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

* פיצ'ר מזג אוויר: מביא אינפורמציה על מזג האוויר, דרך API בשימוש בJson.
* פיצ'ר מניות מביא אינפורמציה על מניות , דרך API בשימוש בJson.
* יצירה\עריכת פוסטים מוכנים מראש:

המשתמש יכול ליצור פוסטים מוכנים מראש ב editor ,לערוך אותם,למחוק, להשתמש בהם כtemplate ולפרסם בפוסט. ההודעות המתאימות ישמרו על קובץ מקומי במחשב.

* פיצ'ר לסידור פוסטים שפורסמו לפי הא"ב.
* **יצירת יום הולדת לחבר (חדש**): אפשרות ליצור יום הולדת לחבר קרוב ויום הולדת לחבר רחוק על פי בחירת המשתמש. (פרסום הודעת מזל טוב ואלבום תמונות משותף – עבור חבר קרוב ועבור חבר רחוק יהיו 2 פוסטים שונים . עבור החבר הקרוב יותר תכתב ברכה אישית יותר, ועבור החבר הרחוק תכתב הודעה שאינה אישית). בנוסף הופסנו באיזור החברים באפליקציה לוח שנה משופר המשמש להצגת יום ההולדת הלועזי והעברי של החבר שנבחר מרשימת החברים בlistbox . לוח שנה זה מאפשר קבלת אינפורמציה אודות תאריכים עבריים על ידי בחירת תאריך מסוים. לועזי בלוח).
* הוספנו לפרוייקט אופציית Dark Mode כאשר המשתמש מסמן ב"צ'ק בוקס" אופציית זאת המסך הופך לכהה יותר

### תבנית מס' 1 – Template Method

* סיבת הבחירה / שימוש בתבנית:  
  השתמשנו בתבנית זאת על מנת למממש בפיצ'ר Dark Mode . בפרוייקט יש לנו מחלקה אבסטרקטית ,בעלת פונקציונליות של יצירת ערכת נושא וקיימות 2 מחלקות שצריכות לממש את הפונקציונליות הזאת באופן דומה אך לא זהה.המחלקה האבסטרקטית משאירה "חורים" עבור הבנים שירשו אותה למלא את "החורים" האלה בפונקציונליות קונקרטית.

פונקציונליות זו במחלקה האבסטרקטית משמשת כtemple method.

* אופן המימוש:   
  השחקנים: ThemeMaker היא המחלקה האבסטרקטית המחזיקה את מתודת התבנית MakeTheme (יוצרת את ערכת הנושא באופן כללי ואבסטרקטי) יחד עם ClearSpeisificTheme ("מנקה" את ערכת הנושא הספיצפית) וIntializeSpesificTheme (המאתחלת את ערכת הנושא) .

ThemeControlMaker וThemeComponentMaker הן מחלקות קונקרטיות היורשות מהמחלקה האבסטרקית.

ThemeControlMaker משתמשת במתודות שהוזכרו למעלה עבור איתחול ערכת הנושא עבור אותו "פקד" ספיציפי ואילו ThemeCompnentMaker מאתחלת עבור כל בניה של אותה קומפוננטה.

מתודת "ניקוי" עבור ThemeControlMaker מאתחלת את הפקד לערכת הנושא (הצבעים ) המקוריים ואילו ThemeComponentMaker עבור כל בניה.

MakeTheme עבור מחלקת הרכיב, מאתחלת את הפקד בצבעים הנבחרים (עם מתודה נפרדת של איתחול באיטרצייה) ואילו עבור הפקד רק עבור אותו פקד.





### תבנית מס' 2 - Visitor

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בתבנית זאת עבור פיצ'ר הdark mode .

במערכת שלנו יש לנו 2 רכיבים שעובדים עם הdarkmode שרוצים להימנע משיכפול קוד ע"י קומפוזיציה.

רצון ליצור ערכות נושא – צבעים הפקדים ,רקעים וכו' . יש לנו עבור פקדים (כפתורים,listview ,textbox וכו) רצון לשנות לdark mode ולשנות את ערכת הנושא. ויש לנו עבור רכיבים (panel,pane בעל ילדים (שהם פקדים) רצון לשנות את ערכת הנושא. המימוש הוא לא זהה עבור פקדים ועבור רכיבים ואנו רוצים להימנע משיכפול קוד (וכפי שכבר למדנו, ירושה היא דרך לא נכונה) ולהשתמש בקומפוזיציה "רגילה" תגרום לנו להכרות ציקלית . (הפקדים יכירו את מחלקת הtheme והtheme יאלץ להכיר את הפקדים\רכיבים.

בנוסף לרצון לא לשכפל קוד, שלא תהיה הכרות ציקלית, בצורה זו לא נוכל לעשות reuse למחלקה כי היא צריכה להכיר באופן קונקרטי את הרכיב המסויים ולכן לא להיות כללית.

