Vanier College

Faculty of Science and Technology

GallerieM

System Development

420-942-VA

Deliverable 05

2023-10-06

Red Team

Justin Elmourne

Florjan Blakaj

David Lafleur

Roy Aldner Labado

Seyed Ali Ashrafizadeh

We, the Red Team, certify that this assignment is our own work

I, David Lalfeur, student ID# 0110341, certify that I have contributed to this deliverable, D.L.

I, Justin Elmourne, student ID# 1755556, certify that I have contributed to this deliverable, J.E.

I, Florjan Blakaj, student ID# 2395017, certify that I have contributed to this deliverable, F.B.

I, Roy Aldner Labado, student ID# 2395053, certify that I have contributed to this deliverable, R.L.

I, Seyed Ali Ashrafizadeh, student ID# 2395024, certify that I have contributed to this deliverable, A.S.

.

Table of Contents

Overview

GallerieM, is a local boutique situated in Montreal that specializes in handcrafted and personalized items, making it a top choice for events like weddings, anniversaries, and bridal showers. Their transition to an eCommerce platform comes with challenges such as maintaining a personal connection with customers, data security, competition, team training, and building a website. Selling handmade products adds complexity due to higher production costs and the need to convey the value of quality over mass-produced items.

Presently, GallerieM manages operations via Instagram and an Excel sheet to handle inquiries and custom product requests. They follow a creative process, including sketching, material acquisition, production, and delivery coordination, ensuring a personalized customer experience.

In the new system, there are three critical diagrams which include a flowchart, use case diagram with a description, and a class diagram. The flowchart visually represents the processes and interactions between administrators and users. While, the use case UML outlines the various interactions and activities involving users, administrators, and essential APIs. Lastly, The class diagram provides an overview of system components and their relationships.

Description of Client

Located in the heart of Montreal, GallerieM is a standout boutique, renowned for its handcrafted and personalized items. Their vast portfolio, from tailor-made t-shirts and elegant mugs to detailed invitation cards and delightful baby attire, positions them as a top choice for events like weddings, anniversaries, and more intimate affairs such as bridal showers. GallerieM operates with a clear B2C focus, handling both individualized requests and larger bulk orders, always ensuring a personal touch in every piece.

GallerieM, doesn't have any knowledge of programming but possesses a commendable understanding of digital tools. They're proficient with Excel spreadsheets and using them for various business needs. Furthermore, they are efficient with email communication, ensuring smooth interactions with customers and suppliers. As the business seeks to expand its digital footprint with an eCommerce platform, their foundational computer skills will be an asset, although the technicalities of web development will be a new territory. Her in-house design craftsmanship and a keen eye for detail, combined with her digital proficiency, set GallerieM on a path of promising growth amidst a competitive market, particularly during peak seasons and significant occasions.

Narrative Description

The database design for GallerieM, implemented in Python using Flask-SQLAlchemy, is created to address the unique challenges of transitioning from Instagram and Excel to an online system. It contains key entities like Administrator, Product, Customer, Orders, ShoppingCart, and ShoppingCartItem.

The Administrator class ensures access to the system that can modify and update the Product; The class which captures details like product type, description, size, and color, with the ability to personalized handcrafted products.

The Customer class represents users which allows users to utilize the Orders class which manages order information. The ShoppingCart and ShoppingCartItem classes handle the shopping cart functionality, allowing for seamless tracking of individual items, quantities, and custom details and is the access point for the Order class.

The database structure, with well-defined relationships and foreign key associations, provides a solid foundation for maintaining a personalized customer experience without the need to signup and, ensuring data security, and navigating handmade products online. The class diagram visually articulates the system's processes, interactions, and components, offering a comprehensive understanding for both administrators and users.

Data dictionary

ER diagram of the database

Database Tables

Entity Relationships

Database Code

```
class Administrator(db.Model):
        adminUsername = db.Column(db.String(255), primary_key=True)
        password = db.Column(db.String(255), nullable=False)

 class Product(db.Model):
        productId = db.Column(db.Integer, primary_key=True)
        productType = db.Column(db.String(255))
        productDescription = db.Column(db.Text)
        productSize = db.Column(PickleType)
        productColor = db.Column(PickleType)

 class Customer(db.Model):
        customerId = db.Column(db.Integer, primary_key=True)
        def __repr__(self):
        return f'<Customer {self.customerName} - Email: {self.email}>'

 class Orders(db.Model):
        orderId = db.Column(db.Integer, primary_key=True)
        customerId = db.Column(db.Integer,
db.ForeignKey('customer.customerId'))
        customerName = db.Column(db.String(255), nullable=False)
        address = db.Column(db.Text, nullable=False)
        email = db.Column(db.String(255), unique=True, nullable=False)
        creditCardInfo = db.Column(db.String(255))
        shippingInfo = db.Column(db.Text)
        orderDate = db.Column(db.Date)
        status = db.Column(db.String(50))
        total = db.Column(db.Float)

 class ShoppingCart(db.Model):
        cartID = db.Column(db.Integer, primary_key=True)
        total = db.Column(db.Float)
        customerId = db.Column(db.Integer,
db.ForeignKey('customer.customerId'))
        itemID = db.Column(db.Integer,
db.ForeignKey('shopping_cart_item.itemID'))

 class ShoppingCartItem(db.Model):
        itemID = db.Column(db.Integer, primary_key=True)
        refProductID = db.Column(db.Integer,
db.ForeignKey('product.productId'))
        cartID = db.Column(db.Integer, db.ForeignKey('shopping_cart.cartID'))
        quantity = db.Column(db.Integer, nullable=False)
        customSize = db.Column(db.String(50))
        customColor = db.Column(db.String(50))
        customDesign = db.Column(db.Text)
```

Appendix 3

Given that we're structuring the database using Python models, complex queries are not required, especially since we are relying on API calls for retrieving data related to products, shopping carts, and orders. The design focuses on simplicity and efficiency, utilizing API calls access data. However, while the queries themselves will not be complex within the database, optimizing the API calls and ensuring efficient data retrieval is important. Strategies such as indexing, caching, and querying are still relevant in the context of API interactions for the website to remain responsive and performant.

Appendix 4

The access speed required for GallerieM's platform is crucial to delivering a responsive and seamless user experience. The database will encounter infrequent access, primarily serving the purpose of fetching product listings, creating customer orders, and storing and updating shopping cart information. While the database will not encounter constant queries, the significance of rapid retrieval is important during key user interactions. To meet this standard, the design prioritizes simplicity and efficiency, leveraging indexing on essential columns and implementing a caching mechanism for static data. The infrequent use of the database access emphasizes the need for optimized response times to ensure smooth and efficient operations for both administrators and customers.