CHRIST
UNIVERSITY
BENGALURU, INDIA

Declared as Deemed to be University under Section 3 of UGC Act 1956

ECO HARBOUR

A Group Project Submitted for Undergraduate DBMS Lab

(BCA 481) 2023-2024

By

GAURAV JAIN (2241129)

GREESHMA GIRISH C (2241130)

NAVANEETH KISHORE C (224144)

Bachelor of Computer Application

Under the supervision of

Dr. SREEJA C.S

Project report submitted in partial fulfilment

Of the requirements of IV semester BCA, CHRIST (Deemed To Be University)

April - 2024

CHRIST UNIVERSITY
BENGALURU, INDIA
Declared as Deemed to be University under Section 3 of UGC Act 1956

CERTIFICATE

This is to certify that the report titled **Eco Harbour** is a bona fide record of work done by **Gaurav Jain(2241129), Greeshma Girish C(2241130) and Navaneeth Kishor (2241144)** of CHRIST (Deemed to be University), Bangalore, in partial fulfilment of the requirements of IV Semester **BCA** during the year 2023.

Head of the Department **Project Guide**

_____ _____

Valued-by:

Name: Gaurav Jain, Greeshma Girish C, Navaneeth Kishor C H

Register Number: 2241129, 2241130, 2241144

Examination Centre: CHRIST (Deemed to be University)

Date of Exam:

ACKNOWLEDGEMENT

First of all, we thank God almighty for his immense grace and blessings showered on us at every stage of this work. We are grateful to our respectable Head, Department of Computer Science, CHRIST (Deemed to be University), **Dr Ashok Immanuel V**, for providing the opportunity to take up this project as part of my curriculum.

We also pay our gratitude to the Coordinator, Department of Computer Science, CHRIST (Deemed to be University) **Dr Beaulah Soundarabai P** for their support throughout.

We are grateful to our guide, Assistant Professor, Department of Computer Science, CHRIST (Deemed to be University), **Dr Sreeja**, whose insightful leadership and knowledge benefited us to complete this project successfully. Thank you so much for your continuous support and presence whenever needed.

We would also like to thank our Alumni evaluator **NIL,** whose knowledge and guidance benefited us in making the changes as per the industry requirement. Thank you so much for your support and presence whenever needed.

We express our sincere thanks to all faculty members and staff of the Department of Computer Science, CHRIST (Deemed to be University), for their valuable suggestions during the course of this project. Their critical suggestions helped us to improve the project work.

Last but not the least; we would like to thank everyone who is involved in the project directly or indirectly.

Abstract

The Fishing Industry Sustainability Tracker stands as a pioneering database, meticulously crafted to oversee and regulate fishing practices with an unwavering commitment to sustainability. At its core lies a wealth of data, spanning across various dimensions such as catch records, regulatory measures, and metrics gauging sustainability. Through a dynamic fusion of real-time data acquisition, advanced analytical tools, and innovative visualization techniques, this tracker serves as a cornerstone for promoting informed decision-making within the fishing industry. Its overarching goal is to cultivate a culture of responsible stewardship, minimizing ecological footprints while maximizing the longevity of marine resources.

By harnessing the power of user-friendly interfaces, scalable infrastructure, and resilient data management protocols, this initiative endeavors to foster a collaborative ecosystem among stakeholders. Through seamless integration and accessibility, the tracker empowers policymakers, fisheries managers, researchers, and industry players to work in tandem towards shared sustainability objectives. Moreover, its capability for comprehensive compliance monitoring ensures that regulatory frameworks are upheld, while proactive measures can be swiftly implemented to address emerging challenges. Ultimately, the Fishing Industry Sustainability Tracker emerges as a beacon of hope for the future of our oceans, championing responsible fishing practices for the preservation and prosperity of marine ecosystems for generations to come.

EcoHarbour is an important step toward empowering fishermen and preserving sustainable fishing methods. The easily navigable website is made to accommodate the requirements of all types of fishermen, even those who are not tech-savvy. The user experience on the website is flawless and it is easy to browse. Fishermen can feel easy knowing that their personal and sailing information is protected on this secure website.

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1 Overview of the system

Existing fishing applications primarily concentrate on recommending fishing spots and logging catches, but they often lack accuracy and sustainability in data collection. This approach overlooks the critical need for responsible fishing practices aimed at preserving marine resources for the long term. Consequently, there is a pressing necessity for a comprehensive solution that addresses the shortcomings of current fishing applications and promotes sustainable fishing practices.

1.2 Functionalities of System

1.2.1 Abundance Detection: This is a digital guide for fishermen looking to explore the underwater world. It offers information on fish species that can be found in specific locations, giving fishermen an advantage by allowing them to plan their fishing expeditions based on the types of fish that are present. Whether they want to discover new locations or check the diversity of species in a particular region, this guide has you covered.

1.2.2 Education and Outreach: Beyond being a tool, this functionality is a platform for change. It's a beacon of education and awareness, promoting sustainable fishing practices. By collaborating with other fishermen and organizations dedicated to preserving marine life, it not only educates but also encourages users to be a part of the solution, fostering a community focused on preserving our oceans for future generations.

1.2.3 History and records: It provides all the previous and current catch records of a particular user and also visualizes it in graphical form. The module provides insights into fishes caught, trends over time, and comparisons between various vessels.

1.2.4 Reward System: Consider this functionality as a pat on the back for sustainable efforts. By incentivizing sustainable fishing practices, it encourages users to align with conservation goals. The rewards offered by the organization act as a motivation, recognizing and prioritizing users who contribute to maintaining a sustainable ecosystem.

2. SYSTEM ANALYSIS

2.1 Existing Systems

The idea of the fish tracker is not that well known but still some of the systems do exist in market that provides such services. The existing modules of our domain area focus on the achieving sustainability in a fishing industry and its practices. It gives emphasis to the empowering the fisherman.

The following are a few projects that the team has worked on improving (along with the hyperlink).

2.1.1 FishTrack

This mobile app, primarily targeting recreational fishing, uses GPS and environmental data to suggest potential fishing spots based on specific fish species and preferred bait. It also allows users to record catches and share locations with fellow anglers. (FishTrack)

2.1.2 FishAngler

Another popular app for recreational fishing, FishAngler helps users find fishing spots, share catches, and connect with other fishermen. It boasts a massive community and offers features like weather forecasts, tide charts, and lunar phases. (FishAngler)

2.1.3 GoFish

Focused on a broader audience, GoFish combines fishing location recommendations with a social media platform. Users can discover fishing spots, learn about different species, and share their experiences with a community of enthusiasts. (GoFish)

2.1.1 Limitation of Existing Systems

2.1.1.A Data accuracy and relevance

Many of these apps rely on user-reported data, which can be inaccurate or incomplete. This can lead to unreliable recommendations and frustration for other users. Also environmental factors, fishing techniques, and specific target species might not be adequately considered, leading to recommendations that don't match the individual user's needs.

2.1.1.B Sustainability focus

These apps primarily focus on maximizing catch rates, which can potentially contribute to overfishing and unsustainable practices. Majority apps present in this domain lack emphasis on responsible fishing methods, protected species awareness, and catch reporting limitations might hinder overall sustainability efforts.

2.2 Proposed System

The proposed system is a software application or platform designed to assist fishermen in tracking and managing their fishing activities more effectively. It would centralize data related to fishing trips, catches, equipment usage, and environmental conditions, providing fishermen with valuable insights to improve their practices and promote sustainability. The proposed system aims to streamline processes, enhance decision-making, and contribute to the conservation of marine ecosystems.

2.2.1 Benefits of Proposed System

2.2.1.A Contribution to Marine Conservation

By facilitating data-driven decision-making and promoting sustainable fishing practices, the project aligns with the goals of marine conservation and biodiversity preservation, contributing to the sustainable development goals and the livelihoods of future generations of fishermen.

2.2.1.BCompliance and Regulation Adherence

The Fishing Tracker system will assist fishermen in adhering to fishing regulations and quotas by providing alerts and notifications regarding legal limits, endangered species, and restricted zones, thereby promoting responsible fishing practices.

2.2.1.CReal-time Environmental Monitoring

Integrating real-time environmental data, such as water temperature, currents, and weather conditions, the system will offer fishermen valuable insights into optimal fishing times and locations while also promoting safety at sea.

2.3 Requirements Specifications

2.3.1 Functional Requirements

- Provides details of different fish species and their availability at specific locations.
- Users can view all their previous and current catch records.
- There is an option which provides some rewards to users who maintain a sustainable fishing practice.
- Users can share their experiences through community forums.

2.3.2 Technical Requirements

- **Performance**: To access the website the user must have secure internet connection
- **Data Security**: The data and information are stored in a secure manner through account passwords and ensures that there is no unauthorized access.
- **Data Storage:** The information is safely kept in a database so that only authorized program administrators have access to the database.

2.4 Software and Hardware Requirements

2.4.1 Software Requirements

OS: Windows 10 or above

Front End: HTML, CSS / React Js, Java Script

Back End: Node Js / Express Js

Database Required: MongoDB Atlas

2.4.2 Hardware Requirements

OS: Windows 10 or above

Processor: Core 2 Duo

Memory: 4GB RAM

Storage: 1 GB

2.4.3 Network Requirements

Network: Google Chrome

Wi-Fi: Internet Required

3. SYSTEM DESIGN

3.1 Block Diagram

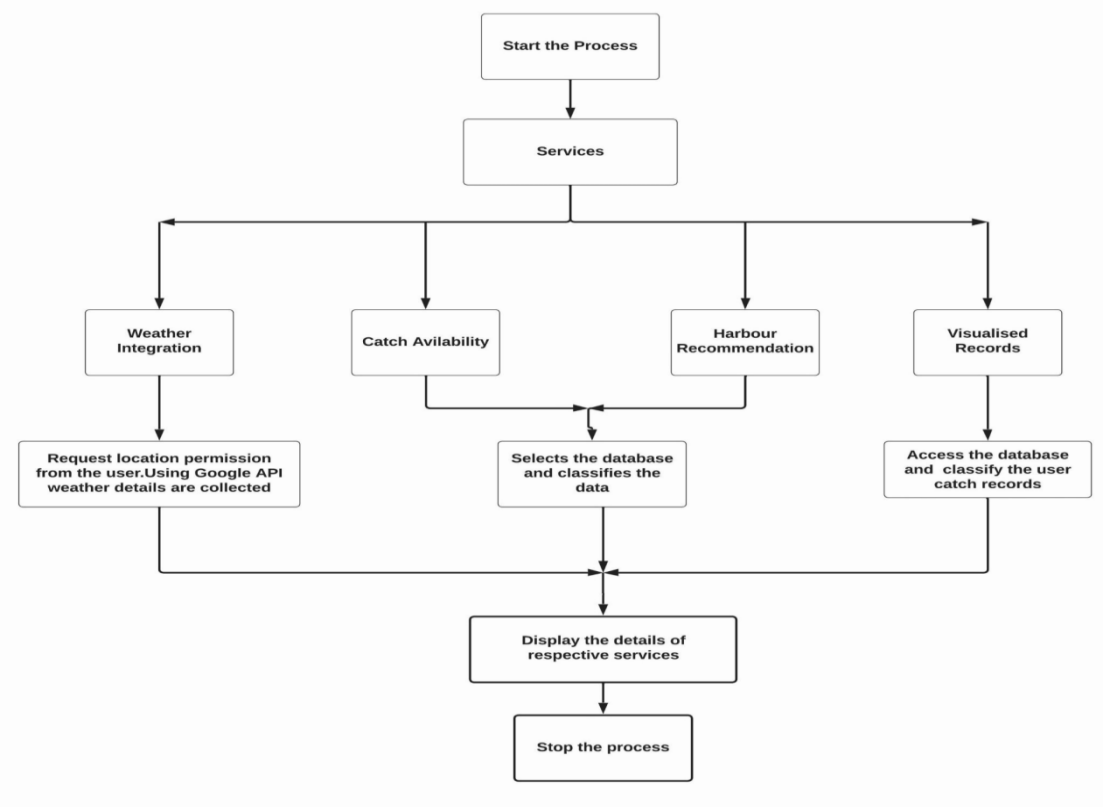

Fig 3.1 Block Diagram showing graphical representation of the system

3.2 Database Design

3.2.1 Database Tables

User Table

Attributes	Data Types
_id	Object()
first_name	String
user_name	String
password	String
email	String
contact no	Number
status	String
score	Number
isAdmin	Boolean

Table No. 1 User Table Data Dictionary

Fish Table

Attributes	Data Types
_id	Object()
local_name	String
scientific_name	String
seasonal availability	String
catch_limit	String
category	String
image	String

Table No. 2 Fish Table Data Dictionary

Harbour Table

Attributes	Data Types
_id	Object()
name	String
latitude	String
longitude	String
location	String
district	String
image	String
rating	String
description	String

Table No. 3 Harbour Table Data Dictionary

MRA Table

Attributes	Data Types
_id	Object()
name	String
latitude	Float
longitude	Float
area	Float
state	String

Table No. 4 MRA Table Data Dictionary

Abundances Table

Attributes	Data Types
_id	Object()
fish_id	Object()
state	String
abundance	String

Table No. 5 Abundances Table Data Dictionary

Catch Record Table

Attributes	Data Types
_id	Object()
user_id	Object()
longitude	Float
latitude	Float
image	String
status	String
admin_id	Object()

Table No. 6 Catch Record Table Data Dictionary

Feedback Table

Attributes	Data Types
_id	Object()
name	String
email	String
subject	String
feedback	String

Table No. 7 Feedback Table Data Dictionary

Reward Table

Attributes	Data Types
_id	Object()
name	String

description	String
points	Number
claimed	Boolean
claimed_user	Object()
image	String

Table No. 8 Reward Table Data Dictionary

Forum Table

Attributes	Data Types
_id	Object()
user	Object()
title	String
content	String
category	String
likes	Array
views	Array

