

E-Pharm

Site web pentru comenzi de medicamente

Proiect realizat de: Cezara Nedelea, Andreea Popovici
Grupa 30237

1.1 Scopul proiectului	3
1.2 Funcționalități principale	3
Tehnologii folosite	3
2.1 Frontend	3
2.2 Backend	3
1. Component-based Architecture (în frontend)	6
2. Singleton Pattern (în backend)	7
3. Navigarea în Aplicație	9
1 Pagina principală (HomePage)	9
2 Pagina cu produsele unei farmacii (PharmacyProducts)	10
3 Autentificare și Creare cont	11
4 Coșul de cumpărături (CartPage)	11
4. Fluxul de Date	11
5. Finalizarea comenzii	12

Introducere

1.1 Scopul proiectului

Acest proiect are ca scop dezvoltarea unei aplicații web de farmacii online numită E-Pharm. Utilizatorii pot naviga printre farmacii, vizualiza produsele disponibile în fiecare farmacie și comanda produsele dorite.

1.2 Funcționalități principale

- Vizualizarea unei liste de farmacii.
- Afișarea produselor disponibile în fiecare farmacie.
- Crearea unui cont pentru utilizatori și autentificarea acestora.
- Posibilitatea de a adăuga produse în coșul de cumpărături.
- Posibilitatea de a ieși din cont.
- Crearea unui cont pentru curieri și autentificarea acestora.
- Posibilitatea unui curier de a accepta sau a respinge o comandă.

Tehnologii folosite

2.1 Frontend

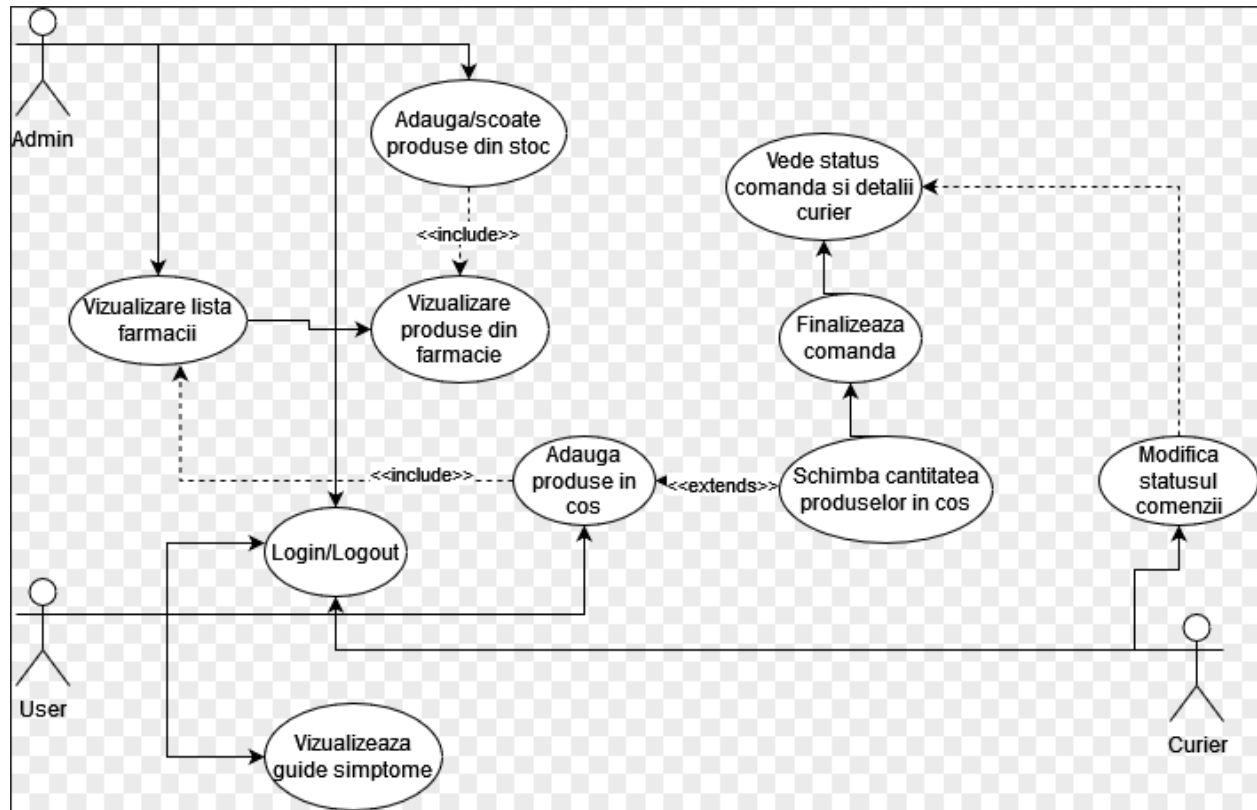
- **React.js** - Folosit pentru a crea interfața utilizatorului.
- **React Router** - Pentru gestionarea rutelor și navigarea între pagini.
- **Axios** - Pentru a face cereri HTTP către backend.
- **CSS** (sau **SCSS**) - Pentru stilizarea paginilor și componentelor.

