CSE370 : Database Systems
Project Report
Project Title : DrugWeb

Group No : 13, CSE370 Lab Section : 19, FALL 2025		
ID	**Name**	**Contribution**
22201512	SWAGATA DEY	Customer dashboard, Search, add to cart, review.
22301298	SAJID MUHAMMOD	Admin dashboard, assign deliveryman, customer's requested medicine, handle request.
21301269	MUSHFIRAT CHOWDHURY	Deliveryman dashboard, Deliverymen confirm payment, customer payment procedure and earn points.

Table of Contents

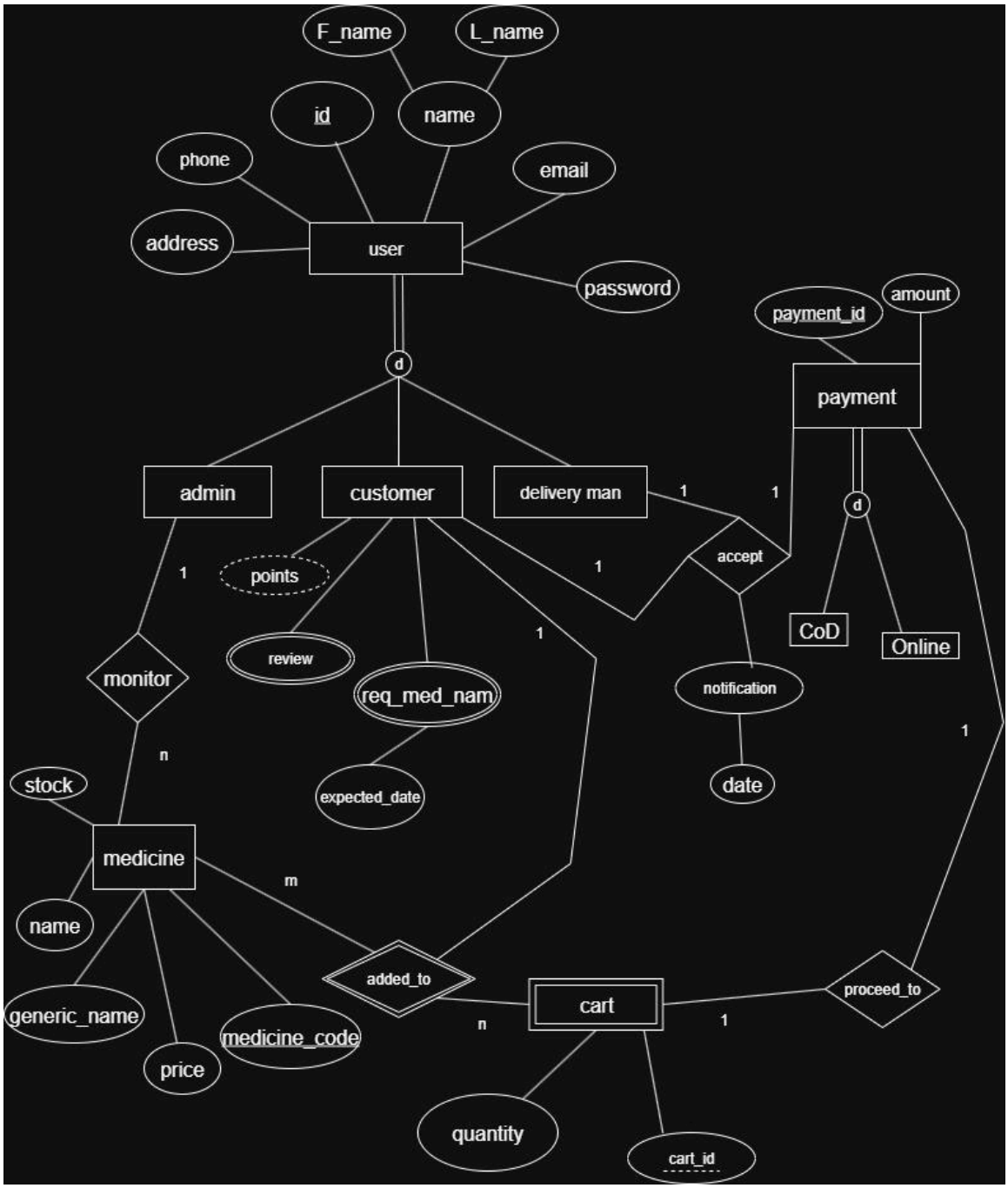
Introduction

“DrugWeb” is an online pharmacy where a customer can buy medicines according to their needs. Our website has three types of users such as customer, admin, deliveryman. Firstly, a customer can easily search and buy medicines which are in our stock. But in case a medicine which is not in our stock, then customers can request the name of the medicines with the expected date. Then our admin can accept or reject the request according to the availability of that medicine. Again, customers can give reviews and can see other customers reviews of our previous services. Customers can add to cart any medicines and after a successful payment they can earn points. Secondly, admin can see all reviews, monitor the stock of the medicines, handle the requested medicines and payments by assigning a deliveryman. Thirdly, a deliveryman can see the assigned order and can choose a date for the delivery. After confirmation by the deliveryman, a notification will notify the customers with the delivery date.

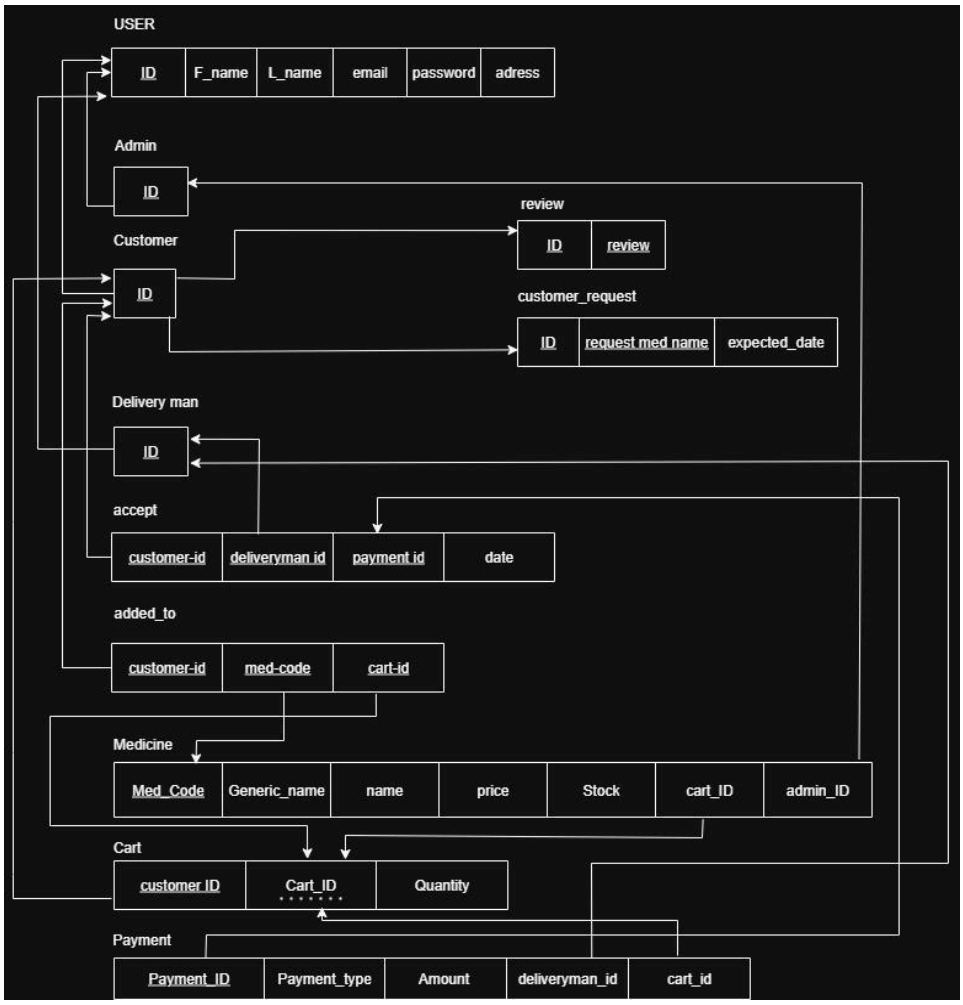
Project Features

ID, Name	Features [3 per member]	
22201512 SWAGATA DEY	Ft 1	Customer dashboard and notification
	Ft 2	Search by different order
	Ft 3	Add to cart and review
22301298 SAJID MUHAMMOD	Ft 1	Admin dashboard
	Ft 2	Assign deliveryman
	Ft 3	Customer’s requested medicine and handle request by admin
21301269 MUSHFIRAT CHOWDHURY	Ft 1	Deliveryman dashboard
	Ft 2	Deliveryman’s confirm payment
	Ft 3	Customer payment procedure and earn points

ER/EER Diagram



Schema Diagram



Normalization

a. Explain if your converted Schema is in 1NF or not. If not, decompose it to 1NF.
b. Explain if your converted Schema is in 2NF or not. If not, decompose it to 2NF. Can there be any partial functional dependencies in your relational schema?
c. Explain if your converted Schema is in 3NF or not. If not, decompose it to 3NF. Can there be any transitive dependencies in your relational schema?

