<script setup lang="ts">

import { ref, computed, onMounted } from 'vue';

import { useRoute } from 'vue-router';

import Header from './Header.vue';

import Footer from './Footer.vue';

import carsData from '@/assets/cars.json';

interface Car {

  \_id: string;

  createdAt: string;

  updatedAt: string;

  image: string;

  brand: string;

  model: string;

  year: number;

  place: string;

  number: number;

  \_\_v: string;

  date: string;

  userId: string;

}

const route = useRoute();

const name = ref('Guest');

const cars = ref<Car[]>([]);

const currentPage = ref(1);

const pageSize = 6;

const totalPages = computed(() => Math.ceil(cars.value.length / pageSize));

const paginatedCars = computed(() => {

  const start = (currentPage.value - 1) \* pageSize;

  return cars.value.slice(start, start + pageSize);

});

const paginationRange = computed(() => {

  const range: (number | string)[] = [];

  const total = totalPages.value;

  if (total <= 6) {

    for (let i = 1; i <= total; i++) range.push(i);

  } else {

    if (currentPage.value <= 4) {

      range.push(1, 2, 3, 4, '...', total);

    } else if (currentPage.value >= total - 3) {

      range.push(1, '...', total - 3, total - 2, total - 1, total);

    } else {

      range.push(1, '...', currentPage.value, '...', total);

    }

  }

  return range;

});

function goToPage(page: number | string) {

  if (typeof page === 'number' && page >= 1 && page <= totalPages.value) {

    currentPage.value = page;

  }

}

onMounted(async () => {

  const userStr = localStorage.getItem('loggedInUser');

  if (userStr) {

    const user = JSON.parse(userStr);

    name.value = user.name || 'Guest';

  }

  const stored = localStorage.getItem('cars');

  if (stored) {

    cars.value = JSON.parse(stored);

  } else {

     cars.value = carsData;

    localStorage.setItem('cars', JSON.stringify(cars.value));

  }

});

</script>

<template>

  <Header />

  <div class="container my-5">

    <div class="welcome-card shadow mb-4">

      <h1 class="greeting">Welcome, {{ name }}</h1>

      <p class="tagline">We're glad to have you back 🎉</p>

    </div>

    <div v-if="paginatedCars.length === 0" class="text-center mb-4">

      <p>No cars available.</p>

    </div>

    <div class="row g-4 mb-4">

      <div class="col-md-4" v-for="car in paginatedCars" :key="car.\_id">

        <div class="card h-100 shadow-sm">

          <img :src="`/images/${car.image}`" class="card-img-top" alt="Car Image" />

          <div class="card-body">

            <h5 class="card-title">{{ car.brand }} {{ car.model }} ({{ car.year }})</h5>

            <p class="card-text">

              <strong>Created:</strong> {{ car.createdAt }}<br />

              <strong>Updated:</strong> {{ car.updatedAt }}<br />

              <strong>Location:</strong> {{ car.place }}<br />

              <strong>Reg No:</strong> {{ car.number }}<br />

              <strong>Date:</strong> {{ car.date }}<br />

              <strong>User ID:</strong> {{ car.userId }}

            </p>

            <RouterLink class="btn btn-primary mt-2" :to="{ name: 'CarEdit', params: { id: car.\_id }, query: { name } }">

              Edit

            </RouterLink>

          </div>

        </div>

      </div>

    </div>

    <nav class="pagination-nav mb-4">

      <ul class="pagination justify-content-center">

        <li class="page-item" :class="{ disabled: currentPage === 1 }">

          <a class="page-link" href="#" @click.prevent="goToPage(currentPage - 1)">«</a>

        </li>

        <li class="page-item" v-for="page in paginationRange" :key="page" :class="{ active: page === currentPage, disabled: page === '...' }">

          <a class="page-link" href="#" @click.prevent="goToPage(page)">{{ page }}</a>

        </li>

        <li class="page-item" :class="{ disabled: currentPage === totalPages }">

          <a class="page-link" href="#" @click.prevent="goToPage(currentPage + 1)">»</a>

        </li>

      </ul>

    </nav>

  </div>

  <Footer />

</template>

<style scoped>

.greeting {

  font-size: 2.5rem;

  color: blue;

}

.tagline {

  font-size: 1.25rem;

  color: #555;

}

.welcome-card {

  background-color: white;

  padding: 2rem;

  border-radius: 1rem;

  text-align: center;

  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);

}

.card {

  border: none;

  border-radius: 0.5rem;

}

.card-img-top {

  height: 200px;

  object-fit: cover;

}

.btn-primary {

  background-color: #007bff;

  border-color: #007bff;

}

.btn-primary:hover {

  background-color: #0056b3;

}

.pagination-nav .page-link {

  color: #003399;

}

.pagination-nav .active .page-link {

  background-color: #007bff;

  border-color: #007bff;

  color: white;

}

.pagination-nav .disabled .page-link {

  color: #6c757d;

  pointer-events: none;

}

</style>

**Line-by-Line Code Explanation**

**📦 1. Imports**

ts

CopyEdit

import { ref, computed, onMounted } from 'vue';

* ref: To create reactive variables like name, cars, currentPage.
* computed: To create derived (calculated) reactive values (like pagination logic).
* onMounted: Vue lifecycle hook to run logic when the component is mounted.

ts

CopyEdit

import { useRoute } from 'vue-router';

* useRoute: Access current route data (like query params, if needed).

ts

CopyEdit

import Header from './Header.vue';

import Footer from './Footer.vue';

* Imports reusable layout components.

ts

CopyEdit

import carsData from '@/assets/cars.json';

* Loads fallback/default car data from a static JSON file.
* Used when localStorage is empty (first-time app usage).

