|  |  |  |
| --- | --- | --- |
| **Ex. No. 10**  **Date:** | **MINI PROJECT** | **Page No.:** |

**Title:**

PET ADOPTION SYSTEM

**GitHubProject Link:**

<https://github.com/Madhumitha524/pet-adoption-system>

**GitHubProject DocumentLink:**

https://github.com/archanamarikani-cmd/PetAdoption

**Team Member:**

S.Soniya-23uit028

S.Madhumitha-23uit093

M.Archana-23uit100

**Frontend : HTML, CSS, ReactJS**

**Backend :NodeJS**

**Database :MongoDB**

**Objective of the Project:**

The Pet Adoption System aims to create a seamless platform that connects pets in need of homes with potential adopters. It focuses on making the adoption process easy, transparent, and accessible for everyone. By digitizing pet listings and adoption requests, the system helps reduce the time animals spend in shelters. It encourages responsible pet ownership and raises awareness about the importance of adoption over buying. The platform also supports shelters and rescue groups in managing their pets efficiently. Overall, it promotes compassion and care for animals seeking loving families.

**Problem Statement:**

Many animals in shelters or with owners are in urgent need of adoption but face challenges due to lack of awareness, inefficient communication, and manual processes. Potential adopters often find it difficult to access comprehensive information about available pets or to connect with shelters quickly. This leads to longer shelter stays for animals and missed opportunities for adoption. There is a need for a centralized, easy-to-use digital platform that streamlines the adoption process, improves pet visibility, and facilitates better communication between adopters and shelters.

**Code:**

Backend

Config/db.js

import mongoose from "mongoose";

import dotenv from "dotenv";

dotenv.config();

constconnectDB = async () => {

  try {

    await mongoose.connect(process.env.MONGO\_URI, {

      useNewUrlParser: true,

      useUnifiedTopology: true,

    });

    console.log("✅ MongoDB Connected");

  } catch (error) {

    console.error("❌ MongoDB connection failed:", error.message);

    process.exit(1);

  }

};

export default connectDB;

middleware/authMiddleware.js

import jwt from "jsonwebtoken";

import User from "../models/User.js";

export const protect = async (req, res, next) => {

  let token;

  if (req.headers.authorization?.startsWith("Bearer")) {

    token = req.headers.authorization.split(" ")[1];

  }

  if (!token) {

    return res.status(401).json({ message: "Not authorized" });

  }

  try {

    const decoded = jwt.verify(token, process.env.JWT\_SECRET);

    req.user = await User.findById(decoded.id).select("-password");

    next();

  } catch (error) {

    res.status(401).json({ message: "Token failed" });

  }

};

Models/Adoption.js

import mongoose from "mongoose";

constadoptionSchema = new mongoose.Schema({

  user: { type: mongoose.Schema.Types.ObjectId, ref: "User" },

  pet: { type: mongoose.Schema.Types.ObjectId, ref: "Pet" },

  date: { type: Date, default: Date.now },

});

export default mongoose.model("Adoption", adoptionSchema);

Pet.js

import mongoose from "mongoose";

constpetSchema = new mongoose.Schema({

  name: String,

  type: String,

  image: String,

  adopted: { type: Boolean, default: false },

});

export default mongoose.model("Pet", petSchema);

petModels.js

import mongoose from "mongoose";

constpetSchema = new mongoose.Schema({

  name: String,

  breed: String,

  age: Number,

  image: String,

});

const Pet = mongoose.model("Pet", petSchema);

export default Pet;

User.js

import mongoose from "mongoose";

import bcrypt from "bcryptjs";

constuserSchema = new mongoose.Schema({

  name: { type: String, required: true },

  age: Number,

  address: String,

  hasSpace: Boolean,

  nearbyVet: Boolean,

  salary: Number,

  phone: String,

  email: { type: String, required: true, unique: true },

  password: { type: String, required: true },

});

userSchema.pre("save", async function (next) {

  if (!this.isModified("password")) return next();

  const salt = await bcrypt.genSalt(10);

  this.password = await bcrypt.hash(this.password, salt);

  next();

});

const User = mongoose.model("User", userSchema);

export default User;

routes/adoptionRoutes.js

import express from "express";

import jwt from "jsonwebtoken";

import Pet from "../models/petModel.js";

import User from "../models/user.js";

const router = express.Router();

const auth = (req, res, next) => {

  const token = req.headers.authorization?.split(" ")[1];

  if (!token) return res.status(401).json({ message: "Unauthorized" });

  try {

    const decoded = jwt.verify(token, process.env.JWT\_SECRET);

    req.userId = decoded.id;

    next();

  } catch (err) {

    res.status(401).json({ message: "Invalid token" });

  }

};

// Adopt pet

router.post("/", auth, async (req, res) => {

  const{ petId } = req.body;

  try {

    const pet = await Pet.findById(petId);

    if (!pet) return res.status(404).json({ message: "Pet not found" });

    const user = await User.findById(req.userId);

    res.json({ message: `Pet ${pet.name} adopted successfully by ${user.name}` });

