PokéWorld!

Introduction

The website that I built is titled, PokéWorld, and is an interactive web based Pokémon game. The idea originated from my childhood memories of playing Flash games online and playing Pokémon: Soul silver on my Nintendo DS. Unfortunately, Adobe Flash had shut down, and most games shifted from being web applications to downloadable files. Now, my goal was to recreate the feeling of playing a fun version of Pokémon, with consideration of the first three generations, and allow users to create an account, select different regions and starter Pokémon, and try to capture all of them to become the very best trainer!

Objective

Build a complete MERN stack project which utilizes MongoDB Cloud, Node, Express and React to create a unique Pokémon game.

Users should be able to login and view an interactive selection form for regions and starters. Upon selecting both, they should reach the game dashboard, wherein they can go through an interactive Pokédex to see all available Pokémon, be able to encounter wild Pokémon at random and try to catch them, and view their profile, number of Pokémon they have collected, and the option to log out or delete their account. The website should have JavaScript based interactivity and event handling, use Express in Node with a MongoDB Cloud connection to access, query through, and update various collections of data, use reusable React components with a UI that is extremely responsive to varying window sizes, use different types of Bootstrap components, and have a stylish modern design.

Hardware and Software Requirements

Hardware

- Intel Core i5 or AMD Ryzen 5 series processor is recommended to handle the simultaneous execution of various tasks during development.
- Minimum of 8 GB of RAM is recommended to ensure responsiveness and stability while running the development environment and database server concurrently.

- SSD Storage minimum 4 GB (for all packages installation)
- Stable broadband connection (using MongoDB Cloud, and if working with APIs)

Software

- Windows operating system was used, macOS and Linux distributions also exist
- Visual Studio Code IDE with extension plugins for JavaScript, MongoDB, and Node.js (other popular IDEs include Atom and Sublime Text)
- Installation of Node.js, npm (Node Packet Manager), and Express
- Installation of React and Mongoose
- MongoDB Atlas (cloud based service), local MongoDB Compass can be used
- Web browser (I've used Microsoft Edge, Chrome and Firefox can also be used)
- Middleware (CORs and Body Parser)

Code Snippets

Backend (Port: 4000)

Two collections: Users and Records

Recordschema.js

```
// Import the mongoose module to interact with MongoDB
const mongoose = require("mongoose");

// Define a schema for user sign-up data
// The schema specifies the expected structure and data types for documents in
a MongoDB collection
const recordTemplate = new mongoose.Schema({
    ID: Number,
    Name: String,
    Type: String,
    PixelImg: String,
    FirstImg: String,
    SecondImg: String,
    Description: String,
    WeakAgainst: String,
    StrongAgainst: String,
});


const collection2 = mongoose.model('record', recordTemplate);

module.exports = collection2;
```

Signupschema.js

```
// Import the mongoose module to interact with MongoDB
const mongoose = require("mongoose");

// Define a schema for user sign-up data
// The schema specifies the expected structure and data types for documents in
a MongoDB collection
const signupTemplate = new mongoose.Schema({
    uid: String,
    email: String,
    password: String,
    selectedRegion: String,
    selectedStarter: String,
    pokemonCollected: {
        type: Array,
        of: [String]
    }
});

// Create a model from the schema
// A model allows you to create instances of your documents, called documents
// 'data' is the name of the collection in the MongoDB cloud database
const collection = mongoose.model('user', signupTemplate);

// Export the model
// This makes the model available to our server.js file
module.exports = collection;
```

server.js

```
// Import required modules
const express = require("express");
const bodyParser = require("body-parser");
const mongoose = require("mongoose");
const session = require("express-session");
const cors = require("cors"); // Import cors package

// Create an Express application
const app = express();

// Connect to MongoDB cloud at the login database
mongoose.connect("mongodb+srv://sanskarjadhav:***********@projects.0wab4zm.mon
godb.net/pokemon?retryWrites=true&w=majority&appName=Projects");

// Import the MongoDB model for handling sign up data
const collection = require('./model/Signupschema');
const Record = require('./model/Recordshema');
```

```
// Configure body-parser middleware to parse JSON and URL encoded data
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// Use cors middleware to enable CORS
app.use(cors());

// POST endpoint for both sign in and sign up
app.post('/login', (req, res) => {
    // Determine if it's a sign in or sign up request based on the request
body
    if (req.body.action === 'signin') {
        collection.findOne({ uid: req.body.uid })
        .then((result) => {
            if (!result) {
                // If no document is found, respond with 'No user found'
                return res.status(404).json({ message: 'User is not
registered' })
            } else {
                // Check if the password in the database matches the one
provided in the request
                let validPassword = result.password === req.body.password;
                if (validPassword) {
                    // returns OK response if login is successful
                    return res.status(200).json({ message: 'Login successful'
});
                } else {
                    // If the password is invalid, respond with 'Password is
incorrect'
                    return res.status(404).json({ message: 'Password is
incorrect' })
                }
            }
        })
        .catch(err => {
            // If server error, display error
            res.status(500).json("Error in signing in: " + err);
        });
    } else if (req.body.action === 'signup') {
        collection.findOne({ uid: req.body.uid })
        .then((result) => {
            if (!result) {
                // If the username doesn't exist, proceed with sign-up
                const data = {
                    uid: req.body.uid,
                    email: req.body.email,
                    password: req.body.password,
                };
```

```
                // Use Mongoose to create a new document in the database
                collection.create([data])
                .then(() => res.status(200).json("Successfully Signed Up"))
                .catch(err => res.status(500).json("Error signing up: " +
err));
                return;
            }
            else{// else if username is taken, response is Conflict
                return res.status(409).json("Username is taken");
            }
        })
        .catch(err => {
            // On error, respond with an error message
            res.status(500).json("Error checking username: " + err);
        });
    } else {
        res.status(400).json({ message: 'Invalid request' });
    }
});

// Route for saving selected region
app.post('/save-region', async (req, res) => {
    const {username, region} = req.body;
    try {
        await collection.findOneAndUpdate({ uid: username }, { $set: {
selectedRegion: region } }, { upsert: true });
        // send a success response
        res.status(200).json({ success: true });
    } catch (error) {
        console.error('Error saving region:', error);
        res.status(500).json({ error: 'Failed to save region' });
    }
});

app.post('/save-starter', async (req, res) => {
    const {username, starter} = req.body;
    try{
        await collection.findOneAndUpdate({ uid: username }, { $set : {
selectedStarter: starter } }, { upsert: true} );
        res.status(200).json({success: true});
    } catch (error) {
        console.error('Error saving starter:', error);
        res.status(500).json({error: 'Failed to save starter' });
    }
});

app.get('/records', async (req, res) => {
    try {
```

```
      const records = await Record.find();
      res.json(records);
    } catch (err) {
      res.status(500).send(err.message);
    }
  });

app.get('/random', async (req, res) => {
    try {
      const records = await Record.find();
      // Select a random Pokémon from the array
      const randomPokemon = records[Math.floor(Math.random() *
records.length)];
      res.json(randomPokemon);
    } catch (error) {
      console.error('Error fetching random Pokémon:', error);
      res.status(500).json({ error: 'Internal server error' });
    }
});

app.get('/starter-pokemon', async (req, res) => {
    try {
      const starterPokemonName = req.query.name.toLowerCase();; // Get the
starter Pokémon name from the query parameters
      const starterPokemon = await Record.findOne({ Name: starterPokemonName
});

      if (!starterPokemon) {
        return res.status(404).json({ error: 'Starter Pokémon not found' });
      }

      res.json(starterPokemon); // Return the starter Pokémon document as JSON
    } catch (error) {
      console.error('Error fetching starter Pokémon:', error);
      res.status(500).json({ error: 'Internal server error' });
    }
});

app.post('/catch-pokemon', async (req, res) => {
    const { username, caughtPokemonName } = req.body;
    try {
        // ensuring caughtPokemonName is a string
        const pokemonName = String(caughtPokemonName);
        let user = await collection.findOneAndUpdate(
            { uid: username },
            { $addToSet: { pokemonCollected: pokemonName } },
            // avoiding duplicates
        );
```

```
        if (user) {
            res.status(200).json({ success: true, message: 'Pokemon added
successfully.' });
        } else {
            res.status(404).json({ success: false, message: 'User not found.'
});
        }
    } catch (error) {
        console.error('Error:', error);
        res.status(500).json({ success: false, message: 'Internal server
error.' });
    }
});

app.get('/users', async (req, res) => {
    const username = req.query.name;

    try {
      const user = await collection.findOne({ uid: username });
      if (!user) {
        return res.status(404).json({ message: 'User not found' });
      }
      res.status(200).json({ pokemonCollected: user.pokemonCollected });
    } catch (error) {
      console.error('Error fetching pokemonCollected data:', error);
      res.status(500).json({ message: 'Server error' });
    }
});

app.delete('/delete/:username', async (req, res) => {
    const username = req.params.username;
    try {
      // Delete the user document from the collection
      const result = await collection.deleteOne({ uid: username });
      if (result.deletedCount === 1) {
        // User account deleted successfully
        res.status(200).json({ success: true, message: 'User account deleted
successfully.' });
      } else {
        // User account not found
        res.status(404).json({ success: false, message: 'User account not
found.' });
      }
    } catch (error) {
      console.error('Error deleting user account:', error);
      res.status(500).json({ success: false, message: 'Internal server error.'
});
    }
```

```
});
```

Frontend (Port: 3000)

App.js

```
import 'bootstrap/dist/css/bootstrap.min.css';
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Home from './components/home';
import Login from './components/login';
import Regions from './components/regions';
import Starters from './components/starters';
import Play from './components/play';
import './App.css';

function App() {
  return (
    <div className="App">
    <Router>
      <Routes>
        <Route path="/" exct Component={Home} />
        <Route path="/Regions" exct Component={Regions} />
        <Route path="/Login" exct Component={Login} />
        <Route path="/Starters" exct Component={Starters} />
        <Route path="/Play" exct Component={Play} />
      </Routes>
    </Router>
    </div>
  );
}

export default App;
```

App.css

```
button {
  width: 100%;
  height: 45px;
  background-color: #ffff00;
  border: none;
  outline: none;
  border-radius: 40px;
  cursor: pointer;
  font-size: 1em;
  color: navy;
  font-weight: 500;
}
button:hover{
  /* invert colours upon hover */
```

```
  background-color: navy;
  color: #ffff00;
}
```

home.js

```
import React from "react";
import logo from "../assets/logo.png";
import Navbar from './navbar';
import "../styles/home.css";

function Home(){
    return (
        <div>
        <Navbar/>
        <div className="home">
            <div className="hometop">
                <h1>Venture into the World of Pokémon</h1>
            </div>
            <div className="homediv2">
                <div className="leftSide">
                    <img src={logo} alt="logo"/>
                </div>
                <div className="rightSide">
                    <div className="rightText">
                    <h3>Become the Ultimate Trainer & Collect All
Pokémon!</h3>
                    <h5>Travel to different regions, encounter wild pokémon,
and catch them to complete your pokédex!</h5>
                    </div>
                </div>
            </div>
        </div>
        </div>
    );
}

export default Home;
```

home.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700
;800;900&display=swap');
* {
    font-family: 'Poppins', sans-serif;
}
.hometop {
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    /* span across entire viewport size */
    background-image: url("../assets/pokemonhouse.jpg");
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
}
.hometop h1 {
    margin-top: 5%;
    font-size: 4em;
    font-weight: bold;
    text-align: center;
    color: whitesmoke;
    text-shadow: 0px 5px 15px black;
}

.homediv2 {
    height: 90vh;
    width: 100%;
    display: flex;
    flex-direction: row;
    background-image: url(../assets/background2.jpg);
    background-repeat: no-repeat;
    background-size: cover;
    background-position: center;
}
.homediv2 .leftSide {
    display: flex;
    width: 50%;
    height: 100%;
    justify-content: center; /* Center horizontally */
    align-items: center; /* Center vertically */
}
.homediv2 img {
    object-fit: contain;
    max-width: 90%;
    max-height: 90%;
    width: auto;
    height: auto;
}
.homediv2 .rightSide {
    display: flex;
    width: 50%;
    padding-right: 5%;
    align-items: center;
```

