1. Áttekintés

A WebShop alkalmazás egy egyszerű internetes kereskedelmi platform mintapéldája, amely lehetővé teszi:

Felhasználók számára a regisztrációt, bejelentkezést és profiladatok (számlázási, szállítási cím) kezelését.

Termékek böngészését, kosárba helyezését, majd megrendelését.

Admin vagy kibővített jogosultságú felhasználók számára a termékek létrehozását, szerkesztését és törlését.

A rendszer Java Spring Boot keretrendszeren alapuló REST API-t és egy React alapú frontend klienst tartalmaz. Az adatok tárolására (példában) H2 in-memory adatbázist használunk, de bármely SQL adatbázissal is helyettesíthető (pl. MySQL, PostgreSQL).

2. Funkcionális követelmények

Az alábbiakban felsoroljuk a rendszer főbb funkcionális követelményeit.

2.1. Felhasználói fiókok kezelése

Regisztráció

A rendszernek lehetővé kell tennie az új felhasználók regisztrációját (felhasználónév, jelszó, e-mail, számlázási adatok, szállítási adatok).

A jelszavakat biztonságos módon (titkosítva, pl. BCrypt) kell tárolni.

Bejelentkezés

Felhasználónév és jelszó megadásával.

Sikeres azonosítás után a rendszerben a felhasználó hitelesített státuszba kerül.

Felhasználói adatok módosítása

Bejelentkezés után a felhasználó módosíthassa saját e-mail címét, számlázási és szállítási adatait.

Jogosultságok

Alapértelmezett szerepkör: USER. Bizonyos funkciókhoz (pl. termék hozzáadás, törlés) ADMIN jogkör szükséges.

Szükség esetén további szerepkörök is definiálhatók.

2.2. Termékek kezelése

Listázás

A rendszer API végpontján (pl. GET /api/products) a termékeknek lekérdezhetőeknek kell lenniük.

Lehetőség van lapozásra (page, size paraméterek), illetve szűrésre (min/max ár).

Részletek megtekintése

Egy termék adatainak lekérdezése (pl. GET /api/products/{id}).

Kategorizálás

Minden termék rendelhető egy adott kategóriához (pl. AMD, INTEL).

A kategóriák lekérdezhetők (opcionális API végpont).

CRUD műveletek (Create, Read, Update, Delete)

Hozzáadás: POST /api/products (ADMIN vagy speciális jogosultság).

Módosítás: PUT /api/products/{id} (ADMIN).

Törlés: DELETE /api/products/{id} (ADMIN).

2.3. Kosár funkció

Termék hozzáadása a kosárhoz

Bejelentkezés nélkül is lehetséges lehet (frontend oldali session/localStorage), de éles esetben javasolt felhasználóhoz kötni.

Mennyiség módosítása

A felhasználó a kosárban tudja módosítani a mennyiséget.

Termék eltávolítása

Külön gomb vagy funkció a kosárban lévő tétel törlésére.

Kosár kiürítése

Egy lépéssel törölni lehet az összes kosár-tételt.

2.4. Rendelés leadása (Checkout)

Rendelés létrehozása

A kosár tartalmából hoz létre a rendszer egy Order entitást, amely tartalmazza a rendelés tételeket (OrderItem) és a felhasználó adatait.

Fizetési mód

A felhasználó kiválasztja: COD, BANK\_TRANSFER, CREDIT\_CARD, PAYPAL, stb.

Ha a felhasználó tiltott a COD (pl. korábbi visszautasítás miatt), akkor ne engedje a rendszer.

Szállítási mód

Választható: GLS, MPL, Foxpost, Packeta, Sameday stb.

Szállítási költség kiszámítása

Súlyalapú sávok: pl. 0-2 kg, 2-10 kg, 10+ kg.

A rendszer automatikusan kalkulálja az összeget az összes termék súlya alapján.

Rendelés státuszkezelés

Új, Kiszállításra vár, Teljesítve, stb.

Az adminisztrátor változtathatja (pl. PUT /api/orders/{orderId}/status?status=...).

Elállási jog, visszaküldés

A felhasználó élhet a 8 munkanapos elállási joggal (pl. POST /api/orders/{orderId}/return).

2.5. Csomag visszautasítása (COD esetén)

COD visszautasítás

Ha a futárnál a felhasználó nem veszi át, a rendszer jelezheti (pl. admin felületen): POST /api/orders/{orderId}/cod-refuse.

Ekkor a felhasználó onlyPrepaid mezője true lesz, és jövőben nem választhat COD-ot.

2.6. Kapcsolattartás / Ügyfélszolgálat

E-mail küldés

Rendszer e-mail küldése (regisztráció megerősítés, rendelés visszaigazolás).

Contact form (opcionális)

A felhasználó kitölt egy űrlapot, a rendszer e-mailt generál az ügyfélszolgálatnak.

Közösségi linkek

Facebook, Twitter, Instagram linkek a frontend oldalon.

3. Nem funkcionális követelmények

Az alábbi szempontok meghatározzák a rendszer minőségi elvárásait és működési környezetét.

