InfoEducație 2023
Etapa Națională

SCHOOLSYNC

DOCUMENȚATIA TEHNICĂ

SOFT EDUCAȚIONAL

Elev: Neaună Mădălin
Profesor: Amarie Bogdan

Liceul Regina Maria Dorohoi, Județul Botoșani

Cuprins

1. Prezentarea generală a temei

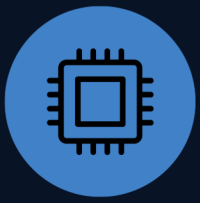
SchoolSync este o aplicație educațională inovatoare care facilitează comunicarea și colaborarea între elevi și cadrele didactice. Aceasta are drept scop creșterea eficienței procesului de învățare și îmbunătățirea experienței educaționale în general.

SchoolSync este o aplicație educațională care îmbină:

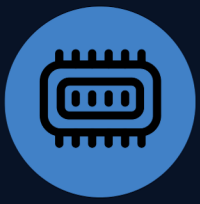
- **EduMentor**: platforma care oferă un mediu centralizat în care profesorii pot încărca și partaja materiale didactice precum lecții, prezentări, fișiere și alte resurse relevante;
- **InvațăUnit:** elevii pot posta întrebări referitoare la teme sau subiecte pentru care au nevoie de ajutor sau clarificări;
- **EduClass:** platformă inovatoare care oferă o clasă virtuală, concepută pentru a facilita procesul de învățare și predare online într-un mediu interactiv și eficient. EduClass oferă o experiență educațională captivantă, permițând studenților și profesorilor să se conecteze și să interacționeze într-un mod ușor și accesibil;
- FlowTalk: este o platformă de comunicare și colaborare care facilitează interacțiunea și schimbul de informații între utilizatori. Acesta este conceput pentru a crea un mediu fluid și în flux, în care conversațiile pot să curgă fără efort și în care participanții pot comunica într-un mod natural și eficient;
- **TimePlan:** este o platformă avansată de gestionare a timpului și programare, concepută pentru a vă ajuta să organizați și să coordonați eficient activitățile și evenimentele din viața dumneavoastră.

2.1. Resurse hardware necesare rulării aplicației

OS: Windows 10 (minim)



Procesor: x86 sau x64

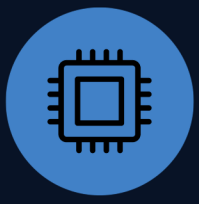


RAM: 1GB (minim) 2GB (recomandat)

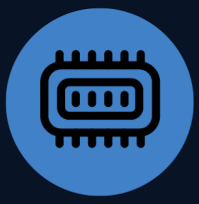
Stocare: 50 MB necesar pentru instalare

2.2. Resurse hardware necesare rulării serverului

OS: Distribuție Linux, recomandat cu cPanel



Core CPU: 1,8GHz (minim), 3 x 2,2GHz (recomandat)



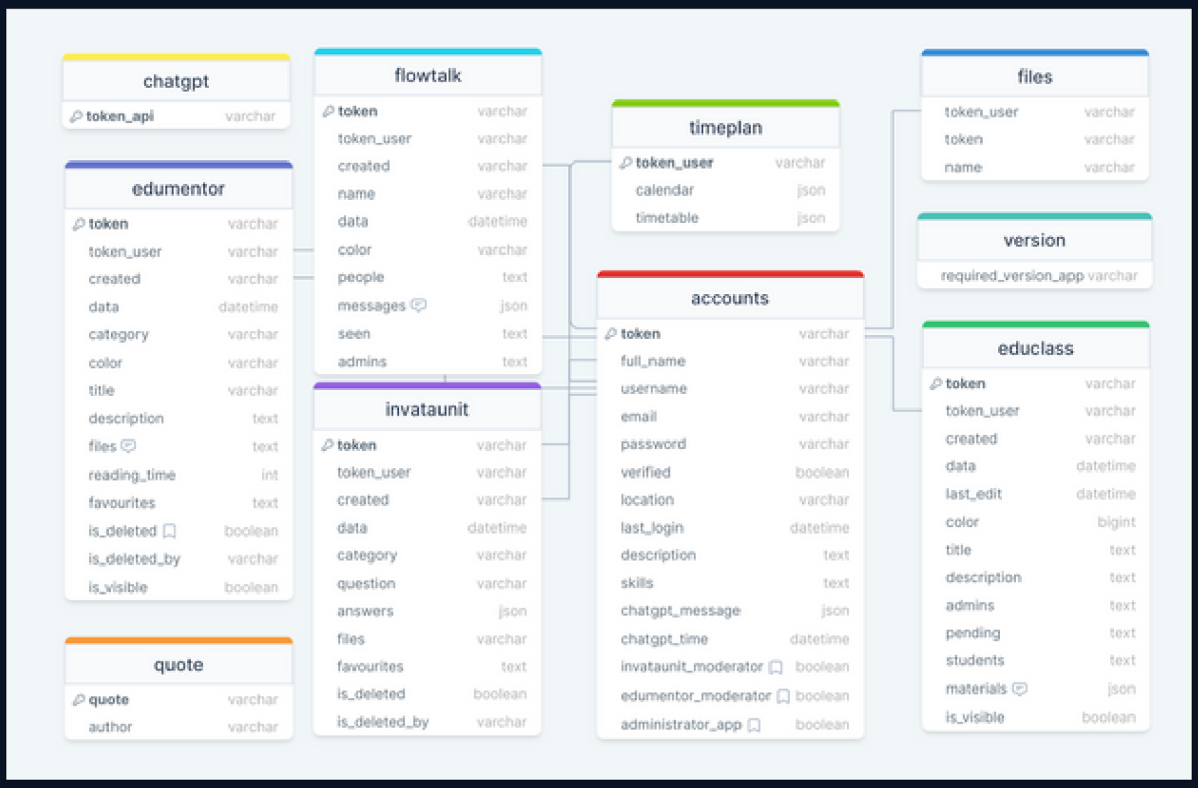
RAM: 1GB (minim), 3GB (recomandat)

Stocare: 10GB (minim), 50GB (recomandat)

3. Realizarea aplicației

3.1 Structura bazei de date

- Baza de date: MySql;
- Fiecare tabelă are un token care este generator automat, având un format de 128 caractere și conține litere mici, litere mari, numere și semne;
- Parola fiecărui utilizator este mai întâi criptată (SHA256), apoi este adăugată în baza de date;
- Atunci când înregistrăm un utilizator, valorile **username** si **email** trebuie să fie unice în baza de date (nu pot exista 2 utilizatori cu același email sau nume de utilizator);



