TEST PROJECT

Web Application Development (Online)

2022 BRICS Skills Competition

(BRICS Future Skills Challenge)

Contents

Introduction

Sunshop is an online mall in China that focuses on the sales of digital products. They currently need to re-develop the existing online mall system to accommodate customers using different terminal devices to access their mall for browsing and shopping.

Sunshop specifies that you complete the API development according to the RESTfull specification, and write a Vue application to complete the functions of the online store, including the store management page for employees. The customer-oriented store pages (including browsing, shopping, and order pages) need to support different types of Device resolution.

Description of project and tasks

The project is divided into two phases：

- Phase one: You need to complete the development of the database and back-end RESTfull API on day1 (2.5 hours in the morning and 2.5 hours in the afternoon), including the background management API for employees and the browsing and purchasing API for customers.
- Phase two: You need to complete the development of Vue applications and pages on day2 (2.5 hours in the morning and 2.5 hours in the afternoon), including the background management pages of employees and the browsing and shopping pages of customers. Your Vue application needs to interface with the RESTfull API you completed in phase one. You can make changes to your first-day RESTfull API without affecting your results in phase one.

The projects you have completed need to be deployed to the server respectively, and only the content of the server will be judged when scoring:

- Phase one: The first-phase works will be recycled after the end of the afternoon of day1. The API completed by your Phase one should be accessed through the URL of http://<hostname>/API, and subsequent API development is based on this URL.

- Phase two: The second stage works will be recycled after the end of the afternoon of day2, and the Vue application completed by phase two can be accessed through the URL of http://<hostname>/VUE.
- Note: The <hostname> in the URL will be provided separately using a separate file.

Phase one: RESTfull API

In this stage, you need to complete function development and database design according to the specific requirements of the provided RESTfull API. All API specific requirements have detailed requirements.

In addition to this document, you will be provided with additional documents to describe the specific details of the API requirements, these documents include:

- API documentation file (html)
- Postman file (json)

JSON data structure

The data returned by the API in the system should comply with the unified JSON structure, and all data needs to be encapsulated in the data field of the structure in the form of objects and returned. As shown in the following example:

JSON Data Structure

Status: 200
Content-Type: application/json
Body:

```
{
   "status":200,
   "success":true,
   "message":"register successful",
   "data": {
        "nickname": "Jones"
   },
   "time":"2021-08-23 14:50:28"
}
```

status - int，HTTP status code
success - boolean, true when the API handles results, false on errors or failures
message - string，message
data - object | null, the actual data, defined according to the API requirements
time - datetime，response time

Error handling

A standardized error handling mechanism needs to be defined in the API. Error prompt data such as 404 errors, validation errors, and authentication errors should also comply with the requirements of the JSON data structure above. As shown in the following example:

- 404 error example: when the requested address or resource does not exist

Not found

Status: 404
Content-Type: application/json
Body:

```
 {
    "status": 404,
    "success": false,
    "message": "Not found",
    "time":"2022-05-23 14:41:53"
}
```

status - int
success - boolean, This field must be false on API errors
message - string
time - datetime

- Validation error example: when the request data does not meet the validation requirements of the API interface

Validation error

Status: 422
Content-Type: application/json
Body:

```
{
```

```
    "status": 422,
    "success": false,
    "message": "Validation error",
    "time":"2022-05-23 14:41:53"
}
```

- Example of authentication error: When a valid and correct token is not provided when requesting an API that requires authentication

Unauthorized

Status: 401
Content-Type: application/json
Body:

```
{
    "status": 401,
    "success": false,
    "message": "Unauthorized",
    "time":"2022-05-23 14:41:53"
}
```

Information security

The API requires some security restrictions to prevent the API from being abused by hackers and malicious scripts. Your security measures include at least the following forms:

- For the same IP address, the number of visits to the same API per unit time (60 seconds) is limited to 30 times.
  - If the limit is exceeded, an error message that meets the requirements of the <Error Handling> paragraph shall be returned, and its status code shall be HTTP 429.

