HIAST

المعهد العالي للعلوم االتطبيقية والتكنولوجيا

الجمهورية العربية السورية

قسم المعلوميات

العام الدراسي 2026/2025

نظام حفظ واسترجاع وحدات تعلم الطالب في المعهد العالي باستخدام تقنية الويب3

مشروع سنة رابعة

إعداد:

زينب يونس علي

إشراف:

د. مصطفى دقاق

م. عمار مخلوف

12/8/2025

كلمة شكر

أتقدم بالشكر إلى كل من ساهم في هذا العمل,

وأخص بالشكر الدكتور مصطفى دقاق على دعمه الدائم ,

والمهندس عمار مخلوف على نصائحه و متابعته طوال مدة المشروع .

الخلاصة

يقدم هذا المشروع تطبيقاً لامركزياً لإدارة وتوثيق السجلات الأكاديمية الجامعية باستخدام تقنية البلوك تشين. الواجهة الخلفية للنظام هي عقد ذكي بلغة Solidity يوفر تخزيناً آمناً وغير قابل للتغيير لبيانات الطلاب، الخطط الدراسية، والنتائج. تفصل معمارية النظام بين عمليات الكتابة الإدارية، التي تتم عبر سكربت وسيط آمن، وبين الواجهة الأمامية للقراءة فقط، والتي تم بناؤها باستخدام React. الميزة الأساسية هي نموذج التحقق على السلسلة، حيث يقوم العقد الذكي بالتحقق من صحة البيانات المحسوبة مسبقاً قبل تخزينها، مما يضمن سلامة البيانات وكفاءة الحسابات خارج السلسلة.

Abstract

This project presents a decentralized application (DApp) for managing and verifying university academic records using blockchain technology. The system's back-end is a Solidity smart contract that provides immutable storage for student data, academic plans, and results. The architecture separates administrative write operations, which are handled by a secure intermediary script, from the public read-only front-end, which was built with React. A key feature is the on-chain verification model, where the smart contract validates pre-calculated data from an external system before permanently storing it, ensuring both data .integrity and off-chain computational efficiency

جدول المحتويات

جدول الأشكال

الفصل الأول

التعريف بالمشروع

مقدمة

في ظل التحول الرقمي الذي يشهده قطاع التعليم العالي، تبرز الحاجة الماسة إلى أنظمة حديثة لإدارة وتوثيق السجلات الأكاديمية للطلاب. تعاني الأنظمة التقليدية من عدة تحديات، أبرزها مركزية البيانات التي تجعلها عرضة للتلاعب والأخطاء البشرية، وصعوبة التحقق من صحة الشهادات والسجلات من قبل أطراف خارجية مثل جهات التوظيف أو الجامعات الأخرى. هذه التحديات تؤثر سلبًا على موثوقية وشفافية العملية الأكاديمية.

لمعالجة هذه المشكلات، يقدم هذا المشروع حلاً مبتكرًا يعتمد على تقنية البلوك تشين لبناء نظام لامركزي لإدارة السجلات الأكاديمية. من خلال استخدام العقود الذكية، يهدف النظام إلى إنشاء سجل دائم، آمن، وغير قابل للتغيير للبيانات الأكاديمية. يضمن هذا النهج أن كل معلومة يتم تسجيلها، من علامات المقررات إلى شهادات التخرج، تكون موثوقة تمامًا وقابلة للتحقق الفوري من قبل أي طرف مصرح له، مما يعزز الثقة في النظام التعليمي ويسهل حركة البيانات بين المؤسسات المختلفة.

اهداف المشروع

الهدف الرئيسي

الهدف الأساسي لهذا المشروع هو تصميم وتنفيذ نظام لامركزي لإدارة وتوثيق السجلات الأكاديمية للطلاب. يهدف النظام إلى استغلال تقنية البلوك تشين لإنشاء مصدر وحيد للمعلومة (single source of truth) يتميز بالأمان من خلال التشفير، والثبات (Immutability) لمنع تغيير السجلات بعد توثيقها، والشفافية التي تتيح التحقق من البيانات، مما يزيد من الثقة والمصداقية في الشهادات والسجلات الأكاديمية.

الأهداف الفرعية

1) **بناء عقد ذكي قوي (Back-end):**

تطوير عقد ذكي شامل بلغة Solidity ليكون بمثابة السجل الآمن للواجهة الخلفية. هذا يتضمن تعريف هياكل بيانات (structs) قوية لكل الكيانات الأساسية (الطالب، المقرر، الخطة، العلامات، وسجل التخرج) وتنفيذ مجموعة كاملة من الدوال التي تغطي جميع العمليات الضرورية مع ضمان سلامة البيانات.

2) **تطبيق نموذج التحقق (Verification Model):**

تطبيق نموذج تحقق متطور حيث يكون الدور الأساسي للعقد الذكي هو التحقق وليس الحسابات المعقدة. يقتضي النموذج أن يقوم نظام خارجي بالحسابات (مثل المعدل التراكمي)، ويستقبل العقد الذكي هذه القيم المحسوبة مسبقًا. يقوم العقد بعد ذلك بإعادة حساب النتائج على السلسلة للتحقق من صحتها قبل تخزينها بشكل دائم. هذه الهيكلية تقلل من تكاليف الغاز مع الحفاظ على موثوقية التحقق على السلسلة.

3) **تأسيس طبقة وسيطة آمنة لإدخال البيانات:**

تأسيس فصل واضح وآمن للمهام بين إدخال البيانات وعرضها. تتم عملية إدخال البيانات من قبل المسؤول باستخدام أداة خارجية (في حالتنا Postman) لإرسال الطلبات إلى سكربت وسيط (api.js). يعمل هذا السكربت كبوابة آمنة، حيث يدير المفتاح الخاص بالمسؤول لتوقيع وإرسال المعاملات إلى العقد الذكي. هذا يضمن عزل العمليات الحساسة عن التطبيق العام.

4) **تطوير واجهة أمامية للعرض والتحقق (Front-end):**

تطوير واجهة أمامية سهلة الاستخدام ومخصصة للقراءة فقط باستخدام مكتبة React. تتصل الواجهة بالبلوك تشين عبر web3.js و MetaMask، مما يسمح لأي مستخدم بعرض السجلات الأكاديمية والتحقق منها بشكل آمن. الوظيفة الأساسية للواجهة هي عرض تقارير شاملة للطلاب يتم جلبها مباشرة من العقد الذكي، بالإضافة إلى توفير أدوات للتحقق الفوري من صحة العلامات والمعدلات.

المتطلبات الوظيفية وغير الوظيفية

المتطلبات الوظيفية (Functional Requirements)

1) **إدارة البيانات الأساسية (للمسؤول فقط):**

- يجب أن يسمح النظام للمسؤول بإضافة طالب جديد بمعلومات فريدة (رقم الطالب، الاسم الكامل، سنة التسجيل).
- يجب أن يسمح النظام للمسؤول بإضافة مقرر دراسي جديد (رمز المقرر، الاسم، النوع).
- يجب أن يسمح النظام للمسؤول بإنشاء خطة دراسية جديدة لمقرر معين، وتحديد السنة الدراسية والتقويمية، وعدد الساعات، وأوزان العلامات.

2) **إدارة العلامات والنتائج (للمسؤول فقط):**

- يجب أن يسمح النظام للمسؤول بتسجيل علامات مقرر أكاديمي (شفهي، مذاكرة، نهائي) مع علامة نهائية للتحقق.
- يجب أن يسمح النظام للمسؤول بتسجيل علامة مقرر مشروع أو حلقة بحث.
- يجب أن يسمح النظام للمسؤول بتخزين النتيجة السنوية لطالب بعد التحقق من صحة المعدل (GPA).
- يجب أن يسمح النظام للمسؤول بتسجيل بيانات التخرج النهائية للطالب.

3) وظائف العرض والتحقق (للمستخدم العام):

- يجب أن يسمح النظام لأي مستخدم بالاستعلام عن التقرير الأكاديمي الكامل لطالب باستخدام رقمه الذاتي فقط.
- يجب أن يوفر النظام أداة للتحقق من صحة المعدل السنوي لطالب معين.
- يجب أن يوفر النظام أداة للتحقق من صحة العلامة النهائية المعتمدة لطالب في مقرر معين.

المتطلبات غير الوظيفية (Non-functional Requirements)

1) الأمان (Security):

- يجب أن تكون جميع وظائف كتابة وتعديل البيانات مقيدة بصلاحيات المالك (onlyOwner) فقط، ولا يمكن لأي مستخدم آخر استدعاؤها.
- يجب أن تكون البيانات المخزنة على البلوك تشين محمية بالتشفير المتأصل في الشبكة.

2) سلامة البيانات (Data Integrity):

- يجب أن تكون السجلات الأكاديمية غير قابلة للتغيير (Immutable) بمجرد تخزينها على البلوك تشين.
- يجب أن يقوم العقد الذكي بالتحقق من صحة البيانات المحسوبة (مثل المعدلات والعلامات النهائية) قبل تخزينها لمنع إدخال بيانات خاطئة.

