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

* **תיאור הפי'צר הראשון- Find My Match:**

הפיצ׳ר מאפשר למשתמש לפלטר לפי מין וטווח גילאים ולראות את חבריו הרווקים שעונים על הקריטריונים שהוזנו.( SelectPreferencesForm)

לאחר שהמשתמש ימלא את הטופס, יפתח חלון חדש (FindMyMatchForm)ובו תוצג רשימת התוצאות מסודרת ב-listBox .המשתמש יכול לבחור את אחד מן ההתאמות ויפתח מסך(MyMatchForm) שיכיל את פרטי האדם שיכול להיות הזיווג עבורו.

במסך זה ניתן לצפות בתמונות, לראות רשימת עמודים אהובים, תמונת פרופיל ואף לפרסם פוסט ולתייג את ההתאמה שנבחרה.

בנוסף המשתמש יכול לצפות בתמונת המשתמש שאהב אותו הכי הרבה(על פי כמות הלייקים) .

וכך לנסות למצוא התאמה.

**היכן הפיצר ממומש?**

ממשק המשתמש נמצא: **SelectPreferencesForm.**

הלוגיקה נמצאת: **FindMyMatchLogic**

**\*הערה:** המשתמש לא יכול לצפות בתמונת המשתמש שאהב אותו הכי הרבה **ולא ניתן לפרסם פוסטים בגלל הרשאות ה  API.**

* **תיאור הפי'צר השני- Post Statistics:**

הפיצ׳ר מאפשר למשתמש לקבל מידע אודות הפוסטים שלו.( PostRankForm)

המשתמש יכול לצפות בפוסט שקיבל הכי הרבה תגובות כמו כן המשתמש יכול לצפות בפוסטים שלו ממוינים בסדר עולה\יורד לבחירתו לפי כמות התגובות.

בעת לחיצה על הפוסט יפתח מסך(PostInformationForm) שמציג פרטים על הפוסט:

תוכן הפוסט, תאריך כתיבת הפוסט, כמות התגובות ואפשרות לשתף את הפוסט.

כמו כן המשתמש יכול לבחור לצפות ב2 גרפים. גרף אחד מייצג את כמות הפוסטים שהמשתמש העלה ביחס לשנה, גרף שני מייצג את כמות הפוסטים ביחס לחודש.

פיצ'ר זה נותן ערך מוסף למשתמש מפני שבשימוש בפיצ'ר זה המשתמש יכול לעקוב אחרי איזה תוכן גרר יותר תגובות ובכך להגדיל את קהל היעד על ידי פרסום פוסטים דומים.

**היכן הפיצר ממומש?**

ממשק המשתמש נמצא: **PostRankForm.cs.**

הלוגיקה נמצאת: **PostRankFormLogic.cs**

**\*הערה:** המשתמש לא יכול **לפרסם פוסטים בגלל הרשאות ה  API.  
כמו כן, ישנם פוסטים שלא ניתן לצפות בתוכן שלהם עקב הרשאות הAPI.**

### תבנית מס' 1 – Strategy

* סיבת הבחירה / שימוש בתבנית:

זוהי תבנית לפיה ניתן לבחור אלגוריתמים בזמן ריצה. באופן רשמי, התבנית מגדירה משפחה של אלגוריתמים, מבצעת אנקפסולציה של כל אחד מהם, והופכת אותם לניתנים להחלפה.

אנו בחרנו להשתמש ב-strategy על מנת להציג למשתמש האפליקציה את ההתאמות שפולטרו עבורו בשתי צורות שונות:

* בצורה מקוצרת ותמציתית הכוללת את שמו של החבר ואת גילו.
* בצורה נרחבת יותר הכוללת בנוסף לפרטים שציינו לעיל גם את המין של החבר וגם את המיקום שלו.

מפני שהפונקציות הממשות את שתי צורות ההצגה הנ"ל ממומשות בצורה דומה,

על מנת להימנע משכפול קוד החלטנו להחליף רק את המנגנון שאוסף ומשרשר את המידע.

אם בעתיד נרצה להוסיף עוד הצגות שונות להצגת נתוני ההתאמה בעזרת שימוש בתבנית זו נוכל לעשות זאת בצורה קלה, נוחה, וללא כפל קוד.

