לכבוד

יחידת הפרויקטים מה״ט

הצעה לפרויקט גמר

א. פרטי הסטודנטים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תאריך סיום הלימודים | טלפון נייד | כתובת | ת.ז. 9 ספרות | שם הסטודנט |
| תמוז תשע"ח | 0583227453 | בגין 96 פתח תקוה | 206503070 | נאוה אייזנשטיין |
| תמוז תשע"ח | 0548494053 | בגין 96 פתח תקוה | 315163659 | אביגיל פדר |

שם המכללה: המכללה למנהל ראשון לציון

סמל המכללה:

מסלול ההכשרה: הנדסאים

מגמת לימוד: תכנות מחשבים

מקום ביצוע הפרויקט: בסמינר/בבית/במקום העבודה

ב. פרטי המנחה האישי

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מקום עבודה/תפקיד | תואר | טלפון נייד | כתובת | שם המנחה \* |
|  |  |  |  | המורה ברגמן |

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

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

דרך מנחם בגין 86 תל אביב ת.ד. 36049 מיקוד 67138 טלפון :03-7347521 פקס: 03-7347644

1. שם הפרויקט : Dynamic Turn
2. רקע:
   1. תיאור ורקע כללי:

הפרויקט מציג אתר לניהול מערכת חכמה של תורים. האפשרות לקביעת התורים היא נוחה, מהירה ודינמית.

משתמש יכול לקבוע, לשנות ולבטל תור לעצמו, ללא המתנה לנציג שירות.

בנוסף, משתמש יכול להיכנס לפורום של כל בעלי התורים, ולמצוא לעצמם פרטנר להחלפת התור, בשביל שקביעת שהתור תהיה חוויה מהירה ונוחה, והתור הנקבע יהיה התור האופטימלי ביותר.

* 1. מטרות המערכת:

רשימת מטרות עיקריות של המערכת, ממוספרת או בתבליטים.

* המערכת תספק אתר לניהול תורים לארגונים שונים.
* המערכת תציע למשתמשים רשימת יעדים לקביעת תורים.
* המערכת תאפשר למשתמשים לחסוך בזמן ולקבוע את התור לעצמם בעצמם.
* מהערכת תאפשר למשתמשים לשנות את התור לעצמם לפי הצורך.
* המערכת תאפשר למשתמשים לתקשר ביניהם תוך שמירה על אתיקה מקצועית ודיסקרטיה. המשתמשים יוכלו להחליף ביניהם את התורים, ליידע אחד את השני על עיכובים במערכת ולהחליף רשמים.
* המערכת תאפשר למנהל לנהל את התורים שקבעו הלקוחות, למחוק ולשנות את כל התורים לפי הצורך .

1. סקירת מצב קיים בשוק, אילו בעיות קימות.

קיימים אתרים רבים ומערכות שונות לניהול וארגון תורים, אבל תכנון תורים דינמי כמו בפרויקט שלנו- לא מצאנו.

1. מה הפרויקט אמור לחדש או לשפר.

הפרויקט נותן אפשרות של דינמיקה ותקשורת בין הלקוחות, דבר שמערכות רגילות לא מספקות, והמענה האנושי לא מקיף.

1. דרישות מערכת ופונקציונאליות
   1. דרישות מערכת, סביבת הטמעה ושימוש.

* המערכת תעבור קומפילציה והפצה בסביבת eclipce.
* בצד השרת המערכת תרוץ בסביבת שרת אשר מריץ tomcat\_8 לקבלת בקשות לנתונים שיוטמעו בדפי אינטרנט.
* המשתמש יוכל להריץ את האתר בכל מכשיר אשר מותקן עליו דפדפן אינטרנט.
  1. דרישות פונקציונאליות

רשימת דרישות המשתמש מהמערכת,

* המערכת תאפשר רישום חופשי של משתמשים חדשים.
* בדף הכניסה תהיה אפשרות לשחזור סיסמה.
* הכניסה למערכת תבוצע רק במקרה שתעודת הזהות והסיסמא קיימים במערכת בהתאמה.
* משתמש יכול לקבוע לעצמו תור לכל אחד מהיעדים.
* משתמש יכול לבטל ולשנות את התורים שלו.
* משתמש רשום יכול להיכנס לפורום ולהשאיר הודעות.
* משתמש יכול ללחוץ על כפתור 'החלף' ולהכניס מספר זוג להחלפה.
* במקרה שיש אישור מהזוג, ההחלפה בין שתי התורים מבוצעת.
* מנהל יכול לעקוב אחר התורים שנקבעו.
* עבור כל תור שנקבע נשלחת הודעה למנהל, והוא יכול לבטל אותה כרצונו.
* מנהל יכול לנהל את המשתמשים לפי שקול דעתו.

1. בעיות צפויות במהלך הפיתוח ופתרונות (תפעוליות, טכנולוגיות, עומס ועוד ):
   1. תיאור הבעיות- הללו כפועל יוצא של דרישות המשתמש מהתוכנה.

בעיה 1: משתמש יכול לקבוע לעצמו תורים ללא הגבלה.

* 1. פתרונות אפשריים. (נא ציין פתרונות אפשריים וחלופות ארכיטקטוניות)

לבעיה 1: פתרון 1?: הגבלה של מספר התורים ללקוח.

פתרון 2: מנהל צריך לאשר כל תור שנקבע.

פתרון 3 : מנהל מקבל הודעה על כל תור שנקבע ויכול לבטל את התורים כרצונו.

הפתרון הנבחר עבור כל אחת מהבעיות:

לבעיה 1: מנהל מקבל הודעה על כל תור שנקבע ויכול לבטל את התורים כרצונו.

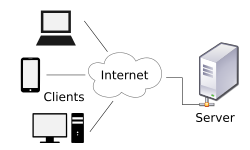
7. פתרון טכנולוגי נבחר:

1. טופולוגית הפתרון

המערכת מורכבת משרת TOMCAT\_8 המריץ את האתר בסביבת ה- server,

מסד נתונים - DB’s של sql-server

ממשק משתמש בצד הלקוח: דפדפן אינטרנט כלשהו: chrome, firefox , internet explorer,



1. טכנולוגיות בשימוש. (איזה ומדוע בכמה מילים).

צד הלקוח: Angular+typescript

שמוש בbootstrap4 ובספריות עצוב נוספות

צד השרת: שירות אינטרנט באמצעות web api בשיטת שלושת השכבות.

מסד נתונים: sql-server

1. שפות הפיתוח: (איזה שפות ומדוע בכמה מילים?)

שפות הפיתוח בצד השרת:JAVA

שפות הפיתוח בצד הלקוח: TYPE SCRIPT

JAVA SCRIPT

CSS

HTML5

1. תיאור הארכיטקטורה הנבחרת- הסבר בכמה מילים מדוע.

אנגולר היא טכנולוגיה מתקדמת המבוקשת מאד בתעשייה.

היא מונחית עצמים, מובנית, נכתבת בצורה ממודרת ומאורגנת. קל לאתר ולתקן בה שגיאות.

מצאנו לנכון שכדאי להתמקצע בה.

JAVA היא שפה חזקה, אמינה ובעלת ביצועים חזקים. חוצה פלטפורמות ומרובת תהליכים.

העבודה בה נוחה, מהירה ומקדמת.

1. חלוקה לתכניות ומודולים.

בצד השרת:

Web api– controllers המכילים שירותים שונים אשר ניתן לבצע אליהם קריאות מהלקוח.

Bissness logic-פונקציות המכילות חישובים לוגייים. (יתכן ונוותר על שכבה זו).

DAL – פונקציות הגישה למסדי הנתונים

Entities - ישויות המוגדרות מול מסד הנתונים.

בצד הלקוח:

Component-המערכת מחולקת לקטעים מסודרים, כל קומפוננטה מכילה קוד html, css, typescript

Service-מחלקות המשמשות לשירותים שונים עבור המערכת כמו גישה לserver

Model-מבנה המחלקות.

Module-ספריות שונות המיובאות לפרויקט.

1. סביבת השרת

לצורך הפרויקט נשתמש בשרת מקומי – tomcat\_8 , אשר משמש לסביבת הפיתוח eclipce. עבור לקוח מסוים, נרכוש עבור הפרויקט מקום בשרת אירוח.

1. ממשק המשתמש/לקוח – GUI.

ממשק המשתמש יעבור קומפילציה לשפות JAVA-SCRIPT, ,CSSHTML ויהיה מוצג באמצעות דפדפן אינטרנט .

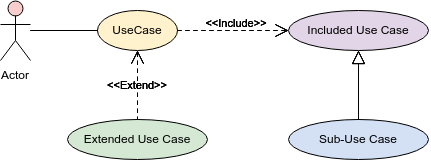
1. ממשקים למערכות אחרות / API :

לא רלוונטי

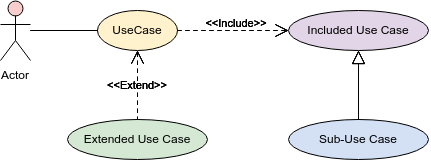
1. שימוש בחבילות תוכנה.
   * ECLIPCE-LINK
   * angularJS
   * AngularCore
   * Bootstrap
2. שימוש במבני נתונים וארגון קבצים
   1. נא פרט את מבני הנתונים.
      1. **טבלת לקוחות ומנהלים:** שם פרטי, שם משפחה, תעודת זהות, כתובת, מייל, קוד, האם מנהל
      2. **טבלת יעדים:** תעודת זהות, שם פרטי, שם משפחה, תאור, מייל
      3. **טבלת תורים:** מזהה לקוח, מזהה יעד
      4. **טבלת ימי עבודה:** מזהה יעד, יום בשבוע, התחלה, סיום
      5. **טבלת ימי חופשות:** מזהה יעד, תאריך
   2. שיטת האחסון

הנתונים ישמרו במסד נתונים של sql-server.

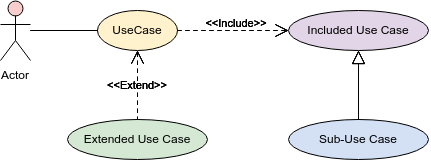
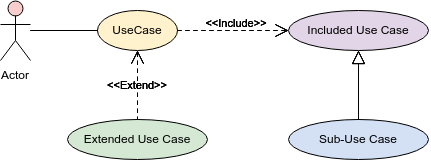
1. תרשימי מערכת מרכזיים
   1. Use Case



**manager**



**customer**



1. תיאור המרכיב האלגוריתמי – חישובי.
   1. איזה בעיה בא לפתור, איך יפתור?

בשביל ביצוע החלפה בין שתי תורים, יש צורך לבדוק שאכן שתי הצדדים מעונינים בהחלפה. לשם כך יש לשמור מבנה נתונים נוסף, שבו נשמור כל פניה להחלפה שהגיע למערכת. במבנה נשמור את מזהה התור של המבקש, ואת מזהה התור של בן זוגו. עבור כל הוספה למבנה הנתונים החדש, נבדוק האם קימת כבר רשומה שמכילה את בקשת ההחלפה מן הזוג. אם כן, נמחוק מן הטבלה את הרשומות ונבצע את ההחלפה במבנה הנתונים של התורים.

* 1. איסוף מידע וניתוחים סטטיסטיים (אנליטיקות)

ניתן לסכום את מספר התורים שנשארו סטטיים כפי שהיו בתחילה, ואת מספר התורים שהוחלפו על ידי משתמשי הפורום ולבצע השוואה יחסית בין התורים.

11. תיאור/התייחסות לנושאי אבטחת מידע

נא לציין אזורים הדורשים אבטחה, כגון: שרת, בקרת גישה לאתר, חשבונות משתמשים, מאגרי מידע וכיצד ניתן מענה.

יש לדאוג לאבטחת השרת מעומסים מופרזים הנגרמים יל ידי גורמים זדוניים.

מלבד זאת, יש לאבטח את המערכת מקביעת תורים רבים ומיותרים וזאת על ידי אישור מנהל.

מנהל המערכת – יקבל הרשאות שונות בשביל למנוע מכלל המשתמשים לעקוב אחר תורים אחרים, למחוק אותם ולחבל בהם.

1. משאבים הנדרשים לפרויקט:
   1. מספר שעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות.

עבור הפרויקט נדרשות כ700 שעות עבודה.

חלוקת העבודה תעשה לפי הצורך בין הקומפוננטות השונות של הפרויקט.

* 1. ציוד נדרש:

מחשב, מערכת הפעלה win10 ומעלה

* 1. תוכנות נדרשות

Eclipce, visual studio code

* 1. ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט

Observable בין שרת ללקוח עבור הפורום.

* 1. ספרות ומקורות מידע

<https://ej2.syncfusion.com>

<https://stackoverflow.com/>

<https://reshetech.co.il/>

<https://www.npmjs.com/>

<https://www.quackit.com/>

<https://github.com/>

[https://online.visual-paradigm.com](https://online.visual-paradigm.com/)

<http://www.zindell.com/books>

1. תכנית עבודה ושלבים למימוש הפרויקט:
   * ייזום הרעיון
   * ניתוח מערכת
   * ניתוח מבנה נתונים
   * איפיון UX - UI
   * כתיבת הלוגיקה העסקית
   * כתיבת ממשק המשתמש
   * עיצוב
   * בדיקות התוכנה
   * התקנה והטמעה
2. תכנון הבדיקות שיבוצעו
   1. נא פרט בטבלה, בדיקות תהליכיות ברמת משתמש בהן נדרשת המערכת לעמוד .(full Flow)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מספר בדיקה | מס` דרישה במסמך אפיון | מקרי הבדיקה | ידנית/ אוטומטית | חשיבות | הערות – תוצאת הבדיקה הרצויה |
| 1 | 5.3.1 | רישום למערכת מתוך דף הכניסה | ידנית | בינונית | בדיקה האם המשתמש יכול להיכנס למערכת באמצעות שם הרישום החדש. |
| 2 | 5.3.2 | ניסיון לשחזור סיסמה | ידנית | גבוהה | בדיקה האם נשלח מייל לשחזור סיסמה לכתובת המשתמש הרשום |
|  | 5.2.3 | ניסיון לקביעת תור | ידנית | גבוהה | בדיקה האם התור שנקבע אכן התווסף למסד ונשלחה הודעה למנהל. |
|  | 5.2.4 | ביצוע החלפת התור | ידנית | גבוהה | בדיקה שאכן התור הוחלף מול מסד הנתונים. |
|  | 5.2.5 | החלפה באישור שני צדדים | ידנית | גבוהה | בדיקה שההחלפה מתבצעת רק במקרה שבו שני הצדדים מבקשים את ההחלפה. |
|  | 5.2.6 | נהול תורים ע"י מנהל | ידנית | גבוהה | בדיקה שהמנהל רואה את כל התורים שנקבעו ויכול למחוק ולשנות נתונים |

* 1. ~~בדיקות יחידה (unit test)~~

~~-לא יתבצעו בדיקות יחידה אוטומטיות. [לא נכלל בתכנית הלימודים].~~

1. בקרת גרסאות (version control)

חתימת הסטודנט חתימת המנחה האישי

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

ג. הערות ראש המגמה במכללה

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

ד. אישור ראש המגמה:

שם: חתימה: תאריך:

ה. הערות הגורם המקצועי מטעם מה״ט:

ו. 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 השכבות שימוש בתצורת שרת לקוח.
   2. שימוש ב- desi
   3. gn patterns במודולי התוכנה השונים- באיזורים מתאימים.
   4. דגש על הפרדה בין לוגיקה עסקית השייכת לצד לקוח וצד שרת. FrontEnd,B ackend)

(ServerBL ,ClientBL,

* 1. 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(
  2. תמיכה בשיקולי חווית משתמש )צד מנהל מערכת וצד משתמש קצה(
  3. תמיכה היכן שניתן בניהול פרופילי משתמשים.

1. 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 C# ,JAVA או nodeJS.
2. לתכולה חישובית\ אלגוריתמית- מימוש בשפת native נניח ++C, C.
3. לצד לקוח AngularJS,Asp.net וכו'
4. 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 .
  1. ישנה חשיבות רבה להגדרת שכבת גישה למאגר הנתונים כזו שתנהל מרכיבים טרנזקטיביים וסנכרון. נין להשתמש גם במסגרת frameworks קיימים כדוגמת dotNet.

1. 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'.

1. 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 תהליכיות (Full 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:

code a little test a little .5.1.4.1

, test driven development, .5.1.4.2

Regression Tests .5.1.4.3

**פרקים מרכזיים נוספים לספר הפרויקט:**

1. דרישות מערכת ופונקציונאליות
   1. הנחות יסוד
      1. הנחות יסוד הקשורות לסביבה הטכנולוגית ולתפקוד .
   2. דרישות מערכת

סביבת הטמעה ושימוש. שרידות, ביצועים\התמודדות עם עומסים.

* 1. 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.
2. ארכיטקטורה נבחרת: הצגה בגישת UML את פריטת מרכיבי הפתרון בחלוקה למכלולי תוכנה ראשיים ומשניים כמו-גם הדיאגרמות הרלוונטיות.
   1. שימוש במבני נתונים וארגון קבצים
      1. נא פרט את מבני הנתונים.
      2. נא פרט את שיטת האיחסון (מאגר, קבצים ובאיזה טכנולוגיה)
      3. נא ציין מנגנוני התאוששות מנפילה\קריסה\תמיכה בטראנזקציות.
   2. תרשים זרימת המידע במערכת

Use Cases .7.2.1

Sequence diagram .7.2.2 Data flow .7.2.3

* 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) וכף.

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 וכף).
3. ברמת הקוד- ווידוא סכימות הודעות בין מכלולי התוכנה ובתקשורת בניהם.
4. 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 מושפל!

[http://www.methodacloud.com/content/pages/kit maxsum/H Guide-map.asp](http://www.methodacloud.com/content/pages/kit_maxsum/H_Guide-map.asp)