3) الشفافية وقابلية التدقيق (Transparency & Auditability):

- يجب أن تكون جميع المعاملات التي تغير البيانات مسجلة بشكل عام وشفاف على البلوك تشين، مما يسمح بتتبع أي تغيير وتدقيقه.

4) الإتاحة (Availability):

- بما أن النظام لامركزي، يجب أن تكون البيانات متاحة للقراءة في أي وقت طالما أن الشبكة تعمل، دون الاعتماد على خادم مركزي واحد قد يتعطل.

5) الكفاءة (Efficiency):

- تم تصميم العقد الذكي مع مراعاة كفاءة استهلاك الغاز (Gas). تم ذلك من خلال استخدام أنواع بيانات بالحجم المناسب (مثل uint8 بدلاً من uint256)، وتطبيق نموذج التحقق الذي ينقل الحسابات المنطقية المعقدة إلى خارج السلسلة، مما يقلل من تكلفة تنفيذ المعاملات.

6) قابلية التشغيل البيني (Interoperability):

- بما أن البيانات مخزنة على بلوك تشين متوافق مع معايير الإيثيريوم، يمكن للنظام أن يتكامل بسهولة مع تطبيقات لامركزية أخرى أو أنظمة خارجية في المستقبل، مما يسمح بالتحقق من الشهادات والسجلات بشكل آلي وموثوق.

الفصل الثاني

الدراسة المرجعية

مقدمة

يتناول هذا الفصل الإطار النظري والتقني الذي بُني عليه المشروع. سنبدأ بتحليل الأنظمة التقليدية المستخدمة حاليًا في إدارة السجلات الأكاديمية، مع تسليط الضوء على التحديات ونقاط الضعف التي تعاني منها. بعد ذلك، سنقدم شرحًا لمفاهيم تقنية البلوك تشين الأساسية التي تشكل الحل المقترح لهذه التحديات، موضحين كيف تساهم خصائصها مثل اللامركزية والثبات في بناء نظام أكثر أمانًا وموثوقية. كما سيستعرض هذا الفصل عددًا من الدراسات والأبحاث السابقة التي تناولت استخدام البلوك تشين في قطاع التعليم، لننتهي بتحديد الفجوة البحثية التي يسعى هذا المشروع لمعالجتها والمساهمة التي يقدمها في هذا المجال.

الأنظمة التقليدية لإدارة السجلات وتحدياتها

تعتمد معظم المؤسسات الأكاديمية حاليًا على أنظمة إدارة معلومات مركزية، حيث يتم تخزين جميع بيانات الطلاب وسجلاتهم الأكاديمية في قاعدة بيانات مركزية تقع تحت سيطرة وإدارة الجامعة. على الرغم من أن هذه الأنظمة خدمت الغرض منها لعقود، إلا أنها تعاني من تحديات جوهرية في العصر الرقمي:[i]

- **مركزية التحكم ونقطة الفشل الواحدة:** بما أن البيانات مخزنة في مكان واحد، فإن أي عطل تقني في الخادم الرئيسي أو هجوم إلكتروني عليه قد يؤدي إلى فقدان البيانات أو عدم إتاحتها بشكل كامل.
- **قابلية البيانات للتزوير والتعديل:** تمنح الهيكلية المركزية صلاحيات واسعة لمديري الأنظمة، مما يفتح الباب أمام إمكانية تعديل السجلات أو تزويرها بشكل غير مصرح به، سواء كان ذلك عن طريق الخطأ البشري أو بسوء نية.
- **صعوبة التحقق الخارجي:** يواجه أصحاب العمل أو الجامعات الأخرى صعوبة في التحقق من صحة الشهادات والسجلات المقدمة إليهم. تتطلب هذه العملية غالبًا إجراءات ورقية أو تواصلًا مباشرًا مع الجامعة، وهي عملية بطيئة وغير فعالة.
- **انعدام ملكية الطالب لبياناته:** في النموذج المركزي، لا يمتلك الطالب سيطرة حقيقية على بياناته الأكاديمية، بل تظل تحت رحمة المؤسسة التي أصدرتها.

هذه التحديات مجتمعة تخلق حاجة ملحة لنظام جديد يوفر درجة أعلى من الأمان والموثوقية والشفافية، وهو ما تقدمه تقنية البلوك تشين.[ii]

تقنية البلوك تشين كحل مقترح

لمواجهة تحديات الأنظمة المركزية، تقدم تقنية البلوك تشين نموذجًا بديلاً يعتمد على اللامركزية والتشفير. في سياق السجلات الأكاديمية، توفر البلوك تشين حلاً قويًا من خلال أربع خصائص أساسية تم استغلالها في هذا المشروع:

- **اللامركزية (Decentralization):** بدلاً من تخزين البيانات على خادم مركزي واحد، يتم توزيع نسخ متطابقة من السجل على شبكة واسعة من أجهزة الكمبيوتر. هذه الهيكلية تلغي نقطة الفشل الواحدة وتجعل النظام أكثر قوة ومقاومة للهجمات.
- **الثبات وعدم القابلية للتغيير (Immutability):** بمجرد كتابة البيانات (مثل علامة طالب) على البلوك تشين، يصبح من المستحيل تقريبًا تعديلها أو حذفها. يتم ربط كل معاملة بالتي تسبقها عبر بصمة تشفيرية، وأي محاولة لتغيير سجل قديم ستؤدي إلى كسر هذه السلسلة، مما يجعل التلاعب مكشوفًا على الفور.
- **العقود الذكية (Smart Contracts):** هي برامج تعمل بشكل مستقل وتلقائي على البلوك تشين. في مشروعنا، العقد الذكي هو الذي يحتوي على كل القواعد المنطقية، مثل التحقق من صحة العلامات، تقييد صلاحيات الكتابة للمالك، وحساب النتائج. هذه القواعد مبرمجة ولا يمكن تجاوزها، مما يضمن تنفيذ العمليات بدقة ونزاهة.
- **الشفافية وقابلية التدقيق (Transparency):** على الرغم من أن هوية المشاركين محمية، إلا أن جميع المعاملات التي تحدث على الشبكة تكون مسجلة ويمكن للجميع رؤيتها. هذه الشفافية تتيح لأي طرف (مثل جهة توظيف) التحقق من صحة سجل معين وتتبع تاريخه، مما يبني الثقة في النظام.[ii]

استعراض الأبحاث السابقة (Literature Review)

لاستيعاب موقع هذا المشروع ضمن المشهد التقني والأكاديمي الحالي، يستعرض هذا القسم عددًا من المشاريع والدراسات السابقة التي تناولت تطبيق تقنية البلوك تشين في قطاع التعليم.[ii], [iii]

مشروع Blockcerts (مختبر MIT Media Lab)

يعتبر Blockcerts من أوائل المشاريع الرائدة في هذا المجال وأكثرها تأثيرًا، حيث تم تطويره بواسطة مختبر MIT Media Lab.

- **الهدف**: إنشاء معيار مفتوح وموحد لإصدار السجلات والشهادات الأكاديمية على البلوك تشين بشكل آمن وموثوق وقابل للتحقق.
- **آلية العمل:** يقوم المشروع على إنشاء "إثبات وجود" رقمي للشهادة على البلوك تشين. يتم تخزين بصمة تشفيرية (hash) للشهادة على السلسلة، بينما يبقى ملف الشهادة الفعلي خارج السلسلة. يمكن لأي شخص يمتلك الشهادة الرقمية أن يثبت ملكيتها وصحتها دون الحاجة للرجوع إلى المؤسسة المصدرة.
- **المساهمة:** قدم المشروع بنية تحتية ومعيارًا مفتوح المصدر أصبح أساسًا للعديد من التطبيقات اللاحقة، مركزًا بشكل أساسي على إصدار الشهادات النهائية والتحقق منها.[iv]

دراسة "نظام إدارة سجلات الطلاب القائم على البلوك تشين" (A Blockchain-Based Student Records Management System)

تناولت العديد من الأبحاث الأكاديمية بناء أنظمة مشابهة. على سبيل المثال، تقترح دراسة منشورة في IEEE بناء نظام متكامل لإدارة السجلات.

- **الهدف**: تصميم نظام لامركزي يسمح للطلاب بالتحكم في سجلاتهم ومشاركتها، ويتيح للجامعات تحديث هذه السجلات بشكل آمن.
- **آلية العمل**: تعتمد الدراسة على استخدام عقد ذكي لتخزين البيانات الأكاديمية مباشرة على السلسلة. يتم منح صلاحيات مختلفة لكل طرف (طالب، جامعة، جهة توظيف) للتفاعل مع العقد، مما يسمح للطالب بإعطاء إذن مؤقت لجهة التوظيف للاطلاع على سجلاته.
- **المساهمة**: ركزت هذه الدراسة على تمكين الطالب ومنحه سيطرة أكبر على بياناته، بالإضافة إلى إدارة الصلاحيات بشكل دقيق بين الأطراف المختلفة.[v]

تحديد الفجوة البحثية ومساهمة المشروع

من خلال استعراض الدراسات السابقة مثل مشروع Blockcerts والأبحاث المنشورة في IEEE، نلاحظ وجود توجه واضح نحو استخدام البلوك تشين لإصدار الشهادات النهائية وتمكين الطالب من التحكم في بياناته. على الرغم من أهمية هذه المساهمات، إلا أنها تترك فجوة تتعلق بالعمليات الإدارية الداخلية للجامعة وآلية التحقق من البيانات الأكاديمية التفصيلية قبل إصدار الشهادة.

