Prezentarea unui site web de jucării folosind JavaScript, Node.js și React

Numele proiectului: KIDProf.ro

Slogan: „Aduceți bucurie cu fiecare clic”

Tehnologii utilizate:

Frontend: React

Backend: Node.js

Limbaj: JavaScript

Baza de date: MongoDB

Introducere în site-ul web

Obiectiv:

Creat un site web de jucării interactiv și atrăgător din punct de vedere vizual, unde utilizatorii pot:

* Vizualizati jucăriile după categorie.
* Vedeți detaliile jucăriei.
* Adăugați jucării într-un coș și faceți cumpărături.

Public țintă:

Părinți, cumpărători de cadouri și copii care caută cele mai noi jucării.

Caracteristici cheie

1.Interfață ușor de utilizat

Creat cu React pentru o interfață de utilizare dinamică și receptivă.

Navigare lină între pagini.

Listări interactive de produse.

Filtrați jucăriile după categorii.

Funcționalitate de căutare rapidă și dinamică cu JavaScript.

2.Coș și comandă

Adăugați, eliminați și editați articole din coș.

Proces de plată securizat.

Funcționalitatea backend

Node.js gestionează solicitările API și comunică cu baza de date.

Slide 3: Tehnologia Stack

Frontend: React

Componente reutilizabile pentru design modular.

Managementul stării cu Context API/Redux.

Backend: Node.js cu Express

API RESTful pentru o comunicare perfectă între interfață și baza de date.

Baza de date:

Am utilizat MongoDB pentru stocarea datelor despre jucării, conturi de utilizator și informații despre comandă.

JavaScript:

Se ocupă atât de logica frontend, cât și de backend.

Proiectarea paginii de pornire

Caracteristici de pe pagina principală:

Banner atrăgător care prezintă jucării prezentate.

Categorii pentru navigare rapidă (de exemplu, „Figurine de acțiune”, „Jucării educaționale”).

Pagina de produs

Caracteristici de pe pagina de produs:

Detalii jucărie: nume, descriere, preț.

Carusel interactiv de imagini.

Funcționalitatea „Adaugă în coș”.

Rol de backend:

Node.js preia datele despre produse din baza de date.

Punctele finale API gestionează cererile individuale de produse.

Coș și comandă

Caracteristici ale căruciorului:

Lista jucăriilor selectate cu cantitate și preț.

Calcularea prețului total în timp real folosind JavaScript.

Opțiune pentru a edita sau elimina elemente.

Vizualizarea costului de transport.

Procesul de plată:

Node.js procesează cererile de plată printr-un gateway de plată (de exemplu, Stripe).

Sistem securizat de confirmare a comenzii.

Arhitectura backend

Cum funcționează Node.js în proiect:

rutare:

/api/list - Preluați toate jucăriile.

/api/cart - Gestionați operațiunile coșului.

/api/order - Gestionează plasările de comenzi.

/api/add – Adaugati jucarii noi.

api/remove – Stergeti jucarii din lista de produse.

Middleware:

Middleware de autentificare pentru accesul utilizatorului securizat.

Provocări și soluții

Provocare: asigurarea unei filtrari rapide a produselor.

Soluție: Implementarea managementului eficient al statului în React.

Provocare: backend sigur și scalabil.

Soluție: am folosit Node.js cu autentificare adecvată și apeluri API optimizate.

Provocare: actualizări dinamice ale coșului.

Soluție: S-a folosit starea React pentru a reda eficient căruciorul.

Îmbunătățiri viitoare

Adăugați o funcție de listă de dorințe pentru utilizatori.

Implementați recenzii și evaluări ale utilizatorilor pentru jucării.

Integrați recomandări personalizate de jucării folosind AI.

Extinde opțiunile de plată.

Demo

Pagina de pornire.

Lista de produse.

Funcționalitatea coșului.

Procesul de plată.

Concluzie

Rezumat:

Site-ul cu jucării demonstrează modul în care JavaScript, React și Node.js pot construi o platformă de comerț electronic modernă, receptivă și scalabilă.

Daca sunt intrebari va stau la dispozitie.

* **React, useContext, useState:** Enables creating stateful and context-aware functional components.
* **LoginPopup.css:** Styles the modal popup.
* **assets:** Provides reusable assets like icons.
* **toast:** Displays success or error messages to users. Uses toast to provide real-time feedback for success or errors.
* **StoreContext:** Accesses shared state, such as the API base URL, token, and cart data.
* **setShowLogin:** A function passed as a prop to toggle the visibility of the login popup.

 **setToken:** Updates the token in the context upon successful login.

 **url:** The base URL for API calls.

 **loadCartData:** Loads the user's cart data after authentication.

 **currState:** Tracks whether the popup is in "Login" or "Sign Up" mode.

*  **data:** Holds form input values for name, email, and password.

 **React, useContext, useState:** Enables stateful and context-aware functional components.

 **Navbar.css:** Provides styles for the navbar.

 **assets:** Provides reusable assets such as icons and images.

 **Link, useNavigate:** Facilitates navigation within a React Router-enabled app.

*  **StoreContext:** Accesses shared state, such as the cart and user authentication token.