

Egyptian E-Learning University

Faculty of Computers & Information Technology

PharmaBridge: A website to manage the Medications donation process to reduce medical waste

By

Shahd Alaa Mostafa	21-01595
Salma Emad Mohamed	21-02292
Yomna Ali Ibrahim	21-01701
Mohamed Abd-Elrheem	21-01149
Kareem Ahmed Mohamed	21-00839
Ahmed Essam Eldeen Okasha	21-00328
Ahmed Abd-ElHalim Ali	20-01089

Supervised by

DR. Ahmed ElSayed Yakoub

Assistant

Eng. Khaled Mohamed Mahmoud
[sohag]-2025

Abstract

This web site is designed to streamline the collection and redistribution of surplus medications, addressing two critical issues:

medical waste and healthcare accessibility. By connecting donors with unused, unexpired medications to individuals in need, the platform offers essential drugs at significantly reduced costs.

The primary goals of the application are to reduce medical waste and enhance healthcare accessibility. This initiative aims to minimize the environmental impact of discarded medications and alleviate the financial burden many individuals face when seeking necessary treatments. Donors will register on the platform and list available medications. Recipients can browse these listings and submit requests. The system will match donors with recipients based on location and needs, with automated responses assisting throughout the process.

Acknowledgments

First and foremost, praises and thanks to God, the Almighty, for his blessing throughout our years in college and our graduation project to complete this stage of our life successfully.

We have made efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Thanks to Egyptian E-Learning University especially Sohag center for helping us to reach this level of awareness.

We would like to express our deep and sincere gratitude to our project supervisor **DR. Ahmed ElSayed Yakoub** for giving us the opportunity to lead us and providing invaluable guidance throughout this project. It was a great privilege and honor to work and study under her guidance. We are extremely grateful for what she has offered us.

We are especially indebted to say many thanks to **Eng. Khaled Mohamed Mahmoud** for guidance and constant supervision as well as for his patience, friendship, and empathy.

Finally, we would like to express our gratitude to everyone who helped us during the graduation project.

Contents

Abstract.....	2
Acknowledgments.....	3
1- Introduction.....	7
1.1 Introduction.....	8
1.2 Background and motivation for the project.....	8
1.3 Importance of the problem being addressed.....	8
1.4 Problem Statement.....	8
Clear definition of the problem your project addresses.....	8
Justification for why this problem is worth solving.....	8
1.5 Objectives.....	9
1.5.1 Main Objectivet.....	9
1.5.2 Specific Objectives.....	9
1.6 Brief overview of the proposed solution.....	9
2- Literature Review / Related Work.....	10
2.1Summary of existing research and technologies related to your project.....	11
2.2 Gaps in current solutions that your project aims to fill.....	17
2.3 Summary.....	18
3- Proposed system.....	21
3.1 Approach used to solve the problem.....	22
3.2System architecture.....	22
3.3 Algorithms or frameworks used.....	32
4- Implementation.....	36
4.1Technologies, tools, and programming languages used.....	37
4.2Key components/modules of the system.....	42
4.3Challenges faced and how they were resolved.....	72

4.4 UI Design.....	75
5- Testing & Evaluation.....	82
5.1 Testing strategies.....	85
5.2 Performance metrics.....	86
5.3 Comparison with existing solutions	86
6- Results & Discussion.....	88
6.1 Introduction.....	89
6.2 Summary of findings.....	89
6.3 Interpretation of results	90
6.4 Limitations of the proposed solution.....	94
7- Conclusion & Future Work.....	96
7.1 Summary of contributions.	97
7.2 Possible improvements or extensions for future work.....	98
References	100

Figures

Figer 1: use case diagram.....	23
Figer 2: sequence diagram (sign up process).....	24
Figer 3: sequence diagram (login process)	25
Figer 4: sequence diagram (add medicine process).....	26

Figur 5: sequence diagram (add offer process).....	27
Figur 6: sequence diagram (oeder process).....	28
Figur 7: sequence diagram (rating process)	29
Figur 8: communication architecture.....	30
Figur 9: class diagram.....	31

Tables

Table 1: gap analysis Table.....	20
Table 2: challenges Table.....	73
Table 3: performance Table.....	87
Table 4: pack log Table.....	92
Table 5: future work Table.....	99

Chapter 1:

Introduction

1.1 Introduction

The increasing gap between the availability of essential medications and the ability of individuals to afford them has become a significant global health challenge. Simultaneously, the issue of medical waste, especially the disposal of surplus and unused medications, poses serious environmental risks. This project introduces a web-based platform, PharmaBridge, designed to bridge this gap by facilitating the collection and redistribution of surplus medications to individuals in need. The platform aims to simultaneously address two critical problems: minimizing medical waste and improving healthcare accessibility.

1.2 Background and Motivation for the Project

Worldwide, millions of tons of medications are discarded annually due to expiration, surplus stock, or changes in prescription, contributing to environmental pollution and resource wastage. Concurrently, a substantial portion of the global population faces difficulties in accessing affordable medication, often leading to untreated health conditions and increased healthcare costs. These interconnected issues call for an innovative, technology-driven approach that maximizes the use of available resources while supporting vulnerable communities.

1.3 Importance of the Problem Being Addressed

The improper disposal of medications leads to contamination of soil and water sources, affecting ecosystems and public health. At the same time, financial constraints prevent many from obtaining necessary treatments, exacerbating health disparities. By creating an efficient platform for medication redistribution, this project addresses the urgent need to reduce environmental hazards and enhance equitable access to healthcare.

1.4 Problem Statement

The central problem addressed by this project is the dual challenge of excessive medical waste due to unused medications and the limited

accessibility of affordable medications for underprivileged populations. Current healthcare systems lack efficient mechanisms to reclaim and redistribute surplus medications, leading to unnecessary wastage and health inequity. This project seeks to develop a solution that not only mitigates medication wastage but also ensures timely and fair access to essential drugs for those in need.

1.5 Objectives

- Main Objective:

To design and implement a web-based platform that streamlines the collection, matching, and redistribution of surplus medications to individuals requiring affordable healthcare solutions.

-Specific Objectives:

1. Develop a user-friendly interface for donors to register and list available medications with relevant details such as expiration date, and quantity.
2. Enable recipients to browse available medication listings and submit requests.
3. Design a notification system to inform donors and recipients of successful matches and transaction status.
4. Ensure system security, data privacy, and compliance with relevant healthcare regulations.

1.6 Brief Overview of the Proposed Solution

PharmaBridge operates as an interactive platform connecting donors and recipients through a streamlined process. Donors list their surplus medications, while recipients can search and request medications they need. An intelligent matching algorithm processes these inputs to optimize distribution efficiency.. By leveraging technology, the platform promotes environmental sustainability and enhances healthcare accessibility in a socially responsible manner.

Chapter 2

Literature Review / Related Work

2.1 Existing Research and Technologies

A number of research initiatives and digital platforms have been developed in recent years to address the problem of medication waste and improve access to essential healthcare. Many of these systems focus on improving logistics in pharmaceutical supply chains, tracking expiration dates, and promoting donation of surplus medications. Some notable technologies include inventory management systems in hospitals, electronic medical records that flag unused prescriptions, and mobile applications that facilitate community-driven medical donations.

However, despite the development of such systems, the majority are either localized, lack scalability, or are not tailored to individual user needs in underserved regions. Existing systems often require extensive human intervention or lack automation in matching donors with recipients. Furthermore, many of the available solutions do not integrate environmental concerns or focus solely on institutional settings, neglecting individual contributors and beneficiaries.

These solutions vary in scope, features, and target users. Below is a summary of key platforms relevant to this project :

1. Dawaey

Dawaey launched in 2020 as a mobile health application, designed to help users manage their medications and prescriptions in an organized, and user-friendly way. The app offers features like medication reminders, dose tracking, and notifications for upcoming doses, making it easier for users to stick to their treatment schedules.

Dawaey also provides detailed information about various medications, including side effects and ingredients, along with a comprehensive database for searching drugs and finding available alternatives in the market.

This app is ideal for individuals who need to manage multiple medications and want a reliable way to keep track of their dosage times.

Additionally, the app sometimes offers health tips and guidelines for optimal medication use, enhancing the user experience and supporting health practices.

2. HealthWarehouse

HealthWarehouse launched in 2007 ,IT is an online pharmacy based in the U.S. that offers affordable prescription and over-the-counter medications directly to consumers without requiring health insurance. With a commitment to transparency and accessibility, the platform provides a Broad range of healthcare products, including prescriptions, pet medications,medical supplies, and health and beauty products.

HealthWarehouse stands out for its user-friendly features like prescription history tracking, medication alerts, loyalty programs, and monthly savings options. It also offers extensive educational resources and a comparison tool that allows users to evaluate generic and brand-name drugs, helping them make informed decisions. HealthWarehouse ships nationwide and internationally on select items, ensuring convenience and accessibility for customers.

3. PharmAct

PharmAct is a digital platform launched in 2020 that aims to enhance the efficiency and utilization of the pharmaceutical industry. Designed for healthcare professionals, patients, and pharmaceutical companies, it features an intuitive interface that facilitates seamless data management and patient engagement This is through its main functions such as :

Ensure that all operations align with industry regulations and standards, minimizing legal risks. Track drug performance and outcomes through comprehensive data analysis, helping stakeholders make informed decisions.

It allows healthcare professionals to share information about available medicines, enhance collaboration and address shortages. A newly added feature allows users to display their surplus medications under specific conditions and other users can purchase them. This feature was added to the platform some time after its launch, as they saw it as an important feature to enhance the platform and serve the community. Despite its innovative features, PharmAct faces challenges such as adoption resistance, data security concerns, drug security, regulatory hurdles, and interoperability issues with existing systems.

Despite these hurdles, PharmAct stands out as an innovative solution that has the potential to transform the pharmaceutical landscape by improving communication and patient care.

4. RX Outreach

Rx Outreach was launched in 2010 as a nonprofit pharmacy. The organization's mission is to provide affordable prescription medications for individuals and families who are uninsured, or facing financial challenges. By offering access to a broad range of medications at reduced costs, Rx Outreach has served millions of Americans in need. Since its inception, it has saved patients over \$1 billion on prescription medications.

Its main functions include:

- Affordable Medications:** offers over 1,000 different prescription medications at reduced prices, often less than typical pharmacies.
- Convenient Online Access:** Allowing patients to order medications easily via its online platform.
- Eligibility Screening:** Using a simple, income-based eligibility process to ensure support for those in need.

Educational Resources: Rx Outreach also provides resources on medication management, wellness, and healthcare literacy to empower patients in managing their health. Rx Outreach faces several challenges like :

Regulatory Compliance: Navigating complex state and federal pharmacy regulations. **Funding and Sustainability:** Being a nonprofit limits resources for growth. **Awareness and Outreach:** Reaching target populations who may lack Digital access.

5. CVS

CVS Pharmacy is a leading retail pharmacy chain that provides a wide range of health and wellness products and services. With thousands of locations nationwide, CVS offers prescription medications over-the-counter drugs, personal care items, and health essentials. In CVS provides immunizations, health screenings, and consultation services to addition to pharmacy services, support patient care.

The store also features a convenient variety of everyday items, from snacks to household goods, making it a one-stop shop for community health needs. With a focus on accessibility and customer service, CVS Pharmacy aims to enhance the health and well-being of its customers.

6. Shefa

Comprehensive Online Pharmacy: Shefa offers a vast array of pharmaceutical products, including over-the-counter medications, vitamins, supplements, and more. **Convenience and Accessibility:** With just a few clicks, users can purchase their needed medications and have them delivered directly to their doorstep. **Competitive Pricing:** Shefa often provides competitive prices on a wide range of products, making it a cost-effective option for healthcare needs.

Wide Product Range: The platform caters to various health needs, offering products for different age groups and health conditions. **Easy-to-Use Interface:** The user-friendly interface makes it simple for customers to navigate the website and find the products they need. **Secure Transactions:** Shefa prioritizes the security of customer information and transactions, ensuring a safe shopping experience. **Reliable Delivery:** Orders are typically delivered promptly and reliably, ensuring that customers receive their medications on time. **Customer Support:** Shefa provides customer support to assist users with any questions or concerns they may have. **Integration with Healthcare Providers:** In some cases, Shefa may offer integrations with healthcare providers, allowing for a more seamless healthcare experience. **Educational Resources:** The platform may also provide educational resources and information about various health conditions and medications.

