



ברוכים הבאים ל Jobit

תהיתם היכן תוכלו למצא עובד בזמן קצר?

האם אתם נמצאים במלא קבוצות Whatsapp של עבודות מזדמנות?

מתי בפעם האחרונה ייחלתם לעבודה מיידית ולא מצאתם את הפלטפורמה המתאימה להשיג אותה?

בשביל זה אנחנו כאן!

Jobit נותנת פתרון מידי לנושא.

על ידי שימוש מהיר ופשוט באפליקציה תוכלו להכנס לעולם מלא תוכן בעבודות.

במידה ואתה מעביד – תוכל לפרסם מודעה של סוג העבודה, פרטי התקשרות, מיקום בארץ, למחוק מודעות וכדומה..

במידה ואתה עובד – תוכל לראות את כלל המודעות ועל ידי חיפוש דינאמי למצא את המשרה הרצויה ולהגיש בקלות הפניה להעסקה. בנוסף, תוכל לראות את כלל המשרות ששלחת אליהן הפניות.

הוספנו במיוחד בשבילכם פונקציונאליות נוספת של צ'אט שבה תוכלו לשוחח ביניכם ולתאם ציפיות. עם זאת כל אחד מכם יוכל להעלות תמונה לפרופיל דרך המצלמה או הגלריה בפלאפון.

מהעמוד הבא אנחנו נרחיב על אופן יישום האפליקציה.

שלכם,

Jobit

Login page

שדות :

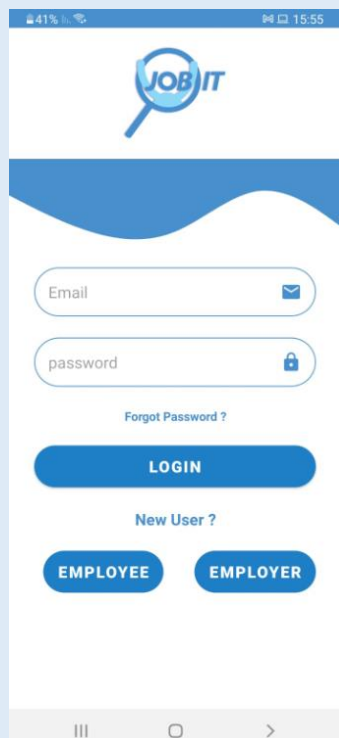
```
6
7 public class logInPage extends AppCompatActivity {
8
9     public static final String TAG = "TAG";
10    private EditText eMail, pass;
11    private FirebaseAuth fAuth;
12    private FirebaseFirestore fStore;
13    private Button registerEmployerBtn , registerEmployeeBtn;
14    private Button logInButton;
15
```

EditText: שדה שאתה יכול להכניס אליו מידע (קלט).

firebaseAuth : שדה שבו שומרים את כל המשתנים בענן-אוסף .. collection

FirebaseFirestore : שומרים את כל המידע לגבי המשתמשים document

Button : כפתור רגיל כמובן יש לו הרבה מאפיינים כמן מספר סידורי ID לכל כפתור שמזהה אותו בתוך "הקוד".



יש לנו שני כפתורים להכנסת שם משתמש וסיסמה של המשתמש כדי להתחבר, מתחברים ל firebase ומאתחלים את המשתנים שהגדרנו. בודקים האם הם באמת שדות שנמצאים

ב firebase או לא. כדי להתחבר, ישנם תנאים להקלדת סיסמה, כלומר, למשתמש אסור לכתוב סיסמא הפוחתת מ-6 תווים, אם המשתמש לא כתב את מה שנדרש הוא יקבל הודעת שגיאה במקום.

```
String getEmail = eMail.getText().toString();
String getPass = pass.getText().toString().trim();

//checks if email and pass correct
if (getEmail.isEmpty()) {
    eMail.setError("Email is empty!");
    eMail.requestFocus();
    return;
}
if (!Patterns.EMAIL_ADDRESS.matcher(getEmail).matches()) {
    eMail.setError("enter a valid email!");
    eMail.requestFocus();
    return;
}
if (getPass.isEmpty() || getPass.length() < 6) {
    pass.setError("minimum 6 characters! ");
    pass.requestFocus();
    return;
}
```

לאחר מכן יצרנו בדאטא-בייס שני אוספים אחד של העובדים ואחד של המעבידים ואז לפי זה אנו מחפשים את כל הנתונים הנדרשים כדי להתחבר למשתמש הנכון.

```
fAuth.signInWithEmailAndPassword(getEmail, getPass)
    .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
        @Override
        public void onSuccess(AuthResult authResult) {
            FirebaseAuth fUser = fAuth.getCurrentUser();
            if (fUser.isEmailVerified()) {
                //check if the user is employee on database
                CollectionReference col = fStore.collection("Employees");
                DocumentReference docIdRef = col.document(fUser.getId());
                docIdRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        if (task.isSuccessful()) {
                            DocumentSnapshot document = task.getResult();
                            if (document.exists()) {
                                Toast.makeText(context, "Welcome User", Toast.LENGTH_SHORT).show();
                                openEmployeePage();
                            }
                        } else {
                            Log.d(TAG, "Failed with: ", task.getException());
                        }
                    }
                });
            }
        }
    });
```

התחברות המעביד והעובד זהה לחלוטין אבל ההרשמה שלהם לא אותו דבר (נפרט בהמשך). בודקים אם נעשה אימות למייל שנשלח אחרי ההרשמה לאישור המשתמש

