COLEGIUL TEHNIC “SAMUIL ISOPESCU”

PROIECT

Pentru susținerea examenului de certificare a calificării profesionale a

absolvenților învățământului postliceal

Specializare:

Analist programator

Candidat:

Cebotari Vlad

Îndrumător:

Pînzariu Marian

Sesiunea

* Februarie 2025 -

TEMA PROIECTULUI

Plate Snap

Platformă web, rețea socială cu rețete de bucătărie, componenta server.

# 

# ARGUMENT

Această platformă digitală pentru rețete culinare este concepută special pentru pasionații de gătit,ca să-și împărtășească ideile sau să descopere noi rețete inspiraționale.  
 Ideea de a dezvolta această aplicație a apărut deoarece gătitul este o activitate indispensabilă, ritmul alert al vieții moderne face ca multe persoane să întîâmpine dificultăți în a găsi rețete rapide și practice. Scopul nostru este să construim o comunitate activă, unde utilizatorii pot explora idei noi, schimba experiențe și găsi soluții adaptate nevoilor lor, într-un cadru bine organizat.  
 Una dintre principalele provocări pe care această aplicație o rezolvă este lipsa unui acces rapid la rețete simple și bine structurate. De prea multe ori, oamenii petrec mult timp navigând printre site-uri sau aplicații, fără a găsi o sursă de încredere sau un sistem eficient de filtrare. În plus, utilizatorii își doresc să fie siguri de calitatea rețetelor înainte de a le încerca, bazându-se pe opiniile și experiențele altor persoane.  
 Prin această platformă, ne propunem să sprijinim reducerea risipei alimentare prin sugestii bazate pe ingrediente disponibile,vom promova gătitul acasă și vom oferi un spațiu prietenos atât pentru cei care sunt la începători, cât și pentru cei experimentați.

..

**CUPRINS**

[ARGUMENT](#_gjdgxs) 3

[Capitolul 1 - Descrierea aplicației](#_30j0zll) 6

[1.1 Misiunea PlateSnap.](#_1fob9te) 6

[1.2 Impactul PlateSnap](#_3znysh7) 7

[Capitolul 2 – Cerintele aplicație](#_2et92p0) 8

[2.1 Cerințe funcționale principale(core features)](#_tyjcwt) 8

[2.2 Cerințe non-funcționale](#_3dy6vkm) 9

[Capitolul 3 - Interfața cu utilizatorul](#_1t3h5sf) 10

[Capitolul 4 - Instrumente utilizate 1](#_4d34og8)3

[Capitolul 5 - Structura aplicației 1](#_2s8eyo1)4

[5.1 Separarea pe compartimente 1](#_17dp8vu)4

[5.2 Structura bazei de date 1](#_3rdcrjn)5

[5.3 API Routes 1](#_26in1rg)6

[5.4 Structura Git 1](#_lnxbz9)7

[Capitol 6 – Organizarea echipei](#_35nkun2) 20

[6.1 Membrii echipei și rolurile lor 20](#_1ksv4uv)

[Capitolul 7 – Dezvoltarea aplicației](#_44sinio) 21

7.1Cum s-a realizat proiectarea bazei de date 21  
7.2Cum s-a realizat un design la API-urile REST 21  
7.3Cum a fost testat API-ul folosind Postman 21  
7.4Cum s-a utilizat Git pentru a versiona codul sursă 21  
7.5Contribuția proprie în fiecare etapă 21  
7.6Utilizarea platformei Azure DevOps pentru organizarea sarcinilor 21  
7.7Descrierea/lista cu sarcinile proprii 22

[Capitolul 8 - Contribuția proprie](#_2jxsxqh) 23

8.1 Structura proiectului 23

8.3 Baza de date 27

8.4 API, Postman27

[Capitolul 9 – Configurarea și rularea aplicației](#_z337ya) 28

[Capitolul 10 - Ghid de utilizare](#_3j2qqm3) 30

[10.1 Authentificare și înregistarare](#_1y810tw) 31

[10.2 Gestionarea profilului](#_4i7ojhp) 32

[10.3 Vizualizarea și căutarea rețetelor](#_2xcytpi) 33

[10.4 Adaugarea unei rețete noi](#_3whwml4) 34

[Capitolul 11 – Concluzii/ Lecții învățate](#_2bn6wsx) 35

[11.1](#_147n2zr)  [Lecții învățate](#_3o7alnk)

[ANEXE](#_qsh70q) 36

[BIBLIOGRAFIE](#_3as4poj)

# *Capitolul 1 - Descrierea aplicației*

**PlateSnap** este o platformă web modernă care funcționează ca o rețea socială dedicată pasionaților de gătit și bucătărie. Scopul său principal este de a conecta utilizatorii prin intermediul rețetelor culinare, oferindu-le posibilitatea de a-și împărtăși creațiile, de a descoperi rețete noi și de a interacționa cu alți utilizatori prin funcții sociale.

Aplicația oferă o experiență interactivă și prietenoasă (user-friendly), adresându-se atât bucătarilor profesioniști, cât și amatorilor care doresc să exploreze noi rețete sau să-și perfecționeze abilitățile culinare.

Designul minimalist si intuitiv constituie un factor important pentru construirea unei experiente fluide si favorabile pentru utilizator.

***Ideea de la care am pornit*** si care sta la baza aplicatiei este cea a crearii unui spatiu web confortabil si relaxant, destinat iubitorilor de gastronomie.

***Cui se poate adresa platforma PlateSnap?***

Publicul tinta este construit conform legislatiei GDPR (în Uniunea Europeană). Reglementările prevăd un prag minim de 16 ani pentru colectarea datelor personale, dacă nu există consimțământ parental explicit.

Astfel, accesul in platforma dezvoltata de noi este permis adolescentilor de peste 16 ani, neexistand un prag de varsta superior impus.

## 1.1 Misiunea PlateSnap.

***Care este misiunea PlateSnap?***

**Inspirația culinară** se datoreaza faptului că utilizatorii au acces la o colecție vastă și diversificată de rețete, fiind postate de utilizatori din arii geografice diverse. Platforma este un spațiu unde pot descoperi preparate inedite, pot învăța noi tehnici și pot explora bucătării internaționale.

O comunitate activă și prietenoasă prin funcțiile de socializare, cum ar fi like-urile sau funcția de follow. PlateSnap facilitează conexiunile între utilizatori. Aceștia pot primi feedback constructiv prin aprecieri, pot interacționa cu alți pasionați de gătit și pot forma conexiuni bazate pe interese comune.

Un spațiu pentru creativitate. PlateSnap încurajează utilizatorii să fie creativi și să-și exprime unicitatea prin rețetele proprii. Fiecare rețetă publicată este unica, însoțită de imagini și descrieri care reflectă stilul și personalitatea utilizatorului.

Educație și perfecționare. Pentru utilizatorii care doresc să învețe mai mult, platforma oferă postari cu retete ce contin tehnici culinare, utilizarea corectă a ingredientelor și tendințele actuale din gastronomie.

## 1.2 Impactul PlateSnap

Prin modul in care ne raportam la utilizator, creandu-i o experienta placuta si prietenoasa, ne dorim sa cream o comunitate cu aceleasi interese gastronomice, sa sporim interactiunea si incluziunea sociala.

# *Capitolul 2 – Cerintele aplicație*

## 2.1 Cerințe funcționale principale(core features)

Printre principalele cerinte a fost crearea unei experiente interactive si unice pentru utilizator. *PlateSnap* are la baza o interfata intuitiva, cu un aspect usor de vizualizat.

**Crearea și gestionarea conturilor utilizatorilor**

* Pagina de Bun-Venit/ *HomePage* ( constuituie un element care are ca scop imbunatatirea interfetei utilizatorului si a experientei acestuia)
* Înregistrare (cu campurile: nume, prenume, username, email, parola, confirmare parola numar de telefon). Aceasta cerinta se afla integrata in pagina de *SignUp* a platformei.
* Autentificare (cu campurile: username si parola). Aceasta componenta se afla in pagina numita *SignIn.*
* Posibilitatea de a edita profilul (cu campurile: nume, prenume, email, parola, username, imagine de profil, numar de telefon), integrata in pagina de tip *modal* in cadrul paginii *Profile.*

**Funcționalități sociale**

* Opțiunea de a aprecia (like) rețetele altor utilizatori. Aceasta functionalitate se regaseste in pagina HomePage.
* Posibilitatea de a urmări alți utilizatori(follow), ce se poate regasi in pagina HomePage, respectiv Profile.

**Publicarea/ afisarea rețetelor**

* Adăugarea unei rețete noi cu (titlu, descriere, ingrediente, pași de preparare și imagini) este afisata sub forma butonului *“Add recipe”* in paginile *HomePage,* respectiv *Profile.*

Astfel, userul poate sa adauge o reteta accesand butonul precizat din fiecare pagina, dupa ce trece de procesul de autentificare.

**Căutarea și descoperirea rețetelor**

* Sistem de căutare pe baza cuvintelor-cheie ( nume, ingrediente) cu ajutorul componentei “Search” din pagina HomePage.
* Recomandări de postari pe baza continutului adaugat de alti utilizatori in pagina Home.

## 2.2 Cerințe non-funcționale

Cerintele non-functionale ale platformei sunt acele cerinte care definesc caracteristici generale ce nu sunt legate de o funcție specifică, dar influențează calitatea sa.

