الجمهـوريـة العربيــة الســـورية

المعهد العالي للعلوم التطبيقية والتكنولوجيا

قسـم الشبكات ونظم التشغيل

العام الدراسي 2024/2025

**مشروع سنة رابعة**

تطبيق ويب لبث الوسائط المتعددة باستخدام QUIC و HTTP/3

تقديم

روان سلامة

إشراف

د.حسين خيو

ما.عمار مخلوف

ملخص

مع تزايد الطلب على محتوى الوسائط المتعددة عالي الجودة عبر الإنترنت، برزت الحاجة إلى حلول بث أكثر كفاءة وموثوقية للتغلب على تحديات زمن الوصول المرتفع والتقطيع المزعج (Buffering) التي تواجه البروتوكولات التقليدية. يهدف هذا المشروع إلى استعراض المزايا المتقدمة لبروتوكولي QUIC و HTTP/3 من خلال تصميم وتنفيذ تطبيق ويب متكامل لبث الفيديو.

تم بناء النظام وفق معمارية العميل-الخادم، حيث يعتمد الخادم على إطار العمل Quart ويعمل على خادم Hypercorn لدعم اتصالات QUIC/HTTP/3. تم تأمين النظام عبر تشفير كلمات المرور (Password Hashing) والمصادقة باستخدام رموز JWT. في جانب العميل، تم استخدام مكتبة HLS.js لتنفيذ تقنية البث التكيفي التي تتجاوب مع ظروف الشبكة.

Abstract

W

ith the growing demand for high-quality multimedia content online, the need has emerged for more efficient and reliable streaming solutions to overcome the challenges of high latency and annoying buffering faced by traditional protocols. This project aims to demonstrate the advanced advantages of the QUIC and HTTP/3 protocols by designing and implementing an integrated web application for video streaming.

The system was built based on a client-server architecture, where the server relies on the Quart framework and runs on a Hypercorn server to support QUIC/HTTP/3 connections. The system was secured through Password Hashing and authentication using JWT tokens. On the client-side, the HLS.js library was used to implement adaptive streaming technology that responds to network conditions.

جدول المحتويات

[ملخص 2](#_Toc205069992)

[Abstract 2](#_Toc205069993)

[الفصل الأول 5](#_Toc205069994)

[مقدمة عامة 5](#_Toc205069995)

[أهداف المشروع 6](#_Toc205069996)

[المتطلبات الوظيفية وغير الوظيفية 7](#_Toc205069997)

[المتطلبات الوظيفية (Functional Requirements) 7](#_Toc205069998)

[المتطلبات غير الوظيفية (Non-Functional Requirements) 8](#_Toc205069999)

[الفصل الثاني 9](#_Toc205070000)

[الفصل الثالث 9](#_Toc205070001)

[تقنيات جانب الخادم (Backend Technologies) 10](#_Toc205070002)

[تقنيات جانب العميل (Frontend Technologies) 12](#_Toc205070003)

[أدوات معالجة الفيديو (Video Processing Tools) 12](#_Toc205070004)

[أدوات الاختبار والتحقق (Testing & Verification Tools) 12](#_Toc205070005)

[الفصل الثالث 13](#_Toc205070006)

[حالات الاستخدام (Use Cases) 14](#_Toc205070007)

[الفصل الخامس 14](#_Toc205070008)

[تصميم بنية النظام 15](#_Toc205070009)

[مخطط معمارية النظام (System Architecture Diagram) 15](#_Toc205070010)

[المخطط المفاهيمي للنظام (Conceptual Class Diagram) 16](#_Toc205070011)

[مخططات التدفق للعمليات الرئيسية (Flowcharts) 17](#_Toc205070012)

[مخطط تدفق عملية تسجيل الدخول 17](#_Toc205070013)

[مخطط تدفق عملية تسجيل مستخدم جديد 18](#_Toc205070014)

[مخطط تدفق طلب فيديو محمي 19](#_Toc205070015)

[الفصل الثاني 20](#_Toc205070016)

[معمارية النظام المتبعة (System Architecture) 20](#_Toc205070017)

[النظام وطريقة التكامل 22](#_Toc205070018)

[خطة الاختبارات 24](#_Toc205070019)

الفصل الأول

مقدمة والتعريف بالمشروع والهدف منه

تتضمن هذه الفقرة مقدمة عامة حول المشروع مع توضيح الهدف منه

مقدمة عامة

منذ مطلع القرن الحادي والعشرين، شهد الإنترنت تحولاً جذرياً ليصبح جزءاً لا يتجزأ من الحياة اليومية، ومع هذا التحول، برز محتوى الفيديو كالشكل المهيمن على حركة مرور الشبكة العالمية. من منصات البث المباشر والأفلام حسب الطلب إلى مؤتمرات الفيديو والتعليم عن بعد، أصبحت جودة تجربة المستخدم في استهلاك الوسائط المتعددة عاملاً حاسماً.

إلا أن البنية التحتية التقليدية للويب، التي تعتمد بشكل أساسي على بروتوكول TCP، لم تُصمم في الأصل لتلبية متطلبات البث الفوري وعالي الجودة. هذا الأمر أدى إلى ظهور مشاكل متكررة يعاني منها المستخدمون، مثل التأخير الملحوظ عند بدء تشغيل الفيديو (Latency)، والتقطيع المزعج أثناء المشاهدة (Buffering)، خاصة عبر الشبكات غير المستقرة مثل شبكات الهاتف المحمول. هذه التحديات كشفت عن الحاجة الماسة لبروتوكولات شبكة أكثر تطوراً وكفاءة.

لمعالجة هذه المشاكل، ظهر الجيل الجديد من بروتوكولات الويب، وتحديداً QUIC و HTTP/3، والتي تم تصميمها من الألف إلى الياء لتكون أسرع وأكثر موثوقية. لذلك، قمنا في هذا المشروع ببناء تطبيق ويب متكامل لبث الفيديو يعتمد بشكل أساسي على هذين البروتوكولين. يهدف هذا التطبيق إلى استعراض المزايا العملية لهذه التقنيات الحديثة، من خلال توفير منصة تسمح للمستخدمين ببث الفيديو بشكل آمن وفعال، مع تطبيق تقنية البث التكيفي (Adaptive Streaming) لضمان أفضل تجربة مشاهدة ممكنة، وإثبات أن استخدام QUIC و HTTP/3 هو الحل الأمثل لتحديات بث الوسائط المتعددة في العصر الحديث.