Additional features to highlight depending on the platform:

Subscription Services: If offered, highlight the convenience of subscription services for recurring purchases. **Personalized Recommendations:** Mention any features that provide personalized product recommendations based on a user's health profile. **Mobile App:** If available, emphasize the convenience of using the Shefa app for on-the-go purchases. **Loyalty Programs:** If there are loyalty programs or rewards, mention the benefits for frequent customers. **Integration with Health Insurance:** If the platform works with specific health insurance providers, highlight this feature.

7. Drugs.com

Drugs.com is an extensive online pharmaceutical encyclopedia that provides reliable and comprehensive drug information primarily targeted toward healthcare professionals and consumers, particularly in the United States. It is one of the most popular and well-regarded resources for up-to-date

medication details on the over 24,000 prescription medications, over-the-counter drugs, and natural products.

The domain Drugs.com was initially registered in 1994 by Bonnie Newbrough. In 1999, it was acquired by Eric MacIver and then sold to Venture Frogs. Officially launched in September 2001 after being acquired by a private investor. Drugs A-Z Guide: An extensive database offering detailed information on over 24,000 prescription and over-the-counter medications, including uses, dosages, side effects, and precautions.

Pill Identifier: A tool that helps users identify medications based on physical characteristics such as imprint, shape, and color. **Interaction Checker:** Allows users to check for potential interactions between multiple drugs, helping to prevent adverse effects.

Mobile Applications: Offers mobile apps for iOS and Android devices, enabling users to access drug information, identify pills, and check interactions on the go. **FDA Alerts:** Provides updates and alerts from the U.S. Food and Drug Administration regarding drug approvals, recalls, and safety warnings. **Over-the-Counter (OTC) Database:** Contains information on non-prescription medications, including their uses and recommended dosages.

Pregnancy and Breastfeeding Warnings: Offers guidance on the safety of medication use during pregnancy and breastfeeding, assisting users in making informed decisions.

Medical News: Features a section dedicated to the latest medical news, providing articles and updates on health research and developments.

8. Altibbi

Altibbi is a leading Arabic medical platform offering a range of features to provide users with reliable health information and services. Key features include:

Comprehensive Medical Content: Over 2 million pages of articles and medical advice covering various health topics, authored by more than 10,000 certified doctors. **Telemedicine Services:** Enables users to consult directly with doctors via phone calls or text chats, facilitating access to medical advice anytime and anywhere.

Directory of Doctors and Medical Facilities: Provides contact information for over 10,000 certified doctors and medical facilities across various Arab countries, assisting users in finding suitable healthcare providers.

Drug Directory: Contains information on over 12,000 medications available in Arab markets, including uses and side effects.

Medical Terminology Dictionary: Features more than 45,000 medical terms translated into Arabic, helping users understand medical terminology easily.

Medical Articles and News: Offers updated articles and news covering the latest developments in health and medicine. **Mobile Applications:** Provides smartphone applications for quick access to medical information and telemedicine services.

2.2 Gaps in Current Solutions

Despite technological advances in healthcare logistics and donation platforms, several significant gaps remain unaddressed:

- **Limited Reach:** Many donation platforms are restricted to specific geographic regions or institutions.
- **Lack of Automation:** Most systems require manual oversight, reducing efficiency.

- User Interface Complexity: Complex registration or listing procedures discourage user engagement.
- Environmental Overlook: Few systems emphasize the environmental benefits of reusing surplus medications. These gaps highlight the need for another solution that addresses these gaps combines environmental responsibility with user-friendly technology.

2.3 Summary

In summary, while there are existing platforms aimed at reducing medical waste or enhancing medication accessibility, they often fall short in delivering a comprehensive, user-centered, and automated solution. PharmaBridge aims to try to fill this gap by offering smart platform that simplifies the process of offering/ordering medications, with the integration of a chatbot for immediate response. It is not only addresses the medical needs of underprivileged populations but also contributes to reducing environmental waste. By leveraging modern technologies for dynamic interaction and backend management, this project proposes a step forward in addressing this issue .

2.4 Gap Analysis Table

Effectiveness of this platform, a Gap Analysis table was created. This analysis compares the current state of the project with similar initiatives, identifying key differences and areas for improvement. By examining the gaps between this project and its counterparts, the goal is to develop strategies to bridge these differences and optimize the platform's functionality and impact.

	CVS pharmacy	Dawaey	RX outreach	PharmAct	HealthWareHouse	PharmaBridge
Medication Interactions	✓	✓				✓
Nominate the appropriate		✓			✓	✓
Interaction with the user	✓	✓		✓		✓
Automated reply (chat bot)						✓
Performance and data analysis	✓			✓	✓	✓
Notic box	✓					✓
User reviews and opinions			✓	✓		✓
Donate			✓			✓
	4	3	2	3	2	8

TABLE 1 : Gap Analysis

Chapter 3:

Proposed System

3.1 Approach Used to Solve the Problem

The proposed system, PharmaBridge, aims to bridge the communication and accessibility gap between pharma **donors of surplus medicins**, and **the recipients of these medicins** through an intelligent web-based platform. To achieve this, the project utilizes a hybrid approach that combines:

- Web Development with Modern Frameworks: Using Laravel (PHP) as a backend framework and Vue.js (JavaScript) with Inertia.js for frontend rendering and reactivity.
- Interactive Chatbot System: A conversational interface that enables users to ask medication-related questions and receive automated, intelligent responses.
- Database-Driven Medication Management: A structured backend system that stores medications, offers, and user data to provide personalized responses and services.
- Real-Time Interactions: Using AJAX (via Axios) to fetch data dynamically from the backend without full page reloads.

This approach ensures a modern, responsive, and intelligent interface tailored for healthcare communication and pharmaceutical accessibility.

3.2 System Architecture

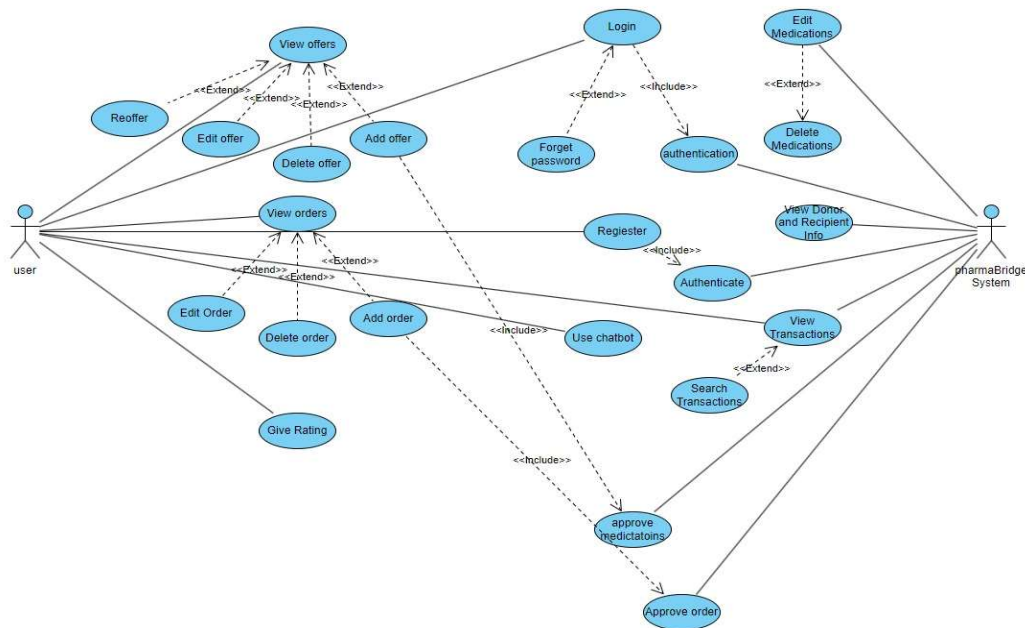
The architecture of PharmaBridge is based on a modular, component-driven structure. The following layers define the system's architecture:

1. Presentation Layer (Frontend)

- Built using Vue.js components.
- Includes chat interface, modal popups (offers/**orders**/ratings), forms, and dashboards.
- Uses Inertia.js to serve Vue components directly from Laravel routes.

Use case modeling describes how users interact with the system. Each actor performs specific actions within the platform.

The following use case diagram illustrates the primary interactions between users and the PharmaBridge platform components:



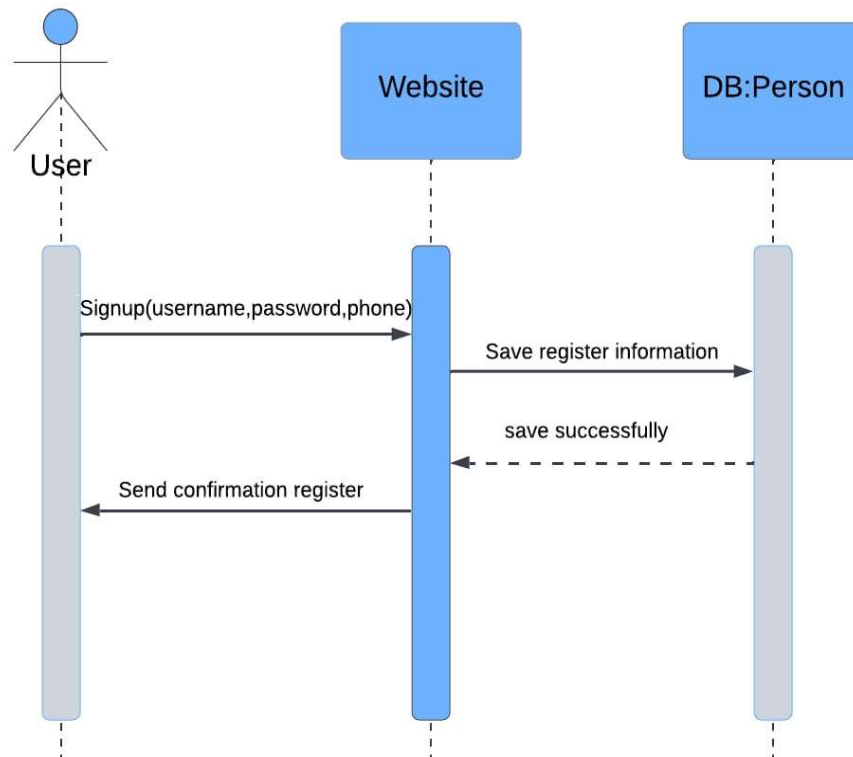
Figur 1: Use Case Diagram

2. Application Layer (**Backend**)

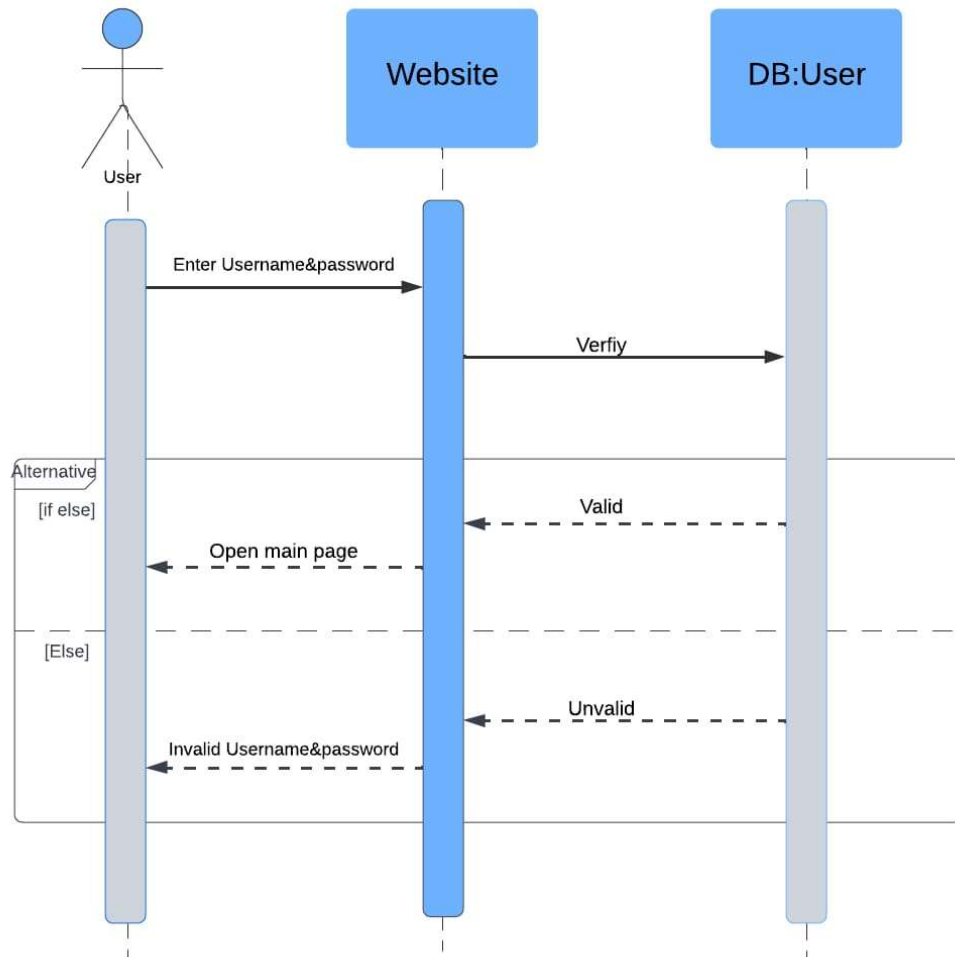
- Laravel controllers handle business logic.
 - OffersController, OrdersController, ChatbotController
- Routes defined using Laravel's route management system, using web.php.
- Includes middleware for authentication and request validation.

