پروژه درس مهندسی اینترنت

فاز دوم – پیاده‌سازی Back-end

ترم دوم

سال تحصیلی ۹۶-۹۷

با توجه به مفاهیم و فناوری‌هایی که در طول کلاس درس مهندسی اینترنت باید در این پروژه به پیاده‌سازی سمت سرور فروشگاه لباس خود خواهید پرداخت.

تعریف پروژه

در این قسمت پروژه شما باید کد مربوط به سرور و دیتابیس فروشگاه را پیاده‌سازی کنید. کار شما با طراحی و پیاده‌سازی دیتابیس مورد استفاده توسط پروژه شروع می‌شود. شما سپس می‌بایست با استفاده از یکی زبان‌های PHP و یا Node.js، کد مربوط به برنامه‌ی سرور را نوشته، آن را به دیتابیس وصل کنید و سپس داده‌ها را به طریقی در اختیار کد سمت کاربر (Front-end) قرار دهید. شما می‌توانید این پیاده‌سازی را با استفاده‌ی خام از این زبان‌ها و یا **هر فریم‌ورکی از این زبان‌ها** می‌توانید انجام دهید.

باید به این نکته توجه داشته باشید که پیاده‌سازی Back-end باید در قالب یک API باشد؛ به این معنی که در نهایت باید سرویس‌های پروژه را به شکل مجموعه‌ای از endpoint ها به کاربر ارائه کنید. برای مثال فرض کنید می‌خواهید سرویسی را پیاده‌سازی کنید که لیست محصولات را به شما برگرداند. شما در زبان منتخب خود کدی را می‌زنید و سرور خود را روی localhost اجرا می‌کنید. حال باید بتوانید با زدن یک درخواست HTTP از نوع GET به آدرسی خاص (برای مثال localhost/products) لیستی از محصولات را به صورت JSON به شما ارائه دهد. تمامی سرویس‌های خود را باید در چنین قالبی پیاده‌سازی کنید.

در ادامه ابتدا ساختار و فیلدهای موجود در داده‌های پروژه را به شما ارائه می‌کنیم و سپس به توصیف هریک از سرویس‌ها در قالب مجموعه‌ای از endpoint ها می‌پردازیم.

ساختار داده‌ها

شما می‌توانید از هر تکنولوژی‌ای برای برپاسازی دیتابیس خود استفاده کنید، تنها نکته‌ای که الزام به رعایت آن دارید پیروی از ساختار داده‌های توصیف شده است.

در این پروژه چندین نوع داده وجود دارد که هرکدام می‌توانند در قالب یک جدول یا Schema پیاده‌سازی شوند. این جداول عبارتند از:

- محصول: هریک از محصولات موجود در سایت از نوع هستند.
- دسته‌بندی: هریک از دسته‌بندی‌های مردانه، زنانه و بچه‌گانه.
- آدرس
- کاربر
- سفارش

در ادامه به توصیف این schema ها می‌پردازیم.

محصول

هر محصول دارای یک شناسه‌ی منحصر به فرد می‌باشد که به وسیله‌ی آن شناسایی و جستجو می‌شود. محصولات دارای فیلدهای زیر نیز هستند.

```
Product = new schema ({
    id: String, //unique key
    name: String,
    sizes: Array, // Array of Strings eg: ['12','12.5'] or ['L', 'SM']
    colors: Array, // Array of Strings: ['red', 'blue']
    isAvailable: Boolean,
    price: Number,
    discount: Number,
    description: String,
    thumbnail: String, //this is the URL of small image
    images: Array, //An array of URLs, for each image
});
```

توضیحات مربوط به هر فیلد در قسمت روبروی آن ارائه شده است.

شما می‌توانید به هر محصول یک فیلد Categories نیز اختصاص دهید و از آن برای دسترسی آسان‌تر به دسته‌بندی هر لباس استفاده کنید. انتخاب نحوه‌ی ارتباط این جداول با یکدیگر بر عهده‌ی شماست.

