שאלה 1

1. שיטה מס׳ 1 – Information hiding– גישה זו מנסה להפוך את המערכת למודולים שונים אשר תלויים זה בזה רק באמצעות ממשקים ביניהם. לכן אם אנו צריכים לשנות/לתקן מודול מסוים, נוכל לדאוג לשנות רק את הקוד שבתוכו מבלי לפגוע בממשקים, ובכך להקטין את כמות הקוד שיש לנו לשנות בכל התוכנית, שכן שאר המודולים ישתמשו באותם ממשקים.
שיטה מס׳ 2 – מודל המפל- שיטה זו היא למעשה מתודולוגיה לפיתוח תוכנה, אשר שמה דגש על הבנה אבסולוטית ודוקומנטציה של הדרישות, עוד לפני תחילת שלב הפיתוח ועיצוב של התוכנה. למעשה, היא מכריחה אותנו להבין את עיצוב התוכנה הכולל לפני שנמשיך לשלב הבא.
שיטה מס׳ 3 – Structured Programming – שיטה זו היא פרדיגמת תכנות אשר נועדה לשפר את איכות ובהירות הקוד אותו אנו כותבים, ע״י שימוש נרחב במבנים אשר מאפשרים לנו שליטה על הזרימה של התוכנית. נוסף לכך, שימוש במבנים אלה מאפשר לקוד שלנו להיות מסודר יותר ומודולורי יותר.

2. עבור כל אחת מהשיטות אציג חסרונות אשר מונעות ממנה להית ״כדור כסף״ :
שיטה מס׳ 1 – דרישות חדשות עלולות להיות חוצות – מודולים , דבר הדורש תכנון ועיצוב מחדש של כלל המודולים, פעולה שהיא כבדה וגוזלת זמן.
שיטה מס׳ 2 – בגלל שמתודולוגית הפיתוח דורשת ממנו תחילה לנתח את כלל הדרישות, לעצב את המערכת ורק אז להתחיל לתכנת, התהליך מתארך דבר אשר מגדיל את עלות הפרויקט. כמו כן, מכיוון שהתהליך הוא ארוך, אנו עלולים להתעלם מדרישות חדשות שעלולות לצוץ תוך כדי שלב העיצוב/פיתוח דבר היכול לפגוע במשתמשים שלנו.
שיטה מס׳ 3 - במקרה של שינוי שנדרש באחד המבנים, המתכנת נתקל בשני אפשרויות :
1. לעשות פאץ׳ או תיקון לקטע הקוד - דבר הגורם לשתי בעיות : הראשונה היא שבדר״כ תיקון של קטע קוד כזה מוביל לאפקט של ״גל״ אשר גורר עוד שינויים ובעיות והשנייה היא שתיקון/שינוי הקוד מוביל לכך שהקוד ה״חדש״ סותר את המבנה ואת האבסטרקציה אותה הוא מייצג, דבר המחמיר עבור כל שינוי שנערך בקוד.
2. לתכן ולפתח את כל המבנה מחדש לגמרי, דבר אשר מבזבז יותר זמן ומוביל לסתירות אל מול הקוד הישן.

3. BMUF – שמה דגש על להבין את כל הדרישות עוד לפני שלב העיצוב והפיתוח, ואף מנסה לחזות את הדרישות שעלולות לצוף תוך כדי התהליך.
IRUF לעומת זאת היא גישה אשר ״מאמצת״ את השינוי לחיקה, ולכן היא לא מנסה לחזות את כל הדרישות מראש, אלא מבינה שדרישות יכולות לצוץ תוך כדי התהליך ושיש להגיב לדרישות אלו.
במידה והייתי החברה היחדה בשוק, הייתי בוחר ב IRUF בגלל שתי סיבות :

1. IRUF מאפשר הוצאה מהירה יותר של תוכנה עובדת, ובגלל שהייתי רוצה לנצל את היותי החברה היחידה בשוק הייתי רוצה להוציא תוכנה שעובדת כמה שיותר מהר.
2. מכיוון שזהו שוק לא מנוצל, יכולות להיות כמות עצומה של דרישות אשר בלתי אפשרי לחזות אותן מראש, לכן הבחירה ב IRUF שהיא גישה גמישה יותר עדיפה.

4. הבעיה העיקרית בשימוש בשטות פורמליות, ובאימות תוכנה, היא עם שינוי הדרישות. מכיוון שדרישות באופן מובנה הן גלובליות, כך גם הטענות וההוכחות הפורמליות המייצגות אותן הן כאלה, ולכן כל שינוי בדרישות עלול לגרור ניסוח מחדש של כל הטענות האלו, מכיוון שהן מבוססות על טענות לא רלוונטיות.

