Projektmunka dokumentáció

Barber Kereső

Premontrei Szakgimnázium és Technikum Keszthely

A projekt munkát ketten valósítottuk meg: Temleitner Marcell és Besze Marcell.

A mi általunk megálmodott alkalmazás neve Barber kereső.

Azért erre a témakörre esett a választásunk mivel csapatunk egyik tagjának nagy betekintése van a barber világában.Hosszas piackutás és felhasználói igények feltérképezése alapján realizáltuk, hogy a magyar piacon hiányzik egy olyan webalkalmazás ahol összegzi a magyar fodrászatokat és fodrászokat. A különböző városokban információ hiány miatt a felhasználóknak gyakran nehézséget okoz a megfelelő szakember vagy szalon megtalálása, miközben a fodrászoknak is hiányzik egy központi platform, ahol könnyen elérhetik potenciális ügyfeleiket. A program egy frontend és backendből épül fel:

**Backend**: Ez a rész felelős a program funkcióiért, azaz az adatok kezeléséért, tárolásáért és feldolgozásáért. A backend kezeli a felhasználói regisztrációkat, a bejelentkezéseket, a szalonok és fodrászok adatainak tárolását, valamint az ügyfelek és fodrászok közötti interakciókat, például az időpontfoglalást. Az adatokat egy adatbázisban tárolja, és biztosítja a felhasználói kérések feldolgozását és a szükséges adatok küldését a frontendnek.

**Frontend**: A program megjelenítéséért felelős réteg, amelyet a felhasználók közvetlenül látnak és használnak. Itt valósulnak meg az interakciók, például a fodrászok keresése, időpontfoglalás, vélemények írása, valamint a felhasználói felület általános kezelése. A frontend biztosítja a felhasználóbarát felületet, ahol a vizuális elemek (gombok, űrlapok, listák) segítenek az információ könnyű megtalálásában és az alkalmazás használatában. A frontend kommunikál a backendel, és megjeleníti a feldolgozott adatokat a felhasználóknak.

**Projekthez használt programok:**

**VisualStudioCode(VSCode)**  
Kódfejlesztő környezet, amelyet különböző programozási nyelvekben való fejlesztések megvalósítására használható. Bővítményekkel és kiegészítőkkel nagyon testreszabható, így ideális webfejlesztéshez és egyéb projektekhez.

**Docker**  
Konténerizálási platform, amely lehetővé teszi, hogy alkalmazásokat futtass konténerekben, elszigetelve a környezettől. Ez segít biztosítani, hogy a programok minden gépen ugyanúgy működjenek, függetlenül az adott környezettől.

**BeekeeperStudio**  
Adatbázis-kezelő eszköz, amelyet relációs adatbázis amely a mi esetünkben PostgreSQL megtekintésére és szerkesztésére használhatsz. Segít az adatbázis-lekérdezések futtatásában és kezelésében.

**Insomnia**  
REST API-k tesztelésére szolgáló eszköz. Használható API-hívások készítésére, fejlesztés közbeni tesztelésre, és könnyen nyomon követheted az API-válaszokat is.

**ReactDeveloperTools**  
Böngészőbővítmény, amely segít React-alapú webalkalmazások fejlesztésében és hibakeresésében. Különösen hasznos a React komponensfa és az állapotok nyomon követésére.

**GitHub**  
Verziókezelő platform, amely lehetővé teszi, hogy projekteket közösségi formában kezelj, megossz, és követni tudd a változtatásokat. Git alapú, így fontos a csapatmunkához és az együttműködéshez.

**Discord**  
Kommunikációs platform, amelyet csapatmunka során használható. Támogatja a szöveges, hang- és videohívásokat is, így ideális projektmegbeszélésekhez és kollaborációhoz.

**Figma**  
Felhőalapú tervezőeszköz, amely grafikai tervek és prototípusok készítésére szolgál. Segítségével UI/UX terveket készíthetsz és együttműködhetsz a csapattal valós időben.

