Sapientia Erdélyi Magyar Tudományegyetem

Marosvásárhelyi Kar

Uniting Bikers through Cloud based Mobile Solutions

„Social media of Bikes”

Készítette:

Fodor László-Carlos

Hegyeli András

Koordinátor:

drd. Oltean-Péter

Boróka

2023-2024

Tartalom

Bevezető

Az alkalmazás, amivel az alábbiakban részletesebben is megismerkedünk, egy biciklis szociális felület. Ez egy React Native alapú cross-platform telefonos applikáció. A projekt a BicyHub nevet viseli, amit mottója is takar: „Social media of bikes”. Lényegében egy biciklis közösségnek egy platformot ad az egymás közötti kapcsolattartásra, segít a témában való informálódásban.[1]

Az applikáció teret biztosít, azoknak a biciklizni szeretőknek, akik egy közösséghez szeretnének tartozni. Konkrétan, bejegyzések, kommentek, tanácsok segítségével lehet ezt a különleges szociális hálót fenttartani. Nagy előnye a valós idejű üzenet küldési lehetőség, ami még személyessebbé teszi az alkalmazás felhasználói számára a biztosított teret, hiszen nem kell felületet váltani a kommunikációra, hanem ugyanazon az alkalmazáson belül is lehetőség van erre.

Azoknak, akik a kerékpározási teljesítményüket nyomon szeretnék követni, azoknak is alkalmas a felület, hiszen lehetőségük nyílik itt a GPS alapú útrögzítésre. A felhasználó mentheti a megtett útjait a telefonja segítségével, amiket később visszanézhet és következtetéseket vonhat le belőle, például, hova szeretne vissza menni még, illetve hova nem. Ezek mellett lehetősége van ezek megosztására is másokkal, amik mind a közösségi, mind az egyéni biciklizési élményt javítani tudják a kollektív tapasztalat álltal.

Az online kapcsolattartás mellett a BicyHub lehetőséget ad, azok számára, akik ezt választják, arra, hogy fizikailag is kapcsolatot tartsanak azokkal, akik a környékükön vannak. A GPS alapú követés lehetőséget ad arra, hogy egy felhasználó tervezzen egy biciklis túrát többek számára. Ez előnyös azoknak, akik el szeretnének menni tekerni és társaságra is vágynak a hozzájuk közel sportolókkal.

Ezek mellett egy biciklinyilvántartó jellege is van, ami segítségével nyomon lehet követni milyen változtatásokon esett át, szenvedett el valami féle sérülést, esett át valamilyen fejlesztésen. Ez hasonlít a járműveknél használt szervízkönyv logikához, azzal a különbséggel, hogy itt a felhasználó vezeti be az általa vagy más által végzett módosításokat.

[1] vö. [8]

Jelentős előnyt tud biztosítani a felhasználók számára a lehetőség, hogy konkrét bicikli szervizekre tudnak rákeresni a környékükön, de nem csak, vagy olyan felhasználókat is megkereshetnek, akik képesek kisebb-, nagyobb javításokat elvégezni.

Lehetősége van a felhasználónak ezek mellett szervízt létrehozni két kategóriában. Az első lehetőség a szakképesített szervíz, amit majd az alkalmazáson belül elérhetnek az erre a szolgáltatásra vágyók, illetve a nem szakképesített-, „házi" szerelők, akik más bicikliseknek tudnak segíteni javítani, elemeket cserélni nem szakképzett módon.

Ezek mellett lehetővé teszi az applikáció a biciklik cseréjét is felhasználók között a tranzakció megvalósítással, ami által nem kell külön bevezetni egy biciklit, ha az alkalmazáson belül tettünk szert rá, hanem meg lehet örökölni az egész történelmét ennek az eszköznek.

A projekt célja

A munkánk célja az volt, hogy egy alkalmazást biztosítsunk azok százára, akik rendszeresen szeretnek a biciklikkel foglalkozni. Éppen ezért hoztuk létre ezt az applikációt, ami segítséget nyújthat a biciklink (biciklikjeink), nyilvántartásában, valamint támpontokat adhat azoknak, akik nem otthonosak még egészen a témában. Az alapötlet egy biciklis szociális háló kialakítása lenne, amin keresztül a felhasználók információt vagy eszközöket cserélhetnek a kerékpárjaikhoz.

Szeretnénk egy közös teret biztosítani azok számára, akik kerékpárokat felhasználnak és azoknak, akik ezek szervizelését, javítását, karbantartását kezelik. Ezek mellett teret szeretnénk biztosítani a kommunikációra és eszmecserére a témán belül, ami egy összetartó közösségek hálóját lenne képes biztosítani.

Ugyancsak cél volt az online és offline világ összecsatolása, az által, hogy követni lehet mások túráit, tapasztalatait, véleményét, ami mellett csatlakozni lehet egy mások által szervezett túrához, amik nem csak a virtuális-, de a fizikai interhumánus kapcsolatokat is nagyban elősegíti.

Hasonló alkalmazások

BuyCycle

Ebben a kategóriában hasonló alkalmazás a Buycyle, ami kifejezetten egy bicikli kereskedő oldal. Az első változat az Amerikai Egyesült Államokban látott napvilágot, majd

későbbi terjeszkedése során Európába is eljutott. Lényegében egy európai szintű teljes bicikli és bicikli alkatrészeket árusító alkalmazásról beszélhetünk, ami inkább használt kerékpárokra specializálódott.

Előnye közé tartozik a kereskedelmi terület mérete, amit lefed, a felhasználók nagyszámú jelenléte és a részletes leírások jelenléte. Hátrány a BicyHub-al szemben, hogy nem tartalmaz semmilyen szervízkönyvhöz hasonló tulajdonságot. Ebből kifolyólag, míg az első csak az eladás pillanatában biztosít információt a kerékpárról, addig a második visszamenőleg is képes a bicikli történetét megmutatni, így képes egy sokkal részletesebb és komplexebb képet biztosítani a biciklirő l.

Cycling Plus Magazine

Egy másik hasonló alkalmazás a Cycling Plus Magazine. Az utóbb említett alkalmazás jobban fókuszál a közösségére a biciklis életnek. Mindez azt jelenti, hogy nem magára az eszközre összpontosít, hanem az emberek, a biciklihasználók véleményére alapoz. Előnyben érezheti magát a Cycling Plus Magazine felhasználó az által, hogy egyes helyeken szakemberek véleményéből kiindulva ír cikkel egy-egy kerékpárról, ez mellett nem csak az eszközről, hanem a hozzá kapcsolódó életmódról, táplálkozásról, eseményekről is hoz érdekes adatokat.

