

איך בונים אפליקציית AI ששווה 650 מיליון דולר — Jake Heller על CoCounsel ו-Casetext

איך בוחרים רעיון, איך בונים AI אמין בעזרת evals ואיך מוכרים מוצר שמחליף או מסייע לבעלי מקצוע. השיעורים מהאקזיט של Casetext ל-Thomson Reuters

Jake Heller · Y Combinator AI Startup School (זוהה לפי הבאנר על הבמה; הסרטון הועלה מחדש ב-X על ידי Defileo) · תאריך האירוע המקורי לא מופיע בהקלטה; פורסם ב-X ב-21.05.2026

משך: כ-39 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו הרצאה של Jake Heller, מייסד ומנכ"ל Casetext, בכנס AI Startup School של Y Combinator. Heller הוא מתכנת שהפך לעורך דין והקים את Casetext ב-2013. בקיץ 2022 החברה קיבלה גישה מוקדמת ל-GPT-4, עצרה כמעט את כל הפעילות ובנתה את CoCounsel, עוזר AI לעורכי דין. החברה נמכרה ל-Thomson Reuters ב-650 מיליון דולר במזומן. ההרצאה בנויה משלושה חלקים, בחירת רעיון, בנייה ושיווק ומכירה, ואחריהם כ-9 דקות של שאלות מהקהל.

הטענה המרכזית: הבעיה של "מה אנשים רוצים" נעשתה פשוטה יותר, כי מספיק לבדוק על מה הם כבר משלמים לבני אדם. ההבדל בין דמו מרשים למוצר שעובד בפועל נקבע בשני דברים שכמעט אף אחד לא עושה כמו שצריך: פירוק מדויק של העבודה שמומחה אמיתי עושה, ו-evals שיטתיים עם עבודה עיקשת על ה-prompts. במכירה Heller מדגיש מוצר מצוין, תמחור לפי ערך, בניית אמון והטמעה אצל הלקוח. הוא גם מזכיר מ"הכנסות פיילוט" שלא יבשילו להכנסות אמיתיות.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, וגם מה לא נאמר.
- 2. **הרקע:** Casetext, המעבר ל-GPT-4 והאקזיט.
- 3. **איך בוחרים רעיון:** שלוש קטגוריות של רעיונות, והחישוב של שוק פוטנציאלי לפי משכורות.
- 4. **איך בונים:** פירוק עבודת המומחה לשלבים, prompts מול קוד, ו-workflow מול agent.
- 5. **Evals:** הדרך מ-60% דיוק ל-97%. זה החלק הטכני החשוב בהרצאה.
- 6. **שיווק ומכירה:** תמחור, אמון, פיילוטים והטמעה.
- 7. **שאלות מהקהל:** מתחרים, בחירת שוק, מיקוד בכל שלב, תמחור ויכולת להתגונן מפני מתחרים (defensibility).
- 8-9. **מה עושים עכשיו ומילון מונחים:** צעדים מעשיים, שאלות פתוחות והסבר למונחים באנגלית.

הערות מקור המסמך הופק אוטומטית מהתמליל ומהפריימים של ההקלטה. התמליל הוא תמלול אוטומטי (ASR), ולכן שמות ומונחים עלולים להופיע משובשים. למשל, שם החברה מופיע בתמליל כ-KSTX, K-Text ו-case text, ושם המוצר כ-Co-Council. כאן הם נכתבים Casetext ו-CoCounsel. הציטוטים מועתקים מילה במילה מהתמליל ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים. הכותרת והתיאור של הפוסט ב-X הם של מי שהעלה את הסרטון מחדש, ולא שימשו כמקור לעובדות.

1. תקציר מנהלים

Heller מציג מתכון בן שלושה חלקים ומדגיש שהוא נשמע מובן מאליו, אבל כמעט אף אחד לא מיישם אותו. הוא נשען בעיקר על הניסיון של Casetext ומביא מעט מספרים. הנה הטענות המרכזיות:

- **בוחרים רעיון לפי מה שכבר משלמים עליו לבני אדם:** תמיכת לקוחות, שמאי ביטוח, פרליגלים, עוזרים אישיים. כך כבר יודעים שיש ביקוש [3:01].
- **שלוש קטגוריות של רעיונות:** לסייע לבעל מקצוע, להחליף את העבודה לגמרי (למשל משרד עורכי דין שכולו AI), או לעשות דברים שעד עכשיו לא היו אפשריים, כמו קריאה ומיון של מאות מיליוני מסמכים [4:33].
- **השוק הפוטנציאלי (TAM) הוא סך המשכורות, לא מספר ה-seats:** אלפי דולרים בחודש לבעל מקצוע לעומת 20 דולר בחודש למנוי SaaS. Heller קורא לזה פער של פי אלף בערך [6:04].
- **בונים לפי מה שהמומחה הטוב ביותר היה עושה:** מפרקים את המשימה לשלבים, כל שלב הופך ל-prompt או לקוד, ואם התהליך דטרמיניסטי כותבים workflow ב-Python פשוט, בלי [13:49], [9:08]. framework.
- **החלק הקשה הוא לא לבנות אלא לדייק:** דמו שמגיע ל-60%-70% דיוק מספיק לגיוס כסף ולפיילוט, אבל לא עובד בפרודקשן [15:20].
- **Eval ושבועיים של עבודה על prompt אחד:** מתחילים ב-60%, ועם evals שהולכים ומתרבים מגיעים ל-97%. לפני בטא כדאי להגיע ל-100 בדיקות לכל prompt ו-100 בדיקות למשימה כולה [19:53].
- **רוב ה-evals הגיעו מלקוחות אמיתיים:** כל תלונה של לקוח הופכת לבדיקה חדשה. בהמשך היו להם יותר בדיקות מלקוחות מאשר כאלה שנכתבו במעבדה [21:25].
- **מוצר מעולה חשוב יותר ממכירות:** אחרי שהוציאו את המוצר מבוסס ה-LLM, אנשי המכירות שלהם הפכו ל"order takers" [24:26].
- **סכנת ה-PRR:** הרבה מה-ARR שסטארטאפים מדווחים עליו הוא בעצם תשלום על פיילוטים שלא יומר להכנסה קבועה, ו-Heller צופה שזה יגמר ב-"mass extinction event" [27:28].

מה לא נאמר

- **אין מדדי דיוק של CoCounsel עצמו:** הנתונים של 60%, 97% ו-99/100 מוצגים כהמלצה כללית, לא כתוצאות שנמדדו במוצר.
- **אין מספרים עסקיים אחרי ההשקה:** Heller מזכיר הכנסות של כ-20 מיליון דולר ו-100 עובדים בזמן המעבר ל-GPT-4 [1:30], אבל לא אומר מה קרה להכנסות, לשימוש או לעלויות ה-tokens אחר כך.
- **הפער של פי אלף לא מבוסס:** בתמליל מופיע גם "[6:04] \$10,000, \$100,000 or even \$1,000, factor of", משפט משובש בתמלול או בדיבור. זה לא חישוב.
- **אין פרטי ארכיטקטורה:** אין שמות של מודלים בפרודקשן מלבד GPT-4, אין פרטים על retrieval, על שילוב מערכות או על טיפול בהזיות מעבר ל-evals.
- **הטענות על פיילוטים ועל defensibility נשענות על אנקדוטות:** הן מבוססות על ניסיונו כמשקיע אנג'ל ולא על נתונים.

2. הרקע: מ-Casetext ל-CoCounsel

Heller גדל כמתכנת, עבר להיות עורך דין (לימודי משפטים, התמחות אצל שופט, משרד גדול) ועזב אחרי זמן קצר כי לא האמין לשיטות העבודה בתחום. הוא הקים את Casetext ב-2013 ומדגיש שלוקח זמן עד שחברה כזו מצליחה. כבר בתקופת BERT, לפני שקראו לזה AI, החברה עבדה על NLP לשיפור החיפוש המשפטי. אחד מחוקרי ה-AI שלהם, ששמו מופיע בתמליל כ-Javan (ייתכן שהשם משובש), זיהה את הפוטנציאל מוקדם.

בזכות המחקר הזה הם קיבלו גישה מוקדמת ל-GPT-4. כשהיו בהיקף של כ-20 מיליון דולר הכנסות וכ-100 עובדים, הם החליטו לעצור הכול ולבנות את CoCounsel.

