תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

- **סינון פוסטים/צ'ק-אין ע"פ זמן**
הפיצ'ר מהווה סינון לנוחיות המשתמש של הפוסטים/צ'ק אין ע"י בחירת המשתמש (כל הזמנים, יום נוכחי, חודש נוכחי, שנה נוכחית).

תוצאת החיפוש תחזיר רשימת פוסטים/צ'ק אין בהתאם לבחירת המשתמש , ובנוסף תאריך פרסומם.

  - הסינון של הפוסטים כולל **הצגת הפוסט הפופולרי ביותר בהתאם.**
  - **בצ'ק אין ניתן גם לראות מיקום על גבי google maps** בעת בחירת הפריט.

מחלקות הקשורות לפיצ'ר:

  - ממשק משתמש: FormPosts, FormCheckins תחת FaceBookApp.UI.
  - החלק הלוגי: eNumFilter, FacebookObjectFilter<TTT> where TTT : Post תחת FaceBookApp.Logic.

- **YouTube Music:**
הפיצ'ר מציג למשתמש את דפי המוזיקה שלו ותמונותיהם ובהתאם לבחירתו יפתח דפדפן המציג לו את תוצאות חיפוש של דף המוזיקה אותו הוא בחר והוא יכול לשמוע את השירים של אותו אומן.

מחלקות הקשורות לפיצ'ר:

  - ממשק משתמש: FormMusic תחת FaceBookApp.UI.

- **Active Friend:**

בממשק המשתמש מופיע רשימת החברים, ע"י בחירת חבר מהרשימה מופיע פאנל מתחת לרשימה המציג את תמונת החבר והאם החבר פעיל, הבדיקה מתבצעת ע"י משיכת הפוסט האחרון של החבר ובדיקת תאריך פרסומו בהשוואה לחודש הנוכחי, בנוסף תוצג גם תאריך פרסום הפוסט האחרון.

מחלקות הקשורות לפיצ'ר:

  - ממשק משתמש: FormMain תחת FaceBookApp.UI.

- **חיפוש בפוסטים של החברים ע"י טקסט** (פיצ'ר שנוסף בעקבות הערה בתרגיל הקודם)
הפיצ'ר נותן אפשרות למשתמש לחפש ע"י מילת מפתח פוסטים של חבריו , תוצאת החיפוש תחזיר רשימת פוסטים בהתאם למילת המפתח.

מחלקות הקשורות לפיצ'ר:

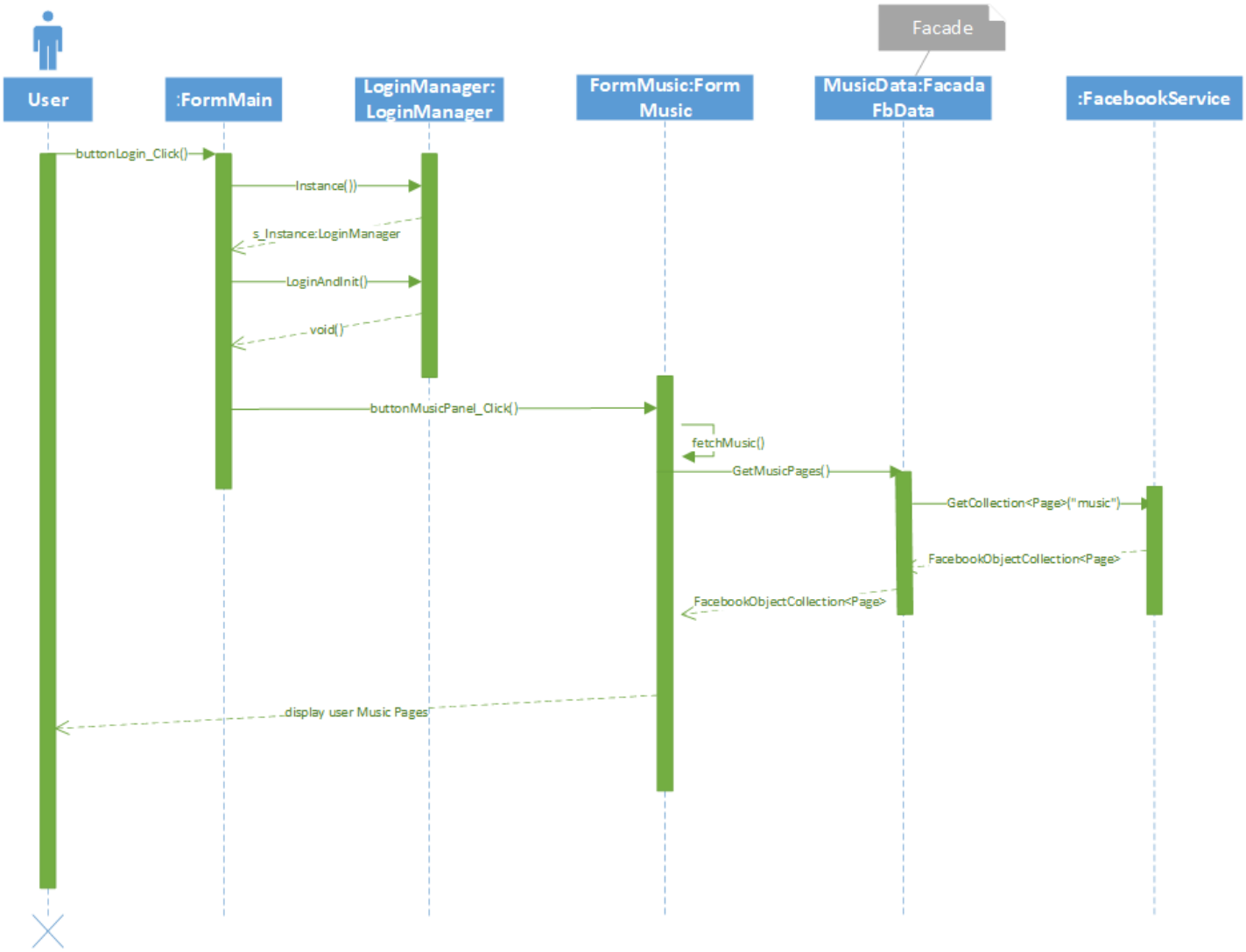
  - ממשק משתמש: FormSearchPosts תחת FaceBookApp.UI.
  - החלק הלוגי: SearchFriendsPosts.

