בס"ד 30.05.18

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

כללי:

במסגרת פרויקט זה נפתח שרות אינטרנט מבוזר של "חברה" הכולל אתר אינטרנט עם מערכת ניהול משתמשים ומערכת להעברת מסרים ותקשורת בין חברי קבוצות או קהילות. על כל זוג לבחור לעצמו נושא מועדף ומבנה מדויק של האתר.

נושא- אתר לקהילת she codes;. המשתמשות יהיו משתתפות בסניפים, צוותי הסניפים, ומנהלות האתר. מבנה האתר יהיה:

תפריט עליון: בחירת שפה לאתר, שאלות ותשובות, רישום לאתר

לאחר התחברות תיווסף אפשרות גישה לכרטיסיה- קבוצות סניפיות, משתתפות מחוברות

תפריט תחתון: צור קשר, מי אנחנו, שותפים לדרך

מערכת העברת ההודעות תאפשר- פנייה לכל חברות הסניף. פנייה למשתתפת או חברת צוות באופן אישי. פנייה רק לחברות שמחוברות למערכת. ניתן יהיה להגיב בlike או unlike.

**צד שרת של הפרויקט יתבצע אך ורק בסביבת nodejs ו-expressjs ב-JavaScript:**

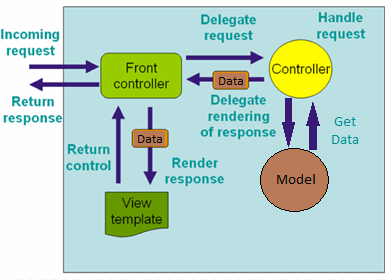
* המערכת תכלול שרת אינטרנט כדלקמן
* אימות המשתמש חייב להתבצע רק בצד שרת, בעזרת middleware (למשל passport), אסור להעביר את הסיסמה על קווי תקשורת (חוץ מרישום – וגם אז בצורה מוצפנת). כל הדרישות לפחות כמו בתרגיל 6.
* המערכת תנהל משתמשים ותפקידם ( אדמין, אחראי קבוצה, משתמש רשום וכו' וכדומה)
* על אדם הרוצה להקים קבוצה או להצטרף לקבוצה קיימת או לבצע כל פעילות אישית באתר יש להירשם באמצעות תופס מתאים ולקבל אישור מאדמיניסטרטור במערכת, או מנהל הקבוצה. הרישום כולל פרטים אישיים, פרטי קשר ותמונה
* מצב כניסה ונתוני גלישה אחרים הרלוונטים לגלישה אחת תנוהל ע"י session שפוקע תוך זמן סביר (למשל 15 דקות)
* מבנה הפרויקט צד שרת יהיה תואם לארכיטקטורה MVC\MVVM\MVW וכהנה גם המודולים
* השירות לא ישלח לכיוון הלקוח שום נתון ושום אלמנט אתר שלא תואם את הרשאות ואת הסמכויות של המשתמש הגולש – ההחלטה תתקבל בעזרת בקרה של השרת על בקשות AJAX ובקשות אנגולר, כמו כן ע"י טעינה עצלה של אנגולר, כמו כן Guard-ים של אנגולר
* האתר יכלול כותרת, מערכת בחירה וחלק תחתון כמקובל באתרים המודרניים, כמו כן את המידע הכללי על, ניהול משתמשים, שירות הודעות, כמו כן חנות או כל פונקציונליות אחרת שבחרתם עבור האתר
* השירות יתמוך בהספקת מידע וחלקי האתר ללקוח על בסיס סגנון ארכיטקטוני REST
* השרת יבדוק את כל הנתונים המתקבלים מהלקוח ולעולם לא יסמוך על בדיקה בקוד צד לקוח
* המערכת תנהל את הקבוצות עבור מערכת המסרים ופונקציות נוספות באתר
* לאחר רישום משתמש ניתן לבקש להצטרף לכל קבוצה, ובמידה ויוצר הקבוצה מאשר, המשתמש מצורף, כמו כן ניתן לפתוח קבוצה חדשה ולאשר למבקשים להצטרף
* מערכת העברת המסרים תיענה על הדרישות הבאות:
  + ספרית socket.io תשמש את המערכת בצד שרת וצד לקוח ואת ההודעות
  + 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 יכול כל חבר קבוצה לבצע Like או Unlike לפוסט זה. לכל Post נשמר תאריך, שעת השליחה, מספר ההמלצות החיוביות והשליליות, לרבות זיהוי משתמש שממנו הגיע
  + המערכת תאפשר לחוות דעה לגבי כל הודעה ( בעד או נגד ) כללית או קבוצתית ותסכום את סוגי חוות הדעת
  + 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
  + המערכת תאפשר, לגבי כל קבוצה, לראות 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
  + באמצעות מנגנון חיפוש ניתן לסנן את כל ההודעות בהן מופיעה מילה מסוימת ולדפדף ביניהם

**צד לקוח של הפרויקט יפותח כדלקמן:**

* html5, css3 תוך שימוש בספריות BootStrap ו-JQuery או חילופיהם לפי הצורך
* האתר יפותח אך ורק בטכנולוגיה SPA (Single Page Application) בעזרת פריימוורק Angular 6 וספריות נלוות, יש לבצע ניתוב אנגולרי
* משום סיבה לא ייטען האתר במלואו מחדש בעכבות גלישה רציפה – כל הנתונים וחלקים של האתר ייטענו בטכנולוגית AJAX על בסיס סגנון ארכיטקטוני REST
* מבנה קוד צד לקוח חייב להיבנות בארכיטקטורת MVVM\MVC\MVW

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. יש לארגן את קוד צד שרת באמצעות מולודציה המבוססת על תבנית Front-Controller או כל ואריאציה המחלקת תחומי אחריות בין בקרה, ניהול נתונים ויצירת תצוגה, ניתן להשתמש בחבילות מוכנות  
   או לחלק את הקוד בעיצוב עצמי, ככלל לכל משימה ( Task ) הגדירו Controller נפרד:



1. 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.
2. 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>).
3. ניתן לפרוס את הפתרון לתשתיות אחסון ושירותי ענן ( לדוגמא Cloud9 או Heroku ):

<https://c9.io/>  
<https://devcenter.heroku.com/articles/getting-started-with-nodejs#introduction>

1. **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.