



ביה"ס: קריית חינוך ע"ש דוד אלעזר, מקיף ט'

ראשון לציון

שם הפרויקט: חנות למוצרי חשמל

שם החנות: **PowerPlus**



שם המגיש: רז עייני

כיתה: י"ב - 5

תעודת זהות: 328153101

שם החלופה: עבודת גמר תכנון ותכנות מערכות - טלפונים חכמים באנדרואיד  
- 883589

שמות המנחים: גנזיה סוזי, באורך מירי, ירמיה גיל

תאריך: יוני 2025

## תוכן עניינים:

2	תוכן עניינים:
3	הקדמה אישית:
3	הקדמה לפרויקט:
4	מבוא:
7	תיאור תחום הידע:
11	טבלת המסכים באפליקציה:
22	מבנה האפליקציה – תרשים זרימה:
23	טבלאות המחלקות בפרויקט:
32	תרשים UML של האפליקציה:
33	מדריך למשתמש:
38	סיכום אישי / רפלקציה:
40	נספחים:

## הקדמה אישית:

פרויקט זה מהווה עבורי את שיאו של תהליך למידה ממושך, שבו רכשתי ידע וניסיון בתחומי התכנות, פיתוח מערכות ופתרון בעיות טכנולוגיות. במהלך העבודה על הפרויקט התמודדתי עם אתגרים רבים, שלימדו אותי על חשיבות ההתמדה, החשיבה היצירתית והיכולת ללמוד ולהסתגל למצבים חדשים.

ברצוני להודות בראש ובראשונה למורי ולמדריכי, שליוו אותי לאורך הדרך, העניקו לי הכוונה מקצועית, תמיכה וסבלנות אין-קץ. תודה מיוחדת למשפחתי ולחבריי, שתמיד היו שם לעודד, ליעץ ולהזכיר לי שאין אתגר שלא ניתן להתגבר עליו. כמו כן, אני מודה לכל מי שסייע לי, במישרין או בעקיפין, בהשלמת הפרויקט.

אני מקווה שהעבודה על פרויקט זה תשמש עבורי קרש קפיצה להמשך הדרך בעולם הטכנולוגי ותאפשר לי להמשיך לפתח, ליצור וללמוד.

## הקדמה לפרויקט:

בעידן שבו הקניות המקוונות הופכות לחלק בלתי נפרד מחיינו, עלתה בי המחשבה לפתח אפליקציה ייעודית לחנות מוצרי חשמל, שתספק חוויית קנייה נוחה, יעילה ומתקדמת. הפרויקט נועד לתת מענה לצורך הגובר ברכישת מוצרי חשמל בצורה מהירה ומאובטחת, תוך שיפור חוויית המשתמש והתאמתה לדרישות השוק הדיגיטלי.

מטרת האפליקציה היא לאפשר למשתמשים לעיין במגוון מוצרים, לבצע חיפוש חכם, להוסיף מוצרים לעגלת הקניות ולבצע רכישה בצורה פשוטה ומהירה. האפליקציה עושה שימוש ב-Firebase - לניהול מסד הנתונים, ומציגה את המוצרים בצורה דינמית באמצעות RecyclerView, מה שמאפשר עדכון תכנים בצורה גמישה ויעילה.

במהלך פיתוח הפרויקט, התמודדתי עם אתגרים טכנולוגיים כמו שילוב מסד נתונים בענן, יצירת ממשק משתמש ידידותי ושיפור ביצועי האפליקציה. העבודה עליו חיזקה את יכולותיי בפיתוח אפליקציות Android, עבודה עם בסיסי נתונים, והתמודדות עם בעיות בזמן אמת.

אני מקווה כי האפליקציה שיצרתי תוכל להוות דוגמה לפתרון דיגיטלי מתקדם בתחום המסחר האלקטרוני, ולספק חוויית קנייה משופרת עבור המשתמשים.

חשוב לציין כי מדובר בפרויקט הדמיה, ולכן האפליקציה לא כוללת ממשק אמיתי עם ספקים או מלאי פיזי, אלא מתמקדת בתהליכי הקנייה עצמם ובחוויה הדיגיטלית של המשתמש.

## מבוא:

### תיאור הפרויקט והרקע למשימה

הפרויקט מתמקד ביצירת אפליקציית חנות למכשירי חשמל, אשר מדמה תהליך קנייה מלא – החל מעיון במוצרים ועד לרכישה. האפליקציה נבנתה כפרויקט גמר במגמת מדעי המחשב, ומטרתה להציג חוויית משתמש נוחה, עיצוב מודרני וניהול נתונים מאורגנים תוך שימוש בטכנולוגיות עכשוויות.

הבעיה המרכזית שהאפליקציה מדמה פתרון עבורה היא שיפור חוויית הקנייה בתחום מוצרי החשמל, תוך מיקוד בממשק משתמש אינטואיטיבי, השוואת מחירים, וסינון נוח לפי קטגוריות. מדובר באפליקציה המיועדת לצרכי תצוגה ולימוד בלבד, ואין בה ניהול מלאי אמיתי או חיבור למערכת לוגיסטית.

### מטרת האפליקציה והאופציות למשתמש

#### האפליקציה מספקת למשתמשים מגוון אפשרויות:

- הוספת מוצרים לעגלת הקניות כחלק מתהליך רכישה מדומה.
- התחברות למשתמש לצורך שמירת היסטוריית רכישות.
- שיפור חוויית המשתמש באמצעות עיצוב נקי ונוח לניווט.
- יכולת חיפוש וסינון מוצרים לפי קטגוריות.
- מערכת ניהול למנהלים הכוללת צפייה בהזמנות ועדכון סטטוס משלוח.
- שמירת נתונים ב - Firebase לאחסון מאובטח וזמין בענן.

### קהל היעד

האפליקציה מיועדת לכל אדם המעוניין לרכוש מוצרי חשמל, החל מבני נוער ועד למבוגרים. עם זאת, הקהל העיקרי הוא משתמשים בגילאי 18–50, בעלי זיקה לקניות מקוונות, המעוניינים בחוויית רכישה מהירה ומתקדמת.

## תיחום המערכת

תחום	פירוט
מה האפליקציה כוללת	<ul style="list-style-type: none"> <li>• הרשמה והתחברות משתמשים</li> <li>• הצגת מוצרים לפי קטגוריה</li> <li>• עגלת קניות אישית לכל משתמש</li> <li>• תהליך רכישה (סיכום ותשלום מדומה)</li> <li>• מערכת ניהול עבור מנהל (הזמנות, מבצעים)</li> <li>• הודעות קופצות למשתמשים על מבצעים.</li> </ul>
מה האפליקציה לא כוללת	<ul style="list-style-type: none"> <li>• ניהול מלאי אמיתי</li> <li>• שילוב עם ספקים חיצוניים</li> <li>• תשלום באשראי או סליקה אמיתית</li> <li>• ניהול החזרות / תקלות במוצרים</li> </ul>

## דרישות ומגבלות להפעלת התוכנה

אילוץ	תיאור
מערכת הפעלה	אנדרואיד 8.0 ומעלה
חומרה נדרשת	טלפון חכם עם חיבור אינטרנט
מסד נתונים	Firebase Realtime Database
הרשאות	חיבור לאינטרנט, הרשאת משתמש (Login)

## מונחים שכיחים במערכת

- **RecyclerView** - רכיב ב-Android להצגת רשימות דינמיות של מוצרים.
- **Firebase** - פלטפורמה לניהול מסדי נתונים בענן.
- **Authentication** - מערכת אימות משתמשים.

## תהליך המחקר והאתגרים המרכזיים:

### מחקר מקדים

ביצעתי סקירת שוק על אפליקציות מסחר אלקטרוני, תוך התמקדות בחוויית משתמש, זמני טעינה ומנגנוני חיפוש. בדקתי פתרונות אפשריים לניהול נתונים בענן, ובחרתי להשתמש ב-Firebase בשל נוחות האינטגרציה עם Android.

### אתגרים מרכזיים

- אינטגרציה עם **Firebase** - כיצד לשלוף נתונים ולהציגם בצורה אופטימלית.
- שיפור חוויית המשתמש - התאמה של ממשק המשתמש לשימוש יומיומי.
- ניהול נתונים דינמי - הצגת מוצרים ב-RecyclerView תוך עדכון בזמן אמת.

### חידושים והתאמות

- שילוב חיפוש חכם למוצרים לפי שם, קטגוריה ומחיר.
- שילוב חיפוש קולי למוצרים.
- שימוש בעיצוב מודרני עם Material Design.

## תיאור תחום הידע:

### האפליקציה נוגעת בנושאים מגוונים במדעי המחשב:

- בסיסי נתונים – שימוש ב - Firebase לניהול נתונים בענן.
- פיתוח ממשק משתמש ב- Android Studio – כולל RecyclerView, עיצוב דינמי, והתאמה למובייל.

### אובייקטים עיקריים:

#### מוצר (Item)

מייצג פריט בחנות. כולל שדות כמו:

- name – שם המוצר
- price – מחיר
- aboutItem – תיאור
- type – קטגוריה
- pic – תמונה
- company – חברה

#### עגלת קניות (Cart)

רשימת מוצרים שנבחרו על ידי המשתמש לפני ביצוע הזמנה.  
כל משתמש רשום מחזיק עגלה אישית הנשמרת תחת נתוניו במסד הנתונים.  
כל פריט בעגלה כולל את שם המוצר, מחיר, כמות, ותמונה.

### **משתמש (User)**

כל משתמש שנרשם לאפליקציה.

שדות עיקריים:

- uid – מזהה ייחודי
- fName, lName – שם פרטי ושם משפחה
- email, password, phone – פרטי התחברות ותקשורת
- cart – עגלת הקניות האישית של המשתמש

### **הזמנה (Order)**

נוצרת לאחר תשלום מדומה ומכילה את כל הפריטים שנבחרו.

שדות עיקריים:

- orderId, userId, items, totalPrice
- status – סטטוס הטיפול בהזמנה (Pending, Processing, Shipped, Delivered)
- timestamp – מועד ההזמנה
- address, paymentMethod – פרטי משלוח ותשלום

### **מבצע (Deal)**

מאפשר להחיל הנחה על מוצרים לפי קטגוריה או סוג.

שדות עיקריים:

- title, description, discountPercentage
- validUntil – תאריך תפוגה
- itemType – קטגוריה שעליה המבצע חל
- id – מזהה ייחודי



### קטגוריה (Category)

מייצגת קבוצת מוצרים מאותו סוג. לדוגמה: טלפונים, מחשבים, מקררים.  
משמשת לסינון והצגת פריטים לפי נושא.  
כל קטגוריה כוללת תמונה וכותרת.

### תגובה (Comment)

מאפשרת למשתמשים להגיב ולדרג מוצרים.  
שדות עיקריים:

- commentId, userId, userName
- commentText, rating – תוכן התגובה ודירוג

### ייצוג מידע

האפליקציה עושה שימוש במספר מבני נתונים כדי לארגן ולנהל את המידע ביעילות:

- Firebase Realtime Database** - מאגר נתונים מבוסס JSON, המתעדכן בזמן אמת.
- RecyclerView** - משמש להצגת רשימת המוצרים בצורה דינמית וחסכונית במשאבים.
- ArrayList** - משתמש להצגת רשימת המוצרים בעגלת הקניות בצורה מסודרת.

### נימוק לבחירה:


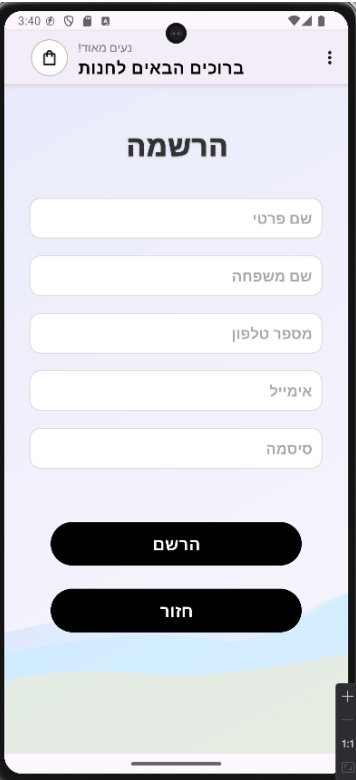
- Firebase** מאפשר גישה מהירה לנתונים ועדכונים בזמן אמת.
- השימוש ב- **RecyclerView** מאפשר תצוגה דינמית וחווית משתמש חלקה גם ברשימות גדולות.
- השימוש ב- **ArrayList** מאפשר ניהול וסידור של רשימת המוצרים בצורה גמישה ונוחה, כולל הצגה לפי סדר מוגדר.

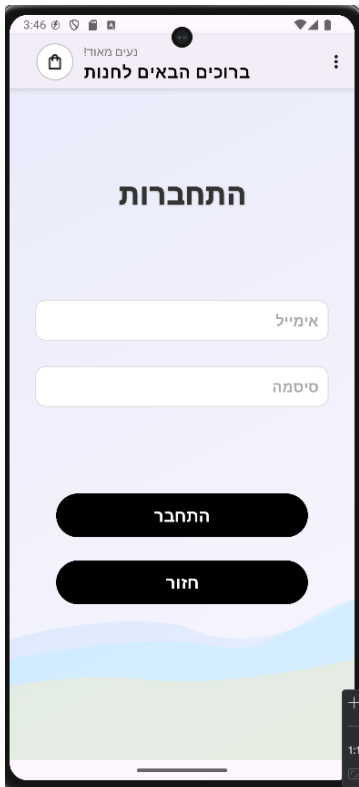
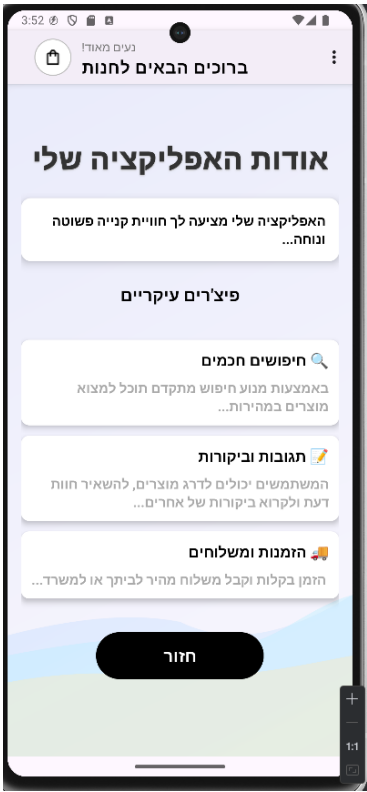
## הרחבות מתקדמות בפרויקט

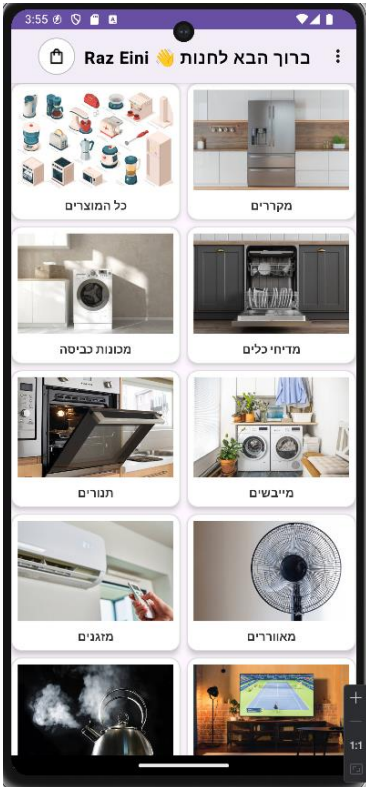
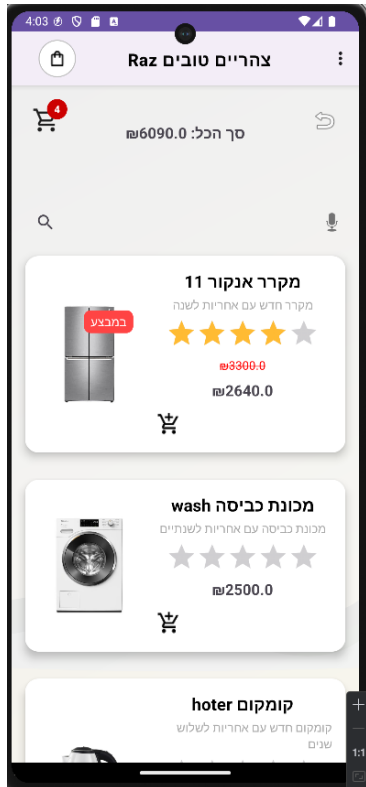
במהלך הפיתוח בחרתי לא להסתפק בפיצ'רים הבסיסיים, אלא להרחיב את האפליקציה בצורה משמעותית, בין השאר על ידי שילוב הרכיבים הבאים:

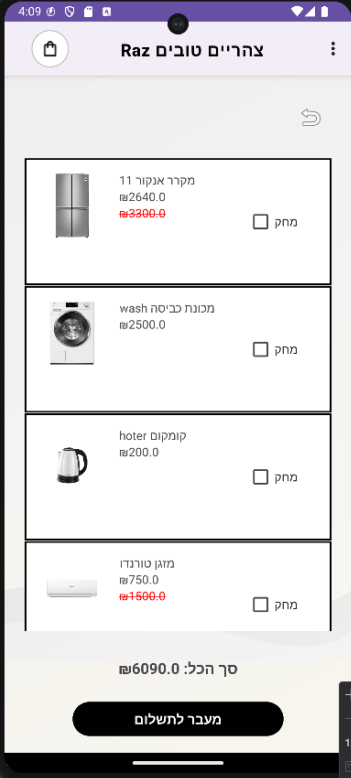

- **SpeechRecognizer** - אפשרות לחיפוש קולי של מוצרים, לשיפור חוויית המשתמש ולנגישות גבוהה יותר.
- **AlarmManager** - תזמון התראות על מבצעים חדשים, כולל יצירת NotificationChannel מותאם עם צליל ייחודי, תמיכה בגרסאות אנדרואיד חדשות ובדיקה של הרשאות נדרשות.
- **SharedPreferences** - שמירת נתונים לאורך זמן (כמו התחברות, תפקיד המשתמש, הגדרות), מה שאפשר חוויית משתמש עקבית ומודרנית.
- **AuthenticationService** - בניית שירות עצמאי לניהול התחברות והרשמה דרך Firebase, תוך הפרדה ברורה מהקוד שמנהל את ממשק המשתמש. זה הקל על ניהול אבטחה והפך את הקוד למודולרי, קריא וניתן לתחזוקה.
- **DatabaseService** - בניית שכבת שירות נפרדת המנהלת את כל העבודה מול מסד הנתונים בענן (Firebase Realtime Database). היא מאפשרת שליפה, עדכון ושמירה של נתונים בצורה נקייה ומסודרת, תוך שימוש ב-Callbacks ו-Interfaces.

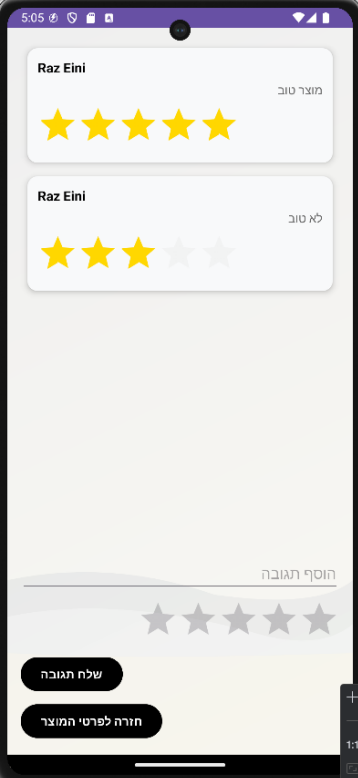
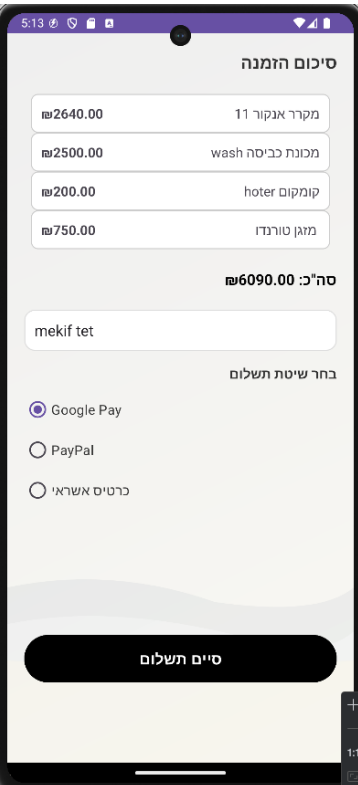
## טבלת המסכים באפליקציה:



שם המסך	תיאור המסך	תמונת המסך
<b>MainActivity</b>	מסך פתיחה לאפליקציה המכיל רקע מעוצב, מכיל 3 כפתורי ניווט, למסך ההרשמה למסך ההתחברות ולמסך האודות.	
<b>Register</b>	מסך הרשמה המכיל שדות למשתמש למילוי: שם פרטי, שם משפחה, מספר טלפון, אימייל סיסמה. המסך מכיל כפתור ששומר את פרטי המשתמש בפייר-בייס לאחר מילוי כל השדות בצורה תקנית וכפתור חזרה אם המשתמש מעוניין לחזור למסך הפתיחה.	

	<p>מסך ההתחברות מכיל שדות למשתמש שכבר נרשם למילוי, האימייל לאיתור נרשם והסיסמה איתה נרשם. המסך מכיל כפתור ששולח משתמש שממלא את השדות בצורה תקנית לעמוד הקטגוריות של החנות, וגם מכיל כפתור חזרה למסך הפתיחה.</p>	<p><b>Login</b></p>
	<p>מסך האודות מכיל כרטיסיות ובהם כתוב אודות הפיצ'רים העיקריים של האפליקציה, בנוסף העמוד מכיל כפתור חזרה למסך הפתיחה.</p>	<p><b>Odor</b></p>

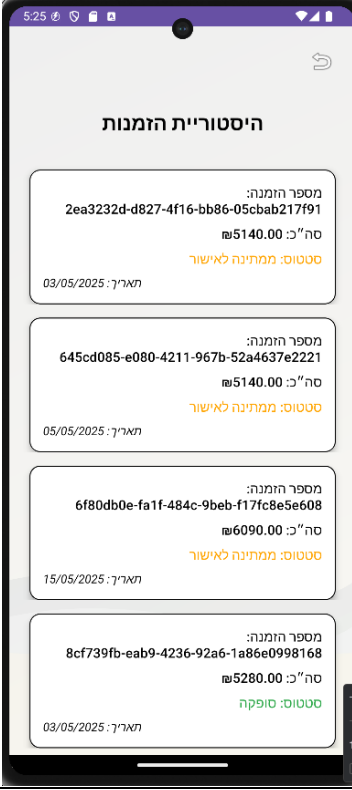
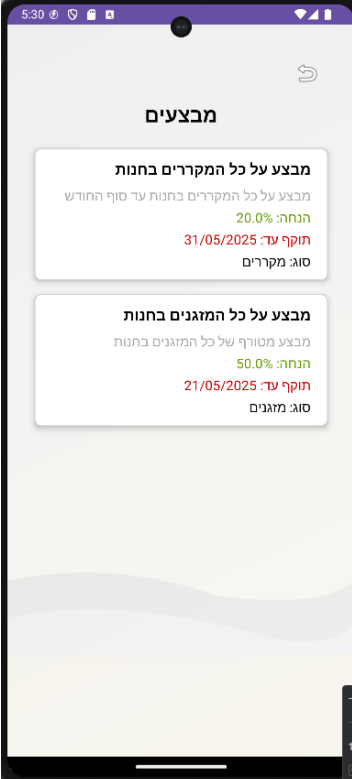
	<p>מסך קטגוריות המכיל תמונות של קטגוריות של מוצרים המוצגות בפורמט גלילה, לחיצה על תמונה של אחת מהקטגוריות תוביל את המשתמש לעמוד החנות המסוננת לפי אותה קטגוריה.</p>	<p><b>CategoriesActivity</b></p>
	<p>עמוד החנות המכיל כפתור המעביר את המשתמש לעגלת הקניות שעליו מופיעים מספר הפריטים בעגלת המשתמש, כפתור חזרה לעמוד הקטגוריות, חיפוש חכם של מוצרים, חיפוש קולי של מוצרים, כותרת עם המחיר הכולל של כל המוצרים אותם המשתמש הוסיף לעגלה, רשימת המוצרים בחנות, הרשימה מכילה כרטיסיות של המוצרים, בכרטיסיות מופיעה תמונת המוצר, שם המוצר, תיאור קצר של המוצר, דירוג המוצר הממוצע לפי התגובות על המוצר, מחיר המוצר (ואם יש למוצר הנחה גם המחיר לאחר ההנחה) (וכפתור הוספת המוצר לעגלת הקניות של המשתמש).</p>	<p><b>ShopActivity</b></p>

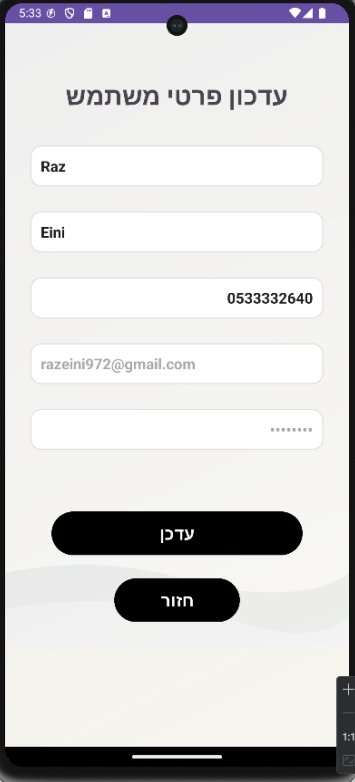

	<p>עמוד העגלה מכיל כפתור חזרה לעמוד החנות, רשימה של כל המוצרים אותם המשתמש הוסיף לעגלה בעמוד החנות, כל כרטיסיה של מוצר בעגלה מכילה את תמונת המוצר, שם המוצר, מחיר המוצר (ואם יש לו הנחה אז גם מחיר לאחר ההנחה, וכפתור מחיקה המוחק את המוצר מעגלת המשתמש. בנוסף עמוד העגלה מכיל כותרת של המחיר הכולל של המוצרים בעגלה וכפתור שמעביר את המשתמש לעמוד התשלום.</p>	<p><b>CartActivity</b></p>
	<p>עמוד פרטי המוצר המכיל את שדות המוצר המלאים בנשלפים מהפייר-בייס, בנוסף להם ישנו גם דירוג המוצר הממוצע לפי התגובות, בנוסף ישנו כפתור מעבר לעמוד התגובות וכפתור חזרה לעמוד החנות.</p>	<p><b>ItemDetailActivity</b></p>

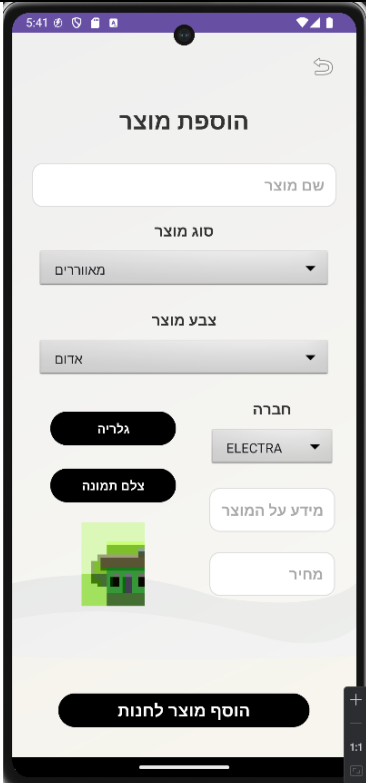
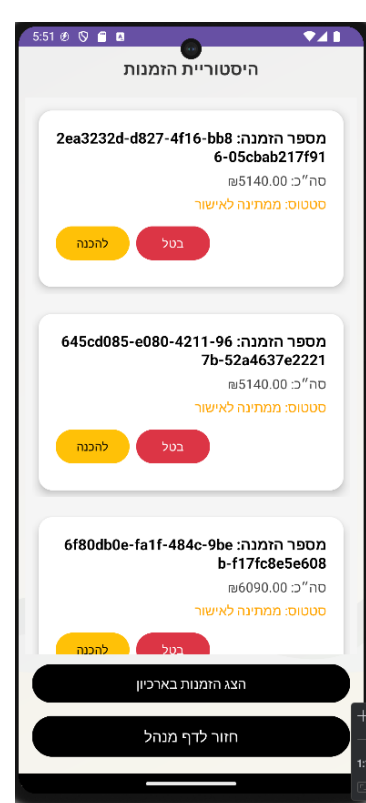
	<p>עמוד תגובות המשתמשים, המכיל אפשרות הוספת תגובה, כל כרטיסיית תגובה מכילה את שם המשתמש המלא, התגובה של המשתמש על המוצר, והדירוג שהמשתמש נותן בתגובה למוצר, בנוסף ישנו כפתור שלח תגובה כאשר המשתמש ממלא את שדות התגובה באופן תקין וגם כפתור חזרה לעמוד פרטי המוצר.</p>	<p><b>CommentActivity</b></p>
	<p>עמוד תשלום המכיל רשימה של המוצרים בעגלה (שם המוצר והמחיר), בנוסף שדות למשתמש למילוי בכדי לבצע הזמנה, שדה אחד של כתובת ושדה אחר של שיטת תשלום, בנוסף ישנו כפתור המעביר את המשתמש לעמוד ה"תודה על שקנית" ושומר את ההזמנה של המשתמש אם המשתמש מלא את השדות באופן תקין.</p>	<p><b>PaymentActivity</b></p>

