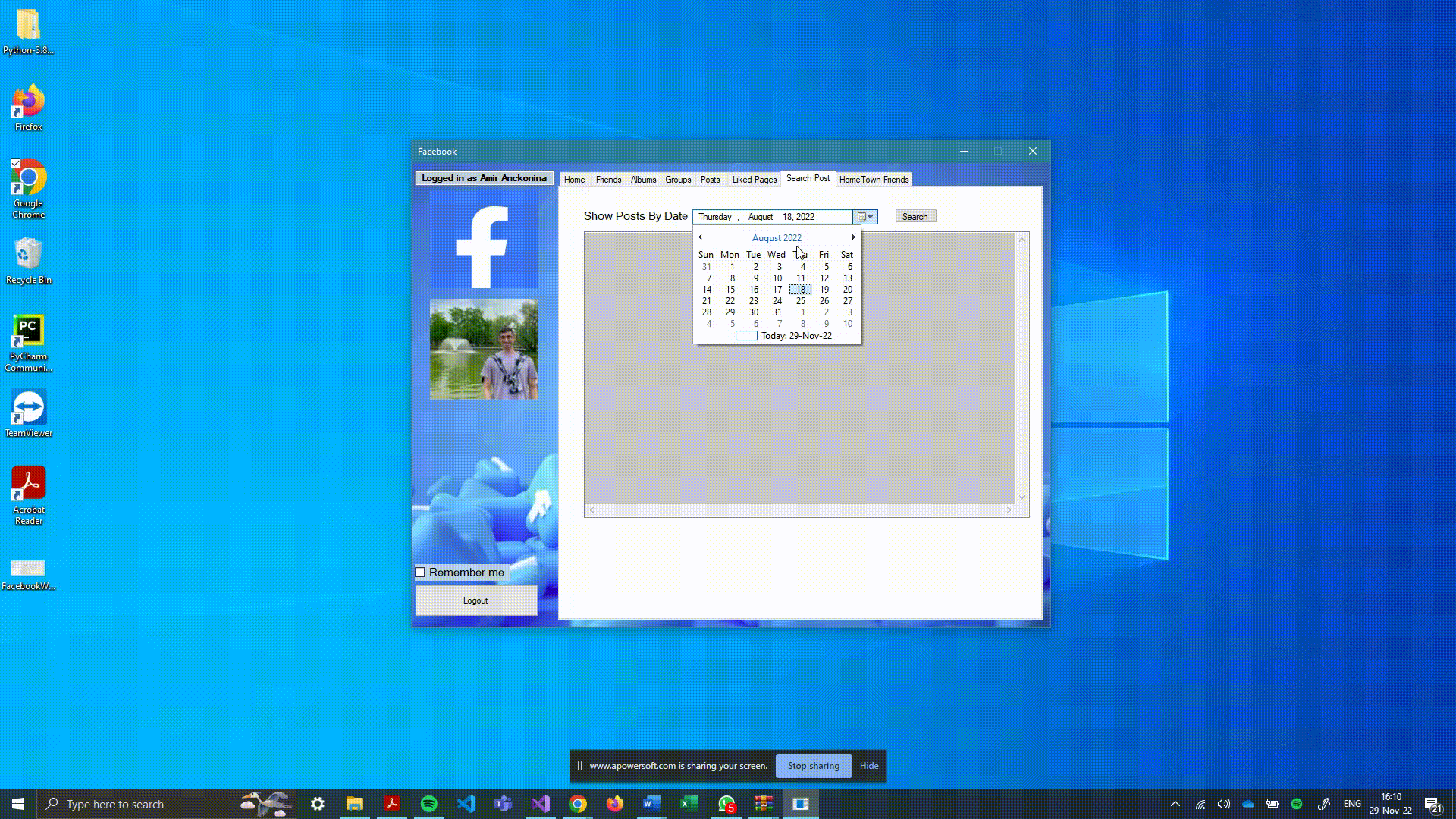
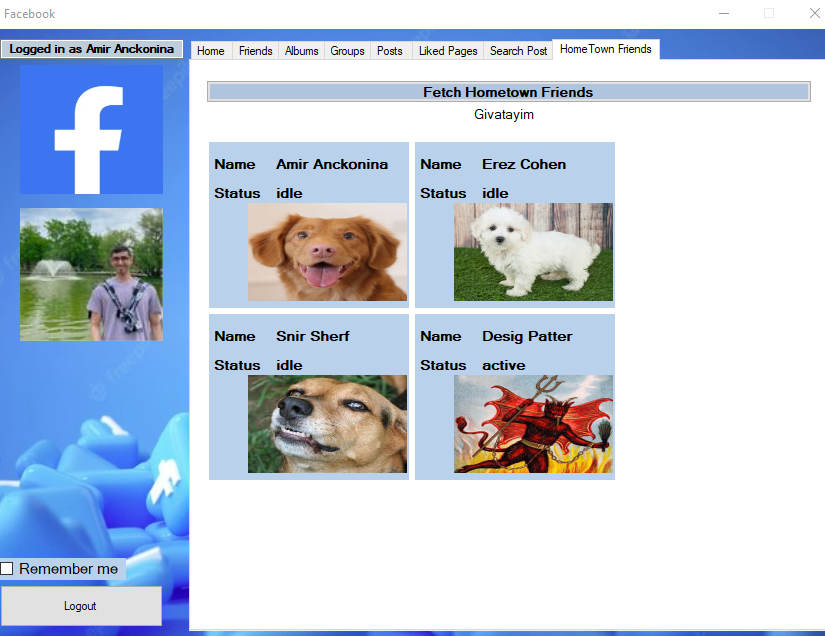
### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