  } catch (err) {

    res.status(500).json({ message: "Error adopting pet" });

  }

});

export default router;

authRoutes.js

import express from "express";

import User from "../models/user.js";

import bcrypt from "bcryptjs";

import jwt from "jsonwebtoken";

const router = express.Router();

// Register

router.post("/register", async (req, res) => {

  const{ name, age, address, hasSpace, nearbyVet, salary, phone, email, password } = req.body;

  try {

    const existing = await User.findOne({ email });

    if (existing) return res.status(400).json({ message: "User already exists" });

    const user = new User({ name, age, address, hasSpace, nearbyVet, salary, phone, email, password });

    await user.save();

    const token = jwt.sign({ id: user.\_id }, process.env.JWT\_SECRET, { expiresIn: "1d" });

    res.json({ message: "Register successful", token });

  } catch (err) {

    res.status(500).json({ message: "Error registering user" });

  }

});

// Login

router.post("/login", async (req, res) => {

  const{ email, password } = req.body;

  try {

    const user = await User.findOne({ email });

    if (!user) return res.status(400).json({ message: "Invalid credentials" });

    constisMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) return res.status(400).json({ message: "Invalid credentials" });

    const token = jwt.sign({ id: user.\_id }, process.env.JWT\_SECRET, { expiresIn: "1d" });

    res.json({ message: "Login successful", token });

  } catch (err) {

    res.status(500).json({ message: "Error logging in" });

  }

});

export default router;

petRoutes.js

import express from "express";

import Pet from "../models/petModel.js";

const router = express.Router();

router.get("/", async (req, res) => {

  try {

    const pets = await Pet.find({});

    res.json(pets);

  } catch (err) {

    res.status(500).json({ message: "Error fetching pets" });

  }

});

export default router;

seeders/petSeeders.js

import mongoose from "mongoose";

import dotenv from "dotenv";

import Pet from "../models/petModel.js";

dotenv.config();

mongoose.connect(process.env.MONGO\_URI)

  .then(() =>console.log("✅ MongoDB Connected"))

  .catch(err => console.log(err));

const pets = [

  { name: "Bella", breed: "Labrador", age: 2, image: "bella.jpg" },

  { name: "Luna", breed: "Golden Retriever", age: 3, image: "luna.jpg" },

  { name: "Milo", breed: "Beagle", age: 1, image: "milo.jpg" },

  { name: "Coco", breed: "Poodle", age: 4, image: "coco.jpg" },

  { name: "Buddy", breed: "German Shepherd", age: 3, image: "buddy.jpg" },

  { name: "Max", breed: "Bulldog", age: 2, image: "max.jpg" },

];

constseedPets = async () => {

  try {

    await Pet.deleteMany();

    await Pet.insertMany(pets);

    console.log("🌱 Pets seeded successfully!");

    process.exit();

  } catch (err) {

    console.log(err);

    process.exit(1);

  }

};

seedPets();

.env

PORT=5000

MONGO\_URI=mongodb://127.0.0.1:27017/petadoption

JWT\_SECRET=mysecret

Package.json

{

  "name": "pet-adoption-backend",

  "version": "1.0.0",

  "description": "Backend for Pet Adoption System using Node.js, Express, MongoDB",

  "main": "server.js",

  "type": "module",

  "scripts": {

    "start": "node server.js",

    "dev": "nodemon server.js",

    "seed": "node seeders/petSeeder.js"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "bcryptjs": "^2.4.3",

    "cors": "^2.8.5",

    "dotenv": "^16.4.5",

    "express": "^4.18.2",

    "jsonwebtoken": "^9.0.2",

    "mongoose": "^8.0.3"

  },

  "devDependencies": {

    "nodemon": "^3.1.0"

  }

}

seedPets.js

import mongoose from "mongoose";

import dotenv from "dotenv";

import connectDB from "./config/db.js";

import Pet from "./models/Pet.js";

dotenv.config();

constseedPets = async () => {

  try {

    await connectDB();

    constexistingPets = await Pet.find();

    if(existingPets.length> 0) {

      console.log("✅ Pets already exist, skipping seeding");

      process.exit();

    }

    const pets = [

      { name: "Buddy", type: "Dog", image: "https://placedog.net/400/400?id=1" },

      { name: "Luna", type: "Cat", image: "https://placekitten.com/400/400" },

      { name: "Snowy", type: "Rabbit", image: "https://placebear.com/400/400" },

      { name: "Charlie", type: "Dog", image: "https://placedog.net/400/400?id=2" },

      { name: "Milo", type: "Cat", image: "https://placekitten.com/401/401" },

      { name: "Coco", type: "Parrot", image: "https://www.fillmurray.com/400/400" }

    ];

    await Pet.insertMany(pets);

    console.log("🌱 Pets seeded successfully!");

    process.exit();

  } catch (error) {

    console.error(error);

    process.exit(1);

