**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Проект 2 з дисципліни

**«Розробка програмного забезпечення під мобільні платформи»**

на тему:

**«Покращення додатка для пошуку ресторанів за гео-позицією»**

**Виконав**:

студент 4 курсу ОС «Бакалавр»

спеціальності «Компʼютерні науки»,

освітньої програми «Інформатика»,

групи ТТП-42

*Маєвський Любомир Андрійович*

**Київ-2022**

ЗМІСТ

[ОПИС ЗАВДАННЯ](#_gjdgxs) 3

[КОРОТКИЙ ВИКЛАД ТЕОРІЇ](#_30j0zll) 4

[ОПИС РОБОТИ З ПРОГРАМОЮ ТА ЗНІМКИ ЕКРАНУ](#_3znysh7) 7

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ](#_2et92p0) 11

[ДОДАТКИ](#_tyjcwt) 12

[Додаток А](#_3dy6vkm) Фрагмент Коду, Карта 12

[Додаток Б](#_1t3h5sf) Фрагмент Коду, Пошук Ресторанів на карті 13

[Додаток В](#_4d34og8) Фрагмент Коду, Авторизація 15

[Додаток Г](#_2s8eyo1) Фрагмент Коду, Організація звязуваня ресторанів та гео-позиції 16

# ОПИС ЗАВДАННЯ

Завдання полягає у розробці додатку, яку якому буде можливість пошуку ресторанів за їх гео-позицію:

1. Ввід:

* Назва геопозиції на карті

1. Вивід:

* Карта гео-позиції
* Маркери на карті, які позначають ресторона
* Маркери є інтерактивними, нажимаючи буде вилазити pop-up меню і картинкою та назвою ресторану. Нажимаючи ще раз, робиться redirect на сторінку магазину у додатку

1. Робота з онлайн базою даних (Firebase):

