

Aim:

To design and Implement Mini-project on Machine learning / Blockchain topics

PawFund - Decentralized Rescue System (DRS)**Theory**

Animal rescue and adoption processes across the world face serious issues such as lack of transparency, fragmented systems, and misuse of donor funds. Many rescue centers and shelters work independently without a unified platform, making it difficult to track animals, manage resources, or build trust with donors and adopters. Inconsistent record-keeping and the absence of a verifiable system often discourage public participation and lead to inefficiencies in helping animals in need.



PawFund – Decentralized Rescue System (DRS) addresses these challenges by providing a blockchain-based platform that ensures transparent, secure, and verifiable rescue operations. Using smart contracts, all activities like animal registration, adoptions, and donations are immutably recorded on the blockchain, making them easily traceable and tamper-proof. PawFund connects shelters, volunteers, adopters, and donors in a decentralized network, removing the need for centralized authorities while ensuring fair and automated transactions.

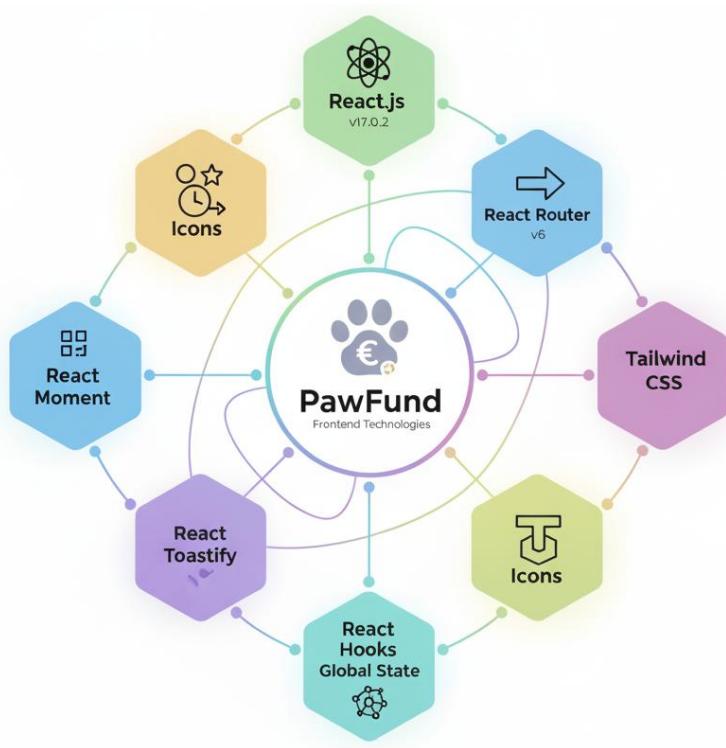


By leveraging blockchain technology, PawFund empowers the animal welfare community with tools for trustless collaboration, verifiable fund distribution, and global accessibility.

The platform also introduces incentive mechanisms to encourage active participation from volunteers and donors, creating a self-sustaining ecosystem for animal rescue.

Tech Stack

Frontend:



Blockchain & Web3:



Steps performed

The implementation of the PawFund Decentralized Animal Rescue System involves several steps to ensure secure donation, rescue, and adoption management using blockchain technology.

Here's the core algorithm followed:

1. User Authentication

- Connect user's digital wallet (MetaMask / Google) to the platform.
- Authenticate user through wallet signature verification.

2. Pet Listing

- Admin/shelter registers a new pet.
- Pet details (name, breed, age, health info) are entered.
- A smart contract function addPet() is called to store the pet on blockchain.

3. Rescue Request

- Any user can create a rescue request by submitting rescue details.
- Rescue request is stored on blockchain via createRescueRequest().

4. Donation Process

- Donors select a rescue case to fund.
- Donation amount is entered, and the smart contract donateToRescue() is invoked.
- Transaction details are recorded immutably.

