שיעור 13 · 1.11

ניהול דרישות, User Stories ו-MVP

3:01 שעות הקלטה, בלי פיצול, ומתוכן 35 דקות בלי קול בזמן הפסקת ארוחת הערב. בכיתה ובזום. איך מפת הדרכים יורדת לפיתוח: סוגי דרישות, MRD ו-PRD, אפיון אמיתי של הזמנת תרופות הביתה, use case ו-user story, ומה נחשב MVP. בסוף, ההנחיות למצגת הסיום. ההקלטה נגמרת כשהקבוצות נכנסות לחדרים.

להאזין לשיעור כ-7 דקות

0:00 / 0:00

מפת השיעור 18 חלקים ▲

דקות	מה קורה שם
0–8	ההקלטה רצה לפני תחילת השיעור, בלי קול
8–22	פתיחה, ואיפה כל קבוצה עומדת עם ה-OKR ומפת הדרכים
22–28	הדרך עד עכשיו: מהבעיה ועד now, next, later
28–33	ניהול דרישות: לבקש את המספרים
33–39	**ארבעה סוגי דרישות, ומי כותב כל אחד**
39–45	MRD ו-PRD
45–60	**אפיון אמיתי: הזמנת תרופות הביתה**
60–79	**PRD או ג'ירה, ולמה עדיין כותבים מסמך**
79–84	מה נכנס ל-PRD, פיצ'ר במוצר קיים, ודו-משמעות
84–96	ניתוח מערכות: תרשימים ו-use case
96–106	סרטון: הוראות מדויקות לכריך
107–141	הפסקת ארוחת ערב. ההקלטה רצה בלי קול
141–148	חזרה, וניהול פרויקטים בקצרה
148–156	**User story: המבנה וקריטריוני הקבלה**
156–162	**MVP: מה נחשב, והפינג**
162–167	המניפסט האג'ילי, סקראם וקנבן
167–175	MVP בתמונה, שיפורים שגורעים, מהירות מול איכות
175–181	ההנחיות למצגת הסיום, וחדרים

הדרך עד עכשיו

מפת הדרכים נסגרה בשיעור הקודם. רוב הקבוצות כבר עברו את ה-OKR ובנו יוזמות. קבוצה שעוד לא הגיעה ל-OKR קיבלה המלצה להיפגש גם מחוץ לשעות הקורס: השעות בכיתה לא יספיקו כדי לסגור את הפער.

לפני שנכנסים לדרישות, המרצה עברה על מה שכבר נבנה:

1. **צרכי המשתמש, השוק והתחרות.** אילו פיצ'רים יש למתחרים, אילו pains הם פותרים ואילו לא
2. **Value Proposition Canvas.** מה שכולם עושים הוא ה-must, הסטנדרט, וחייבים אותו גם אצלנו. **הייחוד בא מעליו:** מענה לכאבים, או gains שאחרים לא חשבו עליהם
3. **החזון וה-OKR.** לקוחות חדשים, שיחזרו, שיהיו פעילים כל יום, שישלמו, ושותפים עסקיים
4. **הפיצ'רים מול היעדים.** מכאן המודולים המרכזיים
5. **מסע הלקוח: איפה תופסים אותו.** בפרויקט של אחת הקבוצות, בתחום הטיפוח: עוד לפני שפגש קוסמטיקאית? אחרי? **מתי נכנסים לתהליך היא שאלה מהותית**, וממנה יודעים אילו פיצ'רים יבדלו

ושרשרת אספקה: בכל תחום יש שרשרת, ולאורכה ספקים דיגיטליים יותר ופחות. אפשר להתחיל בנקודה אחת ולנוע לאורכה, ולעזור לחלק מהספקים להיות דיגיטליים יותר. כמו רשת המזון שהמוצרים התחליפיים שלה הכניסו אותה לייצור: היא לא רק מוכרת, היא גם מייצרת, ולכן מרוויחה יותר.

משם יוזמות, שמתורגמות לפיצ'רים שמקדמים יעד, ומפת דרכים: **now, next, later.** המרצה סיפרה שכשהקימה מוצר בארגון גדול, לקחה את השלב הראשון ודייקה אותו עוד יותר, וקראה לו **שלב 0**. זה היה ה-MVP שלה.

והשאלה שפתחה את השיעור: **נניח שה-now, next, later אושר. איך זה יורד לפיתוח?** בשם של פיצ'ר ושתי נקודות עליו?

