

# BurgerHub

## FINAL PROJECT REPORT

### Project Description

Introducing BurgerHub, your one-stop destination for delicious eats delivered straight to your doorstep in 25 minutes or less. With our innovative app and efficient delivery network, satisfying your cravings has never been easier. From mouthwatering burgers to crispy fries, refreshing drinks, and indulgent desserts, we offer a wide range of options to please every palate.

At BurgerHub, we prioritize convenience and quality. Our user-friendly platform allows you to browse our extensive menu, place orders for speedy delivery or convenient pickup, and track your food in real-time. Plus, with our commitment to prompt service, you can count on your order arriving piping hot and ready to enjoy in no time.

Behind the scenes, our dedicated team is hard at work sourcing the freshest ingredients and crafting flavorful offerings to tantalize your taste buds. Whether you're craving a classic cheeseburger, crispy fries, a refreshing soda, or a decadent dessert, BurgerHub has something for everyone.

Join us today and experience the convenience of fast, reliable food delivery with BurgerHub. Because when hunger strikes, we've got you covered, delivering your favorite eats right to your doorstep in 25 minutes or less.

In addition to offering speedy delivery and a diverse menu, BurgerHub provides flexibility and peace of mind to its customers. With our easy-to-use app, you can not only place orders for prompt delivery or convenient pickup but also have the option to cancel your order before delivery if needed. While we strive for accuracy in every order, we understand that mistakes can happen. In the rare event that the wrong order is delivered, rest assured that you will receive a full refund without the hassle of returning the items.

BurgerHub's menu is designed to cater to all tastes and preferences. From Specialty Burgers crafted with unique flavors to Classic Burgers that never disappoint, we have something to satisfy every craving. Our selection of beverages and desserts adds the perfect finishing touch to your meal, ensuring a truly satisfying dining experience. Plus, with our customizable burger toppings—including cheese, bacon, lettuce, pickles, tomatoes, and more—you can create your perfect burger without any additional charges.

For added convenience, BurgerHub accepts online payments, making the ordering process seamless and hassle-free. Whether you choose pickup or delivery, you can trust BurgerHub to deliver your favorite eats promptly and with the utmost care. Join us today and discover the joy of convenient, delicious dining with BurgerHub.

# BurgerHub

In conclusion, BurgerHub is more than just a food delivery service—it's a culinary experience designed to delight and satisfy. With our diverse menu, efficient delivery, and commitment to quality, we're redefining the way you enjoy your favorite eats. Whether you're craving a classic burger, crispy fries, refreshing drinks, or indulgent desserts, BurgerHub delivers it all with speed, convenience, and a dash of flavor. Join us today and let BurgerHub bring the taste of satisfaction right to your doorstep.

Overall, BurgerHub is an online food ordering platform designed to streamline the process of ordering delicious burgers and other fast-food items for customers. The platform consists of a user-friendly interface for customers to place orders, as well as an administrative interface for managing products, orders, and delivery logistics.

Key Features:

## **1. User Registration and Authentication:**

Customers can create accounts by providing their first name, last name, email address, and password. Authentication mechanisms ensure secure access to user accounts.

## **2. Order Placement:**

Users can browse the menu, add items to their cart, and place orders. Order details include selected items, order status, and delivery preferences (if applicable).

## **3. Order Tracking:**

Customers can track the status of their orders in real-time, from placement to delivery.

## **4. Admin Dashboard:**

Administrators have access to a dashboard for managing various aspects of the platform. Features include adding/editing products, managing user accounts, and assigning delivery agents.

## **5. Delivery Management:**

Admins can assign delivery agents to orders and track delivery progress. Delivery details such as address, contact information, and delivery status are maintained.

## **6. Database Structure:**

The database comprises three main collections: Users, Orders, and Admins. Relationships between collections ensure seamless data flow and efficient operation.

