Shomer.AI — ספר פרומפטים (Prompt Book)

מטרה: תיעוד כל שימוש משמעותי בסוכן AI בפרויקט — דרישת הקורס (חוק הזהב + ספר פרומפטים, שקופיות 4–5, 14). **Path A (סוכן ככלי פיתוח):** הרשומות כאן מתעדות את **Claude Code** כסוכן פיתוח לאורך הפרויקט — ניסוח, הרכבה, אימות, ארגון. זהו חלק ממנדט הסוכנים (ראו `preparatory_report.md` רכיב 7). **פורמט:** 7 שדות לכל רשומה — Decision · Evaluation · Output · Model · Prompt · Context · Goal.

> **הערה:** הפרומפטים מצוטטים בתמצית. הניסוח המלא נמצא בהיסטוריית השיחה ובהגדרות הסוכנים שהורצו.

רשומה 1 — הרכבת תוצרי מפגש 3 (במקביל, מבוססת-סוכנים)

שדה	תוכן
Goal (מטרה)	להכין את ארבעת תוצרי מפגש 3 — דו״ח מכין, שאלת מחקר, מאמרי דגל, תכנית עסקית — בחלון של 4 ימים.
Context (הקשר)	מקורות קיימים: `research_questions.he.md`, `plan-he/`, `related_work.he.md`, Proposal, ה-decision log. דרישת ״הרכבה והתאמה״, לא יצירה מאפס.
Prompt (פרומפט)	*״בוא נמשיך לבצע את ההכנות לקראת מחר 28-05 ... להכין דו״ח מכין מפורט, למצוא שאלת מחקר ... להכין תכנית עסקית ולהכין מאמרי דגל ... במקביל על ידי skills מתאימים לכל משימה.״*
Model (מודל)	orchestrator — Claude Opus 4.7 (1M context); שני (`general-purpose`) sub-agents במקביל ברקע.
Output (פלט)	חלוקה: סוכן A → תכנית עסקית LaTeX; סוכן B → אימות ציטוטים; ה-thread הראשי → דו״ח מכין + ספר פרומפטים.
Evaluation (הערכה)	זוהתה מראש מתיחות מסגור (Proposal מול docs/) ואומתה מול ה-decision log לפני כתיבה — נמנע תוצר סותר.
Decision (החלטה)	✅ התקבל. לעקוב אחרי המסגור המתועד (RQ3 ראשית + ארכיטקטורה פתוחה למפגש 4), לא לפתוח החלטות נעולות.

רשומה 2 — אימות ציטוטים של מאמרי הדגל

שדה	תוכן
Goal (מטרה)	לאמת ולתקן את הציטוטים של מאמרי הדגל לפני ההגשה; לפתור את ה- ⚠️ על ציטוט SinaLab.
Context (הקשר)	`literature_flagship.md` סימן את ציטוט SinaLab כטעון-אימות; "Jarrar et al." היה ניחוש. המסגור (שני עוגנים) נעול ואסור לשנותו.
Prompt (פרומפט)	*"Verify and format citations ... resolve the SinaLab ⚠️ ... never fabricate a citation, arXiv id, or DOI ... do NOT change the framing."*
Model (מודל)	Claude (`general-purpose` sub-agent) + WebSearch/WebFetch (arXiv, IEEE Xplore, ACL Anthology, GitHub).
Output (פלט)	ציטוט SinaLab מתוקן: **Hamad, N., Jarrar, M., Khalilia, M., & Nashif, N. (2023). "Offensive Hebrew Corpus and Detection using BERT." AICCSA. arXiv:2309.02724.** + תיקון DictaLM (Shmidman et al., 2024, arXiv: 2407.07080) + יצירת 2.0 `references.bib`.
Evaluation (הערכה)	מאומת מול arXiv/IEEE/ACL. ה-DOI של SinaLab נגזר מ-doc 10479258 וסומן לאימות-סופי (IEEE החזיר 401). שום ציטוט לא הומצא.
Decision (החלטה)	✅ התקבל ועודכן in-place. נותר: אימות סופי של ה-DOI לפני ההגשה הסופית.

