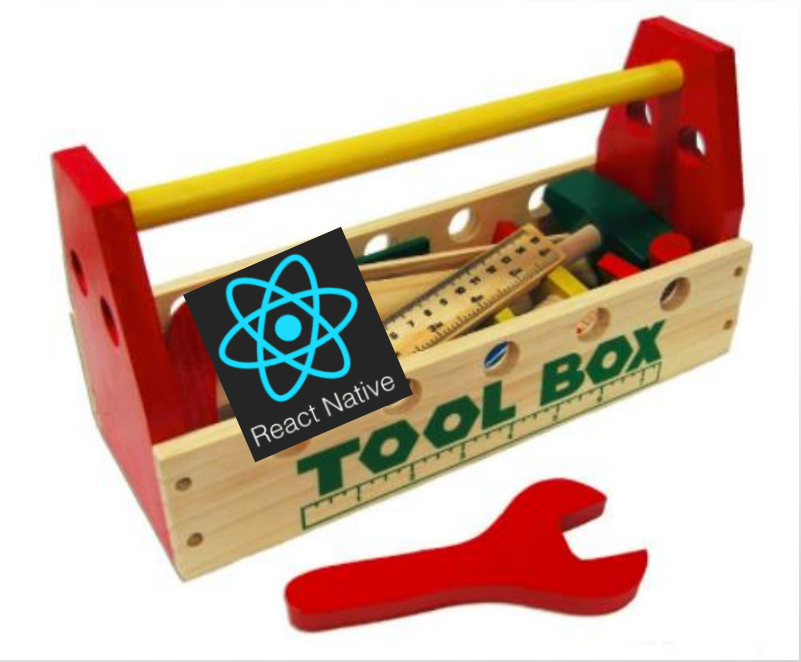


REACT NATIVE TOOLBOX

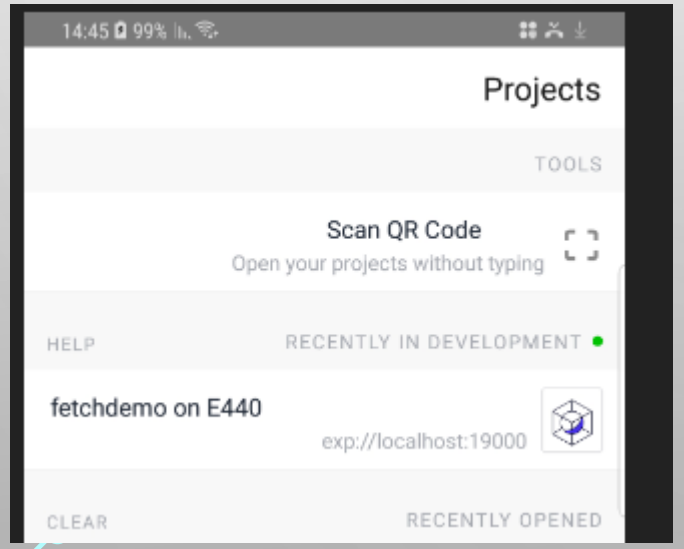
01 – 18

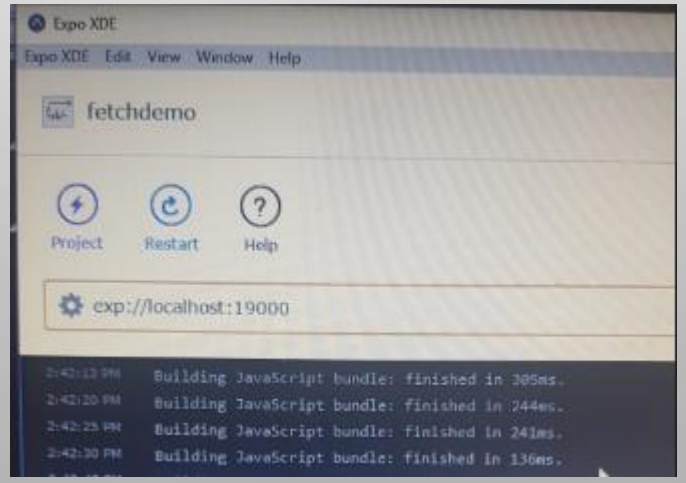
SYLLABUS

- **-01 Expo**
- 00 Debug
- 01 APK Creation and google play
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation v6
- 07 Image Upload
- 08 Push Notification
- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib
- 15 Linking
- 17-1 Maps
- 17 Fingerprint authentication
- 18 Async Storage
- 19 Images

EXPO

1. הכי מהיר זה לעבוד ב-LOCALHOST כאשר המכשיר וגם המחשב מחוברים על אותה רשת WIFI.
2. ניתן לחבר את המכשיר למחשב עם כבל USB ולאפשר DEBUGGING ("איתור באגים של USB") דרך הUSB ב"אפשרויות למפתחים"

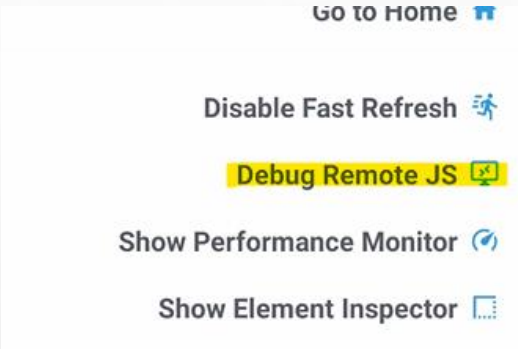




SYLLABUS

- **00 Debug**
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

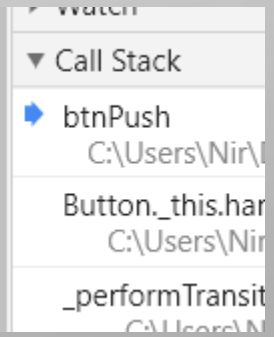
DEBUG



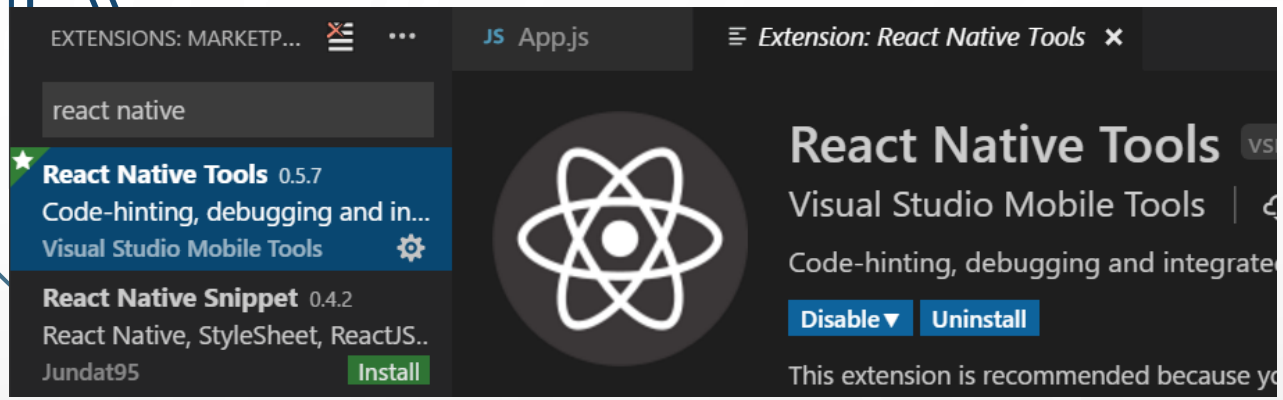
1. לשקשק\לנענע\לנער את המכשיר ואז לבחור בתפריט את "Debug JS Remotely"
2. יפתח הכרום בצורה אטומטית ואז יש ללחוץ F12
3. יש לבחור את הטאב של Sources
4. כאשר נגיע לBREAKPOINT הקוד יעצר בשורה המתאימה ואז אפשר לדאבג רגיל.

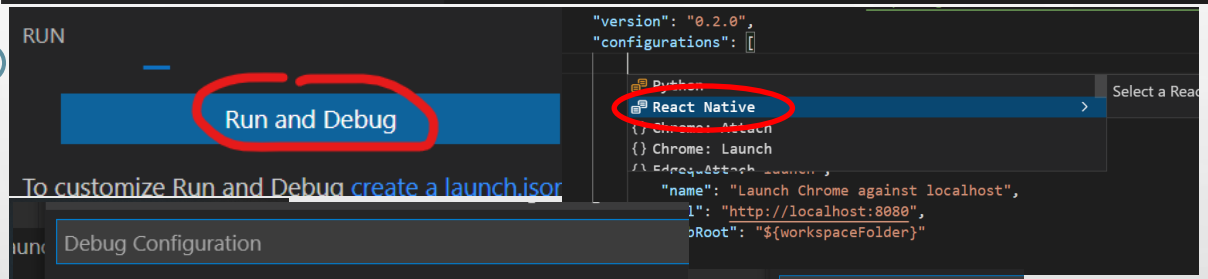
```
function btnPush() {
    debugger;
    console.log(1);
    let num=7;
```

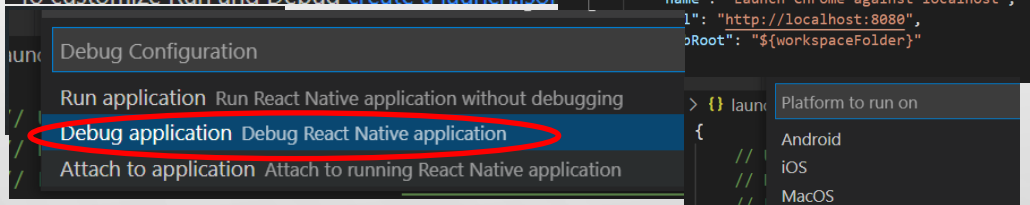
```
10  function btnPush() {
11    debugger;
12    console.log(1);
13    let num=7;
14    console.log(num);
15  }
16
17  function txtName(n, e) {
```

