**FootFrenzy(mobil)**

Fejlesztői dokumenáció

App.js:

* PaperProvider: Ez a tag teszi lehetővé a React native Paper(Material Design library) használatát
* NavigationContainer és Stack.Navigator: Ez a tag teszi lehetővé a screenek közti navigálást
* Stack.Screen: Ez a tag teszi lehetővé a screenek hozááadását

**Screenek**

AccountScreen.js:

* getUserData(): ez a funkció kéri le a bejelentkezett fiók információit
* logOut(): ez a funkció teszi lehetővé a kijelentkezést a fiókból, ezután átnavigál a „Store” screenre.

CartScreen.js:

* getUserData(): ez a funkció kéri le a bejelentkezett fiók információit
* getData(): ez a funkció kéri le a kosár tartalmát
* getTotalPrice(): ez a funkció kéri le a kosár tartalmának az összesített árát.
* deleteCartItem(): a kosár egyik elemét törli ki
* updateStock(): ha a kosárból kitöröl a felhasználó egy elemet, akkor ez a funkció visszatölti azt az elemet az adatbázisba

Checkout.js:

* getUserData(): ez a funkció kéri le a bejelentkezett fiók információit
* validateFullName(): az egész név textinputot validálja(Minimum 2 szó, nem üres)
* validateCounty(): a megye textinputot validálja(Nem lehet üres)
* validateCity(): a város textinputot validálja(Nem lehet üres)
* validatePostalCode(): az irányítószám textinputot validálja(4 karakternek kell lennie)
* validateStreetAddress(): az utcai cím textinputot validálja(Legalabb egy szó + szám)
* addressDone(): az összes eddigi validációt meghívja, ha minden rendben és a kiszáttítási módszer is kivan választva, akkor lekéri a megfelelő kiszállítási módszert és átléptet a következő oldalra
* validateFullNameCr(): az egész név bankártya textinputot validálja(Minimum 2 szó, nem üres)
* validateCardNumber(): a kártyaszám textinputot validálja(16 karakter)
* editCardNumber(): a kártyaszám textinputjába lévő szöveget szerkeszti(minden 4. szám után space)
* validateExpirationDate(): a lejárati idő textinputot validálva(4 karakter, se a hónap sem az év nem járt le)
* editExpirationDate():a lejárati idő textinputjába lévő szöveget szerkeszti(2 szám után „/” jel beillesztése, ha a hónap nem 2 számjegyű 0 beillesztése)
* validateSecurityCode(): a cvc kód textinputot validálja(3 karakter)
* validatePaypalEmail: a paypal emial mezőt validálja(email szintaxis regex használatával)
* validatePaypalPassword(): paypal jelszó mezőt validálja(legalább 8 karakter, nagybetű, szám vagy speciális karakter regex használatával)
* paymentDone(): a kiválasztott fizetési módszertől függően validálja a megfelelő mezőket, lekéri a kiválasztott fizetési módszert, majd tovább lép a következő oldalra.
* submitOrder(): először postolja a cím adatait és visszaadja a feltöltött address objektumot, aztán ezt az address objektumot segítségül hívva postolja az rendelés adatait ami visszaadja a feltöltött order objektumot, majd ezt az order objektumot segítségül hívva a kosár tartalmán végigmegy, és meghívja a postOrderItem() függvényt a megfelelő adatokkal. Ezután a deleteCart() funkciót meghívja és tovább lép a következő oldalra
* getPaymentMethod(): lekéri a megfelelő fizetési metódust.
* getData(): lekéri a kosár tartalmát
* getTotal(): ez a funkció kéri le a kosár tartalmának az összesített árát.
* postOrderItem(): Feltölti a kosár tartalmát az order\_item táblára
* deleteCart(): kitörli a kosár tartalmát

HomeScreen.js

Liked.js:

* getUser():ez a funkció kéri le a bejelentkezett fiók információit
* getLikedItemIds(): lekéri az aktuális felhasználó liekolt cipői id-jait, majd ezeken végigmenve meghívja a getLikedItems() funkciót.
* getLikedItems(): lekéri a cipő adatait id alapján