רשומה 3 — כתיבת תכנית עסקית ב-LaTeX

שדה	תוכן
Goal (מטרה)	תכנית עסקית 6 סעיפים ב-LaTeX (עברית RTL), מוכנה לקימפול ב-Overleaf (XeLaTeX).
Context (הקשר)	מקורות: `plan-he/03` + Proposal §2.3/§3/§4/§9.1/§11.3. אין מנוע LaTeX מקומי → חייב להיות Overleaf-ready. דרישה: לסמן מספרים לא-ממוקרים `[למקור]`, לא להמציא.
Prompt (פרומפט)	*"Author a 6-section business plan in LaTeX ... XeLaTeX + polyglossia + bidi ... token-economics formula + worked example ... mark unsourced numbers [למקור]. No invented citations."*
Model (מודל)	Claude (`general-purpose` sub-agent) + WebSearch (גודל שוק, דמוגרפיה CBS, תמחור מתחרים, תמחור טוקנים).
Output (פלט)	`docs/business_plan/business_plan.tex` + `README.md` עם הוראות קימפול ורשימת מספרים טעוני-מקור. *(ראו דוח הסוכן בסיום הריצה.)*
Evaluation (הערכה)	אומתו בוויב (footnotes): גודל שוק ~2.25M; ~9.8–12.25% CAGR/$1.74B←$1.55B משפחות בישראל (CBS); תמחור Canopy/Qustodio/Bark; מחירי טוקנים Gemini Flash/GPT-4o-mini/Haiku Claude. נשארו `[למקור]`: קוהורט SAM מדויק (6–16 עם סמארטפון), SOM (40₪ / 5,000 — יעד פנימי), מחירי Bosco/Keepers, והנחות נפח ה-token-economics. שום ציטוט לא הומצא.
Decision (החלטה)	טיוטה מובנית להצגה במפגש 3; מספרי token-economics ממוקרים-במלואם נדחים לאחרי המפגש (ראו `next-meeting-prep`).

רשומה 4 — הרכבת חבילת התכן המלאה (מפגש 4, 10 LLDs במקביל)

שדה	תוכן
Goal (מטרה)	לייצר חבילת תכן מלאה לפני מימוש — Low-Level Design לכל מודול + גיבוי משימות JSON — מוכנה לאישור מפגש 4.
Context (הקשר)	PRD v1.0 + ארכיטקטורה B נעולים. נדרש תכן מפורט (תבנית 11 סעיפים) לכל מודול, עם דגש / OOP ports-and-adapters ו-swappability מלאה (החלפת Context-Agent / DictaBERT / OCR בלי לגעת בשרת).
Prompt (פרומפט)	*"Use product manager + project organizer agents to analyze the PRD and produce one Low-Level Design per module ... JSON tasks per module; everything under a* `design/` *folder." + "Every component must be swappable ... without changing the server. Strong OOP"*
Model (מודל)	**3** + Claude Opus (orchestrator) **סוכני** `product-strategist` **במקביל** (/ AI-core / client-edge orchestration).
Output (פלט)	9 קבצי `design.md` (~5,470 שורות) + `docs/design/README.md` (עקרונות ports-and-adapters) + §Interface boundary" 2.5" בכל `tasks.json` 9 + `tasks_index.json` + LLD (124 משימות, מורחב מאוחר יותר ל-144).
Evaluation (הערכה)	זוהו מראש פערים לתיקון (`non_offensive` ↔ `none-offensive` , חוסר , `train_dictabert.py` refactor של `OcrBackend`); כל ה-JSON אומת לפרסור ול-cross-references.
Decision (החלטה)	✅ דפוס composition-root נעול: רק `lifespan` `server/app/main.py` `()` בונה adapters קונקרטיים. ה-README הוא מקור-האמת למי-מייבא-את-מי.

רשומה 5 — סקירת ארכיטקטורה + פתרון 3 חוסמים (מפגש 4)

