The State of the State

זיו כהן <zivc888@gmail.com ,326178266>

תאריך הגשת הפרויקט: 23/4/23

מנחה: דני בן ישי

בית הספר: אורט ע"ש שמעון פרס, יקנעם עילית

תוכן עניינים:

מבוא:

הרקע לפרויקט:

שם הפרויקט: The State of the State

תיאור קצר: הפרויקט מכיל בתוכו מספר שאלות הנוגעות לאקלים הפוליטי במדינת ישראל ומאפשר למשתמשים להביע את דעתם על הסוגיות המוצגות ובמקביל לראות את דעותיהם של שאר המשתמשים.

קהל היעד: תושבים ישראלים המעוניינים ב-"תמונת מצב" על הדעות הרווחות במדינה.

הסיבות לבחירת הנושא: חשבתי על הרעיון לפרויקט ב-27/3/23 כאשר ראש הממשלה בנימין נתניהו פיטר את שר הביטחון יואב גלנט על רקע התנגדותו לרפורמה המשפטית שמובילה הממשלה, מהלך שהוביל ללילה סוער של הפגנות משני צידי המתרס. לאחר מקרה זה, התחלתי לחשוב על כך שמכיוון שיש הפגנות משני צידי המתרס, לא ניתן להבין מהי למעשה הדעה הרווחת בעם בנוגע למהלכים מדיניים ועל כן החלטתי למצוא פתרון לבעיה.

תהליך המחקר:

נכון לכעת, סיקור הדעות הפוליטיות בקרב הציבור הישראלי הוא נחלתם הבלעדית של סוקרים מקצועיים מטעם חברות החדשות ופוליטיקאים. לעיתים, אתרי חדשות למיניהם מאפשרים למשתמשים להצביע על סקרים, אולם תוצאות הסקר אינן מופיעות לאחר ההצבעה ולעיתים קרובות עוברים הנתונים מניפולציות סטטיסטיות על מנת לשרת נרטיב מסוים (כפי שניתן ללמוד, למשל, מהפערים המשמעותיים בין סקריו של ערוץ 14 לשאר הערוצים הגדולים).

בנוסף, סקרים פוליטיים נעשים נפוצים בשני מצבים: לקראת בחירות ובעיתות משבר פוליטיות. לעומת זאת, בתקופות "שגרה" קשה עד מאוד לדעת את דעת הקהל הישראלי על סוגיות פוליטיות.

אתגרים מרכזיים:

בעיות שדרשו התמודדות במהלך הפיתוח: היו 2 בעיות עיקריות שהקשו עליי בפיתוח הפרויקט – הראשונה, חשבתי על הרעיון זמן קצר יחסית לפני מועד ההגשה ולכן הייתי צריך לסיים את הפרויקט תוך פרק זמן קצר. השנייה, על מנת לממש את הפרויקט נדרשתי ללמוד כיצד לייצר גרפים ב-C# במטרה להציג את תוצאות הסקרים באופן ויזואלי.

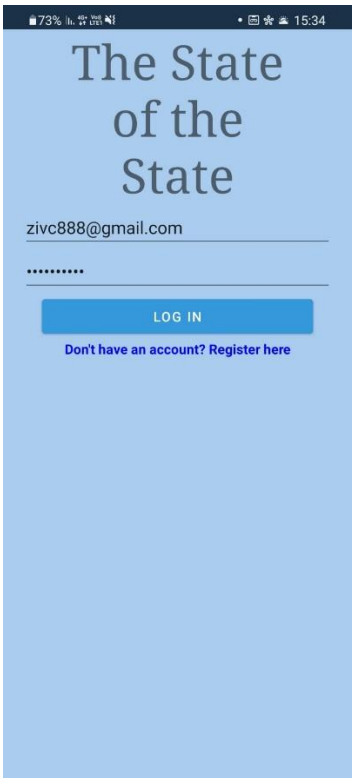
הצורך עליו הפרויקט עונה / איזה פתרון הוא נותן: הפרויקט מאפשר למשתמש ישראלי המעוניין לדעת את "תמונת המצב" של דעת הקהל על המהלכים הפוליטיים במדינה, להביע את דעתו ולחזות בדעותיהם של משתמשים אחרים באופן מיידי.

מבנה / ארכיטקטורה:

תיאור מסכי הפרויקט:

מסך התחברות (מסך הפתיחה):

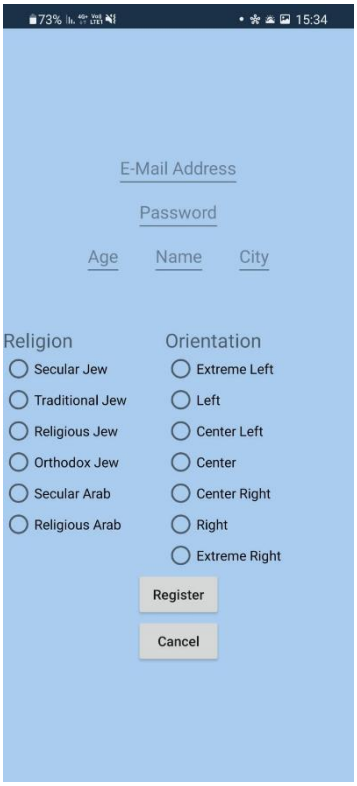
מסך זה מאפשר למשתמש להתחבר לחשבונו ולהתחיל להשתמש באפליקציה.



