### 

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

* **Trivia Feature:**

בפיצר זה אנו בוחרים חבר באקראי מבין חבריו של המשתמש ובודקים את הידע שלו על חברו ע"י מתן 4 שאלות. המשתמש צריך לענות על השאלות ובהתאם לתשובותיו יקבל ניקוד.

המשתמש יכול להגריל שאלון על חבר אחר ע"י לחיצה על כפתור  
 "Randomize another friend".

* **Create Common Groups Feature:**

בפיצר זה אנו מאפשרים למשתמש לראות את רשימת החברים שלו שגרים באותה העיר וכן לראות את רשימת החברים שלו שנולדו באותו התאריך.בנוסף אנו פותחים קבוצה לכל רשימת החברים המשותפים לפי אותה העיר ולפי אותו תאריך וכן ניתן לעלות פוסטים לקבוצות ולראות את הפוסטים שנמצאים בקבוצות אלה.

### תבנית מס' 1 – [Observer]

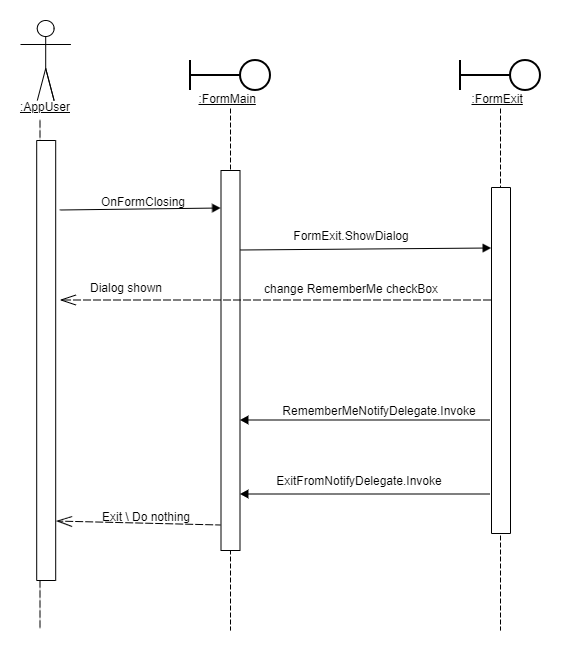
* סיבת הבחירה / שימוש בתבנית:

יצרנו טופס חדש, FormExit, אשר מאשר\לא מאשר את היציאה מהטופס הראשי, FormMain. כאשר יצרנו את הטופס החדש רצינו שהוא יודיע שהשתנה לו ה-State מבלי שהוא יכיר קונקרטית את אלה שהוא צריך להודיע להם, על מנת שיהיה ריוזבילי. לכן השתמשנו בפטרן Observer אשר עוזר לנו בבעיה זו. FormExit מודיע ל FormMainעל החלטת המשתמש אם לאשר את היציאה מהאפליקציה או לא. המודיע בתבנית שלנו הוא FormExit והמקשיב הוא לFormMain. תבנית זו עוזרת בפתרון הבעיה בכך שבעזרת delegates הטופס החדש , FormExit,מפעיל בעזרת invoke מתודה ב FormMainמבלי שיכיר אותו. בכך FormExit מודיע למאזינים מבלי להכיר אותם קונקרטית.

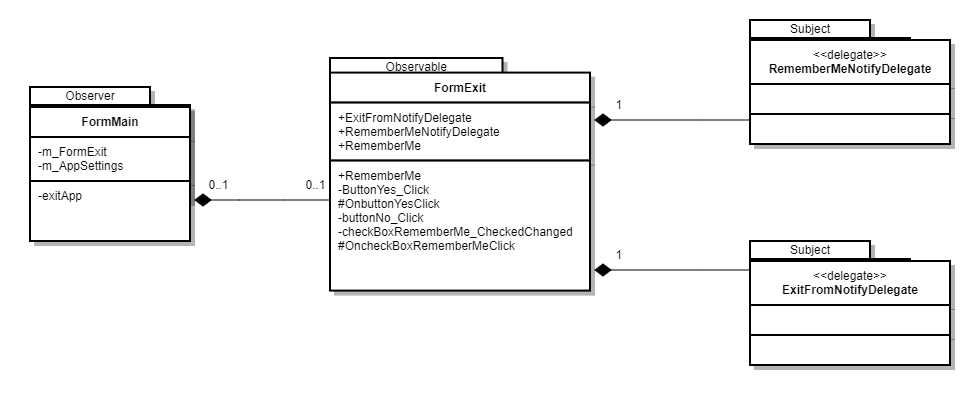
* אופן המימוש:

יצרנו טופס חדש, FormExit , שבו יש 2 משתני מחלקה delegates בעלי הרשאת גישה של event, ExitFormNotifyDelegate ו-RemmemberMeNotifyDelegate.   
בFormMain- הוספנו לExitFormNotifyDelegate- את המתודה ExitApp אשר שומרת נתונים לפני היציאה.   
ל RemmemberMeNotifyDelegateהוספנו מתודה למבדא אקספרשן, שכאשר המשתמש לוחץ על הremmember me בטופסFormExit נתון זה נשמר בAppSetting שבFormMain -. כלומר ה-Observer שזה ה- FormMainנרשם כמאזין לDelegates וברגע שהמשתמש בוחר לסגור את האפליקציה מופעלות המתודות בעזרת invoke בהתאם לבחירת המשתמש.

* Sequence Diagram



* Class Diagram



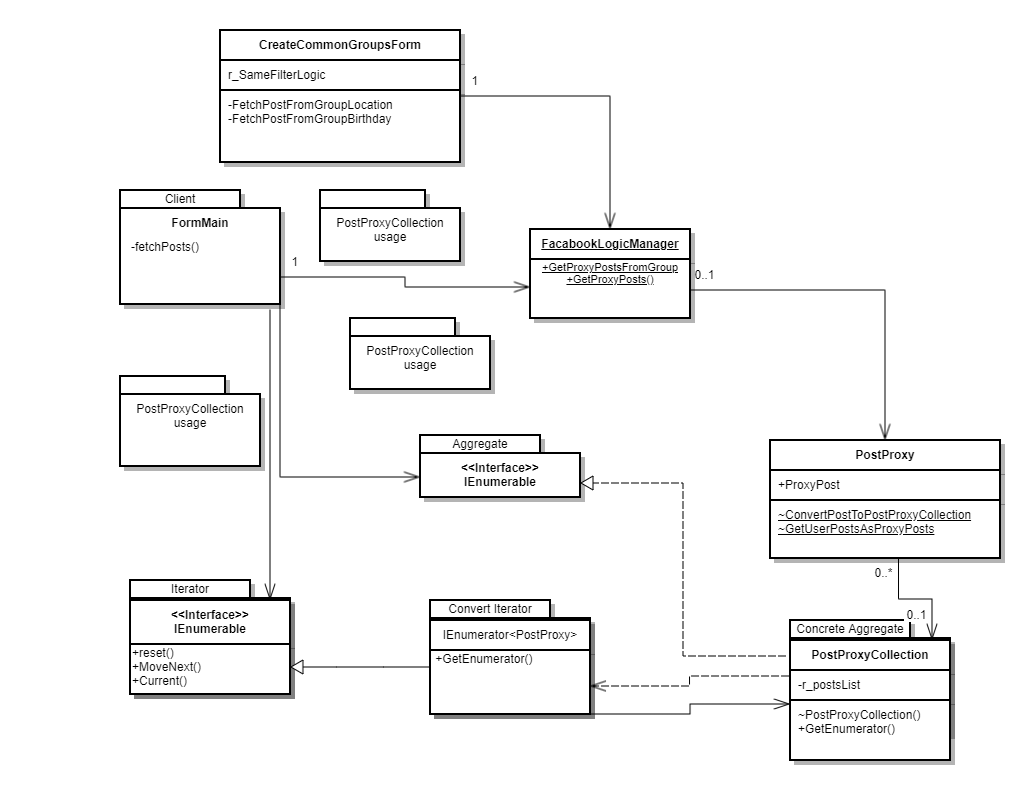
### תבנית מס' 2 – [Iterator]