Table No. 9 Forum Table Data Dictionary

3.3 ER Diagram

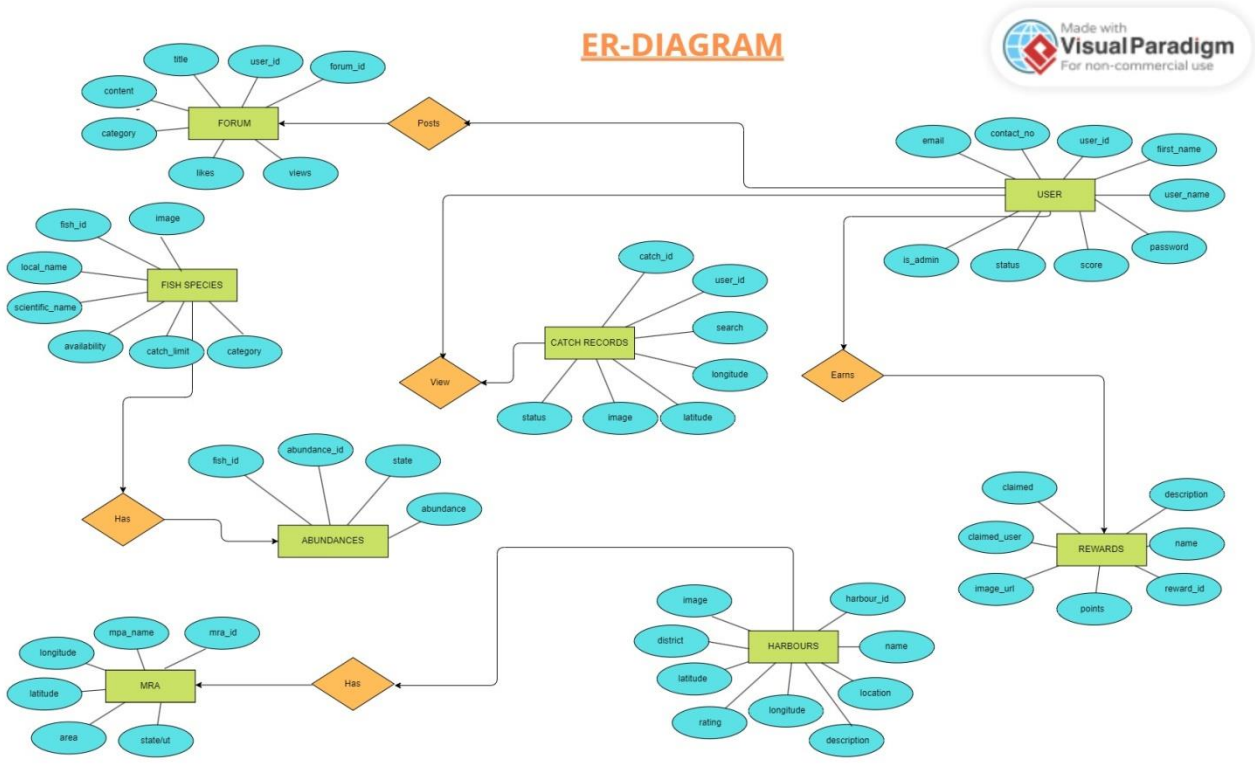

Fig 3.2 Entity Relationship Diagram of Eco-Harbour

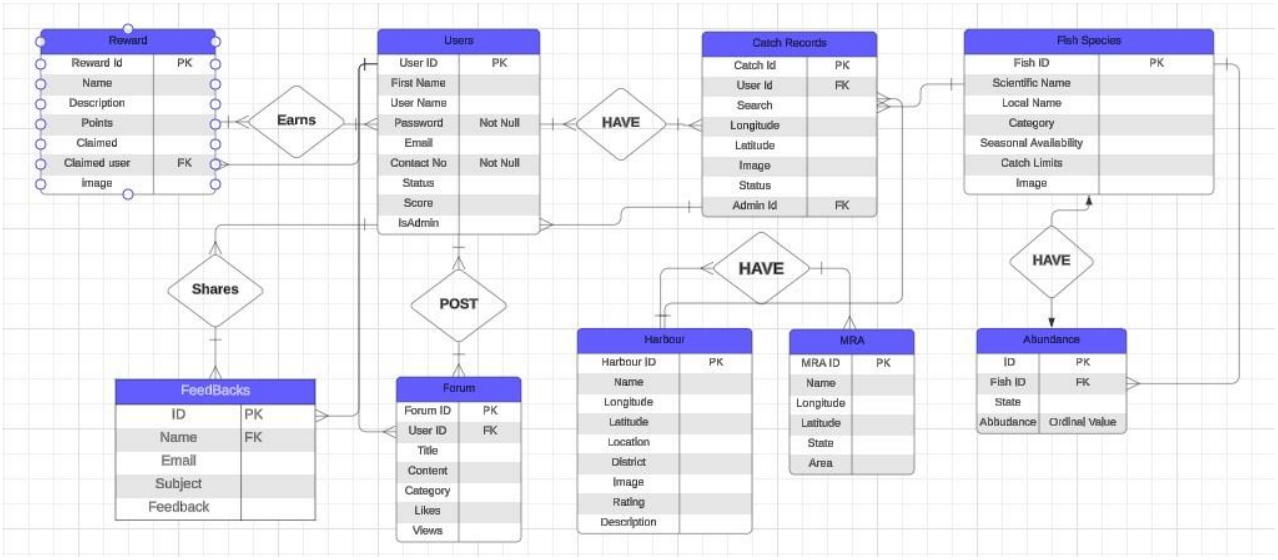

Fig 3.3 Class Diagram of Eco-Harbour

3.4 Data Flow Diagram

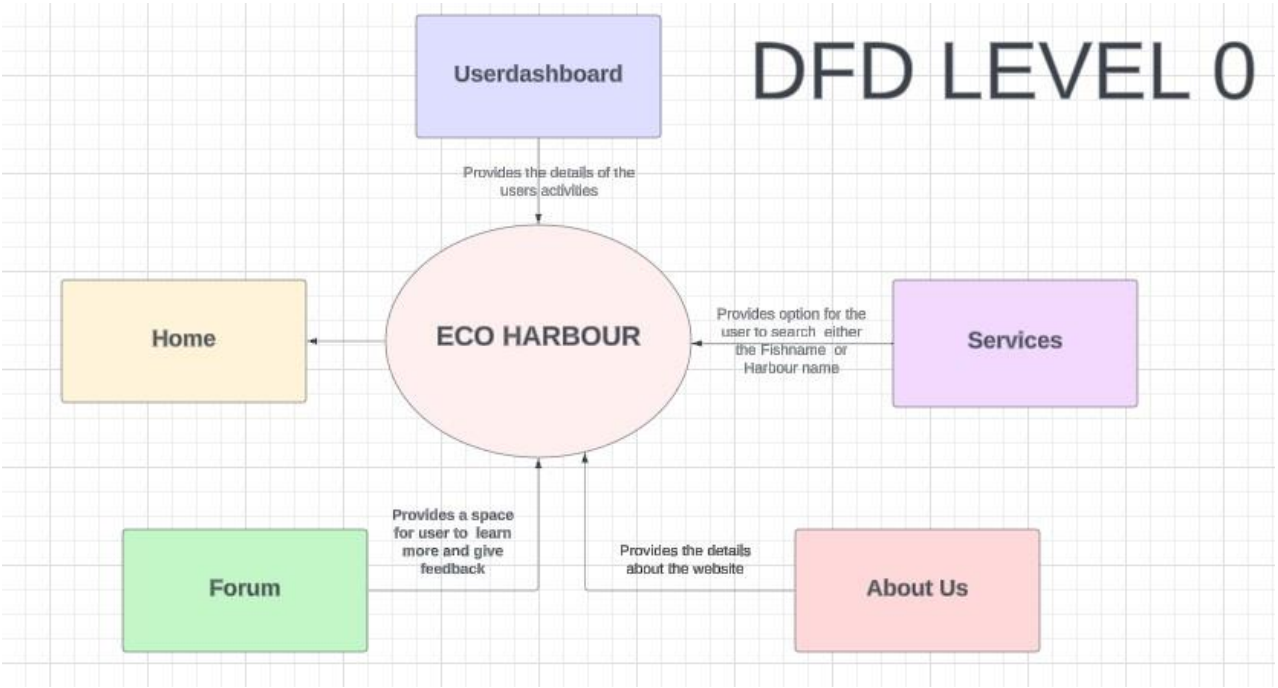

Fig 3.4 Data Flow Diagram Level 0 depicting overview of the entire system

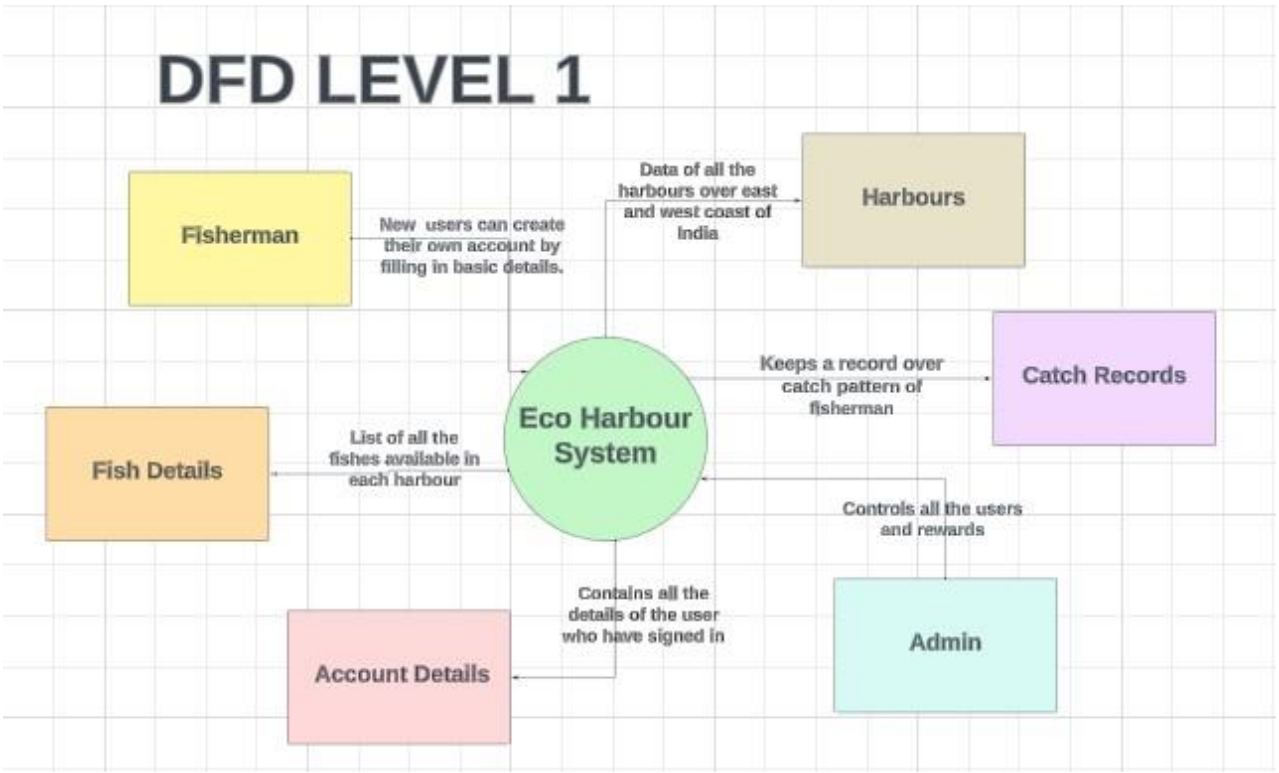

Fig 3.5 Data Flow Diagram Level 1 showing main sub-processes of Eco-harbour system

