cct | College Dublin
Computing • IT • Business

## Assessment Cover Page

***Student Full Name*** : Ozan Baran Bozdag

***Student Number*** : 2023382

***Module Title*** : **Distributed Digital Transactions**

***Assessment Title:*** *CA2 Individual*

***Lecturer/Supervisor*** : Dr. Muhammad Iqbal

***Assessment Due Date*** : Sunday, 4 January 2026, 11:59 PM

***Date of Submission*** : Saturday, 3 January 2026

### Use of AI Tools

I acknowledge the use of ChatGPT for structuring the report, summarising sources, checking grammar, and helping troubleshoot coding issues in Solidity.

# Contents

Introduction ..... 1

1. Question 1: Background and Concept of Blockchain-based Legal Escrow ..... 3
2. Question 2: Smart Contract Implementation and Testing ..... 5
    3.1 Smart Contract Design ..... 6
    3.2 Deployment and Transactions (Screenshots) ..... 7
3. Question 3: System Architecture Diagram ..... 11
4. References..... 13
5. Reflection..... 14

# Question 1: Blockchain Technology in Legal Services

Blockchain technology is a distributed digital ledger that allows data to be recorded across multiple nodes in a secure and tamper-resistant manner. Unlike traditional centralised databases, blockchain does not rely on a single authority to manage or verify records. Instead, transactions are validated through consensus mechanisms, making the system more transparent and resistant to unauthorised modification (Zheng et al., 2018).

In the context of legal services, blockchain has the potential to enhance how client records, case information, and legal documents are managed. For a law firm such as ReliableLaw Solicitors, blockchain could be used to create immutable records of client interactions, document submissions, and case milestones. For example, hashing legal documents and storing their hashes on a blockchain can help verify document integrity without exposing the actual content. This ensures that documents have not been altered, while still maintaining confidentiality.

Compared to traditional legal information management systems, blockchain offers several advantages. Traditional systems typically rely on centralised databases that can be altered by administrators or become vulnerable to single points of failure. Blockchain, by contrast, provides a permanent audit trail, which improves transparency and accountability. This can be particularly valuable in dispute resolution or compliance audits, where an accurate and verifiable history of actions is required (Mougayar & Buterin, 2016).

However, blockchain adoption also presents limitations. One major challenge is data privacy. Legal services involve highly sensitive client information, and storing personal data directly on an immutable ledger may conflict with data protection requirements. Additionally, blockchain systems can introduce scalability and cost issues, particularly when public blockchains are used. Errors recorded on a blockchain are also difficult to correct, which may be problematic in legal environments where amendments are sometimes necessary.

From a social and ethical perspective, blockchain may increase client trust by improving transparency and reducing opportunities for data tampering. At the same time, law firms must ensure that blockchain solutions do not undermine client confidentiality or professional ethical obligations. Trust in legal services depends not only on transparency but also on responsible data handling.

Legally, compliance with the General Data Protection Regulation (GDPR) is a significant concern. GDPR principles such as data minimisation and the right to erasure can conflict with blockchain's immutability. As highlighted by Finck (2018), organisations adopting blockchain must carefully design their systems to avoid storing personal data on-chain and instead rely on off-chain storage with on-chain references. Law firms in Ireland must also comply with the professional standards set by the Law Society of Ireland, particularly regarding confidentiality and data protection.

In conclusion, while blockchain technology offers meaningful benefits for transparency, security, and efficiency in legal services, its adoption must be carefully planned. A hybrid approach that combines blockchain with traditional systems may offer the most practical solution for law firms seeking innovation without compromising legal and ethical responsibilities.

# Question 2 – Smart Contracts in Legal Services

Smart contracts are programs that run on a blockchain and automatically carry out actions when certain conditions are met. Unlike traditional contracts, which require manual enforcement and trusted intermediaries, smart contracts operate based on predefined logic written directly into code. This makes them particularly useful in legal and financial processes where trust, accuracy, and efficiency are important.

One of the main design principles of smart contracts is **deterministic execution**. This means that the contract will always behave in the same way when the same conditions are met. In the implemented escrow contract, actions such as depositing funds, releasing payment, or issuing a refund always produce predictable outcomes. This is important in legal services, as all parties involved can clearly understand what will happen and under what circumstances.

Another important principle is **immutability**. Once a smart contract is deployed to the blockchain, its code cannot be changed. This prevents either party from modifying the agreement after deployment. In a legal escrow scenario, immutability helps build trust between the client and the solicitor, as both sides know that the rules governing the transaction are fixed and cannot be manipulated.

**Autonomy and self-execution** is another key feature of smart contracts. After deployment, the contract operates independently without the need for manual intervention. In this project, once the client deposits funds into the escrow, the contract itself controls whether the funds are released to the solicitor or refunded to the client. This reduces reliance on third parties and helps minimise delays and administrative errors.

The principle of **transparency and verifiability** also plays an important role. All interactions with the smart contract are recorded on the blockchain and can be viewed and verified. This provides a clear audit trail showing when funds were deposited, released, or refunded. In legal services, such transparency can be useful in resolving disputes and demonstrating compliance with agreed procedures.