A BicyHub előnye ezzel szemben a tényleges biciklihasználó élménye és tapasztalata, ami túlmutat a puszta elméletiségen. Sokkal kézzel foghatóbban láthatjuk egyes biciklikről, hogy mire is képesek vagy hol vannak buktatói, gyengeségei a konkrét eszköznek, hiszen nem csak egy cikk elméleti követelményeire támaszkodik, hanem valós tapasztalatokra.

Komoot - hike, bike \& run

A komoot - hike, bike \& run is összehasonlítható a BicyHub-bal. Közel áll két alkalmazás abban a tekintetben, hogy mind a kettőben jelen van a biciklis közösség élményének a megtartása. Ugyanakkor különbség az, hogy az első a bicikli ösvényekkel, biciklis tippekkel szolgál, erre teszi a hangsúlyt, addig a második a eszközre, annak kiegészítőjére, történetére fókuszál. Mindezt az mellett, hogy rendelkezik hasonló funkciókkal, mint a túraútvonalak és élménymegosztás.

A komoot - hike, bike & run egy utazó, túrázó alkalmazás nem kizárólag biciklisek számára, amiben a hangsúly a pontos GPS helymeghatározására és a túrautak pontos leírására kerül. Jellemző erre az alkalmazásra a közösségi tapasztalat megosztása, amit posztok, kommentek formájában láthatunk a felületen. Ezzel szemben a BicyHub a kerékpárra teszi a hangsúlyt.

Mondhatnánk úgyis, hogy az érem másik oldala, ahol a felhasználók a kerékpárjaikról oszthatnak meg élményeket, tapasztalatokat. Egy jó biciklis túrázáshoz egy jó túraút és egy megfelelő kerékpár kell, ami ezzel a két alkalmazással sokkal könnyebben elérhető.

Követelmények

Rendszerkövetelmények

Az alkalmazás használatához szükség van egy eszközre, amin Android vagy IOS operációs rendszer fut. Jelenlegi állapotában az ExpoGo-n keresztül, fejlesztői környezetben lehet futtatni mind a két platformon. Androidra egy APK is ki van telepítve szintén az Expo-ra, amit meghívással lehet elérni csak.

Képernyőméret szempontjából az alkalmazás egy átlagos telefonméretre volt tervezve (~4 – 6 inch) az ideális felhasználói élmény szempontjából.

Az alkalmazás offline használata nem támogatott, a jellegéből adódóan, ezért az internethozzáférés szükséges.

Fejlesztői környezet:

Keretrendszernek a React Native-et választottuk (Figure 1). React Native egy mobilalkalmazás-fejlesztési keretrendszer, amelyet a Facebook fejlesztett ki. Segítségével egyetlen kódbázisból tudtunk készíteni iOS és Android alkalmazásokat is, mivel React alapú és JavaScript nyelvet használ.

A React egy JavaScript alapú könyvtár, amit első sorban felhasználói felületek megvalósítására használnak. Egy viszonylag fiatal technológiáról van most, szó, amit 2011-ben Jordan Walke a Facebook mérnöke fejlesztett ki.

Elsődleges célja az volt, hogy egy környezetet biztosítson a külön komponens alapú fejlesztésre. Azt értem ez alatt, hogy a fejlesztő ennek a könyvtárnak a segítségével könnyedén tudja megvalósítani a felhasználói felületet, oly módon, hogy az egyes komponensek szinte vagy teljesen különállóak. A fejlesztőnek a dolga hasonló a moduláris építkezéshez, mert különálló komponenseket hoz létre, amiket majd később összekapcsol, ilyen módon a kinézet, a logika és a komponensalkotás egy helyre került.

Ebből a megvalósításból nőtte ki magát a React-Native. Ez egy cross-platform keretrendszer, ami a telefonos fejlesztésben hozott megújulást. Az alap elképzelés az, hogy egy

forráskóddal több platformra lehessen egyszerre fejleszteni, legyen az Android vagy IOS. Ez úgy valósul meg, hogy a fejlesztő ennek a keretrendszernek a segítségével létrehozza az applikáció kódbázisát. Pusztán a kódot nézve azt gondolhatnánk, hogy egy weboldalról van szó, hiszen a React elemeket használja, illetve bővíti ki. Fordításkor a keretrendszer (a beállításoktól függően) natív Android vagy IOS kódot generál, ami hozza magával a könnyen újrahasznosíthatóságot.

Az alkalmazás készítésekor a fent említett ok játszott jelenős szerepet, mivel egy kódbázisból két platformra lehet egyszerre fejleszteni. Szerepet játszott még a React-Native azon tulajdonsága, hogy az újrahasznosítás központi elemet foglal el a felépítési koncepciójában. Ez abban nyilvánul meg, hogy a többször felhasznált elemeket nem szükséges több helyen megírni, hanem egy helyen megírva és az szükséges területeken felhasználva egy gyorsított folyamatot biztosít a fejlesztésben. Ez mellett nagy mértékben csökkenti a kódtöbbszörözést és a hibakeresést is nagy mértékben egyszerűsíti.

A több felületen fejlesztés, ami a nagy előnye, az egyben a hátránya is, mivel a ebben e keretrendszerben megírt alkalmazások valamivel lassabbak, mint a natív Android vagy IOS platformon megírtak

Fejlesztői környezetünk az Expo volt. Expo egy nyílt forráskódú keretrendszer, amely megkönnyíti a React Native alkalmazások fejlesztését. A legnagyobb előnye, hogy cross-platform alkalmazást hozhattunk létre és az Expo CLI használatával könnyedén lehetőség van az alkalmazást tesztelni és futtatni különböző eszközökről.

Az Expo és Expo Go egy ingyenes platformot alkot úgy Android, mint IOS alkalmazások megépítéséhez. Ez react alapokon nyugszik. Ami miatt fontos lett a fejlesztés szempontjából, belátva azt, hogy a React-Native is képes a számunkra szükséges funkciókat ellátni, az az, hogy a keretrendszer képes lokálisan összekötni a megírt kódot és a saját Android vagy IOS eszközünket.

Ez a fejlesztésben nyújtott nagy segítséget, hiszen kódgépelés közben nyílt lehetőség az adott kódsor vizualizációjára egy konkrét fizikai eszközön. Ebből kifolyólag a fejlesztés és a tesztelés is egyszerre le tudott zajlani. Nem mellesleg a megvalósítása is viszonylag egyszerű, hiszen ha helyesen telepítve van a csomag és az applikációval is rendelkezik a fejlesztő, akkor egy

egyszerű QR kód leolvasása után vagy az URL beírása után máris "varázsütés szerűen" össze lett kapcsolva a két eszköz.[2]

Az applikáció több nyelven való megjelenítéséért az i18next keretrendszer felel. A központ logikája ennek a keretrendszer az egyszerű kulcs-érték párokon alapuló nyelvimplementáció.