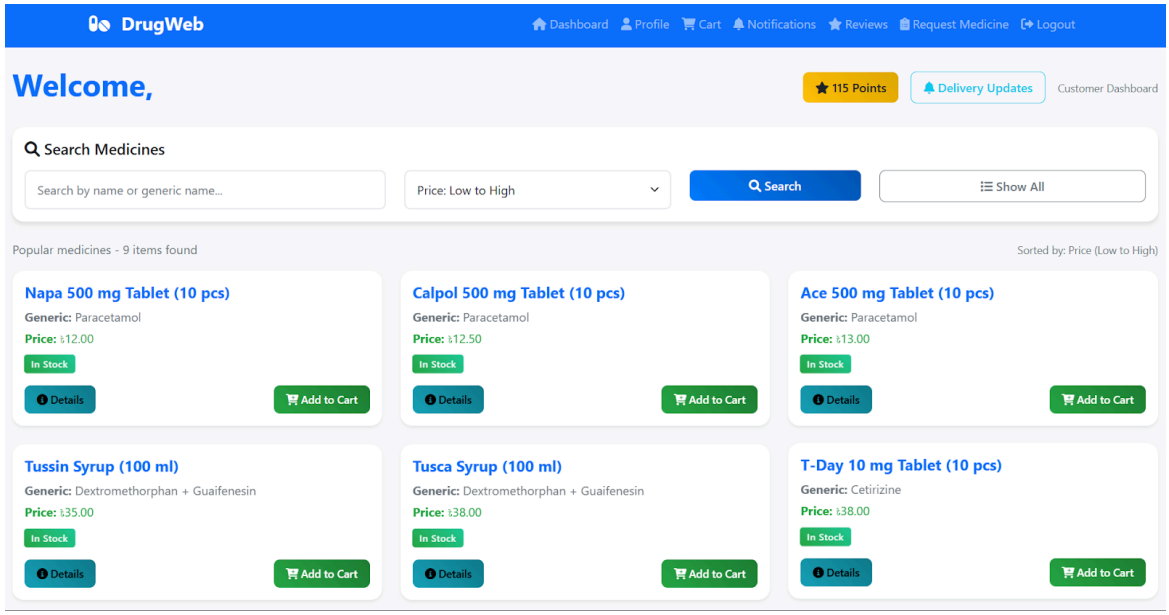
- Our schema diagram is already in 3NF. So, we didn't have to convert our schema into 3NF.

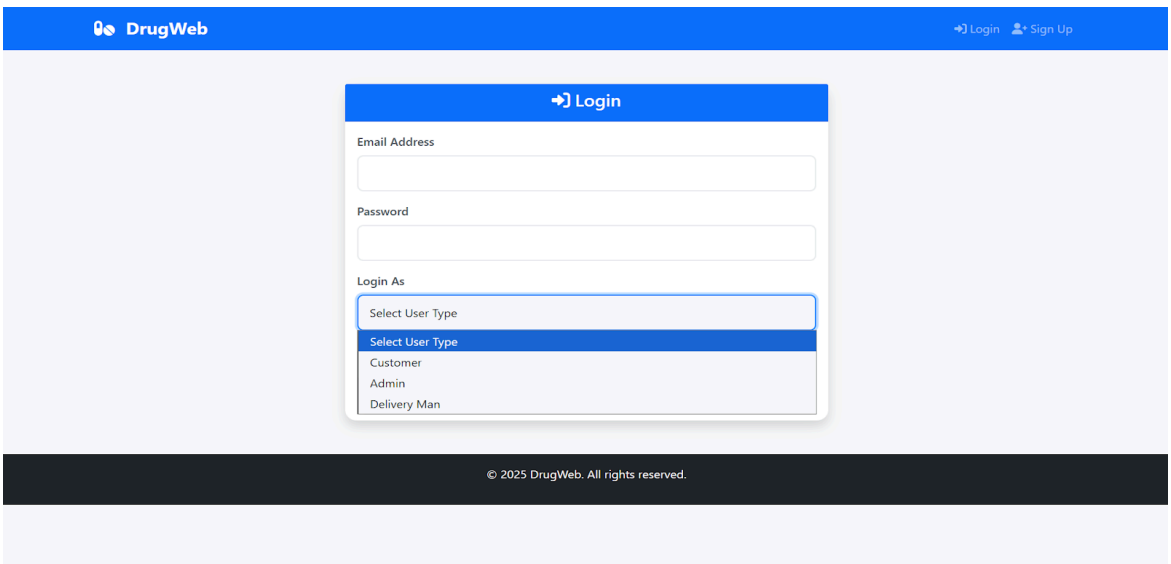
Frontend Development

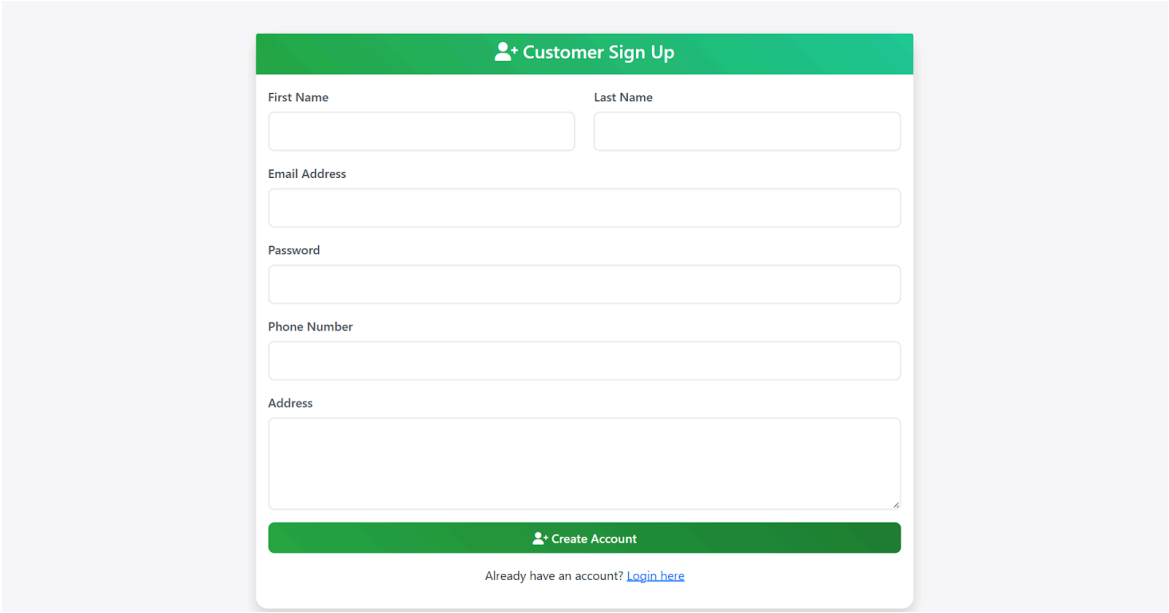
In our frontend, we used HTML and CSS to show the data and basic templates.

Contribution of ID : 22201512, NAME: SWAGATA DEY

- Customer homepage design where a customer can find reviews, request medicine, profile, cart, notification section along with the search option.
- Login and signup page where a user has to login according to his role.







