



**הטלת אחריות לקיום אילוצי נכונות:**

1. "**למנוי יש שם יחיד המזהה אותו במערכת**".

* המחלקה האוכפת את האילוץ: User Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: בעת הרשמה של משתמש (אורח) למערכת, יתבצע ווידוא כי שם המשתמש שבחר אינו תפוס. במידה והשם שבחר תפוס – תוצג למשתמש הודעה מתאימה והוא יתבקש לבחור בשם אחר.

1. "**יש למערכת לפחות מנהל אחד. מנהל-מערכת חייב להיות מנוי (עבר תהליך רישום)**".

* המחלקה האוכפת את האילוץ: User Component Facade ו- Administrator.
* כיצד האילוץ נאכף ע"י המחלקה: בעת עליית המערכת בפעם הראשונה לאוויר, במידה ואין מנהל-מערכת אשר רשום במערכת (כלומר, לא נרשם ידנית במסד הנתונים שלה), המערכת תבקש ליצור הרשמה של אחד כזה בכדי שניתן יהיה להשתמש בה (User component Facade). כמו כן, לאורח אין גישה/הרשאות מלבד רכישה של מוצרים מחנויות/מוכרים, על כן בעת מינוי של משתמש למנהל-מערכת, תתבצע בדיקה האם המשתמש הנ"ל רשום למערכת (Administrator).

1. "**בעל חנות או מנהל חנות חייב להיות מנוי**".

* המחלקה האוכפת את האילוץ: Store Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: כפי שהוסבר באילוץ הקודם, לאורח אין גישה/הרשאות מלבד רכישה של מוצרים מחנויות/מוכרים, על כן בעת מינוי של משתמש לתפקיד של בעל חנות או מנהל חנות – תתבצע בדיקה האם המשתמש הנ"ל רשום למערכת.

1. "**פעולות בשוק מבוצעות רק ע"י משתמשים המבקרים בשוק**".

* המחלקה האוכפת את האילוץ: User Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: כל משתמש אשר מבקר בשוק מקבל זיהוי ייחודי כאשר הוא נכנס לשוק – כלומר, הוא הופך ל-Guest במערכת, וככזה הוא יכול לבצע פעולות בשוק. מי שאינו אורח של השוק, אינו יכול לבצע פעולות בו.

1. "**לחנות פעילה (שאינה סגורה) חייב להיות לפחות בעל-חנות אחד**".

* המחלקה האוכפת את האילוץ: Store Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: בעת שהחנות נפתחת, נקבע לה founder. באם ה-founder רוצה לעזוב את החנות – עליו לסגור אותה, או להתחלף עם אחד מבעלי-החנות (אם קיימים) או למנות אחד כזה. החנות לא תאפשר להימצא ללא בעלים אחד לפחות (founder).

1. **א.** "**לחנות חייבים להיות מוגדרים תהליכי קניה (רכישה) והנחה. יתכן ויש ברירת מחדל עבור אופן קנייה ועבור סוג הנחה**".

* המחלקה האוכפת את האילוץ: Store Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: בעת פתיחת החנות (לראשונה), חייבים להיות מוגדרים תהליכי קניה והנחה – אחרת החנות לא תתאפשר להיפתח. לכן, בעת פתיחת החנות תחול ברירת המחדל. ניתן יהיה לערוך את תהליכי הקניה וההנחה, אך לא ניתן להסירם מן החנות בשום עת.

**ב.** "**לחנות חייבים להיות מוגדרים מדיניות רכישה ומדיניות הנחה. תיתכן ברירת מחדל של כללי רכישה או חישוב הנחה, כמו למשל "אין מגבלות רכישה או הנחה**".

* המחלקה האוכפת את האילוץ: Store Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: בעת פתיחת החנות (לראשונה), חייבים להיות מוגדרים מדיניות רכישה ומדיניות הנחה – אחרת החנות לא תתאפשר להיפתח. לכן, בעת פתיחת החנות תחול ברירת המחדל. ניתן לערוך את המדיניות, אך לא ניתן להסירה מן החנות בשום עת.