3.2 API

```php
<?php
    ob_start();
    include "database.php";
    ob_end_clean();

    $Allowedcommands = array(
        "SELECT",
        "select",
    );
    $token  = $_POST["token"];

    include "functions.php";
    $token_valid = getEnvValue("API_KEY");

    if($data['Database connection'] == 'Failed'){
        $data = ['message' => 'database no connection'];
        echo json_encode( $data );
        die();
    }

    if($token_valid != $token){
        $data = ['message' => 'token invalid'];
        echo json_encode( $data );
        die();
    }

    $command = $_POST['command'];
    $params = json_decode($_POST['params']);

    $temp= explode(" ", $command);
    if(count(array_intersect($Allowedcommands, $temp)) == 0){
        $data = ['message' => 'SQL command error'];
        echo json_encode( $data );
        die();
    }

    $stmt = $conn->prepare($command);
    if(!$stmt){
        $data = ['message' => 'something went wrong - stmt prepare'];
        echo json_encode( $data );
        die();
    }
```

```php
if (!empty($params) && is_object($params)) {
    $paramTypes = '';
    $paramValues = [];

    foreach ($params as $key => $value) {
        $paramValues[] = $value;

        if (is_int($value)) {
            $paramTypes .= 'i';
        } elseif (is_float($value)) {
            $paramTypes .= 'd';
        } else {
            $paramTypes .= 's';
        }
    }

    $paramPlaceholders = implode(', ', array_fill(0, count($paramValues), '?'));
    $commandWithParams = str_replace('?', $paramPlaceholders, $command);
    $stmt->bind_param($paramTypes, ...$paramValues);
}

$data = ['message' => 'something went wrong'];

try{
    if ($stmt->execute()) {

        $data = array();
        $index = 0;

        $data = ["message" => "success"];
        $result = $stmt->get_result();
        if(mysqli_num_rows($result) > 0){
            while ($row = $result->fetch_assoc()) {
                $data[$index] = $row;
                $data[$index]['profile'] = json_decode($data[$index]['profile']);
                $index++;
            }
        }
        else{
            $data = ['message' => "database no value"];
        }

        $result->free();
```

Functia de mai sus este de tip GET, și utilizează următoarele elemente:

1. **Utilizarea output buffering:** Folosirea **ob_start()** și **ob_end_clean()** pentru gestionarea bufferelor de ieșire este o practică bună pentru a asigura că nu există niciun output neașteptat înainte de generarea răspunsului JSON. Astfel, se evită potențialele erori legate de outputul anterior.
2. **Separarea funcționalităților:** Codul este împărțit în mai multe fișiere, cum ar fi "database.php" și "functions.php", pentru a separa diferitele funcții și responsabilități. Aceasta facilitează întreținerea și reutilizarea codului.
3. **Validarea tokenului:** Se verifică validitatea tokenului trimis în cererea HTTP POST, comparându-l cu valoarea tokenului preluată din mediul de execuție. Acest lucru asigură că doar cererile cu un token valid pot accesa API-ul și execută comenzile SQL.
4. **Filtrarea comenzilor SQL permise:** Prin utilizarea unui array (**$Allowedcommands**) care conține doar comenzile SQL permise, se previne executarea altor tipuri de comenzi care ar putea afecta baza de date sau aduce vulnerabilități de securitate.
5. **Utilizarea declarațiilor pregătite (prepared statements):** Pentru a evita atacurile de tipul SQL Injection, codul utilizează declarații pregătite pentru a executa comenzile SQL cu parametri. Aceasta face ca datele introduse de utilizatori să fie tratate ca date și să nu fie interpretate ca instrucțiuni SQL.
6. **Manipularea rezultatelor din baza de date:** După executarea comenzii SQL, se prelucrează rezultatele și se transformă într-un format JSON corespunzător pentru a fi returnat către clientul API-ului.
7. **Tratarea excepțiilor:** Codul include o construcție try-catch pentru a captura și trata eventualele excepții care pot apărea în timpul execuției comenzilor SQL. Aceasta contribuie la o mai bună gestionare a erorilor și oferă un răspuns controlat în cazul unei excepții.
8. **Eliberarea resurselor:** După obținerea rezultatelor din baza de date, se eliberează resursele asociate pentru a evita consumul inutil de memorie.

```json
{
    "message": "success",
    "0": {
        "token": "05ry3c40xnC5s7i5M4K6125mTs3eRTP...",
        "full_name": "Marius Sabotnicu",
        "username": "Marius",
        "email": "sabotnicumarius@yahoo.com",
        "password": "...",
        "verified": 1,
        "location": "D",
        "last_login": "2023-06-11 23:53:28",
        "description": "LRMD",
        "skills": "",
        "invataunit_moderator": 0,
        "edumentor_moderator": 0,
        "administrator_app": 0,
        "profile": null
    }
}
```


Exemplu de fișier JSON, returnat de API, GET

3.3 Conexiunea dintre baza de date și aplicație

```csharp
public async Task<dynamic> PostRequestAsync(string url, Dictionary<string, string> data)
{
    var client = new HttpClient();
    var content = new FormUrlEncodedContent(data);
    var response = await client.PostAsync(url, content);
    var responseString = await response.Content.ReadAsStringAsync();
    dynamic json = JsonConvert.DeserializeObject(responseString);
    return json;
}
```

Funcția **PostRequestAsync** primește un **URL** și un dicționar de date ca parametri și returnează un obiect dinamic **Task<dynamic>**.

1. Se creează un nou obiect **HttpClient**, care este utilizat pentru a efectua cereri **HTTP**.
2. Datele din dicționarul data sunt convertite într-un conținut de tip **FormUrlEncodedContent**. Aceasta reprezintă o reprezentare a datelor în formatul **URL-encodat**.
3. Se trimite o cerere de tip **POST** către **URL-ul** specificat
4. Răspunsul de la cerere este citit asincron și convertit într-un șir de caractere utilizând **ReadAsStringAsync()**.
5. Șirul de caractere al răspunsului este deserializat folosind **JsonConvert.DeserializeObject**, care convertește șirul de caractere **JSON** într-un obiect dinamic.
6. Obiectul rezultat este returnat.

Funcția este marcată cu **async** pentru a permite utilizarea operațiilor asincrone, ceea ce înseamnă că pot exista întârzieri în timpul așteptării răspunsului de la cerere și în timpul citirii răspunsului. Utilizarea **async** permite apelarea acestei funcții **fără a bloca firul principal de execuție**, astfel încât alte operații pot continua în timpul așteptării răspunsului **HTTP**.

3.4 Incărcarea fișierelor pe server

```
public async Task<dynamic> UploadFileAsync( Dictionary<string, string> data, string filePath = null)
{
    string url = "https://schoolsync.nnmadalin.me/api/upload_file.php";
    var client = new HttpClient();
    var multipartContent = new MultipartFormDataContent();
    foreach (var keyValuePair in data)
    {
        multipartContent.Add(new StringContent(keyValuePair.Value), keyValuePair.Key);
    }
    if (!string.IsNullOrEmpty(filePath))
    {
        var fileContent = new StreamContent(File.OpenRead(filePath));
        var filePart = new ByteArrayContent(await fileContent.ReadAsByteArrayAsync());
        filePart.Headers.ContentDisposition = new ContentDispositionHeaderValue("form-data")
        {
            Name = "file",
            FileName = Path.GetFileName(filePath)
        };
        multipartContent.Add(filePart);
     }
    var response = await client.PostAsync(url, multipartContent);
    var responseString = await response.Content.ReadAsStringAsync();
    dynamic json = JsonConvert.DeserializeObject(responseString);
    return json;
}
```