5. ברי במאמרו מציין כי test cases הן דרך נוחה להצגת הדרישות בגישת xp. את ה test cases האלה כותבים בשילוב המפתחים וגם הלקוח, דבר ההופך את הבדיקות ליותר מדויקות ומאפשרות לבחון שהתוכנה אכן עומדת בדרישות. כמו כן, מכיוון שלפי מתודולוגית הפיתוח xp את הבדיקות האלו כותבים, בתחילת ובסוף כל איטרציה, הבדיקות עוזרות לנו להימנע מבאגים מסובכים וקשים.

שאלה 2

1. סיפור המשתמש "הוספת פרויקט":

1. פעולה	2. מידע	3. תוצאה רצויה
הוספת פרויקט	שם הפרויקט : תקין, תיאור : תקין, היקף משוער בשעות : תקין, ופרטי המציע : דוא"ל : תקין, שם פרטי : תקין, שם משפחה : תקין וטלפון : תקין.	הצלחה: קבלת קוד פרויקט, מצב הפרויקט "בבדיקה".
הוספת פרויקט	שם הפרויקט : **ריק**, תיאור : תקין, היקף משוער בשעות : תקין, ופרטי המציע : דוא"ל : תקין, שם פרטי : תקין, שם משפחה : תקין וטלפון : תקין.	כישלון: לא ניתן להוסיף פרויקט ללא שם.
הוספת פרויקט	שם הפרויקט : תקין, תיאור : תקין, היקף משוער בשעות : **לא תקין**, ופרטי המציע : דוא"ל : תקין, שם פרטי : תקין, שם משפחה : תקין וטלפון : תקין.	כישלון: היקף השעות לא תקין.

הוספת פרויקט	שם הפרויקט : תקין, תיאור : תקין, היקף משוער בשעות : תקין, ופרטי המציע : דוא״ל : תקין, שם פרטי : **ריק**, שם משפחה : תקין וטלפון : תקין.	כישלון: לא ניתן להוסיף פרויקט ללא שם המציע.
הוספת פרויקט	שם הפרויקט : תקין אך קיים כבר במערכת, תיאור : תקין, היקף משוער בשעות : תקין, ופרטי המציע : דוא״ל : תקין, שם פרטי : **ריק**, שם משפחה : תקין וטלפון : תקין.	כישלון : שם פרויקט תפוס כבר.

סיפור המשתמש :״רישום פרויקט״ :

1. פעולה	2. מידע	3. תוצאה רצויה
רישום פרויקט	פרויקט אשר **קיים כבר** במערכת, ת״ז תקינה של **שלושה סטודנטים**, שם מנחה תקין.	הצלחה: קבלת קוד גישה המאפשר לפתוח אתר לפרויקט.
רישום פרויקט	פרויקט אשר **קיים כבר** במערכת, ת״ז תקינה של **ארבעה סטודנטים**, שם מנחה תקין.	הצלחה: קבלת קוד גישה המאפשר לפתוח אתר לפרויקט.
רישום לפרויקט	פרויקט אשר **קיים כבר** במערכת, ת״ז **שלא שייכת לסטודנט**, שם מנחה תקין.	כישלון: הודעה של ״ת״ז לא תקינה״.
רישום לפרויקט	פרויקט אשר **קיים כבר** במערכת, ת״ז **לא תקינה**, שם מנחה תקין.	כישלון: הודעה של ״ת״ז לא תקינה״.
רישום לפרויקט	פרויקט אשר **קיים כבר** במערכת, ת״ז תקינה של סטודנט ושותפיו, שם מנחה אשר **לא מופיע במערכת.**	כישלון: הודעה של ״מנחה לא נמצא״.

2. עבור סיפור המשתמש הראשון :
   1. המשתמש חייב להזין את כל פריטי המידע. (הוא לא יכול לדלג על פריט מידע מסוים).
   2. המשתמש חייב להזין את פריטי המידע באנגלית.

הנחה מס׳ 1 מהווה התנגשות בין נק׳ מבט של המשתמש אשר לא מעוניין להכניס את כל השדות (כמו : תיאור הפרויקט ודוא״ל), אל מול נק׳ המבט של המתכנת שמניח שכל פריטי המידע מוזנים.
הנחה מס׳ 2 מהווה התנגשת בין נק׳ המבט של הלקוח שרגיל לכתוב בעברית אל מול נק׳ המבט של המתכנת אשר גריל לכתוב קוד עם תמיכה באנגלית בלבד.

