**לכבוד**

**יחידת הפרויקטים**

**מה"ט**

הצעה לפרויקט גמר

פרטי הסטודנטים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **שם הסטודנט** | **ת.ז** | **כתובת** | **טלפון נייד** | **תאריך סיום הלימודים** |
| הדסה מנוחה טלר | 323052829 | מודיעין עילית | 053.316.9452 | 2023 |
| פנינה חיון | 213879133 | מודיעין עילית | 058.320.2191 | 2023 |

**שם המכללה:** סמינר הליכות בית יעקב

סמל מכללה: 76074

**מסלול ההכשרה:** הנדסאים

**מגמת לימוד: הנדסת תוכנה**

**מקום ביצוע הפרויקט:** סמינר הליכות בית יעקב

פרטי המנחה האישי

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם המנחה | כתובת | טלפון נייד | תואר | מקום עבודה/תפקיד |
| **נוביק צביה** | **מסילת יוסף 40 מודיעין עילית** | **0527676885** | **B.A מדעי המחשב** | **הליכות בית יעקב** |



ה.מ. טלר ,פ.חיון

**חתימת הסטודנט חתימת המנחה האישי חתימת הגורם המקצועי מטעם מה"ט**

1.שם הפרויקט:

**Shop4you**

2. רקע:

**2.1** **תיאור ורקע כללי:**

**מערכת לתאום בין אנשים שנמצאים בחנות לבין אנשים שצריכים משהו מהחנות.**

האפליקציה מאפשרת לאדם להודיע שהוא בחנות ולבחור קבוצות מסוימות שלהם הוא מעונין לקנות, וכן אדם הנמצא בבית יכול ליצור הזמנה ולבחור קבוצות שהוא מעונין שהם יקנו. המערכת מקשרת בין השנים, ופותחת להם צ'אט להמשך תקשורת ושם נותנת אפשרות למעבר לאתר העברת כספים לצורך תשלום.

**2.2** **מטרות המערכת:**

**המערכת מאפשרת קניה לאחרים בצורה נוחה ויעילה, בדגש על הקפצות הודעות רלוונטיות בלבד, יוצרת מודעות בקרב האוכלוסייה לעזרה הדדית.**

3.סקירת מצב, אילו בעיות קימות :

קיימים הרבה אפשרויות לקבוצה לשלוח הודעות כלליות לכולם.   
 בקבוצות אלו יש הרבה הטרדות מיותרות לאנשים, לא לכולם הודעות אלו רלוונטיות.   
 בנוסף, אדם שמחפש הודעה מסוימת צריך לחפש בין הרבה הודעות נוספות...

4**.** מה הפרויקט אמור לחדש או לשפר:

מערכת זו שולחת הודעות רלוונטיות בלבד, רק מי שעדכן שנמצא בחנות יקבל הודעה על בקשת קניה.  
 וכן, האדם שצריך משהו מהחנות לא יצטרך לחפש מי השאיר הודעה שנמצא בחנות, בהנחה שעדין נמצא בחנות, אלא יכול להכניס הזמנה והיא תקפוץ למי שנמצא בחנות כרגע או כאשר מישהו יעדכן שהולך לחנות בתוך פרק זמן מוקצב אישית.

5. דרישות מערכת ופונקציונאליות**:**

5.1 **דרישות מערכת**:

הפרויקט יעבור קומפילציה בסביבת VSC תוך שימוש בספריית React לממשק המשתמש, וצד השרת ייכתב בסביבת הפיתוח של Node.js, ע"י טכנולוגית Nest.js.

הנתונים ישמרו במסד נתונים של MongoDB.

5.2 **שרידות, ביצועים / התמודדות עם עומסים**:

צד השרת המריץ IIExpress S מסוגל להתמודד עם מספר קריאות בו זמנית

5.3 **דרישות פונקציונאליות:**

המשתמש נכנס למערכת ומעדכן שהולך לחנות, כאשר תקפוץ הודעה של הזמנה חדשה, הוא יוכל לבחור אם לקנות או לא. אם הוא יבחר לקנות יפתח צ'אט להמשך תקשורת.

משתמש נוסף נכנס למערכת ויוצר הזמנת מוצר, כאשר תקפוץ הודעה שנמצא קונה מתאים, הוא יוכל לעבור לצ'אט שנפתח ביניהם.