אלמנט	תפקיד
קלט מייל (EditText)	לקלוט את המייל של המשתמש.
קלט סיסמה (EditText)	לקלוט את הסיסמה של המשתמש.
כפתור התחברות (Button)	להכניס את המשתמש לאפליקציה במידה והפרטים שהזין תואמים משתמש קיים.
"לינק" הרשמה (TextView)	להעביר את המשתמש למסך Register על מנת לפתוח חשבון חדש.

מסך רישום:

מסך זה מאפשר למשתמש לפתוח חשבון חדש באפליקציה.



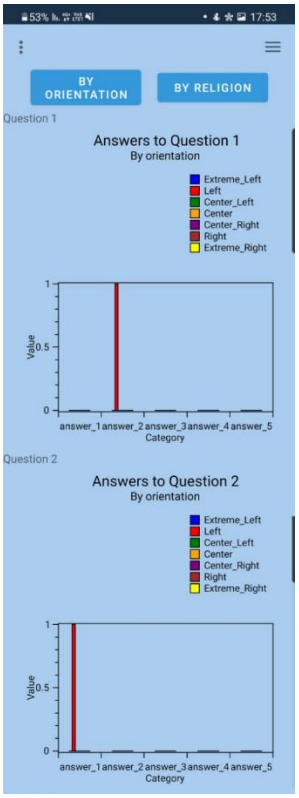
אלמנט	תפקיד
קלט מייל (EditText)	לקלוט את המייל של המשתמש.
קלט סיסמה (EditText)	לקלוט את הסיסמה של המשתמש.
קלט גיל (EditText)	לקלוט את הגיל של המשתמש.
קלט שם (EditText)	לקלוט את השם של המשתמש.
קלט עיר (EditText)	לקלוט את העיר של המשתמש.
דת (RadioGroup)	לקלוט את מידת אמונתו הדתית של המשתמש.
עמדה פוליטית (RadioGroup)	לקלוט את נטייתו הפוליטית של המשתמש.
כפתור הרשמה (Button)	לפתוח משתמש חדש ב-FireBase על סמך הנתונים שהוזנו ולחזור למסך ההתחברות.
כפתור ביטול (Button)	לחזור למסך ההתחברות מבלי לשמור את המשתמש.

מסך בית (סקרים):

מסך זה מאפשר למשתמש להצביע בסקרים.

אלמנט	תפקיד
עבור כל שאלה – תשובות אפשריות (RadioGroup)	לקלוט את המייל של המשתמש.
כפתור שליחה (Button)	מעלה את ההצבעות של המשתמש ל-FireBase.
סרגל הכלים (Toolbar)	ה-NavigationView (צד ימין) מאפשר הצגה של שם המשתמש והניקוד. ה-Menu (צד שמאל) מאפשר מעבר למסך Results.

הערה: התמונה "מוארכת" על מנת לכלול את כל תוכן ה-ScrollView.



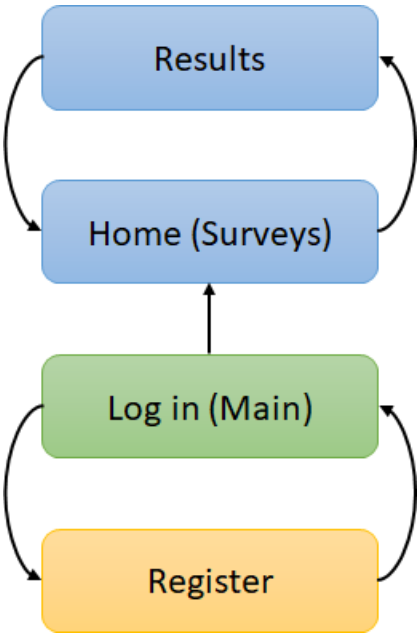
מסך תוצאות:

מסך זה מאפשר למשתמש לפתוח חשבון חדש באפליקציה.

