Sadna Express

סדנא ליישום פרויקט תוכנה

מגישים:

שי קרסנר - 209120278

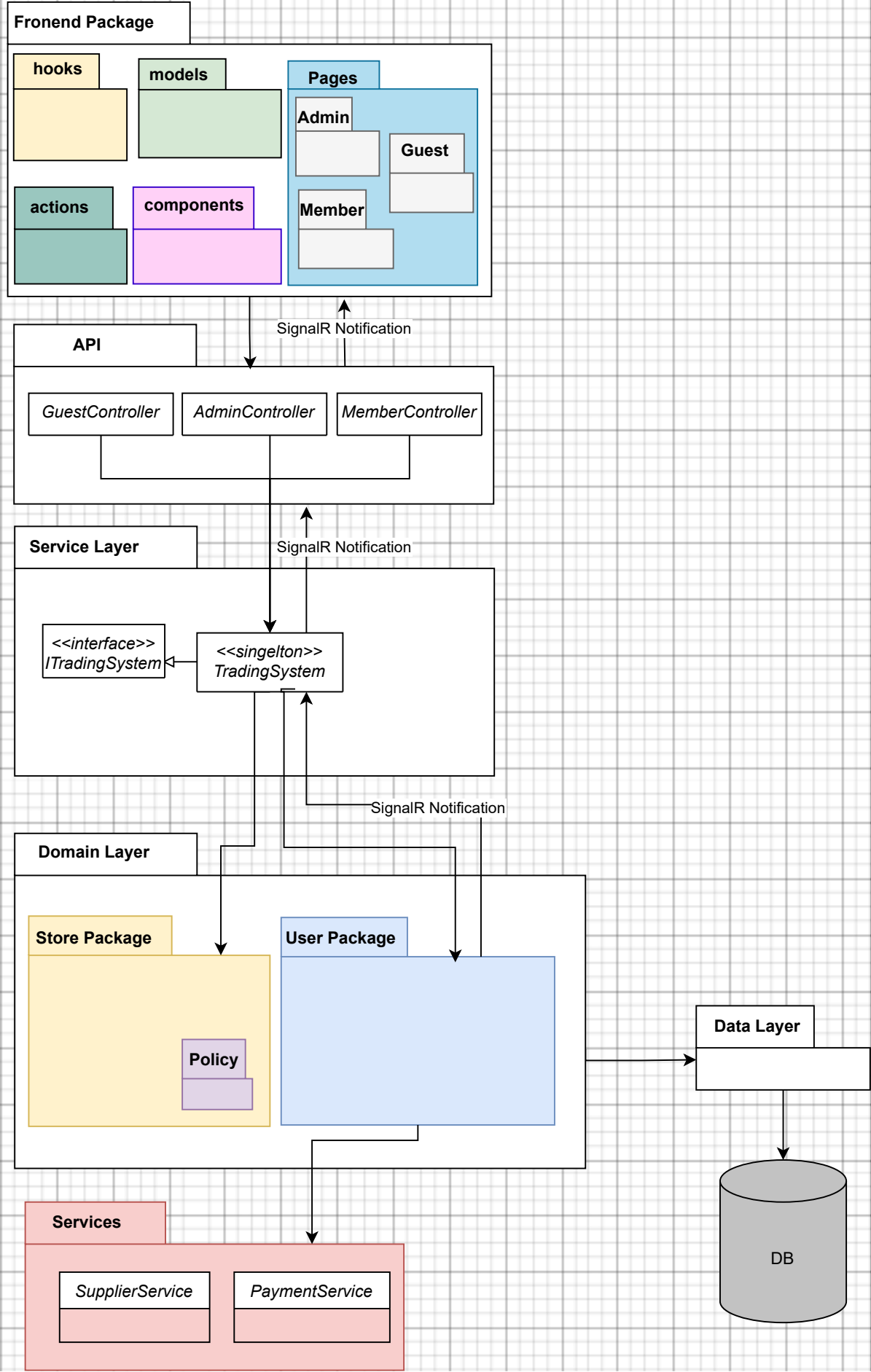
נוגה שוורץ - 207687849

דינה אגפוב - 207181629

טל גלמור - 318416575

גיל חאיס – 207919374

רדואן גאנם - 322509951


Fronend Package
hooks
models
Pages
Admin
Guest
Member
actions
components
SignalR Notification
API
GuestController
AdminController
MemberController
Service Layer
SignalR Notification
<<interface>>
ITradingSystem
<<singelton>>
TradingSystem
SignalR Notification
Domain Layer
Store Package
Policy
User Package
Data Layer
DB
Services
SupplierService
PaymentService

Fronend Package

App

actions
- AdminActions
- GuestActions
- MemberActions

components
- AdminNav
- DiscountCondition
- Condition
- StoreItem
- MemberNav
- GuestNav
- StoreInfo
- Navigation
- CartItem
- Footer
- Popup
- DiscountPolicy

hooks
- UseLocalStorage

SignalR
- UseSignalRNotifications

context
- SignalR - reducer
- SignalRProvider

models
- Bid
- Purchase
- Permission
- Review
- Notification
- Respone
- User
- Shop

Images

Pages

Admin
- AdminInitializeSystemPage
- AdminManageAllStoresPage
- AdminManageComplaintsPage
- AdminRevenuePage
- AdminViewAllPurchasesPage
- AdminViewAllUsersPage
- AdminUserActivityPage

Member
- ManageItemsPage
- DiscountPoliciesPage
- MessagesPage
- ManageStoreEmployeesPage
- PurchasePoliciesPage
- ManageStoreBidsPage
- PurchasedStoreItemPage
- PurchasedItemsPage
- StoreManagmentPage
- StoreRevenuePage

Guest
- RegisterPage
- LoginPage

- CartPage
- ShoppingPage
- Home
- SystemNotInit
- PaymentPage
- Exit

<<use>>

API Package

Controllers
- APIConstants
- GuestController
- MemberController
- AdminController

use

ClientRequests
- ClientRequest
- each request is a client request
- LoginRequest
- RegisterRequest
- OpenStoreRequest
- ConditionRequest
- StoreRequest
- ItemCartRequest
- SearchItemRequest
- PurchaseRequest
- WriteItemReviewRequest
- ItemIdRequest
- ItemEditRequest
- ItemRequest
- StoreIDRequest
- PolicyRequest
- StoreManagerRequest
- ItemEditRequest
- ComplexConditionRequest
- StoreManagerPerRequest
- DiscountPolicyRequest
- NotificationRequest
- ReactToBidRequest
- StoreRevenueRequest
- DiscountPolicyRequest
- SystemDateRequest
- DiscountPolicyRemoveRequest
- SystemDataActivityRequest
- ConditionIDRequest
- ReactToJobOfferRequest
- BidRequest
- BidsInStoreRequest

SignalR
- SignalRSerciveHost
- SignalRServerConfig
- NotificationHub
- <<singelton>> NotificationNotifier
- SystemActivityHub
- <<singelton>> SystemActivityNotifier

- ServerServiceHost
- ServerConfig

- Program
- State
- ApplicationOptions

Service Package

- Response
  - +ErrorMessage: string
  - +ErrorOccured: bool
- ResponseT
  - +Value: T

- <<Interface>> ITradingSystem
- <<singelton>> TradingSystem
- ProxyBridge
- 1
- 1

- StoreManager
- <<interface>> IStoreManager
- <<interface>> IUserManager
- UserManager

<<use>>
<<use>>

Service object, The object are from primitive object

SModels
- SBid
- SMember
- SPermission
- SReview
- SPolicy
- SItem
- SPaymentDetails
- SSupplyDetails

<<uses>>
<<uses>>

Domain Package

Store Package

- <<Interface>> IStoreFacade
- <<interface>> IOrders
- <<singelton>> Orders
  - -Dict<userID, order>
  - -Dict<storeID, order>
- Data object

- StoreFacade
- <<persistent>> Store
- <<persistent>> Review
- <<persistent>> Order
- Each store has at one founder - member who opened the store becomes founder
- const object
- <<persistent>> ItemForOrder
  - -storeID
- <<persistent>> Inventory
  - -ItemQuantity: Map<Item,int>
- <<persistent>> Item

Policy

- foreach discountPolicy in DiscountPolicyTree discountPolicy.caculate(...)
- foreach Condition in ComplexCondition Condition .evaluate(...)
- evaluate return bool. True if the item include the condition, false otherwise
- The policy can contain only valueCondition, QuantityCondition
- DiscountPolicyTree
  - caculate(dict <item,quentitity>)
- we get dictinary of all the items that there price updated
- ComplexCondition
  - evaluate(dict <item,quentitity>)
- <<abstract>> Condition
- leaf
- <<abstract>> DiscountPolicy
  - caculate(dict <item,quentitity>)
- SimpleDiscount
- MaxDiscount
- AddDiscount
- ConditionalDiscount
- <<abstract>> LogicalPolicy
- AndCondition
- OrCondition
- ConditionalCondition
- TimeCondition
- QuantityCondition
- ValueCondition
- XorDiscount
- AndDiscount
- OrDiscount
- DB obj for policy & condition
- <<persistent>> PolicyDB
- <<persistent>> ConditionDB
- The condion have a field message, that have the reason the evalute fail.

Composition!

User Package

- <<persistent>> Shopping Basket
  - -storeID
  - -list<Item_ID>
- <<persistent>> Shopping Cart
- Member Password is encrypted
- <<Interface>> IUserFacade
- <<persistent>> User
- UserFacade
  - -Dict<userId, string> macs
- <<interface>> IObserver
  - update(Notification):void
- <<persistent>> Member
  - -Dict<storeID, Dict<Bid>> PendingPermission
- <<singelton>> UserUsageData
- <<persistent>> Notification
- <<persistent>> PromotedMember
  - -Dict<storeID, DirectManager>
  - -Dict<storeID, List<appoint>>
  - -Dict<storeID, List<string>> permission
  - -Dict<storeID, List<Bid>> BidOffer
- <<singelton>> NotificationSystem
  - -Dict<storeID, List<Member>>
- <<persistent>> Visit
- <<singelton>> Permission
- the allowed actions for each type of member
- the permission includes founder(one to each store) and owners
- <<interface>> ISubject
  - RemoveObserver(Guid,Member): void
  - RegisterObserver(Guid,Member): void
  - NotifyObserver(Guid,string,Guid): void
- <<persistent>> Bid
  - -itemID
- <<persistent>> InitializeSystem
  - + Id : Guid
  - + IsInit: bool
- The Bid hold the list of workers need to approve the Bid. If someone deny the bid- the bid closed and removed
- <<interface>> IRegistation
  - ValidateStrongPass (String):bool
  - ValidateEmail(String) :bool
- Registation
- <<interface>> IPasswordHash
  - Hash(String):String
  - Rehash(String, String):bool
  - Validate(String):bool
- PasswordHash
- <<persistent>> Macs
  - + Id : Guid
  - + mac: string

use

Data Layer

- DataBaseContextFactory
  - + TestMode : bool
- DBHandler
- DataBaseContext
- DataBaseTestsContext

- Logger
- we can add log from every layer

Services

- iSupplierService
- IPaymentService
- SupplierService
- PaymentService

מילון מונחים - Glossary:

מילון מושגים עבור תרחישי השימוש במערכת.

1. Trading system - מערכת המסחר, האתר המרכזי שמציג את כלל החנויות.
2. Login - כניסה למערכת עבור מנויים בעזרת פרטי זיהוי שנקבעו מראש.
3. User - משתמש במערכת - יכול להיות אורח/מנוי
4. Guest - מבקר באתר שאין לו חשבון.
5. Member - משתמש הרשום למערכת המסחר. משתמש מנוי היכול לשמש בתפקידים שונים בשוק - בעל-חנות, מנהל-חנות.
6. System manager - מנהל מערכת המסחר. אחראי על הניהול השותף של המערכת ובעל הרשאות אדמין באתר.
7. Store owner - בעל חנות והוא גם מנהל בחנות עם כל ההרשאות.
8. Store manager - מנהל בחנות עם הרשאות מוגבלות.
9. Store Founder - מייסד החנות , בעל החנות הראשון.
10. User notification center - מקום בו מאוחסנים כלל ההתראות עבור משתמש.
11. User shopping cart - עגלת הקניות של משתמש, המקום בו מאוחסנים כלל המוצרים שהמשתמש מעוניין לקנות, אוסף של סלי קניות מחנויות שונות.
12. User shopping basket - סל קניות של משתמש עבור חנות ספציפית במערכת המסחר.
13. Store policy– מכיל את המדיניות של החנות.
14. Discount policy – מפרט הנחה מסוימת שקיימת ברכישה.
15. Purchase policy – מפרט הגבלה מסוימת שקיימת ברכישה.
16. Inventory – המלאי של חנות ספציפית, כלומר מכיל את הזמינות של המוצרים של החנות.
17. Item – פריט של חנות ספציפית. מכיוון שלכל חנות יש תיאור ומחיר משלה לכל מוצר, אז המוצרים של חנויות שונות יהיו אובייקטים שונים זה מזה.
18. External services - שירותים חיצוניים שמערכת המסחר נעזרת בהן על מנת לנהל אותה - כמו שירותי אספקה ושירותי רכישה ותשלומים.
19. Limited server time - זמן מוגדר בשניות במערכת עבור זמן המתנה מקסימלי לתגובה משירות חיצוני כלשהוא.
20. Password protocol - פרוטוקול במערכת המסחר המגדיר מה היא סיסמא חוקית.
21. Promoted Member - משתמש בעל הרשאות מיוחדות.
22. Order - הזמנה במערכת המסחר שנעשתה ע"י אורח/מנוי.
23. NotificationSystem - מערכת ניהול שליחת ההודעות של מערכת המסחר.