* **Search Post By Date**
  + הפיצ'ר הנ"ל נותן למשתמש את היכולת "לפלטר" פוסטים שפירסם בפייסבוק על בסיס תאריך.
  + המשתמש לוחץ על בחירת תאריך ונפתח עבורו תפריט נוח ואינטואיטיבי, בו ניתן לראות לוח שנה ולנוע בקלות בין שנים, חודשים וימים.
  + לאחר הבחירה, המשתמש לוחץ על Search וכל הפוסטים מהתאריך המצוין יופיעו.

****

* **Hometown Friends**
  + הפיצ'ר הנ"ל מאפשר למשתמש לקבל תצוגה נחמדה של חבריו בפייסבוקאשר מאותו עיר.
  + היות והגישה דרך הAPI חסומה, מימשנו Dummy Data בו רשימת חברים ולכל חבר יש רשומה של העיר בה הוא גר.
  + כמו כן, אנו מציגים למשתמש את סטטוס החבר כרגע באפליקציה.

****

### תבנית מס' 1 – Singleton

* סיבת הבחירה / שימוש בתבנית:

בחרנו לממש את המחלקה AppManager כמחלקה סינגלטונית כיוון שלאפליקציה שלנו נדרש אך ורק מנהל אחד ויחיד אשר יכיל בתוכו את ה Form's השונים ביניהם LoginForm, AppSettingsForm.במידה והאפשרות ליצור 2 ומעלה instances של המחלקה הייתה קיימת, זהו פוטנציאל לבאגים כיוון שיחד איתה יווצרו הפורמים השונים אשר המחלקה מכילה, אין זה הגיוני להחזיק כמה מופעים של SettingsForm כמו כן LoginForm. בנוסף זהו בזבוז משאבים.

* אופן המימוש:

יצרנו מחלקה – AppManager שמטרתה לנהל את הform's השונים במערכת ולהגביל את מספר מנהלי האפליקציה ל1.

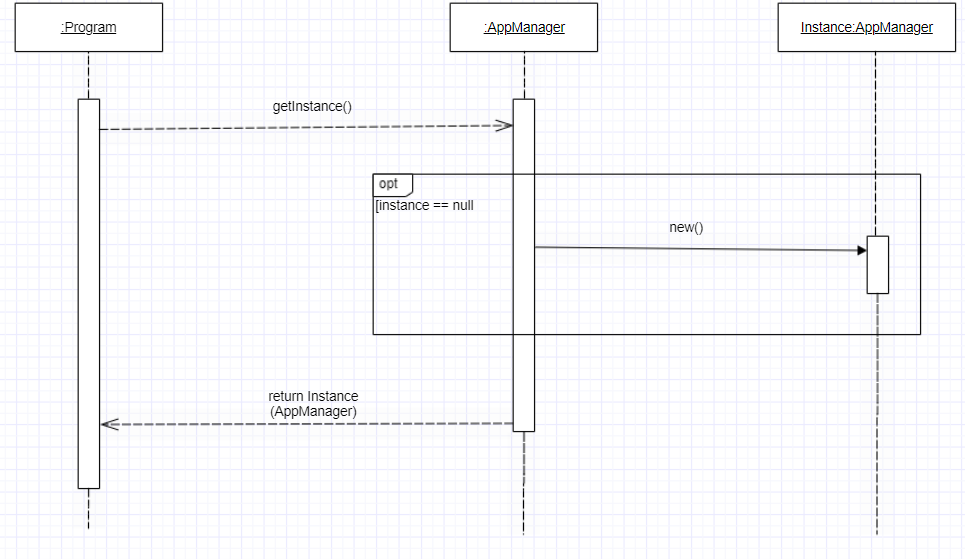
Access modifier של המחלקה שונה ל private. המחלקה הינה Sealed כי לא נרצה שתהיה אפשרות לרשת מהמחלקה (כי במידה וכן אז כשניצור אובייקט מהמחלקה היורשת אז נפעיל את הבנאי של הנורש והוא private).