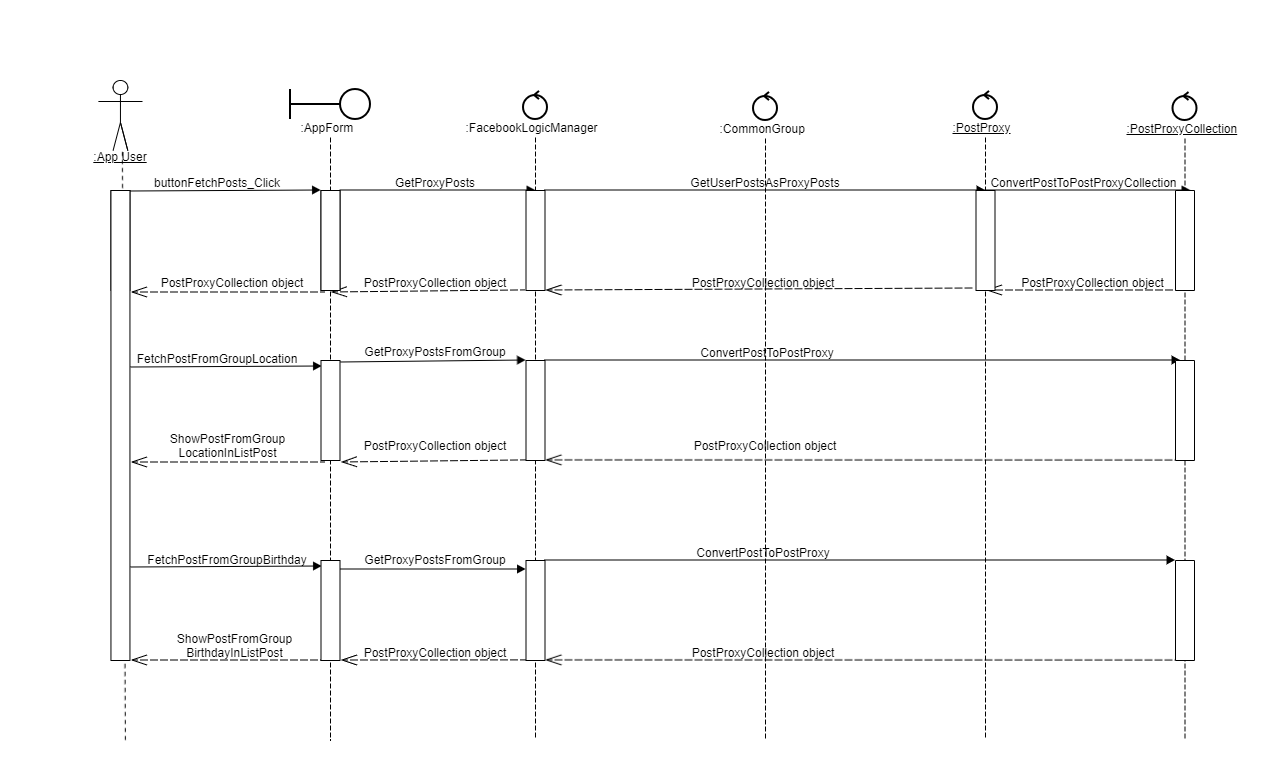
* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בתבנית Iterator מכיוון שבמערכת יש מעבר על הפוסטים של המשתמש כך שהם מוחזקים בקולקציה של פוסטים (בפועל PostProxy). באמצעות Iterator אנו מאפשרים לעבור על הפוסטים מבלי להתייחס לסוג הקולקציה בה הם מוחזקים, ואם בעתיד נרצה לשנות את מבנה הנתונים של הקולקציה אז נוכל לעבור עליו בצורה דומה באמצעות המתודות של ה-Iterator ונבצע את השינויים רק ב-Iterator. כלומר אנחנו נרצה רכיב המאפשר סריקה על קולקציה מבלי שיכירו מבחוץ את מבנה הנתונים שהקולקציה מוחזקת ויחשוף לClient רק את המתודות הרלוונטיות לו, דבר זה שומר על encapsulation ופילטור מידע.

