

National College of Computer Studies

Tribhuvan University

Institute of Science and Technology (IoST)



Project Report On

“HAVOCAURA”

PC Parts & Laptops Ecommerce Website

Submitted To:

National College of Computer Studies

Department of Bachelor of Science in Computer Science and Information

Technology (BSc.CSIT)

*In partial fulfillment of the requirement for the degree of Bachelor of Science
in Computer Science and Information Technology (BSc.CSIT)*

Submitted By:

Mandish Nanda Vaidya (28872/078)

Utsarga Manandhar (28897/078)

Waibhawa Mishra (28898/078)

(BSc.CSIT 6th Semester)

Acknowledgement

We would like to extend our heartfelt thanks and gratitude to our supervisor, Mr. Teksan Gharti Magar, for granting us the valuable opportunity to work on **“HAVOCAURA – PC Parts & Laptop Ecommerce Website”** project. His insightful guidance and support has greatly contributed to the success of this project, allowing us to explore new tools and technologies.

Our sincere appreciation goes to NCCS College for their constant supervision, guidance, and the necessary resources provided, which played a crucial role in the completion of this project. The support from the library and staff members of NCCS has been invaluable, and we are grateful for their cooperation and encouragement. Here, we would like to show an attitude of reverence to all of them who have contributed directly or indirectly to making the study a reality.

Finally, we shall be grateful to those people who read this project and who shall benefit from this project at present and in future.

Yours sincerely,

Mandish Nanda Vaidya

Utsarga Manandhar

Waibhawa Mishra

Abstract

This document outlines the core objectives for the development of an e-commerce platform specializing in PC parts and laptops. The primary goal is to establish a reliable online source for purchasing these products, catering to both novice and experienced users. To achieve this, the platform will prioritize a user-friendly and intuitive shopping experience, incorporating a seamless "build a PC" tool to simplify custom PC creation. A diverse selection of brands and price points will be offered, with continuous expansion of the product range to include new and in-demand items. Essential e-commerce functionalities, such as user registration/login, an "add to cart" feature, detailed invoice generation, and a secure online payment environment, will be implemented to facilitate a smooth and efficient purchasing process. The platform aims to provide a comprehensive and accessible solution for individuals seeking to purchase PC components and laptops, or to build custom computer systems.

Table of contents

Contents

Acknowledgement	i
Abstract.....	ii
Chapter 1: Introduction	1
1.1 Introduction	1
1.2 Problem Statement	1
1.3 Objectives.....	2
1.4 Scope and Limitation	2
1.5 Development Methodology	3
1.6 Report Organization	4
Chapter 2: Background Study and Literature Review	6
2.1 Background Study	6
2.2 Literature Review	6
General E-commerce Platforms:	7
Specialized PC Hardware Retailers:	7
Custom PC Building Tools:	7
Chapter 3: System Analysis	8
3.1. System Analysis	8
3.1.1. Requirement Analysis	8
3.1.2. Feasibility Analysis	10
3.1.3. Analysis	12
Chapter 4: System Design	15
4.1. Design.....	15
4.1.1 Database Design	15

4.1.2 Forms and Report Design.....	25
4.1.3 Interface and Dialogue Design	26
Chapter 5: Implementation and Testing	27
5.1. Implementation.....	27
5.1.1 Tools and Software used	27
5.1.2 Implementation Details of Modules	27
5.1.3 Data Extraction and Data Scrapping	32
5.2. Testing.....	36
5.2.1 Test Cases for Unit Testing	36
5.2.2 Test Cases for System Testing	39
5.3. Result Analysis.....	41
Chapter 6: Conclusion and Future Recommendations	42
6.1 Conclusion.....	42
6.2 Future Recommendations.....	42
References	44
Appendices	45

List of Abbreviations

PC – Personal Computer
JS – JavaScript
IDE – Integrated Development Environment
DB - Database
FDD - Feature Driven Development
MERN – MongoDB Express-js React Node-js
API – Application Programming Interface
DS – Data Scrapping
JSON – JavaScript Object Notation

List of Figures

Figure 1: Feature Driven Development.....	4
Figure 2: Use Case Diagram of HAVOCAURA.....	9
Figure 3: Gantt Chart of HAVOCAURA.....	12
Figure 4: ER Diagram of HAVOCAURA.....	13
Figure 5: DFD of HAVOCAURA.....	14
Figure 6: Code Snippets For User Database.....	17
Figure 7: User Database	17
Figure 8: Code Snippet For Laptop Database	18
Figure 9: Laptop Database.....	19
Figure 10: Code Snippet of Part Database	20
Figure 11: Part Database	21
Figure 12: Code Snippet For Order Database	24
Figure 13: Order Database.....	24
Figure 14: Registration Page	25
Figure 15: Login page	25
Figure 16: Login Successful Popup.....	26
Figure 17: Added To Cart PopUp	26
Figure 18: Code Snippets for Python Data Scraper.....	34
Figure 19: Snippets of JSON file.....	35
Figure 20: Landing Page Without Login.....	45
Figure 21: Registration Page	45
Figure 22: Login Page	45
Figure 23: Landing Page With Login.....	46
Figure 24: Laptop Store.....	46
Figure 25: PC Parts Store	46
Figure 26: Laptop Display.....	47
Figure 27: PC Parts Display	47
Figure 28: Build A PC.....	47
Figure 29: Cart Display	48
Figure 30: Khalti Payment Gateway	48
Figure 31: Order Page	48

List of Tables

Table 1: Unit testing 1 36

Table 2: Unit testing 2 37

Table 3: Unit testing 3 38

Table 4: System testing 1 39

Table 5: System testing 2 40

Chapter 1: Introduction

1.1 Introduction

The world of personal computing thrives on constant innovation and customization. From gaming enthusiasts seeking peak performance to professionals requiring reliable workstations, the demand for high-quality computer parts and laptops is ever-present. This platform directly addresses this need by providing a comprehensive online marketplace dedicated to these essential tools. We are building an e-commerce experience centered on empowering users to find, select, and purchase the exact components and systems they require. Our focus is on offering an extensive catalog of PC parts, ranging from the latest CPUs and GPUs to specialized peripherals, alongside a diverse selection of laptops designed for various needs and budgets. Recognizing the growing popularity of custom-built PCs, we will also feature an intuitive "build a PC" tool, enabling users to effortlessly design and assemble their ideal systems. By prioritizing a vast product range, user-friendly navigation, and a streamlined purchasing process, we aim to become the premier destination for individuals seeking to enhance their computing experience.

1.2 Problem Statement

The "**HAVOCAURA – PC Parts & Laptop Ecommerce Website**" project addresses the existing challenges within the online market for purchasing PC components and laptops. Currently, consumers face difficulties in navigating a fragmented market, often encountering compatibility issues and lacking intuitive tools for custom PC building. The search for desired components and competitive pricing necessitates visits to multiple online retailers, resulting in a time-consuming and frustrating experience. Furthermore, many existing platforms fail to provide a seamless and user-friendly shopping environment, hindering customer satisfaction and potentially leading to lost sales. Therefore, the "**HAVOCAURA**" project aims to create a centralized, intuitive, and comprehensive platform that simplifies the acquisition of PC parts and laptops, empowering users to effortlessly design and build custom PCs while providing a superior online shopping experience.

1.3 Objectives

The objectives of “**HAVOCAURA – PC Parts & Laptop Ecommerce Website**” project are as follows:

- Become a source for purchasing PC parts and laptops.
- Provide a user-friendly and intuitive online shopping experience.
- Offer a seamless "build a PC" tool that simplifies custom PC creation.
- Expand product range by continuously adding new and in-demand PC parts and laptop models.
- Offer a diverse selection of brands and price points.
- Allow users to register and log in to the website to purchase items and view their order history.
- Create an “add to cart” feature to add items to a cart while browsing through the purchasable items.
- Provide an invoice after confirming purchase along with user information, purchased items, costs and expected delivery date.
- Offer an online payment environment for item purchases.

1.4 Scope and Limitation

Scope:

The "**HAVOCAURA – PC Parts & Laptop Ecommerce Website**" project will encompass the development of a fully functional online platform for the purchase of computer parts and laptops. The scope includes:

- **Product Catalog:** A comprehensive and searchable database of PC components (CPUs, GPUs, motherboards, RAM, storage, etc.) and laptops, with detailed specifications and images.
- **"Build a PC" Tool:** An intuitive interface enabling users to select compatible components and design custom PC builds.
- **User Accounts:** Functionality for user registration, login and viewing order history.
- **Shopping Cart and Checkout:** A secure and user-friendly shopping cart system with invoice generation, payment option and order confirmation.
- **Payment Gateway Integration:** Integration with secure payment gateways for online transactions.

- **Product Search and Filtering:** Advanced search and filtering options to help users find specific products.

Limitations:

The "**HAVOCAURA – PC Parts & Laptop Ecommerce Website**" project will have the following limitations:

- **Order Management:** Tools for managing orders, including order tracking and shipping information.
- **Customer Support:** Implementation of a contact form and FAQ section.
- **Responsive Design:** Ensuring the website is accessible and functional across various devices (desktops, tablets, and smartphones).
- **Initial Product Inventory:** The initial product inventory will be limited, and the project will focus on establishing core functionalities before expanding the product range significantly.
- **Advanced Product Comparison:** A fully featured advanced product comparison tool may be implemented at a later stage.

1.5 Development Methodology

This project is on a small scale and has well-defined requirements with a focus on flexibility and adaptability to changing market trends. Under such development circumstances, the Agile development model, specifically the Feature Driven Development (FDD), is used.

FDD stands for **Feature-Driven Development**. It is an agile iterative and incremental model that focuses on progressing the features of the developing software. The main motive of feature-driven development is to provide timely updated and working software to the client. In FDD, reporting and progress tracking is necessary at all levels. [1]

Feature-Driven Development (FDD) is an agile methodology well-suited for the **HAVOCAURA** project. FDD emphasizes the rapid delivery of small, functional features. For the "**HAVOCAURA**" project, Feature-Driven Development (FDD) will be used to prioritize delivering user-focused features. Initially, a high-level system model is created, followed by a detailed feature list. Features are then prioritized and planned for development. Each feature undergoes detailed design, implementation using the MERN stack, and rigorous testing. Code reviews and user acceptance testing ensure quality. This

iterative approach, repeating design, build, and inspection, allows for continuous progress. FDD's emphasis on user value and quality, coupled with MERN's flexibility, makes it ideal for developing **HAVOCAURA**.



Figure 1: Feature Driven Development

1.6 Report Organization

Chapter 1: Introduction

Chapter 1 introduces the "HAVOCAURA" project, outlining its purpose, addressing the online PC parts/laptop market's problems, and defining the project's objectives. It details the project's scope, limitations, and the FDD development methodology used. Finally, it provides a roadmap of the report's structure.

Chapter 2: Background Study and Literature Review

Chapter 2 establishes the project's theoretical foundation. It provides background on e-commerce, PC technology, and web development, and reviews similar projects and relevant research.

Chapter 3: System Analysis

Chapter 3 analyzes the project's requirements and feasibility. It covers functional and non-functional requirements, feasibility analysis (technical, operational, economic, schedule), and object-oriented analysis using diagrams.

Chapter 4: System Design

Chapter 4 details the design phase, refining object-oriented diagrams and creating component and deployment diagrams. It also describes any specific algorithms used.

Chapter 5: Implementation and Testing

Chapter 5 documents the development and testing process, listing tools used, describing module implementation, and providing unit and system test cases. It concludes with an analysis of test results.

Chapter 6: Conclusion and Future Recommendations

Chapter 6 summarizes the project's achievements and provides recommendations for future enhancements.

Chapter 2: Background Study and Literature Review

2.1 Background Study

The "HAVOCAURA" project, an e-commerce platform, is grounded in several core theoretical and conceptual areas. Firstly, fundamental e-commerce principles are essential, encompassing online transaction protocols, effective user interface design, and secure payment processing. Understanding customer behavior in the online marketplace, including browsing habits, cart abandonment, and conversion rates, is crucial. This necessitates the application of digital marketing strategies and principles of user experience (UX) design to create an intuitive and engaging platform.

Secondly, the development relies heavily on web development technologies, specifically the MERN stack (MongoDB, Express.js, React.js, Node.js). This requires a deep understanding of database management using MongoDB, backend development with Node.js and Express.js for building robust APIs, and frontend development using React.js for creating dynamic and interactive user interfaces. Proficiency in RESTful API design, state management in React, and asynchronous JavaScript programming is critical for building a responsive and scalable application.