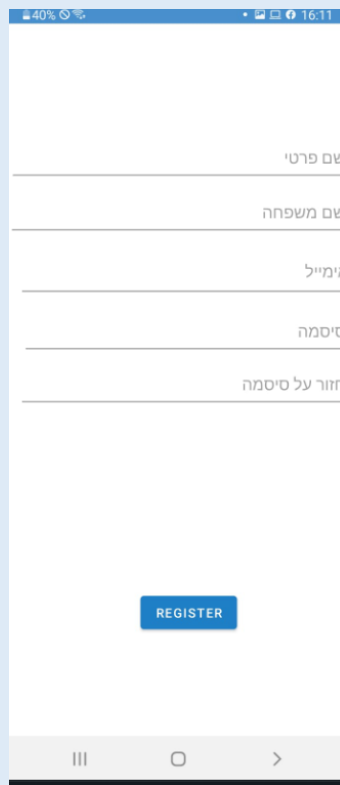
אז אם אושר נבדוק מאיזה סוג משתמש והמשתמש אכן יראה את הדף המתאים לו, בין אם הוא עובד או מעביד (שני דפים שונים).

```
44 }
45 public void openEmployeeRegisterPage(){
46     Intent intent = new Intent( packageContext: this , employeeRegister.class);
47     startActivity(intent);
48 }
49
50 public void openEmployerRegisterPage(){
51     Intent intent = new Intent( packageContext: this , employerRegister.class);
52     startActivity(intent);
53 }
54
55
56 public void openEmployeePage() {
57     Intent intent = new Intent( packageContext: this, employeePage.class);
58     startActivity(intent);
59 }
60
61 public void openEmployerPage() {
62     Intent intent = new Intent( packageContext: this, employerPage.class);
63     startActivity(intent);
64 }
65 }
```

כפתור הרשמת עובד/מעביד ישלח אותי לדף ההרשמה.

אם הוא זוהה כעובד עובר לדף של העובד, ואם הוא זוהה כמעביד אז ישר לדף של המעביד.

המסך של employerRegister (הרשמת משתמש בתור מעביד):



The screenshot shows a mobile application interface for employer registration. It features five text input fields with labels in Hebrew: 'שם פרטי' (First Name), 'שם משפחה' (Last Name), 'אימ"ל' (Email), 'סיסמה' (Password), and 'חזור על סיסמה' (Repeat Password). Below the fields is a blue button labeled 'REGISTER'. The status bar at the top shows 40% battery and the time 16:11. The bottom navigation bar has three icons: a list, a home icon, and a right arrow.

```
public static final String TAG = "TAG";
private Button rButton;
private EditText fName, lName, eMail, pass, cPass;
private String userID;

private FirebaseAuth fAuth;
private FirebaseFirestore fStore;
```

בהתחלה הגדרנו את כל השדות שצריך בהמשך ואז הגדרנו את

FirebaseAuth : בגלל השימוש ב firebase Authentication בכדי לשמור את כל המשתמשים באפליקציה.

FirebaseFirestore : בגלל השימוש ב firebase Firestore בכדי לשמור את הפרטים שקשורים למשתמשים באפליקציה (חילוק המשתמשים לשתי קבוצות עובדים, מעבידים וכל אחד יש לו את השדות שלו..)

```

fName = findViewById(R.id.firstName);
lName = findViewById(R.id.lastName);
eMail = findViewById(R.id.editTextTextEmailAddress);
pass = findViewById(R.id.password);
cPass = findViewById(R.id.confirmPassword);

fAuth = FirebaseAuth.getInstance();
fStore = FirebaseFirestore.getInstance();
rButton = (Button) findViewById(R.id.employerRegisterButton);

rButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String emailText = eMail.getText().toString().trim();
        String fNameText = fName.getText().toString();
        String lNameText = lName.getText().toString();
        String passwordText = pass.getText().toString().trim();
        String conPassText = cPass.getText().toString().trim();
    }
});

```

ב onCreate נתנו לכל שדה את ה id המתאים (בהתאם לקובץ xml) ואז השתמשנו בפונקציית setOnClickListener <- onClick (כשלוחצים על כפתור register הפונקציה רצה אוטומטית) :

בהתחלה שלפנו את כל המידע מהשדות שכתב המשתמש (איפה שמילא את המייל, שם והסיסמה שלו).

```

//checks if all fields correct
if (fNameText.isEmpty()) {
    fName.setError("First name empty!");
    fName.requestFocus();
    return;
}
if (lNameText.isEmpty()) {
    lName.setError("Last name empty!");
    lName.requestFocus();
    return;
}
if (emailText.isEmpty()) {
    eMail.setError("Email is empty!");
    eMail.requestFocus();
    return;
}
if (!Patterns.EMAIL_ADDRESS.matcher(emailText).matches()) {
    eMail.setError("Enter a valid email!");
    eMail.requestFocus();
    return;
}
}

```

לקחנו בחשבון את כל בדיקות השדות (בדיקות קצה) כדי שלא נגיע למצב שהאפליקציה עוצרת ..

```

if (!Patterns.EMAIL_ADDRESS.matcher(emailText).matches()) {
    eMail.setError("Enter a valid email!");
    eMail.requestFocus();
    return;
}
if (passwordText.isEmpty() || passwordText.length() < 6) {
    pass.setError("Minimum 6 characters!");
    pass.requestFocus();
    return;
}
if (conPassText.isEmpty() || (conPassText.length() != passwordText.length())) {
    cPass.setError("Confirm pass don't match!");
    cPass.requestFocus();
    return;
}
for (int i = 0; i < conPassText.length(); i++) {
    if (passwordText.charAt(i) != conPassText.charAt(i)) {
        cPass.setError("Confirm pass don't match!");
        cPass.requestFocus();
        return;
    }
}

registerEmployer(fNameText, lNameText, emailText, passwordText);

