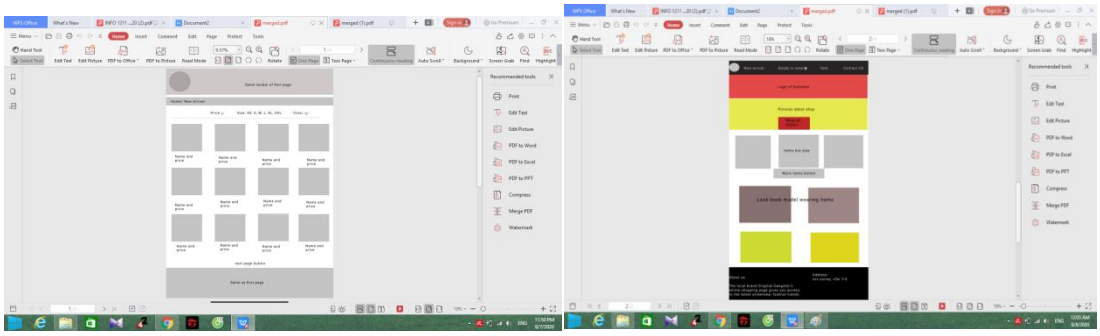
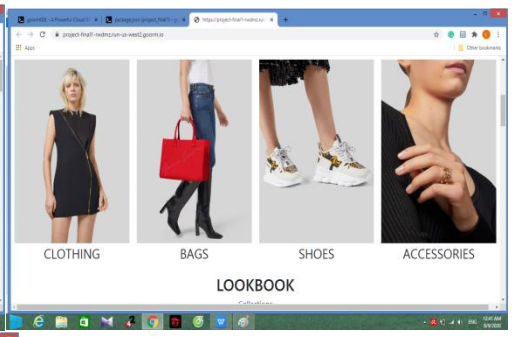
Description of my business:

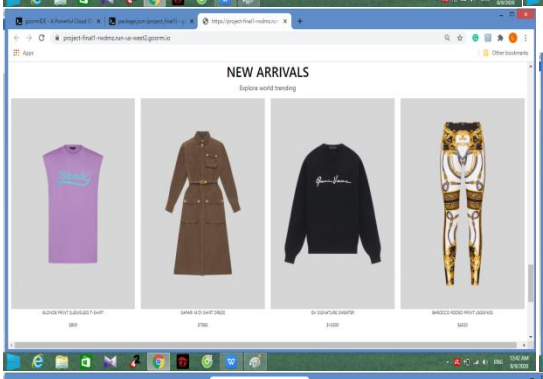
My business is providing for customers about fashion design, this is like a local brand that open for age 15 to 35, it like a street wear fashion. Business is called Original Gangster (OG), this name means about Underground. And all designs descriptions are information about introducing the quality characteristics, functions, benefits ... of a product posted on the sales website to give viewers full information about the product for them to have. Base selection when purchasing. The inspiration from our product samples comes from ordinary farmers, not too luxurious, suitable for them in all kinds of weather from men to women. These outfits are for them both fashionable and suitable for all people in our communities. The costs of items are also effort for all people even a kid want to try to be fashionista, it is not expensive or much more for all people. Our target is fashions for everyone. Moreover, our location is great and easy to find, including whether the community has a large enough population, whether its economy is stable enough for us to give customers best services.