Thirdly, the project incorporates principles of software engineering, including version control using Git, testing methodologies like unit and integration testing, and effective deployment strategies. Understanding agile development practices, particularly Feature-Driven Development (FDD), is crucial for efficient project management and delivery. Security considerations, such as data encryption, user authentication, and protection against common web vulnerabilities, are also paramount to ensure a safe and reliable platform for users.

2.2 Literature Review

This section examines the landscape of existing e-commerce websites, focusing on their functionalities and the underlying theories and research that inform their design and operation, particularly within the context of PC parts and laptop sales.

General E-commerce Platforms:

Platforms like Amazon and eBay demonstrate the effectiveness of large-scale, multi-vendor marketplaces. Their functionalities, such as personalized recommendations (based on collaborative filtering and content-based filtering, as studied by researchers in information retrieval and machine learning), extensive product filtering, and robust search engines (informed by theories of information architecture and search engine optimization), serve as benchmarks for "**HAVOCAURA**." Research in human-computer interaction (HCI) highlights the importance of intuitive navigation and clear visual hierarchy, principles these platforms often employ. Studies by Nielsen Norman Group on e-commerce usability reinforce the significance of streamlined checkout processes and transparent pricing. [2] [3] [4]

Specialized PC Hardware Retailers:

Websites like Newegg and Micro Center exemplify specialized e-commerce for PC hardware. They prioritize detailed product specifications, user reviews, and component compatibility information. Their success is rooted in providing a knowledge-rich environment for tech-savvy consumers. Theories of information design and knowledge representation are crucial here, ensuring data accuracy and accessibility. Functionalities like comparison tools and "build a PC" wizards, often seen on these sites, are informed by research in constraint satisfaction and knowledge-based systems, aiming to simplify complex purchasing decisions. [5] [6]

Custom PC Building Tools:

Platforms like PCPartPicker provide specialized tools for building custom PCs. Their core functionality lies in component compatibility checking, real-time price aggregation, and user-generated build lists. Research in database design and algorithm efficiency is essential for these tools. Studies on graph theory and constraint programming have been applied to develop efficient algorithms for component compatibility checks. Theories of user interface design focusing on task-oriented interfaces are key to creating a streamlined experience for users. [7]

By analyzing these diverse platforms and research that informs their design, "**HAVOCAURA**" can leverage best practices and address potential challenges, creating a comprehensive and user-friendly e-commerce experience for PC parts and laptops. This analysis will ensure **HAVOCAURA** is built upon proven methods and current research.

Chapter 3: System Analysis

3.1. System Analysis

This chapter details the analysis of the "**HAVOCAURA**" system, focusing on understanding the requirements, assessing feasibility, and modeling the system's structure and behavior.

3.1.1. Requirement Analysis

Requirements analysis, also known as requirements engineering, is the process of determining the needs and expectations of stakeholders for a new or modified product, ensuring the final product meets those needs.

3.1.1.1 Functional Requirements

The "**HAVOCAURA**" system must fulfill the following functional requirements to provide comprehensive and effective e-commerce experience:

- **Product Management:**
 - Users must be able to browse products by category, brand, and model.
 - Users must be able to search for products using keywords or product names.
 - The system must display detailed product information, including specifications, descriptions and images.
- **User Accounts:**
 - Users must be able to create and manage user accounts.
 - Users must be able to save invoices.
 - Users must be able to view order history and track order status.
- **Shopping Cart and Orders:**
 - Users must be able to add products to the shopping cart.
 - Users must be able to view and modify the contents of the shopping cart.
 - Users must be able to calculate the total cost of items in the shopping cart.
 - Users must be able to place orders securely.

- The system must generate order confirmations and invoices.
 - Users must be able to track the status of their orders.
 - The system must integrate with secure payment gateways.
- **"Build a PC" Tool:**
 - Users must be able to select PC components to build custom PCs.
 - The tool must calculate the total cost of the custom PC build.
- **Administration:**
 - Administrators must be able to manage orders.
 - Administrators must be able to manage inventory.
 - This feature is not implemented but will be implemented in the future.

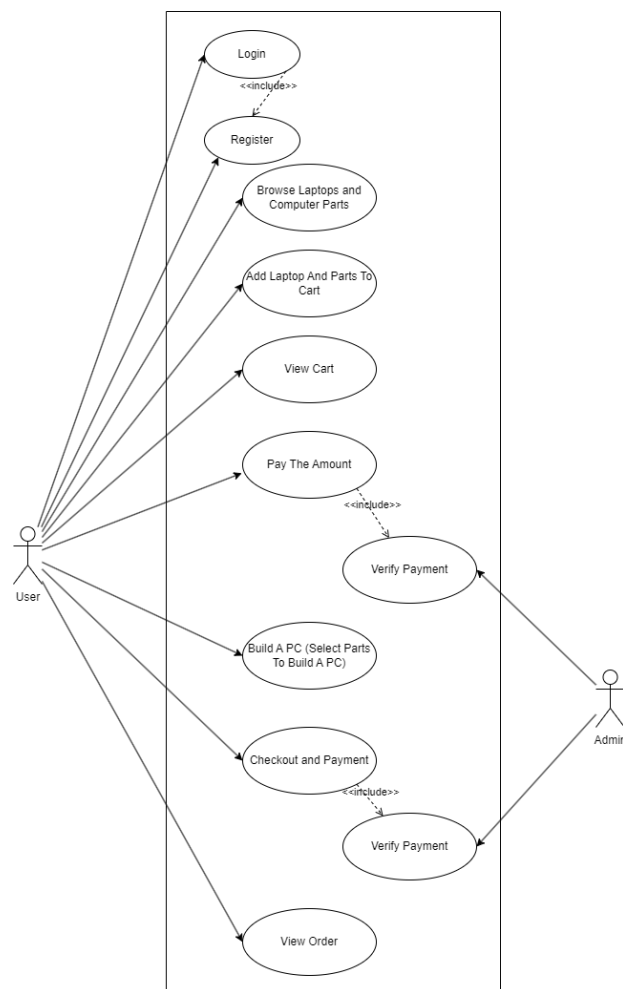


Figure 2: Use Case Diagram of HAVOCAURA

The above use case diagram has two entities user and admin. The user first logs in which requires registration. Then the user browses the laptops and parts. Then the user add the items to cart then he checks out. Then payment is done which is confirmed by the admin. Similarly, user also can build a pc after which user will checkout and pay which will be confirmed by the admin.

3.1.1.2 Non-Functional Requirements

The "**HAVOCAURA**" system must adhere to the following non-functional requirements to ensure high-quality user experience and maintain system integrity:

- **Accuracy:** The system must ensure that all data, including product specifications, pricing, inventory levels, and order details, is consistently correct and up to date, minimizing errors that could lead to customer dissatisfaction.
- **Processing Speed:** The system must respond rapidly to user actions and transactions, providing seamless and efficient experience, minimizing delays in loading pages, search results, and checkout processes.
- **Security:** The system must rigorously protect user and payment data from unauthorized access, employing robust encryption and authentication mechanisms to safeguard sensitive information and maintain user trust.
- **Usability:** The website must be designed to be easy and intuitive for all users, regardless of their technical expertise, with clear navigation, logical layouts, and accessible interfaces that enhance the overall user experience.
- **Maintainability:** The system's design must be modular and well-documented, allowing for easy updates, modifications, and bug fixes, ensuring the system can adapt to changing requirements and remain sustainable over time.

3.1.2. Feasibility Analysis

A feasibility analysis, also known as a feasibility study, is a comprehensive assessment to determine the likelihood of a project's success by evaluating critical factors like economic, marketing, technical, financial, and management aspects.

3.1.2.1 Technical feasibility

The "**HAVOCAURA**" project demonstrates strong technical feasibility, primarily due to the selection of the MERN stack. This technology suites, comprising MongoDB, Express.js, React.js, and Node.js, is a well-established and widely utilized framework for

modern web application development. Each component of the MERN stack is mature and well-documented, providing a stable foundation for the project. Its inherent scalability ensures the system can accommodate increasing traffic and data loads, crucial for an e-commerce platform. Furthermore, the readily available pool of MERN stack developers and extensive online resources offer ample support for troubleshooting and development. MongoDB's flexible NoSQL database allows for efficient management of diverse product data, while React.js facilitates the creation of a dynamic and responsive user interface, including the "build a PC" tool. Secure payment gateway integration is also readily achievable within this environment, solidifying the project's technical viability.

3.1.2.2 Operational feasibility

The operational feasibility of "**HAVOCAURA**" is strongly supported by its focus on enhancing user experience and streamlining the purchase of PC parts and laptops. Central to this is the development of an intuitive user interface, designed to be easily navigable by customers of all technical proficiencies. The platform will simplify the complexities often associated with online shopping by consolidating product information, compatibility checks, and purchasing functionalities into a single, cohesive system, thereby reducing the need for users to navigate multiple websites. This streamlined approach not only saves time and effort but also contributes to an enhanced user experience, from product discovery to order fulfillment. Features like the "Build a PC" tool, detailed product information, and responsive customer support will further ensure a positive and efficient interaction. By simplifying the component building process and integrating seamlessly into existing user workflows, "**HAVOCAURA**" aims to provide a superior and accessible online shopping experience.

3.1.2.3 Economic feasibility

The economic viability of "**HAVOCAURA**" is centered on its ability to generate revenue that significantly outweighs both development and operational expenses, ensuring a strong return on investment. This will be realized through several key strategies. Firstly, the project will target the expanding market for PC parts and laptops, catering to diverse consumer needs, and conduct a thorough market analysis to accurately forecast revenue. Secondly, cost-effective development using the MERN stack, coupled with streamlined operational processes and inventory management, will minimize expenses. Thirdly, a

competitive pricing strategy will be employed, balancing customer attraction with healthy profit margins. Fourthly, a strategic marketing plan will utilize digital channels and SEO to drive traffic and reduce customer acquisition costs. Finally, a comprehensive cost-benefit analysis will evaluate the project's financial viability and long-term sustainability.

3.1.2.4 Schedule feasibility

The schedule feasibility of "HAVOCAURA" is contingent upon meticulous planning and efficient resource allocation. A detailed project timeline, outlining key milestones and deadlines, will be developed to ensure timely completion. Adequate resources, including skilled developers, experienced designers, and effective project managers, will be allocated to the project. Contingency plans will be established to address potential delays and mitigate risks. The adoption of agile development methodologies, such as Feature-Driven Development (FDD), will provide flexibility and adaptability to changes in the project schedule. With careful planning, resource allocation, and contingency measures in place, the schedule feasibility of the "HAVOCAURA" project is considered achievable.

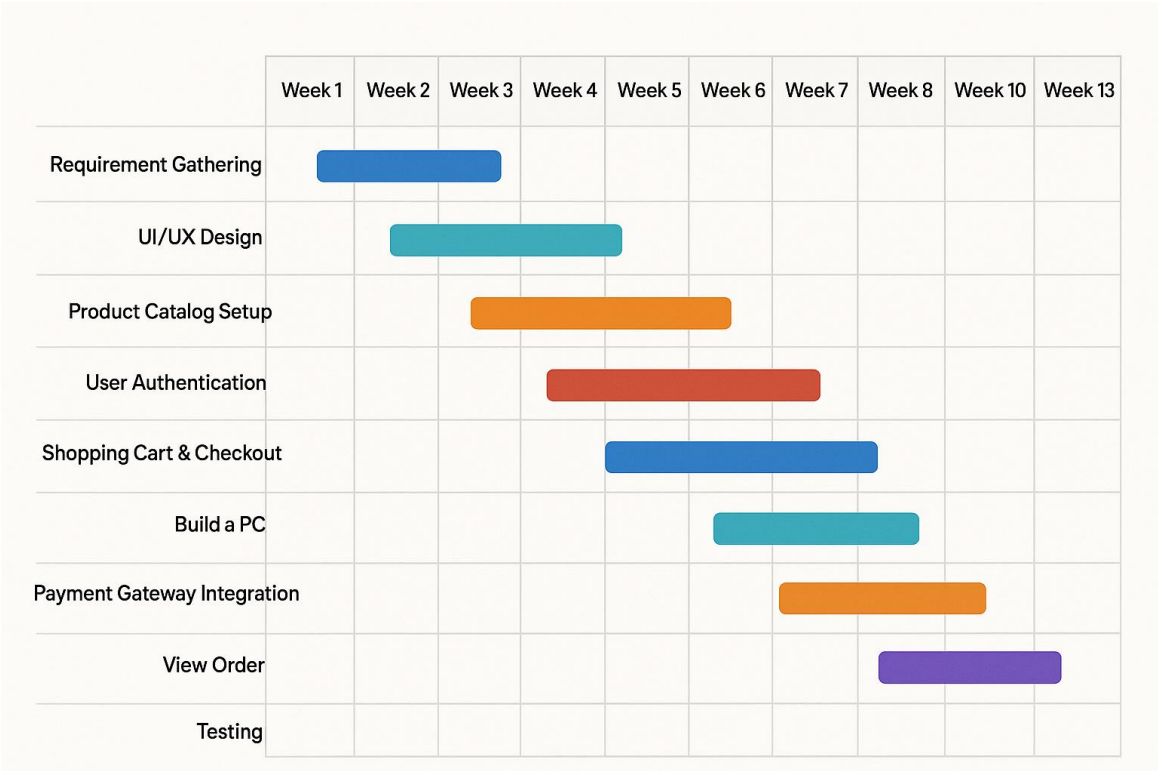


Figure 3: Gantt Chart of HAVOCAURA

3.1.3. Analysis

A structured analysis for "HAVOCAURA" would prioritize processes and data flow over objects. Functional requirements are shown through Data Flow Diagrams, while Entity-

Relationship diagrams model data relationships.

