

## Specification

The Earth Restoration Investment Platform (ERIP) has a set of technical and functional requirements that are crucial for its successful implementation and operation. These requirements are as follows:

**Technology Stack** - The platform uses MongoDB for flexible and scalable data storage, Express.js for backend logic and API development, and Node.js for server-side operations and real-time data processing. React.js with Vite.js framework is used for dynamic and responsive user interfaces, while Material-UI is used for improved design aesthetics and user experience.

**Cloud Infrastructure** - ERIP leverages AWS IoT Core to manage and connect IoT devices for weather and soil monitoring and Google Cloud Platform for geolocation analysis and real-time weather calculations.

**Integration** - The platform is integrated with third-party APIs for geolocation services, weather data providers, and secure payment gateways. Stripe is the payment gateway of choice for handling financial transactions securely.

**Testing Tools** - Postman is used for comprehensive API testing to ensure functionality, security, and data accuracy. The browser's console is used for front-end testing and validation.

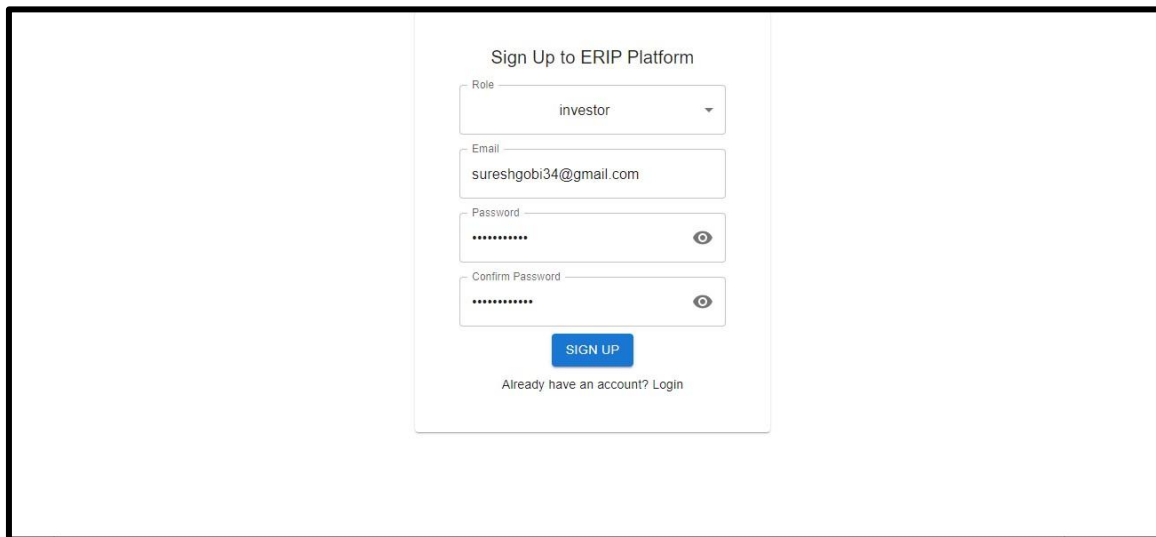
**Scalability and Performance** - The platform is designed for scalability to accommodate growing user bases and data volumes. It is also optimized for fast loading times, seamless interactions, and real-time updates.

**Security** - The platform has implemented robust security measures, including encryption protocols, secure authentication mechanisms, and data privacy safeguards. Regular security audits and updates are conducted to ensure compliance with industry standards and regulations.

**User Experience** - ERIP offers user-friendly interfaces with intuitive navigation, responsive design, and accessibility features. It also has an inbuilt CDN for content optimization and delivery, enhancing overall user experience.

## Frontend

### Signup to ERIP Platform

A screenshot of the 'Sign Up to ERIP Platform' form. The form is centered on a white background with a thin black border. It contains the following fields: a 'Role' dropdown menu with 'investor' selected, an 'Email' text field with 'sureshgobi34@gmail.com', a 'Password' text field with masked characters and a toggle icon, and a 'Confirm Password' text field with masked characters and a toggle icon. Below these fields is a blue 'SIGN UP' button. At the bottom, there is a link that says 'Already have an account? Login'.

*Figure 21 Signup to ERIP Platform*

The signup form where both investors and environmental activists can sign up. After successful signup, an OTP is sent to the provided email. Once the OTP is confirmed and verified, the signup process is completed, and the user is redirected to the login page.

## Login

The screenshot shows a login page with a dark header bar. On the left, there is a 'Go Back' link with a left-pointing arrow. On the right, there is a button labeled 'ADMIN DASHBOARD'. Below the header, there are two main login sections. The 'Investor Login' section on the left includes a title, a descriptive paragraph, an 'Email \*' input field, a 'Password \*' input field, and a blue 'LOGIN' button. The 'Activist Login' section on the right includes a title, a descriptive paragraph, an 'Email' input field, a 'Password' input field, and a blue 'LOGIN' button. At the bottom of the page, there is a link that reads 'Both Investor and Activist Don't have an account? Sign Up'.

Figure 22 Login

Where investor login and environment activist login are in the same page, and the desire users can be able to access their accounts with the login credentials and admins can be able to access the admin dashboard login by clicking the “Admin Dashboard” button.

The screenshot shows a single login form titled 'Admin Login'. It contains two input fields: 'Email' and 'Password'. Below these fields is a wide blue button labeled 'LOGIN'.

Figure 23 Admin Login

## Environment Activist Dashboard

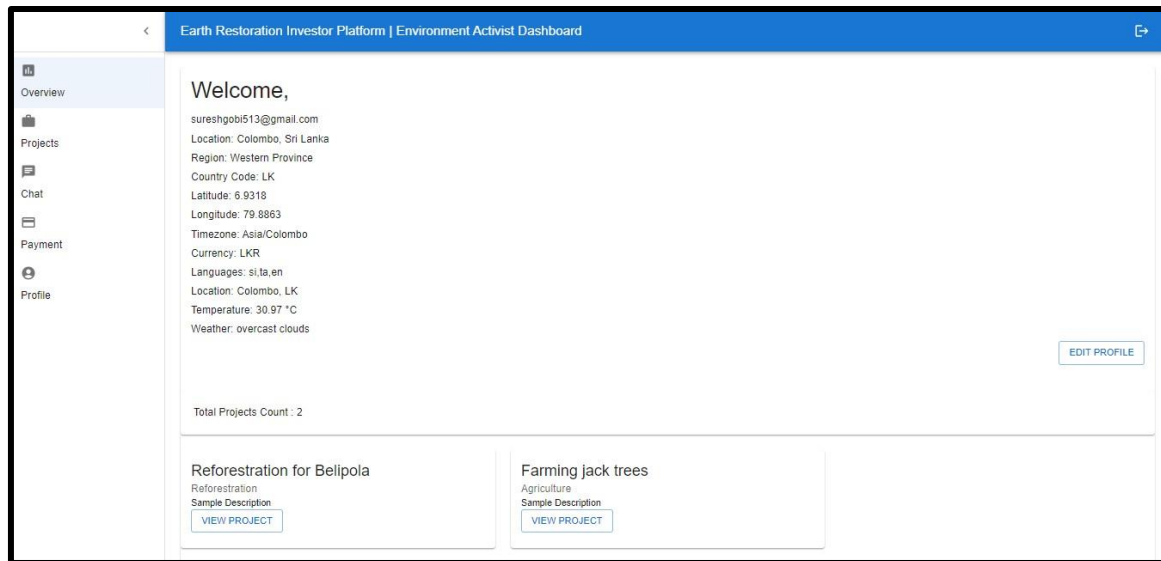


Figure 24 Environment Activist Dashboard

he ERIP's Environment Activist Dashboard shows the logged user's data and utilizes the current IP address to retrieve the current location, time zone, currency, languages spoken. It also uses the Open Weather Data API to display the current weather conditions of the location. In the dashboard overview, it displays the current total projects created by activists and approved by admins.