أهداف المشروع

يهدف هذا البحث إلى تسليط الضوء على الجيل الجديد من بروتوكولات الشبكة QUIC و HTTP/3 وأهميتها في مجال تطبيقات الوسائط المتعددة، وذلك من خلال تحقيق مجموعة من الأهداف التقنية التي تشكل تطبيقاً وظيفياً آمناً. يبدأ تحقيق ذلك ببناء بيئة تطوير متكاملة قادرة على دعم وتشغيل تطبيق ويب يعتمد بشكل أساسي على اتصالات QUIC و HTTP/3 وضمن هذه البيئة، يتمثل الهدف في تطبيق آلية بث فعالة بمعدل بت متكيف Adaptive Bitrate Streaming باستخدام تقنية HLS، لضمان قدرة التطبيق على تقديم تجربة مشاهدة سلسة تتجاوب ديناميكياً مع ظروف الشبكة المتغيرة للمستخدم. ولحماية المحتوى وخصوصية المستخدمين، يتم تطوير منظومة أمان شاملة تتضمن تأمين بيانات الاعتماد عبر تشفير كلمات المرور (Password Hashing)، وتأمين جلسات الاستخدام من خلال نظام يعتمد على رموز الويب (JWT) للتحكم بالوصول. وأخيراً، يتم التحقق من وظائف النظام عبر اختبارات دقيقة للتأكد من أن جميع المكونات تعمل بشكل متكامل، مما يجعل النتيجة النهائية تطبيقاً ذا جودة وموثوقية عالية وقابل للنشر بأمان، ويكون بمثابة إثبات عملي لفعالية التقنيات المستخدمة.

المتطلبات الوظيفية وغير الوظيفية

المتطلبات الوظيفية (Functional Requirements)

هي الوظائف التي يجب على النظام تنفيذها.

إدارة المستخدمين (User Management):

* + يجب أن يسمح النظام للمستخدمين الجدد بإنشاء حساب عن طريق توفير اسم مستخدم وكلمة مرور فريدين.
  + يجب أن يوفر النظام واجهة لتسجيل دخول المستخدمين الحاليين.
  + يجب أن يتحقق النظام من صحة بيانات الدخول (اسم المستخدم وكلمة المرور).
  + يجب أن يمنع النظام إنشاء حسابات بأسماء مستخدمين مكررة.

التحكم بالوصول والمصادقة (Access Control & Authentication):

* + يجب على النظام إصدار توكن مصادقة (JWT) للمستخدم بعد تسجيل الدخول بنجاح.
  + يجب أن يكون الوصول إلى محتوى الفيديو مقتصراً على المستخدمين الذين يمتلكون توكن مصادقة صالحاً.
  + يجب على النظام رفض أي طلب للوصول إلى محتوى الفيديو لا يحتوي على توكن صالح.

بث الفيديو (Video Streaming):

* + يجب أن يعرض النظام قائمة بالفيديوهات المتاحة للمستخدم بعد تسجيل الدخول.
  + يجب أن يبدأ النظام ببث الفيديو الذي يختاره المستخدم.
  + يجب أن يدعم النظام تقنية البث التكيفي (Adaptive Bitrate Streaming)، حيث يتم تعديل جودة الفيديو تلقائياً لتتناسب مع سرعة شبكة المستخدم.
  + يجب أن يسمح النظام للمستخدم بتجاوز البث التكيفي واختيار جودة الفيديو المطلوبة يدوياً من قائمة الجودات المتاحة.

المتطلبات غير الوظيفية (Non-Functional Requirements)

هي الخصائص ومعايير الجودة التي يجب أن يمتلكها النظام.

الأداء (Performance):

* + يجب أن يتميز النظام بزمن استجابة منخفض عند طلب بدء تشغيل الفيديو (Low Startup Time) .
  + يجب أن يقلل النظام من حدوث التقطيع (Buffering) أثناء المشاهدة، حتى في ظل ظروف الشبكة غير المثالية.
  + يجب أن يتم الاتصال بين العميل والخادم بشكل أساسي عبر بروتوكول QUIC / HTTP/3 لتحقيق الأهداف المذكورة .
  + يجب أن يكون الخادم قادراً على التعامل مع عدد معقول من طلبات البث المتزامنة دون تدهور كبير في الأداء.

الأمان (Security):

* + يجب تخزين جميع كلمات مرور المستخدمين في قاعدة البيانات بشكل مشفر (Hashed) باستخدام خوارزمية آمنة مثل bcrypt، وليس كنص عادي.
  + يجب أن تكون جميع الاتصالات بين العميل والخادم مؤمنة باستخدام بروتوكول TLS المدمج ضمنياً في QUIC)).
  + يجب أن يكون لتوكنات المصادقة (JWT) تاريخ انتهاء صلاحية محدد لتقليل مخاطر إعادة الاستخدام في حال تسريبها.

سهولة الاستخدام (Usability):

* + يجب أن تكون واجهة المستخدم لتسجيل الدخول وإنشاء الحساب واضحة وسهلة الفهم.
  + يجب أن تكون عناصر التحكم في مشغل الفيديو (تشغيل، إيقاف، اختيار الجودة) بديهية وسهلة الوصول.

الموثوقية (Reliability):

* + يجب أن يعمل النظام بشكل مستقر وأن يتعافى من الأخطاء الشائعة (مثل فشل الاتصال المؤقت) دون أن يتعطل بالكامل.

الفصل الثاني

الدراسة النظرية

الفصل الثالث

الأدوات والتقنيات المستخدمة

يعرض هذا الفصل شرحاً تفصيلياً للأدوات والتقنيات التي تم الاعتماد عليها في تطوير هذا المشروع، مع توضيح سبب اختيار كل تقنية والدور الذي تلعبه في بنية النظام.