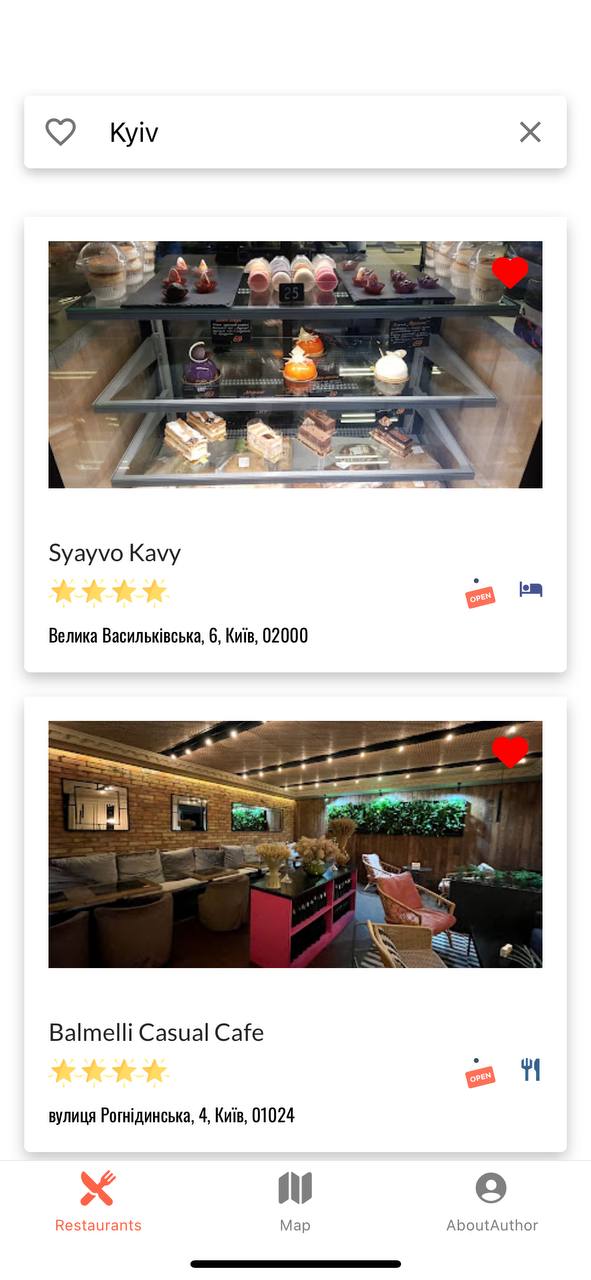
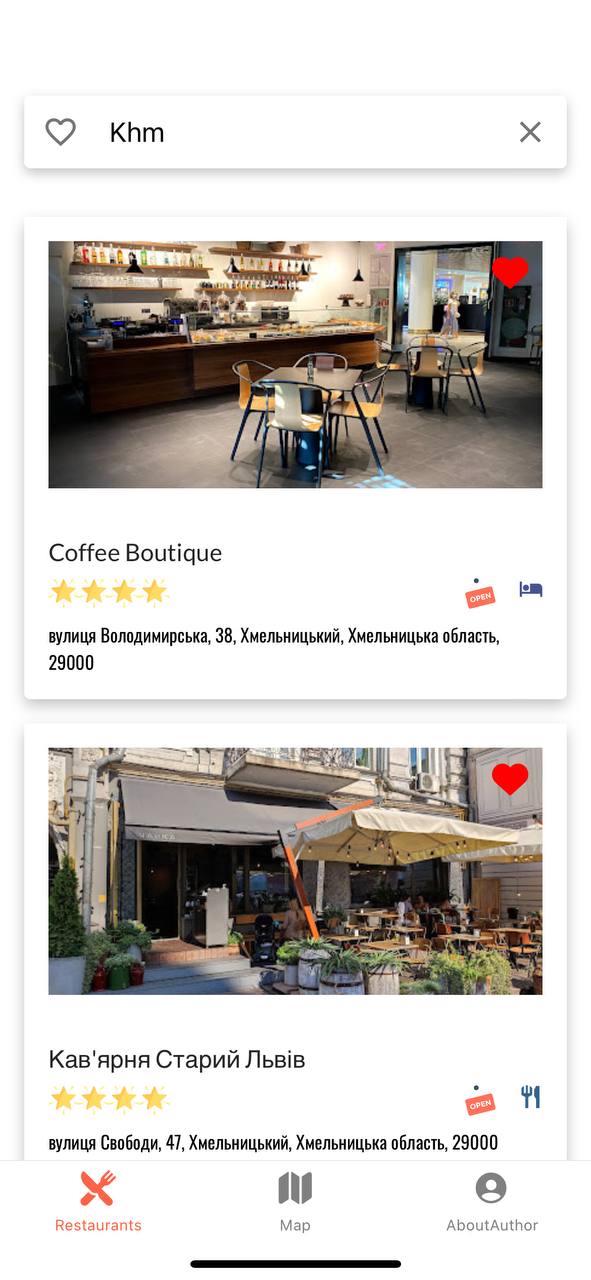
* Регістрація у додатку, із збереженням усіх даних про користувача у базі даних
* Авторизація у додатку, із провіркою коректності введених даних користувачем

# КОРОТКИЙ ВИКЛАД ТЕОРІЇ

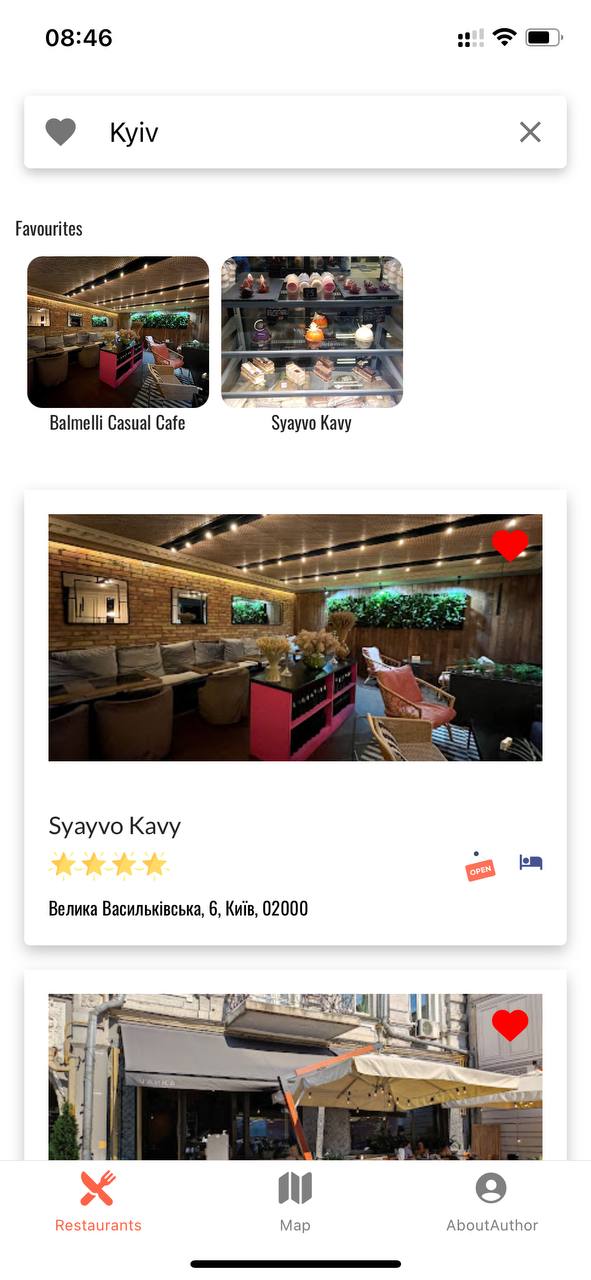
Firebase – це платформа для розробки мобільних додатків з величезним функціоналом. Починалася вона як стартап, а сьогодні її використовують при розробці кращих кроссплатформних додатків. Головна перевага платформи в тому, що вона дозволяє розробнику не відволікатися на створення backend, тобто прихованої від користувача програмної частини проєкту, наприклад, серверного коду. І це спрощує і прискорює створення мобільних додатків, дає можливість повністю зосередитися саме на UX / UI, тобто, на призначеному для користувача інтерфейсі і досвіді.  
  
