

PROJEKTNI ZADATAK

WEB SAJT Book Tracker



Predmet : Praktikum korisnički interfejs Student: Jovana Davidović
Predmeni profesor : Uroš Dimitrijević Broj indeksa : 111/22

Sadržaj

Uvod	5
1. Korišćeno	5
2. Funkcionalnosti	5
ORGANIZACIJA	6
Dizajn router-a	6
Mapa sajta	7
Opis funkcionalnosti	7
Kodovi:	13
App.vue	13
Components	15
TheHeader.vue	15
TheFooter.vue	16
TheNav.vue	16
CoverImg.vue	18
BookForm.vue	18
FilterBooks.vue	21
TheFormComment.vue	23
BaseButton.vue	25
CommentItem.vue	26
CustomInput.vue	26
Views	28
AboutView.vue	28
AdminView.vue	29
HomeView.vue	32
LoginView.vue	36
MyProfileView.vue	38
SingleBookView.vue	42

SingUpView.vue.....	44
Store	48
auth	48
Index.js.....	48
getters.js	48
mutations.js	49
actions.js	49
books	50
index.js.....	50
getters.js	51
mutations.js	52
actions.js	53

13.06..2024. Jovana Davidović 111/22

Uvod

1. Korišćeno

Jezici korišćeni prilikom kreiranja sajta su : HTML, CSS, JavaScript, jQuery, PHP

Radi lakšeg dizajniranja sajta korišćen je template: Bootstrap

Radno okruženje : Visual Studio Code

2. Funkcionalnosti

Zajedničke funkcionalnosti svih stranica :

1. Prikaz navigacionih linkova

2. Burger taster (pomoć bootstrap-a, klikom na njega prikazuje sve linkove)

- Na index.php strani :

1. Dinamičko ispis knjiga iz mockAPI

2. Straničenje, pretraga, filtriranje i sortiranje

- Na /myprofil strani :

1. Svi komentari tog korisnika

2. Sve knjige u listi tog korisnika

- Na /login strani :

1. Forma za logovanje

- Na /signup strani :

1. Forma za registraciju

- Na /admin strani :

1. Tabela sa korisnicima (i onim koji su admin)

2. Tabela sa knjigama

3. Dodavanje nove knjige

4. Dodavanje novog žanra

ORGANIZACIJA

Dizajn router-a

```
[
  {
    path: '/',
    name: 'home',
    component: HomeView,
    meta: {transition: "fade",isUnauth: true,}
  },{
    path: '/about',
    name: 'about',
    component: AboutView,
    meta: {transition: "fade",isUnauth: true,}
  },{
    path: "/book/:id",
    name: "book",
    component: SingleBookView,
    props: true,
    meta: {transition: "slide-fade",isUnauth: true,},children: [{path: "comment",name: "comment",component:
TheForm,meta: {requiresAuth: true,},},],
  },{
    path: "/signup",
    name: "signup",
    component: SingUpView,
    meta: {transition: "slide-fade",isUnauth: true,requiresGuest: true}
  },{
    path: "/login",
    name: "login",
    component: LogInView,
    meta: {transition: "slide-fade",isUnauth: true,requiresGuest: true}
  },{
    path: "/myprofile",
    name: "myprofile",
    component: MyProfileView,
    meta: { transition: "slide-fade",requiresAuth: true, requiredRole: "User"}
  },{
    path: "/admin",
    name: "admin",
    component: AdminView,
    meta: { transition: "slide-fade",requiresAuth: true, requiredRole: "Admin"}
  }
]
```

Mapa sajta

Sitemap kod :