تقنيات جانب الخادم (Backend Technologies)

Python

تم اختيار لغة البرمجة بايثون (Python) كلغة أساسية لتطوير الجانب الخادم من التطبيق، وذلك لعدة أسباب جوهرية. تشتهر بايثون ببساطة قواعدها وسهولة قراءتها، مما يسرّع من عملية التطوير ويجعل صيانة الكود أكثر فعالية. الأهم من ذلك، تمتلك بايثون نظاماً بيئياً (Ecosystem) ضخماً وغنياً بالمكتبات وأطر العمل المتخصصة في تطوير الويب والشبكات، مثل Quart و Hypercorn، والتي كانت ضرورية لتحقيق الأهداف التقنية للمشروع، خاصة فيما يتعلق بدعم البرمجة غير المتزامنة وبروتوكولات الويب الحديثة.

venv

بيئة العمل الافتراضية لضمان تنظيم المشروع وعزل تبعياته (Dependencies) عن أي مشاريع أخرى على نظام التشغيل، تم استخدام أداة venv المدمجة مع بايثون. تقوم venv بإنشاء بيئة عمل افتراضية، وهي عبارة عن مجلد معزول يحتوي على نسخة خاصة من مفسر بايثون وجميع المكتبات التي يحتاجها المشروع فقط. هذا الأسلوب يمنع تعارض الإصدارات بين المكتبات المختلفة ويجعل المشروع قابلاً للنقل والتشغيل بسهولة على أي جهاز آخر.

Quart

تم اختيار Quart كإطار عمل لتطوير تطبيق الويب. Quart هو إطار عمل بايثون حديث متوافق مع معيار ASGI (Asynchronous Server Gateway Interface)، وهو مصمم خصيصاً لدعم البرمجة غير المتزامنة باستخدام async و await تم اختياره لأنه يوفر واجهة برمجية (API) شبيهة جداً بإطار العمل الشهير Flask، ولكنه يضيف القدرة على التعامل مع عمليات الإدخال والإخراج (I/O) بشكل غير متزامن. هذه الميزة حيوية لتطبيقات البث، حيث يحتاج الخادم إلى التعامل مع عدد كبير من الاتصالات طويلة الأمد وخدمة ملفات الفيديو بكفاءة دون أن يتم حظر (Blocking) العمليات الأخرى.

Hypercorn

لتشغيل تطبيق Quart وخدمته للعملاء، تم اختيار خادم Hypercorn و هو خادم ويب متوافق مع معيار ASGI، ويتميز بدعمه المدمج لمجموعة واسعة من البروتوكولات الحديثة. تم اختياره كعنصر أساسي في هذا المشروع لقدرته على تفعيل بروتوكول QUIC و HTTP/3 بسهولة من خلال ملف الإعدادات. هذا الدعم الأصلي هو الذي مكّن التطبيق من الاستفادة من مزايا السرعة والموثوقية التي تقدمها هذه البروتوكولات، مما يحقق الهدف الرئيسي للبحث.

PyJWT

لتطبيق نظام مصادقة آمن ومستقل (Stateless)، تم استخدام مكتبة PyJWT هذه المكتبة توفر تطبيقاً كاملاً لمعيار رموز الويب المميزة (JSON Web Tokens) تم استخدامها في الخادم لتنفيذ وظيفتين رئيسيتين: أولاً، إنشاء (Encoding) توكن جديد وتوقيعه بمفتاح سري بعد نجاح عملية تسجيل دخول المستخدم. ثانياً، التحقق من (Decoding) صحة التوكنات القادمة مع طلبات الوصول إلى الفيديو، وذلك للتأكد من أن التوكن لم يتم التلاعب به وأنه لم تنتهِ صلاحيته.

bcrypt

لتحقيق أعلى مستوى من الأمان لبيانات اعتماد المستخدمين، تم استخدام مكتبة bcrypt لتشفير كلمات المرور. تعتبر bcrypt واحدة من أقوى وأشهر خوارزميات التشفير (Hashing) المخصصة لكلمات المرور. بدلاً من تخزين كلمات المرور كنص عادي في قاعدة البيانات، وهو ما يمثل ثغرة أمنية خطيرة، تقوم bcrypt بتحويل كلمة المرور إلى سلسلة نصية مشفرة فريدة ومعقدة لا يمكن عكسها. تم استخدامها في وظيفتين: bcrypt.hashpw لإنشاء النسخة المشفرة عند تسجيل مستخدم جديد، و bcrypt.checkpw لمقارنة كلمة المرور التي يدخلها المستخدم مع النسخة المشفرة المحفوظة أثناء عملية تسجيل الدخول.

SQLite

لتخزين بيانات المستخدمين بشكل دائم، تم اختيار نظام إدارة قواعد البيانات SQLite يتميز SQLite بأنه نظام خفيف الوزن، لا يتطلب خادماً منفصلاً (Serverless)، ويقوم بتخزين قاعدة البيانات بالكامل في ملف واحد (.db) ضمن مجلد المشروع. تم اختياره لسهولة إعداده واستخدامه، حيث أنه مدعوم بشكل أصلي في بايثون عبر مكتبة sqlite3، مما يجعله خياراً مثالياً للتطبيقات التي لا تتطلب عمليات معقدة على قواعد البيانات أو التعامل مع عدد هائل من المستخدمين المتزامنين.

SSL/TLS و OpenSSL

لتفعيل بروتوكول QUIC/HTTP/3، والذي يتطلب اتصالاً آمناً (HTTPS)، تم استخدام أداة سطر الأوامر OpenSSL لإنشاء شهادات SSL/TLS موقعة ذاتياً (Self-Signed Certificates) تم استخدام هذه الشهادات cert.crt و key.pem لتكوين خادم Hypercorn لخدمة التطبيق عبر قناة مشفرة، وهو شرط أساسي لعمل البروتوكولات الحديثة.

تقنيات جانب العميل (Frontend Technologies)

HTML, CSS, JavaScript

تم بناء الواجهة الأمامية التفاعلية باستخدام التقنيات الأساسية للويب. استُخدم HTML لهيكلة المحتوى، و CSS لتطبيق الأنماط البصرية والتصميم الجذاب، و JavaScript لإدارة تفاعل المستخدم، والتعامل مع نماذج تسجيل الدخول، والتواصل مع الخادم، وتشغيل منطق مشغل الفيديو.

HLS.js

لتشغيل الفيديو ودعم البث التكيفي، تم استخدام مكتبة HLS.js هذه المكتبة المكتوبة بلغة JavaScript مسؤولة عن تحميل ملفات قائمة التشغيل (.m3u8) ومقاطع الفيديو (.ts)، وقياس سرعة الشبكة، والتبديل بين الجودات المختلفة بسلاسة لضمان أفضل تجربة مشاهدة ممكنة.

