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

* הצעות שידוכים – הפיצ'ר מציג למשתמש את כל החברים של החברים שלו אשר סטטוס מערכת היחסים שלהם הוא רווק או לא מוגדר להם סטטוס כלשהו.

המשתמש יכול לבחור האם לסנן את הרשימה על פי גיל, מגדר ועיר מגורים והרשימה תתעדכן בהתאם עם אחוזי התאמה ושמות האנשים הרלוונטים.

ניתן ללחוץ על השם המבוקש ולראות את תמונתו, פרטיו האישיים וכן לכתוב לו הודעה על הקיר.

* הצעות לתמונת פרופיל חדשה – הפיצ'ר מציג למשתמש את התמונות מכל האלבומים שלו בפייסבוק כך שהן ממויינות לפי הכמות הכי גדולה של לייקים ותגובות.

המשתמש יכול למיין את התמונות כרצונו לפי כמות לייקים או לפי כמות תגובות.

לאחר בחירת תמונה כלשהי מהרשימה, תופיע תצוגה מקדימה של התמונה והפרטים שיש לתמונה כגון: תאריך פרסום, כמות הלייקים והתגובות שיש לתמונה וכו'.

בנוסף, המשתמש יכול לבחור אפקט לתמונה (שחור לבן/חום/שקוף/רגיל).

לאחר מכן ניתן להעלות את התמונה כתמונת פרופיל חדשה.

### תבנית מס' 1 – Singleton

* סיבת הבחירה / שימוש בתבנית:

בחרנו לממש Pattern זה במחלקה "FormLoginSingleton" מאחר ורצינו למנוע מצב שיתאפשר ליצור יותר ממופע אחד של המחלקה FormLoginSingleton.

מטרת מחלקה זו היא ביצוע תהליך הLogin לFacebook עבור המשתמש באפליקציה, והחזקת הפרטים אודות ההתחברות (AccessToken וכו') ואם יהיה יותר ממופע אחד של המחלקה זה יהיה לא תקין כי יכולות להיווצר סתירות בין המופעים השונים של הטפסים ולכן לא נאפשר שיהיה יותר ממופע אחד.

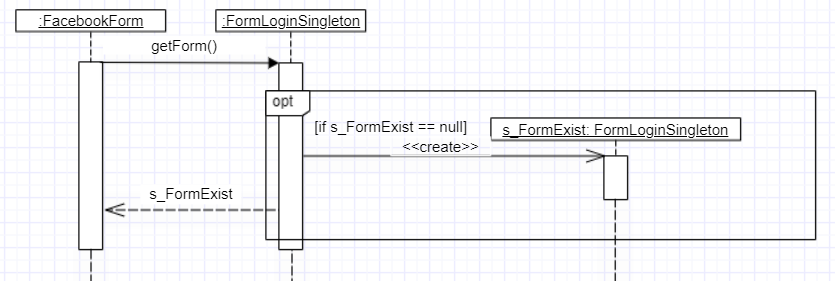
* אופן המימוש:

ניתן למצוא את המימוש במחלקה FormLoginSingleton.

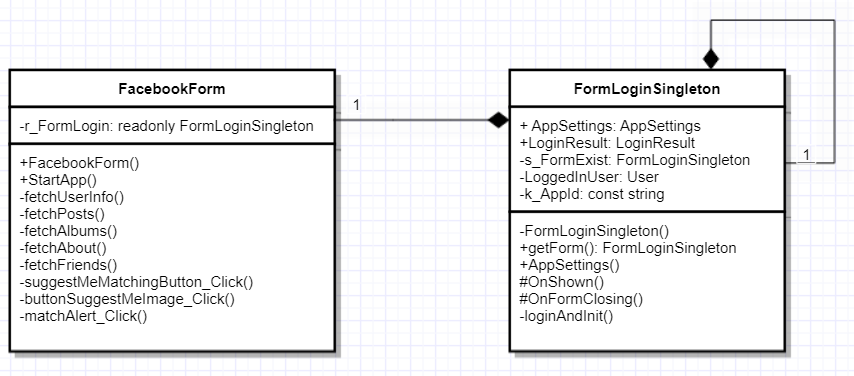
תחילה הגדרנו את הבנאי של המחלקה FormLoginSingleton כ-Private ולכן הגדרנו את המחלקה כ-seald (על מנת למנוע ירושה עתידית).

