



Marmara University  
Faculty of Engineering

CSE 4067  
INTRODUCTION TO BLOCKCHAIN PROGRAMMING

---

## Rental Agreement and Eviction System

---

Instructure: Goshgar Can Ismayilov Date:07.01.2025

	Department	Student Id Number Name & Surname
1	CSE	150120022 Tolga Fehmioğlu
2	CSE	150121538 Muhammed Enes Gökdeniz
3	CSE	130319659 Cihan Erdoğanılmaz

## **1: Problem Definition:**

In Turkey, rent disputes frequently arise, often leading to prolonged court processes. Although mediation was introduced to streamline resolution, it has not achieved the desired efficiency. The government has proposed tracking lease agreements through centralized platforms like E-Government; however, concerns over cybersecurity, data consistency, and central database reliability limit its effectiveness. This project proposes a blockchain-based solution, confined to the Turkish Code of Obligations, to create a smart contract for lease agreements that automates enforcement and ensures contract execution in disputes. Blockchain is necessary as it provides trustless, decentralized data management, removing the need for a third-party intermediary and allowing state intervention only when strictly required.

## **2: Motivation and Objectives:**

### **Motivation:**

Traditional lease agreement management, primarily paper-based, is often inefficient, prone to delays, and lacks transparency. The need for frequent reliance on intermediaries like legal representatives and courts leads to high transaction costs and prolonged dispute resolutions. Although recent initiatives such as mediation and the proposal of centralized digital platforms (e.g., E-Government) aim to streamline agreement tracking, they have not fully addressed the core issues, including trust deficits, data security concerns, and enforcement challenges.

Blockchain technology, with its decentralized and immutable structure, has the potential to transform lease management by offering a secure, transparent, and efficient framework that eliminates the need for intermediaries. This project integrates blockchain with advanced natural language processing (NLP) to automate the conversion of complex lease terms into smart contracts, embedding enforceable legal compliance directly into digital contracts.

This innovation could redefine lease agreement processes by reducing human error, enhancing trust between tenants and landlords, and expediting dispute resolution while aligning with statutory requirements. Through blockchain and NLP, the project aims to establish a cutting-edge system that delivers robust, autonomous, and legally compliant contract management.

### **Objectives:**

This project aims to build a decentralized system to automate the creation, enforcement, and management of lease agreements on the blockchain. Key objectives include:

- **LLM-Driven Smart Contract Formation:** Develop an innovative LLM-based pipeline to interpret and transform lease agreement terms into smart contracts. By uploading a lease contract, users can generate a smart contract that accurately reflects the terms, as outlined in the Turkish Code of Obligations, ensuring legal alignment and enhancing user confidence in automated contract enforcement.
- **Token Integration:** Integrate ERC-721 tokens to manage rental transactions and uniquely represent each lease agreement, respectively. This will enable secure monthly rent payments, rental price adjustments, and verification of tenant-landlord relationships on-chain.
- **Decentralized Enforcement:** Design a mechanism within the smart contract for enforcement processes, including controlling eviction conditions, cautioning parties on payment dues, and validating the execution of the agreement with state oversight when needed.
- **Rental Price Adjustment:** Program the smart contract to automatically update rental prices based on specified legal conditions, such as inflation or local regulations, while ensuring fairness and compliance with predefined limits.
- **Legal Compliance and Flexibility:** Establish a system that complies with Turkish legal requirements and is adaptable to include new legal stipulations or clauses. This includes adding specialized provisions from the Turkish Code of Obligations to the smart contract.
- **Future Expansion:** Based on project timelines, consider expanding NLP functionality to incorporate more detailed legal provisions directly into the smart contract, making it capable of addressing complex legal scenarios automatically.

### 3: Project Scope and System Components:

#### 3.1: Smart Contract(s):

The core of this project will be a smart contract to enforce the terms of the lease agreement. This smart contract will include clauses for rent payments, security deposits, eviction conditions, and rental price adjustments, all compliant with Turkish law. The contract will interact with blockchain tokens to manage payments and ownership verification.

#### 3.2: Token Integration:

**3.2.1: ERC-721:** Each lease agreement will be represented as a unique ERC-721 token (NFT), serving as a tamper-proof record of the lease terms and conditions. By inheriting the ERC-721 standard, each lease contract will have a distinct, verifiable identifier on the blockchain, allowing straightforward tracking and validation.

### **3.3: Web Interface(s):**

The project will include a user-friendly web interface where users (landlords and tenants) can interact with the smart contract. Key features of this interface include:

1. UI Design (Frontend): The frontend will be designed for a seamless user experience, displaying lease information and contract details.
2. Metawallet Integration: Users will be able to connect their cryptocurrency wallets (such as MetaMask) to manage transactions.
3. External Components: Integration of libraries and APIs that support interactions with Ethereum or another blockchain network, as well as other UX/UX components for a smooth experience.