**Designul general al aplicației**

* Interfața a fost conceputa să fie prietenoasă și intuitivă, punând accent pe simplitate și estetică.
* Utilizarea unui design minimalist, cu culori calde și atrăgătoare, potrivite tematicii culinare.
* Structură clară, cu secțiuni bine definite: profil utilizator, rețete, explorare(HomePage).

Componenta vizuală

Galerie de imagini pentru fiecare rețetă.

Iconițe tematice (de exemplu: simboluri pentru ingrediente sau tipuri de preparate).

**Feedback-ul utilizatorului**

* Mesaje și notificări clare care confirmă acțiunile utilizatorilor (ex. „Rețeta a fost publicată cu succes”; ,,Parola a fost schimbata cu succes” etc).
* Afișarea like-urilor , a numarului de utilizatori care urmaresc contul utilizatorului(followers) a numarului de persoane urmarite (following) și altor interacțiuni în timp real.

# *Capitolul 3 - Interfața cu utilizatorul*

Interfata utilizatorului a fost proiectata cu ajutorul instrumentului ***Figma.***Aceastaplatformaa constituit punctul de plecare a tuturor ideilor si a reusit sa cuprinda in mod sistematic designul actual al aplicatiei ***PlateSnap.***

Paralela din punct de vedere estetic intre ***Figma*** si aplicatia noastra a fost implementata majoritar in fisiere de tip #css.

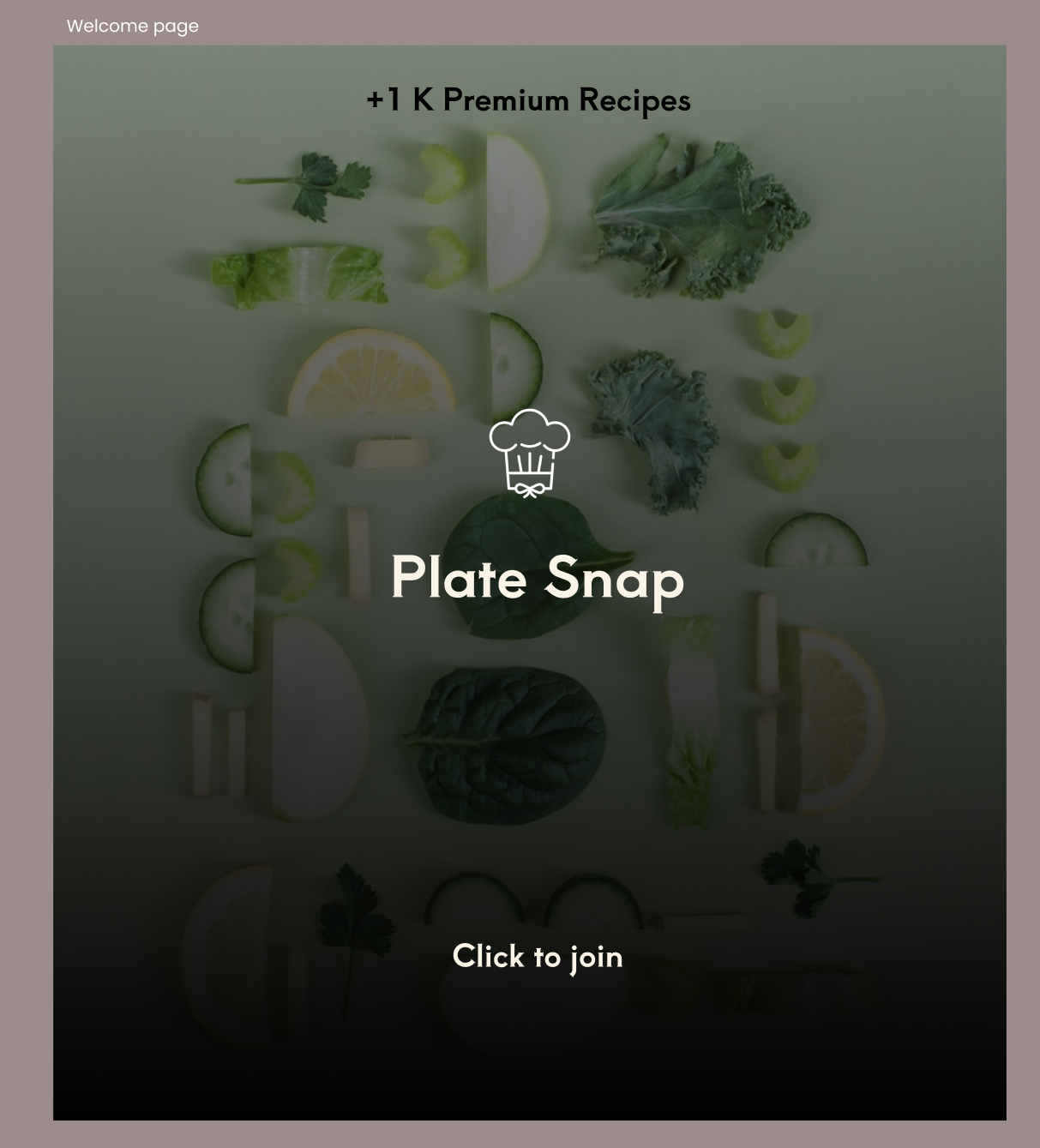
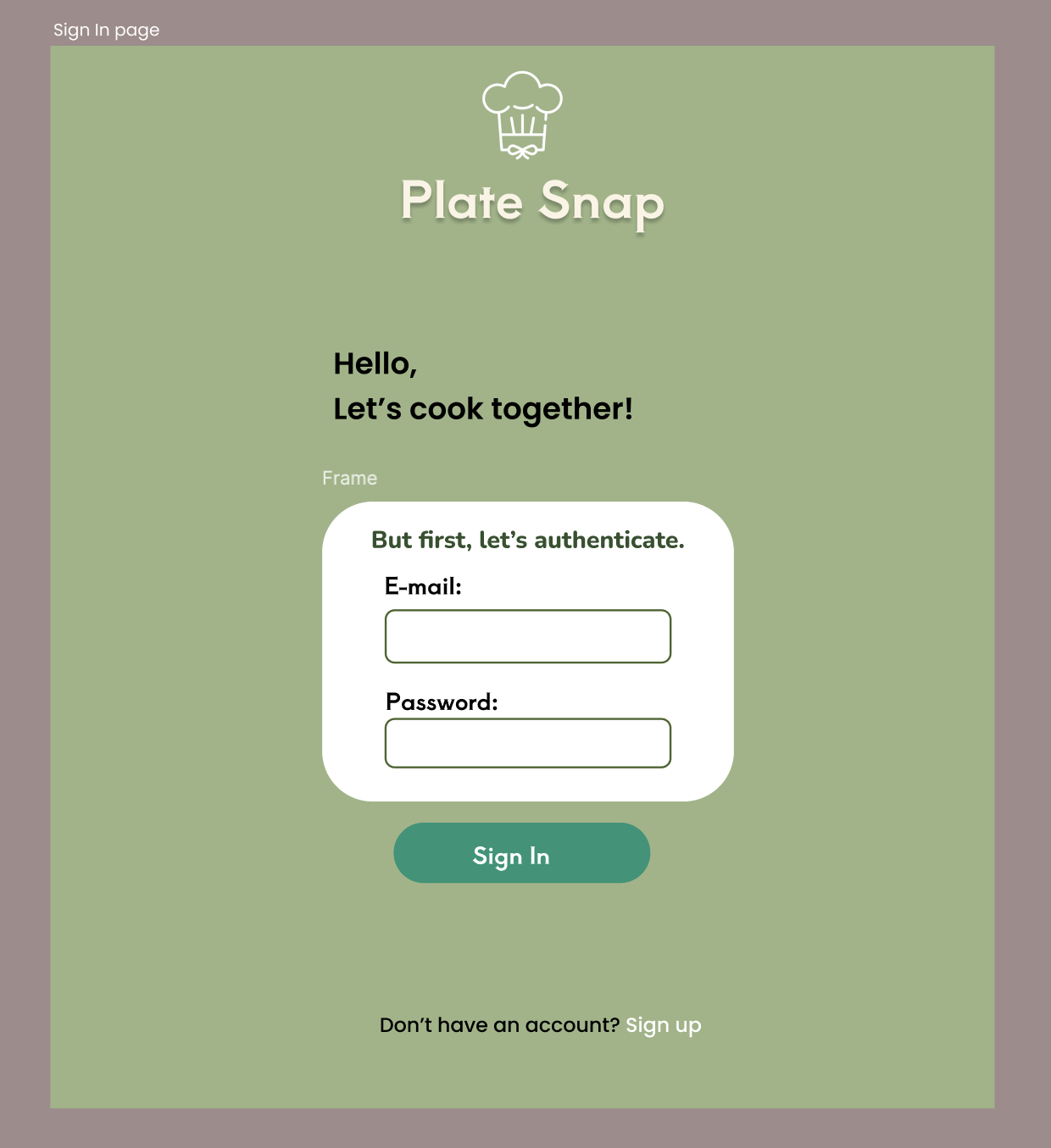
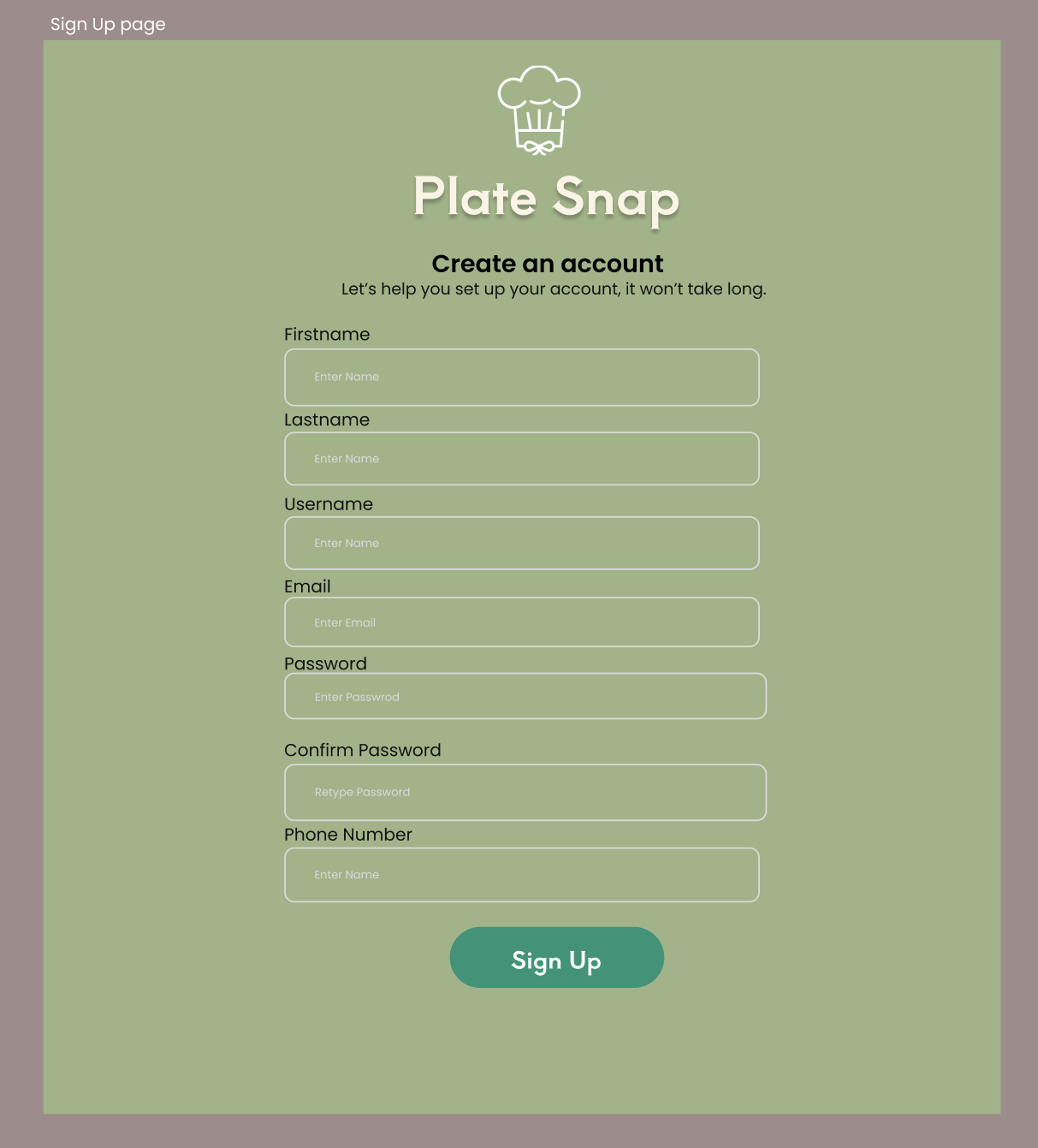
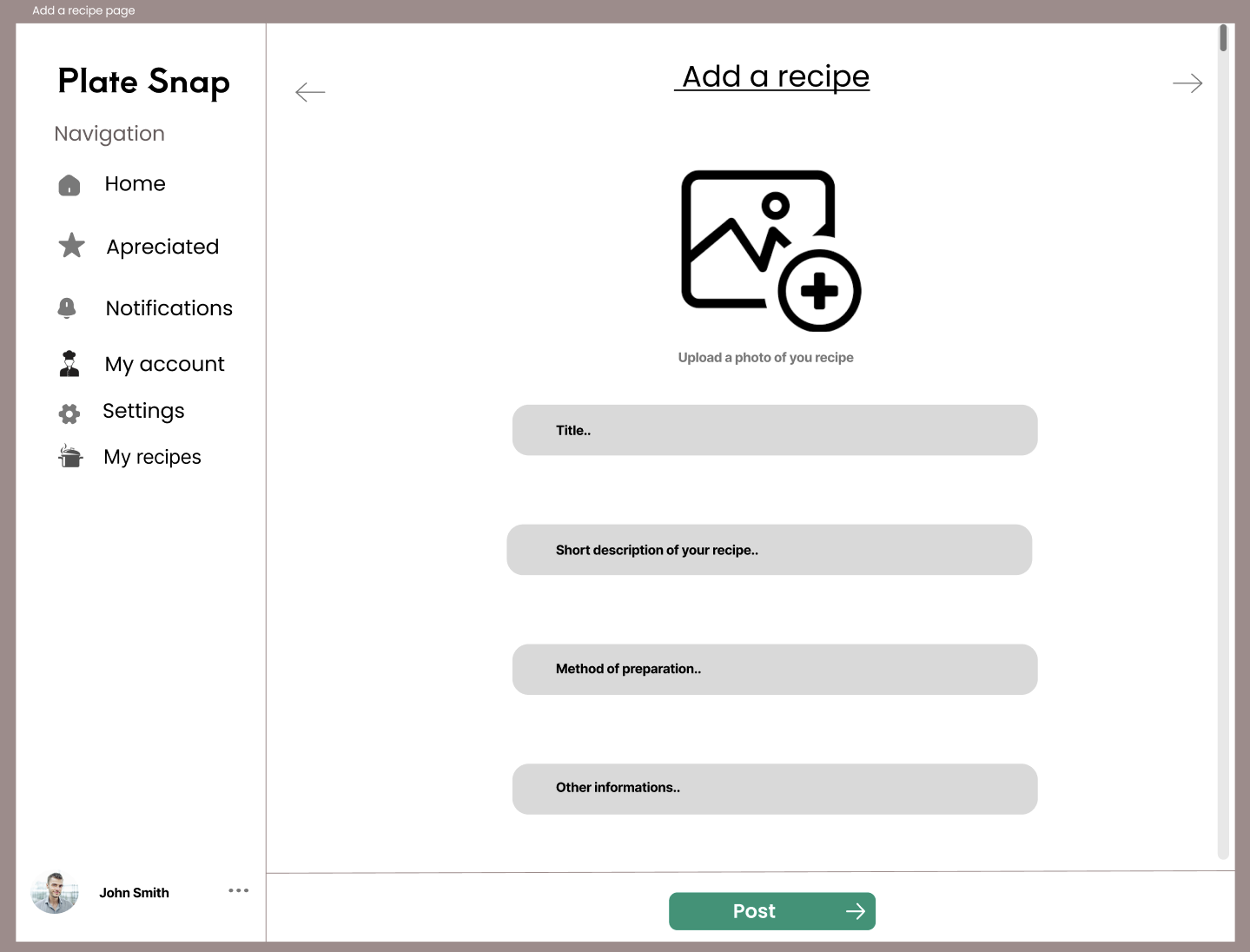
UI-ul (user interface-ul) aplicatiei a fost gandit si implementat astfel incat sa fie user-friendly, dupa cum am mentionat in capitolele anterioare, dar si intuitiv, astfel încât utilizatorii să poată înțelege rapid cum să navigheze și să utilizeze funcționalitățile aplicației, fără să fie nevoie de instruire sau tutoriale complexe.