وتتمثل مساهمة هذا المشروع في سد هذه الفجوة من خلال تقديم:

1) **نموذج التحقق على السلسلة (On-Chain Verification Model):** على عكس الأنظمة التي تركز على التخزين المباشر، يقدم مشروعنا آلية تحقق فريدة. يقوم النظام الخارجي بإجراء الحسابات المعقدة، بينما يقوم العقد الذكي بدور "المدقق" الذي يتأكد من صحة هذه الحسابات قبل توثيقها بشكل دائم. هذا النموذج يجمع بين كفاءة الحسابات خارج السلسلة وموثوقية وأمان التحقق على السلسلة.
2) **معمارية آمنة تفصل بين مهام الكتابة والقراءة:** يقدم المشروع تصميمًا معماريًا يفصل بشكل كامل بين صلاحيات إدخال البيانات (المحصورة في طبقة وسيطة آمنة) وصلاحيات عرضها (المتاحة للعامة)، وهو تفصيل دقيق لم تركز عليه الدراسات السابقة التي كانت تدمج الصلاحيات المختلفة في واجهة واحدة.
3) **نظام شامل لإدارة السجلات:** بدلاً من التركيز على الشهادة النهائية فقط، يقدم العقد الذكي في هذا المشروع حلاً متكاملاً لإدارة دورة حياة السجل الأكاديمي بالكامل، بدءًا من الخطط الدراسية، مرورًا بالعلامات التفصيلية لكل مقرر، وانتهاءً بالنتائج السنوية وبيانات التخرج.

بهذه المساهمات، لا يقدم المشروع حلاً لتوثيق الشهادات فحسب، بل يقدم إطار عمل متكامل وآمن للعمليات الأكاديمية اليومية للمؤسسة التعليمية.

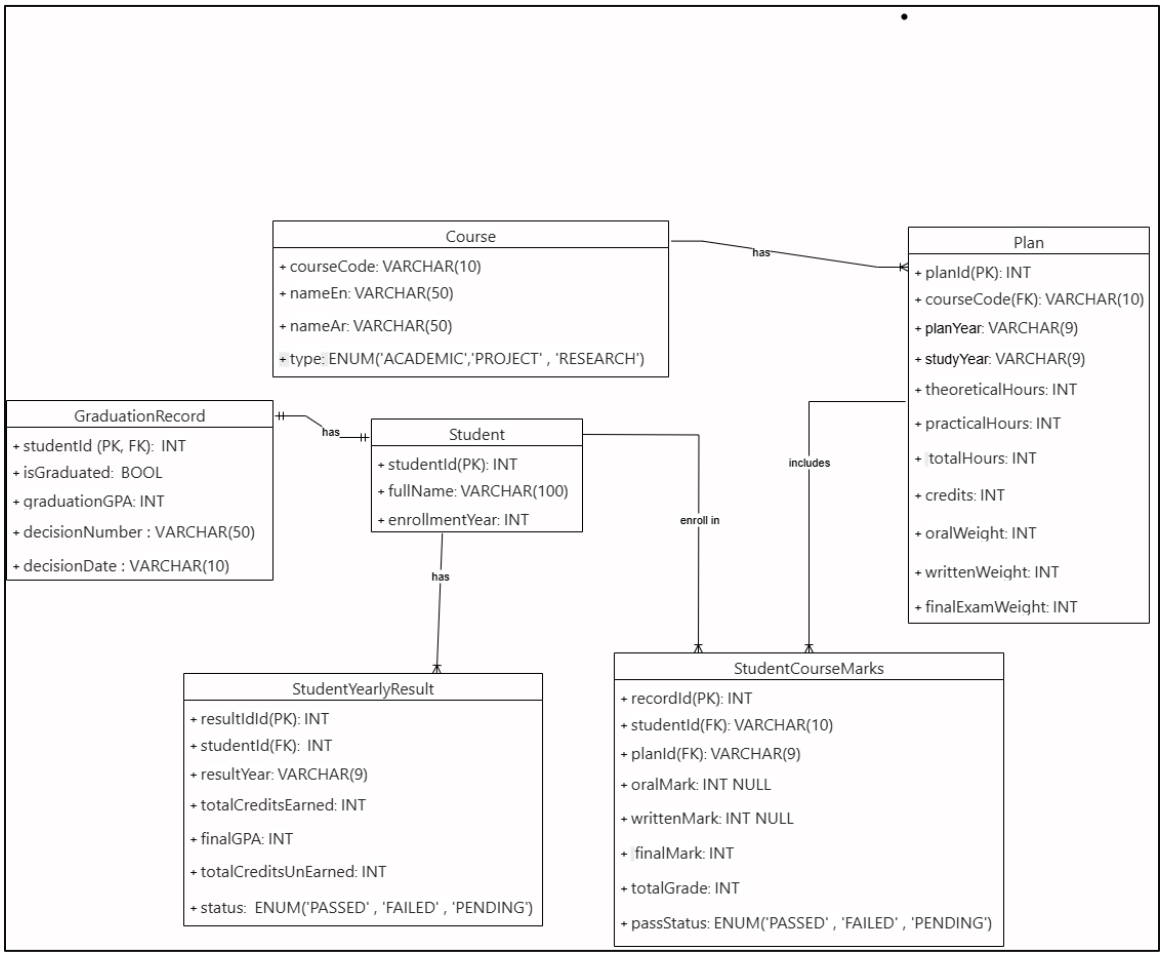
الفصل الثالث

الدراسة التحليلية

مقدمة

يعرض هذا الفصل التصميم الهندسي والهيكلي لنظام السجلات الأكاديمية اللامركزي. تهدف المخططات التالية إلى توفير فهم مرئي وعميق لمكونات النظام، بدءًا من هيكل البيانات الأساسي، مرورًا بالهيكلية العامة للتطبيق، وانتهاءً بتدفق العمليات المنطقية الرئيسية. تم اختيار هذه المخططات لتوثيق كيفية تحقيق النظام لأهدافه الوظيفية وغير الوظيفية بشكل فعال وآمن.

مخطط نموذج البيانات (ERD / Class Diagram)