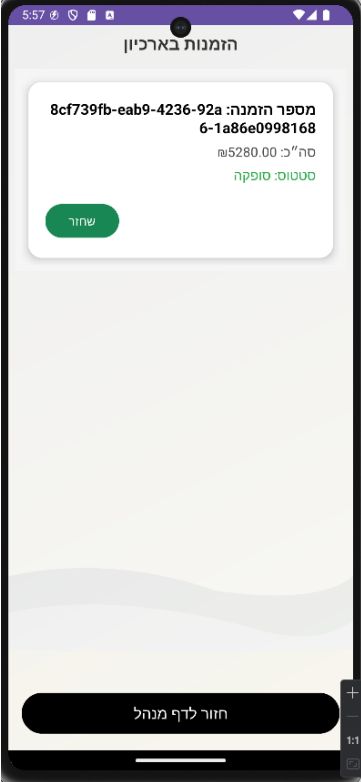
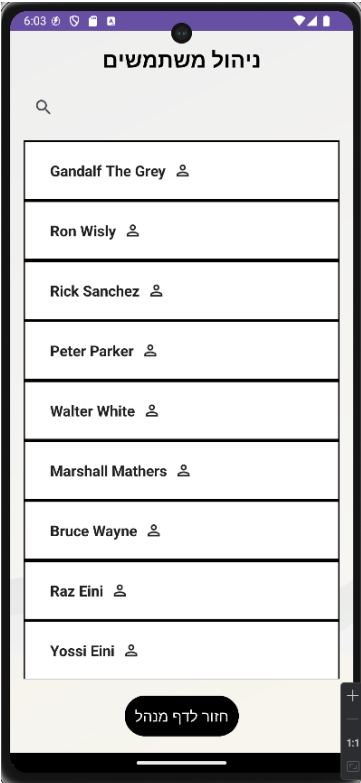
	<p>עמוד "תודה על שקנית" המכיל כותרת של "תודה על ההזמנה!" וטקסט של "ההזמנה שלך התקבלה ותישלח אליך בהקדם", בנוסף ישנו גם כפתור חזרה לעמוד החנות שגם מאפשר את עגלת המשתמש לביצוע הזמנה חדשה.</p>	<p><b>ThankYouActivity</b></p>
	<p>עמוד המשתמש מכיל בתוכו ארבעה כפתורים, ישנו כפתור המוביל את המשתמש לעמוד היסטורית ההזמנות שלו, ישנו כפתור המוביל את המשתמש לעמוד המבצעים של החנות, ישנו כפתור המוביל את המשתמש לעמוד בו הוא יכול לעדכן את פרטיו וישנו גם כפתור המאפשר למשתמש להתנתק ומעביר אותו לעמוד ההתחברות.</p>	<p><b>UserAfterLoginPage</b></p>

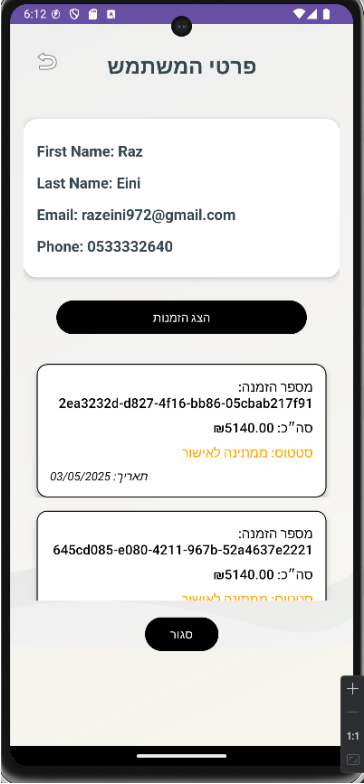
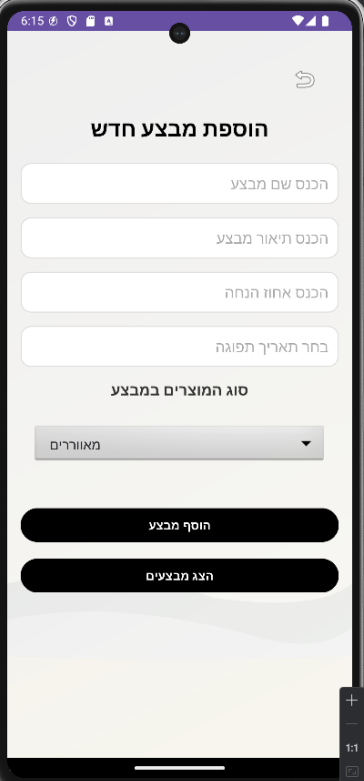


	<p>בעמוד ההזמנות ישנה רשימה של כל ההזמנות אותם ביצע המשתמש, כל כרטיסייה של הזמנה מכילה את מזהה ההזמנה, את המחיר הכולל של ההזמנה, סטטוס ההזמנה, שיכול להשתנות ל: "ממתינה לאישור", "בהכנה", "נשלחה" ורק כאשר המשתמש מאשר שהוא קיבל את ההזמנה, הוא יכול לחלוף על כפתור מוסתר במסך המופיע רק בהזמנות שנשלחו המשנה את סטטוס ההזמנה ל"סופקה". בנוסף ההזמנה מכילה את תאריך ביצוע ההזמנה. בעמוד ישנו גם כפתור חזרה לעמוד המשתמש.</p>	<p><b>OrderHistoryActivity</b></p>
	<p>בעמוד המבצעים ישנה רשימה של מבצעים אותם הוסיף המנהל, כל כרטיסייה של מבצע מכילה את שם המבצע, המידע אודות מבצע, אחוז ההנחה, תוקף ההנחה וסוג המוצר בהנחה. בעמוד ישנו גם כפתור חזרה לעמוד המשתמש.</p>	<p><b>DealsActivity</b></p>

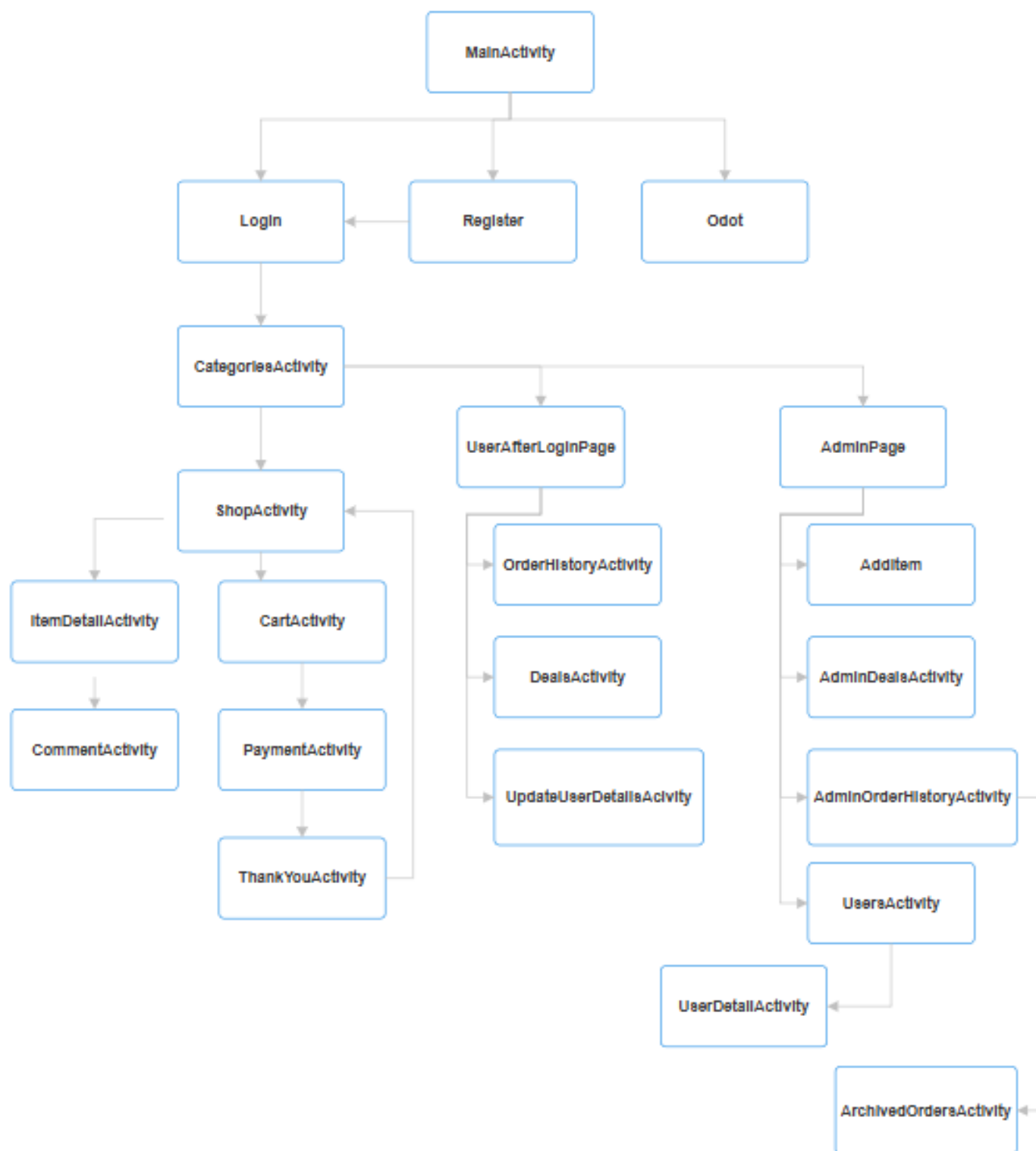
	<p>בעמוד עדכון פרטי המשתמש ישנם שדות עם פרטי המשתמש אשר הוא מילא אותם בעמוד ההרשמה, המשתמש ברשותו לשנות את שמו, שם משפחתו ומספר הטלפון שלו בלבד. בעמוד ישנו גם כפתור חזרה לעמוד המשתמש.</p>	<p><b>UpdateUserDetailsActivity</b></p>
	<p>בעמוד המנהל ישנם חמישה כפתורים, ישנו כפתור המעביר את המנהל לעמוד הוספת מוצר, ישנו כפתור המעביר את המנהל לעמוד היסטוריית הזמנות של כלל המשתמשים, ישנו כפתור המעביר את המנהל לעמוד המשתמשים, ישנו כפתור המעביר את המנהל לעמוד הוספת מבצע וישנו כפתור התנתקות המעביר את המנהל לעמוד ההתחברות.</p>	<p><b>AdminPage</b></p>

	<p>עמוד הוספת המוצר מכיל שדות אודות פרטי המוצר אותו המנהל רוצה להוסיף לעמוד החנות, ישנו שדה של שם המוצר, שדה של סוג המוצר, שדה של צבע המוצר, שדה של החברה של המוצר, שדה של מידע על המוצר, שדה של מחיר המוצר, וגם אפשרות לצלם את המוצר או להוסיף תמונה שלו מהגלריה. בנוסף ישנו גם כפתור הוסף מוצר לחנות המוסיף את המוצר אם כל השדות מולאו בצורה תקנית וגם כפתור חזור המחזיר את המנהל לעמוד המנהל.</p>	<p><b>AddItem</b></p>
	<p>בעמוד ההזמנות של המנהל, המנהל רואה את רשימת ההזמנות של כל המשתמשים, בכל כרטיסייה של הזמנה ישנו מזהה ההזמנה, המחיר הכולל של ההזמנה, סטטוס ההזמנה אותו הוא יכול לשנות עם הכפתורים, הוא יכול לשלוח את ההזמנה ל"הכנה", ולאחר מכן לשלוח אותה למשתמש וזה ישנה לה את הסטטוס ל"נשלחה", או שהמנהל יכול להחליט לבטל את ההזמנה. בנוסף ישנו כפתור המעביר את המנהל לעמוד ארכיון ההזמנות של המשתמשים, הזמנות המועברות לארכיון רק אם המשתמש אישר שהוא קיבל אותם והן מקבלות סטטוס "סופקה", וישנו כפתור המחזיר את המנהל לעמוד המנהל.</p>	<p><b>AdminOrderHistoryActivity</b></p>

	<p>עמוד הארכיון של ההזמנות אותן המנהל מעביר לארכיון, העמוד מכיל רשימה של ההזמנות שהועברו לארכיון, כרטיסייה של הזמנה בארכיון מכילה גם כן את מזהה ההזמנה, המחיר הכולל של ההזמנה וסטטוס ההזמנה, ובנוסף כפתור המאפשר למנהל ל"שחזר" את ההזמנה ולהחזיר אותה לעמוד ההזמנות של המנהל. בעמוד ישנו גם כפתור חזרה לעמוד ההזמנות של המנהל.</p>	<p><b>ArchivedOrdersActivity</b></p>
	<p>בעמוד המשתמשים ישנה רשימה של כל המשתמשים הרשומים באפליקציה, הרשימה מעילה רק את שם המשתמש המלא ובלחיצה על אחד המשתמשים, המנהל יועבר לעמוד ובו פרטי המשתמש המלאים. בנוסף המנהל יכול לבצע מחיקה של משתמש בעת לחיצה ארוכה על שם המשתמש. בנוסף יש פיצ'ר של חיפוש חכם של משתמשים לפי שם המשתמש. ישנו גם כפתור חזרה המחזיר את המנהל לדף המנהל.</p>	<p><b>UsersActivity</b></p>

	<p>בעמוד פרטי המשתמש בדומה לעמוד פרטי המוצר, ישנם כל פרטי המשתמש אותם מילא בעמוד ההרשמה, הנשלפים מהפייר-בייס, בנוסף לכך ישנו כפתור הפותח את רשימת ההזמנות של המשתמש אותה אפשר לסגור עם כפתור.</p>	<p><b>UserDetailActivity</b></p>
	<p>עמוד הוספת המבצע של המנהל מכיל שדות למילוי אודות המבצע: שם המבצע, תיאור המבצע, אחוז ההנחה, תאריך תפוגה וגם סוג המוצרים במבצע. לאחר מילוי שדות המבצע באופן תקין המנהל יכול ללחוץ על כפתור הוספת המבצע. ישנו גם כפתור של הצגת המבצעים הפותח רשימה של מבצעים בדומה לרשימה בעמוד המבצעים של המשתמש אותה אפשר לסגור עם כפתור. בנוסף ישנו כפתור חזרה המחזיר את המנהל לעמוד המנהל.</p>	<p><b>AdminDealsActivity</b></p>

## מבנה האפליקציה – תרשים זרימה:



## טבלאות המחלקות בפרויקט

שם המחלקה	תפקיד המחלקה	הסבר תכונות המחלקה	הסבר פעולות המחלקה
User	מייצגת משתמש באפליקציה – כולל פרטים אישיים, פרטי התחברות, ועגלת קניות משויכת.	<b>uid:</b> - מזהה ייחודי של המשתמש <b>email:</b> - כתובת האימייל של המשתמש <b>password:</b> - סיסמת המשתמש <b>fName:</b> - שם פרטי <b>lName:</b> - שם משפחה <b>phone:</b> - מספר טלפון <b>cart:</b> - עגלת קניות של המשתמש מסוג Cart	<b>getId() / setId():</b> מחזירה/מעדכנת את מזהה המשתמש <b>getEmail() / setEmail():</b> מחזירה/מעדכנת את כתובת האימייל <b>getPassword() / setPassword():</b> מחזירה/מעדכנת את הסיסמה <b>getfName() / setfName():</b> מחזירה/מעדכנת את השם הפרטי <b>getlName() / setlName():</b> מחזירה/מעדכנת את שם המשפחה <b>getPhone() / setPhone():</b> מחזירה/מעדכנת את מספר הטלפון <b>getCart() / setCart():</b> מחזירה/מעדכנת את עגלת הקניות <b>toString():</b> - מחזירה מחרוזת טקסט עם כל פרטי המשתמש לצרכי הדפסה או ניפוי שגיאות

<p><b>User():</b> - בנאי ריק המאתחל את השדות עם ערכים ברירת מחדל</p> <p>-</p> <p><b>User(userId,fName,lName, email,password, cart):</b> בנאי המאתחל את כל השדות</p>			
<p><b>- addItem(Item item):</b> מוסיפה פריט לעגלה. אם הרשימה ריקה, יוצרת רשימה חדשה.</p> <p><b>getItems():</b> מחזירה את רשימת הפריטים בעגלה.</p> <p><b>-setItems(List&lt;Item&gt; items):</b> קובעת את רשימת הפריטים בעגלה.</p> <p><b>- removeItem(int index):</b> מסירה פריט מהעגלה לפי מיקומו ברשימה.</p> <p><b>toString():</b> מחזירה מחרוזת עם כל פריטי העגלה – לצרכים של הדפסה/ניפוי שגיאות.</p>	<p><b>items:</b> - רשימה של פריטים (&lt;List&lt;Item&gt;) שנמצאים בעגלה.</p>	<p>מייצגת עגלת קניות של משתמש, שמכילה רשימת פריטים מהחנות.</p>	<p><b>Cart</b></p>



Category	מייצגת קטגוריה של מוצרים באפליקציה, כולל שם ותמונה מייצגת.	<b>name:</b> - שם הקטגוריה (לדוגמה: "מקררים", "טלוויזיות"). <b>imageResId:</b> - מזהה של תמונת הקטגוריה (משאב מסוג int מתוך הקבצים הגרפיים של האפליקציה).	<b>getName():</b> - מחזירה את שם הקטגוריה. <b>getImageResId():</b> - מחזירה את מזהה המשאב של התמונה. <b>Category(name, imageResId):</b> - מבניי את שם הקטגוריה ומזהה התמונה
Item	מייצגת פריט/מוצר בחנות – כולל מאפיינים כמו שם, סוג, צבע, מחיר, תמונה ועוד.	<b>id:</b> - מזהה ייחודי של הפריט <b>name:</b> - שם הפריט (לדוגמה: "טלפון סמסונג") <b>type:</b> - סוג המוצר או הקטגוריה שלו <b>color:</b> - צבע המוצר <b>company:</b> - שם החברה המייצרת <b>aboutItem:</b> - תיאור טקסטואלי של הפריט <b>price:</b> - מחיר המוצר (מספר מסוג double) <b>pic:</b> - כתובת של תמונת המוצר (URL או נתיב פנימי באפליקציה)	<b>get/set</b> - לכל שדה – מאפשרים גישה ועריכה לכל אחד מהשדות הנ"ל <b>toString():</b> - מחזירה מחרוזת עם כל המידע על הפריט, לצורכי הדפסה או ניפוי שגיאות <b>Item():</b> - בנאי ריק המאתחל את השדות עם ערכים ברירת מחדל <b>Item(id, name, type, color, company, aboutItem, price, pic):</b> - בנאי המאתחל את כל השדות

<p><b>Comment</b></p>	<p>מייצגת תגובה של משתמש על פריט – כולל טקסט, דירוג, מזהה משתמש ושם.</p>	<p><b>commentId:</b> - מזהה ייחודי של התגובה</p> <p><b>userId:</b> - מזהה המשתמש שכתב את התגובה</p> <p><b>commentText:</b> - הטקסט של התגובה שנכתבה</p> <p><b>rating:</b> - דירוג מספרי (מסוג float) שהמשתמש נתן למוצר</p> <p><b>userName:</b> - שם המשתמש שכתב את התגובה</p>	<p><b>get/set</b> - לכל שדה – מאפשרים גישה ועריכה לכל אחד מהשדות הנ"ל</p> <p><b>Comment():</b> - בנאי ריק המאתחל את השדות עם ערכים ברירת מחדל</p> <p><b>Comment(commentId, userId, commentText, rating, userName):</b> - המאתחל את כל השדות</p>
<p><b>Deal</b></p>	<p>מייצגת מבצע על מוצרים באפליקציה – כולל שם, תיאור, אחוז הנחה, תאריך תפוגה וסוג פריט.</p>	<p><b>id:</b> - מזהה ייחודי של המבצע</p> <p><b>title:</b> - כותרת המבצע (למשל "מבצע קיץ")</p> <p><b>description:</b> - תיאור של מהות המבצע</p> <p><b>discountPercentage:</b> - אחוז ההנחה שמופעל על המוצר</p> <p><b>validUntil:</b> - תאריך תפוגת המבצע (בפורמט טקסטואלי dd/MM/yyyy)</p>	<p><b>get/set</b> - לכל שדה – גישה ועריכה לכל פרטי המבצע</p> <p><b>isValid():</b> - בודקת האם המבצע עדיין בתוקף לפי תאריך התפוגה לעומת התאריך הנוכחי</p> <p><b>Deal():</b> - בנאי ריק המאתחל את השדות עם ערכים ברירת מחדל</p> <p><b>Deal(id, title, description, discountPercentage, validUntil, itemType):</b> - המאתחל את כל השדות</p>

	<b>itemType: סוג</b> המוצרים עליהם חל המבצע		
<b>Order(userId, items):</b> - בנאי מלא שמחשב גם את הסכום הכולל, מייצר מזהה ותאריך <b>Order():</b> - בנאי ריק שמבין רשימה ריקה של מוצרים ומזהה חדש <b>calculateTotalPrice():</b> מחשב את המחיר הכולל של ההזמנה על בסיס מחירי הפריטים <b>getFormattedDate():</b> - מחזיר את התאריך בפורמט קריא (dd/MM/yyyy) <b>toString():</b> - מציג מחרוזת תיאור של ההזמנה <b>getters &amp; setters:</b> - גישה ועריכה לשדות כמו סטטוס, כתובת, תאריך, מחיר, ועוד	<b>orderId:</b> - מזהה ייחודי להזמנה (נוצר אוטומטית בעזרת UUID) <b>items:</b> - רשימת פריטים (Item) שהוזמנו <b>totalPrice:</b> - המחיר הכולל של ההזמנה status :- מצב ההזמנה ("Pending", "Processing", "Shipped", "Delivered") <b>timestamp:</b> - תאריך ביצוע ההזמנה (ב- milliseconds) <b>userId:</b> - מזהה המשתמש שהזמין <b>address:</b> - כתובת למשלוח	ייצוג של הזמנה שבוצעה ע"י משתמש – כוללת פרטי מוצרים, משתמש, סכום כולל, סטטוס ועוד.	<b>Order</b>

	<b>- paymentMethod:</b> אמצעי תשלום (כגון "אשראי", "פייפאל", "מזומן")		
--	--	--	--

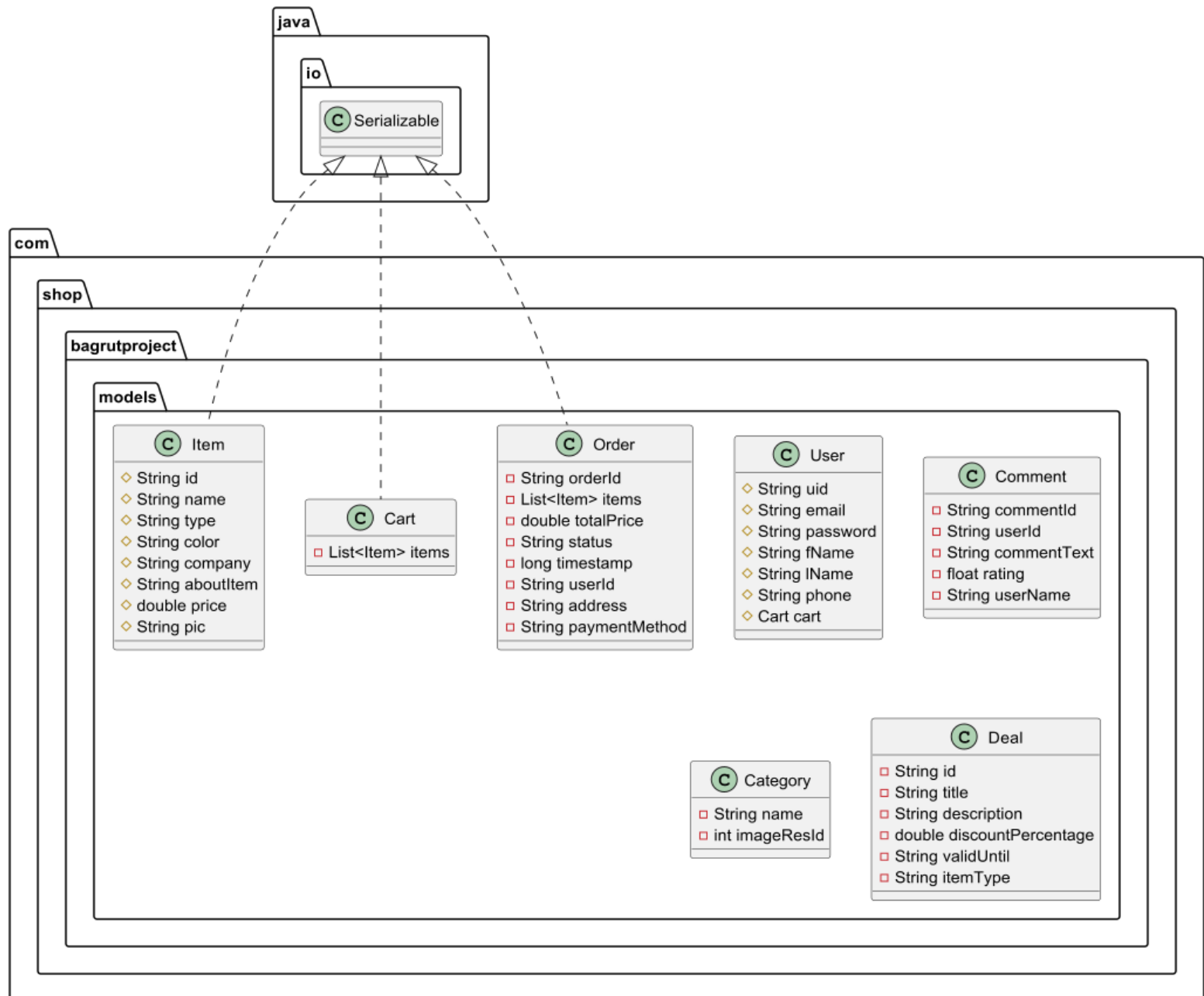
שיטת השמירה	השדות	שם הקובץ
Realtime Firebase	cart, email, fName, lName, password, phone, uid	<b>Users</b>
Realtime Firebase	aboutItem, color, company, id, name, numberRate, pic, price, rate, sumRate, type	<b>items</b>
Realtime Firebase	commentId, userId, commentText, rating, userName	<b>comments</b>
Realtime Firebase	id, title, description, discountPercentage, validUntil, itemType, valid	<b>deals</b>
Realtime Firebase	orderId, items, totalPrice, status, timestamp, userId, address, paymentMethod	<b>Orders</b>

Realtime Firebase	orderId, items, totalPrice, status, timestamp, userId, address, paymentMethod	archivedOrders
-------------------	---	----------------

## תמונות להמחשה



## תרשים UML של האפליקציה:





## מדריך למשתמש:

### מדריך משתמש – PowerPlus

ברוך הבא ל-PowerPlus – אפליקציית קנייה אונליין למוצרי חשמל. האפליקציה מאפשרת ללקוחות לעיין, לבחור ולהזמין מוצרים בקלות ובנוחות, תוך ניהול עגלה אישית, קבלת הצעות מבצע, מעקב אחר הזמנות ועוד. בנוסף, קיים ממשק ייעודי למנהלים לניהול החנות וההזמנות.

#### א. מהות האפליקציה

האפליקציה משמשת חנות וירטואלית למוצרי חשמל, ומציעה:

- דפדוף לפי קטגוריות או חיפוש חופשי.
- עגלת קניות אישית לכל משתמש.
- מבצעים תקופתיים והנחות.
- תהליך הזמנה פשוט עם אפשרות למעקב.
- ממשק ניהול עבור בעלי החנות.

#### ב. התקנה והפעלה

##### דרישות מוקדמות:

- מכשיר אנדרואיד (נבדק על גרסאות 10, 11, 12).
- חיבור אינטרנט פעיל (Wi-Fi או נתונים סלולריים).
- אין צורך ברכיבי חומרה מיוחדים.
- שטח אחסון פנוי: לפחות MB100 להתקנה תקינה.

##### התקנה:

1. הורד את קובץ ההתקנה (APK) דרך קישור או מהחנות הרלוונטית.
2. אשר התקנה ממקורות לא ידועים (אם מתבקש).
3. לחץ על קובץ ההתקנה ובחר "התקן".
4. בסיום – לחץ "פתח".

## הרשאות נדרשות:

- גישה לאינטרנט – לצורך טעינת המוצרים, עגלות, ושליחת הזמנות.
- גישה למיקרופון – עבור חיפוש קולי (אופציונלי, באישור המשתמש בלבד).
- גישה להתראות – עבור התראות על מבצעים באפליקציה.

## ג. הפעלת ממשק המשתמש (מפת ניווט)

מסך פתיחה: 🏠

- בחירה בין התחברות למשתמש קיים או הרשמה למשתמש חדש.

התחברות / הרשמה: 👤

- משתמש חדש ימלא טופס: שם, אימייל, סיסמה, טלפון.
- לאחר ההרשמה תיווצר באופן אוטומטי עגלת קניות אישית.
- משתמש קיים יתחבר בעזרת אימייל וסיסמה.