In plus, platforma contine elemente interactive, cum ar fi animații deasupra butoanelor, sectiune de testimoniale , feedback vizual la apăsarea butoanelor de save, change și efecte dinamice care oferă o experiență mai captivantă.

Elementele UI sunt consistente în întregul design, astfel încât utilizatorii să recunoască rapid funcționalitățile (de exemplu, toate butoanele principale au același stil).

Interfața PlateSnap este în pas cu tendințele actuale de design, utilizând elemente vizuale plăcute, animații subtile și culori care atrag privirea fără a fi obositoare.

De exemplu, paginile de login/autentificare(WelcomePage, SignUp, SignIn) surprind cel mai bine caracterul minimalist al interfetei utilizatorului:

# *Capitolul 4 - Instrumente utilizate*

**Back-end:**

1. **PostgreSQL**: Baza de date folosită, inițial utilizată local și implementată ulterior pe *Supabase* pentruaadministrareașiahostingacloud.  
2.**SQLAlchemy:ORM** utilizat pentru conectarea și interacțiunea cu PostgreSQL.  
3. **FastAPI**: Framework utilizat pentru dezvoltarea aplicației pe back-end,*Uvicorn* fiind folosit caaserveruladeadezvoltare.  
4.**API:**  
 ○SchemaaaafostacreatăașiasimulatăacuaMockoon.  
 ○ImplementarearealizatăaînaFastAPI.  
 ○Testarea a fost efectuată folosind Swagger UI și Postman pentru verificarea funcționalitățiiaAPI.  
5.**GitHub:**Repositoriuautilizatapentruagestionareaacodului.  
6.**Git:**Utilizatapentruaversionacontrol.  
7.**PyCharm:***IDE*aprincipalautilizatapentruadezvoltareaaaplicațieiapeaback-end.  
8. **Koyeb**: Platformă utilizată pentru implementarea aplicației în mediu de producție.