Firebase – це одне з BaaS-рішень (Backend as a Service), яке дає розробнику масу можливостей.Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Так, Firebase Realtime Database надає розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх в хмарному сховищі. Додаток підключається до бази даних через WebSocket, який відповідає за синхронізацію даних протягом усього сеансу.

# ОПИС РОБОТИ З ПРОГРАМОЮ ТА ЗНІМКИ ЕКРАНУ

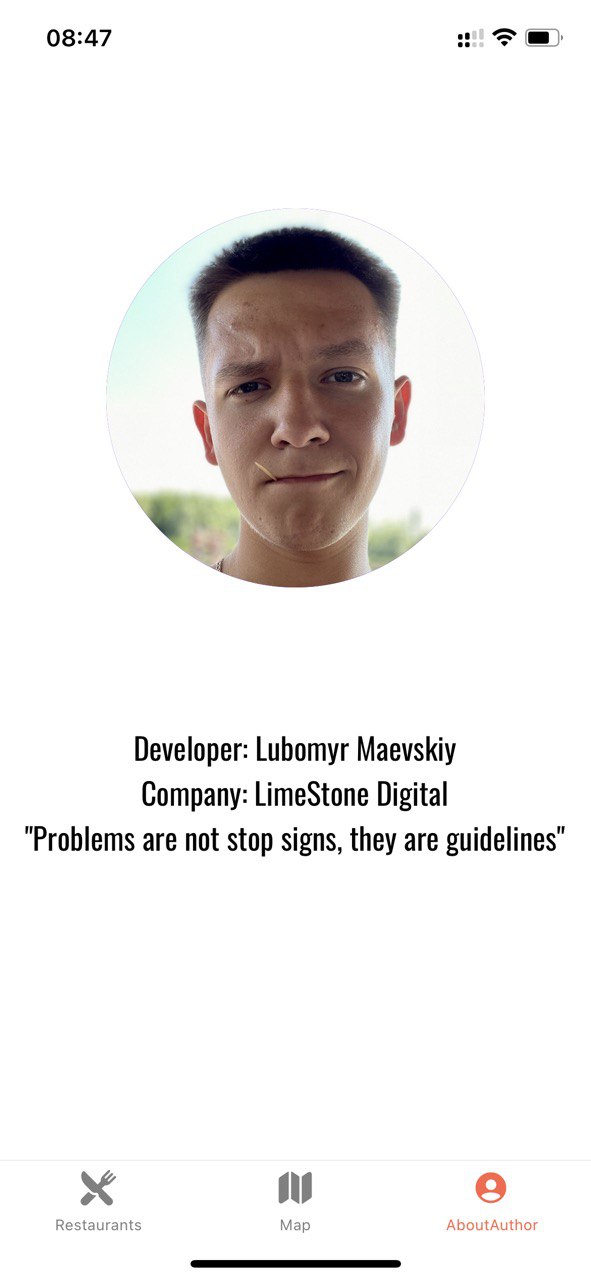
1. Початковий екран – пошук ресторанів за гео-позиціює. Навігаційна панель знаходиться у низу екрану, що є звичним для iOS.