2.2 Backend

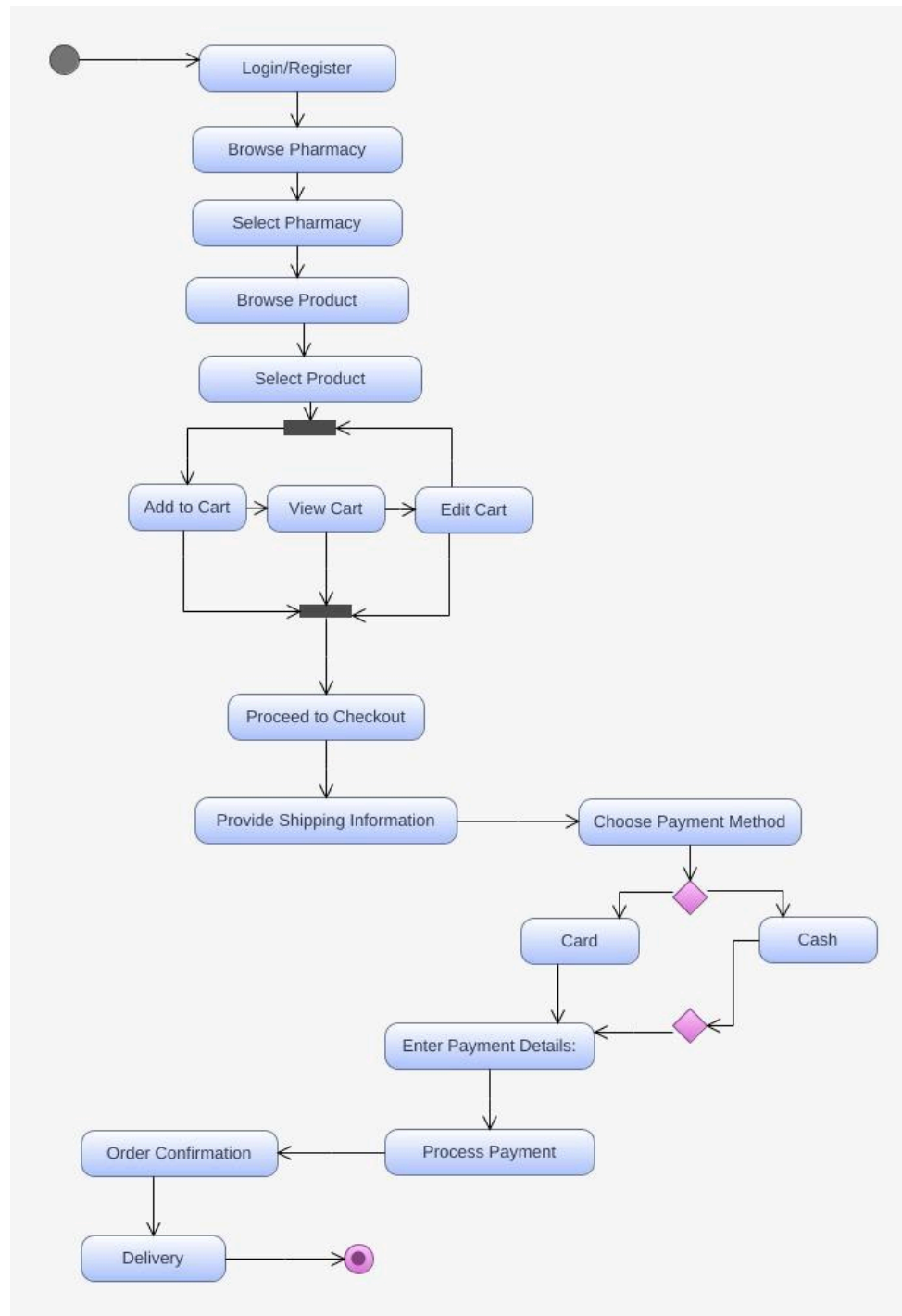
- **Node.js** - Folosit ca runtime pentru aplicația server.
- **Express.js** - Framework pentru crearea serverului și a API-urilor.
- **MongoDB** - Baza de date NoSQL folosită pentru stocarea datelor despre farmacii și produse.
- **Mongoose** - ODM pentru MongoDB, utilizat pentru manipularea bazei de date.

