مدرس: دکتر احسان خامس‌پناه سنا ساری نوایی، علی هدایی، میثاق محقق

MioBook

مقدمه

سیستمی که در طول درس به توسعه آن خواهید پرداخت، یک سیستم کتاب‌فروشی است. در این سیستم، کاربران لیستی از کتاب‌های موجود را مشاهده می‌کنند، به فیلتر کردن کتاب‌ها بر اساس ویژگی‌های آنها می‌پردازند و پس از خرید کتاب، برای آن نظر خود را ثبت می‌کنند.

این فاز تمرین به دو قسمت تقسیم می‌شود:

- در بخش اول، شما منطق دامنه پروژه را در Java پیاده‌سازی می‌کنید. این کار نیازی به پیاده‌سازی مفاهیم بک‌اند وب ندارد و به آن در تمارین بعدی پرداخته خواهد شد. نحوه ارتباط با برنامه در این فاز توسط یک مفسر کامندلاین درون برنامه خواهد بود.
- در بخش دوم، شما با استفاده از HTML، CSS و فریم‌ورک Bootstrap، به پیاده‌سازی چند صفحه Static از این سیستم می‌پردازید. توجه داشته باشید که این صفحات کمی ساده‌سازی شده و کامل نیستند. در فاز چهارم صفحات کامل‌تری به شما داده می‌شود تا با فریم‌ورک React پیاده‌سازی کنید.

همچنین بخش اول نیز ممکن است دچار تغییراتی در طی پروژه‌ها بشود. از این رو مهم است که کد شما تمیز و خوانا بوده و به راحتی و با هزینه نگهداری پایین، تغییر پذیر باشد.

بخش اول: منطق سیستم

خط فرمان

در این تمرین باید تعدادی از عملیات پایه‌ای برنامه خود را پیاده‌سازی کنید. این عملیات در قالب دستوراتی در خط فرمان (کامندلاین) به برنامه شما داده خواهند شد. برای سهولت پیاده‌سازی، نیازی نیست که داده‌ای را جایی ذخیره کنید و همه موجودیت‌های برنامه خود را در حافظه اصلی نگه داشته و متناسب با آنها به دستورات پاسخ دهید. هر دستور، شکل کلی زیر را دارد:

```
command <INPUT_JSON>
```

command نشان‌دهنده نام دستور و INPUT_JSON داده serialize-شده مربوط به آن دستور در قالب JSON است. برای استفاده از این داده، باید ابتدا با استفاده از کتابخانه Jackson آن را deserialize کنید. همچنین ممکن است که INPUT_JSON برای یک دستور وارد نشود.

همانند ورودی‌ها، پاسخ‌ها نیز باید به صورت JSON در خط فرمان چاپ شوند. فرمت پاسخ‌ها به صورت زیر است:

```
{"success": true, "message": "sample message", "data": <RESPONSE_DATA>}
```

همه دستورات دو مورد success و message را دارند ولی ممکن است که data را نداشته باشند.
در ادامه، دستوراتی که باید در این تمرین پیاده‌سازی کنید آمده است.

1. اضافه کردن کاربر

این دستور یک کاربر را به لیست کاربران سامانه اضافه می‌کند. سیستم کتاب‌فروشی دارای دو نوع کاربر است.

- کاربر عادی (customer): این نوع کاربر در سایت ثبت‌نام می‌کند و می‌تواند کتاب بخرد.
- کاربر ادمین (admin): این نوع کاربر در صورت ورود به سایت، قابلیت افزودن کتاب جدید را دارد.

پس اطلاعات کاربران ما شامل نقش (role)، نام کاربری (username)، رمز عبور (password)، ایمیل (email)، آدرس او (address) و مبلغ داخل کیف پول کاربر است.

- نام کاربری و ایمیل برای همه کاربران یکتا می‌باشد. در صورت تکراری بودن هر یک از این دو، باید خطای مناسب داده شود.
- نام کاربری نمی‌تواند شامل حرف‌هایی نظیر نقطه، کاما، علامت سوال و تعجب، اسپیس و نیم‌فاصله و... باشد. صرفا اعداد، حروف بزرگ و کوچک انگلیسی، _ و - مجاز می‌باشند.
- رمز عبور کاربر می‌تواند شامل هر حرفی باشد ولی باید طول آن حداقل 4 کاراکتر باشد.
- ساختار ایمیل باید صحیح باشد (به طور مثال، test@example.com).
- آدرس کاربر شامل کشور و شهر او خواهد بود.
- نقش کاربر صرفا دو مورد ذکر شده (یعنی customer و admin) است. در صورت دریافت ورودی غیر از این دو حالت باید خطای مناسب برگردانده شود.
- مبلغ داخل کیف پول هر کاربر پس از ثبت‌نام باید صفر باشد.

نمونه دستور:

```
add_user {"role": "customer", "username": "test-name", "password": "1234", "email": "my.mail@mail.com", "address": {"country": "Iran", "city": "Karaj"}}
```

نمونه خروجی:

```
{"success": true, "message": "User added successfully."}
```

2. اضافه کردن نویسنده

این دستور یک نویسنده را به لیست نویسندگان کتاب‌ها اضافه می‌کند. هر نویسنده شامل نام نویسنده (name)، اسم مستعار (penName)، ملیت (nationality)، تاریخ به دنیا آمدن (born) و تاریخ فوت (died) است.