לאחר מכן הוספנו static member למחלקה בשם s\_FormExist אשר מאותחל בnull והגדרנו עבורו properties אשר בודק אם s\_FormExist==null אז הוא יוצר מופע חדש של FormLoginSingleton ועושה השמה ל s\_FormExistולאחר מכן מחזיר את המופע. אחרת, יש כבר מופע קיים למחלקה FormLoginSingleton ולכן יוחזר המופע הקיים.

מימוש זה lazy ואינו .thread safe

* Sequence Diagram
* Class Diagram

FormLoginSingleton מהווה את הSingleton, FacebookForm מהווה את הClient.



### תבנית מס' 2 – Facade

* סיבת הבחירה / שימוש בתבנית:

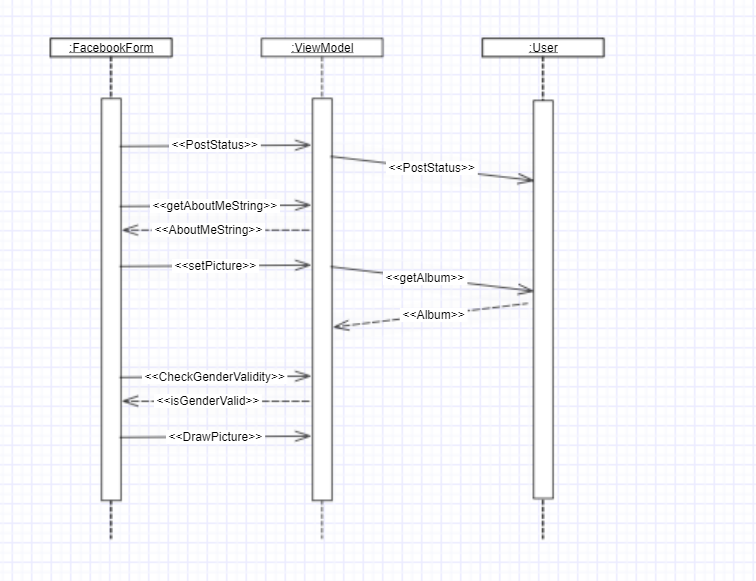
בחרנו ב- Pattern זה על מנת "להסתיר" מימושים של תתי פעולות אשר עלולות לפגוע בקריאות הקוד. בין תתי פעולות אלה –יצירת StringBuilder ארוך, וכן ביצוע פעולות כמו פרסום סטטוס. כאשר הפעולות הנ"ל היו ממומשות ב- client עצמו, היה קשה יותר להתמצא בקוד ולהבין מהן הפעולות העיקריות. לאחר "הסתרת" פעולות אלה, כלומר השמתן תחת קורת גג המאחדת אותן, הקוד יותר מסודר וברור וכן יותר מודולרי ותחזוקתי.