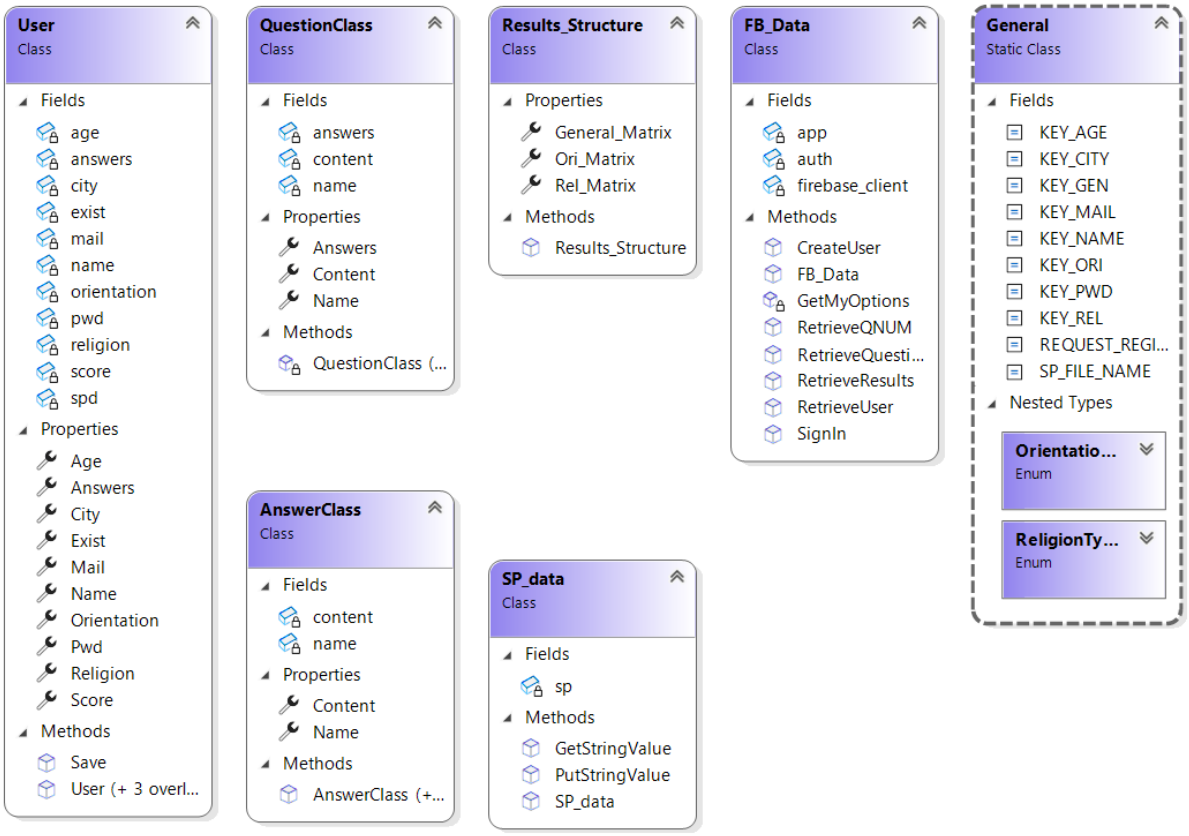
אלמנט	תפקיד
עבור כל שאלה – גרף התשובות (PlotView)	מציג את תשובות כלל המשתמשים.
כפתור "By Orientation" (Button)	משנה את הגרפים כך שיציגו התפלגות לפי עמדה פוליטית.
כפתור "By Religion" (Button)	משנה את הגרפים כך שיציגו התפלגות לפי אמונה דתית.
סרגל הכלים (Toolbar)	ה-NavigationView (צד ימין) מאפשר הצגה של שם המשתמש והניקוד. ה-Menu (צד שמאל) מאפשר מעבר למסך Results.

הערה: התמונה "מוארכת" על מנת לכלול יותר מתוכן ה-ScrollView.

תרשים מסכים:



תיאור מחלקות הפרויקט:



מימוש הפרויקט:

מחלקה 1 - User:

תפקיד: אחסון המידע על המשתמש.

שדות:

שם השדה	תחום הכרה	תפקיד ושימוש
name	public	אחסון שם המשתמש.
mail	public	אחסון כתובת המייל של המשתמש.
pwd	public	אחסון סיסמת המשתמש.
city	public	אחסון עיר המשתמש.
age	public	אחסון גיל המשתמש.
religion	public	אחסון דת המשתמש.
orientation	public	אחסון עמדתו הפוליטית של המשתמש.
exist	public	מציין האם קיים משתמש שמור ב-FireBase שמכיל את נתוני ההזדהות של user.
spd	public readonly	משתנה מסוג shared preference המאפשר העברת נתונים אודות user בין activities.
score	public	אחסון ניקוד המשתמש.
answers	public	אחסון התשובות העדכניות שענה המשתמש.

מתודה 1 – Save():

המתודה שומרת את השדות name, mail ו-pwd לתוך spd ומחזירה ערך בוליאני המציין אם השמירה צלחה.

```
public bool Save()
{
	bool success = spd.PutStringValue(General.KEY_NAME, this.Name);
	success = success && spd.PutStringValue(General.KEY_PWD, this.Pwd);
	return success && spd.PutStringValue(General.KEY_MAIL, this.Mail);
}
```

מחלקה 2 - AnswerClass:

תפקיד: אחסון נתוני אופציית תשובה לשאלה.

שדות:

שם השדה	תחום הכרה	תפקיד ושימוש	הערות
name	private	אחסון שם התשובה.	הערך מהצורה: "A" + Id
content	private	אחסון תוכן התשובה.	

מחלקה 3 - QuestionClass:

תפקיד: אחסון נתוני שאלה.

שדות:

שם השדה	תחום הכרה	תפקיד ושימוש	הערות
name	private	אחסון שם השאלה.	הערך מהצורה: "Q" + Id
content	private	אחסון תוכן השאלה.	
answers	private	אחסון רשימה של אופציות תשובה לשאלה.	המשתנה מהטיפוס: List<AnswerClass>

מחלקה 4 – Results_Structure:

תפקיד: קריאת נתוני המענה על אחת השאלות מה-FireBase ואחסונם.