### **3.4: System Workflow**

#### **User Uploads Lease Agreement**

1. The process starts when the user uploads a lease agreement in PDF format. 2. The system performs necessary file validations (e.g., verifying the file type and checking for required fields).

#### **Data Parsing for Smart Contract Creation**

3. The uploaded PDF is parsed to extract relevant information (such as rental terms, deposit amount, eviction clauses).
4. The extracted data is used to create a smart contract that will govern the rental process.

#### **Deployment of Smart Contract**

5. The parsed lease agreement data is deployed to the blockchain as a smart contract.
6. The smart contract is configured to include control mechanisms for managing rent payments, enforcing eviction conditions, and other rental-related processes.

#### **User Initiates Rent Payment**

7. **User Selects Rent Payment Option:** The user logs into the system and selects the option to make a rent payment via the system's interface (e.g., a button or link labeled "Pay Rent").
8. **System Retrieves Payment Details:** The system fetches the current rental amount, payment due date, and the payment method (e.g., SepoliaETH). This information is retrieved from the smart contract or database, ensuring the correct and up-to-date payment terms are displayed.
9. **Wallet Connection Prompt:** If the user's cryptocurrency wallet (such as MetaMask) isn't already connected, the system prompts the user to connect it. The wallet will be used to sign and authorize the payment transaction.

### Token Transfer and Blockchain Update

10. The rent payment is processed using SepoliaETH
11. The transaction is recorded on the blockchain, and the smart contract is updated with the new payment status.

### Process Status and Automated Eviction

12. The user can view the real-time status of the payment and contract execution.
13. If the payment is not made on time or other conditions are not met, the smart contract triggers eviction procedures.

### 4) Challenges:

**Data Privacy:** Managing sensitive lease agreement data securely without compromising user privacy. (It will be handled in future works.)

**LLM Accuracy:** Ensuring high accuracy in converting natural language lease contracts into enforceable smart contracts, especially given potential legal nuances.

**Scalability:** Developing a scalable solution to handle potentially large numbers of lease contracts and payments on-chain without high gas fees.

**Legal Compliance:** Adhering to Turkish regulations for contract enforcement and ensuring that the smart contract operates within these legal boundaries.

### 5) Project Timeline

Task Duration(week)
---------------------

Initial Research and Requirement Analysis 1
Metawallet Integration and Smart Contract Design 2
Front End UI Design 3
Backend Development 4
NLP Feature Development (if time allows) 5,6
Documentation and Presentation Prep 7

## 6. Relative Work:

A recent study by Barbàra et al. [1] systematically investigates the possibility of using GPT-4 to autonomously generate production-ready Solidity contracts from legal documents. Their methodology centers on prompting a Large Language Model (LLM) with a lease agreement in order to produce self-contained Solidity code that encodes all the relevant clauses. Although various prompt-engineering strategies are introduced—chiefly the CO-STAR framework—the study notes several recurrent issues. These include logical flaws in payment handling, incomplete function logic, and missing mechanisms for accurate currency management (e.g., always using Ether instead of handling fiat-equivalent amounts). In many cases, automated vulnerability scanners failed to detect these shortcomings, underscoring that syntactic correctness does not guarantee robust real-world utility. The authors conclude that the current iteration of GPT-4 is not yet capable of generating fully dependable smart contracts without substantial manual review.

In contrast, our work takes a more modular approach by leveraging LLMs for parameter extraction rather than one-shot contract generation. Rather than relying on the LLM to produce the entire Solidity code, we employ it primarily to parse crucial data—such as rent amounts, party details, and property addresses—from uploaded lease agreements. We then inject these parameters into carefully designed smart-contract templates that have already been vetted for logical consistency and compliance with the Turkish Code of Obligations. This way, we preserve the efficiency gains offered by large-scale language modeling while significantly reducing the risk of inconsistent contract logic or incorrect function implementations. Moreover, our proposed system is equipped to handle additional mechanisms—like ERC-721 token management, rent adjustment strategies, and eviction workflows—within a structured, legally aligned framework. By isolating contract logic in well-tested code and harnessing LLMs only for selective tasks, our approach mitigates the issues Barbàra et al. identified and ensures a more reliable path to deploying enforceable rental contracts on the blockchain.

## 7. NLP Tools Implementation:

The NLP module leverages a Node.js/Express server, integrated with the OpenAI API, to parse lease agreements uploaded in PDF form. The front-end first converts the PDF to raw text and sends it to an endpoint (`/api/gpt`) where a GPT-based model is prompted to return key fields—tenant name, landlord name, property address, lease duration, monthly rent, dates, and security deposit—in JSON format. This structured response is then returned to the client, allowing the web interface to automatically

populate the smart contract form. By confining GPT's role to parameter extraction rather than generating complete contracts, the system ensures a focused, reliable integration of NLP insights.

