משרד התעשיה המסחר והתעסוקה

מה"ט המכון הממשלתי להכשרה בטכנולוגיה ובמדע.

#### 



# 

## 

## 

## המגמה: תוכנה נג'

פרויקט גמר

הנושא:ISM , ניהול מפגשים וטיולים

##### **המגיש/ה: ­­­\_בהאא טאהא\_\_**

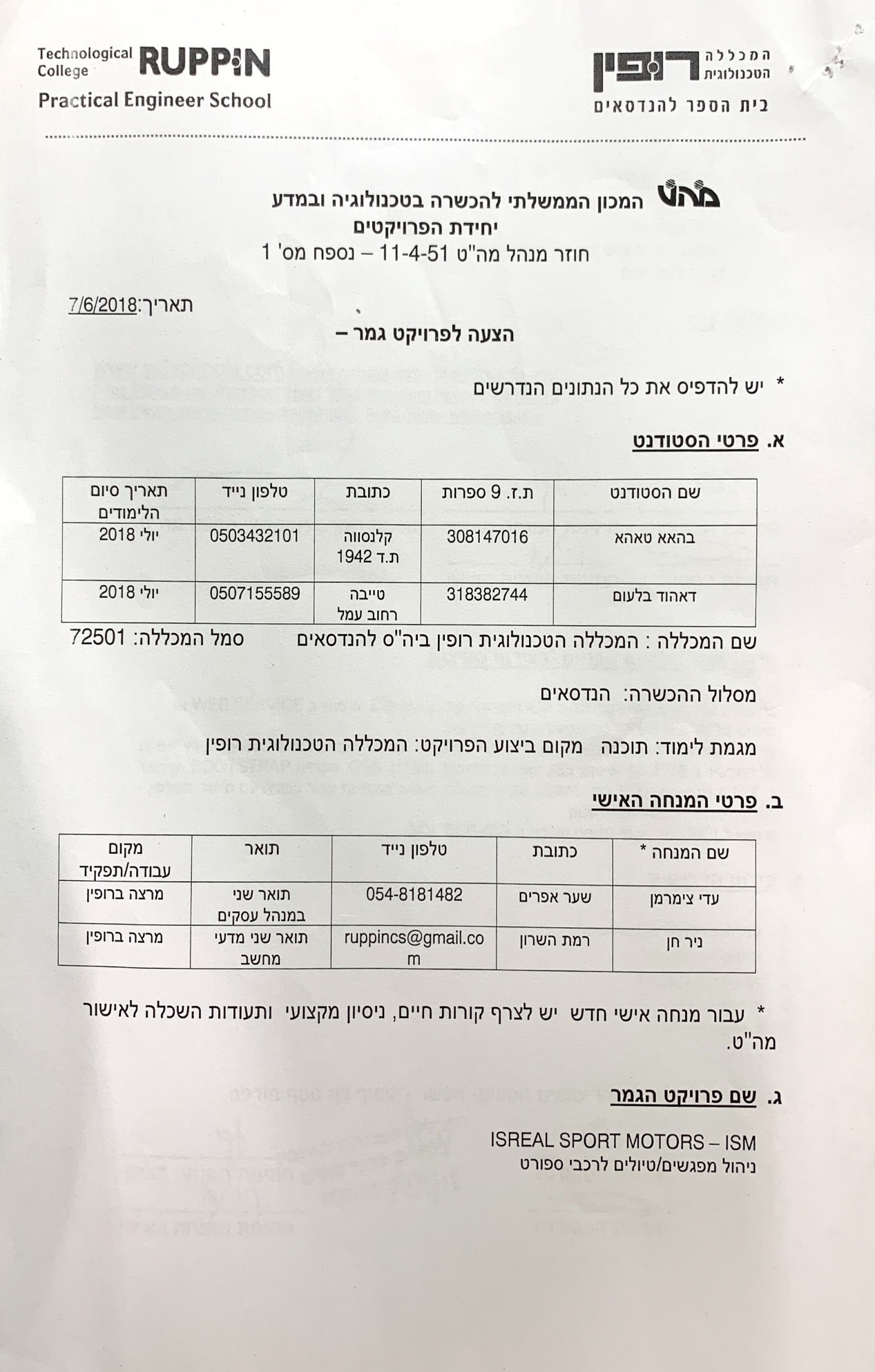
##### **ת.ז.:\_ 308147016**

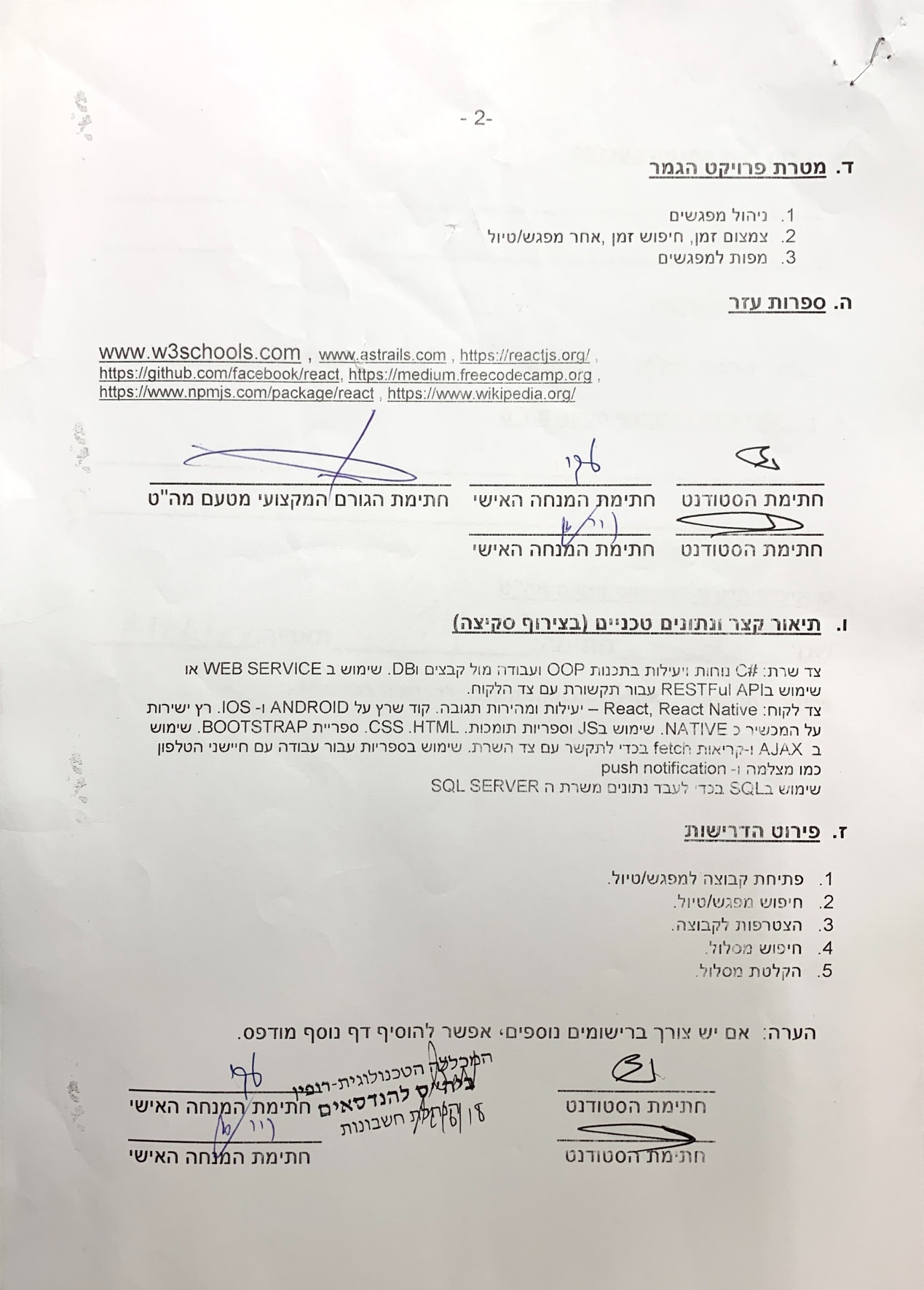
##### **המגיש/ה: דאהוד בלעום\_\_**

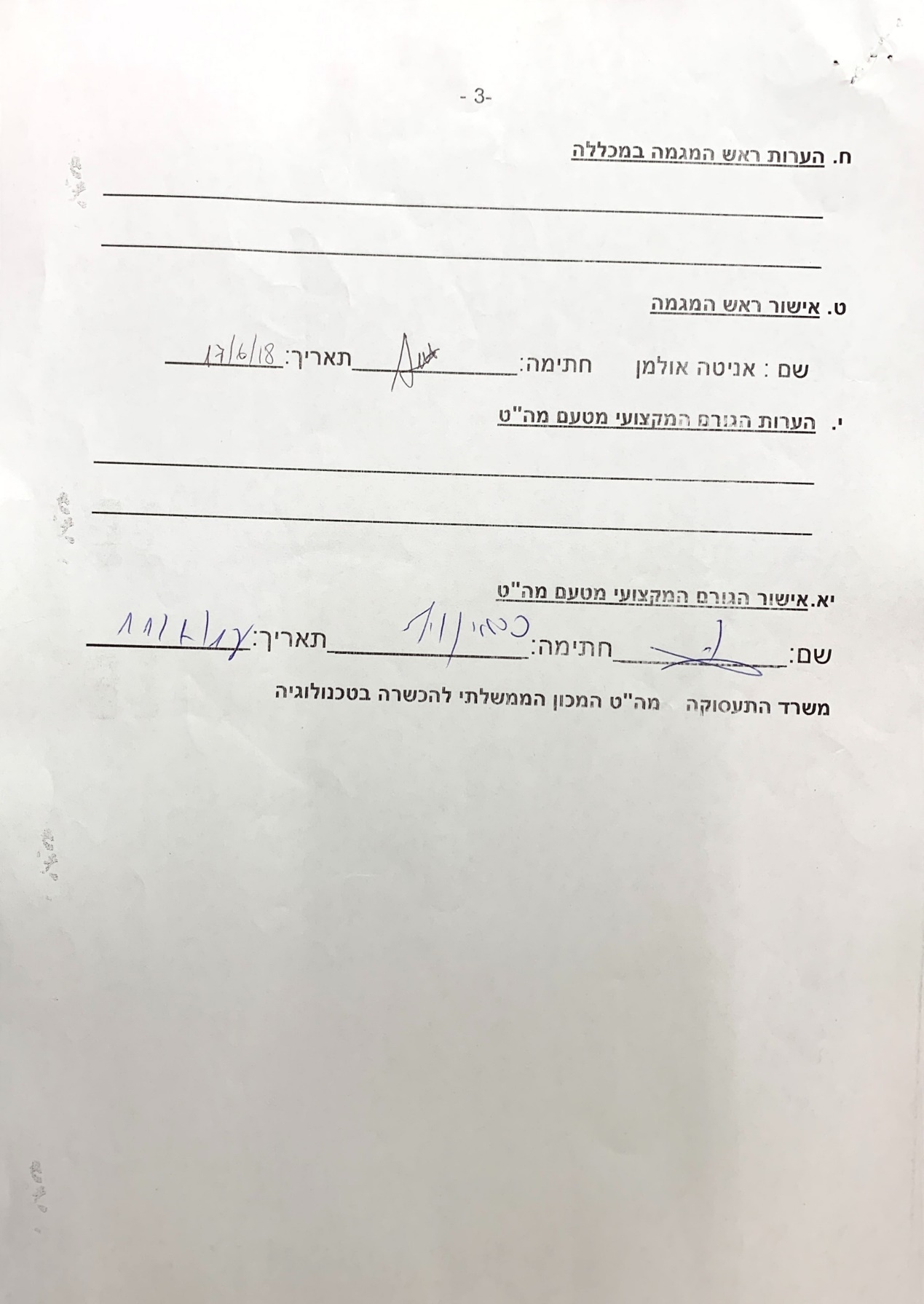
##### **ת.ז.:** **318382744**

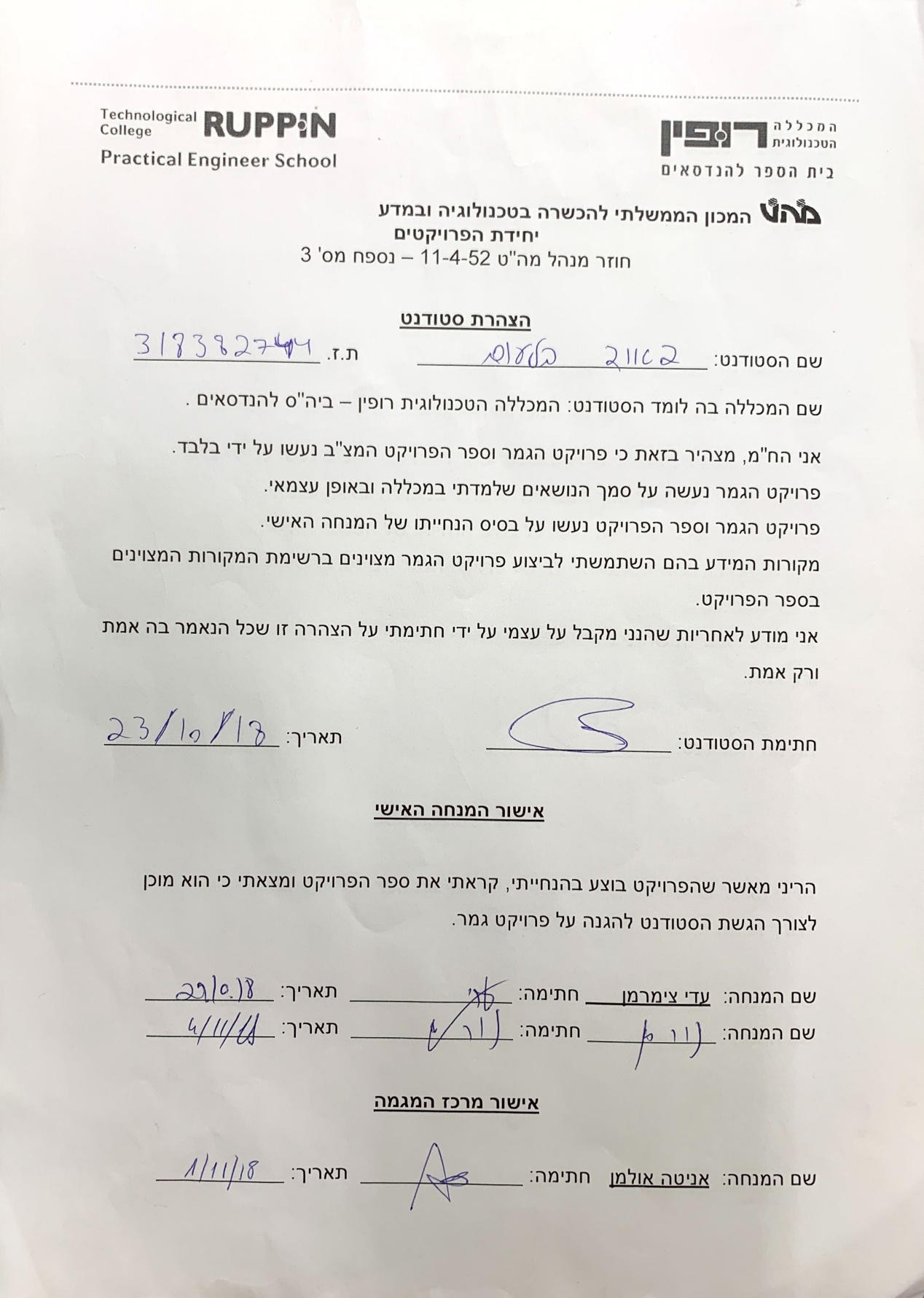
**המנחה: עדי צימרמן**

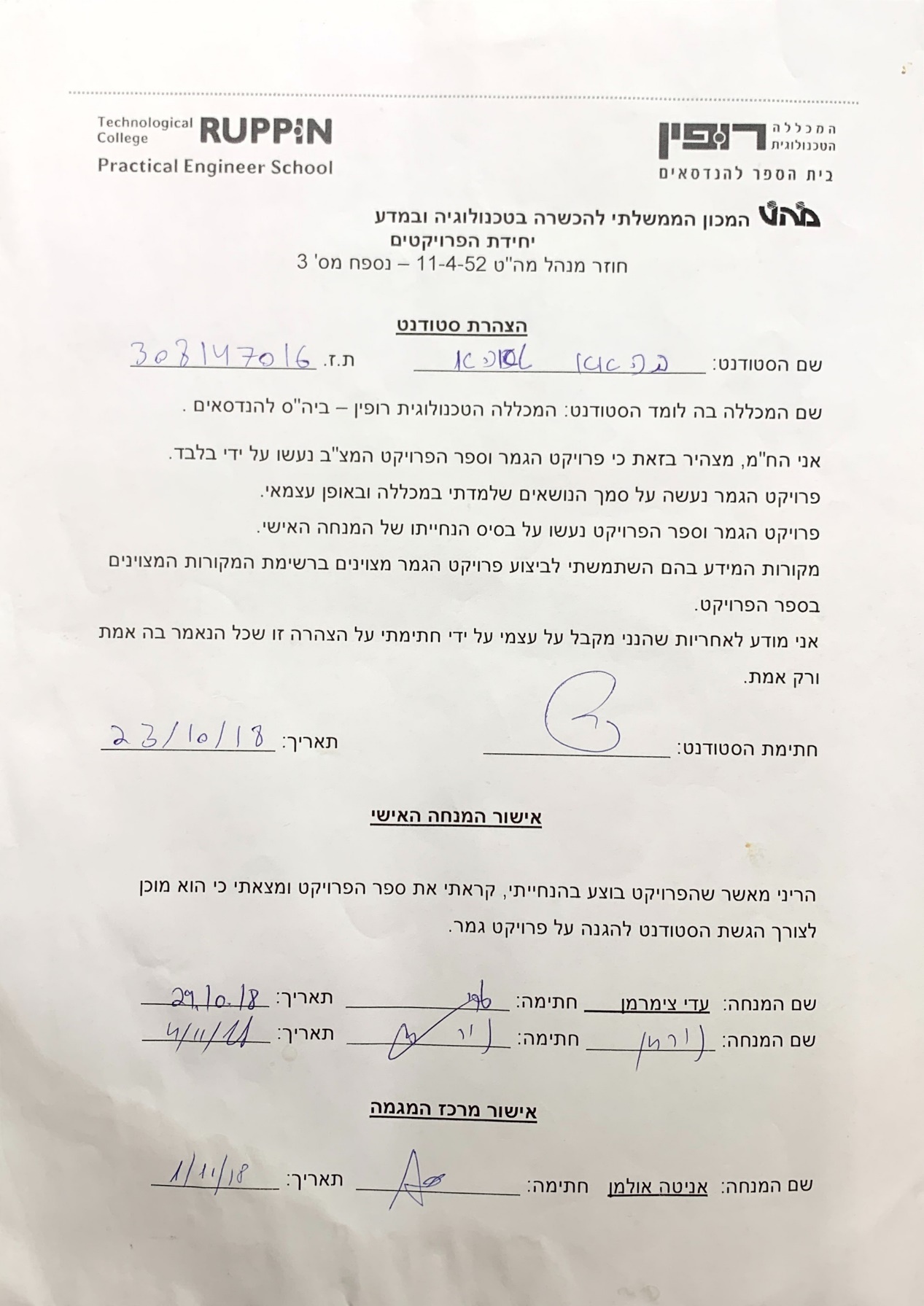
**המנחה: ניר חן**











1. **שם הפרויקט**

ISM – ISREAL SPORT MOTORT

ניהול מפגשים/טיולים לרכבי ספורט

1. **רקע**
   1. **תיאור ורקע כללי**

ישנם הרבה בעלי רכב שמטיילים. כדי להצטרף לטיול על בעל הרכב להיכנס לפייסבוק ולא תמיד ימצא הטיול/המסלול הרצוי