  }

};

seedPets();

server.js

import express from "express";

import dotenv from "dotenv";

import mongoose from "mongoose";

import cors from "cors";

import path from "path";

import { fileURLToPath } from "url";

import authRoutes from "./routes/authRoutes.js";

import petRoutes from "./routes/petRoutes.js";

import adoptionRoutes from "./routes/adoptionRoutes.js";

dotenv.config();

const app = express();

app.use(cors());

app.use(express.json());

const \_\_filename = fileURLToPath(import.meta.url);

const \_\_dirname = path.dirname(\_\_filename);

// Serve images

app.use("/pet-images", express.static(path.join(\_\_dirname, "public/pet-images")));

// Routes

app.use("/api/auth", authRoutes);

app.use("/api/pets", petRoutes);

app.use("/api/adoptions", adoptionRoutes);

mongoose.connect(process.env.MONGO\_URI)

  .then(() =>console.log("✅ MongoDB connected"))

  .catch((err) => console.log(err));

const PORT = process.env.PORT || 5000;

app.listen(PORT, () =>console.log(`Server running on port ${PORT}`));

Frontend

Src/components/Login.js

import React, { useState } from "react";

import API from "../services/api";

import { useNavigate, Link } from "react-router-dom";

import "../styles/auth.css";

export default function Login() {

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const [msg, setMsg] = useState("");

  const [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  consthandleLogin = async (e) => {

    e.preventDefault();

    setMsg("");

    setLoading(true);

    try {

      const res = await API.post("/auth/login", { email, password });

      setLoading(false);

      setMsg(res.data.message || "Login successful");

      // save token if backend returns it

      if (res.data.token) localStorage.setItem("token", res.data.token);

      // redirect after short delay so user sees the message

      setTimeout(() => navigate("/pets"), 900);

    } catch (err) {

      setLoading(false);

      constserverMsg = err?.response?.data?.message;

      setMsg(serverMsg || "Login failed. Check email/password.");

      console.error("Login error:", err?.response || err);

    }

  };

  return (

    <div className="auth-container">

      <form className="auth-form" onSubmit={handleLogin}>

        <h2>Login</h2>

        {msg&&<p className="message">{msg}</p>}

        <input type="email" placeholder="Email" value={email} onChange={(e)=>setEmail(e.target.value)} required />

        <input type="password" placeholder="Password" value={password} onChange={(e)=>setPassword(e.target.value)} required />

        <button type="submit" disabled={loading}>{loading ? "Logging in..." : "Login"}</button>

        <p style={{marginTop: "8px"}}>

          New user? <Link to="/register">Register here</Link>

        </p>

      </form>

    </div>

  );

}

Navbar.js

import React from "react";

import { Link } from "react-router-dom";

export default function Navbar() {

  return (

    <nav style={{ padding: "10px", background: "#f0f0f0" }}>

      <Link to="/" style={{ marginRight: "10px" }}>Login</Link>

      <Link to="/register" style={{ marginRight: "10px" }}>Register</Link>

      <Link to="/pets">Pets</Link>

    </nav>

  );

}

PetList.js

import React, { useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

import "../styles/petlist.css";

constPetList = () => {

  const [pets, setPets] = useState([]);

  const [msg, setMsg] = useState("");

  const navigate = useNavigate();

  useEffect(() => {

    fetch("http://localhost:5000/api/pets")

      .then((res) =>res.json())

      .then((data) => {

        console.log("Pets fetched:", data);

        setPets(data);

      })

      .catch((err) => console.log(err));

  }, []);

  consthandleAdopt = async (petId) => {

    const token = localStorage.getItem("token");

    if (!token) {

      alert("Please login first to adopt a pet");

      return;

    }

    try {

      const res = await fetch("http://localhost:5000/api/adoptions", {

        method: "POST",

        headers: {

          "Content-Type": "application/json",

          Authorization: `Bearer ${token}`,

        },

        body: JSON.stringify({ petId }),

      });

      const data = await res.json();

      if (res.ok) {

        setMsg("Pet adopted successfully!");

        setTimeout(() => navigate("/thankyou"), 1000);

      } else {

        setMsg(data.message || "Adoption failed");

      }

    } catch (err) {

      setMsg("Error adopting pet");

    }

  };

  return (

    <div className="pet-container">

      {msg&&<p className="adopt-msg">{msg}</p>}

      <div className="pet-grid">

        {pets.map((pet) => (

          <div key={pet.\_id} className="pet-card">

            <img

              src={`/pet-images/${pet.image}`}

              alt={pet.name}

              className="pet-img"

              onError={(e) => (e.target.src = "/pet-images/default.png")}

            />

            <h3>{pet.name || "Unknown"}</h3>

            <p>Breed: {pet.breed || "Unknown"}</p>

            <p>Age: {pet.age || "Unknown"}</p>

            <button onClick={() =>handleAdopt(pet.\_id)}>Adopt</button>

          </div>

        ))}

      </div>

    </div>

  );

};

export default PetList;

