

## פרויקט קורס הנדסת תוכנה

151060

### כלים וטכניקות לתהליך פיתוח מערכת תוכנה

מטרת הפרויקט היא לדמות תהליך אמיתי של פיתוח תוכנה בצוות, משלב הדרישות ועד למסירה. דרך הפרויקט תתרגלו 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 וכלים כולל Sprint Planning ראשון (שבוע 3)
9	שלב 3 – ספרינט 1 מערכת בסיסית עובדת מקצה לקצה (שבוע 4)
10	שלב 4 – ספרינטים נוספים (שבועות 5–10)
11	שלב 5 – בדיקות מערכת וטיפול בבאגים (שבועות 11–12)
11	הגשה סופית (שבוע 13)
12	נספחים
12	נספח 1: הערכת Story Points בשיטת Agile
14	נספח 2 - שיטת MoSCoW לקביעת עדיפויות
15	נספח 3: תרשימי 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	..... administrator מתן הרשאת גישה
23	..... דרישות איכות

## קווים כלליים על הפרויקט

### מוטיבציה

פרויקט זה אינו רק משימה אקדמית; הוא סימולציה מודרכת של עבודתכם כמהנדסי תוכנה בעולם האמיתי. כאן תתנסו לא רק בכתיבת קוד, אלא בניהול פרויקט מקצה לקצה – החל מרעיון ועד למוצר עובד – תוך שימוש בכלים ובמתודולוגיות המובילות כיום בתעשייה. ההצלחה בפרויקט תלויה ביכולתכם לעבוד בצוות, לנהל זמן, לקבל החלטות טכנולוגיות ולהתמודד עם אתגרים, בדיוק כפי שתעשו בתפקידכם הבא. ברוכים הבאים למסע המעשי שלכם בעולם הנדסת התוכנה.

### מטרת הפרויקט

התנסות בתהליך פיתוח תוכנה תוך יישום טכניקות ושיטות לניהול משימות וביצוען, בשילוב גישות קלאסיות עם גישות זריזות (AGILE) וגישת DevOps. סביבות העבודה תהיינה: azure devops כמסגרת לפיתוח התוכנה כולל ניהול משימות, תיעוד ושימוש בכלים נוספים. הפרויקט מתבצע בקבוצות של 4 סטודנטים. תוך שימת דגש על שיתוף פעולה מלא וחלוקת אחריות ראויה.

בפרויקט זה נפתח מוצר תוכנה משלב הגדרת הדרישות, דרך תכנון ארכיטקטורה ובחירת טכנולוגיות, ניהול גרסאות ופריסת המערכת. נחוה את העבודה בצוות תוך שימת דגש על איכות הקבוצה, טיב ההתקדמות שלה, והיכולת לנהל את הזמן וביצוע המשימות בצורה מושכלת ומתאימה ליכולות הקבוצה.

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

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

### מבנה הקבוצות

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

### שיטת עבודה

- כל הפרויקט יעשה תוך שימוש בסביבת הפיתוח של azure devops – הסביבה מספקת כלים לניהול ומעקב אחרי דרישות ועבודת הצוות, תיעוד, ניהול גרסאות קוד ועוד.
- הפרויקט כאמור משלב בין שיטות קלאסיות לשיטות זריזות וdevops. בתחילה יוגדרו הדרישות ונשתמש בדיאגרמות UML על מנת להבין את הדרישות. בהמשך נפתח את המערכת בשיטת Agile – כך שיהיו לנו 3 ספרינטים כל ספרינט יימשך שבועיים.
- כל תיעוד שיידרש למערכת יהיה ב Wiki – תוך שמירה על סדר הגיוני וקריא של קבצי התיעוד
- במהלך הפרויקט תתבצע הערכת Story Points לכל User Story. הנחה זו תיבדק לאחר ביצוע סיפור המשתמש תוך מטרה לדייק את עצמינו באומדן המשאבים הנצרכים לכל סיפור משתמש.
- במהלך כל הפרויקט מתחילתו ועד סופו, ננהל מעקב אחר משימות לפי חתכים שונים ובאמצעות דיאגרמות ורשימות שיתעדכנו בכל פעם Dashboard
- השימוש בדיאגרמות UML בתחילת הדרך נועד לתת לנו 'מבט-על' ולהבין את תמונת המערכת הגדולה והאינטראקציות המרכזיות בה. לאחר מכן, נעבור לעבודה איטרטיבית ב Agile – כדי לפתח את המערכת בחלקים קטנים, מה שמאפשר לנו גמישות, קבלת משוב מהיר ויכולת להתאים את עצמנו לשינויים.