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>https://bookstracker.netlify.app/#/</loc>
    <lastmod>2024-06-30</lastmod>
    <changeFreq>daily</changeFreq>
    <priority>1.0</priority>
  </url>
  <url>
  </url>
</urlset>
```

Opis funkcionalnosti

Prikaz navigacionih linkova.

Meni je drugačije prikazan za korisnike koji nisu ulogovani, za koje su ulogovani i za one koji administratori.

Burger taster(pomoć bootstrap-a, klikom na njega prikazuje sve linkove)

List of books Logout Author User

List of books Logout Author Admin

List of books Singup Login Author

Dinamički prikaz knjiga iz mockAPI.

Klikom na knjigu ide se na posebnu stranu gde je prikaz celog opisa o knjizi kao i mogućnost dodavanja komentara (samo ako je korisnik ulogovan) i prikaz svih komentara za taj jedan post.

☐ Action
☐ Adventure
☐ Comedy
☐ Drama
☐ Fantasy
☐ Mystery
☐ Romance
☐ Tragedy



THE GREAT ADVENTURE
Al-Fayed's Rollercoaster Ride with Fulham FC
Tony Banks

The Great Adventure: Al-F...
Author: Tony Banks



The Romantic ESCAPADES of an ADVENTUROUS BACHELOR
DR. JERRY LOVE

The Romantic Escapades ...
Author: Jerry Love



FairyRealm: Enter the Realm
THREE ADVENTURES
the charm bracelet
the flower fairies
the third wish
EMILY RODDA

Fairy Realm: Enter the Rea...
Author: Emily Rodda

Add a Comment

Comment

Submit



FairyRealm: Enter the Realm
THREE ADVENTURES
the charm bracelet
the flower fairies
the third wish
EMILY RODDA

Fairy Realm: Enter the Realm: Three Adventures

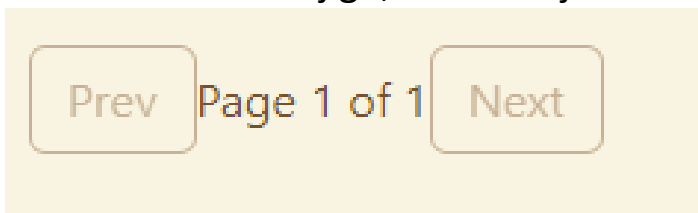
Emily Rodda

Follow Jessie on her adventures in these three sparkling, magical tales. The Charm Bracelet: When Jessie visits her grandmother at beautiful Blue Moon, she discovers an amazing secret and enters the Realm for the first time. The Flower Fairies: Jessie must protect the flower fairies when they follow her out of the Realm and into her own world! The Third Wish: When forest fires threaten Jessie's home, only the magic of the Realm can help her!

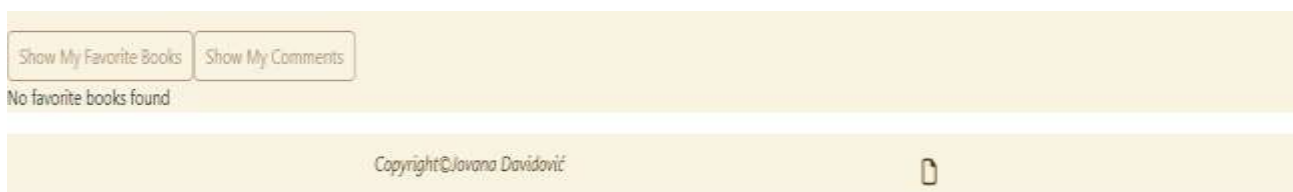
Comments

pera (2024-06-28T20:45:08.386Z):
lepo

Pretraga,filtriranje, sortiranje i straničenje - pretraga se vrši na osnovu naziva knjige, sortiranje na osnovu izabrane opcije, filtriranje se vrši na osnovu žanra za knjigu, straničenje na osnovu broja knjiga u mockAPI.



Sve knjige jednog korisnika- ukoliko nema knjiga dobija se poruka: „No



favorite books found“

Svi komentari jednog korisnika- ukoliko nema komentara dobija se poruka: „No comment found“

Forma za logovanje – korisnik unosi svoje korisničko ime i lozinku, proverava se da li je tačno uneo podatke i ako jeste proverava se da li je upitanje administrator ili obični korisnik. U zavisnosti ko se ulogovao biće drugačije redirektovan.

Forma za registraciju – korisnik unosi svoj username, mejl, lozinku i confirm lozinku.

13.06..2024. Jovana Davidović 111/22

Username ne sme biti prazan.

Username mora biti jedinstven tj. ne sme biti već unet.

Mejl mora da sadrzi @ u sebi.

Lozinka mora imati više od 6 karaktera.

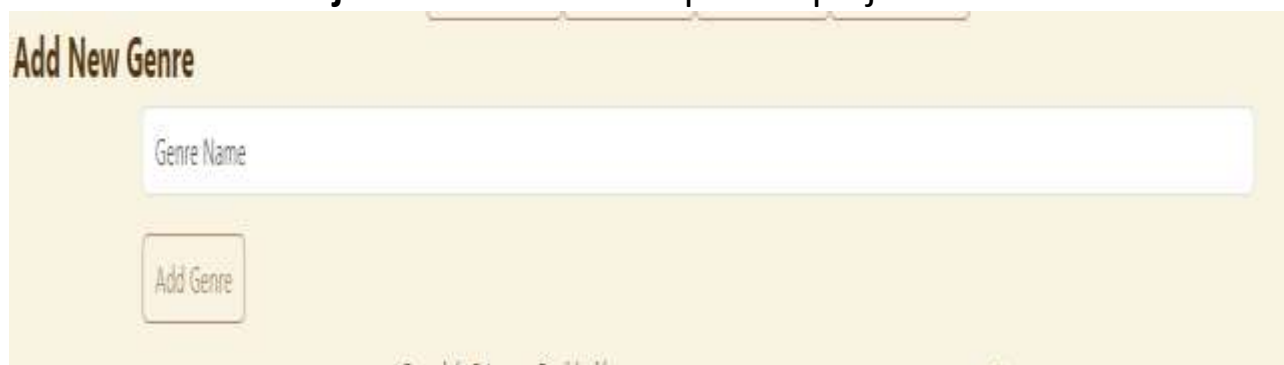
Confirm lozinka mora biti ista kao lozinka.

A sign-up form with a light yellow background. At the top left, the text "Sign up" is written in a bold, dark brown font. Below this, there are four input fields, each with a label to its left: "Username", "Email", "Password", and "Confirm password". The labels are in a small, dark brown font. The input fields are white with a thin grey border. At the bottom left of the form, there is a button with the text "Sign up" in a dark brown font, enclosed in a rounded rectangle with a thin brown border.

Tabela sa korisnicima (i onim koji su admin). Pored podataka iz baze nalazi se button za brisanje korisnika.

Username	Email	Role	Actions
jovana06	jovanadavidovic2003@gmail.com	User	<button>Delete</button>
jovana10	jovanadavidovic2003@gmail.com	User	<button>Delete</button>
mika	mika1@gmail.com	Admin	<button>Delete</button>
pera	pera@gmail.com	User	<button>Delete</button>
jovanaa	jovanadavidovic2003@gmail.com	User	<button>Delete</button>
jovanadavidovic11122	jovana1@gmail.com	User	<button>Delete</button>

Forma za dodavanje žanra-ne sme biti prazno polje.



The screenshot shows a web form titled "Add New Genre". It contains a single text input field with the placeholder text "Genre Name". Below the input field is a button labeled "Add Genre". The form is set against a light yellow background.

Forma za dodavanje knjige-polja ne smeju biti prazna, za sliku se proverava da li je ispravan URL i da mora da ima bar jedan žanr.

Book Name
Description
Image URL
Published Year
Author
Genres: <input type="checkbox"/> Action <input type="checkbox"/> Adventure <input type="checkbox"/> Drama <input type="checkbox"/> Comedy <input type="checkbox"/> Fantasy <input type="checkbox"/> Tragedy <input type="checkbox"/> Romance <input type="checkbox"/> Mystery

Tabela sa knjigama. Pored naziva knjige nalazi se button za brisanje knjige.

Name	Actions
The Great Adventure: Al-Fayed's Rollercoaster Ride with Fulham FC	Delete
The Romantic Escapades of an adventurous Bachelor	Delete
Fairy Realm: Enter the Realm: Three Adventures	Delete
Laugh Out Loud	Delete
It Starts with Us: A Novel	Delete
Romeo And Juliet	Delete

Kodovi:

App.vue

```
<template>
  <the-header></the-header>
  <cover-img/>
  <router-view v-slot="{ Component, route }">
    <transition :name="route.meta.transition">
      <component :is="Component" />
    </transition>
  </router-view>
  <the-footer></the-footer>
</template>
<script>
import TheHeader from './components/fix/TheHeader.vue';
import CoverImg from './components/fix/CoverImg.vue';
import TheFooter from './components/fix/TheFooter.vue';
export default {
  components: {
    TheHeader, CoverImg, TheFooter
  },
  created() {
    this.$store.dispatch("books/loadBooks");
    this.$store.dispatch("books/getCategories");
  },
}
</script>
<style scoped>
* {
  margin: 0px;
  padding: 0px;
  background-color: #F8F4E1;
  box-sizing: border-box;
}
.slide-enter-active,
.slide-leave-active {
  transition: all 0.75s ease-out;
}

.slide-enter-to {
  position: absolute;
  right: 0;
}

.slide-enter-from {
```

```
position: absolute;  
right: -100%;  
}
```

```
.slide-leave-to {  
position: absolute;  
left: -100%;  
}
```

```
.slide-leave-from {  
position: absolute;  
left: 0;  
}
```

```
.fade-enter-active,  
.fade-leave-active {  
transition: opacity 0.5s ease;  
}
```

```
.fade-enter-from,  
.fade-leave-to {  
opacity: 0;  
}
```

```
.route-enter-from {  
opacity: 0;  
transform: translateY(-30px);  
}
```

```
.route-leave-to {  
opacity: 0;  
transform: translateY(30px);  
}
```

```
.route-enter-active {  
transition: all 0.3s ease-out;  
}
```

```
.route-leave-active {  
transition: all 0.3s ease-in;  
}
```

```
.route-enter-to,  
.route-leave-from {  
opacity: 1;  
transform: translateY(0);  
}  
.slide-fade-enter-active {
```

```
    transition: all 0.3s ease-out;
  }

.slide-fade-leave-active {
  transition: all 0.8s cubic-bezier(1, 0.5, 0.8, 1);
}

.slide-fade-enter-from,
.slide-fade-leave-to {
  transform: translateX(20px);
  opacity: 0;
}
</style>
```

Components

Fix

TheHeader.vue

```
<template>
  <header>
    <router-link to="/"><h1>Book Tracker</h1></router-link>
    <the-nav></the-nav>
  </header>
</template>
<script>
import TheNav from "./TheNav.vue";
export default {
  components: {
    TheNav
  }
}
</script>
<style scoped>
header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
a {
  text-decoration: none;
  margin-left: 10px;
}
```

```
h1 {  
  font-style: italic;  
  font-weight: bold;  
  color: #543310;  
}  
</style>
```

TheFooter.vue

