תאריך: \_\_\_\_\_\_\_\_\_\_\_

### לכבוד

### יחידת הפרויקטים

מה"ט

### **הצעה לפרויקט גמר**

1. **פרטי הסטודנטים**

| שם הסטודנט | ת.ז. 9 ספרות | כתובת | טלפון נייד | תאריך סיום הלימודים |
| --- | --- | --- | --- | --- |
| ישי אזגורי | 305089963 | טבריה | 305089963 | 2024 |
| איתן צ'רקסוב | 320968498 | קיבוץ מסילות | 0526484332 | 2024 |

שם המכללה \_ מכללה טכנולוגית כנרת \_\_\_\_\_\_\_\_\_\_\_ סמל המכללה: 72228

מסלול ההכשרה: הנדסאים.

מגמת לימוד: \_תוכנה מקום ביצוע הפרויקט: \_\_ מכללה טכנולוגית כנרת

**פרטי המנחה האישי**

| שם המנחה \* | כתובת | טלפון נייד | תואר | מקום עבודה/תפקיד |
| --- | --- | --- | --- | --- |
| אמיר דגן |  |  |  |  |

\* עבור מנחה אישי חדש יש לצרף קורות חיים, ניסיון מקצועי ותעודות השכלה לאישור מה"ט.

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

# שם הפרויקט: חניון חכם

# רקע

# תיאור ורקע כללי

# מערכת של חניון על מרכיביו המתנהל באופן אוטומטי כולל ניהול מנויים, בקרת כניסה ויציאה, בקרת חניה תקינה, מיפוי וחיווי מצב התפוסה, שליחת התראות למשתמשים ולגורם המפעיל.

# מטרות המערכת התנהלות מושכלת ומיטבית של החניון תוך מעורבות אנושית מינימלית.

# סקירת מצב קיים בשוק, אילו בעיות קימות

# לצד החניונים ה"ידניים" קיימים בשוק כיום חניונים "חכמים" רבים אך רובם עדיין מתנהלים בעזרת התערבות אנושית ברמה שאינה מאפשרת מעטפת של אוטומציה מלאה. המאפיינים ה"חכמים" הינם מוגבלים יחסית למה שהטכנולוגיה היום מאפשרת ועל כך יפורט בסעיף הבא.

# תחזוקה של חניון חכם כרוכה לעלויות גבוהות יחסית ולצורך תמיכה טכנית שחלק מבעלי החניונים אינם יכולים או רוצים להשקיע.

# מה הפרויקט אמור לחדש או לשפר

# הפרויקט בא להמחיש כי באמצעות הכלים הנגישים היום לציבור המפתחים, ניתן ליצור מערכת שתאפשר שילוב בין חומרה, תכנה וממשק ידידותי למשתמש אשר מיועד לתמוך בניהול חניון אוטומטי ככל שניתן.

# באמצעות התקנת מצלמות בכל תא חניה, ניתן לקבל שליטה מלאה הרבה יותר שתאפשר בקרת שימוש ע"י החונים, אכיפת חניה תקינה, ניהול מנויים.

# לא מצאנו בשוק פתרונות התומכים באכיפת חניה תקינה (מניעת חניה כפולה או חורגת), מערכות לניהול החניה נתפסות כמותרות

# דרישות מערכת ופונקציונאליות

# דרישות מערכת:

# ההטמעה נעשית דרך ממשקי תקשורת, חיבורי רשת ורכיבים אלקטרוניים.

# בניהול מנויים, ההטמעה כוללת מערכת ייעודית לניהול בסיס נתונים של המנויים, שמאפשרת עדכון וניהול יעיל של רשומות. זה מספק יכולת לזהות מנויים, לבדוק תקופות חיוב ולנהל את החשבונות שלהם, כולל תמיכה בתקשורת מול המשתמש באמצעות התראות.

# בקרת כניסה ויציאה ממומשת בצורה אוטומטית דרך מערכת זיהוי באמצעות מצלמות ייעודיות. זה מבטיח שרכבים מורשים יכנסו ויצאו בצורה מהירה ויעילה, מפחיתה את הצורך בהתערבות אנושית.

# בקרת חניה תקינה ומיפוי התפוסה נעשים באמצעות מיפוי השטח המוקצה לכל חניה. המערכת יכולה לזהות את מצב התפוסה בזמן אמת, לספק למשתמשים מידע עדכני בנוגע לזמינות של חניות בזמן אמת. במקביל מערכת זו תאפשר בצורה יעילה ואוטומטית התראות למשתמשים ולגורמים המפעילים במקרה של כל תקלה או שינוי חריג במצב החניון או במצב חניה לא תקינה של רכב ספציפי.

# שימוש וביצועים:

# המערכת מתופעלת באמצעות ממשק Web, אשר יהפוך את השימוש והניהול שלה לנגיש וידידותי למשתמש. כל משתמש יכול לגשת למערכת דרך הגישה האישית שלו ולבצע פעולות שונות, כמו בדיקת סטטוס המנוי, או קבלת התראות בזמן אמת.

# על מנת להתגבר על תקלות כתוצאה מעומס, הקוד של הממשק צריך להיות כתוב בצורה יעילה ככל האפשר. ישנם פתרונות תכנות רבים לניתוב מושכל של העומסים ואין כל מניעה מליישם אותם.

# דרישות פונקציונאליות רשימת דרישות המשתמש מהמערכת, מהן הפעולות בהן נדרשת המערכת לתמוך. זיהוי מספרי לוחית בכל תא חניה, בכניסה וביציאה – באמצעות מצלמות ייעודיות

# זיהוי חניה תקינה בכל תא חניה ובאזורי הגבולות בין התאים – באמצעות חיישנים ייעודיים

# חיווי חניה בלתי תקינה – באמצעות הבהוב נורה ושליחת מסרון לבעל המנוי ו\או מפעיל החניון

# מיפוי מקומות תפוסים \ פנויים

# חיווי תפוסה – הן במסכים והן באפליקציה

# חיווי תפוסה – באמצעות נורות בצבעים שונים

# ניהול מנויים – באמצעות מסד נתונים ייעודי

# מנגנון לשליחת התראות

# מערכת זיהוי תקלות בחומרה ובשכבת ניהול החומרה

# בעיות צפויות במהלך הפיתוח ופתרונות (תפעוליות, טכנולוגיות, עומס ועוד):

# תיאור הבעיות - הללו כפועל יוצא של דרישות המשתמש מהתוכנה.

# ניהול מסד נתונים:

# נתונים הנכנסים למסד עלולים להיות בלתי תקינים ולגרום לשיבוש בתפקוד המערכת.

# פעולות עדכון\יצירה\ביטול (להלן: CRUD) עלולות להתבצע באופן המנוגד למדיניות מפעיל החניון ו\או לגרום לשיבוש המערכת.

# נתוני המנויים האישיים עלולים להיחשף לגורם שלישי שלא כחוק.

# תפעול רכיבי חומרה (חיישנים, מצלמות, נורות ומצגים)

# צפויות בעיות של כיול ותקשורת עם המערכת

# זרימת נתונים בין רכיבי המערכת:

# היות והמערכת מורכבת ממספר רכיבים נפרדים שמתופעלים בנפרד זה מזה, ישנו צורך באספקת מעבר הנתונים ביניהם וכך הן נוצרת תלות באיכות התקשורת והן נוצר סיכון שכמות הפניות בין הרכיבים עלולה ליצור עומסים.

