Universitatea Stefan cel Mare din Suceava
Facultatea de Inginerie Electrică și Știința Calculatoarelor
Domeniul: Calculatoare și Tehnologia Informației
Program de studii: Calculatoare

Alex - Asistent Virtual

Profesor îndrumător
prof. dr. ing. Turcu Cristina Elena

Student
Șandru Alexandru

Iulie 2025

Cuprins

1. Introducere

Acest proiect reprezintă o nouă funcționalitate integrată în site-ul oficial al Facultății de Inginerie Electrică și Știința Calculatoarelor (FIESC), sub forma unui asistent virtual inteligent denumit Alex. Proiectul are obiectivul principal de a automatiza procesul de informare a studenților, reducând astfel interacțiunea directă cu secretariatul și timpul pierdut în căutarea informației pe site.

Alex este conceput pentru a răspunde întrebărilor frecvente ale studenților, pentru a genera documente oficiale (precum adeverințe), și pentru a oferi sprijin în completarea anumitor formulare sau documente disponibile pe site-ul facultății.

Interacțiunea cu Alex are loc printr-o interfață de tip chat, integrată direct în site-ul facultății. După autentificare, utilizatorul beneficiază de funcționalități suplimentare, precum posibilitatea de a genera adeverințe, păstrarea istoricului conversațiilor și pusul de întrebări mai ușor, deoarece nu mai este necesară reintroducerea constantă a datelor personale, iar răspunsurile sunt mai relevante datorită contextului conversațiilor anterioare.

Acest proiect demonstrează cum inteligența artificială poate fi integrată eficient într-o aplicație web educațională pentru a îmbunătăți accesul la informații și experiența utilizatorului.

1.1. Descrierea problemei

Studenții facultății se confruntă frecvent cu dificultăți în obținerea de documente necesare sau a informațiilor precum cele de bursă, taxe, sau grupa în care face parte, din cauza programului limitat de lucru cu publicul al secretariatului. În multe cazuri, acest lucru implică statul la cozi lungi și pierderea unui timp considerabil doar pentru a depune sau solicita o simplă adeverință. În plus, multe dintre întrebările adresate secretariatului au deja răspunsuri disponibile pe site-ul facultății, însă acestea pot fi greu de găsit din cauza volumului mare de informații. Prin urmare, era necesară o soluție care să fie mai eficientă în accesul la informații și să reducă interacțiunile fizice inutile.

1.2. Justificarea temei

Studenții se confruntă adesea cu dificultăți atunci când trebuie să obțină informații sau documente de la secretariat, mai ales din cauza programului scurt de lucru și a cozilor lungi. În plus, multe dintre întrebările lor au deja răspunsuri pe site-ul facultății, dar sunt greu de găsit.

Alex a fost creat pentru a rezolva aceste probleme. Acesta ajută studenții să găsească rapid informațiile de care au nevoie, cum ar fi detalii despre taxe, burse sau grupa din care fac

parte. De asemenea, studenții pot genera documente oficiale, cum ar fi adeverințe, fără a fi nevoie să aștepte la cozi sau să interacționeze fizic cu secretariatul.

Astfel, proiectul își propune să economisească timp și să simplifice procesul de obținere a informațiilor și documentelor, oferind o soluție rapidă și accesibilă pentru studenți. Totodată, contribuie la digitalizarea activităților administrative, reducând interacțiunile fizice și îmbunătățind eficiența întregului sistem.

1.3. Situația actuală pe plan național/internațional

În România, implementarea asistenților virtuali în mediul educațional este încă la început, dar câteva instituții au început să adopte astfel de soluții pentru a sprijini studenții și a îmbunătăți accesul la informații și servicii administrative.

Un exemplu relevant este Academia de Studii Economice (ASE) din București, care a lansat asistentul virtual „Saro”, un sistem bazat pe inteligență artificială ce ajută studenții cu informații academice și administrative. Acest proiect a fost realizat cu ajutorul unui grant de 700.000 de dolari din partea Google.org, având scopul de a sprijini digitalizarea activităților administrative ale universității.

Această inițiativă demonstrează tendința de integrare a inteligenței artificiale în educație, chiar dacă utilizarea asistenților virtuali în România este încă la început. Proiectele existente au ca scop îmbunătățirea accesului la informații și reducerea timpului pierdut de studenți în procesele administrative.