```
<template>  
  <footer>  
    <p>Copyright&#169;Jovana Davidović</p>  
    <a href="../../assets/dokumentacija"><i class="fa-regular fa-file"></i></a>  
  </footer>  
</template>  
<style scoped>  
  footer {  
    padding-top: 10px;  
    display: flex;  
    justify-content: space-evenly;  
    align-items: center;  
    color: #543310;  
  }  
  a {  
    text-decoration: none;  
    color: #543310;  
  }  
  i {  
    font-size: 20px;  
  }  
  p {  
    font-style: italic;  
    margin-left: 10px;  
  }  
</style>
```

TheNav.vue

```
<template>  
  <nav class="navbar navbar-expand-lg bg-body-tertiary">  
    <div class="container-fluid">  
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-  
target="#navbarTogglerDemo01" aria-controls="navbarTogglerDemo01" aria-expanded="false" aria-  
label="Toggle navigation">  
        <span class="navbar-toggler-icon"></span>  
      </button>
```



```

<div class="collapse navbar-collapse" id="navbarTogglerDemo01">
  <ul class="navbar-nav me-auto mb-2 mb-lg-0">
    <li class="nav-item"><router-link class="nav-link" to="/">List of books</router-link></li>
    <li class="nav-item" v-if="!isLoggin">
      <router-link class="nav-link" to="/signup">Signup</router-link>
    </li>
    <li class="nav-item" v-if="!isLoggin">
      <router-link class="nav-link" to="/login">Login</router-link>
    </li>
    <li class="nav-item" v-if="isLoggin" @click="logout">
      <router-link class="nav-link" to="/">Logout</router-link>
    </li>
    <li class="nav-item"><router-link class="nav-link" to="/about">Author</router-link></li>
    <li class="nav-item" v-if="isAdmin">
      <router-link class="nav-link" to="/admin">Admin</router-link>
    </li>
    <li class="nav-item" v-if="isUser">
      <router-link class="nav-link" to="/myprofile">User</router-link>
    </li>
  </ul>
</div>
</div>
</nav>
</template>

```

```

<script>
import { mapGetters } from "vuex";

export default {
  computed: {
    ...mapGetters('auth', ['isAuthenticated', 'isAdmin', 'isUser']),
    isLoggin() {
      return this.isAuthenticated;
    },
  },
  methods: {
    logout() {
      this.$store.dispatch("auth/logout").then(() => {
        this.$router.replace({ name: "login" });
      });
    },
  },
};
</script>

```

```

<style scoped>
nav {

```

```
background-color: #F8F4E1 !important;
}
a {
color: #74512D;
}
</style>
```

CoverImg.vue

```
<template>
  <div></div>
</template>
<style scoped>
div{
width: auto;
height: 200px;
background-image: url("../assets/img/books.jpg");
background-repeat: no-repeat;
background-size: cover;
background-position:center bottom;
}
</style>
```

Books

BookForm.vue

```
<template>
  <form @submit.prevent="handleSubmit">
    <div class="mb-3">
      <input v-model="book.name" placeholder="Book Name" class="form-control" required />
    </div>
    <div class="mb-3">
      <input v-model="book.description" placeholder="Description" class="form-control" required />
    </div>
    <div class="mb-3">
      <input v-model="book.img" placeholder="Image URL" class="form-control" required
@input="validateURL" />
      <span v-if="errors.img" class="error">{{ errors.img }}</span>
    </div>
    <div class="mb-3">
      <input v-model="book.published_year" type="number" class="form-control" placeholder="Published Year"
required />
    </div>
```

```

<div class="mb-3">
  <input v-model="book.author" placeholder="Author" class="form-control" required />
</div>
<div class="mb-3 form-check" v-if="genres.length">
  <label>Genres:</label>
  <div v-for="genre in genres" :key="genre.id">
    <input type="checkbox" class="form-check-input" :value="genre.id" @change="toggleGenre(genre.id)">
      {{ genre.name }}
    </div>
  </div>
  <button type="submit" :disabled="isSubmitDisabled" class="btn">Add Book</button>
</form>
</template>

```

```

<script>
import { mapGetters } from 'vuex';

```

```

export default {
  data() {
    return {
      book: {
        name: "",
        description: "",
        img: "",
        published_year: "",
        author: "",
        genreId: []
      },
      errors: {
        img: ""
      },
      isSubmitDisabled: true
    };
  },
  computed: {
    ...mapGetters('books', ['getGenres']),
    genres() {
      return this.getGenres;
    }
  },
  methods: {
    handleSubmit() {
      this.$emit('submit', this.book);
      this.resetForm();
    },
    validateURL() {

```

```

const urlPattern = new RegExp('^((https?:\\|\\|)?' + // validate protocol
'(((\\[a-z\\d](\\[a-z\\d-]*[a-z\\d])*)\\|\\|)+[a-z]{2,})' + // validate domain name
'((\\d{1,3}\\|\\|){3}\\|\\d{1,3}))' + // OR validate ip (v4) address
'(\\|:\\d+)?(\\|\\|[-a-z\\d%_.~+]*)*' + // validate port and path
'\\|?[:&a-z\\d%_.~+=-]*)?' + // validate query string
'\\|\\#[-a-z\\d_]*)?$', 'i'); // validate fragment locator
if (!urlPattern.test(this.book.img)) {
  this.errors.img = 'Invalid URL';
  this.isSubmitDisabled = true;
} else {
  this.errors.img = "";
  this.isSubmitDisabled = false;
}
},
toggleGenre(genreId) {
  const index = this.book.genreId.indexOf(genreId);
  if (index === -1) {
    this.book.genreId.push(genreId);
  } else {
    this.book.genreId.splice(index, 1);
  }
},
resetForm() {
  this.book = {
    name: "",
    description: "",
    img: "",
    published_year: "",
    author: "",
    genreId: []
  };
  this.isSubmitDisabled = true;
  this.errors.img = "";
},
created() {
  this.$store.dispatch('books/getCategories');
}
};
</script>

<style>
form{
  margin:auto;
  width:80%;
}
.error {

```

```
color: red;
font-size: 0.9em;
}
.btn{
border-color: #543310;
color:#543310;
}
</style>
```

FilterBooks.vue

```
<template>
  <div id="f">
    <div id="fil">
      <div class="form-check" v-for="g of genres" :key="g.id">
        <input
          class="form-check-input"
          type="checkbox"
          :value="g.id"
          :id="g.id+g.name"
          v-model="selectedGenres"
        />
        <label class="form-check-label" :for="g.id+g.name">
          {{ g.name }}
        </label>
      </div>
    </div>
    <select class="form-select" v-model="sortOption">
      <option value="0">Sort by</option>
      <option value="name_asc">Books name Asc</option>
      <option value="name_desc">Books name Desc</option>
      <option value="year_asc">Publish year Asc</option>
      <option value="year_desc">Publish year Desc</option>
    </select>
    <input
      class="form-control search"
      type="search"
      placeholder="Search"
      v-model="searchQuery"
    />
  </div>
</template>

