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

* SortingFriends: המשתמש מקבל את רשימת חבריו ויכול למיין אותם לפי סוג המיון אותו בוחר. כאשר המשתמש בוחר חבר מסוים, מוצגות תכונות רלוונטיות עבור אותו סוג מיון
* FindBestFriend: המשתמש מחפש את החבר הכי טוב לו (ע"פ מתן משקולת לכל תכונה) שיש לו יום הולדת ב4 חודשים הקרובים ומציג את פרטיו. לאחר מציתא החבר הכי טוב, ניתן ליצור לחבר מסיבת הפתעה

### תבנית מס' 1 – Facade

* סיבת הבחירה / שימוש בתבנית:

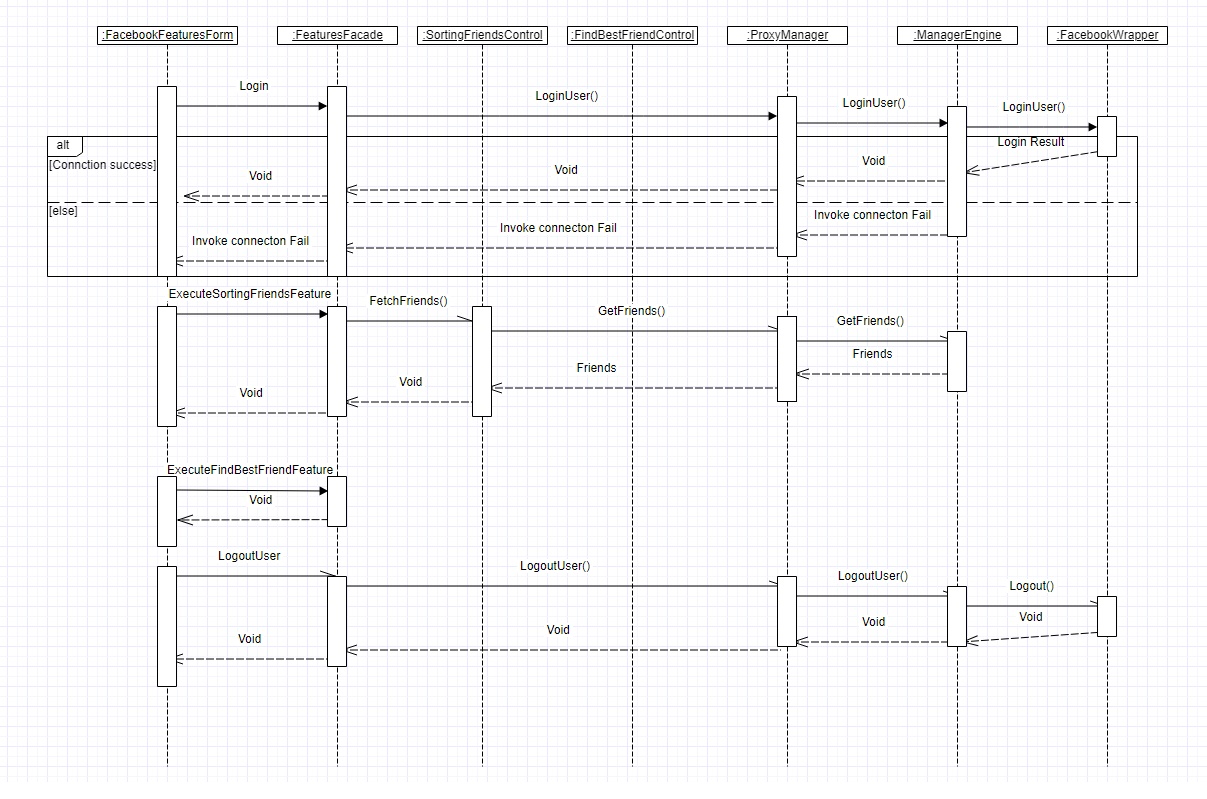
במימוש שלנו מחלקת FeaturesFacade היא המימוש של התבנית ובתפקיד הclient יהיה FacebookFeaturesForm. סיבת הבחירה היא שאם בעתיד נרצה להשתמש בפיצ'רים הללו בcontainer אחר, נצטרך בסך הכל לבצע reuse למחלקת FaturesFacade והוא יאפשר לנו את הממשק עם שני הפיצרים. כלומר הבעיה שאנחנו באים לפתור היא שאם בעתיד נרצה להשתמש באפליקציה רק בcontainer אחר, נוכל לבצע reuse למחלקת FeaturesFacade. התפקיד של המחלקה היא להסתיר מהclient את אופן המימוש והגישה הישירה לתתי המחלקות ולעטוף הכל במחלקה אחת שתאפשר למשתמש נוחות שימוש ובטיחות שימוש.

