קורס בדוטנט תשע"ו
מיני פרויקט

תוכן עניינים

תוכן עניינים **1**
מבוא **2**
הנחיות להגשה **2**
תיאור הארגון **3**
תיאור ישויות המערכת **4**
תיאור חלקי הפרויקט בקצרה **5**
חלק א – הגדרת הפרויקטים עבור מודל השכבות ומימושן **6**
שלב א - ישויות 6
שלב ב' – DAL במימוש רשימות 8
שלב ג' – שכבת ה- BL 9
שלב ד' – שכבת ה- UI 11
חלק ב – עדכון שכבת ה- DAL **12**
מימוש באמצעות LINQ-To-XML 12
הגשות: **15**

מבוא

במהלך הקורס אנו נכתוב פרויקט שבאמצעותו נתרגל את עקרונות שפת #C, עקרונות ארכיטקטורה של תוכנה, ויצירת ממשקי משתמש באמצעות תשתית הפתוח המודרנית WPF.
הערה: עבודה פושרת ולא יצירתית תגרור בעקבותיה גם ציון פושר. יש להדגיש שחלק נכבד מן המבחן הסופי יתבסס על נושאים תכנותיים שתתבקשו להתמודד איתם במהלך התרגול. חשוב להדגיש שציון מתחת ל-85 בקורס זה כמו בכל קורס מעשי אחר איננו משמעותי בעיניהם של המעסיקים בתעשייה...

העבודה תתבצע אך ורק בזוגות ! לא בבודדים ולא בשלישיות ! כל אחד מהשותפים חייב להיות שותף מלא בכל אחד מן השלבים ! הפרויקט חייב לרוץ על מחשב של כל אחד מן השותפים. לא תתקבל טענה ששותף אחד עשה שלב מסוים ואלו שותף אחר עשה את החלק האחר ! במקרה כזה הציון יהיה פרופורציונלי למספר השלבים שנעשו ע"י כל אחד מן השותפים... לא ניתן לעבור לשותף אחר במהלך הסמסטר. לא ניתן לעבור במהלך הסמסטר מקבוצה לקבוצה. לא ניתן להיות שותף עם חבר שרשום בקבוצה שונה. בכל אחד מן המקרים האלה התרגיל ייפסל.