<script>
import { mapGetters, mapMutations } from 'vuex';
```

```

export default {
  data() {
    return {
      genres: [],
      selectedGenres: [],
      sortOption: '0',
      searchQuery: ""
    };
  },
  computed: {
    ...mapGetters({
      genresFromStore: 'books/getGenres'
    })
  },
  watch: {
    genresFromStore: {
      immediate: true,
      handler(newGenres) {
        if (newGenres && Array.isArray(newGenres)) {
          this.genres = newGenres.sort((a, b) => {
            if (a.name < b.name) return -1;
            if (a.name > b.name) return 1;
            return 0;
          });
        } else {
          this.genres = [];
        }
      }
    },
    selectedGenres(newGenres) {
      this.setGenres(newGenres);
    },
    sortOption(newSortOption) {
      this.setSortOption(newSortOption);
    },
    searchQuery(newSearchQuery) {
      this.setSearch(newSearchQuery);
    }
  },
  methods: {
    ...mapMutations('books', ['setGenres', 'setSearch', 'setSortOption'])
  },
  created() {
    this.$store.dispatch('books/getCategories');
  }
};
</script>

```

```
<style scoped>
#f {
  margin-left: 20px;
  display: flex;
  flex-direction: column-reverse;
  align-items: flex-start;
  gap: 15px;
  margin-right: 20px;
  color: #74512d;
}
#fil {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
}
.form-select {
  width: auto !important;
}
.search {
  width: auto !important;
}
</style>
```

TheFormComment.vue

```
<template>
  <div class="comment-form">
    <h3>Add a Comment</h3>
    <form @submit.prevent="submitComment">
      <custom-input
        v-model="commentText"
        label="Comment"
        id="comment"
      ></custom-input>
      <button type="submit" class="btn btn-outline-light">Submit</button>
    </form>
  </div>
</template>

<script>
import CustomInput from '../ui/CustomInput.vue';

export default {
  components: {
    CustomInput,
```

```

},
props: ['bookId'],
data() {
  return {
    commentText: "",
  };
},
methods: {
  async submitComment() {
    if (this.commentText.trim() === "") {
      alert('Comment cannot be empty');
      return;
    }
  }
}

```

```

const commentData = {
  bookId: this.bookId,
  comment: this.commentText,
  user: this.$store.getters['auth/currentUser'].username,
  date: new Date().toISOString(),
};

```

```

  try {
    await this.$store.dispatch('books/addComment', commentData);
    alert('Comment added successfully');
    this.$emit('commentAdded', commentData); // Emit event to parent
    this.commentText = ""; // Reset the form
  } catch (error) {
    console.error('Error submitting comment:', error);
    alert('Error submitting comment');
  }
},
},
};
</script>
<style scoped>
h3{
  font-style: italic;
  color:#543310;
  margin-left:10px;
}
form{
  color:#543310;
}
.btn-outline-light{
  border-color: #AF8F6F;
  color:#AF8F6F;
}

```


</style>

Ui

BaseButton.vue

```
<template>
  <router-link v-if="isLink" :to="path">{{ text }}</router-link>
  <button v-else>
    <slot></slot>
  </button>
</template>
<script>
export default {
  props: {
    path: {
      type: String,
      required: false,
      default: "/",
    },
    text: {
      type: String,
      required: true,
    },
    isLink: {
      type: Boolean,
      required: false,
      default: true,
    },
  },
};
</script>
<style scoped>
a {
  border: 1px solid #543310;
  text-decoration: none;
  display: inline-block;
  margin: 10px 5px;
  border-radius: 10px;
  padding: 5px 15px;
  color: #543310;
  background-color: rgba(175, 143, 111,0.5);
}
</style>
```

CommentItem.vue

```
<template>
  <div class="comment-item">
    <p><strong>{{ comment.user }}</strong> ({{ comment.date }}):</p>
    <p>{{ comment.comment }}</p>
  </div>
</template>

<script>
export default {
  name: 'CommentItem',
  props: {
    comment: {
      type: Object,
      required: true,
    },
  },
};
</script>

<style scoped>
.comment-item {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
.comment-item p {
  margin: 0;
}
.comment-item strong {
  font-weight: bold;
}
</style>
```

CustomInput.vue

```
<template>
  <div class="custom-input">
    <label :for="id">{{ label }}</label>
    <input
```

```
      :id="id"
      :type="type"
      :value="modelValue"
      @input="$emit('update:modelValue', $event.target.value)"
    />
    <p v-if="error">{{ error }}</p>
  </div>
</template>
```

```
<script>
export default {
  name: 'CustomInput',
  props: {
    modelValue: {
      type: String,
      default: ""
    },
    label: {
      type: String,
      required: true
    },
    id: {
      type: String,
      required: true
    },
    type: {
      type: String,
      default: 'text'
    },
    error: {
      type: String,
      default: ""
    }
  },
};
</script>
```

```
<style scoped>
.custom-input {
  margin-bottom: 1rem;
}
.custom-input label {
  display: block;
  margin-bottom: 0.5rem;
}
.custom-input input {
  width: 100%;
```

```
padding: 0.5rem;
border: 1px solid #ddd;
border-radius: 5px;
}
.custom-input p {
color: red;
margin-top: 0.5rem;
}
</style>
```

Views

AboutView.vue

```
<template>
<div id="author">
  <h3>Author</h3>
  <div class="card ms-5" style="max-width: 85%;">
    <div class="row g-0">
      <div class="col-md-3">
        
      </div>
      <div class="col-md-9">
        <div class="card-body">
          <h5 class="card-title">Jovana Davidović</h5>
          <p class="card-text"><small class="text-muted">111/22</small></p>
        </div>
      </div>
    </div>
  </div>
</div>
</div>
</div>
</div>
</div>
</template>
<style scoped>
h3{
font-style: italic;
color:#543310;
margin-left:10px;
}
.card{
background-color: transparent !important;
color:#74512D !important;
border: 0px;
}
.card-body {
text-align: justify;
```

```
}
```

```
.img-fluid {  
  object-fit: cover;  
  height: 100%;  
}  
</style>
```

AdminView.vue

```
<template>  
  <div id="admin">  
    <h3>Admin Page</h3>  
    <div class="dugmad">  
      <button @click="toggleSection('books')" class="btn btn-outline-light">Show All Books</button>  
      <button @click="toggleSection('users')" class="btn btn-outline-light">Show All Users</button>  
      <button @click="toggleSection('addBook')" class="btn btn-outline-light">Add New Book</button>  
      <button @click="toggleSection('addGenre')" class="btn btn-outline-light">Add New Genre</button>  
    </div>  
    <div v-if="showBooks">  
      <h4>All Books</h4>  
      <table>  
        <thead>  
          <tr>  
            <th>Name</th>  
            <th>Actions</th>  
          </tr>  
        </thead>  
        <tbody>  
          <tr v-for="book in books" :key="book.id">  
            <td>{{ book.name }}</td>  
            <td>  
              <button @click="deleteBook(book.id)" class="btn btn-outline-light">Delete</button>  
            </td>  
          </tr>  
        </tbody>  
      </table>  
    </div>  
  
    <div v-if="showUsers">  
      <h4>All Users</h4>  
      <table>  
        <thead>  
          <tr>
```

```

    <th>Username</th>
    <th>Email</th>
    <th>Role</th>
    <th>Actions</th>
  </tr>
</thead>
<tbody>
  <tr v-for="user in users" :key="user.id">
    <td>{{ user.username }}</td>
    <td>{{ user.email }}</td>
    <td>{{ user.role }}</td>
    <td>
      <button @click="deleteUser(user.id)" class="btn btn-outline-light">Delete</button>
    </td>
  </tr>
</tbody>
</table>
</div>

```

```

<div v-if="showAddBook">
  <h4>Add New Book</h4>
  <book-form @submit="addBook"></book-form>
</div>

```