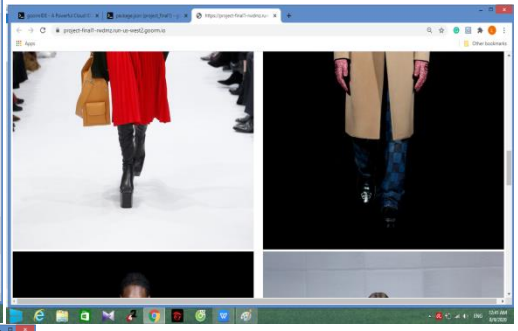
Compare Figma Wireframes with the Screenshots of your website

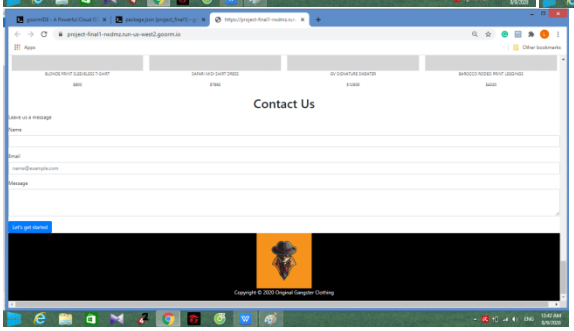


Home page:



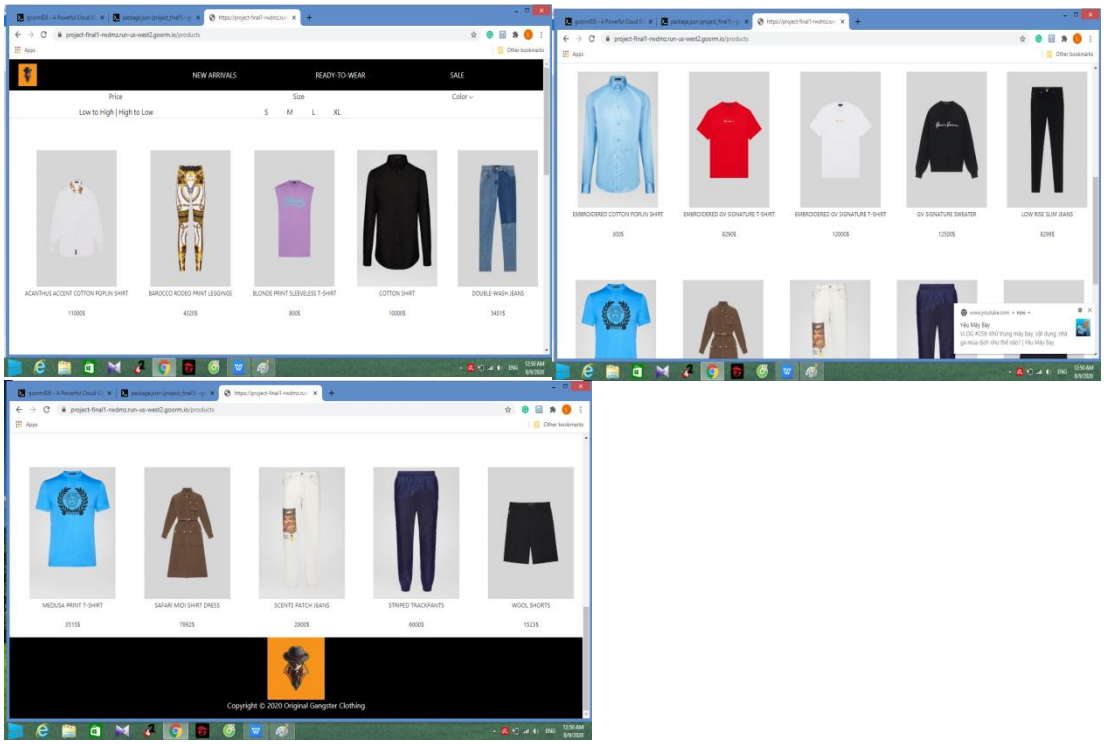






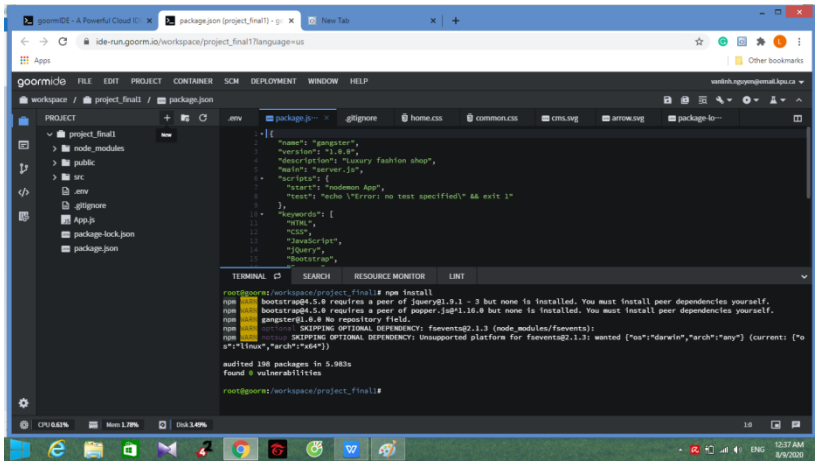
I add one more contact us in order to get information of customers and that have us can understand what customers want. I have not added information of shop in here because I have already told in description of business and that not necessary to add in website.

New arrival page:

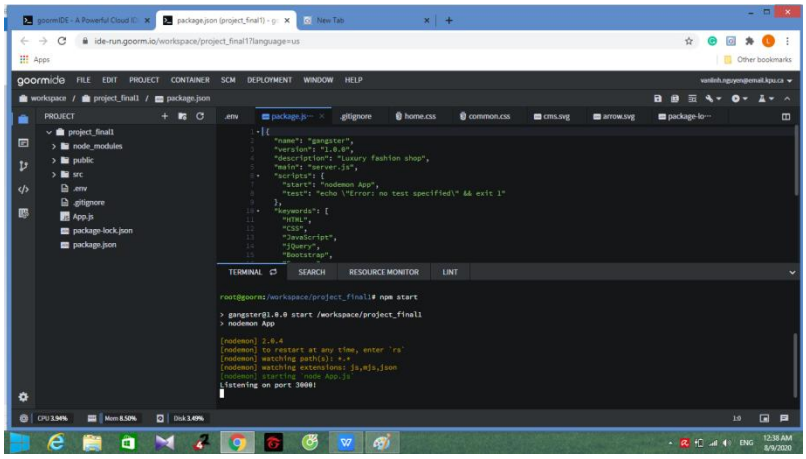


Instructions to run your project in GoormIDE

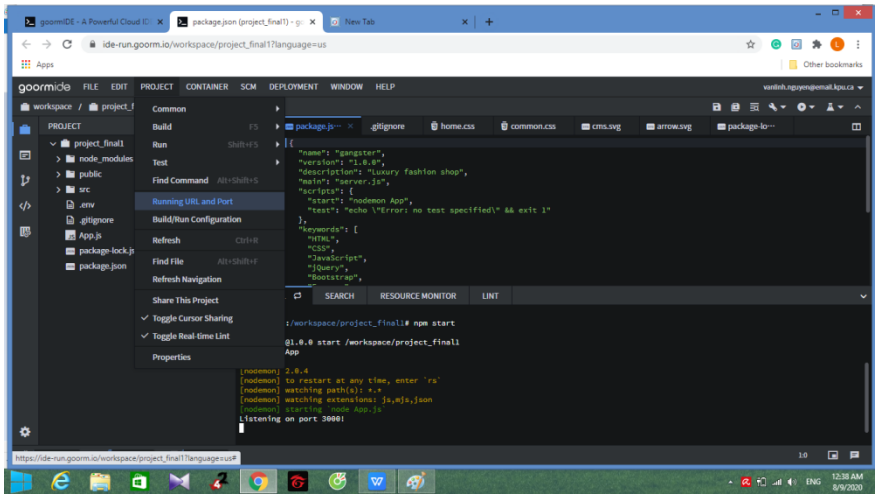
Step 1: run ‘ npm install’



