פרויקט גמר דוטנט

מסמך הנחיות ודגשים לפרויקט

מבוא:

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) וראוי לסיים את הפרויקט ולהגישו בזמן. השתדלו לא להיגרר ללימוד ויישום עוד ועוד נושאים אם זה יפריע לכם להגיש את הפרויקט בזמן . נדרשת השקעה של כ 120 שעות פיתוח ויש עד 3 חודשים להגשה, המרצה/ הרכזות יגדירו לכם את תאריך ההגשה. לטובת הפרויקט תבנה מערכת שיש בה צד לקוח עם צד שרת תומך - בנושא שתבחר. על הפרויקט להיראות כמו אתר/ אפליקציה אמיתית עם תמונות ותוכן טקסטואלי שלא כוללים ipsum lorem וכדומה. כמו כן הפרויקט צריך להיות רספונסיבי ומתאים לכל גדלי המסך האפשריים

תיאור הפרויקט:

פיתוח אפליקציה אינטרנטית הכוללת פתרון end to end (צד שרת וצד לקוח). התוכן שיפורסם יהיה זמין בחלקים שונים של האתר. (לדוגמה אם מדובר בחנות אינטרנטית, לאחר שמוסיפים מוצר הוא צריך להופיע בדף הראשי או בדף מוצרים)

דגשים מרכזיים:

- אתר מרכזי הכולל עמוד תצוגת תוכן.
- מערכת התחברות הכוללת גישה לממשק ניהול אתר.
- ממשק ניהול האתר יאפשר: הוספה, עריכה או מחיקה של פריטים או תוכן.
- תוכן האתר יישמר בצד שרת.

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.

הערה כללית:

לא ניתן להגיש פרויקט סוף מודול כפרויקט גמר.

דרישות כלליות:

- יש לשמור על קוד נקי ומסודר: לנקות console.log וקטעי קוד שהפכו להערות, למחוק using statements וimports מיותרים, יש לרווח את הקוד כמקובל.
- מומלץ לבנות קוד שמספר סיפור ולתת לפונקציות ולמשתנים שמות משמעותיים.
- יש להקפיד על naming conventions.
- 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 פרויקט 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 בדאטה בייס:

- **כל המידע של התוכנית** ישמר בדאטהבייס SQL (יכול להיות PostgreSQL, SQL Server, SQLite, MySQL וכו')
- **טבלת משתמשים** תשמר כמובן בדאטה בייס, על כל משתמש להכיל לפחות את התכונות ההבאות (ניתן להוסיף מעבר לכך)- שם משתמש, סיסמא מוצפנת, 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).
- **פעולות CRUD** שים לב כי נבצע פעולות CRUD ואף על הטבלאות (נממש זאת באמצעות EF).
- **פעולות JOIN** יש להשתמש לפחות בJOIN אחד בפרוייקט (נממש זאת באמצעות EF).
- **אכלוס המידע הראשוני** יתבצע דרך הקוד
- **שדות חובה** יהיו not null
- **יש להגדיר אורכים מקסימליים לשדות**
- 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 צד לקוח:

- **יש לבנות צד לקוח באמצעות React**
- **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. בפרוייקט ריאקט 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 וכו')
- **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 שהאפליקציה שלך תכיל מס' דפים לפחות שיש קישוריות בינהם (בפרוייקט ריאקט יש להשתמש בReact Router). במידה ויש טבלאות באפליקציה יש לאפשר חלוקה לעמודים לפי כמות רשימות בכל עמוד (Pagination).
- **ואלידציות** יש לבצע ואליציות של שדות, ניתן להשתמש בRegex. המטרה שהמשתמש יוכל להזין רק נתונים תקינים.
- **קריאות http** בפרוייקט ריאקט ניתן לבצע קריאות http לשרת שלנו באמצעות fetch או axios, בפרוייקט אנגולר ניתן לבצע באמצעות httpClient.
- **עיצוב ורספונסיביות** בפרוייקט ריאקט מומלץ להשתמש בספריית bootstrap או Material UI, בפרוייקט אנגולר מומלץ להשתמש בngx-bootstrap.
- **אייקונים** מומלץ להשתמש באייקונים לדוגמה מספריית font awesome או material icons.
- **גישה לדפים,** אם ליוזר אין גישה לדף מסויים כי הוא לא מחובר יש לנווט או לדף התחברות, אם הנתיב אינו תקין יש להפנות לדף 404, אם היוזר הוא לא מהסוג הנכון, נגיד לקוח ולא אדמין יש להציג דף 401.

דרישות צד שרת:

- **יש לבנות Web API** באמצעות Asp.Net Core .
- **שימוש בעקרונות OOP-** ירושות, אינטרפייסים, אבסטרקט, כימוס.
- **יש להקפיד על מודולריות,** יצירת קונטרולרים שונים לישויות שונות, חלוקה למודולים, הפרדה לשכבות (כל הקריאות לDB חייבות להיות בשכבה נפרדת, DAL, אין לבצע קריאות לDB משכבת הAPI).
- **שימוש בDB באמצעות EF Core:** העבודה מול הDB חייבת להתבצע בכללותה באמצעות EF Core.
- **Design Patterns** יש להשתמש לפחות בשני design patterns.
- **קבצי קונפיגורציה** יש לאכסן את הconnection string וכל מידע רלוונטי אחר בקבצי קונפיגורציה מסוג json שכמובן התוכנה שלנו תקרא אותם.
- **Connection string** יש להשתמש בLOCALHOST.
- **מנגנוני Authentication & Authorization** יש לעבוד עם JWT Token, יש לבצע Authorization על routes או על משאבים שתבחרו באתר או האפליקציה, כך שרק יוזר מחובר או יוזר מסוג מסוים יכולים לגשת אליהם (ייתכן שיש כאן חומר להשלים בלימוד עצמי).
- **ואלידציות** יש לבצע ואלידציות לכל הנתונים שאני מקבלים מהקליינט.

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 או sass
- **שימוש בstate management** (במידה ומדובר בפרוייקט ריאקט ניתן להשתמש בRedux או useContext)
- **הטמעה של Logging בצד השרת** כמובן תעשה באמצעות Asp.Net Core
- **תהליך שרץ על Background Thread** יש ליצור תהליך כלשהו שרץ באופן אוטומטי בתזמון מסויים על גבי Background Thread
- **התחברות באמצעות פרוביידר** כמו פייסבוק או גוגל (יש להקפיד לא לשמור את הסיסמא)
- **שימוש ב API חיצוני** לצורך שמירת תמונות/מסמכים וכדומה, או העלאה של הפרוייקט לסביבת ענן.