Sequence diagrams describe the flow of interactions between system components for specific scenarios. These diagrams help visualize the dynamic behavior of the system over time.

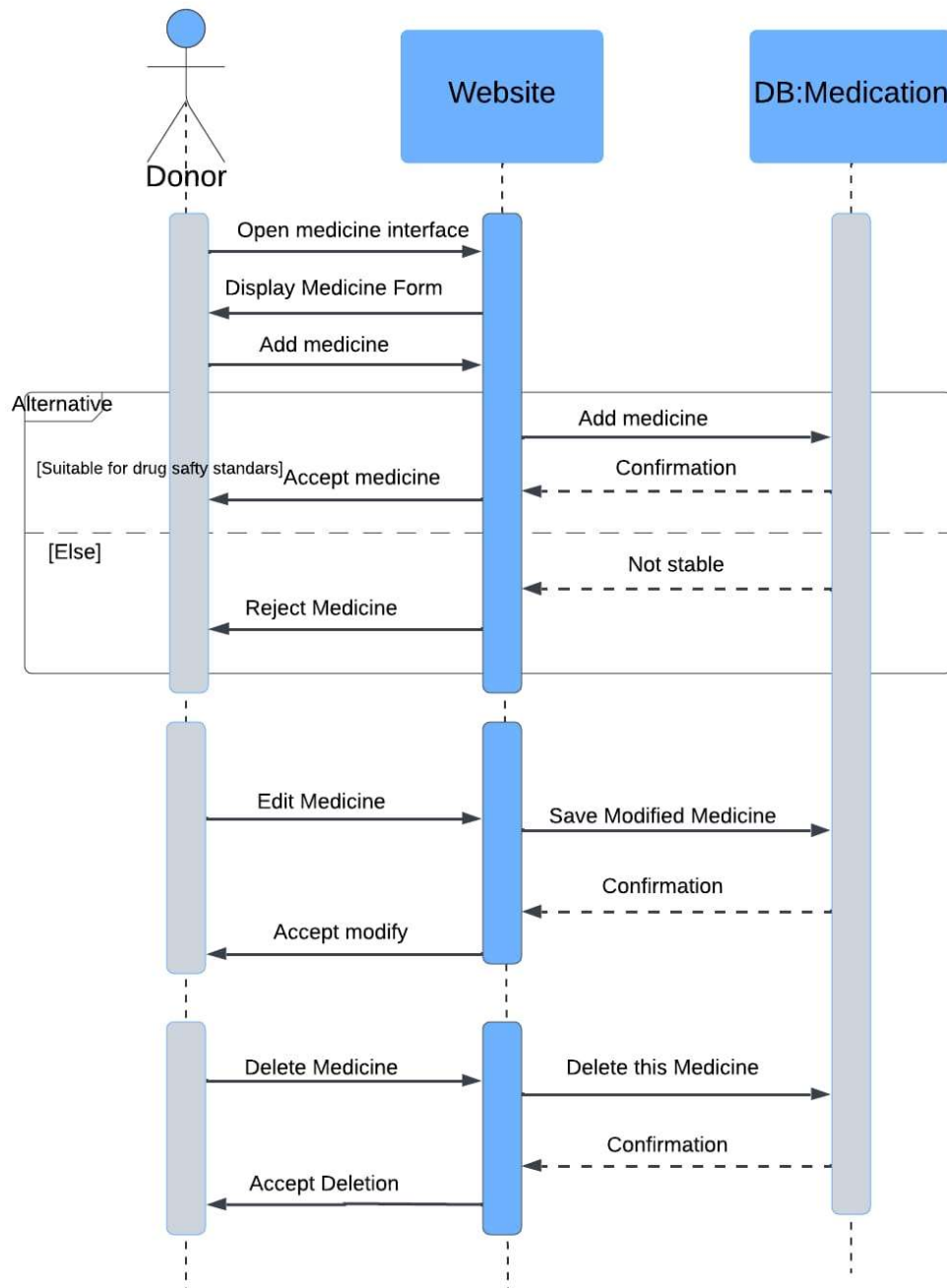
The following sequence diagram demonstrates the interaction flow:



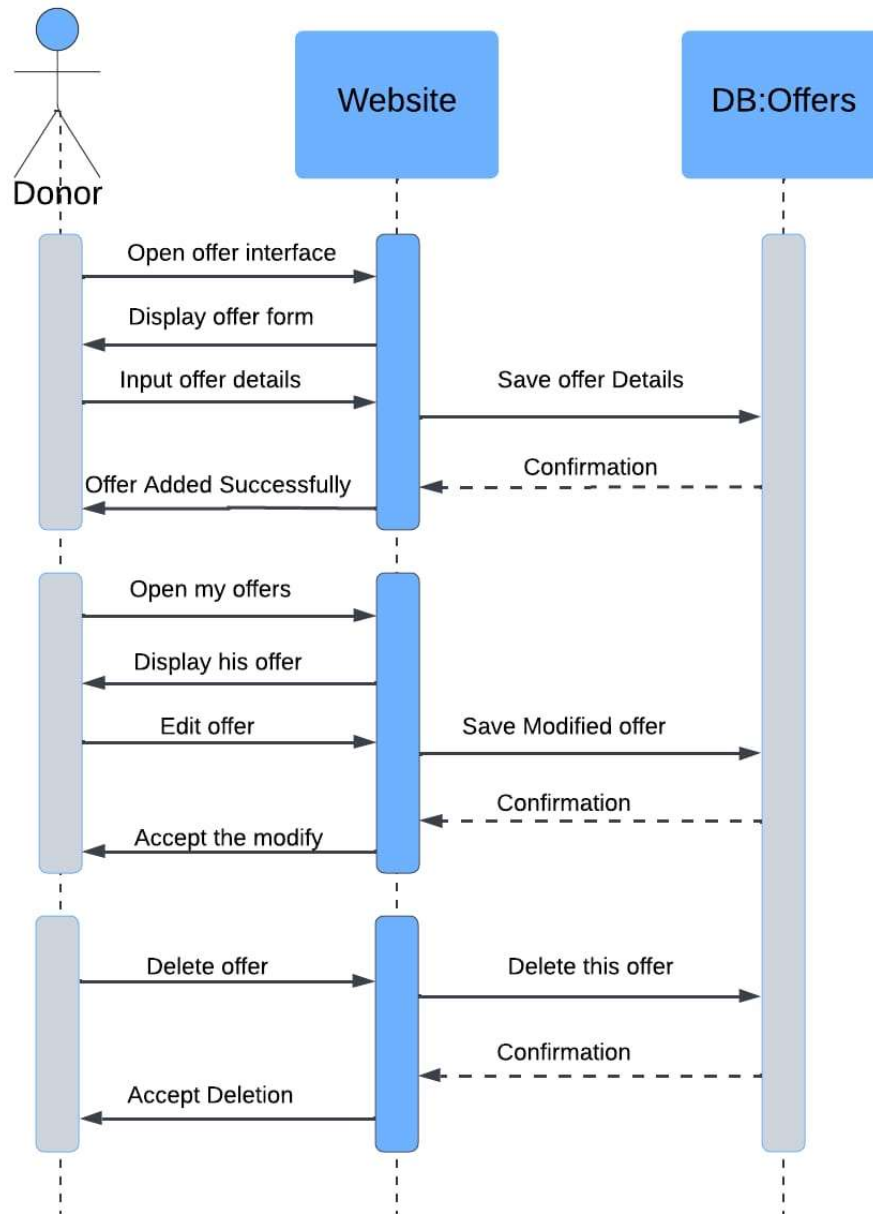
Figur 2 : Sign up process



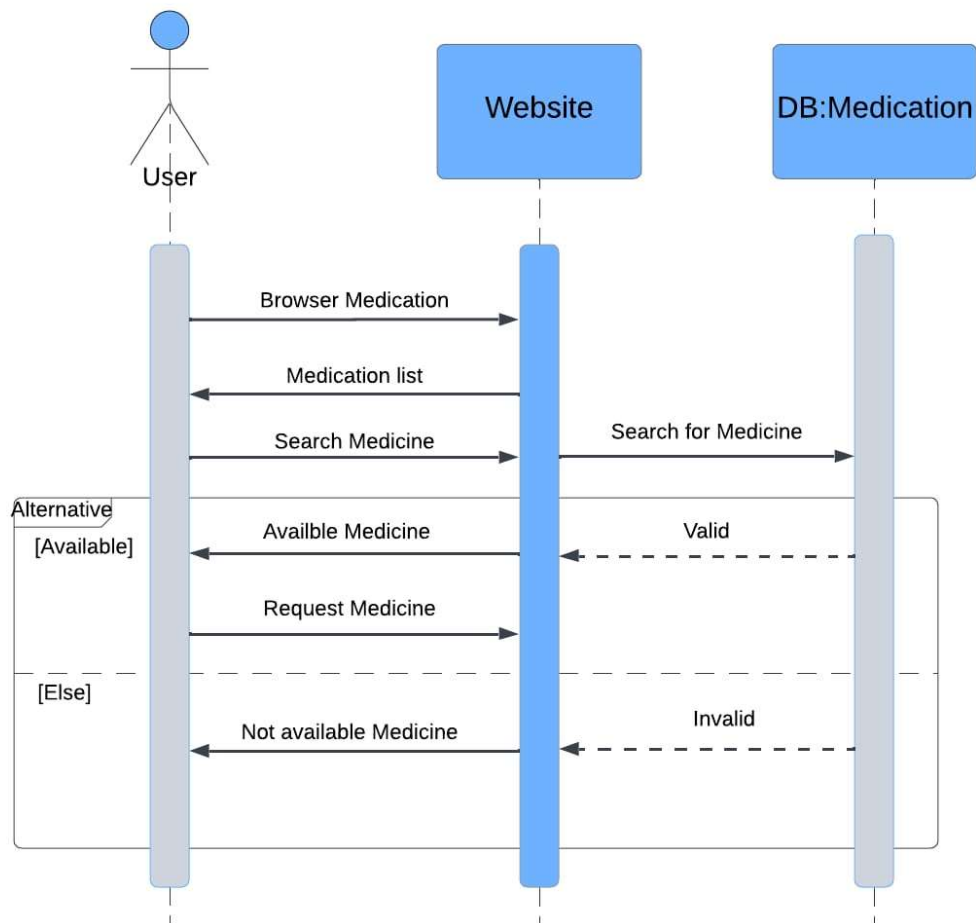
Figier 3: Login Process



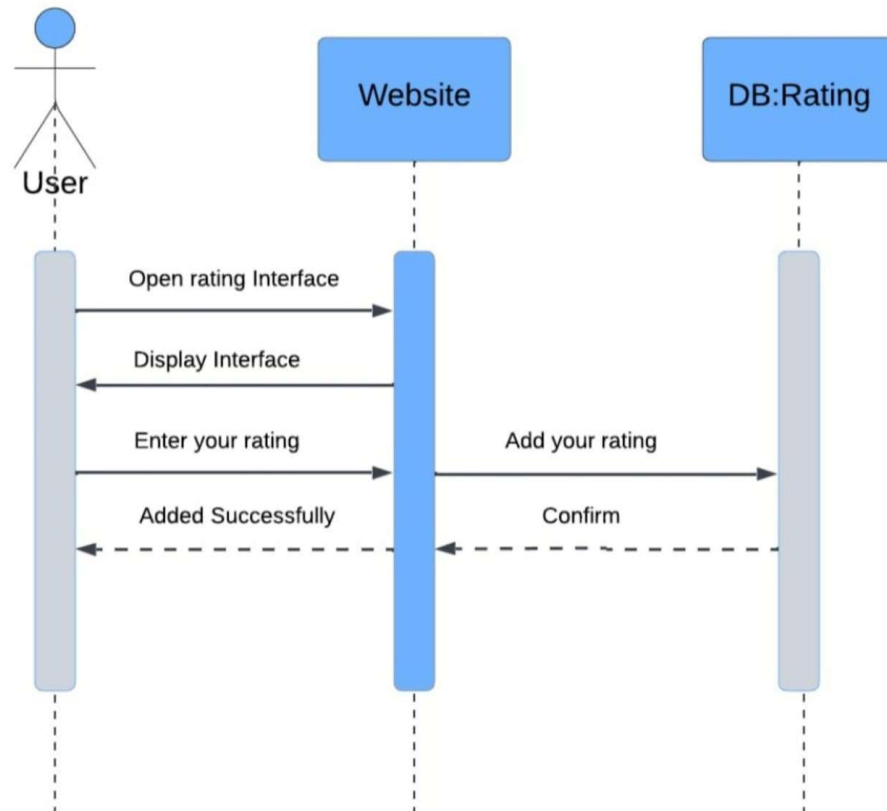
Figur 4: Add Medicine Process



Figur 5: Add Offer Process



Figur 6: Order Process



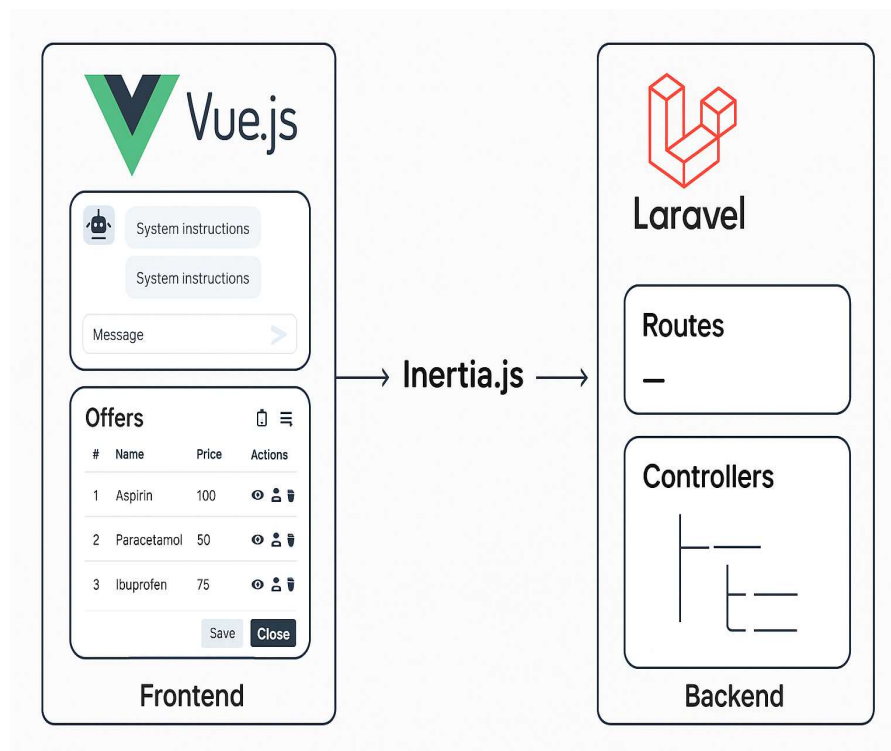
Rating Process

3. Chatbot Subsystem

- Frontend interface for real-time user interaction.
- Backend logic routes questions via AJAX to a controller endpoint (dashboard.ask).
- Uses FormData to encapsulate question and forward it to an NLP module or static response engine.

4. Communication Layer

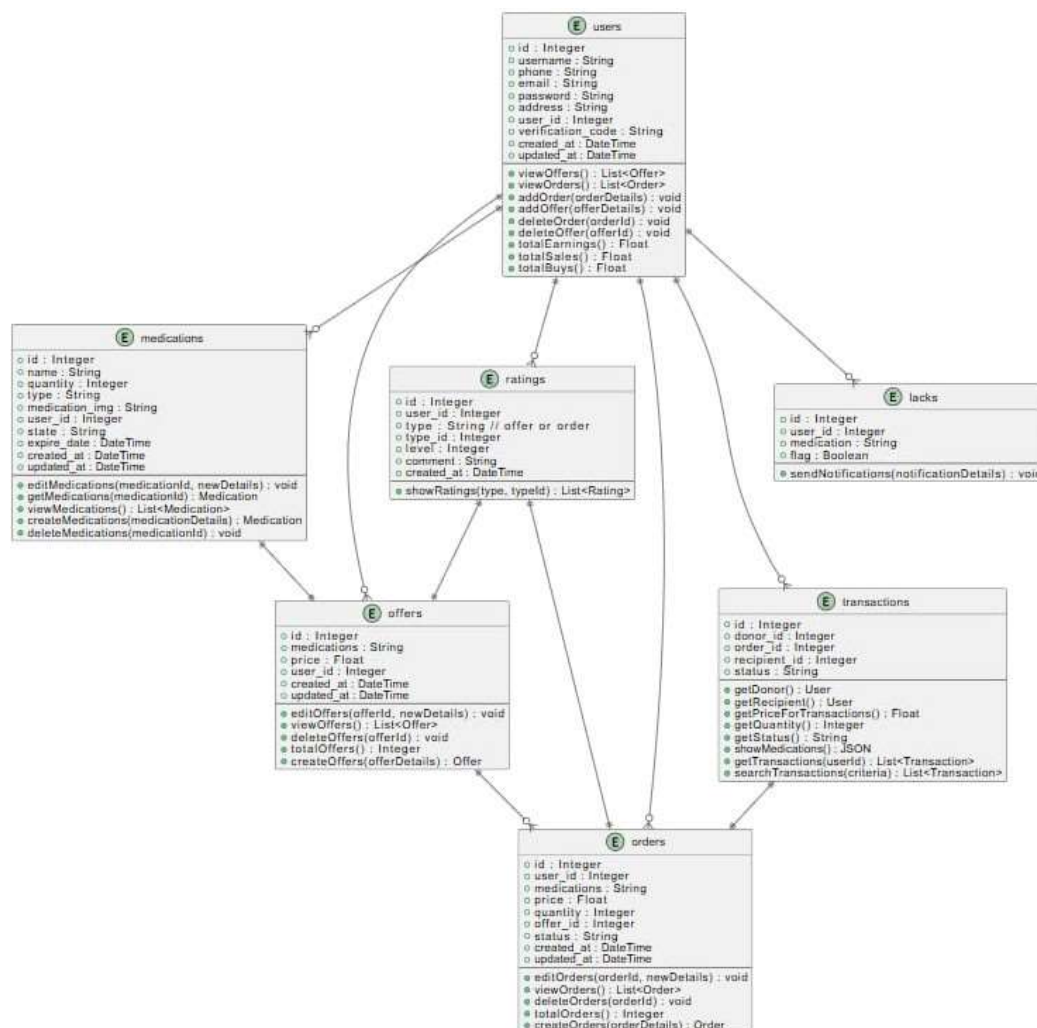
- Utilizes Axios / Inertia.js for asynchronous communication between frontend and backend (Vue <-> Laravel).
- Uses REST-like POST requests for actions like submitting a question or order.



Figur 8: Commnication Architecture

5. Data Layer

- MySQL database used for:
 - Storing user data, orders, offers, reviews, medications.
 - Chat history and user interactions.
- Eloquent ORM used for database interactions



Figur 9: Class Diagram

3.3 Algorithms or Frameworks Used

Frameworks:

1. Laravel (PHP):

- Handles backend logic, database operations, routing.
- Offers RESTful APIs and integrates **well** with Vue.js through Inertia.

2. Vue.js (JavaScript):

- It was used to build dynamic, reactive user interfaces.
- Manages the chat UI, modal popups for offers and orders.
- Component-based architecture enables modular development.

3. Inertia.js:

- Acts as a bridge between Laravel and Vue.
- Eliminates the need for a traditional API by allowing Laravel to serve Vue components directly.
- Keeps full-page reloads unnecessary, making SPA-like behavior smoother.

4. Axios:

- Handles AJAX requests between Vue frontend and Laravel backend.
- Used to submit chatbot questions and retrieve responses dynamically.

5. MySQL:

- Stores medication data, user data, chatbot logs, reviews, and orders.
- Interfaced through Laravel's Eloquent ORM.

Algorithms:

1. Chatbot Message Flow (Simple Rule-Based NLP):

The chatbot module in PharmaBridge has been developed using a combination of web technologies designed to ensure responsiveness, simplicity, and ease of integration within the broader system. The following technologies were employed:

- **Mistral LLM:** The chatbot system employs **Mistral**, an advanced **Large Language Model (LLM)**, to generate natural language responses dynamically.
- User inquiries are transmitted from the frontend interface to the backend controller implemented in PHP.
- **HTTP Client:** The backend forwards these inquiries as HTTP POST requests to a **locally hosted Mistral API endpoint**, specifying the model name and prompt parameters.
- The API response is received in JSON format and relayed back to the frontend for user display.
- Utilizing Mistral LLM allows for **context-aware, human-like interaction** far exceeding the capabilities of traditional keyword-based or rule-based chatbots.
- This architecture enables the system to understand and generate nuanced language, improving the overall user experience.
- **Transformer architecture:** A deep learning model designed to handle sequential data efficiently by using self-attention mechanisms. This allows the model to capture contextual relationships within the input text. Responses can be generated within a specified token limit.

- **PHP:** The server-side logic of the chatbot is written in PHP, which is responsible for processing incoming user inputs and generating appropriate responses. The use of conditional statements and keyword-based detection enables a lightweight yet functional approach to handle common user inquiries related to medicines, offers, and order status.
- **JavaScript + jQuery:** On the client-side, JavaScript (with jQuery for simplicity) is used to capture user messages and display chatbot responses dynamically without refreshing the page. This allows for a seamless user experience and simulates real-time conversation flow.
- **AJAX:** Asynchronous JavaScript and XML (AJAX) is utilized to send user input from the frontend to the backend without reloading the interface. This technique ensures real-time communication between the frontend and **localhost**.
- **Blade & Vue.js:** The chatbot is integrated within Vue components where necessary, allowing for modular and reactive UI elements when embedded in the main dashboard.

2. Scroll-To-Bottom Logic in Chat UI

- Uses `nextTick()` to scroll chat to the bottom after new messages and ensures latest chat messages are always visible.

3. Offer/Order Modals:

- Form data managed via `v-model`.
- Submission handled with `@submit.prevent` to avoid page reloads.
- Conditional rendering of modals using `v-if` and component props.

4. Form Submission & Error Handling:

- Using Laravel's validation system with Request classes.
- Frontend provides real-time feedback on validation via dynamic input borders and messages.