1. "**לקונה יש עגלת קניות יחידה, המורכבת מאוסף כל סלי הקנייה שלו. לקונה יש לכל היותר סל קניות יחיד עבור חנות כלשהי**".

* המחלקה האוכפת את האילוץ: User.
* כיצד האילוץ נאכף ע"י המחלקה: בעת תהליך רכישה של קונה במערכת, תיפתח עבורו עגלת קניות אחת ויחידה, ולא תתאפשר פתיחה של מספר עגלות קניה עבור אותו הקונה (כן ניתן לרוקן את עגלת הקניות).

כמו כן, בעבור כל חנות שבה הוא קונה (כלומר, בחר לפחות במוצר אחד שאותו הוא רוצה לקנות מהחנות), ייפתח עבורו סל קניות יחיד בלבד – לא תתאפשר פתיחה של מספרי סלי קנייה (ניתן לרוקן את סל הקנייה שלו בחנות כלשהי). המחלקה User תאכוף את הדרישה בכך שלא תאפשר למשתמש להיות בעלים של יותר מעגלת קניות אחת.

1. "**עגלת הקניות של קונה (אורח או מנוי) הינה בבעלותו הבלעדית ואינה ניתנת לשינוי על ידי שום משתמש אחר**".

* המחלקה האוכפת את האילוץ: User.
* כיצד האילוץ נאכף ע"י המחלקה: המחלקה תפתח בעבור כל משתמש עגלת קניות ייחודית, אשר ניתנת לשינוי אך ורק על ידו – וכן גם מוצגת אך ורק לעיניו הפרטיות שלו.

1. "**ניתן לקנות פריטים של מוצר בחנות לכל היותר בכמות הקיימת במלאי**".

* המחלקה האוכפת את האילוץ: Store Component Façade ו-Observer.
* כיצד האילוץ נאכף ע"י המחלקה: בעת רכישה של מוצר מסוים בחנות כלשהי, ע"י קונה במערכת – תוצג לו כמות המלאי הזמינה באותה חנות מאותו המוצר (אשר מתעדכנת בזמן אמת, כלומר אם בדיוק התבצעה רכישה של המוצר הנ"ל על ידי קונה אחר – הדבר יתבטא בעדכון "בלייב" של הכמות של אותו המוצר – ע"י ה-Observer), ובמידה והקונה בחר בכמות גדולה יותר מהכמות הזמינה בחנות – תוצג לו הודעה מתאימה וההזמנה לא תתבצע.

1. **א.** "**ניתן לגבות כסף מקונים רק עבור עסקאות שבוצעו ולגבות רק את הסכומים שהוצהרו**".

* המחלקה האוכפת את האילוץ: Cart.
* כיצד האילוץ נאכף ע"י המחלקה: הרכישה מתבצעת במערכת אך ורק כאשר הלקוח בוחר בכך – כלומר, עליו לבחור בביצוע רכישה של המוצרים שבחר בכדי שתתבצע גבייה כספית ממנו עבורן. כמו כן, הסכומים נקבעים בעת הוספת המוצרים לסל הקניה שלו – כלומר, אם המחיר שונה לאחר מכן ע"י המוכר, הוא לא ישתנה בעבור אותו הקונה משום שהוא כבר בחר במוצר במחיר הקודם שלו. בנוסף, פרטי האשראי של קונה מסוים הינם חסויים ואינם גלויים לאף אחד אחר מלבדו.

**ב.** "**תהליך קנייה מסתיים בהצלחה רק אם שולם הסכום הדרוש ורק אם האספקה אושרה**".

- המחלקה האוכפת את האילוץ: Cart.

* כיצד האילוץ נאכף ע"י המחלקה: הרכישה מאושרת אך ורק לאחר שההובלה אושרה ע"י ה-“Delivery Module” – שאחראי על בדיקת ההובלה, **וגם** בוצעה ואומתה ע"י ה-“Payment System”, אשר אחראי על בדיקת ואישור התשלום. על שניהם אחראי ה-Order Component Facade בעת ביצוע הרכישה ע"י הקונה.

