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

* להעביר תמונות בין אלבומים שונים אצל אותו המשתמש(לראות ולהעביר לאלבום הרלוונטי)
* להעביר פוסטים בין קבוצות כלומר לשלוח את אותו הפוסט להעלאה לקבוצה אחרת מקבוצה מסוימת

### תבנית מס' 1 – Observer

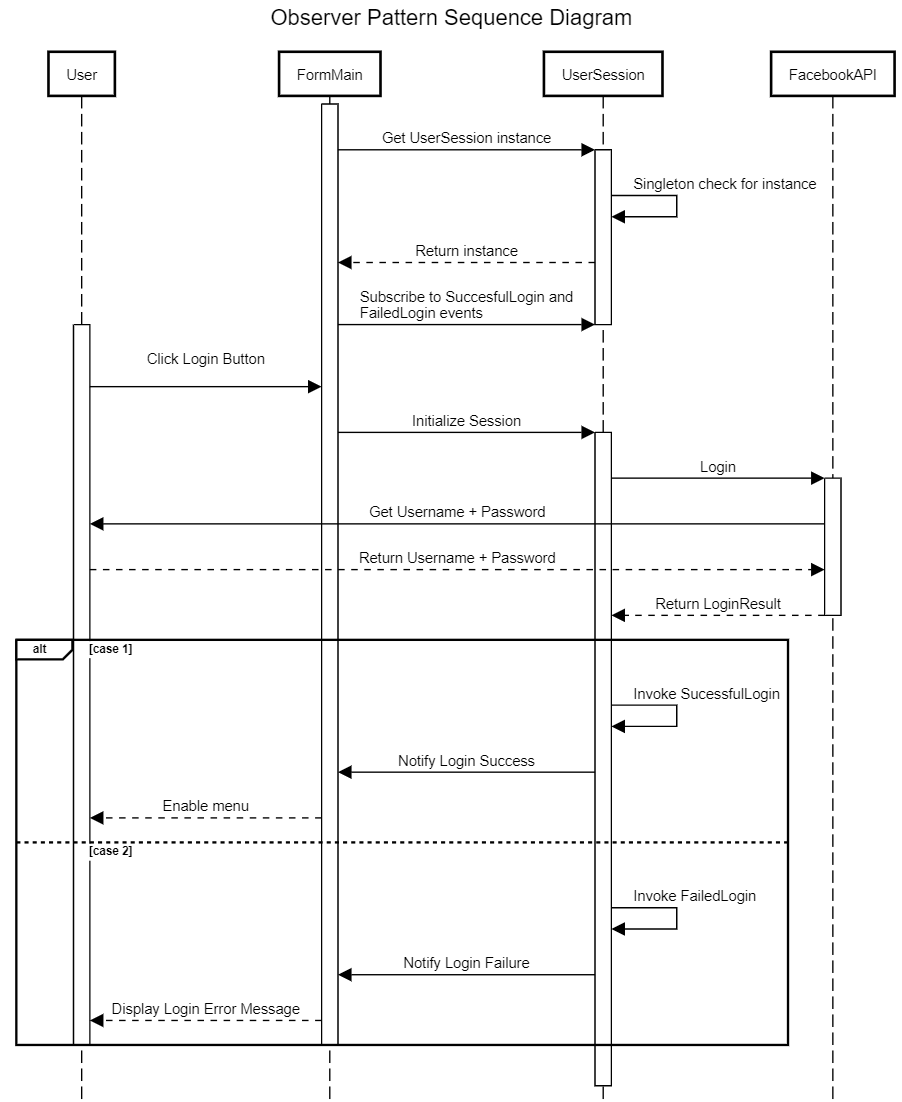
* סיבת הבחירה / שימוש בתבנית:

בתרגיל הקודם מימשנו תפריט ראשי לאפליקצייה עם כפתורים שרק נתינים ללחיצה בהינתן התחברות מוצלחת לפייסבוק. עשינו זאת באמצעות מתודה בקליינט שבודקת אקטיבית אם ההתחברות הייתה מוצלחת ואז מתנהלת בהתאם, שזה לא התנהלות נכונה מהצד שלנו כי אנחנו שמים אחריות על הקליינט להבין איך התחברות מוצלחת מבוצעת ובכך מטילים על הקליינט לבצע שינויים אם משהו בהליך משתנה. מה שהיה יותר נכון זה שהאובייקט שאחראי להתחברות המשתמש יתריע למי שמעוניין במצב ההתחברות שלו ושהקליינט רק יירשם כמאזין לאירוע זה.

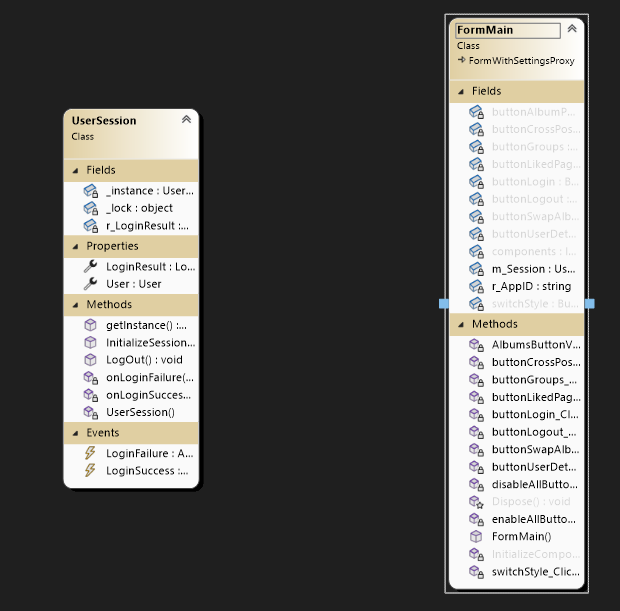
* אופן המימוש:

המימוש בוצע באמצעות שני Actions בתוך מחלקת UserSession: LoginSuccess וLoginFailure. הUserSession מבצע Invoke לאקשן הרלוונטי בהינתן התחברות מוצלחת או לא, וכל מה שהקליינט צריך לעשות זה להרשם לאקשנים בתהליך יצירת הטופס, מה שהוא מבצע בקונסטרקטור של FormMain. המתודה שנרשמת לLoginSuccess היא enableAllButtons והמתודה שנרשמת לLoginFailure היא Lambda Expression שפשוט מבצעת הקפצת התראה למשתמש של כשל בחיבור