```

אחרי הבדיקה האחרונה של מילוי הסיסמה פעמיים ושהיו זהות כמובן צריך כמובן להירשם ב firebase ולשמור את המידע של המשתמש שם , לכן קוראים לפונקציית ההרשמה registerEmployer (הרשמת מעביד) .

```

private void registerEmployer(String fname, String lname, String email, String password) {
    FirebaseAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener((task) -> {
        if (task.isSuccessful()) {
            FirebaseAuth fuser = FirebaseAuth.getCurrentUser();
            //send verification email
            fuser.sendEmailVerification().addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    Toast.makeText(context, "Verification Email Has been Sent",
                        Toast.LENGTH_SHORT).show();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Log.d(TAG, "onFailure Email not sent" + e.getMessage());
                }
            });
        }
    });
}

```

הפונקציה מקבלת:

fname: שם פרטי.

lname: משפחה.

Email: המייל של המשתמש שרוצה להירשם.

Password: סיסמה.

ואז משתמשים בפונקציית createUserWithEmailAndPassword שמייצרת משתמש לפי מייל וסיסמה שקיבלנו. לאחר מכן המשתמש הנוכחי מקבל מייל לאימות ישירות לכתובת מייל שלו (כדי למנוע ממצב של מיילים שגויים והרשמות ממיילים של אחרים).


```

Toast.makeText( context: employerRegister.this, text: "User Created", Toast.LENGTH_SHORT).show();
userID = fAuth.getCurrentUser().getUid();
//find the user in database
DocumentReference documentReference = fStore.collection( collectionPath: "Employers").document(userID);
//put the user fields on database
Map<String, Object> user = new HashMap<>();
user.put("fName", fname);
user.put("lName", lname);
user.put("eMail", email);
user.put("pass", password);
documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void unused) {
        Log.d(TAG, msg: "user profile is created for" + userID);
    }
});
startActivity(new Intent(getApplicationContext(), loginPage.class));
else {
    Toast.makeText( context: employerRegister.this, text: "Error !" + task.getException().getMessage(),
        Toast.LENGTH_SHORT).show();
}

```

מייצרים משתמש חדש באוסף של קבוצת המעבידים שנמצאו ב-firebase store (מכניסים כמסמך לכן מדובר בכמה מסמכים שכוללים שדות שמתארים את המעביד ושנכללות באוסף של המעבידים) .

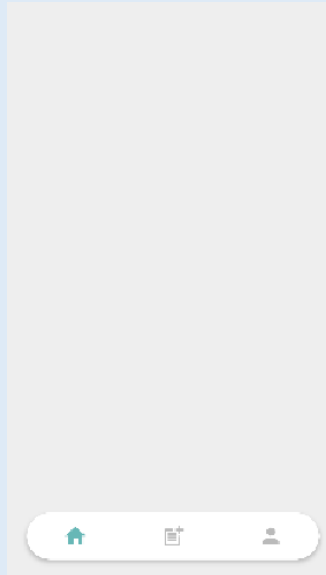
הערה : יש אוסף זהה גם לעובדים.

מייצרים hashmap ומכניסים את השדות של המסמכים שדיברנו עליהם קודם לטבלה, ולבסוף דוחפים את הטבלה למסמך ואז נוצר לנו מסמך שכולל כמה שדות שמאפיינים ומתארים את המעביד.

ובסוף עוברים למסך התחלה (מסך כניסה לאפליקציה).

EmployerPage

מסך ראשי מעביר



מסך המעביר מורכב מ- Bottom Navigation המשמש כתפריט.

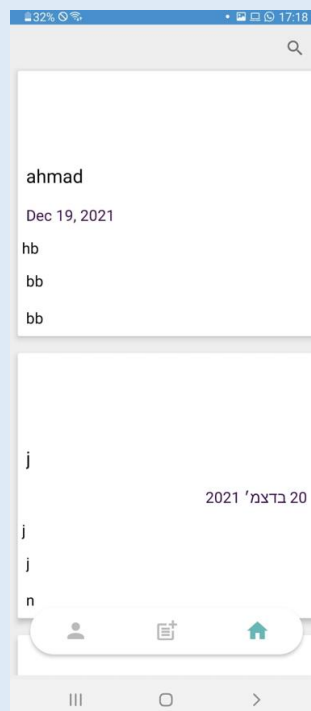
```
loginPage.java x employerPage.java x activity_employer_page.xml x
17 BottomNavigationView bottomNavigationView;
18
19
20 @Override
21 protected void onCreate(Bundle savedInstanceState) {
22     super.onCreate(savedInstanceState);
23     setContentView(R.layout.activity_employer_page);
24
25     bottomNavigationView = findViewById(R.id.bottom_navigation);
26     getSupportFragmentManager().beginTransaction().replace(R.id.main_containe
27     bottomNavigationView.setSelectedItemId(R.id.nav_home);
28
29     bottomNavigationView.setOnItemSelectedListener(new NavigationBarView.OnIt
30     @Override
31     public boolean onNavigationItemSelected(@NonNull MenuItem item) {
32         Fragment fragment=null;
33         switch (item.getItemId()){
34             case R.id.nav_home:
35                 fragment = new HomeFragment();
36                 break;
37
38             case R.id.nav_post:
39                 fragment = new PostFragment();
40                 break;
41
42             case R.id.nav_profile:
43                 fragment = new ProfileFragment();
44                 break;
45         }
46
47         getSupportFragmentManager().beginTransaction().replace(R.id.main_
48
49         return true;
```

