1. Setup Laravel

Laravel Backend:

```
composer require laravel/breeze --dev
```

```
php artisan breeze:install api
```

Inside .env

```
SANCTUM_STATEFUL_DOMAINS=localhost:3000
SESSION_DOMAIN=localhost
```

```
SESSION_DRIVER=cookie
```

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=465
MAIL_USERNAME=yubrajkoirala7278@gmail.com
MAIL_PASSWORD=bzclhlgvxxxaepxv
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="yubrajkoirala7278@gmail.com"
MAIL_FROM_NAME="${APP_NAME}"
```

Inside RedirectIfAuthenticated.php

```
foreach ($guards as $guard) {
    if (Auth::guard($guard)->check()) {
        return response()->noContent();
    }
}
```

2. Install below necessary packages for vuejs:

```
npm create vite@latest
npm install
npm install vue-router@4
npm install pinia
npm install axios
```

Clear App.Vue and delete HelloWorld.vue and delete files inside assets.
Create src/router/index.js

```
npm install lodash
```

3. Change vuejs port to 3000 from vite.config.js

```
// https://vite.dev/config/
export default defineConfig({
  plugins: [vue()],
  server: {
    port: 3000,
  },
});
```

4. Inside src/axios.js

```
import axios from "axios";

// Base configuration
axios.defaults.baseURL = "http://localhost:8000"; // Ensure this matches your
backend
axios.defaults.withCredentials = true; // For cookies and CSRF

// Request interceptor to add CSRF token
axios.interceptors.request.use(
  (config) => {
    try {
      const cookies = document.cookie.split('; ');
      const tokenRow = cookies.find((row) => row.startsWith('XSRF-TOKEN='));
      if (tokenRow) {
        const token = decodeURIComponent(tokenRow.split('=')[1]);
        config.headers['X-XSRF-TOKEN'] = token;
```

```
      }
      return config;
    } catch (error) {
      console.error("Error setting CSRF token:", error);
      return config; // Proceed anyway to avoid breaking requests
    }
  },
  (error) => {
    return Promise.reject(error);
  }
);

export default axios;
```

5. How to setup vuetify in vuejs

```
npm install vuetify@latest
```

Inside src/plugins/vuetify.js

```
// src/plugins/vuetify.js
import { createVuetify } from 'vuetify';
import 'vuetify/styles'; // Import Vuetify styles

// Optional: Import components and directives if you want to use them explicitly
import * as components from 'vuetify/components';
import * as directives from 'vuetify/directives';

const vuetify = createVuetify({
  components,
  directives,
});

export default vuetify;
```

Inside main.js

```
// Import Vuetify configuration
import vuetify from './plugins/vuetify'
// Use Vuetify for UI components
app.use(vuetify);
```

Now, you can use vuetify..
For icon

```
npm install @mdi/font

// src/main.js
import '@mdi/font/css/materialdesignicons.css';


 <v-btn icon>
   <v-icon>mdi-home</v-icon>
 </v-btn>
```

a) Inside main.js

```
// src/main.js

// Import Vue's createApp function
import { createApp } from 'vue'

// Import the main App component
import App from './App.vue'

// Import global Axios configuration
import './axios'

// Import Pinia (State Management)
import { createPinia } from 'pinia'

// Import Vue Router (Routing)
import router from './router'

// Import Vuetify configuration
import vuetify from './plugins/vuetify'


import '@mdi/font/css/materialdesignicons.css';

// Create an instance of Pinia
const pinia = createPinia();

// Create a Vue application instance
const app = createApp(App);

// Use Pinia for state management
app.use(pinia);
```

```
// Use Vue Router for handling routes
app.use(router);

// Use Vuetify for UI components
app.use(vuetify);

// Mount the app to the DOM
app.mount('#app');
```

Note: uncomment router code when writing routing code in router/index.js

b) For routing..in src/router/index.js

```
import { createRouter, createWebHistory } from "vue-router";
import { useAuthStore } from "../stores/auth";
const routes = [
    {
        path: "/",
        name: "Frontend",
        component: () =>
import("../components/frontend/layouts/FrontendLayout.vue"),
        children: [
            {
                path: "",
                name: "Home",
                component: () => import("../components/frontend/Home.vue"),
            },
        ],
    },
    {
        path: "/auth",
        name: "Auth",
        component: () => import("../components/auth/layouts/AuthLayout.vue"),
        children: [
            {
                path: "/login",
                name: "Login",
                component: () => import("../components/auth/Login.vue"),
                meta: { guestOnly: true },
            },
            {
                path: "/register",
                name: "Register",
                component: () => import("../components/auth/Register.vue"),
```

```
                meta: { guestOnly: true },
            },
            {
                path: "/forget-password",
                name: "ForgetPassword",
                component: () => import("../components/auth/ForgetPassword.vue"),
                meta: { guestOnly: true },
            },
            {
                path: "/password-reset/:token",
                name: "ResetPassword",
                component: () => import("../components/auth/ResetPassword.vue"),
                meta: { guestOnly: true },
            },
        ],
    },
    {
        path: "/admin",
        name: "Admin",
        component: () =>
import("../components/backend/layouts/BackendLayout.vue"),
        children: [
            {
                path: "home",
                name: "Dashboard",
                component: () => import("../components/backend/Home.vue"),
                meta: { requiresAuth: true },
            },
            {
                path: "category",
                name: "Category",
                component: () =>
import("../components/backend/category/Category.vue"),
                meta: { requiresAuth: true },
            },
        ],
    },
];
const router = createRouter({
    history: createWebHistory(),
    routes,
});
// Global Navigation Guard
router.beforeEach(async (to, from, next) => {
    const authStore = useAuthStore(); // Access the auth store
```