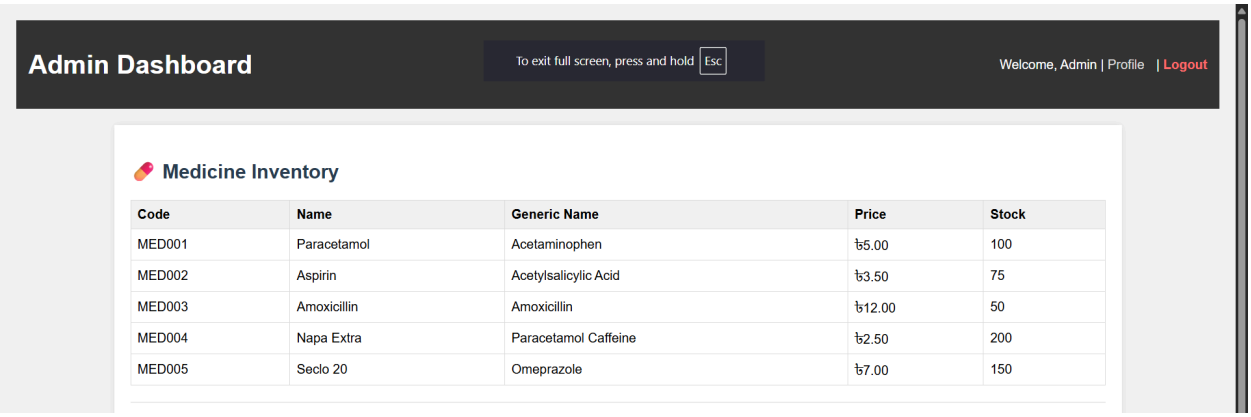
Contribution of ID : 22301298, NAME: SAJID MUHAMMOD

1. Admin Interface

- **Central Dashboard:** Provides real-time tables for monitoring Medicine Stock, Customer Reviews, and Pending Medicine Requests.
- **Quick Actions:** "Accept" or "Decline" buttons for medicine requests directly on the dashboard to handle customer needs instantly.
- **Order Management:** A specialized "Payments" page to view all orders and a dropdown interface to **Assign Deliverymen** to specific orders.

2. Customer Interface

- **Request System:** A dedicated form to request unavailable medicines, complete with a history table that tracks the status (Pending/Accepted) of each request.



Medicine Requests

Customer Name	Medicine Requested	Expected Date	Status	Action
Musfirat choudhury	Omeprazole	2026-01-23	Pending	Accept Decline
Musfirat choudhury	Adovas	2026-01-30	Accepted	Accepted
Hello World	Don A 10 mg	2026-01-30	Accepted	Accepted
Musfirat choudhury	Motrin	2026-01-10	Pending	Accept Decline
Hello World	Norium 10 mg	2026-01-31	Pending	Accept Decline
Musfirat choudhury	Rolac 10 mg	2026-01-07	Pending	Accept Decline
Hello World	Rolac 10 mg	2026-01-15	Pending	Accept Decline
Hello World	Tryptin 10 mg	2026-01-06	Declined	Declined

