

# PRD-Engine System

## ארכיטקטורה מלאה

מדריך ויזואלי שמסביר איך PRD-Engine עובד מאחורי הקלעים

גרסה: | תאריך: פברואר 2026

1. איך הכל מתחבר.

2. כובעים, לא סוכנים נפרדים.

3. Sub-Agents (Task Tool)

4. DOC\_SOURCE

5. lessons.md

6. checkpoint.json

7. prd-index.json

8. Cross-Review

9. מבנה הקבצים.

10. Hooks

11. 18 כללי בrzl

12. Sweet Spot

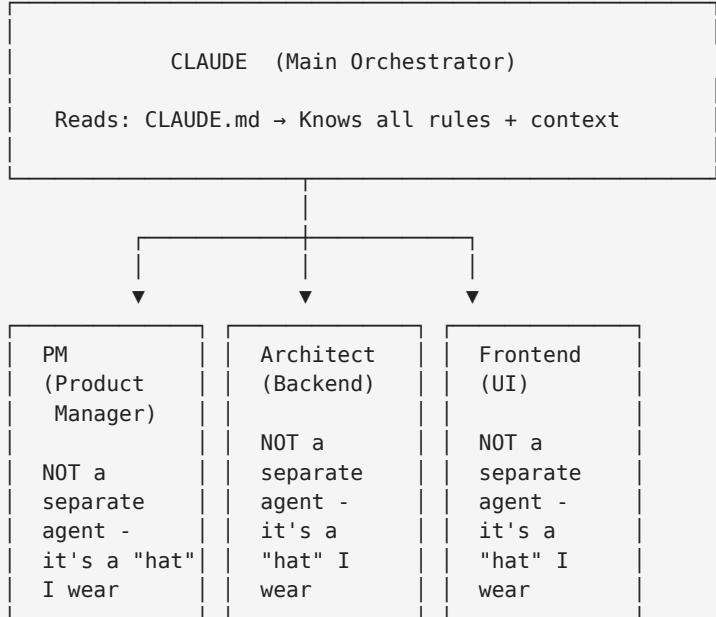
13. Epic מבנה

14. גמישות הוליסטית

15. אפס קצויות פתוחים

16. תרחיש מקצתה למקצתה

## אין הכל מתחבר



עסקיות. כשהוא — **Architect** שאלות טכניות. כשהוא — **Frontend** שאלות UX/UI. ו/אוabel תמיד אותו. הרעיון: **Claude** הוא מוח אחד שמחליף 'קובעים'. כשהוא — **PM** הוא שואל שאלות

## קובעים, לא סוכנים נפרדים

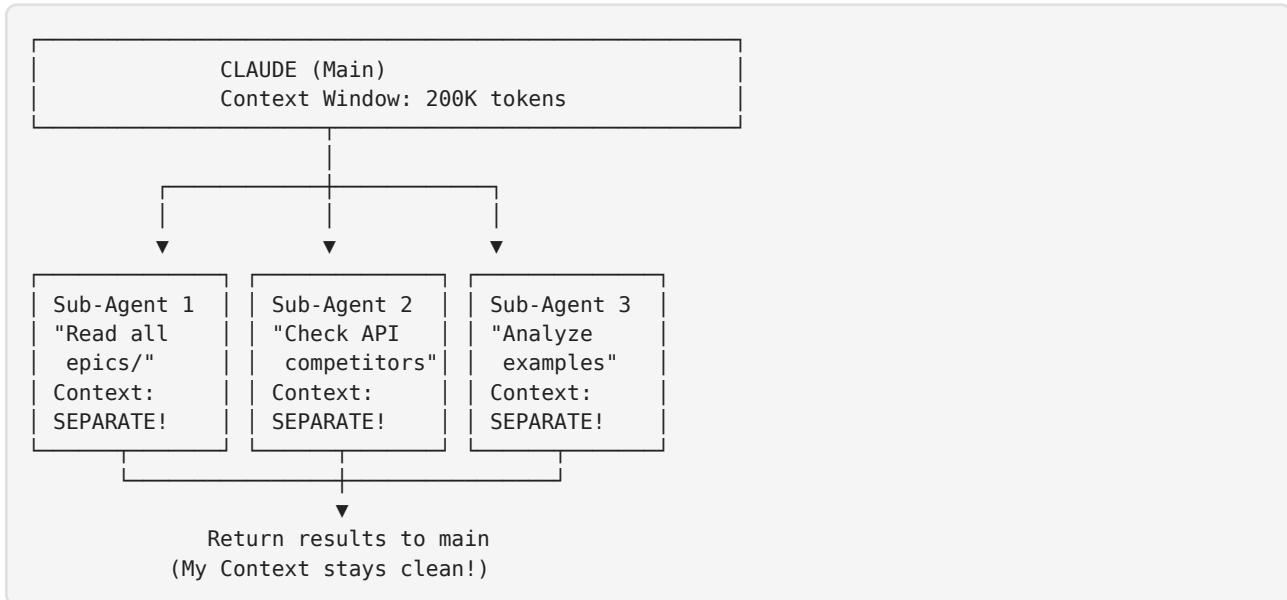
המבנה חשובה: כשהאני 'PM' אני לא מפעיל סוכן אחר — אני פשוט קורא את ה- **SKILL.md** של **PM** וממתנהג לפוי.

```
User: "Let's spec a Login system"  
|  
|  
Claude reads: product-manager/SKILL.md  
|  
|  
Asks 9 business questions (PM hat)  
|  
|  
"Switching to Architect!"  
Claude reads: architect/SKILL.md  
|  
|  
Asks 8 technical questions (Architect hat)  
|  
|  
"Switching to Frontend!"  
Claude reads: frontend/SKILL.md  
|  
|  
Asks 11+1 UI questions (Frontend hat)
```