* אופן המימוש:

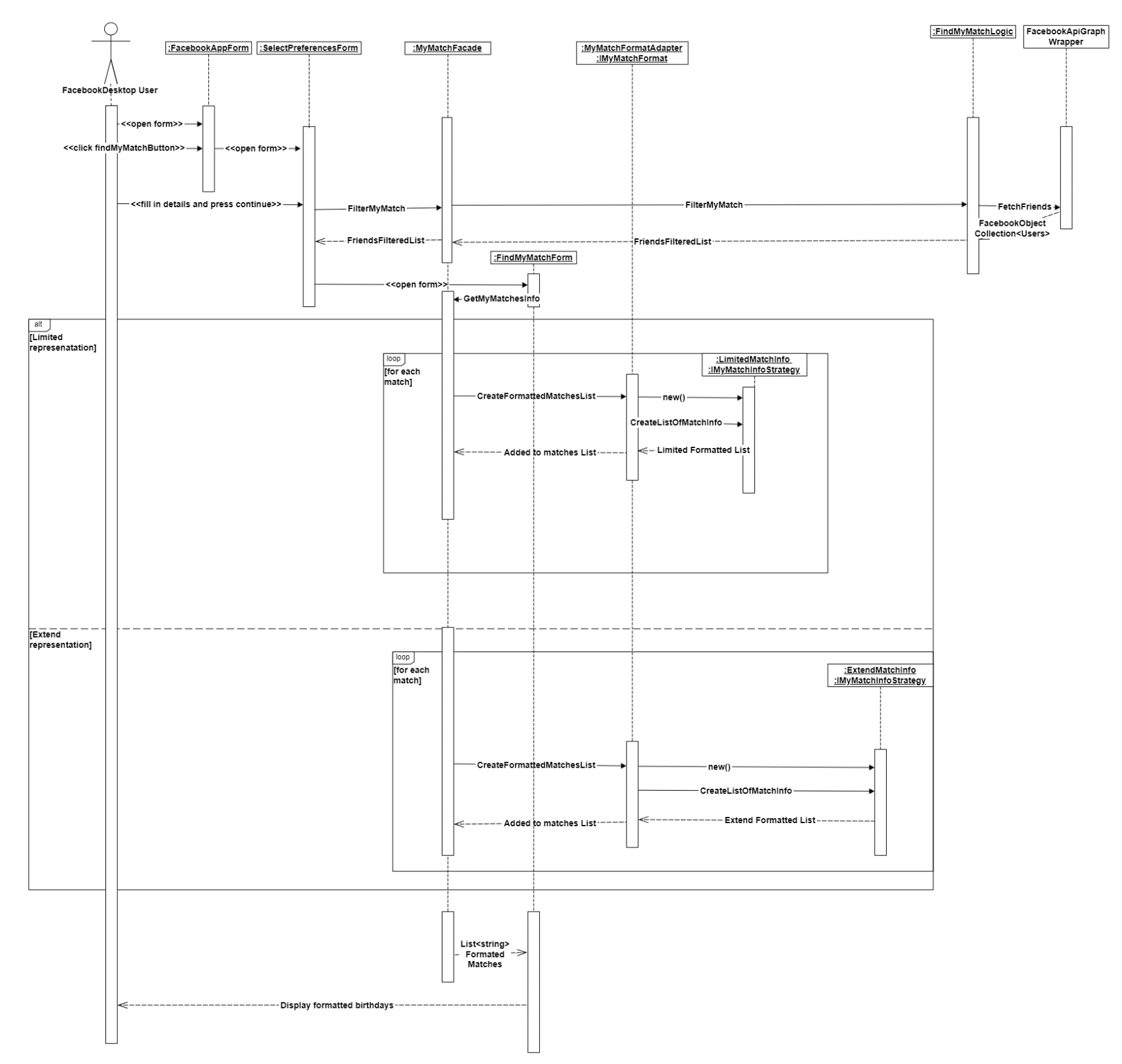
בכדי למנוע את הצורך בשינוי הלוגיקה בכל פעם שתתווסף צורת הצגה חדשה של פרטי   
ההתאמה יצרנו IMyMatchInfoStrategy ובו הגדרנו מתודה שהיא אבסטרקטית CreateListOfMatchInfo המשמשת אותנו כ- injection point , הממשק IMyMatchInfoStrategy ממומש על ידי שתי מחלקות:

* LimitedMatchInfo -אשר מציגה את פרטי ההתאמה בפורמט מצומצם.
* ExtendMatchInfo -אשר מציגה את פרטי ההתאמה בצורה מלאה.

