+++++++++++++++++++++++++++++++++

| Day 3

| Nuxt3 Vuetify for scalable project

+++++++++++++++++++++++++++++++++

Step 1: Create new nuxt3 project

---

npx nuxi init nuxt3-scalable-vuetify

เปิดเข้า vscode

---

code nuxt3-scalable-vuetify -r

Step 2: Install sass dependencie

---

npm install -D sass@1.56.1

Step 3: Install Vuetify 3 dependencie

---

npm install -D vuetify@3.1.2

Step 4: Install Vite Plugin Vuetify

---

npm install vite-plugin-vuetify@1.0.2

Step 5: Install Nuxt Icon

---

npm install -D nuxt-icon@0.2.10

Step 6: Install Nuxt Google Fonts

---

npm install -D @nuxtjs/google-fonts@2.0.0

Step 7: Install Nuxt Robot

---

npm install -D @nuxtjs/robots@3.0.0

Step 8: Create assets/scss/style.scss

---

@use "vuetify/styles";

Step 9: Config nuxt.config.ts

---

import vuetify from "vite-plugin-vuetify"

// https://nuxt.com/docs/api/configuration/nuxt-config

export default defineNuxtConfig({

ssr: true,

css: ["@/assets/scss/style.scss"],

build: {

transpile: ["vuetify"],

},

vite: {

ssr: {

noExternal: ["vuetify"],

},

define: {

"process.env.DEBUG": false,

},

},

modules: [

"nuxt-icon",

// [

// "@nuxtjs/robots",

// {

// UserAgent: "\*",

// Disallow: "",

// },

// ],

// this adds the vuetify vite plugin

// also produces type errors in the current beta release

async (options, nuxt) => {

// @ts-ignore

nuxt.hooks.hook("vite:extendConfig", (config) => config.plugins.push(vuetify()))

},

],

app: {

head: {

htmlAttrs: {

lang: 'en'

},

bodyAttrs: {

class: 'demo'

},

charset: 'utf-8',

titleTemplate: '%s - Nuxt 3 Vuetify',

meta: [

{

name: 'viewport',

content: 'width=device-width, initial-scale=1, maximum-scale=5'

},

{

name: 'author',

content: "IT Genius Engineering Ltd., Thailand"

},

]

}

},

})

Step 10: Create components/MIcon.vue

---

<script setup>

</script>

<template>

<div>

<Icon v-bind="$attrs" />

</div>

</template>

Step 11: Create helpers/customIcons.ts

---

import { h } from "vue"

import type { IconSet, IconAliases, IconProps } from "vuetify"

// Custom icon component

import MIcon from "~~/components/MIcon.vue"

/\*\*

\* Code for the icon component looks like this

\*

<template>

<div>

<Icon v-bind="$attrs" />

</div>

</template>

<script setup></script>

\*/

/\*\*

\* Using the Phospor pack here, but any icon name can be used.

\* All of vuetify's core icons MUST be added in the `aliases` object

\*/

const aliases: IconAliases = {

complete: "ph:check-circle",

cancel: "ph:x-circle",

close: "ph:x-circle",

delete: "ph:trash",

clear: "ph:x-circle",

success: "ph:check-circle",

info: "ph:info",

warning: "ph:warning",

error: "ph:x-circle",

prev: "ph:caret-left",

next: "ph:caret-right",

checkboxOn: "ph:check-square-fill",

checkboxOff: "ph:square",

checkboxIndeterminate: "ph:square",

delimiter: "ph:circle",

sort: "ph:caret-up",

expand: "ph:caret-down",

menu: "heroicons:bars-2",

subgroup: "ph:caret-down",

dropdown: "ph:caret-down",

radioOn: "ph:radio-button-fill",

radioOff: "ph:circle",

edit: "ph:pencil-simple",

ratingEmpty: "ph:star",

ratingFull: "ph:star-fill",

ratingHalf: "ph:star-half-fill",

loading: "ph:spinner",

first: "ph:caret-double-left",

last: "ph:caret-double-right",

unfold: "ph:arrows-out",

file: "ph:file",

plus: "ph:plus",

minus: "ph:minus",

sortAsc: undefined,

sortDesc: undefined,

}

const custom: IconSet = {

component: (props: IconProps) =>

// Return render function

h(MIcon, {

name: props.icon,

tag: props.tag,

disabled: props.disabled,

}),

}

// export both aliases and the custom object created

export { aliases, custom }

Step 12: Create helpers/themes.ts

---

import { ThemeDefinition } from 'vuetify'

// String that represents the name of the light theme

export const LIGHT\_THEME = 'lightTheme'

export const lightTheme: ThemeDefinition = {

dark: false,

colors: {

background: "#FFFFFF",

surface: "#FFFFFF",

primary: "#4f46e5",

secondary: "#9333ea",

error: "#ef4444",

info: "#3b82f6",

success: "#22c55e",

warning: "#f59e0b",

},

}

// String that represents the name of the dark theme

export const DARK\_THEME = 'darkTheme'

export const darkTheme: ThemeDefinition = {

dark: true,

colors: {

background: "#0C111B",

surface: "#1f2937",

primary: "#6366f1",

secondary: "#9333ea",

error: "#ef4444",

info: "#3b82f6",

success: "#22c55e",

warning: "#f59e0b",

},

}

Step 13: Create helpers/defaults.ts

---

import { DefaultsInstance } from "vuetify/lib/framework.mjs"

export const defaults: DefaultsInstance = {

VAppBar: {

elevation: 0,

},

VBtn: {

variant: "flat",

height: 38,

rounded: "lg",

size: "small",

},

VTextField: {

color: "primary",

variant: "outlined",

density: "comfortable",

},

}

Step 14: Create plugins/vuetify.ts

---

// import createVuetify from "vuetify"

import { createVuetify } from "vuetify"