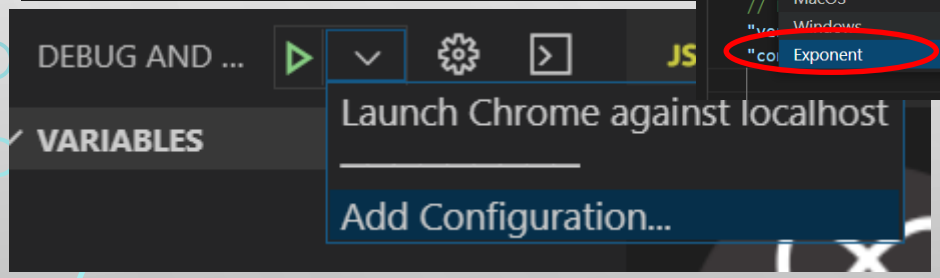


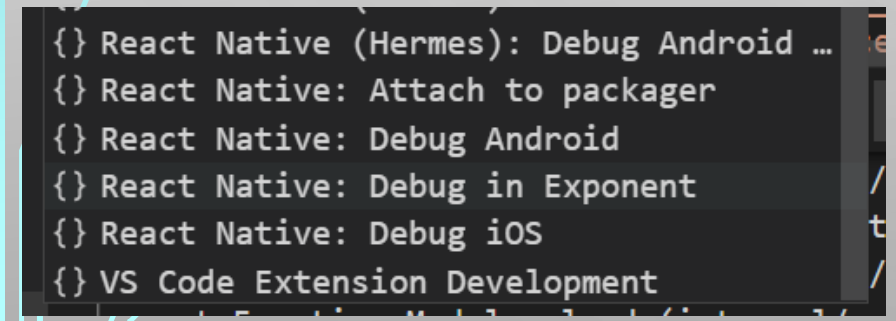
DEBUG IN VS CODE

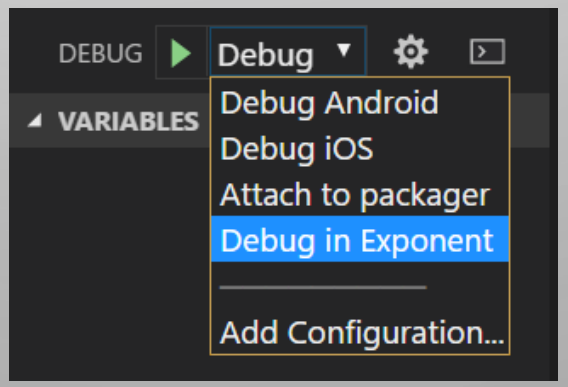




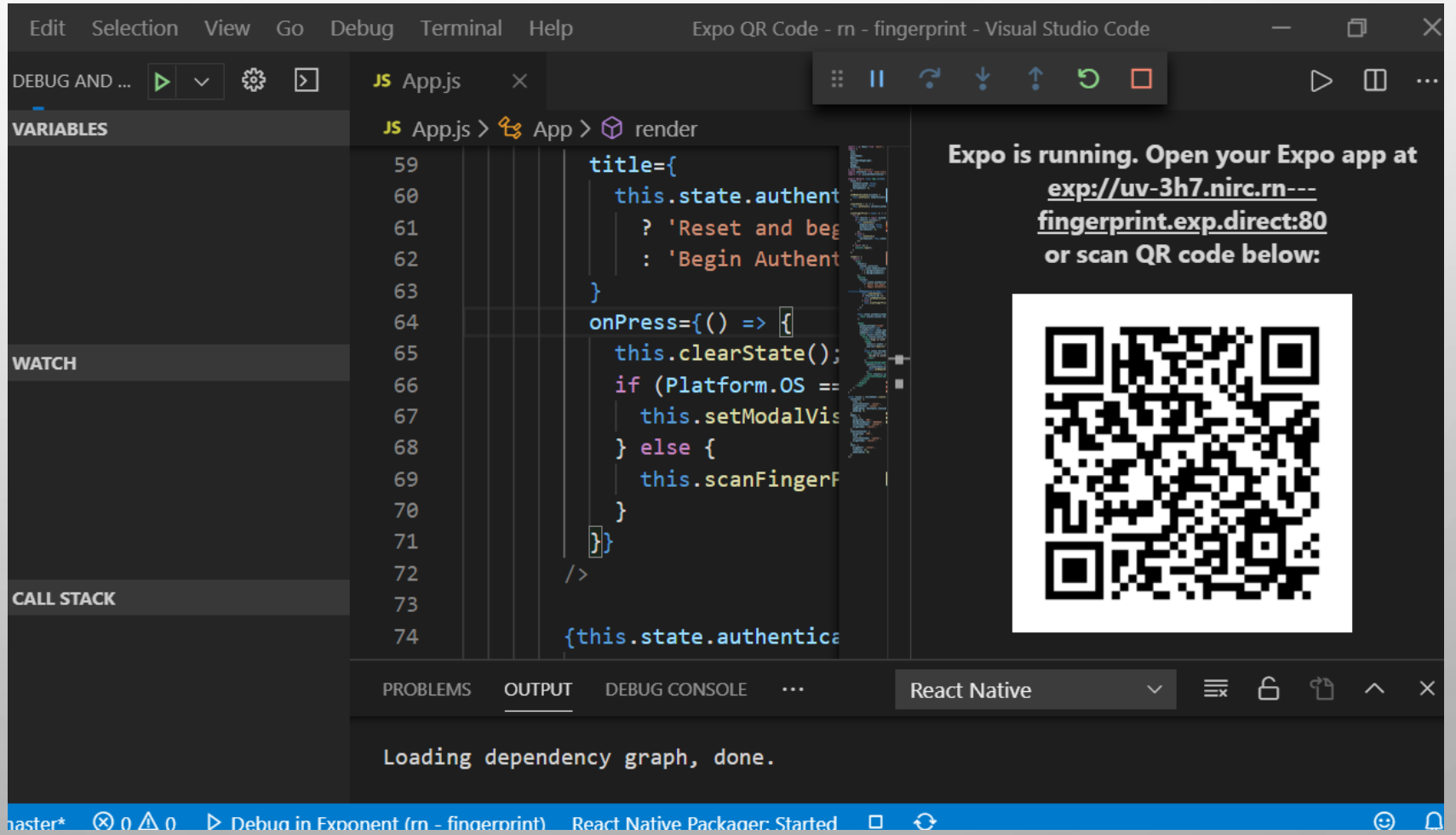






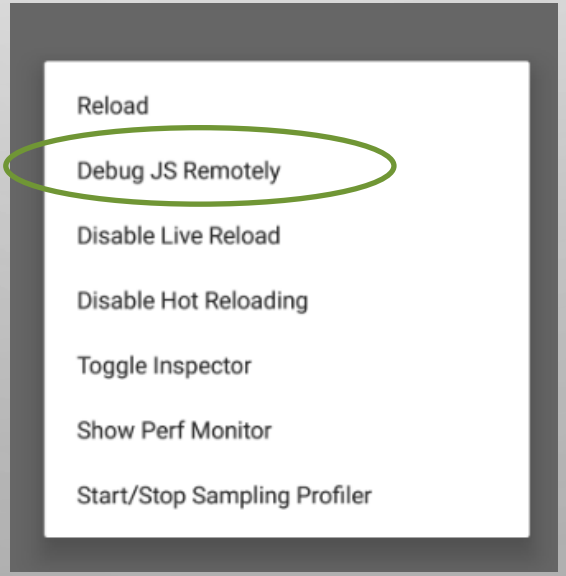


- https://github.com/Microsoft/vscode-react-native/blob/master/doc/expo.md
- ניתן להתקין תוסף שמאפשר לדאבג במקום בכרום בתוך ה- visual code עצמו.
- להתקין את התוסף react native tools
- יתכן ותתבקשו להכניס שם וסיסמא של EXPO
- לבחור


Edit Selection View Go Debug Terminal Help
Expo QR Code - rn - fingerprint - Visual Studio Code
DEBUG AND ...
App.js
VARIABLES
WATCH
CALL STACK
App.js > App > render
title={
this.state.authent
? 'Reset and beg
: 'Begin Authent
}
onPress={() => {
this.clearState();
if (Platform.OS ==
this.setModalVis
} else {
this.scanFingerP
}
}}
/>
{this.state.authentica
Expo is running. Open your Expo app at
exp://uv-3h7.nirc.rn---fingerprint.exp.direct:80
or scan QR code below:
PROBLEMS
OUTPUT
DEBUG CONSOLE
React Native
Loading dependency graph, done.
Debug in Exponent (rn - fingerprint)
React Native Packager: Started

DEBUG IN VS CODE

- חייב להיות שרת אחד רגיל שרץ ברקע!
- לחכות למסך שייציג QR לסריקה – לוקח זמן, סבלנות☺
- ברגע שמסתיים – לשקשק את המכשיר ולבחור את האופציה " Debug JS Remotely"
- לדאבג☺



SYLLABUS

- 00 Debug
- **01 APK Creation and google play**
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

APK CREATION AND GOOGLE PLAY

אם רוצים את האפליקציה על המכשיר אבל בתוך האפליקציה של EXPO ניתן רק לעשות PUBLISH .

- אם רוצים כאפליקציה נפרדת יש לייצר APK

- https://www.youtube.com/watch?v=IlveG3Qp0no

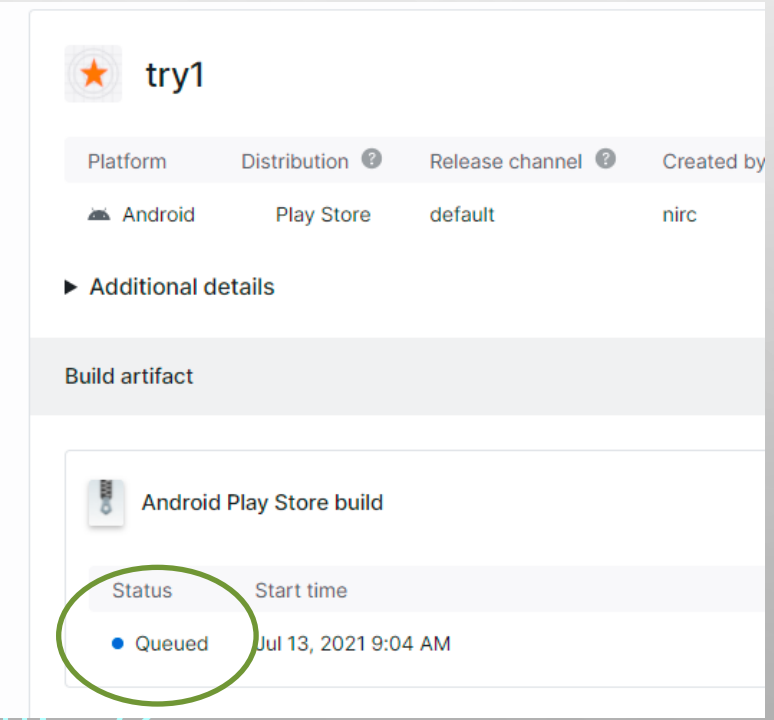
1. להתקין

```
npm install -g exp-cli
```

APK CREATION CONT'

```
PS C:\Users\Nir\Desktop\try1> expo build:android
? Choose the build type you would like: » - Use arr
>   apk
    Build a package to deploy to the store or insta
    app-bundle
```

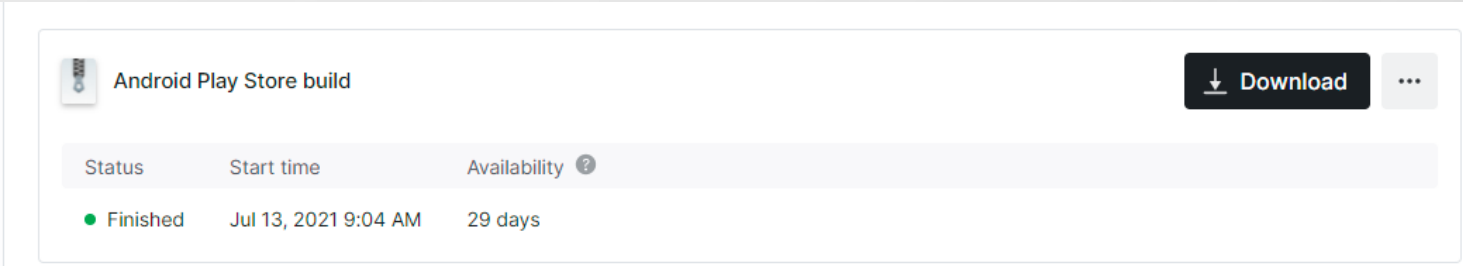
```
You can monitor the build at

 https://expo.io/accounts/nirc/projects/try1/builds/08382cb3-c95f
053bd6

Waiting for build to complete.
You can press Ctrl+C to exit. It won't cancel the build, you'll b
```



2. expo login
3. npm start
4. לאחר הרצה לסגור את השרת ע"י Ctrl+c
5. expo build:android
6. לוקח הרבה זמן ~יותר מעשר דקות
7. ניתן לראות את הסטטוס ע"י exp build:status וניתן גם לראות את הסטטוס דרך האתר שלהם ע"י הלינק שמקבלים.

APK CREATION CONT'

8. ברגע שיסתיים (לוקח הרבה זמן...) תוכלו להוריד מהאתר APK!