# BurgerHub

## **Database Description:**

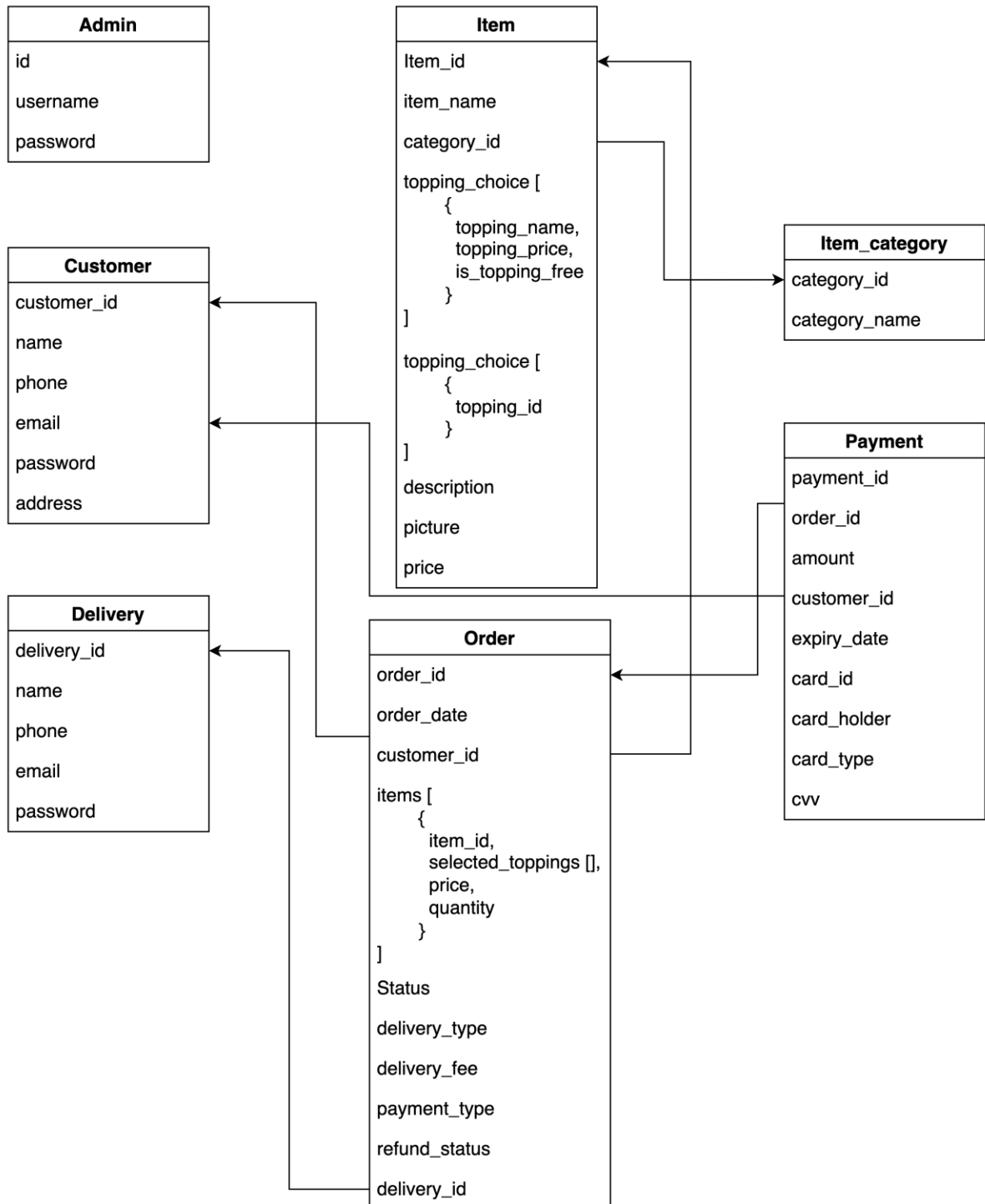
The BurgerHub's database comprises several collections to facilitate its operations. The "Users" collection stores customer information, including first name, last name, email address, and password for account creation and authentication. Additionally, the "Orders" collection tracks customer orders, including the items purchased, order status, and delivery details if applicable. A separate "Admin" collection holds information for system administrators, enabling them to manage lists, product details, and assign delivery agents. Each collection maintains relationships where necessary, ensuring seamless data flow and efficient operation of the BurgerHub platform.

## **DATABASE DESIGN:**

- Customer collection consists of customer\_id, name, phone\_num, email\_id, password and house address.
- Admin collection consists of details such as id, name, password and username.
- Item collection consists of item\_id, item\_name, category\_id, topping\_choice[(topping\_name, topping\_price, is\_topping\_free), description, price and picture.
- Item\_Category collection contains category\_id, category\_name.
- Delivery collection consists of delivery\_id, name, phone\_num, email\_id, password.
- Order collection consists of order\_id, order\_date, customer\_id, order\_status, delivery\_type, delivery\_fee, payment\_type, refund\_status, delivery\_id, items[(item\_id, selected\_toppings[], quantity).
- Payment collection consists of payment\_id, order\_id, amount, card\_number, expiry\_date, card\_holder, card\_type, cvv.