— **Jake Heller [1:30]** We were like \$20 million in revenue. We were doing great. I had like 100 people, and we stopped everything that we were doing and said, we're gonna build something totally new based on this new technology

לפי Heller, ההחלטה הזו הובילה לרכישה על ידי Thomson Reuters ב-650 מיליון דולר במזומן, בערך שנתיים לפני ההרצאה. בהמשך, בשאלות מהקהל, הוא מודה בטעות אסטרטגית: לפני עידן ה-LLM שוק התוכנה המשפטית היה קטן. עורכי דין מרוויחים בערך טריליון דולר בשנה, אבל מוציאים מעט מאוד על תוכנה [35:02].

3. איך לבחור רעיון: לחפש עבודה שכבר משלמים עליה

Heller מתחיל מהמוטו של "YC, 'make something people want'", וטוען שבעידן ה-AI קל הרבה יותר לדעת מה אנשים רוצים: מסתכלים על מה שהם כבר משלמים לבני אדם לעשות. LLMs יכולים לבצע חלק גדול מהעבודה הזו, ובעולם הפיזי רובטיקה יכולה להשלים את השאר. משם הוא מגדיר שלוש קטגוריות: לסייע (זה מה ש-CoCounsel עושה: קריאת מסמכים, מחקר, סקירת חוזים ו-redlines), להחליף (משרד עורכי דין, רואה חשבון או פיזיותרפיסט שמופעלים על ידי AI), ולעשות את מה שעד עכשיו לא היה אפשרי (להריץ אלפי מופעים של Gemini 2.0 Flash על כל המסמכים של משרד עורכי דין).

— **Jake Heller [6:04]** But today, the actual amount of money that we already know people and companies are willing to spend is the combined salaries of all the people they're currently paying to do the job.

הוא מתייחס בעצמו לטענה שזה נשמע דיסטופי, ומציע שתי תשובות. הראשונה היא דוגמת מדליקי הפנסים, שאותה הוא מייחס למאמר של Sam Altman: תפקידים נעלמים ומשאירים מקום לעתיד שקשה לדמיין היום. השנייה היא הנגשה של שירותים יקרים. לדבריו, "over 85% of people who are low income don't get access to legal services" [7:37], ועורכי דין מהירים פי 100 וזולים פי 10 יוכלו לקבל לקוחות שהיום הם דוחים. באותו הקשר הוא מזכיר את Cursor ומוצר שמופיע בתמליל כ-"Rinsurf", כנראה Windsurf.

4. איך לבנות: לפרק את עבודת המומחה לשלבים

Heller מדגיש את המילה reliable ומבחין בין דמו מגניב למשהו שעובד בפועל. השלב הראשון הוא לדעת בדיוק מה בעל המקצוע עושה. ב-Casetext ל-30%-40% מהעובדים היה רקע משפטי, גם בין המתכנתים. מי שאין לו ידע כזה יכול "to be an undercover agent" בחברה מהתחום, או למצוא שותף מייסד עם מומחיות. אחר כך שואלים איך המומחה הטוב ביותר היה מבצע את המשימה אם היו לו זמן ומשאבים בלי הגבלה, ובונים את התהליך אחורה מהתשובה.

הדוגמה שלו היא גרסה של deep research שהם בנו כבר עם GPT-4, כשנתיים וחצי לפני ההרצאה: שאלות הבהרה, תוכנית מחקר, עשרות חיפושים, קריאה של מאות תוצאות, סינון מה שלא רלוונטי, רשימות שמסבירות למה כל מקור רלוונטי, כתיבת חיבור, ולבסוף בדיקה שכל הפניה נכונה. כל שלב הופך ל-prompt. למשל: לדרג רלוונטיות של פסק דין בסולם 0-7, לכתוב את החיבור מתוך הרשימות, או לבדוק אם הערת שוליים תואמת את המקור.

— **Jake Heller [13:49]** Prompts are slow and expensive. Tokens are still expensive. So when you're breaking down these steps, some of these things might just be good old software engineering, right? Do that when you can

אחרי הפירוק צריך להחליט: אם המומחה תמיד מבצע את אותם חמישה-שבעה שלבים, בונים workflow פשוט. לדבריו, כך נבנה רוב CoCounsel, בלי LangChain ועם "Just Python code" שבו הפלט של פונקציה אחת עובר לפונקציה הבאה [13:49]. אם הגישה של המומחה תלויה בנסיבות, למשל תוכנית מחקר או מקורות שמשתנים, עוברים לפתרון agentic, שקשה יותר לוודא את איכותו. בשני המקרים הוא חוזר על אותו כלל: "don't fly blind".

