### 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 – Strategy

* סיבת הבחירה / שימוש בתבנית:

כפי שלמדנו בכיתה פטרן זה בא לפתור לנו את הבעיה בה יש מנגון ואנו מזהים נקודות " "Injection Point הניתנות להחלפה כך שבעתיד נוכל לשנות את התוצאה ואת ההתנהגות של המנגון מבלי לשנות את הקוד ומבלי לשכפל את הקוד.

השתמשנו ב Strategy על מנת לפתור את הבעיה ההנדסית בה נתקלנו בעת כתיבת התוכנית.

בפיצ'ר השני – "Image Suggestion" המשתמש יכול לבחור אפקט לתמונה כך שצבעה יהיה בשחור-לבן/חום/שקוף. נאלצנו לשכפל קוד עבור כל אפקט שהרי הלוגיקה זהה ויש רק נקודה אחת שונה בין שלושת המתודות והיא "colorMatrix".

בעקבות כך, היה צורך בשימוש ב Strategyשמאפשר לנו להחליף בהתאמה את מטריצת הצבעים בעת בחירת המשתמש מבלי לשכפל קוד ומבלי לשנות את הקוד כך שאם בעתיד נרצה להוסיף עוד אפקט נוכל לעשות זאת בקלות ובכך שיפרנו את הקריאות של הקוד, את התחזוקתיות של הקוד ומנענו שכפול קוד.

**אופן המימוש:**

מכך שמדובר על interface של מתודה אחת בלבד וב3 מחלקות שממשות את ה interface, והמתודה "פשוטה" מימשנו ע"י lambda expression ובכך המתודות לא "יפריעו" ברחבי המחלקה ובכך נקל על הקורא.

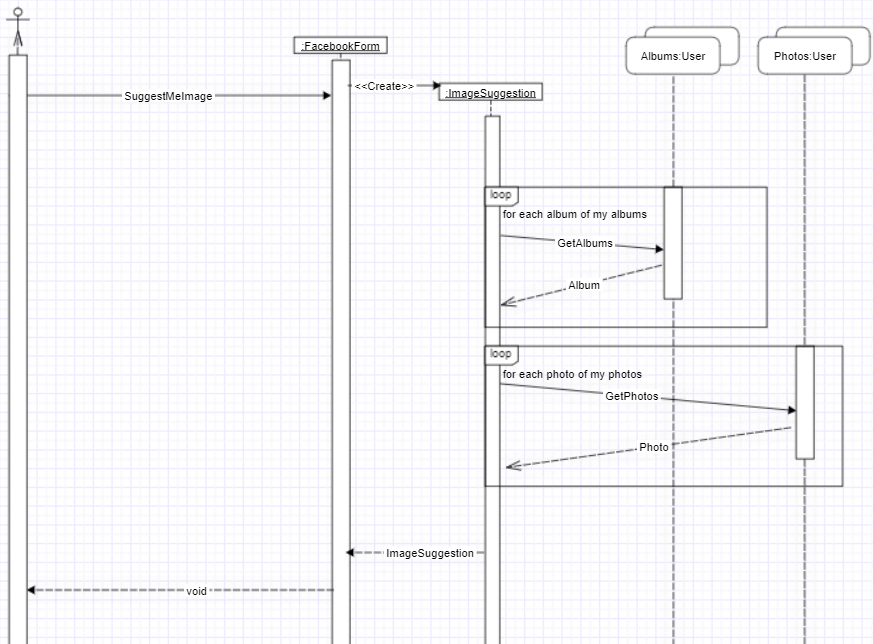
ImageSuggestion – במחלקה זו הגדרנו Func< ColorMatrix > delegate

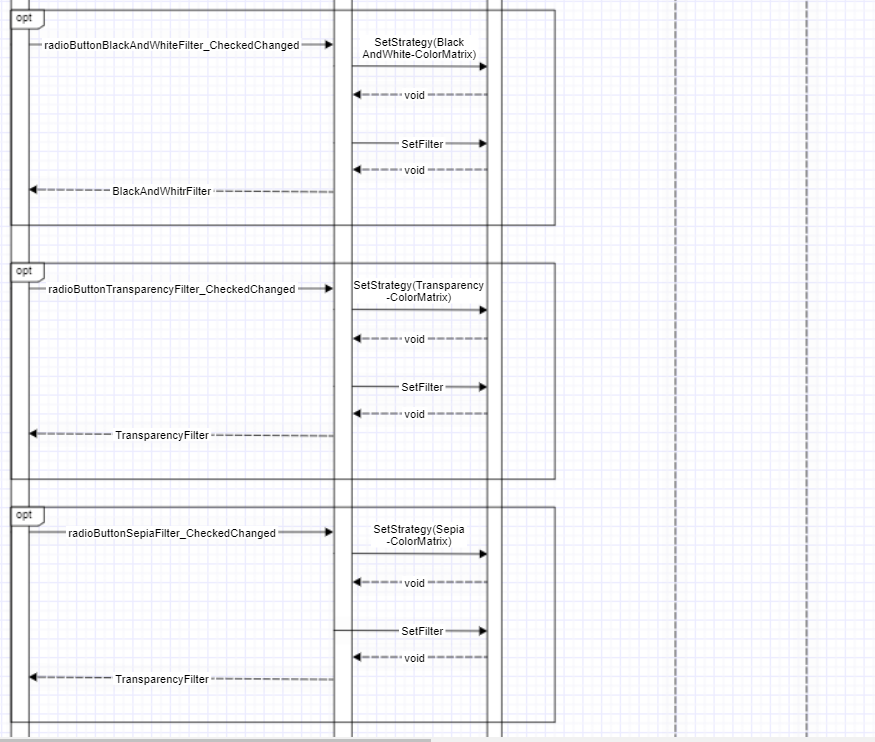
בנוסף, במחלקה זו נמצאת המתודה SetFilter אשר בתוכה הstrategy method מבצע invoke למתודה אותה רוצים להפעיל. (האפקט אותו מעוניינים לשים על התמונה)