**Front-end:**

● **Node.js:** Utilizat pentru managementul aplicației.  
● **React:** Framework utilizat pentru dezvoltarea UI.  
● **Figma:** Instrument utilizat pentru proiectarea designului aplicației.  
● **Git și GitHub:** Pentru version control și colaborarea pe partea de front-end.  
● **Visual Studio Code** (VS Code): Code editor utilizat pentru dezvoltarea aplicației front-end.  
Instrumente utilizate pentru managementul proiectului și colaborare în echipă: Microsoft Azure,Slack și Discord

# *Capitolul 5 - Structura aplicației*

## 5.1 Separarea pe compartimente

● API  
○ Locație: Aplicația back-end este dezvoltată folosind FastAPI și este deployed pe Koyeb.  
○ Comunicare: Primește cereri REST de la front-end și trimite răspunsuri JSON.

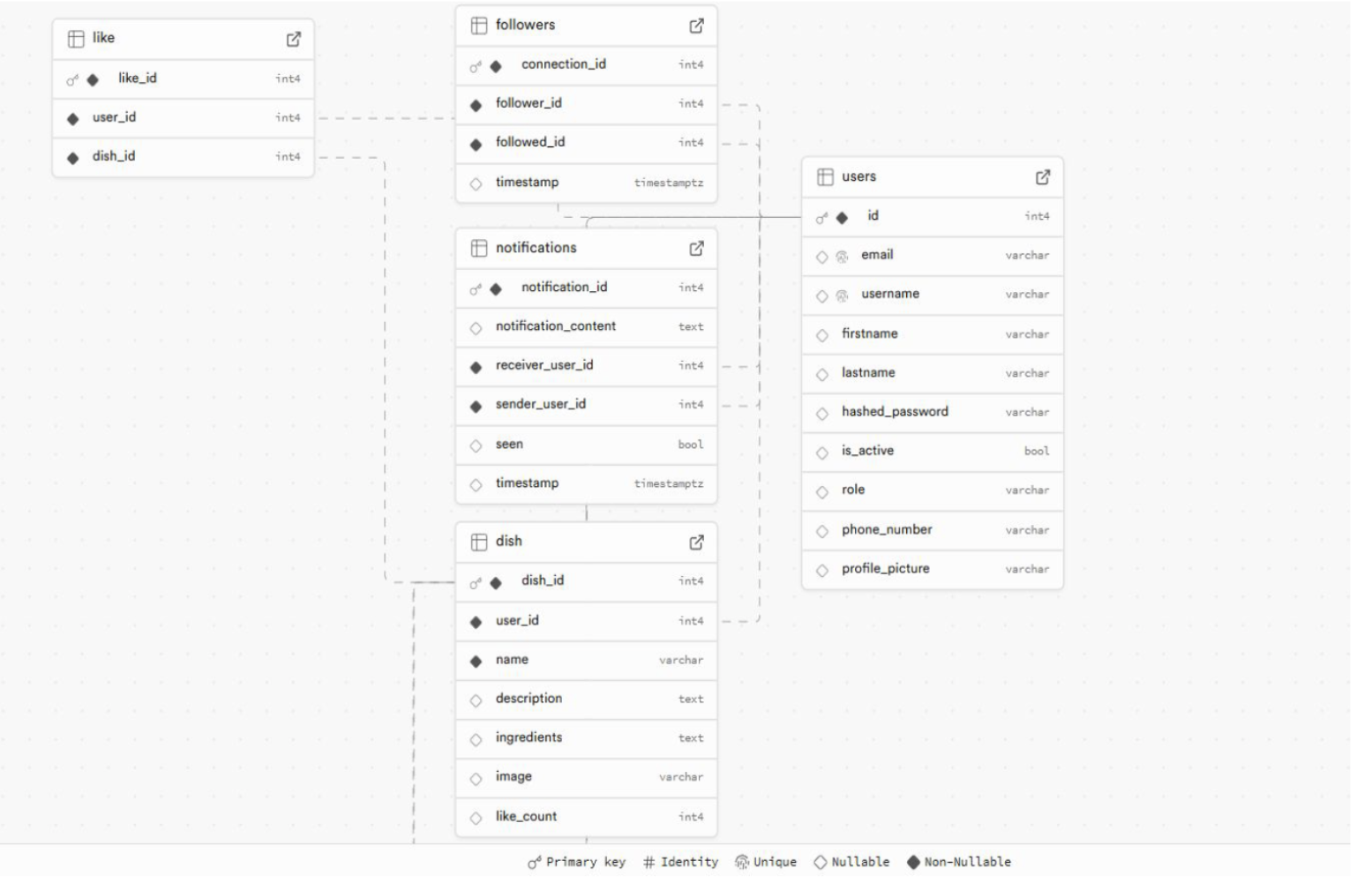
Conectarea la baza de date Supabase pentru gestionarea datelor utilizatorilor, postărilor, notificărilor etc.

● Baza de Date (Supabase)  
 ○ Locație: Baza de date este deployed pe Supabase, care utilizează PostgreSQL.

**Conexiune**:  
Modelele implementate cu SQLAlchemy, interacționează cu baza de date pentru a executa interogări și operațiuni CRUD.  
Supabase gestionează conexiunea securizată printr-un URL de bază de date configurat în aplicația back-end.

**Front-end**   
 Locație: Aplicația front-end este locală, React ca framework de bază.  
 Funcționalitate:  
 ■ Trimite cereri către API pentru autentificare, gestionarea postărilor,   
vizualizarea notificărilor și alte funcționalități.  
 ■ Primește răspunsuri JSON de la API și afișează informațiile   
utilizatorului în UI.  
○ Conexiune:  
 ■ Trimite cereri HTTPS către API-ul deployat pe Koyeb   
 (<https://elegant-kimbra-cooking-app-89b03224.koyeb.app/>).  
 ■ Afișează datele obținute din baza de date prin interacțiunea cu API-ul.

## 5.2 Structura bazei de date



**Users:**

* **id** (int4, primary key): Identificator unic al utilizatorului.
* **email** (varchar): Adresa de email a utilizatorului.
* **username** (varchar): Numele de utilizator.
* **firstname** (varchar): Prenumele utilizatorului.
* **lastname** (varchar): Numele de familie al utilizatorului.
* **hashed\_password** (varchar): Parola utilizatorului, stocată în formă hash.
* **is\_active** (bool): Starea activă a contului.
* **role** (varchar): Rolul utilizatorului (ex: admin, utilizator normal).
* **phone\_number** (varchar): Numărul de telefon al utilizatorului.
* **profile\_picture** (varchar): URL către poza de profil.

**Followers:**

* **connection\_id** (int4, primary key): Identificator unic al conexiunii.
* **follower\_id** (int4, foreign key către users.id): ID-ul utilizatorului care urmărește.
* **followed\_id** (int4, foreign key către users.id): ID-ul utilizatorului care este urmărit.
* **timestamp** (timestamptz): Data și ora la care a fost creată conexiunea.

**Notifications:**

* **notification\_id** (int4, primary key): Identificator unic al notificării.
* **notification\_content** (text): Conținutul notificării.
* **receiver\_user\_id** (int4, foreign key către users.id): ID-ul utilizatorului care primește notificarea.
* **sender\_user\_id** (int4, foreign key către users.id): ID-ul utilizatorului care trimite notificarea.
* **seen** (bool): Starea de vizualizare a notificării.
* **timestamp** (timestamptz): Data și ora notificării.

**Dish:**

* **dish\_id** (int4, primary key): Identificator unic al preparatului.
* **user\_id** (int4, foreign key către users.id): ID-ul utilizatorului care a creat preparatul.
* **name** (varchar): Numele preparatului.
* **description** (text): Descrierea preparatului.
* **ingredients** (text): Ingredientele preparatului.
* **image** (varchar): URL către imaginea preparatului.
* **like\_count** (int4): Numărul de aprecieri ale preparatului.

**Like:**

* **like\_id** (int4, primary key): Identificator unic al aprecierii.
* **user\_id** (int4, foreign key către users.id): ID-ul utilizatorului care a dat like.
* **dish\_id** (int4, foreign key către dish.dish\_id): ID-ul preparatului care a primit like.

## 5.3 API Routes

**APIAuth**

* **/auth/signup** - Creează un user nou.
* **/auth/token** - Creează un token JWT și autorizează userul în aplicație.

**Users**

* **GET /users/get\_user**: Returnează informațiile unui utilizator.
* **PUT /users/change\_email**: Actualizează adresa de email.
* **PUT /users/change\_firstname**: Actualizează prenumele utilizatorului.
* **PUT /users/change\_lastname**: Actualizează numele de familie al utilizatorului.
* **PUT /users/change\_username**: Actualizează numele de utilizator.
* **PUT /users/change\_password**: Schimbă parola utilizatorului.
* **PUT /users/update\_phone\_number**: Actualizează numărul de telefon.

**Account**

* **GET /account/get\_all\_posts**: Returnează toate postările utilizatorului.
* **GET /account/get\_profile\_picture**: Returnează poza de profil a utilizatorului.
* **POST /account/add\_profile\_picture**: Adaugă o poză de profil.
* **GET /account/get\_notifications**: Returnează notificările utilizatorului.

