LectioSphere – Prompt Engineering

1. Durdun Andrei George

Am utilizat ChatGPT în mai multe etape ale proiectului:

1. Identificarea unei alternative gratuite la ISBN API: Modelul mi-a oferit mai multe opțiuni de API-uri gratuite, fiecare cu propriile limite de utilizare. În urma acestor sugestii, am ales să folosesc Google Books API.

2. Generarea de idei pentru sistemul de recomandare: Am primit sugestii utile privind metode de recomandare.

3. Informații despre embeddings: ChatGPT m-a ajutat să înțeleg cum funcționează vectorii de embedding și cum pot fi utilizați pentru recomandări personalizate.

De asemenea, am folosit GitHub Copilot cu GPT-4.1, în special pentru debugging. Acesta m-a ajutat să identific și să rezolv rapid erori subtile, economisind astfel timp prețios în dezvoltare.

2. Dumitru Andreea-Alexandra

Am folosit ChatGPT și GitHub Copilot pentru:

1) Configurare mediu de dezvoltare:
   a) Recomandări pentru pașii de instalare npm, node, expo-cli.
   b) Curățarea corectă a cache-ului și a fișierelor corupte (ex: node_modules, package-lock.json)

   Am primit indicații bune într-un timp mult mai rapid decât dacă aș fi căutat în documentație sau alte site-uri.

2) Debugging:
   a) Analiza fișierului package.json și sugerarea corectă a versiunilor.
   b) GET requests cu query params
   Exemplu: Cand am primit Network Error la axios.get, deoarece am scris gresit un parametru si am primit un raspuns clar sa ii schimb numele corespunzator: { q: text } în { title: text }.

3) UI si React Native:
   a) Generare componente

   Rezultatul a fost acceptabil, a fost nevoie să vin ulterior și să fac ajustări, în special pe partea de stil unde aveam nevoie sa vizualizez în timp real schimbările pentru a ajunge la interfața dorită.

b) Navigație între ecrane, folosind @react-navigation

Am cerut indicații referitoare la funcționarea navigării și am primit un răspuns foarte bun.

4) Generare de funcții logice: apeluri API folosind axios

Am primit răspunsuri parțial corecte, deoarece nu au funcționat din prima (lipsa de parametri, logica slaba și chiar erori pe care a trebuit să le repar ulterior singură).

3. Nedelcu Ionuț-Daniel

Am folosit o serie de modele AI: Github Copilot (integrat in Visual Studio Code cu Claude Sonnet 3.7, Claude Sonnet 4.0, GPT 4.1), ChatGPT, Gemini, Figma AI.

1) Configurarea frontend:

a) Integrarea bibliotecilor și modulelor esențiale (react-native, axios, @react-navigation, react-native-vector-icons etc.)
b) Corelarea versiunilor între pachete și identificarea conflictelor din package.json
c) Configurarea corectă a mediului de rulare folosind Expo CLI și gestionarea erorilor cauzate de cache sau incompatibilități

Am primit sfaturi clare și concrete privind instalarea și compatibilitatea modulelor, precum și pași de depanare.

2) Conectare cu backend și permisiuni:

a) Configurarea conexiunii frontend–backend, inclusiv gestionarea adreselor IP și a permisiunilor de port în context de rețea locală (implicit legarea de docker a aplicației)
b) Gestionarea corectă a requesturilor către API (inclusiv `CORS`, whitelist de IP-uri, metode permise etc.)

AI-ul m-a ajutat să înțeleg și să configurez mai ușor permisiunile pentru comunicarea între client și server, evitând erori comune precum `Network Error`, `CORS` sau `405 Method Not Allowed`.

3) Debugging și troubleshooting:

a) Diagnosticarea și rezolvarea erorilor HTTP:
    a. 405 Method Not Allowed: am învățat să verific dacă metoda este permisă de views.py sau router.
    b. 500 Internal Server Error: am identificat rapid sursa în backend cu ajutorul mesajelor de eroare și sugestiilor de logare.
b) Sugestii privind debugging cu console.log, try/catch, status code checking, și network tab în dezvoltare mobile

4) Generare de cod și componente:

a) Generare automată de componente React Native reutilizabile
b) Cod repetitiv pentru requesturi axios, validări simple sau layouturi de ecran
c) Recomandarea de a folosi Stack.Navigator pentru gestionarea navigației între ecrane în aplicația mobilă

Am folosit aceste sugestii ca punct de pornire, personalizând ulterior codul în funcție de contextul aplicației mele.