המחלקה מחזיקה שדה סטטי –

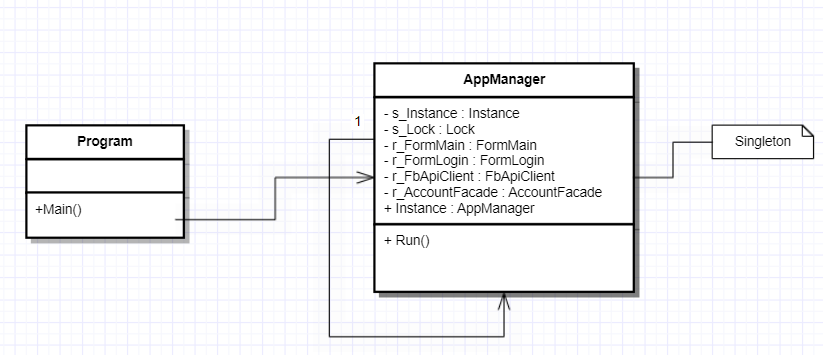
private static AppManager s\_Instance = null;

שדה זה מאותחל באופן Lazy רק במידה ונוצרת בקשה לקבל את הinstance.  
היווצרות האובייקט נוצרת בתוך getter של הפרופרטי Instance באמצעות שיטת double-check locking מה שהופך את היצירה ל thread-safe.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Facade

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### תבנית מס' 3 – FactoryMethod

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### עבודה אסינכרונית -

כאשר מתבצעת התחברות של המשתמש לאפליקציה, המערכת מבצעת קריאות לשרתי פייסבוק על מנת להביא את כל המידע הדרוש עבור אותו משתמש (פוסטים, תמונות, רשימת חברים, דפים אהובים ואלבומים).  
קריאות אלו דורשות זמן ומשאבים משום שמדובר בפניות לשרת חיצוני, ועד שהן לא מסתיימות האפליקציה נראת כ"תקועה", היא אינה רספונסיבית ולא ניתן להשתמש בממשק המשתמש.  
על מנת לקצר את זמני הטעינה של האפליקציה ולשפר את חוויית המשתמש, העברנו את הטיפול בהבאת המידע לטרדים נוספים.  
דבר זה מאפשר לאפליקציה להיות תגובתית (responsive) גם כשהמערכת עדיין לא סיימה להביא את כל המידע ובנוסף מקצרת את זמני הטעינה של האפליקציה. עבור כל שינוי ברכיבי ה UI מהטרדים שיצרנו, השתמשנו במטודה Invoke הנמצאת על כל אחד מהפקדים, ובכך עדכון המידע מתבצע דרך הטרד שאחראי על ה UI.  
  
השינויים הללו נמצאים במחלקה FormMain בבמטודות הבאות:

initBasicUserInfo(),initForm(),initLikedPages(),initAlbums(),initFriends(),

initGroups()

### עבודה עם Data Binding –

בקוד שלנו עשינו שימוש ב Data Bindingבtab אשר מציג את ה posts של המשתמש באופן הבא:   
המערכת מפעילה את הטופס הראשיForm Main , אשר שולח בקשה לשרת פייסבוק ובין היתר מחזיר collection של post. אנו עוברים על רשימת הפוסטים ויוצרים רשימה חדשה של אובייקט שיצרנו PostDTO. יצרנו postDTOBindingSource. במתודה initPosts() הגדרנו את DataSource להיות רשימת ה postDTO אשר קיבלנו. בנוסף, הגדרנו את הproperty Data Source של postListBox להיות postDTOBindingSource. וכך יצרנו data binding.

התהליך:

1. נוצרת בקשה לשרת לקבל רשימת פוסטים
2. postDTOBindingSource מתמלא בנתונים
3. postListBox מתמלא בנתונים מתוך postDTOBinding Souce
4. postPanel (חלונית בצבע תכלת) שואבת נתונים משתנים כאשר index של listbox משתנה. (הפקדים שנמצאים בתוך postPanel מקושרים ל postDTOBindingSource.