* Sequence Diagram



* Class Diagram



Invoke

Notify Listener

### תבנית מס' 2 – Chain Of Responsibility

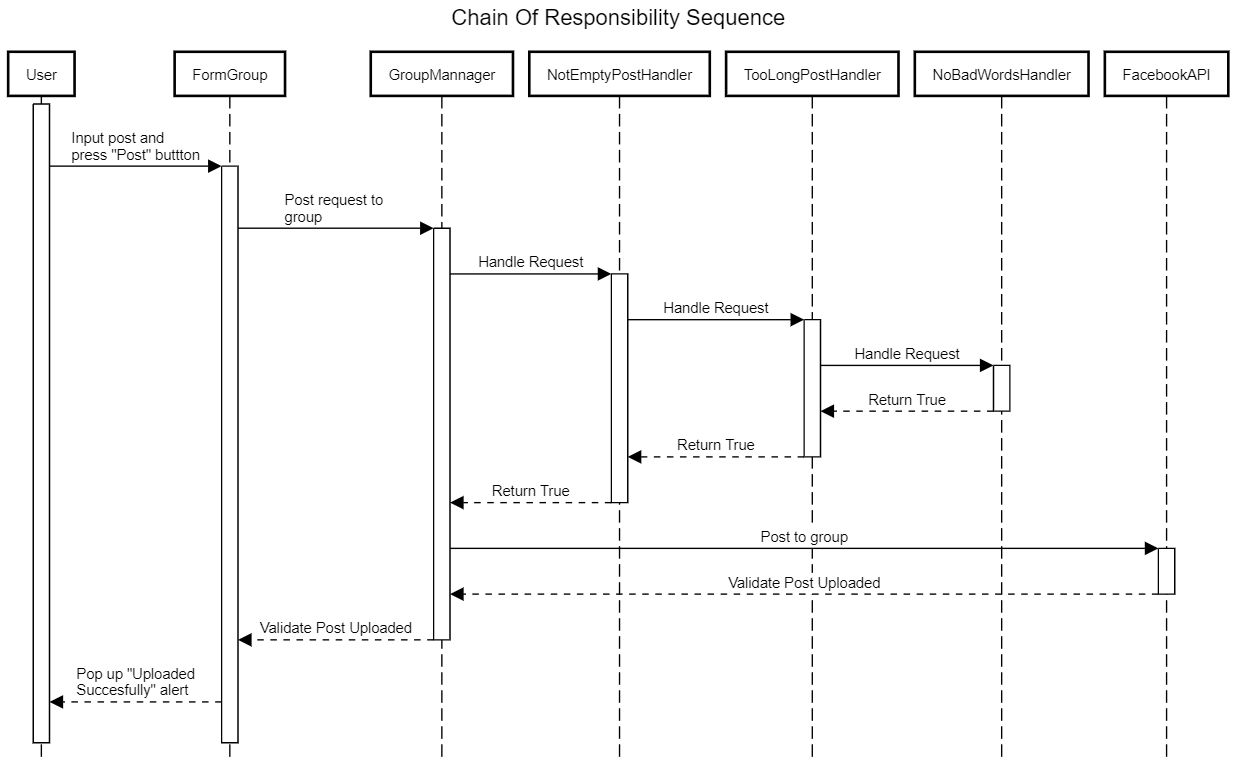
* סיבת הבחירה / שימוש בתבנית:

רצינו לייצר אפשרות לשים למשתמש מגבלות לתוכן שהוא יכול לפרסם בפוסט שלו לפני שהוא מפרסם לקבוצה, למשל מגבלות על אורך הפוסט או המלל שיש בו, ורצינו להיות יכולים להפעיל כמה פילטרים שונים על אותו הפוסט. בגלל שאנחנו מצפים שפוסט ייעבור את כל הפילטרים בשביל להתקבל, אנחנו רצינו שכל אחד מהפילטרים יוכל או להפיל את תהליך פרסום הפוסט או להעביר לפילטר הבא, וכן תבנית Chain Of Responsibility מתאימה בדיוק.

* אופן המימוש:

בנינו אינטרפייס IGroupHandler ומחלקה אבסטרקטית שמממשת אותו BaseGroupHandler שמספקות את המתודות והmembers הבסיסיים של הפאטרן, ואז ייצרנו Handlers קונקרטים שיורשים מהם NotEmptyPostHandler,NoBadWordsHandler,TooLongPostHandler. בנוסף, מחלקת GroupManager שמנהלת את העבודה מול קבוצות מחזיקה IGroupHandler כהHandler בסיס שלה. מצד הקליינט, בטופס FormGroupView אנו מגדירים שרשרת של 3 Handlers אחד אחרי השני.