```css
}
.homediv2 h3, h5 {
    text-align: center;
    font-weight: bold;
    padding: 8% 0;
    text-shadow: 0 4px 5px black;
}
.homediv2 h3 {
    color: gold;
    font-size: 6dvh;
}
.homediv2 h5 {
    color: whitesmoke;
    font-size: 4dvh;
}
```

login.js

```js
import React, { useState } from 'react';
import "../styles/login.css";
import img from "../assets/pokemonstarters.png";
import Navbar from './navbar';
import { useNavigate } from 'react-router-dom';

function Login() {
    const [isLoginFormVisible, setIsLoginFormVisible] = useState(true);
    const [password, setPassword] = useState('');
    const [passwordError, setPasswordError] = useState('');
    const navigate = useNavigate();

    const toggleForm = () => {
        setIsLoginFormVisible(!isLoginFormVisible);
    };

    const handlePasswordChange = (e) => {
        const newPassword = e.target.value;
        setPassword(newPassword);

        if (!isLoginFormVisible && newPassword.length < 8) {
            setPasswordError('*Password must be at least 8 characters long');
        } else {
            setPasswordError('');
        }
    };

    // For sign in
    const handleSignInSubmit = (event) => {
        event.preventDefault();
```

```
        const formData = new FormData(event.target);
        const uid = formData.get('username');
        const password = formData.get('password');
        fetch('http://localhost:4000/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ action: 'signin', uid: uid, password:
password }),
        })
        .then(response => {
            if (!response.ok) {
                throw response;
            }
            // save username to local storage in browser
            localStorage.setItem('username', uid);
            const isregion = localStorage.getItem('region');
            const isstarter = localStorage.getItem('starter');
            // if user had logged in before, take them directly to play
            // mimics real websites
            if (isstarter) {
                navigate('/Play');
            }
            else if (isregion) {
                navigate('/Starters');
            }
            else {
                navigate('/Regions');
            }
        })
        .catch(error => {
            // Display error message based on server response
            error.json().then(data => {
                showBootstrapAlert(data.message, 'danger');
            });
        });
    };

    // For sign up
    const handleSignUpSubmit = (event) => {
        if (passwordError !== '') {
            event.preventDefault(); // Prevent form submission if there are
errors
            return; // Exit early if password is too short
        }
        event.preventDefault();
        const formData = new FormData(event.target);
```

```
        const uid = formData.get('username');
        const email = formData.get('email');
        const password = formData.get('password');
        fetch('http://localhost:4000/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ action: 'signup', uid: uid, email: email,
password: password }),
        })
        .then(response => response.json())
        .then(data => {
            // Check if the response indicates success or failure
            if (data === 'Successfully Signed Up') {
                // Display success message using Bootstrap alert
                showBootstrapAlert(data, 'success');
            } else {
                // Display error message for username already taken
                showBootstrapAlert(data, 'danger');
            }
        })
        .catch(error => {
            // Display error message for network or server error
            showBootstrapAlert('An error occurred. Please try again later.',
'danger');
        });
    };

    // Function to show Bootstrap alert
    const showBootstrapAlert = (message, type) => {
        const alertDiv = document.createElement('div');
        alertDiv.classList.add('alert', `alert-${type}`, 'fixed-top', 'z-
index-alert');
        alertDiv.role = 'alert';
        alertDiv.textContent = message;

        document.body.appendChild(alertDiv);

        setTimeout(() => {
            alertDiv.remove();
        }, 3000); // Remove alert after 3 seconds
    };

    return (
        <div>
        <Navbar/>
        <div className='loginpage'>
```

```
            <div className={`wrapper ${isLoginFormVisible ? '' : 'active'}`}>
                <img src={img} alt="Starter Pokemons"/>
                <div className={`form-wrapper ${isLoginFormVisible ? 'login' :
'register'}`}>
                    <form onSubmit={isLoginFormVisible ? handleSignInSubmit :
handleSignUpSubmit}>
                        <h2>{isLoginFormVisible ? 'Login' : 'Sign Up'}</h2>
                        <div className="input-box">
                            <input type="text" name="username"
placeholder="Username" required />
                        </div>
                        {!isLoginFormVisible && (
                            <div className="input-box">
                                <input type="email" name="email"
placeholder="Email" required />
                            </div>
                        )}
                        <div className="input-box">
                            <input type="password" name="password"
placeholder="Password" value={password} onChange={handlePasswordChange}
required />
                            {passwordError && <span
className="error">{passwordError}</span>}
                        </div>
                        <button type="submit">{isLoginFormVisible ? 'Continue'
: 'Register'}</button>
                        <div className="sign-link">
                            <p>{isLoginFormVisible ? "Don't have an account? "
: "Already have an account? "}
                                <button className="btn btn-link"
onClick={toggleForm}>
                                    {isLoginFormVisible ? 'Sign Up' : 'Login'}
                                </button>
                            </p>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        </div>
    );
}

export default Login;
```

login.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700
;800;900&display=swap');
* {
    /* to create float effect, set intial margins to zero */
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}
.loginpage {
    display: flex;
    justify-content: center;
    align-items: center;
    /* span across entire viewport size */
    height: 100vh;
    width: 100vw;
    padding-top: 40px;
    background-image: url("../assets/background1.png");
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
}
.wrapper {
    position: absolute;
    width: 60%;
    height: 80%;
    min-width: 300px;
    min-height: 480px;
    /* gradient of blue as background colour */
    background: linear-gradient(90deg, #1c5fb1, #2f0d4f);
    border-radius: 50px;
    /* 3d depth effect using shadow */
    box-shadow: 0 0 30px rgba(0, 0, 0, .5);
    padding: 65px;
    display: flex;
    align-items: center;
    /* image overflows so to limit it to wrapper shape */
    overflow: hidden;
}
.wrapper img {
    position: absolute;
    right: -24px;
    bottom: -18px;
    width: 50%;
}
.form-wrapper {
```

```
    /* form in wrapper appears above */
    z-index: 1;
    width: 40%;
}
.wrapper .form-wrapper.login {
    position: absolute;
    margin-top: 25px;
    /* creates float down effect */
    transition: .5s ease-in-out;
    transition-delay: .2s;
    width: 40%;
}

.wrapper .form-wrapper.register {
    position: absolute;
    margin-top: 15px;
    /* creates float up effect */
    transition: .5s ease-in-out;
    transition-delay: .2s;
    width: 40%;
}

.wrapper .form-wrapper h2 {
    font-size: 2em;
    font-weight: bold;
    text-align: center;
    color: #fff;
}
.input-box {
    position: relative;
    width: 100%;
    /* vertical spacing between input boxes */
    margin: 30px 0;
}
.input-box input {
    width: 100%;
    height: 50px;
    background: transparent;
    border: 2px solid #fff;
    /* removes the default black outline when entering input */
    outline: none;
    border-radius: 40px;
    font-size: 1em;
    color: #fff;
    /* padding on left and right */
    padding: 0 30px 0 30px;
}
.input-box input::-ms-reveal {
```

```
    /* makes password reveal icon white instead of default black*/
    filter: invert(100%);
  }
.input-box input::placeholder {
    /* placeholder text should be lighter blue */
    color: rgba(147, 195, 230, 0.5);
}
.input-box input[type="password"] + .error{
    color: white;
    font-size: 12px;
}

.sign-link {
    font-size: .9em;
    text-align: center;
    margin-top: 25px;
}
.sign-link p {
    color: #fff;
}
.sign-link button {
    color: #fff;
    font-weight: bold;
    text-decoration: none;
    width: fit-content;
    height: fit-content;
    border: none;
    padding: 0.25rem;
}
.sign-link button:hover{
    text-decoration: underline;
    color: #fff;
}
.z-index-alert {
    z-index: 100; /* or any higher value */
    display: inline-block;
    justify-self: center;
    width: fit-content;
}
```

navbar.js

```
import React from "react";
import { Link } from "react-router-dom";
import logo from "../assets/logo.png";
import "../styles/navbar.css";

function Navbar(){
```

```
    return (
        <div className="navbar">
            <div className="leftSide">
                <img src={logo} alt="pokemon_badge"/>
            </div>
            <div className="rightSide">
                <Link to="/">Home</Link>
                <Link to="/Login">Login</Link>
            </div>
        </div>
    );
}

export default Navbar;
```

navbar.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700
;800;900&display=swap');
* {
    font-family: 'Poppins', sans-serif;
}
.navbar{
    position: fixed;
    width: 100vw;
    height: 40px;
    background-color: rgba(0,0,0,0.7);
    display: flex;
    /* items are placed in a row */
    flex-direction: row;
    /* navbar overlaps on top */
    z-index: 9;
    align-items: center;
}
.navbar .leftSide{
    flex: 20%;
    justify-content: left;
    padding-left: 30px;
}
.navbar img{
    width: 25px;
}
.navbar .rightSide{
    justify-content: right;
    padding-right: 40px;
}
.navbar a{
```

```
    color: white;
    font-weight: 500;
    /* separate the hyperlinks with some space */
    padding-left: 30px;
    text-decoration: none;
}
.navbar a:hover{
    text-decoration: underline;
}
```

navbar2.js

```
import React from "react";
import { Link } from "react-router-dom";
import logo from "../assets/logo.png";
import "../styles/navbar.css";

function Navbar2(){
    const handleLogout = () => {
        // remove username from localStorage
        localStorage.removeItem('username');
    };

    return (
        <div className="navbar">
            <div className="leftSide">
                <img src={logo} alt="pokemon_badge"/>
            </div>
            <div className="rightSide">
                <Link to="/Regions">Change Region</Link>
                <Link to="/Starters">Change Starter</Link>
                <Link to="/Login" onClick={handleLogout}>Logout</Link>
            </div>
        </div>
    );
}

export default Navbar2;
```

regions.js

```
import React, { useState } from 'react';
import hoenn from "../assets/Hoenn.mp4";
import johto from "../assets/Johto.mp4";
import kanto from "../assets/Kanto.mp4";
import "../styles/regions.css";
import { useNavigate } from 'react-router-dom';
```

```
function Regions() {
  const [activeSlide, setActiveSlide] = useState(0);
  const navigate = useNavigate();
  const username = localStorage.getItem('username');

  const sliderNav = (direction) => {
    if (direction === 'prev') {
      setActiveSlide(activeSlide === 0 ? 2 : activeSlide - 1);
    } else {
      setActiveSlide(activeSlide === 2 ? 0 : activeSlide + 1);
    }
  };

  const selectRegion = (region) => {
    // Send selected region to backend
    fetch('http://localhost:4000/save-region', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ username, region })
    })
      .then(response => {
        if (!response.ok) {
          throw new Error('Failed to save region');
        }
        // save region chosen to local browser storage
        localStorage.setItem('region', region);
        // redirect to Starters page
        navigate('/Starters');
      })
      .catch(error => {
        console.error('Error saving region:', error);
      });
  };

  return (
    <div className="region">
      <video className={activeSlide === 0 ? "videobox active" : "videobox"}
src={kanto} autoPlay muted loop></video>
      <video className={activeSlide === 1 ? "videobox active" : "videobox"}
src={johto} autoPlay muted loop></video>
      <video className={activeSlide === 2 ? "videobox active" : "videobox"}
src={hoenn} autoPlay muted loop></video>

      <div className={activeSlide === 0 ? "content active" : "content"}>
        <h1>Kanto</h1>
```

```
        <p>Gen I Pokémon</p>
        <button onClick={() => selectRegion('Kanto')}>Select this
region</button>
      </div>