- نام نویسندگان در کل سیستم یکتا است و هنگام افزودن نویسنده جدید بررسی می‌شود.
- فرمت تاریخ‌ها به صورت "yyyy-mm-dd" می‌باشد (ولی نباید در کد به صورت استرینگ ذخیره شوند).
- تاریخ فوت برای نویسنده‌ای که زنده است، حاوی مقدار نیست. این یعنی در JSON ورودی فیلد آن وجود نخواهد داشت.
- دستور اضافه کردن نویسنده، نام کاربری که دارد او را اضافه می‌کند هم دریافت می‌کند و سپس بررسی می‌کند که حتما کاربر ادمین باشد. اگر کاربر عادی باشد، خطا بازگردانده می‌شود.

نمونه دستور:

```
add_author {"username": "admin_user", "name": "author name", "penName": "abc",
"born": "1982-04-12"}
```

نمونه خروجی:

```
{"success": true, "message": "Author added successfully."}
```

3. اضافه کردن کتاب

این دستور یک کتاب را به لیست کتاب‌های فروشگاه اضافه می‌کند. هر کتاب شامل نام (title)، نویسنده (author)، ناشر (publisher)، سال نشر (year)، ژانرها (genres)، قیمت (price)، خلاصه (synopsis) و محتوا (content) است.

- نام کتاب در بین همه کتاب‌ها یکتا است. یعنی در صورت دریافت نام تکراری، خطا می‌دهیم.
- اگر نویسنده‌ای با نام مشخص شده وجود نداشته باشد، خطای مناسب می‌دهیم.
- ناشر صرفا اسم ناشر است و سال نشر باید عددی طبیعی باشد.
- ژانر کتاب یک لیست از ژانرها مانند mystery و romance است. هر کتاب حداقل یک ژانر باید داشته باشد.
- قیمت کتاب عددی حسابی است که در واحد cent نگهداری می‌شود. بنابراین اگر قیمت یک کتاب 1.99 دلار است، ما 199 سنت برای آن نگه می‌داریم.
- خلاصه کتاب به همه کاربران نشان داده می‌شود ولی محتوای آن فقط پس از خریدن کتاب در دسترس خواهد بود.
- دستور اضافه کردن کتاب، نام کاربری که دارد کتاب را اضافه می‌کند هم دریافت می‌کند و سپس بررسی می‌کند که حتما کاربر ادمین باشد. اگر کاربر عادی باشد، خطا بازگردانده می‌شود.

نمونه دستور:

```
add_book {"username": "admin_user", "title": "sample book", "author": "sample
name", "publisher": "name", "year": 2012, "price": 250, "synopsis": "lorem",
"content": "lorem ipsum", "genres": ["horror", "thriller"]}
```

نمونه خروجی:

```
{"success": true, "message": "Book added successfully."}
```

4. اضافه کردن کتاب به سبد خرید

کاربران می‌توانند تعدادی کتاب را به سبد خرید خود اضافه کنند تا در نهایت آنها را بخرند. برای این کار به نام کاربری فرد مد نظر و نام کتاب نیاز داریم.

- کاربر و کتاب باید در سیستم وجود داشته باشند و در غیر این صورت خطا دریافت می‌کنیم.
- نام کاربری وارد شده باید متعلق به یک کاربر عادی باشد. در صورتی که دستور توسط یک ادمین اجرا شود، خطا بازگردانده می‌شود.
- سبد خرید کاربر حداکثر می‌تواند دارای 10 کتاب باشد. در صورتی که 10 کتاب در سبد او وجود داشته باشد و اقدام به اضافه کردن کتاب جدید کند، خطای مناسب برگردانده می‌شود.

نمونه دستور:

```
add_cart {"username": "user_name", "title": "book title"}
```

نمونه خروجی:

```
{"success": true, "message": "Added book to cart."}
```

5. حذف کردن کتاب از سبد خرید

کاربران می‌توانند کتاب‌های اضافه شده به سبد خریدشان را پاک کنند. این دستور مانند دستور اضافه کردن آن، به نام کاربر و کتاب نیاز دارد.

- در صورتی که نام کاربری وجود نداشته باشد خطا داریم. همچنین نام کاربری باید مربوط به یک کاربر عادی باشد و در غیر این صورت خطا داریم.
- در صورتی که نام کتاب وارد شده در کل سیستم وجود نداشته باشد، سیستم خطای مناسب می‌دهد. در صورتی که کتاب وارد شده در سبد خرید کاربر نباشد نیز خطای مربوطه را اعلام می‌کنیم.

نمونه دستور:

```
remove_cart {"username": "user_name", "title": "book title"}
```

نمونه خروجی:

```
{"success": true, "message": "Removed book from cart."}
```

6. واریز به کیف پول

این دستور به کیف پول کاربر مقدار گفته شده را واریز می‌کند.

- این دستور فقط برای کاربران عادی بوده و باید شروط آن بررسی شود.
- مبلغ به واحد cent بوده و نباید صفر یا منفی باشد.
- حداقل مبلغ قابل واریز 1 دلار یا 1000 سنت می‌باشد.

نمونه دستور:

```
add_credit {"username": "name", "credit": 8200}
```

نمونه خروجی:

```
{"success": true, "message": "Credit added successfully."}
```

7. تکمیل خرید کتاب‌ها

با این دستور، کاربر تمام کتاب‌هایی که در سبد خرید خود اضافه کرده است را می‌خرد تا بتواند بعدا آنها را مطالعه کند.