שדה של Bottom Navigation

כל כפתור מייצג Fragment חדש (Fragment מייצג חלון הניתן לשימוש חוזר, פרגמנט מגדיר ומנהל את הפריסה שלו, יש לו מחזור חיים משלו ויכול לטפל באירועי קלט משלו. הם אינם יכולים לחיות בפני עצמם - עליהם להתארח בפעילות או קטע אחר במקרה זה מסך ראשי מעביד)

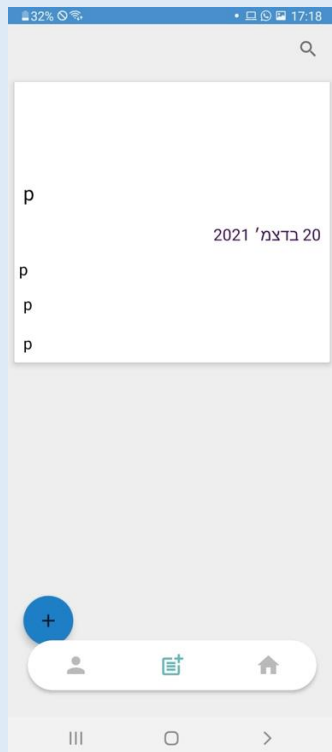
ללחיצה על המסך, ישנם 3 פרגמנטים: `onNavigationItemSelectedListener` – פונקציה המחליטה איזה פרגמנט להראות בהתאם

`homeFragment` המשמש להצגת לוח המשרות ובר לחיפוש מודעות

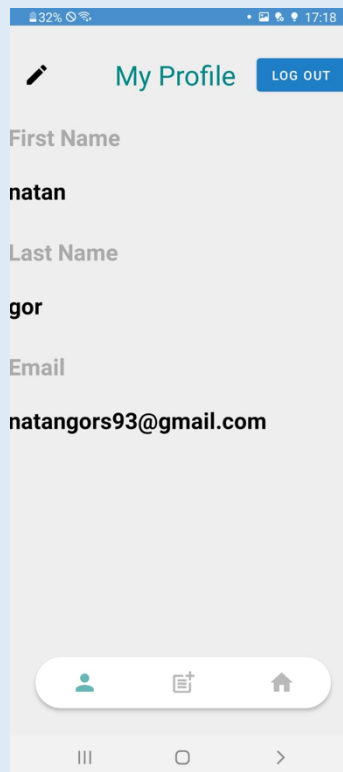


בהמשך – נדאג לעיצוב לוח המודעות.

`postFragment` המראה את המשרות שלי, בר לחיפוש במשרות שלי ואפשרות לפרסום מודעה חדשה.



profileFragment המשמש להצגת פרופיל אישי כפתור להתנתקות וכפתור לעריכת פרטים אישיים אשר יכלול הרשאת גישה ל טלפון בכדי לצלם/להעלות מהגלריה לתמונת פרופיל.



מסך PostFragment:

במחלקה זו אנחנו קוראים את נתוני המודעות של user ספציפי המופיעים ב RealtimeDatabase ומציגים אותם בתור מודעה.

בתור התחלה אנו משיגים הפנייה ל Authentication ומושכים את ה id של ה user הנוכחי. לאחר מכן אנחנו ניגשים לצומת של "PostJobInfo" <= על פי ה ID של היוזר הנוכחי.

```
//Get an instance to data
fauth = FirebaseAuth.getInstance();
FirebaseUser fUser = fauth.getCurrentUser();
assert fUser != null;
String uID = fUser.getId();

ref = FirebaseDatabase.getInstance().getReference().child("PostJobInfo").child(uID);
```

לאחר מכן אנחנו בונים את כל המערך שמכיל בתוכו את המידע שנמצא במודעות. מקבלים גישה למסך הגלילה ומצמידים לו LinearLayoutManager כדי לקבל את הרשימה שלנו בצורה אנכית. נבנה אדפטר עם הכולל את כלל המידע שבמודעות ואז ניתן למסך הגלילה את כלל הדאטה שבאדפטר.

בנוסף אנחנו מקבלים גישה לחלון החיפוש שמופיע בראש המסך.

```
dataList = new ArrayList<Data>();
recyclerView = findViewById(R.id.rv);
LinearLayoutManager layoutManager = new LinearLayoutManager(context, this);
recyclerView.setLayoutManager(layoutManager);
adapterClass = new adapterClass(context, this, dataList);
recyclerView.setAdapter(adapterClass);

searchView = findViewById(R.id.searchView1);
}
```