* אופן המימוש:

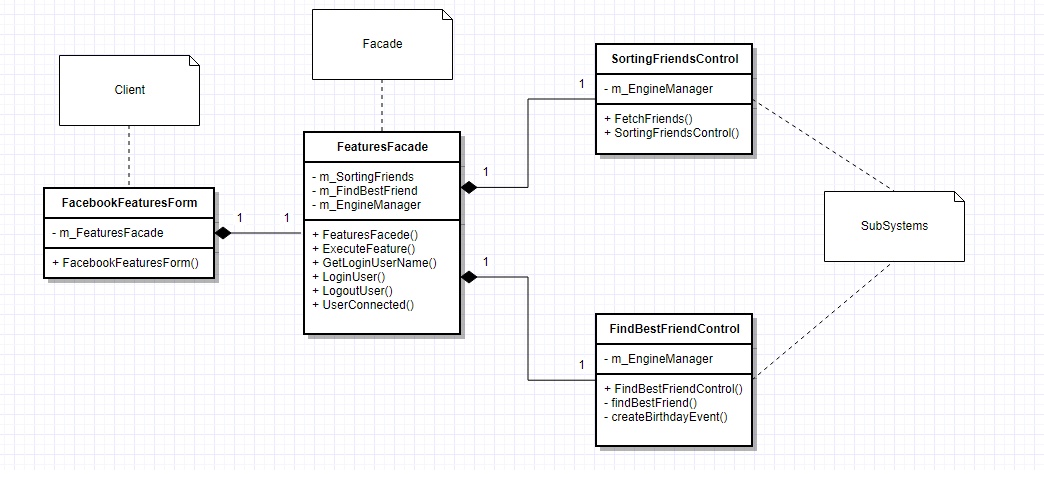
מחלקת FacebookFeturesForm תחזיק מופע של FeaturesFacade אשר בתוכה תכיל את תתי המחלקות שהclient יצטרך את שירותיו והם SortingFriendsControl,FindBestFriendControl,IManager.

בעת פנייה של המשתמש לFacebookFeaturesForm עם פעולה כלשהי שירצה לבצע, הפעולה תועבר לטיפול הFacade ותוחזר שגיאה במידת הצורך למחלקה.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Singleton

* סיבת הבחירה / שימוש בתבנית:

במימוש שלנו מחלקת ProxyManager היא המימוש של התבנית ובתפקיד הclients יהיו FacebookFeaturesForm, SortingFriendsControl, FindBestFriendControl.

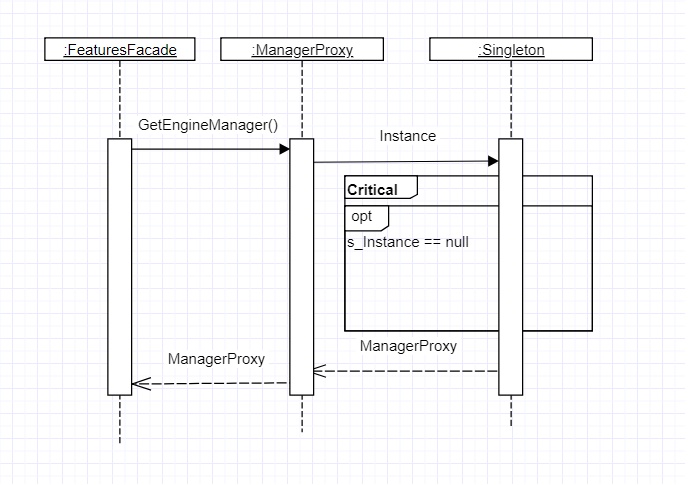
הבעיה שהתבנית באה לפתור היא שאם בעתיד נרצה להשתמש בשירותי מחלקת ProxyManager במקומות שונים בקוד נוכל להשתמש במחלקה ביוזמתה.