שדות:

שם השדה	תחום הכרה	תפקיד ושימוש	הערות

General_Matrix	public	אחסון נתוני המענה הכלליים.	הנתונים שמורים באופן הבא:
			{
			"answer_1": #,
			"answer_2": #,
			…
			}

Ori_Matrix	public	אחסון נתוני השאלה בהתפלגות על פי עמדה פוליטית.	הנתונים שמורים באופן הבא:
			{
			"answer_1":
			{
			"option_0": #,
			"option_1": #,
			…
			"option_6": #
			},
			"answer_2":
			{
			"option_0": #,
			"option_1": #,
			…
			"option_6": #
			},
			…
			}

Rel_Matrix	public	אחסון נתוני השאלה בהתפלגות על פי אמונה דתית.	הנתונים שמורים באופן הבא:
			{
			"answer_1":
			{
			"option_0":#,
			"option_1":#,
			…
			"option_5":#
			},
			"answer_2":
			{
			"option_0":#,
			"option_1":#,
			…
			"option_5":#
			},
			…
			}

מחלקה 5 – FB_Data:

תפקיד: ניהול ה-FireBase.

שדות:

שם השדה	תחום הכרה	תפקיד ושימוש
app	private readonly	מאפשר חיבור של האפליקציה ל-FireBase.
auth	private readonly	מאפשר חיבור של האפליקציה ל-FireBase.
firebase_client	private readonly	מאפשר גישה לבסיס הנתונים השמור ב-FireBase Realtime.

מתודה 1 – CreateUser():

המתודה יוצרת משתמש חדש ב-FireBase Authentication.

```
public Android.Gms.Tasks.Task CreateUser(string email, string password)
{
    return auth.CreateUserWithEmailAndPassword(email, password);
}
```

מתודה 2 – SignIn():

המתודה מחברת את המשתמש אל ה-user שלו.

```
public Android.Gms.Tasks.Task SignIn(string email, string password)
{
    return auth.SignInWithEmailAndPassword(email, password);
}
```

מתודה 3 – RetrieveUser():

המתודה שולפת את הנתונים השמורים על המשתמש ב-FireBase Realtime.

```
public async System.Threading.Tasks.Task<User> RetrieveUser(string userId)
{
    var tmp = firebase_client.Child("users/" + userId).OnceSingleAsync<User>();
    return await tmp;
}
```

מתודה 4 – RetrieveResults():

המתודה שולפת את נתוני המענה השמורים על השאלה שמספרה questionId ב-FireBase Realtime.

```
public async System.Threading.Tasks.Task<Results_Structure> RetrieveResults(int
questionId)
{
    Results_Structure res = new Results_Structure();
    res.General_Matrix = await firebase_client.Child("Results/Q" + questionId +
"/general").OnceSingleAsync<Dictionary<string, int>>();
    res.Ori_Matrix = await firebase_client.Child("Results/Q" + questionId +
"/orientation").OnceSingleAsync<Dictionary<string, Dictionary<string, int>>>();
    res.Rel_Matrix = await firebase_client.Child("Results/Q" + questionId +
"/religion").OnceSingleAsync<Dictionary<string, Dictionary<string, int>>>();
    return res;
}
```

מתודה 5 – RetrieveQuestions():

המתודה שולפת את נתוני השאלות והתשובות האפשריות השמורים ב-FireBase Realtime.

```
public async System.Threading.Tasks.Task<List<QuestionClass>> RetrieveQuestions()
{
    List<QuestionClass> questions = new List<QuestionClass>();
    var tmp = await
firebase_client.Child("Questions").OnceSingleAsync<Dictionary<string,
Dictionary<string, string>>>();
    foreach(var q in tmp.Keys)
    {
        QuestionClass question = new QuestionClass();
        var question_info = tmp[q];

        for (int i = 0; i < question_info.Count-1; i++)
        {
            AnswerClass answer = new AnswerClass();
            answer.Name = "answer_" + (i + 1);
            answer.Content = question_info["answer_" + (i + 1)];
            question.Answers.Add(answer);
        }
        question.Content = question_info["question"];
        question.Name = q;

        questions.Add(question);
    }

    return questions;
}
```

מתודה 6 – RetrieveQNUM():

המתודה שולפת את מספר השאלות השמורות ב-FireBase Realtime.

```
public async System.Threading.Tasks.Task<int> RetrieveQNUM()
{
    return await firebase_client.Child("Q_NUM").OnceSingleAsync<int>();
}
```

מתודה 7 – GetMyOptions():

המתודה מאתחלת נתונים המאפשרים גישה ל-FireBase.

```
private Firebase.FirebaseOptions GetMyOptions()
{
    return new Firebase.FirebaseOptions.Builder()
        .SetProjectId("the-state-of-the-state")
        .SetApplicationId("the-state-of-the-state")
        .SetApiKey("AIzaSyDQFpQpc70lnIzRo3DGnfp9cb91xwUqroI")
        .SetStorageBucket("the-state-of-the-state.appspot.com")
        .Build();
}
```

מחלקה 6 –General:

תפקיד: אחסון קבועים.