**PostgreSQL**  
Nyílt forráskódú relációs adatbázis-kezelő rendszer, amely strukturált adatokat tárol táblák formájában. Kiválóan alkalmas nagy teljesítményt igénylő adatbázisok kezelésére és komplex SQL lekérdezések végrehajtására, valamint támogatja a JSON típusú adatok kezelését is.

**Kihívások és Tanulságok a Fejlesztés Során**

A **Barber Kereső** fejlesztése során több kihívással is szembesültünk, amelyek különböző szempontokból tettek próbára minket, de egyúttal fontos tanulságokat is hoztak.

Az egyik legnagyobb **kihívás** az adatbázis-kezelés és az adatstruktúra tervezése volt. Mivel az alkalmazásban számos különböző típusú adatot kellett tárolnunk (felhasználói profilok, fodrászszalonok, időpontfoglalások, vélemények), komoly tervezést igényelt az adatbázis logikus és jól skálázható felépítése. A relációk közötti kapcsolatok megteremtése, például a felhasználók és a fodrászok közötti interakciók kezelése, kritikus pont volt a rendszer működése szempontjából. A **tanulság** az volt, hogy az adatbázis szerkezetének alapos megtervezése elengedhetetlen a hosszú távú fenntarthatóság érdekében. Megértettük, hogy a normalizálási szabályok következetes alkalmazása segít elkerülni a redundanciát és a későbbi teljesítményproblémákat.

A másik jelentős **kihívás** a felhasználói hitelesítés és biztonság kérdése volt. Egy olyan platform esetében, ahol a felhasználók személyes adatokat adnak meg, különösen fontos volt a megfelelő biztonsági intézkedések bevezetése. Ez nemcsak a jelszavak titkosított tárolását jelentette, hanem az adatbázis védelmét is, illetve az API-hívások biztonságos kezelését. A **tanulság** itt az volt, hogy a biztonsági elemeket már a fejlesztés korai szakaszában érdemes beépíteni, mivel ezek alapvetően befolyásolják az alkalmazás megbízhatóságát és a felhasználók bizalmát.

A **frontend és backend kommunikációja** szintén kihívást jelentett. A felhasználói felület interakcióit (pl. időpontfoglalás vagy fodrászok keresése) összekötni a háttérrendszerrel és az adatbázissal olyan technológiai kérdéseket vetett fel, amelyek alapos tervezést és tesztelést igényeltek. A **tanulság** ebből az volt, hogy az API-k pontos és megbízható implementálása, valamint a hibakezelés különösen fontos a felhasználói élmény szempontjából, hiszen minden lassulás vagy adatfeldolgozási hiba rontja az alkalmazás használhatóságát.

Összességében a projekt során számos **technológiai és tervezési kihívással** szembesültünk, de ezek segítettek minket abban, hogy mélyebb megértést szerezzünk az alkalmazások fejlesztésének kritikus pontjairól, különösen a biztonság, a hatékony adatkezelés és a skálázhatóság területén.

**Tesztelési és Hibakezelési Stratégiák**

A **tesztelési és hibakezelési stratégiák** kiemelt szerepet kaptak a **Barber Kereső** fejlesztése során, mivel fontos volt, hogy az alkalmazás minden funkciója hibamentesen működjön, és a felhasználói élmény zavartalan legyen. A tesztelés három fő aspektusra koncentrált: a **unit tesztelés** a backend logikai részének biztosítását, az **API tesztelés** a frontend és backend közötti kommunikáció helyességét, valamint a **skálázhatósági tesztelés** a rendszer hosszú távú stabilitásának és teljesítményének biztosítását célozta meg.

