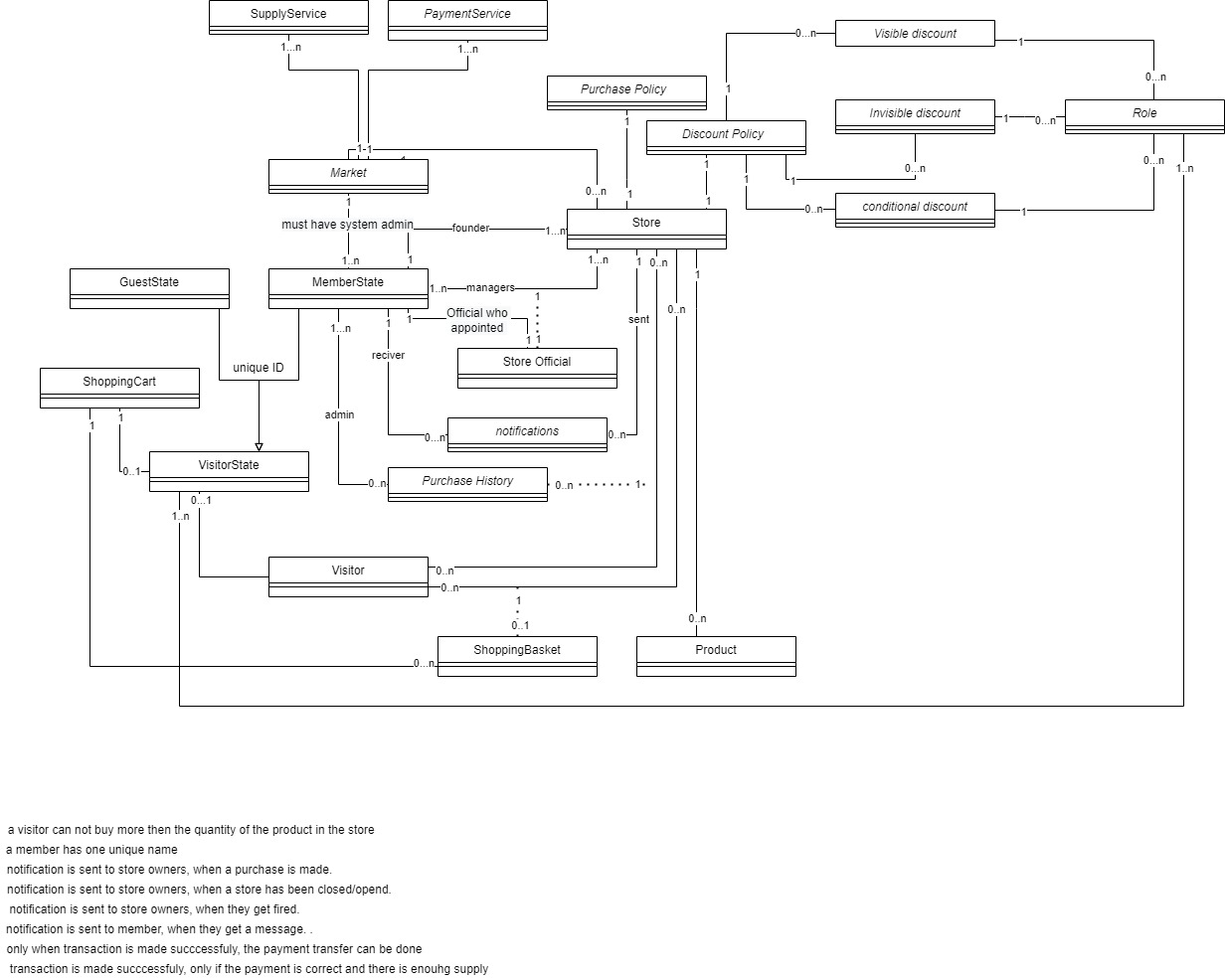
**דו"ח סכום – גרסה 1**

* תיקונים מגרסה 0: הכל תוקן
* מודל מחלקות לבן classDiagram – נעשה שינויים על פי דגשים מגרסה 0.
* Use-cases: תוקן ושוונה לתבנית המתאימה
* State chart: נמצא תחת stateChart ומכיל את הstate charts הרלוונטים.
* LOG: ממומש ב-sevice ונכתב בקובץ SystemLog.log בתיקיית Test
* מודל system architecture: תוקן על פי דגשים מגרסה 0.
* קוד:
  + שכבת ה- service:
    - מכילה את כל הדרישות, מחולקים בהערות לפי מסמך הדרישות. כך שלמשל דרישות משתמשים של מבקר אורח יקבלו בקוד ID II.1.נד
    - כמו כן מה שממומש זה מה שנדרש בגרסה זו, השאר רק מצוינים כפונקציה אך לא ממומשים.
    - מחזיקה 3 קונטרולרים: user, store and market controller
    - מחזיקה קלאס Response אשר אחראי לדבר עם שכבת ה-front בעתיד
    - מחזיקה אובייקטים שונים בתיקיית objects המדמים כעת אובייקטים אשר נקבל מהמשתמש ב-front
  + שכבת ה- domain:
    - מכילה את כל האובייקטים המפורטים בתרשים הקומפננטות (נמצא בקובץ תרחשי קבלה) , ביחד עם אובייקטים נוספים המשרתים את מטרת האובייקטים המרכזיים.
    - מכילה 3 אובייקטי controller הנ"ל
  + Tests:
    - מכילה unit tests לכל 3 הcontroller
    - מכילה תיקיית acceptance test אשר בודקת את ה-service ואת הכל ביחד

