ZÁRÓDOLGOZAT

Gaschler Gergő
14/Sz

Kovács Zoltán
14/Sz

BMSzC Bolyai János Műszaki Technikum és Kollégium

Paté Kft. Webshop

Tartalomjegyzék

1. Bevezetés, téma ismertetése

A modern technológia és az online piac térhódítása egyre inkább meghatározza mindennapi életünket és üzleti tevékenységeinket. Az internetes kereskedelem, vagy más néven e-commerce, olyan dinamikusan fejlődő terület, melyet egyre többen fedeznek fel üzleti lehetőségként. Ebben a kontextusban különösen fontossá válik egy jól működő, felhasználóbarát webshop létrehozása és üzemeltetése.

A jelen program elkészítésének szakmai célkitűzése és indoklása egyaránt összefonódik a gyakorlati hasznosulással és a személyes érintettséggel. Az elkészült webshop Kovács Zoltán apukájának szól, aki éppen ezzel a területtel foglalkozik. A téma választását részben Kovács Zoltán saját személyes élethelyzete motiválta, hiszen az apja tapasztalatait és szakértelmét kihasználva valósította meg a projektet. Emellett a webshop létrehozása egyértelmű szakmai célokat is szolgál: lehetővé teszi az online jelenlét megerősítését, a termékek szélesebb körű elérhetőségét, valamint a digitális értékesítési csatornák hatékonyabb kihasználását.

2. Fejlesztői dokumentáció

a) Fejlesztőkörnyezet

A projekt fejlesztése során a Visual Studio Code (VSC), az XAMPP (MySQL) és a GitHub voltak az alapvető eszközeink. Ezek a kiválasztott eszközök különféle szempontok alapján lettek kiválasztva, és hatékonyan segítették minket a webshop létrehozásában.

A Visual Studio Code (VSC) egy könnyűsúlyú, mégis erőteljes fejlesztői környezet, amely lehetővé teszi a kódolás gyors és hatékony végrehajtását. A sokoldalúsága révén támogatja a különféle programozási nyelveket és eszközöket, ami ideálissá teszi a webfejlesztéshez. Emellett a VSC integrált Git támogatással rendelkezik, így könnyen és hatékonyan tudtunk dolgozni a GitHubon tárolt projekt verzióin.

Az XAMPP egy olyan fejlesztőkörnyezet, amely teljesen integrált szerveralkalmazást kínál, beleértve az Apache HTTP szerver, a MySQL adatbázis és a PHP szkriptmotor szoftvereket. A webshop adatbázisát MySQL segítségével kezeltük, és az XAMPP biztosította a megfelelő környezetet az adatbázisfejlesztéshez és teszteléshez.

A GitHub egy népszerű verziókezelő platform, amely lehetővé teszi a fejlesztők számára, hogy együtt dolgozzanak, nyomon kövessék a változásokat és együttműködjenek a projektben. A GitHubot a kódbázis verzióinak tárolására és nyilvántartására használtuk, így könnyen tudtunk együtt dolgozni és koordinálni egymással a fejlesztést.

Ezek az eszközök kombinációja lehetővé tette számunkra, hogy hatékonyan dolgozzunk együtt a projekt fejlesztése során, és segített nekünk elérni a kitűzött célokat. A választásukat az egyszerű használat, a funkcionalitás és a kompatibilitás jellemzi, amelyek kulcsfontosságúak voltak a projekt sikere szempontjából.

b) Hardveres Infrastruktúra

A projekt fejlesztése során összesen három különböző gépet használtunk. Két gépet az iskolai munkához és egyet otthoni feladatokhoz. Ez az elrendezés lehetővé tette, hogy a munkafolyamatunkat hatékonyan és gördülékenyen végezzük mind az iskolai, mind az otthoni környezetben.

Az iskolai munkavégzés során két különböző gépet használtunk, amelyek a sulis teremben voltak elhelyezve. Ezeket a gépeket az iskola biztosította, és azokon dolgoztunk, amikor az iskolában voltunk.

- **Operációs Rendszer**: Windows 10.
- **Hardver**: A gépek általában közepes teljesítményű asztali számítógépek voltak, amelyek képesek voltak futtatni a fejlesztői környezetet és a webböngészőt is.

    - **CPU**: Intel Core i5-6500 3.20Hz
    - **RAM**: 16GB
- **Internetkapcsolat**: Az iskolai hálózaton keresztül csatlakoztunk az internethez, amely lehetővé tette a külső erőforrások elérését és a projekt tárolását a felhőben.

Otthoni munkavégzés során egy asztali gépet használtunk, amely a saját otthoni környezetünkben volt elhelyezve. Ez az otthoni gép lehetővé tette számunkra, hogy otthonról is dolgozzunk, amikor az iskolai munka nem volt lehetséges.

