Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 3, No. 2, 2023

Patient-oriented online healthcare services platform: development based on Vue3 and Ts

Shuaina Wang, Hang Yin, Xiangkun Wang

Computer Science and Software Engineering Department, University of Science and Technology Liaoning, Anshan, China

Abstract: This paper presents a patient-oriented online medical service platform. The aim is to be able to alleviate the problem of shortage of medical resources and inadequate supply of services, and for patients to be provided with more convenient medical services. The platform uses Vue3+Ts technology to develop the front-end, and uses Elements-plus for optimising the front-end interface, and Piana for status management. In addition, the platform uses SignalR technology to enable real-time communication between patients and doctors. This makes it easier for patients to use. Platform functionality includes drug purchasing, patient question posting, doctor answering and patient-doctor one-to-one consultation functions. This paper describes in detail the technical approach to the platform and the system implementation. The platform has also been tested. The results show that the platform is easy to maintain and simple to operate and is expected to play a more important role in future healthcare systems.

Keywords: Online healthcare platform; Vue.js; Typescript; Pinia; SignalR.

1. Introduction

With the increasing normalisation of the new crown epidemic and an ageing population, healthcare needs are growing. According to statistics, the global share of people aged 65 and over is expected to grow from 9% in 2019 to 16% in 2050. This means that healthcare systems will be under greater pressure and more healthcare resources will be needed to meet the growing demand. By the end of 2022, a range of cold and flu medicines such as ibuprofen will be in short supply as the new epidemic is gradually liberalised. In early 2023, the outbreak of influenza A led to a renewed shortage of oseltamivir. These are all problems that we need to actively seek solutions to. In order to solve these problems, many countries and regions are actively promoting the development of health information technology. As an important part of healthcare informatisation, patient-oriented online medical service platforms can provide convenient and efficient medical services to patients. Based on the above background, this paper adopts the development model of Vue family bucket, using .Net6 on the back end and separating the front and back ends, to design and develop a medical platform that can realize the functions of online medical consultation and online medicine purchase for patients in their vicinity.

2. Relevant technologies

2.1. Vue3.2

Vue.js, a technical framework for front-end web design and development, is very convenient and powerful. It is essentially a JavaScript MVVM library, a front-end framework known for its light weight and high performance, with fast data rendering [1]. Also Vue.js is a progressive JavaScript framework for building user interfaces and single page applications [2]. Vue 3.2 is a new version of vue.js that introduces a new single-file component feature called <script setup>. It is a compile-time syntactic sugar that can greatly improve productivity when using the Composition API within a single-file component. It reduces sample code and makes the code more concise and readable. The actual principle of <script setup> is that the code in the <script setup> section is converted to the contents of the setup function at compile time. This means that when Vue compiles a single file component, it checks to see if the <script> tag contains a setup attribute. If so, Vue automatically converts all the top-level variables and functions in the <script> section to the return value of the setup function. Vue 3.2 has made significant improvements to support TypeScript, bringing many benefits to developers. One example is that the <script setup> tag supports declaring props and custom events using the pure TypeScript language. This helps developers to make better use of the capabilities of TypeScript.

2.2. Vite build tools

The Vue CLI is a more mature and stable build tool. It offers a rich set of features and plugins that can help you quickly build and configure Vue projects. However, as it is based on webpack, it can be slow to package in large projects. vite is a new front-end build tool that offers features such as extremely fast service startup and light and fast hot reloading. It uses native ES modules for on-demand file serving, which means Vite can load files as and when they are needed, rather than loading them all at once and without packaging. This means that during development, developers can see their changes immediately, without having to wait for the packaging process. This can dramatically increase the speed of service starts and hot updates. In addition, Vite offers rich features such as out-of-the-box support for TypeScript, CSS and more, which means that Vite can support these languages and technologies directly, without additional configuration. as well as pre-configured Rollup builds, etc.

