גרסה 2: רכיב ייצוג, התראות זמן אמת, והגדרות מדיניות – 2019

כל גרסאות הפרויקט מתייחסות אל ותלויות במסמך הדרישות הכללי ובמסמך מתודולוגיית פיתוח הפרויקט. כל גרסה מרחיבה או משנה את הגרסה שקדמה לה. בכל שלב יש לשמור על תאימות בין המודלים והמימוש.

גרסה 2 עוסקת בהעשרת המערכת ברכיב ייצוג (presentation) התומך בהתראות זמן אמת. דרישות הקיבול והזמינות מהמערכת מכתיבות בחירה בארכיטקטורת שרת-לקוח, כאשר רכיב שכבת השירות שנבנה בגרסה 1 משמש כמנשק בין רכיב הייצוג לשרת. בגרסה זו, הגדרות מדיניות קנייה והנחה מועשרות במספר סוגים ובתלויות שונות.

תפקידי צוות: מנהל/ת גרסה המשמש גם כבודק מטעם הלקוח, ומפתחים. פירוט תחומי האחריות של התפקידים השונים נמצא במסמך המתודולוגיה. במיוחד יש לשים לב לתחומי האחריות של מנהל הגרסה.

דרישות נוספות עבור גרסה 2

1. **דרישות רמת שירות:** הוספת רכיב ייצוג, תוך שימוש בטכנולוגיית web והקפדה על קיום דרישות רמת השירות הרלוונטיות:
   a. **דרישה 3c - ממשק משתמש נוח:**
      i. **דרישות איפיון ממשק:** לצורך מימוש דרישה זו, אחד מחברי הקבוצה (בהתייעצות עם מנהל הקבוצה) יכתוב מסמך המגדיר את המושג "ממשק נוח" באופן הניתן לבדיקה.
      בנספח של גרסה זו מופיעה תזכורת לגבי המאפיינים של דרישת-תוכנה המוגדרת היטב.
      יש להקפיד על קיום הדרישות לאיפיון ממשק, במימוש של ממשקי משתמשים .
      ii. **בדיקות ממשק:** חובה להגדיר בדיקות (לאו דווקא אוטומטיים) לדרישות שתכתבו.
   b. **דרישה 3d - ממשק המשתמש תואם את תפקיד המשתמש במערכת:** הממשק יחשוף למשתמש רק את הפעולות אותן הוא רשאי לבצע בכל הקשר, בהתאם לתפקיד המשתמש ולהרשאותיו.
   c. **דרישה חדשה 3e - המערכת מספקת הסברים למשתמש לגבי פעולותיה:** למשל, אם פעולת קנייה נכשלת, המערכת נדרשת להציג למשתמש מידע אודות סיבת הכישלון (הפרה של מדיניות קנייה, מערכת גביית הכספים דחתה את התשלום וכד'). כמו כן, על המערכת להודיע למשתמש גם במקרים בהם הפעולות אותן ביקש לבצע הסתיימו בהצלחה.
   d. **דרישת מעקב 7 -** אחר פעולות ותקלות (היה כבר בגרסה קודמת). לצורך מימוש הדרישה עליכם לתחזק יומן אירועים (event log) ויומן שגיאות (error log) בקבצים חיצוניים למערכת.
2. **דרישות פונקציונאליות:** גרסה זו מרחיבה את הפעולות של המערכת, אך אין תוספת של תמיכה בתרחישי שימוש חדשים, מעבר לגרסאות הקודמות. הפעולות החדשות מופיעות במספר תרחישי שימוש. יש לעדכן את תרחישי השימוש הקיימים, את מימושם וכן את המודלים הקיימים, בהתאם.
   a. **התראות זמן אמת – דרישה 10a:** התראות זמן אמת **אינן תרחיש שימוש**, אלא איפיון של התנהגות המערכת בהקשרים שונים, כפי שמפורט במסמך הדרישות הכללי. יש לבדוק את סיפורי

השימוש הרלבנטיים, ולוודא שהם מפרטים את הביצוע של התראות זמן אמת (צריך לבוא לידי ביטוי ב-system sequence diagram.