```
    // Fetch the authenticated user if not already loaded
    if (!authStore.authUser) {
        await authStore.getUser(); // Fetch the user information from the backend
    }
    // Handle protected routes
    if (to.meta.requiresAuth && !authStore.authUser) {
        // Redirect to login if the user is not authenticated
        return next({ name: "Login" });
    }
    // Handle guest-only routes (e.g., /login, /register)
    if (to.meta.guestOnly && authStore.authUser) {
        // Redirect authenticated users to the dashboard
        return next({ name: "Dashboard" });
    }
    // Reset the fields on every route change
    authStore.resetFields();
    // Allow navigation
    next();
});
export default router;
```

c) Create below files

a) Src/components/auth/layouts/AuthLayout.vue, src/components/auth/Login.vue,Register.vue,ResetPassword.vue,ForgetPassword.vue

b) Src/components/frontend/layouts/FrontendLayout.vue, frontend/Home.vue

c) Src/components/backend/layouts/BackendLayout.vue,backend/Home.vue, backend/category/Category.vue

d) Inside App.vue

```
<script setup>
import { ref } from 'vue';
import { useRouter } from 'vue-router';

const isLoading = ref(false);
const router = useRouter();

// Show loader when navigation starts
router.beforeEach((to, from, next) => {
    isLoading.value = true;
    next();
});

// Hide loader when navigation is done
```

```
router.afterEach(() => {
    setTimeout(() => {
        isLoading.value = false;
    }, 500); // Small delay for smoother transition
});
</script>

<template>
    <main>
        <!-- Page Loader -->
        <v-overlay v-model="isLoading" class="loader-overlay" :scrim="false">
            <v-progress-circular indeterminate size="64" color="primary" />
        </v-overlay>

        <!-- The child routes will be injected here -->
        <router-view />
    </main>
</template>

<style scoped>
/* Page Loader Styles */
.loader-overlay {
    display: flex;
    align-items: center;
    justify-content: center;
    z-index: 9999;
    background: rgba(255, 255, 255, 0.8);
}
</style>
```

Note: Uncomment router code from main.js

e) Inside AuthLayout.vue, FrontendLayout.vue and BackendLayout.vue

```
<script setup>
</script>
<template>
    <main>
        <!-- The child routes will be injected here -->
        <router-view />
    </main>
</template>
```

f) For authentication inside src/stores/auth.js

```
import { defineStore } from "pinia";
import { ref } from "vue";
```

```
import axios from "axios";
import router from "../router";
export const useAuthStore = defineStore("auth", () => {
    // ========= State =========
    const authUser = ref(null); // Holds the authenticated user data
    const authErrors = ref([]); // Holds validation or other errors
    const authStatus = ref(null);
    const loading = ref(false);
    // =======Fetch the CSRF token==============
    const getToken = async () => {
        try {
            await axios.get("/sanctum/csrf-cookie");
        } catch (error) {
            console.error("Error fetching CSRF token:", error);
        }
    };
    // ====Handle user login===============
    const handleLogin = async (credentials) => {
        try {
            loading.value = true;
            await getToken(); // Ensure CSRF token is set
            const response = await axios.post("/login", credentials);
            authUser.value = response.data.user;
            router.push("/admin/home");
        } catch (error) {
            if (error.response && error.response.status === 422) {
                authErrors.value = error.response.data.errors;
            }
        } finally {
            loading.value = false;
        }
    };
    // ==========Handle user registration================
    const handleRegister = async (userDetails) => {
        try {
            loading.value = true;
            await getToken(); // Ensure CSRF token is set
            const response = await axios.post("/register", userDetails);
            authUser.value = response.data.user;
            router.push("/admin/home");
        } catch (error) {
            if (error.response && error.response.status === 422) {
                authErrors.value = error.response.data.errors;
            }
        } finally {
```

```
            loading.value = false;
        }
    };
    // =======Fetch authenticated user's information==========
    const getUser = async () => {
        try {
            await getToken(); // Ensure CSRF token is set
            const { data } = await axios.get("/api/user");
            authUser.value = data;
        } catch (error) {
            authUser.value = null;
        }
    };
    // =======Handle user logout=============
    const handleLogout = async () => {
        try {
            await axios.post("/logout");
            authUser.value = null;
            router.push("/login");
        } catch (error) {
            console.error("Logout failed:", error);
        }
    };
    // =====Handle forgot password============
    const handleForgetPassword = async (email) => {
        try {
            loading.value = true;
            await getToken(); // Ensure CSRF token is set
            const response = await axios.post("/forgot-password", { email });
            authStatus.value = response.data.status;
        } catch (error) {
            if (error.response && error.response.status === 422) {
                authErrors.value = error.response.data.errors;
            }
        } finally {
            loading.value = false;
        }
    };
    // =========Handle reset password=============
    const handleResetPassword = async (resetData) => {
        try {
            loading.value = true;
            await getToken(); // Ensure CSRF token is set
            const response = await axios.post("/reset-password", resetData);
            // authUser.value = response.data.user;
```

```
            authStatus.value = response.data.status;
            router.push("/login");
        } catch (error) {
            if (error.response && error.response.status === 422) {
                console.log(error.response.data)
                authErrors.value = error.response.data.errors;
            }
        } finally {
            loading.value = false;
        }
    };
    // ===========reset properties==========
    const resetFields = () => {
        authErrors.value = [];
        authStatus.value = null;
        loading.value = false;
    };
    return {
        authUser,
        authErrors,
        getUser,
        handleLogin,
        handleRegister,
        handleLogout,
        handleForgetPassword,
        resetFields,
        handleResetPassword,
        authStatus,
        loading,
    };
});
```

Inside Login.vue

```
<script setup>
import { ref } from "vue";
import { useAuthStore } from "../../stores/auth";

// Access the authentication store
const authStore = useAuthStore();