24. Bid - הצעת קנייה, משתמשים יכולים להתמקח עם בעלי החנות ולבקש מחיר זול יותר ממה שמוצע.
25. System Revenu - חתך יומי של כמות הרווחים של מערכת המסחר מרכישות של לקוחות/אורחים.
26. System Activity - מידע על מספר המבקרים היומי במערכת בטווח תאריכים מבוקש, המבקרים מחולקים על פי: אורחים, משתמשים שאינם מנהלי או בעלי חנות, מנהלי חנות שאינם בעלים של אף חנות, בעלי חנות ומנהלי מערכת. נתונים אלו מתעדכנים בזמן אמת במערכת.
27. Employement agreement - הסכם מינוי של בעל חנות חדש, בעת מינוי חדש, כל הבעלי חנות צריכים לאשר את העובד החדש.

הטלת אחריות לקיום אילוצי נכונות

מס' אילוץ	אילוץ נכונות	הטלת אחריות לקיום האילוץ
1	למנוי יש שם יחיד המזהה אותו במערכת.	תרחיש שימוש - Register To the system
2	יש למערכת לפחות מנהל אחד. מנהל- מערכת חייב להיות מנוי (עבר תהליך רישום).	תרחיש שימוש – Initialization of the trading system
3	בעל-חנות או מנהל-חנות חייב להיות מנוי.	תרשים מחלקות - לפי יחס הירושה (Promoted_Member יורש מmember ובעל חנות או מנהל חנות הוא כזה)
4	פעולות בשוק מבוצעות רק ע"י משתמשים המבקרים בשוק.	תרחישי שימוש – כלל פעולות השוק הרלוונטיות למנויים מותנות ברישום למערכת
5	לחנות פעילה (שאינה סגורה) חייב להיות לפחות מייסד-חנות אחד.	תרשים מחלקות - Promoted Member מחזיק מילון ID של חנות והרשאות (כל חנות פתוחה נמצאת באיחוד המילונים)

6 א	לחנות חייבים להיות מוגדרים תהליכי קנייה (רכישה) והנחה. יתכן ויש ברירות מחדל עבור אופן קנייה ועבור סוג הנחה.	תרשים מחלקות – אובייקט store חייב להחזיק policy (לא בגרסה הזו)
6 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".	תרשים מחלקות – אובייקט store חייב להחזיק policy (לא בגרסה הזו)
7	לקונה יש עגלת קניות יחידה, המורכבת מאוסף כל סלי הקנייה שלו. לקונה יש לכל היותר סל קניות יחיד עבור חנות כלשהי.	תרשים מחלקות - לפי קרדינליות הקשר (קשר יחיד ליחיד) שדות storeID ב,ShoppingBasket Item
8	עגלת הקניות של קונה (אורח או מנוי) הינה בבעלותו הבלעדית ואינה ניתנת לשינוי על ידי שום משתמש אחר.	תרשים מחלקות – shopping cart מקושרת לuser אחד בלבד
9	ניתן לקנות פריטים של מוצר בחנות לכל היותר בכמות הקיימת במלאי.	תרשים מחלקות – inventory מחזיקה את הitems לפי כמותם.
10 א	ניתן לגבות כסף מקונים רק עבור עסקאות שבוצעו ולגבות רק את הסכומים שהוצהרו.	תרחיש שימוש מערכת : payment
10 ב	תהליך קנייה מסתיים בהצלחה רק אם שולם הסכום הדרוש ורק אם האספקה אושרה	תרחיש שימוש מערכת : payment, supply
10 ג	מוכר יכול לקבל תשלום רק כתוצאה מתהליך קנייה מוצלח	לאחר שתהליך קניה יסתיים בהצלחה (10 ב) המערכת תבצע העברת תשלום למוכר
11 א	נדרש לפחות קשר אחד לשירות גביית כספים	תרשים מחלקות - קיים אובייקט payment service עם קרדינליות של לפחות אחד
11 ב	נדרש לפחות קשר אחד לשירות אספקה	תרשים מחלקות - קיים אובייקט supplier service עם קרדינליות של לפחות אחד

תרחישי שימוש – Use Cases :

System use cases:

Use case: Initialization of the trading system. (1.1)

- Actor: user
- Precondition: user is logged in.
- Postcondition: trading system is initialized.
- Parameter: none
- Actions:
    1. User request to open (initialize) the trading system.
    2. System checks that the user has permissions to perform this operation.
    3. System if user has permissions – checks whether there is a connection to payment and supply services and there is a system manager.
    4. System if action succeeded – opens the trading system for clients.

Participants	**Parameters**	**Expected Result**	**Scenario**
Member	User is system Manager	Trading system is initialized	Good
Member	User isn't System Manager	Trading system staying closed	Bad

External services	There is a connection to payment and supply services	Trading system is initialized	Good
External services	There isn't a connection to payment and supply services	Trading system staying closed	Bad

Use case: Update connection with external services. (1.2)

- Actor: System Manager
- Precondition: user is logged in
- Postcondition: external services are connected to the trading system.
- Parameter: old external service, new external service
- Actions:
    1. System checks that the user has permissions to perform this operation.
    2. System if user has permissions – checks whether is it possible to connect to the new external service.
    3. System if action succeeded – checks that old and new services have the same API.
    4. System if action succeeded – waits until there is no use in old service.
    5. System replaces the old service with the new one.
    6. System sends an indication to the user about the exchange.

Participants	**Parameters**	**Expected Result**	**Scenario**
System Manager	old external service, new good external service	Service replaces successfully	Good
System Manager	old external service, new bad external service (unavailable service/service doesn't exist)	Service is not replaced	Bad

Use case: Payment. (1.3)

- Actor: System
- Precondition: user triggered a request to make a payment, User shopping cart is valid and not empty.
- Postcondition: Store received the payment from the system.
- Parameter:
  - Transaction details
  - User details
  - User shopping cart details
- Actions:
  1. System sends request to payment external service.
  2. System waits Limited server time for service response.

3. System if action succeeded –
    - Sends user shopping cart and user details to supply external service.
    - Sends positive indication for the user and emptying the user shopping cart.
    - Store received the payment from the system.
4. System if action failed–Send negative indication to the user.

Participants	**Parameters**	**Expected Result**	**Scenario**
System, external service	Good transaction details	Payment completed successfully	Good
System, external service	bad transaction details (for example wrong ccv)	Payment is not completed	Bad
System, external service	Good transaction details, external service is not responding	User receives error message from the system with appropriate message	Good

Use case: Supply. (1.4)

- Actor: System
- Precondition: Payment completed (for some shopping cart)
- Postcondition: Supply process started (for some shopping cart)
- Parameter:
  - User details
  - User shopping cart details
- Actions:
  1. System sends user shopping cart and user details to supply external service.
  2. System waits Limited server time for service response.
  3. System if action succeeded sends positive indication for the user that supply process has begun successfully.
  4. System if action failed– sends negative indication for the user.

Participants	**Parameters**	**Expected Result**	**Scenario**
System, external service	Good user shopping cart and user details	Receiving a supply confirmation from the service	Good
System, external service	Bad user shopping cart and user details (errors in user details - bad address ext.)	Receiving error from the supply service	Bad

System, external service	Good user shopping cart and user details, external service is not responding	User receives error message from the system with appropriate message	Good

Use case: User online notifications. (1.5)

- Actor: System
- Precondition:
  - Message has received or
  - purchase has occurred or
  - Store re-opened or closed or
  - permission in a store has been removed.
  - review was written for item in the store of the owner
- Postcondition: Message received by the user if logged in, or added to the user notification system if not logged in.
- Parameter:
  - User Message with notification info
  - User details
- Actions:
  1. System checks if the user with the user details is logged in.
  2. System if logged in - sends the message to the user.
  2. System if not logged in - adds the message to the user notification center which the user will receive when he logs into the system.

Participants	**Parameters**	**Expected Result**	**Scenario**
System	purchase has occurred to a specific store; store manager is connected to the trading system	The store manager receives a notification about the purchase	Good
System	purchase has occurred to a specific store; store manager is connected to the trading system	The store manager does not receive a notification about the purchase	Bad

Use case: User offline notifications. (1.6)

- Actor: System
- Precondition:
  - Message has received or
  - purchase has occurred or
  - Store re-opened or closed or
  - permission in a store has been removed.
  - review was written for item in the store of the owner
- Postcondition: Message received by the user if logged in, or added to the user notification system if not logged in.
- Parameter:
  - User Message with notification info
- Actions:

1. System checks if the user with the user details is logged in.
2. System if logged in - sends the message to the user.
3. System if not logged in - adds the message to the user notification center which the user will receive when logs into the system.

Participants	Parameters	Expected Result	Scenario
System	purchase has occurred to a specific store; store manager is not connected to the trading system.	The store manager logs in to the system and accepts a notification about the purchase	Good
System	purchase has occurred to a specific store, store manager is not connected to the trading system	The store manager logs in to the system and does not accept a notification about the purchase	Bad

Users use cases:

Guest user:

Use case: Enter the system. (1.1)

- Actor: User
- Precondition: User not logged in
- Postcondition: -

- Parameter: none
- Actions:

    1. User entering the system.
    2. System defines user as Guest-user.
    3. System assigns the user a shopping cart.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	The user entered to the system	Successfully entered the system and accepts a shopping cart	Good
User	The user entered to the system	The system did not upload correctly to the user, user got null id	Bad

Use case: Exit from the system. (1.2)

- Actor: User
- Precondition: User not logged in
- Postcondition: -
- Parameter: none
- Actions:

    1. User exits from the system.

2. System Delete User from current users list and all user guest information.

Participants	Parameters	Expected Result	Scenario
User	-	Successfully exit the system and his shopping cart deleted	Good
User	-	Successfully exit the system and did not delete from Guests's list	Bad

Use case: Register To the system. (1.3)

- Actor: User
- Precondition: User not logged in and not registered to the system before
- Postcondition: System registered the user to the system.
- Parameter: Identifying details
- Actions:

1. User enters the system.
2. User register to the system by giving Identifying details.
3. System validates user email and checks it's unique in the system.
4. System creates new member.
5. System assigns registered user a new empty shopping cart.

Participants	Parameters	Expected Result	Scenario
User	Unique and valid email, strong password, valid id	Successfully registered to the system	Good
User	username that already in the system, strong password , valid id	Username already used in the system , msg will be thrown	Bad

Use case: Login To the system. (1.4)

- Actor: User
- Precondition: User not logged (guest) in and registered to the system before
- Parameter: username and password
- Actions:

1. User enters the system.
2. User login to the system using the username and password.
3. System defines user as member.

4. System will update the shopping cart , according to member’s shopping history.

5. System will send all offline notification to the user.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	Unique username, strong password	Successfully logged in to the system	Good
User	Username and password that don't match the username and password from the system’s db.	Failed to log in to the system	Bad

Use case: Getting information about stores in the market and the products in the stores. (2.1)

- Actor: User
- Precondition: User not logged in
- Postcondition: User received the info.
- Parameter: none
- Actions:

1. User asked from the system to accept information about specific store or all stores and items.
2. System if store doesn't exist, raises appropriate message.
3. System displays information for the chosen store

Participants	**Parameters**	**Expected Result**	**Scenario**
User	-	User see's the updated and valid store info	Good
User	-	User see's old/non updated store info	Bad

Use case: Guest search products by general search or filters. (2.2)

- Actor: User
- Precondition: User not logged in
- Postcondition: User received the list of items according to the search.
- Parameter: Info of product
- Actions:

    1. User asks from the system to search according to the info of the product he has.
    2. System if store doesn't exist, raises appropriate message.
    3. All stores search for items that match the search terms.
    4. System presents the results from all the stores, if there is no results- system present appropriate message.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	Search on known product in the searching tool of the system	System shows list of products according to the info provided	Good
User	Search work that doesn't has related items	System shows a message that no match items has found	Good
User	Search on known product in the searching tool of the system	System shows list of products that some of	Bad

		them are from inactive stores	
User	Enter product that didn't exists	System shows list of products that unrelated to the info provided	Bad

Use case: Guest saving item in the shopping cart for some store. (2.3)

- Actor: User
- Precondition: User not logged in
- Postcondition: System saved the item in shopping cart.
- Parameter: -
- Actions:

1. User request to put item in the shopping cart.
2. Store checks that item is in stock, if no sends appropriate error
3. Shopping cart creates a shopping basket if necessary for the item store.
4. Shopping basket adds the item to the basket
5. System updates the shopping cart to be visible for the client.