**🚗 2. Interface for Car Object**

ts

CopyEdit

interface Car {

\_id: string;

createdAt: string;

updatedAt: string;

image: string;

brand: string;

model: string;

year: number;

place: string;

number: number;

\_\_v: string;

date: string;

userId: string;

}

* Type definition for each car object.
* Helps TypeScript enforce structure and prevents bugs.

**🔄 3. Route & State Variables**

ts

CopyEdit

const route = useRoute();

* Gives access to route query like ?name=Guest (not directly used here but available).

ts

CopyEdit

const name = ref('Guest');

* Stores the name of the logged-in user (default is 'Guest').

ts

CopyEdit

const cars = ref<Car[]>([]);

* Holds the array of car objects (either from localStorage or cars.json).

ts

CopyEdit

const currentPage = ref(1);

* Stores the currently active page for pagination.

ts

CopyEdit

const pageSize = 6;

* Number of car cards to display per page.

**📊 4. Computed Properties for Pagination**

**a. Total Pages**

ts

CopyEdit

const totalPages = computed(() => Math.ceil(cars.value.length / pageSize));

* Calculates how many pages are needed to display all cars.
* Uses Math.ceil to ensure no leftover cars are missed.

**b. Current Page Cars**

ts

CopyEdit

const paginatedCars = computed(() => {

const start = (currentPage.value - 1) \* pageSize;

return cars.value.slice(start, start + pageSize);

});

* Slices the full car list into only the cars to display on the current page.
* slice(start, end) creates a new array for that page.

**c. Pagination Range (Page Numbers)**

ts

CopyEdit

const paginationRange = computed(() => {

const range: (number | string)[] = [];

const total = totalPages.value;

if (total <= 6) {

for (let i = 1; i <= total; i++) range.push(i);

} else {

if (currentPage.value <= 4) {

range.push(1, 2, 3, 4, '...', total);

} else if (currentPage.value >= total - 3) {

range.push(1, '...', total - 3, total - 2, total - 1, total);

} else {

range.push(1, '...', currentPage.value, '...', total);

}

}

return range;

});

* Builds an array like: [1, 2, 3, '...', 10]
* Adds ellipsis ('...') when the number of pages is high to avoid clutter.
* Useful for a clean UI when there are many pages.

**🔁 5. Page Navigation**

ts

CopyEdit

function goToPage(page: number | string) {

if (typeof page === 'number' && page >= 1 && page <= totalPages.value) {

currentPage.value = page;

}

}

* Changes the current page number.
* Ignores invalid input like strings or out-of-bound numbers.
* Only accepts number pages between 1 and totalPages.

**🕒 6. onMounted Lifecycle Hook**

ts

CopyEdit

onMounted(async () => {

* This runs **after the component is inserted into the DOM**.

**a. Load User Name**

ts

CopyEdit

const userStr = localStorage.getItem('loggedInUser');

if (userStr) {

const user = JSON.parse(userStr);

name.value = user.name || 'Guest';

}

* Checks localStorage for a logged-in user.
* If found, parse it and set name.value to display on the UI.
* Defaults to 'Guest' if no name is found.

**b. Load Car Data**

ts

CopyEdit

const stored = localStorage.getItem('cars');

if (stored) {

cars.value = JSON.parse(stored);

} else {

cars.value = carsData;

localStorage.setItem('cars', JSON.stringify(cars.value));

}

* Loads car list from localStorage if available.
* If **not**, fallback to static JSON carsData, then **save** it to localStorage.
* This makes it a self-initializing app for the first visit.

**🧠 Summary Table**

| **Line/Section** | **Purpose** | **Why It’s Needed** |
| --- | --- | --- |
| ref() | Creates reactive variables | Reacts to changes in UI |
| computed() | Derives data like pages | Avoids recalculation |
| useRoute() | Reads route/query params | To show personalized greeting |
| interface Car | Type safety | Helps with TS autocomplete & error prevention |
| onMounted() | Load user and car data | Setup after component is ready |
| paginationRange | Clean page buttons | Improves UX for big lists |
| goToPage() | Changes pages | Handles user clicks |

This is the **UI** of your "Welcome Page" which displays:

* A greeting
* A list of cars (paginated)
* A link to edit each car
* Pagination controls

**✅ TEMPLATE STRUCTURE (Line-by-Line)**

**📌 Line 1**

html

CopyEdit

<template>

Starts the Vue component template — where your UI is defined.

**✅ HEADER COMPONENT**

html

CopyEdit

<Header />

* Reusable header (imported at the top of <script>).
* Usually contains logo, navigation, etc.