أدوات معالجة الفيديو (Video Processing Tools)

FFmpeg

لتحويل ملفات الفيديو المصدر إلى صيغة مناسبة للبث التكيفي، تم استخدام أداة سطر الأوامر القوية FFmpeg تم استخدامها لترميز الفيديو بنسخ متعددة بجودات مختلفة 720p و p144، ثم تقطيع كل نسخة إلى مقاطع صغيرة بصيغة .ts، وإنشاء ملفات قائمة التشغيل الرئيسية (.m3u8) التي تحتاجها تقنية HLS .

**أدوات الاختبار والتحقق** (Testing & Verification Tools)

Wireshark

للتأكد بشكل قاطع من أن الاتصال بين العميل والخادم يتم بالفعل عبر QUIC، تم استخدام محلل بروتوكولات الشبكة Wireshark من خلال مراقبة حركة مرور الشبكة وتطبيق فلتر quic، تم التحقق من أن حزم البيانات تستخدم هذا البروتوكول الحديث.

Clumsy

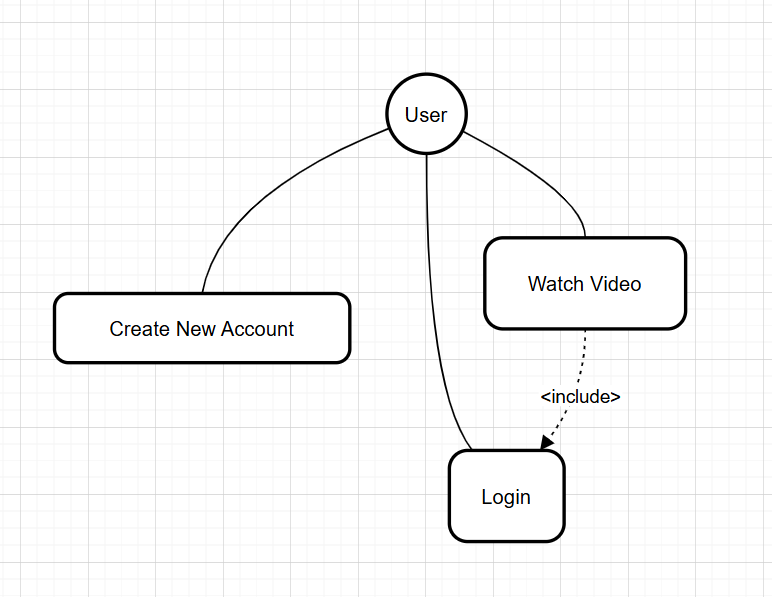
لمحاكاة ظروف الشبكة السيئة واختبار مدى استجابة البث التكيفي، تم استخدام أداة Clumsy على نظام ويندوز. سمحت هذه الأداة بإدخال تأخير (Latency) متحكم به على الشبكة، مما أتاح مراقبة كيفية تحول المشغل (hls.js) إلى الجودة المنخفضة عند تدهور أداء الشبكة.

الفصل الرابع

الدراسة التحليلية

تقدم هذه الفقره آلية تفاعل النظام مع المستخدم

حالات الاستخدام (Use Cases)



الصورة 1 Use Case Diagram

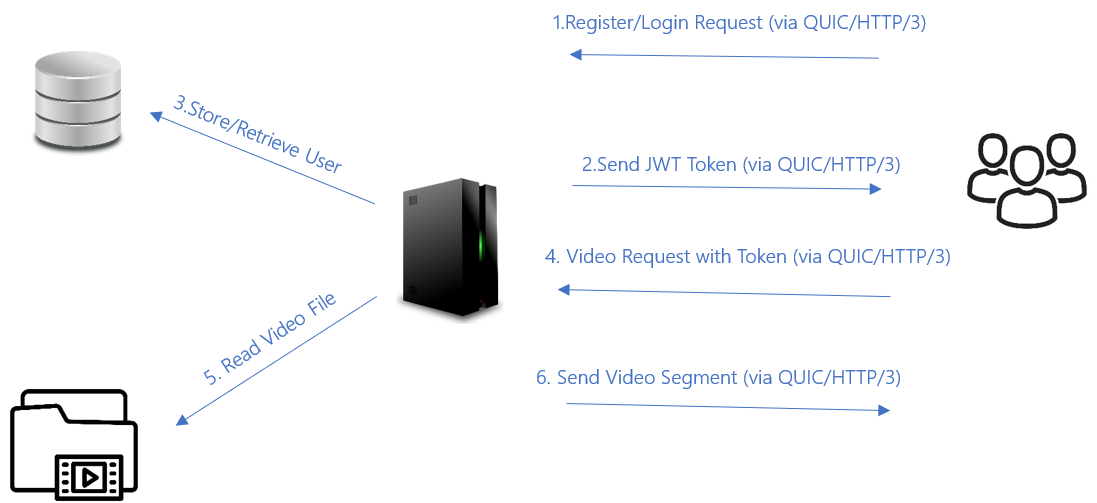
الفصل الخامس

الدراسة التصميمية

تقدم هذه الفقره البنى التي يتكون منها النظام آليه ربطها مع بعضها

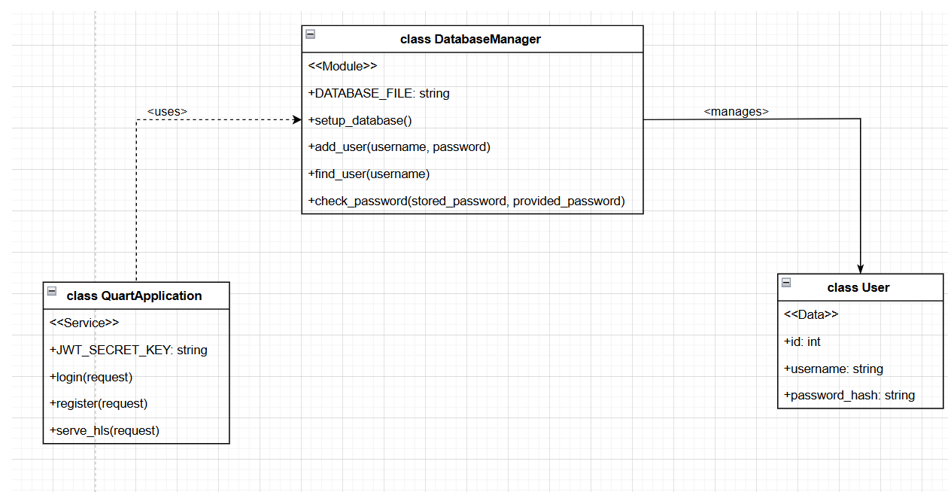
تصميم بنية النظام

مخطط معمارية النظام (System Architecture Diagram)



