מערכת YumYum

אפליקציית מתכונים ואוכל

עבודת גמר תכנון ותכנות מערכות 883599

פיתוח אפליקציות Android – בסביבת Xamarin

מגיש: יאיר לייטוס

ת.ז: 325065340

מנחות: ויויאנה טרנר, דקלה סוקולר

תיכון טשרניחובסקי

תאריך הגשה: 18.05.2020

תוכן עניינים

מבוא - תיאור קצר של האפליקציה

תיאור תכולת הספר

ספר הפרויקט הנ"ל מתאר במפורט את תכולת פרויקט האפליקציה, אופן השמתו, את הליך העבודה וכיצד האפליקציה באה לידי ביטוי.
הספר מכיל כמה נושאים המכילים מידע על האפליקציה ועל אופן הכנתה.
הספר כולל את הפרקים הבאים: מבנה וארכיטקטורת האפליקציה וכיצד הם תורמים לה; מדריך למשתמש אשר מסביר במפורש על המסכים השונים באפליקציה, מה הקשרים ביניהם וכיצד האינטראקציה באה לידי ביטוי; מסד הנתונים שבאפליקציה, מרכיביו וכיצד היא משתמשת בו ולאילו מטרות; מדריך למפתח עם הסבר מפורט על הקבצים השונים שבפרויקט ועל תכנם ורפלקציה שתציג את הליך העבודה והקשיים בה.

הרקע לפרויקט

פרויקט הגמר שלי בתכנון ותכנות מערכות עוסק באפליקציית מתכונים ואוכל אשר מקנה שירותים ואפשרויות הייחודיות לה. הרקע לפרויקט הוא ליצור אפליקציה אשר תהווה אפליקציית שיתוף של מתכונים אך גם תאפשר שמירה של מתכונים כך שהמשתמש יכול לראות אותם בכל רגע נתון.
מכאן הגיע הרעיון של יצירת אפליקציית YumYum אשר מטפלת באותם נושאים ובמהלך העבודה נוספו אפשרויות נוספות אשר מהוות תוכן נוסף.

תהליך המחקר

במהלך עבודתי התבוננתי במספר אפליקציות מצליחות של מתכונים וכיצד הן מעוצבות ומה הן מציעות.
לאחר מכן לקחתי נקודות מסוימות ויישמתי אותן בפרויקט למשל: עיצוב דפי המתכון והדף הראשי, אלמנטים באפשרות עריכת מתכון וכן הפרדה בין המתכונים שהעלה המשתמש בדף אחד וכלל המתכונים של כלל המשתמשים בדף הראשי.
החידושים בפרויקט הם אפשרות לאהוב מתכון ובכך לשמור אותו בדף נפרד המכיל את המתכונים האהובים על המשתמש ואפשרויות סינון וחיפוש מתכונים בדפים השונים.
כמו כן, הפרדת המתכונים ל-4 קטגוריות נרחבות (סלטים, מרקים, מנות בשריות ומאפים) במטרה ליצור הבדל בין המתכונים לצורך קליטת נתונים הייחודיים להם וכן אפשרות נוספת בסינון המתכונים.

אתגרים מרכזיים

נושא הפרויקט שבחרתי הוא אפליקציית מתכונים ואוכל בשם YumYum.

תיאור קצר לבעיה

הבעיה בפרויקט היא העלאת מתכון בידי משתמש לאפליקציה, שמירתו במסד הנתונים והצגתו בדף הבית ובדפים רלוונטיים נוספים.

פתרון הפרויקט והסיבות לבחירתו

האפליקציה מכילה דף ראשי בו יוכלו המשתמשים לראות את כל המתכונים שהועלו, יוכלו ללחוץ על כל מתכון ולראות את המידע הרלוונטי עליו, יוכלו לשמור את המתכונים האהובים עליהם בדף נפרד וכן יוכלו להעלות בעצמם את המתכונים שהם מעוניינים לשתף עם שאר האנשים.

האפליקציה מתאימה לכל האנשים המעוניינים במתכונים טעימים ומיוחדים, אלו שרוצים לגוון את התפריט שלהם ואלו שרוצים לחלוק עם אחרים את המתכון האהוב עליהם. האפליקציה שימושית מאוד מאחר והיא מכילה בתוכה את כל המתכונים הללו, וכן במידה והמשתמשים מעוניינים לשמור את המתכון שלהם, הם מוזמנים לשתף אותו באפליקציה והם יוכלו לגשת אליו בכל רגע נתון בדף נפרד אשר יכיל את כל המתכונים שהעלו.

בחרתי בפרויקט בנושא זה, מאחר ואני רואה בזאת דרך נוחה מאוד לשתף אנשים שחיבתם היא הכנת אוכל, במאגר נרחב של מתכונים שחלקו ימצא חן בעיניהם. מאז ומתמיד אנשים רבים מחפשים מתכונים חדשים שלעתיד יהפכו למנה האהובה עליהם ואפליקציית YumYum באה לתת לכך פתרון, מאחר והיא מהווה כאפליקציה המאחסנת מתכונים רבים וטעימים, אותם מעלים באופן דינמי המשתמשים עצמם.

בתחילת הפרויקט נתקלתי בכמה קשיים. תחילה הייתי צריך להחליט כיצד לעצב את האפליקציה. לאחר מעבר על אפליקציות מתכונים שונות החלטתי שעליי להחליט על גוון צבעים קבוע לאורך האפליקציה ושלפקדים מסוימים יהיה אותו צבע בכל דפי האפליקציה ליצירת אחידות.

לאחר מכן, התקשיתי בתכנון מסד הנתונים של האפליקציה מאחר וישנה הורשה בחלק מהמחלקות המייצגות את טבלאות המסד. לשם כך, הכנתי מחלקת עזר סטטית אשר תטפל בכל הדרישות באפליקציה בנוגע למסד הנתונים. בנוסף לכך, נתקלתי בקשיים בנוגע לService אשר מצד אחד יהיה נוח לשימוש וגם יהווה חלק אינטגרלי באפליקציה. במהלך העבודה התעמקתי בנושא הנ"ל והגעתי למסקנה ששירות הודעות יתאים לאפליקציה ויספק אינטראקציה עם המשתמש.

פתרונות לבעיה

תיאור מטרת המערכת והערך המוסף

מטרת המערכת היא בנייה דינמית של מתכון והעלאתו לדף הראשי. כאשר המשתמש יוצר מתכון הוא נדרש להעלות את תמונת המתכון דרך הגלריה של הטלפון או לצלם את התמונה בעזרת המצלמה. בדף הראשי המשתמש יוכל לחפש את המתכון הרצוי בעזרת חיפוש וסינון התואמים את דרישות המשתמש. בדף המתכון יכולים המשתמשים לסמן את המתכון כאהוב ובתחילת כל מתכון יצוין מספר האנשים שאהבו את המתכון הנ"ל. דף המתכון כולל את כל השדות שמולאו בעת הכנת המתכון, כך שמשתמשים אחרים יוכלו לראות את תכולת המתכון והפרטים הרלוונטיים. המערכת תאפשר עריכה ומחיקה של מתכון אך ורק אם הוא נוצר על ידי המשתמש הנ"ל.

למשתמש גישה לדפים נפרדים הכוללים את כל המתכונים שהעלה וכן כל המתכונים שסימן שאהב.

משתמש שהוא גם מנהל יכול לגשת לדף רשימת המשתמשים שם הוא יכול לראות את פרטי כל המשתמשים באפליקציה ואף למחוק משתמשים.

בנוסף לכך, ישנה אפשרות למתן שירות הודעות אשר יעדכן את המשתמש בפעולות שונות שהוא עושה באפליקציה.

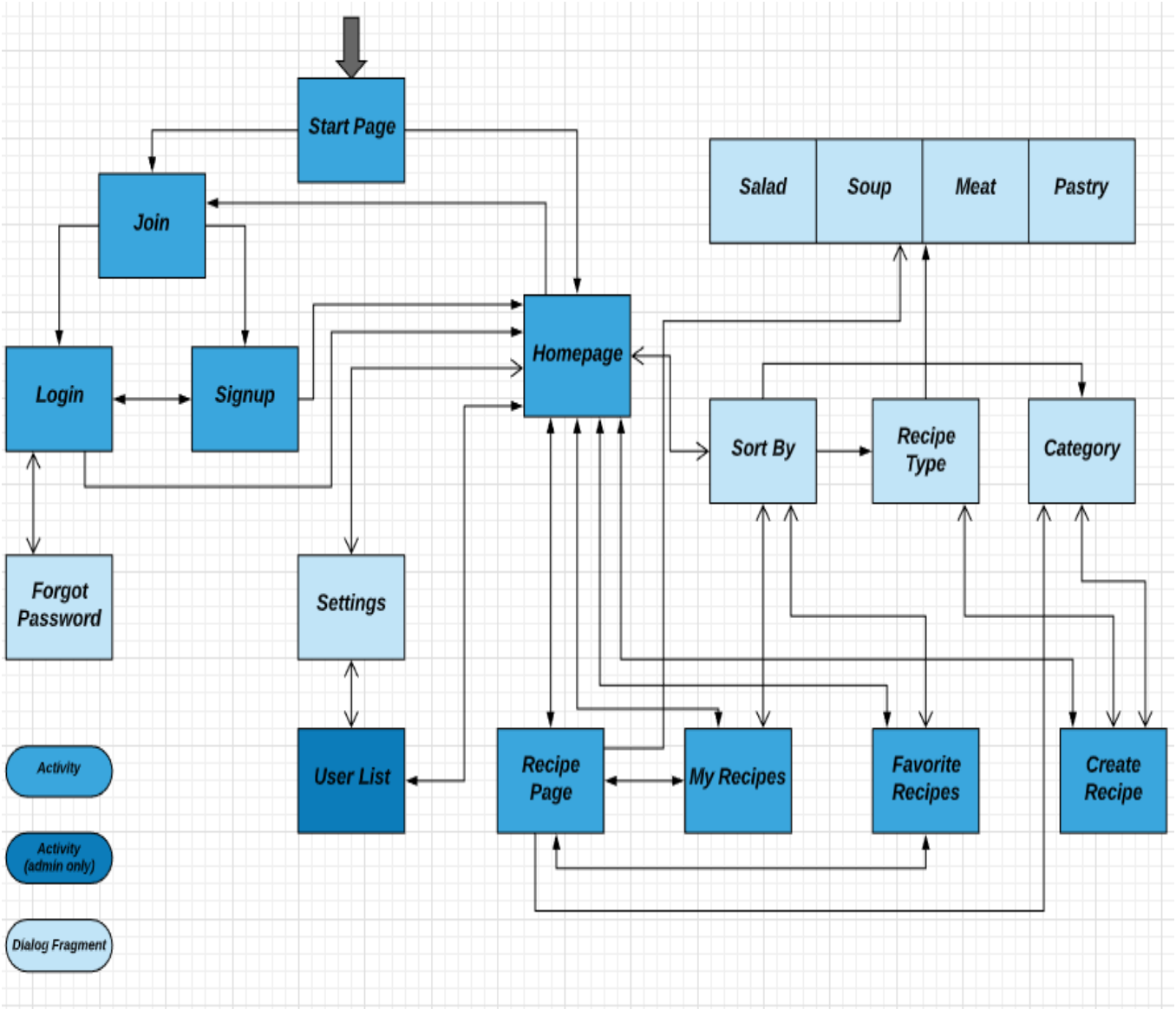
יכולות המערכת

1. העלאת מתכונים לדף הראשי של האפליקציה.
2. העלאת תמונת מתכון ותמונת פרופיל בעזרת שימוש במצלמה או הפנייה לגלריה.
3. הצגת פרטי המתכון הנבחר.
4. חיפוש וסינון מתקדם למציאת מתכונים בדפים השונים.
5. שמירת מתכונים שהועלו על ידי המשתמש.
6. מחיקה ועדכון מתכון שנוצר על ידי המשתמש.
7. הודעה על הרשמה למערכת, התחברות למערכת, העלאת מתכון, עדכון מתכון, מחיקת מתכון בהצלחה ומתן לייק למתכון.
8. האזנה לכמות הסוללה במכשיר ורישום הודעה מתאימה במקרה והסוללה נמוכה.
9. שמירת מתכון שסומן כאהוב ועדכון מספר האנשים שאוהבים את המתכון.
10. הצגת רשימת משתמשים הרשומים לאפליקציה.
11. למנהל גישה לכל המתכונים וביכולתו למחוק ולעדכן כל מתכון וכן למחוק משתמשים ממסד הנתונים.

מבנה / ארכיטקטורה

תרשים זרימה בין המסכים – Screen Flow Diagram

התרשים הבא מציג את המעבר בין המסכים (activity) השונים באפליקציה באמצעות intent באופן המאפשר פעולה תקינה של האפליקציה. התרשים כולל חלונות DialogFragment שנעשה בהם שימוש במסכים השונים לאורך האפליקציה.



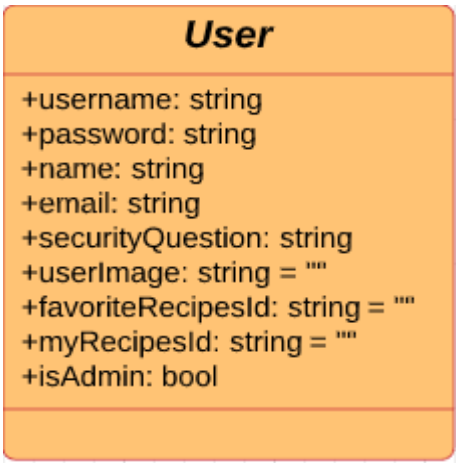
כפי שניתן לראות האפליקציה מכילה דף ראשי, שהוא הדף המרכזי, אשר מכיל את הקישורים לכל דפי האפליקציה. דף הפתיחה מוביל לדף ההצטרפות או לדף הבית זאת בהתאם לסטטוס המשתמש (זכור במערכת או לא). דף ההצטרפות מוביל לדף ההתחברות ולדף ההרשמה. דף הבית מוביל לדפים השונים באפליקציה: דף הכנת מתכון, דף מתכונים אהובים, דף המתכונים שלי ודף רשימת המשתמשים (למנהלים בלבד). כמו כן, לחיצה על מתכון מסוים תפנה לדף פרטי המתכון. ישנו שימוש ב- Dialog Fragments לשם נוחות והצגת טפסי מילוי או תצוגה ללא מעבר לדף אחר, אלא הדף הנוכחי נמצא ברקע.

דפי הפרויקט

הפרויקט כולל דפים שונים לתפעול האפליקציה.
דף הפתיחה הוא הדף הראשון המופיע לאחר פתיחת האפליקציה. הדף בודק האם המשתמש זכור במערכת ולפיכך מפנה לדף ההצטרפות או לדף הבית.
דף ההצטרפות מכיל קישורים לדף ההתחברות ולדף ההרשמה. דף ההרשמה כולל שדות שונים שיש למלא לפי בדיקות התקינות השונות ולאחר המילוי המשתמש מתחבר למערכת ונשאר זכור בה.
דף ההתחברות כולל שתי שדות למילוי שם משתמש וסיסמא ובאפשרות המשתמש לבחור האם להיות זכור במערכת או לא. בשני הדפים לאחר סיום המילוי המשתמש מועבר לדף הבית.
דף הבית כולל בתוכו את כל המתכונים שהועלו לאפליקציה ועושה זאת באמצעות שליפת המתכונים ממסד הנתונים והשמתם ברשימה. ישנה רשימת מתכונים נוספת אשר מתעדכנת לפי אפשרויות החיפוש והסינון ומציגה את המתכונים בדף עצמו, בהתאם למתכונים ברשימת כל המתכונים הכללית.
הדף כולל סרגל חיפוש וטופס לסינון שמאפשרים סינון מתקדם. כמו כן, הדף כולל תפריט עם הקישורים לדפים השונים באפליקציה, קישור לדף ההגדרות וכן אפשרות להתנתקות.
דפי המתכונים האהובים של המשתמש והמתכונים שהמשתמש יצר עובדים על אותו העיקרון, אלא שבתחילה המתכונים מסוננים בהתאמה (מתכונים אהובים בלבד ומתכונים של המשתמש בלבד).
דף הכנת מתכון כולל בתוכו את כל השדות שיש למלא על המתכון ונעשה שימוש ב-Dialog Fragment להצגת טפסים שונים לבחירת פרטים על המתכון.
דף המתכון מופיע כאשר לוחצים על מתכון בדף הבית, בדף המתכונים האהובים ודף המתכונים שלי. הוא מכיל את כל פרטי המתכון כפי שמולאו תחילה בדף הכנת המתכון וניתן לעדכן או למחוק את המתכון דרך הדף הנ"ל במידה והמשתמש המחובר הוא יוצר המתכון או מנהל באפליקציה.
דף רשימת המשתמשים הוא דף שניתן לגשת אליו מדף הבית במידה והמשתמש המחובר הוא גם מנהל. הדף כולל רשומות של כל המשתמשים באפליקציה וניתן לגשת להגדרות כל המשתמשים בלחיצה על רשומה. הדף מבצע שליפה של כל המשתמשים ממסד הנתונים ושם אותם ברשימה.
בדומה לדף הבית, גם כאן ישנה רשימה נוספת אשר מציגה את המשתמשים בהתאמה לפי סינון.
הדף מכיל סרגל חיפוש לסינון משתמשים לפי שם המשתמש. בנוסף לכך, באפשרות מנהל לקדם משתמש לדרגת מנהל וכן למחוק משתמשים ע"י לחיצה ארוכה על רשומה.
בנוסף לכך, ישנן כמה מחלקות עזר שנעשה בהם שימוש: מחלקת עזר הכוללת פעולות עזר בנוגע למסד הנתונים, מחלקת עזר השולפת מידע השמור בזיכרון המכשיר בנוגע לפרטי המשתמש המחובר זאת באמצעות ממשק ISharedPreferences ומחלקת עזר המטפלת בהמרת Bitmap של פקד תמונה למחרוזת וההפך.

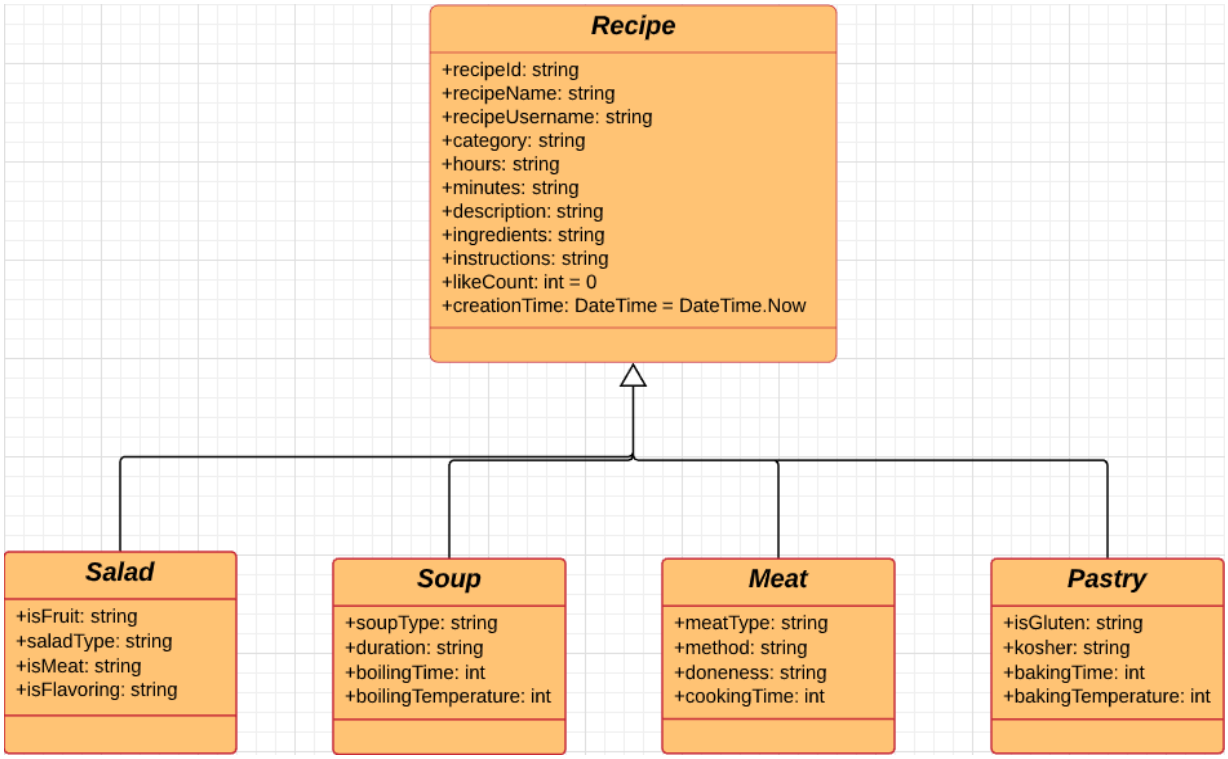
מחלקות הפרויקט

בפרויקט ישנו שימוש ב6 מחלקות מרכזיות אשר מסמלות את המידע הנשמר במסד הנתונים.
בפרויקט מחלקת User אשר מסמלת משתמש הרשום לאפליקציה. משתמש יכיל שם משתמש, סיסמא, שם פרטי, אימייל, תשובה על שאלת אבטחה, תמונת פרופיל, מחרוזת שכוללת את קודי המתכונים האהובים, מחרוזת שכוללת את קודי המתכונים שהמשתמש עצמו העלה ומשתנה בוליאני הקובע אם המשתמש הנ"ל הוא גם מנהל באפליקציה. משתמש נוצר לאחר הרשמה לאפליקציה. המשתמש הראשון שנרשם לאפליקציה יהיה מנהל, כלומר המשתנה הבוליאני יהיה true. רק מנהל רשאי להעלות משתמש לדרגת מנהל וכל מנהל יכול להוריד מנהל אחר לדרגת משתמש. מנהל רשאי למחוק משתמש, למחוק מתכון של משתמש, לעדכן מתכון של משתמש ולשנות למשתמש סיסמא.
לכל תכונה במחלקה יש get וset כך שניתן במחלקות אחרות באפליקציה לגשת לתכונות השונות של המשתמש. בעת ההרשמה נקבעים שם המשתמש, הסיסמא, השם, האימייל, התשובה לשאלת האבטחה והאם המשתמש הוא מנהל כל זאת לפי הנתונים שנקלטים. המחרוזת של תמונת הפרופיל מתעדכנת כאשר מוכנסת תמונה חדשה באמצעות המצלמה והמערכת ממירה את Bitmap התמונה למחרוזת באמצעות פעולת עזר במחלקת ImageManager.
מחרוזות קודי המתכונים מתעדכנים בהתאמה כאשר מתכון חדש מועלה ע"י המשתמש או כאשר המשתמש אהב מתכון ונתן לו "לייק". הערך ההתחלתי של המחרוזות הנ"ל הוא מחרוזת ריקה.
כל הנתונים הנ"ל נשמרים בטבלת Users שבמסד הנתונים.
משתמש יכול לשנות את סיסמתו באפליקציה כאשר הוא מחובר.
כאשר משתמש אינו מחובר ושכח את סיסמתו הוא יכול לשחזר אותה ועליו למלא שלושה שדות הכוללים: שם משתמש, אימייל והתשובה על שאלת האבטחה.



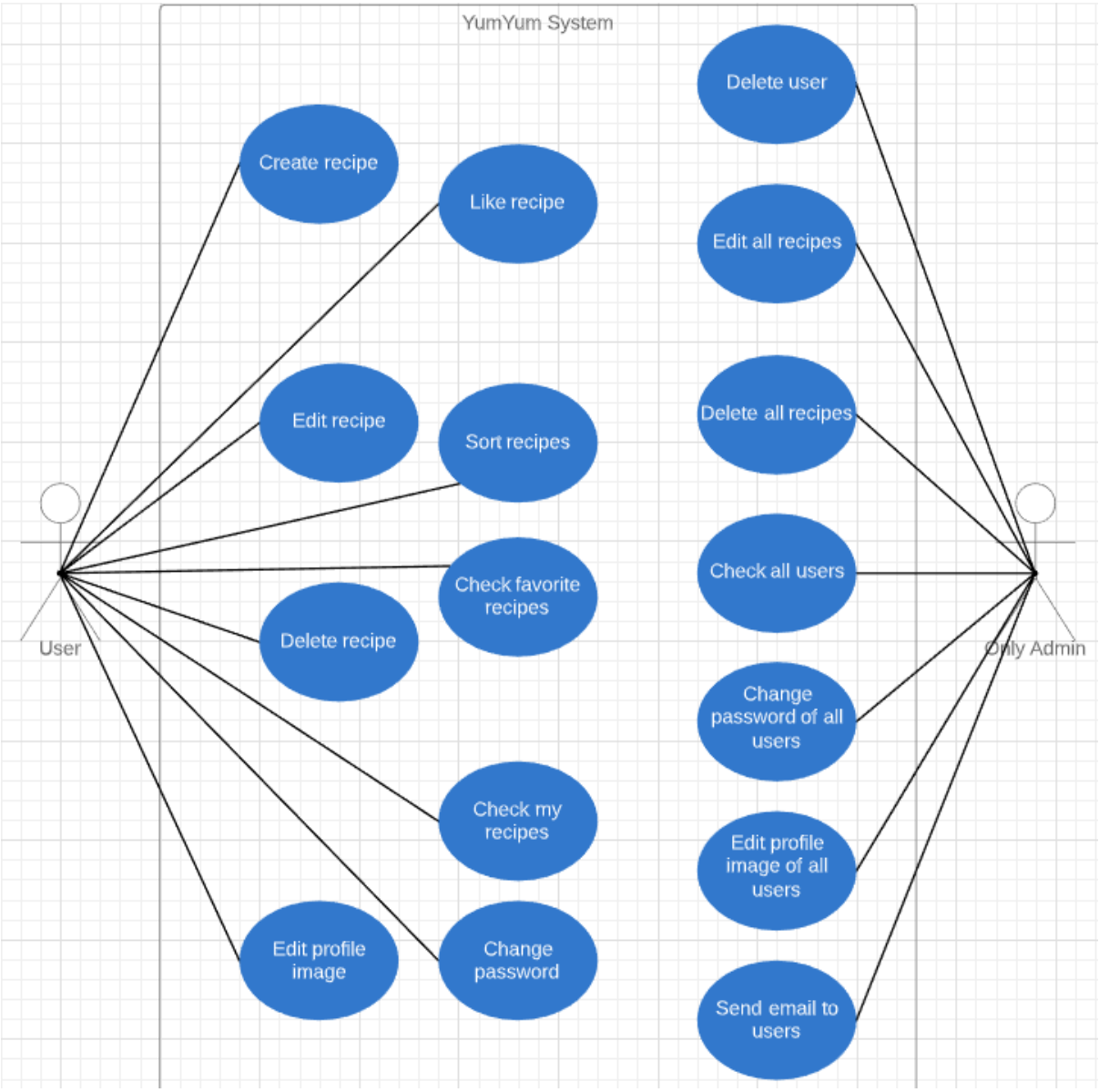
מחלקה נוספת בפרויקט היא מחלקת המתכון. מחלקת Recipe אשר מסמלת מתכון שהועלה לאפליקציה כוללת: קוד מתכון, שם מתכון, שם המשתמש שהעלה את המתכון, קטגוריות, שעות ודקות הכנה, תיאור מתכון, מרכיבים, הוראות הכנה, כמות "לייקים" וזמן העלאה לאפליקציה.
מתכון נוצר לאחר שהמשתמש סיים להזין פרטים תקינים לדף הכנת מתכון ולאחר אישור המתכון נשמר במסד הנתונים ומופיע בדף הבית. קוד המתכון מורכב מסוג המתכון, שם מהמתכון ושם המשתמש שיצר את המתכון; אשר מופרדים בנקודתיים בכדי שלכל מתכון יהיה קוד משלו.

הלייקים של המתכון נקבעים לפי כמות הפעמים שמשתמשים אהבו את המתכון, כאשר בדף שמציג את המתכון הם לחצו על אייקון הלב. בעת יצירת המתכון המספר הנ"ל שווה לאפס.
כמו כן, אחד מתכונות המחלקה היא זמן העלאת המתכון מטיפוס DataTime אשר מקבלת את הזמן הנוכחי בטלפון בעת יצירת האובייקט.
למחלקה Recipe יש 4 תתי מחלקות אשר למעשה קובעים את סוג המתכון זאת לפי בחירת המשתמש.
המחלקות הן: Salad, Soup, Meat, Pastry. כאשר משתמש יוצר מתכון הוא בוחר את סוג המתכון (סלטים, מרקים, מנות בשר או מאפים) ולפיכך המערכת יוצרת אובייקט מהטיפוס הנ"ל שיורש ממחלקת מתכון; כלומר מתכון לא נשמר במערכת כRecipe אלא מוכנס לטבלה הרלוונטית כ-Salad, Soup, Meat או Pastry. לכל אחת מתתי המחלקות ישנם ארבעה תכונות אשר מייצגות נתוני מידע שונים הייחודיים לטיפוס הנ"ל.
במחלקת סלט: מחרוזת האם הסלט הוא סלט פירות או ירקות, מחרוזת סוג הסלט, מחרוזת האם הסלט מכיל בשר (כן או לא) ומחרוזת האם הסלט מכיל טיבול.
במחלקת מרק: מחרוזת סוג המרק, מחרוזת אורך ההכנה (ארוך או קצר), מספר שלם דקות זמן הבישול ומספר שלם טמפרטורת הבישול.
במחלקת בשר: מחרוזת סוג בשר, מחרוזת שיטת ההכנה, מחרוזת רמת הצלייה ומספר שלם דקות זמן הבישול.
במחלקת מאפה: מחרוזת האם הוא מכיל גלוטן, מחרוזת כשרות (חלבי, בשרי, פרווה), מספר שלם זמן אפייה בדקות ומספר שלם טמפרטורת אפייה.
כל הנתונים ניתנים לשליפה בפעולות get וset.
לאחר יצירת המתכון המערכת מכניסה את האובייקט לטבלה התאימה לפי סוג המתכון המתאים.
באפשרות משתמש לעדכן את נתוני המתכון, אך אין באפשרותו לשנות את שם המתכון וסוגו; כך שלא יהיה שינוי בקוד המתכון.



Use Case Diagram

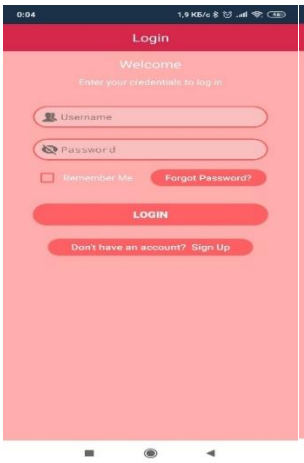
תרשים use case diagram אשר מראה את הפעולות השונות שמנהל ומשתמש באפליקציה יכולים לעשות.



מדריך למשתמש – הנחיות כלליות

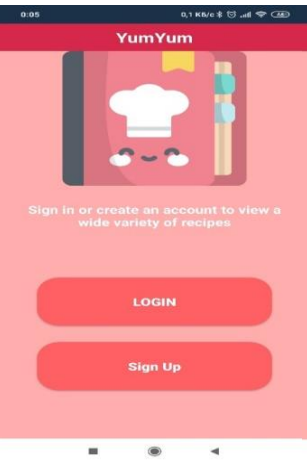
מסכי האפליקציה וכיצד מתבצעת האינטראקציה

Login



Signup

Join



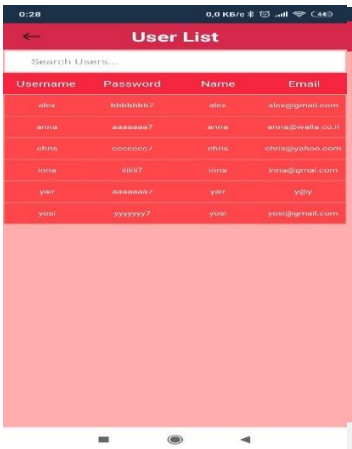
Start Page

My Recipes

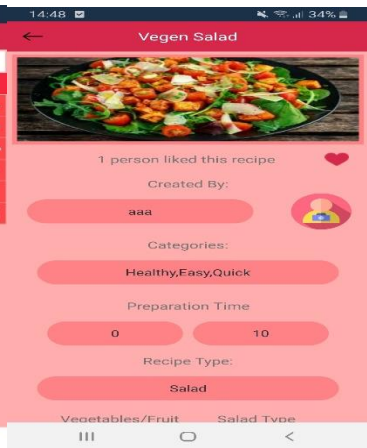
Favorite Recipes

Homepage

User List



Recipe Page

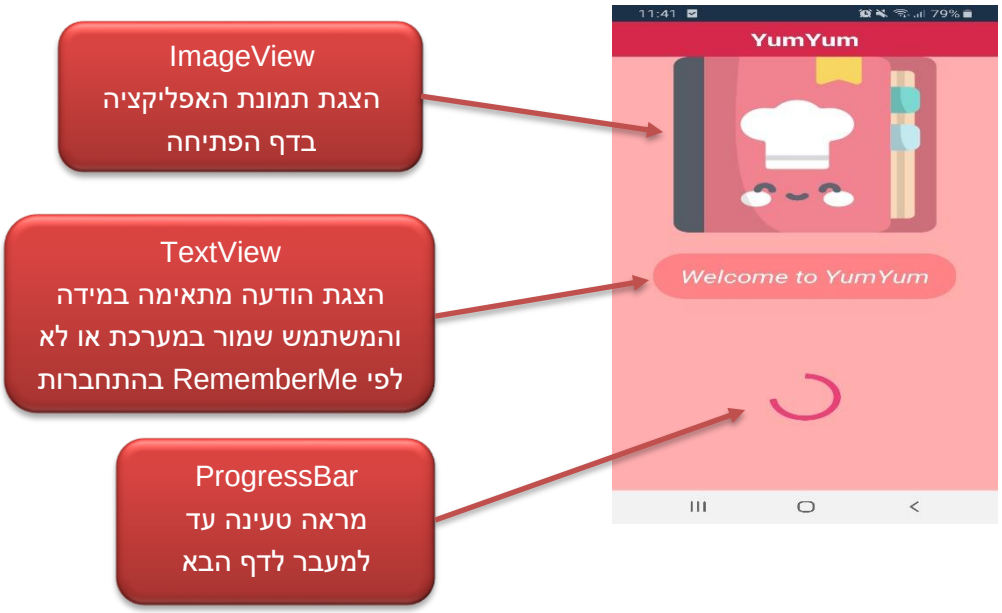


Create Recipe

Start Page

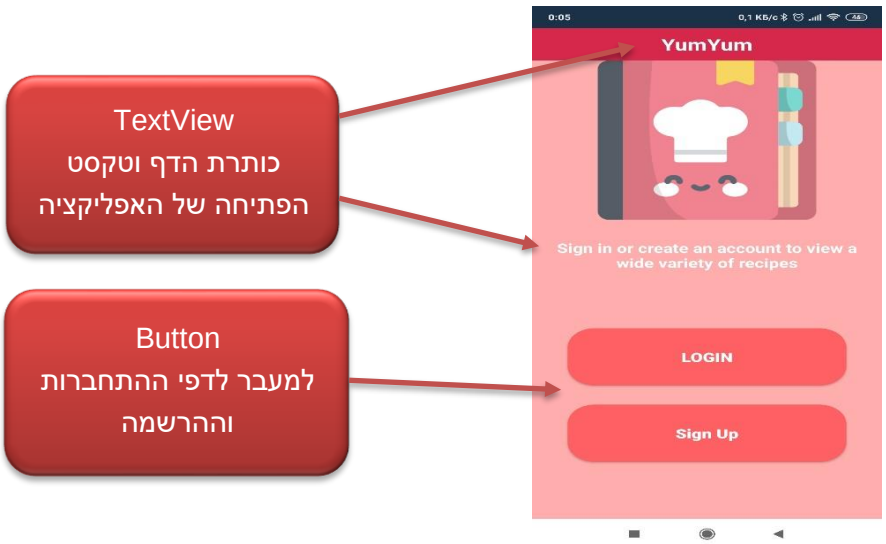
מסך זה מופעל עם הפעלת האפליקציה. כאשר מסך זה מוצג מתבצעת בדיקה שקובעת האם המשתמש מחובר למערכת (באמצעות אפשרות RememberMe) או לא. זוהי הכניסה הראשונה של המשתמש לאפליקציה. במידה והוא לא מחובר למערכת, הוא יועבר למסך הכניסה של האפליקציה ובמידה והוא כן מחובר, יועבר למסך דף הבית של האפליקציה המכילה את כל המתכונים.

פעולה נוספת שמתבצעת היא עדכון סטטוס המשתמש – אם המשתמש לא בחר באפשרות RememberMe במסך ההתחברות, פרטי המשתמש הנוכחי המנוהל על ידי ISharedPreferences יתאפסו ולמעשה לא יהיה משתמש נוכחי ויצטרך להירשם או להתחבר מחדש; אם המשתמש אכן בחר באפשרות זו, לא יעשה שינוי בפרטי המשתמש הנוכחי והוא יתחבר כרגיל לדף הבית.