3.1.3.1 Data modelling

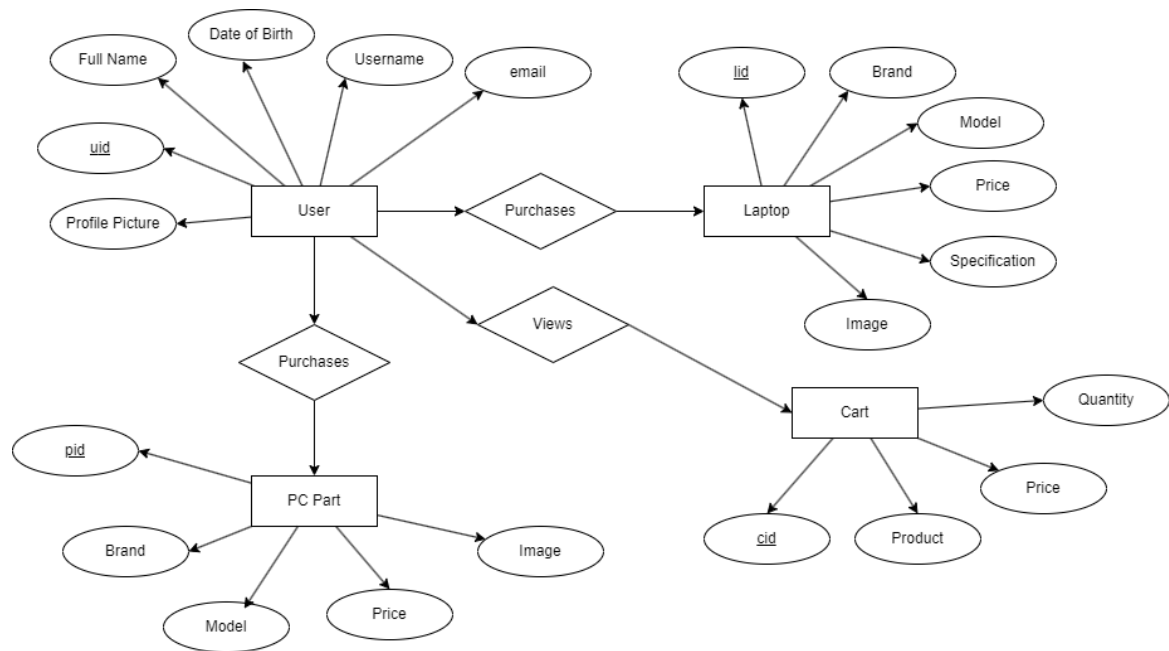


Figure 4: ER Diagram of HAVOCAURA

The above ER diagram has 4 entities: User, Laptop, PC Part, and Cart. The user entity has attributes uid, Full Name, Username, Date of Birth, Email, and Profile Picture. Similarly, laptop entity has attributes lid, brand, model, price, specification, and image. PC Part entity has attributes pid, brand, model, price, and image. Finally, cart has attributes cid, product, price, and quantity.

3.1.3.2 Process modelling

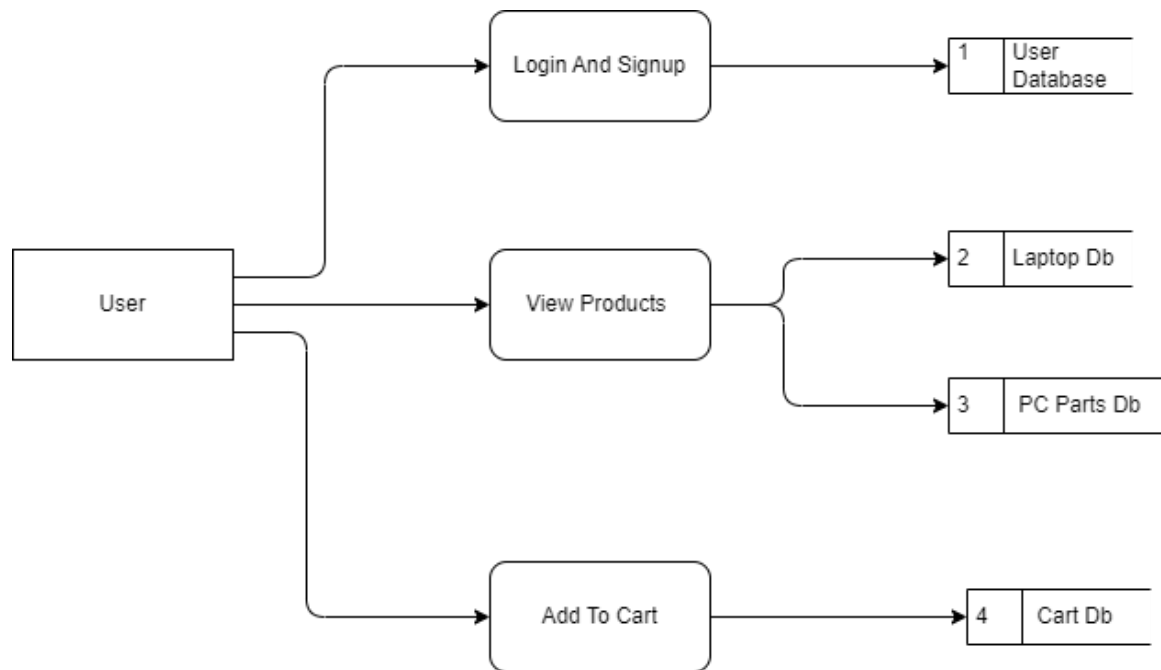


Figure 5: DFD of HAVOCAURA

The above DFD represents the processes a user can do. The user can login and signup which uses the User Database. Then they can view products which uses the Laptop Database and Pc Parts Database. Finally, the user can add to cart which uses the cart database.

Chapter 4: System Design

4.1. Design

System design is the process of creating a blueprint for a system, defining its architecture, components, modules, interfaces, and data to meet specific requirements and goals.

4.1.1 Database Design

Talking about the database design, HAVOCAURA has 4 databases: User, Laptop, Parts, and Order. The user database consists of all the data of the user and it also consists of the cart information of the user. Similarly, the laptop database consists of all the laptops available on HAVOCAURA. Moreover, the parts database consists of all the computer parts that are available in HAVOCAURA. Laptop and Parts database consists of model, name, price, etc. of products. Finally, order database is the database consist of the checkout completed orders of the user.

The code snippets of database and the database snippets are listed below:

```

password: {
  type: String,
  required: true,
},
profilePicture: {
  type: String,
  required: true,
},
cart: {
  laptops: [
    {
      part: { type: mongoose.Schema.Types.ObjectId, ref: "Laptop" },
      price: Number,
      quantity: { type: Number, required: true, min: 1 },
    },
  ],
  parts: {
    CPU: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
    GPU: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
    Motherboard: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
    RAM: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
    Storage: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
  },
  price: Number,
  quantity: { type: Number, required: true, min: 1 },
},
Cooling_System: [
  {
    part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
    price: Number,
    quantity: { type: Number, required: true, min: 1 },
  },
],
Case: [
  {
    part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
    price: Number,
    quantity: { type: Number, required: true, min: 1 },
  },
],
Peripherals: [
  {
    part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
    price: Number,
    quantity: { type: Number, required: true, min: 1 },
  },
],

```



```

        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
    HDD: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
    SSD: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number, required: true, min: 1 },
      },
    ],
  ],
},
{ timestamps: true }
);

module.exports = mongoose.model("User", userSchema);

```

Figure 6: Code Snippets For User Database

▶

```

_id: ObjectId('67d78ddcc31aba94097f7d21')
fullName: "Utsarga Manandhar"
userName: "hlubabz"
gender: "Male"
dob: 2004-11-29T00:00:00.000+00:00
email: "utsargam44@gmail.com"
password: "$2b$10$ML778whw9q/o0a7SgB1.NOg0XKTz4X9l/jdtGg6IW0L7kpv.Lhd4W"
profilePicture: "data:image/jpeg;base64,/9j/4bKVRXhpZGAASUkqAAgAAAAAABBAABAAAoA8AAAE="
createdAt: 2025-03-17T02:50:04.938+00:00
updatedAt: 2025-04-20T07:07:53.778+00:00
__v: 29
cart: Object

```

✎ 🔄 📄 🗑️

```

_id: ObjectId('67d793bbc31aba94097f7d34')
fullName: "Mandish Nanda Vaidya"
userName: "mandish_nv"
gender: "Male"
dob: 2004-11-29T00:00:00.000+00:00
email: "mandishvaidya12345@gmail.com"
password: "$2b$10$Zt0LDa2NWL/sI195p6B0ReYL5EsVxvPudrxcfcz0t34fv3s1MS037i"
profilePicture: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAQ/4gIoSUNDX1BSt0Z3TEU="
createdAt: 2025-03-17T03:15:07.565+00:00
updatedAt: 2025-03-17T03:15:07.565+00:00
__v: 0

```

Figure 7: User Database

```

backend > models > laptop.js > (X) laptopSchema > graphicsCard
1  const mongoose = require("mongoose");
2
3  const laptopSchema = new mongoose.Schema({
4    brand: {
5      type: String,
6      required: true,
7      trim: true,
8    },
9    model: {
10     type: String,
11     required: true,
12     trim: true,
13   },
14   processor: {
15     type: String,
16     required: true,
17   },
18   ram: {
19     type: String,
20     required: true,
21   },
22   storage: {
23     type: String,
24     required: true,
25   },
26   graphicsCard: {
27     type: String,
28     default: "Integrated",
29   },
30   screenSize: {
31     type: String, // in inches
32     required: true,
33   },
34   operatingSystem: {
35     type: String,
36     default: "Windows",
37   },
38   price: {
39     type: Number,
40     required: true,
41     min: 0,
42   },
43   description: {
44     type: String,
45     trim: true,
46   },
47   image: {
48     type: String,
49     default: "",
50   },
51 }, { timestamps: true });
52
53 const Laptop = mongoose.model("Laptop", laptopSchema);
54
55 module.exports = Laptop;

```

Figure 8: Code Snippet For Laptop Database

```
_id: ObjectId('67d8137f6d85fe5add449cde')
brand: "Lenovo"
model: "Lenovo IdeaPad 5 2-in-1 14.0" Touchscreen Laptop Core 7 150U Intel Gra_"
price: 79198
description: "Lenovo IdeaPad 5 2-in-1 14.0" Touchscreen Laptop Core 7 150U Intel Gra_"
Processor: "Intel Core 7 150U"
RAM: "16GB"
Storage: "1 TB PCIe SSD"
Graphics Card: "Intel Graphics"
Screen Size: "14.0""
Operating System: "Windows 11 Home"
image: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/2wBDAAgGBgcGBQgHBwc..."
```

```
_id: ObjectId('67d8137f6d85fe5add449cdf')
brand: "Lenovo"
model: "Lenovo IdeaPad Slim 5 16.0" Laptop Core 7 150U Intel Graphics 16GB RAM_"
price: 79198
description: "Lenovo IdeaPad Slim 5 16.0" Laptop Core 7 150U Intel Graphics 16GB RAM_"
Processor: "Intel Core 7 150U"
RAM: "16GB"
Storage: "1 TB PCIe SSD"
Graphics Card: "Intel Graphics"
Screen Size: "16.0""
Operating System: "Windows 11 Home"
image: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/2wBDAAgGBgcGBQgHBwc..."
```