2.3. TypeScript + Volar

The typeScript also has many advantages over JavaScript. It is a superset of JavaScript, with added features such as a type system and classes. These features improve the readability and maintainability of code, help developers write accurate code faster, and catch errors before they run. It can help developers with robust, more maintainable code. Static type checking can help developers catch potential errors while writing code, rather than at runtime. This can significantly reduce debugging time and increase development efficiency.

In addition, TypeScript provides object-oriented programming features such as classes and interfaces to help developers organise and maintain their code. typeScript also provides support for the latest ECMAScript features, allowing developers new JavaScript language features to write code. In addition, TypeScript has a rich ecosystem of tools and libraries to help you develop your projects. Once a project has been developed and compiled using Vue and TypeScript, the TypeScript code is compiled into JavaScript code to run in the browser. This process is usually done automatically by the build tool (Vite). The compiled JavaScript code is indistinguishable from normal JavaScript code and will run normally in the browser. Currently browsers only support JavaScript as a client-side scripting language. This means that all code that runs in the browser must be JavaScript code. Nevertheless, it is possible to develop projects in other languages such as TypeScript. These languages are usually compiled into JavaScript code in order to run in the browser.

Volar is a VSCode extension plugin that provides powerful TypeScript support specifically for Vue 3, replacing the previous Vetur plugin. When using Volar and TypeScript, we can enjoy the benefits of type detection in Vue 3.2+ ts projects, such as reduced runtime errors, enhanced refactoring reliability, increased code completion intelligence and development efficiency. volar also supports new Vue 3 features such as script setup syntactic sugar, ref sugar in templates, custom directives and more. In addition, Volar provides useful features such as CSS hover hints in templates, style variable renaming, component property documentation, etc. Volar makes it easier to develop and maintain Vue 3 projects.

2.4. The Pinia state management library

Pinia is a state management library that allows developers to share and modify data across different components, making it simpler and more convenient than Vuex. This makes it clearer and easier to manage.

3. System solution analysis

3.1. System requirements analysis

The design and implementation of a platform begins with a requirements analysis of the system, i.e. functional and non-functional requirements[3]. The online medical platform based on Vue+Vite+Ts+Pinia has three main roles for users: patients, doctors and administrators. The different roles correspond to different business requirements and play different roles in the platform. The main pages are the login and registration page, the main platform page, the medicine purchase screen, the shopping cart order screen, the doctor search and the online consultation screen. The login page can be used to verify different identities, while the main page of the platform allows users to access medical services online, add orders to the shopping cart, share information between users and display the platform. The administrator can update the drug data through the drug interface to facilitate the purchase of drugs by patients and to avoid the phenomenon that one drug is hard to find. The doctor-seeking interface also provides the contact information of the same doctor, so that patients can search for and obtain information of nearby doctors.

3.2. Entity analysis

The following entities are mainly involved in online healthcare platforms:

1)User: The user is the core of the platform and all operations in the platform are authenticated and authorised by the user. The user entity includes user ID, username, password, email, address and other information.

2)Medicine: Medicine entity includes medicine ID, medicine name, description, price, stock and other information.

3)Cart: The Cart entity includes the Cart ID, User ID, Medicine ID, Quantity, Total Price, etc.

4)Order: The Order entity includes the Order ID, User ID, Drug ID, quantity, total price, order status, etc.

5)Doctor: The physician entity includes information such as doctor ID, doctor name, contact information, hospital, address, etc.

3.3. Database Analysis

In this project, we use SQL Server as the database management system, and access and manipulate data through Entity Framework Core, a powerful, secure and stable relational database management system that can meet the data storage and management needs of online healthcare platforms. Core is an ORM framework based on .NET that enables developers to easily map objects to databases and provides a convenient interface for data access and manipulation, simplifying the writing of code for data access. By using Entity Framework Core, we can easily carry out database modelling, data reading, writing and querying operations.

4. Design and implementation

4.1. Project build and configuration

First, we use the Vite build tool to create a Vue+Ts based project. Initialise the project by running the npm init vite@latest command and selecting the Vue+Ts template. Then, install the required dependency packages for the project by running the npm install command.

