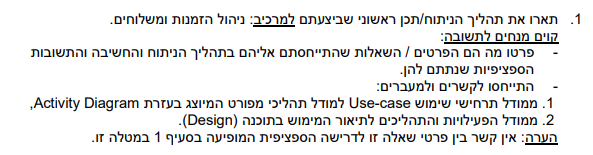
**קבוצה מספר 15**

שאלה 1



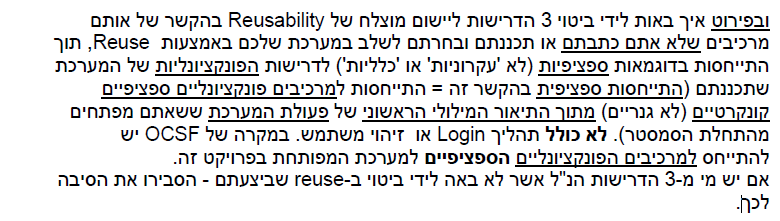
תשובה :

1. במודל הUSE CASE פירטנו את אופן פעולת המערכת כדי להבין מה המערכת מאפשרת לעשות בלי הכוונה לתהליך מסוים מהתחלה עד הסוף, לדוגמא בהזמנה הראנו כי אפשר לעשות כמה תהליכים מתהליך רכישה – כמו בחירת מוצר, תשלום וכו' אבל לא הראנו את רצף הפעולות בדיאגרמה ויכלנו לראות תיאור כללי בflow of events .

בדיאגרמת האקטיביטי, העמקנו והפרטנו את הפעולות ליחידות קטנות יותר ומפורטות יותר בהן ניתן לראות את התרחשות התהליך ההזמנה, הראנו מה קורה בצד הלקוח וכי הוא זה שמתחיל את תהליך ההזמנה (בUSE CASE זה משתמע מהעובדה ששחקן הלקוח מקושר ל- UC הזמנה) באופן מדויק יותר ובאמצעות השמת הפעולות ב-partitions השונים מידלנו היכן פעולות אלו יקרו במערכת, וכיצד הcontroller-ים במערכת יתקשרו אחד עם השני בכדי להוציא לפועל את התהליך ההזמנה. ולדוגמא אם לקוח בחר במשלוח אז כיצד לחיצת הכפתור הזו מפעילה תהליך אחר של שליחת המוצר.

1. שבוע הבא אחרי שנלמד

שאלה 2:



תשובה:

בחרנו להשתמש בreuse במערכת שלנו בכך שהשתמשנו בקוד קיים של OCSF שעבדנו איתו בתרגול 5 במקום לממש בעצמנו מהתחלה את השרת-לקוח בבניית האבטיפוס.

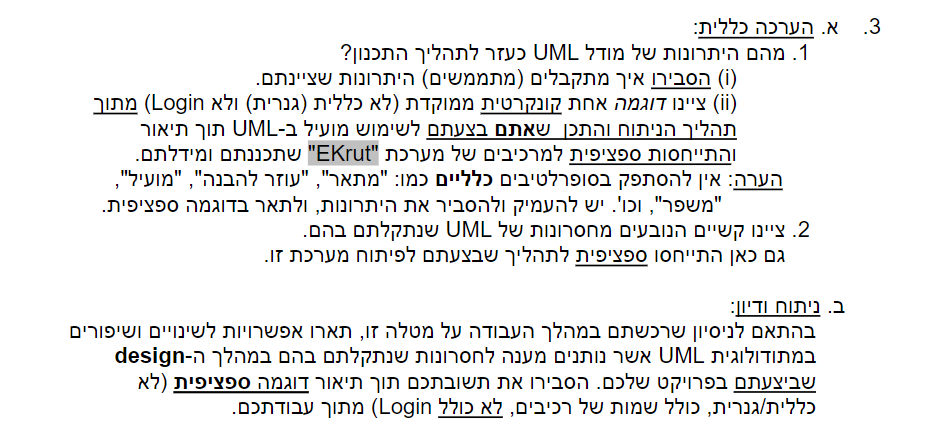
שלושת הדרישות ליישום מוצלח הן: availability, understandability, flexibility.

Availability- זמינות: הקוד שלOCSF היה זמין לנו מהתרגול וגם אפשר למצוא אותו בקלות באינטרנט. ה,Nנו בקוד מהתרגול והתאמנו אותו למערכת שלנו.