Participants	Parameters	Expected Result	Scenario
User	User adds item to the shopping cart	Shopping cart is updated with the new item in it	Good
User	User adds item to the shopping cart	Shopping cart is not updated with the new item.	Bad

Use case: Guest checking the content of the shopping cart and making changes. (2.4)

- Actor: User
- Precondition: User not logged in.
- Precondition: User can see his shopping cart items and changes has been done to the shopping cart.
- Parameter: -
- Actions:

1. User request to see the items in the shopping cart.
2. System Displays the shopping cart items.
3. User request to make changes in the shopping cart (change item count/delete item from shopping cart).
4. Store of the item checks that the item quantity fits to the user request, if no present appropriate mesage.
5. Shopping cart preforming the user requests and adding/removing/editing the item by the item shopping basket.
6. User shopping cart is updated.
7. System presents the updated shopping cart to the user.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	User requests to see items in his shopping cart	when checking the content of the cart, the item appears	Good
User	User requests to see items in his shopping cart	when checking the content of the cart, the item doesn't appear	Bad

User	User requests to delete item from his shopping cart	when checking the content of the cart, the item doesn't appear	Good
User	User requests to delete item from his shopping cart	when checking the content of the cart, the item appears	Bad

Use case: Guest making a purchase of the shopping cart. (2.5)

- Actor: User
- Precondition: User not logged in, User's Shopping cart is not empty, payment method is provided.
- Postcondition: System validate the order and remove the purchased items from the store they brought from.
- Parameter: Shopping list
- Actions:

1. User puts items in his shopping cart.
2. User request to buy his shopping cart.
3. System checks that shopping cart is not empty
3. Store - Each store that has items from the user shopping cart checks that discount policy and purchase policy are applied and that the items are in stock.
4. Shopping cart calculate the payment for the user and the payment to be transfer to each store
4. System request from the user transaction details and delivery details.
5. User inserts payment method.
6. System starting the scenario of Payment use case and then moving to Supply use case.

Participants	Parameters	Expected Result	Scenario
User	Valid payment information	Successful purchase: Payment is accepted, and supply process began to the user	Good
User	invalid payment information	Payment is not received, and the purchase is not made	Bad
User	One of the items in the shopping cart is not in stock	Purchase failed, and appropriate message presents to user	Good
User	One of the items in the shopping cart is not in stock	Successful purchase	Bad
User	User shopping cart has 5 dresses from "renunar" and renuar limiting 4 dresses per a purchase	Purchase is not approved by one of the stores purchace policy that involves items in the shopping cart	Good
User	User shopping cart has 5 dresses from "renunar" and renuar limiting 4 dresses per a purchase	Successful purchase: Payment is accepted and supply process began to the user	Bad

Member:

Use case: Exit from the system. (3.0)

- Actor: Member
- Precondition: User logged in
- Postcondition: -
- Parameter: none
- Actions:
    1. User exits from the system.
    2. System logs out the user, and then exit from system (3.1, 1.2).

Participants	**Parameters**	**Expected Result**	**Scenario**
User	-	Successfully exit the system and his shopping cart saved	Good
User	-	Successfully exit the system and his shopping cart deleted	Bad

Use case: Logout. (3.1)

- Actor: Member
- Precondition: User logged in
- Postcondition: User not logged in.
- Parameter: none
- Actions:

1. User request to logs out from the system.
2. User change his login status to false
2. System define user as guest.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	-	Successfully logged out	Good
User	-	User logged out but the system recognizes the user as log in	Bad

Use case: Opening a store. (3.2)

- Actor: Member
- Precondition: User logged in.
- Postcondition: New store is opened.
- Parameter: Store's detailed information
- Actions:

1. User request to create store in the system.
2. System checks that store name is unique, is no present appropriate message to user.
3. System marks the user as the 'Store Founder' of the store.
4. System set the store to 'open.'

Participants	**Parameters**	**Expected Result**	**Scenario**
User	Good and unique store name	Store created Successfully	Good
User	Invalid store name (exist in the system)	Store is not created	Bad

Use case: Writing a review on items the user purchased. (3.3)

- Actor: Member
- Precondition: User logged in and he has purchased products before
- Postcondition: Written review appears in the item's comment section
- Parameter: The product the user wants to write a review about
- Actions:

1. User request to inserts his review about a specific item.
2. System checks that item exist, if not present appropriate message.
3. System checks that the review is not empty
4. Item adds the review to its comments section.
5. System sends a notification to the store owner that a review was written about an item in his store

Participants	**Parameters**	**Expected Result**	**Scenario**
User	Well written review, user purchased in the past the specific item	The users' review is added to the comments on the product	Good
User	Well written review, user purchased in the past the specific item	The users' review is not added to the comments on the product	Bad
User	Well written review, user did not purchased in the past the specific item	User cannot add a review to this item- appropriate message is presented	Good

Use case: Rating item and store by user. (3.4)

- Actor: Member
- Precondition: User logged in, user has purchased the item before / user has purchased from the store before.
- Postcondition: Rating appears in the item/store's ratings section.
- Parameter: The item/ store the user wants to rank
- Actions:
  1. User search for the item/ store he wants to rate. (for more info 2.1, 2.2)
  2. User inserts his ranking.
  3. Store/ Item adds the new ranking to their ratings section.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	A rating of the user about the store/item	Ranking has been added to the store/item	Good
User	A rating of the user about the store/item	Ranking has not been added to the store/item	Bad

Use case: User sends message to store. (3.5)

- Actor: User
- Precondition: User logged in, store exists.
- Postcondition: Written message appears in store's messages section.
- Parameter: store details, message
- Actions:

1. User search for the store he wants to send message to (Getting information about stores in the market and the products in the stores use case)
2. User writing the message and send it to the store.
3. Store adds the message to store's user message list

Participants	**Parameters**	**Expected Result**	**Scenario**
User	Well written message	The user's message is found in the store's messages	Good
User	Well written message	The user's message is not found in the store's messages	Bad

Use case: Filing a complaint by a user about a purchase. (3.6)

- Actor: User, System Managers
- Precondition: User logged in, user has purchased history
- Postcondition: complaint appears in the store's complaints section.
- Parameter: The purchase, complaint
- Actions:

1. User search for the purchase in the 'past purchases' section
2. User chooses the option of 'complaint' in the purchase.
3. User describes the reason for complaining about the purchase.
4. User sends the complaint.
5. System adds the complaint to the system manager complaints section.
6. System Managers receiving the complaint.

Participants	**Parameters**	**Expected Result**	**Scenario**
User, System Managers	Well written complaint	The user's message received in the store's messages	Good

User, System Managers	A complaint that is inappropriate, containing rude words	The user's message didn't receive in the store's messages	Bad

Use case: Receiving information about personal purchase history. (3.7)

- Actor: Member
- Precondition: User logged in
- Postcondition: User receives info about his personal purchase history.
- Parameter: -
- Actions:
    1. User enters his personal user profile.
    2. User chooses the 'purchases' history' section.
    3. System displays purchases' history' for the user

Participants	**Parameters**	**Expected Result**	**Scenario**
User	User purchased some product in different purchases	The user's purchases' history printed and matching the user's purchases	Good
User	User purchased some product in different purchases	The system does not show all user purchases	Bad

Use case: Editing identifying details of user. (3.8)

- Actor: User
- Precondition: User logged in
- Postcondition: User's details updated in the system.
- Parameter: The info that the user wants to update
- Actions:

1. User enters his personal user profile.
2. User update the info he wants.
3. System cheeks that new details are approved by the password protocol and that the details are real and verified.
4. System if validation succeed updates the user that the info was updated, if no- system present appropriate message to the user.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	new email address	Email has updated	Good
User	new email address that doesn't exist	Email is not updated and appropriate message present to user	Bad

User	new password that doesn't comply with the password protocol	Password is not updated and appropriate message present to user	Bad

Use case: Registration security for the trading system. (3.9)

- Actor: Member
- Precondition: User logged in.
- Postcondition: User is requested to answer questions when security checks are needed.
- Parameter: Questions and Answers
- Actions:

1. User enters his personal user profile.
2. User enters 'registration security' section.
3. User enters Questions and Answers to strengthen the security of his subscription.
4. User adds Questions and Answers to his security section.

Participants	**Parameters**	**Expected Result**	**Scenario**
User	Good Q & A - only user knows	System updated the new Q & A	Good

User	Bad Q & A	System don't update the new Q & A	Bad

Store owner:

Use case: product management. (4.1)

- Actor: store owner
- Precondition: user is logged in, store exist, user is the owner of the store
- Postcondition: new item appears in the inventory of the store.
- Parameter: product information (name, price, etc.)
- Actions:
    1. User request to Add/Remove/Edit item in the store inventory.
    2. Inventory checks that operation is possible:
        a. If new Item is added - check that his name is unique
        b. If user requested to edit/remove item- check that item exist
    3. If operation is not possible Inventory sends appropriate message
    4. Inventory updates the new item/new item quantity and sends positive indication to user..
    5. Item updates his new details and sends positive indication to user.

Participants	Parameters	Expected Result	Scenario
Store owner	Valid product data	The item is added to store	good
Store owner	Invalid product price	The item was not added to the store, and an error is displayed	bad
Store owner	Edit product that does not exist price	Operation failed - Appropriate message sends to the user	bad

Use case: changing store policy. (4.2)

- Actor: store owner
- Precondition: user is logged in, store exist, user is the owner of the store
- Postcondition: discount policy of the store updated.
- Parameter: discount policy (items affected, discount amount…)
- Actions:
    1. User chooses to edit the store policy.
    2. User chooses to change an existing discount policy.
    3. User enters the new discount policy details.
    4. Store checks that new discount policy is consistent with traceability constraints of the store founder, if yes - updated the discount policy, if no- present an appropriate message.

Participants	Parameters	Expected Result	Scenario
Store owner	Valid discount policy	The discount policy of the store is changed	good
Store owner	Valid discount policy	The discount policy was not changed, and an error is displayed	bad
Store owner	Valid discount policy that does not fit to founder constraints	The discount policy was not changed, and an appropriate message is displayed	good

Use case: appointing a new store owner - single owner. (4.4)

- Actors: store owner, member
- Precondition: store owner is logged in, member is registered, store exists, store owner is the owner of the store, member is not the owner of the store
- Postcondition: member is storing owner in the store.
- Parameter: member to appoint details.
- Actions:
    1. Store owner requests to add new store owner.
    2. System asks for new store owner details.

3. System checks that new store owner details is a member in the system. If not system present appropriate message to user.

4. Permissions checks that the new store owner is not already a store owner in the store. If he is, store sends to system an appropriate message that the system will display to the user.

5. Permissions adds member as the owner of the store and gives the new store owner permissions of management and store policy and adds the store owner as the appointer of the new store owner.

6. System sends a positive indication to the user.

Participants	**Parameters**	**Expected Result**	**Scenario**
store owner, member	New store owner is a member and is not yet the store's owner	member is added as the owner of the store.	good
store owner, member	Store owner enter details of un-registered user	The operation fails and an error is displayed	bad

Use case: removing appointment of a store owner. (4.5)

- Actors: store owner, appointed store owner
- Precondition: store owner is logged in, store exists, store owner is the owner of the store, appointed store owner is also the owner of the store
- Postcondition: appointed store owner and all his appointies are no longer owners in the store. And all the appointies revice a notification that their permission to he store has been removed
- Parameter: appointed store owner details.

- Actions:

    1. Store owner chooses to remove the permissions of the appointee store owner.
    2. The system asks for identifying details of store owner.
    3. Store owner enters the identifying details of store owner.
    4. System checks that store manager details are valid (member and store owner) and that store owner is not the appointer of the store - if not present an appropriate message.
    5. System removes the appointments of the appointed store owner and of all his appointies.
    6. System sends a notification to all the appointies that their permission to he store has been removed

Participants	**Parameters**	**Expected Result**	**Scenario**
store owner, appointed store manager	The store owner of the specific store and the store owner is the appointer of the store owner	store manager permissions are updated.	good
store owner, appointed store manager	store owner is not a manager of this store / store owner is not the appointer of the store owner	The operation fails and an error is displayed	bad

Use case: appointing a new store manager. (4.6)

- Actors: store owner, member
- Precondition: store owner is logged in, member is registered, store exist, store owner is the owner of the store, member is not the store manager.
- Postcondition: member is store manager in the store.
- Parameter: member to appoint details.
- Actions:

    1. Store owner requests to add new store manager.
    2. System asks for new store manager details.
    3. System checks that new store manager details is a member in the system. If not system present appropriate message to user.
    3. Permissions checks that the new store manager is not already a store owner/store manager in the store. If he is, store sends to system an appropriate message that the system will display to the user.
    4. Permissions adds member as the manager of the store and gives the new store manager permissions to receive information (4.12, 4.13) and adds the store owner as the appointer of the new store manager.
    5. System sends a positive indication to the user.