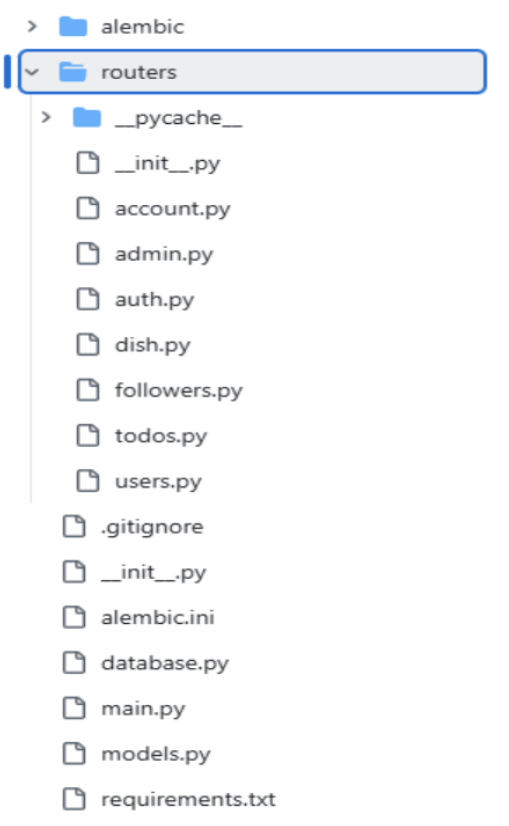
**Dish**

* **POST /dish/dishes**: Creează o nouă postare de preparat.
* **GET /dish/get\_all**: Returnează toate postările de preparate.
* **GET /dish/search**: Caută preparate după anumite criterii.
* **POST /dish/dishes/{dish\_id}/like**: Adaugă un "like" pentru un preparat.
* **POST /dish/dishes/{dish\_id}/unlike**: Elimină un "like" de la un preparat.

**Followers**

* **POST /followers/follow/{username}**: Urmărește un utilizator.
* **DELETE /followers/unfollow/{username}**: Oprește urmărirea unui utilizator.
* **GET /followers/get\_followers**: Returnează lista utilizatorilor care urmăresc utilizatorul curent.
* **GET /followers/get\_followings**: Returnează lista utilizatorilor pe care îi urmărește utilizatorul curent.

## 5.4 Structura Git



**Legendă:**

alembic/ # Migrarea bazei de date  
 routers/ # Definirea rutelor API  
│ account.py # Enpointuri pentru gestionarea contului  
│ admin.py # Enpointuri pentru funcționalități de administrare  
│ auth.py # Enpointuri pentru autentificare  
│ dish.py # Enpointuri pentru postări  
│ followers.py # Enpointuri pentru a da follow  
│ todos.py # Enpointuri pentru gestionarea unei liste "To-Do"  
│ users.py # Enpointuri pentru operațiuni legate de schimbarea datelor contului  
.gitignore # Ignorarea fișierelor inutile  
database.py # Configurație pentru conectarea la baza de date  
main.py  
models.py # Definirea modelelor din baza de date  
requirements.txt # Listează dependințele necesare proiectului

Structura frontend-ului aplicației a fost organizată pentru a asigura claritatea și separarea responsabilităților fiecărui element.

În cadrul directorului **src**, care conține întreg codul sursă al aplicației, structura este împărțită pe module distincte, fiecare având un rol bine definit:

1. Resurse media și design
   1. assets: Include toate resursele media utilizate în aplicație, cum ar fi imagini, logo-uri sau alte fișiere grafice.
   2. fonts: Conține fonturile personalizate utilizate în proiect, pentru a menține un stil vizual consistent.
2. Componente React
   1. Pagini principale: Fiecare pagină principală a aplicației este reprezentată printr-un fișier individual .js (de exemplu, HomePage.js). Aceste fișiere definesc structura generală și logica fiecărui ecran din aplicație.
   2. Componente individuale: Elementele mai mici și reutilizabile ale interfeței sunt implementate separat (de exemplu, Modal.js),

pentru a spori modularitatea codului.

1. Fișiere CSS
   1. Fiecare fișier de stil .css este organizat logic împreună cu componenta sau pagina pe care o stilizează. De exemplu, fișierul homepage.css este utilizat pentru definirea stilurilor specifice componentei HomePage.js.

Exemple de fișiere CSS: styles.css, Homepage.css, SignIn.css, SignUp.css, Testimonials.css, Profile.css etc.

# *Capitol 6 – Organizarea echipei*

În dezvoltarea aplicației, echipa noastră a fost formată din 3 membri, fiecare având roluri bine definite, astfel încât să asigure o dezvoltare eficientă a proiectului.

## 6.1 Membrii echipei și rolurile lor

1. *Cebotari Vlad* – Backend Developer

• Responsabil de dezvoltare părții de backend a aplicației.

• A asigurat funcționalitatea bazei de date, gestionarea cererilor API și loica serverului.

1. *Ciobanu Iulia* – Frontend Developer

• A contribuit la dezvoltarea interfeței utilizatorului(UI).

• A lucrat la implementarea designului.

1. *Bolohan Denisa* – Frontend Developer

• A contribuit la dezvoltarea interfeței utilizatorului(UI).

• A lucrat la proiectarea și implementarea aspectului vizual al aplicației și la funcționalitățile vizuale.

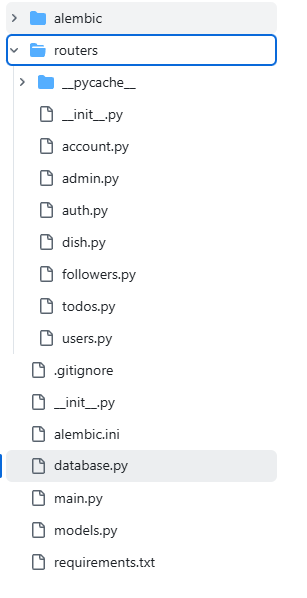
# ***Capitolul 7*** *– Dezvoltarea aplicației*

1. **Descrierea etapelor parcurse în dezvoltarea proiectului:**  
   Proiectul a urmat mai multe etape, incluzând planificarea aplicației, împărțirea sarcinilor în echipă, realizarea unui research amănunțit și implementarea lucrurilor esențiale. După ce am finalizat planificarea, am creat schema API-ului și baza de date, iar ulterior, am învățat și implementat treptat funcționalitățile necesare.
2. **Cum s-a realizat proiectarea bazei de date?**  
   Baza de date a fost proiectată conform cerințelor aplicației și obiectivelor stabilite. Elementele indispensabile, precum utilizator, postare și follow, au fost definite inițial. Am creat câteva tabele de bază, iar pe măsură ce aplicația s-a extins, am adăugat alte tabele prin migrare.
3. **Cum s-a realizat un design la API-urile REST?**  
   Pentru designul API-urilor, am început cu funcționalitățile de bază, cum ar fi autentificarea și logarea. Ulterior, am extins API-ul prin adăugarea altor endpoint-uri, în funcție de necesitățile aplicației.
4. **Cum a fost testat API-ul folosind Postman?**  
   Fiecare endpoint nou a fost testat inițial cu Swagger UI, pentru o verificare de bază, și apoi în Postman. În Postman, am utilizat instrumentele disponibile pentru a testa toate scenariile posibile, inclusiv cazurile de eroare și validările.
5. **Cum s-a utilizat Git pentru a versiona codul sursă?**  
   Pentru fiecare funcționalitate nouă, se crea un branch separat. După implementarea și testarea funcționalității, branch-ul respectiv era unit cu branch-ul principal (main).
6. **Contribuția proprie în fiecare etapă:**  
   Am lucrat la dezvoltarea API-ului, proiectarea bazei de date și implementarea funcționalităților aplicației. De asemenea, m-am ocupat de testarea și dezvoltarea backend-ului.
7. **Utilizarea platformei Azure DevOps pentru organizarea sarcinilor:**  
   În Azure DevOps, sarcinile necesare erau create și distribuite în funcție de disponibilitatea fiecărui membru al echipei.
8. **Lista cu sarcinile proprii:**

* Implementarea autentificării și logării
* Securizarea bazei de date
* Validarea datelor
* Crearea schemei API
* Dezvoltarea API-urilor
* Conectarea cu ORM
* Testarea funcționalităților în Postman

# *Capitolul 8 - Contribuția proprie*

8.1 Structura proiectului back-end



# 8.2 Tabel din baza de date create cu SQLAlchemy

Crearea tokenului de autorizare



Endpoint pentru a da like la o postare

# 

**8.3 Contribuția proprie pentru baza de date, schema:**Am proiectat și creat întreaga bază de date, care include toate tabelele necesare funcționării aplicației. Tabelele principale sunt:

* **Utilizator:** pentru gestionarea informațiilor despre utilizatori (nume, email, parola).
* **Postare:** pentru stocarea rețetelor, cu atribute precum titlu, descriere, ingrediente și utilizatorul care le-a publicat.
* **Follow:** pentru gestionarea relațiilor dintre utilizatori (cine urmărește pe cine).
* **Like Dish:** pentru stocarea aprecierilor la postările de rețete (utilizatorul care a dat like și postarea respectivă).
* **Notifications:** pentru gestionarea notificărilor, cum ar fi alerte pentru like-uri, comentarii sau follow.

Am realizat toate relațiile între tabele și am extins baza de date prin migrare, pe măsură ce aplicația s-a dezvoltat.