This approach significantly lowers the risk of transcription errors and speeds up contract generation. It provides clear benefits for landlords and tenants who no longer have to manually parse lengthy legal texts. Furthermore, the added data validation on the client side (e.g., cleaning currency inputs) helps ensure higher data quality. Although GPT can occasionally produce incomplete or misformatted fields depending on PDF clarity, the combination of user review and lightweight server-side error handling addresses most issues, making the solution both efficient and practical for everyday lease processing.

## 8. Relevant Legislation and Integration into the Smart Contract

Under the Turkish Code of Obligations, several articles directly govern both the rent payment process and permissible actions if the tenant defaults. In particular, Article 315 stipulates that a landlord must provide the tenant with a written notice—and a specified time window of at least ten days (thirty days for residential or roofed-business premises)—prior to terminating the lease for nonpayment. Additionally, Article 344 caps any annual rent increase at the twelve-month average consumer price index, safeguarding the tenant from exorbitant hikes. Finally, Article 352 allows contract termination if the tenant, within a single rental year, fails to pay rent on time after receiving two valid written warnings from the landlord. Together, these sections form the legal foundation for automated notices, eviction procedures, and regulated rent adjustments within our blockchain-based system.

To operationalize these clauses in code, our **smart contract** checks payment timeliness, triggers warning notifications when deadlines are missed, and enforces rent adjustments up to the legally prescribed cap. Importantly, we have integrated **NLP-based parameter extraction** to parse essential contractual terms—such as rent amount, due dates, and deposit details—directly from uploaded PDF agreements. By automatically populating the contract's variables (e.g., `monthlyRent`, `startDate`, and `deposit`) using the GPT-powered pipeline, we reduce the risk of human error and ensure that the final on-chain agreement aligns with Articles 315, 344, and 352 of the Turkish Code of Obligations. This synergy of NLP tools and Solidity guarantees both legal compliance and technical robustness.

## 9. Smart Contract Implementation

### 9.1. Functions and Gas Consumptions

#### Modifiers:



```

// allow only landlord to access
modifier onlyLandLord() {
    require(landlords[msg.sender] == true , "Only landloard call this function." );
    _;
}

// allow only tenants to access
modifier onlyTenant(){
    require(tenants[msg.sender] == true , "Only tenant call this function." );
    _;
}

```

## Functions:

Function Name	Purpose	Gas Consumption (Wei)
addLandlord()	add landlord to landlords map	43,571
addTenant()	add tenant to tenants map	43,594
createRentalAgreement()	The landlord creates a rental agreement. Only the landlord calls this function.	301,728
notifyTenant()	The landlord notifies the tenant after the rental agreement was created. Only the landlord calls this function.	50,492
acceptRentalAgreement()	The tenant accepts a rental agreement. Only tenants call this function.	162,038
refuseRentalAgreement()	The tenant refuses the rental agreement. Only tenants call this function.	70,136
payRent()	The tenant pays rent amount to the landlord. Tenants can pay rent in the payment range. Only tenants call this function. otherwise they can not .	56,816

sendWarning()	The landlord sends a warning to the tenant if the tenant does not pay rent in the payment range. Only the landlords call this function.	92,190
revokeUsageRights()	The landlord takes back NFT to the landlord address if tenants get two warnings. Only landlords call this function.	86,472

## 10 . Simple Workflow

**Landlord address : 0x5EBf29f15143966dF7Ff7b282E18270E864BEA5A**

**Tenant address : 0xDda20E112e4989153f93D0eD6625fCe17b21d71F**

As you can see below, after the landlord calls the createRentalAgreement(), NFT which names RentalNFT and symbol (RENT) is created with 2 NFT id. Also, this NFT is transferred to the landlord in the beginning.

[ This is a Sepolia Testnet transaction only ]

Transaction Hash:

0x20f84c893e0e965c6573f8fda6414397f71237d38ee328b7332b601982252926

Status:

Success

Block:

7433240 1072 Block Confirmations

Timestamp:

3 hrs ago (Jan-06-2025 12:56:48 PM UTC)

Transaction Action:

Call 0x6f8980d0 Method by 0x5EBf29f1...E864BEA5A on 0x54Dd2f1e...3953CD105

From:

0x5EBf29f15143966dF7Ff7b282E18270E864BEA5A

Interacted With (To):

0x54Dd2f1edc8D2D733eB28173c7B7c4c3953CD105

ERC-721 Tokens Transferred:

ERC-721 Token ID [2] RentalNFT(RENTNF...)

From 0x00000000...000000000 To 0x5EBf29f1...E864BEA5A

After the creation of the rental agreement, agreement details will be notified to the tenant and then the tenant calls the acceptAgreement(). After calling this function, the NFT whose symbol RENT is transferred from the landlord to the tenant.