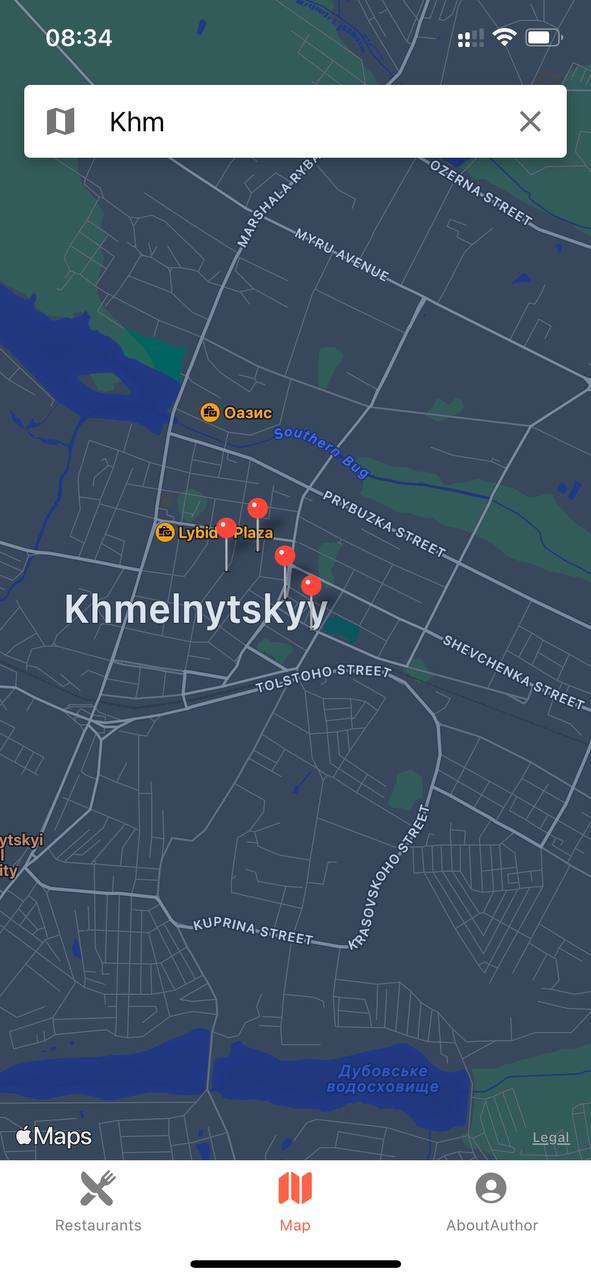
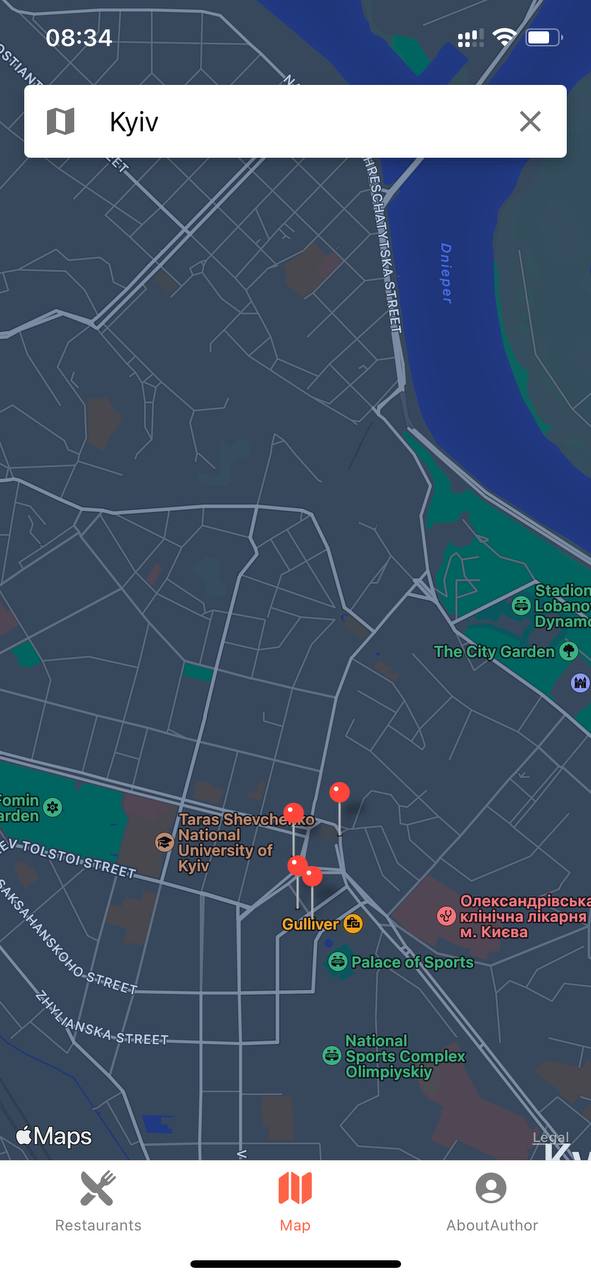


