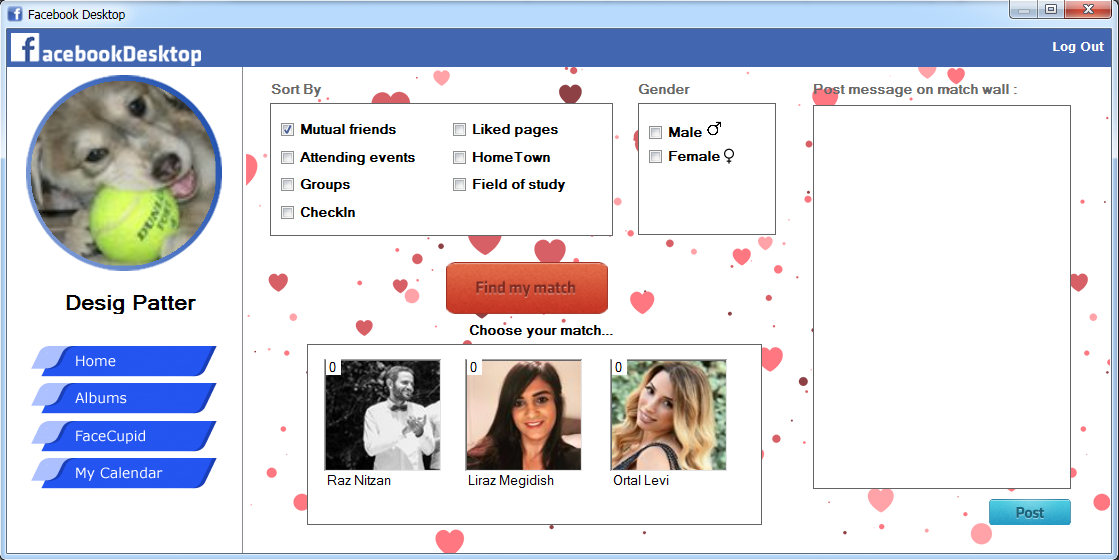
### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

* **Facebook Cupid**

הפיצ'ר מהווה פלטפורמת היכרויות בין חברים בפייסבוק.

המשתמש יכול לבחור עבור אילו פרמטרים הFacebook Cupid יחשב עבורו את ההתאמה האידיאלית (כגון: העדפה מינית, חברים משותפים, אירועים, קבוצות, צ'ק אין, לייקים, עיר מגורים וכו').

תמונת מסך Facebook Cupid:

* **My Calendar**

הפיצ'ר מספק למשתמש יומן חודשי לניהול אירועי הFacebook של המשתמש, ימי הולדת של החברים שלו וניהול אירועים מחיי היום יום של המשתמש.