Diagrame

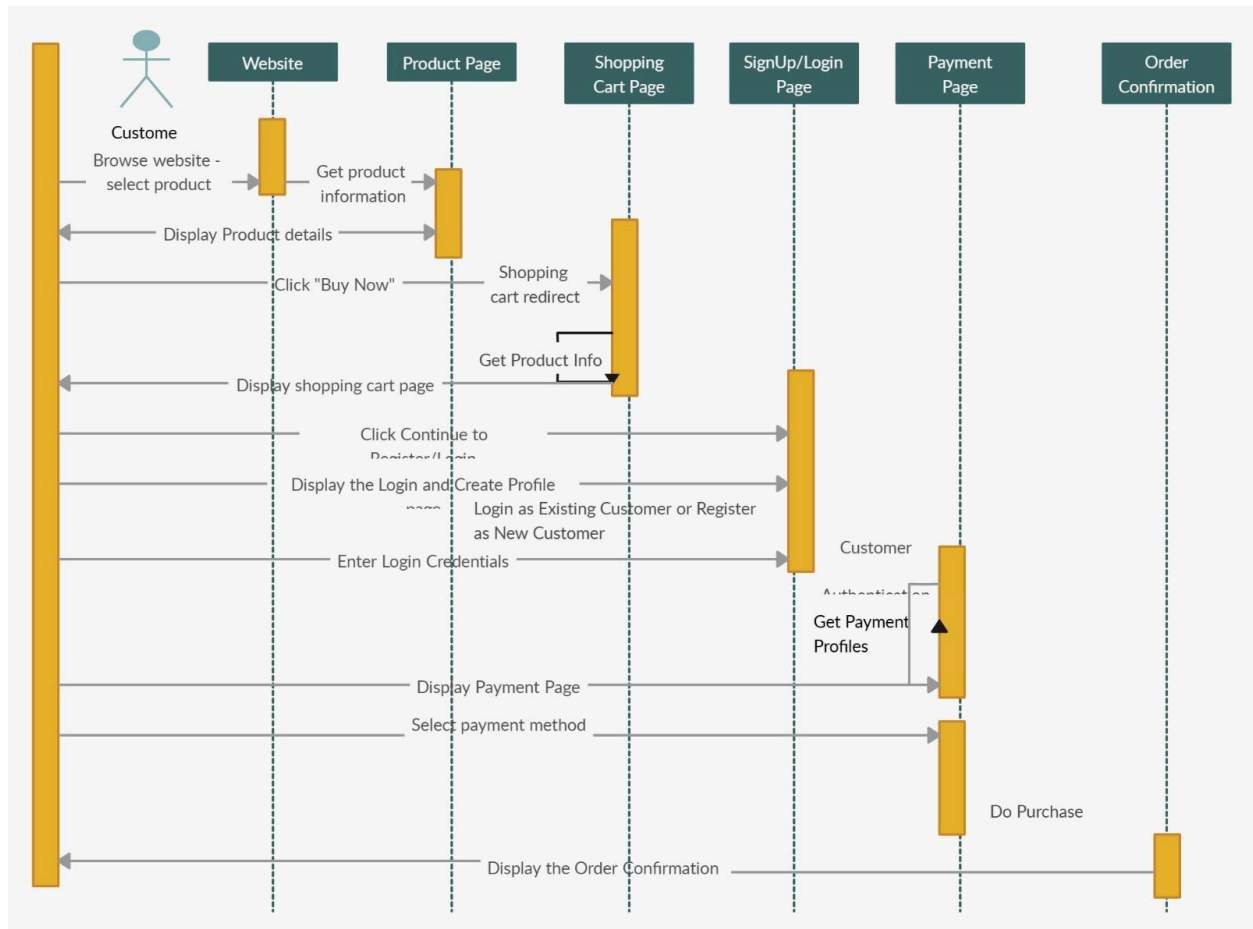
Use Case Diagram



Activity Diagram



Sequence Diagram



Design Patterns

1. Component-based Architecture (în frontend)

Deși nu este un "design pattern" clasic în sensul formal, arhitectura bazată pe componente din React urmează principiile unui pattern de arhitectură. React încurajează crearea de componente mici, reutilizabile care sunt ușor de întreținut și testat. În acest proiect, fiecare secțiune a paginii (ex. [HomePage](#), [PharmacyProducts](#), [CartPage](#), etc.) este o componentă separată, ceea ce ajută la menținerea unui cod modular.

- Principiul utilizat: Componentizarea
- Exemplu: Componente precum [HomePage](#), [PharmacyProducts](#), [CourierPage](#), fiecare gestionând o parte a UI-ului.

2. Singleton Pattern (în backend)

Singleton Pattern este un design pattern care asigură că o clasă are o singură instanță și furnizează un punct de acces global la această instanță. În backend-ul tău, acest pattern este folosit pentru gestionarea conexiunii la baza de date **MongoDB**.

```
import mongoose from "mongoose";

export const connectDB = async () => {
  try {
    console.log("mongo_uri: ", process.env.MONGO_URI);
    const conn = await mongoose.connect(process.env.MONGO_URI);
