Advanced Full-Stack Take-Home Assignment

Video Survey Platform with Face Detection

Time Limit: 72 Hours

Tech Stack: Next.js, FastAPI, SQLAlchemy, PostgreSQL/MySQL, Docker

---

1. Problem Statement

You are required to build a **privacy-first video survey platform** where users complete a 5-question Yes/No survey while their video is recorded. For each question:

- A short video segment is captured
- Face detection is performed
- A face visibility score is generated
- A snapshot of the detected face is stored
- The survey response is recorded

No personal identity information (name, email, phone, etc.) should be collected. However, system metadata must be captured.

---

2. High-Level Flow

Admin Flow

1. Create a new survey
2. Add exactly **5 Yes/No questions**
3. Publish the survey
4. Share the public URL:
   `/survey/{survey_id}`

User Flow

1. User opens the survey link
2. Grants camera permission
3. Completes 5 screens, each containing:
   - A Yes/No question
   - Live camera feed
   - Face detection feedback
   - Error handling
4. After submission, a completion screen is shown

---

3. Metadata Requirements (Mandatory)

For every survey submission, the system must capture:

Field	Source
IP address	Backend
Browser	User-Agent
Device type	User-Agent
OS	User-Agent
Timestamp	Server
Location	IP-based lookup

Do NOT collect name, email, phone, or any personal identifiers.

---

4. Functional Requirements

4.1 Frontend (Next.js)

Core

- Use browser camera APIs (`getUserMedia`)
- Live video preview
- Multi-step UI (5 screens)
- Yes/No buttons
- Progress indicator

Face Detection

- Must detect:
  - No face → error
  - Multiple faces → error
  - Single face → proceed
- Must assign a score (0–100)

  You may use a JS face detection library (e.g., face-api.js, MediaPipe, etc.)

4.2 Backend (FastAPI)

Survey Management

- Create surveys
- Add questions
- Publish surveys

Submission Flow

- Create submission
- Save answers
- Save face score per question
- Upload video segments
- Upload face snapshots

4.3 Media Storage

Type	Storage
Full survey video	Server filesystem
Face images	Server filesystem
Metadata	Database

Only file paths should be stored in the database.

---

5. Database Design (Minimum)

You must design normalized tables for:

Survey

```
id
title
is_active
created_at
```

SurveyQuestion

```
id
survey_id
question_text
order (1-5)
```

SurveySubmission

```
id
survey_id
ip_address
device
browser
```

```
os
location
started_at
completed_at
overall_score
```

SurveyAnswer

```
id
submission_id
question_id
answer (Yes/No)
face_detected
face_score
face_image_path
```

MediaFile

```
id
submission_id
type (video/image)
path
created_at
```

---

6. API Requirements (Minimum)

Survey APIs

```
POST   /api/surveys
POST   /api/surveys/{id}/questions
GET    /api/surveys/{id}
POST   /api/surveys/{id}/publish
```

Submission APIs

```
POST   /api/surveys/{id}/start
POST   /api/submissions/{id}/answers
POST   /api/submissions/{id}/media
POST   /api/submissions/{id}/complete
```

---

7. Export API (Mandatory)

`GET /api/submissions/{submission_id}/export`

Must return a ZIP containing:

```
/metadata.json
/videos/full_session.mp4
/images/q1_face.png
/images/q2_face.png
/images/q3_face.png
/images/q4_face.png
/images/q5_face.png
```

---

8. metadata.json Structure

```json
{
  "submission_id": "abc123",
  "survey_id": "survey001",
  "started_at": "2026-01-13T10:00:00Z",
  "completed_at": "2026-01-13T10:05:00Z",
  "ip_address": "1.2.3.4",
  "device": "Mobile",
  "browser": "Chrome",
  "os": "Android",
  "location": "India",
  "responses": [
    {
      "question": "Are you in a well-lit room?",
      "answer": "Yes",
      "face_detected": true,
      "score": 82,
      "face_image": "/images/q1.png"
    }
  ],
  "overall_score": 78
}
```

---

9. Docker Requirements

You must provide:

- Dockerfile (Frontend)
- Dockerfile (Backend)
- docker-compose.yml with:
  - Frontend
  - Backend
  - DB
  - Volume mounts for media

---

10. GitHub Submission Rules (Mandatory)

1. Code must be pushed to your **personal GitHub account**
2. Repository must be:
   - Public OR
   - Access granted to reviewers
3. Repository must contain:
   - Meaningful commits
   - Clear README
   - Setup instructions
   - Architecture overview
   - Trade-offs
   - Known limitations