### שימוש ב AI

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

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

### סוג הפרויקט הנבחר השנה

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

### תיעוד

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

התיעוד נועד לעזור לכם להבין את המערכת ולתקשר בצוות. אל תכתבו מסמכים רק בשביל הסימון, אלא השתמשו בהם כמצפן לפיתוח.

### תכנון כללי של שלבי הפרויקט

שבועות 1-3 → דרישות + Backlog + UML

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.

## שלב הפרויקט

שלב 1 – תכנון ודרישות (שבועות 1–3)

- כל איש צוות צריך להירשם לazure devops , ואחד מאנשי הצוות פותח ארגון ובונה פרויקט מסוג AGILE ומשתף את כל אנשי הצוות כולל המתרגל – מתן הרשאות של administrator לכל חברי הקבוצה ([ראה נספח 8](#))
- כתיבת מסמך ייזום מקוצר : תיעוד בWIKI של azure devops הכולל
  - עמוד עבור "אודות" - יש לכתוב את שמות מלאים של המגישים עם תעודות זהות, אפשר להוסיף פרטים אודות המגישים.
  - עמוד עבור מסמך הייזום המקוצר – כולל
- **כותרת** - שם הפרויקט, תאריך וסטודנטים (מספר זהות ושם).
- **תקציר** - תיאור קצר וממצה של הפרויקט, כולל המטרה העיקרית וקהל היעד.
- **רקע** - תיאור מפורט של הבעיה או ההזדמנות שהפרויקט נועד לפתור, סקירה קצרה של טכנולוגיות רלוונטיות, הסבר על החשיבות של הפרויקט והתרומה הפוטנציאלית שלו.
- **הגדרת יעדים ומטרות** - הגדרה של לפחות שני יעדים כלליים , ולכל יעד להגדיר לפחות שתי מטרות ספציפיות, מדידות, ניתנות להשגה, רלוונטיות בזמן ומוגבלות בהיקפן (SMART).
- **קהל יעד** - הגדרת קהל היעד של הפרויקט (משתמשים, לקוחות פוטנציאליים וכו') כולל תיאור הצרכים והציפיות של קהל היעד.
- **היקף הפרויקט – דרישות** - תיאור של הפונקציונליות העיקרית של המערכת המתוכננת, הגדרת פונקציונליות בסיסיות [חובה] ופונקציונליות נוספות (Nice-to-have). (יש לקשר מ WIKI לסיפורי המשתמש המוגדרים ב Board)
- **אילוצים** - זיהוי אילוצים טכניים, תקציביים, זמן, משאבי אנוש וכו'.

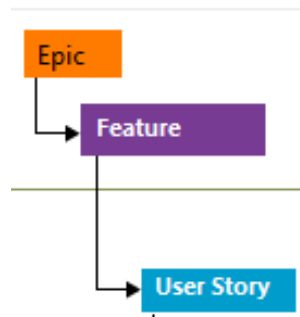
## • תרשים context diagram

- כתיבת סיפורי משתמש – לכל סיפור משתמש תהיה כותרת משמעותית, ותיאור (שדה description בתוך סיפור המשתמש) אשר יהיה כתוב בתבנית הקלאסית לכתיבת סיפורי משתמש:  
**"כ-[מי], אני רוצה [מה], כדי [למה]."**  
 יש ליצור work items עבור סיפורי המשתמש :

- יש להגדיר לפחות 30 סיפורי משתמש עבור דרישות פונקציונאליות. סיפורי המשתמש ייתכן וישתנו בהמשך. יתכן שירדו סיפורים שכתבתנו בהתחלה ונוסיף אחדים, הדרישות בפרויקט האגיל הן דינאמיות ויכולות להשתנות. בפועל במסגרת הקורס, לא נפתח בפועל את כולם אלא בהמשך נבחר מתוך הרשימה את הסיפורים אותם נבחר לפתח.
- יש להגדיר לפחות 5 סיפורי משתמש עבור דרישות איכות ניתן להיעזר ברשימה במצגת הדרישות
- יש לשים לב שסיפורי המשתמש מוגדרים היטב – INVEST  
**Independent, Negotiable, Valuable, Estimable, Small, Testable**
- קביעת מאפיינים לסיפורי המשתמש :