9. נותר להתקין על הטלפון

a) פשוט לחבר את הטלפון בUSB, להעתיק ל DOWNLOADS ואז בטלפון ללחוץ עליו ולהתקין. תצטרכו לאשר התקנה של צד שלישי.

b) ע"י למשל התוכנה יעודיות של התקנת APK.

UPLOAD TO GOOGLE PLAY

- בסרטון הבא ניתן לראות כיצד להעלות את ה APK שיצרתם לחנות של גוגל
- https://www.youtube.com/watch?v=7K6l3Jxi6SE

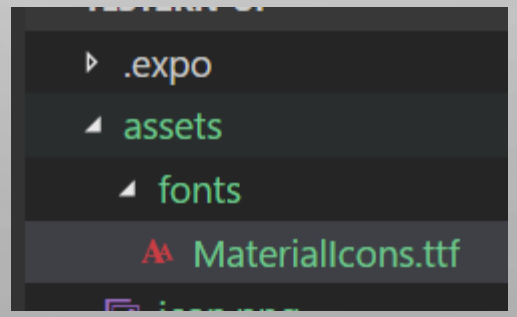
SYLLABUS

- 00 Debug
- 01 APK Creation
- **02 React Native Material UI**
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

REACT NATIVE MATERIAL UI

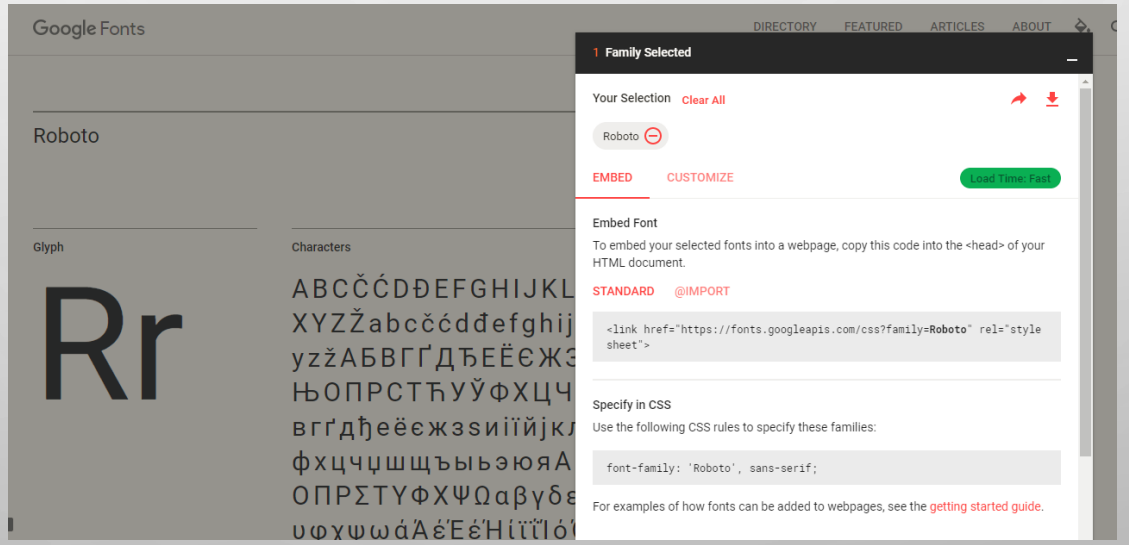
- https://www.npmjs.com/package/react-native-material-ui
- npm **install** react-native-material-ui –save
- npm i react-native-cli
- react-native link react-native-vector-icons
- copy from : ./node_modules/react-native-vector-icons/Fonts/MaterialIcons.ttf

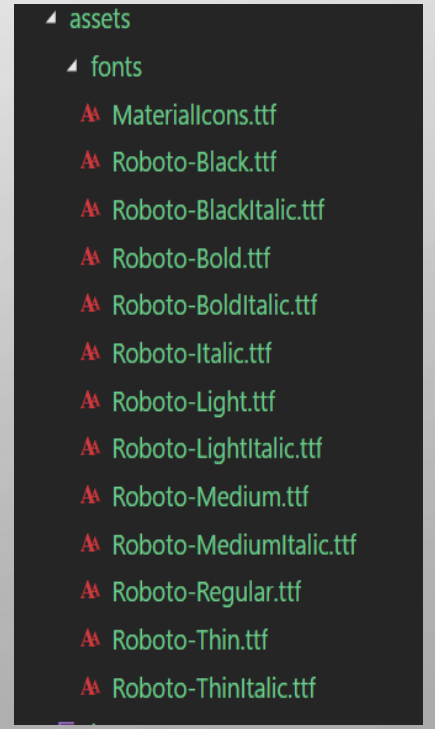
  to : assets/fonts



REACT NATIVE MATERIAL UI

- This project uses Roboto as the main font for text. Make sure to add Roboto to your project





REACT NATIVE MATERIAL UI

- Warp every thing in <ThemeProvider>
- import { Button, ThemeProvider } from 'react-native-material-ui';
- onPress

```
btnPrimaryPress(){
  alert("primary pressed!");
}

render() {
  return (
    <ThemeProvider>
    <View style={styles.container}>
      <Text>Open up App.js to start working on your app!7 {new Date().to
      <Button primary text="Primary" onPress={this.btnPrimaryPress} />
      <Button accent text="Accent" />
```

REACT NATIVE MATERIAL UI

- link to demo site: https://github.com/xotahal/react-native-material-ui-demo-app/tree/master/src

- Icons list: https://blog.foswiki.org/System/MaterialIcons

- **react-native-material-design** is similar BUT tested only for android and not for IOS!!!

SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- **03 Fetch**
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

FETCH

- נעשה אותו דבר כמו בREACT רגיל. עובד רגיל בEXPO
- דוגמאות לקריאות ל WEB SERVICE וגם לWEB API

SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- **04 Geolocation**
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

GEOLOCATION

- מתוך האתר יש ל-API כל מיני יכולות

```
navigator.geolocation.getCurrentPosition(
      (position) => {
        const output=
        'latitude=' + position.coords.latitude +
        '\nlongitude=' + position.coords.longitude +
        '\naltitude=' + position.coords.altitude +
        '\nheading=' + position.coords.heading +
        '\nspeed=' + position.coords.speed

        alert(output);
      },
      (error) => alert(error.message),
      { enableHighAccuracy: true, timeout: 20000, maximumAge: 1000 }
    );
```

SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- **05 Camera**
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

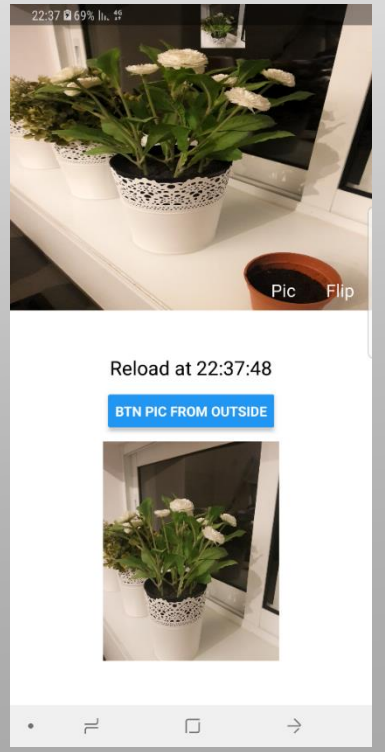
CAMERA-HOOKS

```
import { Camera } from 'expo-camera';
...
  const [hasPermission, setHasPermission] = useState(null);
  const [type, setType] = useState(Camera.Constants.Type.back);
  const [camera, setCamera] = useState(null);
  const [picUri, setPicUri] = useState('https://reactjs.org/logo-og.png');

  useEffect(() => {
    (async () => {
      const { status } = await Camera.requestCameraPermissionsAsync();
      setHasPermission(status === 'granted');
    })();
  }, []);
...
<Camera style={styles.camera}
        ref={ref => setCamera(ref)}
        type={type}>
        <View style={styles.buttonContainer}>
 ...
  onPress={async () => {
            if (camera) {
              const data = await camera.takePictureAsync(null);
              console.log(data.uri)
              setPicUri(data.uri);
            }
...
onPress={() => {
            setType( type === Camera.Constants.Type.back ? Camera.Constants.Type.front : Camera.Constants.Type.back
            );...
```



```
expo install expo-camera
```

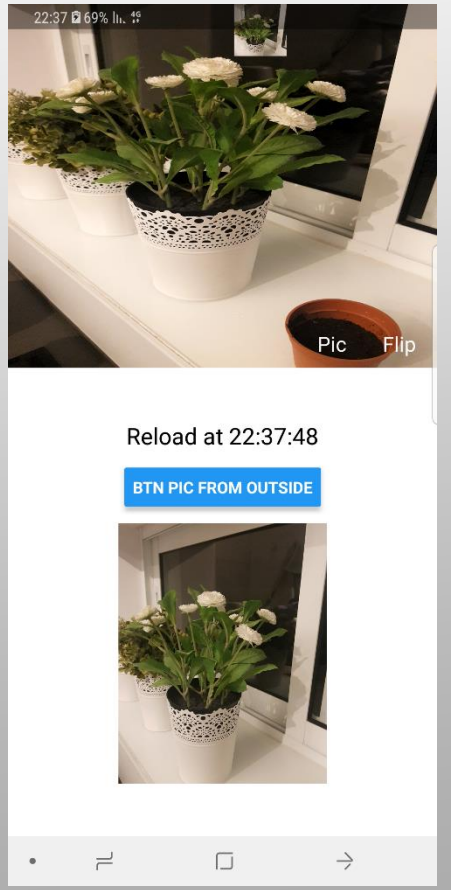
```
<Drawer.Screen name="CameraPage" component={CameraPage}
  options={{
    drawerLabel: 'CameraPage', unmountOnBlur: true
  }} />
```



```
import { Camera } from 'expo-camera';
...
 this.state = {
      hasCameraPermission: null,
      type: Camera.Constants.Type.back,
      picUri: 'https://facebook.github.io/react-native/docs/assets/favicon.png'
    };
...
 async componentWillMount() {
    const { status } = await Camera.requestPermissionsAsync();
    this.setState({ hasCameraPermission: status === 'granted' });
  }

  btnPic = async () => {
    debugger;
    let photo2 = await this.camera.takePictureAsync();
    //alert(photo2.uri);
    this.setState({ picUri: photo2.uri });
    Vibration.vibrate();
  }
...
onPress={() => {this.setState({
     type: this.state.type === Camera.Constants.Type.back ? Camera.Constants.Type.front : Camera.Constants.Type.back });
...
<Camera
```

expo install expo-camera

CAMERA



SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- **06 Navigation**
- 07 Image Upload
- 08 Push Notification

NAVIGATOR

- https://reactnavigation.org/
- V6

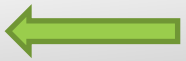
- npm install @react-navigation/native
- expo install react-native-screens react-native-safe-area-context
- npm install @react-navigation/native-stack
- npm install @react-navigation/material-bottom-tabs react-native-paper react-native-vector-icons
- npm install @react-navigation/drawer
- expo install react-native-gesture-handler react-native-reanimated