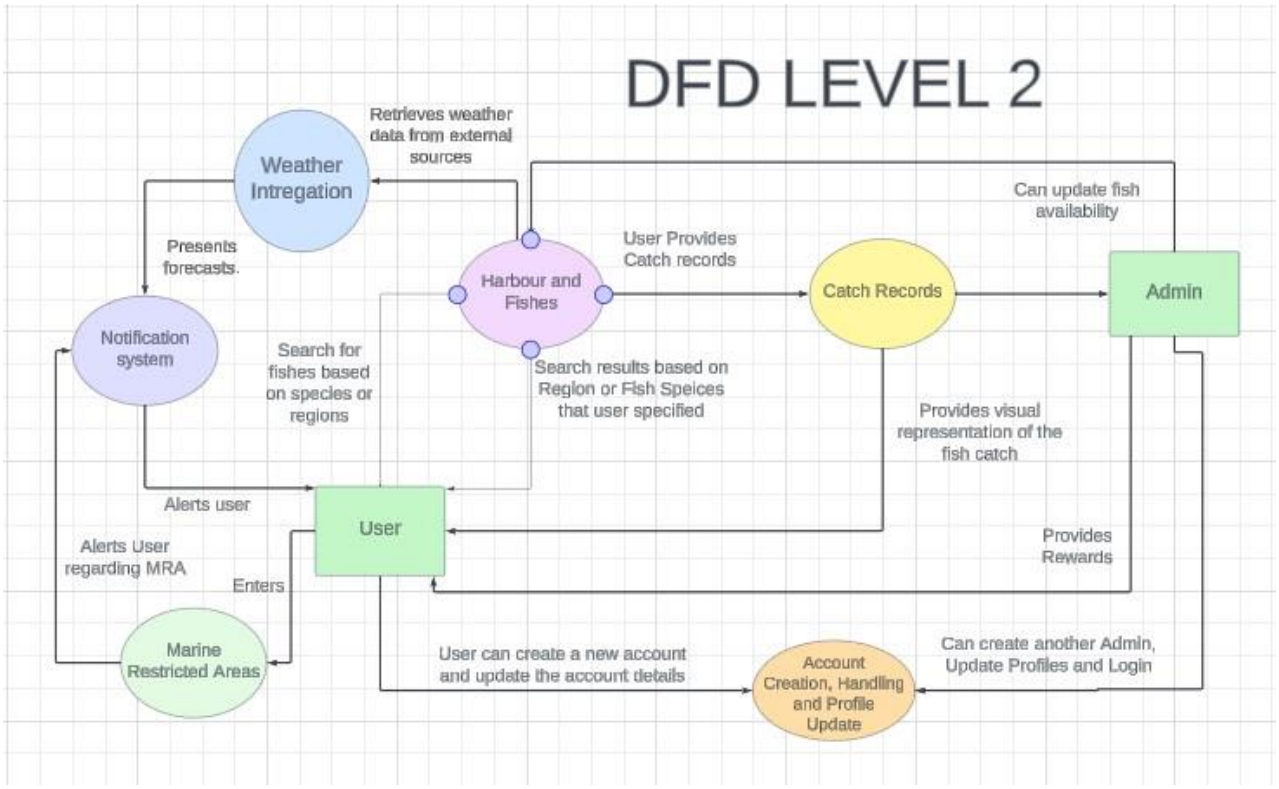

Fig 3.6 Data Flow Diagram Level 2 showing each sub-process as a separate process

3.5 User Interface Design

3.5.1 Use Case Diagram

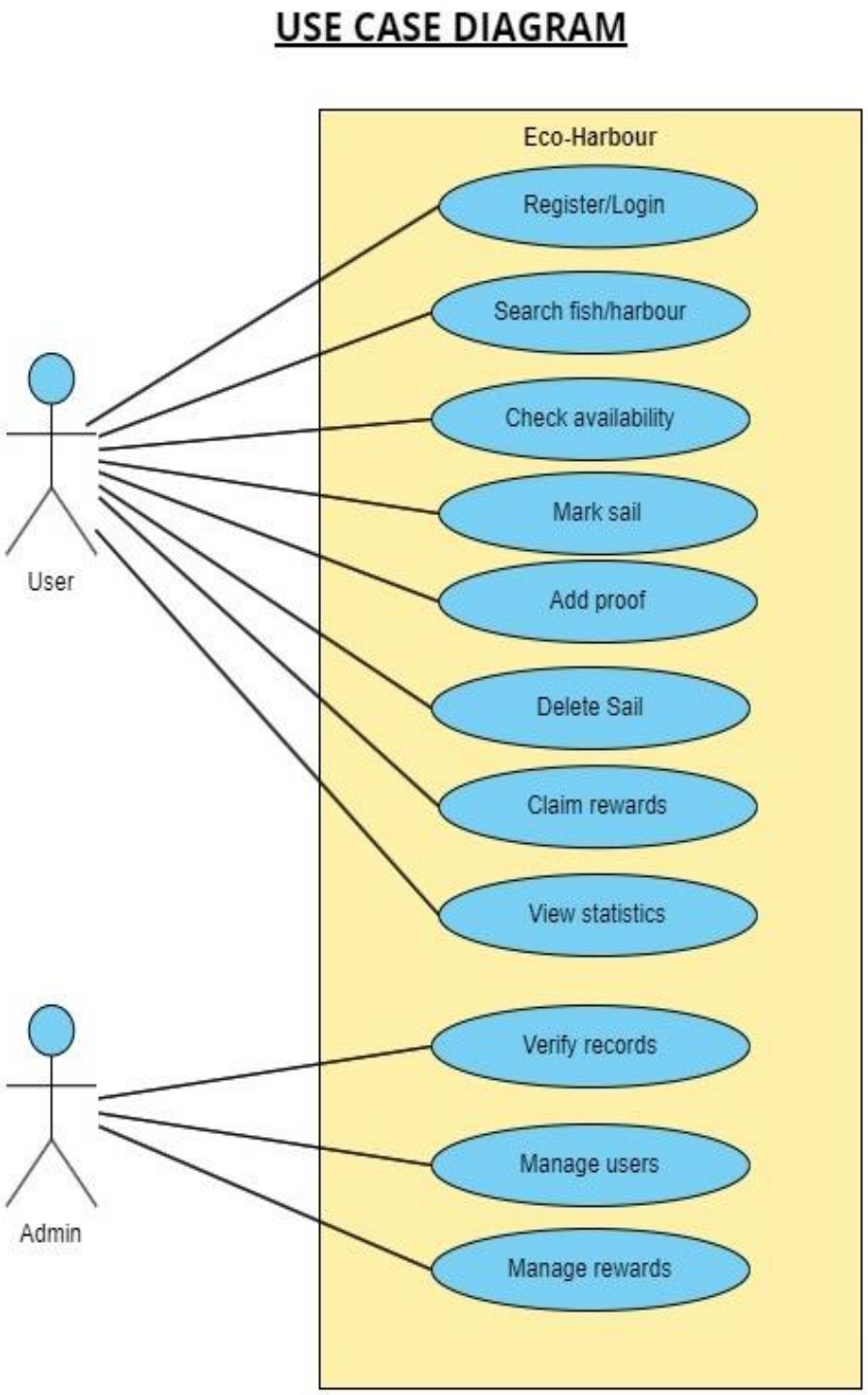

Fig 3.7 Use case Diagram for Users and Admin

4 IMPLEMENTATION

4.1Source Code

Login Page:

```
import React, { useState } from 'react'
import { Link, useNavigate } from 'react-router-dom';
import {FaEye, FaEyeSlash, FaUser} from 'react-icons/fa';
import axios from 'axios'
import { toast } from 'react-toastify'
import 'react-toastify/dist/ReactToastify.css'

const Login = ({ login, tooglelogin }) => {
    const [user_name, setUserName] = useState('');
    const [password, setPassword] = useState('');
    const [eye, setEye] = useState(false);
    const [userType, setUserType] = useState()

    const navigate = useNavigate()

    const handleSubmit = async (e) => {
        e.preventDefault()

        if(userType === "User"){
            try{
                const response = await axios.post('user/auth', {
                    user_name,
                    password
                })
                toast.success('Login Successful !')
                navigate('/')
            } catch (error){
                if(error.response.data.message === 'Invalid Email or Password'){
                    toast.error("Invalid Email or Password")
                }
                else {
                    toast.error("Login Failed")
                    console.log("Login Failed", error)
                }
            }
        }
        else if (userType === "Admin") {
            try{
```

```
            const response = await axios.post('user/authAdmin', {
                user_name,
                password
            })
            console.log(response)
            toast.success('Login Successful !')
            navigate('/admin')
        } catch (error){
            if(error.response.data.message === 'Invalid Email or Password'){
                toast.error("Invalid Email or Password")
                console.log(error)
            }
            else if(error.response.data.message === 'Unauthorized Admin'){
                toast.error("Unauthorized Admin")
            }
            else {
                toast.error("Login Failed")
                console.log("Login Failed", error)
            }
        }
    }
    else {
        toast.warning("Please Select a User Type")
    }
}

if(login){
    return null
}

return (
    <div id='login-wrapper'>
        <div id='login'>
            <div className='wrapper'>
                <img src='/images/login.jpg' alt='login'/>
            </div>
            <div style={{width: '45%', height: '100%'}}>
                <form onSubmit={handleSubmit} name="login">
                    <h1>Login</h1>
                    <div className='input'>
                        <p>Login as : </p>
                        <input type="radio" name="userType" value="User" onChange={(e) =>
setUserType(e.target.value)}/><p>User</p>
```

```
                    <input type="radio" name="userType" value="Admin" onChange={(e) =>
setUserType(e.target.value)}/><p>Admin</p>
                </div>
                <div className="input">
                    <input type="text" placeholder="Username" onChange={(e) =>
setUserName(e.target.value)} value={user_name}/>
                    <i id="user"><FaUser/></i>
                </div>
                <div className="input">
                    <input type={eye ? "text" : "password"} placeholder="Password" onChange={(e)
=> setPassword(e.target.value)} value={password}/>
                    <i id="eye" onClick={() => setEye(!eye)}>{eye ? <FaEyeSlash/>:
<FaEye/>}</i>
                </div>
                <div className="remember">
                    <a href="#">Forgot Password ?</a>
                </div>
                <button type="submit" className="btn">Login</button>
                <div className="register">
                    <p>Dont have an account ?<Link to="/regitration">Create One</Link></p>
                </div>
            </form>
        </div>
    </div>
  </div>
 )
}

export default Login
```

Registration Page:

```
import React, { useState } from 'react'
import { Link, useNavigate } from 'react-router-dom'
import { toast } from 'react-toastify'
import axios from 'axios'
import { FaEye, FaEyeSlash } from 'react-icons/fa'

import 'react-toastify/dist/ReactToastify.css'

const Register = () => {
   const [password, setPassword] = useState(false)
```

```
const validateName = (name) => {
    const nameRegex = /^[a-zA-Z'-]{2,30}( [a-zA-Z'-]{2,30})?$/;
    if(nameRegex.test(name)){
        return true
    }
    return false
}

const validateEmail = (email) => {
    const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
    if(emailRegex.test(email)){
        return true
    }
    return false
}

const validatePassword = (password) => {
    return password.length >= 5
}

const navigate = useNavigate()

const handelSubmit = async (e) => {
    e.preventDefault()

    const user_name = document.forms["register"].name.value;
    const contact_no = document.forms["register"].phone.value;
    const password = document.forms["register"].password.value;
    const cpassword = document.forms["register"].cpassword.value;

    if(user_name && contact_no && password && cpassword && (password === cpassword)){
        const user = {
            user_name,
            contact_no,
            password
        }

        try {
            const { data } = await axios.post('/user/', user)

            if(data._id){
                toast.success("Registration successful")
                navigate('/login')
            }
```

```
            else{
                toast.success("Registration Failed")
                console.log(data)
            }

        } catch (error) {

            if(error.response.data.message === 'Mobile No. already Exist'){
                toast.error("Mobile No. Already Exist")
            }
            else if (error.response.data.message === 'Invalid User Data'){
                toast.error("Invalid Data")
            }
            else{
                toast.error("Registration Failed")
                console.log(error)
            }

        }
    }
    else{
        toast.error("Please complete all fields.")
    }
}



return (
  <>
    <div className='registration'>
        <img src='/images/registration.jpg' alt='registration page'></img>
    </div>
    <div className="container">
        <div className='reg'>
            <h1>Join us Today</h1>
            <form name="register">
                <div id='form-content'>
                    <div id='form-label'>
                        <label>First Name</label>
                        <label>Mobile No.</label>
                        <label>Password</label>
                        <label>Confirm Password</label>
                    </div>
```

```
                <div id='form-input'>
                    <div>
                        <input type="text" placeholder="Enter your Name" required name='name'/>
                    </div>
                    <div>
                        <input type="text" placeholder="Enter your Mobile No." required
name='phone'/>
                    </div>
                    <div style={{display: 'flex'}}>
                        <input type={password ? 'text' : 'password'} placeholder="Enter your
Password" required name='password'/>
                        <i id='eye-reg' onClick={() => setPassword(!password)}>{password ?
<FaEyeSlash/> : <FaEye/> }</i>
                    </div>
                    <div>
                        <input type="password" placeholder="Confirm your Password" required
name='cpassword'/>
                    </div>
                </div>
            </div>
            <button type="submit" className="btn" style={{width: '120px', margin: '20px'}}
onClick={handelSubmit}>
                Register
            </button>
        </form>
        <div className="register">
            <p>Already have an account ?<Link to="/login">Login</Link></p>
        </div>
    </div>
    <div className='wrapper'>
        <img src='/images/registration.jpg' alt='registration'></img>
    </div>
  </div>
 </>
 )
}

export default Register
```

Home Page NavBar:

```
import React, { useState, useEffect } from 'react'
import { Link, useLocation } from 'react-router-dom'
import { FaUser } from 'react-icons/fa'
import axios from 'axios'

const Navbar = ({login, tooglelogin}) => {
  const [loggedIn, setLoggedIn] = useState(false)
  const [user, setUser] = useState({})
  const location = useLocation()

  const checkLogin = async () => {
    try {
      const { data } = await axios.get('/user/profile')

      if(data._id){
        setLoggedIn(true)
        setUser(data)
      }
    } catch (error) {
      if (error.response && error.response.status === 404) {
        console.log("No user found");
      } else if (error.message === "Request failed with status code 401") {
        console.log("User not logged in");
      } else {
        console.error("Error fetching user profile:", error.message);
      }
    }
  }

  useEffect(() => {
    checkLogin()
  }, [loggedIn])

  return (
      <header>
        <div>
          <Link to='/'><img className='logo' src="/images/logo2.png" alt='logo'></img></Link>
        </div>
        <div id='nav'>
          <ul>
            <li><strong><Link to='/' className={location.pathname === '/' ? "active" :
"}>Home</Link></strong></li>
```

```
          <li><strong><Link to='/services' className={location.pathname === '/services' ? 'active' :
''}>Services</Link></strong></li>
          <li><strong><Link to='/forum' className={location.pathname === '/forum' ? 'active' :
''}>Forum</Link></strong></li>
          <li><strong><Link to='/about'className={location.pathname === '/about' ? 'active' :
''}>Contact Us</Link></strong></li>
          <li>
            <Link to={loggedIn ? '/dashboard' : '/login'}>
              {loggedIn ? `${user.first_name.slice(0, 1).toUpperCase()}` : <FaUser
onClick={toogglelogin}/>}
            </Link>
          </li>
        </ul>
      </div>
    </header>
  )
}

export const AdminNavbar = () => {
  const [loggedIn, setLoggedIn] = useState(false)
  const [user, setUser] = useState({})
  const location = useLocation()

  const checkAdminLogin = async () => {
    try {
      const { data } = await axios.get('/user/authAdmin')
      console.log(data)

      if(data._id){
        setLoggedIn(true)
        setUser(data)
      }
    } catch (error) {
      if (error.response && error.response.status === 404) {
        console.log("No admin found");
      } else if (error.message === "Request failed with status code 401") {
        console.log("Admin not logged in", error);
      } else {
        console.error("Error fetching admin profile:", error.message);
      }
    }
  }

  useEffect(() => {
```

```
    checkAdminLogin()
  }, [loggedIn])

  return (
     <header>
       <div>
         <Link to='/'><img className='logo' src="/images/logo2.png" alt='logo'></img></Link>
       </div>
       <div id='nav'>
         <ul>
           <li><strong><Link to='/' className={location.pathname === '/' ? "active" :
"}>Home</Link></strong></li>
           <li><strong><Link to='/services' className={location.pathname === '/services' ? 'active' :
"}>Users</Link></strong></li>
           <li><strong><Link to='/forum' className={location.pathname === '/forum' ? 'active' :
"}>Rewards</Link></strong></li>
           <li><strong><Link to='/about'className={location.pathname === '/about' ? 'active' :
"}>Contact Us</Link></strong></li>
           <li>
             <Link to={loggedIn ? '/dashboard' : '/login'}>
               {loggedIn ? `${user.first_name.slice(0, 1).toUpperCase()}` : <FaUser/>}
             </Link>
           </li>
         </ul>
       </div>
     </header>
  )
}

export default Navbar
```

