Perfect. Iată un rezumat al conversației noastre, structurat exact în stilul solicitat, pentru a putea fi folosit ca parte din documentația proiectului vostru.

---

## Documentația Interacțiunii cu Agentul Conversațional (Gemini)

Data: 22 octombrie 2025

Subiect: Definirea și rafinarea strategiei de implementare pentru proiectul "SmarTest".

**Context Inițial:**

Echipa avea o înțelegere generală a modulelor (`core_logic`, `engine`, `GUI`) și explora două direcții principale pentru generarea de întrebări: o abordare programatică și una bazată pe antrenarea unui model lingvistic propriu.

---

**Interacțiunea 1: Clarificarea Constrângerilor Proiectului**

Promptul Utilizatorului (Eu):

"Am primit o clarificare de la profesor care spune: 'componenta de generare de intrebari face parte din sistemul final [...] trebuie sa genereze real time intrebari [...] Asta inseamna ca nici pentru ea nu puteti folosi LLMs. [...] Dar in timpul rularii, nu veti putea folosi LLMs.'. Cum ne schimbă acest mesaj abordarea?"

Răspunsul Rezumat al Agentului Conversațional:

Agentul a confirmat că această regulă este una fundamentală. A explicat că:

1. Ideea de a antrena și a folosi un model lingvistic propriu în aplicația finală este **invalidată** și trebuie abandonată.
2. Abordarea pe care o discutasem anterior, numită **"Modelul 1: Generatorul Parametric Bazat pe Șabloane"** (care folosește șabloane de text și funcții Python pentru a genera date aleatorii în timp real), este de fapt **singura strategie corectă și permisă**.
3. Rolul agenților AI în acest proiect este strict cel de **asistent de dezvoltare** (pentru a genera cod, a oferi idei), nu de componentă software la runtime.

---

**Interacțiunea 2: Integrarea Materialelor de Curs**

Promptul Utilizatorului (Eu):

"Dacă vreau să folosesc informații din cursurile PDF, care arată ca niște prezentări, pentru a ajuta la generarea de întrebări, cum aș putea integra această parte?"

Răspunsul Rezumat al Agentului Conversațional:

Agentul a propus o strategie de pre-procesare, pe care a numit-o "Faza Offline", pentru a evita încălcarea regulilor. Planul sugerat a fost:

1. **Extragere Offline:** Să creăm un script Python care citește PDF-urile cursurilor o singură dată, în faza de dezvoltare.
2. **Crearea unei Baze de Cunoștințe:** Scriptul ar trebui să extragă informații cheie (concepte, definiții, exemple clasice) și să le structureze într-un fișier, de exemplu `knowledge_base.json`.
3. **Utilizare la Runtime:** Aplicația finală nu citește PDF-urile, ci doar acest fișier `knowledge_base.json` pentru a genera întrebări teoretice mai inteligente (ex: "Comparați algoritmii X și Y") sau pentru a folosi exemple clasice din curs.

---

**Interacțiunea 3: Crearea unui Plan Tactic și a unui Backlog**

Promptul Utilizatorului (Eu):

"Ținând cont de toate acestea, poți să adaugi noua parte de integrare a cursurilor în fluxul de lucru și să creezi un backlog detaliat pentru tot proiectul, pentru a-l putea împărți pe sarcini în echipă, mapat pe livrabilele de la L6, L8 și L12?"

Răspunsul Rezumat al Agentului Conversațional:

Agentul a generat un plan tactic complet și un backlog structurat pe "Epopei", care reflectă toate deciziile luate.

- **A combinat strategiile:** Planul final include atât extragerea de șabloane din PDF-urile de examen, cât și crearea bazei de cunoștințe din PDF-urile de curs, ambele în faza "Offline".
- **A structurat backlog-ul pe faze:**
    - **Faza 1 (până la L6):** Crearea unui prototip funcțional pentru un singur tip de întrebare (recomandat: Nash Equilibrium) pentru a valida întregul flux.
    - **Faza 2 (până la L8):** Adăugarea unui al doilea tip de întrebare (recomandat: MinMax) și implementarea `AnswerGenerator`.
    - **Faza 3 (până la L12):** Finalizarea tuturor celor 4 tipuri de întrebări, rafinarea GUI și adăugarea de funcționalități extra.
- **A sugerat alocarea sarcinilor:** Pentru fiecare fază, a propus o modalitate de a împărți modulele de lucru (ex: o persoană pe `core_logic`, una pe `engine`, etc.) pentru a facilita lucrul în paralel.

**Concluzie:** La finalul discuției, am obținut un plan de acțiune clar, detaliat și fezabil, care respectă toate constrângerile proiectului și este optimizat pentru lucrul în echipă.

