ROADMAP for Implementing
LYON'S AI HOUSING INVESTMENT WEB PLATFORM

Phase 1: Initial Setup and Database Design

1. Set Up Development Environment

Install Local Web Server Environment:

- **Download and Install XAMPP:**
    1. Visit the XAMPP download page.
    2. Download the appropriate version for your operating system.
    3. Run the installer and follow the prompts to install XAMPP.
- **Configure XAMPP:**
    1. Open XAMPP Control Panel.
    2. Start the Apache and MySQL modules.
    3. Open your web browser and go to http://localhost/ to test if XAMPP is running correctly.

Configure Code Editor:

- **Set Up Visual Studio Code (VS Code):**
    1. Download and install Visual Studio Code.
    2. Open VS Code and install the following extensions:
        - PHP IntelliSense
        - MySQL
        - HTML CSS Support
    3. Open the project folder within VS Code.

2. Design MySQL Database

Create Database Schema:

Use MySQL Workbench:

1. Download and install MySQL Workbench.
2. Open MySQL Workbench and connect to your local MySQL server.
3. Create a new database by running the following SQL script in MySQL Workbench:

```
CREATE DATABASE lyon_housing;
USE lyon_housing;
CREATE TABLE User (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE Property (
    property_id INT AUTO_INCREMENT PRIMARY KEY,
    prediction_id INT,
    address VARCHAR(255) NOT NULL,
    city VARCHAR(100) NOT NULL,
    province VARCHAR(100) NOT NULL,
    postal_code VARCHAR(20) NOT NULL,
    latitude DECIMAL(10, 7),
    longitude DECIMAL(10, 7),
    property_type VARCHAR(50),
    bedrooms INT,
```

```
    bathrooms INT,
    square_footage INT,
    current_value FLOAT,
    FOREIGN KEY (prediction_id) REFERENCES Prediction
(prediction_id)
);
CREATE TABLE Prediction (
    prediction_id INT AUTO_INCREMENT PRIMARY KEY,
    property_id INT,
    price_history JSON,
    prediction_airesponse STRING,
    predicted_price int,
    FOREIGN KEY (property_id) REFERENCES User(property_id)
);
```

Database Relationships and Indexes:

- **Define Relationships:**
    - The Search_History table contains a user_id foreign key referencing the User table.
    - The AI_Prediction table contains a user_id foreign key referencing the Property table.
- **Verify Database Schema:**

  Ensure all tables and relationships are correctly implemented by running the following query to show the table structure:

```
```sql
SHOW TABLES;
DESCRIBE User;
DESCRIBE Property;
```
```

```
DESCRIBE Prediction;
```

Verify the Database Setup:

1. **Write a PHP script to connect to the MySQL database:**

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "lyon_housing";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

2. **Place the PHP script in the htdocs directory of XAMPP:**
    - Save the file as db_test.php.
    - Place it in the htdocs folder inside the XAMPP installation directory.
3. **Run the script in the browser:**

- Go to http://localhost/db_test.php.

- Ensure it prints "Connected successfully" indicating the database connection is properly configured.

3. Summary of Phase 1

Summary of Phase 1	
Step 1	Install XAMPP and configure the local web server environment.
Step 2	Set up Visual Studio Code with necessary plugins for development.
Step 3	Design the database schema using MySQL Workbench and create essential tables.
Step 4	Define and create indexes to optimize database performance.
Step 5	Verify the database setup by connecting to it through a PHP script.

Phase 2: Front-End Development

In this phase, the focus is on developing the user interface with HTML, CSS, and JavaScript.

---

1. Develop Basic HTML Structure

Create Main Pages:

Homepage (`index.php`):

```
<!DOCTYPE html>
   <html lang="en">
   <head>
       <meta charset="UTF-8">
       <meta name="viewport" content="width=device-width,
initial-scale=1.0">
       <title>LYON AI Housing Investment</title>
       <link rel="stylesheet" href="css/styles.css">
   </head>
   <body>
       <header>
```

```
            <h1>LYON AI Housing Investment</h1>
            <nav>
                <ul>
                    <li><a href="index.php">Home</a></li>
                    <li><a href="search.php">Search</a></li>
                    <li><a href="profile.php">Profile</a></li>
                </ul>
            </nav>
        </header>
        <main>
            <h2>Welcome to LYON AI Housing Investment</h2>
            <p>Find the best properties with AI-enhanced
predictions.</p>
            <form action="search.php" method="get">
                <label for="search">Search Properties:</label>
                <input type="text" id="search" name="search">
                <button type="submit">Search</button>
            </form>
        </main>
        <footer>
            <p>&copy; 2025 LYON AI Housing Investment</p>
        </footer>
    </body>
    </html>
```