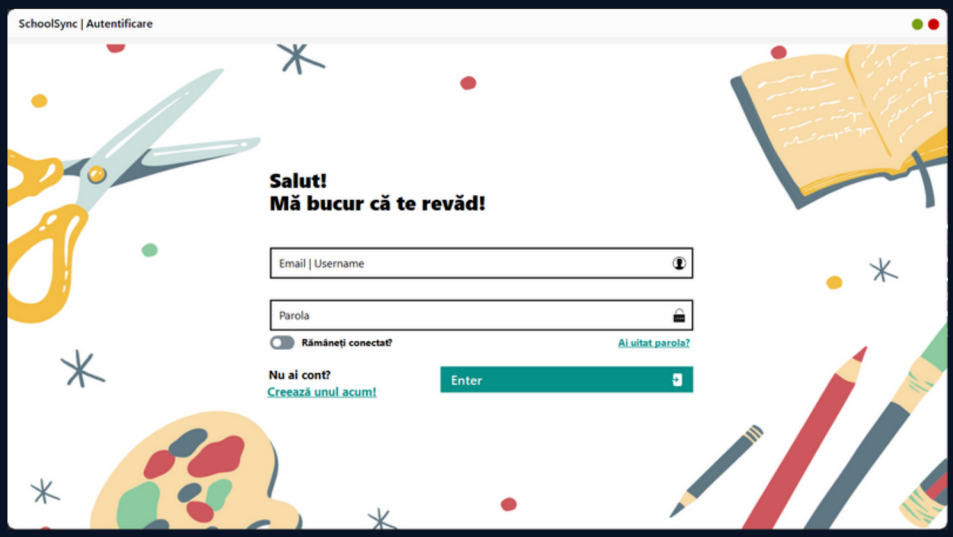
Funcția **UploadFileAsync** primește un dicționar de date și un șir de caractere **filePath** opțional și returnează un obiect dinamic **Task<dynamic>**.

1. Se definește **URL-ul** către care se va face încărcarea fișierului.
2. Se creează un nou obiect **HttpClient**, care va fi utilizat pentru a efectua cereri **HTTP**.
3. Se creează un obiect **MultipartFormDataContent**, care va conține atât datele cât și fișierul pentru încărcare.
4. Pentru fiecare pereche cheie-valoare din dicționarul data, se adaugă un conținut de tip **StringContent** în obiectul **multipartContent**.
5. Dacă șirul de caractere **filePath** nu este nul sau gol, se adaugă și conținutul fișierului pentru încărcare în obiectul **multipartContent**.
   - Se deschide fișierul specificat și se creează un obiect **StreamContent** pentru a citi conținutul fișierului.
   - Se creează un obiect **ByteArrayContent** folosind conținutul fișierului ca un vector de octeți.
   - Se configurează antetul **ContentDisposition** al obiectului **ByteArrayContent** pentru a specifica numele și numele fișierului în formularul de încărcare.
   - Obiectul **ByteArrayContent** este adăugat la obiectul multipartContent.
6. Se trimite o cerere de tip **POST** către **URL-ul** specificat, utilizând conținutul multipart creat anterior.
7. Răspunsul de la cerere este citit asincron și convertit într-un șir de caractere utilizând **ReadAsStringAsync()**.
8. Șirul de caractere al răspunsului este deserializat folosind **JsonConvert.DeserializeObject**, care convertește șirul de caractere **JSON** într-un obiect dinamic.
9. Obiectul rezultat este returnat.

4. Utilizarea aplicației

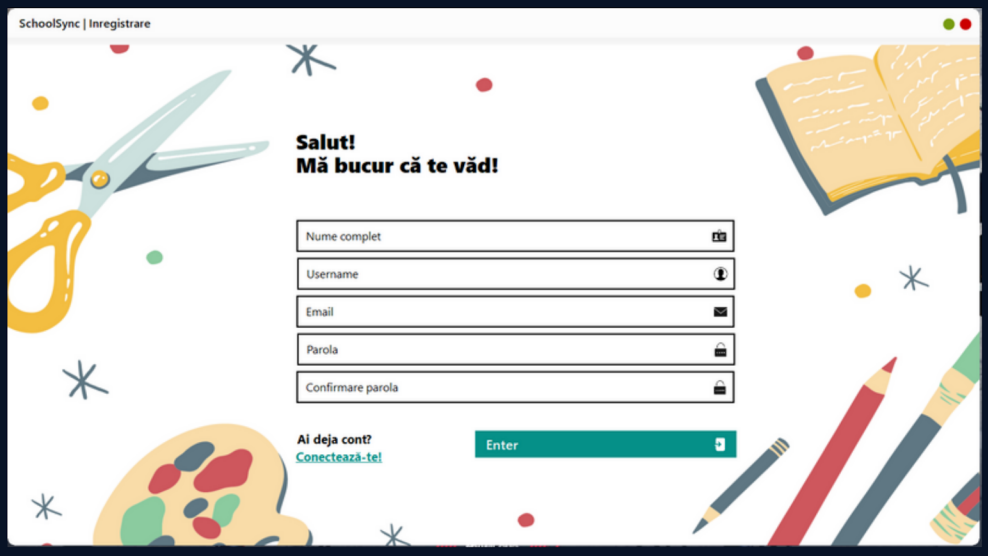
Pagina de autentificare

Atunci când aplicația este deschisă, utilizatorul este întâmpinat de pagina de autentificare. Acesta trebuie să introducă **email-ul** sau **numele de utilizator** și parola. Dacă autentificarea a avut loc cu succes, utilizatorul este salvat în memorie **(aplicația permite autentificarea automată)** și va fi redirecționat către pagina principală! Dacă utilizatorul și-a uitat parola, acesta poate selecta opțiunea "**Ai uitat parola**" și va primi pe email o parola nouă!



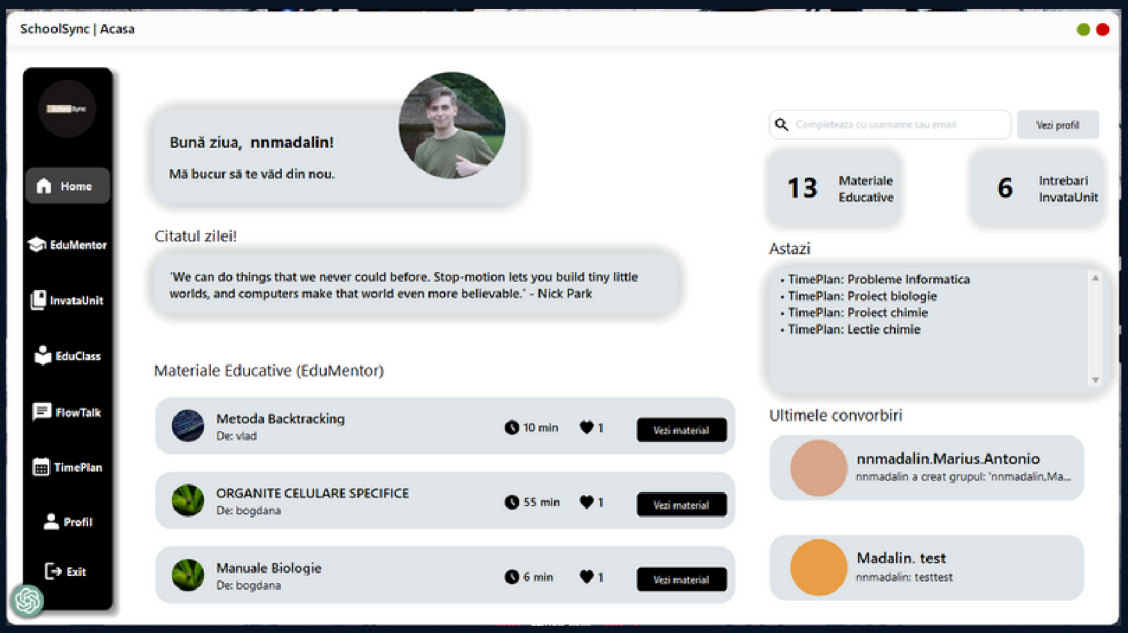
Pagina de înregistrare

În pagina de înregistrare, utilizatorul trebuie să completeze toate casetele. Caseta „**username**" și „**email**" trebuie să fie unice în baza de date. Dacă acestea nu sunt unice, APi-ul va returna un mesaj de eroare. După ce utilizatorul și-a creeat un cont, acesta trebuie să îl verifice (va primi pe email un **link de activare**)



