Baranya Vármegyei Szakképzési Centrum

Simonyi Károly Technikum és Szakképző Iskola

Vizsgaremek

Készítették: Pál Rajmund, Futó Csaba

Pécs

2025

Baranya Vármegyei Szakképzési Centrum

Simonyi Károly Technikum és Szakképző Iskola

Szakma megnevezése: Szoftverfejlesztő és –tesztelő

A szakma azonosító száma: 5 0613 12 03

Vizsgaremek

Wimu Webshop

Készítették: Pál Rajmund, Futó Csaba

Pécs

2025

TARTALOMJEGYZÉK

1. Bevezető

1.1 Projekt célja és motiváció

A Wimu Webshop létrehozásával célunk egy egyszerű, de hatékony online vásárlási platform fejlesztése volt, ahol elsajátíthattuk a webalkalmazások teljes életciklusát. A projektet azért választottuk, mert lehetőséget adott a gyakorlati ismeretek megszerzésére, kezdve az adatbázis-tervezéstől a felhasználói felület optimalizálásáig. A kezdeti kihívások (pl. kosárkezelés, session-kezelés) után komplex funkciókat valósítottunk meg, például fizetési módok integrálását és raktárkészlet-nyilvántartást.

1.2 Tanultak és új ismeretek

A fejlesztés során elsajátítottuk:

Adatbázis-kezelés: MySQL táblák tervezése (pl. termékek, felhasználók, rendelések).

Biztonsági technikák: Jelszavak titkosítása (Hash), SQL injection elleni védelem.

Folyamatkezelés: Felhasználói regisztráció, kosár tartalmának mentése, rendelés véglegesítése.

Fizetési integráció: PayPal SDK használata bankkártyás tranzakciókhoz.

Reszponzív design: Bootstrap segítségével mobilbarát felületek kialakítása.

1.3 Későbbi tervek

Jövőbeli fejlesztési céljaink:

Email-es regisztráció: Jelszó-visszaállítás és regisztráció megerősítése.

Rendeléskövetés: Felhasználók számára áttekinthető szállítási állapot.

Admin felület: Termék- és rendeléskezelés intuitív vezérlőpultja.

Értékelési rendszer: Vásárlói visszajelzések megjelenítése.

1.4 Csapatmunka és szerepkörök

A csapatban a feladatok így oszlottak meg:

Csaba: Backend fejlesztés (PHP, adatbázis logika, rendelésfeldolgozás).

Rajmund: Frontend tervezés (HTML/CSS, Bootstrap).

A kommunikációhoz Discordot használtunk. A kódot GitHubon tároltuk.

Projektszervezési Eszközök

Discord (Kommunikáció): Feladatok szétosztása és állapotuk nyomon követése (pl. "Kosár implementálása", "Fizetési módok hozzáadása").

GitHub: Kódmegosztás és együttműködés (branch-ek használata).

Messenger: Megbeszéltük, hogy mikor csináljuk a vizsgaremeket.

1.5 Összegzés

A Wimu Webshop projekt lehetővé tette, hogy egy valós alkalmazás fejlesztésének minden szakaszában részt vegyünk. A nehézségek (pl. adatbázis-frissítések szinkronizálása) megoldása közben mélyebb betekintést kaptunk a webes rendszerek működésébe. A projekt továbbfejlesztésével célunk, hogy egy teljes körű, felhasználóbarát webshoppá nőjön ki.

2. Témaválasztás:

Miért ezt választottuk?

Én Csaba azért ezt szerettem volna választani, mert nagyon sok webshop van a világban, de én is meg akartam nézni a folyamatát, hogy hogyan épül fel egy webshop.

Én Rajmund azért ezt választottam, mert mindig is érdekelt az üzleti oldala a dolgoknak, illetve mindig akartam egy saját boltot/webshopot nyitni.

2.1 Közös cél: egy életképes prototípus létrehozása

Bár különböző motivációk vezettek minket, a cél egyértelmű volt: egy működő webshop prototípus, ami:

Technikailag megalapozott: Biztonságos, skálázható, könnyen bővíthető.

Felhasználóbarát: Intuitív navigáció, reszponzív design, gyors betöltés.

Üzletileg releváns: Valós igényeket szolgál (pl. címkezelés külföldre szállításhoz).

A projekt során megtanultuk, hogy a kommunikáció és a kompromisszumok nélkülözhetetlenek egy csapatban. Például amikor Rajmund egy összetett animációt akart a kosárhoz, de Csaba rámutatott, hogy az lelassítaná az oldalt, kompromisszumként egyszerűbb, de hatékony megoldást választottunk.

2.2 Mit hoz a jövő?

A webshop jelenleg egy **MVP** (Minimum Viable Product), de alapozóként szolgál további fejlesztésekhez:

Email-es regisztráció: Jelszó-visszaállítás és automatikus értesítések implementálása.

Analitika: Google Analytics integráció a vásárlói viselkedés nyomon követésére.

SEO-optimalizálás: A termékoldalak tartalmának finomhangolása keresőmotorok számára.

Csaba számára ez a projekt egy technikai referenciapont, Rajmundnak pedig egy üzleti alapkő. Mindketten abban reménykedünk, hogy a jövőben ezt a prototípust egy teljes értékű online vállalkozássá lehet fejleszteni.

3. Az alkalmazott fejlesztői eszközök:

3.1. Programozási és leíró nyelvek & keretrendszerek

PHP: A backend logika (pl. kosárkezelés, felhasználókezelés) megvalósítása.

MySQL: Relációs adatbázis a termékek, felhasználók és rendelések tárolására.

HTML/CSS/JavaScript: Frontend felület reszponzív design-nal és interaktív elemekkel.

Bootstrap 5: Gyors és egységes felhasználói felület kialakítása (pl. kártyák, navigációs sáv).

3.2. Fejlesztői környezetek & eszközök

Visual Studio Code: Fő IDE kódírásra, debugolásra (PHP, HTML, CSS, JS).

XAMPP: Lokális szerver környezet (Apache, MySQL, PHP) teszteléshez.

phpMyAdmin: Adatbázisok kezelése, SQL lekérdezések futtatása.

Git & GitHub: Verziókövetés és csapatmunka koordinálása.

3.3. Design & Médiaeszközök

PiktoChart: Banner szerkesztése (pl. img/banner2.png).

Leonardo.ai, MS Copilot: Termék képek

Bootstrap Icons: Ikongyűjtemény a felhasználói felülethez.

3.4. KÖNYVTÁRAK & FÜGGŐSÉGEK

jQuery: Dinamikus űrlapkezelés és AJAX kérések (pl. kosár frissítése).

Popper.js & Bootstrap JS: Interaktív elemek (dropdown menük, tooltip-ek).

3.5. EGYÉB ESZKÖZÖK

Messenger/Discord: Feladatok szétosztása és állapotkövetés.

Google Chrome DevTools: Frontend hibakeresés.

3.6. BIZTONSÁGI ESZKÖZÖK

BCrypt: Jelszavak titkosítása (password_hash() és password_verify()).