[ This is a Sepolia **Testnet** transaction only ]

Transaction Hash: 0x0b57414b98e668020a168e30798982c667563e3007d1e4c0ad65627258a81c69

Status: Success


Block: 7433245 1119 Block Confirmations

Timestamp: 3 hrs ago (Jan-06-2025 12:57:48 PM UTC)

Transaction Action: Call 0x334491eb Method by 0xDda20E11...17b21d71F on 0x54Dd2f1e...3953CD105

From: 0xDda20E112e4989153f93D0eD6625fCe17b21d71F

Interacted With (To): 0x54Dd2f1edc8D2D733eB28173c7B7c4c3953CD105

ERC-721 Tokens Transferred:  ERC-721 Token ID [2] RentalNFT(RENTNF...)  
From 0x5EBf29f1...E864BEA5A To 0xDda20E11...17b21d71F

In the first month, the tenant pays its rent. As you can see 10000 wei is transferred from the tenant to the landlord.

[ This is a Sepolia **Testnet** transaction only ]

Transaction Hash: 0x218c2a70c11dfb313a5fafa365bca5c1e361308244698e1d611a2fb216c6e286

Status: Success

Block: 7433251 1511 Block Confirmations

Timestamp: 5 hrs ago (Jan-06-2025 12:59:12 PM UTC)

Transaction Action: Call Pay Rent Function by 0xDda20E11...17b21d71F on 0x54Dd2f1e...3953CD105

From: 0xDda20E112e4989153f93D0eD6625fCe17b21d71F

To: 0x54Dd2f1edc8D2D733eB28173c7B7c4c3953CD105

Transfer 0.000000000000001 ETH From 0x54Dd2f1e...3953CD105 To 0x5EBf29f1...E864BEA5A

Then the tenant does not pay its rent in the second and third month. So that the landlord sends warnings by using `sendWarning()` method in the second and third month.

	<a href="#">0x574d611bf99...</a>	<span>0x93c7aaf4</span>	<span>7433278</span>	7 mins ago	<a href="#">0x5EBf29f1...E864BEA5A</a>	<span>IN</span>	<span>0x54Dd2f1e...3953CD105</span>	0 ETH	0.00087555
	<a href="#">0x8bdc66c5f48...</a>	<span>0x93c7aaf4</span>	<span>7433265</span>	9 mins ago	<a href="#">0x5EBf29f1...E864BEA5A</a>	<span>IN</span>	<span>0x54Dd2f1e...3953CD105</span>	0 ETH	0.00173226
	<a href="#">0x218c2a70c1...</a>	<span>Pay Rent</span>	<span>7433251</span>	12 mins ago	<a href="#">0xDda20E11...17b21d71F</a>	<span>IN</span>	<span>0x54Dd2f1e...3953CD105</span>	0 ETH	0.00104791

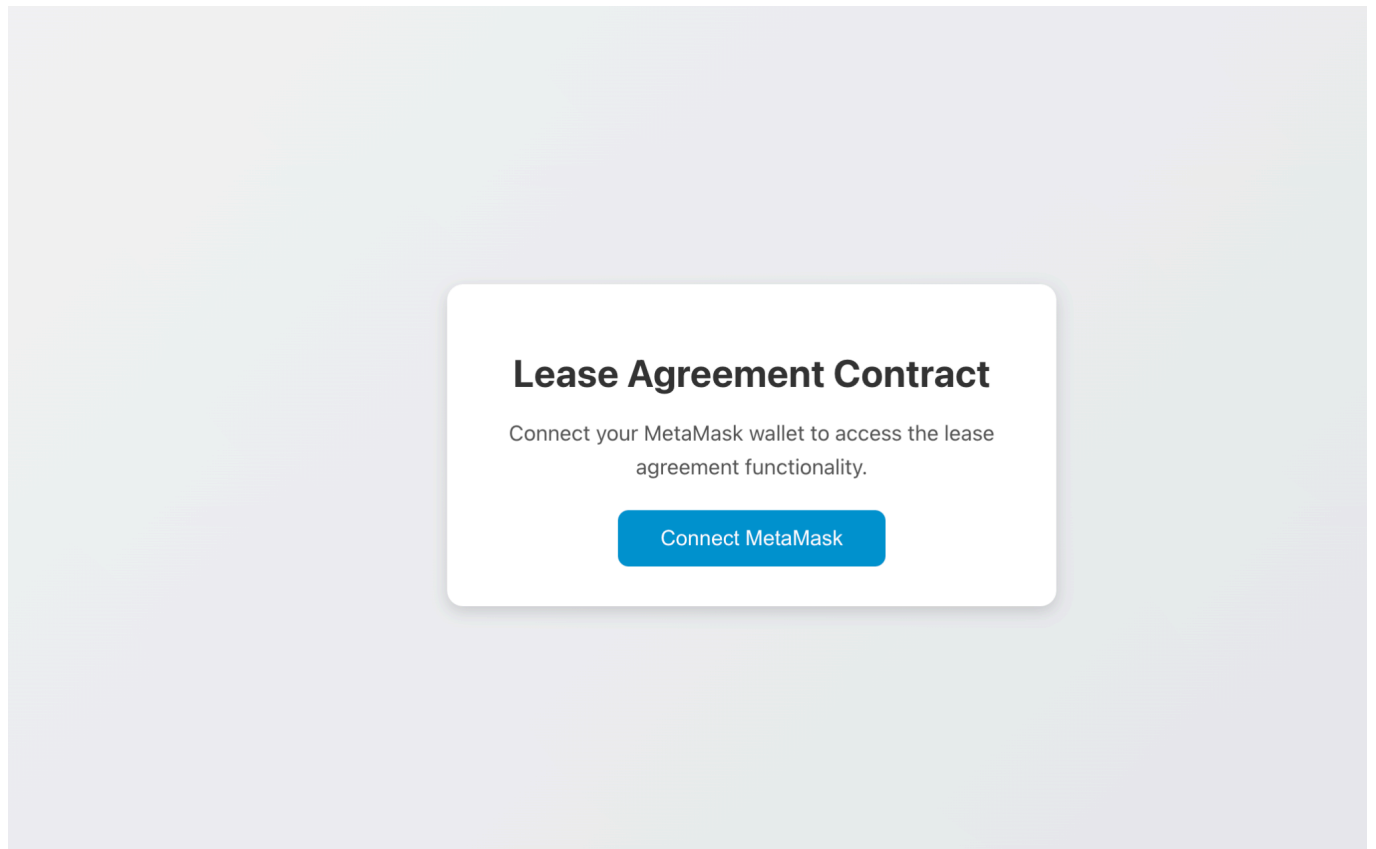
After two warnings with `0x93c7aaf4` hash code , the landlord can send `revokeUsageRights()` methods to take back NFT. And the landlord calls the `revokeUsageRights()` and NFT whose token id is 2 is transferred from tenant to landlord.