מסך קטגוריות: 📁

- בחירה בין תחומים שונים: טלוויזיות, מקררים, מזגנים ועוד.
- לחיצה על קטגוריה תוביל לדף מוצרים מותאם.

דף החנות: 🛒

- הצגת מוצרים לפי קטגוריה או כלליים.
- לכל מוצר: תמונה, שם, תיאור קצר ומחיר.
- כפתור "הוסף לעגלה".

חיפוש: 🔍

- שורת חיפוש בראש המסך.
- חיפוש לפי שם מוצר או מילות מפתח.
- כולל אפשרות חיפוש קולי.


עגלת קניות: 🛒

- רשימת הפריטים שבבחרו.

- הצגת מחיר כולל (כולל הנחות).
- אפשרות למחוק פריטים ע"י כפתור מחיקה.
- מעבר לתשלום.

 תשלום:

- מילוי כתובת.
- סימולציה של בחירת שיטת תשלום (אין תשלום אמיתי).
- הודעה שההזמנה התקבלה תוצג לאחר מכן.

 סטטוס הזמנה:

- צפייה ברשימת ההזמנות שבוצעת.
- סטטוס ההזמנה מוצג:
  - **Pending** – ממתינה לשליחה.
  - **Processing** – בהכנה.
  - **Shipped** – משלחה.
  - **Delivered** – התקבלה (מאושר ע"י הלקוח).

 מבצעים:

- עמוד ייעודי להצגת מבצעים פעילים.
- אם מוצר משתתף במבצע – יוצג מחירו לאחר הנחה.




 הגדרות ושינוי פרטים:

- שינוי שם פרטי, משפחה, טלפון.
- הנתונים מתעדכנים מיידית.

#### ד. מגבלות ואילוצים טכניים


מגבלה	רכיב/שדה
חייבת להיות תקינה וייחודית	כתובת אימייל
עד 20 תווים	שדה שם פרטי/משפחה
לפחות 6 תווים	אורך סיסמה
עד MB2	תמונת מוצר
דורש הרשאת מיקרופון	חיפוש קולי
צפייה מוגבלת בלבד – אין אפשרות להזמין	מצב לא מקוון

#### ה. הודעות למשתמש

- "שגיאה בחיבור לאינטרנט – בדוק את החיבור ונסה שוב."
-  "המוצר נוסף לעגלה!"
-  "ההרשמה הושלמה בהצלחה."
-  "אנא מלא את כל השדות המדרשים."

#### ו. הגדרות ועיצוב

- אפשרות לעבור בין מצב בהיר ל-מצב כהה (אוטומטי לפי מסך).
- הגדרות משתמש זמינות בתפריט "החשבון שלי".
- העיצוב כולל פינות מעוגלות, צבעים מודרניים, וצבעים בגווני שחור-לבן.
- ניווט פשוט עם כפתורי חזרה וניווט תחתון

 ממשק מנהל – Admin בלבד

(נגיש רק למנהלים בעלי הרשאה מתאימה)

- ניהול מוצרים – הוספת פריטים לחנות.
- ניהול מבצעים – יצירת מבצע עם אחוז הנחה, תיאור ותאריך תפוגה.
- ניהול הזמנות – צפייה ומעבר בין סטטוסים של הזמנות.

- ארכיון הזמנות – צפייה בהיסטוריית הזמנות, שחזור לפי צורך.
- ניהול משתמשים – צפייה במשתמשים רשומים, עדכון/חיפוש לפי שם/מייל.

#### טיפ לסיום:💡

אם נתקלת בשגיאה או משהו לא עובד, ודא שיש חיבור אינטרנט פעיל.  
רוב הבעיות נפתרות ביציאה וכניסה מחדש לאפליקציה.

## סיכום אישי / רפלקציה:

העבודה על פרויקט PowerPlus הייתה עבורי תהליך משמעותי, מאתגר ומלמד. מהרגע הראשון ידעתי שזו לא תהיה משימה פשוטה – בניית אפליקציה שלמה מאפס, עם כל הפיצ'רים שרציתי לכלול, דרשה ממני תכנון מדויק, התמדה, והמון למידה עצמאית.

### א. תהליך העבודה – הצלחות ואתגרים

הצלחתי לבנות אפליקציה שעונה על הצרכים שהצבתי לעצמי: מערכת הרשמה והתחברות, עיון במוצרים לפי קטגוריות, עגלה אישית, תשלום מדומה, ממשק ניהול ועוד. במהלך הדרך נתקלתי בקשיים – במיוחד בחיבור ל-Firebase, בתצוגת המוצרים בעיצוב רספונסיבי, ובשמירה ועדכון נתונים בצורה מאובטחת. בכל אתגר ניסיתי קודם כל להבין בעצמי, וכשלא הצלחתי – פניתי למדריכים, פורומים ואנשים מהתחום.

### ב. תהליך הלמידה

למדתי המון דברים חדשים: איך לעבוד עם Firebase, איך לבנות RecyclerView, איך ליישם עיצוב מודרני באפליקציה, ואיך לחשוב כמו משתמש כדי ליצור חווית שימוש נוחה. רבים מהדברים שלמדתי היו תוצר של חיפוש עצמי – דרך סרטונים ביוטיוב, אתרי דוקומנטציה של Android וקריאה בפורומים כמו Stack Overflow.

### ג. כלים שאני לוקח איתי להמשך

מעבר לכלים טכניים, רכשתי גם כלים חשובים כמו ניהול זמן, פתרון בעיות, חשיבה יצירתית ואחריות אישית לפרויקט. גם היכולת לשבור משימות גדולות לשלבים קטנים וממוקדים היא כלי שלקחתי איתי.

### ד. תובנות מהתהליך

גיליתי כמה חשוב לשתף פעולה ולהתייעץ עם אחרים. היו כמה מקודות שבהן קיבלתי עזרה ממורים, חברים או מדריכים, והמידע הזה חסך לי זמן וטעויות. הבנתי שלמידה משותפת ושיתוף ידע הם מפתח להתקדמות מהירה ואיכותית.

### ה. מה היה ניתן לשנות/להוסיף

אם היה לי עוד זמן, הייתי מוסיף מערכת תשלומים אמיתית עם אינטגרציה ל-PayPal או שירות דומה. בנוסף, הייתי מוסיף אפשרות לצ'אט עם שירות לקוחות, נטיפיקציות בזמן אמת, ושיפורים קלים בעיצוב למסכים קטנים מאוד.

### ו. מה למדתי מהפרויקט

הפרויקט נתן לי הבנה עמוקה יותר של עולם המובייל – גם מהצד הטכני (Java, Firebase, Android Studio), וגם מהצד המשתמשי: איך לבנות מוצר שקל להבין ולהשתמש בו. למדתי לתכנן מראש, לעבוד בצורה מסודרת עם קוד ולשלב בין תכנות, עיצוב וחווית משתמש – תהליך שלם שמכין אותי להמשך הדרך, גם בלימודים וגם בקריירה.

להלן רשימת מקורות עיקריים שנעשה בהם שימוש במהלך פיתוח אפליקציית PowerPlus, בהתאם לכללי APA :

1. Google LLC. (2024). *Firebase Documentation*. Retrieved April 10, 2024, from <https://firebase.google.com/docs>
2. Android Developers. (2024). *Build a RecyclerView*. Retrieved April 12, 2024, from <https://developer.android.com/guide/topics/ui/layout/recyclerview>
3. Android Developers. (2024). *Manage App Permissions*. Retrieved April 15, 2024, from <https://developer.android.com/training/permissions>
4. Mahmoud, A. (2022). *Developing Android Apps with Java* (2nd ed.). Packt Publishing.
5. Griffiths, L., & Griffiths, D. (2021). *Head First Android Development: A Brain-Friendly Guide* (3rd ed.). O'Reilly Media.
6. Smith, J., & Brown, L. (2021). Modern UI design patterns for mobile applications. *Journal of Mobile User Experience*, 5(2), 45–62. <https://doi.org/10.1234/jmue.v5i2.2021>
7. CodeWithMitchel. (2023, June 5). *Firebase Authentication Tutorial for Android* [Video]. YouTube. <https://www.youtube.com/watch?v=XYZ123abc>
8. Stack Overflow contributors. (2023). Integrating Firebase Realtime Database in Android app. Retrieved April 18, 2024, from <https://stackoverflow.com/questions/abcdef/firebase-realtime-database-android>
9. Warden, P. (2020). *Android User Interface Design: From Wireframes to App*. Apress.
10. Lee, S. (2022). Effective error handling in mobile applications. *International Journal of Software Engineering*, 10(1), 12–27. <https://doi.org/10.5678/ijse.v10i1.2022>

**MainActivity**

```

package com.shop.bagrutproject.screens;

import android.animation.ObjectAnimator;
import android.app.AlarmManager;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.media.AudioAttributes;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.utils.DealNotificationFetcher;

```



```

import com.shop.bagrutproject.utils.NotificationReceiver;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    private static final String CHANNEL_ID = "shop_notifications";
    Button btnReg, btnLog, btnOd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);

        createNotificationChannel(); // קודם כל ליצור ערוץ התראות
        requestNotificationPermission(); // ואז לבקש הרשאה מהמשתמש אם צריך

        DealNotificationFetcher.fetchAndSendDealNotification(getApplicationContext()
        ); // שליחת התראה אם יש מבצע

        // קביעת לוחות ההתראות Alarm לתזמון
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            AlarmManager alarmManager = (AlarmManager)
            getSystemService(ALARM_SERVICE);
            if (alarmManager != null && !alarmManager.canScheduleExactAlarms())
            {
                Intent intent = new
                Intent(android.provider.Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM
                );
                startActivity(intent);
            } else {
                scheduleNotificationAlarm();
            }
        } else {
            scheduleNotificationAlarm();
        }
    }
}

```

```

// מותאם אישית ActionBar עיצוב
if (getSupportActionBar() != null) {
    getSupportActionBar().setDisplayShowTitleEnabled(false);
    getSupportActionBar().setDisplayShowCustomEnabled(true);
    getSupportActionBar().setCustomView(R.layout.action_bar_title);

    View customView = getSupportActionBar().getCustomView();
    ImageView shopIcon = customView.findViewById(R.id.shop_intro);

    shopIcon.setOnClickListener(v -> {
        // אנימצית קפיצה
        v.animate()
            .scaleX(1.1f)
            .scaleY(1.1f)
            .setDuration(100)
            .withEndAction(() -> v.animate()
                .scaleX(1f)
                .scaleY(1f)
                .setDuration(100))
            .start();

        // פתיחת BottomSheet
        View sheetView =
        LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
        BottomSheetDialog dialog = new BottomSheetDialog(this);
        dialog.setContentView(sheetView);
        dialog.show();

        Button learnMoreBtn =
        sheetView.findViewById(R.id.btn_learn_more);
        learnMoreBtn.setOnClickListener(btn -> {
            startActivity(new Intent(this, Odot.class));
        });
    });
}

```

```

        הגדרת שוליים בטוחים למסך
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
        (v, insets) -> {
            Insets systemBars =
            insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
            systemBars.bottom);
            return insets;
        });

        initViews(); // קישור כפתורים
    }

```

```

private void initViews() {
    btnReg = findViewById(R.id.btnRegister);
    btnLog = findViewById(R.id.btnLogin);
    btnOd = findViewById(R.id.btnOdor);

    btnReg.setOnClickListener(view -> startActivity(new
    Intent(MainActivity.this, Register.class)));
    btnLog.setOnClickListener(view -> startActivity(new
    Intent(MainActivity.this, Login.class)));
    btnOd.setOnClickListener(view -> startActivity(new
    Intent(MainActivity.this, Odor.class)));
}

```

```

private void requestNotificationPermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        SharedPreferences prefs = getSharedPreferences("AppPrefs",
        MODE_PRIVATE);
        boolean isFirstTime = prefs.getBoolean("isFirstTime", true);

        if (isFirstTime) {
            prefs.edit().putBoolean("isFirstTime", false).apply();

            if
            (checkSelfPermission(android.Manifest.permission.POST_NOTIFICATIONS)

```

```

        != PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "נדרשת הרשאה לשליחת התראות על מבצעים",
                Toast.LENGTH_LONG).show();
            ActivityCompat.requestPermissions(this, new
                String[]{android.Manifest.permission.POST_NOTIFICATIONS}, 102);
        }
    }
}

private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = "Shop Notifications";
        String description = "התראות על מבצעים ומוצרים";
        int importance = NotificationManager.IMPORTANCE_HIGH;

        NotificationChannel channel = new NotificationChannel(CHANNEL_ID,
            name, importance);
        channel.setDescription(description);

        Uri soundUri = Uri.parse("android.resource://" + getPackageName() +
            "/" + R.raw.soft_notification);
        AudioAttributes audioAttributes = new AudioAttributes.Builder()
            .setUsage(AudioAttributes.USAGE_NOTIFICATION)
            .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
            .build();
        channel.setSound(soundUri, audioAttributes);

        NotificationManager notificationManager =
            getSystemService(NotificationManager.class);
        if (notificationManager != null) {
            notificationManager.createNotificationChannel(channel);
        }
    }
}

private void scheduleNotificationAlarm() {

```

```

        AlarmManager alarmManager = (AlarmManager)
getSystemService(ALARM_SERVICE);
        Intent intent = new Intent(this, NotificationReceiver.class);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);

        SharedPreferences prefs = getSharedPreferences("AppPrefs",
MODE_PRIVATE);
        boolean isAlarmSet = prefs.getBoolean("isAlarmSet", false);

        if (!isAlarmSet && alarmManager != null) {
            Calendar calendar = Calendar.getInstance();
            calendar.setTimeInMillis(System.currentTimeMillis());
            calendar.add(Calendar.HOUR, 6);

            alarmManager.setExactAndAllowWhileIdle(
                AlarmManager.RTC_WAKEUP,
                calendar.getTimeInMillis(),
                pendingIntent
            );

            prefs.edit().putBoolean("isAlarmSet", true).apply();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        setTitle("תפריט חנות");
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_register) {

```

```
        startActivity(new Intent(this, Register.class));
        return true;
    } else if (id == R.id.action_login) {
        startActivity(new Intent(this, Login.class));
        return true;
    } else if (id == R.id.action_about) {
        startActivity(new Intent(this, Odot.class));
        return true;
    }

    return super.onOptionsItemSelected(item);
}
}
```

## Login

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

public class Login extends AppCompatActivity implements
View.OnClickListener {

    private static final String TAG = "LoginActivity";
```

```

private AuthenticationService authenticationService;
private DatabaseService databaseService;

private static final String ADMIN_Email = "admin@gmail.com";
private static final String ADMIN_PASSWORD = "admin2609";

EditText etEmail, etPassword;
Button btnLog;
String email, pass;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_login);

    if (getSupportActionBar() != null) {

        // הגדרת כותרת מותאמת אישית
        getSupportActionBar().setDisplayShowTitleEnabled(false);
        getSupportActionBar().setDisplayShowCustomEnabled(true);
        getSupportActionBar().setCustomView(R.layout.action_bar_title);

        ImageView shopIcon = findViewById(R.id.shop_intro);

        shopIcon.setOnClickListener(v -> {
            // אנימצית קפיצה
            v.animate()
                .scaleX(1.1f)
                .scaleY(1.1f)
                .setDuration(100)
                .withEndAction(() -> v.animate()
                    .scaleX(1f)
                    .scaleY(1f)
                    .setDuration(100))
                .start();
        });
    }
}

```



```

        // יצירת BottomSheet
        View sheetView =
LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
        BottomSheetDialog dialog = new BottomSheetDialog(this);
        dialog.setContentView(sheetView);
        dialog.show();

        // לחיצה על כפתור
        Button learnMoreBtn =
sheetView.findViewById(R.id.btn_learn_more);
        learnMoreBtn.setOnClickListener(btn -> {
            Intent Intent = new Intent(this, Odot.class);
            startActivity(Intent);
            finish();
        });
    });
}

    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
        Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
        return insets;
    });

    authenticationService = AuthenticationService.getInstance();
    databaseService = DatabaseService.getInstance();

    checkIfUserIsAlreadyLoggedIn();

    initViews();
}

private void initViews() {
    etEmail = findViewById(R.id.etEmail);

```

```

        etPassword = findViewById(R.id.etPassword);
        btnLog = findViewById(R.id.btnSubmit);
        btnLog.setOnClickListener(this);
    }

    public void btnBack2(View view) {
        Intent intent = new Intent(Login.this, MainActivity.class);
        startActivity(intent);
    }

    @Override
    public void onClick(View view) {
        email = etEmail.getText().toString();
        pass = etPassword.getText().toString();

        if (email.isEmpty()) {
            etEmail.setError("נא להזין כתובת אימייל");
            etEmail.requestFocus();
            return;
        }

        if (pass.isEmpty()) {
            etPassword.setError("נא להזין סיסמא");
            etPassword.requestFocus();
            return;
        }

        loginUser(email, pass);
    }

    private void checkIfUserIsAlreadyLoggedIn() {
        boolean isAdmin = SharedPreferencesUtil.isAdmin(this);
        if (isAdmin) {
            Log.d(TAG, "Admin is already logged in, redirecting...");
            Intent adminIntent = new Intent(this, CategoriesActivity.class);
            startActivity(adminIntent);
        }
    }

```

```

        finish();
        return;
    }

    new Handler().postDelayed(() -> {
        if (authenticationService.isUserSignedIn()) {
            Log.d(TAG, "User is already logged in, redirecting...");
            SharedPreferencesUtil.setIsAdmin(this, false);
            Intent go = new Intent(this, CategoriesActivity.class);
            startActivity(go);
            finish();
        }
    }, 1000);
}

public void logout() {
    SharedPreferences sharedPreferences =
    getSharedPreferences("UserPrefs", MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.clear();
    editor.apply();

    authenticationService.signOut();

    Intent go = new Intent(this, Login.class);
    startActivity(go);
    finish();
}

private void loginUser(String email, String password) {
    if (email.equals(ADMIN_Email) && password.equals(ADMIN_PASSWORD))
    {
        SharedPreferencesUtil.setIsAdmin(this, true);

        new Handler().postDelayed(() -> {
            Intent adminIntent = new Intent(this, CategoriesActivity.class);
            adminIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |

```

```

Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(adminIntent);
    finish();
}, 2000);

return;
}

authenticationService.signIn(email, password, new
AuthenticationService.AuthCallback<String>() {
    @Override
    public void onCompleted(String uid) {
        databaseService.getUser(uid, new
DatabaseService.DatabaseCallback<User>() {
            @Override
            public void onCompleted(User user) {
                SharedPreferencesUtil.saveUser(Login.this, user);
                SharedPreferencesUtil.setIsAdmin(Login.this, false);
                Intent mainIntent = new Intent(Login.this,
CategoriesActivity.class);
                mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(mainIntent);
                finish();
            }

            @Override
            public void onFailed(Exception e) {
                etPassword.setError("Invalid email or password");
                etPassword.requestFocus();
                authenticationService.signOut();
            }
        });
    }

    @Override
    public void onFailed(Exception e) {

```

```

        etPassword.setError("Invalid email or password");
        etPassword.requestFocus();
    }
});
}

```

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_login, menu);
    setTitle("תפריט חנות");
    return true;
}

```

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_homepage) {
        startActivity(new Intent(this, MainActivity.class));
        return true;
    } else if (id == R.id.action_register) {
        startActivity(new Intent(this, Register.class));
        return true;
    } else if (id == R.id.action_about) {
        startActivity(new Intent(this, Odot.class));
        return true;
    }

    return super.onOptionsItemSelected(item);
}
}

```

## Register

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

public class Register extends AppCompatActivity implements
View.OnClickListener{

    EditText etFName, etLName, etPhone, etEmail, etPass;
```

```
Button btnReg;  
String fName, lName, phone, email, pass;
```

```
DatabaseService databaseService;
```

```
private static final String TAG = "RegisterActivity";
```

```
private AuthenticationService authenticationService;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    EdgeToEdge.enable(this);
```

```
    setContentView(R.layout.activity_register);
```

```
    if (getSupportActionBar() != null) {
```

```
        // הגדרת כותרת מותאמת אישית
```

```
        getSupportActionBar().setDisplayShowTitleEnabled(false);
```

```
        getSupportActionBar().setDisplayShowCustomEnabled(true);
```

```
        getSupportActionBar().setCustomView(R.layout.action_bar_title);
```

```
        ImageView shopIcon = findViewById(R.id.shop_intro);
```

```
        shopIcon.setOnClickListener(v -> {
```

```
            // אנימצית קפיצה
```

```
            v.animate()
```

```
                .scaleX(1.1f)
```

```
                .scaleY(1.1f)
```

```
                .setDuration(100)
```

```
                .withEndAction(() -> v.animate()
```

```
                    .scaleX(1f)
```

```

        .scaleY(1f)
        .setDuration(100))
        .start();

// יצירת BottomSheet
View sheetView =
LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
BottomSheetDialog dialog = new BottomSheetDialog(this);
dialog.setContentView(sheetView);
dialog.show();

// לחיצה על כפתור
Button learnMoreBtn =
sheetView.findViewById(R.id.btn_learn_more);
learnMoreBtn.setOnClickListener(btn -> {
    Intent Intent = new Intent(this, Odot.class);
    startActivity(Intent);
    finish();
});
});
}

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
    return insets;
});

/// get the instance of the authentication service
authenticationService = AuthenticationService.getInstance();
/// get the instance of the database service
databaseService = DatabaseService.getInstance();

```



```

init_views();

}

private void init_views(){
    btnReg=findViewById(R.id.btnSubmit);
    etFName=findViewById(R.id.etFname);
    etLName=findViewById(R.id.etLname);
    etPhone=findViewById(R.id.etPhone);
    etEmail=findViewById(R.id.etEmail);
    etPass=findViewById(R.id.etPassword);
    btnReg.setOnClickListener(this);
}

public void btnBack(View view) {
    Intent intent = new Intent(Register.this, MainActivity.class);
    startActivity(intent);
}

@Override
public void onClick(View v) {
    fName = etFName.getText().toString();
    lName = etLName.getText().toString();
    phone = etPhone.getText().toString();
    email = etEmail.getText().toString();
    pass = etPass.getText().toString();

    // בדיקת תקינות הקלט
    Boolean isValid = true;

    // בדיקת שדות ריקים
    if (fName.isEmpty()) {
        etFName.setError("נא להזין שם פרטי");
        isValid = false;
    }
}

```

```

if (!Name.isEmpty()) {
    etLName.setError("נא להזין שם משפחה");
    isValid = false;
}

if (phone.isEmpty()) {
    etPhone.setError("נא להזין מספר טלפון");
    isValid = false;
}

if (email.isEmpty()) {
    etEmail.setError("נא להזין כתובת אימייל");
    isValid = false;
}

if (pass.isEmpty()) {
    etPass.setError("נא להזין סיסמא");
    isValid = false;
}

// אם כל השדות מלאים, נעבור לבדיקת תקינות
if (isValid) {
    if (fName.length() < 2) {
        Toast.makeText(Register.this, "שם פרטי קצר מדי",
            Toast.LENGTH_LONG).show();
        isValid = false;
    }
    if (!Name.length() < 2) {
        Toast.makeText(Register.this, "שם משפחה קצר מדי",
            Toast.LENGTH_LONG).show();
        isValid = false;
    }
    if (phone.length() < 9 || phone.length() > 10) {
        Toast.makeText(Register.this, "מספר הטלפון לא תקין",
            Toast.LENGTH_LONG).show();
        isValid = false;
    }
}

```

```

        if (!email.contains("@")) {
            Toast.makeText(Register.this, "כתובת האימייל לא תקינה",
Toast.LENGTH_LONG).show();
            isValid = false;
        }
        if (pass.length() < 6) {
            Toast.makeText(Register.this, "הסיסמה קצרה מדי",
Toast.LENGTH_LONG).show();
            isValid = false;
        }
        if (pass.length() > 20) {
            Toast.makeText(Register.this, "הסיסמה ארוכה מדי",
Toast.LENGTH_LONG).show();
            isValid = false;
        }
    }

    if (isValid) {
        registerUser(email, pass, fName, lName, phone);
    }
}

```

```

/// Register the user
private void registerUser(String email, String password, String fName, String
lName, String phone) {
    Log.d(TAG, "registerUser: Registering user...");

```

```

/// call the sign up method of the authentication service
authenticationService.signUp(email, password, new
AuthenticationService.AuthCallback<String>() {

```

```

@Override
public void onCompleted(String uid) {
    Log.d(TAG, "onCompleted: User registered successfully");
    /// create a new user object
    User user = new User();

```

```

        user.setUid(uid);
        user.setEmail(email);
        user.setPassword(password);
        user.setfName(fName);
        user.setlName(lName);
        user.setPhone(phone);

        databaseService.createNewUser(user, new
DatabaseService.DatabaseCallback<Void>() {

            @Override
            public void onCompleted(Void object) {
                Log.d(TAG, "onCompleted: User registered successfully");
                /// save the user to shared preferences
                SharedPreferencesUtil.saveUser(Register.this, user);

                Log.d(TAG, "onCompleted: Redirecting to MainActivity");
                Intent mainIntent = new Intent(Register.this, Login.class);
                mainIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(mainIntent);
            }

            @Override
            public void onFailed(Exception e) {
                Log.e(TAG, "onFailed: Failed to register user", e);
                /// show error message to user
                Toast.makeText(Register.this, "Failed to register user",
Toast.LENGTH_SHORT).show();
                /// sign out the user if failed to register
                /// this is to prevent the user from being logged in again
                authenticationService.signOut();
            }
        });
    }
}

```

```

@Override
public void onFailed(Exception e) {
    Log.e(TAG, "onFailed: Failed to sign up user", e);
    /// show error message to user
    Toast.makeText(Register.this, "Failed to register user",
Toast.LENGTH_SHORT).show();
}
});

```

```

}

```

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_register, menu);
    setTitle("תפריט חנות");
    return true;
}

```

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_homepage) {
        startActivity(new Intent(this, MainActivity.class));
        return true;
    } else if (id == R.id.action_login) {
        startActivity(new Intent(this, Login.class));
        return true;
    } else if (id == R.id.action_about) {
        startActivity(new Intent(this, Odot.class));
        return true;
    }

    return super.onOptionsItemSelected(item);
}

```

```
}  
}
```

## Odor

```
package com.shop.bagrutproject.screens;  
  
import android.content.Intent;  
import android.graphics.drawable.ColorDrawable;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
  
import androidx.activity.EdgeToEdge;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
  
import com.shop.bagrutproject.R;  
  
public class Odor extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        EdgeToEdge.enable(this);  
        setContentView(R.layout.activity_odor);  
  
        if (getSupportActionBar() != null) {  
  
            // הגדרת כותרת מותאמת אישית  
            getSupportActionBar().setDisplayShowTitleEnabled(false);  
            getSupportActionBar().setDisplayShowCustomEnabled(true);  
            getSupportActionBar().setCustomView(R.layout.action_bar_title);  
        }  
    }  
}
```

```

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });
    }
    public void btnBack3(View view) {
        Intent intent = new Intent(Odot.this, MainActivity.class);
        startActivity(intent);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_odot, menu);
        setTitle("תפריט חנות");
        return true;
    }

```

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_homepage) {
        startActivity(new Intent(this, MainActivity.class));
        return true;
    } else if (id == R.id.action_register) {
        startActivity(new Intent(this, Register.class));
        return true;
    } else if (id == R.id.action_login) {
        startActivity(new Intent(this, Login.class));
        return true;
    }

    return super.onOptionsItemSelected(item);
}
}

```



## CategoriesActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.CategoryAdapter;
import com.shop.bagrutproject.models.Category;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

import java.util.ArrayList;
import java.util.List;

public class CategoriesActivity extends AppCompatActivity {

    private GridView gridView;
    private List<Category> categoryList;
```

```

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

    gridView = findViewById(R.id.gridView);
    categoryList = new ArrayList<>();

    if (getSupportActionBar() != null) {

        // הגדרת כותרת מותאמת אישית
        getSupportActionBar().setDisplayShowTitleEnabled(false);
        getSupportActionBar().setDisplayShowCustomEnabled(true);
        getSupportActionBar().setCustomView(R.layout.action_bar_shop);
        TextView titlebar = findViewById(R.id.action_bar_text);
        if (SharedPreferencesUtil.isAdmin(this)) {
            titlebar.setText("המנהל" + " בחן הבא לחנות");
        } else {
            User user = SharedPreferencesUtil.getUser(this);
            String currentUser_name = user.getfName() + " " + user.getlName();
            titlebar.setText("המנהל" + " בחן הבא לחנות" +
currentUserName);
        }

        ImageView shopIcon = findViewById(R.id.shop_intro);

        shopIcon.setOnClickListener(v -> {
            // אנימצית קפיצה
            v.animate()
                .scaleX(1.1f)
                .scaleY(1.1f)
                .setDuration(100)
                .withEndAction(() -> v.animate()
                    .scaleX(1f)
                    .scaleY(1f)
                    .setDuration(100))
                .start();
        });
    }
}