A **unit tesztelés** során a fő célunk az volt, hogy minden egyes funkcionális egységet (pl. felhasználói regisztráció, időpontfoglalás, szűrési logika) külön-külön teszteljük, hogy megbizonyosodjunk azok helyes működéséről. A **Jest** keretrendszert használtuk, mivel egyszerűen integrálható a JavaScript-alapú backend alkalmazásunkba, és biztosítja, hogy minden funkció külön-külön tesztelhető legyen. A tesztelés során kiemelt figyelmet fordítottunk arra, hogy a különböző logikai ágakat (pl. hibás adatbevitel, helytelen felhasználói művelet) is ellenőrizzük, ezzel biztosítva, hogy az alkalmazás ne csak a normál működés alatt, hanem a szélsőséges esetekben is megfelelően viselkedjen. A tesztelési eredmények alapján gyorsan lokalizáltuk és javítottuk a logikai hibákat, minimalizálva ezzel a későbbi problémák kialakulásának esélyét.

A **API tesztelés** elengedhetetlen volt ahhoz, hogy a frontend és backend közötti adatkommunikáció hibamentesen működjön. Erre a célra az **Insomnia** eszközt használtuk, amely lehetővé tette számunkra, hogy könnyedén teszteljük az API végpontokat. Az Insomnia segítségével API-hívásokat indítottunk a backend szerver felé, és ellenőriztük, hogy a várt válaszok helyesen érkeznek-e vissza. Különös figyelmet fordítottunk arra, hogy minden egyes végpont megfelelő HTTP státuszkóddal, az elvárt adatstruktúrával és a megfelelő válaszidővel működjön. Az Insomnia lehetőséget biztosított arra is, hogy minden API-hívást előre konfiguráljunk, és tesztelés során ismételten futtassunk, így könnyen azonosíthattuk a hibás válaszokat és a nem megfelelő működést. A hibás API-válaszok gyors feltárása lehetővé tette a backend kód optimalizálását és az API működésének finomhangolását.

A **skálázhatósági tesztelés** szintén fontos része volt a fejlesztésnek, mivel biztosítani szerettük volna, hogy az alkalmazás képes legyen kezelni a növekvő felhasználói igényeket és adatforgalmat, különösen a várhatóan megnövekedett látogatószámok és interakciók esetén. Ezt a tesztelést terheléses tesztekkel végeztük, amelyek segítségével szimuláltuk a magas forgalmat, és figyeltük a rendszer válaszidejét, erőforrás-használatát és teljesítményét. A tesztelés során például több ezer egyidejű felhasználói bejelentkezést és időpontfoglalást generáltunk, hogy meghatározzuk, hogyan reagál a rendszer a terhelésre. Az eredmények alapján optimalizáltuk a backend szervereket és az adatbázis lekérdezéseket, hogy a rendszer gyorsan és hatékonyan tudja kezelni a megnövekedett adatforgalmat, így biztosítva a skálázhatóságot és a hosszú távú stabilitást.

Összességében a **unit tesztelés**, az **API tesztelés** és a **skálázhatósági tesztelés** segítettek minket abban, hogy a **Barber Kereső** alkalmazás ne csak funkcionálisan, hanem teljesítmény szempontjából is optimálisan működjön. A tesztelési folyamatoknak köszönhetően biztosíthattuk, hogy az alkalmazás minden komponense megbízható, gyors és biztonságos, és képes lesz kezelni a jövőbeli felhasználói igényeket is.

**Biztonság és Felhasználói Adatok Védelme**  
A Barber Kereső alkalmazás kifejlesztése során kiemelt figyelmet fordítottunk a felhasználói adatok biztonságára és védelmére, hogy egy biztonságos és megbízható platformot nyújtsunk a fodrászok és ügyfelek számára. Mivel az alkalmazás olyan érzékeny adatokat kezel, mint például felhasználói profilok, foglalási információk és vélemények, elengedhetetlen volt, hogy minden adatvédelmi és biztonsági szempontnak megfeleljünk. A biztonság nem csupán technikai kihívásként, hanem a felhasználói bizalom alapvető feltételeként is szerepelt a projekt során. Az egyik legfontosabb szempont a felhasználói adatok védelme volt, különösen a bejelentkezési adatoké, mint a jelszavak. Az alkalmazásban **bcrypt algoritmust** alkalmaztunk a jelszavak titkosított tárolására, amely garantálja, hogy még egy esetleges adatbázis-kompromittálás esetén is gyakorlatilag lehetetlen visszafejteni a jelszavakat. A bcrypt segítségével minden jelszóhoz egyedi "salt"-ot adunk, amely tovább növeli a titkosítás biztonságát. A felhasználói jelszavak titkosítása nem csupán egy egyszerű hash-elési eljárás, hanem egy olyan módszer, amely az ismételt próbálkozásokkal szemben is ellenálló, így megnehezíti a brute force támadásokat.