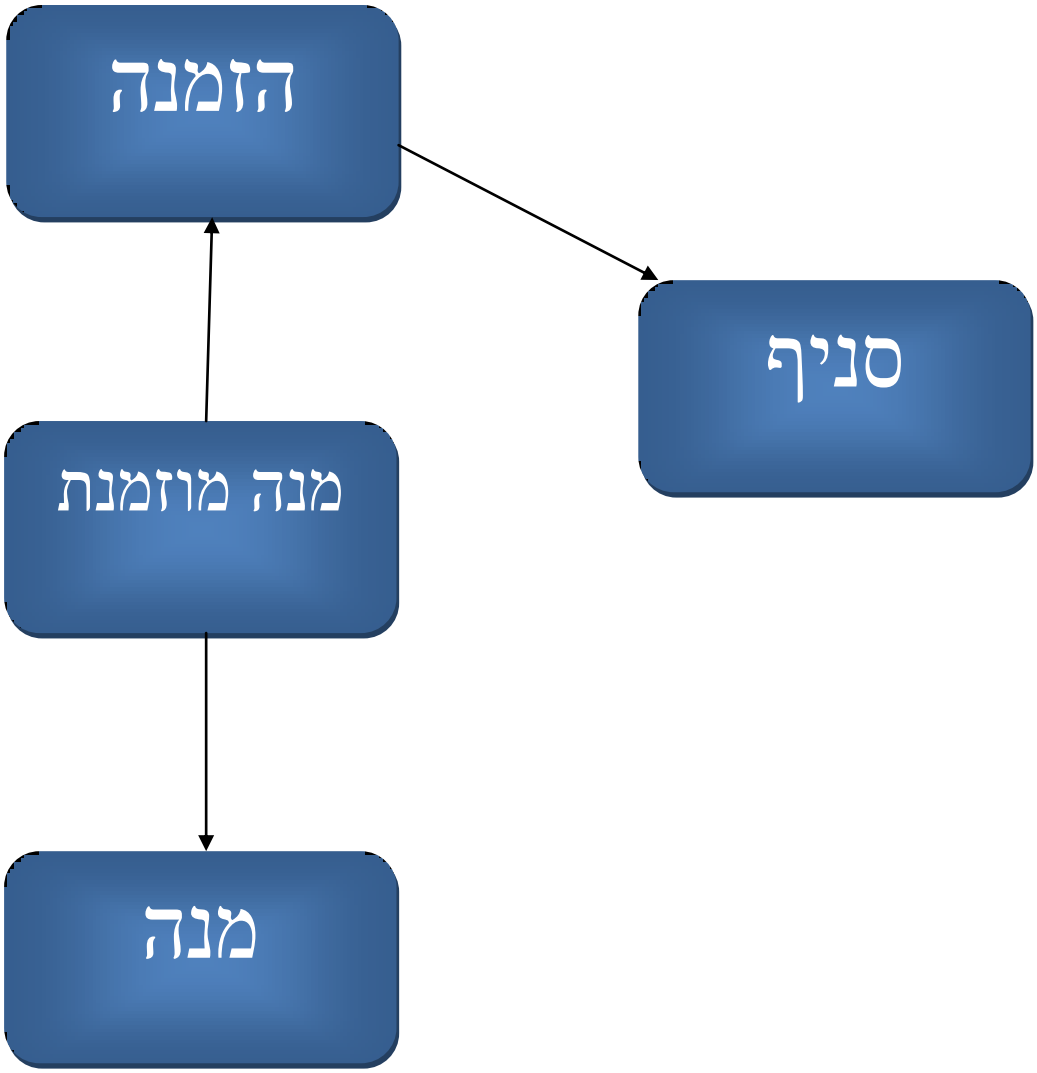
הבדיקה תיעשה בשלושה מועדים בלבד. כל שלב הגשה יימסר בתאריך היעד. השלבים ייבדקו בצורה כללית (בעיקר תיבדק הרצה תקינה, המרצה ישמור בקובץ אקסל הערות לגבי כל שלב כולל הערות איכות עבודה ועמידה מדויקת בלוח הזמנים הנדרש) ורק לבסוף עם הגשת השלב האחרון והמסכם תתבצע בדיקה יסודית של העבודה שתכלול גם הגנה של שני השותפים על העבודה כ"א בנפרד. הציון בהגנה יהיה שבר בין 0..1 ויוכפל בציון הכללי שישוקלל מבדיקת כל השלבים. הציונים הסופיים עבור כל אחד מהשלבים יינתנו רק בסוף הסמסטר. צין המעבדה ישוקלל עם ציון המבחן רק בתנאי שהציון במבחן הוא לפחות 60%.

הנחיות להגשה

העבודה בזוגות, יש להעלות קובץ zip עם הפתרון למודל (הכולל את כל תיקיית ה solution – ראו הנחיות בהמשך)
שם הקובץ יהיה dotNet5776_Project01_xxxx_yyyy
dotNet5776 = זה הקורס והשנה העברית
01 = זה השלב (1,2,3)
xxxx = זה 4 ספרות אחרונות בתעודת הזהות של בן הזוג הראשון
yyyy = זה 4 ספרות אחרונות בתעודת הזהות של בן הזוג השני
נא להקפיד על פורמט זה על מנת למנוע מצב של אי קבלת ציון על תרגיל מסוים.

תיאור הארגון

בפרויקט זה נכתוב מערכת (מוגבלת) לניהול מסעדה בעלת רשת סניפים בפריסה ארצית.
הלקוח מתקשר למרכז ההזמנות של הרשת. הלקוח נותן את פרטיו ומיקומו הנוכחי, את שם הסניף הקרוב למיקומו, את המנות שהוא מעונין להזמין ומספרן. לאחר ביצוע ההזמנה נשלחת הוראה ע"י מרכז ההזמנות לסניף לבצע את ההזמנה והארוחה נשלחת ללקוח באמצעות שליח (בשלב זה המערכת לא תטפל בשליחת ההזמנה בפועל)– **אז שיהיה בתיאבון!**



תיאור ישויות המערכת

הזמנה
מכילה פרטים כגון: מספר ההזמנה, תאריך ההזמנה, מספר הסניף שממנו מתבצעת ההזמנה, פרטי הלקוח כגון: טלפון, ככתובת מיקום נוכחי, רמת הכשר, מחיר כולל של ההזמנה.