      <div className={activeSlide === 1 ? "content active" : "content"}>
        <h1>Johto</h1>
        <p>Gen II Pokémon</p>
        <button onClick={() => selectRegion('Johto')}>Select this
region</button>
      </div>

      <div className={activeSlide === 2 ? "content active" : "content"}>
        <h1>Hoenn</h1>
        <p>Gen III Pokémon</p>
        <button onClick={() => selectRegion('Hoenn')}>Select this
region</button>
      </div>

      <div className="nav-arrows">
        <div className="arrow left" onClick={() =>
sliderNav('prev')}>&#8249;</div>
        <div className="arrow right" onClick={() =>
sliderNav('next')}>&#8250;</div>
      </div>
    </div>
  );
}

export default Regions;
```

regions.css

```
.region {
    position: relative; /* Changed to relative */
    height: 100vh;
    justify-content: center;
    width: auto;
    display: flex;
    flex-direction: column;
    background-image: url("../assets/ambientbg.jpg");
    background-size: cover;
  }

  .content {
    z-index: 3;
    color: #fff;
    display: none;
```

```css
}

.content.active {
  display: flex;
  flex-direction: column;
  text-align: center;
  left: 50%;
  bottom: 5%;
  transform: translateX(-50%);
  position: absolute;
}

.content.active h1 {
  color: gold;
  text-transform: uppercase;
  font-weight: 900;
  font-size: 4em;
  text-shadow: 0 0 15px goldenrod;
  -webkit-text-stroke: 3px blue;
}

.content.active p {
  font-size: 1.2em;
  font-weight: bold;
  text-shadow: 0 0 5px black;
}

.content button {
  background-color: #fff;
  padding: auto 35px;
  color: blue;
  font-size: 1.1em;
  font-weight: 500;
  width: 270px;
  text-decoration: none;
  border-radius: 10px;
}

.content button:hover {
  background-color: gold;
  color:#fff;
}

.video {
  z-index: 1;
  position: relative;
  width: 100%;
  object-fit: cover;
```

```css
}

.videobox {
  position: absolute;
  transform: translateY(-10%);
  width: 60%;
  max-height: 65%;
  left: 20%; /* Center video horizontally */
  clip-path: polygon(0% 0%, 0% 0%, 0% 100%, 0% 100%);
}

.videobox.active {
  clip-path: polygon(0% 0%, 100% 0%, 100% 100%, 0% 100%);
  transition: 0.9s ease;
  transition-property: clip-path;
}

.nav-arrows {
  position: fixed;
  z-index: 4;
  width: 100%;
  display: flex;
  justify-content: space-between;
}

.arrow {
  color: white;
  font-size: 15vw;
  padding: 0% 8%;
  cursor: pointer;
}

.arrow:hover {
  color: gold;
}
```

*pokemoncard.js (**Reusable component**)*

```jsx
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';

const PokemonCard = ({ type, imageUrl, pokemon, hoverImage, info}) => {
    const [showModal, setShowModal] = useState(false);
    const navigate = useNavigate();
    const username = localStorage.getItem('username');
    // extracting the 'pokemon' prop as a constant
    const pokemonName = pokemon;
```

```
    const openModal = () => {
        setShowModal(true);
    };

    const closeModal = () => {
        setShowModal(false);
    };

    const selectStarter = (starter) => {
        // Send selected region to backend
        fetch('http://localhost:4000/save-starter', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({ username, starter })
        })
          .then(response => {
            if (!response.ok) {
              throw new Error('Failed to save starter pokemon');
            }
            // save region chosen to local browser storage
            localStorage.setItem('starter', starter);
            // redirect to Starters page
            navigate('/Play');
          })
          .catch(error => {
            console.error('Error saving starter:', error);
          });
      };

    return (
        <div className="pokemon-card">
            <div className="card">
                <div className="card-inner">
                    <div className="card-front">
                        <img src={imageUrl} className="card-img-top"
alt={type} />
                        <h4 className="card-title">Hover to reveal!</h4>
                    </div>
                    <div className="card-back">
                        <img src={hoverImage} className="card-img-top"
alt={pokemon} />
                        <h5 className="card-title">{pokemon}</h5>
                        <button className="btn btn-secondary" onClick={() =>
selectStarter(pokemonName)}>
                            I Choose You!
```

```
                        </button>
                    </div>
                </div>
            </div>
            <button className="btn btn-primary" onClick={openModal}>
                Learn More
            </button>
            {showModal && (
                <div className="modal">
                    <div className="modal-content">
                        <span className="close"
onClick={closeModal}>&times;</span>
                        <div>
                            <h2>{type}</h2>
                            <p>{info}</p>
                        </div>
                    </div>
                </div>
            )}
        </div>
    );
};

export default PokemonCard;
```

pokemoncard.css

```
.pokemon-card {
    display: flex;
    flex-direction: column;
    width: 250px;
    margin: auto;
    justify-content: center;
    align-items: center;
  }

  .card {
    width: 100%;
    height: 370px;
    justify-content: center;
    display: flex;
    background-color: rgba(25, 25, 25, 0.6);
    border: 3px solid rgba(255, 255, 255, 0.2);
    perspective: 900px; /* defines the depth of 3D space */
  }

  .card-inner {
```

```css
  width: 100%;
  height: 100%;
  position: relative;
  transition: transform 0.6s;
  transform-style: preserve-3d;
  display: flex;
  align-items: center;
}

.card:hover .card-inner {
  transform: rotateY(-180deg);
}

.card-front, .card-back {
  width: 100%;
  height: 100%;
  align-items: center;
  text-align: center;
  color: whitesmoke;
  text-shadow: 0 0 6px black;
  backface-visibility: hidden;
  position: absolute;
}

.card-front img {
  padding-top: 10%;
}

.card-inner h4 {
  padding: 20% 0;
}

.card-back {
  transform: rotateY(180deg);
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

.btn-primary {
  margin: 15px 0;
  color: black;
  font-weight: bold;
  background-color: gold;
  border: 2px dashed white;
}
```

```css
.btn-primary:hover,
.btn-primary:focus {
  background-color: black !important; /* important to override bootstrap default blue */
  color: whitesmoke;
  border: 2px solid white !important;
}

.btn-secondary {
  color: gold;
  font-family: 'Segoe UI', Tahoma, Verdana, sans-serif;
  background-color: rgba(0,0,0,0.4) !important;
  position: inherit;
  bottom: 0;
}

.modal {
  display: flex;
  justify-content: center;
  align-items: center;
  position: fixed;
  z-index: 3;
  width: 100%;
  padding: 0 25%;
  height: 100%;
  background-color: rgba(0,0,0,0.4);
}

.modal h2 {
  color: black;
  padding-bottom: 10px;
  text-align: center;
}


.modal-content {
  padding: 20px;
  border-radius: 15px;
  color: black;
}

.modal-content p {
  text-align: center;
}

.close {
  color: #aaa;
  font-size: 28px;
```

```
    font-weight: bold;
  }

  .close:hover {
    color: black;
    text-decoration: none;
    cursor: pointer;
  }
```

starters.js

```
// Starters.js
import React from 'react';
// using a resuable component PokemonCard
import PokemonCard from './pokemoncard';
import fireType from '../assets/fire.png';
import waterType from '../assets/water.png';
import grassType from '../assets/grass.png';
import charmander from '../assets/charmander.png';
import squirtle from '../assets/squirtle.png';
import bulbasaur from '../assets/bulbasaur.png';
import cyndaquil from '../assets/cyndaquil.png';
import totodile from '../assets/totodile.png';
import chikorita from '../assets/chikorita.png';
import torchic from '../assets/torchic.png';
import mudkip from '../assets/mudkip.png';
import treecko from '../assets/treecko.png';
import '../styles/pokemoncard.css';
import '../styles/starters.css';

function Starters() {
    const region = localStorage.getItem('region');
    // let variables for Pokemon and hover image
    let pokemon1, hoverImage1, pokemon2, hoverImage2, pokemon3, hoverImage3;
    // switch case to determine which Pokemon and hover image to use based on
region
    switch(region) {
        case 'Kanto':
            pokemon1 = "Charmander";
            hoverImage1 = charmander;
            pokemon2 = "Squirtle";
            hoverImage2 = squirtle;
            pokemon3 = "Bulbasaur";
            hoverImage3 = bulbasaur;
            break;
        case 'Johto':
            pokemon1 = "Cyndaquil";
```

```
            hoverImage1 = cyndaquil;
            pokemon2 = "Totodile";
            hoverImage2 = totodile;
            pokemon3 = "Chikorita";
            hoverImage3 = chikorita;
            break;
        case 'Hoenn':
            pokemon1 = "Torchic";
            hoverImage1 = torchic;
            pokemon2 = "Mudkip";
            hoverImage2 = mudkip;
            pokemon3 = "Treecko";
            hoverImage3 = treecko;
            break;
        default:
            // Default to Kanto Pokemon and hover image if region is not
recognized
            pokemon1 = "Charmander";
            hoverImage1 = charmander;
            pokemon2 = "Squirtle";
            hoverImage2 = squirtle;
            pokemon3 = "Bulbasaur";
            hoverImage3 = bulbasaur;
    }
    return (
        <div className="card-container">
            <h1>Select your Starter!</h1>
            <div className="row">
                <PokemonCard
                    type="Fire"
                    imageUrl={fireType}
                    pokemon={pokemon1}
                    hoverImage={hoverImage1}
                    info="Fire types are notoriously rare in the earlier games
so choosing the Fire variation starter is often a plus!"
                />
                <PokemonCard
                    type="Water"
                    imageUrl={waterType}
                    pokemon={pokemon2}
                    hoverImage={hoverImage2}
                    info="Water is the most common type with over 150 Pokémon
that are all based on a variety of fish and other sea-dwelling creatures!"
                />
                <PokemonCard
                    type="Grass"
                    imageUrl={grassType}
                    pokemon={pokemon3}
```

```
                    hoverImage={hoverImage3}
                    info="Grass is one of the weakest types statistically, so
select a Grass type starter if you are looking for a real challenge!"
                />
            </div>
        </div>
    );
}

export default Starters;
```

starters.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700
;800;900&display=swap');
* {
    /* to create float effect, set intial margins to zero */
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}
.card-container {
    height: 100vh;
    width: 100%;
    display: flex;
    flex-direction: column;
    align-items: center;
    padding-top: 10vh;
    background-image: url("../assets/background2.jpg");
    background-size: cover;
}

.card-container h1 {
    margin-top: 10px;
    font-size: 2em;
    font-weight: bold;
    text-shadow: 0 0 8px black;
    color: #fff;
}

.row {
    margin-top: 10px; /* Add some space above the cards */
    display: flex;
    align-items: center;
    justify-content: center;
}
```

play.js

```
import React, { useState } from 'react';
import Pokedex from './pokedex';
import Forest from './forest';
import Profile from './profile';
import Navbar2 from './navbar2';
import '../styles/play.css';

function Play() {
    const [activeOption, setActiveOption] = useState(null);
    const [isPokedexOpen, setIsPokedexOpen] = useState(false);
    const [isForestOpen, setIsForestOpen] = useState(true);
    const [isProfileOpen, setIsProfileOpen] = useState(false);

    const handleOptionClick = (option) => {
        setActiveOption(option);
        setIsPokedexOpen(option === 'pokedex');
        setIsForestOpen(option === 'forest');
        setIsProfileOpen(option === 'profile');
    }

    const renderActiveComponent = () => {
        switch (activeOption) {
            case 'pokedex':
                return <Pokedex isOpen={isPokedexOpen} handleClose={() =>
setIsPokedexOpen(false)} />; // Pass isOpen and handleClose as props
            case 'forest':
                return <Forest isOpen={isForestOpen} handleClose={() =>
setIsForestOpen(false)} />; // Pass isOpen and handleClose as props
            case 'profile':
                return <Profile isOpen={isProfileOpen} handleClose={() =>
setIsProfileOpen(false)} />;
            default:
                return null;
        }
    }

    return (
        <div>
        <Navbar2/>
        <div className='visual'>
            <h1>Begin Your Pokémon Adventure!</h1>
            <div className="options">
                <button onClick={() => handleOptionClick('pokedex')}>View
Pokédex</button>
                <button onClick={() => handleOptionClick('forest')}>Explore
the Forest</button>
```

```
                <button onClick={() => handleOptionClick('profile')}>View
Profile</button>
            </div>
            <div className="play">
                {renderActiveComponent()}
            </div>
        </div>
        </div>
    );
}

export default Play;
```