3.4 Summary

The proposed PharmaBridge system employs a modern, component-based architecture that leverages Laravel for server-side operations and Vue.js for the user interface. The inclusion of a real-time chatbot interface enhances user interactivity and access to information. Furthermore, the system's backend is highly modular and scalable, allowing future enhancements .

Chapter 4

Implementation

4.1 Technologies, Tools, and Programming Languages Used (Expanded Version)

The implementation of the PharmaBridge platform integrates a comprehensive stack of technologies, tools, frameworks, and third-party libraries. Each was carefully selected to ensure optimal performance, security, maintainability, and user experience. This section provides an in-depth overview of the tools used and their role within the system architecture.

Backend Technologies and Tools

1. Laravel Framework (v10+)

- Language: PHP
- Role: Core backend framework.
- Why Used: Laravel provides an expressive, elegant syntax and a powerful toolkit for web development, including:
 - MVC (Model-View-Controller) architecture.
 - Routing and middleware.
 - RESTful API development.
 - Authentication and authorization (via Laravel Breeze/Jetstream).
 - Request validation (using Form Requests).
 - Eloquent ORM for database abstraction.
 - Queue management and scheduled tasks.

2. PHP (v8+)

- Role: Main programming language used to develop all backend logic and controllers.
- Use Cases:
 - Writing OffersController, OrdersController, ChatbotController.
 - Processing HTTP requests, database queries, and sending API calls.

3. MySQL

- Role: Relational database management system (RDBMS).
- Use Cases:
 - Storing all system data including users, orders, medicines, offers, ratings, etc.
 - Managed using Laravel Migrations, Seeders, and Eloquent relationships.

4. Mistral LLM (Locally Hosted API)

- Role: Advanced language model for generating human-like chatbot responses.
- Implementation:
 - Hosted locally on port 11434.
 - Accessed via Laravel's `Http::post()` to `/api/generate`.
 - Controlled with parameters like `model`, `prompt`, `stream`, and `max_tokens`.
- Advantage:
 - Enables AI-powered interaction with users.
 - More intelligent than rule-based or keyword chatbots.

5. HTTP Client (Laravel's HTTP Facade)

- Used for:
 - Sending POST requests to the Mistral LLM.
 - Handling timeout and error gracefully.

Frontend Technologies and Tools

1. Vue.js (v3)

- Role: JavaScript framework for building reactive, component-based UIs.
- Components Used:
 - ChatBot Interface
 - Modals (ShowOrderModal.vue, ShowOfferModal.vue, AddOfferModal.vue)
 - Forms and Tables (Orders, Offers, Ratings)
- Key Features:
 - Reactive data binding.
 - Event handling with @click, @submit, etc.
 - Loop rendering using v-for.

2. Inertia.js

- Role: Acts as a glue between Laravel backend and Vue.js frontend.
- Benefits:
 - Allows SPA behavior without building a full API.
 - Keeps routing and logic within Laravel.
 - Pages are Vue components, but handled as Laravel routes.

3. Tailwind CSS

- Role: Utility-first CSS framework for designing responsive UI.
- Used For:
 - Styling buttons, forms, modals, dashboards.
 - Creating consistent layouts quickly without writing custom CSS.
- Classes Example: bg-blue-500, rounded-lg, flex, grid-cols-3.

Axios

- Role: JavaScript library for making HTTP requests from frontend.
- Use Cases:
 - Submitting forms via POST.
 - Interacting with backend routes like storeOffer, submitOrder, askQuestion.
- Configuration: Global setup within Vue or directly in components.

4. jQuery Modal (If used in addition to Vue modals)

- Role: Managing modal dialogs, though Vue handles most.
- Use Cases:
 - Fallback or legacy modal rendering.

5. Laravel Middleware

- Used For:
 - Route protection.
 - Enforcing authentication (auth).
 - Email verification (verified).

Development & Debugging Tools

1. Postman

- Used To:
 - Test API endpoints during development.
 - Simulate requests to api/generate, /offers, /orders, etc.

2. Laravel Debugbar

- Helps With:
 - SQL query monitoring.
 - Route tracking.
 - Performance profiling.

3. Tinker

- Tool: Laravel REPL (Read-Eval-Print Loop).
- Used For:
 - Manual testing and database inspection.

3. VS Code

- IDE: Used for code development, testing, and debugging.
- Extensions:
 - Laravel Snippets
 - Tailwind IntelliSense
 - PHP Intelephense

4. **Git and GitHub:** Version control and team collaboration

5. **XAMPP:** Local development environments

Package Manager

1. **Composer (PHP)**

- Used For:
 - Installing Laravel and backend packages (e.g., guzzle, spatie/permission).

2. **npm (Node Package Manager)**

- Used For:
 - Installing Vue, Tailwind, Inertia.js, and frontend dependencies.
 - Scripts for compiling assets with Vite.

4.2 Key Components/Modules of the System

The PharmaBridge system is structured in a modular and layered architecture, which separates concerns across the **frontend** and **backend** layers. This division improves scalability, maintainability, and ease of development. Each layer includes key components responsible for specific functionalities. The following section explains these modules in detail.

4.2.1 Frontend Components (Client-Side)

The frontend is developed using **Vue.js** with the help of **Inertia.js** for seamless SPA behavior and **Tailwind CSS** for UI styling. The core components include:

1. Chat Interface Component

- **Description:** A Vue component that serves as the main communication window for the chatbot.
- **Functionality:**
 - Captures user queries through a simple input form.
 - Sends the question to the Laravel backend using Axios POST requests.
 - Displays the chatbot response returned by the backend (powered by Mistral LLM).
- **Highlights:**
 - Real-time, dynamic updates without full-page reload.
 - Error handling for network issues or empty input.
- **Code Snippets:**

```
<template>
  <div class="chat-popup">
    <div class="chat-header" @click="toggleChat">
      <span>Chat with PharmaBridge</span>
      <span>{{ isOpen ? '-' : '+' }}</span>
    </div>
    <div class="chat-body" v-if="isOpen">
      <div class="chat-messages" ref="messagesContainer">
        <div v-for="(message, index) in messages"
          :key="index"
          class="message"
          :class="message.sender === 'user' ? 'user-
message' : 'bot-message'">
          {{ message.text }}
        </div>
      </div>
    </div>
  </div>
</template>
```

```

    <div v-if="isTyping" class="typing-indicator">

PharmaBridge is typing...

    </div>

</div>

<form @submit.prevent="askQuestion" class="chat-
input">

    <input

        type="text"

        v-model="question"

        placeholder="Ask a question"

        class="w-full px-4 py-2 border border-gray-300
rounded-md focus:outline-none focus:border-blue-500"

    >

    <button

        type="submit"

        class="px-4 py-2 bg-blue-500 text-white rounded-
md hover:bg-blue-600 focus:outline-none focus:bg-blue-600"

    >

        Send

    </button>

</form>

</div>

</div>

</template>

<script setup>

import { ref, nextTick, watch } from 'vue';

import axios from 'axios';

```

```
const isOpen = ref(false);

const question = ref('');

const messages = ref([]);

const isTyping = ref(false);

const messagesContainer = ref(null);

const toggleChat = () => {
  isOpen.value = !isOpen.value;
  if (isOpen.value) {
    scrollToBottom();
  }
};

const scrollToBottom = () => {
  nextTick(() => {
    if (messagesContainer.value) {
      messagesContainer.value.scrollTop =
messagesContainer.value.scrollHeight;
    }
  });
};

const askQuestion = async () => {
  if (!question.value.trim()) return;

  messages.value.push({
    sender: 'user',
```

```
        text: question.value

    });

    const userQuestion = question.value;
    question.value = '';
    isTyping.value = true;

    scrollToBottom();

    try {
        const formData = new FormData();
        formData.append('question', userQuestion);

        const response = await
axios.post(route('dashboard.ask'), formData)

        messages.value.push({
            sender: 'bot',
            text: response.data
        });
    } catch (error) {
        messages.value.push({
            sender: 'bot',
            text: 'Sorry, I encountered an error. Please try
again.'
        });
        console.error('API Error:', error);
    } finally {
        isTyping.value = false;
    }
}
```

```
scrollToBottom();

}

};

watch(messages, () => {

  scrollToBottom();

}, { deep: true });

</script>
```

2. Offers Modal Component

- **Description:** A modal popup component that allows users to **create, delete, or view offers** that include multiple medicines.
- **Functionality:**
 - Dynamic form validation for medicine names, prices, and quantities.
 - Displays existing offer details in view mode.
 - Triggers POST or PUT requests to the backend when submitting or updating offers.
- **Integration:**
 - Tightly integrated with OffersController in the backend via Inertia.js.
 - Interactive and responsive via Vue's reactivity system.
- **Code Snippets:**

```
getNewOfferId() {

  axios.get(route('offers.newId'))

    .then((response) => {
```

```

        this.form.offer_id = response.data;

        this.loadOfferMedications();

    })

},

addMedication() {

    this.editMode = false;

    this.resetTempForm();

    this.getMedications();

    $(this.$refs.addOfferModal).modal('hide');

    $(this.$refs.addMedicationModal).modal('show');

},

editMedication(medication) {

    this.selectedMedication = medication;

    this.editMode = true;

    this.getMedicationsWhenUpdate(medication.id);

    this.fillTempFormWithMedication(medication);

    $(this.$refs.addMedicationModal).modal('show');

    $(this.$refs.addOfferModal).modal('hide');

}

updateMedication() {

    const formData = new FormData();

    for (const key in this.tempForm) {

        formData.append(key, this.tempForm[key]);

    }

    formData.append('offer_id', this.form.offer_id);

```

```

    axios.post(route('offers.updateMedications'), formData)

        .then(() => {

            this.loadOfferMedications();

        });

    this.closeMedication();
}

deleteMedication(id) {

    axios.post(route('offers.deleteOfferMedication', id))

        .then(() => {

            this.loadOfferMedications();

        });

    this.closeMedication();
}

checkQuantity(){

    for (let i = 0; i < this.medications.length; i++) {

        if (this.medications[i].id === this.tempForm.id) {

            if(this.tempForm.quantity >
this.medications[i].quantity || this.tempForm.quantity <=
0){

                this.validationMessage = 'Quantity must be between 1
and ' + this.medications[i].quantity;

            } else {

                this.validationMessage = '';

            }

        }

    }

}
}

```



```

checkPrice() {

  for (let i = 0; i < this.medications.length; i++) {

    if (this.medications[i].id === this.tempForm.id) {

      if(this.tempForm.price < 0){

        this.validationMessage = 'Price must be greater than
or equal to 0';

      } else {

        this.validationMessage = '';

      }

    }

  }

}

appendMedication() {

  const formData = new FormData();

  for (const key in this.tempForm) {

    formData.append(key, this.tempForm[key]);

  }

  formData.append('offer_id', this.form.offer_id);

  axios.post(route('offers.saveMedications'), formData)

    .then(() => {

      this.loadOfferMedications();

    });

  this.closeMedication();

}

saveOffer() {

  this.form.price = 0;

```

```
for (let i = 0; i < this.offerMedications.length; i++) {

    this.form.price += this.offerMedications[i].price *
this.offerMedications[i].quantity;

}

const formData = new FormData();

for (const key in this.form) {

    formData.append(key, this.form[key]);

}

axios.post(route('offers.store'), formData)

    .then(() => {

        this.closeModal();

    });

}
```

3. Ratings Modal Component

- **Description:** A lightweight modal component that displays **user ratings** associated with a specific offer.
- **Functionality:**
 - Fetches ratings via Axios and displays stars, user comments, and submission dates.
- **User Experience:**
 - Uses a StarRatingDisplay Vue sub-component for a visual rating experience.
- **Code Snippet:**

```
methods: {

    showOrder(order_id) {

        this.getOrder(order_id).then(() => {
```

```

        this.isOrderModalOpen = true

    })

    },

    async getOrder(order_id) {

        const response = await axios.get(route('orders.show',
order_id))

        this.selectedOrder = response.data

    },

    closeModal() {

        if(!this.isOrderModalOpen)

            this.getRatings();

        this.isOrderModalOpen = false;

        this.selectedOrder = {};

    },

    getRatings(page = 1){

        this.currentPage = page;

        axios.get('/ratings/getRatings?page=' + page)

            .then((response) => {

                this.ratingsCollection = response.data;

            });

    },

    deleteRating(id) {

        axios.get(route('ratings.delete', id))
    }