### Fő funkciók

#### 1. **Fodrász keresése**

A felhasználók könnyedén kereshetnek fodrászokat a rendszerben:

* **Szűrés város szerint**: A felhasználók szűrhetnek városok vagy földrajzi elhelyezkedés alapján, hogy megtalálják a legközelebbi vagy legmegfelelőbb fodrászt.
* **Térkép megjelenítése**: Egy beépített térkép funkció az összes partner fodrász üzletet megjeleníti, így a felhasználók vizuálisan is könnyedén megtalálhatják a legközelebbi vagy preferált üzletet.
* **Szűrés fodrászok elérhetősége alapján**: A felhasználók látják, mely fodrászok elérhetők egy adott napon vagy időpontban, így egyszerűbbé válik a választás.

#### 2. **Időpontfoglalás**

Az alkalmazás lehetővé teszi az időpontok egyszerű foglalását:

* A felhasználó kiválaszthat egy fodrászt és időpontot, amikor a fodrász szabad.
* Az időpontfoglalás státuszai: a foglalás kezdetben "confirmed" (megerősített), majd ha az időpontot sikeresen megtartják, "completed" (befejezett) státuszba kerül.
* Az időpont foglalása során a felhasználók megadhatnak speciális megjegyzéseket vagy kéréseket.

#### 3. **Fodrász üzletek kezelése**

A fodrász üzletek (vagy fodrász szalonok) szintén megjelennek az adatbázisban:

* Az üzletekhez kapcsolódóan látható a nevük, címük, városuk, telefonos elérhetőségük, e-mail címük, valamint képek is tartozhatnak hozzájuk (pl. borítóképek vagy alapképek).
* Az üzletekhez kapcsolódnak a dolgozók (fodrászok), akik ott dolgoznak, és így az időpontokat is hozzájuk lehet foglalni.

#### 4. **Kapcsolattartás (Chat)**

* Az alkalmazás lehetőséget nyújt a felhasználóknak és fodrászoknak, hogy privát üzeneteket váltsanak egymással, például a foglalás részleteiről vagy különleges kérésekről.
* A felhasználók könnyedén kapcsolatba léphetnek a fodrásszal, ha bármilyen kérdésük van.

#### 5. **Barátságok és Közösségi Funkciók**

* A felhasználók más felhasználókkal is baráti kapcsolatokat hozhatnak létre, ami segítheti a közösségi interakciót. Például megoszthatnak ajánlásokat, véleményeket a szolgáltatásokról.

#### 6. **Értékelések és Vélemények**

* A szolgáltatás teljesítése után a felhasználók lehetőséget kapnak arra, hogy értékeléseket adjanak a fodrászoknak vagy az üzleteknek.
* Ez segít az új felhasználóknak, hogy megalapozott döntést hozzanak, melyik fodrászt válasszák.

#### 7. **Fodrász elérhetőségének kezelése**

* A fodrászok az alkalmazáson keresztül megadhatják elérhetőségeiket (mely napokon és időpontokban dolgoznak), így az ügyfelek ezek alapján foglalhatnak időpontokat.
* Az elérhetőségi időpontok automatikusan frissülnek, amikor foglalás történik.

