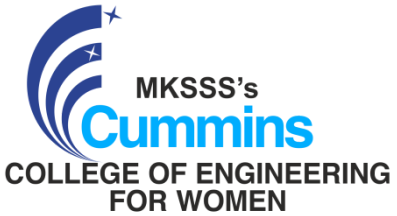


MINI PROJECT REPORT

ON

"Intelligent Knowledge-Based System Using Database Management System"

Submitted By

UIT2021863: Manali Hemant Selmokar

UIT2021864: Ojashri Kamaleshkumar Thakur

UEC2021139: Akanksha Abhijit Kale

DEPARTMENT OF INFORMATION TECHNOLOGY

MKSSS's CUMMINS COLLEGE OF ENGINEERING FOR WOMEN, PUNE

SAVITRIBAI PHULE PUNE UNIVERSITY

2022 – 2023

MKSSS's Cummins COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF INFORMATION TECHNOLOGY

Certificate

This is to certify that,

UIT2021863: Manali Hemant Selmokar
UIT2021864: Ojashri Kamaleshkumar Thakur
UEC2021139: Akanksha Abijit Kale

have successfully completed the Mini Project entitled **"Intelligent Knowledge-Based System Using Database Management System"**, under my guidance in partial fulfillment of the requirement for the Database Management Laboratory second year in Department of Information Technology of Savitribai Phule Pune University during the Academic Year 2022 – 2023

Date :
Place : Pune

Mrs. Leena Panchal
Name of subject Teacher

Dr. Anagha Kulkarni
(Head, Dept. of Information Technology)

Dr. Madhuri Khambete

Principal

ACKNOWLEDGMENT

We would like to express our sincere gratitude to all those who have provided their support and guidance throughout this project.We would like to thank our project guide Leena ma'am, whose valuable guidance and support has been instrumental in shaping this project.It has been a great learning experience and we would like to extend our sincere appreciation for providing us with the opportunity to work on this project.

Working on this project has been an enriching and rewarding experience. It provided us with the opportunity to learn and implement various technical skills and knowledge.This project challenged us to think creatively, problem-solve effectively, and communicate efficiently with our team members. It has been an excellent platform for us to grow and improve as professionals, and we are grateful for the experience.

ABSTRACT

This project aims to create a database management system for a Pokemon game, similar to a famous game called Akinator which allows users to access information about various Pokemon characters. The system will provide a user-friendly interface that will allow users to input characteristics of a Pokemon they are thinking of and receive a list of possible Pokemon matches. The system will also allow users to access information about different Pokemon characters such as their abilities,types, and other characteristics. The system will be designed to store and retrieve information in a secure and efficient manner and will aim to provide an improved user experience for players of this Pokemon guessing game.

LIST OF PAGES

LIST OF TABLES

CONTENTS

1. INTRODUCTION

The Catch Em' All game is a fun and interactive game that allows players to guess the name of a Pokemon character that the player is thinking of. Similar to the famous game Akinator, the Catch Em' All game uses a series of questions to narrow down the possible options and provide the player with a list of possible Pokemon matches.

In this game, players are asked to input characteristics of a Pokemon they are thinking of such as its type, secondary type, ability, secondary ability, evolution stage, and other characteristics. Based on the player's answers, the system provides a list of possible Pokemon matches.

The Catch Em' All game is designed to be user-friendly, engaging, and challenging, and aims to provide an improved user experience for players of this Pokemon guessing game.

The users of this game could be anyone who is a fan of Pokemon and enjoys playing games that challenge their knowledge about Pokemon characters. It could be children, teenagers or even adults who enjoy playing games related to Pokemon.

2. SCOPE

The scope of this project is to create a functional database management system for a Pokemon guessing game.The system will provide a user-friendly interface that will allow users to input information about the Pokemon they are thinking of. Assuming the information entered by the user is accurate, the game guesses the Pokemon with 100% accuracy.

The system will also include a feature that allows users to create an account,update the details and delete the account. The game will be designed to be both entertaining and educational, providing a fun way for players to learn more about Pokemon characters and their unique traits.

Additionally, the system will be designed to store and retrieve information in a secure and efficient manner. The project will focus on creating a robust and scalable database management system that can accommodate a large number of users and data entries.

The scope of the project also includes the development of a responsive and visually appealing user interface that will work seamlessly on various devices and platforms.

3. FUNCTIONAL REQUIREMENTS OF THE SYSTEM

The Pokemon Akinator game is a fun and interactive game designed for fans of the Pokemon franchise. It challenges players to guess the name of a Pokemon character that they are thinking of by asking a series of questions that narrow down the possibilities. To ensure that the game is engaging and challenging, the system must be designed with several functional requirements.

These requirements will enable players to input characteristics of a Pokemon they are thinking of, such as its type, secondary type, secondary ability, evolution stage, and other characteristics. Based on the player's answers, the system must provide a list of possible Pokemon matches. In this section, we will outline the functional requirements of the Pokemon Akinator game, which are designed to provide an improved user experience for players of this Pokemon guessing game.

1. The system allows users to create accounts, log in, and manage their profiles.
2. The system allows users to select characteristics of the Pokemon they are thinking of, such as its type, secondary type, ability, secondary ability, evolution stage, and other characteristics.
3. The system uses a matching algorithm that considers the player's inputs to provide a list of possible Pokemon matches.
4. The system has a user-friendly interface that is visually appealing and easy to navigate.

3.1 POSSIBLE USERS OF THE SYSTEM WITH PRIVILEGES:

Possible users of the Pokemon Akinator game with privileges may include : players and developers.

3.1.1. Players:

These are the users who will access the game to play and interact with the system by inputting characteristics of a Pokemon they are thinking of. They will have the privilege to access the game, input their answers, and receive possible matches.

3.1.2.Developers:

These are the users who will have the privilege to design and develop the game. They will have the ability to write, test, and deploy code to the production environment. Developers will also have access to the system's source code and development environment.

4. TECHNOLOGIES USED

4.1. Back-end technology:

Java Database Connectivity (JDBC) is an Application Programming Interface (API) used to connect Java applications to databases. It enables Java programs to interact with databases by sending SQL statements, retrieving data, and manipulating data. JDBC provides a set of interfaces and classes that allow Java applications to communicate with different types of databases.

JDBC follows a driver-based architecture. It consists of two main components: the JDBC API and the JDBC driver. The JDBC API is a set of interfaces and classes that enable Java applications to interact with databases. The JDBC driver is a software component that implements the JDBC API and provides connectivity to specific databases. There are four types of JDBC drivers: JDBC-ODBC bridge driver, Native-API driver, Network Protocol driver, and Thin driver.