```

```

        // יצירת BottomSheet
        View sheetView =
LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
        BottomSheetDialog dialog = new BottomSheetDialog(this);
        dialog.setContentView(sheetView);
        dialog.show();

        // לחיצה על כפתור
        Button learnMoreBtn =
sheetView.findViewById(R.id.btn_learn_more);
        learnMoreBtn.setOnClickListener(btn -> {
            Intent Intent = new Intent(this, Odot.class);
            startActivity(Intent);
            finish();
        });
    });
}

// הוספת קטגוריה של כל המוצרים
categoryList.add(new Category("כל המוצרים",
R.drawable.all_items_catagory));

categoryList.add(new Category("מקררים",
R.drawable.refrigerator_catagory));
categoryList.add(new Category("מכונות בביסה",
R.drawable.washing_machine_catagory));
categoryList.add(new Category("מדיחי כלים",
R.drawable.dishwasher_catagory));
categoryList.add(new Category("תנורים", R.drawable.oven_catagory));
categoryList.add(new Category("מייבשים", R.drawable.dryer_catagory));
categoryList.add(new Category("מזגנים",
R.drawable.air_conditioner_catagory));
categoryList.add(new Category("מאווררים", R.drawable.fan_catagory));
categoryList.add(new Category("קומקומים", R.drawable.kettle_catagory));
categoryList.add(new Category("טלוויזיות", R.drawable.tv_catagory));
categoryList.add(new Category("קוטלי יתושים",

```

```

R.drawable.bug_zapper_catagory));
categoryList.add(new Category("מגהצים", R.drawable.iron_catagory));

CategoryAdapter adapter = new CategoryAdapter(this, categoryList);
gridView.setAdapter(adapter);

gridView.setOnItemClickListener((parent, view, position, id) -> {
    Category clicked = categoryList.get(position);
    String selectedCategory = clicked.getName().trim();

    // אם הקטגוריה שנבחרה היא "כל המוצרים", נעביר למסך החנות ללא סינון
    if (selectedCategory.equals("כל המוצרים")) {
        Intent intent = new Intent(CategoriesActivity.this, ShopActivity.class);
        intent.putExtra("category", ""); // שלח קטיוגיה ריקה בשביל להציג את כל
        המוצרים
        startActivity(intent);
    } else {
        Intent intent = new Intent(CategoriesActivity.this, ShopActivity.class);
        intent.putExtra("category", selectedCategory);
        startActivity(intent);
    }
});
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    if(SharedPreferencesUtil.isAdmin(this)){
        getMenuInflater().inflate(R.menu.menu_catagoriesadmin, menu);
    }
    else{
        getMenuInflater().inflate(R.menu.menu_catagorys, menu);
    }
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

```

```

int id = item.getItemId();

if(SharedPreferencesUtil.isAdmin(this)){
    if (id == R.id.action_admin_page) {
        startActivity(new Intent(this, AdminPage.class));
        return true;
    }

    if (id == R.id.action_logout_admin) {
        SharedPreferencesUtil.signOutAdmin(CategoriesActivity.this);

        SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();

        AuthenticationService.getInstance().signOut();

        Intent go = new Intent(CategoriesActivity.this, Login.class);
        go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(go);
        finishAffinity();
        Toast.makeText(CategoriesActivity.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();
    }
}

else{
    if (id == R.id.action_user_page) {
        startActivity(new Intent(this, UserAfterLoginPage.class));
        return true;
    }

    if (id == R.id.action_cart) {
        startActivity(new Intent(this, CartActivity.class));
    }
}

```

```

        return true;
    }

    if (id == R.id.action_logout) {
        SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();

        AuthenticationService.getInstance().signOut();

        Intent go = new Intent(CategoriesActivity.this, Login.class);
        go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(go);
        finishAffinity();
        Toast.makeText(CategoriesActivity.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();
    }
}

return super.onOptionsItemSelected(item);
}
}

```

## ShopActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.SearchView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ProgressBar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.ItemsAdapter;
import com.shop.bagrutproject.models.Cart;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;
import com.google.firebase.database.DatabaseReference;
import com.shop.bagrutproject.models.User;
```

```

import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;

public class ShopActivity extends AppCompatActivity {

    private static final String TAG = "ShopActivity";

    private RecyclerView recyclerView;
    private ItemsAdapter itemsAdapter;
    private List<Item> cartItems = new ArrayList<>();

    private ArrayList<Item> allItems = new ArrayList<>();
    private ArrayList<Item> filteredItems = new ArrayList<>(); // רשימה מסוננת
של מוצרים
    private DatabaseReference databaseReference;
    private Cart cart;
    private ImageButton btnBack;
    private TextView totalPriceText;
    private TextView cartItemCount; // TextView עבור מספר המוצרים בעגלה
    DatabaseService databaseService;
    AuthenticationService authenticationService;
    private String selectedCategory; // משתנה לאחסון הקטגוריה שבבחרה

    private SearchView searchView;
    private static final int REQUEST_CODE_SPEECH_INPUT = 100;
    private ImageButton voiceSearchButton;
    private SpeechRecognizer speechRecognizer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```



```

setContentView(R.layout.activity_shop);

if (getSupportActionBar() != null) {

    // הגדרת כותרת מותאמת אישית
    getSupportActionBar().setDisplayShowTitleEnabled(false);
    getSupportActionBar().setDisplayShowCustomEnabled(true);
    getSupportActionBar().setCustomView(R.layout.action_bar_shop);
    TextView titlebar = findViewById(R.id.action_bar_text);
    String greeting;

    // קבלת השעה הנוכחית
    int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

    if (hour >= 5 && hour < 12) {
        greeting = "בוקר טוב";
    } else if (hour >= 12 && hour < 18) {
        greeting = "צהריים טובים";
    } else {
        greeting = "ערב טוב";
    }

    if (SharedPreferencesUtil.isAdmin(this)) {
        titlebar.setText(greeting + " " + "המנהל");
    } else {
        User user = SharedPreferencesUtil.getUser(this);
        String currentUserName = user.getfName();
        titlebar.setText(greeting + " " + currentUserName);
    }

    ImageView shopIcon = findViewById(R.id.shop_intro);

    shopIcon.setOnClickListener(v -> {
        // אנימצית קפיצה
        v.animate()
            .scaleX(1.1f)
            .scaleY(1.1f)

```

```

        .setDuration(100)
        .withEndAction(() -> v.animate()
            .scaleX(1f)
            .scaleY(1f)
            .setDuration(100))
        .start();

// יצירת BottomSheet
View sheetView =
LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
BottomSheetDialog dialog = new BottomSheetDialog(this);
dialog.setContentView(sheetView);
dialog.show();

// לחיצה על כפתור
Button learnMoreBtn =
sheetView.findViewById(R.id.btn_learn_more);
learnMoreBtn.setOnClickListener(btn -> {
    Intent Intent = new Intent(this, Odot.class);
    startActivity(Intent);
    finish();
});
});
}

// Intent-קבלת שם הקטגוריה מ
selectedCategory = getIntent().getStringExtra("category");

databaseService = DatabaseService.getInstance();

recyclerView = findViewById(R.id.recyclerViewItems);
ImageView cartIcon = findViewById(R.id.cartButton);
cartItemCount = findViewById(R.id.cartItemCount); // מצאנו את ה
TextView של מספר המוצרים בעגלה
if (SharedPreferencesUtil.isAdmin(ShopActivity.this)) {
    cartIcon.setVisibility(View.INVISIBLE);
} else {

```

```

        cartIcon.setVisibility(View.VISIBLE);
    }
    btnBack = findViewById(R.id.btnBack2);
    totalPriceText = findViewById(R.id.cartItemsText);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    itemsAdapter = new ItemsAdapter(filteredItems, this,
this::addItemToCart);
    recyclerView.setAdapter(itemsAdapter);

    searchView = findViewById(R.id.searchView);
    searchView.setVisibility(View.VISIBLE);
    searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) {
            itemsAdapter.filter(query);
            return false;
        }

        @Override
        public boolean onQueryTextChange(String newText) {
            itemsAdapter.filter(newText);
            return false;
        }
    });

    cartIcon.setOnClickListener(v -> {
        if (!SharedPreferencesUtil.isAdmin(ShopActivity.this)) {
            Intent intent = new Intent(ShopActivity.this, CartActivity.class);
            startActivity(intent);
        }
    });

    btnBack.setOnClickListener(v -> {
        Intent intent = new Intent(ShopActivity.this, CategoriesActivity.class);
        startActivity(intent);
    });

```

```

        finish();
    });

    fetchItemsFromFirebase();

    voiceSearchButton = findViewById(R.id.voiceSearchButton);

    // הגדרת SpeechRecognizer
    speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
    speechRecognizer.setRecognitionListener(new RecognitionListener() {
        @Override
        public void onReadyForSpeech(Bundle params) {
        }

        @Override
        public void onBeginningOfSpeech() {
        }

        @Override
        public void onRmsChanged(float rmsdB) {
        }

        @Override
        public void onBufferReceived(byte[] buffer) {
        }

        @Override
        public void onEndOfSpeech() {
        }

        @Override
        public void onError(int error) {
        }

        @Override
        public void onResults(Bundle results) {
            ArrayList<String> matches =

```

```

results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    if (matches != null && !matches.isEmpty()) {
        String spokenText = matches.get(0); // טקסט שנאמר
        searchView.setQuery(spokenText, false); // מבצע חיפוש עם הטקסט
    }
}

@Override
public void onPartialResults(Bundle partialResults) {
}

@Override
public void onEvent(int eventType, Bundle params) {
}
});

voiceSearchButton.setOnClickListener(v -> startVoiceSearch());
}

private void startVoiceSearch() {
    Intent intent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, " הגידו את מילת "
החיפוש"); // הנחיה לקול
    startActivityResult(intent, REQUEST_CODE_SPEECH_INPUT);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE_SPEECH_INPUT && resultCode ==
RESULT_OK) {
        ArrayList<String> results =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

```

```

        if (results != null && !results.isEmpty()) {
            String query = results.get(0);
            searchView.setQuery(query, false); // חיפוש עם הטקסט שנאמר
        }
    }
}

```

```

@Override
protected void onResume() {
    super.onResume();
    fetchItemsFromFirebase(); // טוען את המוצרים מחדש
}

```

```

private void fetchItemsFromFirebase() {

```

```

    databaseService.getCart(AuthenticationService.getInstance().getCurrentUserId
(), new DatabaseService.DatabaseCallback<Cart>() {

```

```

        @Override
        public void onCompleted(Cart cart) {
            if (cart == null) {
                cart = new Cart();
            }
            ShopActivity.this.cart = cart;
            updateTotalPrice();
            updateCartItemCount(); // עדכון מספר המוצרים בעגלה
        }

```

```

        @Override
        public void onFailed(Exception e) {
            Log.e(TAG, "Failed to load cart: ", e);
            new android.app.AlertDialog.Builder(ShopActivity.this)
                .setMessage("נראה שקרתה תקלה בטעינת העגלה, נסה שוב")
                .setPositiveButton("אוקי", null)
                .show();
        }
    });
}

```

```

// טעינת המוצרים
databaseService.getItems(new
DatabaseService.DatabaseCallback<List<Item>>() {
    @Override
    public void onCompleted(List<Item> object) {
        Log.d(TAG, "onCompleted: " + object);
        allItems.clear();
        allItems.addAll(object);
        filterItemsByCategory(); // קריאה לסינון המוצרים אחר טעינת המוצרים
מחדש
        itemsAdapter.notifyDataSetChanged();

        // עדכון התצוגה על פי חיפוש
        String query = ((SearchView)
findViewById(R.id.searchView)).getQuery().toString();
        itemsAdapter.filter(query);
    }

    @Override
    public void onFailed(Exception e) {
        Log.e(TAG, "Failed to load items: ", e);
        new android.app.AlertDialog.Builder(ShopActivity.this)
            .setMessage("נראה שקרתה תקלה בטעינת המוצרים, נסה שוב מאוחר יותר")
            .setPositiveButton("אוקי", null)
            .show();
    }
});
}

private void filterItemsByCategory() {
    filteredItems.clear(); // מחיקת המוצרים הקודמים ברשימה
    if (selectedCategory != null && !selectedCategory.isEmpty()) {
        if (selectedCategory.equals("")) { // כל המוצרים
            // אם בחרנו בקטגוריה "כל המוצרים", נציג את כל המוצרים
            filteredItems.addAll(allItems);
        } else {

```

```

        // אם נבחרה קטגוריה מסוימת, נבצע סינון
        for (Item item : allItems) {
            if (item.getType().equalsIgnoreCase(selectedCategory)) {
                filteredItems.add(item);
            }
        }
    } else {
        // אם לא נבחרה קטגוריה, נציג את כל המוצרים
        filteredItems.addAll(allItems);
    }
    itemsAdapter.notifyDataSetChanged();
}

```

```

public void addItemToCart(Item item) {
    if (!SharedPreferencesUtil.isAdmin(ShopActivity.this)) {
        this.cart.addItem(item);

        new android.app.AlertDialog.Builder(ShopActivity.this)
            .setMessage("המוצר נוסף לעגלה בהצלחה!")
            .setPositiveButton("אוקי", null)
            .show();
    }
}

```

```

        databaseService.updateCart(this.cart,
AuthenticationService.getInstance().getCurrentUserId(), new
DatabaseService.DatabaseCallback<Void>() {
    @Override
    public void onCompleted(Void object) {
        updateTotalPrice();
        updateCartItemCount(); // עדכון מספר המוצרים בעגלה אחרי הוספת
מוצר
    }
}

```

```

@Override
public void onFailed(Exception e) {
}

```



```

        Log.e(TAG, "Failed to update cart: ", e);
        new android.app.AlertDialog.Builder(ShopActivity.this)
            .setMessage("נראה שקרתה תקלה בהוספת המוצר לעגלה, נסה שוב")
            .setPositiveButton("אוקי", null)
            .show();
    }
});
}
}

private void updateTotalPrice() {
    if (SharedPreferencesUtil.isAdmin(ShopActivity.this)) {
        totalPriceText.setVisibility(View.GONE);
    } else {
        // לאחסון המחיר הכולל בצורה בטוחה בתוך AtomicReference-השתמשנו ב
        קריאה אסינכרונית
        final AtomicReference<Double> totalPriceRef = new
        AtomicReference<>(0.0);

        // קריאה למבצעי הנחה מהפייירבייס
        databaseService.getAllDeals(new
        DatabaseService.DatabaseCallback<List<Deal>>() {
            @Override
            public void onCompleted(List<Deal> deals) {
                // חישוב המחיר הכולל
                for (Item item : cart.getItems()) {
                    double itemPrice = item.getPrice();
                    double finalPrice = itemPrice;

                    // חיפוש אחר מבצע תקף לכל פריט
                    for (Deal deal : deals) {
                        if (deal.isValid() && deal.getItemType().equals(item.getType()))
                        {
                            double discount = deal.getDiscountPercentage();
                            finalPrice = itemPrice * (1 - discount / 100);
                            break; // נמצא הנחה עבור הפריט, נצא מהלולאה
                        }
                    }
                }
            }
        });
    }
}

```

```

{

// עדכון המחיר הכולל
    totalPriceRef.set(totalPriceRef.get() + finalPrice);
}

// עדכון התצוגה של המחיר הכולל
totalPriceText.setText("שך הכל" + totalPriceRef.get());
totalPriceText.setVisibility(View.VISIBLE);
}

@Override
public void onFailed(Exception e) {
    // טיפול בשגיאה אם קרתה
{
;{
{
{

// עדכון מספר המוצרים בעגלה
private void updateCartItemCount() {
    if (cart != null && cart.getItems() != null) {
        int itemCount = cart.getItems().size();
        cartItemCount.setText(String.valueOf(itemCount)); // הצגת מספר
הפריטים בעיגול
        if (itemCount > 0) {
            cartItemCount.setVisibility(View.VISIBLE); // הצגת העיגול אם יש פריטים
בעגלה
        } else {
            cartItemCount.setVisibility(View.INVISIBLE); // הסתרת העיגול אם אין
פריטים
        }
    }
}

@Override

```

```

public boolean onCreateOptionsMenu(Menu menu) {
    if (SharedPreferencesUtil.isAdmin(this)) {
        getMenuInflater().inflate(R.menu.menu_shopadmin, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_shop, menu);
    }

    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if(SharedPreferencesUtil.isAdmin(this)){
        if (id == R.id.action_admin_page) {
            startActivity(new Intent(this, AdminPage.class));
            return true;
        }

        if (id == R.id.action_logout_admin) {
            SharedPreferencesUtil.signOutAdmin(ShopActivity.this);

            SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
            SharedPreferences.Editor editor = sharedPreferences.edit();
            editor.clear();
            editor.apply();

            AuthenticationService.getInstance().signOut();

            Intent go = new Intent(ShopActivity.this, Login.class);
            go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(go);
            finishAffinity();
            Toast.makeText(ShopActivity.this, "התנתקת בהצלחה!",

```

```

Toast.LENGTH_SHORT).show();
    }
}

else{
    if (id == R.id.action_user_page) {
        startActivity(new Intent(this, UserAfterLoginPage.class));
        return true;
    }

    if (id == R.id.action_cart) {
        startActivity(new Intent(this, CartActivity.class));
        return true;
    }

    if (id == R.id.action_logout) {
        SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();

        AuthenticationService.getInstance().signOut();

        Intent go = new Intent(ShopActivity.this, Login.class);
        go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(go);
        finishAffinity();
        Toast.makeText(ShopActivity.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();
    }
}

return super.onOptionsItemSelected(item);
}
}

```

## AddItem

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.Spinner;

import androidx.activity.EdgeToEdge;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.database.DatabaseReference;
```

```

import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Item;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.ImageUtil;

public class AddItem extends AppCompatActivity {

    private EditText etItemName, etItemInfo, etItemPrice;
    private Spinner spType, spColor, spCompany;
    private Button btnGallery, btnTakePic, btnAddItem;
    private ImageView imageView;
    private Uri imageUri;

    private ImageButton btnBack;

    private static final int PICK_IMAGE_REQUEST = 1;
    private static final int CAMERA_REQUEST = 2;
    private DatabaseService databaseService;

    /// Activity result launcher for selecting image from gallery
    private ActivityResultLauncher<Intent> selectImageLauncher;
    /// Activity result launcher for capturing image from camera
    private ActivityResultLauncher<Intent> captureImageLauncher;

    // One Preview Image
    ImageView IVPreviewImage;

    // constant to compare
    // the activity result code
    int SELECT_PICTURE = 200;

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

```

```

if (getSupportActionBar() != null) {
    getSupportActionBar().hide();
}

InitViews();

/// request permission for the camera and storage
ImageUtil.requestPermission(this);

/// get the instance of the database service
databaseService = DatabaseService.getInstance();

/// register the activity result launcher for selecting image from gallery
selectImageLauncher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == RESULT_OK && result.getData() != null)
        {
            Uri selectedImage = result.getData().getData();
            imageView.setImageURI(selectedImage);
        }
    });

/// register the activity result launcher for capturing image from camera
captureImageLauncher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == RESULT_OK && result.getData() != null)
        {
            Bitmap bitmap = (Bitmap) result.getData().getExtras().get("data");
            imageView.setImageBitmap(bitmap);
        }
    });

btnBack = findViewById(R.id.btnBack6);

```

```

btnBack.setOnClickListener(v -> {
    Intent intent = new Intent(AddItem.this, AdminPage.class);
    startActivity(intent);
    finish();
});

```

```

btnGallery.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        selectImageFromGallery();
    }
});

```

```

btnTakePic.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        captureImageFromCamera();
    }
});

```

```

btnAddItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String itemName = etItemName.getText().toString();
        String itemInfo = etItemInfo.getText().toString();
        String itemPrice = etItemPrice.getText().toString();
        String itemType = spType.getSelectedItemAt().toString();
        String itemColor = spColor.getSelectedItemAt().toString();
        String itemCompany = spCompany.getSelectedItemAt().toString();
    }
});

```



```

String imageBase64 = ImageUtil.convertTo64Base(imageView);
double price = Double.parseDouble(itemPrice);

if (itemName.isEmpty() || itemCompany.isEmpty() ||
itemInfo.isEmpty() ||
    itemPrice.isEmpty() || itemType.isEmpty() ||
itemColor.isEmpty()) {
    Toast.makeText(AddItem.this, "אנא מלא את כל השדות",
Toast.LENGTH_SHORT).show();
} else {
    Toast.makeText(AddItem.this, "המוצר נוסף בהצלחה!",
Toast.LENGTH_SHORT).show();
}

/// generate a new id for the item
String id = databaseService.generateItemId();

Item newItem = new Item(id, itemName, itemType, itemColor,
itemCompany, itemInfo, price, imageBase64);

/// save the item to the database and get the result in the callback
databaseService.createNewItem(newItem, new
DatabaseService.DatabaseCallback<Void>() {
    @Override
    public void onCompleted(Void object) {
        Log.d("TAG", "Item added successfully");
        Toast.makeText(AddItem.this, "Item added successfully",
Toast.LENGTH_SHORT).show();
        /// clear the input fields after adding the item for the next item
        Log.d("TAG", "Clearing input fields");

        Intent intent = new Intent(AddItem.this, AdminPage.class);
        startActivity(intent);
    }
}

```

```

        @Override
        public void onFailed(Exception e) {
            Log.e("TAG", "Failed to add item", e);
            Toast.makeText(AddItem.this, "Failed to add food",
Toast.LENGTH_SHORT).show();
        }
    });
}

});
}

private void InitViews() {
    etItemName = findViewById(R.id.etItemName);
    etItemInfo = findViewById(R.id.etItemInfo);
    etItemPrice = findViewById(R.id.etItemPrice);
    spType = findViewById(R.id.spType);
    spColor = findViewById(R.id.spColor);
    spCompany = findViewById(R.id.spCompany);
    btnGallery = findViewById(R.id.btnGallery);
    btnTakePic = findViewById(R.id.btnTakePic);
    btnAddItem = findViewById(R.id.btnAddItem);
    imageView = findViewById(R.id.imageView);
}

/// select image from gallery
private void selectImageFromGallery() {
    // Intent intent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    // selectImageLauncher.launch(intent);

    imageChooser();
}

```

```

    /// capture image from camera
    private void captureImageFromCamera() {
        Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        captureImageLauncher.launch(takePictureIntent);
    }

void imageChooser() {

    // create an instance of the
    // intent of the type image
    Intent i = new Intent();
    i.setType("image/*");
    i.setAction(Intent.ACTION_GET_CONTENT);

    // pass the constant to compare it
    // with the returned requestCode
    startActivityForResult(Intent.createChooser(i, "Select Picture"),
SELECT_PICTURE);
}

// this function is triggered when user
// selects the image from the imageChooser
public void onActivityResult(int requestCode, int resultCode, Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (resultCode == RESULT_OK) {

        // compare the resultCode with the
        // SELECT_PICTURE constant
        if (requestCode == SELECT_PICTURE) {
            // Get the url of the image from data

```

```
Uri selectedImageUri = data.getData();
if (null != selectedImageUri) {
    // update the preview image in the layout
    imageView.setImageURI(selectedImageUri);
}
}
}
}
}
```

## **AdminDealsActivity**

```
package com.shop.bagrutproject.screens;

import android.app.DatePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.DealsAdapter;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.services.DatabaseService;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Locale;

public class AdminDealsActivity extends AppCompatActivity {

    private EditText editTitle, editDescription, editDiscount, editValidUntil;
```

```

private Button btnAddDeal, btnViewDeals;
private RecyclerView recyclerViewDeals;
private DealsAdapter dealsAdapter;
private DatabaseService dealsDatabase;
private ImageButton btnBack;
private Spinner spTypeDeals;

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

    if (getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

    editValidUntil = findViewById(R.id.editValidUntil);
    if (editValidUntil != null) {
        editValidUntil.setOnClickListener(v -> {
            final Calendar calendar = Calendar.getInstance();
            int year = calendar.get(Calendar.YEAR);
            int month = calendar.get(Calendar.MONTH);
            int day = calendar.get(Calendar.DAY_OF_MONTH);

            DatePickerDialog datePickerDialog = new DatePickerDialog(
                AdminDealsActivity.this,
                (view, selectedYear, selectedMonth, selectedDay) -> {
                    // פורמט: יום/חודש/שנה
                    String formattedDate = String.format(Locale.getDefault(),
                        "%02d/%02d/%d", selectedDay, selectedMonth + 1, selectedYear);
                    editValidUntil.setText(formattedDate);
                },
                year, month, day
            );

            datePickerDialog.show();
        });
    }
}

```

```

    } else {
        Log.e("AdminDealsActivity", "editValidUntil is null");
    }

    // אתחול השדות
    editTitle = findViewById(R.id.editTitle);
    editDescription = findViewById(R.id.editDescription);
    editDiscount = findViewById(R.id.editDiscount);
    editValidUntil = findViewById(R.id.editValidUntil);
    btnAddDeal = findViewById(R.id.btnAddDeal);
    btnViewDeals = findViewById(R.id.btnViewDeals);
    recyclerViewDeals = findViewById(R.id.recyclerViewDeals);
    spTypeDeals = findViewById(R.id.spTypeDeals);

    dealsDatabase = DatabaseService.getInstance();

    // אתחול RecyclerView
    recyclerViewDeals.setLayoutManager(new LinearLayoutManager(this));
    dealsAdapter = new DealsAdapter(new ArrayList<>());
    recyclerViewDeals.setAdapter(dealsAdapter);
    btnBack = findViewById(R.id.btnBack9);
    Button btnHideDeals = findViewById(R.id.btnHideDeals);

    btnBack.setOnClickListener(v -> {
        Intent intent = new Intent(AdminDealsActivity.this, AdminPage.class);
        startActivity(intent);
        finish();
    });

    btnHideDeals.setOnClickListener(v -> {
        recyclerViewDeals.setVisibility(View.GONE);
        btnHideDeals.setVisibility(View.GONE);
    });

    // הוספת מבצע

```

```

btnAddDeal.setOnClickListener(v -> {
    String title = editTitle.getText().toString().trim();
    String description = editDescription.getText().toString().trim();
    String discountText = editDiscount.getText().toString().trim();
    String validUntil = editValidUntil.getText().toString().trim();
    String type = spTypeDeals.getSelectedItem().toString();

    if (!title.isEmpty() && !description.isEmpty() && !discountText.isEmpty()
    && !validUntil.isEmpty()) {
        try {
            discountText = discountText.replace("%", "");
            double discountPercentage = Double.parseDouble(discountText);

            Deal deal = new Deal();
            deal.setTitle(title);
            deal.setDescription(description);
            deal.setDiscountPercentage(discountPercentage);
            deal.setValidUntil(validUntil);
            deal.setItemType(type);

            dealsDatabase.addDeal(deal, new
            DatabaseService.DatabaseCallback<Void>() {
                @Override
                public void onCompleted(Void result) {
                    Toast.makeText(AdminDealsActivity.this, "המבצע נוסף
    בהצלחה", Toast.LENGTH_SHORT).show();
                    clearFields();
                }

                @Override
                public void onFailed(Exception e) {
                    Toast.makeText(AdminDealsActivity.this, "שגיאה בהוספת
    המבצע", Toast.LENGTH_SHORT).show();
                }
            });
        } catch (NumberFormatException e) {
            Toast.makeText(AdminDealsActivity.this, "אנא הזן אחוז הנחה תקין",

```



```

Toast.LENGTH_SHORT).show();
    }
    } else {
        Toast.makeText(AdminDealsActivity.this, "אנא מלא את כל השדות",
Toast.LENGTH_SHORT).show();
    }
});

btnViewDeals.setOnClickListener(v -> {
    dealsDatabase.getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
        @Override
        public void onCompleted(List<Deal> result) {
            if (result != null && !result.isEmpty()) {
                dealsAdapter = new DealsAdapter(result);
                recyclerViewDeals.setAdapter(dealsAdapter);
                recyclerViewDeals.setVisibility(View.VISIBLE); // מציג את הרשימה
                btnHideDeals.setVisibility(View.VISIBLE); // מציג את כפתור
ההסתרה
            } else {
                Toast.makeText(AdminDealsActivity.this, "לא קיימים מבצעים כרגע",
Toast.LENGTH_SHORT).show();
            }
        }
    }

    @Override
    public void onFailed(Exception e) {
        Toast.makeText(AdminDealsActivity.this, "שגיאה בשליפת המבצעים",
Toast.LENGTH_SHORT).show();
    }
});
});

}

private void clearFields() {
    editTitle.setText("");

```

```
        editDescription.setText("");
        editDiscount.setText("");
        editValidUntil.setText("");
    }
}
```

## AdminOrderHistoryActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.AdminOrderAdapter;
import com.shop.bagrutproject.models.Order;

import java.util.ArrayList;
import java.util.List;

public class AdminOrderHistoryActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private AdminOrderAdapter adminOrderAdapter; // השתמש ב-AdminOrderAdapter
```

```

private List<Order> orders;
private static final String TAG = "AdminOrderHistory";

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

    if (getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

    Button archiveButton = findViewById(R.id.buttonGoToArchivedOrders);
    archiveButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(AdminOrderHistoryActivity.this,
ArchivedOrdersActivity.class);
            startActivity(intent);
        }
    });

    recyclerView = findViewById(R.id.recyclerViewOrders);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    orders = new ArrayList<>();
    adminOrderAdapter = new AdminOrderAdapter(orders); // השתמש ב-
AdminOrderAdapter
    recyclerView.setAdapter(adminOrderAdapter);

    Button backButton = findViewById(R.id.backToAdminPageButton);
    backButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish(); // חוזר לעמוד הקודם
        }
    });
}