## Project Tabs

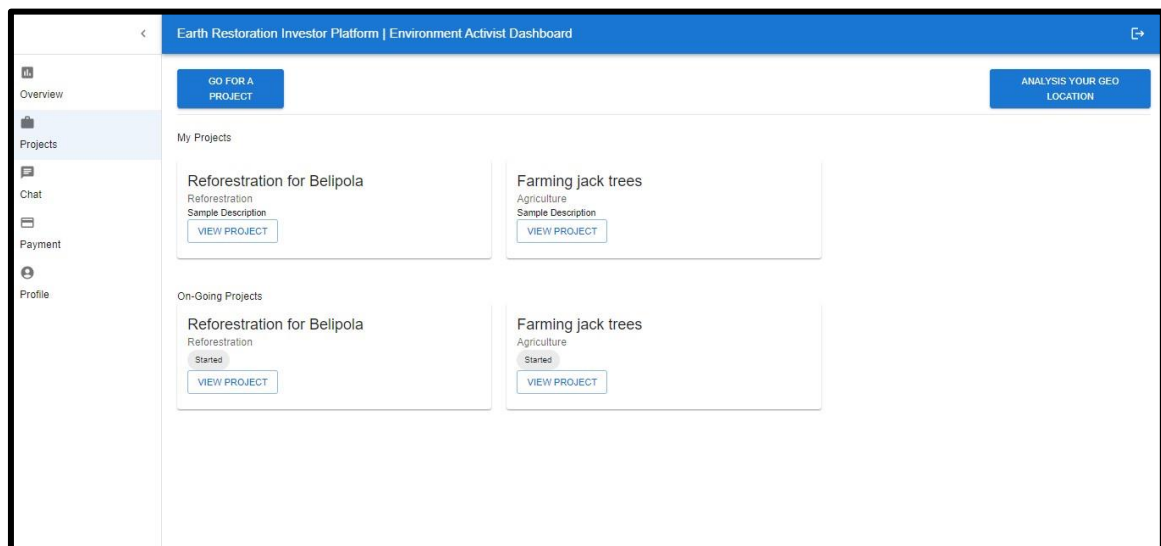
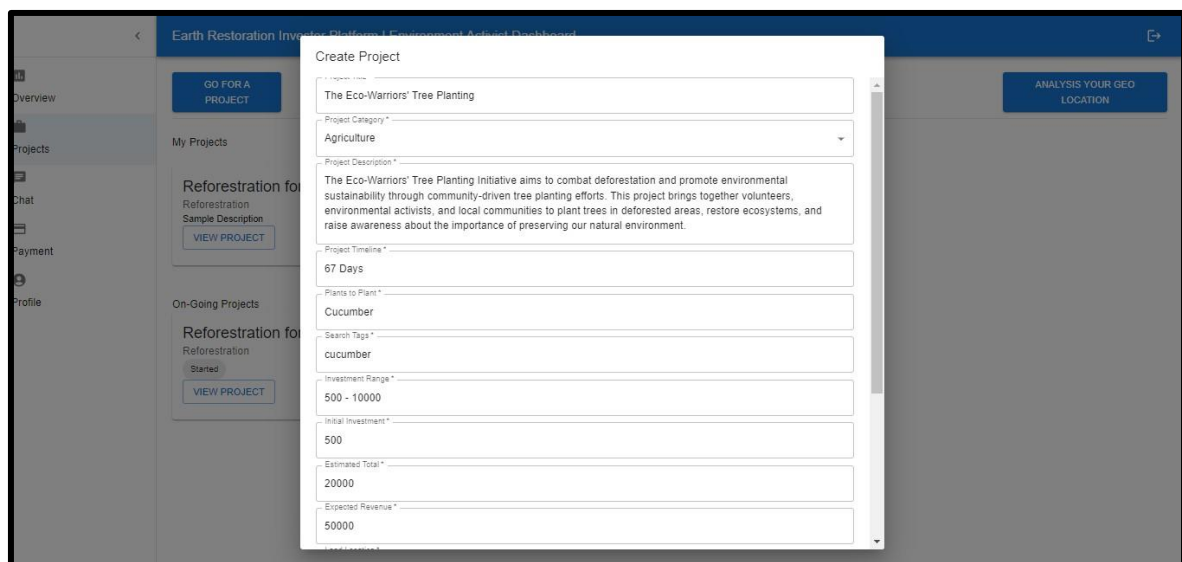


Figure 25 Project Tabs

The Projects tab displays the projects created by activists and ongoing projects initiated by investors who are investing in the projects. If activists wish to create a new project, they can simply click the button called **“Go for a Project”** and fill out the form. The project will then be reviewed and approved by the admin. Once approved, the project will become visible to investors. Before creating a project on a specific land or geo location, it's advisable to analyze the land by clicking the button **"Analyze Your GEO Location"**.

## Create New project



The screenshot shows the 'Create Project' form within the Earth Restoration Investor Platform. The form is titled 'Create Project' and contains the following fields:

- Title:** The Eco-Warriors' Tree Planting
- Project Category:** Agriculture
- Project Description:** The Eco-Warriors' Tree Planting Initiative aims to combat deforestation and promote environmental sustainability through community-driven tree planting efforts. This project brings together volunteers, environmental activists, and local communities to plant trees in deforested areas, restore ecosystems, and raise awareness about the importance of preserving our natural environment.
- Project Timeline:** 67 Days
- Plants to Plant:** Cucumber
- Search Tags:** cucumber
- Investment Range:** 500 - 10000
- Initial Investment:** 500
- Estimated Total:** 20000
- Expected Revenue:** 50000

The form is displayed over a dashboard background that includes a sidebar with navigation links (Overview, Projects, Chat, Payment, Profile) and a main content area with buttons for 'GO FOR A PROJECT' and 'ANALYSIS YOUR GEO LOCATION'.

