---

## Functional Module Description:

### 1. Contract Activation Function:

\*\*a. Applicable Personnel:\*\*

Landlord

\*\*b. Preconditions:\*\*

The landlord has initialized the contract code (provided their blockchain account address and rental agreement content to the contract code) and deployed this contract code using their account.

\*\*c. Function Description:\*\*

After linking their pre-set account address, the landlord fills in the relevant information and signs the agreement to activate all functions of this contract.

\*\*d. Interface Logic Description:\*\*

After linking the MetaMask account, the system uses the contract method `isActivate()` to check if the contract is activated. If not activated, it redirects to the activation page. The landlord needs to fill in fields such as name, email, phone, nationality, and passport. After completing and submitting, the `addTenant()` method is called to register their information into the contract. Once the landlord's information is registered, the contract code logic will default to the contract being activated.

\*\*e. Security Analysis:\*\*

Once the landlord adds their account address and agreement content and deploys it, the contract code cannot be modified. Anyone can view the contract content, addressing the transparency issue of traditional contracts. Although the landlord needs to provide some personal information, this information will not be displayed in the contract. Only the landlord can view their information, and once confirmed, it cannot be modified. This not only resolves potential disputes but also addresses data leakage and unauthorized access threats.

\*\*f. Gas and Contract Method Logic:\*\*

- `isActivate()`: Self-evaluated gas, accessible only by the landlord.

- `addTenant()`: Self-evaluated gas, accessible by any account.

### 2. View Room Status and Perform Daily Tasks:

\*\*a. Applicable Personnel:\*\*

Landlord

\*\*b. Preconditions:\*\*

The contract has been activated.

\*\*c. Function Description:\*\*

The landlord can view the total number of rooms, the number of vacant rooms, and the number of booked rooms on their homepage. By clicking the button, they can perform daily functions to update the contract status (since the contract cannot automatically update room status, the landlord needs to perform this function themselves. The agreement signed during contract activation will specify the consequences if the landlord fails to perform this function).

\*\*d. Interface Logic Description:\*\*

Upon entering the homepage, the `getAllRooms` method is called to get all room information. The front end determines the availability of each room based on the `isAvailable` attribute and renders this information on the page. Additionally, the landlord must call the `dailyAction` method daily and pass the current time to update the room and rental information status. The `dailyAction` method updates the corresponding rental and room status based on the comparison between the passed time and the `endTime` attribute under the rental information.

\*\*e. Security Analysis:\*\*

Due to the immutability of smart contracts, the rental information still needs to be updated daily by the landlord by passing the current time, which does not affect transparency and security. The landlord updates only the current time to update the status, which does not affect the interests of both parties (however, if the landlord fails to update the daily tasks, it may affect the landlord's interests, which is specified during contract activation). Thus, the entire process is transparent and secure. Moreover, this daily task can only be accessed by the landlord and can only be called once a day, preventing misuse and spam transactions, and avoiding the consumption of network resources.

\*\*f. Gas and Contract Method Logic:\*\*

- `getAllRooms()`: No gas required, accessible by legitimate accounts.

- `dailyAction()`: No gas required, accessible only by the landlord, can be called once a day.

### 3. Add and Modify Room Status:

\*\*a. Applicable Personnel:\*\*

Landlord

\*\*b. Preconditions:\*\*

The contract has been activated.

\*\*c. Function Description:\*\*

The landlord can modify the room status to available or unavailable based on the real situation and can add a new room as needed. However, to ensure security, rooms cannot be deleted; the landlord can only mark rooms as unavailable. Therefore, the landlord needs to add each room cautiously.

\*\*d. Interface Logic Description:\*\*

First, use the `getAllRooms()` method to get the room list.

To update the room status, call the `updateRoomState()` method with parameters `roomId` and `state`. If `state` is true, the room's `isAvailable` is set to true; if false, `isAvailable` is set to false. When adding a room, call the `addRoom` method with parameters `rent`, `addressInfo`, `location`, and `description` to add a new room.

\*\*e. Security Analysis:\*\*

To ensure the immutability of rooms, once a room is created, it will be permanently stored in the contract. The landlord can only modify the room status. The parameters passed to the contract will be validated to ensure they are valid before executing the corresponding logic, ensuring the contract methods cannot be arbitrarily modified. Moreover, only the landlord can access the contract methods, effectively preventing unauthorized access risks while ensuring transparency.