ניהול דרישות

הרבה פיצ'רים לא מגיעים מאיתנו, אלא **מדרישות**: של לקוחות, ושל לקוחות פנימיים שמגיעים מהשטח. מנהל המוצר לוקח את הדרישה, מתרגם אותה, בודק אם היא נכנסת לתוכנית העבודה ומה היא מזיזה הצידה, ואז **מאפיין אותה מול הפיתוח.**

- **מפת הדרכים היא תמיד באחריות מנהל המוצר**, או ה-product owner, ותמיד מתעדכנת. זה כבר לא רק מוצר חדש: זה העבודה על מוצר קיים
- **product discovery לא נגמר.** גם בתחום שאתם מכירים, כשמגיעים לדומיין חדש בתוכו, בודקים את המספרים. לא לוקחים כמובן מאליו

לבקש את המספרים

> **הטיפ הכי טוב שאפשר:** כשמישהו בא עם "יש לנו כזה עומס", מבקשים ממנו את הנתונים. לפעמים יש מועקה גדולה, ומאחוריה מספר קטן.

הדוגמה ממשרד הבריאות: אנשי הסיעוד אמרו שיש להם מלא פניות. כשבדקו את המספרים, היו להם כ-6,000 פניות, מתוך מוקד של 100,000. **טיפה בים.**

ויש שתי תוצאות אפשריות: - **הם הולכים לבדוק, ואין כלום.** הם היו בשיטת "ננסה, תפתחו לי". הם לא חוזרים, ואין פיתוח - **הם חוזרים עם מספרים**, ומתברר שיש בעיה שלא ראינו

ולכן **לא להתבאס מדרישות.** הן מביאות ידע על בעיות של אנשים.

ואין תבנית אחת. לכל חברה ולכל מוצר יש מסמך אפיון משלו. מישהו בכיתה אמר שהתבנית הכי טובה שלו היא דף וורד ריק. גם זה אפשרי, אבל יש דברים שחייבים להיות בפנים.

סוגי הדרישות

סוג	מה זה	דוגמה
Business requirement	למה הארגון צריך את הפתרון. נגזר מהיעדים העסקיים	להוריד ב-30% את זמן הטיפול בפניות שירות
User requirement	מה המשתמש צריך	נציג שמוצא את כל ההיסטוריה של הלקוח במקום אחד, ולא רץ בין חמש מערכות
Functional	מה המערכת צריכה לבצע	המערכת תאפשר לפתוח קריאת שירות ולשלוח אותה לנציג מסוים
Non-functional	מה שהמשתמש לא מרגיש, אבל חייב להיות	סוג מסד הנתונים, קיבולת השרתים, שלוש סביבות: פיתוח, בדיקות וייצור

ולצידן, אילוצים: רגולציה שצריך לעמוד בה, ואילוצים טכנולוגיים, כמו סביבה שסגורה לאינטרנט.

"של הלקוח או שלנו?" שאלה מהכיתה. Business requirement הוא **שלנו, כחברה**: באיזה מדד המוצר או המחלקה צריכים לעמוד. זה יכול להיות יעד מתוך ה-OKR. הדרישה של הנציג, לפי המרצה, היא בעצם של מנהל המחלקה: כך הוא יעמוד ביעד העסקי. פעם בנו מערכת לכל פרויקט, ועכשיו יש מיליון פרויקטים פזורים, וצריך לתכלל.

ואיך זה מתחבר: כדי להוריד 30% מזמן הטיפול, הפנייה הראשונה לא תגיע לנציג Tier 1. משהו יבדוק אותה, יטפל לבד במה שאפשר, ויעביר ל-Tier 2 רק כשצריך. ואפשר גם צ'טבוט שעונה מיד.

מי כותב את הלא פונקציונליות

הפיתוח, מול אנשי התשתיות. הרבה פעמים מנהל המוצר לא כותב אותן בכלל.

- **"SQL? איזה מסד נתונים?"** שאלה מהכיתה. **הפיתוח מכתיב. אתם רק מוודאים שהוא הכתיב**
- **ועוד תיקון:** זה לא backend. backend הוא גם ממשק. **זו התשתית**

אצל המרצה, ראש צוות התשתיות היה מגיע אליה ואומר "לא קיבלתי". הוא לא דיבר ישירות עם הפיתוח, ורצה שהיא תתכלל.

לא מיקרו-מנג'מנט, אלא ניהול התהליך. לבדוק שהדרישות עברו, גם בלי לנהל כל פיפס. כך אתם ה-go to person בכל מה שקשור למוצר.