שדה	תוכן
Goal (מטרה)	לוודא שה-LLDs מכסים את כל הפונקציונליות הנדרשת, ולזהות שאלות פתוחות לפני תחילת מימוש.
Context (הקשר)	9 LLDs קיימים ; נדרש coverage matrix מול §1–§15 PRD ובדיקת concerns חוצי-מערכת (trace_id, A/B switch, privacy, retention, label consistency, confidence direction).
Prompt (פרומפט)	*"Review the architecture design and charts and verify the LLD architecture contains all necessary functionality. Are there open questions to resolve before tasks execution?"* → *"Yes [fix the 3 blockers]".*
Model (מודל)	סוכן `architecture-reviewer` (סקירה) + סוכן `product-strategist` (LLD למודול audit_log).
Output (פלט)	`docs/design/review.md` (486 שורות), · **7 Important** · **3 Blockers** · **Conditional Ready** verdict. 3 **Nice** 5 החוסמים נפתרו : LLD `audit_log/` — **G-01** חדש (758 שורות + 14 משימות, סכמת SQLite 5 טבלאות) ; **G-02** — תווית נעולה **G-03** ; `non_offensive` — תיקון polarity ב- `decide_inner_` + מטריצת בדיקה 6 שורות. 144 → Backlog משימות.
Evaluation (הערכה)	כל חוסם סומן ✅ RESOLVED ב- `review.md` כלוח-מצב חי ; באג G-03 (`is_offensive=False, confidence=0.92` היה מנותב בטעות ל-ALERT_DIRECT) נתפס ונמנע.
Decision (החלטה)	✅ אושר ; 7 בעיות (G-04…G-14) Important נדחו במכוון לשבוע 1 של מימוש.

רשומה 6 — בניית צינור השרת המודולרי (מפגש 5, 3 agents במקביל)

שדה	תוכן
Goal (מטרה)	לבנות את מודולי השרת — classifier (DictaBERT), OCR, Context Agent — מאחורי ה-Protocols, חי end-to-end.
Context (הקשר)	ממשקי Protocol מוגדרים; שכבת ה-foundation (4 + `schemas.py` קבצי `protocol.py`) הוכנה ידנית מראש כדי לאפשר עבודה מקבילית בלי התנגשויות.
Prompt (פרומפט)	*"Can we start building the server modules: classifier, OCR, Context Agent — using backend-developer + ml-developer in parallel ... now that we have the Protocol interfaces."*
Model (מודל)	`ai-researcher-developer` (classifier) + 2× `backend-developer` (ocr, context_agent) **במקביל**.
Output (פלט)	classifier (~2,150 שורות, 57 tests; Ollama+HuggingFace+Stub adapters) + ocr (~1,400, 45 tests; Tesseract extraction-only) + context_agent (~2,350, 70 tests; Mock+OpenAI+Anthropic clients + LlmRouter + SqliteTokenManager) + `InMemoryAuditStore` stop-gap + `main.py` כ-composition root. **172 tests passing**; smoke test `POST /classify` → `"שלום עולם"` `non_offensive`.
Evaluation (הערכה)	smoke test ירוק דרך Ollama אמיתי; כל החלפת adapter = env-var flip יחיד (`CLASSIFIER_MODEL_VERSION`).
Decision (החלטה)	✅ ה-composition-root + ייבוא Protocol-only בין מודולים הם כעת אמיתיים בקוד, לא רק במסמכים.

רשומה 7 — נעילת ארכיטקטורת הרשת של DictaBERT (מפגש 5)

שדה	תוכן
Goal (מטרה)	לתכנן בקפידה, שכבה-אחר-שכבה, את ארכיטקטורת הרשת (DictaBERT + שכבות חדשות) לסיווג תוכן פוגעני — לפני אימון.
Context (הקשר)	ה-LLD של ה-classifier החזיק רק היפר-פרמטרים בסיסיים (lr, batch, epochs) — לא ארכיטקטורה מלאה ולא בחירת loss.
Prompt (פרומפט)	*"Plan the full NN architecture carefully — do we have it? Use ai-educator-architect + ai-researcher-developer." + "What ways do we have to change data for best results — accurate and fast?"*
Model (מודל)	סוכן `ai-educator-architect` .
Output (פלט)	`docs/concepts/dictabert_classifier_architecture.md` (903 שורות, 12 סעיפים): MLP head (`[CLS]→Dropout→Linear(768→256)→GELU→Dropout→Linear(256→5)`), Focal Loss γ=2 + `alpha=class_weights` , ε=0.05 label smoothing, AdamW+cosine, 5 epochs BF16, fallback chain, §9 data contract. (~500 שורות) עם §8 combination-safety. + `dictabert_data_techniques.md`
Evaluation (הערכה)	הושוו 5 חלופות ארכיטקטורה (Vanilla/Multi-task/Hierarchical/DAPT); ε=0.05 (ולא 0.1) הומלץ בגלל השילוב עם §8 Focal; תיעד 4 סיכוני stacking עם מיטיגציות.
Decision (החלטה)	הארכיטקטורה נעולה — המסמך עצמו הוא רשומת ההחלטה (עם נימוק, חלופות וציטוטים). הערה: param count אמיתי התברר כ-**184.3M** (לא ~110M) עקב אוצר-המילים העברי.

