PC PARTS PICKER

A Mini Project Report submitted by

ALWIN K J
(VAS23MCA-2007)

to APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the degree of
Master of Computer Applications

Department of Computer Applications
Vidya Academy of Science & Technology
Thalakkottukara, Thrissur - 680 501
November 2024

Department of Computer Applications

Vidya Academy of Science & Technology

Thalakkottukara, Thrissur - 680 501

(`http://www.vidyaacademy.ac.in`)

CERTIFICATE

This is to certify that the report titled **PC PARTS PICKER** is a bona-fide record of the work related to the paper 20MCA245 Mini Project done by

ALWIN K J (Reg. No. VAS23MCA-2007)

of S3 MCA (2023 admissions) class of Vidya Academy of Science & Technology, Thrissur - 680501 in partial fulfillment of the requirement for the award of the Degree of Master of Computer Applications of APJ Abdul Kalam Technological University.

Guide/Supervisor		**Head of Department**	
Name	: Mr. Manesh. D	Name	: Dr. Reji C Joy
Signature	:	Signature	: ....
Date	: November 13, 2024	Date	: November 13, 2024

(Seal of Department of Computer Applications)

Declaration

I, **ALWIN K J** , studying in Third Semester MCA (2023 admissions) class of Vidya Academy of Science & Technology, Thrissur – 680501, hereby declare that the project report ("PC PARTS PICKER") submitted by me is a bona fide work done by me under supervision of Mr. Manesh. D. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place	: Thrissur – 680501	Signature of student	: ....
Date	: November 13, 2024	Name of student	: ALWIN K J

Acknowledgment

I wish to record my indebtedness and thankfulness to all who helped me prepare this Report titled "PC PARTS PICKER " and present it in a satisfactory way. This Report is part of my work related to the paper 20MCA245 Mini Project.

I am especially thankful to my guide and supervisor Mr. Manesh. D in the Department of Computer Applications for giving me valuable suggestions and critical inputs in the preparation of this report. I am also thankful to Dr. Reji C Joy, the Head of Computer Applications Department for encouragement.

My friends in my class have always been helpful and I am grateful to them for patiently listening to my presentations on my work related to the Project.

ALWIN K J
(Reg. No. VAS23MCA-2007)
S3 MCA (2023 Admissions)
Vidya Academy of Science & Technology
Thrissur -680 501.

Synopsis

PC Part Picker is a comprehensive online platform designed to assist users in building custom personal computers. The website provides a user-friendly interface where users can select, compare, and purchase individual components for their PC builds. It offers a range of features to streamline the PC building process, including compatibility checks, price comparisons, and community-driven build guides.

Contents

List of Tables

List of Figures

PROJECT REPORT

Chapter 1

INTRODUCTION

The PC Parts Picker is a Django-based tool developed to assist users in designing custom PC configurations with ease. The application allows users to select and combine components—such as CPUs, GPUs, RAM, and storage drives—while automatically checking for compatibility among parts. With detailed specifications, real-time price calculation, and the option to download a PDF summary of their selected build, users are guided through the complex process of PC building. This project aims to make custom PC configuration accessible for both beginners and experienced PC enthusiasts by providing a streamlined and error-free experience.

1.1 Problem Definition

Building a custom PC can be challenging, especially for beginners, due to the complexities of component compatibility, technical specifications, and budgeting. Users must ensure that parts such as CPUs, GPUs, motherboards, and RAM are compatible to avoid issues in performance or functionality. Furthermore, manually comparing specifications and prices across components can be time-consuming and confusing.

Therefore, there is a need for a user-friendly, web-based tool that simplifies the PC building process by providing compatibility checks, detailed specifications, and a price calculation feature. This tool should guide users in selecting compatible parts, allow them to view a summary of their build, and enable easy download or printing of the configuration for future reference

1.2 Objectives

The objective of the PC Parts Picker is to develop an intuitive, web-based platform that simplifies the custom PC building process. The tool is designed to assist users in selecting essential components—such as CPUs, GPUs, RAM, and storage—by providing a structured interface that guides them through each part of the configuration. To prevent compatibility issues, the application includes automatic checks to ensure that selected components work seamlessly together. Additionally, it displays detailed specifications and prices for each component, allowing users to make informed decisions and manage their budget effectively. The tool also calculates the total cost of the build in real time and offers users the option to download a PDF summary of their final configuration for easy reference or sharing. Overall, this project aims to make PC building more accessible, accurate, and efficient for both beginners and experienced enthusiasts.

1.3 Purpose of the Project

The purpose of the PC Parts Picker is to provide a streamlined and accessible solution for individuals interested in building custom PCs. By offering a centralized platform that guides users through the selection and configuration of compatible components, this project aims to simplify the often complex and technical process of custom PC assembly. It eliminates common challenges, such as ensuring component compatibility and calculating total build costs, which can be daunting for beginners. Additionally, the application aims to enhance the user experience by providing detailed specifications and real-time budget tracking, making it easier for users to make informed decisions. Ultimately, this project serves to empower users to create personalized, functional, and cost-effective PC builds with confidence.

Chapter 2

SYSTEM ANALYSIS

2.1 The Existing System

In the existing system, individuals interested in building custom PCs often rely on manual research across multiple websites and resources. This process typically involves visiting different vendor sites to compare prices, specifications, and compatibility information for components like CPUs, GPUs, RAM, and motherboards. Compatibility information, in particular, can be complex to verify, as users must ensure that selected parts work together without issues. For example, pairing an incompatible CPU and motherboard or choosing insufficient power for the system can lead to performance problems or even hardware failure. Additionally, the existing system requires users to manually calculate the total cost of their build, which can be tedious and prone to error.

Moreover, while some online tools offer basic compatibility checking or price comparison, they are often limited in scope, lack detailed specifications, and may not provide a comprehensive, user-friendly interface for managing and visualizing the entire build. Consequently, this fragmented approach can make the custom PC building process overwhelming and time-consuming, especially for beginners who may not have in-depth technical knowledge. Therefore, there is a need for a more integrated, efficient solution that addresses these limitations and provides a seamless experience for users.

2.2 Proposed System

The proposed system, PC Parts Picker, aims to address the limitations of the existing system by providing a comprehensive, user-friendly platform that streamlines the custom PC building process. This web-based application integrates key features to make selecting and configuring components easier, faster, and more accurate. With an intuitive interface, users can browse and choose essential PC parts—such as CPUs, GPUs, RAM, and storage—while the application automatically checks for compatibility, helping to prevent common issues like incompatible CPU and motherboard pairings.

In addition to compatibility checks, the proposed system offers detailed specifications and real-time price updates for each component, allowing users to stay informed and within budget throughout the building process. The application also calculates the total cost of the build automatically, providing users with a clear picture of their expenses as they select parts. Furthermore, users can generate a downloadable PDF summary of their build configuration, making it easy to reference or share with others.

Overall, the proposed system offers a one-stop solution for PC enthusiasts and beginners alike, enabling them to confidently create custom PC builds that meet their performance and budget requirements without the hassle of manual research and calculations.

Chapter 3

FEASIBILITY STUDY

3.1 Technical Feasibility

The proposed PC Parts Picker is technically feasible, utilizing standard web technologies such as HTML, CSS, JavaScript, and Python (Django) for the backend development. The system will rely on a secure database management system like PostgreSQL or MySQL to store and manage component information, user data, and configuration details. The application will be hosted on cloud platforms such as AWS or Heroku, offering scalability, high availability, and easy maintenance. This cloud infrastructure will also ensure seamless integration and smooth performance as user demand grows.