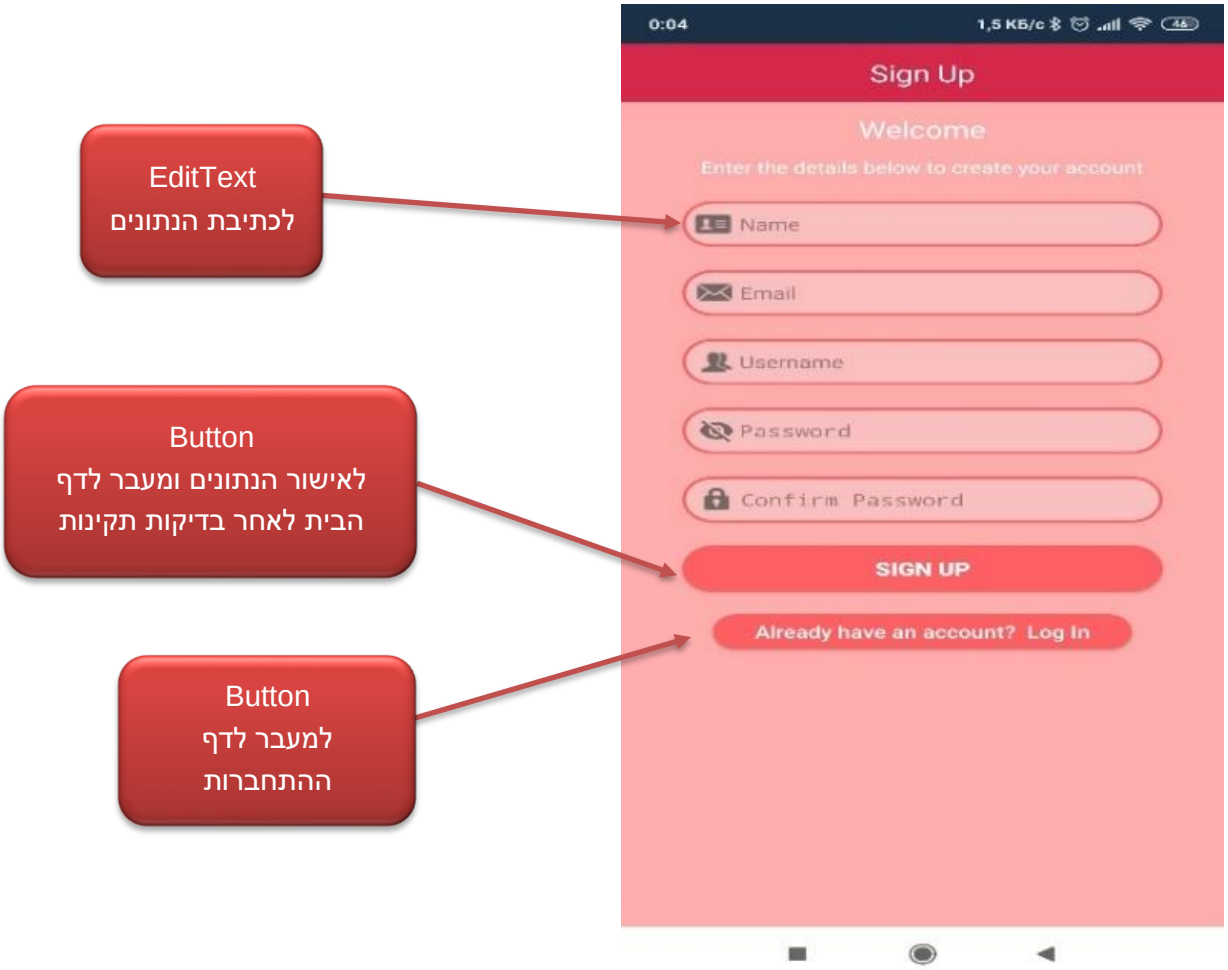
Join

מסך זה מופיע לאחר מסך הפתיחה במידה והמשתמש לא מחובר למערכת. מכיל קישורים לדפי ההרשמה והכניסה.



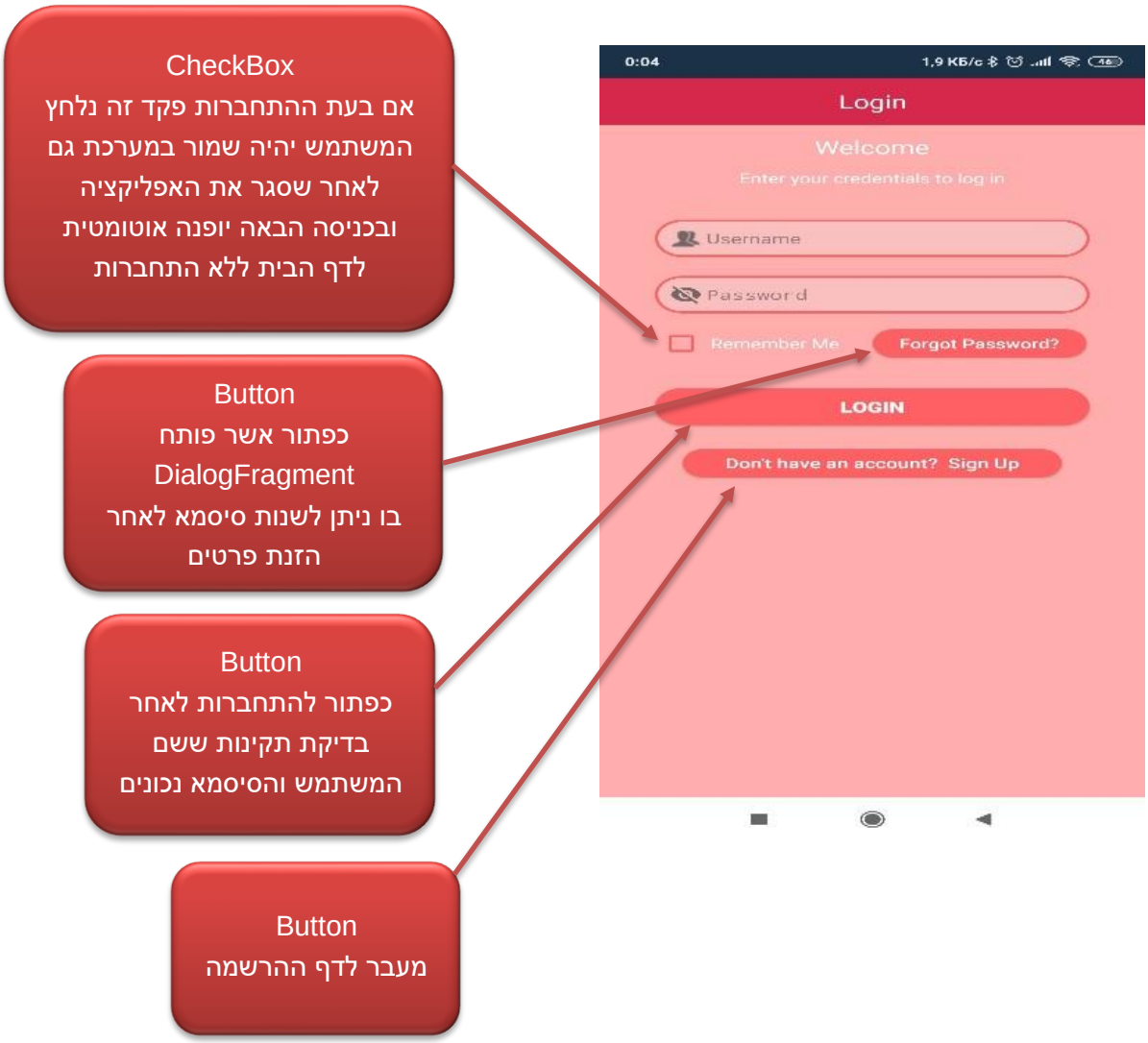
Signup

מסך ההרשמה של האפליקציה. דף הרשמה מכיל את הדרישות להזין שם, אימייל, שם משתמש, סיסמא ותשובה לשאלת אבטחה. לאחר ההרשמה הדף מפנה לדף הבית. ישנו קישור לדף ההתחברות.
דף ההרשמה מטפל בתקינות הקלט ומודיע שגיאות באמצעות AlertDialog כאשר הקלט אינו תקין.
בדיקות התקינות הן: שם יכיל עד ל-20 תווים והוא יכיל אותיות באנגלית בלבד, אימייל צריך להיות תקין כלומר ספרות לפני @ ואחרי על פי בדיקה בעזרת מחלקת MailAddress וכן הוא חייב להיות אימייל שלא שייך למשתמש אחר, שם משתמש יכיל בין 3-15 תווים, יכיל אותיות באנגלית ומספרים בלבד, לפחות אות אחת ואין משתמש אחר בעל אותו שם משתמש, סיסמא תכיל בין 8-16 תווים, תכיל אותיות באנגלית ומספרים, לפחות אות אחת ומספר אחד וכן יש לאשר סיסמא זו בהזנה נוספת של אותה הסיסמא, בסוף יש לענות על שאלת האבטחה והתשובה תהיה בין 3-30 תווים ותכיל אותיות באנגלית בלבד.
בעת סיום ההרשמה נוצר משתמש חדש המוכנס למסד הנתונים וישנו שימוש ב- ISharedPreferences לשמירת נתוני המשתמש במערכת כמשתמש נוכחי (במצב זה אפשרות RememberMe תקפה).



Login

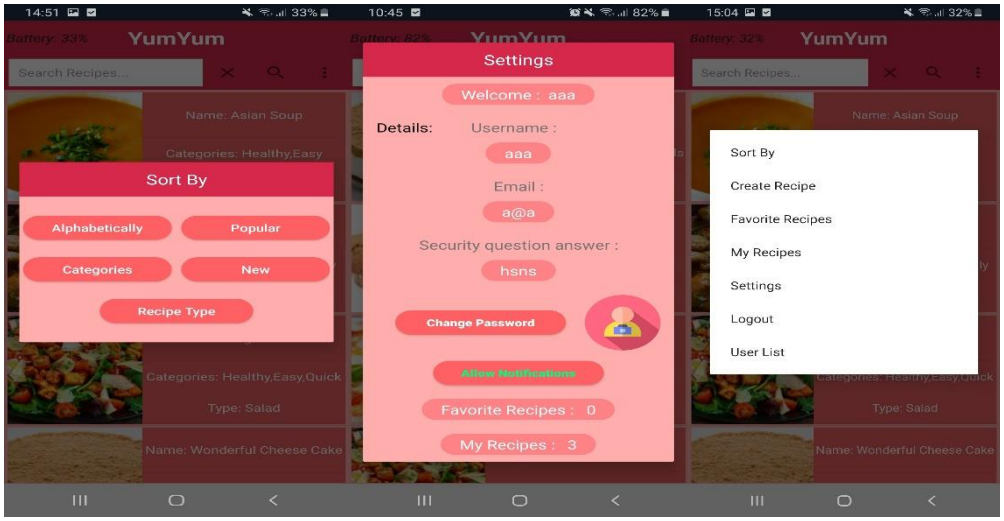
מסך ההתחברות של האפליקציה. דף התחברות המכיל את הדרישות להזין שם משתמש וסיסמא קיימים. לאחר ההתחברות הדף מפנה לדף הבית. ישנו קישור לדף ההרשמה. דף ההתחברות מטפל בתקינות הקלט ומודיע על שגיאה באמצעות AlertDialog, כאשר שם המשתמש והסיסמא אינם תואמים. דף ההתחברות מכיל כפתור לשכחתי סיסמא ולאפשרות של RememberMe.
אפשרות זו משמע שהמשתמש יוכל להישאר מחובר לאפליקציה גם במידה וסגר את האפליקציה לחלוטין. במטרה לממש זאת ישנו שימוש ב- ISharedPreferences – שמירת נתוני המשתמש וסטטוס המשתמש (האם המכשיר זוכר את המשתמש) בזיכרון המכשיר. בעת סיום ההתחברות פרטי המשתמש הנוכחי מתעדכנים בקובץ הנ"ל והוא מופנה לדף הבית.



Homepage

דף הבית של האפליקציה. דף הבית מכיל את כל המתכונים שהועלו על ידי המשתמשים הרשומים. ניתן לסנן ולחפש מתכון באמצעות כתיבת טקסט ובחירת קטגוריות למיון והסינון הנבחר.
בדף הבית ניתן לראות את כל המתכונים הזמינים המכילים את תמונת המתכון, שם המתכון, קטגוריות המתכון וסוג המתכון. לחיצה על מתכון מסוים תפנה את המשתמש לדף המתכון עם הפרטים הרלוונטיים. דף הבית מכיל כפתור (:) הפותח תפריט המכיל קישורים למסכים השונים באפליקציה הכוללים את דף יצירת המתכון, דף המתכונים שהמשתמש יצר ודף המתכונים האהובים של המשתמש.
בתפריט זה ישנה אפשרות להתנתקות אשר מפנה את המשתמש לדף Join ובעצם מאפסת את פרטי המשתמש הנוכחי באמצעות ISharedPreferences. כמו כן, ישנה אפשרות לפתוח את ההגדרות אשר מראות את פרטי המשתמש וכן נותנות אפשרות לשינוי סיסמא ושינוי תמונת פרופיל.
דרך התפריט ניתן לגשת להגדרות סינון על פי קריטריונים: הצגה אלפביתית על פי שם המתכון, פופולריות המתכון לפי כמות הלייקים, סינון לפי קטגוריות, הצגת מתכונים חדשים וסינון לפי סוג המתכון; המוצגים ע"י DialogFragment.
בנוסף לכך, למנהל בלבד האפשרות לגשת לדף רשימת המשתמשים דרך התפריט.

תפריט דף הבית *הגדרות* *בחירת קריטריון הסינון*



בדף ההגדרות ניתן לשנות תמונת פרופיל בלחיצה על תמונת הפרופיל (CircleImageView) ותיפתח אפשרות באמצעות AlertDialog לצלם תמונת פרופיל דרך המצלמה וגם אפשרות להסיר את תמונת הפרופיל הקיימת.

כמו כן, בדף ישנה אפשרות לשינוי סיסמא בלחיצת הכפתור Change Password לאחר מכן יש למלא את הסיסמא לפי בדיקות התקינות (כמו בדף ההרשמה) ולאחר מכן לאשר את הסיסמא.

בדף ישנה גם אפשרות לאשר או לבטל את שירות ההודעות של האפליקציה אשר נדון עליה בהמשך.

ב- DialogFragment של אפשרות הסינון המשתמש רשאי לבחור את אפשרות הסינון המתאימה לו.

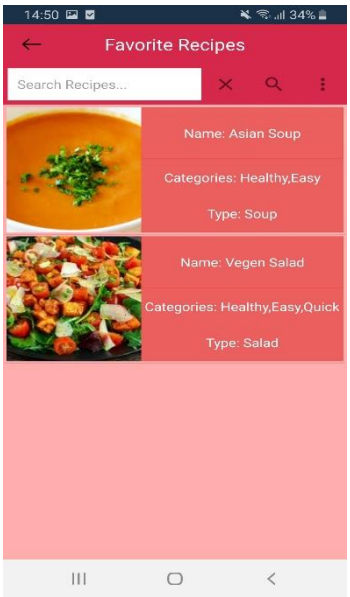
במידה והוא בחר סינון לפי קטגוריות ייפתח DialogFragment חדש עם אפשרויות הקטגוריות והמשתמש רשאי לבחור עד 3 קטגוריות.

במידה והוא בחר לפי סוג המתכון ייפתח DialogFragment עם ארבעה אפשרויות לבחירה (Salad,Soup,Meat,Pastry) – על המשתמש לבחור אחד ולאחר מכן ייפתח DialogFragment נוסף שם המשתמש יצטרך למלא את הפרטים אודות סוג המתכון שבחר.

לאחר אישור ייסגר החלון ובדף הראשי רשימת המתכונים תתעדכן כנדרש.

Favorite Recipes

בדף המתכונים האהובים יופיעו המתכונים אותם סימן המשתמש בלב (פרטים ב- Recipe Page).
ישנו כפתור לחזרה לדף הראשי. מבנה הדף הוא כמו הדף הראשי כך שישנן אפשרויות חיפוש, סינון, איפוס הטקסט ואפשרות לגשת לפרטי המתכון בלחיצה על המתכון.
כפתור (:) פותח את אפשרויות הסינון כמו בדף הבית.



My Recipes

בדף המתכונים שלי יופיעו המתכונים אותם העלה המשתמש.
כל מתכון שהעלה המשתמש הנ"ל יופיע בדף זה.
ישנו כפתור לחזרה לדף הראשי. מבנה הדף הוא כמו הדף הראשי כך שישנן אפשרויות חיפוש, סינון, איפוס הטקסט ואפשרות לגשת לפרטי המתכון בלחיצה על המתכון.
כפתור (:) פותח את אפשרויות הסינון כמו בדף הבית.

Create Recipe

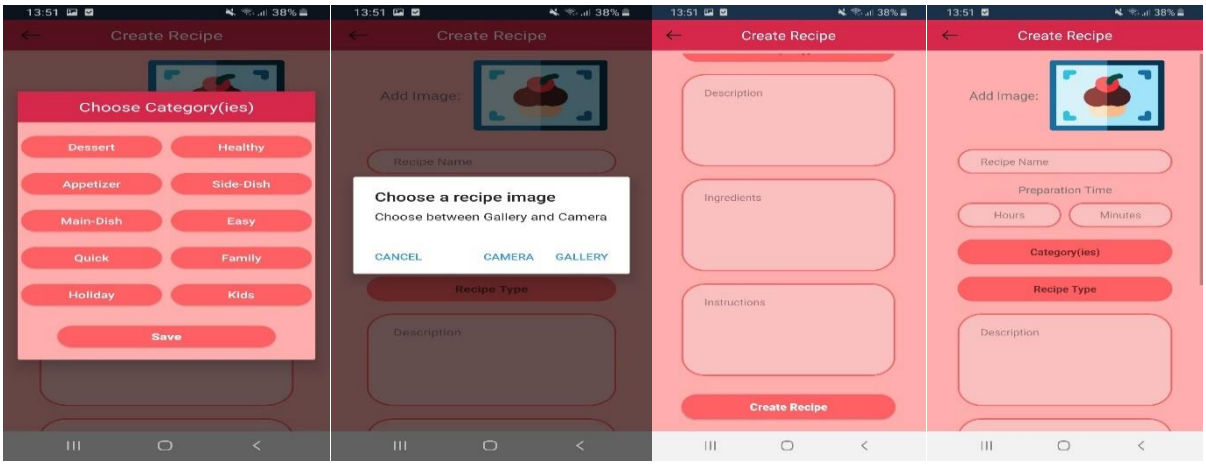
דף הכנת המתכון באפליקציה. בדף זה משתמשים יכולים ליצור את המתכון המועדף עליהם ובסיום, המתכון מועלה לדף הבית ולדף המתכונים שהמשתמש העלה. בכדי ליצור מתכון על המשתמש להעלות את תמונת המתכון וזאת ניתן לעשות באמצעות לחיצה על תמונת הוספת מתכון (ImageView) ובחירת תמונה מהגלריה או צילום דרך המצלמה. בנוסף לכך, ישנן דרישות נוספות – שם המתכון, זמן הכנת המתכון, קטגוריות, סוג המתכון, תיאור המתכון, רכיבי המתכון, ואופן הכנתו (EditText). במידה והקלט אינו תקין ישנה הודעה על שגיאה באמצעות AlertDialog.

בדיקות התקינות הן: חובה להעלות תמונה של המתכון, יש להזין שם מתכון אשר יהיה מורכב מאותיות באנגלית בלבד, יש למלא את זמן ההכנה המורכב משעות ודקות, כאשר בוחרים קטגוריה יש לבחור בין 1-3 קטגוריות, חובה לבחור את סוג המתכון, שדות תיאור המתכון, המרכיבים והוראות ההכנה לא יכולים להיות ריקים.

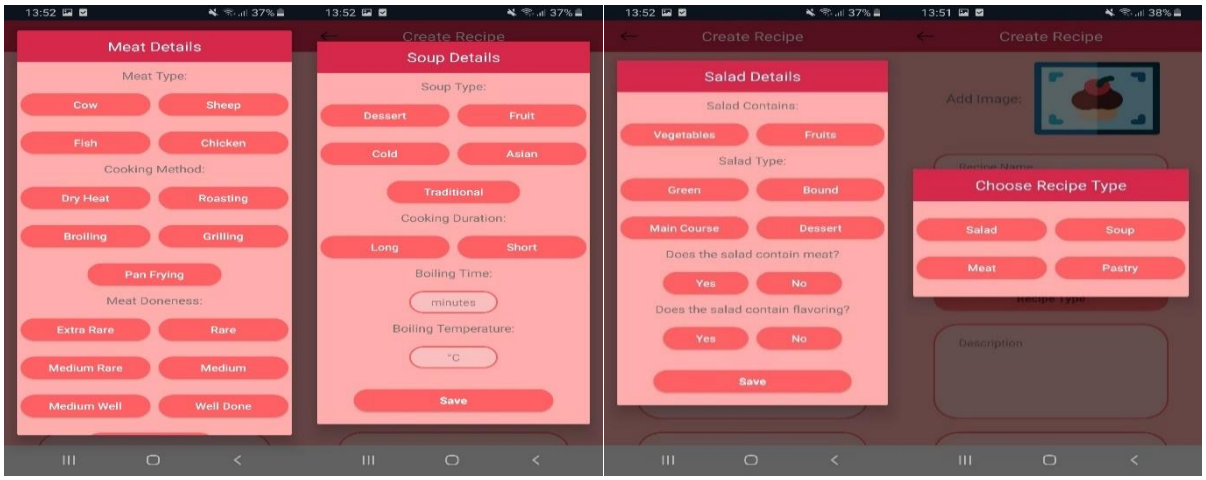
לאחר בחירת סוג המתכון (סלט/מרק/בשר/מאפה) יפתח דיאלוג נוסף בו יש למלא את הפרטים הרלוונטים הקשורים לסוג המתכון אותו בחר המשתמש. לאחר מילוי הפרטים ייסגר הדיאלוג, סוג המתכון בדף ישתנה למה שהמשתמש בחר והפרטים שמילא יופיעו בדף.

בסיום יצירת המתכון יש ללחוץ על הכפתור שבסוף הדף, הפרטים ישמרו במסד הנתונים וישנה הפנייה לדף הבית - עתה המתכון יופיע בדף הבית. כמו כן, ישנו כפתור לחזרה לדף הבית.

דף הכנת המתכון בחירת תמונה לפי גלריה ומצלמה בחירת קטגוריות



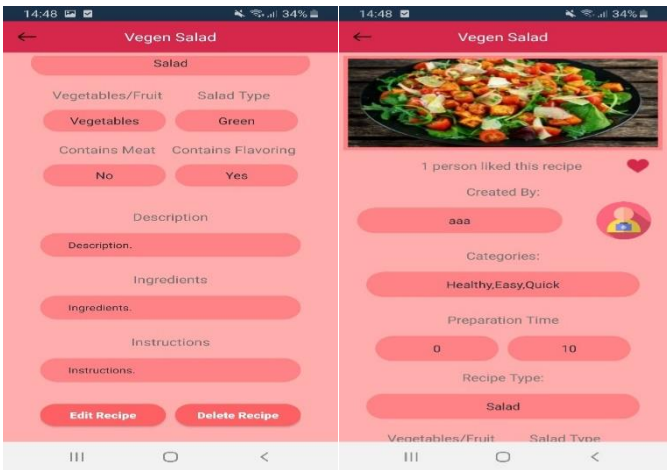
בחירת סוג מתכון דפי פרטי סוג המתכון לפי בחירת המשתמש



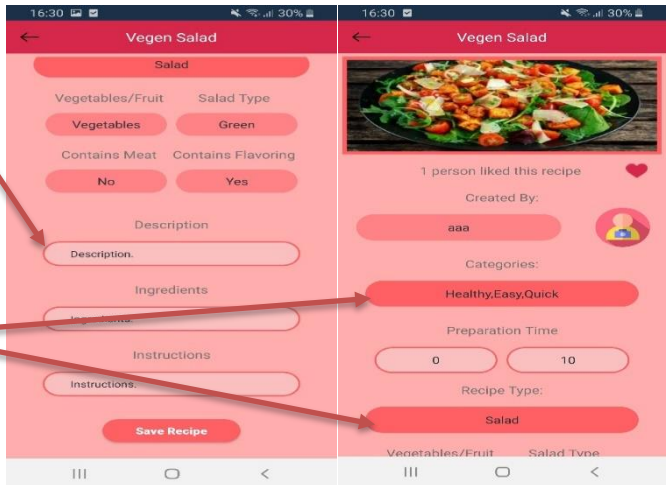
Recipe Page

לאחר לחיצה על מתכון בדפים השונים מופנה המשתמש לדף המתכון אשר מכיל את פרטי המתכון.
דף המתכון מכיל את הפרטים הבאים: שם המתכון, תמונת המתכון, שם משתמש יוצר המתכון ותמונת הפרופיל שלו, מספר הלייקים של המתכון, הקטגוריות, סוג מתכון (סלט/מרק/בשר/מאפה) ופרטיו, שעות ודקות הכנה, תיאור המתכון, מרכיבים והוראות הכנה.
בדף המתכון ישנו לב ובלחיצה עליו מספר הלייקים של המתכון יעלה באחד, הלב יהפוך לאדום ועתה המתכון יתווסף לדף המתכונים האהובים של המשתמש הנ"ל. אם ילחץ בשנית ירד מספר הלייקים באחד, הלב יהפוך ללבן והמתכון ירד מדף המתכונים האהובים של המשתמש.
בנוסף משתמש שיצר מתכון יוכל לערוך ולמחוק את המתכון שיצר. כמו כן, אפשרות זו קיימת למנהלים אשר באפשרותם לערוך ולמחוק מתכונים של משתמשים אחרים. (כאשר מתכון נמחק ע"י מנהל באפשרות המנהל לשלוח למשתמש הודעה על כך דרך האימייל).
מחיקת מתכון תמחק את המתכון במסד הנתונים של האפליקציה.
עריכת מתכון משנה את צבעי הפקדים הניתנים לעריכה הכוללים: תמונת המתכון, קטגוריות, פרטי סוג המתכון, זמני ההכנה, תיאור המתכון, מרכיבים ואופן ההכנה. לאחר שינוי הפרטים ואישורם לפי בדיקות התקינות (כפי שמוצגים בדף הכנת מתכון) ניתן לאשר ועתה פרטי המתכון ישתנו במסד הנתונים.
ישנו כפתור לחזרה לדף הבית.

דף המתכון



דף המתכון בעת עריכה

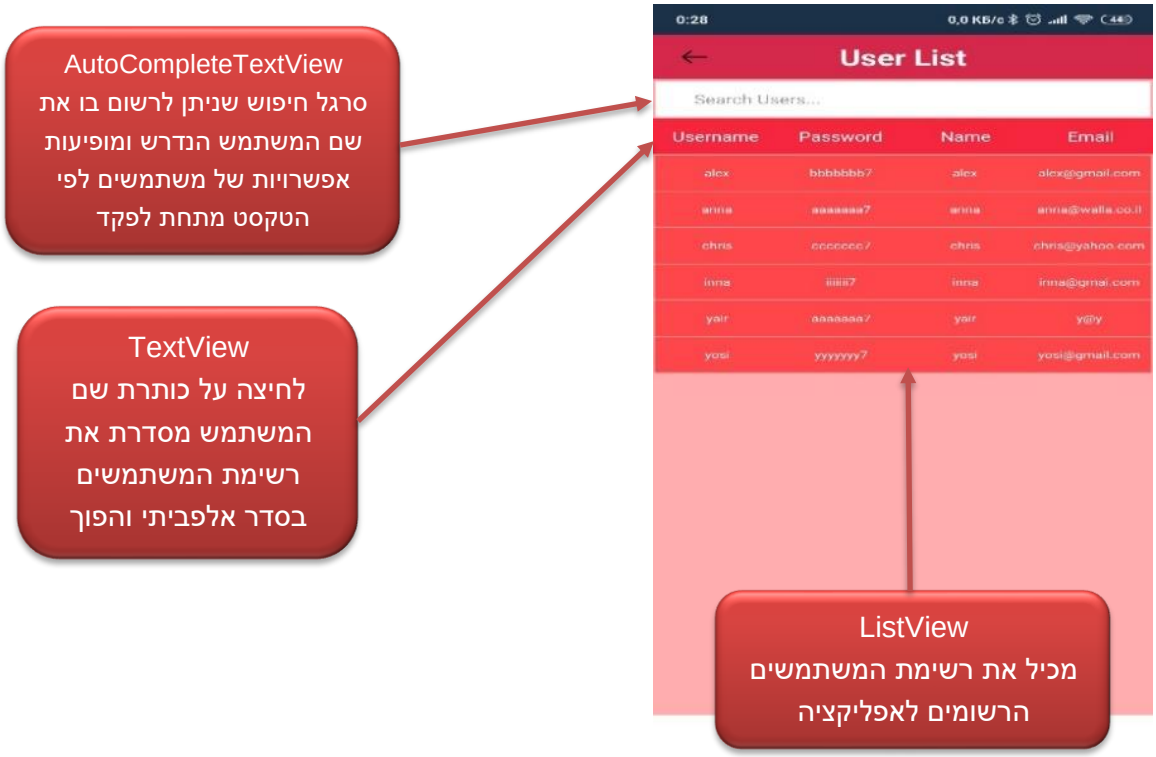


EditText
ישנו שימוש בפקדים מסוג זה כדי לבצע עריכת נתונים בעת במצב עריכה. במצב רגיל תכונת עריכת הפקד מושבתת

TextView
שימוש בפקד זה בשינוי קטגוריות ופרטי סוג מתכון.
בעת עריכה ניתן ללחוץ על הפקד ולבצע שינוי ובמצב רגיל לא ניתן

User List

דף רשימת המשתמשים הרשומים לאפליקציה. בדף זה מוצגים פרטי כל המשתמשים הכוללים את שם המשתמש, סיסמא, שם, ואימייל. רק למנהלי האפליקציה גישה לדף זה.
לחיצה על משתמש פותחת אפשרות להגדרות כמו בדף הראשי ומקנה בעצם למנהל אפשרות לשנות סיסמא ותמונת פרופיל למשתמש הנ"ל.
לחיצה ארוכה על משתמש פותחת אפשרות למחיקת משתמש מסוים ואפשרות לעלות משתמש לדרגת מנהל או הורדה לדרגת משתמש רגיל; דברים אלו יוצגו באמצעות AlertDialog.
במידה ומשתמש נמחק נתוני המשתמש נמחקים ממסד הנתונים וכן כל המתכונים שהעלה.
(כאשר משתמש נמחק ע"י מנהל באפשרות המנהל לשלוח למשתמש הודעה על כך דרך האימייל).
בדף אפשרות לחיפוש משתמש באמצעות הקלדת טקסט וכן אפשרות לסידור אלפביתי והפוך על סמך שמות המשתמשים (בלחיצה על כותרת שמות המשתמשים - Username).
ישנו כפתור חזרה לדף הבית.



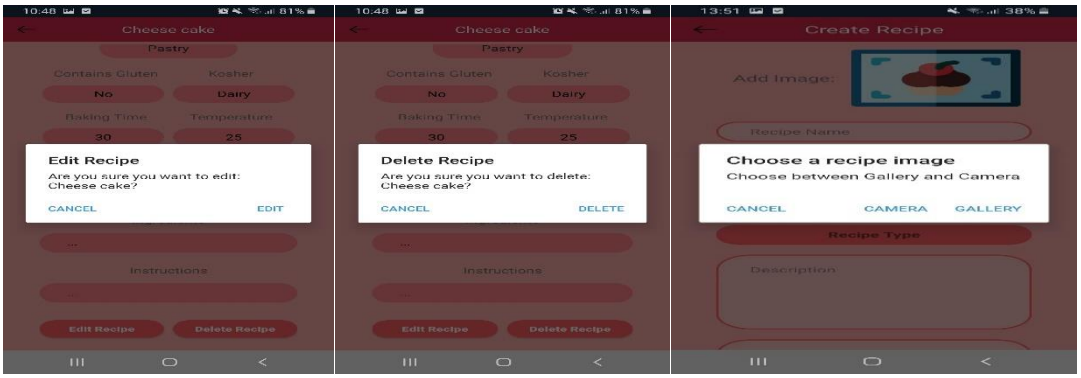
AlertDialog

באפליקציה ישנו שימוש בתיבות קופצות מסוג AlertDialog שמטרתן להודיע על שגיאות תקינות, על כך שלא כל פרטי השדות מולאו, אישור עדכון ומחיקת מתכון, בחירת דרך העלאת תמונת מתכון (גלריית תמונות או מצלמה) וכדומה.

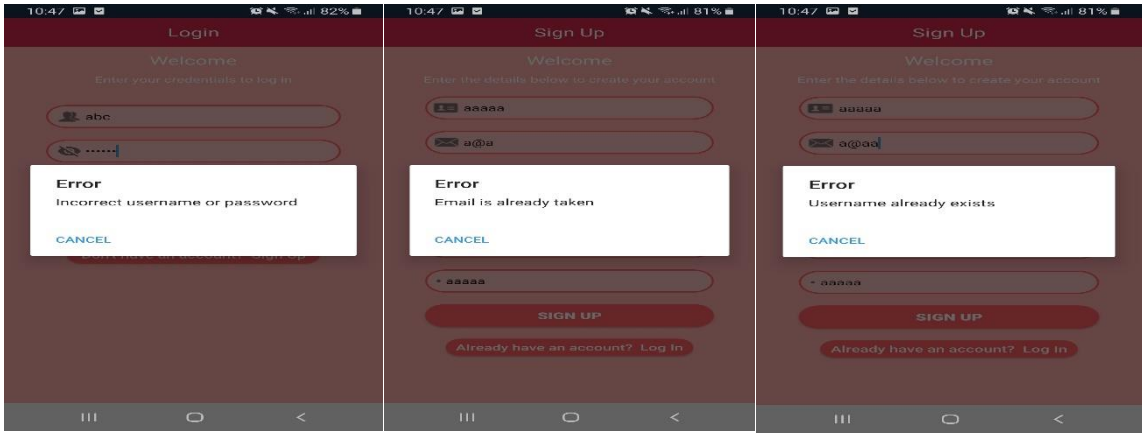
השימוש בהודעות אלו הוא תדיר באפליקציה כדי שתהיה אינטראקציה עם המשתמש לאורך האפליקציה.

לדוגמא:

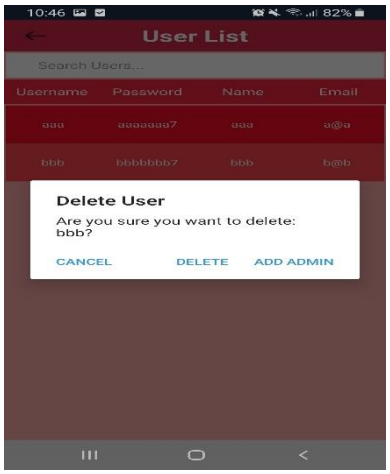
בחירת תמונה לפי גלריה או מצלמה *אישור מחיקת מתכון* *אישור עדכון מתכון*



שם המשתמש קיים באפליקציה *האימייל כבר קיים באפליקציה* *שם המשתמש או הסיסמא לא תקינים*



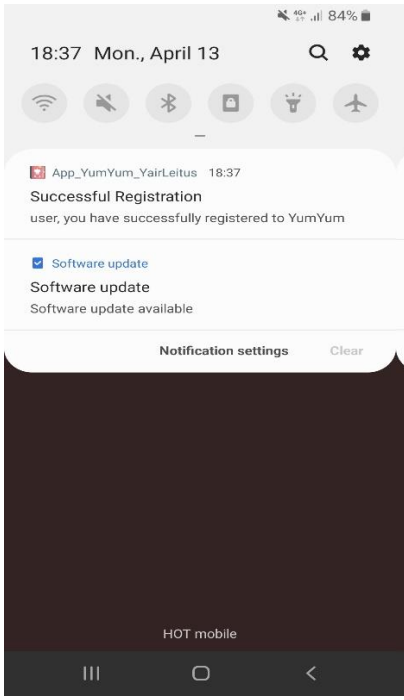
הודעה לאישור מחיקת משתמש בידי מנהל



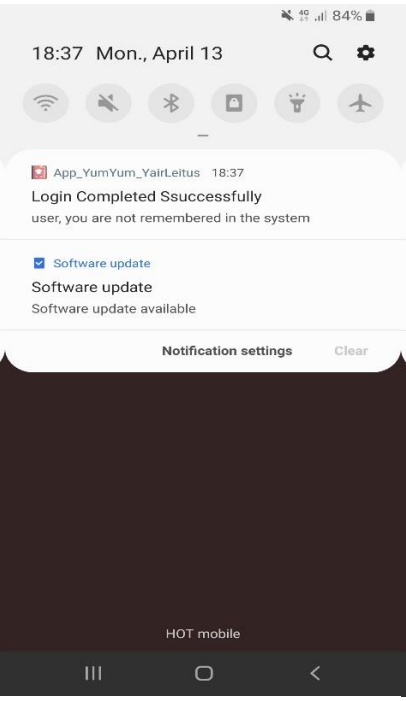
StatusNotificationService

במהלך העבודה נעשה שימוש ב-Service במטרה להציג הודעות שונות באמצעות Notification. ההודעות השונות מתקבלות בכמה מקרים: הודעה לאחר הרשמה, הודעה לאחר התחברות, הודעה לאחר העלאת מתכון בהצלחה, הודעה לאחר עדכון מתכון, הודעה לאחר מחיקת מתכון והודעה לאחר מתן לייק למתכון והוספתו למתכונים האהובים. ההודעות למעשה פועלות כסטטוס של המשתמש ומעירות לו על פעולותיו באפליקציה. ההודעות יישארו ברקע המסך כל עוד המשתמש מחובר למערכת האפליקציה, ובעת התנתקות או כניסה חוזרת לאפליקציה כאשר המשתמש לא נשמר במערכת באמצעות RememberMe בעת ההתחברות, ה-Service יופסק וכן ההודעות יסגרו.