**2.2. מטרות המערכת**

**1.** ניהול מפגשים

2. צמצום זמן חיפוש זמן אחר מפגש/טיול

3. מפות למפגשים

1. **המצב בשוק והבעיות**
   1. **סקירת מצב קיים בשוק**

כיום אנשים שרוצים לארגן או להצטרף ל טיול/מפגש נכנסים לפיסבוק או כל אפלקציה חברתית כדי לחפש אם יש משהוא זמין ורוב הזמן לא מוצאים מה שהם מחפשים.

* 1. **בעיות במצב הקיים**:

1. **עמוד ענן**יתרונות
2. האפליקציה מציגה מידע על המקום המבוקש
3. האפליקציה מציגה מפה תלת ממד
4. אפשרות למשתמשים להוסיף מידע

חסרונות

1. לא ניתן לבצע מסלול לרכב
2. אין אפשרות לנהל מפגש
3. **Off-road**יתרונות
4. האפליקציה מבוססת על מפת אונליין מסוג צילום אוויר של גוגל

חסרונות

1. לא ניתן לערוך קבוצות לטיולים
2. לא ניתן למקד חיפוש לפי מיקום
3. נדרש הרבה זמן למציאת קבוצה לטיול.
4. לא ניתן לנהל מפגש/טיול בצורה מסודרת
5. האפליקציה מיועדת לטיולי ג'יפים
6. **מה הפרויקט אמור לחדש או לשפר**
7. צמצום זמן חיפוש אחר מפגשים/טיולים
8. האפליקציה תכיל רוב סוגי הטיולים/מפגשים (הכל במקום אחד)
9. ניתן יהיה לנהל מפגש/טיול
10. ניתן יהיה למקד חיפוש לפי פרמטר
11. **דרישות מערכת ופונקציונאליות**
    1. **דרישות מערכת**
12. זמן הצגת נתונים מרגע לחיצה על כפתור חיפוש – 2 שניות
13. 24 שעות לפני המפגש להצטרפות למפגש
14. 10000 משתמשים חיצוניים
15. עד 10 קבוצות למשתמש

* 1. **דרישות פונקציונאליות**

1. פתיחת קבוצה למפגש/טיול.
2. חיפוש מפגש
3. הצטרפות לקבוצה.
4. חיפוש מסלול.
5. שיתוף מפגשים
6. **בעיות צפויות במהלך הפיתוח ופתרונות** 
   1. **תיאור הבעיות**

(תפעוליות, טכנולוגיות, עומס ועוד) - הללו כפועל יוצא של דרישות המשתמש מהתוכנה.

