**תיאור קצר על הפיצ'רים שבחרנו לממש בתרגיל מספר 1 :**

1. הצגת סטטיסטיקה למשתמש לפי רמת פעילות .  
   בפיצ'ר זה נתנו למשתמש את האופציה לבחור את הפעילות המינימלית והמקסימלית שלו בפייסבוק . המשמש מקבל פירוט על איזה חודש הפעילות שלו הייתה מקסימלית \ איזה חודש הפעילות שלו הייתה הכי מינימלית ומה מספר הפרסומים שהעלה באותם חודשים .למשתמש נפתח באפליקציה טופס חדש עם שני כפתורים אשר לפי בחירת המשתמש מוצגת לו התוצאה המבוקשת.
2. הפיצ'ר השני הוא הצגה למשתמש את הפוסט שקיבל את המקסימום \ מינימום \ ממוצע של לייקים.   
   פיצ'ר זה מקנה למשתמש את הידיעה לגבי הפוסטים שלו , מי הכי פחות אהוב ומי הכי אהוב מבחינת הלייקים . למשתמש נפתח באפליקציה טופס חדש עם שלושה כפתורים אשר לפי בחירת המשתמש מוצגת לו התוצאה המבוקשת.
3. הפיצ'ר השלישי הינו פיצ'ר של תזמן פוסטים לעמוד פייסבוק של המשתמש .  
   הפיצ'ר הזה פותח טופס חדש למשתמש ובו שני מקומות למלא את הקלט המבוקש , הראשון הוא מספר השעות עד להעלאת הפוסט והשני הוא הסטטוס של הפוסט שאותו המשתמש רוצה לעלות . הפיצ'ר הזה מחזיר תגובה למשתמש אם אכן תוזמן בהצלחה והועלה הפוסט או שהפוסט לא הועלה .

**Design Pattern - Iterator**

סיבת הבחירה \ מטרת השימוש–

בחרנו בתבנית Iterator מכיוון שבקוד יש לנו כמה מקומות שבו אנו עוברים על הפוסטים של המשתמש ולפי מידע מן הפוסטים מפיקים מסקנות.   
Iterator הוא תבנית עיצוב ממשפחת " behavioral design pattern" אשר מקנה לנו את האפשרות לעבור על מבנה נתונים מורכב מבלי לחשוף את פרטיו הפנימיים, באמצעות שימוש בתבנית עיצוב זו אם בעתיד ישתנה המבנה נתונים אנו לא נצטרך לשנות את הקוד שלנו ,בנוסף באמצעות השימוש בתבנית זו הקוד יותר מודולרי וקל תחזוקתית.

אופן המימוש –

את המימוש ניתן לראות בכמה מחלקות עיקריות , תחת "FacebookLogic" יצרנו מחלקה בשם " PostIterator" אשר מממשת את הממשק IEnumerable .   
כאשר המשתמש בוחר לדוגמא את הפיצר של : הצגת סטטיסטיקה לפי רמת פעילות או הצגה למשתמש את הפוסט שקיבל את המקסימום \ מינימום \ ממוצע של לייקים , מתבצע מעבר במחלקת ה Façade שהינה ה: "LogicManger" שמה יש data member פומבי בשם   
" PostIterator" ובכל מעבר של משתמש על התמונות בשני הפיצרים האלו במקום לעבור על מבנה נתונים מסוג: FacebookObjectCollection<Post>   
( למשל בלולאת for במתודה ActivityStatics) נעבור על " LogicManager.PostIterator".