Customer Reviews

Customer	Review
Hello World	After Taking Aspirin my pain didn't decrease

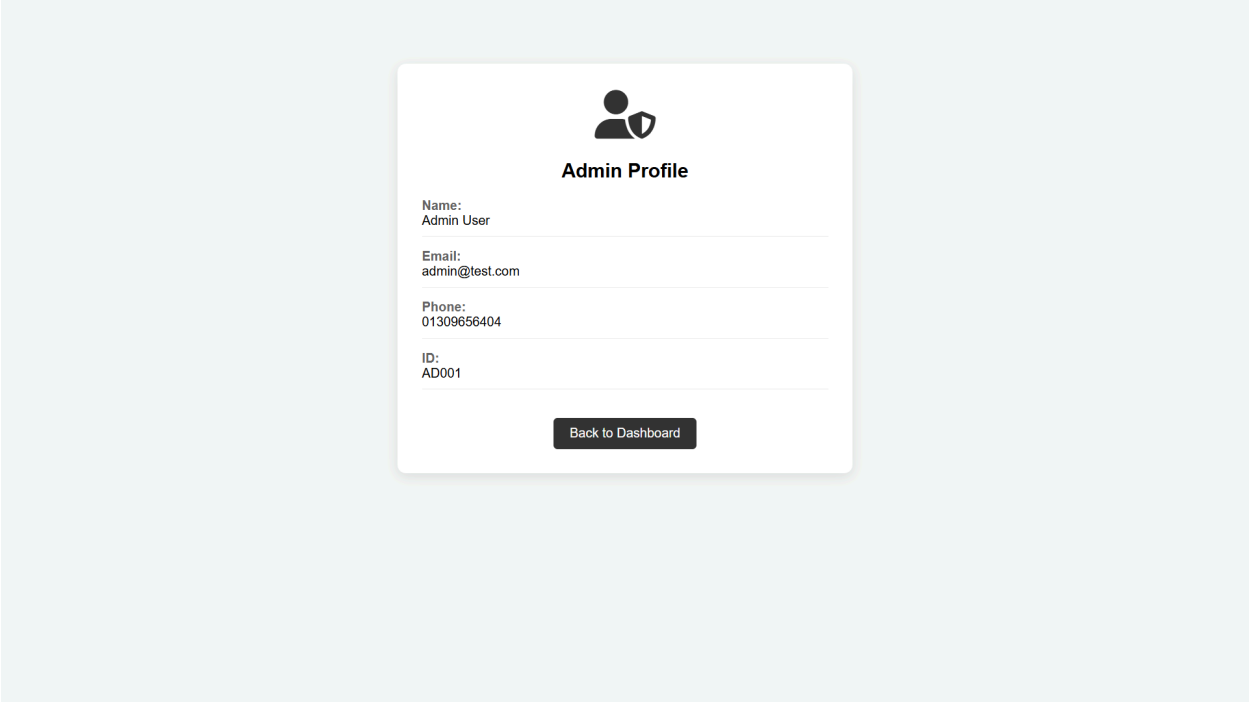
View All Payments

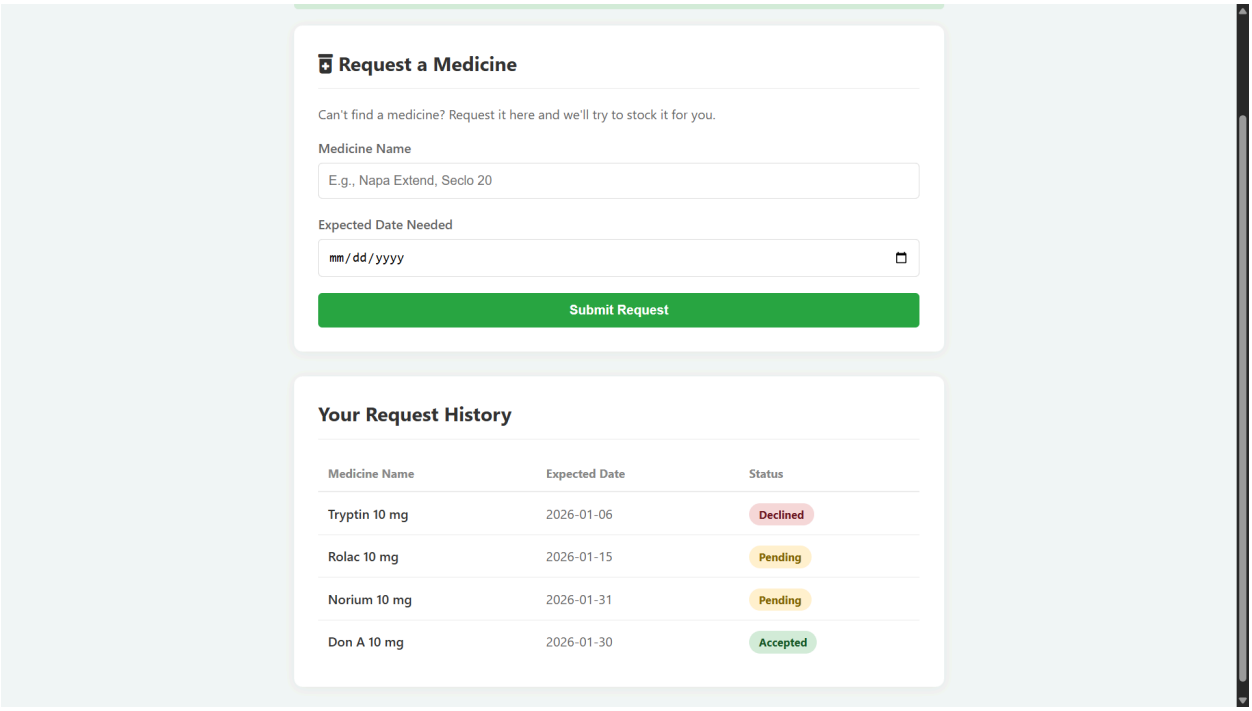
DrugWeb Admin Dashboard Logout

Customer Orders & Payments

← Back to Dashboard

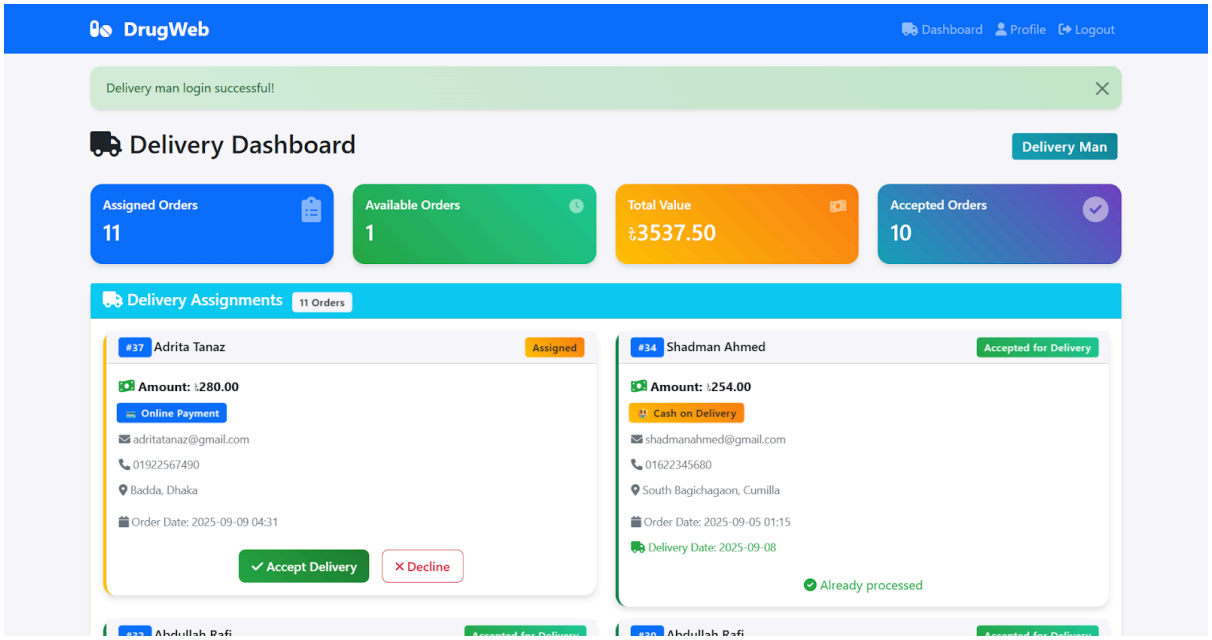
Order ID	Customer	Amount	Payment	Address	Delivery Status	Action
#PAY954658	Hello World 01309656404	৳42.00	Cash on Delivery	Mirpur 12	Pikachu Delivery 01309656404	Select Delivery Man... Assign
#PAY475781	Musfirat choudhury 01309656404	৳27.00	Bkash	Mirpur 13	Pikachu Delivery 01309656404	Select Delivery Man... Assign
#PAY372201	Hello World 01309656404	৳30.00	Cash on Delivery	Mirpur 12	Pikachu Delivery 01309656404	Select Delivery Man... Assign
#PAY277301	Hello World 01309656404	৳14.50	Bkash	Mirpur 12	Unassigned	Select Delivery Man... Assign
#1001	Hello World 01309656404	৳50.00	Cash on Delivery	Mirpur 12	Pikachu Delivery 01309656404	Select Delivery Man... Assign

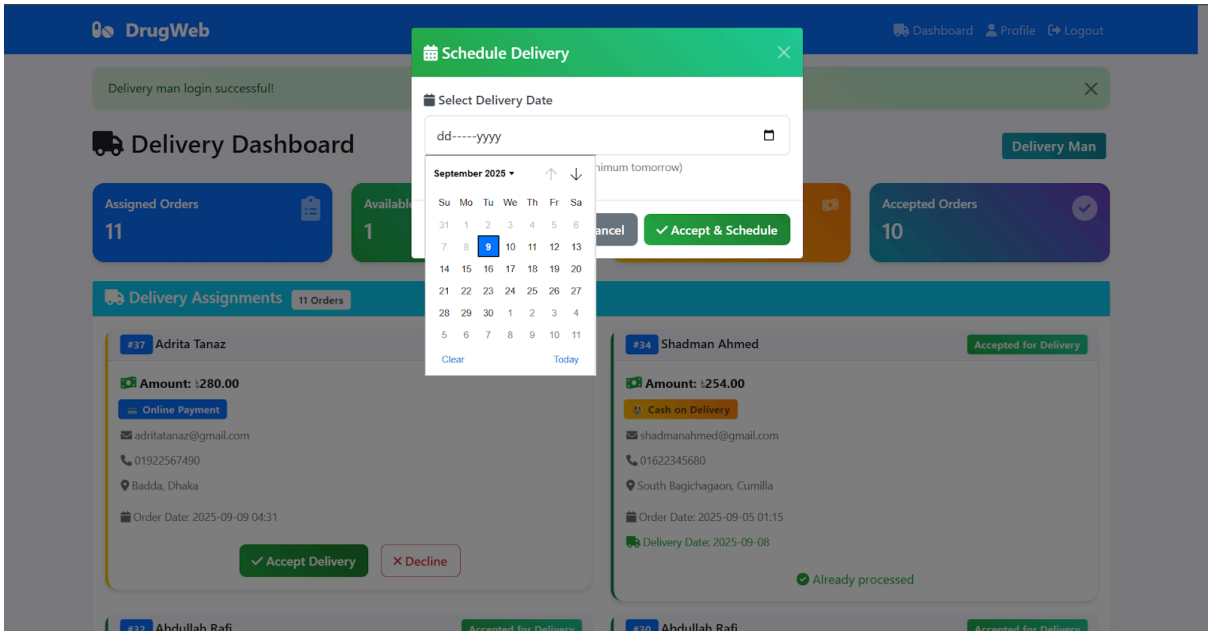




Contribution of ID : 21301269, Name : Mushfirat Chowdhury

- Deliveryman dashboard where he can find his profile option and can see the assigned orders.
- Deliveryman's confirmation of an order with a date.





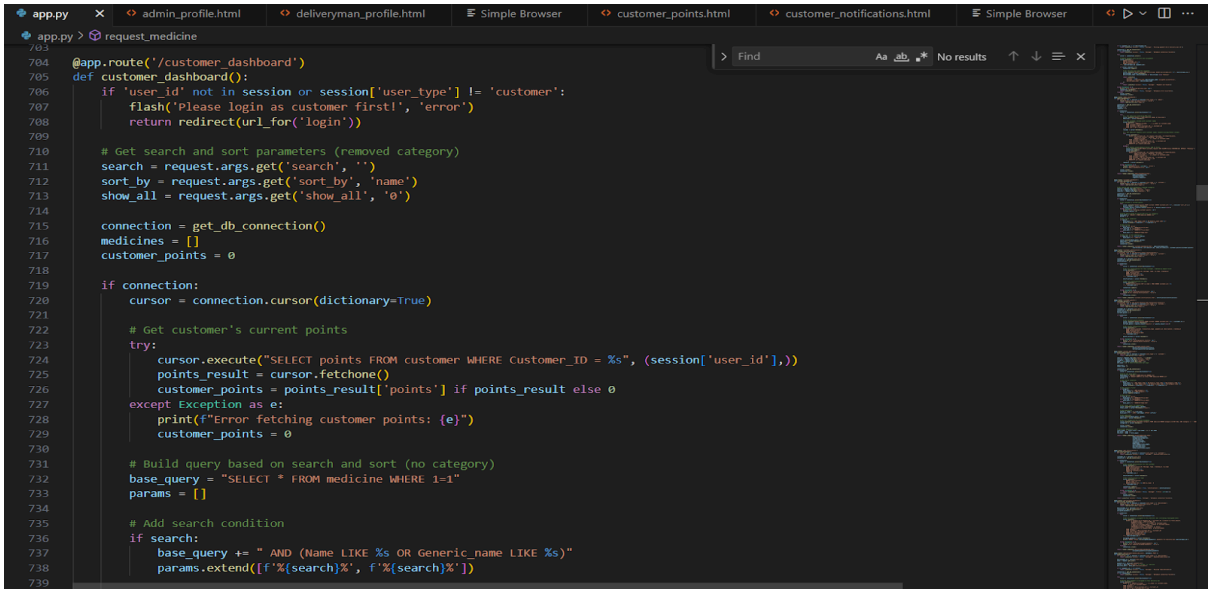
Backend Development

For the backend, we used the FLASK framework of python and MYSQL for handling queries and creating tables in the database.