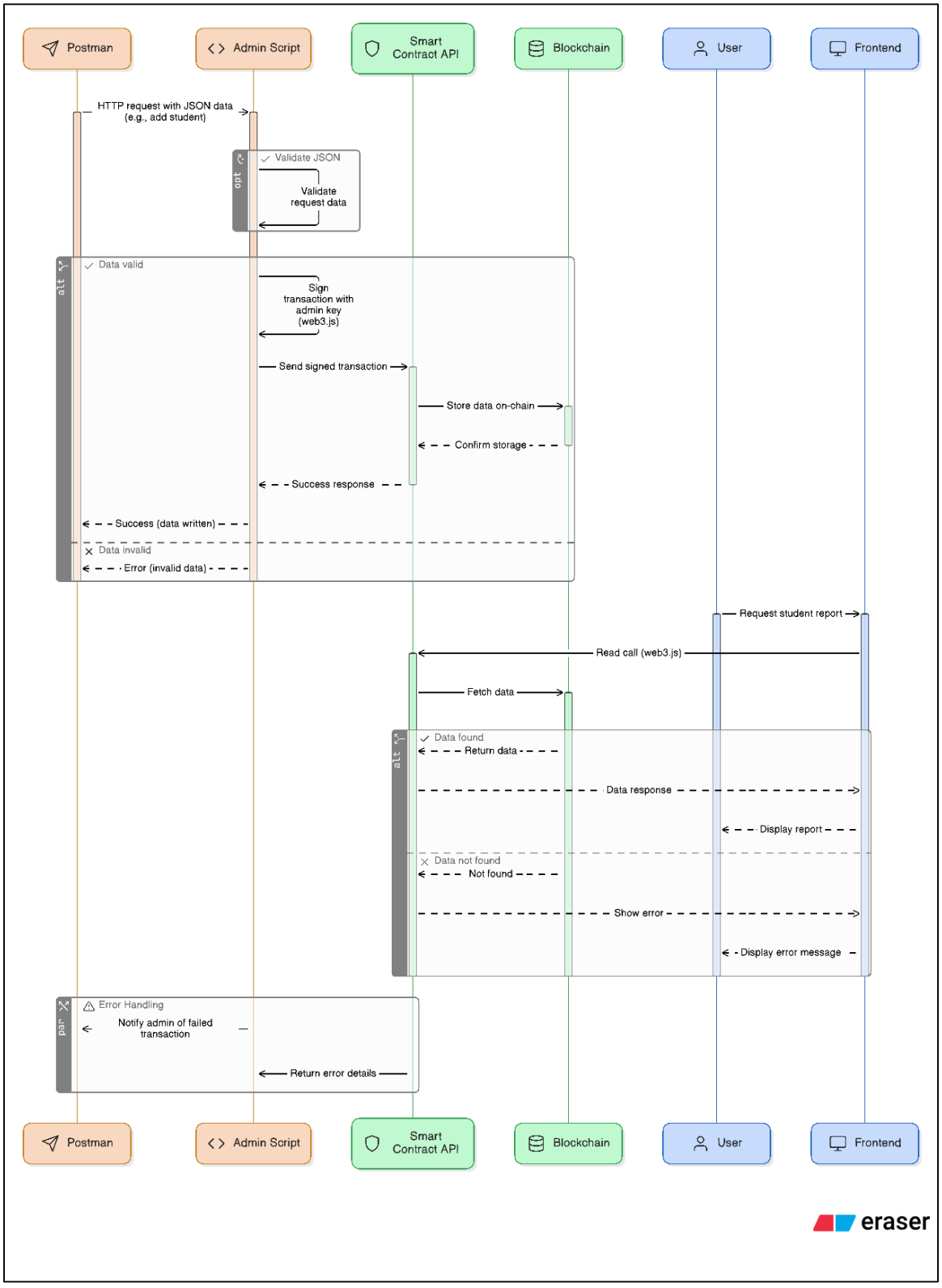
1 Figure

يوضح الشكل (1) مخطط الكيان والعلاقات (ERD) الذي يمثل نموذج البيانات للواجهة الخلفية للنظام. تم تصميم هذا النموذج ليتم تنفيذه باستخدام هياكل البيانات (structs) وآليات التخزين (mappings) في عقد Solidity الذكي. يتكون النموذج من الكيانات الرئيسية التالية:

- **كيان الطالب (Student):** يخزن المعلومات الأساسية والثابتة لكل طالب، مثل رقمه الجامعي الفريد (studentId) الذي يعمل كمفتاح أساسي، اسمه الكامل، وسنة التحاقه بالجامعة.
- **كيان المقرر (Course):** يمثل المقررات الدراسية التي تقدمها الجامعة بشكل عام. يحتوي على رمز المقرر الفريد (courseCode)، اسم المقرر باللغتين، ونوعه (أكاديمي، مشروع، أو بحث).
- **كيان الخطة الدراسية (Plan):** يمثل نسخة محددة من مقرر معين في سنة تقويمية معينة (planYear) وينتمي إلى سنة دراسية محددة (studyYear) في المنهج. هذا الكيان هو المسؤول عن تحديد القواعد الأكاديمية للمقرر في تلك السنة، مثل عدد الساعات المعتمدة (credits) وأوزان العلامات المختلفة. العلاقة بين المقرر والخطة هي واحد لمتعدد (One-to-Many).
- **كيان علامات الطالب في المقرر (StudentCourseMarks):** هذا الكيان هو سجل لنتيجة محاولة طالب معين في خطة دراسية معينة ، ويرتبط بكيان الخطة (Plan) بعلاقة واحد لمتعدد. يخزن العلامات التفصيلية (شفهي، مذاكرة، نهائي)، العلامة الإجمالية، وحالة النجاح النهائية للمقرر.
- **كيان النتيجة السنوية (StudentYearlyResult):** يمثل السجل الرسمي لأداء الطالب في سنة دراسية معينة (studyYear). يرتبط هذا الكيان بكيان الطالب (Student) بعلاقة واحد لمتعدد، حيث يمكن للطالب أن يمتلك عدة نتائج سنوية عبر مسيرته الدراسية. يتم تخزين هذا السجل بعد انتهاء العام الدراسي، ويحتوي على ملخص الأداء مثل المعدل السنوي النهائي (finalGPA)، مجموع الوحدات المكتسبة وغير المكتسبة، والحالة النهائية (ناجح أو راسب).
- **كيان سجل التخرج (GraduationRecord):** كيان يرتبط بالطالب بعلاقة واحد لواحد (One-to-One)، ويخزن البيانات النهائية المتعلقة بتخرجه، مثل معدل التخرج، رقم قرار التخرج، وتاريخه.

مخطط التسلسل (Sequence Diagram)

بعد توضيح الهيكلية الثابتة للبيانات، يهدف هذا القسم إلى شرح الهيكلية الديناميكية للنظام. يوضح الشكل (2) مخطط التسلسل الذي يمثل تدفق العمليات والتفاعلات بين المكونات الرئيسية للنظام مع مرور الزمن.


Postman
Admin Script
Smart Contract API
Blockchain
User
Frontend
HTTP request with JSON data (e.g., add student)
opt
Validate JSON
Validate request data
alt
Data valid
Sign transaction with admin key (web3.js)
Send signed transaction
Store data on-chain
Confirm storage
Success response
Success (data written)
Data invalid
Error (invalid data)
Request student report
Read call (web3.js)
Fetch data
alt
Data found
Return data
Data response
Display report
Data not found
Not found
Show error
Display error message
par
Error Handling
Notify admin of failed transaction
Return error details
Postman
Admin Script
Smart Contract API
Blockchain
User
Frontend
eraser

2 Figure

يوضح هذا المخطط المعمارية المتبعة في النظام، والتي تقوم على مبدأ فصل الصلاحيات والمهام بين عمليات كتابة البيانات وقراءتها لضمان أقصى درجات الأمان والكفاءة. يتكون النظام من مسارين رئيسيين:

1) **مسار كتابة البيانات**، وهو مخصص للمسؤول فقط. تبدأ العملية من أداة خارجية مثل Postman، التي تعمل كواجهة إدخال للمسؤول لإرسال البيانات الجديدة (مثل إضافة طالب أو تسجيل علامات) على هيئة طلبات HTTP بصيغة JSON. يتم استقبال هذه الطلبات من قبل سكربت إدارة وسيط (api.js) يعمل كخادم ويب صغير. يقوم هذا السكربت بدور "الوكيل الآمن"، حيث يستقبل البيانات، ويستخدم المفتاح الخاص بالمسؤول لتوقيع معاملة إيثيريوم، ثم يرسلها عبر مكتبة web3.js إلى واجهة برمجة التطبيقات (API) الخاصة بالعقد الذكي. يقوم العقد بدوره بالتحقق من صحة المعاملة والبيانات ثم تخزينها بشكل دائم على البلوك تشين.
2) **مسار قراءة البيانات**، وهو متاح للمستخدم العام. يتفاعل المستخدم مع الواجهة الأمامية المبنية باستخدام React. تقوم الواجهة الأمامية، عند طلب المستخدم لمعلومات معينة (مثل تقرير طالب)، بإجراء استدعاء قراءة (call) مباشرة إلى واجهة برمجة التطبيقات (API) الخاصة بالعقد الذكي باستخدام web3.js. يقوم العقد بجلب البيانات المطلوبة من البلوك تشين وإعادتها إلى الواجهة الأمامية لعرضها للمستخدم.

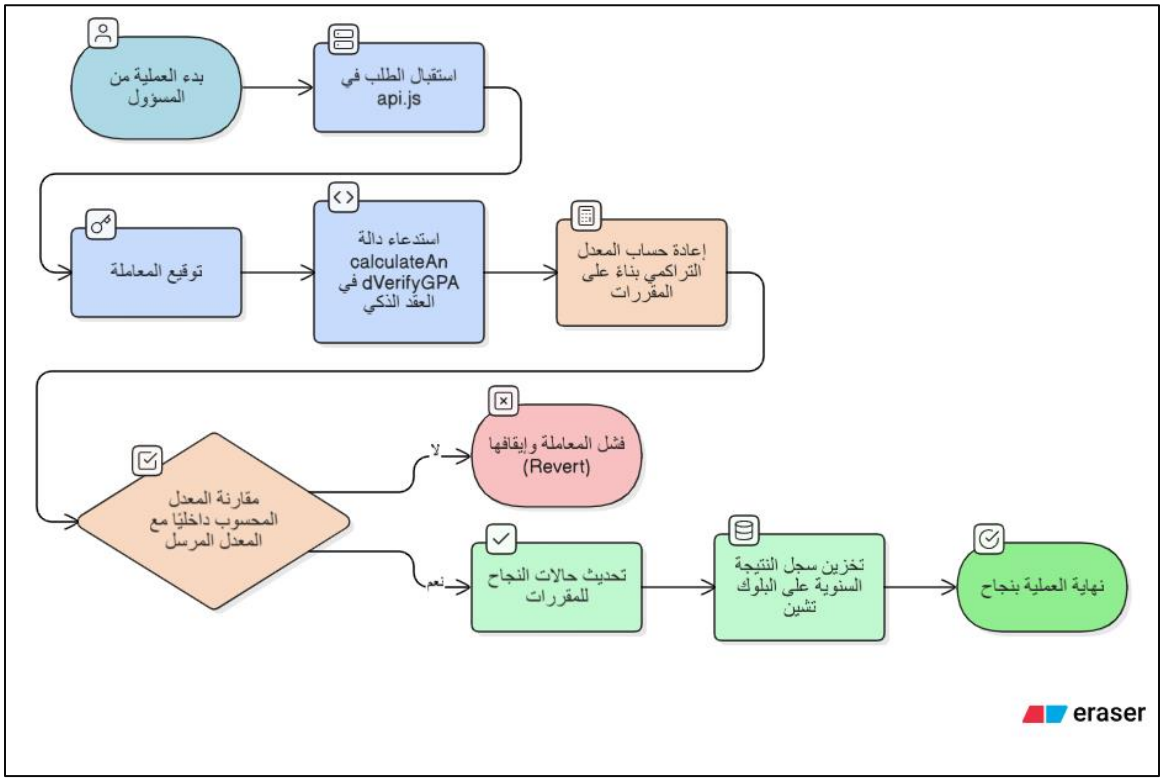
هذه الهيكلية تضمن أن العمليات الحساسة التي تتطلب توقيعًا ومفتاحًا خاصًا معزولة تمامًا في طبقة خلفية آمنة، بينما تظل الواجهة الأمامية بسيطة وآمنة ومخصصة للعرض فقط.