ברגע שהמסך מופעל אנחנו מפנים listener לצומת של ה User ID וברגע שמשתנה/נוסף דאטה מסוים אנחנו סורקים את ה Snapshot שקיבלנו מה listener והופכים אותו למערך של דאטה.

```

@Override
protected void onStart() {
    super.onStart();

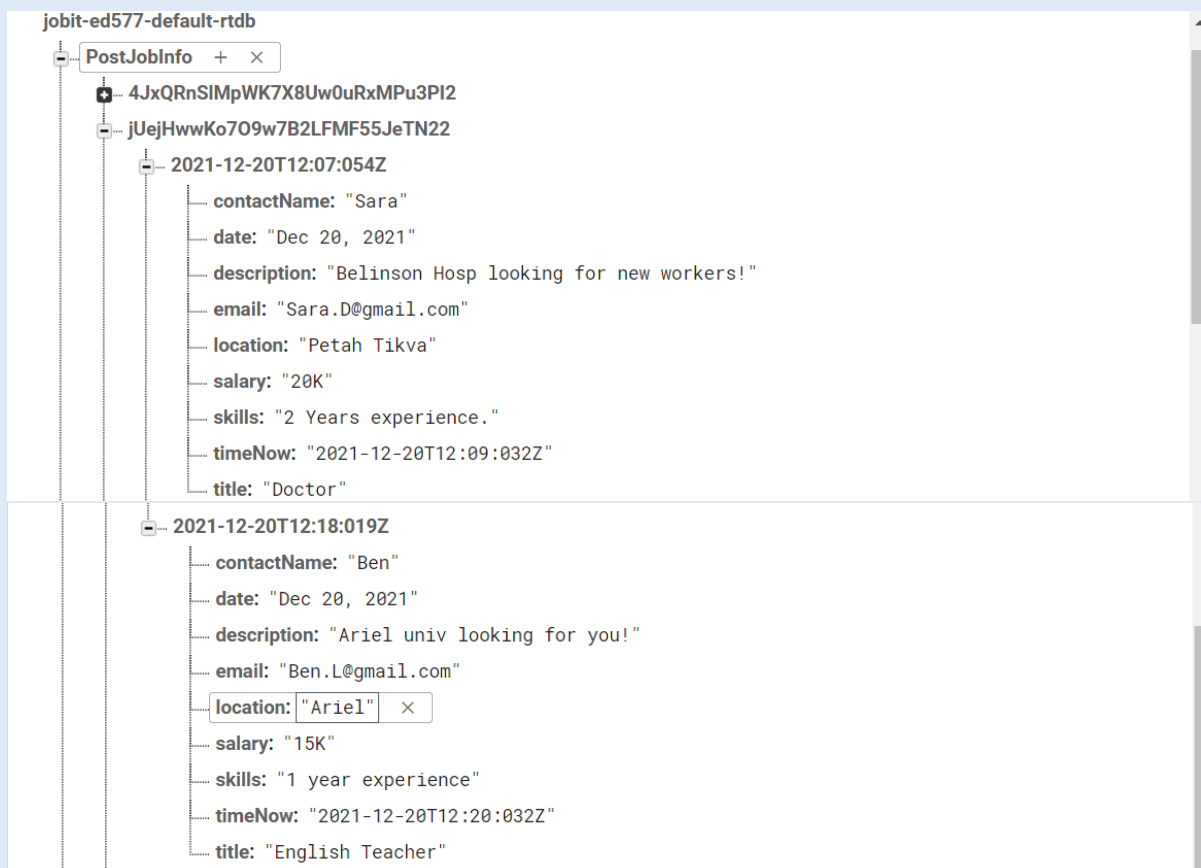
    if(ref != null){
        ref.addValueEventListener(new ValueEventListener() {

            @NonNull
            @Override
            protected Object clone() throws CloneNotSupportedException {
                return super.clone();
            }

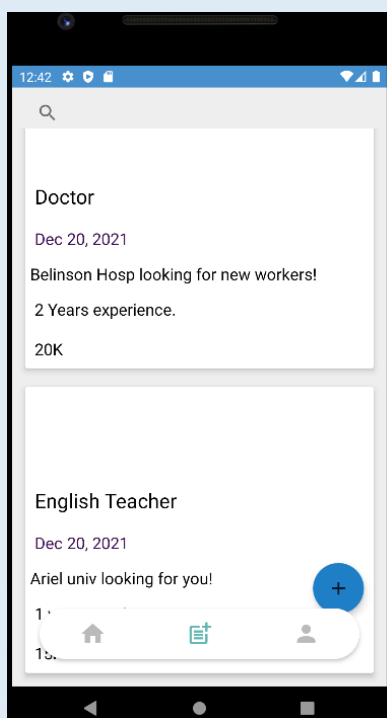
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                int counter = 0;
                if(snapshot.exists()){
                    dataList = new ArrayList<>();
                    for(DataSnapshot ds : snapshot.getChildren()){
                        for(DataSnapshot ds1 : ds.getChildren()) {
                            if (counter == 4) {
                                dataList.add(ds1.getValue(Data.class));
                            }
                            counter++;
                        }
                        counter = 0;
                    }
                    adapterClass adapterClass = new adapterClass(dataList);
                    recyclerView.setAdapter(adapterClass);
                }
            }
        });
    }
}

```

אנחנו עוברים על כל המידע שמזן לנו ב Database בצורה הבאה:



ואז אנחנו מעלים למסך הגלילה את הנתונים העדכניים.



בתחילת הצגת המסך יש לנו גם את התצוגה של הפונקציה `SearchView`:

```

if(searchView != null){
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) { return false; }

        @Override
        public boolean onQueryTextChange(String newText) {
            search(newText);
            return true;
        }
    });
}

```

במידה ומוזן טקסט אנחנו מפעילים את הפונקציה הבאה:
 בפונקציה הזו אנחנו מאפשרים הצגת חיפוש לפי שם המקצוע.

```

private void search(String str){
    ArrayList<Data> searchList = new ArrayList<>();
    if(!str.equals("")) {
        for (Data object : dataList) {
            if (object.getTitle().toLowerCase().contains(str.toLowerCase())) {
                searchList.add(object);
            }
        }
        adapterClass adapterClass = new adapterClass(searchList);
        recyclerView.setAdapter(adapterClass);
    }else{
        adapterClass adapterClass = new adapterClass(dataList);
        recyclerView.setAdapter(adapterClass);
    }
}

```

מסך הבית (HomeFragement) הוא מבוצע אותו הדבר כמו המסך לעיל רק ללא תוספת העלאת מודעה. ומיקום הצומת ההתחלתי שונה כדי להגיע לכלל המודעות ולא רק למודעות של User ספציפי.

