

תוכן עניינים

2	1. מטרת
3	2. תכנון הבדיקות
3	2.1. חומרה לצורך ביצוע הבדיקות
3	2.2. תמהיל השימוש במערכת
4	3. תרחישי בדיקות הקיבול
4	תרחיש 1 – רישום למערכת
6	תרחיש 2 – ביקור במערכת
7	תרחיש 3 – התחברות למערכת
7	תרחיש 4 – חיפוש במערכת
9	תרחיש 5 – פתיחת חנות ומינוי בעל חנות נוסף
10	תרחיש 6 – רכישה
11	3.1. תרחישי בדיקות עומס
11	3.2. ניתוח תוצאות בדיקות הקיבול
11	תרחיש 1 – רישום למערכת
12	תרחיש 2 – ביקור במערכת
13	תרחיש 3 – התחברות למערכת
14	תרחיש 4 – חיפוש במערכת
15	תרחיש 5 – פתיחת חנות ומינוי בעל חנות נוסף
16	תרחיש 6 – רכישה
18	3.3. ניתוח תוצאות בדיקות העומס
18	תרחיש 1 – רישום למערכת
19	תרחיש 3 – התחברות למערכת
21	תרחיש 4 – חיפוש במערכת
22	תרחיש 5 – פתיחת חנות ומינוי בעל חנות נוסף
23	תרחיש 6 – רכישה
24	4. סיכום:

ביצועי המערכת

1. מטרות:

מערכת מסחר היא מערכת מרובת משתמשים ולכן צריכה להיות מסוגלת לתמוך במספר לא מוגבל של משתמשים מכל הסוגים וכן במספר לא מוגבל של חנויות, מוצרים, חיפושים ורכישות.

היעדים אליהם נשאף להגיע בכדי להבטיח ביצועי מערכת טובים ויעילים הם :

- על המערכת לתמוך במספר לא מוגבל של משתמשים מכל הסוגים וכן במספר לא מוגבל של חנויות, מוצרים, חיפושים ורכישות.
 - יש לתמוך בקצב מהיר של הצטרפות ועזיבה של משתמשים – אורחים ומנויים, ופתיחת חנויות.
 - על המערכת להיות זמינה ונגישה באופן תמידי (למעט פעולות סגירה יזומות.
- בכדי לעמוד ביעדים שהגדרנו, נגדיר מדדים עבור עמידה ביעדים , המטרה העיקרית היא לעמוד ביעדים שאנו מגדירים. מדדים אלו הם :
- על המערכת להתמודד עם 100 בקשות(אירועים כגון התחברות, רישום, חיפוש רכישה וכו') בו זמנית תוך עמידה בזמן תגובה של לכל היותר שניה לכל בקשה.
 - על המערכת להיות מסוגלת לתמוך בהיקף של עד 1,000 חנויות, כאשר בכל חנות יש בממוצע 1,000 מוצרים, בהיקף של 10,000 משתמשים רשומים ובהיסטוריית של עד 1,000,000 רכישות.
 - תמיכה ב- 1,000 מבקרים במערכת בכל רגע נתון.
 - המערכת איננה מפסיקה לפעול, גם כשיש אירועים לא צפויים, כמו נפילות תקשורת או קשר לרכיבים שונים (למעט פעולות סגירה יזומות.

אנו נבטיח ללקוח את הדברים הבאים :

- סך הבקשות יהיו עם זמן תגובה החורג משניה אחת ומגיע לכל היותר לחמש שניות (ללא בקשות החורגות מחמש שניות).
- עמידה מלאה במדדים.
- המערכת יכולה שלא לפעול 0.05% מהזמן (כ- 4.5 שעות בפרק זמן של שנה).

לצורך הבטחה של עמידה מלאה במדדים, נבצע בדיקות קיבול והעמסה וננתח את תוצאות הבדיקות לצורך שיפור והבנה של ביצועי המערכת.

2. תכנון הבדיקות:

2.1. חומרה לצורך ביצוע הבדיקות:

- מערכת הפעלה: מערכת הפעלה Windows 10 64 bit
- מעבד: Intel i7 7th gen
- זכרון : 8GB RAM

הערה: כל הבדיקות יתבצעו באמצעות Apache JMeter

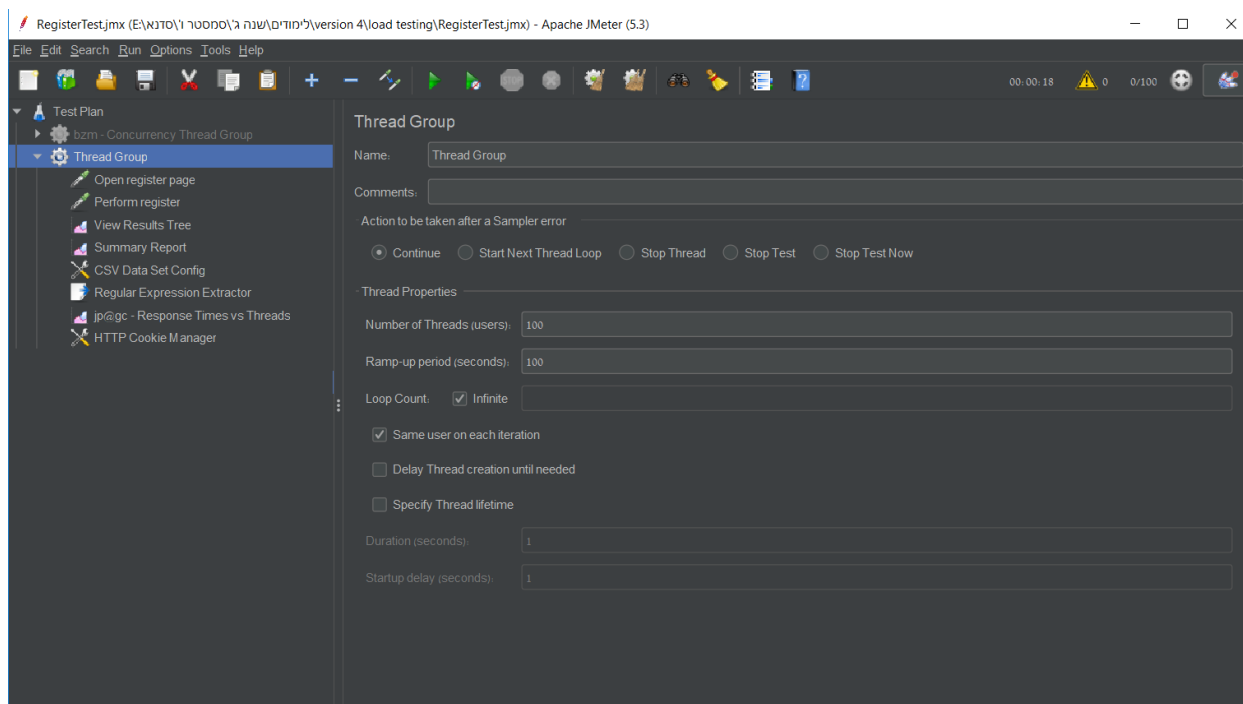
2.2. תמהיל השימוש במערכת:

- אחוז גדול יותר ממשתמשי המערכת יהיו אורחים שמטרתם לבצע קנייה חד פעמית, חלקם ימשיכו כמשתמשים קבועים ורשומים במערכת אך הציפייה היא כי כמות גדולה יותר של משתמשים יבצעו קנייה בתור אורחים.
- קצב ההצטרפות של משתמשים למערכת הצפוי הוא כ 100 משתמשים ביום.
- 10% מתוך סך פעולות המערכת הן פעולות רכישה.
- במקרים בהם יתקיימו במערכת הנחות גדולות הצפי הוא לכמות גדולה של משתמשים אורחים במערכת שייכנסו על מנת לממש את ההנחה.
- כמות בעלי החנות הצפויים במערכת הוא ככפול 2 מכמות החנויות במערכת, כמו כן כמות מנהלי החנויות הצפוי הוא ככמות החנויות במערכת.
- כל משתמש שמחובר למערכת עלול לבצע חיפוש בכדי לחפש את המוצר שבשבילו נכנס למערכת.
- מנהל המערכת יבצע פעולות ספציפיות שרק הוא רשאי לבצע ולא פעולות אחרות נוספות.

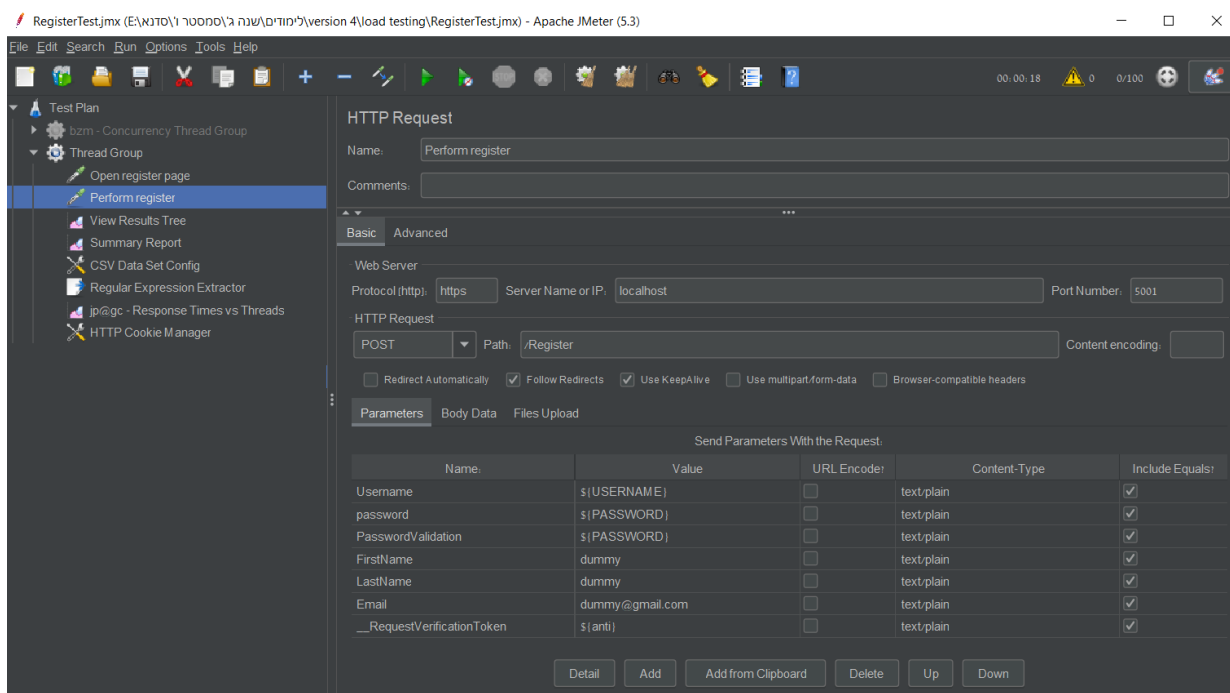
3. תרחישי בדיקות הקיבול:

תרחיש 1 – רישום למערכת

התרחיש כולל כמות משתמשים המנסים להירשם למערכת, תהליך הרישום מתבצע בזמן זמנית עבור כל המשתמשים. תחילה נתחיל במשתמש אחד שמנסה לבצע רישום ונראה כיצד המערכת מגיבה. במהלך דקה שלמה נגדיל את מספר המשתמשים המנסים לבצע רישום בזמן זמנית עד שנגיע לעומס צפוי של 100 משתמשים שמנסים לבצע רישום בזמן זמנית. סוג המשתמשים שנבנים למערכת הם משתמשים אורחים שברצונם להפוך למשתמשים רשומים. המשתמשים צריכים לנווט בעמוד הראשי לעמוד ההרשמה ומשם להכניס את פרטיהם ולבצע רישום למערכת. לצורך ביצוע הבדיקה נגדיר Thread group של 100 משתמשים מדומים המנסים לבצע הרשמה במקביל. בכדי לבדוק את כל האפשרויות נתחיל עם משתמש מדומה אחד ובכל שנייה נוסיף משתמש מדומה נוסף. נראה זאת:



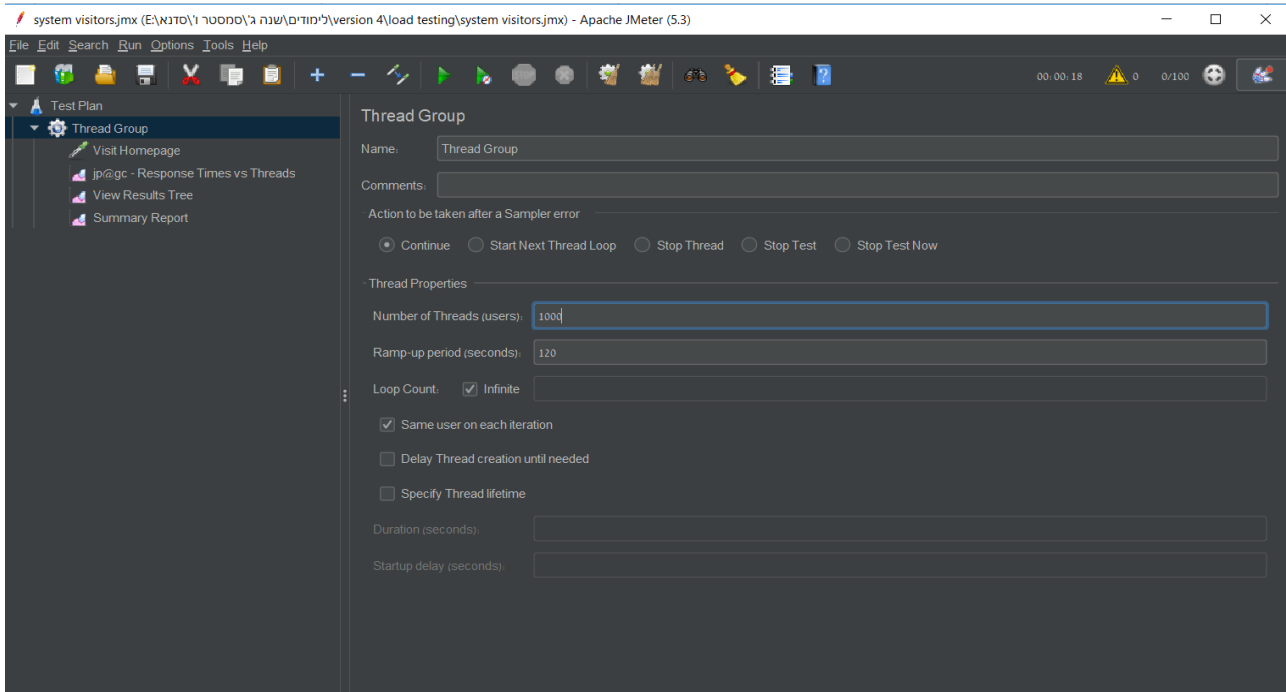
כמו כן לצורך ביצוע הבדיקה נגדיר HTTP GET request לעמוד ההרשמה ו POST request בכדי להכניס את נתוני ההרשמה. נתוני ההרשמה יילקחו מקובץ csv המכיל פרטי משתמשים והסיסמאות שלהם. נציג את בקשת ה POST שנבצע:



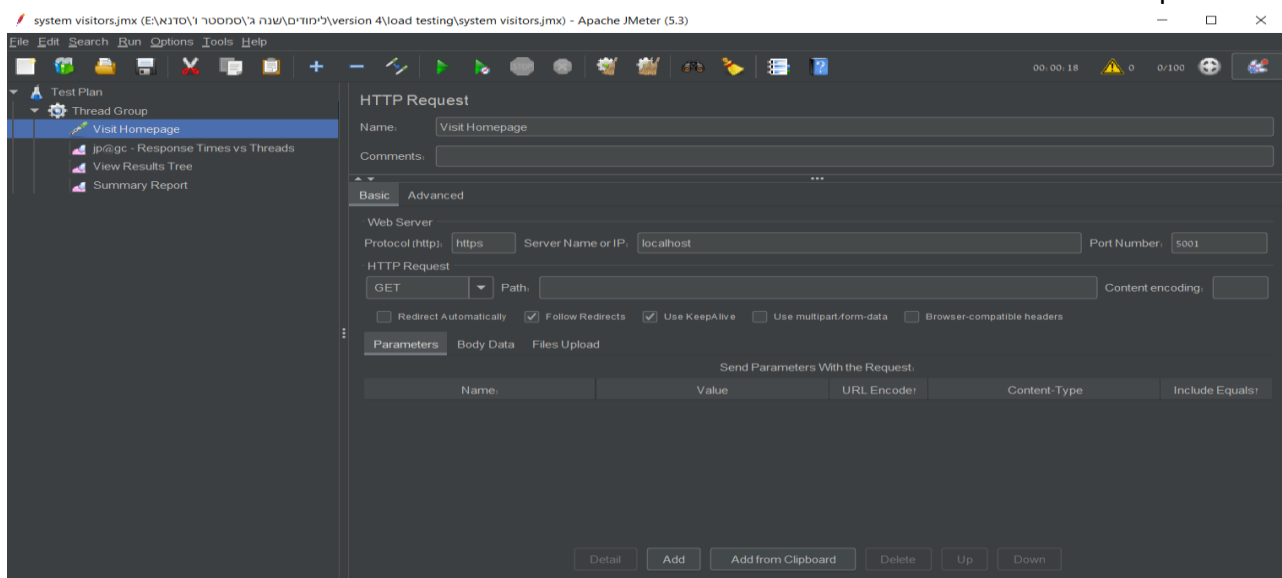
נשים לב כי הפרמטרים USERNAME ו PASSWORD ניתנים בעזרת ה CSV Data Set Config שהגדרנו.כמו כן נבצע generate ל session token באמצעות Regular Expression Extractor.כמו כן בעזרת ה Listeners ננתח את התוצאות בצורות שונות בהמשך.

תרחיש 2 – ביקור במערכת

המערכת צריכה לתמוך בביקור של 1000 משתמשים בכל רגע נתון. באמצעות תרחיש בדיקה זה נסמלץ ביקור של 1000 משתמשים בו זמנית לעמוד הבית של מערכת המסחר. יש לציין כי עמוד זה כולל בתוכו תמונות רבות, דבר אשר עלול להשפיע על זמן התגובה של השרת. תרחיש בדיקה זה יסמלץ עומס שונה ומשתנה במערכת כך שלאחר 2 דקות נגיע לעומס צפוי של 1000 משתמשים במקביל. לצורך כך נגדיר Thread group שיסמלץ לנו את כמות המשתמשים המדומים שלנו בצורה הבאה:



נשים לב כי במהלך 2 דקות אנו מעלים את קצב ההצטרפות של המשתמשים המדומים שלנו עד שנגיע לכמות שאנו רוצים – 1000 משתמשים מדומים. בנוסף, לצורך הדימוי של גלישת משתמשים באתר נבצע HTTP GET request עבור כל משתמש מדומה כזה בצורה הבאה:



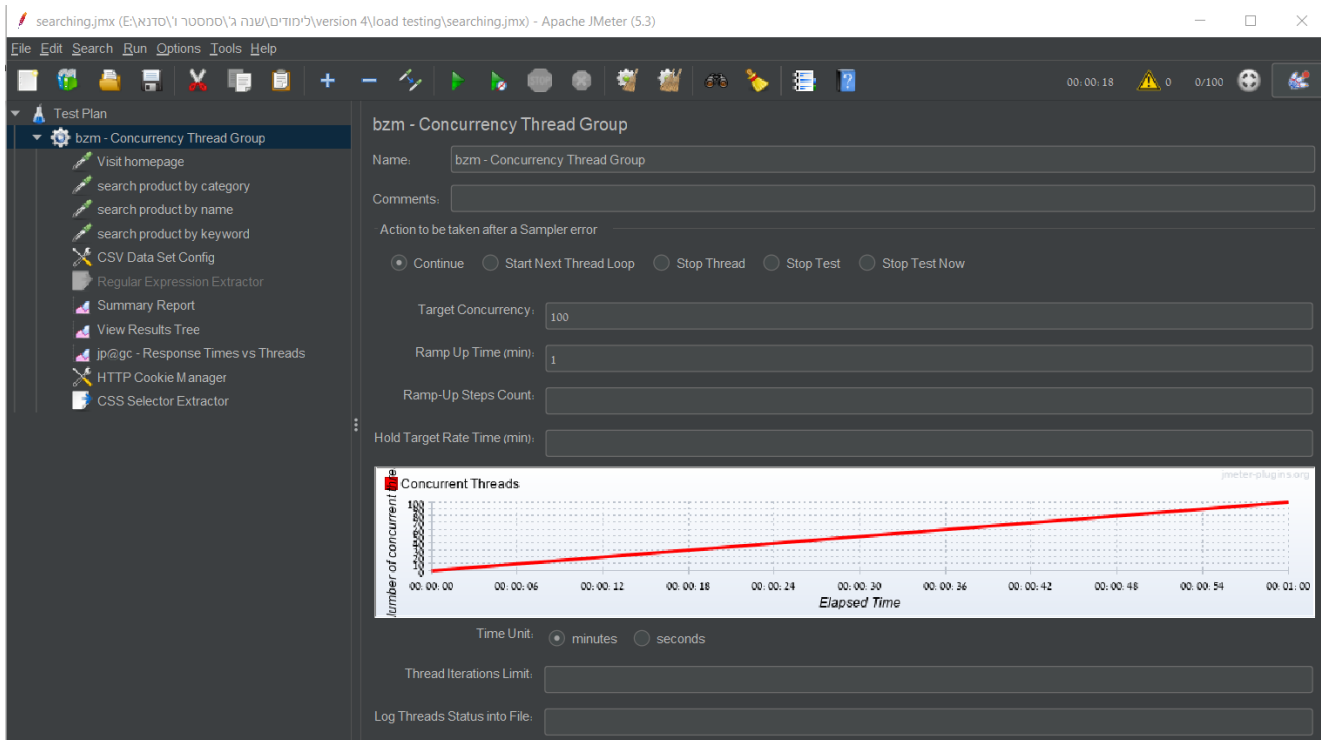
כמו כן בעזרת ה Listeners ננתח את התוצאות בצורות שונות בהמשך.

תרחיש 3 – התחברות למערכת

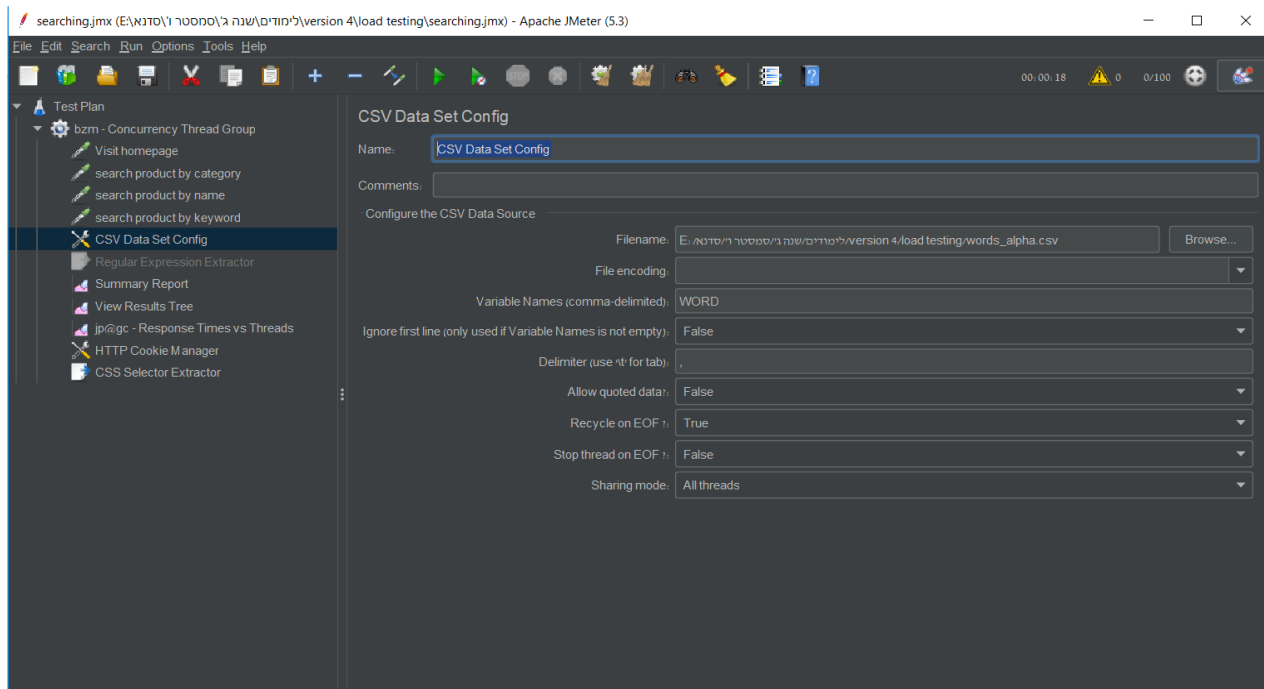
בדומה לתרחיש 1, תרחיש זה כולל כמות משתמשים המנסים להתחבר למערכת, תהליך ההתחברות מתבצע בו זמנית עבור כל המשתמשים. תחילה נתחיל במשתמש אחד שמנסה לבצע התחברות ונראה כיצד המערכת מגיבה. במהלך דקה שלמה נגדיל את מספר המשתמשים המנסים לבצע התחברות בו זמנית עד שנגיע לעומס צפוי של 100 משתמשים שמנסים לבצע התחברות בו זמנית. סוג המשתמשים שמבצעים את ההתחברות הם משתמשים אורחים שברצונם להתחבר למשתמש שלהם. תכנון הבדיקה יעשה באותה צורה של התכנון של תרחיש 1 כך ש נגדיר HTTP GET request לעמוד ההתחברות ו POST request בכדי להכניס את נתוני ההתחברות. נתוני ההתחברות יילקחו מקובץ csv המכיל פרטי משתמשים והסיסמאות שלהם.