الصورة 2 مخطط معمارية النظام

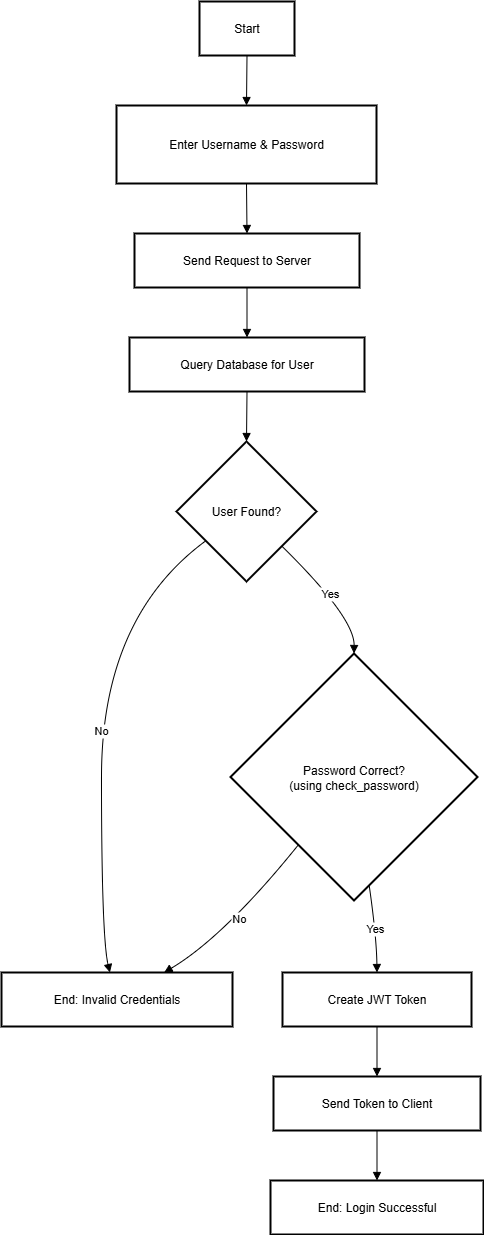
المخطط المفاهيمي للنظام (Conceptual Class Diagram)



الصورة 3 المخطط المفاهيمي للنظام

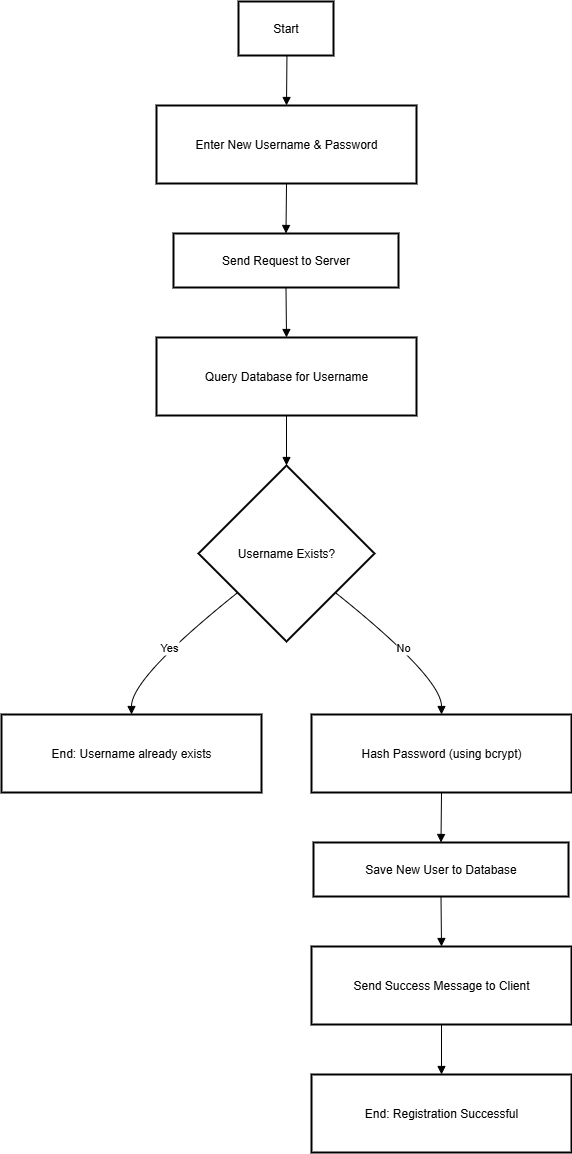
مخططات التدفق للعمليات الرئيسية (Flowcharts)