יש לשים לב לדרישת ההתראות המושהות: ההתראות של מנויים שאינם מבקרים בזמן ההתראה נשמרות במערכת ומוצגות להם בעת ביקורם הבא.

b. הנחות בקניה: השקיעו מחשבה והיו מוכנים להצדיק את המבנה והחתימות (קלט, פלט, תנאי קדם) של הטיפול בהנחות.

i. **סוגי הנחות:** גלויה ומותנית. יש להבהיר את מהות הפעולה של הנחה מותנית.

ii. **תרחישי שימוש רלבנטיים:** 4.2, 5.1: בעל חנות (או מנהל חנות עם הרשאות מתאימות) יכול להגדיר הנחות עבור מוצרים בחנותו.

iii. **מדיניות הנחה:**

- הנחה יכולה להיות מוגדרת עבור מוצר או עבור חנות.
- **הנחה מורכבת:** הנחה יכולה להיות מורכבת מהנחות אחרות. למשל, "הנחה על מוצרי חלב או מאפים, אבל לא שתיהן" – אין כפל מבצעים. ההרכבה של הנחות היא משמעותית במיוחד עבור הנחה מותנית, כפי שמוסבר להלן.
- בהנחה מותנית, גם התנאי להנחה יכול להיות מורכב: למשל, "קנה (פטיש וגם מברג) או מסור-דיסק וקבל מסמרים ב-60% הנחה".
- **אופני הרכבה של הנחות או תנאים להנחה מותנית:** כדאי לחשוב על ההרכבה של הנחות או תנאים להנחה כעל שימוש בפעולות לוגיות מסוג "וגם", "או" ו-"או בררני" (XOR). תבנית העיצוב composite, המאפשרת בנייה היררכית של ישויות, יכולה להיות לעזר.
- הקבוצה יכולה לקבל החלטות פרטניות לגבי סדר הפעלת ההנחות וכד'.
- ההרכבה הנחות ותנאיהן ניתנת להגדרה באמצעות הממשק. כלומר, הממשק מאפשר שימוש באופני ההרכבה. בצורה גמישה, לבעלי הרשאת מתאימות.

c. סוג ומדיניות קניה: השקיעו מחשבה והיו מוכנים להצדיק את המבנה והחתימות (קלט, פלט, תנאי קדם) של הטיפול מדיניות קניה.

i. **סוגי קנייה:** מיידית.

ii. **תרחישי שימוש רלבנטיים:** 4.2, 5.1: בעל חנות (או מנהל חנות עם הרשאות מתאימות) יכול להגדיר מדיניות קניה בחנותו.

iii. **מדיניות קנייה:**

- מדיניות קנייה יכולה להיות מוגדרת עבור כל החנות או עבור מוצר בודד.
- **מדיניות קנייה מורכבת:** כללי מדיניות קנייה יכולים להיות מורכבים מצירופים של כללים אחרים. דוגמאות למדיניות קניה:
  1. עבור חנות: רכישת לכל היותר 5 מוצרים מהחנות בקנייה יחידה.
  2. עבור מוצר: ניתן לרכוש לכל היותר חמישה עותקים של הספר "מיץ פטל" ולכל הפחות שני עותקים.
  3. עבור פרטי משתמש. למשל, מיקום גאוגרפי: לא ניתן למכור למשתמשים מחוץ לישראל.

- **הרכבה של כללי מדיניות קנייה:** המערכת צריכה לתמוך באופני שילוב של כללי מדיניות קניה, בדומה להרכבת הנחות.
- ההרכבה של כללי מדיניות קנייה ניתנת להגדרה באמצעות הממשק. כלומר, הממשק מאפשר שימוש באופני ההרכבה. בצורה גמישה, לבעלי הרשאת מתאימות.