שדות:

שם השדה	תחום הכרה	תפקיד ושימוש
SP_FILE_NAME	public const	קבוע המכיל את שם קובץ ה-shared preference.
KEY_NAME	public const	קבוע המכיל את המחרוזת "Name".
KEY_MAIL	public const	קבוע המכיל את המחרוזת "mail".
KEY_PWD	public const	קבוע המכיל את המחרוזת "pwd".
KEY_AGE	public const	קבוע המכיל את המחרוזת "age".
KEY_CITY	public const	קבוע המכיל את המחרוזת "city".
KEY_REL	public const	קבוע המכיל את המחרוזת "religion".
KEY_ORI	public const	קבוע המכיל את המחרוזת "orientation".
KEY_GEN	public const	קבוע המכיל את המחרוזת "general".
REQUEST_REGISTER	public const	קבוע המכיל את המספר 1.

מחלקה 7 – SP_Data:

תפקיד: אחסון קבועים.

שדות:

שם השדה	תחום הכרה	תפקיד ושימוש
sp	private readonly	משתנה של shared preference.

מתודה 1 – GetStringValue():

המתודה שולפת את הערך ששמור ב-shared preference תחת המפתח key.

```
public string GetStringValue(string key)
{
    return sp.GetString(key, string.Empty);
}
```

מתודה 2 – PutStringValue():

המתודה מאחסנת את הערך שמוכל ב-value תחת המפתח key ב-shared preference.

```
public bool PutStringValue(string key, string value)
{
    ISharedPreferencesEditor editor = sp.Edit();
    editor.PutString(key, value);
    return editor.Commit();
}
```

בסיסי נתונים:

בסיס נתונים 1 (FireBase Realtime) – "users":

User-id	User Content	
	Field Name	Field Content
user_id_#1	age	
	city	
	mail	
	name	
	orientation	
	password	
	religion	
	score	
	answers	**Question** / **Chosen Answer** Q1 /
		…
…		

פעולות:

- שליפת משתמש ונתוניו באמצעות id.
- עדכון נתוניו של משתמש קיים.
- יצירת משתמש חדש.

בסיס נתונים 2 (FireBase Realtime) – "Questions":

Question Id	Question Content	
	Field Name	Field Content
Q1	answer_1	
	…	
	question	
…		

פעולות:

- שליפת כל השאלות הקיימות ונתוניהן.

בסיס נתונים 3 (FireBase Realtime) – "Results":

<table>
<tr><th rowspan="2">Question Id</th><th colspan="2">Question Content</th></tr>
<tr><th>Field Name</th><th>Field Content</th></tr>
<tr><td rowspan="3">Q1</td><td>general</td><td><table><tr><th>Answer</th><th>Voters</th></tr><tr><td>answer_1</td><td></td></tr><tr><td colspan="2">…</td></tr></table></td></tr>
<tr><td>orientation</td><td><table><tr><th>Answer</th><th>Voters</th></tr><tr><td rowspan="4">answer_1</td><td><table><tr><th>Orientation</th><th>#</th></tr><tr><td>option_0</td><td></td></tr><tr><td colspan="2">…</td></tr><tr><td>Option_6</td><td></td></tr></table></td></tr><tr><td colspan="2">…</td></tr></table></td></tr>
<tr><td>religion</td><td><table><tr><th>Answer</th><th>Voters</th></tr><tr><td rowspan="4">answer_1</td><td><table><tr><th>Orientation</th><th>#</th></tr><tr><td>option_0</td><td></td></tr><tr><td colspan="2">…</td></tr><tr><td>Option_5</td><td></td></tr></table></td></tr><tr><td colspan="2">…</td></tr></table></td></tr>
<tr><td colspan="3">…</td></tr>
</table>

פעולות:

- שליפת נתוני המענה של שאלה באמצעות ה-id שלה.
- עדכון נתוני המענה של שאלה כאשר משתמש עונה על השאלות.

בסיס נתונים 4 – FireBase Authentication:

Identifier	Providers	Created	Signed In	User UID
user_#1_mail	email			
…				

פעולות:

- שמירת משתמש חדש.
- אימות הזדהות עבור משתמש קיים לצורך התחברות.
- בדיקה האם משתמש כלשהו כבר קיים במערכת.

מדריך למשתמש:

גרסאות שעליהן נבדק הפרויקט:

הפרויקט נבדק והורץ על טלפון מדגם Galaxy S21+ 5G בעל מערכת הפעלה של Android 11.

אופן הפעולה של הפרויקט:

התחברות:

האפליקציה מאפשרת למשתמש לפתוח חשבון חדש באמצעות המייל שלו ומשייכת למייל מספר נתונים – סיסמת המשתמש, גיל המשתמש, שם המשתמש, עיר מגוריו של המשתמש, עמדתו הפוליטית של המשתמש ואמונתו הדתית של המשתמש.

מרגע פתיחת החשבון יכול המשתמש להתחבר לחשבונו באמצעות פרטיו המזהים (מייל וסיסמה) ולגשת לאפליקציה.

הצבעה בסקרים:

לאחר ההתחברות לאפליקציה, מופיעות בפני המשתמש מספר שאלות שמורות הנטענות מתוך מאגר. המשתמש יכול לענות על השאלות הללו בהתאם לעמדותיו ותפיסת עולמו ולאחר שבחר בתשובות שאיתן הוא מסכים, הוא ילחץ על כפתור ה-submit וישלח את התשובות אל מאגר התשובות.