# BurgerHub

## Database Design and data dictionary:



# BurgerHub

admin				
Storage size: 20.48 kB	Documents: 1	Avg. document size: 69.00 B	Indexes: 1	Total index size: 20.48 kB
customer				
Storage size: 20.48 kB	Documents: 1	Avg. document size: 134.00 B	Indexes: 1	Total index size: 20.48 kB
delivery				
Storage size: 20.48 kB	Documents: 1	Avg. document size: 105.00 B	Indexes: 1	Total index size: 20.48 kB
item				
Storage size: 20.48 kB	Documents: 0	Avg. document size: 0 B	Indexes: 1	Total index size: 20.48 kB
item_category				
Storage size: 20.48 kB	Documents: 1	Avg. document size: 49.00 B	Indexes: 1	Total index size: 36.86 kB
order				
Storage size: 20.48 kB	Documents: 0	Avg. document size: 0 B	Indexes: 1	Total index size: 20.48 kB
payment				
Storage size: 20.48 kB	Documents: 1	Avg. document size: 184.00 B	Indexes: 1	Total index size: 20.48 kB

# BurgerHub

## Sample Data:

### BurgerHub.admin

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

Filter



Type a query: { field: 'value' }

+ ADD DATA

EXPORT COLLECTION

```
_id: ObjectId('65f78d13ccdb329d440a7c70')
email: "admin@gmail.com"
password: "admin"
```

### BurgerHub.customer

Documents

Aggregations

Schema

Explain Plan

Filter



Type a query: { field: 'value' }

+ ADD DATA

EXPORT COLLECTION

```
_id: ObjectId('65f78dae0b6b51bf8b92fb15')
name: "Rajesh"
phone: "8765384623"
email: "rajesh@gmail.com"
password: "12345"
address: "Warrensburg"
```

# BurgerHub

## BurgerHub.delivery

Documents

Aggregations

Schema

Explain Plan

Filter



Type a query: { field: 'value' }

+ ADD DATA

EXPORT COLLECTION

```
_id: ObjectId('65f78dfc6438e85035270fbc')  
name: "raju"  
phone: "8734857436"  
email: "raju@gmail.com"  
password: "12345"
```

# BurgerHub

## BurgerHub.item

Documents

Aggregations

Schema

Explain Plan

Indexes

V

Filter 



Type a query: { field: 'value' }

 ADD DATA ▼

 EXPORT COLLECTION

```
_id: ObjectId('660476469334be243b79b8ae')
name: "Spicy Chicken Burger"
picture: "burger.png"
price: "18.99"
category_id: ObjectId('65f78cb5ccdb329d440a7c6e')
description: "Spicy burger topped with hot chicken patty"
▼ topping_choice: Array
  ▼ 0: Object
    topping_name: "Ranch"
    topping_price: 0.65
    is_topping_free: true
  ▼ 1: Object
    topping_name: "Lettuce"
    topping_price: 0.65
    is_topping_free: true
  ▼ 2: Object
    topping_name: "Pickle"
    topping_price: 0.65
    is_topping_free: false
```



# BurgerHub

## BurgerHub.item\_category

Documents

Aggregations

Schema

Explain Plan

Filter



Type a query: { field: 'value' }

+ ADD DATA

EXPORT COLLECTION

```
_id: ObjectId('65f78cb5ccdb329d440a7c6e')
category_name: "burgers"
```

## BurgerHub.order

Documents

Aggregations

Schema

Explain Plan

Indexes

Filter



Type a query: { field: 'value' }

+ ADD DATA

EXPORT COLLECTION

```
_id: ObjectId('66047756a03989cb1026dd7d')
order_date: 2024-03-27T14:45:26.216+00:00
customer_id: ObjectId('65f78dae0b6b51bf8b92fb15')
order_status: "pending"
▼ items: Array
  ▼ 0: Object
    product_id: ObjectId('660476469334be243b79b8ae')
    ▼ selected_toppings: Array
      0: "Ranch"
      1: "Lettuce"
    price: "18.99"
    quantity: 1
  delivery_type: "delivery"
  amount: 18.99
  delivery_id: ObjectId('65f78dfc6438e85035270fbc')
  delivery_charges: 3.33
  payment_type: "card"
  refund_status: "null"
```

# BurgerHub

## BurgerHub.payment

[Documents](#)[Aggregations](#)[Schema](#)[Explain Plan](#)[Index](#)[Filter](#)