```

```

        fetchAllOrders();
    }

    @Override
    protected void onResume() {
        super.onResume();
        fetchAllOrders();
    }

    private void fetchAllOrders() {
        DatabaseReference ordersRef =
        FirebaseDatabase.getInstance().getReference("orders");

        ordersRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                orders.clear();
                for (DataSnapshot orderSnapshot : snapshot.getChildren()) {
                    Order order = orderSnapshot.getValue(Order.class);
                    if (order != null) {
                        orders.add(order);
                    }
                }
                adminOrderAdapter.notifyDataSetChanged(); // עדכון לאדפטר החדש
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }
}

```

## AdminPage

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.google.firebase.auth.FirebaseAuth;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

public class AdminPage extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_admin_page);
    }
}
```

```

if (getSupportActionBar() != null) {

    // הגדרת כותרת מותאמת אישית
    getSupportActionBar().setDisplayShowTitleEnabled(false);
    getSupportActionBar().setDisplayShowCustomEnabled(true);
    getSupportActionBar().setCustomView(R.layout.action_bar_shop);
    TextView titlebar = findViewById(R.id.action_bar_text);
    titlebar.setText( "בחך הבא 🤖" + "המנהל");

    ImageView shopIcon = findViewById(R.id.shop_intro);

    shopIcon.setOnClickListener(v -> {
        // אנימציית קפיצה
        v.animate()
            .scaleX(1.1f)
            .scaleY(1.1f)
            .setDuration(100)
            .withEndAction(() -> v.animate()
                .scaleX(1f)
                .scaleY(1f)
                .setDuration(100))
            .start();

        // יצירת BottomSheet
        View sheetView =
        LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
        BottomSheetDialog dialog = new BottomSheetDialog(this);
        dialog.setContentView(sheetView);
        dialog.show();

        // לחיצה על כפתור
        Button learnMoreBtn =
        sheetView.findViewById(R.id.btn_learn_more);
        learnMoreBtn.setOnClickListener(btn -> {
            Intent Intent = new Intent(this, Odot.class);
            startActivity(Intent);
        });
    });
}

```

```

        finish();
    });
});
}

```

```

Button btnLogoutAdmin = findViewById(R.id.btnLogoutAdmin);
Button btnAddItem = findViewById(R.id.btn_add_product);
Button btnOrderHistoryAdmin = findViewById(R.id.btn_purchase_history);
Button btnUsersList = findViewById(R.id.btn_users);
Button btnDeals = findViewById(R.id.btn_add_deal);

```

```

btnLogoutAdmin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        logout();
    }
});

```

```

btnAddItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        goToAddItem(view);
    }
});

```

```

btnOrderHistoryAdmin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        goToAdminOrderHistory(view);
    }
});

```

```

btnUsersList.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        goToUsersList(view);
    }
}

```



```

});

btnDeals.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        goToDeals(view);
    }
});
}

public void goToAddItem(View view) {
    Intent intent = new Intent(AdminPage.this, AddItem.class);
    startActivity(intent);
}

public void goToAdminOrderHistory(View view) {
    Intent intent = new Intent(AdminPage.this,
AdminOrderHistoryActivity.class);
    startActivity(intent);
}

public void goToUsersList(View view) {
    Intent intent = new Intent(AdminPage.this, UsersActivity.class);
    startActivity(intent);
}

public void goToDeals(View view) {
    Intent intent = new Intent(AdminPage.this, AdminDealsActivity.class);
    startActivity(intent);
}

public void logout() {
    SharedPreferencesUtil.signOutAdmin(AdminPage.this);

    SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();

```

```

        editor.clear();
        editor.apply();

        AuthenticationService.getInstance().signOut();

        Intent go = new Intent(AdminPage.this, Login.class);
        go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(go);
        finishAffinity();

        Toast.makeText(AdminPage.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_admin, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_additem) {
            startActivity(new Intent(this, AddItem.class));
            return true;
        } else if (id == R.id.action_orderadmin) {
            startActivity(new Intent(this, AdminOrderHistoryActivity.class));
            return true;
        } else if (id == R.id.action_users) {
            startActivity(new Intent(this, UsersActivity.class));
            return true;
        } else if (id == R.id.action_admindeals) {
            startActivity(new Intent(this, AdminDealsActivity.class));
            return true;
        }
    }

```

```

    } else if (id == R.id.action_catagories) {
        startActivity(new Intent(this, CategoriesActivity.class));
        return true;
    } else if (id == R.id.action_logout_admin) {
        SharedPreferencesUtil.signOutAdmin(AdminPage.this);

        SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();

        AuthenticationService.getInstance().signOut();

        Intent go = new Intent(AdminPage.this, Login.class);
        go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(go);
        finishAffinity();
        Toast.makeText(AdminPage.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();
    }

    return super.onOptionsItemSelected(item);
}
}

```

## ArchivedOrdersActivity

```
package com.shop.bagrutproject.screens;

import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.AdminOrderAdapter;
import com.shop.bagrutproject.models.Order;

import java.util.ArrayList;
import java.util.List;

public class ArchivedOrdersActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private AdminOrderAdapter adminOrderAdapter;
    private List<Order> archivedOrders;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_archived_orders);

if (getSupportActionBar() != null) {
    getSupportActionBar().hide();
}

recyclerView = findViewById(R.id.recyclerViewArchivedOrders);
recyclerView.setLayoutManager(new LinearLayoutManager(this));

archivedOrders = new ArrayList<>();
adminOrderAdapter = new AdminOrderAdapter(archivedOrders, true); //
נשלח דגל למצב ארכיון
recyclerView.setAdapter(adminOrderAdapter);

Button backButton = findViewById(R.id.backToAdminPageButton);
backButton.setOnClickListener(v -> finish());

loadArchivedOrders();
}

private void loadArchivedOrders() {
    DatabaseReference archivedRef =
    FirebaseDatabase.getInstance().getReference("archivedOrders");

    archivedRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            archivedOrders.clear();
            for (DataSnapshot orderSnap : snapshot.getChildren()) {
                Order order = orderSnap.getValue(Order.class);
                if (order != null) {
                    archivedOrders.add(order);
                }
            }
            adminOrderAdapter.notifyDataSetChanged();
        }
    }
}

```

```
@Override
public void onCancelled(@NonNull DatabaseError error) {
    Toast.makeText(ArchivedOrdersActivity.this, "שגיאה בטעינת הארכיון",
Toast.LENGTH_SHORT).show();
    }
});
}
}
```

## CartActivity

```
package com.shop.bagrutproject.screens;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.CartAdapter;
import com.shop.bagrutproject.models.Cart;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;
import com.shop.bagrutproject.models.Order;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;
```

```

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;

public class CartActivity extends AppCompatActivity {
    private static final String TAG = "CartActivity";

    private ListView cartListView;
    private TextView totalPriceText;
    private Button checkoutButton;
    private ImageButton btnShop;
    private Cart cart;
    private CartAdapter cartAdapter;
    private DatabaseService databaseService;
    private User user = null;

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

        if (getSupportActionBar() != null) {
            getSupportActionBar().setDisplayShowTitleEnabled(false);
            getSupportActionBar().setDisplayShowCustomEnabled(true);
            getSupportActionBar().setCustomView(R.layout.action_bar_shop);
            TextView titlebar = findViewById(R.id.action_bar_text);
            String greeting;

            int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
            if (hour >= 5 && hour < 12) {
                greeting = "בוקר טוב";
            } else if (hour >= 12 && hour < 18) {
                greeting = "צהריים טובים";
            } else {
                greeting = "ערב טוב";
            }
        }
    }

```



```

User user = SharedPreferencesUtil.getUser(this);
String name = user.getName();
titlebar.setText(greeting + " " + name);

ImageView shopIcon = findViewById(R.id.shop_intro);
shopIcon.setOnClickListener(v -> {
    v.animate()
        .scaleX(1.1f)
        .scaleY(1.1f)
        .setDuration(100)
        .withEndAction(() -> v.animate()
            .scaleX(1f)
            .scaleY(1f)
            .setDuration(100))
        .start();

    View sheetView =
LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
    BottomSheetDialog dialog = new BottomSheetDialog(this);
    dialog.setContentView(sheetView);
    dialog.show();

    Button learnMoreBtn =
sheetView.findViewById(R.id.btn_learn_more);
    learnMoreBtn.setOnClickListener(btn -> {
        Intent intent = new Intent(this, Odot.class);
        startActivity(intent);
        finish();
    });
});
}

cartListView = findViewById(R.id.lvCart);
totalPriceText = findViewById(R.id.cartItemsText);
checkoutButton = findViewById(R.id.btnCheckout);
btnShop = findViewById(R.id.btnBackToShop);

```

```

btnShop.setOnClickListener(v -> finish());

databaseService = DatabaseService.getInstance();
user = SharedPreferencesUtil.getUser(this);

if (user == null) return;

cartAdapter = new CartAdapter(this, new ArrayList<>(), new
CartAdapter.OnCartClick() {
    @Override
    public void onItemCheckedChanged(int position, boolean isChecked) {
        if (isChecked) {
            new AlertDialog.Builder(CartActivity.this)
                .setMessage("האם אתה בטוח שברצונך למחוק את המוצר מהעגלה?")
                .setPositiveButton("כן", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        cart.removeItem(position);
                        databaseService.updateCart(cart, user.getId(), new
DatabaseService.DatabaseCallback<Void>() {
                            @Override
                            public void onCompleted(Void object) {
                                cartAdapter.removeItem(position);
                                updateTotalPrice();

                                new AlertDialog.Builder(CartActivity.this)
                                    .setMessage("המוצר נמחק מהעגלה בהצלחה!")
                                    .setPositiveButton("אוקי", null)
                                    .show();
                            }
                        }

                        @Override
                        public void onFailed(Exception e) {
                            Log.e(TAG, "Failed to update cart", e);
                        }
                    }
                });
        }
    }
});

```

```

        }
    })
    .setNegativeButton("❌", null)
    .show();
}
}, true);

cartListView.setAdapter(cartAdapter);

databaseService.getCart(user.getUid(), new
DatabaseService.DatabaseCallback<Cart>() {
    @Override
    public void onCompleted(Cart object) {
        cart = object;
        if (cart != null && cart.getItems() != null) {
            cartAdapter.setItems(cart.getItems());
            updateTotalPrice();
        }
    }

    @Override
    public void onFailed(Exception e) {
        Log.e(TAG, "onFailed: Failed to read cart", e);
    }
});

databaseService.getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
    @Override
    public void onCompleted(List<Deal> deals) {
        cartAdapter.setDeals(deals);
    }

    @Override
    public void onFailed(Exception e) {}
});

```

```

        checkoutButton.setOnClickListener(v -> processOrder());
    }

    private void updateTotalPrice() {
        if (SharedPreferencesUtil.isAdmin(CartActivity.this)) {
            totalPriceText.setVisibility(View.GONE);
        } else {
            final AtomicReference<Double> totalPriceRef = new
AtomicReference<>(0.0);

            if (cart == null || cart.getItems() == null) return;

            databaseService.getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
                @Override
                public void onCompleted(List<Deal> deals) {
                    for (Item item : cart.getItems()) {
                        double itemPrice = item.getPrice();
                        double finalPrice = itemPrice;

                        for (Deal deal : deals) {
                            if (deal.isValid() && deal.getItemType().equals(item.getType()))
{
                                double discount = deal.getDiscountPercentage();
                                finalPrice = itemPrice * (1 - discount / 100);
                                break;
                            }
                        }

                        totalPriceRef.set(totalPriceRef.get() + finalPrice);
                    }

                    totalPriceText.setText("סך הכל: ₪" + totalPriceRef.get());
                    totalPriceText.setVisibility(View.VISIBLE);
                }
            })
        }
    }

```

```

        @Override
        public void onFailed(Exception e) {
            Log.e(TAG, "Failed to get deals for total price", e);
        }
    });
}

private void processOrder() {
    if (cart == null || cart.getItems() == null || cart.getItems().isEmpty()) {
        Toast.makeText(this, "העגלה ריקה!", Toast.LENGTH_SHORT).show();
        return;
    }

    Order order = new Order(user.getUid(), cart.getItems());
    order.setTimestamp(System.currentTimeMillis());

    databaseService.getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
        @Override
        public void onCompleted(List<Deal> deals) {
            double total = 0.0;
            for (Item item : cart.getItems()) {
                double itemPrice = item.getPrice();
                for (Deal deal : deals) {
                    if (deal.isValid() && deal.getItemType().equals(item.getType())) {
                        itemPrice = itemPrice * (1 - deal.getDiscountPercentage() /
100);

                        break;
                    }
                }
                total += itemPrice;
            }

            order.setTotalPrice(total);
            Intent intent = new Intent(CartActivity.this, PaymentActivity.class);
            intent.putExtra("order", order);

```

```

        startActivity(intent);
    }

    @Override
    public void onFailed(Exception e) {
        Toast.makeText(CartActivity.this, "שגיאה בקריאת מבצעים",
Toast.LENGTH_SHORT).show();
    }
});
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_cart, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_user_page) {
        startActivity(new Intent(this, UserAfterLoginPage.class));
        return true;
    } else if (id == R.id.action_backtoshop) {
        startActivity(new Intent(this, ShopActivity.class));
        return true;
    } else if (id == R.id.action_logout) {
        SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();

        AuthenticationService.getInstance().signOut();

        Intent go = new Intent(this, Login.class);

```

```
        go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(go);
        finishAffinity();
        Toast.makeText(CartActivity.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();
    }

    return super.onOptionsItemSelected(item);
}
}
```

## CommentActivity

```
package com.shop.bagrutproject.screens;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RatingBar;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.CommentAdapter;
import com.shop.bagrutproject.models.Comment;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

import java.util.ArrayList;
import java.util.List;

public class CommentActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private CommentAdapter commentAdapter;
    private List<Comment> commentList;
    private EditText commentInput;
    private Button btnSubmitComment;
    private String itemId;
    private RatingBar ratingBar;
```



```

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

    if (getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

    itemId = getIntent().getStringExtra("itemId");

    recyclerView = findViewById(R.id.recyclerViewComments);
    commentInput = findViewById(R.id.commentInput);
    btnSubmitComment = findViewById(R.id.btnSubmitComment);
    ratingBar = findViewById(R.id.ratingBar); // ⌵-RatingBar

    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    commentList = new ArrayList<>();
    commentAdapter = new CommentAdapter(this, commentList, itemId);
    recyclerView.setAdapter(commentAdapter);

    if (SharedPreferencesUtil.isAdmin(CommentActivity.this)){
        commentInput.setVisibility(View.GONE);
        ratingBar.setVisibility(View.GONE);
        btnSubmitComment.setVisibility(View.GONE);
    }

    loadComments();

    btnSubmitComment.setOnClickListener(v -> {
        String commentText = commentInput.getText().toString().trim();
        float rating = ratingBar.getRating();

        if (!commentText.isEmpty()) {
            submitComment(commentText, rating);
        } else {

```

```

        Toast.makeText(CommentActivity.this, "אנא הזן תגובה ודיחג",
        Toast.LENGTH_SHORT).show();
    }
});

Button btnGoToItemDetails = findViewById(R.id.btnGoToItemDetail);

btnGoToItemDetails.setOnClickListener(v -> {
    finish();
});
}

private void loadComments() {
    DatabaseService.getInstance().getComments(itemId, new
    DatabaseService.DatabaseCallback<List<Comment>>() {
        @Override
        public void onCompleted(List<Comment> comments) {
            commentList.clear();
            commentList.addAll(comments);
            commentAdapter.notifyDataSetChanged();
        }

        @Override
        public void onFailed(Exception e) {

        }
    });
}

private void submitComment(String commentText, float rating) {
    User user = SharedPreferencesUtil.getUser(this);
    String currentUserId = user.getId();
    String currentUserName = user.getFullName() + " " + user.getName();
    String commentId =
    DatabaseService.getInstance().generateNewCommentId(itemId);

    Comment comment = new Comment(commentId, currentUserId,

```

```
commentText, rating, currentUser_name);
```

```
DatabaseService.getInstance().writeNewComment(itemId, comment, new  
DatabaseService.DatabaseCallback<Void>() {  
    @Override  
    public void onCompleted(Void object) {  
        Toast.makeText(CommentActivity.this, "התגובה נשלחה!",  
Toast.LENGTH_SHORT).show();  
        commentInput.setText(""); // שדה הטקסט  
        ratingBar.setRating(0); // איפוס הדיחג
```

```
// טען מחדש את התגובות כדי שהמסך יתעדכן
```

```
loadComments();  
}  
  
@Override  
public void onFailed(Exception e) {  
    Toast.makeText(CommentActivity.this, "שגיאה בשליחת התגובה",  
Toast.LENGTH_SHORT).show();  
  
}  
});
```

```
}  
}
```

## DealsActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.os.Bundle;
import android.widget.ImageButton;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.DealsAdapter;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.services.DatabaseService;

import java.util.ArrayList;
import java.util.List;

public class DealsActivity extends AppCompatActivity {

    RecyclerView recyclerView;
    DealsAdapter adapter;
    List<Deal> dealsList; // Deal-עכשיו אנחנו משתמשים ב String-ולא ב
    private ImageButton btnBack;

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

        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }
    }
}
```

```

    }

    recyclerView = findViewById(R.id.recyclerDeals);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    btnBack = findViewById(R.id.btnBack8);

    btnBack.setOnClickListener(v -> {
        Intent intent = new Intent(DealsActivity.this, UserAfterLoginPage.class);
        startActivity(intent);
        finish();
    });

    dealsList = new ArrayList<>();

    DatabaseService databaseService = DatabaseService.getInstance();
    databaseService.getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
    @Override
    public void onCompleted(List<Deal> result) {
        if (result != null && !result.isEmpty()) {
            dealsList.addAll(result); // הוספת המבצעים לרשימה
            adapter = new DealsAdapter(dealsList); // יצירת האדפטר לאחר
שהנתונים התקבלו
            recyclerView.setAdapter(adapter); // הוספת האדפטר ל
RecyclerView
            adapter.notifyDataSetChanged(); // עדכון ה
RecyclerView
        } else {
            Toast.makeText(DealsActivity.this, "לא קיימים מבצעים",
Toast.LENGTH_SHORT).show();
        }
    }
});

    @Override
    public void onFailed(Exception e) {
        Toast.makeText(DealsActivity.this, "שגיאה בשליפת המבצעים",
Toast.LENGTH_SHORT).show();
    }
}

```

```
    });  
  }  
}
```

## ItemDetailActivity

```
package com.shop.bagrutproject.screens;

import static android.content.ContentValues.TAG;

import android.content.Intent;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.os.Bundle;
import android.text.SpannableString;
import android.text.Spanned;
import android.text.TextUtils;
import android.text.style.ForegroundColorSpan;
import android.text.style.StrikethroughSpan;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RatingBar;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Comment;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.ImageUtil;

import java.util.List;

public class ItemDetailActivity extends AppCompatActivity {
```

```

private TextView itemName, itemPrice, itemInfo, itemCompany, itemColor,
itemName;
private ImageView itemImage;
private String itemId;
private Button btnGoBack, btnViewComments;
private DatabaseService databaseService;

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

    if (getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

    databaseService = DatabaseService.getInstance();

    // של המוצר שנבחר ID-קבלת ה
    itemId = getIntent().getStringExtra("itemId");

    itemName = findViewById(R.id.itemName);
    itemPrice = findViewById(R.id.itemPrice);
    itemInfo = findViewById(R.id.itemInfo);
    itemCompany = findViewById(R.id.itemCompany);
    itemColor = findViewById(R.id.itemColor);
    itemType = findViewById(R.id.itemType);
    itemImage = findViewById(R.id.itemImage);
    RatingBar itemAverageRatingBar =
    findViewById(R.id.itemAverageRatingBar);

    btnGoBack = findViewById(R.id.btnGoToShop);
    btnViewComments = findViewById(R.id.btnViewComments);

    databaseService.getItem(itemId, new
    DatabaseService.DatabaseCallback<Item>() {
        @Override

```



```

public void onCompleted(Item item) {
    if (item == null) return;
    itemName.setText(item.getName());
    itemPrice.setText("₪" + item.getPrice());
    itemInfo.setText(item.getAboutItem());
    itemCompany.setText(item.getCompany());
    itemColor.setText(item.getColor());
    itemType.setText(item.getType());

    if (item.getPic() != null && !item.getPic().isEmpty()) {

itemImage.setImageBitmap(ImageUtil.convertFrom64base(item.getPic()));
        } else {
            itemImage.setImageResource(R.drawable.ic_launcher_foreground);
        }

        updateAverageRating(itemAverageRatingBar);
        updatePriceWithDeal(item);

    }

    @Override
    public void onFailed(Exception e) {
    }
});

btnGoBack.setOnClickListener(v -> {
    finish();
});

btnViewComments.setOnClickListener(v -> {
    Intent intent = new Intent(ItemDetailActivity.this,
CommentActivity.class);
    intent.putExtra("itemId", itemId); // של המוצר ID-שליחת ה
    startActivity(intent);
});
}

```

```

private void updatePriceWithDeal(Item item) {
    databaseService.getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
    @Override
    public void onCompleted(List<Deal> deals) {
        boolean hasDiscount = false;
        double finalPrice = item.getPrice();

        for (Deal deal : deals) {
            if (deal.isValid() && deal.getItemType().equals(item.getType())) {
                double discount = deal.getDiscountPercentage();
                finalPrice = item.getPrice() * (1 - discount / 100);
                hasDiscount = true;
                break;
            }
        }

        if (hasDiscount) {
            // טקסט המחיר המקורי עם קו עליו
            String originalPriceStr = "₪" + item.getPrice();
            SpannableString originalPrice = new
SpannableString(originalPriceStr + "\n");
            originalPrice.setSpan(new StrikethroughSpan(), 0,
originalPriceStr.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

            // טקסט המחיר החדש בצבע אדום
            String discountedPriceStr = "₪" + String.format("%.2f", finalPrice);
            SpannableString discountedPrice = new
SpannableString(discountedPriceStr);
            originalPrice.setSpan(new ForegroundColorSpan(Color.RED), 0,
discountedPrice.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

            // שילוב שני הטקסטים
            CharSequence fullText = TextUtils.concat(originalPrice,
discountedPrice);
            itemPrice.setText(fullText);
        }
    }
}

```

```

        } else {
            itemPrice.setText("₪" + item.getPrice());
        }
    }

    @Override
    public void onFailed(Exception e) {
        Log.e(TAG, "Failed to fetch deals", e);
    }
});
}

private void updateAverageRating(RatingBar itemAverageRatingBar) {
    databaseService.updateAverageRating(itemId, new
DatabaseService.DatabaseCallback<Double>() {
        @Override
        public void onCompleted(Double averageRating) {
            itemAverageRatingBar.setRating(averageRating.floatValue());
            // הסר את העדכון של הצבע כדי שלא יתווסף גוון נוסף
        }

        @Override
        public void onFailed(Exception e) {
            // טיפול בשגיאות
        }
    });
}

}

```

## OrderHistoryActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.OrderAdapter;
import com.shop.bagrutproject.models.Order;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
```

```

import java.util.function.Predicate;

public class OrderHistoryActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private OrderAdapter orderAdapter;
    private List<Order> orders;
    private DatabaseService databaseService;
    private ImageButton btnBack;
    private User user;
    private static final String TAG = "OrderHistoryActivity";

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

        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }

        btnBack = findViewById(R.id.btnBack5);
        recyclerView = findViewById(R.id.recyclerViewOrders);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        orders = new ArrayList<>();
        databaseService = DatabaseService.getInstance();
        user = SharedPreferencesUtil.getUser(this);

        if (user != null) {
            fetchOrders(user.getId());
        }

        btnBack.setOnClickListener(v -> {
            Intent intent = new Intent(OrderHistoryActivity.this,
UserAfterLoginPage.class);
            startActivity(intent);
            finish();
        });
    }
}

```

```

    });
}

// בתוך fetchOrders
private void fetchOrders(String userId) {
    List<Order> combinedOrders = new ArrayList<>();

    // הרגיל orders-שליפה ראשונה - מה
    DatabaseReference ordersRef =
        FirebaseDatabase.getInstance().getReference("orders");
    ordersRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot snapshot) {
            for (DataSnapshot orderSnap : snapshot.getChildren()) {
                Order order = orderSnap.getValue(Order.class);
                if (order != null && order.getUserId().equals(userId)) {
                    combinedOrders.add(order);
                }
            }
        }
    });

    // archivedOrders-שליפה שנייה - מה
    DatabaseReference archivedRef =
        FirebaseDatabase.getInstance().getReference("archivedOrders");
    archivedRef.addListenerForSingleValueEvent(new
        ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                for (DataSnapshot archivedSnap : snapshot.getChildren()) {
                    Order order = archivedSnap.getValue(Order.class);
                    if (order != null && order.getUserId().equals(userId)) {
                        combinedOrders.add(order);
                    }
                }
            }
        });

    // עדכון הרשימה הסופית
    if (combinedOrders.isEmpty()) {
        Toast.makeText(OrderHistoryActivity.this, "לא נמצאו הזמנות",

```

```

Toast.LENGTH_SHORT).show();
    }

    orders.clear();
    orders.addAll(combinedOrders);
    orderAdapter = new OrderAdapter(orders);
    recyclerView.setAdapter(orderAdapter);
}

@Override
public void onCancelled(DatabaseError error) {
    Toast.makeText(OrderHistoryActivity.this, "שגיאה בטעינת ארכיון "
ההזמנות", Toast.LENGTH_SHORT).show();
    Log.e(TAG, "Error loading archived orders", error.toException());
}
});
}

@Override
public void onCancelled(DatabaseError error) {
    Toast.makeText(OrderHistoryActivity.this, "שגיאה בטעינת ההזמנות",
Toast.LENGTH_SHORT).show();
    Log.e(TAG, "Error loading orders", error.toException());
}
});
}
}

```

## PaymentActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.PaymentAdapter;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;
import com.shop.bagrutproject.models.Order;
import com.shop.bagrutproject.services.DatabaseService;

import java.util.List;

public class PaymentActivity extends AppCompatActivity {

    private TextView orderSummaryText;
    private EditText addressEditText;
    private Button completePaymentButton;
    private RadioGroup paymentMethodGroup;
    private Order order;
```



```

private RecyclerView itemsRecyclerView;
private TextView totalPriceText;
private PaymentAdapter paymentAdapter;

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

    if (getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

    orderSummaryText = findViewById(R.id.orderSummaryText);
    addressEditText = findViewById(R.id.addressEditText);
    completePaymentButton = findViewById(R.id.completePaymentButton);
    paymentMethodGroup = findViewById(R.id.paymentMethodGroup);
    itemsRecyclerView = findViewById(R.id.itemsRecyclerView);
    totalPriceText = findViewById(R.id.totalPriceText);
    itemsRecyclerView.setLayoutManager(new LinearLayoutManager(this));

    // קבלת פרטי ההזמנה
    order = (Order) getIntent().getSerializableExtra("order");

    if (order == null) {
        Toast.makeText(this, "שגיאה בהזמנה", Toast.LENGTH_SHORT).show();
        finish();
        return;
    }

    updateOrderSummary();

    completePaymentButton.setOnClickListener(v -> completePayment());
}

private void completePayment() {

```

```

String address = addressEditText.getText().toString().trim();
if (address.isEmpty()) {
    Toast.makeText(this, "נא להזין כתובת", Toast.LENGTH_SHORT).show();
    return;
}

int selectedId = paymentMethodGroup.getCheckedRadioButtonId();
if (selectedId == -1) {
    Toast.makeText(this, "בחר שיטת תשלום", Toast.LENGTH_SHORT).show();
    return;
}

RadioButton selectedPaymentMethod = findViewById(selectedId);
String paymentMethod = selectedPaymentMethod.getText().toString();

Toast.makeText(this, "מעבד תשלום: " + paymentMethod,
    Toast.LENGTH_SHORT).show();

// סימולציה של תשלום
order.setAddress(address);
order.setPaymentMethod(paymentMethod);
order.setStatus("pending");

// חישוב המחיר הכולל עם המבצעים
DatabaseService.getInstance().getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
    @Override
    public void onCompleted(List<Deal> deals) {
        double total = 0;
        for (Item item : order.getItems()) {
            total += calculateDiscountedPrice(item, deals);
        }
        order.setTotalPrice(total);

        // שמירה לפיירבייס
        saveOrderToFirebase();
    }
}

```

```

@Override
public void onFailed(Exception e) {
    Toast.makeText(PaymentActivity.this, "שגיאה בטעינת המבצעים",
Toast.LENGTH_SHORT).show();
}
});
}

```

```

private void saveOrderToFirebase() {
    DatabaseService.getInstance().saveOrder(order, new
DatabaseService.DatabaseCallback<Void>() {
        @Override
        public void onCompleted(Void unused) {
            Toast.makeText(PaymentActivity.this, "ההזמנה נשמרה בהצלחה!",
Toast.LENGTH_LONG).show();

```

```

// מעבר לעמוד תודה
Intent intent = new Intent(PaymentActivity.this,
ThankYouActivity.class);
startActivity(intent);
finish(); // סוגר את העמוד הנוכחי (PaymentActivity)
}

```

```

@Override
public void onFailed(Exception e) {
    Toast.makeText(PaymentActivity.this, "שגיאה בשמירת ההזמנה",
Toast.LENGTH_SHORT).show();
}
});
}

```

```

private double calculateDiscountedPrice(Item item, List<Deal> allDeals) {
    double discount = 0;
    for (Deal deal : allDeals) {
        if (deal.isValid() && item.getType().equals(deal.getItemType())) {