\*\*f. Gas and Contract Method Logic:\*\*

- `getAllRooms()`: No gas required, accessible by legitimate accounts.

- `updateRoomState()`: Automatically evaluated gas, accessible only by the landlord.

- `addRoom()`: Automatically evaluated gas, accessible only by the landlord.

### 4. View and Modify Tenant Request Status:

\*\*a. Applicable Personnel:\*\*

Landlord

\*\*b. Preconditions:\*\*

The contract has been activated, and a customer has rented a room and submitted a related request.

\*\*c. Function Description:\*\*

The landlord can view the history list of tenant requests, each including `roomInfo`, `requestType`, `description`, `createTime`, and `state` attributes. The landlord can resolve the request offline based on the information. After resolving the request, the landlord can mark it as resolved. For resolved requests, the landlord can view the resolution time.

\*\*d. Interface Logic Description:\*\*

First, use the `getAllRequest()` method to get all request lists.

After resolving offline, use the `modifyRequest` method with parameters `requestId` and `endTime`. This method will set the `isResolved` attribute to true and update the `endTime`, which is the request resolution time.

\*\*e. Security Analysis:\*\*

When a customer encounters an issue related to the room, they can submit a problem description. These descriptions do not include personal privacy information and are stored in the contract, ensuring data immutability. This not only ensures privacy but also resolves potential future disputes.

\*\*f. Gas and Contract Method Logic:\*\*

- `getAllRequest()`: No gas required, accessible only by the landlord.

- `modifyRequest()`: Automatically evaluated gas, accessible only by the landlord.

### 5. View Rental Information by Address:

\*\*a. Applicable Personnel:\*\*

Landlord

\*\*b. Preconditions:\*\*

The contract has been activated, and a room has been rented by a customer.

\*\*c. Function Description:\*\*

The landlord can view the rental information of an address provided by the tenant and process the corresponding offline procedures based on this information.

\*\*d. Interface Logic Description:\*\*

Use the `getTenantRentInfoByAd()` method with the parameter `tenantAddress` to get the rental information of this address. If not available, it will display empty.

\*\*e. Security Analysis:\*\*

After a tenant rents a house through a smart contract, they will check in offline using their account address. The landlord only verifies whether the user has signed the rental information through the smart contract using this address, without asking for other details. This process not only protects the personal privacy of both parties but also increases efficiency.

\*\*f. Gas and Contract Method Logic:\*\*

- `getTenantRentInfoByAd()`: Automatically evaluated gas, accessible only by the landlord.

### 6. Tenant Registration Function:

\*\*a. Applicable Personnel:\*\*

Tenant

\*\*b. Preconditions:\*\*

The contract has been activated.

\*\*c. Function Description:\*\*

The tenant needs to fill in the relevant information to register their information in the smart contract.

\*\*d. Interface Logic Description:\*\*

Use the `addTenant()` method with parameters `tenantAddress`, `createTime`, `name`, `email`, `phone`, `nationality`, and `passport` to register their basic information in the contract.

\*\*e. Security Analysis:\*\*

This information is immutable and only viewable by the individual, ensuring privacy security.

\*\*f. Gas and Contract Method Logic:\*\*

- `addTenant()`: Automatically evaluated gas, accessible by the blockchain, one account only once.

### 7. Tenant Rental Function:

\*\*a. Applicable Personnel:\*\*

Tenant

\*\*b. Preconditions:\*\*

The contract has been activated, the tenant has registered their information in the smart contract, and there are currently available rooms.

\*\*c. Function Description:\*\*

The tenant can view available rooms, choose the room they want to rent, and click the booking button. The front end will automatically use the current date as the rental start date. The tenant can choose the end date, and the front end will adjust the total fee based on the chosen end date. After clicking next, the tenant can view the relevant contract terms. After carefully reading the contract terms, if the tenant agrees to all terms, they can click confirm to sign the rental contract.