Figure 26 Create New project

# View Created Project

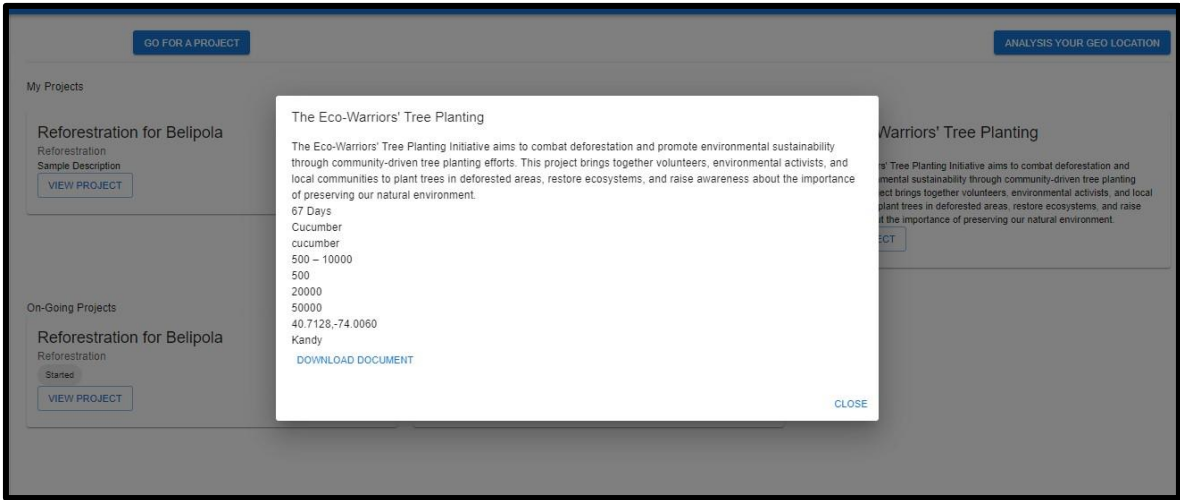


Figure 27 View Created Project

# On-going Project Management

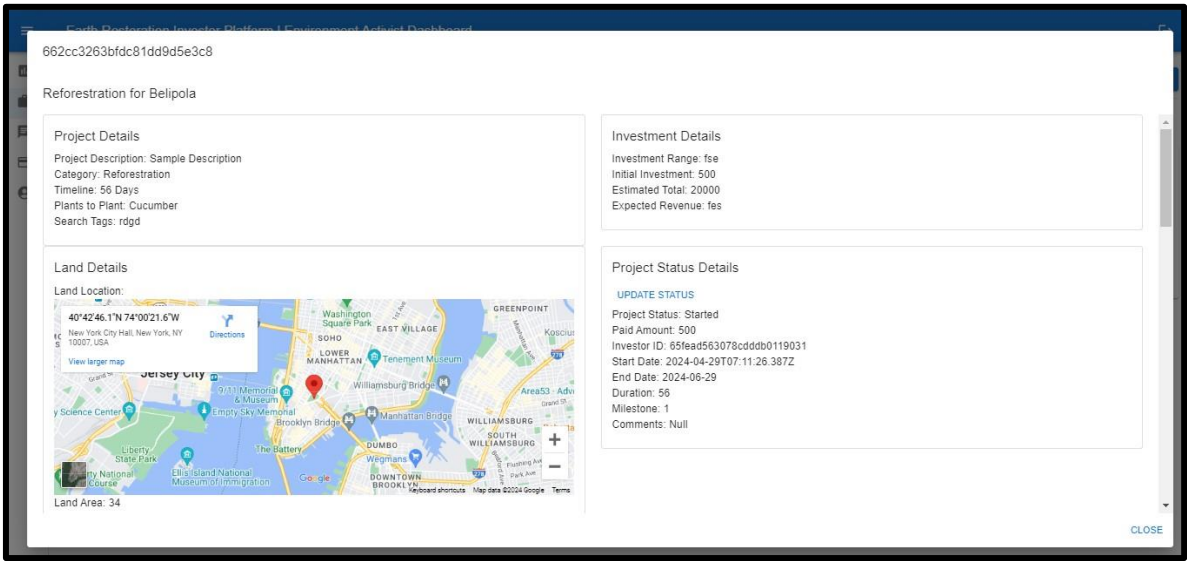


Figure 28 On-going Project Management 1

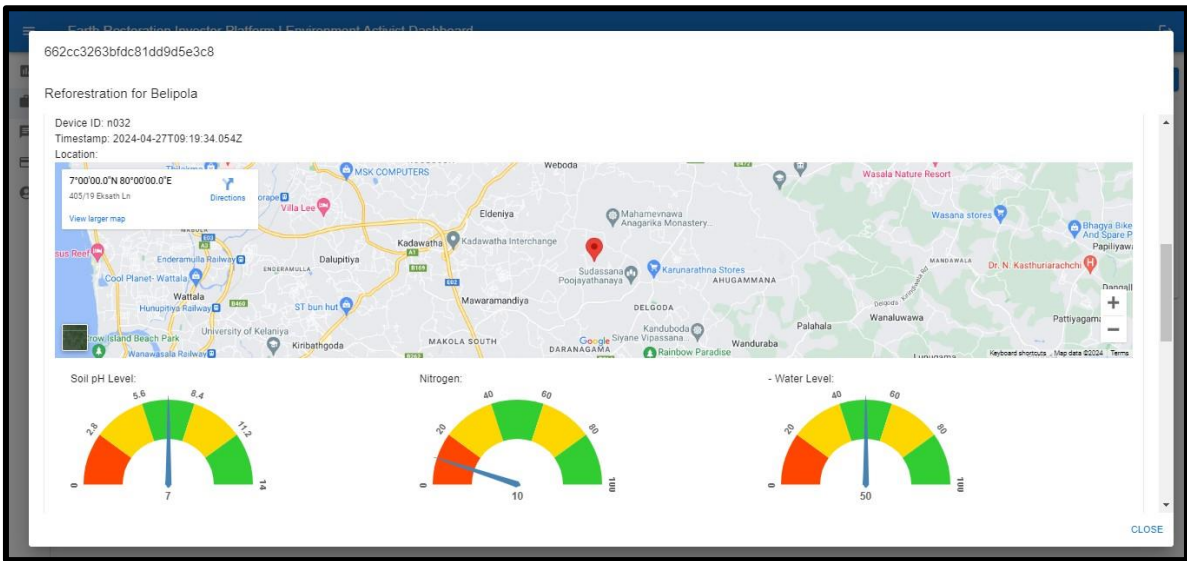


Figure 29 On-going Project Management 2



Figure 30 On-going Project Management 3

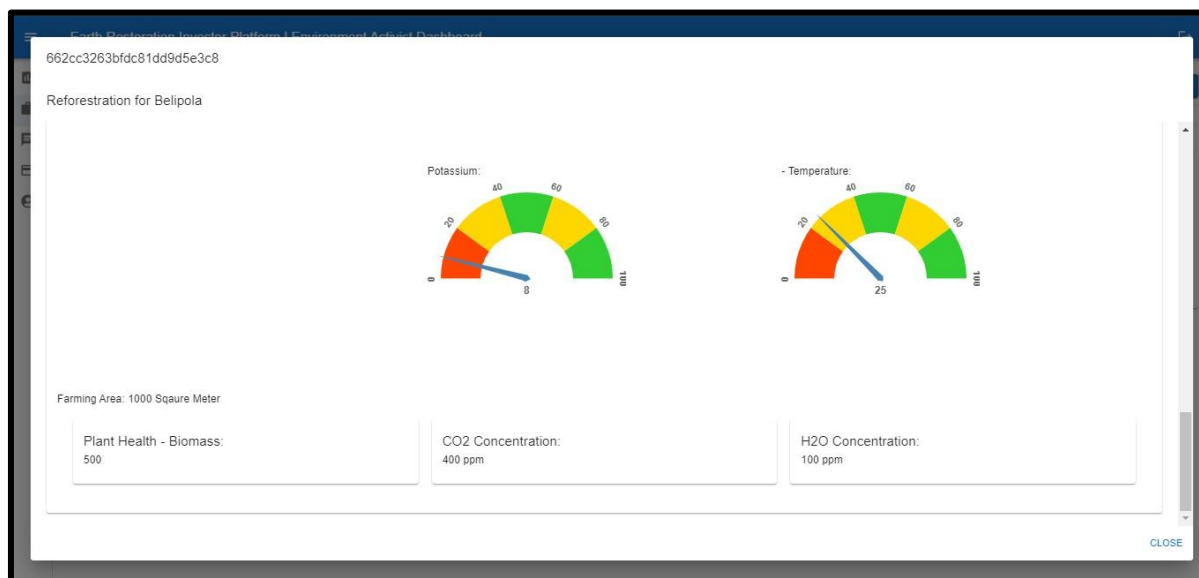


Figure 31 On-going Project Management 4

In the ongoing project environment analysis, the system can display the entire project details such as project status and land details. Activists can request the admin to set up IoT devices to monitor weather and soil, providing better insights for investors.