// Properties
const form = ref({
  email: "",
```

```
  password: "",
});
</script>

<template>
  <v-container class="py-5">
    <v-row justify="center">
      <v-col cols="12" md="6">
        <v-card elevation="8" class="border-0">
          <v-card-title class="bg-primary white--text text-center">
            <h4 class="mx-auto">Sign In</h4>
          </v-card-title>
          <v-card-text class="pa-4">
            <v-form @submit.prevent="authStore.handleLogin(form)">
              <v-text-field v-model="form.email" label="Email address"
type="email" placeholder="Enter your email"
                id="email" variant="outlined" class="mb-4"
                :error-messages="authStore.authErrors.email?.[0] || ''"></v-text-
field>

              <v-text-field v-model="form.password" label="Password"
type="password" placeholder="Enter your password"
                id="exampleInputPassword1" variant="outlined" class="mb-4"
                :error-messages="authStore.authErrors.password?.[0] || ''"></v-
text-field>

              <div class="d-flex justify-space-between mb-4">
                <router-link :to="{ name: 'ForgetPassword' }" class="text-
decoration-none">
                  Forgot Password?
                </router-link>
              </div>

              <v-btn type="submit" color="primary" block class="mb-3"
:loading="authStore.loading"
                :disabled="authStore.loading">
                {{ authStore.loading ? "Loading..." : "Login" }}
              </v-btn>
            </v-form>

            <div class="text-center">
              <p class="mb-0">
                Not a member yet?
                <router-link :to="{ name: 'Register' }" class="text-decoration-
none primary--text font-weight-bold">
```

```
                Sign Up
              </router-link>
            </p>
          </div>
        </v-card-text>
      </v-card>
    </v-col>
  </v-row>
 </v-container>
</template>

<style scoped>
/* Optional: Add custom styles if needed */
.bg-primary {
  background-color: #1976d2 !important;
  /* Vuetify primary color */
}

.white--text {
  color: white !important;
}
</style>
```

Inside Register.vue

```
<script setup>
import { ref } from "vue";
import { useAuthStore } from "../../stores/auth";

// Access the authentication store
const authStore = useAuthStore();

// Properties
const form = ref({
    name: "",
    email: "",
    password: "",
    password_confirmation: "",
});
</script>

<template>
    <v-container class="py-5">
        <v-row justify="center">
            <v-col cols="12" md="6">
```

```
                <v-card elevation="8" class="border-0">
                    <v-card-title class="bg-primary white--text text-center">
                        <h4 class="mx-auto">Register</h4>
                    </v-card-title>
                    <v-card-text class="pa-4">
                        <v-form @submit.prevent="authStore.handleRegister(form)">
                            <v-text-field v-model="form.name" label="Name"
type="text"
                                placeholder="Enter your full name" id="name"
variant="outlined" class="mb-4"
                                :error-messages="authStore.authErrors.name?.[0]
|| ''"></v-text-field>

                            <v-text-field v-model="form.email" label="Email
address" type="email"
                                placeholder="Enter your email" id="email"
variant="outlined" class="mb-4"
                                :error-messages="authStore.authErrors.email?.[0]
|| ''"></v-text-field>

                            <v-text-field v-model="form.password"
label="Password" type="password"
                                placeholder="Enter your password" id="password"
variant="outlined" class="mb-4"
                                :error-
messages="authStore.authErrors.password?.[0] || ''"></v-text-field>

                            <v-text-field v-model="form.password_confirmation"
label="Password Confirmation"
                                type="password" placeholder="Re-enter your
password" id="password_confirmation"
                                variant="outlined" class="mb-4"></v-text-field>

                            <v-btn type="submit" color="primary" block class="mb-
3" :loading="authStore.loading"
                                :disabled="authStore.loading">
                                {{ authStore.loading ? "Loading..." : "Register"
}}
                            </v-btn>
                        </v-form>

                        <div class="text-center">
                            <p class="mb-0">
                                Already have an account?
                                <router-link :to="{ name: 'Login' }"
```

```
                                    class="text-decoration-none primary--text
font-weight-bold">
                                    Sign In
                                </router-link>
                            </p>
                        </div>
                    </v-card-text>
                </v-card>
            </v-col>
        </v-row>
    </v-container>
</template>

<style scoped>
/* Optional: Add custom styles if needed */
.bg-primary {
    background-color: #1976d2 !important;
    /* Vuetify primary color */
}

.white--text {
    color: white !important;
}
</style>
```

Inside ForgetPassword.vue

```
<script setup>
import { ref } from "vue";
import { useAuthStore } from "../../stores/auth";

// Access the auth store
const authStore = useAuthStore();

// Reactive properties
const email = ref("");

// Handle the forget password form submission
const handleForgetPassword = async () => {
    // Call the store method to handle the forgot password logic
    await authStore.handleForgetPassword(email.value);
    // Reset email and display success message
    email.value = "";
};
</script>
```

```
<template>
    <v-container class="py-5">
        <v-row justify="center">
            <v-col cols="12" md="6">
                <v-alert v-if="authStore.authStatus" type="success" class="mb-4"
text>
                    {{ authStore.authStatus }}
                </v-alert>
                <v-card elevation="8" class="border-0">
                    <v-card-title class="bg-primary white--text text-center">
                        <h4 class="mx-auto">Forget Password</h4>
                    </v-card-title>
                    <v-card-text class="pa-4">
                        <v-form @submit.prevent="handleForgetPassword">
                            <v-text-field v-model="email" label="Email address"
type="email"
                                placeholder="Enter your email" id="email"
variant="outlined" class="mb-4"
                                :error-messages="authStore.authErrors.email?.[0]
|| ''"></v-text-field>

                            <v-btn type="submit" color="primary" block class="mb-
3" :loading="authStore.loading"
                                :disabled="authStore.loading">
                                {{ authStore.loading ? "Loading..." : "Submit" }}
                            </v-btn>
                        </v-form>
                    </v-card-text>
                </v-card>
            </v-col>
        </v-row>
    </v-container>
</template>