**מה לא עשינו:**

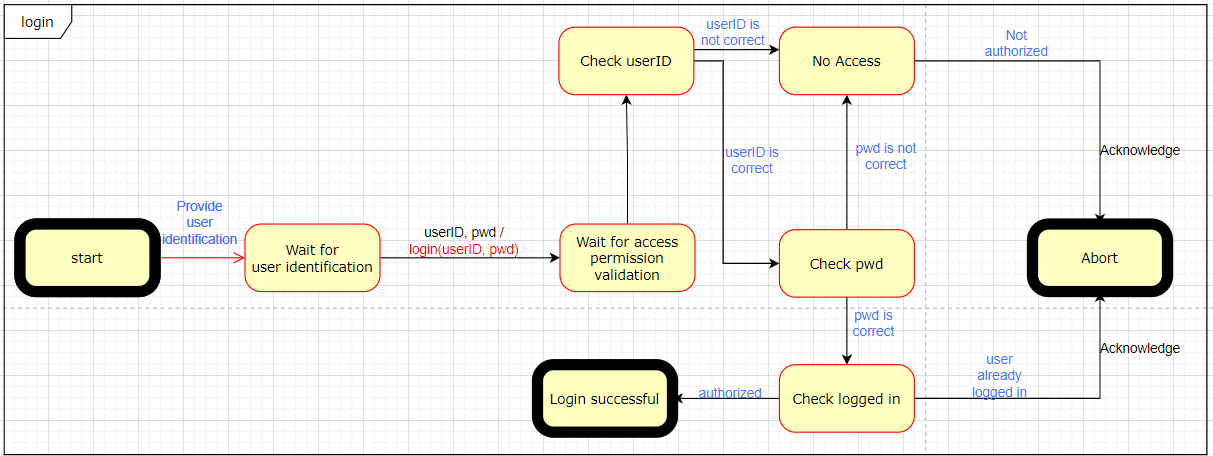
* מקביליות של יוזרים במערכת. בוצע מקביליות על קנייה, אבל לא על סקלביליות של המערכת
* לא הבנו בדיוק איך עובד ה- purchase policy ואיך זה נראה מבחינת ארכיטקטורה ומבחינת מימוש

Whit Diagram

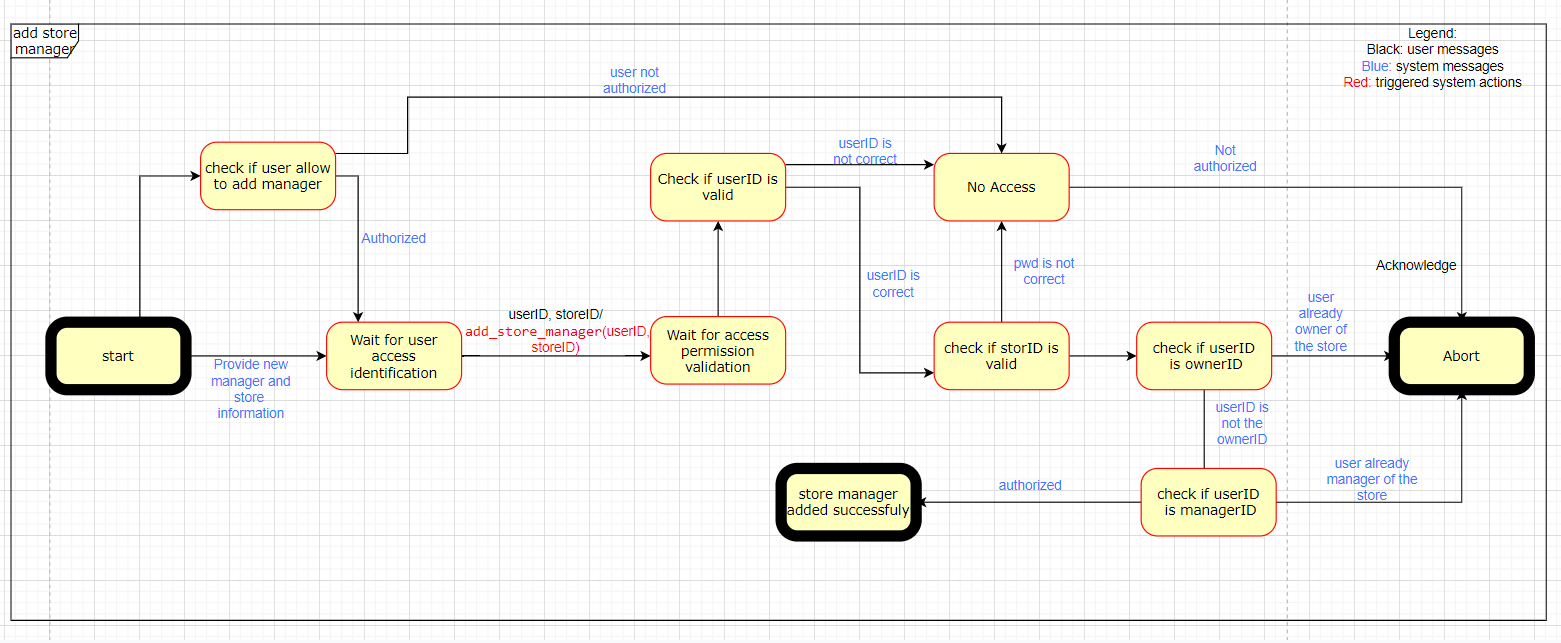


State Chart

Log in:



Add store manager:



תרחישי קבלה:

מערכת:

1. **פתיחה (אתחול) של מערכת המסחר (שוק)**:

* מקרה טוב: המערכת עולה ומאותחלת עם הנתונים הנכונים בצורה תקינה
* מקרה רע: המערכת עולה ומאותחלת בצורה לא תקינה או לא עולה בכלל

1. **שינוי/החלפה/הוספה של קשר עם שירותים חיצוניים:**

* מקרה טוב: השינוי/החלפה/הוספה של השירותים החיצונים בוצע בהצלחה עם הנתונים הנכונים.
* מקרה רע: השינוי/החלפה/הוספה של השירותים החיצוניים בוצע עם נתונים שגויים.

1. **תשלום**

* מקרה טוב: פניה למערכת התשלומים המוכרת על ידי השוק עם פרטי העסקה הנכונים בהצלחה וקבלת אישור בהתאם להצלחה או כישלון
* מקרה רע: פניה למערכת התשלומים המוכרת על ידי השוק עם פרטי העסקה הנכונים אבל לא תואמים למדיניות הרכישה שהוגדרה וקבלת אישור הצלחה.

1. **אספקה**

* מקרה טוב: פנייה למערכת אספקת מוצרים המוכרת על ידי השוק עם פרטי חבילה ופרטי לקוח נכונים וקבלת תשובה בהתאם (אישור אספקה או סירוב)
* מקרה רע: פניה למערכת אספקת מוצרים עם פרטי חבילה שגויים או פרטי לקוח שגויים או פנייה למערכת אספקת מוצרים לא עדכנית וביצוע העסקה עם הפרטים השגויים.

1. **התראות בזמן אמת**

* מקרה טוב: בעלי חנות מקבלים התראות בזמן אמת כאשר לקוח קנה מוצר מחנותם, כאשר החנות נסגרת או נפתחת מחדש או כאשר המינוי שלהם מוסר. נשלחת התראה למנוי כאשר קיבל הודעה/פניה.
* מקרה רע: בעלי חנות לא מקבלים התראות בזמן אמת כאשר לקוח קנה מוצר מחנותם או כאשר החנות נסגרת או נפתחת מחדש או כאשר המינוי שלהם מוסר או קבלת התראה לא נכונה. לא נשלחת התראה למנוי כאשר קיבלה הודעה/פניה או שנשלחת התראה עם הודעה לא נכונה.

1. **התראות מושהות**

* מקרה טוב: נשלחת התראה למנויים שאינם מבקרים בשוק בזמן ההתראה, ההתראה נשמרת ומוצגת ללקוח בכניסתו הבאה למערכת.
* מקרה רע: נשלחת התראה למנויים שאינם מבקרים בשוק בזמן ההתראה, ההתראה לא נשמרת ואינה מוצגת ללקוח בכניסתו הבאה למערכת, או שההתראה כלל לא נשלחת.

משתמשים:

1. **פעולות כלליות של מבקר-אורח:**
   1. **כניסה:**

* מקרה טוב: אורח נכנס למערכת, מקבל עגלת קניות ויכול לתפקד כקונה
* מקר רע: אורח נכנס למערכת, מקבל עגלת קניות אך אינו יכול לתפקד כקונה
  1. **יציאה:**
* מקרה טוב: אורח עוזב את השוק. עם עזיבתו, הוא מאבד את עגלת הקניות שלו ואינו מוגדר כמבקר
* מקר רע: אורח עוזב את השוק. עם עזיבתו, עגלת הקניות שלו נשמרת במערכת.
  1. **רישום למערכת המסחר:**
* מקרה טוב: אורח נרשם ע"י מסירת פרטים מזהים ייחודיים. בסיום תהליך הרישום האורח נרשם במערכת כמנוי.
* מקר רע: אורח נרשם ע"י מסירת פרטים מזהים ייחודיים. בסיום תהליך הרישום האורח לא נרשם במערכת כמנוי.
  1. **כניסה מזוהה**
* מקרה טוב: בסיום תהליך כניסה המשתמש מזוהה כמבקר-מנוי.
* מקר רע: בסיום תהליך כניסה המשתמש אינו מזוהה כמבקר-מנוי.

**1.4.1) וידוא סיסמא ומשתמש:**

* + מקרה טוב: אם הסיסמא והמשתמש תואמים מחזיר תקין
  + מקר רע: אם הסיסמא והמשתמש תואמים ומחזיר לא תקין

1. **פעולות קנייה של מבקר-אורח:**
   1. **קבלת מידע:**

* מקרה טוב: אורח מקבל מידע נכון על חנויות בשוק ועל המוצרים בחנויות.
* מקרה רע: אורח מקבל מידע לא נכון על חנויות בשוק ועל המוצרים בחנויות.
  1. **חיפוש מוצרים:**
* מקרה טוב: אורח מוצא את המוצר שחיפש אם קיים בשוק.
* מקרה רע: אורח לא מוצא את המוצר שחיפש אפילו שקיים בשוק.
  1. **שמירת מוצרים:**
* מקרה טוב: אורח שומר מוצר והמוצר נשמר בסל הקניות של הקונה, עבור חנות כלשהי.
* מקרה רע: אורח שומר מוצר והמוצר לא נשמר בסל הקניות של הקונה.
  1. **בדיקת תכולת עגלת הקניות וביצוע שינויים:**

**2.4.1) קבלת מידע על תכולת העגלה:**

- מקרה טוב: בקשה לקבלת מידע על תכולת העגלה וקבלה של המידע הנכון.