To use JDBC, a Java program must first load the JDBC driver. This is typically done using the Class.forName() method. Once the driver is loaded, the Java program can create a connection to the database using the DriverManager.getConnection() method. The getConnection() method takes three parameters: the database URL, the database username, and the database password. The URL specifies the type of JDBC driver to use and the location of the database.

Once a connection is established, the Java program can send SQL statements to the database using the Statement object. The Statement object is created using the Connection.createStatement() method. The SQL statement is executed using the Statement.execute() method. The execute() method returns a boolean value indicating whether the statement executed successfully.

The ResultSet object is used to retrieve data from the database. The ResultSet object is created using the Statement.executeQuery() method. The ResultSet object contains the results of the SQL query. The ResultSet.next() method is used to iterate through the rows in the ResultSet. The ResultSet.getXXX() methods are used to retrieve the values of the columns in the ResultSet.

JDBC supports the use of PreparedStatements, which are precompiled SQL statements. PreparedStatements are created using the Connection.prepareStatement() method. PreparedStatements can be used to execute the same SQL statement multiple times with different parameter values.

Given below is a brief overview of how the code is using JDBC to connect to the database and store,update,delete and retrieve data. The following steps were undertaken:

1. **Loading the JDBC driver:** The first line of code, "Class.forName("com.mysql.jdbc.Driver");", loads the MySQL JDBC driver so that it can be used to connect to a MySQL database.
2. **Creating a connection:** The next line, "Connection con = DriverManager.getConnection("jdbc:mysql://localhost/trail", "root", "root");", creates a connection to the "trail" database on the local MySQL server, using the username and password "root".
3. **Creating a statement:** The "Statement stmt = con.createStatement();" line creates a Statement object, which will be used to send SQL statements to the database.
4. **Executing a query:** The "ResultSet rs = stmt.executeQuery("SELECT * FROM mytable");" line executes a SQL SELECT statement that retrieves all rows and columns from the "mytable" table in the database. The results are returned as a ResultSet object.
5. **Processing the results:** Required results were processed.
6. **Closing resources:** The "rs.close();", "stmt.close();", and "con.close();" lines close the ResultSet, Statement, and Connection objects, respectively, to release any resources they may be holding.

4.2. Database:

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to interact with data. It is widely used for web applications and is supported by all major operating systems including Windows, Linux, macOS, and Unix.

MySQL is known for its scalability, reliability, and ease of use. It is widely used by small businesses, startups, and large enterprises alike. Some of the key features of MySQL include:

1. **ACID compliance:** MySQL ensures that all transactions are Atomic, Consistent, Isolated, and Durable.
2. **High performance:** MySQL has a high-performance engine that can handle large volumes of data and high traffic loads.
3. **Cross-platform compatibility:** MySQL is available for all major operating systems, including Windows, Linux, macOS, and Unix.
4. **Open source:** MySQL is an open-source software, which means that it can be downloaded and used for free.
5. **Easy to use:** MySQL has a simple and easy-to-use interface, which makes it ideal for beginners.

Here are brief explanations of some common SQL commands in MySQL that have been used in the mini project:

1. **SELECT:** SELECT statement is used to query data from a database table. It allows users to select specific columns and rows based on certain conditions using various clauses such as WHERE, ORDER BY, GROUP BY, HAVING, and LIMIT.
2. **UPDATE:** UPDATE statement is used to modify the existing data in a database table. It allows users to change values in specific columns of a table based on certain conditions specified using the WHERE clause.
3. **DELETE:** DELETE statement is used to remove data from a database table. It allows users to delete specific rows based on certain conditions specified using the WHERE clause.
4. **INSERT:** INSERT statement is used to add new data to a database table. It allows users to insert new rows into a table and specify values for each column.
5. **JOINS:** JOIN operation is used to combine data from multiple database tables based on a related column between them. It allows users to retrieve data from multiple tables in a single query. Types of JOIN operations include INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN.

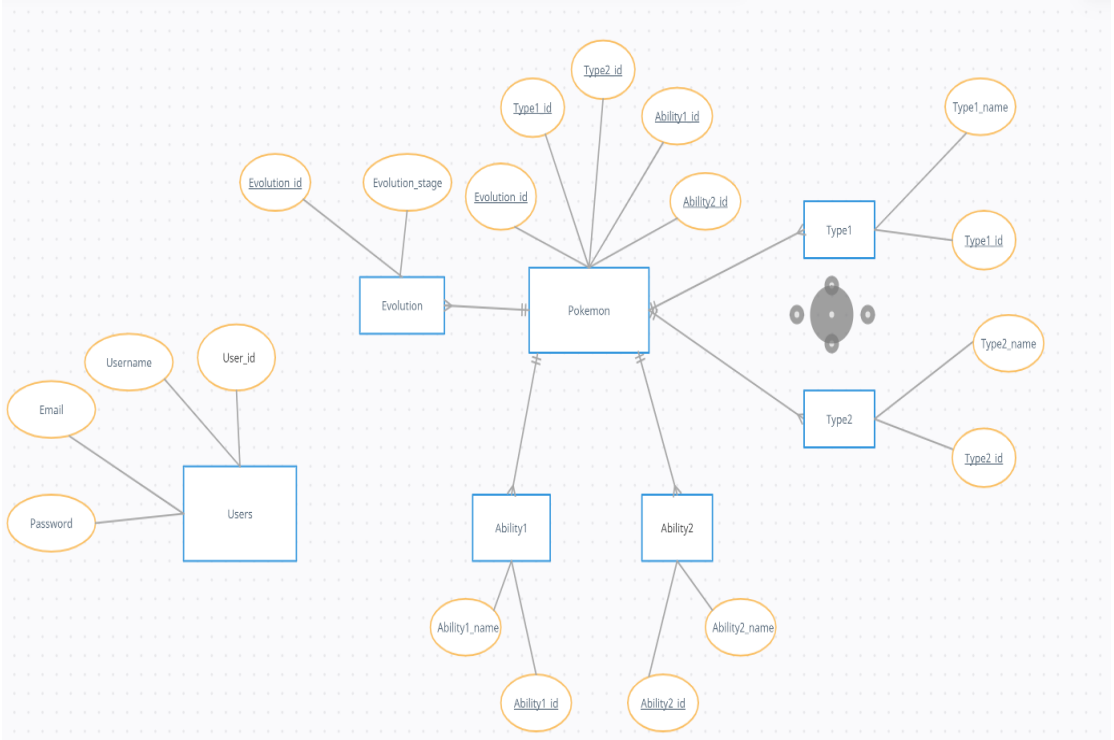
4.3 Front-end

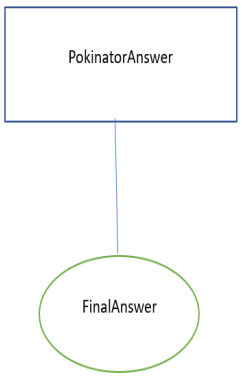
1. **JSP :** JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.Using JSP, you can collect input from users through Web Page forms, present records from a database or another source, and create Web Pages dynamically.JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.