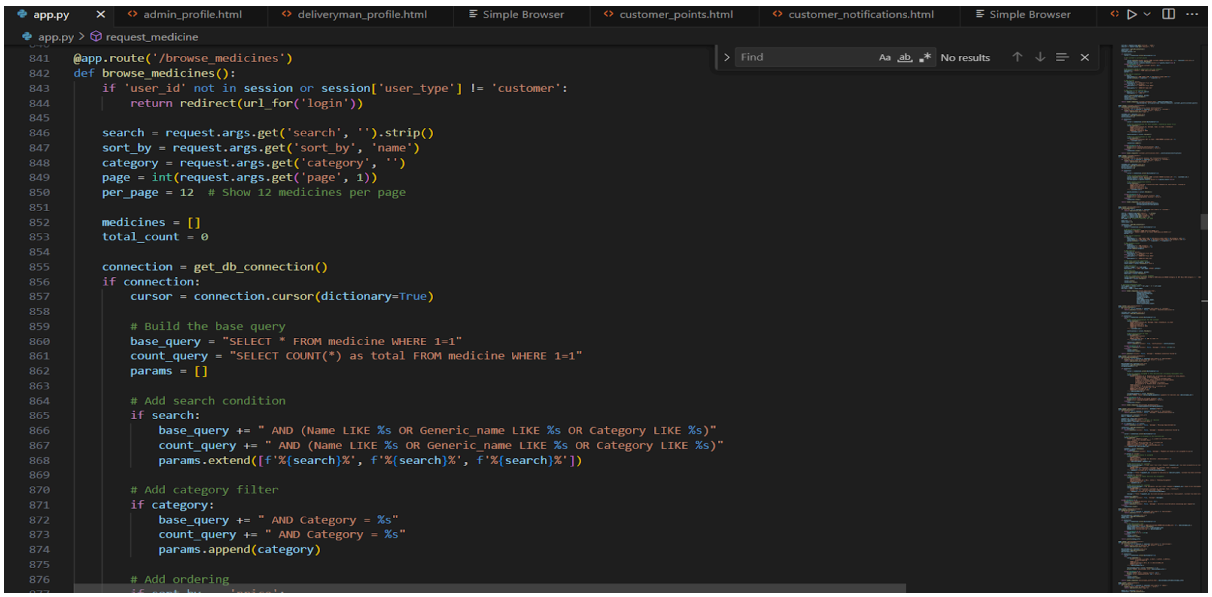
Contribution of ID : 22201512 , NAME: SWAGATA DEY

- Database creation with tables and assigning primary keys and foreign keys.

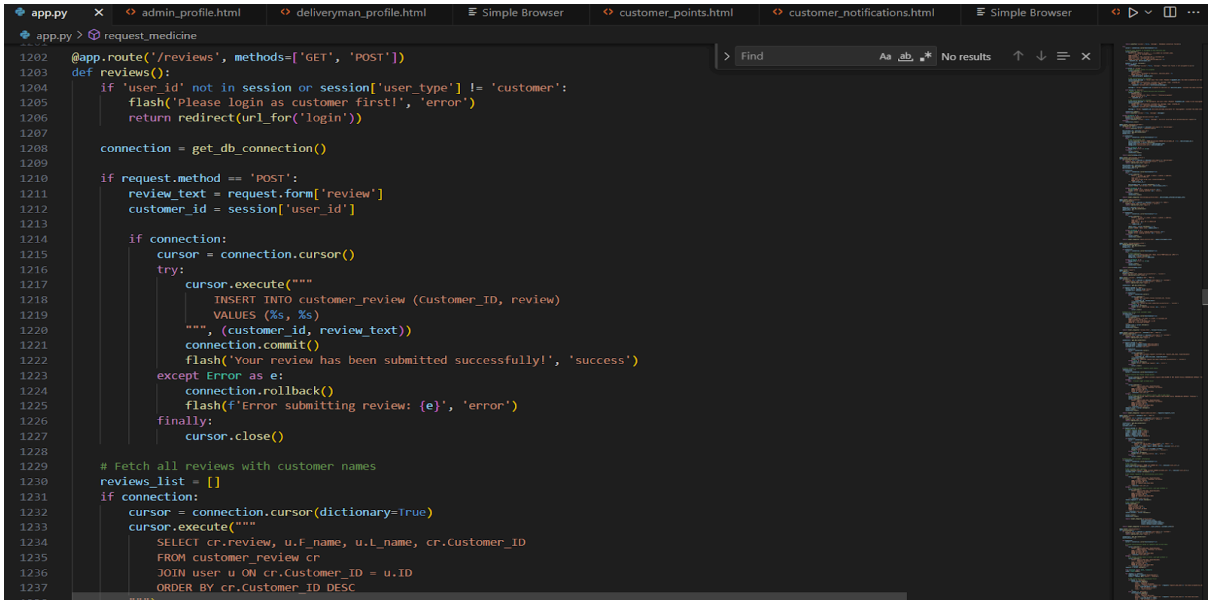
- Customer dashboard where a customer can see reviews and can give reviews, get notifications.
- Search any medicine and can sort by price or name.
- Add to cart and in the cart increase or decrease quantity of the medicines.