הסיבה לבחירת התבנית היא שמטרת האובייקט ProxyManager היא לרכז את כל המנועים של הפיצ'רים ואנחנו משתמשים במנועים האלה במחלקות שונות בקוד אשר כל אחת מהן צריכה גישה לProxyManager אשר מכילה מידע משותף הנחוץ למשתמשים במחלקה ולכן אופן היצירה של האובייקט צריך להיות מנוהל במקום אחד והוא במחלקה עצמה.

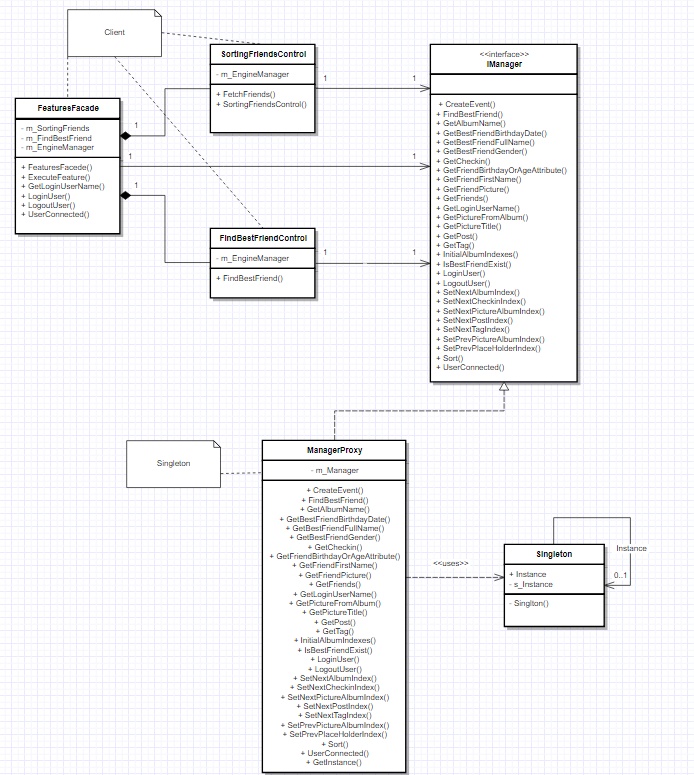
* אופן המימוש:

מחלקות FacebookFeaturesForm, SortingFriendsControl, FindBestFriendControl מחזיקות אובייקט מסוג IManager (ממשק אשר מאפשר למשתמש לבצע פעולות בצד השרת הנוגעות לפיצרים בהם ירצה להשתמש). ובעת יצירת כל אחד מהאובייקטים נפנה למתודה סטטית של ProxyManager ששמה GetInstance אשר תפנה למחלקת Singleton שתבדוק האם קיים כבר אובייקט כזה ואם כן תחזיר את אותו אובייקט אחרת, תיצור בעצמה אובייקט, תשמור אותו ותחזיר אותו למשתמש. לאחר מכן המחלקה יכולה להשתמש בשירותי המחלקה בקוד.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Proxy

* סיבת הבחירה / שימוש בתבנית:

במימוש שלנו מחלקת ProxyManager היא המימוש של התבנית. בתפקיד הclients יהיו FacebookFeaturesForm, SortingFriendsControl, FindBestFriendControl, הממשק הוא IManager והrealSubject הוא מחלקת EngineManager.

בעת הרצת הפיצרים, מתבצעים חישובים אשר לוקחים זמן (עקב בקשות לשרתי פייסבוק) ולכן רצינו לשמור אותם ולאפשר למשתמש להשתמש בהם גם אם התנתק והתחבר מחדש.

לכן סיבת הבחירה היא שהאפליקציה מאפשרת למשתמש אשר התנתק מהאפליקציה (לא סגר אותה) והתחבר מחדש תוך 20 דקות לשחזר את נתוניו ובכך להנות ממהירות השימוש בתכני האפליקציה. הבעיה שהתבנית באה לפתור היא שהמשתמש רוצה להשתמש בשירותי האפליקציה אשר לא מאפשרת שמירה של נתוניו לפרק זמן של 20 דקות ולכן מחלקת ProxyManager מאפשרת לclient שימוש דומה כפי שהיה משתמש בשירותי EngineManager אך גם מאפשרת לו לשמור את נתוניו לפרק זמן של 20 דקות.