מנה
מכילה פרטים כגון: מספר המנה, שם המנה, גודל המנה (יש מנה קטנה, בינונית, גדולה) מחיר מנה בודדת. הכשר..

סניף
מכילה פרטים כגון: מספר הסניף, שם הסניף, שם האחראי על הסניף, מספר העובדים בסניף, מספר טלפון של הסניף, כתובת הסניף, מספר השליחים העומדים לרשות הסניף בכל רגע נתון, רמת ההכשר של הסניף.

הזמנה-מנה
ישות המקשרת בין ההזמנה למנה ומכילה את הפרטים הבאים: מספר ההזמנה, מספר המנה, כמות המנות מאותו סוג.

הערה: הישות מנה מוזמנת תחבר בין מנה להזמנה. מאחר ומנה יכולה להופיע בהרבה הזמנות ובהזמנה יכולות להיות מספר מנות (מסוגים שונים) אז יש כאן רשימה מקשרת של NXM. אנחנו לא מעוניינים שבתוך ההזמנה תופיע רשימה פנימית וגם לא מעוניינים שבתוך ההזמנה יהיו מוכלים אובייקטים כלשהם אלא רק את ה- ID של האובייקטים. כלומר הדרישה היא ליצור "מחבר" שיקשר בין מספר ההזמנה למנה ויציין גם את מספר המנות מאותו סוג שנרשמו בהזמנה.

תיאור חלקי הפרויקט בקצרה

חלק ראשון:
נבנה את הפרויקט במודל השכבות, מקור הנתונים יופיע בפרויקט נפרד ויכיל רשימות של אובייקטים. הגישה לנתונים תתבצע באמצעות ה- dal וממשק המשתמש יהיה ממשק ויזואלי גרפי. תשתית הפתוח הרשמית של הממשק הוויזואלי הינה WPF.
(חלק זה יחולק לשני שלבים ולאחר כל שלב תתבצע הגנה, ראה בפנים.)

חלק שני:
נשנה את שכבת ה- DAL שייגש למקור נתונים שהפעם יתבסס על קובצי xml

חלק א – הגדרת הפרויקטים עבור מודל השכבות ומימושן

בחלק זה של הפרויקט נגדיר את כל אחת מן השכבות שיהוו יחד את המיני פרויקט שיאפיין את רשת הסניפים של המסעדה.

בשלב א' נגדיר את המחלקות של הישויות, כלומר את שכבת ה- BE.
בשלב ב' נגדיר את דרך העבודה מול הנתונים, כלומר את שכבת ה- DAL.
בשלב ג' נגדיר את הלוגיקה של המיני פרויקט. כלומר את שכבת ה- BL
בשלב ד' בחלק זה נממש את ממשק המשתמש של המיני פרויקט, כלומר שכבת ה- UI.

שלב א - ישויות

א. הגדר פרויקט **מסוג Class Library** בשם **BE** ובו תגדיר את המחלקות הבאות.חובה
להגדיר כל מחלקה בקובץ נפרד.
אנו נפרט שדות שיכללו במחלקות. כל אחד מוזמן וחייב להוסיף שדות כפי שנראה לו מתאים.