תרחיש 4 – חיפוש במערכת

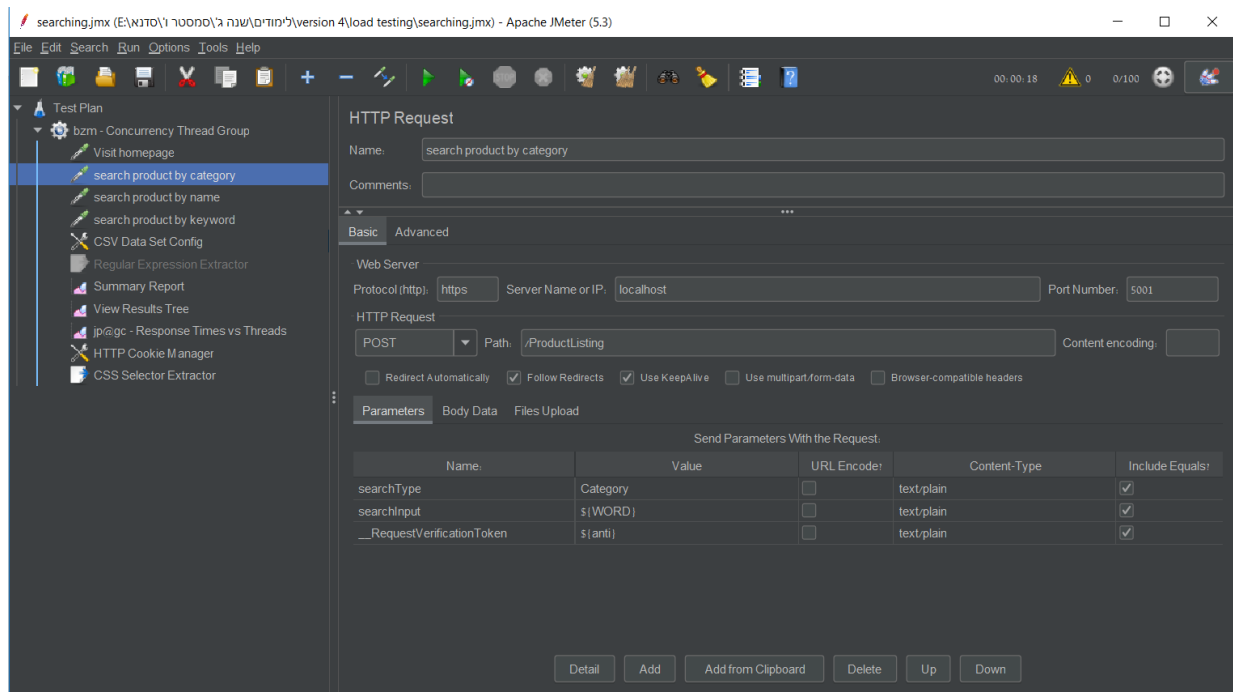
תרחיש זה כולל חיפוש לפי קטגוריה, שם ומילות מפתח. נגדיר משתמשים מדומים שיעלו בצורה לינארית, כלומר תחילה נתחיל עם כמות קטנה של משתמשים מדומים ובמהלך דקה שלמה כמות המשתמשים יעלו בצורה לינארית עד ל 100 משתמשים שמבצעים חיפוש בו זמנית. הגדרת המשתמשים המדומים תתבצע בצורה הבאה :



נשים לב כי המטרה שלנו היא 100 משתמשים מאחר ואנו רוצים שהמערכת תתמוך בפעולות שמתבצעות על ידי 100 משתמשים במקביל. לצורך החיפוש נטען מילון בקובץ csv שממנו יילקחו המילות חיפוש. נגדיר CSV Data Set Config שבאמצעותו נחלץ את המילים מהמילון ששמור לנו במחשב:



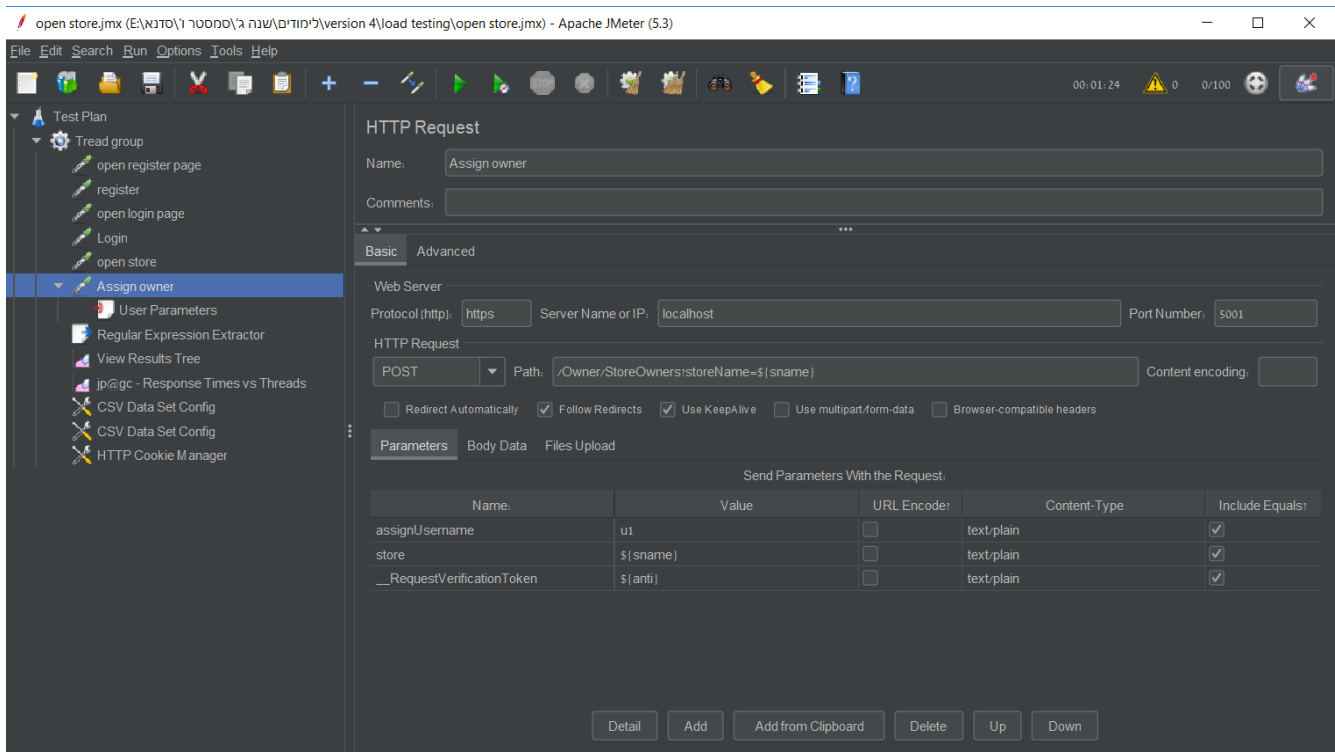
ולכל בקשת חיפוש ניצור HTTP POST request שהפרמטרים בבקשה זו יהיו סוג החיפוש ומילת החיפוש שנחליץ מהמילון :



בדוגמא זו ניתן לראות בקשה לחיפוש לפי קטגוריה, כאשר נשלפת מילה מהמילון לתוך הפרמטר searchInput.

תרחיש 5 – פתיחת חנות ומינוי בעל חנות נוסף

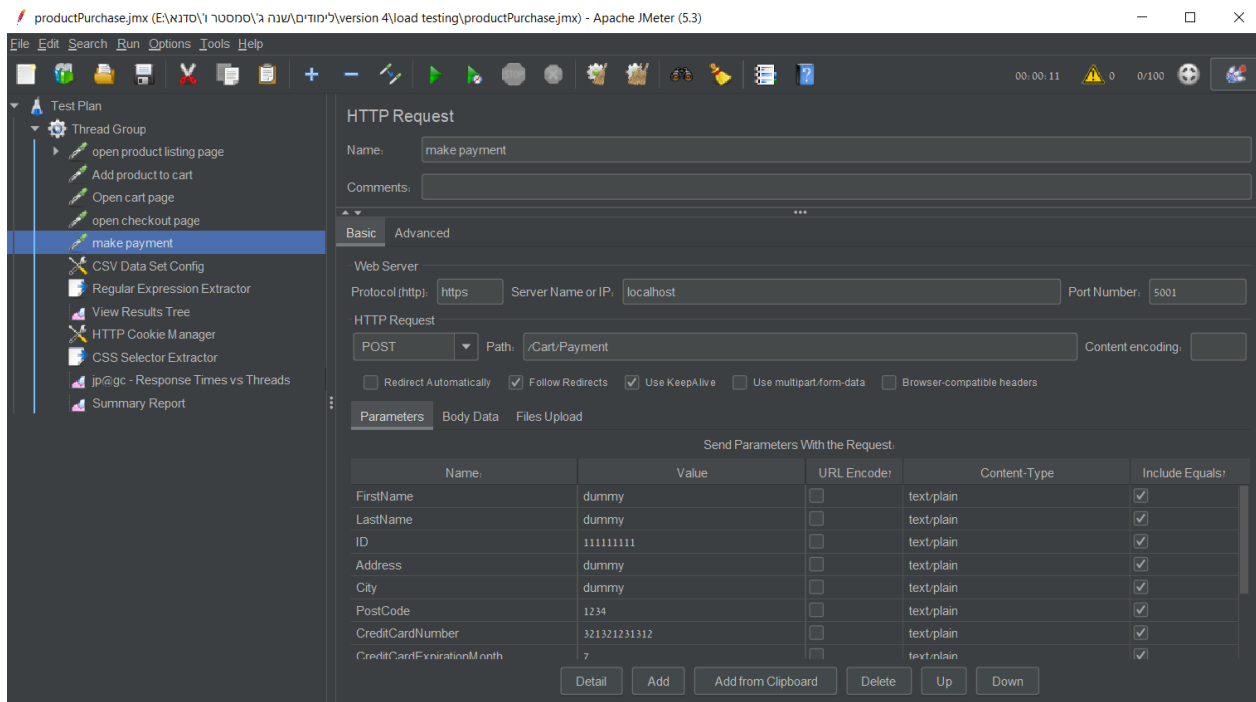
בתרחיש זה נסמלץ מצב בו כמות גדולה של משתמשים מנסים לפתוח חנות במערכת. נשים לב כי לצורך פתיחת חנות יש צורך תחילה להירשם ולהתחבר ולכן נבצע סימולציה של כל התהליך פתיחת החנות. לאחר פתיחת החנות כל אחד מהמשתמשים המדומים ימנה לבעל חנות משתמש ספציפי כדי לסמלץ מצב בו יש לנו פי 2 בעלי חנות מכמות החנויות שנפתחו. לצורך ביצוע סימולציה זו נבצע HTTP POST requests עבור הרשמה, חיבור, פתיחת חנות ומינוי בעל חנות נוסף. הבקשות לחיבור והרשמה יתבצעו כפי שהתבצעו בתרחישים הקודמים. לצורך פתיחת החנות ניקח כפרמטר לשם החנות מילה מהמילון שנמצא בקובץ ה csv ועבור מינוי בעל חנות נמנה משתמש אקראי u1 וניקח את שם החנות מהמילון בעזרת הוספה של pre processor בצורה הבאה:



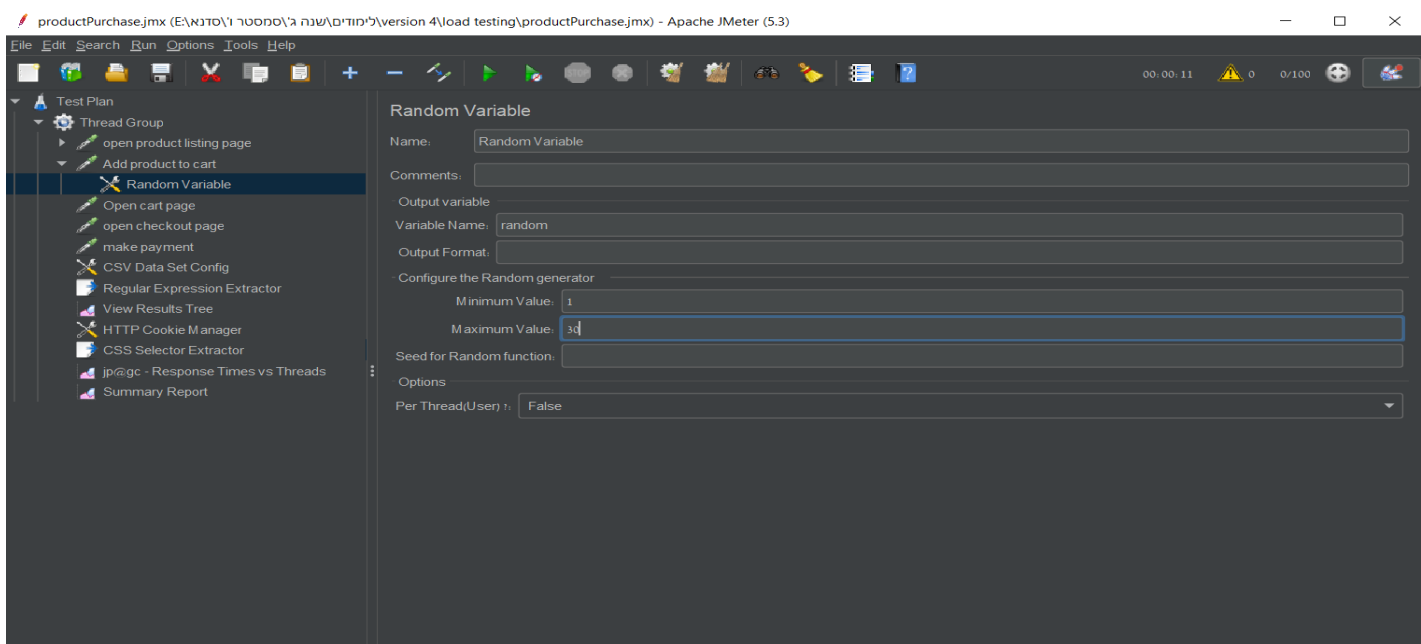
נשים לב שהpath שנקבע לצורך מינוי בעל חנות נקבע על ידי המילה הנשלפת מהמילון(שם החנות). שאר הדברים כמו session token נקבעים באותה צורה שנקבעו בתרחישים הקודמים. סימולציה זאת נבצע באמצעות 100 משתמשים מדומים שיפתחו חנויות אקראיות במערכת שלנו. בנוסף נציין כי יצרנו טיימר של חצי שנייה בין בקשה לבקשה בכדי לדמות מצב אמיתי ולא מצב בו כל התהליך מתבצע בו זמנית(בפועל לוקח יותר מחצי שנייה בין בקשה לבקשה) .