Participants	**Parameters**	**Expected Result**	**Scenario**
store owner, member	New store manager is a member and is not yet the store's owner/manager	member is added as the manager of the store.	good

store owner, member	Store owners enter details of un registered user/ member is already owner/manager in the store	The operation fails and an error is displayed	bad

Use case: changing a store manager permission (4.7)

- Actors: store owner, store manager.
- Precondition: store owner is logged in, store manager is registered and is a manager in the store, store exist, store owner is the owner of the store, and store owner is the appointer of store manager.
- Postcondition: store manager permission updated.
- Parameter: store manager's identifying details.
- Actions:
    1. Store owner chooses to edit the permissions of the store manager.
    2. The system asks for identifying details of store manager.
    3. Store owner enters the identifying details of store manager.
    4. System checks that store manager details are valid (member and store manager) and that store owner is not the appointer of the store - if not present an appropriate message.
    5. Store owner enters new permissions of the store manager. If permissions are not valid - present an appropriate message.

6. System updates the permission of the store manager.

Participants	Parameters	Expected Result	Scenario
store owner, store manager	The store manager of the specific store and the store owner is the appointer of the store manager	store manager permissions are updated.	good
store owner, store manager	store manager is not a manager of this store / store owner is not the appointer of the store manager	The operation fails and an error is displayed	bad

Use case: closing a store. (4.9)

- Actors: store founder.
- Precondition: user is logged in, store exist, user is the founder of the store
- Postcondition: store closed and a notification was sent to all the store owners.
- Parameter: store id
- Actions:

    1. Store owner request to close his store.
    2. Store sets itself as inactive.
    3. Store sends notifications to all the store managers and owners.
    4. System updated the store as an inactive store.

Participants	Parameters	Expected Result	Scenario
store owner	Store owner is the owner of the store	Store is now inactive, and all the managers and owners get notified, store items are not available on search.	good
store owner	Store owner is not the owner of the store	The operation fails and an error is displayed	bad
store owner	Store owner is the owner of the store	Store is now inactive, and all the managers and owners get notified, store items are still available on search.	bad

Use case: request store employees' information. (4.11)

- Actors: store owner.
- Precondition: user is logged in, store exist, user is the owner of the store

- Postcondition: store owner receives the info.
- Parameter: store id
- Actions:
    1. Store owner requests store employee information.
    2. Store displays all the managers and owners of the store as well as their permissions.

Participants	**Parameters**	**Expected Result**	**Scenario**
store owner	Store id is valid and store owner is the owner of the store	System displays the details	good
store owner	Store id is invalid/ store owner is not the owner of the store	The operation fails and an error is displayed	bad

Use case: Get info and read users complaints and respond. (4.12)

- Actor: Store owner
- Precondition: User logged in, user is store manager with permissions or owner
- Postcondition: Complaint received on the store manager notifications, his response received by the user / user got the wanted info.
- Parameter: Complaint response
- Actions:

    1. Store owner request to get info about the store and to see customers complaints.
    2. Store check that the user is a Store owner/store manager with permissions. If not present appropriate message.
    3. Store present customers complaints to the store owner.
    4. Store owner added a comment to customer complaint.
    5. System sends a notification to the user about the store owner comment (1.5, 1.6)

Participants	**Parameters**	**Expected Result**	**Scenario**
store, store owner	complaint response	response added and notification sent to user successfully	Good
store, store owner	complaint response	complaint response didn't get sent, the user got an error.	Bad

Use case: request store purchase history. (4.13)

- Actors: store owner.
- Precondition: user is logged in, store exists, user is the owner of the store
- Postcondition: store owner receives the info.
- Parameter: store id
- Actions:
    1. Store owner requests the store purchase information.
    2. Store checks that the user is store owner/ manager with permissions to see it's purchase history. If not present appropriate message.
    2. Store displays it's purchase history.

Participants	**Parameters**	**Expected Result**	**Scenario**
store owner	Store id is valid	System displays the details	good
store owner	Store id is invalid/ user is not store owner/user is store manager without the right permissions.	The operation fails and an error is displayed	bad

System Manager use cases:

Use case: removing membership of a member. (6.2)

- Actors: system manager, member
- Precondition: system manager is logged in and is a system manager, member registered to the system
- Postcondition: member is removed from the system and is turned into a guest
- Parameter: member details.
- Actions:

    1. system manager goes to manage employee page
    2. system manager chooses to remove the member
    3. System checks that system manager is a system manager and that the member does not have any permissions in the system. - if not present an appropriate message. -if member has permissions present an appropriate message
    5. System log out the member and removes him from the system
    6. System sends a notification to all the appointies that their permission to he store has been removed

Participants	Parameters	Expected Result	Scenario
store owner, appointed store manager	The store owner of the specific store and the store owner is the appointer of the store owner	store manager permissions are updated.	good
store owner, appointed store manager	store owner is not a manager of this store / store owner is not the appointer of the store owner	The operation fails and an error is displayed	bad

Use case: Purchases information history. (6.4)

- Actor: System Manager user
- Precondition: System Manager is logged in.
- Postcondition: System Manager receives store purchases information history.
- Parameter: None
- Actions:
    1. User request to see store purchases information history.
    2. System checks that the user has permissions to perform this operation.

3. System if user has permissions – asks from the user for store details

3. User inserts store details

4. System checks that store exist

5. System if action succeeded – search for Store purchases information history.

6. Store if search succeeded – present purchases information history. if not - show an appropriate message

Participants	**Parameters**	**Expected Result**	**Scenario**
System Manager	Good and valid Store details	Store purchases information history is presented	Good
System Manager	Bad and invalid Store details	Store purchases information history is not presented, appropriate message presented to user	Bad
System Manager	No specific store details	All Stores purchases information history is presented	Bad

Use case: Receiving members inforamtion. (6.6)

- Actor: System Manager user

- Precondition: System Manager is logged in.

- Postcondition: System Manager receives members inforamtion.
- Parameter: None
- Actions:
    1. User request to see store members information.
    2. System checks that the user has permissions to perform this operation.
    3. System if user has permissions – present members information. if not - show an appropriate message

Participants	**Parameters**	**Expected Result**	**Scenario**
System Manager	System manager has the correct permissions	membres information is presented	Good
System Manager	System manager does not have the correct permissions	membres information is is not presented, appropriate message presented to user	Bad

Use case: Placing a bid - guest.

- Actor: guest, store managers
- Precondition: item exists in shop, bid price is positive and is lower than the current item price.
- Postcondition: Bid offer is sent to all shop owners
- Parameter: Shop, Item, bid-price
- Actions:

1. Guest requests to place a new bid on an item.
2. Guest enters the bid offer price.
3. System checks that the price entered is is positive and is lower than the current item price - if not presents an error message
4. System creates a new Bid object.
5. System notifies all the store managers with the right permissions
6. If one of the managers rejected the offer:
   - System notifies the Guest that his offer was denied, and the bid is deleted from hr system.
7. If one of the managers made a counter offer:
   - System changes the bid price to the new counter offer
   - System waits for the reminding store managers who haven't approved the offer yet to respond
   - If they accept, the item price is changed for the guest, and the guest is notified about the new price
   - If one of them declines then the system notifies the Guest that his offer was denied, and the bid is deleted from hr system.
   - System notifies the Guest that his original offer offer was denied but he got a counter offer that was approved.

8. If all the managers approved the the bid offer:

   - System changes the item price for the guest, and the guest is notified about the new price

9. If at any point the guest exists the system, then the bid is deleted

Participants	**Parameters**	**Expected Result**	**Scenario**
Guest	Valid price	Bid offer is added to the system	Good
Guest	Invalid price	Bid offer is not added to the system, and appropriate erro message presented to user	Bad

Use case: Placing a bid - member.

- Actor: member, store managers
- Precondition: item exists in shop, bid price is positive and is lower than the current item price.
- Postcondition: Bid offer is sent to all shop owners
- Parameter: Shop, Item, bid-price
- Actions:

1. Member requests to place a new bid on an item.
2. Member enters the bid offer price.
3. System checks that the price entered is is positive and is lower than the current item price - if not presents an error message
4. System creates a new Bid object.
5. System notifies all the store managers with the right permissions
6. If one of the managers rejected the offer:

   - System notifies the member that his offer was denied, and the bid is deleted from hr system.

7. If one of the managers made a counter offer:

   - System changes the bid price to the new counter offer
   - System waits for the reminding store managers who haven't approved the offer yet to respond
   - If they accept, the item price is changed for the member, and the member is notified about the new price
   - If one of them declines then the system notifies the member that his offer was denied, and the bid is deleted from hr system.
   - System notifies the member that his original offer offer was denied but he got a counter offer that was approved.

8. If all the managers approved the the bid offer:

   - System changes the item price for the member, and the member is notified about the new price

Participants	**Parameters**	**Expected Result**	**Scenario**
Member	Valid price	Bid offer is added to the system	Good
Member	Invalid price	Bid offer is not added to the system, and appropriate erro message presented to user	Bad

Use case: Viewing system daily revenue - admin

- Actor: System Manager user
- Precondition: System Manager is logged in.
- Postcondition: System Manager receives revenue inforamtion.
- Parameter: date
- Actions:

1. User request to see system daily revenue.
2. System checks that the user has permissions to perform this operation.
3. System if user dosent have permissions - show an appropriate message
4. System if user has permission - request the user to enter a date for wich to show the revenue
5. System displays the revenue on the given date.

Participants	**Parameters**	**Expected Result**	**Scenario**
System Manager	System manager has the correct permissions	system daily revenue is presented	Good
System Manager	System manager does not have the correct permissions	system daily revenue is not presented, appropriate message presented to user	Bad

Use case: Viewing system daily revenue - store manager

- Actor: Store Manager user
- Precondition: Store Manager is logged in.
- Postcondition: Store Manager receives revenue inforamtion.
- Parameter:store, date
- Actions:

1. User request to see store daily revenue.
2. System checks that the user has permissions to perform this operation.
3. System if user dosent have permissions - show an appropriate message
4. System if user has permission - request the user to enter a date for wich to show the revenue
5. System displays the revenue of the store on the given date.

Participants	**Parameters**	**Expected Result**	**Scenario**
Store Manager	System manager has the correct permissions	system daily revenue is presented	Good
Store Manager	System manager does not have the correct permissions	system daily revenue is not presented, appropriate message presented to user	Bad

Use case: Viewing system user activity - admin

- Actor: System Manager user
- Precondition: System Manager is logged in.
- Postcondition: System Manager receives user inforamtion.
- Parameter: from-date, to-date
- Actions:

1. User request to see system user activity.
2. System checks that the user has permissions to perform this operation.
3. System if user dosent have permissions - show an appropriate message
4. System if user has permission - request the user to enter a form-date and a to-date for wich to show the user activity
5. System displays the user activity on the given dates.

Participants	**Parameters**	**Expected Result**	**Scenario**
System Manager	System manager has the correct permissions	system user activity is presented	Good
System Manager	System manager does not have the correct permissions	system user activity is not presented, appropriate message presented to user	Bad

Use case: appointing a new store owner - multiple owners. (4.4)

- Actors: appointing store owner - A, all store owners in store, member
- Precondition: A is logged in, member is registered, store exists, store owner is the owner of the store, member is not the owner of the store
- Postcondition: Employement agreement is sent to all store owners
- Parameter: member to appoint details.
- Actions:

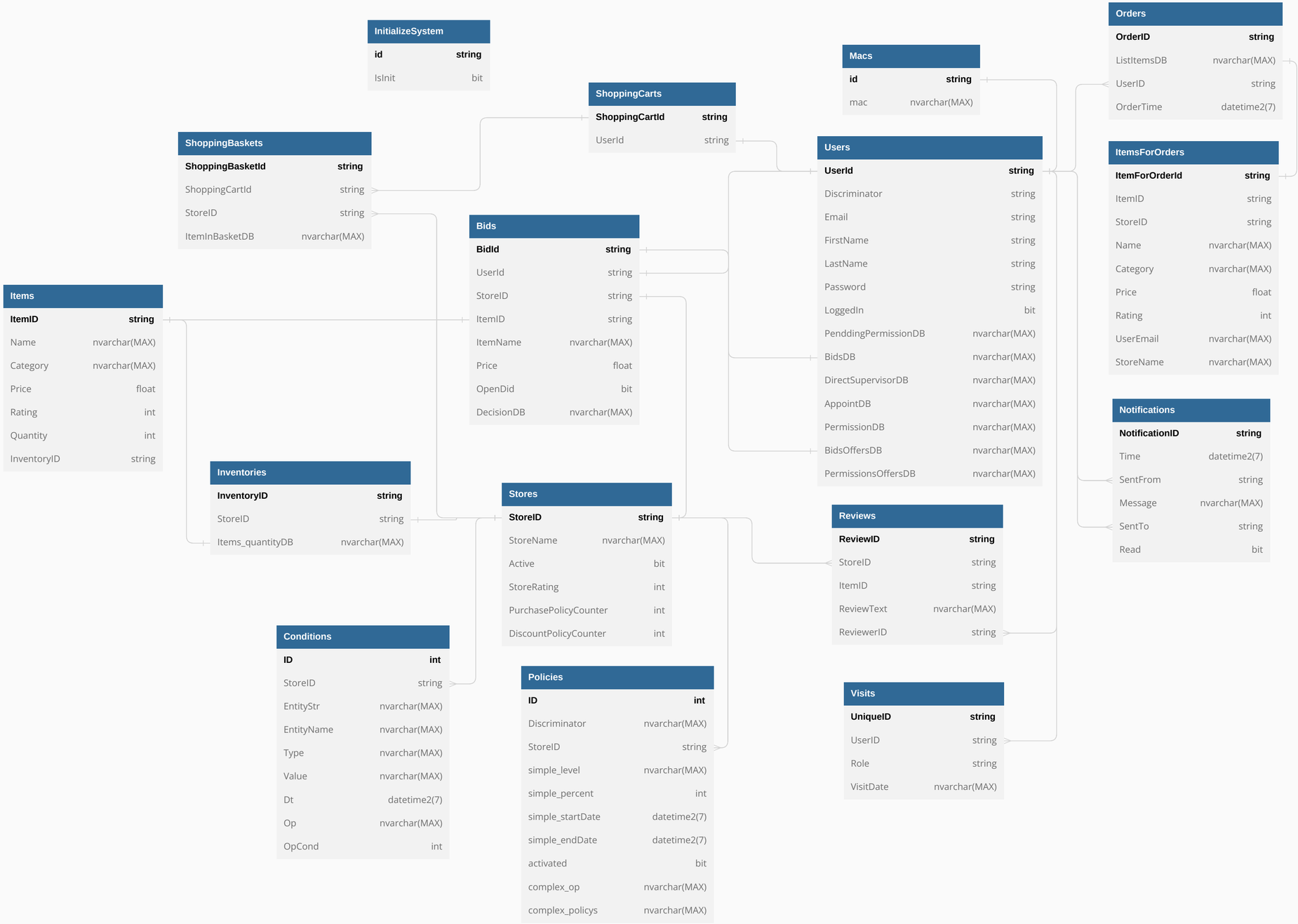
1. Store owner requests to add new store owner.
2. System asks for new store owner details.
3. System checks that new store owner details is a member in the system. If not system present appropriate message to user.
4. Permissions checks that the new store owner is not already a store owner in the store. If he is, store sends to system an appropriate message that the system will display to the user.
5. System creates a new Employement agreement.
6. System notifies all the store owners with the right permissions.
7. If one of the owners rejected the offer:

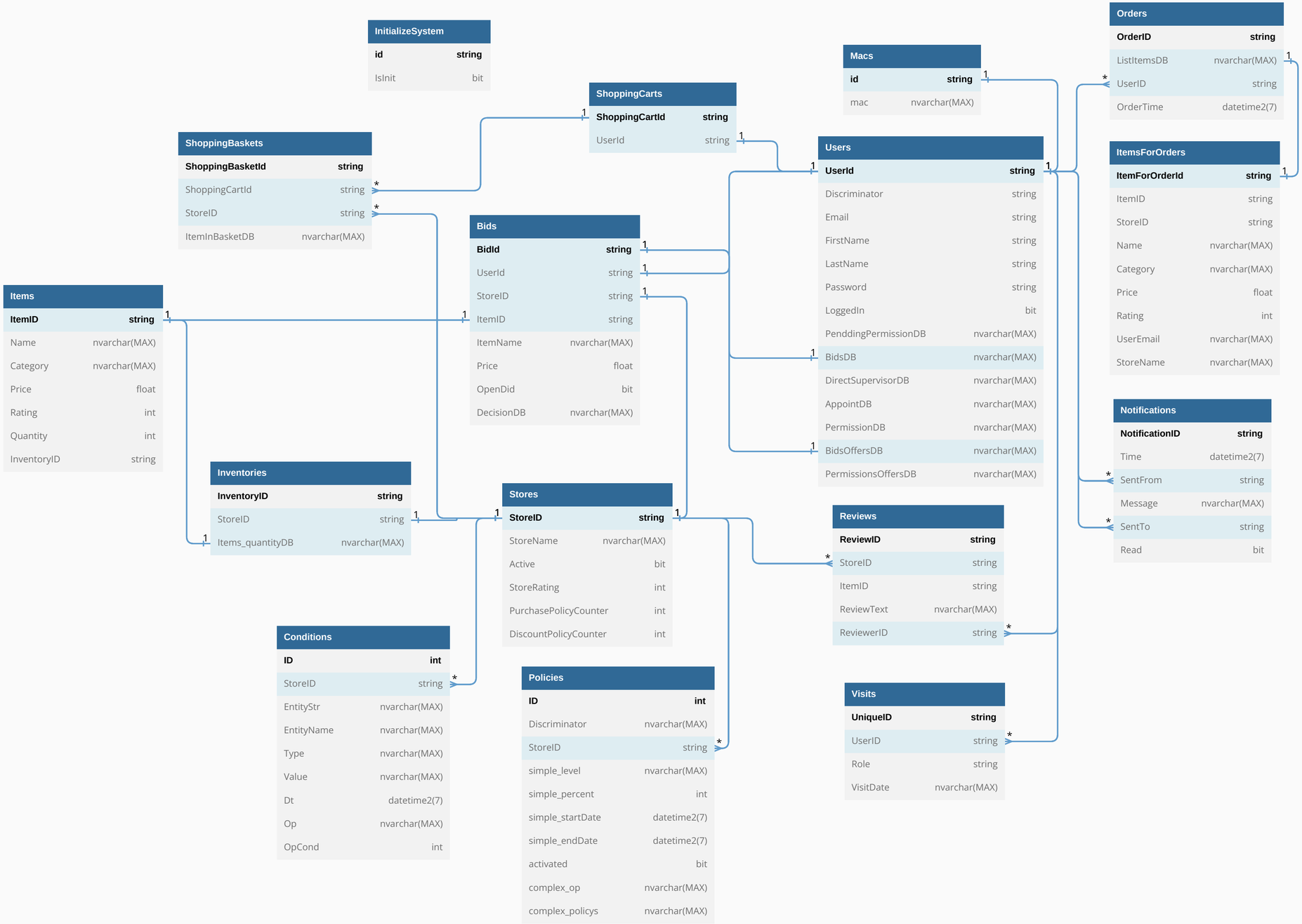
   - System notifies A that his offer was denied, and the agreement is deleted from the system.

8. If all the owners approved the the agreement

- Permissions adds member as the owner of the store and gives the new store owner permissions of management and store policy and adds A as the appointer of the new store owner.
- System sends a notification to the user about his new employment.

Participants	Parameters	Expected Result	Scenario
store owners, member	New store owner is a member and is not yet the store's owner	member is added as the owner of the store.	good
store owner, member	Store owner enter details of un-registered user	The operation fails and an error is displayed	bad





Stress & Load Test

ציפיות המערכת:

יש לנו 5 סוגים של משתמשים במערכת (לא כולל הSystem manager שיש אחד):

1. Guest - מבקר באתר שאין לו חשבון. הציפייה ל**50%** מהמשתמשים.
2. Member - משתמש הרשום למערכת המסחר. הציפייה ל**30%** מהמשתמשים.
3. Store Owner - מנהל חנות עם כל ההרשאות. הציפייה ל**10%** מהמשתמשים.
4. Store Manager - מנהל חנות עם מספר ההרשאות מוגבל. הציפייה ל**8%** מהמשתמשים.
5. Founder – פותח החנות יש לו את כל ההרשאות. הציפייה ל**2%** מהמשתמשים.

יש לנו 4 פעולות עיקריות במערכת:

1. רכישות- כולל את צפייה בעגלה, תהליך הרכישה ואת הcheckout. ציפייה ל**30%** מהפעולות .
2. חיפוש מוצרים – כולל חיפוש מוצר עם סינונים, והוספת המוצר לעגלת הקניות. ציפייה ל**50%** מהפעולות.
3. הגשת הצעת קנייה – (bid) ציפייה ל**5%** מהפעולות.
4. פעולות ניהול- הכוללות פתיחת חנות, ניהול מוצרים (עדכון, הוספה והסרת מוצרים), הוספת מנהל חנות והוספת בעל חנות. ציפייה ל**15%** מהפעולות.

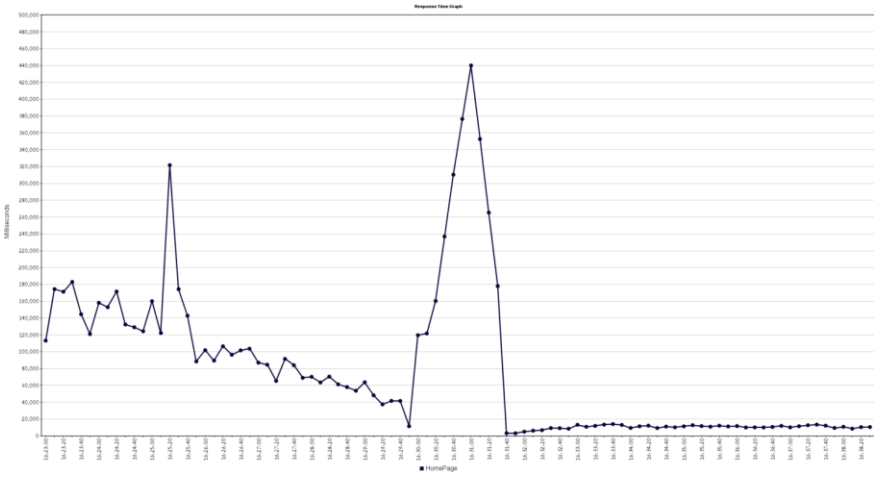
מספר המשתמשים הצפויים להירשם למערכת 10 ביום.

1. User **Load**

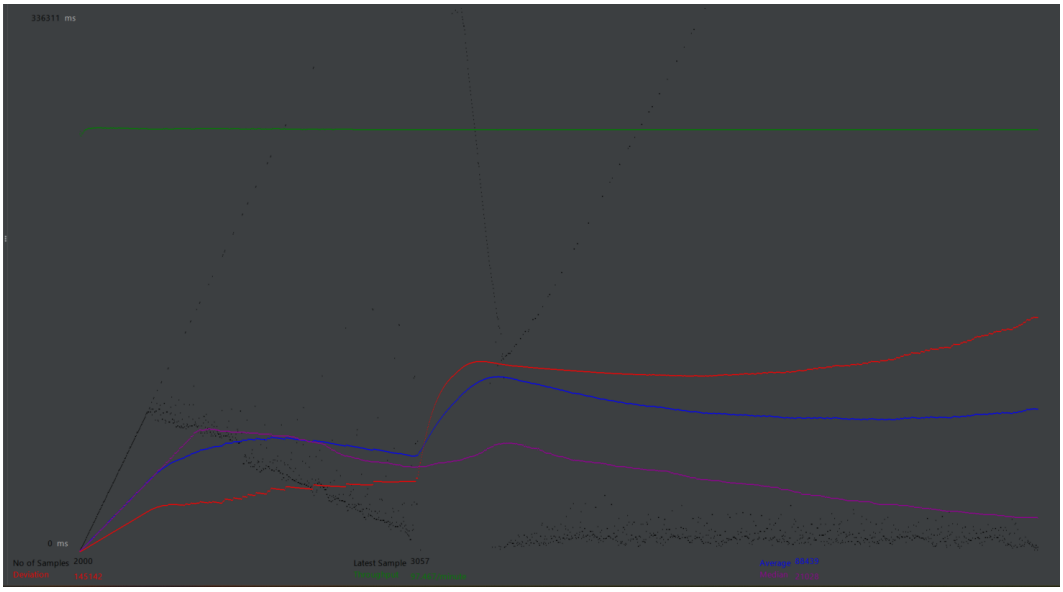
בדיקת כניסה של 200 משתמשים שכל אחד מהם מבצע "enter" למערכת של 10 פעמים.

קיבלנו: 97.468 -throughput ובנוסף 0 שגיאות.

מצורף גרף הresponse time אשר ניתן להסיק ממנו את זמן הresponse לפי השעה.



וגרף אשר ניתן לראות הthrougput.



2. Search Functionality **Load**:

חיפוש מוצרים על ידי members 9 (שנמצאים בloaddata) ועל ידי עוד 100 guest. כאשר כל משתמש מחפש 10 פעמים. הmembers מחפשים את הkeyword "tow" בעוד הguest מחפשים מוצרים שמחירם עם 100 ₪.