Admin Dashboard

```
import React, { useEffect, useState } from 'react'
import Spinner from './Spinner'
import { Link, useNavigate } from 'react-router-dom'
import { FaUser, FaGift, FaCheckCircle, FaBan, FaImage, FaSearch } from 'react-icons/fa'
import { IoIosCloseCircle } from "react-icons/io";
import { MdLogout, MdDashboard  } from "react-icons/md";
import { GrDocumentVerified } from "react-icons/gr";
import axios from 'axios';
import { toast } from 'react-toastify';
```

```
const ImageViewer = ({ imageUrl, onClose }) => {
    console.log(imageUrl)
    return (
        <div className="image-viewer">
            <IoIosCloseCircle onClick={onClose}/>
            <img src={imageUrl} alt="Proof" />
        </div>
    );
};
```

```
const Admin = () => {
    const [activeTab, setActiveTab] = useState('dashboard')
    const [admin, setAdmin] = useState({})
    const [userCount, setUserCount] = useState()
    const [users, setUsers] = useState({})
    const [proof, setProof] = useState()
    const [record, setRecord] = useState([])
    const [imageViewerOpen, setImageViewerOpen] = useState(false);
    const [selectedImage, setSelectedImage] = useState(null);
    const [userName, setUserName] = useState('')

    const navigate = useNavigate()

return (
        <>
            <section id='admin'>
                <div className="admin-sidebar">
                    <div>
                        <Link to='#dashboard'><img className='logo' src="/images/logo.png" alt='logo'
style={{margin: "0px 30px", borderTop: "none"}}></img></Link>
                    </div>
                    <div>
                    <ul className="admin-nav">
                        <li className={activeTab === 'dashboard' ? 'active' : ''}>
                            <Link onClick={() => toogleTab('dashboard')}>
                                <MdDashboard/>
                                DashBoard
                            </Link>
                        </li>
                        <li className={activeTab === 'verify' ? 'active' : ''}>
                            <Link onClick={() => toogleTab('verify')}>
                                <FaCheckCircle />
                                Verify
```

```
                    </Link>
                </li>
                <li className={activeTab === 'users' ? 'active' : ''}>
                    <Link onClick={() => toogleTab('users')}>
                        <FaUser/>
                        Manage Users
                    </Link>
                </li>
                <li className={activeTab === 'rewards' ? 'active' : ''}>
                    <Link onClick={() => toogleTab('rewards')}>
                        <FaGift/>
                        Rewards
                    </Link>
                </li>
            </ul>
            </div>
            <div id='logout'>
                <button className='btn' onClick={handelClick}>
                    <MdLogout/>
                    Logout
                </button>
            </div>
        </div>
    </section>

    <div className="admin-content" id="dashboardContent" style={{ display: activeTab === 'dashboard' ? 'flex' : 'none' }}>
        {admin ? (
            <>
            <h1>Admin DashBoard </h1>
            <div className='dashBoard-card'>
                <div className='admin-box' style={{backgroundColor: '#5454ffb5'}}>
                    <div className='box-icon'>
                        <GrDocumentVerified/>
                    </div>
                    <div className='box-content'>
                        <h2> {proof} </h2>
                        <p>Proof Verified</p>
                    </div>
                </div>
                <div className='admin-box' style={{backgroundColor: '#ff4040c9'}}>
                    <div className='box-icon'>
                        <FaCheckCircle/>
                    </div>
```

```
                    <div className='box-content'>
                        <h2 style={{padding: '0px', fontSize: '50px'}}>  Admin </h2>
                        <p> Status </p>
                    </div>
                </div>
                <div className='admin-box' style={{backgroundColor: '#f351e4c7'}}>
                    <div className='box-icon'>
                        <FaUser/>
                    </div>
                    <div className='box-content'>
                        <h2> {userCount} </h2>
                        <p> User Count </p>
                    </div>
                </div>
            </div>
        </>
        ) : (
        <Spinner/>
        )}
      </div>

      {imageViewerOpen && selectedImage && (
        <ImageViewer imageUrl={selectedImage} onClose={closeImageViewer} />
      )}

      <div className="verify-document" id="verify" style={{ display: activeTab === 'verify' ?
'flex' : 'none' }}>
        <h1>Verify Record Documents</h1>
        <div id='sails'>
          {record.length > 0 ? (
            <>
            <h2>Recent Records</h2>
            <table>
              <thead>
              <tr>
                <th>S.No</th>
                <th>Searched</th>
                <th>Upload Date</th>
                <th>Proof</th>
                <th>Action</th>
              </tr>
              </thead>
              <tbody>
              {record.map((r, index) => {
```

```
                        return (
                        <tr key={r._id}>
                            <td> {index + 1} </td>
                            <td> {r.search} </td>
                            <td> {r.updatedAt.slice(0, 10)} </td>
                            <td>
                                {r.image !== "" ? (
                                    <button className='btn view' onClick={() =>
viewProof(r._id)}>View <FaImage/></button>
                                ) : (
                                    <p>Proof Not Uploaded</p>
                                )}
                            </td>
                            <td>
                                <button className="btn approve" onClick={() =>
rewardSail(r._id)}>Reward <FaCheckCircle/> </button>
                                <button className="btn reject" onClick={() => rejectSail(r._id)}>Reject
<FaBan/> </button>
                            </td>
                        </tr>
                        )
                    })}
                    </tbody>
                </table>
                </>
            ) : (
                <p>No Records Uploaded Recently </p>
            )}
        </div>
    </div>

    <div className='manage-users' style={{display : activeTab === "users" ? 'flex' : 'none'}}>
        <h1> Manage Users</h1>
        <div className='actions'>
            <input type="text" className='search' placeholder='Search User' id="search"
onChange={(e) => setUserName(e.target.value)}/>
            <button type='button' className='btn search-btn' onClick={handelSearch}>Search
<FaSearch/></button>
            <button type='button' className='btn suspend-
btn'onClick={handelSuspend}>Suspend Account</button>
        </div>
        <div id="users">
            {users.length > 0 ? (
                <>
```

```
                <table>
                  <thead>
                  <tr>
                    <th>Select</th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>Contact No</th>
                    <th>Approved</th>
                    <th>Rejected</th>
                  </tr>
                  </thead>
                  <tbody>
                  {users.map((user) => {
                    return (
                    <tr key={user._id}>
                      <td><input className='suspend' type="checkbox" data-user-id={user._id}
onChange={() => handleSelect (user._id)}/></td>
                      <td> {user.user_name} </td>
                      <td> {user.email ? user.email : " - "} </td>
                      <td> {user.contact_no}</td>
                      <td> {user.approvedSails} </td>
                      <td> {user.rejectedSails} </td>
                    </tr>
                    )
                  })}
                  </tbody>
                </table>
                </>
              ) : (
                <p>No User Found</p>
              )}
            </div>
          </div>
        </>
    )
}

export default Admin
```

4.2 Screenshots

4.2.1 User Interface Design

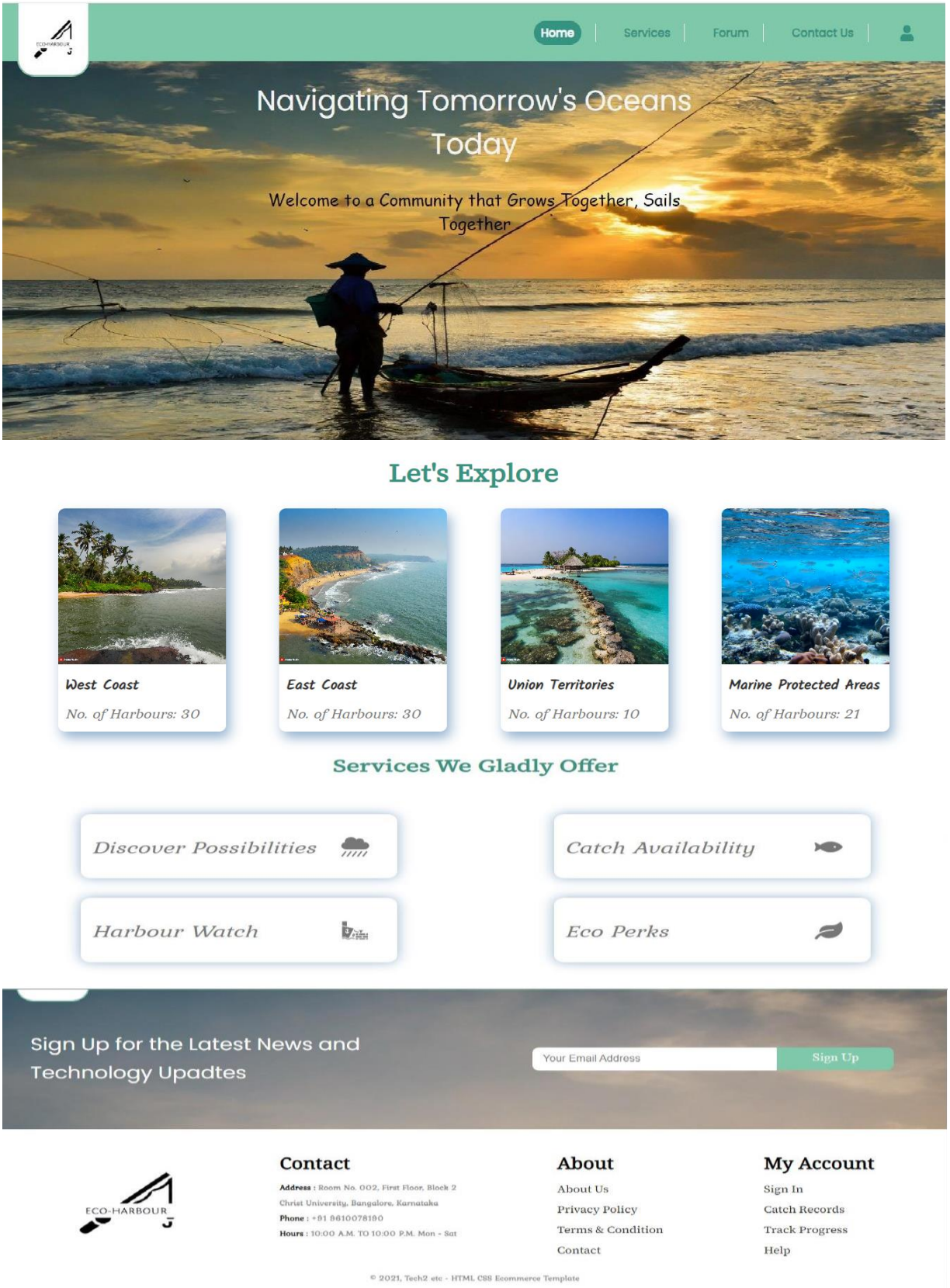

Fig 3.8 Home Page of EcoHarbour

Service Page:

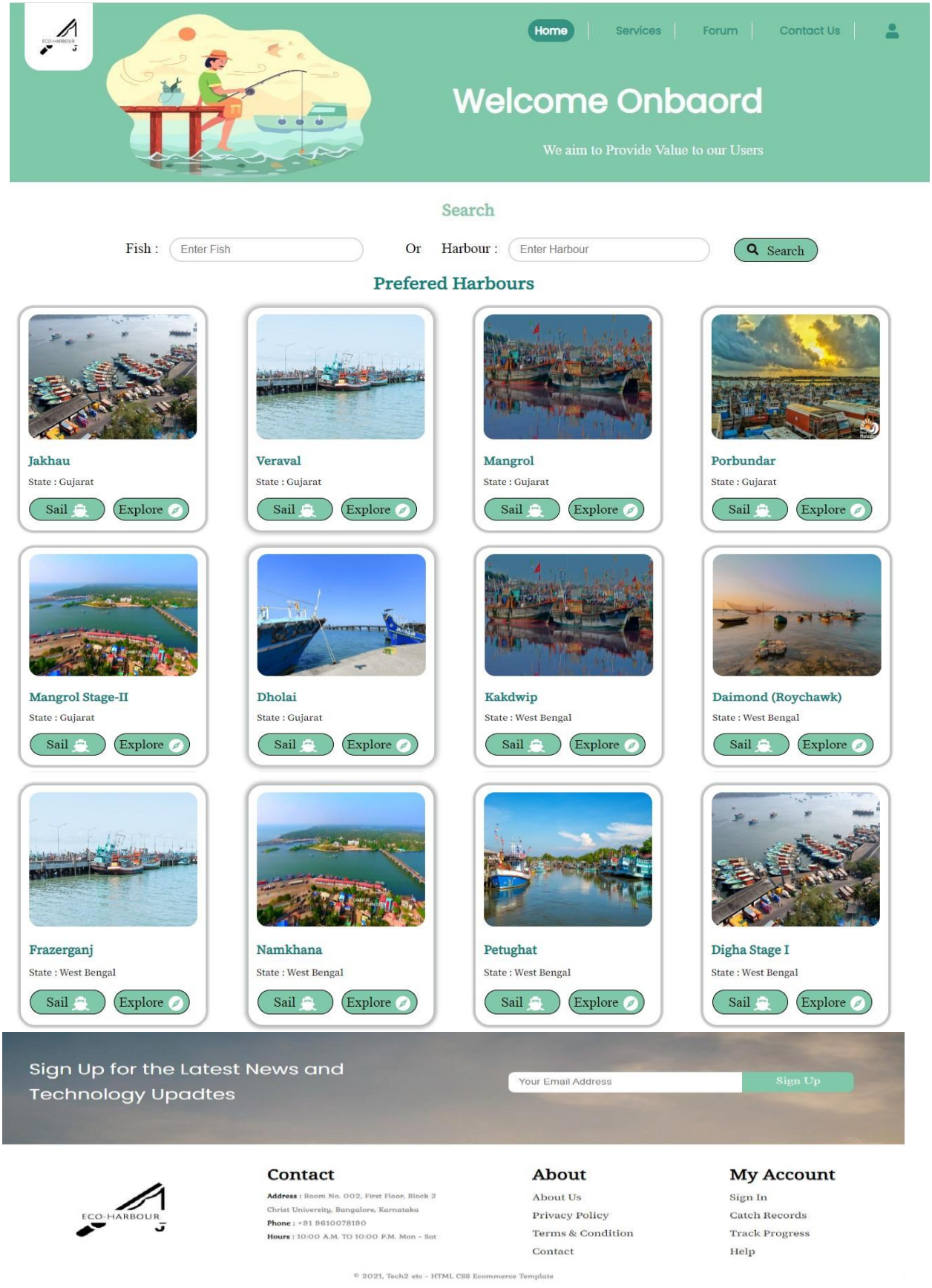

Fig 3.9 Service Page of EcoHarbour

Forum Page:

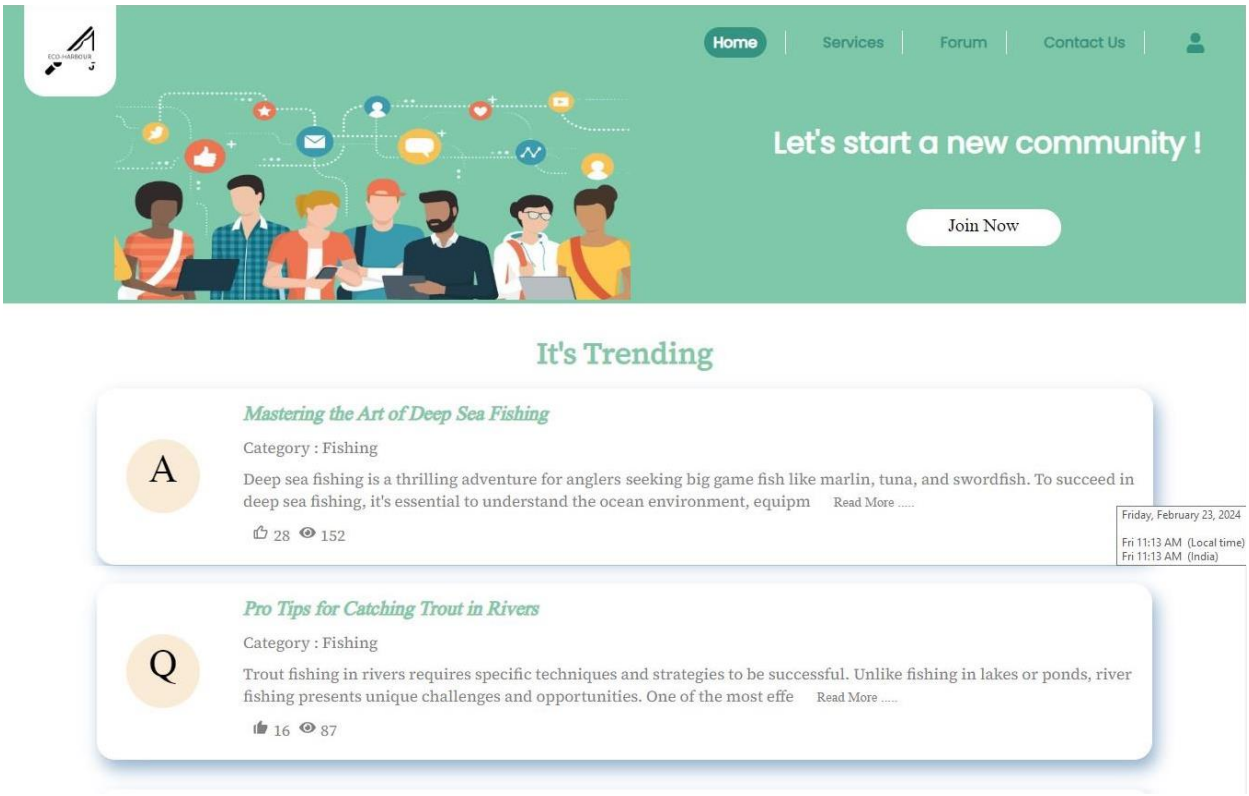

Fig 3.19 Forum Page of EcoHarbour

Create Post Page:

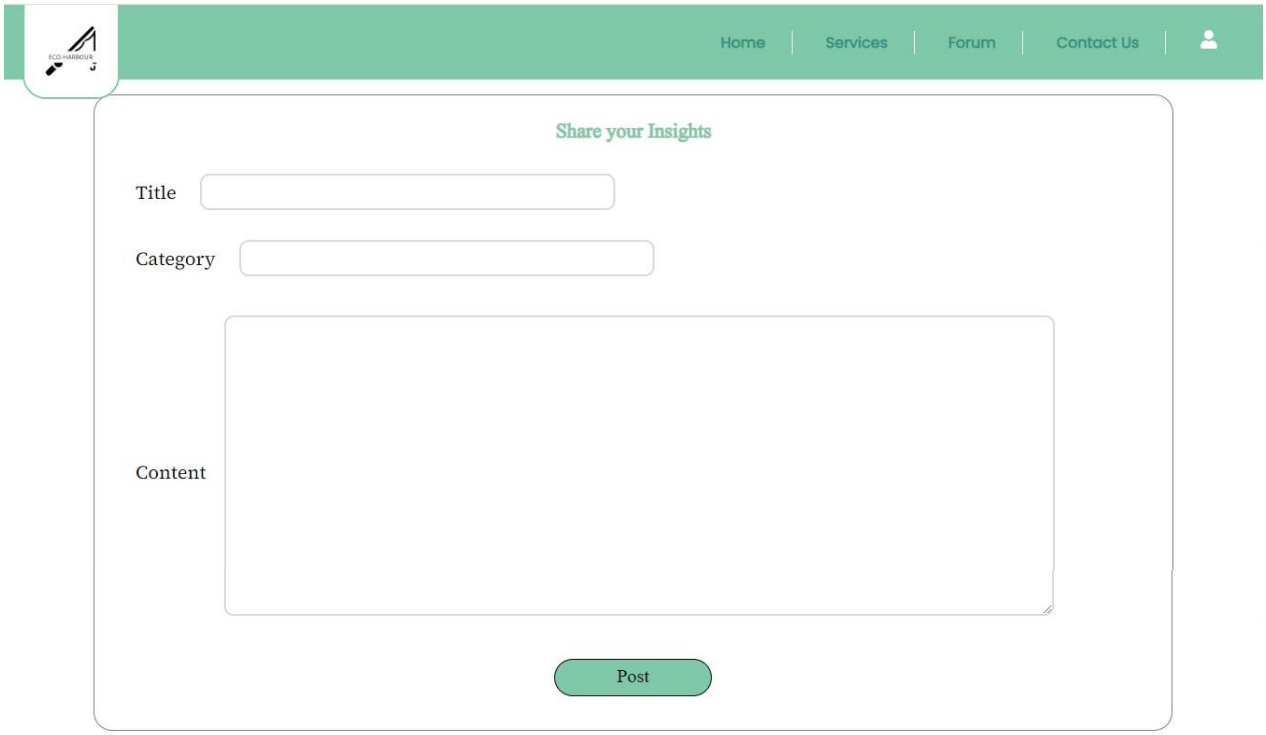

Fig 3.10 Create Post Page of EcoHarbour

Contact Us Page:

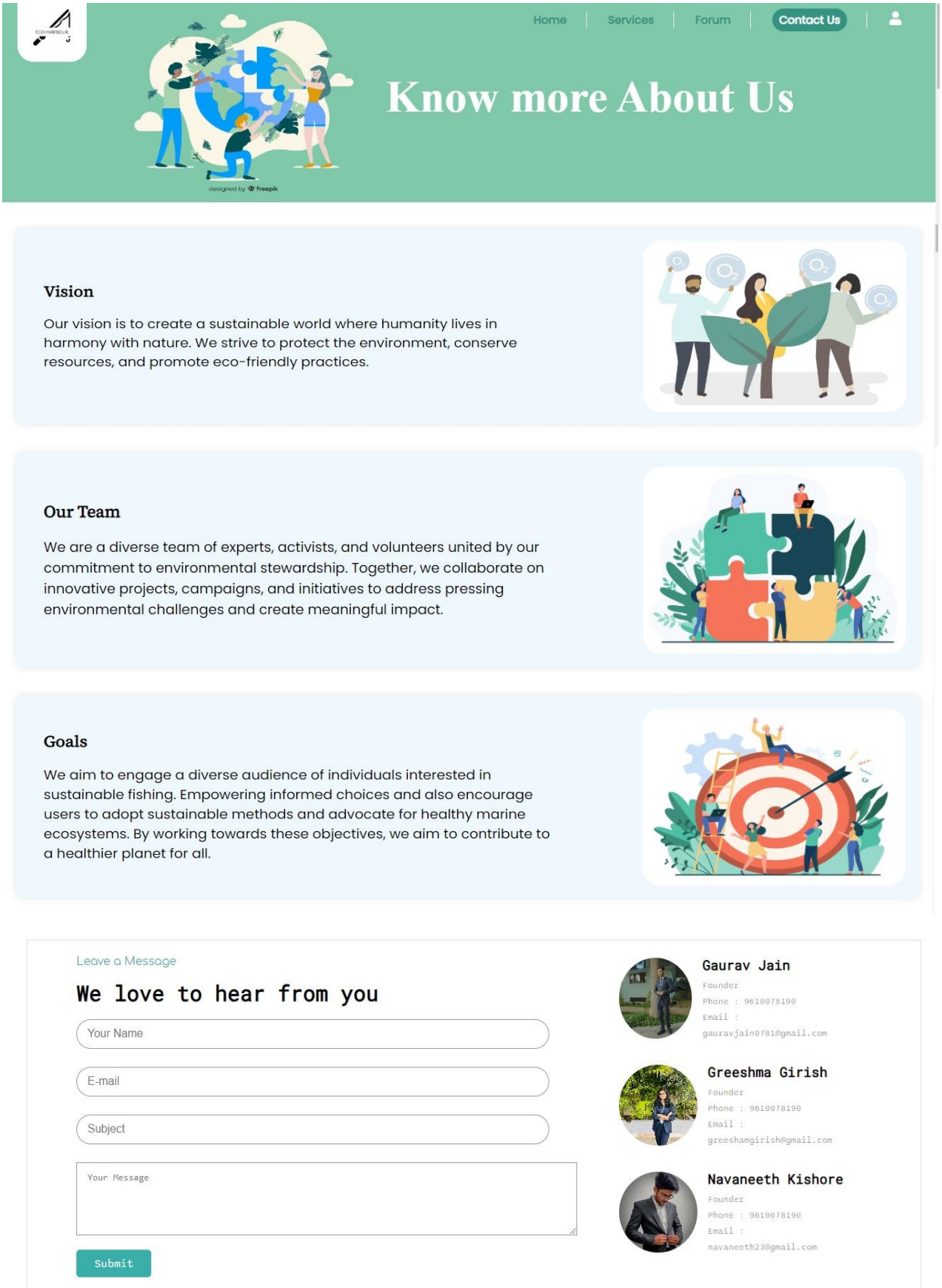

Fig 3.11 Contact Us page of EcoHarbour

Login Page:

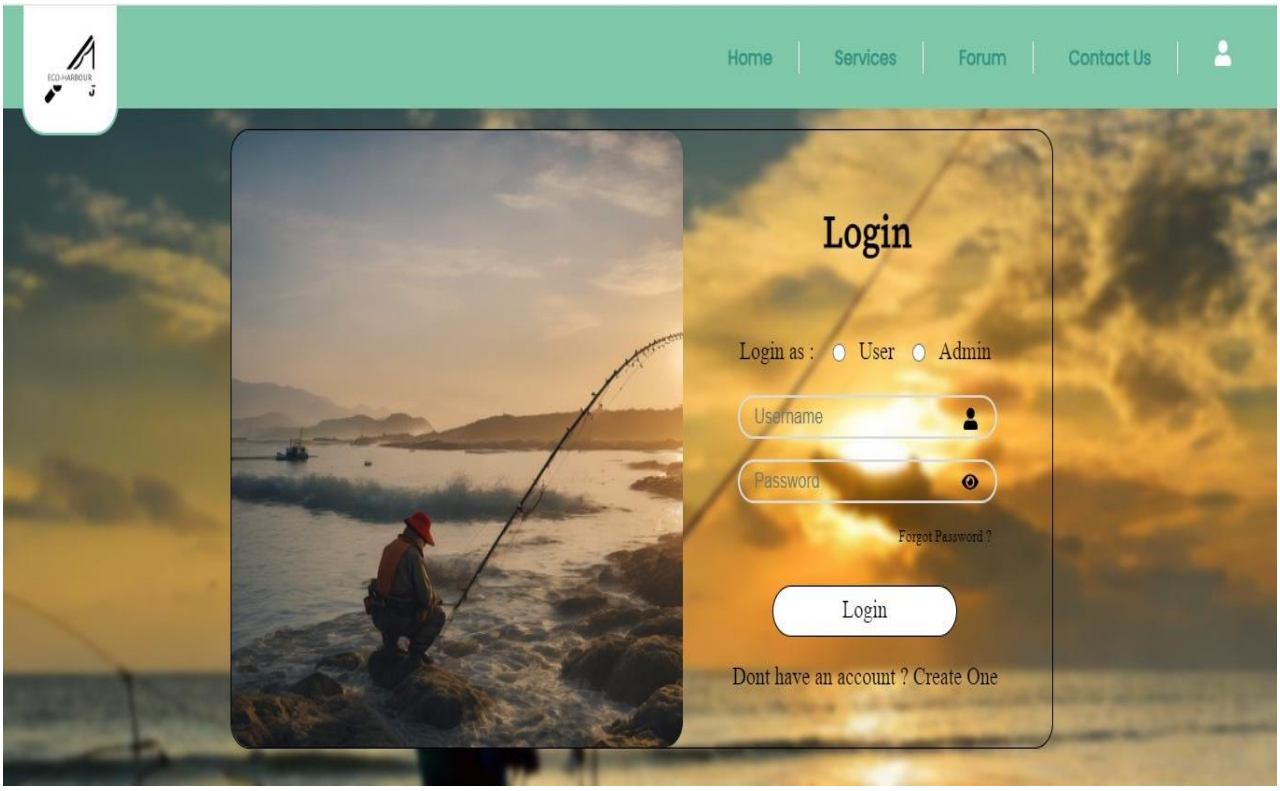

Fig 3.12 Login Page of EcoHarbour

Registration Page:

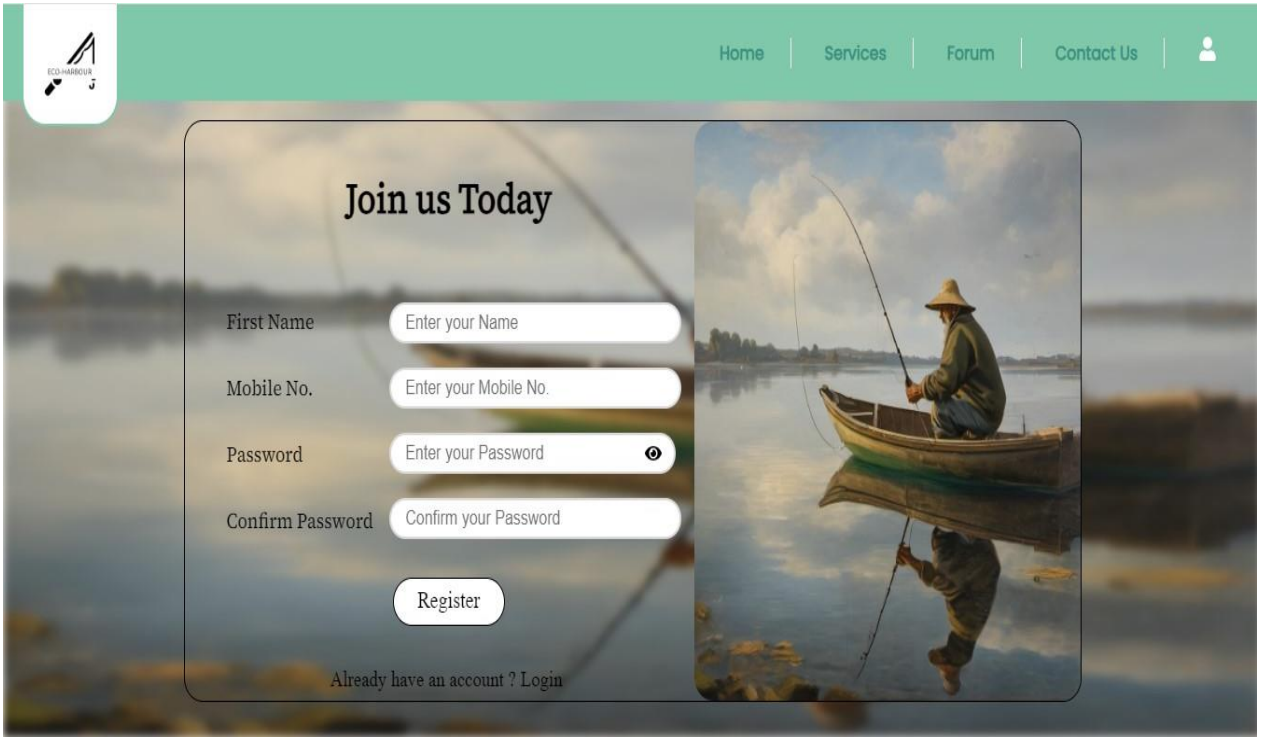

Fig 3.13 Registration Page of EcoHarbour

User Dashboard:

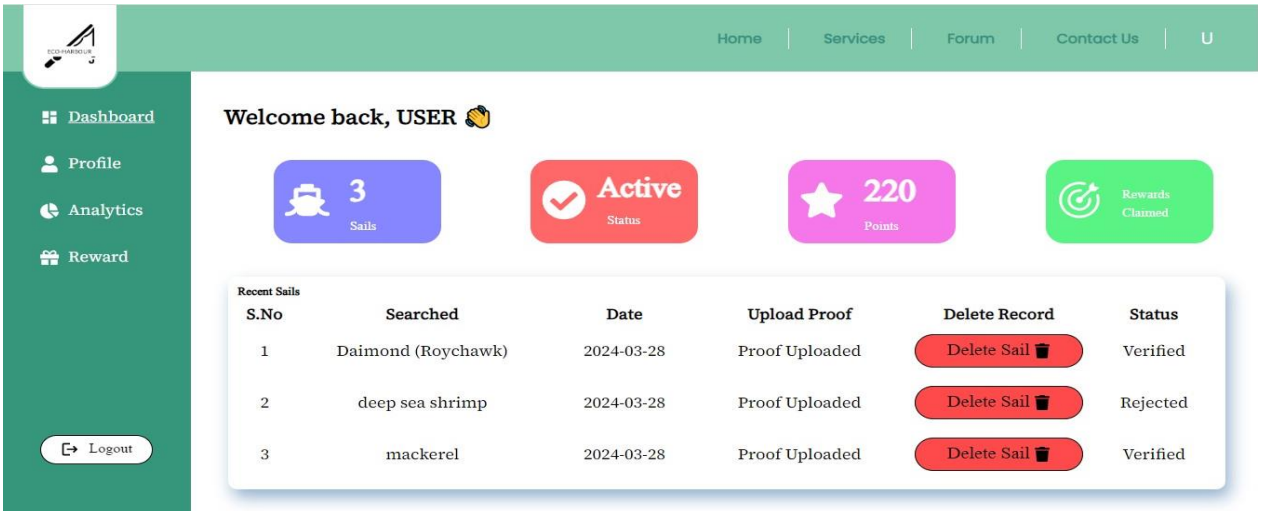

Fig 3.14 User Dashboard Page of EcoHarbour

User Dashboard (Profile Section):