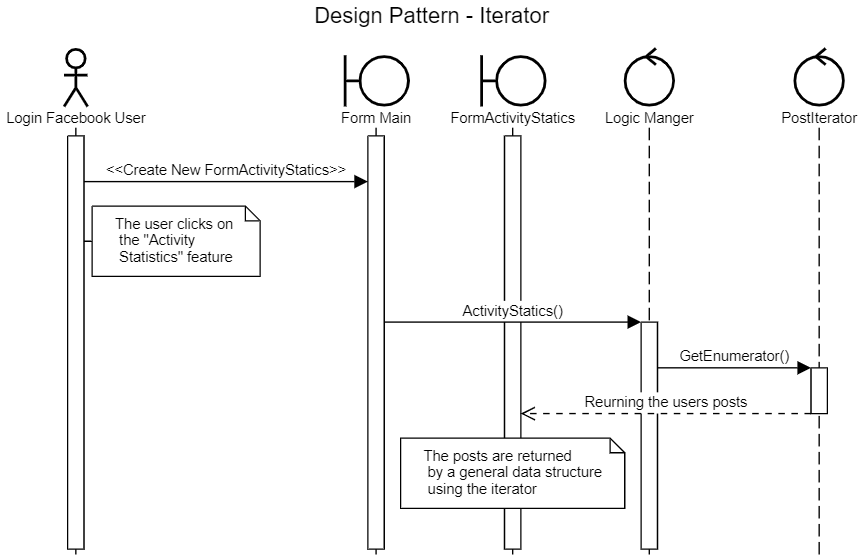
המחלקות בקוד שמייצגות את ה"שחקנים" בתבנית –

Client - FriendsLogic ActivityStaticsLogic,

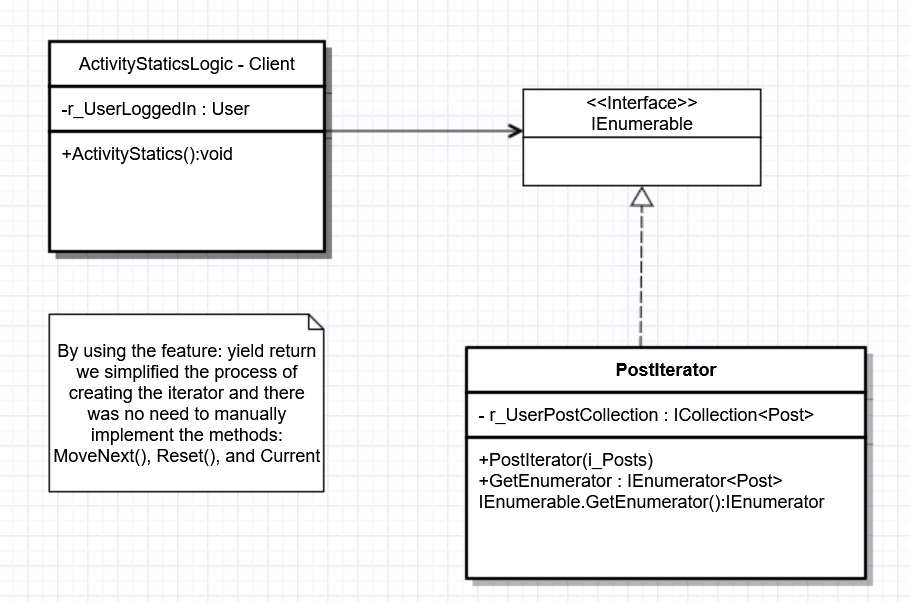
ConcreteAggregate – PostIterator.

[ מחלקות נוספות אשר מעורבות בFlow של התוכנית – LogicManger ]

**Sequence Diagram**



**Class Diagram**



**Design Pattern – Visitor**

סיבת הבחירה \ מטרת השימוש–

בחרנו בתבנית Visitor מכיוון שניתן לראות בפרויקט כי ישנם המון פעמים שאותן הערות בעקבות חריגות מסוימות חוזרות על עצמן כי יש הרבה מקרים שלא ניתן לממש באופן מלא את הפ'יצר החדש במערכת בעקבות העובדה ששרתי פייסבוק חסמו את הגישה ועל כן הערות שגיאה למשתמש חוזרות על עצמן כמה וכמה פעמים במקרה זה . בפרויקט ישנן הערות נוספות כמו התחברות שגויה למערכת , או העובדה שהמשתמש צריך להתחבר כדי להשתמש בתוכנית ועל כן לא יכול ללחוץ על אף פ'יצר עד התחברות לפרופיל שלו בפייסבוק .  
ולכן השתמשנו בתבנית העיצוב Visitor כך שכל פעם שיש את אותה הערת חריגה נוכל להשתמש באובייקט מתאים וכך נוכל לנהל תחזוקתית יותר טוב את המערכת וכן המערכת תיהיה יותר reusable ומודולרית .   
נוסף על כך אם בעתיד נגדיל את מרחב ומימד המערכת שלנו ויהיו חריגות נוספות , נוכל להתמודד איתן במקום אחד ולשנות אותן במקום אחד מה שגורם לקוד להיות קריא יותר .