Figure 9: Laptop Database

```

backend > models > parts.js > (x) computerPartSchema > category > enum
1  const mongoose = require("mongoose");
2
3  const computerPartSchema = new mongoose.Schema(
4  {
5    model: {
6      type: String,
7      required: true,
8      trim: true,
9    },
10   category: {
11     type: String,
12     required: true,
13     enum: [
14       "CPU",
15       "GPU",
16       "Motherboard",
17       "RAM",
18       "Storage",
19       "Power Supply",
20       "Cooling System",
21       "Case",
22       "Peripherals",
23       "Other",
24       "HDD",
25       "SSD"
26     ],
27   },
28   brand: {
29     type: String,
30     required: true,
31     trim: true,
32   },
33   price: {
34     type: Number,
35     required: true,
36     min: 0,
37   },
38   specifications: {
39     type: String,
40   },
41   description: {
42     type: String,
43     trim: true,
44   },
45   image: {
46     type: String,
47     default: "",
48   }
49 },
50 { timestamps: true }
51 },
52 { timestamps: true }
53 );
54
55 const ComputerPart = mongoose.model("ComputerPart", computerPartSchema);
56
57 module.exports = ComputerPart;

```

Figure 10: Code Snippet of Part Database

```
_id: ObjectId('67d81e51c596f2ced94199c7')
brand: "AMD"
model: "AMD Ryzen 9 9950X"
price: 71146
description: "AMD Ryzen 9 9950X - Ryzen 9 9800 Series Granite Ridge (Zen 5) 16-Core ..."
image: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/4gHYSUNDX1BST0ZJTEU..."
category: "CPU"
```

```
_id: ObjectId('67d81e51c596f2ced94199c8')
brand: "AMD"
model: "AMD Ryzen 7 7700X"
price: 36001
description: "AMD Ryzen 7 7700X - Zen 4 8-Core 4.5 GHz - Socket AM5 - 105W Desktop P..."
image: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/4gHYSUNDX1BST0ZJTEU..."
category: "CPU"
```

```
_id: ObjectId('67d81e51c596f2ced94199c9')
brand: "AMD"
model: "AMD Ryzen 9 7900X"
price: 47122
description: "AMD Ryzen 9 7900X - Zen 4 12-Core 4.7 GHz - Socket AM5 - 170W Desktop ..."
image: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/4gHYSUNDX1BST0ZJTEU..."
category: "CPU"
```

Figure 11: Part Database

```

backend > models > order.js > ...
1  const mongoose = require("mongoose");
2
3  const checkoutSchema = new mongoose.Schema(
4    {
5      user: {
6        type: mongoose.Schema.Types.ObjectId,
7        ref: "User",
8      },
9      laptops: [
10       {
11         part: { type: mongoose.Schema.Types.ObjectId, ref: "Laptop" },
12         price: Number,
13         quantity: { type: Number },
14       },
15     ],
16     parts: {
17       CPU: [
18         {
19           part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
20           price: Number,
21           quantity: { type: Number },
22         },
23       ],
24       GPU: [
25         {
26           part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
27           price: Number,
28           quantity: { type: Number },
29         },
30       ],
31       Motherboard: [
32         {
33           part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
34           price: Number,
35           quantity: { type: Number },
36         },
37       ],
38       RAM: [
39         {
40           part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
41           price: Number,
42           quantity: { type: Number },
43         },
44       ],
45       Storage: [
46         {
47           part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
48           price: Number,
49           quantity: { type: Number },
50         },
51       ],
52     },
53   },
54 );

```

```

    ],
    Power_Supply: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number },
      },
    ],
    "Cooling System": [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number },
      },
    ],
    Case: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number },
      },
    ],
    Peripherals: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
      },
    ],
    Other: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number },
      },
    ],
    HDD: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number },
      },
    ],
    SSD: [
      {
        part: { type: mongoose.Schema.Types.ObjectId, ref: "ComputerPart" },
        price: Number,
        quantity: { type: Number },
      },
    ],
  ],
  totalPrice: {
    type: Number,
    required: true,
    min: 0,
  },

```

```

totalPrice: {
  type: Number,
  required: true,
  min: 0,
},
shippingCost: {
  type: Number,
  default: 100,
},
tax: {
  type: Number,
  default: 0.1, // 10% tax
},
discount: {
  type: Number,
  default: 0,
},
expectedDeliveryDate: {
  type: Date,
  default: () => {
    const today = new Date();
    return new Date(today.setDate(today.getDate() + 7)); // 7-day delivery
  },
},
status: {
  type: String,
  default: "Pending"
}
},
{ timestamps: true }
);

const Checkout = mongoose.model("Checkout", checkoutSchema);

module.exports = Checkout;

```

Figure 12: Code Snippet For Order Database

```

_id: ObjectId('68049c15479104b438fc593a')
user : ObjectId('67d78ddcc31aba94097f7d21')
laptops : Array (empty)
parts : Object
  totalPrice : 269595
  shippingCost : 100
  tax : 0.1
  discount : 0
  status : "Pending"
  expectedDeliveryDate : 2025-04-27T07:02:45.744+00:00
  createdAt : 2025-04-20T07:02:45.750+00:00
  updatedAt : 2025-04-20T07:02:45.750+00:00
__v : 0

```

```

_id: ObjectId('68049c34479104b438fc59a7')
user : ObjectId('67d78ddcc31aba94097f7d21')
laptops : Array (empty)
parts : Object
  totalPrice : 207488
  shippingCost : 100
  tax : 0.1
  discount : 0
  status : "Pending"
  expectedDeliveryDate : 2025-04-27T07:03:16.774+00:00
  createdAt : 2025-04-20T07:03:16.778+00:00
  updatedAt : 2025-04-20T07:03:16.778+00:00
__v : 0

```

Figure 13: Order Database

4.1.2 Forms and Report Design

Talking about Forms and Report Design, there are two forms: one is login and another is register that helps the user to register into our website and login to their existing account.

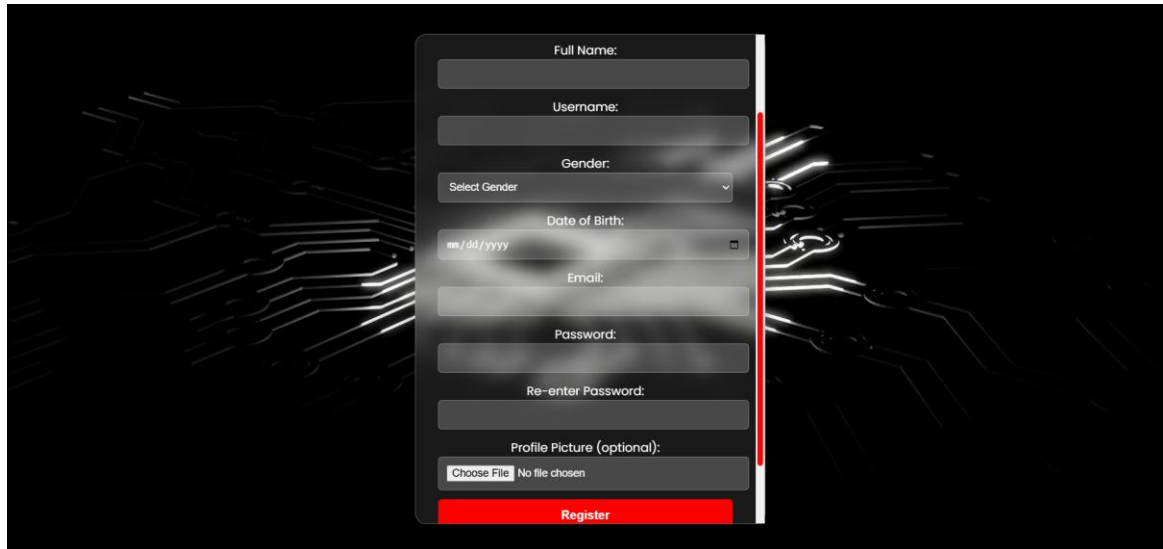
A registration form with a dark background and glowing circuit patterns. The form is centered and contains the following fields: Full Name, Username, Gender (a dropdown menu with 'Select Gender' text), Date of Birth (with a date icon and 'dd/mm/yyyy' placeholder), Email, Password, Re-enter Password, and Profile Picture (optional) with a 'Choose File' button and 'No file chosen' text. A red 'Register' button is at the bottom.

Figure 14: Registration Page

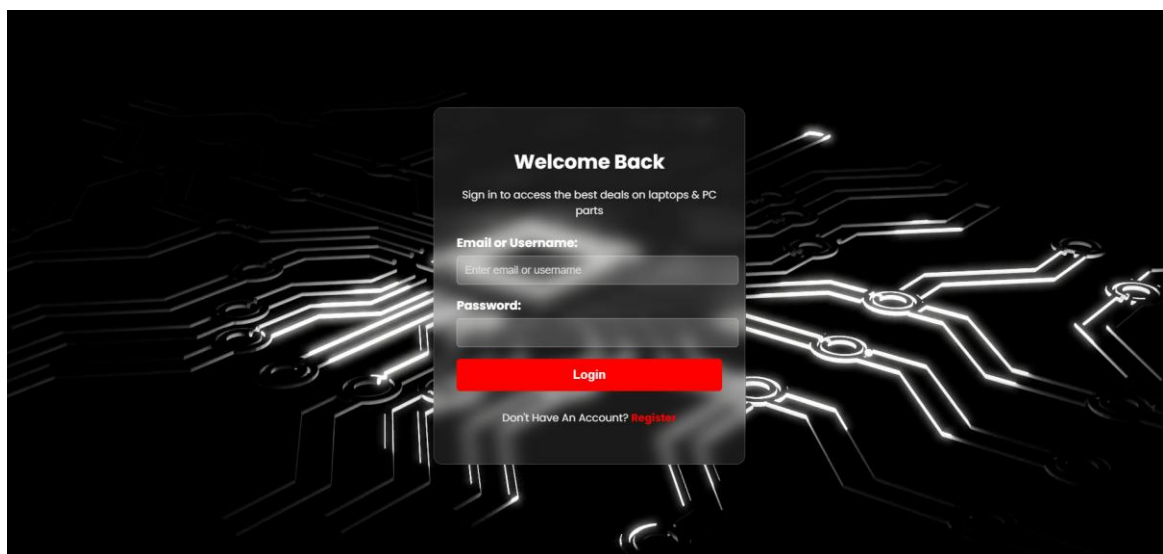
A login form with a dark background and glowing circuit patterns. The form is centered and contains the following elements: A 'Welcome Back' heading, a sub-heading 'Sign in to access the best deals on laptops & PC parts', an 'Email or Username' field with a placeholder 'Enter email or username', a 'Password' field, a red 'Login' button, and a link 'Don't Have An Account? Register' at the bottom.

Figure 15: Login page

4.1.3 Interface and Dialogue Design

Interface and Dialogue Design refer to any dialogue boxes or pop ups in the webpage. There are several pop ups in web page like the login successful popup that comes after a successful login. Another dialogue is added to cart successfully alert that comes after adding to cart.