Search Page (`search.php`):

```
<!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Search Properties</title>
        <link rel="stylesheet" href="css/styles.css">
    </head>
    <body>
        <header>
            <h1>Search Properties</h1>
            <nav>
                <ul>
```

```
                    <li><a href="index.php">Home</a></li>
                    <li><a href="search.php">Search</a></li>
                    <li><a href="profile.php">Profile</a></li>
                </ul>
            </nav>
        </header>
        <main>
            <h2>Search Results</h2>
            <!-- This section will be dynamically populated with
search results using PHP -->
        </main>
        <footer>
            <p>&copy; 2025 LYON AI Housing Investment</p>
        </footer>
        <script src="js/search.js"></script>
    </body>
    </html>
```

Property Details Page (`property.php`):

```
<!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Property Details</title>
        <link rel="stylesheet" href="css/styles.css">
    </head>
    <body>
        <header>
            <h1>Property Details</h1>
            <nav>
                <ul>
                    <li><a href="index.php">Home</a></li>
                    <li><a href="search.php">Search</a></li>
                    <li><a href="profile.php">Profile</a></li>
                </ul>
            </nav>
        </header>
        <main>
```

```
            <h2>Details</h2>
            <!-- This section will be dynamically populated with
property details using PHP -->
        </main>
        <footer>
            <p>&copy; 2025 LYON AI Housing Investment</p>
        </footer>
    </body>
    </html>
```

User Profile Page (`profile.php`):

```
<!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>User Profile</title>
        <link rel="stylesheet" href="css/styles.css">
    </head>
    <body>
        <header>
            <h1>User Profile</h1>
            <nav>
                <ul>
                    <li><a href="index.php">Home</a></li>
                    <li><a href="search.php">Search</a></li>
                    <li><a href="profile.php">Profile</a></li>
                </ul>
            </nav>
        </header>
        <main>
            <h2>Profile Information</h2>
            <!-- This section will be dynamically populated with
user details using PHP -->
            <h2>Saved Searches</h2>
            <!-- This section will display the user's saved
searches -->
        </main>
        <footer>
```

```
            <p>&copy; 2025 LYON AI Housing Investment</p>
        </footer>
    </body>
    </html>
```

---

2. Apply CSS Styling

Design Visual Theme:

Create a `styles.css` file in the `css` folder:

For Example:

```
body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 0;
        background-color: #f4f4f4;
    }
    header {
        background-color: #4CAF50;
        color: white;
        padding: 1em;
        text-align: center;
    }
    nav ul {
        list-style-type: none;
        padding: 0;
    }
    nav ul li {
        display: inline;
        margin: 0 10px;
    }
    nav ul li a {
        color: white;
        text-decoration: none;
    }
    main {
        padding: 1em;
    }
    footer {
```

```
        background-color: #333;
        color: white;
        text-align: center;
        padding: 1em;
        position: fixed;
        width: 100%;
        bottom: 0;
    }
    form label {
        display: block;
        margin-bottom: 0.5em;
    }
    form input, form button {
        padding: 0.5em;
        margin-bottom: 1em;
    }
```

Responsive Design:

Add Media Queries in `styles.css`:

For Example:

```
@media (max-width: 600px) {
        nav ul li {
            display: block;
            margin: 5px 0;
        }
    }
```

---

3. Implement Client-Side Interactivity

Form Validation and Interactivity:

Create a `search.js` file in the `js` folder:

```
document.addEventListener("DOMContentLoaded", function() {
        const searchForm = document.querySelector("form");
        const searchInput = document.querySelector("#search");

        searchForm.addEventListener("submit", function(e) {
            if (searchInput.value.trim() === "") {
```

```
                e.preventDefault();
                alert("Please enter a search term.");
            }
        });
    });
```

Dynamic Content Updates with AJAX (optional):