Register.js

import React, { useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

import "../styles/petlist.css";

constPetList = () => {

  const [pets, setPets] = useState([]);

  const [msg, setMsg] = useState("");

  const navigate = useNavigate();

  useEffect(() => {

    fetch("http://localhost:5000/api/pets")

      .then((res) =>res.json())

      .then((data) => {

        console.log("Pets fetched:", data);

        setPets(data);

      })

      .catch((err) => console.log(err));

  }, []);

  consthandleAdopt = async (petId) => {

    const token = localStorage.getItem("token");

    if (!token) {

      alert("Please login first to adopt a pet");

      return;

    }

    try {

      const res = await fetch("http://localhost:5000/api/adoptions", {

        method: "POST",

        headers: {

          "Content-Type": "application/json",

          Authorization: `Bearer ${token}`,

        },

        body: JSON.stringify({ petId }),

      });

    const data = await res.json();

      if (res.ok) {

        setMsg("Pet adopted successfully!");

        setTimeout(() => navigate("/thankyou"), 1000);

      } else {

        setMsg(data.message || "Adoption failed");

      }

    } catch (err) {

      setMsg("Error adopting pet");

    }

  };

  return (

    <div className="pet-container">

      {msg&&<p className="adopt-msg">{msg}</p>}

      <div className="pet-grid">

        {pets.map((pet) => (

          <div key={pet.\_id} className="pet-card">

            <img

              src={`/pet-images/${pet.image}`}

              alt={pet.name}

              className="pet-img"

              onError={(e) => (e.target.src = "/pet-images/default.png")}

            />

            <h3>{pet.name || "Unknown"}</h3>

            <p>Breed: {pet.breed || "Unknown"}</p>

            <p>Age: {pet.age || "Unknown"}</p>

            <button onClick={() =>handleAdopt(pet.\_id)}>Adopt</button>

          </div>

        ))}

      </div>

    </div>

  );

};

export default PetList;

Thamkyou.js

import React from "react";

import { Link } from "react-router-dom";

export default function ThankYou() {

  return (

    <div style={{ textAlign: "center", marginTop: "50px" }}>

      <h2>Thank you for adopting the pet!</h2>

      <p>Take care of your new friend 🐾</p>

      <Link to="/pets">

        <button style={{ padding: "10px 20px", marginTop: "20px", cursor: "pointer" }}>

          Back to Pets

        </button>

      </Link>

    </div>

  );

}

Index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json provides metadata used when your web app is installed on a

      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>

Services/api.js

import axios from "axios";

const API = axios.create({

  baseURL: "http://localhost:5000/api",

});

export default API;

App.js

import React from "react";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import Login from "./components/Login";

import Register from "./components/Register";

import PetList from "./components/PetList";

import ThankYou from "./components/ThankYou";

import Navbar from "./components/Navbar";

function App() {

  return (

    <Router>

      <Navbar />

      <Routes>

        <Route path="/" element={<Login />} />

        <Route path="/register" element={<Register />} />

        <Route path="/pets" element={<PetList />} />

        <Route path="/thankyou" element={<ThankYou />} />

      </Routes>

    </Router>

  );

}

export default App;

package.json

{

  "name": "pet-adoption-frontend",

  "version": "1.0.0",

  "private": true,

  "dependencies": {

    "axios": "^1.7.2",

    "react": "^18.2.0",

    "react-dom": "^18.2.0",

    "react-router-dom": "^6.23.0",

    "react-scripts": "5.0.1",

    "web-vitals": "^5.1.0"

  },

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": [

      "react-app",

      "react-app/jest"

    ]

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "notop\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