אופן המימוש –

בתבנית זו מגדירים ממשק ומחלקה אשר מממשת את ממשק זה .  
בממשק נמצאות כל המתודות הפומביות אבסטרקטיות שמטרתן להציג את הודעת השגיאה למשתמש , המחלקה שמממשת את הממשק , מיישמת את המתודות הללו .   
כלל הערות השגיאה נשמרו כ data member וכconst , לשם כך אם בעתיד נשנה את הודעת השגיאה לא נצטירך לשנות את בכל מקום אלא רק data member.

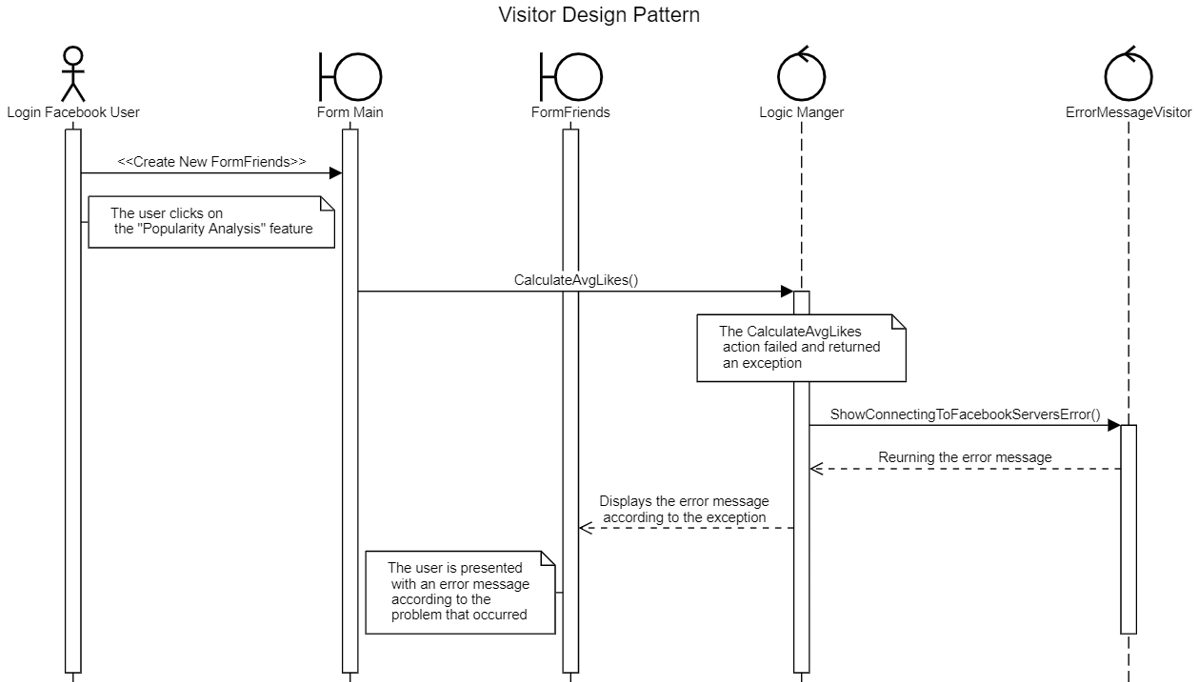
המחלקות בקוד שמייצגות את ה"שחקנים" בתבנית –

Interface - IErrorMessageVisitor.