- לכל סיפור משתמש נקבע את ה story point שלו, המשקף את קושי ומורכבות הביצוע שלו. נשתמש בסדרת פיבונאצ'י ומשחק פוקר המאפשרת לצוות להעריך את סיפורי משתמש בצורה יחסית, פשוטה ומוסכמת. נשקיע בהחלטה לגבי אומדן ה story points תוך דגש על שיח צוותי, הפחתת אי-ודאות, ותכנון ריאלי של העבודה. [פירוט השיטות ראה בנספח 1](#)
- לכל סיפור משתמש 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 משתמש יש לתת סיווג classification - לכל סיפור משתמש נקבע האם הוא מסוג Business (פונקציונאלי) או מסוג Architectural (לא פונקציונאלי, או משהו שמשפיע על הארכיטקטורה)
- לכל סיפור משתמש נגדיר Acceptance Criteria (פירוט של דברים שצריכים להיות כדי לוודא שסיפור המשתמש בוצע)

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



דוגמא להיררכיה של EPIC-FEATURE-USER STORY

ניהול משתמשים – EPIC

תהליך הרשמה ואימות – FEATURE

רישום משתמש חדש : 'כמשתמש חדש, אני רוצה – USER STORY 1  
'להירשם עם אימייל וסיסמה כדי ליצור חשבון אישי

אימות דו שלבי – כמשתמש רשום ארצה להוסיף – USER STORY 2  
אפשרות לאימות דו שלבי כדי להגן על החשבון שלי

יש להגדיר את work items המתאימים בחלק ה board של azure devops ולקשר ביניהם לפי ההיררכיה המתוכננת.

- שימוש ב UML : בניית **Use Case Diagram** עבור הדרישות העיקריות של המערכת ופירוט ב full use case documentation. [ראה נספח 3](#)
- שימוש ב UML : **Activity diagrams** לתיאור הלוגיקה הנדרשת עבור על דרישה עיקרית של המערכת. [ראה נספח 4](#)
- יצירת **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 אחרי פעולות אנשי הצוות

## שלב 2 – בחירת ארכיטקטורה טכנולוגיות וכלים כולל Sprint Planning ראשון (שבוע 4)

- בחירת ארכיטקטורה – המבנה הכללי של המערכת. ניתן לבחור ארכיטקטורה ידועה (3- / Monolith / Microservices / ClientServer/MVC/tier וכדומה) או לשלב אפשרויות שונות.
- בחירת טכנולוגיות וכלים : יש לבחור סביבת עבודה, שפה, וכלים למימוש המערכת, יש להתייחס ל front end | backend.
- יש לבחור כיצד ישמרו הנתונים של המערכת. בחירת בסיס הנתונים כגון pgAdmin, sql+ וכדומה
- יש לפתוח SPRINT חדש בשם sprint1 ולהכניס לתוכו סיפורי משתמש (לפחות אחד) שאותם תרצו 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 סיפורי המשתמש שבחרנו לספרינט הראשון נפרק למשימות ברורות ונשייך כל משימה לאיש צוות. כל איש צוות רשאי לבחור את המשימה שמתאימה לו, יש לחלק את האחריות באופן אחיד בין אנשי הצוות.
- עדכון ה dashboard יש להוסיף לפרויקט לפחות עוד חמש שאליות רלוונטיות נוספות הנותנות לנו מידע מקיף על הפרויקט והמשימות כולל על משימות הספרינט. את תוצאות השאליות יש להציג ב dashboard בתצוגות מתאימות. ניתן להוסיף לוחות מעקב לפי הצורך.
- פתיחת מאגר repository לשמירה ומעקב אחרי קוד הפרויקט: ניתן לפתוח מאגר ישירות ב azure devops או לקשר מאגר קיים ב github.