STACK NAVIGATOR

```
import React from 'react';
import { View, Text } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from  '@react-navigation/native-stack';

import FirstPage from './Pages/FirstPage';
import SecondPage from './Pages/SecondPage';
import TabbedPageNavigator from './Pages/TabbedPage';
import MaterialTabbedPage  from './Pages/MaterialTabbedPage';

const Stack = createNativeStackNavigator();
function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="FirstPage">
        <Stack.Screen name="FirstPage" component={FirstPage} />
        <Stack.Screen name="SecondPage" component={SecondPage} />
        <Stack.Screen name="TabbedPageNavigator" component={TabbedPageNavigator} />
        <Stack.Screen name="MaterialTabbedPage" component={MaterialTabbedPage} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
export default App;
```



```
<TouchableOpacity onPress={() => {
          props.navigation.navigate('SecondPage');
        }}>
  <Text style={{ ...}} >
          Goto Second Page!</Text>
</TouchableOpacity>
```



```
import React from 'react';
import { createMaterialBottomTabNavigator } from '@react-navigation/material-bottom-tabs';
import MaterialCommunityIcons from 'react-native-vector-icons/MaterialCommunityIcons';
import FirstTabPage from './FirstTabPage';
import SecondTabPage from './SecondTabPage';
import { Text } from 'react-native';
const Tab = createMaterialBottomTabNavigator();

export default function MaterialTabbedPageNavigator() {
  return (
    <Tab.Navigator
      initialRouteName="FirstTabPage"
      activeColor="#55ff00"
      inactiveColor='black'
      barStyle={{ backgroundColor: '#694fad' }}
    >
      <Tab.Screen
        name="FirstTabPage"
        component={FirstTabPage}
        options={{
          tabBarLabel: <Text style={{
            fontSize: 9, fontWeight: '700',
            color: 'white'
          }}> My Tickets </Text>,
          tabBarIcon: ({ color }) => (
            <MaterialCommunityIcons name="bell" color={color} size={25} />
          ),
        }}
      />
```

MATERIAL BOTTOM TABBED NAVIGATOR

There is also Material**Top**TabNavigator

```
<Tab.Screen
        name="SecondTabPage"
        component={SecondTabPage}
        options={{
          tabBarLabel: 'Second Page',
          tabBarIcon: ({ color }) => (
            <MaterialCommunityIcons name="baseball" color={color} size={25} />
          ),
        }}
      />
    </Tab.Navigator>
  );
}
```

MATERIAL BOTTOM TABBED NAVIGATOR CONT

There is also Material**Top**TabNavigator

DRAWER NAVIGATOR

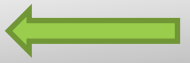
```
import { createDrawerNavigator } from '@react-navigation/drawer';
const Drawer = createDrawerNavigator();

function MyDrawer() {
  return (
    <Drawer.Navigator initialRouteName="FirstPage">
      <Drawer.Screen
        name="FirstPage"
        component={FirstPage}
        options={{ drawerLabel: 'FirstPage' }}
      />
      <Drawer.Screen…
    </Drawer.Navigator>
  );
}

//pay attention that the stack navigator screens must match the drawer screens!
const Stack = createNativeStackNavigator();
function App() {
  return (
      <NavigationContainer>
        <MyDrawer>
          <Stack.Navigator initialRouteName="FirstPage">
            <Stack.Screen name="FirstPage" component={FirstPage} />
…
            <Stack.Screen name="MaterialTabbedPageNavigator" component={MaterialTabbedPageNavigator} />
          </Stack.Navigator>
        </MyDrawer>
      </NavigationContainer>
```

SEND INFO BETWEEN PAGES PARAMS

```
this.props.navigation.navigate('SecondPage', { user: 'Lucy' + new Date().getSeconds() });
```

- Sending

```
props.route.params != undefined ?
        props.route.params.user : '...'
```



- recieving

NAVIGATION EVENTS – REFRESH PAGE THOUGH NAVIGATION - HOOKS

```
import { useFocusEffect } from '@react-navigation/native';

export default function SecondPage({ route }) {

    useFocusEffect(
        React.useCallback(() => {
            // Do something when the screen is focused
            console.log('focus');

            return () => {
                // Do something when the screen is unfocused
                // Useful for cleanup functions
                console.log('unfocus');
            };
        }, [])
    );

    return (
        <View>
            <Text>Second Page</Text>
```

בגלל שהאירועים הללו רצים פעם אחת אין בעיה לקרוא ל setstate בפנים. הוא יקרא רק פעם אחת.

- בכדי לאפשר הרצת קוד במעבר בין עמודים.
- נוכל לדעת מאיפה הגענו ועם איזה מידע ולהריץ למשל setState

NAVIGATION EVENTS – REFRESH PAGE THOUGH NAVIGATION

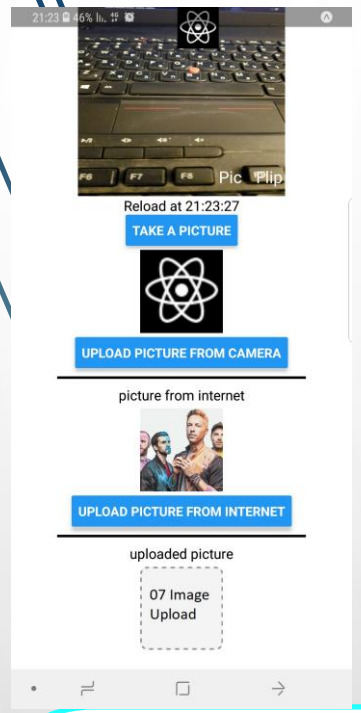
```
componentDidMount() {
  this._unsubscribeFocus = this.props.navigation.addListener('focus', (payload) => {
    console.log('will focus', payload);
    this.setState({stam:'stam'});
  });
  this._unsubscribeBlur = this.props.navigation.addListener('blur', (payload) => {
    console.log('will blur', payload)
  });
}

componentWillUnmount() {
  this._unsubscribeFocus();
  this._unsubscribeBlur();
}
```

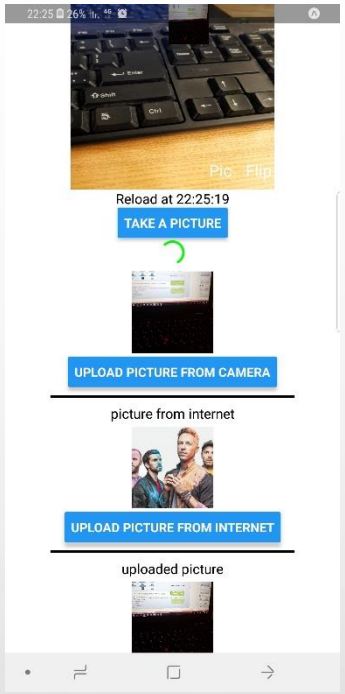
בגלל שהאירועים הללו רצים פעם אחת אין בעיה לקרוא ל setstate בפנים. הוא יקרא רק פעם אחת.

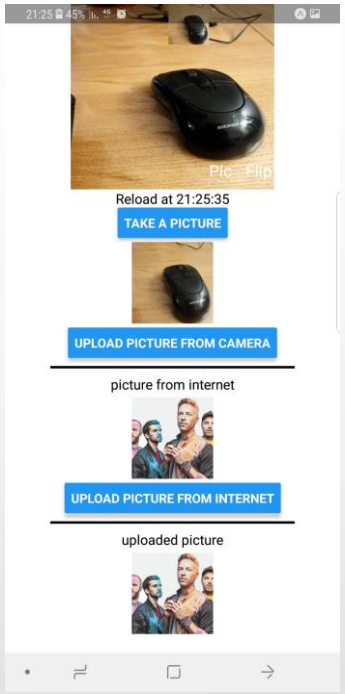
- בכדי לאפשר הרצת קוד במעבר בין עמודים.
- נוכל לדעת מאיפה הגענו ועם איזה מידע ולהריץ למשל setState

SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- **07 Image Upload**
- 08 Push Notification







WEB SERVICE

IMAGE UPLOAD – FETCH, CLIENT SIDE

- תמונה מהמצלמה: נצלם תמונה כרצף של ביטים בבסיס 64 ונשלח אותו לשרת כמחרוזת ארוכה בליווי השם של התמונה.
- תמונה מהאינטרנט: נשלח לשרת את הURL ואת השם של התמונה והשרת ב C# יוריד את התמונה וישמור אותה אצלו.

```
let photo = await this.camera.takePictureAsync({
    quality: 0.1,
    base64: true,
  });
  this.setState({
    pic64base: photo.base64,
    picName64base: 'image1_' + new Date().getTime() + '.jpg',
    picUri: `${photo.base64}`,
  });
```

להקטין את גודל התמונה שצולמה. 1 מקסימום 0 מינימום

חשוב!!! בRN כאשר מזמנים תמונה היא נשמרת תחת השם שלה ב CACHE ולכן אם מזמנים תמונה עם אותו שם תופיע התמונה הראשונה שוב פעם. לכן כאשר נעלה את התמונה לשרת נצטרך ליצר לה שם חדש בכל פעם. פה אני מייצר את השם החדש. השם החדש מורכב מתחילית אשר ביקשנו ועוד מספר שנוצר מהתיקים של המחשב.

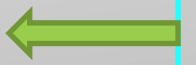
WEB SERVICE

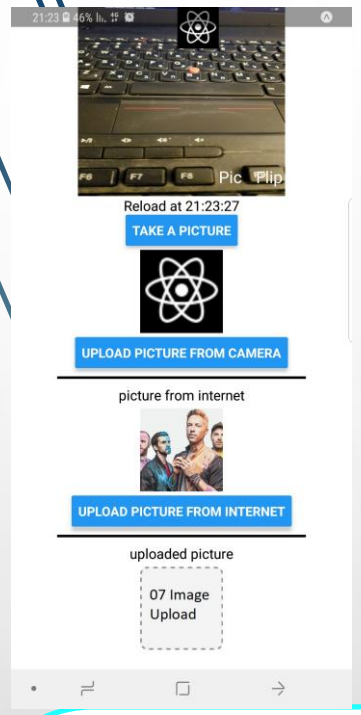
IMAGE UPLOAD – FETCH, CLIENT SIDE

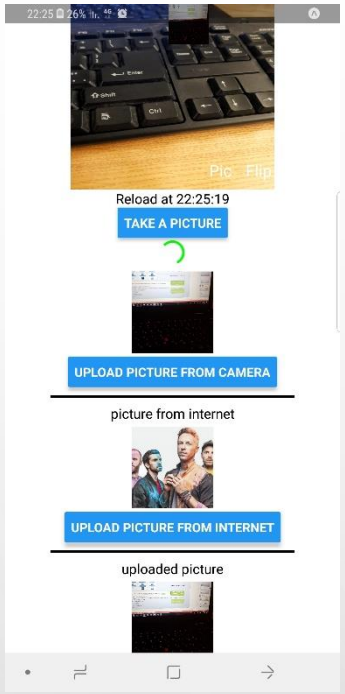
```
uploadBase64ToASMX = () => {
   this.setState({ animate: true });
   let urlAPI =
     'http://ruppinmobile.tempdomain.co.il/site01/webservice.asmx/ImgUpload';

   fetch(urlAPI, {
     method: 'POST',
     body: JSON.stringify({
       base64img: this.state.pic64base,
       base64imgName: this.state.picName64base,
     }),
```

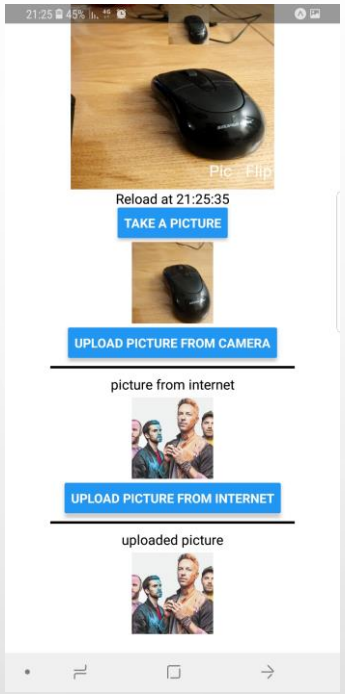
WEB SERVICE
IMAGE UPLOAD – C#, SERVER SIDE

```
[WebMethod]
    public string ImgUpload(string base64img, string base64imgName)
    {
        //for example - pay attention the first '/' is part of the image!
        //
        //File.AppendAllText(Server.MapPath("images/file1.txt"), base64imgName + "\r\n");
        File.WriteAllBytes(Server.MapPath("images/"+ base64imgName), Convert.FromBase64String(base64img));

        return new JavaScriptSerializer().Serialize(new { res = "OK" });
    }
```









