Clothing E-Commerce Website

Team Members Name: **Ritika Desai, Vedant Kansara**

Team Members Pitt Email: **rhd11@pitt.edu, vbk10@pitt.edu**

Course Name: **INFSCI 2710 – Database Management System**

Instructor: **Dmitriy Babichenko**

Date: **25th April, 2022**

Table of Contents

Introduction/Abstract

Project Description

We are designing an e-commerce website that offers men and women clothing. The first category of users would be customers. A customer can register themselves to purchase clothes. Once they place an order, they can login to their account and look at the status of the order. Once they receive their clothes, and for whatever reason they want to return it, they can do so by logging into their account. They can provide a reason for return if they wish to. They can also then check that status of the refund. Customers also have the option to leave a review for the products which includes a short description and a star rating (out of 5).
The other categories of users would be the sellers. They can create an account and register their company and sell clothes. A seller would ideally be able to sell more than one brand, but we will be sticking to 1 seller sells 1 brand. However, a seller could sell different clothes under a single brand.
The clothes are categorized into 8 categories: Tops, bottoms, dresses, and coats for women and activewear, shirts, pants, and shorts for men. For now, the website sells 8 products 1 under each category.

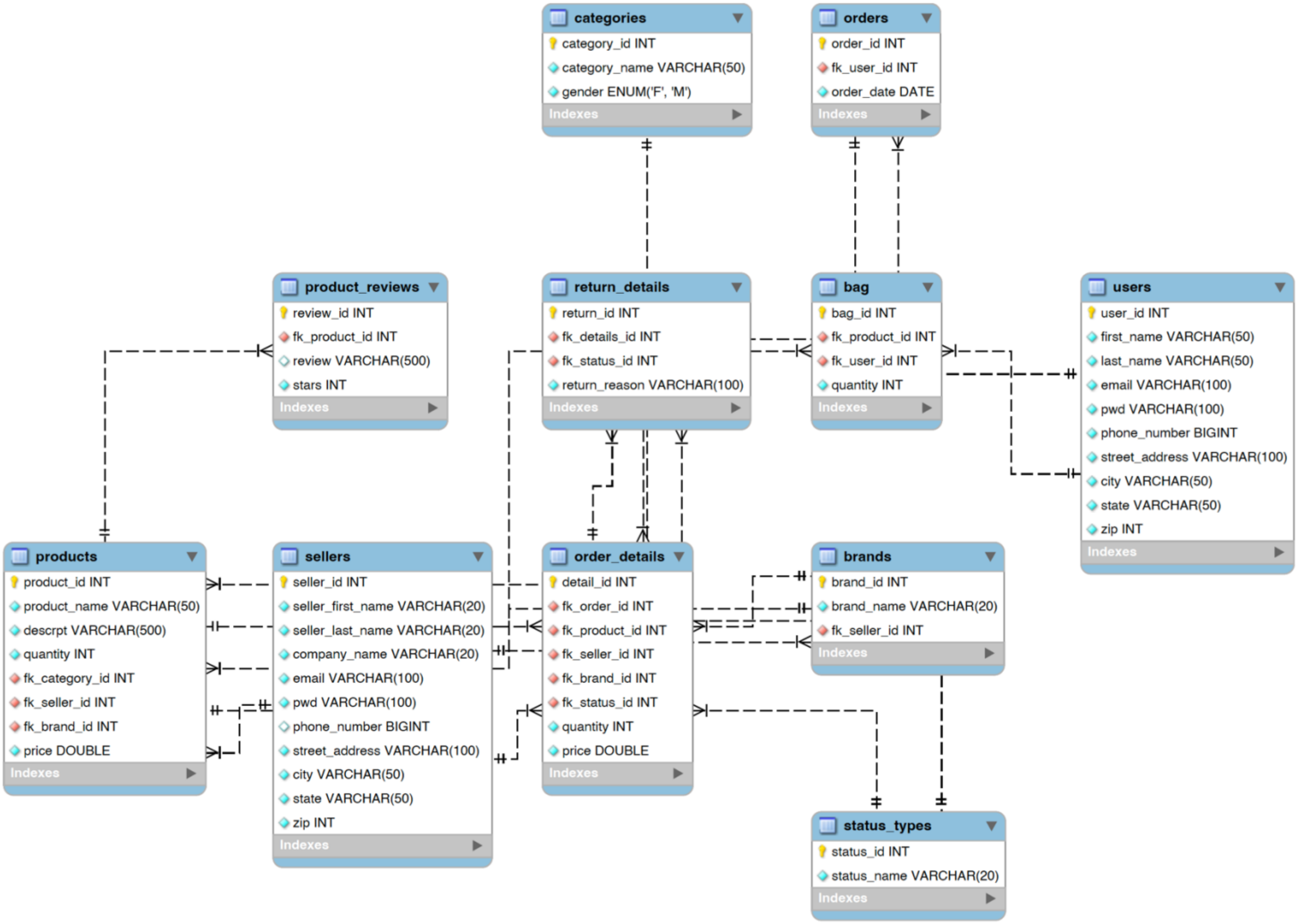
Target Audience

The website sells luxury clothing from high end brands like Louis Vuitton, Chanel, Gucci, Dior, and Armani. So, the target audience would be people who can afford these brands. We also make sure the sellers are registered sellers that are genuinely allowed to sell these brands and are selling the legitimate items.