User Login

Requests are used for user login or authentication.

If the client sends data:

- Username and password are correct: HTTP 200
- Username or password error: HTTP 401, with a reasonable prompt message.

In subsequent APIs that require authentication, you need to carry the user token in the HTTP header.

User Register

The request is used to register a new user.

There should be two types of users in the system:

- user: Ordinary users can place orders and settle in the mall
- staff: Employee users can enter the management background for management

If the client sends data:

- Successful registration: HTTP 200
- Incorrect data or validation failure: HTTP 422, with a reasonable prompt message.

User Logout

The request is used to log out the current user.

If the client sends data:

- Successful logout: HTTP 200
- Invalid user token: HTTP 401, with a reasonable prompt message.

User Session

The request is used to get the current user information.

If the client sends data:

- Successful logout: HTTP 200
- Invalid user token: HTTP 401, with a reasonable prompt message.

List User (All)

[*Login required] [*Staff only] The request is used to get a list of all users.

If the client sends data:

- Successful operation: HTTP 200
- Invalid user token, incorrect permissions: HTTP 401, with a reasonable

prompt message.

- Request data error: HTTP 422, with a reasonable prompt message.

List User (Paged)

[*Login required] [*Staff only] The request is used to get a list of users.

If the client sends data:

- Successful operation: HTTP 200
- Invalid user token, incorrect permissions: HTTP 401, with a reasonable prompt message.
- Request data error: HTTP 422, with a reasonable prompt message.

Get User

[*Login required] [*Staff only] The request is used to get a user.

If the client sends data:

- Successful operation: HTTP 200
- user token is invalid, permission is incorrect: HTTP 401, with a reasonable prompt message.
- User not found error: HTTP 404, with a reasonable prompt message.

Delete User

[*Login required] [*Staff only] The request is used to delete a user.

If the client sends data:

- Successful operation: HTTP 200
- user token is invalid, permission is incorrect: HTTP 401, with a reasonable prompt message.
- User not found error: HTTP 404, with a reasonable prompt message.

Modify User

[*Login required] [*Staff only] The request is used to modify a user's nickname and permissions.

If the client sends data:

- Successful operation: HTTP 200
- Invalid user token, incorrect permissions: HTTP 401, with a reasonable prompt message.
- User not found error: HTTP 404, with a reasonable prompt message.

Upload Image

[*Login required] [*Staff only] This request can upload an image in Base64 encoding format and return image information. You need to pass an object with the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object for uploading image information
- Incorrect data or validation failure: HTTP 422, return validation error, and there should be a reasonable prompt

Delete Image

[*Login required] [*Staff only] This request can delete an image, you must pass the deleted image id in the URL and the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object for the deleted image information
- Image not found error: HTTP 404, 404 error returned

List Image (All)

[*Login required] [*Staff only] All images can be fetched with this request. You must carry the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays all data
- Invalid user token, incorrect permissions: HTTP 401, return authentication error

- Request data error: HTTP 422, return validation error

List Image (Paged)

[*Login required] [*Staff only] All images can be fetched with this request. You must carry the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays all data
- Invalid user token, incorrect permissions: HTTP 401, return authentication error
- Request data error: HTTP 422, return validation error

Create Banner

[*Login required] [*Staff only] This request can create a banner image. You have to pass an object with the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays banner image information
- Invalid user token: HTTP 401, return authentication error
- Incorrect data or validation failure: HTTP 422, return validation error, and there should be a reasonable prompt

Delete Banner

[*Login required] [*Staff only] This request can delete the banner image. You have to pass the removed banner id in the URL and the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays the deleted banner image information
- Invalid user token, incorrect permissions: HTTP 401, return authentication error

- Incorrect data or validation failure: HTTP 422, return validation error, and there should be a reasonable prompt
- Banner not found error: HTTP 404, 404 error returned

List Banner (All)

This request can get all banner images.