2. Tehnologii utilizate

Pentru realizarea proiectului *Alex – Asistent Virtual* au fost utilizate tehnologii moderne, atât pe partea de server (back-end), cât și pe partea de interfață (front-end). Alegerea acestora a fost făcută astfel încât să permită integrarea eficientă a unui model de inteligență artificială, dar și o experiență plăcută și intuitivă pentru utilizator.

2.1. Back-End

2.1.1. Django

Pentru partea de back-end, aplicația folosește Django, un framework web scris în Python. Acesta are mai multe roluri esențiale:

- expune un API REST prin care interfața din browser comunică cu serverul,
- gestionează autentificarea utilizatorilor și drepturile de acces,
- salvează întrebările și răspunsurile în baza de date,
- procesează comenzile speciale, cum ar fi generarea de adeverințe în format PDF,
- accesează fișierele și extrage date relevante pentru procesul de generare a răspunsurilor.

2.1.2. Integrarea cu LLaMA 3.2 și RAG

Pentru generarea de răspunsuri inteligente, sistemul integrează modelul LLaMA 3.2 (Large Language Model Meta AI 3.2) printr-un sistem de tip RAG (Retrieval-Augmented Generation). Procesul funcționează astfel:

a. Când utilizatorul trimite o întrebare, aceasta este salvată și analizată.
b. Se extrag cuvintele-cheie (de exemplu „criterii”, „bursă” etc.), ignorând cuvintele comune.
c. Aceste cuvinte-cheie sunt folosite pentru a căuta documente relevante care conțin informațiile necesare.
d. Conținutul documentului găsit este extras și folosit drept contex.
e. Întrebarea și contextul sunt trimise către LLaMA 3.2, care generază un raspuns adaptat.
f. Răspunsul este apoi validat automat și trimis către utilizator.

Dacă întrebarea este o comandă, cum ar fi „Generează o adeverință”, sistemul:

- verifică dacă utilizatorul este logat,

- încarcă șablonul PDF,
- completează automat datele personale,
- generează fișierul și oferă un link de descărcare.

2.2. Front-End

2.2.1. React + Vite

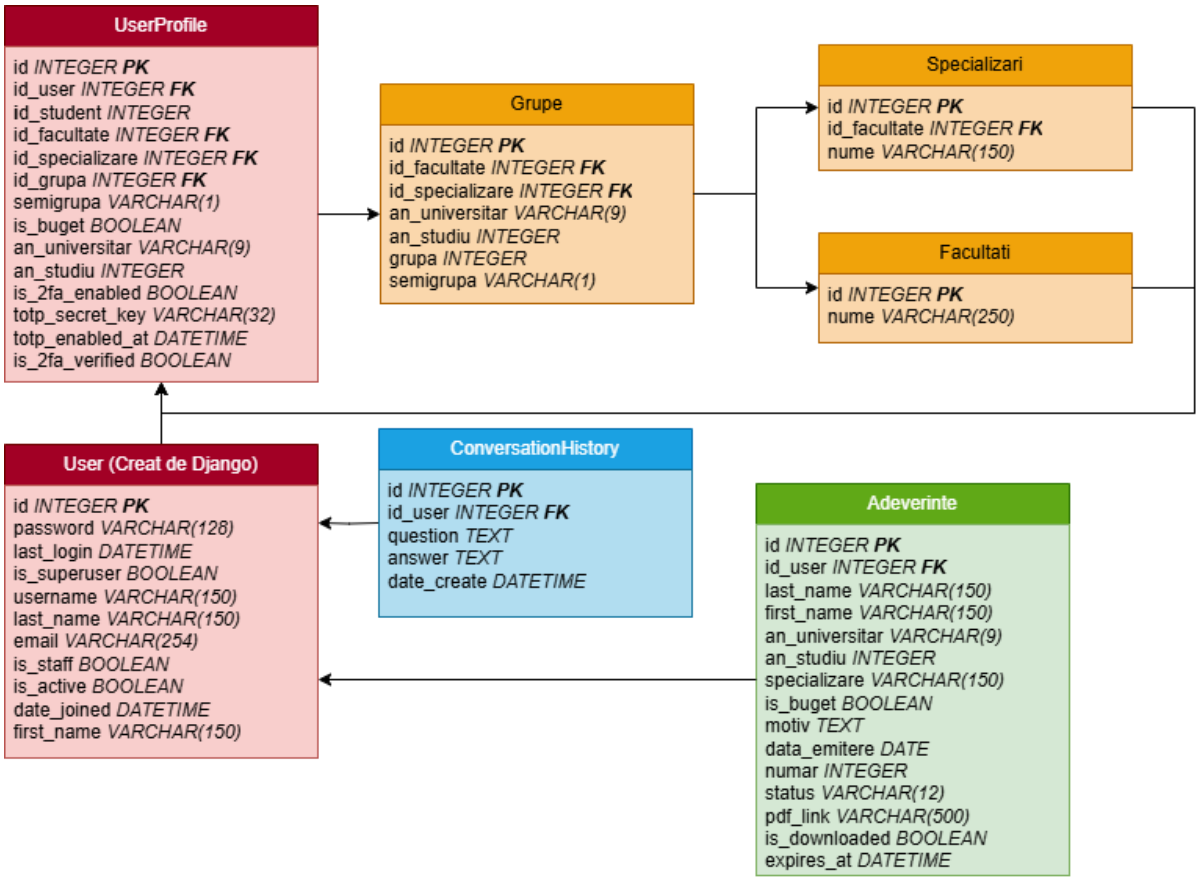
Interfața aplicației este dezvoltată în React, folosind Vite pentru o încărcare rapidă și un flux de lucru modern. Utilizatorul interacționează astfel:

- La accesarea site-ului, apare o versiune a paginii oficiale a FIESC.
- În colțul din dreapta jos există un buton „Alex – Asistent Virtual”.
- La apăsarea butonului, se deschide o fereastră de tip chat, unde utilizatorul poate pune întrebări.
- Dacă utilizatorul se autentifică, are acces la funcții suplimentare:
    - Poate genera documente (de exemplu adeverințe),
    - Are acces la istoricul conversațiilor,
    - Întrebările sunt mai ușor de pus, pentru ca Alex reține contextul anterior.

3. Considerații de proiectare

3.1. Proiectarea bazei de date

Baza de date este implementată în SQLite, un sistem de gestiune a bazelor de date relaționale, integrat cu Django. Aceasta conține mai multe tabele care stochează informațiile necesare pentru funcționarea aplicației. Structura este construită astfel încât să permită extinderea ușoara a sistemului și menținerea unei organizări clare a datelor.



Figură 3.1 - Schema bazei de date

3.1.1. Tabele și relațiile dintre tabele

Baza de date este compusă din următoarele tabele:

1. Facultăți:

- Conține lista facultăților din cadrul instituției.
- Câmpuri: id (PK), nume.

2. Specializări:
- Reprezintă specializările disponibile în cadrul unei facultăți.
- Relație: FK către Facultăți.
- Câmpuri: id (PK), id_facultate (FK), nume.
3. Grupe:
- Conține informațiile despre grupele și semi-grupele studenților.
- Relații: FK către Facultăți și Specializări.
- Câmpuri: id (PK), id_facultate (FK), id_specializare (FK), an_universitar, an_studiu, grupa, semigrupa.
4. User (creat automat de Django):
- Conține datele de baza ale conturilor utilizatorilor.
- Câmpuri: id (PK), password, last_login, is_superuser, username, last_name, email, is_staff, is_active, date_joined, first_name.
5. UserProfile:
- Extinde informațiile despre utilizatorii înregistrați.
- Relații: FK către User, Facultăți, Specializări și Grupe.
- Câmpuri: id (PK), id_user (FK), id_student, id_facultate (FK), id_specializare (FK), id_grupa (FK), semigrupa, is_buget, an_universitar, an_studiu, is_2fa_enabled, totp_secret_key, totp_enabled_at, is_2fa_verified
6. Adeverințe:
- Păstrează datele despre adeverințele generate de utilizatori.
- Relație: FK către User.
- Câmpuri: id (PK), id_user (FK), last_name, first_name, an_universitar, an_studiu, specializare, is_buget, motiv, data_emitere, numar, status, pdf_link, is_downloaded, expires_at.
7. ConversationHistory:
- Stochează istoricul întrebărilor și răspunsurilor oferite de asistentul virtual.
- Relație: FK către User.
- Câmpuri: id (PK), id_user (FK), question, answer, date_create.

3.2. Proiectare Back-End

Partea back-end este proiectat în limbajul de programare Python, folosind framework-ul Django, pentru dezvoltarea aplicațiilor web. Acesta gestionează logica aplicației, interacțiunile cu baza de date și comunicarea cu front-end.

3.2.1. Funcționalități administrative

Sistemul include și o serie de endpointuri pentru activități administrative, accesibile doar de către conturile cu rol de administrator. Pentru început ei au posibilitatea de a adăuga utilizatori noi în baza de date folosind un fișier exel. Aceștia mai au posibilitatea de actualiza la apăsarea unui buton, toate documentele esențiale utilizate de modelul LLaMA 3.2 în cadrul mecanismului RAG, pentru a putea furniza răspunsuri corecte și relevante pe baza informațiilor.

3.2.2. Funcționalități de securitate

Utilizatorii au posibilitatea de a se autentifica în aplicație pentru a beneficia de funcționalități suplimentare cum ar fi:

- Posibilitatea de a folosi următoarele comenzi „comenzi” care îți afișează toate comenzile valabile, și „Generează o adeverința cu motivul [motiv]” care pe baza informațiilor aflate în baza de date îți va genera o adeverința pe care o poți descarcă și folosi imediat.
- Salvarea istoricul conversației pentru a face conversația cu LLaMA 3.2 mai ușoară pentru că acesta are acces la ceea ce sa vorbit anterior și nu mai trebuie ca utilizatorul să repete anumite lucruri la fiecare întrebare.
- Pentru utilizatorii care își doresc să își creeze un cont trebuie sa folosească un email de student care se află in baza de date.

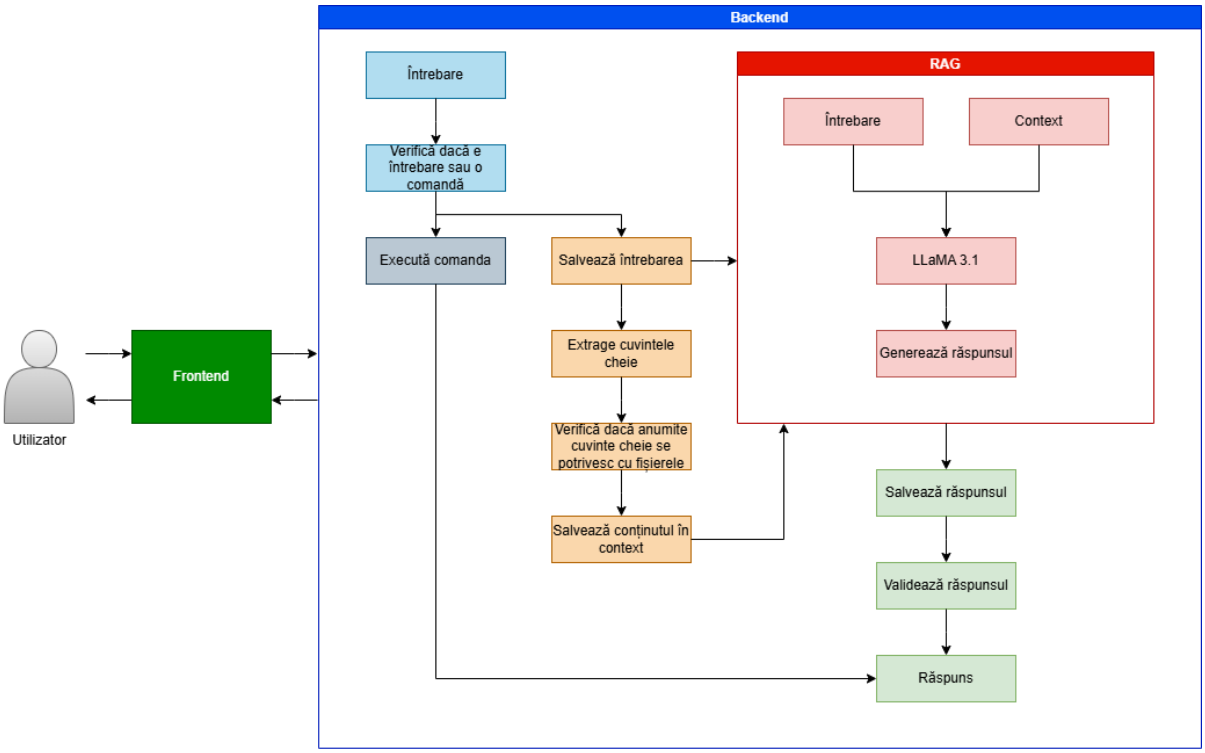
3.2.3. Procesare întrebărilor și comenzilor

Când utilizatorul interacționează cu sistemul prin interfața de tip chat:

- Întrebarea este trimisă prin front-end către back-end.
- Back-end verifică dacă este întrebare sau comandă.
- Dacă este întrebare, se salvează, se extrag cuvintele cheie, acestea se verifică dacă se potrivesc cu un cuvânt aflat în numele unui document, și conținutul acestuia este

salvat si trimis la modulul de RAG, care generează un răspuns, acesta este validat și returnat către utilizator.

- Dacă este comandă, se verifică dacă utilizatorul este autentificat, după se verifică ce comandă a fost executată, dacă e „comenzi” returnează toate comenzile valabile, dacă e „Generează o adeverința cu motivul [motiv]” va returna un link către un fișier PDF cu o adeverință completată automat pe baza informației corespunzătoare utilizatorului autentificat.



Figură 3.2 - Schema de funcționare back-end

3.3. Proiectare Front-End

Partea front-end este realizată utilizând biblioteca JavaScript React, împreuna cu Vite. Acesta este responsabilă de interacțiunea directă cu utilizatorul, afișarea datelor preluate de la back-end si transmiterea cererilor către server.

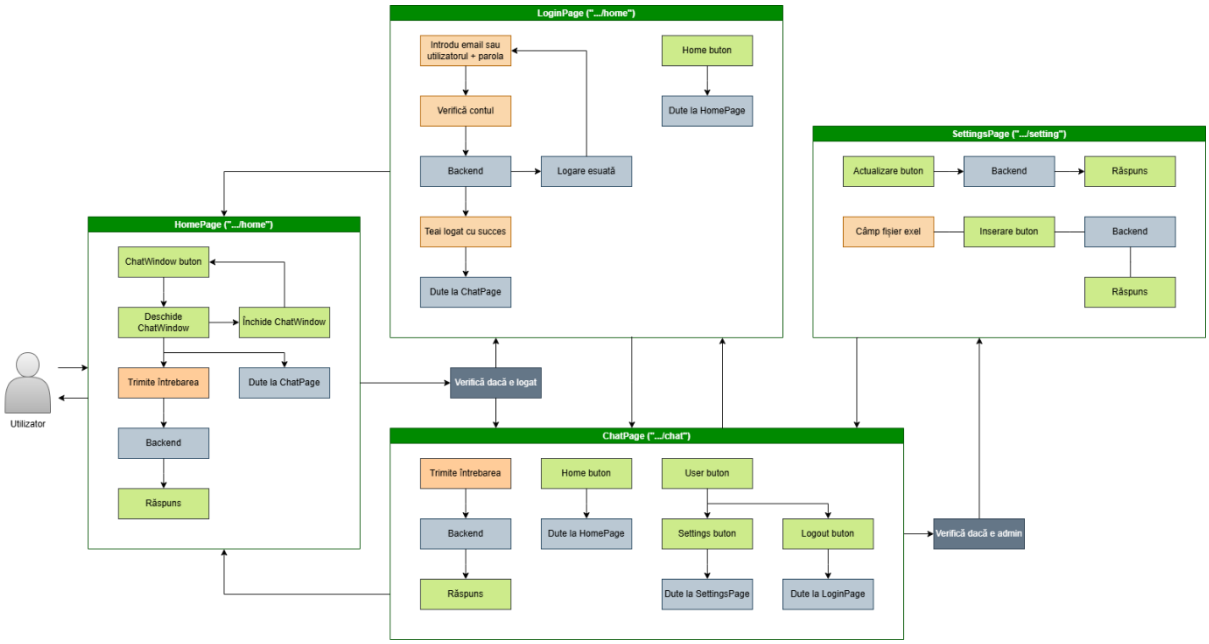
3.3.1. Paginile si componentele aplicatiei

Aplicația front-end este alcătuită din patru pagini principale si o componentă importantă. Fiecare pagină contribuie la definirea structurii aplicației și a fluxul de utilizare, având un rol important în interacțiunea utilizatorului.

Componenta „ChatWindow” funcționează ca punct de acces principal către funcționalitatea aplicației și este integrată în pagina principală, care reproduce vizual site-ul oficial al facultății prin intermediul unui „iframe”. Această componentă este un chat rapid si ușor de accesat pentru întrebări scurte si rapide adresate asistentului virtual.

Din această componentă utilizatorul poate să se autentifice redirecționându-l către „LoginPage”, din această pagină utilizatorul își poate crea un cont redirecționându-l către „RegisterPage”, dar pentru acest lucru el trebuie să fie în posesia unui cont de student care se află în baza de date. Dacă contul e valid e automat redirecționat către „ChatPage” acesta este un chat mai mare în care îi este afișat istoricul conversației.

Dacă utilizatorul este admin atunci acesta poate intra pe pagina de setări redirecționându-l către „SettingPage” în care va avea acces la un buton de update a documentelor esențiale asistentului virtual pentru a genera răspunsuri corecte, și poate insera noi utilizatori în baza de date folosind un fișier exel.



Figură 3.3 - Schema de funcționare front-end

4. Considerente de implementare

4.1. Implementarea Back-End

4.2. Implementarea Front-End

5. Manual de utilizare

5.1. Cerințe de sistem

Cerințe minime:

- Placă video: NVIDIA RTX 3050 Ti cu 4 GB VRAM
- Procesor: AMD Ryzen 7 6800H
- Memorie RAM: 16 GB DDR5
- Sistem de operare: Windows 10 sau mai recent

Cerințe recomandate:

- Placă video: NVIDIA RTX 4070 cu 8 GB VRAM
- Procesor: AMD Ryzen 7 7745HX
- Memorie RAM: 32 GB DDR5
- Sistem de operare: Windows 10 sau mai recent

5.2. Instalare

Clonează repository-ul folosind comanda:

git clone https://github.com/Xhadrines/Alex-VirtualAssistant.git

5.2.1. Configurare Back-End

Pașii pentru configurarea back-end:

1. Pentru a rula aplicația, trebuie să ai instalat:

- Python (v3.13.2 sau o versiune mai mare)
- pip (v24.3.1 sau o versiune mai mare)

2. Asigură-te că ai instalat gtk3-runtime de pe https://github.com/tschoonj/GTK-for-Windows-Runtime-Environment-Installer/releases
3. Asigură-te că te afli în directorul backend
4. Creează mediul virtual folosind comanda: *python -m venv .venv*
5. Activează mediul virtual folosind comanda: *.\.venv\Scripts\activate*
6. Instalează dependențele folosind comanda: *pip install -r requirements.txt*
7. Crează migrațiile folosind comanda: *python manage.py makemigrations*
8. Aplică migrațiile folosind comanda: *python manage.py migrate*
9. Instalează Ollama mergând pe site-ul oficial Ollama.
10. Instalează modelul Llama 3.2 folosind comanda: *ollama pull llama3.2*

11. Rulează proiectul folosind comanda: *python manage.py runserver*
12. Proiectul rulează pe: *http://127.0.0.1:2307/*

5.2.2. Configurare Front-End

Pașii pentru configurare front-end:

1. Pentru a rula aplicația, trebuie să ai instalat:
- Node.js (v22.13.1 sau o versiune mai mare)
- npm (v10.9.2 sau o versiune mai mare)
2. Asigură-te că te afli în directorul frontend
3. Creează un fișier .env și adaugă următoarea linie:

VITE_CHAT_API=http://<IP_BACKEND>:<PORT_BACKEND>

Note: Înlocuiește <IP_BACKEND> și <PORT_BACKEND> cu valorile corespunzătoare pe care le obții de la backend.

4. Instalează dependențele folosind comanda: *npm install*
5. Rulează proiectul folosind comanda: *npm run dev*
6. Proiectul rulează pe: *http://127.0.0.1:2002/*

6. Rezultate experimentale

7. Concluzii și direcții viitoare de dezvoltare

Stadiul actual al proiectului

Baza de date:

- Complet, posibil câteva retușuri.

Back-End:

- Trebuie optimizat sistemul de RAG. (dacă sunt puține fișiere merge bine, dacă sunt prea multe fișiere nu mai merge)
- De făcut retușuri în cod.

Front-End:

- Complet, posibil câteva retușuri.

Bibliografie

https://ase.ro/

https://www.djangoproject.com/

https://www.drawio.com/

https://www.geeksforgeeks.org/large-language-model-llm/

https://www.geeksforgeeks.org/what-is-retrieval-augmented-generation-rag/

https://newsbucuresti.ro/ase-bucuresti-prima-universitate-din-romania-care-introduce-un-asistent-ai-pentru-studenti/

https://ollama.com/

https://react.dev/

https://sqlite.org/index.html

https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/

Anexe