1. Якщо нажати на кнопку “Сердечко” на ресторані, то обраний ресторан додасться до улюблених. Якщо нажати на кнопку “Сердечко” напроти пошуку, то відкриється панель із улюбленими ресторанами.

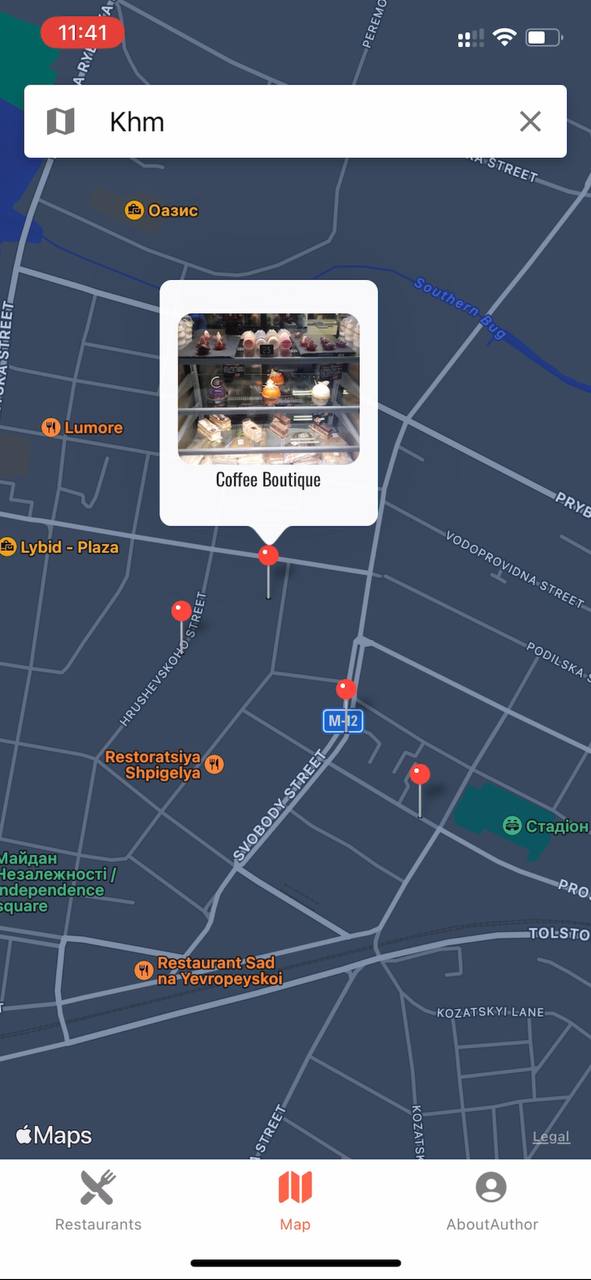


1. Екран з даними про виконавця.