### Felhasználói út az alkalmazásban

Az alábbi lépéseken keresztül mutatom be egy tipikus felhasználó útját az alkalmazásban a regisztrációtól az időpontfoglalásig.

#### 1. **Regisztráció**

* A felhasználó először regisztrál az alkalmazásban. Ehhez meg kell adnia egy felhasználónevet, e-mail címet és jelszót.
* A felhasználó opcionálisan megadhatja a keresztnevét, vezetéknevét, valamint feltölthet egy profilképet is.
* A regisztráció során ki kell választania, hogy ügyfélként (client) vagy munkavállalóként (worker) szeretné használni az alkalmazást.

#### 2. **Bejelentkezés**

* A regisztráció után a felhasználó bejelentkezhet az e-mail címével és jelszavával, és hozzáfér az alkalmazás funkcióihoz.

#### 3. **Fodrász keresése**

* A bejelentkezést követően a felhasználó kereshet a különböző fodrászok között.
  + A felhasználó használhatja a **város szerinti szűrést** vagy kereshet konkrét **fodrász üzletek** alapján.
  + A térképes keresés különösen hasznos: a felhasználó látja az összes partner fodrász üzletet, és kiválaszthatja, hogy melyikhez szeretne időpontot foglalni.

#### 4. **Fodrász kiválasztása**

* Miután kiválasztotta a megfelelő fodrászt, a felhasználó ellenőrizheti a fodrász **elérhetőségét**. Az elérhető időpontok automatikusan frissülnek, ha más felhasználók foglalnak.
* A felhasználó megtekintheti a fodrász **profilját**, **értékeléseit** és esetleges különleges ajánlatait vagy szolgáltatásait.

#### 5. **Időpont foglalása**

* A felhasználó kiválasztja a kívánt **időpontot** és foglalást indít. A foglalás automatikusan megerősítésre kerül, és a státusza "confirmed" lesz.
* Amennyiben szükséges, a felhasználó a foglalás során megadhat **speciális megjegyzéseket** vagy egyéb kéréséket a fodrász számára.

#### 6. **Kapcsolattartás a fodrásszal**

* Az időpont foglalása után a felhasználó használhatja a **chat** funkciót, hogy kapcsolatba lépjen a fodrásszal, ha bármilyen kérdése vagy különleges kérése van.

#### 7. **Értesítések és megerősítések**

* A felhasználó értesítést kap, ha a foglalás állapota változik (pl. a fodrász elfogadja vagy elutasítja a foglalást), illetve ha az időpont közeledik.

#### 8. **Értékelés**

* Miután az időpont lezajlott és a foglalás státusza "completed"-re változott, a felhasználó értékelést adhat a fodrásznak.
* Az értékelések segítik a közösség többi tagját abban, hogy tájékozott döntéseket hozzanak.

### Összefoglalás

Az alkalmazásod egyszerű és felhasználóbarát módon segíti a felhasználókat abban, hogy fodrászokat találjanak, időpontot foglaljanak, és kapcsolatba lépjenek a szolgáltatókkal. A térképes keresési funkció és az értékelési rendszer kiemelten hasznos funkciók, amelyek fokozzák a felhasználói élményt. Az egyértelmű lépések és értesítések pedig garantálják, hogy a felhasználók könnyedén végigmenjenek az időpontfoglalás folyamatán.

**Adatbázis-terv és Adatszerkezet**

**1. User (Felhasználó)**

A User entitás a felhasználókat képviseli. Ez az alap entitás az alkalmazásodban, és számos más entitáshoz kapcsolódik.

* Fő mezők: userId, username, email, password, opcionális mezők mint firstName, lastName, profilePic.
* A role mezőben két típusú felhasználó létezik: client (ügyfél) és worker (munkavállaló).
* **Kapcsolatok**:
  + **AvailabilityTimes**: A felhasználó elérhetőségei.
  + **Appointment**: Foglalások ügyfelek és dolgozók között.
  + **Friendship**: Felhasználói kapcsolatok (barátságok).
  + **StoreWorker**: Az üzletek dolgozói és tulajdonosai.
  + **ChatRoom**: Chat szobák, ahol felhasználók beszélgethetnek egymással.