Understandability-הבנה: הקוד ברור, מתועד ובעל שמות משמעותיים כך שניתן להבין בקלות את התפקיד של כל מחלקה\מתודה וכו'. לדוגמה, השתמשנו במתודה

Flexibility- גמישות: הקוד הוא גנרי ואנחנו השתמשנו בו והתאמנו אותו למערכת שלנו. לדוגמה, השתמשנו ב

שאלה 3:



תשובה:

1. **סטנדרטיזציה:**  
   מודל UML הינו הסטנדרט העדכני לעבודה עם שפות תכנות מונחות עצמים. כאשר אנו ניצור מחלקות ואובייקטים עם יחסים ביניהם, מודל UML הוא מה שישמש אותנו כדי לתאר באופן חזותי את היחסים האלה. מכיוון שהמודל הינו הסטנדרט, הוא מובן וידוע היטב על ידי מתכנתים רבים, דבר המקל על מתכנתים חדשים.

**ייצוג ויזואלי:**  
מודל UML הינו הדרך לייצג חזותית יחסים בין ישויות ומחלקות במערכות מחשב שונות. באובייקט מחלקה אנו נארגן משתנים ופונקציות במקום אחד, וכדי להבין תוכנית, חיוני להבין מה כל אובייקט עושה, איזה מידע הוא מאחסן וכיצד הוא מתקשר עם אובייקטים אחרים. על ידי ייצוג האובייקטים במודל UML, קל להבין ולייצג את קשרי הגומלין בתוכנית.

**תכנון מקדים:**  
באמצעות מודל UML ניתן לתכנן מערכות לפני יישומן, דבר המפחית את העומס בפרויקט באופן משמעותי. בנוסף, קיימים כלים רבים במודל UML המנפיקים קוד המתאר מחלקות ואת היחסים ביניהן, מה שמוסיף לעובדה כי שינוי פרט בתרשים יותר פשוט מאשר תכנון של קטע קוד מחדש.

1. מודל UML מספק לנו דיאגרמות שונות המציגות את היתרונות לעיל:  
   דיאגרמת ה-Activity מספקת לנו תיאור של השתלשלות האירועים בתוכנית (Flow Of Events), מציגה תנאים, אירועים ותהליכים סינכרוניים ואסינכרוניים, מה שמספק לנו את התכנון המקדים לפני מימוש המערכת בשפת תכנות.  
   דיאגרמת ה-Class מאפשרת לנו לייצג את כל הישויות והמחלקות במערכת ולתאר את המבנה הפנימי של כל מחלקה.  
   מכיוון שמודל UML מוכר ומובן על ידי מתכנתים רבים, נוצרה מוסכמות לגבי תכנון מקדים של מערכות. כתוצאה מכך, ידע והבנת המודל הינה דרישה בעולם התכנות. דבר זה גרם לסטנדרטיזציה של מודל UML כשפה משותפת.
2. השתמשנו בדיאגרמת Class על מנת לתאר את המחלקות והישויות השונות במערכת EKrut ובפרט את תהליך הזמנת מוצר על ידי הלקוח. לאחר ניתוח ארוך של התהליך, תיארנו את המחלקות השונות, את קשרי הגומלין ביניהן ,את המשתנים והפונקציות בכל מחלקה ואת הקשר של התהליך לשאר המערכת. ללא הניתוח והתיאור בעצמאות דיאגרמת Class, לא היינו מצליחים להרחיב ולתאר מחלקות נוספות, כגון – ReviewOrderController, המשתתפות בתהליך ההזמנה. כתוצאה מכך המון פונקציות משמעותיות היו מתפספסות.
3. קשיים אשר נתקלנו בהם:
4. **אי יכולת להתייחס לשגיאות זמן ריצה וקומפילציה**
5. **עומס רב של תרשימים בדיאגרמת Sequence**
   * בגלל גודל המערכת שלנו, אנו זקוקים להמון תנאים ולולאות כדי לתאר כל סיטואציה אפשרית. מפני שלולאות ותנאים נמצאים בתוך מלבן, דבר זה יוצר עומס וקושי להבין אילו פעולות ופקודות נכנסות תחת איזה תנאי או לולאה.
6. **לא ניתן להציג מקרים חריגים ומקרי קצה**

3. ב.

כאן צריך להציע פתרונות ושיפורים לקשיים