תרחיש 6-רכישה

תהליך הרכישה הוא תהליך שעלול להתבצע על ידי משתמשים מרובים בו זמנית כאשר יהיו הנחות גדולות או בתקופות מסוימות בהן יש דרישה גבוה למוצרים מסוימים. לצורך כך נבצע סימולציה של רכישה של מוצר מסוים שיש לו ביקוש גבוה, נבצע את הרכישה על ידי 100 משתמשים מדומים במקביל. לצורך ביצוע של בדיקה זו נבצע הדמיה של כל תהליך הרכישה שכולל: פתיחה של עמוד המוצרים, הוספת מוצר לעגלה, פתיחת העגלה, ביצוע checkout ורכישה באמצעות הכנסת פרטי התשלום. לצורך כך נקבע מוצר בעל prodID ספציפי שאותו נרצה לרכוש שקיים במלאי ונבצע את הסימולציה שברצוננו לבצע. כמו כן נגדיר טיימר בין כל בקשה לשרת כדי לדמות מצב אמיתי. פרטי התשלום שנכניס יכנסו בצורה הבאה באמצעות HTTP POST request :



הכמות שנוסיף לעגלה תהיה כמות רנדומלית באמצעות random variable שנגדיר בצורה הבאה:



3.1. תרחישי בדיקות עומס:

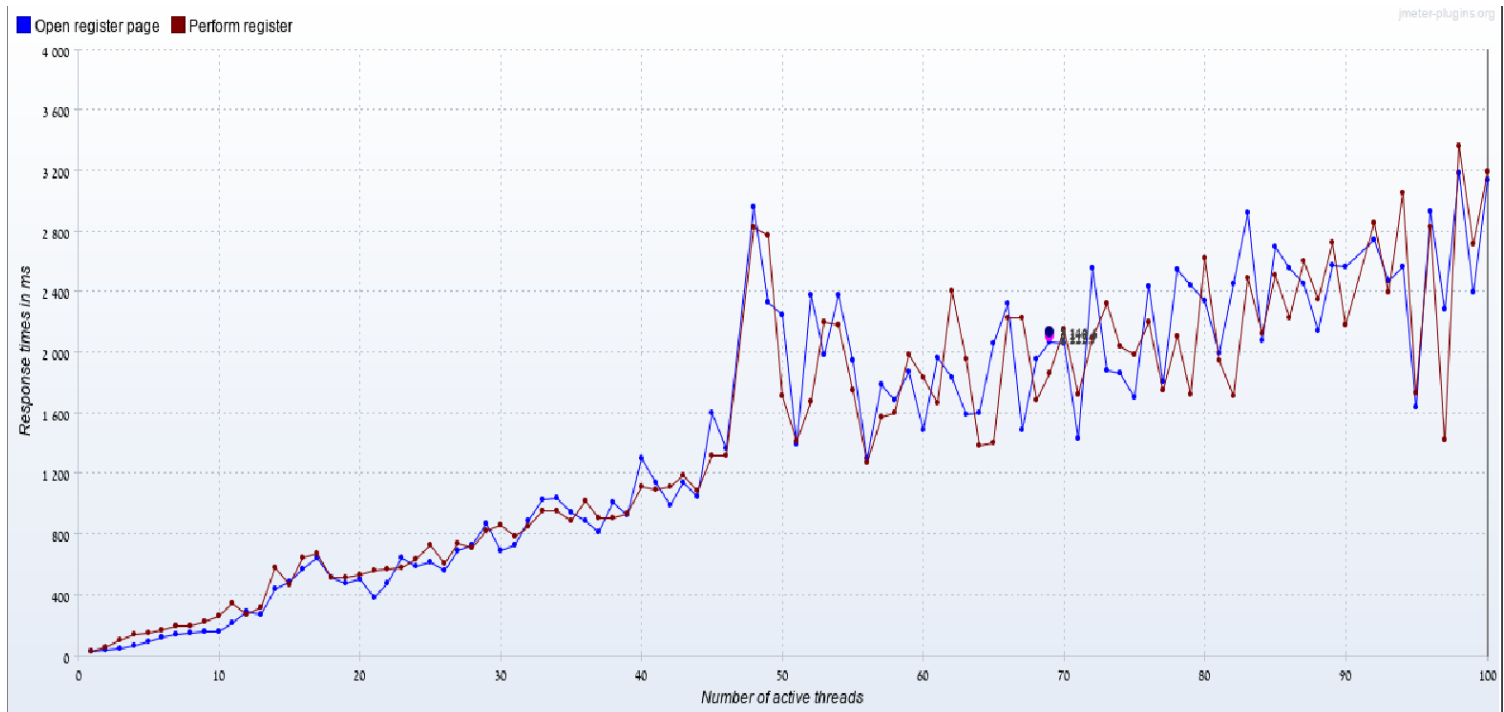
בדיקות העומס ייקבעו בהתאם לתוצאות של בדיקות הקיבול ובהתאם למדדים שהגדרנו, כלומר: הגדרנו כי בכל רגע נתון יהיו 1000 משתמשים במערכת ולכן לצורך בדיקת העומס נסמלץ מספר גדול יותר של משתמשים שמשתמשים במערכת בניסיון להקריס את המערכת שלנו ולראות כיצד היא מתאוששת משגיאות. ננתח את התוצאות בהמשך.

3.2. ניתוח תוצאות בדיקות הקיבול:

תרחיש 1 – רישום למערכת

בעת ביצוע בדיקה לרישום במערכת התוצאות שקיבלנו הן התוצאות הבאות:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Open register page	1712	1430	21	7433	1203.05	0.00%	17.2/sec	160.15	5.28	9526.6
Perform register	1668	1430	28	8979	1044.01	0.00%	16.8/sec	160.54	11.66	9794.1
TOTAL	3380	1430	21	8979	1127.37	0.00%	34.0/sec	320.56	16.93	9658.6

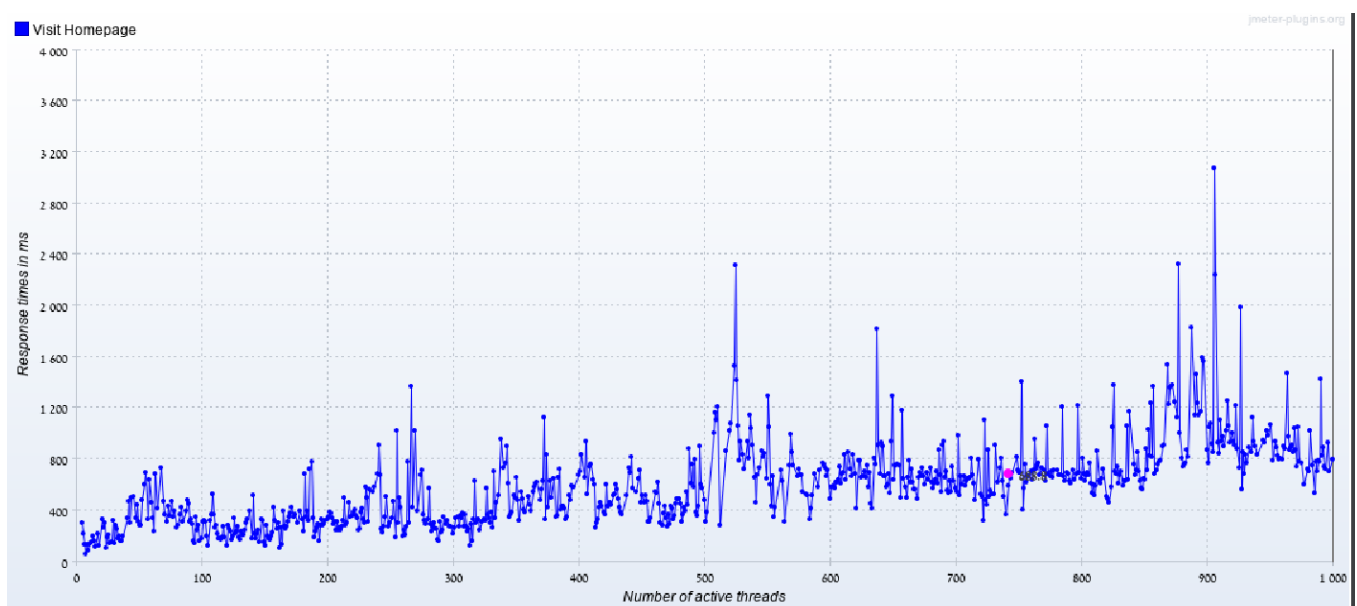
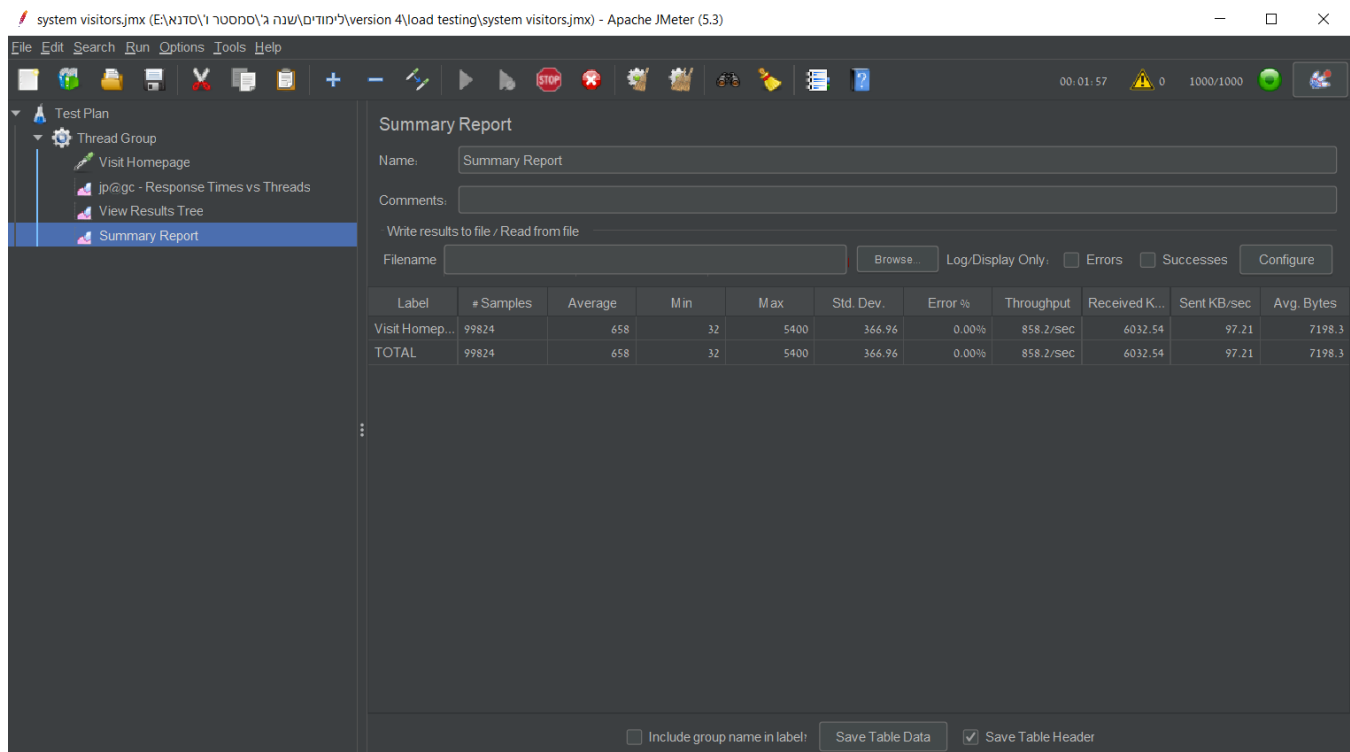


נשים לב לפי הטבלה כי יש 0% שגיאות, כלומר כל בקשה שהתקבלה בשרת טופלה והמערכת לא קרסה כתוצאה מפעילות זו. כמו כן זמן התגובה של השרת לא חרג 5 שניות גם עבור כמות גדולה מאוד של משתמשים וניתן לראות על פי הגרף כי זמן התגובה הגבוה ביותר של השרת היה 3.5 שניות בערך עבור 95 משתמשים מדומים שרצו במקביל. ניתן לראות גם על פי הטבלה כי זמן התגובה הממוצע הוא כ-1.4 שניות וכי השרת מסוגל לטפל ב-34 בקשות מסוג זה לשנייה. נשים לב כי הגרף המוצג הוא זמן תגובת השרת כפונקציה של מספר המשתמשים שרצים במקביל ולכן ככל שמספר

המשתמשים עולה הגרף נמצא במגמת עלייה כצפוי. על פי תוצאות אלו ניתן לראות כי בבדיקה זו המערכת עמדה במדדים שהוגדרו לה מכיוון שעמדה ב 100 בקשות רישום במקביל וזמן התגובה של השרת לא עלה על 5 שניות. ניתן לראות בנוסף כי המערכת פעלה 100% מהזמן.