יש לשים לב שהטלפון ישמיע קול רטט קצר בעבור כל שאלה שהתשובה עליה השתנתה (גם בעבור שאלה שנענתה בפעם הראשונה).

צפייה בתוצאות:

ניתן לעבור ממסך הסקרים למסך התוצאות באמצעות ה-menu שבפינה השמאלית העליונה של המסך.

במסך התוצאות ניתן לראות גרפים המציגים את התפלגות התוצאות לסקרים השונים – ניתן להציג את ההתפלגות על פי עמדה פוליטית או על פי אמונה דתית (השינוי נעשה באמצעות הכפתורים שמופיעים בראש המסך).

הודעות למשתמש:

במסך ההתחברות:

- ההודעה "Error" – מעידה על כך שהתרחשה שגיאה בעת שמירת המשתמש.
- ההודעה "Enter Password" – מעידה על כך שהמשתמש לא הזין סיסמה במקום הדרוש.
- ההודעה "Login Successful" – מעידה על כך שההתחברות למשתמש קיים צלחה.
- ההודעה "User saved successfully" – מעידה על כך שפתיחת המשתמש החדש צלחה.
- הודעה נוספת יכולה להופיע בעקבות שגיאת תקשורת בין המכשיר הסלולרי לבין שרת ה-FireBase בעת לחיצה על כפתור ההתחברות.

במסך ההרשמה:

- ההודעה "Enter all values" – מעידה על כך שהמשתמש לא מילא את כל השדות המופיעים במסך.
- הודעה נוספת יכולה להופיע בעקבות שגיאת תקשורת בין המכשיר הסלולרי לבין שרת ה-FireBase בעת לחיצה על כפתור ההרשמה.

רפלקציה:

בתחילת השנה התחלתי לעבוד על אפליקציה שונה בתכלית – אפליקציית מסרים מיידיים לתלמידים וסטודנטים שתהיה מיועדת לעבודה על מטלות קבוצתיות.

אולם, כפי שציינתי בפרק המבוא, את המעבר לאפליקציה זו עשיתי בשלב מאוחר מאוד של העבודה ובידיעה כי הדבר יעורר קושי נוסף מתוך תחושה של שליחות – ראיתי את הכאוס הפוליטי השורר במדינה ואת חוסר האונים של האזרחים שלא מצליחים להבין הם דעתם בדעת מיעוט או בדעת רוב שכן כל מקור מספק תמונה שונה של המציאות.

על כן, האתגר הגדול ביותר מבחינתי בעבודה על הפרויקט היה חוסר הזמן לעבוד עליו, כמו גם הצורך לשלב את העבודה עליו ביחד עם לימודיי האקדמיים.

האתגרים הטכניים העיקריים שאיתם התמודדתי היו קשורים לשימוש ב-FireBase כמאגר נתונים לאחסון נתוני משתמשים, שאלות לסקרים ונתוני מענה על השאלות. למרות הקושי, אני מרגיש שהעבודה עם FireBase לימדה אותי רבות על ניהול מאגרי נתונים ועבודה עם שרתים.

הכלי בו נעזרתי על מנת לפתור את הבעיות שבהן נתקלתי היה ChatGPT – כלי הבינה המלאכותית של חברת OpenAI – שהצליח להנחות אותי כיצד להתמודד עם שגיאות ובאגים בלתי מובנים.

בעבודתי, למדתי כיצד לייצר גרפים ב-#C ולהציגם באפליקציה לטלפון הנייד. תכונה זו לא נכללה בתוכנית הלימודים ולכן הייתי צריך ללמוד אותה באופן עצמאי לגמרי אך אני מרגיש שזו התכונה שהכי אהבתי בפרויקט שכן היא משקפת את אהבתי לתחום ה-data science.

במהלך העבודה על הפרויקט, למדתי בקורס "מבוא לתכנות מערכות" בטכניון על יתרונותיו של התכנות המודולארי ועל כן ניסיתי לממש את הפרויקט באופן מודולארי ככל הניתן. אף על פי כן, אני מרגיש שישנם מספר מסכים שהייתי יכול לממש באופן מיטבי יותר אילו הייתי מתכנן את הפרויקט באופן מודולארי מלכתחילה ולא מנסה להפוך אותו לכזה במהלך הפיתוח.

אילו היו בידי משאבים נוספים, הייתי שוקל לשלב את הפרויקט עם ChatGPT על מנת להציג בנוסף לגרפים גם מסקנות מילוליות הנובעות מאופן מענה המשתמשים. כמו כן, הייתי משתמש במאגר נתונים גדול יותר על מנת לאחסן "snapshots" של דפוסי המענה בימים שונים ובעזרתם הייתי יכול להציג את השינוי המתמשך בדעת הקהל לאורך הזמן.

