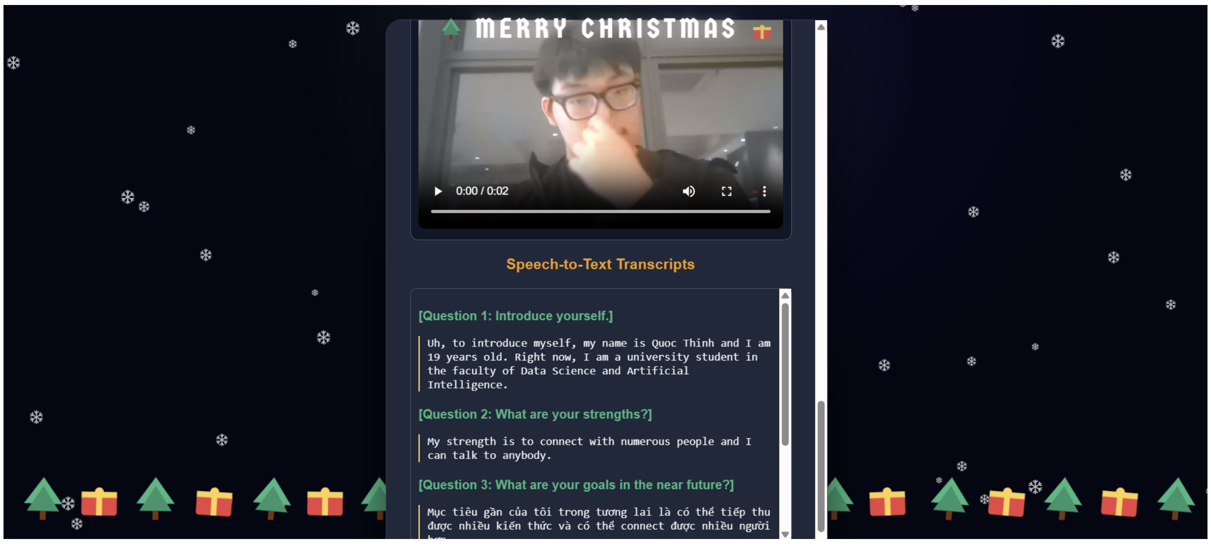
1. **Why must the token be validated on the server?** Server validation guarantees that the access control is authoritative and cannot be tampered with by a malicious client.
2. **What are the benefits of per-question recording and upload?** Per-question recording and upload minimizes the risk of catastrophic data loss. If the network connection fails midway through a long interview, only the current question's video is at risk, while all preceding answers are already secured on the server. Furthermore, it ensures a smaller file size for each upload, improving reliability and reducing network congestion compared to a single large file transfer.
3. **Why is HTTPS required publicly for camera/microphone permissions?** Browsers mandate the use of HTTPS (TLS/SSL encryption) for the `getUserMedia` API to prevent sensitive media data from being eavesdropped upon or modified by Man-in-the-Middle (MITM) attacks.
4. **How do you design retry/backoff and decide when to allow moving to the next question?** The retry/backoff mechanism utilizes the Exponential Backoff algorithm, attempting up to three retries with exponentially increasing delays. The system decides to allow moving to the next question $(i+1)$ only after the upload for question i is confirmed successful by the server response.
5. **Which Speech-to-Text engine and why? Accuracy/performance/cost trade-offs; per-question labeling.** The project uses the Gemini 2.5 Flash API for STT due to its high accuracy and speed. This ensures the prompt generation of transcripts, which are clearly segmented and labeled per question, supporting the granular data analysis requirement.

4 AI Analysis: Speech-to-Text

4.1 STT Implementation and Justification

The project integrated the **Gemini 2.5 Flash API** to perform per-question STT conversion.

- **Engine Choice:** Gemini 2.5 Flash was selected for its high accuracy and speed.
- **Trigger:** The STT process is integrated into the `/api/upload-one` route.



Speech-to-Text Transcripts

[Question 1: Introduce yourself.]

```
Uh, to introduce myself, my name is Quoc Thinh and I am
19 years old. Right now, I am a university student in
the faculty of Data Science and Artificial
Intelligence.
```

[Question 2: What are your strengths?]

```
My strength is to connect with numerous people and I
can talk to anybody.
```

[Question 3: What are your goals in the near future?]

```
Mục tiêu gần của tôi trong tương lai là có thể tiếp thu
được nhiều kiến thức và có thể connect được nhiều người
```

Figure 6: Frontend View of Speech-to-Text Transcripts: Showing segmented, per-question output.

4.2 Data Flow and Output Structure

The successful STT process requires robust data handling between the Server and the client.

- **Server Persistence:** After Gemini API processes the video file, the resulting text is formatted with the question header and saved locally on the server as `transcript_Q<index>.txt` in the session folder.
- **Client Feedback:** The raw transcript content is immediately included in the JSON payload response of the `/api/upload-one` endpoint. This ensures the client can quickly display the segmented transcript in the final review screen, confirming the per-question processing objective.

5 Conclusion, Limitations, and Future Work

5.1 Conclusion

The system successfully meets all core objectives, delivering a functional and reliable Web Interview Recorder, satisfying the critical per-question upload model, and successfully implementing the STT bonus feature.

5.2 Limitations

1. **Compatibility:** Compatibility is limited strictly to the `video/webm` format.
2. **UX Feedback:** A true byte-level upload progress bar is missing.

5.3 Future Work

- **Transcoding Integration:** Integrate `FFmpeg` on the server to automatically accept and standardize various video formats.
- **One-time Re-record:** Implement a feature allowing users to discard and re-record an answer once per question.
- **Enhanced Logging:** Introduce a client-side logging API to send detailed error information to the server for improved diagnostics.

References

[1] Project Brief: Web Interview Recorder (Per-Question Upload). National Economics University, 2025.

[2] Network and Communication Technology Course Material (CNM Concepts: Client-Server, Exponential Backoff, HTTPS).

[3] Google Generative Language API (Gemini 2.5 Flash API Documentation). Accessed: December 2025.

[4] Technical Stack: Node.js, Express.js, Multer Middleware, etc.