תבנית מס' 1 – [Facade]

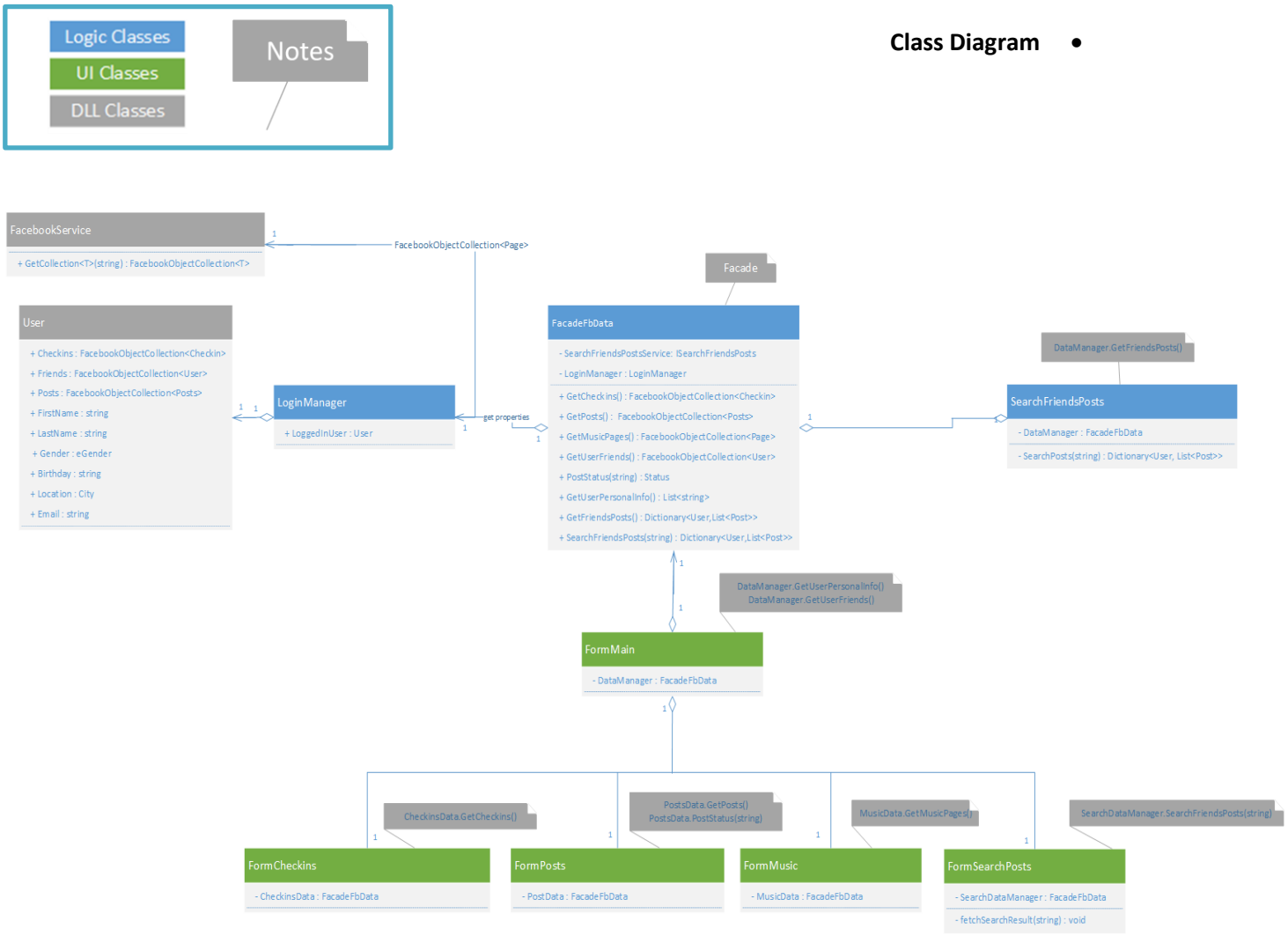
- **סיבת הבחירה / שימוש בתבנית:**
  בחרנו להשתמש בתבנית עיצוב זו על מנת לרכז בה את הפונקציונליות של המערכת כתתי מערכות כגון: שליפת נתונים על מידע אישי של היוזר, שליפת פוסטים של החברים. זאת בנוסף כי Façade עוזר לנו ביצירת ממשק פשוט למשתמש (המתכנת), והסתרת מידע לא הכרחי (אנקפסולציה).
  מדובר בFacade שקוף שכן הקליינט יכול לגשת ישירות לחלק הלוגי ולעשות בו שימוש, אך אנו עוטפים אותו כך שיהיה דרך נוחה יותר למשתמש.