לא ניתן להקליט מסלול -אין GBS   
לא ניתן להצטרף לקבוצה

* 1. **פתרונות אפשריים**

(נא ציין פתרונות אפשריים וחלופות ארכיטקטוניות).  
  
הקלטת מסלול באופן ידני על המפה

1. **פתרון טכנולוגי נבחר**
   1. **טופולוגית הפתרון**

אפליקציה: מהווה את צד הלקוח של המערכת. ממשק משתמש יעיל ונוח שיותקן מהחנות או מהתוסף EXPO אשר מותקן על המכשיר הנייד. האפליקציה תקושר לצד שרת אשר יורץ על שרתים של חברת LIVEDNS. השרתים מריצים סביבת.NET CORE התקשורת בין צד הלקוח לצד השרת תעשה ע"י תקשורת אינטרנט. המידע ישמר על גבי שרתי SQL SERVER אשר נמצאים גם כן בחברת LIVEDNS.

* 1. **טכנולוגיות בשימוש**

טכנולוגיות אשר תומכות בתכנות שרת-לקוח כמו: Asp Mvc.Core, Ajax

טכנולוגית EXPO אשר נותנות מענה לתכנות אפליקציות עבור ANDROIDו-IOS

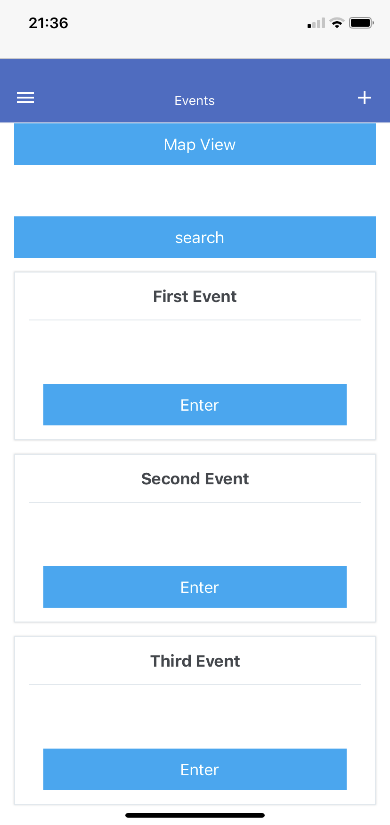
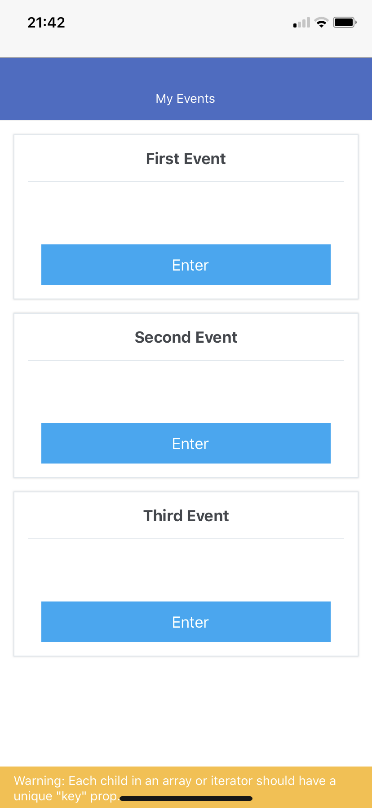
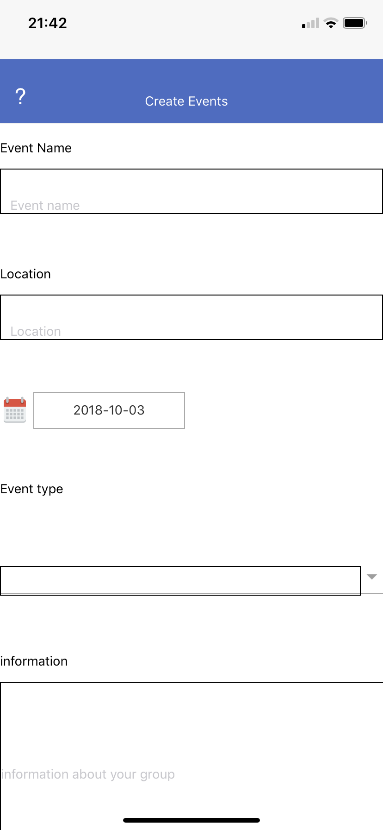
* 1. **שפות הפיתוח.**צד שרת: C# נוחות ויעילות בתכנות OOP ועבודה מול קבצים וDB. שימוש ב WEB SERVICE או שימוש בRESTFul API עבור תקשורת עם צד הלקוח.  
       
     צד לקוח: React, React Native – יעילות ומהירות תגובה. קוד שרץ על ANDROID ו- IOS. רץ ישירות על המכשיר כ NATIVE. שימוש בJS וספריות תומכות. HTML. CSS. ספריית BOOTSTRAP. שימוש ב AJAX ו-קריאות fetch בכדי לתקשר עם צד השרת. שימוש בספריות עבור עבודה עם חיישני הטלפון כמו מצלמה ו- push notification  
     שימוש בSQL בכדי לעבד נתונים משרת ה SQL SERVER

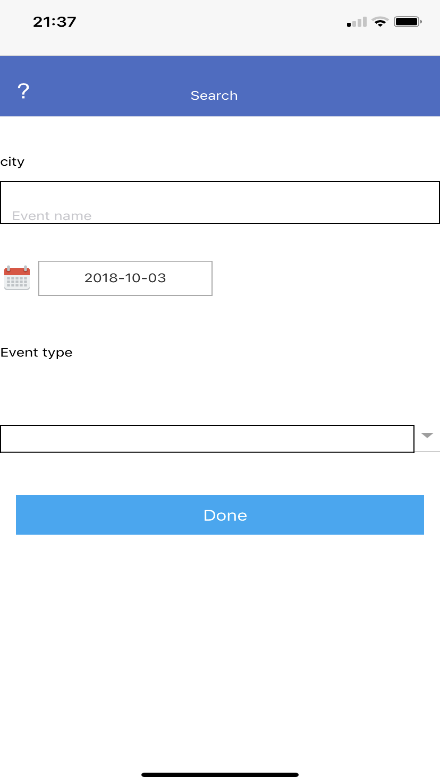
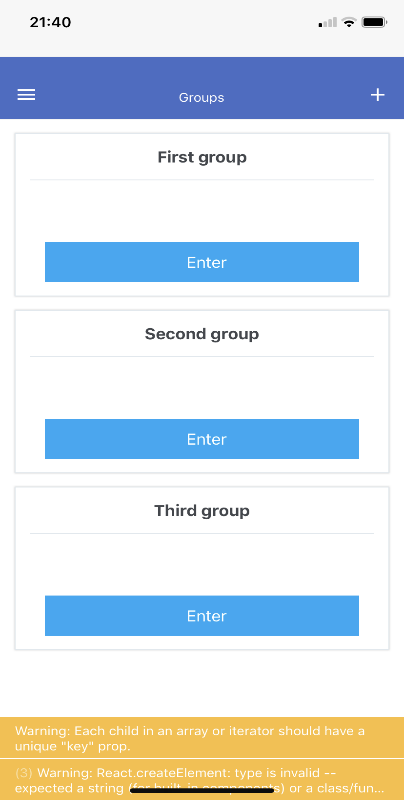
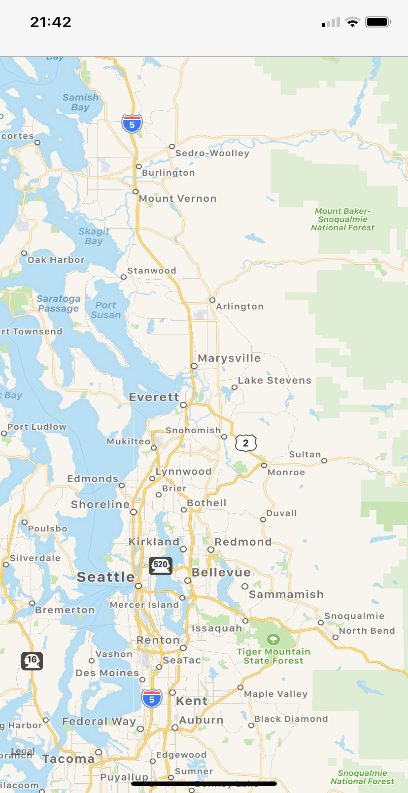
**תיאור הארכיטקטורה הנבחרת**חלוקה לשלוש השכבות : GUI, BAL, DAL כ MVC.  
View יקודד כצד לקוח ע"י React, React Native.  
Controller ו- Model יתוכנתו בצד שרת ע"י C# בWEB SERVICE או RESTFul API