[ This is a Sepolia Testnet transaction only ]

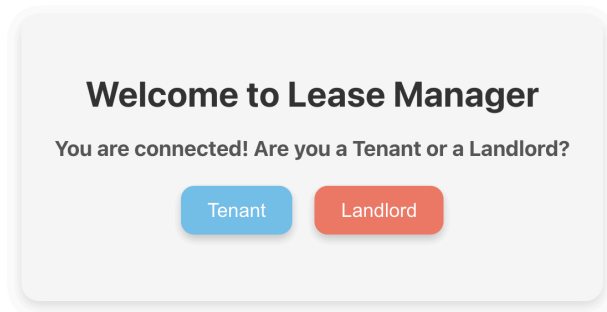
Transaction Hash:	0x79aad213ce90a64335745192a5fde28e6d94364949a724b3d82c7ee8b6d88cb
Status:	Success
Block:	7433294 1494 Block Confirmations
Timestamp:	5 hrs ago (Jan-06-2025 01:08:00 PM UTC)
Transaction Action:	Call 0xf04c9421 Method by 0x5EBf29f1...E864BEA5A on 0x54Dd2f1e...3953CD105
From:	0x5EBf29f15143966dF7Ff7b282E18270E864BEA5A
Interacted With (To):	0x54Dd2f1edc8D2D733eB28173c7B7c4c3953CD105
ERC-721 Tokens Transferred:	ERC-721 Token ID [2] RentalNFT(RENTNF...) From 0xDda20E11...17b21d71F To 0x5EBf29f1...E864BEA5A

## 11.UI IMPLEMENTATION

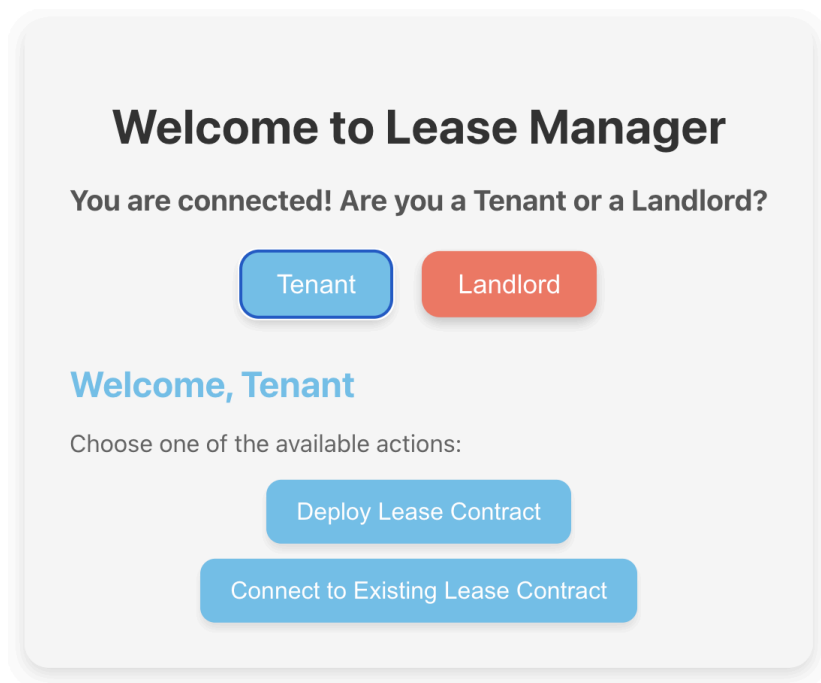
**Metamask Connection:** A simple interface that enable users to connect their metamask wallets.