MRD ו-PRD

	MRD	PRD
השם	Market Requirements Document	Product Requirements Document
על מה	השוק, המתחרים, הפרסונות. כל מה שעשינו ב-discovery	מה מפתחים: אילו מודולים יהיו, ואיך המערכת עובדת
הסיפור	המוצר ברמת השוק	הדרישות של הלקוח, ה-MRD והמחשבות שלי, מתורגמים למערכת שעובדת

ומי כותב? פעם ה-MRD הגיע מהשיווק. לשיווק היו שני כובעים: המחקר, השוק והמתחרים עברו בהייטק ל-business development, והשיווק נשאר עם המוצר הקיים, כמו שר חוץ שמשווק את מה שיש. **היום מנהל המוצר עושה את שניהם**, ברבדים אחרים.

מה לא נכנס לגוף ה-PRD

ה-PRD מתפתח יחד עם המערכת. היעדים יכולים להשתנות משנה לשנה, והמערכת נשארת. מפת הדרכים משתנה כל הזמן. **לכן יעדים, מפת דרכים ולוחות זמנים לא נכנסים לגוף המסמך**, אלא כנספח או כהפניה למסמך אחר. באפיון של פיתוח מסוים, לעומת זאת, לוחות זמנים כן נכנסים.

והאינסטינקט: כמה קבוצות התחילו את הפרויקט בתיאור המערכת, ונעצרו: קודם הצרכים והשוק. לפי המרצה, **האינסטינקט שלנו הוא לתת פתרונות מהר.** את הרעיונות כבר יש לכם. קודם להבין את הבעיה ולשאול את השאלות הנכונות.

אפיון אמיתי: הזמנת תרופות הביתה

המרצה הראתה אפיון שכתבה ב-2019, מטעם ספק, לקופת חולים: אתר שבו מזמינים תרופות מהבית ומקבלים אותן במשלוח. **אז זה היה חדשני.** היו רגולציה והרבה נושאים בזמן קצר מאוד, ועבודה מול כעשרה צוותי פיתוח, של הספק ושל הקופה.

קודם ה-flow

הדבר הראשון היה ה-flow. איך הלקוח מגיע לאתר של הקופה ועובר משם אלינו. איך הוא בוחר, עם כל צמתי ההחלטה: התרופות שלו או של הילדים, מה במלאי ומה לא. מה קורה כשההזמנה מתקבלת, מי אוסף, איך הרוקחים בודקים שזו התרופה הנכונה, המשלוח, והחיוב. **כל ריבוע בתרשים הוא פונקציונליות שצריך לפתח.**

ה-UX בא אחרי. רק כשה-flow אושר מול מערכות המידע והתפעול, התחיל עיצוב ה-UX, במקביל לאפיון. בהתחלה היא עשתה אותו בעצמה, ובשלב מסוים עם חברה חיצונית.

מסמך מתגלגל

האפיון יצא בגרסאות: 1.7, 1.8, 1.9. **בכל פעם שהגיע מידע** על הממשקים, איך שולפים את התרופות, איך נראה ה-handshake בין המערכות ומה עם האבטחה, נוספה גרסה.

שאלה מהכיתה: פיצ'ר חדש במוצר קיים הוא גרסה חדשה של ה-PRD? **כן. מסמך מתגלגל, אבל ברמה גבוהה.**

ופרט מ-2019: היא כתבה שם פרסונות ו-user stories, והצד השני צחק. מישהי ממערכות המידע שאלה את הבת שלה, שעובדת בהייטק, למה כותבים את זה. התשובה: ככה מקובל עכשיו בהייטק.

מה יש באפיון

הפרק	מה בו
תיאור כללי, יעדים, מצב קיים, בעיות	הקלאסי. מגדיר את עולם הבעיה
שוק ותחרות, ומערכות משיקות	מי מתחבר למי
הבעיה של הלקוח, ומסע הלקוח	ה-Value Proposition Canvas פחות: הוא כלי פנימי. מסע הלקוח כן
מונחי יסוד	מילון. מי שחדש קורא, ומבין מיד את השפה של המערכת
תיאור המערכת, ורשימת המודולים המרכזיים	המודולים הם ה-use cases המרכזיים. באג'ייל: ה-epics
מפרט לכל use case	המטרה, התיאור, השחקנים, תנאים מוקדמים, הפעילויות והבדיקות

user stories לא נכנסים ל-PRD. הם יושבים בכלי הפיתוח: Jira, או DevOps של מיקרוסופט. **ה-PRD הוא מסמך על**, על האבנים הגדולות שאחר כך יתפרקו.

ואם למוצר שלכם אין PRD: קחו את האתגר וכתבו baseline. איפה המוצר עומד היום, ומה המספרים. כך אפשר להראות מה היה, ואיפה אנחנו עכשיו.