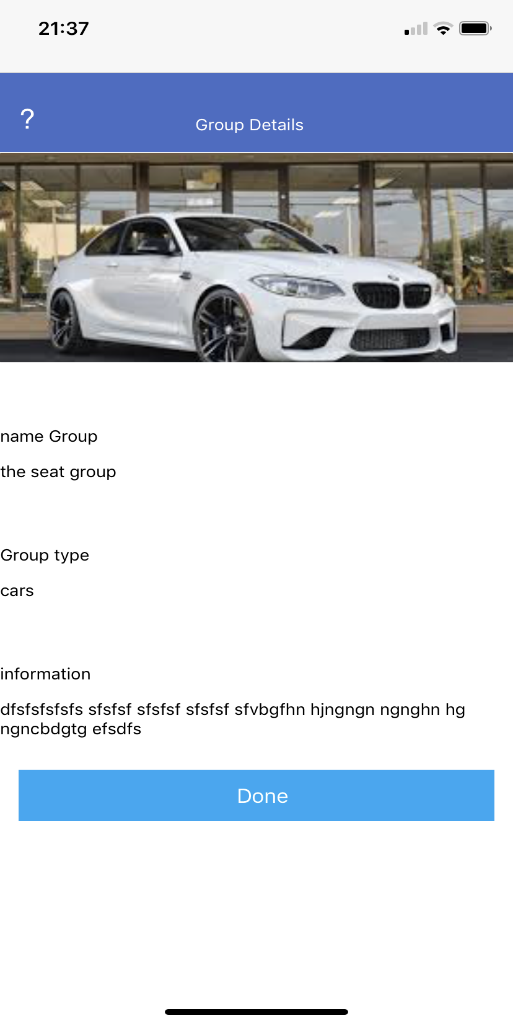
* 1. **תיאור הארכיטקטורה הנבחרת**חלוקה לשלוש השכבות : GUI, BAL, DAL כ MVC.  
     View יקודד כצד לקוח ע"י React, React Native.  
     Controller ו- Model יתוכנתו בצד שרת ע"י C# בWEB SERVICE או RESTFul API
  2. **חלוקה לתכניות ומודולים**