\*\*d. Interface Logic Description:\*\*

First, call the `getAllRooms` method to get all room information. After the tenant chooses their preferred room, the front end will generate the current time as the contract start time using JavaScript. Then, it will guide the tenant to choose the rental end time and adjust the total fee based on the chosen end time. It's important to note that the tenant's chosen time interval is 28 days. After clicking next, the `getAgreement` method is called to get the relevant contract terms. If the tenant has no issues with the contract terms, they click submit, which calls the `rentRoom` method with parameters `tenantId`, `\_roomId`, `createTime`, `endTime`, `nextTime`, `nextPay`, `payTotal`, and `description` to rent the room (`nextTime` is the next payment time for the tenant, initially equal to `createTime`, meaning the tenant needs to complete the payment immediately after completing this function).

\*\*e. Security Analysis:\*\*

This function is a core function. After the tenant chooses their desired room, the front end automatically sets the current time as the contract start time to ensure the contract's effectiveness. The end time interval is set to 28 days to ensure uniformity. All visible rooms are real and valid, and cannot be modified by anyone, ensuring the authenticity of the information and solving the trust issues of traditional contracts. The contract terms viewed by the tenant are also immutable. All these aspects ensure the transparency and efficiency of the rental process, providing higher security for the tenant.

\*\*f. Gas and Contract Method Logic:\*\*

- `getAllRooms`: Automatically evaluated gas, accessible by legitimate accounts.

- `getAgreement`: Automatically evaluated gas, accessible by legitimate accounts.

- `rentRoom`: Automatically evaluated gas, accessible by tenants.

### 8. Tenant Payment Function:

\*\*a. Applicable Personnel:\*\*

Tenant

\*\*b. Preconditions:\*\*

The contract has been activated, and the tenant has rented a room but has not fully paid the rent.

\*\*c. Function Description:\*\*

The tenant needs to click the payment button to pay one month's rent before the `nextTime` attribute in the rental information. After the tenant confirms the payment, the system will automatically update the next payment time.

\*\*d. Interface Logic Description:\*\*

Call the `hasRentInfo` method with parameters `address tenantAddress` and `uint256 curTime`. The contract will automatically check if the tenant has signed the relevant rental contract and if the contract has not expired. This method returns the tenant's rental information and extracts the `nextTime` attribute to compare with the current time. If the current time is less than `nextTime`, it displays the `nextTime` for the tenant to view (this `nextTime` is the latest payment time for the tenant). The tenant clicks the payment button, which calls the `payRent` method with parameters `uint256 rentId` and `uint256 nextTime` to automatically pay one month's rent and update the `nextTime` value, i.e., the next payment time.

\*\*e. Security Analysis:\*\*

The payment process is separate from signing the rental contract. If the tenant signs the rental contract but does not pay according to the contract, the contract will immediately become invalid. These processes are automated to ensure both parties' rights and improve the rental market's efficiency and transparency. Moreover, the tenant does not need to perform any extra calculations as the payment time and amount are automatically calculated by the contract, increasing efficiency. The entire process is anonymous, with payment made via Ether, ensuring both parties' privacy.

\*\*f. Gas and Contract Method Logic:\*\*

- `hasRentInfo`: Automatically evaluated gas, accessible by tenants.

- `payRent`: Automatically evaluated gas, accessible by tenants.

### 9. Tenant Request Function:

\*\*a. Applicable Personnel:\*\*

Tenant

\*\*b. Preconditions:\*\*

The contract has been activated, and the tenant has rented a room that has not yet expired.

\*\*c. Function Description:\*\*

If there are any issues during the rental period, the tenant can submit a request to describe their problems and wait for the landlord to resolve them.

\*\*d. Interface Logic Description:\*\*

Click the request tab on the front-end page, fill in the `Category` and `Description`, and click submit to call the `createRequest` method with parameters `roomInfo`, `TenantId`, `requestType`, `description`, and `createTime` to create a request in the smart contract.

\*\*e. Security Analysis:\*\*

Only room information and related descriptions are provided, without any personal information, ensuring privacy. The request is recorded in the contract and cannot be changed. If the request is not addressed, or if the landlord does not fulfill the contract terms, the tenant can use this request information to protect their rights. This security mechanism avoids future disputes.

\*\*f. Gas and Contract Method Logic:\*\*

- `createRequest`: Automatically evaluated gas, accessible by tenants.