ה-handshake

הלקוח מזדהה באתר של הקופה, ועובר למערכת של הספק ב-SSO. **מי מחליט אם לקוח של הקופה מזוהה?** הקופה. הספק מקבל אותו, וצריך handshake מאובטח מאוד: לפי המרצה, **בריאות מאובטחת לפחות כמו בנקים.**

כאן היא ירדה עד פסאודו-קוד, וזה לא הרגיל. הסיבה: **שתי חברות.** בתוך אותה מחלקה, נותנים לאנשים לדבר. בין שתי חברות:

תובילו את ה-handshake, אחרת זה בלאגן. לשבת יחד, לרשום מי מעביר מה, ולוודא שאין מצב שזה נופל.

PRD או ג'ירה?

שאלה מהכיתה, שהתפתחה לדיון ארוך: אם הכל ממילא בג'ירה, והפיתוח מפרק שם את הדרישה, **מה ה-PRD נותן?**

- **ההנהלה והגורמים העסקיים לא נכנסים לג'ירה.** למנהל שלכם אין שם אפילו משתמש. ואת ה-PRD הגדול שלה ואת ה-flow **אישרו גורמים עסקיים, לא טכניים.** לכן וורד או PDF
- **המכירות** רוצות לדעת מה יהיה במוצר. גם הדרכה על המוצר צריכה מקור
- **קריאה מוקדמת.** ג'ירה אי אפשר לשלוח לעיון לפני פגישה. ב-design review בעולם העסקי יושבים גם אנשים בלי משתמש בג'ירה. לא שמישהו קורא מסמכים לפני פגישה, אמרה המרצה, אבל באמזון דווקא כן
- **העברת מקל.** כשמנהל מוצר מתחלף, ה-PRD הוא מה שמעבירים. **אל תזלזלו בזה:** התחלופה גבוהה, והזמן שהולך על ללמוד מה קורה במערכת נוראי
- **דרישה על משהו שכבר קיים.** הרבה בקשות מגיעות כי אנשים לא יודעים שהיכולת כבר שם. מסמך שמראה מה יש חוסך את הבקשה
- **ג'ירה הוא לא שימור ידע.** הוא הכלי הטקטי של הפיתוח. ה-PRD הוא שימור הידע ברמה גבוהה. **ואפשר לחבר אותו למדדים:** אם פיתוח הסתיים בתאריך מסוים ורואים קפיצה, מתחילים לראות קשר לאורך זמן

איך זה נראה אצלה: יושבים סביב מסמך וורד. נציג הפיתוח, נציג ה-QA, מנהל המוצר, והלקוח הפנימי שביקש את הדרישה בשביל לקוח הקצה. מראים את ה-UX ואת התהליך כמו שהבנו אותו. הלקוח אומר "בדיוק מה שרציתי", ורק אז זה עובר לפיתוח.

ובשורה התחתונה

שתי הדרכים בסדר. אפשר לנהל הכל בג'ירה, עם board טוב. באג'ייל ה-PRD הרבה פעמים מתקצר ונכנס לג'ירה. **אבל רוב מנהלי המוצר שהמרצה מכירה עדיין מחזיקים PRD**, וה-user stories הם הכלי של הפיתוח, שבו הוא עוקב אחרי פירוק ה-epic.

הדרך שהיא ממליצה עליה: מקום אחד, רשימה מסודרת של הפיצ'רים. מה שמאושר נכנס לפיתוח, ובסיום מוסיפים קישור למה שפותח בפועל, כי לפעמים יש שינויים. **ככה לא מעדכנים שני מקומות.**

ומי מפרק ל-user stories? הפיתוח, למשימות בג'ירה. והיום, לפי המרצה, הרבה פעמים בוט: נותנים לו את הדרישה, הוא מפרק ל-user stories ומכניס לג'ירה. סקיל פשוט.

מה נכנס ל-PRD

ההגדרה: מסמך שמתאר את הדרישות העסקיות, הפונקציונליות והטכניות, משמש תוכנית לפיתוח, ומבטיח הבנה משותפת לכל בעלי העניין. **הוא בא אחרי הגדרת היעדים העסקיים וה-discovery.**

הפרקים: Overview, משתמשים וצרכים, Solution scope, use cases ו-flow, פונקציונליות (המרצה מכניסה את ה-use cases לתוכה), UX/UI, דרישות לא פונקציונליות, אנליטיקס, Delivery ו-Implementation, ואיזה מידע שומרים.

פיצ'ר במוצר קיים

לא צריך את כל זה. **מספיק:** מה המטרה מבחינת המשתמש, מי הקהל, הדרישות והפיצ'רים, ה-use cases, ודרישות לא פונקציונליות אם צריך. אם הפרסונות לא השתנו, לא חוזרים עליהן. שיקול דעת.