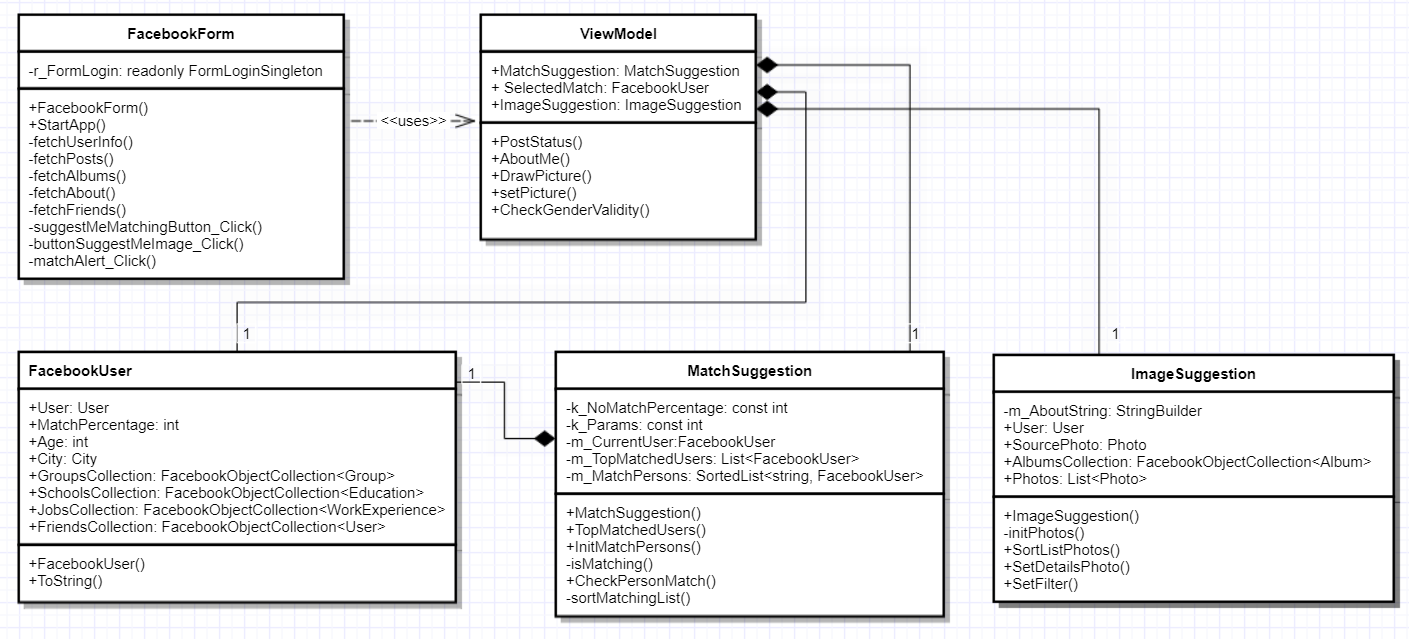
* אופן המימוש:

ניתן למצוא את המימוש במחלקה ViewModel שהיא מהווה את הFacade עצמו ומוגדרת כמחלקה סטטית.

הfacade מהווה חוצץ בין FacebookForm (המשתמש) לבין ImageSuggestion FacebookUser ,MatchSuggestion (הממשקים)