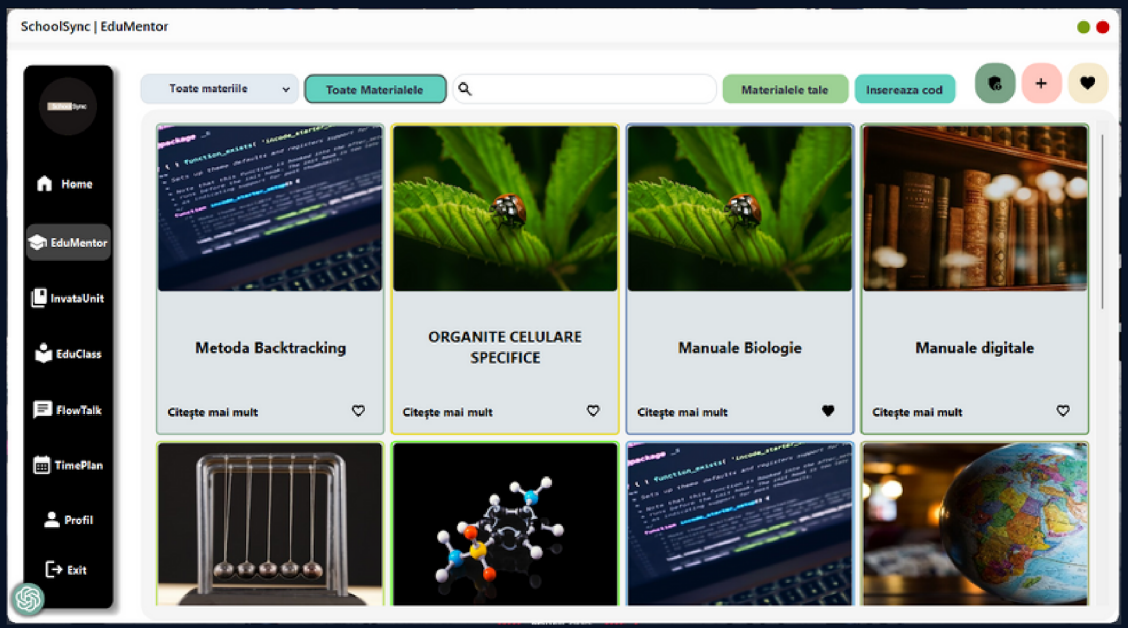
Pagina principală

Conține mai multe informații referitoare la cont, materiale educative, activitățile de astăzi și ultimele convorbiri. Utilizatorul are posibilitatea, folosind caseta din partea dreaptă, să caute un utilizator după **numele de utilizator (username)** sau **email**!



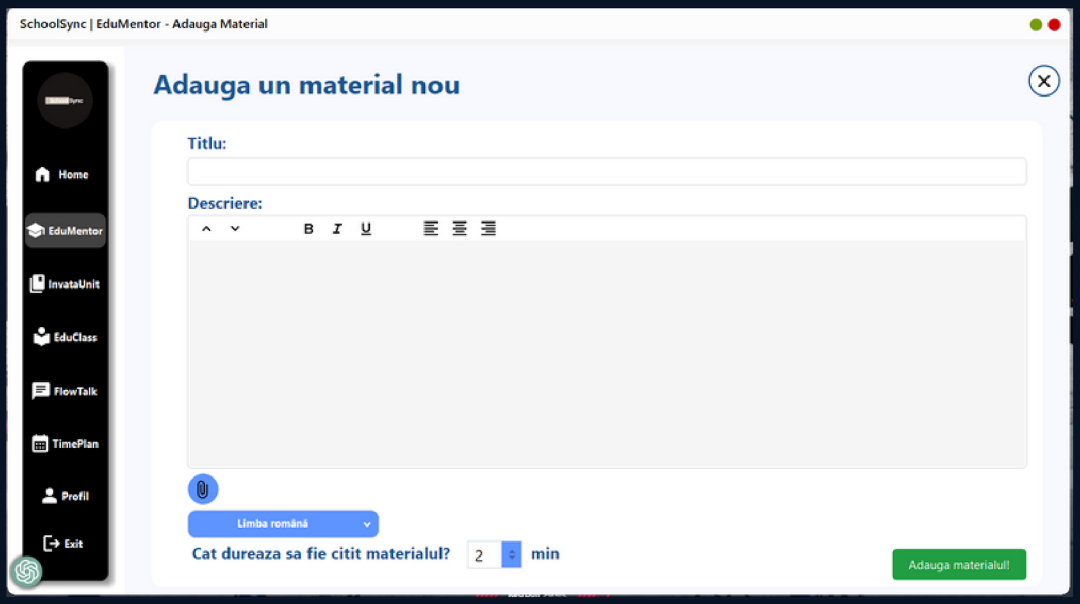
EduMentor

Aici sunt afișate toate **materialele educative** adăugate pe platformă. În partea de sus avem mai multe opțiuni, precum sortarea întrebărilor dupa categorii, posibilitatea de a căuta un material după titlul acestuia, adăugarea de materiale educative, secțiunea administrativă **(vizibilă doar moderatorilor)**, afișarea materialelor adăugate de utilizator și materialele adăugate la favorite!



EduMentor - adaugă material

Aici utilizatorul trebuie să completeze toate casetele, (ex **titlu, descriere, timpul de citire**) . Acesta poate adăuga până la **5 fișiere de maximum 10 MB**. Caseta descriere permite editarea textului (ex **bold, italic, underline** etc)

