PROJECT BRIEF – Web Interview Recorder (Per-Question Upload) — Full Terms

Audience: Team of 4 students

Duration: 5 weeks

Environment: Web (Frontend and Backend), runs on localhost; Hypertext Transfer Protocol Secure (HTTPS) is required if public

1) Learning Objectives

• Use MediaDevices.getUserMedia to request camera/microphone access; record one video per question.

• Design a sequential User Interface (UI) for interviews (≤ 5 questions); use a Next button to end the current question and proceed.

• Upload immediately after each question (per-question upload); manage the session via Application Programming Interface (API) endpoints: verify-token, session/start, upload-one, session/finish.

• Organize server storage as: DD_MM_YYYY_HH_mm_ten_user/ (timezone: Asia/Bangkok) with metadata.

• (Bonus) Per-question Speech-to-Text (STT) to produce transcript.txt.

1.x) Connection to Computer Networking (simple explanation)

• This is a client–server exercise: the browser (client) sends data to the server over Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS).

• You will create a few simple Application Programming Interfaces (APIs) — start session, upload each question, finish — so both sides communicate correctly.

• The browser enables camera/microphone and sends each question's video to the server (similar to uploading files over a network).

• The system must handle network errors (loss, retries with backoff) to avoid data loss.

• The server names folders by time and keeps logs for verification and grading.

2) Problem Description

Candidate opens the link, enters Token and Name, and grants camera/microphone permissions. The app shows up to five questions sequentially; each question is recorded as a separate file. When Next is pressed, recording stops and that question's video is uploaded immediately. Press Finish to close the session. The server stores files under DD_MM_YYYY_HH_mm_ten_user/ with metadata; (Bonus) generate transcript.txt per question.

3) Scope and Constraints (mandatory)

• Maximum five questions; display one question at a time.

• Per-question upload: stop recording question i → upload immediately; proceed to i+1 only after successful upload (or with a clear retry/backoff policy).

• Session flow: session/start → upload-one (each question) → session/finish.

• Token validated on the server; client-side checks are advisory only.

• Sanitize candidate name to create a filesystem-safe folder name.

• Server structure: DD_MM_YYYY_HH_mm_ten_user/ with Q1.webm..., metadata (for example, meta.json), and (Bonus) transcript.txt.

• State size limits and accepted Multipurpose Internet Mail Extensions (MIME) types clearly in the README.

• HTTPS is required when deployed publicly to access camera/microphone.

4) Functional Requirements

• Start: Enter Token and Name → verify-token → request camera/microphone.

• Interview: preview, current question, Next (stop and upload), Finish (end session; no more uploads).

• Per-question upload: call upload-one (multipart/form-data) with token, folder, questionIndex, and video; show status (uploading/success/retry).

• Finish: call session/finish with questionsCount to finalize metadata.

5) API Contract (description – no code)

• POST /api/verify-token → Body { token } → { ok: true } or status 401.

• POST /api/session/start → Body { token, userName } → { ok: true, folder: "DD_MM_YYYY_HH_mm_ten_user" }.

• POST /api/upload-one (multipart/form-data, per question) → Fields: token, folder, questionIndex, video → { ok: true, savedAs: "Q<index>.webm" }.

• POST /api/session/finish → Body { token, folder, questionsCount } → { ok: true }.

• Notes: Create the folder at session/start; update metadata incrementally after each upload-one.

6) User Experience (UX), Errors and Safety

• Clear statuses: permission, recording, stopped, uploading, retry, done.

• Errors: invalid token, permission denied, network/size/MIME errors, missing fields → helpful messages with guidance.

• Retry with exponential backoff at least two to three times; provide a manual Retry button.

• Security decisions on the server; logs with ISO 8601 timestamps, timezone Asia/Bangkok.

7) Allowed Creativity (optional)

• Custom question set (≤ five), per-question timer, one-time re-record per question (define file naming rules).

• Progress bar, size warnings, per-question upload progress, remux/transcode after upload, webhook/email on Finish.

• (Bonus) Speech-to-Text (STT): choose engine, trigger timing, transcript.txt format labeled per question.

8) Acceptance Criteria

• Start: only a valid token proceeds to interview.

• Each question: show → record → Next → Q*.webm appears immediately on the server.

• Folder name matches DD_MM_YYYY_HH_mm_ten_user/ (timezone Asia/Bangkok).

• Metadata includes at least userName, uploadedAt (ISO 8601), timeZone, and list of received questions.

• Finish closes the session; no extra uploads.

• (Bonus) transcript.txt exists with per-question sections.

9) Deliverables

• Shared Git repository (all four students contribute).

• README: architecture and flow; API contract; run instructions; HTTPS requirement; file limits and retry policy; folder/file naming; (Bonus) Speech-to-Text.

• Screenshots: Start, recording, per-question upload status, server folder after each question, (Bonus) transcript.

• Demo video ≤ three minutes: demonstrate two to three questions end-to-end.

• PDF report (five–seven pages): objectives, design, security, testing, risks and mitigations, limitations and future work.

• Task allocation sheet: roles, contribution ratios, Pull Request (PR) and issue links as evidence.

10) Grading (100 points)

• Functionality (60): token and server-side validation (10); sequential and per-question recording (20); per-question upload with status and retry (20); storage and metadata (10).

• Technical and documentation (25): architecture, API, README/report (10); code quality, testing, Git workflow (10); basic UX and error messages (5).

• Bonus (15): per-question Speech-to-Text → transcript.txt (10); automated pipeline (5).

11) Five-Week Plan (suggested)

• Week 1 – Inception and Architecture: confirm core rules, flow, API contract, plan and repository.

• Week 2 – Core Frontend: per-question recording, state management, local prototype.

• Week 3 – Backend and Upload: session/start, upload-one, metadata, Frontend↔Backend integration, end-to-end for two to three questions.

• Week 4 – Reliability and UX: retry/backoff, file limits, UX polish, README v2 and screenshots.

• Week 5 – Bonus and Packaging: Speech-to-Text (optional), cross-browser tests, acceptance run, demo video, report, allocation sheet.

12) Academic Integrity

• All references must be cited; no turn-key templates. Plagiarism results in zero points.

13) Self-Check Questions (include in report)

• Why must the token be validated on the server?

• What are the benefits of per-question recording and upload?

• Why is HTTPS required publicly for camera/microphone permissions?

• How do you design retry/backoff and decide when to allow moving to the next question?

• (Bonus) Which Speech-to-Text engine and why? Accuracy/performance/cost trade-offs; per-question labeling.