מידול ומימוש

את רכיב הלקוח יש לממש כ- Web client שישמש כממשק גרפי (GUI) המסוגל לקבל ולהציג התראות למשתמש בזמן אמת. מומלץ להשתמש בפרוטוקול WebSockets עבור המימוש של התראות זמן אמת; הקדישו מחשבה אילו מהבקשות צריכות להישלח באמצעות WebSockets ואילו לא.
ארכיטקטורה: הארכיטקטורה העדכנית של המערכת תכלול שכבות:

- רכיב לקוח **(presentation layer)**: זהו ה-Web Client שאותו יש לממש. על מנת לתמוך בתקשורת בין רכיב הלקוח לבין השרת.
- יש להוסיף **רכיב תקשורת.** רכיב התקשורת יהיה אחראי להעביר את הפניות וההודעות המגיעות מרכיב הלקוח ואליו. על רכיב זה לתמוך בשני אופני התקשורת מ- ואל המערכת (HTTPS ו-WSS).
- שכבת שירות (**service layer**) עדכנית.
- שכבת **domain** עדכנית.

מידול ממשק המשתמש באמצעות Statecharts: רכיב הלקוח מאופיין במצבים המגיבים לאירועים (events) מבחוץ. על כן, טוב לתאר את פעולתו באופן מופשט על ידי דיאגראמת מצבים היררכית (statechart). **לכל סיפור שימוש יהיה מצב היררכי ייעודי**, ואולי מצבים היררכיים מקוננים נוספים. ההיררכיה חיונית לצורך הפרדה ושליטה.
הממשק הגרפי (GUI) מממש את המידול כ-statechart באופן ישיר.

התראות זמן אמת: רכיב הלקוח הוא חלק מהארכיטקטורה השכבתית, המאופיינת בתקשורת חד כיוונית. התראות זמן אמת סותרות את הכיוון השיכבתי, מאחר וזוהי תקשורת מכיוון שכבת הdomain, לרכיב הלקוח (דרך רכיב התקשורת). סוג כזה של תקשורת מטופל על ידי תבנית העיצוב **Observer**. בשכבת השירות ממוקם **publisher** המתחזק אוספי לקוחות מסוגים שונים, ואחראי על שליחת התראות לפי סיווגים שונים ללקוחות. ההתראות גורמות ללקוחות לפנות למערכת לקבלת המידע הרלוונטי.
שים לב: להפריד בין ה-**publisher** לבין המרכיב הטכנולוגי של התקשורת.

תוצרי גרסה 2:

מסמך המתודולוגיה מתאר את התוצרים השונים ואופן תיאורם. התוצרים בגרסה זו מבוססים על התוצרים מגרסה 1 ומרחיבים עליהם. יש להקפיד על תיאום מוחלט בין מודלים למימוש, ולהציג באופן בולט את כל השינויים וההוספות שנעשו ביחס לגרסה הקודמת. מנהל הגרסה אחראי על הצגת תוצרי הגרסה.

1. דו"ח גרסה:
   a. פירוט ההספק לגרסה זו: אילו משימות מוכנות במלואן, אילו באופן חלקי ואילו נדחו לגרסה עתידית.
   b. תיעוד ניהול הגרסה, ע"פ העקרונות המוצגים במסמך המתודולוגיה (2.1.1).
2. תיקונים מגרסה 1, במידת הצורך ובהתאם להנחיות המנחה.
3. מילון מונחים עדכני.
4. תרחישי שימוש עדכניים.
5. ארכיטקטורה עדכנית.