دسته‌بندی

هر دسته‌بندی نیز دارای یک شناسه‌ی یکتا و نام است. هر دسته‌بندی می‌تواند مجموعه‌ای از دسته‌بندی‌های زیر مجموعه‌ی خود نیز داشته باشد. برای این‌کار می‌توانید آرایه‌ای از ID های این دسته‌بندی‌ها را در هر دسته‌بندی نگه‌داری کنید. کار دیگری که می‌توانید انجام دهید استفاده از فیلدی به نام parentCategory است که در آن ID دسته‌بندی والد این دسته در آن قرار می‌گیرد. نحوه‌ی ارتباط جداول با یکدیگر بر عهده‌ی شماست و بنابراین می‌توانید هر فیلد دیگری که لازم می‌دانید را به این schema اضافه کنید.

```
Category = new schema({

    id: String,
    name: String,
    subCategories: Array, // an array containing a list of category IDs
})
```

آدرس

```
address = new schema({
    id: String,
    addressText: String,
    city: String,
    lat: String, // latitude
    lon: String, // longitude
})
```

دو مشخصه‌ی lat و lon مشخص کننده‌ی طول و عرض جغرافیایی آدرس انتخاب شده اند.

کاربر

هر کاربر باید دارای مشخصات زیر باشد.

```
user = new schema({
    id: String,
    username: String,
    password: String, //encrypted
    avatarURL: String,
    credit: Number, // amount of user credit in Tomans
    addresses: Array(AddressSchema), // as described in Address section
    orders: Array(OrderSchema), //an array of order IDs referring to Order Schema
})
```

سفارش

هر سفارش دارای یک شناسه‌ی یکتا و لیستی از محصولات است، سفارش فیلدهای دیگری مانند آدرس و غیره نیز دارد که در زیر به آن اشاره شده است.

```
Order = new schema({
    id: String,
    totalPrice: Number,
    /*
        Objects of this array are in form of this:
        [
            {
                id: product.id,
                quantity: Number,
            }
        ]
    */
    products: Array,
    address: String, // it is the ID address of a user
    paymentType: String, // one of these:  'CREDIT', 'CASH' or 'ONLINE'
    status: String, // one of these: 'submitted', 'paid', 'sent', 'delivered'
```

```
    deliveryTime: String, // a UTC string
});
```

چیزی که این جدول را به سایر جداول متصل می‌کند، یکی آرایه‌ی محصولات و دیگری فیلد آدرس است. آرایه‌ی محصولات لیستی از ID محصولات به همراه تعداد هرکدام است. آدرس نیز دربردارنده‌ی ID آدرسی از کاربر است که برای ارسال انتخاب شده است.

پیاده‌سازی API

مجموعه endpoint های زیر باید برای پروژه پیاده‌سازی شوند. در زیر نوع هر endpoint به همراه توضیحی مختصر از کارایی آن آورده شده است.

صفحه‌ی خانه

```
/**
 * HTTP Method: GET
 * /homepage
 * @param {}
 */
function homepage() {
    return {
        banners: Array({
            bannerURL: String,
            bannerLink: String,
        }),
        aboutUsText: String,
        promotions: Array({
            text: String,
        })
    }
}
```

این تابع مجموعه‌ای از بنرهای مورد نمایش در صفحه‌ی خانه را نشان می‌دهد. از آن‌جایی که می‌خواهیم متن موجود در صفحه دینامیک باشد این متون را نیز باید به طریقی در خروجی این endpoint بازگردانیم.

دسته‌بندی‌ها

```
/**
 * HTTP GET
 * @returns Array(categories)
 */
function categories() {
    return {
        categories: Array(CategorySchema),
```

```
    }
}
```

لیستی از دسته‌بندی‌های موجود را برمی‌گرداند.