- پس از بررسی وجود کاربر عادی، باید سبد خرید او دارای حداقل یک کتاب باشد. در صورت خالی بودن سبد خرید، به او خطا می‌دهیم.
- کاربر باید به اندازه جمع مبالغ کتاب‌های موجود در سبد خریدش، پول در کیف پول خود داشته باشد. در غیر این صورت باید خطا به او داده شود و خرید صورت نمی‌گیرد.
- با انجام عمل خرید، از موجودی کیف پول کاربر کاسته شده و یک خرید جدید ایجاد می‌شود. ما در این سیستم تاریخچه خریدهای کاربران (که شامل مجموعه نام کتاب‌ها و مبالغ است) را نگه می‌داریم.
- در خروجی این دستور، در فیلد data اطلاعاتی از جمله تعداد کتاب‌های خریداری شده (برابر تعداد کتاب‌هایی که در سبد خرید بوده‌اند)، مبلغ کل خرید و تاریخ و زمان انجام خرید گفته می‌شوند.
- به فرمت تاریخ و زمان داده شده توجه داشته باشید.

نمونه دستور:

```
purchase_cart {"username": "name"}
```

نمونه خروجی:

```
{"success": true, "message": "Purchase completed successfully.", "data": {
"bookCount": 2, "totalCost": 4200, "date": "2025-02-04 15:02:28"}}
```

8. قرض گرفتن کتاب

این سیستم کتاب‌فروشی، قابلیت قرض کتاب را به مشتریان خود می‌دهد. این قابلیت به این صورت کار می‌کند که هر کاربر می‌تواند تا حداکثر 10 روز، یک کتاب را قرض بگیرد. مبلغ قرض کتاب بنا بر تعداد روز قرض کتاب

محاسبه می‌شود. به عنوان مثال، اگر کتابی 20 دلار باشد، در صورت قرض آن به مدت 4 روز، کاربر صرفا 4/10 مبلغ آن را پرداخت می‌کند (در اینجا 8 دلار).
کتاب قرض شده نیز همانند خرید عادی، ابتدا وارد سبد خرید کاربر شده و سپس کاربر اقدام به خرید آنها می‌کند. در صورت خرید سبد خرید، کتاب خریده شده از مدت انجام تراکنش خرید تا تعداد روز تعیین شده آینده، در دسترس کاربر خواهد بود. در صورت گذشتن تاریخ استفاده از کتاب، آن کتاب از لیست دارایی‌های کاربر حذف شده و دیگر نمی‌تواند محتوای آن را مطالعه کند.

- این دستور نام کاربر عادی و کتاب مد نظر او را گرفته و در ابتدا همه شروط معتبر بودن آنها را بررسی می‌کند.
- مقدار روزهای قرض باید بین 1 تا 9 باشد (چون که قرض کردن به مدت 10 روز، قیمتی معادل قیمت خرید کل کتاب دارد).
- مبلغ وارد شده در سبد خرید، کسری از قیمت کل کتاب است و تعداد روز آن، از ساعتی که سبد خرید خریداری می‌شود شروع می‌شود (به طور مثال، اگر کاربر امروز ساعت 8 صبح یک کتاب را به مدت یک روز قرض کرده باشد، تا 8 صبح روز بعد به آن دسترسی خواهد داشت).

نمونه دستور:

```
borrow_book {"username": "name", "title": "book1", "days": 2}
```

نمونه خروجی:

```
{"success": true, "message": "Added borrowed book to cart."}
```

9. اضافه کردن بازخورد

با این دستور کاربر می‌تواند بازخورد خود را نسبت به یک کتاب ثبت کند. هر بازخورد متشکل از سه بخش است:

- امتیاز: یک عدد طبیعی بین 1 تا 5.
- کامنت: یک متن نوشته شده توسط کاربر.
- تاریخ: زمانی که کاربر بازخورد خود را ثبت کرده است.

در ثبت بازخورد به نکات زیر توجه کنید:

- در این فاز فرض کنید که کاربر عادی می‌تواند به همه کتاب‌ها بازخورد دهد.
- در صورتی که کاربر وجود نداشته باشد، خطای مناسب برگردانده شود. توجه داشته باشید که کاربر با نقش ادمین نمی‌تواند بازخورد ثبت کند.
- در صورت عدم وجود کتاب، خطای مناسب برگردانده شود.
- امتیازدهی باید صحت سنجی شود و اگر خارج از بازه گفته شده بود خطای مناسب داده شود.
- متن کاربر می‌تواند شامل هر رشته‌ای باشد و تاریخ بازخورد توسط سیستم ثبت می‌شود.

نمونه دستور:

```
add_review {"username": "user", "title": "book1", "rate": 4, "comment": "This is
the perfect book!"}
```

نمونه خروجی:

```
{"success": true, "message": "Review added successfully."}
```

10. مشاهده مشخصات کاربر

این دستور مشخصات کاربر شامل نام (username)، نقش (role)، ایمیل (email)، آدرس (address) و مبلغ کیف پول او را برمی‌گرداند.

- در صورت عدم وجود کاربر، خطای مناسب برگردانده شود.
- در صورتی که کاربر ادمین است، نیازی به فیلد مبلغ کیف پول (balance) نیست.

نمونه دستور:

```
show_user_details {"username": "name"}
```

نمونه خروجی:

```
{"success": true, "message": "User details retrieved successfully.", "data": {
"username": "book name", "role": "customer", "email": "mail@mail.com", "address":
{"country": "Iran", "city": "Karaj"}, "balance": 2000}}
```

11. مشاهده اطلاعات نویسنده

این دستور مشخصات یک نویسنده کتاب شامل نام (name)، اسم مستعار (penName)، ملیت (nationality)، تاریخ به دنیا آمدن (born) و تاریخ فوت (died) را بازمی‌گرداند.

- در صورت عدم وجود نویسنده، خطای مناسب برگردانده شود.

نمونه دستور:

```
show_author_details {"username": "name"}
```

نمونه خروجی:

```
{"success": true, "message": "Author details retrieved successfully.", "data": {
"name": "author name", "penName": "abc", "nationality": "German", "born":
"1982-04-12", "died": "2011-10-08"}}
```

12. مشاهده مشخصات کتاب

این دستور مشخصات کتاب شامل نام (title)، نویسنده (author)، ناشر (publisher)، سال نشر (year)، ژانرها (genres)، قیمت (price)، خلاصه (synopsis) و همچنین میانگین بازخوردهای کاربران برای آن کتاب (averageRating) را بازمی‌گرداند.

- در صورت عدم وجود کتاب، خطای مناسب برگردانده شود.

نمونه دستور:

```
show_book_details {"title": "book title"}
```

نمونه خروجی:

```
{"success": true, "message": "Book details retrieved successfully.", "data": {
"title": "book title", "author": "author1", "publisher": "publisher1", "genres":
["science", "novel"], "year": "2024", "price": 2200, "synopsis": "lorem",
"averageRating": 4.5}}
```

13. مشاهده محتوای کتاب

کاربر با اجرای این دستور، محتوای کتاب‌هایی که خریده است را مشاهده می‌کند. به این منظور نام کتاب داده شده و اگر کتاب در دارایی‌های کاربر بود (یا به طور کامل خریداری شده و یا قرض گرفته شده است و مدت استفاده از آن نگذشته است)، قابلیت خواندن آن را خواهد داشت.

- در صورت عدم وجود کتاب یا کاربر، خطای مناسب برگردانده شود.
- در صورتی که کتاب در دارایی‌های کاربر نیست، باید خطا نشان داده شده و از دیدن محتوای کتاب جلوگیری شود.

نمونه دستور:

```
show_book_content {"username": "name", "title": "book title"}
```

نمونه خروجی:

```
{"success": true, "message": "Book content retrieved successfully.", "data": {
"title": "book title", "content": "Lorem ipsum."}}
```

14. مشاهده بازخوردهای کتاب

با این دستور، بازخوردهای ثبت شده برای کتاب به همراه میانگین امتیاز آنها بازگردانده می‌شود.

- در صورت عدم وجود کتاب، خطای مناسب برگردانده شود.

نمونه دستور:

```
show_book_reviews {"title": "book1"}
```

نمونه خروجی:

```
{"success": true, "message": "Book reviews retrieved successfully.", "data": {
"title": "book1", "reviews": [ {"username": "user1", "rate": 4, "comment": "Nice."}
, {"username": "user2", "rate": 5, "comment": "Very nice."} ], "averageRating":
4.5}}
```

15. مشاهده سبد خرید

این دستور سبد خرید کاربر (که حداکثر 10 کتاب می‌توانست در آن باشد) را به همراه جمع مبلغ آنها نشان می‌دهد.

- در صورت عدم وجود کاربر یا ادمین بودن او، خطای مناسب برگردانده شود.
- در صورت خالی بودن سبد خرید، totalCost مقدار 0 گرفته و items لیستی خالی خواهد بود.
- فیلد isBorrowed برای کتاب‌هایی که قرار است قرض شوند مقدار true و در غیر این صورت false می‌گیرد.
- در صورتی که کتاب قرضی باشد، فیلد borrowDays تعداد روزهای آن را نشان می‌دهد.
- فیلد finalPrice در صورتی که کتاب قرضی نیست، معادل همان price و در غیر این صورت، مقدار قابل پرداخت با توجه به borrowDays را نشان می‌دهد.
- توجه کنید که فیلد اعشاری برای قیمت نداریم و در انجام تقسیم، رو به پایین قطع می‌کنیم.

نمونه دستور:

```
show_cart {"username": "name"}
```

نمونه خروجی:

```
{"success": true, "message": "Buy cart retrieved successfully.", "data": {
"username": "name", "totalCost": 200, "items": [{"title": "book1", "author":
"author1", "publisher": "publisher1", "genres": ["horror"], "year": 2024, "price":
1000, "isBorrowed": true, "finalPrice": 200, "borrowDays": 2}] }}
```

16. مشاهده تاریخچه خرید

با این دستور تاریخچه خرید یک کاربر بازگردانده می‌شود. پس از نهایی کردن سبد خرید توسط هر کاربر، آیتم‌های خریداری شده به همان شکل که در سبد خرید قابل مشاهده بود، به تاریخچه خرید او اضافه می‌شود.

- در صورت عدم وجود کاربر یا ادمین بودن او، خطای مناسب برگردانده شود.
- در صورت خالی بودن تاریخچه خرید کاربر، لیست purchaseHistory خالی خواهد بود.

نمونه دستور:

```
show_purchase_history {"username": "name"}
```

نمونه خروجی:

```
{"success": true, "message": "Purchase history retrieved successfully.", "data": {
"username": "name", "purchaseHistory": [{"purchaseDate": "2025-02-10 14:00:00",
"items": [{"title": "book1", "author": "author1", "publisher": "publisher1",
"genres": ["science", "novel"], "year": "2024", "isBorrowed": true, "borrowDays":
2, "price": 1000, "finalPrice": 200}], "totalCost": 200}]}}
```

17. مشاهده لیست کتاب‌های خریداری شده

کاربر با این دستور می‌تواند لیست کتاب‌هایی که خریده و یا قرض گرفته را مشاهده کند. با گذشتن تاریخ قرض کتاب، این دستور دیگر آن کتاب را نمایش نمی‌دهد (و دستور مشاهده محتوای کتاب نیز به طبع کار نمی‌کند).

- در صورت عدم وجود کاربر یا ادمین بودن او، خطای مناسب برگردانده شود.

نمونه دستور:

```
show_purchased_books {"username": "user"}
```

نمونه خروجی:

```
{"success": true, "message": "Purchased books retrieved successfully.", "data":
{"username": "user1", "books": [{"title": "book1", "author": "author1",
"publisher": "publisher1", "category": ["fiction"], "year": 2010, "price": 4600,
"isBorrowed": false}]}}
```

18. جست‌وجو کتاب‌ها

در این سیستم 4 نوع جست‌وجو برای کتاب‌ها داریم:

- جست‌وجو بر اساس نام کتاب
- جست‌وجو بر اساس نام نویسنده
- جست‌وجو بر اساس ژانر کتاب
- جست‌وجو بر اساس سال انتشار کتاب

18.1. جست‌وجو بر اساس نام کتاب

با این دستور کاربر می‌تواند کتاب‌ها را بر اساس نام آنها جست‌وجو کند. در اینجا اگر نام کتاب صرفا شامل عبارت نوشته شده هم باشد بازگردانده می‌شود.

نمونه دستور:

```
search_books_by_title {"title": "book1"}
```

نمونه خروجی:

```
{"success": true, "message": "Books containing 'book1' in their title:", "data": {
"search": "book1", "books": [{"title": "the book1", "author": "author1",
"publisher": "publisher1", "genres": ["fiction"], "year": 2024, "price": 1000,
"synopsis": "Lorem"}]}}
```

18.2. جست‌وجو بر اساس نام نویسنده

با این دستور کاربر می‌تواند کتاب‌ها را بر اساس نام نویسنده آنها جست‌وجو کند. اگر نویسنده کتاب شامل عبارت نوشته شده باشد بازگردانده می‌شود.

نمونه دستور:

```
search_books_by_author {"name": "author1"}
```

نمونه خروجی:

```
{"success": true, "message": "Books by 'author1':", "data": { "search": "author1",
"books": [{"title": "book1", "author": "author1", "publisher": "publisher1",
"genres": ["fiction"], "year": 2024, "price": 1000, "synopsis": "Lorem"}]}}
```

18.3. جست‌وجو بر اساس ژانر کتاب

با این دستور کاربر می‌تواند کتاب‌ها را بر اساس یک ژانر جست‌وجو کند. اگر کتابی به طور دقیق دارای ژانر وارد شده باشد، در لیست خروجی بازگردانده شود.

نمونه دستور:

```
search_books_by_genre {"genre": "fiction"}
```

نمونه خروجی:

```
{"success": true, "message": "Books in the 'fiction' genre:", "data": { "search":
"fiction", "books": [{"title": "book1", "author": "author1", "publisher":
"publisher1", "genres": ["fiction"], "year": "2024", "price": 1000, "synopsis":
"Lorem"}]}}
```

18.4. جست‌وجو بر اساس سال انتشار کتاب

با این دستور کاربر می‌تواند کتاب‌ها را بر اساس سال انتشار کتاب آنها جست‌وجو کند. در اینجا اگر سال انتشار کتاب در بازه مشخص شده (ابتدا و انتهای بازه هم شامل می‌شود) باشد، در خروجی برگردانده می‌شود.

- باید بازه داده شده صحت سنجی شود و ابتدای بازه کوچکتر مساوی انتهای بازه باشد.

نمونه دستور:

```
search_books_by_year {"from": "2023", "to": "2024"}
```

نمونه خروجی:

```
{"success": true, "message": "Books published from 2023 to 2024:", "data": {
"search": "2023-2024", "books": [{"title": "book1", "author": "author1",
"publisher": "publisher1", "genres": ["fiction"], "year": 2023, "price": 1000,
"synopsis": "Lorem"}]}}
```

نکات تکمیلی

- برنامه شما باید هرگونه ورودی کامندلاین (از جمله ورودی خالی، دستور ناموجود، ندادن پارامترها، اشتباه بودن مقدار یا تایپ پارامترها، خراب بودن JSON و...) را با خطاهای مناسب هندل کرده و خراب نشود.
- تمامی داده‌های وارد شده در برنامه، در حافظه برنامه نگه داشته می‌شوند و نیازی به ثبت آنها در فایل یا دیتابیس نیست.
- بهتر است وقتی یک کلاس از فیلدهای یک کلاس دیگر است، به جای نگهداری ID آن، پوینتر به خود اینستنس آن کلاس را قرار دهیم.
- سعی کنید در طراحی خود، بخش پردازش JSON داده شده و بخش منطق سیستم را از هم جدا کنید. این کار موجب کاهش Coupling بین نوع ورودی داده شده و کارکرد سیستم می‌شود. به این منظور، پس از خواندن خط ورودی داده شده، در یک کلاس Validation-های مورد نیاز JSON داده شده متناسب با کامند آن را انجام دهید، و پس از آن در کلاسی دیگر تابع‌های اصلی سیستم را قرار دهید که پارامترها و خروجی‌های آنها تایپ‌های جاوا هستند و دانشی از JSON ندارد.

بخش دومی: صفحات استتیک

در این قسمت از پروژه، صفحات پیاده‌سازی شده به صورت ایستا هستند و ارتباطی با سرور و منطق برنامه در قسمت قبلی نخواهند داشت. در فازهای بعدی، رابط کاربری بیشتر توسعه یافته و در نهایت به سرور متصل خواهد شد.

سمت کاربر

هدف این بخش آشنایی هرچه بیشتر شما با HTML، CSS و فریم‌ورک Bootstrap است. در این فاز مجاز به استفاده از هیچ‌گونه کد و کتاب‌خانه به زبان JavaScript نیستید. شما باید صفحات داده شده را مانند طراحی‌هایی که از محیط کاربری پروژه انجام شده و از این لینک قابل دسترسی اند، پیاده‌سازی نمایید.
توجه کنید که اندازه‌ها در یک طراحی فیگما، بر حسب یک اندازه صفحه خاص هستند و پیاده‌سازی شما باید در عرض‌های صفحه مختلف سایت را به خوبی نشان بدهد. در این فاز نیازی به طراحی Responsive صفحات نیست ولی در فازهای بعدی اجباری می‌شود. پس خوب است که این مورد را در نظر داشته باشد.

طراحی‌های کنونی شامل صفحه خانه (Home)، نتایج جست‌وجو (Search)، پنل مشتری (Customer)، صفحه اطلاعات کتاب (Book) و صفحات لاگین و ثبت‌نام است. همچنین در کنار اینها، یک صفحه برای خطای 404 و یک صفحه برای سایر خطاها (شامل متن خطا) را با خلاقیت خود طراحی کنید که از تم کلی سایت پیروی کند.
تمامی موارد لازم در لینک داده شده در اختیار شما قرار داده شده است. به مواردی مانند box-shadow و border-radius و... نیز توجه کنید.

نکات تکمیلی

- صفحات شما باید به زبان انگلیسی باشد.
- فایل‌های HTML و CSS باید از هم جدا باشند و استفاده از صفت style در صفحه HTML مجاز نیست.
- در مورد Reset CSS و Normalize CSS تحقیق کنید و نمونه‌هایی از آنها را مطالعه کنید. مسئولیت آنها در فریم‌ورک Bootstrap توسط ماژول Reboot انجام شود و نیازی به استفاده مستقیم از آنها نیست.
- از آخرین نسخه فریم‌ورک Bootstrap (نسخه 5.3) استفاده کنید. فایل CSS مربوط به آن را در کنار فایل‌ها خود گذارید (از CDN استفاده نکنید). همچنین استفاده از فایل JS این فریم‌ورک نیز مجاز نمی‌باشد.
- نکات مربوط به شبکه‌بندی صفحات را حتما رعایت کنید و از امکان Grid در Bootstrap استفاده کنید.
- پیاده‌سازی صفحات باید کاملا منطبق بر طراحی‌های داده شده به شما باشد. توجه داشته باشید که در این طراحی‌ها اندازه اجزای صفحات نسبی هستند و نیازی نیست که دقیقا همان اندازه‌ها استفاده شوند. ولی پیاده‌سازی شما باید از نظر ظاهر و ترکیب‌بندی همانند عکس‌ها باشد، یعنی نسبت اندازه و مکان اجزا رعایت شده باشد.

- به ضخامت متون، موارد سایه، شعاع و رنگ‌ها توجه کنید.
- از شما انتظار می‌رود از کلاس‌های CSS و نحوه استفاده از آنها توجه داشته باشید. کلاس‌ها باید تکرار کد جلوگیری کنند و با طراحی عمومی آنها، امکان استفاده در جاهای مختلف را داشته باشند.
- شما باید تا جای ممکن از کلاس‌هایی که Bootstrap در اختیار شما قرار می‌دهد استفاده کنید و فقط در صورت نیاز از CSS خود استفاده کنید.
- کدهای CSS باید تا جای ممکن خلاصه باشند (با کمترین property-ها به نتیجه مد نظر برسند) و دارای قوانین اضافی که روی المان تأثیری نمی‌گذارند نباشد.
- از متغیرهای CSS و همچنین دستورات پیشرفته‌تری مانند calc می‌توانید استفاده کنید.
- با واحدهای اندازه‌گیری در CSS مانند px، em، rem، %، ch، vh و vw آشنا شوید و از آنها در جاهای مناسب استفاده کنید. به طور مثال، اندازه فونت نباید از واحد px باشد.
- یکی از ابزارهای بسیار کاربردی که در هنگام توسعه کدهای HTML/CSS به وفور از آن استفاده می‌شود، Inspector مرورگر است. این ابزار در مرورگرهای مختلف با نام‌های مختلفی شناخته می‌شود. از این ابزار می‌توانید برای تغییر در لحظه مشخصات CSS و همچنین دیدن مشخصات اعمال شده بر روی المان‌های صفحه استفاده کنید. حتما نحوه کار با این ابزار را یاد بگیرید تا با آن بتوانید هم ساختار HTML و هم زنجیر CSS را تحلیل کنید. با این ابزار می‌توانید نحوه پیاده‌سازی قسمت‌های مختلف سایت‌ها را ببینید و کار خود را دیباگ کنید.
- توصیه می‌شود که برای پیاده‌سازی اجزای صفحات که اکثرا اجزاء معقول در طراحی وب هستند، از گوگل بسیار کمک بگیرید. وبسایت‌هایی نظیر CSS-Tricks و W3Schools مطالب مفید با بیان خوبی دارند. همچنین می‌توانید داک Bootstrap را برای دیدن اجزای آماده آنها بررسی کنید.
- توجه داشته باشید که عکس‌های کتاب‌ها لینک به صفحه آن کتاب می‌باشند.
- شاخصه position در CSS نحوه قرارگیری یک آیتم در صفحه را تعیین می‌کند. مثلا برای ثابت کردن نوار header بالای صفحه باید از این شاخصه استفاده کنید. همچنین جایی که دو المان روی هم قرار می‌گیرند، احتمالا نیاز به استفاده از ترکیب relative و absolute positioning خواهد بود. با این شاخصه به طور کامل آشنا شوید.
- شاخصه display در CSS نحوه نمایش یک آیتم در صفحه را تعیین می‌کند. راجع به مقادیری که این شاخصه قبول می‌کند مطالعه کنید. علاوه بر block و inline، نحوه نمایش flexbox را داریم که با استفاده از آن می‌توانیم به راحتی چیدمان المان‌ها در صفحه را تعیین کنیم. با flexbox به طور کامل آشنا شوید. شما برای چینش کلی صفحات خود از قابلیت Grid دوازده ستونی Bootstrap و همچنین کلاس‌های مربوط به flexbox آن استفاده خواهید کرد.

Best Practice-ها

این قسمت از صورت پروژه، به معرفی برخی از Best Practice-ها در حوزه مهندسی نرم افزار می‌پردازد. در هر بخش، هر یک از اصول به صورت خلاصه معرفی می‌شوند تا صرفا با آنها آشنا بشوید. خوب است که این اصول را در ذهن خود داشته باشید و برای درک بهتر و پیروی از آنها در کد خود، درباره آنها تحقیق کنید.

KISS (Keep It Simple, Stupid!)

اصل KISS یک اصل طراحی است که بیان می‌کند که طرح‌ها و یا سیستم‌ها باید تا حد امکان ساده باشند و تا جایی که امکان دارد، باید از پیچیدگی‌ها در یک سیستم پرهیز کرد. این اصل بر پایه این است که سادگی، بیشترین سطح پذیرش از سوی کاربر را دارد و امکان تعامل هرچه بهتر کاربر با سیستم را تضمین می‌کند. این اصطلاح برای اولین بار در نیروی دریایی ایالات متحده توسط یک مهندس استفاده شد و بر پایه این فرضیه مطرح شد که طرح‌ها باید به اندازه‌ای ساده باشند که توسط یک شخص در شرایط جنگی، تنها با برخی آموزش‌های اولیه، قابل درک باشد.

DRY (Don’t Repeat Yourself)

این اصل بیان می‌کند که تکرار در منطق برنامه باید از طریق Abstraction، و تکرار در فرآیندها باید از طریق خودکارسازی حذف شود. این اصل بر این اساس است که افزودن کد تکراری، میزان کار مورد نیاز برای گسترش و نگهداری (Maintenance) نرم‌افزار را در آینده افزایش می‌دهد. همچنین، این اصل بیان می‌کند که اگر یک فرآیند قابل خودکارسازی باشد (مانند آزمون نرم‌افزار) ولی به صورت خودکار انجام نشود، به هدر رفتن وقت منجر می‌شود و ممکن است به همین دلیل توسط برنامه‌نویسان به صورت منظم انجام نشود.

YAGNI (You Aren’t Gonna Need It)

این اصل بیان می‌کند که فقط زمانی باید یک Feature به یک نرم‌افزار اضافه شود که مورد نیاز باشد. به عبارت دیگر، اضافه کردن فیچرها و کدهای اضافی با فرض اینکه در آینده مورد استفاده قرار می‌گیرند، باعث اضافه شدن پیچیدگی غیرضروری به نرم‌افزار می‌شود. این اصل بخشی از فلسفه Extreme Programming است.

SOLID

SOLID مخفف پنج اصل طراحی در برنامه‌نویسی شی‌گرا می‌باشد.

- Single Responsibility Principle: هر موجودیت در نرم‌افزار (کلاس، تابع و...) باید تنها یک مسئولیت داشته باشد.
- Open-Closed Principle: هر موجودیت در نرم‌افزار باید برای توسعه باز باشد، اما برای اصلاح بسته باشد. به عبارت دیگر، این اصل بیان می‌کند که کد خود را طوری بنویسید تا بتوانید بدون تغییر کد موجود،

فیچر جدیدی اضافه کنید. این اصل از موقعیت‌هایی جلوگیری می‌کند که در آن تغییر یکی از کلاس‌های شما، نیازمند تطبیق کلاس‌های دیگری است که به کلاس تغییر داده شده، وابسته هستند.