תרחיש 2 – ביקור במערכת

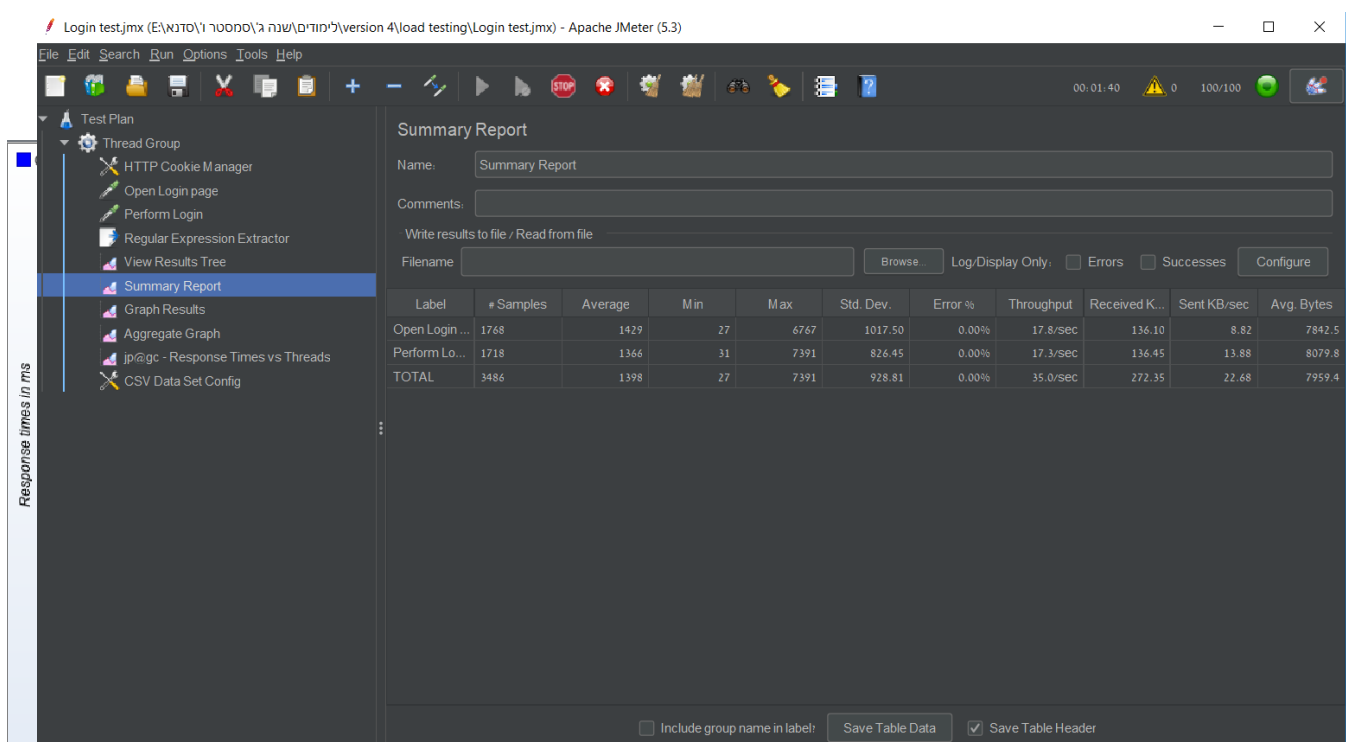
בעת ביצוע בדיקה לרישום במערכת התוצאות שקיבלנו הן התוצאות הבאות:



ניתן לראות מהתמונה הראשונה כי רצים ברקע 1000 משתמשים מדומים, הטבלה מציגה כי יש 0% שגיאות והמערכת לא קרסה ורצה כרגיל. על פי הגרף ניתן לראות כי בקשות ה GET לביקור בעמוד הראשי של האתר התקבלו בזמן התגובה של השרת לא עלה על 5 שניות ובפועל הזמן הארוך ביותר שלקח לשרת להגיב הוא כ 3.2 שניות. זמן התגובה הממוצע של השרת הוא כ 0.6 שניות. כמו כן ניתן לראות כי הגרף נמצא במגמת עלייה אך לא עלייה חדה כמו בתרחיש הקודם. על פי תוצאות אלו ניתן לראות כי בבדיקה זו המערכת עמדה במדדים שהוגדרו לה מכיוון שעמדה ב 1000 משתמשים מבקרים במקביל וזמן התגובה של השרת לא עלה על 5 שניות. ניתן לראות בנוסף כי המערכת פעלה 100% מהזמן.

תרחיש 3 – התחברות למערכת

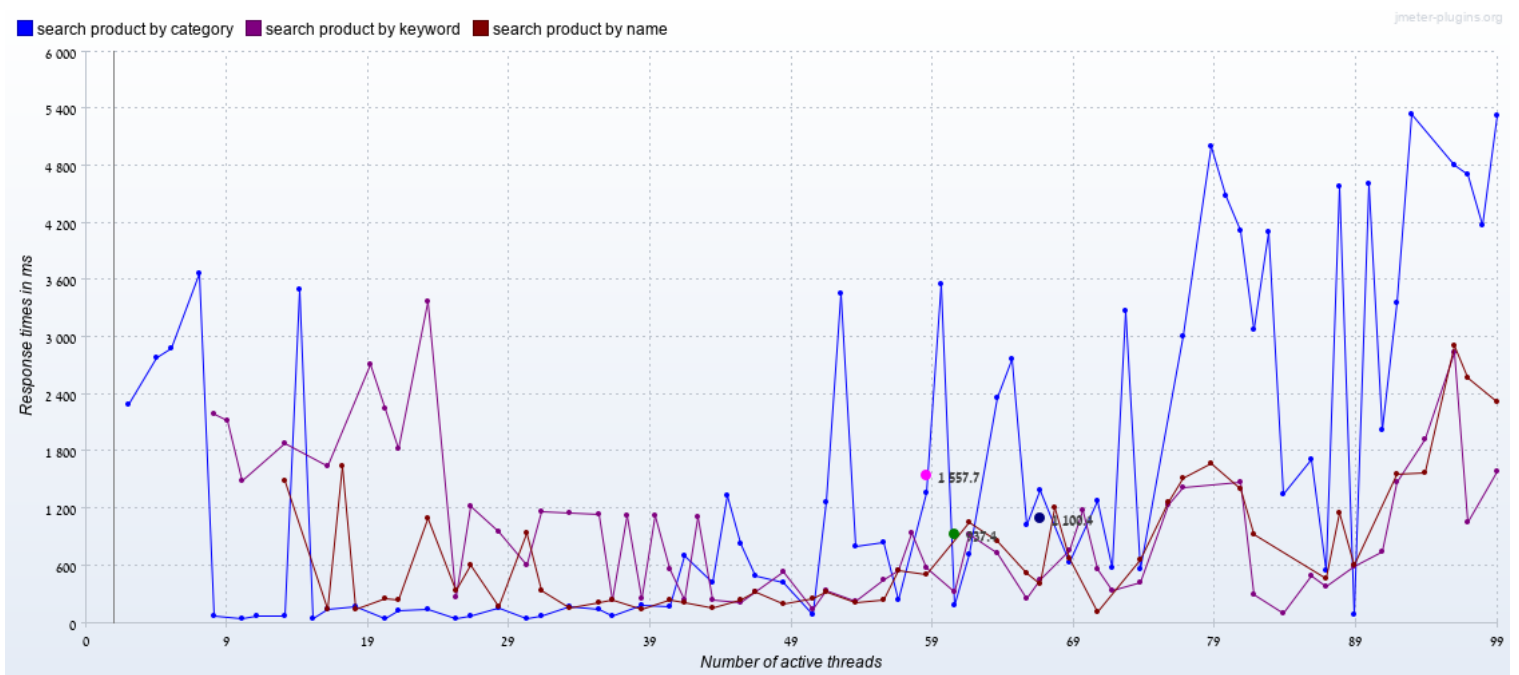
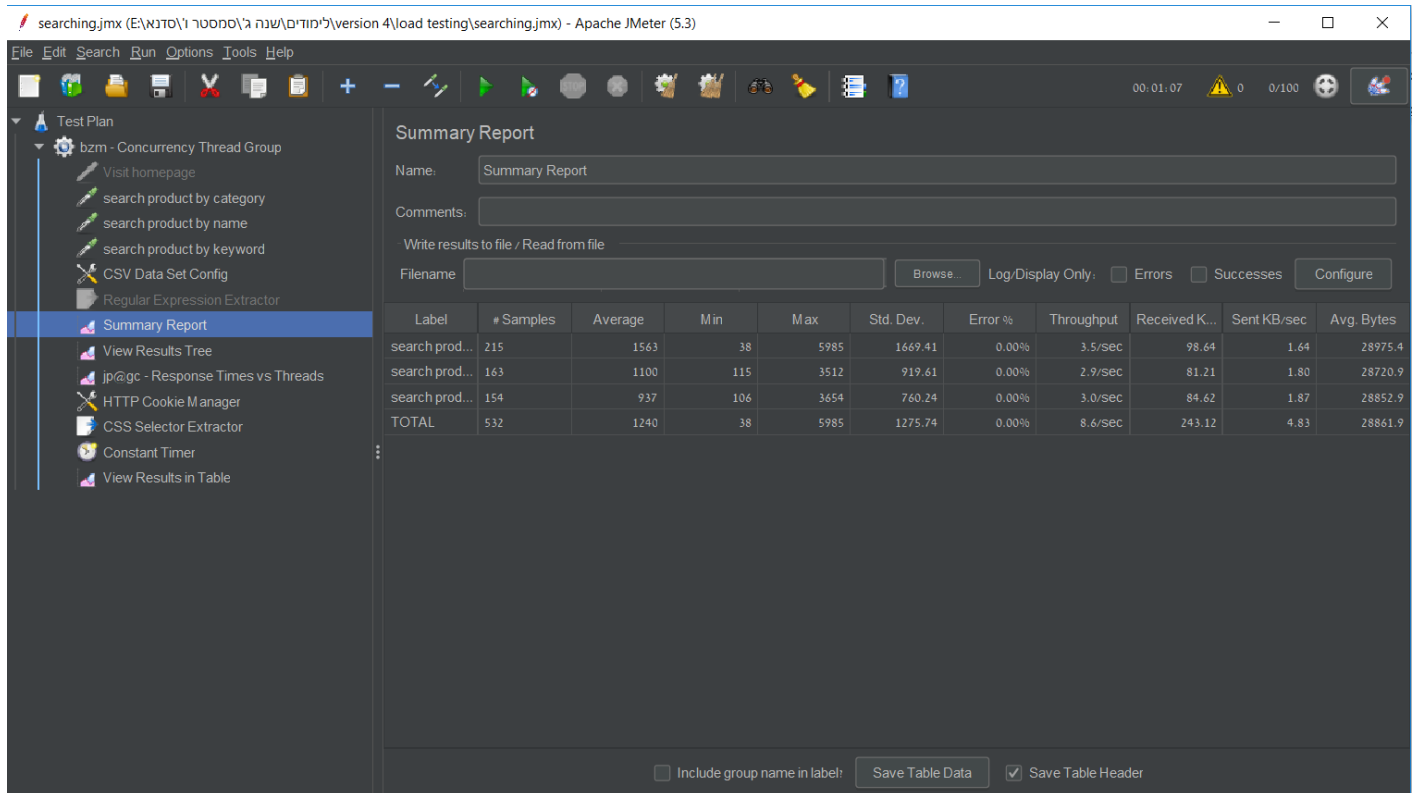
בעת ביצוע בדיקה לרישום במערכת התוצאות שקיבלנו הן התוצאות הבאות:



נשים לב לפי הטבלה כי יש 0% שגיאות, כלומר כל בקשה שהתקבלה בשרת טופלה והמערכת לא קרסה כתוצאה מפעילות זו. כמו כן זמן התגובה של השרת לא חרג 5 שניות גם עבור כמות גדולה מאוד של משתמשים וניתן לראות על פי הגרף כי זמן התגובה הגבוה ביותר של השרת היה כ 3 שניות בערך עבור 100 משתמשים מדומים שרצו במקביל. ניתן לראות גם על פי הטבלה כי זמן התגובה הממוצע הוא כ 1.4 שניות וכי השרת מסוגל לטפל ב 35 בקשות מסוג זה לשנייה. נשים לב כי הגרף המוצג הוא זמן תגובת השרת כפונקציה של מספר המשתמשים שרצים במקביל ולכן ככל שמספר המשתמשים עולה הגרף נמצא במגמת עלייה כצפוי. על פי תוצאות אלו ניתן לראות כי בבדיקה זו המערכת עמדה במדדים שהוגדרו לה מכיוון שעמדה ב 100 בקשות התחברות במקביל וזמן התגובה של השרת לא עלה על 5 שניות. ניתן לראות בנוסף כי המערכת פעלה 100% מהזמן.