**סביבת השרת**עבודה שוטפת עבודה בסביבה מקומית ואז העלאה לשרת מארח של LIVEDNS.

* 1. **ממשק המשתמש/לקוח – GUI**

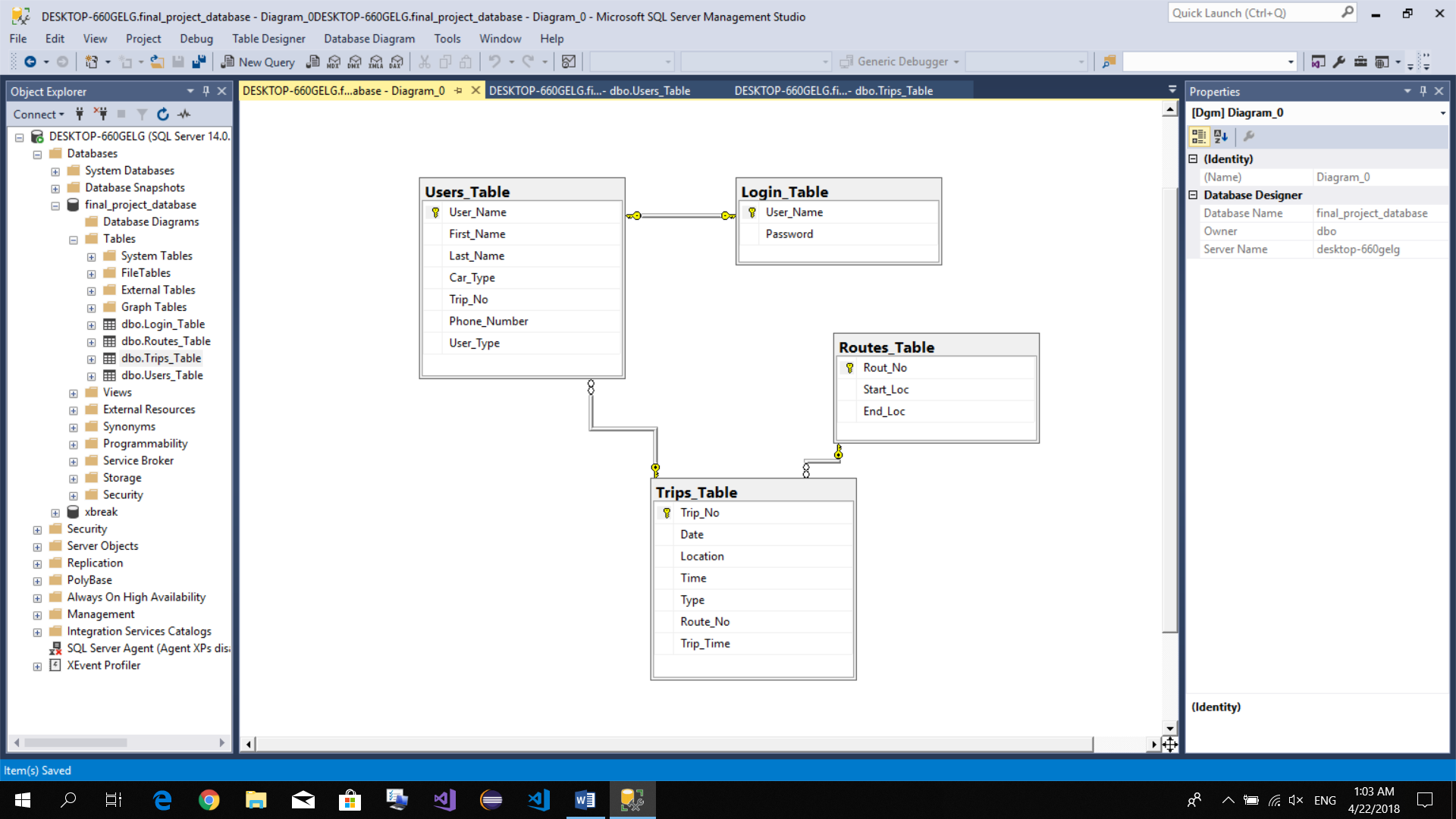
  





* 1. **ממשקים למערכות אחרות/API**עבודה עם API של EXPO לתמיכה בחיישניי הטלפון הנייד למשל מצלמה או שימוש בpush notification כמו גם האפשרות להריץ קוד על IOS ללא המעבר והתשלום עבור פתיחת משתמש מפתח לIOS.
  2. **שימוש בחבילות תוכנה**שימוש React, React Native  
     שימוש ב material-ui   
     שימוש ב BOOTSTRAP  
     עבודה עם ספריות של EXPO לתמיכה בחיישניי הטלפון הנייד למשל מצלמה או שימוש בpush notification כמו גם האפשרות להריץ קוד על IOS ללא המעבר והתשלום עבור פתיחת משתמש מפתח לIOS.

1. **שימוש במבני נתונים וארגון קבצים** 
   1. **מבני הנתונים**





**משתמשים**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| שם משתמש | אותיות ומספרים. שדה חובה, מפתח | nvarchar | User name |
| שם | רק אותיות | nvarchar | First name |
| **משפחה** | **רק אותיות** | **nvarchar** | **Last name** |
| רכב | אותיות ומספרים | nvarchar | Car type |
| טיול/מפגש | אותיות ומספרים | nvarchar | Trip |
| מס טלפון |  | numric | Phone number |
| סוג משתמש |  | nvarchar | User type |

**טיולים**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| מס טיול/מפגש | מספר, שדה חובה מפתח | numric | Meeting no. |
| תאריך | תאריך | datetime | Date |
| מיקום | אותיות | nvarchar | Location |
| שעה | מספר | time | Time |
| סוג הטיול/מפגש | אותיות | nvarchar | Type |
| מסלול |  |  | Map |
| זמן טיול/מפגש | שעות | time | Meeting time |

**מסלול**

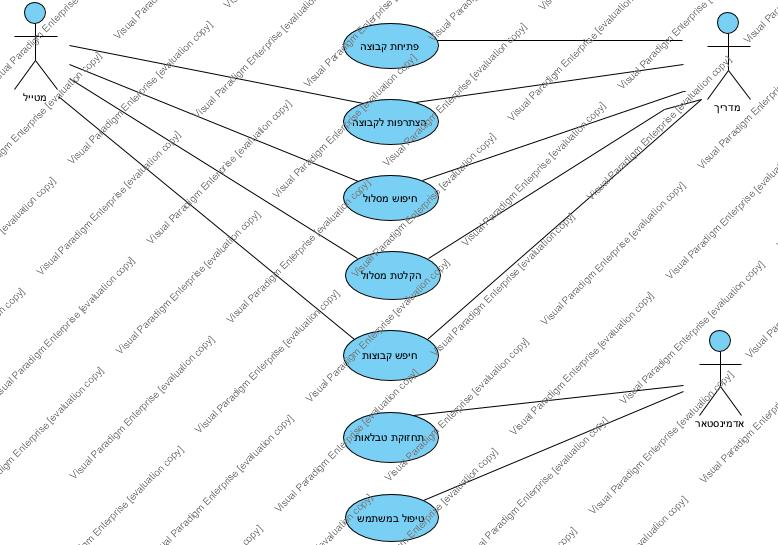
|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| מס מסלול | חובה | numric | Route no. |
| נקודת התחלה | מיקום | nvarchar | Start loct |
| נקודת סיום | מיקום | nvarchar | End loct |

**התחברות**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| שם משתמש | חובה ,אותיות ומספרים מפתח | nvarchar | User name |
| סיסמה | חובה, אותיות ומספרים | nvarchar | Password |

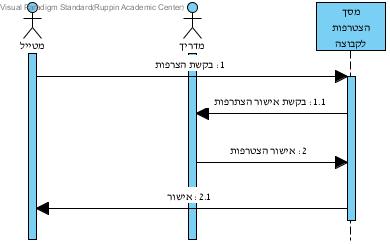
* 1. **שיטת האחסון**שימוש ב ms sql לבניית database בשיטת טבלאות
  2. **מנגנוני התאוששות**   
     יעשה שימוש בטרנזקציות בעדכון מחיקה והכנסה של נתונים למאגר על מנת לשחזר נתונים בזמן התאוששות

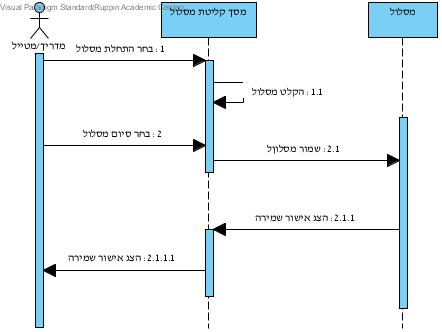
1. **תרשימי מערכת מרכזיים**

**9.1. Use Case  
  
**

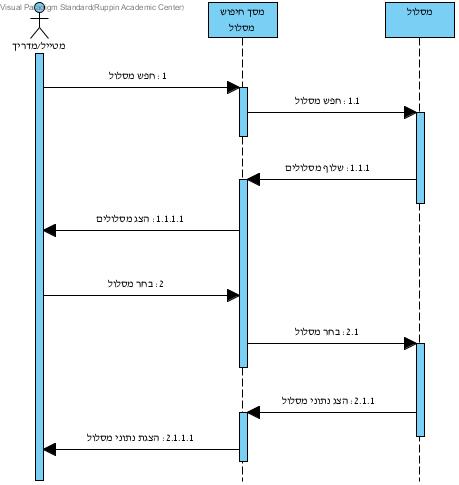
**9.2. Sequence Diagram**

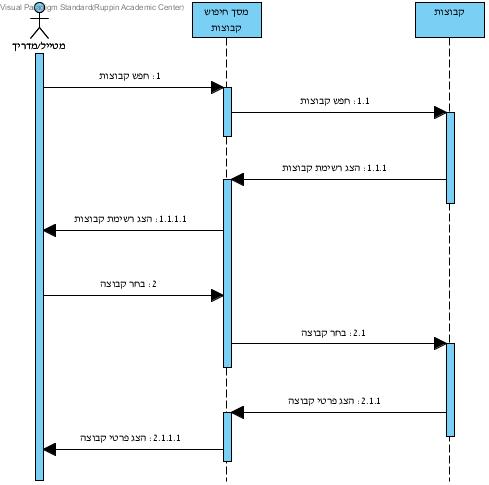
רצף קריאות פונקציות מרכזיות בלוגיקה העסקית המרכזית של הפרויקט

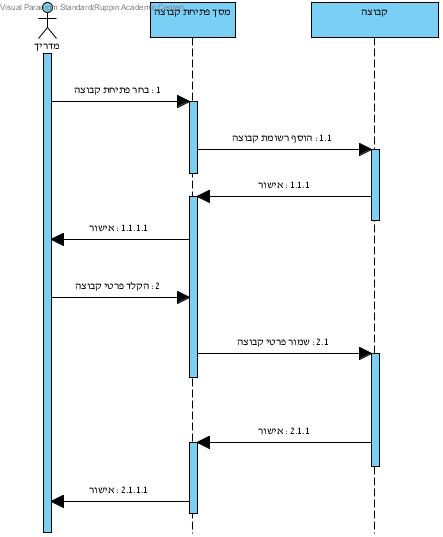




.

