ויטלי ניהול משימות

יום חמישי 16 דצמבר 2021 11:05

What “risk” means

- Risk of being late with your feature
- Risk of providing not optimal solution for the task
- Risk of getting stuck on something during development
- Risk of having unexpected things to do during feature development
- Risk of disappointing your stakeholders - PMs/team mates/managers.

איך מונעים את הסיכון כדי לא לאכזב את האנשים שמחכים לתוצר של העבודה

מי שמגדיר את הדרישות PM
Product request זה הבקשה. פיצ'ר, התנהגות, ביצועים, כל דבר.

כל יום במחשבה של עמידה בזמנים, מה קורה אם צפוי עיכוב, האם המשימות שיש בפייפליין הן היעילות ביותר לסיום הבקשה.
כל הזמן לתעדף. לפי:
מה המהות המוצרית? מה הקטעים הכי קשים טכנולוגית- אולי בהם להתקדם?
בכל רגע אם ישאלו אותי איך התקדמתי, מה אני יכול להראות? אולי להתחיל מה UI או מוק בלשהו שירגיע אנשים.

כשבונים פיצ'ר גדול, כל הזמן לדמיין שזה מערכת אינסטלציה/צינורות, מידע שעובר- ולהבין אם כל צינור שעשיתי עד עכשיו, מים עוברים דרכו?
לא לחכות ליום האחרון לעשות אינטגרציה!
על ידי מידע מוק, או טסטים, או לייצר endpoint UI כדי לבדוק פונקציונליות כמה שיותר מהר.

The mindset:

- Risk management starts from day 1
- Risk management starts from day 2
- Risk management starts from day 3...
- Think you are already late - what is important now? What is the main tech-part, what is the product essence?
- What can I show today if I am asked to share progress?
- Strive push water through the pipelines

The formula:

1. Understand it
2. Break it
3. Prioritize
4. Execute
5. Go to 3

Strive to understand:

- Product & business motivation
- Core of the feature - the essence, the hypothesis
- Alternatives discussed
- Context - is it a POC/test/showing abilities?
- Stakeholders, decision makers and other interfaces
- Tech changes required (architecture, design, API's, frameworks) vs. your knowledge in those areas
- License/legal/3rd parties involved etc...

לשאול את השאלות. לאתגר, האם בדקנו חלופות. האם מותר לנו לעשות משהו. להבין את ההקשר, האם זה דמו? מי צריך? מי יראה?

איך מפרקים משימה:
גם להפגש עם מעצב, או קוד ריוויו או איש משפטי גם זה שלב בתהליך. לפרק הכל לתתי שלבים.
קודם לפרק מוצרית- לפרק ליכולות קטנות.
כל יכולת, איך לספק אותה טכנית.
ואז לתעדף.

How to break a feature?

1. Break the product
2. Break the code
3. Break the process

Break the product to sub-features, for each of those break the tech aspects, prioritize, plan and execute

איך לשבור דרישה מוצרית?
כנראה שלא הכל חשוב בפנים. תלוי מה רוצים. מה המטריקה שרוצים להזיז? שיגיעו יותר יוזרים? שיותר יוזרים יקנברטו?
איך עוד שוברים= איזה קומפוננטות יש? איזה התנהגויות יש?
איך יודעים אם אפשר לשבור משהו לשלב נפרד משלו:
אם יש לקומפוננטה/כפתור/יכולת הזו, יכולת קיום עצמאית במקרה או דוגמה או עמוד או תהליך אחר- אפשר להפריד את זה.
נגיד מבקשים מסך עם לוגו, מקום להקליד ביטוי לחיפוש, כפתור, אפשרות חיפוש קולי וכו , למעלה כניסה לאיזור האישי (גוגל הומפייג).
מה המהות? הסרץ' בר! כל השאר יכול להיות פיצ'ר בפני עצמו.
אפשר לשבת עם הפרודקט להסביר מה לדעתנו הכי חשוב, אחרי זה אולי התנאים המשפטיים, אולי אחב זה האינטגרציה עם האיזור האישי, אולי אחב זה החיפוש הקולי וכו'.