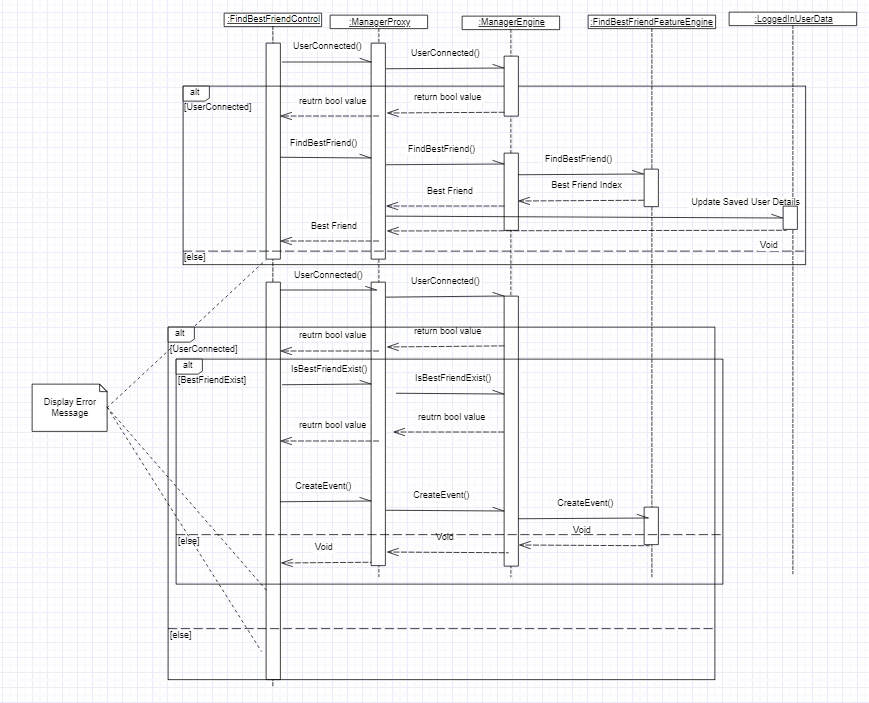
בנוסף אם בעתיד ירצה המשתמש שירות נוסף הקשור לנתונים המובאים מהאפליקציה, יהיה קל יותר לתחזק את הקוד שמרוכז במחלקה אחת.

* אופן המימוש:

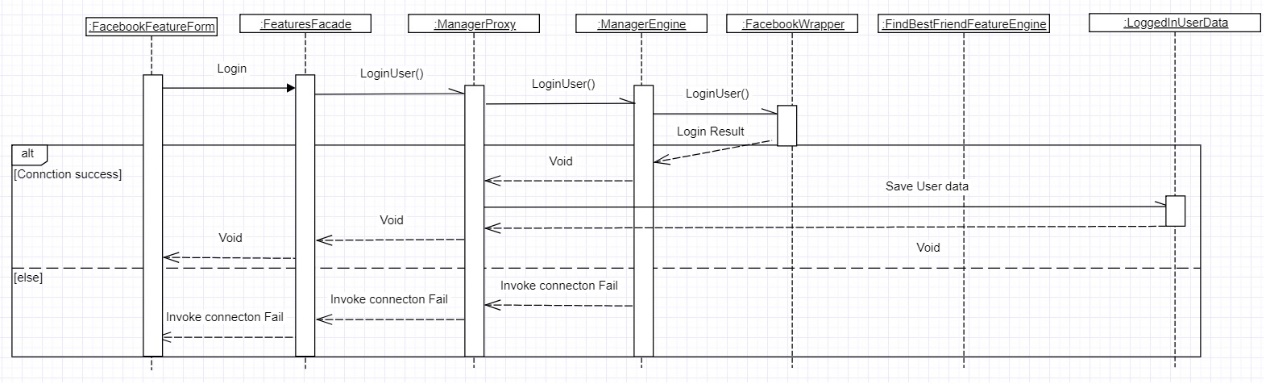
מחלקות FacebookFeaturesForm, SortingFriendsControl, FindBestFriendControl מחזיקות אובייקט מסוג IManager. הפנייה אל שירותי EngineManager(realSubject) נשארים כפי שהיו.