Ez az entitás központi szerepet játszik a foglalások, üzletek és üzenetek kezelésében.

**2. Appointment (Időpontfoglalás)**

Az Appointment entitás az ügyfelek és dolgozók közötti időpontfoglalásokat reprezentálja.

* Fő mezők: appointmentId, status (lehetséges értékek: confirmed vagy completed), opcionális notes.
* **Kapcsolatok**:
  + **User**: Kapcsolat ügyfelek és dolgozók között (client, worker).
  + **AvailabilityTimes**: Az időpont elérhetősége (melyik időpontban történik a foglalás).

Az Appointment entitás lehetővé teszi az időpontok nyilvántartását és státuszának kezelését.

**3. AvailabilityTimes (Elérhetőségi idők)**

Az AvailabilityTimes entitás tárolja a dolgozók elérhetőségeit.

* Fő mezők: timeSlotId, day, timeSlot, status (lehetséges értékek: accepted vagy available).
* **Kapcsolatok**:
  + **User**: Munkavállalókhoz kapcsolódik, akik megadják az elérhetőségi időpontjaikat.

Ez az entitás lehetővé teszi, hogy a dolgozók elérhetőségeit könnyen nyomon kövessék és időpontokat foglaljanak az ügyfelek.

**4. ChatRoom (Chat szoba)**

A ChatRoom entitás felhasználók közötti privát beszélgetéseket képvisel.

* Fő mezők: chatroomId, name.
* **Kapcsolatok**:
  + **User**: Két felhasználó (user1 és user2) között létrejövő beszélgetések.

A felhasználók privát üzeneteket válthatnak egymással ezen keresztül.

**5. Message (Üzenet)**

A Message entitás egy adott beszélgetésen belüli üzeneteket tartalmaz.

* Fő mezők: messageId, content, createdAt (üzenet létrehozásának ideje).
* **Kapcsolatok**:
  + **ChatRoom**: Az üzenetek egy adott chat szobában találhatók.
  + **User**: Az üzenet feladója.

Az üzenetküldő funkciót kezeli a chat szobákon belül.

**6. Friendship (Barátság)**

A Friendship entitás a felhasználók közötti baráti kapcsolatokat tárolja.

* Fő mezők: friendshipId, status (lehetséges értékek: pending, accepted, rejected).
* **Kapcsolatok**:
  + **User**: Két felhasználó között hoz létre kapcsolatot (user és friend).

Ez az entitás kezeli a felhasználói kapcsolatokat (barátkérések és státuszuk).

**7. Store (Üzlet)**

A Store entitás üzleteket reprezentál, ahol a dolgozók foglalásokat kezelhetnek.

* Fő mezők: storeId, name, description, address, city, postalCode, phone, email, latitude, longitude.
* **Kapcsolatok**:
  + **StoreWorker**: Az üzlethez tartozó dolgozók.
  + **StorePictures**: Az üzlethez tartozó képek.

Ez az entitás az üzletek menedzselését támogatja, beleértve a helyszíneket és dolgozókat.

**8. StoreWorker (Üzlet dolgozója)**

A StoreWorker entitás az üzlet dolgozóit és tulajdonosait tárolja.

* Fő mezők: storeWorkerId, role (lehetséges értékek: owner, worker).
* **Kapcsolatok**:
  + **Store**: Az üzlet, ahol a dolgozó munkavégzést végez.
  + **User**: A dolgozó felhasználó.

Ez az entitás kezeli, hogy ki dolgozik vagy tulajdonos egy adott üzletben.

**9. StorePictures (Üzlet képek)**

A StorePictures entitás az üzletekhez tartozó képeket kezeli.