<style scoped>
/* Optional: Add custom styles if needed */
.bg-primary {
    background-color: #1976d2 !important;
    /* Vuetify primary color */
}

.white--text {
    color: white !important;
}
```

```
</style>
```

Inside ResetPassword.vue

```
<script setup>
import { ref } from "vue";
import { useAuthStore } from "../../stores/auth";
import { useRoute } from "vue-router";

// Access the auth store and route
const authStore = useAuthStore();
const route = useRoute();

// Reactive properties
const form = ref({
    password: "",
    password_confirmation: "",
    email: route.query.email,
    token: route.params.token,
});
</script>

<template>
    <v-container class="py-5">
        <v-row justify="center">
            <v-col cols="12" md="6">
                <v-alert v-if="authStore.authStatus" type="success" class="mb-4"
text>
                    {{ authStore.authStatus }}
                </v-alert>
                <v-card elevation="8" class="border-0">
                    <v-card-title class="bg-primary white--text text-center">
                        <h4 class="mx-auto">Reset Password</h4>
                    </v-card-title>
                    <v-card-text class="pa-4">
                        <v-form
@submit.prevent="authStore.handleResetPassword(form)">
                            <v-text-field v-model="form.password" label="New
Password" type="password"
                                placeholder="Enter your password" id="password"
variant="outlined" class="mb-4"
                                :error-
messages="authStore.authErrors.password?.[0] || authStore.authErrors.email?.[0]
|| ''
                                "></v-text-field>
```

```
                            <v-text-field v-model="form.password_confirmation"
label="Re-enter Password" type="password"
                                placeholder="Re-enter your password"
id="password_confirmation" variant="outlined"
                                class="mb-4"></v-text-field>

                            <v-btn type="submit" color="primary" block class="mb-
3" :loading="authStore.loading"
                                :disabled="authStore.loading">
                                {{ authStore.loading ? "Loading..." : "Submit" }}
                            </v-btn>
                        </v-form>
                    </v-card-text>
                </v-card>
            </v-col>
        </v-row>
    </v-container>
</template>

<style scoped>
/* Optional: Add custom styles if needed */
.bg-primary {
    background-color: #1976d2 !important;
    /* Vuetify primary color */
}

.white--text {
    color: white !important;
}
</style>
```

g) Setup Dashboard layouts.

a) Create backend/layouts/Dashboard.vue

```
<script setup>
import { ref } from "vue";
import { useRoute } from "vue-router";

// Drawer state
const drawer = ref(false);

// Navigation items
const navItems = [
    { title: "Dashboard", icon: "mdi-home", route: "/admin/home" },
```

```
    { title: "Orders", icon: "mdi-cart", route: "/admin/orders" },
    { title: "Products", icon: "mdi-package", route: "/admin/products" },
    { title: "Reports", icon: "mdi-chart-line", route: "/admin/reports" },
];

// Get current route
const route = useRoute();

// Check if the item is active
const isActive = (itemRoute) => {
    return route.path === itemRoute;
};

// Methods
const logout = () => {
    console.log("Logged out");
};
</script>

<template>
    <!-- App Bar -->
    <v-app-bar app elevation="2" color="white" class="px-4">
        <v-app-bar-nav-icon @click="drawer = !drawer">
            <v-icon color="teal darken-2">mdi-menu</v-icon>
        </v-app-bar-nav-icon>
        <v-toolbar-title class="teal--text text--darken-2 font-weight-medium">
            E-Commerce Hub
        </v-toolbar-title>
        <v-spacer></v-spacer>
        <v-btn icon class="mx-1">
            <v-icon color="teal darken-2">mdi-bell</v-icon>
        </v-btn>
        <v-btn icon class="mx-1" @click="logout">
            <v-icon color="teal darken-2">mdi-logout</v-icon>
        </v-btn>
    </v-app-bar>

    <!-- Navigation Drawer -->
    <v-navigation-drawer v-model="drawer" app temporary color="teal darken-3">
        <v-list dense class="pt-4">
            <v-list-item v-for="item in navItems" :key="item.title"
:to="item.route" :active="isActive(item.route)"
                class="d-flex align-center pa-2" active-class="active-item">
                <v-icon color="white" class="mr-3">{{ item.icon }}</v-icon>
                <span class="white--text text-subtitle-1">{{ item.title }}</span>
```

```
            </v-list-item>
        </v-list>
    </v-navigation-drawer>
</template>



<style scoped>
.active-item {
    background-color: rgba(255, 255, 255, 0.2) !important;
    color: white !important;
}

.active-item .v-icon {
    color: white !important;
}
</style>
```

b) Inside backend/layouts/BackendLayout.vue

```
<script setup>
import Dashboard from './Dashboard.vue';
</script>
<template>
    <v-app>
        <!-- Import Dashboard -->
        <Dashboard/>
        <!-- Main Content Area -->
        <v-main>
            <main>
                <!-- Child routes will be injected here -->
                <router-view />
            </main>
        </v-main>
    </v-app>
</template>
```

c) Inside backend/Home.vue

```
<script setup>
</script>

<template>
    <v-container class="py-6">
        <h1 class="teal--text text--darken-2">Dashboard</h1>
        <p>Welcome to the Dashboard page!</p>
```

```
    </v-container>
</template>
```

d) For Crud Operation

a) Make migration and model

```
use HasFactory;
protected $fillable = ['slug', 'name', 'status', 'image'];

// use slug instead of id
public function getRouteKeyName()
{
    return 'slug';
}
```

b) Make resource

```
php artisan make:resource CategoryResource
```

c) Inside CategoryResource

```
public function toArray(Request $request): array
    {
        return [
            'id' => $this->id,
            'slug' => $this->slug,
            'name' => $this->name,
            'status' => $this->status,
            'image' => $this->image ? asset('storage/' . $this->image) : null,
            'created_at' => $this->created_at->diffForHumans(),
            'updated_at' => $this->updated_at->toDateTimeString(),
        ];
    }
```

d) Make resource controller.