מחלקה בשם: Order שתייצג הזמנה ותכלול:

א. מאפיין שמציין את מספר ההזמנה. **(קוד רץ ייחודי בן 8 ספרות)**
ב. מאפיין שמציין את תאריך ההזמנה, מסוג תאריך.
ג. מאפיין שמציין את מספר הסניף שממנו מתבצעת ההזמנה.
ד. מאפיין enum שמציין את רמת ההכשר (רגיל, בינוני, מהודר)
ה. מאפיין שמציין את מספר השליחים הנוכחי שעומדים לרשות הסניף.
ו. מאפיין שמציין את שם הלקוח.
ז. מאפיין שמציין את כתובת הלקוח.
ח. מאפיין שמציין את כתובת מיקום הלקוח.
ט. מאפיין שמציין את מספר כרטיס האשראי שממנו נגבה הכסף של מחיר ההזמנה.
י. מאפיינים נוספים לפי הצורך.
יא. *ToString*

מחלקה בשם: Dish שמייצגת מנה ותכלול:

א. מאפיין שמציין את מספר הזיהוי של המנה.
ב. מאפיין שמציין את שם המנה.
ג. מאפיין שמציין את גודל המנה.
ד. מאפיין שמציין מחיר מנה בודדת.
ה. מאפיין enum שמציין את רמת ההכשר (רגיל, בינוני, מהודר)
ו. מאפיינים נוספים לפי הצורך.
ז. *ToString*.

מחלקה בשם Branch שמייצגת סניף ותכלול:

א. מאפיין שמציין את מספר הסניף.
ב. מאפיין שמציין את שם הסניף
ג. מאפיין שיציין את כתובת הסניף.
ד. מאפיין שיציין את מספר הטלפון של הסניף.
ה. מאפיין שמציין את שם האחראי על הסניף.
ו. מאפיין שיציין את מספר העובדים בסניף
ז. מאפיין שמציין את מספר השליחים הפנויים בכל רגע נתון..
ח. מאפיין enum שמציין את רמת ההכשר של הסניף (רגיל, בינוני, מהודר)
ט. מאפיינים נוספים לפי הצורך.
י. *ToString*.

מחלקה בשם: Ordered-Dish - שמייצגת את הקשר בין הזמנה למנה ותכלול:

א. מאפיין שמציין את מספר ההזמנה.
ב. מאפיין שמציין את מספר המנה.
ג. מאפיין שמציין את כמות המנות מאותו סוג מנה שהוזמן.
ד. מאפיינים נוספים לפי הצורך.
ה. *ToString*.

אופציונלי בלבד: אם רוצים ניתן להוסיף עוד ישות בשם: client אבל זה מחייב להוריד את פרטי הלקוח מההזמנה ולדאוג שההזמנה תכיל רק את מספר הלקוח.

<u>הערה: כל תקלה, כתוצאה מפעולה כלשהי תגרור אחריה טיפול בחריגות לפי מנגנון החריגות הקיים ב- #C. לדוגמא: ייתכן שהלקוח יבקש לבצע הזמנה בסניף שרמת ההכשר שלו לא שווה לרמת ההכשר שהוא דורש בהזמנה עצמה.</u>

<u>הערות:</u>

- יש להוסיף שדות ומאפיינים במידת הצורך.
- יש להימנע משימוש בפונקציות public במחלקות, הלוגיקה המרכזית מתבצעת בשכבת ה- BL כפי שבעז"ה יפורט בהמשך.

שלב ב' – DAL במימוש רשימות

הוסף פרויקט מסוג class library בשם DAL ובו בצע את הפעולות הבאות:

א. הגדר (interface) ממשק בשם Idal .
בממשק הנ"ל הגדר חתימה של פונקציות שימושיות עבור האפליקציה כגון:

- הוספת מנה.
- מחיקת מנה.
- עדכון פרטי מנה קיימת.

- הוספת סניף.
- מחיקת סניף.
- עדכון סניף קיים.

- הוספת הזמנה.
- מחיקת הזמנה.
- עדכון הזמנה.

- הוספת מנה_מוזמנת
- עדכון מנה_מוזמנת
- מחיקת מנה_מוזמנת

- קבלת רשימת כל ההזמנות
- קבלת רשימת כל המנות.
- קבלת רשימת כל הסניפים.

ב. צור פרויקט חדש בשם: DS והגדר בתוכו מחלקה בשם DataSource שתכיל את הנתונים (רשימות) של הישויות שלנו.
מחלקה זו תכיל 4 רשימות סטטיות של המחלקות הנמצאות ב- BE.
ג. הגדר מחלקה בשם: Dal_imp אשר מממשת את ממשק ה- Idal הנ"ל.
הפונקציות של המחלקה הזאת יעבדו מול האוספים מסוג list<> הנמצאים במחלקה DataSource.