- **אופן המימוש:**
  יצרנו מחלקה בשם FacadaFbData שמרכזת בה את הפעולות הלוגיות של האפליקציה ונעזרת בLoginManager בפנייה לUser בשביל טעינת המידע בFaceBookService, לטעינת דפי המוזיקה ובISearchFriendsPosts בשביל ביצוע תהליך החיפוש בפוסטים של החברים.
  מחלקות הUI פונות לFacadaFbData, שמאותחל בFormMain ומועבר כרפרנס לאחרים, לצורך פעולות הלוגיקה.

- **Sequence Diagram** (בחרנו להציג את התהליך כאשר הוא עובד מול music)



- **Class Diagram**

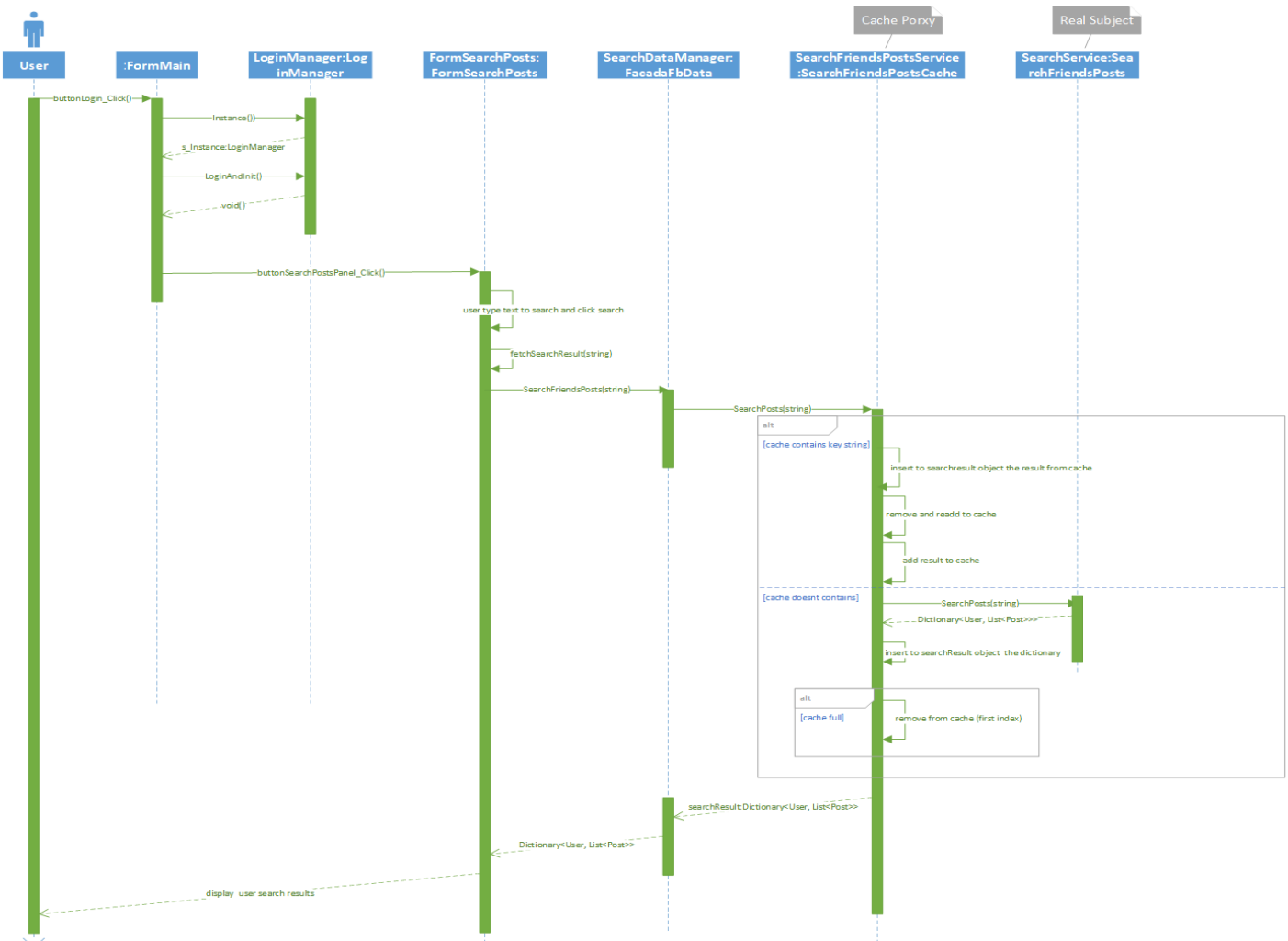


תבנית מס' 2 – [ProxyCache]

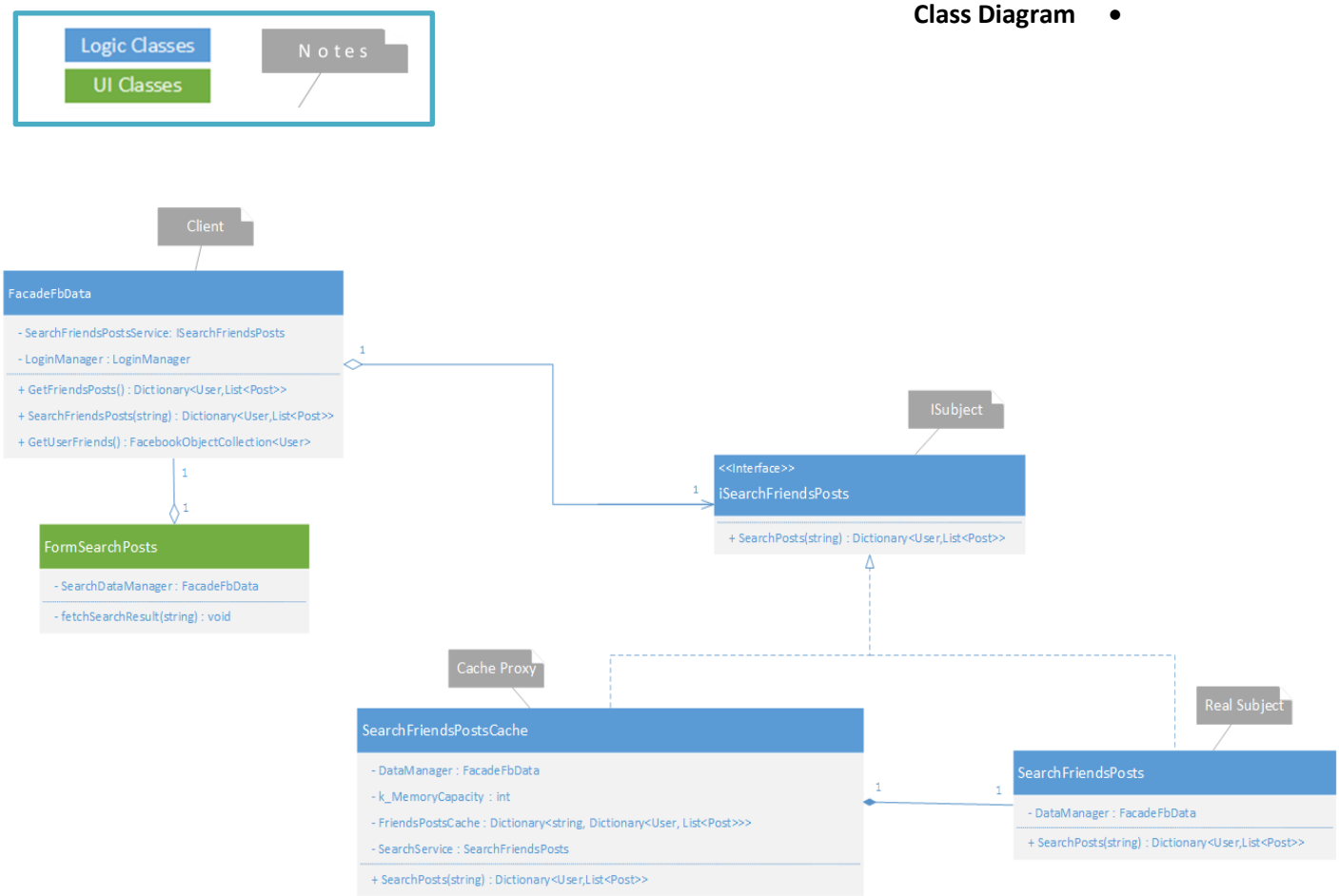
- **סיבת הבחירה / שימוש בתבנית:**
בחרנו להשתמש בתבנית זו מכיוון שפעולות החיפוש לוקחות זמן ומשאבים ועל מנת לייעל זאת ניתן לשמור מספר מסויים של תוצאות החיפושים אחרונים ולצמצם את זמן ההמתנה עבור משתמש האפליקציה במקרים שהבקשה כבר נמצאת בזכרון.

