# Enable 3000 Prt

## Step 1: Open Command Prompt as Administrator

Press Win + S, type cmd.

Right-click Command Prompt → Run as administrator.

## Step 2: Add a Firewall Rule

Run this command to allow inbound traffic on port 3000 for TCP:

### Command

netsh advfirewall firewall add rule name="Open Port 3000" dir=in action=allow protocol=TCP localport=3000

If you also want to allow UDP (less common for web apps):

### Command

netsh advfirewall firewall add rule name="Open Port 3000 UDP" dir=in action=allow protocol=UDP localport=3000

## Step 3: Verify the Rule

You can check if the rule was added with:

### Command

netsh advfirewall firewall show rule name="Open Port 3000"

Step 4: Check if Port is Listening

If your app is running, verify it’s listening on port 3000:

### Command

netstat -ano | findstr :3000

Quasar

Common Built-in Vue Directives

# 1. Create a Vue 3 + Vite Project

npm create vite@latest my-quasar-app

cd my-quasar-app

npm install

Choose vue as the framework when prompted.

# 2. Install Quasar and Required Packages

npm install quasar @quasar/extras

# Also install:

npm install @vitejs/plugin-vue

npm install @quasar/vite-plugin --save-dev

npm install --save-dev @quasar/vite-plugin sass-embedded@^1.80.2

# 3. Update vite.config.js

import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'

import vue from '@vitejs/plugin-vue'

import { quasar, transformAssetUrls } from '@quasar/vite-plugin'

export default defineConfig({

plugins: [

vue({

template: { transformAssetUrls }

}),

quasar({

sassVariables: fileURLToPath(new URL('./src/quasar-variables.sass', import.meta.url))

})

],

resolve: {

alias: {

'@': fileURLToPath(new URL('./src', import.meta.url))

}

}

})

# 4. Import Quasar CSS and Icons in main.js

import { createApp } from 'vue'

import { Quasar } from 'quasar'

import App from './App.vue'

// Import Quasar styling and icons

import 'quasar/src/css/index.sass'

import '@quasar/extras/material-icons/material-icons.css'

const app = createApp(App)

app.use(Quasar, {

plugins: {} // You can add Quasar plugins here

})

app.mount('#app')

# 5. Create quasar-variables.sass in src/

This file lets you customize theme colors (optional):

// src/quasar-variables.sass

$primary: #027be3

$secondary: #26a69a

$accent: #9c27b0

$dark: #1d1d1d

$positive: #21ba45

$negative: #c10015

$info: #31ccec

$warning: #f2c037

# Quasar CSS Class Cheatsheet

# 1. Spacing (Margin & Padding)

* q-mt-sm → margin-top: small
* q-pa-md → padding: medium
* Format: q-[m/p][t/r/b/l/x/y]-[none/xs/sm/md/lg/xl]

# 2. Flexbox

* row, column
* justify-center, justify-between, justify-end
* items-center, items-start, items-end
* wrap, no-wrap

# 3. Typography

* text-h1, text-subtitle1, text-body2, text-caption
* text-bold, text-italic
* text-center, text-right, text-left
* text-uppercase, text-capitalize, text-lowercase

# 4. Sizing

* fit, full-width, full-height

# 5. Shadow

* shadow-1 to shadow-24
* no-shadow

# 6. Display

* block, inline, inline-block
* hidden, invisible

# 7. Borders & Radius

* border, border-top, border-bottom
* rounded, radius-xs, radius-md, radius-xl

# 8. Colors

* text-primary, text-negative, bg-accent, bg-grey-3
* Color palette: https://quasar.dev/style/color-palette

🔗 Official Docs: https://quasar.dev/style/typography

# Common Built-in Vue Directives

|  |  |
| --- | --- |
| Directive | Description |
| v-if | Conditionally render an element |
| v-else | Used with v-if for "else" block |
| v-else-if | Used with v-if for chained conditions |
| v-show | Toggle element visibility via CSS (display: none) |
| v-for | Loop through arrays or objects |
| v-bind | Dynamically bind an attribute (: is shorthand) |
| v-model | Two-way data binding for form inputs |
| v-on | Listen to DOM events (@ is shorthand) |

# Install Vue Router

## First, install Vue Router if you haven't already:

npm install vue-router

# Configure the Router

In your src folder, create a router folder and inside that, create a index.js file. This file will contain the configuration for the routes.

### src/router/index.js

import { createRouter, createWebHistory } from 'vue-router'

import Home from '../views/Home.vue' // import your views

import About from '../views/About.vue'

const routes = [

{

path: '/',

name: 'Home',

component: Home

},

{

path: '/about',

name: 'About',

component: About

},

// add more routes here

]

const router = createRouter({

history: createWebHistory(process.env.BASE\_URL), // this enables HTML5 History mode

routes

})

export default router

# Add Router to the Vue Instance

In your src/main.js file, you will need to import the router and tell Vue to use it.