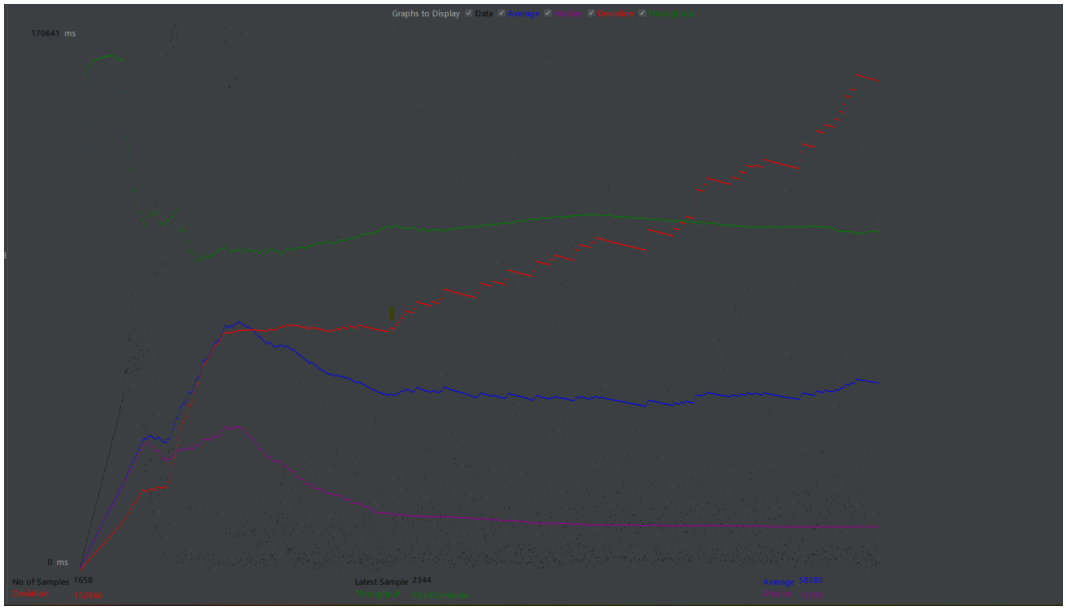
נזכיר כי תהליך הmember בתרחיש הזה הוא כניסה->התחברות->חיפוש->התנתקות,

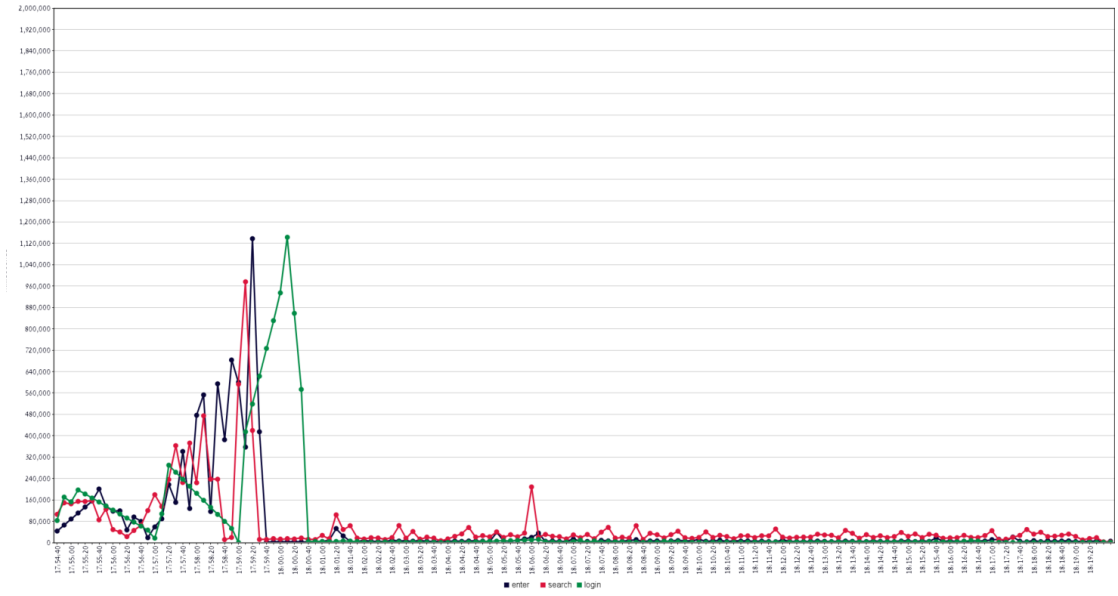
ותרחיש ה guest הוא כניסה->חיפוש.

ראשית נשים לב כי באמת אנחנו מקבלים response מתאים:

```
{
"value": [
  {
    "itemId": "2f185d98-7dcb-4c8f-9f47-7dc5df60a202",
    "name": "Teddy bear toy",
    "category": "children toys",
    "price": 65.0,
    "priceDiscount": -1.0,
    "offerPrice": -1.0,
    "openBid": false,
    "rating": 0,
    "storeId": "c574df63-e73a-45f1-9639-bceca4c3e799",
    "inStock": true,
    "count": 0
  },
  {
    "itemId": "69953967-81fe-4476-91ba-3ea6e3634d63",
    "name": "mouse",
    "category": "animals",
    "price": 65.0,
    "priceDiscount": -1.0,
    "offerPrice": -1.0,
    "openBid": false,
    "rating": 0,
    "storeId": "c574df63-e73a-45f1-9639-bceca4c3e799",
    "inStock": false,
    "count": 0
  },
  {
    "itemId": "f443b6d8-c8f0-4112-9c56-7f3ac4c15154",
    "name": "Towel",
    "category": "Home",
    "price": 40.0,
    "priceDiscount": -1.0,
    "offerPrice": -1.0,
    "openBid": false,
    "rating": 0,
    "storeId": "c574df63-e73a-45f1-9639-bceca4c3e799",
    "inStock": true,
    "count": 0
  },
```


קיבלנו: 65.938-throughput ובנוסף 0 שגיאות.





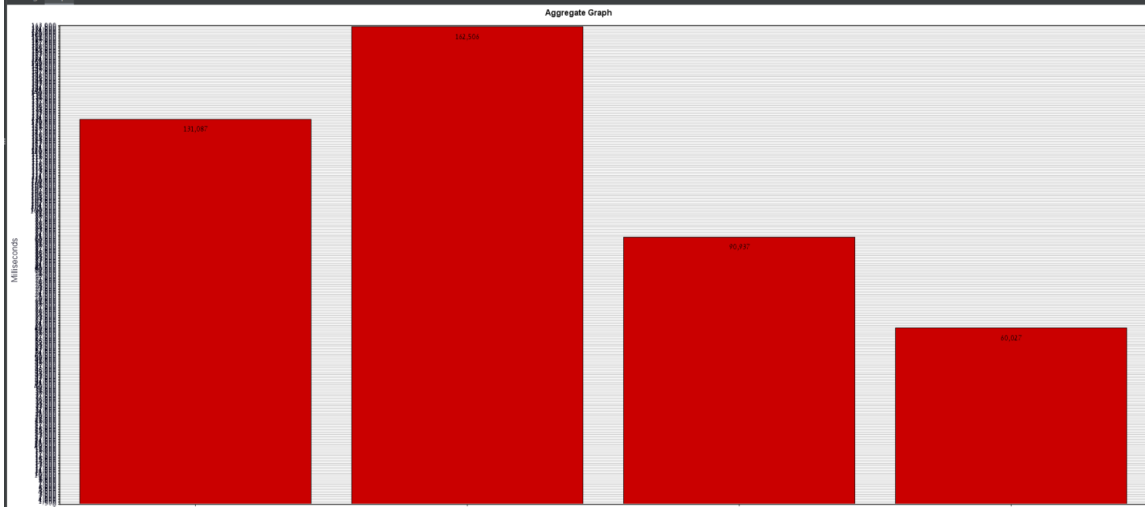
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
enter	828	57231	5576	153480	257267	941338	955	1243755	0.00%	31.8/min	0.10	0.07
search	804	68311	21173	178850	219150	1101861	967	1279527	0.00%	30.8/min	1.06	0.18
login	90	35736	2409	62500	185276	414778	438	1142607	0.00%	3.6/min	0.02	0.02
TOTAL	1722	61281	13710	167326	224696	1068478	438	1279527	0.00%	1.1/sec	1.17	0.27

3. Search Functionality **Stress**:

נבצע בדיוק את אותו הטסט כמו מקודם רק עם 300 משתמשים במקום 100. ונצפה בעצם שהמערכת שלנו תקרוס.

כמו שניתן לראות מספר השגיאות בחיפוש המוצרים היה כ9.56% מכלל הבקשות. בנוסף היה קושי גם להתחבר כאשר הגענו לפיק אפ.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
enter	1276	130992	17337	444585	636335	1418525	806	1594292	5.96%	32.5/min	0.17	0.07
search	1172	162506	42587	644985	680772	782165	1403	1226502	9.56%	29.9/min	1.12	0.18
login	29	90937	3931	128763	645687	651962	472	651962	10.34%	54.0/hour	0.01	0.01
logout	27	60027	2320	121494	473794	527144	436	527144	3.70%	57.5/hour	0.01	0.01
TOTAL	2504	144513	34718	561970	677968	1213338	436	1594292	7.67%	1.1/sec	1.30	0.23



Aggregate Graph

Name: Aggregate Graph

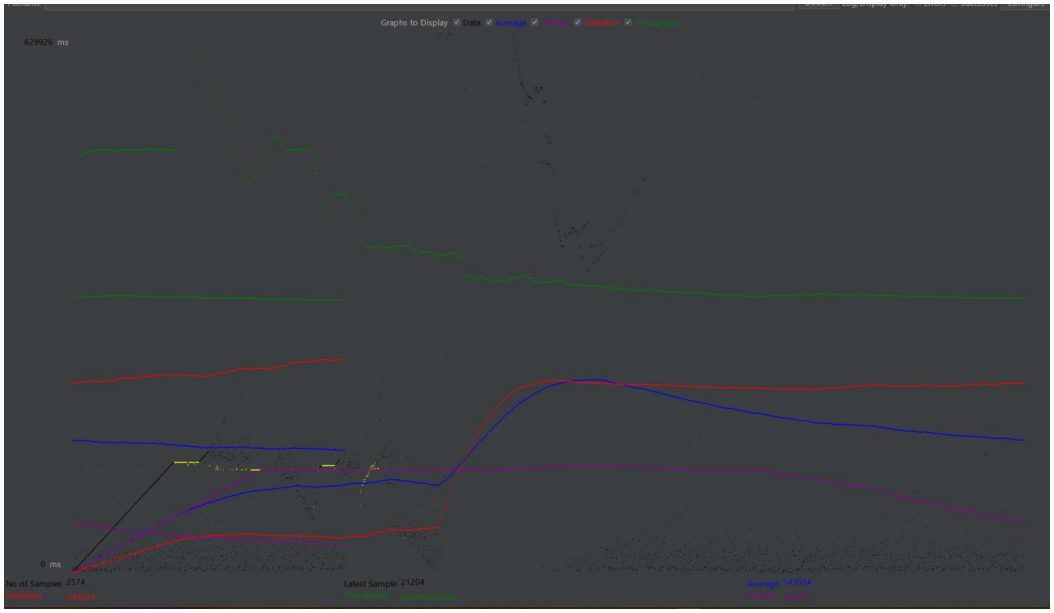
Comments:

Write results to file / Read from file

Filename

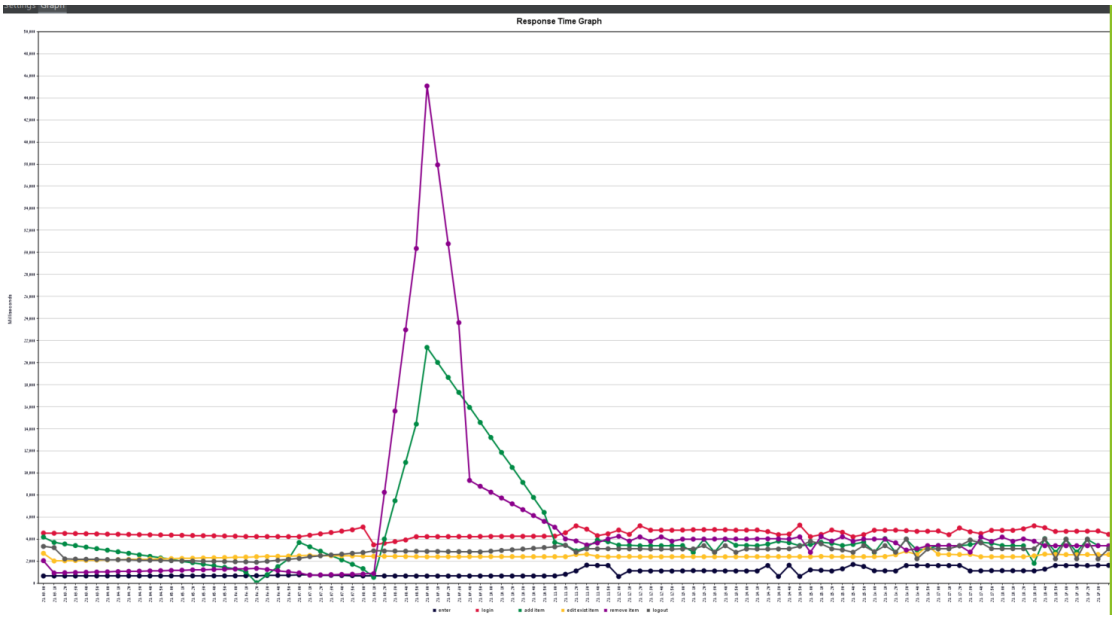
Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
enter	1258	131483	18374	444789	636335	1399802	806	1570438	6.04%	32.5/min	0.17	0.0
search	1154	164400	44140	645387	681176	782165	1403	1226502	9.71%	29.9/min	1.12	0.1
login	29	90937	3931	128763	645687	651962	472	651962	10.34%	54.0/hour	0.01	0.0
logout	27	60027	2320	121494	473794	527144	436	527144	3.70%	57.5/hour	0.01	0.0
TOTAL	2468	145616	35499	562962	678031	1205957	436	1570438	7.78%	1.1/sec	1.30	0.2



4. Product Listing and Update **Load**:

בדיקת של שני בעלי החנות שלנו מוסיפים פריטים חדשים, עורכים פריט קיים ולאחר מכן מסירים את הפריט החדש כך 100 פעמים.