- **Operációs Rendszer**: Windows 10 és 11.
- **Hardver**: Az otthoni gép egy asztali számítógép volt, amely erősebb teljesítményt biztosított a fejlesztéshez és a teszteléshez.
    - **CPU**: Intel Core i7-9700K
    - **RAM**: 32GB
- **Internetkapcsolat**: Az otthoni hálózaton keresztül csatlakoztunk az internethez, ami lehetővé tette az online források elérését és a projekt tárolását a felhőben.

Mind a három gépen ugyanazon fejlesztői környezetet és szoftvereket használtuk, így biztosítva a konzisztens fejlesztési környezetet és a hatékony együttműködést a projekt során.

c) Programnyelvek

HTML és CSS: Az HTML és a CSS alapvetőek a webfejlesztésben. Az HTML a weboldalak szerkezetét határozza meg, míg a CSS a megjelenésüket. Ezek a nyelvek alapvetően szükségesek bármilyen webes projekt megvalósításához, és a webshop esetében is elengedhetetlenek a felhasználóbarát és vonzó felület kialakításához.

JavaScript: A JavaScript egy dinamikus és sokoldalú nyelv, amely lehetővé teszi interaktív felhasználói élmények létrehozását a weboldalakon. Egy webshop esetében a JavaScript segítségével élő kereső funkciókat, kosárkezelést, valamint dinamikus tartalomfrissítéseket valósíthatsz meg, ami javítja az oldal felhasználói élményét és funkcionalitását.

Node.js: A Node.js egy olyan szerveroldali JavaScript környezet, amely lehetővé teszi a JavaScript futtatását a szerveren. Ennek használatával egy olyan egységes nyelvet használhatsz mind a kliensoldali, mind a szerveroldali fejlesztéshez. Ez hatékonyabbá és összehangoltabbá teszi a fejlesztési folyamatot, valamint lehetővé teszi a könnyebb adatkezelést és szerveroldali funkciók kialakítását, például az adatbáziskezelés szempontjából.

SQL (Structured Query Language): Az SQL egy adatbáziskezelési nyelv, amelyet a relációs adatbázisok lekérdezésére és manipulálására használnak. A webshop adatbázisának

kezeléséhez SQL-t alkalmaztál, amely lehetővé teszi az adatbázisból történő adatlekérdezéseket, adatok beillesztését, módosítását és törlését.

d) Adatszerkezet

1. Felhasználók (users)

Tábla azonosítója: „users”

Mezők részletes leírása
id: Egyedi azonosító, automatikusan növekszik.
username: Felhasználónév, maximum 255 karakter hosszú, nem lehet üres.
email: Email cím, maximum 255 karakter hosszú, nem lehet üres.
password: Jelszó, maximum 255 karakter hosszú, nem lehet üres.
admin: Adminisztrátor jogosultságokat jelzi, kis egész szám (0 vagy 1), alapértelmezetten 0.
last_name: Vezetéknév, maximum 255 karakter hosszú, nem lehet üres.
first_name: Keresztnév, maximum 255 karakter hosszú, nem lehet üres.
phone_number: Telefonszám, maximum 20 karakter hosszú, nem lehet üres.
city: Város, maximum 255 karakter hosszú, nem lehet üres.
zip_code: Irányítószám, maximum 10 karakter hosszú, nem lehet üres.
address: Lakcím, maximum 255 karakter hosszú, nem lehet üres.
tax_number: Adószám, maximum 20 karakter hosszú.
company_name: Cégnév, maximum 255 karakter hosszú.

2. Termékek (products)

Tábla azonosítója: „products”

Mezők részletes leírása
id: Egyedi azonosító, automatikusan növekszik.
name: Termék neve, maximum 255 karakter hosszú, nem lehet üres.
description: Termék leírása, szöveges típus.
category: Termék kategóriája, maximum 255 karakter hosszú.
price: Termék ára, decimális típusú (10,2), nem lehet üres.

image_url: Termék képének URL-je, maximum 255 karakter hosszú.
stock: Készletmennyiség, egész szám, nem lehet üres.

3. Kosár (cart)

Tábla azonosítója: „cart”

Mezők részletes leírása
id: Egyedi azonosító, automatikusan növekszik**.**
user_id: Felhasználó azonosítója, idegen kulcs a **users** táblához.
product_id: Termék azonosítója, idegen kulcs a **products** táblához**.**
quantity: Termék mennyisége a kosárban, egész szám, nem lehet üres.