Prepared Statements: SQL injection elleni védelem (pl. mysqli->prepare()).

Session Hardening: Cookie-k biztonságos beállításai (httponly, secure flag-ek).

3.7. KOMPATIBILITÁS & TESZTELÉS

Cross-Browser Tesztelés: Chrome, Firefox, Safari.

Reszponzív Design Tesztelés: Mobil-, tablet- és asztali nézet.

4. TERVEZÉSI MÓDSZER

4.1 ALKALMAZÁS TERVEZÉSE:

A Wimu Webshop tervezése során a felhasználói igények és üzleti célok összehangolására fókuszáltunk. A tervezés három fő szakaszra oszlott:

Követelményelemzés: A vásárlók és adminisztrátorok igényeinek feltárása (pl. egyszerű regisztráció, kosárkezelés, rendeléskövetés).

Rendszerfelépítés: A háromrétegű modell (frontend, backend, adatbázis) kialakítása.

Funkcionális specifikációk:

Felhasználói szerepkörök (vendég, regisztrált felhasználó, admin).

Modulok (termékkatalógus, kosár, fizetési rendszer).

A tervezés során kiemelt szerepet kapott a reszponzív design, hogy a webshop minden eszközön (mobil, tablet, asztali) optimálisan működjön.

4.2 TERVEZÉSI MÓDSZERTAN:

A projekt agilis módszertant követett, amely lehetővé tette a folyamatos visszajelzés alapján történő iteratív fejlesztést.

Csapatmunka: Megbeszélés alapján összedolgoztunk, hogy mikor ér rá a másik kolléga.

Felhasználó tesztelés: User-ként bejelentkeztünk leteszteltünk főbb funkciókat.

4.3 VIZUÁLIS MODELLEZÉS:

A rendszer vizuális reprezentációja kulcsfontosságú volt a tervezésben:

Bootstrap-ek:

Főoldal elrendezése (termékkártyák, navigációs sáv).

Kosár oldal felépítése (táblázatos nézet, gombok).

Adatbázis séma tervezése:

ER-diagram készült a táblák és kapcsolatok szemléltetésére (pl. arucikk ↔ kategoriak, ugyfel ↔ rendelesek).

Normalizálás az ismétlődések elkerülése végett.

Folyamatábrák:

Belépési és rendelési folyamatok vizualizációja.

4.4 OOP MEGVALÓSULÁSA:

Bár a projekt főként PHP-t használ, objektumorientált elemeket is építettünk bele:

Adatbázis kezelés OOP stílusban:

A mysqli osztály használata prepared statement-ekkel:

```
$parancs = $kapcsolat->prepare("SELECT * FROM ugyfel WHERE email=?");
```

```
$parancs->bind_param("s", $email);
```

```
$parancs->execute();
```

Transaction kezelés a rendeléseknél

Osztálytervek jövőbeli bővítéshez:

User osztály: Felhasználói adatok és jogosultságok kezelése.

Cart osztály: Kosár műveletek (hozzáadás, törlés, frissítés).

Product osztály: Termékek lekérdezése és szűrése.

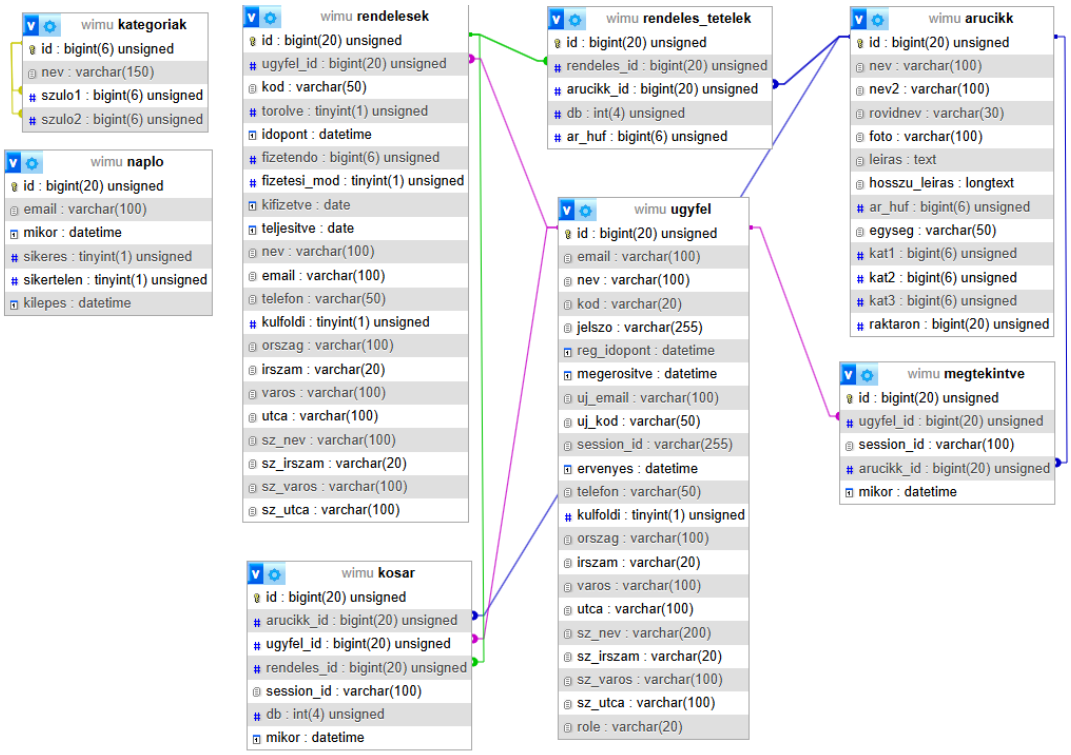
Öröklés és egységbezárás:

A Payment főosztályból származó alosztályok (pl. PayPalPayment, BankTransfer) a fizetési módok kezelésére. (fejlesztés alatt)

Privát metódusok biztonságos adatkezeléshez (pl. jelszó titkosítás).

MySql: Adatbáziskapcsolat egyszeres példányosítása.

5. ADATMODELL LEÍRÁSA:



5.1 FŐBB TÁBLÁK:

- **Termékek** (arucikk)
  - Adatok: név, ár, készlet, kategóriák (3 szint), leírások.
  - Példa: Laptopok, workstationok.
- **Kategóriák** (kategoriak)
  - Hierarchikus struktúra (pl. Elektronika → Laptopok → Gaming).

- Szülő-mezőkkel (szulo1, szulo2).

- **Felhasználók** (ugyfel)
  - Regisztráció, címek, titkosított jelszó (BCrypt), admin jogok (role).
- **Kosár** (kosar)
  - Session vagy felhasználóhoz kötött, ideiglenes adatok a rendelés előtt.
- **Rendelések** (rendelesek + rendeles_tetelek)
  - Fizetési módok, címek, egyedi kód (67e7c1d056397).
  - Tételek: termék ID, darabszám, ár.