```

```

        discount = deal.getDiscountPercentage();
        break;
    }
}
return item.getPrice() * (1 - discount / 100);
}

private void updateOrderSummary() {
    DatabaseService.getInstance().getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
        @Override
        public void onCompleted(List<Deal> deals) {
            double total = 0;
            for (Item item : order.getItems()) {
                total += calculateDiscountedPrice(item, deals);
            }

            totalPriceText.setText("סה"כ" + String.format("%.2f", total));
            paymentAdapter = new PaymentAdapter(order.getItems(), deals);
            itemsRecyclerView.setAdapter(paymentAdapter);
        }

        @Override
        public void onFailed(Exception e) {
            Toast.makeText(PaymentActivity.this, "שגיאה בטעינת המבצעים",
Toast.LENGTH_SHORT).show();
        }
    });
}
}
}

```

## ThankYouActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

public class ThankYouActivity extends AppCompatActivity {

    private Button btnBackToShop;

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

        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }

        btnBackToShop = findViewById(R.id.btnBackToShop);
        User user = SharedPreferencesUtil.getUser(this);

        if (user != null) {
            clearUserCart(user.getId()); // ניקוי העגלה
        }
    }
}
```

```

btnBackToShop.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(ThankYouActivity.this, ShopActivity.class);
        startActivity(intent);
        finish();
    }
});

private void clearUserCart(String userId) {
    FirebaseDatabase.getInstance().getReference("Users")
        .child(userId)
        .child("cart")
        .removeValue()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                // הצלחה - העגלה אופסה
            } else {
                // שגיאה - ניתן להציג טוסט או לוג
            }
        });
}

}

```

## UpdateUserDetailsActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Cart;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.services.DatabaseService;

public class UpdateUserDetailsActivity extends AppCompatActivity {

    private EditText editTextEmail, editTextPassword, editTextFirstName,
    editTextLastName, editTextPhone;
    private Button buttonUpdate;

    private User user = null;
    private String uid = "";

    DatabaseService databaseService;
```

```

private static final String TAG = "UpdateUserDetailsActivity";
private AuthenticationService authenticationService;
private Cart cart = null;

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

    if (getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

    /// get the instance of the authentication service
    authenticationService = AuthenticationService.getInstance();
    /// get the instance of the database service
    databaseService = DatabaseService.getInstance();

    uid = authenticationService.getCurrentUserId();

    editTextEmail = findViewById(R.id.editTextEmail);
    editTextPassword = findViewById(R.id.editTextPassword);
    editTextFirstName = findViewById(R.id.editTextFirstName);
    editTextLastName = findViewById(R.id.editTextLastName);
    editTextPhone = findViewById(R.id.editTextPhone);
    buttonUpdate = findViewById(R.id.buttonUpdate);

    editTextEmail.setEnabled(false);
    editTextPassword.setEnabled(false);

    buttonUpdate.setOnClickListener(v -> updateUserDetails());

```



```

        databaseService.getUser(uid, new
DatabaseService.DatabaseCallback<User>() {
    @Override
    public void onCompleted(User object) {
        user = object;
        if (user != null) {
            editTextEmail.setText(user.getEmail());
            editTextFirstName.setText(user.getfName());
            editTextLastName.setText(user.getlName());
            editTextPhone.setText(user.getPhone());
            editTextPassword.setText(user.getPassword());

            cart = user.getCart();
        }
    }

    @Override
    public void onFailed(Exception e) {

        Toast.makeText(UpdateUserDetailsActivity.this, "Failed to load user
data", Toast.LENGTH_SHORT).show();
    }
});
}

public void btnBack4(View view) {
    Intent intent = new Intent(UpdateUserDetailsActivity.this,
UserAfterLoginPage.class);
    startActivity(intent);
}

private void updateUserDetails() {
    String email = editTextEmail.getText().toString().trim();
    String password = editTextPassword.getText().toString().trim();
    String fName = editTextFirstName.getText().toString().trim();
    String lName = editTextLastName.getText().toString().trim();

```

```

String phone = editTextPhone.getText().toString().trim();

// בדיקות תקינות
if (TextUtils.isEmpty(email) || TextUtils.isEmpty(fName) ||
TextUtils.isEmpty(lName) || TextUtils.isEmpty(phone)) {
    Toast.makeText(this, "Please fill all required fields",
Toast.LENGTH_SHORT).show();
    return;
}

// בדיקת תקינות מספר הטלפון (למשל, לפחות 10 תווים)
if (phone.length() < 10) {
    Toast.makeText(this, "Phone number must be at least 10 digits",
Toast.LENGTH_SHORT).show();
    return;
}

user.setName(lName);
user.setfName(fName);
user.setPhone(phone);

databaseService.updateUser(user, new
DatabaseService.DatabaseCallback<Void>() {
    @Override
    public void onCompleted(Void object) {
        Toast.makeText(UpdateUserDetailsActivity.this, "User updated
successfully!", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(UpdateUserDetailsActivity.this,
UserAfterLoginPage.class);
        startActivity(intent);
        finish();
    }
}

@Override

```

```
        public void onFailed(Exception e) {  
            Toast.makeText(UpdateUserDetailsActivity.this, "Failed to update  
User: " + e.getMessage(), Toast.LENGTH_SHORT).show();  
        }  
    });  
  
    }  
}
```

## UserAfterLoginPage

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.google.firebase.auth.FirebaseAuth;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

public class UserAfterLoginPage extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_user_after_login_page);

        if (getSupportActionBar() != null) {
```

```

// הגדרת כותרת מותאמת אישית
getSupportActionBar().setDisplayShowTitleEnabled(false);
getSupportActionBar().setDisplayShowCustomEnabled(true);
getSupportActionBar().setCustomView(R.layout.action_bar_shop);
TextView titlebar = findViewById(R.id.action_bar_text);
User user = SharedPreferencesUtil.getUser(this);
String currentUserName = user.getFName() + " " + user.getIName();
titlebar.setText("בחר הבא \uD83D\uDC4B " + currentUserName);

ImageView shopIcon = findViewById(R.id.shop_intro);

shopIcon.setOnClickListener(v -> {
    // אנימצית קפיצה
    v.animate()
        .scaleX(1.1f)
        .scaleY(1.1f)
        .setDuration(100)
        .withEndAction(() -> v.animate()
            .scaleX(1f)
            .scaleY(1f)
            .setDuration(100))
        .start();

    // יצירת BottomSheet
    View sheetView =
    LayoutInflater.from(this).inflate(R.layout.bottom_sheet_shop, null);
    BottomSheetDialog dialog = new BottomSheetDialog(this);
    dialog.setContentView(sheetView);
    dialog.show();

    // לחיצה על כפתור
    Button learnMoreBtn =
    sheetView.findViewById(R.id.btn_learn_more);
    learnMoreBtn.setOnClickListener(btn -> {
        Intent Intent = new Intent(this, Odot.class);
        startActivity(Intent);
    });

```

```

        finish();
    });
});
}

AuthenticationService.getInstance();

Button btnLogout = findViewById(R.id.btnLogout);

btnLogout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        logout();
    }
});
}

public void goToPurchaseHistory(View view) {
    Intent intent = new Intent(UserAfterLoginPage.this,
OrderHistoryActivity.class);
    startActivity(intent);
}

public void goToDeals(View view) {
    Intent intent = new Intent(UserAfterLoginPage.this, DealsActivity.class);
    startActivity(intent);
}

public void goToPersonalArea(View view) {
    Intent intent = new Intent(UserAfterLoginPage.this,
UpdateUserDetailsActivity.class);
    startActivity(intent);
}

public void logout() {
    SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);

```

```

SharedPreferences.Editor editor = sharedPreferences.edit();
editor.clear();
editor.apply();

AuthenticationService.getInstance().signOut();

Intent go = new Intent(UserAfterLoginPage.this, Login.class);
go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(go);

finishAffinity();
Toast.makeText(UserAfterLoginPage.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_user, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_catagories) {
        startActivity(new Intent(this, CategoriesActivity.class));
        return true;
    } else if (id == R.id.action_deals) {
        startActivity(new Intent(this, DealsActivity.class));
        return true;
    } else if (id == R.id.action_orders) {
        startActivity(new Intent(this, OrderHistoryActivity.class));
        return true;
    }
}

```

```

    } else if (id == R.id.action_update) {
        startActivity(new Intent(this, UpdateUserDetailsActivity.class));
        return true;
    } else if (id == R.id.action_logout) {
        SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();

        AuthenticationService.getInstance().signOut();

        Intent go = new Intent(this, Login.class);
        go.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(go);
        finishAffinity();
        Toast.makeText(UserAfterLoginPage.this, "התנתקת בהצלחה!",
Toast.LENGTH_SHORT).show();

    }

    return super.onOptionsItemSelected(item);
}
}

```



## UserDetailActivity

```
package com.shop.bagrutproject.screens;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.OrderAdapter;
import com.shop.bagrutproject.models.Order;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.function.Predicate;

public class UserDetailActivity extends AppCompatActivity {
```

```

TextView tvFName, tvLName, tvEmail, tvPhone;
ImageButton btnBack;
String uid, fName, lName, email, phone;
private List<Order> orders;
private RecyclerView recyclerView;
private OrderAdapter orderAdapter;
private DatabaseService databaseService;
private static final String TAG = "UserDetailActivity";

```

```

@Override

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user_detail);

    if (getSupportActionBar() != null) {
        getSupportActionBar().hide();
    }

```

```

    Button showRecyclerViewBtn = findViewById(R.id.showRecyclerViewBtn);
    Button closeRecyclerViewBtn = findViewById(R.id.closeRecyclerViewBtn);
    RecyclerView recyclerViewOrders =
findViewById(R.id.recyclerViewOrders2);

```

```

    OrderAdapter orderAdapter = new OrderAdapter(orders);
    recyclerViewOrders.setAdapter(orderAdapter);

```

```

    showRecyclerViewBtn.setOnClickListener(v -> {
        recyclerViewOrders.setVisibility(View.VISIBLE);

```

```

        closeRecyclerViewBtn.setVisibility(View.VISIBLE);
    });

```

```

    closeRecyclerViewBtn.setOnClickListener(v -> {
        recyclerViewOrders.setVisibility(View.GONE);

```

```

        closeRecyclerViewBtn.setVisibility(View.GONE);
    });

    tvFName = findViewById(R.id.tvFName);
    tvLName = findViewById(R.id.tvLName);
    tvEmail = findViewById(R.id.tvEmail);
    tvPhone = findViewById(R.id.tvPhone);
    orders = new ArrayList<>();
    databaseService = DatabaseService.getInstance();
    recyclerView = findViewById(R.id.recyclerViewOrders2);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    btnBack = findViewById(R.id.btnBack);

    Bundle extras = getIntent().getExtras();
    if (extras != null) {
        uid = extras.getString("USER_UID");
        fName = extras.getString("USER_FNAME");
        lName = extras.getString("USER_LNAME");
        email = extras.getString("USER_EMAIL");
        phone = extras.getString("USER_PHONE");
        fetchOrders(extras.getString("USER_UID"));

        // Set the data to the TextViews
        tvFName.setText("First Name: " + fName);
        tvLName.setText("Last Name: " + lName);
        tvEmail.setText("Email: " + email);
        tvPhone.setText("Phone: " + phone);
    }

    btnBack.setOnClickListener(v -> {
        onBackPressed();
    });
}

private void fetchOrders(String userId) {

```

```

List<Order> combinedOrders = new ArrayList<>();

// הרגיל orders-שליפה ראשונה - מה
DatabaseReference ordersRef =
FirebaseDatabase.getInstance().getReference("orders");
ordersRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot snapshot) {
        for (DataSnapshot orderSnap : snapshot.getChildren()) {
            Order order = orderSnap.getValue(Order.class);
            if (order != null && order.getUserId().equals(userId)) {
                combinedOrders.add(order);
            }
        }
    }
});

// archivedOrders-שליפה שנייה - מה
DatabaseReference archivedRef =
FirebaseDatabase.getInstance().getReference("archivedOrders");
archivedRef.addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot snapshot) {
        for (DataSnapshot archivedSnap : snapshot.getChildren()) {
            Order order = archivedSnap.getValue(Order.class);
            if (order != null && order.getUserId().equals(userId)) {
                combinedOrders.add(order);
            }
        }
    }
});

// עדכון הרשימה הסופית
if (combinedOrders.isEmpty()) {
    Toast.makeText(UserDetailActivity.this, "לא נמצאו הזמנות",
Toast.LENGTH_SHORT).show();
}

orders.clear();
orders.addAll(combinedOrders);

```

```

        orderAdapter = new OrderAdapter(orders);
        recyclerView.setAdapter(orderAdapter);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        Toast.makeText(UserDetailActivity.this, "שגיאה בטעינת ארכיון  
ההזמנות", Toast.LENGTH_SHORT).show();
        Log.e(TAG, "Error loading archived orders", error.toException());
    }
}

    @Override
    public void onCancelled(DatabaseError error) {
        Toast.makeText(UserDetailActivity.this, "שגיאה בטעינת ההזמנות",  
Toast.LENGTH_SHORT).show();
        Log.e(TAG, "Error loading orders", error.toException());
    }
}
}

```

## UsersActivity

```
package com.shop.bagrutproject.screens;

import android.os.Bundle;
import android.widget.Button;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.adapters.UsersAdapter;
import com.shop.bagrutproject.models.User;
import java.util.ArrayList;
import java.util.List;

public class UsersActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private UsersAdapter usersAdapter;
    private FirebaseDatabase database;
    private DatabaseReference myRef;
    private Button backButton;
    private SearchView userSearchView;
    private List<User> userList;

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

```

if (getSupportActionBar() != null) {
    getSupportActionBar().hide();
}

// XML-קישור רכיבים מה
userSearchView = findViewById(R.id.userSearchView);
recyclerView = findViewById(R.id.UserRecyclerView);
backButton = findViewById(R.id.backButton);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

database = FirebaseDatabase.getInstance();
myRef = database.getReference("Users");

userList = new ArrayList<>();
usersAdapter = new UsersAdapter(userList, this);
recyclerView.setAdapter(usersAdapter);

// טעינת המשתמשים מהפירבייס
fetchUsers();

// כפתור חזור
backButton.setOnClickListener(v -> finish());

// מאזין לחיפוש משתמשים
userSearchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        usersAdapter.filter(query);
        return false;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
        usersAdapter.filter(newText);
        return false;
    }
}

```

```

    }
    });
}

private void fetchUsers() {
    myRef.addListenerForSingleValueEvent(new
com.google.firebase.database.ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        userList.clear();
        for (DataSnapshot data : snapshot.getChildren()) {
            User user = data.getValue(User.class);
            if (user != null) {
                userList.add(user);
            }
        }
        usersAdapter.filter(""); // טוען את כל המשתמשים ומציג אותם
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // טיפול בשגיאה
    }
    });
}
}

```



## AuthenticationService

```
package com.shop.bagrutproject.services;

import android.util.Log;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import org.jetbrains.annotations.NotNull;

/// a service to interact with the Firebase Authentication.
/// this class is a singleton, use getInstance() to get an instance of this class
/// @see FirebaseAuth
public class AuthenticationService {

    /// tag for logging
    /// @see Log
    private static final String TAG = "AuthenticationService";

    /// callback interface for authentication operations
    /// @param <T> the type of the object to return
    /// @see AuthCallback#onCompleted(Object)
    /// @see AuthCallback#onFailed(Exception)
    public interface AuthCallback<T> {
        /// called when the operation is completed successfully
        void onCompleted(T object);

        /// called when the operation fails with an exception
        void onFailed(Exception e);
    }

    /// the instance of this class
    /// @see #getInstance()
    private static AuthenticationService instance;
```

```

    /// the reference to the authentication
    /// @see FirebaseAuth
    private final FirebaseAuth mAuth;

    /// use getInstance() to get an instance of this class
    private AuthenticationService() {
        mAuth = FirebaseAuth.getInstance();
    }

    /// get an instance of this class
    /// @return an instance of this class
    /// @see AuthenticationService
    public static AuthenticationService getInstance() {
        if (instance == null) {
            instance = new AuthenticationService();
        }
        return instance;
    }

    /// sign in a user with email and password
    /// @param email the email of the user
    /// @param password the password of the user
    /// @param callback the callback to call when the operation is completed
    /// the callback will receive true if the operation is successful
    /// if the operation fails, the callback will receive an exception
    /// @return void
    /// @see AuthCallback
    public void signIn(@NotNull final String email, @NotNull final String
password, @NotNull final AuthCallback<String> callback) {
        mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                callback.onCompleted(getCurrentUserId());
            } else {
                Log.e(TAG, "Error signing in", task.getException());
                callback.onFailed(task.getException());
            }
        });
    }

```

```

    }
    });
}

/// sign up a new user with email and password
/// @param email the email of the user
/// @param password the password of the user
/// @param callback the callback to call when the operation is completed
/// the callback will receive the FirebaseAuth object if the operation is
successful
/// if the operation fails, the callback will receive an exception
/// @see AuthCallback
/// @see FirebaseAuth
public void signUp(@NotNull final String email, @NotNull final String
password, @NotNull final AuthCallback<String> callback) {
    mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            callback.onCompleted(getCurrentUserId());
        } else {
            Log.e(TAG, "Error signing up", task.getException());
            callback.onFailed(task.getException());
        }
    });
}

/// sign out the current user
public void signOut() {
    mAuth.signOut();
}

/// get the current user ID
/// @return the current user ID
public String getCurrentUserId() {
    if (mAuth.getCurrentUser() == null) {
        return null;
    }
}

```

```
        return mAuth.getCurrentUser().getUid();
    }

    /// check if a user is signed in
    /// @return true if a user is signed in, false otherwise
    public boolean isUserSignedIn() {
        return mAuth.getCurrentUser() != null;
    }

}
```

## DatabaseService

```
package com.shop.bagrutproject.services;

import android.util.Log;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;
import com.shop.bagrutproject.models.Cart;
import com.shop.bagrutproject.models.Comment;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;
import com.shop.bagrutproject.models.Order;
import com.shop.bagrutproject.models.User;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/// a service to interact with the Firebase Realtime Database.
/// this class is a singleton, use getInstance() to get an instance of this class
/// @see #getInstance()
/// @see FirebaseDatabase

public class DatabaseService {
```

```

/// tag for logging
/// @see Log
private static final String TAG = "DatabaseService";

/// callback interface for database operations
/// @param <T> the type of the object to return
/// @see DatabaseCallback#onCompleted(Object)
/// @see DatabaseCallback#onFailed(Exception)
public interface DatabaseCallback<T> {
    /// called when the operation is completed successfully
    void onCompleted(T object);

    /// called when the operation fails with an exception
    void onFailed(Exception e);
}

/// the instance of this class
/// @see #getInstance()
private static DatabaseService instance;

/// the reference to the database
/// @see DatabaseReference
/// @see FirebaseDatabase#getReference()
private final DatabaseReference databaseReference;

/// use getInstance() to get an instance of this class
/// @see DatabaseService#getInstance()
private DatabaseService() {
    FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
    databaseReference = firebaseDatabase.getReference();
}

/// get an instance of this class
/// @return an instance of this class
/// @see DatabaseService
public static DatabaseService getInstance() {

```

```

    if (instance == null) {
        instance = new DatabaseService();
    }
    return instance;
}

// private generic methods to write and read data from the database

/// write data to the database at a specific path
/// @param path the path to write the data to
/// @param data the data to write (can be any object, but must be
serializable, i.e. must have a default constructor and all fields must have
getters and setters)
/// @param callback the callback to call when the operation is completed
/// @return void
/// @see DatabaseCallback
private void writeData(@NotNull final String path, @NotNull final Object
data, final @Nullable DatabaseCallback<Void> callback) {

databaseReference.child(path).setValue(data).addOnCompleteListener(task ->
{
    if (task.isSuccessful()) {
        if (callback == null) return;
        callback.onCompleted(task.getResult());
    } else {
        if (callback == null) return;
        callback.onFailed(task.getException());
    }
});
}

/// read data from the database at a specific path
/// @param path the path to read the data from
/// @return a DatabaseReference object to read the data from
/// @see DatabaseReference

```

```

private DatabaseReference readData(@NotNull final String path) {
    return databaseReference.child(path);
}

/// get data from the database at a specific path
/// @param path the path to get the data from
/// @param clazz the class of the object to return
/// @param callback the callback to call when the operation is completed
/// @return void
/// @see DatabaseCallback
/// @see Class
private <T> void getData(@NotNull final String path, @NotNull final Class<T>
clazz, @NotNull final DatabaseCallback<T> callback) {
    readData(path).get().addOnCompleteListener(task -> {
        if (!task.isSuccessful()) {
            Log.e(TAG, "Error getting data", task.getException());
            callback.onFailed(task.getException());
            return;
        }
        T data = task.getResult().getValue(clazz);
        callback.onCompleted(data);
    });
}

/// remove data from the database at a specific path
/// @param path the path to remove the data from
/// @param callback the callback to call when the operation is completed
/// @see DatabaseCallback
private void deleteData(@NotNull final String path, @Nullable final
DatabaseCallback<Void> callback) {
    readData(path).removeValue((error, ref) -> {
        if (error != null) {
            if (callback == null) return;
            callback.onFailed(error.toException());
        } else {
            if (callback == null) return;
        }
    });
}

```



```

        callback.onCompleted(null);
    }
});
}

/// get a list of data from the database at a specific path
/// @param path the path to get the data from
/// @param clazz the class of the objects to return
/// @param callback the callback to call when the operation is completed
private <T> void getDataList(@NotNull final String path, @NotNull final
Class<T> clazz, @NotNull Map<String, String> filter, @NotNull final
DatabaseCallback<List<T>> callback) {
    Query dbRef = readData(path);

    for (Map.Entry<String, String> entry : filter.entrySet()) {
        dbRef = dbRef.orderByChild(entry.getKey()).equalTo(entry.getValue());
    }

    dbRef.get().addOnCompleteListener(task -> {
        if (!task.isSuccessful()) {
            Log.e(TAG, "Error getting data", task.getException());
            callback.onFailed(task.getException());
            return;
        }
        List<T> tList = new ArrayList<>();
        task.getResult().getChildren().forEach(dataSnapshot -> {
            T t = dataSnapshot.getValue(clazz);
            tList.add(t);
        });

        callback.onCompleted(tList);
    });
}

/// generate a new id for a new object in the database
/// @param path the path to generate the id for

```

```

    /// @return a new id for the object
    /// @see String
    /// @see DatabaseReference#push()

    private String generateNewId(@NotNull final String path) {
        return databaseReference.child(path).push().getKey();
    }

    // end of private methods for reading and writing data

    // public methods to interact with the database

    /// create a new user in the database
    /// @param user the user object to create
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive void
    ///         if the operation fails, the callback will receive an exception
    /// @return void
    /// @see DatabaseCallback
    /// @see User
    public void createUser(@NotNull final User user, @Nullable final
DatabaseCallback<Void> callback) {
        writeData("Users/" + user.getUid(), user, callback);
    }


    /// get a user from the database
    /// @param uid the id of the user to get
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive the user object
    ///         if the operation fails, the callback will receive an exception
    /// @return void
    /// @see DatabaseCallback
    /// @see User
    public void getUser(@NotNull final String uid, @NotNull final
DatabaseCallback<User> callback) {

```

```

        getData("Users/" + uid, User.class, callback);
    }

    /// get a cart from the database
    /// @param uid the id of the cart to get
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive the cart object
    ///         if the operation fails, the callback will receive an exception
    /// @return void
    /// @see DatabaseCallback
    /// @see Cart
    public void getCart(@NotNull final String uid, @NotNull final
DatabaseCallback<Cart> callback) {
        getData("Users/" + uid + "/cart" , Cart.class, callback);
    }

    /// generate a new id for a new item in the database
    /// @return a new id for the item
    /// @see #generateNewId(String)
    /// @see Item
    public String generateItemId() {
        return generateNewId("items");
    }

    /// generate a new id for a new cart in the database
    /// @return a new id for the cart
    /// @see #generateNewId(String)
    /// @see Cart
    public String generateCartId() {
        return generateNewId("carts");
    }

    /// create a new item in the database
    /// @param item the item object to create
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive void
    ///         if the operation fails, the callback will receive an exception

```

```

    /// @return void
    /// @see DatabaseCallback
    /// @see Item
    public void createNewItem(@NotNull final Item item, @Nullable final
DatabaseCallback<Void> callback) {
        writeData("items/" + item.getId(), item, callback);
    }

    /// create a new cart in the database
    /// @param cart the cart object to create
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive void
    ///         if the operation fails, the callback will receive an exception
    /// @return void
    /// @see DatabaseCallback
    /// @see Cart
    public void updateCart(@NotNull final Cart cart,String uid ,@Nullable final
DatabaseCallback<Void> callback) {
        writeData("Users/" + uid+"/cart", cart, callback);
    }

    public void updateUser(@NotNull final User user ,@Nullable final
DatabaseCallback<Void> callback) {
        writeData("Users/" + user.getUid(), user, callback);
    }

```

```

    /// get a item from the database
    /// @param itemId the id of the item to get
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive the item object
    ///         if the operation fails, the callback will receive an exception
    /// @return void

```

```

    /// @see DatabaseCallback
    /// @see Item
    public void getItem(@NotNull final String itemId, @NotNull final
DatabaseCallback<Item> callback) {
        getData("items/" + itemId, Item.class, callback);
    }

    /// get all the items from the database
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive a list of item objects
    ///         if the operation fails, the callback will receive an exception
    /// @return void
    /// @see DatabaseCallback
    /// @see List
    /// @see Item
    /// @see #getData(String, Class, DatabaseCallback)
    public void getItems(@NotNull final DatabaseCallback<List<Item>> callback)
{
        getDataList("items", Item.class, new HashMap<>(), callback);
    }

    /// get all the users from the database
    /// @param callback the callback to call when the operation is completed
    ///         the callback will receive a list of item objects
    ///         if the operation fails, the callback will receive an exception
    /// @return void
    /// @see DatabaseCallback
    /// @see List
    /// @see Item
    /// @see #getData(String, Class, DatabaseCallback)
    public void getUsers(@NotNull final DatabaseCallback<List<User>> callback)
{
        getDataList("Users", User.class, new HashMap<>(), callback);
    }

```

```

public void deleteUser(String uid, DatabaseCallback<Void> callback) {
    DatabaseReference userRef =
FirebaseDatabase.getInstance().getReference("Users").child(uid);
    userRef.removeValue().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            callback.onCompleted(null);
        } else {
            callback.onFailed(task.getException());
        }
    });
}

public void saveOrder(Order order, DatabaseCallback<Void> callback) {
    DatabaseReference ref = FirebaseDatabase.getInstance()
        .getReference("orders")
        .child(order.getOrderId());

    ref.setValue(order)
        .addOnSuccessListener(aVoid -> callback.onCompleted(null))
        .addOnFailureListener(callback::onFailed);
}

public void getOrder(String orderId, DatabaseCallback<Order> callback) {
    getData("orders/"+orderId, Order.class, callback);
}

public void getOrders(final DatabaseCallback<List<Order>> callback) {
    getDataList("orders", Order.class, new HashMap<>(), callback);
}

public void getComments(String itemId, DatabaseCallback<List<Comment>>
callback) {
    getDataList("comments/" + itemId, Comment.class, new HashMap<>(),
callback);
}

```

```

    public void writeNewComment(String itemId, Comment comment,
DatabaseCallback<Void> callback) {
        writeData("comments/" + itemId + "/" + comment.getCommentId(),
comment, callback);
    }

```

```

    public String generateNewCommentId(String itemId) {
        return generateNewId("comments/" + itemId);
    }

```

```

    public void removeComment(String itemId, String commentId,
DatabaseCallback<Void> callback) {
        deleteData("comments/" + itemId + "/" + commentId, callback) ;
    }

```

```

// פונקציה שתחשב את הממוצע של הדיחגים
    public void updateAverageRating(String itemId, DatabaseCallback<Double>
callback) {
        DatabaseReference commentsRef =
FirebaseDatabase.getInstance().getReference("comments").child(itemId);
        commentsRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                double sum = 0;
                int count = 0;
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Comment comment = snapshot.getValue(Comment.class);
                    if (comment != null) {
                        sum += comment.getRating();
                        count++;
                    }
                }
                double averageRating = count > 0 ? sum / count : 0;
                // עדכון הדיחג הממוצע של המוצר
                DatabaseReference itemRef =
FirebaseDatabase.getInstance().getReference("items").child(itemId);

```

```

        itemRef.child("averageRating").setValue(averageRating)
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    callback.onCompleted(averageRating);
                } else {
                    callback.onFailed(task.getException());
                }
            });
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        callback.onFailed(databaseError.toException());
    }
});
}

// הוספת מבצע חדש
public void addDeal(Deal deal, @NotNull DatabaseCallback<Void> callback) {
    // שנקבע מראש ID ללא Firebase יצירת מפתח אוטומטי על ידי
    DatabaseReference ref =
    FirebaseDatabase.getInstance().getReference("deals").push();
    deal.setId(ref.getKey()); // ID אם אתה לא חצה לשמור
    הוצאת
    ref.setValue(deal)
        .addOnSuccessListener(unused -> callback.onCompleted(null))
        .addOnFailureListener(callback::onFailed);
}

// שליפת כל המבצעים
public void getAllDeals(@NotNull DatabaseCallback<List<Deal>> callback) {
    DatabaseReference ref =
    FirebaseDatabase.getInstance().getReference("deals");
    ref.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            List<Deal> deals = new ArrayList<>();