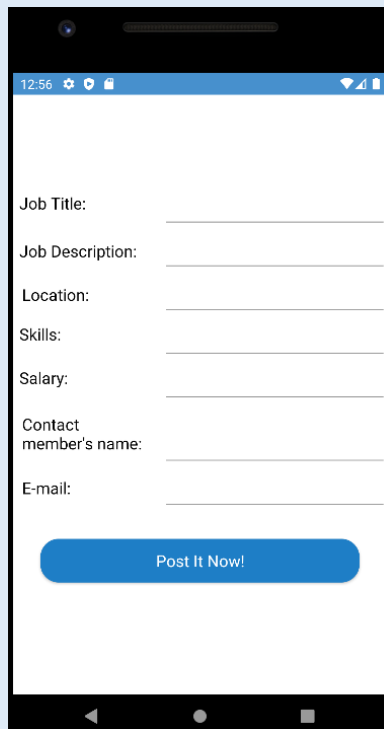
בשביל להגיע למסך של פרסום מודעה אנו לוחצים על כפתור fabBtn המשייך את המחלקה הנוכחית למחלקת InsertEmployerPost.

```

fabBtn = findViewById(R.id.post_add);
fabBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getApplicationContext(), insertEmployerPost.class));
    }
});

```


מחלקת InsertEmployerPost:



מחלקה זו מפרטת לנו על התהליך של הזנת נתוני העלאת המודעה בדאטה בייס.

```
public class insertEmployerPost extends AppCompatActivity {  
  
    private EditText job_title;  
    private EditText job_desc;  
    private EditText job_skills;  
    private EditText job_salary;  
    private EditText job_location;  
    private EditText job_email;  
    private EditText job_contact_name;  
  
    private Button button_post_job;  
  
    FirebaseDatabase firebaseDatabase;  
    FirebaseAuth mAuth;  
    DatabaseReference databaseReference;  
    Data data;
```

בתור התחלה אנחנו מהתחלים את השדות כתיבה שלנו ואת הכפתור פרסום ומשייכים אותם ל id שלהם בקובץ התצוגה xml. לאחר מכן אנחנו משיגים הפנייה לדאטה בייס כולו בצומת של "PostJobInfo" => על פי ה ID של היוזר הנוכחי => ועל פי הזמן פרסום מודעה העכשווי.

זמן פרסום המודעה מוצג על פי התבנית הבאה:

```
// Present date by specific format  
SimpleDateFormat ISO_8601_FORMAT = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:sss'Z'");  
String timeNow = ISO_8601_FORMAT.format(new Date());
```

יצרנו מחלקה חדשה המציינת את האובייקט Data שלנו שבתוכה יש את כלל הנתונים שאנו מזינים ב Firebase.

```

3      public class Data {
4
5          private String title;
6          private String description;
7          private String skills;
8          private String salary;
9
10         private String id;
11         private String date;
12         private String timeNow;
13         private String location;
14         private String contactName;
15         private String email;
16     }

```

```

71     FirebaseAuth fAuth = FirebaseAuth.getInstance();
72     FirebaseUser fUser = fAuth.getCurrentUser();
73     assert fUser != null;
74     String uID = fUser.getId();
75     // below line is used to get the
76     // instance of our Firebase database.
77     FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
78
79     // below line is used to get reference for our database.
80     DatabaseReference databaseReference = firebaseDatabase.getReference().child("PostJobInfo").child(uID).child(now);
81
82     // initializing our object
83     // class variable.
84     data = new Data();

```

התחלנו משתנה data מהסוג של הקלאס הנ"ל שדרכו אנו מבצעים את הזנת הנתונים לפיירבייס.

ברגע שנלחץ הכפתור button_post_job מתבצעים הדברים הבאים:

אנו לוקחים את הטקסט שהוזן בתיבות טקסט שבמסך ושומרים אותם בערכים שונים מסוג String.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_insert_employer_post);

    // initializing our edittext and button
    job_title = findViewById(R.id.job_title);
    job_desc = findViewById(R.id.job_desc);
    job_skills = findViewById(R.id.job_skills);
    job_salary = findViewById(R.id.job_salary);
    job_email = findViewById(R.id.job_email);
    job_location = findViewById(R.id.job_loc);
    job_contact_name = findViewById(R.id.job_content_memeber1);
    button_post_job = findViewById(R.id.job_post);
}

```

```
// adding on click listener for our button.
button_post_job.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        //Present date by specific format
        SimpleDateFormat ISO_8601_FORMAT = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss'Z'");

        String timeNow = ISO_8601_FORMAT.format(new Date());
        // getting text from our edittext fields.
        String title = job_title.getText().toString();
        String description = job_desc.getText().toString();
        String skills = job_skills.getText().toString();
        String salary = job_salary.getText().toString();
        String location = job_location.getText().toString();
        String contactName = job_contact_name.getText().toString();
        String email = job_email.getText().toString();
        String date = DateFormat.getDateInstance().format(new Date());

        // below line is for checking weather the
        // edittext fields are empty or not.
        if (TextUtils.isEmpty(title) || TextUtils.isEmpty(description) || TextUtils.isEmpty(skills) || TextUtils.isEmpty(salary)) {

            // if the text fields are empty
            // then show the below message.
            Toast.makeText( context: insertEmployerPost.this, text: "Data is missing...", Toast.LENGTH_SHORT).show();
        } else {
            // else call the method to add
            // data to our database.
            addDataToFirebase(title,description,skills,salary,date,timeNow,location,contactName,email);
            //Return to EmployerPage
            openEmployerPage();
        }
    }
});
```

לאחר מכן אנו נשלח listener לצומת הנ"ל וה listener ידאג לדברים הבאים:

- הזנת הנתונים בדאטהבייס.
- אם השדות הללו ריקים אנו מחזירים למשתמש הערה שחסר המידע המבוקש ולכן הוא לא יכול לשלוח את המודעה.
- במידה והשדות מולאו אנו מזינים במשתנה data שלנו את המידע שהמשתמש הזין.