Additionally, the web application will be responsive, ensuring that users can access the platform across various devices including desktops, laptops, tablets, and smartphones. The use of Django's robust framework allows for efficient handling of requests, data management, and rendering dynamic content, while ensuring security features such as user authentication and secure data transactions. Given the availability of current web technologies, the development of the PC Parts Picker can be efficiently achieved within existing technological capabilities.

3.2 Economical Feasibility

Economically, the PC Parts Picker is a viable solution as it reduces the time and effort required for users to manually research, compare, and configure PC components. The automation of compatibility checks, real-time price calculations, and streamlined configuration process minimizes errors, such as selecting incompatible parts or overspending on components. While the initial development will require investment in terms of time, resources, and hosting infrastructure, the long-term benefits include reduced customer support costs, improved user experience, and a more efficient PC building process.

The application's potential to attract a wide user base—ranging from beginners to PC enthusiasts—also presents opportunities for monetization, such as through affiliate marketing with component vendors or offering premium features. Additionally, the application's scalability ensures that it can handle increased traffic without significant additional costs. The long-term savings, increased customer satisfaction, and the ability to tap into the growing market of custom PC builders make the project a sound investment with substantial financial returns.

3.3 Operational Feasibility

The PC Parts Picker is operationally feasible, as it automates and streamlines the custom PC building process, reducing manual work for users and administrators. With an intuitive, user-friendly design, the application allows users to easily browse, select, and configure compatible PC components independently. The platform's compatibility checks and real-time price calculations ensure a smooth and error-free experience, enhancing user satisfaction.

On the administrative side, the system integrates seamlessly with the existing infrastructure, allowing for efficient management of component data, user profiles, and build configurations. The application's backend, built on Django, provides easy maintenance and minimal overhead for administrators. Additionally, the system's responsive design ensures that it can be accessed and operated across various devices, reducing the need for specialized hardware or software. The ease of use for both users and administrators ensures smooth adoption, with minimal training required for staff to manage the system effectively.

Chapter 4

SYSTEM DESIGN

4.1 Input Design

Input Design for the PC Parts Picker focuses on creating a user-friendly interface that allows users to input their preferences and select components for their custom PC builds. The system will take various forms of input, including user selections, configuration details, and payment information.

4.1.1 User registration form design

```
{% load static %}
<body>

    <div class="container">
        <div class="card">
            <a class="register">Sign Up</a>
            <form id="signupForm" method="post">
                {% csrf_token %}
                <!-- Username input -->
                <div class="inputBox">
                    <input type="text" name="username"
                        placeholder="" required="required">
                    <span>Username</span>
                </div>
                <!-- Password input -->
                <div class="inputBox">
                    <input type="password" id="password" name="
                        password" placeholder="" required="
                        required">
                    <span>Password</span>
                </div>
                <!-- Confirm Password input -->
                <div class="inputBox">
                    <input type="password" id="confirmPassword"
                        name="confirmPassword" placeholder=""
                        required="required">
                    <span>Confirm Password</span>
                </div>
                <!-- Error Message -->
                <div id="error-message" class="error-message">
                    Passwords do not match.
                </div>
                <!-- Submit Button -->
                <button type="submit" class="enter">Sign Up</
                    button>
            </form>
            <div class="sign-up-container">
            <p>Already have an account? <a href="{% url 'login'
                %}">Login</a></p>
            </div>
        </div>
    </div>
    <script>
        // Password match validation logic
        document.getElementById('signupForm').addEventListener('
            submit', function(event) {
            var password = document.getElementById('password').
                value;
            var confirmPassword = document.getElementById('
                confirmPassword').value;
            var errorMessage = document.getElementById('error-
                message');

            if (password !== confirmPassword) {
                errorMessage.style.display = 'block';
                event.preventDefault(); // Prevent form
                    submission if passwords don't match
            } else {
                errorMessage.style.display = 'none';
            }
        });
    </script>
```

4.2 Output Design

Output design is the process of planning how the system will present information to users. It involves defining the layout, format, and method of displaying data, reports, and other results generated by the system. The purpose of output design is to ensure that information is presented clearly, effectively, and in a way that supports users' decision-making and understanding.Good output design improves usability, supports informed decision-making, and enhances the overall user experience.

