נכתב ע"י: אביה צ'ריקר :)

הנדסת תוכנה שיעור 3

דרישות:

שאלת השיעור: מה – מה צריך לעשות?

הנדרש: נצטרך להבין יותר את הדרישות ולחקור מה צריך לעשות.

SRS = מסמך דרישות

לפעמים שלב הדרישות יכול להיות מאוד ארוך, מה עושה כל הצוות עושה בינתיים? הרי לא צריך את כל הצוות בשביל לכתוב מסמך דרישות אחד.

1. הרבה פעמים בשלב הדרישות עוד לא גייסנו את כל הצוות שאנו צריכים – ואז אין בעיה
2. אנחנו עושים כל מיני מחקרים בינתיים לגבי טכנולוגיה שכדאי להשתמש, spikes – לקחת איזה טכנולוגיה ולבדוק אם היא מתאימה לנו, מה שגורם להנמכת סיכונים
3. מכינים אב טיפוס
4. הכנת תשתיות פיתוח

<u>פער התקשורת:</u>

יש מצד אחד - הלקוחות והמשתמשים, ומצד שני - מפתחים/מהנדסים

הלקוחות לא מבינים בתוכנה והמפתחים לא מבינים עד הסוף את הבעיה.

מה יקרה אם אחד הצדדים יהיה יותר דומיננטי?

- אם המפתחים יהיו הצד הדומיננטי - נשלוט בתהליך ולא נעשה את הדבר הנכון.
- אם הלקוח יותר דומיננטי ויגיד לנו כל הזמן מה לעשות – הוא יקבל את מה שהוא ביקש ולא מה שהוא צריך.

אחד הפתרונות – מסמך דרישות תוכנה, דרך המסמך נוכל לדבר עם הלקוח ולפתח תקשורת.

(פתרון שני – אג'ייל – בהמשך)

דרישה טובה:

1. נכונה
2. חד משמעית - לא דו משמעית
3. שלימה – לא שכחנו משהו
4. קונסיסטנטית – לא מתנגשת בדברים אחרים
5. אפשר לתעדף אותה
6. אפשר להגיד אם היא חשובה או לא חשובה
7. אפשר לאמת אותה+

8. .\זסב
9. אפשר לשנות אותה
10. עקבות – האם אני מבין מה הצורך בדרישה

<u>תהליך הדרישות כולל כמה שלבים:</u>

1. **איסוף** הדרישות – לברר מול הלקוח את הדרישות הרצויות וצרכיו.

2. **ניתוח** דרישות – ניתוח יותר מעמיק

3. **פירוט** דרישות – תיאור המוצר איך יראה ואת הפונקציונאליות

4. **אימות** דרישות – אחרי שכתבתנו, ניתחנו ופירטנו את הדרישות נאמת את הדרישות מול הלקוח

בכל שלב אנחנו יכולים לחזור אחורה

<u>שני סוגים של דרישות:</u>

דרישות **פונקציונאליות** ו - דרישות **לא פונקציונאליות**

<u>דרישות פונקציונאליות</u> – כל השירותים שהמערכת מספקת.

<u>דרישות לא פונקציונאליות</u> – כל מה שאנחנו מצפים מהמוצר מהשרות שהוא נותן (ביצועים, אמינות, פרטיות ואבטחה, תיעוד וקלות שימוש, קלות הרחבה, סביבת הפעלה וממשקים, עלות וזמנים, תקנים וחוקים.

<u>הגדרת Use – case</u>

למה נועד?

1. מתאר דרך מסוימת להשתמש במערכת

2. מייצג דו-שיח בין משתמש והמערכת מנקודת הראות של המשתמש -קופסה שחורה (מה שחשוב זה שהמערכת כקופסה שחורה והדו שיח זה בין המשתמש למערכת.)

3. כלי ללכידת דרישות פונקציונליות

4. חלק מתרשימי UML

הגדרות:

1. **שחקן** – מישהו שבא במגע עם המערכת (לא חייב להיות בנאדם).
2. **בעלי עניין** – מישהו שיש לו ענין במערכת המפותחת (רוצים להבין מי עוד בעליי עניין) – לא משתמשים בפועל במערכת אך בעלי עניין בה. לפעמים בעל עניין יכול להיות מתחרה.
3. **תרחיש שימוש** – איזשהו חוזה על איך המערכת תפעל.

4. **שחקן ראשי** – זה שמניע את התרחיש, נקרא גם UC.
5. **שם\מטרה -** התוצאה הרצויה לשחקן הראשי ולבעלי העניין.
6. **הקף ורמה -** הפרוט המופיע בתרחיש והיעד: ארגוני -מערכתי- תת-מערכתי.