```
Make controller php artisan make:controller Api/CategoryController --model=Category
```

e) Inside CategoryController

```
/**
     * Display a listing of the resource.
     */
    public function index(Request $request)
    {
```

```
        $perPage = $request->query('per_page', 10);
        $search = $request->query('search', '');
        $status = $request->query('status', ''); // New status filter
        $sortBy = $request->query('sort_by', 'created_at');
        $sortDirection = $request->query('sort_direction', 'desc');

        $query = Category::query();

        if ($search) {
            $query->where(function ($q) use ($search) {
                $q->where('name', 'like', '%' . $search . '%')
                    ->orWhere('status', 'like', '%' . $search . '%');
            });
        }

        if ($status) {
            $query->where('status', $status); // Exact match for status
        }

        $allowedSorts = ['name', 'status', 'updated_at']; // Updated to include
updated_at
        $sortBy = in_array($sortBy, $allowedSorts) ? $sortBy : 'updated_at';
        $sortDirection = in_array(strtolower($sortDirection), ['asc', 'desc']) ?
$sortDirection : 'desc';

        $query->orderBy($sortBy, $sortDirection);
        $categories = $query->paginate($perPage);

        return CategoryResource::collection($categories);
    }



    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name' => 'required|string|max:255|unique:categories,name',
            'status' => 'required|in:active,inactive',
            'image' => 'required|image|max:2048', // Max 2MB
        ]);

        if ($validator->fails()) {
```

```
            return response()->json([
                'errors' => $validator->errors()
            ], 422);
        }

        $data = $request->all();
        if ($request->hasFile('image')) {
            $data['image'] = $request->file('image')->store('categories',
'public');
        }

        $category = Category::create($data);

        return new CategoryResource($category);
    }

    /**
     * Display the specified resource.
     */
    public function show(Category $category)
    {
        return new CategoryResource($category);
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, Category $category)
    {
        $validator = Validator::make($request->all(), [
            'name' => 'sometimes|required|string|max:255',
            'status' => 'sometimes|required|in:active,inactive',
            'image' => 'nullable|image|max:2048',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'errors' => $validator->errors()
            ], 422);
        }

        $data = $request->all();
        if ($request->hasFile('image')) {
            if ($category->image) {
                Storage::disk('public')->delete($category->image);
```

```
            }
            $data['image'] = $request->file('image')->store('categories',
'public');
        }

        $category->update($data);

        return new CategoryResource($category);
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Category $category)
    {
        if ($category->image) {
            Storage::disk('public')->delete($category->image);
        }
        $category->delete();

        return response()->json(null, 204);
    }

    /**
     * Delete multiple categories
     */
    public function deleteMultiple(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'slugs' => 'required|array',
            'slugs.*' => 'required|string|exists:categories,slug',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'errors' => $validator->errors()
            ], 422);
        }

        $slugs = $request->input('slugs');

        DB::transaction(function () use ($slugs) {
            $categories = Category::whereIn('slug', $slugs)->get();

            foreach ($categories as $category) {
```

```
                if ($category->image) {
                    Storage::disk('public')->delete($category->image);
                }
                $category->delete();
            }
        });

        return response()->json(null, 204);
    }
```

f) Api.php

```
Route::group(['prefix' => 'categories'], function () {
    Route::get('/', [CategoryController::class, 'index'])-
>name('categories.index');
    Route::post('/', [CategoryController::class, 'store'])-
>name('categories.store');
    Route::delete('/multiple', [CategoryController::class, 'deleteMultiple'])-
>name('categories.deleteMultiple'); // Moved up
    Route::get('/{category}', [CategoryController::class, 'show'])-
>name('categories.show');
    Route::put('/{category}', [CategoryController::class, 'update'])-
>name('categories.update');
    Route::delete('/{category}', [CategoryController::class, 'destroy'])-
>name('categories.destroy');
});
```

g) Inside src/stores/category.js

```
// src/stores/category.js
import { defineStore } from "pinia";
import { ref } from "vue";
import axios from "axios"; // Assuming your axios configuration is here

export const useCategoryStore = defineStore("category", () => {
    // ========= State =========
    const categories = ref([]); // List of all categories
    const currentCategory = ref(null); // Single category for view/edit
    const categoryErrors = ref([]); // Validation or API errors
    const loading = ref(false); // Loading state for async operations
    const pagination = ref({}); // Pagination metadata

    // ======= Fetch all categories (View/List) =======
    const fetchCategories = async (page = 1, perPage = 10, search = '', sortBy =
'updated_at', sortDirection = 'desc', status = '') => {
        try {
            loading.value = true;
```

```
            // Start with basic params
            const params = {
                page,
                per_page: perPage,
                sort_by: sortBy,
                sort_direction: sortDirection,
            };

            // Add 'search' to params if it's provided
            if (search) {
                params.search = encodeURIComponent(search);
            }

            // Add 'status' to params if it's provided
            if (status) {
                params.status = encodeURIComponent(status);
            }

            // Build the query string
            const query = new URLSearchParams(params).toString();

            // Fetch data using axios
            const response = await axios.get(`/api/categories?${query}`);
            console.log('Full API Response:', response.data);

            // Set the categories and pagination values
            categories.value = response.data.data || [];
            pagination.value = response.data.meta;

            return response;  // Return full response for consistency
        } catch (error) {
            console.error("Fetch error:", error.response?.data || error.message);

            // Reset values on error
            categories.value = [];
            pagination.value = { total: 0, current_page: 1, last_page: 1,
per_page: perPage };

            throw error;
        } finally {
            loading.value = false;
        }
    };
```