Next, we install axios, Pinia, element-plus and echarts via npm. ehcarts is a JavaScript-based charting library and element-plus is a Vue 3-based UI framework that provides a rich set of components and styles. It allows us to draw various types of data visualisation charts. Then, under the src folder, we create the router folder and the store folder, and create the index.ts file in them. In the router folder, we import the createRouter and createWebHashHistory functions and create a routing instance. We define some routing rules and pass them to the routing instance. Finally, we export the routing instance so that we can use it elsewhere. In the store folder, we import the createStore function and create a state store instance. We define some state variables and pass them to the state store instance. Finally, we export the state store instance so that it can be used elsewhere.

Next, we create an axios instance by wrapping it in a second layer. The purpose of our two-tier wrapping of axios is to unify the interceptors that handle requests and responses, as well as features such as error handling. Set some default configurations in the axios instance, such as the base URL, timeout time, etc. Next, we add a request interceptor and a response interceptor to the axios instance. The request interceptor allows us to do something after the response is received, such as determining the response status code, handling error messages, etc.

Finally, we introduce Pinia, router and element-plus in main.ts and register them with the Vue instance. main.ts is the project entry file where we can introduce some global libraries or configuration. We import Pinia and router in main.ts and pass them as arguments to the createApp function. This will allow the Vue instance to use the Pinia and router functions. Then, import element-plus in main.ts and call the use function to register it with the Vue instance. This will allow the Vue instance to use the components and styles provided by element-plus.

In addition, to solve the cross-domain problem on the front and back ends, we also configure the cross-domain configuration in vite.config.ts. We use the proxy option to set up a proxy server that forwards local requests to the target server. We define an object where the key name is the local request path and the key value is the address of the target server. This way, when we send a local request, it will automatically be forwarded to the target server and return the response data.

4.2. Login and Registration Module

Users of the online medical platform can register through their mailboxes, which will send a verification code to the user, and input the code to achieve verification. The registration module display diagram is shown in Figure 1:

Figure 1. Presentation of the registration module

Mailbox and password rule verification is achieved by combining element-plus's form validation component. In the user's axios instance, the response interceptor section outputs detailed information about the back-end response and the different error status codes using the ElMessage component. In the request interceptor the user's token and the request header are set uniformly.

When the user logs in successfully, the user's personal information is retrieved according to the request login interface and stored in the response interceptor using Pinia, and the state of the store is stored in localStorage according to the piniaPlugin. For example, different route hops are implemented depending on the role of the user in the userstore. When the user logs out or the local token expires, the contents of the userstore will be emptied and the local localStorage will be emptied as well. The login registration flowchart is shown in Figure 2.

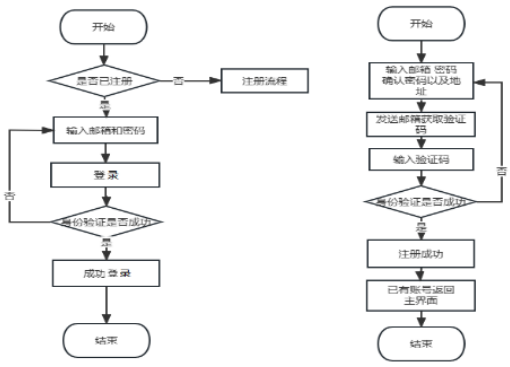

Figure 2. Login and Registration Flowchart

4.3. Main Page Introduction Module

Figure 3. Main page

In Figure 3 the main page module is shown, which includes a visual line chart and a navigation bar for the different services. The combination of Echarts technology gives the user more visual insight into how quickly the flu is spreading and the months of the different types of flu outbreaks. In the top right-hand corner of the main page there is a log out button, once the user has logged out, My Cart will change to Log In. At this point the user can go to the main page by clicking on the 'Login' button. The main page also incorporates javescript for dom manipulation, which makes the UI much more aesthetically pleasing.