נספחים:

תיעוד מחלקות:

מחלקה User – 1:

```
// Represents a user with various properties such as name, email, password, city,
age, religion, orientation, score, and answers.
internal class User
{
    private string name, mail, pwd, city;
    private int age;
    private General.ReligionTypes religion;
    private General.OrientationTypes orientation;
    private bool exist;
    private readonly SP_data spd;

    int score;
    Dictionary<string, string> answers;

    // Initializes a new instance of the User class with a context object.
    public User(Context ctx)
    {
        spd = new SP_data(ctx);
        this.Name = spd.GetStringValue(General.KEY_NAME);
        this.Exist = this.name != String.Empty;

        if (this.exist)
        {
            this.Mail = spd.GetStringValue(General.KEY_MAIL);
            this.Pwd = spd.GetStringValue(General.KEY_PWD);
        }
    }

    // Initializes a new instance of the User class with the specified
properties.
    public User(string name, string mail, string pwd, string city, int age,
General.ReligionTypes religion, General.OrientationTypes orientation, bool exist)
    {
        this.name = name.Trim();
        this.mail = mail.Trim();
        this.pwd = pwd.Trim();
        this.city = city.Trim();
        this.age = age;
        this.religion = religion;
        this.orientation = orientation;
        this.exist = exist;

        score = 0;
        answers = new Dictionary<string, string>();
    }

    // Initializes a new instance of the User class with the properties of
another User object.
    public User(User user)
    {
        this.name = user.Name;
        this.mail = user.Mail;
        this.pwd = user.Pwd;
        this.city = user.City;
        this.age = user.Age;
        this.religion = user.Religion;
        this.orientation = user.Orientation;
        this.exist = user.Exist;
```

```
        this.score = user.Score;
        this.answers = user.answers;
    }

    // Initializes a new instance of the User class with default values.
    public User()
    {
        name = "";
        mail = "";
        pwd = "";
        city = "";
        age = 0;
        religion = 0;
        orientation = 0;
        exist = false;
        spd = null;
        score = 0;
        answers = new Dictionary<string, string>();
    }

    public string Name { get => name; set => name = value; }
    public string Mail { get => mail; set => mail = value; }
    public string Pwd { get => pwd; set => pwd = value; }
    public string City { get => city; set => city = value; }
    public int Age { get => age; set => age = value; }
    public General.ReligionTypes Religion { get => religion; set => religion =
value; }
    public General.OrientationTypes Orientation { get => orientation; set =>
orientation = value; }
    public int Score { get => score; set => score = value; }
    public Dictionary<string, string> Answers { get => answers; set => answers =
value; }

    public bool Exist { get => exist; set => exist = value; }

    // Saves the user information to the 'spd' object (shared preference).
    // Returns whether the saving operation was successful.
    public bool Save()
    {
        bool success = spd.PutStringValue(General.KEY_NAME, this.Name);
        success = success && spd.PutStringValue(General.KEY_PWD, this.Pwd);
        return success && spd.PutStringValue(General.KEY_MAIL, this.Mail);
    }
}
```

מחלקה 2 – AnswerClass:

```
// Represents an answer in the system.
public class AnswerClass
{
    private string name;
    private string content;

    // Initializes a new instance of the AnswerClass class with the provided name
and content.
    public AnswerClass(string name, string content)
    {
        this.name = name;
        this.content = content;
    }

    // Initializes a new instance of the AnswerClass class with the provided
answerId and content.
    public AnswerClass(int answerId, string content)
    {
        this.name = "A" + answerId;
        this.content = content;
    }

    // Initializes a new instance of the AnswerClass class with default values
for name and content.
    public AnswerClass()
    {
        name = "";
        content = "";
    }

    public string Name { get => name; set => name = value; }
    public string Content { get => content; set => content = value; }
}
```

מחלקה 3 – QuestionClass:

```
// Represents a question in the system.
public class QuestionClass
{
    private string content, name;
    private List<AnswerClass> answers;
    /*
        * 1: "answer_1":"..."
        * 2: "answer_2":"..."
        * ...
        */

    // Initializes a new instance of the QuestionClass class with the provided
list of answers, name, and content.
    private QuestionClass(List<AnswerClass> answers, string name, string content)
    {
        this.answers = answers;
        this.name = name;
        this.content = content;
    }

    // Initializes a new instance of the QuestionClass class with the provided
list of answers, questionId, and content.
    private QuestionClass(List<AnswerClass> answers, int questionId, string
content)
    {
        this.answers = answers;
        this.name = "Q" + questionId;
        this.content = content;
    }

    // Initializes a new instance of the QuestionClass class with default values
for answers, name, and content.
    public QuestionClass()
    {
        answers = new List<AnswerClass>();
        name = "Q";
        content = "Empty";
    }

    public List<AnswerClass> Answers { get => answers; set => answers = value; }
    public string Name { get => name; set => name = value; }
    public string Content { get => content; set => content = value; }

}
```

מחלקה 4 – Results_Structure:

```
// Represents the results structure for storing matrix data.
public class Results_Structure
{
    public Dictionary<string, int> General_Matrix { get; set; }

    public Dictionary<string, Dictionary<string, int>> Ori_Matrix { get; set; }

    public Dictionary<string, Dictionary<string, int>> Rel_Matrix { get; set; }

    // Initializes a new instance of the Results_Structure class with default
values for the general matrix, orientation matrix, and religion matrix.
    public Results_Structure()
    {
        General_Matrix = new Dictionary<string, int>();
        Ori_Matrix = new Dictionary<string, Dictionary<string, int>>();
        Rel_Matrix = new Dictionary<string, Dictionary<string, int>>();
    }
}
```