תרחיש 4 – חיפוש במערכת

בעת ביצוע בדיקה לפתיחת חנות ומינוי בעל חנות במערכת התוצאות שקיבלנו הן התוצאות הבאות:

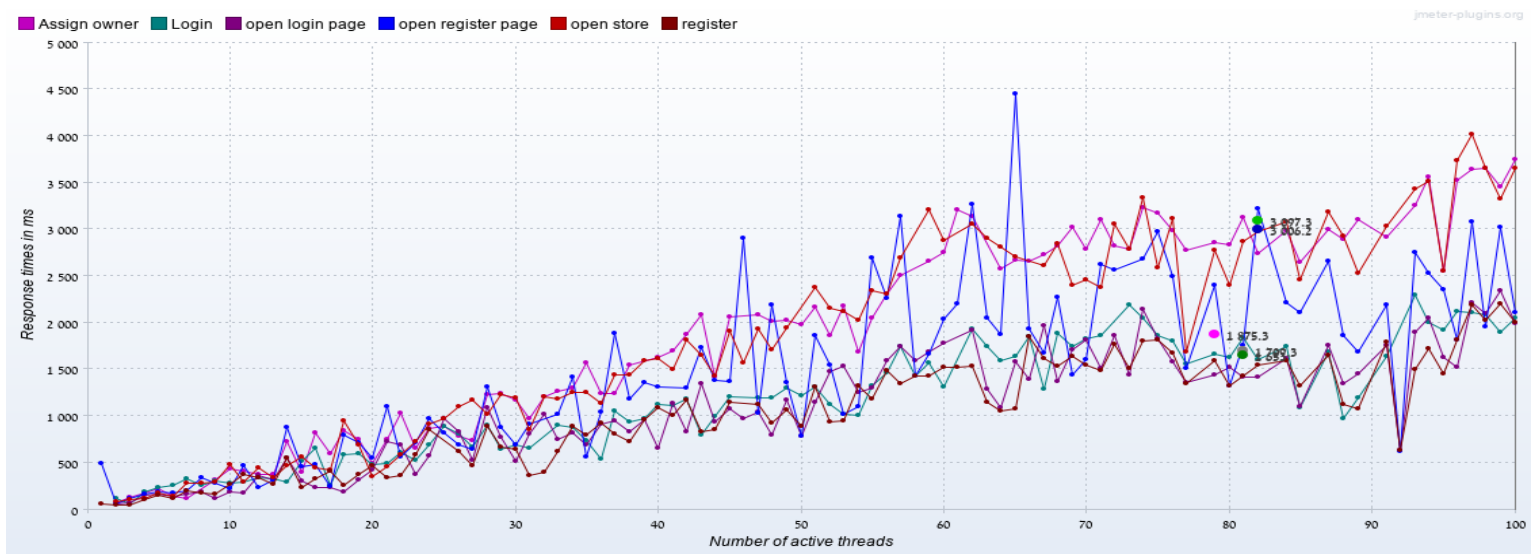
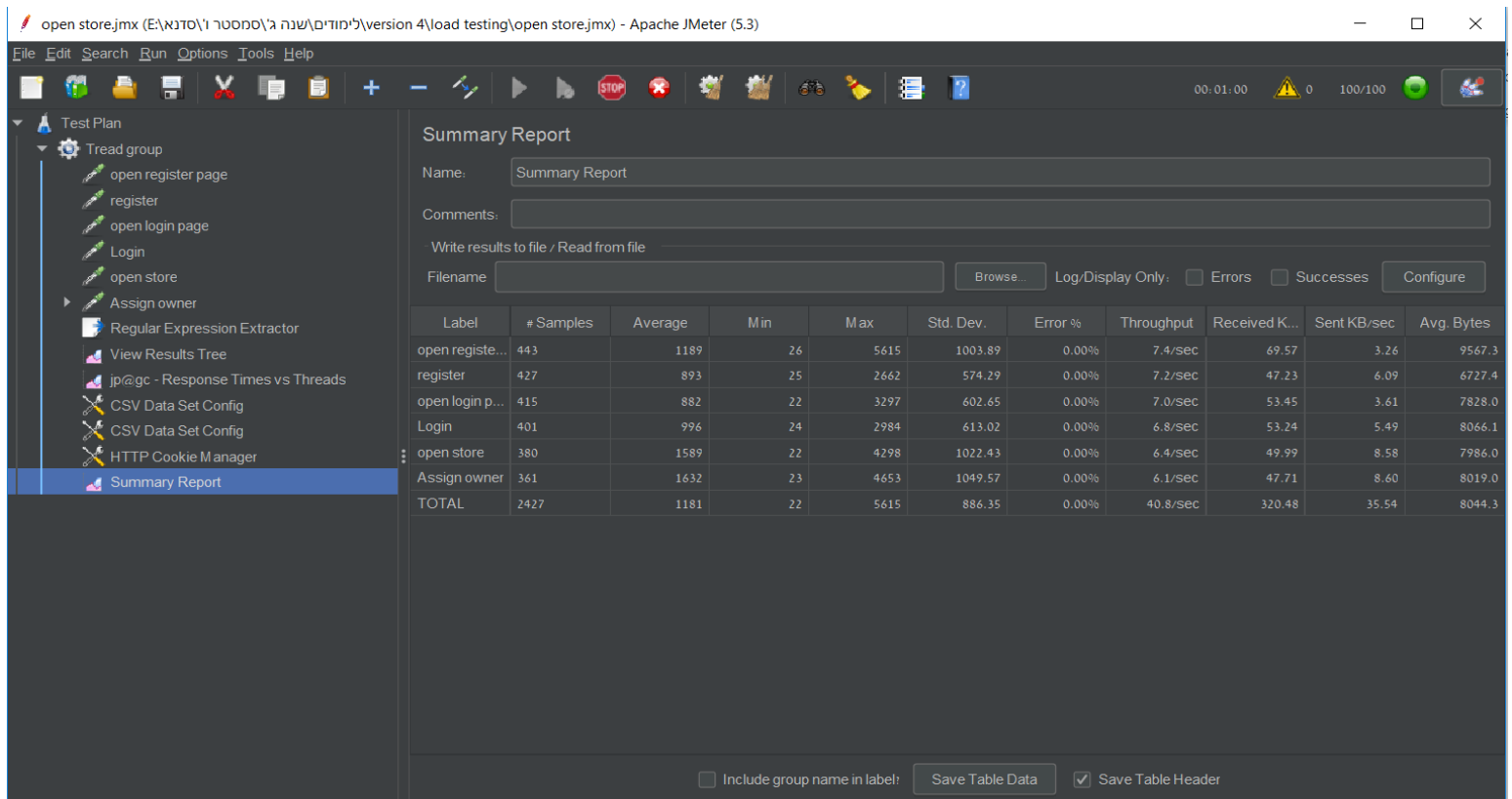


על פי הטבלה קיבלנו 0% שגיאות אם כי בזמן התגובה של השרת חרגנו במקצת מ-5 השניות עבור כ-94 משתמשים מדומים והגענו לזמן תגובה של כ-5.3 שניות. עדיין ניתן לראות כי זמן התגובה הממוצע של

השרת הוא כ 1.2 שניות. נשים לב כי התפוקה של השרת בסימולציה זו נמוכה יותר מהתפוקה של השרת בתרחישים קודמים, מכיוון שמדובר בפעולה קשה יותר לביצוע (פעולת החיפוש כוללת תיקון שגיאות וכו'). המערכת שלנו עמדה בעומס הצפוי וקיבלנו 0% שגיאות, אם כי טיפה חרגה מזמן של 5 שניות לזמן של 5.3 שניות לתגובה.

תרחיש 5- פתיחת חנות ומינוי בעל חנות נוסף

בעת ביצוע בדיקה לפתיחת חנות ומינוי בעל חנות במערכת התוצאות שקיבלנו הן התוצאות הבאות:



על פי התוצאות שקיבלנו ניתן לראות כי יש 0% שגיאות וכי השרת לא קרס, בגרף שקיבלנו הוא גרף מורכב שכולל ניתוח של המון פעולות שקרו במערכת אך ניתן לראות כי הזמן הגבוה ביותר לתגובה של השרת הוא כ 4.5 שניות עבור 65 משתמשים מדומים. בגרף נמצא בעלייה מתמדת בכל שכמות המשתמשים עולה, כלומר זמן התגובה של השרת עולה ביחד עם העלייה במספר המשתמשים. על פי תוצאות אלו ניתן לראות כי בבדיקה זו המערכת עמדה במדדים שהוגדרו לה מכיוון שעמדה ב 100 משתמשים במקביל וזמן התגובה של השרת לא עלה על 5 שניות. ניתן לראות בנוסף כי המערכת פעלה 100% מהזמן.

תרחיש 6- רכישה

בעת ביצוע בדיקה לפתיחת חנות ומינוי בעל חנות במערכת התוצאות שקיבלנו הן התוצאות הבאות:

productPurchase.jmx (E:\סדנא\ו'סמסטר\שנה ג'לימודים\version 4\load testing\productPurchase.jmx) - Apache JMeter (5.3)

File Edit Search Run Options Tools Help

Test Plan

- Thread Group
 - open product listing page
 - Add product to cart
 - Open cart page
 - open checkout page
 - make payment
 - CSV Data Set Config
 - Regular Expression Extractor
 - View Results Tree
 - HTTP Cookie Manager
 - jp@gc - Response Times vs Threads
 - Summary Report**
 - Constant Timer

Summary Report

Name: Summary Report

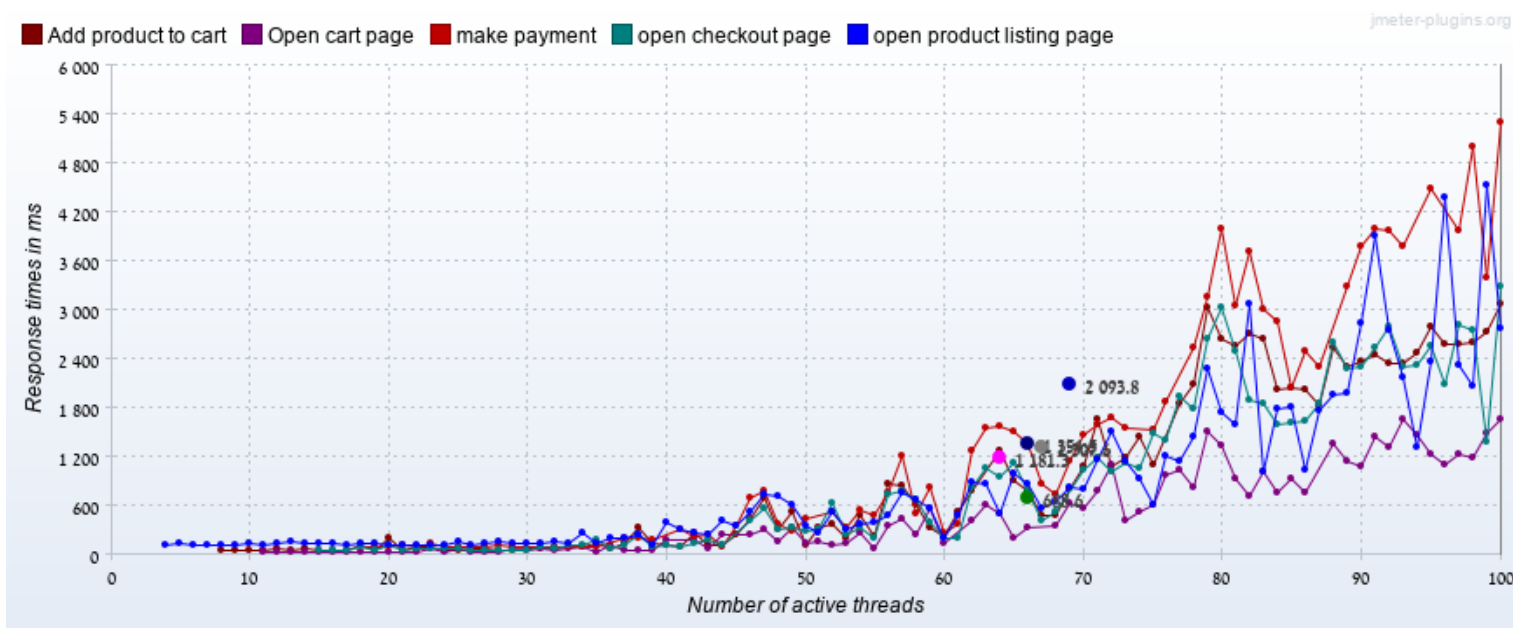
Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
open produc...	286	1291	107	5725	1240.90	0.00%	2.6/sec	75.25	1.02	29708.1
Add product...	262	1529	52	6205	1375.77	0.00%	2.5/sec	23.15	3.39	9673.0
Open cart p...	250	805	28	3220	703.00	0.00%	2.4/sec	21.97	1.27	9300.0
open check...	222	1551	55	8266	1451.03	0.00%	2.2/sec	21.00	2.37	9670.0
make paym...	194	2316	85	8434	2038.37	0.00%	2.0/sec	19.38	4.18	9813.0
TOTAL	1214	1454	28	8434	1461.72	0.00%	11.0/sec	153.79	11.25	14338.0

☐ Include group name in label: ☒ Save Table Header

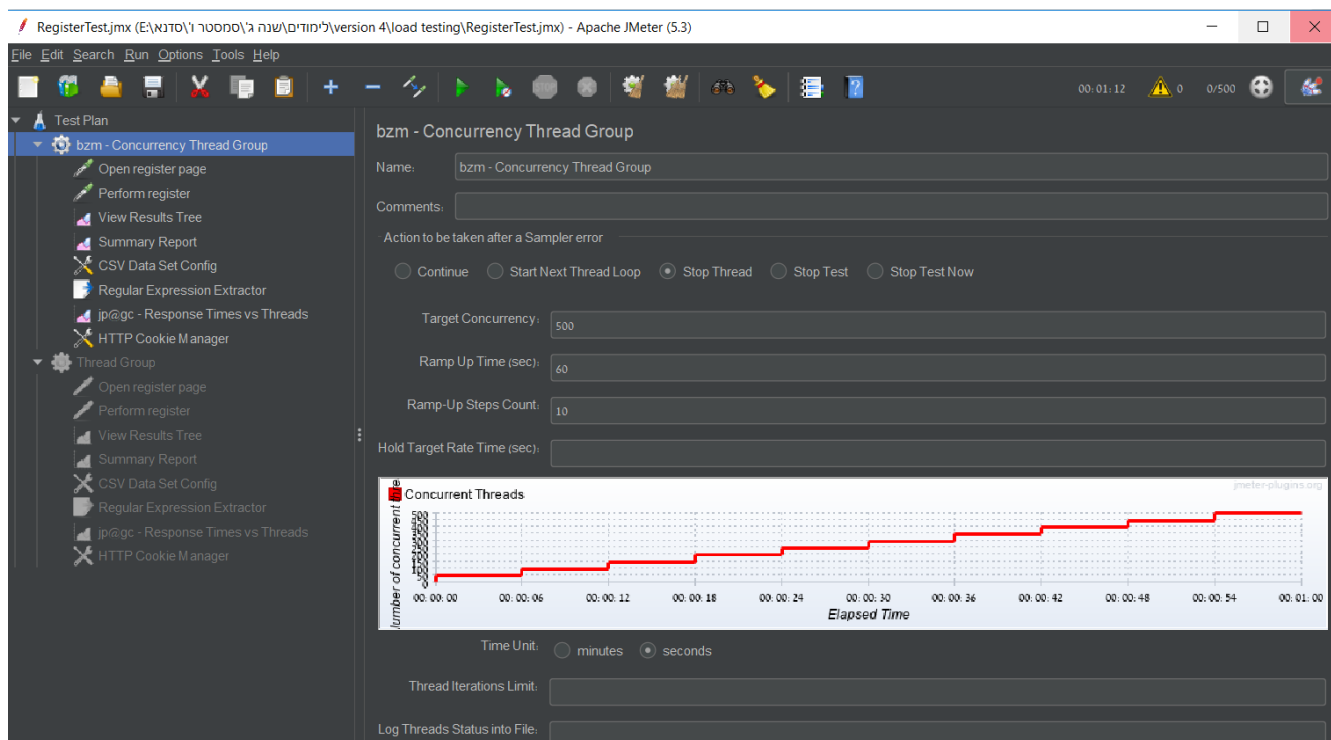


נשים לב כי בזמן ממוצע, בקשת ה HTTP לביצוע תשלום גזלה הכי הרבה זמן תגובה מהשרת. כמו כן גם כאן קיבלנו 0% שגיאות, עם תפוקת שרת דומה לתפוקה שקיבלנו בתרחיש 4. הגרף נמצא העלייה מתמדת עם עליית המשתמשים המדומים ובסופו של דבר כאשר אנו מגיעים ל 100 משתמשים, זמן התגובה של השרת הוא כ 5 שניות לביצוע פעולת התשלום. ניתן להבין כי פעולות התשלום והחיפוש הן פעולות שגוזלות זמן גדול יותר מהשרת לתגובה. המערכת שלנו עמדה בעומס זה וגם כאן קיבלנו 0% שגיאות וזמן תגובה של השרת כפי שהובטח שמגיע למקסימום 5 שניות עבור 100 משתמשים.