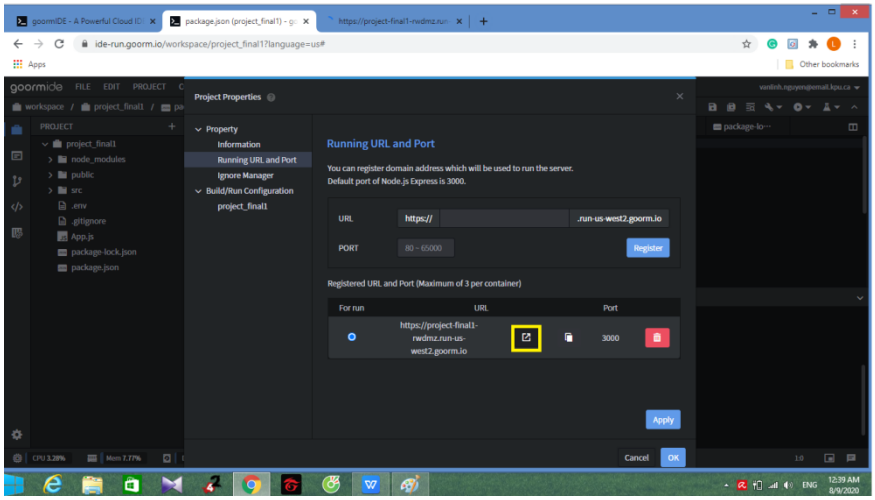
Step 2: run “npm start”



Step 3: Choose Project and Click Running as URL and Port



Step 4 : Click on the yellow icon to open website page



Feature/Content Explanations

Depending on Figma, I have used multiple things about front end and back-end. I mention in here that it not 100% correctly the same as Figma.

First, we need to run NPM INSTALL to get package.json and take dependencies from their to set up or contain information of my project. After when run npm install, it will have folder node_modules, and it will have Mongo or mongodb, database,…

- Create navbar and logo in html :

```
<div class="navigation-section">
        <div class="header-logo">
            <a href="/">
                <img src="img/logo.jpg" alt="Header logo">
            </a>
        </div>
```

- Have div to create New arrival, Ready to wear and Sale that all inside of nav bar:

```
<div class="menu">
                    <a href="#" onclick="handler('5f252a223422d5cd2a81e9cb',
event)">NEW ARRIVALS</a>
                </div>
                <div id="menu-dropdown">
                    READY-TO-WEAR
                    <img src="img/icons/arrow.svg" alt="Down arrow" id="arrow"
class="arrow">
                </div>
                <div class="menu">
                    SALE
                </div>
```

Also, set css. Category by display:flex that will be automatically width in nav bar for easy to look

- Create logo of business easy to look and find in home page, also that have button customers can click in order to view all product.

```
 <div class="banner">
        <img src="img/banner.png" alt="Banner" style="width: 100%" />
        <button type="button" class="btn btn-secondary btn-sm">All
products</button>
    </div>
```

- Create a div about Sale (Discover best-sellers of Ganster) also set css display:flex that automatically wide into page

```
 <div class="hot-sales">
        <h1 style="text-align: center;">SALE</h1>
        <p class="slogan">Discover best-sellers of Gangster</p>
        <div class="d-flex justify-content-around">
            <div class="product">
```

```
                    <img src="img/products/home-img/women-clothing.jpg"
alt="Women clothing" />
                    <p class="section-name">CLOTHING</p>
                </div>
                <div class="product">
                    <img src="img/products/home-img/women-bags.jpg"
alt="Women bags" />
                    <p class="section-name">BAGS</p>
                </div>
                <div class="product">
                    <img src="img/products/home-img/women-shoes.jpg"
alt="Women shoes" />
                    <p class="section-name">SHOES</p>
                </div>
                <div class="product">
                    <img src="img/products/home-img/women-accessories.jpg"
alt="Women accessories" />
                    <p class="section-name">ACCESSORIES</p>
                </div>
            </div>
        </div>
```