مخططات التدفق (Flowcharts)

يركز هذا القسم على توضيح تسلسل العمليات المنطقية لثلاث وظائف رئيسية تمثل الأنواع المختلفة من التفاعلات في العقد الذكي: عملية حساب وتحقق معقدة، عملية إدخال مع تحقق من السلامة، وعملية قراءة وتجميع بيانات.

مخطط تدفق عملية حساب النتيجة السنوية:

هذا المخطط يمثل العملية الأكثر أهمية وتعقيدًا في النظام، وهي عملية التحقق وتخزين النتيجة السنوية (calculateAndVerifyGPA).



3 Figure

وصف المخطط:

1) (بداية): يبدأ المسؤول العملية عن طريق إرسال طلب من Postman يحتوي على بيانات النتيجة المحسوبة.
2) (عملية): يستقبل سكربت api.js الطلب, يوقع المعاملة ويستدعي دالة calculateAndVerify في العقد الذكي.
3) (عملية): يقوم العقد الذكي داخليًا بإعادة حساب المعدل التراكمي (finalGPA) بناءً على قائمة المقررات (planIds_) المستلمة.
4) (قرار): هل المعدل المحسوب داخليًا (finalGPA) يساوي المعدل المرسل من النظام (providedGPA_)؟
   - (إذا لا): (عملية) تفشل المعاملة ويتم إيقافها (Revert). (نهاية)
   - (إذا نعم): (عملية) يقوم العقد بتحديث حالات النجاح (passStatus) للمقررات، ثم يقوم بتخزين سجل النتيجة السنوية (StudentYearlyResult) بشكل دائم على البلوك تشين. (نهاية)

الشرح التفصيلي للمخطط:

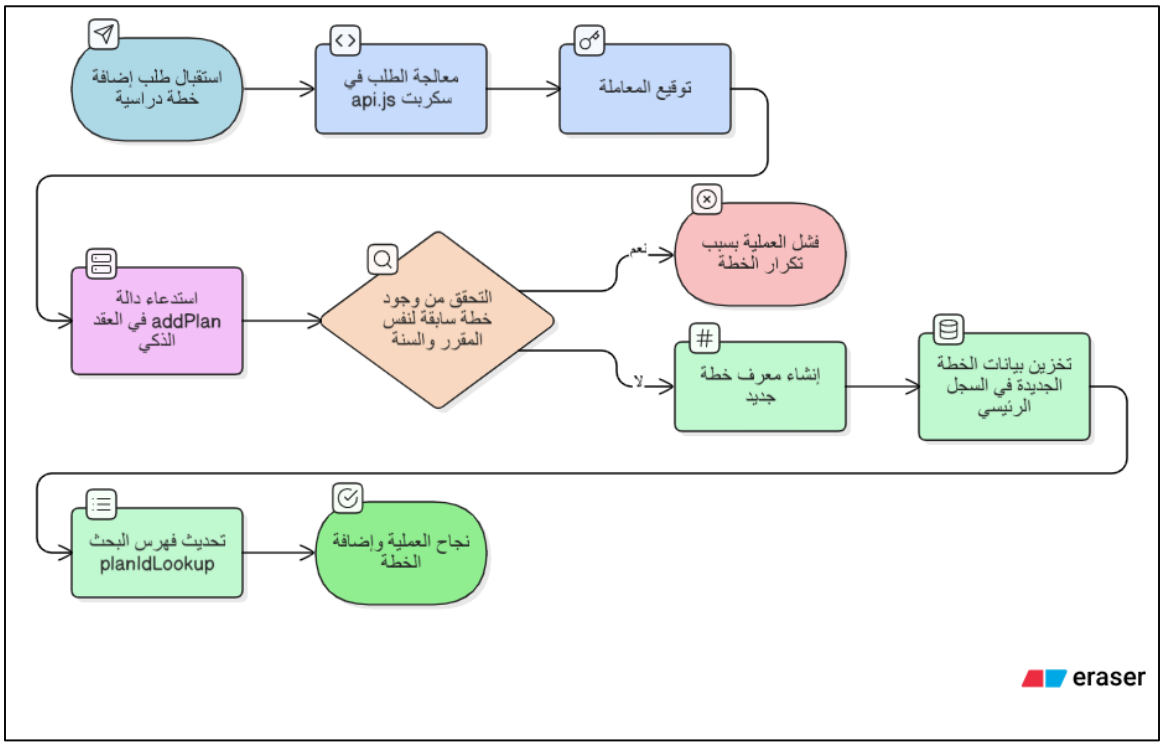
يوضح الشكل (3) مخطط التدفق لعملية التحقق وتخزين النتيجة السنوية، والتي تمثل جوهر نموذج التحقق (Verification Model) المطبق في النظام.

تبدأ العملية من النظام الخارجي (الذي يحاكيه المسؤول عبر Postman) بإرسال البيانات المحسوبة مسبقًا، بما في ذلك المعدل النهائي وقائمة المقررات المعتمدة. يقوم السكربت الوسيط (api.js) بتجهيز وتوقيع المعاملة وإرسالها إلى دالة calculateAndVerifyGPA في العقد الذكي.

عند استلام البيانات، يقوم العقد بتنفيذ دوره كـ"مدقق"؛ حيث يقوم بإعادة حساب المعدل التراكمي بناءً على العلامات والوحدات المخزنة على السلسلة. بعد ذلك، يدخل المخطط في نقطة قرار حاسمة، حيث تتم مقارنة النتيجة المحسوبة داخليًا مع النتيجة المرسلة. في حالة عدم التطابق، يتم رفض المعاملة بالكامل لضمان عدم تخزين أي بيانات غير دقيقة. أما في حالة التطابق، فيستمر التدفق ليقوم العقد بتحديث حالات نجاح المقررات الفردية وتخزين سجل النتيجة السنوية الموثوقة بشكل دائم على البلوك تشين.

مخطط تدفق عملية إضافة خطة دراسية

بعد أن استعرضنا عملية التحقق المعقدة، يوضح هذا المخطط تدفق عملية كتابة بيانات أبسط، وهي إضافة خطة دراسية جديدة (addPlan). يركز هذا المخطط على توضيح كيفية ضمان سلامة البيانات المدخلة ومنع تكرارها.



4 Figure

وصف المخطط

1) (بداية): يتم استقبال طلب إضافة خطة دراسية جديدة.
2) (عملية): يقوم سكربت api.js بمعالجة الطلب وتجهيزه.
3) (عملية): يتم توقيع المعاملة رقميًا باستخدام المفتاح الخاص للمسؤول.
4) (عملية): يتم استدعاء دالة addPlan في العقد الذكي.
5) (قرار): يقوم العقد بالتحقق: هل توجد خطة مسجلة مسبقًا لنفس المقرر والسنة؟
   - (إذا نعم): (عملية) تفشل العملية بسبب تكرار الخطة. (نهاية)
   - (إذا لا): (عملية) يقوم العقد بإنشاء معرّف خطة جديد (planId).
6) (عملية): يتم تخزين بيانات الخطة الجديدة في السجل الرئيسي.
7) (عملية): يتم تحديث فهرس البحث (planIdLookup).
8) (عملية): تنجح العملية ويتم تأكيد إضافة الخطة. (نهاية)