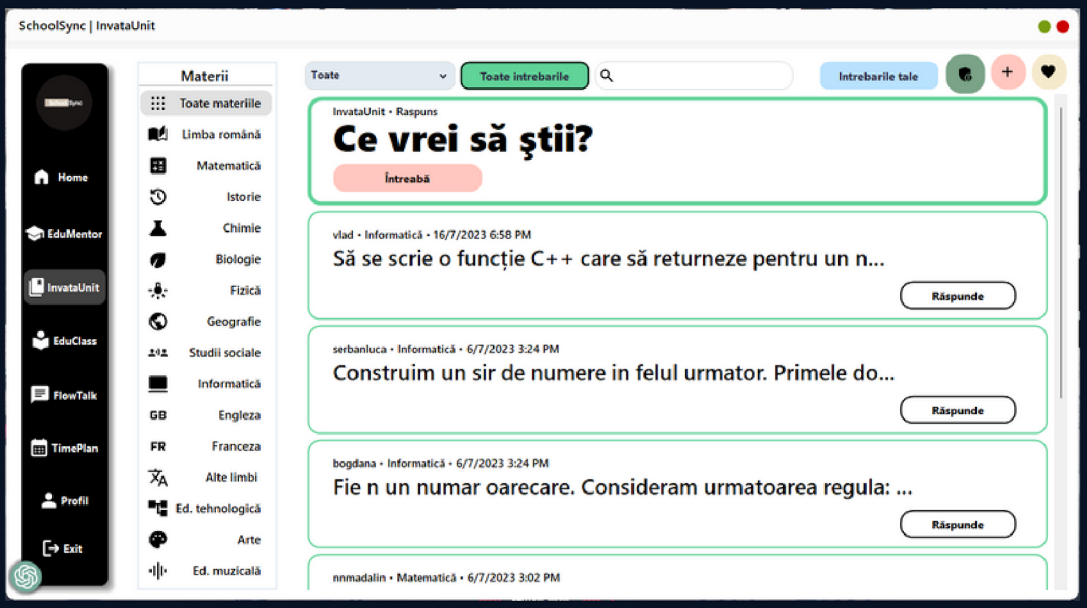


EduMentor - afișarea unui material

Dacă materialul a fost adăugat de utilizator, acesta are opțiunile de a edita, de a șterge materialul și a-l face invizibil (ceilalți utilizatori nu îl vor putea vedea). Toți utilizatori au opțiunea de a copia codul materialului. Dacă acesta nu este invizibil sau dezactivat de administrator, **utilizatori vor putea accesa materialul folosind acel cod**. Moderatorii au un buton numit **"Show/Hide"** care permite dezactivarea materialului. Creatorul este informat că materialul lui este dezactivat!

InvațăUnit

Aici toți utilizatori pot adăuga întrebări despre diferite materii. Aceștia pot sorta întrebările afișate (după materii, după statusul întrebării (**dacă a primit răspuns**)). Moderatorii au un buton pentru a accesa zona de adminitrare a întrebărilor!

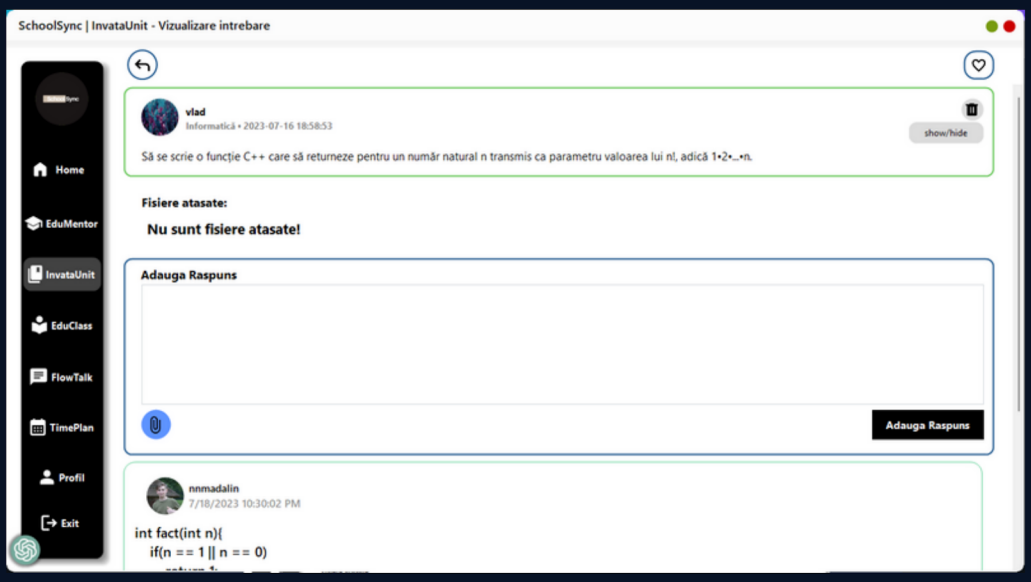


InvațăUnit - adaugă întrebare

Utilizatorii pot adăuga până la 5 fișiere de maximum 5 MB. Caseta întrebare permite editarea textului (ex **bold, italic, underline** etc)

InvațăUnit - vizualizare întrebare

Utilizatorii pot vedea întrebarea pusă, fișierele atașate (dacă acestea există) și pot adăuga ulterior un răspuns (aceștia pot să adauge fișiere). Creatorul poate edita/șterge întrebarea. Atât moderatorul cât și creatorul întrebării pot șterge mesajele utilizatorilor. Moderatorii au acces la butonul **"Show/Hide"** care le permite să dezactiveze întrebarea (creatorul este anunțat că întrebarea acestuia a fost dezactivată)

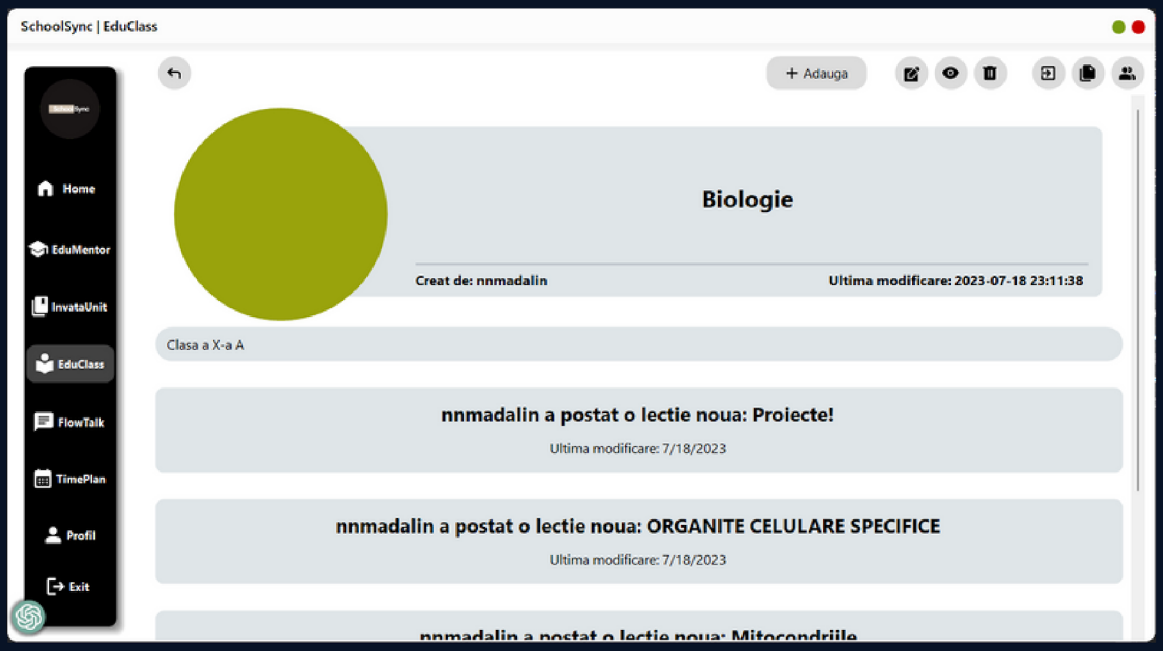


EduClass

EduClass este un sistem inovativ care permite profesorilor să creeze clase virtuale în care se pot adăuga lecții și teme!