מחלקה 5 – FB_Data:

```
// Handles the usage of FireBase.
internal class FB_Data
{
    private readonly FirebaseApp app;
    private readonly FirebaseAuth auth;
    private readonly FirebaseClient firebase_client = new
FirebaseClient("https://the-state-of-the-state-default-rtdb.firebaseio.com");

    // Constructor for FB_Data class: Initializes FirebaseApp and FirebaseAuth
instances.
    public FB_Data()
    {
        app = FirebaseApp.InitializeApp(Application.Context);

        if (app is null)
        {
            Firebase.FirebaseOptions options = GetMyOptions();
            app = FirebaseApp.InitializeApp(Application.Context, options);
        }

        auth = FirebaseAuth.Instance;
    }

    // Retrieves FireBase options for initializing FirebaseApp instance.
    private Firebase.FirebaseOptions GetMyOptions()
    {
        return new Firebase.FirebaseOptions.Builder()
            .SetProjectId("the-state-of-the-state")
            .SetApplicationId("the-state-of-the-state")
            .SetApiKey("AIzaSyDQFpQpc70lnIzRo3DGnfp9cb91xwUqroI")
            .SetStorageBucket("the-state-of-the-state.appspot.com")
            .Build();
    }

    // Creates a new user with the given email and password.
    public Android.Gms.Tasks.Task CreateUser(string email, string password)
    {
        return auth.CreateUserWithEmailAndPassword(email, password);
    }

    // Signs in a user with the given email and password.
    public Android.Gms.Tasks.Task SignIn(string email, string password)
    {
        return auth.SignInWithEmailAndPassword(email, password);
    }

    // Retrieves user information for the given userId.
    public async System.Threading.Tasks.Task<User> RetrieveUser(string userId)
    {
        var tmp = firebase_client.Child("users/" +
userId).OnceSingleAsync<User>();
        return await tmp;
    }

    // Retrieves results for the given questionId.
    public async System.Threading.Tasks.Task<Results_Structure>
RetrieveResults(int questionId)
    {
        Results_Structure res = new Results_Structure();
        res.General_Matrix = await firebase_client.Child("Results/Q" + questionId
+ "/general").OnceSingleAsync<Dictionary<string, int>>();
```

```
        res.Ori_Matrix = await firebase_client.Child("Results/Q" + questionId +
"/orientation").OnceSingleAsync<Dictionary<string, Dictionary<string, int>>>();
        res.Rel_Matrix = await firebase_client.Child("Results/Q" + questionId +
"/religion").OnceSingleAsync<Dictionary<string, Dictionary<string, int>>>();
        return res;
    }

    // Retrieves all questions from the FireBase Realtime database.
    public async System.Threading.Tasks.Task<List<QuestionClass>>
RetrieveQuestions()
    {
        List<QuestionClass> questions = new List<QuestionClass>();
        var tmp = await
firebase_client.Child("Questions").OnceSingleAsync<Dictionary<string,
Dictionary<string, string>>>();
        foreach(var q in tmp.Keys)
        {
            QuestionClass question = new QuestionClass();
            var question_info = tmp[q];

            for (int i = 0; i < question_info.Count-1; i++)
            {
                AnswerClass answer = new AnswerClass();
                answer.Name = "answer_" + (i + 1);
                answer.Content = question_info["answer_" + (i + 1)];
                question.Answers.Add(answer);
            }
            question.Content = question_info["question"];
            question.Name = q;

            questions.Add(question);
        }

        return questions;
    }

    // Retrieves the number of questions from the FireBase Realtime database.
    public async System.Threading.Tasks.Task<int> RetrieveQNUM()
    {
        return await firebase_client.Child("Q_NUM").OnceSingleAsync<int>();
    }
}
```

מחלקה 6 – General:

```
public static class General
{
    public const string SP_FILE_NAME = "data.sp";
    public const string KEY_NAME = "Name";
    public const string KEY_MAIL = "mail";
    public const string KEY_PWD = "pwd";
    public const string KEY_AGE = "age";
    public const string KEY_CITY = "city";
    public const string KEY_REL = "religion";
    public const string KEY_ORI = "orientation";
    public const string KEY_GEN = "general";
    public const int REQUEST_REGISTER = 1;

    public enum ReligionTypes { Secular_Jew, Traditional_Jew, Religious_Jew,
Orthodox_Jew, Secular_Arab, Religious_Arab, Length };
    public enum OrientationTypes { Extreme_Left, Left, Center_Left, Center,
Center_Right, Right, Extreme_Right, Length };

}
```

מחלקה 7 – SP_Data:

```
// This class provides an abstraction for working with shared preferences in
Android using the ISharedPreferences interface.
internal class SP_data
{
    private readonly ISharedPreferences sp;

    // Initializes a new instance of the SP_data class with the specified
context.
    public SP_data(Context ctx)
    {
        sp = ctx.GetSharedPreferences(General.SP_FILE_NAME,
FileCreationMode.Private);
    }

    // Retrieves a string value from shared preferences with the specified key.
    public string GetStringValue(string key)
    {
        return sp.GetString(key, string.Empty);
    }

    // Puts a string value into shared preferences with the specified key.
    public bool PutStringValue(string key, string value)
    {
        ISharedPreferencesEditor editor = sp.Edit();
        editor.PutString(key, value);
        return editor.Commit();
    }
}
```