محصولات

در endpoint زیر، مجموعه‌ای از محصولات به کاربر برگردانده می‌شود. به عنوان پارامتر هم شماره‌ی صفحه، تعداد محصولات در هر صفحه و نیز آرایه‌ای از فیلترها فرستاده می‌شود. فیلترها شامل رنگ، برند و قیمت می‌باشند.

```
/**
 * HTTP GET
 * @param {Array} filters
 * @param {Number} page
 * @param {Number} perPage
 */
function products(filters, page, perPage) {
    return {
        count: Number,
        products: Array, // array of ProductSchema
    }
}
```

اندپوینت /product/{id} برای دریافت جزییات و مشخصات یک محصول استفاده می‌شود.

```
/**
 * HTTP GET
 * @param {String} id
 */
function product(id) {
    return {
        product: Object, // a ProductSchema object as a JSON
    }
}
```

برای دریافت لیست فیلترهای موجود نیز از اندپوینت /filters استفاده می‌شود.

```
/**
 * HTTP GET
 * /filters
 */
```

```
function filters() {
    return {
        filters: Array({
            id: String,
            name: String, // for example: قیمت
            values: Array, // possible values for the filter
            value: Number,// values could also be Number in case of price filter
        })
    }
}
```

ثبت سفارش

برای ثبت سفارش مجموعه‌ای از پارامترهای موردنیاز آن سفارش را به عنوان پارامتر به اندپوینت موردنظر داده و در صورت ثبت موفق، از طرف سرور جواب درست می‌گیرید. در اینجا باید این سفارش را در پایگاه داده‌ی خود ثبت کنید و سپس نتیجه‌ی عملیات را در قالب پارامترهایی معنی‌دار (staus, Order) به کاربر بازگردانید.

این اندپوینت از نوع POST است.

```
/**
 * HTTP POST
 * /order/submit
 * @param {Array} products
 * @param {Number} totalPrice
 * @param {String} address
 * @param {String} paymentType
 */
function order(products, totalPrice, address, paymentType) {
    return {
        status: Boolean, //indicates if the order was submitted or not
        order: Order, //returns the newly submitted order
    }
}
```

بروزرسانی وضعیت سفارش

باید اندپوینتی شبیه تابع بالا نیز برای بروزرسانی وضعیت سفارش پیاده‌سازی کنید. این اندپوینت توسط کاربر admin سایت و برای تغییر وضعیت سفارش استفاده خواهد شد. شکل کلی آن باید به صورت /order/{id}/update بوده و از نوع PUT یا POST باشد.

```
/**
 *
 * @param {String} id
 */
```

```
function order_post(id) {
    return {
        status: Boolean, // if successful true, else false,
        order: Order, // returns the updated order
    }
}
```

پیگیری سفارش

یک اندپوینت از نوع GET قرار دهید که به عنوان پارامتر، ID سفارش را گرفته و مشخصات و جزییات آن سفارش را به کاربر بازمیگرداند.

مدیریت کاربری

برای مدیریت کاربران وجود سه نوع اندپوینت اجباری است. یک اندپوینت از نوع GET برای دریافت مشخصات کاربر، یکی از نوع POST برای ثبت‌نام کاربر جدید و یکی هم از نوع PUT یا POST برای ویرایش مشخصات پروفایل یک کاربر. پیاده‌سازی حساب کاربری و جزییات آن با توجه به آموخته‌هایتان در درس مهندسی اینترنت بر عهده‌ی شماست.

مدیریت آدرس‌های کاربر

هر کاربری که وارد سیستم شده باشد باید بتواند آدرس‌های خود را مشاهده (GET user/addresses)، ویرایش (POST user/address/{id}) و پاک کند. برای هرکدام از این اعمال باید اندپوینت مناسب پیاده‌سازی شود. توجه داشته باشید که این اندپوینت‌ها باید فقط توسط کاربرانی که به سیستم وارد شده‌اند قابل دسترسی باشد.

بخش امتیازی

نظرات

پیاده‌سازی قابلیت ثبت نظر برای هر محصول، مشاهده نظرات و پاک کردن آن‌ها در سمت سرور و پایگاه داده امتیاز اضافی دارد – ۱۰ نمره (از ۱۰۰)

دسترسی‌های ادمین

در صورتی که بتوانید اندپوینت‌هایی را ایجاد کنید که با استفاده از آن admin سایت بتواند محصولات را مشاهده، ویرایش و پاک کند. سفارش‌ها را مشاهده و وضعیت آن‌ها را تغییر دهد نیز به شما نمره‌ی اضافی تعلق خواهد گرفت – ۴۰ نمره (از ۱۰۰)

موفق باشید