مخطط تدفق عملية تسجيل الدخول

e

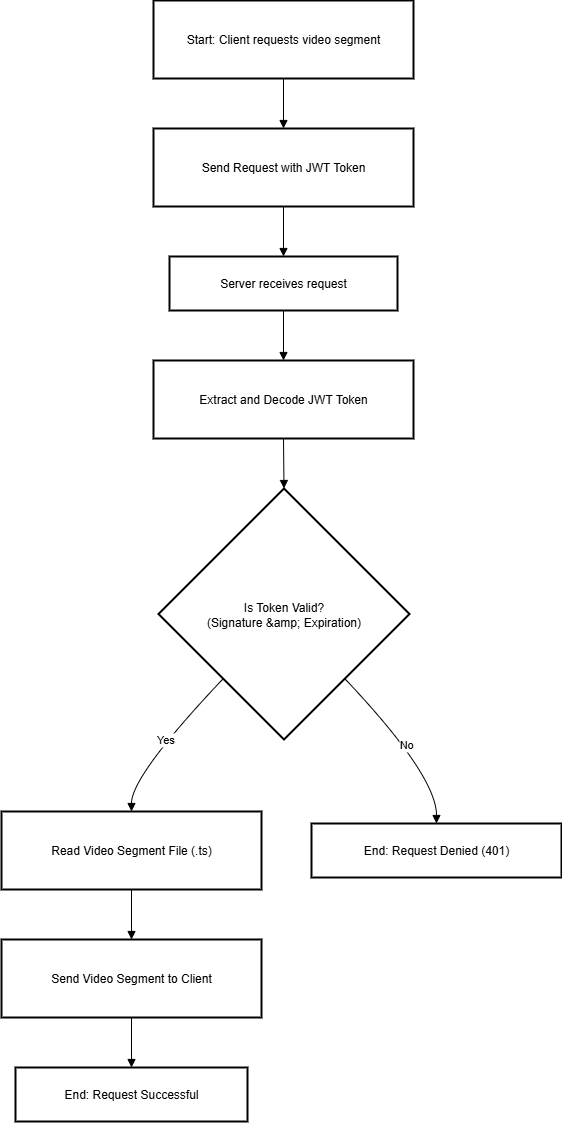
الصورة 4 مخطط تدفق عملية تسجيل الدخول

مخطط تدفق عملية تسجيل مستخدم جديد



الصورة 5 مخطط تدفق عملية تسجيل مستخدم جديد

مخطط تدفق طلب فيديو محمي



الصورة 6 مخطط تدفق طلب فيديو محمي

الفصل السادس

التنفيذ والاختبارات

يصف هذا الفصل بنية فصل نموذجي في التقرير من حيث تقسيمه إلى فقرات وتنسيقها وترقيمها.

معمارية النظام المتبعة (System Architecture)

تم تصميم النظام وتطويره بالاعتماد على معمارية العميل-الخادم (Client-Server Architecture)، وهي بنية قياسية تتيح فصل الواجهات الأمامية التي يتفاعل معها المستخدم عن المنطق البرمجي ومعالجة البيانات التي تتم في الخادم. تتكون هذه المعمارية من جزأين رئيسيين:

* **العميل (Client-Side)**

تم بناء العميل كواجهة ويب من صفحة واحدة (Single-Page Interface) تعمل بالكامل داخل متصفح المستخدم. تعتمد هذه المعمارية على فصل منطق العرض عن منطق العمل، وتتكون من الوحدات التالية:

* وحدة واجهة المستخدم (UI Module)

تم بناؤها باستخدام HTML لهيكلة المحتوى (مثل نماذج الإدخال ومشغل الفيديو) و CSS لتطبيق الأنماط البصرية والتصميم الجذاب. هذه الوحدة مسؤولة عن كل ما يراه المستخدم على الشاشة.

* وحدة إدارة التفاعل والمصادقة (Interaction & Auth Module)

هذه الوحدة مكتوبة بلغة JavaScript وهي مسؤولة عن التقاط تفاعلات المستخدم مثل إدخال البيانات في نماذج التسجيل والدخول، وإرسال هذه البيانات إلى الخادم عبر طلبات fetch غير متزامنة، واستقبال الاستجابة من الخادم مثل استلام توكن JWT أو رسائل الخطأ وعرضها للمستخدم.

* وحدة مشغل الفيديو (Video Player Module)

تتكون من عنصر الفيديو (<video>) في HTML ومكتبة HLS.js هذه الوحدة هي المسؤولة عن منطق البث بالكامل في جانب العميل. تقوم hls.js بتحميل ملف قائمة التشغيل (.m3u8)، وقياس سرعة الشبكة، وطلب مقاطع الفيديو (.ts) بالجودة المناسبة، وتمريرها إلى عنصر الفيديو لعرضها. كما أنها تتكامل مع وحدة المصادقة لإرسال توكن JWT مع كل طلب فيديو.

* **الخادم (Server-Side)**

تم تصميم الخادم كتطبيق ويب غير متزامن (Asynchronous) يعتمد على معيار ASGI، مما يجعله فعالاً في التعامل مع عمليات البث طويلة الأمد. تتكون بنيته من عدة طبقات منطقية:

* + طبقة التوجيه ونقاط النهاية (Routing & Endpoint Layer)

تم تنفيذها باستخدام إطار العمل Quart هذه الطبقة مسؤولة عن تعريف المسارات (Routes) التي يستجيب لها الخادم مثل /login, /register, /hls/<filename> وربط كل مسار بالوظيفة المناسبة له.

* + طبقة المصادقة والتحكم بالوصول (Authentication & Access Control Layer)

هذه الطبقة هي المسؤولة عن أمان النظام. عند وصول طلب إلى نقطة نهاية /login، تقوم هذه الطبقة بالتواصل مع طبقة قاعدة البيانات للتحقق من بيانات الاعتماد، ثم تستخدم مكتبة PyJWT لإنشاء توكن صالح. أما عند وصول طلب إلى نقطة النهاية المحمية /hls ، فتقوم هذه الطبقة باستخلاص التوكن والتحقق من صلاحيته قبل السماح للطلب بالوصول إلى طبقة خدمة الملفات.

* + طبقة التفاعل مع قاعدة البيانات (Database Interaction Layer)

تم عزل كل عمليات قاعدة البيانات في وحدة منفصلة (DB.py). هذه الطبقة مسؤولة حصراً عن التواصل مع قاعدة بيانات SQLite وهي تحتوي على وظائف لإضافة مستخدم جديد، والبحث عن مستخدم، والتحقق من كلمة المرور المشفرة باستخدام bcrypt هذا الفصل يضمن أن بقية أجزاء التطبيق لا تحتاج إلى معرفة تفاصيل كيفية تخزين البيانات.

* + طبقة خدمة الملفات (File Serving Layer)

هذه الطبقة مسؤولة عن إرسال الملفات إلى العميل. يتم استخدام وظائف Quart المدمجة لخدمة الملفات الثابتة مثل player1.html ، وكذلك لخدمة ملفات الفيديو .m3u8 و .ts من مجلد hls\_output بعد أن تتأكد طبقة التحكم بالوصول من صلاحية الطلب.

النظام وطريقة التكامل

يتكون النظام من عدة وحدات برمجية تعمل معاً بشكل متكامل لتقديم تجربة آمنة وفعالة. والتي هي :

* وحدة المصادقة (Authentication Module) :

تعتبر هذه الوحدة حجر الزاوية في أمان النظام، حيث أنها مسؤولة عن التحقق من هوية المستخدمين والتحكم في صلاحيات وصولهم إلى الموارد المحمية، وتحديداً محتوى الفيديو. تم بناء هذه الوحدة لتكون قوية وآمنة من خلال تطبيق آليات تشفير ومصادقة حديثة. تبدأ العملية من خلال نقطتي النهاية (Endpoints) الرئيسيتين في الخادم /register و /login .

