DECENTRALIZED-FREELANCE-MARKETPLACE

A project by

TEAM: SmartContractors

ROLL NO	NAME
230001003	Abhinav Bitragunta
230001006	Aman Gupta
230001068	Rayavarapu Sreechand
230001072	Srinidhi Sai Boorgu
230004005	Ansh Jain
230005011	Bhumika Aggarwal

1. Introduction

The Freelance Marketplace is a decentralized application (dApp) built using Solidity for the Ethereum blockchain. This project enables freelancers and clients to interact in a decentralized environment, where services can be offered, payments can be escrowed, and reputation can be tracked through ratings. By utilizing smart contracts, we remove the need for a centralized intermediary while maintaining transparency, immutability, and security.

The platform facilitates:

- Freelancers to offer services and view their status.
- Clients to hire freelancers by paying into an escrow.
- Clients to release payments upon satisfaction.
- Clients to claim refunds after the deadline expires.
- Clients to rate freelancers post release of funds.
- Rating aggregation and average rating calculation.

2. Contract Structure and Key Components

The contract is written in Solidity (version ^0.8.0) and uses the OpenZeppelin ReentrancyGuard for added security. The major components include:

2.1 Structs

2.1.1 Service

This struct contains all necessary data about a freelance service:

- id: Unique identifier.

- freelancer: Address of the service provider.
- client: Address of the hiring client.
- title: A title for the service provided.
- description: A description of the provided service.
- price: Service cost in wei.
- isActive: Boolean indicating whether the service is still valid.
- isPaid: Boolean indicating payment status.
- rating: Rating from the client (1 to 5).
- deadline: UNIX timestamp denoting the deadline for service completion.

2.1.2 FreelancerRating

Stores cumulative rating data for freelancers:

- total: Sum of all ratings received by the freelancer.
- count: Number of ratings received by the freelancer.

2.2 State Variables

- serviceCounter: Global counter to assign unique service IDs.
- services: Mapping of serviceId to Service.
- escrowedFunds: Mapping of serviceId to escrowed Ether amount.
- freelancerRatings: Mapping of freelancer address to their rating details.

3. Core Functionalities

3.1 Service Offering

Function: offerService

- A freelancer can list their service with a title, price, description, and deadline.
- Input validation ensures that the title and description are non-empty, price is greater than 0, and the deadline is positive.
- The deadline is stored as a UNIX timestamp (block timestamp + _deadline days).
- Emits ServiceOffered event.

3.2 Hiring Freelancers

Function: hireFreelancer

- A client can hire a freelancer by providing the exact Ether value as payment.

- The function checks if the service exists, is active and is not already hired. It also disallows the client from hiring themselves and matches the service price with the payment.
- The Ether is stored in escrow (using a dedicated mapping).
- Emits FreelancerHired event.

3.3 Releasing Payments

Function: releasePayment

- Only the client can release the payment after work is complete and is satisfied.
- The function checks that the caller is the client, that there are enough funds escrowed and that the service hasn't been paid for already.
- Ether is transferred using .call, and security is enhanced using nonReentrant.
- Emits PaymentReleased event.

3.4 Refunds (Bonus feature)

Function: refundClient

- Allows clients to reclaim funds if the service deadline passes.
- The function checks if the caller is the client, payment has not been released already, escrowed funds exist and if the deadline has passed.
- On success, Ether is refunded, and service marked inactive.
- Emits ClientRefunded event.

3.5 Rating System (Bonus feature)

Function: rateService

- Clients can rate services only once and only after releasing payment.
- Ratings between 1 and 5 are allowed.
- Stored in the FreelancerRating struct and used for computing average.
- Emits ServiceRated event.

3.6 Getter Functions

- getAverageRating: Computes average rating for any freelancer.
- getRatingCount: Returns number of ratings received.
- getServiceCount: Returns number of services listed.

4. Gas Optimization Strategies

The Decentralised Freelance Marketplace includes several key optimizations.