לכן השתמשנו בinterface אשר יהיה themable, וכל כל רכיב ופקד יממשו את הממשק (ובעצם יהיו יצור מסוג זה). והמחלקות הקונקרטיות ידעו לעבוד עם יצור מסוג זה בלבד (ולא קונקרטי עם כפתור,טקסטבוקס וכו').

* אופן המימוש:  
  ה"שחקנים" בתבנית הזו : - IThemeableהממשק אשר מחזיק Thememaker עם get בלבד.

ThemeMaker היא מחלקה אבסטרקטית אשר מממשת את הIThemeable ומחזיקה את הthemeable מסוג הממשק

ממחלקה זו יורשות ThemeComponentMaker וThemeControlMaker .

כל קומפוננטה וכל פקד (לדוגמא LabelTheme) תממש את IThemeable ואת פונקציונליות הThemeControlMaker באופן קונקרטי (תהפוך את הטקסט בצבע המסויים של הdarkmode ,תשנה את הרקע וכו')





### תבנית מס' 3 – Builder

* הפיצ'ר של "פרסום ברכת יום הולדת ויצירת אלבום עושה שימוש בתבנית זו. הסיבה שהשתמשנו בפיצ'ר זה משום שהוא מורכב מכמה חלקים ויש חשיבות לסדר הפעולות. אם נרצה בעתיד להוסיף עוד משימות לבנייה,נוכל לשמור על רצף לוגי נכון ולוגיקת הבנייה מתבצעת ברכיב ייעודי שמיועד לכך. לכל סוג יום הולדת נוכל ליצור בילדר ייחודי ופולימורפי ולהרחיב את הפעילות בעתיד.

* אופן המימוש:

השחקנים: BirthDayManager אשר בתפקיד ה Director ומחזיק בקומפוזיציה Interface של BirthdayActivityBuilder.

הדיירקטור אחראי על סיפוק פעילות יום ההולדת על ידי שימוש בבילדר המתאים לפי בחירת המשתמש. הבנייה תתבצע בסדר של בניית סטטוס מזל טוב ולאחר מכן בניית אלבום משותף. הדיירקטור עושה שימוש בבילדר ואינו אחראי על הפעולות השונות אלא רק על הסדר אשר יתבצעו. כך שאם נרצה בעתיד לשנות לוגיקה של אחת מפעולותיו של הבילדר נעשה זאת דרך הבילדר המתאים.

BirthdayActivity - זהו האובייקט שעלינו ליצור ולקבל.

IBirthDayActivityBuilder - Interface אשר מצריך מימוש של שתי מתודות :   
 BuildStatus - פרסום סטטוס "מזל טוב" על ה"קיר" של החבר.  
BuildAlbum - בניית האלבום המשותף.

את IBirthDayActivityBuilder ממשים שני Concreate Builders ובונים את האלבום והסטטוס בצורה היחודית להם:  
CloseFriendBirthdayActivityBuilder - עבור החבר הקרוב יותר תכתב ברכה אישית יותר ואלבום אישי יותר.  
וFarFriendBirthdayBuilder - ועבור החבר הרחוק תכתב הודעה שאינה אישית ואלבום שאינו אישי.





### עבודה אסינכרונית

בתוכנית שלנו השתמשנו בעבודה אסינכרונית בכמה שימושים:

בשליפת מידע מהפייסבוק של המשתמש בתוכנית בכלל הפיצ'רים:

Fetch Albums

Fetch Groups

Fetch Friends

Fetch Liked Pages

Fetch Posts

ע"י יצירת thread חדש לכל אחת מהמתודות אשר שולפות את המידע

וInvoke עבור פעולות של הפקדים שלא מאפשרים גישה עבור Thread שהוא לא ה Main Thread שיצר אותם.

הסיבה שהשתמשנו בעבודה אסינכרונית עבור פעולות אלא,משום שלא רצינו "לתקוע" את הUI עבור המשתמש כאשר אנחנו מבקשים שליפה משרת הפייסבוק, ושהשתמש יוכל לעשות עוד פעולות עד ש הUI יוצג עבורו.

### עבודה עם Data Binding

בתוכנית שלנו השתמשנו בdata binding עבור הצגת הקבוצות של המשתמש והתמונת נושא של הקבוצה.

לצורך כך השתמשנו בהצגת הקבוצות של המשתמש.

השתמשנו ב DataBindingWizard בצורה אוטומטית כך שבחרנו להציג את התמונה של המשתמש כאשר הוא בוחר בlistbox של הקבוצות את הקבוצה.

השורת קוד היחידה שכתבנו הייתה במתודה fetchGroups והיא: groupBindingSource.DataSource = LoginFacade.LoginUser.Groups;

על מנת לקשר בין הdata source והנתונים של הקבוצה של המשתמש.

בנוסף גם בfetchFriends השמשנו בזה.