בס"ד 7.04.19

**הנחיות פרויקט – הנדסת תוכנה לאינטרנט תשע"ט**

כללי:

במסגרת פרויקט זה נפתח שרות אינטרנט של "חברה". השרות יכלול את האפשרויות הבאות באתר שלו:

* משאבים (מוצרים) לבחירה ו-\או מכירה (בכמות שונה, עם ו-\או בלי מחיר וכו') עם קטלוג או כדומה, דוגמאות: חנות מקוונת, שיתוף ספרים, חנות יד שניה, שיתוף תכנות\קודים, וכו' וכדומה
* מספר קטגוריות משתמשים\גולשים (מנהל האתר, מנהל חנות\סניף וכדומה, לקוח\מנוי, גולש מזדמן, וכו')
* ניהול מוצרים (רק ע"י משתמשים מורשים לפי הקטגוריה)
* ניהול משתמשים (רק ע"י מנהל האתר)
* ניהול הזמנות והיסטורית הזמנות (בסיסי)
* בלוג+צ'אט (גם וגם!) שבו משתמשים\גולשים יכולים להחליף מידע, לשתף וכו' במסגרת קבוצות\קהילות

על כל זוג לבחור לעצמו נושא הפרויקט שלכם (מפורט) ומבנה מדויק של האתר ולאשר אצל מרצה מעבדה.

צד שרת של הפרויקט יתבצע אך ורק בסביבת nodejs ו-expressjs ב-JavaScript, הנתונים ישמרו בבסיס נתונים של MongoDB.

**אתר השרות**:

* האתר יפותח בטכנולוגית SPA בלבד (חוץ ממסכי לוג-אין, אם נדרש)
* צד לקוח יפותח בעזרת React על בסיס html5, css3 תוך שימוש בספריות BootStrap ו-JQuery או חילופיהם לפי הצורך
* האתר יכלול כותרת, מערכת בחירה (סרגל, תפריות וכו') וחלק תחתון כמקובל באתרים המודרניים
* כל פונקציונליות נוספת שבחרתם עבור האתר
* אתר לקוח\גולש יכלול לפחות את הדפים:
  + לגבי החברה
  + קטלוג מוצרים \משאבים
  + בחירה, קניה וכדומה
  + עיון בפרטי מוצר\משאב והוספתו לסל קניות (עם כמות)
  + סל קניות
  + היסטורית הזמנות (ללקוח רשום)
  + צ'אט\לוג-אין
* אתר עובד האתר\מוכר יאפשר לפחות:
  + להיכנס לרשימת הזמנות
  + לעדכן\לשנות הזמנות קיימות
  + לעדכן כמות מוצרים\משאבים "במחסן"
* אתר המנהל יאפשר לפחות:
  + כל דבר כמו העובד כנ"ל
  + לנהל משתמשים
  + לנהל קבוצות דיון

החלקים הספציפיים והתצוגה ייקבעו ע"י הסטודנטים המבצעים את הפרויקט. ייתכן שסטודנטים ישקיעו בתכונות ייחודיות על חשבון חלק מפונקציונליות אחרת (אך לא על חשבון החלקים העיקריים!). בכל מקרה אתר דל מידי יקבל ציון בהתאם.

**ניהול משתמשים**:

* אימות המשתמש חייב להתבצע רק בצד שרת, בעזרת middleware (למשל passport), אסור להעביר את הסיסמה על קווי תקשורת (חוץ מרישום – וגם אז בצורה מוצפנת). כל הדרישות לפחות כמו בתרגיל 6. מומלץ להשתמש ב-RSA. ב**טכנולוגיות נוספות** ניתן גם (בנוסף לרגיל) להתבסס על שירותי אימות משתמש של גוגל או פייסבוק וכו'.
* ניהול משתמשים יכלול יכולת למחוק, להוסיף ולשנות את תפקיד המשתמש או פרטיו האחרים (ע"י המנהל)
* המערכת תאפשר רישום עצמי של משתמשים מהקטגוריה הנמוכה, כל משתמש יוכל לעדכן את פרטיו המתאימים – כגון סיסמה, כתובת, טלפון וכו', לבקש לאפס את סיסמתו ("שכחתי את הסיסמה").