- **אופן המימוש:**
מימשנו את תבנית עיצוב זו ע"י יצירת ממשק ISearchFriendsPosts ושתי מחלקות אשר ממשות אותו :
SearchFriendsPosts - משמשת כReal Subject אשר במחלקה זו מבצעים חיפוש של פוסט ע"י מילת מפתח שהעביר המשתמש.
SearchFriendsPostsCache – משמשת כ proxy ומחזיקה רפרנס לsearchFriendsPosts ויכולה להחזיק נתונים של 4 (ניתן לשינוי) חיפושים אחרונים
מבנה הנתונים השתמשנו הוא מילון אשר מכיל חברים כמפתח ורשימת פוסטים כערך `Dictionary<User, List<Post>>`
מחלקת FacadeFbData מחזיק רפרנס לאינטרפייס ISearchFriendsPosts אשר מאותחל כמחלקת הפרוקסי וכאשר המשתמש רוצה לבצע חיפוש נקראת המתודה `SearchPosts` דרך מחלקת הפרוקסי, שם לפני הקריאה למתודה המתאימה הנמצאת בReal Subject מתבצעת בדיקה מול הCache : האם זה נמצא כבר שם אם כן מחזיר את התוצאה מהcache במידה ולא הוא מקבל מ Real Subject את תוצאת החיפוש ומכניס לcache , במידה והזכרון מלא (עד 4 חיפושים) אנחנו מוחקים את האחרון שנעשה בו שימוש אשר יהיה תמיד במיקום הראשון כמו כן במידה ומתבצע חיפוש חוזר של משהו שקיים כבר בcache הוא מוחק ומוסיף אותו מחדש לראש הרשימה על מנת לשמור על סדר המחיקה

- **Sequence Diagram** (הערה: רלוונטי גם לאחד התהלכים של Facade)



- **Class Diagram**



תבנית מס' 3 – [Singleton]

- **סיבת הבחירה / שימוש בתבנית:**
בחרנו להשתמש בתבנית זו עבור המחלקה LoginManager אשר זוהי מחלקה שאחראית על ההתחברות של המשתמש לאפליקציה, שמירת הטוקן.
אובייקט מסוג זה באפליקציה שלנו נעשה בו יצירה רק פעם אחת בלבדכמו כן הוא משמש לפונקציות רבות של משיכת מידע על חברים,פוסטים, דפים ועוד , ולכן על מנת להבטיח שיהיה מופיע אחיד של מחלקה זו השתמשנו בsingleton .