### **תוצרים:**

- מסמך ב Wiki המפרט את הארכיטקטורה הנבחרת, בסיסת העבודה הכלים והשפה, והדרך לשמירת הנתונים.



- Sprint Backlog ראשון (סיפורי משתמש נבחרים עם משימות).
- Dashboard מעודכן כולל התייחסות לSprint Backlog

### שלב 3 – ספרינט 1 מערכת בסיסית עובדת מקצה לקצה (שבוע 5)

- DB ראשוני
- API בסיסי + דף ראשון באתר. מה שיהיה בדף זה לפי סיפורי המשתמש שנבחרו לספרינט הראשון. ניתן לפי הצורך להוסיף דפים.
- במהלך הספרינט כל אחד בצוות עובד על המשימות שבחר לבצע. כאשר אי הצוות מתחיל לעבוד על המשימה הוא מעדכן את מצבה ל active , כאשר הוא מסיים את המשימה הוא מעדכן את מצבה להיות closed. לגבי סיפור משתמש כנ"ל יש לעדכן את מצבו ל active כאשר אחד מאנשי הצוות מתחיל לעבוד על משימה שקשורה אליו, וכאשר מסיימים את כל המשימות וסיפור המשתמש הושלם יש להעביר את מצבו ל Resolved. (שימו לב שרק אחרי הבדיקה הסופית שנעשה בשלב מתקדם יותר נוכל לסגור אותו סופית למצב closed)
- השינוי במצבי המשימות וסיפורי המשתמש צריך לבוא לידי ביטוי בדיאגרמות השונות בלוח המעקב ה dashboard , מומלץ להוסיף לdashboard תרשימים נוספים לפי הצורך למעקב טוב אחרי הפרויקט כולל פילוחים שונים (לפי איש צוות, לפי עדיפות וכדומה)
- יש לבצע פגישות יומיות (ניתן גם בטלפון..) בתבנית daily meeting , בפגישה כל איש צוות עונה על 3 שאלות : מה עשיתי היום, מה מתכנן לעשות מחר ובעיה אחת שעלתה במהלך העבודה.
- לאחר שבועיים יש לכתוב בתיעוד את סיכום ה Sprint כולל מה היו ההספקים, מה היה הקצב של הקבוצה מבחינת story point כמו כן יש לבצע פגישת retrospective כדי להבין מה עבד טוב, מה פחות, מה נלמד, מה אפשר לשפר. יש להתייחס גם לאומדן ה story point האם אמדנו נכון את כמות המשימות/סיפורי 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](#) [בנספח 7](#)

- Dashboard מעודכן User Stories שהושלמו כולל תרשים Burndown

#### שלב 4 – ספרינטים נוספים (שבועות 6–10)

בשלב זה נבצע 5 ספרינטים בזה אחר זה שכל אחד נמשך **שבוע**. בכל ספרינט נתקדם בפיתוח המערכת ונקבל מערכת רצה עם מאפיינים נוספים. בכל ספרינט ייבחרו סיפורי המשתמש על ידי הצוות תוך שהצוות לומד מה כמות העבודה שהוא מסוגל לבצע בשבועיים. הצוות ידייק את עצמו מספרינט לספרינט ויסיק כיצד לשכלל את עבודת הצוות, ואת הניהול של המשימות וביצוען.

להלן הדרישות בכל ספרינט :

- פתיחת SPRINT חדש (יש לתת לספרינט שם לפי מספר הספרינט כדון sprint2)
- בחירת User Stories חדשים מה-Backlog והכנסתם לספרינט. במידה ויש צורך לשנות את ה backlog ניתן לעשות זאת. במידה ויש משימות שלא הושלמו בספרינט קודם יש להכניסן לספרינט הבא.
- פירוק סיפורי המשתמש הנבחרים למשימות tasks וחלוקת המשימות בין אנשי הצוות.
- פיתוח הפונקציונליות (לפי מה שנבחר לספרינט). את כל הקוד יש להכניס למאגר. יש ליצור גרסת שחרור release
- פגישות daily meetings
- עדכון לוח המעקב במידה וצריך
- עדכון מצבי המשימות וסיפורי המשתמש 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 מעודכן User Stories הושלמו/נשארו

### שלב 5 – בדיקות מערכת וטיפול בבאגים (שבועות 11–12)

- Test planning הגדרת הטסטים שמתכננים לעשות, יש לבחור את סיפורי המשתמש אותם רוצים לבדוק לעומק ולתכנן להם בדיקות.
- כתיבת בדיקות לאתר Selenium (UI) - 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 ושיוך הטיפול בבאג לאיש צוות. כל סיפור משתמש שהבדיקה עליו עברה בצורה 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, הבאגים שהתגלו ודוח ביצועים
- אתר מתוקן.
- Dashboard סופי.