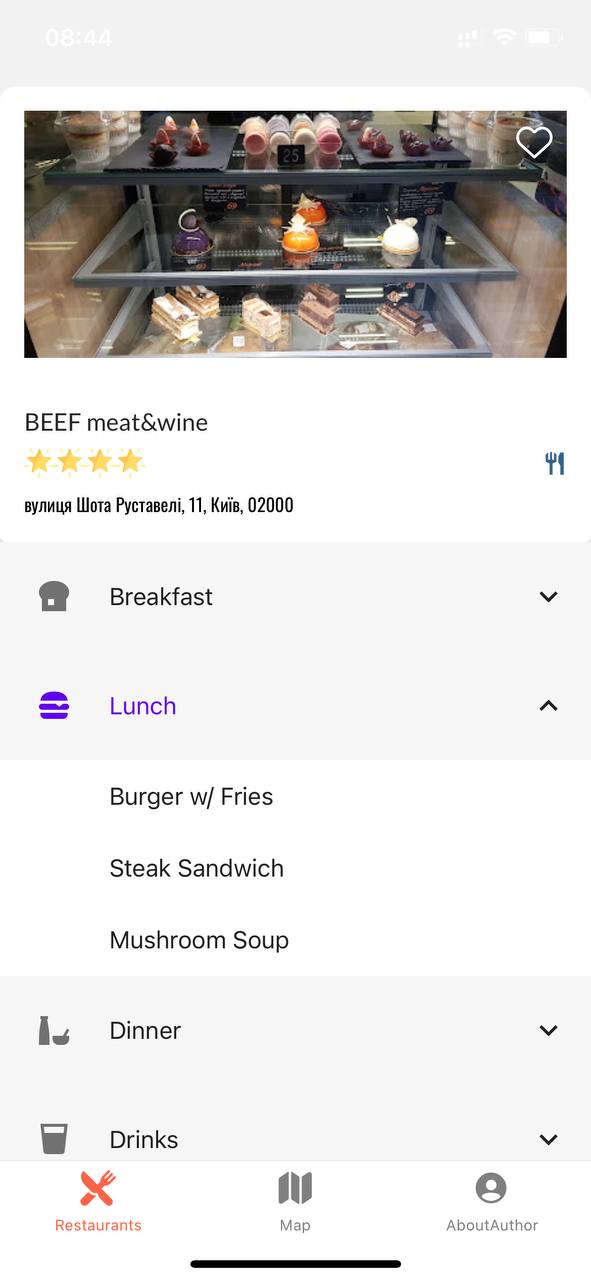


1. Екран із пошуком ресторана за гео-позицією на карті

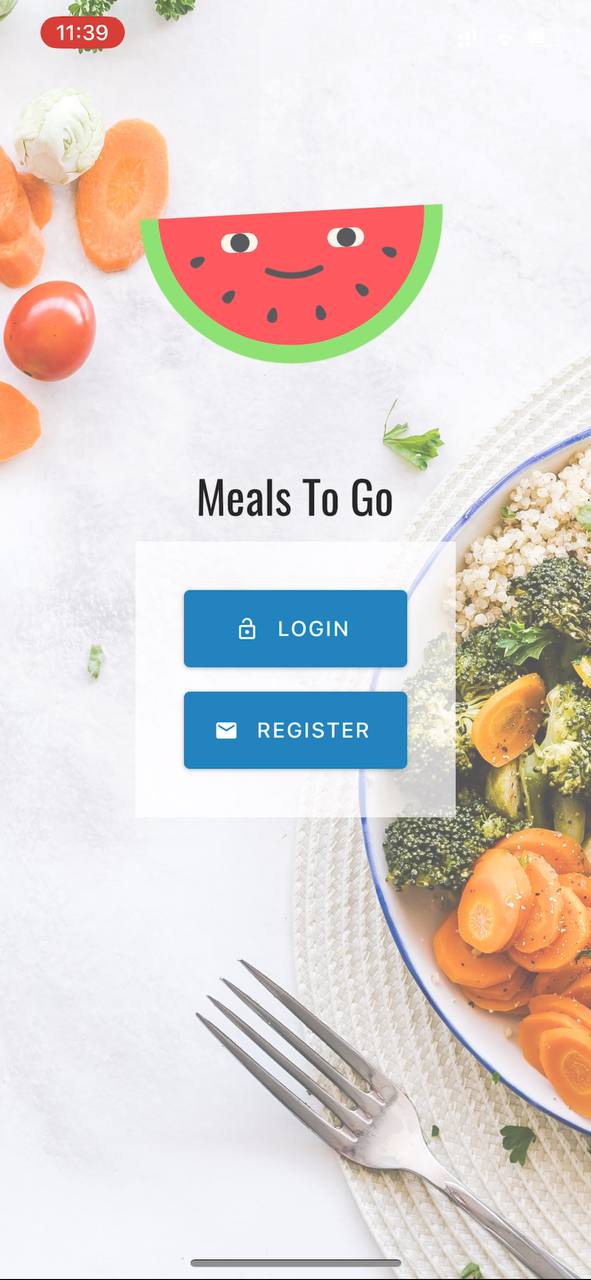
1. Pop-up із інформацією про ресторан на карті