Purpose and benefits of your database

Our database is mainly used for storing all the information related to customers, sellers, brands, orders, returns, reviews. The database helped to retrieve and store data in a place that available for immediate use. The frontend was populated by the database and any changes input on the frontend was stored in the database. Benefits of the database are the ability to keep all the data at one place as it would not be possible to get any information related to the users, orders, returns.

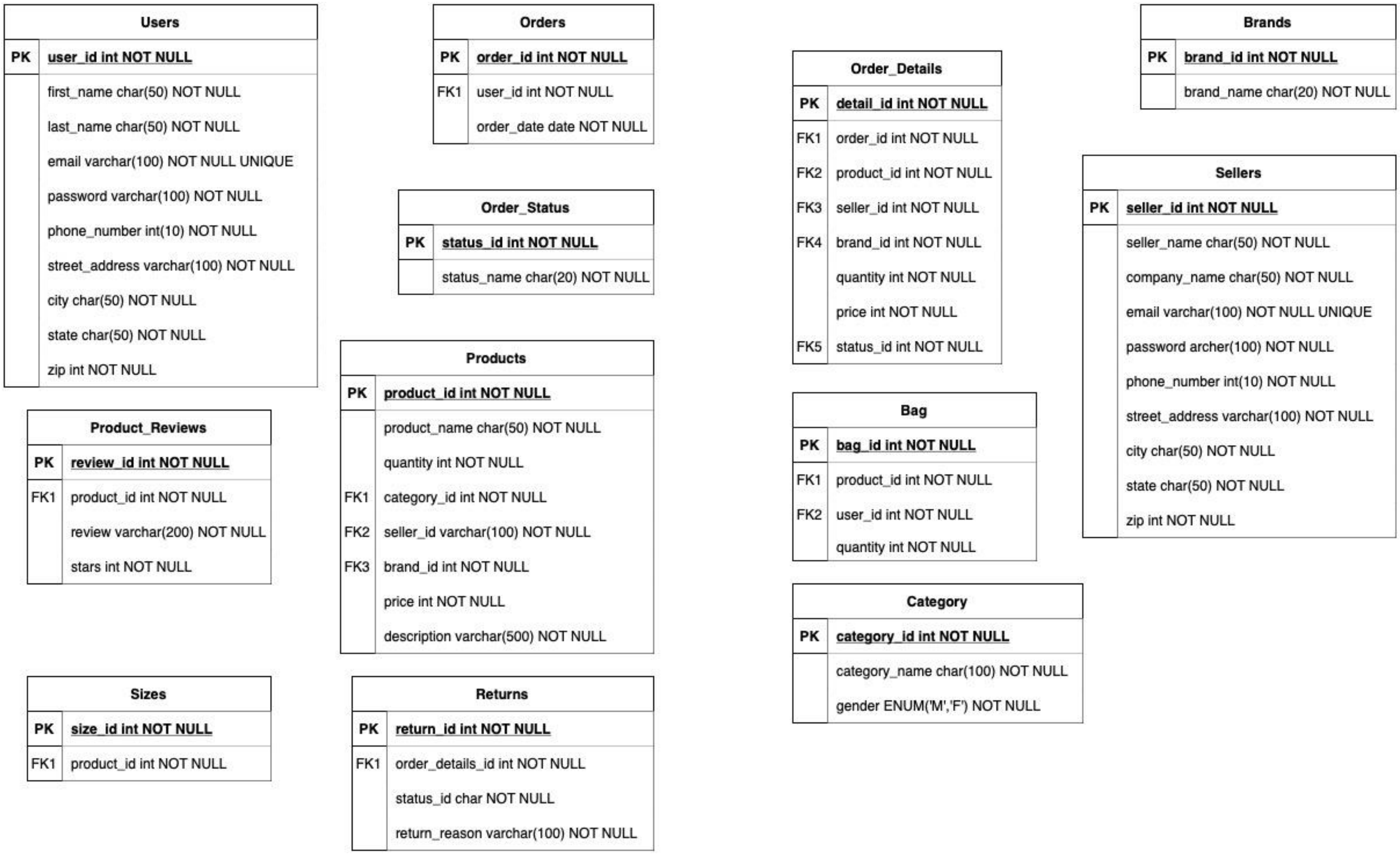
UML-compliant E-R Model



Business Rules

Entity 1	**Entity 2**	**Cardinality on Entity 1 side**	**Cardinality on Entity 2 side**	**Business Rule(s)**
users	orders	1..*	1	1 user can place 1 to many orders but 1 order can have only 1 user that places the order
users	bag	1	1	1 user can create only 1 bag and 1 bag can be created by only one user
products	category	1	1..*	1 product can be categorized into only 1 category but 1 category can contain more than 1 product
products	brand	1	1..*	1 product can be branded by only 1 brand but 1 brand can have more than 1 product
products	sellers	1	1..*	1 product can be sold by only 1 seller but 1 seller can sell more than 1 product
product_reviews	products	1	1..*	1 review can be for only 1 product but 1 product can have more than 1 review
order_details	orders	1	1..*	1 detail can be associated with only 1 order but 1 order can have more than 1 product so it can have more than 1 detail (1 detail for each product in the order)
order_details	seller	1	1..*	1 detail can have only 1 seller but 1 seller can have more than 1 detail as the products sold by that seller can be sold multiple times
order_details	brand	1	1..*	1 detail can have only 1 brand but 1 brand can have more than 1 detail as the products sold by that brand can be sold multiple times

Entity/Attribute descriptions



	Users
PK	**user_id int NOT NULL**
	first_name char(50) NOT NULL
	last_name char(50) NOT NULL
	email varchar(100) NOT NULL UNIQUE
	password varchar(100) NOT NULL
	phone_number int(10) NOT NULL
	street_address varchar(100) NOT NULL
	city char(50) NOT NULL
	state char(50) NOT NULL
	zip int NOT NULL

	Orders
PK	**order_id int NOT NULL**
FK1	user_id int NOT NULL
	order_date date NOT NULL

	Order_Details
PK	**detail_id int NOT NULL**
FK1	order_id int NOT NULL
FK2	product_id int NOT NULL
FK3	seller_id int NOT NULL
FK4	brand_id int NOT NULL
	quantity int NOT NULL
	price int NOT NULL
FK5	status_id int NOT NULL

	Brands
PK	**brand_id int NOT NULL**
	brand_name char(20) NOT NULL

	Order_Status