4.2.1 PC Builder

```
{% load static %}
<body>
<!--HEADER START-->
<div class="ie-panel"><a href="http://windows.microsoft.com/en-
    US/internet-explorer/"><img src="{% static 'images/ie8-panel/
    warning_bar_0000_us.jpg' %}" alt="Warning Bar" height="42"
    width="820" alt="You are using an outdated browser. For a
    faster, safer browsing experience, upgrade for free today."><
    /a></div>
<div class="preloader">
  <div class="preloader-body">
    <div class="cssload-container">
      <div class="cssload-speeding-wheel"></div>
    </div>
    <p>Loading...</p>
  </div>
</div>
```

```
<div class="page">
<!-- Page Header-->
<header class="section page-header">
<!-- RD Navbar-->
<div class="rd-navbar-wrap">
<nav class="rd-navbar rd-navbar-wide" data-layout="rd-navbar-
    fixed" data-sm-layout="rd-navbar-fixed" data-md-layout="rd-
    navbar-fixed" data-md-device-layout="rd-navbar-fixed" data-lg
    -layout="rd-navbar-static" data-lg-device-layout="rd-navbar-
    static" data-xl-layout="rd-navbar-static" data-xl-device-
    layout="rd-navbar-static" data-lg-stick-up-offset="46px" data
    -xl-stick-up-offset="46px" data-xxl-stick-up-offset="46px"
    data-lg-stick-up="true" data-xl-stick-up="true" data-xxl-
    stick-up="true">
  <div class="rd-navbar-main-outer">
    <div class="rd-navbar-main">
      <!-- RD Navbar Panel-->
      <div class="rd-navbar-panel">
        <!-- RD Navbar Toggle-->
        <button class="rd-navbar-toggle" data-rd-navbar-toggle="
            .rd-navbar-nav-wrap"><span></span></button>
        <!-- RD Navbar Brand-->
        <div class="rd-navbar-brand">
          <a class="brand" href="{% url 'index' %}">
            <img class="brand-logo-dark" src="{% static 'images/
                logo-default-261x72.png' %}" alt="" width="261"
                height="72" srcset="{% static 'images/logo-
                default-261x72.png' %} 2x"/>
            <img class="brand-logo-light" src="{% static 'images
                /logo-inverse-133x45.png' %}" alt="" width="133"
                height="45" srcset="{% static 'images/logo-
                inverse-266x90.png' %} 2x"/>
          </a>
        </div>
      </div>
```

```
<div class="rd-navbar-nav-wrap">
  <!-- RD Navbar Nav-->
  <ul class="rd-navbar-nav">
    <li class="rd-nav-item active"><a class="rd-nav-link"
        href="{% url 'index' %}">Home</a></li>
    <li class="rd-nav-item"><a class="rd-nav-link" href="
        {% url 'pcbuild' %}">Builder</a></li>
    <li class="rd-nav-item"><a class="rd-nav-link" href="
        {% url 'about_us' %}">About</a></li>
    <li class="rd-nav-item"><a class="rd-nav-link" href="
        {% url 'contacts' %}">Contacts</a></li>

    {% if user.is_authenticated %}
      <!-- User Info Box with Dropdown -->
      <li class="rd-nav-item" style="position: relative;">
        <div class="user-info-box" style="display: flex;
            align-items: center; justify-content: center;
            cursor: pointer;" onclick="toggleDropdown()">
          {% if request.user.profile.profile_picture %}
            <img src="{{ request.user.profile.
                profile_picture.url }}" alt="Profile
                Picture" style="width:50px;height:50px;
                border-radius:50%; margin-right: 10px;
                margin-top: 20px;" />
          {% else %}
            <img src="{% static 'images/default-profile.
                png' %}" alt="Profile Picture" style="width
                :50px;height:50px;border-radius:50%; margin
                -right: 10px; margin-top: 10px;" />
          {% endif %}
          <span style="margin-top: 15px;">{{ user.username
              }}</span>
        </div>
        <!-- Dropdown Menu -->
        <div id="user-dropdown" class="dropdown-menu"
            style="display: none; position: absolute; top:
                    100%; right: 0; background: #fff; border: 1px
                    solid #ddd; padding: 10px; width: 150px; box-
                    shadow: 0 0 10px rgba(0, 0, 0, 0.1);">
                  <a href="{% url 'profile' %}" class="dropdown-
                     item">Manage Profile</a>
                  <form method="post" action="{% url 'logout' %}"
                     class="dropdown-item">
                    {% csrf_token %}
                    <button type="submit" style="background: none;
                        border: none; padding: 0; color: #007bff;"
                      >Logout</button>
                  </form>
                </div>
              </li>
            {% else %}
              <li class="rd-nav-item"><a class="rd-nav-link" href=
                 "{% url 'login' %}">Login</a></li>
              <li class="rd-nav-item"><a class="rd-nav-link" href=
                 "{% url 'signup' %}">Sign up</a></li>
            {% endif %}
          </ul>
        </div>
      </div>
  </div>
</nav>
</div>
</header>
<script>
function toggleDropdown() {
var dropdown = document.getElementById("user-dropdown");
if (dropdown.style.display === "none" || dropdown.style.display
   === "") {
  dropdown.style.display = "block";
} else {
  dropdown.style.display = "none";
}
}
</script>

<center>

      <form action="" method="POST"  id="build-form">
          {% csrf_token %}

          <!-- Build Name Input -->
                      <!-- Build Name Input on Same Row -->
          <div class="form-group">
              <label for="build_name">Build Name:</label>
              <input type="text" id="build_name" name="build_name"
                  required>
          </div>
          <!-- CPU Selection (Button and Dropdown in Single Row)
              -->
          <div class="form-group">
              <div class="input-row">
                  <label for="cpu">Processor:</label>
                  <!-- Image Between Label and Button -->
                  <img src="{% static 'build_ico/1.png' %}" alt="
                      CPU Demo" class="component-image">
                  <button type="button" onclick="openModal('cpu')"
                      >+</button>
                  <select name="cpu" id="cpu" required>
                      <option value="">Select Processor</option>
                      {% for cpu in cpus %}
                          <option value="{{ cpu.id }}">{{ cpu.name
                              }}</option>
                      {% endfor %}
                  </select>
              </div>
          </div>
```

```
<!-- Motherboard Selection (Button and Dropdown in
    Single Row) -->
<div class="form-group">
    <div class="input-row">
        <label for="motherboard">Motherboard:</label>
        <!-- Image Between Label and Button -->
        <img src="{% static 'build_ico/2.png' %}" alt="
            Motherboard Demo" class="component-image">
        <button type="button" onclick="openModal('
            motherboard')">+</button>
        <select name="motherboard" id="motherboard"
            required>
            <option value="">Select Motherboard</option>
            {% for motherboard in motherboards %}
                <option value="{{ motherboard.id }}">{{
                    motherboard.name }}</option>
            {% endfor %}
        </select>
    </div>
</div>

<!-- GPU Selection (Button and Dropdown in Single Row)
    -->
<div class="form-group">
    <div class="input-row">
        <label for="gpu">GPU:</label>
        <!-- Image Between Label and Button -->
        <img src="{% static 'build_ico/3.png' %}" alt="
            GPU Demo" class="component-image">
        <button type="button" onclick="openModal('gpu')"
            >+</button>
        <select name="gpu" id="gpu" required>
            <option value="">Select GPU</option>
            {% for gpu in gpus %}
                <option value="{{ gpu.id }}">{{ gpu.name
                    }}</option>
            {% endfor %}
        </select>
    </div>
</div>

<!-- RAM Selection (Button and Dropdown in Single Row)
   -->
<div class="form-group">
    <div class="input-row">
        <label for="ram">RAM:</label>
        <!-- Image Between Label and Button -->
        <img src="{% static 'build_ico/4.png' %}" alt="
           RAM Demo" class="component-image">
        <button type="button" onclick="openModal('ram')"
           >+</button>
        <select name="ram" id="ram" required>
            <option value="">Select RAM</option>
            {% for ram in rams %}
                <option value="{{ ram.id }}">{{ ram.name
                    }}</option>
            {% endfor %}
        </select>
    </div>
</div>

<!-- SSD Selection (Button and Dropdown in Single Row)
   -->
<div class="form-group">
    <div class="input-row">
        <label for="ssd">SSD:</label>
        <!-- Image Between Label and Button -->
        <img src="{% static 'build_ico/5.png' %}" alt="
           SSD Demo" class="component-image">
        <button type="button" onclick="openModal('ssd')"
           >+</button>
        <select name="ssd" id="ssd" required>
```

```
                <option value="">Select SSD</option>
                {% for ssd in ssds %}
                    <option value="{{ ssd.id }}">{{ ssd.name
                        }}</option>
                {% endfor %}
            </select>
        </div>
    </div>

    <!-- PSU Selection (Button and Dropdown in Single Row)
        -->
    <div class="form-group">
        <div class="input-row">
            <label for="psu">PSU:</label>
            <!-- Image Between Label and Button -->
            <img src="{% static 'build_ico/6.png' %}" alt="
                PSU Demo" class="component-image">
            <button type="button" onclick="openModal('psu')"
                >+</button>
            <select name="psu" id="psu" required>
                <option value="">Select PSU</option>
                {% for psu in psus %}
                    <option value="{{ psu.id }}">{{ psu.name
                        }}</option>
                {% endfor %}
            </select>
        </div>
    </div>

<!-- Buttons Row -->
<div class="form-group" style="display: flex; justify-
    content: space-between; align-items: center;">
    <!-- Add to Build Button -->
    <button class="button button-default" type="submit"
        style="padding: 15px 30px; font-size: 18px; line-
        height: normal;">
```

```
            Build PC
        </button>

        <!-- Print Button -->
        <button class="button button-default" type="button"
            onclick="printBuild();" style="padding: 15px 30px;
            font-size: 18px;">
            Print Build
        </button>

        <!-- Download Button >
        <button class="button button-default" type="button"
            onclick="downloadBuild();" style="padding: 15px 30px;
             font-size: 18px; line-height: normal;">
            Download Build
        </button-->
    </div>
    </form>
    <script>
        // Print function
        function printBuild() {
            var printContent = document.getElementById('build-
                form').innerHTML;
            var printWindow = window.open('', '', 'height=600,
                width=800');
            printWindow.document.write('<html><head><title>Print
                 Build</title></head><body>');
            printWindow.document.write(printContent);
            printWindow.document.write('</body></html>');
            printWindow.document.close();
            printWindow.print();
        }

        // Download function using jsPDF
        function downloadBuild() {
            const { jsPDF } = window.jspdf; // Access jsPDF
            const doc = new jsPDF();

            // Get content inside the form with id 'build-form'
            const content = document.getElementById('build-form
                ').innerHTML;

            // Add HTML content to PDF
            doc.html(content, {
                callback: function (doc) {
                    // Save the generated PDF
                    doc.save('build.pdf');
                },
                margin: [10, 10, 10, 10],  // Set margins for
                    the PDF
                x: 10,
                y: 10
            });
        }
    </script>
</body>

</html>
```