# Geo Location Analysis

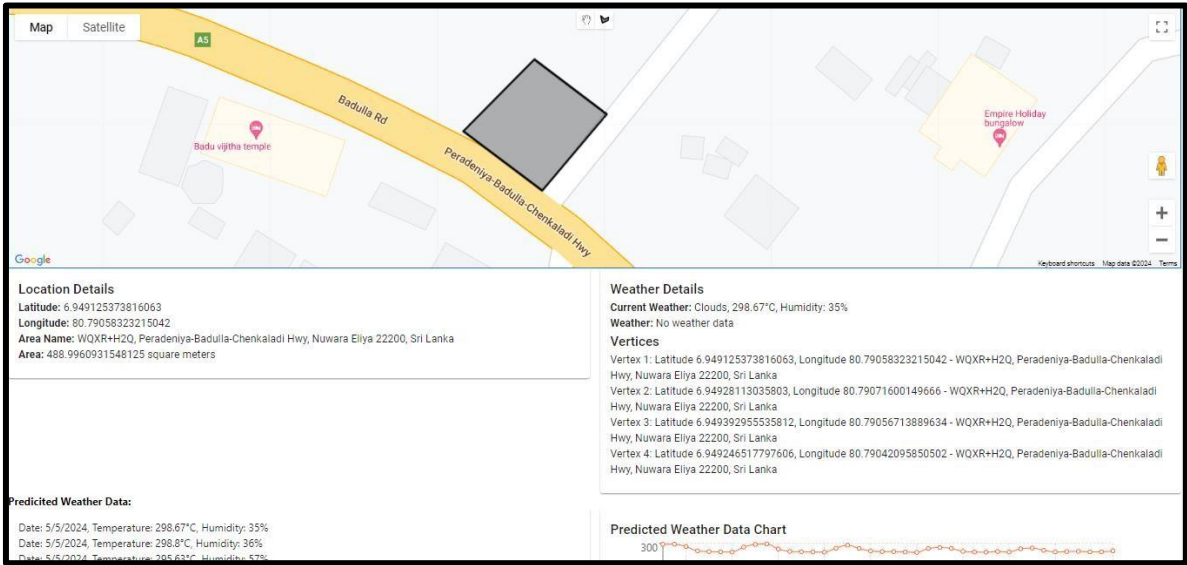


Figure 32 Geo Location Analysis 1

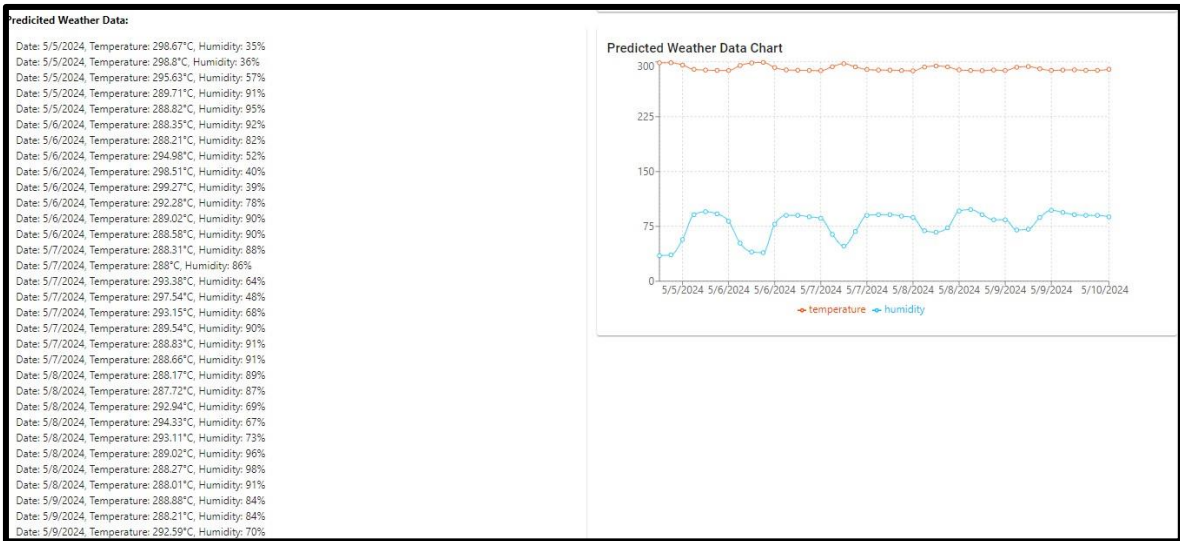


Figure 33 Geo Location Analysis 2

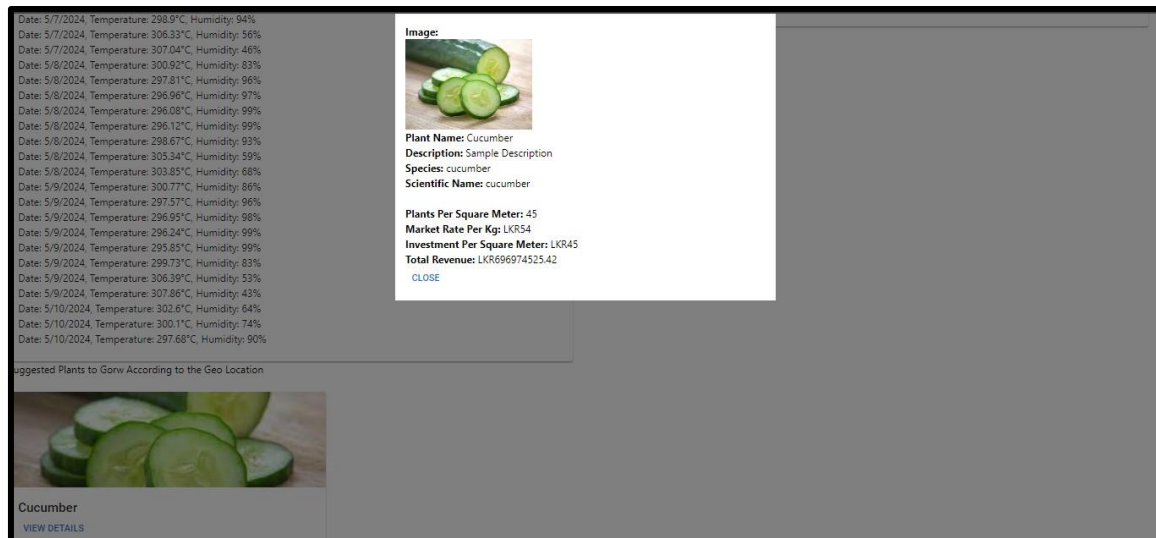


Figure 34 Geo Location Analysis 3

Geo Location Analysis in the ERIP project is vital for informed decision-making. It provides real-time and historical weather data, aiding in weather-related risk assessment and resource optimization. Predicted weather data forecasts future conditions, enhancing planning accuracy. By correlating weather data with plant requirements, suitable species are suggested, crucial for agricultural and restoration efforts. Economic analysis integrates market data, estimating revenue potential and guiding financial decisions. This comprehensive approach optimizes project planning, resource allocation, and environmental impact, making Geo Location Analysis a cornerstone in ensuring the success, sustainability, and resilience of the ERIP project.

(Intentionally blank)

## Code for GEO Location Analysis

```
const GeoLocationAnalysis = () => {
  useEffect(() => {
    // Load the Google Maps API script dynamically
    const script = document.createElement("script");
    script.src = `https://maps.googleapis.com/maps/api/js?key=AIzaSyDnOmZ9Nv82BJpiRuNHZlT55c`;
    script.async = true;
    script.onload = initializeMap;
    document.head.appendChild(script);

    return () => [
      document.head.removeChild(script);
    ];
  }, []);

  const initializeMap = () => {
    const mapOptions = {
      center: { lat: 0, lng: 0 },
      zoom: 2,
    };
    const newMap = new window.google.maps.Map(
      document.getElementById("map"),
      mapOptions
    );
  };
}
```

Figure 35 Code for GEO Location Analysis 1

```

67     const newDrawingManager = new window.google.maps.drawing.DrawingManager({
68         drawingMode: window.google.maps.drawing.OverlayType.POLYGON,
69         drawingControl: true,
70         drawingControlOptions: {
71             position: window.google.maps.ControlPosition.TOP_CENTER,
72             drawingModes: ["polygon"],
73         },
74     });
75
76     newDrawingManager.setMap(newMap);
77
78     window.google.maps.event.addListener(
79         newDrawingManager,
80         "overlaycomplete",
81         (event) => {
82             if (selectedShape) {
83                 selectedShape.setMap(null);
84             }
85
86             const newShape = event.overlay;
87             newShape.type = event.type;
88             setSelectedShape(newShape);
89
90             const area = window.google.maps.geometry.spherical.computeArea(

```

Figure 36 Code for GEO Location Analysis 2

```

95         // Get the vertices of the polygon
96         const vertices = newShape
97             .getPath()
98             .toArray()
99             .map((vertex) => ({
100                 latitude: vertex.lat(),
101                 longitude: vertex.lng(),
102             }));
103
104         setPolygonVertices(vertices);
105
106         // Get the area name for each vertex
107         getAreaNames(vertices);
108         // Get weather data for the area
109         getWeatherData(vertices[0].latitude, vertices[0].longitude);
110     }
111 });
112
113 setMap(newMap);
114 setDrawingManager(newDrawingManager);
115 
```

Figure 37 Code for GEO Location Analysis 3

## Code Weather Analysis

```
const getWeatherData = async (latitude, longitude) => {
  try {
    const response = await axios.get(
      `https://api.openweathermap.org/data/2.5/forecast?lat=${latitude}&lon=${longitude}&`
    );
    setWeatherData(response.data);
    if (response.data && response.data.list) {
      // Filter past three years data by month
      const currentYear = new Date().getFullYear();
      const currentMonth = new Date().getMonth();
      const pastYearsData = response.data.list.filter((item) => {
        const itemDate = new Date(item.dt * 1000);
        return (
          itemDate.getFullYear() >= currentYear - 3 &&
          itemDate.getMonth() <= currentMonth
        );
      });
      setPastWeatherData(pastYearsData);
      setCurrentWeatherData(response.data.list[0]); // Set current weather data
      filterPlantsByHumidity(response.data.list); // Pass weather data to filter function
    }
  } catch (error) {
    console.error("Error fetching weather data:", error);
    setWeatherData(null); // Clear weather data if an error occurs
  }
}
```

Figure 38 Code Weather Analysis 1

## Real-Time one to one chat system

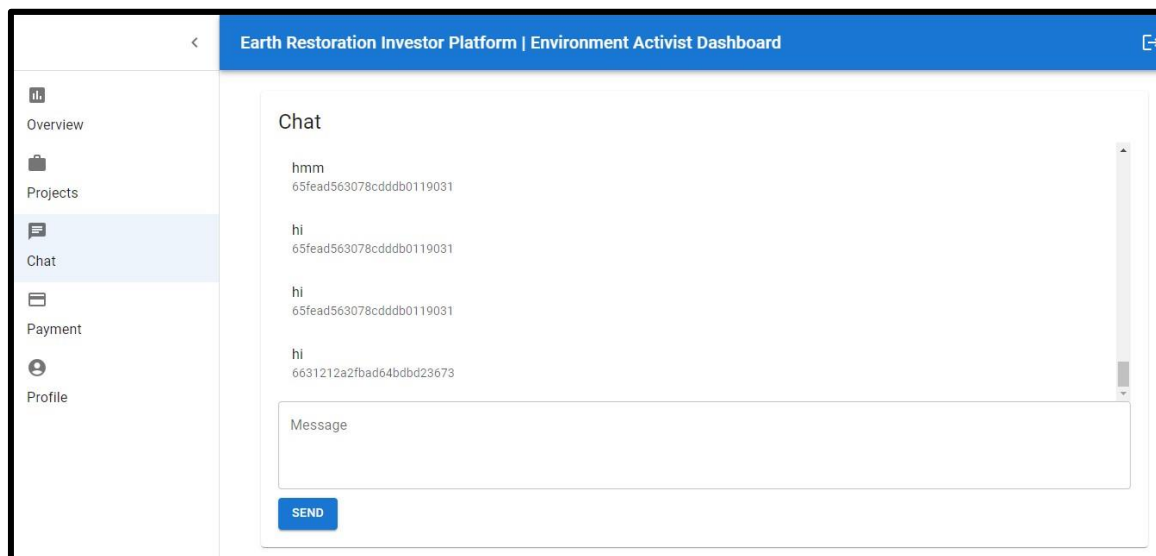


Figure 39 Real-Time one to one chat system

Investors can engage in real-time conversations with activists directly through their associated projects. The chat functionality leverages Socket.io for seamless, instant communication between users. This integration ensures that investors and activists can discuss project details, updates, and collaborate effectively, fostering a dynamic and interactive environment within the platform. Socket.io's capabilities enable live messaging, enhancing engagement and facilitating meaningful interactions between stakeholders involved in the investment process.