WEB API

IMAGE UPLOAD – FETCH, CLIENT SIDE

```
...
let photo = await this.camera.takePictureAsync({quality : 0.7});
...
 imageUpload = (imgUri, picName) => {
    let urlAPI = "http://.../site01/uploadpicture";
    let data1 = new FormData();
    data1.append('file', {
      uri: imgUri,
      name: picName,
      type: 'image/jpg'
    });
```

להקטין את גודל התמונה שצולמה. 1 מקסימום 0 מינימום

מיקום הקוד של צד השרת

IMAGE UPLOAD – FETCH, CLIENT SIDE

```
...
  const config = {
    method: 'POST',
    body: datal,
  }

fetch(urlAPI, config)
  .then((res) => {
    console.log(res.status);
    if (res.status == 200) {
      console.log(res.status);
      return res.json();
    }
    else { return "err"; }
  })
  .then((responseData) => {
    console.log(responseData);
    if (responseData != "err") {
      console.log(responseData.filePath);
    }
    else { alert('error uploding ...'); }
  })
  .catch(err => { alert('err upload= ' + err); });
```

חשוב!!! בRN כאשר מזמנים תמונה היא נשמרת תחת השם שלה ב CACHE ולכן אם מזמנים תמונה עם אותו שם תופיע התמונה הראשונה שוב פעם. לכן כאשר נעלה את התמונה לשרת נצטרך ליצר לה שם חדש בכל פעם. זאת ניתן לעשות בצד השרת. ולקבל את השם החדש לצד הלקוח. פה אני מחלץ את השם החדש. השם החדש מורכב מתחילית אשר ביקשנו כאשר שלחנו את התמונה לשרת ועוד GUID שנוצר בשרת

IMAGE UPLOAD – C# WEB API CORE, SERVER SIDE

```
[ApiController]
[Route("api/[controller]")]
public class FilesController : ControllerBase
{
    private readonly string _uploadFolderPath; // Define the folder path where files will be
uploaded

    public FilesController(IWebHostEnvironment env)
    {
        _uploadFolderPath = Path.Combine(env.ContentRootPath, "Uploads"); // Example path,
adjust as needed
    }


    [HttpPost("upload")]
    public async Task<IActionResult> Upload([FromForm] FileUploadModel model)
    {
        if (model.File == null || model.File.Length == 0)
        {
            return BadRequest("No file uploaded.");
        }

        try
        {
            // Ensure the uploads folder exists
            Directory.CreateDirectory(_uploadFolderPath);
```

IMAGE UPLOAD – C# WEB API, SERVER SIDE

```
// Get the original filename and extension
        var originalFileName = Path.GetFileNameWithoutExtension(model.File.FileName);
        var fileExtension = Path.GetExtension(model.File.FileName);

        // Delete any existing files with the same original filename
        DeleteExistingFiles(originalFileName);

        // Generate a unique filename by concatenating original filename with timestamp
        var timeStamp = DateTime.Now.ToString("yyyyMMddHHmmssfff");
        var fileName = $"{originalFileName}_{timeStamp}{fileExtension}";
        var filePath = Path.Combine(_uploadFolderPath, fileName);

        // Save the uploaded file to the server
        using (var stream = new FileStream(filePath, FileMode.Create))
        {
            await model.File.CopyToAsync(stream);
        }

        // Return the URL to access the uploaded file
        var baseUrl = $"{Request.Scheme}://{Request.Host}";
        var fileUrl = Path.Combine(baseUrl, "", "Uploads", fileName); // Adjust the route as needed

        return Ok(new { FilePath = fileUrl });
    }
    catch (Exception ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, $"Error uploading file: {ex.Message}")
    }
}
```

IMAGE UPLOAD – C# WEB API, SERVER SIDE

```
private void DeleteExistingFiles(string originalFileName)
{
    // Get all files in the upload folder that start with the original filename
    var files = Directory.GetFiles(_uploadFolderPath, $"{originalFileName}_*");

    // Delete each file
    foreach (var file in files)
    {
        System.IO.File.Delete(file);
    }
}

 …

public class FileUploadModel
{
    public IFormFile File { get; set; }
}
```

IMAGE UPLOAD – C# WEB API, SERVER SIDE

IN PROGRAM.CS

```
app.UseStaticFiles(new StaticFileOptions
{
    FileProvider = new PhysicalFileProvider(
        Path.Combine(builder.Environment.ContentRootPath, "Uploads")),
    RequestPath = "/Uploads" // Optional: change the request path if needed
});
```

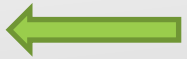
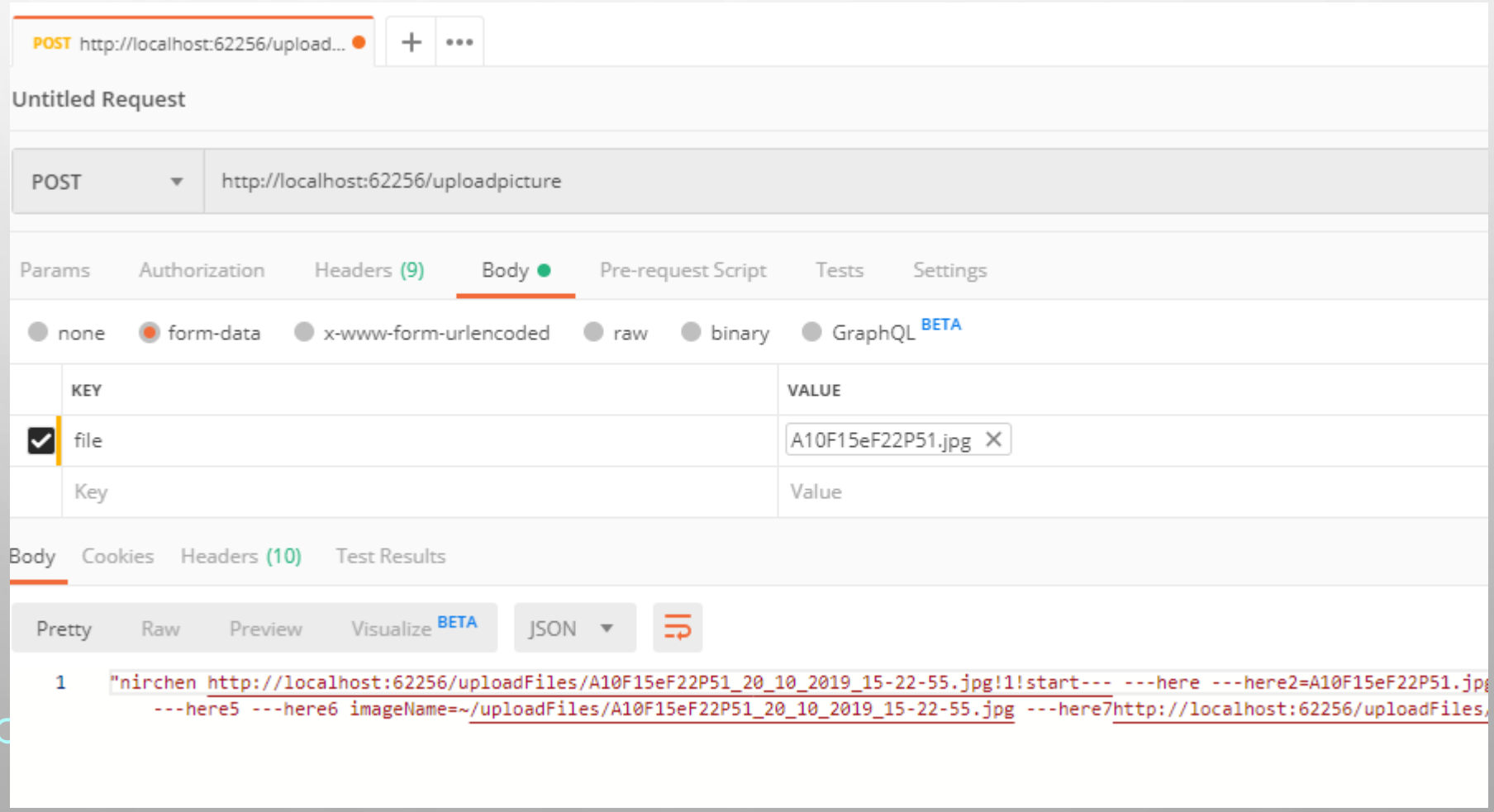
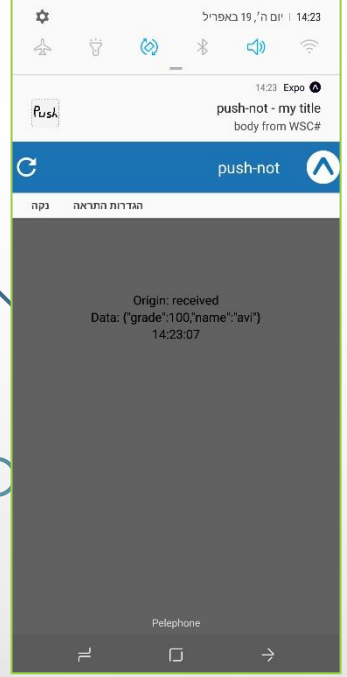


IMAGE UPLOAD – C# WEB API, SERVER SIDE TESTING



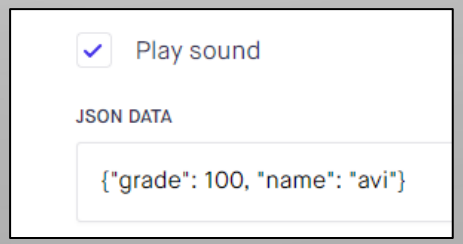
SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- **08 Push Notification**



PUSH NOTIFICATION

- נעשה שימוש בספריה של EXPO בכדי לקבל יכולות של PN עבור ANDROID ו-IOS באותו קוד. כמו כן לא צריך לייצר משתמש של APPLE.
- https://docs.expo.io/versions/latest/guides/push-notifications
- שימו לב שצריך לאפשר את ההתראות במכשיר. בחלק ממכשירי ה- ANDROID לא הצלחתי לאפשר זאת ויכול להיות שצריך לחפש\"לשחק" הכן מאפשרים את ההתראות.
- אין צורך לפתוח חשבון בשירות ענן כלשהוא. EXPO כבר מבצעים הכל עבורינו.
- בכדי לשלוח הודעה כל מה שצריך הוא לשלוח הודעת POST לשרת של EXPO שמעביר את ההודעה לשירות ענן בכדי לשלוח PN.
- יש מערכת ששולחת הודעות ישירות מהאתר של EXPO בכדי לבחון את הקוד בצד לקוח https://expo.io/dashboard/notifications



חשוב!!!
חייבים להיות מחוברים לEXPO גם בטלפון עצמו!!!!
וגם בvisual code ע"י expo login והכנסת שם וסיסמה!!!