**ג.** "**מוכר יכול לקבל תשלום רק כתוצאה מתהליך קנייה מוצלח**".

* המחלקה האוכפת את האילוץ: Order Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: רק לאחר שתהליך הקנייה הסתיים בהצלחה (אילוץ 10 ב'), התשלום יועבר למוכר. במידה ותהליך הקנייה לא הסתיים בהצלחה – המוכר לא יקבל את התשלום והרכישה תבוטל.

1. **א.** "**נדרש לפחות קשר אחד לשירות גביית כספים**".

* המחלקה האוכפת את האילוץ: Order Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: המחלקה הנ"ל יוצרת קשר עם ה-“Payment System” בעת ביצוע רכישה על ידי קונה, בכדי לוודא שהתשלום שמבוצע על ידו מאושר והינו תקין.

**ב.** "**נדרש לפחות קשר אחד לשירותי אספקה**".

* המחלקה האוכפת את האילוץ: Order Component Facade.
* כיצד האילוץ נאכף ע"י המחלקה: המחלקה הנ"ל יוצרת קשר עם ה-“Delivery Module” בעת ביצוע רכישה על ידי קונה, בכדי לוודא שניתן לספק את המוצרים שנבחרו על ידו.

**Application’s Glossary**

| **Term** | **Explanation** | **Comment** |
| --- | --- | --- |
| **Administrator** | A registered user of the system, which has access to some specific tools, such as: opening/closing stores, monitoring user’s actions, viewing order’s history, etc. | An admin can be appointed by another admin only. |
| **Bucket** | A list of all the products which a buyer wants to purchase - from a SPECIFIC store, that he chose items from. | A bucket contains products that were chosen by the buyer from the same store. |
| **Cart** | A list of all the products which a buyer wants to purchase (from all of the stores that he chose from). | A buyer can choose different products from many stores and pay for all of them in the same purchase. |
| **Contact information** | The email address and the phone number of a specific Member. | Each member should fill this mandatory information when registering. |
| **Database Module** | An external system that stores all of the market’s information, such as: user’s information, store’s information – including products, stock, purchases history, etc. |  |
| **Delivery Module** | An external system that checks the availability of a delivery for the desired products that has been purchased by the buyer – and informs the market about it. |  |
| **Employment** | The connection between a seller and a store:  A seller may be a store owner or a store manager. | A store may have a lot of employees, of different roles. |
| **Founder** | The first owner of a specific store.  Has all of the store’s management permissions | Hence, the Member who has opened the store. |
| **Guest** | A buyer, who isn’t logged-in into the system (or even not registered to the system). | Any user who isn’t logged-in nor registered, is considered as a guest. |
| **Item** | A product which the user chooses from a specific store (which he wants to purchase). | A specific product which the user chooses from a store – turns into an item which the user may checkout with and pay for. |
| **Manager** | An employee of the store (a seller) who has some additional permissions, in addition to the basic seller. The permissions may be set by an owner of the store – by defining management permissions for this employee when hiring him. | A manager may be ‘promoted’ – meaning that he may have his permissions upgraded, or ‘demoted’. |
| **Market** | The whole system is called a ‘market’ – which has stores, users and orders. |  |
| **Management permissions** | Each Store Manager (Manager) may have different permissions that has been set for him, in a specific store, by his employer (one of the owners of this store). | The permissions may vary from the stock management of the store, and to managing the employees of the store. |
| **Default managing permissions** | The most basic permissions that may be set for a Store Manager. | Will be decided on later on. |
| **Member** | A guest that has logged in with his unique password. | Anyone can become a member by registering to the system. |
| **Notification** | A general component which can be shared between users of the system.  A notification may be an email notification, popup notification, special notification or a message. | For example: a buyer may contact a seller via a message, or a seller may contact a buyer by sending him a popup notification. |
| **Email Notification** | A notification that may be sent by an email, by either a buyer or a seller – whereas if a seller sends an email notification to a buyer – the buyer must be a registered member of the system. | This notification may be sent only to a registered member of the system (the sending user may be a guest), or to unregistered user (guest) who gets his receipt by email. |
| **Popup Notification** | A notification that may be presented to potential buyers of the system, ONLY by a seller. | This kind of notification may be presented to a buyer of the system, by a seller. |
| **Special Notification** | A notification that may be sent from the store to its employees, such as: “The products stock is 0”. Or a complaint that has been sent by a buyer to the store, etc. |  |
| **Message** | A private notification which can be sent from and to a registered user of the system – hence, between two members. | A guest cannot receive or send messages. |
| **Owner** | An owner of a specific store.  Has access to performing some actions, in regard to the store’s management, such as: Appointing new sellers, store managers, stock management, etc. | A store may have many owners. |
| **Payment methods** | The acceptable payment methods, such as: PayPal, Apple-Pay, Google-Pay, etc. | The accepted payment methods will be decided on later on. |
| **Product** | A unique item (per store), which can be bought by a buyer - if it has a sufficient amount of quantity.  Each product belongs to one and only store or a seller (until it is b  ought). | Products may be sold by a sole seller, and not in a particular shop. |
| **Purchase Policy** | A ‘contract’ between the store and the buyers of the store – such as: payment methods, products return policy, etc. | Each store must have a purchase policy set for it when being opened. |
| **Purchase Strategy** | May be a restriction of the buying policy, such as ‘forcing’ the buyer to buy at least 3 products from the store, or buying at most 5 products from the store. May also be in the form of a discount for some products of the store. |  |
| **Payment System** | An external system that charges the money from the buyer and informs the market that it has ben successfully done (or not). |  |
| **Store** | A virtual store component that must have a founder, purchase policy, purc  hase strategy and products (available stock). The store may also have managers and owners. The store sells its products to the buyers of the system.  Each store may have a lot of products and must have at least one product | A store cannot be opened without having a founder, and neither be active without one. |
| **Stock** | The items in a specific store, that are available to be purchased. | Each store has its own stock. |
| **User** | A person who is interacting with the system; he can be either a member or a guest. | Any User is a potential buyer – and is considered as one. |