```
private void addDataToFirebase(String title, String description, String skills, String salary, String date, String timeNow, St
//Setting data in our object class
data.setTitle(title);
data.setDescription(description);
data.setSkills(skills);
data.setSalary(salary);
data.setDate(date);
data.setTimeNow(timeNow);
data.setLocation(location);
data.setEmail(email);
data.setContactName(contactName);

//adding value event listener method which is called with database reference.
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        // inside the method of on Data change we are setting
        // our object class to our database reference.
        // data base reference will sends data to firebase.
        databaseReference.setValue(data);

        // after adding this data we are showing toast message.
        Toast.makeText( context: insertEmployerPost.this, text: "Data as been updated", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // if the data is not added or it is cancelled then
        // we are displaying a failure toast message.
        Toast.makeText( context: insertEmployerPost.this, text: "Fail to add data " + error, Toast.LENGTH_SHORT).show();
    }
});
```

במידה והמידע השתנה אנו מזינים את המידע בצומת שציינו לעיל ופעולה זו מעדכנת את הדאטה בייס במידה העדכני ביותר.

ברגע שהמידע עודכן המשתמש יקבל את ההודעה הבאה: Data as been updated. במידה והעלאת המידע לא הצליחה המשתמש יקבל את ההודעה: Fail to add data.

:AdapterClass

בתוך המחלקה שדואגת לספק לנו את הנתונים מהדאטה בייס למסך הגלילה יש מחלקה נוספת שאוחזת בסך הנתונים שאותם נרצה לשלוף.

אנחנו מאתחלים את הנתונים הללו ומשייכים אותם לתצוגה.

```
static class MyViewHolder extends RecyclerView.ViewHolder{
    TextView title, description, skills, salary, date;
    ImageView imageView;
    CardView cardView;

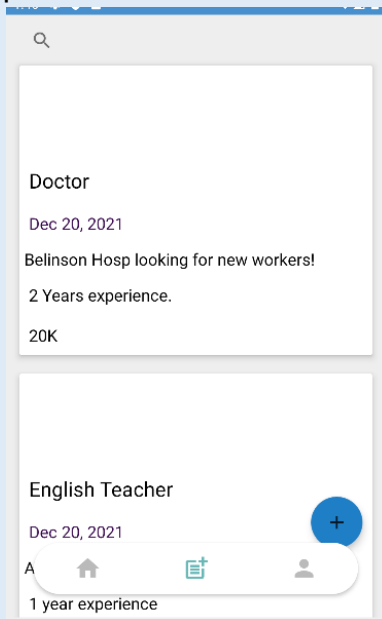
    public MyViewHolder(@NonNull View itemView, final OnItemClickListener listener) {
        super(itemView);
        title = itemView.findViewById(R.id.job_title1);
        description = itemView.findViewById(R.id.job_desc1);
        skills = itemView.findViewById(R.id.job_skills1);
        salary = itemView.findViewById(R.id.job_salary1);
        date = itemView.findViewById(R.id.job_date_board1);
        imageView = itemView.findViewById(R.id.profilePic);
        cardView = (CardView) itemView.findViewById(R.id.card_view);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (listener != null) {
                    int position = getAdapterPosition();
                    if (position != RecyclerView.NO_POSITION) {
                        listener.onItemClick(position);
                    }
                }
            }
        });
    }
}
```

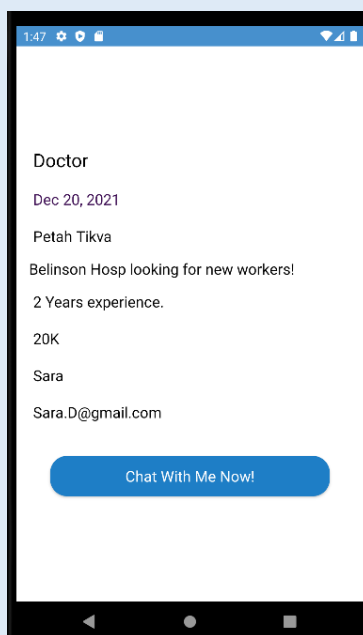
בגלל שהתצוגה שלנו היא בנויה ממסך גלילה שכל פריט בו מוצג כ CardView רצינו לאפשר למשתמש ללחוץ על המודעה ולקבל פירוט רחב יותר שלה. אז משום כך בשליפה אנחנו קובעים listener שבמידה ומסך ספציפי נלחץ אז הוא ישלוף את כל המידע הרצוי למסך זה.

לדוגמא:

אנו רואים פה את המודעה שפורסמה לתפקיד Doctor.



ברגע שנלחץ עליה נקבל את המסך הבא:



עכשיו נראה את הפונקציונאליות של האדפטר:

הכרזנו על מערך הנתונים שלנו ועוד משתנה שמהווה listener ברגע שנלחץ item.

```
public class adapterClass extends RecyclerView.Adapter<adapterClass.MyViewHolder>{  
  
    ArrayList<Data> dataList;  
    private OnItemClickListener mListener;
```

הפונקציה הבאה מייצרת מסך גלילה שאליו נשלפים הנתונים עם היכולת ללחוץ על ה item.

```

@NonNull
@Override
public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.job_board_item, parent, attachToRoot: false);
    MyViewHolder mvh = new MyViewHolder(v, mListener);
    return mvh;
}

```

בפונקציה הבאה אנחנו מזינים את הנתונים שאותם אנחנו נרצה להציג. בין אם בcardView או לאחר לחיצה עליו.