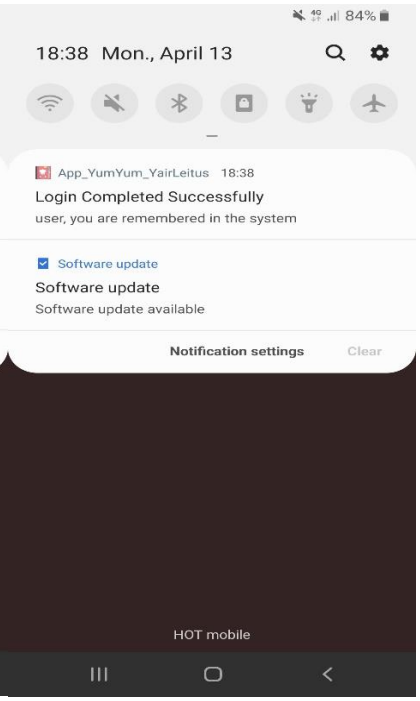
הודעת הרשמה לאפליקציה



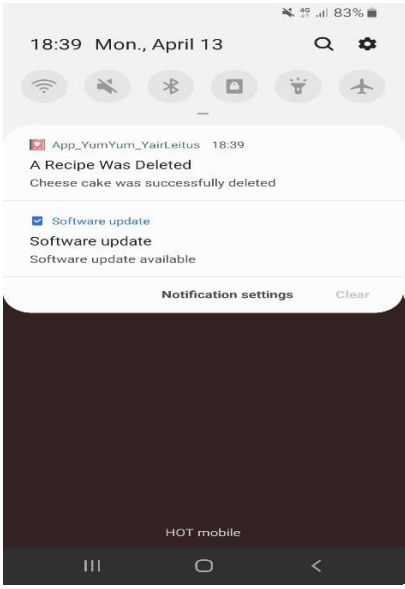
הודעת התחברות ללא RememberMe



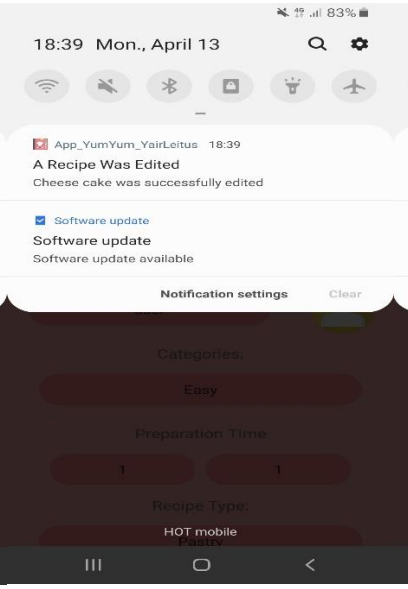
הודעת התחברות עם RememberMe



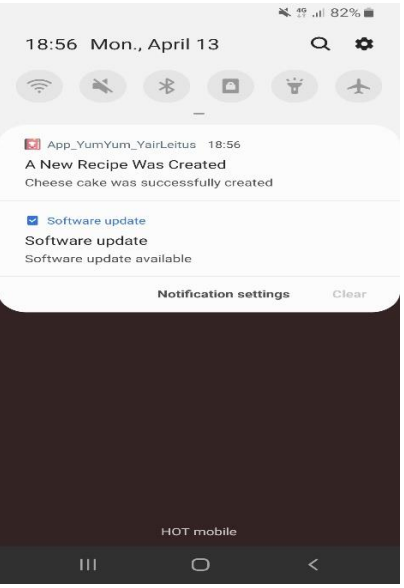
הודעת יצירת מתכון חדש



הודעת עדכון מתכון



הודעת מחיקת מתכון



בסיס הנתונים

בפרויקט השתמשתי בבסיס נתונים מקומי (local), SQLite. בבסיס נתונים זה נשמרים פרטי המשתמש בעת הרשמתו למערכת וכן פרטי מתכון שמועלים על ידי המשתמשים.

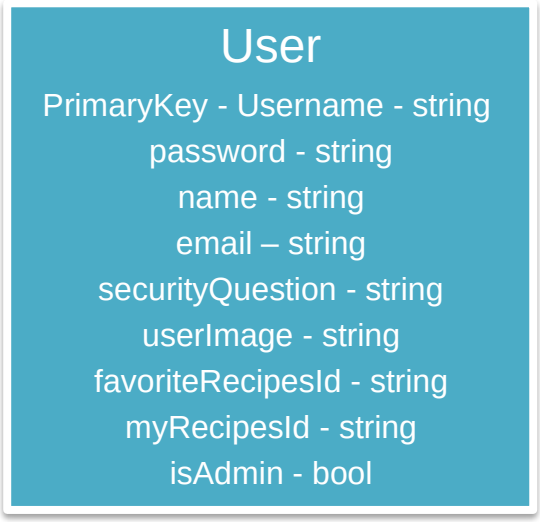
מבנה בסיס הנתונים

בסיס הנתונים כולל שש טבלאות:

טבלה אחת מאחסנת את פרטי המשתמש שנרשם לאפליקציה. בעת הרשמתו המשתמש מקליד את הנתונים ואלו לאחר בדיקות התקינות הרלוונטיות נשמרים במסד הנתונים.

טבלת User כוללת בתוכה תשעה פרטים: שם משתמש, סיסמא, שם, אימייל, שאלת אבטחה, תמונת משתמש, מחרוזת של קודי המתכונים האהובים, מחרוזת של קודי המתכונים שלי, והאם המשתמש מנהל.

שם המשתמש הוא המפתח הראשי אשר ייחודי לכל משתמש והוא רשאי להופיע פעם אחת, כלומר במערכת שם משתמש מסוים יופיע פעם אחת בלבד.



טבלה שנייה היא טבלת המתכון אשר מאחסנת את הפרטים שכל מתכון מכיל.
טבלת Recipe כוללת את הפרטים הבאים: קוד המתכון, שם המתכון, שם המשתמש שיצר את המתכון, תמונת מתכון שמאוחסנת בתור מחרוזת, קטגוריות המתכון, שעות ההכנה, דקות ההכנה, תיאור המתכון, מרכיבי המתכון, הוראות ההכנה, מספר לייקים, והזמן בו המתכון נוצר מטיפוס DateTime.
קוד המתכון RecipeId הוא המפתח הראשי שייחודי לכל מתכון ולכן לא יופיעו מתכונים עם קוד זהה.
הקוד מורכב משלושה מרכיבים המופרדים בנקודותיים (:) לשם נוחות מאחר ומרכיבים אלו אינם מכילים את התו הנ"ל. המרכיב הראשון הוא סוג המתכון (Salad, Soup, Meat, Pastry), השני הוא שם המתכון, והשלישי הוא שם המשתמש שהכין את המתכון - (type:recipeName:username).
המפתח הזר שבעצם מקשר את הטבלה הנ"ל לטבלת המשתמשים הוא recipeUsername כך שניתן לדעת מי המשתמש שיצר את המתכון.

Recipe
Primary Key – RecipeId - string
Foreign Key - recipeUsername - string
recipeName - string
recipeImage - string
category - string
hours - int
minutes - int
description - string
ingredients - string
instructions - string
likeCount - int
creationTime - DateTime

ארבעת הטבלאות הבאות (Salad, Soup, Meat, Pastry) מכילות את הערכים בטבלת Recipe מאחר והמחלקות שהן מייצגות יורשות מהמחלקה Recipe ולכן המפתח הראשי בטבלאות אלו הוא המפתח הראשי שבטבלת המתכון והוא RecipeId.

הטבלה השלישית היא טבלת Salad המכילה ארבעה פריטים המציינים: האם הסלט הוא סלט פירות או ירקות, סוג הסלט, האם הסלט מכיל בשר, והאם מכיל טיבול.

Salad
isFruit - string
saladType - string
isMeat - string
isFlavoring - string

הטבלה הרביעית היא טבלת Soup המכילה ארבעה פריטים המציינים: את סוג המרק, האם מדובר בבישול ארוך או קצר, זמן הבישול בדקות, וטמפרטורת הבישול בצלזיוס.

Soup
soupType - string
duration - string
boilingTime - int
boilingTemperature - int

הטבלה החמישית היא טבלת Meat המכילה ארבעה פריטים המציינים: את סוג הבשר, את דרך ההכנה של הבשר, את רמת הצלייה (doneness), זמן הבישול בדקות.

Meat
meatType - string
method - string
doneness - string
cookingTime - int

הטבלה השישית היא טבלת Pastry המכילה ארבעה פריטים המציינים: האם המאפה מכיל גלוטן, האם המאפה הוא בשרי/חלבי/פרווה, זמן הבישול בדקות, וטמפרטורת הבישול בצלזיוס.

Pastry
isGluten - string
kosher - string
bakingTime - int
bakingTemperature - int

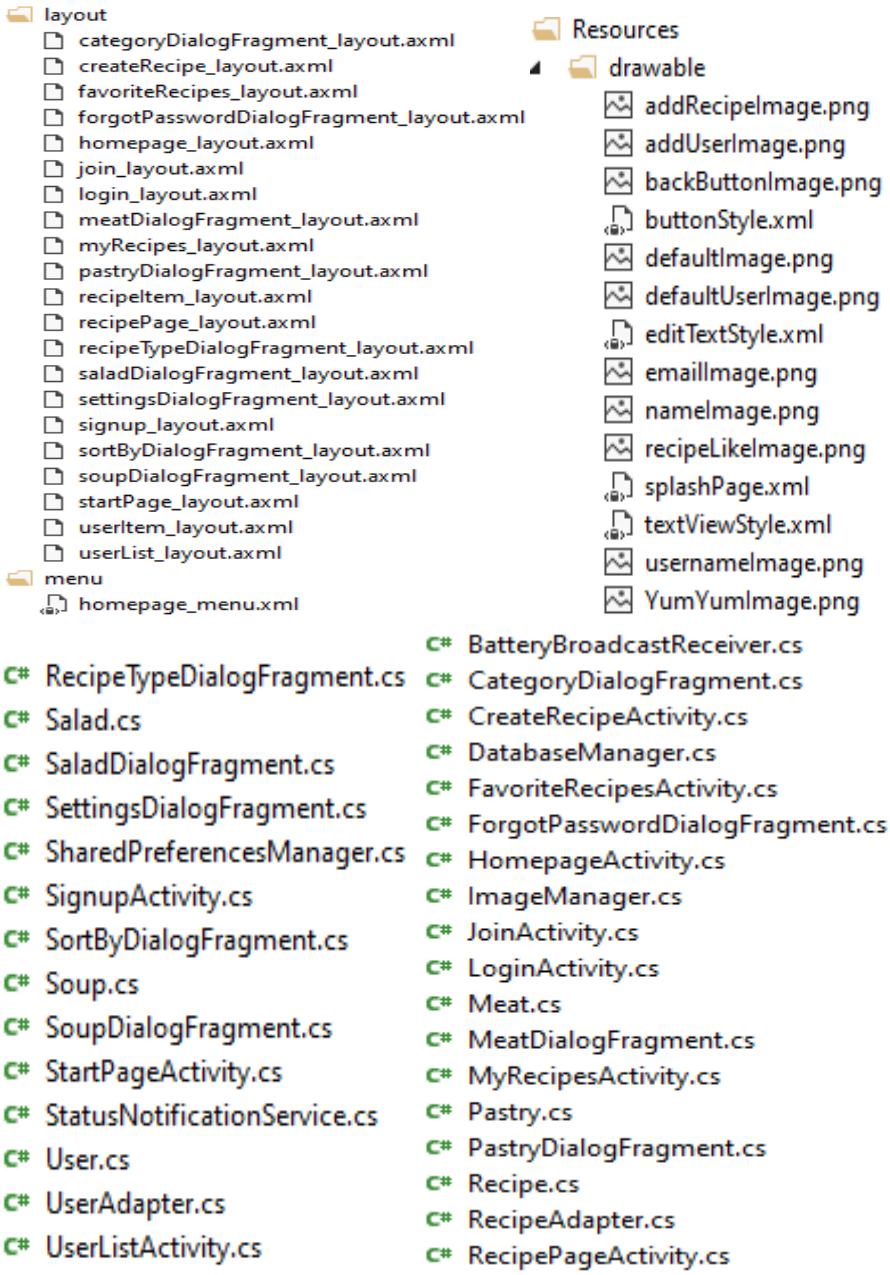
עבודה עם בסיס הנתונים

לשם עבודה עם בסיס הנתונים נעזרתי במחלקת עזר סטטית שיצרתי בשם DatabaseManager על מנת לבצע פעולות על בסיס הנתונים כמו שמירה, קריאה, מחיקה ועדכון וכן פעולות עזר נוספות הדורשות שימוש במסד הנתונים. במחלקה זו התייחסתי לכל הטבלאות הכוללות את Pastry, Meat, Soup, Salad, Recipe, User. תוכן המחלקה הנ"ל יוצג בהמשך כחלק מהמדריך למפתח.

מדריך למפתח

קבצי הפרויקט

קבצי הפרויקט הרלוונטיים מורכבים מכמה רכיבים: תיקיית Resources אשר מכילה בתוכה את תיקיית drawable אשר כוללת בתוכה קבצי xml לעיצוב פקדים שונים ותמונות שונות; היא מכילה גם את תיקיית layout אשר כוללת בתוכה את קבצי הדפים של האפליקציה; ובנוסף מכילה את תיקיית menu אשר כוללת קובץ xml של תפריט דף הבית. כמו כן, הפרויקט מכיל קבצי #c המקושרים ל-layout שונים כך שהאפליקציה יכולה לפעול כראוי.



Drawable

קבצי תמונות

addRecipeImage – תמונה להוספת תמונת מתכון.
addUserImage – תמונה להוספת תמונת פרופיל משתמש.
backButtonImage – תמונה של כפתור החזרה.
defaultImage – תמונת ברירת מחדל אשר לא מופיע בפועל באפליקציה.
defaultUserImage – תמונת ברירת מחדל של פרופיל משתמש.
emailImage – אייקון אימייל בשדות למילוי אימייל.
nameImage – אייקון שם בשדות למילוי שם פרטי.
recipeLikeImage – אייקון לב להוספת מתכון לרשימת המתכונים האהובים.
usernameImage – אייקון משתמש בשדות למילוי שם משתמש.
YumYumImage – תמונת האפליקציה.

buttonStyle

קובץ xml אשר משמש לעיצוב פקדי הכפתורים שבאפליקציה. עיצוב פקד הכפתור משנה את צבעו לכתום ומעניק לו קצוות מעוגלים.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<!-- הגדרת עיצוב כפתור באפליקציה - צורה, צבע ועיגול קצוות הפקד -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
  <!-- צבע רקע כתום -->
  <solid android:color="#ffff6064"/>
  <corners android:radius="30dp"/>
<shape/>
```

editTextStyle

קובץ xml אשר משמש לעיצוב פקדי EditText שבאפליקציה. עיצוב הפקד משנה את צבעו ללבן כאשר המסגרת בצבע כתום ומעניק לו קצוות מעוגלים.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<!-- הגדרת עיצוב פקד EditText באפליקציה - צורה, צבע, עיגול קצוות הפקד וצבע הקצוות -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
  <!-- צבע רקע לבן - ורדרד -->
  <solid android:color="#4ceeeeee"/>
  <corners android:radius="30dp"/>
  <!-- צבע קצוות כתום -->
  <stroke android:width="2dp" android:color="#ffff6064"/>
<shape/>
```

splashPage

קובץ xml אשר משמש כ-Splash Page אשר בעצם פשוט מראה רקע ורוד, כרקע האפליקציה. בקובץ styles שבתיקיית values נעשה שימוש בקובץ הנ"ל כך שהוא מהווה מסך רקע כללי, כלומר בעת טעינת מסך ארוכה באפליקציה במקום מסך לבן יופיע הרקע הנ"ל.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- הגדרת צבע דף Splash לצבע ורוד - מופיע בטעינת האפליקציה בפתיחתה  -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
  <!-- צבע רקע ורוד בהיר -->
  <solid android:color="#ffffadad"/>
<shape/>
```

textViewStyle

קובץ xml אשר משמש לעיצוב פקדי TextView שבאפליקציה. עיצוב הפקד משנה את צבעו לורוד ומעניק לו קצוות מעוגלים.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- הגדרת עיצוב פקד TextView באפליקציה - צורה, צבע ועיגול קצוות הפקד -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
  <!-- צבע רקע ורוד -->
  <solid android:color="#ffff8386"/>
  <corners android:radius="30dp"/>
<shape/>
```

Values

Styles

קובץ xml מובנה המכיל רכיבים שונים של עיצוב. בקובץ הוספתי שתי שורות כך שאחת מטפלת בעיצוב כל הפקדים כך שתהיה אפשרות לשים בהם טקסט עם אותיות קטנות באנגלית (בלי השורה פקדי כפתורים יציגו אותיות גדולות בלבד). השורה השנייה מטפלת בכך שרקע האפליקציה יהיה ורוד תמיד בעיקר ברגע פתיחת האפליקציה, כאשר ללא השורה מראה רקע לבן בזמן הטעינה.

```
<!-- Button no longer shows only capital letters -->
<item name="android:textAllCaps">false</item>
<!-- Custom Splash Page רקע כללי ורוד בזמן טעינה בפתיחת האפליקציה -->
<item name="android:windowBackground">@drawable/splashPage</item>
```

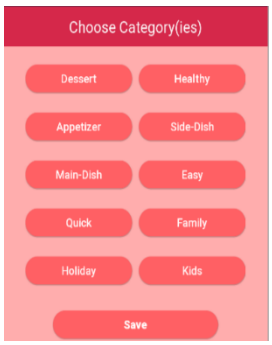
Menu

homepage_menu

קובץ xml אשר מייצג את תפריט דף הבית הכולל: סינון, יצירת מתכון, דף המתכונים האהובים, דף המתכונים שלי, הגדרות, אפשרות להתנתקות ודף רשימת משתמשים אשר מיועד למנהלים בלבד.

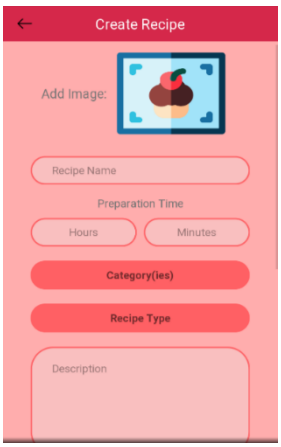
```xml
<?xml version="1.0" encoding="utf-8" ?>
<!-- תפריט המוצג בדף הבית -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- שדה מעבר לבחירת סינון מתכונים -->
  <item
    android:id="@+id/homepageMenu_sortBy"
    android:title="Sort By"/>
  <!-- שדה מעבר לדף הכנת מתכון -->
  <item
    android:id="@+id/homepageMenu_createRecipe"
    android:title="Create Recipe"/>
  <!-- שדה מעבר לדף מתכונים אהובים -->
  <item
    android:id="@+id/homepageMenu_favoriteRecipes"
    android:title="Favorite Recipes"/>
  <!-- שדה מעבר לדף מתכונים שהמשתמש יצר -->
  <item
    android:id="@+id/homepageMenu_myRecipes"
    android:title="My Recipes"/>
  <!-- שדה לפתיחת הגדרות -->
  <item
    android:id="@+id/homepageMenu_settings"
    android:title="Settings"/>
  <!-- שדה להתנתקות מהאפליקציה -->
  <item
    android:id="@+id/homepageMenu_logout"
    android:title="Logout"/>
  <!-- שדה מעבר לדף המשתמשים - למנהלים בלבד -->
  <item
    android:id="@+id/homepageMenu_userList"
    android:title="User List"/>
<menu/>
```

Layout



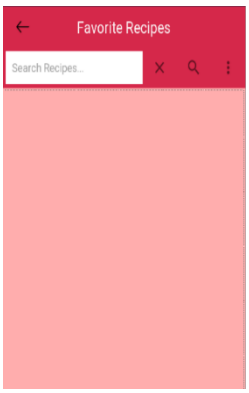
categoryDialogFragment_layout

דף בחירת הקטגוריות למטרת הצגת הקטגוריות שנבחרו או לסינון מתכונים לפי הקטגוריות. הדף מכיל 10 כפתורים המציגים את הקטגוריות השונות שניתן לבחור. ניתן לבחור עד 3 קטגוריות. ישנו כפתור לשמירת הקטגוריות ומעבר לדף המתאים. הדף מופיע בתור DialogFragment כלומר הוא לא מכסה את כל המסך כמו דף רגיל, אלא מופיע בתור טופס למילוי, כאשר ברקע ישנו הדף (Activity) הנוכחי.



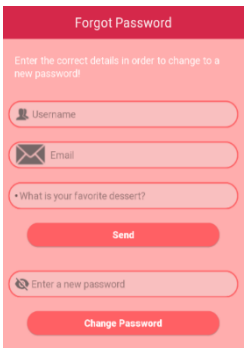
createRecipe_layout

דף ליצירת מתכון חדש אשר מכיל את כל השדות למילוי. כולל פקד תמונה להעלאת תמונת מתכון, פקדי EditText של שם המתכון, שעות ההכנה, דקות הכנה, תיאור המתכון, מרכיבי המתכון והוראות ההכנה, פקד TextView של הקטגוריות אשר ניתן ללחוץ עליו והוא מפנה לדף בחירת הקטגוריות ולאחר מכן מציג את הקטגוריות שנבחרו ופקד TextView של סוג המתכון אשר מפנה לדף בחירת סוג המתכון ולאחר מכן מציג את סוג המתכון הנבחר. בנוסף לכך ישנו LinearLayout אשר תחילה מסתיר את פרטי סוג המתכון ולאחר שנבחר הסוג מציג את הפרטים הרלוונטיים שנבחרו ע"י המשתמש. ישנו פקד כפתור אשר שומר את המתכון במערכת ומפנה לדף הבית לאחר מילוי כל פרטי המתכון. כמו כן, ישנו כפתור חזרה לדף הבית. בנוסף לכך, ישנו ScrollView הנדרש לגלילת הדף.



favoriteRecipes_layout

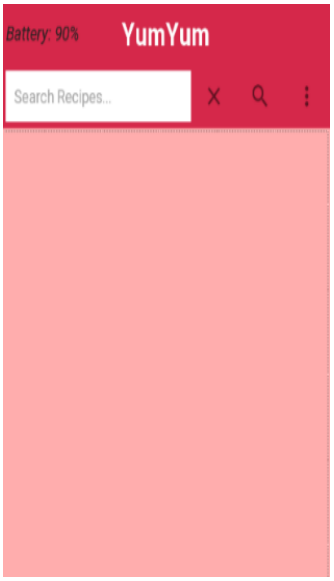
דף המתכונים האהובים של המשתמש. הדף כולל פקד AutoCompleteTextView המהווה סרגל חיפוש מתכונים אהובים. מתחת לסרגל החיפוש תופיע רשימת שמות המתכונים האהובים זאת בהתאם לסינון הנבחר. ישנם ארבעה כפתורים שונים: כפתור למחיקת הטקסט המוקלד בסרגל החיפוש, כפתור לחיפוש וסינון מתכונים לפי הטקסט המוקלד, כפתור הפותח DialogFragment אשר מראה את אפשרויות הסינון השונות וכן כפתור חזרה לדף הבית. הדף כולל ListView אשר מכיל רשומות שונות המייצגות את המתכונים האהובים של המשתמש. לחיצה על מתכון מסוים תפנה את המשתמש לדף פרטי המתכון.



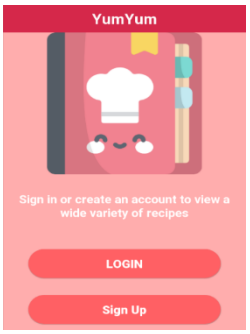
forgotPasswordDialogFragment_layout

דף שכחתי סיסמא. את הדף ניתן לפתוח בדף ההתחברות במקרה והמשתמש שכח סיסמא. הדף מכיל שלושה פקדי EditText שמייצגים את שדות המילוי של השם הפרטי, האימייל ותשובה לשאלת האבטחה. לאחר שמולאו הפרטים התקינים, LinearLayout המכיל את שדות שינוי הסיסמא יפתח ויוצג למשתמש. הוא מכיל פקד EditText המייצג את שדה הסיסמא החדשה וכפתור אישור לשינוי סיסמא ומעבר לדף הבית.

homepage_layout

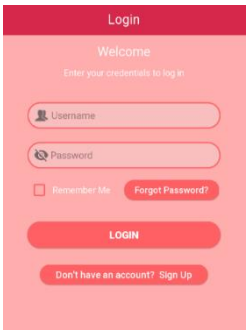


דף הבית של האפליקציה אשר מכיל את כל המתכונים השמורים באפליקציה. הדף כולל פקד AutoCompleteTextView המהווה סרגל חיפוש המתכונים. מתחת לסרגל החיפוש תופיע רשימת שמות כל המתכונים שיש באפליקציה זאת בהתאם לסינון הנבחר. ישנם שלושה כפתורים שונים: כפתור למחיקת הטקסט המוקלד בסרגל החיפוש, כפתור לחיפוש וסינון מתכונים לפי הטקסט המוקלד וכפתור הפותח את תפריט דף הבית אשר מקשר בין הדפים השונים באפליקציה. התפריט מכיל כפתורים למעבר לדף הכנת מתכון, לדף המתכונים האהובים, לדף המתכונים שלי, לדף ההגדרות ולדף רשימת המשתמשים (למנהלים בלבד). כמו כן, הוא מכיל כפתור לפתיחת טופס המראה את אפשרויות הסינון השונות וכן כפתור להתנתקות. הדף כולל ListView אשר מכיל רשומות שונות המייצגות את המתכונים השמורים באפליקציה (של כל המשתמשים). לחיצה על מתכון מסוים תפנה את המשתמש לדף פרטי המתכון. בנוסף לכך, לצד כותרת הדף מופיעה כמות סוללת המכשיר באמצעות BatteryBroadcastReceiver, אשר מעדכנת את כמות הסוללה ומודיעה הודעה Toast מתאימה כשהסוללה נמוכה (קטנה מ15%).



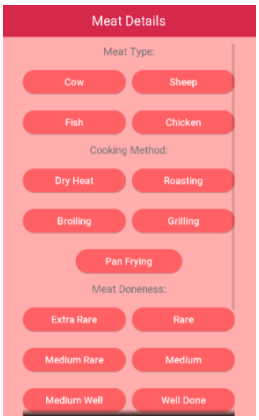
join_layout

דף ההצטרפות לאפליקציה במידה ואין משתמש מחובר במערכת. כאשר משתמש חדש נכנס לאפליקציה או משתמש מחובר אשר לא זכור במערכת נכנס חזרה לאפליקציה יופיע דף ההצטרפות אשר מכיל שני כפתורי מעבר לדף ההתחברות ולדף ההרשמה.
לוגו האפליקציה מופיע בדף כפקד תמונה.



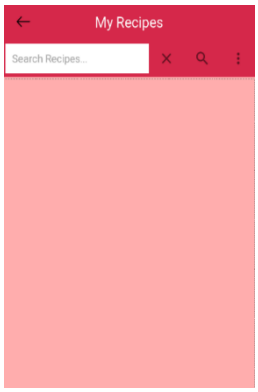
login_layout

דף ההתחברות לאפליקציה. הדף מכיל שני פקדי EditText המייצגים את שדות שם המשתמש והסיסמא. הדף מכיל כפתור שכחתי סיסמא המפנה לטופס שחזור הסיסמא. ישנו פקד CheckBox אשר קובע האם המשתמש יהיה זכור במערכת בכניסה הבאה או לא. ישנם שני כפתורים נוספים להתחברות ומעבר לדף הבית במידה ונתוני השדות תקינים וכפתור למעבר לדף ההרשמה.



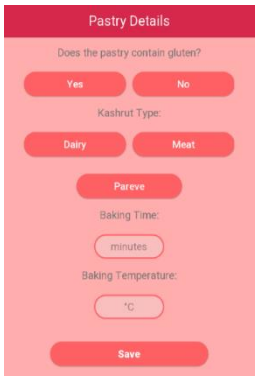
meatDialogFragment_layout

דף בחירת נתוני מנת הבשר להצגה בטקסט או לסינון מתכונים. הדף מכיל ארבעה חלקים המחולקים לפי נושא ועל המשתמש לבחור את פרטי מנת הבשר על פי כל נושא. בחלק מהנושאים על המשתמש ללחוץ על כפתור אחד, כאשר לחיצה על הכפתור משנה את צבע הטקסט של הכפתור לאדום. פרטי מנת הבשר כוללים את הנושאים: כפתורים לסוג הבשר, כפתורים לדרך הכנת הבשר, כפתורים לרמת צליית/בישול הבשר ופקד EditText לזמן בישול הבשר. בסוף הדף ישנו כפתור לשמירה לאחר שכל הפרטים נבחרו ומולאו, ולבסוף הטופס נסגר ופרטי מנת הבשר מוצגים בדף (בדף הכנת המתכון או בעת עדכון מתכון) או משמשים לסינון מתכונים בדפים הרלוונטיים. בנוסף לכך, ישנו ScrollView הנדרש לגלילת הדף.



myRecipes_layout

דף המתכונים שמשתמש המחובר יצר. הדף כולל פקד AutoCompleteTextView המהווה סרגל חיפוש מתכונים שהמשתמש הכין. מתחת לסרגל החיפוש תופיע רשימת שמות המתכונים שיצר זאת בהתאם לסינון הנבחר. ישנם ארבעה כפתורים שונים: כפתור למחיקת הטקסט המוקלד בסרגל החיפוש, כפתור לחיפוש וסינון מתכונים לפי הטקסט המוקלד, כפתור הפותח DialogFragment אשר מראה את אפשרויות הסינון השונות וכן כפתור חזרה לדף הבית. הדף כולל ListView אשר מכיל רשומות שונות המייצגות את המתכונים שהמשתמש יצר. לחיצה על מתכון מסוים תפנה את המשתמש לדף פרטי המתכון.

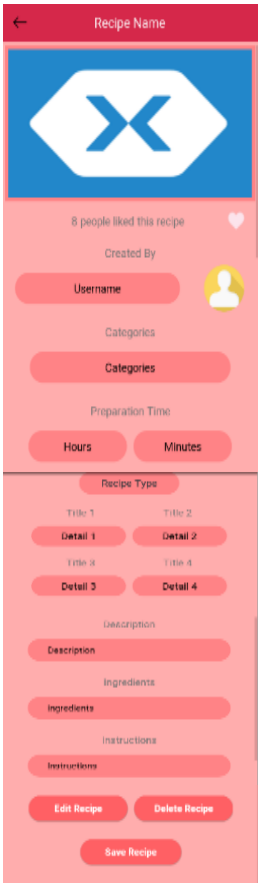


pastryDialogFragment_layout

דף בחירת נתוני המאפה להצגה בטקסט או לסינון מתכונים. הדף מכיל ארבעה חלקים המחולקים לפי נושא ועל המשתמש לבחור את פרטי המאפה על פי כל נושא. בחלק מהנושאים על המשתמש ללחוץ על כפתור אחד, כאשר לחיצה על הכפתור משנה את צבע הטקסט של הכפתור לאדום. פרטי המאפה כוללים את הנושאים: 2 כפתורים המציינים האם המאפה מכיל גלוטן, כפתורים לסוג הכשרות של המאפה ופקדי EditText לזמן וטמפרטורת האפייה. בסוף הדף ישנו כפתור לשמירה לאחר שכל הפרטים נבחרו ומולאו, ולבסוף הטופס נסגר ופרטי המאפה מוצגים בדף (בדף הכנת המתכון או בעת עדכון מתכון) או משמשים לסינון מתכונים בדפים הרלוונטיים.

recipeItem_layout

רשומת מתכון - מופיעה בתור איבר ב- ListView בדפים השונים המציגים מתכונים. איבר מתכון מכיל פקד תמונה של תמונת המתכון ושלושה פקדי TextView המייצגים את שם המתכון, קטגוריות המתכון וסוג המתכון.



recipePage_layout

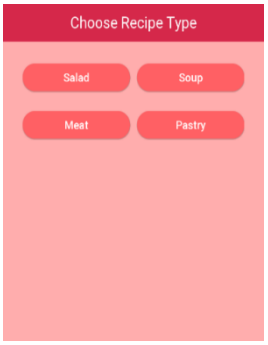
דף הצגת המתכון. הדף מכיל פקד תמונה של תמונת המתכון. ישנו פקד TextView שמראה את כמות המשתמשים שאוהבים את המתכון ופקד תמונה שמכיל אייקון לב ולמעשה מראה האם המשתמש אוהב את המתכון וניתן ללחוץ על פקד התמונה וצבע הלב ישתנה לאדום והמתכון יתווסף לדף המתכונים האהובים, לחיצה נוספת תחזיר את צבע הלב ללבן והוא ימחק מדף המתכונים האהובים.

הדף כולל גם פקד CircleImageView אשר מכיל את תמונת הפרופיל של יוצר המתכון ופקד TextView המראה את שם משתמש יוצר המתכון. ישנם פקדי TextView נוספים שמייצגים את קטגוריות המתכון, סוג המתכון, פרטי סוג המתכון וכותרות פרטי סוג המתכון.

ישנם 5 פקדי EditText שמייצגים את שעות ודקות זמן ההכנה, תיאור המתכון, המרכיבים והוראות ההכנה אשר תחילה תכונות עריכת הטקסט שלהם מושבתות וברגע עדכון מתכון תכונות אלו באות לידי ביטוי.

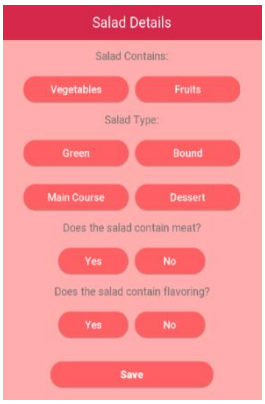
במידה והמשתמש המחובר הוא יוצר המתכון או מנהל מופיעים שני פקדי כפתור המאפשרים עדכון מתכון ומחיקת מתכון.