**Use cases**

1) Use case: **Assigning a member as a manager of a store**.

-Actors: store owner

-Precondition: the member must be a registered, and not a manager of this store.

-Input Parameters: user identifier to assign as manager, store identifier.

-Actions:

1. The store owner types the member and store identifiers to the system.
2. The store owner selects managing permission for the new manager.
3. The userRepo promotes the member to be a manager with the appropriate permissions .

2) Use case: **A member signs in.**

-Actors: member of the system.

-Precondition: the member is registered and not already signed-in in the system.

-Input Parameters: username, password

-Actions:

1. The member types In username and password.
2. The system validates the data:
   1. if data is correct the member will be signed in
   2. if data is incorrect, the member will receive an error message

3) Use case: **A guest adds a product from a store to the cart**.

-Actors: guest

-Precondition: The product must be in the store stock.

-Input Parameters: product id, quantity (maybe more options)

-Actions:

1. The guest chooses a product from a given products list.
2. The guest selects the "add to cart" option.
3. The guest selects purchase strategy , s.a. quantity , color if it is relevant… .

4.1 if the quantity is bigger than the stock then the system alerts.

1. The mediator accesses the guest’s cart via user component and adds the product to the corresponding store basket.

5.1. if it's the first product from this store , then initiate the basket.

1. The mediator will update number of products.

5.1. if it is from a store then it updates the store’s quantity.

5.2. if it is not from a store then it updates the seller’s quantity.

1. The cart updates cart's total cost (including sales and discounts).

4) Use case: **A guest checks out**.

-Actors: guest

-Precondition: The cart contains at least one product.

-Input Parameters: None

-Actions:

1. The guest chooses the option check-out.
2. If the guest is a guest the system suggests the visitor to sign in.

2.1 if the guest signs in -> use case: A member checks out.

