Nexus PC

By:

Kunal Yadav

Devesh Kumar

I K Kevin Samuels

Vikash Kumar

Submission Date:

30 April 2025

K R Mangalam University

Table Of Contents:

4. Technologies

5. Design & Architecture

6. Implementation Steps

7. Features & Functionalities

8. Testing & Validation

9. Challenges & Solutions

10. Future Enhancements

11. Conclusion

12. References

13.Appendices

Introduction

1.1 Purpose

The platform aims to demystify PC building by combining component education with a compatibility-driven assembly tool, catering to both novices and enthusiasts.

1.2 Problem Statement

Despite growing interest in custom PCs, users face fragmented information, compatibility risks, and decision fatigue—highlighting the need for an intelligent, all-in-one PC parts builder.

1.3 Target Audience

The platform serves PC builders of all skill levels—from beginners needing guided assistance to experts seeking advanced customization—while catering to gamers, professionals, and budget-focused users.

Objectives & Scope

2.1 Objectives

To democratize PC building by combining education, compatibility checks, and personalized recommendations in one platform

2.2 Scope

Covers virtual build assembly, real-time part validation, and purchasing support, excluding physical services or legacy hardware.

System Requirements

3.1 Functional Requirements

Core features like build validation, real-time pricing, and user accounts.

3.2 Non-Functional Requirements

Performance, security, and scalability standards for the platform.

Technologies

- **Static Data Storage**

- **HTML**

- **CSS**

- **JavaScript (Node.js)**

- **React**

- **Python**

Design & Architecture

Frontend Framework: Utilizes React's component-based architecture for a modular and interactive user interface.

Data Management: PC part data and specifications are likely managed either statically within the frontend code (e.g., JSON) or dynamically fetched from external APIs.

Core Features (Frontend Implementation):

PC Part Comparison: Implemented in JavaScript/React, displaying attributes for user comparison.

- **Virtual Build Assembly:** Achieved through JavaScript logic and React state management, handling part selection and compatibility.
- **Real-time Part Validation:** JavaScript code enforces compatibility rules defined within the application.
- **Purchasing Support:** Primarily through affiliate links or generating parts lists due to the absence of a backend.
- **Recommendation System:** Guides user to what he needs.

Implementation Steps

- Project Setup and Environment Configuration
- UI/UX Design and Prototyping
- Component Development (React)
- Styling (CSS)
- Data Integration (Frontend-Based)
- Implementing Core Functionalities
- State Management (React)
- ML Implementation
- Routing
- Testing

Features & Functionalities

- ChatBot

  Voice/text commands

  Can redirect user to google if query is outside its capability

- Filter section

  Can filter on basis of price, brands and other factors

  Can search product on the webpage

- An ultimate PC building step by step guide for beginners
- Recommendation System
- A News section related to latest PC Tech

Testing & Validation

Testing Approach: Likely involved a multi-faceted approach to ensure functionality and quality.

Unit Testing:

- Focused on individual React components and JavaScript functions.
- Aimed to verify isolated behavior (e.g., data rendering, logic).
- Likely employed tools like Jest and React Testing Library (speculative).

Integration Testing:

- Tested the interaction between different parts of the application.
- Example: Ensuring seamless data flow between part selection and comparison views.

Functional Testing:

- Targeted core features like real-time part validation.

- Involved testing scenarios with compatible and incompatible parts.
- Aimed to verify the accuracy of validation rules and user feedback.

UI and Responsiveness Testing:

- Likely tested on different web browsers.
- Aimed to ensure consistent appearance and functionality.
- Likely tested on various screen sizes for responsiveness.

Validation Strategies:

- Real-time part validation was a key focus.
- Testing involved various compatibility scenarios.
- Aimed to ensure accurate identification of incompatible parts and clear user feedback.

Challenges & Solutions

Challenges	Solution
Managing a Large Dataset	Used a static Dataset
Ensuring Data Accuracy	Cross checked each data
Lack of knowledge in beginners	Made an Ultimate Beginner's step by step guide for PC Building.
Large amount of Product for user	Added search filter feature

Future Enhancements

- Replacing Static Dataset with a Dynamic Dataset.

- Adding Recommendation system using API integration

- Adding User Login

- Adding an ability to save pc builds.

- A Dark Mode.

- Adding an ability to compare parts.

Conclusion

Our project offers a functional, frontend-only platform using React for PC builders to compare parts, virtually assemble systems, and receive real-time compatibility feedback. Leveraging client-side technologies for a dynamic UI, it addresses data management and validation complexities through lazy loading and modular design. While acknowledging limitations in data persistence and advanced features inherent to its architecture, the project establishes a solid foundation, with future potential in backend integration for enhanced functionality.

References

Core Technologies Documentation:

- **React Documentation:** https://react.dev/
- **CSS Specifications:** You can refer to the main entry point for CSS specifications at the World Wide Web Consortium (W3C): https://www.w3.org/Style/CSS/specs.en.html. For general understanding, MDN Web Docs is also an excellent resource: https://developer.mozilla.org/en-US/docs/Web/CSS
- **JavaScript (ECMAScript) Documentation:** For the official language specification, you can refer to the ECMAScript Language Specification: https://tc39.es/ecma262/. For more developer-friendly documentation, MDN Web Docs is highly recommended: https://developer.mozilla.org/en-US/docs/Web/JavaScript
- **HTML Standard:** For more developer-friendly documentation, MDN Web Docs is highly recommended: https://developer.mozilla.org/en-US/docs/Web/HTML