To demonstrate these principles in practice, a prototype escrow smart contract was developed using Solidity and tested in the Remix IDE. The contract defines two roles: a client and a solicitor. The client is responsible for depositing funds and approving payment release, while the solicitor can refund the client if the transaction is cancelled. Access control mechanisms ensure that only authorised parties can perform specific actions.

Multiple test scenarios were carried out to validate the behaviour of the smart contract. These included successful and failed deposit attempts, successful and failed fund release attempts, and a successful refund scenario. Each test confirmed that the contract correctly enforces its rules and responds appropriately to both valid and invalid actions. Screenshots of these tests are provided as evidence of correct functionality.

Overall, this smart contract demonstrates how blockchain technology can be applied to legal services to improve trust, efficiency, and accountability. By combining deterministic logic, immutability, transparency, and automated execution, smart contracts offer a practical solution for managing escrow agreements in a secure and reliable manner.

# Smart Contract Implementation and Testing Evidence

*Figure 1 – Code compiled*

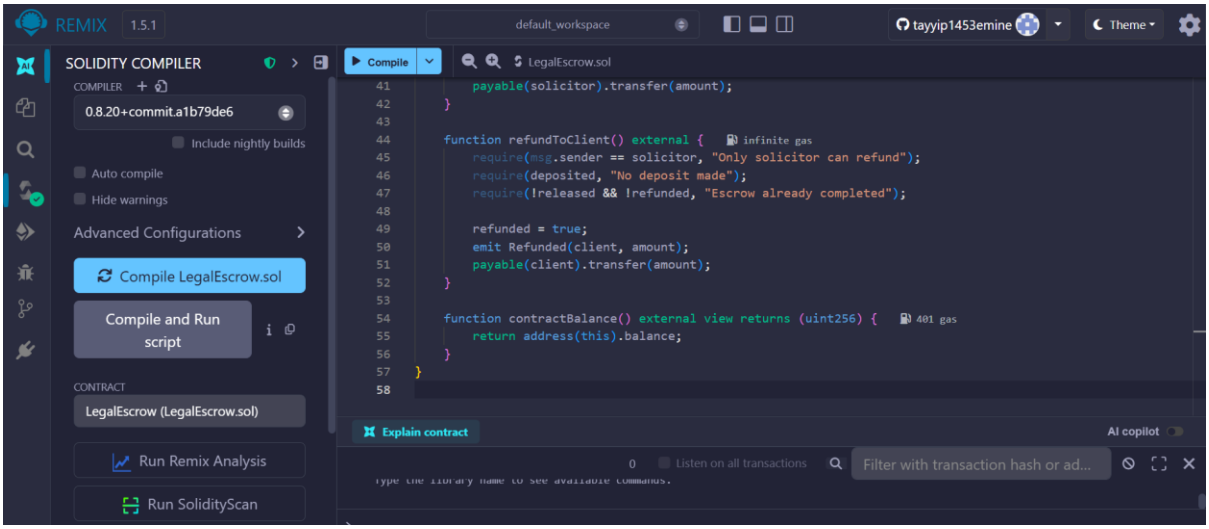



*Figure 2 – Deploy success*

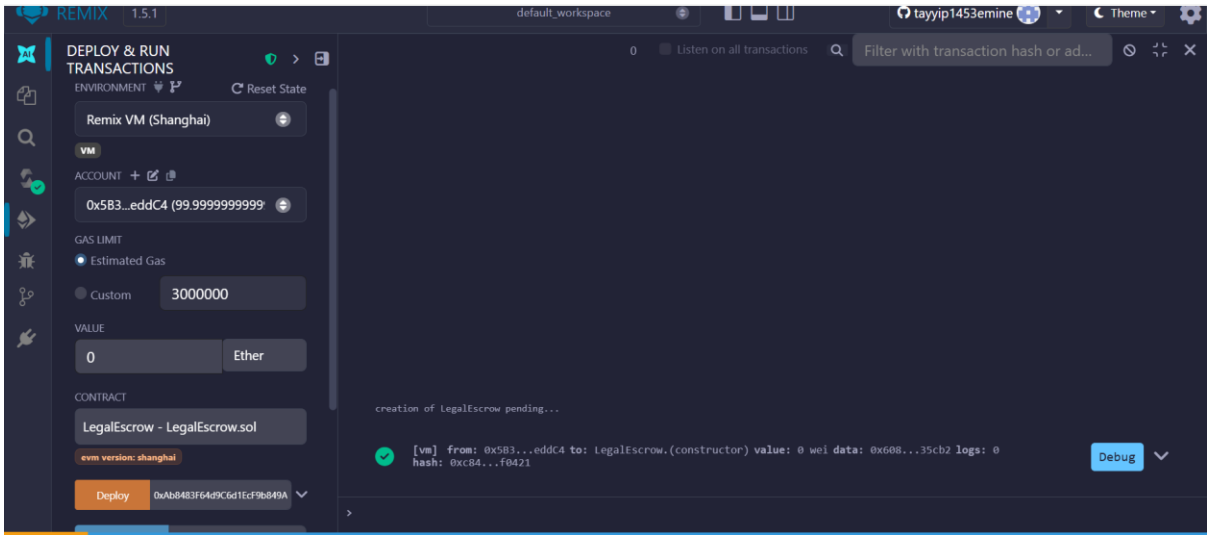


*Figure 3 – Deposit success*

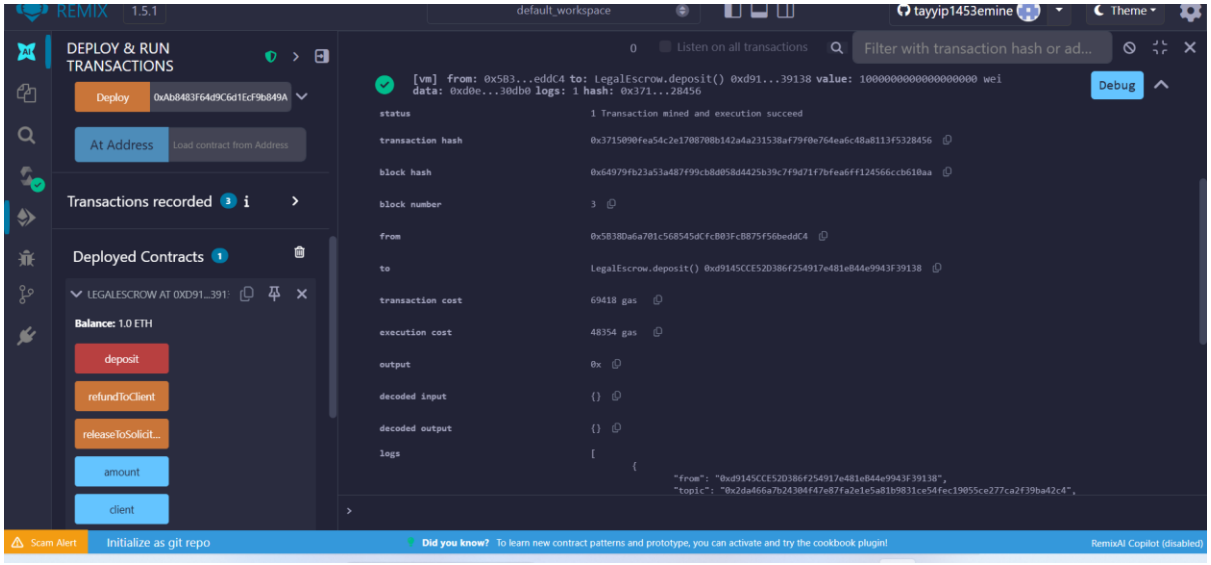



*Figure 4 – Deposit fail*

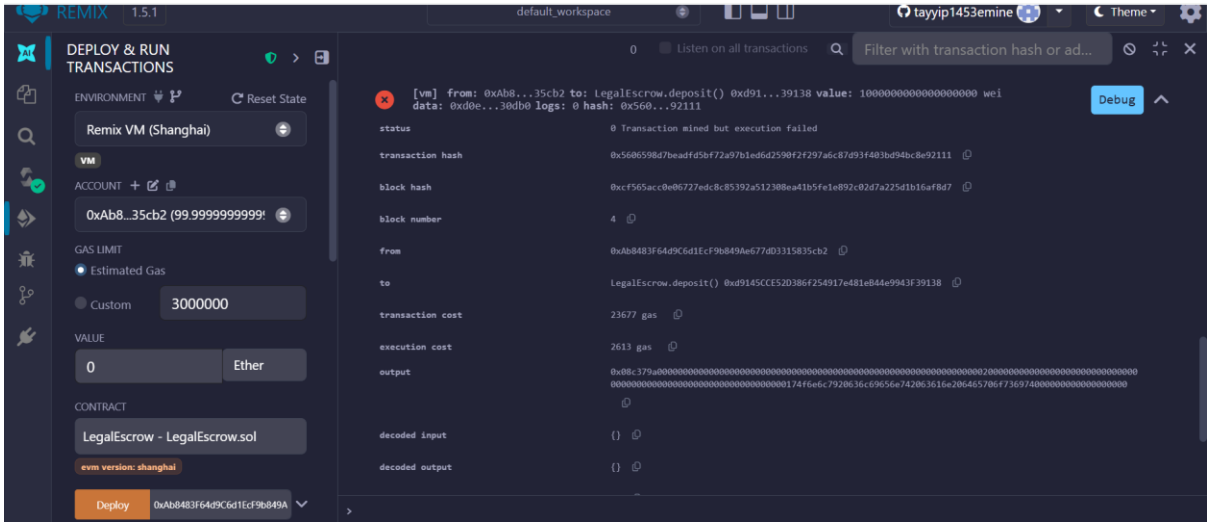


*Figure 5 – Release success*

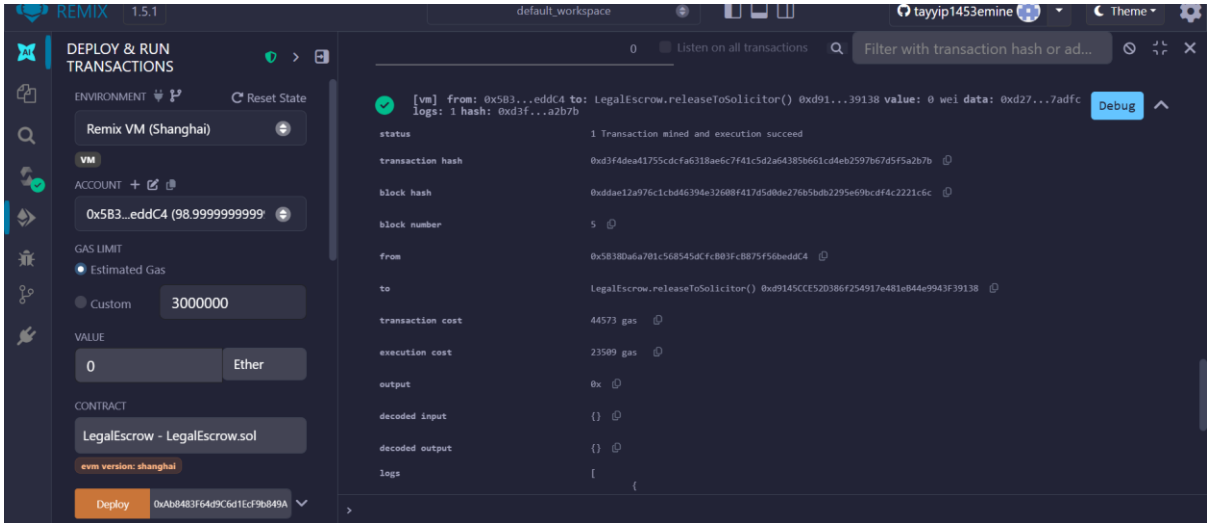
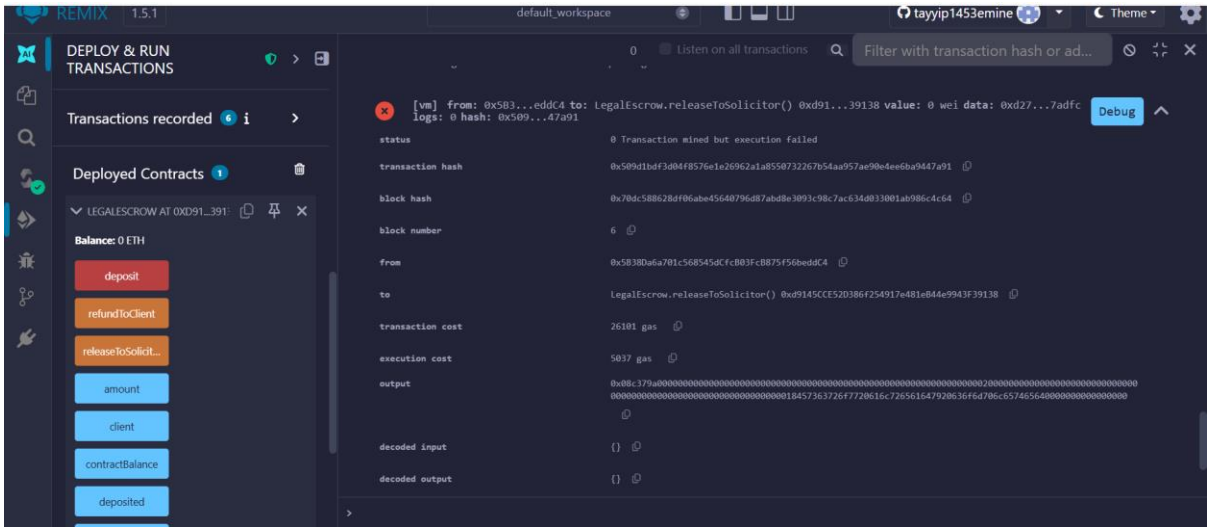
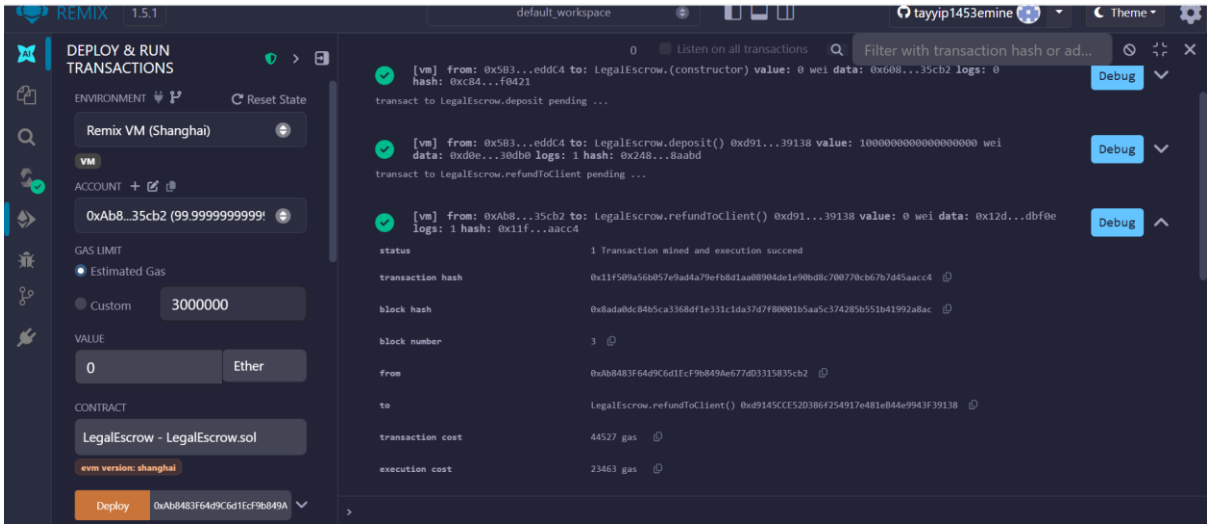