Működését tekintve három nagyobb részre lehetne szétszedni. A legmélyebb szint egy json-ban megadott kulcs-érték pár logika szerint megadott szófordítás. A második szint a keretrendszer inicializálása és az első szinttel való összekötés. A harmadik a komponensen való meghívás, valamint egy nyelvváltó megalkotása, ami az i18next "jelenlegi nyelv" állapotának befolyásolására szolgál.

UI könyvtárunk kezdetben a NativeBase volt (Figure 2), ami egy React Native alapú UI, amely előre elkészített, stílusos és hasznos UI elemeket kínált. A NativeBase segítségével egyszerűbben és gyorsabban tervezhettünk és fejleszthettünk mobilalkalmazásokat anélkül, hogy minden UI elemet kézzel kellene elkészítened. [3]

Idő közben ez elavulttá vált, ezért ennek a továbbfejlesztett választottuk, ami a GlueStack UI (Figure 5). A Gluestack UI egy univerzális UI könyvtár, amely opcionálisan stílusolt és hozzáférhető komponenseket biztosít React Native, Next.js, Expo és React számára. Célja, hogy egyszerűsítse a fejlesztési folyamatot egy készen használható komponensekből álló készlettel. Ez mellet fontos szempont volt, hogy az átállást lépésenként biztosította, így nem kellett egyszerre az egész felhasználói felületet újraírni.

A fő elképzelés a megvalósítás mögött az volt, hogy szükség van egy olyan könyvtárra, ami könnyen felhasználható, előre elkészített komponenseket képes biztosítani a fejlesztők számára, amik kompatibilisek a React-Nativel. Ezek a komponensek csak alap stílusozást kaptak, így lehetőséget hagytak a fejlesztőnek az igény szerinti felhasználásra, amit az elemek testre szabása biztosít.

Ezek után következett egy váltása a Gluestack UI-ra. Ennek oka az volt, hogy a NativeBase támogatottsága megszűnt. A Gluestack UI a NativeBase újragondolt változata. A választás azért

[2] vö. [1]
[3] vö. [5]

esett erre, mert lehetőség nyílt ezzel az eszközzel az iteratív átállásra, így nem volt szükség a teljes alkalmazás újraírására.[4]

A könyvtár fejlesztői szerint azért volt szükség az átállásra, mert a NativeBase határozottan lassabb volt, mint a natív Andriod vagy IOS kódok, mivel a NativeBase saját komponenseket adott a felhasználónak. Ezzel szemben a továbbfejlesztett változatban nem előre megírt, önálló komponenseket kapunk, hanem a React-ben található és felhasználható elemek stilizált változtatás. Ebből kifolyólag az új megoldás majdnem 15-ször gyorsabb lett mint a régi, összehasonlítva a NativeBase-el, de még mindig 1,5-ször lassabb, mint a natív nyelven megírt kód.[5]

Visual Studio Code volt a kódszerkesztőnk (Figure 3), ami egy könnyű, de erőteljes eszköz, amelyet a Microsoft fejlesztett ki. Ideális választás volt a React Native alkalmazás fejlesztéséhez, mivel támogatja a TypeScriptet és rengeteg kiegészítővel rendelkezik, amelyek megkönnyítette a fejlesztési folyamatot.

Programozási nyelvünk a TypeScript volt (Figure 4). Ez egy szuperszett a JavaScriptről, amely statikus típusellenőrzést tesz lehetővé a fejlesztők számára. A React Native alkalmazás fejlesztéséhez használva növelhettük a kód biztonságát és olvashatóságát, valamint segített a hibák korai azonosításában és kezelésében.

A TypeScript egy programozási nyelv, ami obiektumorientált és prototípus alapú. A köztudatban úgy terjedt el, mint a weboldal fejlesztés nyelve, de ezzel szemben számos más helyen használják, mint eszközt, például a Node.js, az Adobe Acrobat, az Apache és CouchDB. Nagy előnye az egyszerű JavaScripttel szemben, hogy típusozást biztosít (amint fentebb említettem), ezért statikusan, még futtatás előtt kiderülhetnek az esetleges hibalehetőségek. Ennek okán sokkal könnyebb felépíteni és karban tartani egy kódot és ez mellett sokkal hamarabb meg lehet találni egyes hibák forrását, illetve ezeket a hibákat sokkal könnyebb elkerülni ennek a programozási nyelvnek a segítségével.

Függőségeinek kezelésére NPM-et (Node Package Manager) használtunk (Figure 6). Az NPM segítségével könnyedén telepíthettünk és frissíthettünk különböző csomagokat, amelyek segítettek a fejlesztési folyamatok során.

[4] vö. [4]
[5] vö. [5]

Az ESLint(Figure 7) és a Prettier(Figure 8) biztosítják a kód minőségét és a következetes formázást. Az ESLint egy statikus kódvizsgáló eszköz JavaScript-hez. Lehetőséget biztosít a kódminőség ellenőrzésére és a fejlesztői szabályok betartatására. A Prettier egy egyszerűen konfigurálható kódfomázó eszköz, amely segít a kód egységes és olvasható formázásában. Függetlenül működhet az ESLint-től, de könnyen integrálható vele, így a kódformázás és a statikus kódvizsgálat is biztosítva volt ez a két eszköz álltal.

Az alkalmazás Expo Application Services-el van kitelepítve Androidra (APK). Az EAS egy eszközkészlet, amelyet az Expo, az univerzális React Native alkalmazások építésére és telepítésére szolgáló platform biztosít. Az EAS a fejlesztők számára átfogó funkciókat kínál a fejlesztési folyamat racionalizálására, többek között, de nem kizárólag a natív buildeket, az over-the-air frissítéseket és az analitikai nyomon követést.[6]

Funkcionális követelmények:

Felhasználói szempontból funkcionális követelménynek tudható be a fiókkal, illetve fiók nélküli használat. Első esetben teljes lehetőséget nyújt az app. Ilyen esetben, sikeres regisztráció és bejelentkezés után, a felhasználó máris használhatja az applikációt. Böngészhet a főoldalon megjelenő biciklik között, biciklik részletes leírásait tekintheti meg, profilokat tekinthet meg, ezekről részletes információkhoz is hozzájuthat. Ezek mellett írhat kommenteket más felhasználóknak, like-olhatja más felhasználók bejegyzéseit, illetve kommentelhet ezekre.

Bejelentkezett módban a felhasználónak lehetősége nyílik ezek mellett a mások által létrehozott túrákhoz csatlakozni, megtekinteni az elmentett útvonalait, megosztani ezeket másokkal. A regisztráció és bejelentkezés feljogosítja a felhasználót arra, hogy szervizek között válogasson, illetve saját szervizt hozzon létre, amennyiben szeretne, ezeket értékelheti, illetve megjegyzéseket írhat ezekhez. Lehetőséget nyújt az alkalmazás ebben az esetben a valós idejű üzenetküldésre, valamint a felhasználók közötti gyors keresésre.