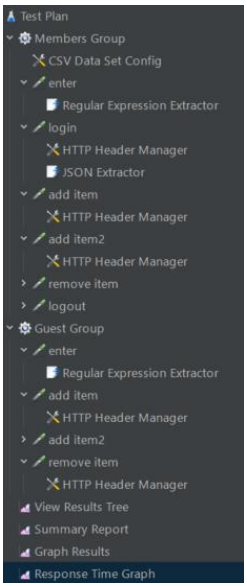
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
enter	66	1308	597	2387	484.66	0.00%	6.5/min
login	66	4732	3403	6489	520.94	0.00%	6.5/min
add item	65	3402	1795	4832	747.81	0.00%	6.5/min
edit exist item	65	2491	1854	3387	200.81	0.00%	6.5/min
remove item	64	3706	2397	4246	423.15	0.00%	6.5/min
logout	64	3141	2192	5024	668.22	0.00%	6.5/min
TOTAL	390	3128	597	6489	1191.13	0.00%	38.3/min

5. Shopping Cart **Load**:

הוספת והסרת מוצרים מהעגלה הטסט מתבצע על ידי 9 members (שנמצאים בloaddata) ועל ידי עוד 100 guest. כאשר כל משתמש מחפש 10 פעמים.

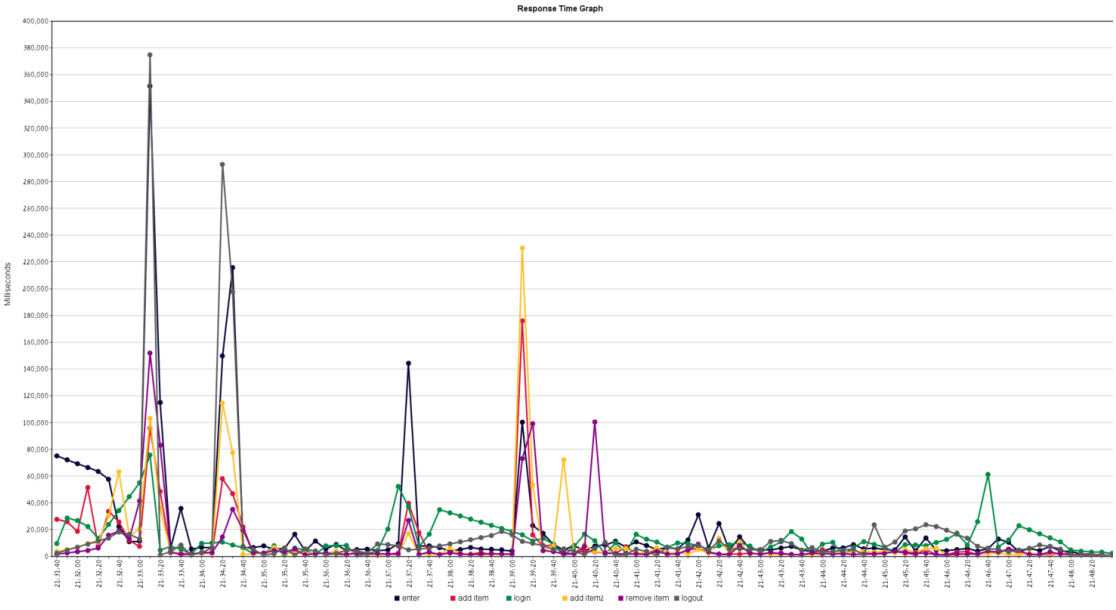
נזכיר כי תהליך הmember בתרחיש הזה הוא כניסה->התחברות->הוספת פריט-> הוספת פריט 2 -> הסרת פריט 1->התנתקות,

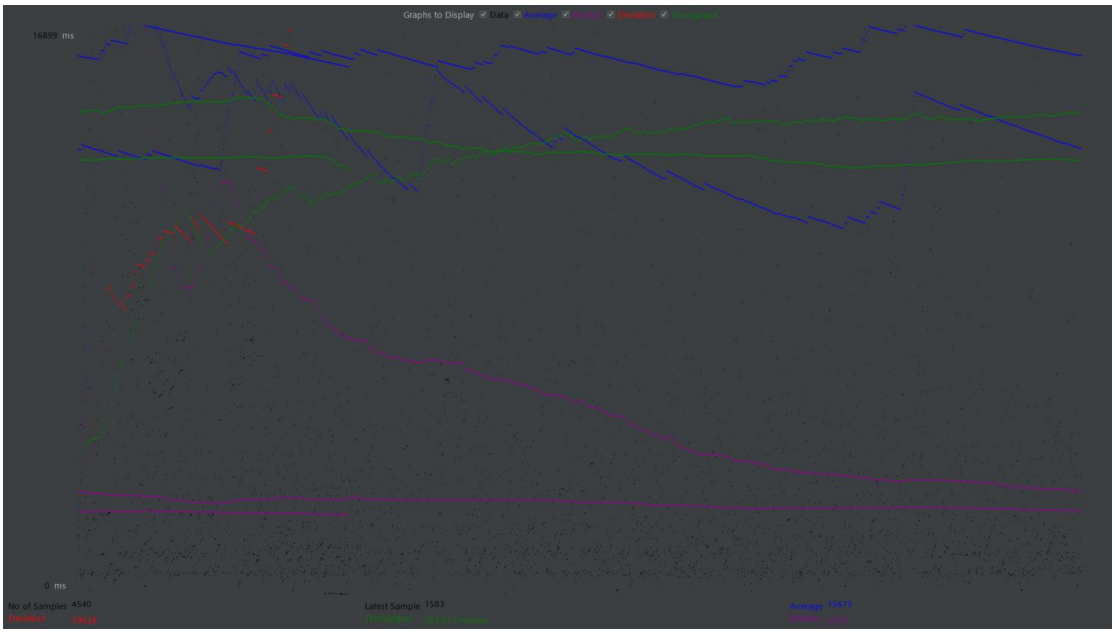
ותרחיש ה guest הוא כניסה ->הוספת פריט-> הוספת פריט 2 -> הסרת פריט 1



סיכום תוצאות הטסט:

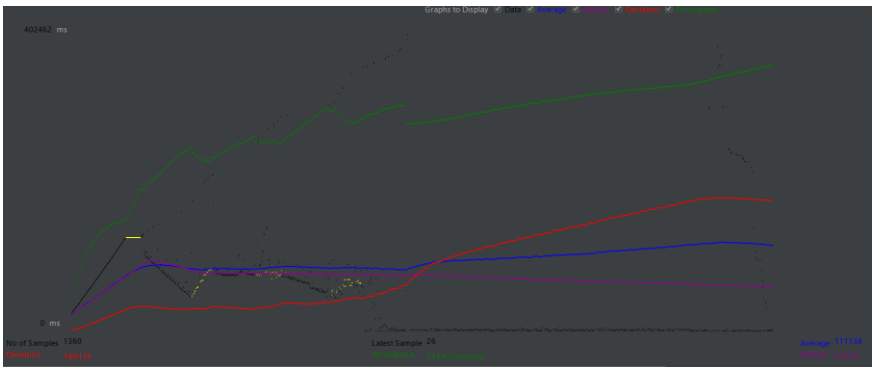
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
enter	1090	30272	590	628040	88759.36	0.00%	1.1/sec
add item	1090	9252	23	447723	38874.38	0.00%	1.1/sec
login	90	11214	33	75713	12709.26	0.00%	5.3/min
add item2	1090	12455	2	459393	52572.66	0.00%	1.1/sec
remove item	1090	11242	22	395715	46460.34	0.00%	1.1/sec
logout	90	13726	26	374722	48944.93	0.00%	5.5/min
TOTAL	4540	15673	2	628040	59628.40	0.00%	4.4/sec

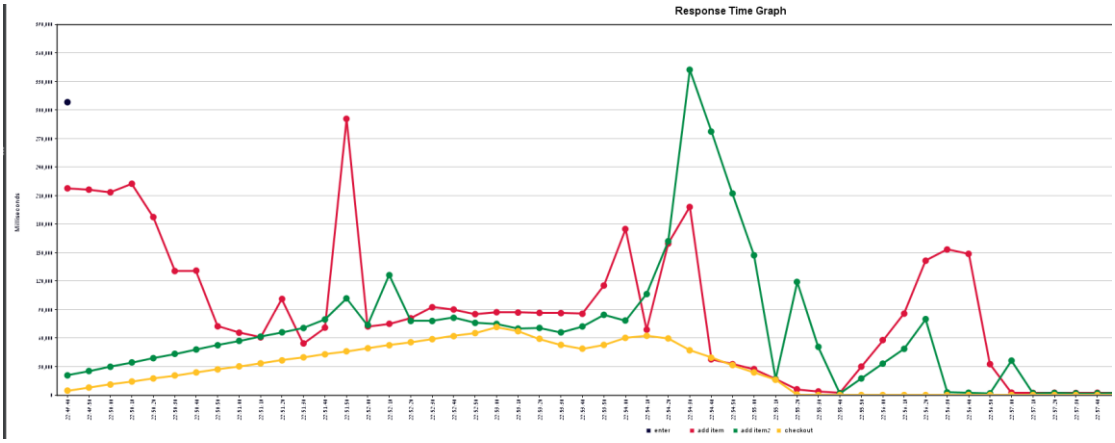


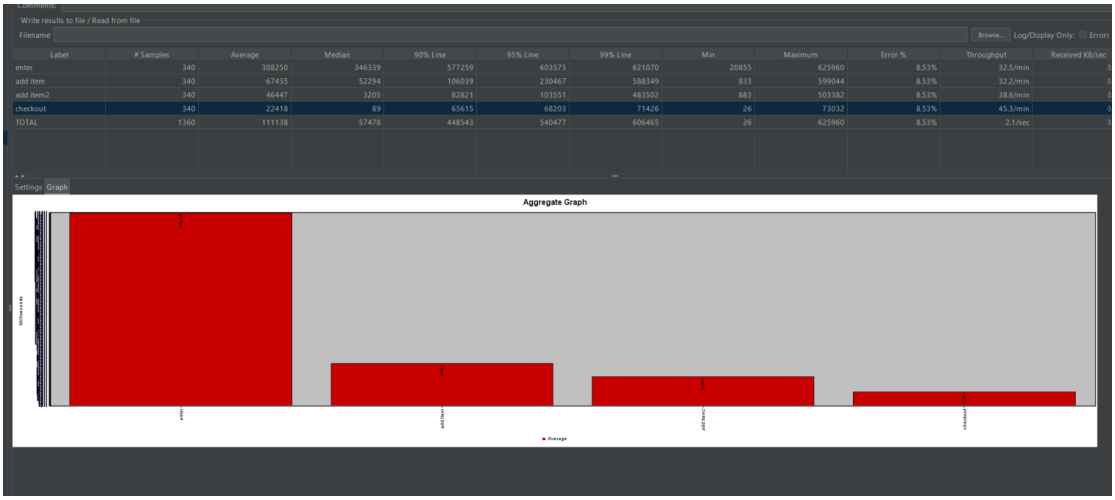


6. Checkout Process **Stress**:

נבצע את הטסט עם 350 משתמשים (guest) כאשר בטסט התרחיש יהיה להכנס למערכת, להוסיף מוצר 1 ולהוסיף מוצר 2 ולבצע checkout בסוף (כלומר בדיקה שהקונדיישנס עובדים).