6. מודל מחלקות עדכני: דיאגרמה לבנה ודיאגרמה מפורטת יותר. יש להדגיש את השינויים/תוספות
7. מסמך דרישות לממשק המשתמש
8. מודל ממשק המשתמש כתרשים מצבים היררכי (statechart)
9. מימוש:
   a. שכבות ה-**domain** והשירות **(service)** באופן התואם למודל המחלקות, למודל הארכיטקטורה, ולתרחישי השימוש.
   b. שכבת הייצוג **(presentation)** באופן התואם את ה-statechart ההיררכי, ואת מסמך הדרישות שהכנתם.
   c. מימוש מפורש לתמיכה בהתראות זמן אמת, על פי תבנית העיצוב Observer.
10. בדיקות: הוספה ועדכון בדיקות לפי הדרישות החדשות. עבור כל הסוגים, כולל טסטים לממשק הגרפי. **לא יתקבל מימוש ללא רכיב בדיקות משמעותי!**

מצגת כיתה: עקביות ושמירת נתונים לאורך זמן

הגרסה הבאה של המערכת תאפשר גיבוי נתונים לאורך זמן באמצעות רכיב מסד נתונים.

- בחר/י ספרייה המספקת אבסטרקציה של מודל מסד הנתונים ומאפשרת קישור בין המערכת לבסיס נתונים. למשל, עבור מסד נתונים עם מודל טבלאי, תרצו לבחור ספריית **ORM** שתתרגם בין המודל מונחה העצמים לבין המודל הטבלאי. כלים כאלה מאפשרים למפתח להגדיר מיפוי של ישויות של המערכת לישויות של בסיס הנתונים, ומטפל בשמירה ואחזור של מידע בבסיס הנתונים.
- הסבר כיצד הכלי משמש כגשר (**Data access layer**) בין המערכת לבין בסיס הנתונים.
- הדגם עבודה עם הכלי.
- הסבר והדגם את אופן עבודת הכלי עבור מצבים של אי תאימות בין האפליקציה לבין הכלי:
  - התמודדות עם היררכיית מחלקות.
  - התמודדות עם ההקשר (context) של אובייקטים: שמירת הקשרים בין אובייקטים מסוגים שונים: persistent ל non-persistent ולהיפך.

נספח: עקרונות לדרישה מוגדרת היטב

דרישה היא הגדרת מאפיין או אילוץ של התוכן ההכרחיים לקבלת המוצר או התהליך. מבחינת מאפיין - הדרישה משפיעה על יכולות התוכנה, כאשר תוכן הדרישה משפיע על אופן מימוש התוכנה. מבחינת אילוץ – הדרישה היא הגבלה כלשהי על המערכת – עקב מצב קיים או דרישה עתידית.

התכונות הנדרשות של דרישה טובה הינן:

- <u>נכונות</u> -כל דרישה צריכה לייצג את הפונקציונאליות הנדרשת עבור המערכת.
- <u>בדידות ומזוהות</u> – אפשרות לקרוא כל דרישה באופן עצמאי, וזיהוי ייחודי וחד-ערכי של כל דרישה.
- <u>חד משמעיות</u> - על דרישה להיות ניתנת לפירוש בדרך אחת בלבד. כל דרישה צריכה להיות קצרה, מפורשת וברורה. יש להשתמש במילון מלווה כאשר אנו משתמשים בביטוי שיכולים להיות לו מספר משמעויות.
- <u>שלמות</u> - על הדרישות לכסות בצורה מוגדרת היטב את כל היבטי התוכנה.
- <u>עקביות</u> - יש למנוע הגדרת דרישות הסותרות זו את זו.
- <u>סתירות לוגיות</u> – יש להימנע ממצב של סתירה לוגית, כגון אירוע מפעיל כתנאי קדם לתהליך מסוים שאמור להתרחש בו זמנית עם אותו תהליך.

- עקבות למקור - יש לזהות את מקורה של כל דרישה: דרישה מפורשת או דרישה נגזרת.
- הימנעות מתוכן – יש לוודא שאילוצי/הנחיות התוכן מהווים צורך אמיתי.
- בדיקתיות/ מדידות – נדרשת קביעה מפורשת כיצד יהיה ניתן להוכיח את העמידה בדרישה, בדרך שתיקח זמן סופי. על כל הדרישות להיות מובנות הן על ידי הלקוח והן על ידי המפתחים.