גרסה 1 - מימוש ראשוני – 2022

גרסה 1 עוסקת בהקמה ובמימוש שכבות ה-domain והשירות (service) של מערכת המסחר. שכבת השירות משמשת להפעלת סיפורי השימוש. בסיום גרסה זו, הפעלת המערכת תהיה באמצעות מבחנים בלבד. גרסה 1 נבנית על בסיס עבודת המידול המקדימה שנעשתה בגרסה 0. בכל שלב **יש לשמור על תאימות בין המודלים והמימוש**.

תפקידי צוות: מנהל/ת גרסה, אחראי בדיקות, ומפתחים.

הדרישות לגרסה 1:

המימוש של חלק מהדרישות מגרסה 0 נדחה לגרסאות עתידיות. הייצוג של הדרישות הללו במודלים צריך להישאר.

1. **דרישות רמת שירות**:
   a. דרישת העקביות (1), ודרישת הפרטיות (2), מתקיימות. בפרט, כל אילוצי הנכונות המפורטים במסמך הדרישות הכללי נשמרים לאחר כל פעולה.
   b. דרישת קיבול וזמינות (5): על המערכת לתמוך במספר לא חסום של משתמשים מכל הסוגים בו זמנית ועל המערכת להיות נגישה באופן תמידי.
   c. דרישת המעקב (9): יש לתחזק יומן אירועים (event log) ויומן שגיאות (error log). היומנים תמיד זמינים לצפייה. יומן האירועים מתעד את הפניות למערכת (אילו תרחישים מופעלים עם אילו פרמטרים). יומן השגיאות מתעד שגיאות של המערכת. תרחישים שליליים אינם שגיאות.
2. **דרישות פונקציונאליות**:
   a. **מערכת:** דרישות 1,3,4.
   b. **משתמשים:** מימוש כל הדרישות הפונקציונליות שפורטו בגרסה 0, מלבד מימוש דרישה II.4.2 – שינוי סוגי וכללי (מדיניות) קניה והנחה של חנות.
   c. קבוצות של שישה סטודנטים נדרשות למדל ולממש גם את דרישה II.3.3 – כתיבת ביקורת על מוצר.
3. **בניית מודל עבור ממשק משתמש**: ממשק משתמש הוא מערכת ריאקטיבית המגיבה לפעולות של המשתמש. הממשק מפעיל תרחישי שימוש ע"י מעקב אחרי סדרת הפעולות של תרחיש מצד המפעיל (Actor). נמדל את הממשק באמצעות מודל-מצבים היררכי (hierarchical statechart שמכונה SC). SC יתאר את סדרת הפעולות של תרחיש שימוש.
   סעיף הדגשים בהמשך כולל הסבר לגבי השימוש ב-SCs. יש למדל את התרחישים הבאים באמצעות SC:
   a. תרחיש ההזדהות במערכת (תרחיש II.1.4).
   b. תרחיש מינוי מנהל חנות (תרחיש II.4.5).

הנחיות למימוש:

מערכת בארכיטקטורה דו-שכבתית, בהתאם למודל הארכיטקטורה של המערכת.

1. **שכבת ה-domain**: לב המערכת בה ממומש המודל הלבן, והיא מחולקת לרכיבים אנכיים.
2. **שכבת השירות** (Service Layer): מנשק עבור סיפורי השימוש.
3. **תשתית בדיקות** ומימוש בדיקות קבלה ובדיקות פיתוח (יחידה ושילוב).
4. **מנגנון Traceability**: ניתן לעקוב אחר שינויי דרישות, קוד, מבחנים, והקשר ביניהם.