*Figure 6 – Release fail*



- Figure 7 – Refund success



# Question 3: Blockchain Adoption and Client Satisfaction

The diagram illustrates a high-level architecture of a blockchain-based system designed to improve client satisfaction within legal services. One of the main concerns in legal practice is **client privacy**. In this model, sensitive legal documents are stored off-chain in secure storage rather than directly on the blockchain. This ensures that confidential client information is not publicly exposed, while still allowing the system to benefit from blockchain technology.

To maintain **document integrity**, a cryptographic hash of each legal document is generated and recorded on the blockchain ledger. This hash acts as a digital fingerprint, allowing any future changes to the document to be detected. By storing only the hash on the blockchain, the system ensures that documents remain tamper-resistant without compromising privacy.

The **blockchain smart contract** plays a central role in controlling access and managing interactions between the client and the solicitor. Only authorised parties can perform actions such as approving transactions or requesting verification. All actions are recorded on the blockchain ledger, creating an immutable and transparent audit trail.

From a regulatory perspective, this approach supports compliance with **GDPR and the Irish Data Protection Act 2018**. By avoiding the storage of personal data on-chain and relying on off-chain storage with blockchain references, the system aligns with data minimisation and confidentiality requirements. Overall, this design enhances transparency, trust, and security, leading to improved client confidence and satisfaction in legal services.

## High-level architecture of the blockchain-based legal escrow system

# References

Finck, M. (2018) *Blockchain and the General Data Protection Regulation: Can distributed ledgers be squared with European data protection law?* European Parliament Research Service. Available at: https://www.europarl.europa.eu/thinktank/en/document/EPRS_STU(2018)604972 (Accessed: 4 January 2026).

Mougayar, W. and Buterin, V. (2016) *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. Hoboken: John Wiley & Sons.

Szabo, N. (1997) *Formalizing and Securing Relationships on Public Networks*. First Monday, 2(9). Available at: https://firstmonday.org/ojs/index.php/fm/article/view/548 (Accessed: 4 January 2026).

Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H. (2018) *Blockchain challenges and opportunities: A survey*. IEEE Transactions on Industrial Informatics, 14(8), pp. 3523–3532. doi: 10.1109/TII.2018.2848937.

Wood, G. (2014) *Ethereum: A secure decentralised generalised transaction ledger*. Ethereum Yellow Paper. Available at: https://ethereum.github.io/yellowpaper/paper.pdf (Accessed: 4 January 2026).

European Union (2016) *General Data Protection Regulation (EU) 2016/679*. Official Journal of the European Union. Available at: https://eur-lex.europa.eu/eli/reg/2016/679/oj (Accessed: 4 January 2026).

# Appendix: Use of Artificial Intelligence Reflection

During the completion of this assignment, artificial intelligence tools were used as a supportive learning aid rather than as a replacement for my own work. AI was mainly used to help me better understand complex concepts, improve the structure of my work, and reflect on areas where I was making repeated mistakes.

Firstly, AI was used to assist with planning the assignment. I asked for guidance on how to break the project into manageable stages, how much time to allocate to each question, and how to organise the work across several days. This helped me approach the assignment in a more structured way and reduced unnecessary stress.

AI was also helpful during the research process. Instead of directly copying information, I used it to summarise and clarify ideas from different academic and industry sources. This encouraged me to explore a wider range of materials and develop a better understanding of blockchain technology, legal services, and regulatory considerations. The summaries helped me identify key points, which I then rephrased and integrated into my own writing.