play.css

```
.visual {
    display: flex;
    flex-direction: column;
    height: 100vh;
    width: 100%;
    background-image: url('../assets/forestbg.png');
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center;
    text-align: center;
    align-items: center;
    justify-content: center;
}

.visual h1 {
    position: absolute;
    top: 50px;
    font-size: 1.4em;
    font-weight: bold;
    text-align: center;
    color: whitesmoke;
    text-shadow: 0px 5px 15px black;
}

.play {
    position: relative;
    width: 60%;
}

.options {
    position: absolute;
    bottom: 5%;
    width: 100%;
```

```
    display: flex;
    flex-direction: row;
}

.options button {
    width: 25%;
    margin: 0 auto;
}
```

pokedex.js

```
import React, { useState, useEffect } from 'react';
import '../styles/pokedex.css';

const Pokedex = ({ isOpen, handleClose }) => { // Pass isOpen and handleClose
as props
  const [searchQuery, setSearchQuery] = useState('');
  const [pokemonList, setPokemonList] = useState([]);
  const [selectedPokemon, setSelectedPokemon] = useState(null); // State to
track the selected Pokemon

  useEffect(() => {
    fetchData();
  }, []);

  const fetchData = async () => {
    try {
      const response = await fetch('http://localhost:4000/records');
      const data = await response.json();
      setPokemonList(data);
    } catch (error) {
      console.error('Error fetching data:', error);
    }
  };

  const handleSearch = (e) => {
    setSearchQuery(e.target.value);
  };

  const handlePokemonClick = (pokemon) => {
    setSelectedPokemon(pokemon);
  };

  const filteredPokemonList = pokemonList.filter((pokemon) =>
    pokemon.Name.toLowerCase().includes(searchQuery.toLowerCase())
  );

  return (
```

```
    <div className={`pokedex-container ${isOpen ? 'open' : 'closed'}`}>
      <button className="close-btn" onClick={handleClose}>x</button>
      <input
        className="search-bar"
        type="text"
        placeholder="Search Pokémon"
        value={searchQuery}
        onChange={handleSearch}
      />
      <div className="pokemon-list">
        {filteredPokemonList.map((pokemon) => (
          <div key={pokemon._id} className="pokemon-eachcard" onClick={() =>
handlePokemonClick(pokemon)}>
            <img src={pokemon.PixelImg} alt={pokemon.Name} />
            <h3>{pokemon.Name}</h3>
          </div>
        ))}
      </div>
      {selectedPokemon && (
        <div className={`pokemon-details ${selectedPokemon ? 'show' : ''}`}>
          <button className="close-btn" onClick={() =>
setSelectedPokemon(null)}>x</button>
          <img src={selectedPokemon.FirstImg} alt={selectedPokemon.Name} />
          <h2>{selectedPokemon.Name}</h2>
          <p>{selectedPokemon.Description}</p>
          <p><strong>Type:</strong> {selectedPokemon.Type}</p>
          <p><strong>Weak Against:</strong> {selectedPokemon.WeakAgainst}</p>
          <p><strong>Strong Against:</strong>
{selectedPokemon.StrongAgainst}</p>
        </div>
      )}
    </div>
  );
};

export default Pokedex;
```

pokedex.css

```
.pokedex-container {
    max-width: 600px;
    margin: 0 auto;
    padding: 0;
    border: 2px solid #ccc;
    background-color: rgba(0,0,0,0.2);
    border-radius: 10px;
    transition: all 0.3s ease; /* Smooth transition for opening and closing */
    opacity: 1; /* Default to visible */
```

```css
  visibility: visible; /* Default to visible */
}

.pokedex-container.closed {
  opacity: 0; /* invisible when closed */
  visibility: hidden; /* hidden when closed */
  pointer-events: none;
}

.search-bar {
  width: 90%;
  padding: 10px;
  margin-bottom: 20px;
  font-size: 16px;
  border-radius: 10px;
  outline: none !important;
}

.pokemon-list {
  height: 40vh;
  overflow-y: auto;
  overflow-x: hidden; /* Hide horizontal scrollbar */
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
  grid-gap: 10px;
  scrollbar-color: white rgba(0,0,0,0.5); /* For Firefox */
}

.pokemon-eachcard {
  text-align: center;
  cursor: pointer;
}

.pokemon-eachcard img {
  max-width: 100px;
  margin-bottom: 10px;
}

.pokemon-eachcard h3 {
  font-size: 18px;
  color: whitesmoke;
  text-transform: capitalize;
}

.close-btn {
  position: relative;
  top: 0;
  background-color: transparent !important;
```

```
    border: none;
    font-size: 20px; /* Smaller font size */
    cursor: pointer;
    padding: 5px;
    color: #fff !important; /* White color */
    z-index: 1; /* Ensure it's above other elements */
  }

  .close-btn:hover {
    color: #ccc !important;
    background-color: rgba(0, 0, 0, 0.5) !important; /* Lighter color on hover
*/
  }

  .pokemon-details {
    display: none; /* Initially hide the detailed view */
    position: absolute;
    top: 0;
    height: 100%;
    width: 100%;
    border: 3px solid #ccc;
    max-width: 600px;
    background-color: rgba(0, 0, 0, 0.9);
    z-index: 99;
    overflow-y: auto;
    padding: 20px;
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .pokemon-details.show {
    display: block;
  }

  .pokemon-details img {
    display: block;
    margin: 0 auto;
    max-width: 100%;
    max-height: 300px;
  }

  .pokemon-details h2 {
    text-align: center;
    color: #fff;
    text-transform: capitalize;
  }
```

```
  .pokemon-details p {
    color: #fff;
    margin-bottom: 10px;
  }
```

forest.js

```
import React, { useState, useEffect, useCallback } from 'react';
import '../styles/forest.css';

const Forest = ({ isOpen, handleClose }) => {
  const [randomPokemon, setRandomPokemon] = useState(null);
  const [catchProbability, setCatchProbability] = useState(0);
  const [isLoading, setIsLoading] = useState(true); // State to track loading
status

  const fetchRandomPokemon = useCallback(async () => {
    try {
      setIsLoading(true); // Set loading to true before fetching data
      const response = await fetch('http://localhost:4000/random');
      const data = await response.json();
      setRandomPokemon(data);
      calculateCatchProbability(data);
    } catch (error) {
      console.error('Error fetching random Pokémon:', error);
    } finally {
      setIsLoading(false); // Set loading to false after fetching data
    }
  }, []);

  useEffect(() => {
    // fetching a new random Pokémon when the component is opened or closed
    if (isOpen) {
      fetchRandomPokemon();
    }
  }, [isOpen, fetchRandomPokemon]);

  const calculateCatchProbability = async (pokemon) => {
    try {
      // retrieving the user's starter Pokémon name from localStorage
      const starterPokemonName = localStorage.getItem('starter');
      // gettint the type
      const response = await fetch(`http://localhost:4000/starter-
pokemon?name=${starterPokemonName}`);
      if (!response.ok) {
        throw new Error('Failed to fetch starter Pokémon');
      }
      const starterPokemon = await response.json();
```

```
      // Now that I have the starter Pokémon document, I can access its type
      const starterPokemonTypes = starterPokemon.Type.split(','); // Split by
commas to handle multiple types

      // Calculate catch probability based on the starter Pokémon's types
      let probability = 0.5;
      for (const type of starterPokemonTypes) {
        if (pokemon.WeakAgainst.includes(type)) {
          probability = 0.75;
          break;
        }
        if (pokemon.StrongAgainst.includes(type)) {
          probability = 0.25;
          break;
        }
      }

      setCatchProbability(probability);
    } catch (error) {
      console.error('Error calculating catch probability:', error);
    }
  };

  const handleCatch = async () => {
    const catchResult = Math.random() < catchProbability;
    if (catchResult) {
      showBootstrapAlert('Congratulations! You caught the Pokémon!',
'success');
      try {
        const username = localStorage.getItem('username');
        const response = await fetch('http://localhost:4000/catch-pokemon', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({ username, caughtPokemonName:
randomPokemon.Name })
        });
        const data = await response.json();
        if (data.success) {
          console.log('Pokemon collected successfully!');
        } else {
          console.error('Failed to collect Pokemon:', data.error);
        }
      } catch (error) {
        console.error('Error collecting Pokemon:', error);
      }
```

```
    } else {
      showBootstrapAlert('Oh no! The Pokémon got away...', 'secondary');
    }
    handleClose();
  };

  const showBootstrapAlert = (message, type) => {
    const alertDiv = document.createElement('div');
    alertDiv.classList.add('alert', `alert-${type}`, 'fixed-top', 'z-index-
alert');
    alertDiv.style.top = '80px';
    alertDiv.role = 'alert';
    alertDiv.textContent = message;
    document.body.appendChild(alertDiv);

    setTimeout(() => {
        alertDiv.remove();
    }, 1500);
  };

  return (
    <div className={`forest-container ${isOpen ? 'open' : 'closed'}`}>
      <h2>You Encountered a Wild Pokémon!</h2>
      {isLoading ? ( // displays loading spinner if data is loading
        <div className="loading-spinner"></div>
      ) : (
        randomPokemon && (
          <div className="random-pokemon">
            <img src={randomPokemon.SecondImg} alt={randomPokemon.Name} />
            <h3>{randomPokemon.Name}</h3>
            <p>Catch Probability: {Math.round(catchProbability * 100)}%</p>
            <button onClick={handleCatch}>Throw a Pokéball</button>
          </div>
        )
      )}
    </div>
  );
};

export default Forest;
```

forest.css

```
.forest-container {
    max-width: 600px;
    height: 40%;
    margin: 0 auto;
    padding: 20px;
```

```css
  transition: all 0.3s ease;
  opacity: 1; /* default to visible */
  visibility: visible;
}

.forest-container.closed {
  opacity: 0; /* invisible when closed */
  visibility: hidden; /* hidden when closed */
  pointer-events: none;
}

.forest-container h2{
  font-size: 1.7em;
  font-weight: bold;
  text-align: center;
  color: whitesmoke;
  text-shadow: 0px 5px 15px black;
}

.random-pokemon {
  text-align: center;
}

.random-pokemon img {
  max-width: 100%;
  height: auto;
  max-height: 200px;
}

.random-pokemon h3 {
  font-size: 18px;
  margin-bottom: 10px;
  text-transform: capitalize;
}

.random-pokemon p {
  margin-bottom: 10px;
}

.random-pokemon button {
  padding: 10px 20px;
  width: fit-content;
  font-size: 16px;
  border: none;
  background-color: #d80c0c;
  color: #fff;
  cursor: pointer;
}
```

```css
.random-pokemon button:hover {
  background-color: #991010;
  color: #fff;
}

/* in case it takes time to load */
.loading-spinner {
  background-image: url('../assets/pokeball.png');
  border: 5px solid #f3f3f3;
  border-radius: 50%;
  width: 10px;
  height: 10px;
  animation: spin 1s linear infinite; /* Apply rotation animation */
  margin: 0 auto; /* Center the spinner */
}
```