4. Rendelések (orders)

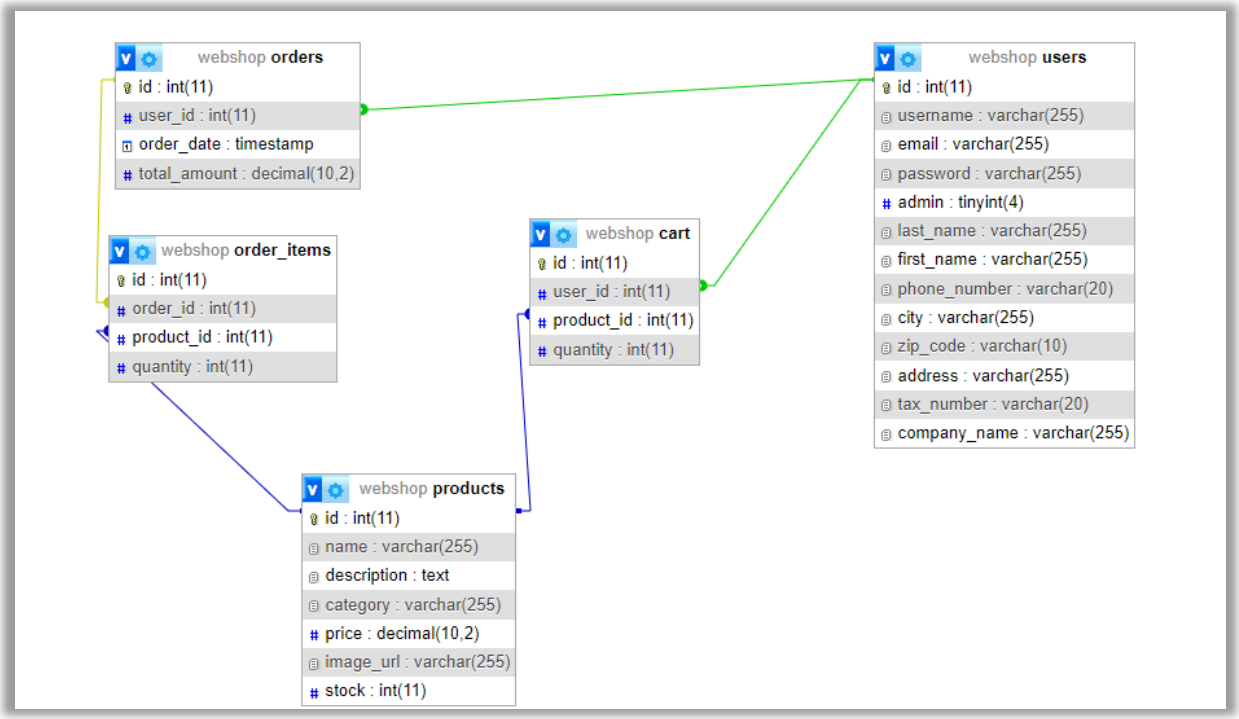
Tábla azonosítója: „orders”

Mezők részletes leírása
id: Egyedi azonosító, automatikusan növekszik.
user_id: Felhasználó azonosítója, idegen kulcs a **users** táblához.
order_date: Rendelés dátuma, időbélyeg típus, alapértelmezett érték a jelenlegi időpont.
total_amount: Rendelés teljes összege, decimális típusú (10,2), nem lehet üres.

5. Rendelés tételek (order_items)

Tábla azonosítója: „order_items”

Mezők részletes leírása
id: Egyedi azonosító, automatikusan növekszik.
order_id: Rendelés azonosítója, idegen kulcs az **orders** táblához.
product_id: Termék azonosítója, idegen kulcs a **products** táblához.
quantity: Termék mennyisége a rendelésben, egész szám, nem lehet üres.



e) Könyvtárszerkezet

A projektünk könyvtár szerkezete jól szervezett és átlátható. Két fő mappával kezdődik: "frontend" és "backend".

A "backend" mappában a routok egy külön mappában vannak elhelyezve, ami segíti az egyszerű navigációt és karbantarthatóságot. A "backend" mappában a "route" mappán kívül további fontos fájlok is találhatóak. Ezek közé tartozik az adatbázis kapcsolatot kezelő "db.js", a JWT tokenekkel foglalkozó "jwt.js", valamint maga a szerver indítását és konfigurációját végző "server.js" fájlok.

A "frontend" mappa további két almappát tartalmaz. Az egyik az "html" mappa, ami az HTML fájlokat tartalmazza, míg a másik a "css" mappa, ami a stíluslapokat tárolja.

Emellett a "frontend" mappában egy további "img" almappát találunk, amely a webshopban használt képeket tartalmazza.