- מקרה רע: בקשה לקבלת מידע על תכולת העגלה וקבלה של מידע לא נכון.

**2.4.2) שינוי תכולת העגלה:**

- מקרה טוב: בקשה לשינוי תכולת העגלה ועדכון העגלה.

- מקרה רע: בקשה לשינוי תכולת העגלה אך העגלה לא מעודכנת.

* 1. **רכישת עגלת הקניות:**

**2.5.1) ביצוע תשלום:**

- מקרה טוב: התשלום בוצע בהצלחה עם כל הנתונים שהוזנו.

- מקרה רע: הסכום של התשלום לא תואם את הנתונים שהוזנו.

**2.5.1.1) וידוא מספיק מימון:**

- מקרה טוב: מחזיר שיש מספיק מימון אם יש מספיק.

- מקרה רע: מחזיר שיש מספיק מימון גם אם אין מספיק.

**2.5.2) ביצוע משלוח:**

- מקרה טוב: המשלוח בוצע בהצלחה עם כל הנתונים שהוזנו.

- מקרה רע: המשלוח בוצע עם נתונים שלא הוזנו.

**2.6.2.1) בדיקת מלאי:**

- מקרה טוב: מחזיר שיש את המוצרים במלאי אם הם באמת במלאי.

- מקרה רע: מחזיר שיש את המוצרים במלאי אפילו שהם לא במלאי.

1. **פעולות קנייה של מבקר-מנוי:**

**3.1) ביטול זיהוי (logout):**

- מקרה טוב: עם ביטול הזיהוי הקונה חוזר להיות אורח.

- מקרה רע: עם ביטול הזיהוי הקונה נשאר מנוי.

**3.1.1) שמירת עגלת קניות:**

- מקרה טוב: עם יציאת המנוי עגלת הקניות נשמרת במערכת

- מקרה רע: עם יציאת המנוי עגלת הקניות לא נשמרת במערכת

**3.2) פתיחת חנות:**

- מקרה טוב: מנוי של השוק פותח חנות, והוא מייסד/פותח החנות.

- מקרה רע: מנוי של השוק פותח חנות, והוא אינו מייסד/פותח החנות.

1. **פעולות של מבקר-מנוי בתפקידו כבעל חנות:**
   1. **ניהול מלא:**

**4.1.1) הוספת מוצר:**

- מקרה טוב: בעל-חנות מוסיף מוצר למלאי והמלאי מתעדכן.

- מקרה רע: בעל-חנות מוסיף מוצר למלאי והמלאי לא מתעדכן.

**4.1.2) מחיקת מוצר:**

- מקרה טוב: בעל-חנות מוחק מוצר מהמלאי והמלאי מתעדכן.

- מקרה רע: בעל-חנות מוחק מוצר מהמלאי והמלאי לא מתעדכן.

**4.1.3) שינוי מוצר:**

- מקרה טוב: בעל-חנות משנה פרטים של מוצר מהמלאי והמוצר מתעדכן.

- מקרה רע: בעל-חנות משנה פרטים של מוצר מהמלאי והמוצר לא מתעדכן.

* 1. **שינוי סוגי וכללי (מדיניות) קניה והנחה של חנות:**

- מקרה טוב: בעל-חנות משנה את סוגי הרכישה וההנחה האפשריים בחנות שבבעלותו, ואת מדיניות הקניה וההנחה עבור מוצרים בחנות והמדיניות מתעדכנת

- מקרה רע: בעל-חנות משנה את סוגי הרכישה וההנחה האפשריים בחנות שבבעלותו, ואת מדיניות הקניה וההנחה עבור מוצרים בחנות והמדיניות לא מתעדכנת

* 1. **מינוי בעל חנות:**

- מקרה טוב: בעל-חנות ממנה מנוי של השוק להיות בעל חנות שעדין איננו בעל-החנות.

- מקרה רע: בעל-חנות ממנה מנוי של השוק להיות בעל חנות כאשר הוא כבר בעל-החנות.

* 1. **מינוי מנהל חנות:**

- מקרה טוב: בעל-חנות ממנה מנוי של השוק להיות מנהל חנות שעדין איננו מנהל או בעל החנות.

- מקרה רע: בעל-חנות ממנה מנוי של השוק להיות מנהל חנות כאשר הוא כבר בעל או מנהל החנות.

* 1. **שינוי הרשאות של מנהל-חנות:**

- מקרה טוב: בעל-חנות משנה בהצלחה את אפשרויות הניהול עבור מנהל שהוא מינה.

- מקרה רע: בעל-חנות משנה את אפשרויות הניהול עבור מנהל שהוא מינה אך למנהל הממונה עדיין אין גישה לאפשרויות הניהול.