במחלקה MyMatchFormatAdapter יצרנו משתנה מסוג IMyMatchInfoStrategy אשר מאותחל בקונסטרקטור עם ה-strategy המתאים, שנבחר על ידי המשתמש ב- ComboBox שנמצא בטופס.SelectPreferencesForm

המתודה CreateFormattedMatchesList במחלקה- MyMatchFormatAdapter קוראת למתודה בstrategy- המתאים עבור כל ההתאמות של המשתמש ומחזירה את פרטי ההתתאמה בפורמט המתאים על ידי רשימה של מחרוזות ב- FindMyMatchForm שם הרשימה שמכילה את פרטי ההתאמה תוצג למשתמש ב- recommendedMatchesListBox.

* Sequence Diagram



* Class Diagram

Diagram

Description automatically generated

**Context** - MyMatchFormatAdapter

**Strategy** – IMyMatchInfoStrategy

**ConcreteStrategy1** - ExtendMatchInfo

**ConcreteStrategy2** - LimitedMatchInfo

### תבנית מס' 2 – Iterator

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בתבנית זו על מנת לאפשר לממשק המשתמש לעבור על חברי המשתמש העונים על הדרישה שיום ההולדת שלהם נמצא בטווח התאריכים שנבחר על ידי המשתמש, מבלי להכיר את מבנה הנתונים שמומש בשכבה הלוגית. בנוסף, אם בעתיד נרצה לשנות מבנה נתונים זה נוכל לעשות זאת מבלי לשנות כלל את ממשק המשתמש, ובכך נתרום לתחזוקתיות המערכת.

מימשנו את הiterator על ידי שימוש בפיצ'ר yield return ובכך יצרנו protective iterator ובכך אנו מגנים על המידע שנמצא במבנה הנתונים.

כמו כן Iterator זה,מסנן את המידע במבנה הנתונים בהתאם לקרטריון שנבחר על ידי המשתמש בכך אנו מייעלים את המערכת.

* אופן המימוש:

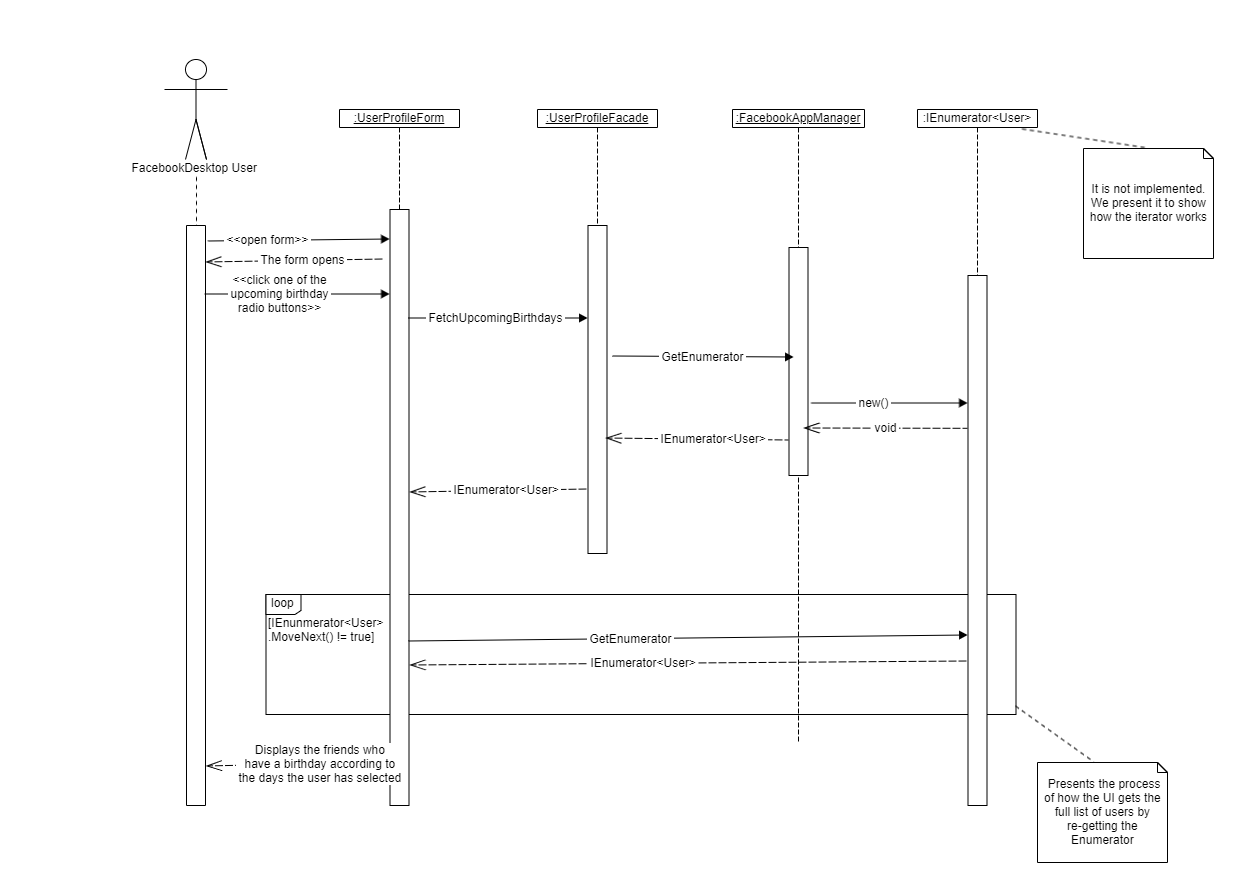
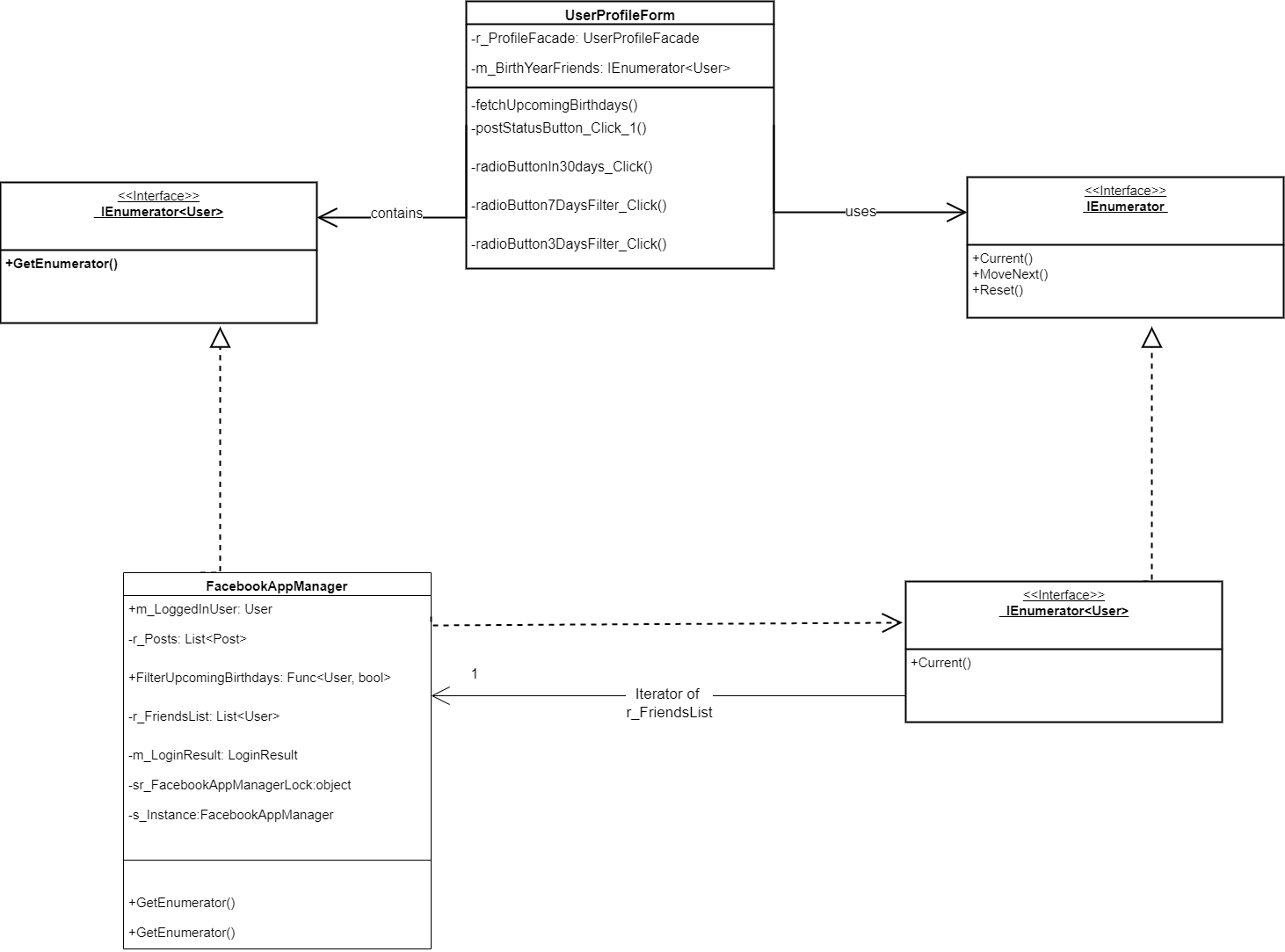
המחלקה FacebookAppManager מממשת את הInterface: IEnumerable<User>