```

<div v-if="showAddGenre">
  <h4>Add New Genre</h4>
  <form @submit.prevent="addGenre">
    <div class="mb-3">
      <input v-model="newGenre.name" class="form-control" placeholder="Genre Name" required />
    </div>
    <button type="submit" class="btn btn-outline-light">Add Genre</button>
  </form>
</div>
</div>
</template>

```

```

<script>
import { mapGetters } from 'vuex';
import BookForm from './components/books/BookForm.vue';

```

```

export default {
  components: {
    BookForm
  },
  data() {
    return {
      showBooks: true,

```

```

    showUsers: false,
    showAddBook: false,
    showAddGenre: false,
    newGenre: {
      name: ""
    },
  },
  computed: {
    ...mapGetters({
      books: 'books/getBooks',
      users: 'auth/users'
    })
  },
  methods: {
    toggleSection(section) {
      this.showBooks = section === 'books';
      this.showUsers = section === 'users';
      this.showAddBook = section === 'addBook';
      this.showAddGenre = section === 'addGenre';
    },
    async addBook(book) {
      await this.$store.dispatch('books/addBook', book);
    },
    async addGenre() {
      await this.$store.dispatch('books/addGenre', this.newGenre);
      this.newGenre = { name: "" };
    },
    async deleteBook(bookId) {
      await this.$store.dispatch('books/deleteBook', bookId);
    },
    async deleteUser(userId) {
      await this.$store.dispatch('auth/deleteUser', userId);
    },
  },
  created() {
    this.$store.dispatch('books/loadBooks');
    this.$store.dispatch('books/getCategories');
    this.$store.dispatch('auth/fetchUsers');
  }
};
</script>

<style scoped>
#admin{
  padding-top:25px;
  background-color: #F8F4E1;

```

```
}
h3{
  font-style: italic;
  color:#543310;
  margin-left:10px;
}
h4{
  font-weight: bold;
  color:#543310;
  margin-left:10px;
}
.dugmad{
  display: flex;
  align-items: center;
  justify-content: center;
  flex-wrap: wrap;
}
table {
  width: auto;
  border-collapse: collapse;
  margin:auto;
}
th, td {
  padding: 8px 12px;
  border: 1px solid rgba(175, 143, 111,0.5);
}
th {
  background-color: rgba(175, 143, 111,0.5);
}
.btn-outline-light{
  border-color: #AF8F6F;
  color:#AF8F6F;
}
</style>
```

HomeView.vue

```
<template>
  <main>
    <h2>Books</h2>
    <div id="list">
      <filter-books @genres-loaded="setGenres"></filter-books>
      <div id="l">
```



```

<div class="card mb-3 me-3" style="width: 18rem;" v-for="book in paginatedData" :key="book.id">
  <a :href="'/book/'+book.id">
    
    <div class="card-body">
      <h5 class="card-title">{{ book.name }}</h5>
      <p class="card-text">Author: {{ book.author }}</p>
      <p class="card-text">
        Genre: {{ getGenreNames(book.genreId).join(', ') }}
      </p>
      <p class="card-text">Published: {{ book.published_year }}</p>
    </div>
  </a>
  <button v-if="isUser" @click="addToFavorites(book.id)" class="btn btn-outline-light">
    <i class="fa fa-heart"></i> Add to Your Book List
  </button>
</div>
</div>
</div>
<div class="pagination">
  <button @click="prevPage" :disabled="currentPage === 1" class="btn btn-outline-light">Prev</button>
  <span>Page {{ currentPage }} of {{ totalPages }}</span>
  <button @click="nextPage" :disabled="currentPage === totalPages" class="btn btn-outline-
light">Next</button>
</div>
</main>
</template>

<script>
import FilterBooks from "../components/books/FilterBooks.vue";
import { mapGetters, mapActions } from 'vuex';

export default {
  data() {
    return {
      genres: [],
      currentPage: 1,
      itemsPerPage: 6
    };
  },
  components: {
    FilterBooks
  },
  computed: {
    ...mapGetters({
      isUser: 'auth/isUser',
      currentUser: 'auth/currentUser',
      books: 'books/getBooks',

```

```

    genresFromStore: 'books/getGenres'
  )),
  totalPages() {
    return this.books.length ? Math.ceil(this.books.length / this.itemsPerPage) : 1;
  },
  paginatedData() {
    const start = (this.currentPage - 1) * this.itemsPerPage;
    const end = start + this.itemsPerPage;
    return this.books.slice(start, end);
  }
},
watch: {
  genresFromStore: {
    immediate: true,
    handler(newGenres) {
      this.genres = newGenres;
    }
  }
},
methods: {
  ...mapActions([
    addToFavorites: 'books/addToFavorites'
  ]),
  setGenres(genres) {
    this.genres = genres;
  },
  getGenreNames(genreIds) {
    return genreIds.map(genreId => {
      const genre = this.genres.find(genre => genre.id === genreId);
      return genre ? genre.name : '';
    });
  },
  prevPage() {
    if (this.currentPage > 1) {
      this.currentPage--;
    }
  },
  nextPage() {
    if (this.currentPage < this.totalPages) {
      this.currentPage++;
    }
  }
}
];
</script>

```

```
<style scoped>
```

```
main{
  overflow-x: hidden;
}
h2{
  font-style: italic;
  font-weight: bold;
  color:#543310;
  margin-left:10px;
}
a{
  text-decoration: none;
  color:#74512D !important;
}
#list{
  display: flex;
  align-items: flex-start;
}
#lista #f{
  position: -webkit-sticky;
  position: sticky;
  top: 0;
}
#1{
  display: flex;
  flex-wrap: wrap;
}
.card{
  background-color: rgba(175, 143, 111,0.5) !important;
}
.card-img-top{
  height: 401.67px;
}
.card-title {
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
  width: 100%;
}
.pagination{
  margin-left:20px;
  display: flex;
  align-items: center;
  color:#74512D;
}
.btn-outline-light{
  border-color: #AF8F6F;
  color:#AF8F6F;
```

```

}
</style>

```

LoginView.vue

```

<template>
  <div class="">
    <h2>Login</h2>
    <form @submit.prevent="submitForm">
      <div class="mb-3">
        <label for="username" class="form-label">Username</label>
        <input
          type="text"
          class="form-control"
          id="username"
          v-model="form.username"
          :class="{ 'is-invalid': errors.username }"
        />
        <div v-if="errors.username" class="invalid-feedback">{{ errors.username }}</div>
      </div>
      <div class="mb-3">
        <label for="password" class="form-label">Password</label>
        <input
          type="password"
          class="form-control"
          id="password"
          v-model="form.password"
          :class="{ 'is-invalid': errors.password }"
        />
        <div v-if="errors.password" class="invalid-feedback">{{ errors.password }}</div>
      </div>
      <button type="submit" class="btn btn-outline-light">Login</button>
    </form>
    <div v-if="loginMessage" class="alert mt-3" :class="{ 'alert-success': loginSuccess, 'alert-
danger': !loginSuccess }">
      {{ loginMessage }}
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {

```

```

    form: {
      username: "",
      password: ""
    },
    errors: {},
    loginMessage: "",
    loginSuccess: false
  };
},
methods: {
  validateForm() {
    this.errors = {};

    if (!this.form.username) {
      this.errors.username = 'Korisničko ime je obavezno.';
    }

    if (!this.form.password) {
      this.errors.password = 'Lozinka je obavezna.';
    }

    return Object.keys(this.errors).length === 0;
  },
  async submitForm() {
    if (this.validateForm()) {
      try {
        const response = await this.$store.dispatch("auth/login", {
          username: this.form.username,
          password: this.form.password
        });
        this.loginMessage = response.message;
        this.loginSuccess = response.success;
        if (response.success) {
          this.$router.push({ name: 'myprofile' });
        }
      } catch (error) {
        console.error(error);
        this.loginMessage = 'Došlo je do greške prilikom logovanja.';
        this.loginSuccess = false;
      }
    }
  }
};
</script>