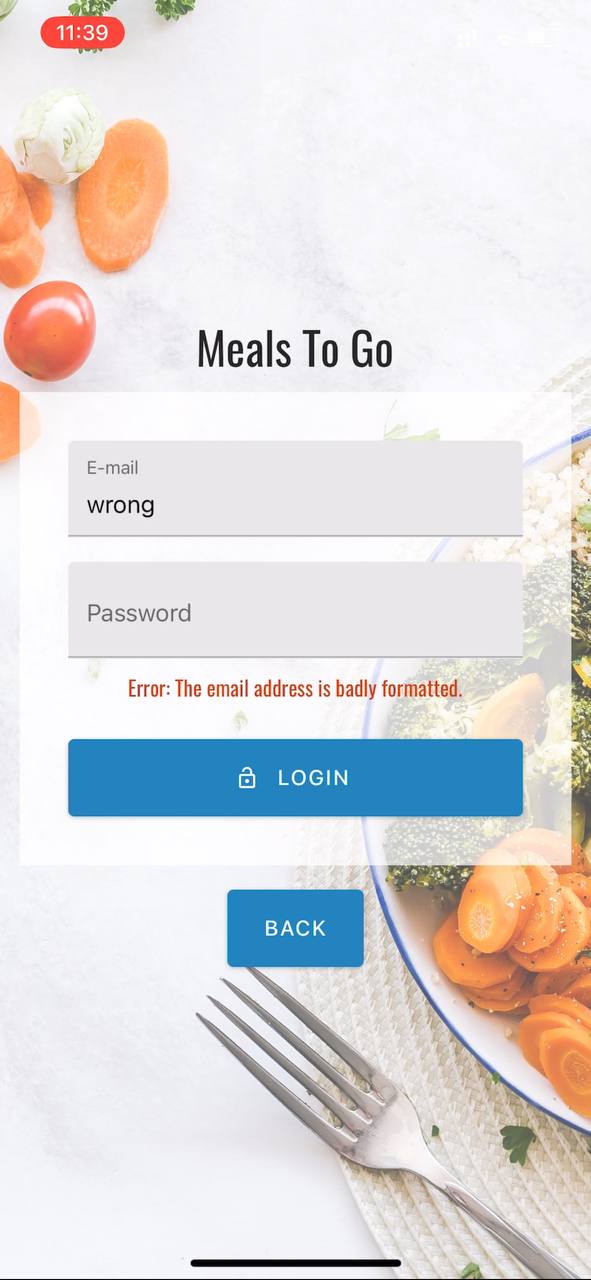


1. Інформаційна сторінка ресторану



1. Екран авторизації/регістрації користувача





1. Екран загрузки додатку



# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is React Native And When to Use It For Your App? (Updated). *elpassion.com*. URL: <https://www.elpassion.com/blog/what-is-react-native-and-when-to-use-it> (дата звернення: 22.10.2022).
2. What is Firebase URL: https://avada-media.ua/ua/services/firebase/

# ДОДАТКИ

## ДОДАТОК А

Фрагмент Коду, Карта

import React, { useContext, useState, useEffect } from "react";

import MapView from "react-native-maps";

import styled from "styled-components/native";

import { LocationContext } from "../../../services/location/location.context";

import { RestaurantsContext } from "../../../services/restaurants/restaurants.context";

import { Search } from "../components/search.component";

import { MapCallout } from "../components/map-callout.component";

const Map = styled(MapView)`

  height: 100%;

  width: 100%;

`;

export const MapScreen = ({ navigation }) => {

  const { location } = useContext(LocationContext);

  const { restaurants = [] } = useContext(RestaurantsContext);

  const [latDelta, setLatDelta] = useState(0);

  const { lat, lng, viewport } = location;

  useEffect(() => {

    const northeastLat = viewport.northeast.lat;

    const southwestLat = viewport.southwest.lat;

    setLatDelta(northeastLat - southwestLat);

