דף אפיון פרויקט אמצע- ריאקט

למידה עצמית היא Skill מאוד מרכזי במקצוע שלנו, ראוי להשתמש בו ככל האפשר במהלך הפרויקט. עם זאת, חשבו את הזמנים מראש (עוד Skill חשוב במקצוע שלנו) וראו לסיים את הפרויקט ולהגישו בזמן (ישנם גם שיעורי בית שוטפים לתרגול ונושאים חדשים שאתם לומדים בקורס). השתדלו לא להיגרר ללימוד ויישום עוד ועוד נושאים אם זה יפריע לכם להגיש את הפרויקט בזמן.

נדרשת השקעה של כ 60 שעות פיתוח ויש חודשיים להגשה

דרישות כלליות:

- יש לשמור על קוד נקי ומסודר: לנקות console.log וקטעי קוד שהפכו להערות.
- מומלץ לבנות קוד שמספר סיפור ולתת לפונקציות ולמשתנים שמות משמעותיים.
- יש לחלק את הפרויקט למודולים לפי נושאים
- יש לשים את כל הקשור לעיצוב בקבצי css, את התמונות בתיקיית images וכו'.
- כמו כן יש להקפיד על המוסכמות לכתיבת קוד.
- יש ליצור פרוייקט ריאקט מלא באמצעות הפקודה npx create-react-app
- עיצוב הוא חלק בלתי נפרד מהצגת הפרויקט והיכולות של הפיתוח. גם אם אינך מעצב באופי, הקפד על עיצוב נקי ורספוניסבי לגדלים שונים של מסכים!

חשוב לזכור שפרויקט זה יהיה חלק מתיק העבודות שלכם וייצג אותך בכבוד מול מעסיקים פוטנציאליים ולכן יש לשמור על אסתטיקה של קוד ועיצוב.

ניהול ואפיון המידע המידע:

- **רשימת משתמשים מוכנה** יש לשמור רשימת משתמשים מוכנה מראש בקובץ JSON שיקרא users.json. את הקובץ ניתן לקרוא באמצעות fetch או axios. על כל משתמש להכיל לפחות את התכונות ההבאות (ניתן להוסיף מעבר לכך)- שם משתמש, סיסמא, מייל, תמונת פרופיל (ניתן להוריד תמונות פרופיל דמו מהאינטרנט), סוג המשתמש (יש לתכנן את האפליקציה שתכיל שני סוגים של משתמשים לפחות לדוגמא: אדמין ויוזר, אדמין ועובד, אדמין ולקוח). שים לב כי רשימת המשתמשים היא קבועה ולא נבצע עליה פעולות של הוספה, מחיקה וכו' דרך הקוד אלא רק נקרא ונציג, כמו כן נבצע התחברות אל מול רשימת המשתמשים הזו.
- **המידע של התוכנית** (אם נגיד מדובר בחנות אז המודלים הם קטגוריות, מוצרים, הזמנות) יאוחסן בlocal storage, בעת העלייה הראשונית בלבד של התוכנית, יש לבדוק האם המידע קיים ב local storage במידה וכן מעולה, במידה ולא אכלסו כמה פריטים מכל מודל. על כל מודל יש לאפשר לתמיכה בביצוע של פעולות CRUD, שים לב שלעיתים יש פעולות שרק יוזר מסוג מסויים יכול לבצע לדוגמא בחנות רק אדמין יכול להוסיף מוצרים חדשים. וגם יש מקרים שבהם יש גישה רק לדברים ששייכים ליוזר שאנחנו מחוברים איתו, לדוגמא אם לקוח צופה בהזמנות בחנות הוא יראה רק את ההזמנות שלו. על מנת לקשר בין מודלים שונים כמו לקוח והזמנה יש לקשר זאת כמו כמו שעושים בDB, להגדיר שדה בהזמנה שייצג את הID של הלקוח הרלוונטי.
- על הפרויקט להכיל לפחות 3 דרגות עומק של פעולות. לדוגמה:
  1. בחר משתמש
  2. לקוחות
     א. רשימת לקוחות => צלילה ללקוח ספציפי והצגת כל ההזמנות שלן => צלילה לפרטי הזמנה
  3. הזמנות
     א. רשימת הזמנות
        I. הזמנה 1

II. הזמנה 2
III. הזמנה 3

ב. הוספת הזמנה
ג. עריכת הזמנה
ד. מחיקת הזמנה

דרישות טכנולוגיות צד לקוח:

- **דף כניסה** צריך לכלול כותרת ראשית, כותרת משנית, טקסט ותמונה שיתאימו לאופי האתר/ האפליקציה. אם מדובר באתר של חנות אינטרנטית כלשהי, יש להציג בדף הפתיחה שדה חיפוש ולפחות שלושה כרטיסי מוצר. מדף הפתיחה צריך להיות ברור לאיזה סוג של אתר/ אפליקציה הגענו וצריך להיות מעוצב בצורה כזאת שתזמין את הגולש להמשיך להשתמש באתר.
- **תפריט ניווט** על האתר/ אפליקציה להכיל תפריט ניווט דינאמי שמשותף לכל דפי האתר
- **Footer** על האתר / אפליקציה להכיל footer עם לוגו, זכויות יוצרים ואמצעי ליצור קשר עם האתר. במידת הצורך ניתן להוסיף גם בתפריט הניווט, קישורים למדיה חברתית או כל דבר אחר שיתאים לאזור זה באפליקציה/ אתר
- **נגישות** יש לשים את שם האפליקציה בתגית ה – title בקובץ ה – index הראשי, וכן תמונה/ לוגו ב– link:favicon. כל תמונה חייבת לכלול את האטריבוט alt עם כיתוב שיתאר את התמונה.
- **דף אודות** יש ליצור דף אודות בו תספקו הסבר מעמיק על האתר ודרך ההתממשקות עמו.
- **התחברות** על ממשק צד לקוח להציג דף התחברות הכולל וולידציות על שדות הטפסים השונים. יש לאפשר שליחה של טופס אך ורק לאחר שכל שדות החובה מלאים ועוברים את שאר הולידציות. בלחיצה על כפתור השלח בטופס, יש לעדכן את הגולש בהצלחה או כישלון שליחת הנתונים. על מנת לעדכן את הגולש בהצלחה או כישלון ניתן להשתמש בספריית react-toastify או SweetAlert2 או בקומפוננות מוכנות של Material-UI
- **Crud** לאחר התחברות יש לאפשר למשתמש את פעולות ה – crud, קרי: קריאה, יצירה, עדכון ומחיקה של תוכן. התוכן שיוצרים צריך להיות זמין בחלקים שונים של האתר. לדוגמה אם מדובר בחנות אינטרנטית, לאחר שמוסיפים מוצר הוא צריך להופיע בדף הראשי או בדף מוצרים. יש לעדכן את הגולש בכישלון/הצלחה של הפעולות ע"י שימוש באחת הספריות שצויינו בסעיף התחברות.
- **דף פרטי תוכן** בלחיצה על כרטיס/ משתמש/ תוכן, הגולש יעבור לדף דינאמי בו יינתנו פרטים נוספים על פריט התוכן עליו לחץ הגולש
- **שדה חיפוש** יש ליצור שדה חיפוש לתוכן (כרטיס/ מוצר/משתמש וכ"ו)
- **ארכיטקטורה** יש לשמור על סדר הגיוני ומקובל בתעשייה של קבצים. על הקוד להיות נקי וקריא, עם חלוקה נכונה לתיקיות וקומפוננטות.
- **Console** יש להקפיד על עבודה נכונה עם ה – console. על הקונסול להיות נקי מהערות אזהרה, שגיאות ותוכן, כך שיהיה ניתן לראות בקלות שגיאות קריטיות מהשרת.
- **סינון תוכן** יש לתת לגולש אפשרות לסנן את התוכן המוצג בדף מסוים לפי פרמטרים שונים
- **חלוקה לדפים** שים לב שהאפליקצייה שלך תכיל מס' דפים לפחות שיש קישוריות בינהם (יש

להשתמש בReact Router). במידה ויש טבלאות באפליקציה יש לאפשר חלוקה לעמודים לפי כמות רשימות בכל עמוד. (Pagination)

בונוס (למידה עצמית):

- **קובץ העיצוב הראשי** אם קובץ העיצוב (css) הוא מעל 100 שורות, יש לחלק אותו לקבצים נפרדים לפי הנושאים השונים. לצורך העניין מומלץ להשתמש בספריית scss
- **קריאות http** יש למצוא API שקשור לתוכנית שלנו, לדוגמא API דמו של מוצרים במידה והפרוייקט שלנו הוא חנות. יש לבצע קריאה לAPI זה. ניתן להשתמש בfetch או axios. יש להשתמש במנגנון try & catch בקריאות אסינכרוניות לצד שרת.
- **עיצוב ורספונסיביות** מומלץ להשתמש בספריית bootstrap או Material UI
- **אייקונים** מומלץ להשתמש באייקונים לדוגמה מספריית font awesome או material icons
- **Regex** יש להשתמש ב – regex עבור ואלידציות של שדות מסויימים לדוגמא: מייל, סיסמא, טלפון.

אופן הגשת הפרויקט:
את הפרויקט יש להעלות ל – git ללא תיקיות ה – node_modules (יש להשתמש בgitignore), ולהעלות לאתר הקמפוס את הקישור יחד עם ספר הפרוייקט שהוא קובץ טקסט ReadMe.md שמסביר על הפרויקט, תכולתו, הפונקציונאליות ודרך ההתממשקות עמו, יש לדאוג שהקובץ יתממשק עם הגיטהאב.