```



```

        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            Deal deal = snapshot.getValue(Deal.class);
            if (deal != null) {
                deals.add(deal);
            }
        }
        callback.onCompleted(deals);
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        callback.onFailed(databaseError.toException());
    }
});
}
}

```

## DealNotificationReceiver

```
package com.shop.bagrutproject.utils;

import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Build;

import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.screens.Login;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class DealNotificationFetcher {

    public static final String CHANNEL_ID = "shop_notifications";
    private static final int NOTIFICATION_ID = 1001;

    public static void fetchAndSendDealNotification(Context context) {
        if (context == null) return;

        FirebaseDatabase.getInstance().getReference("deals")
            .addListenerForSingleValueEvent(new ValueEventListener() {
```

```

@Override
public void onDataChange(DataSnapshot snapshot) {
    List<Deal> deals = new ArrayList<>();
    for (DataSnapshot data : snapshot.getChildren()) {
        Deal deal = data.getValue(Deal.class);
        if (deal != null) {
            deals.add(deal);
        }
    }

    if (!deals.isEmpty()) {
        Deal randomDeal = deals.get(new
Random().nextInt(deals.size()));
        sendNotification(context, randomDeal.getTitle(),
randomDeal.getDescription());
    }
}

@Override
public void onCancelled(DatabaseError error) {
    // אפשר להוסיף לוג או טוסט במקרה של שגיאה
}
});
}

private static void sendNotification(Context context, String title, String
message) {
    if (context == null) return;

    //ומעלה 13-Android בדיקת הרשאות להתראות (מ)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU &&
context.checkSelfPermission(android.Manifest.permission.POST_NOTIFICATIONS)
!= PackageManager.PERMISSION_GRANTED) {
        return;
    }
}

```

```

//אימוג'ים רנדומליים לבותרת ולתיאור
String[] titleEmojis = {"🍌", "🔥", "🌟", "💣", "⚡"};
String[] messageEmojis = {"🏠", "😬", "📦", "🔑", "🛒"};

Random random = new Random();
String emojiTitle = titleEmojis[random.nextInt(titleEmojis.length)] + " " +
title + " " + titleEmojis[random.nextInt(titleEmojis.length)];
String emojiMessage =
messageEmojis[random.nextInt(messageEmojis.length)] + " " + message + " " +
messageEmojis[random.nextInt(messageEmojis.length)];

//באשר לוחצים על ההתראה – פותח את מסך הלוגין (אפשר לשנות למסך אחר)
Intent intent = new Intent(context, Login.class);
PendingIntent pendingIntent = PendingIntent.getActivity(
    context, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
);

NotificationCompat.Builder builder = new
NotificationCompat.Builder(context, CHANNEL_ID)
    .setSmallIcon(R.drawable.ic_electric_plug)
    .setContentTitle(emojiTitle)
    .setContentText(emojiMessage)
    .setPriority(NotificationCompat.PRIORITY_HIGH)
    .setAutoCancel(true)
    .setContentIntent(pendingIntent);

NotificationManagerCompat.from(context).notify(NOTIFICATION_ID,
builder.build());
}
}

```

## ImageUtil

```
package com.shop.bagrutproject.utils;

import android.Manifest;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.BitmapDrawable;
import android.util.Base64;
import android.widget.ImageView;

import androidx.core.app.ActivityCompat;

import org.jetbrains.annotations.NotNull;
import org.jetbrains.annotations.Nullable;

import java.io.ByteArrayOutputStream;

/// Utility class for image operations
/// Contains methods for requesting permissions, converting images to base64
and vice versa
public class ImageUtil {

    /// Request permissions for camera and storage
    /// @param activity The activity to request permissions from
    /// @see ActivityCompat#requestPermissions(Activity, String[], int)
    public static void requestPermission(@NotNull Activity activity) {
        // Request permissions for camera and storage
        ActivityCompat.requestPermissions(activity,
            new String[]{
                Manifest.permission.CAMERA,
                Manifest.permission.WRITE_EXTERNAL_STORAGE,
                Manifest.permission.READ_EXTERNAL_STORAGE
            }, 1);
    }
}
```

```

    }

    /// Convert an image to a base64 string
    /// @param postImage The image to convert
    /// @return The base64 string representation of the image
    public static @Nullable String convertTo64Base(@NotNull final ImageView
postImage) {
        if (postImage.getDrawable() == null) {
            return null;
        }
        Bitmap bitmap = ((BitmapDrawable)
postImage.getDrawable()).getBitmap();
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG, 100,
byteArrayOutputStream);
        byte[] byteArray = byteArrayOutputStream.toByteArray();
        return Base64.encodeToString(byteArray, Base64.DEFAULT);
    }

    /// Convert a base64 string to an image
    /// @param base64Code The base64 string to convert
    /// @return The image represented by the base64 string
    public static @Nullable Bitmap convertFrom64base(@NotNull final String
base64Code) {
        if (base64Code.isEmpty()) {
            return null;
        }
        byte[] decodedString = Base64.decode(base64Code, Base64.DEFAULT);
        return BitmapFactory.decodeByteArray(decodedString, 0,
decodedString.length);
    }
}

```

## NotificationReceiver

```
package com.shop.bagrutproject.utils;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class NotificationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (context != null) {
            DealNotificationFetcher.fetchAndSendDealNotification(context);
        }
    }
}
```

## SharedPreferencesUtil

```
package com.shop.bagrutproject.utils;

import android.content.Context;
import android.content.SharedPreferences;

import com.shop.bagrutproject.models.Cart;
import com.shop.bagrutproject.models.User;

/// Utility class for shared preferences operations
/// Contains methods for saving and retrieving data from shared preferences
/// Also contains methods for clearing and removing data from shared
preferences
/// @see SharedPreferences
public class SharedPreferencesUtil {

    /// The name of the shared preferences file
    /// @see Context#getSharedPreferences(String, int)
    private static final String PREF_NAME =
"com.shop.bagrutproject.PREFERENCE_FILE_KEY";

    /// Save a string to shared preferences
    /// @param context The context to use
    /// @param key The key to save the string with
    /// @param value The string to save
    /// @see SharedPreferences.Editor#putString(String, String)
    private static void saveString(Context context, String key, String value) {
        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString(key, value);
        editor.apply();
    }
}
```



```

    /// Get a string from shared preferences
    /// @param context The context to use
    /// @param key The key to get the string with
    /// @param defaultValue The default value to return if the key is not found
    /// @return The string value stored in shared preferences
    /// @see SharedPreferences#getString(String, String)
    private static String getString(Context context, String key, String
defaultValue) {
        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        return sharedPreferences.getString(key, defaultValue);
    }

    /// Save an integer to shared preferences
    /// @param context The context to use
    /// @param key The key to save the integer with
    /// @param value The integer to save
    /// @see SharedPreferences.Editor#putInt(String, int)
    private static void saveInt(Context context, String key, int value) {
        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putInt(key, value);
        editor.apply();
    }

    /// Get an integer from shared preferences
    /// @param context The context to use
    /// @param key The key to get the integer with
    /// @param defaultValue The default value to return if the key is not found
    /// @return The integer value stored in shared preferences
    /// @see SharedPreferences#getInt(String, int)
    private static int getInt(Context context, String key, int defaultValue) {
        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        return sharedPreferences.getInt(key, defaultValue);
    }

```

```

// Add more methods for other data types as needed

/// Clear all data from shared preferences
/// @param context The context to use
/// @see SharedPreferences.Editor#clear()
public static void clear(Context context) {
    SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.clear();
    editor.apply();
}

/// Remove a specific key from shared preferences
/// @param context The context to use
/// @param key The key to remove
/// @see SharedPreferences.Editor#remove(String)
private static void remove(Context context, String key) {
    SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.remove(key);
    editor.apply();
}

/// Check if a key exists in shared preferences
/// @param context The context to use
/// @param key The key to check
/// @return true if the key exists, false otherwise
/// @see SharedPreferences#contains(String)
private static boolean contains(Context context, String key) {
    SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    return sharedPreferences.contains(key);
}

```

```

// Add more utility methods as needed

/// Save a user object to shared preferences
/// @param context The context to use
/// @param user The user object to save
/// @see User
public static void saveUser(Context context, User user) {
    SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString("uid", user.getUid());
    editor.putString("email", user.getEmail());
    editor.putString("password", user.getPassword());
    editor.putString("fName", user.getfName());
    editor.putString("lName", user.getlName());
    editor.putString("phone", user.getPhone());
    editor.apply();
}

/// Get the user object from shared preferences
/// @param context The context to use
/// @return The user object stored in shared preferences
/// @see User
/// @see #isUserLoggedIn(Context)
public static User getUser(Context context) {
    SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    // check if user is logged in
    if (!isUserLoggedIn(context)) {
        return null;
    }
    String uid = sharedPreferences.getString("uid", "");
    String email = sharedPreferences.getString("email", "");
    String password = sharedPreferences.getString("password", "");
    String fName = sharedPreferences.getString("fName", "");

```

```

        String lName = sharedPreferences.getString("lName", "");
        String phone = sharedPreferences.getString("phone", "");
        return new User(uid, email, password, fName, lName, phone, null);
    }

    /// Sign out the user by removing user data from shared preferences
    /// @param context The context to use
    public static void signOutUser(Context context) {
        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.remove("uid");
        editor.remove("email");
        editor.remove("password");
        editor.remove("fName");
        editor.remove("lName");
        editor.remove("phone");

        editor.apply();
    }

    /// Check if a user is logged in by checking if the user id is present in shared
preferences
    /// @param context The context to use
    /// @return true if the user is logged in, false otherwise
    /// @see #contains(Context, String)
    public static boolean isUserLoggedIn(Context context) {
        return contains(context, "uid");
    }

    public static boolean isAdmin(Context context) {
        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        return sharedPreferences.getBoolean("isAdmin", false);
    }

    public static void setIsAdmin(Context context, boolean isAdmin) {

```

```

        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putBoolean("isAdmin", isAdmin);
        editor.apply();
    }

    public static boolean isAdminLoggedIn(Context context) {
        return isUserLoggedIn(context) && isAdmin(context);
    }

    public static void signOutAdmin(Context context) {
        SharedPreferences sharedPreferences =
context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.remove("isAdmin");
        editor.apply();
    }
}

```

## Cart

```
package com.shop.bagrutproject.models;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class Cart implements Serializable {

    private List<Item> items;

    public Cart( List<Item> items) {
        this.items = items;
    }

    public Cart() {
        this.items = new ArrayList<>();
    }

    public void addItem(Item item) {

        if(this.items==null){
            this.items=new ArrayList<>();
        }
        this.items.add(item);
    }

    public List<Item> getItems() {
        return this.items;
    }

    public void setItems(List<Item> items) {
        this.items = items;
    }
}
```

```
public void removeItem(int index) {
    this.items.remove(index);
}

@Override
public String toString() {
    return "Cart{" +

        ", items=" + this.items +
        '}';
}
}
```

## Category

```
package com.shop.bagrutproject.models;

public class Category {
    private String name;
    private int imageResId;

    public Category(String name, int imageResId) {
        this.name = name;
        this.imageResId = imageResId;
    }

    public String getName() {
        return name;
    }

    public int getImageResId() {
        return imageResId;
    }
}
```



## Comment

```
package com.shop.bagrutproject.models;
```

```
public class Comment {
```

```
    private String commentId;  
    private String userId;  
    private String commentText;  
    private float rating;  
    private String userName;
```

```
    public Comment() {
```

```
    }
```

```
    public Comment(String commentId, String userId, String commentText, float  
rating, String userName) {
```

```
        this.commentId = commentId;  
        this.userId = userId;  
        this.commentText = commentText;  
        this.rating = rating;  
        this.userName = userName;  
    }
```

```
    public String getUserName() {
```

```
        return userName;  
    }
```

```
    public void setUserName(String userName) {
```

```
        this.userName = userName;  
    }
```

```
    public String getCommentId() {
```

```
        return commentId;  
    }
```

```
public void setCommentId(String commentId) {
    this.commentId = commentId;
}

public String getUserId() {
    return userId;
}

public void setUserId(String userId) {
    this.userId = userId;
}

public String getCommentText() {
    return commentText;
}

public void setCommentText(String commentText) {
    this.commentText = commentText;
}

public float getRating() {
    return rating;
}

public void setRating(float rating) {
    this.rating = rating;
}
}
```

## Deal

```
package com.shop.bagrutproject.models;

import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.Date;

public class Deal {
    private String id;
    private String title;
    private String description;
    private double discountPercentage;
    private String validUntil;
    private String itemType;

    public Deal() {}

    public Deal(String id, String title, String description, double
discountPercentage, String validUntil, String itemType) {
        this.id = id;
        this.title = title;
        this.description = description;
        this.discountPercentage = discountPercentage;
        this.validUntil = validUntil;
        this.itemType = itemType;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
}
```

```

}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public double getDiscountPercentage() {
    return discountPercentage;
}

public void setDiscountPercentage(double discountPercentage) {
    this.discountPercentage = discountPercentage;
}

public String getValidUntil() {
    return validUntil;
}

public void setValidUntil(String validUntil) {
    this.validUntil = validUntil;
}

public String getItemType() {
    return itemType;
}

```

```

public void setItemType(String itemType) {
    this.itemType = itemType;
}

public boolean isValid() {
    String validUntilString = this.validUntil; // התאריך "31/05/2025"

    try {
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        Date validUntilDate = sdf.parse(validUntilString);

        // אם אתה חצה להשוות לתאריך נוכחי:
        Date currentDate = new Date();
        return !validUntilDate.before(currentDate);
    } catch (Exception e) {
        // טיפול בשגיאות (אם הפורמט לא תואם)
        return false;
    }
}
}

```

## Item

```
package com.shop.bagrutproject.models;

import java.io.Serializable; // הוספתי את ה-Serializable
import java.util.List;

public class Item implements Serializable {

    protected String id;

    protected String name;
    protected String type;
    protected String color;
    protected String company;
    protected String aboutItem;
    protected double price;
    protected String pic;

    public Item() {
    }

    public Item(String id, String name, String type, String color, String company,
String aboutItem, double price, String pic) {
        this.id = id;
        this.name = name;
        this.type = type;
        this.color = color;
        this.company = company;
        this.aboutItem = aboutItem;
        this.price = price;
        this.pic = pic;
    }

    public String getId() {
        return id;
    }
}
```

```

}

public void setId(String id) {
    this.id = id;
}

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getType() {
    return type;
}

public void setType(String type) {
    this.type = type;
}

public String getCompany() {
    return company;
}

public void setCompany(String company) {
    this.company = company;
}

public String getColor() {
    return color;
}

public void setColor(String color) {
    this.color = color;
}

public double getPrice() {
    return price;
}

```

```

public void setPrice(double price) {
    this.price = price;
}

public String getAboutItem() {
    return aboutItem;
}

public void setAboutItem(String aboutItem) {
    this.aboutItem = aboutItem;
}

public String getPic() {
    return pic;
}

public void setPic(String pic) {
    this.pic = pic;
}

@Override
public String toString() {
    return "Item{" +
        "id=" + id + "\" +
        ", name=" + name + "\" +
        ", type=" + type + "\" +
        ", color=" + color + "\" +
        ", company=" + company + "\" +
        ", aboutItem=" + aboutItem + "\" +
        ", price=" + price +
        '"';
}
}

```



## Order

```
package com.shop.bagrutproject.models;

import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.UUID;

public class Order implements Serializable {
    private String orderId;
    private List<Item> items;
    private double totalPrice;
    private String status;
    private long timestamp;
    private String userId;
    private String address;
    private String paymentMethod;

    public Order(String userId, List<Item> items) {
        this.orderId = UUID.randomUUID().toString();
        this.items = items != null ? items : new ArrayList<>();
        this.totalPrice = calculateTotalPrice();
        this.status = "Pending";
        this.timestamp = System.currentTimeMillis();
        this.userId = userId;
    }

    public Order() {
        this.orderId = UUID.randomUUID().toString();
        this.items = new ArrayList<>();
        this.totalPrice = 0;
        this.status = "Pending";
    }
}
```

```

        this.timestamp = System.currentTimeMillis();
    }

    private double calculateTotalPrice() {
        double sum = 0;
        for (Item item : items) {
            sum += item.getPrice();
        }
        return sum;
    }

    public String getOrderId() {
        return orderId;
    }

    public List<Item> getItems() {
        return items;
    }

    public double getTotalPrice() {
        return totalPrice;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public long getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(long timestamp) {
        this.timestamp = timestamp;
    }

```

```

    }

    public String getUserId() {
        return userId;
    }

    public String getFormattedDate() {
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy",
Locale.getDefault());
        return sdf.format(new Date(this.timestamp));
    }

    public void setTotalPrice(double totalPrice) {
        this.totalPrice = totalPrice;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getPaymentMethod() {
        return paymentMethod;
    }

    public void setPaymentMethod(String paymentMethod) {
        this.paymentMethod = paymentMethod;
    }

    @Override
    public String toString() {
        return "Order{" +
            "orderId=" + orderId + "\" +
            ", userId=" + userId + "\" +

```

```
    ", totalPrice=" + totalPrice +  
    ", status=" + status + "\" +  
    ", timestamp=" + timestamp +  
    ", address=" + address + "\" +  
    ", paymentMethod=" + paymentMethod + "\" +  
    ", items=" + items +  
    '};'  
  }  
}
```

## User

```
package com.shop.bagrutproject.models;

import org.jetbrains.annotations.NotNull;

public class User {

    protected String uid;
    protected String email;
    protected String password;
    protected String fName;
    protected String lName;
    protected String phone;

    protected Cart cart;

    public User() {
        cart = new Cart();
    }

    public User(String uid, String email, String password, String fName, String
lName, String phone, Cart cart) {
        this.uid = uid;
        this.email = email;
        this.password = password;
        this.fName = fName;
        this.lName = lName;
        this.phone = phone;
        this.cart = cart;
    }

    public String getUid() {
        return uid;
    }
}
```

```
public void setUid(String uid) {
    this.uid = uid;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getfName() {
    return fName;
}

public void setfName(String fName) {
    this.fName = fName;
}

public String getlName() {
    return lName;
}

public void setlName(String lName) {
    this.lName = lName;
}
```

```

public String getPhone() {
    return phone;
}

public void setPhone(String phone) {
    this.phone = phone;
}

public Cart getCart() {
    return cart;
}

public void setCart(Cart cart) {
    this.cart = cart;
}

@Override
public String toString() {
    return "User{" +
        "uid=" + uid + "\" +
        ", email=" + email + "\" +
        ", password=" + password + "\" +
        ", fName=" + fName + "\" +
        ", lName=" + lName + "\" +
        ", phone=" + phone + "\" +
        ", cart=" + cart +
        '"';
}
}

```

## CustomBackgroundView

```
package com.shop.bagrutproject.backgrounds;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.LinearGradient;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Shader;
import android.util.AttributeSet;
import android.view.View;

public class CustomBackgroundView extends View {
    private Paint gradientPaint;
    private Paint wavePaint;

    public CustomBackgroundView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init() {
        gradientPaint = new Paint();
        wavePaint = new Paint();
        wavePaint.setStyle(Paint.Style.FILL);
        wavePaint.setAntiAlias(true);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        int width = getWidth();
        int height = getHeight();
```



```

        canvas.drawRect(0, 0, width, height, gradientPaint);

        drawSubtleWaves(canvas, width, height);
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        // רקע עדין – אפרפר בהיר מאוד עם גוון שמנת
        LinearGradient gradient = new LinearGradient(
            0, 0, w, h,
            new int[]{0xFFEFEFEF, 0xFFF9F8F3}, // יותר רך
            null, Shader.TileMode.CLAMP
        );
        gradientPaint.setShader(gradient);
    }

    private void drawSubtleWaves(Canvas canvas, int w, int h) {
        // גל ראשון – אפור-כחלחל יוקרתי
        Shader shader1 = new LinearGradient(0, 0, w, h,
            0x70D0D3D4, 0x70E0E3E4, Shader.TileMode.CLAMP); // שקיפות
        פחותה (70)
        wavePaint.setShader(shader1);
        Path wave1 = new Path();
        wave1.moveTo(0, h * 0.76f);
        wave1.quadTo(w * 0.3f, h * 0.73f, w * 0.6f, h * 0.78f);
        wave1.quadTo(w * 0.9f, h * 0.80f, w, h * 0.76f);
        wave1.lineTo(w, h);
        wave1.lineTo(0, h);
        wave1.close();
        canvas.drawPath(wave1, wavePaint);

        // גל שני – פנינה אפרפר
        Shader shader2 = new LinearGradient(0, 0, w, h,
            0x60F0F0F0, 0x60FFFFFF, Shader.TileMode.CLAMP); // פחות שקוף
        wavePaint.setShader(shader2);
        Path wave2 = new Path();
    }

```

```

    wave2.moveTo(0, h * 0.81f);
    wave2.quadTo(w * 0.3f, h * 0.78f, w * 0.7f, h * 0.83f);
    wave2.quadTo(w * 0.9f, h * 0.86f, w, h * 0.82f);
    wave2.lineTo(w, h);
    wave2.lineTo(0, h);
    wave2.close();
    canvas.drawPath(wave2, wavePaint);

    // גל שלישי – גוון שמנת רכה
    Shader shader3 = new LinearGradient(0, 0, w, h,
        0x50FFFDf4, 0x50FFF8E1, Shader.TileMode.CLAMP); // גם כאן פחות
שקוף
    wavePaint.setShader(shader3);
    Path wave3 = new Path();
    wave3.moveTo(0, h * 0.86f);
    wave3.quadTo(w * 0.4f, h * 0.87f, w, h * 0.86f);
    wave3.lineTo(w, h);
    wave3.lineTo(0, h);
    wave3.close();
    canvas.drawPath(wave3, wavePaint);
}

}

```

## CustomBackgroundView2

```
package com.shop.bagrutproject.backgrounds;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.LinearGradient;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Shader;
import android.util.AttributeSet;
import android.view.View;

public class CustomBackgroundView2 extends View {
    public CustomBackgroundView2(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        int w = getWidth();
        int h = getHeight();

        // רקע עדין עם גרדיאנט שקוף
        Paint gradientPaint = new Paint();
        LinearGradient gradient = new LinearGradient(0, 0, w, h,
            0x80D4E0F6, 0x80F2F6FC, Shader.TileMode.CLAMP);
        gradientPaint.setShader(gradient);
        canvas.drawRect(0, 0, w, h, gradientPaint);

        // ציור גלים רכים
        drawSoftWaves(canvas, w, h);
    }
}
```

```

private void drawSoftWaves(Canvas canvas, int w, int h) {
    Paint paint = new Paint();
    paint.setStyle(Paint.Style.FILL);
    paint.setAntiAlias(true);

    // גל ראשון – תכלת
    paint.setShader(new LinearGradient(0, 0, w, h, 0x80B9D6FF, 0x80D7E8FF,
    Shader.TileMode.CLAMP));
    Path wavePath1 = new Path();
    wavePath1.moveTo(0, h * 0.82f); // היה 0.75
    wavePath1.quadTo(w * 0.3f, h * 0.78f, w * 0.5f, h * 0.80f);
    wavePath1.cubicTo(w * 0.7f, h * 0.85f, w * 0.9f, h * 0.79f, w, h * 0.76f);
    wavePath1.lineTo(w, h);
    wavePath1.lineTo(0, h);
    wavePath1.close();
    canvas.drawPath(wavePath1, paint);

    // גל שני – כחול בהיר יותר
    paint.setShader(new LinearGradient(0, 0, w, h, 0x80B4E6FF, 0x80C8F1FF,
    Shader.TileMode.CLAMP));
    Path wavePath2 = new Path();
    wavePath2.moveTo(0, h * 0.85f); // היה 0.78
    wavePath2.quadTo(w * 0.4f, h * 0.81f, w * 0.6f, h * 0.83f);
    wavePath2.cubicTo(w * 0.75f, h * 0.85f, w * 0.85f, h * 0.78f, w, h * 0.76f);
    wavePath2.lineTo(w, h);
    wavePath2.lineTo(0, h);
    wavePath2.close();
    canvas.drawPath(wavePath2, paint);

    // גל שלישי – שמנת בהירה
    paint.setShader(new LinearGradient(0, 0, w, h, 0x60FFF5C1, 0x60FFE8B5,
    Shader.TileMode.CLAMP));
    Path wavePath3 = new Path();
    wavePath3.moveTo(0, h * 0.89f); // היה 0.83
    wavePath3.quadTo(w * 0.2f, h * 0.87f, w * 0.4f, h * 0.88f);
    wavePath3.cubicTo(w * 0.6f, h * 0.89f, w * 0.8f, h * 0.87f, w, h * 0.85f);
    wavePath3.lineTo(w, h);

```

```
    wavePath3.lineTo(0, h);  
    wavePath3.close();  
    canvas.drawPath(wavePath3, paint);  
}  
}
```

## AdminOrderAdapter

```
package com.shop.bagrutproject.adapters;

import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Order;

import java.util.List;
import java.util.Locale;

public class AdminOrderAdapter extends
RecyclerView.Adapter<AdminOrderAdapter.AdminOrderViewHolder> {
    private List<Order> orders;
    private boolean isArchived;

    public AdminOrderAdapter(List<Order> orders) {
        this(orders, false);
    }

    public AdminOrderAdapter(List<Order> orders, boolean isArchived) {
        this.orders = orders;
        this.isArchived = isArchived;
    }
}
```

```

@NonNull
@Override
public AdminOrderViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
    View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_admin_order,
parent, false);
    return new AdminOrderViewHolder(view);
}

@Override
public void onBindViewHolder(@NonNull AdminOrderViewHolder holder, int
position) {
    Order order = orders.get(position);
    String status = order.getStatus().toLowerCase();
    String displayStatus;

    holder.orderId.setText("מספר הזמנה: " + order.getId());
    holder.totalPrice.setText(String.format(Locale.getDefault(), "סה"כ: ₪%.2f",
order.getTotalPrice()));

    switch (status) {
        case "pending":
            displayStatus = "ממתינה לאישור";
            holder.status.setTextColor(Color.parseColor("#FFA500"));
            break;
        case "processing":
            displayStatus = "בהכנה";
            holder.status.setTextColor(Color.parseColor("#800080"));
            break;
        case "shipped":
            displayStatus = "נשלחה";
            holder.status.setTextColor(Color.parseColor("#007bff"));
            break;
        case "delivered":
            displayStatus = "סופקה";
    }
}

```

```

        holder.status.setTextColor(Color.parseColor("#28a745"));
        break;
    case "cancelled":
        displayStatus = "בוטלה";
        holder.status.setTextColor(Color.parseColor("#dc3545"));
        break;
    default:
        displayStatus = "לא ידוע";
        holder.status.setTextColor(Color.BLACK);
        break;
}
holder.status.setText("סטטוס: " + displayStatus);

if (!isArchived) {
    // בפתח ניהול במצב רגיל
    holder.btnProcessing.setVisibility(status.equals("pending") ?
View.VISIBLE : View.GONE);
    holder.btnShip.setVisibility(status.equals("processing") ? View.VISIBLE :
View.GONE);
    holder.btnCancel.setVisibility(status.equals("pending") ||
status.equals("processing") ? View.VISIBLE : View.GONE);
    holder.btnRestore.setVisibility(View.GONE);

    holder.btnProcessing.setOnClickListener(v -> {
        order.setStatus("processing");
        updateOrderStatus(order, holder);
    });

    holder.btnShip.setOnClickListener(v -> {
        order.setStatus("shipped");
        updateOrderStatus(order, holder);
    });

    holder.btnCancel.setOnClickListener(v -> {
        new AlertDialog.Builder(holder.itemView.getContext())
            .setTitle("אישור ביטול הזמנה")
            .setMessage("האם אתה בטוח שברצונך לבטל את ההזמנה?")
            .setPositiveButton("כן") { dialog, which -> {
                order.setStatus("cancelled");
                updateOrderStatus(order, holder);
            }}
            .setNegativeButton("לא") { dialog, which -> {
            }}
            .show();
    });
}

```



```

        לשחזר פעולה זו")
        .setPositiveButton("בטל הזמנה", (dialog, which) -> {
            order.setStatus("cancelled");
            updateOrderStatus(order, holder);
        })
        .setNegativeButton("חזור", null)
        .show();
    });

    if (status.equals("delivered")) {
        holder.archiveButton.setVisibility(View.VISIBLE);
        holder.archiveButton.setOnClickListener(v -> {
            new AlertDialog.Builder(v.getContext())
                .setTitle("העבר לארכיון")
                .setMessage("האם אתה בטוח שברצונך להעביר את ההזמנה "
לארכיון?")
                .setPositiveButton("כן", (dialog, which) -> {
                    FirebaseDatabase database =
FirebaseDatabase.getInstance();
                    String orderId = order.getOrderId();

                    database.getReference("archivedOrders").child(orderId).setValue(order)
                        .addOnSuccessListener(aVoid -> {

                            database.getReference("orders").child(orderId).removeValue()
                                .addOnSuccessListener(aVoid1 -> {
                                    orders.remove(position);
                                    notifyItemRemoved(position);
                                    Toast.makeText(v.getContext(), "ההזמנה
הועברה לארכיון", Toast.LENGTH_SHORT).show();
                                });
                            });
                    });
                .setNegativeButton("ביטול", null)
                .show();
            });
    });