אני זוכר מה PM שאל, ואני יכול לשלב את זה כש-Architect שואל. אין 'העברת מידע' בין סוכנים — הכל אצל למה זה חשוב? כי כל ה-context נשאר אצל.

## □ Sub-Agents (Task Tool)

Sub-Agent הוא סוכן נפרד שרך במקביל — משהו אחר לגמרי מ'קובעים'!



#	תרחיש	סוג	מה עושה
1	Session start / compact	Explore	קורא כל הקבצים + סיכום 60שורות
2	DOC_SOURCE	Explore	קורא מסמך גדול + סיכום דרישות
3	epics	Explore	בודק + nepics קישורים בין
4	Cross-Review	general-purpose	מנתח סתירות בין 3 Agents

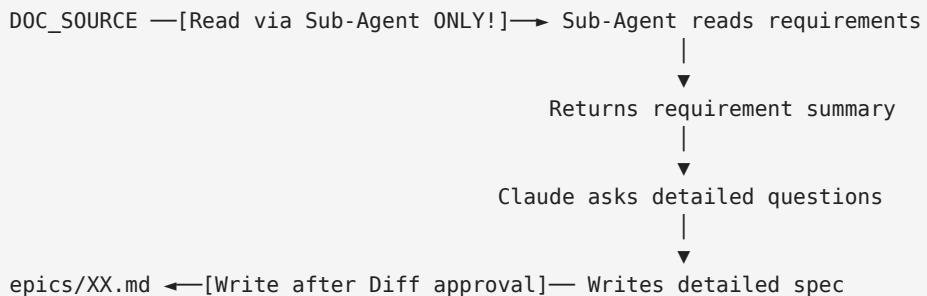
כל בrzil: כל סאב-אייגנט חייב — 'Opusmodel: 'sonnet Haiku אף פעם לא!

## □ — DOC\_SOURCE

בתחילת כל אפיון חדש, המשתמש מספק קישור למסמך הדרישות שלו (Google Doc, Notion וכו').

DOC\_SOURCE = READ-ONLY!  
- Source of truth for requirements  
- Never write to it  
- Never read in main context (too large!)  
- Always read via Sub-Agent

OUTPUT = epics/ files only!  
- .claude/memory/epics/XX-name.md  
- Written only after Diff approval



### 3 אופציות בהתחלה (AskUserQuestionTool):

1. יש לי קישור → תדיביך ונתחיל.
2. אין לי עדין → לך ליצור מסמך ותחזיר.
3. אפיון מאפס' → נשאל שאלות מפורחות בלי מסמך.

## □ lessons.md = Long-term Memory

```
.claude/memory/lessons.md

## Lesson 1: Question Format
- Mistake: Asked without options
- Fix: Always use AskUserQuestionTool with numbers

## Lesson 2: Google Doc
- Mistake: Wrote to Doc before approval
- Fix: Always show Diff and wait for approval

## Lesson 3: User preferences
- Hebrew in conversation, English for tech terms
- Tables with examples
```

### איך זה עובד:

1. אתה מatkן אותו → אני מזהה שאתה שיעור.
2. אני מוסיף ל-lessons.md.
3. lessons.md → סאב-אינטנסיבית קורא Session.
4. לא חוזר על אותן טיעות!