profile.js

```js
import React, { useState, useEffect } from 'react';
import '../styles/profile.css';

const Profile = ({ isOpen, handleClose }) => {
  const [userData, setUserData] = useState(null);

  useEffect(() => {
    if (isOpen) {
      fetchUserData();
    }
  }, [isOpen]);

  const fetchUserData = async () => {
    try {
      const username = localStorage.getItem('username');
      const response = await
fetch(`http://localhost:4000/users?name=${username}`);
      if (!response.ok) {
        throw new Error('Failed to fetch user data');
      }
      const data = await response.json();
      setUserData(data);
    } catch (error) {
      console.error('Error fetching user data:', error);
    }
  };

  const handleDeleteAccount = async () => {
```

```
    // prompt for onfirmation
    const confirmDelete = window.confirm('Are you sure you want to delete your
account? This action cannot be undone.');

    if (confirmDelete) {
      try {
        const username = localStorage.getItem('username');
        const response = await
fetch(`http://localhost:4000/delete/${username}`, {
          method: 'DELETE'
        });
        if (!response.ok) {
          throw new Error('Failed to delete user account');
        }
        showBootstrapAlert('User account has been deleted', 'primary');
        localStorage.removeItem('username');
        localStorage.removeItem('region');
        localStorage.removeItem('starter');
        console.log('User account deleted successfully');
      } catch (error) {
        showBootstrapAlert('Error in deleting account', 'secondary');
        console.error('Error deleting user account:', error);
      }
    }
  };

  const showBootstrapAlert = (message, type) => {
    const alertDiv = document.createElement('div');
    alertDiv.classList.add('alert', `alert-${type}`, 'fixed-top', 'z-index-
alert');
    alertDiv.style.top = '80px';
    alertDiv.role = 'alert';
    alertDiv.textContent = message;
    document.body.appendChild(alertDiv);

    setTimeout(() => {
        alertDiv.remove();
    }, 1500);
  };

  const renderProfileData = () => {
    if (!userData) {
      return <p>Loading...</p>;
    }
    const username = localStorage.getItem('username');
    const region = localStorage.getItem('region');
    const starter = localStorage.getItem('starter');
    const { pokemonCollected } = userData;
```

```
    // Convert the pokemonCollected Map values to an array before joining
    const collectedPokemonArray = pokemonCollected ?
Array.from(pokemonCollected.values()) : [];

    return(
      <div className="profile-details">
        <div className="profile-header">
          <p><strong>Name:</strong> {username}</p>
          <p><strong>Region:</strong> {region}</p>
          <p><strong>Starter:</strong> {starter}</p>
        </div>
        <div className="profile-pokemon-list">
          <strong>Pokemon Collected:</strong>
          {collectedPokemonArray.map((pokemon, index) => (
            <div className="profile-pokemon-list-item" key={index}>
              {pokemon}
            </div>
          ))}
        </div>
        <button className="delete-account-btn"
onClick={handleDeleteAccount}>Delete Account</button>
      </div>
    );
  };

  return (
    <div className={`profile-container ${isOpen ? 'open' : 'closed'}`}>
      <button className="close-btn" onClick={handleClose}>x</button>
      {renderProfileData()}
    </div>
  );
};

export default Profile;
```

profile.css

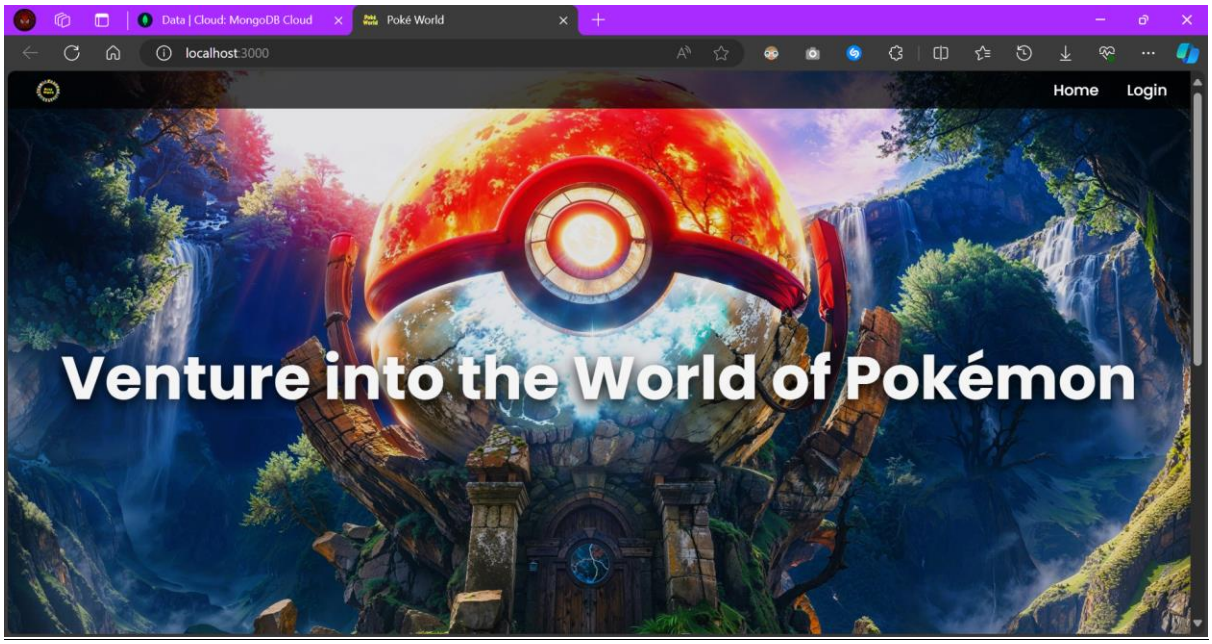
```
.profile-container {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    border: 2px solid #ccc;
    color: white;
    text-shadow: 0 0 4px black;
    background-color: rgba(0,0,0,0.4);
    border-radius: 10px;
```

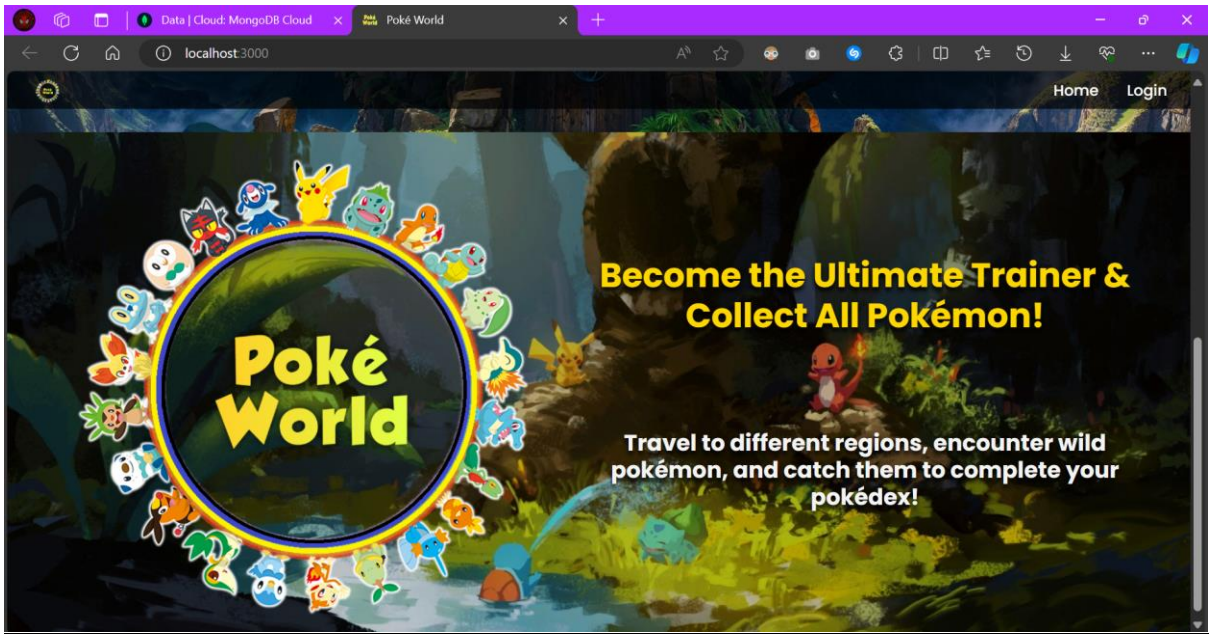
```css
  transition: all 0.3s ease;
  opacity: 1;
  visibility: visible;
  overflow-y: auto;
}

.profile-container.closed {
  opacity: 0;
  visibility: hidden;
}

.profile-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.close-btn {
  background: none;
  border: none;
  font-size: 1.2rem;
  cursor: pointer;
}

.profile-details {
  margin-top: 20px;
}

.profile-details button {
  background-color: red;
  color: white;
  width: fit-content;
  padding: 0 10px;
  margin-top: 10px;
}

.profile-details button:hover {
  background-color: darkred;
  color:white;
}

.profile-details-item {
  margin: 0 5px 10px 5px;
}

.profile-details-item strong {
  font-weight: bold;
}
```

```css
.profile-pokemon-list {
  margin-top: 20px;
}

.profile-pokemon-list-item {
  margin-bottom: 5px;
  text-transform: capitalize;
}
```