```

@Override
public void onBindViewHolder(@NonNull MyViewHolder holder, @SuppressWarnings("RecyclerView") int position) {

    holder.title.setText(dataList.get(position).getTitle());
    holder.description.setText(dataList.get(position).getDescription());
    holder.skills.setText(dataList.get(position).getSkills());
    holder.salary.setText(dataList.get(position).getSalary());
    holder.date.setText(dataList.get(position).getDate());

    holder.cardView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(v.getContext(), detailedJobData.class);
            intent.putExtra( name: "title", dataList.get(position).getTitle());
            intent.putExtra( name: "date", dataList.get(position).getDate());
            intent.putExtra( name: "description", dataList.get(position).getDescription());
            intent.putExtra( name: "skills", dataList.get(position).getSkills());
            intent.putExtra( name: "salary", dataList.get(position).getSalary());
            intent.putExtra( name: "email", dataList.get(position).getEmail());
            intent.putExtra( name: "contactName", dataList.get(position).getContactName());
            intent.putExtra( name: "location", dataList.get(position).getLocation());
            v.getContext().startActivity(intent);
        }
    });
}

```

בנוסף יש עוד פונקציה שהיא מחזירה את מס' ה Item ים שיהיו במסך

```

@Override
public int getItemCount() { return dataList.size(); }

```

לאחר מכן נעבור ל

מסך DetailedJobData:

אנחנו מתחילים את כל השדות שאנו רוצים להציג במסך.

```
public class detailedJobData extends AppCompatActivity {

    private Button goToChat;
    EditText email1;

    private TextView dTitle;
    private TextView dDate;
    private TextView dLocation;
    private TextView dSalary;
    private TextView dEmail;
    private TextView dContentName;
    private TextView dSkills;
    private TextView dDescription;
```

לאחר מכן אנחנו משייכים את המשתנים לסך התצוגה.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_detailed_job_data);

    goToChat = findViewById(R.id.button_enter_chat);
    email1 = findViewById(R.id.emailEditText);

    dTitle = (TextView) findViewById(R.id.job_detail_title);
    dDate = (TextView) findViewById(R.id.job_details_date_board);
    dDescription = (TextView) findViewById(R.id.job_details_desc);
    dSkills = (TextView) findViewById(R.id.job_details_skills);
    dSalary = (TextView) findViewById(R.id.job_details_salary);
    dEmail = (TextView) findViewById(R.id.job_details_email);
    dContentName = (TextView) findViewById(R.id.job_details_contact_name);
    dLocation = (TextView) findViewById(R.id.job_details_location);
```

מפעילים קישור בין המסך ל item ספציפי ב cardView ומקשרים בין המידע של השדות בשני המסכים.

```
//Receive data from adapter class job activity using intent
Intent intent = getIntent();
String title = intent.getStringExtra( name: "title");
String date = intent.getStringExtra( name: "date");
String description = intent.getStringExtra( name: "description");
String skills = intent.getStringExtra( name: "skills");
String salary = intent.getStringExtra( name: "salary");
String email = intent.getStringExtra( name: "email");
String contactName = intent.getStringExtra( name: "contactName");
String location = intent.getStringExtra( name: "location");
```

ובסוף אנחנו מזינים את המידע לתוך המשתנים שלנו במסך.

```
//Setting values
dTitle.setText(title);
dDate.setText(date);
dDescription.setText(description);
dSkills.setText(skills);
dSalary.setText(salary);
dEmail.setText(email);
dContentName.setText(contactName);
dLocation.setText(location);
```


:Chat

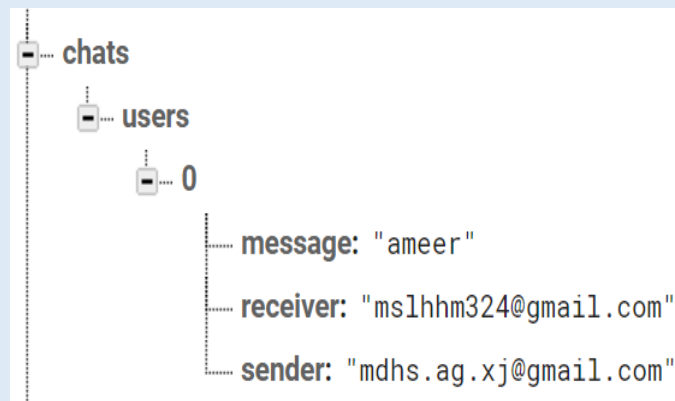
אחרי התעניינות המשתמש במודעה אפשר לעבור ל chat בין המשתמש ומפרסם המודעה דרך כפתור שנמצא על המודעה.

במסך ה chat יש חלון שאפשר לכתוב בו וכפתור send שכשלוחצים עליו ההודעה נשלחה ותוצג כ recyclerview.

איך זה מתנהל במסדי הנתונים :

כשלוחצים על כפתור שליחת ההודעה send - ההודעה שנכתבה בחלון תועלה למסדי הנתונים firebase לזמן אמת realTime database ואז כדי להציג אותה ב recyclerview שולפים אותה ממסדי הנתונים.

הערה : לכל שני משתמשים שונים יש שדה שמתאר את מקור ויעד ההודעה ואת ההודעה עצמה.



למשל פה בין שני המשתמשים בעלי המיילים שרואים אותם ביעד ומקור נפח chat 0 ואז כשיש בקשה ל chat בין שני משתמשים ששניהם שונים מאלה(המקור וגם היעד) תיווצר עוד בן 1 אלה אם המקור וגם היעד נמצאים ונפתח להם chat כבר ואז רק מעדכנים רק את השדה message.

סקיצה למסך ה chat:

