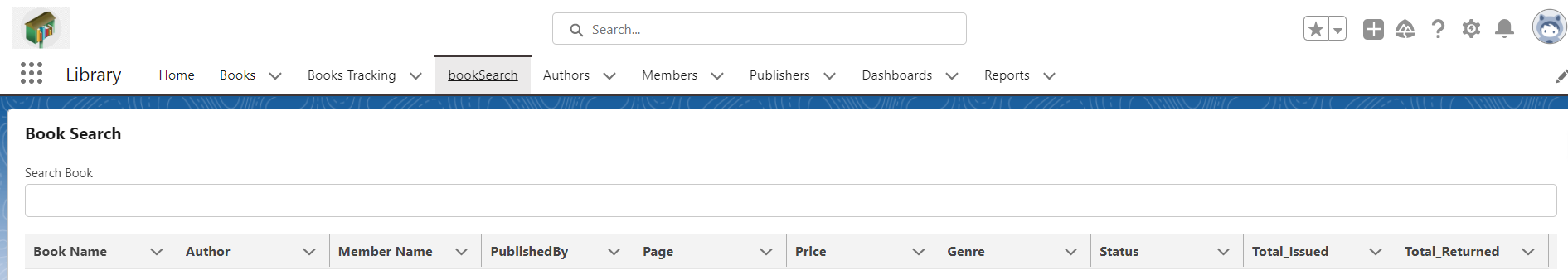
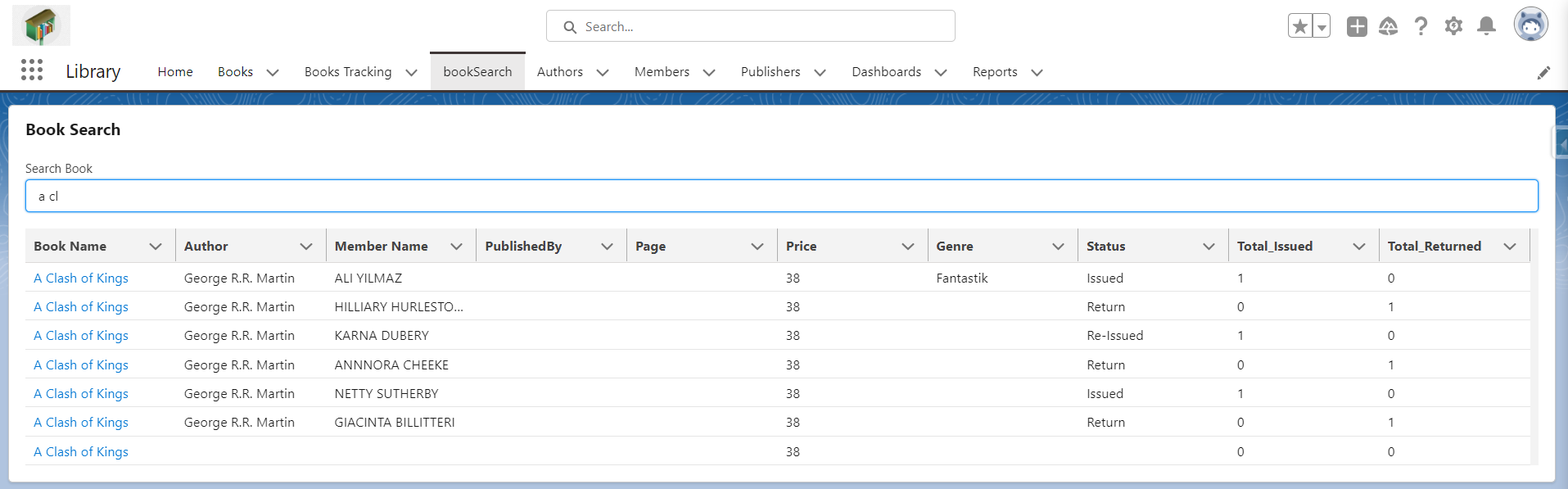
**BOOK SEARCH LİGHTNİNG WEB COMPENENT (LWC)**

**Salesforce Lightning Web Components (LWC),** Salesforce platformunda kullanılmak üzere özel olarak geliştirilen modern ve hafif web bileşenlerini tanımlayan bir geliştirme çerçevesidir. LWC, Salesforce uygulamalarını daha hızlı, daha kullanıcı dostu ve daha özelleştirilebilir hale getirmek için tasarlanmıştır.

Burada kulanıcı için rahat ulaşabileceği ,tab olarak bir LWC compenent ile bir ‘Search’ olanağı sunuldu. Ayrıca BookName için link özelliği ile ilgili kitap detaylarına rahatça ulaşacaktır.

****

****

**BookSearch LWC ; Salesforce Developer Console Ve Visual Studio Code (VsCode) kullanılarak hazırlanmıştır. Vscode içerisinde**

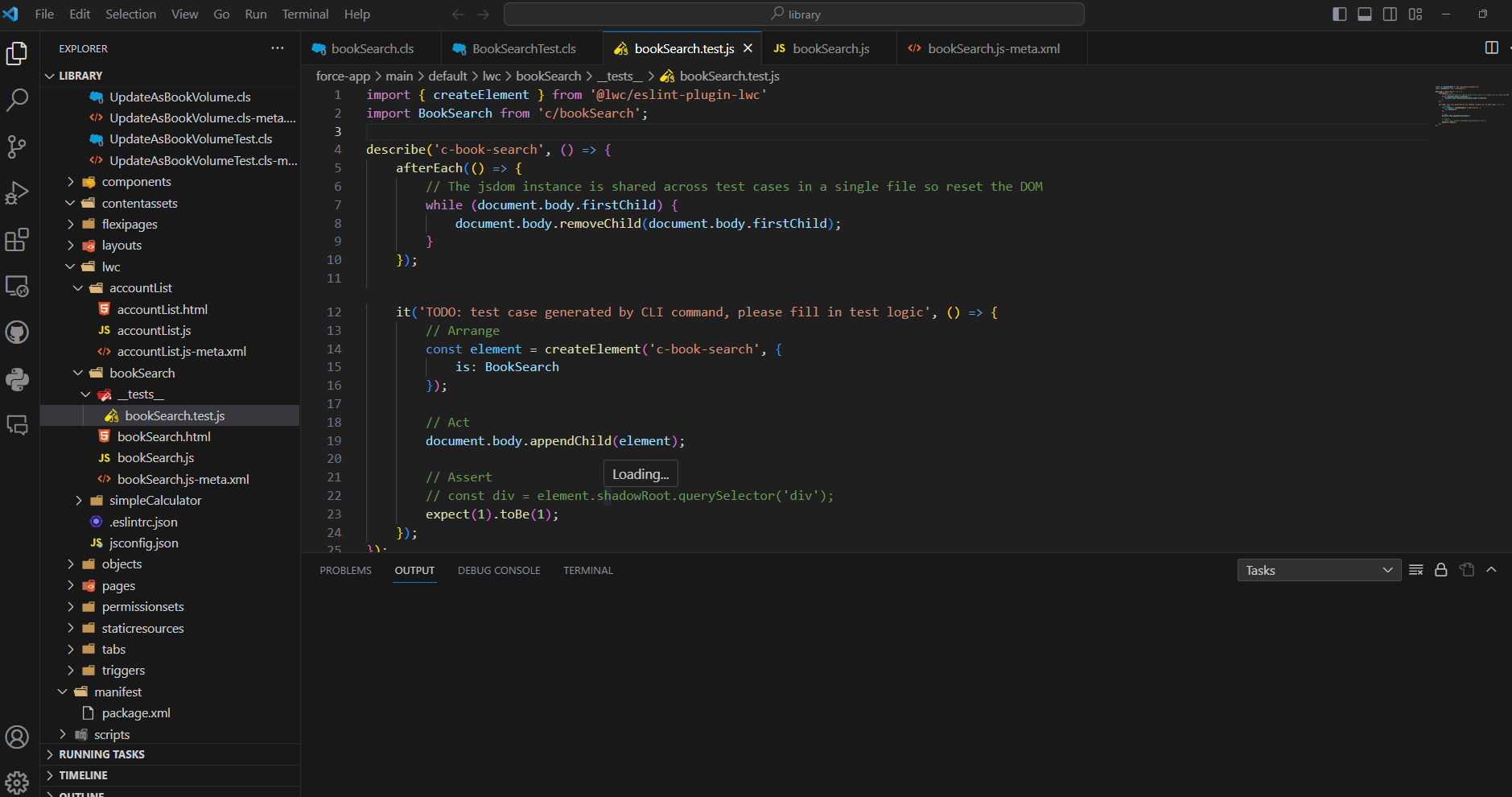
**SFDX:Create Lightning Web Component,**

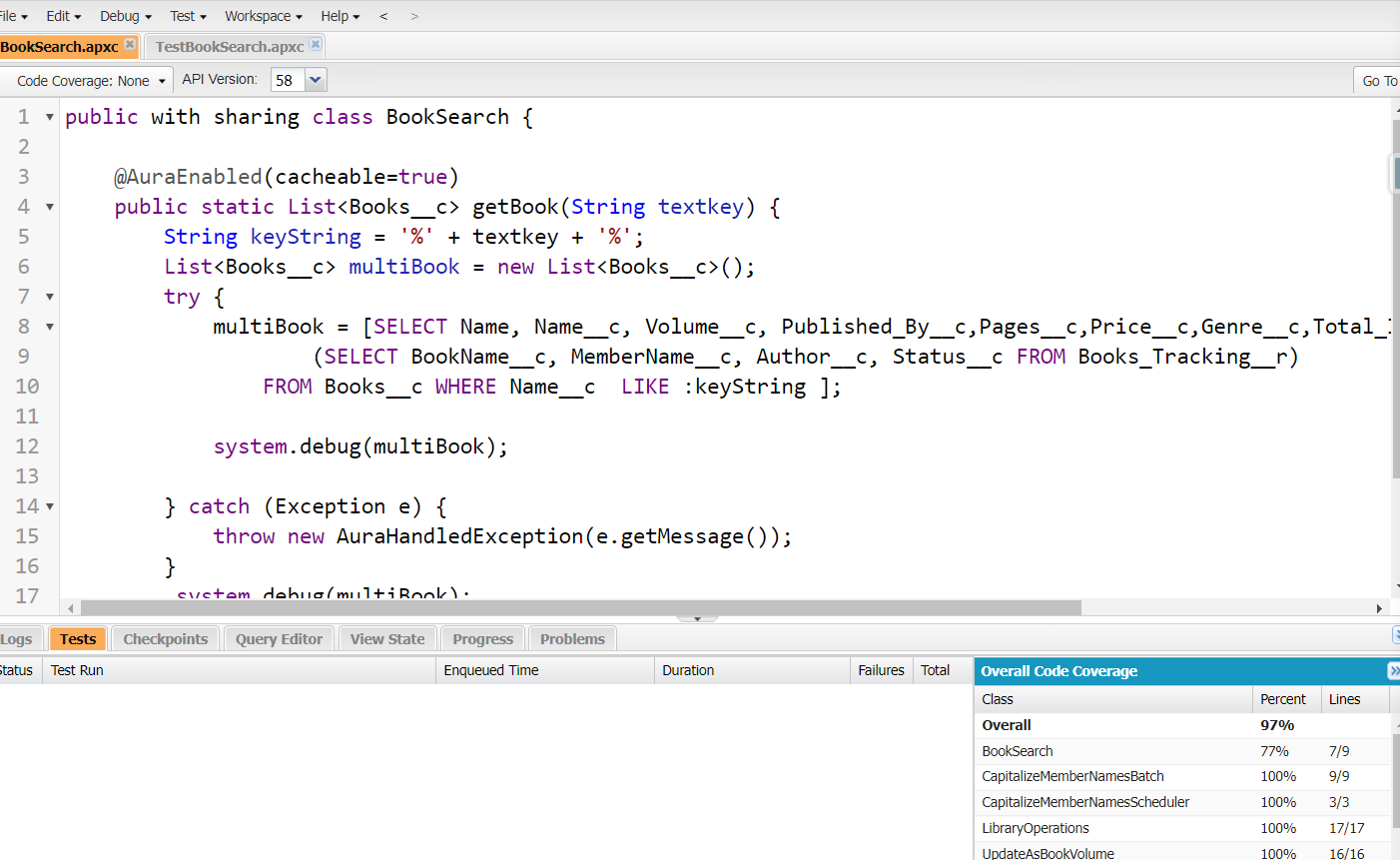
**SFDX: Authorize an Org**

**SFDX: Deploy Source the Org ,**

**SFDX:Retrieve Source in Manifest from Org**

**Yapıları kullanılarak Vscode to Org , Org to Vscode karşılıklı olarak ilişki kurulmuştur.**

****

****

**BookSearch LWC Component yapılandırılırken aşağıdaki kod yapıları oluşturulmuştur.**

**APEX CLASS BookSearch (Developer Console)**

**getBook** adlı bir Aura Enabled yöntemi tanımlar. Bu yöntem, kullanıcı tarafından sağlanan arama metni ile eşleşen kitapları veri tabanından getirir ve bir liste olarak döndürür.

**textkey** parametresi, arama için kullanılacak metin anahtarını alır.

**LIKE :keyString** ifadesi, veri tabanında metin anahtarı ile eşleşen kitapları sorgular.

Veri tabanından çekilen kitap verileri, hata ayıklama amacıyla system.debug ile konsola yazdırılır.

Hata durumunda, AuraHandledException kullanılarak bir hata mesajı yakalanır ve döndürülür.

Bu Apex Class, Salesforce LWC bileşenlerinin kullanabileceği kitap arama işlevselliğini sağlar.

public with sharing class BookSearch {

/\*\*

\* @AuraEnabled(cacheable=true)

\* Bu yöntem, kitap araması yapmak için çağrılır ve veritabanından kitap verilerini getirir.

\* @param textkey Arama için kullanılacak metin anahtarı

\* @return Kitap verilerini içeren bir Liste

\*/

public static List<Books\_\_c> getBook(String textkey) {

// Arama metni ile eşleşen kitapları getirmek için kullanılacak metin anahtarı oluşturuluyor.

String keyString = '%' + textkey + '%';

List<Books\_\_c> multiBook = new List<Books\_\_c>();

try {

// Veritabanından kitapları çekme sorgusu yapılıyor.

multiBook = [SELECT Name, Name\_\_c, Volume\_\_c, Published\_By\_\_c, Pages\_\_c, Price\_\_c, Genre\_\_c, Total\_Issued\_\_c, Total\_Returned\_\_c,

(SELECT BookName\_\_c, MemberName\_\_c, Author\_\_c, Status\_\_c FROM Books\_Tracking\_\_r)

FROM Books\_\_c WHERE Name\_\_c LIKE :keyString ];

// Elde edilen kitap verileri hata ayıklama amacıyla konsola yazdırılıyor.

system.debug(multiBook);

} catch (Exception e) {

// Hata durumunda hata mesajı yakalanıp bir AuraHandledException ile döndürülüyor.

throw new AuraHandledException(e.getMessage());

}

// Kitap verileri döndürülüyor.

return multiBook;

}

}

**APEX TEST CLASS TestBookSearch (Developer Console)**

@isTest

public class TestBookSearch{

@testSetup

static void setupTestData() {

// Create test data for Authors\_\_c (assuming this is the object where Authors are stored)

List<Author\_\_c> authors = new List<Author\_\_c>();

for (Integer i = 0; i < 5; i++) {

Author\_\_c author = new Author\_\_c(

Name\_\_c = 'Author ' + i,

Surname\_\_c = 'Test ' + i

);

authors.add(author);

}

insert authors;

// Create test data for Books\_\_c

List<Books\_\_c> books = new List<Books\_\_c>();

for (Integer i = 0; i < 5; i++) {

Books\_\_c book = new Books\_\_c(

Name\_\_c = 'Book ' + i + ' Name',

Volume\_\_c = 1,

Author\_\_c = authors[i].Id, // Use the Id of the corresponding Author

Price\_\_c = 120.00,

Pages\_\_c=245,

Genre\_\_c='Roman'

);

books.add(book);

}

insert books;

// Create test data for Borrowers\_\_c (assuming this is the object where Borrowers are stored)

List<Member\_\_c> borrowers = new List<Member\_\_c>();

for (Integer i = 0; i < 5; i++) {

Member\_\_c borrower = new Member\_\_c(

Name\_\_c = 'Borrower ' + i,

Surname\_\_c = 'Test ' + i,

Email\_\_c= 'hy@hy.hy'+i

);

borrowers.add(borrower);

}

insert borrowers;

// Create related Book\_Tracking\_\_c records

List<Book\_Tracking\_\_c> trackingRecords = new List<Book\_Tracking\_\_c>();

for (Books\_\_c book : books) {

for (Integer j = 0; j < 2; j++) {

Book\_Tracking\_\_c tracking = new Book\_Tracking\_\_c(

Book\_Name\_\_c = book.Id,

Borrowed\_By\_\_c = borrowers[j].Id, // Use the Id of the corresponding Borrower

Status\_\_c = 'Issued',

Issue\_Date\_\_c=date.today()

);

trackingRecords.add(tracking);

}

}

insert trackingRecords;

}

@isTest

static void testGetBook() {

// Test querying books based on search key

Test.startTest();

List<Books\_\_c> result = BookSearch.getBook('Book 1');

Test.stopTest();

// Verify results

System.assertEquals(1, result.size());

System.assertEquals('Book 1 Name', result[0].Name\_\_c);

System.assertEquals(1.0, result[0].Volume\_\_c); // Volume\_\_c is of type Decimal

System.assertEquals(2, result[0].Books\_Tracking\_\_r.size());

}

@isTest

static void testExceptionHandling() {

Test.startTest();

try {

List<Books\_\_c> result = BookSearch.getBook(null);

} catch (AuraHandledException e) {

System.assert(e.getMessage() != null);

}

Test.stopTest();

}

}

**bookSearch.js-meta.xml (VsCode)**

Bu XML dosyası, bir Salesforce Lightning bileşeninin özelliklerini ve kullanımını tanımlar:

* <apiVersion>: Bu etiket, bileşenin hangi Salesforce Metadata API sürümüyle uyumlu olduğunu belirtir. Bu sürüm, bileşenin yapılandırma ve özelliklerini destekler.
* <isExposed>: Bu etiket, bileşenin dışa açık (kullanılabilir) olup olmadığını belirtir. "true" olarak ayarlandığında, bileşen Salesforce içinde kullanılabilir ve dağıtılabilir hale gelir.
* <targets>: Bu etiket, bileşenin hangi hedeflerde (kullanım senaryolarında) kullanılabileceğini belirtir. Örneğin, lightning App Page hedefi, bileşenin uygulama sayfalarında kullanılabileceğini gösterir. Bu hedefler, bileşenin nerede ve nasıl kullanılabileceğini kontrol etmek için kullanılır.

Bu XML dosyası, bir Salesforce Lightning bileşeninin tanımını içerir ve bu bileşenin ne tür uygulamalarda ve sayfalarda kullanılabileceğini belirler. Bileşen paketlemesi ve dağıtımı sırasında bu tür bir tanımlama, bileşenin doğru şekilde entegre edilmesini ve kullanılmasını sağlar.

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">

<!-- Salesforce Metadata API sürümü -->

<apiVersion>57.0</apiVersion>

<!-- Bileşen paketinin dışa açık (kullanılabilir) olup olmadığını belirtir -->

<isExposed>true</isExposed>

<!-- Bileşenin hedeflerini (kullanım senaryolarını) belirtir -->

<targets>

<!-- Uygulama sayfası (App Page) hedefi -->

<target>lightning\_\_AppPage</target>

<!-- Ana sayfa (Home Page) hedefi -->

<target>lightning\_\_HomePage</target>

<!-- Kayıt sayfası (Record Page) hedefi -->

<target>lightning\_\_RecordPage</target>

<!-- Sekme (Tab) hedefi -->

<target>lightning\_\_Tab</target>

</targets>

</LightningComponentBundle>

**bookSearch.html (VsCode)**

Bu kod bloğu, Salesforce Lightning Web Components (LWC) kullanarak bir kitap arama bileşeninin kullanıcı arayüzünü tanımlar. Bileşen, kullanıcıların bir kitap adıyla kitapları arayabileceği ve sonuçları bir tablo şeklinde görebileceği bir kullanıcı arayüzü sağlar

Bu kullanıcı arayüzü şunları içerir:

**lightning-card:** Bu bileşen, bir kart görünümü sağlar ve "Book Search" başlığını taşır. Bunu, kullanıcı arayüzünün ana bileşeni olarak kullanır.

**lightning-input:** "Search Book" etiketli bir arama girdisi içerir. Kullanıcılar buraya kitap adını girebilirler. onchange olayı, kullanıcının girdiyi değiştirdiğinde çağrılacak handleSearch işlevini tetikler. value özelliği, girdinin değerini searchText değişkenine bağlar.

**lightning-datatable:** Bu bileşen, kitap sonuçlarını görüntülemek için bir tablo sağlar. key-field, verilerin anahtar alanını belirtir. data, kitap sonuçlarını bookList değişkeninden alır. columns, tablonun sütun yapılandırmasını columns değişkeninden alır. hide-checkbox-column özelliği, tabloya varsayılan olarak eklenen seçim kutularını gizler. is-loading, veriler yüklenirken bir yükleniyor göstergesi göstermek için isNoRecordFound durumunu izler.

**template if:true={isNoRecordFound}**: Bu şablon, isNoRecordFound değişkeni true olduğunda, yani kayıt bulunamadığında görünen bir mesajı içerir. Bu, kullanıcılara hiçbir kaydın bulunamadığını bildirir.

Bu kod bloğu, kullanıcıların kitap aramalarını yapmalarını ve sonuçları kullanıcı dostu bir şekilde görüntülemelerini sağlayan bir arayüzü tanımlar.

<template>

<lightning-card title="Book Search">

<!-- Kitap arama kartı başlığı -->

<div class="slds-m-around\_medium">

<!-- Arama girdisi ve olay dinleyici -->

<lightning-input

label="Search Book"

onchange={handleSearch}

value={searchText}

></lightning-input>

</div>

<div class="slds-m-around\_medium">

<!-- Sonuçları görüntülemek için veri tablosu -->

<lightning-datatable

key-field="Id"

data={bookList}

columns={columns}

hide-checkbox-column="true"

is-loading={isNoRecordFound}

></lightning-datatable>

<!-- Kayıt bulunamadı durumunu kontrol etmek için şablon -->

<template if:true={isNoRecordFound}>

<p>No records found.</p>

</template>

</div>

</lightning-card>

</template>

**bookSearch.js (VsCode)**

Bu kod, Salesforce Lightning Web Components kullanarak bir kitap arama bileşeni oluşturur ve kullanıcıların aradıkları kitapları bulmalarını sağlar. Bileşen, bir arama metni girdisi ve sonuçları bir tablo şeklinde görüntülemek için kullanılır. @api ve @track özellikleri, bileşenin dışarıdan alınan verilere ve bileşen içindeki duruma erişimini sağlar. Apex yöntemi olan getBook ise arama sorgusunu yürüten ve sonuçları getiren bir sunucu tarafı işlemi yapar.

import { LightningElement, track, api } from 'lwc';

import getBook from '@salesforce/apex/BookSearch.getBook';

const columns = [

    { label: 'Book Name', fieldName: 'BookURL', type: 'url', typeAttributes: { label: { fieldName: 'Name\_\_c' } } },

    { label: 'Author', fieldName: 'Author\_\_c', type: 'text' },

    { label: 'Member Name', fieldName: 'MemberName\_\_c', type: 'text' },

    { label: 'PublishedBy', fieldName: 'Published\_By\_\_c', type: 'text' },

    { label: 'Page', fieldName: 'Pages\_\_c', type: 'text' },

    { label: 'Price', fieldName: 'Price\_\_c', type: 'text' },

    { label: 'Genre', fieldName: 'Genre\_\_c', type: 'text' },

    { label: 'Status', fieldName: 'Status\_\_c', type: 'text' },

    { label: 'Total\_Issued', fieldName: 'Total\_Issued\_\_c', type: 'text' },

    { label: 'Total\_Returned', fieldName: 'Total\_Returned\_\_c', type: 'text' }

];

export default class BookSearch extends LightningElement {

@track searchText = ''; // Kullanıcının arama metnini tutar

@track bookList = []; // Kitap sonuçlarını saklar

@track isNoRecordFound = false; // Kayıt bulunamadı durumunu izler

columns = columns; // Sütun tanımlarını bileşen içinde kullanmak için atama yapar

@api title; // Bileşen başlığını dışarıdan alır

// Kullanıcının arama girdisini dinleyen işleyici

handleSearch(event) {

this.searchText = event.target.value;

this.loadBooks(); // Kitapları yükleme işlemini başlatır

}

// Kitapları yüklemek için Apex yöntemini çağırır

loadBooks() {

getBook({ textkey: this.searchText }) // Apex yöntemine arama metnini ileterek çağırır

.then(result => {

if (result && result.length > 0) {

// Sonuçlar bulunduysa

this.bookList = result.map(book => {

const { Name\_\_c, Published\_By\_\_c, Pages\_\_c, Price\_\_c, Genre\_\_c, Total\_Issued\_\_c, Total\_Returned\_\_c, Books\_Tracking\_\_r } = book;

if (!Books\_Tracking\_\_r || Books\_Tracking\_\_r.length === 0) {

return {

Name\_\_c,

Published\_By\_\_c,

Pages\_\_c,

Price\_\_c,

Genre\_\_c,

Total\_Issued\_\_c,

Total\_Returned\_\_c,

MemberName\_\_c: null,

Author\_\_c: null,

Status\_\_c: null,

BookURL: '/lightning/r/Books\_c/' + book.Id + '/view'

};

} else {

return {

Name\_\_c,

Published\_By\_\_c,

Pages\_\_c,

Price\_\_c,

Genre\_\_c,

Total\_Issued\_\_c,

Total\_Returned\_\_c,

...Books\_Tracking\_\_r[0],

BookURL: '/lightning/r/Books\_c/' + book.Id + '/view'

};

}

});

this.isNoRecordFound = false; // Kayıt bulunamadı durumunu kapatır

} else {

// Sonuç yoksa

this.bookList = []; // Sonuçları temizler

this.isNoRecordFound = true; // Kayıt bulunamadı durumunu açar

}

})

.catch(error => {

this.isNoRecordFound = true; // Hata durumunu açar

console.error(error); // Hata mesajını konsola yazdırır

});

}}