1. the system asks the guest to type in the address and contact information.
2. The guest types the address and contact information.
3. The system will send to the external delivery system the information about the delivery.

5.1 if the external delivery system throws an error, the system will inform the guest it or about the error and will disapprove the checkout.

1. The system shows the final price (including the delivery and sales) and the purchase options.
2. The guest selects the desired option and fills in the required details.
3. The system validates that the guest can afford the cart via external payment system.

8.1 if the external payment system throws an error, the system will inform the guest about the error and will disapprove the checkout.

1. the system confirms the deal.
2. the system will access communication component and will send a receipt using email notification.

5) Use case: **A member checks out**.

-Actors: member

-Precondition: The cart contains at least one product.

-Input Parameters: None

-Actions:

1. The member chooses the option check-out.
2. the system shows member‘s address and contact information.
3. The member can edit the address and contact information for this purchase.
4. The system will send to the external delivery system the information about the delivery.

4.1 if the external delivery system throws an error, the system will inform the member about the error and will disapprove the checkout.

1. The system shows the final price (including the delivery and sales) and the purchase options.
2. The member selects the desired option and fills in the required details.
3. The system validates that the member can afford the cart via external payment system.

7.1 if the external payment system throws an error, the system will inform the member about the error and will disapprove the checkout.

1. the system confirms the deal.
2. the system will access communication component and will send a receipt using email notification.

6) Use case: **member-A assigning member-B as an additional owner**.

-Actors: member-A.

-Precondition: member-A is an owner of the store, member-B is not an owner of the store also a registered user in the system.

-Input Parameters: member-B identifier, Store identifier.

-Actions:

1. member-A types member-B and store identifiers to the system.
2. The userRepo promotes the memberto be an owner with the appropriate permissions.

7) Use case: **owner changes manager managing permissions.**

-Actors: owner.

-Precondition: the owner is logged in.

-Input Parameters: manager identifier, permissions.

-Actions:

1. The owner selects "employees" option.
2. The system presents the store's employees list.
3. The owner selects desired manager.
4. The owner selects "change permissions" option.
5. The owner selects desired permissions.
6. The userRepo updates the manager's permissions.

8) Use case: **seller assigning a new sale on a product / list of products.**

-Actors: seller.

-Precondition: the seller is really the seller of the product / list of products.

-Input Parameters: sale type , list of products identifiers , sale percent , sale start date, sale end date , .

-Actions:

1. The seller selects "sales" option.
2. The system presents the sales list.
3. The seller selects "add new sale".
4. The seller selects type of sale.
   1. If it is a public sale , the seller selects the product / products, start date , end date and sale percent.
   2. If it is a conditional sale, the seller selects the product / products, start date , end date ,sale percent and conditions for the sale.
   3. If it is a hidden sale , the seller selects the product / products, start date , end date ,sale percent and sale coupon.
5. The system applies the sale according to it's type.

**תרחישי קבלה**

1) **Assigning a user as a manager of a shop**.

| Action | Data | Expected Result |
| --- | --- | --- |
| The shop owner assigns a user by typing username and shop identifiers to the system | Legal and Existing user, and shop identifiers. and the user is registered and not a manager in this store. | Success: assign request is approved, and received an indication message that the user is a manager now. |
| Existing user that registered to the system and not a manager. | Failure: Assign request has failed because every added user must have a specific shop (in that case there is no date for the shop). |
| Not existing or registered user with adding a shop identifiers. | Failure: Assign request failed because user must be registered to the system to get an assign approved as well. |
| Existing user that registered to the system but is already a manager with the given shop identifiers. | Failure: Assign request failed because user must be not a manager to the given shop, he must be not a manger of it to get an assign approved. |

