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

* SoundManager

הפיצ'ר הינו מנהל מוזיקה, המאפשר למשתמש להעלאות, להפעיל ולעצור שיר דף הפייסבוק שלו. השיר מהווה מעין שיר נושא עבור כל משתמש, כאשר אעבור לדף של משתמש אחר אוכל לשמוע את שיר הנושא של הדף שלו. באפשרות המשתמש להפעיל ולעצור את השיר בנוסף לאפשרות להעלות שיר לדף שלו.

את הפיצ'ר ניתן למצוא בדף הראשי של המשתמש(FormMain) וגם בדף של חברים.

* Advanced search

הפיצ'ר הינו כלי לחיפוש חברים בעלי תחומי עניין ספציפיים, הכלי מאפשר למשתמש לחפש חברים אשר חולקים LIKED PAGES ואירועים משותפים בטווח גילאים מסוים.

את הפיצ'ר ניתן בדף ייעודי לכלי(AdvancedSearchForm)

### שימוש ב- MultiThreading:

השתמשנו במולטי טרדינג בFormMain על מנת לחסוך בזמן של הקיראות מGraphAPI ובעצם שחלון הזמן בין שהעמוד עלה לבין שהוא מלא בכל המידע הרלוונטי יהיה קצר ככל האפשר.

כיצד: בFormMain במתודה בשם FetchData יצרנו thread עבור כל משיכת נתונים שאנחנו מבצעים מהשרת כך שקריאת הנתונים והפעולות שעושים עליהם(במידה ועושים) מתבצעות ב thread אחר ועדכון הנתונים בUI מתבצע בעזרת Invoke של ה control אותו נרצה לעדכן.

### שימוש ב- DataBinding:

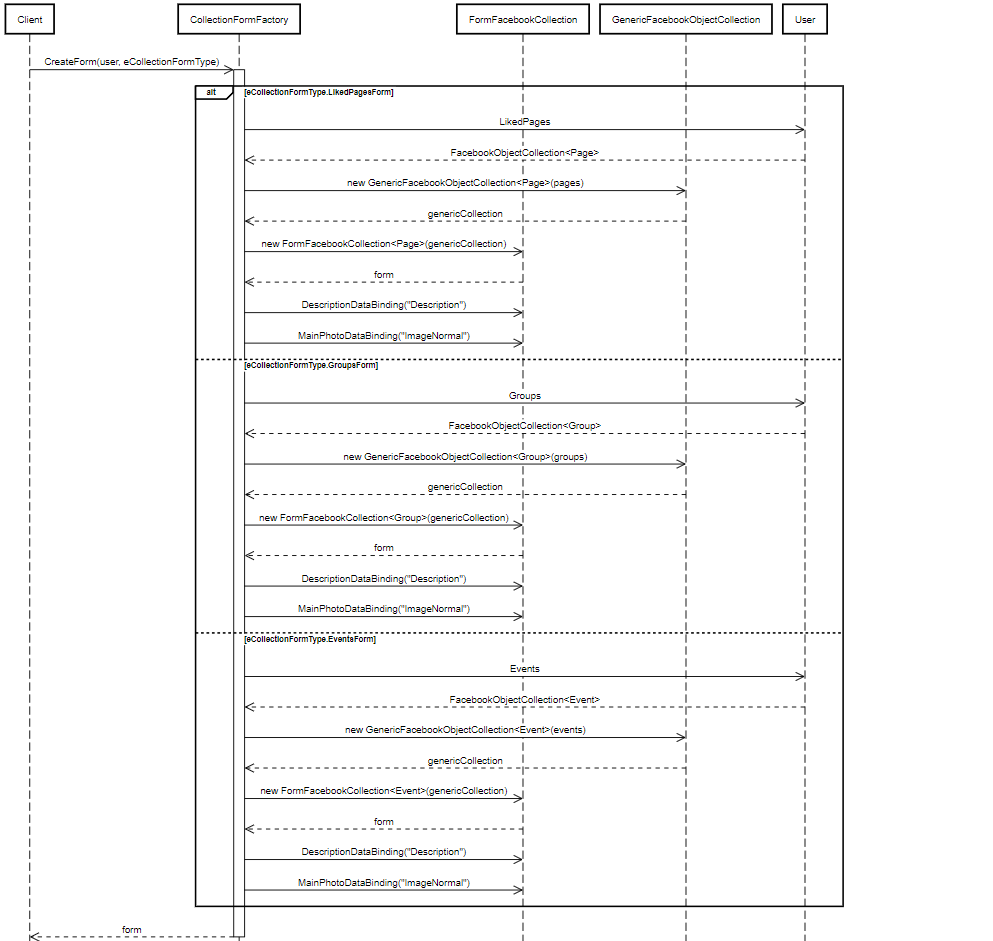
* ב FormAdvancedSearch: השתמשנו ב dataBinding עבור הרשימות יראה את התמונה המתאימה כל פעם לאובייקט שנבחר.

