קיבלתי. אתה מכוון גבוה מאוד, וזה המקום שבו אנחנו צריכים להיות. אתה מתאר מערכת שהיא למעשה **Knowledge Operating System (OS)** ולא סתם LMS. הבקשה שלך להוסיף שכבה סמנטית (בסגנון Stardog) היא ה"גיים-צ'יינג'ר" כאן. היא הופכת את המערכת מ"מחסן נתונים" ל"מוח" שמבין את הקשרים בין פיסת וידאו, שרבוט ידני, טקסט גמרא/מאמר, והערה של תלמיד.

להלן מסמך ה-PRD המורחב, המעמיק והטכני ביותר, המשלב את התובנות מ-NotebookML (הקונספט של "מקורות מחוברים") עם הארכיטקטורה הארגונית הפתוחה שביקשת.

# EduSphere Enterprise: מסמך אפיון מערכת (PRD) - v6.0

**הגדרה:** פלטפורמת ידע סמנטית, מבוזרת ומונחית-קהילה (Semantic Distributed Knowledge Fabric).

**דרישת יסוד:** 100% Open Source Stack, Air-Gapped Ready, Container-Native.

## 1. תפיסת העולם והחזון (The Paradigm Shift)

המודל הקיים של "קורסים" הוא ארכאי. ידע אינו ליניארי. החזון של EduSphere הוא ליצור **רשת ידע חיה**.

במקום "מורה משדר לתלמיד", אנחנו בונים **Data Fabric** שבו:

1. **הכל מקושר (Linked Data):** דקה 12:05 בוידאו קשורה סמנטית לפסקה 4 במאמר PDF, שקשורה לשרטוט שהוסיף תלמיד א', שקשור לקוד Python שהריץ חוקר ב'.
2. **דמוקרטיזציה של ה-AI:** המשתמשים (חוקרים, תלמידים, ראשי צוותים) לא מחכים למפתחים. יש להם כלים לבנות **Micro-Agents** משלהם על גבי המידע (למשל: "סוכן שמסכם לי כל מה שנאמר על 'היטלי בטון' בכל הקורסים של השנה האחרונה").
3. **ריבונות (Sovereignty):** הארגון הוא הבעלים של ה"מוח" (הגרף הסמנטי), והוא רץ אצלו בברזלים (On-Prem).

## 2. הליבה הסמנטית (The Semantic Core Layer) - "המוח"

זוהי הדרישה החדשה והקריטית ביותר. כדי לספק פונקציונליות דמוית-Stardog (וירטואליזציה של דאטה וגרף ידע) באמצעות כלי Open Source, נבנה את הארכיטקטורה הבאה:

### 2.1. ארכיטקטורת גרף הידע (Knowledge Graph Architecture)

במקום טבלאות SQL רגילות, המערכת תמפה את כל המידע ל-RDF Triples (נושא-נשוא-מושא).

* **הטכנולוגיה (Open Source Stack):**
  + **Graph Database:** שימוש ב-**JanusGraph** (על גבי ScyllaDB לביצועים) או **Apache Jena Fuseki**.
  + **Virtual Graph / Federation:** שימוש ב-**Ontop** (כלי Open Source שהופך בסיסי נתונים רלציוניים לגרף ידע וירטואלי בזמן אמת) כדי לא לשכפל מידע סתם.
* **האונטולוגיה (The Ontology):** נפתח אונטולוגיה ייעודית (Edu-Ontology) שתגדיר ישויות כגון: Lecture, Timestamp, Sketch, UserNote, Citation, Concept.
  + *דוגמה לקשר:* (User:Moshe) -> [ANNOTATED] -> (Video:Physics\_101 @ 14:30) -> [REFERENCES] -> (Concept:Thermodynamics).

### 2.2. מנוע ההסקה (Inference Engine)

המערכת לא רק תשלוף מידע, היא תסיק מסקנות.

* אם סרטון א' מסומן כעוסק ב"רמב"ם", וסרטון ב' מצטט את סרטון א', המערכת תסיק (Infer) שגם סרטון ב' רלוונטי למי שמחפש "רמב"ם", גם אם המילה לא נאמרה בו.