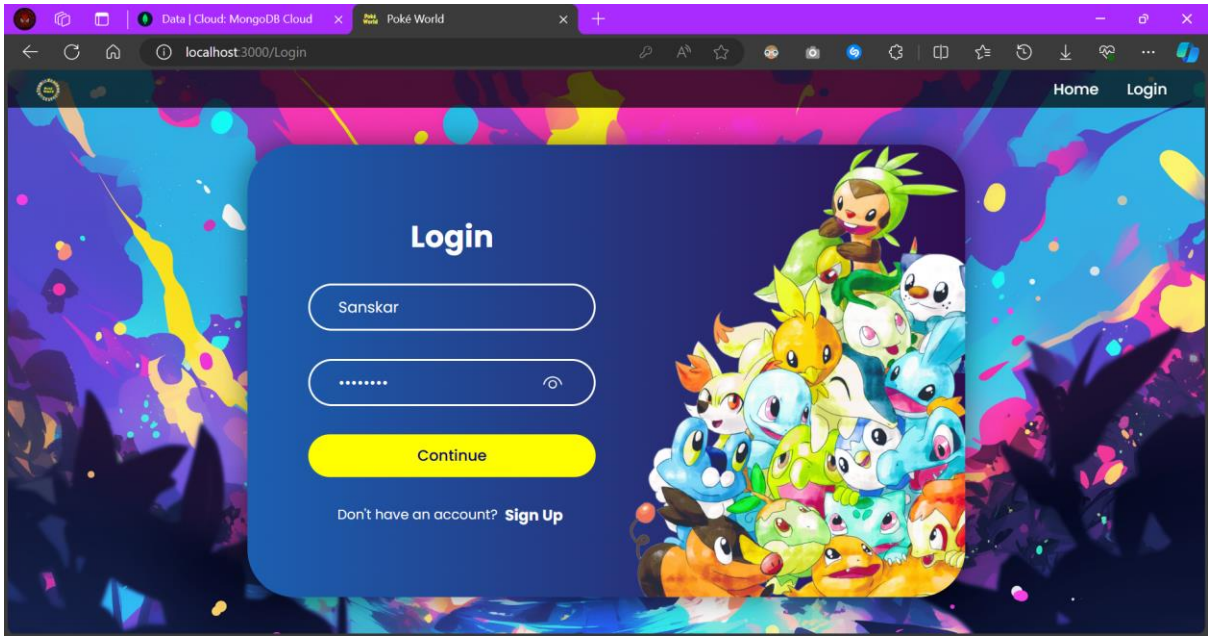
Output Screen Snippets

Home



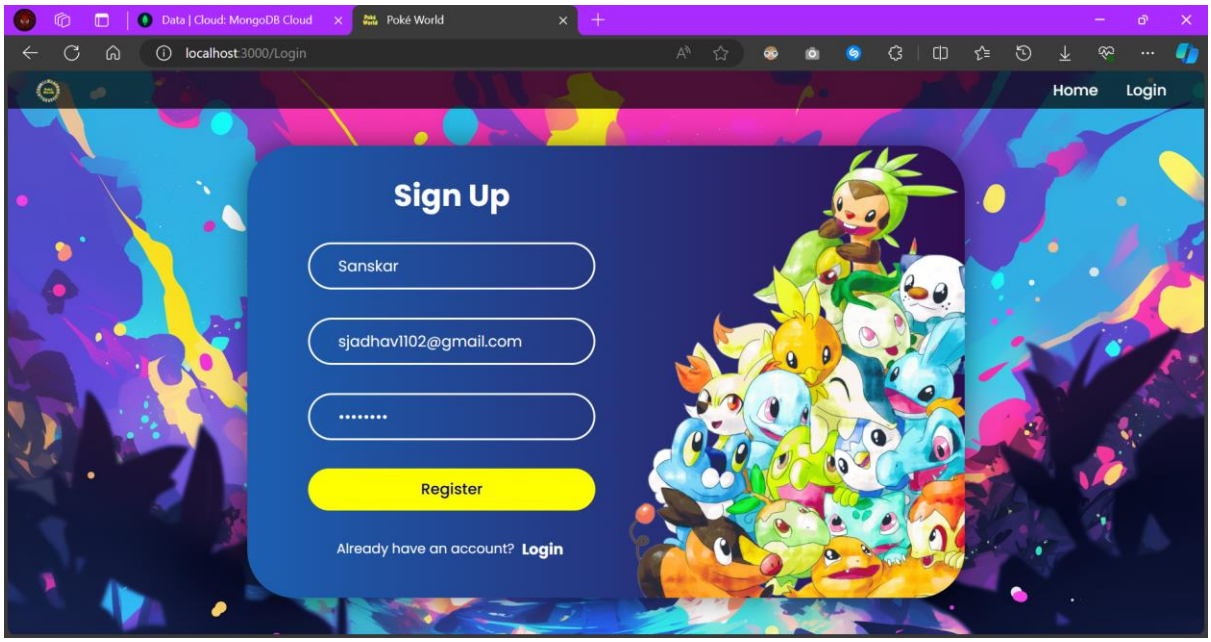


Login



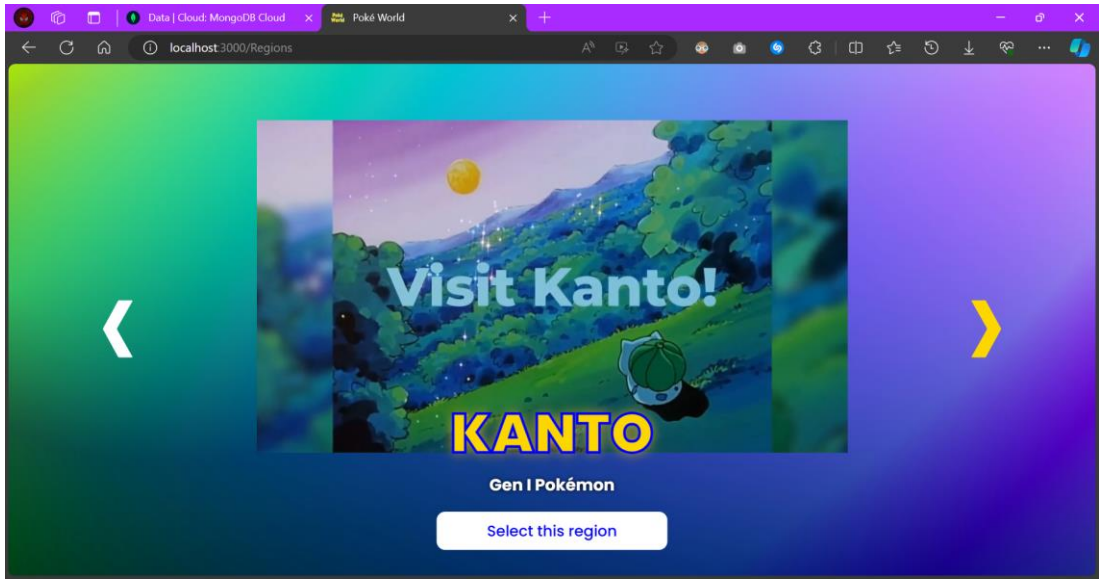
Have added Bootstrap alerts for successful login or user does not exist.

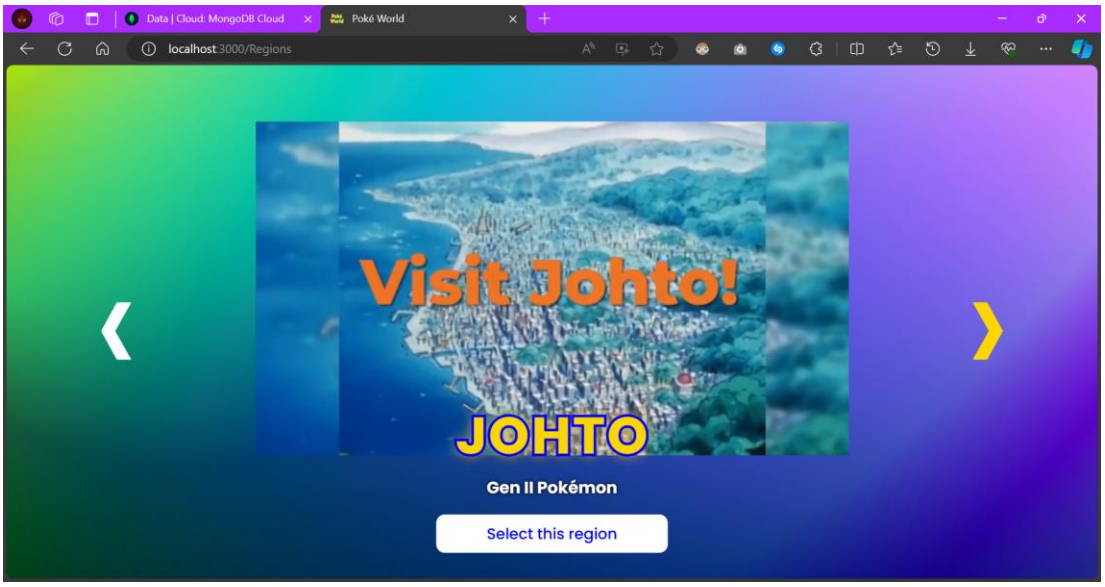
Signup

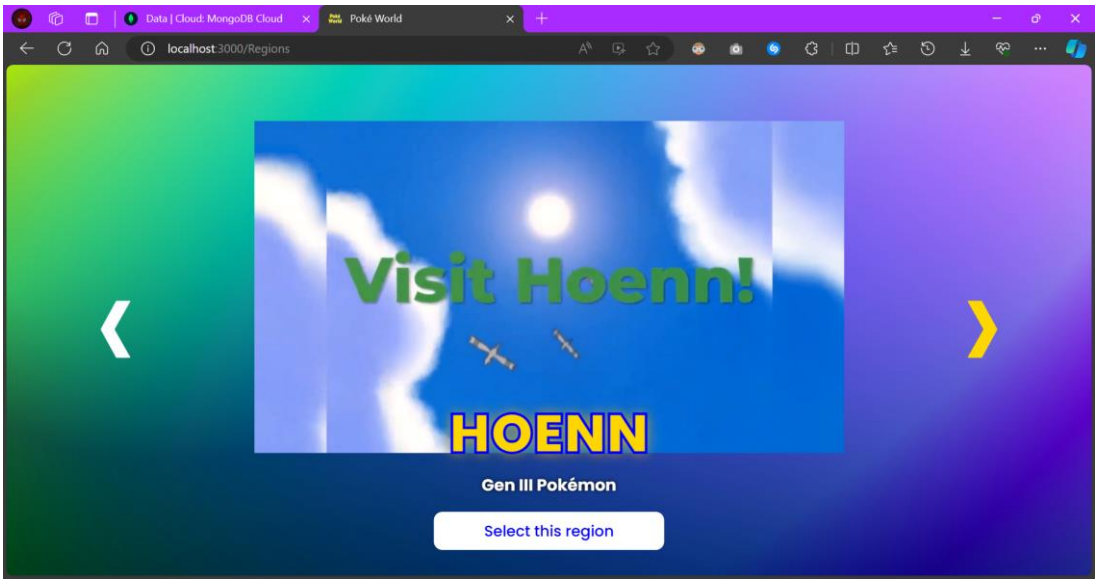


Added Bootstrap alerts for successful sign up or username is taken + Error message if password is below 8 characters in length

Regions

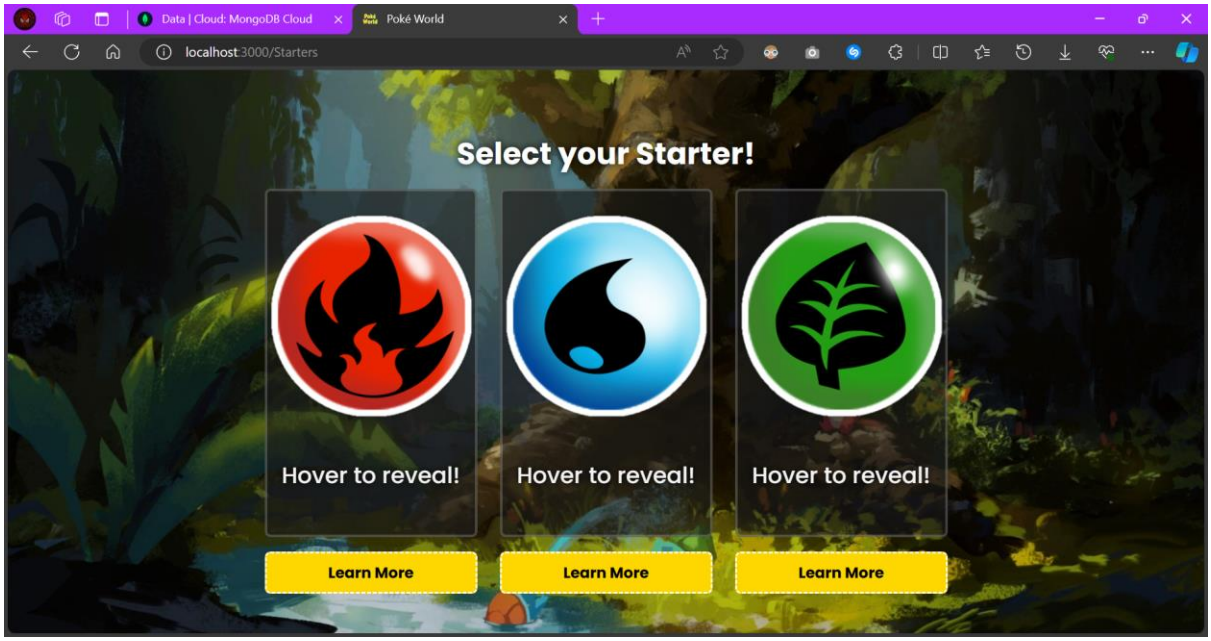

Data | Cloud: MongoDB Cloud
Poké World
localhost:3000/Regions
Visit Kanto!
KANTO
Gen I Pokémon
Select this region


Data | Cloud: MongoDB Cloud
Poké World
localhost:3000/Regions
Visit Johto!
JOHTO
Gen II Pokémon
Select this region