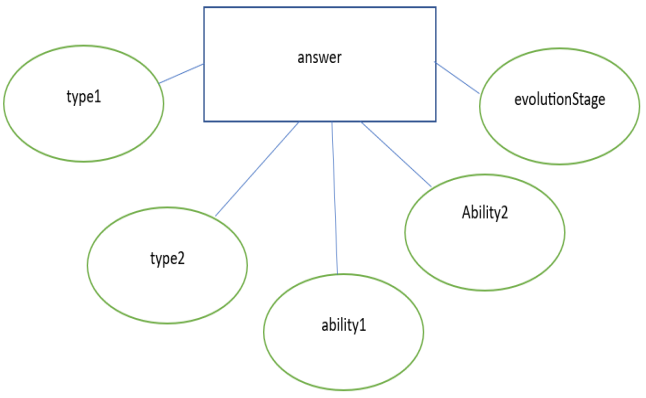
2. **CSS:** Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in HTML. It provides a way to separate the content of a web page from its visual design, allowing developers to create a consistent and standardized look and feel across multiple pages. CSS allows for precise control over the layout, typography, colors, and other visual elements of a web page. It also supports responsive design, allowing pages to adapt to different screen sizes and devices. CSS can be used in conjunction with other technologies like JavaScript and SVG to create complex and interactive web applications.

3. **JavaScript:** JavaScript is a dynamic, object-oriented programming language that is widely used to add interactivity and dynamic behavior to web pages. It is the programming language of the web, and is supported by all major web browsers. JavaScript allows developers to create animations, handle user input, manipulate the DOM, and communicate with servers using AJAX. It is often used in conjunction with HTML and CSS to create complex and interactive web applications. JavaScript frameworks like React, Angular, and Vue have become very popular in recent years, making it easier to develop complex web applications with fewer lines of code.

5. ER DIAGRAM


Type2_id
Type1_id
Ability1_id
Type1_name
Evolution_id
Evolution_stage
Evolution_id
Ability2_id
Type1
Type1_id
Evolution
Pokemon
Username
User_id
Type2_name
Email
Type2
Type2_id
Password
Users
Ability1
Ability2
Ability1_name
Ability2_name
Ability1_id
Ability2_id


PokinatorAnswer
FinalAnswer


answer
type1
evolutionStage
Ability2
type2
ability1

6. NORMALIZATION OF TABLES

Table 6.1: Normalization

TABLE NAME	NORMALIZED FORM
Pokemon	3 NF
Type1	3 NF
Type2	3 NF
Ability1	3 NF
Ability2	3 NF
Ability3	3 NF
EvolutionT	3 NF
Answer	3 NF
PokinatorAns	3 NF

7. DATA DICTIONARY

Table Name: Pokemon

Field Name	Data Type	Length	Constraint	Description
pokemon_id	int	10	Primary Key, Autoincrement	Unique identifier for each Pokemon character
pokemon_name	varchar	50	Not Null	The name of the Pokemon character
type1_id	int	10	Foreign Key to Type1.type1_id	The primary type of the Pokemon character
type2_id	int	10	Foreign Key to Type2.type2_id	The secondary type of the Pokemon character
ability1_id	int	10	Foreign Key to Ability1.ability1_id	The primary ability of the Pokemon character
ability2_id	int	10	Foreign Key to Ability2.ability2_id	The secondary ability of the Pokemon character
evolution_id	int	10	Foreign Key to Evolution.evolution_id	The evolution stage of the Pokemon character

Table Name: Type1

Field Name	Data Type	Length	Constraint	Description
type1_id	int	10	Primary Key, Autoincrement	Unique identifier for each primary Pokemon type
type1_name	varchar	50	Not Null	The name of the primary Pokemon type

Table Name: Type2

Field Name	Data Type	Length	Constraint	Description
type2_id	int	10	Primary Key, Autoincrement	Unique identifier for each secondary Pokemon type
type2_name	varchar	50	Not Null	The name of the secondary Pokemon type

Table Name: Ability1

Field Name	Data Type	Length	Constraint	Description
ability1_id	int	10	Primary Key, Autoincrement	Unique identifier for each primary Pokemon ability
ability1_name	varchar	50	Not Null	The name of the primary Pokemon ability

Table Name: Ability2

Field Name	Data Type	Length	Constraint	Description
ability2_id	int	10	Primary Key, Autoincrement	Unique identifier for each secondary Pokemon ability
ability2_name	varchar	50	Not Null	The name of the secondary Pokemon ability

Table Name: Evolution

Field Name	Data Type	Length	Constraint	Description
evolution_id	int	10	Primary Key, Autoincrement	Unique identifier for each Pokemon evolution stage
evolution_stage	varchar	50	Not Null	The name of the Pokemon evolution stage

Table name: Users

Field Name	Data Type	Key	Constraints
user_id	integer	Primary Key	Auto-increment, Not Null
username	varchar(50)		Unique, Not Null
password	varchar(50)		Not Null
email	varchar(100)		Unique, Not Null

Table: Answer

Column Name	Data Type	Length	Description
id	INT		Unique identifier for record
type1	VARCHAR	50	Primary type of Pokémon
type2	VARCHAR	50	Secondary type of Pokémon (optional)
ability1	VARCHAR	50	Primary ability of Pokémon
ability2	VARCHAR	50	Secondary ability of Pokémon (optional)
evolutionStage	VARCHAR	50	Evolution stage of Pokémon (e.g. basic, stage 1)

Table: PokinatorAnswer

Column Name	Data Type	Length	Description
id	INT		Unique identifier for record
FinalAnswer	VARCHAR	50	Final answer for question

8. BUSINESS LOGIC

Effectively deploy the Catch Em' All game. One approach is to publish the game on popular app stores such as the Apple App Store, Google Play Store, or Amazon App Store, which will allow users to easily download and install the game on their mobile devices. Additionally, the game can be made available on the developer's own website or social media pages, which can help drive traffic and increase visibility.