* Sequence Diagram



* Class Diagram

A screenshot of a computer

Description automatically generated with medium confidence

Concrete Handlers

Client

### תבנית מס' 3 – Strategy

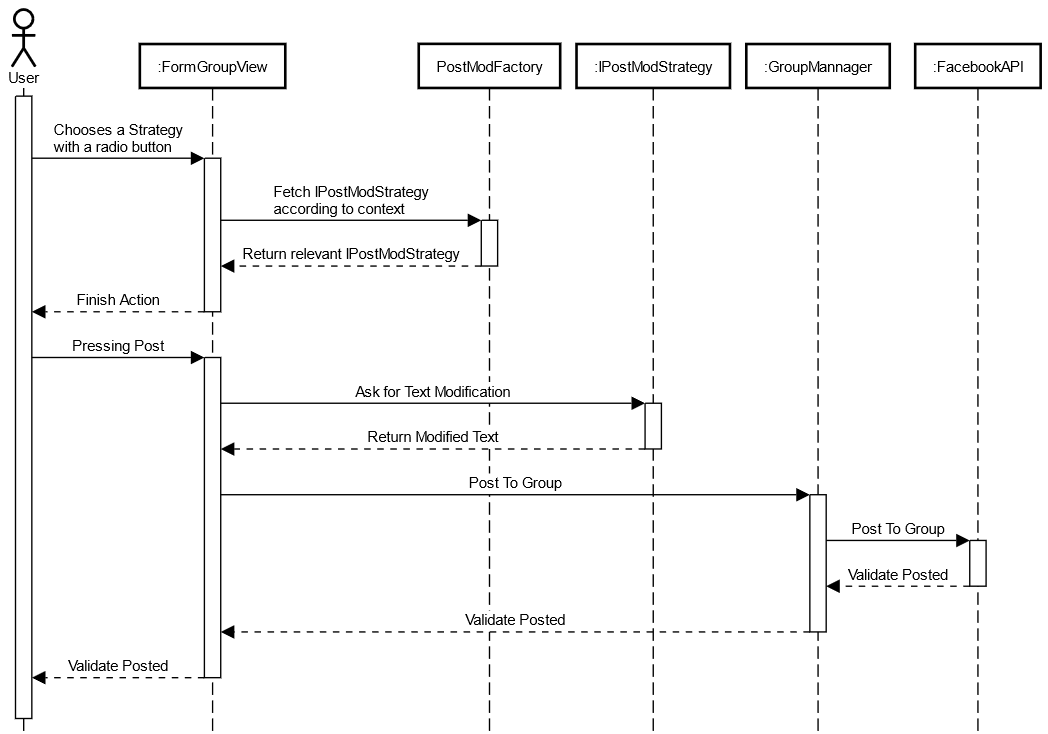
* סיבת הבחירה / שימוש בתבנית:

כמו שיש אפשרות להוסיף חתימה למייל, רצינו להוסיף יכולת להוסיף איזושהי מודיפיקציה באופן אוטומטי לפוסט של משתשמש לפני שהוא מפרסם אותו ואם כבר להוסיף משהו בסוף הפוסט אז כבר אפשר לתת מודיפקציות מסוגים שונים, למשל משהו בהתחלה ("שלום לכולם!" למשל) או אפילו שינוי של התוכן שהמשתמש כתב בעצמו. בשביל זה תבנית Strategy התאימה בדיוק בגלל שהיא נותנת לנו לבצע פעולה (שינוי טקסט) על קונטקסט קבוע (הפוסט של המשתמש) ולהיות יכולים לשנות אותה בזמן ריצה (המשתמש בוחר הרי את האפשרות שהוא מעוניין בה).