חלקי UC:

1. טבלת שחקנים - רוצים לפרט את השחקנים השונים.
2. דיאגרמת תרחישים, נתמכת בכלים שונים (UML) –מבט על חלק\כלל תרחישי המערכת.
3. תרחיש שימוש (UML).
4. ובהמשך: סיפורי משתמש.

יתרונות UC:

1. זיהוי שחקנים.
2. מבט כללי.
3. נקודת התחלה לפירוט הדרישות.

איך מפרטים תרחיש ב- Use Case?

1. כותבים את התרחיש כמעין סיפור אחרי התרשים - נכתב כפסקה המתארת תרחיש\אינטראקציה מלא –עם נתונים ספציפיים.
   - *דוגמא לתרחיש:*

   תרחיש המתאר קורא שמאבד ספר:

   015

   הקורא מדווח לספרן שהוא איבד ספר. הספרן מדפיס את רשומת הספר ומבקש מהקורא לדבר עם מנהלת הספרייה, שתקבע את גובה התשלום. המערכת תעודכן בנתוני הספר שאבד וכן כרטיס הקורא. מנהלת הספרייה עשויה להורות על רכישת תחליף.
2. בתרחיש שימוש חייב למלא טבלה (שגם אנחנו נצטרך למלא) שמתארת את התהליך.
   - *דוגמא לטבלה:*

   טבלה המתארת תרחיש של שקורא מאבד ספר:

נכתב ע"י: אביה צ'ריקר :)

שם	הזמנת ספר
שחקן ראשי	קורא
מטרה	קורא מעוניין לשריין ספר מתוך הקטלוג המקוון
הקף	מערכת הספריה
רמה	משתמש (cockburn)
בעלי עניין ואינטרסים	קורא – לשריין ספר בעל הספריה – שרות מורחב לרווחת הלקוחות
טריגר	הקורא נכנס למערכת
תנאי-קדם	הקורא עבר את מסך ההזדהות (login) ונחת בעמוד הבית
תנאי סיום מוצלח	הספר שמור עבור הקורא (האם זה תנאי מוצלח?)
תנאי כישלון	הספר אינו שמור
תרחיש הצלחה עיקרי	1. הקורא לוחץ בתפריט על הזמנת ספר 2. המערכת מציגה קטלוג עם מסך חיפוש 3. הקורא מזין את שם הספר 4. המערכת מציגה התאמות עם מיקומם 5. הקורא בוחר התאמה ובקשה לשמירה 6. המערכת מאשר את ההזמנה ומציגה את הקטלוג בחזרה
הרחבות (שגיאות)	2א. פג תוקף ה-login 2א.1. המערכת מחזירה את הקורא למסך הכניסה 2א.2. הקורא מתייאש או מנסה שוב 4א המערכת אינה מוצאת את הספר 5א.1. ...
תרחישים חלופיים	3. הקורא מזין מחבר או נושא

הערות:

- תנאי קדם – מה שאנו לא רוצים להתעסק בו בתרחיש.
- טריגר – מה גרם לתרחיש לקרות.

צעדים ליצירת תרחיש ביצוע:

1. זיהוי שחקנים ומטרותיהם:

א. אלו אנשים, מכונות ומערכות נוספות יהיו בקשר עם המערכת שלנו (שחקנים).

ב. מה כל שחקן צריך שהמערכת שלנו תבצע.

ג. כדאי גם לפרט את המטרות של בעלי עניין אחרים.

2. יצירת דיאגרמת תרחישים - מפרטים סידרה של תהליכים שקורים בין השחקן למערכת לקבלת התוצאה.

3. פירוט לתרחישים פורמליים \ לא-פורמליים - במקרה שלנו צריך לפרט רק על 2 use-case.

אג'ייל: סיפורי משתמשים

מסתמכים על כך שעולם הדרישות מאוד מפותח.

1. הגדרת דרישות ברמה כללית בלבד.
2. כוללים מספיק מידע שיאפשר הערכה למימוש.
3. בד"כ קצרים מתרחישי שימוש –יכולים להיות כותרת של תרחיש.
4. תזכורת לשיחה עם הלקוח.

דוגמאות :

- הקורא יכול להשאיל ספר באופן מקוון.
- המערכת שולחת תזכורת במייל כאשר תאריך ההשאלה פג.

כל מיני שיטות לפתח את זה:

הפורמט:

בתור (שחקן) **אני מעוניין** (בפעולה מסוימת מהמערכת) **כך ש** (מטרה מסוימת)

לדוגמא:

בתור חובבת מוסיקה **אני מעוניינת** לראות את הכותרים האחרונים **כך ש** אוכל להזמין וליהנות מהדיסק של ...