3.1. Teljesítmény

Válaszidő

A legtöbb API-hívás (termékek listázása, kosár műveletek) ~500 ms alatt kiszolgálandó átlagos terhelés mellett.

Terhelhetőség

A rendszer képes legyen egyszerre több tucat vagy száz felhasználó kiszolgálására anélkül, hogy a válaszidő jelentősen romlana.

Skálázhatóság

Alkalmazás horizontális skálázása (például Docker/Kubernetes) esetén az alkalmazás példányai képesek legyenek párhuzamosan működni.

3.2. Biztonság

Hitelesítés és jogosultságkezelés

Spring Security-alapú védelmi réteg, a jelszavak titkos tárolása (BCrypt), biztonságos bejelentkezés.

Admin funkciók csak megfelelő (ROLE\_ADMIN) jogosultsággal érhetőek el.

Adatvédelem

Lehetőség szerint SSL/TLS kapcsolat használata (HTTPS).

GDPR megfelelés (adatkezelési tájékoztató, törlési kérelem kezelése).

Támadások elleni védelem

Basic védelmek: CSRF kikapcsolása/konfigurálása, SQL injection elleni védelem, XSS elleni védelem (React és paraméterezett lekérdezések).

Ha szükséges, WAF, tűzfal, IP-korlátozás, captcha integrálása.

3.3. Megbízhatóság és rendelkezésre állás

Adatkonzisztencia

Rendelés létrehozásakor tranzakciók alkalmazása, hogy a rész-feladatok egyidőben kerüljenek mentésre.

Hiba- és kivételkezelés

A rendszer megfelelő hibakódokat (HTTP 4xx, 5xx) küld a kliensnek.

Megfelelő naplózás (logolás) kritikus eseményekről.

3.4. Karbantarthatóság és bővíthetőség

Moduláris felépítés

Projektstruktúra (controller, service, repository, entity), frontenden komponens alapú React felépítés.

Dokumentáltság

Kódon belüli kommentek, README, Swagger/OpenAPI dokumentáció a REST végpontokról.

Könnyű integrálhatóság

Külső fizetési szolgáltatók (pl. Stripe, PayPal) moduláris illesztése.

3.5. Üzemeltethetőség

Konfigurálhatóság

Rendszer paraméterezhetősége (pl. adatbázis címe, SMTP beállítások).

Naplózás

Megfelelő log szintek: DEBUG, INFO, WARN, ERROR.

Logok hosszú távú tárolása (pl. ELK stack), elemzése.

Konténerizáció

Dockerfile és Docker Compose a telepítés egyszerűsítésére.

3.6. Felhasználói felület és felhasználói élmény

Reszponzív UI

A React frontend mobil, tablet, desktop környezetben is használhatóan jelenjen meg.

Egyszerű, intuitív kezelőfelület

Átlátható navigáció (pl. menüsor, kosár elérés, fiókadatok).

Magyar nyelvi támogatás

Gombok és feliratok magyar nyelven, a felhasználó számára érthetően.

4. Rendszer-architektúra rövid leírása

Háromrétegű felépítés

Frontend: React-alapú SPA, mely HTTP hívásokat végez a REST API-hoz.

Backend: Spring Boot alkalmazás (Controller, Service, Repository rétegek).

Adatbázis: H2 in-memory (fejlesztéshez/demóhoz), éles környezetben MySQL/PostgreSQL javasolt.

REST API

Végpontok (példák):

/api/auth/register, /api/auth/login

/api/products, /api/orders

stb.

Adattáblák

users, orders, order\_items, products, categories, stb.

Kapcsolatok (one-to-many, many-to-one) JPA/Hibernate segítségével.

Biztonsági réteg

Spring Security, jelszó kódolás (BCrypt), jogosultságkezelés.

5. Egyéb megjegyzések, javaslatok

Bővíthetőség

A kód könnyen módosítható, pl. új fizetési módok, új riportok, admin felület.

Teljesítményteszt

JMeter, Locust, stb. eszközök használata nagyobb terhelések tesztelésére.

Testkörnyezet

Integrációs és unit tesztek (JUnit, Mockito) alkalmazása a stabilitásért.

Adatkezelési szabályzat

GDPR/NIS2 előírásoknak megfelelő adat- és logkezelés, cookie tájékoztató, stb.

6. Összefoglalás

A fenti funkcionális és nem funkcionális követelmények alapján a WebShop alkalmazásnak:

Biztosítania kell az alap e-kereskedelmi folyamatokat (terméklista, kosár, rendelés, fizetés, szállítás).

Megfelelnie a biztonsági, megfelelőségi (GDPR), teljesítménybeli és felhasználói élmény elvárásoknak.

Olyan architektúrával rendelkeznie, amely könnyen karbantartható, skálázható és bővíthető.

A dokumentáció tovább bővíthető részletes use-case diagramokkal, adattábla-struktúra leírásokkal, pontos API specifikációval (OpenAPI/Swagger), valamint részletes telepítési útmutatóval.