// import custom icons from helpers

import { aliases, custom } from "@/helpers/customIcons"

// import theme from "@/helpers/themes"

import { LIGHT\_THEME, lightTheme, DARK\_THEME, darkTheme } from "@/helpers/themes"

// import defaults from "@/helpers/defaults"

import { defaults } from "@/helpers/defaults"

export default defineNuxtPlugin((nuxtApp ) => {

// Create a new Vuetify instance

const vuetify = createVuetify({

ssr: true,

defaults,

theme: {

defaultTheme: LIGHT\_THEME,

themes: {

lightTheme,

darkTheme,

},

// add color variations

variations: {

colors: ["primary", "secondary"],

lighten: 3,

darken: 3,

}

},

// Add the custom iconset

icons: {

defaultSet: "custom",

aliases,

sets: {

custom,

},

},

})

// Inject it to nuxtApp

nuxtApp.vueApp.use(vuetify)

})

Step 15: Update app.vue

---

<template>

<div>

<v-card class="mx-auto" width="400">

<v-card-title>

Card Title

</v-card-title>

<v-card-text>

This is content

</v-card-text>

<v-card-actions>

<v-btn variant="text" color="teal-accent-4">

Learn More

</v-btn>

</v-card-actions>

</v-card>

</div>

</template>

Step 16: Run project in dev mode

---

npm run dev

Step 17: Create layouts/default.vue

---

<template>

<div>

<VApp>

<VMain>

<slot />

</VMain>

</VApp>

</div>

</template>

Step 18: Create pages/index.vue for login page

----

code in example file...

Step 19: custom scss/style.scss

---

// Override the textfield's radius

.v-text-field {

.v-field\_\_outline\_\_start {

border-radius: 10px 0 0 10px !important;

}

.v-field\_\_outline\_\_end {

border-radius: 0 10px 10px 0 !important;

}

}

.v-btn {

font-weight: 700 !important;

}

.label {

font-weight: 600;

font-size: 15px;

margin-bottom: 6px;

display: inline-block;

cursor: pointer;

}

// Theme gradients

.gradient {

&.primary {

color: white;

background: linear-gradient(

318deg,

rgba(67, 56, 202, 1) 12%,

rgba(79, 70, 229, 1) 38%,

rgba(99, 102, 241, 1) 100%

) !important;

}

&.cancel {

// color: #1e293b;

background: linear-gradient(

318deg,

rgba(203, 213, 225, 1) 12%,

rgba(226, 232, 240, 1) 38%,

rgba(241, 245, 249, 1) 100%

) !important;

}

&.success {

color: white;

background: linear-gradient(

318deg,

rgba(21, 128, 61, 1) 12%,

rgba(22, 163, 74, 1) 38%,

rgba(34, 197, 94, 1) 100%

) !important;

}

&.info {

color: white;

background: linear-gradient(

318deg,

rgba(30, 64, 175, 1) 12%,

rgba(37, 99, 235, 1) 38%,

rgba(59, 130, 246, 1) 100%

) !important;

}

&.warn {

color: white;

background: linear-gradient(

318deg,

rgba(180, 83, 9, 1) 12%,

rgba(217, 119, 6, 1) 38%,

rgba(245, 158, 11, 1) 100%

) !important;

}

&.error {

color: white;

background: linear-gradient(

318deg,

rgba(153, 27, 27, 1) 12%,

rgba(220, 38, 38, 1) 38%,

rgba(248, 113, 113, 1) 100%

) !important;

}

&.gray {

color: white;

background: linear-gradient(

318deg,

rgba(30, 41, 59, 1) 12%,

rgba(44, 55, 74, 1) 38%,

rgba(71, 85, 105, 1) 100%

) !important;

}

}

// enforce font globally

\*,

html,

body {

font-family: 'Inter','IBM Plex Sans Thai', sans-serif!important;

-webkit-font-smoothing: antialiased !important;

}

Step 20: Set fonts in nuxt.config.ts

---

link: [

{

rel: "stylesheet",

href: "https://fonts.googleapis.com/css2?family=IBM+Plex+Sans+Thai:wght@200;300;400;500;600;700&family=Inter:wght@200;300;400;500;600;700;800;900&display=swap",

},

{ rel: "icon", type: "image/x-icon", href: "/favicon.ico" },

],

Step 21: Create register in pages/signup/index.vue

---

code in example file...

Step 22: Validation form create composables/useFormRules.ts

---

// Create a custom hook to use in your components

export const useFormRules = () => {

return {

ruleRequired: (value: string) => !!value || 'This field is required',

ruleEmail: (value: string) => {

const pattern =

/^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

return pattern.test(value) || "Enter a valid email";

},

rulePassLen: (v: string) => (!!v && v.length >= 6) || "Password must be 6 chars or more",

}

}

Step 23: Create Backend Server with Strapi

---

npx create-strapi-app@latest stra