الشرح التفصيلي للمخطط

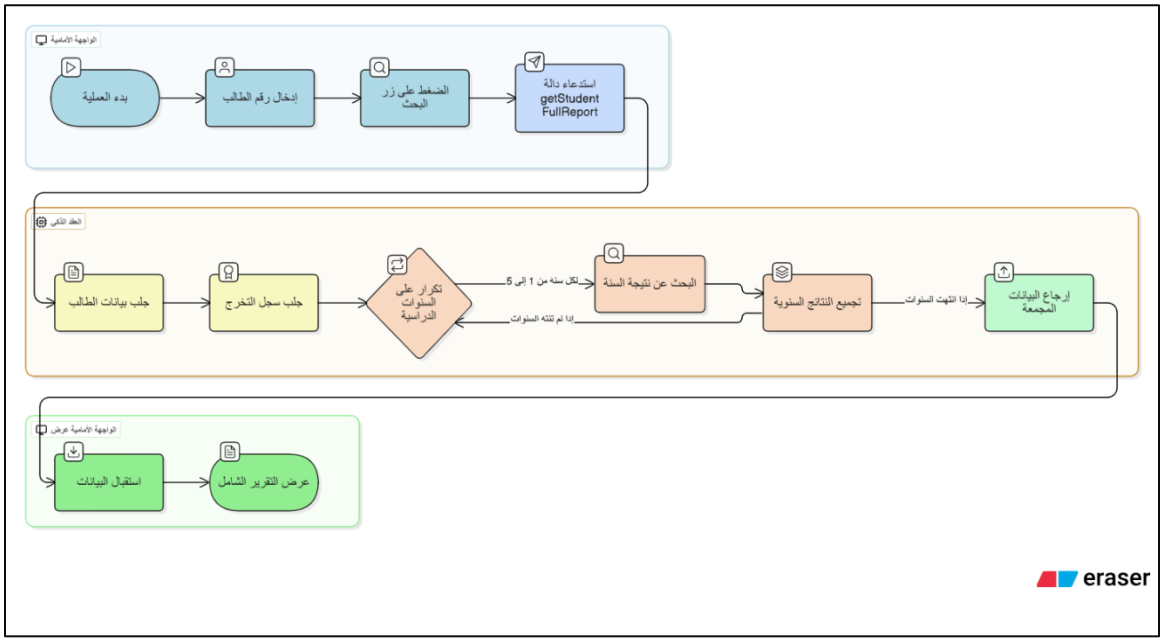
يوضح الشكل (4) مخطط التدفق لعملية إضافة خطة دراسية جديدة. تبرز أهمية هذا المخطط في توضيح آليات التحقق المدمجة في العقد لضمان سلامة واتساق البيانات (Data Integrity).

تبدأ العملية كالمعتاد من المسؤول، وتمر عبر السكربت الوسيط لتصل إلى دالة addPlan في العقد. قبل تخزين أي بيانات، يدخل التدفق في نقطة قرار حاسمة حيث يستخدم العقد فهرس البحث (planIdLookup) للتأكد من عدم وجود خطة مسجلة مسبقًا لنفس المقرر في نفس السنة.

في حالة وجود سجل مكرر، يتم رفض المعاملة فورًا، مما يمنع إدخال بيانات متضاربة. أما في حالة عدم وجود تكرار، فيستمر التدفق ليقوم العقد بتنفيذ عملية الكتابة، حيث يتم إنشاء معرّف فريد للخطة (planId) وتخزينها في السجل الرئيسي وتحديث فهرس البحث. هذه الآلية تضمن أن كل خطة دراسية فريدة من نوعها.

مخطط تدفق عملية قراءة التقرير الشامل

يوضح هذا المخطط الأخير تدفق عملية قراءة وتجميع البيانات. سنقوم بوصف مخطط التدفق لدالة getStudentFullReport، التي تمثل الوظيفة الأساسية للواجهة الأمامية في عرض معلومات الطالب.



5 Figure

وصف المخطط

1) (بداية): تبدأ العملية من الواجهة الأمامية.
2) (عملية): يقوم المستخدم بإدخال رقم الطالب في حقل البحث.
3) (عملية): يضغط المستخدم على زر البحث.
4) (عملية): تقوم الواجهة الأمامية باستدعاء دالة getStudentFullReport من العقد الذكي.
5) (عملية): يقوم العقد بجلب البيانات الأساسية للطالب (studentDetails).
6) (عملية): يقوم العقد بجلب سجل التخرج الخاص بالطالب (gradRecord).
7) (عملية - حلقة تكرارية): يبدأ العقد حلقة تكرارية على السنوات الدراسية (من 1 إلى 5).
8) (قرار داخل الحلقة): هل توجد نتيجة مسجلة لهذه السنة الدراسية؟
   - (إذا نعم): (عملية) يتم تجميع النتيجة السنوية في مصفوفة مؤقتة.
   - (إذا لا): يستمر التدفق للحلقة التالية.
9) (عملية): بعد انتهاء الحلقة، يقوم العقد بإرجاع كل البيانات المجمعة (بيانات الطالب، سجل التخرج، ومصفوفة النتائج).
10) (عملية): تستقبل الواجهة الأمامية البيانات.
11) (نهاية): يتم عرض التقرير الشامل للمستخدم.

الشرح التفصيلي للمخطط

يوضح الشكل (5) مخطط التدفق لعملية قراءة التقرير الشامل للطالب، وهي عملية قراءة فقط (view) لا تستهلك أي غاز.

تبدأ العملية من المستخدم في الواجهة الأمامية، الذي يقوم بطلب تقرير طالب معين. تستدعي الواجهة دالة getStudentFullReport من العقد الذكي، والتي تعمل كـ "منسق بيانات". تقوم الدالة بتنفيذ سلسلة من عمليات القراءة الداخلية المنظمة؛ حيث تجلب أولاً البيانات الشخصية للطالب وسجل تخرجه من الـ mappings المخصصة لهما.

بعد ذلك، يدخل التدفق في حلقة تكرارية للبحث عن النتائج السنوية المسجلة للطالب عبر جميع السنوات الدراسية المحتملة (من الأولى إلى الخامسة). يقوم العقد بتجميع كل النتائج التي يجدها في مصفوفة واحدة. في النهاية، يتم إرجاع كل البيانات المجمعة في استجابة واحدة ومنظمة إلى الواجهة الأمامية، التي تقوم بدورها بعرض التقرير الكامل للمستخدم. هذه الآلية تبرز قدرة العقد الذكي على العمل كقاعدة بيانات فعالة وقادرة على تجميع وتقديم معلومات معقدة من مصادر متعددة في استدعاء واحد.

الفصل الرابع

الدراسة النظرية

مقدمة

لفهم الأساس التقني الذي يقوم عليه هذا المشروع، يتناول هذا الفصل المفاهيم النظرية للتقنيات الأساسية المستخدمة. سنبدأ بمقدمة عن تقنية البلوك تشين، موضحين خصائصها الجوهرية التي تجعلها مناسبة للتطبيقات التي تتطلب درجة عالية من الأمان والثقة. بعد ذلك، سنتعمق في منصة الإيثيريوم كبيئة رائدة لتنفيذ العقود الذكية، ثم نستعرض لغة البرمجة Solidity المستخدمة في كتابة هذه العقود، وأخيرًا نوضح المعمارية العامة للتطبيقات اللامركزية (DApps) وكيف يندرج هذا المشروع ضمنها.

مقدمة عن تقنية البلوك تشين

البلوك تشين (Blockchain) هي سجل معاملات موزع (Distributed Transaction Log) تسمح بتسجيل المعاملات وتخزين البيانات بطريقة آمنة، شفافة، وغير قابلة للتغيير. على عكس قواعد البيانات المركزية التقليدية التي يتم التحكم فيها من قبل جهة واحدة، يتم توزيع نسخة من سجل البلوك تشين على شبكة واسعة من أجهزة الكمبيوتر (تسمى العقد - Nodes).

تتكون البلوك تشين من سلسلة من "الكتل" (Blocks)، حيث تحتوي كل كتلة على مجموعة من المعاملات. يتم ربط كل كتلة بالكتلة التي تسبقها باستخدام بصمة تشفيرية (Cryptographic Hash)، مما يخلق سلسلة مترابطة. أي محاولة لتغيير البيانات في كتلة ستؤدي إلى تغيير بصمتها التشفيرية، وبالتالي كسر الرابط مع جميع الكتل اللاحقة، وهو ما يجعل التلاعب بالبيانات شبه مستحيل ومكشوفًا على الفور.[vi]

منصة الإيثيريوم والعقود الذكية

الإيثيريوم (Ethereum) هي منصة بلوك تشين لامركزية ومفتوحة المصدر، تُعتبر تطورًا لتقنية البلوك تشين التي قدمتها البيتكوين. الميزة الأساسية التي قدمتها الإيثيريوم هي العقود الذكية (Smart Contracts).

العقد الذكي هو برنامج كمبيوتر يتم تخزينه وتشغيله على شبكة البلوك تشين. يحتوي العقد على مجموعة من القواعد والشروط المبرمجة مسبقًا، ويقوم بتنفيذ بنود "اتفاق" معين بشكل تلقائي عند تحقق هذه الشروط. بمجرد نشر العقد الذكي على الشبكة، لا يمكن تغييره، وتكون نتائجه حتمية ومضمونة بالكود، مما يلغي الحاجة إلى وسيط مركزي لتنفيذ الاتفاقيات. في مشروعنا، العقد الذكي هو الذي يحتوي على كل منطق إدارة السجلات الأكاديمية.[vii]

لغة البرمجة Solidity

Solidity هي اللغة الرئيسية والمخصصة لكتابة العقود الذكية على شبكة الإيثيريوم. الجميع يفهمها ويدعمها، وهناك كم هائل من الأدوات والمكتبات والمطورين الذين يعملون بها، مما يجعلها الخيار الأكثر أمانًا .

