Gaming shop project documentation

Summary:

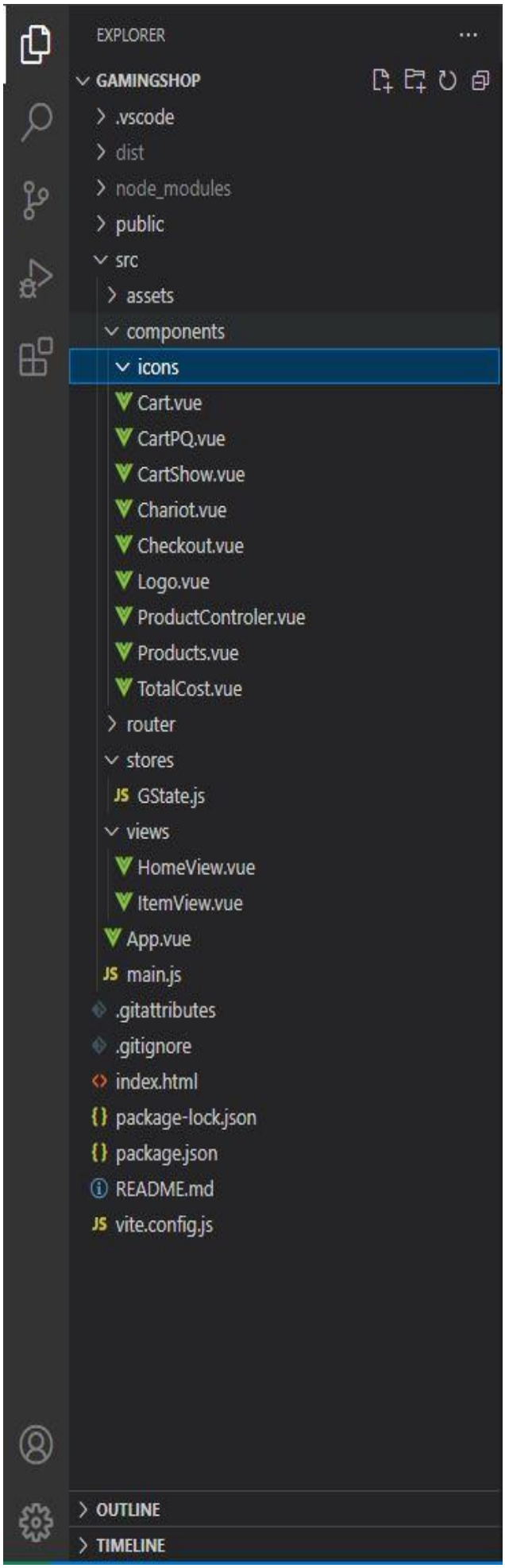
1-Introduction:

This project is front-end part of an e-commerce app. Entirely build with technologies including Html, Css and JavaScript. This App is powered by Vue.js Framework.

All the images contained in this project are provided by unsplash.com with free license. These are used for product image example all over this project.

We will see all the components detail and functionalities bellow.

2-The Project tree:



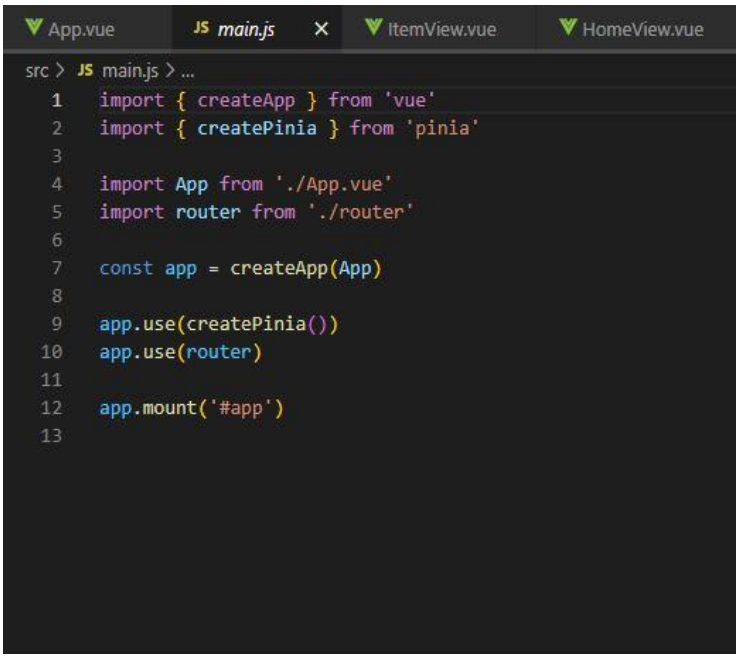
This is the representation of the project tree in Visual Studio code.