בעת לחיצה על עדכון, פקדי הEditText והשדות האחרים הרלוונטיים וניתנים לשינוי משנים את צבעם וסגנונם כך שהמשתמש יידע שניתן לעדכן אותם, כמו בדף הכנת מתכון. בעת עדכון, 2 הכפתורים הללו מוסתרים ומופיע כפתור נוסף לאישור עדכון המתכון. בנוסף לכך, ישנו ScrollView הנדרש לגלילת הדף.



recipeTypeDialogFragment_layout

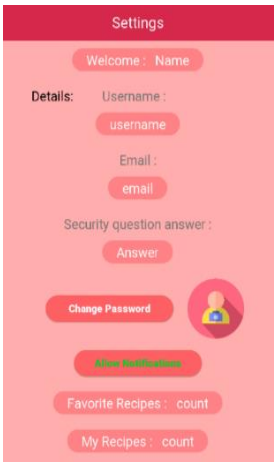
דף בחירת סוג מתכון. הדף כולל ארבעה פקדי כפתור המייצגים את ארבעת סוגי המתכון שבאפליקציה: סלט, מרק, מנת בשר ומאפה. לחיצה על אחד מסוגי המתכון תפתח טופס למילוי ובחירת פרטי סוג המתכון הנבחר למטרות סינון על פי הפרטים או הצגת הפרטים וסוג המתכון. הדפים הם: סלט – saladDialogFragment, מרק – soupDialogFragment, מנת בשר – meatDialogFragment, מאפה – pastryDialogFragment.



saladDialogFragment_layout

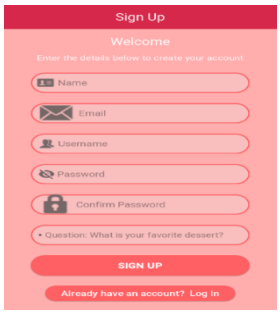
דף בחירת נתוני הסלט להצגה בטקסט או לסינון מתכונים. הדף מכיל ארבעה חלקים המחולקים לפי נושא ועל המשתמש לבחור את פרטי הסלט על פי כל נושא. בכל נושא על המשתמש ללחוץ על כפתור אחד, כאשר לחיצה על הכפתור משנה את צבע הטקסט של הכפתור לאדום. פרטי הסלט כוללים את הנושאים: 2 כפתורים המציינים סלט ירקות או פירות, כפתורים לסוג הסלט, 2 כפתורים המציינים האם הסלט מכיל בשר ו2 כפתורים המציינים האם הסלט מכיל טיבול. בסוף הדף ישנו כפתור לשמירה לאחר שכל הפרטים נבחרו ומולאו, ולבסוף הטופס נסגר ופרטי הסלט מוצגים בדף (בדף הכנת המתכון או בעת עדכון מתכון) או משמשים לסינון מתכונים בדפים הרלוונטיים.

settingsDialogFragment_layout



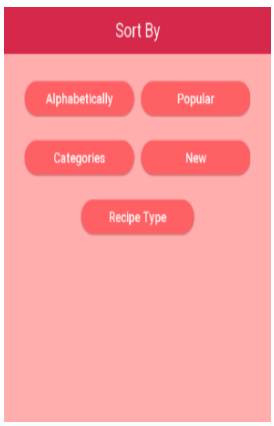
דף הגדרות המשתמש. הדף כולל שישה פקדי TextView אשר מציגים את פרטי המשתמש הנ"ל: שם פרטי, שם משתמש, אימייל המשתמש, תשובה על שאלת האבטחה, כמות המתכונים האהובים וכמות המתכונים שיצר.
הדף כולל פקד ImageView של תמונת פרופיל המשתמש, כאשר לחיצה על התמונה מאפשרת העלאה של תמונת פרופיל או מחיקתה. כמו כן, הדף כולל כפתור לאפשרות שינוי סיסמא כאשר לחיצה עליו מציגה LinearLayout מוסתר, אשר כולל פקד EditText לכתיבת סיסמא חדשה ופקד ImageButton, שכולל תמונה של חץ, לאישור שינוי הסיסמא. בנוסף לכך, ישנו כפתור להפעלת/השבתת שירות ההודעות של האפליקציה אשר מציג טקסט ירוק כאשר השירות פועל ומציג טקסט אדום כאשר השירות מושבת. לחיצה על הכפתור משנה את צבע הטקסט ומפעילה/משביתה את השירות.

signup_layout



דף ההרשמה לאפליקציה. הדף מכיל שישה פקדי EditText המייצגים את שדות השם הפרטי, אימייל המשתמש, שם המשתמש, הסיסמא, אישור הסיסמא ותשובה על שאלת אבטחה (מהו המתכון האהוב ביותר). ישנם שני כפתורים להרשמה ומעבר לדף הבית במידה ונתוני השדות תקינים וכפתור למעבר לדף ההתחברות.

sortByDialogFragment_layout

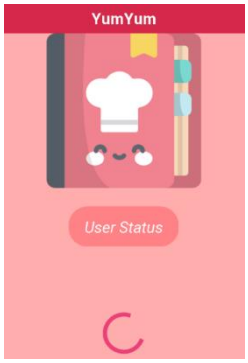


דף בחירת אמצעי סינון המתכונים. הדף מכיל חמישה כפתורים אשר מייצגים את אפשרויות סינון המתכונים השונות: לפי סדר אלפביתית, לפי פופולריות המתכון, לפי קטגוריות המתכון, לפי מתכונים חדשים ולפי סוג מתכון.
כאשר כפתור הסינון לפי קטגוריות נלחץ נפתח טופס חדש לבחירת הקטגוריות (categoryDialogFragment_layout) שעל פיהם המתכונים יסוננו.
כאשר כפתור הסינון לפי סוג מתכון נלחץ נפתח טופס לבחירת סוג מתכון (recipeTypeDialogFragment_layout) ולאחר בחירת סוג המתכון יפתח דף פרטי סוג המתכון הרלוונטי שעל פיו המתכונים יסוננו.
בחירה בשלושת אפשרויות הסינון האחרות יציגו ישירות את דף המתכונים (דף הבית, דף המתכונים האהובים או דף המתכונים שלי) מסונן.

soupDialogFragment_layout

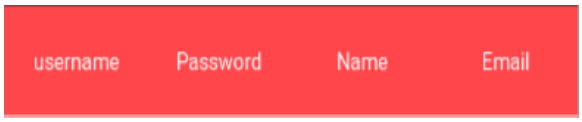
דף בחירת נתוני המרק להצגה בטקסט או לסינון מתכונים. הדף מכיל ארבעה חלקים המחולקים לפי נושא ועל המשתמש לבחור את פרטי המרק על פי כל נושא. בחלק מהנושאים על המשתמש ללחוץ על כפתור אחד, כאשר הלחיצה על הכפתור משנה את צבע הטקסט של הכפתור לאדום. פרטי המרק כוללים את הנושאים: כפתורים לסוג המרק, 2 כפתורים המציינים בישול ארוך או קצר ופקדי EditText לזמן וטמפרטורת הבישול. בסוף הדף ישנו כפתור לשמירה לאחר שכל הפרטים נבחרו ומולאו, ולבסוף הטופס נסגר ופרטי המרק מוצגים בדף (בדף הכנת המתכון או בעת עדכון מתכון) או משמשים לסינון מתכונים בדפים הרלוונטיים.

startPage_layout



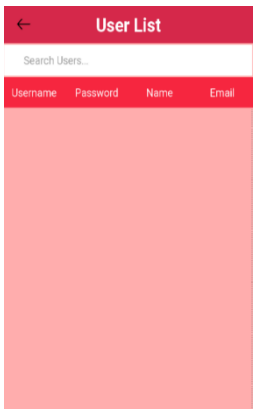
דף טעינת הפתיחה של האפליקציה. זהו הדף הראשון שרואים בכל פתיחה חדשה של האפליקציה. הדף כולל פקד תמונה של אייקון האפליקציה. ישנו פקד TextView שתפקידו להציג את סטטוס המשתמש, כלומר אם המשתמש נכנס לראשונה לאפליקציה או היה מחובר אך לא זכור במערכת תופיע הודעה Welcome to YumYum; במידה והמשתמש היה מחובר למערכת והוא זכור בה תופיע ההודעה Welcome back 'username'.
ישנו פקד ProgressBar אשר מסתובב בזמן הטעינה עד למעבר לדף ההצטרפות או לדף הבית לפי סטטוס המשתמש.

userItem_layout



רשומת משתמש - מופיעה בתור איבר ב- ListView בדף רשימת המשתמשים הרשומים לאפליקציה. איבר משתמש מכיל ארבעה פקדי TextView המייצגים את שם המשתמש, הסיסמא, השם הפרטי ואימייל המשתמש.

userList_layout



דף רשימת המשתמשים - גישה למנהל בלבד. הדף כולל פקד AutoCompleteTextView המהווה סרגל חיפוש משתמשים לפי שם המשתמש. הדף כולל ארבעה פקדי TextView של כותרות פרטי המשתמש: שם משתמש, סיסמא, שם פרטי ואימייל.
הדף כולל ListView אשר מכיל רשומות שונות המייצגות את המשתמשים הרשומים לאפליקציה. לחיצה על רשומת משתמש מסוים תפנה את המנהל לדף הגדרות המשתמש. לחיצה ארוכה על רשומת משתמש מאפשרת למנהל לקדם/להוריד את המשתמש לדרגת מנהל/משתמש רגיל, וכן למחוק את המשתמש ממסד הנתונים.
כמו כן, ישנו כפתור חזרה לדף הבית.

Activity

StartPageActivity

משויך ל-startPage_layout. האקטיביטי ברגע ייווצרו יוצר את טבלאות מסד הנתונים במידה ואין אינן קיימות ברגע פתיחת האפליקציה באמצעות מחלקת עזר (כניסה ראשונה). לאחר מכן הוא יוצר קובץ פנימי בזיכרון המכשיר באמצעות הממשק ISharedPreferences בעזרת מחלקת עזר נוספת במטרה לשמור את נתוני המשתמש המחובר במידה והקובץ אינו קיים (כניסה ראשונה). לאחר מכן ישנה בדיקה האם המשתמש זכור במערכת בעזרת המידע השמור בקובץ הנ"ל, במידה הוא לא זכור נתוני המשתמש בקובץ הפנימי מתאפסים, תופיע הודעה כללית מתאימה והדף בסוף יעבור לדף ההצטרפות. במידה והמשתמש מחובר וזכור במערכת תופיע הודעה מתאימה עם שם המשתמש הנ"ל והדף בסוף יעבור לדף הבית. בנוסף, ישנה בדיקה אם גרסת המכשיר גדולה או שווה ל-Oreo בכדי ליצור NotificationChannel להפעלת סרוויס ההודעות אשר מצריך זאת בגרסאות חדשות אלו לשם הצגת הודעה במכשיר. ישנו שימוש במחלקת Timer אשר מאפשרת טעינה של הדף ל2 שניות ומעבר לדף המתאים.

```
[Activity(Label = "YumYum", Theme = "@style/AppTheme", MainLauncher = true, Icon
 ="@drawable/yumyumimage")]
 public class StartPageActivity : Activity
 {
     // פקד TextView הרושם את הודעת הפתיחה בהתאמה
     TextView tvStatus;
     // משתנה המסוגל לשמור סוג טיפוס, שומר את האקטיביטי אליו יעבור הדף
     Type typeofNextActivity;
     // אובייקט מטיפוס טיימר
     Timer timer;
     protected override void OnCreate(Bundle savedInstanceState)
     {
         base.OnCreate(savedInstanceState);
         SetContentView(Resource.Layout.startPage_layout);

         // יצירת טבלאות משתמשים ומתכונים בכניסה לאפליקציה במידה והן אינן קיימות
         DatabaseManager.CreateTables();

         // Create ISharedPreferences using helper class when entering the app, initialized once
            when opening the app
         SharedPreferencesManager.CreateSharedPreferences(ApplicationContext);
         tvStatus = FindViewById<TextView>(Resource.Id.startPage_tvStatus);
         // Checking the status of user
         // בדיקה אם המערכת זוכרת את המשתמש באמצעות לקיחת המידע מקובץ בזכרון"ISharedPreferences"
            באמצעות פעולה ממחלקת עזר
         if (SharedPreferencesManager.GetRememberMe() == false)
         {
             // איפוס נתוני המשתמש השמורים בקובץ השמור בזכרון כאשר המשתמש לא זכור במכשיר או נכנס בפעם הראשונה
             SharedPreferencesManager.ResetPreferences();

             // JoinActivity המשתנה מצביע על סוג טיפוס
             typeofNextActivity = typeof(JoinActivity);
             // הצגת הודעת ברוכים הבאים
             tvStatus.Text = "Welcome to YumYum";
         }
         else
         {
             // HomepageActivity המשתנה מצביע על סוג טיפוס
             typeofNextActivity = typeof(HomepageActivity);
             // הצגת הודעה מתאימה ולקיחת שם המשתמש מקובץ "ISharedPreferences"
             tvStatus.Text = "Welcome back " + SharedPreferencesManager.GetUsername();
         }

         // בדיקה אם גרסת המכשיר גדולה או שווה לגרסת Oreo
         if (Build.VERSION.SdkInt >= BuildVersionCodes.O)
         {
             // בכדי ששירות ההודעות יפעל בגרסאות אלו יש ליצור NotificationChannel במידה ואינו קיים
             // יצירתו היא חובה מגרסת Oreo ומעלה לשם הצגת Notification
             NotificationChannel notificationChannel = new
             NotificationChannel("Channel_StatusNotification", "Status Notification Channel",
             NotificationImportance.Default);
             NotificationManager notificationManager =
             (NotificationManager)GetSystemService(NotificationService);
             notificationManager.CreateNotificationChannel(notificationChannel);
         }
         // יצירת אובייקט מטיפוס טיימר
         timer = new Timer();
         // delay for 2 seconds
         timer.Interval = 2000;
         // לאחר שהטיימר מפסיק הפעולה מתממשת
         timer.Elapsed += delegate
         {
             // Stops at first tick which takes 2 seconds
             timer.Stop();

             // מעבר לאקטיביטי הבא הנקבע ע"י typeofNextActivity
             Intent intent = new Intent(this, typeofNextActivity);
             StartActivity(intent);
             Finish();
         };
         // התחלת הטיימר
         timer.Start();
     }
}
```

JoinActivity

משויך ל-join_layout. האקטיביטי כולל קישור לשני כפתורים למעבר לדף ההתחברות ולדף ההרשמה.

LoginActivity

משויך ל-login_layout. האקטיביטי של דף ההתחברות כולל פעולה הבודקת ברגע הלחיצה על כפתור ההתחברות את תקינות טקסט הפקדים שמייצגים את שם המשתמש והסיסמא ובעזרת מחלקת עזר בודק האם המשתמש הנ"ל קיים במסד הנתונים. במידה והוא לא קיים תופיע הודעת AlertDialog מתאימה. במידה והוא קיים הפעולה מוסיפה את שם המשתמש ואת הסטטוס (האם הוא זכור במערכת או לא), לקובץ פנימי בזיכרון המכשיר באמצעות מחלקת עזר והממשק ISharedPreferences.
כמו כן ישנה בדיקה שהמשתמש הנ"ל מאשר את שירות ההודעות של האפליקציה ואם כן תופיע הודעה מתאימה על התחברותו למערכת. פעולת העזר StartNotificationService משומשת בכמה Activities והיא שולחת לסרוויס טרם הפעלתו מערך מחרוזות המכיל את הכותרת וההודעה המתאימה שתוצג בסופו של דבר. לאחר ההתחברות ישנו מעבר לדף הבית. בנוסף לכך, ברגע לחיצה על כפתור שחכתי סיסמא ישנה פעולה הפותחת את הDialogFragment המתאים לשם הצגת טופס מילוי לשחזור סיסמא.

```
[Activity(Label = "LoginActivity")]
public class LoginActivity : Activity
{
    // פקד EditText של שם המשתמש
    private EditText etUsername;
    // פקד EditText של הסיסמא
    private EditText etPassword;
    // פקד CheckBox לשמירת המשתמש במערכת בכניסה הבאה
    private CheckBox cbRememberMe;
    // פקד כפתור להתחברות
    private Button btnLogin;
    // פקד כפתור לשכחתי סיסמא
    private Button btnForgotPassword;
    // כפתור מעבר לדף ההרשמה
    private Button btnGotoSignup;
    // לחיצה על כפתור התחברות
    private void BtnLogin_Click(object sender, EventArgs e)
    {
        // בדיקה האם במסד הנתונים ישנו משתמש שפרטי שם המשתמש והסיסמא תואמים
        if (DatabaseManager.IsValidUser(etUsername.Text, etPassword.Text))
        {
            // שמירת נתוני שם המשתמש וסטטוס המשתמש באמצעות ISharedPreferences
            // סטטוס המשתמש נקבע על פי הCheckBox
            SharedPreferencesManager.SetUsername(etUsername.Text);
            SharedPreferencesManager.SetRememberMe(cbRememberMe.Checked);
            // StatusNotificationService
            // בדיקה שהמשתמש מרשה הודעות
            if (DatabaseManager.GetUser(etUsername.Text).allowNotification)
            {
                // הפעלת שירות הודעות לאחר התחברות
                StartNotificationService();
            }
            // סגירת דף ההתחברות ומעבר לדף הבית
            Intent intent = new Intent(this, typeof(HomepageActivity));
            StartActivity(intent);
            Finish();
        }
        else
        {
            // הצגת הודעת שגיאה מתאימה
            ShowAlertDialog();
        }
    }
    // פעולה המתחילה את שירות ההודעות ושולחת תוכן הודעה מתאים
    private void StartNotificationService()
    {
        // הפעלת שירות ההודעות על פי הCheckBox ע"י שליחת תוכן הודעה מתאים לסרוויס
        Intent serviceIntent = new Intent(this, typeof(StatusNotificationService));
        string[] loginDetails;
        // בדיקה האם המשתמש לחץ על הCheckBox וזכור במערכת, תוכן ההודעה יופיע בהתאמה
        if (cbRememberMe.Checked)
        {
            loginDetails = new string[] { "Login Completed Successfully", etUsername.Text + ",
            you are remembered in the system" };
        }
        else
        {
            loginDetails = new string[] { "Login Completed Ssuccessfully", etUsername.Text + ",
            you are not remembered in the system" };
        }
        serviceIntent.PutExtra("StatusNotification_Details", loginDetails);
        StartService(serviceIntent);
    }

    // פתיחת ForgotPasswordDialogFragment להצגת טופס שינוי סיסמא
    private void BtnForgotPassword_Click(object sender, EventArgs e)
    {
        FragmentTransaction transaction = FragmentManager.BeginTransaction();
        ForgotPasswordDialogFragment forgotPasswordDialogFragment = new
        ForgotPasswordDialogFragment();
        forgotPasswordDialogFragment.Show(transaction, "Forgot Password dialog fragment");
    }
{
```

SignupActivity

משויך ל-signup_layout. האקטיביטי של דף ההרשמה כולל פעולות עזר לשם בדיקה תקינה של השדות שיש למלא. בפרט נעשה שימוש במחלקה הסטטית Regex לבדיקה של מחרוזות והאם הן מכילות מספרים ואותיות באנגלית בלבד. כמו כן, נעשה שימוש גם במחלקה המובנית MailAddress לשם בדיקה פשוטה של האימייל האם הוא תקין מבחינה דקדוקית. ברגע הלחיצה על כפתור ההרשמה נעשית בדיקה אם ישנם משתמשים קיימים באפליקציה; אם אין אזי מדובר במשתמש הראשון וייווצר משתמש שתכונת המנהל הבוליאנית שלו תהיה אמת, אם קיימים כבר משתמשים באפליקציה תכונת המנהל של המשתמש החדש שנוצר תהיה שקר. לאחר מכן שם המשתמש וסטטוס המשתמש מוכנסים לקובץ פנימי באמצעות הממשק ISharedPreferences כך סטטוס המשתמש יהיה true כלומר הוא יהיה זכור במערכת בכניסה הבאה לאפליקציה. לאחר מכן מופעל הסרוויס ומופיעה הודעת הרשמה מתאימה וישנו מעבר לדף הבית.

```
[Activity(Label = "SignupActivity")]
    public class SignupActivity : Activity
    {
        // פקד EditText של שם פרטי
        private EditText etName;
        // פקד EditText של אימייל
        private EditText etEmail;
        // פקד EditText של שם המשתמש
        private EditText etUsername;
        // פקד EditText של הסיסמא
        private EditText etPassword;
        // פקד EditText של אישור הסיסמא
        private EditText etConfirmPassword;
        // פקד EditText של תשובה לשאלת האבטחה
        private EditText etSecurityQuestion;
        // פקד כפתור להרשמה
        private Button btnSignup;
        // כפתור מעבר לדף ההתחברות
        private Button btnGotoLogin;
        // מחרוזת עם המידע המתאים ברגע אי תקינות הנתונים
;private string errorText

        // פעולה בוליאנית הבודקת אם הנתונים תקינים
        private bool IsValid()
        {
            // Name Validation
            if (!IsValidName())
            {
                return false;
            }
            // Email Validation
            if (!IsValidEmail())
            {
                return false;
            }

            // Username Validation
            if (!IsValidUsername())
            {
                return false;
            }
            // Password Validation
            if (!IsValidPassword())
            {
                return false;
            }
            // Security Question Validation
            if (!IsValidSecurityQuestion())
            {
                return false;
            }
            return true;
        }
        // בדיקת תקינות לשם פרטי
        private bool IsValidName()
        {
            // Name Validation
            // בדיקה שהם לא ריק
            if (etName.Text == "")
            {
                errorText = "Insert name";
                return false;
            }
            // בדיקה אם השם מכיל אך ורק אותיות לועזיות
            // פעולה שעוברת על המחרוזת ובודקת שהיא מכילה רק אותיות באנגלית
            if (!Regex.IsMatch(etName.Text, "^[a-zA-Z]+$"))
            {
                errorText = "Name must contain English letters only";
                return false;
            }
            return true;
        }
        // בדיקת תקינות לאימייל
        private bool IsValidEmail()
        {
            // Email Validation
            // בדיקה שהאימייל לא ריק
            if (etEmail.Text == "")
            {
                errorText = "Insert email";
                return false;
            }
            // בדיקה שהאימייל לא תפוס באפליקציה
            if (DatabaseManager.IsExistEmail(etEmail.Text)
```

```
{
                errorText = "Email is already taken";
                return false;
            }
            // בדיקה בסיסית אם האימייל תקין מבחינת דקדוקית
            try
            {
                // MailAddress בעזרת מחלקת עזר מובניתי יוצר טיפוס
                MailAddress addr = new MailAddress(etEmail.Text);
                // מחזיר ערך בוליאני true אם המייל תקין, כאשר לא מתקיים יוצר קריסה
                return addr.Address == etEmail.Text;
            }
            // catch ו try במקרה של אי תקינות גורם לקריסה ולכן יש להשתמש ב
            catch
            {
                errorText = "Invalid email";
                return false;
            }

        }
        // בדיקת תקינות לשם משתמש
        private bool IsValidUsername()
        {
            // Username Validation
            // בדיקה ששם המשתמש לא ריק
            if (etUsername.Text == "")
            {
                errorText = "Insert username";
                return false;
            }
            // בקובץ axml יש בדיקה ששם המשתמש לא יכיל יותר מ15 תווים
            // בדיקה שאורך שם המשתמש בין 3 ל15 תווים
            if (etUsername.Text.Length < 3)
            {
                errorText = "Username length must be between 3 - 15 letters";
                return false;
            }
            // בדיקה ששם המשתמש יכיל אך ורק אותיות ומספרים
            // פעולה שעוברת על כל המחרוזת ובודקת שהיא מכילה רק מספרים ואותיות באנגלית
            if (!Regex.IsMatch(etUsername.Text, "^[a-zA-Z0-9]+$"))
            {
                errorText = "Username must contain English letters and numbers only";
                return false;
            }
            // בדיקה אם שם המשתמש מכיל לפחות אות באנגלית אחת
            // Match a single alphabetic character(a through z or A through Z)
            if (!Regex.IsMatch(etUsername.Text, "[a-zA-Z]"))
            {
                errorText = "Username must contain at least one English letter";
                return false;
            }
            // בדיקה בעזרת מחלקת עזר ששם המשתמש לא קיים במערכת
            if (DatabaseManager.IsExistUsername(etUsername.Text))
            {
                errorText = "Username already exists";
                return false;
            }
            return true;
        }
        // בדיקת תקינות לסיסמא
        private bool IsValidPassword()
        {
            // Password Validation
            // בדיקה שהסיסמא לא ריקה
            if (etPassword.Text == "")
            {
                errorText = "Insert password";
                return false;
            }
            // בקובץ axml יש בדיקה שהסיסמא לא תכיל יותר מ16 תווים
            // בדיקה שאורך הסיסמא בין 8 ל16 תווים
            if (etPassword.Text.Length < 8)
            {
                errorText = "Password length must be between 8 - 16 letters";
                return false;
            }
            // בדיקה שהסיסמא תכיל אך ורק אותיות ומספרים
            // פעולה שעוברת על כל המחרוזת ובודקת שהיא מכילה רק מספרים ואותיות באנגלית
            if (!Regex.IsMatch(etPassword.Text, "^[a-zA-Z0-9]+$"))
            {
                errorText = "Password must contain English letters and numbers only";
                return false;
            }
```

```
// בדיקה אם הסיסמא מכילה לפחות אות באנגלית אחת ומספר אחד
            // Match a single alphabetic character(a through z or A through Z) and numeric character
            if (!(Regex.IsMatch(etPassword.Text, "[a-zA-Z]") && Regex.IsMatch(etPassword.Text, "[0-9])))
            {
                errorText = "Password must contain at least one English letter and one number";
                return false;
            }
            // Password Confirmation Validation
            // בדיקה שאישור הסיסמא תואם לסיסמא
            if (!etPassword.Text.Equals(etConfirmPassword.Text))
            {
                errorText = "Invalid password confirmation";
                return false;
            }
            return true;
        }

        // בדיקת תקינות לתשובה על שאלת האבטחה
        private bool IsValidSecurityQuestion()
        {
            // Security Question Validation
            // בדיקה שהתשובה לא ריקה
            if (etSecurityQuestion.Text == "")
            {
                errorText = "Answer the security question";
                return false;
            }
            // בקובץ axml יש בדיקה שהתשובה לא תכיל יותר מ20 תווים
            // בדיקה שאורך הסיסמא בין 8 ל16 תווים
            if (etSecurityQuestion.Text.Length < 3)
            {
                errorText = "Security question answer must contain at least 3 letters";
                return false;
            }
            // בדיקה שהסיסמא תכיל אך ורק אותיות ומספרים
            // פעולה שעוברת על כל המחרוזת ובודקת שהיא מכילה רק מספרים ואותיות באנגלית
            if (!Regex.IsMatch(etSecurityQuestion.Text, "^[a-zA-Z]+$"))
            {
                errorText = "Security question answer must contain English letters only";
                return false;
            }
            return true;
}

        private void BtnSignup_Click(object sender, EventArgs e)
        {
            // בדיקה האם נתוני ההרשמה תקינים
            if (IsValid())
            {
                // עצם משתמש שיצביע על משתמש חדש שנוצר
                User user;
                // בדיקה האם מספר המשתמשים באפליקציה שווה ל0
                if (DatabaseManager.GetAllUsers().Count == 0)
                {
                    // אם כן יווצר משתמש ראשון שתכונת isAdmin תהיה אמת והוא יהווה כמנהל
                    user = new User(etUsername.Text, etPassword.Text, etName.Text, etEmail.Text,
                    etSecurityQuestion.Text, true);
                }
                else
                {
                    // אם כן יווצר משתמש שתכונת isAdmin תהיה שקר והוא יהווה כמשתמש רגיל
                    user = new User(etUsername.Text, etPassword.Text, etName.Text, etEmail.Text,
                    etSecurityQuestion.Text, false);
                }
                // הכנסת המשתמש למסד הנתונים
                DatabaseManager.AddUser(user);
                // שמירת נתוני שם המשתמש וסטטוס המשתמש באמצעות ISharedPreferences
                SharedPreferencesManager.SetUsername(etUsername.Text);
                // לאחר הרשמה המערכת זוכרת את המשתמש
                SharedPreferencesManager.SetRememberMe(true);
                // StatusNotificationService
                // התחלת שירות ההודעות לאחר הרשמה
                StartNotificationService();
                // סגירת דף ההרשמה ומעבר לדף הבית
                Intent intent = new Intent(this, typeof(HomepageActivity));
                StartActivity(intent);
                Finish();
            }
            else
            {
                // הצגת הודעת שגיאה מתאימה
                ShowAlertDialog();
            }
}
```

HomepageActivity

משויך ל-homepage_layout. האקטיביטי של דף הבית כולל קישור של פקד כפתור לhomepage_menu כך שלחיצה עליו תפתח את התפריט הכולל קישורים לכל דפי האפליקציה. כמו כן, ישנה בדיקה האם המשתמש המחובר הוא מנהל בכדי שהתפריט יכלול גם קישור לדף רשימת המשתמשים. ברגע של התנתקות, נתוני המשתמש בקובץ הפנימי מתאפסים וישנו ביטול הסרוויס ושירות ההודעות. הדף מכיל רשימת מתכונים של כל המתכונים באפליקציה ורשימת מתכונים אשר מתעדכנת ובעצם מעדכנת את הListView המציג את רשומות המתכונים. בפעולה OnStart ישנה הצגה של המתכונים כך שישנה שליפה של כל המתכונים ממסד הנתונים והשמתם ברשימה ולאחר מכן ישנו עדכון של הListView המציג את רשומות המתכונים באמצעות עדכון RecipeAdapter. הבחירה בהשמת הפעולה בOnStart היא בכדי לעדכן את דף הבית לאחר סיום הכנת מתכון או עדכון ומחיקה של מתכון מסוים. כמו כן ישנו עדכון של רשימת שמות המתכונים אשר מופיעה מתחת לפקד סרגל החיפוש ברגע כתיבת טקסט. בפעולה OnResume המופעלת לאחר OnStart ישנו חידוש האזנה מה-IntentFilter אשר batteryBroadcastReceiver מאזין לו לשם עדכון אחוז הסוללה. הפעולה OnPauseמופעלת כאשר הדף נמצא ברקע ומבטלת את ההאזנה. ישנה פעולה המעדכנת ומסננת את רשימות המתכונים על ידי קליטה של רשימה מסוננת והשמת ערכיה ברשימות המתכונים שבמחלקה ועדכון הListView המציג את רשומות המתכונים. כמו כן, בעת סינון ע"פ קטגוריות או סוג מתכון, ישנן פעולות המקבלות מחרוזות ומסננות את הרשימה על פי הנתונים המתאימים. בעת לחיצה על כפתור חיפוש מתכונים ישנו סינון של המתכונים על פי הטקסט המוקלד. בנוסף לכך, המחלקה מממשת את הממשק AdapterView.IOnItemClickListener המאפשר לחיצה על רשומת מתכון ומעבר לדף המתכון המתאים.

```
[Activity(Label = "HomepageActivity")]
    public class HomepageActivity : AppCompatActivity, AdapterView.IOnItemClickListener
    {
        // רשימה המכילה את כל המתכונים ששמורים במסד הנתונים או את המתכונים לפי הסינון
        private List<Recipe> recipeList;
        // רשימת המתכונים האהובים המתעדכנת לפי סינון או חיפוש
        private List<Recipe> currentRecipes;
        // currentRecipes אדפטר של המתכונים הנוכחים לפי הרשימה ב-
        private RecipeAdapter recipeAdapter;
        // recipeAdapter המציג את המתכונים הנוכחים לפי ListView
        private ListView lvRecipes;
        // רשימה המכילה את שמות כל המתכונים או את שמות המתכונים לפי הסינון
        private List<string> recipeNameList;
        // אדפטר של רשימת שמות המשתמשים אשר בעזרתו יוצגו מתחת לסרגל החיפוש בעת כתיבת טקסט
        private ArrayAdapter recipeNameAdapter;
        // סרגל החיפוש אשר ניתן לכתוב בו טקסט ומתחת יופיעו אפשרויות של מתכונים לפי הטקסט
        private AutoCompleteTextView actvSearch;
        // כפתור מחיקת חיפוש
        private Button btnClear;
        // כפתור חיפוש מתכונים על פי הטקסט שנכתב בסרגל החיפוש
        private Button btnSearch;
        // כפתור אשר פותח את התפריט של הדף הראשי
        private Button btnMenu;
        // BatteryBroadcastReceiver
        // פקד TextView אשר מראה את כמות הסוללה במכשיר
        private TextView tvBattery;
        // BatteryBroadcastReceiver אובייקט מטיפוס
        private BatteryBroadcastReceiver batteryBroadcastReceiver;

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.homepage_layout);
            // BatteryBroadcastReceiver
            tvBattery = FindViewById<TextView>(Resource.Id.homepage_tvBattery);
            // BatteryBroadcastReceiver יצירת מופע של
            batteryBroadcastReceiver = new BatteryBroadcastReceiver(tvBattery);
            lvRecipes = FindViewById<ListView>(Resource.Id.homepage_lvRecipes);
            lvRecipes.OnItemClickListener = this; // Click on view - recipe
            actvSearch = FindViewById<AutoCompleteTextView>(Resource.Id.homepage_actvSearch);
            btnClear = FindViewById<Button>(Resource.Id.homepage_btnClear);
            btnSearch = FindViewById<Button>(Resource.Id.homepage_btnSearch);
            btnMenu = FindViewById<Button>(Resource.Id.homepage_btnMenu);

            // מגדיר את הפקד כמפעיל את התפריט
            RegisterForContextMenu(btnMenu);

            btnClear.Click += BtnClear_Click;
            btnSearch.Click += BtnSearch_Click;
            btnMenu.Click += BtnMenu_Click;
        }
```

```
/* actvSearch-פעולת עזר שמטרתה לעדכן את האדפטר של שמות המתכונים כך ש
   יראה את שמות המתכונים הרלוונטים כאשר ישנו סינון לפי קטגוריות או סוג מתכון ובפתיחת הדף */
private void UpdateRecipeNameAdapter()
{
    // כל פעם שהפעולה מופעלת יצביע recipeNameList על רשימה חדשה
    recipeNameList = new List<string>();
    // מעבר על כל המתכונים והוספת שמות המתכונים לרשימת recipeNameList
    for (int i = 0; i < recipeList.Count; i++)
    {
        recipeNameList.Add(recipeList[i].recipeName);
    }
    // יצירת ArrayAdapter אשר מייצג אדפטר של שמות המתכונים
    recipeNameAdapter = new ArrayAdapter(this, Android.Resource.Layout.SimpleListItem1,
    recipeNameList);
    // השמת האדפטר הנ"ל ב-actvSearch.Adapter כך שסרגל החיפוש מראה את שמות המתכונים
    actvSearch.Adapter = recipeNameAdapter;
}
// פעולה שמקבלת כפרמטר אובייקט מטיפוס תפריט מתאים ודואגת שיוצג
public override void OnCreateContextMenu(IContextMenu menu, View v, IContextMenuContextMenuInfo
menuInfo)
{
    base.OnCreateContextMenu(menu, v, menuInfo);

    // זימון פעולת המערכת ש"מנפחת" ומציגה את התפריט
    MenuInflater.Inflate(Resource.Menu.homepage_menu, menu);

    // בדיקה שרק מנהל האפליקציה יוכל לגשת לרשימת המשתמשים דרך התפריט הראשי
    if (DatabaseManager.GetUser(SharedPreferencesManager.GetUsername()).isAdmin == false)
    {
        // הסרת עמודה שמפנה לתפריט הראשי כאשר המשתמש הוא לא מנהל
        menu.RemoveItem(Resource.Id.homepageMenu_userList);
    }
}

// תגובה על לחיצה על שורת תפריט כלשהי
public override bool OnContextItemSelected(IMenuItem item)
{
    // פתיחת SortByDialogFragment לבחירת סוג הסינון
    if (item.ItemId == Resource.Id.homepageMenu_sortBy)
    {
        FragmentTransaction transaction = FragmentManager.BeginTransaction();
        SortByDialogFragment sortByDialogFragment = new SortByDialogFragment();
        sortByDialogFragment.Show(transaction, "Sort By dialog fragment");
        return true;
    }
    // מעבר לדף הכנת מתכון
    if (item.ItemId == Resource.Id.homepageMenu_createRecipe)
    {
        Intent intent = new Intent(this, typeof(CreateRecipeActivity));
        StartActivity(intent);
        return true;
    }
    // מעבר לדף המתכונים האהובים
    if (item.ItemId == Resource.Id.homepageMenu_favoriteRecipes)
    {
        Intent intent = new Intent(this, typeof(FavoriteRecipesActivity));
        StartActivity(intent);
        return true;
    }
    // מעבר לדף המתכונים שלי
    if (item.ItemId == Resource.Id.homepageMenu_myRecipes)
    {
        Intent intent = new Intent(this, typeof(MyRecipesActivity));
        StartActivity(intent);
        return true;
    }
    // פתיחת SettingsDialogFragment להצגת הגדרות
    if (item.ItemId == Resource.Id.homepageMenu_settings)
    {
        //Pull up the settingsFragmentLayout (dialog)
        FragmentTransaction transaction = FragmentManager.BeginTransaction();
        // פעולה בונה שקולטת את המשתמש המחובר
        SettingsDialogFragment settingsDialogFragment = new
SettingsDialogFragment(DatabaseManager.GetUser(SharedPreferencesManager.GetUsername()));
        settingsDialogFragment.Show(transaction, "Settings dialog fragment");
        return true;
    }
    // התנתקות מהאפליקציה
    if (item.ItemId == Resource.Id.homepageMenu_logout)
    {
        // איפוס נתוני המשתמש השמורים באמצעות ISharedPreferences בעזרת מחלקת עזר
        SharedPreferencesManager.ResetPreferences();

        // ביטול שירות הודעות הסטטוס בעת התנתקות
```

```
                Intent serviceIntent = new Intent(this, typeof(StatusNotificationService));
                StopService(serviceIntent);

                // מעבר לדף Join וסגירת דף הבית
                Intent intent = new Intent(this, typeof(JoinActivity));
                StartActivity(intent);
                Finish();
                return true;
            }
            // מופיע רק כאשר מנהל מחובר ומפנה לדף רשימת המשתמשים
            if (item.ItemId == Resource.Id.homepageMenu_userList)
            {
                Intent intent = new Intent(this, typeof(UserListActivity));
                StartActivity(intent);
                return true;
            }
            return false;
        }

        /* OnCreaete() פעולה שמתרחשת לאחר
         הפעולה נקראת לאחר OnRestart() כך שאם המשתמש יצר, עדכן או מחק מתכון
         כשהוא יחזור לדף הוא יראה את השינוי */
        protected override void OnStart()
        {
            // makes the activity visible to the user, as the app prepares for the activity to enter the
               foreground and become interactive
            // this method is where the app initializes the code that maintains the UI
            base.OnStart();
            actvSearch.Text = "";
            // DatabaseManager קבלת כל המתכונים ממחלקת עזר
            recipeList = DatabaseManager.GetAllRecipes();
            // סידור המתכונים לפי סדר אלפביתי בכל פתיחה חדשה של הדף
            recipeList = (from recipe in recipeList orderby recipe.recipeName select recipe).ToList();
            // הרשימה המתעדכנת מכילה תחילה את כל המתכונים

            currentRecipes = recipeList.ToList();
            // יצירת אדפטר המתכונים והשמתו בlvRecipes.Adapter כך שהמתכונים מוצגים למשתמש
            recipeAdapter = new RecipeAdapter(this, currentRecipes);
            lvRecipes.Adapter = recipeAdapter;
            // עדכון actvSearch שיציג את כל המתכונים בהתאמה
            UpdateRecipeNameAdapter();
        }
        // פעולה המופעלת אחרי OnStart() ולפני הצגת הדף למשתמש
        protected override void OnResume()
        {
            // This is the state in which the app interacts with the user. The app stays in this state
               until something happens to take focus away from the app
            base.OnResume();
            // רישום של האובייקט batteryBroadcastReceiver תוך שליחת המופע וה- IntentFilter שממנו ה- BroadCast מאזין לו
            RegisterReceiver(batteryBroadcastReceiver, new IntentFilter(Intent.ActionBatteryChanged));
        }
        // כאשר הדף נמצא ברקע או ישנה יציאה מהאפליקציה ללא סגירתה
        protected override void OnPause()
        {
            // ביטול הרישום של batteryBroadcastReceiver כאשר האקטיביטי נכנס לרקע
            UnregisterReceiver(batteryBroadcastReceiver);
            // When an interruptive event occurs, the activity enters the Paused state
            base.OnPause();
        }
        /* פעולת עזר שמסדרת את המתכונים שבדף לפי רשימת סינון מתכונים היא קולטת
         מאחר וישנו שימוש בה במחלקות אחרות public */
        public void ArrangeRecipes(List<Recipe> sortedRecipes)
        {
            actvSearch.Text = "";
            // רשימת כל המתכונים מצביעה על רשימת המתכונים המסוננת
            recipeList = sortedRecipes;
            currentRecipes = recipeList.ToList();
            // שינוי האדפטר לפי רשימת המתכונים המתעדכנים החדשה ועדכון המתכונים המופיעים בדף
            recipeAdapter.SetRecipes(currentRecipes);
            recipeAdapter.NotifyDataSetChanged();
            // שינוי שמות המתכונים בסרגל החיפוש לפי המתכונים המסוננים
            UpdateRecipeNameAdapter();
        }

        /* פעולה שנקראת ב CategoryDialogFragment לאחר בחירת קטגוריות לסינון
         הפעולה מקבלת את כל המתכונים האהובים ומסננת כך שרשימת המתכונים
         תכיל מתכונים בעלי אותם קטגוריות כפי שקלטה */
        public void ReceiveCategories(string categories)
        {
            recipeList = DatabaseManager.GetAllRecipes();
            /* סינון הרשימה לרשימה חדשה כך שתכונת הקטגוריות של כל מתכון
             תכלול את מחרוזת הקטגוריות שהפעולה קלטה
             כלומר המתכונים יכילו רק את הקטגוריות שבמחרוזת הנ"ל, */
            recipeList = (from recipe in recipeList orderby recipe.recipeName where
```

```
        recipe.category.Contains(categories) select recipe).ToList();
        recipeList = (from recipe in recipeList orderby recipe.recipeName select recipe).ToList();
        // עדכון המתכונים המוצגים בדף
        ArrangeRecipes(recipeList);
    }
    /* פעולה הנקראת לאחר סיום בחירת נתוני סוג המתכון למטרת סינון
     הפעולה מופעלת ב SaladDialogFragment, SoupDialogFragment, MeatDialogFragment או PastryDialogFragment
     הפעולה מקבלת את סוג המתכון ומערך עם המידע על סוג המתכון לסינון */
    public void ReceiveRecipeTypeDetails(string recipeType, string[] details)
    {
        // בדיקה איזו סוג מתכון נבחר
        if (recipeType == "Salad")
        {
            // קבלת כל הסלטים
            recipeList = DatabaseManager.GetAllSalads();
            // סינון הרשימה על פי נתוני הסלטים שנבחרו
            recipeList = (from recipe in recipeList
                          orderby recipe.recipeName
                          where ((Salad)recipe).isFruit == details[0]
                          && ((Salad)recipe).saladType == details[1]
                          && ((Salad)recipe).isMeat == details[2]
                          && ((Salad)recipe).isFlavoring == details[3]
                          select recipe).ToList();
        }
        if (recipeType == "Soup")
        {
            // קבלת כל המרקים
            recipeList = DatabaseManager.GetAllSoups();
            // סינון הרשימה על פי נתוני המרקים שנבחרו
            recipeList = (from recipe in recipeList
                          orderby recipe.recipeName
                          where ((Soup)recipe).soupType == details[0]
                          && ((Soup)recipe).duration == details[1]
                          select recipe).ToList();
        }

        if (recipeType == "Meat")
        {
            // קבלת כל הבשרים
            recipeList = DatabaseManager.GetAllMeats();
            // סינון הרשימה על פי נתוני הבשרים שנבחרו
            recipeList = (from recipe in recipeList
                          orderby recipe.recipeName
                          where ((Meat)recipe).meatType == details[0]
                          && ((Meat)recipe).method == details[1]
                          && ((Meat)recipe).doneness == details[2]
                          select recipe).ToList();
        }
        if (recipeType == "Pastry")
        {
            // קבלת כל המאפים
            recipeList = DatabaseManager.GetAllPastries();
            // סינון הרשימה על פי נתוני המאפים שנבחרו
            recipeList = (from recipe in recipeList
                          orderby recipe.recipeName
                          where ((Pastry)recipe).isGluten == details[0]
                          && ((Pastry)recipe).kosher == details[1]
                          select recipe).ToList();
        }
        recipeList = (from recipe in recipeList orderby recipe.recipeName select recipe).ToList();
        // עדכון המתכונים המוצגים בדף
        ArrangeRecipes(recipeList);
    }
    // פעולה שנקראת לאחר לחיצה על כפתור ניקוי סרגל החיפוש
    private void BtnClear_Click(object sender, EventArgs e)
    {
        // מחיקת טקסט המופיע בסרגל החיפוש
        actvSearch.Text = "";
        // מצביעה על רשימה עם כל המתכונים currentRecipes
        currentRecipes = recipeList.ToList();
        // עדכון אדפטר המתכונים ועדכון המתכונים בדף
        recipeAdapter.SetRecipes(currentRecipes);
        recipeAdapter.NotifyDataSetChanged();
    }
    // פעולה שנקראת לאחר לחיצה על כפתור חיפוש מתכונים
    private void BtnSearch_Click(object sender, EventArgs e)
    {
        // foreeach variable in list it selects the right variables and puts them in a list
        // מצביעה על רשימה כך ששמות המתכונים מתחילים בטקסט שהוקלד, אין הפרדה לאותיות קטנות וגדולות בחיפוש currentRecipes
        currentRecipes = (from recipe in recipeList where recipe.recipeName.StartsWith(actvSearch.Text,
        StringComparison.OrdinalIgnoreCase) select recipe).ToList();

        // עדכון אדפטר המתכונים ועדכון המתכונים בדף
        recipeAdapter.SetRecipes(currentRecipes);
```

```
            recipeAdapter.NotifyDataSetChanged();
        }
        // לחיצה על כפתור התפריט
        private void BtnMenu_Click(object sender, EventArgs e)
        {
            //כאשר הכפתור נלחץ הפעולה פותחת את התפריט
            OpenContextMenu((Button)sender);
        }
        /* AdapterView.IOnItemClickListener מימוש הפעולה מתוקף מימוש הממשק
         נלחץ מופעלת הפעולה ListView- כאשר מתכון/איבר ב */
        public void OnItemClick(AdapterView parent, View view, int position, long id)
        {
            // מעבר לדף המתכון כאשר מתכון כלשהו ב-ListView נלחץ ושליחת קוד המתכון כמחרוזת
            Intent intent = new Intent(this, typeof(RecipePageActivity));
            intent.PutExtra("RecipeId", recipeList[position].recipeId);
            StartActivity(intent);
        }
    }
```