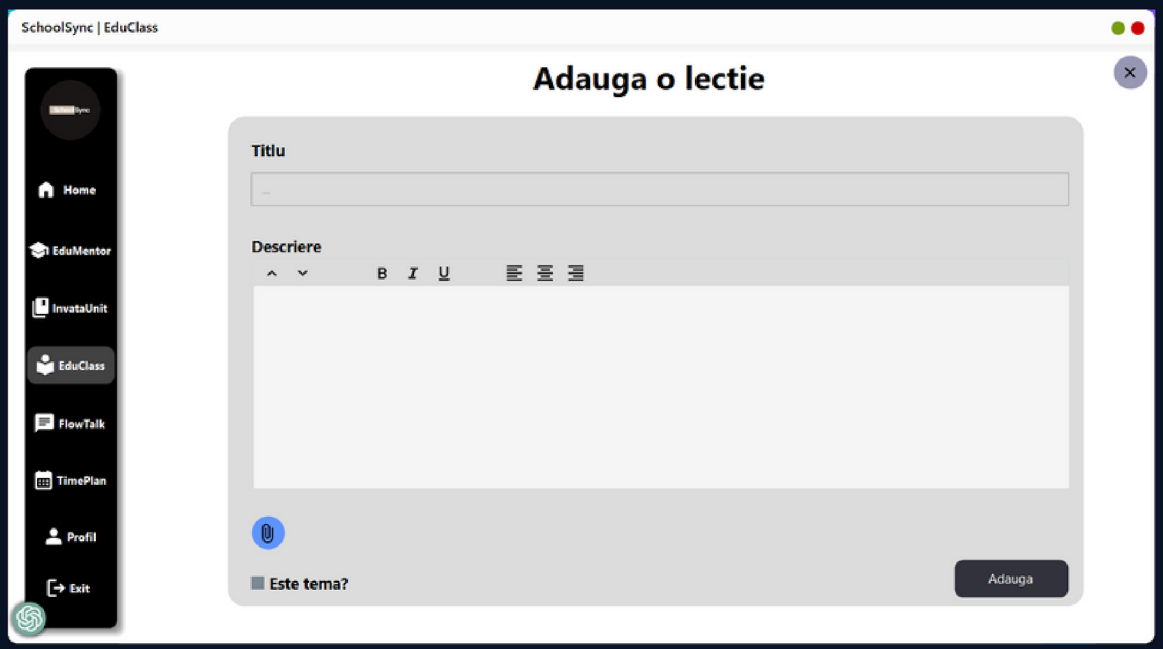
EduClass - vizualizare curs

Elevul poate vedea toate lecțiile/temele postate de profesor. Profesorul are acces la diferite sisteme precum: **să creeze o lecție, să editeze/șteargă cursul și să îl facă invizibil pentru public.**



EduClass - adaugă o lecție

Atunci când profesorii creează o lecție, aceștia au posibilitatea de a încărca până la **5 fișiere de maximum 10 MB**. Dacă profesorul selectează caseta **"Este tema?"**, atunci va trebui să aleagă o dată limită până când elevii vor putea trimite temele!

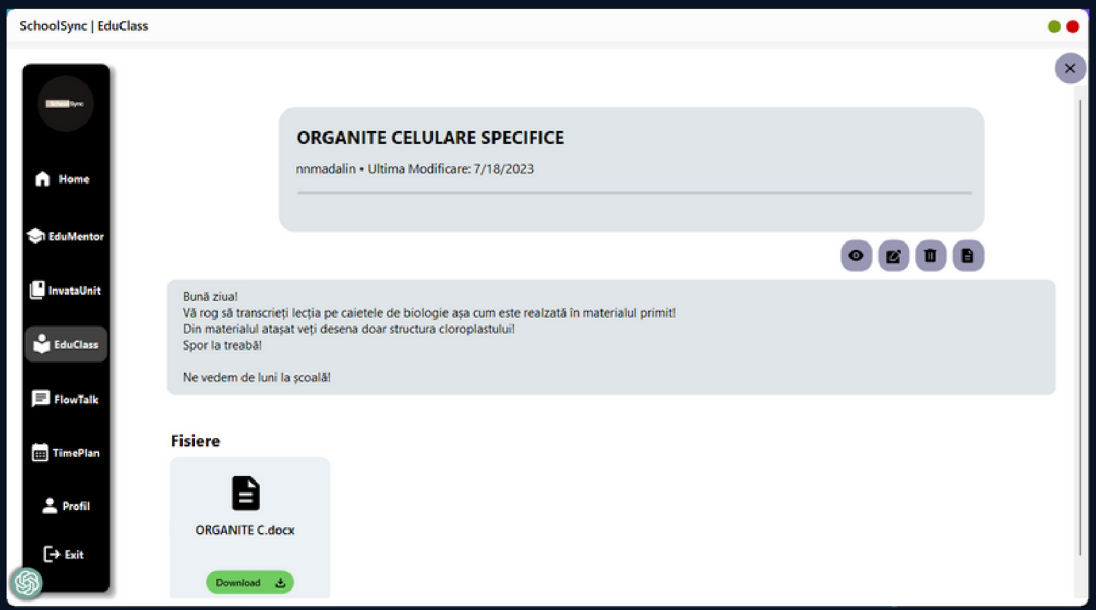


EduClass - vizualizare persoane

Pagina este împărțită în 3 sectiuni: **admini, studenți în așteptare și studenți.**

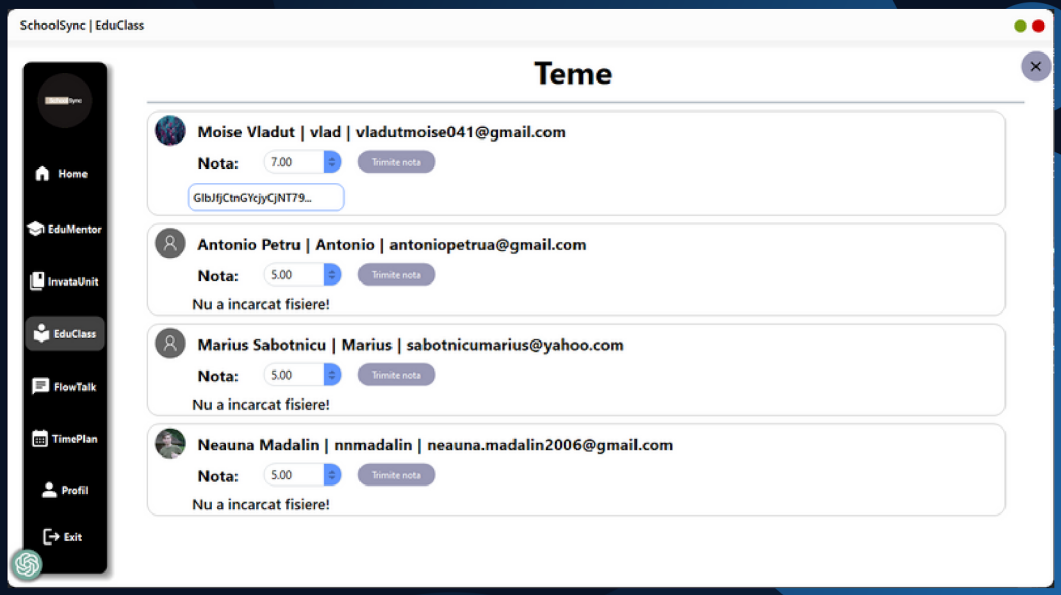
EduClass - vizualizare lectie

Dacă lecției i-a fost atașată o temă, elevilor le vor apărea o secțiune de încărcare a temei. Elevii vor putea încărca până la **5 fișiere de maximum 10 MB** (dacă aceștia au depășit data limită, nu vor mai putea încărca fișiere). **Profesorii vor putea ascunde o lecție, să editeze sau să o șteargă.**