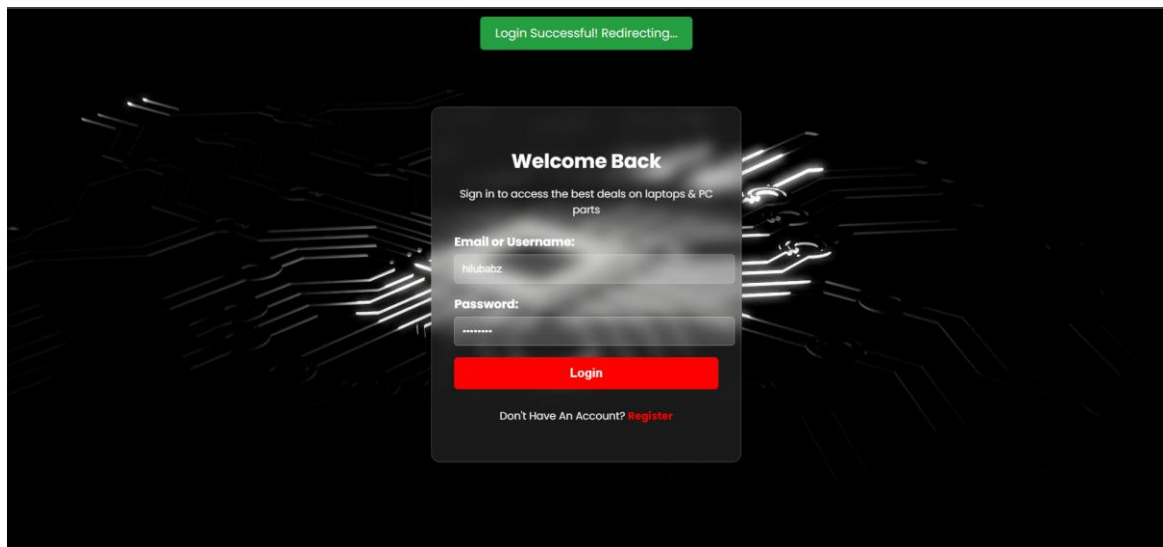


Figure 16: Login Successful Popup

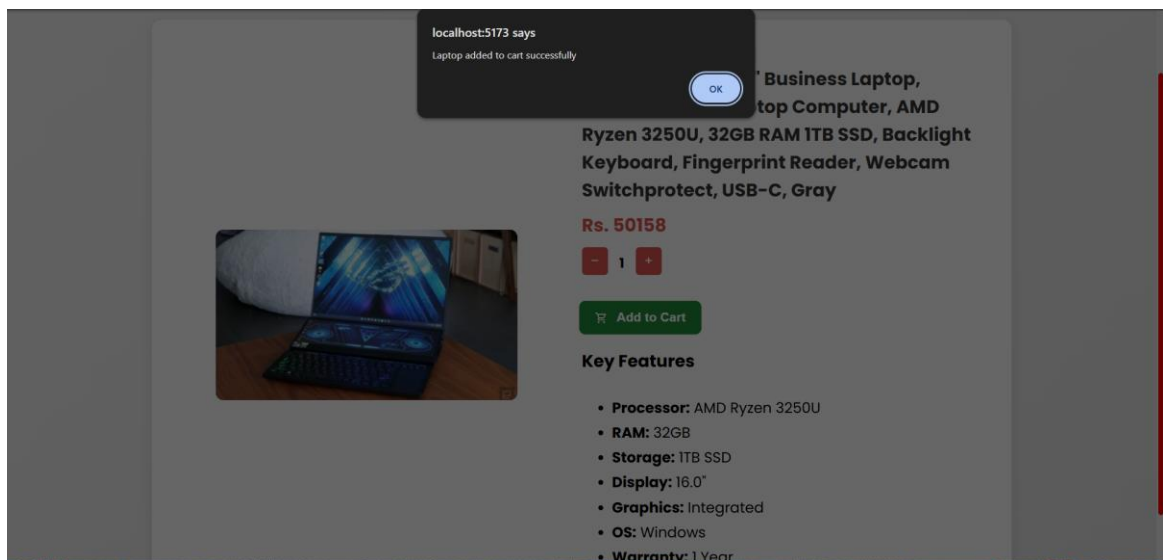


Figure 17: Added To Cart PopUp

Chapter 5: Implementation and Testing

5.1. Implementation

5.1.1 Tools and Software used

- **Platform:** Windows
- **Designing Tool:** Figma
- **Front end:** React
- **Back end:** Express js
- **Database:** MongoDB
- **IDE:** VS Code
- **API Development:** Postman
- **Hosting:** Vite, MongoDB Atlas
- **Data Collection:** Python

5.1.2 Implementation Details of Modules

1. Product Catalog Module Implementation Details

Frontend (React):

- **Data Flow:**
 - On component mount (e.g., a product listing page), a useEffect hook will dispatch an action (using Redux or Context API) to fetch products from the /api/products or /all-laptops, /all-parts backend endpoints.
 - The fetched data will be stored in the global state.
 - Components like ProductList, ProductCard, and ProductDetails will access and display this data.
 - Clicking on a product card (in ProductList) will navigate the user to the /product/:slug route using React Router, and the ProductDetails component will fetch the specific product data using the /api/retrieveById/:id backend endpoint.
- **Key Functions/Methods:**
 - fetchProducts(filters, sort, pagination): Function to make API calls to /api/products or the category-specific endpoints with appropriate query parameters.

- `displayProductCard(product)`: Renders the UI for a single product in the listing.
- `displayProductDetails(product)`: Renders the detailed information for a specific product.

Backend (Express.js):

- **Data Flow:**
 - Incoming HTTP requests to product-related API endpoints are handled by the defined routes (`/api/products`, `/all-laptops`, `/all-parts`, `/retrieveById/:id`, `/retrieveByCategory/:category`).
 - Route handlers (controllers) interact with the Mongoose models (`ComputerPart`, `Laptop`) to query the MongoDB database.
 - Data is retrieved based on the request parameters (query parameters for filtering, sorting, pagination; route parameters for specific IDs or slugs).
 - The retrieved data is formatted as JSON and sent back in the HTTP response.
- **Key Functions/Methods (within route handlers):**
 - `ComputerPart.find(query, projection, options)`: Mongoose method to retrieve computer parts based on a query, select specific fields (projection), and apply options (like limit, skip, sort).
 - `Laptop.find(query, projection, options)`: Similar to `ComputerPart.find` but for laptops.
 - `ComputerPart.findById(id)` and `Laptop.findById(id)`: Mongoose methods to retrieve a single document by its ID.

2. User Authentication and Management Module Implementation Details

Frontend (React):

- **Data Flow:**
 - The `RegistrationForm` component collects user input and sends a POST request to the `/register` backend endpoint.
 - The `LoginForm` component collects login credentials and sends a POST request to the `/login` backend endpoint. On successful login, the backend might send back user data or a token, which is then stored in the frontend state (using Context API or Redux) and potentially in local storage or cookies for persistence.
 - The `UserProfile` component, if the user is authenticated, fetches user data

from the `/findUser` backend endpoint (or a `/api/profile` endpoint if implemented) on mount. It also handles updating user profile information via a PUT request to a similar backend endpoint.

- PrivateRoute components or custom hooks check the authentication state before allowing access to protected routes.

- **Key Functions/Methods:**

- `handleRegistration(userData)`: Sends a POST request to `/register`.
- `handleLogin(credentials)`: Sends a POST request to `/login` and updates the authentication state upon success.
- `fetchUserProfile()`: Sends a POST request to `/findUser` (or GET `/api/profile`) to retrieve user data.
- `updateUserProfile(updatedData)`: Sends a PUT request to update user profile information.
- `isAuthenticated()`: Checks the authentication state (e.g., presence of a token or user object in state).

Backend (Express.js):

- **Data Flow:**

- Incoming POST requests to `/register` and `/login` are handled by their respective route handlers.
- The `/register` handler checks for existing users, hashes the password using `bcrypt`, creates a new User document in MongoDB, and sends a success response.
- The `/login` handler finds the user by identifier (username or email), compares the provided password with the hashed password in the database, and sends a success response (currently without setting a session).
- The `/findUser` handler retrieves user data based on the provided identifier.

- **Key Functions/Methods (within route handlers):**

- `User.findOne({ $or: [...] })`: Mongoose method to find a user by either username or email.
- `bcrypt.hash(password, saltRounds)`: Hashes the user's password.
- `bcrypt.compare(plainTextPassword, hashedPassword)`: Compares a plain text password with a hashed password.
- `User.save()`: Mongoose method to save a new user to the database.
- `req.session.userId = user._id` (Implementation needed): To store the user's

ID in the session upon successful login.

- req.session.destroy() (Implementation needed): To clear the user's session upon logout.

3. Shopping Cart Module Implementation Details

Frontend (React):

- **Data Flow:**

- Clicking the "Add to Cart" button on a product page triggers an API call (POST /add/laptop or /add/part) with the product ID, quantity, and user ID.
- The cart data is fetched from the /cart/:userId backend endpoint on user login or page load and stored in the global state.
- The ShoppingCart component displays the cart items from the global state.
- Updating quantities or removing items in the ShoppingCart component triggers API calls (POST /cart/update, POST /cart/remove) to the backend to update the cart in the database. Upon success, the frontend state is updated to reflect the changes.

- **Key Functions/Methods:**

- addToCart(productId, quantity, type, price): Sends a POST request to /add/laptop or /add/part.
- fetchCart(): Sends a GET request to /cart/:userId and updates the cart state.
- updateCartItemQuantity(itemId, quantity, type, partCategory): Sends a POST request to /cart/update.
- removeCartItem(itemId, type, partCategory): Sends a POST request to /cart/remove.
- displayCartItems(cart): Renders the list of items in the cart.
- calculateSubtotal(cart): Calculates the total price of items in the cart.

Backend (Express.js):

- **Data Flow:**

- POST requests to /add/laptop and /add/part update the cart field within the user's User document in MongoDB.
- GET requests to /cart/:userId retrieve the user's User document and send back the cart data (with populated product details).
- POST requests to /cart/update find the specific item in the user's cart and adjust its quantity.
- POST requests to /cart/remove find the specific item in the user's cart and

remove it.

- **Key Functions/Methods (within route handlers):**

- User.findById(userId): Mongoose method to find a user by their ID.
- user.cart.laptops.push(...), user.cart.parts[category].push(...): Array methods to add items to the cart.
- user.cart.laptops.find(...), user.cart.parts[category].find(...): Array methods to find existing items in the cart.
- user.cart.laptops.filter(...), user.cart.parts[category].filter(...): Array methods to remove items from the cart.
- user.save(): Mongoose method to save the updated user document.
- .populate(...): Mongoose method to replace the stored ObjectIds with the actual product documents when fetching the cart.

4. "Build a PC" Tool Module Implementation Details

Frontend (React):

- **Data Flow:**

- The PCBuilder component will likely have a multi-step form or a visual interface.
- As the user selects a component category, it will fetch available parts for that category from the backend (/api/all-parts?category=...).
- The user's selections for each component will be stored in the component's local state or a global state.
- The BuildSummary component will display the selected parts and the calculated total cost.
- An "Add Build to Cart" button will send the selected part IDs and quantities to the backend (likely a modified /add/part endpoint or a new /api/build-cart endpoint).
- A "Proceed to Checkout" button will send the build configuration to the /checkout/create/buildAPc backend endpoint.

- **Key Functions/Methods:**

- fetchPartsByCategory(category): Sends a GET request to /api/all-parts with the category.
- handleComponentSelect(category, partId): Updates the selected part for a given category in the state.
- checkCompatibility(): Implements or calls an API to check compatibility.

- `calculateBuildCost()`: Calculates the total price of the selected components.
- `addBuildToCart()`: Sends the build configuration to the cart API.
- `proceedToCheckout()`: Sends the build configuration to the checkout API.

Backend (Express.js):

- **Data Flow:**

- GET requests to `/api/all-parts?category=...` are handled to retrieve parts of a specific category.
- POST requests to `/checkout/create/buildAPc` receive the selected part IDs from the frontend.
- The backend retrieves the details of these parts from the `ComputerPart` model.
- A Checkout document is created with the selected parts.

- **Key Functions/Methods (within route handlers):**

- `ComputerPart.find({ category: ... })`: Mongoose method to find parts by category.
- `ComputerPart.findById(id)`: Mongoose method to find a part by its ID.
- Creation of a new Checkout document using the Checkout model.

5.1.3 Data Extraction and Data Scrapping

The entire data collection process for the product catalog was carried out using Python. Web scraping was performed using the BeautifulSoup library, which made it efficient to extract structured data from various e-commerce websites. Using BeautifulSoup, information such as product name, model, price, and description was scraped for laptops, CPUs, GPUs, and other PC components. Additionally, the corresponding product images were also fetched, enabling the creation of a visually rich and informative catalog. This automated approach significantly streamlined the data-gathering process, ensuring up-to-date and consistent information across the platform. It returns the data collected using Python in JSON format.