5. Evals: איך מגיעים מ-60% ל-97%



צילום רחב של הבמה ב-AI Startup School של Y Combinator: מול שקופית Evaluations שבה שאלות כמו "מה נחשב תוצאה מצוינת במשימה כולה ובכל תת-משימה". הטקסט בשקופית קטן מדי לקריאה מלאה

— **Jake Heller [15:20]** The hard part, frankly, isn't building it. The hard part's getting it right

לפי Heller, רוב הצוותים בונים דמו עם 60%–70% דיוק, מגייסים איתו כסף ואפילו מחתימים לקוחות לפיילוט, ואז המוצר נכשל בפרודקשן. הפתרון מתחיל גם כאן במומחיות בתחום: להגדיר איך נראית תוצאה טובה, גם במשימה כולה וגם בכל תת-משימה (למשל, מה מבדיל שאלתת חיפוש טובה מגרועה). ההמלצה שלו היא לנסח את הפלט כך שאפשר יהיה לבדוק אותו באופן אובייקטיבי: true/false או מספר בין 0 ל-7. כך הבדיקה פשוטה: "the answer should be six" [18:23]

לבדיקות הוא משתמש ב-promptfoo, framework בקוד פתוח שרץ משורת הפקודה, אבל אומר שהבחירה בכלי לא קריטית. הסדר שהוא מציע: להתחיל בתריסר בדיקות שדומות למה שלקוחות באמת ישלחו, להגיע לציון מושלם, להרחיב ל-50 ואז ל-100, ולשמור holdout set שלא מסתכלים עליו בזמן העבודה על ה-prompt, כדי לא להתאים את ה-prompt רק ל-evals. לטענתו, אפשר להגיע רחוק עם prompting בלבד בלי fine-tuning, כי הטעויות של המודל חוזרות על עצמן בדפוסים, ואפשר לטפל בהן בהוראות מפורשות ובדוגמאות.

— **Jake Heller [19:53]** you're going to get to something that passes like 97% of the time, and the 3% is kind of explainable. It's like a human would, it's like a judgment call almost

- **הסף לפני בטא:** 100 בדיקות לכל prompt ו-100 בדיקות למשימה כולה. מי שעובר 99 מתוך 100 יכול להרגיש בטוח. 1,000 בדיקות עדיפות, אבל קשה לכתוב אותן [19:53].
- **בטא עם ציפיות ברורות:** אומרים ללקוח שהמוצר עדיין לא מושלם, וכל תלונה הופכת לבדיקה חדשה. "We've added much more evals at this point from real things that happen to real customers than the ones we came up with in the lab." [21:25]
- **לקוחות כותבים prompts מבולגנים:** שאלות אמיתיות כמעט לא קריאות, ועדיין צריך להחזיר עליהן תשובה טובה [21:25].
- **העבודה לא נגמרת:** כשיוצא מודל חדש מריצים אותו מול אותן בדיקות. שינוי של מילה אחת יכול להוסיף אחוז דיוק, ובתחומים כמו פיננסים, רפואה ומשפט זה משמעותי. "There should be a new GitHub pull request like every other day or every day on your prompts." [22:55]

השורה התחתונה של Heller לפי Heller, מי שעושה רק את שני הדברים האלה, פירוק מדויק של העבודה ו-evals לכל שלב ולתהליך כולו, יגיע לכ-90% מהדרך ויבנה מוצר טוב יותר מרוב מה שקיים בשוק. הוא מזכיר שהבסיס לכך הוא ששלב ה-evals דורש נכונות לעבוד שבועיים על prompt אחד [19:53], [22:55].

6. שיווק ומכירה: מוצר מעולה קודם לכל

Heller קורא לחלק הזה "the hardest part" ואומר שגם אחרי האקזיט, בתוך חברה ששווה מיליארדים, הם עדיין לא פיצחו אותו. הוא פותח בויכוח עם חברי הדיקטטורין מטעם קרנות ההון סיכון, שטוענים שמכירות ושיווק חשובים יותר מהמוצר. לדבריו, במשך עשר שנים היה להם מוצר סביר ומנהלי מכירות ושיווק מנוסים, והתוצאות היו סבירות בלבד. כשהגיע מוצר מעולה, הלקוחות הגיעו מהמלצות והתקשורת פנתה אליהם בעצמה. אנשי המכירות שעברו מהמוצר הישן הפכו ל-"order takers" [24:26].