2) **A user signs in.**

| **Action** | **Data** | **Expected Result** |
| --- | --- | --- |
| **A user signs in to the system.** | **the user is registered and not already signed in.** | **Success: assign request is approved, and received an indication message that the user is a signed in.** |
| **the user tried to sign up with correct username but wrong password.** | **Failure: Assign request has failed because the password is incorrect and the user will get an error message.** |
| **typing the username not correctly with the correct password.** | **Failure: Assign request failed because the username is incorrect and the user will get an error message.** |
| **Not Existing user or registered to the system who trying to sign in.** | **Failure: Assign request failed because user must be registered to the system so he can sign in and he will get an error message.** |

3) **A visitor adds a product to the shopping cart**.

| **Action** | **Data** | **Expected Result** |
| --- | --- | --- |
| **A visitor adds a product to the shopping cart.** | The visitor adds the product to the cart within typing the wanted quantity that's existed in the stock. | Success: adding to the cart is done, and the cart total cost and the number of products is updated successfully in the cart. |
| The visitor tried to add the product to the cart within typing the wanted quantity that's not existed in the stock. | Failure: adding to the cart has failed, and the system will send an alert to the visitor. |
| The visitor adds the product to the cart within typing the wanted quantity that's existed in the stock but the server not responding(system error) | Failure: adding to the cart has failed, and the system will send an alert to the visitor. |

4) **A visitor checks out**.

| Action | Data | Expected Result |
| --- | --- | --- |
| A visitor checks out from the system. | The visitor chooses to check-out option then types the correct address and contact information and after that, he pays for the cart. | Success: the checking out is done successfully and the system confirms the deal and also sends a receipt. |
| The visitor tried to type an incorrect address and contact information. | Failure: checking out has failed, and the system will throw an error. |
| The visitor got to the pay page but the system didn’t confirm the payment because the balance is smaller than the total price. | Failure: checking out has failed, and the system will throw an error. |
|  |  | |

5) **A user checks out**.

| **Action** | **Data** | **Expected Result** |
| --- | --- | --- |
| **A user checks out.** | The user chooses the checkout option with an empty shopping cart. | Failure: checking out has failed, the shopping cart must contain at least one product |
| The user chooses the checkout option (then he can change the address and contact information if he wants) and after that, he pays for the cart. | Success: the checking out is done successfully and the system confirms the deal and also sends a receipt. |
| The user got to the pay page but the system didn’t confirm the payment because the balance is smaller than the total price | Failure: checking out has failed, and the system will throw an error. |

6) **User-A assigning user-B as an additional owner**.

| Action | Data | Expected Result |
| --- | --- | --- |
| User-A assigning user-B as an additional owner of a specific shop. | User-A is not an owner of the shop.  user-B is registered and not owner of this shop. | Failure: the assigning has failed, the user-A must be an owner of the shop. |
| User-A is the owner of the shop.  user-B is registered and not the owner of the shop. | Success: the assigning is done successfully and the system updates that. |
| User-A is the owner of the shop.  user-B is not registered and is not the owner of the shop. | Failure: the assigning has failed, user-B must be registered first. |
| User-A is not an owner of the shop.  user-B is registered and is already an owner of the shop. | Failure: the assigning has failed, user-B must not be the owner of the same shop. |

7) **owner changes manager managing permissions.**

| Action | Data | Expected Result |
| --- | --- | --- |
| An owner changes manager managing permissions. | The owner logged in, typed the correct manager identifier, and then changed his permissions. | Success: the action is done successfully and the system updates that. |
| The owner chooses improper permissions for the manager. | Failure: the action has failed, the owner must select proper permissions for a manager. |

8) **seller assigning a new sale on a product/list of products.**

| **Action** | **Data** | **Expected Result** |
| --- | --- | --- |
| **seller assigning a new sale on a product/list of products.** | **The seller is really the seller of this product and typed legal data, especially on sale percent and start and end dates.** | **Success: the sale is approved successfully and the system updates that.** |
| **The seller typed an illegal start date or end date.** | **Failure: the assigning has failed, and the seller must choose legal dates.** |
| **The seller typed illogical percent on sale percent.** | **Failure: the assigning has failed, the seller must type logical percent on sale percent.** |