**4.9)**  **סגירת חנות:**

- מקרה טוב: מייסד-חנות סוגר חנות שפתח והיא הופכת לא פעילה. משתמשים שאינם בעלי החנות או מנהלי מערכת המסחר **אינם** יכולים לקבל מידע על חנות סגורה.

- מקרה רע: מייסד-חנות סוגר חנות שפתח והיא הופכת לא פעילה. משתמשים שאינם בעלי החנות או מנהלי מערכת המסחר **יכולים** לקבל מידע על חנות סגורה.

* 1. **בקשה למידע על תפקידים בחנות:**

- מקרה טוב: בעל-חנות מבקש לקבל מידע על בעלי התפקידים בחנות שבבעלותו, ואילו הרשאות יש למנהלים ואכן מקבל מידע תקין.

- מקרה רע: בעל-חנות מבקש לקבל מידע על בעלי התפקידים בחנות שבבעלותו, ואילו הרשאות יש למנהלים מקבל מידע **לא תקין**.

* 1. **קבלת מידע על היסטוריית רכישות בחנות:**

- מקרה טוב: קבלת מידע תקין על כל הרכישות שבוצעו בחנות.

- מקרה רע: קבלת מידע על הרכישות שבוצעו ללא מוצרים שהוסרו מהמלאי.

1. **פעולות של מבקר-מנוי בתפקידו כמנהל מערכת המסחר:**
   1. **קבלת מידע על היסטוריית רכישות בחנות:**

- מקרה טוב: מנהל של השוק מבקש לקבל מידע על היסטוריית רכישות של קונים וחנויות ומקבל מידע תקין.

- מקרה רע: מנהל של השוק מבקש לקבל מידע על היסטוריית רכישות של קונים וחנויות ומקבל **מידע לא נכון**.

מילון מונחים:

מערכת מסחר – המערכת שמנהלת את השוק.  
משתמש – מונח עבור קליינט של מערכת המסחר.  
מבקר – משתמש שנכנס מערכת המסחר.  
מנוי – משתמש **שנרשם** מערכת המסחר.  
מנהל מערכת מסחר – משתמש שמנהל את מערכת המסחר כולה.  
מנהל חנות – משתמש עם הרשאות חלקיות/מלאות לביצוע פעולות בחנות מסחר.  
בעל חנות – מנהל חנות עם כל ההרשאות לביצוע פעולות בחנות.  
מוכר – מונח כללי עבור בעל/מנהל חנות.

סוגי רכישה - ישנן מספר אפשרויות לרכישת מוצרים בחנות. ניתן למכור מוצרים שונים בסוגי רכישה שונים כל עוד סוג הרכישה מתאפשר ע"י מדיניות הקניה:

1. **קנייה מידית:** הקונה נדרש לשלם את הסכום הנקוב במוצר על מנת לקנות אותו באופן מיידי
2. **הגשת הצעת קנייה:** הקונה מציע מחיר עבור מוצר של חנות.
3. **מכירה פומבית**: היוזם לפעולת רכישה זו היא חנות, שמפרסמת הצעת מחיר התחלתי למוצר ומשך מכירה. כל עוד המכירה לא הסתיימה, מנויים יכולים להגיש הצעות עבור מוצר
4. **קנייה בהגרלה**: היוזם לפעולת רכישה זו היא חנות, שמפרסמת מחיר מוצר ותקופת מכירה. באפשרות קניה זו, המנויים רוכשים סיכויי זכייה במוצר

מדיניות רכישה: מדיניות קניה מגדירה מי רשאי לקנות מוצרים בחנות, מהם מסלולי הרכישה האפשריים, ומהם הכללים החלים עליהם. למשל, מינימום פריטים למוצר ברכישה. מדיניות קנייה מוגדרת עבור חנות ויכולה לחול על מוצרים או על לקוחות או על שניהם.

סוגי הנחה: יש מספר סוגי הנחות אפשריים

1. **הנחה גלויה:** מורכבת מאחוז הנחה ומשך הנחה, ויכולה להתייחס למוצר אחד, קבוצת מוצרים, או לחול על כלל המוצרים בחנות. במהלך תקופת ההנחה, מחיר המוצר עליו חלה ההנחה, מוצג לקונים לאחר הנחה, כאשר בנוסף, הקונים רואים גם את המחיר המקורי ואת מועד סיום ההנחה.
2. **הנחה מותנית:** מורכבת מאחוז הנחה, משך הנחה, ותנאי לקבלת ההנחה. בדומה להנחה גלויה, גם הנחה זו מוצגת לקונים, אך מגדירה תנאים בהם צריך לעמוד כדי לקבל את ההנחה.
3. **הנחה נסתרת:** מורכבת מ אחוז הנחה, משך הנחה, וקוד הנחה. בניגוד לסוגי ההנחות הקודמות, הנחה זו אינה גלויה לקונים. קונה יכול להזין קוד הנחה במעמד הקניה, ואם קוד ההנחה בתוקף ומוגדר עבור המוצר, הקונה מקבל את ההנחה הנקובה. 5

מדיניות הנחה: מדיניות הנחה מגדירה מי רשאי לקבל הנחה, מהם סוגי ההנחה, ומהם הכללים החלים עליהם. למשל, מינימום עלות לקנייה כוללת. מדיניות הנחה מוגדרת עבור חנות ויכולה לחול על מוצרים או על לקוחות או על שניהם.