4.4. Drug purchase module

4.4.1. Drug list module

In the drug list module, Elements-plus' auto-completion input box and paging components are introduced to greatly improve development efficiency and enable users to quickly access the drugs they need. As shown in Figure 4, the drug information retrieved from the database is loaded onto the page. This module enables the display of drug information in the drug purchase screen by introducing a custom drug list component. The child component receives the list of drug properties from the parent component via the props property, so that the component can use these properties to display the details of each drug in its own template. Here the custom drug list component acts as a child component that receives and displays drug information, while the parent component passes the drug property list data to the child component via the props property. In this way, the idea of componentization allows for reusability and maintainability of the code, while also making the application more flexible and easily extensible.

In this module, the cartStore state management instance stores a list of all products and orders for pharmaceuticals. When a user adds an item to the cart, not only is a request sent to the back-end, but also changes are made to the items in the cartStore.

Figure 4. List of drugs on the Display

4.4.2. Personal shopping cart and order module

In this module, the shopping cart data is retrieved, products are added and deleted, and orders are partially or fully placed, using the cartStore state management example. In addition, the components provided by the Elements-plus component library are used for quick and easy interaction. The CSS flex layout and Grid layout allows for a more aesthetically pleasing page layout and is more adaptable to different devices and screen sizes. The advantage of this approach is that the use of a state management library allows for the separation of data and logic, improves maintainability and readability of the code and facilitates communication and data sharing between different components. At the same time, the use of off-the-shelf component libraries can significantly reduce the amount of front-end development effort and increase development efficiency. In addition, the use of CSS layout techniques allows for better visual presentation and improved responsiveness of the page. When the user is not logged in, entering the corresponding shopping cart url directly into the navigation will cause the page to jump to the login page via a route blocker. The results are shown in Figure 5.

In an order, if the user has completed the payment, the payment status is usually determined based on the status code of the order, and then the payment button on the page is modified accordingly. For example, when the status of an order is paid, the user payment button will become paid and unclickable, and the paid information will be displayed on the page so that the user can clearly know the payment status of the order. This design helps to improve the user experience and reduce user confusion, and also facilitates the management and tracking of order status by the merchant.

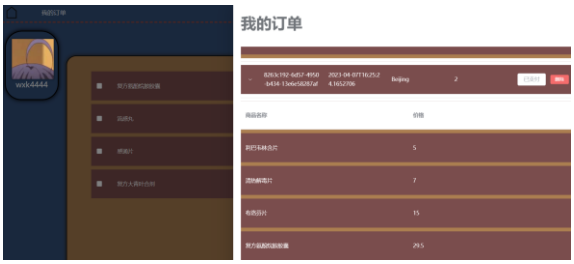

Figure 5. Shopping cart and list display

4.5. Medical consultation page

4.5.1. Patient Question Posting and Find a Doctor Module

In this module, users can post questions in different categories according to the questions they want to ask, which will be reviewed by the administrator and displayed in the corresponding section. Doctors can also view the content of their own section and can reply to patients' questions, or click on the doctor's avatar in the reply to enter a private chat. Find a Doctor binds the list of doctors sent from the back-end, so that patients can find different specialists for one-to-one communication according to their needs, or users can search by address or specialist name and bind the list of doctors returned from the search to continue the communication.

4.5.2. One-to-one consultation module

This module uses the SignalR library to implement real-time communication. SignalR is a technology provided by Microsoft that enables the establishment of long connections between the server side and the client side, encapsulating WebSocket to enable two-way communication[4]. the WebSocket protocol is a TCP-based full-duplex communication protocol that allows for It allows two-way communication between the server and the client, making data exchange easier. Unlike HTTP, the WebSocket protocol not only allows the client to send requests to the server, but also allows the server to actively push messages to the client [5]. A comparative diagram of the specific flow of a WebSocket connection is shown in Figure 6.. The process is mainly divided into a one-to-one handshake phase and a data communication phase.