```
    // ======= Fetch a single category (View/Edit) =======
    const fetchCategory = async (slug) => {
        try {
            loading.value = true;
            const response = await axios.get(`/api/categories/${slug}`);
            currentCategory.value = response.data.data;
        } catch (error) {
            console.error("Error fetching category:", error);
            categoryErrors.value = error.response?.data?.errors || ["Category not
found"];
            currentCategory.value = null;
        } finally {
            loading.value = false;
        }
    };

    // ======= Store/Create a new category =======
    const storeCategory = async (categoryData) => {
        try {
            loading.value = true;
            categoryErrors.value = [];
            const response = await axios.post("/api/categories", categoryData, {
                headers: { "Content-Type": "multipart/form-data" },
            });
            const newCategory = response.data.data;
            return newCategory;
        } catch (error) {
            if (error.response && error.response.status === 422) {
                categoryErrors.value = error.response.data.errors;
            } else {
                console.error("Error creating category:", error);
            }
            throw error;
        } finally {
            loading.value = false;
        }
    };

    // ======= Update an existing category =======
    const updateCategory = async (slug, categoryData) => {
        try {
            loading.value = true;
            categoryErrors.value = [];
            categoryData.append('_method', 'PUT');
```

```
            const response = await axios.post(`/api/categories/${slug}`,
categoryData, {
                headers: { "Content-Type": "multipart/form-data" },
            });
            const updatedCategory = response.data.data;
            return updatedCategory; // Return updated category for local update
        } catch (error) {
            if (error.response && error.response.status === 422) {
                categoryErrors.value = error.response.data.errors;
            } else {
                console.error("Error updating category:", error);
            }
            throw error;
        } finally {
            loading.value = false;
        }
    };

    // ======= Delete a category =======
    const deleteCategory = async (slug) => {
        try {
            loading.value = true;
            await axios.delete(`/api/categories/${slug}`);
            return slug;
        } catch (error) {
            console.error("Delete error:", error.response?.data ||
error.message);
            throw error;
        } finally {
            loading.value = false;
        }
    };

    // ==========Delete multiple categories=========
    const deleteMultipleCategories = async (slugs) => {
        try {
            loading.value = true;
            await axios.delete("/api/categories/multiple", {
                data: { slugs } // Ensure slugs are sent in the request body
            });
        } catch (error) {
            console.error("Delete multiple error:", error.response?.data ||
error.message);
            throw error;
        } finally {
```

```
            loading.value = false;
        }
    };

    // ======= Reset errors =======
    const resetErrors = () => {
        categoryErrors.value = [];
    };

    // ======= Return state and actions =======
    return {
        categories,
        currentCategory,
        categoryErrors,
        loading,
        pagination,
        fetchCategories,
        fetchCategory,
        storeCategory,
        updateCategory,
        deleteCategory,
        deleteMultipleCategories,
        resetErrors,
    };
});
```

a) Inside src/components/backend/category/Category.vue

```
<script setup>
import { ref, watch } from "vue";
import { useCategoryStore } from "../../../stores/category";
import debounce from "lodash/debounce";

const categoryStore = useCategoryStore();

// Table state
const page = ref(1);
const itemsPerPage = ref(10);
const search = ref("");
const sortBy = ref([{ key: "updated_at", order: "desc" }]);
const statusFilter = ref("");
const selectedCategories = ref([]); // New state for selected categories

// Image Modal state
const showImageModal = ref(false);
```

```
const selectedImage = ref(null);
const hoverImage = ref(false);

// Add/Edit Modal state
const showAddModal = ref(false);
const newCategory = ref({ name: "", status: "active", image: null });
const addForm = ref(null);
const serverErrors = ref({ name: [], status: [], image: [] });

// Edit Modal trigger
const showEditModal = ref(false);

// Validation rules
const rules = {
    name: [
        (v) => !!v || "Name is required",
        (v) => (v && v.length <= 255) || "Name must be less than 255 characters",
    ],
    status: [
        (v) => !!v || "Status is required",
        (v) => ["active", "inactive"].includes(v) || "Status must be 'active' or
'inactive'",
    ],
    image: [
        (v) => !!v || "Image is required",
        (v) => !v || v.size <= 2 * 1024 * 1024 || "Image must be less than 2MB",
    ],
};

// Table headers with checkbox
const headers = [
    {
        title: "",
        key: "data-table-select",
        sortable: false,
        width: "50px"
    },
    { title: "Image", key: "image", sortable: false },
    { title: "Name", key: "name", sortable: true },
    { title: "Status", key: "status", sortable: true },
    { title: "Created At", key: "created_at", sortable: true },
    { title: "Actions", key: "actions", sortable: false },
];

// Load items from server
```

```
const loadItems = debounce(async (options) => {
    page.value = options.page;
    itemsPerPage.value = options.itemsPerPage;
    sortBy.value = options.sortBy.length ? options.sortBy : [{ key: "created_at",
order: "desc" }];

    const sort = sortBy.value[0];
    await categoryStore.fetchCategories(
        page.value,
        itemsPerPage.value,
        search.value,
        sort.key,
        sort.order,
        statusFilter.value
    );
}, 300);

// Debounced search and status filter
const updateFilters = debounce(() => {
    page.value = 1;
    loadItems({ page: page.value, itemsPerPage: itemsPerPage.value, sortBy:
sortBy.value });
}, 500);

// Watch filters
watch([search, statusFilter], updateFilters);

// Image modal handlers
const openImageModal = (imageUrl) => {
    selectedImage.value = imageUrl;
    showImageModal.value = true;
};

// Add modal handlers
const resetNewCategory = () => ({
    name: "",
    status: "active",
    image: null,
});

const openAddModal = () => {
    newCategory.value = resetNewCategory();
    categoryStore.resetErrors();
    serverErrors.value = { name: [], status: [], image: [] };
    showAddModal.value = true;
```

```
};

const closeAddModal = () => {
    showAddModal.value = false;
};