* + عملية تسجيل مستخدم جديد (/register)

عندما يطلب مستخدم جديد إنشاء حساب، يستقبل الخادم البيانات التي تم إدخالها (اسم المستخدم وكلمة المرور). قبل تخزين هذه المعلومات، يتم تنفيذ خطوة أمنية حاسمة وهي تشفير كلمة المرور باستخدام مكتبة bcrypt تقوم هذه المكتبة بتطبيق خوارزمية تشفير (Hashing) قوية ومعقدة، مع إضافة قيمة عشوائية (Salt) لكل كلمة مرور لجعلها فريدة. ينتج عن هذه العملية سلسلة نصية مشفرة (Hash) لا يمكن عكسها أو فك تشفيرها للوصول إلى كلمة المرور الأصلية. هذا الإجراء يضمن أنه حتى في أسوأ السيناريوهات، مثل اختراق قاعدة البيانات، تبقى كلمات المرور الحقيقية للمستخدمين محمية وغير قابلة للكشف.

* + عملية تسجيل الدخول (/login)

عندما يحاول مستخدم مسجل مسبقاً تسجيل الدخول، يقوم الخادم أولاً بالبحث في قاعدة البيانات عن سجل يطابق اسم المستخدم المدخل. إذا تم العثور على المستخدم، تنتقل العملية إلى الخطوة الثانية وهي التحقق من صحة كلمة المرور. يتم ذلك من خلال مقارنة كلمة المرور التي أدخلها المستخدم مع النسخة المشفرة المحفوظة في قاعدة البيانات، وذلك باستخدام دالة bcrypt.checkpw. تقوم هذه الدالة بإعادة تشفير الكلمة المدخلة بنفس الطريقة ومقارنة الناتج مع القيمة المخزنة. إذا تطابقت القيمتان، تعتبر عملية المصادقة ناجحة.

* + إنشاء وإدارة التوكن (JWT Generation)

بعد نجاح عملية المصادقة، يقوم الخادم بإنشاء رمز ويب مميز (JSON Web Token - JWT). هذا التوكن هو عبارة عن هوية رقمية مؤقتة للمستخدم، وهو عبارة عن سلسلة نصية مشفرة تتكون من ثلاثة أجزاء:

الترويسة (Header): تحتوي على معلومات عن نوع التوكن وخوارزمية التشفير.

الحمولة (Payload): تحتوي على معلومات هوية المستخدم (مثل اسم المستخدم) وتاريخ انتهاء صلاحية التوكن.

التوقيع (Signature): يتم إنشاؤه عن طريق تشفير الترويسة والحمولة مع مفتاح سري خاص بالخادم. هذا التوقيع يضمن أن التوكن لم يتم التلاعب به.

يتم إرسال هذا التوكن إلى العميل، الذي يقوم بتخزينه واستخدامه في جميع الطلبات اللاحقة للوصول إلى الموارد المحمية (مثل طلبات الفيديو)، مما يتيح نظام مصادقة فعال ومستقل (Stateless).

* وحدة البث المحمي (Secure Streaming Module) :

تعتبر هذه الوحدة هي المسؤولة عن ضمان أن محتوى الفيديو لا يمكن الوصول إليه إلا من قبل المستخدمين المصادق عليهم، مما يشكل طبقة حماية أساسية للمحتوى. تعمل هذه الوحدة من خلال تكامل وثيق بين منطق الحماية في الخادم ومنطق إرسال الطلبات في العميل.

* + **آلية الحماية في الخادم**

في جانب الخادم، يتم تأمين جميع ملفات البث ملفات قائمة التشغيل .m3u8 ومقاطع الفيديو .ts من خلال خدمتها عبر مسار محمي ومحدد هو /hls/<path:filename> عند وصول أي طلب إلى هذا المسار، يتم تنفيذ سلسلة من خطوات التحقق الأمنية قبل إرسال أي بيانات:

استخلاص التوكن: يقوم الخادم أولاً بفحص ترويسة الطلب Authorization لاستخلاص توكن JWT الذي أرسله العميل.

التحقق من الصحة: بعد استخلاص التوكن، يقوم الخادم بالتحقق من صحته من خلال فك تشفيره باستخدام المفتاح السري الخاص بالخادم. تشمل عملية التحقق التأكد من صحة التوقيع (لمنع التلاعب) والتحقق من تاريخ انتهاء صلاحية التوكن.

اتخاذ القرار:

في حالة النجاح: إذا كان التوكن صالحاً، يسمح الخادم للطلب بالمرور ويقوم بقراءة الملف المطلوب (سواء كان .m3u8 أو .ts) من نظام الملفات وإرساله إلى العميل.

في حالة الفشل: إذا كان التوكن غير صالح، أو مفقوداً، أو منتهي الصلاحية، يتم رفض الطلب فوراً، ويرسل الخادم رمز الحالة 401 Unauthorized إلى العميل، مما يمنعه من الوصول إلى المحتوى.

* + **التكامل في العميل**

في جانب العميل، تم تحقيق التكامل مع هذه الآلية الأمنية من خلال إعداد مكتبة hls.js بشكل مخصص. تم استخدام وظيفة xhrSetup، وهي النقطة الرئيسية التي تربط بين الواجهة الأمامية والخادم. هذه الوظيفة تعترض تلقائياً كل طلب يتم إرساله من قبل المشغل لجلب مقطع فيديو. قبل إرسال الطلب، تقوم بإرفاق توكن JWT الذي تم استلامه أثناء تسجيل الدخول إلى ترويسة Authorization. بهذه الطريقة، يتم تأمين كل جزء من عملية البث، مما يضمن أن جميع طلبات الفيديو التي تصل إلى الخادم تكون مصادق عليها، ويخلق تكاملاً قوياً بين وحدة المصادقة ووحدة البث.