Ez a struktúra segíti a projektünk szervezettségét és könnyű kezelhetőségét.

f) Algoritmusok

1) JWT.js fájl

```
1   const { sign, verify } = require("jsonwebtoken");
2   const connection = require("./db");
3
4   const createTokens = (id) => {
5     const accessToken = sign({ id: id }, "123");
6
7     return accessToken;
8   };
9
10  const validateToken = (req, res, next) => {
11    const accessToken = req.cookies["access-token"];
12
13    if (!accessToken)
14      return res.status(400).json({ error: "User not Authenticated!" }
15
16    try {
17      const validToken = verify(accessToken, "123");
18      if (validToken) {
19        req.user = validToken.id;
20        return next();
21      }
22    } catch (err) {
23      return res.status(400).json({ error: err });
24    }
25  };
26
27  module.exports = { createTokens, validateToken };
```

Először is, a kód betölt két függvényt a **jsonwebtoken** modulból, amelyeket a későbbiekben használni fog:

- **sign**: Ezzel a függvénnyel lehet JWT-t generálni a megadott adatok alapján.
- **verify**: Ez a függvény ellenőrzi egy JWT érvényességét.

Ezután a kód egy adatbázis kapcsolatot betölt, valószínűleg a **./db** fájlban lévő kapcsolatot.

A **createTokens** függvény egy paramétert vár, az **id**-t, amelyet a JWT payload-jában tárol. Ez a függvény egy JWT-t generál az adott **id**-val, és egy fix titkossal aláírja (**"123"**).

A **validateToken** függvény egy middleware, amelyet útvonalakhoz lehet csatolni, hogy ellenőrizze a kliens által küldött JWT-t. Először megpróbálja kinyerni az "access-token" nevű sütit a kérésből, ami a JWT-t tartalmazza. Ha nincs JWT az accessToken-ban, akkor hibát küld vissza a válaszban.

Ha van JWT, akkor megpróbálja ellenőrizni annak érvényességét a **verify** függvénnyel, használva a korábban említett fix titkot. Ha az ellenőrzés sikeres, akkor a **req** objektumba beállítja a felhasználó

azonosítóját, majd hívja a **next()** függvényt, ami továbbítja a kérést a következő middleware-nek vagy a kérés kezelőjének.

2) Login.js fájl

```
router.get("/", (req, res) => {
  res.sendFile(path.join(__dirname, "../../frontend/login.html"));
});

router.post("/", (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).json({ message: "Hiányzó email vagy jelszó" });
  }

  connection.query(
    "SELECT * FROM users WHERE email = ?",
    [email],
    async (error, results, fields) => {
      if (error) {
        return res
          .status(500)
          .json({ message: "Hiba történt a bejelentkezés során" });
      }

      if (results.length === 0) {
        return res.status(401).json({ message: "Hibás email cím vagy jelszó" });
      }

      const user = results[0];

      const match = await bcrypt.compare(password, user.password);
      if (!match) {
        return res.status(401).json({ message: "Hibás email cím vagy jelszó" });
      }

      const token = jwt.createTokens(user.id);

      res
        .cookie("access-token", token, { httpOnly: true })
        .json({ message: "Sikeres bejelentkezés" });
    }
  );
});

module.exports = router;
```

Az első útvonal egy GET kérésre válaszol, és visszaküldi a login oldal HTML fájlját, amelyet a **sendFile** metódus segítségével olvas be és küld vissza.

A második útvonal egy POST kérést fogad a bejelentkezési adatokkal. Ezeket az adatokat, az email és a jelszó, az **req.body** objektumból kinyerve használja. Ha valamelyik adat hiányzik, akkor egy 400-as hibakóddal visszaküld egy JSON választ a hiányzó adatokról.

Ezután a kód egy adatbázis lekérést futtat, hogy ellenőrizze, hogy van-e a megadott email címmel regisztrált felhasználó az adatbázisban. Ha hiba történik a lekérdezés során, akkor egy 500-as hibakóddal tér vissza. Ha a lekérdezés eredménye üres, azaz nem található felhasználó az adott email címmel, akkor egy 401-es hibakóddal visszaküldi a hibát.

Ha van eredmény a lekérdezésben, akkor ellenőrzi a megadott jelszót a tárolt jelszóval a **bcrypt.compare** segítségével. Ha a jelszavak egyeznek, akkor egy JWT-t generál a felhasználó azonosítójával, és visszaküldi ezt a tokent HTTP sütiként az "access-token" néven. Ezután egy JSON választ küld vissza a sikeres bejelentkezésről.

Végül, a modul exportálja a **router** objektumot, hogy más helyeken is használható legyen. Ez a kód a bejelentkezés kezelését végzi Express alkalmazásban.