* אופן המימוש:

יצרנו מחלקה מקוננת, PostProxyCollection, בתוך PostProxy אשר מקבלת קולקציה של פוסטים. יש לה מבנה נתונים List של פוסטים אשר מאותחלת בקונסטרקטור ומתודה GetEnumerator אשר מחזירה מחזירה איטרטור בעזרת yield return על כל אחד מהפוסטים ברשימה לאחר הפיכת Post ל- PostProxy. השימוש באיטרטור בא לידי ביטוי ב FormMainוcreateCommonGroupsForm.

* Class Diagram 
* Sequence Diagram



### תבנית מס' 3 – [Strategy]

* סיבת הבחירה / שימוש בתבנית:

זיהינו כי יצירת הקבוצות הוא לפי בחירת המיקום ותאריך הלידה, והבנו שלא חייב ליצור קבוצות לפי סינון זה בלבד, לדוגמה אפשר לבחור ליצור קבוצת כל האנשים עם אותו שם. לכן החלטנו להשתמש בStrategy שמבטיחה שאם נרצה להוסיף/לשנות את סינון יצירת הקבוצות בעתיד , נוכל לעשות את זה באופן תחזוקתי יעיל כך שניצור מחלקה חדשה המתארת את הסוג החדש מבלי לפגוע בשאר המחלקות שמשתמשות ביצירת הקבוצה ומבלי לשנות אותן.

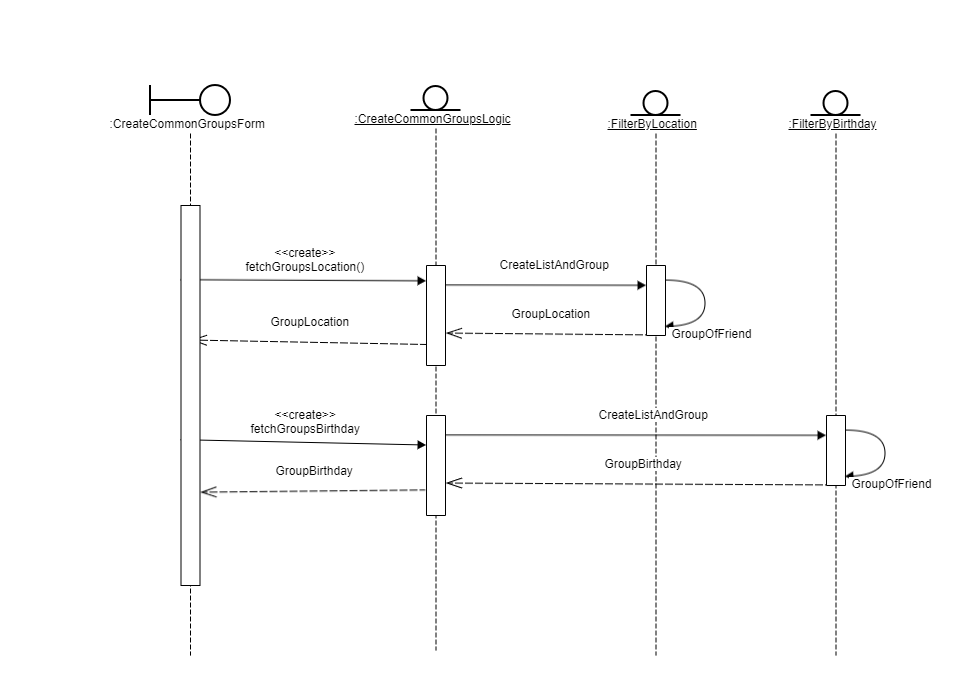
* אופן המימוש:

איתרנו את הinjection points ויצרנו אינטרפייס IFilterGroupStrategy אשר מכיר את המתודות המבטאות את קטגוריית הסינון שנבחר ליצירת הקבוצה.

המחלקה CreateCommonGroupsLogic מחזיקה את המופעים של המחלקה המממשת את הממשק. לכל קטגוריית סינון יצרנו מחלקה המממשת את הממשק והן FilterByLocation ו FilterByBirthday.

-נמצאות ב Facebooklogic/CreateCommonGroupsLogic

* Sequence Diagram



* Class Diagram