אבל פיצ'ר גדול מקבל PRD, גם באותה מערכת. הדוגמה: להכניס בוט. פתאום יש דרישות חדשות: אזור שממנו שולפים את התוכן, ומנגנון שמוציא אותו החוצה, כי לא מחברים בוט ישר למערכות הפנימיות. כלים כמו Snowflake משקפים את הדאטה החוצה, ומעליהם ה-LLM. ודרישה כמו: לשלוף דוח מסוים ממערכת ה-ERP. **את כל זה צריך לפרט.**

המלכודת: דו-משמעות

> **כל דבר שאפשר לפרש בשתי דרכים לא כתוב טוב.**

"שיפור חוויית המשתמש": אתה תשפר בדרך אחת, ואני בדרך אחרת. מה כן: **"הכניסה תהיה אוטומטית: המערכת מזהה את המשתמש דרך ה-SSO."**

ועוד שלושה כללים: **להתמקד במה, ולא באיך הטכני** (ראו מה מול איך). לתעדף. ולא לשכוח את הדרישות הלא פונקציונליות.

ניתוח מערכות: מה מספיק להכיר

החלק הזה נוסף כי אחת הקבוצות ביקשה. **המטרה: שתכירו את השפה**, כדי לדבר עם אנשים טכניים.

תרשים	מה הוא מראה	של מי
Flow chart	תהליך: מי עושה מה, ובאילו מערכות	**מנהל המוצר.** משתמשים בו הרבה
Activity diagram	צמתי החלטה: מה קורה אם כן, ומה אם לא. השחקנים יכולים להיות אנשים, מערכות, או AI	הפיתוח, או מנתח מערכות
Sequence diagram	הפעילות של משתמש בין מערכות שונות	הפיתוח

הדוגמה ל-flow chart: הזמנה בארגון. המכירות מוכרות, התפעול בודק במחסן הנתונים שיש מלאי ומספק, החיוב סוגר את ההזמנה, והכספים משלמים לאיש המכירות את העמלה. לא מפורט מה קורה בכל ריבוע, אבל

רואים את כל המערכות שמשתתפות בתהליך.

ובדוגמה ל-activity diagram: הלקוח לחץ על רישום? מתחילים תהליך רישום. הוא כבר לקוח קיים? מדלגים ישר לבדיקה אם עבר אימות.

> **אתם לא עושים את זה בשום פנים ואופן. אבל שזה לא יהיה לכם זר.**

ולתיעוד יש היום פתרון: אם נותנים לכלי AI גישה לג'ירה ולקוד, הוא ממפה את זה בקלות. פרויקטים של תיעוד היו סיוט, לאנשים הטכניים ולמנהלי המוצר. **תנו ל-AI לעשות את זה.**

Use case

תיאור מובנה של איך המשתמשים פועלים מול המערכת, מנקודת המבט שלהם. הוא מצמצם את הפער בין מה שהעסק צריך לבין הביצוע הטכני, **כי המשתמש יושב באמצע.**

הדוגמה מהאפיון: הלקוח מזדהה בקופה ועובר לאתר ההזמנה בלחיצת כפתור, בלי שם משתמש וסיסמה נוספים. **שירגיש שהוא עדיין באתר של הקופה, רק באזור אחר.**

מה כותבים	מה זה
כותרת	על איזה תהליך מדובר
השחקנים	משתמשים, מערכות, או שניהם
המטרה	מה השחקן רוצה להשיג
הצעדים	איך הוא משיג אותה
חריגים	מה קורה כשהתהליך לא הולך במסלול הרגיל
נושאים פתוחים	מה עוד לא נסגר. למשל: ידוע שזה יהיה SSO, אבל החתימה הדיגיטלית עוד לא הועברה
תנאים מוקדמים ומאוחרים	איך הגיעו לכאן, ומה קורה אחרי
קישור ל-UX	ה-wireframe, סגור

תנאי מוקדם להצגת התרופות הוא שהלקוח מזוהה. אחרת אי אפשר להציג לו מה זמין להזמנה. **תנאי מאוחר:** אם זוהה, הוא רואה את העמוד עם כל התרופות.

> **"תביא את כל התרופות" זה לפחות ארבע או חמש קריאות לממשקים של מערכות שונות.** אי אפשר להגיד למפתח שלא מכיר את הממשקים של הקופה "תעשה קריאות לממשקים". צריך לפרט.

אותו דבר בכל ממשק למערכת שאינה של המפתחים שלכם: **זה לא הדומיין שלהם.** בין צוותים פנימיים, הם ידברו. מול גורם חיצוני, יושבים ומתארים.

ה-UX חייב להגיע סגור. היום מציירים wireframe לבד בפיגמה. פעם היה צריך draw.io.