**Use case:**

Initialize system (I.1)**:**

**Actors:** Admin

**Parameters:** admin\_id, admin\_password, payment\_service, supply\_service

**Pre-conditions:**

**Post-conditions:** System is initialized

**Scenario:** An admin of the system is initiated the system by entering his admin user, and the connection to the relevant services.

Contact payment service (I.3):

**Actors:** Admin

**Parameters:** payment\_info

**Pre-conditions:** System is initialized

**Post-conditions:** if all data is correct, transaction approve, else faild.

**Scenario:** payment transaction with the current payment service, with the given payment information. If all is correct, transaction is successful, else transaction faild.

Contact supply service (I.4):

**Actors:** Admin

**Parameters:** package\_info

**Pre-conditions:** System is initialized

**Post-conditions:** Supply service gets package information

**Scenario:** A request for a delivery is made by the market system and relevant supply service get the package information.

Enter as guest (II.1.1)**:**

**Actors:** Visitor

**Parameters:**

**Pre-conditions:** System is initialized

**Post-conditions:** Visitor have unique Id and shopping cart

**Scenario:** A user open the system, he enter as a guest and get his own unique id and cart.

Exit system(II.1.2)**:**

**Actors:** Visitor

**Parameters:** user\_id

**Pre-conditions:** Visitor is identified by the system

**Post-conditions:** guest matching user\_id cannot be detected, member is not login.

**Scenario**: A guest need to be a visitor to exits the system (enter as guest or log out as a member), his cart is not saved.

Register (II.1.3)**:**

**Actors:** Visitor

**Parameters:** user\_id**,** user\_info

**Pre-conditions:** Visitor is not identified by the system

**Post-conditions:** Visitor is identified by the system

**Scenario**: A guest can register to the system and become a member by giving personal unique information, to enter as a member, a guest still needs to login.

Login (II.1.4)**:**

**Actors:** Visitor

**Parameters:** guest\_id**,** username, password

**Pre-conditions:** Visitor

**Post-conditions:** Visitor is identified by the system as a member

**Scenario**: A guest can login to the system to identify as a member by giving his personal information (need to be registered to the system first). If the information doesn't match to what is in the system, should return an error message that says the user doesn't exists and the guest need to enter information again. If password doesn’t correct, should send error message of wrong information. If the user is already connected, should send error message that the user is already logged in.

Get information about stores (II.2.1)**:**

**Actors:** Visitor

**Parameters:** user\_id, store\_id

**Pre-conditions:** System is initialized

**Post-conditions:** Visitor can see store information

**Scenario**: A guest can get information about the stores in the system and their products by giving the store id.

Product search (II.2.2)**:**

**Actors:** Visitor

**Parameters:** user\_id, product\_filters

**Pre-conditions:** System is initialized, visitor in the system

**Post-conditions:** Visitor can see products information

**Scenario**: A guest can search for products in the system, he can search by product name or category and can filter the results by price range, product's rate and store's rate.

Add product to cart (II.2.3)**:**

**Actors:** Visitor

**Parameters:** user\_id, store\_id, product\_id, quantity

**Pre-conditions:** System is initialized, visitor in the system

**Post-conditions:** product is added to visitor's shopping cart

**Scenario**: A guest can add a product to his basket that is held by his cart by specifying the product id, if there is not enough items of that product in the inventory the action can fail and the product won't be added.

Get shop cart (II.2.4.1)**:**

**Actors:** Visitor

**Parameters:** user\_id

**Pre-conditions:** System is initialized, visitor in the system

**Post-conditions:** Visitor can see his shopping cart information

**Scenario**: A guest can get his shopping cart so he can see the products inside it.

Remove product from shopping cart (II.2.4.2)**:**

**Actors:** Visitor in the system

**User Parameters:** Product id, store id, user id, quantity

**Pre-conditions:** product is within the user's and the store's shopping basket withing the user's shopping cart

**Post-conditions:** product removed from the basket

**Main scenario:** A guest can remove product from his shopping cart by specifying the product id to be removed.

Immediate buy **(**II.2.5.1)**:**

**Actors:** Visitor in the system

**User Parameters:** user id, payment information

**Pre-conditions:** user has a shopping cart loaded with products (in baskets)

**Post-conditions:** user cart is empty, stores reduced quantities according to the shopping cart quantities, payment taken from the user payment info, supply service received the task

**Main scenario:** A visitor can buy products immediately by giving payment info, meaning he can buy his cart if all the products are available in the inventory (if some product are not available the action is cancelled and the visitor get notified) and he is allowed to immediate buy according to each store's Purchase policy (if he is not allowed to buy according to the store's Purchase policy, the action is cancelled and the visitor is notified). He also get discount according to each store's Discount policy. After checking that inventory his full and checking the store's policies, the transaction happens, giving back a response about failure or success. If a failure response is given, the system cancels the action.

Bid(II.2.5.2):

**Actors:** Visitor in the system, Store Owners

**User Parameters:** Product id, store id, user id

**Pre-conditions:** product is allowed to be purchased in a bid, product is in the store stock

**Post-conditions:** product quantity removed from the store's stock

**Main scenario:** A visitor can buy a product in a Bid giving payment info and offering a price, only if the product is available in the inventory (if product is not available the action is cancelled and the visitor get notified) and he is allowed to Bid according to the store's Purchase policy (if he is not allowed to buy according to the store's Purchase policy, the action is cancelled and the visitor is notified). After checking that inventory his full and checking the store's policy and getting a confirmation from the store owners about the price he bided, the transaction happens, giving back a response about failure or success. If a failure response is given, the system cancels the action.

