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

* פיצ'ר שמראה הצעות לניצול זמן טוב יותר. כאשר עוברים לTAB של Timers, מופיע טיימר שמראה כמה זמן המשתמש "בזבז" באפליקציה, בבנוסף יש כפתור כשלוחצים עליו החלון יציג למשתמש משפט של מה המשתמש יכל לעשות בהתאם לזמן שבזבז באפליקציה, המשפטים בהתאם לזמן שאפליקציה פתוחה נמצאים במילון תחת המחלקה TimeWellSpend אשר נמצא בModel.
* פיצ'ר עריכת תמונות מהאלבומים ושמירתם. כאשר פותחים את TAB האלבומים, ניתן לערוך את התמונה באמצעות כפתור Edit וגם ניתן לשמור לוקאלית את התמונה שנערכה. ניתן למצוא בקוד: בView תחת חלון FormEditPicture.

### תבנית מס' 1 – [Singleton]

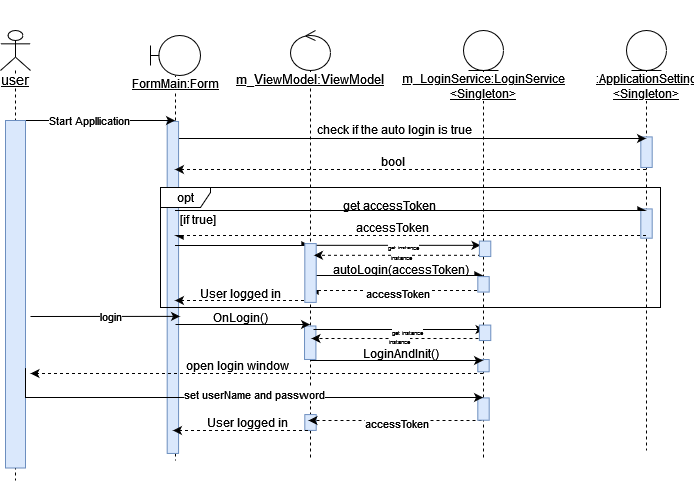
* סיבת הבחירה / שימוש בתבנית:

יצרנו במערכת מחלקה שנקראת LoginService. המחלקה אחראית על התחברות (והתנתקות) אל שרתי פייסבוק. מכיוון שהאפליקציה מיועדת לשרת משתמש אחד בכל "סשן" לא רצינו שיהיו כמה מופעים למחלקה זו, מכיוון שמופעים רבים היו גורמים להתנגשויות בתוכנית. לכן "עטפנו" את מחלקת LoginService בתבנית של סינגלטון, וכך, גם היצירה של אותה מחלקה קוראת רק כשיש צורך, וגם בצורה זו אנחנו מבטיחים שהמחלקה תהיה נגישה מכל מקום שיש לכך צורך בתוכנית, ותמיד תחזיר את אותו מופע, ו/או את המצב הקיים של המחלקה.

* אופן המימוש:

המימוש של התבנית הוא באמצעות יצירת מחלקה גנרית שנקראת Singleton שנמצאת בModel, תוך התחשבות בצורך שלMulti Threading בשיטת Double check Lock. בתוך מחלקת LoginService יצרנו קונסטרקטור פרטי והוספנו תכונה של Instance סטטית וציבורית שמחזירה את המופע היחיד, אם קיים, ואם לא יוצרת אותו בפעם הראשונה.

# אפשר למצוא בקוד תחת FACEBOOK MODEL את מחלקת Singleton וגם את LoginService



* Sequence Diagram
* Class Diagram



### תבנית מס' 2 – [Adapter]

* סיבת הבחירה / שימוש בתבנית:

עצם מימוש הAdapter מעניק לנו הפרדה בין שכבת הmodel לview, ומאפשר לנו לממש MVVM בצורה נכונה.

בנוסףת המחלקות הממומשות בFacebook wrapper כמו Post, Group, Album, Page, Event בעלות תכונות רבות שלא משומשות כלל באפליקציה.

הadapters עוטפים את המורכבות של אובייקטי ה-API ומציגים ממשק פשוט (IPost, IGroup וכו') שהאפליקציה יכולה לעבוד איתו. הפשטה זו משפרת את קריאת הקוד, יכולת התחזוקה והמודולריות.

על ידי התאמת כל אובייקטי ה-API השונים של Facebook לinterface משותף, האפליקציה יכולה להתייחס אליהם בצורה אחידה. זה מאפשר טיפול עקבי בסוגים שונים של אובייקטים ומפשט את היגיון היישום.

הadapters מקל על הרחבת האפליקציה עם תכונות חדשות או תמיכה באובייקטים נוספים של פייסבוק API.