# פתרונות אפשריים. (נא ציין פתרונות אפשריים וחלופות ארכיטקטוניות)

# ניהול מסד נתונים:

# פעולות ה-CRUD יתאפשרו למשתמש (מפעיל החניון) אך ורק באמצעות ממשק משתמש יעודי (להלן: GUI) כאשר הכנסת הנתונים תלווה במגבלות מובנות בהתאם לדרישות המערכת. הדבר ידרוש בין היתר ניהול הרשאות ובדיקות פורמט הנתונים הנכנסים.

# המערכת תשלב בתוכה מנגנוני אבטחת מידע לצורך הגנה על נתוני המשתמשים ועל התפקוד התקין שלה באופן כללי

# תפעול רכיבי חומרה:

# יש לדאוג שרכיבי החומרה יתאימו לדרישות בשטח מבחינת ביצועיהם וכן לערוך ניסויים בתנאים המדמים ככל שניתן את המציאות כדי לוודא שתפקודם הינו תקין

# ניתוב עומסים בצד שרת:

# אמנם השרת יסופק ע"י צד שלישי שמתחייב לתפקודו אך במסגרת הכנת הפרויקט ניתן לדאוג לכך שהתקשורת תתבצע באופן שמקל על זרימת נתונים חלקה יותר.

# פתרון טכנולוגי נבחר:

# טופולוגית הפתרון - כלומר: פרישת המערכת , היכן יתבצע יישום המערכת (deployment), מרכיבי הפרישה. הנ"ל ברמת מערכת (לדוג' פרויקט פיתוח אתר אינטרנט: המערכת מורכבת משרת, ממשק משתמש בצד הלקוח,, DB’s, טווח תקשורתי-אינטרנט, המערכת תיושם ברשת האינטרנט , יש להציג את דיאגרמת המערכת וכו')

# המערכת מורכבת מצד לקוח, צד שרת, מסד נתונים ומערכת אלקטרונית (ארדואינו) שתשלוט בצד הפיזי.

# חיישני המערכת ישלחו בקשות לשרת על מנת לאחסן נתונים במסד הנתונים ועל מנת לשלוח ללקוח או למפעיל החניון התראות וכו׳.

# טכנולוגיות בשימוש

# Arduino:

# אנחנו צריכים במסגרת הפרויקט לתפעל רכיבי וחיישני אלקטרוניקה מסוימים. הדרך בה אנו יודעים לעשות זאת היא דרך שפת ארדואינו.

# React:

# בחרנו בפריימוורק של ריאקט מכיוון שלדעתנו זה יקל עלינו לייצר את ממשק המשתמש. בנוסף לכך, במידה ונרצה בעתיד לפתח אפליקציה בנוסף לאתר, נוכל לעשות זאת בעזרת React Native.

# Node JS:

# זוהי סביבת שרת התומכת בעבודה שוטפת מול ריבוי ארועים הצפויים בפעילות חניון. כמו כן שפה זו מתממשקת באופן נוח עם שכבות המשתמש (React), מסד הנתונים (SQL) וחומרה (Arduino).

# SQL:

# כמות השאילתות שנצטרך במסגרת הפרויקט היא יחסית גבוהה לכן החלטנו שיהיה יותר יעיל לעבוד עם SQL בשל הנגישות ליכולות המובנות להפעלת שאילתות מרובות ומורכבות.

# שפות הפיתוח:

# Javascript: שפה זו מאופיינת בהתאמתה לפיתוח של כלים הפועלים באמצעות האינטרנט, נתמכת ע"י כל הדפדפנים הגדולים. השפה מאפשרת פיתוח של צד לקוח וצד שרת תוך שימוש בהרחבות וסביבות המתאימות לכך (React ו- NodeJS)

# C++: כתיבת שכבת ניהול החומרה. השפה תומכת מחד בשימוש באובייקטים, פקודות, מבני נתונים מורכבים ומאידך אינה דורשת משאבי מחשוב רבים כמו שפות עיליות ומאפשרת התנהלות יעילה יותר בזמן אמת בשל "קרבתה" לחומרה.

# SQL: כתיבת שאילתות במסד הנתונים. לשפה פונקציונליות מסעופת ומיוחדת המתאימה במיוחד לתפעול מבני נתונים והקשרים ביניהם.

# תיאור הארכיטקטורה

# בחרנו לעבוד בסביבה אינטרנט עם הפריימוורק של ריאקט ועם ארדואינו משום שנדרשת תקשורת בין הלקוח למפעיל החניון.

# גם מפעיל החניון וגם הלקוח צריכים לקבל התראות/ על כניסת רכבים, חניות לא תקינות וכו׳.

# חלוקה לתכניות ומודולים

# שכבת ממשק משתמש:

# צד לקוח, ייושם באמצעות REACT, מתחלק שני תת-מודולים הבאים:

# ממשק מנויים - תצורת יישומון למכשירים ניידים. תמיכה ביכולות הנ"ל בהתאם למגבלות שייקבעו ע"י המפעיל

# הוספה\עדכון\ביטול: פרטי מנוי, רכבים הרשומים על המנוי, הזמנות חניה,

# חיפוש: תאיי חניה פנויים, מקום בו הוחנה הרכב

# התראות: הודעות מערכת

# הפקת דוחות

# ממשק מפעיל - תצורת תכנת שולחן. תמיכה ביכולות הנ"ל:

# הוספה\עדכון\ביטול: תאי ואזורי חניה, מנויים, פרטי מנויים, רכבי המנויים, הזמנות חניה

# חיפוש: כל רכיבי מסד הנתונים

# התראות: תקלות בשכבת החומרה, בקשות עדכון מהמנויים, הפרות השימוש בחניון

# הפקת דוחות

# שכבת צד שרת:

# תיושם באמצעות NODE JS. ניתן לחלק את הפונקציונליות לפי התממשקות עם השכבות השכנות

# שכבה זו תקבל פניות מהשכבות של ממשק המשתמשים וניהול החומרה ותתרגם אותן לשאילתות עדכון\אחזור שתישלחנה למסד הנתונים. התשובות שתתקבלנה ממסד הנתונים, יישלחו לשכבות האחרות.

# תכונות ייחודיות לממשקים השונים:

# התקשורת מול צד לקוח תכלול גם אימות בקשות מהמנויים, בהתאם למדיניות שתיקבע

# חיבור ל-API שיאפשר תקשורת מול מכשירי סלולר

# בצד המתקשר מול ניהול החומרה יתקבלו נתוני הקלט של חיישנים ומצלמות שיועברו למסד הנתונים. כמו כן יתקבלו הוראות ממסד הנתונים (שמקורן בממשק המשתמש) ויישלחו לרכיבי הפלט של החומרה.