FavoriteRecipesActivity

משויך ל-favoriteRecipes_layout. האקטיביטי של דף המתכונים האהובים מתבסס על הפעולות השונות במחלקת HomepageActivity לשם הצגת מתכונים, עדכון וסינון מתכונים ומעבר לדף מתכון אולם תחילה הוא שולף ממסד הנתונים ומציג רק את המתכונים האהובים של המשתמש באמצעות פעולות עזר שונות ממחלקת DatabaseManager כך שנעשה שימוש ושמירה רק של מתכונים אהובים.

```
// קבלת כל המתכונים האהובים ממחלקת עזר
recipeList = DatabaseManager.GetAllFavoriteRecipes();
```

MyRecipesActivity

משויך ל-myRecipes_layout. האקטיביטי של דף המתכונים שהמשתמש יצר גם כן מתבסס על הפעולות השונות במחלקת HomepageActivity לשם הצגת מתכונים, עדכון וסינון מתכונים ומעבר לדף מתכון אולם תחילה הוא שולף ממסד הנתונים ומציג רק את המתכונים של המשתמש באמצעות פעולות עזר שונות ממחלקת DatabaseManager כך שנעשה שימוש ושמירה רק של מתכונים שאותם המשתמש הכין.

```
// קבלת כל המתכונים שהמשתמש יצר ממחלקת עזר
recipeList = DatabaseManager.GetAllMyRecipes();
```

CreateRecipeActivity

משויך ל-createRecipe_layout. האקטיביטי של דף הכנת מתכון כולל קישורים לכל פקדי השדות בדף. בלחיצה על פקד תמונה להעלאת תמונת מתכון מציג AlertDilaog את האפשרויות להעלאת המתכון: דרך הגלריה או המצלמה. קוד הגלריה – 0, וקוד המצלמה – 1 כך שישנה פעולה המבוצעת כאשר המשתמש חוזר לדף אחרי שבחר תמונה או צילם אותה ועתה התמונה תופיע בפקד התמונה.
האקטיביטי כולל בדיקות תקינות של השדות שיש למלא. בפרט נעשה שימוש במחלקה הסטטית Regex לבדיקה של שם המתכון כך שהוא יכיל אותיות באנגלית בלבד. ישנם שתי פעולות אחת המקבלת את הקטגוריות מCategoryDialogFragment ואחת המקבלת את סוג המתכון ופרטי סוג המתכון מ-SaladDialogFragment , SoupDialogFragment, MeatDialogFragment או PastryDialogFragment בהתאם לסוג המתכון הנבחר ב-RecipeTypeDialogFragmen.
לאחר לחיצה על כפתור העלאת המתכון לאפליקציה הפעולה בודקת האם המתכון לא הועלה כבר קודם ע"י המשתמש בעזרת מחלקת עזר המקבלת את קוד המתכון המורכב לפי סוג המתכון, שם המתכון ושם המשתמש שיצר את המתכון - recipeType:recipeName:recipeUsername.
לאחר מכן ממירה את Bitmap התמונה למחרוזת ויוצרת את המתכון. הערך המוכנס למסד הנתונים הוא לא מטיפוס מתכון אלא מטיפוס: סלט, מרק, בשר או מאפה בהתאם לסוג המתכון הנבחר והוא מוכנס לטבלה המתאימה במסד הנתונים. לאחר מכן קוד המתכון מתווסף לתכונת מחרוזת קודי המתכונים שהמשתמש יצר ומעדכנת זאת במסד הנתונים. במידה והמשתמש מאפשר את שירות ההודעות, הסרוויס יראה הודעה עם התוכן המתאים וישנו מעבר לדף הבית.

```
[Activity(Label = "CreateRecipeActivity")]
    public class CreateRecipeActivity : Activity
    {
        // פקד ImageView של תמונת המתכון
        private ImageView ivImage;
        // משתנה שמציין האם הוכנסה תמונת מתכון
        private bool validImage;
        // פקד EditText של שם המתכון
        private EditText etRecipeName;
        // פקד EditText של מספר שעות הכנת המתכון
        private EditText etHours;
        // פקד EditText של מספר דקות הכנת המתכון
        private EditText etMinutes;
        // פקד TextView של סוג המתכון
        private TextView tvRecipeType;
        // פקד TextView של קטגוריית המתכון
        private TextView tvCategory;
        // פקד EditText של תיאור המתכון
        private EditText etDescription;
        // פקד EditText של מרכיבי המתכון
        private EditText etIngredients;
        // פקד EditText של הוראות הכנת המתכון
        private EditText etInstructions;
        // פקד Button כפתור סיום ושמירת המתכון במערכת
        private Button btnCreateRecipe;
        // פקד Button כפתור חזרה לדף הבית
        private Button btnGoBack;
        // משתנה השומר את ההודעה המתאימה במקרה של שגיאה
        private string errorText;
        // LinearLayout שכולל בתוכו את הכותרות והמידע של סוג המתכון
        private LinearLayout layoutRecipeType;
        // מערך TextView של כותרות המידע של סוג המתכון
        private TextView[] tvTypeTitles;
        // מערך TextView של המידע של סוג המתכון
        private TextView[] tvTypeDetails;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.createRecipe_layout);
            // false תחילה המשתנה יהיה מאחר ולא הוכנה תמונת מתכון
            validImage = false;
            ivImage = FindViewById<ImageView>(Resource.Id.createRecipe_ivAddRecipeImage);
            etRecipeName = FindViewById<EditText>(Resource.Id.createRecipe_etRecipeName);
            tvCategory = FindViewById<TextView>(Resource.Id.createRecipe_tvCategory);
            etHours = FindViewById<EditText>(Resource.Id.createRecipe_etHours);
            etMinutes = FindViewById<EditText>(Resource.Id.createRecipe_etMinutes);
            tvRecipeType = FindViewById<TextView>(Resource.Id.createRecipe_tvRecipeType);
            etDescription = FindViewById<EditText>(Resource.Id.createRecipe_etDescription);
            etIngredients = FindViewById<EditText>(Resource.Id.createRecipe_etIngredients);
            etInstructions = FindViewById<EditText>(Resource.Id.createRecipe_etInstructions);
            btnCreateRecipe = FindViewById<Button>(Resource.Id.createRecipe_btnCreateRecipe);
            btnGoBack = FindViewById<Button>(Resource.Id.createRecipe_btnGoBack);
            layoutRecipeType =FindViewById<LinearLayout>(Resource.Id.createRecipe_layoutRecipeType);
            // לכל סוג מתכון יש 4 כותרות ו4 מקומות למידע הרלוונטי לסוג המתכון
            tvTypeTitles = new TextView[4];
            tvTypeTitles[0] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeTitle1);
            tvTypeTitles[1] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeTitle2);
            tvTypeTitles[2] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeTitle3);
            tvTypeTitles[3] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeTitle4);
            tvTypeDetails = new TextView[4];
            tvTypeDetails[0] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeDetail1);
            tvTypeDetails[1] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeDetail2);
            tvTypeDetails[2] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeDetail3);
            tvTypeDetails[3] = FindViewById<TextView>(Resource.Id.createRecipe_tvTypeDetail4);
            ivImage.Click += IvImage_Click;
            tvCategory.Click += TvCategory_Click;
            tvRecipeType.Click += TvRecipeType_Click;
            btnCreateRecipe.Click += BtnCreateRecipe_Click;
            btnGoBack.Click += delegate { Finish(); };
        }
```

```
// פעולה שמתרחשת בעת לחיצה על תמונת המתכון ובאפשרותה לבחור תמונה למתכון
private void IvImage_Click(object sender, EventArgs e)
{
    // בניית AlertDialog שבו ניתן לבחור את האמצעי להעלת תמונת המתכון, מצלמה או גלריית התמונות
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
    alertDialog.SetTitle("Choose a recipe image");
    alertDialog.SetMessage("Choose between Gallery and Camera");
    alertDialog.SetCancelable(true);
    alertDialog.SetPositiveButton("GALLERY", delegate
    {
        // Intent שמפנה לגלריית התמונות, ישנה בחירה בין Gallery ו-Photos
        Intent galleryIntent = new Intent(Intent.ActionPick,
        MediaStore.Images.Media.ExternalContentUri);
        // הפעלת הפעולה הנ"ל אשר שולחת קוד 0 אשר מציין שהחזרה לאפליקציה היא מהגלרייה
        StartActivityForResult(Intent.CreateChooser(galleryIntent, "Select an Image"), 0);
    });
    alertDialog.SetNegativeButton("CAMERA", delegate
    {
        // Intent שמפנה למצלמת הטלפון
        Intent cameraIntent = new Intent(MediaStore.ActionImageCapture);
        // הפעלת הפעולה הנ"ל אשר שולחת קוד 1 אשר מציין שהחזרה לאפליקציה היא מהמצלמה
        StartActivityForResult(cameraIntent, 1);
    });
    alertDialog.SetNeutralButton("CANCEL", delegate
    {
        alertDialog.Dispose();
    });
    alertDialog.Show();
}
// פעולה שמתבצעת בעת החזרה לאפליקציה מהגלרייה או מהמצלמה
protected override void OnActivityResult(int requestCode, Result resultCode, Intent data)
{
    base.OnActivityResult(requestCode, resultCode, data);
    // עצם מטיפוס Bitmap אשר בתוכו נשמר תוכן התמונה שנבחרה
    Bitmap imageBitmap = null;
    // בדיקה האם החזרה בוצעה בהצלחה
    if (resultCode == Result.Ok)
    {
        // Choose from gallery
        if (requestCode == 0)
        {
            // שומר את תוכן התמונה מהגלרייה
            imageBitmap = MediaStore.Images.Media.GetBitmap(ContentResolver, data.Data);
        }
        // Take a photo
        else if (requestCode == 1)
        {
            // Get the image bitmap from the intent extras
            // שומר את תוכן התמונה מהמצלמה
            imageBitmap = (Bitmap)data.Extras.Get("data");
        }
        // שינוי תמונת המתכון באמצעות Bitmap
        ivImage.SetImageBitmap(imageBitmap);
        // כאשר המשתמש שם תמונה משלו המשתנה יהיה true
        validImage = true;
    }
}

// פעולה הפותחת את CategoryDialogFragment כאשר פקד tvCategory נלחץ
private void TvCategory_Click(object sender, EventArgs e)
{
    // Pull up the categoryFragmentLayout (dialog)
    // יצירת אובייקט מטיפוס CategoryDialogFragment
       ואובייקט מטיפוס FragmentTransaction בכדי לאפשר את פתיחתו
    FragmentTransaction transaction = FragmentManager.BeginTransaction();
    CategoryDialogFragment categoryDialogFragment = new CategoryDialogFragment();
    categoryDialogFragment.Show(transaction, "Category dialog fragment");
}
/* פעולת עזר שמופעלת ב-CategoryDialogFragment לאחר סיום בחירת הקטגוריות
 הפעולה מקבלת מחרוזת המכילה את הקטגוריות המופרדות בפסיקים ומציגה את המחרוזת בפקד tvCategory */
public void ReceiveCategories(string categories)
{
    tvCategory.Text = categories;
}
// פעולה הפותחת את RecipeTypeDialogFragment כאשר פקד tvRecipeType נלחץ
private void TvRecipeType_Click(object sender, EventArgs e)
{
    // Pull up the recipeTypeDialogFragment (dialog)
    // יצירת אובייקט מטיפוס RecipeTypeDialogFragment
```

```
            ואובייקט מטיפוס FragmentTransaction בכדי לאפשר את פתיחתו
                FragmentTransaction transaction = FragmentManager.BeginTransaction();
                RecipeTypeDialogFragment recipeTypeDialogFragment = new RecipeTypeDialogFragment();
                recipeTypeDialogFragment.Show(transaction, "Recipe Type dialog fragment");
            }

            /* פעולת עזר שמופעלת ב-
            PastryDialogFragment או MeatDialogFragment, SoupDialogFragment ,SaladDialogFragment
            לאחר סיום בחירת פרטי סוג המתכון, הפעולה מקבלת מחרוזת המכילה את סוג המתכון
            מערך מחרוזות המכיל את כותרות המידע ומערך מחרוזות המכיל פרטי המידע של סוג המתכון, */
            public void ReceiveRecipeTypeDetails(string recipeType, string[] titles, string[] details)
            {
                // הצבת סוג המתכון בפקד
                tvRecipeType.Text = recipeType;
                //פרטי סוג המתכון לאחר בחירת הפרטים מופיעים
                layoutRecipeType.Visibility = ViewStates.Visible;
                //הצבת הכותרות והפרטים
                tvTypeTitles[0].Text = titles[0];
                tvTypeTitles[1].Text = titles[1];
                tvTypeTitles[2].Text = titles[2];
                tvTypeTitles[3].Text = titles[3];
                tvTypeDetails[0].Text = details[0];
                tvTypeDetails[1].Text = details[1];
                tvTypeDetails[2].Text = details[2];
                tvTypeDetails[3].Text = details[3];
            }
            // פעולה כללית הבודקת האם הפרטים שיש למלא תקינים
            private bool IsValid()
            {
                if (!IsValidRecipeImage())
                {
                    return false;
                }
                if (!IsValidRecipeName())
                {
                    return false;
                }
                if (!IsValidCategory())
                {
                    return false;
                }
                if (!IsValidHours())
                {
                    return false;
                }
                if (!IsValidMinutes())
                {
                    return false;
                }
                if (!IsValidRecipeType())
                {
                    return false;
                }
                if (!IsValidDescription())
                {
                    return false;
                }
                if (!IsValidIngredients())
                {
                    return false;
                }
                if (!IsValidInstructions())
                {
                    return false;
                }
                return true;
        }

            // בדיקות התקינות הבאות מעדכנות את המשתנה errorText לפי השגיאה
            // בדיקת תקינות של תמונת המתכון שנבדק לפי המשתנה validImage
            private bool IsValidRecipeImage()
            {
                // Recipe Image Validation
                if (!validImage)
                {
                    errorText = "Choose recipe image";
                    return false;
                }
                return true;
```

```
        }
        // בדיקת תקינות של שם המתכון
        private bool IsValidRecipeName()
        {
            // Recipe Name Validation
            // בדיקה שהשם לא ריק

            if (etRecipeName.Text == "")
            {
                errorText = "Insert recipe name";
                return false;
            }
            // IsMatch בעלת פעולה Regex בדיקת תקינות באמצעות מחלקה סטטית
            // בדיקת תקינות שהמחרוזת של שם המתכון מתחילתו ועד סופו מכיל רק אותיות באנגלית
            if (!Regex.IsMatch(etRecipeName.Text, "^[a-zA-Z ]+$"))
            {
                errorText = "Recipe name must contain English letters only";
                return false;
            }
            return true;
        }
        // בדיקת תקינות הקטגוריות
        private bool IsValidCategory()
        {
            // Category Validation
            // בדיקה שנבחרו קטגוריות למתכון
            if (tvCategory.Text == "Category(ies)")
            {
                errorText = "Must choose at least one category";
                return false;
            }
            return true;
        }
        // בדיקת תקינות שעות ההכנה
        private bool IsValidHours()
        {
            //Hours Validation
            if (etHours.Text == "")
            {
                errorText = "Insert hours";
                return false;
            }
            return true;
        }
        // בדיקת תקינות דקות ההכנה
        private bool IsValidMinutes()
        {
            //Minutes Validation
            if (etMinutes.Text == "")
            {
                errorText = "Insert minutes";
                return false;
            }
            // בדיקה שהדקות בין 0 ל59
            if (int.Parse(etMinutes.Text) > 59)
            {
                errorText = "Minutes' range must be between 0 - 59 minutes";
                return false;
            }
            return true;
        }
        // בדיקת תקינות שנבחר סוג מתכון
        private bool IsValidRecipeType()
        {
            //Recipe Type Validation
            if (tvRecipeType.Text == "Recipe Type")
            {
                errorText = "Must choose a recipe type";
                return false;
            }
            return true;
        }
        // בדיקת תקינות תיאור המתכון
        private bool IsValidDescription()
        {
            if (etDescription.Text == "")
            {
                errorText = "Insert Description";
                return false;
```

```
        }
        return true;
    }
    // בדיקת תקינות מרכיבי המתכון
    private bool IsValidIngredients()
    {
        if (etDescription.Text == "")

        {
            errorText = "Insert Ingredients";
            return false;
        }
        return true;
    }
    // בדיקת תקינות הוראות המתכון
    private bool IsValidInstructions()
    {
        if (etDescription.Text == "")
        {
            errorText = "Insert Instructions";
            return false;
        }
        return true;
    }
    // פעולה הנקראת לאחר לחיצה על כפתור יצירת מתכון
    private void BtnCreateRecipe_Click(object sender, EventArgs e)
    {
        // בדיקה האם הפרטים תקינים
        if (IsValid())
        {
            // קבלת שם המשתמש המחובר באמצעות מחלקת העזר SharedPreferencesManager
            string recipeUsername = SharedPreferencesManager.GetUsername();

            // יצירת קוד מתכון אשר מורכב מסוג המתכון, שם המתכון ושם המשתמש, אשר מופרדים בנקודותיים
            string recipeId = tvRecipeType.Text + ":" + etRecipeName.Text + ":" +
            recipeUsername;
            // בדיקה באמצעות מחלקת העזר DatabaseManager האם המתכון קיים לפי קוד המתכון הנ"ל
            if (!DatabaseManager.IsExistRecipe(recipeId))
            {
                // יצירת אובייקט Bitmap שמצביע על ביטמפ של פקד תמונת המתכון
                Bitmap imageBitmap = ((BitmapDrawable)ivImage.Drawable).Bitmap;
                // מחרוזת המכילה את תוכן התמונה באמצעות שימוש במחלקת עזר ImageManager שממירה ביטמפ למחרוזת
                string recipeImage = ImageManager.BitmapToBase64(imageBitmap);
                // בדיקה איזה סוג מתכון נבחר
                if (tvRecipeType.Text == "Salad")
                {
                    // יצירת אובייקט סלט והצבתו במסד הנתונים באמצעות מחלקת העזר DatabaseManager
                    Salad salad = new Salad(tvTypeDetails[0].Text, tvTypeDetails[1].Text,
                    tvTypeDetails[2].Text, tvTypeDetails[3].Text, recipeId, etRecipeName.Text,
                    recipeUsername, recipeImage, tvCategory.Text, int.Parse(etHours.Text),
                    int.Parse(etMinutes.Text), etDescription.Text, etIngredients.Text,
                    etInstructions.Text);
                    DatabaseManager.AddRecipe(salad);
                }
                if (tvRecipeType.Text == "Soup")
                {
                    // יצירת אובייקט מרק והצבתו במסד הנתונים באמצעות מחלקת העזר DatabaseManager
                    Soup soup = new Soup(tvTypeDetails[0].Text, tvTypeDetails[1].Text,
                    int.Parse(tvTypeDetails[2].Text), int.Parse(tvTypeDetails[3].Text),
                    recipeId, etRecipeName.Text, recipeUsername, recipeImage, tvCategory.Text,
                    int.Parse(etHours.Text), int.Parse(etMinutes.Text),
                    etDescription.Text, etIngredients.Text, etInstructions.Text);
                    DatabaseManager.AddRecipe(soup);
                }
                if (tvRecipeType.Text == "Meat")
                {
                    // יצירת אובייקט מנת בשר והצבתו במסד הנתונים באמצעות מחלקת העזר DatabaseManager
                    Meat meat = new Meat(tvTypeDetails[0].Text, tvTypeDetails[1].Text,
                    tvTypeDetails[2].Text, int.Parse(tvTypeDetails[3].Text), recipeId,
                    etRecipeName.Text, recipeUsername, recipeImage, tvCategory.Text,
                    int.Parse(etHours.Text), int.Parse(etMinutes.Text), etDescription.Text,
                    etIngredients.Text, etInstructions.Text);
                    DatabaseManager.AddRecipe(meat);
                }
                if (tvRecipeType.Text == "Pastry")
                {
                    // יצירת אובייקט מאפה והצבתו במסד הנתונים באמצעות מחלקת העזר DatabaseManager
                    Pastry pastry = new Pastry(tvTypeDetails[0].Text, tvTypeDetails[1].Text,
                    int.Parse(tvTypeDetails[2].Text), int.Parse(tvTypeDetails[3].Text),
```

```
                    recipeId, etRecipeName.Text, recipeUsername, recipeImage, tvCategory.Text,
                    int.Parse(etHours.Text), int.Parse(etMinutes.Text), etDescription.Text,
                    etIngredients.Text, etInstructions.Text);
                    DatabaseManager.AddRecipe(pastry);
                }
                // עדכון קודי המתכונים של המשתמש במסד הנתונים
                User currentUser = DatabaseManager.GetUser(recipeUsername);
                // קוד המתכון הנ"ל מתווסף למחרוזת קודי המתכונים של המשתמש המופרדים בפסיקים
                currentUser.myRecipesId += recipeId + ",";
                DatabaseManager.SetUser(currentUser);

                // StatusNotificationService
                // בדיקה שהמשתמש מרשה הודעות
                if (currentUser.allowNotification)
                {
                    // הפעלת שירות הודעות לאחר יצירת מתכון
                    StartNotificationService();
                }

                // סיום וסגירת האקטיביטי וחזרה לדף הבית
                Finish();
            }
            else
            {
                // אם המתכון קיים במסד הנתונים תוצג הודעה מתאימה
                errorText = "Recipe already exists";
                ShowAlertDialog();
            }
        }
        else
        {
            // במידה ולא, תופיע הודעת שגיאה מתאימה
            ShowAlertDialog();
        }
    }
```

RecipePageActivity

משויך ל-recipePage_layout. האקטיביטי של דף המתכון כולל קישורים לכל פקדי השדות בדף. ישנו משתנה למשתמש יוצר המתכון והמשתמש המחובר למערכת. ישנם פקדי EditText אשר תחילה תכונות העריכה שלהם מבוטלות בעזרת שימוש בממשק IKeyListener אשר שומר את תכונות הפקד ובעת העדכון מחזיר את התכונה הזו לפקדים. בעת לחיצה על כפתור אייקון הלב, הוא משנה את צבעו מלבן לאדום ולהפך בלחיצה שנייה, מעדכן את מספר האנשים שאוהבים את המתכון ושומר את השינוי של המתכון במסד הנתונים. כמו כן, במידה והמשתמש מאפשר את שירות ההודעות (סרוויס) תופיע הודעה מתאימה ברגע הלחיצה על כך שהמתכון נמחק/התווסף לרשימת המתכונים האהובים של המשתמש המחובר. ישנה בדיקה האם המשתמש המחובר הוא גם המשתמש שיצר את המתכון או מנהל דבר המאפשר עדכון ומחיקת מתכון. בעת מצב עדכון מתכון פקדי השדות השונים משנים את סגנונם וניתן ללחוץ עליהם או לערוך אותם. מצב זה דומה לדף הכנת המתכון הכולל בדיקות תקינות וקישורים לטפסי מילוי. לאחר אישור העדכון, פרטי המתכון נשמרים במסד הנתונים והדף מתעדכן ונפתח מחדש. בעת מחיקת מתכון, קוד המתכון נמחק מתכונת מחרוזת קודי המתכונים שהמשתמש יצר והפעולה גם עוברת על כל המשתמשים באפליקציה ובודקת האם הם אהבו את המתכון הנ"ל, במידה וכן קוד המתכון ימחק מתכונת מחרוזת קודי המתכונים האהובים של המשתמשים; לבסוף המתכון ימחק ממסד הנתונים. בסוף עדכון ומחיקת מתכון במידה והמשתמש מאפשר את שירות ההודעות, הסרוויס יופעל ויציג הודעה מתאימה בהתאם לנעשה.