Update `search.php` to include AJAX functionality:

```
<main>
        <h2>Search Results</h2>
        <div id="results"></div>
    </main>
    <script>
    document.addEventListener("DOMContentLoaded", function() {
        const form = document.querySelector("form");
        form.addEventListener("submit", function(e) {
            e.preventDefault();
            const searchQuery =
document.querySelector("#search").value;
            fetch(`search_results.php?q=${searchQuery}`)
                .then(response => response.json())
                .then(data => {
                    const resultsDiv =
document.querySelector("#results");
                    resultsDiv.innerHTML = "";
                    if (data.length === 0) {
                        resultsDiv.innerHTML = "<p>No results
found.</p>";
                    } else {
                        data.forEach(property => {
                            const propertyDiv =
document.createElement("div");
                            propertyDiv.innerHTML = `
                                <h3>${property.address}</h3>
                                <p>${property.city},
${property.province}</p>
                                <a
href="property.php?id=${property.property_id}">View Details</a>
                            `;
                            resultsDiv.appendChild(propertyDiv);
```

```
                        });
                    }
                });
        });
    });
    </script>
```

Mapping API Integration:

Integrate Leaflet.js for Maps:

- **Include Leaflet.js in the header of your HTML files that require maps:**

```
<link rel="stylesheet"
href="https://unpkg.com/leaflet/dist/leaflet.css" />
     <script
src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
```

- **Add a Div for the map and initialize it in `property.php`:**

```
<div id="map" style="height: 400px; width: 100%;"></div>
     <script>
     const map = L.map('map').setView([latitude, longitude],
13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'
, {
         attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
     }).addTo(map);
     L.marker([latitude, longitude]).addTo(map)
         .bindPopup('Property Location')
         .openPopup();
     </script>
```

---

4. Summary of Phase 2

Summary of Phase 2		
Step 1	Develop HTML Structure	Create main pages with basic HTML layout.

Step 2	Apply CSS Styling	Design a clean visual theme and ensure responsiveness using media queries.
Step 3	Implement Client-Side Interactivity	- Form validation and dynamic content updates using AJAX in JavaScript. - Integrate Leaflet.js for interactive maps on property detail pages.

Phase 3: Back-End Development

This phase covers user authentication, property data CRUD operations, AI prediction integration, and external API interaction.

---

1. User Authentication and Access Control

Registration & Login

- **File Setup:** Create register.php and login.php in project root.

Registration (register.php):

1. Create an HTML form to collect user data (e.g., name, email, password).
2. Use PHP's password_hash() to securely hash the password.
3. Validate inputs and ensure email uniqueness in the database.
4. Insert the user record into the **User** table.
5. Display error messages if validation fails.

Example snippet:

PHP Documentation

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    require_once 'db_connection.php';

    $name = trim($_POST["name"]);
    $email = trim($_POST["email"]);
    $password = $_POST["password"];

    // Validate email and password strength here

    $hashed_password = password_hash($password,
PASSWORD_DEFAULT);

    $stmt = $conn->prepare("INSERT INTO User (name, email,
password) VALUES (?, ?, ?)");
    $stmt->bind_param("sss", $name, $email, $hashed_password);

    if ($stmt->execute()) {
        echo "Registration successful.";
    } else {
        echo "Error: " . $stmt->error;
    }
    $stmt->close();
    $conn->close();
}
?>
```

Login (login.php):

1. Create an HTML form for email and password.

2. Retrieve the stored hash for the submitted email.
3. Use password_verify() to check the password.
4. Validate and start a session upon successful authentication.
5. Redirect to a secure page if login is successful.

Example snippet:

```
<?php
session_start();
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    require_once 'db_connection.php';

    $email = trim($_POST["email"]);
    $password = $_POST["password"];

    $stmt = $conn->prepare("SELECT user_id, password FROM User
WHERE email = ?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $stmt->bind_result($user_id, $hashed_password);

    if ($stmt->fetch() && password_verify($password,
$hashed_password)) {
        $_SESSION["user_id"] = $user_id;
        header("Location: profile.php");
        exit();
    } else {
        echo "Invalid login credentials.";
    }
    $stmt->close();
    $conn->close();
```

```
}
?>
```

Session Management and Access Control

- Start sessions at the top of each protected PHP page.
- Check for the existence of $_SESSION["user_id"] to determine if a user is logged in.
- Redirect unauthenticated users to the login page.

---

2. Data Handling Scripts (CRUD Operations)

Property Data Management

Implement separate scripts for inserting, updating, retrieving, and deleting property records.

Create Property (addProperty.php):

1. Validate property details from an HTML form.
2. Use prepared statements to insert data into the **Property** table.

Example snippet:

```
<?php
require_once 'db_connection.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $address = trim($_POST["address"]);
    $city = trim($_POST["city"]);
    $province = trim($_POST["province"]);
    $postal_code = trim($_POST["postal_code"]);
    // Additional property fields...