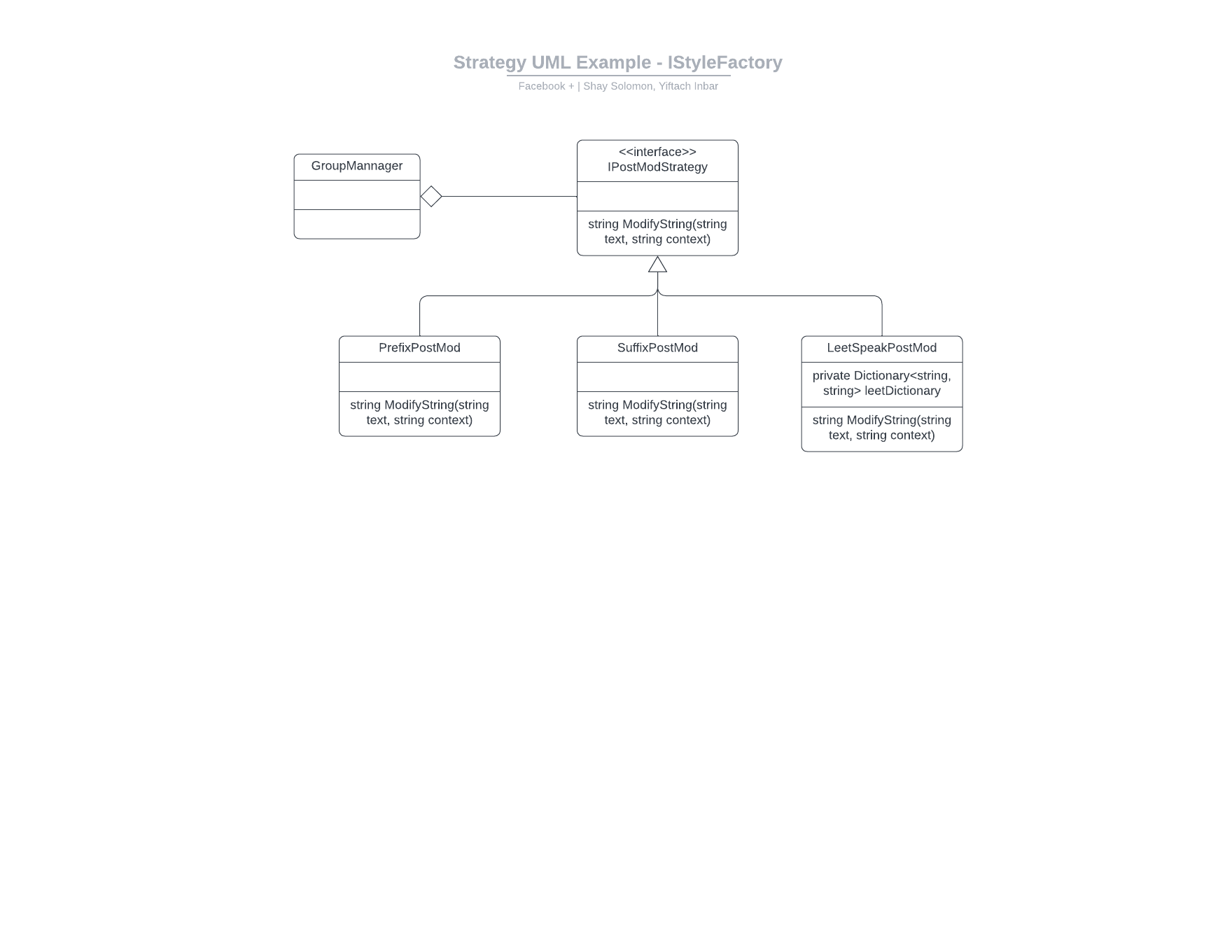
* אופן המימוש:

מימשנו את תבנית זו באמצעות ממשק IPostModStrategy שמחזיק מתודה ModifyText שמקבלת כפרמטרים את הפוסט המקורי ומחרוזת Context ומחזירה מחרוזת חדשה. האסטרטגיות הקונקרטיות הן PrefixPostMod שמחזירה מחרוזת עם תוספת בהתחלה, SuffixPostMod שמחזירה את המחרוזת עם תוספת בסוף וLeetSpeakPostMod שמחזירה את המחרוזת לפי הצופן L33TSP34K האינטרנטי. הטופס FormGroupView מחזיק מופע של ממשק האסטרטגיה, מגדיר אותו באמצעות PostModFactory ואז מפעיל את מתודת ModifyText מהאסטרטגיה הנבחרה לפני שליחת הפוסט

* Sequence Diagram  
  ראוי לציין שכל תהליך הפוסט בתוך הGroupMannager שנכלל בדיאגרמה של Chain Of Responsibility לא תואר פה אבל בוודאי שהוא מתרחש

S

* Class Diagram



Concrete Strategies

Context (Also Client)