Backend Documentation

FastAPI Backend Implementation Guide

ChourangiHealth Video Consultation Platform

Version 1.0 | February 2026

1. Overview

The backend is a Python FastAPI application that acts as a secure middleware between the React frontend and the OpenVidu server. It handles authentication, session management, and recording operations. The backend runs on port 8000 on the EC2 host machine and is proxied through OpenVidu Nginx.

1.1 Why a Backend is Needed

- The OpenVidu admin secret cannot be exposed to the browser
- JWT authentication must be validated server-side
- Recording download must be proxied to avoid browser credentials popup
- Session creation logic (handling 409 conflicts) must be centralized

2. Project Structure

```
openvidu-fastapi-poc/
  app/
    main.py                  # FastAPI application entry point
    config.py                # Settings loaded from .env file
    dependencies.py          # JWT token verification dependency
    routers/
      auth.py                # POST /api/auth/login
      video.py               # All video and recording endpoints
    services/
      auth_service.py        # Password verification logic
      openvidu_service.py    # All OpenVidu REST API calls
    schemas/
      auth_schema.py         # Pydantic models for auth
      video_schema.py        # Pydantic models for video session
  .env                       # Environment variables (not in git)
  .gitignore                 # Excludes .env, __pycache__, *.pyc
  requirements.txt           # Python dependencies
```

3. Environment Variables

Create a .env file in the backend root with these values:

```
OPENVIDU_URL=http://localhost:5443
OPENVIDU_USERNAME=OPENVIDUAPP
OPENVIDU_SECRET=openvidu-v2-32
JWT_SECRET=supersecretjwtkey
JWT_ALGORITHM=HS256
```

NOTE: OPENVIDU_URL must point to localhost:5443 NOT the public domain. OpenVidu server listens on port 5443 HTTP internally. Calling via the public domain adds an extra SSL hop through Nginx which causes timing issues and 500 errors.

4. API Endpoints

4.1 Authentication

POST /api/auth/login

Authenticates a user and returns a JWT token.

Request body:

```
{ "username": "doctor", "password": "doctor123" }
```

Response:

```
{ "access_token": "eyJ...", "token_type": "bearer" }
```

The JWT token contains a role claim (admin for doctor, user for patient). The frontend reads this role using jwt-decode library and stores it in localStorage to control UI elements.

Test credentials:

- Doctor: username doctor, password doctor123, role admin
- Patient: username patient, password patient123, role user

4.2 Video Session

POST /api/video/session

Creates or retrieves an OpenVidu session and returns a connection token. Both doctor and patient call this endpoint with the same session ID (calling ID). The first call creates the session, the second call gets a 409 Conflict from OpenVidu which the backend handles gracefully by returning the existing session.

Request body:

```
{ "session_id": "1" }
```

Response:

```
{
  "sessionId": "1",
  "token": "wss://chourangi.duckdns.org?sessionId=1&token=...",
  "openviduUrl": "http://localhost:5443"
}
```

NOTE: The 409 Conflict handling is critical. Without it, when the second participant joins the session returns a 500 error and nobody can connect.

4.3 Recording Endpoints

POST /api/video/recording/start

Starts COMPOSED recording for a session. The session must be active with at least one publisher connected.

Query parameter: session_id (e.g. ?session_id=1)
Response: { "recordingId": "1", "status": "started" }

POST /api/video/recording/stop

Stops the active recording and makes the MP4 available for download.

Query parameter: recording_id (e.g. ?recording_id=1)
Response:

```
{
  "recordingId": "1",
  "status": "ready",
  "url": "https://chourangi.duckdns.org/openvidu/recordings/1/1.mp4"
}
```

GET /api/video/recording/{recording_id}

Gets the current status and URL of a recording.

Response: { "recordingId": "1", "status": "ready", "url": "..." }

GET /api/video/recording/{recording_id}/download

Proxies the MP4 file download through the backend. The backend fetches the file from OpenVidu using stored admin credentials and streams it to the browser with Content-Disposition: attachment header. This triggers a direct file download without requiring the user to enter OpenVidu credentials.

Response: MP4 binary stream with headers:
Content-Type: video/mp4
Content-Disposition: attachment; filename=recording_1.mp4

GET /health

Health check endpoint used to verify the backend is running.

Response: { "status": "ok" }

5. Key Service Implementation

5.1 openvidu_service.py

All OpenVidu REST API calls are in this service class. Every method uses requests library with HTTP Basic Auth (OPENVIDUAPP / secret) and verify=False to skip SSL verification for internal HTTP calls.

create_session method

Creates an OpenVidu session with the given custom session ID. If the session already exists OpenVidu returns 409 Conflict. The method handles this by returning the session ID directly instead of raising an error. This is essential for multi-participant rooms.

start_recording method

Calls the OpenVidu recording start API with COMPOSED output mode, BEST_FIT layout, 1280x720 resolution at 25fps. COMPOSED mode records all publisher streams into a single MP4 file in a grid layout. BEST_FIT layout automatically arranges streams in the most space-efficient grid.

download proxy

Uses the httpx library (not requests) because httpx supports streaming. The MP4 file is fetched from OpenVidu internally with admin credentials and returned as a FastAPI Response object with the full file content.

NOTE: httpx must be installed separately: pip install httpx --break-system-packages

6. Running the Backend

6.1 Local Development

```
cd openvidu-fastapi-poc
pip install -r requirements.txt
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Swagger API documentation available at: http://localhost:8000/docs

6.2 Production on EC2

```
cd /home/ubuntu/test-1/backend
pkill -f uvicorn
nohup uvicorn app.main:app --host 0.0.0.0 --port 8000 > /tmp/backend.log 2>&1 &
curl http://localhost:8000/health
```

6.3 View Backend Logs

```
tail -f /tmp/backend.log
```

6.4 Python Dependencies

```
fastapi
uvicorn
python-jose[cryptography]   # JWT handling
python-multipart            # Form data
requests                    # OpenVidu API calls
httpx                       # Recording download proxy streaming
python-dotenv               # .env file loading
pydantic                    # Request/response validation
```

7. Git Repository

- Repository URL: https://github.com/AshuuuPatil/openvidu_videocall_microservice_backend
- Branch: master
- .gitignore includes: .env, __pycache__/, *.pyc, *.pyo
- Never commit .env file as it contains secrets

7.1 Deploy Latest Code on EC2

```
cd /home/ubuntu/test-1/backend
git fetch origin master
git reset --hard origin/master
pkill -f uvicorn
nohup uvicorn app.main:app --host 0.0.0.0 --port 8000 > /tmp/backend.log 2>&1 &
curl http://localhost:8000/health
```

End of Backend Documentation