	Tábla	Művelet	Sorok	Típus	Illesztés	Méret	Felülírás
☐	arucikk	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	4	InnoDB	latin2_hungarian_ci	48.0 KB	-
☐	kategoriak	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	72	InnoDB	latin2_hungarian_ci	64.0 KB	-
☐	kosar	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	2	InnoDB	latin2_hungarian_ci	64.0 KB	-
☐	megtekintve	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	1	InnoDB	latin2_hungarian_ci	48.0 KB	-
☐	naplo	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	0	InnoDB	latin2_hungarian_ci	32.0 KB	-
☐	rendelesek	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	3	InnoDB	latin2_hungarian_ci	128.0 KB	-
☐	rendeles_tetelek	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	2	InnoDB	latin2_hungarian_ci	48.0 KB	-
☐	ugyfel	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	2	InnoDB	latin2_hungarian_ci	80.0 KB	-
	8 tábla	Összesen	86	InnoDB	utf8mb4_general_ci	512.0 KB	0 B

5.2 Fő funkciók:

- **Többszintű kategóriák**: Szűrés és navigáció egyszerűsítése.
- **Kosárkezelés**: Session-alapú vagy felhasználói fiókhoz kötött.
- **Rendelésfeldolgozás**: Szállítási/számlázási címek, külföldi támogatás.
- **Biztonság:** Titkosított jelszavak, bejelentkezési napló (`naplo`).

5.3 Táblák kapcsolatai

Az adatbázis relációit idegen kulcsok (`Foreign Keys`) biztosítják:

Termék → Kategóriák:

```
ALTER TABLE arucikk ADD FOREIGN KEY (kat1) REFERENCES kategoriak(id);
```

Egy termék legfeljebb 3 kategóriába sorolható (kat1, kat2, kat3).

Kosár → Termékek:

```
ALTER TABLE kosar ADD FOREIGN KEY (arucikk_id) REFERENCES arucikk(id);
```

Egy kosár tétel mindig egy létező termékre hivatkozik.

Rendelések → Felhasználók:

```
ALTER TABLE rendelesek ADD FOREIGN KEY (ugyfel_id)
REFERENCES ugyfel(id);
```

Minden rendelés egy regisztrált felhasználóhoz tartozik.

5.4 Indexek és optimalizálás

A teljesítmény növelése érdekében stratégiai indexek lettek létrehozva:

Tábla	Indexelt mezők	Cél
`arucikk`	`kat1`, `kat2`, `kat3`	Gyors szűrés kategóriák szerint
`ugyfel`	`email`	Gyors belépés ellenőrzése
`rendelesek`	`kod`	Egyedi rendelésazonosító keresése
`megtekintve`	`arucikk_id`	Statisztikák generálása (pl. "Legnépszerűbb termékek")

5.5 Biztonsági intézkedések

Jelszavak titkosítása:

A ugyfel.jelszo mezőben BCrypt hashelés kerül használatra.

Példa: `$2y$10$...` – a 10 a titkosítási költség faktort jelöli.

SQL Injection védelem:

Prepared statement-ek használata PHP-ben:

```
$parancs = $kapcsolat->prepare("SELECT * FROM ugyfel WHERE
email=?");
```

```
$parancs->bind_param("s", $email);
```

Munkamenet kezelés:

A ugyfel.session_id és ervenyes mezők biztosítják a jogosulatlan hozzáférés kiszűrését.

5.6 JELENTŐS FUNKCIÓK ÉS ADATBÁZIS INTEGRÁCIÓ

Többszintű kategóriák

Megvalósítás: A kategoriak tábla hierarchikus struktúrája lehetővé teszi:

Akár 3 szintű kategória fa (pl. Elektronika → Laptopok → Gaming).

Rugalmas szűrés a terméklistákban (kat1, kat2, kat3 mezők).

Kosárkezelés

Vendég vásárlók: A session_id menti a kosarat 7 napig (`DELETE FROM kosar WHERE mikor<'$egyhete'`).

Készlet ellenőrzés: A kosar és arucikk táblák tranzakcióban frissülnek, hogy elkerüljük a túlértékesítést.

Rendeléskövetés

Státuszok: A rendelesek tábla fizetesi_mod és teljesitve mezői nyomon követik a fizetést és szállítást.

5.7 KÉSŐBBI FEJLESZTÉSEK

Elavult kosarak törlése: Heti automatikus törlés cron job segítségével.

Full-text keresés: A arucikk táblában a termékleírások gyors keresése.

Többnyelvű támogatás: Új mezők (pl. leiras_en) a lokalizációhoz.

Ez az adatbázis szerkezet biztosítja a Wimu Webshop skálázhatóságát és a jövőbeli bővítések lehetőségét.

6. RÉSZLETES FELADATSPECIFIKÁCIÓ, ALGORITMUSOK

6.1. BELEPES.PHP – BELÉPÉSI FOLYAMAT

Funkció: Felhasználó hitelesítése e-mail és jelszó alapján.

Paraméterek:

`$_POST['email']: Felhasználó e-mail címe.`

`$_POST['jelszo']: Felhasználó jelszava.`

```php
<?php
} elseif ($mit == "ellenoriz") {
$email = isset($_POST['email']) ? $_POST['email'] : "";
$jelszo = isset($_POST['jelszo']) ? $_POST['jelszo'] : "";

$most = date("Y-m-d H:i:s");

$parancs = "SELECT * from ugyfel WHERE email='$email'";
$eredmeny = mysqli_query($kapcsolat, $parancs);

if (mysqli_num_rows($eredmeny) > 0) {
  $sor = mysqli_fetch_array($eredmeny);
  if (password_verify($jelszo, $sor['jelszo'])) {
    $feloraja = date("Y-m-d H:i:s", time() - 1800);

    $sql = "SELECT count(*) as darab FROM naplo WHERE email='$email' AND mikor>='$feloraja' AND sikertelen=1";
    $rs_naplo = mysqli_query($kapcsolat, $sql);
    $naplo_sor = mysqli_fetch_array($rs_naplo);
    $darab = $naplo_sor["darab"];

    if ($darab >= 3) {
      header("Location: belepes.php?tilos=1&email=$email");
    } else {
      setcookie("webshop_email", $email, time() + 86400 * 7);
      setcookie("webshop_jelszo", $sor['jelszo'], time() + 86400 * 7);

      $id = $sor["id"];
      $kod = $sor["kod"];
      $jelszo = $sor["jelszo"];

      $ervenyes = date("Y-m-d H:i:s", time() + 3600);

      $sql = "UPDATE ugyfel SET session_id='" . session_id() . "', ervenyes='$ervenyes' WHERE id=$id";
      mysqli_query($kapcsolat, $sql);

      $sql = "insert into naplo (email, mikor, sikeres) values ('$email', '$most', 1)";
      mysqli_query($kapcsolat, $sql);

      $sql = "UPDATE kosar SET ugyfel_id=$id WHERE session_id='" . session_id() . "' AND rendeles_id=0";
      mysqli_query($kapcsolat, $sql);

      header("Location: index.php");
    }
  } else {
    $sql = "insert into naplo (email, mikor, sikertelen) values ('$email', '$most', 1)";
    mysqli_query($kapcsolat, $sql);

    header("Location: belepes.php?hiba=1&email=$email");
  }
```

Működés:

JavaScript validálja az e-mail formátumot és a jelszó meglétét.