טיפים: - **לפרק.** את החבילות הגדולות, עם האימפקט הגבוה והמאמץ הגבוה, מפרקים כמה שיותר. את מודול התרופות היא העבירה חלקים-חלקים. **מפתחים לא יכולים לקחת בלוקים גדולים** - לוודא כל הזמן מול בעלי העניין - flows פשוטים. יותר מדי חצים זו בעיה - לכתוב מה קורה במסלול החלופי, כשהתהליך הפשוט לא קורה

את זה המרצה לא תרגלה בכיתה: רוב הכיתה מאפיינת ביום-יום. **ההמלצה: בפעם הבאה שאתם מאפיינים לפיתוח, כתבו use cases**, תהליך אחרי תהליך.

הסרטון: הוראות מדויקות

לפני ההפסקה, סרטון על השיח בין מוצר לפיתוח: ילדים כותבים הוראות להכנת כריך חמאת בוטנים וריבה, ומי שמבצע עושה **בדיוק** מה שכתוב, מילה במילה. "קצת חמאת בוטנים" לא עובד. בכל סבב ההוראות נעשות ספציפיות יותר, עד שיוצא כריך.

הקשר לאפיון: לפעמים המפתחים עושים בדיוק מה שאמרתם להם. ולמנהל המוצר הישראלי, לפי המרצה, אפיון פיצ'רים לפיתוח הוא חלק גדול מהיום-יום. יותר מאשר לאמריקאי. ולכן החלק הזה של השיעור היה ארוך מהרגיל.

ניהול פרויקטים, בקצרה

פיצ'ר הוא פרויקט קטן, ומנהלים את הפיתוח שלו כפרויקט. פעם ב-waterfall, היום באג'ייל. השלבים: ייזום, תכנון, ביצוע, ניטור ובקרה, וסגירה. **מוצר, לעומת זאת, לא נסגר.** אם נסגר, כנראה לא עבדנו כמו שצריך.

ומה הגבול: את האפיון הטכני עושה לפעמים גורם טכני, או מאפיין. אפשר להעביר רק את הדרישה ואת ה-UX, והם יפרטו. המרצה פירקה בעצמה כי היא מכירה טכנית, אבל:

> **אני רוצה שהזמן שלכם יהיה במדדים, בלראות איך אנשים משתמשים, ובפידבק מלקוחות. יותר מאשר באפיון הטכני.**

ובכל זאת, צריך הבנה טכנית שמספיקה לשיח טוב. **"אתם לא לקוח, אתם עדיין מנהלי מוצר."**

מפרויקטים למוצר: הרבה מכם מתכננים כל פעם פרויקט קטן, עם דרישות של פרויקט. המרצה רוצה להרחיב את זה **להסתכלות מוצרית מתמשכת**: view של מוצר, שגם ישמש יום אחד מקור מידע ל-AI פנימי. ומהעולם של הפרויקטים, דבר אחד שכדאי לקחת: **ניהול סיכונים.**

User stories

באג'ייל, ה-PRD מתקצר ונכנס ל-user story. אבל גם epic גדול, כמו בניית אתר, עדיין מפנה החוצה: קישור לפיגמה, מה עושים ומה לא, שאלות ותשובות, והפניה לאסטרטגיה, לחזון, או ל-PRD מורחב. **כדי שיהיה קצר ולעניין.**

ההיררכיה

רמה	מה זה
Theme	יעד, Key Result: גידול ברווחיות, במשתמשים, עלייה של 10% ב-Retention
Initiative	יוזמה שמקדמת אותו
Epic	פיצ'ר שמכיל כמה דברים
Story	יחידת העבודה הקטנה ביותר של מנהל המוצר
משימה	מה שבאמת נכנס לפיתוח. הפיתוח מפרק את ה-story למשימות

יש ארגונים שמכניסים באמצע **Feature**, שמתפרק ל-user stories. זה משתנה מארגון לארגון.

הגודל: story הוא בערך שבועיים בצוות, בין שבוע לארבעה. **שבועיים הם משך, לא סכום** של העבודה של כל המפתחים. **ואותו מבנה נכון ל-story, ל-epic ול-initiative:** כותבים, ואם יצא גדול מדי, מפרקים עד שבועיים.

story נכתב מהעיניים של המשתמש, לא של מי שבונה. לא שמות של מערכות, אלא מה הלקוח רוצה להשיג.

המבנה

As a	מי מנסה לעשות את העבודה
I want to	מה הוא רוצה לבצע
so that I can	ולמה

דוגמה: כמנהלת, אני רוצה להבין טוב יותר את מצב ההתקדמות של הקולגות שלי, כדי שאוכל לדווח סטטוס מדויק.