    console.log(`MongoDB Connected: ${conn.connection.host}`);
  } catch (error) {
    console.log("Error connection to MongoDB: ", error.message);
    process.exit(1); // 1 is failure, 0 status code is success
  }
};
```

Testare

În cadrul acestui proiect, procesul de testare a fost structurat în două categorii principale: testarea backend-ului și testarea frontend-ului.

Testarea backend-ului

Pentru backend, testarea a fost realizată utilizând Postman, un instrument versatil care ne-a permis să verificăm funcționalitatea API-urilor. Am evaluat următoarele aspecte:

1. Endpoint-uri funcționale: Fiecare endpoint a fost testat pentru a valida răspunsurile corecte la cereri de tip GET, POST, PUT și DELETE.
2. Validarea datelor: Am testat datele trimise către server pentru a ne asigura că respectă structura așteptată și pentru a verifica comportamentul aplicației în cazul datelor incomplete sau incorecte (ex: email invalid, ID-uri inexistente).
3. Erori și coduri de răspuns: Codurile de răspuns HTTP au fost verificate pentru a ne asigura că fiecare cerere primește răspunsul corespunzător:
 - 200OK pentru cereri reușite;
 - 400BadRequest pentru date incorecte;

- 401Unauthorized pentru cereri fără autentificare validă;
- 404NotFoundpentru resurse inexistente;
- 500Internal Server Error pentru erori interne.

Testarea frontend-ului

Pentru frontend, testarea a fost realizată utilizând consola browserului, cu accent pe următoarele aspecte:

1. Interfața utilizatorului (UI): S-a verificat afișarea corectă a tuturor componentelor grafice pe diferite rezoluții și dispozitive.
2. Interacțiunile utilizatorului: Funcționalitățile interactive au fost testate, cum ar fi:
 - Selectarea profilului de utilizator: client sau curier;
 - Selectarea unei farmacie din meniu;
 - Acceptarea sau refuzul unei comenzi.
3. Validarea datelor: Formularele aplicației au fost testate pentru a ne asigura că utilizatorii primesc mesaje de eroare adecvate în cazul introducerii unor date greșite (ex: câmpuri goale, parolă incorectă).
4. Conexiunea cu backend-ul: S-a monitorizat comunicarea dintre frontend și backend, analizând cererile și răspunsurile API prin consola browserului pentru a identifica eventuale probleme.

Rezultatele testării

1. Backend-ul funcționează conform specificațiilor, iar erorile sunt gestionate corespunzător. Datele sunt salvate și prelucrate corect.
2. Frontend-ul oferă o experiență de utilizare fără erori, iar toate interacțiunile sunt funcționale și intuitive

Mod de rulare al aplicației

Aplicația este dezvoltată utilizând React pentru frontend și Java Script pentru backend. Pornirea aplicației presupune configurarea ambelor componente, asigurându-se că toate dependențele și setările necesare sunt corect implementate.

1. Configurarea mediului de lucru

1. Cerințe preliminare:

- Node.js (pentru frontend)– descărcați și instalați de pe Node.js.
- JavaDevelopment Kit (JDK)– versiunea 21 sau mai recentă.
- MongoDB pentru gestionarea bazei de date.
- Postman pentru testarea API-urilor backend (opțional).
- IntelliJ IDEA
- Visual Studio Code pentru frontend.

2. Baza de date:

- Instalați și configurați serverul bazei de date.
- Creați o bază de date nouă.
- Importați schema bazei de date folosind fișierul SQL furnizat.

2. Pornirea aplicației

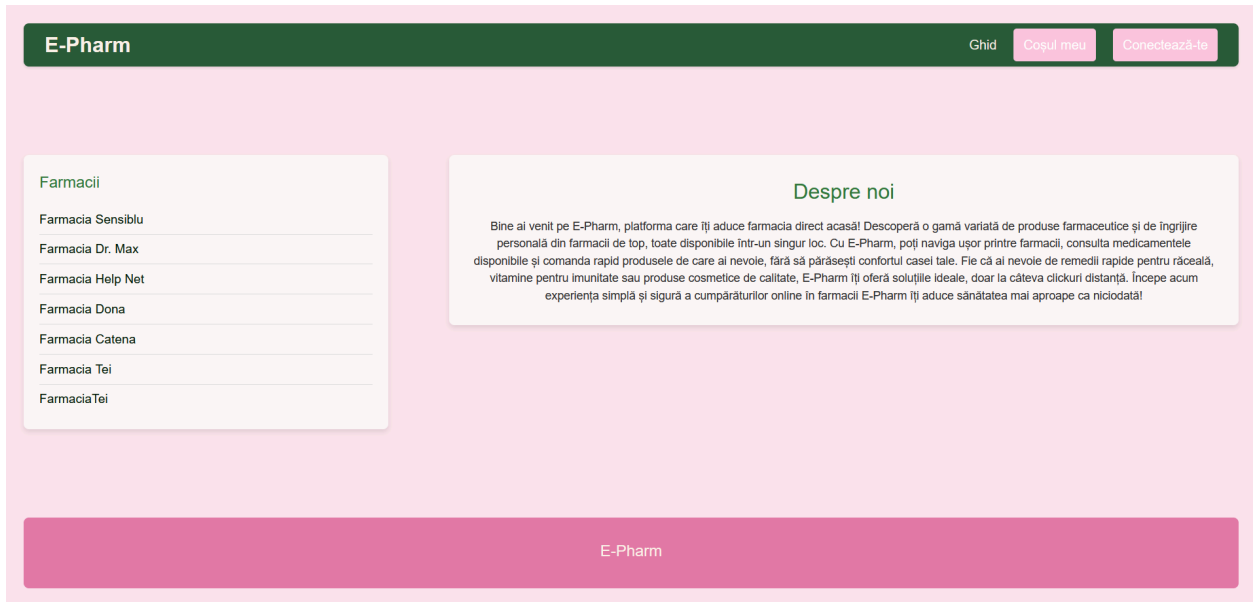
1. Lansați backend-ul din terminal folosind comanda `npm run dev`.
2. Navigați în folderul frontend, iar apoi lansați frontend-ul din terminal folosind comanda `npm run dev`.
3. Accesați aplicația în browser pe <http://localhost:5173>.

3. Navigarea în Aplicație

1 Pagina principală (HomePage)

La deschiderea aplicației, utilizatorul va fi întâmpinat de pagina principală (**HomePage**), unde vor apărea următoarele:

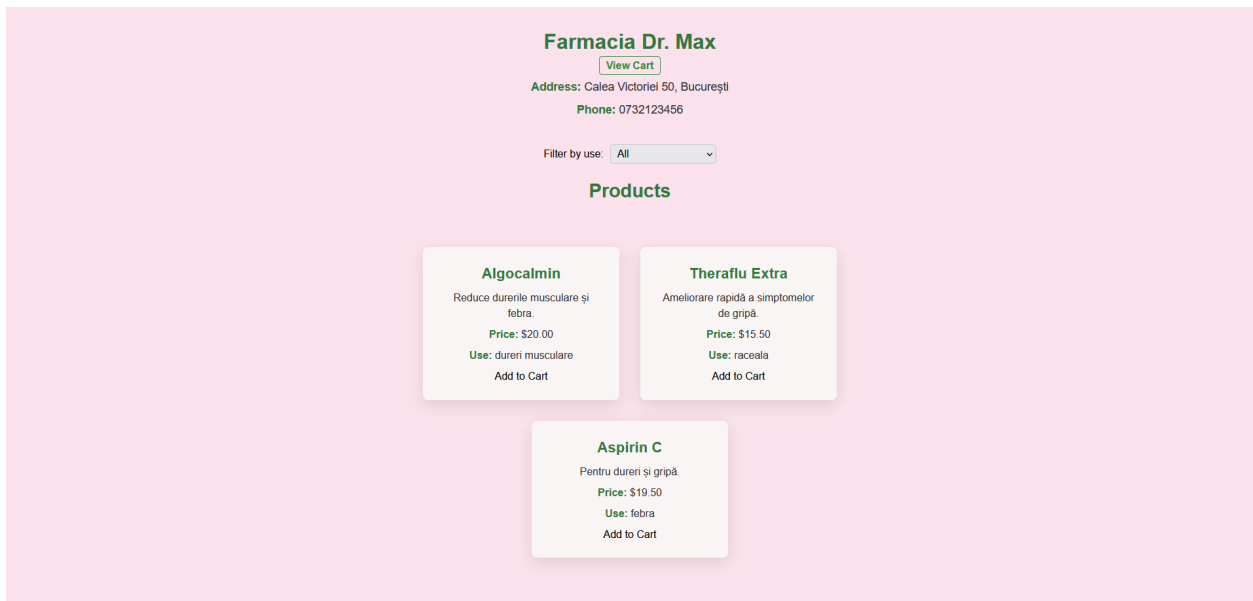
- **Header:** Include logo-ul aplicației și opțiuni de navigare (Cosul meu, Conectează-te).
- **Meniu farmacii:** Afișează o listă de farmacii. Când utilizatorul dă clic pe numele unei farmacii, va fi redirecționat către pagina acesteia cu produsele disponibile.
- **Secțiunea Despre noi:** Afișează informații despre aplicație.
- **Ghid Medicamente:** Afișează o serie de butoane pentru diverse categorii de medicamente (dureri de cap, febră etc.).