Product.js:

* getUserData(): ez a funkció kéri le a bejelentkezett fiók információit
* getData(): lekéri az aktuálisan megnyitott cipő adatait
* getSizes(): lekéri, hogy hány cipő van egy mérethez
* getLikes(): lekéri hány like van a cipőn
* setImage(): beállítja, hogy melyik kép legyen megnyitva nagyba
* addLike(): megnézi , hoyg az adott user likeolta-e már az adott cipőt, ha nem belájkolja, ha igen levonja a likeot, ezután meghívja a getLikes() funkciót
* amountReduce(): mennyiséget csökkenti, hacsak nem 1 alá akar menni
* amountIncrease(): mennyiséget növeli, hacsak nem a mérethez rendelkezésre álló cipőmennyiségen túlmegy.
* amountFix(): 0-ra állítja a mennyiséget
* postCart(): posztolja a cipő adatait a cart\_item táblára
* updateStock(): ha a kosárba rak a felhasználó x cipőt akkor x-el csökkenti a mennyiséget az adatbázisban
* addToCart(): megnézi hogy minden rendesen van-e megadva, és ha igen akkor meghívja a postCart() függvényt

Register.js:

* validateEmail():emial textinputot validálja és a megfelelő hibaüzenetet beállítja(email szintaxis regex használatával, ezzel az email címmel még nem regisztráltak)
* validateUsername(): username textinputot validálja és a megfelelő hibaüzenetet beállítja(minimum 6 karakter, ezzel a névvel még nem regisztráltak)
* validatePassword(): jelszó textinputot validálja(legalább 8 karakter, nagybetű, szám vagy speciális karakter regex használatával)
* validateRePassword() a jelszó újra textinputot validálja(egyezzen a jelszóval
* postRegister(): postolja az adatokat az adatbázisba
* register(): meghívja a validációs függvényeket, majd ha mindent rendben talált, meghívja a postRegister() függvényt

SearchScreen.js:

* getPopularshoes(): lekéri azokat a cipőket amiknél a populáris jelző igaz
* getNewShoes(): lekéri az összes cipőt, majd feltöltési sorrend szerint sorba teszi(id alapján)
* getData(): attól függően, hogy hogyan léptünk át erre az oldalra lekéri a megfelelő cipők adatait
* searchByName():lekéri azokat a cipőket, amiknek a neve egyezik a keresés textinput mezőben lévő szöveg tartalmával

SignIn.js:

* handleLogin(): megnézi, hogy minden adat megfelelő-e, majd megpróbálja feltölteni az adatokat. Ha minden megfelelő akkor beléptet és átdob a Store screenre, ha nem akkor a megfelelő hibaüzenetet dob vissza.

Store.js:

* getReleases(): lekéri a hamarosan megjelenő cipők adatait
* getPopularshoes(): lekéri azokat a cipőket amiknél a populáris jelző igaz
* getNewShoes(): lekéri az összes cipőt, majd feltöltési sorrend szerint sorba teszi(id alapján)
* getShoes(): lekéri az összes cipőt

**Komponensek:**

CarouselItem.js: cipő kategóriákat segít megjeleníteni

CartItem.js: a kosár tartalmát segít megjeleníteni

* getProduct(): a megfelelő id alapján lekéri az adott cipő adatait

CheckoutItem: a Checkout screenen segít a kosár tartalmát megjeleníteni

* getProduct(): a megfelelő id alapján lekéri az adott cipő adatait

Navbar.js: a navigációs sáv

Products.js: egy cipő adatait segít megjeleníteni a Store screenen

ReleaseCarouselItem.js : a hamarosan megjelenő cipők adatait segít megjeleníteni a Store screenen

SneakerScroll.js: SneakerScrollItem-eket mappolja ki

SneakerScrollHeadear.js: a SneakerScroll fejlécét jeleníti meg

SneakerScrollItem.js: a SneakerScrollban segít az adott cipő adatait megjeleníteni