**Payment Tab**

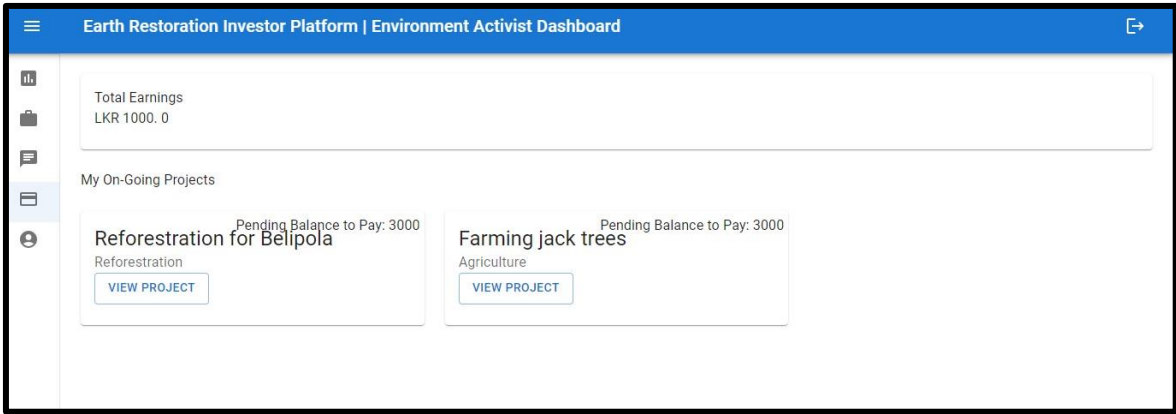


Figure 40 Payment Tab

The activist payment tab displays the total earnings and ongoing projects, as well as the pending amount that the investor intends to pay.

## Account Tab

The screenshot shows the 'Account Tab' of the 'Earth Restoration Investor Platform | Environment Activist Dashboard'. The page has a blue header with the platform name and a user icon. On the left, there is a sidebar with icons for Profile, Bank, Projects, and Payment. The main content area is titled 'Profile' and contains several input fields: 'Name \*', 'Address \*', 'Phone \*', 'NIC \*', 'Nationality \*', 'Bank Name \*', 'Account Number \*', 'Branch \*', 'Bank Code \*', and 'Shift Code \*'. Below these fields is a checkbox for 'Verified Account' and two buttons: 'UPLOAD FILE' and 'SUBMIT'.

Figure 41 Account Tab

This is the page where environment activists submit their bank details and personal information. Only the admin has the authority to verify the activists' bank accounts for fund transactions.

## Investor Dashboard

The screenshot shows the 'Investor Dashboard' of the 'Earth Restoration Investor Platform | Environment Activist Dashboard'. The page has a blue header with the platform name and a user icon. On the left, there is a sidebar with icons for Overviews, Projects, and Payment. The main content area is titled 'Welcome,' and displays user information: 'sureshgobi34@gmail.com', 'Location: Hanwella, Sri Lanka', 'Region: Western Province', 'Country Code: LK', 'Latitude: 6.9345', 'Longitude: 80.0902', 'Timezone: Asia/Colombo', 'Currency: LKR', 'Languages: si,ta,en', 'Location: Hanwella, LK', 'Temperature: 32.21 °C', and 'Weather: overcast clouds'. There is an 'EDIT PROFILE' button. Below this, there is a section titled 'All Projects' with a search bar 'Search by title'. A project titled 'The Eco-Warriors' Tree Planting' is listed with a description: 'The Eco-Warriors' Tree Planting Initiative aims to combat deforestation and promote environmental sustainability through community-driven tree planting efforts. This project brings together volunteers, environmental activists, and local communities to plant trees in deforested areas, restore ecosystems, and raise awareness about the importance of'.

Figure 42 Investor Dashboard



In the investor dashboard, the location, time zone, language, and weather details are displayed using the IP address and OpenWeather. All the approved projects are shown in the Overview.

## All Approved Projects

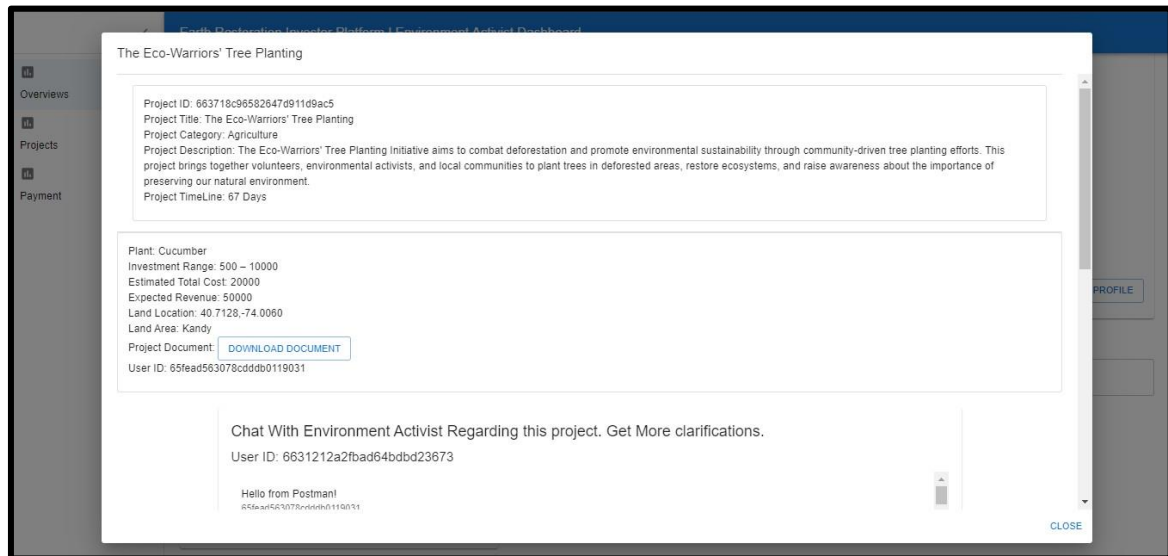


Figure 43 All Approved Projects 1

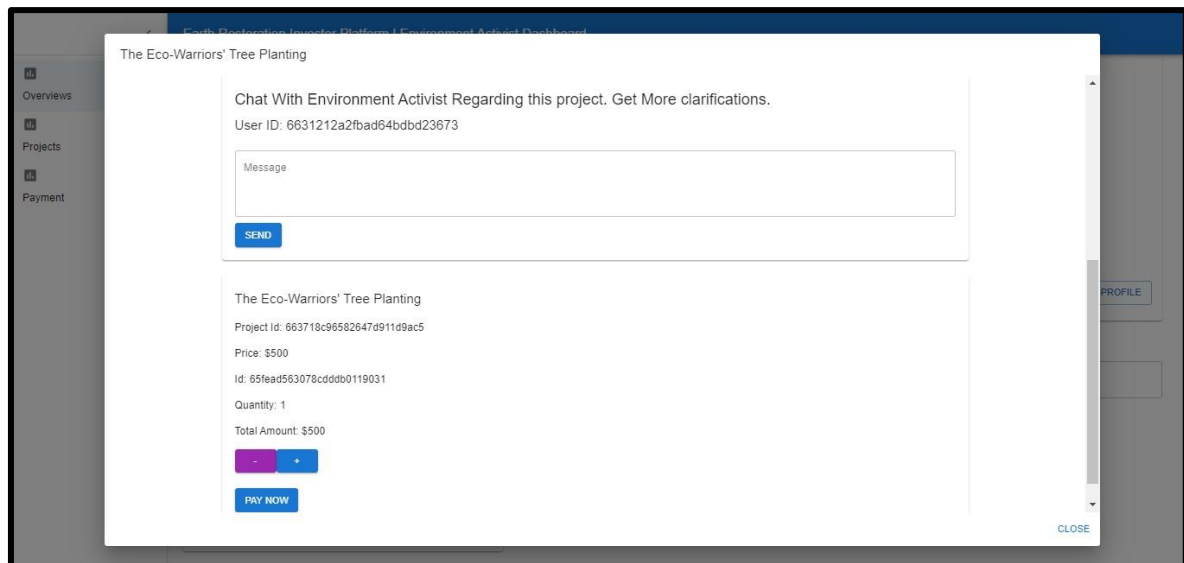


Figure 44 All Approved Projects 2



Investors can view project details comprehensively and engage in discussions with activists using ERIP's reliable chat system. If they decide to proceed, they can initiate the project by making an initial payment securely through Stripe's payment method.