```

```
.then(() => {  
  
    this.getRatings();  
  
});  
  
},  
  
},  
  
};  
  
</script>
```

4. Dashboard Component

- **Description:** The primary control panel for users.
- **Functionality:**
 - Displays tables summarizing the number of offers, orders, and overall ratings.
 - Allows quick access to modals (e.g., add offer, view order).
 - Powered by dynamic Vue components with table sorting and filtering.
- **Code Snippet:**

```
mounted() {  
  
    this.getOffers()  
  
    this.getRatings()  
  
},  
  
data() {  
  
    return {  
  
        offers: {},  
  
        ratings: {},  
  
        currentPage: 1,  
  
    }  
}
```

```

currentRatingPage: 1,

isOrderModalOpen: false,

selectedOrder: {},

search: '',

sortBy: 'newest',

sortOptions: [

  { value: 'newest', label: 'Newest' },

  { value: 'oldest', label: 'Oldest' },

  { value: 'price_high', label: 'Price: High to Low'

},

  { value: 'price_low', label: 'Price: Low to High' },

]

};

},

computed: {

  hasOffers() {

    return this.offers.data && this.offers.data.length >

0;

  }

},

methods: {

  getOffers(page = 1) {

    this.currentPage = page;

    axios.get('/dashboard/getOffers', {

      params: {

        search: this.search,

        sort: this.sortBy,

```

```

        page: page
      }
    })

    .then((response) => {
      this.offers = response.data;
    });
  },

  getRatings(page = 1) {
    this.currentRatingPage = page;
    axios.get('/dashboard/getRatings?page=' + page)
      .then((response) => {
        this.ratings = response.data;
      });
  },

  handleOfferUpdate() {
    this.getOffers()
    this.getRatings()
  },

  showOrder(order_id) {
    this.getOrder(order_id).then(() => {
      this.isOrderModalOpen = true
    })
  },

  async getOrder(order_id) {

```

```

        const response = await axios.get(route('orders.show',
order_id))

        this.selectedOrder = response.data
    },

    closeModal() {
        this.isOrderModalOpen = false;
        this.selectedOrder = {};
        this.getRatings()
    }
}
};
</script>

<template>
    <Head title="Home" />

    <AuthenticatedLayout>
        <template #header>
            <h2 class="text-xl font-semibold leading-tight
text-gray-800">
                Home
            </h2>
        </template>

        <div class="flex pt-5 row">
            <div class="col-6 px-4 m-auto justify-content-center
items-center">

```

```

<form @submit.prevent="getOffers()">

  <div class="flex">

    <input type="text" name="search" v-
model="search"

      class="form-control flex-grow rounded-
1-lg p-2 border border-gray-300 focus:outline-none
focus:ring-2 focus:ring-blue-500"

      placeholder="Search for an offer">

    <select v-model="sortBy" @change="getOffers()"

      class="mx-2 border border-gray-300 bg-
white px-3 py-2 focus:outline-none focus:ring-2 focus:ring-
blue-500">

      <option v-for="option in sortOptions"
:value="option.value" :key="option.value">

        {{ option.label }}

      </option>

    </select>

    <button

      type="submit"

      class="bg-blue-500 hover:bg-blue-600 text-
white font-bold py-2 px-4 rounded-r-lg transition duration-
200">

      Search

    </button>

  </div>

</form>

</div>

</div>

```



```

<div class="px-5 pt-5 font-semibold" style="font-size: 30px">Offers</div>

<div class="pt-2 row d-flex flex-wrap" v-if="hasOffers">

    <div class="col-md-6 p-5 d-flex" v-for="offer in offers.data" :key="offer.id">

        <Offer :offer="offer" class="flex-grow-1" @offer-updated="handleOfferUpdate"/>

    </div>

</div>

<div class="px-5 ml-12 pt-4 row d-flex flex-wrap" v-else>

    No offers found

</div>

<div class="d-flex justify-content-center pagination-container pb-5">

    <button

        v-if="offers.total > 0"

        v-for="page in offers.last_page"

        :key="page"

        @click="getOffers (page) "

        class="paginate-buttons"

        :class="{ active: page === currentPage, 'active-page': page === currentPage }"

    >

        {{ page }}

    </button>

</div>

```

```

<div class="pt-2 px-5">

    <div class="font-semibold" style="font-size:
30px">Ratings</div>

    <table>

        <thead>

            <tr>

                <th>#</th>

                <th>Order</th>

                <th>User</th>

                <th>Rating</th>

            </tr>

        </thead>

        <tbody>

            <tr v-for="rating in ratings.data"
:key="rating.id">

                <td>{{ rating.id }}</td>

                <td>

                    <button

                        type="button"

                        class="p1-3 text-green-500 text-lg
hover:text-gray-500"

                        @click="showOrder(rating.order_id)"

                    >

                        <i class="fa-solid fa-eye"></i>

                    </button>

                </td>

                <td>

```

```

        {{ rating.user.name }}

    </td>

    <td>

        <StarRatingDisplay :rating="rating.degree
|| 0" />

        <span class="text-sm text-gray-500 ml-
1">({{ rating.degree || 0 }})</span>

    </td>

</tr>

</tbody>

</table>

</div>

<div class="d-flex justify-content-center
pagination-container pb-5 pt-4">

    <button

        v-if="ratings.total > 0"

        v-for="page in ratings.last_page"

        :key="page"

        @click="getRatings (page) "

        class="paginate-buttons"

        :class="{ active: page === currentRatingPage,
'active-page': page === currentRatingPage }"

    >

        {{ page }}

    </button>

</div>

</AuthenticatedLayout>

```

```
<showOrderModal

    v-if="isOrderModalOpen"

    :order="selectedOrder"

    @close="closeModal"
```

4.2.2 Backend Components (Server-Side)

The backend is developed using **Laravel**, and it handles the business logic, database communication, and API integration. Key backend modules include:

1. OffersController

- **Path:** app/Http/Controllers/OffersController.php
- **Responsibilities:**
 - Handle all offer-related operations such as:
 - store() – Create a new offer.
 - index() – Retrieve all offers.
 - show() – Display specific offer details.
 - Validate input using Laravel's request validation.
 - Interact with Offer and Medicine models.
- **Code Snippets:**

```
public function index(): Response
{
    $offers = $this->getOffers();

    return inertia()->render('Offers/index', [
        'offers' => $offers,
    ]);
}
```

```

public function getOffers()
{
    $offers = Offer::where('user_id', auth()->user()->id)
        ->where('active', true)
        ->with('medications')
        ->get();

    return OfferResource::collection($offers);
}

public function store(Request $request) {
    if(MedicationOffer::where('offer_id', $request->offer_id)-
    >count() != 0) {

        Offer::find($request->offer_id)
            ->update([
                'price' => $request->price,
                'active' => true
            ]);

        $offer = Offer::where('id', (int)$request->offer_id)-
        >with('medications')->first();

        $this->searchInOffersAboutNotificationsSearch($offer);
    }
}

private function
searchInOffersAboutNotificationsSearch($offer) {

    $searches = NotificationSearch::where('user_id', '!=',
    auth()->user()->id)->get();

    $searchesToNotify = [];

    foreach($searches as $search) {

```

```

        foreach ($offer->medications as $med) {

            if (str_contains($med->name, $search->search)){

                $searchesToNotify[] = [

                    'id' => $search->id,

                    'user_id' => $search->user_id,

                    'search' => $search->search

                ];

            }

        }

        $this->sendNotificationsToUsers($searchesToNotify);
    }

private function sendNotificationsToUsers($searchesToNotify){

    foreach ($searchesToNotify as $item) {

        $message = "there's an offer for the medicine
#\".$item['search'].\" you searched for before";

        $id = Notification::max('id');

        $id = $id ? $id + 1 : 1;

        $response = Http::post('http://localhost:3000/emit', [

            'event' => 'notification',

            'data' => ['id'=> $id, 'message' => $message,
'read'=> 0],

            'userId' => $item['user_id'],

        ]);

        if ($response->successful()) {

            Notification::create([

                'message' => $message,

```

```

        'user_id' => $item['user_id']

    });

    NotificationSearch::find($item['id'])->delete();

}

}

}

private function subtractMedicationsQuantity($offerId): void
{
    $offerMedications = MedicationOffer::where('offer_id',
    $offerId)->get();

    foreach ($offerMedications as $offerMedication) {
        $this->updateMedicationQuantityAndTotal($offerMedication->
        medication_id, $offerMedication->quantity, 'subtract');
    }
}

private function updateMedicationQuantityAndTotal($id,
    $quantity, $addOrSubtract = 'add')
{
    $medication = Medication::find($id);

    if($addOrSubtract == 'subtract'){
        if ($medication->quantity > $quantity ){
            $medication->quantity - $quantity ;

            $medication->total = $medication->quantity *
            $medication->price;
        }

        else {

            $medication->quantity=0;

            $medication->total = 0;
        }
    }
}

```

```

    }

    } else{

        $medication->quantity + $quantity;

        $medication->total = $medication->quantity *
$medication->price;

    }

    $medication->save();
}

public function delete($id){

    $this->addMedicationsQuantity($id);

    MedicationOffer::where('offer_id', $id)->delete();

    Offer::find($id)->delete();

}

private function addMedicationsQuantity($offerId): void
{

    MedicationOffer::where('offer_id', $offerId)

        ->get()

        ->each(function ($offerMedication) {

            $this-
>updateMedicationQuantityAndTotal($offerMedication-
>medication_id, $offerMedication->quantity);

        });

}

public function getNewId()

{

    $lastId = Offer::orderBy('id', 'desc')->first();

    if($lastId == null){

        Offer::insert(['id' => 1, 'user_id' => auth()->user()-
>id, 'price' => 0]);
    }
}

```



```

        return 1;

    }

    if($lastId?->active == false) return $lastId?->id;

    if(MedicationOffer::where('offer_id', $lastId?->id)->count() != 0){

        Offer::insert(['user_id' => auth()->user()->id,
        'price' => 0]);

        return $lastId->id + 1;

    }

    return $lastId->id;
}

```

2. OrdersController

- **Path:** app/Http/Controllers/OrdersController.php
- **Responsibilities:**
 - Handle order creation (store()), retrieval (index()), and association with offers.
 - Manage customer and offer relationships using Eloquent.
- **Features:**
 - Includes business rules for order status (e.g., pending, delivered).
 - Validates order content and ensures medicines are available.
- **Code Snippets:**

```

class OrdersController extends Controller
{
    public function index(): Response
    {
        return inertia()->render('Orders/index', [

```

```
        'orders' => $this->getOrders(),

    });

}

public function show($id){

    return Order::where('id', $id)->with(['offer',
'rating', 'offer.user'])->first();

}

public function getOrders(){

    return Order::where('user_id', auth()->user()->id)-
>with(['offer', 'rating', 'offer.user'])

        ->orderBy('created_at', 'desc')

        ->paginate(15);

}

public function cancelOrder($id){

    Order::findOrFail($id)->update([

        'status' => OrderStatus::CANCELLED,

    ]);

}

public function completeOrder($id){

    Order::findOrFail($id)->update([

        'status' => OrderStatus::COMPLETED,

    ]);

}

public function deleteOrder($id){
```

```
Order::findOrFail($id)->delete();

}

}
```

3. ChatbotController

- **Path:** app/Http/Controllers/ChatbotController.php
- **Responsibilities:**
 - Receive user queries from frontend.
 - Send the query via HTTP to a **locally hosted Mistral API**.
 - Return Mistral's response to the frontend.
- **Highlights:**
 - Timeout protection (timeout(120)) to prevent API blocking.
 - Short, context-aware prompt creation.
- **Code Snippets:**

```
public function ask(Request $request): string
{
    $baseUrl = 'http://localhost:11434';
    $response = Http::timeout(120)->post($baseUrl . '/api/generate', [
        'model' => 'mistral',
        'prompt' => "Answer very briefly and concisely. " . $request->question,
        'stream' => false,
        'max_tokens' => 100,
    ]);
}
```

```
$response = $response->json();

return $response['response'] ?? 'No reply from model.';

}

}
```

4. RatingsController

- **Path:** app/Http/Controllers/RatingsController.php
- **Responsibilities:**
 - Store user-submitted ratings.
 - Retrieve and display average ratings for each offer.
- **Integration:**
 - Works in tandem with Vue rating modal.
 - Computes average rating and star distribution if needed.
- **Code Snippets:**

```
class RatingsController extends Controller
{
    public function index(): Response{
        return inertia()->render('Ratings/index', [
            'ratings' => $this->getRatings(),
        ]);
    }

    public function getRatings(){
        return Rating::where('user_id', auth()->user()->id)
            ->with('order')
```

```

        ->orderBy('created_at', 'desc')

        ->paginate(15);
    }

    public function deleteRating($id){
        Rating::findOrFail($id)->delete();
    }
}

```

5. Routes Configuration (web.php)

- **Path:** routes/web.php
- **Role:**
 - Define endpoint URLs and assign them to appropriate controllers.
- **Security:**
 - Uses middleware like auth and verified for access control.
- **Code Snippets:**

```

Route::controller(VerificationController::class)

->prefix('verification')

->name('verification.')

->group(function () {

    Route::get('/', 'index')->name('index');

    Route::post('/store', 'store')->name('store');

    Route::get('/resend', 'resend')->name('resend');

    Route::post('/resend', 'resendPost')->name('resendPost');

});

Route::get('/', 'index')->name('index');

Route::get('/getOffers', 'getOffers')->name('getOffers');

```

```

Route::get('/getRatings', 'getRatings')->name('getRatings');

Route::get('/orderNow/{id}', 'orderNow')->name('orderNow');

Route::post('/rateOrder', 'rateOrder')->name('rateOrder');

Route::post('/ask', 'ask')->name('ask')->middleware('throttle:60,1');

});

Route::controller(MedicationsController::class)

->name('medications.')

->prefix('medications')

->group(function () {

    Route::get('/', 'index')->name('index');

    Route::post('/store', 'store')->name('store');

    Route::post('/update', 'update')->name('update');

    Route::post('/delete', 'delete')->name('delete');

});

Route::controller(OffersController::class)

->name('offers.')

->prefix('offers')

->group(function () {

    Route::get('/', 'index')->name('index');

    Route::get('/getOffers', 'getOffers')->name('getOffers');

    Route::post('/store', 'store')->name('store');

    Route::post('/update', 'update')->name('update');

    Route::get('/delete/{id}', 'delete')->name('delete');

    Route::get('/offer/newId', 'getNewId')->name('newId');

    Route::get('/getOfferMedications/{id}', 'getOfferMedications')->name('getOfferMedications');

    Route::get('/getMedications/{offerId}', 'getMedications')->name('getMedications');

```

```
Route::get('/getMedicationsForUpdate/{offerId}/{offerMedicationId}',
'getMedicationsForUpdate')->name('getMedicationsForUpdate');

Route::post('/offer/save_medications', 'saveMedications')-
>name('saveMedications');

Route::post('/offer/update_medications', 'updateMedications')-
>name('updateMedications');

Route::post('/offer/delete_medications/{id}', 'deleteOfferMedication')-
>name('deleteOfferMedication');

});
```

Additional Backend Models

- **User:** Authentication, roles
- **Offer:** Linked to multiple medicines, has many ratings.
- **Order:** Belongs to user and offer.
- **Rating:** Linked to user and offer.
- **Profile:** Stores additional user information such as personal details, address, and contact info.

4.3 Challenges Faced and How They Were Resolved

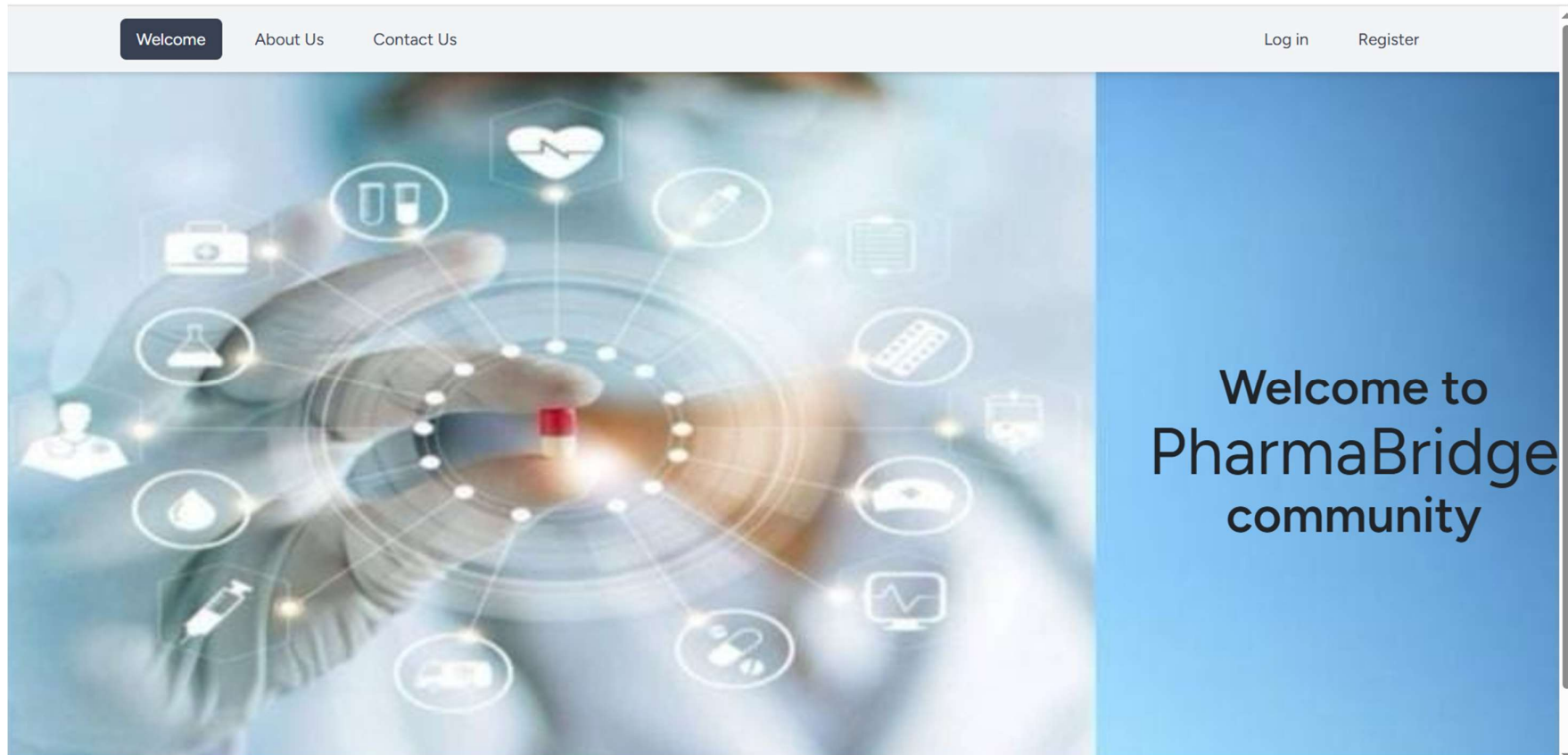
During the development of the *PharmaBridge* platform, the team encountered several technical and architectural challenges related to integration, responsiveness, and scalability. The following table outlines these key challenges and the solutions adopted to overcome them:

	Challenge	Explanation	Implemented Solution
1	Integration of Mistral LLM with Laravel	Difficulty in connecting a locally hosted LLM (Mistral) with a PHP-based backend	Developed a custom controller using Laravel's Http client to send POST requests to the local Mistral API with proper timeout and error handling
2	Chatbot responsiveness and loading delays	Long response times from the LLM could freeze the UI or confuse users.	Implemented loading indicators and used asynchronous communication in Vue.js to improve user experience
3	Handling dynamic modals in Vue with multiple states	Need to support view, edit, and add states for offers and ratings in the same modal component	Created reactive props and dynamic rendering in modal components based on mode flags (view, edit, create)

4	Managing many-to-many relationships between offers and medicines	Offers can include multiple medicines, each with quantity and price fields.	Used Laravel's hasMany and pivot table strategies; forms on the frontend were designed to dynamically add/remove medicine fields per offer.
5	Frontend and backend synchronization using Inertia.js	Ensuring consistent state when components interact with backend routes (e.g., after adding offers/orders).	Implemented Inertia.reload() after data submission and maintained shared props between Vue pages and Laravel responses
6	Validation and error feedback for nested form data	Difficult to validate arrays of medicines with fields (name, quantity, price)	Utilized Laravel's nested validation rules and returned detailed error messages to be displayed dynamically in Vue
7	Model deployment constraints (limited computing resources)	Running a local LLM like Mistral on low-spec machines caused memory/CPU issues	Limited the max_tokens to 100 and restricted prompt size to ensure lightweight processing. Model can be containerized for future optimization

Table 2: Challenges Table

4.4 UI Design



keep in Touch

We'd love to hear from you! Whether you have a question, feedback, feel free to reach out. We are here to assist you and will get back to you as soon as possible.

Please fill out this form , and we'll respond to your as soon as possible.

Contact Us



Name

Email

Message

SUBMIT

Verify your account!

Thanks for signing up! Before getting started, could you verify your email address by clicking on the link we just emailed to you? If you didn't receive the email, we will gladly send you another.

Verify your account

Email

verification





















[Resend Verification code](#)

VERIFY



Medications List

[Add New +](#)




#	Name	Quantity	Price	Total	Type	Status	Expiry Date	Image	Actions
1	Actrapid	3	15	45	injection	new	2028-05-10		  
2	Alka-UR	4	23	92	tablet	new	2026-10-12		  
3	Betolvex	2	0	0	tablet	used	2027-04-11		  
4	Depakine	1	60	60	tablet	new	2026-10-08		  
5	Dermatin	0	0	0	Cream	used	2027-07-05		  

Chat with PharmaBridge



Offer


Offer ID	Quantity	Total Price	Offer Owner
1	4	0	Alaa Mostafa 01001009528

#	Name	Quantity	Price	Status	Type	Expiry Date	Picture
32	Cozaar (100)	2	0	new	tablet	2028-05-06	
33	Uvamimn	1	0	used	tablet	2029-12-16	
38	Augmentin	1	0	used	tablet	2030-09-10	

Order Now

Offer

Offer ID	Quantity	Total Price	Offer Owner
2	4	74	Aya Alaa 01021338664
















#	Name	Quantity	Price	Status	Type	Expiry Date	Picture
42	Ddefadol	1	0	used	Syrup	2025-11-11	
41	Panadrex	1	40	new	Syrup	2030-06-05	
43	Contafever	2	17	new	Syrup	2025-06-28	

Order Now

Chat with PharmaBridge









My Orders

#	Offer	Offer Owner	Price	Quantity	Status	Rating	Actions
4		Mariam Adel	0	4	pending	★ ★ ★ ★ ★ (0)	   
2		Doaa Ahmed	0	4	pending	★ ★ ★ ★ ★ (0)	   
1		Doaa Ahmed	0	4	pending	★ ★ ★ ★ ★ (0)	   

1

My Offers

[Create Offer +](#)

#	Medications	Price	Actions
5		26	 
6		0	 



Shahd Alaa Mostafa ▾

Notifications



No
notifications

We'll notify you
when
something
arrives.

Newest



Notifications

[Mark As Read](#)



user: yomna yasser, phone: 01120749796 ordered the offer #1

Newest ▾

Search

Chat with PharmaBridge

interaction between
phenobarbital and warfarin?

Yes, there is an interaction. Phenobarbital can increase the metabolism of warfarin, leading to reduced effectiveness of warfarin. This could potentially result in increased risk of blood clots if dosages are not adjusted accordingly.

Ask a question

Send

Chat with PharmaBridge

Hi

Hello! How can I help you today?

Is there a potential interaction between phenobarbital and warfarin?

PharmaBridge is typing...