- **אולי אתם לא מוכרים תוכנה:** חברות שמספקות שירות מלא, למשל סקירת חוזה ב-500 דולר במקום 1,000 דולר במשרד עורכי דין. "per month versus \$500 per contract \$20". מתמחרים לפי הערך [25:57].
- **שואלים את הלקוחות איך הם רוצים לשלם:** ב-Casetext שקלו תמחור לפי שימוש, אבל הלקוחות העדיפו תשלום קבוע וצפוי: 6,000 דולר לשנה ל-seat, כלומר 500 דולר בחודש. "I'd rather pay more, but make it consistent" [25:57] "throughout the year, than potentially pay less and pay per use."
- **לגשר על פער האמון:** בארגונים יודעים לנהל אנשים, אבל אין להם ניסיון עם המוצר שלכם. הפתרון הוא השוואה ישירה: הלקוח ממשיך לעבוד עם משרד עורכי הדין או רואה החשבון שלו במקביל למוצר, ומשווים מהירות ואיכות. וגם מחקרים ופיילוטים [27:28].
- **המכירה לא מסתיימת בחתימה:** הדרכה, הטמעה מסודרת ו-Heller onboarding. מזכיר הערה של דובר קודם באירוע, שבתמליל מופיע רק בשמו הפרטי Satya, על הגידול בתפקיד forward-deployed engineers, ומתאר אותם כ-"boots-on-the-ground people" [28:58].

Jake Heller [27:28] It's like instead of ARR, it's like PRR, like pilot recurring revenue or something — that are not even recurring, just pilot revenue, I guess

הוא מסכם את החלק במשפט שחזר עליו הרבה בחברה: "your product isn't just the pixels on the screen" [28:58]. כלומר, גם התמיכה, ה-customer success, הקשר עם המייסד והדרכה הם חלק מהמוצר. מי שמשקיע בהם יותר יכול לנצח גם מתחרה עם ממשק טוב יותר.

7. שאלות מהקהל

שאלה	התשובה של Heller
כדאי להיכנס לתחום שכבר יש בו מתחרים? [30:28]	לא להתעניין במתחרים בכלל. השווקים עצומים ואף חברה לא תשלט בכולם, ואחרי שמתחילים לבנות מגלים "how bad they are". כדאי לחפש תפקידים שכבר עושים להם outsourcing לחו"ל, ולהימנע מעבודה שארגונים רואים בה חלק מהזהות שלהם (דוגמת הסיפור בסרטים של Pixar).
על מה כדאי להתמקד בכל שלב, מ-pre-seed ועד האקזיט? [32:00]	בכל שלב: מוצר מצוין שמגיע ל-product-market fit. לדבריו, HR, תרבות ארגונית, גיוס כסף ומכירות צריכים לשרת את המטרה הזו ולא להפוך למטרה בפני עצמה. "I fell into this trap. Big mistake." [33:30]
מה היה עושה היום, אחרי האקזיט? [35:02]	מתמקד בבעיה הגדולה ביותר שאפשר לפתור עם הטכנולוגיה והכישורים שיש לו. הדוגמה שלו: רובוט ניקיון ביתי, בדומה למדיח הכלים בשנות ה-50. השואל סיפר שמכר סטארטאפ ל-Deloitte בגיל 14, ו-Heller ציין שהוא עצמו עשה אקזיט בגיל 40.
איך מתמחרים משהו שבני אדם לא יכולו לעשות? [36:35]	בהתחלה גובים כמו שבן אדם היה גובה, ועם הזמן התחרות תוריד את המחיר, "Bad for your business, good for society." [38:06]. בקטגוריה חדשה מתחילים מהערך ללקוח וגובים 10%-20 ממנו, ומבררים את המחיר בשיחה עם הלקוח.
איך נמנעים מלהיות GPT wrapper? [38:06]	"Just build it". כשבונים, מגלים כמה רכיבים, אינטגרציות לנתונים, בדיקות וכיוון של prompts צריך. אחרי שנתיים של עבודה כזו יש לכם משהו שאחרים לא יכולים לבנות בקלות.