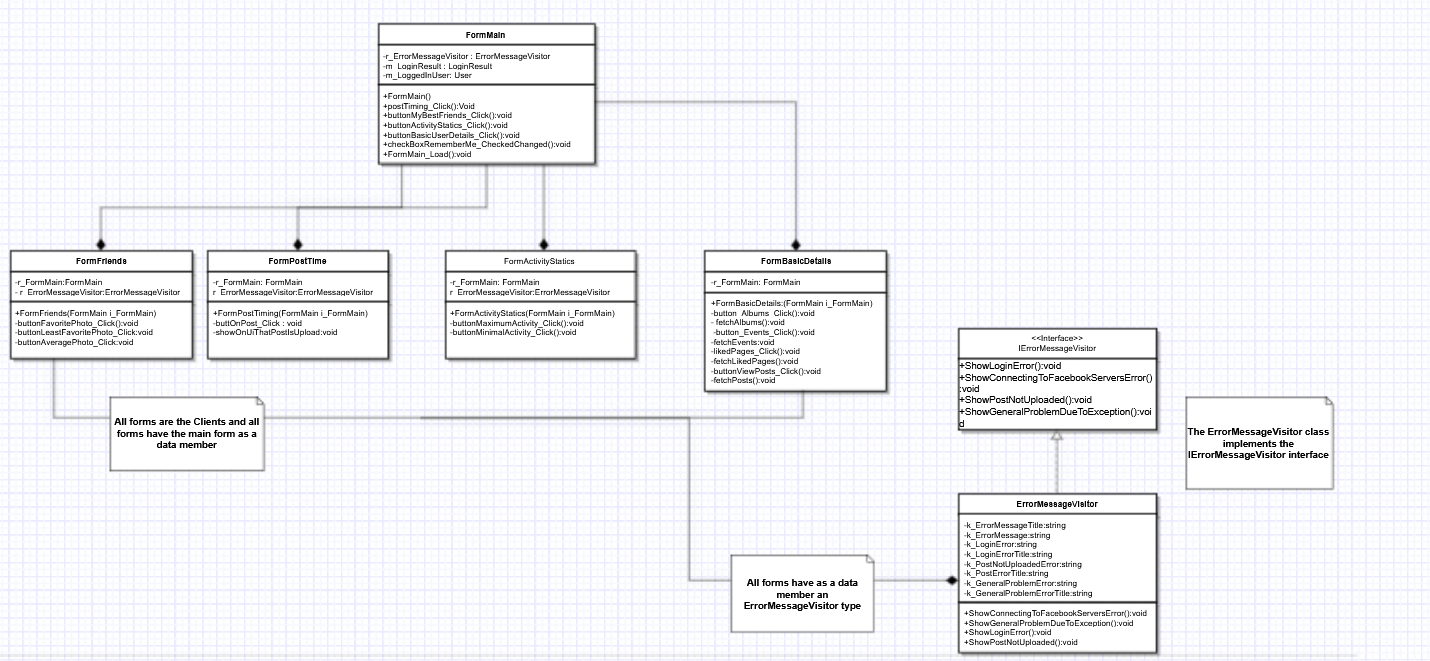
Concrete Visitor – ErrorMessageVisitor.

Client - FormPostTiming, FormFriends, FormMain

**Sequence Diagram**



**Class Diagram**



**Design Pattern – Observer**

סיבת הבחירה \ מטרת השימוש–

בתוכנית שלנו ישנו פיצר שהוא : תזמן פוסטים לעמוד פייסבוק של המשתמש .  
בפ'יצר זה אנו רוצים לתת למשתמש לעלות פוסט מתוזמן לפי מספר שעות וסטטוס אשר הוא רוצה לעלות , ולכן עלינו להפעיל טיימר (מרגע בקשת המשתמש) ובעת סיום הטיימר אנו רוצים לעלות את הפוסט לעמוד הפייסבוק כמתבקש.  
על כן החלטנו לממש את תבנית העיצוב Observer שכן בתכנית שלנו יש מאזין ומודיע ,   
כאשר יש לנו אובייקט והוא רוצה להודיע לכמה אנשים או כאשר המאזין ירצה להאזין לכמה מודיעים ללא היכרות בין המודיע למי שמאזין לו נשתמש בתבנית עיצוב זו .