Szerver oldalon ellenőrzi az adatbázisban a felhasználó létezését (`SELECT * FROM ugyfel WHERE email='$email'`).

Hash-elt jelszó ellenőrzése `password_verify()` függvénnyel.

Naplózza a sikertelen próbálkozásokat (napló tábla).

Ha 3 sikertelen próbálkozás van 30 percen belül, tiltja a belépést.

Sikeres belépés esetén beállítja a session-t és cookie-kat, átirányít a főoldalra.

Visszatérési érték: HTTP átirányítás hibákra (hiba=1, tilos=1) vagy sikeres belépés esetén index.php.

6.2 Főbb függvények és metódusok specifikációja

1. Kosár tartalmának betöltése (1_kosar_tartalma.php)

Funkció: A felhasználó kosarában lévő termékek megjelenítése, összegzés.

```
<tbody>
  <?php
  $osszeg = 0;
  $sql = ($belepve == 0)
    ? "SELECT * FROM kosar WHERE session_id='" . session_id() . "' AND rendeles_id=0"
    : "SELECT * FROM kosar WHERE ugyfel_id=$webshop_id AND rendeles_id=0";

  $eredmeny = mysqli_query($kapcsolat, $sql);
  while ($sor = mysqli_fetch_array($eredmeny)) {
    $kosar_id = $sor["id"];
    $arucikk_id = $sor["arucikk_id"];
    $db = $sor["db"];

    $termek = mysqli_fetch_array(mysqli_query(
      $kapcsolat,
      "SELECT * FROM arucikk WHERE id=".intval($arucikk_id)
    ));

    $osszeg += $db * $termek['ar_huf'];
  ?>
  <tr>
    <td><b><?= $termek['nev'] ?></b> <?= $termek['nev2'] ?></td>
    <td class="text-end"><?= szampontos($termek['ar_huf']) ?> HUF</td>
    <td class="text-center">
      <input name="<?= $kosar_id ?>"
             value="<?= $db ?>"
             class="form-control text-center"
             style="width: 80px;">
    </td>
    <td class="text-end"><?= szampontos($db * $termek['ar_huf']) ?> HUF</td>
  </tr>
  <?php } ?>

  <tr>
    <td colspan="3" class="text-end"><b>Összesen:</b></td>      You, 3 days ago • Javi
    <td class="text-end"><b><?= szampontos($osszeg) ?> HUF</b></td>
  </tr>
</tbody>
```

Paraméterek:

session_id: A felhasználó session azonosítója (ha nincs bejelentkezve).

ugyfel_id: A bejelentkezett felhasználó azonosítója.

Adatbázis lekérdezés:

```
SELECT * FROM kosar WHERE session_id = ? OR ugyfel_id = ?;
```

Kimenet: HTML tábla a termékekkel, mennyiséggel és összesített árakkal.

6.3 VÁSÁRLÓI ADATOK VALIDÁLÁSA (2_VASARLOI_ADATOK.PHP)

JavaScript függvény `ellenoriz()`*:*

Funkció: Ellenőrzi, hogy minden kötelező mező ki van-e töltve.

Ellenőrzött mezők: Név, telefonszám, cím, számlázási adatok.

Hibakezelés: Alert üzenetek üres mezők esetén.

```php
if ($webshop_email != "" && $webshop_jelszo != "") {
  $parancs = "SELECT * FROM ugyfel WHERE email='$webshop_email' AND jelszo='$webshop_jelszo'";
  $eredmeny = mysqli_query($kapcsolat, $parancs);
  if (mysqli_num_rows($eredmeny) > 0) {
    $sor = mysqli_fetch_array($eredmeny);
    $webshop_id = $sor["id"];
    $webshop_nev = $sor["nev"];
    $telefon = $sor["telefon"];
    $kulfoldi = $sor["kulfoldi"];
    $orszag = $sor["orszag"];
    $irszam = $sor["irszam"];
    $varos = $sor["varos"];
    $utca = $sor["utca"];
    $sz_nev = $sor["sz_nev"];
    $sz_irszam = $sor["sz_irszam"];
    $sz_varos = $sor["sz_varos"];
    $sz_utca = $sor["sz_utca"];
```

Adatbázis frissítés:

```
UPDATE ugyfel SET nev=?, telefon=?, ... WHERE id=?;
```

Paraméterek: Felhasználó által megadott adatok (pl. $_POST['nev'], $_POST['telefon']).

6.4. FIZETÉSI MÓD KIVÁLASZTÁSA (3_FIZETESI_MODOK.PHP)

JavaScript függvény `ellenoriz`():

Funkció: Ellenőrzi, hogy legalább egy fizetési mód ki van-e választva.

Integráció: PayPal SDK használata bankkártyás fizetéshez.

Konverzió: HUF → USD fix árfolyammal (pl. osszeg / 350).

Adatátvitel:

```
<input type="hidden" name="fizet" value="1|2|3">
```

Értékek: 1 (banki átutalás), 2 (utánvét), 3 (PayPal).

```php
// Initialize variables
$nev = $_POST['nev'] ?? '';
$telefon = $_POST['telefon'] ?? '';
$kulfoldi = $_POST['kulfoldi'] ?? 0;
$orszag = $_POST['orszag'] ?? 'Magyarország';
$irszam = $_POST['irszam'] ?? '';
$varos = $_POST['varos'] ?? '';
$utca = $_POST['utca'] ?? '';
$sz_nev = $_POST['sz_nev'] ?? '';
$sz_irszam = $_POST['sz_irszam'] ?? '';
$sz_varos = $_POST['sz_varos'] ?? '';
$sz_utca = $_POST['sz_utca'] ?? '';

$webshop_email = $_COOKIE['webshop_email'] ?? "";
$webshop_jelszo = $_COOKIE['webshop_jelszo'] ?? "";

$belepve = 0;
$most = date("Y-m-d H:i:s");

if ($webshop_email != "" && $webshop_jelszo != "") {
  $parancs = "SELECT * FROM ugyfel WHERE email='$webshop_email' AND jelszo='$webshop_jelszo'";
  $eredmeny = mysqli_query($kapcsolat, $parancs);
  if (mysqli_num_rows($eredmeny) > 0) {
    $sor = mysqli_fetch_array($eredmeny);
    $webshop_id = $sor["id"];
    $webshop_nev = $sor["nev"];

    $sql = "UPDATE ugyfel SET nev='$nev', telefon='$telefon', kulfoldi=$kulfoldi, orszag='$orszag', irszam='$irszam', varos='$varos', utca='$utca',
    sz_nev='$sz_nev', sz_irszam='$sz_irszam', sz_varos='$sz_varos', sz_utca='$sz_utca' WHERE id=$webshop_id";
    mysqli_query($kapcsolat, $sql);

    $belepve = 1;
  }
}
```

6.5. RENDELÉS ÖSSZEGZÉSE (4_RENDELES_OSSZEGZES.PHP)

Funkció: Megjeleníti a felhasználó kosarát és a kiválasztott fizetési módot.