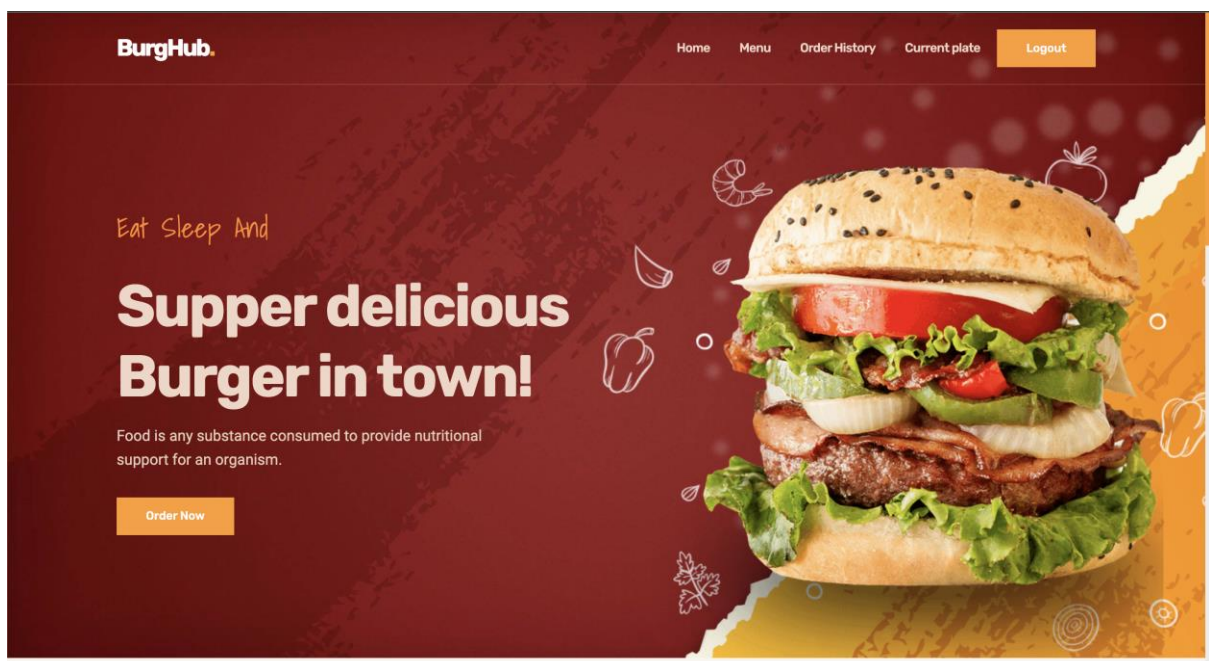
Type a query: { field: 'value' }

[+ ADD DATA](#)[EXPORT COLLECTION](#)

```
_id: ObjectId('65f790acde5aaa325f56b36e')
order_id: ObjectId('65f7906b3f3b70e2a1b7ba9d')
customer_id: ObjectId('65f78dae0b6b51bf8b92fb15')
expiry_date: "05/20"
amount: "22.32"
card_id: "91823718231232"
card_holder: "rajesh"
card_type: "credit"
```

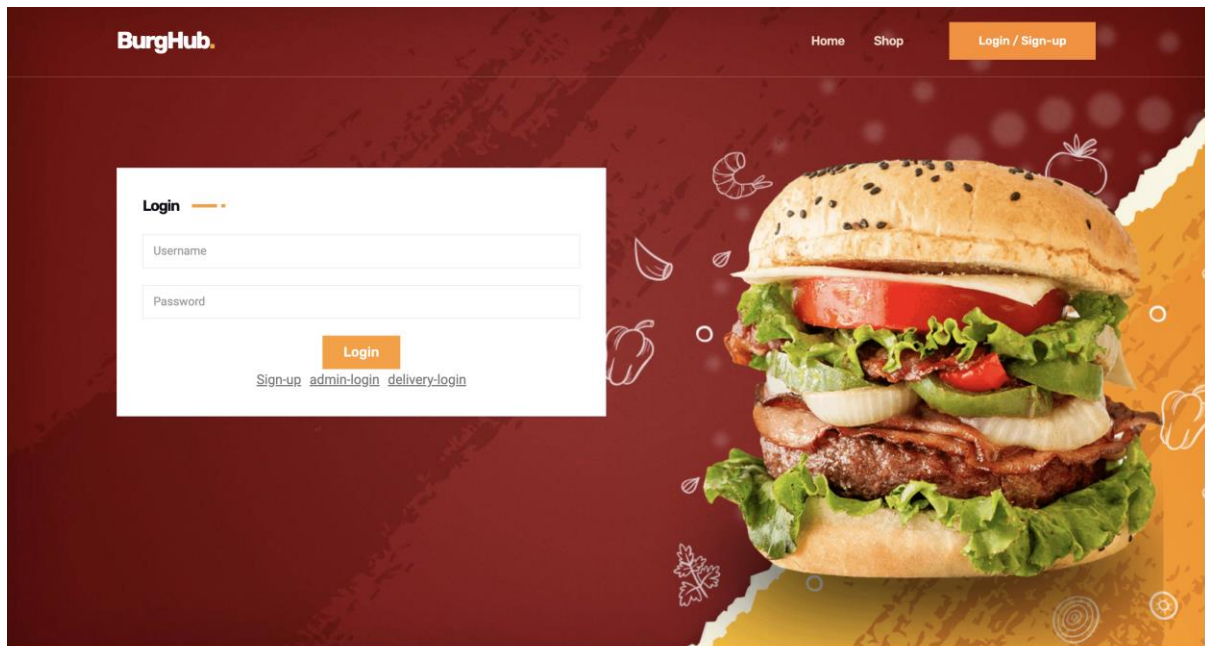
## User Interfaces and Forms:

>> Home Page:



# BurgerHub

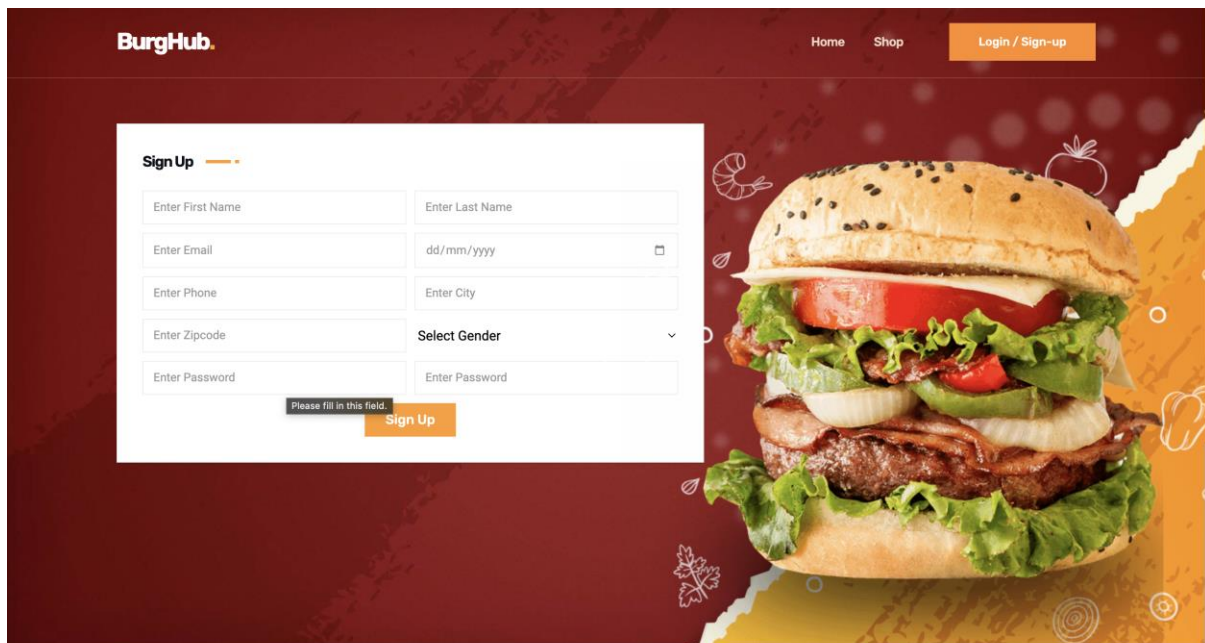
## >> Login Page:



The login page features a dark red background with a large, detailed image of a burger on the right side. The burger is topped with lettuce, tomato, onion, and a beef patty. The background also includes faint white line art of various food items like a lemon, a pepper, and a tomato. The login form is a white box on the left with the following elements:

- BurgHub.** logo in the top left corner.
- Navigation links: [Home](#), [Shop](#), and a [Login / Sign-up](#) button in the top right corner.
- Login** section header with a small orange bar.
- Username input field.
- Password input field.
- [Login](#) button.
- Links for [Sign-up](#), [admin-login](#), and [delivery-login](#).