הערה:
שכבת ה- DAL היא האחראית על כך שהמפתח בכל אחת מהמחלקות יהיה קביל, כך שבמקרה שהמספר הייחודי (שיתקבל בישות שמעוניינים להוסיף) הוא 0 שכבת ה- dal אחראית לתת קוד ייחודי בעצמה, במידה והמספר הייחודי (שיתקבל בישות שמעוניינים להוסיף) שונה מ- 0 עליה לבדוק שאינו קיים כבר.(ה- Dal אחראית לתת את המספרים הרצים)

שלב ג' – שכבת ה- BL

הגדר ממשק (interface) בשם: IBL שבו החתימות של פונקציות בדיוק כמו ב- IDAL ובנוסף חתימת הפונקציות הבאות:

- חישוב תשלום להזמנה ע"פ רשימת המנות בהזמנה .
- אין אפשרות לבצע הזמנה שעולה על תשלום מסוים.
- אין אפשרות להכניס להזמנה מנה שאיננה עם הכשר מסוים.
- באמצעות דליגט למצוא את כל ההזמנות שעונים על תנאי מסוים.
- רווחים ע"פ סוגי מנות. שימוש ב- Grouping
- רווחים ע"פ תקופות זמן. שימוש ב- Grouping
- רווחים מלקוחות ע"פ מקום מגורים. שימוש ב- Grouping .
- אין אפשרות להכניס לקוח מתחת לגיל 18.

חובה לעשות שימוש בכישרון היצירתיות שלכם ולהוסיף לפחות עוד 6 פונקציות שמתאימות להיות בשכבת ה-BL ועובדות על הנתונים המוחזרים מה- DAL.
הגדר מחלקה חדשה בשכבת ה- BL שתממש את הממשק IBL הנ"ל המימוש יכלול שימוש ב Linq to object וביטויי למבדא.

עדכון ל Linq-to-object

ממש את כל פונקציות ה- IDAL ופונקציות ה IBL שעדיין לא מומשו כך באמצעות Linq to object וביטויי למבדא.

יש לעשות שימוש בלפחות 4 ביטויי Linq בשכבת ה- BL- וה- DAL כל אחת. יש לעשות שימוש בלפחות 4 ביטויי למבדה בנוסף לביטויי Linq דלעיל.

<u>הערות:</u>

על שכבה זו לוודא שמתקיימים חוקים לוגיים בסיסים כמו לדוגמה:

- אם מוסיפים הזמנה מסוימת אזי המנה וכן הסניף שמוכלים בהזמנה אכן קיימים.
- אם מעוניינים לבצע הזמנה שצריך עבורה שליח ואין כרגע שליחים בסניף, ההזמנה לא תתאפשר.
- אם הלקוח מנסה לבצע הזמנה עם רמת הכשר מסויימת שלא קיימת בסניף שממנו הוא מבקש לבצע את ההזמנה, לא נוכל לבצע את ההזמנה.

כאן תתבצע הגנה באמצעות ממשק קונסול.

לשם הבדיקה של הדברים אנו ניצור פרויקט זמני בשם PL.
תוכל לממש אותו או בעזרת console application או ממשק גרפי WPF פשוט . אפשר בשלב זה לעשות הכל בחלון אחד.
בכל מקרה עליך לקרוא לפונקציות הנמצאות ב- BL ולבדוק אותן.

<u>הערה</u>: כדי לעבוד בשלבים מומלץ בהתחלה ליצור משהו שבודק את ה- BE אח"כ את הפונקציונליות ב- DAL כדי לראות שאכן זה עובד ואז לחבר את ה- BL.

שלב ד' – שכבת ה- UI

בשלב זה ניצור ממשק גרפי לפרויקט באמצעות תשתית הפתוח WPF.