FacebookForm – במחלקה זו ישנן 3 מתודות - radioButtonBlackAndWhiteFilter\_CheckedChanged, radioButtonTransparencyFilter\_CheckedChanged, radioButtonSepiaFilter\_CheckedChanged

אשר כל אחת מהן מהווה "אסטרטגיה" , עושה השמה ל Func< ColorMatrixשב ImageSuggestionוקוראת למתודה setFilter.

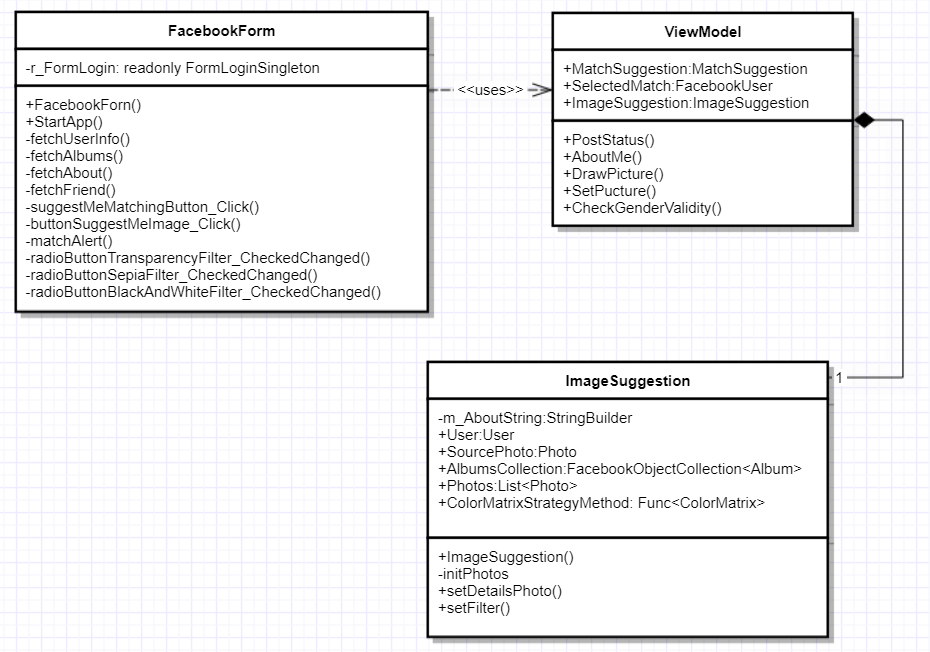
* Sequence Diagram





ניתן לראות כי לאחר הכניסה לפיצ'ר (ולאחר ייבוא התמונות) המשתמש בוחר באחד משלושת האפקטים, בFacebookForm מתבצעת השמה לdelegate של ה"אסטרטגיה" כלומר, של מטריצת הצבעים בהתאם לצבע שנבחר ולאחר מכן תופעל המתודה ששמה את האפקט ע"י invoke ל delegateבמתודה SetFilter

* 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 – Iterator

### סיבת הבחירה / שימוש בתבנית:

כפי שנלמד בכיתה- design parttern Iterator נדרש על מנת לאפשר לנו סריקת איברים בקלוקציית נתונים מבלי לחשוף את מבנה הנתונים בו השתמשנו.

במטלה בוצע שימוש בתבנית זו על מנת לספק שירות סריקה על קולקציה של matchPersons מבלי לחשוף את מבנה הנתונים. ראשית, שימוש בpattern זה משפר את תחזוקתיות המערת. כלומר אם בעתיד נרצה להחליף את מבנה הנתונים של matchPerson לא נצטרך לעשות שינויים בקליינט.

שנית, הpattern מאפשר גמישות – במידה ונרצה לחפש התאמה על פי קטוגריות שונות, יהיה קל יותר לעשות זאת.

* אופן המימוש:

**מימוש ה- Iterator נמצא במחלקה MatchPersonIterator**. במחלקה זו ממומש BuilderDelegate המקבל User ו FacebookUserComposer. הBuildereDelegate מאפשר בחירה של matchPerson עפ"י קטגוריה מסויימת.  
המחלקה מממשת מתודת GetEnumerator הסורקת את רשימת החברים אשר מועברת אליו. המתודה משתמשת ב BuilderDelegate הנ"ל, ומחזיר IEnumerator באמצעות מנגון הyield return.

השימוש במנגון הyield return מסמן לקומפיילר שעליו לממש מחלקה חדשה של IEnumerator בה ימומש המתודות reset, dispose, moveNext, Current.

בנוסף, במחלקה ישנה מתודה initMatches המאפשרת אתחול של רשימת החברים שמתאימים למשתמש בלי לחשוף את מבנה הנתונים.  
**ה- client הוא המחלקה MatchSuggestions**, אשר מכילה אובייקט מסוג MatchIterator ומשתמשת בשירותי הסריקה שלו באמצעות המתודה initMatches

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

* 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' 3 – [שם התבנית]

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

* 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