PN – CLIENT SIDE REGISTRATION

```
import Constants from 'expo-constants';
import * as Notifications from 'expo-notifications';
import { Platform } from 'react-native';

export default async function registerForPushNotificationsAsync() {
  let token;
  if (Constants.isDevice) {
    const { status: existingStatus } = await Notifications.getPermissionsAsync();
    let finalStatus = existingStatus;
    if (existingStatus !== 'granted') {
      const { status } = await Notifications.requestPermissionsAsync();
      finalStatus = status;
    }
    if (finalStatus !== 'granted') {
      alert('Failed to get push token for push notification!');
      return;
    }
    token = (await Notifications.getExpoPushTokenAsync()).data;
    console.log(token);
  } else {
    alert('Must use physical device for Push Notifications');
  }

  if (Platform.OS === 'android') {
    Notifications.setNotificationChannelAsync('default', {
      name: 'default',
      importance: Notifications.AndroidImportance.MAX,
      vibrationPattern: [0, 250, 250, 250],
      lightColor: '#FF231F7C',
    });
  }

  return token;
}
```


- npm i expo-notifications
- npm i expo-constants
- פה צריך לדאוג ל:
  - קבלת הרשאה ליכולת קבלת התראות
  - קבלת מספר ייחודי מזהה של מכשיר הטלפון

PN – CLIENT SIDE RECEIVING MSG

```
import * as Notifications from 'expo-notifications';
import React, { useState, useEffect, useRef } from 'react';
import { Text, View, Button } from 'react-native';
import registerForPushNotificationsAsync from './registerForPushNotificationsAsync';

Notifications.setNotificationHandler({
  handleNotification: async () => ({
    shouldShowAlert: true,
    shouldPlaySound: false,
    shouldSetBadge: false,
  }),
});


export default function PushPage() {
  const [expoPushToken, setExpoPushToken] = useState('');
  const [notification, setNotification] = useState(false);
  const notificationListener = useRef();
  const responseListener = useRef();
```

- פה נעשה רישום של פונקציה לקבלת התראה
- יש אפשרות לראות בהתראה עצמה בין היתר TITLE, BODY,
- באפליקציה - DATA

PN – CLIENT SIDE RECEIVING MSG

```
useEffect(() => {
  registerForPushNotificationsAsync().then(token => setExpoPushToken(token));

  // This listener is fired whenever a notification is received while the app is foregrounded
  notificationListener.current = Notifications.addNotificationReceivedListener(notification => {
    console.log(notification);
    setNotification(notification);
  });

  //This listener is fired whenever a user taps on or interacts with a notification (works when app is foregrounded, backgrounded, or killed)
  responseListener.current = Notifications.addNotificationResponseReceivedListener(response => {
    console.log(response);
    setNotification(response.notification);
  });

  return () => {
    Notifications.removeNotificationSubscription(notificationListener.current);
    Notifications.removeNotificationSubscription(responseListener.current);
  };
}, []);


// Can use this function below, OR use Expo's Push Notification Tool-> https://expo.dev/notifications
async function sendPushNotification(expoPushToken) {
  const message = {
    to: expoPushToken,
    sound: 'default',
    title: 'Original Title',
    body: 'And here is the body!',
    data: { name: "nir", seconds: new Date().getSeconds() }
  };
```

- פה נעשה רישום של פונקציה לקבלת התראה
- יש אפשרות לראות בהתראה עצמה בין היתר TITLE, BODY,
- באייקון - BADGE
- באפליקציה - DATA

PN – CLIENT SIDE RECEIVING MSG

```
  await fetch('https://exp.host/--/api/v2/push/send', {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Accept-encoding': 'gzip, deflate',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(message),
  });
}

return (
  <View
    style={{
      flex: 1,
      alignItems: 'center',
      justifyContent: 'space-around',
    }}>
    <Text>Your expo push token: {expoPushToken}</Text>
    <View style={{ alignItems: 'center', justifyContent: 'center' }}>
      <Text>Title: {notification && notification.request.content.title} </Text>
      <Text>Body: {notification && notification.request.content.body}</Text>
      <Text>Data: {notification && JSON.stringify(notification.request.content.data)}</Text>
    </View>
    <Button
      title="Press to Send Notification"
      onPress={async () => {
        await sendPushNotification(expoPushToken);
      }}
    />
  </View>
);
}
```


- פה נעשה רישום של פונקציה לקבלת התראה
- יש אפשרות לראות בהתראה עצמה בין היתר TITLE, BODY,
- באייקון - BADGE
- באפליקציה - DATA

PN – SERVER SIDE SENDING MSG C# WEB API

```
[Route("sendpushnotification")]
public string Post([FromBody]PushNotData pnd)
{
    // Create a request using a URL that can receive a post.
    WebRequest request = WebRequest.Create("https://exp.host/--/api/v2/push/send");
    // Set the Method property of the request to POST.
    request.Method = "POST";
    // Create POST data and convert it to a byte array.
    var objectToSend = new
    {
        to = pnd.to,
        title = pnd.title,
        body = pnd.body,
        badge = pnd.badge,
        data = pnd.data//new { name = "nir", grade = 100 }
    };

    string postData = new JavaScriptSerializer().Serialize(objectToSend);

    byte[] byteArray = Encoding.UTF8.GetBytes(postData);
    // Set the ContentType property of the WebRequest.
    request.ContentType = "application/json";
    // Set the ContentLength property of the WebRequest.
    request.ContentLength = byteArray.Length;
    // Get the request stream.
    Stream dataStream = request.GetRequestStream();
```


- כל מה שנותר זה רק לקרוא לפונקציה הזו או מהשרת או מהלקוח ע"י fetch

PN – SERVER SIDE SENDING MSG C# WEB API – CONT'

```
// Write the data to the request stream.
    dataStream.Write(byteArray, 0, byteArray.Length);
    // Close the Stream object.
    dataStream.Close();
    // Get the response.
    WebResponse response = request.GetResponse();
    // Display the status.
    string returnStatus = ((HttpWebResponse)response).StatusDescription;
    //Console.WriteLine(((HttpWebResponse)response).StatusDescription);
    // Get the stream containing content returned by the server.
    dataStream = response.GetResponseStream();
    // Open the stream using a StreamReader for easy access.
    StreamReader reader = new StreamReader(dataStream);
    // Read the content.
    string responseFromServer = reader.ReadToEnd();
    // Display the content.
    //Console.WriteLine(responseFromServer);
    // Clean up the streams.
    reader.Close();
    dataStream.Close();
    response.Close();

    return "success:) --- " + responseFromServer + ", " + returnStatus;
}
```

SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

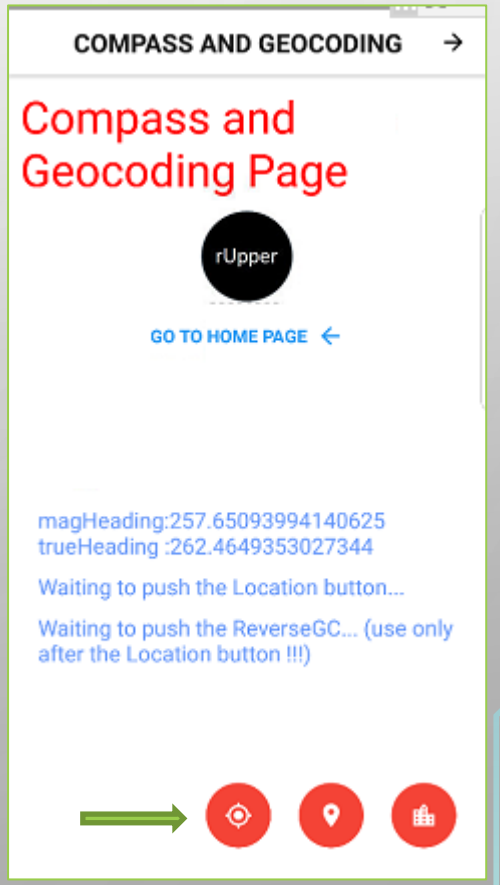
- **09 Compass**
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib

COMPASS

- expo install expo-location
- ניתן לקבל מידע לקבל הכיוון שהטלפון מופנה אליו.
- כיוון אמיתי וכיוון מגנטי

```
import * as Location from 'expo-location';
...
btnHeading = async () => {

    let { status } = await Location.requestPermissionsAsync();
    if (status !== 'granted') {
      alert('Permission to access location was denied');
    }

    let heading = await Location.getHeadingAsync({});
    this.setState({ heading });
  };
```

magHeading:275.7325439453125
trueHeading :280.5465393066406



SYLLABUS

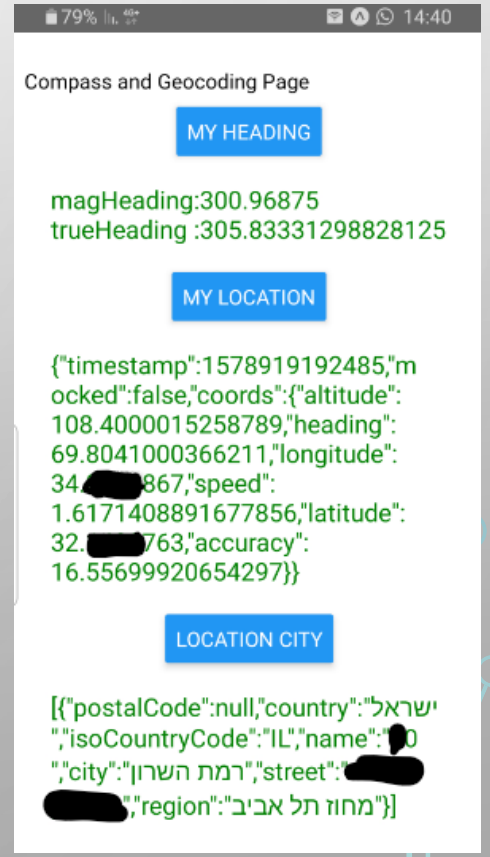
- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification
- 09 Compass
- **10 Geocoding**
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib

GEOCODING

- expo install expo-location
- ניתן לקבל מידע אודות כתובת עבור נ.צ. קיים.
- ניתן לקבל גם הפוך - נ.צ. עבור כתובת (לא בדוגמה הזו)

```
import * as Location from 'expo-location';
...
btnLocation = async () => {
    let { status } = await Location.requestPermissionsAsync();
    if (status !== 'granted') {
      this.setState({errorMessage: 'Permission to access location was denied',  });
    }
    let location = await Location.getCurrentPositionAsync({});
    this.setState({ location });
};
btnReverseGC = async () => {
    let { status } = await Location.requestPermissionsAsync();
    if (status !== 'granted') {
      this.setState({ errorMessage: 'Permission to access location was denied', });
    }
    if (this.state.location) {
      let reverseGC = await Location.reverseGeocodeAsync(this.state.location.coords);
      this.setState({ reverseGC });
    }else{
      alert('You must push the Location button first in order to get the loc
can get the reverse geocode for the latitude and longitude!');
    }
};
```