## ן – checkpoint.json

Claude Code עלול לעשות / compact כל רגע, או ה- Session עלול להיסגר. בלי שמירה — הכל אבוד! הבעה:

```
{  
  "timestamp": "2026-02-10T14:30:00",  
  "epic": "user-authentication",  
  "agent": "architect",  
  "question_number": 5,  
  "completed": ["Q1: Entities", "Q2: Relations",  
                "Q3: APIs", "Q4: Validations"],  
  "pending": "Q5: Error Codes",  
  "doc_source": "https://docs.google.com/...",  
  "notes": "User wants JWT, not sessions"  
}
```

### מטריצת שמירה — מתי שומרים מה:

אזור	checkpoint	epic file	prd-index
תשובה משמעותית	ן	ן	ן
סיום שלב Agent	ן	ן	ן
Epic 100%	ן	ן	ן
50% Context	+ התראת	ן	ן
PreCompact	ן	—	—

## □ PRD — prd-index.json

Agent צריך להכיר את כל ה- PRD שכבר נכתב כדי לשאול שאלות חכמות — אבל בלי לשורף את כל ה-*Context!* הבעה: כל הפתרון: קובץ SON[קומפקטי (~ 500 tokens) ] שמכיל את 'המפה' של כל מה שקיים.

```
{  
    "epics_completed": 2,  
    "epics": {  
        "user-auth": {  
            "entities": ["User", "Role", "Session"],  
            "apis": ["/api/auth/login", "/api/users"],  
            "relations": ["User->Role (N:N)"]  
        },  
        "product-catalog": {  
            "entities": ["Product", "Category"],  
            "apis": ["/api/products"],  
            "relations": ["Product->Category (N:1)"]  
        },  
        "global_entities": ["User", "Role", "Product"],  
        "cross_epic_relations": ["Product->User (created_by)"]  
    }  
}
```

### איך Agent משתמש בהה:

- prd-index.json → 2 epics: user-auth, product-catalog'
- משתמש: 'בוא ונפין' מערכת הזמן'
- Architect שואל שאלה חכמה: 'ראיתי שיש Entity של Product ו-User. האם הזמנה מקוشرת אליהם?'
- זהו קישורים אוטומטי: 'ה- epic ישפיע על': user-auth, product-catalog

## □ Cross-Review — 7 Checks

לפני שכותבים לקובץ, בודקם 7 בדיקות חובה:

#	בדיקה	מה בודקים
1	PM Review	כל User Story מכוסה?
2	Architect Review	עקביות טכנית Entities — מטאימים ל-? APIs
3	Frontend Review	כל endpoint מומופיע ב-? UI
4	Analytics Events	מינימום 12 events מתחועדים
5	SEO Metadata	דומע לכל tags title, description, og:tags
6	i18n Consistency	כל הודעה בעברית + אנגלית
7	Deferred Docs	מה נדחה — מתחועדים ומונזחים

## מבנה הקבצים

```
.claude/
└── CLAUDE.md           <- Brain: 18 Iron Rules
    ├── settings.json      <- 3 Hooks (SessionStart, PreCompact, Stop)
    └── settings.local.json <- WebFetch permissions

    └── memory/            <- Persistent memory
        ├── checkpoint.json   <- ~200 tokens: where we stopped
        ├── prd-index.json     <- ~500 tokens: PRD map
        ├── lessons.md         <- Lessons learned
        ├── session-init.json  <- Sub-agent verification
        └── epics/
            ├── 01-user-auth.md <- Dev-ready spec files
            └── 02-product-catalog.md

    └── skills/prd-engine/   <- Engine
        ├── SKILL.md          <- Main Orchestrator
        ├── config/workflow.json <- Workflow config (v2.1.0)
        ├── agents/
            ├── product-manager/SKILL.md <- 9 business questions
            ├── architect/SKILL.md      <- 8 technical questions
            └── frontend/SKILL.md       <- 11+1 UI questions
        ├── rules/
            ├── 01-questions-format.md
            ├── 02-review-crosscheck.md
            └── 03-reflection.md
        ├── templates/
            ├── epic-template.md     <- Epic template (Parts A-D)
            ├── checkpoint.json
            └── prd-index.json
        └── hooks/
            ├── startup.sh          <- SessionStart
            ├── pre-compact.sh       <- PreCompact
            └── auto-checkpoint.sh   <- Stop (Reflection)
```

## אוטומציה — Hooks

3 שרכזים אוטומטיים — בלי שהמשתמש צריך לעשות ממשו:

Hook	מתי	מה עושה
SessionStart startup.sh	מת开机Session	מופיע "PRD-Engine v2.1.0"!
PreCompact pre-compact.sh	/compact	רמוש checkpoint + timestamp
Stop auto-checkpoint.sh	ונגמרSession	רמוש checkpoint + reflection + lessons

