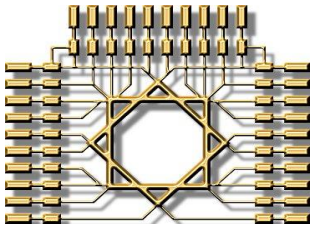


HIAST

المعهد العالي للعلوم االتطبيقية والتكنولوجيا

الجمهورية العربية السورية

السنة الرابعة

نظام حفظ واسترجاع وحدات تعلم الطالب في المعهد العالي باستخدام تقنية ويب3

تقرير مناقشة الكود البرمجي

8 آب 2025

إعداد:

زينب يونس علي

إشراف:

د. مصطفى دقاق

م. عمار مخلوف

جدول المحتويات

الملخص

يقدم هذا المشروع نظامًا لامركزيًا لإدارة وتوثيق السجلات الأكاديمية للطلاب الجامعيين بالاعتماد على تقنية البلوك تشين. تم بناء الواجهة الخلفية للنظام باستخدام عقد ذكي مكتوب بلغة Solidity، وتم نشره واختباره على شبكة تطوير محلية تحاكي بيئة الإيثيريوم (Ganache). العقد مسؤول عن التخزين الآمن والثابت لبيانات الطلاب، الخطط الدراسية، العلامات، والنتائج السنوية بما في ذلك بيانات التخرج.

يتبع النظام معمارية تفصل بين مهام إدخال البيانات وعرضها, حيث تتم محاكاة إدخال البيانات من نظام خارجي باستخدام أداة Postman التي ترسل الطلبات إلى سكربت وسيط (api.js). هذا السكربت يتولى بدوره عملية توقيع المعاملات وإرسالها إلى العقد. بينما تم تطوير واجهة أمامية للعرض فقط باستخدام React و web3.js. يتميز العقد بمنطق تحقق (Verification Model)، حيث يستقبل البيانات المحسوبة مسبقًا ويتحقق من صحتها على البلوك تشين قبل تخزينها، مما يضمن دقة البيانات المسجلة.

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.