This project already contains a distributable file for production, build with npm package.

As we can see, all the components are stored in the "components" directory to build each components in the App, the router is in "router" for routing but this project is in Single Page Application, so the router target is just the HomeView component that we will see below. And the store is in "stores" directory for state management. In this case, we used Pinia, the official state manager for Vuejs application. Views are in the "views" directory.

3-Principal components:

- **The Main.js component**

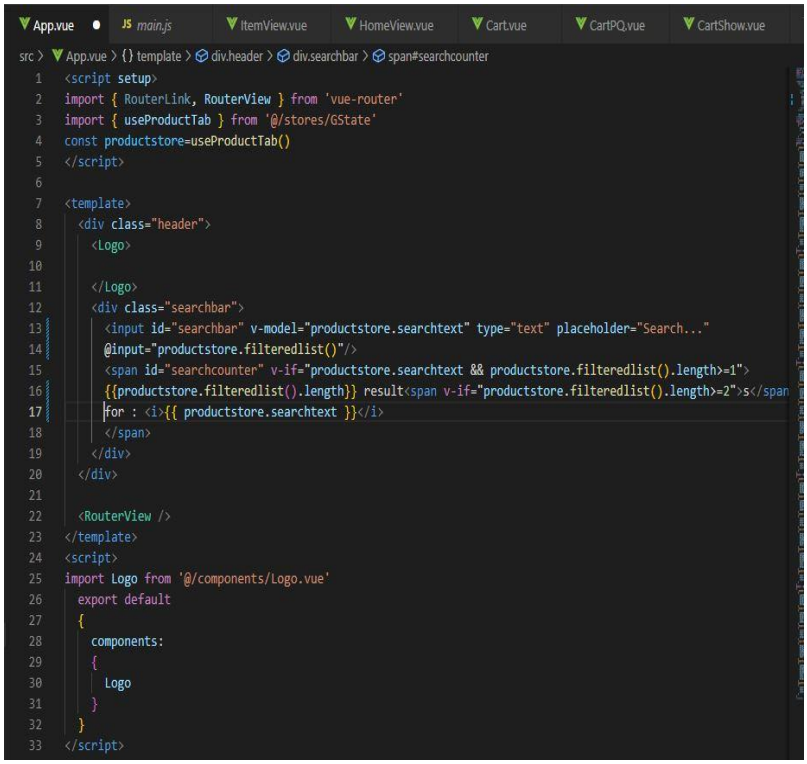


This is the main.js component.

Used to create a Vue instance. Generally, this is the Application root.

As we can see, at the top line , the importation of “CreateApp” function to create a Vue instance and the “CreatePinia” to create an instance of Pinia that we use for the entire Project. This is the state manager that contains the data source, all the functions and methods used in the app.

- **The App.vue component**



This is the App.vue component, appear in the main page. From line 1 to 5, we import the “RouterLink” and the ”RouterView” from “vue-router” to build the route within the app. At the line 3 and 4, this is the importation of “UseProductTab”, a reference in store to use all over the app once assigned to a constant. Here we use “producstore” for that.

From the line 7 is beginning of the App.vue template. The “Logo” component is imported at the line 25 as local component.

After the “Logo” component to line 17, we use an input element for the search field.