**8.4Contribuția proprie pentru API, testat în Swagger/Postman:** Am creat API-ul pentru aplicație, incluzând toate funcționalitățile necesare:

* **Autentificare și logare:** pentru accesul utilizatorilor la platformă.
* **Crearea postărilor:** pentru adăugarea rețetelor de către utilizatori.
* **Obținerea postărilor:** pentru afișarea rețetelor, fie ca listă, fie în detaliu.
* **Funcționalitatea de follow:** pentru gestionarea relațiilor între utilizatori.
* **Like-uri:** pentru a permite utilizatorilor să aprecieze rețetele altor utilizatori.
* **Notificări:** pentru informarea utilizatorilor atunci când cineva le apreciază o postare, îi urmărește sau le comentează la rețete.

# *Capitolul 9 – Configurarea și rularea aplicației*

Acest capitol detaliază procesul complet de configurare și rulare a aplicației. Ghidul include instalarea dependenșelor și pornirea aplicașiei în mod de dezvoltare.

Pentru rularea aplicației, sunt necesare următoarele instrumente și tehnologii:

• **Node.js** (versiune recomandata este v16.0.0).

• **npm (Node Package Manager)** – manager de pachete pentru JavaScript.

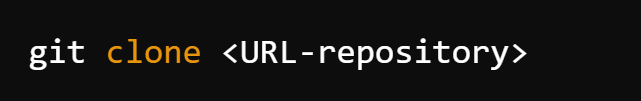
• **Git** pentru gestionarea codului sursă și pentru clonarea repository-ului.

• Un editor de cod – recomandat **Visual Studio Code.** Poate fi descărcat la <https://code.visualstudio.com/>.

Iată pașii necesari pentru a rula aplicația:

1. Descărcarea proiectului

Pentru a efectua clonarea repository-ului deschide terminalul și execută următoarea comandă pentru descărcarea proiectului de pe GitHub:



Înlocuind *URL-repository* cu link-ul repository-ului.

1. Accesarea directorului proiectului

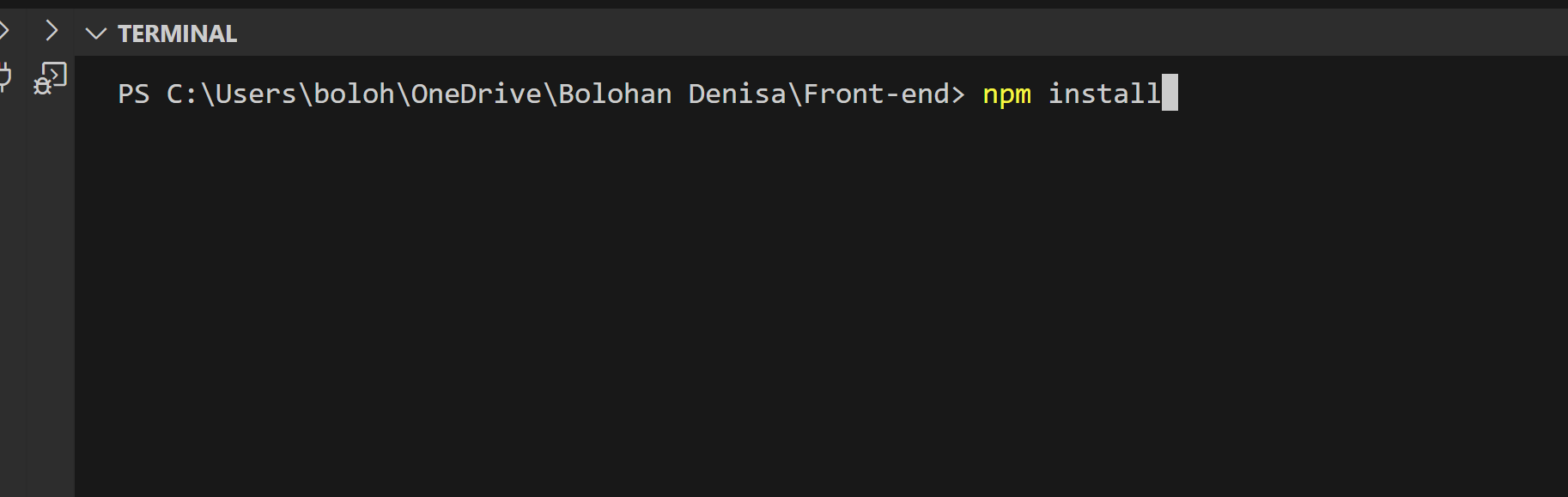
După clonare, navighează în directorul aplicașiei folosind comanda:



1. Instalarea dependențelor:

După descărcarea proiectului, este necesar să instalăm dependențele specificate în fișierul **package.json.**

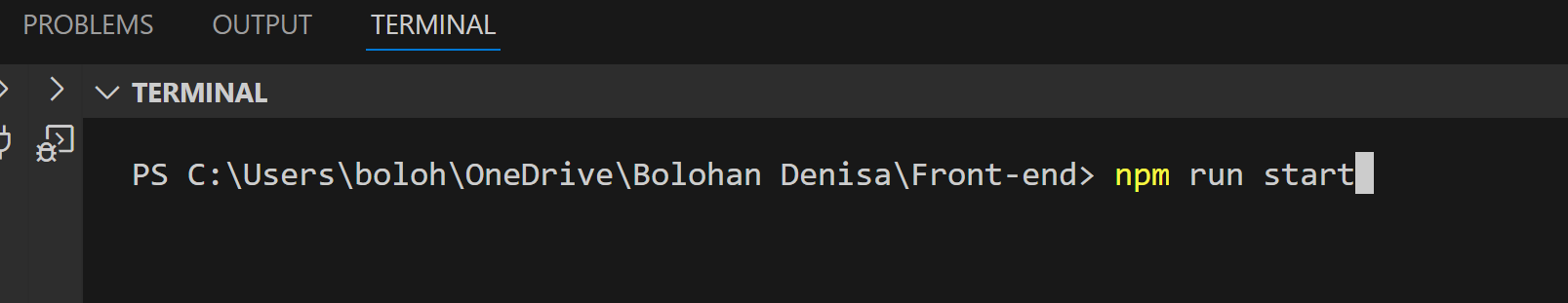
Deschide terminalul în directorul proiectului și rulează comanda pentru instalarea tuturor pachetelor necesaare:



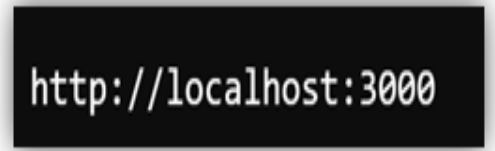
Aceasta va descărca și instala toate modulele necesare, cum ar fi React, axios, redux, sau alte pachete definite în package.json.

1. Pornirea aplicației

Pentru pornirea aplicației în modul de dezvoltare, accesam comanda:



Aplicația va fi disponibilă în browser la adresa:

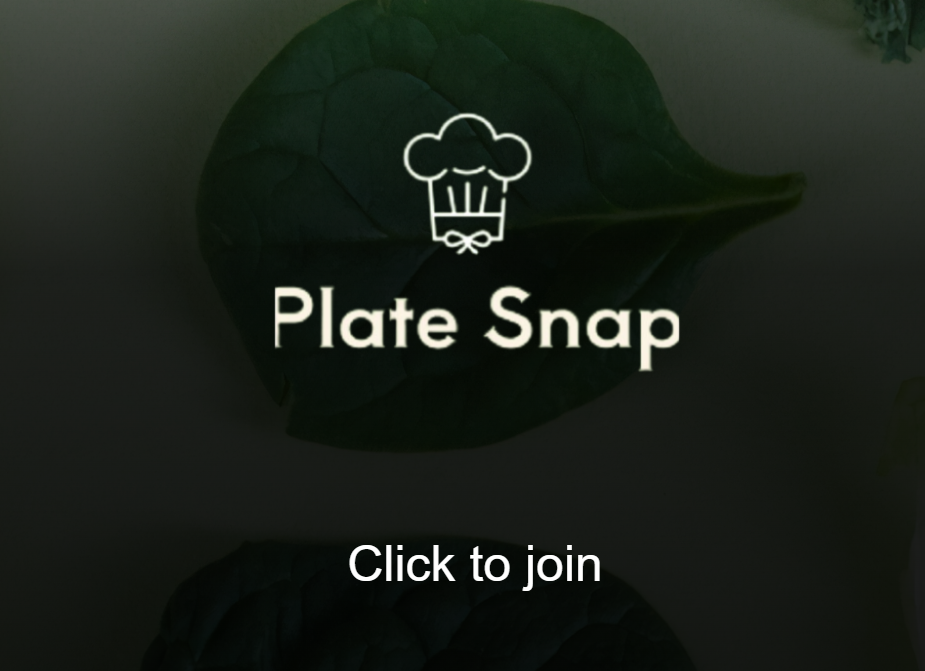


# *Capitolul 10 - Ghid de utilizar*

În acest capitolul se vor prezenta pașii necesari pentru a utiliza aplicația.