If the client sends data:

- Successful operation: HTTP 200, returns an object showing all data

List Banner (Paged)

This request can get a list of banner images.

If the client sends data:

- Successful operation: HTTP 200, returns an object showing all data
- Request data error: HTTP 422, returned validation error

Create Category

[*Login required] [*Staff only] This request can create an item category. You must carry the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, return product category information
- Invalid user token, incorrect permissions: HTTP 401, return authentication error
- Incorrect data or validation failure: HTTP 422, return validation error, and there should be a reasonable prompt

List Category (All)

This request can get all product categories.

If the client sends data:

- Successful operation: HTTP 200, return to product category list

List Category (Paged)

This request can get all product categories.

If the client sends data:

- Successful operation: HTTP 200, return to product category list

Delete Category

[*Login required] [*Staff only] With this request, the product category can be deleted. You have to pass the removed category id in the URL and the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, return the deleted product category
- Invalid user token, permission error: HTTP 401, return authentication error
- Category not found error: HTTP 404, 404 error returned
- There are subcategories or products under the deleted product category, and deletion is prohibited: HTTP 422, return 422 error

List Recharge (All)

[*Login required] [*Staff only] This request can get all recharge records. You must carry the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object showing all data
- Invalid user token: HTTP 401, return authentication error

List Recharge (Paged)

[*Login required] [*Staff only] This request can get all recharge records. You must carry the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object showing all data
- Invalid user token: HTTP 401, return authentication error

Create Recharge

[*Login required] [*Staff only] This request can recharge the user. You have to pass an object with the user token in the header.

If the client sends data:

- Successful operation: HTTP 200, returns an object for recharge user information
- Invalid user token, permission error: HTTP 401, return authentication error
- Incorrect data or validation failure: HTTP 422, return validation error, and there should be a reasonable prompt

Create Product

[*Login required] [*Staff only] This request can create an item. You have to pass an object with the user token in the header.

If the client sends data:

- Successful operation:HTTP 200, the returned content is empty
- Invalid user token, permission error: HTTP 401, return authentication error
- Incorrect data or validation failure: HTTP 422, return validation error, and there should be a reasonable prompt

List Product (All)

This request can get all product.

If the client sends data:

- Successful operation: HTTP 200, returns an object for all product

List Product (Paged)

This request can get all product.

If the client sends data:

- Successful operation: HTTP 200, returns an object for all product

Get Product

Through this request, the user can obtain the details of a product. You can pass an empty object, pass the product ID in the URL.

If the client sends data:

- Successful operation: HTTP 200, returns an object for product details
- Product not found error: HTTP 404, 404 error returned

Modify Product

[*Login required] [*Staff only] This request can modify the product information.

If the client sends data:

- Modified successfully: HTTP 200, return an object for the modified information.
- Invalid user token, permission error: HTTP 401, return authentication error
- Product not found error: HTTP 404, 404 error returned

Delete Product

[*Login required] [*Staff only] This request can delete an item. You must use soft delete to delete it.

If the client sends data:

- Successful operation: HTTP 200, return the deleted product information
- Invalid user token, permission error: HTTP 401, return authentication error
- Product not found error: HTTP 404, 404 error returned

Create Order

[*Login required] This request can create a product order, which only allows users to place an order while browsing in the foreground.

If the client sends data:

- Successful operation:HTTP 200, the returned content is empty
- Invalid user token, permission error: HTTP 401, return authentication error
- Incorrect data or validation failure: HTTP 422, return validation error, and there should be a reasonable prompt

List Order (All)

[*Login required] With this request, the user can get all orders.

Users can only read their own orders, while staff can read all orders.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays all data

List Order (Paged)

[*Login required] [*Staff only] With this request, the user can get a list of orders.

Users can only read their own orders, while staff can read all orders.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays all data

Modify Order

[*Login required] [*Staff only] With this request, order information can be modified.