## 18 כללי ברזל – סקירה מהירה

#	כלל	בקצרה
0	סאב-אייגנטים	Sonnet + DOC_SOURCE via sub-agent
1	שאלות מובנות	AskUserQuestionTool + options
2	מודולריות	500 שורות מקסימום
3	שמירה רציפה	שמירה רציפה checkpoint
4	אפס קצוות פתוחים	כל פרט מוגדר
5	Plan Mode	חויה לפני שימושה ממשמעותית
6	Cross-Review	בדיקות חוות
7	Sweet Spot	MVP / Future / User Decides
8	Epics = Dev-Ready	כל = סונט אפיך לפיתוח
9	Diff לפניה כתיבה	הציג שינויים + אישור
10	גמישות הוליסטית	שאלות = SKILL נקיות פתיחה
11	lolאת שיפור	lessons.md
12	DOC_SOURCE קריית	בדרכם Session סאב-אייגנט
13	PRD Context	prd-index.json
14	החלפת כובעים	הודעה + קריית.md
15	Analytics	12+ events per epic
16	Design System	colors + typography + spacing
17	Reflection	בסוף session

## Sweet Spot = MVP vs Future

ב- Architect, כל שאלה טכנית מופרדת ל- 3 רמות:

רמה	משמעות	דוגמה
חויה ל-MVP	בלי זה ה- epic עוזב	Entity: User + email + password
המליצה לעתיד	יחסון refactor יחסוך refactoring מהמשך	הוסף שדה role_history
המשתמש מchlיט	יש 2 דרכים – תבחר	JWT or Session-based?

## מבנה 4 – אולקים █

epics/03-order-system.md

Part A - Business Requirements (PM)

- └─ User Stories (2-4)
- └─ Acceptance Criteria (8-12)
- └─ User Roles Table
- └─ Edge Cases / Funnel
- └─ KPIs Tables
- └─ 2030 Recommendations
- └─ Key Decisions

Part B - Technical Architecture (Architect)

- └─ Entities (fields, indexes, rules)
- └─ Relations (FK behavior)
- └─ API Endpoints (Auth + Rate Limit)
- └─ Validations (HE + EN)
- └─ Error Codes (7 categories)
- └─ Logging & Monitoring
- └─ Dependencies (3 categories)
- └─ 2030 Recommendations
- └─ Key Decisions

Part C - Frontend Specification

- └─ ASCII Wireframes
- └─ Error Display (Inline/Banner/Toast)
- └─ Responsive Breakpoints
- └─ Accessibility (WCAG AA)
- └─ i18n System
- └─ Design System
- └─ 2030 Recommendations
- └─ Key Decisions

Part D - Cross-Review (7 checks)

- └─ Analytics Events (12+)
- └─ SEO Metadata
- └─ i18n Consistency
- └─ Review Summary

Key Decisions (All Agents) - Unified Table

ל- → Cursor / Claude Code / Copilot / Windsurf / Bolt → המפתח מתחילה לפתח שירות, בלי שאלות נוספת.  
היתרונות: המשתמש לוחץ את הקובץ → מעביר

## גמישות הוליסטית – שאלות חכמת █

השאלות ב- md.SKILL – נקודת פיתיחה, לא רשימה סגורה!

SKILL.md = Minimum required + direction  
Agent Brain = Goes deeper as needed

Example:

PM asks (from SKILL.md): "Who is the user?"  
User answers: "Store manager"

PM continues (from intelligence):

- "Can a store manager manage multiple stores?"
- "Difference between internal and external manager?"
- "Does the manager see all employees or just their team?"

Rules:

- Ask mandatory questions from SKILL.md
- Add your own questions as needed
- Go deeper when there's ambiguity
- Don't ignore mandatory questions
- Don't ask irrelevant questions

## 4. אפס קצוזות פתוחים

כל פרט חייב להיות מוגדר — לא יהיה משהו:

לא מספיק	מספיק
"ויזג הودעת שגיאה"	"ויזג: אירעה שגיאה בשמירה. אנא נסה שוב."
"הכפטור יעשה" submit	"לחיצה: POST /api/x → toast / הודעה אדומה"
"validation" חובה, מייל 2 תווים. אימייל: פורמת. טלפון: 10 ספרות"	"שם: חובה, מייל 2 תווים. אימייל: פורמת. טלפון: 10 ספרות"
"המשתמש יוכל למחוק"	'popup' האם למחוק? → כפתור אדום → 'toast' "

## תרחיש מקצה לkerja — מלא Workflow