### תבנית מס' 1 – [Factory Method]

* סיבת הבחירה / שימוש בתבנית:  
  השתמשנו בתבנית זו בכדי לאפשר יצירה של טופס כללי עבור כמה אובייקטים שונים, ובעצם למנוע מצב של מימוש טפסים זהים עבור כל אובייקט, וגם כאשר רוצים לבצע שינוי אנחנו מונעים חזרתיות בביצוע השינוי בכמה מקומות.
* אופן המימוש:  
  יצרנו מחלקה סטטית CollectionFormFactoryאשר מקבלת enum ואת ה m\_LoggedinUser

לפי הכפתור אשר נלחץ ב FormMain נשלח את ה enum המתאים לfactory ובהתאם ל enum יצרנו את הטופס אשר המשתמש מבקש.

* Sequence Diagram



* Class Diagram

תמונה שמכילה טקסט, תרשים, קו, עלילה

התיאור נוצר באופן אוטומטי

### תבנית מס' 2 – [Adapter]

* סיבת הבחירה / שימוש בתבנית:

כחלק מהצגת הפרופיל של המשתמש, רצינו להציג את ה-About של המשתמש. מטרתנו הייתה לבצע משיכה של המידע בצורה מודולרית ככל האפשר, ולאפשר הוספה של משיכת פרטי About בצורה שונה למקרים עתידיים.

* אופן המימוש:

[יצרנו ממשק (interface) שניגשים אליו דרך ה-facade, ומחלקת Adapter שמממשת את הממשק הזה. מחלקת ה-Adapter מחזיקה את אובייקט המשתמש (user), ממנו היא לוקחת את המידע על ה-About ומחזירה אותו כרשימה של מחרוזות (בהתאמה ל-groups, pages וכו'). כך אנחנו מקלים על השימוש עבור המשתמש.

* Sequence Diagram

תמונה שמכילה טקסט, תרשים, מקביל, תוכנית

התיאור נוצר באופן אוטומטי

* Class Diagram

[תמונה שמכילה טקסט, תרשים, קבלה, מקביל

התיאור נוצר באופן אוטומטי

### תבנית מס' 3 – [Facade]

* סיבת הבחירה / שימוש בתבנית:

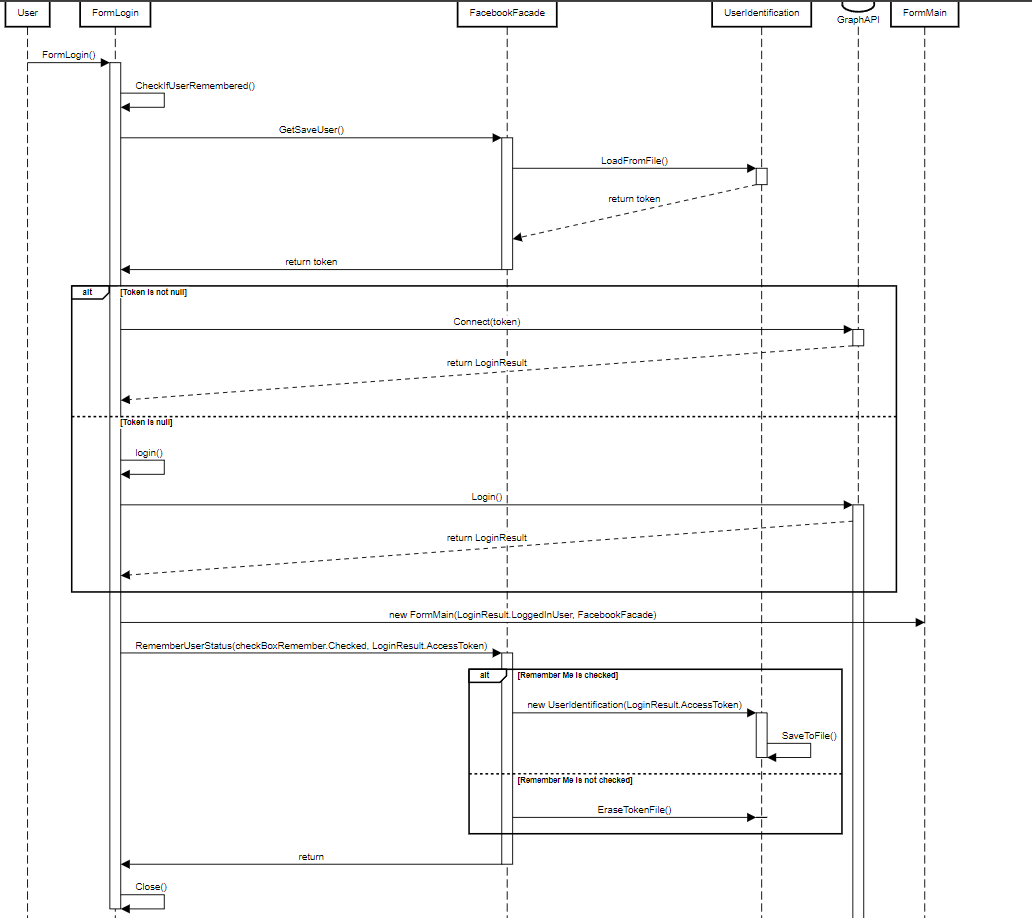
רצינו לבצע הפרדה ברורה בין חלק ה-UI לחלק הלוגי של המערכת. שימוש ב-Facade מאפשר מודולריות גבוהה ותורם לתחזוקתיות המערכת על ידי צמצום התלות הישירה בין ה-UI לרכיבי הלוגיקה.

אופן המימוש:

יצרנו את הFacade בתצורה Lazy כלומר איך שעולה מסך התחברות אנחנו יוצרים את המחלקה, והFacade הינו שקוף יש אפשרות למשתמש לגשת .

המחלקות הלוגיות נמצאות בתוך המחלקה כMembers והוא מבצע שימוש בפונקציונליות הרלוונטית.

* Sequence Diagram



* Class Diagram

תמונה שמכילה טקסט, תרשים, מקביל, קו

התיאור נוצר באופן אוטומטי

### תבנית מס' 4 – [Singleton]

* סיבת הבחירה / שימוש בתבנית:

SoundManager הוא מחלקה מסוג סינגלטון רצינו שכל הניהול של השירים יבוצע במקום אחד, ואם יש מעבר מהפרופיל הראשי לפרופיל של חבר אז המחלקה תטפל בזה בהתאם.

אופן המימוש:

יצרנו את SoundManager בתצורה של סינגלטון המחלקה מחזיקה אינסטנס של עצמה וכאשר נוצרת בפעם הראשונה יש Lock על היצירה של המחלקה והMember מהסוג של המחלקה מתעדכן להיות האובייקט שיצרנו ובכך נמנע יצירה של יותר מאובייקט אחד.

* Sequence Diagram

תמונה שמכילה טקסט, תרשים, מקביל, צילום מסך

התיאור נוצר באופן אוטומטי

* Class Diagram

תמונה שמכילה טקסט, תרשים, קו, מקביל

התיאור נוצר באופן אוטומטי