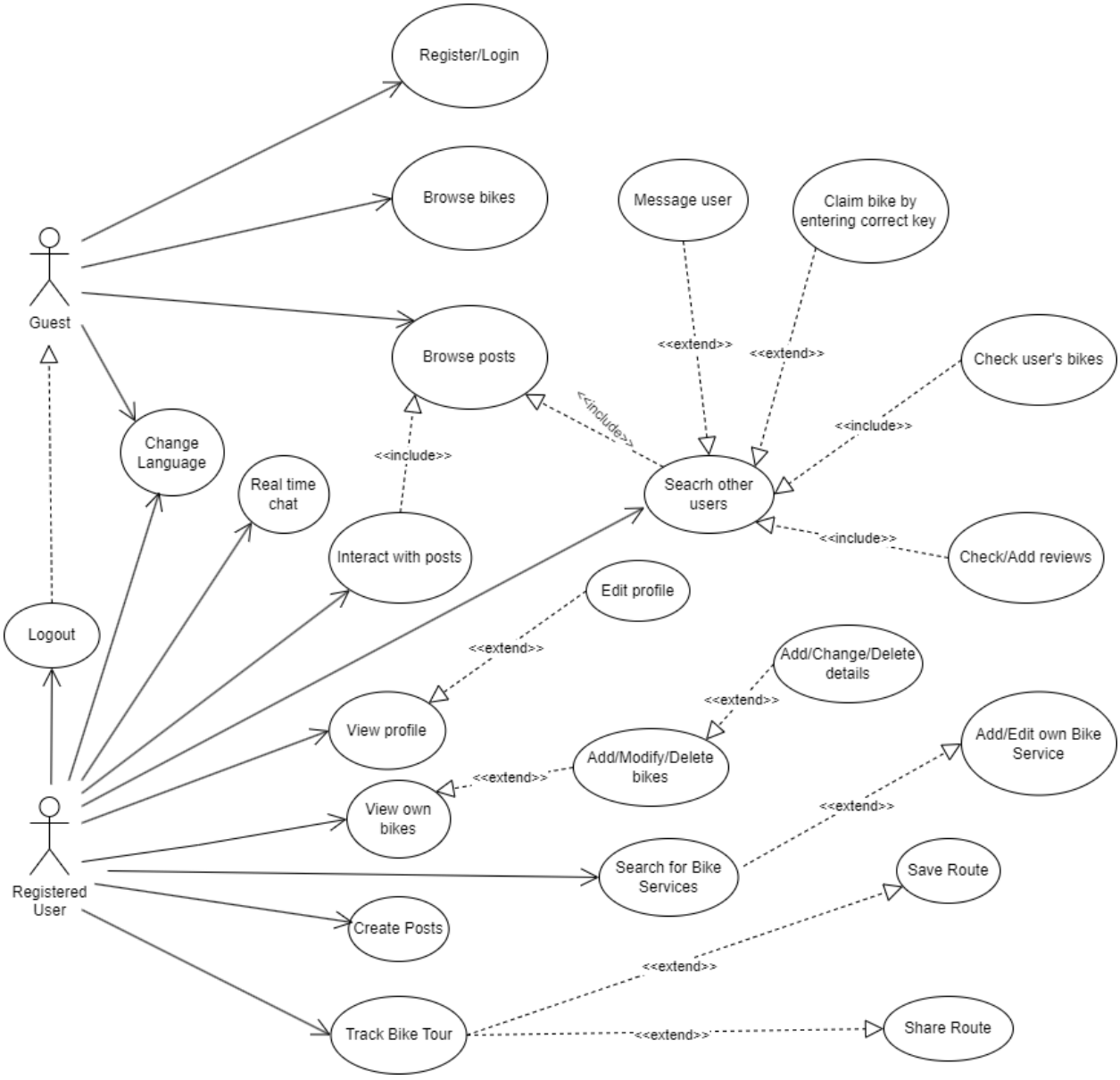
Második esetben, ha „Guest” (vendég) módban lép be, akkor jelentősen korlátozódnak a lehetőségei. Szintén megtekintheti a bicikliket a platformon, ezeknek a részletes leírását is. Profilokat is megtekinthet, de ellenben a bejelentkezetettekkel szemben nem látja azok

[6] vö. [2]

részletes/személyes információit. Kommentek olvasására képes, de nem tud saját bejegyzést kitenni, nem képes ezekre reagálni (legyen az komment vagy like). Ezek mellett a „vendég” szintű felhasználónak lehetősége van a nyelvváltásra is.

Fontos funkció a biciklik cseréje egymás között, amit szintén csak és kizárólag bejelentkezett állapotban lehet megtenni, valamit ilyen funkciók a saját profil szerkesztése, valamint a saját bicikli hozzáadása és azok szerkesztése.

Use case diagram



1. Ábra Use-case diagram

Feltételezzük hogy a felhasználónk először nyissa meg az alkalmazást, természetesen guest-kent fogja használni, ami azt jelenti hogy meg fogja tudni nézni a posztokat es a bicikliket amiket a többi user rakott ki es fog tudni regisztrálni , meg bejelentkezni. Be jelentkezés után létre tud hozni posztokat , likeolni , kommentelni , lesz saját profilja amit tud szerkeszteni , saját

bicikliket fog tudni posztolni es annak az adatait szerkeszteni , más usereket ki értékelni(review) es tranzakciókat végre hajtani biciklikkel a helyes kulcs kód beírásával.

Felhasználás:

Itt is két esetet szeretnék tárgyalni, amik közül az első a vendég módban történő használat, a második a bejelentkezett módban történő használat.

Az applikáció megnyitásakor, a HomePage-el (Figure 12) találkozhatunk, ami bicikliket jelenít meg, valamint bejegyzéseket jelenít meg a hozzájuk tartozó kommentekkel. Ezeket lehet olvasni, illetve egy-egy biciklire rákattintva azok részleteit lehet megtekinteni. Kezdetben mindenki „Guest" (vendég) módban van. Bal oldalt lehet kiválasztani egy előugró menü segítségével a navigációs menüt (Figure 13). [7]

Guset módban rá lehet kattintani a Login vagy Regiszter fülekre, amik a regisztrálást, illetve a bejelentkezést oldják meg. Regisztrálásnál fel van tüntetve, hogy milyen adatokat kell magáról a felhasználó. A bejelentkezésnél e-mailt és jelszót kell megadni, aminek a helyes megadásával jelentkezhet be a felhasználó.

