### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

* Internet checker  
  בדיקה אלו אנשים מתעניינים בפוסטים (סטטוסים וכו') שהיוזר מפרסם בפרק זמן ספציפי.   
  התעניינות נבחנת ע"י בדיקה מי מגיב (תגובה/LIKE) לפוסטים.  
  הפיצ'ר נמצא בכרטיסיית Interest Check באפליקציה.
* פילטור חברים ומיילים ע"י פילטרים מקונפגים מראש.  
  ע"י יצירת פילטר מתקדם (קריטריונים: גיל, מין ושייכות לקבוצת חברים)   
  ניתן לראות רשימה מוקטנת של חברים שמקיימים תנאי הפילטר  
  וכן לראות רק שרשורי הודעות ששותף בהן חבר שמקיים תנאי פילטר.  
  את הפילטר מוסיפים בכרטיסיית filters באפלקיציה.  
  את רשימת החברים המצומצמת (והמורחבת) ניתן לראות בכרטיסיית friends באפליקציה.  
  את שרשורי ההודעות המצומצמת (והמורחבת) ניתן לראות בכרטיסיית messages באפליקציה.

### תבנית מס' 1 – Composite

* סיבת הבחירה / שימוש בתבנית:

במערכת ישנם פילטרים שבודקים קיום תנאי על קבוצת יוזרים.  
פילטר מיוחד שכזה הינו המתקדם שמורכב ממספר פילטרים שיש לקיים את כל התנאים של הפילטרים שבתוכו על היוזרים.  
קל לראות שההתייחסות לפילטר זה אינה שונה מההתייחסות לפילטרים שמוכלים בו,  
התנאי שלו יותר מורכב, אך כל מה שנרצה זה לבדוק את התנאי שלו על קבוצת יוזרים.

* אופן המימוש:

הוגדר ממשק IUsersFilter אשר מהווה הcomponent של התבנית. ממשק זה מגדיר את הפונקציה FilterUsers.  
מחלקת הComposite של התבנית הינה AdvancedFilter אשר ממשת את IUsersFilter ומכילה בתוכה רשימה של IUsersFilter.  
העלים הינם UsersAgeFilter , UsersFriendListsFilter ו- UsersGenderFilter אשר כולם יורשים מ- UsersGenderFilter שממש את IUsersFilter.

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### תבנית מס' 2 – Singelton

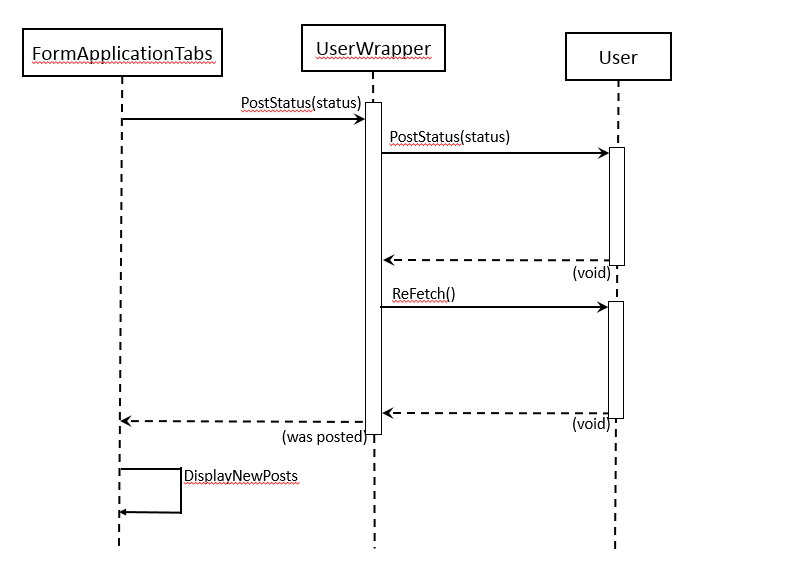
* סיבת הבחירה / שימוש בתבנית:

במערכת לאחר מסך הLOGIN יש יוזר אחד שמחובר. נרצה שכל הרכיבים יפנו באופן יזום לאינסטנס של היוזר הזה במערכת, שכן כל עדכון שיעשה ביוזר (כגון רשימת פוסטים/ רשימת חברים) יצטרך להעשות על היוזר הזה.