4.2.2 User Profile Manager

```
{% load static %}

<body>
<div class="profile-container">

    <h2>User Profile manager</h2>

    <!-- Profile Form -->
    <form method="post" enctype="multipart/form-data" class="
        profile-form">
         {% csrf_token %}
         <div class="form-group">
```

```
            {{ form.as_p }}
        </div>
        <button type="submit" class="btn btn-primary">Update
            Profile</button>
    </form>

    <!-- Profile Picture Section -->
    <div class="profile-picture">
        {% if request.user.profile.profile_picture %}
            <img src="{{ request.user.profile.profile_picture.
                url }}" alt="Profile Picture" class="profile-img"
                />
        {% else %}
            <p>No profile picture available.</p>
        {% endif %}
    </div>

    <!-- User Builds Section -->
    <section class="user-builds">
        <h3>Your Builds</h3>
        {% if builds %}
            <ul class="build-list">
                {% for build in builds %}
                    <li class="build-item">
                        <strong>{{ build.name }}</strong>
                        <div class="build-details">
                            <p><strong>CPU:</strong> {{ build.cpu
                                .name }}</p>
                            <p><strong>Motherboard:</strong> {{
                                build.motherboard.name }}</p>
                            <p><strong>GPU:</strong> {{ build.gpu
                                 }}</p>
                            <p><strong>RAM:</strong> {{ build.ram
                                 }}</p>
                            <p><strong>SSD:</strong> {{ build.ssd
                                 }}</p>
                                <p><strong>PSU:</strong> {{ build.psu
                                    }}</p>
                                 <small>Created on: {{ build.
                                    created_at }}</small>
                            </div>
                        </li>
                    {% endfor %}
                </ul>
            {% else %}
                <p>No builds created yet.</p>
            {% endif %}
        </section>
</div>
        <!-- Go Home Button -->
        <center>
            <div class="col-sm-10 text-sm-center">
                <a class="button button-default">
                    Home Page
                </a>
            </div>
            <div class="col-sm-10 text-sm-center">
                <a class="button button-default" >
                    PC Builder
                </a>
            </div>
        </center>
</body>
```

4.3 Database Design

4.3.1 Data Flow Diagram

A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system. It is commonly used in systems analysis and design to illustrate how data moves through processes, where it is stored, and how it interacts with various components or users. Components of a Data Flow Diagram are Processes: Represented by circles or rounded rectangles, processes transform incoming data flows into outgoing data flows. They perform some action or computation on the data. Data Stores: Represented by open-ended rectangles, these hold data that is used or produced by the processes. They act as repositories where data can be stored and retrieved. External Entities: Represented by squares, these are sources or destinations of data outside the system, like users or other systems. They interact with the system by sending data to or receiving data from it. Data Flows: Represented by arrows, these show the movement of data between processes, data stores, and external entities. They indicate how information moves within the system.

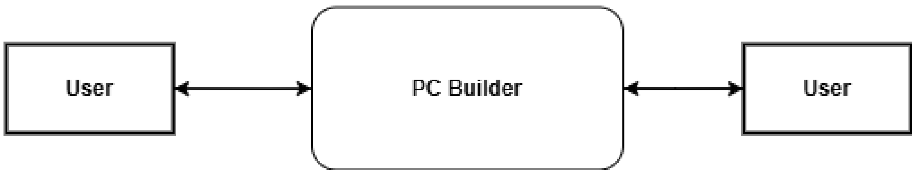


Figure 4.1: DFD LEVEL 0

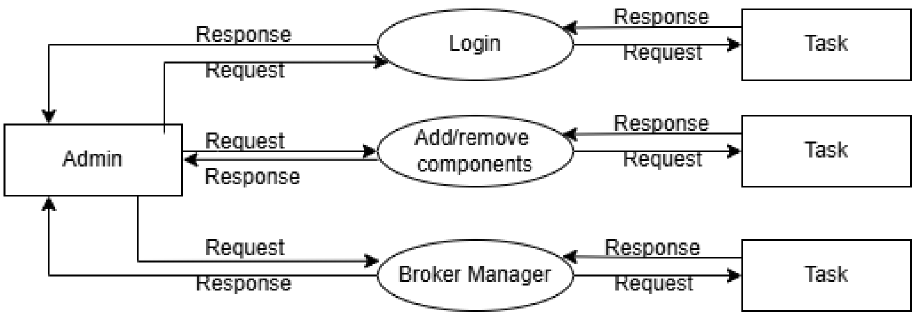


Figure 4.2: DFD LEVEL 1.1

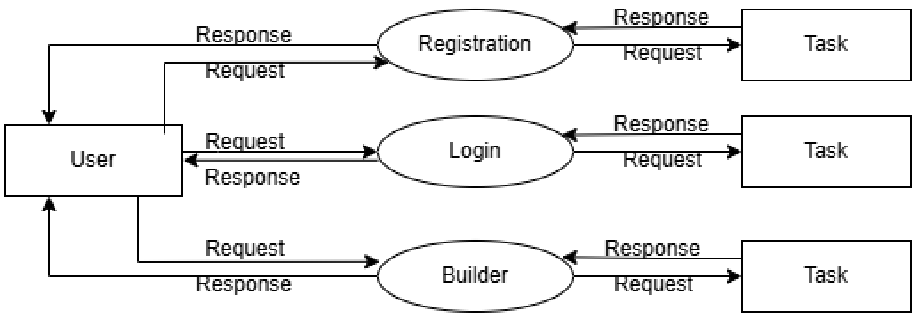


Figure 4.3: DFD LEVEL- 1.2

Chapter 5

IMPLEMENTATION, TESTING AND VALIDATION

5.1 Implementation

The implementation phase is where the PC parts picker application evolves into a functional product. Its success depends on having clear requirements, a well-structured design, and systematic coding practices that result in a reliable, maintainable, and scalable system. For a project like a PC parts picker, this phase is about turning plans into actionable features, allowing users to browse, compare, and select compatible PC components to build their custom PC configurations effectively.

In software development, the implementation phase is when developers take design specifications and convert them into working software. This process involves writing, assembling, and integrating code to meet the project's requirements, ensuring that each feature delivers the intended functionality.

In this phase, the PC parts picker application is built out feature by feature. The main components include:

Frontend (User Interface): The frontend is designed to provide users with an intuitive interface for browsing and selecting parts, including categories like CPUs, GPUs, RAM, motherboards, and storage. This user interface allows users to view detailed specifications, prices, and compatibility options for each part.