If the client sends data:

- Successful operation:HTTP 200, the returned content is empty
- Invalid user token, permission error: HTTP 401, return authentication error
- Order not found error: HTTP 404, 404 error returned

Get Order

[*Login required] Through this request, you can get the order information, you can pass an empty object, pass the id of the order in the URL, and carry the user token in the header. Users can only read their own orders, while staff can read all orders.

If the client sends data:

- Successful operation: HTTP 200, returns an object for an order information.
- Invalid user token, permission error: HTTP 401, return authentication error
- Order not found error: HTTP 404, 404 error returned

Pay Order

[*Login required] [*User only] With this request, the order can be paid, you can pass an empty object, the id of the order in the URL. Only users can pay for orders with their own account.

If the client sends data:

- Successful operation:HTTP 200, the returned content is empty
- Invalid user token, permission error: HTTP 401, return authentication error
- Order not found error: HTTP 404, 404 error returned

List Address (All)

[*Login required] [*User only] This request can get all addresses, you must carry the user token in the header. Only users can operate on their own address information.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays all data

List Address (Paged)

[*Login required] [*User only] This request can get a list of addresses, you must send an object with the user token in the header. Only users can operate on their own address information.

If the client sends data:

- Successful operation: HTTP 200, returns an object that displays all data

Create Address

[*Login required] [*User only] With this request, you can add address information, you must send an object and carry the user token in the header. Only users can operate on their own address information.

If the client sends data:

- Successful operation: HTTP 200, returns an object containing the add address information
- Invalid user token, permission error: HTTP 401, return authentication error

Modify Address

[*Login required] [*User only] This request modifies an address, you need to pass an object, pass the modified address id in the URL, and carry the user token in the header. Only users can operate on their own address information.

If the client sends data:

- Successful operation: HTTP 200, returns an object containing the modified address information
- Invalid user token, permission error: HTTP 401, return authentication error
- Address not found error: HTTP 404, 404 error returned

Delete Address

[*Login required] [*User only] This request can delete the address. You have to pass the removed address id in the URL and the user token in the header. Only users can operate on their own address information.

If the client sends data:

- Successful operation: HTTP 200, returns an object containing the deleted address information
- Invalid user token, permission error: HTTP 401, return authentication error
- Address not found error: HTTP 404, 404 error returned

Database

You are provided with a ready-made database with some initial data. You need to modify the data table structure or create a new data table according to business logic.

Phase two: Vue Application

In this stage, you need to use the Vue framework to complete the online shopping mall application according to the provided page template file. The completed application needs to connect to the API interface completed by your Phase one and follow the business logic specified in Phase one.

You can make some necessary changes to your API in order for you to complete this stage of development. At this stage your tasks are divided into two categories, the customer-facing S*unshop* Page and the employee-facing *Sunshop* Management Page.

Your completed Vue app functionality needs to conform to the following principles:

- The functional logic follows the business logic specified in <Phase one>.
- The function logic and data match the provided page template.
- There are good error prompts and interactive effects for user misoperation and API error reporting, and any error will not cause the single-page application to fail to run normally.

Sunshop Page

The Sunshop Page is mainly for browsing, shopping and other functions of ordinary customers. Customers may access the online store through different terminal devices. You need to perform some responsive optimization on the provided page template, and support at least two device resolutions: 1920px (PC) , 480px (Mobile).

For pages that do not provide templates, you need to complete the page layout by yourself, and the layout style needs to be consistent with other template pages.

Pages and features include but are not limited to:

- Home page
  - Category menu of products
  - Ability to jump to user center page, shopping cart page

- Banner image with carousel
- List of popular categories and their main products
- List of popular products
- Management Page entrance, available only for staff