The code snippets of the python data scrapper are listed below:

```
import os
import time
import json
import requests
from pathlib import Path
from duckduckgo_search import DDGS
from PIL import Image
from bs4 import BeautifulSoup

# Categories of computer parts with specific Newegg category URLs
computer_parts = {
    "CPU": "https://www.newegg.com/p/pl?d=CPU",
    "GPU": "https://www.newegg.com/p/pl?d=Graphics+Card",
    "Motherboard": "https://www.newegg.com/p/pl?d=Motherboard",
    "RAM": "https://www.newegg.com/p/pl?d=Memory",
    "SSD": "https://www.newegg.com/p/pl?d=Solid+State+Drive",
    "HDD": "https://www.newegg.com/p/pl?d=Hard+Drive",
    "Cooling System": "https://www.newegg.com/p/pl?d=Cooling+Fan",
    "Case": "https://www.newegg.com/p/pl?d=PC+Case"
}

# Base directory for storing images and JSON data
base_path = Path("computer-parts")
base_path.mkdir(exist_ok=True)
```

```
def download_images(dest, query, max_results=20):
    """Searches and downloads images from DuckDuckGo."""
    dest.mkdir(exist_ok=True, parents=True)
    with DDGS() as ddgs:
        results = list(ddgs.images(query, max_results=max_results))

    image_paths = []
    for i, result in enumerate(results):
        img_url = result.get("image", "")
        if not img_url:
            continue
        try:
            img_data = requests.get(img_url, timeout=10).content
            img_path = dest / f"{i}.jpg"
            with open(img_path, "wb") as f:
                f.write(img_data)
            print(f"Downloaded: {img_url}")
            image_paths.append(str(img_path))
        except Exception as e:
            print(f"Failed to download {img_url}: {e}")
    return image_paths

def resize_images(folder, max_size=400):
    """Resizes images in a folder to max_size pixels."""
    for img_file in folder.glob("*.jpg"):
        try:
```

```

        img = Image.open(img_file)
        img.thumbnail((max_size, max_size))
        img.save(img_file)
        print(f"Resized: {img_file}")
    except Exception as e:
        print(f"Failed to resize {img_file}: {e}")

def scrape_computer_parts():
    """Scrapes computer parts data from Newegg."""
    headers = {"User-Agent": "Mozilla/5.0"}

    parts_data = {}
    for part, search_url in computer_parts.items():
        print(f"Scraping data for {part}...")
        try:
            response = requests.get(search_url, headers=headers, timeout=10)
            soup = BeautifulSoup(response.text, "html.parser")
            items = soup.select("div.item-cell")[:20] # Newegg uses 'div.item-cell' for product listings

            parts_data[part] = []
            for item in items:
                try:
                    brand_tag = item.select_one(".item-brand img")
                    brand = brand_tag["title"] if brand_tag else "Unknown"

                    model_tag = item.select_one(".item-title")
                    model = model_tag.text.strip() if model_tag else "Unknown"

```

```

                    description = f"{model} from {brand} is a high-quality component built for optimal performance,
                    efficiency, and durability. Ideal for gaming, professional workstations, and high-performance setups."

                    specifications = []
                    spec_list = item.select("ul.item-features li") # Newegg uses 'ul.item-features'
                    for spec in spec_list:
                        specifications.append(spec.text.strip())

                    image_tag = item.select_one(".item-img img")
                    image_url = image_tag.get("src", "") if image_tag else ""

                    parts_data[part].append({
                        "brand": brand,
                        "model": model,
                        "price": price,
                        "description": description,
                        "specifications": specifications if specifications else "Specifications not available.",
                        "image": image_url
                    })
                except Exception as e:
                    print(f"Error extracting data for {part}: {e}")
            except Exception as e:
                print(f"Failed to scrape {part}: {e}")

    return parts_data

```

```

# Scrape and save computer parts data
computer_parts_data = scrape_computer_parts()

# Loop through each computer part category
for part in computer_parts:
    part_path = base_path / part
    image_paths = download_images(part_path, query=f"{part} Computer Part", max_results=20)
    resize_images(part_path, max_size=400)

    # Assign images to data
    if part in computer_parts_data:
        for i, item in enumerate(computer_parts_data[part]):
            if i < len(image_paths):
                item["image"] = image_paths[i]

    time.sleep(3)

# Save updated data with images
json_path = base_path / "computer_parts.json"
with open(json_path, "w") as f:
    json.dump(computer_parts_data, f, indent=4)
print(f"Saved scraped parts data to {json_path}")

print("All images downloaded, resized, and data scraped!")

```

Figure 18: Code Snippets for Python Data Scraper

```
{
  "CPU": [
    {
      "brand": "AMD",
      "model": "AMD Ryzen 9 9950X - Ryzen 9 9000 Series Granite Ridge (Zen 5) 16-Core 4.3 GHz - Socket AM5 170W - Radeon Graphics Processor - 100-100001277WOF",
      "price": "$538.99",
      "description": "AMD Ryzen 9 9950X - Ryzen 9 9000 Series Granite Ridge (Zen 5) 16-Core 4.3 GHz - Socket AM5 170W - Radeon Graphics Processor - 100-100001277WOF from AMD is a high-quality component built for optimal performance, efficiency, and durability. Ideal for gaming, professional workstations, and high-performance setups.",
      "specifications": [
        "Model #: 100-100001277WOF"
      ],
      "image": "computer-parts\\CPU\\0.jpg"
    },
    {
      "brand": "AMD",
      "model": "AMD Ryzen 7 7700X - Zen 4 8-Core 4.5 GHz - Socket AM5 - 105W Desktop Processor (100-100000591WOF)",
      "price": "$272.74",
      "description": "AMD Ryzen 7 7700X - Zen 4 8-Core 4.5 GHz - Socket AM5 - 105W Desktop Processor (100-100000591WOF) from AMD is a high-quality component built for optimal performance, efficiency, and durability. Ideal for gaming, professional workstations, and high-performance setups.",
      "specifications": [
        "Model #: 100-100000591WOF"
      ],
      "image": "computer-parts\\CPU\\1.jpg"
    }
  ],
  "GPU": [
    {
      "brand": "ASRock",
      "model": "ASRock Radeon RX 6600 Challenger D 8GB GDDR6 PCI Express 4.0 Graphics Card RX6600 CLD 8G",
      "price": "$209.99\\u00a0\\u2013",
      "description": "ASRock Radeon RX 6600 Challenger D 8GB GDDR6 PCI Express 4.0 Graphics Card RX6600 CLD 8G from ASRock is a high-quality component built for optimal performance, efficiency, and durability. Ideal for gaming, professional workstations, and high-performance setups.",
      "specifications": [
        "Model #: RX6600 CLD 8G"
      ],
      "image": "computer-parts\\GPU\\0.jpg"
    },
    {
      "brand": "ASUS",
      "model": "ASUS TUF Gaming GeForce RTX 5080 16GB GDDR7 OC Edition TUF-RTX5080-016G-GAMING PCI-Express 5.0 DLSS 4.0 Graphics Card",
      "price": "$1,484.99\\u00a0\\u2013",
      "description": "ASUS TUF Gaming GeForce RTX 5080 16GB GDDR7 OC Edition TUF-RTX5080-016G-GAMING PCI-Express 5.0 DLSS 4.0 Graphics Card from ASUS is a high-quality component built for optimal performance, efficiency, and durability. Ideal for gaming, professional workstations, and high-performance setups.",
      "specifications": [
        "Model #: TUF-RTX5080-016G-GAMING"
      ],
      "image": "computer-parts\\GPU\\1.jpg"
    }
  ]
}
```

Figure 19: Snippets of JSON file

5.2. Testing

5.2.1 Test Cases for Unit Testing

Unit testing is a software testing method where individual components or units of code (like functions or methods) are tested in isolation to verify their correctness and ensure they function as expected.

Test Case ID: 1

Test Title: Verify login with valid username and password.

Test Designed Date: 25 February, 2025

Pre-conditions: User has valid username and password.

Dependencies:

Table 1: Unit testing 1

Test Description	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
Navigate to login page. Provide valid username. Provide valid password. Click login button.	Username: mandish_nv Password: mandish_nv	User should be able to login.	User is navigated to the landing page with successful login.	Pass
Provide invalid username and password.	Username: abc Password: 123	User login invalid.	User login invalid.	Pass

Post-condition:

User is validated and the page is redirected to the landing page with login status.

Test Case ID: 2 Test Title: Display items based on the selected field. Test Designed Date: 25 February, 2025
Pre-conditions: Items of different categories are stored in the database. Dependencies:

Table 2: Unit testing 2

Test Description	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
Select a category. View items in that category.	Category: CPU	Items belonging to “CPU” category should be displayed.	Items belonging to “CPU” category is displayed.	Pass
Select a category. View items in that category.	Category: Laptops	Items belonging to “Laptop” category should be displayed.	Items belonging to “Laptop” category is displayed.	Pass

Post-condition: The items belonging to the selected category are displayed.

Test Case ID: 3

Test Title: Add items to cart.

Test Designed Date: 25 February, 2025

Pre-conditions: The user must be logged in.

Dependencies:

Table 3: Unit testing 3

Test Description	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
Select an item to purchase. Select the quantity of items to purchase. Click on “Add to Cart” button.	Item: AMD - AMD Ryzen 7 7700X	The item should be added to cart along with the quantity.	The item is added to cart along with the quantity.	Pass
Select an item to purchase. Select the quantity of items to purchase. Click on “Add to Cart” button.	Item: Acer - America Acer Aspire 5 15.6" Touchscreen Laptop i7-13620H 16GB RAM 1 TB PCIe SSD Windows 11 Home A515-58PT-746F	The item should be added to cart along with the quantity.	The item is added to cart along with the quantity.	Pass

Post-condition:

The items are added to cart along with the selected quantity.

5.2.2 Test Cases for System Testing

System testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system.

Test Case ID: 4

Test Title: Functionality testing.

Test Designed Date: 25 February, 2025

Pre-conditions:

Table 4: System testing 1

Test Description	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
UI Workflow.	Landing page.	All the buttons and navigation bar contents should navigate to the correct location.	All the buttons and navigation bar contents navigate to the correct location.	Pass
User login and registration.	Correct user registration and login data.	The user should be able to register and then login to the website. The user should also be able to access the login features.	The user can register and then login to the website. The user can access the login features.	Pass

Post-condition:

The functional components are working as intended.

Test Case ID: 5 Test Title: Security testing. Test Designed Date: 25 February, 2025
Pre-conditions:

Table 5: System testing 2

Test Description	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
The user inserts valid username and password. (Authentication)	Login page.	The user should be successfully logged in and should have access to logged in features.	The user is successfully logged in and has access to logged in features.	Pass
Password should be encrypted. (Data Privacy)	Password: test	The password should be stored in database in encrypted format.	The password should is stored in database in encrypted format. Password: \$2b\$10\$1NJA v/81RpIX.YbH4ruWROEejCs/ZmVA7Ur24KlOeHi mtJ80qOzXi	Pass

Post-condition: The security components are working as intended.
--

5.3. Result Analysis

Overall Analysis of Unit Testing:

The unit tests provided show that the individual components tested (login, category-based item display, and adding to cart) are functioning as expected. All the executed unit tests have a "Pass" status. This suggests that the basic building blocks of the application are likely to work correctly in isolation.

Overall Analysis of System Testing:

The system tests provided offer a preliminary validation of the integrated system. They confirm that basic UI navigation, user authentication, and password encryption are functioning correctly. All the executed system tests have "Pass" status.

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

In conclusion, the **"HAVOCAURA – PC Parts & Laptop Ecommerce Website"** project aimed to create a robust and user-friendly online platform for purchasing computer components and laptops, addressing the challenges of a fragmented and often complex market. Through a comprehensive system analysis, utilizing an object-oriented approach, we defined clear functional and non-functional requirements, ensuring the platform's accuracy, efficiency, and security. The selection of the MERN stack, combined with the agile Feature-Driven Development methodology, facilitated the creation of a scalable and maintainable system. Feasibility analyses confirmed the project's technical, operational, economic, and schedule viability. The development of the "Build a PC" tool, alongside standard e-commerce features, provides users with a unique and valuable experience. By prioritizing user experience, product variety, and secure transactions, **"HAVOCAURA"** is poised to become a reliable and preferred destination for PC enthusiasts and general consumers alike, effectively bridging the gap between complex hardware needs and accessible online purchasing.

6.2 Future Recommendations

The project will be upgraded and extended with the following features:

1. Enhanced Order Management and Customer Communication:

- **Real-Time Order Tracking:** Implement a more detailed order tracking system with carrier information and estimated arrival times.
- **Proactive Shipping Notifications:** Develop automated notifications (email, SMS) to keep customers informed about their order's progress.

2. Customer Support and Engagement:

- **FAQ:** Introduce a FAQ forum for knowledge sharing and answering queries.
- **Reviews and Ratings:** Allow user to add their reviews and ratings for a product.
- **Live Chat Support:** Introduce a live chat feature for immediate assistance.

3. Optimization and Expansion:

- **Inventory Expansion:** Gradually expand the product inventory to include a wider

range of products.

- **Advanced Product Comparison Tool:** Develop and implement a fully featured advanced product comparison tool.

References

- [1] geeksforgeeks, "geeksforgeeks," 20 November 2024. [Online]. Available: <https://www.geeksforgeeks.org/fdd-full-form/>.
- [2] Amazon, "amazon," 20 April 2025. [Online]. Available: <https://www.amazon.com/>.
- [3] eBay, "eBay," [Online]. Available: <https://www.ebay.com/>. [Accessed 20 April 2025].
- [4] Nielsen Norman Group, "Nielsen Norman Group," [Online]. Available: <https://www.nngroup.com/topic/e-commerce/>. [Accessed 20 April 2025].
- [5] Newegg.com, "Newegg.com," [Online]. Available: <https://www.newegg.com/>. [Accessed 20 April 2025].
- [6] Micro Center, "Micro Center," [Online]. Available: <https://www.microcenter.com/>. [Accessed 20 April 2025].
- [7] L. PCPartPicker, "PCPartPicker," [Online]. Available: <https://pcpartpicker.com/>. [Accessed 20 April 2025].