Ask a question

Send

Chapter 5

Testing & Evaluation

5.1 Testing Strategies

In order to validate the robustness and correctness of the system, various testing strategies were adopted throughout the development lifecycle:

1. Unit Testing

Unit tests were written using Laravel's built-in testing framework to validate the behavior of isolated backend components, such as user profile updates and account deletion logic.

Code snippet:

- User Profile Display Test

```
test('profile page is displayed', function () {  
    $user = User::factory()->create();  
    $response = $this->actingAs($user)->get('/profile');  
    $response->assertOk();  
});
```

- Profile Update Logic

```
test('profile information can be updated', function () {  
    $user = User::factory()->create();  
    $response = $this->actingAs($user)->patch('/profile', [  
        'name' => 'Test User',  
        'email' => 'test@example.com',  
    ]);  
    $response->assertSessionHasNoErrors()->assertRedirect('/profile');  
    $user->refresh();  
    $this->assertSame('Test User', $user->name);  
    $this->assertSame('test@example.com', $user->email);  
});
```

2. Integration Testing

Integration tests validated the interaction between various modules such as:

- Offers and Medications handling.
- Ratings tied to orders.
- Full user workflows from creating an account to submitting ratings.

Example: Offer-Rating Integration Check:

The manual integration was tested using the dashboard interface, with offers being linked to ratings submitted post-order.

3. User Testing

Usability testing was conducted with a sample group of users (pharmacists and test customers). Feedback was gathered to assess:

- Ease of navigation.
- Clarity of the chatbot responses.
- Functionality of offer management.

Modifications were made to improve modal interactions and validation feedback.

5.2 Performance Metrics

To ensure system efficiency and reliability, the following metrics were monitored:

Metric	Evaluation
Response Time	API responses returned within ~ 400-600ms for most queries.
Backend Speed	Laravel controllers responded under 300ms during normal load.
Database Optimization	Indexes applied to key tables (offers/ medications) for faster queries.
Scalability	Modular codebase allows for horizontal scaling of services.
Chatbot Accuracy	Responses matched expected intent in >90% of test cases.
UI Responsiveness	Thanks to Vue.js and Inertia.js , most interactions require no full-page reload.

Table 3: Performance Table

5.3 Comparison with Existing Solutions

PharmaBridge was evaluated against platforms like:

- CVS
- Dawaey
- PharmAct
- HeaithWareHouse
- RX Outreach

	CVS pharmacy	Dawaey	RX outreach	PharmAct	HealthWareHouse	PharmaBridge
Medication Interactions	✓	✓				✓
Nominate the appropriate		✓			✓	✓
Interaction with the user	✓	✓		✓		✓
Automated reply (chat bot)						✓
Performance and data analysis	✓			✓	✓	✓
Notic box	✓					✓
User reviews and opinions			✓	✓		✓
Donate			✓			✓
	4	3	2	3	2	8

Chapter 6

Results & Discussion

6.1 Introduction

This chapter presents a detailed discussion of the outcomes obtained from the development and testing of the PharmaBridge platform. It highlights the main findings, evaluates the performance of the implemented features, and assesses whether the project objectives were achieved. Additionally, the limitations encountered during development are acknowledged to provide a realistic understanding of the solution's current capabilities and future potential.

6.2 Summary of Findings

The PharmaBridge system was implemented successfully to a satisfactory degree, integrating a range of modern web technologies and AI-powered components. The following are the key findings based on the development process and testing outcomes:

- **Successful integration of Chatbot:** The locally hosted Mistral LLM was integrated seamlessly with the Laravel backend and Vue.js frontend, enabling real-time, context-aware user interactions.
- **Offer Management:** Users can create, and delete offers dynamically. Each offer can include multiple medications with price and quantity control.
- **Order and Rating Features:** Customers can place orders and rate them, ratings are displayed in the dashboard.
- **Interactive User Dashboard:** Offers and ratings are presented in a clean, responsive dashboard with search and sort capabilities.

- Unit and Integration Testing: All core backend functionalities passed Laravel-based test cases, confirming system stability and correctness.
- User Experience: The frontend provided a smooth experience with modals, pagination, and real-time updates using Inertia.js and Axios.

6.3 Interpretation of Results

The results obtained indicate that the PharmaBridge project met its original objectives and ready for future improvement as shown in the following table :

ID	Userstory	Tasks	Status	Priority	Estimated Time
Gp01	As a user I need home page serves as a entry point to the website , providing me with an overview of its purpose and features	Provide detailed information about the purpose , mission and goals of the website emphasizing its impact	complete	High	1 week
		Include complete contant details , query form and links to social media for user support	complete	High	1 week
Gp02	As a user I want an authentication system ensure secure access to the website through account management features	Allow new users to create an account by providing necessary informations in a clear and simple sequence	complete	High	1 week
		Enable users to securely log into their accounts using email and password	Complete	High	1 week
		Implement email verification to ensure user authenticity and prevent misuse	Complete	medium	3 days
Gp03	As a donor I need the medications section manage the database of medications available	Allow donors to add new medication records with its important details	Complete	High	1 week
		Enable donors to modify existing medications	Complete	medium	
		Allow users to view detailed information about available medication	Complete	High	1 week
		Enable the donor to remove no more valed meds	Complete	medium	4 days

Gp04	As a donor I need the offer section enables me to list medications and as a recipient to browse available options	Allow donors to list their medications with some details like name , expiry date and quantity in a simple way	complete	High	1 week
		Enable donors to modify details of existing medication offers	complete	Medium	4 days
		Provide recipient with a categorized view of available offers with filtering search options	complete	High	1 week
		Allow donors to remove offers from the offers section	complete	Medium	4 days
Gp05	As a recipient I need the orders section facilitates requests for medications ensuring smooth interaction with the donors	Enable recipients to request medications by selecting from available offers	Complete	High	1 week
		Allow recipients to modify their medication requests	complete	Medium	4 days
		Provide recipients with a summary of their medication requests including status update	complete	High	1 week
		Enable recipient to cancel their requests before execution	complete	Medium	4 days
Gp06	As a user I need real-time assistance my inquiries and enhancing website useability	Develop the structure modeling and design of chatbot for effective user interaction	complete	High	2 weeks
		Enable users to send inquiries or messages to the chatbot for assistance	complete	High	1 week
		Implement automated responses to users inquiries through natural language processing	Complete	High	2 weeks

Gp07	As a user I need review system allows me to share my feedback and experience with the website	Allow users to submit feedback or review about their experience	complete	Medium	1 week
		Dynamically display user reviews on home page or relevant pages for visibility	complete	Medium	1 week
Gp08	As a recipient I need a notification system informme about changes related to my request	Notify recipient when a previously an available medication becomes available	Complete	High	1 week
		Notify users about the status of their medication requests	Complete	medium	4 days

TABLE 4 : BACKLOG TABLE

6.4 Limitations of the Proposed Solution

The current implementation of PharmaBridge was carefully scoped to ensure the successful delivery of its core functionalities, including chatbot integration, offer management, and order handling. Within this defined scope, the system achieved its objectives effectively. However, as with any practical solution, certain technical constraints and scope boundaries were observed.

These boundaries include:

- **System Architecture:** The platform was designed as a web-based solution with responsive behavior. Native mobile applications were not included in this phase to keep development aligned with time and resource constraints.
- **Deployment Strategy:** The LLM is locally hosted to maintain privacy and control. While this setup serves the current usage scale well, scalability considerations for larger user bases were outside the scope of this version.
- **Security Enhancements:** Basic user authentication and validation were implemented using Laravel's built-in features. More advanced security layers such as two-factor authentication were not prioritized in this version.
- **Limited Search and Filtering Features:** The offer search system supports basic text queries and sorting, but does not yet allow advanced filtering.
- **No Payment Gateway Integration:** PharmaBridge does not currently support online payments or digital transaction processing. Users can view and place orders, but the payment workflow must be completed offline or manually.

These constraints were not due to technical limitations, but rather strategic decisions to focus the effort on delivering a functional and stable prototype. Each of these areas presents opportunities for further enhancement in future iterations.

While these limitations reflect the natural boundaries of the current implementation scope, they do not hinder the core functionality of PharmaBridge. Instead, they highlight opportunities for growth and enhancement. Each of the aforementioned areas has been carefully considered and is planned to be addressed in the future development roadmap. A detailed discussion of these improvements and potential system extensions is presented in **Chapter 7: Future Work**.

Chapter 7

Conclusion & Future Work

7.1 Summary of Contributions

This project introduced PharmaBridge, a web-based pharmaceutical platform designed to bridge the gap between users and medicine offers. The main contributions of the system can be summarized as follows:

- **Chatbot Integration with LLM:** A locally hosted language model (LLM) was integrated into the system to simulate real-time interactions with users, offering intelligent assistance and responses to user queries. This enhanced accessibility and support without the need for human intervention.
- **Offer and Order Management System:** The platform allows pharmacies to create offers for available medications, and users can browse, search, and place orders efficiently. The modular design of the offer system ensures scalability and maintainability.
- **User-Friendly Interface with Vue.js and Inertia.js:** A modern single-page application (SPA) experience was achieved by integrating Laravel with Vue.js via Inertia.js, leading to a seamless and dynamic user interface.
- **Secure User Authentication:** Laravel's built-in authentication mechanisms were utilized to manage user registration, login, and profile updates securely.
- **Dynamic Rating System:** The platform supports order rating , enabling pharmacies to assess customer satisfaction and improve their offerings.
- **Efficient Backend Architecture:** The backend was designed using Laravel's MVC architecture, ensuring organized and maintainable code, along with well-defined RESTful routes.

7.2 Possible Improvements or Extensions for Future Work

While the current version of PharmaBridge meets its foundational goals, there are several areas that offer potential for enhancement in future iterations. These include both technical expansions and user experience improvements:

Feature	Future Improvements
System Architecture	Develop native mobile applications (Android/iOS) to complement the web platform, allowing broader user access and native device features .
Scalability & LLM Deployment	Migrate the chatbot LLM to a cloud-based infrastructure to handle higher loads, enable real-time updates, and reduce latency for large-scale user bases.
Security Enhancements	Implement advanced security measures such as two-factor authentication (2FA), password recovery, and activity logging to strengthen data protection.
Search & Filtering	Add advanced search capabilities, including category-based filtering, price range sliders, availability indicators, and keyword highlighting.
Payment Gateway Integration	Integrate secure online payment solutions to allow users to complete transactions directly on the platform, streamlining the buying experience.
Notifications & Alerts	Introduce email/SMS/push notifications for order updates, offer expirations, or new medicine arrivals to keep users informed.
Admin Dashboard	Develop a dedicated admin panel to manage users, track performance, view statistics, and moderate content.

Table 5: Futur work

These enhancements not only address the limitations outlined in Chapter 6, but also pave the way for PharmaBridge to become a comprehensive, scalable, and fully-featured digital health platform. Future efforts will focus on refining performance, enhancing user engagement, and incorporating cutting-edge technologies to deliver a robust and inclusive experience for all users.

References

1. Dawaey is <https://dawaey.com/>
2. HealthWarehouse is <https://www.healthwarehouse.com/>
3. PharmAct is <https://pharmact.de/en/>
4. Rx Outreach is <https://rxoutreach.org/>
5. CVS is <https://www.cvshealth.com/>
6. drugs.com is <https://www.drugs.com/support/about.html>
7. Altibbi is <https://altibbi.com/>
8. Laravel Documentation. (2025). *The PHP Framework for Web Artisans*. <https://laravel.com/docs>
9. Vue.js Documentation. (2025). *Progressive JavaScript Framework*. <https://vuejs.org/guide/introduction.html>
10. Inertia.js. (2025). *Modern monolith approach to building SPAs*. <https://inertiajs.com>
11. Tailwind CSS. (2025). *Utility-first CSS framework*. <https://tailwindcss.com/docs>
12. Mistral AI. (2024). *Open-weight large language models*. <https://mistral.ai>
13. MySQL Documentation. (2025). *Relational Database Management System*. <https://dev.mysql.com/doc/>
14. Font Awesome. (2025). *Icon toolkit*. <https://fontawesome.com/>

15. Axios GitHub Repository. (2025). *Promise-based HTTP client.*

<https://github.com/axios/axios>

16. PHPUnit. (2025). *Testing framework for PHP.*

<https://phpunit.de/>