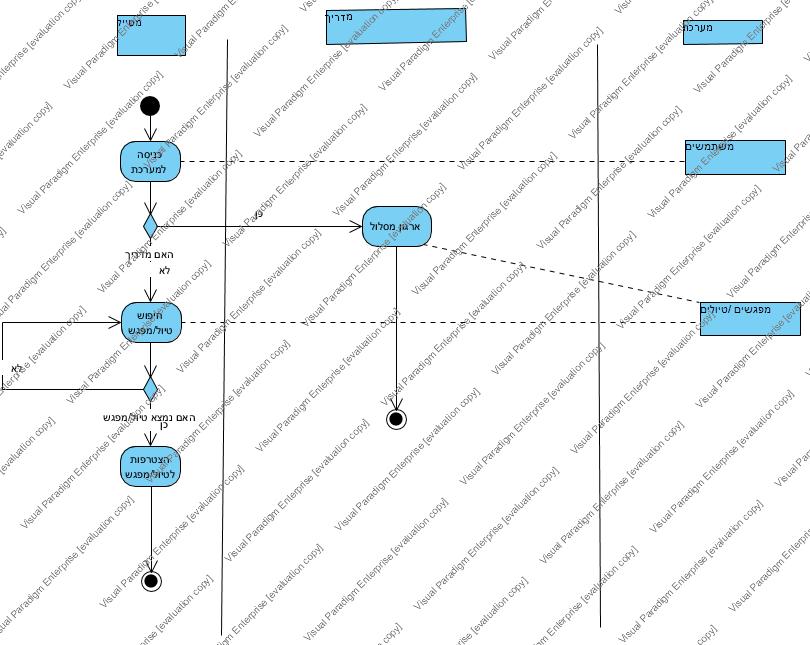






**9.3Data Flow**

Activity Diagram

****

1. **תיאור המרכיב האלגוריתמי - חישובי**

**10.1 הבעיה שבא לפתור**שילוב בין מטייל לקבוצה על ידי התאמת פרמטרים כמו שעת מפגש, מיקום המפגש, וסוג מפגש

**10.2 איסוף מידע וניתוחים סטטיסטיים** כמות משתמשים   
 זמן חיבור ממוצע

כמות קבוצות

1. **אבטחת מידע**

באמצעות דף login ונתינת הרשאות לכל משתמש

1. **משאבים הנדרשים לפרויקט**

**12.1 מספר שעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות**

מספר השעות שהוקדשו לפרויקט סה"כ כ500 שעות כל אחד

**12.2 ציוד נדרש**

מחשב, שרת שמותקן עליו 2018, SQL SERVER

**12.3 תוכנות נדרשות**

VISUAL PRYDIGM…

Visual studio

Visual code

**12.4 ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט**

ANGULAR, MVC,

**12.5 ספרות ומקורות מידע**

* [www.w3schools.com](http://www.w3schools.com)
* [www.astrails.com](http://www.astrails.com)
* <https://reactjs.org/>
* <https://github.com/facebook/react>
* <https://medium.freecodecamp.org>
* <https://www.npmjs.com/package/react>

1. **תכנית עבודה ושלבים למימוש הפרויקט**

* ייזום – ימשך עד סוף 4/2018
* אפיון – ימשך עד סוף 6/2018
* עיצוב המערכת וקידוד – ימשך עד סוף 10/2018
* בדיקות ותיקונים – במהלך חודש 11/2018
* הגשת הפרוייקט – 30/11/2018
* הגנה על הפרוייקט סוף 1/2019

1. **בדיקות**

14.1 **בדיקות תהליכיות**

|  |  |  |  |
| --- | --- | --- | --- |
| # | תיאור צעד | תוצאה רצויה | תוצאה מתקבלת |
| 1 | המשתמש בוחר איזו אירוע הוא רוצה להשתתף | האירוע נאסף מה db אם קיים | האירוע נאסף מה db אם קיים |
| 2 | פרטי האירוע מופיעים על המסך | פרטי האירוע נאספו מ ה db | פרטי האירוע נאספו מ ה db |
| 3 | לחץ על הוספה לאירוע | הודעה הוספת בהצלחה ו חוזר לדף האירועים | הודעה הוספת בהצלחה ו חוזר לדף האירועים |
| 4 | המשתמש רוצה לייצר אירוע | המשתמש מקליט פרטי האירוע | המשתמש מקליט פרטי האירוע |
| 5 | הודעה יוצר בהצלחה | פרטי האירוע נקלטו ב db | פרטי האירוע נקלטו ב db |

**14.2**

**בדיקות יחידה**

|  |  |  |  |
| --- | --- | --- | --- |
| # | תיאור צעד | תוצאה רצויה | תוצאה מתקבלת |
| 1 | המשתמש מכניס פרטים למערכת | הפרטים נבדקים מול db | הפרטים נבדקים מול db |
| 2 | בודק האם קיים משתמש במערכת | הודעה "המשתמש קיים במערכת" | הודעה "המשתמש קיים במערכת" |
| 3 | בקש משתמש אחר | נא לרשום משתמש אחר | נא לרשום משתמש אחר |
| 4 | המשתמש מכניס שם משתמש וסיסמה | הפרטים נבדקים מול db | הפרטים נבדקים מול db |
| 5 | בודק האם קיים משתמש במערכת | הודעה "המשתמש לא קיים במערכת" | הודעה "המשתמש לא קיים במערכת" |

1. **בקרת גירסאות**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם רכיב | סוג רכיב | גירסה | תאריך | שינויים |
| מסמך אפיון | מסמך | 1.0 | 7/6/2018 | מקור |
| אפלקציית ISM | קוד | 1.0 | 31/7/2018 | מקור |
| מסמך אפיון | מסמך | 2.0 | 22/11/2018 | * עדכון פונקציונליות בפרק 5 ואיפה שנדרש באפיון * החלפת עיצובי מסכים בצילומי מסך אמיתיים – פרק 7.7 * עדכון מבנה בסיס הנתונים – פרק 8 |
| מסמך אפיון | מסמך | 2.0 | 27/11/2018 | * עדכון פונקציונליות בתרשימים - פרק 9, 7.5 * הוספת פרקים 14, 15, 16 |

1. **קטעי קוד**

*let* urlAPI = "http://ruppinmobile.tempdomain.co.il/site08/ism-asmx.asmx";

GetGroups = () *=>* {

fetch(urlAPI + '/GetGroups', {

method: 'POST',

body: JSON.stringify({

}),

headers: {

"Content-type": "application/json; charset=UTF-8"

}

})

.then(*res* *=>* res.json())

.then(*res* *=>* {

*console*.log('res=' + res);

*console*.log('res.d=' + res.d);

*let* userObj = JSON.parse(res.d);

*console*.log(userObj);

},

(*error*) *=>* {

alert("error-" + error);

}

);

}

export default *class* Groups extends React.*Component* {

*constructor*(*props*) {

super(props);

fetch(urlAPI + '/GetGroups', {

method: 'POST',

body: JSON.stringify({

}),

headers: {

"Content-type": "application/json; charset=UTF-8"

}

})

.then(*res* *=>* res.json())

.then(*res* *=>* {

*console*.log('res=' + res);

*console*.log('res.d=' + res.d);

this.Groups = JSON.parse(res.d);

*console*.log(this.Groups);