Bejelentkezés után a navigációban két plusz fül jelenik meg, amik a MyBikes, Profile, Cycling Tour, Trips, Services oldalra visznek majd rákattintáskor. A MyBikesban lehetősége van a bejelentkezett felhasználónak a saját biciklijeit megtekinteni, illetve ugyancsak lehetősége van biciklit hozzáadni. Hozzáadáskor egy ablak ugrik fel, amibe be lehet íni a bicikli tulajdonságait, képeket feltölteni, amit ha helyesen tölt ki a felhasználó, akkor a Save gomb megnyomása után az adatok mentésre kerülnek az adatbázisban és láthatóvá válik mindenki számára. Ezen az oldalon lehetősége a bicikli tulajdonosnak a saját biciklije adatainak megváltoztatásához az Edit gombra kattintva, illetve a Delete gombbal törölhető.

A Profile oldalon a saját személyes adataink tekinthetőek meg első sorban. Ezeket lehet itt szerkeszteni, ha szükségét érezzük. Itt megjelennek a saját biciklikjeink, amire, ha rákattintok, ugyancsak a bicikli részletes leírásába visz át. Ez a tranzakcióhoz szolgál, amihez itt lehet a titkos kulcsot megtekinteni, amit egy másik felhasználó tud beírni, ha profilunkra navigálva megkeresi

[7] vö. [7]

az adott biciklit és beírja a kódot. A Profile oldalon lehet ugyancsak megtekinteni a nekünk szánt bejegyzéseket.

A Cycling Tour oldalon a felhasználónak lehetősége van elindítani a GPS követőt túrázás előtt és miután megállította, meg lesz adva a választási lehetőség, hogy el akarja-e menteni ezt az útvonalat. Ha ezt elmentette, következő alkalommal amikor el szeretne menni biciklizni kiválassza az elmentett útvonalat, ami ábrázolva lesz ezúttal a térképen és ezt követheti. Az elmentés mellett, meg is oszthatja ezt az útvonalat más felhasználókkal, akik szintén követhetik az alkalmazás által nyújtott térképen.

A Trips oldalon meg lehet tekinteni különböző bicikli túrákat, amelyeken más kerékpárosok vesznek részt aktuálisan és csatlakozni lehet hozzájuk vagy akár elmenteni azt a bizonyos útvonalat egy későbbi alkalomra.

Amikor a felhasználó a Szerviz oldalra lép, akkor alapértelmezetten találkozhat a hozza legközelebb fekvő szervizekkel, legyen az szakminősített, avagy sem. Ezekre rákattintva megtekintheti azok részletes leírását.

A Szerviz oldalon lehetőséget kap minden felhasználó egy és csakis egy saját szervizt létre hozni, amennyiben szeretne, ami lehet szakminősített vagy nem. Ugyancsak lehetősége van a már meglévő entitás megváltoztatására vagy törlésére.

A navigációban megjelenik a Logout fül, ami még a kijelentkezést szolgálja.

Munkamenet:

A verziókövetést a GitLab segítségével oldottuk meg, valamint ugyanez az eszköz segített a feladatok kezelésében. A GitLab egy webes platform és szolgáltatás, amelyet a szoftvertervezéshez és a forráskód verziókezeléshez használnak. Itt együtt tudtunk dolgozni párhuzamos ágakon (branch-eken), amiket a véglegesítés után egyesítettünk.

A feladatok nyomon követésére ugyanennek a platformnak egy szolgáltatását használtuk, ami az Issue Boards, amiben rövid leírások voltak az egyes feladatokról, azok alfeladatairól, annak határidőjéről, valamint más jelentős részleteiről.

Fejlesztési stratégiánk az Agile volt (sprint, daily, demo, retro rendszer), ami egy inkrementáló fejlesztést tett lehetővé. Nem egyszerre végeztük el az összes fejlesztést, hanem lépésenként haladtunk a kitűzött cél felé.

Adatbázis architektúra:

A backend alapjául a Firebase (Google Cloud) szolgált (Figure 9), amin belül felhasználásra került a Firestore Database, Authentication és Storage. A Firebase egy NoSQL dokumentumalapú adatbázis, amely könnyen skálázható és valós idejű adatok tárolását teszi lehetővé. Az Authentication egy szolgáltatás, ami segít a felhasználók azonosításában és hitelesítésében az alkalmazásban. A Storage nagyméretű fájlok tárolására szolgál, amit mi képtárolásra használtunk.

Az adatok Firestore-ban vannak eltárolva, a bejelentkezést a Firebase Authentication oldja meg, valamint a nagyobb méretű fájlok tárolását a Firebase Storage kezeli.[8]

Az adat ellenőrzést a Zod (Figure 10) végzi, valamint nagy segítséget nyújt az egyes hibák felhasználóbarát kiírásában is. A Zod egy statikus típusellenőrző könyvtár, ami az adatok helyes formaiságát ellenőrzi, valamint a hibakezelést is megkönnyíti.[9]

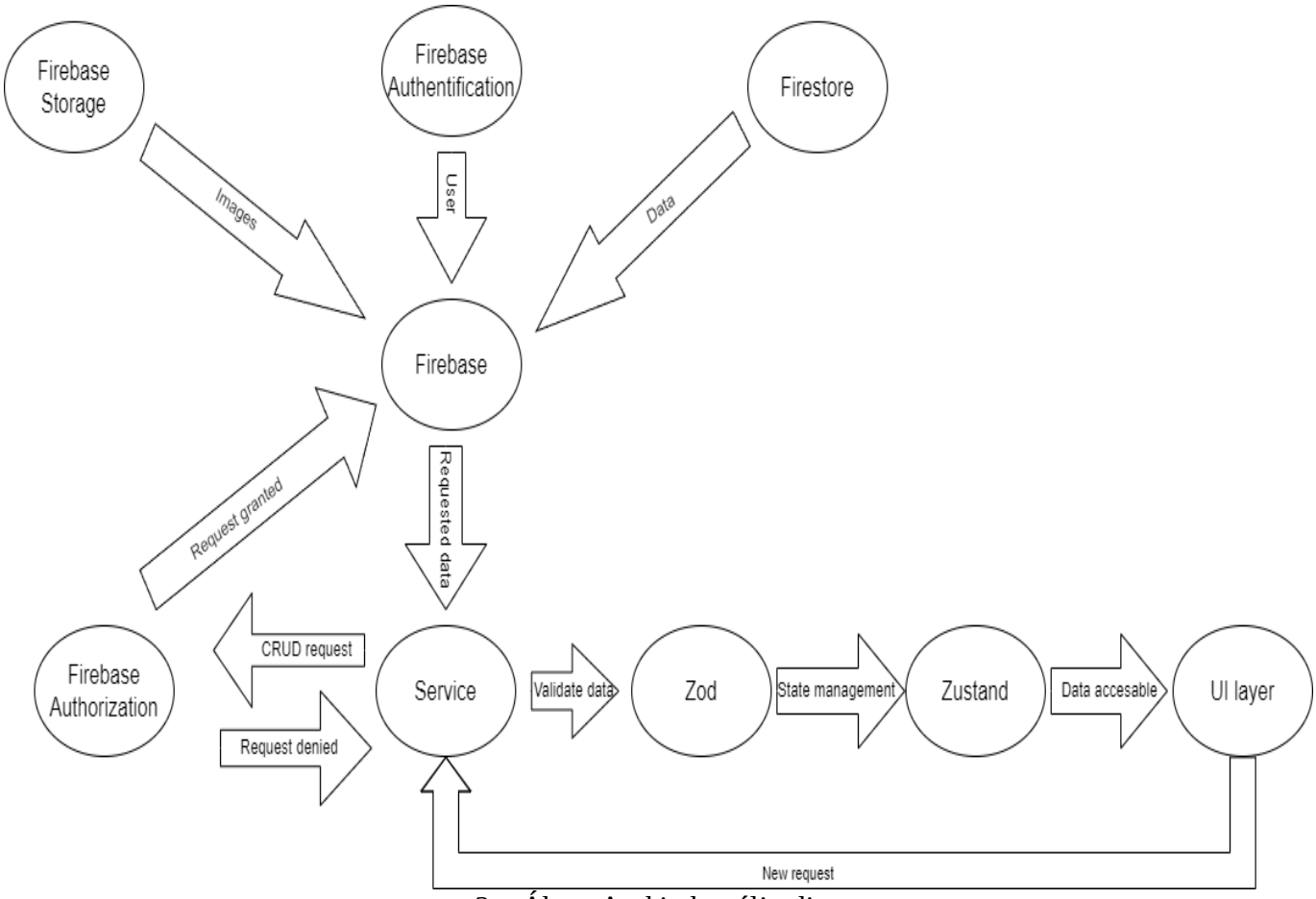
A komponensek közötti adatátvitel a Zustanddal(Figure 11) történik, főleg azokban az esetekben, amikor nem csak két elem között valósul meg ez a folyamat. A Zustand egy állapotkezelő könyvtár a React számára, amely lehetővé teszi az állapot egyszerű és intuitív kezelését.[10]