## 10.1 Authentificare și înregistarare

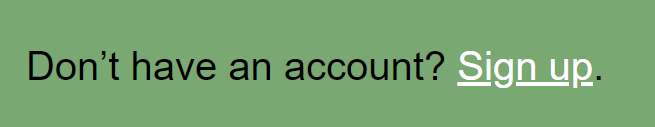
1. Accesează pagina principală și apasă butonul **Click to join.**



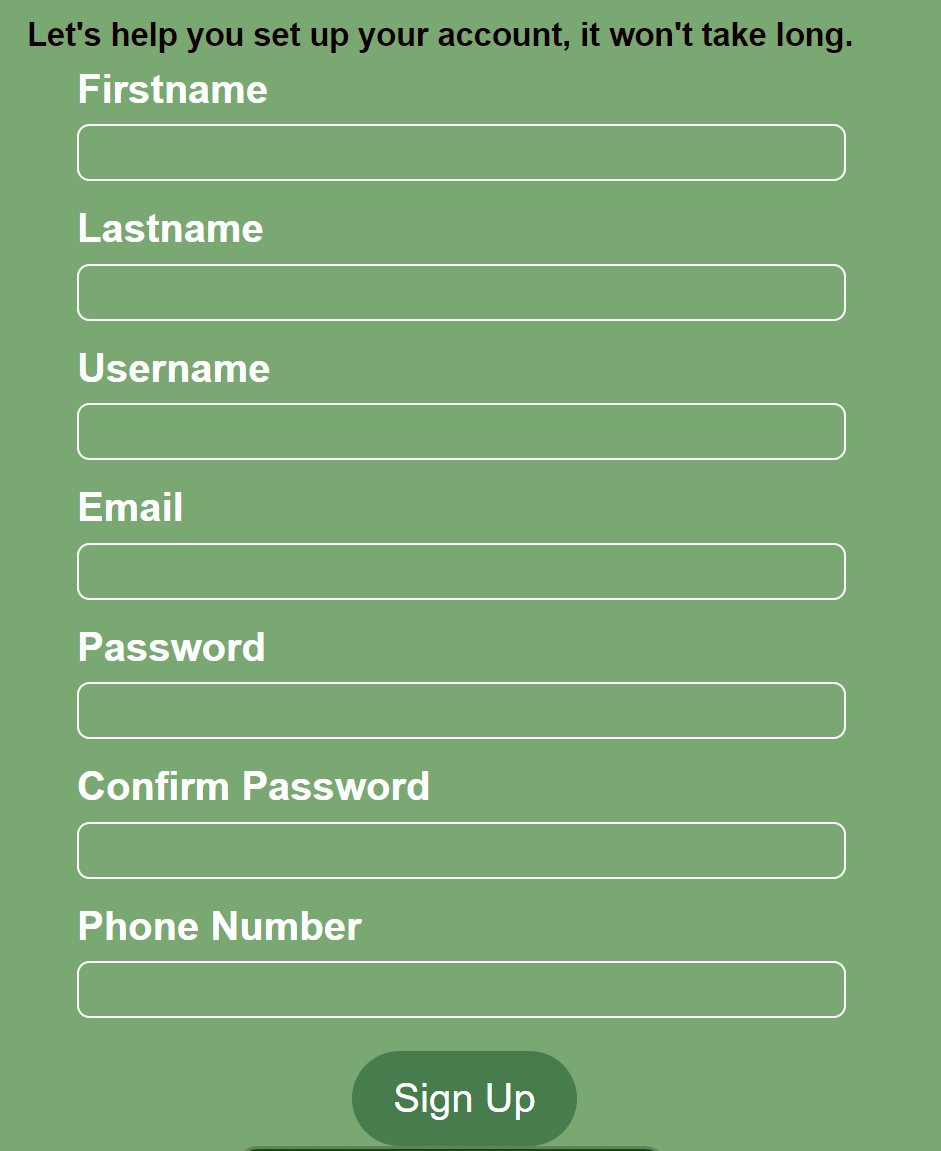
1. Dacă ai un cont în aplicație conectează- te la contul tău după cum urmează:



1. Dacă nu ai cont în josul ferestrei apasă pe **Sign Up:**

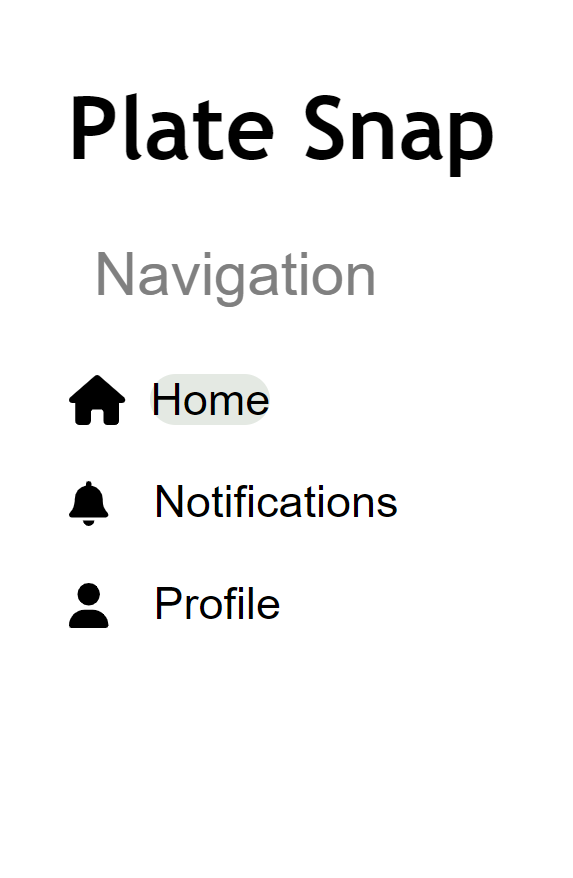


Fiind redirecționat pe o pagină unde te vei putea înregistra în aplicația noastră. Iar după completarea formularului apasă pe **Sign Up**.



## 10.2 Gestionarea profilului

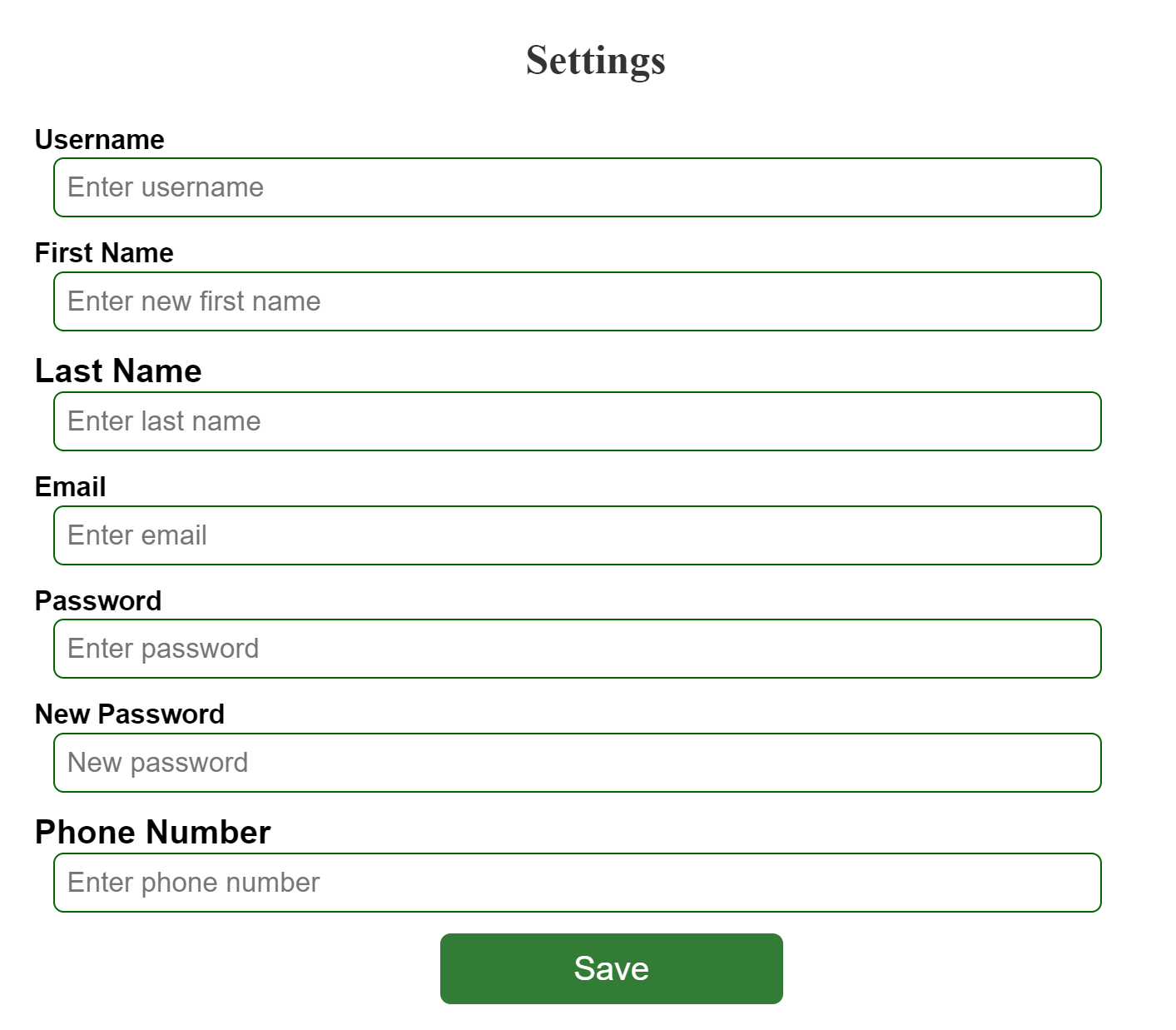
1. Accestează din meniul de navigare secțiunea **Profile.**



1. Aici poți:

• Modifica numele, prenumele, username, emailul sau parola.