מה עוד נכנס: הדוגמה של השיתוף

פרסונה רוצה לשתף מוצר עם חברים. זה לא נשאר משפט: **מצרפים wireframe.** כפתור שיתוף, שפותח את אפשרויות השיתוף. **ג'ימייל, וואטסאפ וסלאק חייבים להיות שם.**

ואז הפיתוח אומר: יש לנו רכיב שמשתף גם לפייסבוק.

> **אין מצב. תורידו.** אפשר להביא לי הצעות של רכיבים קיימים, אבל כל ערוץ שיתוף אני מאשרת.

כותבים לא רק מה כן, אלא גם מה לא: "לא יופיעו רשתות חברתיות", או "חוץ משלושת אלה לא יופיע דבר בלי אישור".

קריטריוני קבלה

Given	ההקשר: הלקוח בעמוד של מוצר
When	הפעולה: לוחץ על כפתור השיתוף
Then	מה צריך לקרות: נפתחות אפשרויות השיתוף, ובהן מייל, וואטסאפ וסלאק

זה מה שה-QA בודק. ועוד דוגמה: משתמש מעלה מסמכים. צריך לבדוק שהמסמכים תקינים, כדי שלא ייכנס למערכת משהו לא תקין. **כל המידע הזה צריך להיות בתוך ה-user story**, יחד עם definition of done ומשימות.

MVP

Minimum Viable Product. הגיע מ-Lean Startup, והאג'ייל אימץ אותו. **המינימום שנותן ערך למשתמש.** במקום אפיונים מסיביים, ניסוי קטן ומהיר שבודק שהפיצ'ר באמת עוזר.

> **המאמץ במינימום, אבל הערך לא. אי אפשר לצאת בלי ערך.**

מה נחשב MVP

סקר משתמשים? ראיונות? מי שמגיע מייעוץ יגיד שגם זה MVP, כי זו בדיקה. ראיונות באמת נותנים תובנות, והלקוח מרגיש שמתייחסים אליו. **מסמך, לעומת זאת, הוא לא MVP:** הוא לא נותן ערך ללקוח.

ההגדרה שהמרצה רוצה שתעבדו איתה: תוצר התוכנה הקטן ביותר שעדיין נותן ערך. התוצר הראשון שיוצא מול הלקוח.

הפינג

השאלה לכיתה: מערכת חדשה תתחבר אליכם יום אחד ב-API, ושלחה רשימה של שדות שהיא רוצה לקבל. זה עוד לא קורה. מה ה-MVP?

התשובה מהכיתה: פינג. בדיקה שהמערכות מדברות. אם תפתחו עכשיו את כל השדות, ייתכן שכשהצד השני יגיע הוא יגיד: לא A, B, C ו-D, אלא A, C, X ו-Y. **ובזבזתם זמן פיתוח.**

ומקרה אמיתי, לפי המרצה: ביקשו מצוות להכין ממשק לפי אפיון מפורט. הם הקימו רק תקשורת בין המערכות, ביט אחד שעובר. שנתיים אחר כך הצד השני עוד לא בא לבקש את השדות, והעניין נעלם. **עשו את המינימום, ומנעו בזבוז.**

המניפסט האג'ילי

בשיעור על אג'ייל ילמדו אותו לעומק. ארבעה צמדים:

מה חשוב יותר	ממה
אנשים ויחסי גומלין	תהליכים וכלים
תוכנה עובדת	תיעוד מקיף
שיתוף פעולה עם לקוחות	משא ומתן על חוזים
תגובה לשינוי	מעקב אחרי תוכנית

העמודה הראשונה חשובה יותר, אבל השנייה לא נעלמת. שמים דגש על תוכנה עובדת, ולא מאבדים את כל התיעוד. מגיבים לשינוי, אבל לא משנים כל הזמן: עדיין צריך תוכנית, מפת דרכים. ולקוחות לא נמצאים בג'ירה, אז עדיין צריך מסמכים לשלוח להם.

> **אם צריך לשלוף את החוזה, התקשורת כבר לא טובה.**

ולמה זה משתלם: ב-waterfall מגלים בעיה מאוחר, ואז התיקון יקר. באג'ייל המחזורים קצרים, ומה-MVP כבר רואים אם בכלל צריך את זה. **המטרה היא לא להיכשל, אלא להיכשל מהר ובקטן**, כדי שלכישלון לא יהיה אימפקט גבוה.