Backend (Server and Database Interactions): The backend handles requests from the frontend, fetches data from the database, and ensures that component compatibility rules are enforced. It manages user actions, such as adding parts to a build, saving configurations, and calculating the total cost.

Business Logic: The business logic ensures compatibility between selected components (e.g., Intel CPUs with Intel motherboards or AMD CPUs with AMD motherboards). It includes rules for power requirements, size constraints, and other compatibility checks essential for building a fully functional PC. This layer also supports features like total price calculation, which automatically updates as users add or remove parts from their build.

Through careful implementation of these components, the PC parts picker application becomes a powerful tool, enabling users to explore, configure, and customize their PC builds confidently. The systematic approach in the implementation phase ensures that the application functions as intended, providing a smooth and reliable experience for users.

5.2 Testing Methods

5.2.1 Unit Testing

Unit testing is a type of software testing that focuses on verifying individual units or components of an application in isolation. In the context of software, a ”unit” refers to the smallest testable part, such as a function, method, or class. The purpose of unit testing is to ensure that each unit performs as expected independently of the other parts of the system.

For a PC parts picker application, unit testing is a critical practice to maintain high-quality and reliable code. By testing individual functions or methods in isolation, unit tests help identify bugs early and confirm that each component behaves as expected.In the PC parts picker, unit tests would focus on essential functions, such as:Compatibility Checker: Ensures that only compatible parts (e.g., CPU and motherboard) can be selected togetherPrice Calculation: Verifies that the total cost is accurately calculated as users add or remove parts.Component Selection: Checks that parts can be selected, deselected, and updated in the configuration without errors.Unit testing these functions ensures that the core features of the PC parts picker work correctly, providing users with a seamless experience and reducing the risk of unexpected errors during the build process.

5.3 Integration Testing

Integration testing ensures that different components or modules of an application work together as intended. It focuses on testing the interactions between integrated units to verify data flow, functionality, and interfaces. The goal is to detect issues related to communication between modules, such as incorrect data handling or broken API calls. It can be performed incrementally or using top-down, bottom-up, or big bang approaches. By identifying integration issues early, this testing improves the overall reliability of the system. Successful integration testing ensures a seamless interaction between all components before the final deployment.

5.4 White Box Testing

White box testing is a software testing technique where the internal structure, design, and implementation of the code are examined and tested. This method requires testers to have knowledge of the code, allowing them to create test cases that cover specific functions, logic, paths, and conditions. It ensures that each line, branch, and pathway of the code functions correctly and meets design specifications. White box testing helps in identifying issues related to code efficiency, logic errors, and security vulnerabilities. Common techniques include path testing, loop testing, and control flow testing. This approach is often used in unit testing and integration testing to improve code quality and robustness.

5.5 Black Box Testing

Black box testing is a software testing technique that focuses on evaluating the functionality of an application without examining its internal code structure. Testers assess the software by providing inputs and verifying the outputs, ensuring that the system meets requirements and performs as expected. This method doesn't require knowledge of the code or internal workings, making it ideal for testing from a user's perspective. Black box testing helps identify issues related to user interface, functionality, performance, and usability. Common techniques include equivalence partitioning, boundary value analysis, and decision table testing. It is widely used in system and acceptance testing phases.

5.6 Validation

Validation is a crucial process that verifies whether inputs, systems, or products meet specified standards and perform as expected. It ensures data integrity, consistency, and correctness, especially in fields like software development and data management. In software, validation checks that applications fulfill user requirements and operate as intended. Data validation prevents errors by ensuring accurate formats, such as date or email inputs, before processing. Similarly, product validation in manufacturing confirms that each item meets quality and safety standards for consumers. In industries like pharmaceuticals, process validation is essential to guarantee consistent, reliable outcomes critical for safety. Overall, validation enhances confidence in system reliability, minimizes errors, and upholds high standards across various applications.

5.6.1 validation in user registration form

```
def your_main_view(request):
    # Your view logic here
    return render(request, 'parts/main_page.html')

def signup_view(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = SignUpForm()
    return render(request, 'parts/signup.html', {'form': form})

@login_required
def profile_view(request):
    # Fetch builds for the logged-in user
builds = Build.objects.filter(user=request.user)

    print(f'Builds for {request.user.username}: {builds}')  #
       Log the builds to check

    if request.method == 'POST':
        form = ProfileUpdateForm(request.POST, request.FILES,
           instance=request.user.profile)
        if form.is_valid():
            form.save()
            return redirect('profile')
    else:
        form = ProfileUpdateForm(instance=request.user.profile)

    return render(request, 'parts/profile.html', {
        'form': form,
        'builds': builds,
    })
```

5.6.2 User Build validation

```
@login_required
def pc_build(request):
    if request.method == 'POST':
        build_name = request.POST.get('build_name')
        cpu_id = request.POST.get('cpu')
        motherboard_id = request.POST.get('motherboard')
        gpu_id = request.POST.get('gpu')
        ram_id = request.POST.get('ram')
        ssd_id = request.POST.get('ssd')
        psu_id = request.POST.get('psu')

        # Validate the form inputs
        if not all([cpu_id, motherboard_id, gpu_id, ram_id,
            ssd_id, psu_id]):
             return render(request, 'parts/pc_builder.html', {
                 'cpus': CPU.objects.all(),
                 'motherboards': Motherboard.objects.all(),
                 'gpus': GPU.objects.all(),
                 'ram': RAM.objects.all(),
                 'ssds': SSD.objects.all(),
                 'psus': PSU.objects.all(),
                 'error': 'Please select all components.'
             })

        # Retrieve the selected components
        cpu = CPU.objects.get(id=cpu_id)
        motherboard = Motherboard.objects.get(id=motherboard_id)
        gpu = GPU.objects.get(id=gpu_id)
        ram = RAM.objects.get(id=ram_id)
        ssd = SSD.objects.get(id=ssd_id)
        psu = PSU.objects.get(id=psu_id)

        # Create the build entry in the database
        build = Build.objects.create(
            name=build_name,
            user=request.user,
            cpu=cpu,
            motherboard=motherboard,
            gpu=gpu,
            ram=ram,
            ssd=ssd,
            psu=psu
        )

        # Redirect to the profile page after successful build
            creation
        return redirect('profile')

    # If GET request, render the form
    return render(request, 'parts/pc_builder.html', {
        'cpus': CPU.objects.all(),
        'motherboards': Motherboard.objects.all(),
        'gpus': GPU.objects.all(),
        'ram': RAM.objects.all(),
        'ssds': SSD.objects.all(),
        'psus': PSU.objects.all(),
    })
```

Chapter 6

CONCLUSION

In conclusion, the PC Parts Picker provides an efficient and user-friendly platform for building custom PCs. With features such as user registration, component selection, compatibility checking, real-time pricing, and PDF export, the system caters to the needs of both novice and experienced PC builders. Users can easily browse and select components, configure their builds, and ensure compatibility without the need for extensive technical knowledge, while administrators maintain control over component data and pricing.

The project's well-structured data models enhance data accuracy and streamline processes, such as automatically updating total build costs based on selected components. This digital solution simplifies the PC building experience, offers a comprehensive platform for component selection and compatibility, and promotes organized management of custom PC configurations, ensuring a reliable tool for PC building enthusiasts.