1. פעולה	2. מידע	3. תוצאה רצויה
הוספת פרויקט	שם הפרויקט : תקין, תיאור : לא מוזן, היקף משוער בשעות : תקין, ופרטי המציע : דוא״ל : תקין, שם פרטי : תקין, שם משפחה : תקין וטלפון : תקין.	הצלחה: קבלת קוד פרויקט לאחר לחיצה על ״אישור״, מצב הפרויקט ״בבדיקה״.
הוספת פרויקט	שם הפרויקט: **כתוב בעברית,** תיאור: תקין ופרטי המציע : תקינים.	כישלון: לא ניתן להזין פרטי מידע בשפה שהיא לא אנגלית.

עבור סיפור המשתמש השני :

1. רק סטודנטים בשנה ד׳ יכולים לבחור פרויקט מהרשימה.
2. סטודנט שנה ד׳ לא יכול לבצע יותר מפרויקט אחד.

שתי ההנחות הסמויות מהוות התנגשות בין נק׳ מבט של הלקוח (האוניברסיטה) אשר דורשת את ההנחות האלו לבין נק׳ מבט של המתכנת אשר מנסה לבצע את המימוש הפשוט ביותר.

1. פעולה	2. מידע	3. תוצאה רצויה
רישום לפרויקט	פרויקט אשר נמצא בערכת, ת״ז של סטודנט אשר **בשנה ג׳**, שם מנחה תקין.	הצלחה: הודעה שהרישום נכשל מכיוון שהסטודנט שנה ג׳ ולא ד׳.
רישום לפרויקט	פרויקט אשר נמצא במערכת, ת״ז של סטודנט אשר **כבר רשום לפרויקט אחר**.	הצלחה: הודעה שהרישום נכשל מכיוון שהסטודנט רשום כבר לפרויקט אחר.

3. טבלת נתונים עבור תרחיש השימוש **"הוספת פרויקט"**:

סוג תרחיש	שם הפרויקט	תיאור	היקף שעות	פרטי המציע	ארגון	תוצאות צפויות
שמח	תקין	תקין	תקין	פרטים תקינים	תקין	קבלת קוד פרויקט, מצב הפרויקט "בבדיקה".
רע	תקין	תקין	מחרוזת לא תקינה	תקין	תקין	קבלת הודעת שגיאה.
רע	תקין	תקין	תקין	שם פרטי: מחרוזת ריקה	תקין	קבלת הודעת שגיאה.
רע	מחרוזת ריקה	תקין	תקין	תקין	תקין	קבלת הודעת שגיאה.
שמח	תקין	מחרוזת ריקה	תקין	תקין	תקין	קבלת קוד פרויקט, מצב הפרויקט "בבדיקה".
עצוב	תקין, אך קיים כבר במערכת.	תקין	תקין	תקין	תקין	קבלת הודעה שגיאה.
עצוב	תקין אך בעברית.	תקין	תקין	תקין	תקין	קבלת הודעת שגיאה.

טבלת נתונים עבור תרחיש השימוש **"רישום לפרויקט"**:

סוג התרחיש	ת"ז סטודנטים	שם המנחה	תוצאות צפויות
שמח	תקין	תקין	קוד גישה תקין.
עצוב	ת"ז שלא מופיעה במערכת	תקין	הודעת כישלון "ת"ז לא תקינה".
עצוב	תקין	שם אשר לא מופיע במאגר	הודעת כישלון: "שם מנחה לא נמצא".
עצוב	ת"ז של סטודנט בשנה ג'.	תקין	הודעת כישלון : "הרישום נכשל כי הסטודנט לא שנה ד'"
עצוב	ת"ז של סטודנט אשר רשום כבר לפרויקט.	תקין	הודעת כישלון : "הרישום נכשל כי הסטודנט רשום כבר לפרויקט אחר"
רע	מחרוזת לא תקינה	מחרוזת לא תקינה	הודעת כישלון : "הקלט אינו תקין".

<u>חלק ג' – סעיף ג' :</u>

על מנת לקשר את הבדיקות עם התוכנה האמיתית נצטרך להכניס ערך(את האפליקציה) לשדה real בתוך ה ProxyBridge, לאחר שנעשה זאת ה ProxyBridge יריץ את התוכנה האמיתית.

לא נצטרך לקמפל מחדש את הבדיקות, היתרון בכך שאנו יכולים לחסוך זמן יקר של קימפול הבדיקות(שעלולות להיות ארוכות).