PK	**status_id int NOT NULL**
	status_name char(20) NOT NULL

	Sellers
PK	**seller_id int NOT NULL**
	seller_name char(50) NOT NULL
	company_name char(50) NOT NULL
	email varchar(100) NOT NULL UNIQUE
	password archer(100) NOT NULL
	phone_number int(10) NOT NULL
	street_address varchar(100) NOT NULL
	city char(50) NOT NULL
	state char(50) NOT NULL
	zip int NOT NULL

	Products
PK	**product_id int NOT NULL**
	product_name char(50) NOT NULL
	quantity int NOT NULL
FK1	category_id int NOT NULL
FK2	seller_id varchar(100) NOT NULL
FK3	brand_id int NOT NULL
	price int NOT NULL
	description varchar(500) NOT NULL

	Product_Reviews
PK	**review_id int NOT NULL**
FK1	product_id int NOT NULL
	review varchar(200) NOT NULL
	stars int NOT NULL

	Bag
PK	**bag_id int NOT NULL**
FK1	product_id int NOT NULL
FK2	user_id int NOT NULL
	quantity int NOT NULL

	Category
PK	**category_id int NOT NULL**
	category_name char(100) NOT NULL
	gender ENUM('M','F') NOT NULL

	Sizes
PK	**size_id int NOT NULL**
FK1	product_id int NOT NULL

	Returns
PK	**return_id int NOT NULL**
FK1	order_details_id int NOT NULL
	status_id char NOT NULL
	return_reason varchar(100) NOT NULL

Closing Section

We had a roller coaster experience with the project. It had its good days and bad. We would say the project taught us a lot about how important it is for each section of a system to work together and in sync with each other, just like how important it is for each team member to contribute to the project to make it a whole. The website has many parts to it, the database, the UI, the connection between the two, handling inputs, showing errors correctly and with everything being interdependent.

The easiest part was creating the UI since tailwind made it easy for us to quickly make the frontend look good for a good user experience. Tailwind was well documented, so it was easy to follow. We had a background in HTML as well so that made it quick to complete.

The difficult part was handling foreign keys. There is a part where we need to retrieve the returns and reference the order_details table for that and it was not easy to do. Scouring the internet was of not much help too. Even while insertion, the task of getting the foreign key to insert into the new record was not easy.

Overall, it was a fun project and something we had been looking forward to working on. The project teaches us a lot of different functionalities offered by Django, also helped us to improve our python skills. It being a complex website, gave us a chance to explore different ways of working with databases and how to structure data for best use.