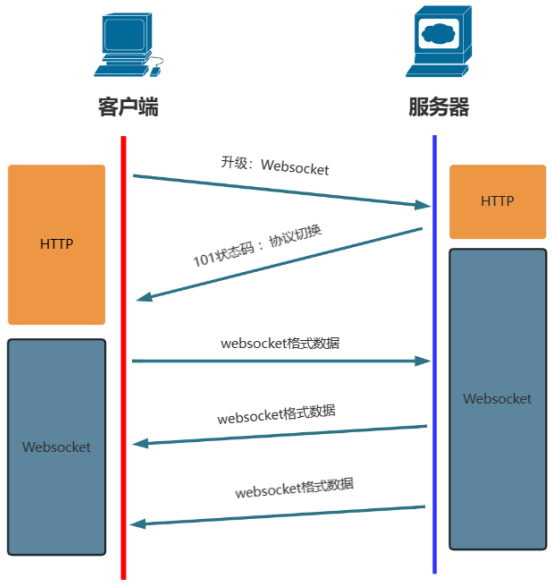

Figure 6. WebSocket connection establishment flowchart

When a patient selects a doctor for a consultation, the client constructs an object for a private chat request and sends the request to the server using SignalR's invoke method. The request contains the identifier of the target doctor, the content of the message, etc. Once the SignalR server receives the private chat request, it forwards the request to the target doctor. After the target doctor receives the private chat request, it is processed accordingly. The message content is displayed in the page window. In summary, the SignalR library provides a convenient real-time communication technology by encapsulating the WebSocket protocol, making two-way communication between the server and the client much easier and more efficient. When implementing the private chat function, the client constructs a private chat request, the server passes the request to the target doctor via SignalR, and the target doctor receives the request and processes it accordingly, thus realising a one-to-one private chat between the patient and the doctor. The specific effect is shown in Figure 7.

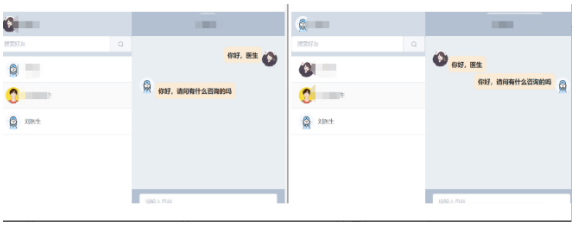

Figure 7. Consultation display

5. Summary

The patient-oriented online medical services platform was developed using technologies such as Vue, TypeScript, Pinia, Element Plus and SignalR to provide patients with easy access to online consultation and medication shopping, as well as the ability to easily find a nearby doctor. The clever combination of these technologies makes it easier to manage the status of the application, the interface is more aesthetically pleasing and real-time communication between doctors and patients is easy. The platform allows patients to easily consult and purchase medicines online, alleviating the shortage of healthcare resources and inadequate healthcare services, and improving the efficiency and convenience of healthcare delivery. It also demonstrates the importance of technology integration in solving practical problems. Studies have shown that the development and application of online medical platforms can alleviate the problems of shortage of medical resources and inadequate medical services and make it more convenient for patients to access medical services. In the future, the functions and services of online healthcare platforms can be further improved to enhance their role in the healthcare system, and research and improvement on patient acceptance and trust in online healthcare platforms can be enhanced so that online healthcare platforms can better serve patients and the healthcare system.

References

[1] Meng Yanqi. Design and implementation of an English core literacy platform based on vue.js [D]. Beijing University of Posts and Telecommunications, 2021. doi: 10.26969 / d.cnki.gbydu.2021.002883.

[2] Li Xiaowei. vue.js front-end application technology analysis [J]. Network Security Technology and Applications, 2022, No.256(04): 44-45.

[3] Xu Yumei. Design and implementation of migration data auditing software[J]. Science Times, 2012, 000(002):106-107.

[4] Li Yan. Implementation of real-time Web functions based on ASP.NET SignalR[J]. Computer Knowledge and Technology, 2016, 12(24):62-63. doi: 10.14004 / j.cnki.ckt. 2016. 3302.

[5] Zhang Y. Research and implementation of WebSocket-based instant messaging system[J]. Software, 2015, 36(03):89-94.