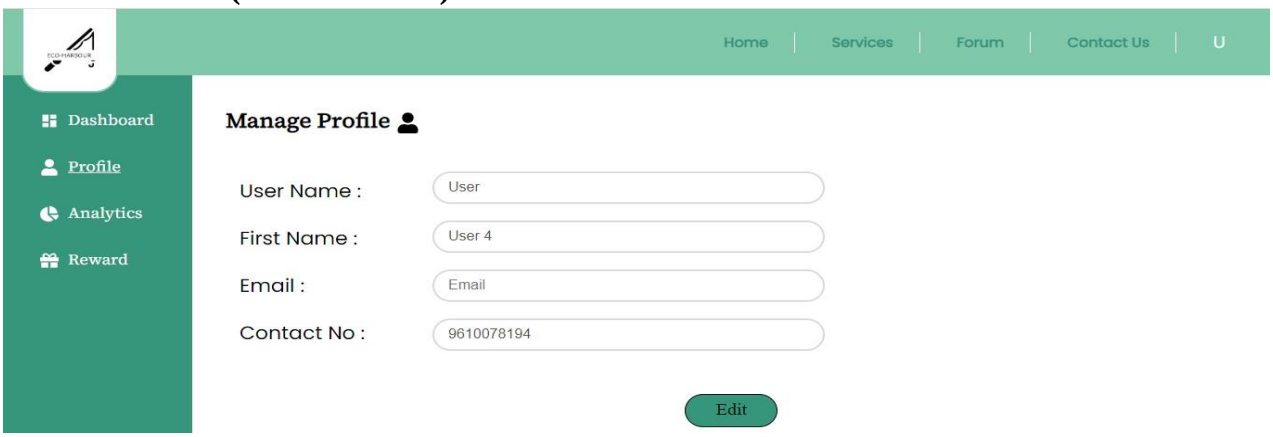

Fig 3.15 User Dashboard Page of EcoHarbour

User Dashboard (Analytics Section):

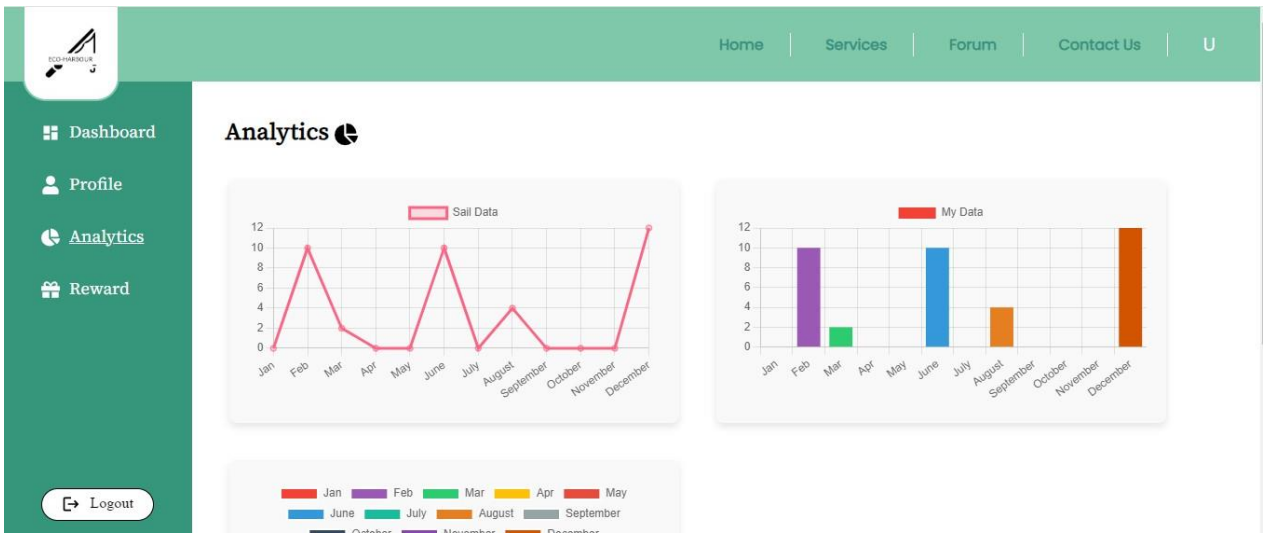

Fig 3.16 User Dashboard Page of EcoHarbour

User Dashboard (Reward Section):

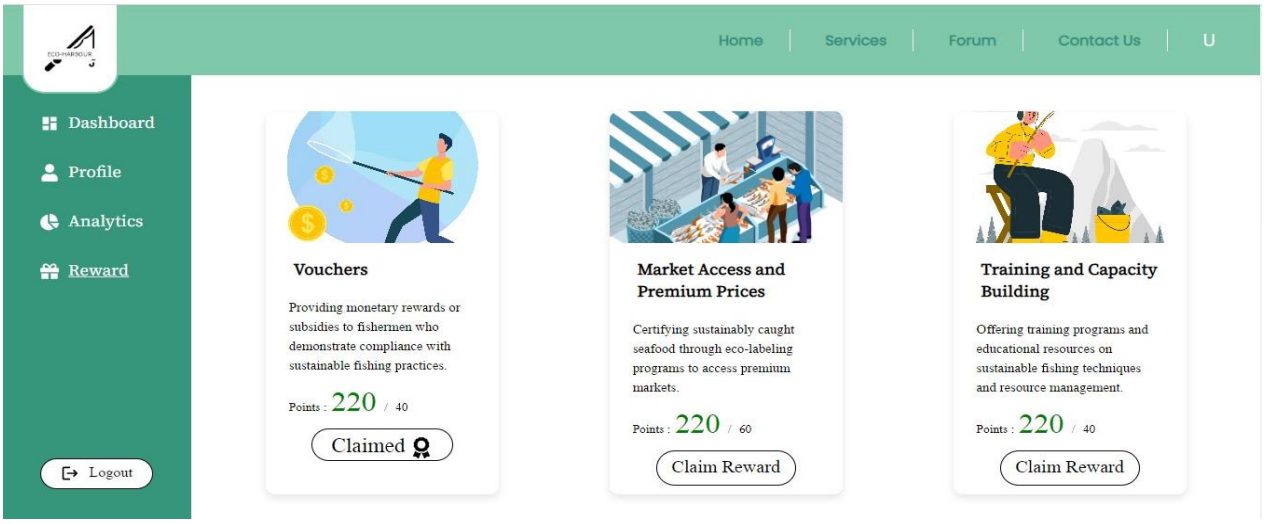

Fig 3.17 User Dashboard Page of EcoHarbour

Admin Dashboard:

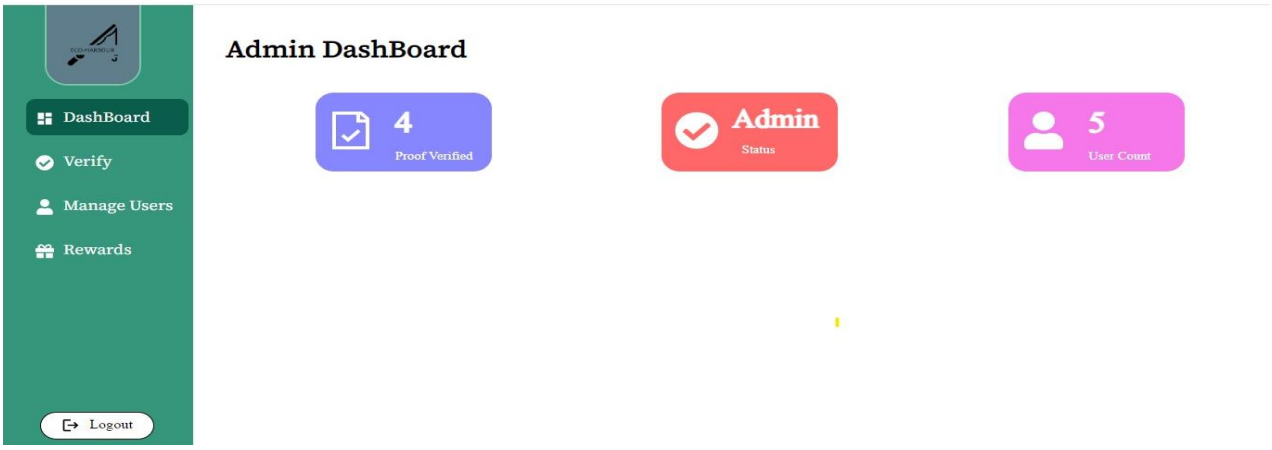

Fig 3.18 Admin Dashboard Page of EcoHarbour

Admin Dashboard (Verify Section):

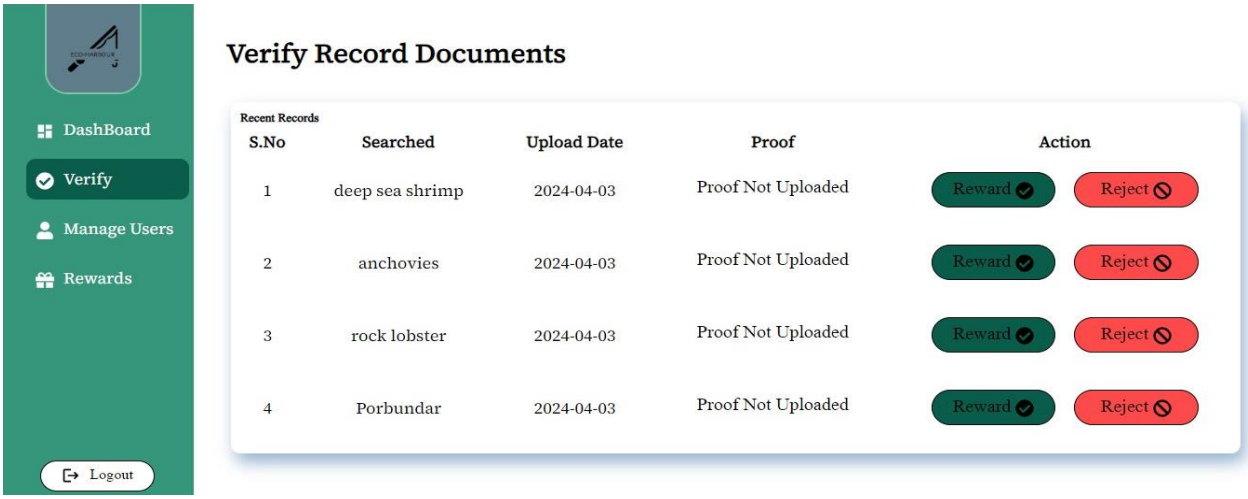

Fig 3.19 Admin Dashboard Page of EcoHarbour

Admin Dashboard (Manage User Section):

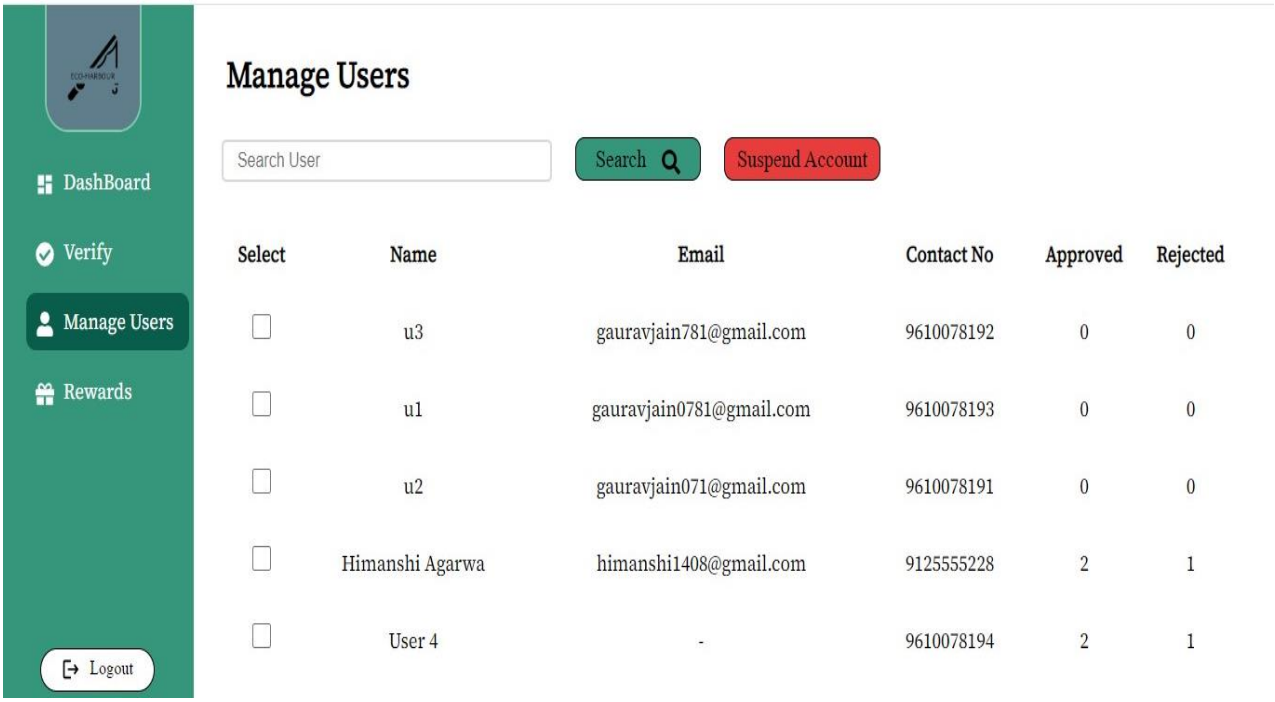

Fig 3.20 Admin Dashboard Page of EcoHarbour

4.2.2 Database Screenshots

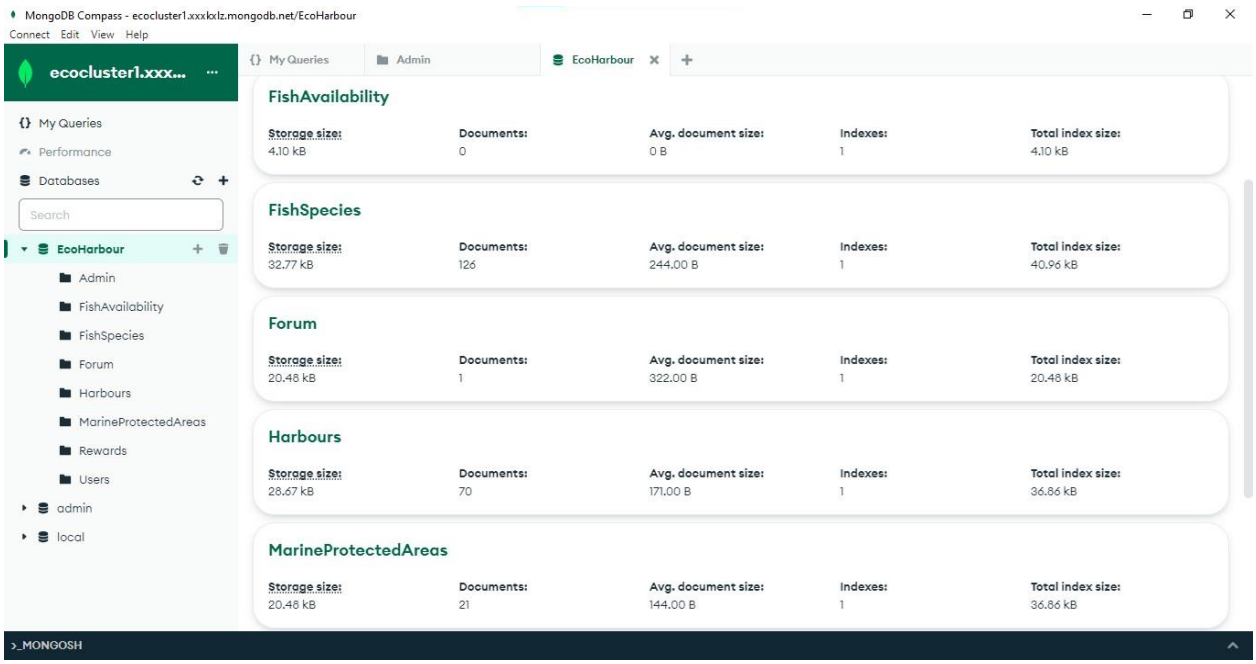

Fig 3.21 MongoDB Database Eco-Harbour

Fish Species Table:

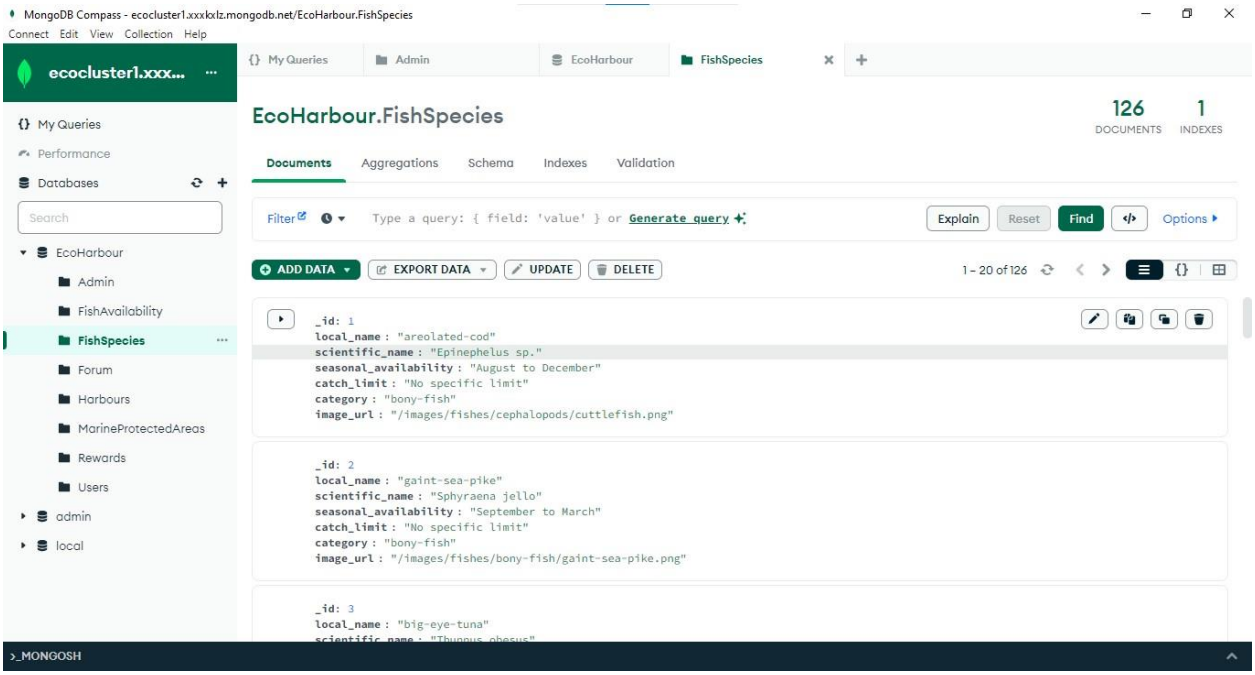

Fig 3.22 Fish Species Table in EcoHarbour Database

Harbours Table:

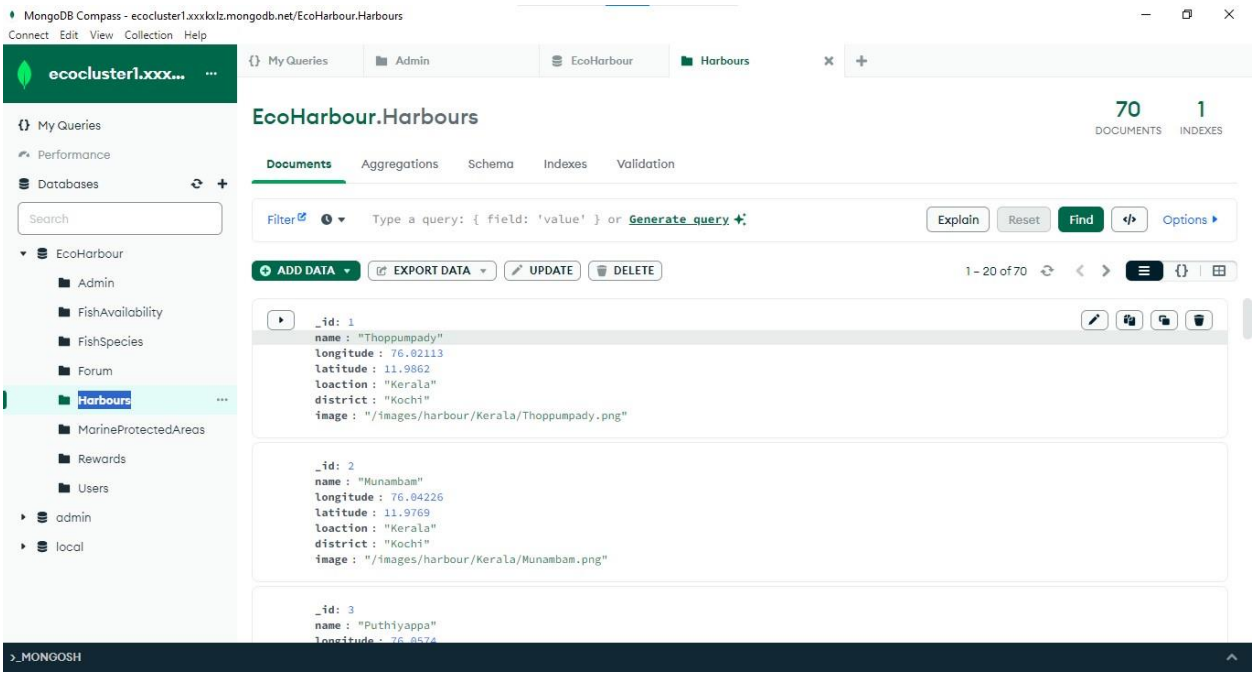

Fig 3.23 Harbour Table in EcoHarbour Database

Marine Protected Area:

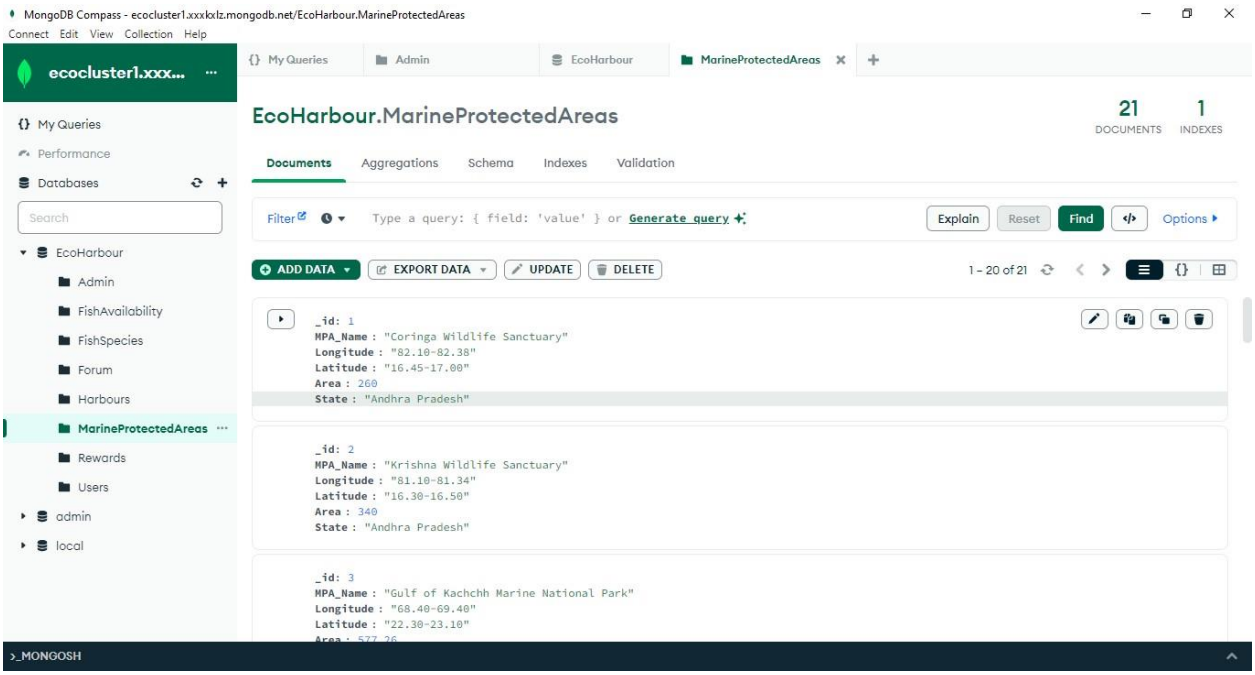

Fig 3.24 Marine Protected Area Table in EcoHarbour Database

Abundances:

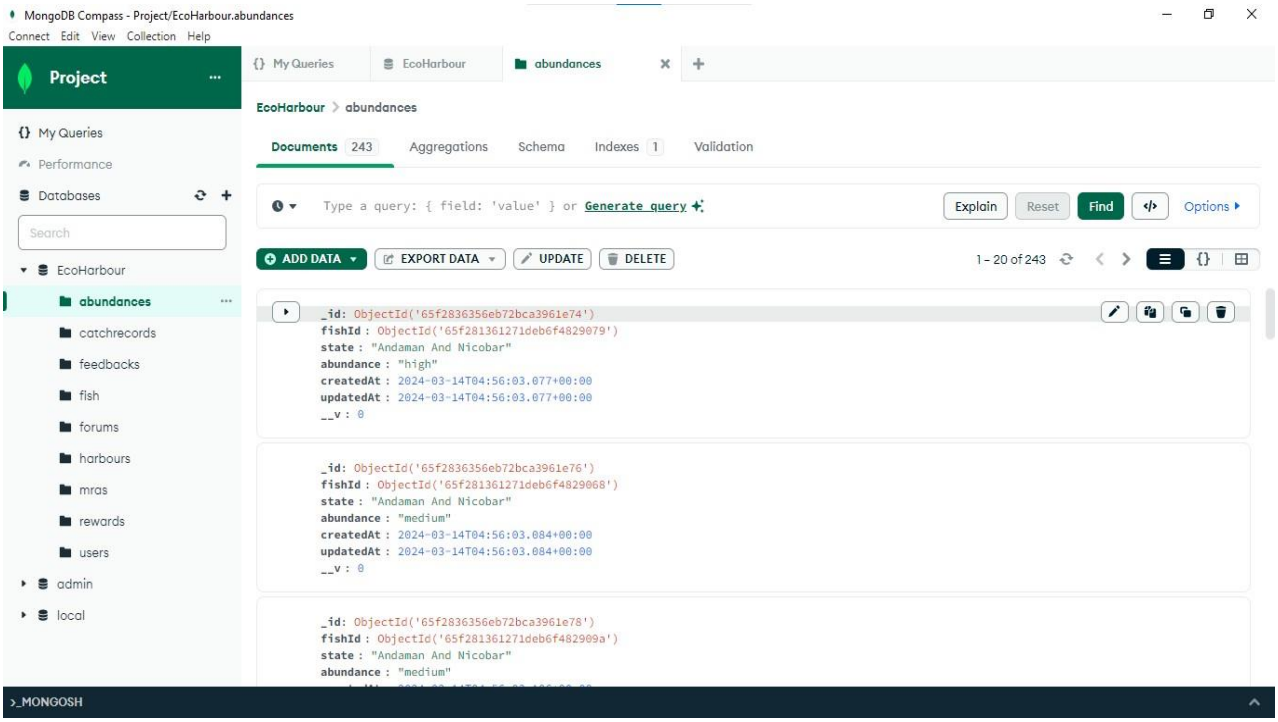

Fig 3.25 Abundances Table in EcoHarbour Database

Catch Records:

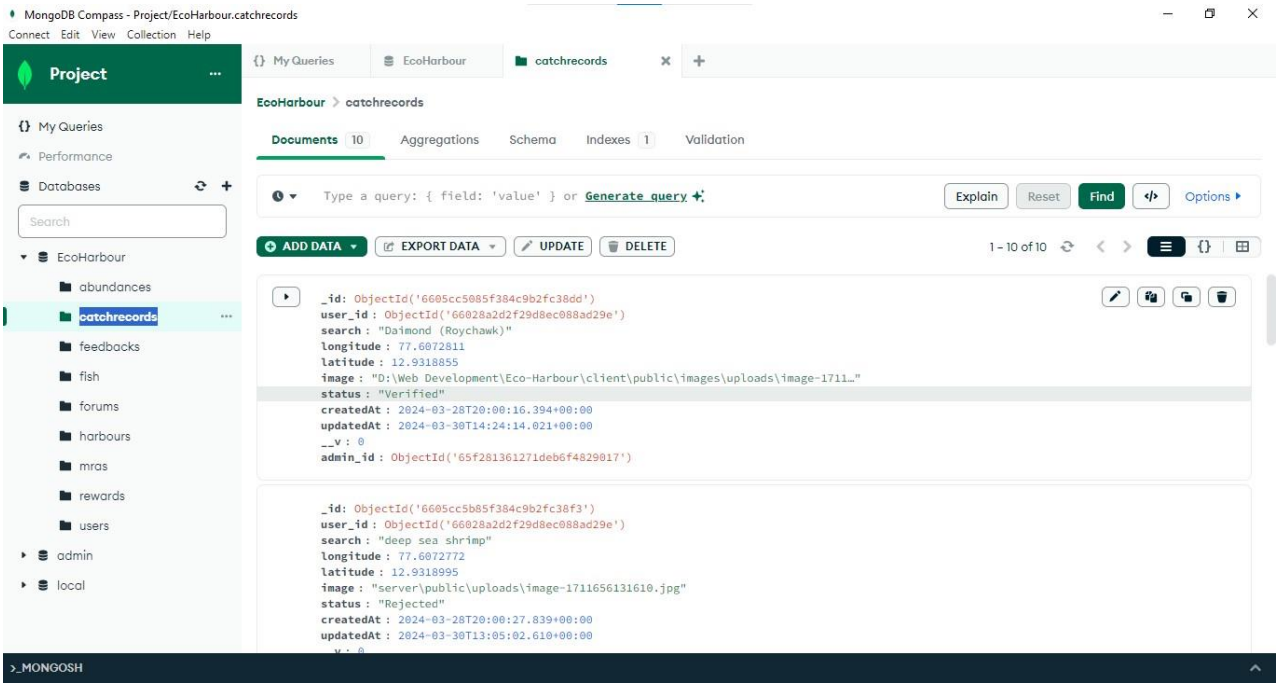

Fig 3.26 Catch Record Table in EcoHarbour Database

Feedback Table:

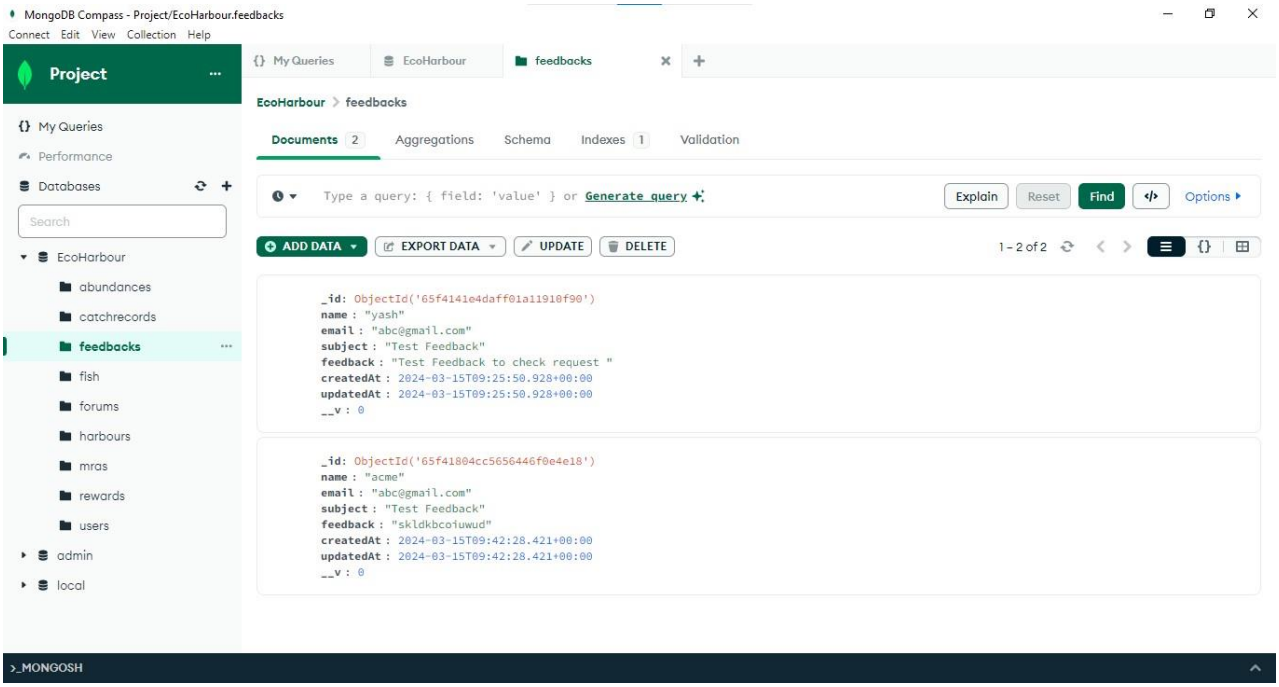

Fig 3.27 Feedback Table in EcoHarbour Database

Forum Table:

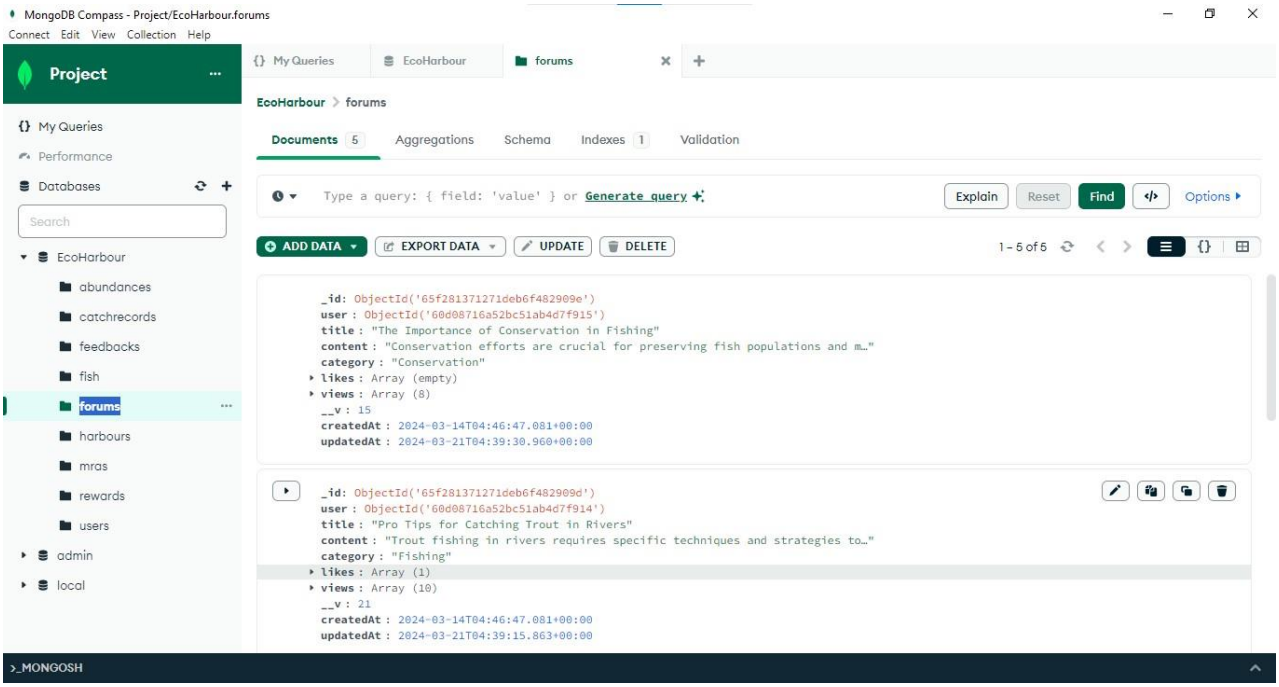

Fig 3.28 Forums Table in EcoHarbour Database

Rewards Table:

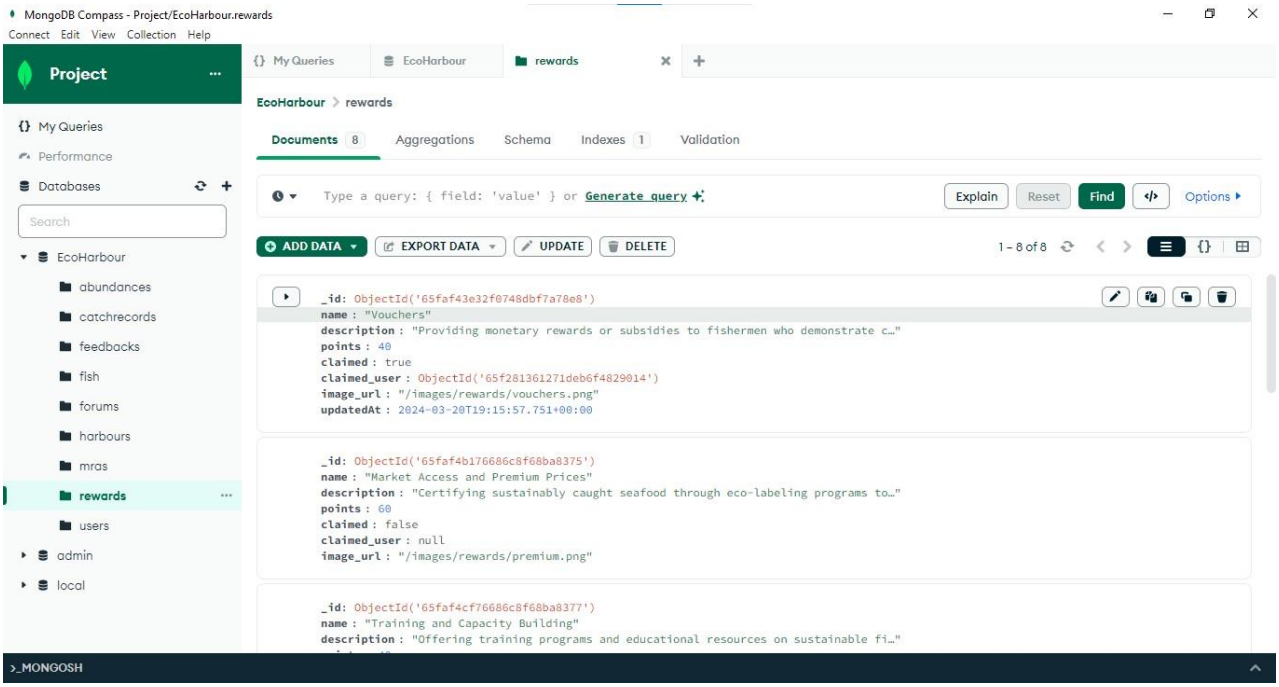

Fig 3.29 Rewards Table in EcoHarbour Database

5 TESTING

5.1 METHODS OF TESTING

A software testing strategy incorporates software test cases into a sequence of meticulously organized stages that culminate in the effective development of software.

Verification and validation are more broadly defined as software testing. The set of procedures known as verification makes sure that the developed software can be linked back to the needs of the client.

The steps involved in testing are-

5.1.1 Unit Testing: Testing each design unit independently is known as unit testing. In this project, we independently tested each design unit to ensure that there no mistakes. For this testing, every design is executed separately. If an error arises after each page is executed, the rectification method is completed immediately.

5.1.2 Integration testing: In our project, we combine many units of modules to form a sub-system. These sub-systems are then tested. This is done to see whether the modules can be integrated properly. Based on integration testing some necessary changes were made to the design.

5.1.3 System testing: System testing is done to ensure the entire software performs its function as intended. In our project all the tested subsystems were integrated and tested for all the possible ranges of coupling variables, based on the testing errors were rectified for a pleasant working experience.

5.1.4 Acceptance testing: The goal of acceptance testing is to see if the software meets all the requirements as needed. The testing was performed by data of all the users of the system. It was found that the software meets all the requirements of the host, tenant, administrator, and service providers as needed.

5.2 Test Cases and Reports

Login Module

1.1	**Input**: Valid Credentials with User Type	**Output**: Login Successful	User logged into account and redirected to home page
1.2	**Input**: Valid Credentials with blank fields	**Output**: Please Fill all the details	User is shown an error message that credentials are empty
1.3	**Input**: Invalid Credentials	**Output**: Invalid Email or Password	User is shown an error message that credentials is wrong.

Registration Module

2.1	**Input**: Valid Credentials with User Type	**Output**: Registration Successful	User registered and redirected to login page
2.2	**Input**: Valid **Credentials** with blank fields	**Output**: Please Fill all the details	User is shown a error message that credentials are empty
2.3	**Input**: Invalid Credentials	**Output**: Invalid Data Entered	User is shown an error message that credentials is wrong.

Abundance Search Module

3.1	**Input**: Valid Fish Name or Harbour Name	**Output**: Preferred Fishes or Harbours	User is shown all the preferred harbours or fishes in a card format

3.2	**Input**: Blank fields	**Output**: Please enter Fish Name or Harbour Name	User is shown an error that enter one of the fields name
3.3	**Input**: Invalid Fish Name or Harbour Name	**Output**: No Fish or Harbour Found	User is shown a message that no such harbour or fish data is there.

Sail Module

4.1	**Input**: Marking Sail without Login	**Output**: Please Login before Sail	User is shown a warning message to login and is redirected to login page.
4.2	**Input**: Marking Sail with Login	**Output**: Sail Marked	User is shown success message that sail has been marked and upload proof button comes.

Feedback Module

5.1	**Input**: Providing Feedback with Login	**Output**: Feedback Recorded	User is shown a success message.
5.2	**Input**: Providing Feedback without Login	**Output**: Please Login before providing Feedback	User is shown warning message to login and is redirected to login page.

Forum Module

6.1	**Input**: Like Post before login	**Output**: Please Login before liking Post.	User is shown warning message to login and is redirected to login page.
6.2	**Input**: Like Post after login	**Output**: Post Liked	The icon is changed to liked post icon.

Post Module

7.1	**Input**: Post Experience before login	**Output**: Please Login before Making a Post	User is shown warning message to login and is redirected to login page.
7.2	**Input**: Post Experience after login	**Output**: Experience Posted.	User is shown a success message that post can be created.

Upload Proof Module

8.1	**Input**: Uploading Image	**Output**: Proof Uploaded Successfully	User is shown a success message that proof has been uploaded successfully.
8.2	**Input**: Uploading Other File Type	**Output**: Invalid File Type	User is shown an error message that invalid file type has been selected.

Delete Sail Module

9.1	**Input**: Delete Sail	**Output**: Sail Deleted Successfully.	User is shown a success message that sail has been deleted successfully.

Manage Account Module

10.1	**Input**: Update Account with Invalid Data	**Output**: Please provide the correct data.	User is shown an error message that invalid data has been entered
10.2	**Input**: Update Account with Valid Data	**Output**: Profile Updated	User is shown a success message that account details has been updated successfully.

Rewards Module

11.1	**Input**: Claim Reward with insufficient Points	**Output**: Insufficient Points	User is an error message that he/she does not have sufficient points to claim the reward.
11.2	**Input**: : Claim Reward with sufficient Points	**Output**: Reward Claimed	User is shown a success message that reward has been claimed.

6 CONCLUSIONS

The EcoHarbour project has come up with a highly accessible and user-friendly fishing sustainability management website that has made it extremely easy for users to search for fish and harbours filtered based on location anywhere, anytime. The website is built keeping in mind the changing needs of today's fishermen and simplifies the entire process of abundance search. Fishermen no longer need to rely on unverified data and now can make informed decisions regarding sail. The website is loaded with features that give fishermen total control over their sails. For instance, uploading proof and deleting sail or managing an account can be done effortlessly.

The EcoHarbour project is a significant step towards empowering fishermen and maintaining sustainability in fishing practices. The user-friendly website is designed to cater all the needs of all kinds of fishermen, including those who are not tech-savvy. The website is easy to navigate and offers a seamless user experience. The website is secure, and fishermen can be assured that their personal and sails information is safe.

In conclusion, the EcoHarbour project is a remarkable innovation in the fishing industry that has transformed the way commercial fishing works. The project has significantly reduced the time and effort needed to retrieve the abundance of information. With its user-friendly interface, safe and practical fishing approach, and total control over sails, the EcoHarbour website is an ideal choice for fishermen who want to increase their productivity and maintain sustainability.

REFERENCES

[1] MPEDA, "Major Harbours Database" *Major Fishing Harbours*. 3 April, 2024.

< https://mpeda.gov.in/?page_id=1007>.

[2] MPEDA, "Marine Fishing Landings" *Marine Fishing Landings and boat Landings*. April, 2021.

< https://mpeda.gov.in/fishers/wp-content/uploads/2021/08/006-MARINE-FISH-LANDINGS-JUNE-2021.pdf >.

[3] PIB, "Ministry of Fisheries, Animal Husbandry & Dairying" *Modernising Fishing Harbours*. 29 March, 2024. <https://pib.gov.in/Pressreleaseshare.aspx?PRID=1810953 >.

[4] Dr.Gopalakrishnan A., "Marine Fish Landings" *Marine Fish Landings in 2023*. April, 2023.

<http://eprints.cmfri.org.in/16042/1/Marine%20Fish%20Landings%20in%20India%202021.pdf >.