5) Design UI:

a) Utilizarea Figma AI pentru rezolvarea incoerențelor de dimensiuni, spațiere, coerență de culoare ale aplicației
b) Generare automată de structuri de ecrane (wireframes) și ajustarea proporțiilor componentei UI în funcție de conținut

Acest lucru mi-a oferit o viteză mai mare de iterare și o viziune clară asupra fluxului aplicației.

4. Rusu Ana-Maria

Am utilizat ChatGPT ca suport în procesul de învățare și dezvoltare a backend-ului cu Django REST Framework. Modelul m-a ajutat să înțeleg conceptele de bază, să clarific aspecte legate de structurarea proiectului și să aplic bune practici încă de la început.

1. Postări: Am fost ghidată în definirea modelelor, serializer-elor și a endpoint-urilor pentru crearea, afișarea și filtrarea postărilor. ChatGPT mi-a oferit exemple clare, pe care le-am adaptat cerințelor aplicației.

2. Funcționalitatea de follow: Am învățat cum să implementez o relație many-to-many între utilizatori, incluzând logica necesară pentru follow și unfollow, precum și validările corespunzătoare.

3. Profilul utilizatorului: Cu ajutorul sugestiilor primite, am reușit să extind modelul `User` și să gestionez corect datele din profil prin serializer-e personalizate.

De asemenea, am folosit GitHub Copilot pentru a corecta automat greșelile de sintaxă și pentru a obține sugestii rapide în timpul scrierii codului.

5. **Văcaru Marta-Patricia**

Am folosit ChatGPT pentru:

1) Configurare mediu de dezvoltare:

a) Recomandări pentru pașii de instalare și configurare a pachetelor esențiale pentru Django și aplicația mea (ex: django, djangorestframework, selenium, beautifulsoup4, djoser, dj-rest-auth).

b) Soluții rapide pentru curățarea fișierelor corupte sau cache-ului: __pycache__, conflicte în migrations, reinițializare a bazei de date sau ștergere de date invalide.

Am primit indicații precise, mai rapide decât dacă le-aș fi căutat în documentație sau forumuri.

2) Debugging:

a) Identificarea rapidă a problemelor din ViewSet și urls.py – de exemplu, când o metodă personalizată pentru un raft nu era accesibilă prin URL, ChatGPT m-a ajutat să folosesc corect @action(detail=False) și url_path.

b) Am cerut ajutor când datele din ShelfBooks nu se încărcau corect – am aflat că trebuia să folosesc select_related("book") pentru optimizarea interogărilor.

Exemplu concret: am avut probleme la book_status când cărțile nu se regăseau după ISBN; am primit o sugestie bună de a verifica cu b.get("ISBN") == isbn în lista de cărți serializate.

3) UI și organizare logică a datelor (în API și răspunsuri JSON):

a) Generarea de răspunsuri structurate în endpoint-uri precum shelves/ și get_shelf_by_name/, cu separare clară între rafturi standard (Read, Reading etc.) și cele personalizate.

Rezultatul a fost acceptabil, dar a trebuit să vin ulterior să stilizez datele pentru a se potrivi cu frontend-ul.

b) Am cerut explicații despre cum să gestionez navigarea logică între pagini (ex: accesarea unui raft după nume URL-encoded) și am primit o soluție bună cu unquote() din urllib.parse.

4) Generare de funcții logice – adăugare cărți în rafturi, progres de citit și scraping:

a) Am cerut ajutor pentru funcțiile care adaugă o carte într-un raft (add_book_to_shelf) – sugestiile au fost utile, dar nu funcționau 100% din prima: am completat cu verificări suplimentare

b) Pentru web scraping, am primit un exemplu cu selenium + BeautifulSoup, care imi arata de unde ar trebui sa ma documentez si ce structura ar trebui sa folosesc.