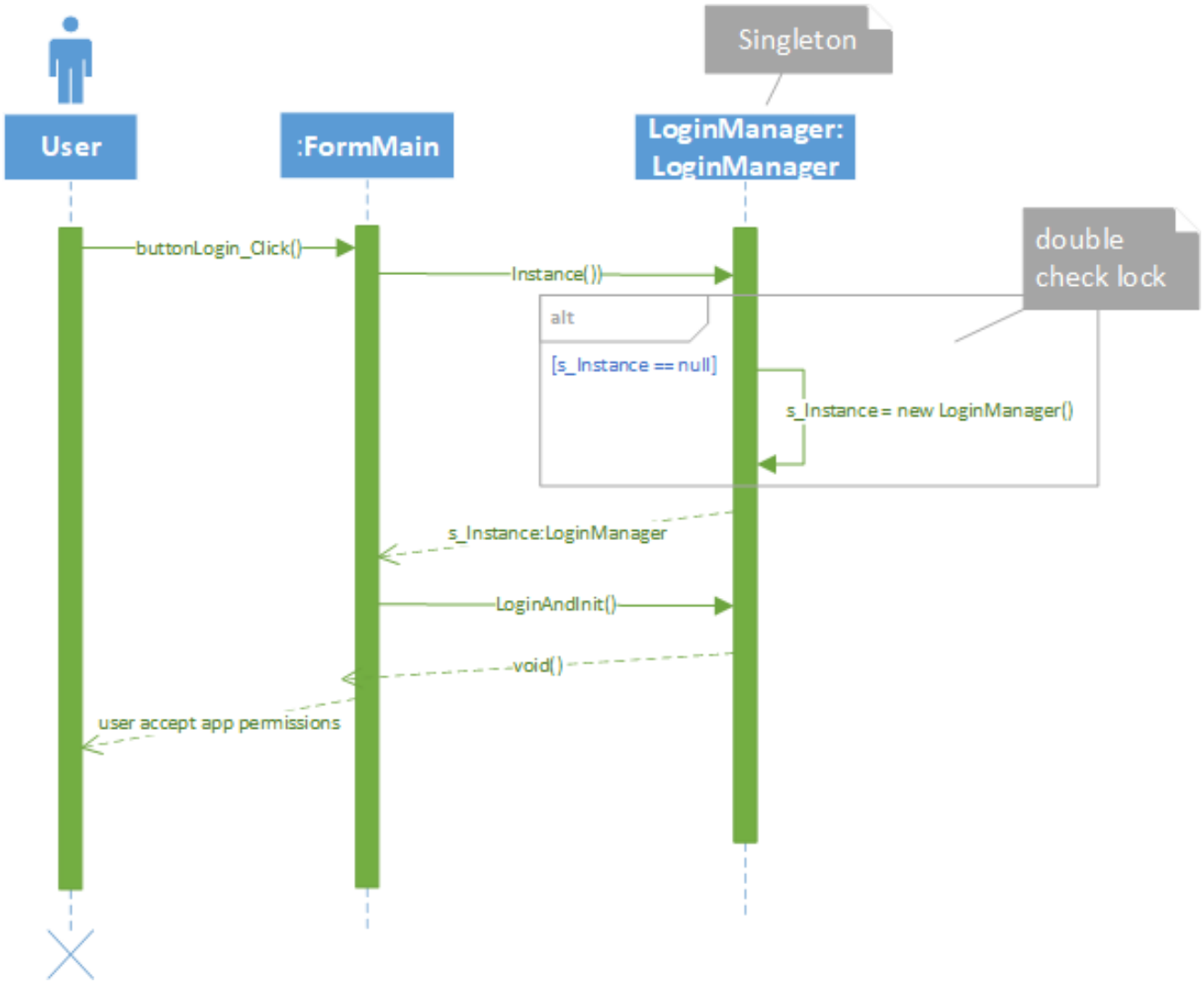
- **אופן המימוש:**
מימשנו את Singleton בתצורת ה double check lock , ניתן למצוא זאת בקוד **במחלקה LoginManager** , הפכנו את הctor לprivate והוספנו שתי משתנים למחלקה :
  - LoginManager s_Instance אשר ישמש לקריאת האובייקט באמצעות property
  - s_LockObj אשר משמש כדגל עבור פעולת הLOCK

על מנת לקרוא לאובייקט נעזר בproperty get בשם Instance אשר בודק האם קיים כבר אובייקט כזה ואם כן מחזיר אותו, אחרת ייצור אותו, על מנת למנוע מצב שהמופיע יווצר מספר פעמיים כאשר עובדים עם threads מימשנו את הget בשיטת double-check-lock כלומר בודק אם s_instance מאותחל אם לא מבצע Lock לבדיקה האם כבר thread אחר עבר לאחר מכן בודק שוב (double) האם הinstance מאתחל ואם לא אז קורא לprivate ctor אחרת מחזיר את האובייקט הקיים:

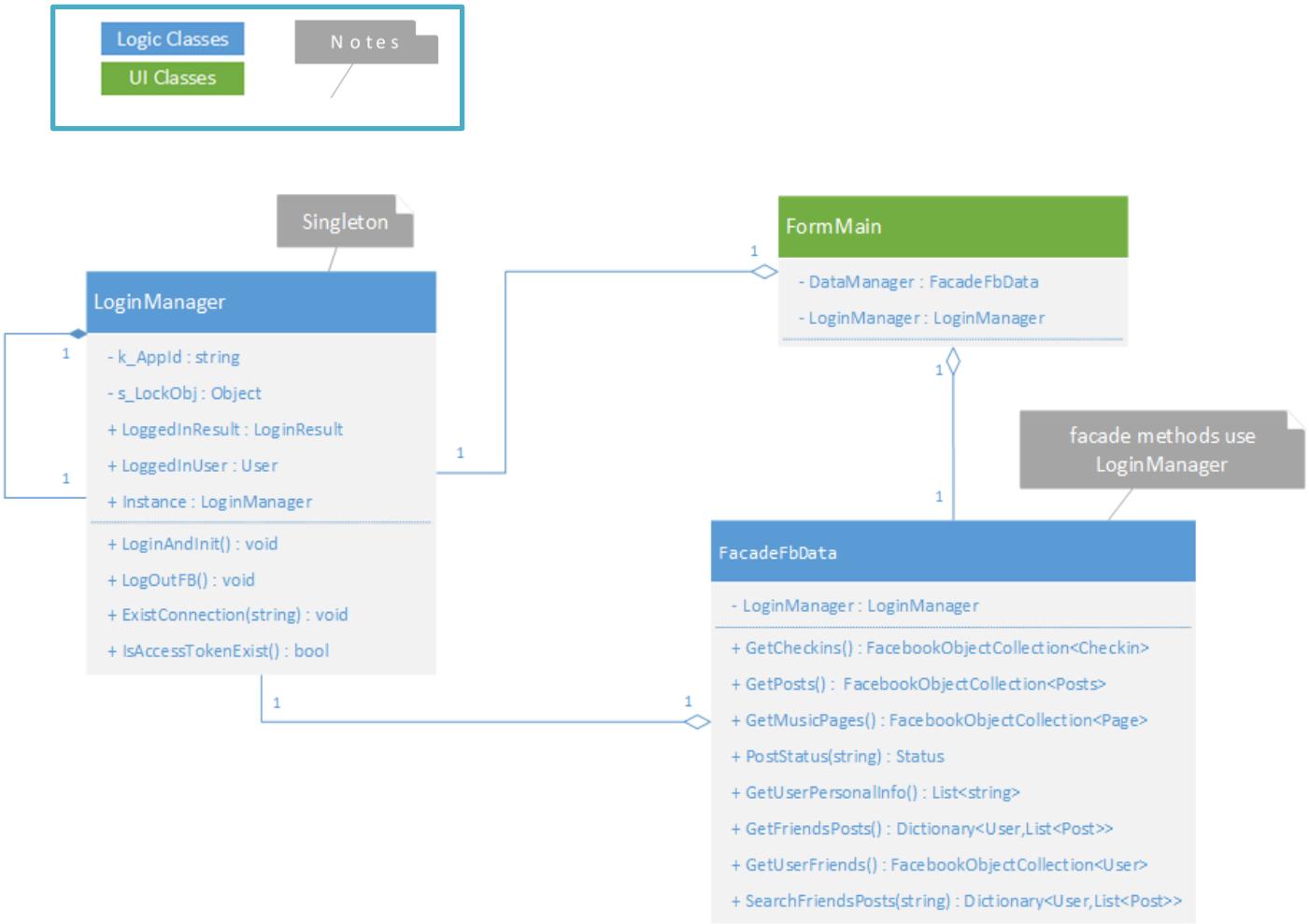
```
public static LoginManager Instance
{
    get
    {
        if (s_Instance == null)
        {
            lock (s_LockObj)
            {
                if (s_Instance == null)
                {
                    s_Instance = new LoginManager();
                }
            }
        }

        return s_Instance;
    }
   }
```