סקראם וקנבן

	סקראם	קנבן
מי מנהל	Scrum master, ו-Product owner שאומר מה צריך לעשות	אין product owner
איך עובדים	ספרינט: תכנון, עבודה, רטרוספקטיבה	מאגר של משימות מאושרות, והפיתוח שולף ממנו

ההמלצה, ממי שעבד בחברה עם שתי השיטות: פיתוח פיצ'רים חדשים בסקראם, כי מישהו צריך לנהל אותו. **תמיכה ובאגים בקנבן:** כל אחד שולף, בודק, ואם זה לא שלו, מעביר לגורם הנכון. המטרה היא להוריד כמה שיותר באגים. רוב הכיתה עובדת בסקראם, ומעטים בקנבן.

MVP בתמונה

המרצה הראתה תמונה עם שלוש שורות של שלבים, ושאלה למה רק אחת מהן היא MVP.

שורה	מה רואים	MVP?
הראשונה	רק בשלב הרביעי יש מכונית שנוסעת	**לא.** שלושת השלבים הראשונים לא נותנים ללקוח כלום
השנייה	כל שלב הוא מוצר אחר	**לא.** לפי המרצה, כל שלב הוא מוצר אחר ושוק אחר, והוא לא מוביל לשלב הבא
השלישית	מכונית בסיסית, עם מקום לנהג בלבד. אחר כך תא מטען, אחר כך עוד נוסעים, ובסוף מראה יוקרתי	**כן.** כבר מהשלב הראשון: ארבעה גלגלים, נוסעת ממקום למקום

ה-MVP נועד לאסוף כמה שיותר מידע על הלקוח: האם בכלל רוצים את זה. הוא חייב להתחבר לערך המרכזי, ה-core value, ובמיוחד לתועלת הייחודית שלנו. והוא נותן אימפקט לארגון: ידע חדש. **ואז איטרציות.**

כשהשיפורים גורעים

משפרים, ומשפרים, ומוסיפים יכולות. **ובשלב מסוים השיפורים כבר גורעים:** העמסנו אייקונים ויכולות, והמשתמשים לא רואים את הפשוט. הדוגמה של המרצה: ביט שמרה על ההעברה כבסיס, ולא הלכה לאיבוד. פייבוקס, לדעתה, הלכה לאיבוד לגמרי.

מהירות מול איכות

מתי	מה
ניסוי, מוצר בהתחלה	לא נורא שזה לא מושלם. תעשו חדשנות. **אבל לא לגרום ללקוח סבל:** צריך ערך, ו"וואו, אני רוצה עוד"
פיצ'ר במוצר קיים	חייבים לאזן. בפלטפורמה האיכות חייבת להיות גבוהה מאוד: פיצ'ר שנבנה מהר מדי יכול להאט את כל הפלטפורמה

מצגת הסיום

המרצה עברה על מסמך ההנחיות שהועלה. **רוב המצגת כבר אמור להיות מאחוריכם.**

חלק	מה בו
המצב הקיים: שוק ותחרות	גודל שוק, צרכים, שינוי טכנולוגי בשוק, מתחרים. כבר עשיתם בתרגולים. לתקן לפי ההערות
הלקוחות והצרכים	פרסונות, צרכים, pains ו-gains. **תובנות מרכזיות מהסקרים והראיונות שעשיתם:** המחקר עצמו, לא איך תעשו אותו. פרסונה מורחבת לאחת או שתיים המרכזיות, והשאר בקטן. **מסע לקוח או Value Proposition Canvas**, לא חייב את שניהם: לפעמים זה כפול
חזון המוצר	**משפט אחד, שמקריאים כמו שהוא, ונותנים לו מקום.** "זה הדגש הכי גדול שלי אליכם"
OKR ומפת דרכים	וגם ה-North Star Metric, מדד הצמיחה. במפת הדרכים: הטבלה עם כל הרשימה, תעדוף must ו-should עם האימפקט, ואז now, next, later

את זה צריך לסיים עכשיו. מכאן ועד הסוף נשאר בעיקר ה-MVP. **ודרישות לא כותבים למצגת:** המרצה סומכת עליכם שתדעו לכתוב אותן.

מה הלאה

- **ביום ראשון הבא, בכיתה:** יועץ אג'ייל מגיע ללמד אג'ייל. המרצה ממליצה מאוד להגיע, **ולהביא שאלות ספציפיות** מהעבודה מול הפיתוח
- **התרגיל:** להשלים את מפת הדרכים: תעדוף must ו-should לפי אימפקט, ו-now, next, later
- **בהמשך:** MVP, כולל בנייה בכלים כמו Lovable, ו-Go-To-Market

---

הוכן על ידי ניתאי אלנקרי · סיכומים אישיים מתוך ההקלטות, ולא חומר רשמי של הקורס.