Auction(II.2.5.3):

**Actors:** Visitor in the system

**User Parameters:** Product id, store id, user id

**Pre-conditions:** product is allowed to be purchased in an auction, product is in the store stock

**Post-conditions:** product removed from the store's stock

**Main scenario:** A visitor can buy a product in an Auction giving payment info and offering a price (higher than the last offer that exists), only if the auction is ongoing and he is allowed to buy according to the store's Purchase policy (if he is not allowed to buy according to the store's Purchase policy, the action is cancelled and the visitor is notified). After checking that the store's policy, the offer is accepted. In the end of the auction, the visitor with the highest offer wins and the transaction happens giving back a response about failure or success. If a failure response is given, the system cancels the action.

Lottery(II.2.5.4):

**Actors:** Visitor in the system

**User Parameters:** Product id, store id, user id, cost of odds

**Pre-conditions:** product is allowed to be purchased in a lottery, product is in the store stock

**Post-conditions:** product removed from the store's stock

**Main scenario:** A visitor can buy a product in a Lottery giving payment info and offering a price (the price need to be smaller than the product's price), only if the Lottery is ongoing and he is allowed to buy according to the store's Purchase policy (if he is not allowed to buy according to the store's Purchase policy, the action is cancelled and the visitor is notified). After checking the store's policy the offer is accepted. The Lottery ends when the combined visitors offers equal or greater to the product price. The visitor get odds according to how much he paid, if the visitor wins the lottery, the transaction happens, giving back a response about failure or success. If a failure response is given, the system cancels the action.

Log out (II.3.1)**:**

**Actors:** Visitor in the system

**User Parameters:** user id

**Pre-conditions:** Visitor status is Member and he is logged in

**Post-conditions:** Visitor status is Guest and he is not logged in

**Main scenario:** A member can log out of the system making him a visitor again, his cart is saved for him if he log in again.

Open store (II.3.2)**:**

**Actor**: member

**Parameters:** user id, store's name.

**Pre-Condition:** visitor is a registered member.

**Post-Condition:** the store is added to the system and the member who opened it is its founder.

**Main-Scenario:**

A member can open a store in the system by giving the store's name. At the end of the action the member becomes that store's founder.

Add product to inventory (II.4.1.1)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id, store id, product id, quantity.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** the product it added to the inventory.

**Main-Scenario:**

A Manager/Store owner can add product to the store's inventory by giving the product id and quantity, if the product does not exists, a new product is added to the system with default values.

Remove product from inventory (II.4.1.2)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id, store id, product id, quantity.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** the product it removed from the inventory.

**Main-Scenario:**

A Manager/Store owner can remove products from his store by giving the product id and quantity, if there is not enough products in the inventory to remove, the action is cancelled.

Edit product info (II.4.1.3)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id, product id, new product info.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** the product is edited in the system.

**Main-Scenario:**

A Manager/Store owner can edit products from his store by giving the product id and new product info.

Change buying policy (II.4.2.1)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id, new buying policy.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** the policy is edited in the store.

**Main-Scenario:**

A Manager/Store owner can change the buying policy of his store by giving the store's id and new buying policy info.

Change discount policy (II.4.2.2):

**Actor**: Manager/Store owner.

**Parameters:** user id store id, new discount policy.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** the policy is edited in the store.

**Main-Scenario:**

A Manager/Store owner can change the discount policy of his store by giving the store's id and new discount policy info.

Add store owner (II.4.4)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id, new owner id.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** The new owner is added to the store.

**Main-Scenario:**

A Manager/Store owner can add a new store owner, a store owner can have only one appointee, if a Manager/Store tries to add existing store owner, the action in cancelled.

Add store manager (II.4.6)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id, new manager id.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** The new manager is added to the store.

**Main-Scenario:**

A Manager/Store owner can add a new manager, a manager can have only one appointee, if a Manager/Store tries to add existing manager, the action in cancelled.

Change manager permissions (II.4.7)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id, manager id, new Permission.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** The new permissions are edited to the manager.

**Main-Scenario:**

A Manager/Store owner can change another manager permission (adding or removing actions). He need to specify which action to change and if he want to deny or give access.

Close store (II.4.9)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:** The Store becomes inactive in the system.

**Main-Scenario:**

A Store owner can close a store he owns, making the store inactive, the store still exists in the system but visitors doesn't have access to the store.

Get store roles(II.4.11)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:**

**Main-Scenario:**

A Manager/Store owner can get the roles on his store, giving back users ids and the rolls and permissions they got.

Get store purchase history(II.4.13)**:**

**Actor**: Manager/Store owner.

**Parameters:** user id store id.

**Pre-Condition:** Manager has permission to make this action.

**Post-Condition:**

**Main-Scenario:**

A Manager/Store owner can get the store purchase history.