To promote the game to a wider audience, advertising is a key strategy. Social media platforms such as Facebook, Twitter, and Instagram offer effective channels to reach out to potential players and create buzz around the game. Paid advertising options such as Google AdWords or Facebook Ads can also be leveraged to target specific demographics who are most likely to be interested in playing the game. This approach can help to increase visibility, drive downloads, and generate revenue through in-app purchases or advertising.

In order to optimize advertising efforts, it is important to identify the target audience and create engaging ad content that resonates with them. Additionally, tracking metrics such as click-through rates and conversion rates can help to refine ad strategies over time, maximizing return on investment and driving revenue growth for the game.

To monetize the "Catch Em' All" game, various options are available. One of the effective ways is to deploy the game on various platforms such as mobile, desktop, and web. By making the game available on different platforms, it can reach a wider audience and attract more users. The game can be sold on various app stores and game distribution platforms, generating revenue through downloads and in-app purchases.

Another way to generate revenue is through advertising. By placing ads within the game, advertisers can reach the game's audience, and the game can earn revenue based on the number of ad clicks or impressions. Native advertising, where ads are seamlessly integrated into the game, can be an effective way to monetize without disrupting the user experience.

9. TESTING DOCUMENTS

Following are some of the validations done at the front end:

Table 9.1: Testing Documents

Field Name	Validations
Username	Accepts a valid E-mail ID
Password	Accepts a password of minimum length of 8 characters .
Mobile number	Accepts a 10 digit mobile number in donor as well as recipient form.
E-mail ID	Accepts a valid e-mail address in donor as well as recipient form (format:--@-.-)

10. SAMPLE SOURCE CODE

Registration servlet - java code:

```
package com.miniProject.registration;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;

import javax.servlet.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class RegistrationServlet
 */
@WebServlet("/register")
public class RegistrationServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;

   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      String uname = request.getParameter("name");
      String uemail = request.getParameter("email");
      String upwd = request.getParameter("pass");
      String re_upwd = request.getParameter("re_pass");
      String umobile = request.getParameter("contact");
      RequestDispatcher dispatcher = null;

      if (upwd.length() < 8) {
         dispatcher = request.getRequestDispatcher("registration.jsp");
         request.setAttribute("status", "Password should have a minimum of 8
characters");
         dispatcher.forward(request, response);
         return;
      }
```

```
        if (uname.isEmpty() || uemail.isEmpty() || upwd.isEmpty() || re_upwd.isEmpty()
|| umobile.isEmpty()) {
            dispatcher = request.getRequestDispatcher("registration.jsp");
            request.setAttribute("status", "Please fill all the fields");
            dispatcher.forward(request, response);
            return;
        }

        if (!upwd.equals(re_upwd)) {
            dispatcher = request.getRequestDispatcher("registration.jsp");
            request.setAttribute("status", "Password and Confirm Password do not
match");
            dispatcher.forward(request, response);
            return;
        }

        if (!umobile.matches("\\d{10}")) {
            dispatcher = request.getRequestDispatcher("registration.jsp");
            request.setAttribute("status", "Mobile number should be a 10-digit number");
            dispatcher.forward(request, response);
            return;
        }

        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/trail?useSSL=false",
"root",
                    "root");
            PreparedStatement pst = con.prepareStatement("select * from users where
uemail=?");
            pst.setString(1, uemail);
            ResultSet rs = pst.executeQuery();
            if (rs.next()) {
                dispatcher = request.getRequestDispatcher("registration.jsp");
                request.setAttribute("status", "User already exists with this Email Id");
                dispatcher.forward(request, response);
                return;
            } else {
                pst = con.prepareStatement("insert into users(uname,upwd,uemail,umobile)
values(?,?,?,?)");
                pst.setString(1, uname);
                pst.setString(2, upwd);
                pst.setString(3, uemail);
                pst.setString(4, umobile);
                int rowCount = pst.executeUpdate();
                dispatcher = request.getRequestDispatcher("registration.jsp");
                if (rowCount > 0) {
                    request.setAttribute("status", "Registration Successful");
```

```
            } else {
               request.setAttribute("status", "Registration Failed");
            }
            dispatcher.forward(request, response);
         }
         rs.close();
         pst.close();
         con.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

}
```

Question JSP code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Ability1  </title>


<!-- Font Icon -->
<link rel="stylesheet"
        href="fonts/material-icon/css/material-design-iconic-font.min.css">

<!-- Main css -->
<link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
<input type="hidden" id="status" value="<%=request.getAttribute("status") %>">
        <div class="main">

                <!-- Sing in  Form -->
                <section class="sign-in">
                        <div class="container">
                                <div class="signin-content">
                                        <div class="signin-image">

                                                <figure>
                                                        <img src="images\1_ability.jpg"
alt="images\pokeball-logo-DC23868CA1-seeklogo.com-removebg-preview.png">
                                                </figure>

                                        </div>
```

```
                    <div class="signin-form">
                        <h2 class="form-title">Which ONE of
the following abilities does your Pokemon Possess?</h2>

                        <form method="post" action="a1_Blaze"
class="register-form"
                            id="login-form">

                            <div class="form-group
form-button">
                                <input type="submit"
name="ability1" id="signin"

class="form-submit" value="Blaze" />
                            </div>

                        </form>

                        <form method="post"
action="a1_Chlorophyll" class="register-form"
                            id="login-form">


                            <div class="form-group
form-button">
                                <input type="submit"
name="ability1" id="signin"

class="form-submit" value="Chlorophyll" />
                            </div>
                        </form>

                        <form method="post"
action="a1_Clear_Body" class="register-form"
                            id="login-form">


                            <div class="form-group
form-button">
                                <input type="submit"
name="ability1" id="signin"

class="form-submit" value="Clear Body" />
                            </div>
                        </form>

                        <form method="post"
action="a1_Compound_Eyes" class="register-form"
                            id="login-form">
```

```
                                                    <div class="form-group
form-button">
                                                        <input type="submit"
name="ability1" id="signin"
class="form-submit" value="Compound Eyes" />
                                                    </div>
                                                </form>
                                                <form method="post" action="a1_2.jsp"
class="register-form"
                                                    id="login-form">


                                                    <div class="form-group
form-button">
                                                        <input type="submit"
name="ability1" id="signin"
class="form-submit" value="None of the Above" />
                                                    </div>
                                                </form>

                                            </div>
                                        </div>
                                    </div>
                                </section>

        </div>

        <!-- JS -->
        <script src="vendor/jquery/jquery.min.js"></script>
        <script src="js/main.js"></script>
        <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
        <
</body>