במידה ומנהל מוחק מתכון של משתמש הוא רשאי לשלוח לו הודעה דרך האימייל ובסוף פעולת המחיקה יפתח דף מילוי לשליחת הודעה לאימייל המשתמש.

```
[Activity(Label = "RecipePageActivity")]
public class RecipePageActivity : Activity
{
    // משתמש המייצג את המשתמש המחובר למערכת
    private User loggedUser;
    // משתמש המייצג את המשתמש שיצר את המתכון
    private User recipeUser;
    // המתכון המוצג בדף
    private Recipe recipe;
    // פקד TextView של שם המתכון - כותרת הדף
    private TextView tvRecipeName;
    // פקד תמונה של תמונת המתכון - ניתן לעדכן
    private ImageView ivRecipeImage;
    // פקד TextView של שם המשתמש שיצר את המתכון
    private TextView tvRecipeUsername;
    // פקד תמונה עגולה של תמונת פרופיל המשתמש של המתכון
    private CircleImageView civUserImage;
    // פקד TextView של קטגוריות המתכון - ניתן לעדכן
    private TextView tvCategory;
    // פקד EditText של דקות ההכנה - ניתן לעדכן
    private EditText etHours;
    // פקד EditText של דקות ההכנה - ניתן לעדכן
    private EditText etMinutes;
    // פקד TextView של סוג המתכון
    private TextView tvRecipeType;
    // מערך פקדי TextView של כותרות סוג המתכון
    private TextView[] tvTypeTitles;
    // מערך פקדי TextView של מידע סוג המתכון - ניתן לעדכן
    private TextView[] tvTypeDetails;
    // פקד EditText של תיאור המתכון - ניתן לעדכן
    private EditText etDescription;
    // פקד EditText של המרכיבים - ניתן לעדכן
    private EditText etIngredients;
    // פקד EditText של הוראות ההכנה - ניתן לעדכן
    private EditText etInstructions;
    // פקד TextView של כמות הלייקים
    private TextView tvLikeCount;
    // פקד עם תמונת לב המסמל אם המשתמש המחובר אוהב את המתכון
    private ImageView ivLike;
    // כפתור חזרה לדף הבית
    private Button btnGoBack;
    // כפתור מצב עדכון מתכון
    private Button btnEdit;
    // כפתור מחיקת מתכון
    private Button btnDelete;
    // כפתור שמירה לאחר עדכון מתכון
    private Button btnSave;
    // מחרוזת עם המידע המתאים ברגע אי תקינות נתונים בעת עדכון מתכון
    private string errorText;
    /* Interface for converting text key events into edit operations on an Editable class
     שימוש במערך של ממשק לביטול תכונת עריכת טקסט ב-EditText כאשר הדף מוצג
     והחזרת תכונות העריכה של EditText כאשר הדף נמצא במצב עדכון מתכון */
    IKeyListener[] keyListeners;
    protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);
        SetContentView(Resource.Layout.recipePage_layout);
        tvRecipeName = FindViewById<TextView>(Resource.Id.recipePage_tvRecipeName);
        ivRecipeImage = FindViewById<ImageView>(Resource.Id.recipePage_ivRecipeImage);
        tvRecipeUsername = FindViewById<TextView>(Resource.Id.recipePage_tvRecipeUsername);
        civUserImage = FindViewById<CircleImageView>(Resource.Id.recipePage_civUserImage);
        tvCategory = FindViewById<TextView>(Resource.Id.recipePage_tvCategory);
        etHours = FindViewById<EditText>(Resource.Id.recipePage_etHours);
        etMinutes = FindViewById<EditText>(Resource.Id.recipePage_etMinutes);
        tvRecipeType = FindViewById<TextView>(Resource.Id.recipePage_tvRecipeType);
        etDescription = FindViewById<EditText>(Resource.Id.recipePage_etDescription);
        etIngredients = FindViewById<EditText>(Resource.Id.recipePage_etIngredients);
        etInstructions = FindViewById<EditText>(Resource.Id.recipePage_etInstructions);
        tvLikeCount = FindViewById<TextView>(Resource.Id.recipePage_tvLikeCount);
        ivLike = FindViewById<ImageView>(Resource.Id.recipePage_ivLike);
        btnEdit = FindViewById<Button>(Resource.Id.recipePage_btnEdit);
        btnDelete = FindViewById<Button>(Resource.Id.recipePage_btnDelete);
        btnSave = FindViewById<Button>(Resource.Id.recipePage_btnSave);
        btnGoBack = FindViewById<Button>(Resource.Id.recipePage_btnGoBack);
        // לכל סוג מתכון (סלט, מרק, מנת בשר, מאפה) ארבעה כותרות
        tvTypeTitles = new TextView[4];
        tvTypeTitles[0] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeTitle1);
        tvTypeTitles[1] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeTitle2);
        tvTypeTitles[2] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeTitle3);
        tvTypeTitles[3] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeTitle4);
        // לכל סוג מתכון ארבעה שדות מידע המייצגים את נתוני תכונותיהם
        tvTypeDetails = new TextView[4];
        tvTypeDetails[0] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeDetail1);
        tvTypeDetails[1] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeDetail2);
```

```
            tvTypeDetails[2] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeDetail3);
            tvTypeDetails[3] = FindViewById<TextView>(Resource.Id.recipePage_tvTypeDetail4);
            // קבלת המשתמש המחובר בעזרת שתי מחלקות עזר
            // אחת מקבלת את שם המשתמש בקובץ השמור בזכרון והשנייה מזהה משתמש מהמסד לפי שם המשתמש
            loggedUser = DatabaseManager.GetUser(SharedPreferencesManager.GetUsername());
            // קבלת קוד המתכון מדף האקטיביטי הקודם וזיהוי המתכון בעזרת מחלקת עזר
            recipe = DatabaseManager.GetRecipe(Intent.GetStringExtra("RecipeId"));
            // קבלת משתמש המתכון לפי תכונת המתכון
            recipeUser = DatabaseManager.GetUser(recipe.recipeUsername);
            // הצבת פרטי המידע השונים בשדות המתאימים לפי תכונות המתכון
            tvRecipeName.Text = recipe.recipeName; // שם המתכון
            tvRecipeUsername.Text = recipe.recipeUsername; // שם משתמש המתכון
           ivRecipeImage.SetImageBitmap(ImageManager.Base64ToBitmap(recipe.recipeImage));//תמונת המתכון
            //בדיקה שאכן למשתמש יש תמונת פרופיל קיימת
            if (recipeUser.userImage != "")
            {
                // השמת תמונת פרופיל המשתמש של המתכון
                civUserImage.SetImageBitmap(ImageManager.Base64ToBitmap(recipeUser.userImage));
            }
            tvCategory.Text = recipe.category; // קטגוריות
            etHours.Text = recipe.hours.ToString(); // שעות הכנה
            etMinutes.Text = recipe.minutes.ToString(); // דקות הכנה
            etDescription.Text = recipe.description; // תיאור מתכון
            etIngredients.Text = recipe.ingredients; // מרכיבים
            etInstructions.Text = recipe.instructions; // הוראות הכנה
            // בדיקה איזה סוג מתכון המתכון הנ"ל והצבת כותרות ומידע מתאים
            if (recipe is Salad)
            {
                tvRecipeType.Text = "Salad";
                tvTypeTitles[0].Text = "Vegetables/Fruit";
                tvTypeTitles[1].Text = "Salad Type";
                tvTypeTitles[2].Text = "Contains Meat";
                tvTypeTitles[3].Text = "Contains Flavoring";
                tvTypeDetails[0].Text = ((Salad)recipe).isFruit;
                tvTypeDetails[1].Text = ((Salad)recipe).saladType;
                tvTypeDetails[2].Text = ((Salad)recipe).isMeat;
                tvTypeDetails[3].Text = ((Salad)recipe).isFlavoring;
            }
            if (recipe is Soup)
            {
                tvRecipeType.Text = "Soup";
                tvTypeTitles[0].Text = "Soup Type";
                tvTypeTitles[1].Text = "Cooking Duration";
                tvTypeTitles[2].Text = "Boiling Time";
                tvTypeTitles[3].Text = "Temperature";
                tvTypeDetails[0].Text = ((Soup)recipe).soupType;
                tvTypeDetails[1].Text = ((Soup)recipe).duration;
                tvTypeDetails[2].Text = ((Soup)recipe).boilingTime.ToString();
                tvTypeDetails[3].Text = ((Soup)recipe).boilingTemperature.ToString();
            }
            if (recipe is Meat)
            {
                tvRecipeType.Text = "Meat";
                tvTypeTitles[0].Text = "Meat Type";
                tvTypeTitles[1].Text = "Cooking Method";
                tvTypeTitles[2].Text = "Meat Doneness";
                tvTypeTitles[3].Text = "Cooking Time";
                tvTypeDetails[0].Text = ((Meat)recipe).meatType;
                tvTypeDetails[1].Text = ((Meat)recipe).method;
                tvTypeDetails[2].Text = ((Meat)recipe).doneness;
                tvTypeDetails[3].Text = ((Meat)recipe).cookingTime.ToString();
            }
            if (recipe is Pastry)
            {
                tvRecipeType.Text = "Pastry";
                tvTypeTitles[0].Text = "Contains Gluten";
                tvTypeTitles[1].Text = "Kosher";
                tvTypeTitles[2].Text = "Baking Time";
                tvTypeTitles[3].Text = "Temperature";
                tvTypeDetails[0].Text = ((Pastry)recipe).isGluten;
                tvTypeDetails[1].Text = ((Pastry)recipe).kosher;
                tvTypeDetails[2].Text = ((Pastry)recipe).bakingTime.ToString();
                tvTypeDetails[3].Text = ((Pastry)recipe).bakingTemperature.ToString();
        }

            // אם המשתמש אוהב את המתכון צבע הלב יהיה אדום אחרת יהיה לבן
            if (loggedUser.favoriteRecipesId.Contains(recipe.recipeId))
            {
                ivLike.SetColorFilter(Color.ParseColor("#ffd5284a")); // צבע אדום
```

```
            }
            else
            {
                ivLike.SetColorFilter(Color.ParseColor("#ffeeeeee")); // צבע לבן

            }
            // כתיבת טקסט מתאים עבור מספר הלייקים
            if (recipe.likeCount == 1)
            {
                // הודעה עבור לייק אחד
                tvLikeCount.Text = recipe.likeCount.ToString() + " person liked this recipe";
            }
            else
            {
                // הודעה על מספר לייקים שונה
                tvLikeCount.Text = recipe.likeCount.ToString() + " people liked this recipe";
            }

            // אם המשתמש המחובר הוא יוצר המתכון או מנהל האפליקציה אזי תהיה לו גישה לכפתורי עדכון ומחיקת המתכון
            if (recipeUser.username == loggedUser.username || loggedUser.isAdmin)
            {
                btnEdit.Visibility = ViewStates.Visible;
                btnDelete.Visibility = ViewStates.Visible;
            }
            ivLike.Click += IvLike_Click;
            ivRecipeImage.Click += IvRecipeImage_Click;
            tvCategory.Click += TvCategory_Click;
            tvRecipeType.Click += TvRecipeType_Click;
            btnEdit.Click += BtnEdit_Click;
            btnSave.Click += BtnSave_Click;
            btnDelete.Click += BtnDelete_Click;
            btnGoBack.Click += delegate { Finish(); };
            // EditText מערך keyListeners השומר את תכונות העריכה של
            keyListeners = new IKeyListener[5];
            keyListeners[0] = etHours.KeyListener;
            keyListeners[1] = etMinutes.KeyListener;
            keyListeners[2] = etDescription.KeyListener;
            keyListeners[3] = etIngredients.KeyListener;
            keyListeners[4] = etInstructions.KeyListener;
            // EditText ביטול תכונות העריכה של שדות
            etHours.KeyListener = null;
            etMinutes.KeyListener = null;
            etDescription.KeyListener = null;
            etIngredients.KeyListener = null;
            etInstructions.KeyListener = null;
            // פקדים המשמשים בעדכון כפקד לחיצה, במצב רגיל לא ניתן ללחוץ עליהם
            ivRecipeImage.Clickable = false;
            tvCategory.Clickable = false;
            tvRecipeType.Clickable = false;
        }
        // פעולה שנקראת כאשר לוחצים על תמונת אייקון הלב
        private void IvLike_Click(object sender, EventArgs e)
        {
            // מערך מחרוזות המכיל את תכני ההודעה שתשלח בהתאמה
            string[] recipePageDetails;

            // בדיקה האם המשתמש המחובר אוהב את המתכון
            if (loggedUser.favoriteRecipesId.Contains(recipe.recipeId))
            {
                // בלחיצה על הלב מספר הלייקים של המתכון יורד באחד
                recipe.likeCount--;
                // קוד המתכון מוסר ממחרוזת קודי המתכונים של המשתמש המחובר
                loggedUser.favoriteRecipesId = loggedUser.favoriteRecipesId.Replace(recipe.recipeId
                + ",", "");
                // צבע הלב משתנה ללבן
                ivLike.SetColorFilter(Color.ParseColor("#ffeeeeee"));
                // מחרוזת המציינת את כותרת ותוכן ההודעה בהתאמה
                recipePageDetails = new string[] { "A Recipe Was Deleted From Favorite Recipes",
                recipe.recipeName + " was successfully deleted" };
            }
            else
            {
                // בלחיצה על הלב מספר הלייקים של המתכון עולה באחד
                recipe.likeCount++;
                // קוד המתכון מתווסף למחרוזת קודי המתכונים של המשתמש המחובר
                loggedUser.favoriteRecipesId += recipe.recipeId + ",";
                // צבע הלב משתנה לאדום
                ivLike.SetColorFilter(Color.ParseColor("#ffd5284a"));
                // מחרוזת המציינת את כותרת ותוכן ההודעה בהתאמה
```

```
            recipePageDetails = new string[] { "A New Favorite Recipe Was Added",
            recipe.recipeName + " was entered successfully" };
        }
        // בדיקה שהמשתמש מרשה שירות הודעות
        if (loggedUser.allowNotification)
        {
            // הפעלת שירות הודעות לאחר לחיצה על הלב בהתאמה
            StartNotificationService(recipePageDetails);
        }
        // עדכון המתכון והמשתמש שאהב אותו במסד הנתונים
        DatabaseManager.SetRecipe(recipe);
        DatabaseManager.SetUser(loggedUser);
        // עדכון טקסט כמות הלייקים של המתכון
        if (recipe.likeCount == 1)
        {
            tvLikeCount.Text = recipe.likeCount.ToString() + " person liked this recipe";
        }
        else
        {
            tvLikeCount.Text = recipe.likeCount.ToString() + " people liked this recipe";
        }
    }

    // פעולה הנקראת כאשר לוחצים על כפתור העדכון
    private void BtnEdit_Click(object sender, EventArgs e)
    {
        // יצירת דיאלוג לאישור תחילת מצב העדכון
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
        alertDialog.SetTitle("Edit Recipe");
        alertDialog.SetMessage("Are you sure you want to edit: " + recipe.recipeName + "?");
        alertDialog.SetCancelable(true);
        // אישור התחלת עדכון מתכון
        alertDialog.SetPositiveButton("EDIT", delegate
        {
            // עיצוב הפקדים השונים כך שניתן לדעת שמדובר במצב עדכון מתכון
            ivRecipeImage.SetBackgroundColor(Color.ParseColor("#ffff6064"));
            tvCategory.SetBackgroundResource(Resource.Drawable.buttonStyle);
            tvRecipeType.SetBackgroundResource(Resource.Drawable.buttonStyle);
            etHours.SetBackgroundResource(Resource.Drawable.editTextStyle);
            etMinutes.SetBackgroundResource(Resource.Drawable.editTextStyle);
            etDescription.SetBackgroundResource(Resource.Drawable.editTextStyle);
            etIngredients.SetBackgroundResource(Resource.Drawable.editTextStyle);
            etInstructions.SetBackgroundResource(Resource.Drawable.editTextStyle);

            // מתן אפשרות ללחיצה על הפקדים השונים
            // לחיצה עליהם תפתח הודעה מתאימה
            ivRecipeImage.Clickable = true;
            tvCategory.Clickable = true;
            tvRecipeType.Clickable = true;

            // EditText החזרת תכונות עריכה לפקדי ה
            etHours.KeyListener = keyListeners[0];
            etMinutes.KeyListener = keyListeners[1];
            etDescription.KeyListener = keyListeners[2];
            etIngredients.KeyListener = keyListeners[3];
            etInstructions.KeyListener = keyListeners[4];

            // הסתרת כפתורי עדכון ומחיקת מתכון והצגת כפתור לשמירת העדכון
            btnEdit.Visibility = ViewStates.Gone;
            btnDelete.Visibility = ViewStates.Gone;
            btnSave.Visibility = ViewStates.Visible;

        });
        alertDialog.SetNeutralButton("CANCEL", delegate
        {
            alertDialog.Dispose();
        });
        alertDialog.Show();
    }
    // פעולה הנקראת כאשר לוחצים על כפתור השמירה לאחר העדכון
    private void BtnSave_Click(object sender, EventArgs e)
    {
        // יצירת דיאלוג לאישור העדכון
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
        alertDialog.SetTitle("Save Recipe");
        alertDialog.SetMessage("Are you sure you want to save: " + recipe.recipeName + "?");
        alertDialog.SetCancelable(true);
        // אישור עדכון מתכון
        alertDialog.SetPositiveButton("SAVE", delegate
```

```
                {
                    // בדיקה האם הנתונים המעודכנים תקינים
                    if (IsValid())
                    {
                        // עדכון אובייקט המתכון הנ"ל לפי השדות המעודכנים
                        Bitmap imageBitmap = ((BitmapDrawable)ivRecipeImage.Drawable).Bitmap;
                        recipe.recipeImage = ImageManager.BitmapToBase64(imageBitmap);
                        recipe.category = tvCategory.Text;
                        recipe.hours = int.Parse(etHours.Text);
                        recipe.minutes = int.Parse(etMinutes.Text);
                        recipe.description = etDescription.Text;
                        recipe.ingredients = etIngredients.Text;
                        recipe.instructions = etInstructions.Text;
                        // בדיקה איזה סוג מתכון, ועדכון פרטים לפי סוג המתכון בהתאמה
                        if (recipe is Salad)
                        {
                            ((Salad)recipe).isFruit = tvTypeDetails[0].Text;
                            ((Salad)recipe).saladType = tvTypeDetails[1].Text;
                            ((Salad)recipe).isMeat = tvTypeDetails[2].Text;
                            ((Salad)recipe).isFlavoring = tvTypeDetails[3].Text;
                        }
                        if (recipe is Soup)
                        {
                            ((Soup)recipe).soupType = tvTypeDetails[0].Text;
                            ((Soup)recipe).duration = tvTypeDetails[1].Text;
                            ((Soup)recipe).boilingTime = int.Parse(tvTypeDetails[2].Text);
                            ((Soup)recipe).boilingTemperature = int.Parse(tvTypeDetails[3].Text);
                        }
                        if (recipe is Meat)
                        {
                            ((Meat)recipe).meatType = tvTypeDetails[0].Text;
                            ((Meat)recipe).method = tvTypeDetails[1].Text;
                            ((Meat)recipe).doneness = tvTypeDetails[2].Text;
                            ((Meat)recipe).cookingTime = int.Parse(tvTypeDetails[3].Text);
                        }
                        if (recipe is Pastry)
                        {
                            ((Pastry)recipe).isGluten = tvTypeDetails[0].Text;
                            ((Pastry)recipe).kosher = tvTypeDetails[1].Text;
                            ((Pastry)recipe).bakingTime = int.Parse(tvTypeDetails[2].Text);
                            ((Pastry)recipe).bakingTemperature = int.Parse(tvTypeDetails[3].Text);
                        }
                        // עדכון המתכון במסד הנתונים
                        DatabaseManager.SetRecipe(recipe);
                        // StatusNotificationService
                        // בדיקה שיוצר המתכון מאשר את שירות ההודעות וגם המשתמש המחובר הוא גם יוצר המתכון - משתמש רגיל
                        // או האם המשתמש המחובר מאשר הודעות - כלומר מדובר במנהל שמעדכן מתכון של משתמש
                        if ((recipeUser.allowNotification && recipeUser.username == loggedUser.username)
                           || loggedUser.allowNotification)
                        {
                            // מחרוזת המציינת את כותרת ותוכן ההודעה לאחר עדכון מתכון
                            string[] recipePageDetails = { "A Recipe Was Edited", recipe.recipeName + "
                            was successfully edited" };
                            // הפעלת שירות הודעות לאחר עדכון מתכון
                            StartNotificationService(recipePageDetails);
                        }
                        // טעינה מחדש של הדף עם אנימציה רגילה של פתיחת דף, למניעת מסך שחור
                        // מופעלת לאחר StartActivity ופותחת את הדף מחדש מלמטה
                        OverridePendingTransition(0, 0);
                        // סגירת הדף ופתיחתו מחדש
                        Finish();
                        StartActivity(this.Intent);
                    }
                    else
                    {
                        // במידה ולא, תופיע הודעת שגיאה מתאימה
                        ShowAlertDialog();
                    }
                });
                alertDialog.SetNeutralButton("CANCEL", delegate
                {
                    alertDialog.Dispose();
                });
                alertDialog.Show();
            }
```

```
        // פעולה הנקראת כאשר לוחצים על כפתור מחיקת המתכון
        private void BtnDelete_Click(object sender, EventArgs e)
        {
            // יצירת דיאלוג לאישור המחיקה
            AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
            alertDialog.SetTitle("Delete Recipe");
            alertDialog.SetMessage("Are you sure you want to delete: " + recipe.recipeName + "?");
            alertDialog.SetCancelable(true);
            // מחיקת מתכון
            alertDialog.SetPositiveButton("DELETE", delegate
            {
                // מחיקת קוד המתכון במחרוזת קודי המתכונים של המשתמש שמתכונו נמחק ועדכון המשתמש
                recipeUser.myRecipesId = recipeUser.myRecipesId.Replace(recipe.recipeId + ",", "");
                DatabaseManager.SetUser(recipeUser);

                //מעבר על כל המשתמשים הרשומים ומחיקת קוד המתכון ברשימת קודי המתכונים האהובים שלהם במידה ואהבו את המתכון
                // רשימה המכילה את כל המשתמשים
                List<User> userList = DatabaseManager.GetAllUsers();
                // לולאה העוברת על כל המשתמשים
                for (int i = 0; i < userList.Count; i++)
                {
                    // בדיקה האם המשתמשים אוהבים את המתכון הנ"ל
                    if (userList[i].favoriteRecipesId.Contains(recipe.recipeId))
                    {
                        //אם כן מחיקת קוד המתכון הנ"ל במחרוזת המתכונים האהובים של כל משתמש ועדכון המשתמש במסד
                        userList[i].favoriteRecipesId =
                        userList[i].favoriteRecipesId.Replace(recipe.recipeId + ",", "");
                        DatabaseManager.SetUser(userList[i]);
                    }
                }
                // מחיקת המתכון הנ"ל ממסד הנתונים
                DatabaseManager.DeleteRecipe(recipe);
                // StatusNotificationService
                // בדיקה שיוצר המתכון מאשר את שירות ההודעות וגם המשתמש המחובר הוא גם יוצר המתכון - משתמש רגיל מחובר
                // או האם המשתמש המחובר מאשר הודעות - כלומר מדובר במנהל שמעדכן מתכון של משתמש
                if ((recipeUser.allowNotification && recipeUser.username == loggedUser.username) ||
                   loggedUser.allowNotification)
                {
                    // מחרוזת המציינת את כותרת ותוכן ההודעה לאחר מחיקת מתכון
                    string[] recipePageDetails = { "A Recipe Was Deleted", recipe.recipeName + " was
                    successfully deleted" };
                    // הפעלת שירות הודעות לאחר מחיקת מתכון
                    StartNotificationService(recipePageDetails);
                }
                // במידה והמוחק הוא מנהל יש לו אפשרות לשלוח אימייל למשתמש שמתכונו נמחק
                // בדיקה שהמשתמש המוחק הוא מנהל ושהוא לא ישלח אימייל לעצמו
                if (loggedUser.isAdmin && recipeUser.username != loggedUser.username)
                {
                    // במצב של מחיקת מתכון של משתמש המנהל רשאי לשלוח לו אימייל על כך
                    SendEmail();
                }
                Finish();
            });
            alertDialog.SetNeutralButton("CANCEL", delegate
            {
                alertDialog.Dispose();
            });
            alertDialog.Show();
        }

        // פעולה הפותחת דף לשליחת אימייל לאחר מחיקת מתכון בידי מנהל
        private void SendEmail()
        {
            // אימייל משתמש המתכון
            string[] email = { DatabaseManager.GetUser(tvRecipeUsername.Text).email };
            Intent sendEmailIntent = new Intent(Intent.ActionSend);
            sendEmailIntent.SetType("message/rfc822"); // Opening email clients
            sendEmailIntent.PutExtra(Intent.ExtraEmail, email); // נמען
            sendEmailIntent.PutExtra(Intent.ExtraSubject, string.Format("The recipe: {0} was deleted
            by an admin.", tvRecipeName.Text)); // כותרת
            sendEmailIntent.PutExtra(Intent.ExtraText, string.Format("Content")); // תוכן
            // הצגת דף שליחת אימייל עם התכנים הרלוונטיים
            StartActivity(sendEmailIntent);
        }

    }
```

UserListActivity

משויך ל-userList_layout. האקטיביטי של דף רשימת המשתמשים ניתן לגישה רק ע"י מנהל מדף הבית. הדף מכיל רשימת משתמשים של כל המשתמשים ורשימת משתמשים אשר מתעדכנת ובעצם מעדכנת את הListView המציג את רשומות המשתמשים. בפעולה OnCreate ישנה הצגה של רשומות משתמשים כך שישנה שליפה של כל המשתמשים ממסד הנתונים והשמתם ברשימה ולאחר מכן ישנו עדכון של הListView המציג את רשומות המשתמשים באמצעות עדכון UserAdapter. כמו כן ישנו עדכון של רשימת שמות המשתמשים אשר מופיעה מתחת לפקד סרגל החיפוש ברגע כתיבת טקסט. בעת הקלדת טקסט בסרגל החיפוש ישנו סינון של משתמשים על פי שם המשתמש וכן מתחת לסרגל החיפוש ישנה רשימה עם שמות המשתמשים הרלוונטיים בהתאם לחיפוש. כמו כן, לחיצה על כותרת שם המשתמש מסדרת את רשומות המשתמשים בסדר אלפביתי עולה/יורד לפי שמות המשתמשים.
הפעולה מממשת את הממשק AdapterView.IOnItemClickListener המאפשר לחיצה על רשומה ומעבר לדף ההגדות של המשתמש הנלחץ. הפעולה גם מממשת את הממשק
`AdapterView.IOnItemLongClickListener` המאפשר לחיצה ארוכה על רשומת משתמש ופתיחת AlertDialog אשר מאפשר העלאה/הורדה בדרגה של מנהל וגם אפשרות למחוק משתמש.
בעת מחיקת משתמש הפעולה תחילה עוברת על קודי המתכונים שהמשתמש הנמחק יצר ועל כל המשתמשים הרשומים לאפליקציה; ובודקת האם המשתמש הנ"ל אוהב את המתכונים של המשתמש הנמחק; במידה ומשתמש אוהב את אחד המתכונים שהמשתמש הנמחק יצר, קוד המתכון נמחק אצל המשתמש הנ"ל בתכונת קודי המתכונים האהובים עליו וכך בעצם המתכון לא יופיע עוד בדף המתכונים האהובים שלו. לאחר מכן, הפעולה עוברת על המתכונים שהמשתמש הנמחק אוהב (ולא יצר) ומורידה אצלם את מספר הלייקים. בסוף הפעולה כל המתכונים שהמשתמש הנמחק יצר נמחקים ממסד הנתונים, המשתמש עצמו נמחק והListView של רשומות המשתמשים מתעדכן.
מנהל רשאי לשלוח הודעה דרך האימייל למשתמש הנמחק ובסוף פעולת המחיקה יפתח דף מילוי לשליחת הודעה לאימייל המשתמש הנמחק.

```
[Activity(Label = "UserListActivity")]
 public class UserListActivity : Activity, AdapterView.IOnItemClickListener,
 AdapterView.IOnItemLongClickListener
 {
     // רשימת כל המשתמשים באפליקציה
     private List<User> userList;
     // רשימת משתמשים מתעדכנת לפי הטקסט בסרגל החיפוש
     private List<User> currentUsers;
     // currentUsers אדפטר של המתכונים לפי רשימת
     private UserAdapter userAdapter;
     // המכיל בתוכו את כל המשתמשים המוצגים במסך ListView
     private ListView lvUsers;
     // usernameList רשימת שמות המשתמשים המתעדכנת לפי
     private List<string> usernameList;
     // אדפטר של שמות המשתמשים שמאפשר הצגת שמות משתמשים ברגע החיפוש בהתאמה
     private ArrayAdapter usernameAdapter;
     // המתפקד כמו EditText ומציג מתחתיו את שמות המשתמשים לפי החיפוש
     AutoCompleteTextView פקד מטיפוס
     private AutoCompleteTextView actvSearch;
     // כותרת טור שמות המשתמשים
     private TextView tvHeaderUsername;
     // משתנה בוליאני שמייצג הצגת משתמשים בצורה אלפביתית או הפוך
     private bool usernameAscending;
     // כפתור חזרה לדף הבית
     private Button btnGoBack;
     protected override void OnCreate(Bundle savedInstanceState)
     {
         base.OnCreate(savedInstanceState);
         SetContentView(Resource.Layout.userList_layout);
         lvUsers = FindViewById<ListView>(Resource.Id.userList_lvUsers);
         tvHeaderUsername =  FindViewById<TextView>(Resource.Id.userList_tvHeaderUsername);
         actvSearch = FindViewById<AutoCompleteTextView>(Resource.Id.userList_actvSearch);
         btnGoBack = FindViewById<Button>(Resource.Id.userList_btnGoBack);
         // קבלת כל המשתמשים ממחלקת עזר
         userList = DatabaseManager.GetAllUsers();
         //סידור רשימת המשתמשים לפי הסדר האלפבתי והמספרי של המשתמשים
         userList = (from user in userList orderby user.username select user).ToList();
         //רשימה שתכיל את כל הערכים המתעדכנים כאשר מסננים את המשתמשים
         currentUsers = userList.ToList();
         // יצירת אדפטר המשתמשים והשמתו בlvRecipes.Adapter כך שהמשתמשים מוצגים למשתמש
         userAdapter = new UserAdapter(this, currentUsers);
         lvUsers.Adapter = userAdapter;
         lvUsers.OnItemClickListener = this; // Click on view - open user settings
         lvUsers.OnItemLongClickListener = this; // Long click on view - delete or promote user
         // תחילה המשתמשים מסודרים בסדר אלפביתי ולכן המשתנה הבוליאני יראה אמת
         usernameAscending = true;
         // עדכון actvSearch שיציג את כל המשתמשים בהתאמה
         UpdateUsernameAdapter();
         tvHeaderUsername.Click += TvHeaderUsername_Click;
         actvSearch.TextChanged += ActvSearch_TextChanged;
         btnGoBack.Click += delegate { Finish(); };
     }
     /* actvSearch-פעולת עזר שמטרתה לעדכן את האדפטר של שמות המשתמשים כך ש
      יראה את שמות המשתמשים הקיימים בפתיחת הדף וכאשר מנהל מוחק משתמש */
     private void UpdateUsernameAdapter()
     {
         // רשימת שמות כל המשתמשים
         usernameList = new List<string>();
         // לולאה שמכניסה את ערכי כל שמות המשתמשים לרשימה
         for (int i = 0; i < userList.Count; i++)
         {
             usernameList.Add(userList[i].username);
         }
         // יצירת ArrayAdapter אשר מייצג אדפטר של שמות כל המשתמשים
 usernameAdapter = new ArrayAdapter(this, Android.Resource.Layout.SimpleListItem1, usernameList);
         // השמת האדפטר הנ"ל בactvSearch.Adapter כך שסרגל החיפוש מראה את שמות המשתמשים
         actvSearch.Adapter = usernameAdapter;
     }
     // פעולה שנקראת כשכותרת שמות המשתמשים נלחצת
     private void TvHeaderUsername_Click(object sender, EventArgs e)
     {
         // בדיקה אם המשתנה הבוליאני מראה אמת כלומר הרשימה מסודרת אלפביתית
         if (usernameAscending)
         {
             // סידור רשימת כל המשתמשים ורשימת המשתמשים המתעדכנת בסדר הפוך
currentUsers = (from user in currentUsers orderby user.username descending select user).ToList();
        userList = (from user in userList orderby user.username descending select user).ToList();
         }
         else
```

```
            {
                // סידור רשימת כל המשתמשים ורשימת המשתמשים המתעדכנת בסדר אלפביתי
                currentUsers = (from user in currentUsers orderby user.username select
                user).ToList();
                userList = (from user in userList orderby user.username select user).ToList();
            }
            // עדכון רשימת המשתמשים המתעדכנת באדפטר ועדכון ListView המציג את המשתמשים
            userAdapter.SetUsers(currentUsers);
            userAdapter.NotifyDataSetChanged();
            // שינוי הערך הבוליאני של המשתנה
            usernameAscending = !usernameAscending;
        }
        // פעולה הנקראת כאשר טקסט מוקלד בסרגל החיפוש
        private void ActvSearch_TextChanged(object sender, Android.Text.TextChangedEventArgs e)
        {
// currentUsers מצביעה על רשימה כך ששמות המשתמשים מתחילים בטקסט שהוקלד, אין הפרדה לאותיות קטנות וגדולות בחיפוש //
            currentUsers = (from user in userList where user.username.StartsWith(actvSearch.Text,
            StringComparison.OrdinalIgnoreCase) select user).ToList();
            // עדכון אדפטר המשתמשים ועדכון המשתמשים בדף
            userAdapter.SetUsers(currentUsers);
            userAdapter.NotifyDataSetChanged();
        }
        // צפייה בפרטי משתמש על ידי לחיצה על רשומת המשתמש
        public void OnItemClick(AdapterView parent, View view, int position, long id)
        {
            // פתיחת טופס הגדרות המשתמש
            FragmentTransaction transaction = FragmentManager.BeginTransaction();
            SettingsDialogFragment settingsDialogFragment = new
            SettingsDialogFragment(currentUsers[position]);
            settingsDialogFragment.Show(transaction, "Settings dialog fragment");
        }
        // פעולה שממומשת בSettingsDialogFragment לאחר עדכון סיסמא של משתמש ע"י מנהל במסד הנתונים
        // הפעולה מקבלת שם משתמש וסיסמא חדשה
        public void UpdatePassword(string username, string newPassword)
        {
            // קבלת המשתמש ממסד הנתונים על פי שם המשתמש
            User currentUser = (from user in userList where user.username == username select
            user).Single();
            // עדכון שם המשתמש הנ"ל ברשומה ועדכון הListview
            currentUser.password = newPassword;
            userAdapter.NotifyDataSetChanged();
        }

        // לחיצה ארוכה על רשומת משתמש פותחת דיאלוג עם אפשרות לקידום או מחיקת משתמש
        public bool OnItemLongClick(AdapterView parent, View view, int position, long id)
        {
            // קבלת המשתמש הנלחץ לפי מקומו ברשימה
            User currentUser = currentUsers[position];
            // מנהל לא יכול למחוק את עצמו או להוריד את עצמו לדרגת משתמש רגיל
            // בדיקה האם המשתמש הנלחץ הוא לא המשתמש המחובר כלומר המנהל
            if (currentUser.username != SharedPreferencesManager.GetUsername())
            {
                // יצירת דיאלוג למחיקת משתמש או העלאתו למנהל ולהפך
                AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
                alertDialog.SetTitle("Delete User");
            alertDialog.SetMessage("Are you sure you want to delete: " + currentUser.username + "?");
                alertDialog.SetCancelable(true);
                // בדיקה האם המשתמש הנלחץ הוא לא מנהל
                if (currentUser.isAdmin == false)
                {
                    // מנהל יכול לקדם כל משתמש לדרגת מנהל
                    alertDialog.SetPositiveButton("ADD ADMIN", delegate
                    {
                        // עדכון התכונה בוליאנית המציינת האם המשתמש מנהל במסד הנתונים
                        // עתה המשתמש הנ"ל מנהל
                        currentUser.isAdmin = true;
                        DatabaseManager.SetUser(currentUser);
                        // עדכון הרשומה ככך שעתה צבע הרשומה הוא אדום כהה
                        userAdapter.NotifyDataSetChanged();
                    });
                    // מנהל יכול למחוק כל משתמש באפליקציה חוץ ממנהלים אחרים
                    alertDialog.SetNegativeButton("DELETE", delegate
                    {
                /* המרת מחרוזת הקודים של המתכונים שהמשתמש הנמחק יצר המופרדים בפסיקים לרשימה אשר מכילה את הקודים
                StringSplitOptions.RemoveEmptyEntries מאחר והתו אחרי הפסיק האחרון הוא מחרוזת ריקה אזי יש להשתמש ב
                            על מנת שהרשימה תכיל את קודי המתכונים בלבד */
                        List<string> myRecipeIdList = currentUser.myRecipesId.Split(',',
                        StringSplitOptions.RemoveEmptyEntries).ToList();
                        // מעבר על כל המשתמשים הרשומים באפליקציה
```

```
                for (int i = 0; i < userList.Count; i++)
                {
                    // מעבר כל פעם על רשימת קודי המתכונים שהמשתמש הנמחק יצר
                    for (int j = 0; j < myRecipeIdList.Count; j++)
                    {
                        // תנאי במידה והמשתמש אכן אוהב את המתכון של המשתמש הנמחק
                        if (userList[i].favoriteRecipesId.Contains(myRecipeIdList[j]))
                        {
                          // מחיקת קוד המתכון האהוב אצל משתמשים אחרים שאוהבים את המתכון של המשתמש הנמחק
                            userList[i].favoriteRecipesId =
                         userList[i].favoriteRecipesId.Replace(myRecipeIdList[j] + ",", "");
                            // עדכון המשתמש במסד הנתונים
                            DatabaseManager.SetUser(userList[i]);
                        }
                    }
                }
                // הורדת מספר הלייקים במתכונים שונים במידה והמשתמש הנמחק אהב אותם
                // רשימת המתכונים האהובים של המשתמש הנמחק
                List<Recipe> favoriteRecipeList =
                DatabaseManager.GetRecipesById(currentUser.favoriteRecipesId);
                // לולאה העוברת על כל המתכונים האהובים של המשתמש הנמחק
                for (int i = 0; i < favoriteRecipeList.Count; i++)
                {
                    // הורדת הלייקים תעשה במתכונים שלא נוצרו ע"י המשתמש הנמחק
                    // בדיקה ששם יוצר המתכונים האהובים של המשתמש הנמחק הוא לא המשתמש עצמו
                    if (favoriteRecipeList[i].recipeUsername != currentUser.username)
                    {
                       // הורדת הלייקים באחד אצל המתכונים האהובים של המשתמש הנמחק ועדכון המתכון במסד הנתונים
                        favoriteRecipeList[i].likeCount--;
                        DatabaseManager.SetRecipe(favoriteRecipeList[i]);
                    }
                }
                // מחיקת המתכונים שהמשתמש הנמחק יצר
                // לולאה העוברת על קודי המתכונים של המשתמש ובעזרת מחלקת עזר מוחקת כל מתכון שיצר
                for (int i = 0; i < myRecipeIdList.Count; i++)
                {
                DatabaseManager.DeleteRecipe(DatabaseManager.GetRecipe(myRecipeIdList[i]));
                }
                // מחיקת המשתמש עצמו ממסד הנתונים
                DatabaseManager.DeleteUser(currentUser);
                // הצגת הודעת Toast מתאימה
                Toast.MakeText(this, "User: " + currentUser.username + " was deleted",
                ToastLength.Long).Show();
                // מחיקת המשתמש ברשימת כל המשתמשים וברשימת המשתמשים המתעדכנת
                userList.Remove(currentUser);
                currentUsers.RemoveAt(position);
                // עדכון אדפטר שמות המשתמשים של סרגל החיפוש
                UpdateUsernameAdapter();
                // עדכון אדפטר המשתמשים המעדכן את ListView המשתמשים
                userAdapter.NotifyDataSetChanged();
                // במצב של מחיקת משתמש המנהל רשאי לשלוח לו אימייל על כך
                SendEmail(currentUser);
            });
        }
        else
        {
            // מנהל יכול להוריד כל מנהל אחר לדרגת משתמש רגיל
            alertDialog.SetPositiveButton("REMOVE ADMIN", delegate
            {
                // עדכון התכונה בוליאנית המציינת האם המשתמש מנהל במסד הנתונים
                // עתה המשתמש הנ"ל מורד מדרגת מנהל ע"י מנהל אחר
                currentUser.isAdmin = false;
                DatabaseManager.SetUser(currentUser);
                // עדכון הרשומה ככך שעתה צבע הרשומה הוא כתום
                userAdapter.NotifyDataSetChanged();
            });
        }
        alertDialog.SetNeutralButton("CANCEL", delegate
        {
            alertDialog.Dispose();
        });
        alertDialog.Show();
        return true;
    }
    else
    {
        // כאשר מנהל לוחץ לחיצה ארוכה על הרשומה שלו לא תהיה תגובה
        return false;
    }
```

Service

StatusNotificationService

הסרוויס של האפליקציה הוא שירות הודעות. שירות ההודעות מופעל כאשר המשתמש מתחבר ונרשם לאפליקציה; מעלה, מעדכן ומוחק מתכון; וכן כאשר הוא לוחץ על אייקון הלב בדף פרטי המתכון ובעצם מוסיף/מוחק את המתכון מדף המתכונים האהובים עליו.

בסרוויס ישנו רישום אוטומטי למניפסט בכדי שהוא יעבוד כראוי. הסרוויס מקבל באמצעות PutExtra את מערך המחרוזות המכיל את כותרת ותוכן ההודעה שתוצג.

הוא מבצע את הפעולה StopForeground(true) אשר בעצם מסירה הודעה קיימת.

לאחר מכן הוא מבצע את הפעולה RegisterForService אשר מקבלת מערך מחרוזות עם פרטי כותרת ותוכן ההודעה, ובודק האם גרסת המכשיר גדולה או שווה לגרסת Oreo בכדי להחליט האם יש להפעיל NotificationChannel אשר הכרחי להצגת הודעה בגרסאות חדשות אלו.

לאחר יצירת ההודעה הוא מפעיל ומקפיץ את ההודעה ומציג אותה במכשיר.

```
// Service -רישום אוטומטי לקובץ המניפסט כ
[Service(Label = "StatusNotificationService")]
public class StatusNotificationService : Service
{
    // NotificatioChannel מפתח ערוץ הודעות הסטטוס במקרה של שימוש ב
    private const string CHANNEL_ID = "Channel_StatusNotification";
    // מפתח Foreground Service של השירות הנ"ל
    private const int FOREGROUND_SERVICE_NOTIFICATION_ID = 1;
    // Activity.StartService פעולה המופעלת כל פעם כאשר הסרוויס נקרא דרך הפעולה
    public override StartCommandResult OnStartCommand(Intent intent,
    StartCommandFlags flags, int startId)
    {
        // מערך מחרוזות המסמל את מידע ההודעה - כותרת ותוכן מתאימים
        string[] notificationDetails = intent.GetStringArrayExtra
        ("StatusNotification_Details");

        // Does not stop the service from running just takes it out of the
           foreground state
        // removes the notification - מחיקת הודעה קודמת
        StopForeground(true);

        // פעולת עזר המטפלת בקישור הסרוויס והפעלתו
        RegisterForService(notificationDetails);

        //.כנדרש  StartCommandResult מחזיר
          מחזיר NotSticky מסמל שהסרוויס יכול להסגר תחת עומס בזכרון
        return StartCommandResult.NotSticky;
    }

    // פעולת עזר המקבלת מערך מחרוזות המכיל את כותרת ותוכן ההודעה ויוצר הודעה מתאימה
    private void RegisterForService(string[] notificationDetails)
    {

        // Notification אובייקט הודעה
        Notification notification;

        // בדיקת אם גרסת המכשיר גדולה או שווה לגרסת Oreo
        if (Build.VERSION.SdkInt >= BuildVersionCodes.O)
        {
            // Oreo and above require Notification Channel
            // מגרסת Oreo ומעלה יש ליצור Notification עם נתינת מפתח הערוץ
            notification = new NotificationCompat.Builder(this, CHANNEL_ID)
                .SetContentTitle(notificationDetails[0])
                .SetContentText(notificationDetails[1])
                .SetSmallIcon(Resource.Drawable.YumYumImage)
                .SetAutoCancel(true)
                .Build();
        }
        else
        {
            // Pre - Oreo behavior
            // בגרסאות שיצאו טרם Oreo לא נדרש שימוש ב NotificationChannel
            notification = new NotificationCompat.Builder(this)
                .SetContentTitle(notificationDetails[0])
                .SetContentText(notificationDetails[1])
                .SetSmallIcon(Resource.Drawable.YumYumImage)
                .SetAutoCancel(true)
                .Build();
        }

        // Enlist this instance of the service as a foreground service
        // יצירת הסרוויס כ Foreground Service, התחלתו והצגת הודעה מתאימה
        StartForeground(FOREGROUND_SERVICE_NOTIFICATION_ID, notification);
    }
    // פעולה שחובה להצהיר עליה בסרוויס
    public override IBinder OnBind(Intent intent)
    {
        return null;
    }
}
```

BroadcastReceiver

BatteryBroadcastReceiver

הBroadcastReceiver של האפליקציה הוא ברודכסט אשר מטרתו היא להאזין לאחוז הסוללה שבמכשיר. בברודכסט ישנו רישום אוטומטי למניפסט בכדי שהוא יעבוד כראוי וכן הצהרה על הIntentFilter אשר בעזרתו הוא מאזין לשינויים בסוללה. ישנה פעולה בונה ריקה מאחר ורישום אוטומטי במניפסט מחייב זאת. הפעולה הבונה השנייה אשר משתמשים בה בפועל מקבלת פקד TextView אשר מוצג בדף הבית ובעצם מציג את אחוז הסוללה. הפעולה OnReceive המופעלת כאשר הברודכסט מתעדכן מקבלת את נתוני הסוללה ומציבה אותם בטקסט הפקד לתצוגה.
כמו כן ישנה בדיקה שאחוז הסוללה לא נמוך מדי וכאשר הוא קטן מ15% ישנה הודעת Toast על כך.

```
// BroadCastReceiver רישום אוטומטי לקובץ המניפסט כ-
[BroadcastReceiver(Label = "BatteryBroadcastReceiver")]
// הצהרה על IntentFilter ממנו נתוני הסוללה מואזנים כאשר יש שינוי בסוללה
[IntentFilter(new[] { Intent.ActionBatteryChanged })]
public class BatteryBroadcastReceiver : BroadcastReceiver
{
    //פקד TextView המצוי בדף הבית המראה את כמות הסוללה
    private TextView tvBattery;

    // מחייב פעולה בונה ברירת מחדל בגלל הרישום האוטומטי למניפסט BroadcastReceiver
    public BatteryBroadcastReceiver()
    {

    }

    // פעולה בונה המקבלת TextView מדף הבית
    public BatteryBroadcastReceiver(TextView tvBattery)
    {
        this.tvBattery = tvBattery;
    }

    // פעולה המופעלת כאשר ה-BroadcastReceiver מתעדכן
    public override void OnReceive(Context context, Intent intent)
    {
        // קבלת מידע על אחוז הסוללה בכל רגע שאחוז הסוללה משתנה
        int battery = intent.GetIntExtra("level", 0);

        // TextView הצגת אחוז הסוללה ב-
        tvBattery.Text = "Battery: " + battery + "%";
        if (battery < 15)
        {
            // הצגת הודעת Toast בדף הבית במידה וסוללת המכשיר נמוכה מ15%
            Toast.MakeText(context, "Battery is low - " + battery + "%,
            charge your phone in order to continue viewing our wide variety
            of recipes!", ToastLength.Long).Show();
        }
    }
}
```

DialogFragment

CategoryDialogFragment

משויך ל-categoryDialogFragment_layout. כולל קישור ל10 כפתורים המציגים קטגוריות וכפתור לאישור. כפתורי הקטגוריות שמורות באמצעות מערך כפתורים, ישנו קבוע מטיפוס int שמייצג את מספר הקטגוריות המקסימלי לבחירה – 3 ומשתנה שמראה את כמות הקטגוריות שנבחרו. ברגע לחיצה על כפתור ישנה בדיקה אם צבע טקסט הכפתור הוא לבן אם כן ישנה בדיקה נוספת האם מספר הקטגוריות שנבחרו קטן מהקבוע ורק אז מוסיף 1 למשתנה הסופר את הקטגוריות ומשנה את צבע טקסט הפקד לאדום. אם הצבע הוא אדום אזי הוא מוריד 1 ומשנה את הצבע ללבן.