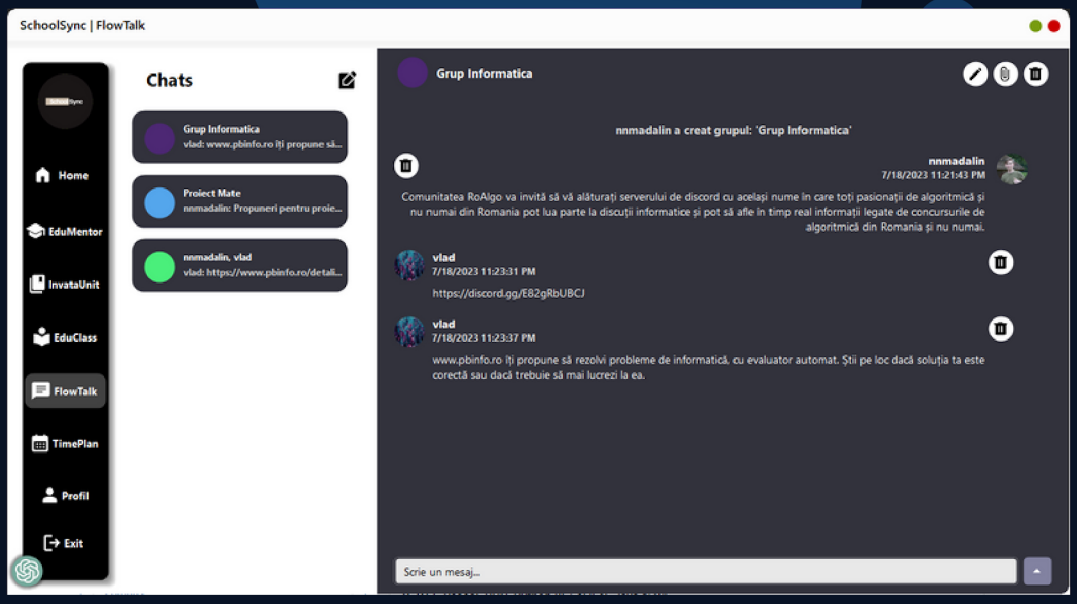
EduClass - vizualizare teme

Pagina "**vizualizare teme**" este accesibilă doar profesorilor. Aceștia pot vizualiza temele încărcate de elevi, ulterior adăugând și o notă.

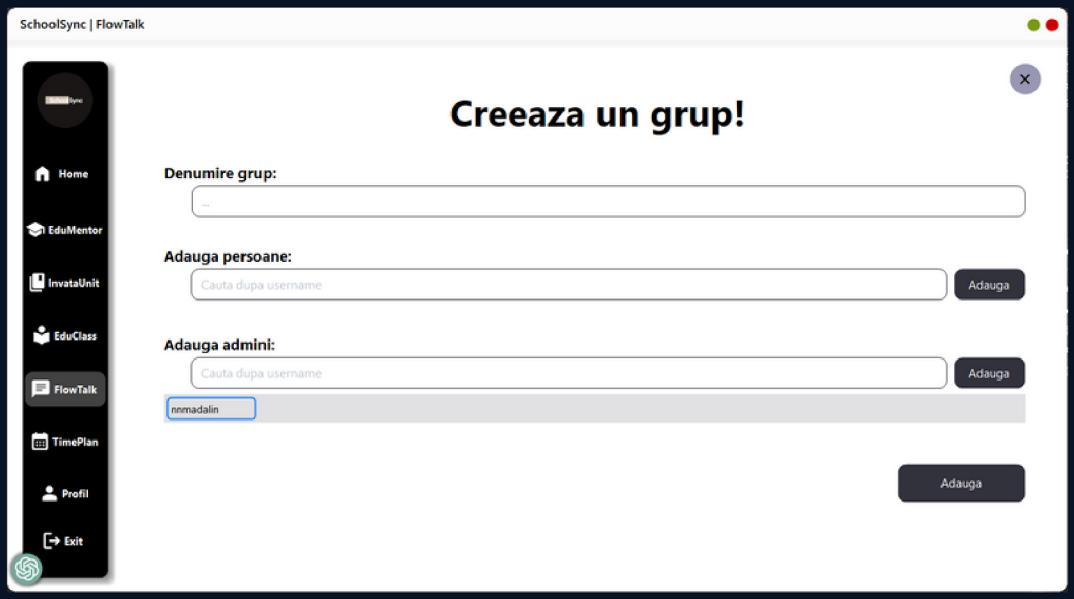


FlowTalk

Cei care fac parte dintr-un grup, **pot încarca fișiere de maximum 5 MB**. Administratorii vor putea șterge mesajele (fiecare utilizator își poate șterge întrebarea lui).