# שכבת מסד נתונים:

# תיושם באמצעות SQL. אחסון כל פריטי המידע הרלוונטיים, קבלת עדכונים (CRUD), תמיכה בבקשות מידע ובהפקת דוחות.

# שכבת ניהול חומרה:

# תיושם באמצעות ARDUINO. השכבה תתמוך בתקשורת מול צד השרת תוך שליחת נתוני הקלט וקבלת נתוני להפעלת רכיבי הפלט.

# בשכבה זו תתבצע בקרת תקינות של רכיבי החומרה וכן של כל השכבה. נתונים אלו יישלחו למסד הנתונים (דרך צד שרת).

# קריאת מספרי זיהוי של הרכבים תתבצע באמצעות חיבור ל-API ייעודי.

# חלק מרכיבי הפלט יופעלו ישירות ע"י הנתונים שיתקבלו מרכיבי הקלט. למשל חיישן חניה תקינה יפעיל ישירות את נורת האזהרה המתאימה, כך גם נורת תא תפוס\פנוי.

# רכיבי קלט:

# מצלמות לזיהוי מספרי לוחית,

# חיישני חניה תקינה (יותקנו באזורי הגבול בין תאי חניה)

# חיישני תנועה להפעלת מצלמות - בכניסה, יציאה ובכל תא חניה

# רכיבי פלט:

# שערי הכניסה והיציאה

# נורות סימון חניה תפוסה\פנויה\שמורה\מושבתת

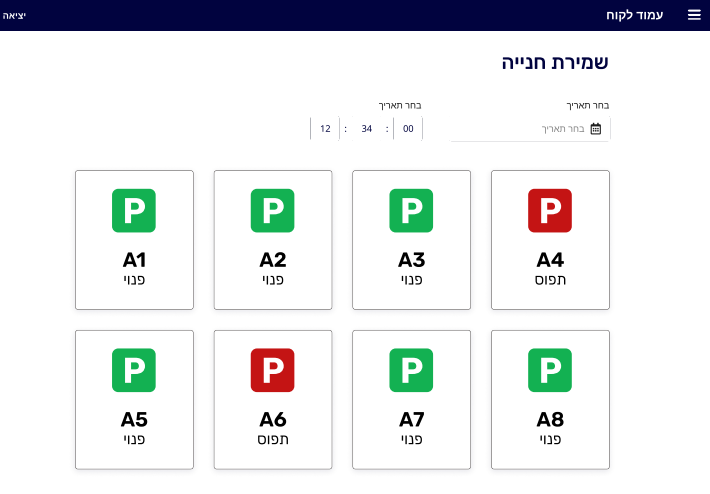
# נורות אזהרה על חניה בלתי תקינה

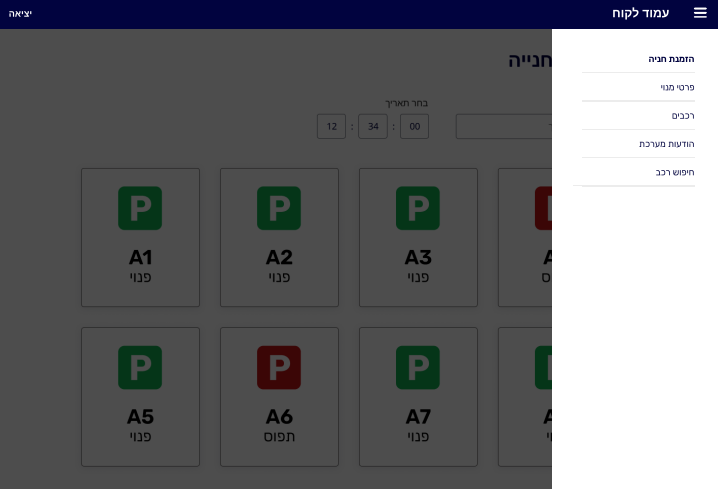
# מסכים עם נתוני התפוסה בכל אזור

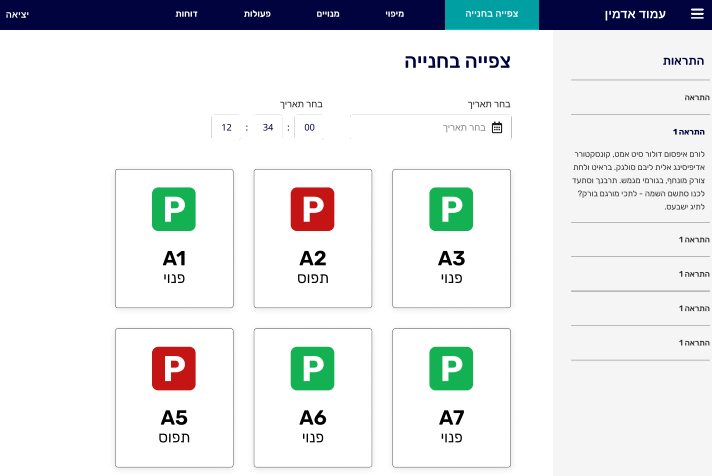
# סביבת השרת

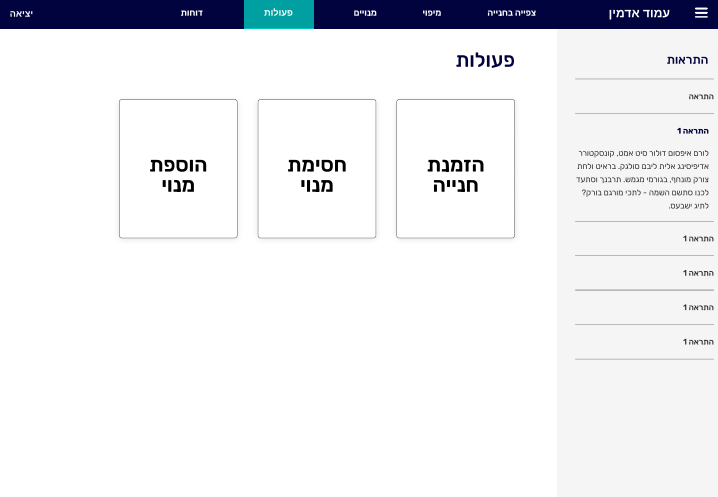
אנו נשתמש בשרת שיתופי והארדואינו יהיה מקומי בשטח הלקוח.

# ממשק המשתמש/לקוח - GUI









מפעיל החניון: ממשק אשר נותן גישה לדוחות, ניהול החניון שליחת התראות

לקוח: ממשק אשר מראה מקומות פנויים בחניון, נותן אופציה לשמור מקום חניה, הצגת פרטי מנוי ואפשרות ליצור/ לחדש מנוי.

# ממשקים למערכות אחרות /:API

נצטרך להשתמש ב API חיצוני שקולט מספר מתוך תמונה ומחזיר אותו כמחרוזת.

שירות API נוסף יאפשר שליחת מסרונים למכשירי סלולר.

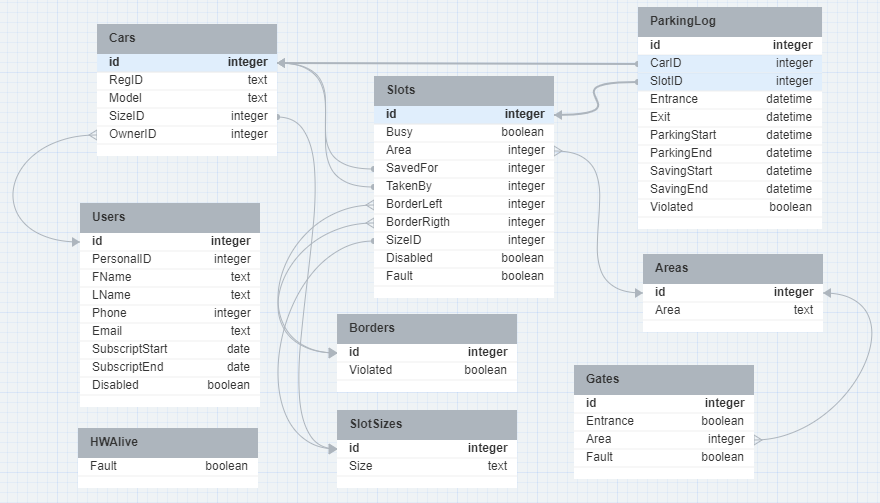
# שימוש בחבילות תוכנה:

# צד הלקוח ייכתב באמצעות React שזוהי ספריה של Javascript המיועדת ליצירת ממשקי משתמשים.

* צד השרת ייכתב באמצעות NodeJS: סביבת פיתוח המנוהלת בשפת Javascript ומאפשרת עבודה גם מחוץ לדפדפן.
* שכבת ניהול החומרה תיכתב באמצעות Arduino: חבילה הכוללת IDE ושורת הקוד, מהווה הרחבה של שפת ++C ומאפשרת תכנות וניהול של רכיבי חומרה ייעודיים.
* מסד הנתונים יתבצע בסביבת mySQL המציע כלי פיתוח בתקן SQL כולל כתיבת קוד ועיצוב טבלאות ושאילתות.

# שימוש במבני נתונים וארגון קבצים

# פירוט מבני נתונים.



# שיטת האחסון

האחסון יתבצע על גבי שרת שיתופי הן בקבצים התומכים בריצת התכנה והן במסד נתונים ייעודי

# מנגנוני התאוששות מנפילה\קריסה\תמיכה בטראנזקציות.

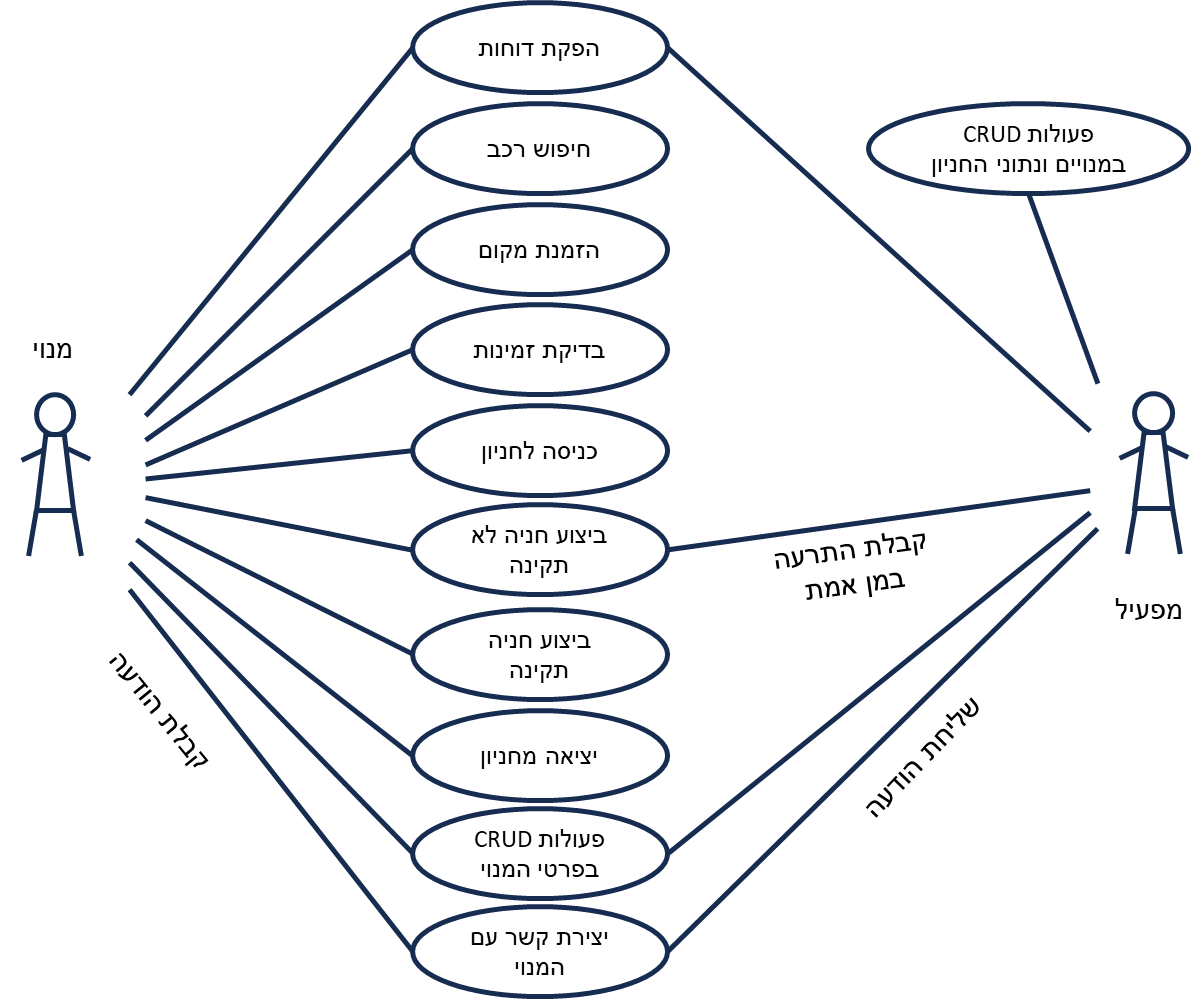
# נתוני מסד: יגובו כחלק משירות המוצע ע"י ספק השרת. בהסכם השירות יש לוודא כי מתקיים גיבוי המתקבל על הדעת (דוגמת שכפול נתונים - Mirroring או Duplication).

# רכיבי חישה שמשדרים את הנתונים מהשטח: יש לוודא כי התכנה בודקת באופן שוטף (אחת למספר שניות) האם הרכיב משדר אות כפי שצפוי

# רכיבי חישה שנמצאו בלתי תקינים יתוקנו או יוחלפו

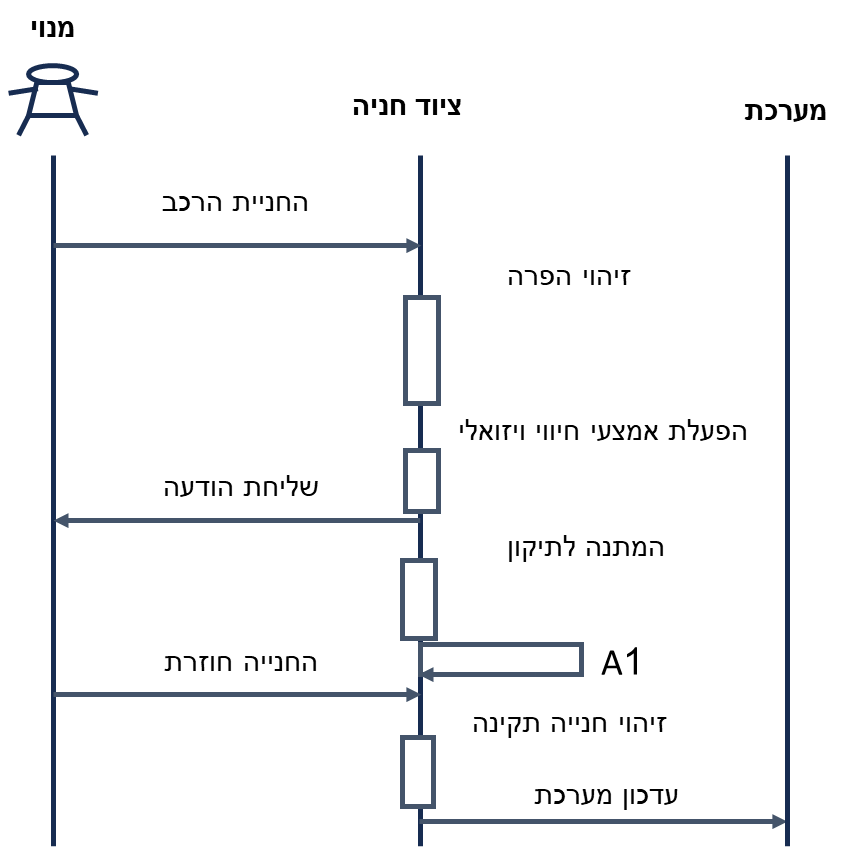
# תרשימי מערכת מרכזיים

# Use Case אופן השימוש

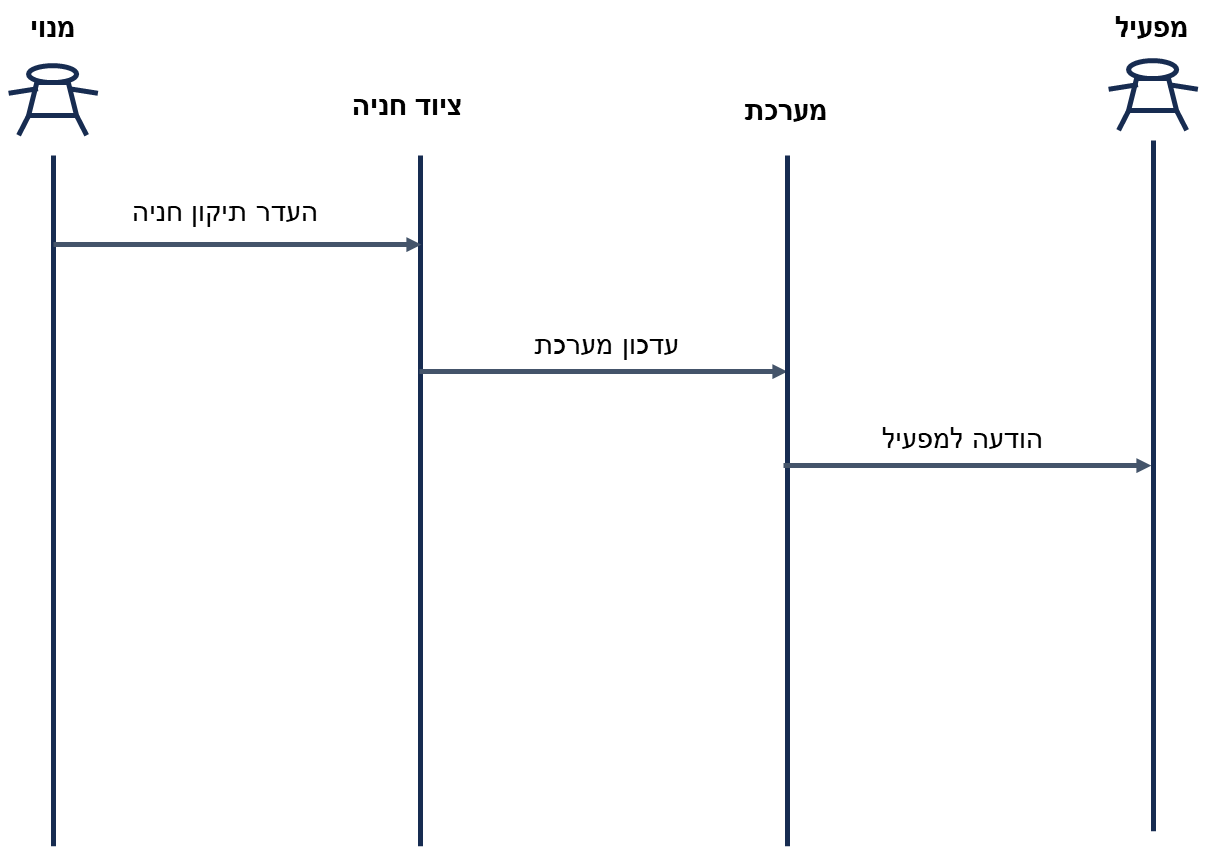


# Sequence diagram - רצף קריאות פונקציות מרכזיות בלוגיקה העסקית המרכזית של הפרויקט

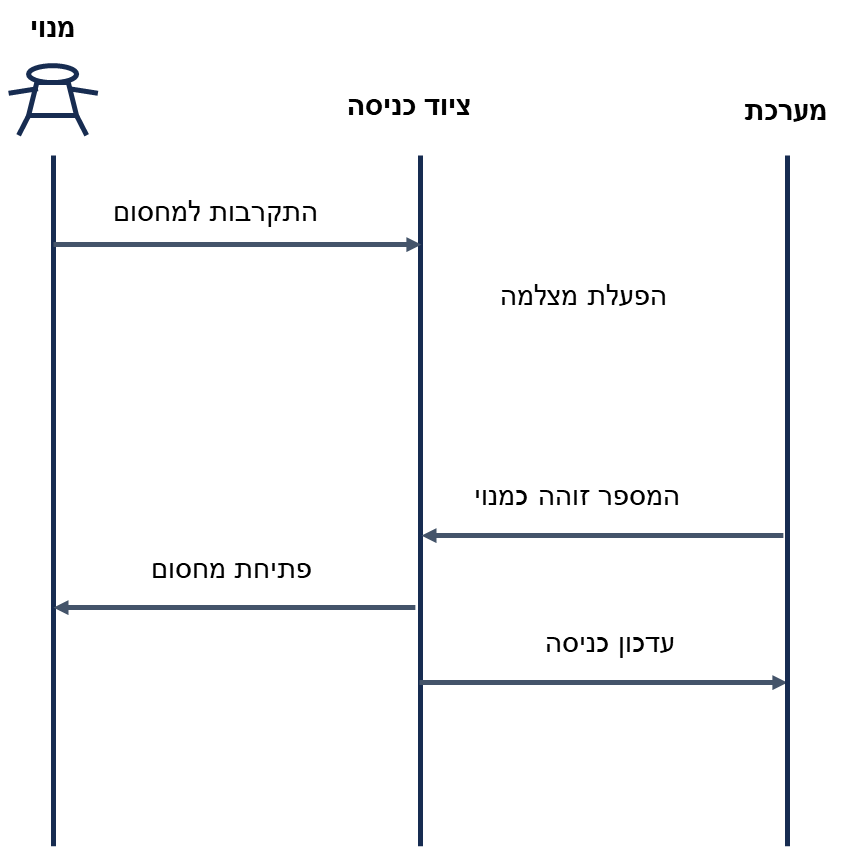
א. חניה בלתי תקינה



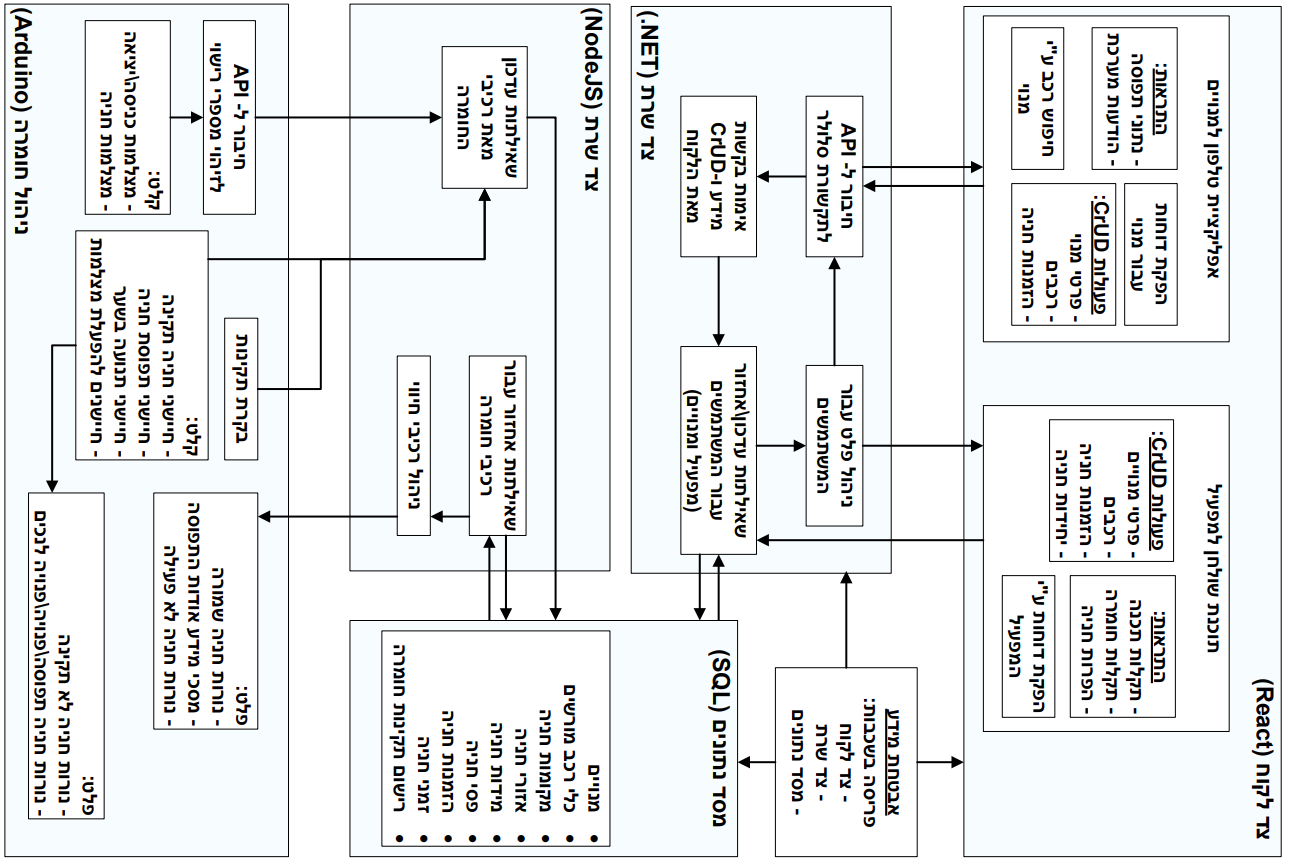
תרחיש A1:



# ב. כניסה לחניון:



# Data flow



# תיאור המרכיב האלגוריתמי – חישובי

לא יהיו אלגוריתמים בקוד שלנו.

# תיאור/התייחסות לנושאי אבטחת מידע

# אזורים הדורשים אבטחה:

# רכיבי קלט בצד הלקוח (תיבות טקסט, URL, תיבות להזנת שם משתמש וסיסמא) אשר עבור חלקם הבקרה תתבצע בצד הלקוח ואחרים - בצד שרת. למקרה שלמרות המאמצים, ייחשף מסד הנתונים לגורם זדוני, יש לדאוג שהמידע הרגיש יוצפן בו מראש (ר' בהמשך).

# אמצעי הגנה מפני תרחישים שונים:

# יש למנוע אפשרות של הזנת קוד זדוני ע"י משתמש בתגיות HTML (התקפות מסוג Reflected/Cross-Site Scripting) באמצעות שימוש בפונקציות ייעודיות כגון htmlspecialchars

# אם יוחלט במהלך הפרויקט להשתמש בעוגיות (cookies), יהיה צורך להתכונן לתרחיש "הרעלת עוגיות" (Cookie Poisoning) ע"י מניעה משמירה בתוכן מידע רגיש מבלי להצפין אותו.

# יש להגביל מצד שרת את כמות ניסיונות הזדהות על מנת להדוף את התקפות Brute Force. יש לבחון אפשרות לאלץ את המשתמש להפעיל הזדהות כפולה.

# בכל התחברות יקבל המשתמש "אסימון" (token) אשר ילווה כל בקשת לקוח שתישלח לשרת מאותו משתמש כדי למנוע מתקפה מסוג Cross Site Request Forgery

# את ססמאות המשתמשים יש לשמור תוך הצפנתם באופן בלתי ניתן לפענוח (דוגמת שיטת md5 או דומות לה) עם תוספת תחילית\סיומת שרירותית "Salt" בכדי להקשות על מאמצי הפענוח.

# משאבים הנדרשים לפרויקט:

# מספר השעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות

# היקף העבודה יעמוד על 600 שעות.

# איתן יעבוד על צד חומרה (Arduino) ומסד נתונים (SQL) וישתתף בפיתוח צד שרת (NodeJS) המקשר בין השכבות הללו.

# ישי יפתח את צד הלקוח (React) וצד השרת (NodeJS) הנושק אליו.

# ציוד נדרש - מחשב, ערכת ארדואינו, ציוד הדמיית החניון וכלי הרכב.

# תוכנות נדרשות - Arduino, Visual Studio Code, MySQL

# ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט - ריאקט, קלאוד, ניהול רכיבי חומרה שלא נעשה בהם שימוש במהלך הלימודים, התממשקות לAPI.

# ספרות ומקורות מידע - stack overflow, legacy.reactjs.org, udemy, nodejs.org

# תכנית עבודה ושלבים למימוש הפרויקט

בניית השלד של מסד הנתונים (איתן)

תכנון והרמת השרת:

הקוד בתווך בין מסד הנתונים לבין שכבת ניהול חומרה (איתן)

הקוד בתווך בין מסד הנתונים לבין שכבת צד לקוח (ישי)

תכנון והרמת צד לקוח (ישי)

תכנון והרמת שכבת ניהול חומרה (איתן)

# תכנון הבדיקות שיבוצעו

# נא פרט בטבלה, בדיקות תהליכיות ברמת משתמש בהן נדרשת המערכת לעמוד (full Flow).

תרחישים:

משתמש מזדהה ונכס למערכת או לוחץ על כפתור "דף ראשי" אם הוא כבר בתוך המערכת.

אם הרכב כבר נמצא בחניה, מקום החניה מוצג בדף הראשי

אם הרכב טרם הוחנה, מוצגת מפת החניון עם סימון תאי חניה פנוים\בלתי פנוים נכון לרגע זה

משתמש לוחץ על כפתור "הזמנת חניה"

נפתח דף ההזמנות

המשתמש בוחר תאריך\שעה של תחילה וסוף ההזמנה

תוצג מפת החניון עם סימון תאי חניה פנוים\בלתי פנוים נכון לזמן שנבחר

בלחיצה על תא חניה פנוי יירשם תא החניה כמוזמן עבור משתמש זה.

משתמש עובר לאזור אישי

פרטי המשתמש מוצגים בדף של האזור האישי

בלחיצה על עדכון פרטים יוצגו הפרטים כאינפוטים של טופס

משתמש עובר לאזור דוחות

תוצג היסטוריית חניות

לצד זה יוצגו אפשרויות של סוגי דוחות אחרים: הפרות, הזמנות

תוצאות צפויות:

בכל שלב המשתמש יוכל לעבור בין דפי האתר בלחיצה אחת וללא מאמץ מיוחד

פרטי החניות, התפוסה, פרטי הלקוח וכד' צריכים להיות מדויקים, קריאים ונכונים

הזמנות צריכות להתבצע בהצלחה ובהתאם לאפשרות שנבחרה. התא הנבחר יוצג כמוזמן ללקוח זה.

יש להפעיל את האתר בדפדנים שונים ולוודא שכל התכונות פועלות באופן זהה בכל הדפדפנים.

יש לוודא תפעול שגיות, למשל אם הלקוח מבצע הזמנה לא נכונה

יש לבדוק שזמני התגובה הינם סבירים.

# נא פרט בטבלה, מס מייצג של בדיקות יחידה למודולים המרכזיים בהן נדרשת המערכת לעמוד. (unit test)

סריקת לוחיות רישוי- נדרשת בדיקה שמוודאת שה API מחזיר מספר רכב תקין.

נדרשת בנוסף בדיקה שה API בכלל מקבל תמונה תקינה.

נדרשת בנוסף בדיקה שמוודאת שנשלחת תמונה רק במידה ורכב נכנס לחניון.

תיאור חניות פנויות באתר- נדרשת בדיקה שתוודא שהאתר יציג תאי חניה שאכן פנויים.

שליחת התראות- נדרשת בדיקה שתוודא שאכן יוצאת התראה ברגע של חנייה לא תקינה (לוודא שההתראה מתקבלת)

-בדיקה שמוודאת אחסון לוחית רישוי תקינה במסד הנתונים, או כל אינפוט אחר לצורך העניין (ללא אותיות וסמלים למשל+באורך מסויים).

-בדיקה לשליפת נתוני לקוח ממסד הנתונים (לוודא שאכן מגיעה תשובה)

-בדיקה לחיישני החניה: החיישנים אמורים להצביע על חניה תקינה רק במידה והרכב עומד על קו מוגדר מראש שעליו נמצא החיישן.

# בקרת גרסאות ( version control )

# הגרסאות ינוהלו באמצעות עבודה שוטפת מול GitHub

###### \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

###### חתימת הסטודנט חתימת המנחה האישי

1. **הערות ראש המגמה במכללה**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **אישור ראש המגמה**

## שם: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה: \_\_\_\_\_\_\_\_\_\_\_\_ תאריך: \_\_\_\_\_\_\_\_\_\_\_\_\_

1. **הערות הגורם המקצועי מטעם מה"ט**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **אישור הגורם המקצועי מטעם מה"ט**

## שם: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה: \_\_\_\_\_\_\_\_\_\_\_\_ תאריך: \_\_\_\_\_\_\_\_\_\_\_\_\_

###### 

**נספח להצעת הפרויקט:**

**קווים מנחים בבחירת נושאים והיקפי עבודה בפרויקט הגמר.**

1. **דגשים ארכיטקטוניים ושיקולים במימוש:**
   1. מומלץ להתנסות בארכיטקטורות השלבות שימוש בתצורת שרת לקוח.
   2. שימוש ב- design patterns במודולי התוכנה השונים- באיזורים מתאימים.
   3. דגש על הפרדה בין לוגיקה עסקית השייכת לצד לקוח וצד שרת.(FrontEnd,Backend ,ServerBL ,ClientBL)
   4. חלופות ארכיטקטוניות נדרשות לתמוך או לספק מענה לדרישות כגון:
      1. תמיכה והתמודדות בוויסות עומסים .
      2. תמיכה והתמודדות עם שיקולי אבטחה והגנה על מידע.
      3. תמיכה בשרידות והתאוששות מתקלה(טרנזקציות שמירה למאגר, והתאוששות)
   5. תמיכה בשיקולי חווית משתמש (צד מנהל מערכת וצד משתמש קצה)
   6. תמיכה היכן שניתן בניהול פרופילי משתמשים.
2. **שפת מימוש הפרויקט-**   
   ישנו משקל גבוה במימוש הפרויקט ביותר משפת מימוש אחת לפרויקט , תוך מתן דגש ליתרונות היחסיים של כל שפה, עבור מודול תוכנה במכלולי הפרויקט.   
   למשל במקרה של תצורת שרת לקוח (אתר אינטרנט):
   * 1. לצד הניהול העיסקי של השרת, בחירה בשפות עיליות JAVA, C# או nodeJS.
     2. לתכולה חישובית\ אלגוריתמית- מימוש בשפת native נניח C, C++.
     3. לצד לקוח AngularJS,Asp.net וכו'
3. **מאגר נתונים Database:**
   1. ישנה חשיבות גבוהה להתנסות בעבודה עם מאגרי נתונים למשל, מאגר רלציוני ומאגר FS Based.   
      למשל:
      1. עבור מאגר רלציוני נבחר ב- Sql server, Sqlite, etc’
      2. עבור מאגר לא רלציוני נבחר ב- mongoDB או NoSql.
   2. ישנה חשיבות רבה להגדרת שכבת גישה למאגר הנתונים כזו שתנהל מרכיבים טרנזקטיביים וסנכרון. נין להשתמש גם במסגרת frameworks קיימים כדוגמת dotNet.
4. **מרכיב אלגוריתמי\ חישובי-**  
   ישנה חשיבות רבה להתנסותו של התלמיד והתמודדותו עם יכולות חישוביות במסגרת מכלולי הפרויקט. ניתן לשלב היבטים אלגוריתמיים או לחילופין ניתוחים וחיתוכים סטטיסטיים בסיסיים מעל מאגר נתונים, למשל:
   1. במקרה של אתרים כניסת משתמשים, גיאומטריה חישובית וכו'.
5. **בדיקות תוכנה:**
   1. יש לגזור מדרישות המוצר אוסף בדיקות שיופרדו לפחות לשתי קטגוריות מרכזיות ויכסו את מרבית הקוד:
      1. בדיקות יחידה (Unit-Test)- אלה הן אותן בדיקות אותן יממש המפתח ברמת פנים מכלולי התוכנה ועד לרמת הפונקציות הציבוריות באותם מכלולי תוכנה.
      2. בדיקות תהליכיותFull Flow) – הללו יעסקו בעיקר בבדיקות בקשר שבין מכלולי תוכנה מרכזיים ויבחנו את הפונקציונאליות האינטגרטיבית של המוצר, מקצה לקצה.
      3. גישות מבורכות לתהליך ניתן לאמץ מתוך גישות שונות
      4. למשל:
         1. code a little test a little
         2. , test driven development,
         3. Regression Tests

**פרקים מרכזיים נוספים לספר הפרויקט:**

# דרישות מערכת ופונקציונאליות

# הנחות יסוד

# הנחות יסוד הקשורות לסביבה הטכנולוגית ולתפקוד .

# דרישות מערכת סביבת הטמעה ושימוש. שרידות, ביצועים\התמודדות עם עומסים.

# דרישות פונקציונאליות. רשימת דרישות המשתמש מהמערכת, מהן הפעולות בהן נדרשת המערכת לתמוך.

* 1. **חלופות ארכיטקטוניות-**   
     דיון בבחירת ארכיטקטורות ברמת המערכת וברמת מכלולי התוכנה.
     1. ברמת המערכת- חשוב להציג בספר הפרויקט בחינה של מספר חלופות ארכיטקטוניות (לפחות 3) בהן יבחן התלמיד את האפשרויות השונות תחת מספר אילוצים רלוונטיים נשוא הפרויקט, בין יתר השיקולים ניתן לשקול: התמודדות עם עומסים וויסותם, שרידות, יכולת מימוש, זמינות טכנולוגית ועוד. יש לשקול שילוב במקומות המתאימים של design patterns מקובלים, הן ברמת המערכת והן ברמת מכלולי התוכנה.
     2. ברמת מכלולי התוכנה- ניתן להציג דיון חלופות מצומצם יותר, אך לשקול בחיוב שילוב של design patterns מקובלים במקומות המתאימים (להימנע משימוש מיותר).

1. **טופולגיית הפתרון הנבחר**- הצגה סכימתית של פרישת המערכת.

# ארכיטקטורה נבחרת: הצגה בגישת UML את פריטת מרכיבי הפתרון בחלוקה למכלולי תוכנה ראשיים ומשניים כמו-גם הדיאגרמות הרלוונטיות.

# שימוש במבני נתונים וארגון קבצים

# נא פרט את מבני הנתונים.

# נא פרט את שיטת האיחסון (מאגר, קבצים ובאיזה טכנולוגיה)

# נא ציין מנגנוני התאוששות מנפילה\קריסה\תמיכה בטראנזקציות.

# תרשים זרימת המידע במערכת

# Use Cases

# Sequence diagram

# Data flow

* 1. **חלופות שפת מימוש-**   
     במסגרת ספר הפרויקט חשוב להציג בחינה של מספר חלופות עבור שפה\ות מימוש הפרויקט. הנ"ל צריך לכלול דרישות אותן יגדיר התלמיד בבחירת השפה המתאימה. בין יתר השיקולים ניתן לכלול: זמני ריצה, היבטי אבטחה והגנה, הגנה על זכויות יוצרים (בינארי או interpreter, קלות במימוש, התאמה לממשקי משתמש או צד שרת) וכו'.

1. **חלופות אבטחתיות והגנה –**   
   במסגרת ספר הפרויקט יש לבחון לפחות 3 חלופות אבטחתיות להגנה ושמירה על נתונים, יש לכלול התמודדות עם מקרים ותגובות בתהליך הבחינה (לפחות 10 מקרים).
   * 1. יש לשים לב- לפרוטוקולי תקשורת, http ,https ,ssl###,
     2. מכלולי תוכנה צד שלישי (אנטי וירוס,מצפינים,firewalls וכו').
     3. ברמת הקוד- ווידוא סכימות הודעות בין מכלולי התוכנה ובתקשורת בניהם.
   1. **פירוט בדיקות תוכנה ואופן ביצוען (STP-DOC) -** לכלול את רשימת בדיקות התוכנה, בדיקות יחידה, בדיקות תהליכיות- full Flow במסגרת מסמך תכנון בדיקות ובדיקות. הנ"ל יוצג בטבלה : תיאור הבדיקה, תוצאה רצויה, תוצאה מתקבלת.

**ניתן לאמץ את נוהל מפתח באופן מושכל!** [**http://www.methodacloud.com/content/pages/kit\_maxsum/H\_Guide-map.asp**](http://www.methodacloud.com/content/pages/kit_maxsum/H_Guide-map.asp)