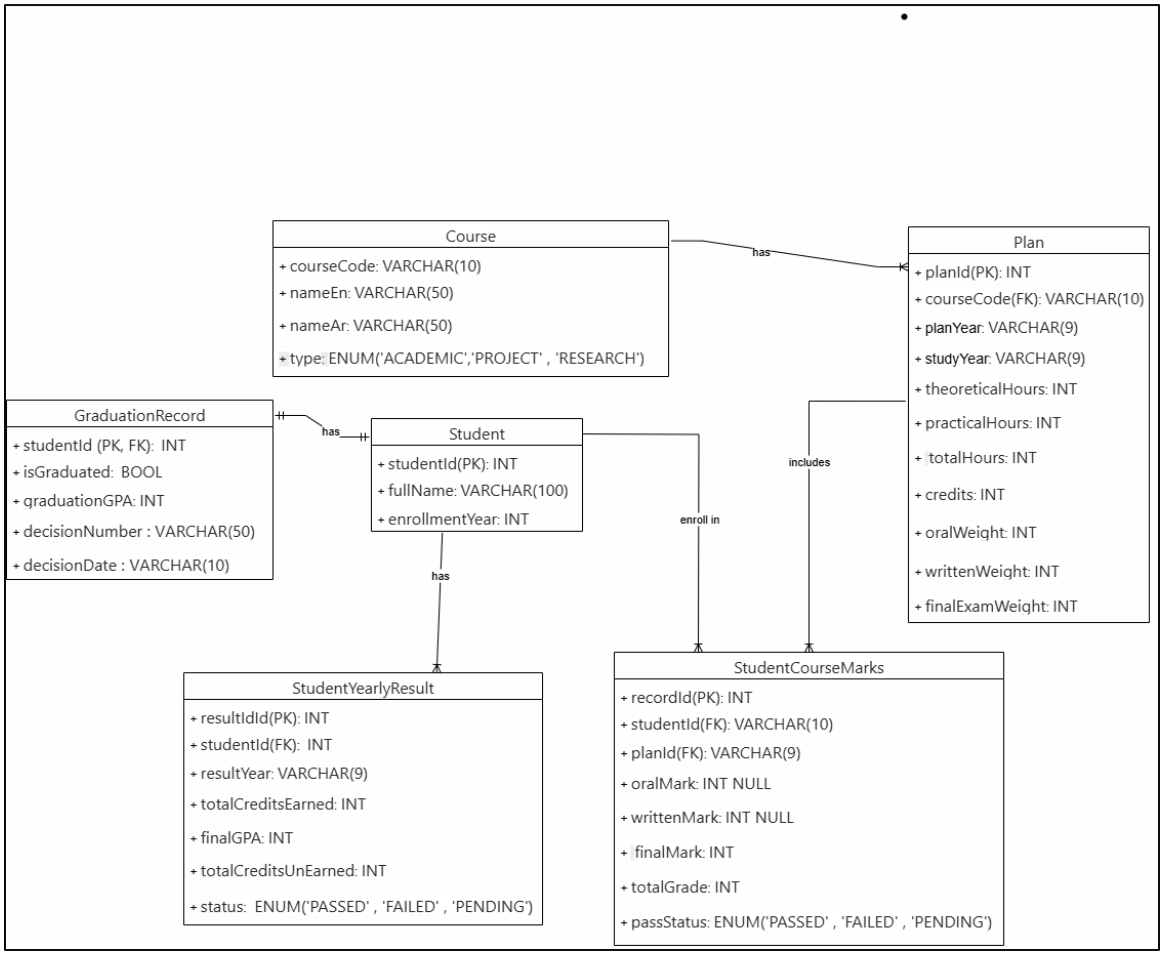
تصميم النظام

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

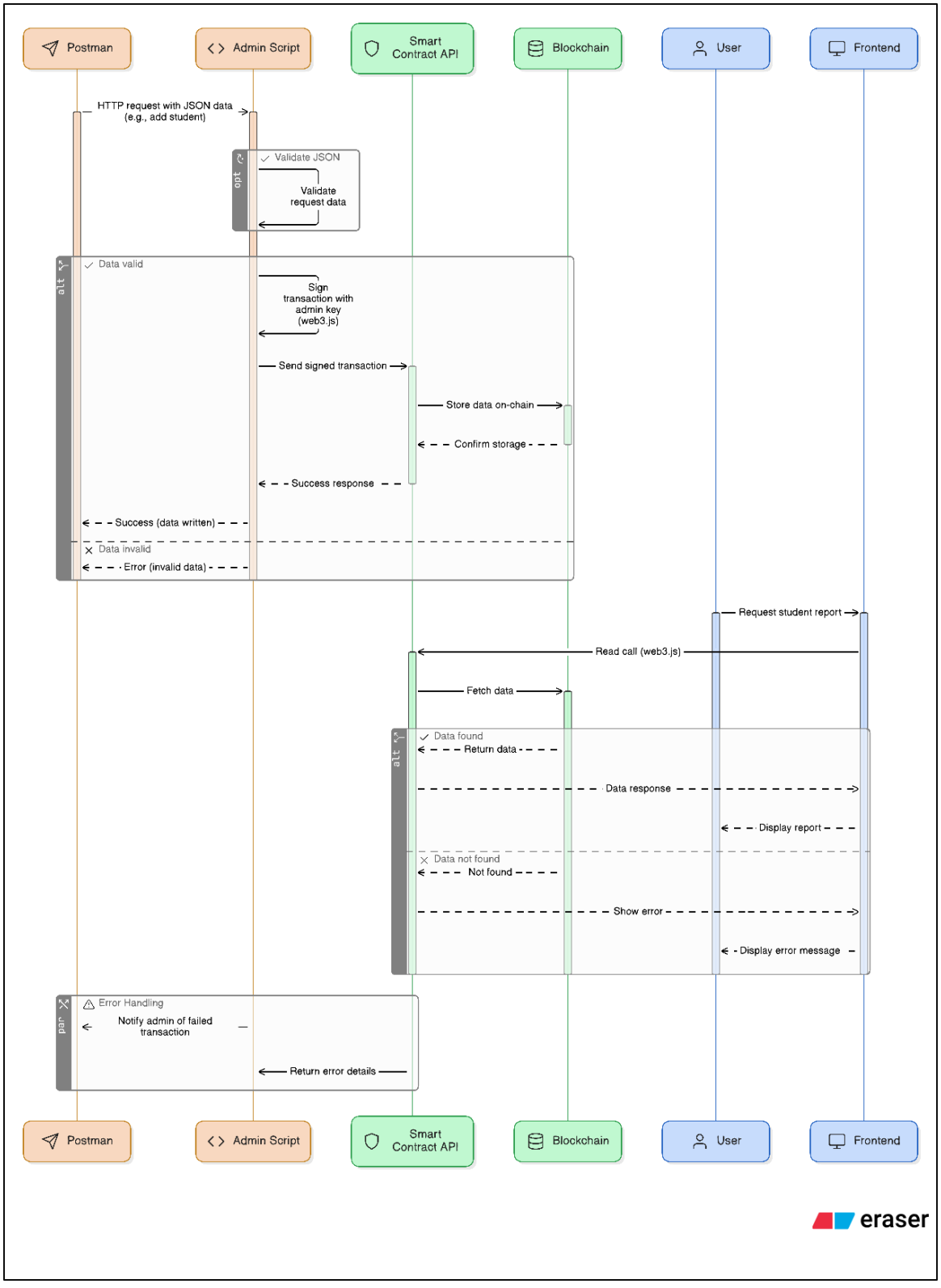
يوضح الشكل (1) مخطط الكيان والعلاقات (ERD) الذي يمثل نموذج البيانات للواجهة الخلفية للنظام. تم تصميم هذا النموذج ليتم تنفيذه باستخدام هياكل البيانات (structs) وآليات التخزين (mappings) في عقد Solidity الذكي. يتكون النموذج من الكيانات الرئيسية التالية:

- **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.

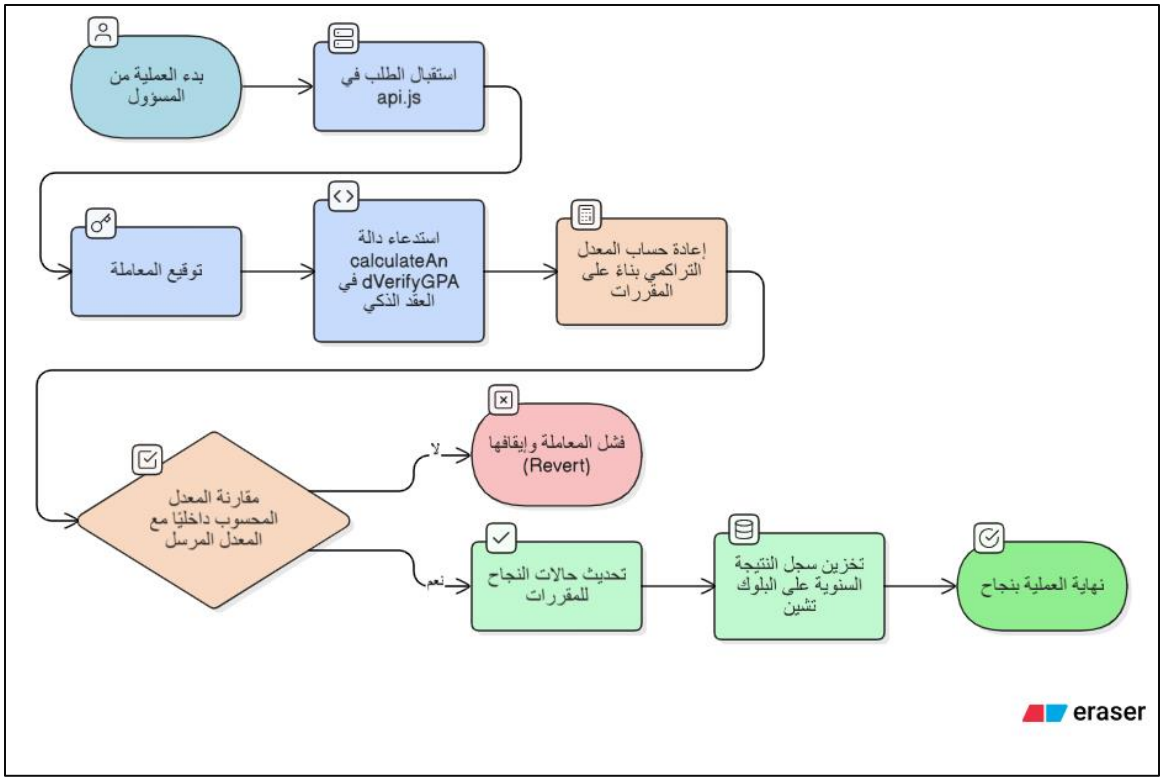
هذه الهيكلية تضمن أن العمليات الحساسة التي تتطلب توقيعًا ومفتاحًا خاصًا معزولة تمامًا في طبقة خلفية آمنة، بينما تظل الواجهة الأمامية بسيطة وآمنة ومخصصة للعرض فقط.