أهم ميزاتها:

- **مصممة خصيصًا للعقود الذكية:** توفر هياكل بيانات وميزات مدمجة تتناسب مع طبيعة البلوك تشين، مثل أنواع العناوين (address) والـ mappings.
- **دعم الوراثة والمكتبات:** تسمح بإنشاء عقود قابلة لإعادة الاستخدام وتنظيم الكود بشكل هرمي.
- **التحكم الدقيق في الموارد:** توفر آليات للتحكم في استهلاك الغاز، وهو أمر حيوي في بيئة البلوك تشين.[viii]

معمارية التطبيقات اللامركزية (DApps)

التطبيق اللامركزي (DApp) هو تطبيق يعمل على شبكة لامركزية (مثل الإيثيريوم) بدلاً من خادم مركزي. يتكون هيكله عادةً من جزأين رئيسيين:

- **الواجهة الخلفية (Back-end)**: هي العقد الذكي نفسه الذي يعمل على البلوك تشين. هو المسؤول عن كل المنطق وتخزين الحالة.
- **الواجهة الأمامية (Front-end):** هي واجهة المستخدم التي تعمل في متصفح الويب (مثل تطبيق React الذي بنيناه). تتفاعل هذه الواجهة مع العقد الذكي باستخدام مكتبات مثل web3.js ومحفظة مثل MetaMask.

مشروعنا يتبع هذه المعمارية، مع إضافة طبقة وسيطة (api.js) كبوابة آمنة لعمليات الكتابة الإدارية، وهو نمط شائع في التطبيقات اللامركزية التي تتطلب صلاحيات محددة.[ix]

الفصل الخامس

الأدوات المستخدمة

مقدمة

تم بناء النظام بالاعتماد على مجموعة من التقنيات والأدوات المتخصصة في تطوير التطبيقات اللامركزية (DApps). تم اختيار كل أداة بعناية لتخدم غرضًا محددًا في معمارية النظام، والتي تنقسم إلى ثلاث طبقات رئيسية:

الواجهة الخلفية (Back-end - العقد الذكي):

- **لغة البرمجة:** تم استخدام لغة Solidity (إصدار ^0.8.0) لكتابة المنطق الكامل للعقد الذكي، بما في ذلك تعريف هياكل البيانات، منطق التحقق، وإدارة الصلاحيات.
- **بيئة التطوير:** تم استخدام Remix IDE، وهي بيئة تطوير متكاملة عبر الإنترنت، لكتابة، ترجمة (compiling)، ونشر العقد الذكي في المراحل الأولية.[x]
- **الشبكة المحلية:** تم استخدام Ganache-CLI لإنشاء شبكة بلوك تشين محلية لغرض الاختبار والتطوير، مما يوفر بيئة سريعة ومجانية لمحاكاة شبكة الإيثيريوم.

الطبقة الوسيطة (Admin's API Layer):

- **بيئة التشغيل:** تم بناء هذه الطبقة باستخدام Node.js.
- **الخادم:** تم استخدام مكتبة Express.js لإنشاء خادم ويب بسيط ومستقر.
- **التفاعل مع البلوك تشين:** تم استخدام مكتبة Web3.js (إصدار 1.10.0) لتكون الجسر بين الخادم والعقد الذكي، حيث تقوم ببناء وتوقيع وإرسال المعاملات.
- **أداة الاختبار:** تم استخدام Postman كعميل لإرسال طلبات HTTP إلى السكربت الوسيط، محاكاةً لدور النظام الخارجي في إدخال البيانات.

الواجهة الأمامية (Front-end):

- **إطار العمل:** تم بناء الواجهة باستخدام مكتبة React (عبر أداة Vite)، مما يوفر واجهة مستخدم سريعة وتفاعلية.

- **التفاعل مع البلوك تشين:** تم استخدام مكتبة Web3.js أيضًا في الواجهة الأمامية، بالإضافة إلى إضافة MetaMask في المتصفح، لتمكين التطبيق من الاتصال بالبلوك تشين وقراءة البيانات مباشرة من حساب المستخدم.
- **محرر الكود:** تم استخدام Visual Studio Code (VS Code) لتطوير الواجهة الأمامية والسكربت الوسيط.

الفصل السادس

القسم العملي

معمارية النظام المتبعة

تم تصميم النظام بناءً على معمارية ثلاثية الطبقات (Three-Tier Architecture)، وهي هيكلية قياسية تهدف إلى فصل المهام وتنظيم الكود. تتكون هذه المعمارية من:

- ***الواجهة الخلفية (Back-end)***:

ممثلة بالعقد الذكي على البلوك تشين، وهي مسؤولة عن منطق العمل وتخزين البيانات بشكل آمن.

- ***الطبقة الوسيطة (Middle Layer):***

ممثلة بسكربت الإدارة (api.js)، وهو المسؤول عن استقبال طلبات الكتابة من النظام الخارجي (Postman)، وتوقيعها، وإرسالها كمعاملات إلى البلوك تشين.

- ***الطبقة الواجهة الأمامية (Front-end):***

ممثلة بتطبيق React المخصص للعرض فقط، والذي يتفاعل مباشرة مع العقد لقراءة البيانات.

تفصيل أجزاء النظام وطريقة تنفيذ التكامل فيما بينها

يعتمد النظام على تكامل تقني دقيق بين ثلاثة أجزاء رئيسية: العقد الذكي، الطبقة الوسيطة، والواجهة الأمامية. ولأن العقد الذكي هو المكون المركزي الذي تتفاعل معه كل الأجزاء الأخرى دون أن تتفاعل مع بعضها البعض، فإن هذا القسم يركز على شرح آلية تكامل كل جزء مع هذا المركز.

الواجهة الخلفية (العقد الذكي):

هو أساس النظام والمرجع الموثوق والوحيد للبيانات (Single Source of Truth). يتكامل العقد مع الأجزاء الأخرى من خلال واجهة التطبيق الثنائية (ABI) الخاصة به. يعمل العقد بآلية الطلب والاستجابة، حيث لا يبدأ أي تفاعل بنفسه، بل ينفذ فقط نوعين من الاستدعاءات القادمة من الخارج: المعاملات (Transactions) الموقعة لتغيير البيانات، والاستعلامات (Calls) لقراءة البيانات.[xi]

الطبقة الوسيطة (لإدخال البيانات):

تم تصميم هذه الطبقة لمحاكاة نظام إداري خارجي، وتتكون من جزأين متكاملين:

1) Postman: يعمل كواجهة للمسؤول لإرسال البيانات الأولية على هيئة طلب HTTP POST يحتوي على JSON.
2) سكربت الإدارة (api.js): يعمل كخادم ويب يستقبل الطلب من Postman. آلية تكامله مع العقد الذكي تتم كالتالي: يقوم بتحليل بيانات الـ JSON، ثم باستخدام مكتبة web3.js، يبني معاملة إيثيريوم متوافقة مع الـ ABI. بعد ذلك، يستخدم المفتاح الخاص للمسؤول لتوقيع المعاملة رقميًا وإرسالها إلى شبكة Ganache. أخيرًا، ينتظر تأكيد المعاملة من البلوك تشين ويعيد الرد النهائي إلى Postman.

الواجهة الأمامية (لعرض البيانات):

هي البوابة المخصصة للمستخدم النهائي للقراءة فقط. آلية تكاملها مع العقد الذكي تتم كالتالي:

1) عند التحميل، يتصل التطبيق المبني بـ React بمحفظة MetaMask للاتصال بالشبكة وقراءة عنوان المستخدم.
2) باستخدام web3.js والـ ABI وعنوان العقد، يتم إنشاء نسخة JavaScript من العقد.
3) عندما يبحث المستخدم، تقوم الواجهة بإجراء استدعاء قراءة (call) مجاني ومباشر إلى دوال الـ view في العقد (مثل getStudentFullReport).
4) يتم استقبال البيانات من البلوك تشين مباشرة، معالجتها، ثم استخدام useState لتحديث الواجهة وعرض التقرير الكامل للمستخدم.

تنفيذ العقد الذكي

تمت كتابة العقد الذكي بلغة Solidity، وهو يمثل جوهر النظام. تم التركيز على بناء هياكل بيانات دقيقة ووظائف منطقية قوية:

- **هياكل البيانات (structs):** تم استخدامها لنمذجة الكيانات الرئيسية مثل Student, Plan, GraduationRecord بشكل منظم، مما يضمن تخزين البيانات بشكل مترابط وواضح.
- **آليات التخزين (mappings):** تم الاعتماد على بنية الـ mapping كقاعدة بيانات أساسية على السلسلة، مما يوفر وصولاً فعالاً للبيانات بكلفة غاز منخفضة، خاصة مع استخدام mapping متداخلة للربط بين الطالب والسنة الدراسية.

- **الدوال المنطقية الرئيسية:**
  - **دوال التحقق (setAcademicMarks و calculateAndVerifyGPA):** تمثل هذه الدوال جوهر "نموذج التحقق"، حيث تستقبل بيانات محسوبة وتقوم بإعادة حسابها داخليًا للتحقق من صحتها قبل تخزينها.
  - **دالة القراءة الشاملة (getStudentFullReport):** هي دالة قراءة (view) متقدمة تقوم بتجميع كل بيانات الطالب الأكاديمية (الشخصية، السنوية، والتخرج) من mappings مختلفة في استدعاء واحد، مما يسهل عمل الواجهة الأمامية.