5. Verification and Updates

- Admin verifies rescue status or adoption post-processing.
- Records are updated via blockchain transaction calls.

Code- Solidity Code File (Blockchain):

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract Genesis {
    address public owner;
    uint public projectTax; uint
    public projectCount; uint
    public balance; statsStruct
    public stats;
    projectStruct[] projects;

    mapping(address => projectStruct[]) projectsOf; mapping(uint
    => backerStruct[]) backersOf; mapping(uint => bool) public
```

```
projectExist;

enum statusEnum {
    OPEN,
    APPROVED,
    REVERTED,
    DELETED,
    PAIDOUT
}

struct statsStruct { uint
    totalProjects; uint
    totalBacking; uint
    totalDonations;
}

struct backerStruct {
    address owner;
    uint contribution;
    uint timestamp;
    bool refunded;
}

struct projectStruct { uint
    id;
    address owner;
    string title;
    string description;
    string imageURL;
    uint cost;
    uint raised;
    uint timestamp;
    uint expiresAt;
    uint backers;
    statusEnum status;
}

modifier ownerOnly(){
    require(msg.sender == owner, "Owner reserved only");
    _;
}

event Action (
    uint256 id,
    string actionType, address
    indexed executor, uint256
    timestamp
);
```

```
constructor(uint _projectTax) {
    owner = msg.sender; projectTax
    = _projectTax;
}

function createProject(
    string memory title,
    string memory description,
    string memory imageURL, uint
    cost,
    uint expiresAt
) public returns (bool) { require(bytes(title).length > 0,
    "Title cannot be
empty");
    require(bytes(description).length > 0, "Description cannot be
empty");
    require(bytes(imageURL).length > 0, "ImageURL cannot be
empty");
    require(cost > 0 ether, "Cost cannot be zero");

    projectStruct memory project; project.id
    = projectCount; project.owner =
    msg.sender; project.title = title;
    project.description = description;
    project.imageURL = imageURL;
    project.cost = cost; project.timestamp =
    block.timestamp; project.expiresAt =
    expiresAt;

    projects.push(project);
    projectExist[projectCount] = true;
    projectsOf[msg.sender].push(project);
    stats.totalProjects += 1;

    emit Action (
        projectCount++,
        "PROJECT CREATED",
        msg.sender,
        block.timestamp
    );
    return true;
}

function updateProject( uint
    id,
    string memory title, string
    memory description, string
    memory imageURL, uint
    expiresAt
```

```
    ) public returns (bool) {
        require(msg.sender == projects[id].owner, "Unauthorized
Entity");
        require(bytes(title).length > 0, "Title cannot be empty");
        require(bytes(description).length > 0, "Description cannot be
empty");
        require(bytes(imageURL).length > 0, "ImageURL cannot be
empty");

        projects[id].title = title; projects[id].description =
description; projects[id].imageURL = imageURL;
        projects[id].expiresAt = expiresAt;

        emit Action (
            id,
            "PROJECT UPDATED",
            msg.sender,
            block.timestamp
        );

        return true;
    }

    function deleteProject(uint id) public returns (bool) {
        require(projects[id].status == statusEnum.OPEN,
"Project no longer opened");
        require(msg.sender == projects[id].owner, "Unauthorized
Entity");

        projects[id].status = statusEnum.DELETED; performRefund(id);

        emit Action (
            id,
            "PROJECT DELETED",
            msg.sender,
            block.timestamp
        );

        return true;
    }

    function performRefund(uint id) internal {
        for(uint i = 0; i < backersOf[id].length; i++) { address
_owner = backersOf[id][i].owner;
        uint _contribution = backersOf[id][i].contribution;

        backersOf[id][i].refunded = true;
        backersOf[id][i].timestamp = block.timestamp;
        payTo(_owner, _contribution);
    }
}
```

```
        stats.totalBacking -= 1; stats.totalDonations -=
        _contribution;
    }