## >> Signup Page



The signup page features a dark red background with a large, detailed image of a burger on the right side. The burger is topped with lettuce, tomato, onion, and a beef patty. The background also includes faint white line art of various food items like a lemon, a pepper, and a tomato. The signup form is a white box on the left with the following elements:

- BurgHub.** logo in the top left corner.
- Navigation links: [Home](#), [Shop](#), and a [Login / Sign-up](#) button in the top right corner.
- Sign Up** section header with a small orange bar.
- Form fields for: Enter First Name, Enter Last Name, Enter Email, dd/mm/yyyy (with a calendar icon), Enter Phone, Enter City, Enter Zipcode, Select Gender (with a dropdown arrow), Enter Password, and Enter Password (repeated).
- [Sign Up](#) button.
- A small grey tooltip that says "Please fill in this field." pointing to the Sign Up button.

# BurgerHub

## Burger Beef

Spicy hjbsd

burgers

Price: \$5

Add to plate



Menu Dishes

## Our Delicious Foods

An original then, an original now.

Allburgersdrink



burgers  
**Burger Beef**  
PRICE: \$5





burgers  
**Spicy Chicken Burger**  
PRICE: \$5

# BurgerHub

BurgHub.

Home Menu Order History Current plate Logout

### Selected items on plate

DELETE	MENU ITEM	ITEM NAME	PRICE	QUANTITY	SUBTOTAL
Delete		Burger Beef	\$5	<input type="text" value="5"/>	25.0
Delete		Spicy Chicken Burger	\$5	<input type="text" value="5"/>	25.0
Select Delivery Mode					SubTotal : \$50.0
Delivery					Tax : \$1.5
					Delivery Charges : \$2.99
					Total : \$58.85

Checkout

BurgHub.

Contact Info

Opening Hours

BurgHub.

Home Menu Order History Current plate Logout

### Add Payment Card Details

Name

Select Card

Total amount(\$)

58.85

Card Number

Expiry MM/YY

Enter CVV

Enter Street Number

Enter City

Enter ZipCode

Complete Payment

# BurgerHub

Add Menu Category

Add Category

ACTION	CATEGORY NAME
Edit	burgers
Edit	drink

127.0.0.1:5001/add-category

Add Menu Item

Select Category

Price

Add Optional Toppings

Add Item

Upload Picture

Choose file

No file chosen

Add Menu Item

# BurgerHub

Add Delivery Guy

Enter Name

Phone

Enter email

Password

Add to team

SNO.	NAME	PHONE	EMAIL
1	raju	raju@gmail.com	8734857436
2	Gayatri	gayatri@gmail.com	77436537464

## Orders to pickup

ORDER ID	STATUS	DELIVERY ADDRESS	ACTION
----------	--------	------------------	--------

BurgHub.

Financial experts support or help you to to find out which way you can raise your funds more.

Contact Info

+1 (062) 109-9222  
Info@burghub.com  
S Magwire St,  
Warrensburg 64093

Opening Hours

Monday-Friday: 08:00-22:00  
Tuesday 4PM: Till Mid Night  
Saturday: 10:00-16:00

© 2024 Spring BurgHub Adb All Rights Reserved.



# BurgerHub

## Source Codes:

```

1  from flask import Flask, render_template, session, redirect, request
2  import pymongo
3  import datetime
4  import os.path
5  from bson import ObjectId
6  import re
7  import string
8  import time, datetime
9  import random
10 import os
11
12 my_client = pymongo.MongoClient("mongodb://localhost:27017")
13 my_db = my_client.get_database("BurgerHub")
14 app = Flask(__name__)
15 app.secret_key = "bh"
16 APP_ROOT = os.path.dirname(os.path.abspath(__file__))
17
18 admin_data = my_db["admin"]
19 customer_data = my_db["customer"]
20 delivery_data = my_db["delivery"]
21 product_data = my_db["item"]
22 product_category_data = my_db["item_category"]
23 order_data = my_db["order"]
24 payment_collection = my_db["payment"]
25 topping_data = my_db['topping']
26

```

[illegible]



# BurgerHub

```
# /add-to-plate
@app.route("/add-to-plate")
def add_to_cart():
    if "role" not in session:
        return redirect(
            "/login?message=Sign in or Sign up to get started with plate"
        )
    if session["role"] == "Customer":
        quantity = request.args.get("qty")
        product_id = request.args.get("product_id")
        product = product_data.find_one({'_id': ObjectId(product_id)})
        product_name = product['name']
        customer_id = session["customer_id"]
        selected_toppings = request.args.getlist('toppings[]')
        query = {"customer_id": ObjectId(customer_id), "status": "cart"}
        count = order_data.count_documents(query)

        query = {"customer_id": ObjectId(customer_id), "status": "cart", "items.product_id": product['_id']}
        existing_order = order_data.find_one(query)

        if existing_order:
            quantity = int(existing_order["items"][0]["quantity"]) + 1
            print(quantity)
            order_data.update_one(
                {
```

```
@app.route('/update-delivery-type', methods=['POST'])
def update_delivery_type():
    customer_id = session['customer_id']
    delivery_type = request.args.get('delivery_type')
    if order_data.count_documents({'customer_id': ObjectId(customer_id), 'status': 'cart'}) > 0:
        order = order_data.find_one({'customer_id': ObjectId(customer_id), 'status': 'cart'})
        order_data.update_one(
            {
                "_id": ObjectId(order['_id'])
            },
            {
                "$set": { "delivery_type": delivery_type }
            }
        )

    return redirect('/view_plate?message=changed delivery mode') # Assuming view_plate is the endpoint for the view-plate
    page

@app.route("/login")
def login():
    return render_template('login.html')
```