מהמהות => אל הדברים הפחות חשובים.
מימשנו חיפוש טקסטואלי. אפשר להתקדם.
אפשר להראות התקדמות, ולנהל מו"מ לגבי שאר המשימות.

How to break code?

- Classic product requirements can be broken into:
  - UI/UX
  - Logic
  - Data
- Each of those has its own representation layer in your codebase consists of its own architecture, languages and frameworks.
- Think what should happen on each layer to make the feature happen

פירוק קלאסי ל- 3 דברים. ממשק גרפי, לוגיקה, טרנספורמציה של דאטה.
לכל אחד יש יצוג בקוד בייס, שפה בה בחרו לממש את זה, פריימוורק וכו.

אחרי הפירוק הזה אפשר לשאול עוד שאלות יותר פרטניות, לדוגמה מעולם הווב:
האם הלוגיקה חדשה?
איזה פונקציות נממש?
כנל דאטה, דאטה זה כל דבר- יכול להיות גם טקסט או מספר שעובר בקוד.
האם צריך קווורי חדש? הדאטה הקיים מספיק או צריך להתחיל לשמור מידע חדש? (מעבשיו צריך גם לשמור תאריך לידה לדוגמה). זה אומר שינוי אחורה? האם במערכת שלי אפשר לשמור מידע חדש רק למשתמשים חדשים?
האם כותבים טסט? (התשובה היא כן)

Questions to ask (web)?

- Is there a new page? Or just a component on the existing page?
- Do you need a new route?
- How are you going to model your new component? What functions it needs? Which sub-components?
- What about the logic? Is it a new one? Where are you going to implement it? Which files are going to be changed/created, which functions - each of those changes can be a subtask in your breakdown
- Same questions about the data and its flow, do you need a new document? Change document? New table? Which layers you have to communicate with the data? Query changes? Index changes? Migration of old data?

פילוסופיה-
שבירת המשימה כך שכל צוות יכול לקבל אותה ולהבין ולבצע.
מאפשר גם ביצוע מקבילי.
ולחגוג הצלחות.

Code breakage philosophy?

- Strive to break code to a level that each task could be executed by another team member.
- Each task should be self-contained with a clear goal.
- The more tasks you have the easier it will be:
  - To track progress
  - To parallelise work
  - To celebrate accomplishment

איך שוברים את התהליך?
לרוב יש כמה שלבים,

דיזיין טכני לפיצ'ר (חלוקה והחלטה על אופן מימוש) -
משתפים עם אחרים ומתקנים.
קוד ריוויו + תיקונים
מקבלים פידבק ממזמין העבודה + תיקונים (וגם ממעצב)
גם לקראת דפלוי יכול להיות שצריך עוד קונפיגורציה..
כל אלה הם מאורעות שמצריכים עוד אנשים, פגישה, זמן וצריך להתכונן אליהם מראש.
אם את הפיצ'ר צריך לסיים בחמישי, אז שבוע לפני נקבע פגישה לקוד ריוויו ליום רביעי.

How to break the process?

- Design + design fixes
- Code review + fixes
- Product review + fixes
- Designer review + fixes
- Deployment + Configuration
- Most of those need your and others people time

What should be a process sub-task?

- Code review fixes
- Product review fixes
- Designer review fixes
- Deployment
- Configuration

כל יום צריך לחשוב מה יותר חשוב.
וזה יכול להשתנות!!

How to prioritize?

1. By essense of the feature - work with your Product to understand what is the most important parts and what are the secondary - those may change as you closer to the deadline
2. Strive to eliminate the unknowns first
3. Keep basic functionality production ready, each time choose the next sub-feature to enrich the experience - **DO NOT CREATE FEATURES DEPENDENCY**.

אם יש מרכיב unknown צריך להתחיל עם זה מוקדם!
נגיד צריך להתממשק עם משהו שטרם עשינו. להתנסות עם זה כמה שיותר מוקדם - כל עוד הם במהות.
לכן צריך לתאם ציפיות עם איש המוצר ולהיות כנים עם עצמנו לגבי מה אנחנו מכירים ומה לא.