### Code for Chat system

```
15  const socket = io("http://localhost:5000");
16
17  export default function Chat({ resId }) {
18    const [messages, setMessages] = useState([]);
19    const [newMessage, setNewMessage] = useState("");
20    const [userId, setUserId] = useState("");
21
22    useEffect(() => {
23      // Fetch userId from localStorage and decode the token
24      const token = localStorage.getItem("token");
25      if (token) {
26        const decodedToken = decodeToken(token);
27        if (decodedToken && decodedToken.id) {
28          setUserId(decodedToken.id);
29        }
30      }
31
32      // Fetch messages when component mounts
33      const fetchMessages = async () => {
34        try {
35          const response = await fetch(
36            `http://localhost:5000/api/messages?recipient=${userId}`
37          );
38          if (response.ok) {
39            const data = await response.json();
40          }
41        } catch (error) {
42          console.error("Error fetching messages:", error);
43        }
44      };
45      fetchMessages();
46    });
47  }
```

Figure 45 Code for Chat system 1

```
51     const intervalId = setInterval(fetchMessages, 2000);
52
53     // Listen for incoming messages from the server
54     socket.on("message", (message) => {
55         setMessages((prevMessages) => [...prevMessages, message]);
56     });
57
58     // Clean up the socket connection on component unmount
59     return () => {
60         clearInterval(intervalId);
61         socket.disconnect();
62     };
63 }, [resId]);
64
65 const handleSendMessage = async () => {
66     if (newMessage.trim() === "") {
67         return;
68     }
69
70     const messageData = {
71         content: newMessage,
72         sender: userId, // Use userId as the sender
73         recipient: resId, // Use resId as the recipient
```

Figure 46 Code for Chat system 2

## Code for Payment Gateway

```
const checkout = async () => {
    try {
        const token = localStorage.getItem("token");

        const checkoutRes = await fetch(
            "http://localhost:5000/api/payments/checkout",
            {
                method: "POST",
                headers: {
                    "Content-Type": "application/json",
                },
                mode: "cors",
                body: JSON.stringify({
                    items: [
                        {
                            id: 1,
                            quantity: quantity,
                            price: initialInvestment,
                            name: itemName,
                        },
                    ],
                }),
            }
        );
    }
};
```

Figure 47 Code for Payment Gateway 1

```
exports.payment = async (req, res) => {
  try {
    const session = await stripe.checkout.sessions.create({
      payment_method_types: ["card"],
      mode: "payment",
      line_items: req.body.items.map((item) => ({
        price_data: {
          currency: "lkr",
          product_data: {
            name: item.name,
          },
          unit_amount: item.price * 100,
        },
        quantity: item.quantity,
      })),
      success_url: "http://localhost:5173/success",
      cancel_url: "http://localhost:5173/cancel",
    });

    res.json({ url: session.url });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
}
```

Figure 48 Code for Payment Gateway 2

## On-Going Project

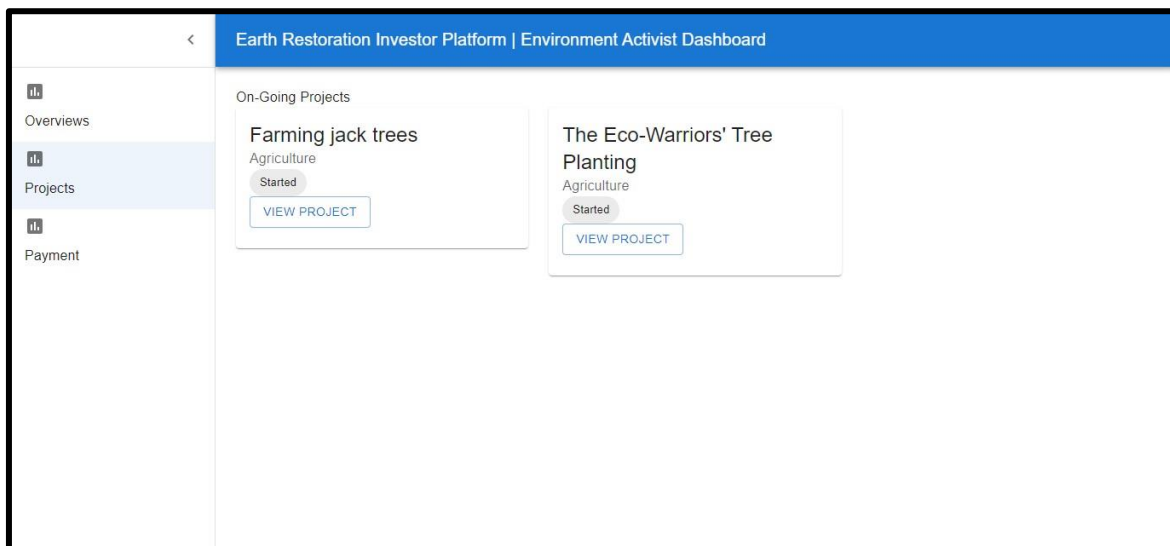


Figure 49 On-Going Project 1

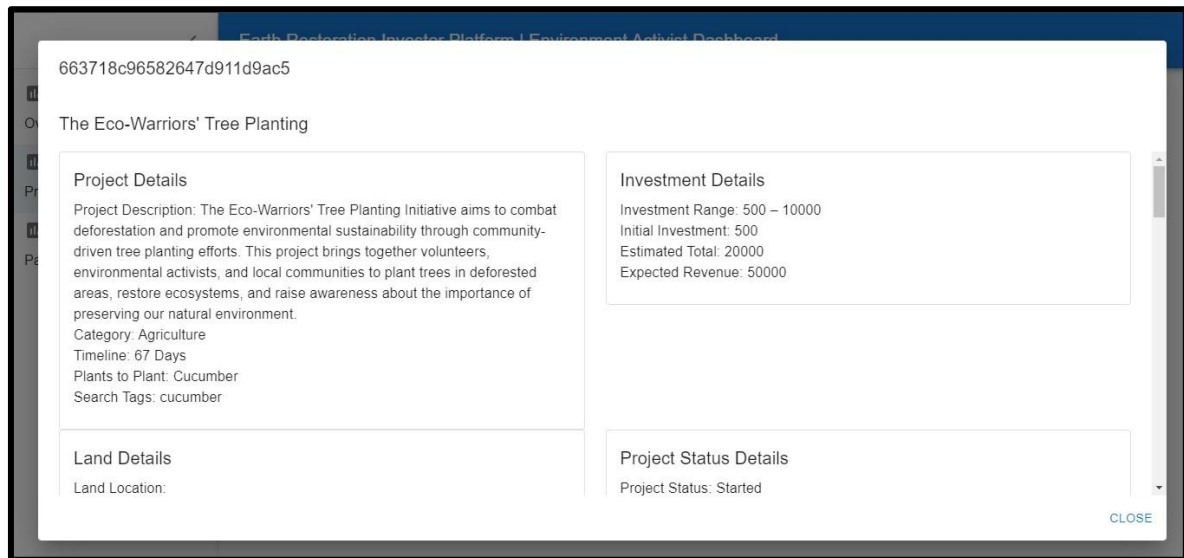


Figure 50 On-Going Project 2

In the ongoing project, it shows the Project Details, Investment Details, Land Details, and other project status details.

# Investor Payment Tab

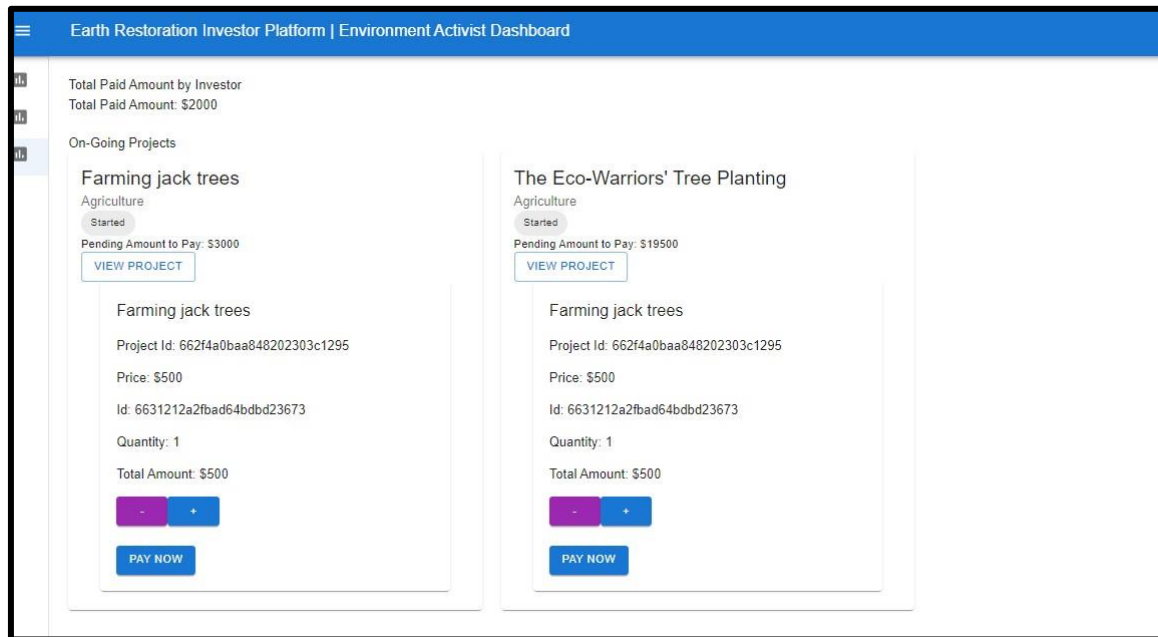


Figure 51 Investor Payment Tab

In the Investor Platform, investors can view the pending payment that needs to be paid. They can select the amount and pay securely using the Stripe payment gateway.