تنفيذ الواجهة الأمامية

تم بناء الواجهة الأمامية كتطبيق صفحة واحدة (Single Page Application) باستخدام مكتبة React، مع التركيز على أن تكون مخصصة للقراءة فقط.

- **المكون الرئيسي (App.jsx):** هو المكون الذي يدير حالة التطبيق العامة، بما في ذلك الاتصال بالبلوك تشين وتخزين بيانات العقد.
- **إدارة الحالة (useState):** تم استخدام هذا "الخطاف" من React لتخزين البيانات العائدة من البلوك تشين (مثل تقرير الطالب) وعرضها بشكل تفاعلي. أي تغيير في هذه الحالة يؤدي إلى إعادة رسم الواجهة تلقائيًا.
- **تهيئة الاتصال (useEffect):** تم استخدام هذا "الخطاف" لتشغيل دالة الاتصال بالبلوك تشين مرة واحدة عند تحميل الصفحة، مما يوفر تجربة مستخدم سلسة.
- **التفاعل مع العقد:** يتم التفاعل مع العقد الذكي عبر مكتبة web3.js ومحفظة MetaMask، حيث يتم استدعاء دوال القراءة (.call()) لجلب البيانات وعرضها في جداول ونماذج منظمة للمستخدم.

الفصل السابع

الاختبارات

مقدمة

تم اتباع خطة اختبار منهجية للتأكد من أن كل مكون من مكونات النظام يعمل بشكل صحيح ومنفرد، وأن جميع المكونات تتكامل وتعمل معًا بشكل فعال وآمن.

الاختبار الوظيفي (Functional Testing)

ركزت هذه الاختبارات على فحص وظائف النظام الرئيسية بشكل منفصل للتأكد من أنها تؤدي وظيفتها المنطقية بشكل صحيح. تم إجراء هذه الاختبارات يدويًا باستخدام بيئة Remix IDE وسكربت الإدارة (api.js).

أمثلة على حالات الاختبار الوظيفي:

1) **اختبار وظيفة addStudent:**
   - **الهدف:** التأكد من إضافة طالب جديد بنجاح.
   - **الإجراء**: استدعاء الدالة ببيانات طالب فريدة.
   - **النتيجة المتوقعة:** نجاح المعاملة، وعند استدعاء دالة القراءة getStudentDetails بنفس رقم الطالب، تعود البيانات الصحيحة.
2) **اختبار وظيفة addStudent (حالة الفشل):**
   - **الهدف:** التأكد من أن العقد يمنع إضافة طالب بنفس الرقم مرتين.
   - **الإجراء:** استدعاء الدالة برقم طالب موجود مسبقًا.
   - **النتيجة المتوقعة:** فشل المعاملة (Revert) مع رسالة الخطأ " Student with this ID already exists.".
3) **اختبار وظيفة calculateAndVerifyGPA:**
   - **الهدف:** التأكد من أن منطق التحقق من المعدل يعمل بشكل صحيح.
   - **الإجراء**: استدعاء الدالة بقائمة مقررات ومعدل نهائي (providedGPA) يتطابق مع الحساب الصحيح.

- **النتيجة المتوقعة**: نجاح المعاملة وتخزين النتيجة السنوية.

اختبارات التكامل (Integration Tests)

هدفت هذه الاختبارات إلى التأكد من أن جميع طبقات النظام (الواجهة الأمامية، الطبقة الوسيطة، والعقد الذكي) تتواصل وتعمل معًا بسلاسة.

حالة اختبار التكامل الرئيسية (End-to-End):

- **الهدف:** التأكد من أن دورة حياة البيانات الكاملة (من الإنشاء إلى العرض) تعمل بشكل صحيح.
- **الإجراءات المتسلسلة**:
  - تشغيل شبكة ganache-cli.
  - نشر النسخة النهائية من العقد الذكي.
  - تحديث ملفات الإعدادات (contractConfig.js و api.js) بالعنوان والـ ABI الجديدين.
  - تشغيل سكربت api.js لإضافة بيانات كاملة لطالب (إضافة الطالب، المقررات، الخطط، العلامات، وحساب النتيجة السنوية).
  - تشغيل الواجهة الأمامية (npm run dev).
  - في المتصفح، البحث عن رقم الطالب الذي تم إدخال بياناته.
- **النتيجة المتوقعة:** عرض التقرير الأكاديمي الكامل للطالب على الواجهة الأمامية، مع تطابق جميع البيانات المعروضة مع البيانات التي تم إدخالها عبر السكربت.

اختبارات الأمان (Security Tests)

ركزت هذه الاختبارات على التحقق من أن القواعد الأمنية المدمجة في العقد تعمل كما هو متوقع.

أمثلة على حالات اختبار الأمان:

1) اختبار صلاحيات المالك (onlyOwner):

- **الهدف:** التأكد من أن الدوال الحساسة لا يمكن استدعاؤها إلا من قبل مالك العقد.
- **الإجراء**: تكوين سكربت api.js بمفتاح خاص لحساب ليس هو المالك، ومحاولة استدعاء دالة addCourse.
- **النتيجة المتوقعة:** فشل المعاملة (Revert) مع رسالة الخطأ "Caller is not the owner".

2) اختبار التحقق من البيانات (require statements):

- **الهدف:** التأكد من أن العقد يرفض البيانات غير المنطقية.
- **الإجراء:** استدعاء دالة setAcademicMarks مع علامة نهائية (totalGrade_) لا تتطابق مع ناتج الحساب الداخلي للعلامات الجزئية.
- **النتيجة المتوقعة:** فشل المعاملة (Revert) مع رسالة الخطأ " Provided total grade does not match calculation.".

الخاتمة

في ختام هذا المشروع، تم بنجاح تصميم وتنفيذ نظام لامركزي متكامل لإدارة وتوثيق السجلات الأكاديمية بالاعتماد على تقنية البلوك تشين. لقد أثبت التنفيذ العملي قدرة النظام على تحقيق أهدافه الرئيسية في توفير سجل آمن، شفاف، وغير قابل للتغيير للبيانات الأكاديمية، مما يعالج بشكل مباشر التحديات الموجودة في الأنظمة المركزية التقليدية.

تمثلت أبرز مساهمات هذا المشروع في تقديم معمارية مبتكرة تفصل بشكل صارم بين عمليات الكتابة الإدارية وعمليات القراءة العامة، مما يعزز من أمان النظام. كما تم تطبيق نموذج تحقق على السلسلة (On-Chain Verification Model) فريد من نوعه، والذي يضمن سلامة البيانات من خلال التحقق من الحسابات المرسلة من نظام خارجي قبل تخزينها، موازناً بذلك بين كفاءة الحوسبة خارج السلسلة وموثوقية البلوك تشين.

أظهرت نتائج الاختبارات أن جميع مكونات النظام، بدءًا من العقد الذكي، مرورًا بالطبقة الوسيطة، وانتهاءً بالواجهة الأمامية، تتكامل وتعمل معًا بسلاسة لتحقيق الوظائف المطلوبة.

كآفاق مستقبلية، يمكن توسيع هذا العمل ليشمل ميزات أكثر تقدمًا مثل إدارة صلاحيات وصول الطلاب لبياناتهم، والتكامل المباشر مع أنظمة الجامعات الحالية، ونشره على شبكات بلوك تشين عامة لزيادة اللامركزية والوصول العالمي. يمثل هذا المشروع أساسًا قويًا يمكن البناء عليه لتطوير حلول أكثر شمولاً تهدف إلى تعزيز الثقة والشفافية في قطاع التعليم.

المراجع

Budhrani, D., Galphat, Y., & Mulchandani, V. (2018). Student Information Management System. *International Journal of Engineering Development and Research*, *6*(1), 8-10. [i]

Rani, P., Sachan, R. K., & Kukreja, S. (2024). A systematic study on blockchain technology in education: initiatives, products, applications, benefits, challenges and research direction. *Computing*, *106*(2), 405-447. [ii]

Delgado-von-Eitzen, C., Anido-Rifón, L., & Fernández-Iglesias, M. J. (2021). Blockchain applications in education: A systematic literature review. *Applied Sciences*, *11*(24), 11811. [iii]

/http://www.blockcerts.org [iv]

Sudha, V., Kalaiselvi, R., & Sathya, D. (2021, October). Blockchain based student information management system. In *2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)* (pp. 1-4). IEEE. [v]

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. [vi]

Oliva, G. A., Hassan, A. E., & Jiang, Z. M. (2020). An exploratory study of smart contracts in the Ethereum blockchain platform. *Empirical Software Engineering*, *25*(3), 1864-1904. [vii]

Crafa, S., Di Pirro, M., & Zucca, E. (2019, February). Is solidity solid enough?. In *International Conference on Financial Cryptography and Data Security* (pp. 138-153). Cham: Springer International Publishing. [viii]

Wöhrer, M., Zdun, U., & Rinderle-Ma, S. (2021, September). Architecture design of blockchain-based applications. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)* (pp. 173-180). IEEE. [ix]

/https://remix.ethereum.org [x]

https://www.youtube.com/watch?v=KkZ6iYnSDRw&t=440s [xi]