## 3. פלטפורמת יצירת האפליקציות (No-Code Agent Builder)

כפי שציינת, חסרה יכולת למשתמשים ליצור כלים בעצמם. כאן נכנס ה-**Agent Fabric**.

### 3.1. הסטודיו לסוכנים (Agent Studio)

ממשק Drag-and-Drop המאפשר למשתמש (ללא ידע בקוד) להגדיר סוכן AI אישי.

* **Data Scope:** המשתמש בוחר על איזה מידע הסוכן יעבוד (למשל: "רק הסיכומים שלי" או "כל מאגר המחקר של האוניברסיטה").
* **Logic Blocks:**
  + *Trigger:* "כשאני מסמן טקסט..."
  + *Action:* "תחפש סתירה לטקסט הזה במקורות אחרים בגרף הידע."
  + *Output:* "תציג לי את התוצאה בחלונית צפה."

### 3.2. שימושים לדוגמה (User-Generated Apps)

1. **"החברותא האוטומטית":** תלמיד ישיבה יוצר סוכן ש"מקשה קושיות". בכל פעם שהתלמיד מסכם סוגיה, הסוכן סורק את הגרף הסמנטי, מוצא פרשנות סותרת ומציג אותה.
2. **"Research Assistant":** חוקר מגדיר סוכן שעוקב אחרי כל הוידאו החדשים שעולים למערכת, ומסמן לו דקות ספציפיות שבהן מוזכרת מילה נרדפת ל"פולימרים".

## 4. חווית המשתמש האימרסיבית (The Deep-Dive Interface)

בהתבסס על NotebookML ועל הדרישות שלך, הממשק המרכזי אינו "נגן וידאו" אלא **"שולחן עבודה מחקרי" (Research Workbench)**.

### 4.1. ה-Canvas הרב-שכבתי

המסך מחולק לשלושה אזורים דינמיים:

1. **מדיה ראשית (Media Pane):** וידאו/אודיו. על גבי הוידאו יש שכבת Canvas שקופה (באמצעות **Fabric.js**) המאפשרת:
   * **שרטוט וקטורי:** השרטוט נשמר כאובייקט ב-JSON ומוצמד ל-Timestamp.
   * **Spatial Comments:** נעיצת הערה בנקודה ספציפית *בתוך* הפריים (למשל: סימון רכיב במנוע תוך כדי סרטון הדרכה).
2. **העמוד המקורות (Source Column):** כאן מופיע הטקסט המקורי (PDF/Transcript). הגלילה בטקסט מסונכרנת לוידאו ולהפך.
3. **הצ'אט הסמנטי (Contextual Copilot):** צ'אט שמודע לכל מה שמוצג כרגע על המסך.

### 4.2. יכולות יצירה משותפת (Co-Creation)

* **Forking:** תלמיד יכול לעשות "Fork" לקורס. הוא יוצר עותק פרטי של שכבת ההערות והשרטוטים, עורך אותם מחדש, ואז מציע "Merge Request" למרצה ("הוספתי הסבר גרפי בדקה 40, תרצה לשלב את זה בקורס הרשמי?").
* **Live Collaboration:** בדומה ל-Google Docs או Figma, מספר משתמשים יכולים לצפות באותו וידאו ולשרטט עליו יחד בזמן אמת (שימוש ב-**Yjs** מעל WebSockets).

## 5. ארכיטקטורה טכנית ותשתית (Infrastructure & Data Flow)

התשתית חייבת להיות מפלצת ביצועים (High Performance), עמידה (Resilient) ופתוחה.

### 5.1. קונטיינרים וניהול (Orchestration)

* **Kubernetes (K8s):** הסטנדרט לניהול. שימוש ב-Helm Charts לפריסה מהירה.
* **עבור Air-Gapped/Edge:** שימוש ב-**K3s** (גרסה קלה של K8s) שיכולה לרוץ גם על שרתים חלשים בתוך ניידת שידור או כיתה מנותקת.

### 5.2. צינור הנתונים (The Nervous System - Kafka)

כפי שציינת, **Apache Kafka** הוא קריטי כאן לניהול ה-State והזרימה.