---

**BACKLOG:**

## Explicația Strategiei Actuale (pentru a o împărtăși cu echipa)

**Salutare echipă,**

**Am rafinat strategia proiectului pentru a face aplicația "SmarTest" mai relevantă și mai puternică, folosind direct materialele de curs. Iată planul actualizat:**

**Strategia Generală: Vom avea un proces în două etape. (1) O etapă Offline, în care vom pre-procesa toate PDF-urile (atât examene, cât și cursuri) pentru a extrage automat "inteligența" necesară. (2) O etapă Online, în care aplicația finală, care rulează la utilizator, va folosi aceste date pre-procesate pentru a genera întrebări în timp real. Această abordare respectă regula "fără LLM la runtime" și face aplicația rapidă și eficientă.**

**Fluxul de Lucru Actualizat:**

1. **FAZA OFFLINE (Development):**
    - **1a. Extragere din Examene: Vom rula un script care parsează PDF-urile cu examene vechi pentru a extrage și a generaliza șabloane de întrebări (ex: "Care este valoarea din rădăcină pentru arborele..."). Rezultatul va fi un fișier `templates.json`.**
    - **1b. Extragere din Cursuri: Simultan, vom rula un alt script care parsează PDF-urile cu slide-uri de curs. Scopul aici este să extragem concepte, definiții și exemple clasice (ex: proprietățile algoritmului A*, graful hărții Australiei). Rezultatul va fi o bază de cunoștințe, un fișier `knowledge_base.json`.**
2. **FAZA ONLINE (Runtime):**
    - **2a. Generarea Întrebărilor: Când utilizatorul cere întrebări, `QuestionGenerator`-ul nostru va folosi ambele fișiere create offline. Poate fie să ia un șablon din `templates.json` și să-l umple cu date noi, aleatorii, fie să folosească `knowledge_base.json` pentru a genera întrebări teoretice sau comparative inteligente (ex: "Comparați BFS și A* din punct de vedere al optimalității.").**
    - **2b. Generarea Răspunsurilor și Evaluarea: Procesul rămâne același, dar acum `AnswerGenerator` poate adăuga definiții din baza de cunoștințe pentru a oferi explicații mai bogate.**

---

## Backlog-ul Actualizat al Proiectului "SmarTest"

**Epopeea 1: Fundația și Colectarea Datelor Sursă (Faza Offline) 📚**

**Obiectiv: Crearea scripturilor care transformă PDF-urile nestructurate în fișiere JSON structurate și utilizabile.**

- **Sarcină 1.1: Setup-ul Proiectului și Versionare (Neschimbat).**
- **Sarcină 1.2 [MODIFICAT]: Implementarea Extractorului de Text din PDF-uri (Examene și Cursuri).**
    - **Descriere: Scriptul din `utils/pdf_parser.py` trebuie să poată gestiona ambele tipuri de documente.**

- **Sarcină 1.3: Dezvoltarea Modulului de Curățare și Segmentare a Textului (Neschimbat).**

---

**Epopeea 2: Crearea "Inteligenței" Proiectului & `core_logic` (Faza Offline + Backend) 🧠**

**Obiectiv: Procesarea textului curățat pentru a crea artefactele inteligente (`templates.json`, `knowledge_base.json`) și, în paralel, implementarea algoritmilor de bază.**

- **Sarcină 2.1 (Track 1 - Motoare de Extracție): Extragerea Șabloanelor din Examene**
  - **Descriere: Crearea modulului `engine/template_miner.py` care generează `templates.json`.**
- **Sarcină 2.2 [NOUĂ] (Track 1 - Motoare de Extracție): Crearea Bazei de Cunoștințe din Cursuri**
  - **Descriere: Crearea unui modul `engine/knowledge_builder.py` care procesează textul curățat din cursuri și generează `knowledge_base.json`.**
- **Sarcină 2.3 (Track 2 - Algoritmi): Implementarea `core_logic` (Partea 1 - Nash & MinMax).**
- **Sarcină 2.4 (Track 2 - Algoritmi): Implementarea `core_logic` (Partea 2 - CSP & Strategii).**

**👥 Alocare în Echipă: Track-ul 1 (extracție) și Track-ul 2 (algoritmi) pot fi lucrate complet în paralel de membri diferiți ai echipei.**

---

**Epopeea 3: Motorul de Generare Inteligentă a Conținutului (Faza Online) ⚙️**

**Obiectiv: Construirea motoarelor principale ale aplicației care folosesc artefactele create anterior.**

- **Sarcină 3.1: Crearea Generatoarelor de Date Procedurale (Neschimbat).**
- **Sarcină 3.2 [MODIFICAT]: Implementarea `QuestionGenerator` cu suport pentru Baza de Cunoștințe.**
  - **Descriere: Clasa trebuie să încarce ambele fișiere JSON și să poată genera atât întrebări bazate pe șabloane, cât și întrebări teoretice bazate pe cunoștințe.**
- **Sarcină 3.3 [MODIFICAT]: Implementarea `AnswerGenerator` cu suport pentru Baza de Cunoștințe.**
  - **Descriere: Clasa poate adăuga definiții sau context suplimentar din `knowledge_base.json` la răspunsurile detaliate.**

---

**Epopeea 4 și 5 (GUI, Evaluare, Finalizare) 🖥️📜**

**Aceste epopei rămân în mare parte neschimbate, dar acum vor beneficia de o fundație de date mult mai bogată.**

- **Sarcină posibilă în GUI: Adăugarea unei opțiuni/unui checkbox "Include întrebări teoretice" care să activeze generarea de întrebări din `knowledge_base.json`.**

**Acest plan actualizat oferă o viziune completă, permite o paralelism excelent în echipă și are potențialul de a produce o aplicație finală mult mai impresionantă.**