- Next, create a lookbook by using bootstrap flex (grid system):

```
<div class="lookbook">
            <h1 style="text-align: center;">LOOKBOOK</h1>
            <p class="slogan">Collections</p>
            <div class="row">
                <div class="col product">
                    <img src="img/products/home-img/look1.jpg" alt="Look 1" />
                </div>
                <div class="col product">
                    <img src="img/products/home-img/look2.jpg" alt="Look 2" />
                </div>
                <div class="w-100"></div>
                <div class="col product">
                    <img src="img/products/home-img/look3.jpg" alt="Look 3" />
                </div>
                <div class="col product">
                    <img src="img/products/home-img/look4.jpg" alt="Look 4" />
                </div>
            </div>
        </div>
```

- Create New arrival is the same as create Sale section:

```
<div class="new-arrivals-section">
            <h1>
                <a href="#" onclick="handler('5f252a223422d5cd2a81e9cb',
event)">NEW ARRIVALS</a>
            </h1>
```

```
            <p class="slogan">Explore world trending</p>
            <div class="d-flex justify-content-around">
                <div class="product">
                    <img src="img/products/home-img/sleeveless-tshirt.jpg"
alt="L'HOMME MOTIF T-SHIRT" />
                    <p class="name">BLONDE PRINT SLEEVELESS T-SHIRT</p>
                    <p class="price">$800</p>
                </div>
                <div class="product">
                    <img src="img/products/home-img/safari.jpg" alt="SAFARI MIDI
SHIRT DRESS" />
                    <p class="name">SAFARI MIDI SHIRT DRESS</p>
                    <p class="price">$7892</p>
                </div>
                <div class="product">
                    <img src="img/products/home-img/gv.jpg" alt="GV SIGNATURE
SWEATER" />
                    <p class="name">GV SIGNATURE SWEATER</p>
                    <p class="price">$12500</p>
                </div>
                <div class="product">
                    <img src="img/products/home-img/print-leggings.jpg"
alt="ACANTHUS PRINT ACCENT PANTS" />
                    <p class="name">BAROCCO RODEO PRINT LEGGINGS</p>
                    <p class="price">$4320</p>
                </div>
            </div>
```

- This is the link that sending a request to database and we do not need to reload page. And that same with function colorFilter(id, color) and function filterSize(id, size, event)

```
 function sortPrice(ord, event) {
                event.preventDefault();
                $(".product-list").children().remove();
                let id = JSON.parse(localStorage.getItem('results'))[0].category;
                $.ajax({
                    url: '/products/sort_by=price',
                    type: 'post',
                    data: { id: id, ord: ord }, (transfer to body )
                    dataType: 'json', ( get a result)
                    success: data => { ( if success it will accept respond from return
res.status(200).send(products) )
                        $.each(data, (index, product) => {
                            $product = $("<div>", { class: "product" }).append([
                                $("<img>", { src: `${product.photo}` }),
                                $("<p>", { text: `${product.name}` }),
                                $("<p>", { text: `${product.price}$` })
```

```
                                ]);
                                $(".product-list").append($product);
                        });
                }
        });
    }
```

- Schema is represent an object of database and is mapping of code and database. Also with productSchema that is the same as Mongodb.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const productSchema = new Schema({
    name: {
        type: String,
        required: true,
    },
    size: {
        type: String,
        required: true
    },
    color: {
        type: String
    },
    price: {
        type: Number,
        required: true
    },
    category: { type: Schema.Types.ObjectId, ref: 'Category' }
});

const Product = mongoose.model('Product', productSchema, 'Product');

module.exports = Product;
```

- * Import dependencies
  * 1. express: web application framework for Node.js
  * 2. bodyParser: extract the entire body portion of an incoming request stream and
  * exposes it on req.body
  * 3. dotenv: to use variables declared in .env file.
  * 4. mongoose: object modeling for Node.js.
  * For simplicity, it provides us the way to query against MongoDB
  * 5. Product, Category: MongoDB models, these represents schema of Collections in MongoDB.
  * We will use these models for mapping with Collections in MongoDB.
  * 6. app: use for routing.
-

```
const express = require('express');
```

```
let bodyParser = require('body-parser');
require('dotenv').config();
const mongoose = require('mongoose');
const path = require('path');
const Product = require('./src/server/models/product');
const app = express();
```

- app.use(bodyParser.urlencoded({ extended: true })) does the same for URL-encoded requests.