על ידי יצירת adapter חדש עבור אובייקט API חדש, אתה יכול לשלב אותו בצורה חלקה ביישום הקיים מבלי לשנות את לוגיקה.

* אופן המימוש:

יצרנו adapters:

AdapterPost, AdapterGroup, AdapterAlbum, AdapterPage, AdapterEvent למחלקות שיש לנו שימוש בהם באפליקציה שלנו. מחלקות אלו ממשות את ה Interfaces שיצרנו (IPost, IGroup ... ) בהתאמה.

בתוך מחלקת FacebookUser, בעת טעינת Posts, Groups, Albums, Pages, Events מ-Facebook API, נוצרים מתאמים עבור כל אובייקט API באמצעות AdapterFactory (שעליו נרחיב בהמשך). כל Adapter מכיל את האובייקט המקורי של FACEBOOK API וכמו כן ממש את הINTERFACE הרלוונטי.

הadapters שנוצרו מתווספים לאחר מכן לאוספים המתאימים (m\_PostCollection, m\_GroupCollection וכו') של אובייקט FacebookUser.

באמצעות הadapters האפליקציה יכולה לטפל בכל אובייקטי ה-API השונים של Facebook בצורה אחידה דרך interface משותף (IPost, IGroup וכו'), וכך להציג אותם למשתמש.

#אפשר למצוא בקוד תחת FACEBOOK MODEL ואת הInterface הרלוונטים תחת COMMON.CONTRACT

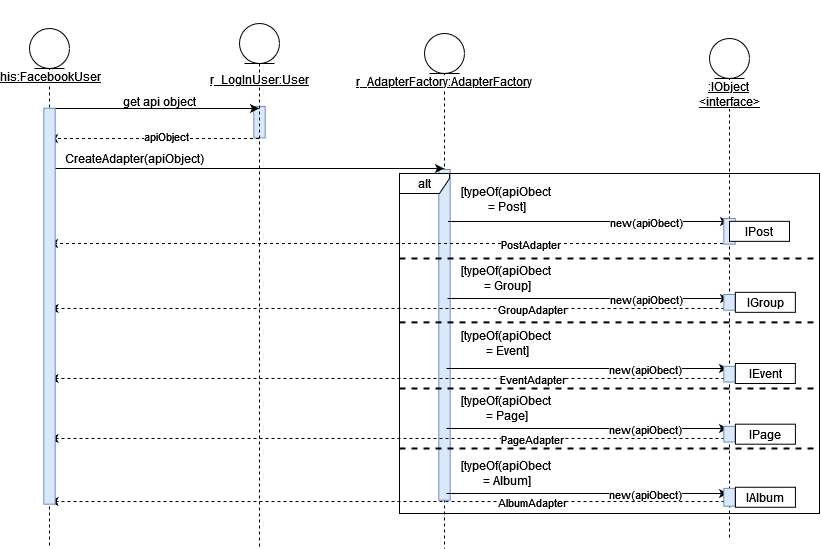
* Sequence Diagram
* Class Diagram

@ ה"שחקנים":

IObject as “Target”

FacebookWrapperOwnerObject as “Adaptee”

**PostAdapter,PageAdapter,EventAdapter,GroupAdapter,AlbumAdapter as “Adapter”**

**AdapterFactory as “Client”**



### תבנית מס' 3 – [Factory Method]

* סיבת הבחירה / שימוש בתבנית:

הAdapterFactory מרכז את תהליך יצירת הadapters במקום אחד, מה שמקל על הניהול והשינוי בעתיד.

הfactory מאפשר למחלקת FacebookUser לעבוד עם interface במקרה שלנו IAdapterFactory במקום ליצור ישירות adapter קונקרטי.

הfactory מקל על הרחבת היישום עם סוגים חדשים של adapters. על ידי הוספת מטודה חדשה לIAdapterFactory ומימוש במחלקה AdapterFactory,

ניתן יהיה ליצור ולהוסיף adapters חדשים מבלי לשנות את קוד הלקוח, מה שמאפשר שיפורים עתיים קלים יותר.אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

* Sequence Diagram

מכיוון ששילבנו בין אדפטר דיזיין פאטרן לבין פאקטורי מטוד דיזיין פאטרן, הSEQUENCE DIAGRAM של האדפטר מכיל את הפקטורי!

* Class Diagram

@ה"שחקנים":

FacebookUser as “the Client”

AdapterFactory as “**Creator”**

**IObject (and his implementer) as “Product”**

**PostAdapter,PageAdapter,EventAdapter,GroupAdapter,AlbumAdapter as “Concrete Product”**