Get system purchase history(II.6.4)**:**

**Actor**: Admin

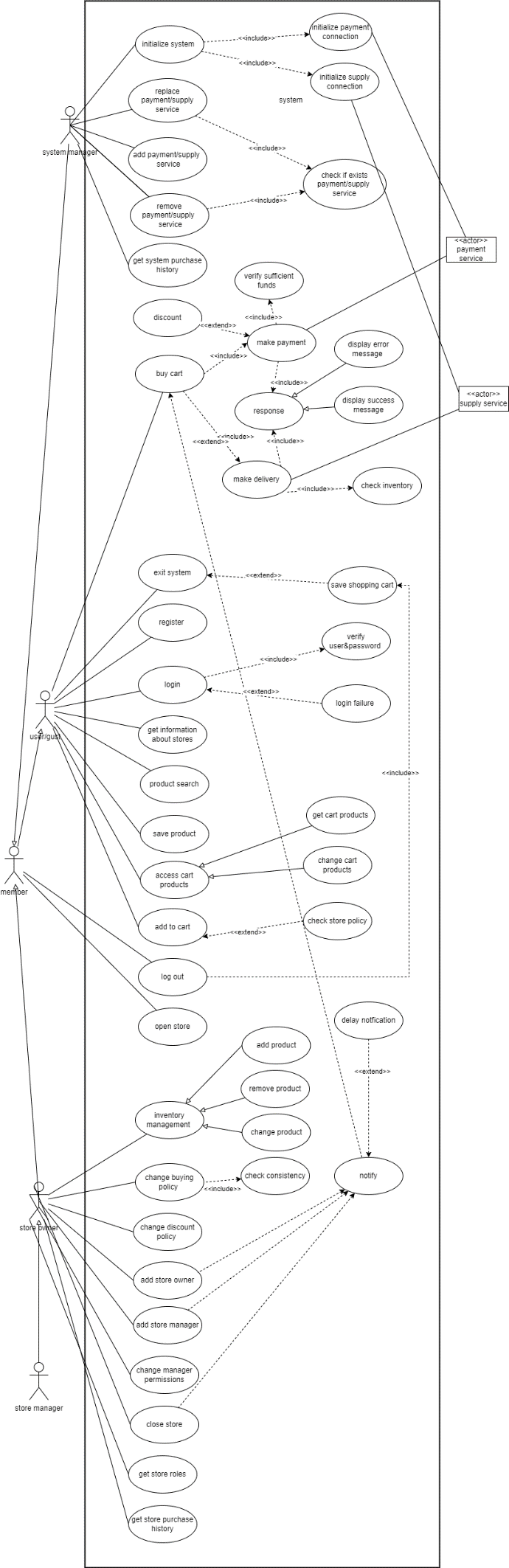
**Parameters:** user id store id.

**Pre-Condition:** Admin is registered as admin in the system

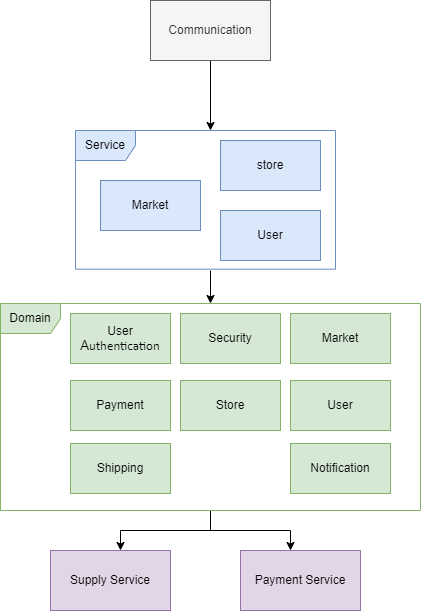
**Post-Condition:**

**Main-Scenario:**

An admin can get all stores purchase history.



**System Architecture**



**תרחישי בדיקה עבור ממשק גרפי:**

1. Register and login
   1. משתמש 1 מנסה להתחבר (לפני רישום)
   2. לא מצליח
   3. מנסה להירשם – מצליח
   4. מנסה להתחבר - מצליח
2. הוספת חנות ומוצרים:
   1. משתמש 1 יוצר חנות ST ומכניס לתוכה שתי מוצרי נעליים שונים בעלי מחיר שונה
   2. משתמש 2 מתחבר ומחפש את החנות ST
   3. בתוך החנות יש את המוצרים שמשתמש 1 הוסיף
3. הוספה לעגלת קניות וקנייה:
   1. משתמש 2 מוסיף את המוצרים של החנות של משתמש 1
   2. נכנס לעגלת קניות והם אמורים להופיע
   3. מנסה לשלם ועובר לדף תשלום
   4. ממלא פרטים בשביל שירותים חיצוניים
   5. משלם
      1. אם מילא לא נכון, או פרטים שאמורים להיות לא טובים על פי הדקומנטציה של השירותים החיצוניים
         1. יקבל שגיאה בהתאם
      2. אם מילא נכון
         1. יקבל הודעה הצלחה
         2. המוצרים ירדו מעגלת הקניות
4. התראות בזמן אמת:
   1. פותחים 2 חלונות שונים שבכל חלון יש משתמש אחר
   2. משתמש 1 פותח חנות ומוסיף מוצרים
   3. משתמש 2 קונה את המוצרים
   4. בזמן אמת משתמש 1 אמור לקבל התראה על כך שקנו מוצרים מהחנות שלו
   5. במידה והוא לא מחובר, כשיתחבר יקבל התראה על כך בהתאם.

**שאלות:**

* Payment service: אנחנו קוראים לו מהדומיין ומשתמשים לו מהסריבס.  
  לא הבנו איך ניתן לסמלץ את זה כראוי. אם אפשר הסבר.