- Shopping cart page
  - Add, delete and modify items in the shopping cart
  - Select the product to jump to the order page [staff unavailable]
- Order page [staff unavailable]
  - Select and edit shipping address
  - List the product information, order price and other information in the order
  - Jump to the payment page after submitting the order
- Payment page [staff unavailable]
  - Display submitted order information
  - Confirm payment for the order
- Product list page
- Product detail page
  - Show product details
- User center page [staff unavailable]
  - User information - including username, balance, etc.
  - Enter the pages of My Favorites, Order List, Recharge Record, Address Management, etc.
- My Favorites page [staff unavailable]
  - List favorites
  - Unfavorite
- Order list page [staff unavailable]
  - List orders and order basic information
  - Enter the order details page
- Order details page [staff unavailable]
  - List order details

- If you place an unpaid order, you can jump to the payment page to re-pay

- Recharge record page [staff unavailable]
  - List recharge records
- Address management page [staff unavailable]
  - List address information
  - Add, delete and modify address information

Sunshop Management Page

The Sunshop Management Page is used for employees to perform various management functions on the mall. You need to complete the development of the store management functions according to the template page provided. For pages that do not provide templates, you need to complete the page layout by yourself, and the layout style needs to be consistent with other template pages.

Pages and features include but are not limited to:

- User Management
  - List page - including delete, modify, recharge and other entrances
  - Modify page - can be a pop-up modal box
  - Recharge page - can be a pop-up modal box
- Category Management
  - List page - including delete, modify, create and other entries
  - Modify page - can be a pop-up modal box
  - Create page - can be a pop-up modal box
- Product Management
  - List page - including delete, modify, create and other entries
  - Modify page - can be a pop-up modal box
  - Create page - can be a pop-up modal box
- Order Management
  - List page - including modification status, modification amount and other entries

- Modify status page - can be a pop-up modal box
- Modify amount page - can be a pop-up modal box

- Recharge Management
  - List page
  - Recharge page - can be a pop-up modal box

Module Distinction

During the competition, the content of different stages will be distinguished according to the module, and your work should be completed within the time specified in the module:

Module A (Sunshop Page API Development)

- User
  - User Login
  - User Register
  - User Logout
  - User Session
- Category
  - List Category (All)
  - List Category (Paged)
- Recharge
  - List Recharge (All)
  - List Recharge (Paged)
- Product
  - List Product (All)
  - List Product (Paged)
  - Get Product
- Order
  - Create Order
  - List Order (All)
  - List Order (Paged)
  - Get Order

- Pay Order
- Address
  - List Address (All)
  - List Address (Paged)
  - Create Address
  - Modify Address
  - Delete Address

Module B (Sunshop Management Page API Development)

- User
  - List User (All)
  - List User (Paged)
  - Get User
  - Delete User
  - Modify User
- Image
  - Upload Image
  - Delete Image
  - List Image (All)
  - List Image (Paged)
- Banner
  - Create Banner
  - Delete Banner
  - List Banner (All)
  - List Banner (Paged)
- Category
  - Create Category
  - Delete Category
- Recharge
  - Create Recharge
- Product
  - Create Product

- Modify Product
- Delete Product

- Order
  - Modify Order

Module C (Sunshop Page Development)

- Sunshop Page
  - Home Page
  - Shopping cart page
  - Order page
  - Payment page
  - Product list page
  - Product detail page
  - User center page
  - My Favorites page
  - Order list page
  - Order details page
  - Recharge record page
  - Address Management page

Module D (Sunshop Management Page Development)

- Sunshop Management Page
  - User Management
  - Category Management
  - Product Management
  - Order Management
  - Recharge Management

Note: Some API interfaces of module A may be related to the Sunshop Management page!

Submission Instructions to the Competitor

For server content scoring only, your work should be deployed to the server as follows:

- Phase one - RESTfull API

- In your deployed application and database, your application's initial administrator account and password must be defined and guaranteed to be available.
  - UserName：admin
  - Password：sunshop
- The API should be deployed to the URL: http://<hostname>/API
- Your designed database structure and data (including fully operational data) should be exported to the URL as db-dump.sql filename: http://<hostname>/db-dump

- Phase two - Vue Application
  - Vue applications should be deployed to the URL: http://<hostname>/VUE
- Note: The <hostname> in the URL will be provided separately using a separate file.