ניצור פרויקט חדש מסוג : WPF
ונקרא לו PLForms.,
כמובן שיהיה עליך ליצור לו reference מתאים ל-BL וכן ל- BE .
עליך להגדיר ישות בשם: BL מטיפוס: IBL כפי שראית בדוגמה בהרצאה. לדוגמא:

```
IBL bl
 public Form1()
{
 bl = FactoryBL.getBL();
}
```

למעשה עליך לתכנן את המסכים אשר יקראו ויאפשרו למשתמש לגשת לפונקציונאליות הנמצאת ב- BL.

עליך לתכנן מסך לכל ישות בדומה למה שראינו בכיתה עם "הסטודנטים והקורסים", כאשר יהיו לפחות המסכים הבאים:

- מסך הוספת, עדכון, מחיקה של מנות.
- מסך הוספת, עדכון, מחיקה של סניפים.
- מסך הוספת, עדכון , מחיקה של הזמנות.
- בנוסף – לפחות עוד 3 מסכים המציגים שאילתות שמימשתם ב- BL.

התכנית תיפתח במסך ראשי המפנה לאפשרויות השונות. כלומר התכנית הראשית גם תגדיר את ה-BL ותעביר אותו למסך שנפתח, כל מסך נוסף שייפתח, יקבל את ה- BL הנ"ל, כי אנו רוצים לדאוג לעבוד כל הזמן עם אותו מופע.
הערה: כאשר ישנו מסך המאפשר הוספת ערך כמו מנה, סניף וכדו' הקיימים במערכת או עדכון, יש לאפשר בחירה של שדה זה ע"י פקד ComboBox שבו נקבל רשימה ממנה יש לבחור.
ישנה אפשרות שהוספה עדכון ומחיקה יהיו למעשה אותו מסך, בהתאם לצורך באותו רגע.

(אין להוסיף פונקציונאליות מעבר ל- CRUD בסיסי ל- DAL !)

כמו כן עליך לדאוג להוסיף זריקה של חריגות במקומות המתאימים ותפיסה של החריגות כדי שהתכנית לא "תעוף" במקרה שפעולה מסוימת נכשלה בזמן הרצה.

הערה חשובה: חובה לעשות שימוש ב:
resources, data binding, converter, data context, dependency property, trigger

כאן תתבצע הגנה.

חלק ב – עדכון שכבת ה- DAL
מימוש באמצעות Linq-To-XML

הוספה ומימוש מחלקה נוספת ב-DAL המממשת את ה- IDAL באמצעות Linq – to –XML:

עליך להוסיף לפרויקט **DAL** מחלקה נוספת בשם Dal_XML_imp

אשר מכילה: כתובת של מיקום קובץ xml (בתוך משתנה מתאים)

יש להכין קבצי xml שיחליפו את האוספים שיצרנו כבר (כלומר יחליפו את מקור הנתונים) אחד לכל אחת מהישויות, אשר ייכתבו בפורמט המתאים למבנה הישות אותה הוא מייצג. קבצים אלו יהיו בתוך תיקיה נפרדת ב- solution.

יש לעשות אותו תהליך שעשינו בכיתה כדי ליצור את האתחולים, אפשרות שמירה וטעינה של קובץ וכן אפשרות תשאול ע"י שאילתת Linq.
לאחר מכן יש לממש את כל המתודות של הממשק של ה- IDAL.

ניתן להיעזר בהדרכה להלן:

עבור אתחול קבצי הXML:
לדוגמה:
studentRoot = new XElement("students");
כעת נוכל כל פעם להוסיף \להוריד ולעדכן אלמנטים בקובץ
במידה והקובץ כבר קיים, פשוט לא ניצור אותו.
הקוד יהיה בערך כך:

```
public XmlSample()         {
            if (!File.Exists(FPath))
                CreateFiles();
        }

        private void CreateFiles()
        {
            studentRoot = new XElement("students");
        }
```

עבור המספר הרץ של חלק מהמחלקות ועוד כמה הגדרות:

הבעיה:
כשמימשנו את המספר הרץ השתמשנו במשתנה סטטי שגדל במהלך התוכנית.
כעת שנשמור את הנתונים ב- xml בפעם הבאה שנפתח את התוכנית אותו משתנה יתאפס שוב ואז המספר הרץ יתחיל מ- 0

הפתרון:
נגדיר קובץ XML בשם config.xml ושם נגדיר את המספר הרץ ועוד הגדרות שנרצה
כל פעם שנשמור אובייקט שמשתמש במספר הרץ נעדכן גם את קובץ ה config.xml
בפעם הראשונה שה dal נוצר הוא ידאג לעדכן זאת במחלקה.