# BurgerHub

```
@app.route("/login")
def login():
    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect('/?message=Logout Successful!')

@app.route("/sign-up")
def sign_up():
    return render_template('sign-up.html')

@app.route("/admin-login")
def admin_login():
    return render_template('admin-login.html')

@app.route("/delivery-login")
def delivery_login():
    return render_template('delivery-login.html')

@app.route("/delivery-home")
def delivery_home():
    return render_template('delivery-home.html')
```

# BurgerHub

```
@app.route('/admin-home')
def admin_home():
    return render_template('admin-home.html')

@app.route('/customer-home')
def customer_home():
    message = request.args.get('message')
    cat_id = request.args.get('_id')
    products = product_data.find({})
    if cat_id:
        products = product_data.find({'category_id': ObjectId(cat_id)})
    categories = product_category_data.find({})
    return render_template('customer-home.html', products=products, categories=categories,
        getCategoryNameById=getCategoryNameById, message=message)

@app.route("/remove")
def remove():
    product_id = request.args.get("product_id")
    order_id = request.args.get("order_id")
    # Remove the specific item from the items array
    query = {
        "_id": ObjectId(order_id),
        "status": "cart",
        "items.product_id": ObjectId(product_id),
    }
    update_operation = {"$pull": {"items": {"product_id": ObjectId(product_id)}}}
    order_data.update_one(query, update_operation)
```

```
@app.route('/view-product')
def view_product():
    message = request.args.get('message')
    product_id = request.args.get('product_id')
    product = product_data.find_one({'_id': ObjectId(product_id)})
    return render_template('product-view.html', product = product, getCategoryNameById=getCategoryNameById,
        message=message, get_toppings_from_item=get_toppings_from_item)

# @app.route('/view-plate')
# def view_plate():
#     return render_template('view-plate.html')

@app.route("/view-plate")
def cart():
    message = request.args.get('message')
    if "role" not in session:
        return render_template(
            "login.html",
            message="Login to your Account",
        )
    if session["role"] == "Customer":
        order = order_data.find_one(
            {"customer_id": ObjectId(session["customer_id"]), "status": "cart"}
        )
        count = order_data.count_documents(
```

# BurgerHub

```
@app.route('/add-item')
def add_item():
    products = product_data.find({})
    message = request.args.get('message')
    categories = product_category_data.find({})
    return render_template('admin-add-product.html', products = products, categories=list(categories),
                           getCategoryNameById=getCategoryNameById, message=message)
```

```
@app.route("/delivery-orders")
def delivery_orders():
    message = request.args.get('message')
    filterType = request.args.get("status")
    print(filterType)
    print({"delivery_assigned": ObjectId(session["delivery_id"])})
    orders = order_data.find(
        {"delivery_id": ObjectId(session["delivery_id"])})
    )
    if filterType:
        query = {
            "delivery_id": ObjectId(session["delivery_id"]),
            "status": filterType,
        }
        orders = order_data.find(
            query
        )
    return render_template(
```

```
@app.route('/change-delivery-status')
def delivery_order():
    order_id = request.args.get('order_id')
    status = request.args.get('status')
    delivery_id = ObjectId(session['delivery_id'])
    if status == 'out for delivery':
        order_data.update_one({'_id': ObjectId(order_id), 'delivery_id': delivery_id}, {'$set': {'status': 'out for delivery'}})
    if status == 'delivered':
        order_data.update_one({'_id': ObjectId(order_id), 'delivery_id': delivery_id}, {'$set': {'status': 'delivered'}})
    return redirect('/delivery-orders?status='+status)
```

```
@app.route('/remove-product')
def admin_remove_product():
    product_id = request.args.get('product_id')
    product_data.delete_one({'_id': ObjectId(product_id)})
    return redirect('/add-item?message=item removed successfully')
```