```
extended: true precises that the req.body object will contain values of any type
instead of just strings.
*/
app.use(bodyParser.urlencoded({ extended: true }));
/*
```

- To serve static files (images, CSS), we use express.static built-in middleware function.

```
*/
app.use('/css', express.static(path.join(__dirname, 'public/css')));
app.use('/img', express.static(path.join(__dirname, 'public/img')));
app.use('/css', express.static(path.join(__dirname,
'node_modules/bootstrap/dist/css')));
```

- Routing declaration section
  Routing refers to how an application's endpoints (URIs) respond to client requests.

```
* We use res.sendFile to render HTML file with express.
 * When error (err) occurs, it will popup error for us to aware of.
 */
app.get('/', (req, res) => {
      res.sendFile(path.join(__dirname, 'src/client/pages/home.html'), err => {
            if (err) {
                  res.status(err.status).end();
            }
      });
});
```

- Render product list page (this page is used for showing all products in NEW ARRIVALS page)

```
app.get('/products', (req, res) => {
      res.sendFile(path.join(__dirname, 'src/client/pages/list.html'), err => {
            if (err) {
                  res.status(err.status).end();
            }
      });
});
```

- Get all products and sort by name ascending with/without category

```
app.post('/products', async (req, res) => {
    let products = null;
    if (req.body.id) {
```

- This line is used to find any products which belong to category has id the same with id in body in request sent by client.

```
            sort({ 'name' : 1 }) is used to sort name ascending, if we want to sort descending,
            we use sort({ 'name': -1 }) or sort('-name').
        */
        products = await Product.find({
            'category': mongoose.Types.ObjectId(req.body.id)
        }).sort({ 'name': 1 });
    } else {
        products = await Product.find({}).sort({ 'name': 1 });
    }
    return res.status(200).send(products);
});
```

- Filter products by color

```
app.post('/products/filter_by=color', async (req, res) => {
    let criteria = {
        'category': mongoose.Types.ObjectId(req.body.id),
    };
      This line is used to find any products which has color the same with array of colors in body sent by request.
    if (req.body.color) {
        criteria.color = { $in: req.body.color };          ( in is a request: in ['Red','Black'];    AND )
    }
    const products = await Product.find(criteria);          (send request to it )
    return res.status(200).send(products);
});
```

- Sort products by price. Also can find by price and from by order low to high and high to low, a request send to database and appear on website

```
app.post('/products/sort_by=price', async (req, res) => {
    const products = await Product.find({
        'category': mongoose.Types.ObjectId(req.body.id)
    }).sort({ 'price': req.body.ord }).exec(function (err, results) {
        return res.status(200).send(results);
    });
});
```

- Filter products by size same with sort by price

```
app.post('/products/filter_by=size', async (req, res) => {
```

```
    const products = await Product.find({
        'category': mongoose.Types.ObjectId(req.body.id),
        'size': req.body.size
    })
    return res.status(200).send(products);
});
```

- Below connects to the database asynchronously, and
  once this is done you can start your Express application.
  params:
    - DATABASE_URL and PORT are environment variables declared in .env file.

```
mongoose
    .connect(process.env.DATABASE_URL, { useNewUrlParser: true,
useUnifiedTopology: true })
    .then(async () => {
        app.listen(process.env.PORT, () => {
            console.log(`Listening on port ${process.env.PORT}!`);
        })
    });
```