[8] vö. [3]
[9] vö. [9]
[10] vö. [10]

Architekturális diagram:

Az architekturális diagram a következő formában zajlik le. A UI layerrol meghívjuk az user által elindított folyamathoz tartozó függvényt a servicebol. A service lekéri a szükséges adatokat, a Firebase Authorization le ellenőrzi a user jogosultságait es annak függvényében fogja megkapni a user a lekert adatokat. Három helyről lehet adatokat megkapni, ez lenne a Firebase Storage(kepék lekérésé) , Firebase Authentification(userId lekérésé vagy függvény meghívása) es Firestore ahonnan userekhez, biciklikhez, posztokhoz szóló adatokat lehet lekérni. Lekeres után az adatok a Service-be kerülnek es ahhoz hogy az adatok a UI layer-be kerüljön a következő folyamatokon megy keresztül. Hitelesítjük őket a Zod segítségével es state management

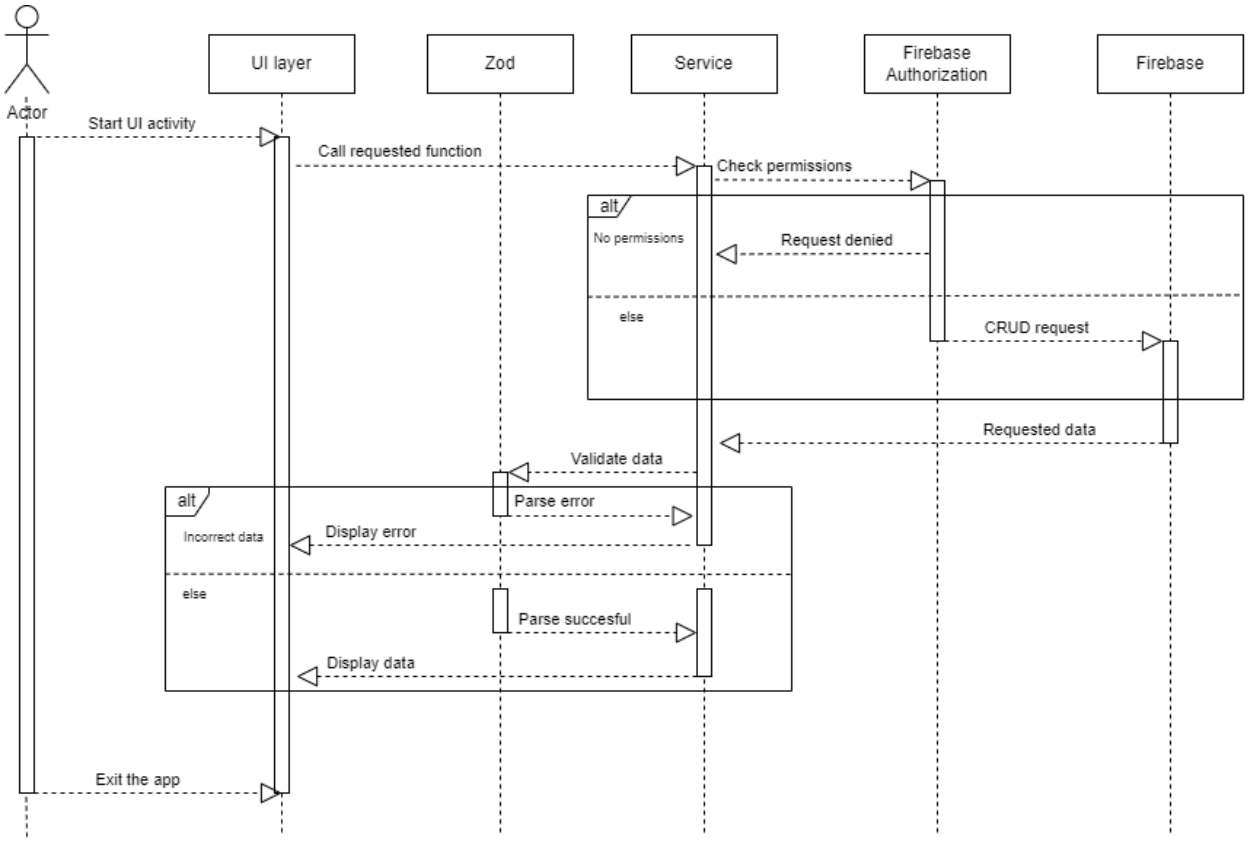
érdekében, a Zustand segítségével elérhetővé tesszük globálisan az adatokat es ezután megjelenítsük a UI layeren.



2. Ábra: Architekturális diagram

Szekvencia diagram:

A user elindít egy folyamatot , ami után a UI layer meghívja a folyamathoz tartozó függvényt a service-ből , mielőtt megtörténné a lekeres az adatbázisból , a Firebase Authorization ellenőrzi ha a usernek megvan az engedélye ahhoz, hogy megkapja a kert adatokat. Ha ezen a rétegen átment a Firebase Authorization életciklusa leáll. Amintán megkapta a service az adatokat, hitelesíteni fogja a Zod segítségével és annak függvényében, hogy sikerült vagy nem ez, megjeleníti az adatokat a UI layer-en. Hitelesítés után a Zod és a service életciklusa leáll, míg a UI layernek az életciklusa csak akkor fog leállni, ha a user kilép az alkalmazásból.