8. מה לעשות עכשיו

- **למפות עבודה שמשלמים עליה:** לבחור תפקיד אחד שכבר עושים לו outsourcing, ולקבוע אם המוצר יסייע, יחליף או יאפשר משהו שעד עכשיו לא היה אפשרי.
- **לתעד איך מומחה עובד בפועל:** לשבת עם בעלי מקצוע או לעבוד בתוך התחום, ולכתוב את השלבים במפורש, כולל שלב בדיקה בסוף.
- **לסווג כל שלב:** קוד דטרמיניסטי, prompt, או החלטה של agent. להעדיף קוד כשאפשר, כי tokens יקרים ואיטיים.
- **לנסח פלט שקל לבדוק:** true/false או ציון מספרי, כדי שאפשר יהיה לכתוב לו evals.
- **להקים מערך promptfoo evals:** או כלי דומה. מתחילים ב-12 בדיקות, מגיעים ל-100 לכל prompt ו-100 למשימה כולה, ושומרים holdout set.
- **לחבר את הפידבק מהלקוחות:** כל תלונה בבטא נשמרת כבדיקה חדשה, וכל מודל חדש עובר את אותן בדיקות.
- **לבדוק את מבנה ההכנסות:** להפריד בין הכנסות מפיילוטים להכנסות קבועות, ולתכנן מראש איך פיילוט הופך לחוזה.

שאלה פתוחה	למה היא חשובה
מה היה הדיוק של CoCounsel בפרודקשן?	ההמלצות של 97% ו-99/100 לא נתמכות בנתונים מהמוצר עצמו.
מתי workflow כבר לא מספיק וצריך agent?	Heller מתאר את ההבחנה באופן כללי ולא נותן קריטריון ברור או דוגמה מ-CoCounsel.
האם prompting בלי fine-tuning מספיק בכל תחום?	הטענה מבוססת על הניסיון בתחום המשפטי, שבו המודלים כבר חזקים בטקסט.
כמה עולים ה-tokens בהשוואה למחיר של 500 דולר בחודש ל-seat?	Heller אומר שה-tokens יקרים, אבל לא מתייחס לשיעור הרווח הגולמי.
איך מזהים מראש פיילוט שלא יבשיל?	ההרצאה מזהירה מפני PRR, אבל לא מציעה מדדים לזיהוי.

9. מילון מונחים

מונח	פירוש
TAM (total addressable market)	הסכום הכולל שאפשר להרוויח מהמוצר בשוק. לפי Heller, בעידן ה-AI הוא סך המשכורות על העבודה שהמוצר מחליף
Seat	רישיון למשתמש אחד. מודל התמחור הקלאסי של SaaS
SaaS	תוכנה כשירות, בדרך כלל במנוי חודשי
Evals / evaluations	סט בדיקות עם תשובות צפויות שמודד את הדיוק של prompt או של תהליך שלם
Holdout set	בדיקות שלא מסתכלים עליהן בזמן הכיוון, כדי לוודא שה-prompt לא הותאם רק ל-evals
promptfoo	framework בקוד פתוח שרץ משורת הפקודה ומשמש להרצת evals על prompts ומודלים
Workflow	רצף קבוע של שלבים (פונקציות ו-prompts) שבו הפלט של שלב אחד עובר לשלב הבא
Agentic	מערכת שבה המודל עצמו מחליט על השלבים לפי הנסיבות
Deep research	משימת מחקר רב-שלבית: תוכנית, חיפושים, סינון, רשימות, כתיבה ובדיקת הפניות
Fine-tuning	אימון נוסף של מודל על נתונים ייעודיים. Heller טוען שאפשר להגיע רחוק גם בלעדיו
Product-market fit	המצב שבו המוצר עונה על צורך אמיתי בשוק
ARR / PRR	הכנסה שנתית חוזרת. PRR הוא כינוי של Heller להכנסה מפיילוטים שלא תהפוך להכנסה קבועה
Forward-deployed engineers	מהנדסים שעובדים אצל הלקוח ומוודאים שהמוצר עובד אצלו בפועל
Redlines	סימון שינויים בטיטת חוזה שנשלחת לצד השני
GPT wrapper	מוצר שהוא שכבה דקה מעל מודל של צד שלישי, בלי יתרון שקשה להעתיק