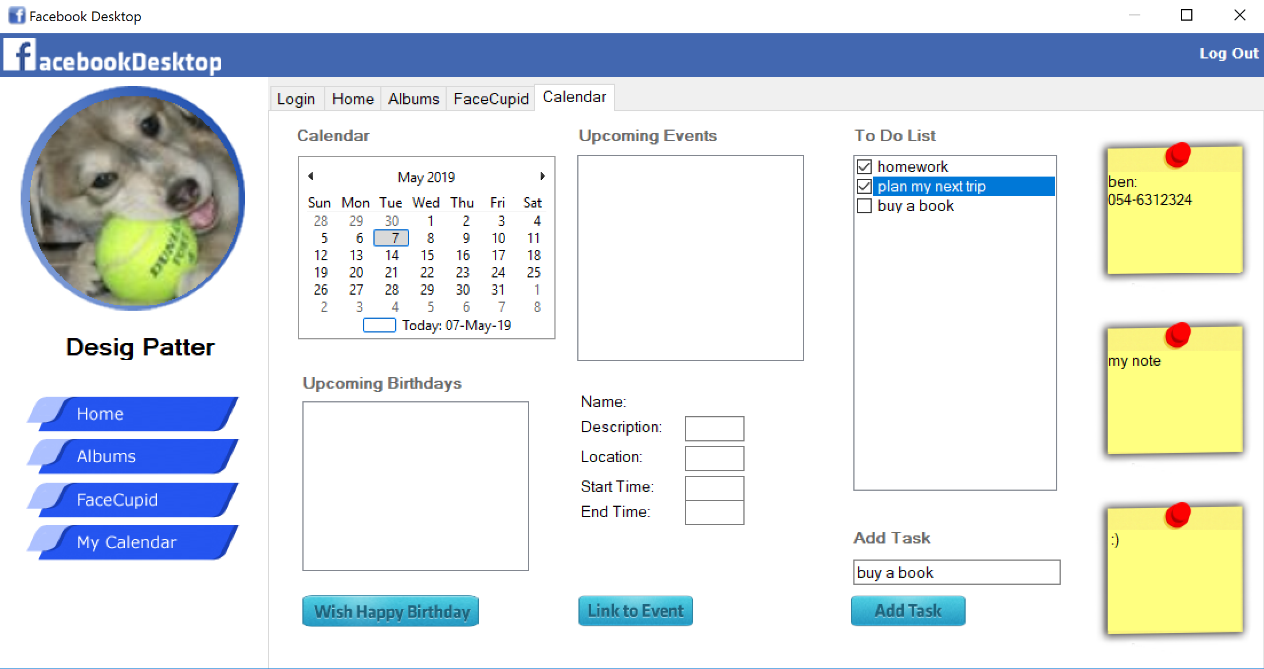
ימי הולדת – ניתן לשלוח ברכות יום הולדת מותאמת אישית למשתמש שיש לו יומולדת בחודש הנוכחי.

אירועים –ניתן לראות את כל האירועים של המשתמש בחודש הנוכחי, כמו כן יש כפתור המקשר לאירוע באתר של פייסבוק.

TDL –ניהול רשימת מטלות חודשית, ניתן להוסיף מטלות ולסמן מטלות שבוצעו.

פתקים – ניתן לערוך 3 פתקים.

תמונת מסך My Calendar:



### תבנית מס' 1 – [Singelton]

* **סיבת הבחירה / שימוש בתבנית:**

בחרנו בתבנית SINGLETON עבור המחלקה FacebookDesktopLogic על מנת להגביל את יצירת מופע המחלקה אחד ויחיד. לא נרצה יותר ממופע אחד של המחלקה מכיוון שכל מופע אחר של המחלקה יהיה בעל State זהה (בזבוז של מקום בזיכרון), כמו כן 2 מופעים עם State שונה הוא מצב לא תקין במערכת.

המחלקה FacebookDesktopLogic הינה "המנוע" של האפליקציה היא משתמשת ברכיבים רבים ולכן נרצה שכל רכיב לוגי עתידי יוכל להשתמש במחלקה ללא צורך בהעברת רפרנס.

נרצה ששאר המחלקות שעושות שימוש במחלקה הסינגלטונית

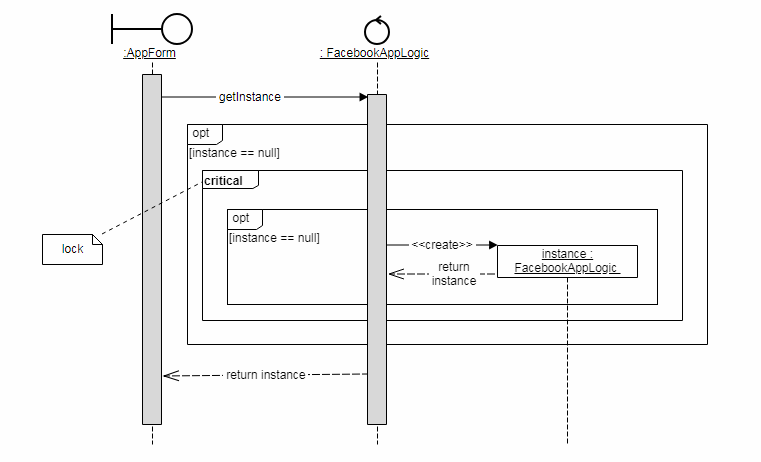
* **אופן המימוש:**

השחקנים בתבנית:

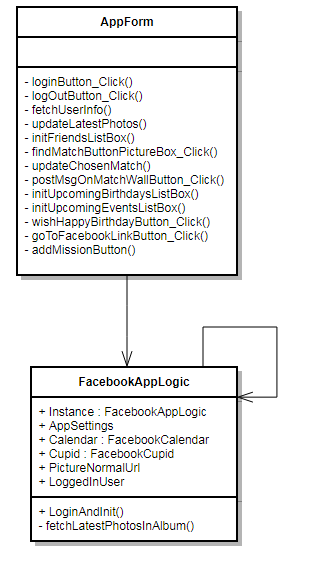
1. Client - FacebookDesktopForm
2. Singleton – FacebookDesktopLogic

בחרנו לממש את מחלקת הסינגלטון ע"י double cheak lock על מנת לצמצם את תקורת פעולת הנעילה בעת יצירת מופע האובייקט. הבדיקה הראשונה (לפני הנעילה) מתבצעת עלמנת למנוע נעילה "מיותר" בעתיד שכן נעילה הינה פעולה "יקרה", הבדיקה השניה (אחרי הנעילה) מתבצעת על מנת לוודר כי לא השתנה מצב המופע (במידה וקיים או לא) בפרק הזמן מהבדיקה הראשונה עד לנעילה.

* **Sequence Diagram**



* **Class Diagram**



### תבנית מס' 2 – [Facade]

* סיבת הבחירה / שימוש בתבנית:

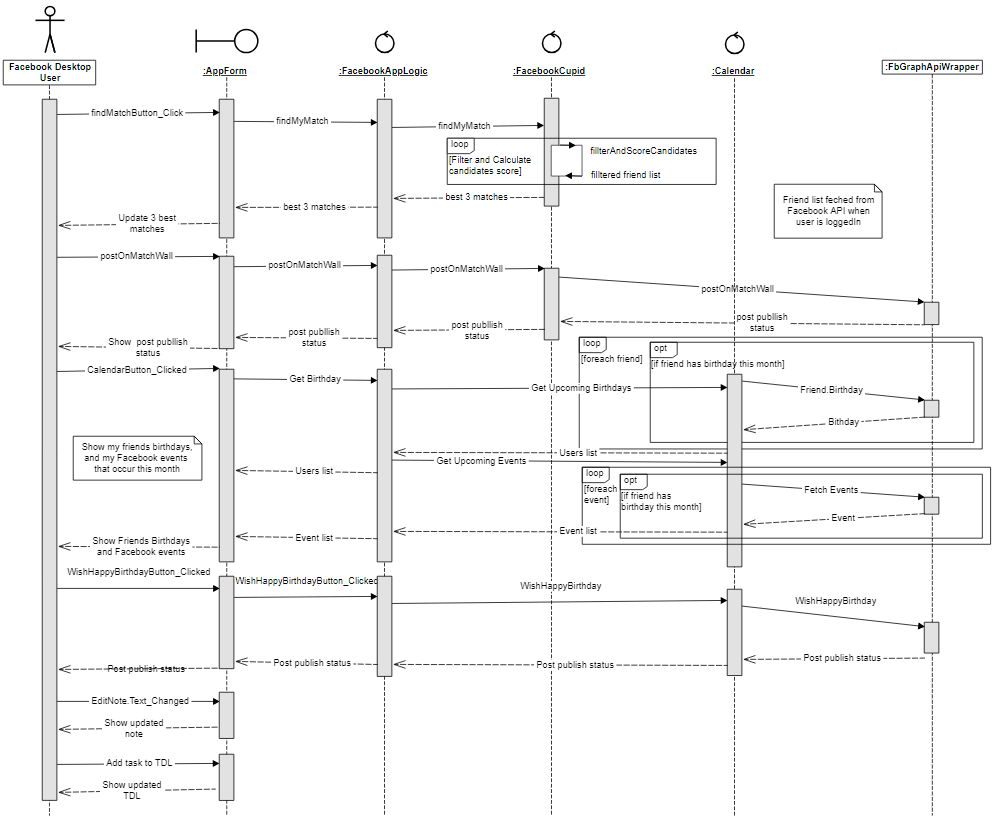
בחרנו לממש את התבנית Facade, על מנת להציג ממשק נקי ומסודר יותר ל Client (במקרה שלנו המחלקה AppForm) בכדי שיוכל לעבוד עם מחלקות הלוגיות של האפליקציה ביתר קלות (FacebookCupid, FacebookCalendar). מחלקת הFacade מהווה חציצה בין ממשק המשתמש appform לשאר המחלקות הלוגיות, גם פיצ'רים עתידיים שייצטרפו למערכת יוכלו להשתמש ב Facade (FacebookAppLogic) ולהינות מיתרונותיו.