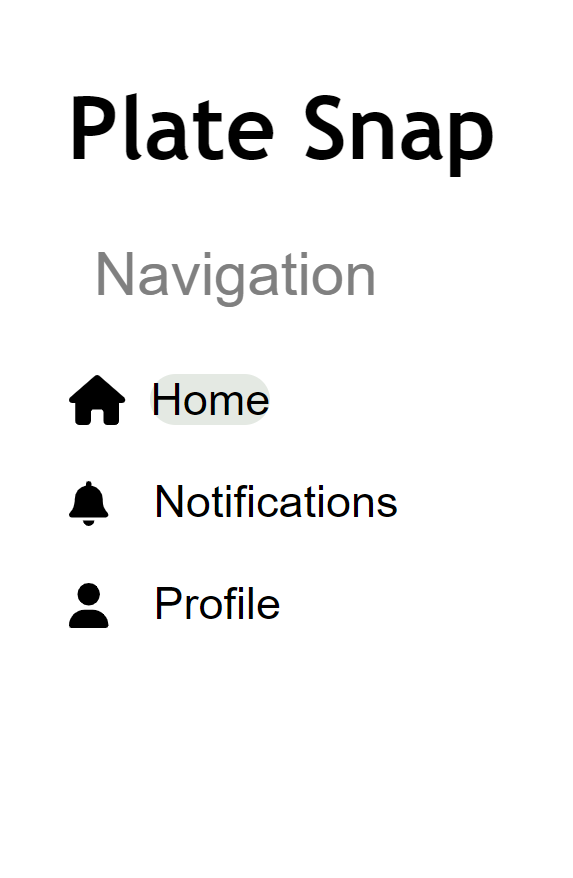
• Schimba poza de profil.



1. Apasă **Save.**

## 10.3 Vizualizarea și căutarea rețetelor

1. Accesează secțiunea **Home** din bara de navigare.



1. Folosește bara de căutare pentru a găsi rețetele după nume.



## 10.4 Adaugarea unei rețete noi

1. După autentificare, accesează butonul **Add recipe** din partea de jos a paginii.



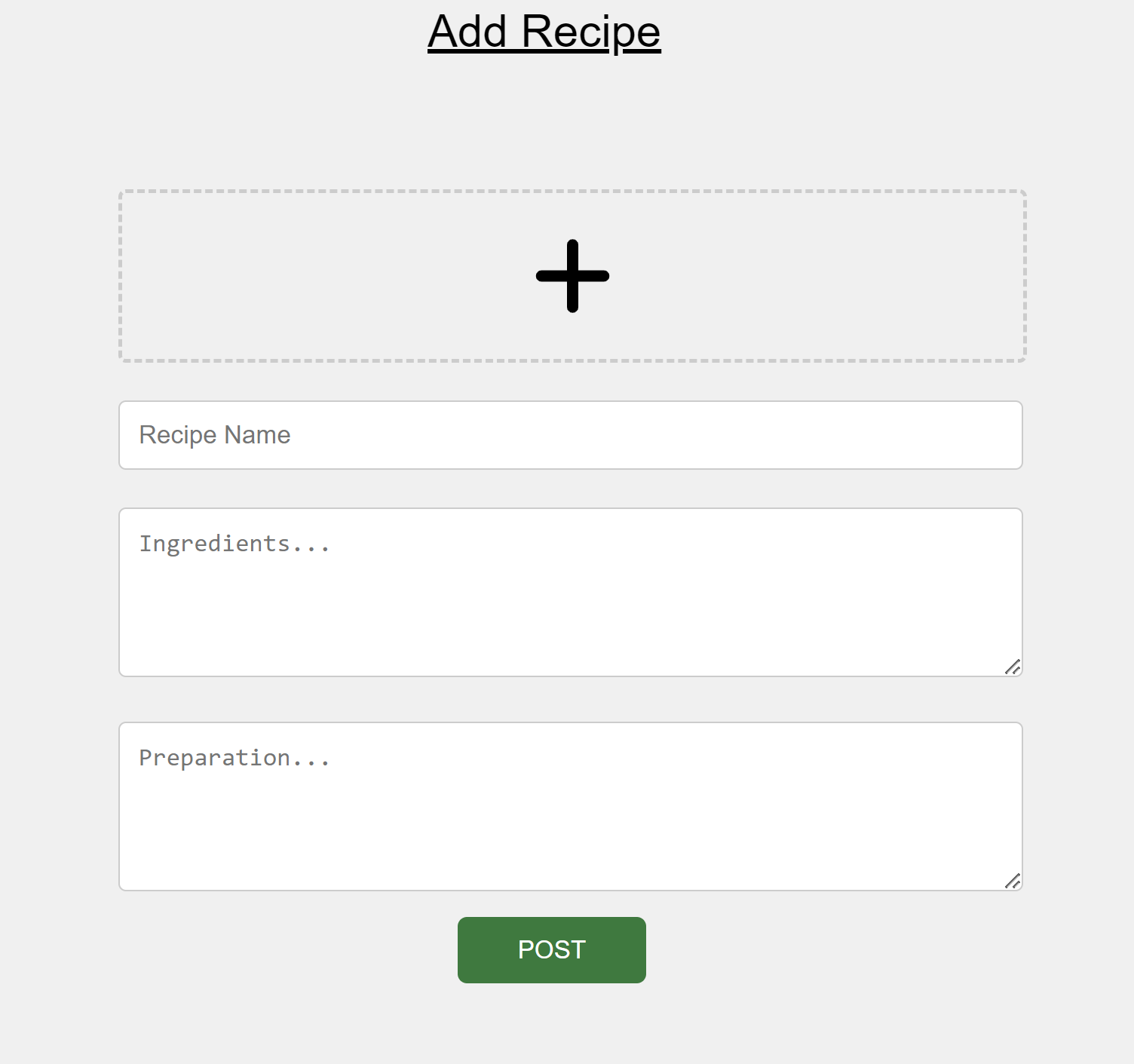
1. Completează formularul cu informațiile rețetei:

• O imagine reprezentativă cu rețeta.

• Numele reței.

• Ingredientele.

• Modul de prepare.



1. Apasă butonul **POST** pentru a finaliza postarea rețetei.

# *Capitolul 11 – Concluzii/ Lecții învățate*

**Concluzii despre procesul de dezvoltare și lecții învățate**

Procesul de dezvoltare al acestui proiect a fost complex, dar extrem de util, oferind o perspectivă clară asupra modului în care se construiește o aplicație funcțională, de la idee până la implementare. Una dintre cele mai importante a fost planificarea și lucrul in echipă.De asemenea, împărțirea sarcinilor în etape clare, precum proiectarea bazei de date, dezvoltarea API-urilor și testarea funcționalităților, a fost esențială pentru un lucru eficient in echipă.

Crearea bazei de date a fost o etapă fundamentală în dezvoltarea proiectului, care a evidențiat importanța planificarii. Am învățat că proiectarea unei baze de date bine gândite, cu relații clar definite, asigură funcționalitatea aplicației, dar facilitează și extinderea ulterioară a acesteia și este extrem de important planificarea corectă pentru eventualele migrări si normalizări.

Dezvoltarea și testarea API-urilor au fost o altă etapă în care am invățat multe. Am realizat cât de esențială este validarea fiecărui endpoint și testarea acestuia în toate scenariile, utilizând instrumente precum Postman. De la funcționalități de bază, cum ar fi autentificarea și logarea, până la implementarea funcțiilor mai complexe, precum notificările și like-uri, întreg procesul a fost extrem de important.

În final, o lecție esențială a fost importanța colaborării și organizării, în special prin utilizarea unor instrumente precum Git și Azure DevOps,totuși cel mai important fiind lucrul în echipă.Acest proces m-a învățat nu doar instrumente pentru a dezvolta aplciații,ci și cum să gândesc pe termen lung, să fiu atent la detalii și să lucrez eficient într-un mediu colaborativ.

# 

# **BIBLIOGRAFIE**

**Ce este un API:**

* [What is an API?](https://www.youtube.com/watch?v=s7wmiS2mSXY)
* API: Beginner to Advance: Building User Registration in FastAPI - Part

**SQLAlchemy:**

* SQLAlchemy Documentation
* [SQLAlchemy ORM Tutorial](https://www.youtube.com/watch?v=woKYyhLCcnU)

**Token JWT:**

* JSON Web Tokens - jwt.io
* [JWT Authentication Tutorial](https://www.youtube.com/watch?v=7Q17ubqLfaM)

**Git și GitHub:**

* [Git & GitHub Crash Course For Beginners](https://www.youtube.com/watch?v=RGOj5yH7evk)

**Cum utilizez Postman:**

* [Postman Beginner's Course](https://www.youtube.com/watch?v=VywxIQ2ZXw4)

**OAuth2:**

* [OAuth2 with Password (and hashing), Bearer with JWT tokens](https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/)

**FastAPI:**

* [Udemy course despre FASTAPI](https://www.udemy.com/course/fastapi-the-complete-course/?srsltid=AfmBOoqwQrmXSAMrFrZTQNEEDmFtJMeZfaGTxlVEfuBliRxpBe5PpUEt&couponCode=NEWYEARCAREER)