- Liskov Substitution Principle: وقتی از یک شی از یک Subclass، به جای یک شی از Superclass آن استفاده می‌کنید، همه چیز همچنان باید به درستی کار کند. به عبارت دیگر، اشیاء Subclass باید دقیقا مانند اشیاء Superclass رفتار کنند.
- Interface Segregation Principle: کاربران نباید مجبور شوند به Interface-هایی که استفاده نمی‌کنند، وابسته باشند. این اصل موجب کاهش Coupling می‌شود.
- Dependency Inversion Principle: ماژول‌های سطح بالا که منطق پیچیده‌ای را ارائه می‌دهند، باید به راحتی قابل استفاده مجدد باشند و تحت تأثیر تغییرات در ماژول‌های سطح پایین قرار نگیرند. برای رسیدن به این اصل باید با استفاده از Abstraction-ها ماژول‌های سطح بالا و پایین را از هم جدا کرد.

HTML

- استفاده درست از تگ‌های معنایی (Semantic Tags)، به بالا رفتن خوانایی کد و دسترسی پذیری سایت کمک می‌کند. به عنوان مثال، برای تیترهای صفحه سلسله مراتب تگ‌های h1، h2 و... داشته باشیم و ترجیحا یک تگ h1 بیشتر نداشته باشیم. یا مثلا از تگ header، main و footer برای بخش‌های صفحه به جای div استفاده کنیم. این کار به بالا رفتن SEO سایت و تشخیص محتوای صفحه کمک می‌کند.
- توصیه می‌شود همیشه برای تگ‌های img از اتریبیوت alt جهت توصیف آن استفاده کنید. با این کار در صورت بروز مشکل در لود شدن عکس، توصیفی از آن در دسترس است و همچنین دسترسی پذیری صفحه را برای کاربران screen reader افزایش می‌دهد.
- از bad practice-های توسعه صفحات وب، استفاده از inline styles در اسناد HTML است. تا جای ممکن separation of concern را در توسعه هر صفحه وب لحاظ کنید.
- در HTML نیازی به استفاده از closing tag برای void element-ها نیست. پس تگ‌ها به دو صورت زیر خواهند بود:

`<p>...</p>` | `<img ...>` (Not `<img ... />`)

- حتما HTML خود را در این لینک جهت انجام بررسی‌ها وارد کنید. همچنین سعی کنید تا جای ممکن از nested شدن های اضافی جلوگیری کنید و ساختار را به صورت minimal-ترین حالت ممکن در آورید.

CSS

- سعی کنید استایل‌های مشترک (global) و محلی (local) را در فایل‌های جدا قرار دهید.
- خوب است که ابتدا صفحه HTML خود را کامل پیاده‌سازی کرده و سپس به آن استایل‌ها را اضافه کنید.
- فایل‌های CSS حاصل خیز هستند و در طول توسعه حجم آنها بیشتر می‌شود. تا حد ممکن از زدن کدهای تکراری و استفاده نشده پرهیز کنید و اصول clean code را در آن رعایت کنید.

- همیشه تلاش کنید که CSS-های استفاده شده در minimal-ترین حالت ممکن باشند و هیچ property-ای بی دلیل گذاشته نشده است. همچنین سعی کنید تا جای ممکن از کلاس‌های Bootstrap استفاده کرده و فقط در صورت نیاز CSS بنویسید.
- بهتر است پالت‌های رنگ، مقادیر border-radius و... را به صورت متغیرهای CSS در فایل global نگهداری کنید تا در صورت تغییر یک متغیر، نیاز به تغییر آن مقدار در همه صفحات نباشد.
- استفاده از !important به معنای عدم رعایت ترتیب اثرگذاری کلاس‌های تعریف شده استایلی است و باید تا حد امکان از این کلیدواژه استفاده نکنید.
- با معرفی نسخه‌ها و کلیدواژه‌های جدید در CSS، خوب است که میزان پشتیبانی از استایل‌های نوشته شده خود را در مرورگرهای مختلف بررسی کنید. برای این کار می‌توانید از CanIUse یا از داک MDN استفاده کنید.
- در نهایت، همانند HTML، درستی فایل‌های CSS خود را نیز توسط این لینک بررسی کنید.

Git Commits

همانطور که در پروژه اول توضیح داده شد، کامیت‌ها اهمیت زیادی در توسعه پروژه‌های نرم‌افزاری دارند. در این پروژه نیز باید مواردی که در پروژه اول گفته شدند، رعایت شوند.

نکات پایانی

- این تمرین در گروه‌های حداکثر دو نفره انجام می‌شود. برای تحویل آن کافی است که یکی از اعضای گروه، لینک مخزن گیت‌هاب و Hash مربوط به آخرین کامیت پروژه را در سایت درس آپلود کند. پروژه شما بر روی این کامیت مورد ارزیابی قرار می‌گیرد.
- حتما کاربر IE-S04 را به پروژه خود اضافه کنید.
- ساختار مناسب و تمیزی کد برنامه، بخشی از نمره همه پروژه‌های شما خواهد بود. بنابراین در طراحی ساختار برنامه و همچنین خوانایی کد دقت زیادی به خرج دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده شباهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاستی که در کلاس گفته شده است کسر خواهد شد.
- سوالات خود را تا حد ممکن در گروه درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آنها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این تمرین ارتباط برقرار کنید.