<style scoped>

```

```
h2{
  font-style: italic;
  font-weight: bold;
  color:#543310;
  margin-left:10px;
}

form{
  color:#74512D;
}

.is-invalid {
  border-color: #dc3545;
}

.invalid-feedback {
  display: block;
  color: #dc3545;
}

.alert {
  color: #721c24;
  background-color: #f8d7da;
  border-color: #f5c6cb;
}

.btn-outline-light{
  border-color: #AF8F6F;
  color:#AF8F6F;
}

</style>
```

MyProfileView.vue

```
<template>
  <div id="profile">
    <h3>My Profile</h3>
    <p><strong>Username:</strong> {{ currentUser.username }}</p>
    <p><strong>Email:</strong> {{ currentUser.email }}</p>

    <button @click="toggleSection('favorites')" class="btn btn-outline-light">Show My Favorite Books</button>
    <button @click="toggleSection('comments')" class="btn btn-outline-light">Show My Comments</button>

    <div v-if="showComments">
```

```

<div v-if="comments && comments.length > 0">
  <h3>Comments</h3>
  <ul>
    <li v-for="comment in comments" :key="comment.date">
      <comment-item :comment="comment"></comment-item>
    </li>
  </ul>
</div>
<div v-else-if="commentsFetched">
  <p>No comments found</p>
</div>
</div>
<div v-if="showFavorites">
  <div v-if="favorites && favorites.length > 0">
    <h4>My Favorite Books</h4>
    <div id="l">
      <div class="card mb-3 me-3" style="width: 18rem;" v-for="book in favorites" :key="book.id">
        <a :href="'/book/'+book.id">
          
          <div class="card-body">
            <h5 class="card-title">{{ book.name }}</h5>
            <p class="card-text">Author: {{ book.author }}</p>
            <p class="card-text">Published: {{ book.published_year }}</p>
          </div>
          </a>
          <button @click="removeFromFavorites(book.id)" class="btn btn-outline-light">
            Remove from Your Book List
          </button>
        </div>
      </div>
    </div>
    <div v-else-if="favoritesFetched">
      <p>No favorite books found</p>
    </div>
  </div>
</template>

<script>
import { mapGetters } from 'vuex';
import CommentItem from '../components/ui/CommentItem.vue';

export default {
  components: {
    CommentItem,
  },
  data() {

```

```

    return {
      comments: [],
      favorites: [],
      commentsFetched: false,
      favoritesFetched: false,
      showComments: false,
      showFavorites: true,
    };
  },
  computed: {
    ...mapGetters('auth', ['currentUser']),
    ...mapGetters('books', ['getBooks']),
  },
  methods: {
    async fetchUserComments() {
      try {
        const response = await this.$store.dispatch('books/getUserComments', this.currentUser.username);
        this.comments = response;
        console.log(this.comments);
        this.commentsFetched = true;
      } catch (error) {
        console.error('Error fetching comments:', error);
        this.commentsFetched = true;
      }
    },
    async fetchFavoriteBooks() {
      try {
        const favoriteBookIds = await this.$store.dispatch('books/getUserFavorites', this.currentUser.id);

        this.favorites = this.getBooks.filter(book => favoriteBookIds.includes(book.id.toString()));
        this.favoritesFetched = true;
      } catch (error) {
        console.error('Error fetching favorite books:', error);
        this.favoritesFetched = true;
      }
    },
    async removeFromFavorites(bookId) {
      try {
        await this.$store.dispatch('books/removeBookFromFavorites', {
          userId: this.currentUser.id,
          bookId
        });
        this.favorites = this.favorites.filter(book => book.id !== bookId);
      } catch (error) {
        console.error('Error removing book from favorites:', error);
      }
    },
  },

```



```

toggleSection(section) {
  if (section === 'comments') {
    this.showComments = !this.showComments;
    this.showFavorites = false;
    if (this.showComments && !this.commentsFetched) {
      this.fetchUserComments();
    }
  } else if (section === 'favorites') {
    this.showFavorites = !this.showFavorites;
    this.showComments = false;
    if (this.showFavorites && !this.favoritesFetched) {
      this.fetchFavoriteBooks();
    }
  }
},
},
};
</script>
<style scoped>
ul li{
  list-style-type:none;
}
a{
  text-decoration: none;
  color:#74512D !important;
}
#l{
  display: flex;
  flex-wrap: wrap;
  justify-content: center
}
.card{
  background-color: rgba(175, 143, 111,0.5) !important;
}
.card-img-top{
  height: 401.67px;
}
.card-title {
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
  width: 100%;
}
.btn-outline-light{
  border-color: #AF8F6F;
  color:#AF8F6F;
}

```

</style>

SingleBookView.vue

```
<template>
  <div id="single">
    <div v-if="typeof singleBook === 'object' && singleBook !== null">
      <h3>Book Details</h3>
      <div class="card mb-3 ms-5" style="max-width: 85%;">
        <div class="row g-0">
          <div class="col-md-3">
            
          </div>
          <div class="col-md-9">
            <div class="card-body">
              <h5 class="card-title">{{ singleBook.name }}</h5>
              <p class="card-text"><small class="text-muted">{{ singleBook.author }}</small></p>
              <p class="card-text">{{ singleBook.description }}</p>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div v-if="isUser && isAuthenticated">
      <h2>Leave a comment</h2>
      <the-form-comment :bookId="singleBook.id" @commentAdded="addComment"></the-form-comment>
    </div>
    <div v-if="comments && comments.length">
      <h3>Comments</h3>
      <ul>
        <li v-for="comment in comments" :key="comment.date">
          <comment-item :comment="comment"></comment-item>
        </li>
      </ul>
    </div>
    <div v-else>
      <h3>{{ singleBook }}</h3>
    </div>
  </div>
</template>
```

<script>

import { mapGetters } from 'vuex';

import TheFormComment from '../components/books/TheFormComment.vue';

```

import CommentItem from '../components/ui/CommentItem.vue';

export default {
  components: {
    TheFormComment,
    CommentItem,
  },
  props: ['id'],
  data() {
    return {
      singleBook: null,
      comments: [],
    };
  },
  computed: {
    ...mapGetters('auth', ['isAuthenticated', 'isUser']),
  },
  async created() {
    await this.$store.dispatch('books/getOneBook', this.id);
    this.getOneBook();
    this.getComments();
  },
  methods: {
    getOneBook() {
      try {
        this.singleBook = this.$store.getters['books/getBook'](this.id);
        console.log(this.singleBook);
      } catch (err) {
        console.error("Error getting the book from store:", err);
      }
    },
    async getComments() {
      try {
        this.comments = await this.$store.dispatch('books/getCommentsByBookId', this.id);
        console.log(this.comments.data);
      } catch (err) {
        console.error("Error getting comments:", err);
      }
    },
    addComment(comment) {
      this.comments.push(comment);
    },
  },
  watch: {
    "$store.state.books.books"() {
      this.getOneBook();
    },
  },
};

```