3. Ábra: Szekvencia diagram

Fejlesztési lehetőségek:

Egy optimalizálási lehetőség lenne a képek újra méretezése/tömörítése, mert jelenleg nem történik semmi fajta helyoptimalizálás, ami nagy mértékben javítana az alkalmazás minőségén, fenntarthatóságán, karbantarthatóságán.

Egy kölcsönzési rendszer kialakítása, amely lehetőséget ad a felhasználóknak, hogy biciklit béreljenek másoktól, ami sokban növelné az alkalmazás látogatottságát és egyben egy új perspektívát is nyitna a felhasználóknak. Ezzel újabb bevonzási területtel bővülne az alkalmazás.

Cikkek megjelenítése az alkalmazás felületén különböző bicikli túra rendezvényekről a felhasználók számára. Az egymás közötti tapasztalat és szemcsere mellett, beszédtéma lehetne így az aktuális kerekpáros hírek megtekintése, aminek eredményeképp a felhasználók több időt tölthetnének a felületen.

Bicikli túra közben lehetővé tenni hogy az alkalmazás pulzust mérjen, figyelembe vegye hogy hány kalóriát égetett a felhasználó, meg egyéb egészségügyi részletek megjelenítése. Segítséget nyújtana az applikáció az egészséges életmód követésében, ami manapság nagyon divatossá nőtte ki magát.

Összefoglaló:

A megvalósítás központi részét a FireBase, a TypeScript és React Native alkotják. A fejlesztés cross-platformon történt, de a kitelepítés csak Androidra valósult meg.

Röviden ez a telefonos applikáció egy felhő alapú alkalmazás, ami biciklik böngészését, cseréjét, szervizelését, a vele történt események nyomon követését valósítja meg. Ezek mellett lehetőséget teremt a felhasználók közötti információ cserére kommentek, posztok, felhasználói profilra rákeresés formájában, valamit igénybe vehetik a valós idejű chat funkciót is, ami nagyban növeli a kommunikációs lehetőségeket.

A felhasználó túra útvonalakat rögzíthet, oszthat meg, amiket a későbbiekben megtekinthet, megoszthat másokkal. Ez mellett biciklis túrákat szervezhet, ezeken részt vehet, amennyiben szeretne. Nagy segítség a szerviz opció, ami által az egy biciklihez tartozó elvárások és szükségletek egy nagy része ki van elégítve.

Bibliográfia:

[1] Expo Go: https://expo.dev/client 2023.12. 19.

[2] Expo Application Services: https://docs.expo.dev/eas/ 2023.12.19

[3] FireBase: https://firebase.google.com/ 2023.12.19.

[4] Gluestack: https://gluestack.io/ 2023.12.19

[5] Native Base: https://nativebase.io/ 2023.12.19.

[6] React: https://legacy.reactjs.org/ 2023. 12. 19.

[7] React Navigation: https://reactnavigation.org/docs/getting-started/ 2023. 12. 19.

[8] React native: https://reactnative.dev/ 2023. 12. 19.

[9] Zod: https://zod.dev/ 2023.12.19.

[10] ZuStand: https://docs.pmnd.rs/zustand/getting-started/introduction 2023.12.19.

Függelék:

Figure 1:

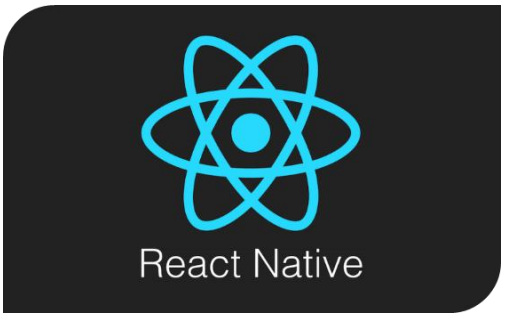


Figure 2:

Figure 3:

Figure 4:

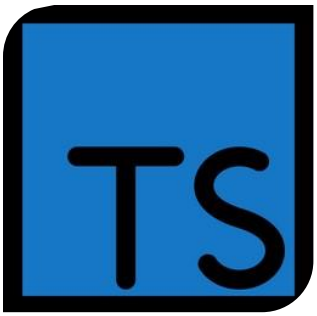


Figure 5:

Figure 6:

Figure 7:

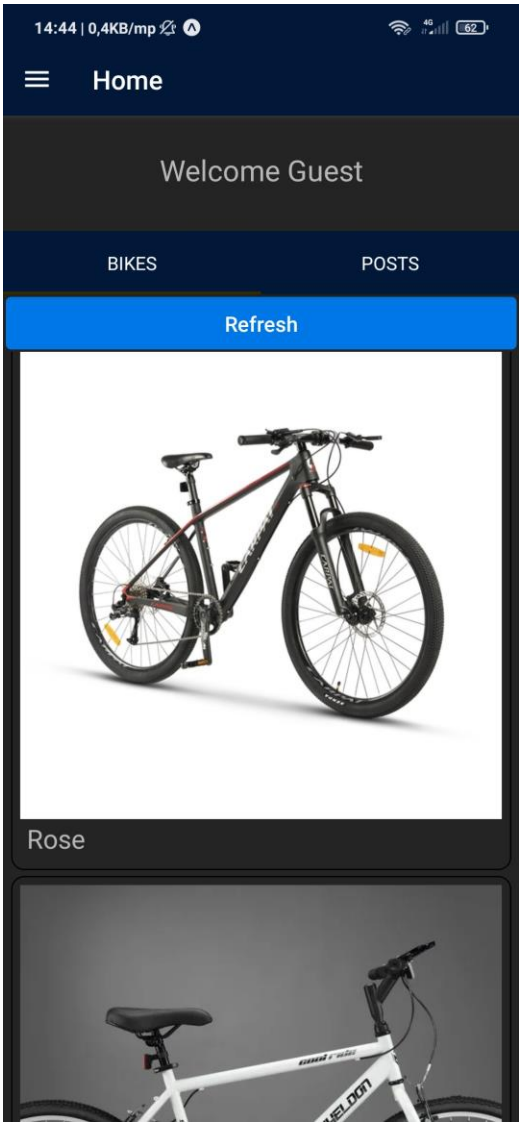
Figure 8:

Figure 9:

Figure 10:

Figure 11:

Figure 12:



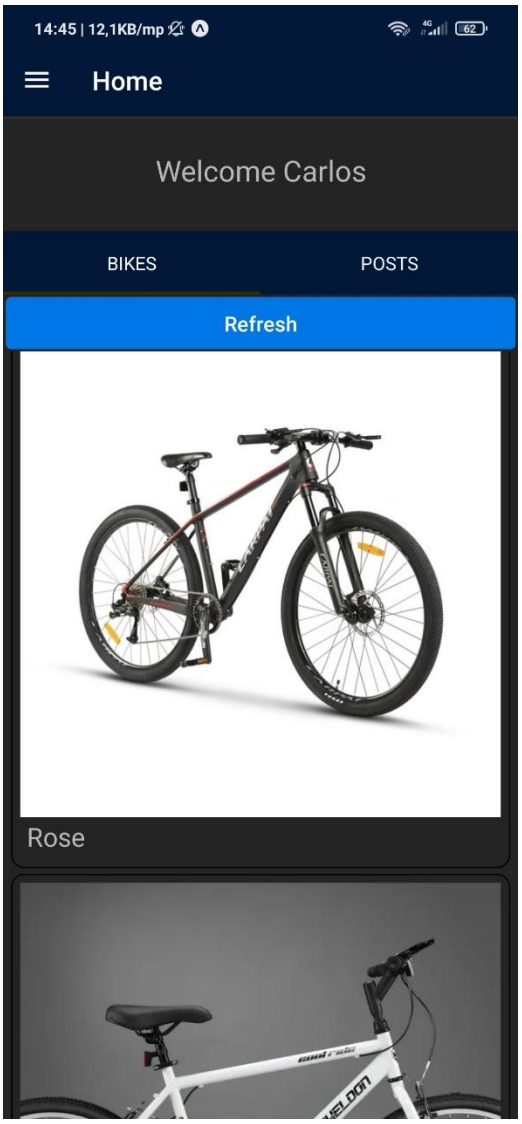
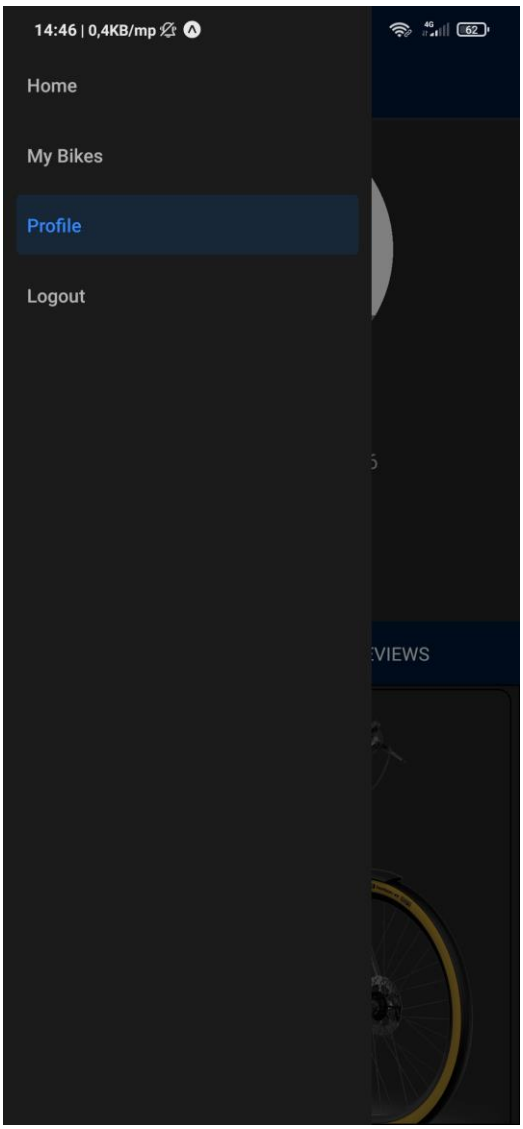


Figure 13


14:46 | 0,4KB/mp
Home
My Bikes
Profile
Logout
EVIEWS

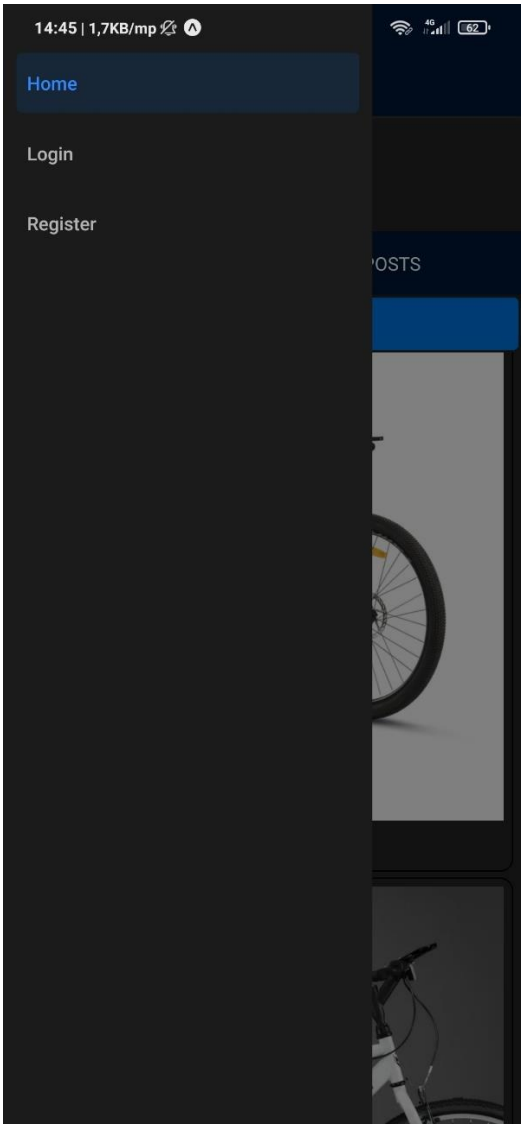

14:45 | 1,7KB/mp
Home
Login
Register
OSTS

Figure 14:

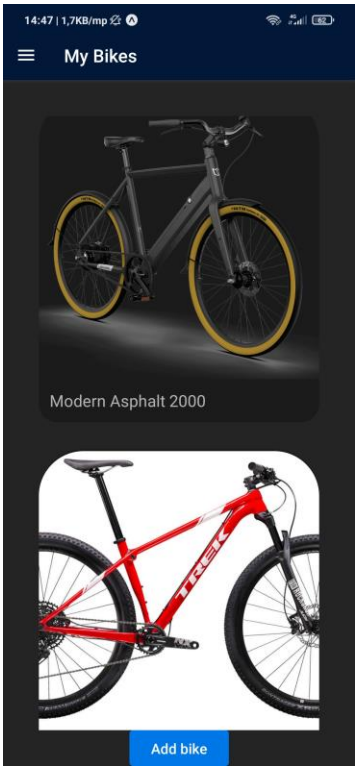


Figure 15: