

פרויקט קורס הנדסת תוכנה

151060

כלים וטכניקות לתחילה פיתוח מערכת תוכנה

מטרת הפרויקט היא לדמות תחילך אמיתי של פיתוח תוכנה בצוות, משלב הדרישות ועד למסירה. דרך הפרויקט תתרגם עבודה בצוות, תכנון, פיתוח, בדיקות ושימוש בכלים מודרניים. המיקוד הוא לא רק בתוצר הסופי, אלא גם בדרך העבודה

תוכן

2	קווים כלליים על הפרויקט
2	מוטיבציה
2	מטרת הפרויקט
3	מבנה הקבוצות
3	שיטת עבודה
3	שימוש ב AI
4	סוג הפרויקט הנבחר השנה
4	תכנון כללי של שלבי הפרויקט
5	שלבי הפרויקט
5	שלב 1 – תכנון ודרישות (שבועות 1–2)
8	שלב 2 – בחירת ארכיטקטורה טכנולוגיות וכלי Sprint Planning כולל ראשוני (שבוע 3)
9	שלב 3 – ספרינט 1 מערכת בסיסית עבדת קצת לפחות (שבוע 4)
10	שלב 4 – ספרינטים נוספים (שבועות 5–10)
11	שלב 5 – בדיקות מערכת וטיפול באגים (שבועות 11–12)
11	הגשה סופית (שבוע 13)
12	נספחים
12	נספח 1: הערכת Story Points בשיטת Agile
14	נספח 2 - שיטת WScCoMo לקביעת עדיפות
15	נספח 3: תרשימים use case
16	נספח 4 תרשימי activity diagrams
17	נספח 5 - לוח מעקב בסיסי דוגמא
18	נספח 6 בונוסים
18	בונוס 1:
19	בונוס 2:
21	הרכב ציון

22	מתן הרשות גישה administrator
23	דרישות איקות

קוויים כלליים על הפרויקט

מטריבציה

פרויקט זה אינו רק משימה אקדמית; הוא סימולציה מודרנית של בעודתכם כהנדסי תוכנה בעולם האמיתי. כאן תנסו לא רק בכתיבת קוד, אלא בניהול פרויקט מוקצה לקצה – החל מרעיון ועד למוצר עובד – תוך שימוש בכלים ובمتודולוגיות המובילות יום בתעשייה. הצלחה בפרויקט תליה ביכולתכם לעבוד בצוות, לנוהל זמן, לקבל החלטות טכנולוגיות ולהתמודד עםאתגרים, בדיקות כדי שתעשו בתפקידם הבא. ברוכים הבאים למסע המעש שיכם בעולם הנדסת התוכנה.

מטרת הפרויקט

התנסות בתהיליך פיתוח תוכנה תוך יישום טכניות ושיטות ניהול משימות וביצוען, בשילוב גישות קלאסיות עם גישות זריזות (AGILE) וגישה DevOps. סביבות העבודה תהיה azure devops כמסגרת לפיתוח התוכנה כולל ניהול משימות, תיעוד ושימוש בכלים נוספים.

הפרויקט מתבצע בקבוצות של 4 סטודנטים. תוך שימוש דגש על שיתוף פעולה מלא וחילוקת אחריות רואיה.

בפרויקט זה נפתח מוצר תוכנה משלב הגדרת הדרישות, דרך תכנון ארכיטקטורה ובחירה טכניות, ניהול גרסאות ופריסת המערכת. נחווה את העבודה בצוות תוך שימוש דגש על איקות הקבוצה, טיפול ההתקדמות שלה, והיכולת לנוהל את הזמן וביצוע המשימות בצורה מושכלת ומתאימה ליכולות הקבוצה.

העבודה תתבצע כולה azure devops תוך שימוש ב- Boards, Repos, -Dashboard, Pipelines, Wiki

אנחנו לא מצפים למערכת מושלמת ברמה תעשייתית, אלא שתראו תהילך מלא – מאפיון, פיתוח, בדיקות והפקת ללקוח

מבנה הקבוצות

- 4 סטודנטים לכל קבוצה, כל קבוצה צריכה לשתף את המתרגל שלה בפרויקט.

שיטות עבודה

- כל הפרויקט יעשה תוך שימוש בסביבת הפיתוח של azure devops – azure devops הסביבה מספקת כלים לניהול ומעקב אחריו דרישות ועובדות הוצאות, תיעוד, ניהול גרסאות קוד ועוד.
- הפרויקט כאמור משלב בין שיטות קלאסיות לשיטות זריזות וdevops.
בתחילת יוגדרו הדרישות ונשתמש בדיagramot UML על מנת להבין את הדרישות. בהמשך נפתחת המערכת בשיטת e-Agile- קר' יהיו לנו 3 ספרינטים כל ספרינט ימשך שבועיים.
- כל תיעוד שיידרש למערכת יהיה בזווית-Wiki- תוך שמירה על סדר הגיוני וקריא של קבצי התיעוד
- במהלך הפרויקט תבוצע הערכת Story Points לכל Story . הנחה זו תיבדק לאחר ביצוע סיפור המשתמש תוך מטרה לדיק את עצמנו באומדן המשאבים הנדרסים לכל סיפור משתמש.
- במהלך כל הפרויקט מתחילה ועד סוף, הנהל מעקב אחר משימות לפי חתכים שונים ובאמצעות Диagramot ורשימות שיתעדכנו בכל פעם Dashboard
- השימוש בדיagramot UML בתחלת הדרך ועוד לתת לנו 'GBT-על' ול להבין את תמנונת המערכת הגדולה והאינטרקציות המרכזיות בה. לאחר מכן, לעבור לעובדה איטרטיבית ב e-Agile- כדי לפתח את המערכת בחלוקת קטנים, מה שמאפשר לנו גמישות, קבלת משוב מהיר ויכולת להתאים את עצמנו לשינויים.

שימוש ב AI

בכל שלב בפרויקט ניתן ומומלץ להשתמש במבנה מלאכותית. יש לבקר כל תוכרולה, ולהחליט האם הוא מתאים לפרויקט או לא. יש להבין ולשלוט בכל מה שנמצא בפרויקט. דרך השימוש במבנה המלאכותית תוכלו למדוד על טכנולוגיות ושיטות שיידרשו לכם לפרויקט, וכן לבצע משימות שיקלו מועליכם ויזרו את התקדמות הפרויקט. עם זאת על הסטודנטים לשלוט בפרויקט שליטה מלאה.

חלק מההגשה הסופית, יש לכלול ב-Wiki-דף בשם '**יום שימוש ב-AI**'. בדף זה, כל צוות יתעד 3-5 מקרים משמעותיים בהם נעשה שימוש בכל'י AI. עבור כל מקרה, יש לפרט: מה הייתה הבעיה/המשימה, באיזה כל' נעשה שימוש, מה היה ה-Prompt-המרכזי, כיצד התוצר של AI-קידם את הפרויקט (לדוגמה: יצירת קוד, כתיבת בדיקות, ניתוח דרישות, פתרון באג).

סוג הפרויקט הנבחר השנה

פיתוח אתר אינטרנט. ככל קבוצת תרגול המתרגל ידריך איזה אתר אינטרנט נדרש לפתח.

תיעוד

התיעוד הנדרש בפרויקט הוא תיעוד לפי הדרישות. חשוב מאוד לשמר על סדר וארגון קבצי התיעוד בצורה ברורה ונוחה למעבר. במהלך התיעוד יש להוסיף קישורים מתוך התיעוד לרכיבים azure.devops. אין צורך להאריך בתיעוד אלא שייהי בו את כל הנדרש להבנת הפרויקט.

התיעוד נדרש לעזרתכם להבין את המערכת ולתCKER בשנות. אל תכתבו מסמכים רק בשבייל הסימון, אלא השתמשו בהם כמצפן לפיתוח.

תכנון כללי של שלבי הפרויקט

שבועות 1–3 → דרישות + UML + Backlog

שבוע 4 → ארכיטקטורה + Sprint Planning

שבוע 5 → ספרינט ראשון → מערכת בסיסית.

שבועות 6–10 → ספרינטים נוספים → הרחבות.

שבועות 11–12 → בדיקות + באגים.

שבוע 13 → הגשה סופית + מדריך למשתמש/וידאו.

שלבי הפרויקט**שלב 1 – תכנון ודרישות (שבועות 1–3)**

- כל איש צוות צריך להירשם לazure devops , ואחד מארגוני הצוות פתוח ארגון ובונה פרויקט מסווג AGILE ומשתף את כל אנשי הצוות כולל המתרגל – מתן הרשאות של administrator לכל חברי הקבוצה ([ראה נספח 8](#))
- כתיבת מסמך ייזום מקוצר : תיעוד בואה של azure devops הכלל
 - عمود עבור "אודות" - יש לכתוב את שמות מלאים של המגישים עם תעודות זהות, אפשר להוסיף פרטים נוספים אודות המגישים.
 - عمود עבור מסמך הייזום המקוצר – כולל
- **cotreta** - שם הפרויקט, תאריך וסטודנטים (מספר זהות ושם).
- **תקציר** - תיאור קצר וממצה של הפרויקט, כולל המטרה העיקרית וקהל היעד.
- **רקע** - תיאור מפורט של הבעיה או ההזדמנויות שהפרויקט נועד לפתור, סקירה קצרה של טכנולוגיות רלוונטיות, הסבר על חשיבותו של הפרויקט והתרומה הפוטנציאלית שלו.
- **הגדרת יעדים ומטרות** - הגדרה של לפחות שני יעדים כלליים , ולכל יען להגדיר לפחות שתי מטרות ספציפיות, מדידות, ניתנות להשגה, רלוונטיות בזמן וモgelות בהיקפן (SMART).
- **קהל יעד** - הגדרת קהל היעד של הפרויקט (משתמשים, לקוחות פוטנציאליים וכו') כולל תיאור הצריכם והציפיות של קהל היעד.
- **היקף הפרויקט – דרישות** - תיאור של הפונקציונליות העיקרית של המערכת המתוכננת, הגדרת פונקציונליות בסיסיות [חוובה] ופונקציונליות נוספת (Nice-to-have). (יש קשר מואה לסייע המשמש המוגדרים ב Board)
- **אלוצים** - זיהוי אלוצים טכניים, תקציביים, זמן, משאבי אנוש וכו'.

• תרשים context diagram

- כתיבת סיפורים משתמש – לכל סיפור משתמש תהיה כותרת משמעותית, ותיאור (שדה description בתוך סיפור המשתמש) אשר יהיה כתוב בתבנית הקלאסית לכתיבה של סיפור משתמש: "כ-[מי], אני רוצה [מה], כדי [למה]."

יש ליצור work items עבור סיפורים המשתמש :

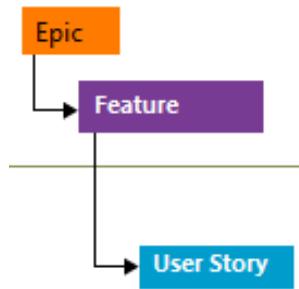
- יש להגדיר לפחות 30 סיפורים המשתמש עבור דרישות פונקציונליות. סיפורים המשתמש יתכן וישתו בהמשך. יתכן שירדו סיפורים שכתבتنם בהתחלה ונוסף אחדים, הדרישות בפרויקט האגיל הן דינמיות ויכולות לשינויים. בפועל במסגרת הקורס, לא נפתח אתcolm אלא בהמשך נבחר מהתה הרשימה את הסיפורים אותם נבחר לפתוח.
- יש להגדיר לפחות 5 סיפורים המשתמש עבור דרישות איות נתן להיעזר ברשימה במצב הדרישות
- יש לשים לב שיש סיפורים המשתמש מוגדרים היטב – INVEST Independent, Negotiable, Valuable, Estimable, Small, Testable

- קביעת מאפיינים לסיפורים המשתמש :

- לכל סיפור המשתמש נקבע את point story שלו, המשקף את קושי ומורכבות הביצוע שלו. נשתמש בסדרת פיבונacci' ומשחק פוקר המאפשרת למצוות להעריך את סיפורים המשתמש בצורה יחסית, פשוטה ומוסכמת. נשקיע בהחלטה לגבי אומדן story points תוך דגש על שיח צוותי, הפקחת איזודאות, ותוכנו ריאלי של העבודה. [פירוט השיטות ראה בנספח 1](#)
 - לכל סיפור המשתמש נקבע את העדיפות שלו במספר 1-4 לפי שיטת MoSCoW :
- 1 - Must have , 2 - Should have , 3 - Could Have , 4 - Won't Have
- כמו כן לכל סיפור המשתמש יש לתת סיווג classification – לכל סיפור משתמש נקבע האם הוא מסוג Business (פונקציונלי) או מסוג Architectural (לא פונקציונלי, או שהוא שמשפיע על הארכיטקטורה)

- לכל סיפור המשתמש נגידיר Acceptance Criteria (פירוט של דברים נדרשים להיות כדי לוודא שישpor המستخدم בוצע)

- בניית backlog - ארגון הדרישות למרכז בגישה האגילית, במבנה של עץ EPIC- FEATURE-USER STORY. נאגד קבוצות של סיפורים משתמש למאפיינים FEATURES, וקבוצות של מאפיינים לפי נושאים EPICS.



דוגמא להיררכיה של EPIC-FEATURE-USER STORY

ניהול משתמשים – EPIC

תהליכי הרשמה וAIMOT – FEATURE

רישום משתמש חדש : 'כمستמש חדש, אני רוצה – 1 – USER STORY 1 – AIMOT דואיל וויסמה כדי ליצור חשבון אישי'. להירשם עם אימייל וויסמה כדי להגן על החשבון שלי.

AIMOT דואיל – כمستמש רשום ארצה להוסיף – USER STORY 2 – AIMOT דואיל וויסמה כדי להגן על החשבון שלי. אפשרות דואיל – כאפשרות לאימוט דואיל כדי לאפשר לאימוט דואיל.

יש להגיד את work items המתאים בחלק ה board של azure devops ולקשר ביניהם לפי היררכיה המתוכנת.

- שימוש ב UML : בניית Use Case Diagram עבור הדרישות העיקריות של המערכת ופירוט ב use case documentation full. [ראה נספח 3](#)
- שימוש ב UML: Activity diagrams לתיאור הלוגיקה הנדרשת עבור על דרישת עיקרית של המערכת. [ראה נספח 4](#)
- ייצרת Dashboard התחלתי לניהול ומעקב אחרי User Stories ומשימות. [ראה נספח 5](#)

תוצרים : Deliverables

- WIKI – תיעוד כולל דיאגרמות UML ומסמכים המתארים את המערכת
- backlog – הדרישות של המערכת מאורגנות לפי היררכיה

- Dashboard - לוח מעקב שמשקף את תכולת הפרויקט ומעקב אחריו
פעולות אנשי הצוות

שלב 2 – בחירת ארכיטקטורה טכנולוגיות וכליים כולל Sprint Planning ראשוני (שבוע 4)

- בחירת ארכיטקטורה – המבנה הכללי של המערכת. ניתן לבחור ארכיטקטורה ידועה (3 / Monolith / Microservices) או לשלב אפשרויות שונות. ClientServer/MVC/tier ()
- בחירת טכנולוגיות וכליים : יש לבחור סביבת עבודה, שפה, וכליים למימוש המערכת, יש להתייחס ל front end ו backend .
- יש לבחור כיצד ישמרו הנתונים של המערכת. בחירת בסיס הנתונים כגון +pgAdmin, sql ו��
- יש לפתח SPRINT חדש בשם sprint1 ולהכניס לתוכו סיפור משתמש (פחות אחד) שאוטם תרצה לפתח בספרינט הראשון. יש לקחת בחשבון שלכל ספרינט יש שבועיים ובספרינט הראשון יש את הקמת המערכת מקצה לקצה. לכן מומלץ לבחור סיפור משתמש פשוט יחסית, או כמה פשוטים. כי בסוף הספרינט הראשון צריכה להיות כבר מערכת עובדת מקצה לקצה.
- פירוק למשימות tasks: את סיפורו/ המשמש שבחרנו לספרינט הראשון נפרק למשימות ברורות ונשייר כל משימה לאיש צוות. כל איש צוות רשאי לבחור את המשימה שמתאימה לו , יש לחלק את האחריות באופן אחד בין אנשי הצוות.
- עדכן ה dashboard יש להוסיף לפרויקט לפחות עוד חמישה שאלות רלוונטיות נוספת לננותנו לנו מידע מודיע על הפרויקט והמשימות כולל על משימות הספרינט. את תוצאות השאלות יש להציג בashboard בתצוגות מתאימות. ניתן להוסיף לוחות מעקב לפי הצורך.
- פיתוח מאגר repository לשמירה ומעקב אחרי קוד הפרויקט: ניתן לפתח מאגר שירות ב azure devops או לקשר מאגר קיימ ב github .

תוצריים:

- מסמך בז'יק W המפרט את הארכיטקטורה הנבחרת, בסיסת העבודה הכלים והשפה, והדרך לשמירת הנתונים.

Sprint Backlog ראשוני (סיפורים משתמש נבחרים עם חשיבות).

Sprint Backlog מעודכן כולל התייחסות לוגר Dashboard.

שלב 3 – ספרינט 1 מערך בסיסית עובדת קצת (שבוע 5)

- DB ראשוני
- API בסיסי + דף ראשון באתר. מה שייה בדף זה לפי סיפורים המשתמש שנבחרו לספרינט הראשון. ניתן לפי הצורך להוסיף דפים.
- במהלך הספרינט כל אחד בצוות עובד על המשימות שבחר לביצוע. כאשר אי הוצאות מתחילה לעבוד על המשימה הוא מעדכן את מצבה ל active , כאשר הוא מסיים את המשימה הוא מעדכן את מצבה להיות closed.
- לגבי סיפור משתמש נג"ל יש לעדכן את מצבו ל active כאשר אחד מהאנשים הוצאות מתחילה לעבוד על משימה שקופה אליו, כאשר מסיים את כל המשימות וסיפור המשתמש הושלם יש להעביר את מצבו ל Resolved. (שים לב שרק אחרי הבדיקה הסופית שנעשתה בשלב מתקדם יותר יוכל לסגור אותו סופית למצב closed)
- השינוי במצבים המשמשים וסיפורים המשתמש צריך לבוא לידי ביטוי בדיאגרמות השונות בלוח המעקב ה dashboard , מומלץ להוסיף ל dashboard תרשימים נוספים לפי הצורך ל邏輯 טוב אחרי הפרויקט כולל פילוחים שונים (לפי איש צוות, לפי עדיפות וכדומה)
- יש לבצע פגישות יומיות (ניתן גם בטלפון..) בתבנית meeting daily , בפגישה כל איש צוות עונה על 3 שאלות : מה עשית היום, מה מתכוון לעשותמחר ובעה אחת שעלה במהלך העבודה.
- לאחר שבועיים יש לכתוב בຕיעוד את סיכום sprint כולל מה היה ההספקים, מה היה הקצב של הקבוצה מבחן story point כמו כן יש לבצע פגישה retrospective כדי להבין מה עבד טוב, מה פחות, מה Learned, מה אפשר לשפר. יש להתייחס גם לאומדן story point האם אמדנו נכון כמות המשימות/סיפורים המשתמש שהקבוצה מסוגלת לעשות בשבוע. האם יכולנו לעשות יותר? האם אנחנו מסוגלים לעשות פחות?

תוצרים:

- אתר בסיסי רץ.
- סיכום sprint עם retrospective : יש לענות על השאלות המנחות [ראה](#)

בנוסף 7

- Burndown Dashboard מעודכן User Stories שהושלמו כולל תרשيم

שלב 4 – ספרינטים נוספים (שבועות 6–10)

בשלב זה נבצע 5 ספרינטיםanza בזה אחר זה שכל אחד נמשך שבוע. בכל ספרינט נתקדם בפיתוח המערכת ונתקבל מערכת רצה עם מאפיינים נוספים. בכל ספרינט יבחרו סיפורים המשתמש על ידי הוצאות תוך שהוצאות לומד מה她们 העבודה שהוא מסוגל לבצע בשבועיים. הוצאות ידיק את עצמו מספרינט לשפרינט ויסיק כיצד לשכלל את עבודות הוצאות, ואת הנהול של המשימות וביצוען.

להלן הדרישות בכל ספרינט:

- פתיחת SPRINT חדש (יש לתת לשפרינט שם לפי מספר הספרינט הנוכחי (stprint2
- בחירת User Stories חדשים מה- backlog והכנסתם לשפרינט. במידה יש צורך לשנות את backlog ניתן לעשות זאת. במידה יש משימות שלא הושלמו בספרינט הקודם יש להכניסם לשפרינט הבא.
- פירוק סיפורים המשמש הנבחרים למשימות tasks וחלוקת המשימות בין אנשי הוצאות.
- פיתוח הפונקציונליות (לפי מה שנבחר לשפרינט). את כל הקוד יש להכניס למגרר. יש ליצור גרסה שחרור release daily meetings
- עדכון לוח המעקב במידה וצריך עדכון מצב המשימות וסיפורים המשמש לפי ההתקדמות בסיום השבועיים:

 - הצגת גרסה עובדת למתרגל.
 - ישיבת Retrospective להבין מה ניתן לשפר, מה עשינו טוב, מה פחות טוב.
 - דוח סיכום ה- SPRINT

תוצרים בכל ספרינט:

- גרסה מורחבת של האתר.
- סיכום SPRINT + Retrospective ב-Wiki [ראה בנספח 7](#)

- Dashboard הושלמו/נשארו User Stories מעודכן

שלב 5 – בדיקות מערכת וטיפול באגים (שבועות 11–12)

- Test planningaghדרת הטסטים שמתכוונים לעשות, יש לבחור את סיפורים המשמש אותם רוצים לבדוק לעומק ולתכנן להם בדיקות.
- כתיבת בדיקות לאתר (UI) Selenium - יש לכתוב לפחות 5 בדיקות End-to-End לזרימות חשובות
- הרצת הבדיקות על האתר.
- יש להכין רשימת פגמים(Bugs) - תיקון בעיות תוך פתיחת באגים azure devops וטיפול באג לאיש צוות. כל סיפור משתמש שהבדיקה עליו עברה בצויה מוצלחת, או שנפתח עליו באג והוא תוקן - יש לעדכן את מצבו ל closed
- כתיבת סיכום בדיקות ב-Wiki.

תוצרים:

- סט בדיקות Selenium
- תיעוד סיכום בדיקות: כולל את תכנון הבדיקות, הבאים שהתגלו ודוח ביצועים
- אתר מתוקן.
- Dashboard סופי.

הגשה סופית (שביע 13)

תוצרים:

- אתר עובד מלא.
- Dashboard מלא עם כל נתוני ההתקדמות.
- Wikipedi: דרישות, ארכיטקטורה, סיכומי ספרינטים עם ותוצאות בדיקות Retrospectives,
- מדריך למשתמש או הדגמת ביצועי האתר בוידאו

נספחים**נספח 1: הערכת Story Points בשיטת Agile**

במסגרת הפרויקט יישא שימוש בערכת **Story Points** לצורך תיעוד ותוכנון עבודה הפיתוח.

הערכת **Story Points** אינה נועדה למדוד זמן עבודה ישיר, אלא את רמת המורכבות, המאמץ, הסיכוןים וחוסר הוודאות הכרוכים במימוש סיפורו המשתמש.

הערכת Story Points**1. שימוש במספרים מסדרת פיבונacci'**

מקובל להשתמש בסדרת פיבונacci' (1, 2, 3, 5, 8, 13, ...) לצורך מתן נקודות. הרעיון הוא שכל שהסיפורים מורכבים יותר, הערך בין ההצעות גדול באופן לא לינארי.

הצעות בוחר סיפור פשוט ומוכר כנקודות "יחום" (למשל: "משתמש יכול להתחבר למערכת עם שם משתמש וסיסמה") ומעניק לו ערך של **3 נקודות**. לאחר מכן שאר הסיפורים נמדדים ביחס אליו.

2. – Planning Poker פוקר תכנון

כאשר קשה להגיע להסכמה לגבי הערכת סיפור מסוים, ניתן להשתמש במשחק פוקר תכנון. כל חבר צוות בוחר קלף עם הערכה משלו (סדרת פיבונacci'), והמספרים נחשפים בו-זמנית. התהילך חוזר מספר סבבים עד להגעה להסכמה.

אם לא מושגת הסכמה – זה סימן שהסיפור גדול מדי או מנוטה באופן לא ברור, ויש לפරקו או לחדד את הדרישות.

דוגמאות להערכת סיפורים משתמש**• סיפור פשוט (3 נקודות):**

"כمستמש, אני רוצה לשנות סיסמה כדי לשמור על חשבונן מאובטח".
→ משימה מוכרת, ישימה בקלות, ללא סיכון מיוחד.

• סיפור מבני (5 נקודות):

"כمستמש, אני רוצה לקבל מייל איפואו סיסמה עם קישור חד-פעמי".
→ כולל עבודה מול שרת דוא"ל, טיפול באבטחה, דרישות נוספות של בדיקות.

• סיפור מורכב (13 נקודות):

"כمستמש, אני רוצה לקבל התראות בזמן אמת לאפליקציה ולמייל

כasher חשבון שלו נפרץ".

- מערב טכנולוגיות מרבות, בדיקות אבטחה וניהול אירועים בזמן אמת
- רמת אי ודיות גבוהה.

שיםו לב -

- ההערכה נעשית **באופן ייחסי** לסייע המשמש האחרים.
- המספרים אינם מתרגמים ישירות לשעות עבודה.
- חוסר בהירות או מחלוקת בהערכת סיפור → סימן ש策יר לפרק את הסיפור לקטעים יותר או **לנסח מחדש**.
- Story Points הם הערכה של **צוות שלם** ולא של אדם בודד. זה מחזק את עקרון האחריות המשותפת.

למה פיבונאצ'י?

– השימוש בסדרה (1, 2, 3, 5, 8, 13...) בא כי ככל שהמספרים גדלים, ההבדל בין רמות הקושי הופך להיות משמעותית יותר. זה מדגיש את העבודה שלא תמיד אפשר להעריך באופן **לינארי**

נקודות ייחוס:(Reference Story)

– הוצאות בוחר סיפור "בסיס" שמכור לכלם (למשל: "משתמש יכול להתחבר למערכת עם שם משתמש וסיסמה"). כל שאר ההצעות נעשות יחסית לסיפור זהה.

לא רק זמן – אלא מרכיבות:

Story points – לא מודדים רק זמן עבודה, אלא שילוב של **מרכבות טכניות, סיכונים, חוסר ודיות, ומאמץ**. חשוב להציג לסטודנטים שלא מדובר בשעות עבודה.

טיפ חשוב לסטודנטים:

אם הוצאות **לא מצליח להחליט** – זה סימן ש:

- הסיפור **גדול מדי** →策יר לפרק לקטעים.
- הסיפור **לא מבן מספיק** →策יר לנסח מחדש או לחזק דרישות.

בسطה 2 - שיטת MoSCoW לקביעת עדיפויות

שיטת MoSCoW היא שיטה פשוטה וברורה לדרג את הדרישות וסיפורים המשמש לפיה רמת החשיבות והעדיפות שלהם. השיטה מחלקת את הסיפורים לארבע קטגוריות:

1. Must Have – חיבים להיות

- סיפורים המשמש שמרכיבים את הדרישות הקritisיות ביותר של המערכת.
- ללא סיפורים אלו המערכת לא תוכל לתקן בrama המינימלית.
- לדוגמה: "כמשתמש, אני רוצה להתחבר עם שם משתמש וסיסמה כדי לגשת למערכת".

2. Should Have – רצוי שייהו

- סיפורים חשובים מאוד, אך אינם קritisים להפעלה הראשונית.
- יגרמו לכך שהמערכת תהיה תחרותית וטובה יותר לעומת פתרונות אחרים.
- לדוגמה: "כמשתמש, אני רוצה לקבל מייל לאישור הרשמה כדי לוודא שכותבת הדוא"ל נכונה".

3. Could Have – אפשר שייהו

- סיפורים שימושיים נוחות ושיפור בחווית המשתמש, אך אינם חיוניים.
- אם יהיה זמן ומשאבים, אפשר למשוך אותם.
- לדוגמה: "כמשתמש, אני רוצה לבחור תמונה פרופיל מותאמת אישית".

4. Won't Have – לא יהיו בגלגול הנוכחי

- סיפורים שפחות משתלים להשקיע בהם כרגע, אך אולי יהיו רלוונטיים בעתיד.
- אפשריים למקד את הוצאות בפיתוח של מה שבאמת נדרש.
- לדוגמה: "כמשתמש, אני רוצה שהמערכת תتمמשק עם רשותות חברותיות".

שימוש לב

שיטת WoSCoM מתאימה במיוחד לניהול פרויקטים בגישה Agile, משומש שהיא מאפשרת:

- **לקבוע מהו ה- MVP - Minimum Viable Product גרסה המוצר המינימלית**
- **למקד את הצוות בדרישות החיוניות ביותר.**
- **לשמר גמישות לטובת פיתוח עתידי בהתאם בזמן ומשאבים.**

נספח 3 : תרשימים use case

יש לבחור 5 פונקציות עיקריות ולתאר אותן באמצעות use case diagram יש לוודא שהתרשים מכיל את המאפיינים הבאים ולפי ה策ור:

- a. Use Cases (מרקרי השימוש) - לפחות 5
- b. Actors שחקנים עם השמות שלהם
- c. קשרים בין Actors ומקרים השימוש בהם הם משתתפים
- d. קשרים בין מקרים שימוש של extends / include במידה ויש צורך
- e. Border - יש להכניס את מקרים השימוש למלבן כך שהשחקנים מחוץ למלבן - גבולות המערכת
- f. קשר ירושא בין מקרים שימוש / בין שחקנים במידת הצורך

את התרשים ניתן ליצור בכל תוכנת עזר כגון UML whiteStar או בכל תוכנה אחרת שנוחה לכם ומתאימה לשימושם שנלמדו בכיתה.

יש להציג צלום מסך ברור של התרשים או לצרף קקובץ לתיעוד ב-WIKI.
(אין להגיש את התרשים בכתב יד סrox)

תיעוד ופירוט מקרים השימוש

יש לתעד מקרה שימוש אחד באמצעות טבלה如下 full use case documentation table:

- שם ה usecase ותיאור מילולי קצר שלו
- את תנאי הקדם והbetter שלו במידה קיימים.

- Main Flow – יש לבחור תרחיש ראשי אחד עיקרי ולציין בפירוט את כל השלבים שלו (התרחיש הראשי הוא התרחיש העיקרי במערכת)
- (s) Alternate Flow – יש לציין במידה ויש, אльтרנטיבות לתרחיש העיקרי, שהן תרחישים אפשריים לתרחיש הראשי.
- (s) Exception Flow – יש לציין חריגות, שהם מצבים פחות צפויים לתהליך הראשי.

לכל תהליך יש תרחיש ראשי, לפחות (s) אחד או (s) Flow אחד.

Use Case:	Use Case Name
Short Description:	A Brief Description of the Use Case
Pre-Conditions:	A description of the conditions that must be satisfied before the use case is invoked
Post-Conditions:	A description of what has happened at the end of the use case
Main Flow:	A list of the system interactions that take place under the most common scenario. For example, for "withdraw money", this would be "enter card, enter pin, etc..."
Alternate Flow(s):	A description of possible alternative interactions
Exception Flow(s):	A description of possible scenarios where unexpected or unpredicted events have taken place

נספח 4 תרשימי activity diagrams

- יש לבחור שני מקרים שימוש מדיאגרמת use case use ולכל אחד מהם ליצור activity diagram שיתאר את הלוגיקה של הפונקציה
- לפני כל תרשימים יוצג שם מקרי השימוש ותיאור קצר שלו. כל תרשימים activity יכול פחות 5 פעוליות. יש להשתמש באלמנטים השונים והמגוונים של תרשימים activity: כמו activity node, fork, join, decision node ועוד.
- חובה להשתמש ב- swim lanes לפחות בתרשימים אחד.
- הציון יקבע בהתאם לרמת התרשימים ובגיוון השימוש באלמנטים השונים.
- את התרשימים ניתן ליצור בכל תוכנת עזר כגון תוכנת UML whiteStar או בכל תוכנה אחרת שנוחה לכם ומתאימה לשימושם שנלמדו בכיתה.
- יש להציג תצלומי מסך ברורים של התרשימים או לצרף קקובצים לתיעוד ב-WIKI. (אין להגיש את התרשימים בכתב יד סרוק)

נספח 5 - לוח מעקב בסיסי דוגמא

להלן הנחיות להכנת לוח מעקב בסיסי.

יש ליצור 3 שאלות כדלקמן:

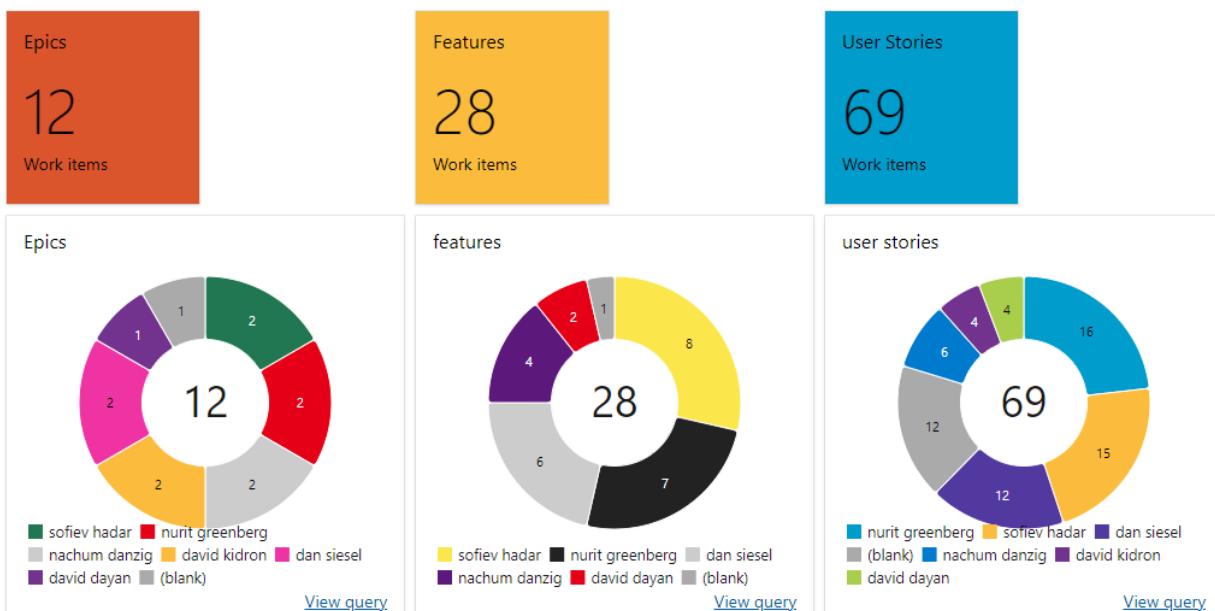
1. סינון כל ה epics
2. סינון כל ה features
3. סינון כל ה user story

לכל שאלתא יש להגדיר עמודה נוספת נספחת של By

ב Dash board בשם overview יש להציג את התרשימים הבאים:

- מספר(epics,features,user stories), ומספר המוגדרים בפרויקט.
- תרשימים המציאן כמה epics כל אחד מחברי הצוות יצר.
- תרשימים המציאן כמה features כל אחד מחברי הצוות יצר.
- תרשימים המציאן כמה stories כל אחד מחברי הצוות יצר.

דוגמאות:



ג'ספה 6 בונוסיפ**בונוס 1:**

הגדרת סיכונים (הבונוס שווה בין 1 ל 10 נקודות, תלוי באיכות הביצוע) יש לכתוב רשימה של 20 סיכונים למערכת. לכל סיכון לכתוב את **רמת הסיכון** (מספר בין 1 ל 4), **משפחת הסיכון** (Category), **סוג הסיכון** ואת **הסבירות** שהוא יקרה.

יש למין את רשימת הסיכונים בסדר יורך לפי רמת סיכון וסבירות. מעברו 3 הסיכונים הנמצאים בתחילת הרשימה (שהם בסיכון גבוהה וסבירות גבוהה) לכתוב תוכנית RMMR שזהה תיאור קצר ומדויק המשפק מידע על תוכנית פעולה הכוללת רעיונות לצמצום הסיכון, אין נזקоб אחריו, ותוכנית מוגרת במידה והסיכון יקרה

יש להוסיף דף "סיכונים" לтиיעוד בויקי המפרט את כל הנדרש לבונוס.

לנוחותכם מידע רלוונטי לבונוס סיכונים:

רמת סיכון:

- 1 – High (Catastrophic) יש לו סיכון שם יקרה יכשיל את המשימה או לעלות סכום משמעותי מאוד של כסף
- 2 – Medium (Critical) יש לו סיכון שיכול להכשיל את המשימה בסבירות גבוהה או לעלות סכום משמעותי של כסף
- 3 – Low (Marginal) יש לו סיכון להוריד קצר בביצועים או לעלות סכום לא משמעותי של כסף
- 4 – Negligible – זניח

משפחות סיכונים (Category):

- גודל המוצר PS Product Size
- השפעות עסקיות BU Business Impact
- מאפייני משתמשים CU Customer characteristics
- הגדרת תהליכי פיתוח PR Process definition
- סביבת הפיתוח DE Development environment
- טכנולוגיה מפותחת TE Technology to be built
- גודל צוות וניסיונו ST Staff size and experience

סוגי סיכונים:

- **סיכון ביצועים** (Performance Risk) – ביצועי המערכת ייפגעו.
- **סיכון עלות** (Cost Risk) – עלות הפרויקט לא תהיה בהתאם למה שהוא חשוב.
- **סיכון סיוע** (Support Risk) – התוכנה שתפותה תהיה: לא נוחה לתיקון, לא נוחה להתאמאה, לא נוחה להרחבתה.
- **סיכון לוחות זמנים** (Schedule Risk) – לוח הזמנים ישתנה

בונוס 2:

הגדרת pipeline בזdevops pipeline – הגדרת azure pipeline כוללת את כל החוקים, המשימות והסיבות שמרכיבים את מחזור החיים של האפליקציה – מהקוד ועד הפרישה.

זרימת העבודה של Azure DevOps Pipeline ב-

Code commit repo → trigger pipeline → Build stage → test stage → publish artifact → deploy to environment

שתי אפשרויות להגדיר Pipeline:

1. **YAML** קובץ טקסט שנמצא במאגר הקוד (מומלץ לפרויקטים מורכבים, נותן גמישות מלאה).
2. **מסך גרפי – (Classic Editor)** מאפשר לבנות Pipeline בצורה ייזואלית דרך דףפן, על ידי גירירה והגדירה של שלבים.

מידע כללי להגדרת pipeline**מקור הקוד(Repository connection**

- חיבור לד GitHub, Azure Repos או מערכת בקרת גרסאות אחרת.
- קובע מייפה ישאב הקוד בכל הפעלה.

2. טריגרים(Triggers)

- מתן הד pipeline ירוץ – למשל בכל commit ל- branch מסוים,
- או באופן pull request, ידני/מתזמן.

3. סוכנים(Agents / Runners)

- השירותים VMs או containers עליהם ירוץ הד pipeline.
- ניתן לבחור self-hosted agents של Azure או hosted agents.

4. שלבים(Stages)

- חלוקה לחידות לוגיות למשל Build, Test, Deploy.
- אפשרותים בקרה והרשאות שונות בכל שלב.

5. משימות(Tasks / Jobs / Steps)

- הפעולות בפועל שה- pipeline מבצע: קומפקטציה, התקנת חבילות, הרצה בדיקות, פריסת קבצים לשרת וכו'.
- נכתבות ב- YAML או מוגדרות דרך מסך גרפי.