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:

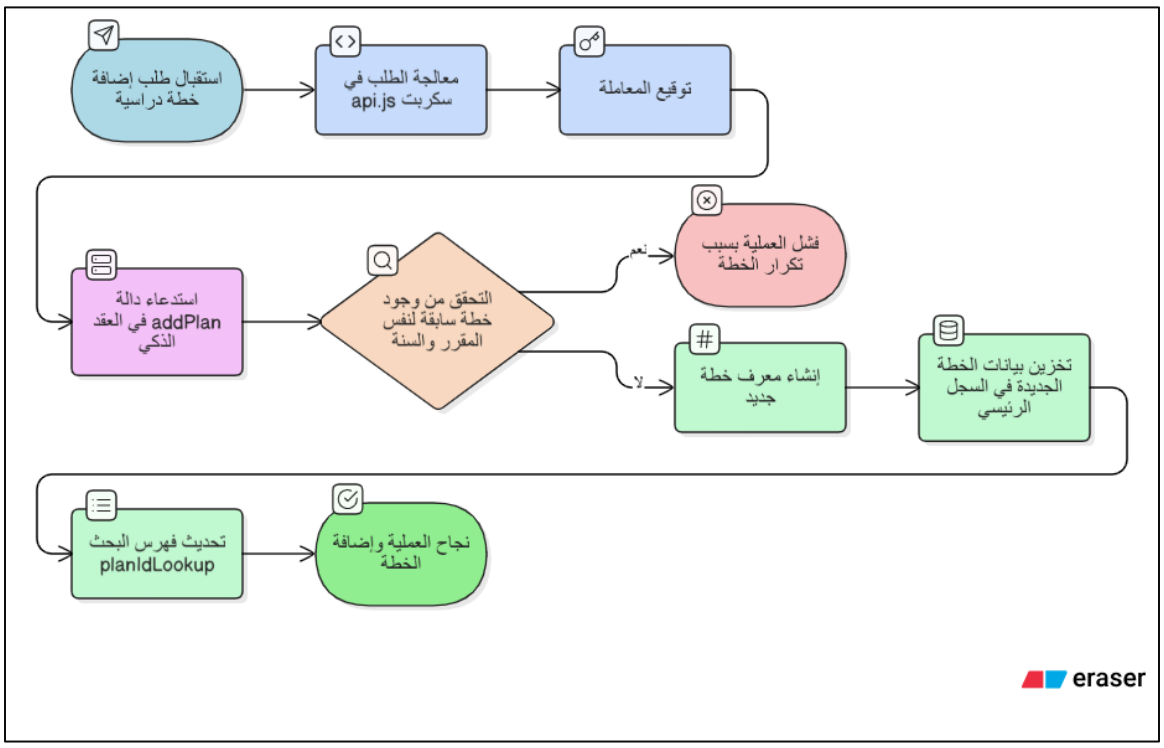
يوضح الشكل (3) مخطط التدفق لعملية التحقق وتخزين النتيجة السنوية، والتي تمثل جوهر نموذج التحقق (Verification Model) المطبق في النظام.