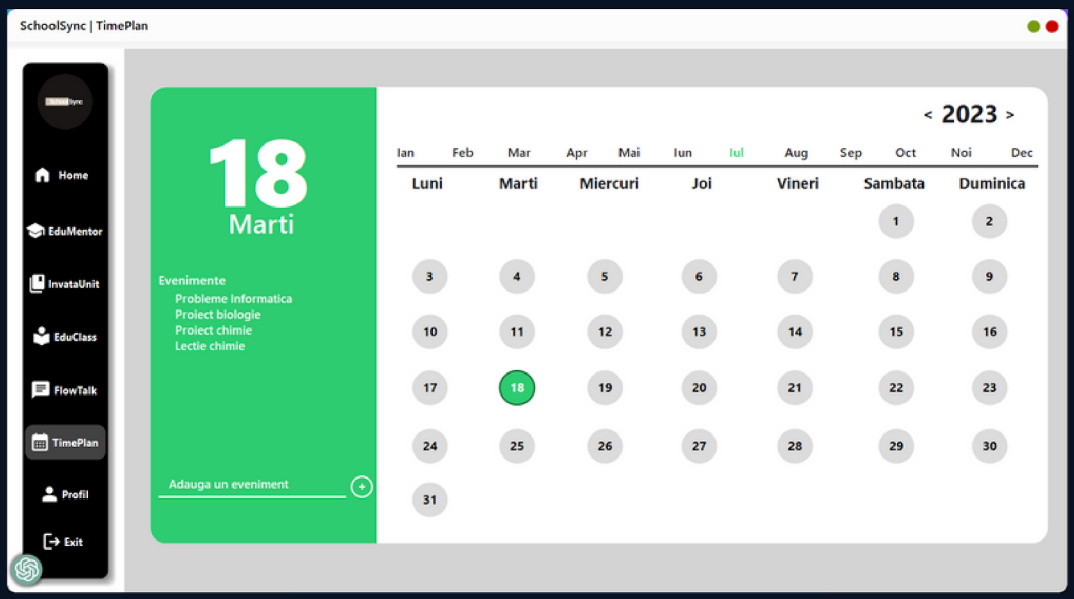


FlowTalk - creează un grup



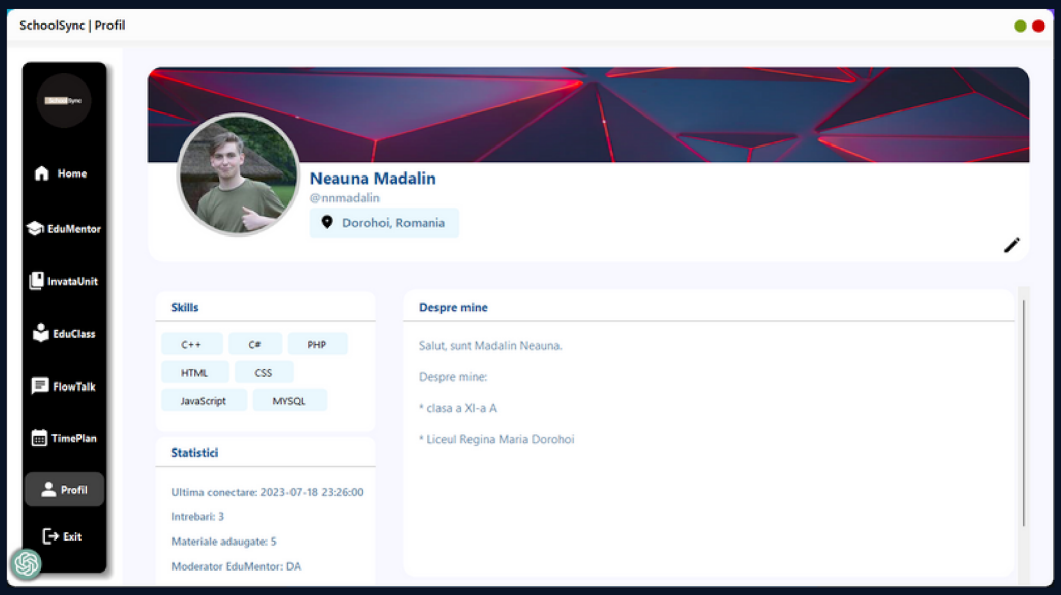
TimePlan

E un modul bine structurat, în care elevii și profesorii vor putea să își gestioneze evenimentele foarte ușor! **Evenimentele adăugate vor apărea și în pagina acasă** (dacă data evenimentului corespunde cu data actuală).

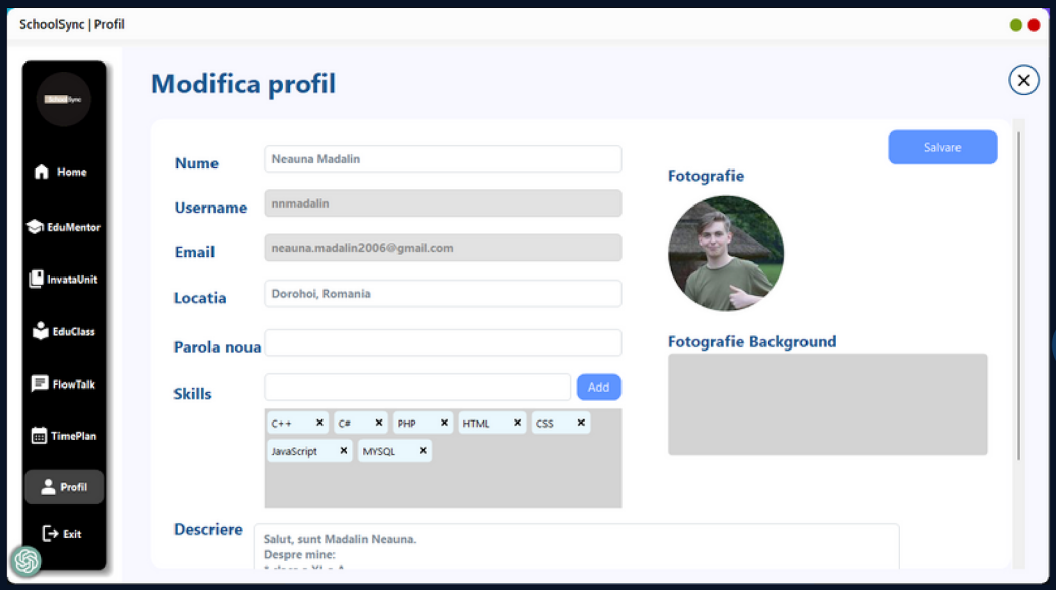


Profil

Administratorii aplicație pot gestiona persoanele care sunt moderatori la modulele **InvataUnit** și **EduMentor**.

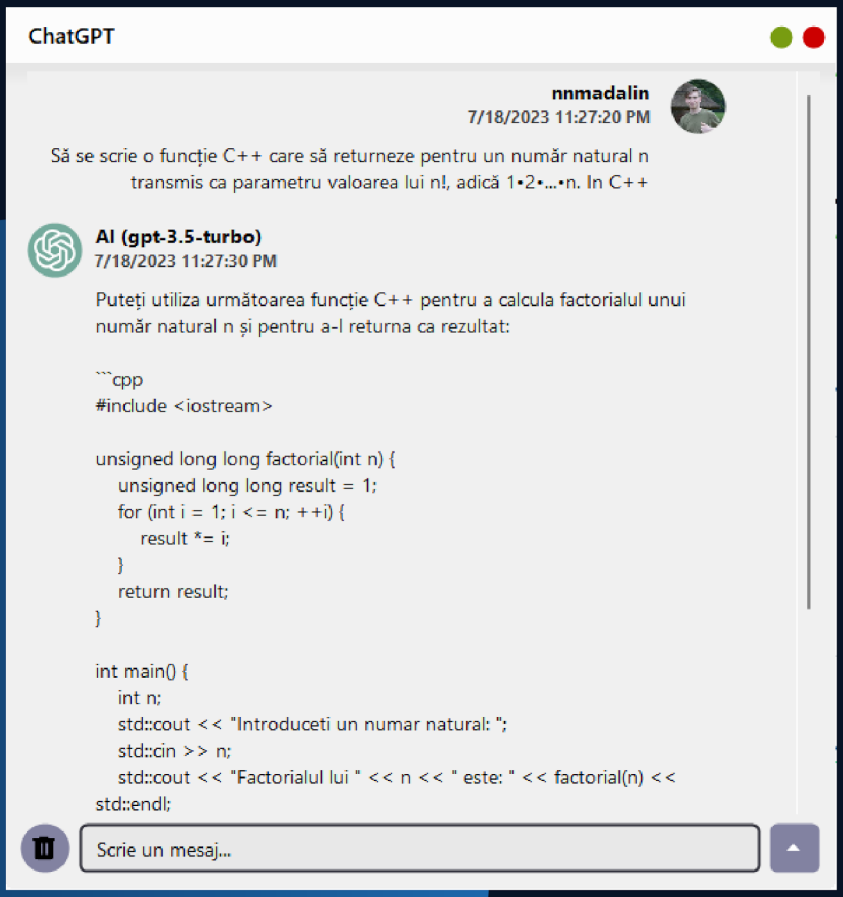


Profil - modifică profil



ChatGPT

Aplicația SchoolSync are implementat un sistem de mesagerie, bazat pe inteligența artificială. Fiecare utilizator poate să **adreseze întrebări robotului!**



5. Resurse

Extensii necesare pentru utilizarea aplicației

- .NET Framework 4.7.2
- GunaUI2
- Newtonsoft.Json

Imagini utilizate în aplicație

- Unsplash
- Google Icons
- Icons8
- Canva

Aplicații folosite

- Visual Studio 2019
- Postman
- PhpMyAdmin
- drawsql.app
- cPanel
- codesnap.dev

Limbaje de programare folosite:

- C#
- HTML
- CSS
- PHP

Citate: api-ninjas.com/api/quotes
Server Status: schoolsync.betteruptime.com