כל משתמש יכול ליצור קבוצה, ולהזמין אנשים לקבוצות להם הוא שייך.

6.בעיות צפויות במהלך הפיתוח ופתרונות :

**6.1 תיאור הבעיות:**

* **יכול לקרות מצב ששני אנשים שנמצאים בחנות יקבלו הודעה על הזמנה מסוימת ושניהם יאשרו אותה.**
* **התצוגה צריכה להיות נוחה וברורה למשתמש**
* **התצוגה צריכה להתאים לכל גודל של מסך.**

**6.2 פתרונות אפשריים:**

* בעת אישור קניה, נבדוק שההזמנה עדין פנויה.
* **פתרונות אפשריים: שימוש באפשריות העיצוב של** CSS, Bootstrap, mui**.  
  הפתרון הנבחר: שימוש בספריית העיצוב** mui**.**
* **הגדרנו את כל האפליקציה בצורה של מסך מובייל, ובגודל יחסי לכל מסך.**

7. פתרון טכנולוגי נבחר:

7.1 **טופולוגית הפתרון:**

המערכת תשתמש בסביבת הפיתוח Node.js בצד ה Server, ובשפת Type script ובספריית React ובשפת Java Script בצד הClient .  
 תוך שימוש במסד הנתוניםMongoDB .

7.3 **שפות הפיתוח:**

**צד שרת:**

NodeJS – שפה פופולארית ויעילה לכתיבת שרת מהיר ואיכותי.

mongoDB – מסד נתונים המתממשק עם סביבת NodeJS.

* אבטחת השרת: Authorization Jwt Tokens,
* אבטחת המסד נתונים: שיבוש סיסמא ע"י Bcrypt

צד לקוח:

React - ספריה חזקה ומתקדמת לניהול צד לקוח.

* שפות בסיס: HTML, CSS, Java Script,
* ספריות עיצוב: Material React, react-spinner, react-hook-popup.
* אחסנת נתונים בצד לקוח: Cookies

7.4 **תיאור הארכיטקטורה הנבחרת**

העבודה תהיה במודל שרת - לקוח כמקובל באפליקציות אינטרנטיות. עיבוד צד שרת BackEnd, ועיבוד צד לקוח FrontEnd

7.5 **חלוקה לתוכניות ומודולים**

הפיתוח יתבסס על חלוקה ל- 3 שכבות:

* שכבת הגישה לנתונים - DAL
* שכבת הלוגיקה העסקית – BL
* Controller?
* שכבת האפליקציה – UI

7.**6** סביבת השרת

שרת אינטרנט.

7.7 **ממשק המשתמש/לקוח - GUI**

שכבת ה-GUI מורכבת מדפי HTML ו-TypeScript, שימוש ב- React.

7.8 **ממשקים למערכות אחרות / :API**

התממשקות לחשבון אימייל, ושליחת אימיילים.

התממשקות לצ'אט

7.9 **שימוש בחבילות תוכנה:**

* Axios
* material react
* Jwt Tokens
* Bcrypt

יישקל שימוש בחבילות נוספות לפי הצורך.

**8.** שימוש במבני נתונים וארגון קבצים:

8.1 **מבני הנתונים**

שימוש במבני נתונים רלציוניים (יחסיים)

8.2 **שיטת אחסון (מאגר, קבצים ובאיזו טכנולוגיה)**

נאחסן ב DB בסיסי נתונים טבלאיים עם MongoDB

8.3 **מנגנוני התאוששות מנפילה\קריסה\תמיכה בטרנזקציות**.

MongoDB בנוי בצורה המותאמת למספר גדול של קריאות.

וכן Node.js פועל בצורה אסינכרונית המאפשרת ניהול התהליכים בצורה יעילה המונעת עומס על המערכת.

כאשר השרת לא מצליח לשלוח מייל למשתמש, הוא שומר את הפרטים בקובץ וינסה מאוחר יותר.

9. תרשימי מערכת מרכזיים:

הולך לחנות

יוצר הזמנה

שינוי הגדרות

צפיה בהזמנות – וביטול הזמנה

10. תיאור המרכיב האלגוריתמי חישובי:

האפליקציה מחפשת להתאים למי שנמצא בחנות – הזמנה, ולמי שיוצר הזמנה - קונה. כל פרק זמן קצר.