* Sequence Diagram



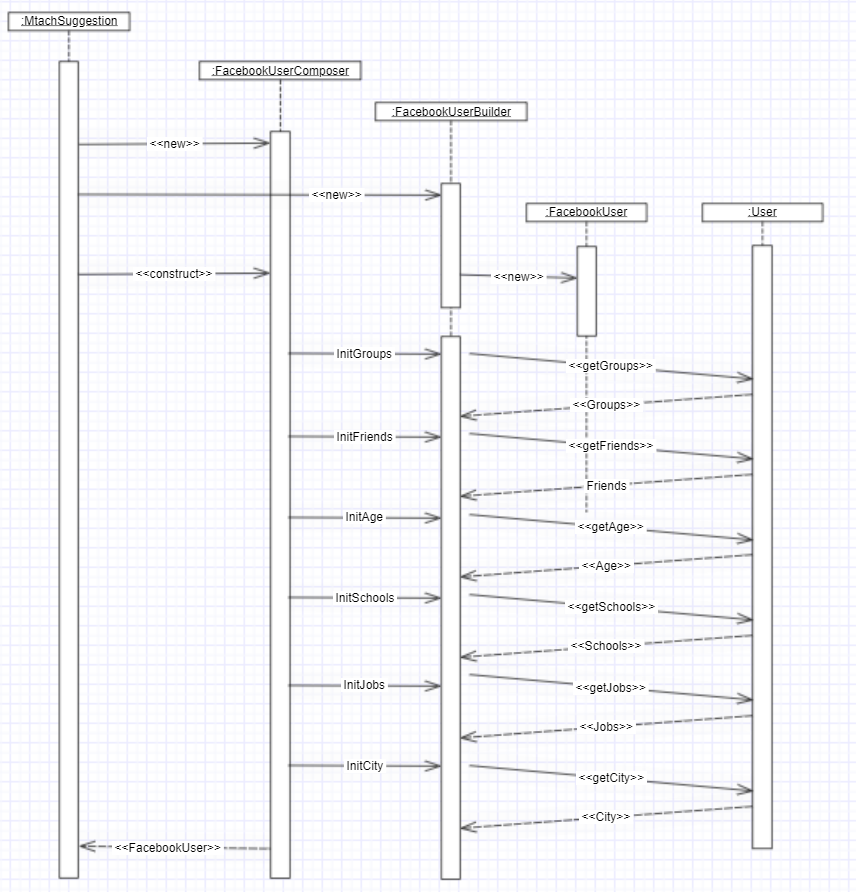
* Class Diagram

### תבנית מס' 3 – Builder

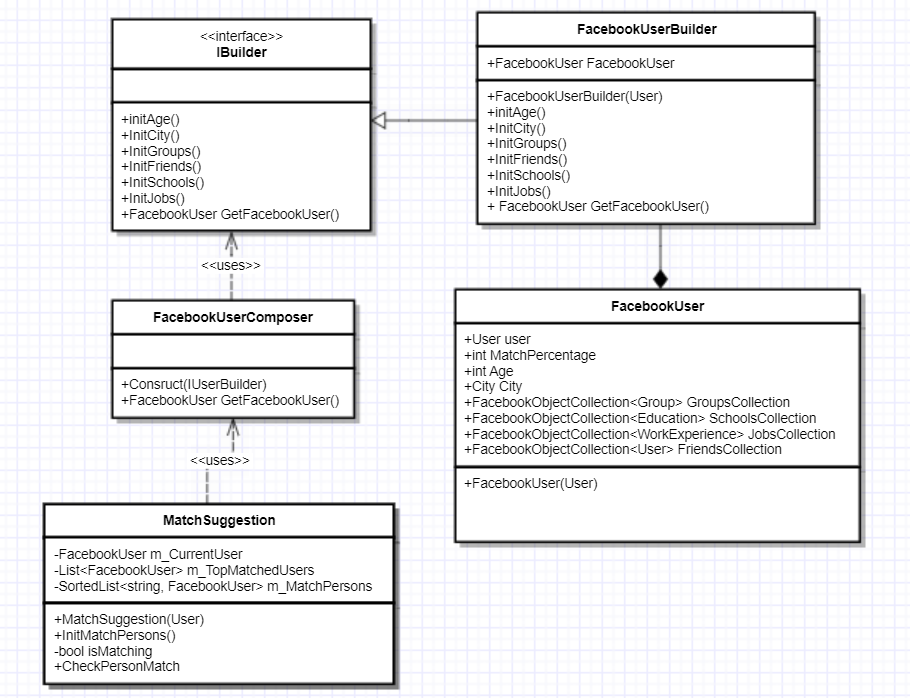
* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בתבנית זו משום ש FacebookUser הינו אובייקט מורכב כאשר יוצרים אובייקט מסוג זה ישנם הרבה מרכיבים שצריך לקלוט ולאתחל בעת השימוש בו, וסדר קבלת הקלט לא בהכרח יתקבל בסדר השימוש בקלט. לכן נרצה להפריד את שלב יצירת האובייקט משלב קליטת הנתונים.

* אופן המימוש:
  + FacebookUser- זו המחלקה של האובייקט המורכב, מכילה את כל התכונות הנדרשות לאובייקט.
  + FacebookUserComposer הוא ה- Composer שלנו ("המלצר")- זו המחלקה שהמשתמש חשוף אליה, ושמאפשרת את יצירת האובייקט בclient.
  + IBuilder- interface שמסייע בבניית האובייקט המורכב ואיתחול התכונות שלו, FacebookUserBuilder יורש ממנו.
  + FacebookUserBuilder הוא ה- Builder עצמו ("הטבח")- מחלקה זו יורשת מIBuilder. במחלקה זו יש את כל המימושים של המתודות לאיתחול התכונות של הfacebookUser.
  + matchSuggestion- זהו הclient שמשתמש באובייקט.
* Sequence Diagram



* Class Diagram



**עבודה אסינכרונית:**

עשינו שימוש בעבודה אסינכרונית במחלקה FacebookForm- בעליית העמוד הראשי של האפליקציה, תוך כדי טעינת פרופיל הפייסבוק.

**עבודה עם Data Binding:**

* ביצענו שימוש ב- Two Way Data Binding בTab הראשי של האפליקציה בלחיצה על Posts ניתן לעיין בפוסטים שברשימה וגם לערוך אותם.
* ביצענו שימוש ב- One Way Data Binding בTab הראשי של האפליקציה בלחיצה על Albums ניתן לעיין באלבומים שברשימה כך שמצד ימין מוצגת תמונת האלבום בהתאמה.