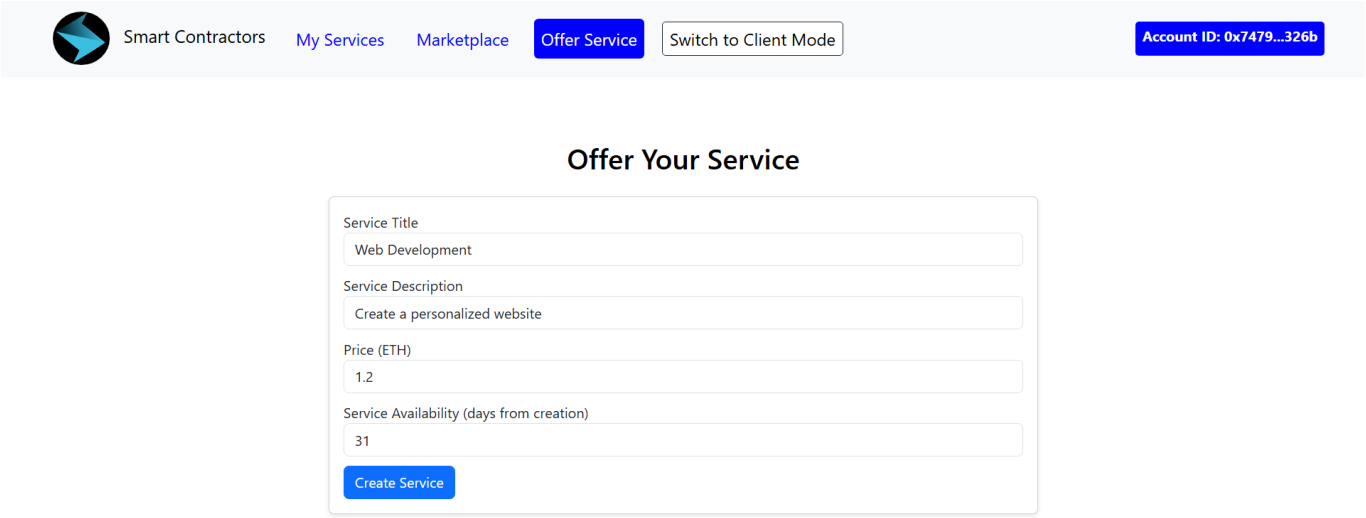
1. **Efficient Struct Design:** The elements of the Service struct are ordered to take advantage of struct packing by placing smaller datatypes next to each other, to minimize padding and maximize memory efficiency.
2. **Minimal Storage Writes:** Escrow funds, ratings are updated only once per service, and data duplication is avoided where possible.
3. **View modifier for getter functions:** Getter functions like getAverageRating, getRatingCount, and getServiceCount that are marked as "view" incur no gas cost when called externally (off-chain such as from front-end), but consume gas when called internally within other functions.
4. **No Loops / Iterations Over Storage:** Avoids `for` or `while` loops, especially over mappings or arrays, which scale gas costs with size.

5. Security Measures

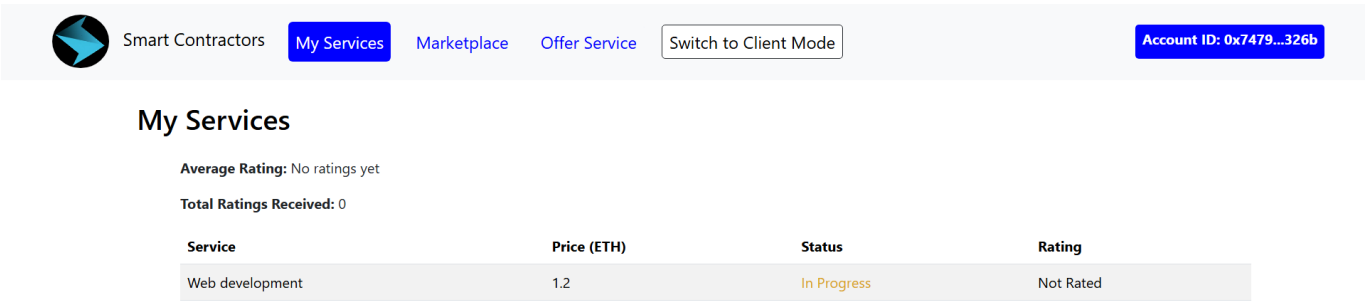
The application eliminates several security vulnerabilities.