// Edit modal handlers
const openEditModal = async (slug) => {
    try {
        await categoryStore.fetchCategory(slug);
        newCategory.value = {
            slug: slug,
            name: categoryStore.currentCategory.name,
            status: categoryStore.currentCategory.status,
            image: null
        };
        categoryStore.resetErrors();
        serverErrors.value = { name: [], status: [], image: [] };
        showEditModal.value = true;
    } catch (error) {
        console.error('Error opening edit modal:', error);
    }
};

const closeEditModal = () => {
    showEditModal.value = false;
    newCategory.value = resetNewCategory();
};

// Submit new category
const submitCategory = async () => {
    const { valid } = await addForm.value.validate();
    if (!valid) return;

    const formData = new FormData();
    formData.append("name", newCategory.value.name);
    formData.append("status", newCategory.value.status);
    if (newCategory.value.image) formData.append("image",
newCategory.value.image);

    try {
        await categoryStore.storeCategory(formData);
        showAddModal.value = false;
        page.value = 1;
        sortBy.value = [{ key: "created_at", order: "desc" }];
```

```
        await loadItems({ page: page.value, itemsPerPage: itemsPerPage.value,
sortBy: sortBy.value });
    } catch (error) {
        if (categoryStore.categoryErrors) {
            serverErrors.value = {
                name: categoryStore.categoryErrors.name || [],
                status: categoryStore.categoryErrors.status || [],
                image: categoryStore.categoryErrors.image || [],
            };
        }
    }
};

// Submit edited category
const submitEditCategory = async () => {
    const { valid } = await addForm.value.validate();
    if (!valid) return;

    const formData = new FormData();
    formData.append("name", newCategory.value.name);
    formData.append("status", newCategory.value.status);
    if (newCategory.value.image) formData.append("image",
newCategory.value.image);

    try {
        await categoryStore.updateCategory(newCategory.value.slug, formData);
        showEditModal.value = false;
        await loadItems({ page: page.value, itemsPerPage: itemsPerPage.value,
sortBy: sortBy.value });
    } catch (error) {
        if (categoryStore.categoryErrors) {
            serverErrors.value = {
                name: categoryStore.categoryErrors.name || [],
                status: categoryStore.categoryErrors.status || [],
            };
        }
    }
};

// Status chip color
const getStatusColor = (status) => (status === "active" ? "green" : "red");

// CRUD operations
const editCategory = (slug) => {
    openEditModal(slug);
```

```
};

const deleteCategory = async (slug) => {
    if (!confirm("Are you sure you want to delete this category?")) return;
    try {
        await categoryStore.deleteCategory(slug);
        await loadItems({ page: page.value, itemsPerPage: itemsPerPage.value,
sortBy: sortBy.value });
    } catch (error) {
        alert("Failed to delete category. Please try again.");
        await loadItems({ page: page.value, itemsPerPage: itemsPerPage.value,
sortBy: sortBy.value });
    }
};

// Multiple delete handler
const deleteMultipleCategories = async () => {
    if (!confirm("Are you sure you want to delete the selected categories?"))
return;
    try {
        await categoryStore.deleteMultipleCategories(selectedCategories.value);
        selectedCategories.value = []; // Clear selection
        await loadItems({ page: page.value, itemsPerPage: itemsPerPage.value,
sortBy: sortBy.value });
    } catch (error) {
        alert("Failed to delete categories. Please try again.");
        await loadItems({ page: page.value, itemsPerPage: itemsPerPage.value,
sortBy: sortBy.value });
    }
};
</script>

<template>
    <v-container class="py-6">
        <!-- Filters and Search -->
        <v-card class="pa-4 mb-4 rounded-lg elevation-2">
            <v-row align="center" justify="space-between">
                <v-col cols="12" sm="6" md="4">
                    <v-text-field v-model="search" label="Search Categories"
prepend-inner-icon="mdi-magnify"
                        variant="outlined" clearable density="comfortable"
class="rounded-lg" />
                </v-col>
                <v-col cols="12" sm="6" md="4">
                    <v-select v-model="statusFilter" :items="[
```

```
                        { title: 'All', value: '' },
                        { title: 'Active', value: 'active' },
                        { title: 'Inactive', value: 'inactive' }
                    ]" label="Filter by Status" prepend-inner-icon="mdi-filter"
variant="outlined"
                        density="comfortable" class="rounded-lg" clearable />
                </v-col>
                <v-spacer />
                <v-col cols="12" sm="auto">
                    <v-btn color="teal-darken-2" dark class="rounded-lg px-4 mr-
2" @click="openAddModal">
                        <v-icon left>mdi-plus</v-icon> Add Category
                    </v-btn>
                    <v-btn color="red-darken-2" dark class="rounded-lg px-4"
                           :disabled="selectedCategories.length === 0"
                           @click="deleteMultipleCategories">
                        <v-icon left>mdi-delete</v-icon> Delete Selected
                    </v-btn>
                </v-col>
            </v-row>
        </v-card>

        <!-- Data Table -->
        <v-card class="rounded-lg elevation-2">
            <v-data-table-server
                v-model:items-per-page="itemsPerPage"
                :headers="headers"
                :items="categoryStore.categories"
                :items-length="categoryStore.pagination.total || 0"
                :loading="categoryStore.loading"
                item-value="slug"
                v-model:sort-by="sortBy"
                v-model="selectedCategories"
                show-select
                :items-per-page-options="[5, 10, 20, 50]"
                @update:options="loadItems">
                <template v-slot:item.image="{ item }">
                    <v-img v-if="item.image" :src="item.image" height="40"
width="40"
                        class="my-2 rounded-lg cursor-pointer"
@click="openImageModal(item.image)" />
                    <span v-else>No Image</span>
                </template>
                <template v-slot:item.status="{ item }">
```