### הגשה סופית (שבוע 13)

### תוצרים:

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

## נספחים

נספח 1: הערכת Story Points בשיטת Agile

במסגרת הפרויקט ייעשה שימוש בהערכת Story Points לצורך תיעדוף ותכנון עבודת הפיתוח.

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

**הערכת Story Points****1. שימוש במספרים מסדרת פיבונאצ'י**

מקובל להשתמש בסדרת פיבונאצ'י (1, 2, 3, 5, 8, 13, ...) לצורך מתן נקודות. הרעיון הוא שככל שהסיפורים מורכבים יותר, הפער בין ההערכות גדל באופן לא לינארי. הצוות בוחר סיפור פשוט ומוכר כנקודת ייחוס (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 (מסדרת פיבונאצ'י), והמספרים נחשפים בו־זמנית. התהליך חוזר מספר סבבים עד להגעה להסכמה. אם לא מושגת הסכמה – זה סימן שהסיפור גדול מדי או מנוסח באופן לא ברור, ויש לפרקו או לחדד את הדרישות.

**דוגמאות להערכת סיפורי משתמש**

- **סיפור פשוט (3 נקודות):**  
"כמשתמש, אני רוצה לשנות סיסמה כדי לשמור על חשבון מאובטח".  
→ משימה מוכרת, ישימה בקלות, ללא סיכונים מיוחדים.
- **סיפור בינוני (5 נקודות):**  
"כמשתמש, אני רוצה לקבל מייל איפוס סיסמה עם קישור חד-פעמי".  
→ כולל עבודה מול שרת דוא"ל, טיפול באבטחה, דרישות נוספות של בדיקות.
- **סיפור מורכב (13 נקודות):**  
"כמשתמש, אני רוצה לקבל התראות בזמן אמת לאפליקציה ולמייל

כאשר חשבון שלי נפרץ".

→ מערב טכנולוגיות מרובות, בדיקות אבטחה וניהול אירועים בזמן אמת  
– רמת אי ודאות גבוהה.

שימו לב -

- ההערכה נעשית באופן יחסי לסיפורי המשתמש האחרים.
- המספרים אינם מתורגמים ישירות לשעות עבודה.
- חוסר בהירות או מחלוקת בהערכת סיפור → סימן שצריך לפרק את הסיפור לקטנים 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 משמעותי יותר. זה מדגיש את העובדה שלא תמיד אפשר להעריך באופן לינארי

### נקודת ייחוס: (Reference Story)

– הצוות בוחר סיפור "בסיס" שמוכר לכולם (למשל: "משתמש יכול להתחבר למערכת עם שם משתמש וסיסמה"). כל שאר ההערכות נעשות יחסית לסיפור הזה.

### לא רק זמן – אלא מורכבות:

Story points – לא מודדים רק זמן עבודה, אלא שילוב של מורכבות טכנית, סיכונים, חוסר ודאות, ומאמץ. חשוב להדגיש לסטודנטים שלא מדובר בשעות עבודה.

### טיפ חשוב לסטודנטים:

אם הצוות לא מצליח להחליט – זה סימן ש:

- הסיפור גדול מדי → צריך לפרק לקטנים.
- הסיפור לא מובן מספיק → צריך לנסח מחדש או לחדד דרישות.

## נספח 2 - שיטת MoSCoW לקביעת עדיפויות

שיטת MoSCoW היא שיטה פשוטה וברורה לדרג את הדרישות וסיפורי המשתמש לפי רמת החשיבות והעדיפות שלהם. השיטה מחלקת את הסיפורים לארבע קטגוריות:

### 1. Must Have חייבים להיות

- סיפורי המשתמש שמרכיבים את הדרישות הקריטיות ביותר של המערכת.
- ללא סיפורים אלו המערכת לא תוכל לתפקד ברמה המינימלית.
- לדוגמה: "כמשתמש, אני רוצה להתחבר עם שם משתמש וסיסמה כדי לגשת למערכת".