11. תיאור/התייחסות לנושאי אבטחת מידע:

* שימוש בפרוטוקול https לצורך הצפנה של הנתונים בתקשורת בין ה client ל server
* ביצוע בקרת גישה לממשק הניהול באמצעות שם משתמש וסיסמה.
* הצפנת הסיסמא לפני השמירה במסד נתונים.
* שימוש בJwt Token לצורך Authorization.

12. משאבים הנדרשים לפרויקט:

12.1 **מספר השעות המוקדש לפרויקט- 700, חלוקת עבודה בין חברי הצוות**

כ 350 שעות לכל אחד מחברי הצוות.

12.2. **ציוד נדרש**

מחשב בעל מעבד I5, 8gb ram, הכולל חיבור לאינטרנט.

12.3. **תוכנות נדרשות:**

* Visual Studio Code
* MongoDB Compass
* Node.js
* **דפדפן**

12.4. **ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט:**

* Nest.js
* Pwa – סגירה לאפליקציה
* Git

עבודה עם שרתי אינטרנט והכרת הפרוטוקולים והאובייקטים בצד השרת

12.5. **ספרות ומקורות מידע:**

* <https://stackoverflow.com>
* <https://reactjs.org/>
* <https://nestjs.com/>
* <https://github.com/>
* <https://talkjs.com/>

13 .תכנית עבודה ושלבים למימוש הפרויקט**:**

יולי : הכנת הצעת פרויקט.

יולי : ניתוח ואפיון.

אוגוסט-ספטמבר-אוקטובר: בניית ממשק המשתמש וכתיבת הקוד הלוגי

אוקטובר: כתיבת ספר פרויקט.

נובמבר: בדיקות וניפוי שגיאות.

דצמבר: הגשת הפרוייקט.

14 . תכנון הבדיקות שיבוצעו:

14.1. **נא פרט בטבלה, בדיקות תהליכיות ברמת משתמש בהן נדרשת המערכת לעמוד**

|  |  |
| --- | --- |
|  | סיווג מוצר חדש |
| בדיקה 1 | האם התמונה תקינה וברורה |
| בדיקה 2 | האם מיד לאחר קבלת התשובה לגבי צבע הפח מוצג הערה לזריקת האשפה |
| בדיקה 3 | האם המערכת מציגה גרף נכון |

14.2. **נא פרט בטבלה, מס מייצג של בדיקות יחידה למודולים המרכזיים בהן נדרשת**

|  |  |
| --- | --- |
|  | קידוד תמונה של מוצר חדש |
| בדיקה 1 | שגיאת שרת |
| בדיקה 2 | נפח נתונים חריג |
| בדיקה 3 | האם התמונה ניתנת לקידוד |

|  |  |
| --- | --- |
| **הפעולה** | **התוצאה הנדרשת** |
| |  | | --- | | הכנסת הזמנה חדשה | | |  | | --- | | מעבר לדף להכנסת פרטים הדרושים להזמנה | |
| לחיצה על אישור הזמנה | המערכת מתחילה לחפש קונה רלוונטי |
| מציאת קונה רלוונטי | פתיחת צ'אט לשתי הכיוונים |
| המשך תקשורת ביניהם | קניית המוצר והעברת תשלום |
| סגירת צ'אט | סגירת התקשורת ביניהם |

|  |  |
| --- | --- |
| **הפעולה** | **התוצאה הנדרשת** |
| ניהול הצ'אט | התקשורת נשמעת ב 2 הצדדים . |
| קבוצות | מי שאינו בקבוצה זו אינו יכול לפרסם שהולך לחנות או רוצה להכניס הזמנה |
| התאמה בין בקונה לבין להזמנה | הנתונים מעוברים בצורה תקינה ב 2 הכיוונים מהאפליקציה לבסיס הנתונים ולהיפך. |

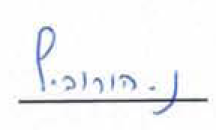
15 . בקרת גרסאות ( version control ):

נשתמש במערכת git השומרת בכל שינוי גרסה חדשה של המערכת, כך שבמידה ותהיה תקלה נוכל לעבור לגרסה קודמת.

בנוסף, נוכל לעבוד כצוות באמצעות עבודה בשרת בענן בGitHub, כך שתהיה עבודה משותפת והצוות יוכל לעבוד במקביל.

**הערות ראש המגמה במכללה:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**אישור ראש המגמה:**

**שם: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה: תאריך: \_\_\_\_\_\_\_**