ברגע האישור והשמירה ישנה בדיקה האם נבחרו קטגוריות במידה ולא תופיע הודעה מתאימה, במדיה וכן ישנו מעבר על כל הקטגוריות והכנסת שם הקטגוריות לרשימה רק אם צבע הטקסט הוא אדום.

בהמשך ישנה המרה מרשימת מחרוזות למחרוזת אחת המכילה את הקטגוריות כאשר כל שם מופרד בפסיק. בסוף הפעולה ישנה בדיקה מאיזה אקטיביטי Dialog זומן בכדי לשלוח את הקטגוריות לאקטיביטי המתאים ולמטרה הנדרשת: הצגת קטגוריות – CreateRecipeActivity, RecipePageActivity; סינון מתכונים לפי קטגוריות – HomepageActivity, FavoriteRecipesActivity, MyRecipesActivity.

```
public class CategoryDialogFragment : DialogFragment
{
    // מערך המייצג את הכפתורים המכילים את הקטגוריות השונות
    private Button[] btnCategories;
    // כפתור שמירה
    private Button btnSave;
    // משתנה המציין את את כמות הקטגוריות שניבחרו
    private int count;
    // מספר המציין את כמות הקטגוריות המקסימלי שניתן לבחור
    private const int MAX_CATEGORY_COUNT = 3;
    // DialogFragment פעולה המופעלת בעת היווצרות
    public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle
    savedInstanceState)
    {
        base.OnCreateView(inflater, container, savedInstanceState);
        // ניפוח DialogFragment כך שהוא יוצג על המסך
  View view = inflater.Inflate(Resource.Layout.categoryDialogFragment_layout, container, false);
        // קישור פקדי כפתורי הקטגוריות - סך הכל עשר קטגוריות
        btnCategories = new Button[10];
        btnCategories[0] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory1);
        btnCategories[1] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory2);
        btnCategories[2] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory3);
        btnCategories[3] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory4);
        btnCategories[4] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory5);
        btnCategories[5] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory6);
        btnCategories[6] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory7);
        btnCategories[7] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory8);
        btnCategories[8] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory9);
        btnCategories[9] =
        view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnCategory10);
        btnSave = view.FindViewById<Button>(Resource.Id.categoryDialogFragment_btnSave);
        // מספר הקטגוריות ההתחלתי הוא 0
        count = 0;
        // לולאה אשר מטפלת בלחיצה על כל כפתור - סך הכל 10 כפתורים
        for (int i = 0; i < btnCategories.Length; i++)
        {
            btnCategories[i].Click += BtnCategories_Click;
        }
        btnSave.Click += BtnSave_Click;
        return view;
    }
    // פעולה המופעלת בעת לחיצה על כפתור קטגוריה
    private void BtnCategories_Click(object sender, EventArgs e)
    {
        // מייצג את פקד הכתור הנלחץ
        Button category = (Button)sender;
        // בדיקה האם צבע טקסט הכפתור הנלחץ הוא לבן
        if (btnCategories[Array.IndexOf(btnCategories, category)].CurrentTextColor==Color.White)
        {
            // אם כן ישנה בדיקה האם כמות הקטגוריות שנבחרו כבר קטנה מהכמות המקסימלית המורשת
            if (count < MAX_CATEGORY_COUNT)
            {
                // אם כן כמות הקטגוריות עולה באחד וצבע טקסט הכפתור משתנה לאדום
                count++;
                btnCategories[Array.IndexOf(btnCategories,category)].SetTextColor
                (Color.ParseColor("#ffd5284a"));
            }
        }
        else
        {
            // אם לא אזי צבע הכפתור הוא אדום, כמות הקטגוריות יורדת באחד וצבע טקסט הכפתור משתנה ללבן
            count--;
            btnCategories[Array.IndexOf(btnCategories, category)].SetTextColor(Color.White);
        }
    }
```

```
        // פעולה המופעלת כאשר כפתור השמירה נלחץ
        private void BtnSave_Click(object sender, EventArgs e)
        {
            // בדיקה אם כמות הקטגוריות שונה מאפס כלומר בין 1 ל-3
            if (count != 0)
            {
                // רשימת מחרוזות אשר תכיל את שמות הקטגוריות שנבחרו
                List<string> categoryList = new List<string>();

           //לולאה אשר עוברת על כל כפתור ובודקת אם טקסט הכפתור אדום, כלומר הכפתור נלחץ ושמה את שם הקטגוריה ברשימה הנ"ל//
                for (int i = 0; i < btnCategories.Length; i++)
                {
                    if (btnCategories[i].CurrentTextColor == Color.ParseColor("#ffd5284a"))
                    {
                        categoryList.Add(btnCategories[i].Text);
                    }
                }
                // שמות הקטגוריות מועברות למחרוזת אחת שמופרדת בפסיקים
                string categories = string.Join(",", categoryList);

                /* מישם בכמה Activities לשם פעולות שונות CategoryDialogFragment
                 לכן ישנה בדיקה איזה אקטיביטי נמצא ברקע */
                if (Activity is CreateRecipeActivity)
                {
                    //קריאה לפעולה שנמצאת ב-CreateRecipeActivity שמציגה את הקטגוריות שנבחרו בעת הכנת מתכון
                    ((CreateRecipeActivity)Activity).ReceiveCategories(categories);
                }
                if (Activity is RecipePageActivity)
                {
                    // קריאה לפעולה שנמצאת ב-RecipePageActivity שמציגה את הקטגוריות שנבחרו בעת עדכון מתכון
                    ((RecipePageActivity)Activity).ReceiveCategories(categories);
                }

                if (Activity is HomepageActivity)
                {
                    // קריאה לפעולה שנמצאת ב-HomepageActivity שמסננת את המתכונים המוצגים לפי הקטגוריות שנבחרו
                    ((HomepageActivity)Activity).ReceiveCategories(categories);
                }
                if (Activity is FavoriteRecipesActivity)
                {
                   //קריאה לפעולה שנמצאת ב-FavoriteRecipesActivity שמסננת את המתכונים המוצגים לפי הקטגוריות שנבחרו
                    ((FavoriteRecipesActivity)Activity).ReceiveCategories(categories);
                }
                if (Activity is MyRecipesActivity)
                {
                    // קריאה לפעולה שנמצאת ב-MyRecipesActivity שמסננת את המתכונים המוצגים לפי הקטגוריות שנבחרו
                    ((MyRecipesActivity)Activity).ReceiveCategories(categories);
                }
                Dismiss();
            }
            else
            {
                // כאשר כמות הקטגוריות שווה לאפס ונלחץ כפתור השמירה תוצג שגיאה באמצעות AlertDialog
                ShowAlertDialog();
            }
        }
        // פעולה אשר מציגה AlertDialog כשהיא נקראת ומודיעה על כך שאף קטגוריה לא נלחצה
        private void ShowAlertDialog()
        {
            AlertDialog.Builder alertDialog = new AlertDialog.Builder(Context);
            alertDialog.SetTitle("Error");
            alertDialog.SetMessage("No categories were saved");
            alertDialog.SetCancelable(true);
            alertDialog.SetNeutralButton("CANCEL", delegate
            {
                alertDialog.Dispose();
            });
            alertDialog.Show();
        }
    }
```

SaladDialogFragment

משויך ל-saladDialogFragment_layout. כולל קישורים לכל פקדי הכפתורים שבדף. בדף פרטי הסלט ישנם 4 חלקים: 2 כפתורים המציינים האם הסלט הוא סלט ירקות או פירות, כפתורים לסוג הסלט, 2 כפתורים המציינים האם יש בשר בסלט ו2 כפתורים המציינים האם יש טיבול. הקישור לכפתורים נעשה ע"י מערך כפתורים לכל נושא. כמו כן לכל נושא ישנה מחרוזת המציינת את בחירת המשתמש.

```
public class SaladDialogFragment : DialogFragment
{
    // מערך המייצג את הכפתורים המכילים האם מדובר בסלט פירות/ירקות
    private Button[] btnIsFruit;
    // מחרוזת המציינת פירות/ירקות
    private string isFruit;
    // מערך המייצג את הכפתורים המכילים את סוגי הסלטים
    private Button[] btnSaladType;
    // מחרוזת סוג הסלט שנבחר
    private string saladType;
    // מערך המייצג את הכפתורים המכילים האם הסלט כולל בשר
    private Button[] btnIsMeat;
    // מחרוזת המציינת כן/לא מכיל בשר
    private string isMeat;
    // מערך המייצג את הכפתורים המכילים האם הסלט כולל טיבול
    private Button[] btnIsFlavoring;
    // מחרוזת המציינת כן/לא מכיל טיבול
    private string isFlavoring;
    // כפתור שמירה בסיום הטופס
    private Button btnSave;
    // משתנה השומר את ההודעה המתאימה במקרה של שגיאה
    private string errorText;

    public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle
    savedInstanceState)
    {
        base.OnCreateView(inflater, container, savedInstanceState);

        // ניפוח DialogFragment כך שהוא יוצג על המסך
     View view = inflater.Inflate(Resource.Layout.saladDialogFragment_layout, container, false);
        isFruit = "";
        // מערך כפתורי האם הסלט מכיל פירות/ירקות מכיל 2 אפשרויות
        btnIsFruit = new Button[2];
        btnIsFruit[0] = view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnIsFruit1);
        btnIsFruit[1] = view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnIsFruit2);
        // לולאה המטפלת בכפתורים הנ"ל ברגע לחיצה
        for (int i = 0; i < btnIsFruit.Length; i++)
        {
            btnIsFruit[i].Click += BtnIsFruit_Click;
        }
        saladType = "";
        // מערך כפתורי סוגי סלט מכיל 4 סוגים
        btnSaladType = new Button[4];
        btnSaladType[0] =
        view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnSaladType1);
        btnSaladType[1] =
        view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnSaladType2);
        btnSaladType[2] =
        view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnSaladType3);
        btnSaladType[3] =
        view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnSaladType4);
        // לולאה המטפלת בכפתורים הנ"ל ברגע לחיצה
        for (int i = 0; i < btnSaladType.Length; i++)
        {
            btnSaladType[i].Click += BtnSaladType_Click;
        }

        isMeat = "";
        // מערך כפתורי האם הסלט מכיל בשר מכיל 2 אפשרויות
        btnIsMeat = new Button[2];
        btnIsMeat[0] = view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnIsMeatYes);
        btnIsMeat[1] = view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnIsMeatNo);
        // לולאה המטפלת בכפתורים הנ"ל ברגע לחיצה
        for (int i = 0; i < btnIsMeat.Length; i++)
        {
            btnIsMeat[i].Click += BtnIsMeat_Click;
        }
        isFlavoring = "";
        // מערך כפתורי האם הסלט מכיל טיבול מכיל 2 אפשרויות
        btnIsFlavoring = new Button[2];
        btnIsFlavoring[0] =
        view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnIsFlavoringYes);
        btnIsFlavoring[1] =
        view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnIsFlavoringNo);
        // לולאה המטפלת בכפתורים הנ"ל ברגע לחיצה
        for (int i = 0; i < btnIsFlavoring.Length; i++)
        {
            btnIsFlavoring[i].Click += BtnIsFlavoring_Click;
        }
        btnSave = view.FindViewById<Button>(Resource.Id.saladDialogFragment_btnSave);
        btnSave.Click += BtnSave_Click;

        return view;
    }
```

לכל נושא בפרטי הסלט ישנה פעולה המופעלת כאשר לוחצים על אחד מכפתורי הנושא הרלוונטי.
הפעולה תחילה בודקת האם צבע טקסט הכפתור הוא לבן, במידה וכן היא בודקת האם המחרוזת השומרת את הערך שנבחר ריק, כלומר לא נבחר עדיין ערך. אם היא ריקה טקסט הכפתור הופך לאדום וערך פרט הכפתור יהיה שמור במשתנה המחרוזת. אם צבע הכפתור הנלחץ הוא אדום הפעולה מחזירה את צבע הטקסט ללבן וערך המחרוזת יהיה ריק כלומר אין ערך שמור שנבחר.
כלומר, בכל נושא בפרטי הסלט ניתן לבחור פרט אחד בלבד. בעת אישור וסיום הפעולה מבצעת בדיקות תקינות שכל השדות נבחרו ובדומה לCategoryDialogFragment גם כאן ישנה בדיקה מאיזה אקטיביטי הDialog זומן בכדי לשלוח את פרטי הסלט לאקטיביטי המתאים.
ההסבר תקף גם לSoupDialogFragment, MeatDialogFragment ו-PastryDialogFragmen מאחר ומטרת טפסי המילוי זהה ועקרון הקוד והפעולות זהה, אולם נושאי הפרטים שונים וכן חלק מהדפים כוללים פקדי EditText ובדיקות תקינות נוספות.

```
// פעולה שנקראת כאשר לוחצים על אחד הכפתורים של האם פירות או ירקות
private void BtnIsFruit_Click(object sender, EventArgs e)
{
    // מייצג את פקד הכתור הנלחץ
    Button button = (Button)sender;
    // בדיקה האם צבע טקסט הכפתור הנלחץ הוא לבן
    if (btnIsFruit[Array.IndexOf(btnIsFruit, button)].CurrentTextColor == Color.White)
    {
        // אם כן ישנה בדיקה האם כבר נלחץ כפתור נוסף ואפשרות כבר שמורה במשתנה isFruit
        if (isFruit == "")
        {
            // אם המחרוזת ריקה ערכה עתה הוא האפשרות הנבחרת וצבע טקסט הכפתור משתנה לאדום
            btnIsFruit[Array.IndexOf(btnIsFruit,
            button)].SetTextColor(Color.ParseColor("#ffd5284a"));
            isFruit = btnIsFruit[Array.IndexOf(btnIsFruit, button)].Text;
        }
    }
    else
    {
        // אם לא אזי צבע הכפתור הוא אדום, המחרוזת תהפוך לריקה וצבע טקסט הכפתור משתנה ללבן
        btnIsFruit[Array.IndexOf(btnIsFruit, button)].SetTextColor(Color.White);
        isFruit = "";
    }
}
```

SoupDialogFragment

משויך ל-soupDialogFragment_layout. כולל קישורים לכל פקדי הכפתורים שבדף. בדף פרטי המרק ישנם 4 חלקים: כפתורים לסוג מרק, 2 כפתורים המציינים האם מדובר בבישול ארוך או קצר ו2 פקדי EditText המציינים את זמן וטמפרטורת הבישול. ישנם בדיקות תקינות שהפקדים מולאו כראוי.

MeatDialogFragment

משויך ל-meatDialogFragment_layout. כולל קישורים לכל פקדי הכפתורים שבדף. בדף פרטי מנת הבשר ישנם 4 חלקים: כפתורים לסוג הבשר, כפתורים לדרך הכנת הבשר, כפתורים לרמת צליית/בישול הבשר ופקד EditText לזמן בישול הבשר. ישנה בדיקת תקינות שפקד הEditText מולא כראוי.

PastryDialogFragment

משויך ל-pastryDialogFragment_layout. כולל קישורים לכל פקדי הכפתורים שבדף. בדף פרטי הסלט ישנם 4 חלקים: 2 כפתורים המציינים האם המאפה מכיל גלוטן, כפתורים לסוג הכשרות של המאפה ופקדי EditText לזמן וטמפרטורת האפייה. ישנם בדיקות תקינות שהפקדים מולאו כראוי.

ForgotPasswordDialogFragment

משויך ל-forgotPasswordDialogFragment_layout. כולל קישור לפקדי EditText של שם פרטי, אימייל ותשובה על שאלת אבטחה. לאחר מילוי השדות ולחיצה על כפתור האישור מופעלת פעולה באמצעות מחלקת העזר DatabaseManager מופעלת בדיקה האם שלושת השדות תקינים ומופיעים בטבלת המשתמשים אצל משתמש מסוים. במידה והפרטים נכונים עריכת השדות הללו משובתת וניתן להקליד סיסמא חדש בפקד נוסף ולאשר זאת. קודם ישנן בדיקות תקינות לסיסמא כמו בדף ההרשמה ולאחר שהכל תקין מעדכנת את הסיסמא של המשתמש במסד הנתונים ובאמצעות ISharedPreferences שומרת את שם המשתמש והסטטוס (זכור במערכת) בקובץ פנימי במכשיר.
לאחר מכן ישנה סגירה של הDialogFragment והאקטיביטי של ההתחברות ומעבר לדף הבית.

SettingsDialogFragment

משויך ל-settingsDialogFragment_layout. כולל קישור לפקדי TextView המציגים את נתוני המשתמש המחובר: שם פרטי, שם משתמש, אימייל, תשובה על שאלת אבטחה, וכמות המתכונים האהובים והמתכונים שהמשתמש יצר. ברגע לחיצה על פקד תמונת המשתמש ניתנת אפשרות לצלם באמצעות המצלמה את תמונת הפרופיל בדומה לדף הכנת המתכון וכן ישנה אפשרות למחוק את התמונה הקיימת. באפשרות משתמש לשנות סיסמא ולאחר שהקליד סיסמא והיא תקינה בהתאם לבדיקות התקינות תופיעה הודעה על כך. מנהל רשאי לשנות סיסמא של משתמשים באפליקציה ולאחר השינוי ישנו קישור לדף לשליחת הודעה מתאימה לאימייל המשתמש. לאחר השינויים הללו פרטי המשתמש מתעדכנים במסד הנתונים.
בנוסף לכך, ישנו כפתור להפעלת/ביטול שירות ההודעות כך שלחיצה עליו משנה את צבעו לירוק/אדום בהתאמה וכן מפעילה/משביתה את שירות ההודעות. בכדי שהמערכת תדע שהמשתמש מעוניין בשירות ההודעות או לא, למשתמש ישנה תכונה בוליאנית המציינת זאת ובעת שינוי התכונה משתנה.

RecipeTypeDialogFragment

משויך ל-recipeTypeDialogFragment_layout. כולל קישורים לפקדי כפתורים המייצגים את סוגי המתכון באפליקציה: סלט, מרק, בשר ומאפה ומקשר את הכפתורים לטפסי מילוי פרטי סוג המתכון הנבחר: SaladDialogFragment, SoupDialogFragment, MeatDialogFragment,
ו-PastryDialogFragment בהתאמה.

SortByDialogFragment

משויך ל-sortByDialogFragment_layout. כולל קישורים לפקדי כפתורים המייצגים את אפשרויות סינון המתכונים שבאפליקציה: סדר אלפביתי, מתכונים פופולריים, סינון לפי קטגוריות, מתכונים חדשים וסינון לפי סוג מתכון. המחלקה כוללת רשימת מתכונים ותחילה ישנה בדיקה מאיזה אקטיביטי הDialog זומן. אם מדף הבית הרשימה תכיל את כל המתכונים באפליקציה, אם מדף המתכונים האהובים היא תכיל את המתכונים שהמשתמש אוהב ואם מדף המתכונים שלי היא תכיל את המתכונים שהמשתמש המחובר יצר. אם המשתמש בחר בסינון לפי סידור אלפביתי, סידור לפי פופולריות מתכונים וסידור לפי מתכונים חדשים המחלקה מסננת את הרשימה על פי הבחירה ומפעילה פעולה של האקטיביטי הנוכחי אשר מסננת ומסדרת את המתכונים כנדרש. אם המשתמש בחר בסינון לפי קטגוריות או סוג מתכון הפעולה פותחת את טפסי המילוי CategoryDialogFragment, RecipeTypeDialogFragment בהתאמה.

```
public class SortByDialogFragment : DialogFragment
{
    // רשימת המתכונים המיועדת לסינון
    private List<Recipe> recipeList;
    // פקד כפתור המציין סינון אלפביתי
    private Button btnAlphabetically;
    // פקד כפתור המציין סינון לפי פופולריות
    private Button btnPopular;
    // פקד כפתור המציין סינון לפי קטגוריות
    private Button btnCategories;
    // פקד כפתור המציין סינון לפי מתכונים חדשים
    private Button btnNew;
    // פקד כפתור המציין סינון לפי סוג מתכון
    private Button btnRecipeType;

    public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle
    savedInstanceState)
    {
        base.OnCreateView(inflater, container, savedInstanceState);
        // ניפוח DialogFragment כך שהוא יוצג על המסך
    View view = inflater.Inflate(Resource.Layout.sortByDialogFragment_layout, container, false);
        // סינון בהתאם לActivity הנוכחי
        if (Activity is HomepageActivity)
        {
            // הרשימה מצביעה על כל המתכונים
            recipeList = DatabaseManager.GetAllRecipes();
        }
        if (Activity is FavoriteRecipesActivity)
        {
            // הרשימה מצביעה על כל המתכונים האהובים של המשתמש המחובר
            recipeList = DatabaseManager.GetAllFavoriteRecipes();
        }
        if (Activity is MyRecipesActivity)
        {
            // הרשימה מצביעה על כל המתכונים שהמשתמש המחובר יצר
            recipeList = DatabaseManager.GetAllMyRecipes();
        }
   btnAlphabetically=view.FindViewById<Button>(Resource.Id.sortByDialogFragment_alphabetically);
        btnPopular = view.FindViewById<Button>(Resource.Id.sortByDialogFragment_popular);
        btnCategories = view.FindViewById<Button>(Resource.Id.sortByDialogFragment_categories);
        btnNew = view.FindViewById<Button>(Resource.Id.sortByDialogFragment_new);
        btnRecipeType = view.FindViewById<Button>(Resource.Id.sortByDialogFragment_recipeType);
        // לחיצה על כפתור סידור אלפביתי
        btnAlphabetically.Click += delegate
        {
            // סידור רשימת המתכונים לפי הסדר האלפבתי של המתכונים
            recipeList = (from recipe in recipeList orderby recipe.recipeName select
            recipe).ToList();
            ArrangeRecipesByActivity();
            Dismiss();
        };
        // לחיצה על כפתור סידור לפי פופולריות
        btnPopular.Click += delegate
        {
            // סידור רשימת המתכונים לפי כמות הלייקים של המתכונים מהגבוה לנמוך
            recipeList = (from recipe in recipeList orderby recipe.likeCount descending select
            recipe).ToList();
            ArrangeRecipesByActivity();
            Dismiss();
        };
        // לחיצה על כפתור סידור לפי קטגוריות
        btnCategories.Click += delegate
        {
            // פותח DialogFragment המהווה טופס בחירה של קטגוריות
            FragmentTransaction transaction = FragmentManager.BeginTransaction();
            CategoryDialogFragment categoryDialogFragment = new CategoryDialogFragment();
            categoryDialogFragment.Show(transaction, "Category dialog fragment");
            Dismiss();
        };
        // לחיצה על כפתור סידור לפי מתכונים חדשים
        btnNew.Click += delegate
        {
            // סידור רשימת המתכונים לפי תאריך העלאתו מחדש לישן
            recipeList = (from recipe in recipeList orderby recipe.creationTime descending
            select recipe).ToList();
            ArrangeRecipesByActivity();
            Dismiss();
        };
```

```
            // לחיצה על כפתור סידור לפי סוג מתכון
            btnRecipeType.Click += delegate
            {
                // פותח DialogFragment המהווה טופס בחירה של המידע הרצוי לגבי סוג המתכון הנבחר
                FragmentTransaction transaction = FragmentManager.BeginTransaction();
                RecipeTypeDialogFragment recipeTypeDialogFragment = new RecipeTypeDialogFragment();
                recipeTypeDialogFragment.Show(transaction, "Recipe Type dialog fragment");
                Dismiss();
            };

            return view;
        }
        // פעולת עזר המזהה את האקטיביטי הנוכחי ושולחת את את הרשימה המסוננת לאקטיביטי
        private void ArrangeRecipesByActivity()
        {
            if (Activity is HomepageActivity)
            {
                // פעולה המופעלת בדף הבית
                ((HomepageActivity)Activity).ArrangeRecipes(recipeList);
            }
            if (Activity is FavoriteRecipesActivity)
            {
                // פעולה המופעלת בדף המתכונים האהובים
                ((FavoriteRecipesActivity)Activity).ArrangeRecipes(recipeList);
            }
            if (Activity is MyRecipesActivity)
            {
                // פעולה המופעלת בדף המתכונים שהמשתמש יצר
                ((MyRecipesActivity)Activity).ArrangeRecipes(recipeList);
            }
        }
    }
```

מחלקות עזר

DatabaseManager

מחלקת העזר המטפלת במסד הנתונים. המחלקה הינה סטטית כדי לבצע את הפעולות הנוגעות למסד הנתונים ע"י שימוש בשם המחלקה בלבד במחלקות האפליקציה השונות. למחלקה גישה לטבלאות המצויות במסד הנתונים והיא מכילה פעולה ליצירת הטבלאות אשר ממומשת בכניסה הראשונה לאפליקציה. הפעולה מטפלת בטבלאות הבאות: User, Recipe, Salad, Soup, Meat, Pastry.

המחלקה מכילה פעולות שליפה, עדכון, הוספה ומחיקה של משתמשים ומתכונים, כאשר בחלק המתכונים היא מבצעת המרה כלפי מטה ומשתמשת בטבלאות הסלטים, המרקים, הבשרים והמאפים. הפעולות כוללות גם שליפה של כל המשתמשים והמתכונים באפליקציה וכן כל הסלטים, המרקים, הבשרים והמאפים. כנ"ל גם כל המתכונים האהובים והמתכונים שהמשתמש המחובר יצר (נעשה שימוש במחלקת העזר SharedPreferencesManager לקבלת המשתמש הנוכחי.

בחלק המתכונים ישנן פעולות נוספות: בדיקה האם מתכון קיים במסד הנתונים לפי קוד מתכון וגם החזרת רשימת מתכונים על פי מחרוזת הכוללת בתוכה של קודי המתכונים אשר מופקדים בפסיקים (בהתאם לתכונת המשתמש).

בחלק המשתמשים ישנן פעולות תקינות ובדיקה: האם משתמש קיים במערכת לפי שם המשתמש שנקלט, האם אימייל כבר קיים במערכת לפי אימייל שנקלט, האם המשתמש קיים במערכת בעת התחברות לפי שם משתמש וסיסמא והאם משתמש קיים במערכת בעת שחזור סיסמא על פי שם פרטי, אימייל ותשובה על שאלת אבטחה.

```
public static class DatabaseManager
{
    // קבוע שמכיל את שם מסד הנתונים של האפליקציה
    private const string DATABASE_NAME = "DatabaseYumYum";
    // Path משתנה סטטי המייצג את ההתחברות למסד הנתונים SQLite הפותח את מסד הנתונים לפי
    private static SQLiteConnection connection = new SQLiteConnection(Path());
    // פעולה שיוצרת את המסלול למסד הנתונים
    private static string Path()
    {
        // מביא את התיקייה של המסד
        string path =System.IO.Path.Combine(System.Environment.GetFolderPath(System.Environment.
        SpecialFolder.Personal), DATABASE_NAME);
        // ניגש לתיקייה פנימית בטלפון ומחזיר את הנתונים משם
        return path;
    }
    // יצירת טבלאות משתמשים ומתכונים במידה ואינם קיימים
    public static void CreateTables()
    {
        connection.CreateTable<User>();
        connection.CreateTable<Recipe>();
        connection.CreateTable<Salad>();
        connection.CreateTable<Soup>();
        connection.CreateTable<Meat>();
        connection.CreateTable<Pastry>();
    }

    // Recipe
    // פעולה המחזירה מתכון המצוי במסד הנתונים לפי קוד מתכון שאותו היא קולטת
    public static Recipe GetRecipe(string recipeId)
    {
        string strsql;
        // המתכון המוחזר
        Recipe recipe = null;
        /* בדיקה באיזה סוג מתכון קוד המתכון מתחיל
         לאחר הבדיקה בוחר את המתכון לפי קוד המתכון בטבלת סוג המתכון המתאים
         לאחר שאילתה ממסד הנתונים עצם מטיפוס מתכון מצביע עליו ומוחזר */
        if (recipeId.StartsWith("Salad"))
        {
            strsql = string.Format("SELECT * FROM Salad WHERE RecipeId='{0}'", recipeId);
            recipe = connection.Query<Salad>(strsql)[0];
        }
        if (recipeId.StartsWith("Soup"))
        {
            strsql = string.Format("SELECT * FROM Soup WHERE RecipeId='{0}'", recipeId);
            recipe = connection.Query<Soup>(strsql)[0];
        }
        if (recipeId.StartsWith("Meat"))
        {
            strsql = string.Format("SELECT * FROM Meat WHERE RecipeId='{0}'", recipeId);
            recipe = connection.Query<Meat>(strsql)[0];
        }
        if (recipeId.StartsWith("Pastry"))
        {
            strsql = string.Format("SELECT * FROM Pastry WHERE RecipeId='{0}'", recipeId);
            recipe = connection.Query<Pastry>(strsql)[0];
        }
        return recipe;
    }
    // פעולה המעדכנת מתכון המצוי במסד הנתונים לפי אובייקט מתכון שאותו היא קולטת
    public static void SetRecipe(Recipe recipe)
    {
        if (recipe is Salad)
        {
            connection.Update((Salad)recipe);
        }
        if (recipe is Soup)
        {
            connection.Update((Soup)recipe);
        }
        if (recipe is Meat)
        {
            connection.Update((Meat)recipe);
        }
        if (recipe is Pastry)
        {
            connection.Update((Pastry)recipe);
        }
    }
```

```
// פעולה המוסיפה מתכון למסד הנתונים לפי אובייקט מתכון שאותו היא קולטת
public static void AddRecipe(Recipe recipe)
{
    if (recipe is Salad)
    {
        connection.Insert((Salad)recipe);
    }
    if (recipe is Soup)
    {
        connection.Insert((Soup)recipe);
    }
    if (recipe is Meat)
    {
        connection.Insert((Meat)recipe);
    }
    if (recipe is Pastry)
    {
        connection.Insert((Pastry)recipe);
    }
}
// פעולה המוחקת מתכון המצוי במסד הנתונים לפי אובייקט מתכון שאותו היא קולטת
public static void DeleteRecipe(Recipe recipe)
{
    if (recipe is Salad)
    {
        connection.Delete((Salad)recipe);
    }
    if (recipe is Soup)
    {
        connection.Delete((Soup)recipe);
    }
    if (recipe is Meat)
    {
        connection.Delete((Meat)recipe);
    }
    if (recipe is Pastry)
    {
        connection.Delete((Pastry)recipe);
    }
}
// פעולה בוליאנית הבודקת האם המתכון קיים במסד הנתונים לפי קוד המתכון שהיא קולטת
public static bool IsExistRecipe(string recipeId)
{
    string strsql = "";
    /* ישנה בדיקה באיזה סוג מתכון קוד המתכון מתחיל
     לאחר מכן מתבצעת שאילתה
     וישנה בדיקה האם המופע הנ"ל מופיע פעם אחת כלומר האם המתכון קיים במסד */
    if (recipeId.StartsWith("Salad"))
    {
        strsql = string.Format("SELECT * FROM Salad WHERE RecipeId='{0}'", recipeId);
        if (connection.Query<Salad>(strsql).Count == 1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    if (recipeId.StartsWith("Soup"))
    {
        strsql = string.Format("SELECT * FROM Soup WHERE RecipeId='{0}'", recipeId);
        if (connection.Query<Soup>(strsql).Count == 1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    if (recipeId.StartsWith("Meat"))
    {
        strsql = string.Format("SELECT * FROM Meat WHERE RecipeId='{0}'", recipeId);
        if (connection.Query<Meat>(strsql).Count == 1)
        {
            return true;
        }
        else
```

```
            {
                return false;
            }
        }
        if (recipeId.StartsWith("Pastry"))
        {
            strsql = string.Format("SELECT * FROM Pastry WHERE RecipeId='{0}'", recipeId);
            if (connection.Query<Pastry>(strsql).Count == 1)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        return false;
    }

    /* מקבלת מחרוזת קודי מתכונים, ממירה את המחרוזת לרשימה המכילה את הקודים ומחזירה את רשימת המתכונים על פי קוד הזיהוי
      המתאים לכל מתכון
      משמש לקבלת רשימת המתכונים האהובים והמתכונים שהועלו בלבד */
    public static List<Recipe> GetRecipesById(string recipesId)
    {
        המרת מחרוזת הקודים המופרדים בפסיקים לרשימה אשר מכילה את הקודים /*
         StringSplitOptions.RemoveEmptyEntries מאחר והתו אחרי הפסיק האחרון הוא מחרוזת ריקה אזי יש להשתמש ב
         על מנת שהרשימה תכיל את קודי המתכונים בלבד */
        List<string> recipeIdList = recipesId.Split(',',
        StringSplitOptions.RemoveEmptyEntries).ToList();
        // רשימת המתכונים שתכיל את המתכונים לפי קודי המתכונים
        List<Recipe> recipeList = new List<Recipe>();
        string strsql;
        // משמש כאובייקט כללי למתכון
        Recipe recipe = null;
        /* לולאה שעוברת על רשימת קודי המתכונים ועבור כל קוד בודקת באיזה סוג מתכון קוד המתכון מתחיל
         מבצעת שאילתה מהטבלה המתאימה במסד הנתונים ולפי קוד המתכון ומקבלת את המתכון הנ"ל
         בסוף היא מוסיפה את המתכון לרשימת המתכונים */
        for (int i = 0; i < recipeIdList.Count; i++)
        {
            if (recipeIdList[i].StartsWith("Salad"))
            {
                strsql = string.Format("SELECT * FROM Salad WHERE RecipeId='{0}'",
                recipeIdList[i]);
                recipe = connection.Query<Salad>(strsql)[0];
            }
            if (recipeIdList[i].StartsWith("Soup"))
            {
                strsql = string.Format("SELECT * FROM Soup WHERE RecipeId='{0}'",
                recipeIdList[i]);
                recipe = connection.Query<Soup>(strsql)[0];
            }
            if (recipeIdList[i].StartsWith("Meat"))
            {
                strsql = string.Format("SELECT * FROM Meat WHERE RecipeId='{0}'",
                recipeIdList[i]);
                recipe = connection.Query<Meat>(strsql)[0];
            }
            if (recipeIdList[i].StartsWith("Pastry"))
            {
                strsql = string.Format("SELECT * FROM Pastry WHERE RecipeId='{0}'",
                recipeIdList[i]);
                recipe = connection.Query<Pastry>(strsql)[0];
            }
            recipeList.Add(recipe);
        }
        return recipeList;
    }
    // פעולה שמחזירה את כל המתכונים שבאפליקציה משומשת בדף הבית להצגת כל המתכונים
    public static List<Recipe> GetAllRecipes()
    {
        // רשימת כל המתכונים אשר מתווספים אליה ערכים בעזרת פעולות העזר הבאות
        List<Recipe> recipeList = new List<Recipe>();
        recipeList.AddRange(GetAllSalads());
        recipeList.AddRange(GetAllSoups());
        recipeList.AddRange(GetAllMeats());
        recipeList.AddRange(GetAllPastries());
        return recipeList;
    }
```

```
// פעולה המחזירה את כל הסלטים שבמסד הנתונים ברשימה מטיפוס מתכון
public static List<Recipe> GetAllSalads()
{
    // רשימה מטיפוס מתכון שמייצגת את כל הסלטים
    List<Recipe> saladList = new List<Recipe>();
    /* מתבצעת שאילתה מטבלת הסלטים שבמסד הנתונים
     saladList הסלטים נשמרים ברשימה מטיפוס סלט ואם ישנם סלטים הם מוספים לרשימה
     */ ישנה המרה כלפי מעלה מטיפוס סלט למתכון כאשר הסלטים מוספים לרשימה
    string strsql = string.Format("SELECT * FROM Salad");
    List<Salad> salads = connection.Query<Salad>(strsql);
    if (salads.Count > 0)
    {
        foreach (Salad salad in salads)
        {
            saladList.Add(salad);
        }
    }
    return saladList;
}
// פעולה המחזירה את כל המרקים שבמסד הנתונים ברשימה מטיפוס מתכון
public static List<Recipe> GetAllSoups()...
// פעולה המחזירה את כל הבשרים שבמסד הנתונים ברשימה מטיפוס מתכון
public static List<Recipe> GetAllMeats()...
// פעולה המחזירה את כל המאפים שבמסד הנתונים ברשימה מטיפוס מתכון
public static List<Recipe> GetAllPastries()...
// פעולה המחזירה את כל המתכונים האהובים של המשתמש המחובר, משומשת בדף המתכונים האהובים
public static List<Recipe> GetAllFavoriteRecipes()
{
    // SharedPreferencesManager קבלת המשתמש המחובר בעזרת פעולות עזר מהמחלקה הנ"ל ומחלקת
    User loggedUser = GetUser(SharedPreferencesManager.GetUsername());
    /* GetRecipesById רשימת המתכונים האהובים, שימוש בפעולת העזר
     */ תוך שליחת קודי המתכונים האהובים של המשתמש המחובר
    List<Recipe> recipeList = GetRecipesById(loggedUser.favoriteRecipesId);
    return recipeList;
}
// פעולה המחזירה את כל הסלטים האהובים של המשתמש המחובר
public static List<Recipe> GetAllFavoriteSalads()
{
    // רשימת המתכונים האהובים של המשתמש המחובר
    List<Recipe> recipeList = GetAllFavoriteRecipes();
    // saladList-מעבר על כל המתכונים האהובים והצבתם של סלטים ב
    List<Recipe> saladList = new List<Recipe>();
    for (int i = 0; i < recipeList.Count; i++)
    {
        if (recipeList[i] is Salad)
        {
            saladList.Add(recipeList[i]);
        }
    }
    return saladList;
}
// פעולה המחזירה את כל המרקים האהובים של המשתמש המחובר
public static List<Recipe> GetAllFavoriteSoups()...
// פעולה המחזירה את כל מנות הבשר האהובים של המשתמש המחובר
public static List<Recipe> GetAllFavoriteMeats()...
// פעולה המחזירה את כל המאפים האהובים של המשתמש המחובר
public static List<Recipe> GetAllFavoritePastries()...
// פעולה המחזירה את כל המתכונים שמשתמש המחובר יצר, משומשת בדף המתכונים שלי
public static List<Recipe> GetAllMyRecipes()
{
    // SharedPreferencesManager קבלת המשתמש המחובר בעזרת פעולות עזר מהמחלקה הנ"ל ומחלקת
    User loggedUser = GetUser(SharedPreferencesManager.GetUsername());
    /*  GetRecipesById רשימת המתכונים שלי, שימוש בפעולת העזר
     */ תוך שליחת קודי המתכונים שמשתמש המחובר יצר
    List<Recipe> recipeList = GetRecipesById(loggedUser.myRecipesId);
    return recipeList;
}
// פעולה המחזירה את כל הסלטים שמשתמש המחובר יצר
public static List<Recipe> GetAllMySalads()
{
    // רשימת המתכונים  שהמשתמש המחובר יצר
    List<Recipe> recipeList = GetAllMyRecipes();
    // saladList-מעבר על כל המתכונים שהמשתמש יצר והצבתם של סלטים ב
    List<Recipe> saladList = new List<Recipe>();
    for (int i = 0; i < recipeList.Count; i++)
    {
        if (recipeList[i] is Salad)
        {
            saladList.Add(recipeList[i]);
```

```
            }
        }
        return saladList;
    }
    // פעולה המחזירה את כל המרקים שמשתמש המחובר יצר
    public static List<Recipe> GetAllMySoups()...
    // פעולה המחזירה את כל מנות הבשר שמשתמש המחובר יצר
    public static List<Recipe> GetAllMyMeats()...
    // פעולה המחזירה את כל המאפים שמשתמש המחובר יצר
    public static List<Recipe> GetAllMyPastries()...

    // User
    // פעולה המחזירה משתמש המצוי במסד הנתונים לפי שם המשתמש שאותו היא קולטת
    public static User GetUser(string username)
    {
        string strsql = string.Format("SELECT * FROM User WHERE Username='{0}'", username);
        User user = connection.Query<User>(strsql)[0];
        return user;
    }
    // פעולה המעדכנת משתמש המצוי במסד הנתונים לפי אובייקט המשתמש שאותו היא קולטת
    public static void SetUser(User user)
    {
        connection.Update(user);
    }
    // פעולה המוסיפה משתמש למסד הנתונים לפי אובייקט משתמש שאותו היא קולטת
    public static void AddUser(User user)
    {
        connection.Insert(user);
    }
    // פעולה המוחקת משתמש המצוי במסד הנתונים לפי אובייקט המשתמש שאותו היא קולטת
    public static void DeleteUser(User user)
    {
        connection.Delete(user);
    }
    // פעולה המחזירה אמת במידה ושם המשתמש המוכנס קיים במסד נתוני המשתמשים, אחרת תחזיר שקר
    public static bool IsExistUsername(string username)
    {
        string strsql = string.Format("SELECT * FROM User WHERE Username='{0}'", username);
        if (connection.Query<User>(strsql).Count == 1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    // פעולה המחזירה אמת במידה והאימייל המוכנס קיים במסד נתוני המשתמשים, אחרת תחזיר שקר
    public static bool IsExistEmail(string email)
    {
        string strsql = string.Format("SELECT * FROM User WHERE email='{0}'", email);
        if (connection.Query<User>(strsql).Count == 1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    // פעולה המחזירה אמת במידה וקיים משתמש התואם את שם המשתמש והסיסמא, אחרת תחזיר שקר
    public static bool IsValidUser(string username, string password)
    {
        string strsql = string.Format("SELECT * FROM User WHERE Username='{0}' AND
        password='{1}'", username, password);
        if (connection.Query<User>(strsql).Count == 1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    // פעולה המחזירה אמת במידה וקיים משתמש התואם את שם המשתמש, האימייל ותשובת שאלת ההבטחה; אחרת תחזיר שקר
    public static bool IsValidUser(string username, string email, string securityQuestion)
    {
        string strsql = string.Format("SELECT * FROM User WHERE Username='{0}' AND email='{1}'
        AND securityQuestion='{2}'", username, email, securityQuestion);
```

```
            if (connection.Query<User>(strsql).Count == 1)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        // פעולה שמחזירה את כל המשתמשים הרשומים לאפליקציה, משומשת בדף רשימת המשתמשים להצגת כל המתכונים
        public static List<User> GetAllUsers()
        {
            // רשימה שתכיל את כל המשתמשים
            List<User> userList = new List<User>();
            string strsql = string.Format("SELECT * FROM User");
            List<User> users = connection.Query<User>(strsql);
            if (users.Count > 0)
            {
                foreach (User user in users)
                {
                    userList.Add(user);
                }
            }
            return userList;
        }
    }
```