חסר משהו

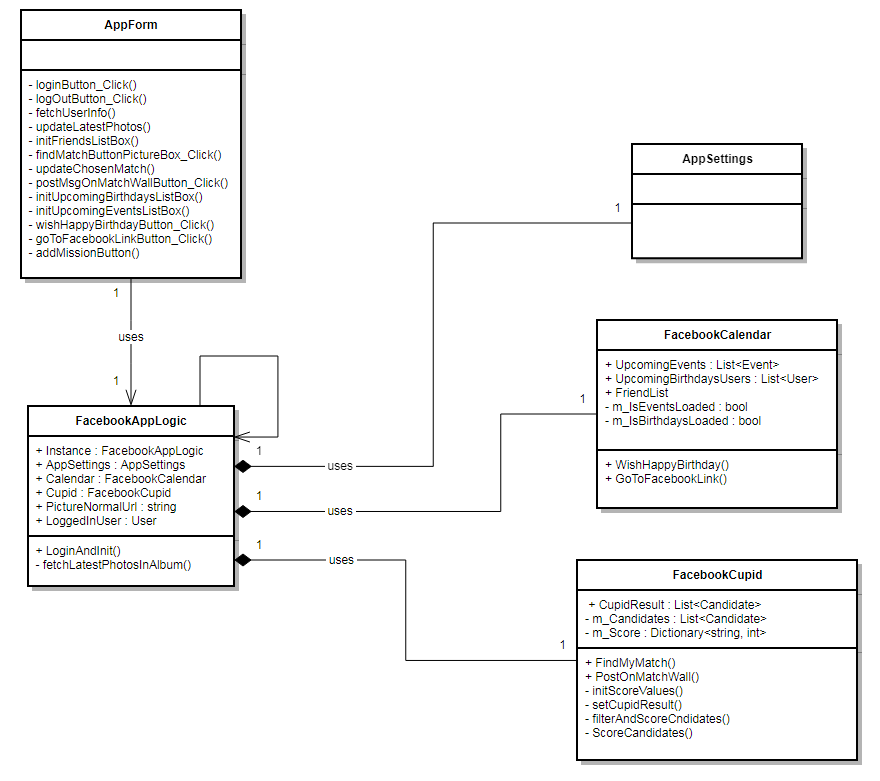
* אופן המימוש:
  + המחלקות המקבילות בתבנית:
    - Client - AppForm
    - Facade - FacebookAppLogic
    - Sub System A - FacebookCupid
    - Sub System B - FacebookCalendar

ה Facade (FacebookAppLogic) "מסתיר" את שני הפיצ'רים העיקריים במערכת (SubsystemA, subSystemB) וחושף ממשק נוח לשימוש עבור הclient (AppForm) וחושף עבורו רק מה שהוא צריך ובכך מאפשר שימוש בטוח ונקי