### 2. Should Have רצוי שיהיו

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

### 3. Could Have אפשר שיהיו

- סיפורים שמוסיפים נוחות ושיפור בחוויית המשתמש, אך אינם חיוניים.
- אם יהיה זמן ומשאבים, אפשר לממש אותם.
- לדוגמה: "כמשתמש, אני רוצה לבחור תמונת פרופיל מותאמת אישית".

### 4. Won't Have לא יהיו בגלגול הנוכחי

- סיפורים שפחות משתלם להשקיע בהם כרגע, אך אולי יהיו רלוונטיים בעתיד.
- מאפשרים למקד את הצוות בפיתוח של מה שבאמת נדרש.
- לדוגמה: "כמשתמש, אני רוצה שהמערכת תתממשק עם רשתות חברתיות".

## שימו לב

שיטת MoSCoW מתאימה במיוחד לניהול פרויקטים בגישת Agile, משום שהיא מאפשרת:

- לקבוע מהו ה-MVP - Minimum Viable Product גרסת המוצר המינימלית
- למקד את הצוות בדרישות החיוניות ביותר.
- לשמור גמישות לטובת פיתוח עתידי בהתאם לזמן ומשאבים.

### נספח 3: תרשימים use case

יש לבחור 5 פונקציות עיקריות ולתאר אותם באמצעות use case diagram  
יש לוודא שהתרשימים מכיל את המאפיינים הבאים ולפי הצורך:

- Use Cases (מקרי השימוש) - לפחות 5
  - Actors שחקנים עם השמות שלהם
  - קשרים בין Actors ומיקרי השימוש בהם הם משתתפים
  - קשרים בין מקרי שימוש של include / extends במידה ויש צורך
  - Border - יש להכניס את מקרי השימוש למלבן כך שהשחקנים מחוץ למלבן - גבולות המערכת
  - קשר ירושה בין מקרי שימוש / בין שחקנים במידת הצורך
- את התרשימים ניתן ליצור בכל תוכנת עזר כגון תוכנת whiteStarUML או בכל תוכנה אחרת שנוחה לכם ומתאימה לסימונים שנלמדו בכיתה.  
יש להציג תצלום מסך ברור של התרשימים או לצרף כקובץ לתיעוד ב WIKI.  
(אין להגיש את התרשימים בכתב יד סרוק)

### תיעוד ופירוט מקרי השימוש

יש לתעד מקרה שימוש אחד באמצעות טבלה full use case documentation  
יש לתעד באמצעות הטבלה את:

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

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

לכל תהליך יש תרחיש ראשי, לפחות Alternate Flow(s) אחד או Exception Flow(s) 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 מקרי שימוש מדיאגרמת use case ולכל אחד מהם ליצור activity diagram שיתאר את הלוגיקה של הפונקציה
- לפני כל תרשים activity יוצג שם מקרי השימוש ותיאור קצר שלו. כל תרשים activity יכול לפחות 5 פעילויות. יש להשתמש באלמנטים השונים והמגוונים של תרשים ה-activity: כמו fork, join, decision node ועוד.
- חובה להשתמש ב-swim lane לפחות בתרשים אחד.
- הציון ייקבע בהתאם לרמת התרשימים ובגיוון השימוש באלמנטים השונים.
- את התרשימים ניתן ליצור בכל תוכנת עזר כגון תוכנת whiteStarUML או בכל תוכנה אחרת שנוחה לכם ומתאימה לסימונים שנלמדו בכיתה.
- יש להציג תצלומי מסך ברורים של התרשימים או לצרף כקובצים 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 כדלהלן:

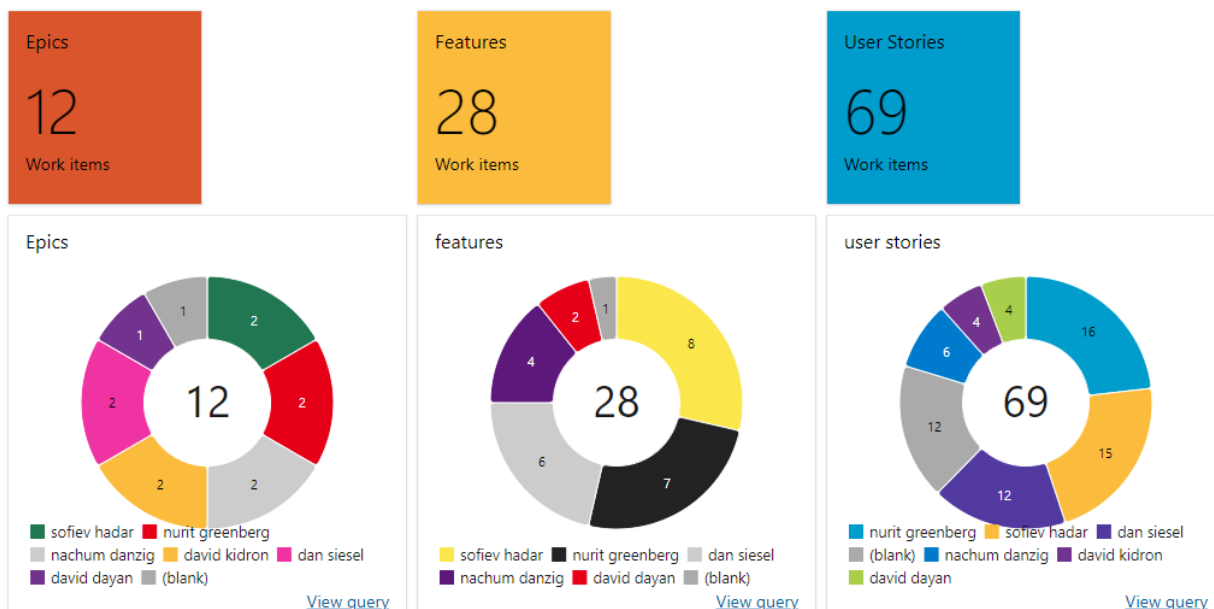
1. סינון כל ה epics
2. סינון כל ה features
3. סינון כל ה user story

לכל שאלתא יש להגדיר עמודה נוספת של Created By

ב Dash board בשם overview יש להציג את התרשימים הבאים:

- מספר ה epics, מספר ה features, ומספר ה user stories המוגדרים בפרויקט.
- תרשים המציין כמה epics כל אחד מחברי הצוות יצר
- תרשים המציין כמה features כל אחד מחברי הצוות יצר
- תרשים המציין כמה stories כל אחד מחברי הצוות יצר

דוגמא:



נספח 6 ביוניםבונס 1:

הגדרת סיכונים (הבונס 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 למיין את רשימת הסיכונים בסדר יורד לפי רמת סיכון וסבירות. ועבור 3 הסיכונים הנמצאים בתחילת הרמינה (שהם בסיכון גבוהה וסבירות גבוהה) לכתוב תוכנית RMMM שזהו תיאור קצת ותמציתי המספק מידע על תוכנית פעולה הכוללת רעיונות לצמצום הסיכון, איך נעקוב אחריו, ותוכנית מגרה במידה והסיכון יקרה

יש להוסיף דף "סיכונים" לתיעוד בויקי המפרט את כל הנדרש לבונס.

לנוחותכם מידע רלוונטי לבונס סיכונים:

רמת סיכון:

- 1 – High (Catastrophic) יש לו סיכון שאם 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 שיכול להוריד קצת בביצועים או לעלות סכום לא משמעותי של כסף
- 4 – Negligible זניח

משפחות סיכונים (Category):

- גודל המוצר PS Product Size
- השפעות עסקיות Business BU
- מאפייני המשתמשים CU Customer characteristics
- הגדרת תהליך 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 שחושב.
- **סיכון סיוע** (Support Risk) - התוכנה שתפותח תהיה: לא נוחה לתיקון, לא נוחה להתאמה, לא נוחה להרחבה.
- **סיכון לוחות זמנים** (Schedule Risk) – לוח הזמנים ישתנה

בונוס 2:

הגדרת pipeline ב-azure devops - הגדרת pipeline כוללת את כל החוקים, המשימות והסביבות שמרכיבים את מחזור החיים של האפליקציה – מהקוד ועד הפריסה.