}

function backProject(uint id) public payable returns (bool)
{
    require(msg.value > 0 ether, "Ether must be greater
than zero");
    require(projectExist[id], "Project not found");
    require(projects[id].status == statusEnum.OPEN,
"Project no longer opened");
    stats.totalBacking += 1;
    stats.totalDonations += msg.value;
    projects[id].raised += msg.value;
    projects[id].backers += 1;

    backersOf[id].push(
        backerStruct(
            msg.sender,
            msg.value,
            block.timestamp,
            false
        )
    );
}

emit Action (
    id,
    "PROJECT BACKED",
    msg.sender,
    block.timestamp
);

if(projects[id].raised >= projects[id].cost) {
    projects[id].status = statusEnum.APPROVED; balance +=
    projects[id].raised; performPayout(id);
    return true;
}

if(block.timestamp >= projects[id].expiresAt) {
    projects[id].status = statusEnum.REVERTED;
    performRefund(id);
    return true;
}

return true;
```

```
}

function performPayout(uint id) internal { uint
    raised = projects[id].raised;
    uint tax = (raised * projectTax) / 100; projects[id].status =
    statusEnum.PAIDOUT; payTo(projects[id].owner, (raised - tax));
    payTo(owner, tax);

    balance -= projects[id].raised; emit

        Action (
            id,
            "PROJECT PAID OUT",
            msg.sender,
            block.timestamp
        );
}

function requestRefund(uint id) public returns (bool) { require(
    projects[id].status != statusEnum.REVERTED ||
    projects[id].status != statusEnum.DELETED, "Project
    not marked as revert or delete"
);

    projects[id].status = statusEnum.REVERTED; performRefund(id);
    return true;
}

function payOutProject(uint id) public returns (bool) {
    require(projects[id].status == statusEnum.APPROVED,
    "Project not APPROVED"); require(
        msg.sender == projects[id].owner || msg.sender ==
        owner,
        "Unauthorized Entity"
);

    performPayout(id);
    return true;
}

function changeTax(uint _taxPct) public ownerOnly { projectTax =
    _taxPct;
}

function getProject(uint id) public view returns (projectStruct
```

```
memory) {
    require(projectExist[id], "Project not found");
    return projects[id];
}

function getProjects() public view returns (projectStruct[] memory)
{
    return projects;
}

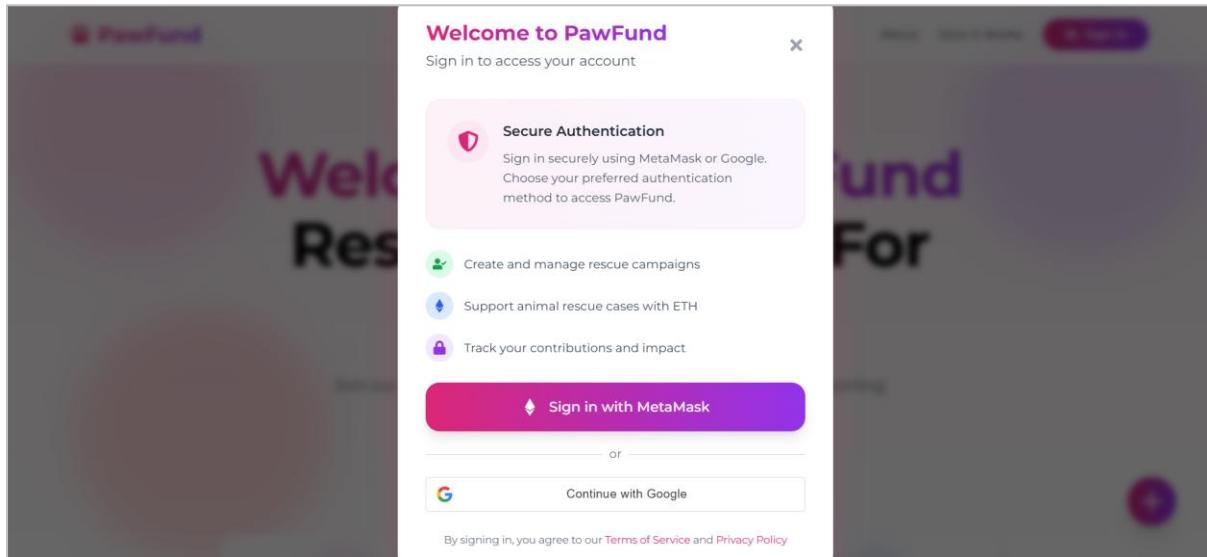
function getBackers(uint id) public view returns (backerStruct[]
memory) {
    return backersOf[id];
}