Adatbázis lekérdezés:

```
SELECT arucikk.nev, kosar.db, arucikk.ar_huf

FROM kosar

JOIN arucikk ON kosar.arucikk_id = arucikk.id

WHERE ugyfel_id = ?;
```

```
    <tbody>
        <?php
        $osszeg = 0;
        $sql = "SELECT k.*, a.nev, a.ar_huf
                FROM kosar k
                JOIN arucikk a ON k.arucikk_id = a.id
                WHERE k.ugyfel_id = ? AND k.rendeles_id = 0";
        $parancs = $kapcsolat->prepare($sql);
        $parancs->bind_param("i", $webshop_id);
        $parancs->execute();
        $eredmeny = $parancs->get_result();

        while ($sor = $eredmeny->fetch_assoc()):
            $osszeg += $sor['db'] * $sor['ar_huf'];
        ?>
        <tr>
            <td><?= htmlspecialchars($sor['nev']) ?></td>
            <td class="text-center"><?= $sor['db'] ?></td>
            <td class="text-end"><?= szampontos($sor['ar_huf']) ?> HUF</td>
            <td class="text-end"><?= szampontos($sor['db'] * $sor['ar_huf']) ?> HUF</td>
        </tr>
        <?php endwhile; ?>

        <tr>
            <td colspan="3" class="text-end fw-bold">Összesen:</td>
            <td class="text-end fw-bold"><?= szampontos($osszeg) ?> HUF</td>
        </tr>
    </tbody>
</table>
```

Kimenet: Tábla a termékek részleteivel és egy véglegesítő gomb.

6.6 RENDELÉS FELDOLGOZÁSA (5_RENDELES_ELKULD.PHP)

```
if (!mysqli_query($kapcsolat, $sql)) {
    die("Hiba a rendelés rögzítésénél: " . mysqli_error($kapcsolat));
}

// Rendelés ID lekérése
$rendeles_id = mysqli_insert_id($kapcsolat);

// 2. Termékek mentése a rendeles_tetelek táblába
$sql_kosar = "SELECT arucikk_id, db FROM kosar WHERE ugyfel_id = ? AND rendeles_id = 0";
$parancs = $kapcsolat->prepare($sql_kosar);
$parancs->bind_param("i", $webshop_id);
$parancs->execute();
$kosar_tetelek = $parancs->get_result();

while ($tetel = $kosar_tetelek->fetch_assoc()) {
    $arucikk_id = $tetel['arucikk_id'];
    $db = $tetel['db'];

    // Egységár lekérése
    $sql_ar = "SELECT ar_huf FROM arucikk WHERE id = ?";
    $parancs_ar = $kapcsolat->prepare($sql_ar);
    $parancs_ar->bind_param("i", $arucikk_id);
    $parancs_ar->execute();
    $ar = $parancs_ar->get_result()->fetch_assoc()['ar_huf'];

    // Beszúrás a rendeles_tetelek táblába
    $sql_insert = "INSERT INTO rendeles_tetelek (rendeles_id, arucikk_id, db, ar_huf)
                   VALUES (?, ?, ?, ?)";
    $parancs_insert = $kapcsolat->prepare($sql_insert);
    $parancs_insert->bind_param("iiii", $rendeles_id, $arucikk_id, $db, $ar);
    $parancs_insert->execute();
}

// 3. Kosár tételek frissítése
$sql = "UPDATE kosar SET rendeles_id = ".intval($rendeles_id)." WHERE ugyfel_id = ".intval($webshop_id)." AND rendeles_id = 0";
if (!mysqli_query($kapcsolat, $sql)) {
    die("Hiba a kosár frissítésénél: " . mysqli_error($kapcsolat));
}
```

Lépések:

Rendelés rögzítése:

```
INSERT INTO rendelesek (ugyfel_id, kod, fizetendo, ...);
```

Termékek mentése:

```
INSERT INTO rendeles_tetelek (rendeles_id, arucikk_id, db,
ar_huf);
```

Raktárkészlet frissítése:

```
UPDATE arucikk SET raktaron = raktaron - ? WHERE id = ?;
```

Kosár ürítése:

```
DELETE FROM kosar WHERE ugyfel_id = ?;
```

Kimenet: Sikeres rendelés üzenet és visszaigazoló e-mail szimuláció.

6.7. BEJELENTKEZÉSI RENDSZER (BELEPES.PHP)

Funkció: Felhasználó hitelesítése és session kezelése.

Ellenőrzések:

Jelszó ellenőrzés: password_verify($jelszo, $sor['jelszo']).

Brute force védelem:

```
SELECT COUNT(*) FROM naplo WHERE email=? AND mikor > ? AND
sikertelen=1;
```

Blokkolás, ha 3+ sikertelen próbálkozás van 30 percen belül.

Cookie-k: webshop_email és webshop_jelszo 7 napig érvényesek.

6.8 ALGORITMUSOK LEÍRÁSA

1. Kosár frissítési algoritmus

Cél: A kosárban lévő termékek mennyiségének módosítása.

Lépések:

A felhasználó módosítja a mennyiséget a kosár oldalon.

Űrlap elküldése a `mit=modosit` paraméterrel.

Adatbázis frissítés:

```
foreach ($_POST as $kosar_id => $db) {
  UPDATE kosar SET db = ? WHERE id = ?;
}
```

Összeg újraszámolása és átirányítás.

6.9 RENDELÉSFELDOLGOZÁS FOLYAMATA

Struktogram:

[START]

↓

[Adatbázis kapcsolat]

↓

[Rendelés létrehozása → rendelesek tábla]

↓

[Termékek másolása → rendeles_tetelek tábla]

↓

[Raktárkészlet csökkentése]

↓

[Kosár ürítése]

↓

[Sikeres oldal megjelenítése]

[END]

6.10. TERMÉKKERESŐ ALGORITMUS (INDEX.PHP)

Megjegyzések:

Biztonsági hiányosságok: SQL injection kockázat néhány lekérdezésben (pl. 5_rendeles_elkuld.php).

Jelszavak tárolása nyers formában helyett password_hash() használata ajánlott.

Optimalizálási lehetőségek:

Tranzakciók használata rendelésfeldolgozásnál (ACID tulajdonságok biztosítása).

Cache-elés a gyakran lekérdezett termékadatokhoz.

6.11. KOSÁR MEGTEKINTÉSE (KOSAR_MEGTEKINTES.PHP)

Funkció: A felhasználó kosarában lévő termékek megjelenítése, törlési műveletek kezelése, összegzés.

Paraméterek:

`$_POST['torles']`: "egy" vagy "osszes" érték (egy elem vagy teljes kosár törlése).

`$_POST['arucikk_id']`: Törlendő termék azonosítója.

`$_POST['db']`: Törlendő mennyiség (csak egy elem törlése esetén).