3) Upload.js

```
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, path.join(__dirname, "../../frontend/uploads"));
  },
  filename: function (req, file, cb) {
    cb(null, Date.now() + "-" + file.originalname);
  },
});

const upload = multer({
  storage: storage,
  limits: { fileSize: 1000000 },
}).single("image");

router.post("/", (req, res) => {
  upload(req, res, (err) => {
    if (err) {
      console.error("Hiba történt a fájl feltöltésekor:", err);
      return res
        .status(500)
        .json({ message: "Hiba történt a fájl feltöltésekor" });
    }

    const { name, description, category, price, stock } = req.body;
    const imageUrl = req.file ? req.file.filename : null;

    const query =
      "INSERT INTO products (name, description, category, price, image_url, stock) VALUES (?, ?, ?, ?, ?, ?)";
    connection.query(
      query,
      [name, description, category, price, imageUrl, stock],
      (error, results, fields) => {
        if (error) {
          console.error(
            "Hiba történt az adatbázisba történő beszúrás során:",
            error
          );
          return res.status(500).json({
            message: "Hiba történt az adatbázisba történő beszúrás során",
          });
        }
        res.status(201).json({ message: "A termék sikeresen feltöltve." });
      }
    );
  });
});

module.exports = router;
```

A **multer** modul segítségével konfigurálva van egy fájltároló rendszer, amely meghatározza, hogy hova és milyen néven mentsen el fájlokat. A **storage** objektum a **diskStorage** metódust használva állítja be a fájlok mentési helyét és nevét. A **destination** metódus meghatározza a célmappát, ahová a fájlok mentésre kerülnek, míg a **filename** metódus a fájlok nevének kialakítását végzi, ami a fájl eredeti nevéhez hozzáadja a feltöltés időbélyegét.

Ezután van egy **upload** middleware, amelyet a POST kérésekhez csatolnak. Ez a middleware végzi el a fájl feltöltését a megadott konfiguráció szerint.

A **router.post()** metódus kezeli a POST kéréseket a **/** útvonalon. A fájlfeltöltés előtt meghívja az **upload** middleware-t. Ha hiba történik a feltöltés során, a middleware visszaküld egy 500-as hibakódot és egy hibaüzenetet a válaszban.

Ha a feltöltés sikeres volt, a metódus kinyeri a többi adatot a **req.body** objektumból (például a termék nevét, leírását, kategóriáját stb.). Az image_url változóba a feltöltött fájl nevét menti el, amelyet a **req.file** objektum tartalmaz. Ezután egy INSERT SQL lekérdezést futtat az adatbázisban, hogy elmentse a termék adatait és a feltöltött fájl nevét. Ha hiba történik a lekérdezés során, akkor a metódus egy 500-as hibakódot küld vissza a hibaüzenettel. Ha minden rendben ment, akkor a metódus egy 201-es státuszkódot küld vissza a sikeres feltöltés üzenetével.

Végül, a modul exportálja a **router** objektumot, hogy más helyeken is használható legyen. Ez a kód tehát a fájlfeltöltési funkcionalitást valósítja meg Express alkalmazásban.

4) Register.js

```
7   router.get("/", (req, res) => {
8     res.sendFile(path.join(__dirname, "../../frontend/register.html"));
9   });
10
11  router.post("/", (req, res) => {
12    const { username, email, password } = req.body;
13
14    if (!username || !email || !password) {
15      return res
16        .status(400)
17        .json({ message: "Hiányzó felhasználónév, email vagy jelszó" });
18    }
19
20    bcrypt.hash(password, 10, (err, hash) => {
21      if (err) {
22        return res
23          .status(500)
24          .json({ message: "Hiba a jelszó titkosítása közben" });
25      }
26
27      const user = { username, email, password: hash };
28      connection.query(
29        "INSERT INTO users SET ?",
30        user,
31        (error, results, fields) => {
32          if (error) {
33            return res
34              .status(500)
35              .json({ message: "Hiba a regisztráció közben" });
36          }
37          res.status(201).json({ message: "Sikeres regisztráció" });
38        }
39      );
40    });
41  });
42
43  module.exports = router;
```

Az útvonalak meghatározzák, hogy a GET kérések esetén a **/** útvonalon a **register.html** fájlt küldi vissza a kliensnek, amely a regisztrációs űrlapot tartalmazza.

A POST kérések esetén kinyeri a felhasználónév, email és jelszó adatokat a **req.body** objektumból. Ha valamelyik adat hiányzik, akkor egy 400-as hibakódot küld vissza a hiányzó adatokról szóló üzenettel.