function payTo(address to, uint256 amount) internal {
    (bool success, ) = payable(to).call{value: amount}("");
    require(success);
}

}
```

Output screenshots

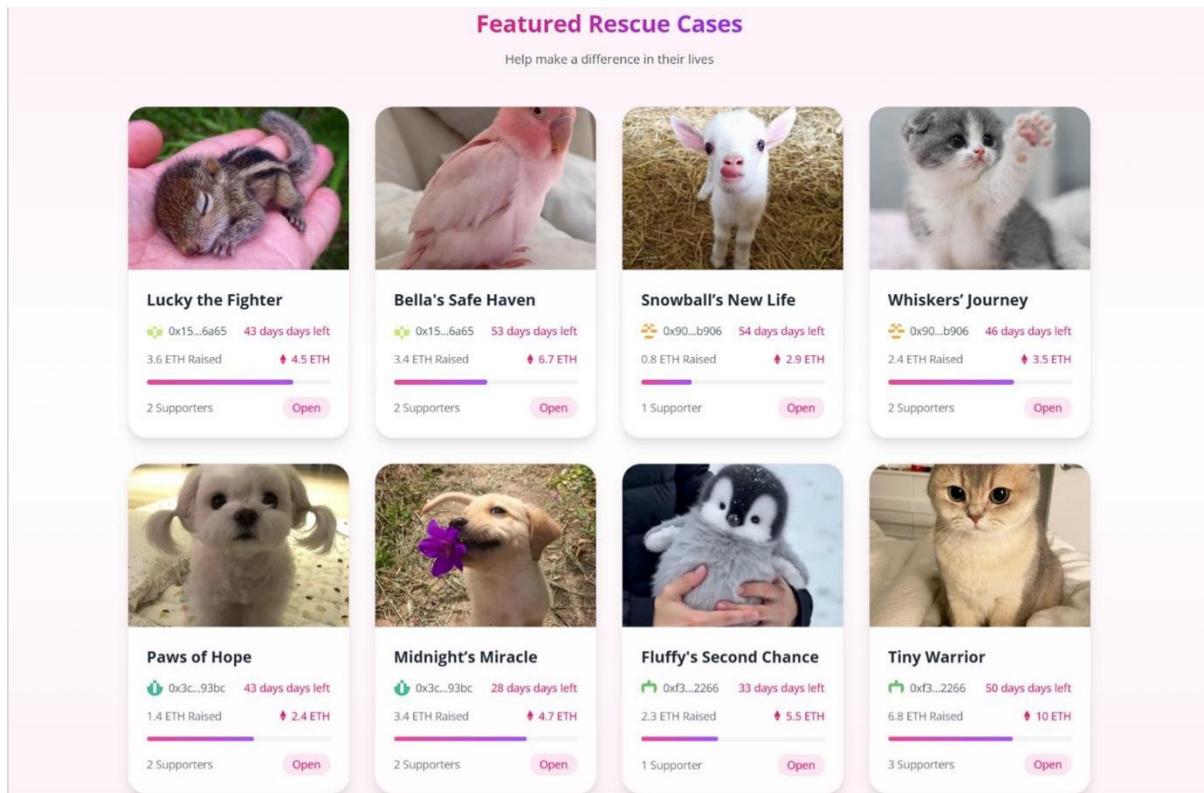
Welcome Page:



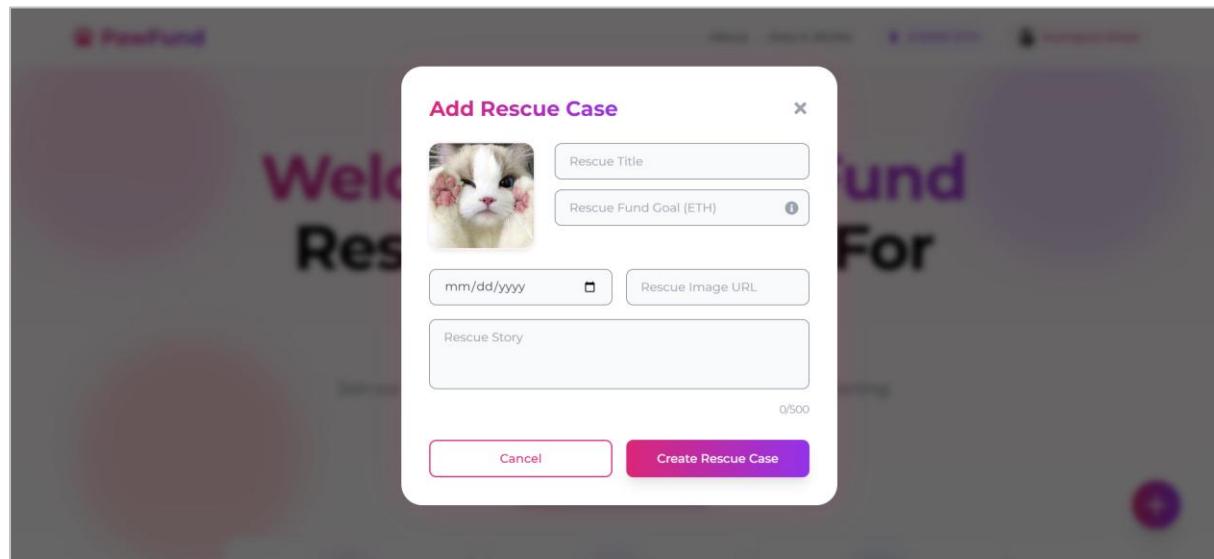
Homepage:

A screenshot of the PawFund homepage. At the top, there's a navigation bar with the PawFund logo, links for "About", "How It Works", and a balance of "0.0000 ETH", and a user profile for "Humayun Khan". Below the header, a large banner features the text "Saving Lives, One Rescue at a Time" above a main title "Welcome To PawFund" in large, bold, magenta letters, with "Rescue Platform For ANIMALS" in black below it. Underneath the title, a subtitle reads: "Join our community of animal lovers and help save lives by supporting animal rescue cases around the world." A prominent pink button labeled "+ ADD RESCUE CASE" is centered. At the bottom, there are three summary cards: "Total Rescues" (0), "Total Supporters" (0), and "Total Donated" (0 ETH). The background of the page has a pink-to-white gradient.

Rescue cases dashboard:



Create rescue case:



About us:

PawFund

[About](#) [How It Works](#) [0.0000 ETH](#) Humayun Khan

Our Mission & Values

About PawFund

Empowering the global community to make a difference in animal welfare through blockchain technology.

Our Mission

PawFund was created with a simple yet powerful mission: to revolutionize animal rescue funding through blockchain technology. We believe that every animal deserves a chance at a better life, and by leveraging the transparency and efficiency of blockchain, we're making it easier than ever for people to help animals in need.

Our platform connects animal rescuers directly with supporters worldwide, ensuring that every contribution makes a real, verifiable impact. Through smart contracts and decentralized technology, we're bringing transparency and trust to animal welfare funding.



Our Values