# BurgerHub

```
@app.route("/admin-order")
def admin_order():
    order_id = request.args.get("order_id")
    action = request.args.get("action")
    order = order_data.find_one({'_id': ObjectId(order_id)})
    if order['delivery_type'] == 'delivery':
        if action == 'accepted':
            order_data.update_one({'_id': ObjectId(order_id)}, {'$set': {'status': 'accepted'}})
        elif action == 'rejected':
            order_data.update_one({'_id': ObjectId(order_id)}, {'$set': {'status': 'rejected', 'refund_status': 'processing'}})
        elif action == 'prepared':
            order_data.update_one({'_id': ObjectId(order_id)}, {'$set': {'status': 'prepared'}})
        elif action == 'assigned':
            delivery_id = request.args.get('delivery_id')
            order_data.update_one({'_id': ObjectId(order_id)}, {'$set': {'status': 'assigned', 'delivery_id': ObjectId(delivery_id)}})
        elif action == 'process_refund':
            order_data.update_one({'_id': ObjectId(order_id)}, {'$set': {'status': 'refund processed'}})
    if order['delivery_type'] == 'pickup':
        if action == 'accepted':
            order_data.update_one({'_id': ObjectId(order_id)}, {'$set': {'status': 'accepted'}})
        elif action == 'rejected':
            order_data.update_one({'_id': ObjectId(order_id)}, {'$set': {'status': 'rejected', 'refund_status': 'processing'}})
        elif action == 'prepared':
```

```
@app.route('/payment-portal')
def payment_portal():
    message = request.args.get('message')
    subtotal = request.args.get('subtotal')
    order_id = request.args.get('order_id')
    total = request.args.get('total')
    delivery_type = request.args.get('delivery_type')
    return render_template('customer-payment.html', total = total, order_id = order_id, subtotal=subtotal, message=message, delivery_type=delivery_type)

@app.route('/verify-transaction', methods=['POST'])
def verify_payment():
    order_id = request.form.get("order_id")
    amount = request.form.get("total")
    customer_id = ObjectId(session["customer_id"])
    name = request.form.get("card_name")
    number = request.form.get("card_number")
    payment_type = request.form.get("payment_type")
    expiry = request.form.get("expiry")
    cvv = request.form.get("cvv")
    street = request.form.get("street")
    zip = request.form.get("zip")
    city = request.form.get("city")
    # address = request.form.get('address')
    # subtotal = request.form.get('subtotal')
    order_details = {}
```

# BurgerHub

```
@app.route('/customer-orders')
def customer_orders():
    message = request.args.get("message")
    if "role" not in session:
        return render_template(
            "login.html",
            message="No Orders So Far",
        )
    if session["role"] == "Customer":
        orders = order_data.find(
            {"customer_id": ObjectId(session["customer_id"]), "status": {"$ne": "cart"}}
        )
        count = order_data.count_documents(
            {"customer_id": ObjectId(session["customer_id"]), "status": {"$ne": "cart"}}
        )
        print( 'customer-orders' )
        return render_template(
            "customer-orders.html",
            orders=orders,
            orders_count = count,
            get_product_by_product_id=get_product_by_product_id,
            getUpperIdFromOrderId=getUpperIdFromOrderId,
            message=request.args.get("message")
        )
    return render_template("customer-orders.html", message=message)
```

# BurgerHub

## NoSQL Commands:

```
query = {
    "customer_id": ObjectId(customer_id),
    "status": "cart",
    "items": [product_to_cart],
    "delivery_type": 'delivery'
}
order_data.insert_one(query)
```

```
payment_details = {
    "order_id": ObjectId(order_id),
    "amount": amount,
    "customer_id": customer_id,
    "card_holder": name,
    "card_id": number,
    "payment_type": payment_type,
    "expiry_date": expiry,
    "cvv": cvv,
    "payment_date": datetime.datetime.now().strftime("%m-%d-%Y")
}
payment_collection.insert_one(payment_details)
```

```
order_data.update_one(
    {
        "_id": existing_order["_id"],
        "status": "cart",
        "items.product_id": product['_id'],
    },
    {"$set": {"items.$.quantity": quantity}},
)
return redirect("/view-plate")
```

# BurgerHub

```
order_data.update_one(  
    {  
        "_id": ObjectId(order_id),  
        "status": "cart",  
        "items.product_id": ObjectId(product_id),  
    },  
    {"$set": {"items.$.quantity": quantity}},  
)
```

```
order_data.delete_one({"_id": ObjectId(order_id)})
```

```
products = product_data.find({'category_id': ObjectId(cat_id)})
```