  }, [location, viewport]);

  return (

    <>

      <Search />

      <Map

        region={{

          latitude: lat,

          longitude: lng,

          latitudeDelta: latDelta,

          longitudeDelta: 0.02,

        }}

      >

        {restaurants.map((restaurant) => {

          return (

            <MapView.Marker

              key={restaurant.name}

              title={restaurant.name}

              coordinate={{

                latitude: restaurant.geometry.location.lat,

                longitude: restaurant.geometry.location.lng,

              }}

            >

              <MapView.Callout

                onPress={() =>

                  navigation.navigate("RestaurantDetail", {

                    restaurant,

                  })

                }

              >

                <MapCallout restaurant={restaurant} />

              </MapView.Callout>

            </MapView.Marker>

          );

        })}

      </Map>

    </>

  );

};

## ДОДАТОК Б

Фрагмент Коду, Пошук Ресторанів на карті

import React, { useContext, useState, useEffect } from "react";

import styled from "styled-components/native";

import { Searchbar } from "react-native-paper";

import { LocationContext } from "../../../services/location/location.context";

const SearchContainer = styled.View`

  padding: ${(props) => props.theme.space[3]};

  position: absolute;

  z-index: 999;

  top: 40px;

  width: 100%;

`;

export const Search = () => {

  const { keyword, search } = useContext(LocationContext);

  const [searchKeyword, setSearchKeyword] = useState(keyword);

  useEffect(() => {

    setSearchKeyword(keyword);

  }, [keyword]);

  return (

    <SearchContainer>

      <Searchbar

        placeholder="Search for a location"

        icon="map"

        value={searchKeyword}

        onSubmitEditing={() => {

          search(searchKeyword);

        }}

        onChangeText={(text) => {

          setSearchKeyword(text);

        }}

      />

    </SearchContainer>

  );

};

## ДОДАТОК В

Фрагмент Коду, Авторизація

import React, { useState, createContext } from "react";

import \* as firebase from "firebase";

import { loginRequest } from "./authentication.service";

export const AuthenticationContext = createContext();

export const AuthenticationContextProvider = ({ children }) => {

  const [isLoading, setIsLoading] = useState(false);

  const [user, setUser] = useState(null);

  const [error, setError] = useState(null);

  firebase.auth().onAuthStateChanged((usr) => {

    if (usr) {

      setUser(usr);

      setIsLoading(false);

    } else {

      setIsLoading(false);

    }

  });

  const onLogin = (email, password) => {

    setIsLoading(true);

    loginRequest(email, password)

      .then((u) => {

        setUser(u);

        setIsLoading(false);

      })

      .catch((e) => {

        setIsLoading(false);

        setError(e.toString());

      });

  };

  const onRegister = (email, password, repeatedPassword) => {

    setIsLoading(true);

    if (password !== repeatedPassword) {

      setError("Error: Passwords do not match");

      return;

    }

    firebase

      .auth()

      .createUserWithEmailAndPassword(email, password)

      .then((u) => {

        setUser(u);

        setIsLoading(false);

      })

      .catch((e) => {

        setIsLoading(false);

        setError(e.toString());

      });

  };

  const onLogout = () => {

    firebase

      .auth()

      .signOut()

      .then(() => {

        setUser(null);

        setError(null);

      });

  };

  return (

    <AuthenticationContext.Provider

      value={{

        isAuthenticated: !!user,

        user,

        isLoading,

        error,

        onLogin,

        onRegister,

        onLogout,

      }}

    >

      {children}

    </AuthenticationContext.Provider>

  );

};

## ДОДАТОК Г

Фрагмент Коду, Організація звязуваня ресторанів та знайденої гео-позиції

import React, { useState, useEffect } from "react";

import { locationRequest, locationTransform } from "./location.service";

export const LocationContext = React.createContext();

export const LocationContextProvider = ({ children }) => {

  const [keyword, setKeyword] = useState("Kyiv");

  const [location, setLocation] = useState(null);

  const [isLoading, setIsLoading] = useState(false);

  const [error, setError] = useState(null);

  const onSearch = (searchKeyword) => {

    setIsLoading(true);

    setKeyword(searchKeyword);

  };

  useEffect(() => {

    if (!keyword.length) {

      return;

    }

    locationRequest(keyword.toLowerCase())

      .then(locationTransform)

      .then((result) => {

        setIsLoading(false);

        setLocation(result);

      })

      .catch((err) => {

        setIsLoading(false);

        setError(err);

      });

  }, [keyword]);

  return (

    <LocationContext.Provider

      value={{

        isLoading,

        error,

        location,

        search: onSearch,

        keyword,

      }}

    >

      {children}

    </LocationContext.Provider>

  );

};