* **Topics:**
  + video-ingest: העלאת חומר גלמי.
  + transcription-done: סיום תמלול (Whisper).
  + user-action: כל לחיצה, עצירה, שרטוט (לצרכי אנליטיקה וסנכרון).
  + semantic-update: טריגר לעדכון גרף הידע.
* **Stream Processing:** שימוש ב-**Apache Flink** או **Kafka Streams** לעיבוד זמן אמת (למשל: זיהוי שמילה מסוימת נאמרה הרגע ב-Live Stream והקפצת התראה למשתמש רלוונטי).

### 5.3. מסדי הנתונים (Polyglot Persistence - 100% Open Source)

1. **Metadata & User Data:** **PostgreSQL** (עם פלאגין **TimescaleDB** אם צריך Time-series לניתוח למידה).
2. **Semantic Graph:** **JanusGraph** / **Apache Jena**.
3. **Vector Search (AI Memory):** **Milvus** או **Qdrant** (קוד פתוח, ביצועים גבוהים).
4. **Raw Blob Storage:** **MinIO** (תחליף S3 מקומי).

### 5.4. מנגנון סנכרון Offline-First (CRDTs)

הבעיה הגדולה ביותר במערכות מבוזרות. הפתרון:

* כל הקליינטים (Web/Mobile) עובדים מול DB מקומי (**RxDB** או **PouchDB**).
* המידע נשמר כמבני נתונים מסוג **CRDT** (Conflict-free Replicated Data Types). זה מבטיח שאם שני חוקרים ערכו את אותה הערה כשהיו מנותקי רשת - כשהם יתחברו, המיזוג יהיה מתמטי ומושלם ללא איבוד מידע.

## 6. תפקידי ניהול (Admin & Governance)

כפי שביקשת, הפרדה ברורה בין רמות הניהול:

### 6.1. Super-Admin (Global/System Level)

* אחראי על ה-Infrastructure (בריאות הקלאסטרים של K8s).
* ניהול ה-Image Registry העולמי (עדכוני גרסאות).
* ניטור Kafka Lag וביצועי חומרה.
* לא רואה תוכן ספציפי של דיירים (Tenants) מטעמי אבטחה.

### 6.2. Local Org Admin (Tenant Level)

* מנהל את הארגון הספציפי (אוניברסיטה/יחידה צבאית).
* **ניהול האונטולוגיה:** יכול להגדיר שדות מידע ייחודיים לארגון בגרף הידע (למשל: הוספת שדה "סיווג ביטחוני" לישויות).
* **ניהול מודלים:** בחירה איזה LLM ירוץ מקומית (Llama 3, Mistral) ואיזה מודל Whisper יתמלל.
* ניהול משתמשים והרשאות גישה למידע רגיש.

## 7. מפת דרכים לפיתוח (Execution Roadmap)

### שלב 1: התשתית והגרף (חודשים 1-4)

* הקמת קלאסטר K8s ופריסת Kafka ו-MinIO.
* הקמת JanusGraph והגדרת האונטולוגיה הבסיסית (Video, Text, User).
* פיתוח Pipeline בסיסי: Video Upload -> Whisper -> Vector Embedding -> Graph.

### שלב 2: ה-Client המחקרי (חודשים 5-8)

* פיתוח ה-Frontend (React/Electron) עם נגן הוידאו ושכבות השרטוט (Canvas).
* אינטגרציה של מנגנון ה-CRDT לסנכרון Offline.
* השקת גרסת "Notebook" בסיסית.

### שלב 3: ה-Agent Fabric (חודשים 9-12)

* פיתוח ה-API הסמנטי (GraphQL מעל גרף הידע).
* יצירת ה-No-Code Builder לסוכנים אישיים.
* בדיקות עומסים (Load Testing) של מנגנון הסקת המסקנות.

### סיכום

המסמך הזה לוקח את EduSphere והופך אותה למערכת תשתיתית קריטית. השילוב של **גרף ידע סמנטי** (המוח), **Kafka** (מערכת העצבים), ו-**CRDTs** (יכולת השרידות), יוצר פלטפורמה שיכולה לשרת מצד אחד בחור ישיבה שצריך עומק טקסטואלי, ומצד שני מהנדס טילים שצריך סימולציות ושרטוטים – כולם באותה סביבה מאובטחת ומסונכרנת.