```
                    <v-chip :color="getStatusColor(item.status)" small
class="text-uppercase font-weight-bold">
                        {{ item.status }}
                    </v-chip>
                </template>
                <template v-slot:item.actions="{ item }">
                    <v-btn icon color="warning" size="small"
@click="editCategory(item.slug)">
                        <v-icon>mdi-pencil</v-icon>
                    </v-btn>
                    <v-btn icon color="red-darken-2" size="small" class="ms-2"
@click="deleteCategory(item.slug)">
                        <v-icon>mdi-delete</v-icon>
                    </v-btn>
                </template>
            </v-data-table-server>
        </v-card>

        <!-- Image Modal -->
        <v-dialog v-model="showImageModal" max-width="500">
            <v-card class="rounded-lg elevation-3">
                <v-card-title class="d-flex align-center" style="background:
linear-gradient(45deg, #00695c, #00897b)">
                    <span class="text-h6 font-weight-bold text-white">Category
Image</span>
                    <v-spacer />
                    <v-btn icon @click="showImageModal = false" variant="plain">
                        <v-icon color="white">mdi-close</v-icon>
                    </v-btn>
                </v-card-title>
                <v-card-text class="text-center py-4">
                    <v-img :src="selectedImage" max-height="400" contain
class="mx-auto rounded-lg" />
                </v-card-text>
            </v-card>
        </v-dialog>

        <!-- Add Category Modal -->
        <v-dialog v-model="showAddModal" max-width="500" persistent
transition="dialog-bottom-transition">
            <v-card rounded="xl" elevation="12">
                <v-card-title class="py-4 px-6" style="background: linear-
gradient(45deg, #00695c, #00897b)">
                    <v-row align="center">
                        <v-col>
```

```
                            <span class="text-h6 font-weight-medium text-
white">Add New Category</span>
                        </v-col>
                        <v-col cols="auto">
                            <v-btn icon @click="closeAddModal" variant="text"
color="white" class="mt-n1">
                                <v-icon>mdi-close</v-icon>
                            </v-btn>
                        </v-col>
                    </v-row>
                </v-card-title>
                <v-card-text class="pa-6">
                    <v-form ref="addForm" @submit.prevent="submitCategory"
class="d-flex flex-column gap-4">
                        <!-- category name -->
                        <v-text-field v-model="newCategory.name" label="Category
Name" prepend-icon="mdi-tag"
                            density="comfortable" variant="outlined" color="teal-
darken-2" class="rounded-lg mb-4"
                            :rules="rules.name" required :error-
messages="serverErrors.name" />
                            <!-- select category status -->
                        <v-select v-model="newCategory.status" :items="[
                            { title: 'Active', value: 'active' },
                            { title: 'Inactive', value: 'inactive' }
                        ]" label="Status" prepend-icon="mdi-toggle-switch"
density="comfortable" variant="outlined"
                            color="teal-darken-2" class="rounded-lg mb-4"
:rules="rules.status" required
                            :error-messages="serverErrors.status" />
                            <!-- upload category image -->
                        <v-file-input v-model="newCategory.image" label="Category
Image" prepend-icon="mdi-image"
                            accept="image/*" density="comfortable"
variant="outlined" class="rounded-lg mb-4"
                            color="teal-darken-2" :error-
messages="serverErrors.image" />
                            <!-- create new category -->
                        <v-btn type="submit" color="teal-darken-2" variant="flat"
size="large" block
                            :loading="categoryStore.loading" class="mt-4 rounded-
lg" elevation="2">
                            Save Category
                        </v-btn>
                    </v-form>
```

```
                </v-card-text>
            </v-card>
        </v-dialog>

        <!-- Edit Category Modal -->
        <v-dialog v-model="showEditModal" max-width="500" persistent
transition="dialog-bottom-transition">
            <v-card rounded="xl" elevation="12">
                <v-card-title class="py-4 px-6" style="background: linear-
gradient(45deg, #00695c, #00897b)">
                    <v-row align="center">
                        <v-col>
                            <span class="text-h6 font-weight-medium text-
white">Edit Category</span>
                        </v-col>
                        <v-col cols="auto">
                            <v-btn icon @click="closeEditModal" variant="text"
color="white" class="mt-n1">
                                <v-icon>mdi-close</v-icon>
                            </v-btn>
                        </v-col>
                    </v-row>
                </v-card-title>
                <v-card-text class="pa-6">
                    <v-form ref="addForm" @submit.prevent="submitEditCategory"
class="d-flex flex-column gap-4">
                        <!-- category name -->
                        <v-text-field v-model="newCategory.name" label="Category
Name" prepend-icon="mdi-tag"
                            density="comfortable" variant="outlined" color="teal-
darken-2" class="rounded-lg mb-4"
                            :rules="rules.name" required :error-
messages="serverErrors.name" />
                            <!-- select category status -->
                        <v-select v-model="newCategory.status" :items="[
                            { title: 'Active', value: 'active' },
                            { title: 'Inactive', value: 'inactive' }
                        ]" label="Status" prepend-icon="mdi-toggle-switch"
density="comfortable" variant="outlined"
                            color="teal-darken-2" class="rounded-lg mb-4"
:rules="rules.status" required
                            :error-messages="serverErrors.status" />
                            <!-- upload category image -->
                        <v-file-input v-model="newCategory.image" label="Category
Image" prepend-icon="mdi-image"
```

```
                            accept="image/*" density="comfortable"
variant="outlined" class="rounded-lg mb-4"
                            color="teal-darken-2" :error-
messages="serverErrors.image" />
                            <!-- update category -->
                        <v-btn type="submit" color="teal-darken-2" variant="flat"
size="large" block
                            :loading="categoryStore.loading" class="mt-4 rounded-
lg" elevation="2">
                            Update Category
                        </v-btn>
                    </v-form>
                </v-card-text>
            </v-card>
        </v-dialog>
    </v-container>
</template>

<style scoped>
.cursor-pointer {
    cursor: pointer;
}
</style>
```