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 מערכת **SERVER-CLIENT** אשר בה עליכם לממש אפליקציה לניהול והרצת מבחנים, המערכת תתמוך במשתמשים מסוג מרצה וסטודנטים כך שהמרצה יוכל לבנות מבחנים לערוך אותם ולראות סטטיסטיקות של הבחינות והסטודנטים יוכלו לצפות בציונים שלהם ולבצע בחינות.

דגשים מרכזיים:

- אתר מרכזי הכולל **עמוד תצוגת תוכן**.
- **מערכת התחברות** הכוללת גישה לממשק ניהול אתר.
- צד ה **Client** יתקשר עם צד השרת באמצעות REST **HTTP**.
- יש להשתמש בפקודות **GET/POST/PUT/DELETE/PATCH**.

אופן הגשת הפרויקט:

את הפרויקט יש להעלות ל – git ללא תיקיות ה-modules_node אם יש (יש להשתמש ב gitignore), 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.
- במידה ומדובר בריאקט, 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.

דרישות למצגת פרויקט:

- About
- General Flow & Architecture
- Modules (Client/Server)
- Views List
- Components and NUGETS
- API List/Documentation

במהלך הצגת הפרויקט הסטודנט יציג ויריץ את הפרויקט ויענה על שאלות שישאלו על ידי הבודק.

ניהול ואפיון המידע בדאטה בייס:

- **כל המידע של התוכנית** ישמר בדאטהבייס SQL/NoSQL (PostgreSQL, SQL Server, SQLite, MySQL, MongoDB 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, סטודנט ומרצה, אדמין סטודנט ומרצה).
- **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 **(Data Seeding)**.
- **שדות חובה** יהיו not null.
- **יש להגדיר אורכים מקסימליים לשדות**.
- המרצה יוסיף בחינה ואת השאלות שבבחינה:

נתוני בחינה:

- שם
- ID
- תאריך
- שם מורה\מרצה
- שעת התחלה
- זמן כולל
- האם יש סידור רנדומאלי של שאלות

- כל בחינה תכלול מספר שאלות ורשימה של תשובות אפשריות ומתוכן התשובה הנכונה.

נתוני שאלה:

- כל שאלה תאפשר הצגה של שאלת טקסט או שאלת תמונה
- אפשרויות לבחירת תשובה
- אינדקס התשובה הנכונה
- האם יש סידור רנדומאלי של אפשרויות בשאלה

- המרצה יוכל לשמור את נתוני הבחינה לשרת באמצעות פקודת API POST לשרת WEB ושמירת נתוני המבחן בבסיס הנתונים

הרצת בחינה עי סטודנט:

סטודנט יחפש וימצא את שם הבחינה לפי שם, יעדכן את פרטיו ויבצע התחלת בחינה במידה והתאריך והשעה תואמים בטווח הנוכחי.
בכל שלב תוצג לנבחן שאלה אחת והנבחן יוכל לעבור בין השאלות כרצונו וכמו כן יוצגו כמה שאלות יש ,כמה שאלות נפתרו וכמה שאלות נשארו ללא פתרון,
בנוסף יש להציג טיימר המראה כמה זמן נשאר לסיום הבחינה.

בתום ביצוע הבחינה יש לחשב את הציון שהתקבל ולשמור את הפרטים הבאים בבסיס הנתונים:

- תעודת זהות הנבחן
- שם הנבחן
- ID הבחינה
- ציון
- רשימת שגיאות
    - שם שאלה
    - תשובת טעות שנבחרה
    - תשובה נכונה

דרישות צד לקוח:

- **יש לבנות צד לקוח** באמצעות React/WPF/HTML-JS/MVC.
- **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 באמצעות שימוש ב-popup/modal, בפרויקט ריאקט ניתן להשתמש בספריית react-toastify או SweetAlert2 .

- **Crud** לאחר התחברות יש לאפשר למשתמש את פעולות ה – crud, 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 שהאפליקצייה 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).

- **ולידציות** יש לבצע ולידציות של שדות, ניתן להשתמש בRegex. המטרה שהמשתמש יוכל להזין רק נתונים תקינים.

- **קריאות http** ניתן לבצע קריאות http לשרת באמצעות fetch או axios.

- **עיצוב ורספונסיביות** בפרוייקט ריאקט מומלץ להשתמש בספריית bootstrap או Material UI.

- **אייקונים** מומלץ להשתמש באייקונים לדוגמה מספריית font awesome או bootstrap 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.

מסכים:

מסכי אורחים :

- **Login**
- **Register**

מסכי מרצה:

- חיפוש בחינה בשרת לפי שם .
- הוספת בחינה חדשה ואפשרות לשמור את המבחן כקובץ לוקאלי עד אשר מעדכנים אותו בשרת.
- עידכון בחינה.
- עידכון שאלות תשובות.
- שמירת נתוני בחינה לוקאלית לקובץ.
- הצגת סטטיסטיקה בבחינה של כל הסטודטים הכוללים לפחות CHART אחד.

מסכי סטודנטים:

- חיפוש בחינה בשרת לפי שם.
- תצוגת הסטוריית מבחנים ציונם והשגיאות המבחנים אלו.
- ביצוע בחינה ומעבר בין שאלות .

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, 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 בReduxאו useContext)
- **הטמעה של Logging בצד השרת** כמובן תעשה באמצעות Asp.Net Core, מומלץ להשתמש ב-Serilog ו-Sec.
- **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.