A jelszót bcrypt segítségével titkosítja a **bcrypt.hash** függvény segítségével. A 10 a só mennyiségét jelöli, amely befolyásolja a titkosítás erősségét. A titkosítás után egy új felhasználó objektumot hoz létre a felhasználónévvel, email címmel és a titkosított jelszóval.

Ezután egy adatbázis lekérdezést futtat, hogy elmentsen egy új felhasználót az adatbázisban. Ha hiba történik a lekérdezés során, a metódus egy 500-as hibakódot küld vissza a hibaüzenettel. Ha minden rendben ment, a metódus egy 201-es státuszkódot küld vissza a sikeres regisztráció üzenetével.

Végül, a modul exportálja a **router** objektumot, hogy más helyeken is használható legyen. Ez a kód tehát a regisztrációs funkciót valósítja meg Express alkalmazásban.

g) Fejlesztési lehetőségek

1. Személyre szabott ajánlatok és tartalom

Az egyik legfontosabb tényező a webshop további fejlesztésében az egyedi és személyre szabott felhasználói élmény nyújtása. Ennek érdekében javasoljuk az algoritmusok és mesterséges intelligencia felhasználását annak érdekében, hogy az oldal a felhasználók viselkedése alapján személyre szabott ajánlatokat és tartalmakat tudjon megjeleníteni. Például, ha egy felhasználó többször böngészi az Adidas futócipőket, az oldal a továbbiakban olyan futócipőket ajánlhat neki, amelyek hasonló stílusúak vagy árkategóriában vannak.

2. Bővített fizetési lehetőségek

A különböző fizetési módok lehetőséget adnak arra, hogy a felhasználók könnyebben és gyorsabban vásároljanak. Javasoljuk a kriptovaluták elfogadását, valamint az alternatív fizetési módokat, mint például a részletfizetés vagy a készpénzmentes fizetési lehetőségek integrálását.

3. Továbbfejlesztett keresési és szűrési lehetőségek

Az egyre növekvő termékkínálat mellett fontos, hogy a felhasználók könnyen és gyorsan megtalálják azokat a termékeket, amelyekre szükségük van. Továbbfejlesztett keresési és szűrési lehetőségekkel, például árkategóriák szerinti szűréssel vagy termékkategóriák közötti gyors váltással javíthatjuk a felhasználói élményt.

4. Többnyelvű támogatás

A nemzetközi piacok elérésének érdekében fontos lehet a többnyelvű támogatás bevezetése. Ennek segítségével a nem angol anyanyelvű felhasználók könnyebben navigálhatnak az oldalon és értik a termékek leírását, ezáltal növelve a vásárlási hajlandóságot.

5. Továbbfejlesztett analitika és adatkezelés

Az adatok és az analitika fontos szerepet játszanak a webshop hatékonyságának mérésében és optimalizálásában. Javasoljuk a továbbfejlesztett analitikai eszközök és adatkezelési technikák bevezetését annak érdekében, hogy pontosabb és részletesebb információkat nyerjünk a felhasználói viselkedésről és a vásárlási szokásokról.

6. Regisztrációs értesítés email

A felhasználók regisztrációja fontos pillanat az üzlet életében. Azonnali értesítést küldhetünk nekik az emailben, hogy köszöntsük őket a webshopunkban, és részletezzük a fiókjukhoz kapcsolódó fontos információkat.

7. Hírlevél funkció

A hírlevél funkció lehetővé teszi számunkra, hogy rendszeresen kommunikáljunk a felhasználókkal, tájékoztassuk őket az új termékekről, akciókról vagy érdekes tartalmakról. Ez lehetőséget nyújt a közvetlen kapcsolattartásra és a vásárlói érdeklődés fenntartására.

3. Felhasználói dokumentáció

a) Program céljának és lényegesebb funkcióinak összefoglalása

1. Cél és Funkciók Összefoglalása

A Cipzár Webshop célja, hogy egyszerű és kényelmes megoldást nyújtson cipzárok vásárlására az interneten keresztül. A webshop főbb funkciói közé tartoznak:

a) Termékek Böngészése és Keresése

- A webshopon könnyen böngészheti a különböző típusú cipzárokat kategóriák szerint.
- Keresőfunkció segítségével gyorsan megtalálhatja a kívánt terméket.

b) Termék Részletek és Leírás

- Minden termékhez részletes leírás tartozik, beleértve az anyagot, méreteket és elérhető színeket.
- Termékfotók segítségével részletesen megtekintheti a termékeket.

c) Kosárkezelés

- Kiválaszthatja és hozzáadhatja a kívánt terméket a kosarához.
- A kosár oldalon ellenőrizheti és módosíthatja a kiválasztott termékeket.
- Biztonságos fizetési folyamat a kosár oldalról.