מחקתי מהדוגמה את הפרטים של הנ.צ. והכתובת



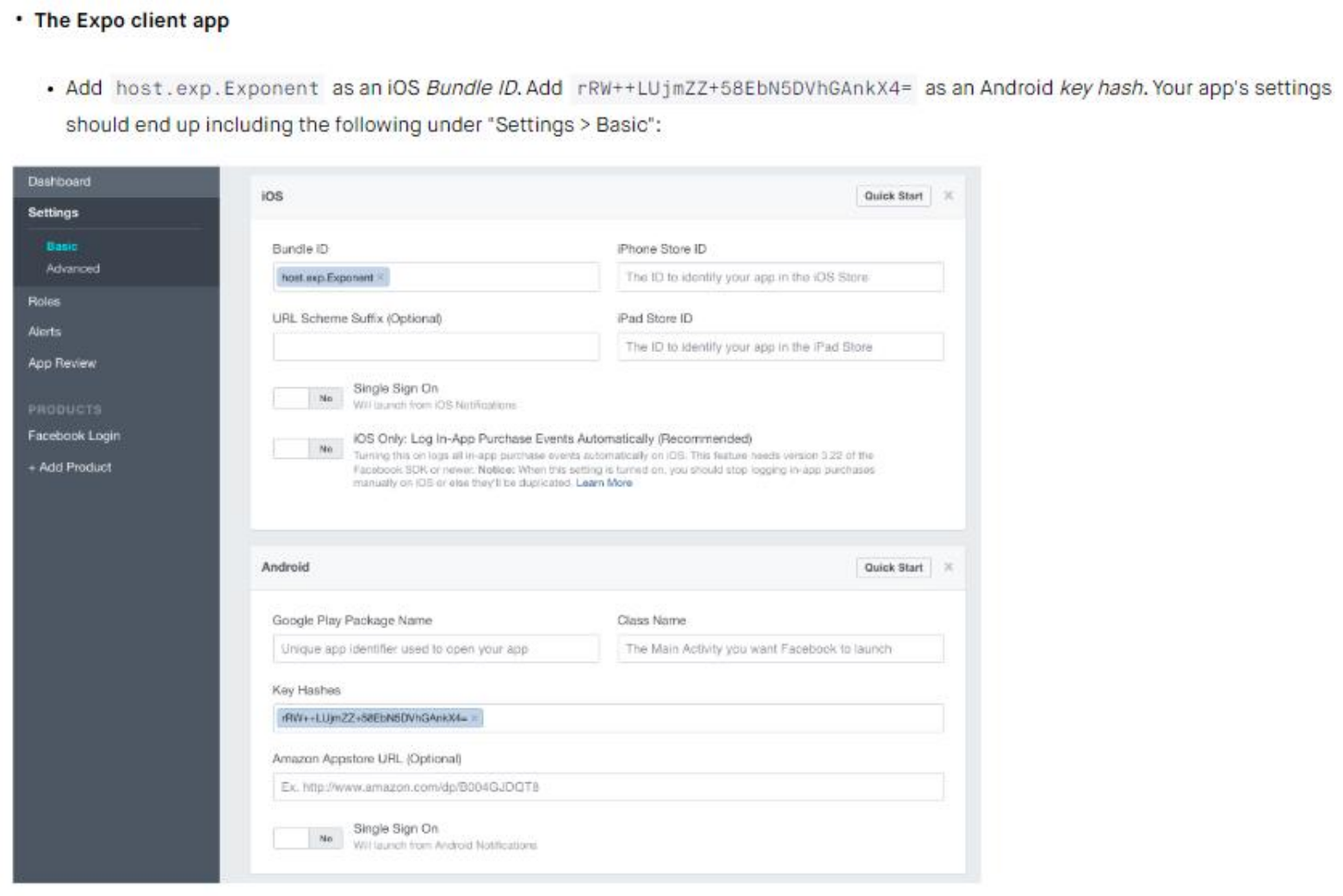
SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload

- 08 Push Notification
- 09 Compass
- 10 Geocoding
- **11 Facebook**
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib

- צריך ליצור פרוייקט ב FB developer API בכדי לקבל appId

- https://docs.expo.io/versions/latest/sdk/facebook/
- rRW++LUjmZZ+58EbN5DVhGAnkX4=
- host.exp.Exponent

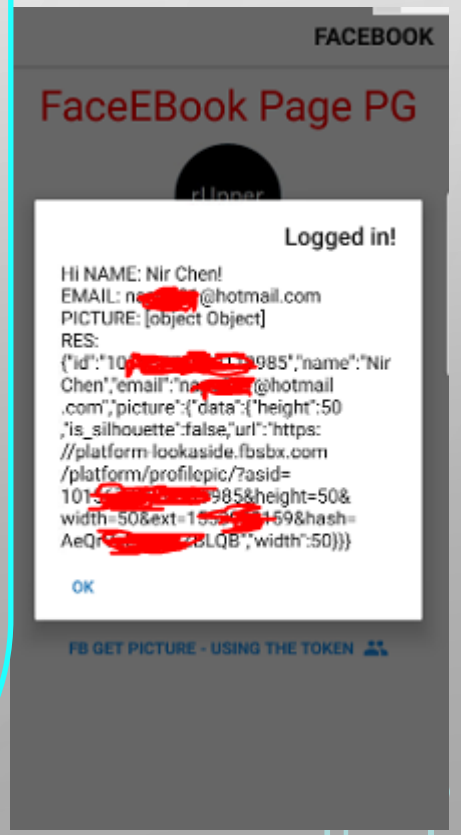


FB

- צריך לקבל token FB יחודי בכדי לעבוד עם ה API שלהם.
- פה אני מקבל שם, מייל, , מספר ID...

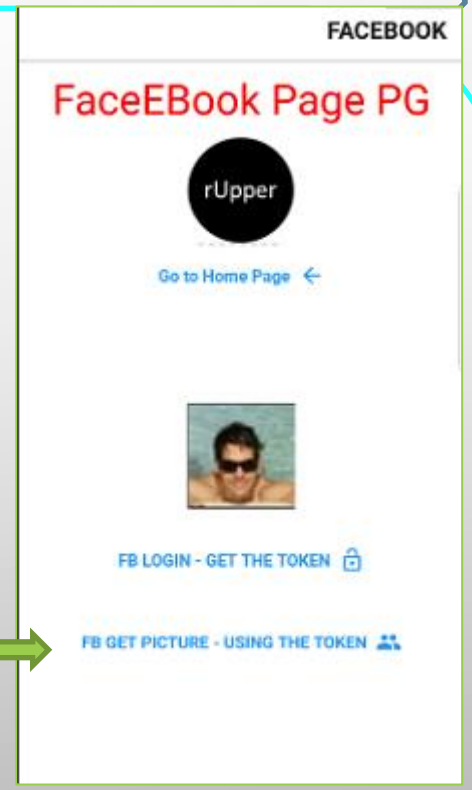
```
import * as Facebook from 'expo-facebook';
...
async function btnFBLogin() {
    try {
      await Facebook.initializeAsync('[YOUR APPID]');
      const { type, token, expires, permissions, declinedPermissions, }
        = await Facebook.logInWithReadPermissionsAsync({
          permissions: ['public_profile'],
        });
      if (type === 'success') {
        // Get the user's name using Facebook's Graph API
        const response = await fetch(`https://graph.facebook.com/me?fields=id,name,email,picture&
access_token=${token}`);
        let res = await response.json();
        Alert.alert('Logged in!', `Hi NAME: ${res.name}!\nEMAIL: ${res.email}\nPICTURE: ${res.picture.
data.url}`);
        //Alert.alert('Logged in!', `Hi NAME: ${res.name}!\nEMAIL: ${res.email}\nPICTURE: ${res.pictu
re}\nRES:${JSON.stringify(res)} `);
      } else {          type === 'cancel'       }
    } catch ({ message }) {
      alert(`Facebook Login Error: ${message}`);
    }
  }
```


מחקתי מהדוגמה את הפרטים האישיים



FB

```
btnFetch_PersonPicture = () => {
        // POST adds a random id to the object sent
        fetch(`https://graph.facebook.com/me?fields=picture&access_token=${this.state.token}`, {
            method: 'POST',
            body: '',
            headers: {
                "Content-type": "application/json; charset=UTF-8"
            }
        })
            .then(response => response.json())
            .then(json => {
                if (json != null) {
                    this.setState({ photoUrl: json.picture.data.url });
                alert(`picture= ${json.picture}\npicture.data.url= ${(json.picture.data.url)}\nRES=${JSON.stringify(json)}`);
                } else {
                    this.setState({ lblErr: true });
                }
            });
    }
```



- פה אני מקבל תמונה

SYLLABUS

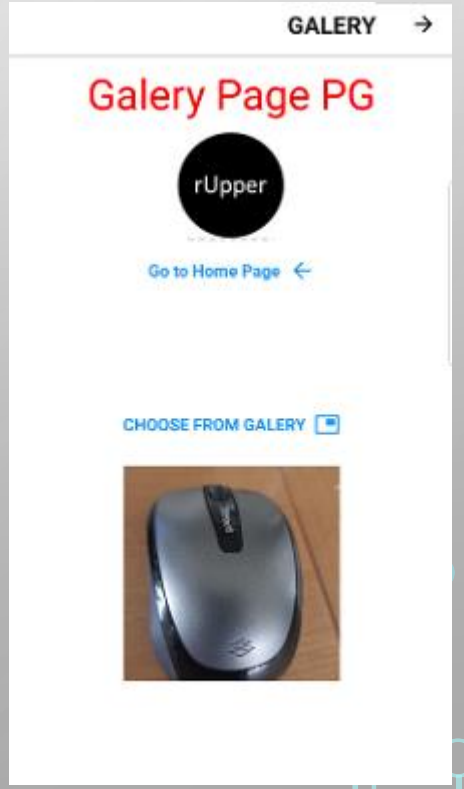
- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload

- 08 Push Notification
- 09 Compass
- 10 Geocoding
- 11 Facebook
- **12 Image gallery**
- 13 Sms
- 14 React Elements UI Lib

IMAGE GALLERY

- ניתן לבחור תמונה מהגלריה
- expo install expo-image-picker

```
import * as ImagePicker from 'expo-image-picker';
...
btnOpenGalery = async () => {
        let result = await ImagePicker.launchImageLibraryAsync({
            //allowsEditing: true,
            //aspect: [4, 3],
          });

          if (!result.cancelled) {
            this.setState({ image: result.uri });
          }
    };
```



SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

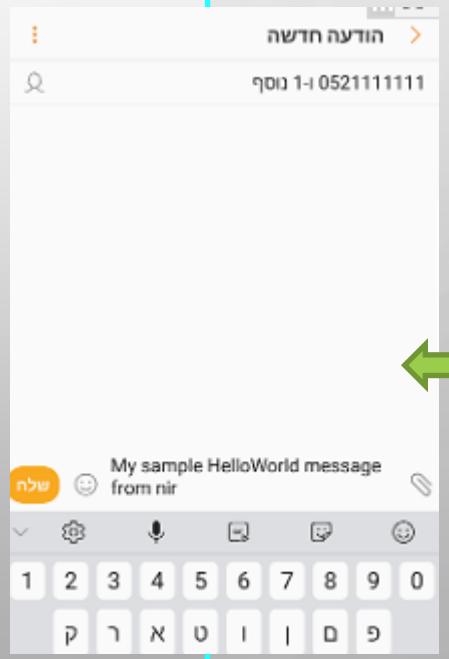
- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- **13 Sms**
- 14 React Elements UI Lib

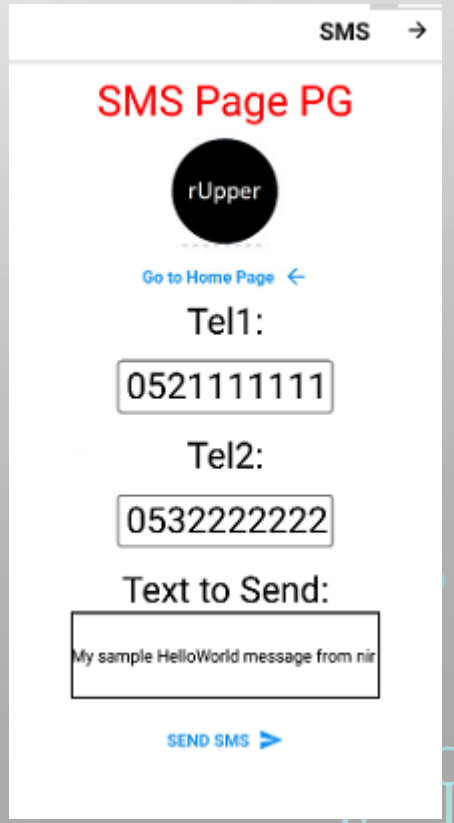
SMS

- ניתן לפתוח את מסך הוסעת ה SMS עם רשימת טלפונים והודעה מוכנה מהאפליקציה שלנו.
- זה לא שולח מהאפליקציה שלנו אלה רק פותח את אפליקציית ה SMS-ים הקיימת

```
import * as SMS from 'expo-sms';
...
btnSendSms = async () => {
        const isAvailable = await SMS.isAvailableAsync();
        if (isAvailable) {
            //alert('send');
            const { result } =
            await SMS.sendSMSAsync(
                [this.state.txtTel1, this.state.txtTel2],
                 this.state.txtTextValue);
            // alert(result);
            this.setState({ txtTextValue: result });
        } else {
            alert('misfortune... there\'s no SMS available on this device');
        }
    };
```