הערה:
עבור קובץ ה- config ועוד קובץ אחד של אחת המחלקות, יש להשתמש ב- linq to xml
עבור כל פעולות ההוספה עדכון ומחיקה ושליפה.
במימוש של שאר המחלקות ניתן להשתמש ב- serialize, כלומר לעבוד עם הנתונים ב- list
ובסופו של דבר לשמור את זה ל- XML באמצעות xmlSerialize

כאן תתבצע בעז"ה ההגנה הסופית

בהצלחה !

אלו הן אמות הבדיקה לבדיקת הפרויקט ב- C#

א. ב- DAL וב- BL יש לבדוק: שימוש ב- Factory Method עבור DAL ועבור BL. רצוי מאוד ש- DALיהיה בתבנית סינגלטון, שימוש מגוון ב- Linq, שימוש ב- new select, שימוש ב- Grouping, שימוש ב- Lambda, שימוש באנונימיוס delegates, שימוש ב- predicates ב- FUNC. רצוי שימוש ב- let בתוך שאילתת Linq.

ב. ב- BL יש לבדוק יצירתיות של בדיקות. כמו למשל אכיפת כללים המיוחדים לארגון הזה ע"פ הבנת התלמיד.

ג. חריגות שעולות מ- DAL ל- BL ומ- BL ל- UI כאשר לכל שכבה יש גם חריגות מובנות משלה !

ד. ב- XML לבדוק שימוש נכון ב- LinQ To XML, כמו כן לסגור אפליקציה ולחזור אליה ולבדוק ששינויים אכן נשמרו. זריקת החריגות מ- DAL צריכה להיבדק.

ה. ב- UI לבדוק שימוש מגוון בפקדים. וחריגות שהגיעו משכבות קודמות שמטופלות וכן חריגות שנזרקות ומטופלות ששייכות לשכבת UI בעצמה כמו למשל שמשתמש לא מילא את כל השדות הנידרשים או שמילא שדות נומריים בתווים וכו'.

ו. כמו כן יש לבדוק תיעוד, ואלו הם הכללים לבדיקת התיעוד:
- ✓ מעל הפונקציות באינטרפסים של IDAL ו- IBL תוך שימוש ב \\\
- ✓ להוסיף תיעוד עבור מימושים רק מעל פונקציות יצירתיות ב- BL שאוכפות נהלים מסוימים שהמצאתם כמו למשל "תחזיר רק את המנות הכי פופולריות שהופיעו מתאריך Xעד תאריך Y" בקיצור, תיעוד רק עבור דברים שאינם טריוויליים או שימוש בשאילתות מורכבות. כל השאר אין צורך לתעד.

כאן תתבצע ההגנה הסופית.

בהצלחה !

הגשות:

הערה: התאריכים הניתנים להלן מסתמכים על כך שהספקתם בהרצאה שלכם להגיע לנקודה זו, באם המרצה חושב שיש לתת לכם תאריך שונה במקצת, הוא יודיע לכם על כך ואתם תהיו מחויבים לתאריך שהמרצה דורש מכם.
יחד עם זאת את קבצי הפתרון כולם יגישו לאותן תיבות הגשה במודל.

הגשה 1 – חלק א עד שלב ג. תאריך הגשה:
הגשה 2 – חלק א שלב ד.:
הגשה 3 – לאחר חלק ב' סופי – **תאריך הגשה** : מפגש אחרון בסמסטר (קבוצה של יום ראשון עד ואלו קבוצה של יום רביעי עד)
(ההגנה בשבוע האחרון בשעות המעבדה). לאחר סיום הסמסטר לא יתבצעו בדיקות.