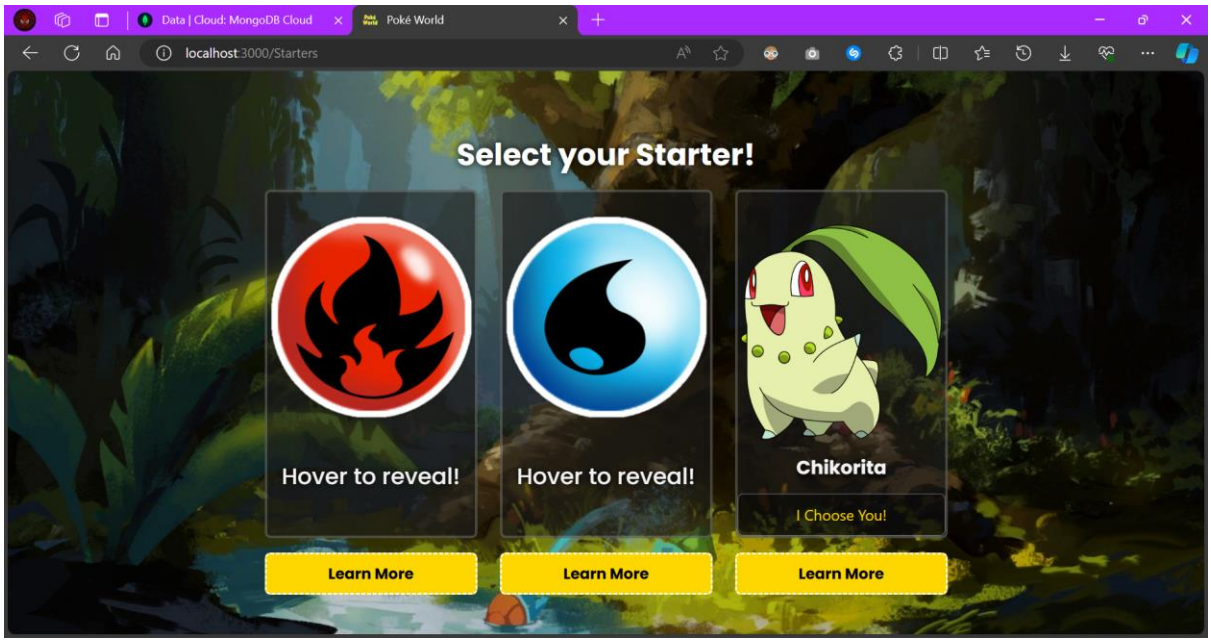

Data | Cloud: MongoDB Cloud
Poké World
localhost:3000/Regions
Visit Hoenn!
HOENN
Gen III Pokémon
Select this region

Videos play for each region, custom JS slider made.

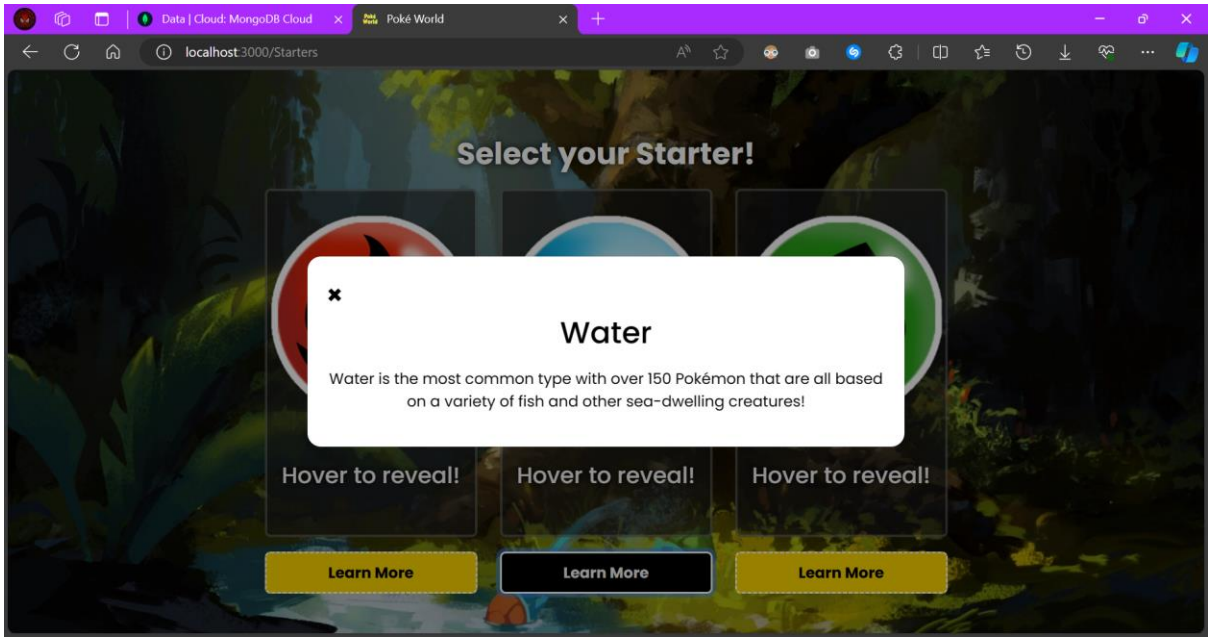
Starters



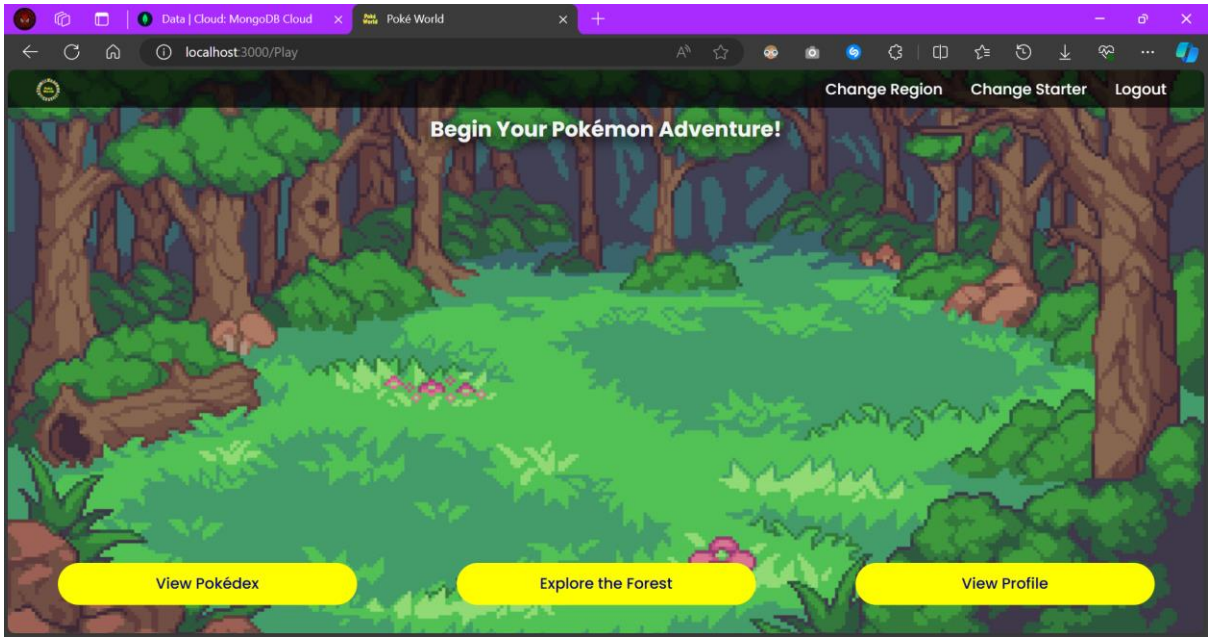
Bootstrap flip cards, unique for each region