Működés:

Session/Cookie alapján felhasználó azonosítása:

Ha bejelentkezett (ugyfel_id), különben session_id használata.

Kosár tartalmának lekérdezése:

```
SELECT * FROM kosar
WHERE (ugyfel_id = ? OR session_id = ?) AND rendeles_id = 0;
```

Raktár visszaállítás minden tételre:

```
UPDATE arucikk SET raktaron = raktaron + ? WHERE id = ?;
```

Kosár ürítése:

```
DELETE FROM kosar WHERE (ugyfel_id = ? OR session_id = ?);
```

Raktár visszaállítás:

```
UPDATE arucikk SET raktaron = raktaron + ? WHERE id = ?;
```

Tétel eltávolítása:

```
DELETE FROM kosar WHERE (ugyfel_id = ? OR session_id = ?)
AND arucikk_id = ? LIMIT 1;
```

Kimenet: HTML tábla a kosár tartalmával és összesített árral.

6.12. TERMÉK KOSÁRBA TÉTELE (KOSARBA_TESZ.PHP)

Funkció: Termék hozzáadása a kosárhoz vagy mennyiség frissítése.

Paraméterek:

`$_POST['arucikk_id']`: Termék azonosítója.

`$_POST['db']`: Hozzáadandó mennyiség.

Készlet ellenőrzése:

```
SELECT raktaron FROM arucikk WHERE id = ?;
```

Ha kert_mennyiseg > raktaron: hibaüzenet.

Felhasználó azonosítása: Bejelentkezett (`ugyfel_id`) vagy vendég (`session_id`).

Ha a termék már szerepel a kosárban:

```
UPDATE kosar SET db = db + ? WHERE (ugyfel_id = ? OR
session_id = ?) AND arucikk_id = ?;
```

Új termék esetén:

```
INSERT INTO kosar (arucikk_id, ugyfel_id/session_id, db)
VALUES (?, ?, ?);
```

Raktár frissítése:

```
UPDATE arucikk SET raktaron = raktaron - ? WHERE id = ?;
```

Visszatérés: JSON válasz (success: true vagy hibaüzenet).

6.13. PROFIL MÓDOSÍTÁS (PROFIL_MODOSIT.PHP ÉS PROFIL_MODOSIT_2.PHP)

Funkció: Felhasználói adatok és jelszó módosítása.

Paraméterek:

`$_POST['nev']`: Új név.

`$_POST['email']`: Új email.

`$_POST['uj_jelszo']` és `$_POST['uj_jelszo_2']`: Új jelszó és megerősítés.

JavaScript:

Üres mezők ellenőrzése.

Email formátum ellenőrzése `(helyescim()).`

Jelszó egyezés ellenőrzése.

Szerver oldali ellenőrzés:

Email egyediség:

```
SELECT id FROM ugyfel WHERE email = ? AND id != ?;
```

Jelszó hash-elés: `password_hash().`

Adatbázis művelet:

```
UPDATE ugyfel SET nev = ?, email = ?, jelszo = ? WHERE id =
?;
```

Kimenet:

Sikeres frissítés: `$_SESSION['siker']` üzenet.

Hiba: `$_SESSION['hiba']` üzenet.

6.14. REGISZTRÁCIÓ (REG.PHP ÉS REG_ELLENORIZ.PHP)

Funkció: Új felhasználó regisztrálása.

Paraméterek:

`$_POST['emailcim']`: Felhasználó email címe.

`$_POST['nev']`: Teljes név.

`$_POST['jelszo']` és `$_POST['jelszo2']`: Jelszó és megerősítés.

Email formátum, jelszó egyezés, üres mezők.

Jelszó komplexitás (8 karakter, nagybetű, szám):

```
preg_match('/^(?=.*[A-Z])(?=.*\d).{8,}$/', $jelszo);
```

Email egyediség:

```
SELECT * FROM ugyfel WHERE email = ?;
```

Adatbázis művelet:

```
INSERT INTO ugyfel (email, nev, jelszo, reg_idopont) VALUES
(?, ?, ?, ?);
```

Kimenet:

Sikeres regisztráció: Átirányítás bejelentkezésre.

Hiba: Üzenet a hibáról (pl. "Email már foglalt").

6.15. TERMÉK LISTÁZÁS ÉS KERESÉS (INDEX.PHP)

Funkció: Termékek szűrése kategóriák és keresési feltételek alapján.

Paraméterek:

`$_POST['kat1'], $_POST['kat2'], $_POST['kat3']: Kategóriák.`

`$_POST['mitkeres']`: Keresett kulcsszó.

`$_POST['sorrend']`: Rendezési szempont (ár, név, népszerűség).

Kategóriák hierarchikus szűrése:

Ha `kat3 > 0`: `kat1`, `kat2`, és `kat3` alapján.

Ha `kat2 > 0`: `kat1` és `kat2` alapján.

Alapértelmezett: `kat1` vagy összes termék.

Keresés a termék nevében, leírásában:

```
SELECT * FROM arucikk
WHERE (nev LIKE ? OR leiras LIKE ?)
AND katszuro_feltetel
ORDER BY sorrend;
```

Paginációl:

oldal és laponkent paraméterek kezelése.

Kimenet: Termékek megjelenítése kártyákon, keresési eredmények frissítése.

6.16. ALGORITMUSOK STRUKTOGRAMOKKAL

1. Regisztrációs Folyamat:

[START]

↓

[Űrlap betöltése (reg.php)]

↓

[Adatok elküldése → reg_ellenoriz.php]

↓

[Validáció: Email, jelszó]

↓

IF (Hiba) → Hibaüzenet

ELSE

↓

[Adatbázisba beszúrás]

↓

[Sikeres regisztráció üzenet]

[END]

2. Profil Módosítás:

[START]

↓

[Adatok bekérése (profil_modosit.php)]

↓

[Validáció: Email, jelszó]

↓

IF (Hiba) → Hibaüzenet

ELSE

↓

[Adatbázis frissítése]

↓

[Sikeres üzenet]

[END]

3. Kosárba Tétel:

[START]

↓

[Termék kiválasztása]

↓

[Készlet ellenőrzése]

↓

IF (Nincs elég készlet) → Hiba

ELSE

↓

[Kosár frissítése (INSERT/UPDATE)]

↓

[Raktárkészlet csökkentése]