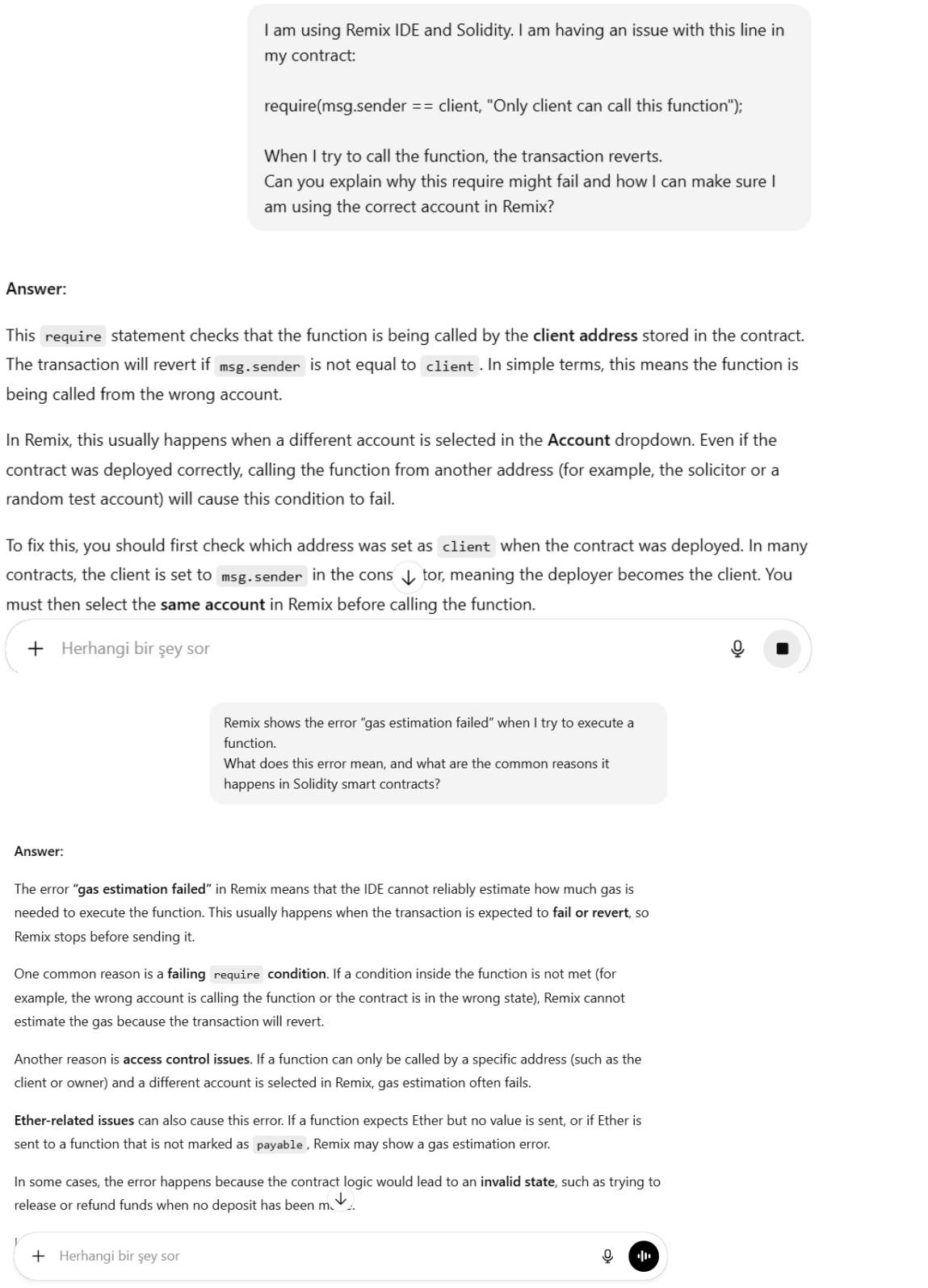
In terms of writing and presentation, AI was used to improve page layout, formatting, and consistency of headings. It also assisted with grammar and sentence clarity, particularly to ensure that the language remained clear and suitable for an academic report. These corrections did not change the meaning of my work but helped improve readability.

During the coding phase, AI was used as a troubleshooting tool. When errors occurred in the Solidity smart contract, I asked questions to understand why the errors happened, what they meant, and where I could learn more about the underlying concepts. This helped me identify mistakes, such as incorrect function usage or deployment issues, and improve my understanding of smart contract development rather than simply fixing errors without learning.

Overall, AI was used as a learning companion to support understanding, organisation, and problem-solving. All core ideas, implementation decisions, and final written content were developed by me, with AI serving only as guidance and clarification throughout the process.

# Appendix: Evidence of AI Tool Usage

## *A. AI Support for Code Understanding and Debugging*



I am using Remix IDE and Solidity. When I try to call a function, the transaction fails with “transaction reverted”.
Can you explain what this error usually means in simple terms and what I should check first in my smart contract?

**Answer:**

The error **“transaction reverted”** means that the smart contract has intentionally stopped the transaction because one of its conditions was not met. In Solidity, this usually happens when a `require` statement fails or when the contract logic prevents an action from being executed.

The first thing to check is any **require conditions** in the function you are calling. These conditions are often used to check things like whether the correct account is calling the function (`msg.sender`), whether enough Ether was sent (`msg.value`), or whether the contract is in the correct state.

You should also check **which account is selected in Remix**. If the function is restricted to a specific role (for example, only the client or only the owner), calling it from the wrong account will cause the transaction to revert.

Another common cause is **sending Ether incorrectly**. If a function expects Ether but is not marked as `payable`, or if no Ether is sent when it is required, the transaction will fail.

In summary, when you see a “transaction reverted” error, you should first check the `require` statements, confirm the correct account is being used, and verify that Ether and function conditions match what the





My contract has this constructor:

```
constructor(address _solicitor) {
    solicitor = _solicitor;
}
```

When deploying in Remix, I am not sure what address to enter.
Can you explain what address should be used here and how it affects msg.sender and permissions later?