לחתור שכמה שיותר מהר הדבר הבסיסי עובד.
ואז אפשר להמשיך למשהו יותר מעשיר.

והכי חשוב לא ליצור dependencies.
כי אז כל עוד משהו לא עובד, משהו אחר לא יכול להגיע לפרודקשן.
יש לזה כמה פתרונות.

מיזעור סיכונים
הפחתת אנוואון

פוש חלקי לפרודקשן- אם מוסכם. לעלות עם המהות והיוזרים יהנו מזה, ואחב השאר.
פוש חלקי יכול להיות ברמת פיצ'ר או ברמת מימוש חלקי. אולי קוד עובד אבל עדיין אין UI. אפשר לעשות פוש גם בלי זה.
לפעמים עושים את זה כי זה עבר כבר קוד ריוויו, טסטים, ואפשר ככה להתקדם כל הזמן בצעדים קטנים למימוש ולא לעלות עם מלא קבצים בבת אחת שמהווים סיכון גדול יותר (ברמת הרייויו שיתארך או טסטים ואז מלא זמן אין כלום שעולה)
אפשר להעלות עמוד בלי לתת לו לינק ואי אפשר להגיע אליו. ככה אפשר בחלקים לעלות.
אפשר גם לעשות ריוויו חלקי- על הקוד, העיצוב וכו' . לא לחכות ליום האחרון.

להתייעץ לגבי מה שלא טריוויאלי- אם בחרנו פתרון לא טריוויאלי! (טכנולוגיה חדשה, חבילה חדשה וכו) יכול לחשוף אותנו לסיכוני אבטחה, משפטית לייסנס, גודל באנדל שחורג או כל מיני דברים. להתיעץ עם טק ליד או מישו.

How to minimize risks?

- Eliminate the unknowns of feature essense ASAP
- Partial push to production
- Partial reviews - code/designer/PM
- Consult on non-obvious solution you choose!
- Partial code on production
- Handle the unknowns first

מה לעשות אם נתקענו

What to do if you stuck?

- Try solve by your own (set timeout 30 min max)
- Check google
- Check internal documentation
- Ask team members
- Ask company expert
- Raise a flag to your manager

אם נעזרנו והרמנו דגל ולמרות זאת לא עומדים בזמנים -

- לייצר דינמיקה עם הפרודקט כדי להבין ממש מה המהות, ולהיות פרואקטיבי בלהציע גרסה רזה. להציע אפשר לעלות בנתיים רק עם X ?
- אפשר לפנות לראש צוות ולהציע לבדוק אם אחרים יכולים להשתלב בפיתוח? או לעזור בתעדוף וכאלה?
- האם אפשר להחליף משהו שיהיה סטטי במקום דינמי? מידע שמתעדכן אחת ליום, אם לא מימשנו עדיין, אפשר בינתיים לעדכן ידנית אחת ליום.
- לישון במשרד וללמוד להבא לתמחר אחרת את המשימה

פרויקט - על מה הוא מסתכל

- יותר מעניין אותו מה הבנאדם למד. הוא מכיר לרוב דרך תפוח או מיסטרביט ולא מפרסם מודעת דרושים. מה מרשים מגייסים אחרים - משהו שהוא מוגמר. יש 3 השכבות, מידע, לוגיקה, UI. ואפשר לדבר עם בנאדם לא רק טכנולוגית אלא גם להבין מוצרית מה המחשבות שלו.
- מעניין אותו הטכנולוגיות שלו - מונגו, ריקאקט, נוד, אפילו AWS עוד יותר. בנאדם שמדבר בלהט - או מהטכנולוגיה או מהבעיה שבא לפתור או מה UI.

מה כן אפשר ורצוי להשלים מבחינת ידע תאורטי

- מבני נתונים וזמני ריצה
- הבנה יותר כללית של מערכות הפעלה ותחשות שליטה על אקוסיסטם טכנולוגי