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

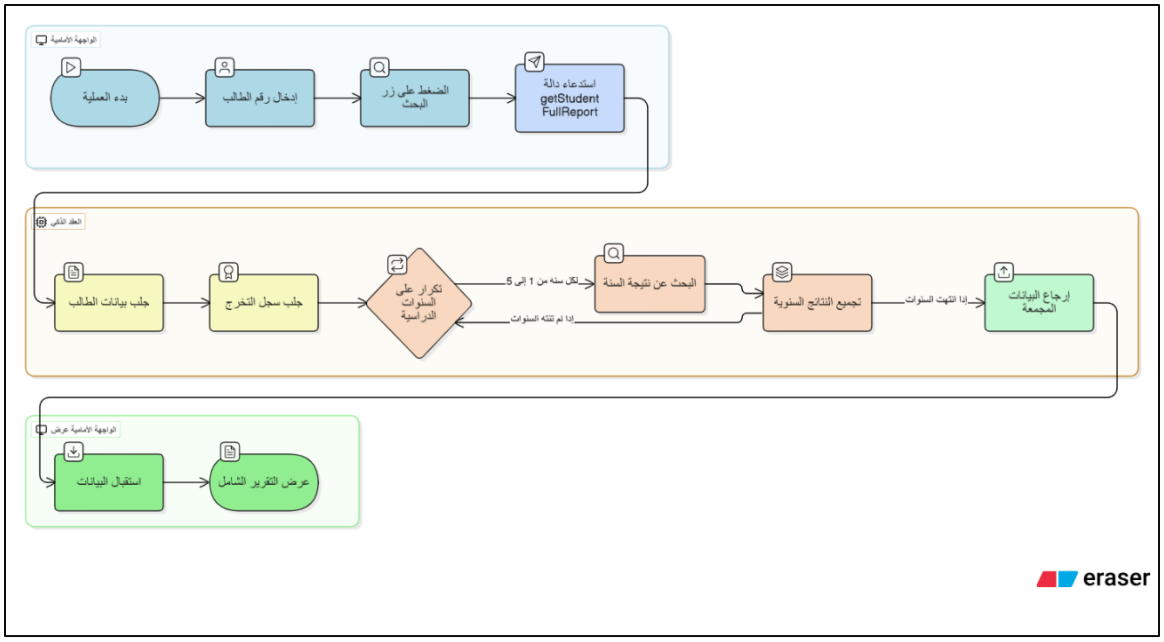
1) (بداية): يتم استقبال طلب إضافة خطة دراسية جديدة.
2) (عملية): يقوم سكربت api.js بمعالجة الطلب وتجهيزه.
3) (عملية): يتم توقيع المعاملة رقميًا باستخدام المفتاح الخاص للمسؤول.
4) (عملية): يتم استدعاء دالة addPlan في العقد الذكي.
5) (قرار): يقوم العقد بالتحقق: هل توجد خطة مسجلة مسبقًا لنفس المقرر والسنة؟
   - (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.

التنفيذ والاختبارات

مقدمة

بعد استعراض تصميم النظام في الفصل السابق، يوثق هذا الفصل الأدوات والتقنيات المستخدمة في عملية التنفيذ، ويشرح بالتفصيل كيفية بناء كل مكون من مكونات النظام، وآلية التكامل التي تضمن تواصل هذه المكونات مع بعضها البعض بشكل فعال وآمن. أخيرًا، يستعرض الفصل خطة الاختبارات التي تم اتباعها للتأكد من أن كل جزء من النظام يعمل كما هو متوقع ويحقق المتطلبات المحددة.

معمارية النظام المتبعة وأدوات التنفيذ

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.

- **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.

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.

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.

خطة الاختبارات (Testing Plan)

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

لقد تم بنجاح بناء وتكامل جميع المكونات الأساسية، بدءًا من العقد الذكي الذي يمثل الواجهة الخلفية الآمنة، مرورًا بالطبقة الوسيطة المسؤولة عن إدخال البيانات، وانتهاءً بالواجهة الأمامية المخصصة للعرض والتحقق.
أثبتت الاختبارات أن النظام يحقق أهدافه الرئيسية في توفير سجل موثوق وغير قابل للتغيير للبيانات الأكاديمية، مع تطبيق نموذج تحقق فعال يضمن سلامة البيانات. يمثل هذا التنفيذ أساسًا متينًا يمكن البناء عليه مستقبلًا لتوسيع قدرات النظام ودمجه مع تطبيقات لامركزية أخرى.