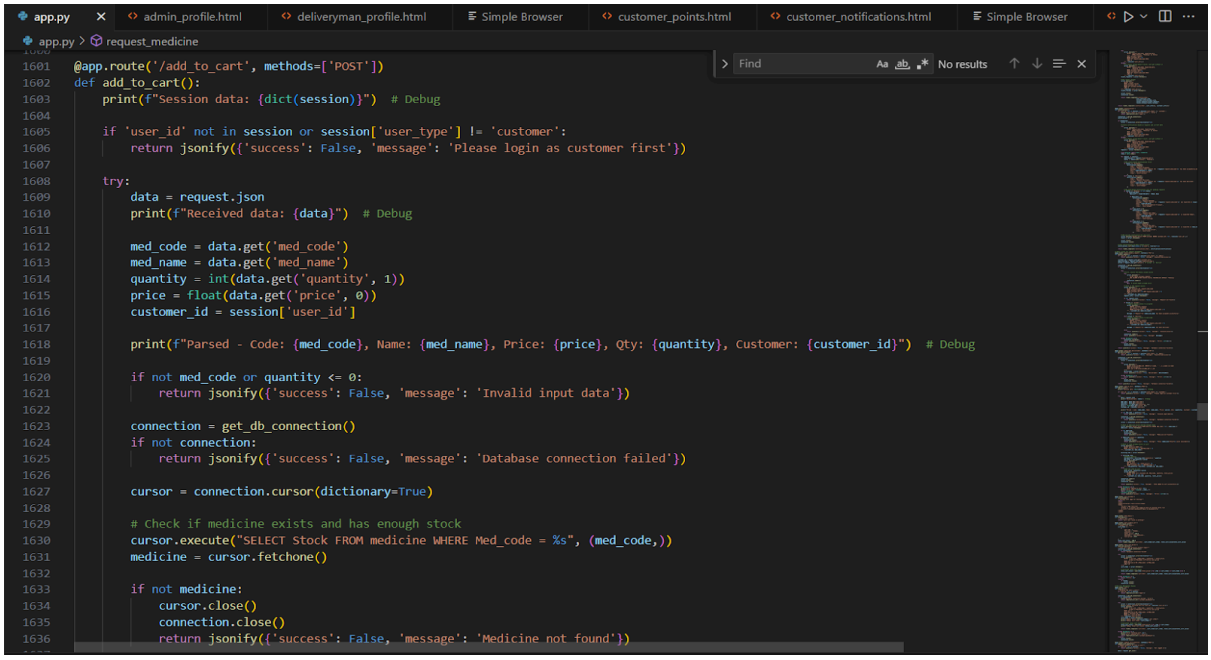
```python
@app.route('/customer_dashboard')
def customer_dashboard():
    if 'user_id' not in session or session['user_type'] != 'customer':
        flash('Please login as customer first!', 'error')
        return redirect(url_for('login'))

    # Get search and sort parameters (removed category)
    search = request.args.get('search', '')
    sort_by = request.args.get('sort_by', 'name')
    show_all = request.args.get('show_all', '0')

    connection = get_db_connection()
    medicines = []
    customer_points = 0

    if connection:
        cursor = connection.cursor(dictionary=True)

        # Get customer's current points
        try:
            cursor.execute("SELECT points FROM customer WHERE Customer_ID = %s", (session['user_id'],))
            points_result = cursor.fetchone()
            customer_points = points_result['points'] if points_result else 0
        except Exception as e:
            print(f"Error fetching customer points: {e}")
            customer_points = 0

        # Build query based on search and sort (no category)
        base_query = "SELECT * FROM medicine WHERE 1=1"
        params = []

        # Add search condition
        if search:
            base_query += " AND (Name LIKE %s OR Generic_name LIKE %s)"
            params.extend([f'%{search}%', f'%{search}%'])
```



```python
@app.route('/browse_medicines')
def browse_medicines():
    if 'user_id' not in session or session['user_type'] != 'customer':
        return redirect(url_for('login'))

    search = request.args.get('search', '').strip()
    sort_by = request.args.get('sort_by', 'name')
    category = request.args.get('category', '')
    page = int(request.args.get('page', 1))
    per_page = 12  # Show 12 medicines per page

    medicines = []
    total_count = 0

    connection = get_db_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)

        # Build the base query
        base_query = "SELECT * FROM medicine WHERE 1=1"
        count_query = "SELECT COUNT(*) as total FROM medicine WHERE 1=1"
        params = []

        # Add search condition
        if search:
            base_query += " AND (Name LIKE %s OR Generic_name LIKE %s OR Category LIKE %s)"
            count_query += " AND (Name LIKE %s OR Generic_name LIKE %s OR Category LIKE %s)"
            params.extend([f'%{search}%', f'%{search}%', f'%{search}%'])

        # Add category filter
        if category:
            base_query += " AND Category = %s"
            count_query += " AND Category = %s"
            params.append(category)

        # Add ordering
```



```python
@app.route('/reviews', methods=['GET', 'POST'])
def reviews():
    if 'user_id' not in session or session['user_type'] != 'customer':
        flash('Please login as customer first!', 'error')
        return redirect(url_for('login'))

    connection = get_db_connection()

    if request.method == 'POST':
        review_text = request.form['review']
        customer_id = session['user_id']

        if connection:
            cursor = connection.cursor()
            try:
                cursor.execute("""
                    INSERT INTO customer_review (Customer_ID, review)
                    VALUES (%s, %s)
                """, (customer_id, review_text))
                connection.commit()
                flash('Your review has been submitted successfully!', 'success')
            except Error as e:
                connection.rollback()
                flash(f'Error submitting review: {e}', 'error')
            finally:
                cursor.close()

    # Fetch all reviews with customer names
    reviews_list = []
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT cr.review, u.F_name, u.L_name, cr.Customer_ID
            FROM customer_review cr
            JOIN user u ON cr.Customer_ID = u.ID
            ORDER BY cr.Customer_ID DESC
```



```python
@app.route('/add_to_cart', methods=['POST'])
def add_to_cart():
    print(f"Session data: {dict(session)}")  # Debug

    if 'user_id' not in session or session['user_type'] != 'customer':
        return jsonify({'success': False, 'message': 'Please login as customer first'})

    try:
        data = request.json
        print(f"Received data: {data}")  # Debug

        med_code = data.get('med_code')
        med_name = data.get('med_name')
        quantity = int(data.get('quantity', 1))
        price = float(data.get('price', 0))
        customer_id = session['user_id']

        print(f"Parsed - Code: {med_code}, Name: {med_name}, Price: {price}, Qty: {quantity}, Customer: {customer_id}")  # Debug

        if not med_code or quantity <= 0:
            return jsonify({'success': False, 'message': 'Invalid input data'})

        connection = get_db_connection()
        if not connection:
            return jsonify({'success': False, 'message': 'Database connection failed'})

        cursor = connection.cursor(dictionary=True)

        # Check if medicine exists and has enough stock
        cursor.execute("SELECT Stock FROM medicine WHERE Med_code = %s", (med_code,))
        medicine = cursor.fetchone()

        if not medicine:
            cursor.close()
            connection.close()
            return jsonify({'success': False, 'message': 'Medicine not found'})
```

Contribution of ID : 22301298, NAME: SAJID MUHAMMOD

- **Admin Dashboard Logic:** Wrote the code to fetch and display the latest medicine stock, reviews, and requests on the admin dashboard.
- **Request Handling:** Implemented the functionality that allows the admin to update a request's status to "Accepted" or "Declined" in the database.
- **Delivery Assignment:** Created the backend logic that saves the assigned deliveryman to a specific customer order.
- **Customer Request Logic:** Wrote the code to save the medicine name and expected date when a customer submits a new request.

app.py | admin_profile.html | deliveryman_profile.html | Simple Browser | customer_points.html | customer_notifications.html | Simple Browser

app.py > request_medicine

Find | No results

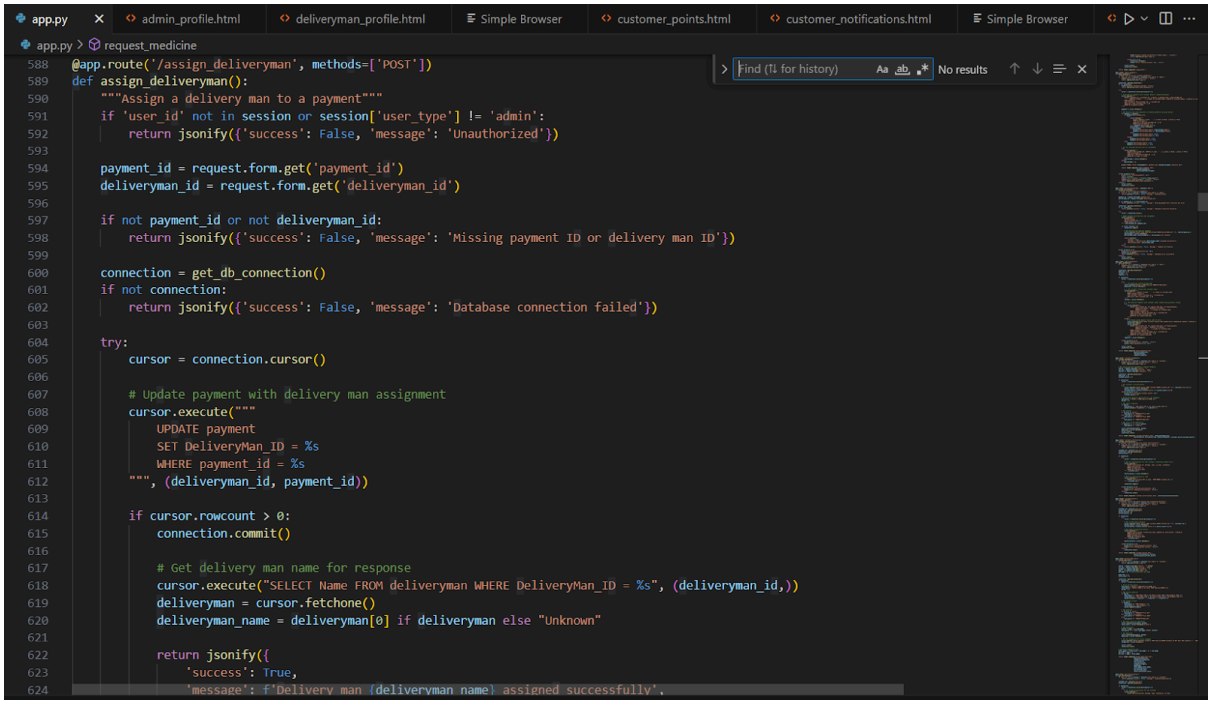
```python
@app.route('/admin_dashboard')
def admin_dashboard():
    if 'user_id' not in session or session['user_type'] != 'admin':
        flash('Please login as admin first!', 'error')
        return redirect(url_for('login'))

    connection = get_db_connection()
    medicines = []
    reviews = []
    requests = []

    if connection:
        cursor = connection.cursor(dictionary=True)

        try:
            # 1. Get medicines ordered by Med_Code
            cursor.execute("SELECT * FROM medicine ORDER BY Med_Code")
            medicines = cursor.fetchall()

            # 2. Get customer reviews with customer names
            cursor.execute("""
                SELECT cr.*, CONCAT(u.F_name, ' ', u.L_name) as customer_name
                FROM customer_review cr
                JOIN customer c ON cr.Customer_ID = c.Customer_ID
                JOIN user u ON c.Customer_ID = u.ID
            """)
            reviews = cursor.fetchall()

            # 3. Get medicine requests with customer names (handle missing Status column)
            try:
                cursor.execute("""
                    SELECT cmr.Customer_ID, cmr.request_med_name, cmr.Expected_date,
                           IFNULL(cmr.Status, 'Pending') as Status,
                           CONCAT(u.F_name, ' ', u.L_name) as customer_name
                    FROM customer_request cmr
                    JOIN customer c ON cmr.Customer_ID = c.Customer_ID
                    JOIN user u ON c.Customer_ID = u.ID
```