↓

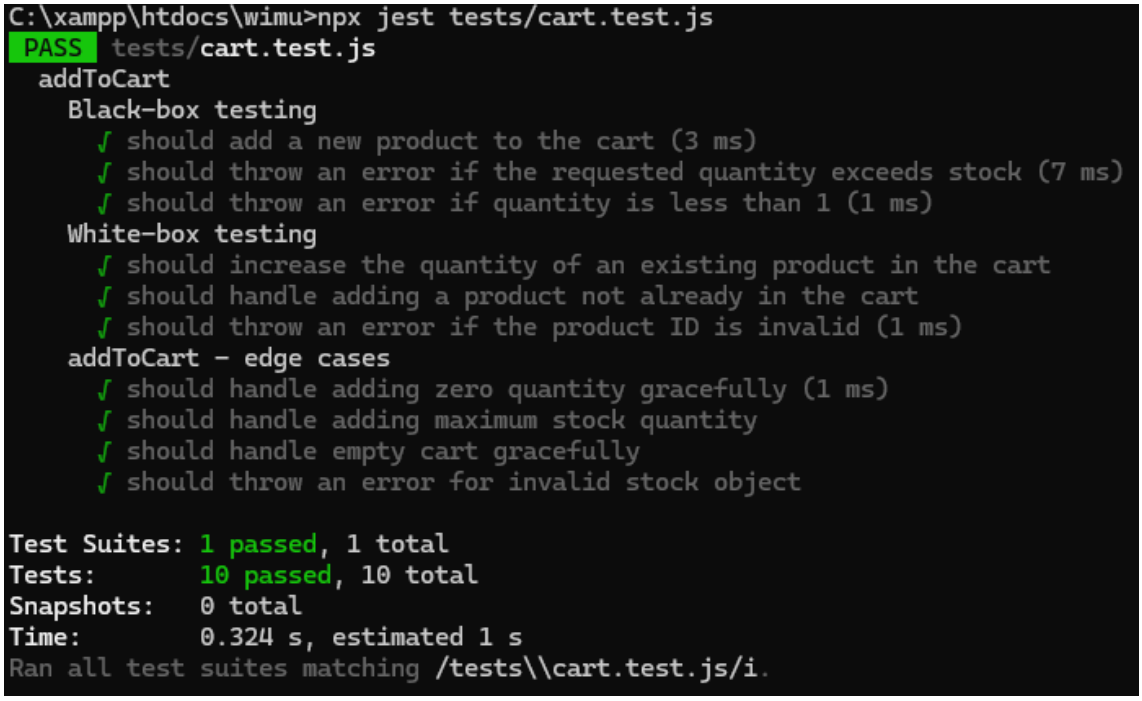
[Sikeres válasz] [END]

7. UNIT TESZT

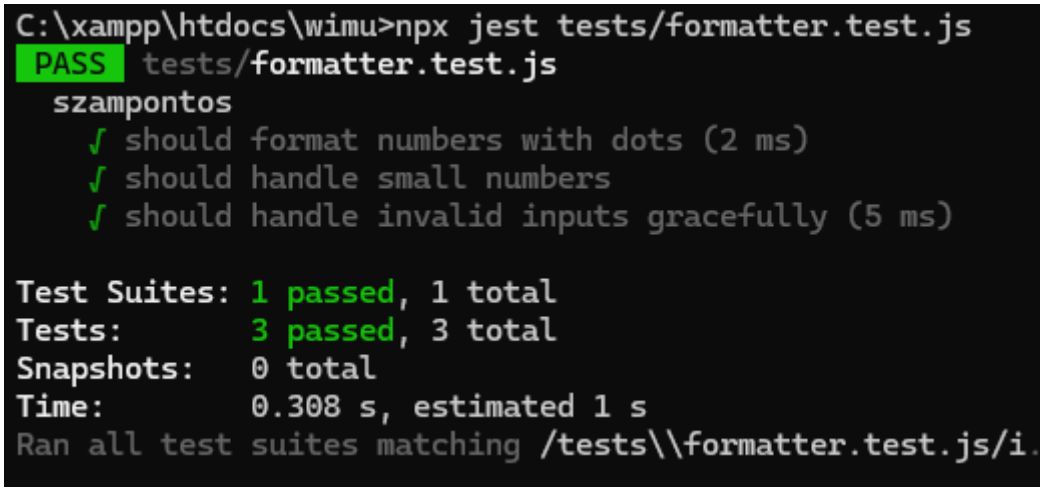
7.1. BEVEZETÉS

A dokumentáció célja, hogy bemutassa a különböző tesztelési módszereket, szinteket és technikákat egy webalkalmazás tesztelési folyamatában. A dokumentáció a következő modulokra fókuszál:

Kosárkezelés `(cart.js)`



Számformázás `(formatter.js)`



Bejelentkezés `(login.js)`

```
C:\xampp\htdocs\wimu>npx jest tests/login.test.js
 PASS  tests/login.test.js
  login
    √ should login successfully with valid credentials (2 ms)
    √ should throw an error if email is missing (5 ms)
    √ should throw an error if password is missing (1 ms)
    √ should throw an error if user is not found
    √ should throw an error if password is incorrect

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.324 s, estimated 1 s
Ran all test suites matching /tests\\login.test.js/i.
```

Rendeléskezelés `(order.js)`

```
C:\xampp\htdocs\wimu>npx jest tests/order.test.js
 PASS  tests/order.test.js
  createOrder
    √ should create a new order successfully (3 ms)
    √ should throw an error if customerId is missing (6 ms)
    √ should throw an error if total is missing (1 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.314 s, estimated 1 s
Ran all test suites matching /tests\\order.test.js/i.
```

Regisztráció (`register.js`)

```
C:\xampp\htdocs\wimu>npx jest tests/register.test.js
 PASS  tests/register.test.js
  register
    ✓ should register successfully with valid inputs (3 ms)
    ✓ should throw an error if any field is missing (7 ms)
    ✓ should throw an error for invalid email format (1 ms)
    ✓ should throw an error if email is already registered (1 ms)
    ✓ should throw an error for weak passwords (1 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.32 s, estimated 1 s
Ran all test suites matching /tests\\register.test.js/i.
```

Felhasználómódosítás (`userModification.js`)

```
C:\xampp\htdocs\wimu>npx jest tests/userModification.test.js
 PASS  tests/userModification.test.js
  modifyUser
    ✓ should modify user details successfully (3 ms)
    ✓ should throw an error if user ID is not found (6 ms)
    ✓ should throw an error if no updates are provided (1 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.313 s, estimated 1 s
Ran all test suites matching /tests\\userModification.test.js/i.
```

Teljes teszt futtatása:

```
C:\xampp\htdocs\wimu>npx jest tests
 PASS  tests/cart.test.js
 PASS  tests/login.test.js
 PASS  tests/register.test.js
 PASS  tests/userModification.test.js
 PASS  tests/order.test.js
 PASS  tests/formatter.test.js

Test Suites: 6 passed, 6 total
Tests:       29 passed, 29 total
Snapshots:   0 total
Time:        0.621 s, estimated 1 s
Ran all test suites matching /tests/i.
```

7.2. TESZTELÉSI MÓDSZEREK

Fekete Doboz Tesztelés

A bemenet-kimenet viselkedést vizsgálja, anélkül, hogy a belső implementációt ismerné.

Példák:

Kosárkezelés `(cart.test.js):`

`should add a new product to the cart`: Ellenőrzi, hogy új termék hozzáadása esetén a kosár tartalmazza-e az új elemet.

`should throw an error if quantity is less than 1`: Ellenőrzi, hogy a rendszer hibát dob-e, ha a mennyiség érvénytelen.