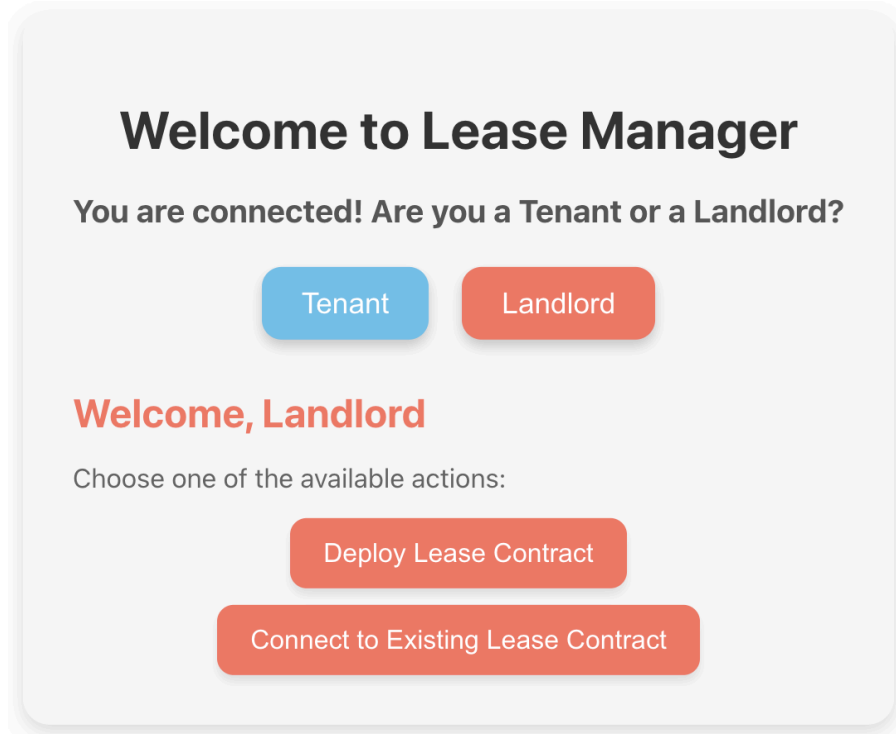
**Home Page:** A homepage to ask users what their role is.



If user enters the system as Tenant;



If user enters the system as Landlord;



The image shows a user interface for a system called "Lease Manager". At the top, it says "Welcome to Lease Manager". Below that, it asks "You are connected! Are you a Tenant or a Landlord?". There are two buttons: "Tenant" (blue) and "Landlord" (red). The "Landlord" button is selected. Below the buttons, it says "Welcome, Landlord" in red. Then it says "Choose one of the available actions:". There are two red buttons: "Deploy Lease Contract" and "Connect to Existing Lease Contract".

## Welcome to Lease Manager

You are connected! Are you a Tenant or a Landlord?

Tenant Landlord

**Welcome, Landlord**

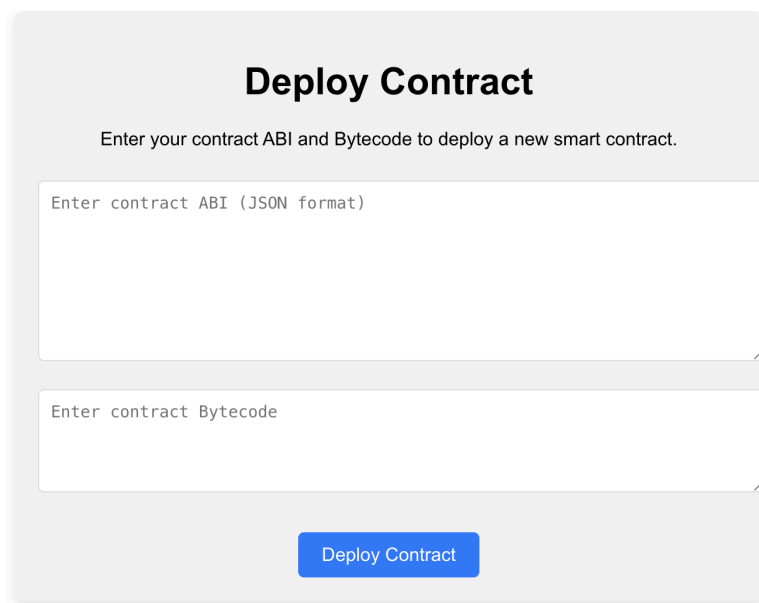
Choose one of the available actions:

Deploy Lease Contract

Connect to Existing Lease Contract

Both the tenant and landlord can deploy the lease contract or connect to the existing contract that is already created.