דגשים למימוש:

1. **מימוש המערכת**:
   a. מימוש דרישת רמת שירות (5) מצריך מימוש מקבילי של המערכת (דרישת הקיבול). יש לשים לב ששימוש לא זהיר במנעולים ובסינכרוניזציה יפגע בדרישת הזמינות.
   b. כדאי להוסיף ממשקים (interfaces) בכל מקום שבו יש סיכוי לשינוי מימוש בעתיד. כדאי גם להגדיר ממשק עבור אוספים של ישויות שממילא כבר יש להם מזהים.
2. **תחומי אחריות ואילוצי נכונות**: אילוצי נכונות צריך לאכוף באמצעות מבנה ופעולות המערכת, ולא להסתמך על בדיקות. יש לשים לב שפעולות מקביליות עלולות לגרום הפרה של אילוצי נכונות.
3. **תשתית הבדיקות:**
   a. אחראי הבדיקות מגדיר וממשש תשתית בדיקות (מכל הסוגים) עבור כל גרסאות הפרויקט, וכן מממש את מבחני הקבלה. תשתית ברורה ופשוטה לניהול חיונית להמשך חלק של הפיתוח.
   b. יש לספק גם בדיקות שליליות (כאלו שכישלונן הוא הצלחתן), ובדיקות למקביליות. הצעות:
      i. שני משתמשים מנסים לקנות את המוצר האחרון במקביל.
      ii. בעל חנות מוחק מוצר ובמקביל משתמש אחר מנסה לקנות את המוצר.
      iii. שני בעלי חנות מנסים למנות את אותו המשתמש למנהל במקביל.
   c. **הנחיות כלליות:** תרחישי בדיקה צריכים להיות ממוקדים. רצוי להשתמש ב-Mocks. בדיקות קבלה ניגשות למערכת דרך שכבת השירות בלבד.
4. **בניית מודל עבור ממשק משתמש**: מודל-מצבים הוא גרף שקודקודיו הם מצבים (States) והקשתות הן מעברים (Transitions). מעבר מורכב מתנאי (Guard), אירוע (Event) ופעולה (Action) (כולם אופציונאליים).
   a. פעולת משתמש היא מצב במודל המצבים (**State**).
   b. בקשת משתמש או תגובה של המערכת הן אירועים (**Event**).
   c. פעולת מערכת בעקבות בקשת משתמש היא פעולה (**Action**).
   d. תנאי בתרחיש השימוש הוא **Guard** על פעולה.

באתר הקורס יש הסבר והדגמה למודל SC המתאר פעולות מימשק

תוצרי גרסה 1:

מסמך המתודולוגיה מתאר את התוצרים השונים ואופן תיאורם. התוצרים בגרסה זו הם:

1. **דו"ח גרסה**: מוצג על ידי מנהל הגרסה ומכיל את פירוט ההספק לגרסה.
2. **מודלים עדכניים**
3. **מימוש המערכת**: לא יתקבל מימוש ללא בדיקות!
4. **מודל SC**

מצגת כיתה: Websockets (10 דקות)

הגרסה הבאה של מערכת המסחר תכלול Web client שישמש כממשק גרפי (GUI) המסוגל להציג התראות למשתמש בזמן אמת. מימוש ההתראות יעשה שימוש בפרוטוקול websockets המאפשר תקשורת דו כיוונית בין שרת ללקוחותיו.

1. הסביר/י את ההבדלים בין שימוש ב-websockets (secured) לבין השימוש בפרוטוקול HTTPS.
2. כברירת מחדל, בעת ניווט לכתובת URL חדשה (למשל, על ידי ניווט לדף אחר בצד הלקוח), חיבור ה-websocket אינו נשמר, הסביר/י כיצד ניתן להתמודד עם הבעיה.
3. הסבר מדוע תקשורת דו-כיוונית סותרת את הכיוניות של מודל הארכיטקטורה שלכם.

4. הסבר כיצד ניתן בכל זאת לממש תקשורת דו-כיוונית בלי להפר את הכיוונים של המודל (רמז: Observer):
   a. האם מימוש התקשורת הדו-כיוונית מצריך הגדרה של רכיב ארכיטקטורה (Component) חדש?
   b. בדיקות הקבלה צריכות גם לתמוך בהתראות זמן אמת, אבל ללא תלות ברכיב הממשק (ללא תלות בפרוטוקול websockets). איך אפשר לתמוך בבדיקות כאלו?