app.py | admin_profile.html | deliveryman_profile.html | Simple Browser | customer_points.html | customer_notifications.html | Simple Browser

app.py > request_medicine

Find | No results

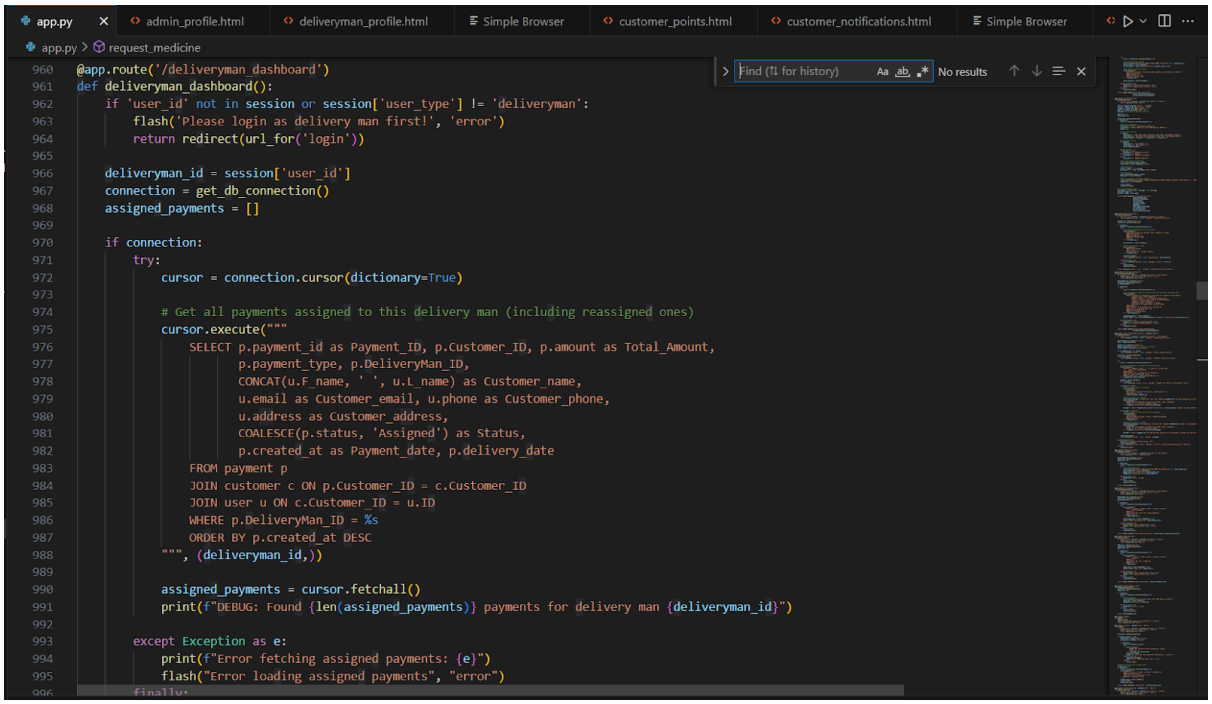
```python
@app.route('/request_medicine', methods=['GET', 'POST'])
def request_medicine():
    if 'user_id' not in session or session['user_type'] != 'customer':
        flash('Please login as customer first!', 'error')
        return redirect(url_for('login'))

    connection = get_db_connection()

    if request.method == 'POST':
        medicine_name = request.form['medicine_name']
        expected_date = request.form['expected_date']
        customer_id = session['user_id']

        if connection:
            cursor = connection.cursor()
            try:
                cursor.execute("""
                    INSERT INTO customer_request (Customer_ID, request_med_name, Expected_date)
                    VALUES (%s, %s, %s)
                """, (customer_id, medicine_name, expected_date))
                connection.commit()
                flash('Your medicine request has been submitted successfully!', 'success')
            except Error as e:
                connection.rollback()
                flash(f'Error submitting request: {e}', 'error')
            finally:
                cursor.close()

    # Fetch customer's previous requests with status
    requests_list = []
    if connection:
        cursor = connection.cursor(dictionary=True)

        # First ensure the Status column exists
        try:
            cursor.execute("ALTER TABLE customer_request ADD COLUMN IF NOT EXISTS Status VARCHAR(20) DEFAULT 'Pending'")
```



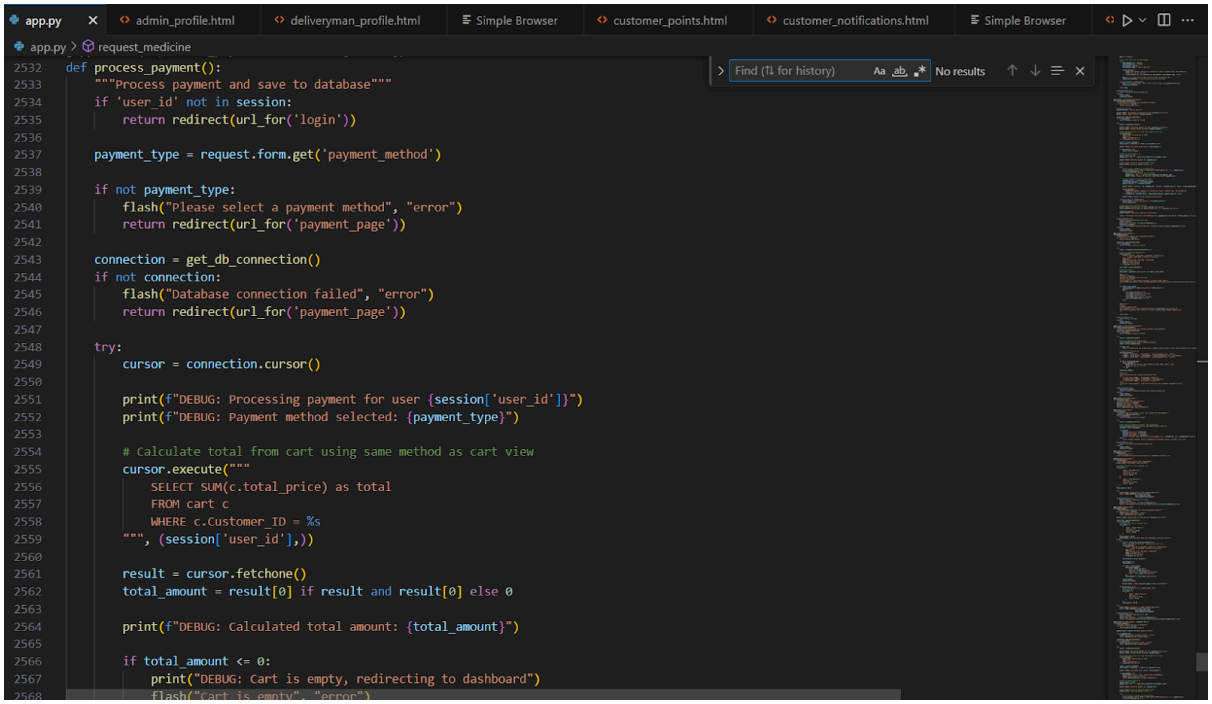
```python
@app.route('/assign_deliveryman', methods=['POST'])
def assign_deliveryman():
    """Assign a delivery man to a payment"""
    if 'user_id' not in session or session['user_type'] != 'admin':
        return jsonify({'success': False, 'message': 'Unauthorized'})

    payment_id = request.form.get('payment_id')
    deliveryman_id = request.form.get('deliveryman_id')

    if not payment_id or not deliveryman_id:
        return jsonify({'success': False, 'message': 'Missing payment ID or delivery man ID'})

    connection = get_db_connection()
    if not connection:
        return jsonify({'success': False, 'message': 'Database connection failed'})

    try:
        cursor = connection.cursor()

        # Update payment with delivery man assignment
        cursor.execute("""
            UPDATE payment
            SET DeliveryMan_ID = %s
            WHERE payment_id = %s
        """, (deliveryman_id, payment_id))

        if cursor.rowcount > 0:
            connection.commit()

            # Get delivery man name for response
            cursor.execute("SELECT Name FROM deliveryman WHERE DeliveryMan_ID = %s", (deliveryman_id,))
            deliveryman = cursor.fetchone()
            deliveryman_name = deliveryman[0] if deliveryman else "Unknown"

            return jsonify({
                'success': True,
                'message': f'Delivery man {deliveryman_name} assigned successfully',
```