Chapter 7

SCOPE FOR FUTURE ENHANCEMENT

For future enhancements, the PC Parts Picker could incorporate several advanced features to expand functionality and improve user experience. Integrating real-time inventory updates and pricing from various vendors would allow users to view current availability and competitive pricing for components. Implementing AI-driven component recommendations based on user preferences and previous selections could enhance the user experience by suggesting compatible and popular components, making it easier to build optimized configurations.

Additionally, a community-based feature allowing users to rate and review components would provide valuable insights and feedback, helping others make informed choices. Introducing multi-currency support could also widen the platform's reach, accommodating international users by displaying prices in local currencies. Furthermore, integrating more detailed compatibility checks, such as power supply recommendations based on selected components' power requirements, would increase accuracy and reliability.

Finally, implementing automated notifications for build updates or availability changes through email or SMS would improve communication and user engagement. These enhancements would position the PC Parts Picker as a comprehensive, user-centric platform for custom PC building, supporting future growth and ensuring high standards of user satisfaction and operational efficiency.

APPENDICES

Appendix A

SCRUM PROCESS ARTIFACTS

A.1 Product Backlog

Priority	Product backlog items	User story	Estimate (Hours)
1	Database creation	As a developer, I want to store all component and compatibility information for building PCs.	240
2	Login page	As a user, I want to log in to save my custom PC builds.	160
3	Component selection page	As a user, I want to search and select components for my PC build.	400
4	Compatibility checker	As a user, I want to ensure selected components are compatible with each other.	240
5	Build summary page	As a user, I want to view a summary of my selected components and the total price.	120
6	PDF export	As a user, I want to download my build configuration as a PDF.	80

A.2 Scrum Meeting Details

Sl. No.	Date	Sprint backlog	Product shippable Increment	Important Decisions
1	05-08-2024	problem identification	done	Discuss different PC builder platforms
2	06-08-2024	topic selection	done	Decided on PC Parts Picker
3	08-08-2024	requirement elicitation	done	Discussed functional requirements
4	09-08-2024	project proposal preparation	done	Finalized project proposal details
5	12-08-2024	project proposal submission	done	Submitted project proposal
6	20-08-2024	designing of login page	done	Chose login page template
7	01-09-2024	designing component selection pages	done	Designed component selection pages
8	10-09-2024	database creation	done	Discussed database schema for components
9	10-10-2024	coding	done	Reviewed initial code structure
10	02-11-2024	validation checks	done	Discussed validation for component compatibility

Appendix B

DIAGRAMS

B.1 Entity Relationship Diagram

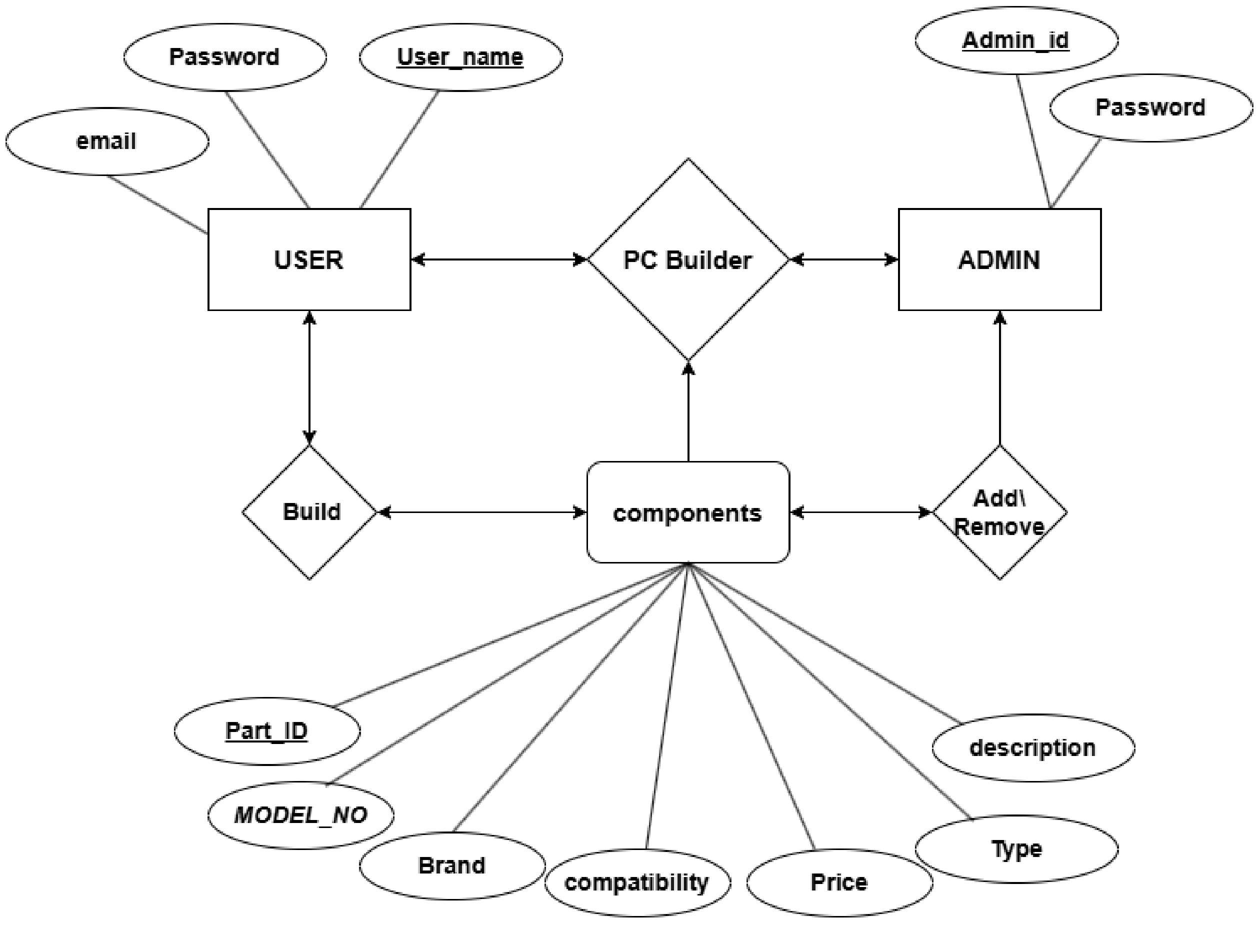


Figure B.1: ER Diagram

Appendix C

TABLE STRUCTURE

Sl. No.	Field	Datatype	Constraint	Description
1	User_name	Varchar(10)	Primary key	User ID
2	Password	Varchar(20)	Not Null	Password for user
3	Email	Varchar(30)	Unique,Not Null	Email address of user

Table C.1: USER

Sl. No.	Field	Datatype	Constraint	Description
1	Admin_id	Varchar(15)	Primary key	Admin ID
2	Password	Varchar(20)	Not Null	Password for Admin