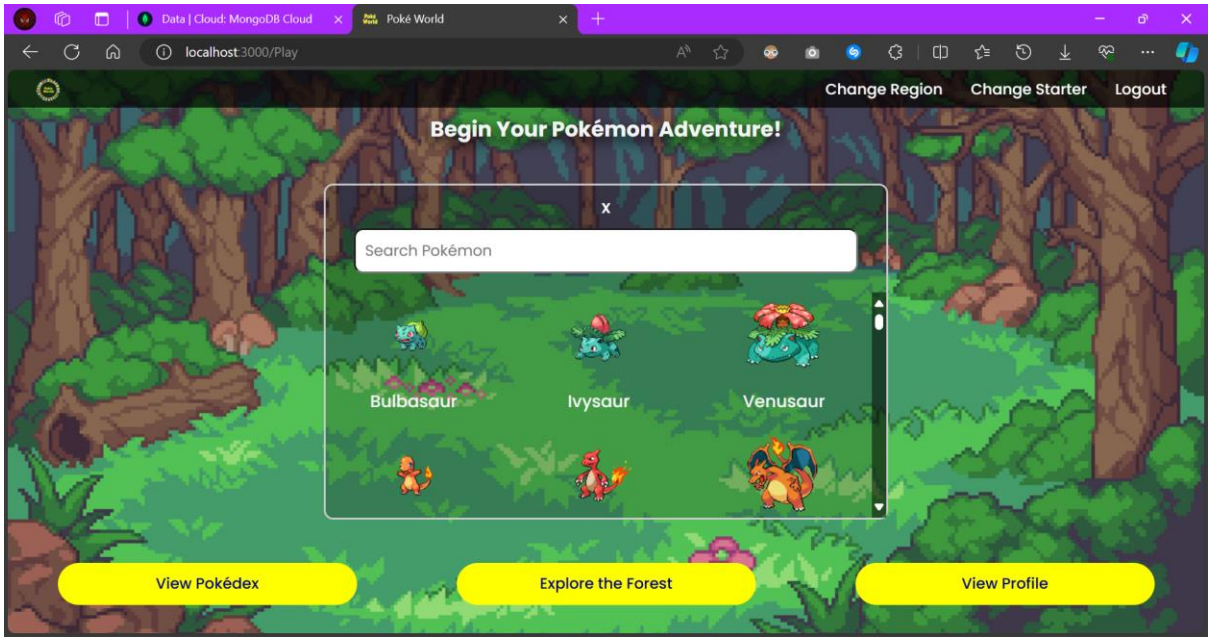
Modals made below



Play

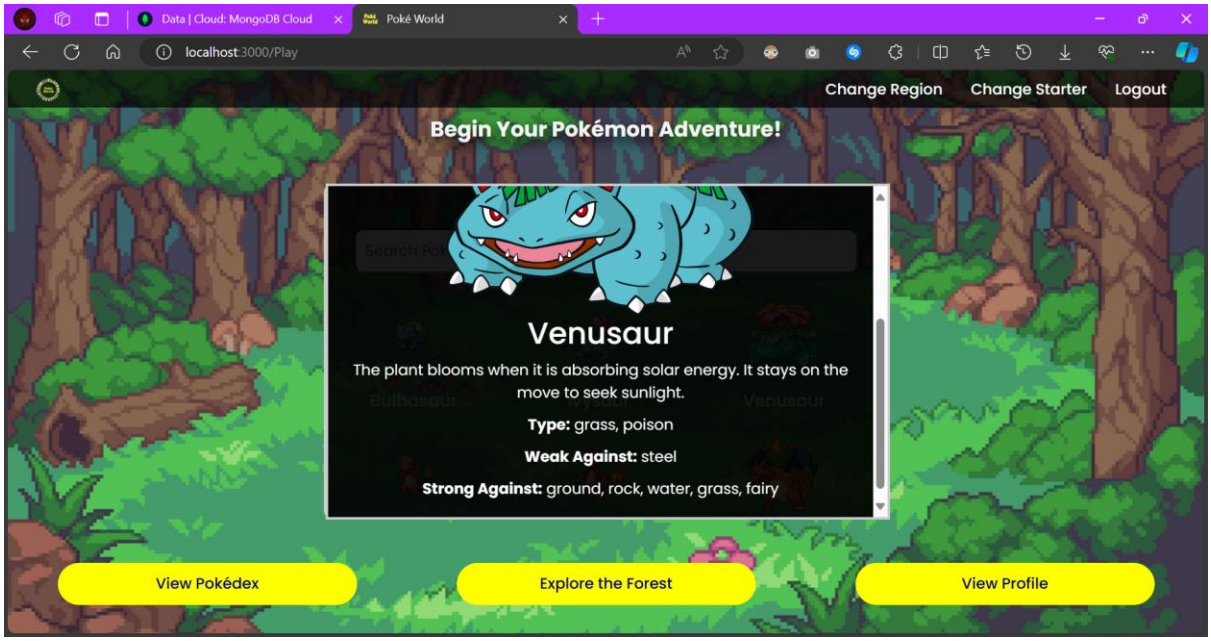


View Pokedex

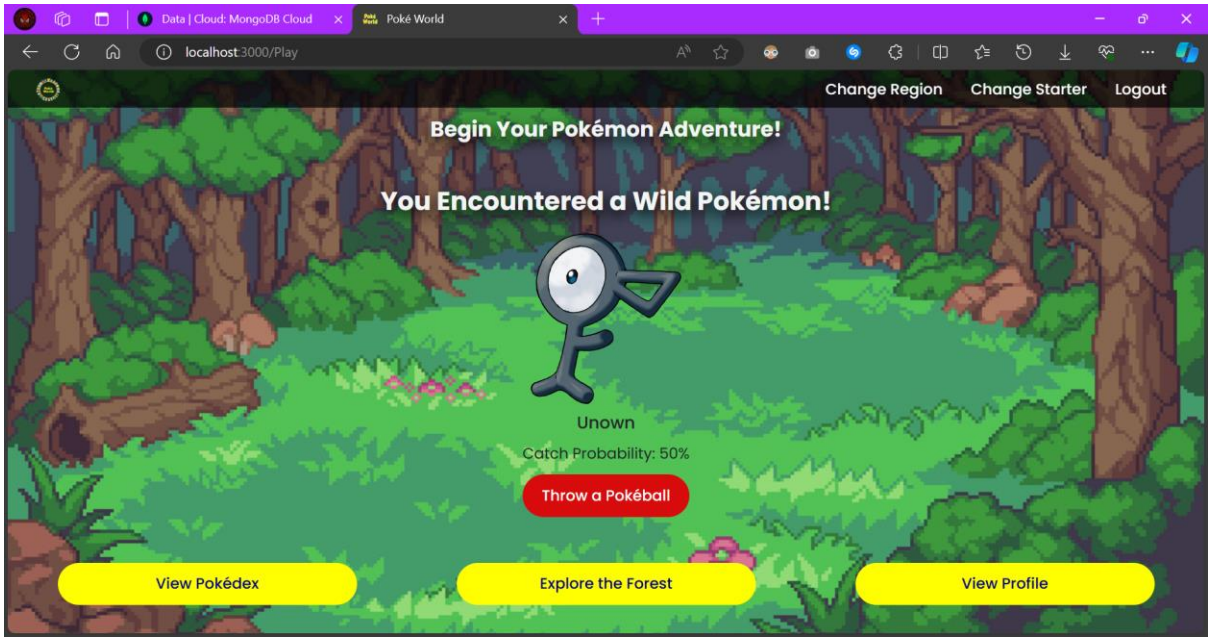


Clickable cards

Give information about each pokemon once clicked on



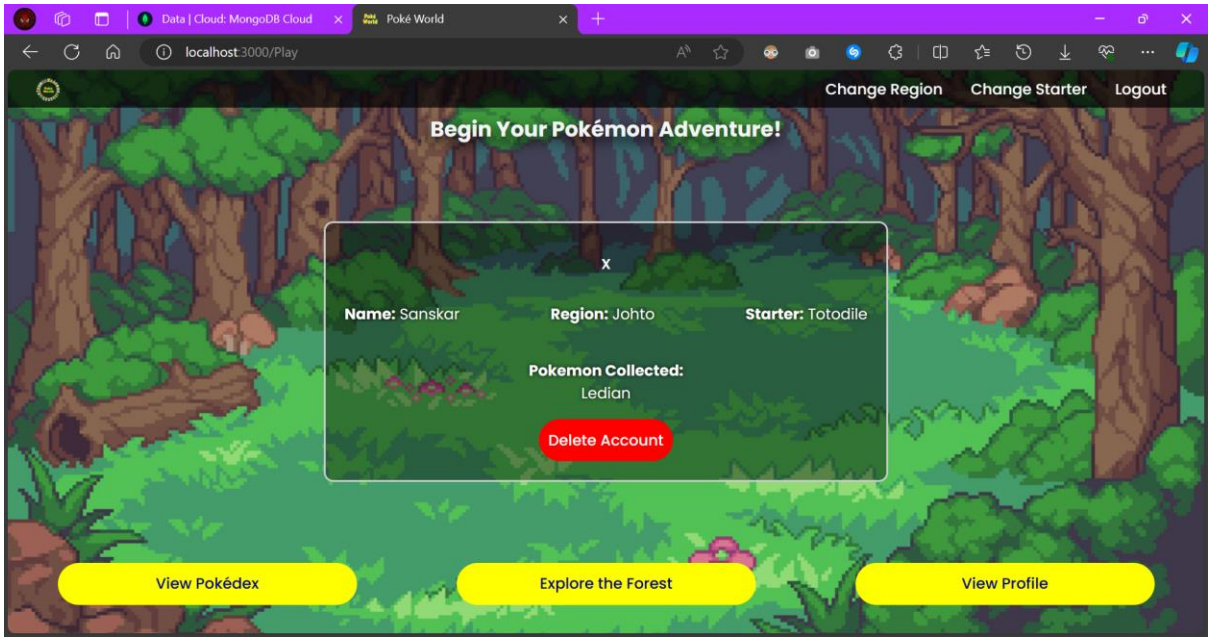
Explore Forest



Can try to catch the pokemon

Bootstrap alerts if successful or not

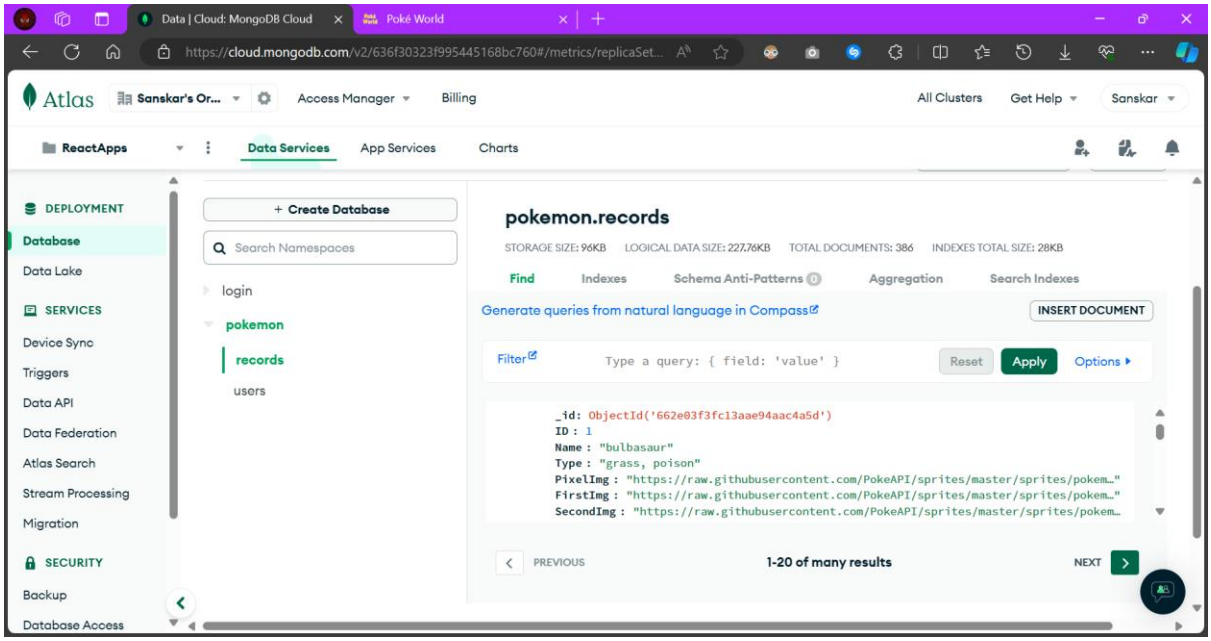
View Profile



Option to delete account with confirmation prompt

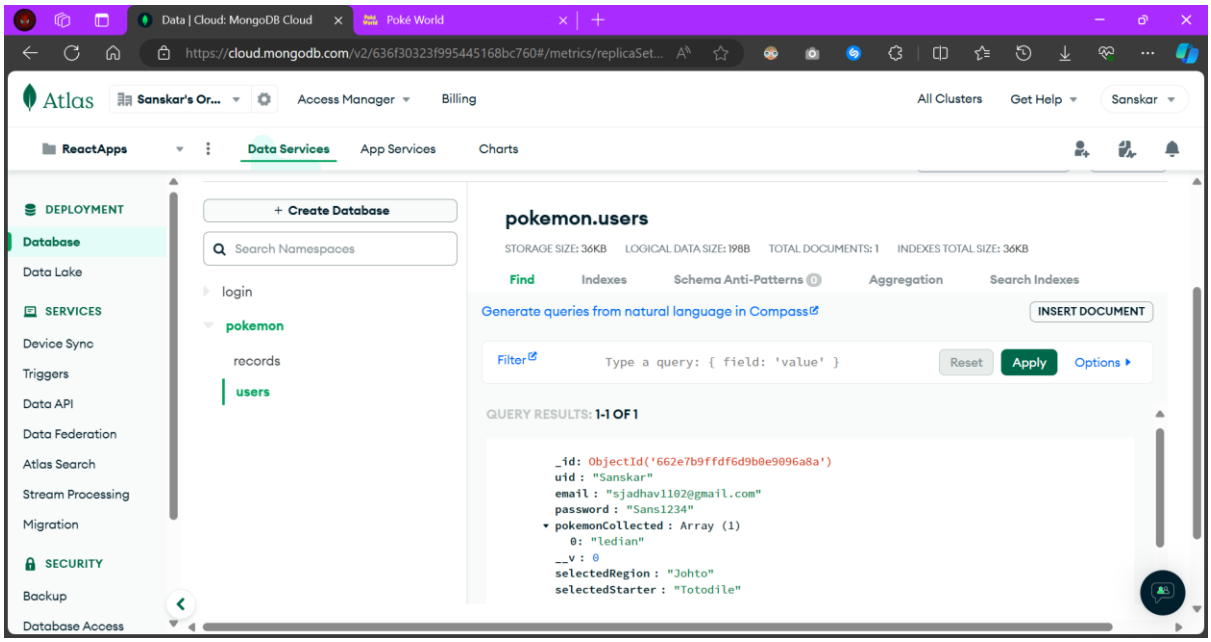
MongoDB Cloud

pokemon.records



Data on all 386 Pokémon

pokemon.users



Data for each user

Conclusion

Thus, in this manner, I built my MERN stack project successfully. I adhered to all expected code implementations and mentioned rubrics. I have carefully designed my websites using HTML and CSS, with multiple Bootstrap components (Buttons, Modals, Cards, Alerts). I have implemented JavaScript interactivity in my Regions page, where users can click left or right to see videos of each region, as well as in my Explore the Forest page, for calculating catch probability and handling whether or not the Pokémon will be caught based on the type. My React components follow all expected guidelines, and I have effectively utilized props and state management throughout multiple components. I built a resuable component, Pokemon Card, for the Starters page. I also designed my Play page to have lifecycles of each component: Pokedex, Forest, and Profile. My Node js + Express code is flawless, with correct usage of app.post and app.get wherever needed, console.logs and alerts being sent with correct network codes, multiple try catch algorithms for error handling, middleware using CORs and Body Parser JSON and URL-encoded, and appropriate routing done so that access to most links are limited until user first logs in. Furthermore, my website will remember if you have logged in before and already selected a region and starter, navigating you directly to the play dashboard, for ease of use. Finally, I have implemented MongoDB Cloud connections correctly with appropriate schemas for both collections and have performed all four: Create, Read, Update, and Delete in multiple components.

I aim to take this project even further, expanding to more regions, adding the ability to send friend requests, hosting battles, and I will then host my website online.