3.3. ניתוח תוצאות בדיקות העומס:

תרחיש 1 – רישום למערכת

בכדי להביא את המערכת למצב קיצון יצרנו Thread group בצורה הבאה:



כך שבכל 6 שניות בערך נוספים 50 משתמשים מדומים חדשים עד להגעה של 500 משתמשים שמנסים להירשם בו זמנית, בניסיון להקריס את התוכנה ולראות כיצד היא מתמודדת עם השגיאות קיבלנו את התוצאות הבאות:

The screenshot displays the Apache JMeter 5.3.1 interface showing the 'Summary Report' for the 'bzm - Concurrency Thread Group'. The report shows the number of samples, average, min, max, std. dev., error %, throughput, received, sent KB/s, and avg. bytes for each sampler.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/s...	Avg. Bytes
Open register page	1374	11121	168	32253	8084.39	6.33%	22.1/sec	198.68	5.00	9195.0
Perform register	888	1173	73	6776	1432.05	0.00%	17.5/sec	167.62	12.17	9794.1
TOTAL	2262	7215	73	32253	8006.47	3.85%	36.4/sec	335.46	14.93	9430.2

נשים לב כי קיבלנו סך הכל 3.85% שגיאות כאשר המערכת נמצאה במצב קיצוני ועומס כבד של ניסיונות רישום למערכת. יש לציין כי המערכת התמודדה עם שגיאות אלה בצורה טובה ולא קרסה, כמו כן לאחר שבקשות מסוימות לא התקבלו המערכת המשיכה לפעול ולקבל בקשות HTTP ואפילו בהצלחה.

תרחיש 3 – התחברות למערכת

בכדי להביא את המערכת למצב קיצון יצרנו Thread group של 600 משתמשים מדומים שמנסים לבצע התחברות למערכת כך שכל פעם מספר המשתמשים עולה מ 0 ועד 600. התוצאות שקיבלנו:

Login test.jmx (E:\סדנא\ו'סמסטר\שנה ג'לימודים\version 4\load testing\Login test.jmx) - Apache JMeter (5.3)

File Edit Search Run Options Tools Help

00:01:11 0 600/600

Test Plan

- Thread Group
 - HTTP Cookie Manager
 - Open Login page
 - Perform Login
 - Regular Expression Extractor
 - View Results Tree
 - Summary Report**
 - Graph Results
 - Aggregate Graph
 - jp@gc - Response Times vs Threads
 - CSV Data Set Config

Summary Report

Name: Summary Report

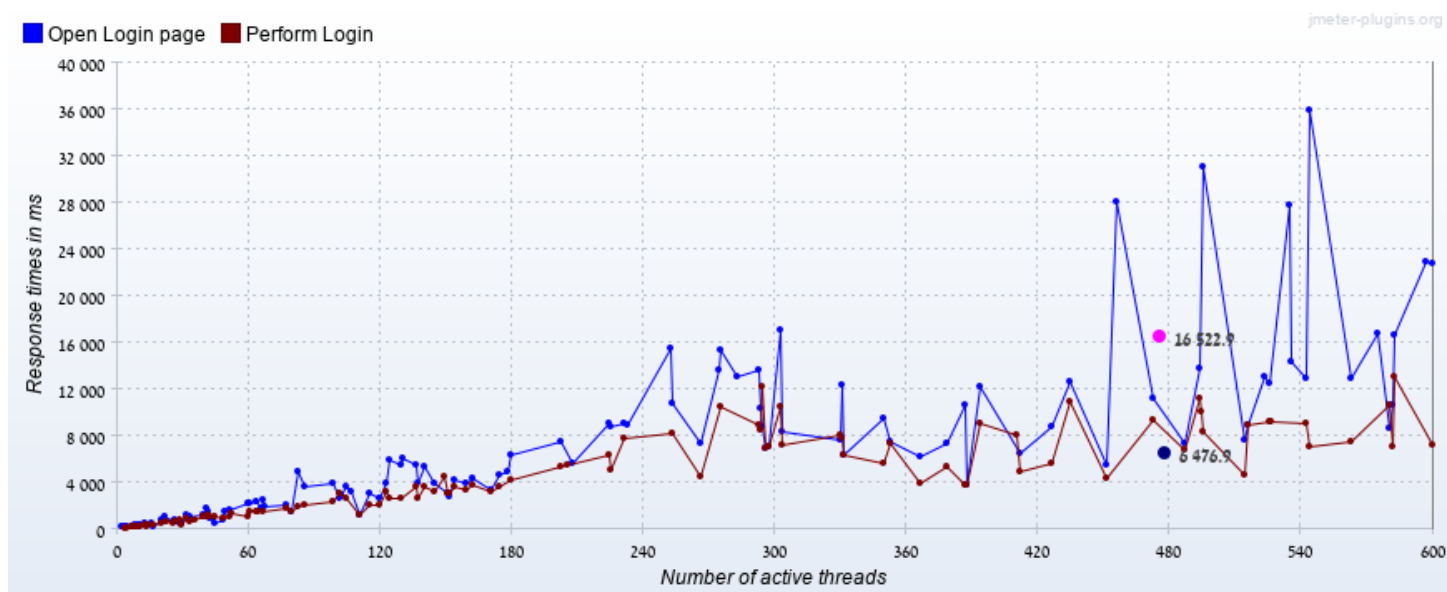
Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
Open Login ...	907	9619	104	50258	9069.83	2.87%	12.9/sec	97.73	5.05	7761.2
Perform Lo...	722	5996	122	18779	3277.96	0.14%	10.3/sec	81.98	7.82	8127.3
TOTAL	1629	8014	104	50258	7335.06	1.66%	23.2/sec	179.20	12.82	7923.5

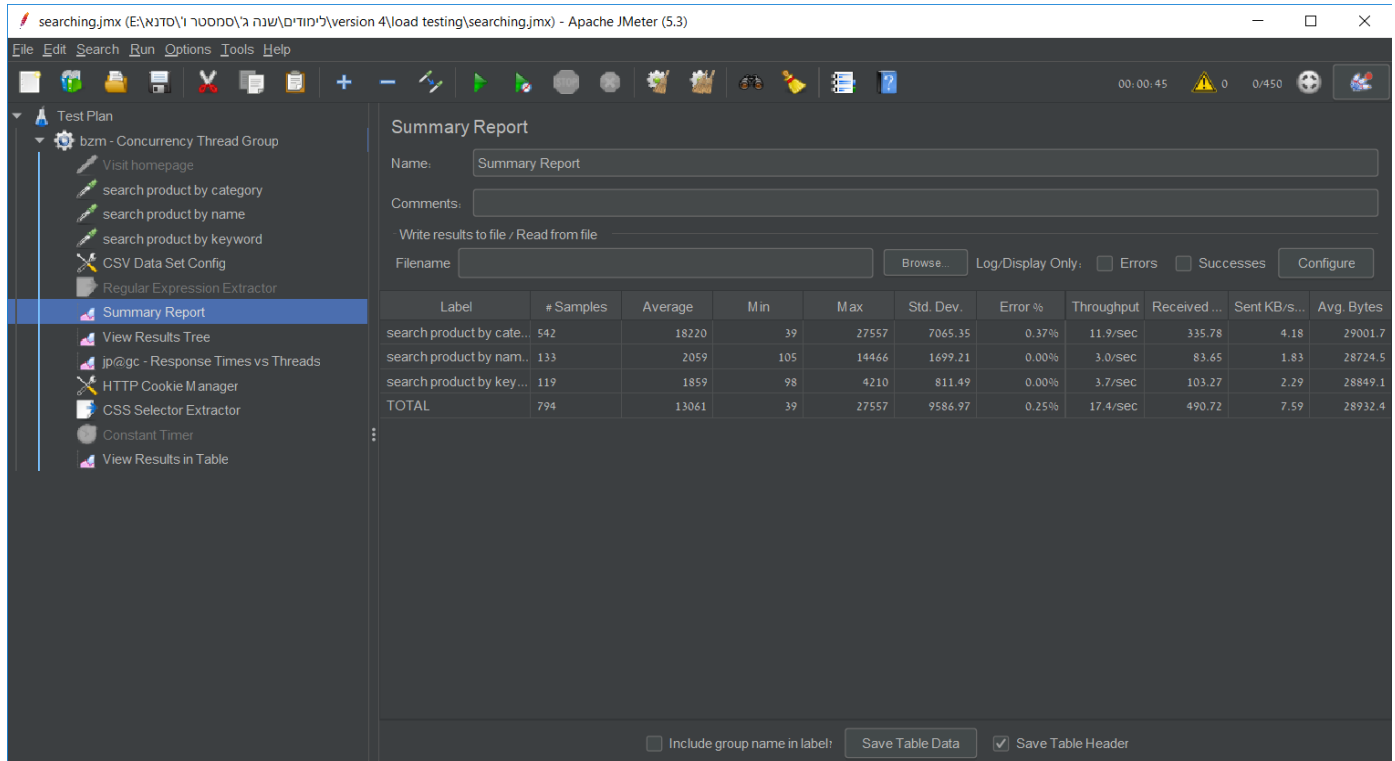
☐ Include group name in label: ☒ Save Table Header



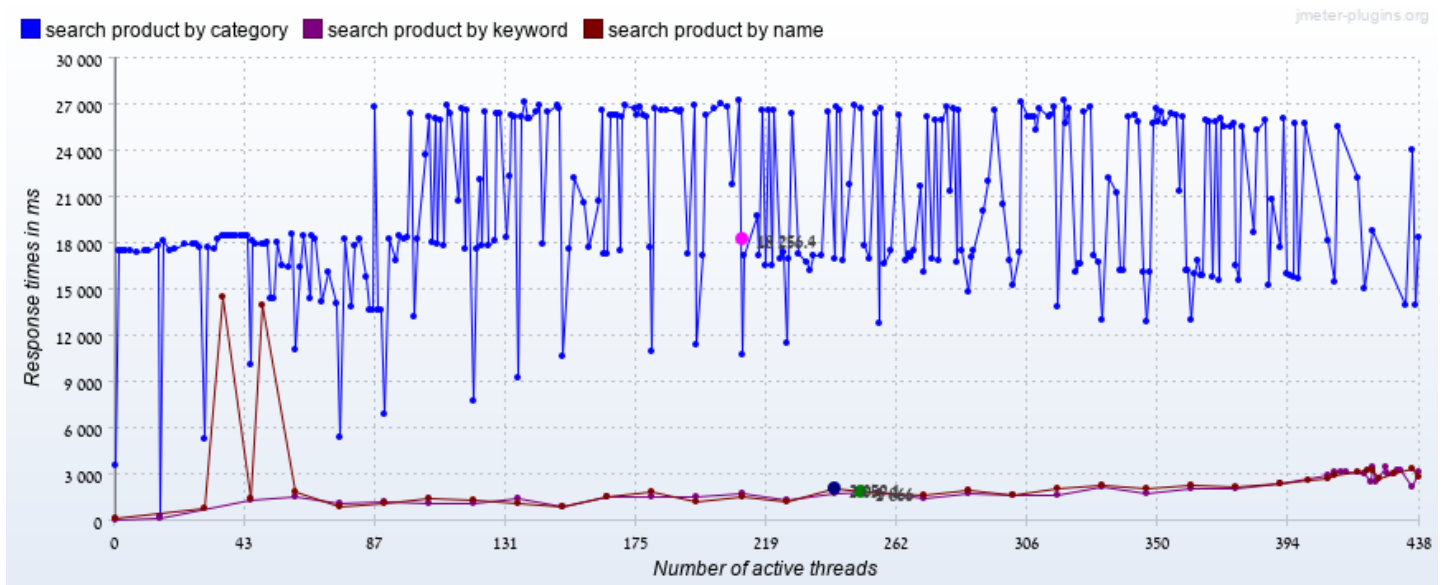
ניתן לראות כי אחוז השגיאות נמוך מאוד מכיוון שנשלחו אלפי בקשות לשרת ומתוכם רק 1.6% לא טופלו. כמו כן ניתן לראות כי זמן התגובה של השרת במצב קיצון זה גבוה מאוד. עם זאת, המערכת שלנו לא קרסה והמשיכה לפעול וגם לאחר שבקשות מסוימות לא טופלו על ידי השרת, לאחר מכן הגיעו בקשות נוספות שכן טופלו על ידי השרת שלנו.