זרימת העבודה של Pipeline ב-Azure DevOps

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. **ממשק גרפי – (Classic Editor)** מאפשר לבנות Pipeline בצורה ויזואלית דרך דפדפן, על ידי גרירה והגדרה של שלבים.

### מידע כללי להגדרת pipeline

מקור הקוד (Repository connection)

- חיבור ל-GitHub, Azure Repos או מערכת בקרת גרסאות אחרת.
- קובע מאיפה יישאב הקוד בכל הפעלה.
- 2. **טריגרים (Triggers)**
  - מתי ה-pipeline ירוץ – למשל בכל commit ל-branch מסוים, pull request, או באופן ידני/מתוזמן.
- 3. **סוכנים (Agents / Runners)**
  - השרתים VMs או containers שעליהם ירוץ ה-pipeline.
  - ניתן לבחור hosted agents של Azure או self-hosted agents
- 4. **שלבים (Stages)**
  - חלוקה ליחידות לוגיות למשל. Build, Test, Deploy.
  - מאפשרים בקרה והרשאות שונות בכל שלב.
- 5. **משימות (Tasks / Jobs / Steps)**
  - הפעולות בפועל שה-pipeline מבצע: קומפילציה, התקנת חבילות, הרצת בדיקות, פריסת קבצים לשרת וכו'.
  - נכתבות ב-YAML או מוגדרות דרך ממשק גרפי.
- 6. **משתנים (Variables)**
  - ערכים שניתן להשתמש בהם לאורך ה-pipeline למשל connection strings, גרסאות.

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

### נספח 7 – שאלות מנחות לסיכום ישיבת Retrospective

- מה עבד טוב בספרינט הזה?
- מה היה מאתגר או לא הלך כמו שציפינו?
- מה הפתיע אותנו?
- מה נרצה לשמר לספרינט הבא?
- מה נרצה לנסות אחרת?
- מה נעבוד עליו יותר טוב בספרינט הבא?
- אילו תובנות קיבלנו לגבי אופן העבודה בצוות?
- מודל ה-Start-Stop-Continue : מה נעצור/נפסיק לעשות, (Stop), מה נמשיך, (Continue), ומה נוסיף (Start) ?

## הרכב ציון

- שבועות 1–2 → דרישות + Backlog + UML 25%
  - Backlog (Epics, Features, User Stories) 6%
  - Use Case Diagram + Activity Diagram ראשוני. 10%
  - Wiki. מלא ב- 3%
  - Dashboard ראשוני עם Stories ו- Story Points. 3%
- שבוע 3 → ארכיטקטורה + Sprint Planning 7%
  - בחירת ארכיטקטורה וטכנולוגיות (DB, API, Frontend, Backend). 2%
  - יצירת SPRING ופרוק למשימות 5%
- שבוע 4 → ספרינט ראשון → מערכת בסיסית. 13%
  - פיתוח תוצר ראשוני DB, API: ראשוני, דף ראשוני באתר. 10%
  - Retrospective ראשון. 3%
- שבועות 5–10 → 3 ספרינטים נוספים → הרחבות. כל ספרינט 14%
  - הרחבת המערכת עם פיצ'רים נוספים. 10%
  - עדכון Wiki ו- Dashboard. 2%
  - Retrospective 2%
- שבועות 11–12 → בדיקות + באגים. 10%
  - כתיבת בדיקות. Selenium. 5%
  - 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 המתרגל

הבונוס 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 כבונוס.

הערה חשובה: יש לעמוד בזמני הפרויקט כפי שהוגדרו, במידה וצוות לא יעמוד בזמני ההגשה יורדו נקודות על אי עמידה בזמנים.

הדגשים כלליים:

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

[מתן הרשאות גישה administrator](#)

מתן הרשאות 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 איכות

#### מאפייני איכות תוכנה חיצוניים External characteristics

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

#### מאפייני איכות תוכנה פנימיים Internal characteristics

- תחזוקה Maintainability : הקלות לשנות תוכנה:
  - או משנים 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