Bejelentkezés `(login.test.js):`

`should login successfully with valid credentials`: Ellenőrzi, hogy érvényes hitelesítő adatok esetén sikeres bejelentkezés történik-e.

7.3 FEHÉR DOBOZ TESZTELÉS

A belső logikára és struktúrára fókuszál, például feltételes ágak, ciklusok vagy hibakezelés tesztelésére.

Példák:

Kosárkezelés `(cart.test.js):`

`should increase the quantity of an existing product in the cart`: Ellenőrzi, hogy a rendszer frissíti-e a mennyiséget, ha a termék már szerepel a kosárban.

`should throw an error if the product ID is invalid`: Ellenőrzi a hibakezelést érvénytelen termékazonosító esetén.

Számformázás (formatter.test.js):

`should handle negative numbers`: Ellenőrzi, hogy a függvény helyesen formáz-e negatív számokat.

7.4. TESZTELÉSI SZINTEK

Unit Tesztelés

Egyedi függvények vagy modulok tesztelése elszigetelten.

Példák:

Számformázás (`formatter.test.js`):

`should format numbers with dots`: Ellenőrzi a szampontos függvényt különböző bemenetekkel (pl. 1000 → 1.000).

Regisztráció (`register.test.js`):

`should throw an error for weak passwords`: Ellenőrzi a jelszóerősség követelményeit.

7.5 INTEGRÁCIÓS TESZTELÉS

Modulok közötti interakciók tesztelése (jelenlegi példák főleg unit tesztek, de példa integrációs tesztre):

Rendelés létrehozása (`order.test.js`):

`should create a new order successfully`: Ellenőrzi, hogy a rendelés létrehozása frissíti-e a rendelések listáját.

7.6 RENDSZERTESZTELÉS

A teljes alkalmazás végpontokon keresztül történő tesztelése (nincs benne a példákban, de példa lehetne):

Bejelentkezés + Kosárkezelés:

Egy felhasználó bejelentkezik, hozzáad egy terméket a kosarához, majd létrehoz egy rendelést.

7.7. TESZTELÉSI TECHNIKÁK

Határérték-elemzés

Kosárkezelés `(cart.test.js)`:

`should handle adding maximum stock quantity`: Ellenőrzi, hogy a rendszer engedélyezi-e a maximális raktárkészlet hozzáadását (pl. 10 darab).

7.8 EKVIVALENCIA PARTICIONÁLÁS

Bejelentkezés `(login.test.js)`:

Érvényes partíció: test@example.com + password123 → Sikeres bejelentkezés.

Érvénytelen partíció: unknown@example.com + bármilyen jelszó → "`User not found`" hiba.

7.9 ÁLLAPOTGÉP-ALAPÚ TESZTELÉS

Felhasználómódosítás `(userModification.test.js)`:

`should modify user details successfully`: Ellenőrzi, hogy a felhasználói adatok frissülnek-e a megfelelő állapotváltozással.

7.10. TESZTKÖRNYEZET ÉS EREDMÉNYEK

Keretrendszer: Jest (JavaScript tesztelési keretrendszer).

Tesztlefedettség: A unit tesztek lefedik a kritikus funkciókat (pl. hibakezelés, formázás).

Talált Hibák:

A `szampontos` függvény nem kezeli a tizedes számokat konzisztensen (pl. 1234.56 → 1.234.56, ami nem szabványos).

A `modifyUser` függvény nem ellenőrzi az érvénytelen mezőket (pl. `role: 'invalid_role'`).

7.11. Összegzés és javaslatok

Erősségek: A tesztek átfogóan lefedik az alapvető funkciókat és hibakezeléseket.

Fejlesztési lehetőségek:

Bővíteni kell az integrációs teszteket (pl. kosár + rendelés létrehozása).

Implementálni kell tizedes számok kezelését a `szampontos` függvényben.

Következő lépések: Tesztelési automatizáció bevezetése CI/CD folyamatokba.

8. Fejlesztés

8.1 Tényleges rendelés

A rendelés még fejlesztés alatt van, még nincsen beépítve, hogy ténylegesen lehessen rendelni.

8.2 Számla kiállítás

A számlát gyorsan be lehet integrálni a webshopba. Csak még fejlesztés alatt van.

8.3 Email cím megerősítés

Egy levelet küld a felhasználónak, aki megadta az email címét, majd ott meg kell erősítenie az ügyfélnek. Így működne a jelszó/email cím változtatás az oldalon, hogy meg kell erősítenie.

8.4 Több termék feltöltés a webshopba

Több kategóriát és termékeket szeretnénk feltölteni az oldalra, hogy még több rendelés legyen.

8.5 Szállítási cím változtatás

Most úgy van, hogy ha rendel az ügyfél, akkor megadja a szállítási és számlázási adatokat. Majd szeretnénk ezen változtatni úgy, hogy többféle szállítási címeket tudjon megadni a felhasználó és módosítani azt.

8.6 Admin felület fejlesztése

Az admin felületnek a feljesztése. Mint például, hogy a rendeléseket a felületen lehessen elfogadni vagy elutasítani. A felhasználók adatainak módosítása, törlése.

9. Összegzés

9.1 Szakmai fejlődés

Webfejlesztési Technológiák:

PHP & MySQL: A backend logika (pl. felhasználókezelés, kosár rendszer) megvalósítása jelentős gyakorlatot adott a PHP és MySQL integrációban, SQL lekérdezések optimalizálásában.

Biztonsági Gyakorlatok: SQL injection elleni védelem (prepared statements), jelszókezelés (password_hash és password_verify), session kezelés.

Komplex Rendszertervezés: A webshop moduljainak (regisztráció, kosár, rendelés) összehangolt működésének kialakítása.

Frontend Fejlesztés:

Bootstrap & JavaScript: Reszponzív felületek kialakítása, dinamikus elemek (pl. kosár frissítése AJAX-szal).

Felhasználói Élmény: Űrlapvalidációk JavaScript-ben, hibakezelés és visszajelzések tervezése.

Tesztelés:

Fekete Doboz & Unit Tesztek: A kritikus funkciók (pl. kosárba helyezés, jelszóváltoztatás) tesztelése, hibák proaktív azonosítása.

9.2 JÖVŐBELI CÉLOK

Rendszerbővítés:

Új Fizetési Módok: Stripe vagy bankkártya direkt integráció.

Ajánlórendszer: Felhasználói viselkedés alapján termékajánlók.

Optimalizálás:

Teljesítmény: Adatbázis indexelés, gyorsítótárazás gyakran lekérdezett adatokhoz.

Felhasználói Élmény: Single Page Application (SPA) átállás React vagy Vue.js segítségével.

Tesztelés Fejlesztése:

Integrációs Tesztek: A rendszer egységeinek összehangolt működésének ellenőrzése.

Automatizált Tesztelés: CI/CD folyamatok bevezetése (pl. GitHub Actions).

Biztonság Erősítése:

Two-Factor Authentication (2FA): Bejelentkezési folyamat bővítése.

Adatbiztonság: Rendszeres biztonsági auditok és sebezhetőségek szkennelése.