* **Sequence Diagram**



* **Class Diagram**



### תבנית מס' 3 – [Proxy]

* סיבת הבחירה / שימוש בתבנית:

בחרנו לממש תבנית Proxy מכיוון שרצינו לשדרג את הרכיב PictureBox (Legacy component), מכיוון ואין לנו את הקוד של המחלקה PictureBox החלטנו לממש רכיב נפרד (RoundPictureBoxProxy). הוספנו לProxy את הלוגיקה שמעגלת את התמונה תוך שהוא מתחזה לרכיב המקורי ובכך אנו יכולים להשתמש ברכיב באותו אופן שהשתמשנו ברכיב הקודם ללא צורך בשינויים בClient (AppForm). גם רכיבים עתידיים יוכלו להשתמש בProxy על מנת ליצור תמונה עגולה.

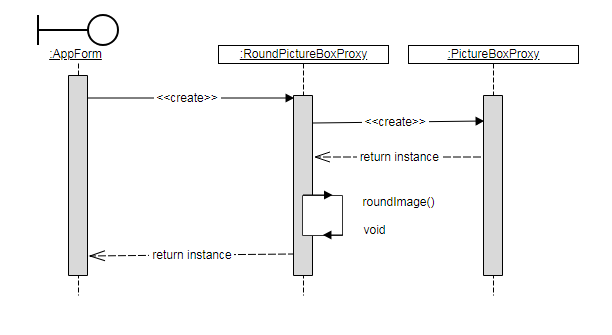
אופן המימוש:

המחלקות המקבילות בתבנית:

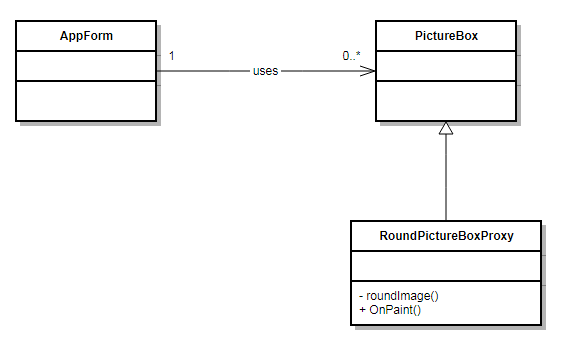
1. Client - FacebookDesktopForm
2. Real Service - Picture Box
3. Class Proxy - RoundPictureBoxProxy

מכיוון שהרכיב PictureBox אינו ממש Interface נאלצנו לממש Class Proxy, הרכיב החדש RoundPictureBoxProxy יורש מהרכיב המקורי pictureBox ומוסיף לו את לוגית עיגול התמונה בעת הדפסה (OnPaint). כעת בכל מקום בו נרצה תמונה עגולה באפליקציה נוכל להשתמש ב RoundPictureBoxProxyבאותו אופן שהשתמשנו ברכיב המקורי ולקבל תמונה עגולה.

* **Sequence Diagram**



* **Class Diagram**



### עבודה עם Data Binding :

נעשה שימוש ב Data Binding במחלקה AppForm תחת המתודה FetchEvents .

### עבודה אסינכרונית :

נעשה שימוש בתכנות אסינכרוני במתודה loadHomeTab שבמחלקה MainForm.

2 המתודות הבאות , מתבצעות ב2 threads נפרדים מה thread הראשי :

FetchFriends

FetchRecentPosts

בשתי המתודות הנ"ל מתבצעת גישה לנתונים אשר "מובאים" מ- FacebookWrapper, כל גישה לשרתים הינה פעולה "יקרה" אשר לוקחת זמן ארוך מהרגיל, בזמן זה האפליקציה "נתקעת" עד לסיום תהליך הבאת הנתונים.

לכן, על מנת להציג למשתמש נתונים אחרים בזמן הבאת הנתונים הנ"ל מFacebookWrapper , השתמשנו בthread נפרד לכל מתודה אשר מונע מהתכנית " להתקע" בזמן זה ולא ממאצת את המשתמש להמתין לסיום התהליכים "היקרים".