The v-model directive of this input field use the “productstore” constant. The target is “searchtext”, a variable declared in the store referred by “useProductTab” state.

After that, we use a element to show the result number once a text is entered in the input field and match an item in the “product_tab” array at the store. “filteredlist()” is the function

called by the input event. The span element appear when searchtext variable return “true” and filteredlist length is more than 1.

- **The router component**

```
src > router > JS index.js > ...
1  import { createRouter, createWebHistory } from 'vue-router'
2  import HomeView from '../views/HomeView.vue'
3
4  const router = createRouter({
5    history: createWebHistory(import.meta.env.BASE_URL),
6    routes: [
7      {
8        path: '/',
9        name: 'home',
10       component: HomeView
11     }
12   ]
13 })
14
15 export default router
16
```

This is the router component that make link all over the application.

We don’t need many routes here because only the “HomeView” component is the target.

At the line 1, this is the importation of “CreateRouter” and the “createWebHistory” function to create routes.

At the second line, we import the “HomeView” component from Views to match the route.

At the line 4, this is how to create an instance of vue router. The “createRouter” function need history and the route path. At the line 8, our path is the main page as “/”, name “home” and use “HomeView” as local component.

4-Views components:

- **HomeView.vue Component:**

```
src > views > HomeView.vue > {} template > div.home > div.home_content > div.itemcheckout > div.itemcheckout_content > table#chktable > tr > td > img
<script setup>
import { useProductTab } from '@/stores/GState'
import { useTabCart } from '@/stores/GState'
import { useItemView } from '@/stores/GState'
const productstore=useProductTab()
const cartstore=useTabCart()
const itemview=useItemView()
</script>
<template>
  <div class="home">
      <div class="home_content">
          <div class="product_container">
              <div :class="[{productContent:itemview.original},{opaque:itemview.isopaque}]">
                  <Products v-for="thumb in productstore.filteredlist()" :key="thumb.id"
                      @describe="itemview.decrire(thumb.id,thumb.idt,thumb.image,thumb.title,thumb.price,itemview.quantitytosend,thumb.descrip
                      @ajouter="cartstore.additem(thumb.id,thumb.idt,thumb.image,thumb.title,thumb.price,thumb.description,productstore.direct
                        :id=thumb.id
                        :productid=thumb.idt
                        :thumbs=thumb.image
                        :thumbtitle=thumb.title
                        :productprice=thumb.price>
                  </Products>
                  <div class="cart_button">
                      <CartShow :cartshowlabel=cartshowlabel :cartquantity=cartstore.getcartquantity
                      @click="cartstore.cartswitch()">
                      </CartShow>
                  </div>
                  <div class="cartview" v-if="cartstore.cartshow">
                      <Chariot v-for="(item,index) in cartstore.tab_cart" :key="item.index"
                      @removeitem="cartstore.removeitem(index)"
                      @cartminus=cartstore.cartminus(index)
                      @cartplus=cartstore.cartplus(index)
                      @moins=cartstore.moins(index)
                      @plus=cartstore.plus(index)
                      :itemthumbs="item.ivimagesource"
                      :itemid="item.productid"
                      :itemtitle="item.thumbtitle"
                      :itemsingleprice="item.productprice"
                      :cartitemqtt="item.itemqtt"
                      :cartitemtotal="item.eachtotalprice">
                      </Chariot>
                      <hr/>
                      <TotalCost :totalcost=totalcostlabel :totalamount=cartstore.totalamount>

                      </TotalCost>
                      <Checkout :clearlabel="clearlabel" :checkoutlabel="checkoutlabel"
                        :ifnoproducttoclear="cartstore.ifnoproducttoclear"
                        :ifnoproducttocheckout="cartstore.ifnoproducttocheckout"
                        @clearcart="cartstore.clearcart()"
                        @checkout="cartstore.checkout()">
                      </Checkout>
                  </div>
              </div>
          </div>
          <transition name="itemviewcontent">
            <ItemView v-if="itemview.agrandit" @quitter="itemview.quitter()"
                @itemplus="itemview.itemplus"
```