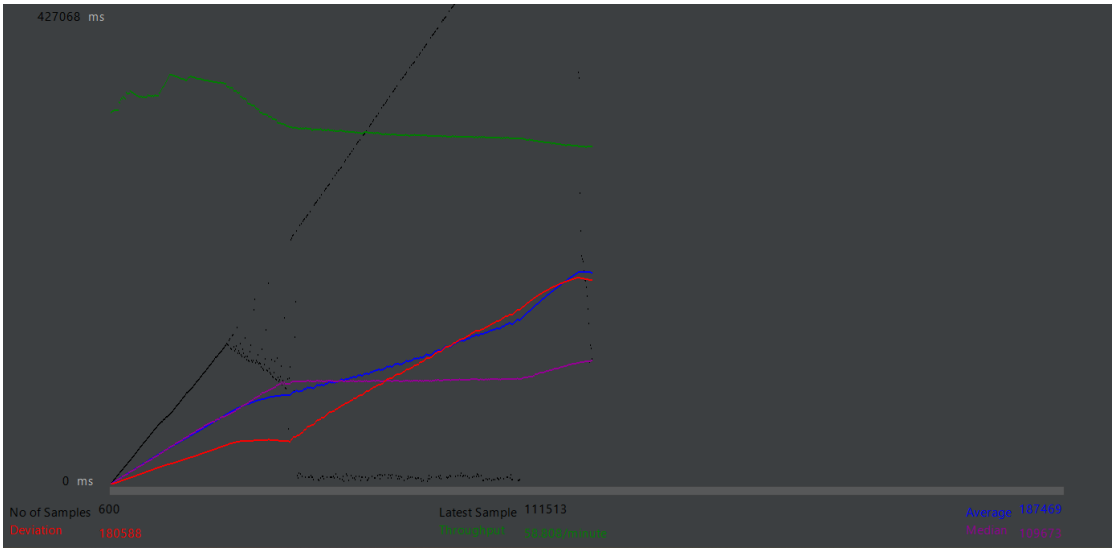


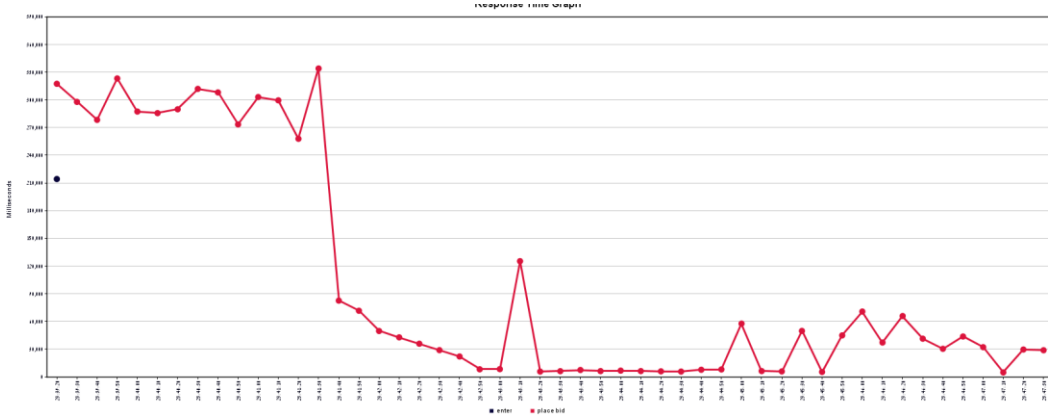




7. Bidding **Load**:

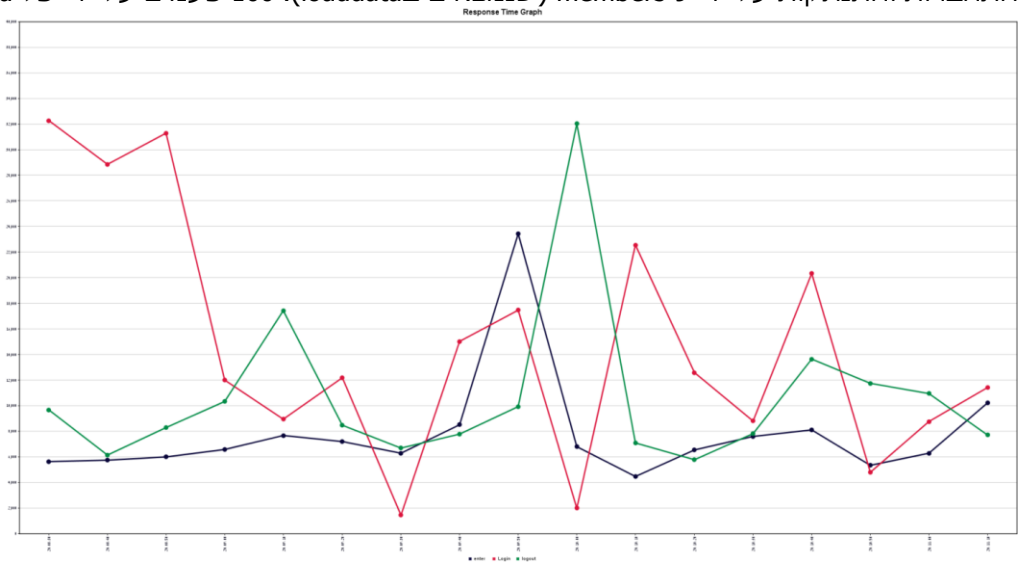
נשלח מ200 משתמשים (guest) הצעת קנייה על מוצר שיש במערכת במטרה להוריד את המחיר. כאשר כל משתמש יבצע בקשה אחת על כל מוצר.





8. Login and Logout **Load**:

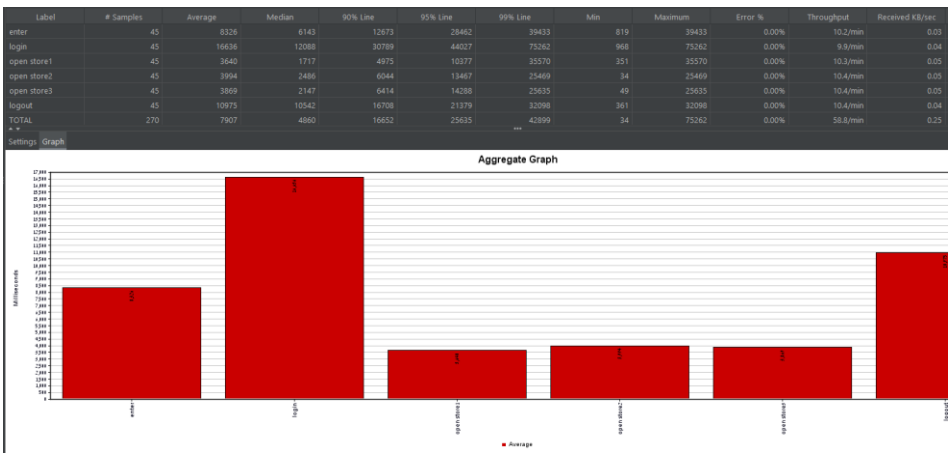
התחברות והתנתקות על ידי 9 members (שנמצאים בloaddata). 100 פעמים על ידי כל thread.



9. Open store **Load**:

פתיחת 3 חנויות חדשות על ידי כל אחד מ9 members שנמצאים ב (loaddata).

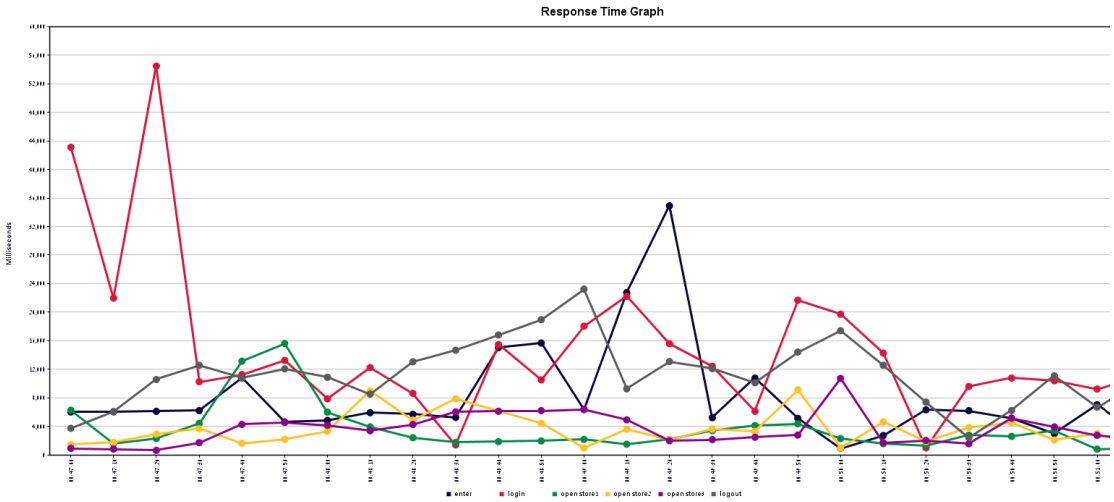
תהליך הmember בתרחיש הזה הוא כניסה->התחברות->הוספת חנות-> התנתקות



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec
enter	45	8326	6143	12673	28462	39433	819	39433	0.00%	10.2/min	0.03
login	45	16636	12088	30789	44027	75262	968	75262	0.00%	9.9/min	0.04
open store1	45	3640	1717	4975	10377	35570	351	35570	0.00%	10.3/min	0.05
open store2	45	3994	2486	6044	13467	25469	34	25469	0.00%	10.4/min	0.05
open store3	45	3869	2147	6414	14288	25635	49	25635	0.00%	10.4/min	0.05
logout	45	10975	10542	16708	21379	32098	361	32098	0.00%	10.4/min	0.04
TOTAL	270	7907	4860	16652	25635	42899	34	75262	0.00%	58.8/min	0.25

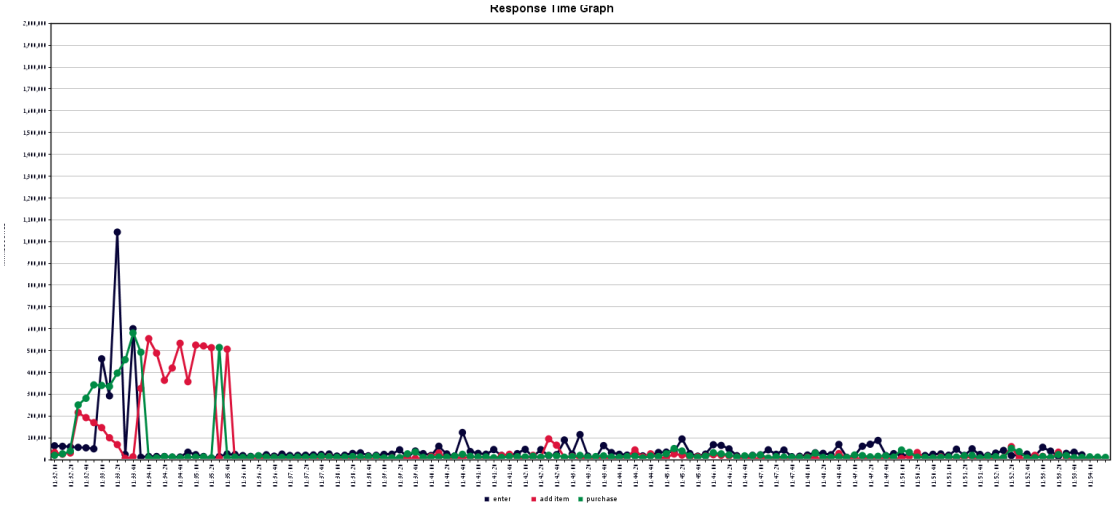
Settings Graph

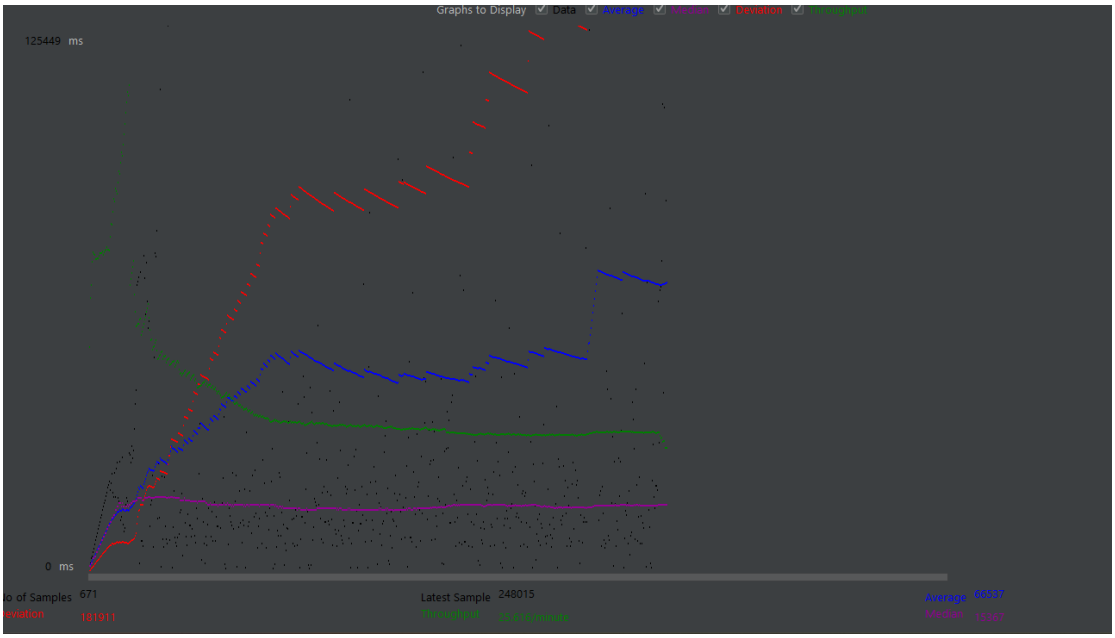
Aggregate Graph



10. Purchase Testing **Load:**

ביצוע רכישה על ידי 50 משתמשים 10 פעמים.





סיכום המערכת:

כל הבדיקות נעשו עם המערכות החיצוניות על מנת לבדוק התמודדות אמיתית עם המערכת.
נשים לב שבגלל כך ביצועי המערכת יותר איטיים מהרגיל.

בדקנו גם מצבי קיצון בהם המערכת קרסה.

המערכת מצליחה להתמודד עם מעל 100 משתמשים במקביל שנכנסים לעמודים שונים!

מספר הבקשות הממוצע שהמערכת מצליחה להתמודד איתם היא אחת לשנייה.

ומספר השגיאות מינורי.