},

(*error*) *=>* {

alert("error-" + error);

}

);

}

render() {

return (

<*View* style={styles.container}>

<*Header*

leftComponent={{ icon: 'menu', color: '#fff' }}

centerComponent={{ text: 'Groups', style: {color:'#fff' } }}

rightComponent={{ icon: 'add', color: '#fff' }}

/>

<*ScrollView* style={{ minHeight: '10%' }}>

<*GroupsList*

navigation={this.props.navigation}

Groups={this.Groups}

/>

</*ScrollView*>

</*View*>

);

}

}

*class* GroupItem extends React.*Component* {

*constructor*(*props*) {

super(props);

}

\_onPressButton = () *=>* {

// this.props.navigation.navigate('CheckCar');

}

render() {

return (

<*Card*

title={this.props.Name}

image={{ uri: this.props.Type }}

>

<*Text* style={{ margin: 20, marginLeft: '40%' }}>

{this.props.Type}

</*Text*>

<*Text* style={{ margin: 20, marginLeft: '40%' }}>

{this.props.Information}

</*Text*>

<*Button*

// icon={<Icon name='code' color='#ffffff' />}

backgroundColor='#03A9F4'

buttonStyle={{ borderRadius: 0, marginLeft: 0, marginRight: 0, marginBottom: 0 }}

title='Enter'

onPress={this.\_onPressButton} />

</*Card*>

);

}

}

*class* GroupsList extends React.*Component* {

*constructor*(*props*) {

super(props);

this.Groups = this.props.Groups;

// GetGroups();

this.Llist = this.Groups.map((*obj*, *index*) *=>* {

return (

<*GroupItem*

navigation={this.props.navigation}

Id={obj.Id}

Name={obj.Name}

Type={obj.Type}

Information={obj.Information}

key={obj.Id}

/>

);

});

}

render() {

return (

<*View*>

{this.Llist}

</*View*>

);

}

}

export default *class* CreateGroup extends React.*Component* {

*constructor*() {

super();

this.groupName = '';

this.groupType = '';

this.groupInfo = '';

}

\_onPressButton = () *=>* {

fetch(urlAPI + '/InsertGroup', {

method: 'POST',

body: JSON.stringify({

Name: this.groupName,

Type: this.groupType,

Information: this.groupInfo

}),

headers: {

"Content-type": "application/json; charset=UTF-8"

}

})

.then(*res* *=>* {

Alert.alert('SUCCESS');

// this.props.navigation.navigate('Home');

},

// Note: it's important to handle errors here

// instead of a catch() block so that we don't swallow

// exceptions from actual bugs in components.

(*error*) *=>* {

alert("error-" + error);

});

}

render() {

return (

<*View* style={styles.container}>

<*Header*

leftComponent={{ icon: 'arrow-left', color: '#fff' }}

centerComponent={{ text: 'Create Group', style: { color: '#fff' } }}

/>

<*ScrollView* style={{ minHeight: '10%' }}>

<*Image* source={require('../assets/EVENTPHOTO.jpg')} style={{ width: '100%' , borderRadius:0.5 }} />

<*Text*>

Group Name:

</*Text*>

<*Text* />

<*TextInput*

value={this.groupName}

placeholder={'Group Name'}

style={styles.input}

onChangeText={(*groupname*) *=>* this.groupName = groupname}

/>

<*Text*>

Group type

</*Text*>

<*Text* />

<*TextInput*

value={this.groupType}

placeholder={'Group type'}

style={styles.input}

onChangeText={(*grouptype*) *=>* this.groupType = grouptype}

/>

<*Text*>

Information

</*Text*>

<*TextInput*

value={this.groupInfo}

placeholder={'Information about your group'}

style={styles.input1}

onChangeText={(*groupinfo*) *=>* this.groupInfo = groupinfo}

/>

<*Button*

icon={<*Icon* name='code' color='#ffffff' />}

backgroundColor='#03A9F4'

buttonStyle={{ borderRadius: 0, marginLeft: 0, marginRight: 0, marginBottom: 0 }}

title='Done'

onPress={this.\_onPressButton}

/>

</*ScrollView*>

</*View*>

);

}

}

public class DBservices

{

string s = ConfigurationManager.ConnectionStrings["LIVEDNS"].ConnectionString;

private SqlConnection con = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataAdapter da = new SqlDataAdapter();

DataSet ds = new DataSet();

DataTable tbGroups=null;

public string Login(string user, string pass)

{

con = new SqlConnection(s);

try

{

con.Open();

cmd = new SqlCommand($"SELECT \* FROM Users WHERE Username='{user}' and Password='{pass}' "

, con);

SqlDataReader reader = cmd.ExecuteReader();

string ree = "";

while (reader.Read())

{

ree += reader["Email"].ToString() + "," + reader["Username"].ToString()

+ "," + reader["Password"].ToString();

}

return ree;

}

catch

{

throw;

}

finally

{

if (con.State == ConnectionState.Open)

{

con.Close();

}

}

}

public string GetGroups()

{

con = new SqlConnection(s);

try

{

con.Open();

da = new SqlDataAdapter($"SELECT \* FROM Groups ", con);

if (ds.Tables["tbGroups"] != null)

{

ds.Tables["tbGroups"].Clear();

}

da.Fill(ds, "tbGroups");

tbGroups = ds.Tables["tbGroups"];

return new JavaScriptSerializer().Serialize(tbGroups)

}

catch

{

throw;

}

finally

{

if (con.State == ConnectionState.Open)

{

con.Close();

}

}

}

public int InsertGroup(string name, string type, string information)

{

con = new SqlConnection(s);

con.Open();

try

{

cmd = new SqlCommand("insert into Groups(Name,Type,Information) " +

"values ('" + name + "','" + type + "','" + information + "')", con);

return cmd.ExecuteNonQuery();

}

catch

{

throw;

}

finally

{

con.Close();

}

}

public string GetEvents()

{

con = new SqlConnection(s);

try

{

con.Open();

cmd = new SqlCommand($"SELECT \* FROM Events ", con);

SqlDataReader reader = cmd.ExecuteReader();

string ree = "";

while (reader.Read())

{

ree += reader["Name"].ToString() + ',' + reader["Location"].ToString() + ',' +

reader["Date"].ToString() + ',' + reader["Type"].ToString() + ',' +

reader["StartTime"].ToString() + ',' + reader["EndTime"].ToString() + ',' +

reader["Information"].ToString() + ',' + reader["imgPath"].ToString();

}

return ree;

}

catch

{

throw;

}

finally

{

if (con.State == ConnectionState.Open)

{

con.Close();

}

}

}

public int InsertEvent(string name, string location, string date, string strat, string end,

string type, string information, string img)

{

con = new SqlConnection(s);

con.Open();

try

{

cmd = new SqlCommand("insert into Events(Name,Location,Date,StartTime,EndTime,Type,Information,ImgPath" + ") " +

"values ('" + name + "','" + location + "','" + date + "','" + strat + "','" + end + "','"

+ type + "','" + information + "','" + img + "')", con);

return cmd.ExecuteNonQuery();

}

catch

{

throw;

}

finally

{

con.Close();

}

}

public int inset(string Email, string Username, string Password)

{

con = new SqlConnection(s);

con.Open();

try

{

cmd = new SqlCommand("insert into Users(Email,Username,Password) values ('" +

Email + "','" + Username + "','" + Password + "')", con);

return cmd.ExecuteNonQuery();

}

catch

{

throw;

}

finally

{

con.Close();

}

}

}