2 Pagina cu produsele unei farmacii (PharmacyProducts)

Când utilizatorul selectează o farmacie din lista de pe HomePage, va fi redirecționat către o pagină care afișează produsele disponibile pentru acea farmacie. Informațiile sunt preluate din backend-ul tău și sunt afișate într-o listă.

- **Produse afișate:** Numele produsului, descrierea și prețul.
- **Mesaje de încărcare:** Dacă datele sunt încărcate, se va afișa mesajul „Se încarcă...”.



3 Autentificare și Creare cont

Dacă utilizatorul nu este autentificat, va putea accesa pagina de autentificare (</login>). Aici, va trebui să completeze un formular cu email-ul și parola.

După autentificare, utilizatorul va fi redirecționat către pagina de profil sau către pagina principală a aplicației.

The image displays three distinct user interface screens for authentication and registration:

- Choose Your Role:** A screen with two green buttons labeled "Client" and "Curier". Below them is a grey button with the text "Select your role to proceed."
- Welcome back!:** A screen with input fields for "Email Address" and "Password", a green "Login" button, and a link "Sign Up" for users who don't have an account.
- Courier Login:** A screen with a pre-filled email field "andreear.popovici@gmail.com", a masked password field "****", a green "Login" button, and a link "Click aici" for users who want to register.

4 Coșul de cumpărături (CartPage)

Utilizatorul poate adăuga produse în coș din paginile farmaciei. În pagina de coș, utilizatorul va putea vizualiza produsele adăugate, va putea modifica cantitățile și va putea finaliza comanda.

The image shows a "Your Cart" page with a pink background. It contains the following information:

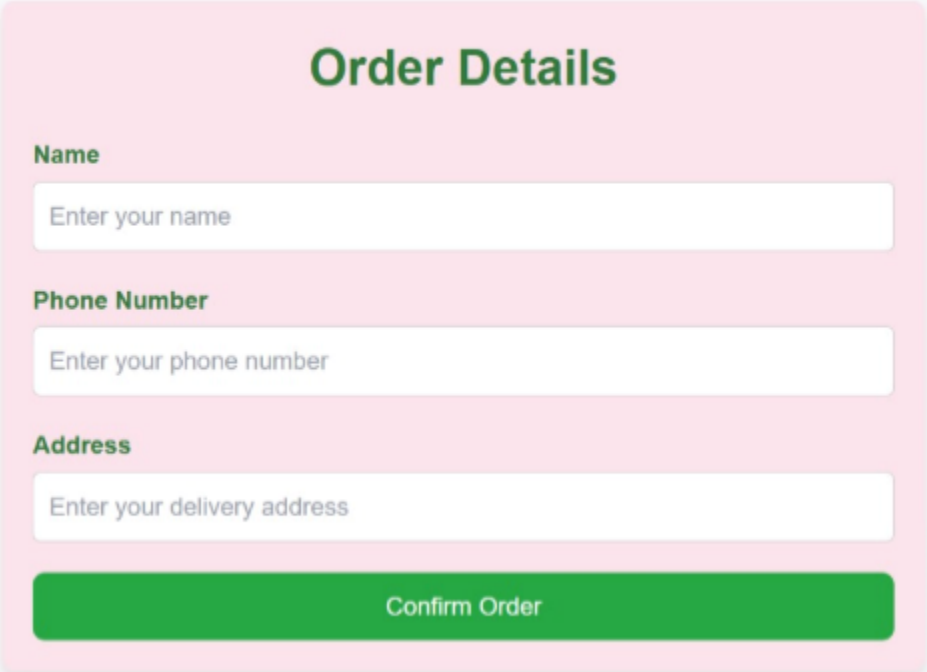
- Algocalmin:** Quantity: 1, Price: \$20.00
- Theraflu Extra:** Quantity: 1, Price: \$15.50
- Total:** \$35.50
- A green button labeled "Proceed to Order Details"

4. Fluxul de Date

1. **Frontend:** Aplicația React va trimite cereri HTTP către backend pentru a obține lista de farmacii și produse disponibile. Folosește **Axios** pentru a trimite cereri GET și pentru a obține datele din backend.
2. **Backend:** Serverul Express va răspunde la cererile din frontend prin rute API care oferă date despre farmacii și produse. Aceste date sunt preluate din baza de date **MongoDB**.
3. **Baza de date:** Datele despre farmacii și produse sunt stocate într-o bază de date **MongoDB**, care este gestionată de **Mongoose**. Backend-ul va interacționa cu această bază de date pentru a adăuga, modifica și șterge înregistrări.

5. Finalizarea comenzii

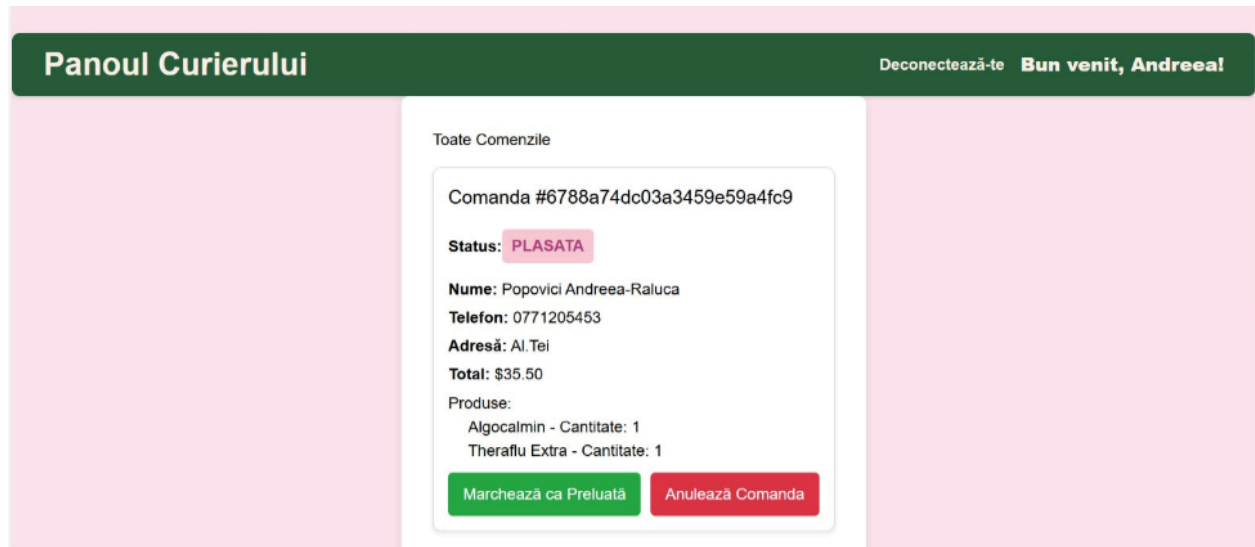
Pentru a finaliza comanda, utilizatorul va trebui să apese pe butonul de finalizare a comenzii, iar aplicația va trimite o cerere către serverul backend pentru a procesa comanda.



The image shows a web form titled "Order Details" with a pink background. It contains three input fields: "Name" with the placeholder "Enter your name", "Phone Number" with the placeholder "Enter your phone number", and "Address" with the placeholder "Enter your delivery address". Below these fields is a green button labeled "Confirm Order".

6 Confirmarea comenzii

Curierul poate vedea toate comenzile, selectează ce comanda vrea și o poate marca drept Preluată, apoi Livrată sau o poate anula.



Rol în echipă

Cezara: Documentație, frontend, Component-based Architecture, Use-case Diagram, Activity diagram

Andreea: Backend, introducere de date în baza de date, Singleton Pattern, Use-Case Diagram, Sequence Diagram