VOIS Summer School
Team 6

Task Management web application

Team members:

Frontend developers: Andreescu Andrei-Vlad
Ungureanu Radu-Ionut
Backend developers: Hagiu Alexandru-Daniel
Nicoara Theodora

1) Project Overview

Scopul aplicatiei este de gestionare a task-urilor pentru utilizatori si echipe
Obiective:

-Crearea, actualizarea, ștergerea și vizualizarea task-urilor (CRUD).
-Organizarea utilizatorilor în echipe.
-Autentificare și autorizare cu JWT (separare între utilizatori și administratori).

Tehnologii utilizate pentru realizarea aplicatiei:

1)Backend:

-**Python 3.11** – limbajul de programare folosit.
-**FastAPI** – framework web rapid, ideal pentru construirea API-urilor REST.
-**SQLAlchemy** – ORM (Object Relational Mapper) pentru manipularea bazei de date.
-**SQLite** – bază de date locală, simplă și ușor de configurat.
-**Passlib** – folosit pentru hashing-ul parolelor (bcrypt).
-**Python-Jose** – utilizat pentru generarea și validarea token-urilor JWT.

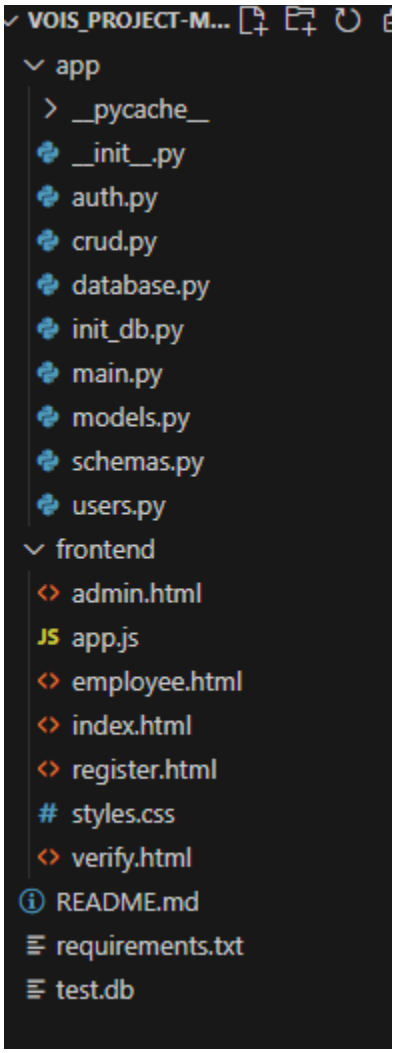
2)Frontend

-**HTML simplu** (admin.html, employee.html, index.html etc.) – fără framework-uri mari (gen React, Vue sau Angular).
-**CSS extern** (styles.css) – pentru temă și layout.
-**JavaScript vanilla** (app.js) – pentru toată logica de interacțiune. Nu au fost folosite biblioteci mari (doar o librărie mică pentru iconițe).
-**Feather Icons** – pentru a desena iconițe (logout, dark/light mode).
-**LocalStorage** – pentru a păstra sesiunea (token, rolul de admin sau user, username, chat fallback etc.).
-**Fetch API** – pentru a comunica cu backend-ul (/auth/login, /tasks/, /teams/ etc.).
-**Progress bar nativ** – pentru progresul taskurilor.
-**Fallback logic** – dacă backend-ul pică sau nu răspunde, salvează chat-ul și cererile în localStorage ca să nu se piardă complet.

3)Alte tool-uri

-**GitHub** – pentru versionare și colaborare în echipă.
-**VS Code + Live Share** – pentru dezvoltare colaborativă.
-**Uvicorn** – server ASGI pentru rularea aplicației FastAPI.

2) Structura Proiectului



Rolurile fisierelor:

-**main.py** – punctul de pornire al aplicației. (unde se afla aproape toate endpoint-urile si dependentele)
-**models.py** – definește tabelele din baza de date (User etc.).
-**database.py** – configurarea conexiunii la baza de date.
-**schemas.py** – definește structurile Pydantic pentru validarea datelor.
-**auth.py** – gestionează login, înregistrare, JWT și protecția endpointurilor.
-**users.py** – endpointuri legate de utilizatori (vizualizare etc.).
-**crud.py** – loc pentru operații cu baza de date (momentan nefolosit).
-**admin.html** - interfata paginii pentru admini
-**app.js** - defineste logica de interactiune a fisierelor intre ele

-**employee.html** - interfata paginii pentru angajati
-**index.html** - pagina principala a aplicatiei
-**register.html** - interfata paginii pentru inregistrare
-**style.css** - fisierul pentru stilurile vizuale utilizate

3) Functionalitati implementate

A)Înregistrare utilizator

Endpoint: POST /auth/register
Permite crearea unui utilizator nou, cu următoarele câmpuri:

-username (unic)
-password (hash-uit înainte de salvare)
-is_admin (boolean – dacă utilizatorul este administrator)

B) Autentificare utilizator

Endpoint: POST /auth/login
Primește un username și password.
Verifică existența utilizatorului și corectitudinea parolei.
Returnează un JWT token, care trebuie trimis în request-urile ulterioare.

C) Vizualizare utilizatori

Endpoint protejat: GET /auth/admin/users
Necesită autentificare ca admin.
Returnează lista tuturor utilizatorilor din baza de date.

D) Vizualizare utilizatori (simplu)

Endpoint: GET /users
Returnează lista de utilizatori.
Util pentru testare și pentru interfața generală.

E) Autentificare și securitate

Sistemul folosește JWT (JSON Web Tokens):

-La login, serverul returnează un token.

-Acest token trebuie trimis în Authorization: Bearer <token> la request-urile protejate.

-Token-ul include informații despre utilizator (username, is_admin) și are o durată limitată de viață (30 minute implicit).

-Parolele nu sunt stocate în clar, ci hash-uite cu bcrypt.

4) Configurare si rulare

A)Instalare dependențe

Toate dependențele se află în fișierul requirements.txt. Instalare:

pip install -r requirements.txt

B)Rulare Server

In terminal se utilizeaza comanda:

uvicorn app.main:app --reload

Aplicația rulează implicit pe: *http://127.0.0.1:8000.*

C)Testare API

Documentația interactivă este disponibilă la: *http://127.0.0.1:8000/docs*

5) Defecte si posibile imbunatatiri

A)Deploy pe un server cloud (ex: AWS).
B)Adăugarea de email verification și reset password.
C)Protectie facuta prin frontend (pagina de angajati nu butoane de acces la functiile unui admin, in schimb nu exista nicio forma de securitate care sa ii blocheze in a accesa privilegiile acestora)
D)Baza de data creata prin SQLite ofera o capacitate de stocare a datelor limitata
E)Avantajul prinicipal este posibilitatea modificarii proiectului si fisierelor