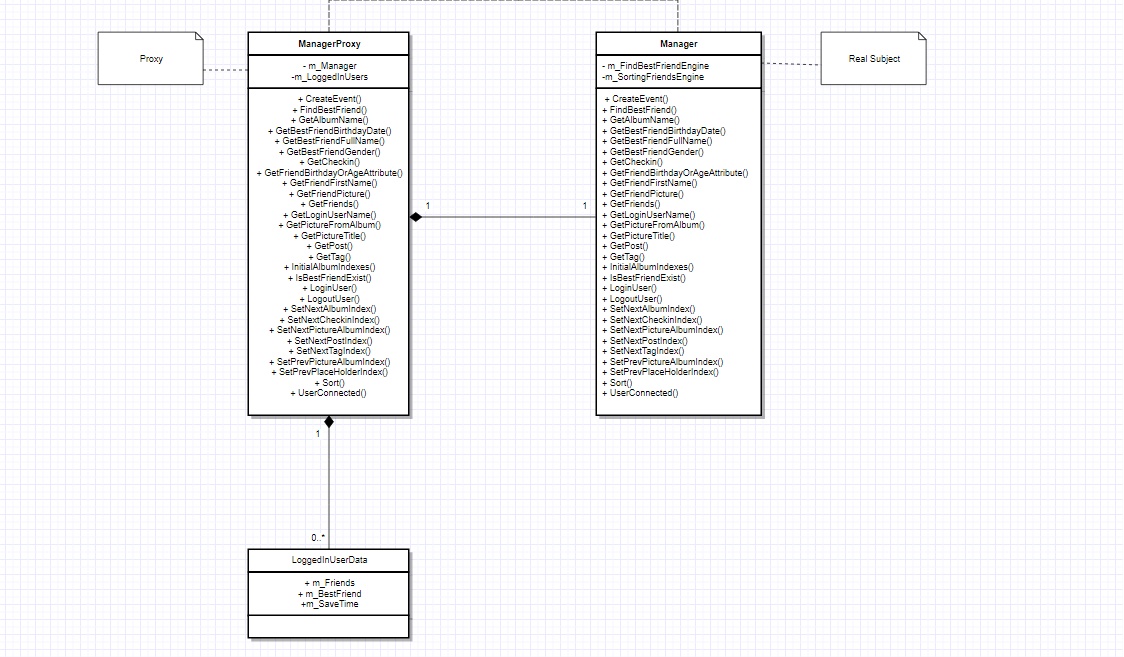
מחלקת proxyManager תקרא לאותה מתודה זהה בEngineManager בתוספת קוד אשר יבצע שמירה ועדכון של נתוני המשתמש.

* Sequence Diagram



המשך האינטרקציה עם המחלקה



* Class Diagram

### 

### עבודה אסינכרונית

במחלקת FacebookFeaturesForm:

מתודות logoutUser עושות שימוש בthread נפרד ובתוך כל אחד מהן, כאשר נרצה לעדכן את פקדי הui נשתמש בinvoke, בכדי שהthread שיצר אותם יפעיל אותם.

במחלקת FeaturesFacade:

מתודות display FindBestFriendFeature, displaySortingFriendsFeature עושות שימוש בthread נפרד ובתוך כל אחד מהן, כאשר נרצה לעדכן את פקדי הui נשתמש בinvoke, בכדי שהthread שיצר אותם יפעיל אותם.

במחלקת FindBestFriendControl:

מתודות findBestFriend, createEvent עושות שימוש בthread נפרד ובתוך כל אחד מהן, כאשר נרצה לעדכן את פקדי הui נשתמש בinvoke, בכדי שהthread שיצר אותם יפעיל אותם.

במחלקת SortingFriendsConrol:

מתודות suitFunctionalityBySelectedIndex, displayFriendAttributes, sortFriendsByUserChoice עושות שימוש בthread נפרד ובתוך כל אחד מהן, כאשר נרצה לעדכן את פקדי הui נשתמש בinvoke, בכדי שהthread שיצר אותם יפעיל אותם.

### עבודה Data Binding

פקדי תצוגת הbest Friend Details עושות שימוש בDataBinding.