- Sequence Diagram

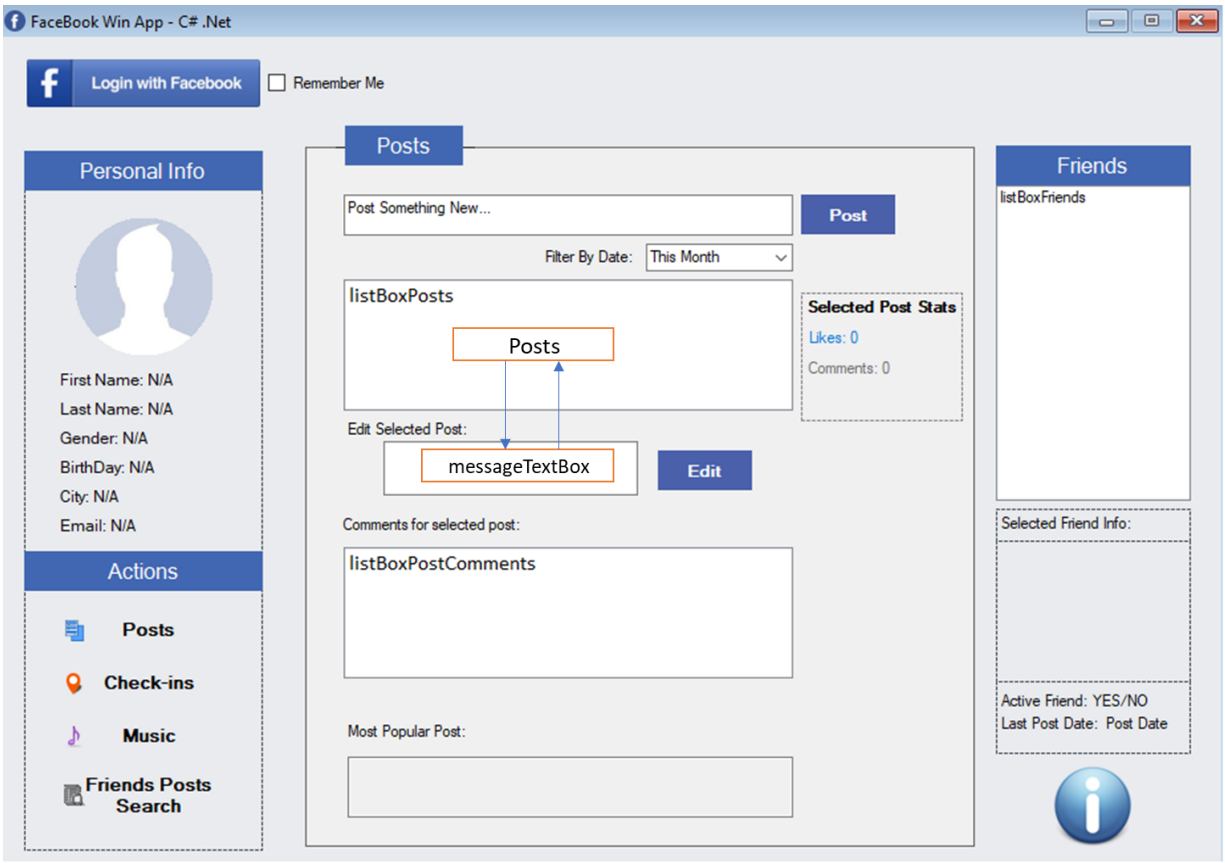


- Class Diagram



Two Way DataBinding

- **מקום השימוש בקוד במחלקת FormPost** עם הצגת הפוסטים בlistBoxPosts כאשר הDataSource הוא הפוסטים של המשתמש (אחרי שעברו סינון)
- המשתנה Message של Post מקושר אל התיבת טקסט messageTextBox וכך יכול המשתמש לערוך את הפוסטים שלו והעריכה מוצגת לו בחזרה בlistBoxPosts



עבודה אסינכרונית

על מנת לייעל את טעינת האפליקציה ואת נגישות המשתמש בזמן טעינת המידע עשינו שימוש בעבודה אסינכרונית במספר מקומות בקוד:

- FormMain:
  - עבור הטעינה והצגת מידע המשתמש (fetchUserInfo) באמצעות הרצתו בThread חדש ושימוש בInvoke על מנת להמנע מCross-Thread
  - עבור טעינה והצגת של רשימת החברים (fetchFriends) הרצתו בThread חדש ושימוש בInvoke בהתאם
- FormMusic:
  - טעינה והצגת רשימת דפי המוזיקה של המשתמש (fetchMusic)
- FormCheckins & FormPosts:
  - טעינת רשימת הפוסטים/צ'ק-אין של המשתמש מה-api וסינונם ע"פ תאריך (fetchCheckins/fetchpost) בהרצת Thread חדש (נמצא בתוך המתודה OnShown) ושימוש בInvoke בהתאם
- FormSearchPosts:
  - שימוש בThread חדש בשביל ביצוע החיפוש בפוסטים של החברים (fetchSearchResult)