SharedPreferencesManager

מחלקת העזר המטפלת בשימוש בממשק ISharedPreferences לשם שמירת נתוני המשתמש באפליקציה (שם משתמש מחובר והאם המשתמש זכור במערכת בכניסה הבאה לאפליקציה).
ISharedPreferences הינו ממשק המאפשר שמירת מידע בזיכרון המכשיר כך שתהיה אפשרות לשלוף את שם המשתמש המחובר והסטטוס שלו בהתאם לצורך.
המחלקה הינה סטטית כדי לבצע את הפעולות הנוגעות לISharedPreferences ע"י שימוש בשם המחלקה בלבד במחלקות האפליקציה השונות.
המחלקה כוללת קבועים אשר מייצגים מפתחות לציון שם הקובץ שבו שמורים הנתונים, שם המשתמש וסטטוס המשתמש. ישנה פעולה היוצרת את הקובץ הפנימי השומר את הנתונים והיא ממומשת בכניסה הראשונה לאפליקציה. המחלקה כוללת פעולות שליפה ועדכון המאפשרות שימוש במידע השמור בקובץ או שינוי סטטוס המשתמש בעת התחברות והרשמה.
בנוסף לכך, ישנה פעולה המאפסת את הנתונים השמורים, בעת התנתקות או כניסה חוזרת לאפליקציה כשאר המשתמש לא זכור במערכת.

```
public static class SharedPreferencesManager
{
    // Interface for accessing and modifying preference data
    // ממשק שמאפשר שמירת מידע בזיכרון המכשיר
    private static ISharedPreferences preferences;
    // מפתח שמציין את שם הקובץ השמור בזיכרון
    private const string USER_INFO_KEY = "UserInfo";
    // מפתח שמציין את שם המשתמש השמור בקובץ
    private const string USERNAME_KEY = "Username";
    // מפתח שמציין האם המכשיר זוכר את המשתמש בקובץ
    private const string REMEMBER_ME_KEY = "RememberMe";
    // פעולה המקבלת את Context האפליקציה הכללי
    public static void CreateSharedPreferences(Context context)
    {
        // יצירת הקובץ בContext האפליקציה כך שרק האובייקט הנ"ל ראשי לכתוב ולקרוא ממנו
        preferences = context.GetSharedPreferences(USER_INFO_KEY, FileCreationMode.Private);
    }
    // פעולה המחזירה את שם המשתמש המחובר מהקובץ
    public static string GetUsername()
    {
        return preferences.GetString(USERNAME_KEY, "");
    }
    // שינוי שם המשתמש המחובר בקובץ
    public static void SetUsername(string username)
    {
        // Interface used for modifying values in a ISharedPreferences
        // משתנה המקבל הרשאת עריכה לקובץ
        ISharedPreferencesEditor editor = preferences.Edit();
        editor.PutString(USERNAME_KEY, username);
        editor.Commit();
    }
    // פעולה המחזירה את סטטוס המשתמש המחובר מהקובץ
    public static bool GetRememberMe()
    {
        return preferences.GetBoolean(REMEMBER_ME_KEY, false);
    }
    // שינוי סטטוס המשתמש המחובר בקובץ
    public static void SetRememberMe(bool status)
    {
        // משתנה המקבל הרשאת עריכה לקובץ
        ISharedPreferencesEditor editor = preferences.Edit();
        editor.PutBoolean(REMEMBER_ME_KEY, status);
        editor.Commit();
    }
    // איפוס נתוני המשתמש כאשר המשתמש מתנתק או יוצא מהאפליקציה ולא זכור במכשיר
    public static void ResetPreferences()
    {
        // משתנה המקבל הרשאת עריכה לקובץ
        ISharedPreferencesEditor editor = preferences.Edit();
        editor.PutString(USERNAME_KEY, "");
        editor.PutBoolean(REMEMBER_ME_KEY, false);
        editor.Commit();
    }
}
```

ImageManager

מחלקת העזר המטפלת בהמרת תמונות למחרוזות. המחלקה הינה סטטית כדי לבצע את הפעולות הנוגעות להמרת תמונה למחרוזת ולהפך ע"י שימוש בשם המחלקה בלבד במחלקות האפליקציה השונות. השימוש במחלקה הוא הכרחי כדי לשמור תמונה (תמונת מתכון או תמונת פרופיל) שהמשתמש העלה, במסד הנתונים בתור מחרוזת. לאחר שמשתמש מעלה תמונה, הפעולה מקבלת את Bitmap התמונה שהועלה וממירה אותה למערך ביטים. לאחר מכן ממירה את מערך הביטים למחרוזת ומחזירה את המחרוזת. כאשר דפי האפליקציה מציגים תמונה ישנו שימוש בפעולה השנייה אשר מקבלת מחרוזת (מתכונת תמונת המשתמש או המתכון) וממירה אותה לBitmap של פקד תמונה ואז ניתן להציג את התמונה בפקד.

```
public class ImageManager
{
    // המרת תמונה למחרוזת ביטים לצורך איחסון ב-DB
    public static string BitmapToBase64(Bitmap bitmap)
    {
        string str = "";
        // שימוש באובייקט מטיפוס MemoryStream לאורך הסוגריים המסולסלות לאחר מכן הוא משוחרר
        using (MemoryStream stream = new MemoryStream())
        {
            // המרת Bitmap למחרוזת
            bitmap.Compress(Bitmap.CompressFormat.Png, 0, stream);
            byte[] bytes = stream.ToArray();
            str = Convert.ToBase64String(bytes);
        }
        return str;
    }
    // המרת מחרוזת ביטים המייצגת תמונה לאובייקט תמונה לצורך תצוגה
    public static Bitmap Base64ToBitmap(string base64String)
    {
        // המרת מחרוזת לBitmap
        byte[] imageAsBytes = Base64.Decode(base64String, Base64Flags.Default);
        return BitmapFactory.DecodeByteArray(imageAsBytes, 0, imageAsBytes.Length);
    }
}
```

מחלקות נוספות

User

מחלקת משתמש המייצגת את טבלת המשתמש ששם שמורים פרטי כל המשתמשים.
המחלקה כוללת את התכונות הבאות: מחרוזת שם משתמש, מחרוזת סיסמא, מחרוזת שם פרטי, מחרוזת אימייל, מחרוזת תשובה על שאלת אבטחה, מחרוזת תמונת פרופיל, מחרוזת קודי מתכונים אהובים, מחרוזת קודי המתכונים של המשתמש, משתנה בוליאני המציין האם המשתמש מאפשר את שירות ההודעות ומשתנה בוליאני המציין האם המשתמש הוא גם מנהל באפליקציה.
בעת יצירת משתמש חדש באמצעות הפעולה הבונה ערך מחרוזות תמונת הפרופיל, קודי המתכונים האהובים וקודי המתכונים של המשתמש הוא מחרוזת ריקה; והערך הבוליאני של משתנה אישור שירות ההודעות הוא true.

```
[Table("User")]
public class User
{
    [PrimaryKey, Column("Username")]
    public string username { get; set; } // שם המשתמש
    [Column("password")]
    public string password { get; set; } // סיסמא
    [Column("name")]
    public string name { get; set; } // שם פרטי
    [Column("email")]
    public string email { get; set; } // אימייל המשתמש
    [Column("securityQuestion")]
    public string securityQuestion { get; set; } // תשובה על שאלת אבטחה
    [Column("userImage")]
    public string userImage { get; set; } // תמונת הפרופיל
    [Column("favoriteRecipesId")]
    public string favoriteRecipesId { get; set; } // מחרוזת קודי המתכונים האהובים
    [Column("myRecipesId")]
    public string myRecipesId { get; set; } // מחרוזת קודי המתכונים שהמשתמש יצר
    [Column("allowNotification")]
    public bool allowNotification { get; set; } // משתנה בוליאני שמציין אם המשתמש מעוניין בשירות ההודעות
    [Column("isAdmin")]
    public bool isAdmin { get; set; } // משתנה בוליאני שמציין אם המשתמש הוא מנהל
    public User()
    {

    }
    public User(string username, string password, string name, string email, string
    securityQuestion, bool isAdmin)
    {
        this.username = username;
        this.password = password;
        this.name = name;
        this.email = email;
        this.securityQuestion = securityQuestion;
        this.isAdmin = isAdmin;
        this.userImage = "";
        this.favoriteRecipesId = "";
        this.myRecipesId = "";
        this.allowNotification = true;
    }
}
```

UserAdapter

אדפטר המשתמשים היורש ממחלקת BaseAdapter. המחלקה כוללת שתי תכונות:
האקטיביטי הנוכחי (UserListActivity) ורשימת כל המשתמשים. המחלקה כוללת פעולה לעדכון הרשימה בעת הצורך וכן פעולות נוספות הנדרשות למימוש מחלקת העל.
פעולה רלוונטית היא GetView לשם הצגת פרטי המשתמשים ברשומה.
הפעולה מתרחשת כמספר האיברים שברשימה וכל פעם "מנפחת" את הLayout של האקטיביטי כך שהוא יכיל את כל הרשומות (userItem_layout) עם פרטי המשתמש הרלוונטיים. כל רשומה מכילה את שם המשתמש, הסיסמא, השם הפרטי והאימייל של משתמש מסוים.
כמו כן, ישנה בדיקה אם המשתמש הוא גם מנהל ובמידה וכן, צבע הView יהיה אדום במקום כתום.

```
public class UserAdapter : BaseAdapter<User>
{
    // האקטיביטי המשוייך לאדפטר
    private Activity activity;
    // רשימת המשתמשים של האדפטר
    private List<User> users;
    public UserAdapter(Activity activity, List<User> users)
    {
        this.activity = activity;
        this.users = users;
    }
    public void SetUsers(List<User> users)
    {
        this.users = users;
    }
    public List<User> GetUserList()
    {
        return users;
    }
    public override long GetItemId(int position)
    {
        return position;
    }
    public override int Count
    {
        get { return users.Count; }
    }
    public override User this[int position]
    {
        get { return users[position]; }
    }
    /* הפעולה יודעת לרוץ כמספר האיברים ברשימה
     כל פעם היא מקבלת את המיקום המתאים */
    public override View GetView(int position, View convertView, ViewGroup parent)
    {
        // activity ניפוח הרשומה בדף המשתמשים לפי
        LayoutInflater layoutInflater = activity.LayoutInflater;
        View view = layoutInflater.Inflate(Resource.Layout.userItem_layout, parent, false);
        // הפרטים המופיעים ב-view וקישורם
        TextView tvUsername = view.FindViewById<TextView>(Resource.Id.userItem_tvUsername);
        TextView tvPassword = view.FindViewById<TextView>(Resource.Id.userItem_tvPassword);
        TextView tvName = view.FindViewById<TextView>(Resource.Id.userItem_tvName);
        TextView tvEmail = view.FindViewById<TextView>(Resource.Id.userItem_tvEmail);
        // הכנסת הפרטים לפי פרטי המשתמש במיקום המתאים
        tvUsername.Text = users[position].username;
        tvPassword.Text = users[position].password;
        tvName.Text = users[position].name;
        tvEmail.Text = users[position].email;
        // צבע התא של מנהל יהיה בגוון אדום
        if (users[position].isAdmin)
        {
            view.SetBackgroundColor(Color.ParseColor("#fffd0d29"));
        }
        return view;
    }
}
```

Recipe

מחלקת מתכון המייצגת את טבלת המתכון ששם שמורים פרטי כל המתכונים.
המחלקה כוללת את התכונות הבאות: מחרוזת קוד מתכון, מחרוזת שם מתכון, מחרוזת שם המשתמש שיצר את המתכון, מחרוזת תמונת המתכון, מחרוזת הקטגוריות, מספר שלם של שעות ההכנה, מספר שלם של דקות ההכנה, מחרוזת תיאור המתכון, מחרוזת המרכיבים, מחרוזת הוראות ההכנה, מספר שלם של מספר הלייקים (מספר אנשים שאוהבים את המתכון) ותכונה מטיפוס DataTime השומרת את זמן העלאת המתכון.
בעת יצירת מתכון חדש באמצעות הפעולה הבונה ערך מספר הלייקים שווה לאפס וערך זמן העלאת המתכון הוא DataTime.Now.
במסד הנתונים המתכונים נשמרים בתור Salad, Soup, Meat או Pasrty היורשים מRecipe כך שתכונות מחלקת המתכון משותפות לכל האובייקטים הנשמרים בעת הכנת מתכון.

```
[Table("Recipe")]
public class Recipe
{
    [PrimaryKey, Column("RecipeId")]
    public string recipeId { get; set; } // קוד מתכון
    [Column("recipeName")]
    public string recipeName { get; set; } // שם מתכון
    [Column("recipeUsername")]
    public string recipeUsername { get; set; } // שם משתמש יוצר המתכון
    [Column("recipeImage")]
    public string recipeImage { get; set; } // תמונת המתכון
    [Column("category")]
    public string category { get; set; } // קטגוריות
    [Column("hours")]
    public int hours { get; set; } // שעות הכנה
    [Column("minutes")]
    public int minutes { get; set; } // דקות הכנה
    [Column("description")]
    public string description { get; set; } // תיאור המתכון
    [Column("ingredients")]
    public string ingredients { get; set; } // מרכיבים
    [Column("instructions")]
    public string instructions { get; set; } // הוראות הכנה
    [Column("likeCount")]
    public int likeCount { get; set; } // מספר לייקים
    [Column("creationTime")]
    public DateTime creationTime { get; set; } // זמן העלאת המתכון
    public Recipe()
    {

    }
    public Recipe(string recipeId, string recipeName, string recipeUsername, string recipeImage,
    string category, int hours, int minutes, string description, string ingredients, string
    instructions)
    {
        this.recipeId = recipeId;
        this.recipeName = recipeName;
        this.recipeUsername = recipeUsername;
        this.recipeImage = recipeImage;
        this.category = category;
        this.hours = hours;
        this.minutes = minutes;
        this.description = description;
        this.ingredients = ingredients;
        this.instructions = instructions;
        this.likeCount = 0;
        this.creationTime = DateTime.Now;
    }
}
```

RecipeAdapter

אדפטר המתכונים היורש ממחלקת BaseAdapter. המחלקה כוללת שתי תכונות:
האקטיביטי הנוכחי (HomepageActivity, FavoriteRecipesActivity או MyRecipesActivity) ורשימת המתכונים בהתאם לאקטיביטי. המחלקה כוללת פעולה לעדכון הרשימה בעת הצורך וכן פעולות נוספות הנדרשות למימוש מחלקת העל.
פעולה רלוונטית היא GetView לשם הצגת פרטי המתכונים ברשומה.
הפעולה מתרחשת כמספר האיברים שברשימה וכל פעם "מנפחת" את הLayout של האקטיביטי הנוכחי כך שהוא יכיל את כל הרשומות (recipeItem_layout) עם פרטי המתכון הרלוונטיים. כל רשומה מכילה את שם המתכון, הקטגוריות, סוג המתכון והתמונה של מתכון מסוים.
הפעולה בודקת מאיזה טיפוס המתכון (Salad, Soup, Meat או Pasrty) בכדי לקבוע את סוג המתכון וכן ממירה את תכונת מחרוזת תמונת המתכון לBitmap של פקד תמונה באמצעות מחלקת העזר ImageManager.

```
public class RecipeAdapter : BaseAdapter<Recipe>
{
    // האקטיביטי המשוייך לאדפטר
    private Activity activity;
    // רשימת המתכונים של האדפטר
    private List<Recipe> recipes;
    public RecipeAdapter(Activity activity, List<Recipe> recipes)
    {
        this.activity = activity;
        this.recipes = recipes;
    }
    public void SetRecipes(List<Recipe> recipes)
    {
        this.recipes = recipes;
    }
    public List<Recipe> GetRecipeList()
    {
        return recipes;
    }
    public override long GetItemId(int position)
    {
        return position;
    }
    public override int Count
    {
        get { return recipes.Count; }
    }
    public override Recipe this[int position]
    {
        get { return recipes[position]; }
    }
    /* הפעולה יודעת לרוץ כמספר האיברים ברשימה
     כל פעם היא מקבלת את המיקום המתאים */
    public override View GetView(int position, View convertView, ViewGroup parent)
    {
        // activity-ניפוח הרשומה בדף המתאים לפי ה
    LayoutInflater layoutInflater = activity.LayoutInflater; // Same adapter in three activities
        View view = layoutInflater.Inflate(Resource.Layout.recipeItem_layout, parent, false);
        // הפרטים המופיעים ב-view וקישורם
        TextView tvRecipeName =view.FindViewById<TextView>(Resource.Id.recipeItem_tvRecipeName);
        TextView tvRecipeType =view.FindViewById<TextView>(Resource.Id.recipeItem_tvRecipeType);
        TextView tvCategory = view.FindViewById<TextView>(Resource.Id.recipeItem_tvCategory);
        ImageView ivImage = view.FindViewById<ImageView>(Resource.Id.recipeItem_ivRecipeImage);
        // הכנסת הפרטים לפי פרטי המתכון במיקום המתאים
        tvRecipeType.Text = "Type: ";
        if (recipes[position] is Salad)
        {
            tvRecipeType.Text += "Salad";
        }
        if (recipes[position] is Soup)
        {
            tvRecipeType.Text += "Soup";
        }
        if (recipes[position] is Meat)
        {
            tvRecipeType.Text += "Meat";
        }
        if (recipes[position] is Pastry)
        {
            tvRecipeType.Text += "Pastry";
        }
        tvRecipeName.Text = "Name: " + recipes[position].recipeName;
        tvCategory.Text = "Categories: " + recipes[position].category;
        Bitmap imageBitmap = ImageManager.Base64ToBitmap(recipes[position].recipeImage);
        ivImage.SetImageBitmap(imageBitmap);

        return view;
    }
}
```

Salad

מחלקת סלט המייצגת את טבלת הסלט ששם שמורים פרטי כל הסלטים. מחלקת סלט יורשת ממחלקת מתכון. המחלקה כוללת את התכונות הבאות: מחרוזת המייצגת האם מדובר בסלט ירקות או פירות, מחרוזת המייצגת את סוג הסלט, מחרוזת המייצגת האם הסלט מכיל בשר ומחרוזת המייצגת האם הסלט מכיל טיבול.

```
[Table("Salad")]
public class Salad : Recipe
{
    [Column("isFruit")]
    public string isFruit { get; set; } // סלט ירקות או פירות
    [Column("saladType")]
    public string saladType { get; set; } // סוג סלט
    [Column("isMeat")]
    public string isMeat { get; set; } // האם מכיל בשר
    [Column("isFlavoring")]
    public string isFlavoring { get; set; } // האם מכיל טיבול
    public Salad()
    {

    }
    public Salad(string isFruit, string saladType, string isMeat, string isFlavoring, string
    recipeId, string recipeName, string recipeUsername, string recipeImage, string category, int
    hours, int minutes, string description, string ingredients, string instructions) :
    base(recipeId, recipeName, recipeUsername, recipeImage, category, hours, minutes,
    description, ingredients, instructions)
    {
        this.isFruit = isFruit;
        this.saladType = saladType;
        this.isMeat = isMeat;
        this.isFlavoring = isFlavoring;
    }
}
```

Soup

מחלקת מרק המייצגת את טבלת המרק ששם שמורים פרטי כל המרקים. מחלקת מרק יורשת ממחלקת מתכון. המחלקה כוללת את התכונות הבאות: מחרוזת המייצגת את סוג המרק, מחרוזת המייצגת בישול ארוך או קצר, מספר שלם המייצג את זמן הבישול ומספר שלם המייצג את טמפרטורת הבישול.

```
[Table("Soup")]
public class Soup : Recipe
{
    [Column("soupType")]
    public string soupType { get; set; } // סוג מרק
    [Column("duration")]
    public string duration { get; set; } // בישול ארוך או קצר
    [Column("boilingTime")]
    public int boilingTime { get; set; } // זמן בישול
    [Column("temperature")]
    public int boilingTemperature { get; set; } // טמפרטורת בישול
    public Soup()
    {

    }
    public Soup(string soupType, string duration, int boilingTime, int boilingTemperature,
    string recipeId, string recipeName, string recipeUsername, string recipeImage, string
    category, int hours, int minutes, string description, string ingredients, string
    instructions) : base(recipeId, recipeName, recipeUsername, recipeImage, category, hours,
    minutes, description, ingredients, instructions)
    {
        this.soupType = soupType;
        this.duration = duration;
        this.boilingTime = boilingTime;
        this.boilingTemperature = boilingTemperature;
    }
}
```

Meat

מחלקת בשר המייצגת את טבלת הבשר ששם שמורים פרטי כל מנות הבשר. מחלקת בשר יורשת ממחלקת מתכון. המחלקה כוללת את התכונות הבאות: מחרוזת המייצגת את סוג הבשר, מחרוזת המייצגת את אופן ההכנה, מחרוזת המייצגת את רמת הצלייה ומספר שלם המייצג את זמן הבישול.

```
[Table("Meat")]
public class Meat : Recipe
{
    [Column("meatType")]
    public string meatType { get; set; } // סוג בשר
    [Column("method")]
    public string method { get; set; } // אופן הכנה
    [Column("doneness")]
    public string doneness { get; set; } // רמת צלייה
    [Column("cookingTime")]
    public int cookingTime { get; set; } // זמן בישול בדקות
    public Meat()
    {

    }
    public Meat(string meatType, string method, string doneness, int cookingTime, string
    recipeId, string recipeName, string recipeUsername, string recipeImage, string category, int
    hours, int minutes, string description, string ingredients, string instructions) :
    base(recipeId, recipeName, recipeUsername, recipeImage, category, hours, minutes,
    description, ingredients, instructions)
    {
        this.meatType = meatType;
        this.method = method;
        this.doneness = doneness;
        this.cookingTime = cookingTime;
    }
}
```

Pastry

מחלקת מאפה המייצגת את טבלת המאפה ששם שמורים פרטי כל המאפים. מחלקת מאפה יורשת ממחלקת מתכון. המחלקה כוללת את התכונות הבאות: מחרוזת המייצגת האם המאפה מכיל גלוטן, מחרוזת המייצגת את סוג הכשרות, מספר שלם המייצג את זמן האפייה ומספר שלם המייצג את טמפרטורת האפייה.

```
[Table("Pastry")]
public class Pastry : Recipe
{
    [Column("isGluten")]
    public string isGluten { get; set; } // האם מכיל גלוטן
    [Column("kosher")]
    public string kosher { get; set; } // סוג כשרות
    [Column("bakingTime")]
    public int bakingTime { get; set; } // זמן אפייה בדקות
    [Column("bakingTemperature")]
    public int bakingTemperature { get; set; } // טמפרטורת אפייה
    public Pastry()
    {

    }
    public Pastry(string isGluten, string kosher, int bakingTime, int bakingTemperature, string
    recipeId, string recipeName, string recipeUsername, string recipeImage, string category, int
    hours, int minutes, string description, string ingredients, string instructions) :
    base(recipeId, recipeName, recipeUsername, recipeImage, category, hours, minutes,
    description, ingredients, instructions)
    {
        this.isGluten = isGluten;
        this.kosher = kosher;
        this.bakingTime = bakingTime;
        this.bakingTemperature = bakingTemperature;
    }
}
```

סיכום אישי / רפלקציה

ראשית, נהניתי מאוד לעבוד על הפרויקט ולמדתי רבות על אופן העבודה ב-Xamarin Android.
בתחילת העבודה השקעתי בעיצוב הפקדים ותוך כדי עבדתי על דפי הפרויקט והשארתי את הטיפול בService ומחלקות הפרויקט להמשך העבודה. בנוסף לכך, השקעתי גם ביצירת דרך אשר תאפשר את שמירת נתוני המשתמש כאשר הוא מחובר לאפליקציה, וזאת הצלחתי להשיג באמצעות הממשק ISharedPreferences ומחלקת העזר המטפלת בכך. ההשקעה בחלקים אלו בפרויקט הייתה רלוונטית והכרחית ולדעתי עיצבה את אבני היסוד של האפליקציה.
אולם במהלך העבודה נתקלתי בכמה קשיים שעיכבו את תהלך העבודה וביניהם: מציאת מחלקות אשר יורשות ממחלקת מתכון ועיצוב האפליקציה בהתאם לכך, הכנת מסד הנתונים בהתאם למחלקות הנ"ל וכן מציאת רעיון לסרוויס אשר יופעל באפליקציה ויהווה מרכיב הכרחי בו.
בסופו של דבר, הגעתי להחלטה שהמחלקות היורשות מRecipe יהוו כסוג המתכון ולכל אחת מהמחלקות הנ"ל תהיה טבלה במסד הנתונים שם הם ישמרו לא כמתכון אלא כסלט, מרק, מנת בשר או מאפה; וכאשר יהיה צורך בשליפת כל המתכונים, תבוצע שליפה מכל טבלה והאובייקטים לאחר מכן יומרו כלפי מעלה לאובייקטים מטיפוס מתכון.
כדי לאפשר להבחין בין סוגי המתכון החלטתי ליצור דפי טופס למילוי באמצעות DialogFragment.
כמו כן, נתקלתי גם בשילוב הסרוויס באפליקציה, אך לאחר בדיקה מעמיקה הבנתי שניתן להראות הודעות שונות באמצעות הסרוויס והסרוויס יהווה כForegroundService לשם הצגת Notification.
העבודה על האפליקציה הייתה מאתגרת בחלקים מסוימים ובעיקר בנושאים שנתקלתי בהם לאורך העבודה, אך לאחר בדיקת הנושאים וחקירתם הצלחתי להתגבר על הקשיים השונים ולמצוא פתרון לאותן הבעיות.
לאורך הכנת האפליקציה למדתי רבות על הLifecycle של האקטיביטי, הברודכסט והסרוויס; על תפעול תקין ויעיל של מסד הנתונים, ביצוע פעולות שונות הקשורות למסד הנתונים וכן ביצוע שליפה של אובייקטים והשמתם ברשימה לשם סינון הרשימה בהתאם; וכיצד ליצור אפליקציה דינמית במערכת אנדרואיד כך שהמשתמש יכול לקיים אינטראקציה עם דפי האפליקציה ולהוסיף רשומות שונות.
יתרה מזאת, לאורך העבודה עסקתי בין היתר בחלק מהמושגים והמונחים שלמדנו לאורך שנות הלימודים במגמת מדעי המחשב, דבר שעזר ותרם רבות לבניית האפליקציה.
אילו הייתי מתחיל את העבודה היום הייתי קובע לעצמי תוכנית עבודה מוסדרת אשר תכלול את כל מרכיבי האפליקציה מתחילתה ועד סופה, כך שלאורך העבודה אני אטפל בכל הסוגיות השונות ולא אשאיר אותן לסוף העבודה. כמו כן, הייתי מעדיף להשקיע יותר בנושאים אשר לא היו מובנים כל כך בתחילת העבודה ולעבוד עליהם קודם כדי שהעבודה תהיה רצופה וללא כל בעיה.
לסיכום, למדתי רבות מהעבודה על הפרויקט, אשר שילבה את המושגים שלמדנו עליהם לאורך השנים וכן מושגים חדשים הקשורים ל-Xamarin Android, ואני מאוד מרוצה מהתוצר הסופי ובהחלט אקח את המידע הנ"ל והכלים הללו להמשך הדרך.

נספחים

מעקב מימוש דרישות הפרויקט לרמת 5 יחידות

	דרישה	**הסבר ופירוט יישום הדרישה בפרויקט**	מספר סעיף בדרישות	מספר סעיף בספר
1	תיאור הפרויקט בתחום אפליקציות לניידים במערכת הפעלה **Android**	תיאור כללי ומבנה פרויקט של האפליקציה.	1,2,7	5,6,7
	תיאור הפרויקט בדגש על הצורה האינטראקטיבית: המנוהלת ע"י ממשק גרפי למשתמש	יצירת מתכונים והעלאתם לדף הראשי של האפליקציה, חיפוש וסינון מתכונים, אפשרות לאהוב מתכון, שמירת מתכונים אהובים ומתכונים שהמשתמש העלה בדפים נפרדים.	3	7,9,12
2	בדיקה שהתוכנית רצה כנדרש	תמונות מסכים. מדריך משתמש - הסבר על מסכי האפליקציה.	3,6	13-24
3	פירוט בשימוש מתקדם בתכנות מונחה עצמים	מחלקות Salad,Soup,Meat,Pastry היורשות ממחלקת Recipe.	4	10,11
4	פירוט שימוש בתפריטים ובתיבות דו-שיח (AlertDialogs)	השתמשתי בתפריט מסוג ContextMenu בדף הבית, הכולל כפתורי ניווט ובמספר תיבות קופצות AlertDialog.	5	17,18,23,31
	פירוט שימוש ב- Activity	בפרויקט נעשה שימוש רחב ב-Activities.		38-63
	פירוט שימוש ב- Intent	בפרויקט נעשה שימוש בIntent למעבר בין הActivities השונים.		8
5	פירוט שימוש ויצירת Service	StatusNotificationService הצגת הודעות מתאימות במצבים שונים כגון: העלאת מתכון, עדכונו ומחיקתו, הוספת מתכון לרשימת המתכונים האהובים והתחברות והרשמה למערכת.	6	24,64,65

6	פירוט רכיבי ממשק אינטראקטיביים	רכיבים אלו אחראים על הצגת מידע למשתמש וקליטת קלט ממנו.	6	14-22 32-37
7	פירוט שימוש בלפחות אחד מתוך: **ContentProvider** **BroadcastReceiver**	BatteryBroadcastReceiver אשר מאזין ומקבל מידע על כמות הסוללה ומשמש לעדכון כמות הסוללה באפליקציה ומאפשר הצגת הודעה מתאימה אם כמות הסוללה נמוכה.	6	17,65
8	פירוט שימוש בבסיס נתונים או אחסון נתונים אחר	בסיס הנתונים (SQLite) אשר משמש לארגון ושמירת הנתונים המטופלים על – ידי האפליקציה. בתוכנה זו משמש בסיס הנתונים לאחסון נתוני המשתמשים והמתכונים. נעשה גם שימוש בממשק ISharedPreferences אשר מאפשר שמירה של פרטי המשתמש המחובר בקובץ פנימי בזיכרון המכשיר.	5	25-27 72-79
9	פירוט הנושאים מתקדמים	שימוש במצלמה להעלאת תמונת מתכון ותמונת פרופיל של משתמש.		18,20,21