  }

}

**Screenshots:**

Figure 1.1: Registeration Page

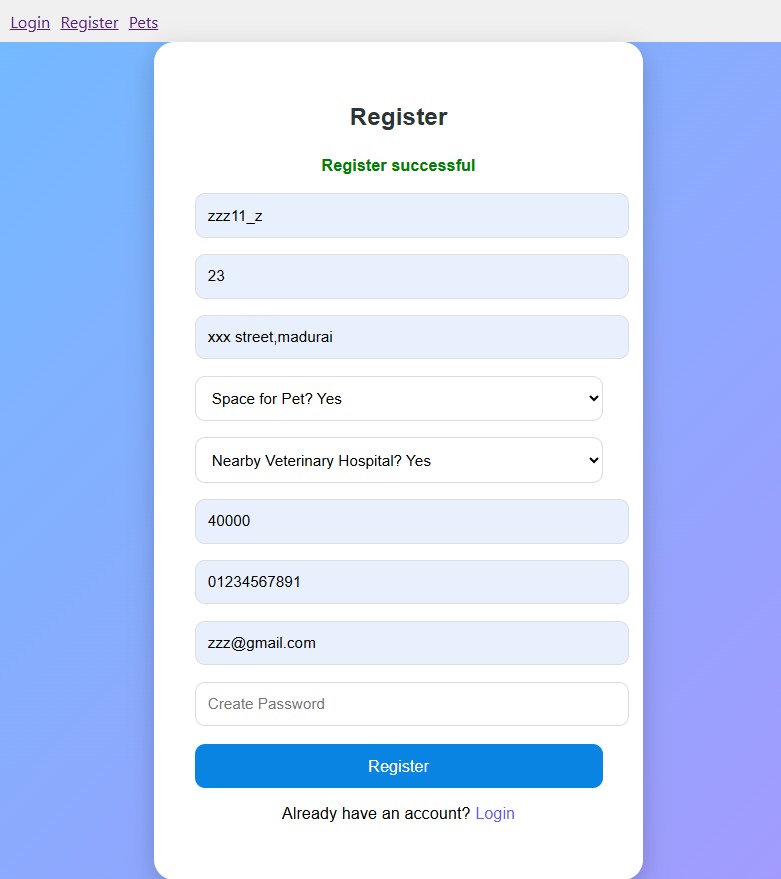


Figure 1.2: Login Page

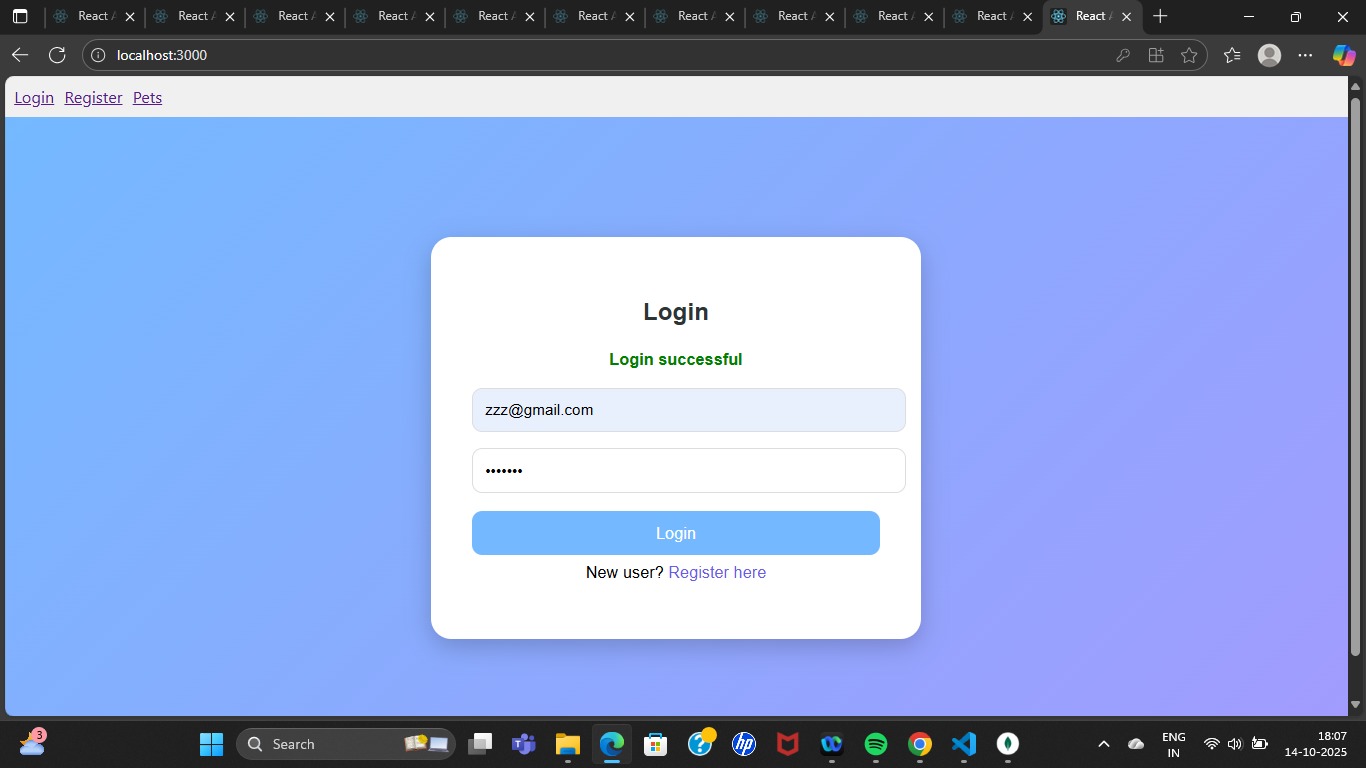


Figure 1.3: Pets List Page to Adopt

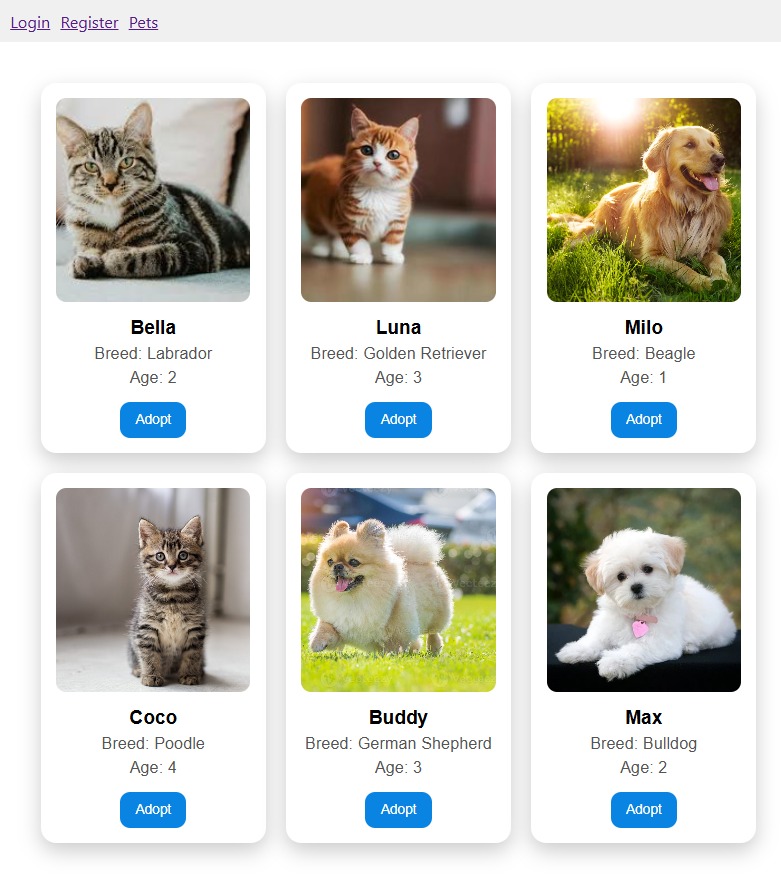


Figure 1.4: Adoption Successful Page

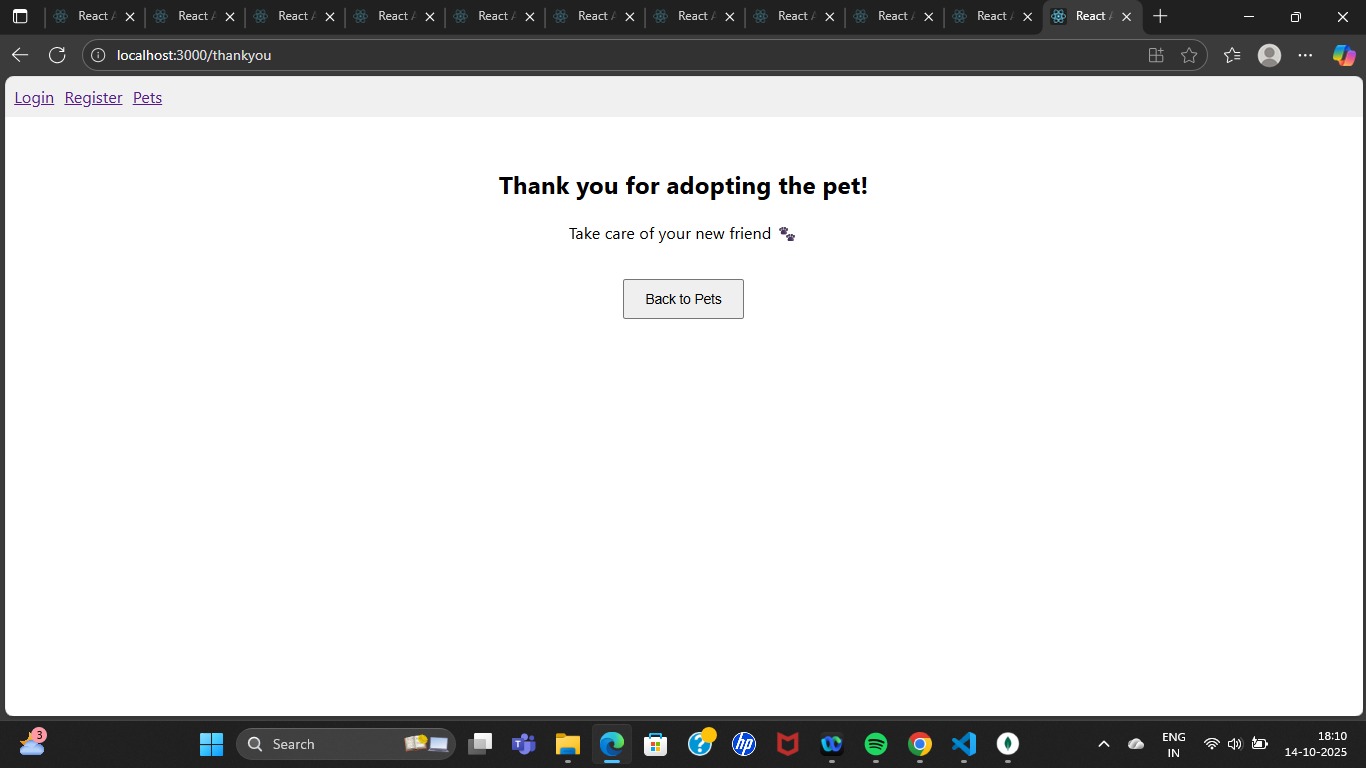