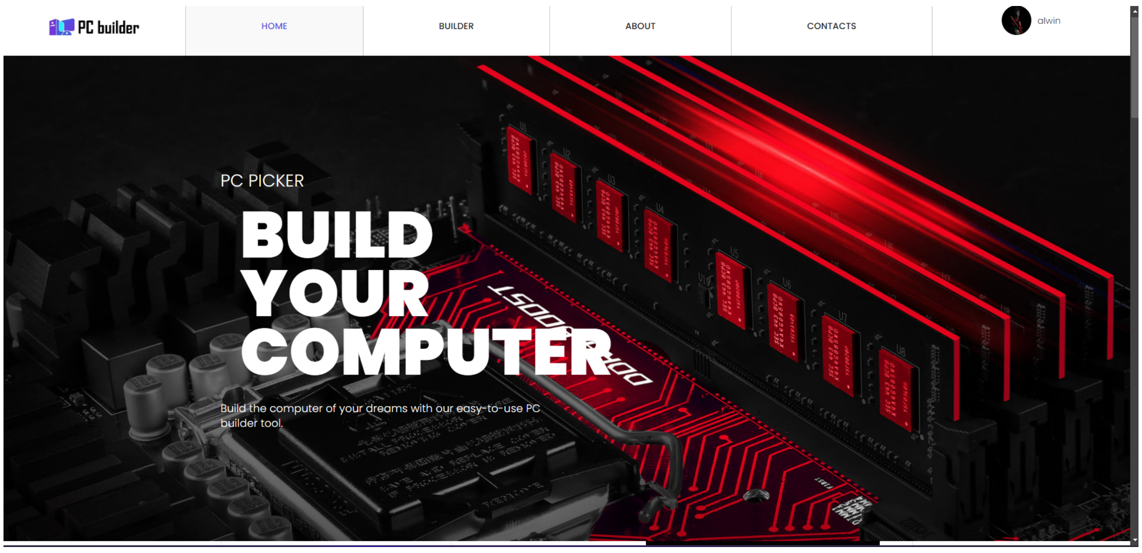
Table C.2: PC Parts Picker Admin

Sl. No.	Field	Datatype	Constraint	Description
1	Part_id	Varchar(10)	Primary key	Part ID
2	Model_no	Varchar(50)	Not Null	Model number of the component
3	Brand	Varchar(50)	Not Null	Brand name
4	Compatibility	Varchar(20)	Not Null	Checks compatibility between parts
5	Price	Int	Not Null	Price of the component
6	Type	Varchar(20)	Not Null	Component type
7	Description	Varchar(400)	Not Null	Description of the parts

Table C.3: Pc Parts Picker Parts DATABASE

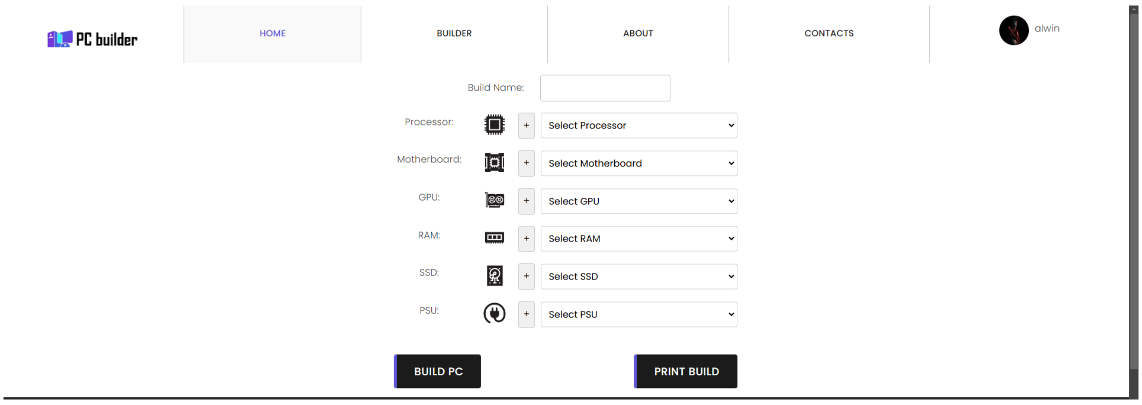
Appendix D

SAMPLE SCREENSHOTS



Build the computer of your dreams with our easy-to-use PC builder tool.

Figure D.1: index page



Build Name:

Processor: + Select Processor

Motherboard: + Select Motherboard

GPU: + Select GPU

RAM: + Select RAM

SSD: + Select SSD

PSU: + Select PSU

Figure D.2: Builder page

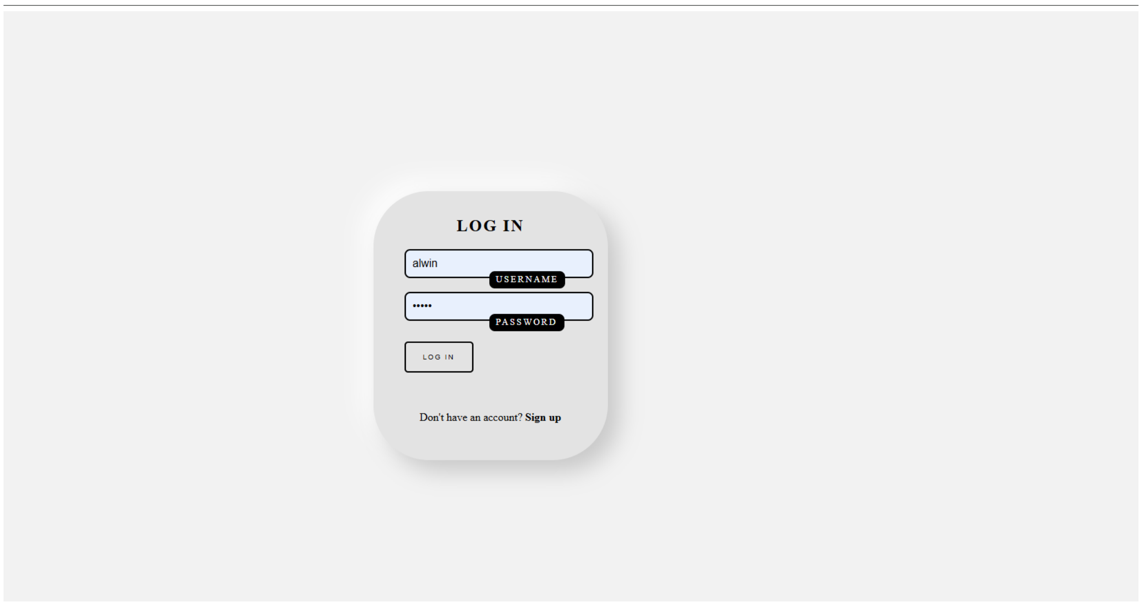


Figure D.3: login page

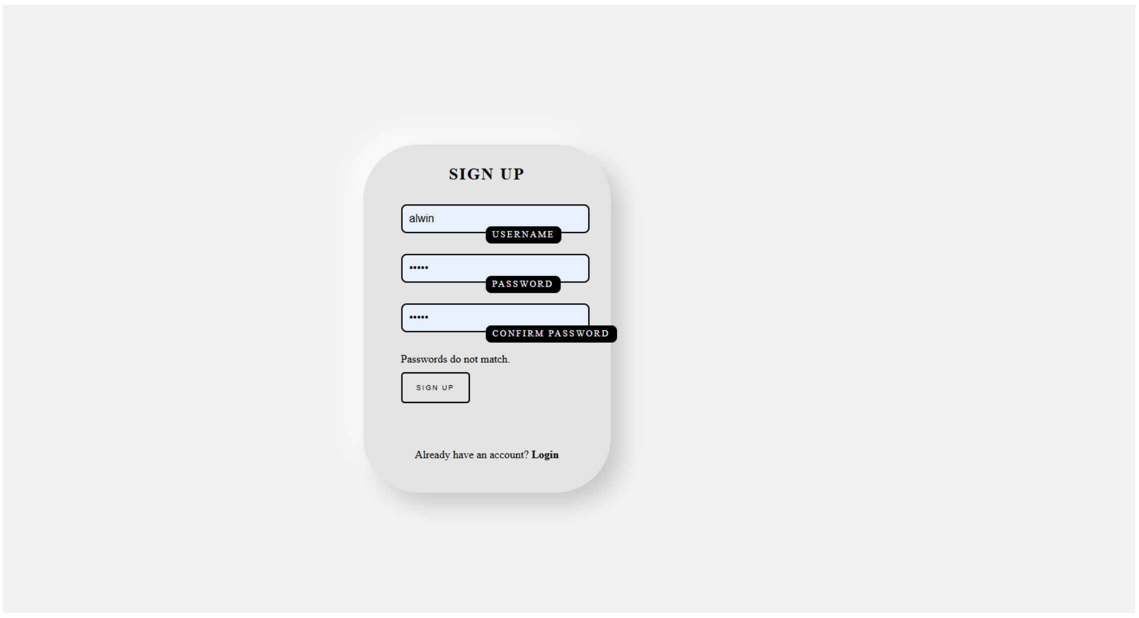
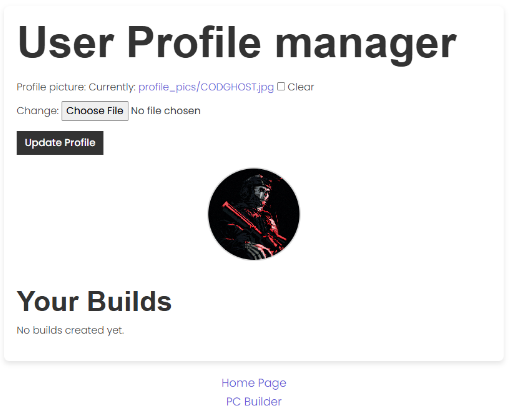


Figure D.4: Register page


User Profile manager

Profile picture: Currently: profile_pics/CODGHOST.jpg ☐ Clear

Change: Choose File No file chosen

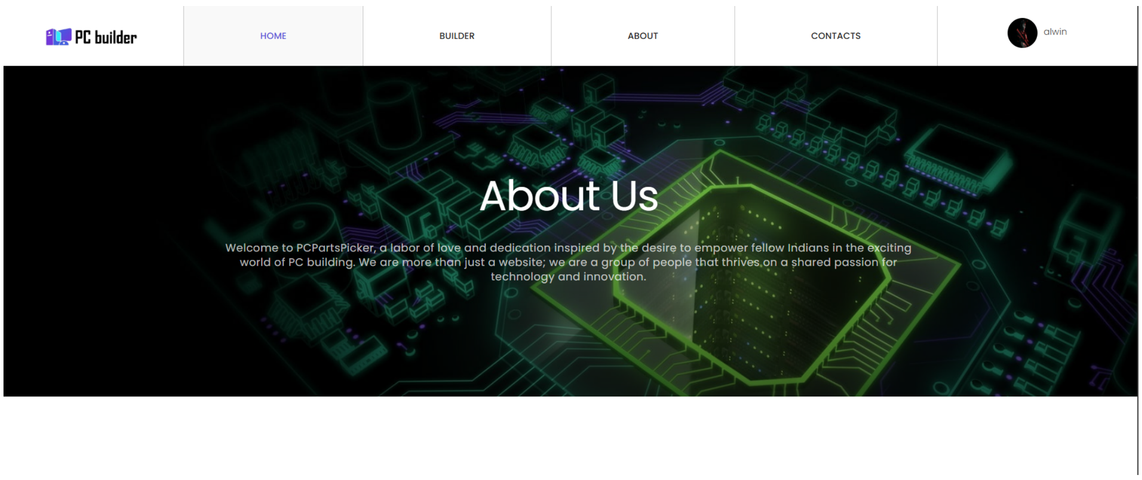
Update Profile

Your Builds

No builds created yet.

Home Page
PC Builder

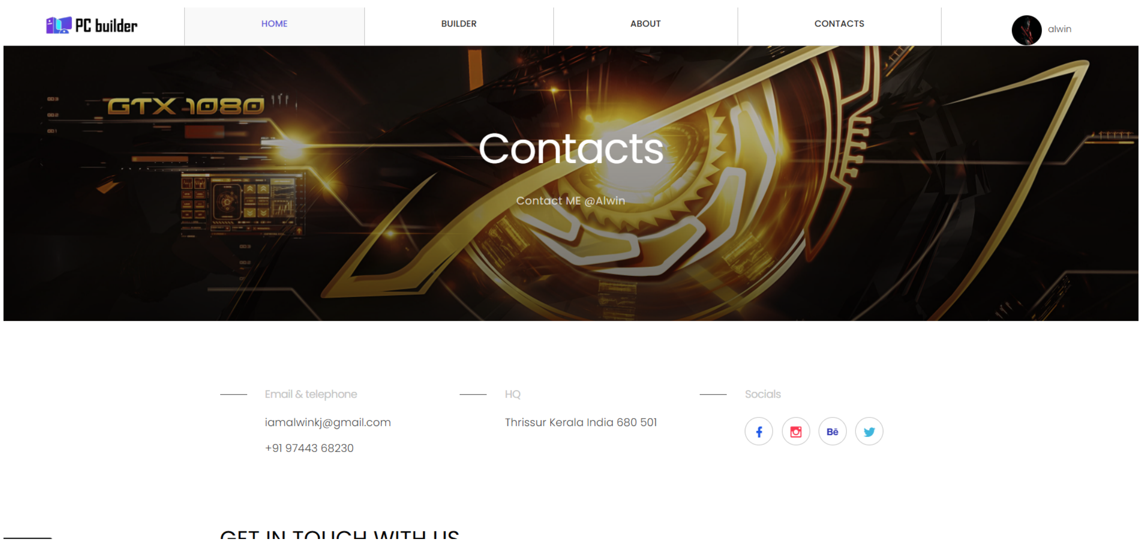
Figure D.5: Profile page



About Us

Welcome to PCPartsPicker, a labor of love and dedication inspired by the desire to empower fellow Indians in the exciting world of PC building. We are more than just a website; we are a group of people that thrives on a shared passion for technology and innovation.

Figure D.6: About page



Email & telephone

iamalwinkj@gmail.com

+91 97443 68230

HQ

Thrissur Kerala India 680 501

Socials

GET IN TOUCH WITH US

Figure D.7: Contact page

Bibliography

[1] *First Lessons in LaTeX*, by Ms. Siji.K.B , Vidya Academy of Science and Technology, Thrissur - 680501, 2024.

[2] *Django Documentation*, Django Software Foundation, Available at: `https://docs.djangoproject.com/`, Accessed 2024.

[3] *Visual Studio Code Documentation*, Django Tutorial in Visual Studio Code, Available at: `https://code.visualstudio.com/docs/python/tutorial-django/`, Accessed 2024.

[4] *Django Tutorial for Beginners (Youtube)*, Python Django Tutorial for Beginners, by Programming with Mosh ,@programmingwithmosh Available at: `https://youtu.be/rHux0gMZ3Eg`, Accessed 2024.

Department of Computer Applications

Vidya Academy of Science & Technology

Thalakkottukara, Thrissur - 680 501

(http://www.vidyaacademy.ac.in)