</html>
```

CSS code sample :

```
img {
align-items: center;
}

body {
background-image: url(https://wallpaperaccess.com/full/21000.jpg);
background-size: cover;
background-color: #cccccc;
}

h2 {
display: flex;
font-size: 24px;
text-align: center;
}

p {
font-size: 24px;
text-align: center;
}

section {
font-size: 24px;
text-align: center;
font-color: rgb(1, 50, 137);
}

.btn-info {
background-color: yellow;
color: black;
}
```

Javascript code sample:

```
 let navbarlinks = select('#navbar .scrollto', true)
  const navbarlinksActive = () => {
    let position = window.scrollY + 200
    navbarlinks.forEach(navbarlink => {
      if (!navbarlink.hash) return
      let section = select(navbarlink.hash)
      if (!section) return
      if (position >= section.offsetTop && position <= (section.offsetTop +
section.offsetHeight)) {
        navbarlink.classList.add('active')
```

```
    } else {
      navbarlink.classList.remove('active')
    }
  })
}
window.addEventListener('load', navbarlinksActive)
onscroll(document, navbarlinksActive)

/**
 * Scrolls to an element with header offset
 */
const scrollto = (el) => {
  let header = select('#header')
  let offset = header.offsetHeight

  if (!header.classList.contains('header-scrolled')) {
    offset -= 10
  }

  let elementPos = select(el).offsetTop
  window.scrollTo({
    top: elementPos - offset,
    behavior: 'smooth'
  })
}

/**
 * Toggle .header-scrolled class to #header when page is scrolled
 */
let selectHeader = select('#header')
if (selectHeader) {
  const headerScrolled = () => {
    if (window.scrollY > 100) {
      selectHeader.classList.add('header-scrolled')
    } else {
      selectHeader.classList.remove('header-scrolled')
    }
  }
  window.addEventListener('load', headerScrolled)
  onscroll(document, headerScrolled)
}

/**
 * Back to top button
 */
let backtotop = select('.back-to-top')
if (backtotop) {
  const toggleBacktotop = () => {
    if (window.scrollY > 100) {
      backtotop.classList.add('active')
```

```
    } else {
      backtotop.classList.remove('active')
    }
  }
  window.addEventListener('load', toggleBacktotop)
  onscroll(document, toggleBacktotop)
}

/**
 * Mobile nav toggle
 */
on('click', '.mobile-nav-toggle', function(e) {
  select('#navbar').classList.toggle('navbar-mobile')
  this.classList.toggle('bi-list')
  this.classList.toggle('bi-x')
})

/**
 * Mobile nav dropdowns activate
 */
on('click', '.navbar .dropdown > a', function(e) {
  if (select('#navbar').classList.contains('navbar-mobile')) {
    e.preventDefault()
    this.nextElementSibling.classList.toggle('dropdown-active')
  }
}, true)

/**
 * Scrool with ofset on links with a class name .scrollto
 */
on('click', '.scrollto', function(e) {
  if (select(this.hash)) {
    e.preventDefault()

    let navbar = select('#navbar')
    if (navbar.classList.contains('navbar-mobile')) {
      navbar.classList.remove('navbar-mobile')
      let navbarToggle = select('.mobile-nav-toggle')
      navbarToggle.classList.toggle('bi-list')
      navbarToggle.classList.toggle('bi-x')
    }
    scrollto(this.hash)
  }
}, true)

/**
 * Scroll with ofset on page load with hash links in the url
 */
window.addEventListener('load', () => {
  if (window.location.hash) {
```

```
    if (select(window.location.hash)) {
      scrollto(window.location.hash)
    }
  }
});

/**
 * Clients Slider
 */
new Swiper('.clients-slider', {
  speed: 400,
  loop: true,
  autoplay: {
    delay: 5000,
    disableOnInteraction: false
  },
  slidesPerView: 'auto',
  pagination: {
    el: '.swiper-pagination',
    type: 'bullets',
    clickable: true
  },
  breakpoints: {
    320: {
      slidesPerView: 2,
      spaceBetween: 40
    },
    480: {
      slidesPerView: 3,
      spaceBetween: 60
    },
    640: {
      slidesPerView: 4,
      spaceBetween: 80
    },
    992: {
      slidesPerView: 6,
      spaceBetween: 120
    }
  }
});

/**
 * Porfolio isotope and filter
 */
window.addEventListener('load', () => {
  let portfolioContainer = select('.portfolio-container');
  if (portfolioContainer) {
    let portfolioIsotope = new Isotope(portfolioContainer, {
      itemSelector: '.portfolio-item',
```

```
      layoutMode: 'fitRows'
    });

    let portfolioFilters = select('#portfolio-flters li', true);

    on('click', '#portfolio-flters li', function(e) {
      e.preventDefault();
      portfolioFilters.forEach(function(el) {
        el.classList.remove('filter-active');
      });
      this.classList.add('filter-active');

      portfolioIsotope.arrange({
        filter: this.getAttribute('data-filter')
      });
      aos_init();
    }, true);
  }