Animal Welfare First

Every decision we make is guided by what's best for the animals in need.



Compassion

We believe in treating every life with kindness, dignity, and respect.



Transparency

Our blockchain technology ensures complete transparency in how funds are used.



Trust & Security

We maintain the highest standards of security and ethical practices.

Our Impact

500+

Animals Rescued

10k+

Global Supporters

50+

Partner Organizations

Join Our Mission

Together, we can make a difference in the lives of animals in need. Join our community and be part of the change.

[Get Started](#)

PawFund

Empowering the global community to make a difference in animal welfare through blockchain technology.

[Twitter](#) [Facebook](#) [Instagram](#) [YouTube](#) [LinkedIn](#)

Quick Links

[Home](#) [About](#) [How It Works](#) [Contact](#)

Stay Updated

Subscribe to our newsletter for the latest rescue updates.

[Subscribe](#)

How it works:

PawFund

About How It Works 0x97...0aa9

Simple & Transparent

How PawFund Works

Making animal rescue funding transparent, efficient, and impactful through blockchain technology.

Create a Rescue Case

Start by creating a detailed rescue case. Include the animal's story, medical needs, and funding goal. Add photos and specifics about the care required.

- 1 Click 'Add Rescue Case' button
- 2 Fill in details about the animal
- 3 Upload clear photos
- 4 Set funding goal and timeline
- 5 Submit for community support

Smart Contract Creation

Once submitted, a smart contract is automatically created on the blockchain. This ensures transparency and security of funds.

- 1 Contract deployed to blockchain
- 2 Unique identifier generated
- 3 Funding goal encoded
- 4 Timeline parameters set
- 5 Ready for contributions

Secure Contributions

Supporters can contribute ETH directly to the rescue case. All transactions are recorded on the blockchain for complete transparency.

- 1 Connect crypto wallet