```
},
};
</script>
<style scoped>
h2{
  font-style: italic;
  font-weight: bold;
  color:#543310;
  margin-left:10px;
}
ul li{
  list-style-type:none;
}
h3{
  font-style: italic;
  color:#543310;
  margin-left:10px;
}
.card{
  background-color: transparent !important;
  color:#74512D !important;
  border: 0px;
}
.card-body {
  text-align: justify;
}

.img-fluid {
  object-fit: cover;
  height: 100%;
}
</style>
```

SingUpView.vue

```
<template>
<div class="">
  <h2>Sign up</h2>
  <form @submit.prevent="submitForm">
    <div class="mb-3">
      <label for="username" class="form-label">Username</label>
      <input
        type="text"
        class="form-control"
```

```

        id="username"
        v-model="form.username"
        :class="{ 'is-invalid': errors.username }"
    />
    <div v-if="errors.username" class="invalid-feedback">{{ errors.username }}</div>
</div>
<div class="mb-3">
    <label for="email" class="form-label">Email</label>
    <input
        type="email"
        class="form-control"
        id="email"
        v-model="form.email"
        :class="{ 'is-invalid': errors.email }"
    />
    <div v-if="errors.email" class="invalid-feedback">{{ errors.email }}</div>
</div>
<div class="mb-3">
    <label for="password" class="form-label">Password</label>
    <input
        type="password"
        class="form-control"
        id="password"
        v-model="form.password"
        :class="{ 'is-invalid': errors.password }"
    />
    <div v-if="errors.password" class="invalid-feedback">{{ errors.password }}</div>
</div>
<div class="mb-3">
    <label for="confirmPassword" class="form-label">Confirm password</label>
    <input
        type="password"
        class="form-control"
        id="confirmPassword"
        v-model="form.confirmPassword"
        :class="{ 'is-invalid': errors.confirmPassword }"
    />
    <div v-if="errors.confirmPassword" class="invalid-feedback">{{ errors.confirmPassword }}</div>
</div>
    <button type="submit" class="btn btn-outline-light">Sign up</button>
</form>
    <div v-if="registerMessage" class="alert mt-3" :class="{ 'alert-success': registerSuccess, 'alert-
danger': !registerSuccess }">
        {{ registerMessage }}
    </div>
</div>
</template>

```

```
<script>
export default {
  data() {
    return {
      form: {
        username: "",
        email: "",
        password: "",
        confirmPassword: ""
      },
      errors: {},
      registerMessage: "",
      registerSuccess: false
    };
  },
  methods: {
    validateForm() {
      this.errors = {};

      if (!this.form.username) {
        this.errors.username = 'Korisničko ime je obavezno.';
      }

      if (!this.form.email) {
        this.errors.email = 'Email je obavezan.';
      } else if (!this.validEmail(this.form.email)) {
        this.errors.email = 'Email nije validan.';
      }

      if (!this.form.password) {
        this.errors.password = 'Lozinka je obavezna.';
      } else if (this.form.password.length < 6) {
        this.errors.password = 'Lozinka mora imati najmanje 6 karaktera.';
      }

      if (!this.form.confirmPassword) {
        this.errors.confirmPassword = 'Potvrda lozinke je obavezna.';
      } else if (this.form.password !== this.form.confirmPassword) {
        this.errors.confirmPassword = 'Lozinke se ne poklapaju.';
      }

      return Object.keys(this.errors).length === 0;
    },
    validEmail(email) {
      const re = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
      return re.test(email);
    }
  }
}
```

```

    },
    async submitForm() {
      if (this.validateForm()) {
        try {
          const response = await this.$store.dispatch("auth/register", {
            username: this.form.username,
            email: this.form.email,
            password: this.form.password
          });
          this.registerMessage = response.message;
          this.registerSuccess = response.success;
          if (response.success) {
            this.resetForm();
            this.$router.push({ name: 'Home' }); // Redirect to home or any other page after registration
          }
        } catch (error) {
          console.error(error);
          this.registerMessage = 'Došlo je do greške prilikom registracije.';
          this.registerSuccess = false;
        }
      }
    },
    resetForm() {
      this.form.username = "";
      this.form.email = "";
      this.form.password = "";
      this.form.confirmPassword = "";
      this.errors = {};
    }
  };
</script>

```

```

<style scoped>
h2{
  font-style: italic;
  font-weight: bold;
  color:#543310;
  margin-left:10px;
}
form{
  color:#74512D;
}
.is-invalid {
  border-color: #dc3545;
}

```

```
.invalid-feedback {  
  display: block;  
  color: #dc3545;  
}  
  
.alert {  
  color: #721c24;  
  background-color: #f8d7da;  
  border-color: #f5c6cb;  
}  
  
.btn-outline-light {  
  border-color: #AF8F6F;  
  color: #AF8F6F;  
}  
</style>
```

Store

auth

Index.js

```
import actions from "./actions";  
import getters from "./getters";  
import mutations from "./mutations";  
  
export default {  
  namespaced: true,  
  state() {  
    return {  
      users: [],  
      currentUser: JSON.parse(localStorage.getItem('currentUser')) || null,  
    };  
  },  
  mutations,  
  getters,  
  actions,  
};
```

getters.js

```
export default {
```



```
isAuthenticated: state => !!state.currentUser,  
currentUser: state => state.currentUser,  
currentUserRole: state => state.currentUser ? state.currentUser.role : null,  
isAdmin: state => state.currentUser && state.currentUser.role === 'Admin',  
isUser: state => state.currentUser && state.currentUser.role === 'User',  
users: state => state.users  
};
```

mutations.js

```
export default {  
  SET_USERS(state, users) {  
    state.users = users;  
  },  
  SET_CURRENT_USER(state, user) {  
    state.currentUser = user;  
    localStorage.setItem('currentUser', JSON.stringify(user));  
  },  
  LOGOUT_USER(state) {  
    state.currentUser = null;  
    localStorage.removeItem('currentUser');  
  },  
  REMOVE_USER(state, userId) {  
    state.users = state.users.filter(user => user.id !== userId);  
  }  
};
```

actions.js

```
import axios from "axios";  
export default {  
  async register({ commit }, userData) {  
    try {  
      const response = await axios.post('http://localhost/bookstrack/server/register.php', userData);  
      if (response.data.success) {  
        const user = {  
          id: response.data.user.id,  
          username: userData.username,  
          email: userData.email,  
          role: 'User'  
        };  
        commit('SET_CURRENT_USER', user);  
        return { success: true, message: 'User registered successfully' };  
      } else {  

```

```

    return { success: false, message: response.data.message };
  }
} catch (error) {
  console.error(error);
  return { success: false, message: 'Error during registration' };
}
},
async logout(context){
  context.commit("LOGOUT_USER");
},
async login({ commit }, userData) {
  try {
    const response = await axios.post('http://localhost/bookstrack/server/login.php', userData);
    if (response.data.success) {
      const user = response.data.user;
      commit('SET_CURRENT_USER', user);
      return { success: true, message: 'User logged in successfully' };
    } else {
      return { success: false, message: response.data.message };
    }
  } catch (error) {
    console.error(error);
    return { success: false, message: 'Error during login' };
  }
},
async fetchUsers({ commit }) {
  const response = await fetch('http://localhost/bookstrack/server/getUsers.php');
  const users = await response.json();
  commit('SET_USERS', users);
},
async deleteUser({ commit }, userId) {
  await fetch(`http://localhost/bookstrack/server/managerUser.php?id=${userId}`, {
    method: 'DELETE'
  });
  commit('REMOVE_USER', userId);
}
};

```

books

index.js