* Fő mezők: pictureId, imageUrl, imageType (lehetséges értékek: cover\_image, basic\_image).
* **Kapcsolatok**:
  + **Store**: Az üzlet, amelyhez a kép tartozik.

Az üzletek vizuális megjelenítését szolgálja.

**10. ExtendedWorker (Kiterjesztett munkavállaló)**

Az ExtendedWorker entitás a munkavállalókhoz kapcsolódó extra adatokat tartalmazza.

* Fő mezők: userId, opcionális description.
* **Kapcsolatok**:
  + **User**: Egy munkavállalóhoz kapcsolódik.

Ez az entitás további információkat tárol a munkavállalókról, például leírásokat.

**Összefoglalás:**

Az adatbázisod komplex és jól struktúrált, amely különböző funkciókat biztosít az alkalmazásod számára. A legfontosabb kapcsolatok a User entitás köré épülnek, amely lehetővé teszi az időpontfoglalásokat, üzletek menedzselését, üzenetküldést, valamint a felhasználói kapcsolatok kezelését.

 **Piackutatás és Felhasználói Igények ~**

* Ebben a szakaszban részletezhetnéd, hogyan és milyen módszerekkel végeztétek a piackutatást, milyen adatokat gyűjtöttetek a felhasználói igényekről, és mik voltak a legfontosabb megállapítások.

 **Funkcionalitás és Felhasználói Út**

* Leírhatnád részletesen a program fő funkcióit (pl. fodrász keresése, időpontfoglalás, értékelések, szűrés városok szerint). Emellett leírhatnád egy tipikus felhasználó "útját" az alkalmazásban a regisztrációtól az időpontfoglalásig.

 **Adatbázis-terv és Adatszerkezet**

* Részletezhetnéd, hogyan épül fel az adatbázis, milyen táblákat használtatok, és hogyan történik az adatok tárolása. Továbbá bemutathatnád az egyes kapcsolódó entitásokat (pl. felhasználók, fodrászok, időpontok).

 **Fejlesztési Eszközök és Technológiák ~**

* Bár már van egy lista a használt eszközökről, itt bővebben kifejthetnéd, miért választottátok ezeket az eszközöket, és hogyan segítették a fejlesztést (pl. miért PostgreSQL, miért Docker stb.).

 **Frontend és Backend Architektúra**

* Itt részletezhetnéd az alkalmazás szerkezeti felépítését. Bemutathatnád, hogyan kommunikál a frontend a backenddel (API hívások, adatfolyamok), és hogyan történik a különböző felhasználói interakciók kezelése.

 **Tesztelési és Hibakezelési Stratégiák ~**

* Leírhatnád, milyen módszerekkel teszteltétek az alkalmazást (pl. unit tesztelés, manuális tesztelés). Továbbá bemutathatnád, hogyan kezelitek a felmerülő hibákat és problémákat a fejlesztés során.

 **Biztonság és Felhasználói Adatok Védelme ~**

* Ebben a szakaszban bemutathatnád, milyen lépéseket tettetek a felhasználói adatok védelme érdekében (pl. jelszavak titkosítása, adatbázis-biztonság, hitelesítés).

 **Jövőbeli Fejlesztési Lehetőségek**

* Itt felsorolhatnád, milyen további funkciókat terveztek hozzáadni az alkalmazáshoz a jövőben (pl. mobilalkalmazás fejlesztése, új szűrési lehetőségek, további fodrászok hozzáadása).

 **Felhasználói Visszajelzések és Fejlesztési Irány**

* Kiemelhetnéd a felhasználói tesztek eredményeit, visszajelzéseket és ezek alapján tett javaslatokat a további fejlesztésekre.

 **Kihívások és Tanulságok a Fejlesztés Során ~**

* Leírhatnád, milyen nehézségekkel találkoztatok a projekt során, és hogyan oldottátok meg ezeket. Továbbá részletezhetnéd, milyen tapasztalatokat és tanulságokat szereztetek a folyamat során.