1. **Re-entrancy attacks:** Uses the ReentrancyGuard contract from OpenZeppelin in all functions dealing with Ether transfers and prevents re-entrancy attacks which enable malicious draining of funds.
2. **Checks-Effects-Interactions Pattern:** All Ether-handling functions (releasePayment, refundClient) follow the best practice of first checking conditions, then updating state, and finally interacting with external contracts. For example, the .call to transfer Ether is placed after all state changes, reducing the chance of re-entrancy vulnerabilities
3. **Validations:** Inputs to all functions are strictly validated (with require statements) to prevent unintended behaviour.
4. **Safe Ether Transfer:** Ether is transferred using .call instead of .transfer or .send to accommodate changing gas costs and avoid potential reentrancy.
5. **Deadline Enforcement:** Refunds are only allowed after the deadline of a service to prevent malicious refunding, such as users requesting a refund while the service is still being delivered or before the freelancer providing the service has had a fair chance to complete the task.
6. **Address based identification:** Personal information of clients and freelancers (E-Mail ID, Name) not stored on chain. Role based accesses in functions are handled using the publicly available addresses.

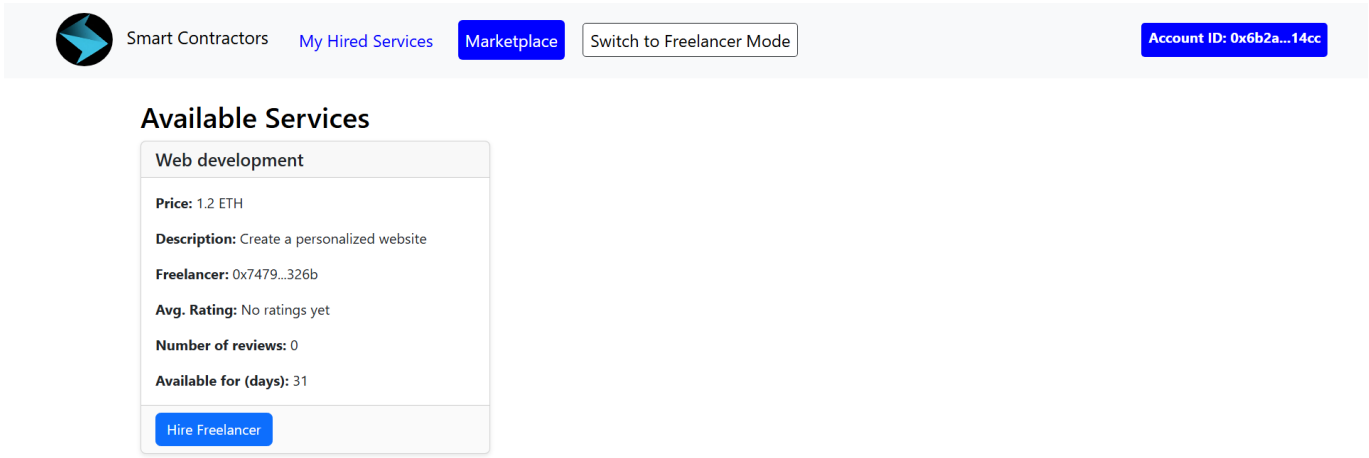
6. Example Walkthrough



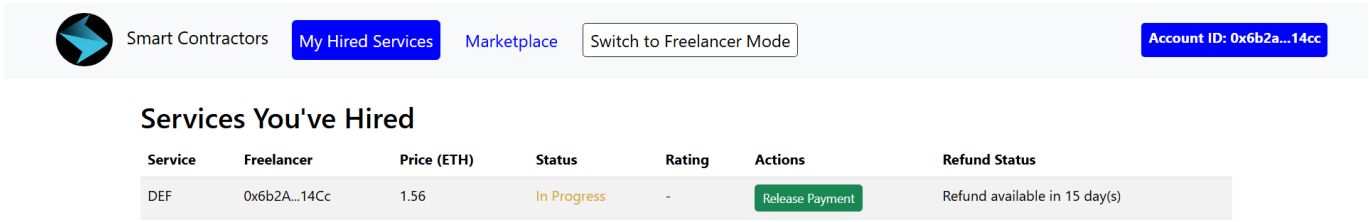
