**Prompt Engineering**

Pe parcursul dezvoltării aplicației „MemorEasy”, echipa a utilizat în mod activ următoarele modele de limbaj și instrumente de asistență:

* **ChatGPT** (OpenAI) pentru majoritatea task-urilor: clarificarea cerințelor de arhitectură și design, debugg și clarificări legate de tehnologiile folosite(ex: JavaFX, Maven, JDBC, etc.);
* **Claude** (Anthropic) pentru cazurile in care folosirea ChatGPT nu ducea la depanarea bug-ului sau implementarea functionalitatii dorite;
* **Cursor** utilizat pentru generarea de cod (clase, module și logică de business) pe baza întregului proiect. Înțelege structura codului în profunzime și oferă sugestii de refactorizare și evidențiere a logicii esențiale, fiind mai eficient.
* **GitHub Copilot** utilizat pentru generarea testelor unitare și a celor de integrare.

Tool-ul AI folosit in principal a fost ChatGPT. Încă din faza de iniţiere proiectului, ca echipă, ne-am gândit să organizăm aplicația într-un mod eficient și scalabil. Astfel, pe baza recomandărilor ChatGPT am optat pentru crearea proiectul cu Apache Maven care automatizează procesul de construire a aplicației și gestionare a dependențelor pentru proiecte Java. L-am adoptat pentru a genera rapid structura standard a proiectului, a administra automat librăriile necesare și a asigura un build uniform și reproductibil pentru întreaga echipă. În plus, am preferat arhitectura MVC, pentru a separa resursele statice ( fisiere .css, fisiere .fxml ) de logica de back-end (Controller-ele).

Deoarece niciunul dintre noi nu folosise până atunci framework-uri pentru crearea de interfețe grafice ale aplicațiilor Java, precum JavaFX împreună cu SceneBuilder, ChatGPT a reprezentat un tool esențial pentru învățare. Am primit explicaţii despre concepte precum: structura unui proiect JavaFX, “scene graph”, modul de iniţializare a aplicaţiei, precum şi gestionarea tranziţiilor între scene şi transmiterea datelor de la o scenă la alta prin Controller.

Cea mai mare contribuție a avut-o însă în diagnosticarea erorilor din timpul dezvoltării aplicației. Prin analizarea mesajelor de excepție și a stack trace-urilor pe care i le furnizam, ChatGPT ne-a explicat erorile și posibilele cauze. (Un exemplu a fost bug-ul din metoda initialize a controller-ului, unde paginarea trebuia să încarce flashcard-urile pentru un capitol cu un anumit id, însă interfața rămânea goală, Pagination era inițializat înainte ca proprietatea chapterId să fie setată și serviciul de încărcare să returneze datele. ChatGPT a identificat că problema venea din ordinea apelurilor și a sugerat să mutăm logica de inițializare a paginării într-o nouă funcție, și să apelăm un setter dedicat înainte de initialize, asigurându-ne astfel că datele necesare există la momentul construirii paginilor. Astfel, flashcard-urile au aparut corect în pagină.)

Totodată am folosit ChatGPT pentru a genera conceptul de logo al aplicației: un personaj animat, având forma unui creier, care ține în mână flashcard-uri, astfel subliniind ideea de memorie activă și învățare interactivă.

O imagine care conține desen animat, clipart, desen, ilustrație

Conținutul generat de inteligența artificială poate fi incorect.

Deși ChatGPT a rezolvat majoritatea sarcinilor rapid , generând cod cu explicații detaliate, au existat situații în care nu a oferit soluția dorită. În aceste cazuri, am apelat la alte unelte AI, cum ar fi Claude, care de multe ori a găsit rezolvarea încă de la primul prompt. (Un exemplu în care Claude a reușit din câteva prompturi să implementeze funcționalitarea dorită îl reprezintă crearea animației de flashcard, ce a presupus crearea unei clase-Controller: FlashCardController + crearea fisierului .fxml).

Un alt tool AI pe care l-am folosit limitat a fost Cursor. Spre deosebire de alte unelte precum ChatGPT, Cursor are acces la întregul proiect, ceea ce îi oferă o înțelegere profundă a întregii aplicații. Astfel, a reușit să genereze cod contextual, care respecta arhitectura existent și a necesitat intervenție minimă din partea noastră.

L-am utilizat în special pentru crearea logicii de înregistrare și autentificare a utilizatorilor (register + login), a creat clasele-Controller: LoginController + DatabaseUtils, dar și View asociat: LoginView. Totuși are și unele dezavantaje. În varianta gratuită, accesul la funcționalitățile sale este limitat. În plus, uneori face modificări nedorite în fișiere pe care nu i-am cerut în mod explicit să le editeze, ceea ce impune o verificare atentă a schimbărilor propuse.

În procesul de creare a testelor automate am folosit GitHub Copilot pentru a genera teste unitare și teste de integrare.

În concluzie, utilizarea instrumentelor AI a fost esențială pentru realizarea acestui proiect, mai ales având în vedere faptul că acesta a fost dezvoltat cu tehnologii relativ noi și nefamiliare pentru echipa noastră. Prin intermediul acestor tool-uri am reușit să învățăm rapid concepte și abordări de implementare, să obținem explicații clare și să accelerăm procesul de dezvoltare.