**Deploy Contract Page:** This page is available for both landlords and tenants. After the contract is compiled, necessary inputs are entered to deploy the contract.



The image shows a user interface for a system called "Deploy Contract". At the top, it says "Deploy Contract". Below that, it says "Enter your contract ABI and Bytecode to deploy a new smart contract.". There are two text input fields: "Enter contract ABI (JSON format)" and "Enter contract Bytecode". At the bottom, there is a blue button labeled "Deploy Contract".

## Deploy Contract

Enter your contract ABI and Bytecode to deploy a new smart contract.

Enter contract ABI (JSON format)

Enter contract Bytecode

Deploy Contract

**Connect To Contract Page:** This page is used for connecting an existing contract. This feature is also accessible by both the tenant and landlord.

## Connect to Contract

Deployed Contract Address:

Enter deployed contract address

Contract ABI (JSON):

Enter Contract ABI in JSON format

Connect



## Landlord Interactions

If the landlord connects to the contract, they will only see the actions that they are allowed to perform. Landlords can create agreements, send warnings in case the rent is not paid and start evacuation if the warning reaches to 2.

### Successfully connected to the contract!

Contract Owner:

0xE3ff8ee6927a9aC60AcD424C86A5F0223d59a

Create Agreement

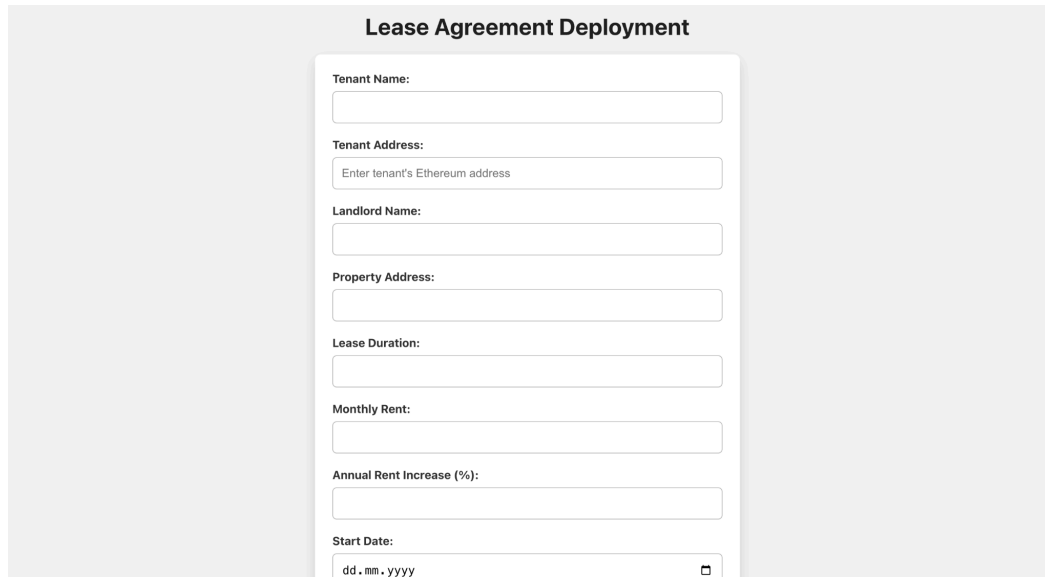
Enter Agreement ID

Send Warning

Enter Agreement ID

Start Evacuation

If the landlord wants to create an agreement, an agreement page will be displayed. The inputs will be filled from a lease agreement, which is a PDF file. We have an NLP server that interacts with GPT and parses the file according to the contract form.



The image shows a web form titled "Lease Agreement Deployment". It contains several input fields for tenant and landlord information, property details, and lease terms. The fields are arranged vertically and include placeholder text for some of them.

**Lease Agreement Deployment**

Tenant Name:

Tenant Address:

Landlord Name:

Property Address:

Lease Duration:

Monthly Rent:

Annual Rent Increase (%):

Start Date:

Here a pdf file is selected.



The image shows a file upload interface. It features a large empty box at the top, followed by the heading "Upload Lease Agreement (PDF):". Below this is a file selection area with a "Choose File" button and the text "No file chosen". At the bottom, the text "Upload Additional Files:" is partially visible.

**Upload Lease Agreement (PDF):**

No file chosen

**Upload Additional Files:**

This is the pdf file that we generated as a sample.