    $stmt = $conn->prepare("INSERT INTO Property (address, city,
province, postal_code) VALUES (?, ?, ?, ?)");
```

```
    $stmt->bind_param("ssss", $address, $city, $province,
$postal_code);

    if ($stmt->execute()) {

        echo "Property added successfully.";

    } else {

        echo "Error: " . $stmt->error;

    }

    $stmt->close();

    $conn->close();

}

?>
```

Update Property (updateProperty.php):

1. Retrieve the existing property data.
2. Validate updates received from an HTML form.
3. Use UPDATE SQL command with prepared statements.

Retrieve Property (getProperty.php):

1. Use the property ID from GET parameters.
2. Query the **Property** table using a prepared statement.
3. Return data in JSON format for AJAX or populate HTML elements.

Delete Property (deleteProperty.php):

1. Use a POST or GET request to specify the property.
2. Confirm deletion, then use a DELETE SQL query.

---

3. AI Prediction Model Integration

JSON Handling for AI Predictions

Data Preparation:

1. Collect historical price data and other relevant metrics.
2. Format these details into a JSON object.

API Endpoint for AI Integration (aiPredict.php):

1. Receive property data via POST and encode it as JSON.
2. Send data to your AI prediction model API (or local service).
3. Decode the JSON response from the AI service that contains predicted price and additional advice.
4. Update the **Prediction** table accordingly by mapping prediction results with property data.

Example snippet:

```
<?php
require_once 'db_connection.php';

$property_id = $_POST["property_id"];
// Assume price history data is collected
$price_history = json_encode($_POST["price_history"]);

$ai_api_url = "https://api.your-ai-service.com/predict";
$postData = json_encode([
    "property_id" => $property_id,
    "price_history" => $price_history
]);

$opts = [
    "http" => [
        "method" => "POST",
        "header" => "Content-Type: application/json\r\n",
        "content" => $postData
    ]
];
```

```
$context = stream_context_create($opts);
$response = file_get_contents($ai_api_url, false, $context);
$ai_result = json_decode($response, true);

if (isset($ai_result["predicted_price"])) {
    $predicted_price = $ai_result["predicted_price"];
    $prediction_airesponse = $ai_result["explanation"] ?? "";

    $stmt = $conn->prepare("INSERT INTO Prediction (property_id,
price_history, prediction_airesponse, predicted_price) VALUES
(?, ?, ?, ?)");
    $stmt->bind_param("issi", $property_id, $price_history,
$prediction_airesponse, $predicted_price);

    if ($stmt->execute()) {
        echo "AI prediction recorded successfully.";
    } else {
        echo "Error: " . $stmt->error;
    }
    $stmt->close();
} else {
    echo "AI prediction failed.";
}

$conn->close();
?>
```

- **Notes:**
    - Validate and sanitize all incoming data.

- Ensure error handling is robust.
- Test with sample JSON payloads.

---

4. API Integration for External Data

Connecting to External APIs

Use Case:

For example, fetching property valuation data from an external service like Zillow Zestimate.

Implementation Steps:

1. Identify and register for access to the external API.
2. Read the API documentation to understand parameters, authentication, and rate limits.
3. In PHP, use file_get_contents() or cURL to send requests.
4. Decode JSON responses and parse relevant data.
5. Store or merge external data with existing property data as needed.

Example using cURL:

```
<?php
$external_api_url = "https://api.external-service.com/property";
$query_params = http_build_query([
    "property_id" => $property_id,
    "apikey" => "your_api_key"
]);
$url = $external_api_url . "?" . $query_params;

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
```

```
$api_response = curl_exec($ch);
curl_close($ch);

$external_data = json_decode($api_response, true);
if (isset($external_data["valuation"])) {
    // Process and store the external valuation data
    echo "External valuation: " . $external_data["valuation"];
}
?>
```

- **Security Considerations:**
  - Keep API keys secure and do not expose them in frontend scripts.
  - Implement error checking and fallback mechanisms in case of API unavailability.

Helpful Resources

- PHP Manual
- MySQL Documentation
- Leaflet.js Documentation
- cURL Guide

---

5. Summary of Phase 3

Summary of Phase 2		
Step 1	User Authentication and Access Control	Develop PHP scripts for user registration and login.
Step 2	Data Handling (CRUD Operations)	Create backend scripts to add, update, retrieve, and delete property records
Step 3	AI Prediction Model Integration	Build an code that collects relevant property data, formats it in JSON, calls an external or internal AI

		prediction service, processes the JSON response, and updates the tables accordingly.
Step 4	External API Integration	Connect with third-party APIs, decode the JSON response, and integrate the data into database.