});
```

11. SCREEN SHOTS

Figure 11.1: Sign up

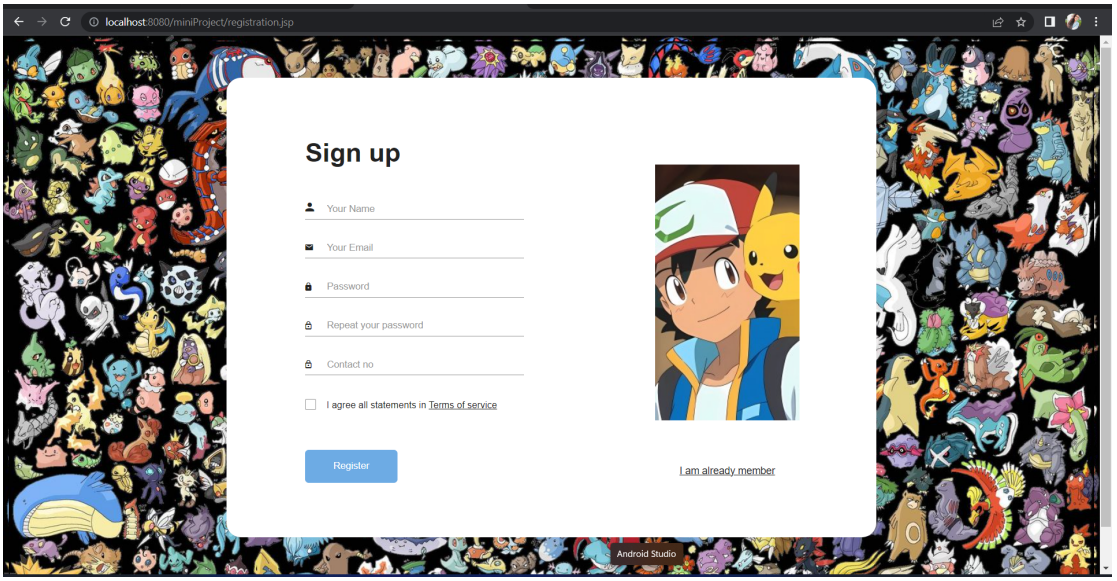


Figure 11.2: Login

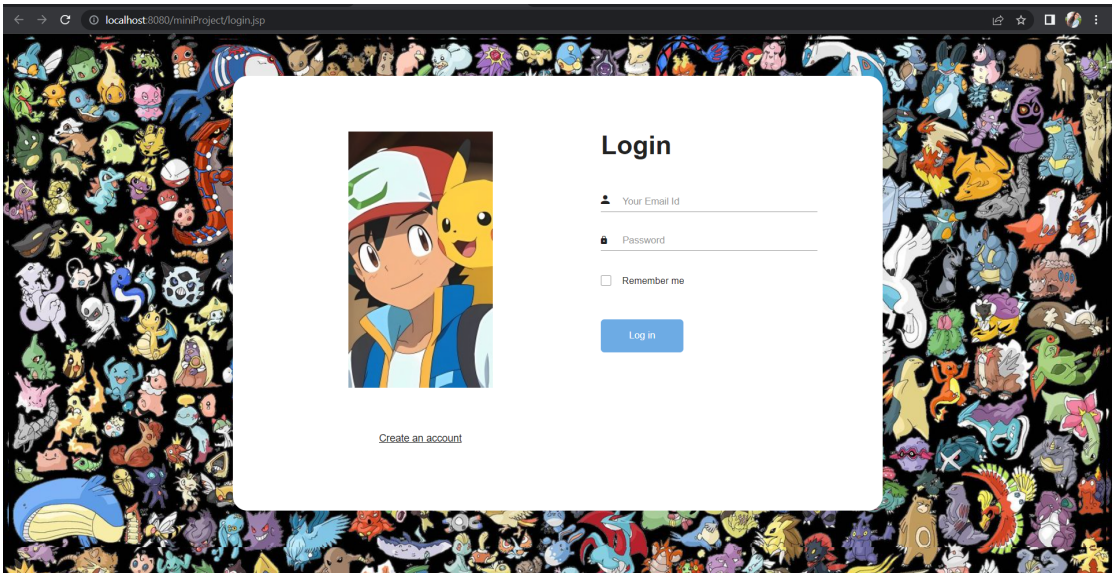


Figure 11.3: Welcome page

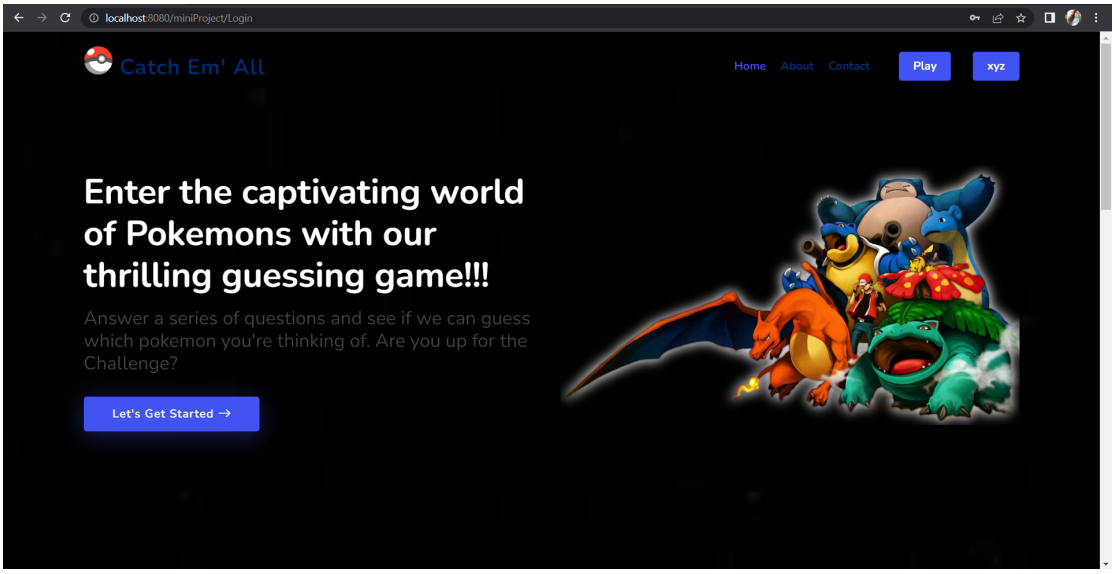


Figure 11.4: About us

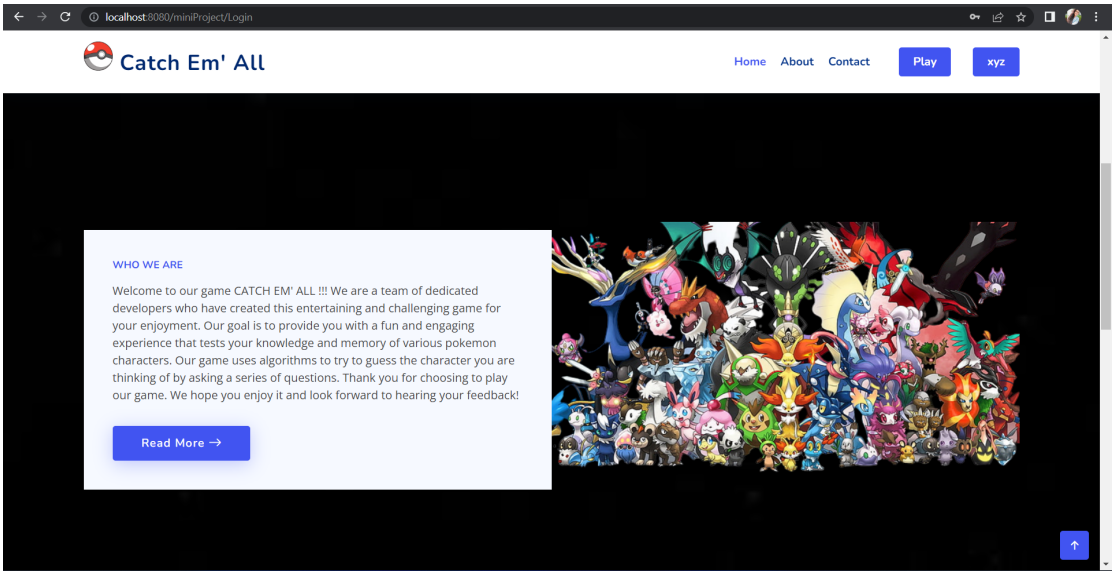


Figure 11.5: Contact us

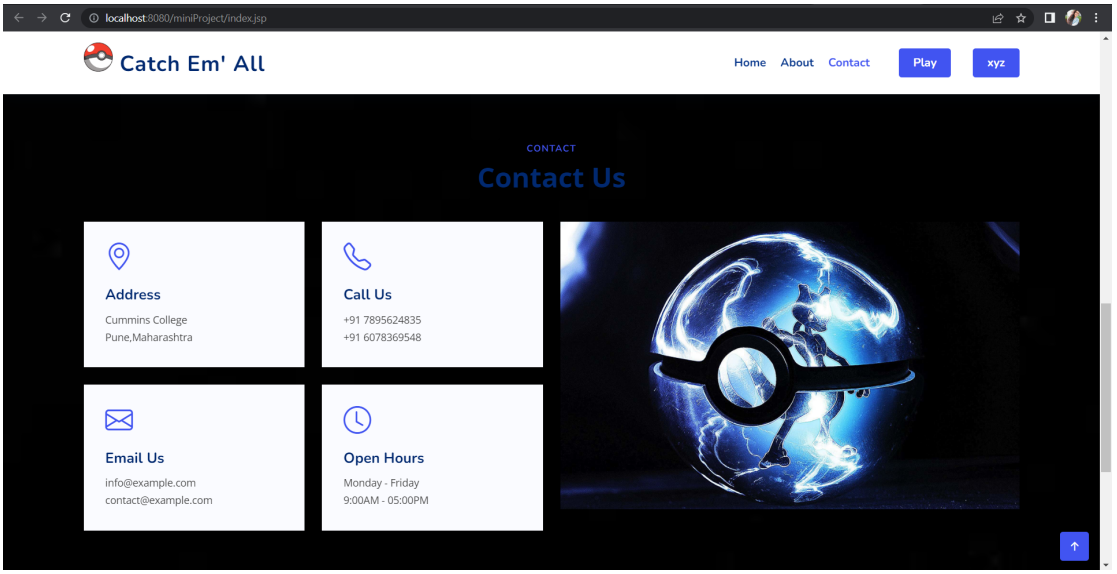


Figure 11.6: Game intro page

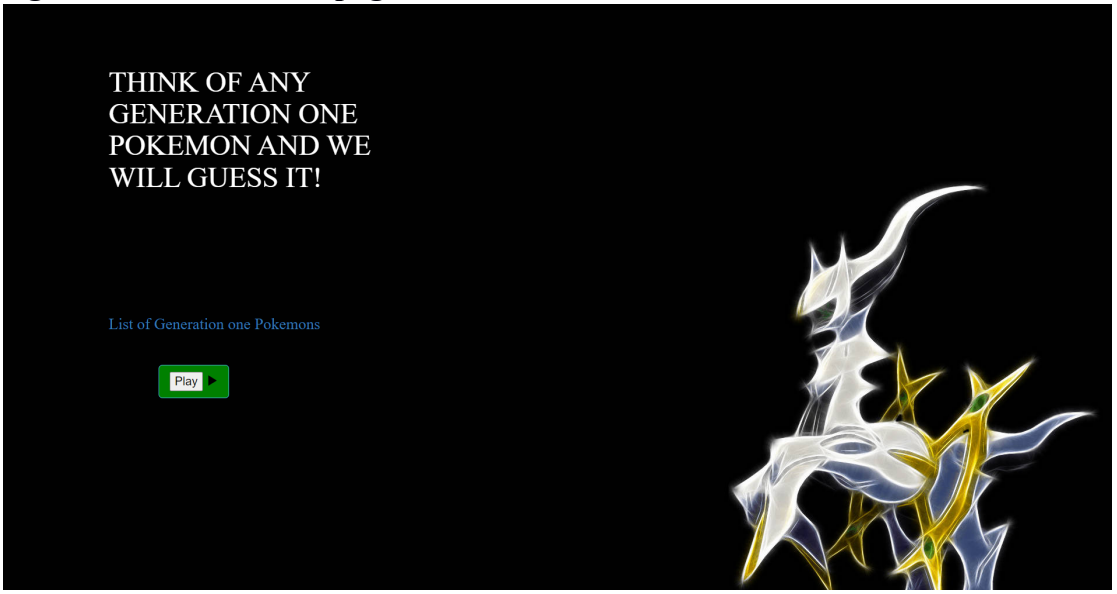


Figure 11.7: Game Questions page

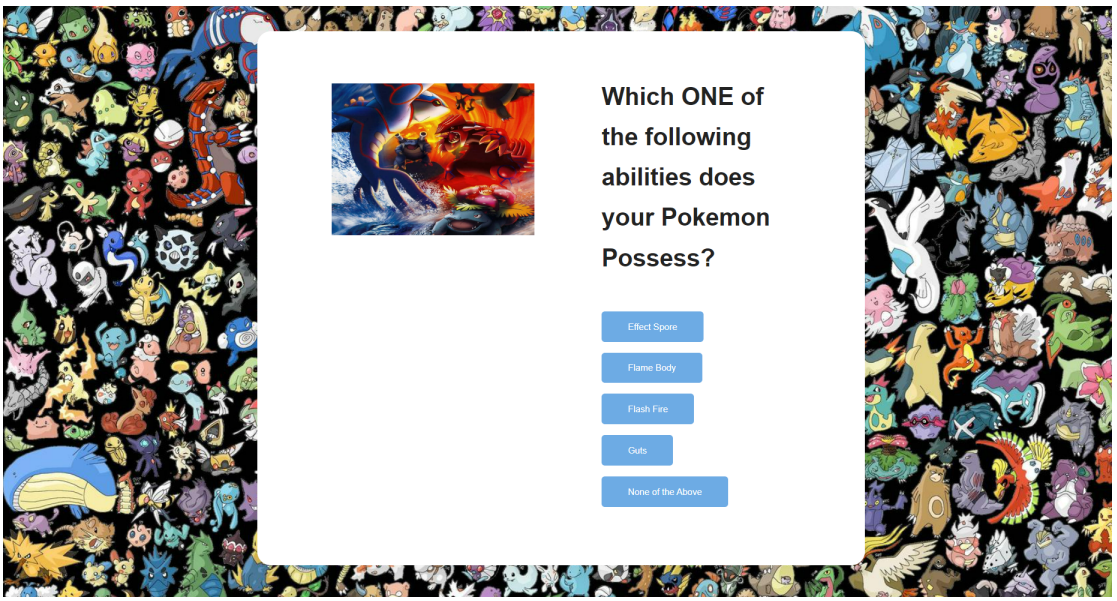


Figure 11.8: Pre-Result Page

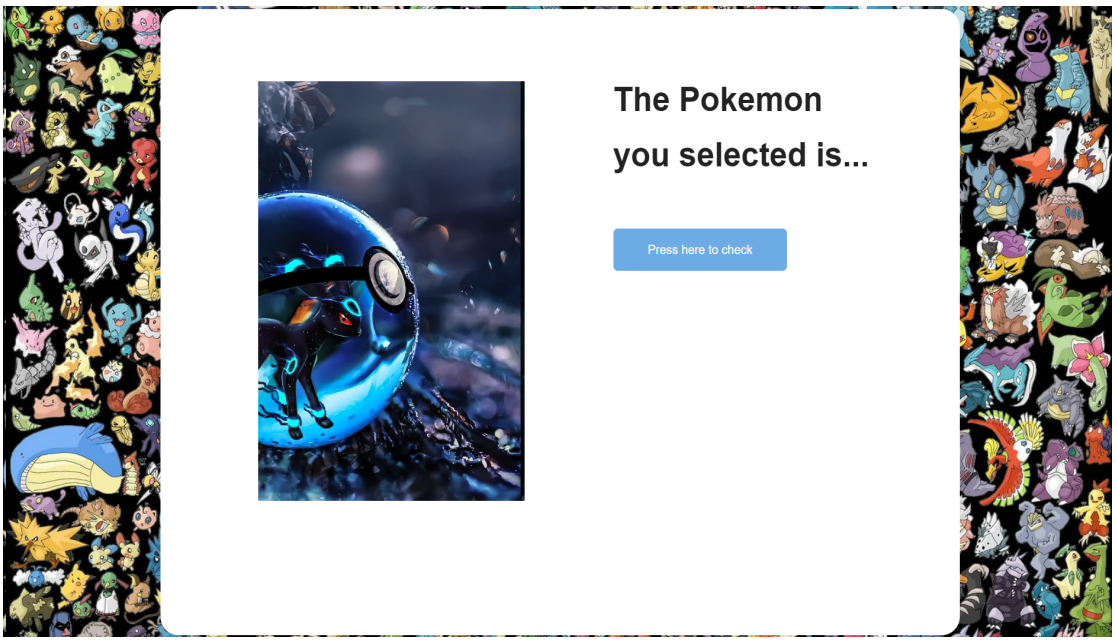
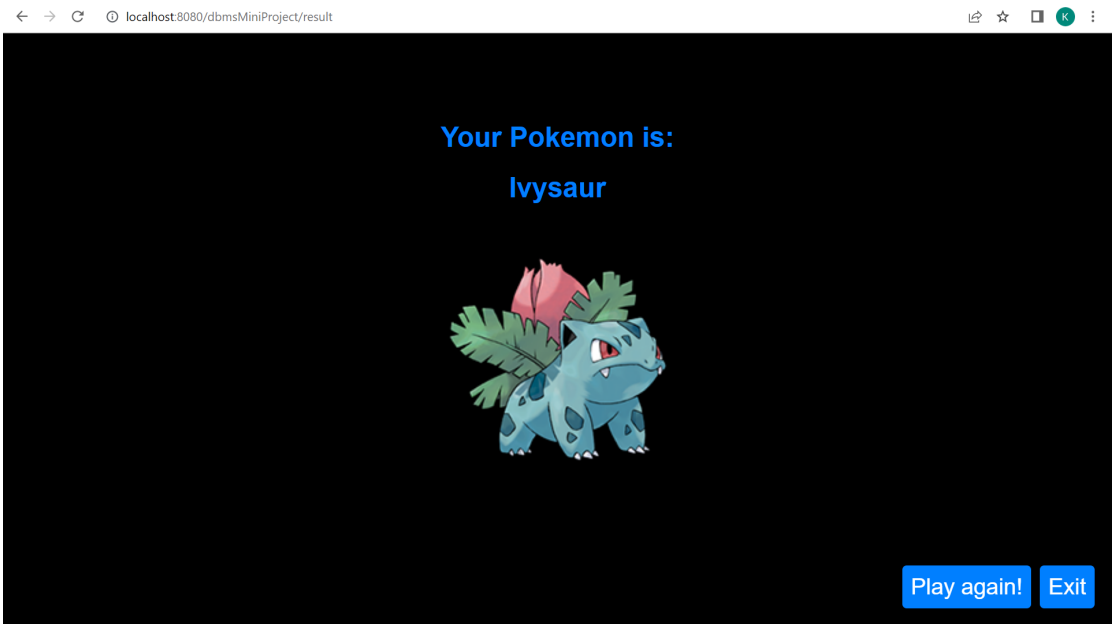


Figure 11.8: Result page


localhost:8080/dbmsMiniProject/result
Your Pokemon is:
Ivysaur
Play again!
Exit

CONCLUSION :

In conclusion, the "Catch 'Em All" Pokemon guessing game is an engaging and educational way to test your knowledge of Pokemon characters. The game was implemented in Java using object-oriented programming principles, and a database management system was used to store and retrieve the Pokemon data based on its abilities and type.

The game allows users to guess the name of a Pokemon based on its abilities and type, and evolution stage.

The project demonstrated the importance of using object-oriented design principles to create modular and reusable code, as well as the versatility of Java in developing interactive applications. The database component of the project also highlighted the importance of database management systems in real-world applications.

Overall, "Catch 'Em All" helped us develop valuable skills in Java programming, database connectivity, Frontend development, and database management. By completing this project, we gained a deeper understanding of these concepts and their applications in software development, as well as enhancing our knowledge of Pokemon characters, abilities, and types.

REFERENCES :

1) https://www.pokemon.com : for information about Pokemon.
2) https://bulbapedia.bulbagarden.net : for information about Pokemon.
3) https://pokemondb.net : for information about Pokemon.
4) https://www.akinator.com : inspiration behind the game.
5) https://stackoverflow.com : for coding resources and problem solving.
6) https://dribbble.com : for UI/UX inspiration.