d) Fizetés és Rendelés

- A fizetés egyszerű és biztonságos módon történik a kiválasztott fizetési mód segítségével.
- Rendelési visszaigazolás e-mailt kap a sikeres tranzakcióról.

e) Fiókkezelés

- Regisztrálhat egy fiókot a webshopon belül, vagy bejelentkezhet meglévő fiókjába.
- Fiókjában megtekintheti korábbi rendeléseit és módosíthatja személyes információit.

b) Szükséges hardvereszközök és szoftverek felsorolása

Szoftverek:

1. **Visual Studio Code (VSC)**: Ez egy könnyűsúlyú, ingyenes és nyílt forráskódú kódszerkesztő, amely hasznos lehet a webshop fejlesztéséhez és teszteléséhez.
2. **Google Chrome**: A Google Chrome böngésző elengedhetetlen a webshop tervezéséhez és teszteléséhez, mivel segítséget nyújt a weboldalak megjelenítéséhez és a fejlesztés során.

3. **XAMPP**: Ez egy ingyenes, nyílt forráskódú szoftvercsomag, amely segíti a szerverkörnyezet (Apache, MySQL, PHP, Perl) helyi telepítését és konfigurálását fejlesztési vagy tesztelési környezetben.

4. **Opera GX**: Az Opera GX egy speciálisan tervezett webböngésző játékosok számára, amely különleges funkciókkal rendelkezik a jobb játékélmény és teljesítmény érdekében. Ez a böngésző tartalmaz olyan eszközöket, mint például a RAM és CPU limiter, amelyek segítenek optimalizálni a rendszer erőforrásainak felhasználását és minimalizálni a háttérben futó folyamatok hatását a játékok teljesítményére. Emellett az Opera GX a hagyományos Opera böngésző összes funkciójával rendelkezik, beleértve a gyors böngészést, beépített biztonsági funkciókat és számos testre szabási lehetőséget.

5. **Teams**: A Microsoft Teams egy teljes körű kommunikációs és együttműködési platform, amely lehetővé teszi a felhasználók számára az üzenetküldést, az online megbeszéléseket, a fájlok megosztását és a projektmenedzsmentet egyetlen alkalmazásban. Ez az eszköz különösen hasznos csapatoknak és csoportoknak, akik együtt dolgoznak egy projekt vagy feladat megoldásán, mivel lehetővé teszi az egyszerű és hatékony kommunikációt és együttműködést a távoli munkavégzés során is.

6. **Discord**: A Discord egy ingyenes és könnyen használható hang- és szöveges kommunikációs platform, amelyet széles körben használnak játékosok, közösségek és csoportok számára. Ez a platform lehetővé teszi a felhasználók számára az azonnali üzenetküldést, hanghívásokat és video chatet, valamint a különböző szerverek létrehozását és kezelését a közösségi és csapatalapú tevékenységekhez. A Discord emellett számos funkciót kínál a közösségi együttműködéshez és szórakozáshoz, például botok, hangcsatornák és szerepjátékok formájában. Telepítés és indítás lépéseinek ismertetése

Hardverek:

1. Iskolai gépek:
    - **Operációs rendszer:** Windows 10
    - **Hardver:**
        - CPU: Intel Core i5-6500 3.20GHz
        - RAM: 16GB
    - Internetkapcsolat: Iskolai hálózat
2. Otthoni gép:
    - **Operációs rendszer:** Windows 10 és 11
    - **Hardver:**
        - CPU: Intel Core i7-9700K
        - RAM: 32GB
    - Internetkapcsolat: Otthoni hálózat

c) Webshop bemutatása

Az újonnan létrehozott webshop rendszerében lehetőség van regisztrációra, mely egy személyre szabott és kényelmes vásárlási élményt biztosít. A regisztráció folyamata egyszerű és gyors, ahol a felhasználók megadhatják személyes adataikat, mint például nevüket, címüket, telefonszámukat és email címüket. Ezek az adatok elengedhetetlenek ahhoz, hogy a webshop hatékonyan tudjon kommunikálni a felhasználókkal, valamint biztosítsa számukra a személyre szabott szolgáltatásokat és értesítéseket.

A regisztráció után a felhasználók bármikor bejelentkezhetnek az általuk megadott felhasználónév és jelszó segítségével. Ezáltal hozzáférnek a webshop teljes kínálatához, és könnyen böngészhetik a termékeket.

A termékek böngészése során a felhasználóknak lehetőségük van szűrők segítségével finomítani a kínálatot az ár szerint csökkenő vagy növekvő sorrendben, valamint a legújabb termékek megjelenítésével. Ez segít abban, hogy a felhasználók könnyen megtalálják a számukra legmegfelelőbb termékeket.