## src/main.js

import { createApp } from 'vue'

import App from './App.vue'

import router from './router' // import the router

createApp(App)

.use(router) // use the router here

.mount('#app')

# 1. Global Axios Setup with Token Handling

## Create a file like axios.js or http.js:

### // src/axios.js

import axios from 'axios'

const axiosInstance = axios.create({

baseURL: 'http://192.168.0.124:9090/users/api',

timeout: 10000,

headers: {

'Content-Type': 'application/json',

},

})

// Token helpers

function getAccessToken() {

return localStorage.getItem('accessToken')

}

function getRefreshToken() {

return localStorage.getItem('refreshToken')

}

function setTokens({ accessToken, refreshToken }) {

localStorage.setItem('accessToken', accessToken)

localStorage.setItem('refreshToken', refreshToken)

}

function clearTokens() {

localStorage.removeItem('accessToken')

localStorage.removeItem('refreshToken')

}

// Refresh logic

let isRefreshing = false

let refreshSubscribers = []

function subscribeTokenRefresh(cb) {

refreshSubscribers.push(cb)

}

function onRefreshed(token) {

refreshSubscribers.forEach(cb => cb(token))

refreshSubscribers = []

}

// Request: add accessToken

axiosInstance.interceptors.request.use((config) => {

const token = getAccessToken()

if (token) {

config.headers.Authorization = `Bearer ${token}`

}

return config

})

// Response: refresh token on 401

axiosInstance.interceptors.response.use(

(res) => res,

async (error) => {

const originalRequest = error.config

if (error.response?.status === 401 && !originalRequest.\_retry) {

originalRequest.\_retry = true

if (isRefreshing) {

return new Promise(resolve => {

subscribeTokenRefresh(token => {

originalRequest.headers.Authorization = 'Bearer ' + token

resolve(axiosInstance(originalRequest))

})

})

}

isRefreshing = true

try {

const res = await axios.post('http://192.168.0.124:9090/api/auth/refresh', {

refreshToken: getRefreshToken(),

})

const { accessToken, refreshToken } = res.data

setTokens({ accessToken, refreshToken })

isRefreshing = false

onRefreshed(accessToken)

originalRequest.headers.Authorization = 'Bearer ' + accessToken

return axiosInstance(originalRequest)

} catch (err) {

clearTokens()

isRefreshing = false

window.location.href = '/login'

return Promise.reject(err)

}

}

return Promise.reject(error)

}

)

export default axiosInstance

## page/Login.vue — Login page using axiosInstance

<template>

<div class="login-wrapper">

<div class="login-box">

<h2>Login</h2>

<form @submit.prevent="handleLogin">

<input v-model="email" type="email" placeholder="Email" required />

<input v-model="password" type="password" placeholder="Password" required />

<button type="submit" :disabled="loading">

{{ loading ? 'Logging in...' : 'Login' }}

</button>

<p v-if="errorMessage" class="error">{{ errorMessage }}</p>

</form>

</div>

</div>

</template>

<script setup>

import { ref } from 'vue'

import { useRouter } from 'vue-router'

import axiosInstance from '@/plugins/axios'

const email = ref('')

const password = ref('')

const loading = ref(false)

const errorMessage = ref('')

const router = useRouter()

const handleLogin = async () => {

loading.value = true

errorMessage.value = ''

try {

const response = await axiosInstance.post('/auth/login', {

email: email.value,

password: password.value,

})

const { accessToken, refreshToken } = response.data

localStorage.setItem('accessToken', accessToken)

localStorage.setItem('refreshToken', refreshToken)

router.push('/')

} catch (err) {

errorMessage.value = 'Invalid credentials'

console.error(err)

} finally {

loading.value = false

}

}

</script>

# Step-by-Step Guide to Use Cookies

## 1. Install js-cookie

npm install js-cookie

## 2. Import and Use js-cookie

### In your axios.js or login.vue (wherever you're storing tokens):

import Cookies from 'js-cookie'

## 3. Replace localStorage with Cookies

### Set Tokens:

Cookies.set('access\_token', access\_token, { expires: 1 }) // expires in 1 day

Cookies.set('refresh\_token', refresh\_token, { expires: 7 }) // optional longer expiry

### Get Tokens:

Cookies.get('access\_token')

Cookies.get('refresh\_token')

### Remove Tokens (on logout):

Cookies.remove('access\_token')

Cookies.remove('refresh\_token')

## 4. Update Your Axios Interceptor

Replace localStorage.getItem(...) with Cookies.get(...):

### // axios.js

axiosInstance.interceptors.request.use((config) => {

const token = Cookies.get('access\_token')

if (token) {

config.headers.Authorization = `Bearer ${token}`

}

return config

})