רשומה 8 — זרימת השרת המלאה (מפגש 6, 4 מודולים במקביל)

שדה	תוכן
Goal (מטרה)	לחבר כל מודול מתוכנן SqliteAuditStore / Gatekeeper / Alerts / Triage — end-to-end — בלי אימון DictaBERT ובלי שינוי קלט ה-classifier.
Context (הקשר)	הצינור הריץ in-memory audit→CA→inline-triage→classifier; חסרו Gatekeeper, Alerts (ALERT_DIRECT היה inert), Triage עצמאי, ו-SQLite persistence.
Prompt (פרומפט)	*"Build the full flow, then the SDK, create tasks, and implement with backend-developer + ai-researcher-developer agents."*
Model (מודל)	3× `ai-researcher-developer` + 1× `backend-developer` **במקביל**.
Output (פלט)	triage (45 tests) + alerts/ `LogNotifier` (105) + `gateway.py` (37) + `SqliteAuditStore` (23); `main.py` v0.6.0-fullflow (retention sweeper, LogNotifier dispatch, `register_gateway()`, conversation persistence); debug SDK (`dev_client replay` / `inspect_audit` / `load_test`) + 7 integration tests. **384 fast tests pass**; live demo 10/10, 6 sent + 2 rate_limited alerts persisted.
Evaluation (הערכה)	4 באגים התגלו בבדיקות ותוקנו: `record_agent_trace` לא נקרא, `trace_id` לא הועבר ל-CA, קונסולת Windows cp1252 הפילה שורת log עברית של `alerts.sent` (→ אכיפת UTF-8 stdio), crash על dict ב- `inspect_audit`.
Decision (החלטה)	`LogNotifier` D1 ברירת-מחדל (+ FCM כצעד הבא); D2 כלי-debug ב-Python (+ קודם `:sdk-` Kotlin `cli` אחריו); D3 `child_id` אופציונלי לזיכרון-שיחה אמיתי.

רשומה 9 — Context Agent על LLM אמיתי (Gemini) + baseline accuracy + test console (מפגש 6)

שדה	תוכן
Goal (מטרה)	להעלות את ה-Context Agent על LLM אמיתי, למדוד את ה-accuracy הבסיסי של המערכת, ולבנות קונסולת בדיקה ידנית ידידותית.
Context (הקשר)	ה-CA רץ עד כה על Mock דטרמיניסטי (כל הסלמה → "לא איום" → silent); לא היה מדד accuracy בסיסי ולא כלי בדיקה ידני נוח (הקלדת עברית מתהפכת ב-Windows terminal).
Prompt (פרומפט)	*"Build a friendly UI for manual full-flow testing ... add Gemini support for the Context Agent; and run the OCR/text eval data through the pipeline to measure baseline accuracy as a PDF (no data changes / no training)."*
Model (מודל)	**`gemini-2.5-flash`** — `GeminiClient` (primary, endpoint תואם-OpenAI) + **`haiku-4.5`** (fallback). הערכה דרך השרת החי.
Output (פלט)	`scripts/test_console.py` (sample-picker + `python-bidi` via RTL ממוספר, per-session + `server.log` + `audit.db`). **Baseline:** `scripts/eval_accuracy.py` + `GeminiClient`. **TEXT 69.5% / IMAGE 66.3%** 5-class, ~80% binary; `hate` recall ≈0.10. PDF: `docs/meeting6_accuracy_report.pdf`.
Evaluation (הערכה)	**ה-classifier הוא צוואר-הבקבוק, לא ה-OCR** → image ≈ text — מוטיבציה כמותית חזקה ל-fine-tune. תוקנו 3 באגי bring-up (model 2.0→2.5, פסי `` ```thinking, `json` שגדע JSON) + באג key-string + מפתחות `env.` שגויים.
Decision (החלטה)	`plan-docs/decisions/gemini-context-agent.decision.md` — עדיפות **Gemini > OpenAI > Anthropic** (מפתח ראשון primary, שני fallback, Mock משלים).

תבנית לרשומה חדשה (להעתיק)

שדה	תוכן
Goal	
Context	
Prompt	
Model	
Output	
Evaluation	
Decision	