Az oldal alján elhelyezett cég kapcsolat adatai lehetővé teszik a felhasználók számára, hogy bármikor felvegyék a kapcsolatot a webshop üzemeltetőivel kérdéseikkel vagy észrevételeikkel kapcsolatban. Ez növeli a felhasználók bizalmát és segíti a problémák gyors megoldását.

Amennyiben a felhasználó kiválaszt egy terméket, és megadja a kívánt mennyiséget, lehetősége van azt a kosárba helyezni. A kosárba helyezett termékeket később áttekintheti és szükség esetén módosíthatja, mielőtt véglegesíti a vásárlást.

A fizetés véglegesítésekor a felhasználóknak szokványosan meg kell adniuk a szükséges fizetési és szállítási adatokat, mint például a bankkártya adatokat és a szállítási címet. Ez biztosítja a zökkenőmentes vásárlási élményt és a biztonságos tranzakciókat.

d) Információkérés lehetőségei

Ha kérdése van a szoftverrel kapcsolatban, vagy segítségre van szüksége a használatával kapcsolatban, kérjük, vegye fel velünk a kapcsolatot. Az alábbiakban található elérhetőségeinken állunk rendelkezésére:

Telefonszám: +36209979662, +36303499114

E-mail: support@patekft.com

Kérjük, vegye figyelembe, hogy támogatási munkatársaink munkaidőben állnak rendelkezésére hétfőtől péntekig 9:00-tól 17:00-ig. Amennyiben kérdése vagy problémája merülne fel a szoftver használata során, ne habozzon felvenni velünk a kapcsolatot. Csapatunk minden erőfeszítést megtesz annak érdekében, hogy segítsen Önnek és megoldja az esetleges problémákat.

Ezenkívül, a támogatási dokumentációhoz is hozzáférhet a weboldalunkon, amely részletes útmutatást és gyakran feltett kérdéseket tartalmaz a szoftver használatával kapcsolatban. Köszönjük, hogy választotta a Webshop Software-t, és bármilyen további kérdése esetén forduljon hozzánk bizalommal.

4. Záró gondolatok és köszönetnyilvánítás

Az elmúlt hónapokban intenzív munkával és elkötelezettséggel dolgoztunk ezen projektünkön, mely egy webshop létrehozására irányult. Most, hogy eljutottunk a végére, szeretnénk összefoglalni a tapasztalatainkat és köszönetet mondani mindazoknak, akik segítettek minket ebben a folyamatban.

Először is, szeretnénk kiemelni Zoli nagyszerű munkáját és hozzájárulását a projekt sikeréhez. Kiválóan működött együttműködésünk, és a közös erőfeszítéseinknek köszönhetően sikerült hatékonyan megoldanunk a felmerülő feladatokat. Mindig megbízhatóan teljesítettük a kitűzött határidőket, és közös erővel dolgoztunk azon, hogy a legjobb eredményt érjük el.

A projekt során számos kihívással szembesültünk, de ezek mind hozzájárultak ahhoz, hogy fejlődjünk és tanuljunk. Az egyik legnagyobb kihívás az volt, hogy az üzleti igényeket és a felhasználói elvárásokat összehangoljuk a technikai megvalósítással. Ennek ellenére sikerült rugalmasan alkalmazkodnunk a változó követelményekhez, és hatékonyan kezeltük a felmerülő problémákat.

Az elkészült webshop egyfajta mérföldkő számunkra, és büszkék vagyunk az elért eredményekre. Úgy érezzük, hogy sikerült megvalósítanunk az eredeti terveket, és egy olyan terméket hoztunk létre, amely megfelel az igényeinknek és elvárásainknak.

A projekt utóélete számunkra is izgalmas lehetőségeket rejt. Továbbra is szeretnénk fejleszteni a webshopot, és új funkciókat vezetnénk be annak érdekében, hogy még jobban ki tudjuk szolgálni a vásárlóink igényeit és javítsuk a felhasználói élményt.

Végül, de nem utolsósorban, szeretnénk köszönetet mondani mindazoknak, akik segítettek minket ebben a projektben. Külön köszönet jár családtagjainknak és szeretteinknek, akik türelmesen támogattak minket ezen az úton. Köszönet illeti szüleinket és támogatóinkat is, akik anyagi segítséget nyújtottak a projektünk megvalósításához.

5. Irodalomjegyzék

HTML segítség: https://www.w3schools.com/html/default.asp

CSS segítség: https://www.w3schools.com/css/default.aspv

JavaScript segítség: https://www.w3schools.com/js/default.asp

SQL segítség: https://www.w3schools.com/sql/default.asp