היא מממשת את המתודה GetEnumerator העוברת על כל User ברשימת החברים -r\_friendsList ששיכת למחלקה. המתודה מופעלת עבור כל יוזר ומסננת לפי הקריטריון שנבחר על ידי המשתמש באמצעות yield return.

המתודה המסננת נשלחת מ- UserProfileFacade עם טווח הימים שנבחר על ידי המשתמש שמגיע ממשק המשתמש. מתודה זו נשמרת במשתנה FilterUpcomingBirthdays מסוג Func.

ה- UserProfileFacade מהווה גורם מקשר בין ממשק המשתמש לבין השכבה הלוגית, בכך שהוא מעביר את הקריטריון לסינון ומחזיר את הiterator ל-UserProfileForm.

כאשר המשתמש יבחר טווח ימים על מנת להציג את הימי הולדת של החברים שעונים על קריטריון טווח הימים שנבחר יוצגו לו כל החברים העומדים בתנאי ואם אין חברים שעונים על דרישה זו תוצג לו הודעה על כך.

* Sequence Diagram
* Class Diagram

### תבנית מס' 3 – Template Method

* סיבת הבחירה / שימוש בתבנית:

השתמשנו בפטרן Template Method pattern על מנת למנוע שכפול קוד בין שתי המחלקות: ChartByMonthsLogicו- ChartByYearLogic, שתפקידן ליצור את מבנה הנתונים המכיל את ערכי )Xחודשים או שנים) וה Y(מספר הפוסטים לפי שנה או חודש) של הדיאגרמות. המחלקה ChartByYearLogic יוצרת דיאגרמה של פוסטים לפי שנה והמחלקה ChartByMonthsLogic יוצרת דיאגרמה של פוסטים לפי חודש.

בשתי מחלקות אלו יש מתודה האחראית, על מילוי ערכי ה x וה-Y,היא מייצרת ומחזירה dictionary אשר הkey הוא הx והvalue הוא ה Y לפי הדיאגרמה הרצויה.

בנוסף אם בעתיד היינו רוצים להוסיף עוד דיאגרמה , הדבר היה גורם לשכפול קוד וכתוצאה מזה לבעיות תחזוקתיות.

לכן באמצעות שימוש בתבנית זו ,אם נרצה להוסיף עוד דיאגרמה, התבנית תאפשר לנו לשנות את התנהגות האלגוריתם מבלי לשנות את הקוד שלו, אלא רק את המחלקה שתממש את המטודה האבסטרקטית .

* אופן המימוש:

יצרנו את המחלקה האבסטרקטית ChartLogic, אשר בה יש רשימה המכילה את הפוסטים ואת המתודה fillInChart שהיא ה-Template method.

בנוסף, היא מכילה את המתודות האבסטרקטיות שהן ה-Injection point :

* averageOfPosts
* MinimumNumberOfX
* GetXValueByPost
* JumpsBetweenEachX
* MaxNumberOfX

שתי המחלקות ChartByYearLogic ו- ChartByMonthsLogic, יורשות מהמחלקה האבסטרקטית ChartLogic ושתיהן מממשות את המתודות האבסטרקטיות לפי הנתונים של הדיאגרמה הרצויה.

המחלקה PostRankFacade משתמשת בשתי מחלקות אלו לצורך יצירת מבנה הנתונים המכיל את נתוני הדיאגרמה ומחזירה את מבנה נתונים זה ל PostsChartByMonthsForm או לPostsChartByYearForm בהתאמה.

* 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