**Answer:**

This constructor takes an address parameter called `_solicitor` and assigns it to the `solicitor` variable in the contract. The address you enter during deployment will define **who the solicitor is** for the entire lifetime of the contract.

When deploying the contract in Remix, you should enter one of the **valid Ethereum addresses** shown in the Remix **Account** dropdown. Typically, this should be a **different address from the deployer** if you want to clearly separate roles. For example, you can deploy the contract using Account 0 (client) and enter Account 1 as the `_solicitor` address.

It is important to understand that `msg.sender` inside the constructor refers to the **deployer of the contract**, not the `_solicitor`. This means that if your contract sets `client = msg.sender` in the constructor, the deployer becomes the client, while the `_solicitor` address you enter becomes the solicitor.

Later, when functions use conditions such as `require(msg.sender == solicitor)`, only the address you

I am having a problem with this line in my Solidity contract:

require(msg.value > 0, "Deposit must be greater than zero");

The transaction fails when I call deposit().
Can you explain how msg.value works and what I need to set in Remix to make this condition pass?

**Answer:**

The `require(msg.value > 0)` condition checks that the transaction is sending **Ether** to the smart contract. The transaction will revert if no Ether is included, meaning `msg.value` is equal to zero.

In Solidity, `msg.value` represents the amount of Ether (in wei) sent with the function call. If you call `deposit()` without sending any value, this condition will fail and the transaction will revert with the message **"Deposit must be greater than zero."**

To make this condition pass in Remix, you must do two things. First, ensure that the `deposit()` function is marked as `payable`, otherwise it cannot receive Ether. Second, before calling the function, set a value in the **Value** field in Remix (for example, `1 ether`) and make sure the unit is correct.

In summary, this error happens when Ether is not sent with the transaction. Selecting the correct account and setting a non-zero value in Remix will allow the `require(msg.value > 0)` condition to pass successfully.



I have this condition in my escrow contract:

require(!released && !refunded, "Escrow already completed");

The function reverts with this message.
Can you explain how state variables like released and refunded work, and why this condition might fail?

**Answer:**

This `require` condition checks the **state of the escrow contract**. The variables `released` and `refunded` are boolean state variables that track whether the escrow process has already been completed. The function will only run if **both values are false**.

The transaction reverts with the message **"Escrow already completed"** when either `released` or `refunded` is already set to `true`. This usually means that the escrow has already been finished and the contract is preventing the same action from happening again.

In an escrow contract, these state variables are important to avoid **double actions**, such as releasing funds twice or refunding after funds have already been released. For example, once `released` is set to `true`, calling the release function again should fail, and the same applies to refunds.

This condition may fail if:

- The release function was already called successfully.
- The refund function was already executed.
- The state variables were not reset correctly after deployment or testing.

## B. AI Support for Writing, Grammar, and Source Summarisation

bunlari cektim yeter li mi

Kısa cevap: **EVET, YETERLİ — hatta fazlası var.** 👍
Şimdi netleştiriyorum ki rapora koyarken kafan karışmasın.

---

### ✅ Q2 İÇİN KULLANMAN GEREKENLER (SEÇİLMİŞ SET)

Aşağıdaki **7 screenshot Q2 için TAMAMEN yeterli**:

**1️⃣ Kod + Compile**

- `LegalEscrow.sol` açık
- "Compiled successfully" görünüyor
  ➡️ *Kodun var ve hatasız derlenmiş*

bu kalan kisimlar ne kadar surecek sence

Kısa ve dürüst cevap vereyim 👍
**Kod kısmı bittiği için işin %60–65'i zaten tamam.**
Kalanları **normal tempoyla** şu sürede bitirirsin:

---

### ⏱ GERÇEKÇİ ZAMAN TAHMİNİ

🔹 **Q2 – Kodun açıklaması (kolay)**

- **Süre:** 30–45 dakika
- Zor değil, çünkü kodu **zaten yaptın**
- Screenshot altlarına kısa açıklamalar

---

🔹 **Q3 – Diagram + açıklama**

- Diagram çizmek: **20–30 dakika**
  - El çizimi bile yeter
- Açıklama yazısı: **20 dakika**

➡️ **Toplam:** ~45–60 dakika

---

🔹 **Q1 – Teorik kısım (en uzun)**



do i need to take screen shot of the code structure

Short answer: **YES — but keep it simple.**
Long answer below so you don't overdo it 👇

---

## ✅ Do you need a screenshot of the code structure?

**Yes, you should take ONE screenshot** showing:

- the **Solidity file open** in Remix
- the **contract code visible**
- preferably the **contract name + comments at the top**

This helps prove:

- you wrote the code
- the structure is clear
- the examiner can quickly see what the contract does

But ⚠️ **you do NOT need**:

- multiple code screenshots
- line-by-line zoomed images
- screenshots of every function separately

**One clean screenshot is enough.**

Summary (Finck, 2018 – Blockchain and GDPR): summarise it please but keep the main structure

**Summary (Finck, 2018 – Blockchain and GDPR):**

Finck (2018) analyses the compatibility of blockchain technology with the General Data Protection Regulation (GDPR). The source is structured around the tension between blockchain's technical features and GDPR's legal requirements, particularly in relation to personal data protection.