6. משתנים(Variables)

- ערכים שנitin להשתמש בהם לאורך הד pipeline למשל connection strings, strings, connection גרסאות.

- חלוקם יכולם להיות secrets שמואחסנים ב-Azure Key Vault
- 7. ארטיפיקטים(Artifacts) – תוצרים של pipeline למשל: קובצי build, חבילות NuGet, Docker images שעובדים לשלב הבא או נשמרים להפצה.
- 8. תנאים ותלוויות(Conditions & Dependencies) – שליטה באילו מצבים שלב יירוץ למשל: רק אם הבדיקות עברו בהצלחה, או רק על branch מסוים.
- Deployment Targets (Environments/Release) .9 deploy (Development, Staging, Production).
- השרתים או הסביבות אליהם מתבצע ה-
- ניתן להגדיר approvals ידניים לפני מעבר בין סביבות.

גופח 7 – שאלות מנהרות לסיכון ישיבת Retrospective

- מה עבד טוב בספרינט זהה?
- מה היה מתאים או לא הילך כמו צייפינו?
- מה הפתיע אותנו?
- מה נרצה לשמר בספרינט הבא?
- מה נרצה לנשות אחרת?
- מה נעבד עליו יותר טוב בספרינט הבא?
- אילו תובנות קיבלנו לגבי אופן העבודה בצוות?
- מודל העמונת Start–Stop–Continue : מה עוצר/נפשיך לעשות, (Stop) ? מה ממשיך, (Continue) ומה נסייף(Continue) ?

הרכב ציון

- שבועות 1–2 → דרישות + Backlog + UML → 25%

 - Backlog (Epics, Features, User Stories) 6%
 - Use Case Diagram + Activity Diagram 10%
 - תיעוד מלא ב-Wiki 3%
 - Dashboard ו-Stories 3% Story Points.

- שבוע 3 → ארQUITטורה + Sprint Planning 7%

 - בחירת ארQUITטורה וטכנולוגיות (DB, API, Frontend, Backend) 2%
 - ייצרת SPRING ופרויקט למשימות 5%

- שבוע 4 → ספרינט ראשון → מערכת בסיסית. 13%

 - פיתוח תוכר ראשון API, DB: ראשון, דף ראשון באתר. 10%
 - Retrospective ראשוני 3%

- שבועות 5–10 → 3 ספרינטים נוספים → הרחבות. כל ספרינט 14% מהכ 42%

 - הרחבת המערכת עם פיצרים נוספים. 10%
 - עדכן Wiki ו-Dashboard 2%
 - Retrospective 2%

- שבועות 11–12 → בדיקות + באגים. 10%

 - כתיבת בדיקות. 5% Selenium
 - תיעוד תוצאות הבדיקות, באגים 5%

- שבוע 13 → הגשה סופית + הערכה כללית + מדריך למשתמש/וידאו. 3%

בונוסים: כל בונוס שווה בין 1 ל 5 נקודות לפי איכות הביצוע, ולפי החלטת המתרgal

הboneos נועד לעודד יצירתיות ולימוד עצמי. אם יש לכם רעיון נוסף של שימוש בגישת devops ומימושה בתוכנת azure devops – תוכלו לשלוח את הרעיון למרצה שיקבל החלטה האם הדבר יחשב כboneos.

הערה חשובה: יש לעמוד בזמן הפרויקט כפי שהוגדרו, במידה ווצאות לא יעמוד בזמן הῆגשה יורדו נקודות על אי עמידה בזמן.

הדגשים כלליים:

- **Retrospective** חובה בסוף כל ספרינט → מטרה שיפור תהליכי הפיתוח והכרות של הוצאות עם היכולות שלהם. [ראה בנספח 7](#)
- **Dashboard** חי מההתחלת → מעקב מתמיד אחרי User Stories ומשימות מעקב שמאפשר לכל אנשי הוצאות (כולל המתרgal..) להבין היטב את מצב הפרויקט לפי פילוחים מגוונים.

מתן הרשות גישה administrator

מתן הרשות גישה של מנהל לחבר בצוות

- Go to the Organization level
- Click on the Organization settings (פינה שמאלית למטה)
- Go to General->Users
- For each User do the following steps:
 - Click on 3 dots (סוף השורה מימין)
 - Click on “Change access level”.
 - Choose Access level “Basic”. Click on Save.
- Go to Security-> Permission
- Click on “Project Collection Administrators”
 - Go to tab Members

- Click on button **Add**. (פינה ימנית עליונה)
- Add all the members of your team in the following POPUP.
Click on Save.

דרישות איכויות

מאפייני איכות תוכנה חיצוניים External characteristics

- **נכונות** Correctness : עד כמה (המידה) מערכת היא ללא אי-הבנות בפרט (Specification), בעיצוב (Design) ובביצוע (Implementation).
- **שימושות Usability** : הקלות שבה משתמשים יכולים ללמוד ולהשתמש במערכת.
- **יעילות Efficiency**: שימוש מינימלי במשאבי המערכת, כולל זיכרון וזמן ביצוע.
- **אמינות Reliability** : היכולת של המערכת להמשיך לעבוד גם אם יש בה כשלים מעטים.
- **שלמות Integrity**: מניעת שימוש המערכת בלתי מורשה או לא תקין.
- **יציבות Robustness**: תפקוד המערכת בנסיבות קלטיים לא חוקיים.

מאפייני איכות תוכנה פנימיים Internal characteristics

- **תחזוקה Maintainability** : הקלות לשנות תוכנה:
 - או שינויים תכונות של התוכנה
 - או מוסיפים תכונות לתוכנה
- **גמישות Flexibility**: הרחבה או שינוי המערכת לשימושים אחרים או לסביבות אחרות
- **ניידות Portability**: הקלות לשנות מערכת כך שתוכל לפעול בסביבה אחרת
- **שימוש חוזר Reusability**: היכולת לשימוש חלקים מהמערכת במערכות אחרות