- expo install expo-sms





SYLLABUS

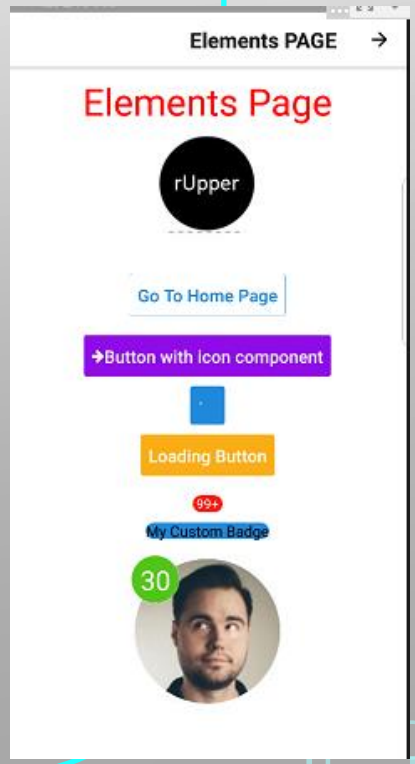
- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- **14 React Elements UI Lib**

REACT NATIVE ELEMENTS UI

```
import { Button } from '@rneui/base';

const MyApp = () => {
  return (
 <Button
        title="HOME"
        iconRight
        icon={{
          name: 'home',
          type: 'font-awesome',
          size: 35,
          color: 'black',
        }}
        iconContainerStyle={{ marginRight: 10 }}
        titleStyle={{ fontWeight: '700' }}
        buttonStyle={{
          backgroundColor: 'rgba(90, 154, 230, 1)',
          borderColor: 'transparent',
          borderWidth: 0,
          borderRadius: 30,
        }}
        containerStyle={{
          width: 200,
          marginHorizontal: 50,
          marginVertical: 10,
        }}
      />
  );
};
```


- https://react-native-training.github.io/react-native-elements/
- npm install @rneui/themed @rneui/base
- npm install react-native-vector-icons
- npx react-native link react-native-vector-icons
- npm install react-native-safe-area-context
- npx react-native link react-native-safe-area-context



SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib
- **15 Linking**

LINKING

```
import * as Linking from 'expo-linking';
import * as WebBrowser from 'expo-web-browser';
Linking.openURL('https://expo.io');
WebBrowser.openBrowserAsync('https://expo.io');
…
Linking.openURL('mailto:support@expo.io?subject=Congrats Snoopy&body=Enjoy your
stay,%0ARegards');
…
Linking.openURL('tel:+123456789');
….
Linking.openURL('sms:+123456789');
…
    //this will open the app through whatsapp
    //1. need to publish the app first!
    //2. when clicked in whatsapp this will go to the published page on expo.io
    //   then you can open the app in the expo if installed.
    btnWhatsapp = () => {
        let text = "hello ";
        text = 'https://expo.io/@nirc/get-a-ride-demo';
        let phoneNumber = '+972523333333';
        Linking.openURL(`whatsapp://send?text=${text}&phone=${phoneNumber}`);
    }
```


Open a url

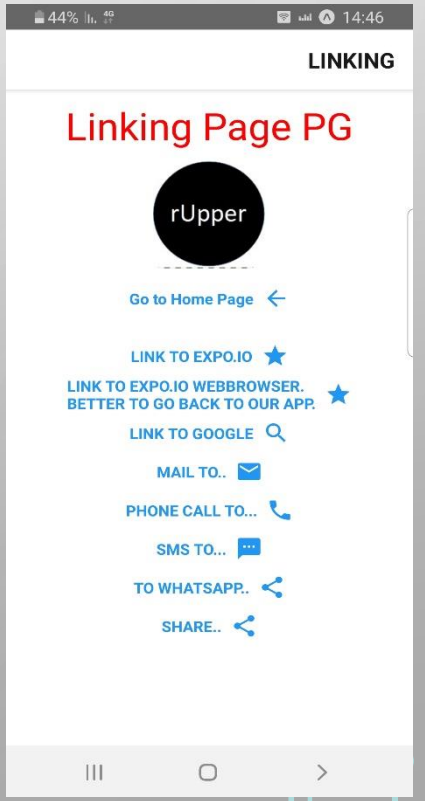
Send mail

Open dialer

Send sms

Send whatsapp

- כל מה שקשור ליצירת קשר עם אפליקציות חיצוניות

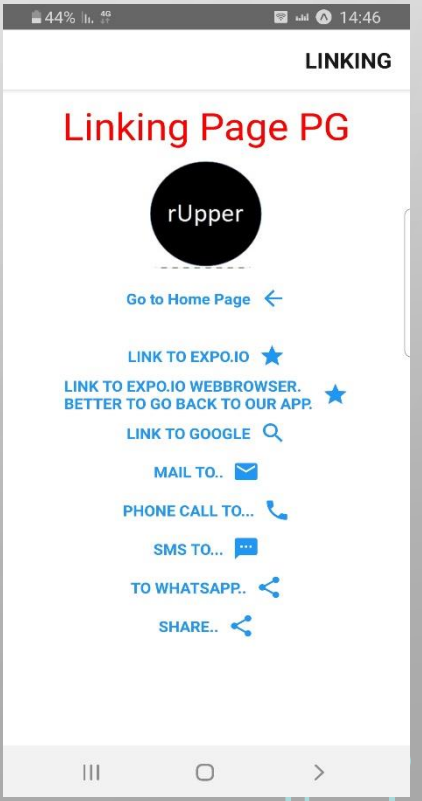


LINKING

```
import { … , Share } from 'react-native';
btnShare = () => {
        //let text =  Expo.Linking.makeUrl();
        let text = 'https://expo.io/@nirc/get-a-ride-demo';

        Share.share({
            message: "Click Here to View More! " + text,
            url: text,
            title: 'nir has invited you to join this activity',
        })
            .then((result) => {
                console.log(result)
                if (result === 'dismissedAction') {
                    return
                }
            })
            .catch((error) => console.log(error))
    }
```

Open all sharable apps

- כל מה שקשור ליצירת קשר עם אפליקציות חיצוניות



SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib
- 15 Linking
- **17-1 Maps**

```
import { StyleSheet, Text, View, Dimensions } from 'react-native';
import MapView from 'react-native-maps';
import { Marker } from 'react-native-maps';
...
<View style={styles.container}>
      <MapView
          style={{flex: 0.7, width: Dimensions.get('window').width - 30,}}
          region={{
            latitude: 32.157154,
            longitude: 34.843893,
            latitudeDelta: 0.0122,
            longitudeDelta: 0.0121,
          }}  >
          <Marker
            coordinate={{
              latitude: 32.15715,
              longitude: 34.843893
            }}
            title='my place:)'
            description='here i am'
            //image={require('../assets/icon.png')}
          />
      </MapView>
</View>
```


MAPS

- expo install react-native-maps
- https://github.com/react-native-community/react-native-maps

SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib
- 15 Linking
- 17-1 Maps
- **17 Fingerprint authentication**

FINGERPRINT AUTHENTICATION

```
import * as LocalAuthentication from 'expo-local-authentication';

scanFingerPrint = async () => {
  try {
    let results = await LocalAuthentication.authenticateAsync();
    if (results.success) {
      this.setState({
        modalVisible: false,
        authenticated: true,
        failedCount: 0,
      });
    } else {
      this.setState({
        failedCount: this.state.failedCount + 1,
      });
    }
  } catch (e) {
    console.log(e);
  }
};
```


- expo install expo-local-authentication

SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib
- 15 Linking
- 17-1 Maps
- 17 Fingerprint authentication
- **18 Async Storage**

ASYNC STORAGE SET

```
import AsyncStorage from '@react-native-async-storage/async-storage';

//opt1 – with async-await
const storeData = async (value) => {
  try {
    const jsonValue = JSON.stringify(value)
    await AsyncStorage.setItem('@storage_Key', jsonValue)
  } catch (e) {
    // saving error
  }
}

//opt2 – WO async-await
const storeData = (value) => {
  try {
    const jsonValue = JSON.stringify(value)
    AsyncStorage.setItem('@storage_Key', jsonValue ,
    () => { do something after the async was done… });
  } catch (e) {
    // saving error
  }
}
```


- כמו localstorage בJS רק ב RN
- יש לשים לב שזה אכן אסינכרוני!!!
- https://react-native-async-storage.github.io/async-storage/
- npm install @react-native-async-storage/async-storage
- שומר רק מחרוזת – תמיד ניתן להשתמש בJSON.stringify

ASYNC STORAGE GET

```
import AsyncStorage from '@react-native-async-storage/async-storage';

//opt1 – with async-await
const getData = async () => {
  try {
    const jsonValue = await AsyncStorage.getItem('@storage_Key')
    return jsonValue != null ? JSON.parse(jsonValue) : null;
  } catch(e) {
    // error reading value
  }
}

//opt2 – WO async-await
const getData = () => {
  try {
     AsyncStorage.getItem('@storage_Key', (err, result) =>{
        do something with result which is the value stored after the async was done…
        return reuslt != null ? JSON.parse(result) : null;
    })
} catch(e) {
    // error reading value
  }
}
```

IMAGES

```
<Image source={require('../assets/cp.jpg')}
  style={{ width: 150, height: 100, borderWidth: 1, borderColor: 'red', margin: 10 }} />
<Image source={require('../Imgs/cp2.jpg')}
  style={{ width: 150, height: 100, borderWidth: 1, borderColor: 'red', margin: 10 }} />
<Image
  source={{ uri:
'https://upload.wikimedia.org/wikipedia/commons/2/25/Coldplay_%282842037407%29.jpg' }}
  style={{ width: 150, height: 100, borderWidth: 1, borderColor: 'red', margin: 10 }} />
```

MORE EXPO CAPABILITIES!!!

SDK API REFERENCE

- Introduction
- Accelerometer
- Admob
- Amplitude
- AppLoading
- ART
- Asset
- Audio
- AuthSession
- AV
- BarCodeScanner
- BlurView
- Branch
- Brightness
- Calendar
- Camera
- Constants
- Contacts
- DeviceMotion
- DocumentPicker
- ErrorRecovery
- FacebookAds
- Facebook
- FaceDetector
- FileSystem
- Fingerprint
- Font
- GestureHandler
- GLView
- Google
- Gyroscope
- ImageManipulator
- ImagePicker
- IntentLauncherAndroid
- KeepAwake
- LinearGradient
- Localization
- Localization
- Location
- Lottie
- Magnetometer
- MailComposer
- MapView
- Notifications
- Payments
- Pedometer
- Permissions
- Print
- registerRootComponent
- ScreenOrientation
- SecureStore
- Segment
- Speech
- SQLite
- Svg
- takeSnapshotAsync
- Updates
- Video
- WebBrowser