#### KİRA SÖZLEŞMESİ

<b>DAİRE NO:</b> 5 <b>MAHALLE:</b> Çamlıca Mahallesi <b>SOKAK:</b> Çamlık Sokak <b>NUMARA:</b> 10 <b>KİRALANAN ŞEYİN CİNSİ:</b> Daire
<b>KİRAYA VERENİN ADI - SOYADI:</b> Muhammed Enes
<b>İKÂMETGAH VE T.C. KİMLİK NO.:</b> Çamlıca Mah., Çamlık Sok. No:10, T.C. Kimlik No: 12345678901
<b>KİRALAYANIN ADI - SOYADI:</b> Tolga Fehmioğlu
<b>İKÂMETGAH VE T.C. KİMLİK NO.:</b> Kavaklı Mah., Güzel Sok. No:20, T.C. Kimlik No: 10987654321
<b>BİR YILLIK KİRA KARŞILIĞI:</b> 60.000 TL <b>BİR AYLIK KİRA KARŞILIĞI:</b> 5.000 TL <b>KİRANIN NE ŞEKİLDE ÖDENECEĞİ:</b> Banka havalesi yoluyla <b>KİRA SÜRESİ:</b> 1 yıl <b>KİRANIN BAŞLANGIÇ TARİHİ:</b> 01.02.2025 <b>KİRALANAN ŞEYİN ŞİMDİKİ DURUMU:</b> Temiz ve bakımlı <b>KİRALANAN ŞEYİN NE İÇİN KULLANILACAĞI:</b> Dernek Merkezi <b>KİRALANAN ŞEY İLE BERABER TESLİM OLUNAN DEMİRBAŞ EŞYANIN BEYANI:</b> 2 adet masa, 6 adet sandalye, 1 adet dolap
<b>DAİRE NO:</b> 5 <b>MAHALLE:</b> Çamlıca Mahallesi <b>SOKAK:</b> Çamlık Sokak <b>NUMARA:</b> 10 <b>KİRALANAN ŞEYİN CİNSİ:</b> Daire
<b>KİRAYA VERENİN ADI - SOYADI:</b> Muhammed Enes
<b>İKÂMETGAH VE T.C. KİMLİK NO.:</b> Çamlıca Mah., Çamlık Sok. No:10, T.C. Kimlik No: 12345678901
<b>KİRALAYANIN ADI - SOYADI:</b> Tolga Fehmioğlu
<b>İKÂMETGAH VE T.C. KİMLİK NO.:</b> Kavaklı Mah., Güzel Sok. No:20, T.C. Kimlik No: 10987654321
<b>BİR YILLIK KİRA KARŞILIĞI:</b> 60.000 TL <b>BİR AYLIK KİRA KARŞILIĞI:</b> 5.000 TL <b>KİRANIN NE ŞEKİLDE ÖDENECEĞİ:</b> Banka havalesi yoluyla <b>KİRA SÜRESİ:</b> 1 yıl <b>KİRANIN BAŞLANGIÇ TARİHİ:</b> 01.02.2025 <b>KİRALANAN ŞEYİN ŞİMDİKİ DURUMU:</b> Temiz ve bakımlı <b>KİRALANAN ŞEYİN NE İÇİN KULLANILACAĞI:</b> Dernek Merkezi <b>KİRALANAN ŞEY İLE BERABER TESLİM OLUNAN DEMİRBAŞ EŞYANIN BEYANI:</b> 2 adet masa, 6 adet sandalye, 1 adet dolap
<b>DAİRE NO:</b> 5 <b>MAHALLE:</b> Çamlıca Mahallesi <b>SOKAK:</b> Çamlık Sokak <b>NUMARA:</b> 10 <b>KİRALANAN ŞEYİN CİNSİ:</b> Daire

After the pdf is selected;

**Tenant Name:**

Tolga Fehmioğlu

**Tenant Address:**

Enter tenant's Ethereum address

**Landlord Name:**

Muhammed Enes

**Property Address:**

Çamlıca Mah., Çamlık Sok. No:10

**Lease Duration:**

1 yıl

**Monthly Rent:**

5.000 TL

**Annual Rent Increase (%):**

**Start Date:**

01.02.2025



**End Date:**

01.02.2026



**Security Deposit:**

**Additional Terms:**

**Contact Info:**

**Upload Lease Agreement (PDF):**

Kira-Sozlesmesi-Ornegi.pdf

### Tenant Interactions:

After the tenant connects to the contract, they must add themselves to the contract for it to be valid. Once this is done, they can perform actions specific to them, such as accepting or refusing the agreement made with the landlord. They can also pay their rent through our interface.

### Successfully connected to the contract!

Contract Owner:

0xE8E3ff8ee6927a9aC60AcD424C86A5F0223d59a

Add Tenant

Enter Agreement ID

Accept Agreement

Enter Agreement ID

Refuse Agreement

Enter rent amount in Etl

Pay Rent

### References:

[1] Barbàra, F., Napoli, E. A., Gatteschi, V., & Schifanella, C. (2024). *Automatic Smart Contract Generation Through LLMs: When The Stochastic Parrot Fails*. In 6th Distributed Ledger Technology Workshop, Turin, Italy.