- 2 Choose contribution amount
- 3 Confirm transaction
- 4 Receive confirmation
- 5 Track contribution impact

Fund Distribution

When the funding goal is reached, funds are automatically released to the rescue organization through the smart contract.

- 1 Goal achievement verified
- 2 Automatic fund release
- 3 Updates posted
- 4 Impact tracked
- 5 Success story shared

Featured cases:

PawFund

About How It Works 0xf3...2266

Featured Rescue Cases

Help make a difference in their lives

Snowball's New Life

0x90...b906 54 days days left

0.8 ETH Raised ⚡ 2.9 ETH

1 Supporter Open

Whiskers' Journey

0x90...b906 46 days days left

1.3 ETH Raised ⚡ 3.5 ETH

1 Supporter Open

Paws of Hope

0x3c...93bc 43 days days left

1 ETH Raised ⚡ 2.4 ETH

1 Supporter Open

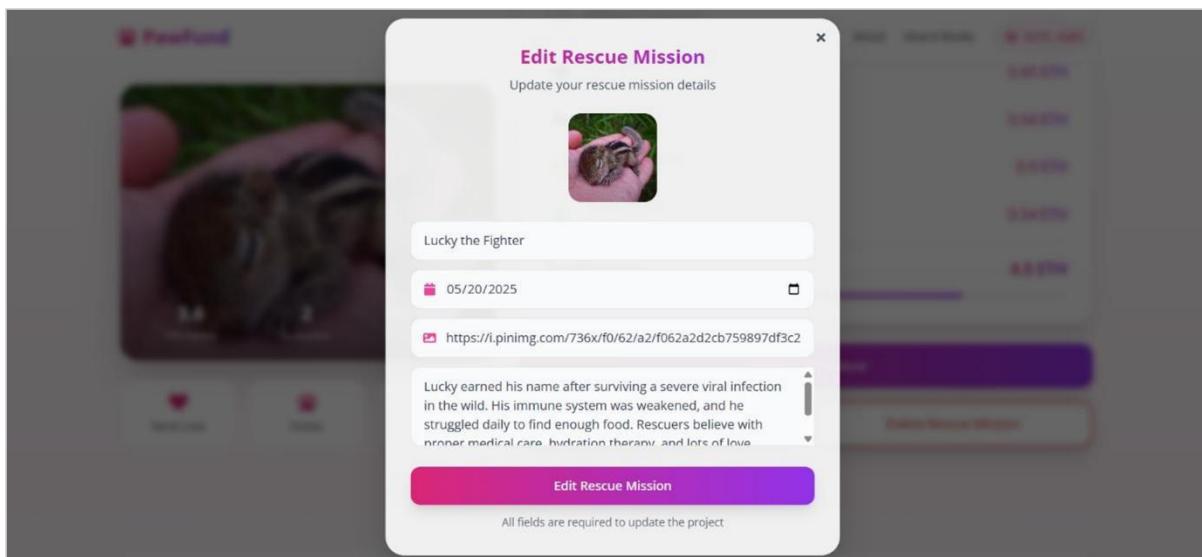
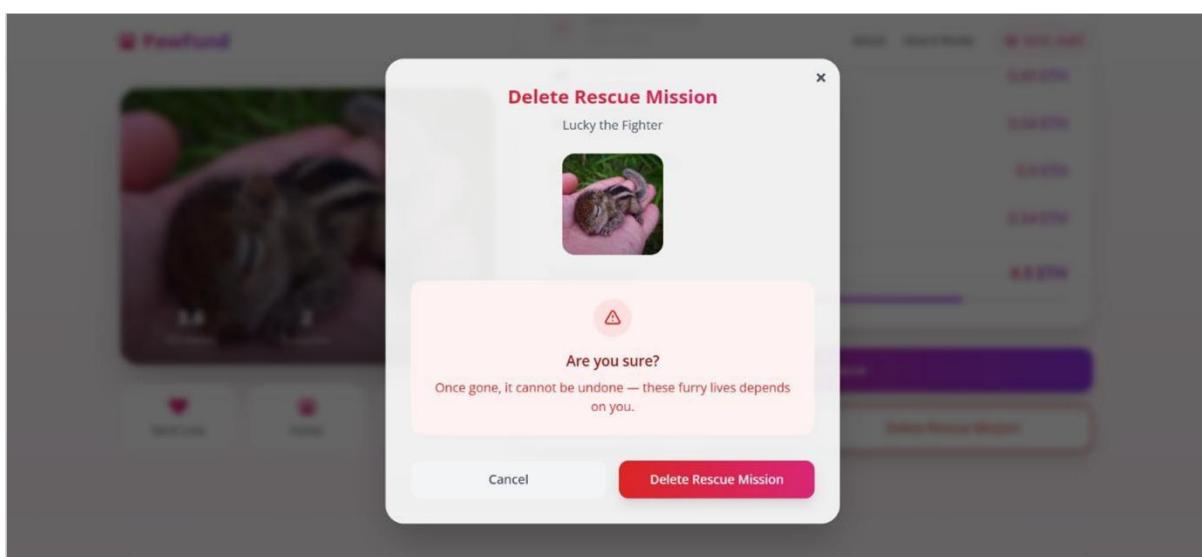
Midnight's Miracle

0x3c...93bc 28 days days left

2.4 ETH Raised ⚡ 4.7 ETH

1 Supporter Open

[Show more Rescue Cases](#)

Create new case campaign:**Edit Mission:****Delete Mission:**

Support & Donation page:

The screenshot shows a rescue campaign for a goat named Snowball. At the top, there is a navigation bar with the PawFund logo, links for 'About', 'How It Works', and a wallet address '0xf3...2266'. Below the title 'Snowball's New Life' is a photo of a white goat standing in straw. A progress bar indicates '0.8 ETH raised of 2.9 ETH'. The 'About This Rescue' section describes how Snowball was found shivering under a car during a winter storm and needs medical attention. The 'Cost Breakdown' section details the following expenses:

Category	Percentage of Total	Amount
Medical Treatment	22%	0.638 ETH
Foster Care	49%	1.421 ETH
Medications	17%	0.493 ETH
Food & Supplies	12%	0.348 ETH
Total Required		2.9 ETH

A large purple button at the bottom says 'Support This Rescue'.

Supporters List:

The supporters list shows one entry for 'Rescue Supporters (1)'. The table includes columns for 'Supporter', 'Donation', 'Refunded', and 'Time'. The single supporter listed is '0x3c...93bc' who donated '0.8 ETH' 8 minutes ago.

Supporter	Donation	Refunded	Time
0x3c...93bc	0.8 ETH	No	8 minutes ago

User testimonials:

The screenshot shows the PawFund homepage with a pink-to-white gradient background. At the top left is the PawFund logo. At the top right are links for "About", "How It Works", and a user handle "0xf3...2266". Below the header is a section titled "Voices of Impact" with the subtitle "Hear from our community members who are making a difference in animal welfare". Three testimonial cards are displayed, each with a quote in a speech bubble and a profile picture of the testimonialer.

- Sarah Johnson**, Rescue Supporter: "PawFund made it incredibly easy to support animal rescue cases. The transparency of blockchain technology gives me confidence that my contributions are making a real difference."
- Dr. Michael Chen**, Veterinarian: "As a vet, I've seen how PawFund has revolutionized animal rescue funding. The platform ensures quick access to funds when animals need urgent care."
- Emma Rodriguez**, Animal Shelter Director: "The smart contract system has streamlined our fundraising process. We can now focus more on saving animals and less on administrative tasks."

Below the testimonials, there's a banner stating "Trusted by animal lovers worldwide" followed by four badges: "500+ Successful Rescues", "100% Transparent", "Global Community", and "Blockchain Secured". A large purple "+" button is located in the bottom right corner.

FAQ Section:

The screenshot shows the "Frequently Asked Questions" section of the PawFund website. The title is "Frequently Asked Questions" with the subtitle "Everything you need to know about using PawFund". Three questions are listed in boxes:

- What is PawFund?**: PawFund is a decentralized platform that connects animal rescuers with supporters worldwide using blockchain technology. It ensures transparent and efficient funding for animal rescue cases.