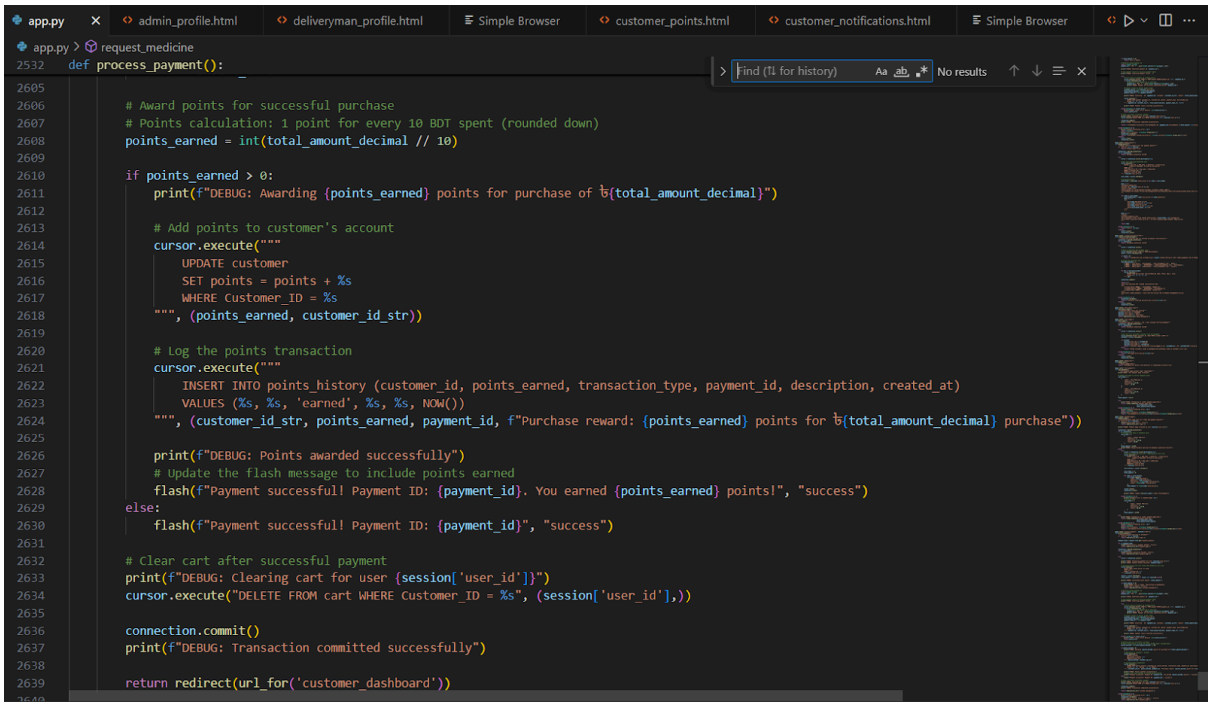
Contribution of ID : 21301269, Name : Mushfirat Chowdhury

- Deliveryman dashboard with accept or decline payment option.
- Customer payment procedure.
- Customer earn point feature according to customer's total payment.



```python
@app.route('/deliveryman_dashboard')
def deliveryman_dashboard():
    if 'user_id' not in session or session['user_type'] != 'deliveryman':
        flash('Please login as delivery man first!', 'error')
        return redirect(url_for('login'))

    deliveryman_id = session['user_id']
    connection = get_db_connection()
    assigned_payments = []

    if connection:
        try:
            cursor = connection.cursor(dictionary=True)

            # Get all payments assigned to this delivery man (including reassigned ones)
            cursor.execute("""
                SELECT p.payment_id as Payment_ID, p.Customer_ID, p.amount as Total_Amount,
                       p.payment_type, p.DeliveryMan_ID,
                       CONCAT(u.F_name, ' ', u.L_name) as Customer_name,
                       u.email as Customer_email, u.phone as Customer_phone,
                       u.address as Customer_address,
                       COALESCE(p.status, 'Assigned') as Status,
                       p.created_at as Payment_date, p.delivery_date
                FROM payment p
                JOIN customer c ON p.Customer_ID = c.Customer_ID
                JOIN user u ON c.Customer_ID = u.ID
                WHERE p.DeliveryMan_ID = %s
                ORDER BY p.created_at DESC
            """, (deliveryman_id,))

            assigned_payments = cursor.fetchall()
            print(f"DEBUG: Found {len(assigned_payments)} payments for delivery man {deliveryman_id}")

        except Exception as e:
            print(f"Error fetching assigned payments: {e}")
            flash("Error loading assigned payments", "error")
        finally:
```


app.py
admin_profile.html
deliveryman_profile.html
Simple Browser
customer_points.html
customer_notifications.html
Simple Browser
app.py > request_medicine
Find (⇅ for history)
Aa
ab
.*
No results
def process_payment():
"""Process payment and save to database"""
if 'user_id' not in session:
return redirect(url_for('login'))
payment_type = request.form.get('payment_method')
if not payment_type:
flash("Please select a payment method", "error")
return redirect(url_for('payment_page'))
connection = get_db_connection()
if not connection:
flash("Database connection failed", "error")
return redirect(url_for('payment_page'))
try:
cursor = connection.cursor()
print(f"DEBUG: Processing payment for user {session['user_id']}")
print(f"DEBUG: Payment method selected: {payment_type}")
# Calculate total from cart using same method as cart view
cursor.execute("""
SELECT SUM(c.total_price) as total
FROM cart c
WHERE c.Customer_ID = %s
""", (session['user_id'],))
result = cursor.fetchone()
total_amount = result[0] if result and result[0] else 0
print(f"DEBUG: Calculated total amount: {total_amount}")
if total_amount <= 0:
print("DEBUG: Cart is empty, redirecting to dashboard")
flash("Cart is empty", "error")


app.py
admin_profile.html
deliveryman_profile.html
Simple Browser
customer_points.html
customer_notifications.html
Simple Browser
app.py > request_medicine
def process_payment():
Find (⇅ for history)
Aa
ab
.*
No results
# Award points for successful purchase
# Points calculation: 1 point for every 10 BDT spent (rounded down)
points_earned = int(total_amount_decimal // 10)
if points_earned > 0:
print(f"DEBUG: Awarding {points_earned} points for purchase of ৳{total_amount_decimal}")
# Add points to customer's account
cursor.execute("""
UPDATE customer
SET points = points + %s
WHERE Customer_ID = %s
""", (points_earned, customer_id_str))
Log the points transaction
cursor.execute("""
INSERT INTO points_history (customer_id, points_earned, transaction_type, payment_id, description, created_at)
VALUES (%s, %s, 'earned', %s, %s, NOW())
""", (customer_id_str, points_earned, payment_id, f"Purchase reward: {points_earned} points for ৳{total_amount_decimal} purchase"))
print(f"DEBUG: Points awarded successfully")
# Update the flash message to include points earned
flash(f"Payment successful! Payment ID: {payment_id}. You earned {points_earned} points!", "success")
else:
flash(f"Payment successful! Payment ID: {payment_id}", "success")
# Clear cart after successful payment
print(f"DEBUG: Clearing cart for user {session['user_id']}")
cursor.execute("DELETE FROM cart WHERE Customer_ID = %s", (session['user_id'],))
connection.commit()
print(f"DEBUG: Transaction committed successfully")
return redirect(url_for('customer_dashboard'))

Source Code Repository

https://github.com/portfoliosajid/cse370

Conclusion

In conclusion, our database based website DrugWeb is a place where customers can easily search for medicines, buy medicines, give their reviews and request medicine according to their needs. Additionally, in future one thing which we want to improve in our website is the point system. Now customers can only earn points but we want to upgrade the system such that customers can have the option to redeem their points while paying. Lastly, the most common issue we faced during the backend development is not giving the proper mysql queries which created issues connecting with the database.

References

https://youtu.be/eDTcSaZcLh4?si=Anr2ggT4MzWl65mx
(this video helped to understand flask framework and how to make a proper e-commerce website)

https://www.w3schools.com/MySQL/default.asp
(This website helped to implement the mysql queries properly)