The first key issue discussed is **immutability**. Blockchain records are designed to be permanent, which conflicts with GDPR rights such as the right to rectification and the right to erasure. Finck explains that once personal data is written to a blockchain, it is extremely difficult or impossible to remove or modify.

The source then examines **data protection principles**, including data minimisation and purpose limitation. Finck argues that storing personal data directly on a distributed ledger may violate these principles, as blockchain systems replicate data across multiple nodes and limit control over how data is used.

Another important topic is **responsibility and accountability**. The paper highlights the difficulty of identifying data controllers and processors in decentralised blockchain systems, which creates legal uncertainty under GDPR.

Finally, Finck proposes **off-chain storage** as a practical solution. In this model, personal data is stored outside the blockchain in controlled environments, while only [illegible]shes or references are recorded on-chain. This approach allows organisations to benefit from blockchain's integrity and transparency while remaining more

solicitor can refund the client if the transaction is cancelled. Access control ensure that only authorised accounts can call specific functions.

Several test scenarios was carried out to validate the behaviour of the contract. These include successful and failed deposit attempts, successful and failed release attempts, and a refund scenario. The results shows that the contract correctly enforces the rules and respond as expected. Screenshots are provided as evidence.

Overall, this smart contract shows how blockchain technology can be applied in legal services to improve trust, efficiency and accountability. correct grammar please

Tabii 👍 işte **grameri düzeltilmiş**, ama **tonu hâlâ B2–C1 öğrenci seviyesi** olan versiyon.
Anlam **hiç değişmedi**, sadece dil temizlendi.

---

## Question 2 – Smart Contracts in Legal Services (corrected version)

Smart contracts are programs that run on a blockchain and automatically execute actions when certain conditions are met. Unlike traditional contracts, which require manual enforcement and trusted third parties, smart contracts work based on logic that is written directly into code. This makes them very useful in legal and financial processes where trust and accuracy are important.

One of the main design principles of smart contracts [illegible]eterministic execution. This means that the contract will always behave in the same way when the same conditions are met. In the implemented escrow contract,

Herhangi bir şey sor

Nick Szabo (1997) – Smart Contracts i like this document but is too long can you summarise it for me but please let me understand all main topics and structure of the source

**Summary:**

Nick Szabo's (1997) work introduces the concept of **smart contracts** as digital protocols that automatically execute and enforce the terms of an agreement. The main idea is to reduce the need for trusted intermediaries by embedding contractual rules directly into computer code. Szabo explains that traditional contracts rely heavily on legal enforcement and manual processes, which can be slow, costly, and prone to disputes.

The source is structured around the problem of **trust in digital interactions**. Szabo argues that smart contracts can increase trust by making agreements more secure, predictable, and resistant to manipulation. He compares smart contracts to everyday mechanisms such as vending machines, where rules are clear and outcomes are automatic once conditions are met.

Another key topic in the document is **security and reliability**. Szabo highlights how cryptography and computer protocols can be used to protect contractual relationships. By using secure systems, smart contracts ensure that agreed actions are carried out exactly as specified, reducing the risk of fraud or misinterpretation.

Herhangi bir şey sor



diagrami yapalim kral nereden yapalim

Tamam kral 👍 **en hızlı, en risksiz yolu** söylüyorum.
Bu Q3'ü **10–15 dakikada** bitireceğiz.

---

## ✅ EN İYİ SEÇENEK: Word veya PowerPoint (Shapes ile)

### Neden?

- ✔ AI değil
- ✔ Online görsel değil
- ✔ "Student-made diagram" hissi var
- ✔ Hocalar bunu **çok net kabul eder**

👉 **PowerPoint biraz daha rahat**, ama Word de olur.
Ben **PowerPoint öneriyorum**, sonra resmi Word'e atarsın.

---

## 🛠️ SEÇENEK A (ÖNERİLEN): PowerPoint

### 1️⃣ PowerPoint aç

- Yeni boş slide
- Title falan koyma, **boş kalsın**

### 2️⃣ Sıraya git:

Can you summarise the key points of this source in simple academic language, focusing on how blockchain can be used in legal services and the main GDPR concerns?

**Summary:**

Blockchain can be used in legal services to improve transparency, trust, and the security of records. By using a distributed ledger, law firms can create tamper-resistant records of activities such as document handling, contract execution, and escrow payments. This provides a clear and verifiable audit trail, which can be useful for compliance, dispute resolution, and accountability.

Despite these benefits, blockchain raises important GDPR concerns. One major issue is immutability, as GDPR allows individuals to request correction or deletion of personal data, which conflicts with permanent blockchain records. Data minimisation is another concern, since legal data is often sensitive and should not be publicly accessible.

To manage these issues, blockchain systems in legal services often rely on off-chain storage. Sensitive legal documents are stored securely outside the blockchain, while only a cryptographic hash or reference is recorded on-chain. This approach helps maintain data integrity while supporting GDPR compliance and protecting client confidentiality.