- How do I contribute to a rescue case?**: Connect your crypto wallet, choose a rescue case, and click 'Support This Rescue'. Enter the amount you wish to contribute in ETH and confirm the transaction.
- Is my contribution secure?**: Yes! All contributions are secured by smart contracts on the blockchain. Every transaction is transparent and traceable.

Profile Page:

The screenshot shows the "My Profile" page for a user named "Humayun Khan". The profile card at the top includes a circular profile picture, the name "Humayun Khan", the title "MERN Stack Developer | Next.js Specialist | Building Real-World Web Apps with Developer-Friendly UI Design & Clean Code", the location "Thane, India", and the member since date "October 5, 2025". There is also an "Edit Profile" button. Below the card are tabs for "Profile", "Activity", and "Preferences", with "Profile" being the active tab. Under the "Social Media" section, there are links for Twitter, GitHub, and LinkedIn. Under the "Account Information" section, it shows "Member Since: October 5, 2025", "Account Status: Verified", and "Authentication: Web3 (MetaMask)". A "Copy" button is located near the wallet address field.

Conclusion

PawFund is a decentralized blockchain-based platform designed to transform global pet fundraising. By removing intermediaries, it ensures donations are transparent, secure, and directly reach animal shelters and rescue organizations. With blockchain's immutability and real-time tracking, donors gain unmatched trust and visibility, while decentralized storage (e.g., IPFS) safeguards sensitive campaign data. Despite its promise, challenges like regulatory compliance, user adoption, blockchain complexity, and scalability remain. Addressing these with user-friendly interfaces, multilingual support, smart contract enhancements, and integration with traditional banking can boost accessibility and impact.