This shows the service creation form within the Smart Contractors platform. Here, a freelancer enters details such as the Service Title, Description, Price in ETH, and Service Availability. Once the fields are filled, clicking the "Create Service" button adds the offering to the marketplace.



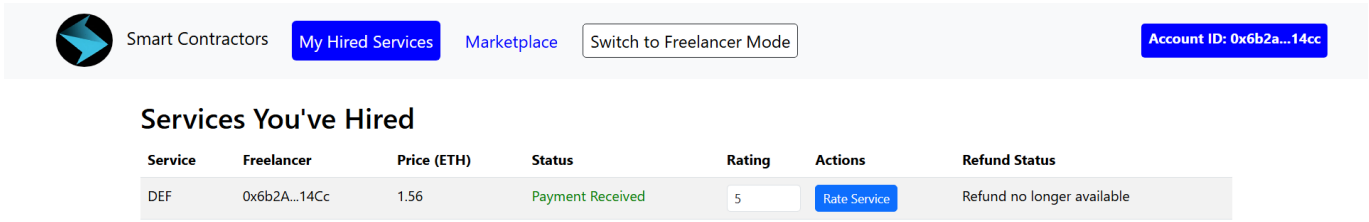
This displays the My Services section of the Smart Contractors platform. It lists the freelancer’s active services, showing details such as the Service Title, Price (in ETH), Status, and Rating.



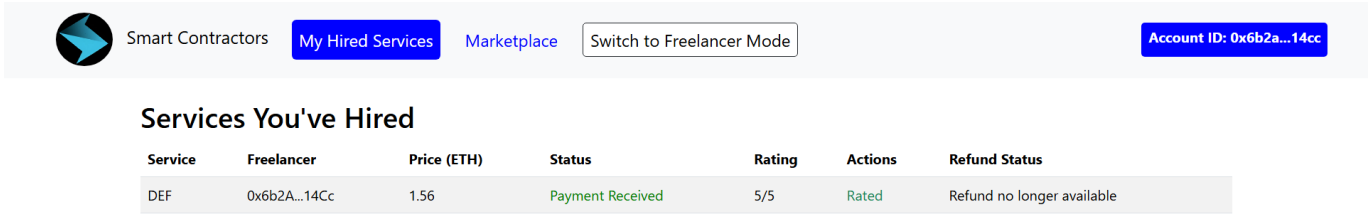
Clients can explore available services on the Smart Contractors platform, such as “web development”, directly from the Marketplace section. They can view details like pricing, description, freelancer ID, availability, and ratings before choosing to hire.



Clients can view the services they've hired in the "My Hired Services" section, which displays key details such as the freelancer's ID, service price, and project status. For active projects, clients have the option to release payment once satisfied with the service. Additionally, in this case there's a refund window of 15 days—if the service is not delivered within this period, the client can claim a refund.



Clients can rate the freelancer after releasing the payment, ensuring feedback is given only once the service has been delivered. After releasing payment, the refund option becomes unavailable.



The given rating affects the freelancer's overall rating and the status for the service is marked as "Rated".

GitHub Repository: https://github.com/Abhinav-Bitragunta/SmartContractors-Freelance-Marketplace