Appendices

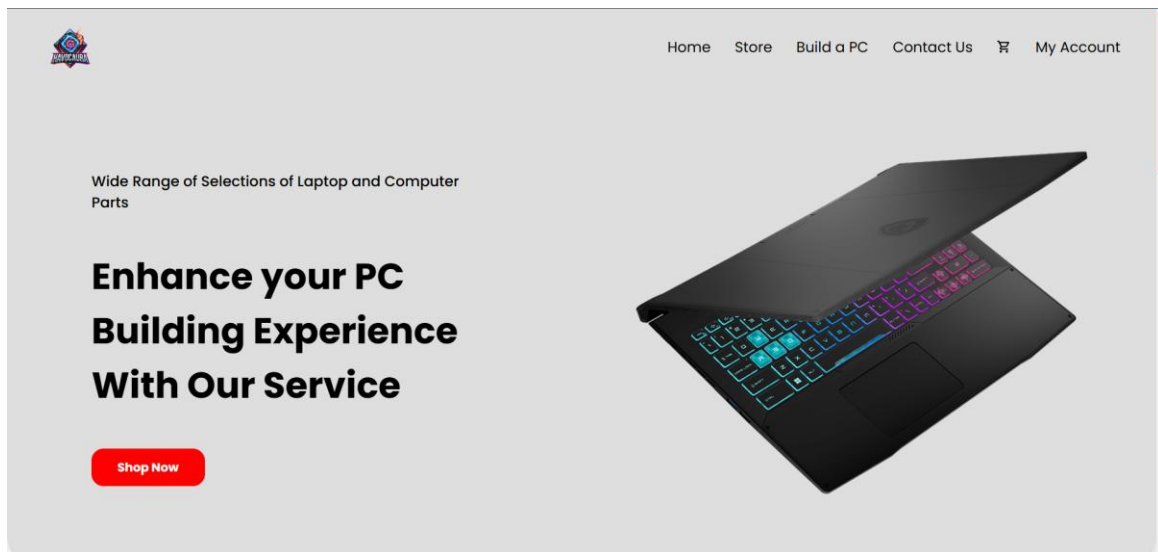


Figure 20: Landing Page Without Login

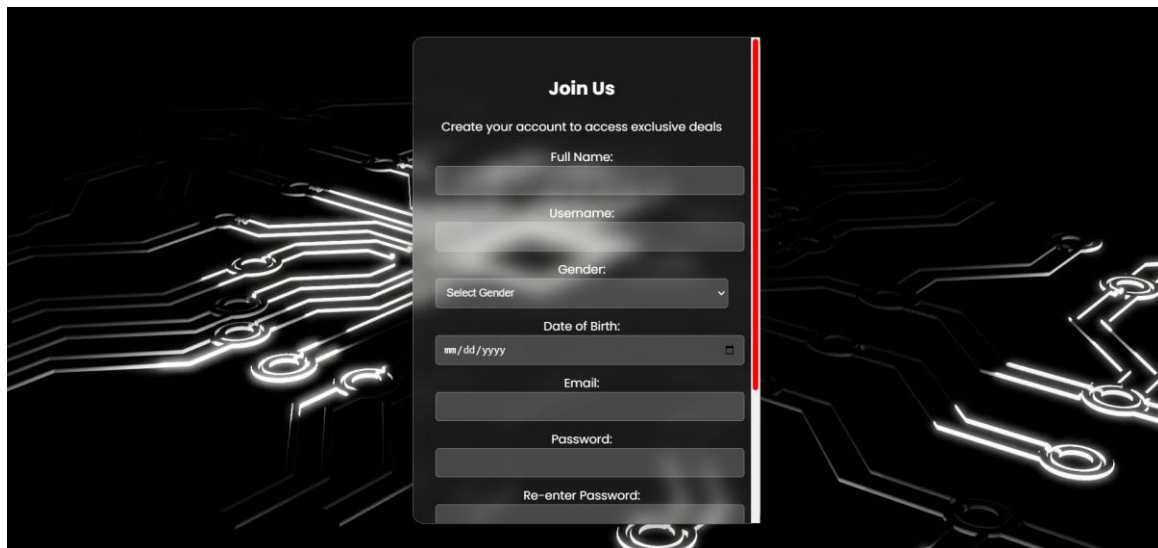


Figure 21: Registration Page

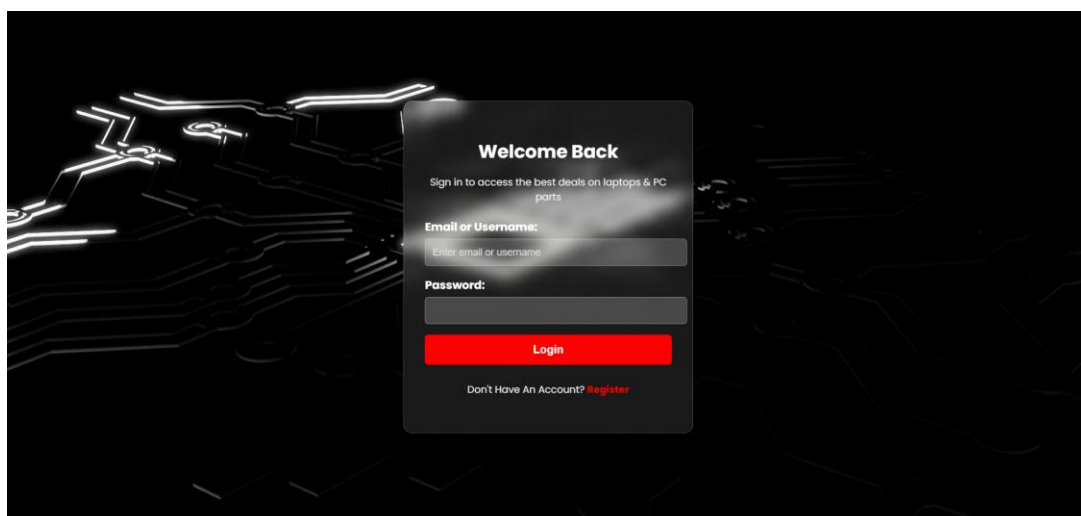


Figure 22: Login Page

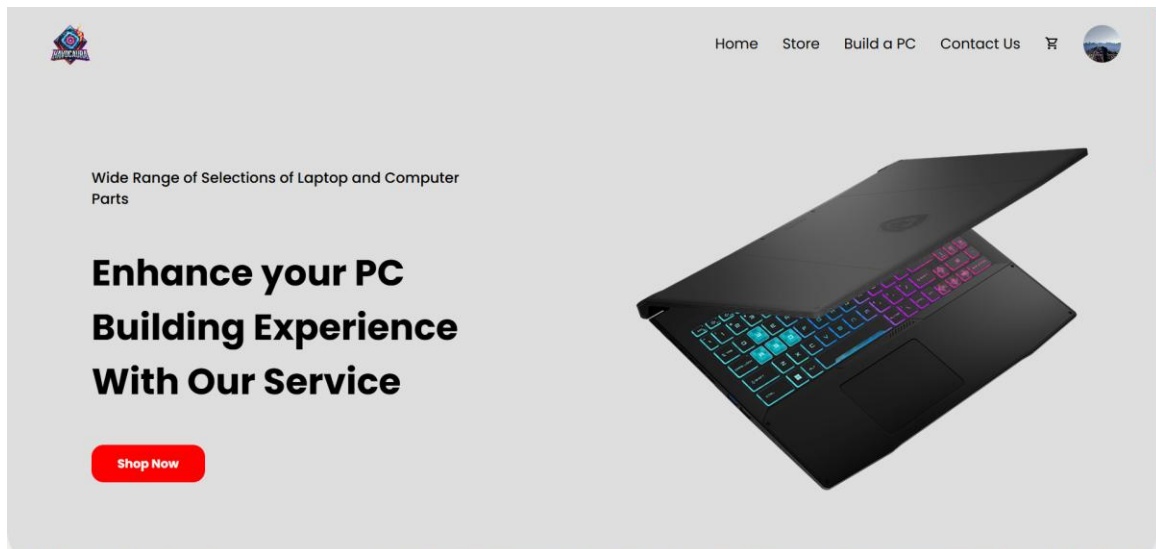


Figure 23: Landing Page With Login

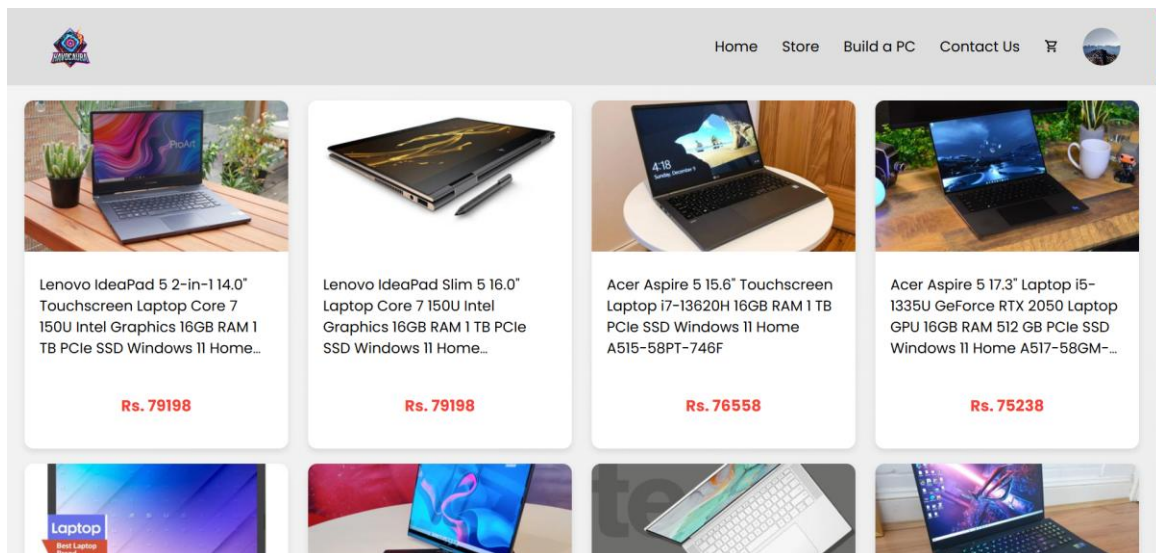


Figure 24: Laptop Store

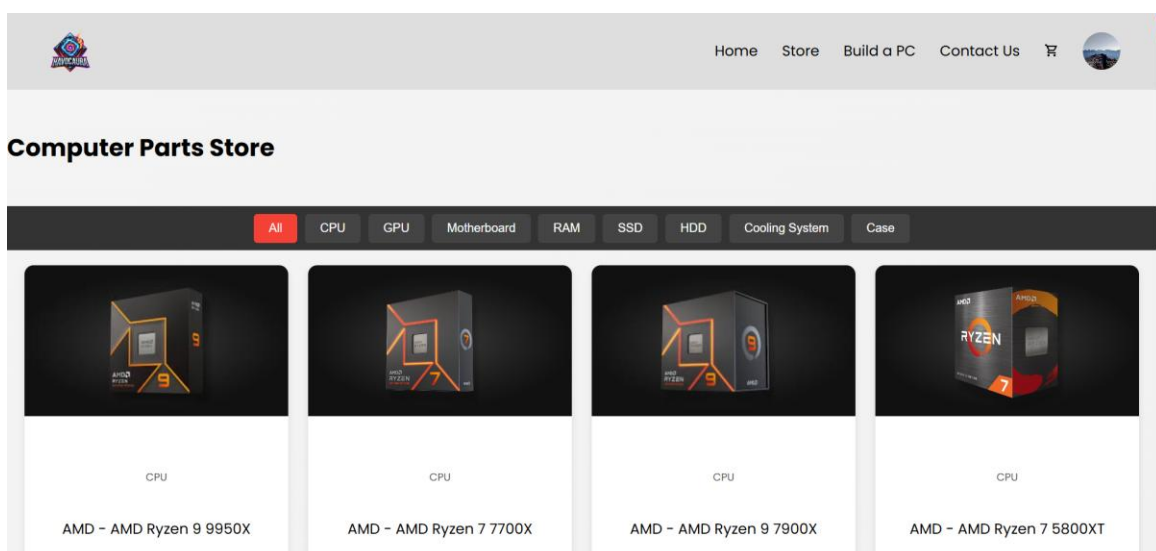