**שרות בלוג\צ'אט**:

* השרות ינוהל בעזרת socket.io – כל שינוי בקבוצות, הודעות וכדומה יתעדכן מיידית אצל כל הגולשים שנמצאים בחלון צ'אט או ניהול קבוצות
* על אדם הרוצה להקים קבוצה או להצטרף לקבוצה קיימת או לבצע כל פעילות אישית באתר יש להירשם באמצעות "תופס" מתאים ולקבל אישור מאדמיניסטרטור במערכת, או ממנהל הקבוצה. הרישום כולל פרטים אישיים, פרטי קשר ותמונה
* המערכת תאפשר לשלוח הודעות (post) הכוללות טקסט, תמונה ו-\או לינק (קישור), בציון האם ההודעה תגיע רק למחוברים כרגע או לכל חברי הקבוצה לכשיתחברו – לכלל המשתמשים, לקבוצה מסוימת או למשתמש מסוים באופן פרטי
* כתגובה לפוסט יוכל כל חבר קבוצה לבצע Like או Unlike לפוסט זה. לכל Post נשמר תאריך, שעת השליחה, מספר ההמלצות החיוביות והשליליות, לרבות זיהוי משתמש שממנו הגיע
* הוספת לייק או אן-לייק תשוקף מיידית במסך גולש הצופה כרגע בפוסט
* המערכת תאפשר לחוות דעה לגבי כל הודעה ( בעד או נגד ) כללית או קבוצתית ותסכם לפי סוגי חוות הדעת – העדכון לכל הגולשים בזמן שליחה שרואים את הפוסט יהיה מיידי
* המערכת תאפשר חיפוש והצגה, במסגרת נתוני משתמש, של כל ההודעות שהוא שלח  
  בתחום תאריכים מסוים ( ניתן לסנן לפי מילות מפתח שהופיעו בהודעה )
* המערכת תאפשר הצגת סה"כ מספר התגובות החיוביות והשליליות שקיבל משתמש על כל הודעותיו
* בכניסה לצ'אט יראה הגולש את 20 ההודעות האחרונות כברירת מחדל (כל הודעה כולל את מספר התגובות החיוביות (מספר הלייקים), השליליות (unlike), וכן אפשרות לבצע פעולות אלה במידה ולא נעשתה כזו עבור הודעה זו מאותו משתמש
* הגולש יוכל לבקש להוסיף ולראות עוד 20 הודעות קודמות, ועוד, ועוד...
* באמצעות מנגנון חיפוש\סינון ניתן לסנן את כל ההודעות בהן מופיעה מילה מסוימת ולדפדף ביניהם

**דרישות כלליות**:

* מבנה הפרויקט צד שרת יהיה תואם לארכיטקטורה MVC וכהנה גם המודולים
* מצב כניסה ונתוני גלישה אחרים הרלוונטים לגלישה (כולל סל קניות של גולש מזדמן) תנוהל ע"י session שפוקע תוך זמן סביר (למשל 15 דקות)
* השירות לא ישלח לכיוון צד הלקוח שום נתון ושום אלמנט אתר שלא תואם את הרשאות ואת הסמכויות של המשתמש – ההחלטה תתקבל בעזרת בקרה של השרת על בקשות AJAX ובקשות לדפי React, כמו כן ע"י טעינה עצלה של React, כמו כן הכלים המתאימים של React.
* השירות יתמוך בהספקת מידע וחלקי האתר ללקוח על בסיס סגנון ארכיטקטוני REST
* השרת יבדוק את כל הנתונים המתקבלים מהלקוח ולעולם לא יסמוך על הבדיקה בקוד צד לקוח

**דרישות לא פונקציונאליות**:

1. המערכת תתבסס על טכנולוגיות MEAN וחבילות הרחבה נוספות שתמצאו לנכון להוסיף.
2. יש לארגן פיזית את הקוד בעץ ספריות סטנדרטי, ניתן להשתמש בחבילת express-generator (<http://expressjs.com/en/starter/generator.html> ) או בכל IDE התומך בתבניות  
   פרוייקטים מסוג זה (VS 2017 vscode ,web storm,).
3. מומלץ לעצב את התצוגה באמצעות BootStrap או לבחור תבנית מוכנה ולהתאימה לצרכים ( לדוגמא: <http://w3layouts.com/> ), בכל מקרה על המערכת להיות אסתטית ומקצועית.
4. השתדלו, במידת האפשר ובאופן סביר, להעדיף עיבודים בצד הלקוח על פני צד השרת, כדי לשפר את היכולת לשרת מקסימום בקשות ומשתמשים בו זמנית – אך יש לבדוק\לאמת נתונים בצד שרת לפני ביצוע כל פעולה.
5. מומלץ להגדיר אוטומציית בדיקות ,ליצור חבילה לפתרון ולשלוח אותה למערכת npm במהלך הפיתוח (<http://blog.npmjs.org/post/127671403050/testing-and-deploying-with-ordered-npm-run-scripts>).
6. ניתן לפרוס את הפתרון ואת תשתיות האחסון ושירותי ענן.
7. בסיס הנתונים של ניהול ה-sessions (למשל backend של express-session) חייב להיות ביחד עם השרת – אסור למקמו בענן אלא עם מפעילים אותו באותה "מכונה" יחד עם השרת.
8. **על כל זוג להוסיף שימוש לפחות בשתי טכנולוגיות שלא נלמדו במסגרת הקורס – כגון שימוש smtp, רישום בעזרת רשתות חברתיות נפוצות, תשלום דרך paypal, וכו' וכדומה**

**בהצלחה!**