תרחיש 4 – חיפוש במערכת

מאחר שראינו בדיקות הקיבול כי מדובר בפעולה שמאלצת את השרת להגיב בזמן גבוה יותר, נסתפק בלבצע את בדיקת העומס עבור עומס משתנה שמגיע תוך חצי דקה לעומס קיצוני של 450 משתמשים שמבצעים חיפוש במקביל. התוצאות שקיבלנו הם:

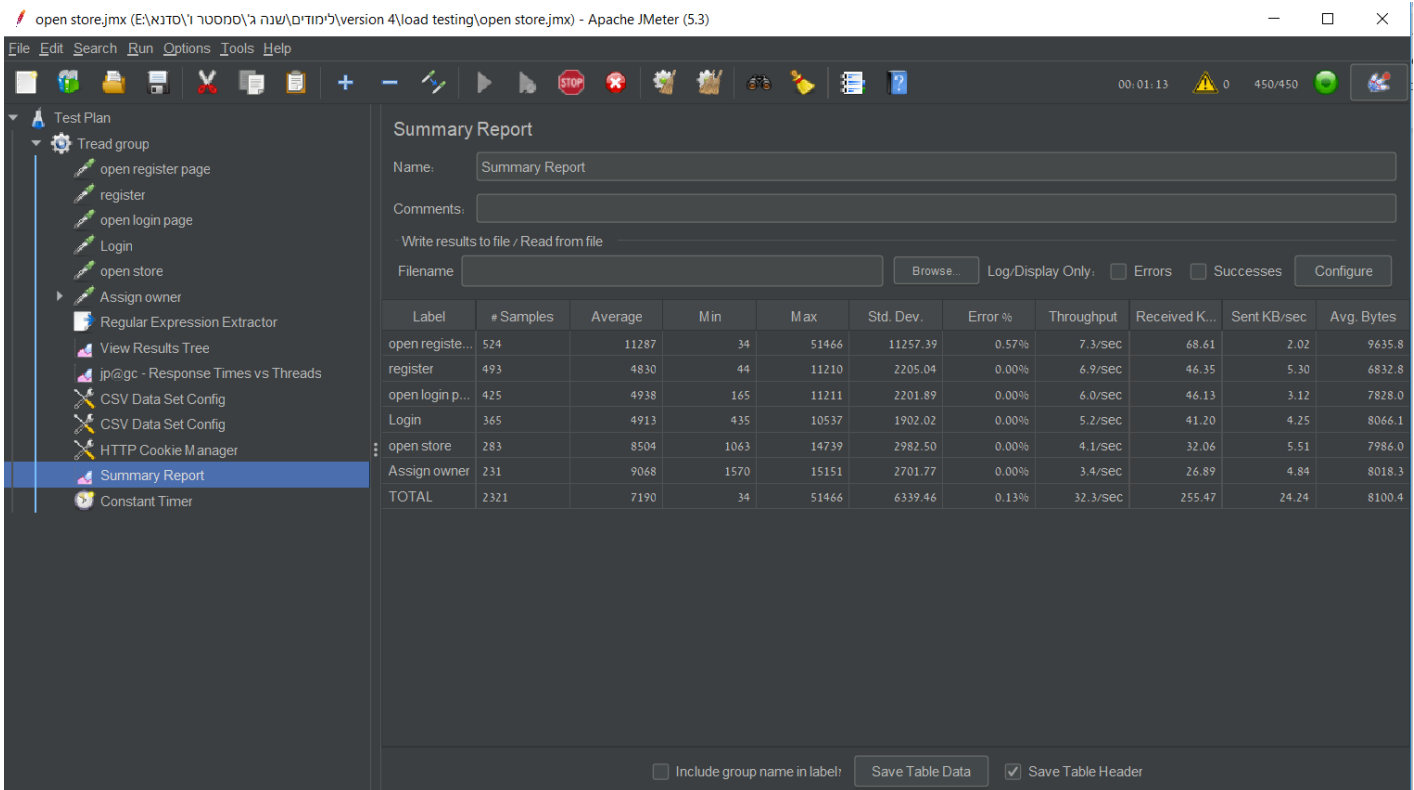


נשים לב כי קיבלנו בסה"כ 0.25% שגיאה, ויש לציין כי אחוזים אלו היו גבוהים יותר אך המערכת שלנו התאוששה מהשגיאות ולאחר השגיאות התבצעו בקשות HTTP בהצלחה דבר אשר הוריד את האחוזים של השגיאות. עם זאת, למרות אחוזי השגיאות הנמוכים יש לציין כי קיבלנו זמני תגובה חריגים וקיצוניים מהשרת עבור כמות גדולה כזאת של משתמשים, נציג זאת באמצעות הגרף:

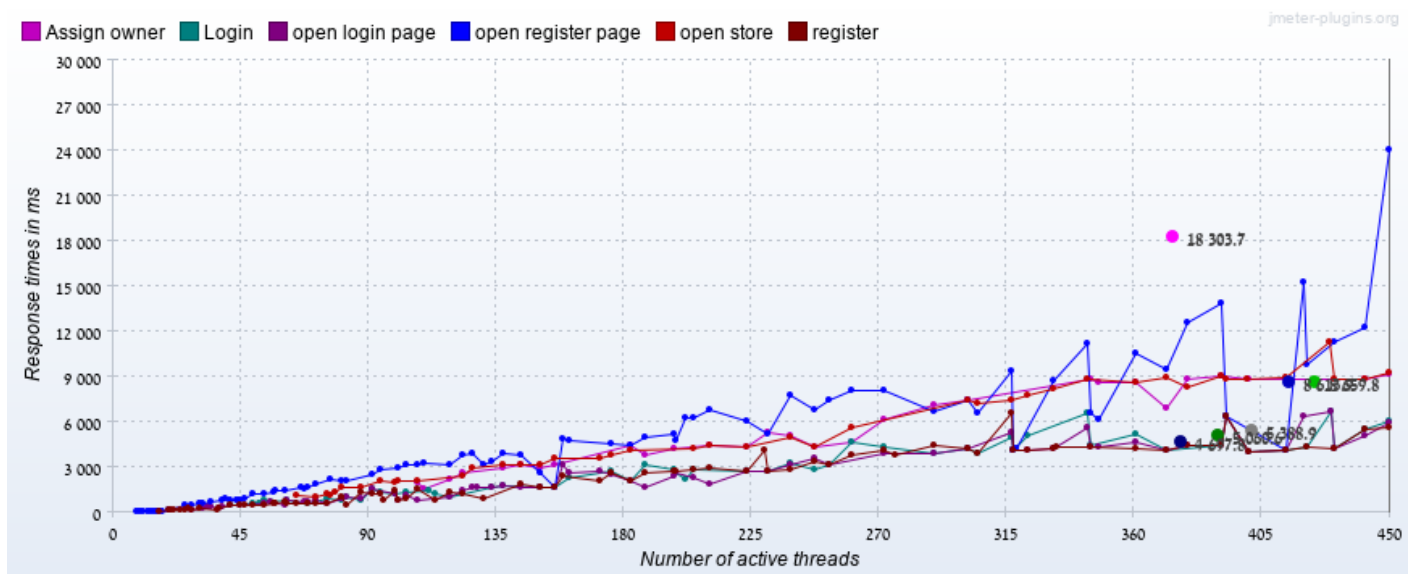


תרחיש 5 – פתיחת חנות ומינוי בעל חנות נוסף

מאחר שראינו בדיקות הקיבול כי מדובר בפעולה שמאלצת את השרת להגיב בזמן גבוה יותר, נסתפק בלבצע את בדיקת העומס עבור עומס משתנה שמגיע תוך חצי דקה לעומס קיצוני של 450 משתמשים שמבצעים חיפוש במקביל. התוצאות שקיבלנו הם:

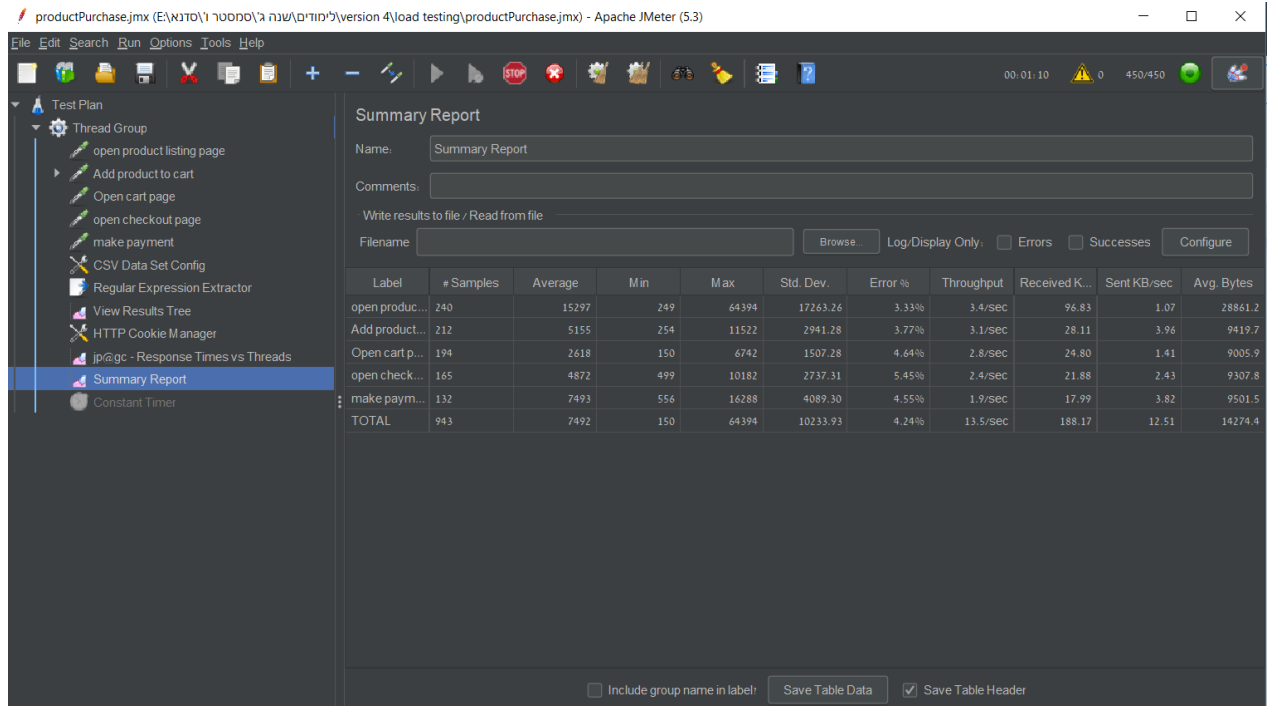


נשים לב כי קיבלנו בסה"כ 0.13% שגיאה, ויש לציין כי אחוזים אלו היו גבוהים יותר אך המערכת שלנו התאוששה מהשגיאות ולאחר השגיאות התבצעו בקשות HTTP בהצלחה דבר אשר הוריד את האחוזים של השגיאות. עם זאת, למרות אחוזי השגיאות הנמוכים יש לציין כי קיבלנו זמני תגובה חריגים וקיצוניים מהשרת עבור כמות גדולה כזאת של משתמשים, נציג זאת באמצעות הגרף:



תרחיש 6- רכישה

מאחר שראינו בבדיקות הקיבול כי מדובר בפעולה שמאלצת את השרת להגיב בזמן גבוה יותר, נסתפק בלבצע את בדיקת העומס עבור עומס משתנה שמגיע תוך חצי דקה לעומס קיצוני של 450 משתמשים שמבצעים חיפוש במקביל. התוצאות שקיבלנו הם:



productPurchase.jmx (E:\סדנא\ר' \מסמסטר \שנה 4\load testing\productPurchase.jmx) - Apache JMeter (5.3)

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
open produc...	240	15297	249	64394	17263.26	3.33%	3.4/sec	96.83	1.07	28861.2
Add product...	212	5155	254	11522	2941.28	3.77%	3.1/sec	28.11	3.96	9419.7
Open cart p...	194	2618	150	6742	1507.28	4.64%	2.8/sec	24.80	1.41	9005.9
open check...	165	4872	499	10182	2737.31	5.45%	2.4/sec	21.88	2.43	9307.8
make paym...	132	7493	556	16288	4089.30	4.55%	1.9/sec	17.99	3.82	9501.5
TOTAL	943	7492	150	64394	10233.93	4.24%	13.5/sec	188.17	12.51	14274.4

☐ Include group name in label: ☒ Save Table Header

קיבלנו סך הכל 4.24% שגיאות בכל הפעולות שביצענו. המערכת המשיכה לפעול ולא קרסה והתאוששה משגיאות אלו בצורה טובה. כמו כן, בדומה לבדיקות העומס שביצענו בתרחישים הקודמים, קיבלנו זמני תגובה גבוהים מאוד מהשרת עבור כמות גדולה כזאת של משתמשים.

4. סיכום:

באמצעות בדיקות העומס והקיבול שביצענו הצלחנו להבין את ביצועי המערכת בצורה טובה יותר ולהבטיח כי המשתמשים יקבלו את חוויית השימוש לה הם מצפים. דרישות רמת השירות מתקיימות עבור העומס המוגדר והמערכת לא קורסת גם כאשר מדובר במקרים בהם משתמשים רבים מחוברים במקביל. עם זאת, כאשר ישנו עומס גבוה מהרגיל לשרת לוקח זמן רב להגיב. המערכת מסוגלת להתמודד עם 1000 משתמשים שנמצאים בה במקביל ומצליחה להתמודד עם 100 בקשות שונות ברחבי האתר כגון: רישום, התחברות, חיפוש, רכישה וכו'.