Figure 1.5: User details Database

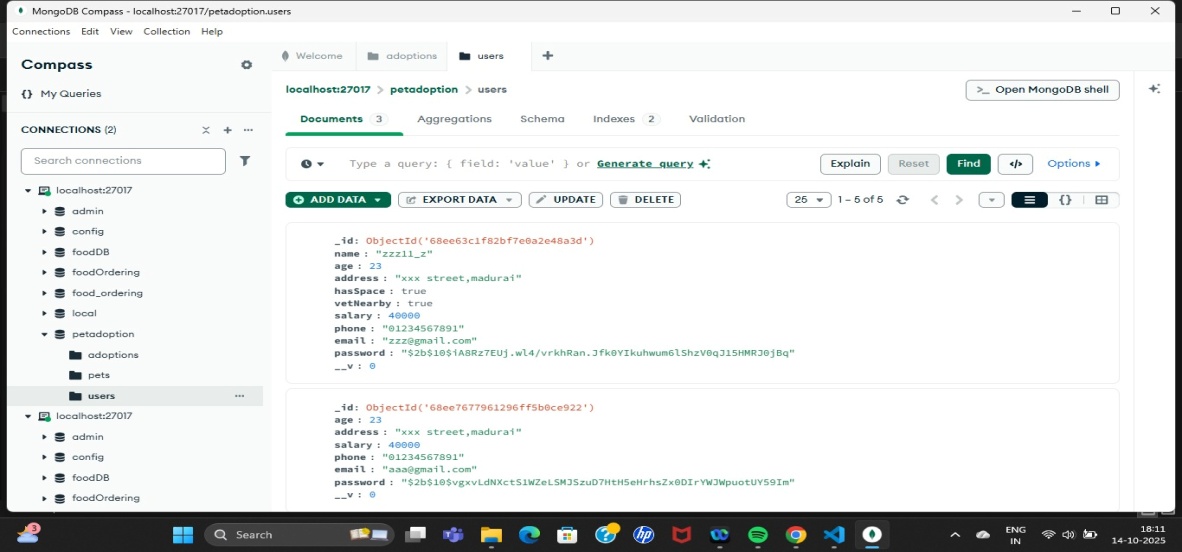


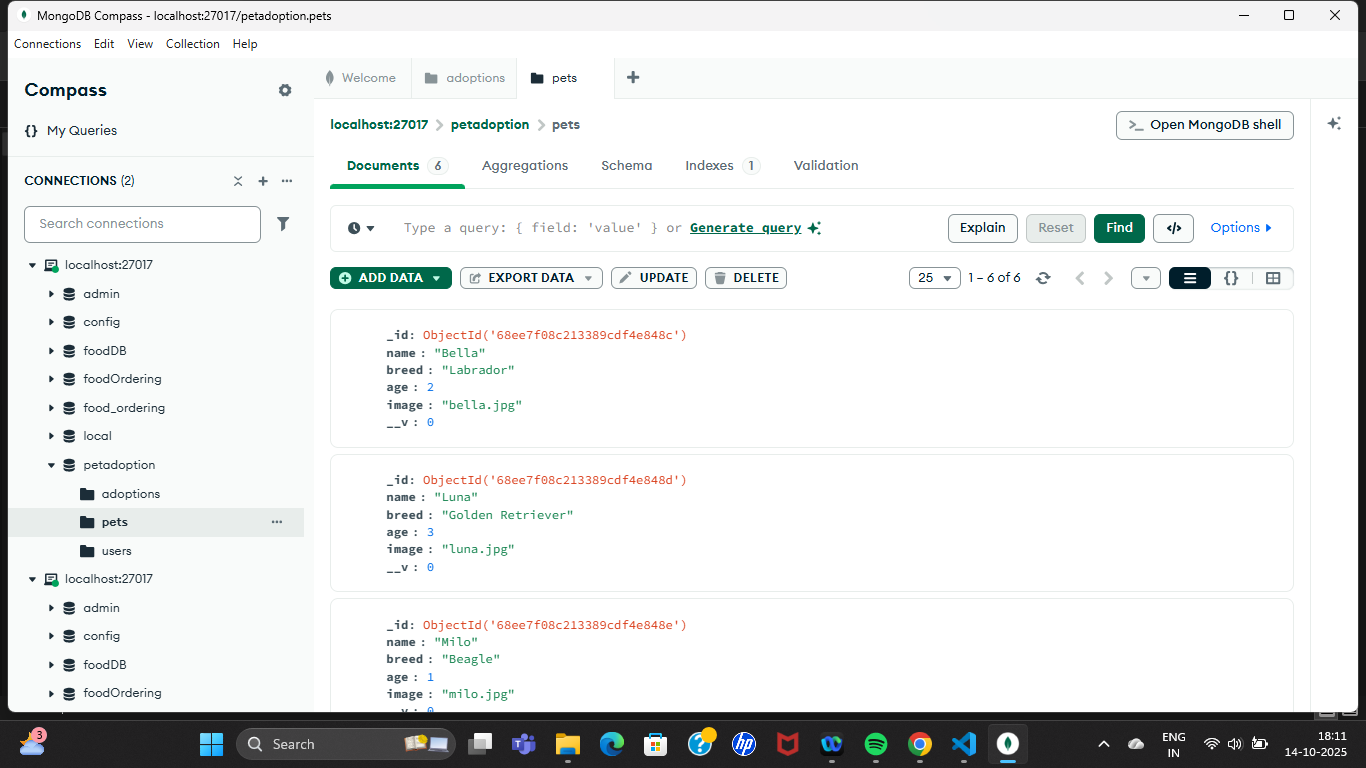
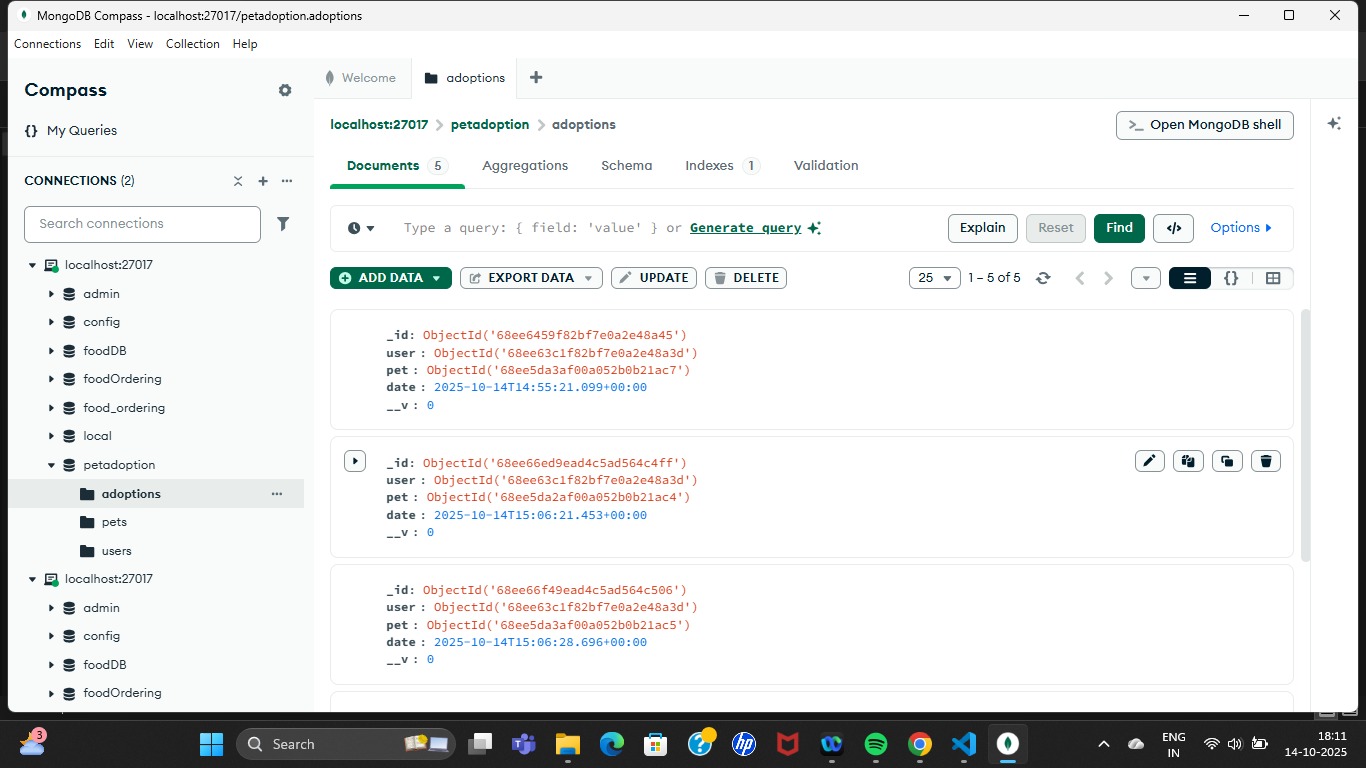
Figure 1.6: Pets Database

Figure 1.7: Adoption Database



**Conclusion:**

The Pet Adoption System successfully provides a streamlined and user-friendly platform to connect pets in need of homes with potential adopters. By digitizing the adoption process, it enhances the visibility of animals available for adoption and simplifies communication between shelters and users. This system not only helps reduce the time pets spend in shelters but also promotes responsible pet ownership and awareness about adoption. Overall, it contributes positively to animal welfare by making pet adoption more accessible and efficient.

|  |  |
| --- | --- |
| **User Interface [20]** |  |
| **Compilation [10]** |  |
| **Client Side and Server Side Validation[15]** |  |
| **Database Integration[15]** |  |
| **Viva [20]** |  |
| **Record Submission [20]** |  |
| **Total [100]** |  |