## Admin Dashboard

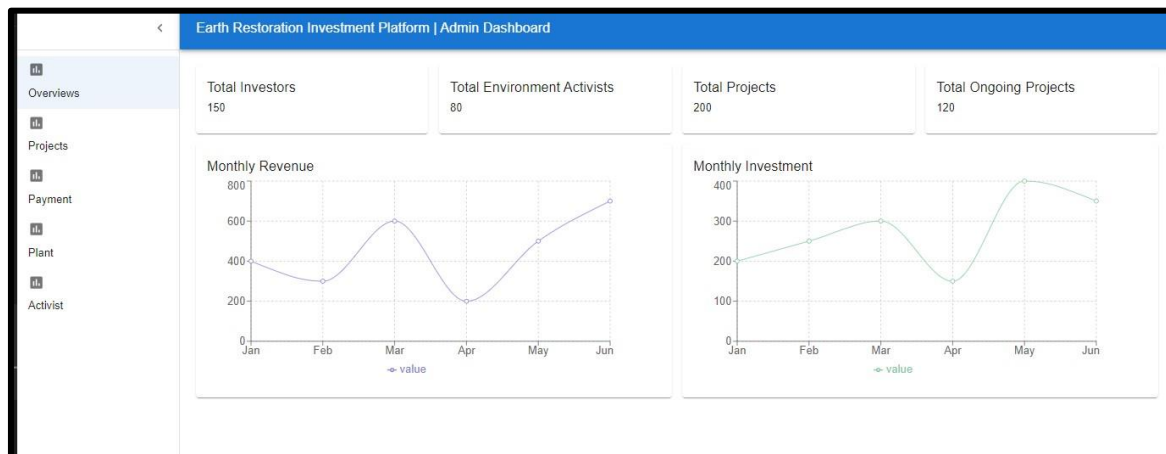


Figure 52 Admin Dashboard 1

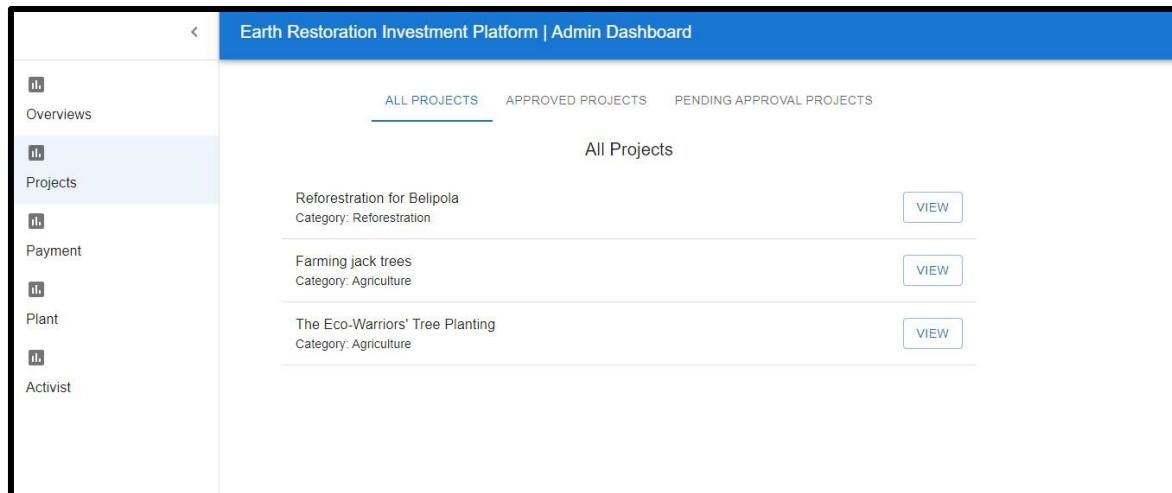


Figure 53 Admin Dashboard 2

In the Admin Dashboard, the admin can verify the land documents and update the approval status.

## All Project Details Admin

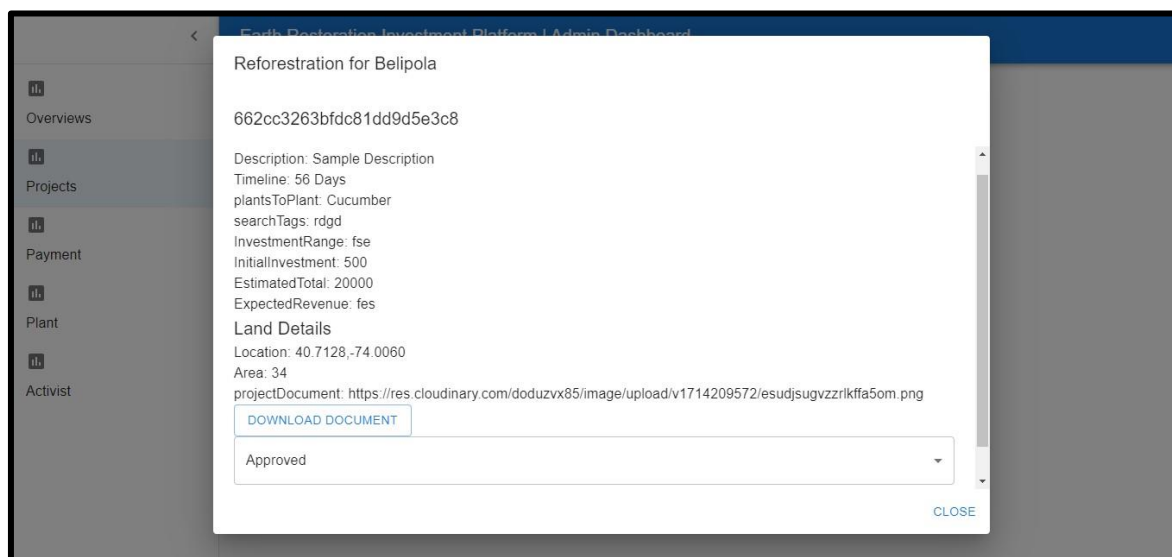


Figure 54 All Project Details Admin

# Pending Approval Projects

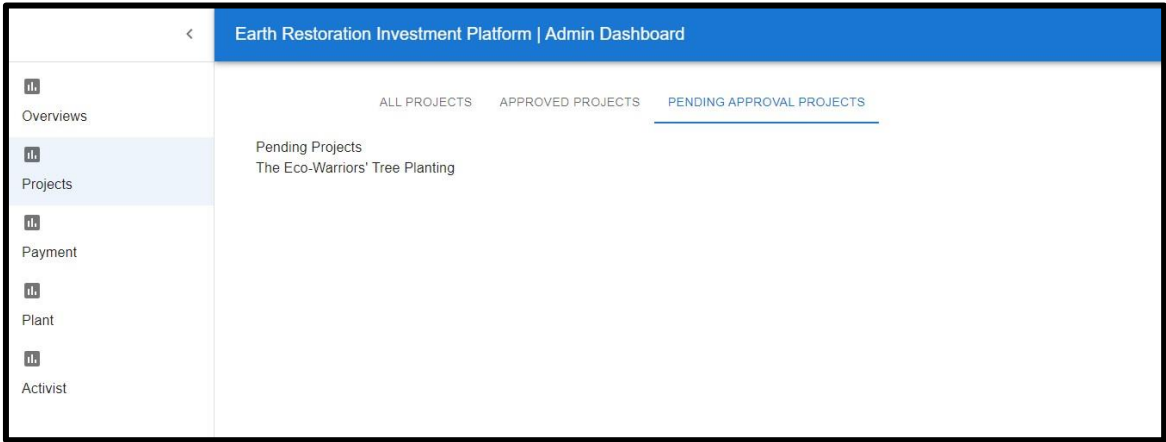


Figure 55 Pending Approval Projects

# Approved Projects

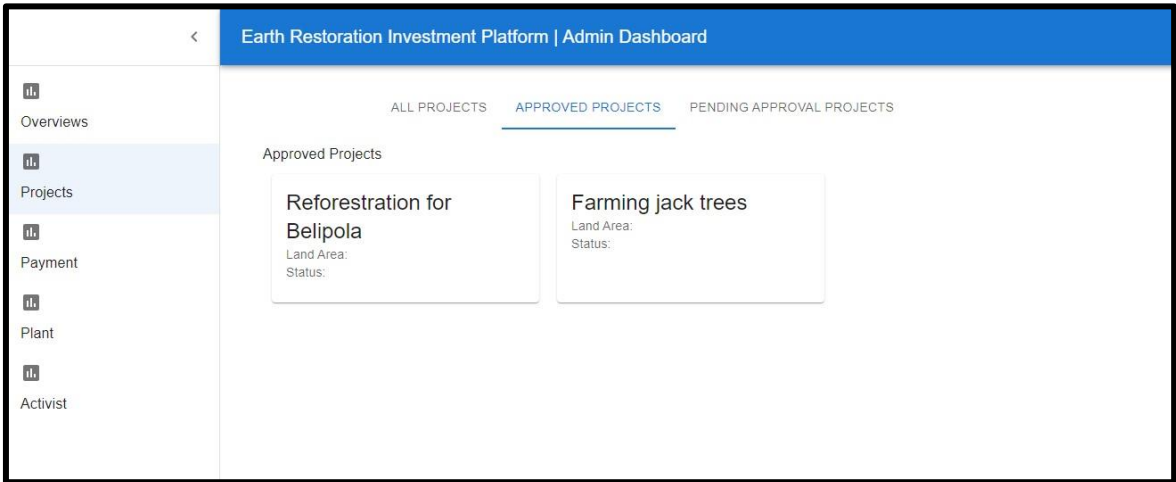


Figure 56 Approved Projects

(Intentionally blank)