خطة الاختبارات

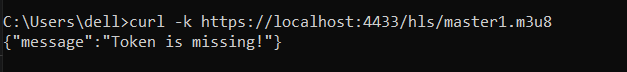
* اختبارات التكامل (Integration Tests)
  + الهدف الرئيسي من اختبارات التكامل هو التأكد من أن جميع الوحدات والمكونات المختلفة في النظام تعمل معاً بشكل متناغم ومتكامل كما هو مصمم لها. يركز هذا الاختبار على تدفق البيانات والتحكم بين الوحدات الرئيسية: وحدة المصادقة (Authentication Module)، وحدة التحكم بالوصول وحماية المسارات (Access Control)، ووحدة البث المحمي (Secure Streaming Module)، للتأكد من عدم وجود أي تعارض أو أخطاء عند تفاعلها معاً.
  + تم تصميم سيناريو اختبار شامل يحاكي رحلة المستخدم الكاملة عبر الوظائف الأساسية للنظام، وذلك للتحقق من أن كل خطوة تسلم التحكم إلى الخطوة التالية بسلاسة. يتكون السيناريو من الخطوات التالية:

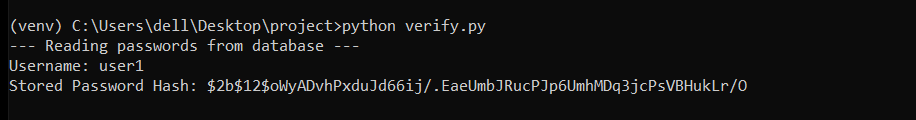
إنشاء حساب جديد: يبدأ المستخدم بالوصول إلى واجهة التسجيل وإنشاء حساب جديد عن طريق إدخال اسم مستخدم وكلمة مرور.

تسجيل الدخول: بعد إنشاء الحساب بنجاح، ينتقل المستخدم إلى واجهة تسجيل الدخول ويقوم بإدخال نفس بيانات الاعتماد التي قام بتسجيلها.

الوصول إلى المحتوى المحمي: بعد نجاح عملية تسجيل الدخول، يحاول المستخدم تشغيل فيديو محمي من القائمة المتاحة.

* + عند تنفيذ سيناريو الاختبار، تمت جميع الخطوات بنجاح كما هو متوقع. تمكن المستخدم من إنشاء الحساب، وتسجيل الدخول، وبدأ تشغيل الفيديو دون أي مشاكل. هذا يثبت أن التكامل بين الوحدات يعمل بشكل صحيح: فوحدة التسجيل (/register) نجحت في حفظ المستخدم، ووحدة تسجيل الدخول (/login) نجحت في التحقق منه وإصدار توكن JWT صالح، ووحدة البث المحمي (/hls) نجحت في التحقق من هذا التوكن والسماح بالوصول إلى محتوى الفيديو.
* الاختبارات الأمنية (Security Tests):
  + الهدف من هذه الاختبارات هو التحقق من فعالية وقوة الإجراءات الأمنية المطبقة في النظام، وتحديداً حماية بيانات اعتماد المستخدمين وحماية محتوى الفيديو من الوصول غير المصرح به.
  + سيناريو الاختبار 1: الوصول إلى الموارد المحمية بدون مصادقة
  + تم محاولة الوصول المباشر إلى رابط أحد ملفات الفيديو المحمية ملف .m3u8 خارج نطاق التطبيق. تم استخدام أداة سطر الأوامر curl لإرسال طلب HTTP مباشر إلى نقطة النهاية /hls/master1.m3u8 دون إرفاق ترويسة Authorization التي تحتوي على توكن JWT .
  + رفض الخادم الطلب وأعاد رمز الحالة 401 Unauthorized، مما يؤكد أن المحتوى محمي بشكل فعال ضد الوصول المباشر وغير المصادق عليه.



* + سيناريو الاختبار 2: التحقق من تشفير كلمات المرور
  + إجراء فحص مباشر لملف قاعدة البيانات users.db للتحقق من كيفية تخزين بيانات اعتماد المستخدمين. تم استخدام سكريبت بايثون مخصص (verify.py) لقراءة محتويات جدول users مباشرة من الملف وطباعة كلمات المرور المخزنة في سطر الأوامر بعد إنشاء حساب مستخدم جديد.
  + أظهرت نتيجة تشغيل السكريبت أن جميع كلمات المرور المخزنة كانت بالفعل عبارة عن سلاسل نصية مشفرة تبدأ بـ $2b$ هذا يؤكد أن بيانات المستخدمين محمية بشكل قوي، وأنه حتى في حالة تسريب قاعدة البيانات، لا يمكن الكشف عن كلمات المرور الأصلية.
* التحقق من وظائف الأداء (Performance Functionality Verification)
* الهدف من هذا القسم هو إثبات أن الميزات الأساسية المتعلقة بالأداء، والتي تم بناؤها في النظام، تعمل كما هو مصمم لها. يركز التحقق على نقطتين محوريتين: التفعيل الناجح لبروتوكول QUIC/HTTP/3، والاستجابة الصحيحة لآلية البث التكيفي عند تغير ظروف الشبكة.
* التحقق من اتصال QUIC/HTTP/3

تم استخدام أدوات تحليل الشبكة، وتحديداً برنامج Wireshark، لمراقبة حركة مرور الشبكة أثناء تشغيل التطبيق. من خلال تطبيق فلتر العرض quic، تم عزل جميع الحزم المتعلقة بالبروتوكول، مما أتاح تأكيداً قاطعاً بأن الاتصال بين العميل والخادم يتم عبر QUIC .

أثبتت كلتا الأداتين أن إعداد خادم Hypercorn كان ناجحاً، وأن التطبيق يتواصل بالفعل عبر بروتوكولات الويب الحديثة.

* التحقق من البث التكيفي (Adaptive Streaming)

للتحقق من أن منطق البث التكيفي يعمل بشكل صحيح، تم محاكاة ظروف شبكة سيئة بشكل مصطنع. تم استخدام أداة محاكاة الشبكة Clumsy على نظام ويندوز لإدخال تأخير (Latency) متحكم به على جميع حزم الشبكة الصادرة والواردة من جهاز الاختبار. تم ضبط الأداة لإضافة تأخير بمقدار 150 ميلي ثانية، وهو ما يكفي لمحاكاة شبكة بطيئة ولكنها لا تزال قابلة للاستخدام.

عند تشغيل الفيديو في الظروف العادية (بدون تأخير)، لوحظ أن مشغل الفيديو (hls.js) يختار الجودة الأعلى (720p)وعند تفعيل التأخير باستخدام Clumsy ، قام المشغل بشكل فوري باختيار وبث مقاطع الفيديو ذات الجودة المنخفضة (144p)هذا السلوك يؤكد أن المشغل نجح في استشعار تدهور أداء الشبكة واتخذ القرار الصحيح لضمان استمرارية البث وتجنب التقطيع.