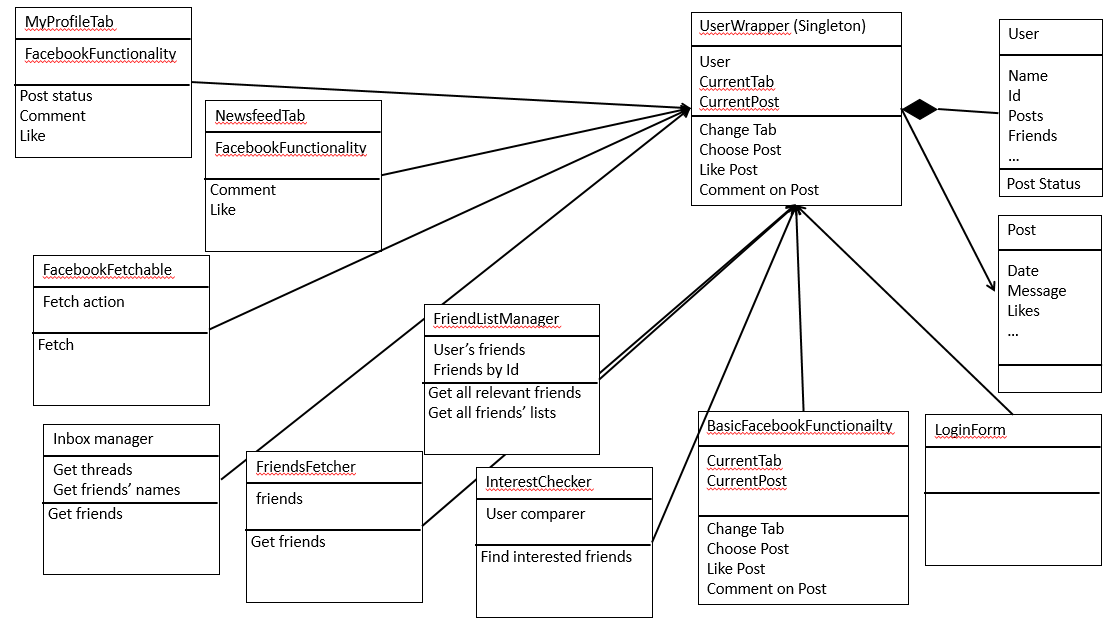
* אופן המימוש:

UserWrapper מממש את התבנית ע"י בנאי פרטי ו-Property סטטי Instance.  
הInstance מומש כך שתהיה נעילה של השדה הפרטי אך ורק אם הוא היה null כשהתרד ניגש לProperty.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Builder

* סיבת הבחירה / שימוש בתבנית:

במערכת הייתה בנייה סבוכה של הפילטרים השונים. העדפנו להכיל את הלוגיקה של הבנייה הזו בתוך מחלקה ייעודית, כך שקוד שחיצוני לה לא צריך להתעסק בפרטי היצירה של כל פיטר ופילטר ובלוגיקה השונה של כל בנאי, אלא רק לשלוח את המשתנים שמגדירים את הפילטר המיועד.

* אופן המימוש:

במחלקה FilterBuilder, המתודה הסטטית CreateFilter מקבלת את המידע לגבי המרכיבים השונים של כל פילטר, קוראת לבנאים הרלוונטיים ומרכיבה את הפילטר הקומפוזיטי מתבנית מס' 1.

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

עבודה אסינכרונית:

הקריאות לפונקציות הבסיס של פייסבוק (Like, Commit, Post) נעשות באופן אסינכרוני, על מנת לא לעצור את ממשק המשתמש בזמן שהקריאה נשלחת לפייסבוק והערכים השמורים במערכת מתעדכנים. את הקריאות מימשנו בעזרת עטיפת הפונקציות הקודמות ב-Task שנוצר ומופעל בלי להמתין לו, מכיוון שאינו מחזיר ערך.

ניתן למצוא את הקריאות האסינכרוניות במחלקה BasicFacebookFunctionality בפונקציות commentAsync, likeAsync ובמחלקה FormApplicationTab בפונקציה buttonSetStatus\_Click.