אופן המימוש –

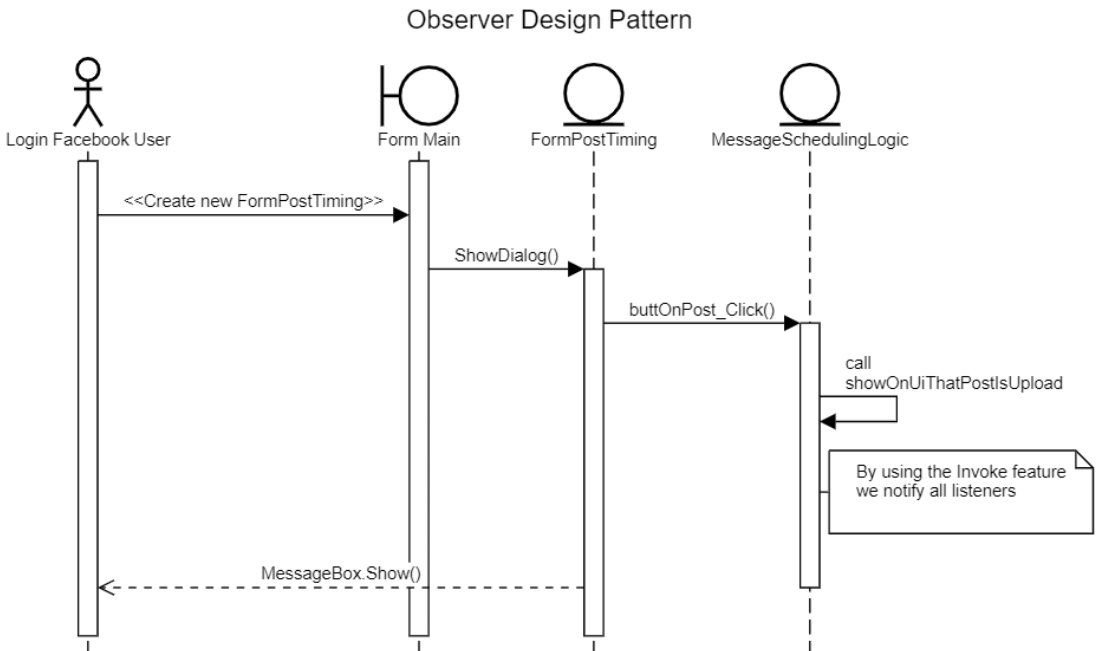
במחלקה FormPostTiming (מייצגת לנו את המאזין בתבנית) ברגע שולחצים על הכפתור "Submit your post" מופעלת המתודה buttOnPost\_Click. מתודה זו תפקידה הוא ליצור את האובייקט MessageSchedulingLogic – שישמש כמודיע ולאחר יצירתו המאזין נרשם באמצעות += לאירוע PostUpload.  
כשאנו נרשמים לאירוע מופעלת המתודה showOnUiThatPostIsUpload ואז במחלקת MessageSchedulingLogic, נוצר טיימר ע"פ בקשת המשתמש, ברגע שהטיימר מסיים הוא מודיע למחלקה MessageSchedulingLogic האם נוצר הפוסט והועלה או שלא באמצעות הInvoke אשר מפעיל את המתודות של כל המאזינים ובמקרה שלנו את המתודה showOnUiThatPostIsUpload (מתודה של המאזין FormPostTiming) .  
אם בעתיד יתווספו עוד מחלקות אשר ירצו "להאזין" לאירוע הם יוכלו להרשם ואז המתודות של כל המאזינים יקבלו את ההודעה שהועלה\ לא הועלה הפוסט וכך המערכת תיהיה מודלורית וreusable מכיןן שנוכל להוסיף "מאזינים" ללא שינוי בקוד.

המחלקות בקוד שמייצגות את ה"שחקנים" בתבנית –

ListenerA – FormPostTiming

Notifier – MessageSchedulingLogic

**Sequence Diagram**



**Class Diagram**