Figure 25: PC Parts Store

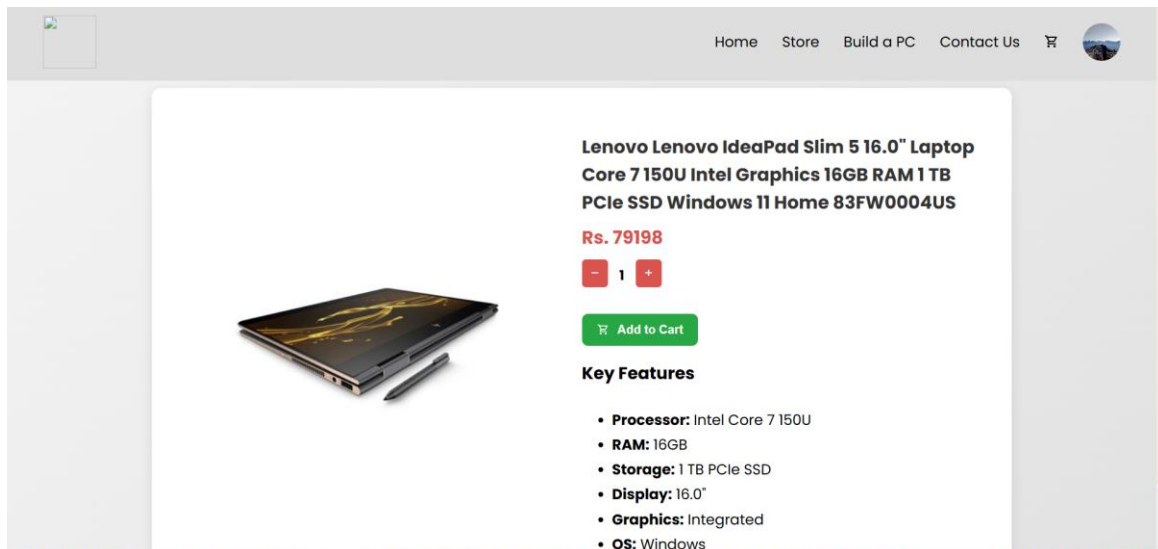


Figure 26: Laptop Display

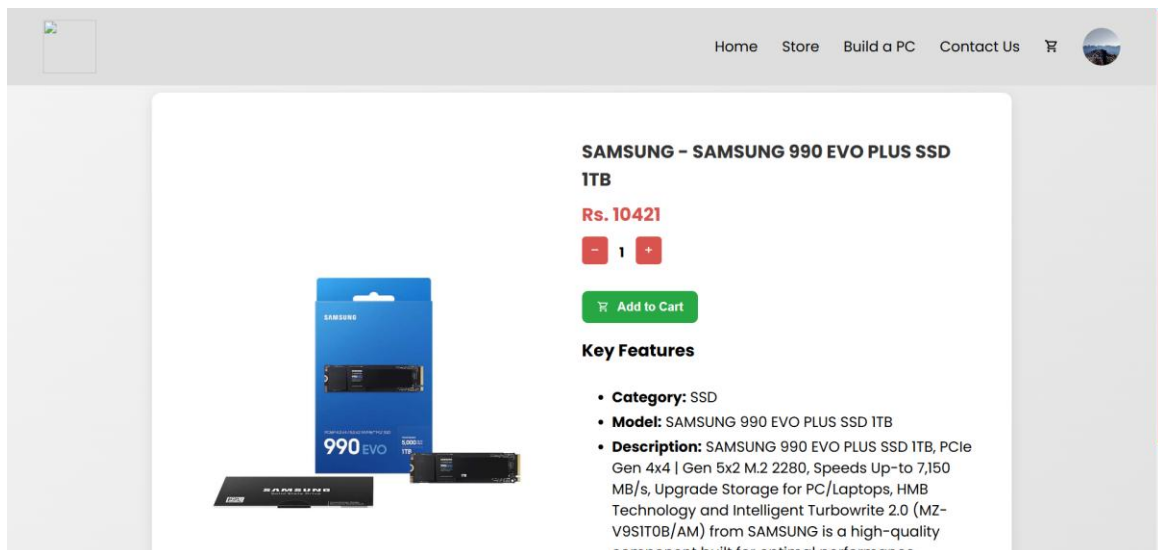


Figure 27: PC Parts Display

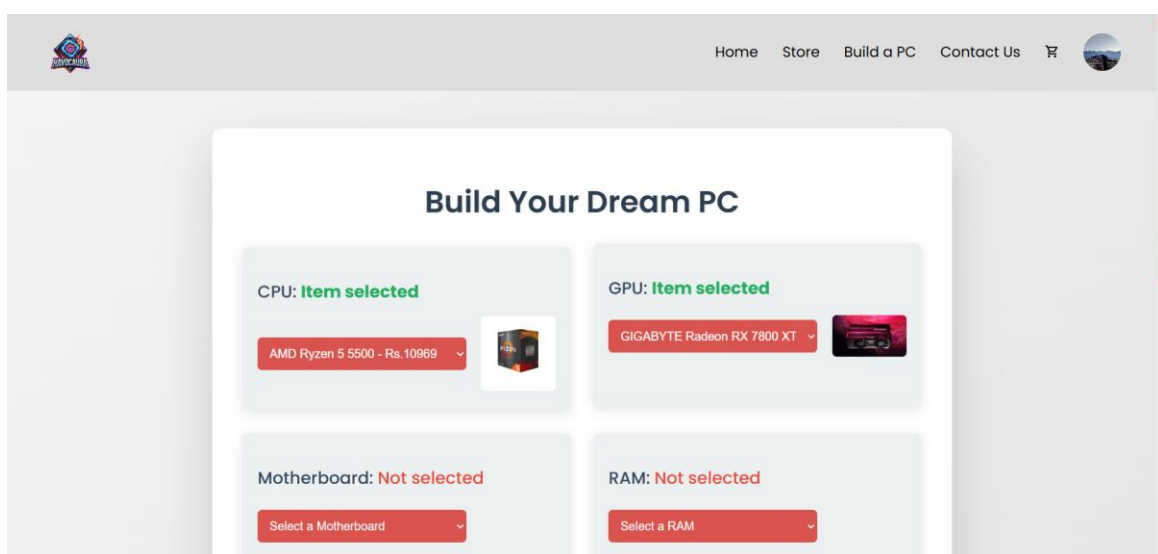


Figure 28: Build A PC

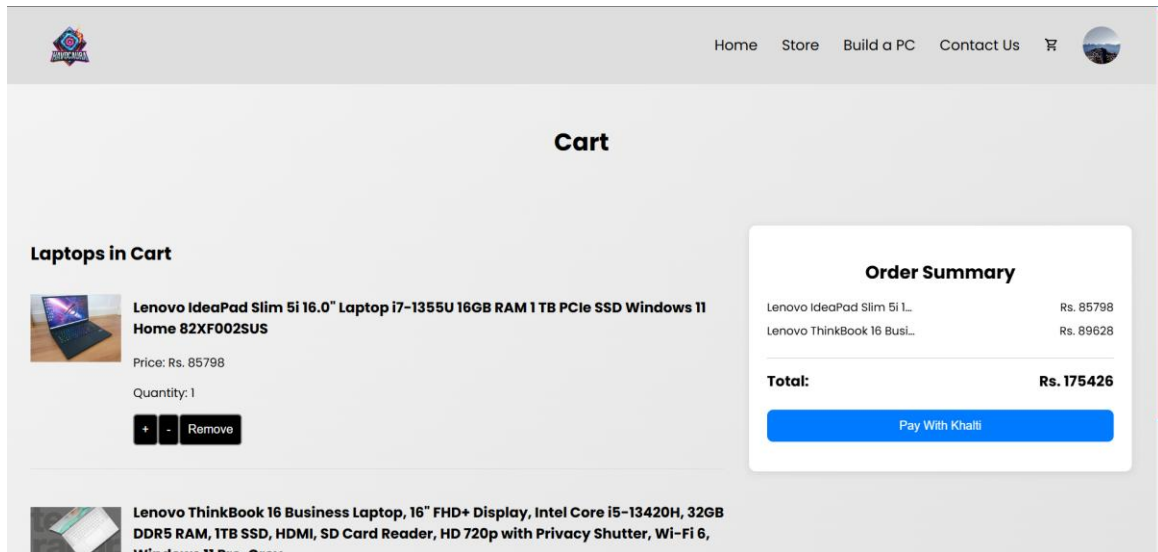


Figure 29: Cart Display

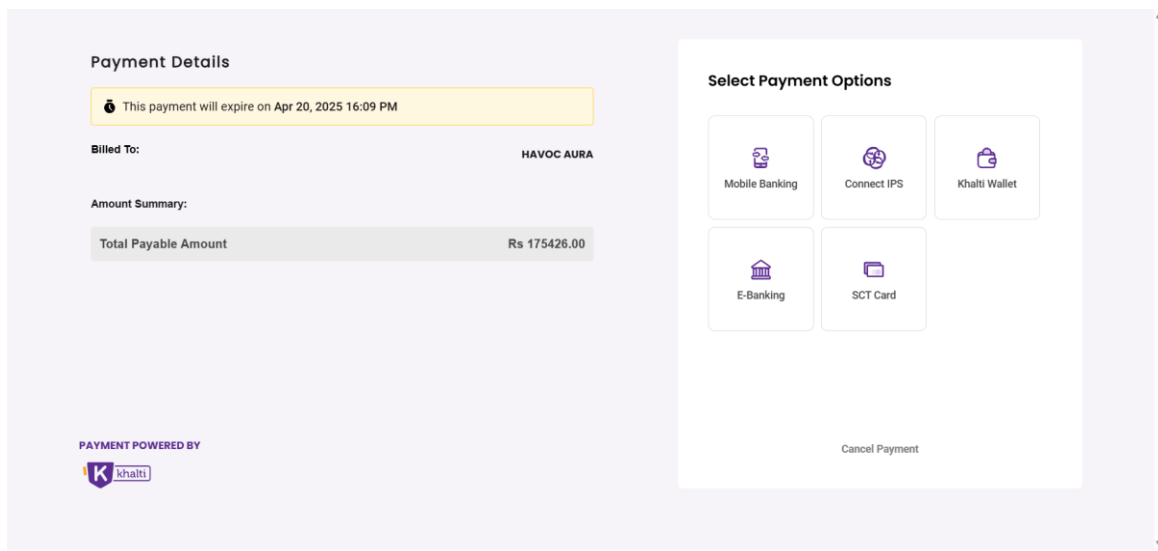


Figure 30: Khalti Payment Gateway

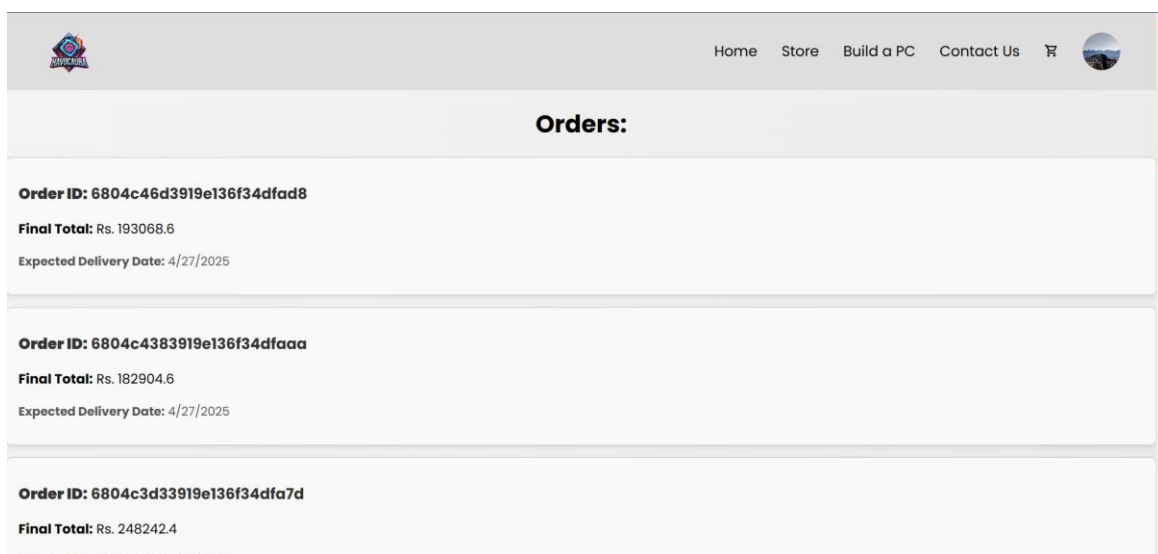


Figure 31: Order Page