```

```

    } else {
        holder.archiveButton.setVisibility(View.GONE);
    }
} else {
    //במצב ארכיון - הסתרת כפתורי ניהול, והצגת כפתור שחזור
    holder.btnProcessing.setVisibility(View.GONE);
    holder.btnShip.setVisibility(View.GONE);
    holder.btnCancel.setVisibility(View.GONE);
    holder.archiveButton.setVisibility(View.GONE);
    holder.btnRestore.setVisibility(View.VISIBLE);

    holder.btnRestore.setOnClickListener(v -> {
        new AlertDialog.Builder(v.getContext())
            .setTitle("שחזור הזמנה")
            .setMessage("האם אתה בטוח שברצונך לשחזר את ההזמנה")
            .setPositiveButton("כן", (dialog, which) -> {
                FirebaseDatabase database = FirebaseDatabase.getInstance();
                String orderId = order.getOrderId();

                database.getReference("orders").child(orderId).setValue(order)
                    .addOnSuccessListener(aVoid -> {

database.getReference("archivedOrders").child(orderId).removeValue()
                    .addOnSuccessListener(aVoid1 -> {
                        orders.remove(position);
                        notifyItemRemoved(position);
                        Toast.makeText(v.getContext(), "ההזמנה שוחזרה",
Toast.LENGTH_SHORT).show();
                    });
                });
            })
            .setNegativeButton("ביטול", null)
            .show();
        });
    }
}

```

```

private void updateOrderStatus(Order order, AdminOrderViewHolder
holder) {
    FirebaseDatabase.getInstance().getReference("orders")
        .child(order.getId())
        .setValue(order)
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Toast.makeText(holder.itemView.getContext(), "-הסטטוס עודכן ל-"
+ order.getStatus(), Toast.LENGTH_SHORT).show();
                notifyItemChanged(holder.getAdapterPosition());
            } else {
                Toast.makeText(holder.itemView.getContext(), "שגיאה בעדכון
הסטטוס", Toast.LENGTH_SHORT).show();
            }
        });
}

```

```

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

```

```

public static class AdminOrderViewHolder extends RecyclerView.ViewHolder
{
    TextView orderId, totalPrice, status;
    Button btnProcessing, btnShip, btnCancel;
    Button archiveButton, btnRestore;

    public AdminOrderViewHolder(View itemView) {
        super(itemView);
        orderId = itemView.findViewById(R.id.orderId);
        totalPrice = itemView.findViewById(R.id.totalPrice);
        status = itemView.findViewById(R.id.status);
        btnProcessing = itemView.findViewById(R.id.btnProcessing);
        btnShip = itemView.findViewById(R.id.btnShip);
        btnCancel = itemView.findViewById(R.id.btnCancel);
        archiveButton = itemView.findViewById(R.id.archiveButton);
    }
}

```

```
        btnRestore = itemView.findViewById(R.id.btnRestore); // ודא שהכפתור
XML-הזה קיים ב
    }
}
}
```

## CartAdapter

```
package com.shop.bagrutproject.adapters;

import android.content.Context;
import android.graphics.Color;
import android.graphics.Paint;
import android.text.SpannableString;
import android.text.Spanned;
import android.text.style.ForegroundColorSpan;
import android.text.style.StrikethroughSpan;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.Nullable;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.ImageUtil;

import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CartAdapter extends BaseAdapter {

    public interface OnCartClick {
        void onItemCheckedChanged(int position, boolean isChecked);
    }
```

```

}

private final Context context;
private final List<Item> cartItems;
private final boolean showCheckbox;
@Nullable
private final OnCartClick onCartClick;
private final DatabaseService databaseService;

Map<String, Deal> deals = new HashMap<>(); // item type -> deal

public CartAdapter(Context context, List<Item> cartItems, @Nullable
OnCartClick onCartClick, boolean showCheckbox) {
    this.context = context;
    this.cartItems = cartItems;
    this.onCartClick = onCartClick;
    this.showCheckbox = showCheckbox;
    this.databaseService = DatabaseService.getInstance();
}

@Override
public int getCount() {
    return cartItems.size();
}

@Override
public Object getItem(int position) {
    return cartItems.get(position);
}

@Override
public long getItemId(int position) {
    return position;
}

@Override

```

```

public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
LayoutInflater.from(context).inflate(R.layout.cart_item_layout, parent, false);
    }

    final Item item = cartItems.get(position);

    TextView itemName = convertView.findViewById(R.id.itemName);
    TextView itemPrice = convertView.findViewById(R.id.itemPrice);
    ImageView itemImage = convertView.findViewById(R.id.itemImage);
    CheckBox deleteCheckBox =
convertView.findViewById(R.id.deleteCheckBox);

    itemName.setText(item.getName());
    itemPrice.setText("₪" + item.getPrice());

    if (item.getPic() != null && !item.getPic().isEmpty()) {
itemImage.setImageBitmap(ImageUtil.convertFrom64base(item.getPic()));
    } else {
        itemImage.setImageResource(R.drawable.ic_launcher_foreground);
    }

    // שליטה על הנראות של CheckBox
    if (showCheckbox) {
        deleteCheckBox.setVisibility(View.VISIBLE);
    } else {
        deleteCheckBox.setVisibility(View.GONE);
    }

    // ניתוק מאזין קודם
    deleteCheckBox.setOnCheckedChangeListener(null);

    // הגדרה מחדש
    deleteCheckBox.setChecked(false); // או שתעדכן לפי המצב האמיתי אם צריך
}

```

```

deleteCheckBox.setOnCheckedChangeListener((buttonView, isChecked) ->
{
    if (onCartClick != null) {
        onCartClick.onItemCheckedChanged(position, isChecked);
    }
});

// עדכון המחיר עם הנחה אם יש
updatePriceWithDeal(item, convertView);

return convertView;
}

```

```

private void updatePriceWithDeal(Item item, View convertView) {
    TextView oldPriceTextView =
convertView.findViewById(R.id.oldPriceTextView);

    Deal deal = this.deals.get(item.getType());
    if (deal == null) {
        oldPriceTextView.setVisibility(View.GONE); // הנחה אין
        return;
    }

```

```

    TextView itemPrice = convertView.findViewById(R.id.itemPrice);

```

```

    double discount = deal.getDiscountPercentage();
    double finalPrice = item.getPrice() * (1 - discount / 100);
    double originalPrice = item.getPrice(); // נשמור את המחיר המקורי

```

```

// הצגת המחיר לאחר הנחה
itemPrice.setText("₪" + finalPrice);

```

```

// אם יש הנחה, נציג את המחיר המקורי עם קו חוצה
oldPriceTextView.setVisibility(View.VISIBLE); // הראה את המחיר המקורי

```



```

        SpannableString spannableString = new SpannableString("₪" +
originalPrice);

        // הוסף קו חוצה
        spannableString.setSpan(new StrikethroughSpan(), 0,
spannableString.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

        // הוסף קו אדום
        spannableString.setSpan(new ForegroundColorSpan(Color.RED), 0,
spannableString.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

        oldPriceTextView.setText(spannableString); // הצגת המחיר הישן עם קו אדום

    }

    public void setItems(List<Item> items) {
        this.cartItems.clear();
        this.cartItems.addAll(items);
        this.notifyDataSetChanged();
    }

    public void removeItem(int position) {
        this.cartItems.remove(position);
        this.notifyDataSetChanged();
    }

    public void setDeals(Collection<Deal> deals) {
        this.deals.clear();
        for (Deal deal : deals) {
            this.deals.put(deal.getItemType(), deal);
        }
        notifyDataSetChanged();
    }
}

```

## CategoryAdapter

```
package com.shop.bagrutproject.adapters;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Category;
import com.shop.bagrutproject.screens.ShopActivity;

import java.util.List;

public class CategoryAdapter extends BaseAdapter {
    private Context context;
    private List<Category> categories;

    public CategoryAdapter(Context context, List<Category> categories) {
        this.context = context;
        this.categories = categories;
    }

    @Override
    public int getCount() {
        return categories.size();
    }

    @Override
    public Object getItem(int position) {
        return categories.get(position);
    }
```

```

    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Category category = categories.get(position);
        if (convertView == null) {
            convertView =
LayoutInflater.from(context).inflate(R.layout.category_item, parent, false);

            ImageView image = convertView.findViewById(R.id.categoryImage);
            TextView title = convertView.findViewById(R.id.categoryTitle);

            image.setImageResource(category.getImageResId());
            title.setText(category.getName());

            //ShopActivity-כאשר לוחצים על הקטגוריה, מעבירים את שם הקטגוריה ל
            convertView.setOnClickListener(v -> {
                Intent intent = new Intent(context, ShopActivity.class);
                intent.putExtra("category", category.getName()); //שליחת שם הקטגוריה
                context.startActivity(intent);
            });

            return convertView;
        }
    }
}

```

## CommentAdapter

```
package com.shop.bagrutproject.adapters;

import android.app.AlertDialog;
import android.content.Context;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Comment;
import com.shop.bagrutproject.services.AuthenticationService;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

import java.util.List;

public class CommentAdapter extends
RecyclerView.Adapter<CommentAdapter.CommentViewHolder> {

    private List<Comment> commentList;
    private DatabaseReference commentsRef;
    private Context context;
```

```

private String itemId;

public CommentAdapter(Context context, List<Comment> commentList,
String itemId) {
    this.context = context;
    this.commentList = commentList;
    this.itemId = itemId;
    this.commentsRef =
FirebaseDatabase.getInstance().getReference("comments").child(itemId);
}

@NonNull
@Override
public CommentViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
    View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_comment,
parent, false);
    return new CommentViewHolder(view);
}

@Override
public void onBindViewHolder(@NonNull CommentViewHolder holder, int
position) {
    Comment comment = commentList.get(position);

    holder.commentText.setText(comment.getCommentText());
    holder.ratingBar.setRating(comment.getRating());
    holder.ratingBar.setIsIndicator(true);
    holder.userName.setText(comment.getUserName());

    if (comment.getCommentId() == null) {
        Log.e("CommentAdapter", "commentId is null at position " + position);
    } else {
        Log.d("CommentAdapter", "commentId: " + comment.getCommentId());
    }
}

```

```

        holder.itemView.setOnLongClickListener(v -> {
            String currentUserId =
AuthenticationService.getInstance().getCurrentUserId();
            if (comment.getUserId().equals(currentUserId) ||
SharedPreferencesUtil.isAdmin(context)) {
                showDeleteConfirmationDialog(comment, position);
            } else {
                Toast.makeText(context, "לא ניתן למחוק תגובה של משתמש אחר",
Toast.LENGTH_SHORT).show();
            }
            return true;
        });
    }
}

```

```

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

```

```

public static class CommentViewHolder extends RecyclerView.ViewHolder {
    TextView commentText;
    RatingBar ratingBar;
    TextView userName;
}

```

```

public CommentViewHolder(View itemView) {
    super(itemView);
    commentText = itemView.findViewById(R.id.commentText);
    ratingBar = itemView.findViewById(R.id.ratingBar);
    userName = itemView.findViewById(R.id.userName);
}
}

```

```

private void showDeleteConfirmationDialog(Comment comment, int
position) {
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
}

```

```

        builder.setTitle("מחיקת תגובה");
        builder.setMessage("האם אתה בטוח שברצונך למחוק את התגובה?");
        builder.setPositiveButton("כן", (dialog, which) ->
deleteComment(comment, position));
        builder.setNegativeButton("לא", (dialog, which) -> dialog.dismiss());
        builder.show();
    }

    private void deleteComment(Comment comment, int position) {
        if (position < 0 || position >= commentList.size()) {
            Log.e("CommentAdapter", "Invalid index: " + position);
            return;
        }

        DatabaseService.getInstance().removeComment(itemId,
comment.getCommentId(), new DatabaseService.DatabaseCallback<Void>() {
            @Override
            public void onCompleted(Void object) {
                Log.d("CommentAdapter", "Comment deleted successfully from
Firebase.");

                // RecyclerView-מחיקה מהרשימה המקומית ועדכון ה
                commentList.remove(position);
                notifyItemRemoved(position);
                notifyItemRangeChanged(position, commentList.size());
            }

            @Override
            public void onFailed(Exception e) {
                Log.e("CommentAdapter", "Failed to delete comment", e);
            }
        });
    }
}

```

## DealsAdapter

```
package com.shop.bagrutproject.adapters;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Deal;

import java.util.List;

public class DealsAdapter extends
    RecyclerView.Adapter<DealsAdapter.DealViewHolder> {

    private List<Deal> dealsList;

    public DealsAdapter(List<Deal> dealsList) {
        this.dealsList = dealsList;
    }

    @Override
    public DealViewHolder onCreateViewHolder(ViewGroup parent, int
        viewType) {
        View view =
            LayoutInflater.from(parent.getContext()).inflate(R.layout.item_deal, parent,
                false);
        return new DealViewHolder(view);
    }

    @Override
    public void onBindViewHolder(DealViewHolder holder, int position) {
```



```

        Deal deal = dealsList.get(position);
        holder.titleTextView.setText(deal.getTitle());
        holder.descriptionTextView.setText(deal.getDescription());
        holder.discountTextView.setText("הנחה: " + deal.getDiscountPercentage()
+ "%");
        holder.validUntilTextView.setText("תוקף עד: " + deal.getValidUntil());
        holder.typeTextView.setText("סוג: " + deal.getItemType());
    }

```

```

@Override

```

```

public int getItemCount() {
    return dealsList.size();
}

```

```

public static class DealViewHolder extends RecyclerView.ViewHolder {

```

```

    TextView titleTextView;
    TextView descriptionTextView;
    TextView discountTextView;
    TextView validUntilTextView;
    TextView typeTextView;

```

```

    public DealViewHolder(View itemView) {
        super(itemView);
        titleTextView = itemView.findViewById(R.id.dealTitle);
        descriptionTextView = itemView.findViewById(R.id.dealDescription);
        discountTextView = itemView.findViewById(R.id.dealDiscount);
        validUntilTextView = itemView.findViewById(R.id.dealValidUntil);
        typeTextView = itemView.findViewById(R.id.dealType);
    }
}

```

## ItemsAdapter

```
package com.shop.bagrutproject.adapters;

import static android.content.Intent.getIntent;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Paint;
import android.text.SpannableString;
import android.text.Spanned;
import android.text.style.ForegroundColorSpan;
import android.text.style.StrikethroughSpan;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.RatingBar;
import android.widget.TextView;
import androidx.annotation.Nullable;
import androidx.recyclerview.widget.RecyclerView;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;
import com.shop.bagrutproject.screens.ItemDetailActivity;
import com.shop.bagrutproject.screens.ShopActivity;
import com.shop.bagrutproject.services.DatabaseService;
import com.shop.bagrutproject.utils.ImageUtil;
import com.shop.bagrutproject.utils.SharedPreferencesUtil;

import java.util.ArrayList;
import java.util.List;
```

```

public class ItemsAdapter extends
RecyclerView.Adapter<ItemsAdapter.ItemViewHolder> {

    private static final String TAG = "ItemsAdapter";
    private List<Item> originalItemsList;
    private List<Item> filteredItemsList;
    private Context context;
    private DatabaseService databaseService;

    public interface ItemClickListener {
        void onClick(Item item);
    }

    @Nullable
    private final ItemClickListener itemClickListener;

    public ItemsAdapter(List<Item> itemsList, Context context, @Nullable
ItemClickListener itemClickListener) {
        this.originalItemsList = itemsList;
        this.filteredItemsList = new ArrayList<>(itemsList);
        this.context = context;
        this.itemClickListener = itemClickListener;
        this.databaseService = DatabaseService.getInstance();
    }

    @Override
    public ItemViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.itemselected,
parent, false);
        return new ItemViewHolder(view);
    }

    @Override
    public void onBindViewHolder(ItemViewHolder holder, int position) {
        Item item = filteredItemsList.get(position);

```

```

        holder.bindItem(item);
    }

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

    public class ItemViewHolder extends RecyclerView.ViewHolder {
        private ImageView previewImageView;
        private TextView previewTextView, previewPriceTextView,
previewDescriptionTextView, oldPriceTextView;
        private RatingBar previewRatingBar;
        private ImageButton addToCartButton;
        private String itemId;
        private TextView dealtag;

        public ItemViewHolder(View itemView) {
            super(itemView);
            previewImageView = itemView.findViewById(R.id.PreviewimageView);
            previewTextView = itemView.findViewById(R.id.PreviewtextView);
            previewPriceTextView =
itemView.findViewById(R.id.PreviewPriceTextView);
            previewDescriptionTextView =
itemView.findViewById(R.id.PreviewDescriptionTextView);
            previewRatingBar = itemView.findViewById(R.id.PreviewRatingBar);
            addToCartButton = itemView.findViewById(R.id.addToCartButton);
            oldPriceTextView = itemView.findViewById(R.id.oldPriceTextView); //
הוספת טקסט למחיר ישן עם קו חוצה
            dealtag = itemView.findViewById(R.id.saleTag);

            itemView.setOnClickListener(v -> {
                Item item = filteredItemsList.get(getAdapterPosition());
                Intent intent = new Intent(context, ItemDetailActivity.class);
                intent.putExtra("itemId", item.getId());
                context.startActivity(intent);
            });

```

```

        // הצגת כפתור הוספה לעגלה רק אם לא מדובר במנהל
        if (SharedPreferencesUtil.isAdmin(context)) {
            addToCartButton.setVisibility(View.GONE);
        } else {
            addToCartButton.setVisibility(View.VISIBLE);
        }
    }
}

public void bindItem(final Item item) {

    previewImageView.setImageBitmap(ImageUtil.convertFrom64base(item.getPic
()));
    previewTextView.setText(item.getName());
    previewDescriptionTextView.setText(item.getAboutItem());

    // הצגת המחיר לאחר הנחה
    updatePriceWithDeal(item);

    itemId = item.getId();
    updateAverageRating(previewRatingBar, itemId);

    addToCartButton.setOnClickListener(v -> {
        if (itemClickListener != null)
            itemClickListener.onClick(item);
    });
}

private void updatePriceWithDeal(Item item) {
    databaseService.getAllDeals(new
DatabaseService.DatabaseCallback<List<Deal>>() {
        @Override
        public void onCompleted(List<Deal> deals) {
            double finalPrice = item.getPrice();
            double originalPrice = item.getPrice(); // נשמור את המחיר המקורי
            boolean hasDiscount = false;

```

```

for (Deal deal : deals) {
    if (deal.isValid() && deal.getItemType().equals(item.getType())) {
        double discount = deal.getDiscountPercentage();
        finalPrice = item.getPrice() * (1 - discount / 100);
        hasDiscount = true;
        break;
    }
}

// הצגת המחיר המעודכן
previewPriceTextView.setText("₪" + finalPrice);

// אם יש הנחה, נציג את המחיר המקורי עם קו חוצה
if (hasDiscount) {
    oldPriceTextView.setVisibility(View.VISIBLE); // הראה את המחיר המקורי
    SpannableString spannableString = new SpannableString("₪" +
originalPrice);

    // הוסף קו חוצה
    spannableString.setSpan(new StrikethroughSpan(), 0,
spannableString.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

    // הוסף קו אדום
    spannableString.setSpan(new ForegroundColorSpan(Color.RED),
0, spannableString.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

    oldPriceTextView.setText(spannableString); // הצגת המחיר הישן
    dealtag.setVisibility(View.VISIBLE);
} else {
    oldPriceTextView.setVisibility(View.GONE); // הסתר אם אין הנחה
    dealtag.setVisibility(View.GONE);
}
}

```

```

        @Override
        public void onFailed(Exception e) {
            Log.e(TAG, "Failed to fetch deals", e);
        }
    });
}

private void updateAverageRating(RatingBar itemAverageRatingBar, String
itemId) {
    databaseService.updateAverageRating(itemId, new
DatabaseService.DatabaseCallback<Double>() {
        @Override
        public void onCompleted(Double averageRating) {
            itemAverageRatingBar.setRating(averageRating.floatValue());
        }

        @Override
        public void onFailed(Exception e) {
            Log.e(TAG, "Failed to fetch average rating", e);
        }
    });
}

public void filter(String query) {
    filteredItemsList.clear();
    if (query.isEmpty()) {
        filteredItemsList.addAll(originalItemsList);
    } else {
        String lowerCaseQuery = query.toLowerCase();
        try {
            double queryPrice = Double.parseDouble(query);
            for (Item item : originalItemsList) {
                if (item.getPrice() == queryPrice) {
                    filteredItemsList.add(item);
                }
            }
        }
    }
}

```

```

    }
} catch (NumberFormatException e) {
    for (Item item : originalItemsList) {
        if (item.getName().toLowerCase().contains(lowerCaseQuery) ||
            item.getCompany().toLowerCase().contains(lowerCaseQuery)
||
            item.getType().toLowerCase().contains(lowerCaseQuery) ||
            item.getColor().toLowerCase().contains(lowerCaseQuery)) {
            filteredItemsList.add(item);
        }
    }
}
}
notifyDataSetChanged();
}
}

```



## OrderAdapter

```
package com.shop.bagrutproject.adapters;

import android.app.AlertDialog;
import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.FirebaseDatabase;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Order;

import java.util.List;
import java.util.Locale;

public class OrderAdapter extends
RecyclerView.Adapter<OrderAdapter.OrderViewHolder> {
    private List<Order> orders;

    public OrderAdapter(List<Order> orders) {
        this.orders = orders;
    }

    @Override
    public OrderViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_order, parent,
false);
```

```

        return new OrderViewHolder(view);
    }

    @Override
    public void onBindViewHolder(OrderViewHolder holder, int position) {
        Order order = orders.get(position);
        holder.orderId.setText("מספר הזמנה: " + order.getId());
        holder.totalPrice.setText(String.format(Locale.getDefault(), "סה"ם: ₪%.2f",
            order.getTotalPrice()));
        holder.date.setText("תאריך: " + order.getFormattedDate());

        String status = order.getStatus().toLowerCase();
        String displayStatus = "";

        switch (status) {
            case "pending":
                displayStatus = "ממתינה לאישור";
                holder.status.setTextColor(Color.parseColor("#FFA500")); // בתם
                break;
            case "processing":
                displayStatus = "בהכנה";
                holder.status.setTextColor(Color.parseColor("#800080")); // סגול
                break;
            case "shipped":
                displayStatus = "נשלחה";
                holder.status.setTextColor(Color.parseColor("#007bff")); // כחול
                break;
            case "delivered":
                displayStatus = "סופקה";
                holder.status.setTextColor(Color.parseColor("#28a745")); // יחק
                break;
            case "cancelled":
                displayStatus = "בוטלה";
                holder.status.setTextColor(Color.parseColor("#dc3545")); // אדום
                break;
            default:
                displayStatus = "לא ידוע";
        }
    }

```

```

        holder.status.setTextColor(Color.BLACK);
        break;
    }
    holder.status.setText("סטטוס: " + displayStatus);

    if (status.equals("shipped")) {
        holder.deliverButton.setVisibility(View.VISIBLE);
        holder.deliverButton.setOnClickListener(v -> {
            new AlertDialog.Builder(v.getContext())
                .setTitle("אישור קבלת הזמנה")
                .setMessage("האם אתה בטוח שקיבלת את ההזמנה?")
                .setPositiveButton("כן", (dialog, which) -> {
                    FirebaseDatabase.getInstance().getReference("orders")
                        .child(order.getId())
                        .child("status")
                        .setValue("delivered")
                        .addOnCompleteListener(task -> {
                            if (task.isSuccessful()) {
                                Toast.makeText(v.getContext(), "ההזמנה סומנה  
כהושלמה", Toast.LENGTH_SHORT).show();
                                order.setStatus("delivered");
                                notifyItemChanged(position);
                            } else {
                                Toast.makeText(v.getContext(), "שגיאה בהשלמת  
ההזמנה", Toast.LENGTH_SHORT).show();
                            }
                        });
                })
                .setNegativeButton("לא", null)
                .show();
        });
    } else {
        holder.deliverButton.setVisibility(View.GONE);
    }
}

```

```

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

public static class OrderViewHolder extends RecyclerView.ViewHolder {
    TextView orderId, totalPrice, status, date;
    Button deliverButton; // כפתור שינוי הסטטוס

    public OrderViewHolder(View itemView) {
        super(itemView);
        orderId = itemView.findViewById(R.id.orderId);
        totalPrice = itemView.findViewById(R.id.totalPrice);
        status = itemView.findViewById(R.id.status);
        date = itemView.findViewById(R.id.date);
        deliverButton = itemView.findViewById(R.id.deliverButton); // כפתור
        השלמת ההזמנה
    }
}
}

```

## PaymentAdapter

```
package com.shop.bagrutproject.adapters;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.Deal;
import com.shop.bagrutproject.models.Item;

import java.util.List;

public class PaymentAdapter extends
RecyclerView.Adapter<PaymentAdapter.ViewHolder> {
    private List<Item> items;
    private List<Deal> deals;

    public PaymentAdapter(List<Item> items, List<Deal> deals) {
        this.items = items;
        this.deals = deals;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_payment,
parent, false);
        return new ViewHolder(view);
    }
}
```

```

    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Item item = items.get(position);
        double price = item.getPrice();
        for (Deal deal : deals) {
            if (deal.isValid() && item.getType().equals(deal.getItemType())) {
                price = price * (1 - deal.getDiscountPercentage() / 100);
                break;
            }
        }
        holder.name.setText(item.getName());
        holder.price.setText("₹" + String.format("%.2f", price));
    }

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

    class ViewHolder extends RecyclerView.ViewHolder {
        TextView name, price;

        ViewHolder(View itemView) {
            super(itemView);
            name = itemView.findViewById(R.id.itemName);
            price = itemView.findViewById(R.id.itemPrice);
        }
    }
}

```

## UsersAdapter

```
package com.shop.bagrutproject.adapters;

import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.shop.bagrutproject.R;
import com.shop.bagrutproject.models.User;
import com.shop.bagrutproject.screens.UserDetailActivity;
import com.shop.bagrutproject.services.DatabaseService;
import java.util.ArrayList;
import java.util.List;

public class UsersAdapter extends
RecyclerView.Adapter<UsersAdapter.UsersViewHolder> {

    private List<User> usersList;
    private List<User> filteredList;
    private Context context;
    private static final String TAG = "UsersAdapter";

    public UsersAdapter(List<User> usersList, Context context) {
        this.usersList = usersList;
        this.filteredList = new ArrayList<>(usersList);
        this.context = context;
    }
}
```

```

@NonNull
@Override
public UsersViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
    View view = LayoutInflater.from(context).inflate(R.layout.item_user,
parent, false);
    return new UsersViewHolder(view);
}

```

```

@Override
public void onBindViewHolder(@NonNull UsersViewHolder holder, int
position) {
    User user = filteredList.get(position);
    holder.nameTextView.setText(user.getfName() + " " + user.getlName());

```

```

holder.itemView.setOnClickListener(v -> {
    Intent intent = new Intent(context, UserDetailsActivity.class);
    intent.putExtra("USER_UID", user.getId());
    intent.putExtra("USER_FNAME", user.getfName());
    intent.putExtra("USER_LNAME", user.getlName());
    intent.putExtra("USER_EMAIL", user.getEmail());
    intent.putExtra("USER_PHONE", user.getPhone());
    context.startActivity(intent);
});

```

```

holder.itemView.setOnLongClickListener(v -> {
    if (user.getId() != null) {
        new AlertDialog.Builder(context)
            .setTitle("Confirm Delete")
            .setMessage("Are you sure you want to delete this user?")
            .setPositiveButton("Yes", (dialog, which) -> {
                deleteUserAndRefresh(user.getId(), position);
            })
            .setNegativeButton("No", null)
            .show();
    } else {
        Toast.makeText(context, "Cannot delete user: UID is null",

```



```

Toast.LENGTH_SHORT).show();
    }
    return true;
});
}

public void filter(String query) {
    filteredList.clear();
    if (query.isEmpty()) {
        filteredList.addAll(usersList);
    } else {
        for (User user : usersList) {
            String fullName = user.getfName() + " " + user.getlName();
            if (fullName.toLowerCase().contains(query.toLowerCase())) {
                filteredList.add(user);
            }
        }
    }
    notifyDataSetChanged();
}

private void deleteUserAndRefresh(String uid, int position) {
    if (uid == null || uid.isEmpty()) {
        Log.e(TAG, "Cannot delete user: uid is null or empty");
        Toast.makeText(context, "Cannot delete user: uid is null or empty",
Toast.LENGTH_SHORT).show();
        return;
    }

    Log.d(TAG, "Trying to delete user with uid: " + uid);
    DatabaseService.getInstance().deleteUser(uid, new
DatabaseService.DatabaseCallback<Void>() {
        @Override
        public void onCompleted(Void object) {
            Log.d(TAG, "User deleted successfully");
            usersList.remove(position);
            filter("");
        }
    });
}

```

```

    }

    @Override
    public void onFailed(Exception e) {
        Log.e(TAG, "Failed to delete user: " + e.getMessage());
        Toast.makeText(context, "Failed to delete user",
Toast.LENGTH_SHORT).show();
    }
});
}

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

public static class UsersViewHolder extends RecyclerView.ViewHolder {
    TextView nameTextView;

    public UsersViewHolder(View itemView) {
        super(itemView);
        nameTextView = itemView.findViewById(R.id.tvName);
    }
}
}

```