```

import getters from "../getters"
import mutations from "../mutations"

```

```
import actions from "./actions"
export default{
  namespaced:true,
  state() {
    return {
      books: [],
      selectedGenres:[],
      search:"",
      comments: [],
      userComments: [],
      userFavorites: [],
      categories:[],
      sortOption: 0,
    }
  },
  getters,
  mutations,
  actions
}
```

getters.js

```
export default{
  getBooks(state) {
    let result = state.books;
    if (state.selectedGenres.length) {
      result = result.filter((book) => {
        return book.genreId.some((genre) => state.selectedGenres.includes(genre));
      });
    }
    if (state.search.length) {
      result = result.filter(
        (book) =>
          book.name.toLowerCase().includes(state.search.toLowerCase())
      );
    }
  }
}
```

```
// Sortiranje
if (state.sortOption !== '0') {
  result = result.sort((a, b) => {
    switch (state.sortOption) {
      case 'name_asc':
        return a.name.localeCompare(b.name);
      case 'name_desc':
        return b.name.localeCompare(a.name);
    }
  });
}
```

```

        case 'year_asc':
            return a.published_year - b.published_year;
        case 'year_desc':
            return b.published_year - a.published_year;
    }
});
}

return result;
},
getBook: (state, getters) => (payload) => {
    if (getters.hasBook(payload)) {
        return state.books.find((x) => x.id === payload);
    } else {
        return "Book is not found";
    }
},
hasBook: (state) => (payload) => {
    return state.books.some((book) => book.id === payload);
},
getCommentsByBookId: (state) => (bookId) => {
    return state.comments.filter(comment => comment.bookId === bookId);
},
getGenres:(state)=>{
    return state.categories;
},
SET_FILTERED_BOOKS(state, books) {
    state.filteredBooks = books;
}
}

```

mutations.js

```

export default{
    setOneBook(state, book){
        state.oneBook=book;
    },
    ADD_COMMENT(state, comment) {
        state.comments.push(comment);
    },
    SET_COMMENTS(state, comments) {
        state.comments = comments;
    },
    SET_USER_COMMENTS(state, comments) {

```

```

    state.userComments = comments;
  },
  SET_USER_FAVORITES(state, favorites) {
    state.userFavorites = favorites;
  },
  ADD_TO_FAVORITES(state, bookId) {
    state.userFavorites.push(bookId);
    localStorage.setItem('favorites', JSON.stringify(state.favorites));
  },
  REMOVE_FAVORITE_BOOK(state, bookId) {
    const index = state.userFavorites.indexOf(bookId);
    if (index > -1) {
      state.userFavorites.splice(index, 1);
    }
  },
  setBooks(state, books) {
    state.books = books;
  },
  ADD_BOOK(state, book) {
    state.books.push(book);
  },
  REMOVE_BOOK(state, bookId) {
    state.books = state.books.filter(book => book.id !== bookId);
  },
  SET_CATEGORIES(state, categories) {
    state.categories = categories;
  },
  setGenres(state, selectedOptions) {
    state.selectedGenres = selectedOptions;
  },
  setSearch(state, searchFromComponent) {
    state.search = searchFromComponent;
  },
  setSortOption(state, sortOption) {
    state.sortOption = sortOption;
  }
}

```

actions.js

```

import axios from "axios";
export default {
  async getCategories({ commit }) {
    try {
      const response = await fetch("https://666c7c3c49dbc5d7145e2a4f.mockapi.io/bt/genre");
      const responseData = await response.json();
    }
  }
}

```

```

    commit('SET_CATEGORIES', responseData);
  } catch (error) {
    console.error('Error fetching categories:', error);
  }
},
async loadBooks(context) {
  const response = await fetch("https://666c7c3c49dbc5d7145e2a4f.mockapi.io/bt/books");
  const data = await response.json();
  console.log(data);
  context.commit("setBooks", data);
},
async getOneBook(_, payload) {
  const response = await axios.get("https://666c7c3c49dbc5d7145e2a4f.mockapi.io/bt/books/" + payload);

  return response;
},
async addToFavorites({ commit, rootState }, bookId) {
  const userId = rootState.auth.currentUser.id;
  if (userId) {
    try {
      const response = await axios.post('http://localhost/bookstrack/server/addToFavorites.php', {
        userId,
        bookId
      });
      if (response.data.success) {
        commit('ADD_TO_FAVORITES', bookId);
      } else {
        console.error(response.data.message);
      }
    } catch (error) {
      console.error('Error adding to favorites:', error);
    }
  } else {
    console.error('User is not authenticated');
  }
},
async addComment({ commit }, commentData) {
  try {
    const response = await axios.post('http://localhost/bookstrack/server/comments.php', commentData);
    if (response.data.success) {
      commit('ADD_COMMENT', commentData);
    } else {
      throw new Error('Failed to add comment');
    }
  } catch (error) {
    console.error('Error adding comment:', error);
  }
}

```

```

    },
    async getCommentsByBookId({ commit }, bookId) {
      try {
        const response = await axios.get(`http://localhost/bookstrack/server/comments.php?bookId=${bookId}`);
        if (response.status === 200) {
          commit('SET_COMMENTS', response.data);
          return response.data;
        } else {
          throw new Error('Failed to fetch comments');
        }
      } catch (error) {
        console.error('Error fetching comments:', error);
        throw error;
      }
    },
    async getUserComments({ commit }, username) {
      try {
        const response = await
        axios.get(`http://localhost/bookstrack/server/getComments.php?username=${username}`);
        commit('SET_USER_COMMENTS', response.data);
        return response.data;
      } catch (error) {
        console.error('Error fetching user comments:', error);
        throw error;
      }
    },
    async getUserFavorites({ commit }, userId) {
      try {
        const response = await axios.get(`http://localhost/bookstrack/server/getFavorites.php?userId=${userId}`);
        commit('SET_USER_FAVORITES', response.data);
        return response.data;
      } catch (error) {
        console.error('Error fetching user favorite books:', error);
        throw error;
      }
    },
    async removeBookFromFavorites({ commit }, { userId, bookId }) {
      try {
        const response = await axios.post('http://localhost/bookstrack/server/removeFavorite.php', {
          userId,
          bookId,
        });
        commit('REMOVE_FAVORITE_BOOK', bookId);
        return response;
      } catch (error) {
        console.error('Error removing book from favorites:', error);
        throw error;
      }
    }
  },

```

```

    },
    async addBook({ commit }, book) {
      if (!book || Object.keys(book).length === 0 ||
        !book.name || !book.description || !book.img ||
        !book.published_year || !book.author || !book.genreId.length) {
        console.error('Book object is incomplete or empty');
        return;
      }
      try {
        const response = await axios.post('https://666c7c3c49dbc5d7145e2a4f.mockapi.io/bt/books', book);
        commit('ADD_BOOK', response.data);
      } catch (error) {
        console.error('Error adding book:', error);
      }
    },
    async addGenre(_, genre) {
      try {
        const response = await axios.post('https://666c7c3c49dbc5d7145e2a4f.mockapi.io/bt/genre', genre);
        console.log(response.data);
      } catch (error) {
        console.error('Error adding genre:', error);
      }
    },
    async deleteBook({ commit }, bookId) {
      try {
        const response = await axios.delete('https://666c7c3c49dbc5d7145e2a4f.mockapi.io/bt/books/' + bookId);
        commit('REMOVE_BOOK', bookId);
        console.log(response);
      } catch (error) {
        console.error('Error:', error);
      }
    },
  },
}

```