**📦 CONTAINER WRAPPER**

html

CopyEdit

<div class="container my-5">

* Bootstrap .container for layout.
* my-5 adds vertical margin (spacing above and below).

**🎉 WELCOME BOX**

html

CopyEdit

<div class="welcome-card shadow mb-4">

<h1 class="greeting">Welcome, {{ name }}</h1>

<p class="tagline">We're glad to have you back 🎉</p>

</div>

* Greets the logged-in user using name from ref.
* welcome-card, greeting, and tagline are styled via <style scoped>.
* shadow mb-4: Bootstrap classes for box-shadow and bottom margin.

🧠 Why? Personalized greeting enhances UX.

**❌ SHOW WHEN NO CARS AVAILABLE**

html

CopyEdit

<div v-if="paginatedCars.length === 0" class="text-center mb-4">

<p>No cars available.</p>

</div>

* If no cars are available for the current page, show a fallback message.
* Controlled by checking the paginatedCars array (computed).

🧠 Why? Avoid blank screen and inform the user.

**🚗 CAR CARDS LOOP**

html

CopyEdit

<div class="row g-4 mb-4">

* Bootstrap grid row.
* g-4: Adds gutter spacing between columns.
* mb-4: Bottom margin.

**➕ Per Car (Loop)**

html

CopyEdit

<div class="col-md-4" v-for="car in paginatedCars" :key="car.\_id">

* Loops over cars for **current page only** (paginatedCars).
* Uses Bootstrap 3-column layout (col-md-4 = 12/3 = 3 per row).
* :key="car.\_id" is important for Vue’s virtual DOM to track elements.

**🚘 Car Card UI**

html

CopyEdit

<div class="card h-100 shadow-sm">

* Bootstrap card component.
* h-100: Full height within its column.
* shadow-sm: Light shadow effect.

**🖼️ Image**

html

CopyEdit

<img :src="`/images/${car.image}`" class="card-img-top" alt="Car Image" />

* Loads car image from /images/ folder dynamically.
* car.image is just the filename.
* class="card-img-top" makes the image the top part of the card.

**📋 Car Info**

html

CopyEdit

<div class="card-body">

<h5 class="card-title">{{ car.brand }} {{ car.model }} ({{ car.year }})</h5>

<p class="card-text">

<strong>Created:</strong> {{ car.createdAt }}<br />

...

</p>

* Displays key car details like:
  + Brand/Model/Year
  + Created and Updated date
  + Location, Reg Number, Date, User ID
* Uses strong for field labels, and <br /> to separate each line.

🧠 Why? Clean presentation of car info.

**✏️ Edit Button**

html

CopyEdit

<RouterLink

class="btn btn-primary mt-2"

:to="{ name: 'CarEdit', params: { id: car.\_id }, query: { name } }"

>

Edit

</RouterLink>

* Navigates to the CarEdit page.
* Passes the car’s \_id as a route param (/car-edit/:id)
* Also passes name as a query parameter (optional).

🧠 Why? Allows user to edit that specific car.

**📄 PAGINATION NAVIGATION**

html

CopyEdit

<nav class="pagination-nav mb-4">

<ul class="pagination justify-content-center">

* Bootstrap pagination structure.
* justify-content-center: Centers pagination buttons.

**◀ Previous Button**

html

CopyEdit

<li class="page-item" :class="{ disabled: currentPage === 1 }">

<a class="page-link" href="#" @click.prevent="goToPage(currentPage - 1)">«</a>

</li>

* Shows "Previous" button.
* Disabled if already on page 1.
* Uses @click.prevent to call goToPage function without page reload.

**🔢 Page Numbers (with Ellipsis Support)**

html

CopyEdit

<li

class="page-item"

v-for="page in paginationRange"

:key="page"

:class="{ active: page === currentPage, disabled: page === '...' }"

>

<a class="page-link" href="#" @click.prevent="goToPage(page)">

{{ page }}

</a>

</li>

* Loops over paginationRange (which may include numbers and '...').
* active: Highlights the current page.
* disabled: Makes '...' unclickable.

🧠 Why? Shows only relevant page numbers with ellipses to reduce clutter.

**▶ Next Button**

html

CopyEdit

<li class="page-item" :class="{ disabled: currentPage === totalPages }">

<a class="page-link" href="#" @click.prevent="goToPage(currentPage + 1)">»</a>

</li>

* Shows "Next" button.
* Disabled if on the last page.

**📥 Footer**

html

CopyEdit

<Footer />

* Reusable Footer component.
* Appears at the bottom of the page.

**✅ Summary Table**

| **Section** | **Purpose** | **Why It's Important** |
| --- | --- | --- |
| <Header /> | Shows navigation/logo | Consistent layout |
| {{ name }} | Greets user | Personalization |
| v-if="paginatedCars.length === 0" | Fallback message | UX improvement |
| v-for="car in paginatedCars" | Loop through cars | Displays dynamic content |
| <RouterLink :to="{ name: 'CarEdit'... }" | Edit navigation | Enables editing |
| Pagination nav | Page switching | Better performance + UX |
| <Footer /> | Site footer | Completes layout |