## Payment



Figure 57 Payment 1



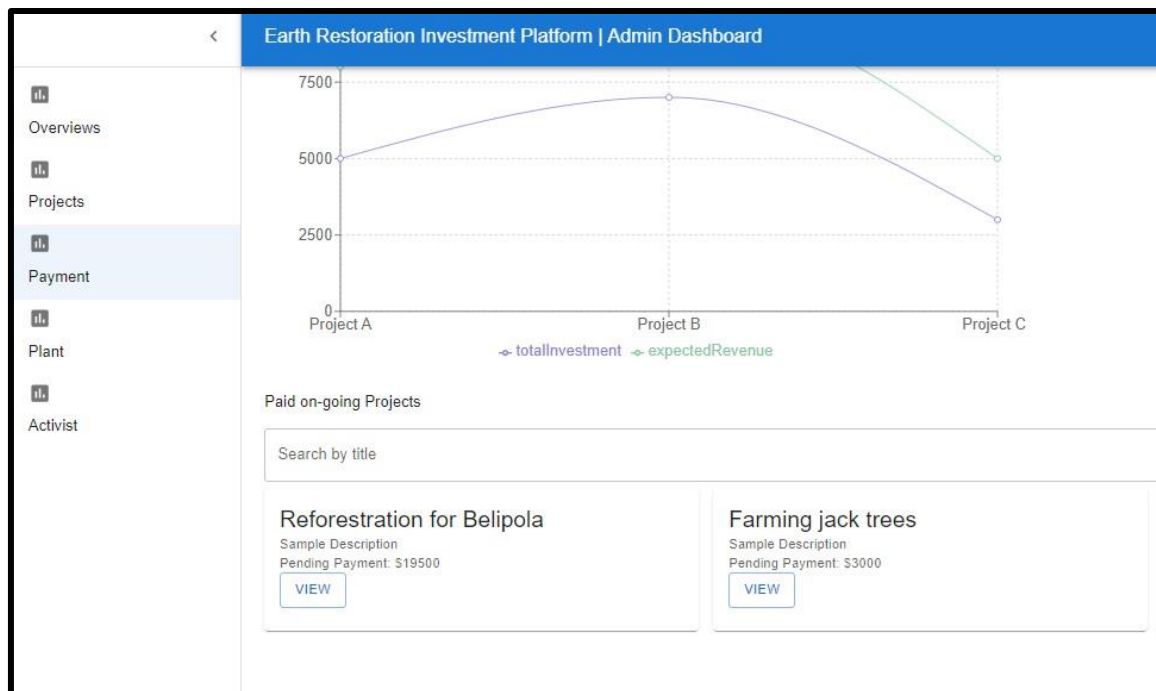


Figure 58 Payment 2

# Plant

The screenshot shows the 'Create Plant' modal form. The background is the 'Earth Restoration Investment Platform | Admin Dashboard' with a sidebar containing 'Overviews', 'Projects', 'Payment', 'Plant', and 'Activist'. The main content area has 'CREATE PLANT' and 'Plant Details' sections. The 'Plant Details' section lists 'Cucumber'. The 'Create Plant' modal is a white box with a title bar and a close button. It contains the following fields:

Create Plant	
Plant Name	Plant Description
Plant Species	Scientific Name
Temperature Range (Min)	Temperature Range (Max)
Humidity Range (Min)	Humidity Range (Max)
Growing Time Limit	Plants Per Square Meter
Market Rate Per Kg	Investment Per Square Meter
Suitable Locations	
<div>CANCEL CREATE PLANT</div>	

Figure 59 Plant create

The screenshot shows the 'Plant list' view. The background is the 'Earth Restoration Investment Platform | Admin Dashboard' with a sidebar containing 'Overviews', 'Projects', 'Payment', 'Plant', and 'Activist'. The main content area has 'CREATE PLANT RECORD' and 'Plant Details' sections. The 'Plant Details' section lists 'Cucumber'. The 'Plant list' view is a white box with a title bar and a close button. It contains the following fields:

Plant list	
<div>CREATE PLANT RECORD</div>	
Plant Details	
Cucumber	<div>VIEW</div>

Figure 60 Plant list

Where can the admin create plant records and display all the plant details. Using the view button, the admin can edit or delete the plant details.

(Intentionally blank)

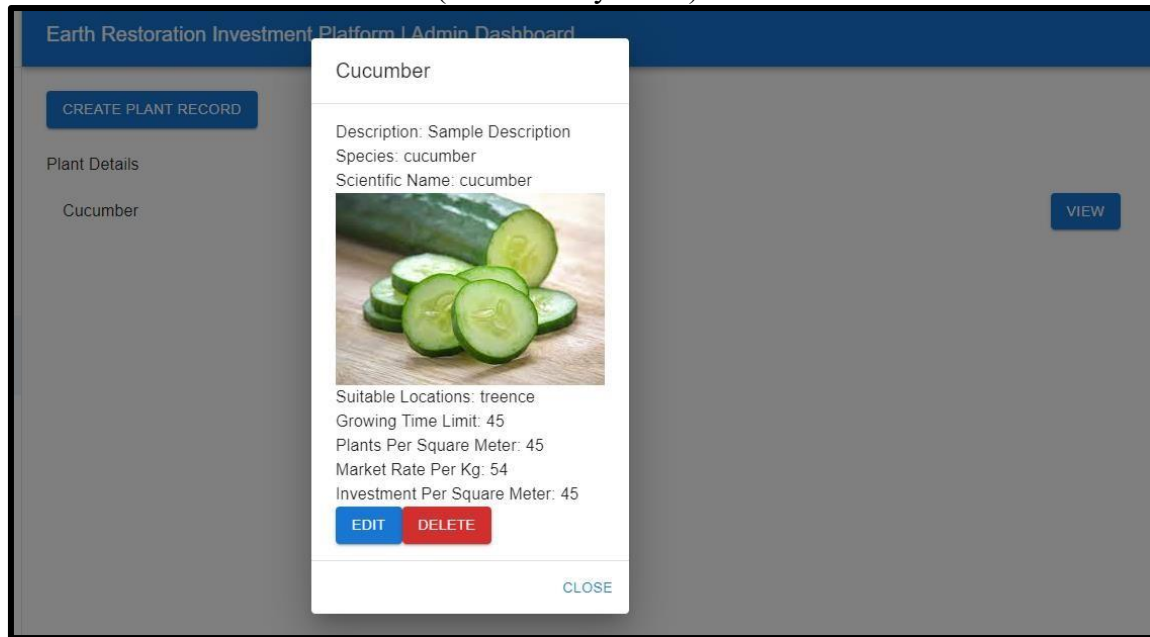


Figure 61 Plant management

## User Management

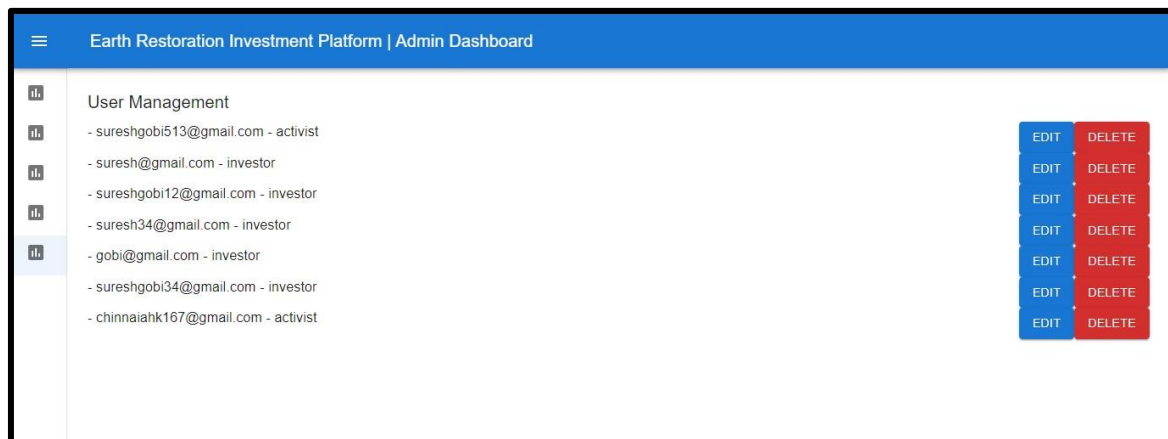


Figure 62 User Management

Admin can manage users of the ERIP system.