CSE370 : Database Systems
Project Report
Project Title : Gotta Catch'em All

Group No : 02, CSE370 Lab Section : 11, Summer 2025		
ID	**Name**	**Contribution**
23301293	Zarin Tasnim	ER Diagram, Schema Diagram, Database, Daily Reward, Battle Queue, Battle with Cards, Balance Transfer, Notification, Card List, Wishlist
23301049	Mehruma Mahmud	ER Diagram, Schema Diagram, Database, Auction, Trade, Chat System (Frontend and backend)
23301105	Sabbir Ahmed Kabbo	ER Diagram, Schema Diagram

Table of Contents

Section No	Content	Page No
1	Introduction	3
2	Project Features	3
3	ER/EER Diagram	4
4	Schema Diagram	5
5	Normalization	6
6	Frontend Development	6
7	Backend Development	30
8	Source Code Repository	51
9	Conclusion	51
10	References	51

Introduction

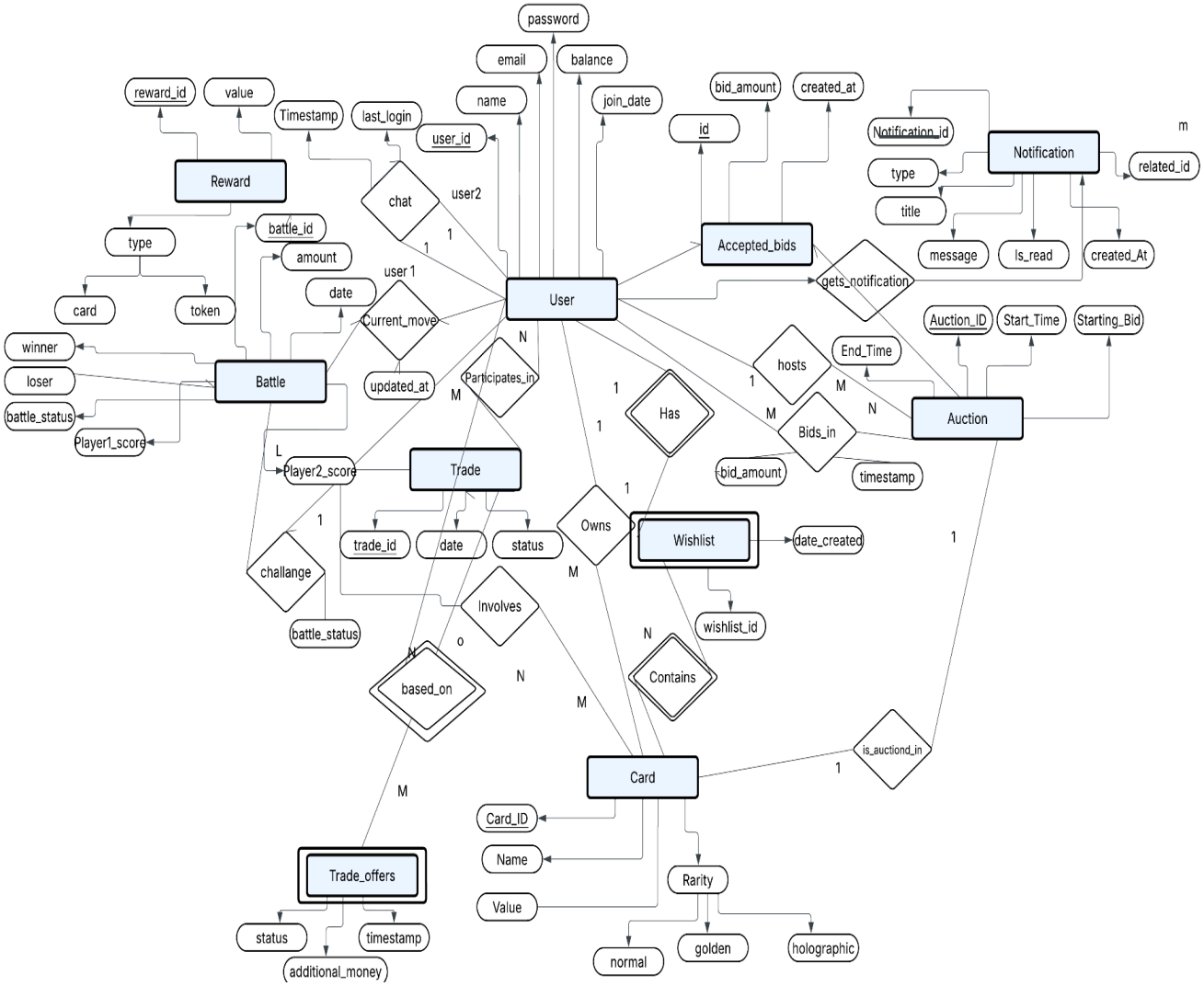
The core idea of the project “Gotta Catch'em All" is to create a safe as well as fun environment for the Pokemon fanatics where they can exchange their pokemon cards to likeminded people through battling, auctioning, and trading. The exchange can be done for other pokemon cards or in-site balance. The interaction is made more enjoyable and comfortable by implementing a private chat system, auto matching for battles, reward systems (daily login bonus, battle winner bonus), and wishlist system.

To make the website engagement enjoyable, the interface is kept simple with a crimson and black theme, Pokemon card displays with image and other details, game-like activities, and progression tracker.

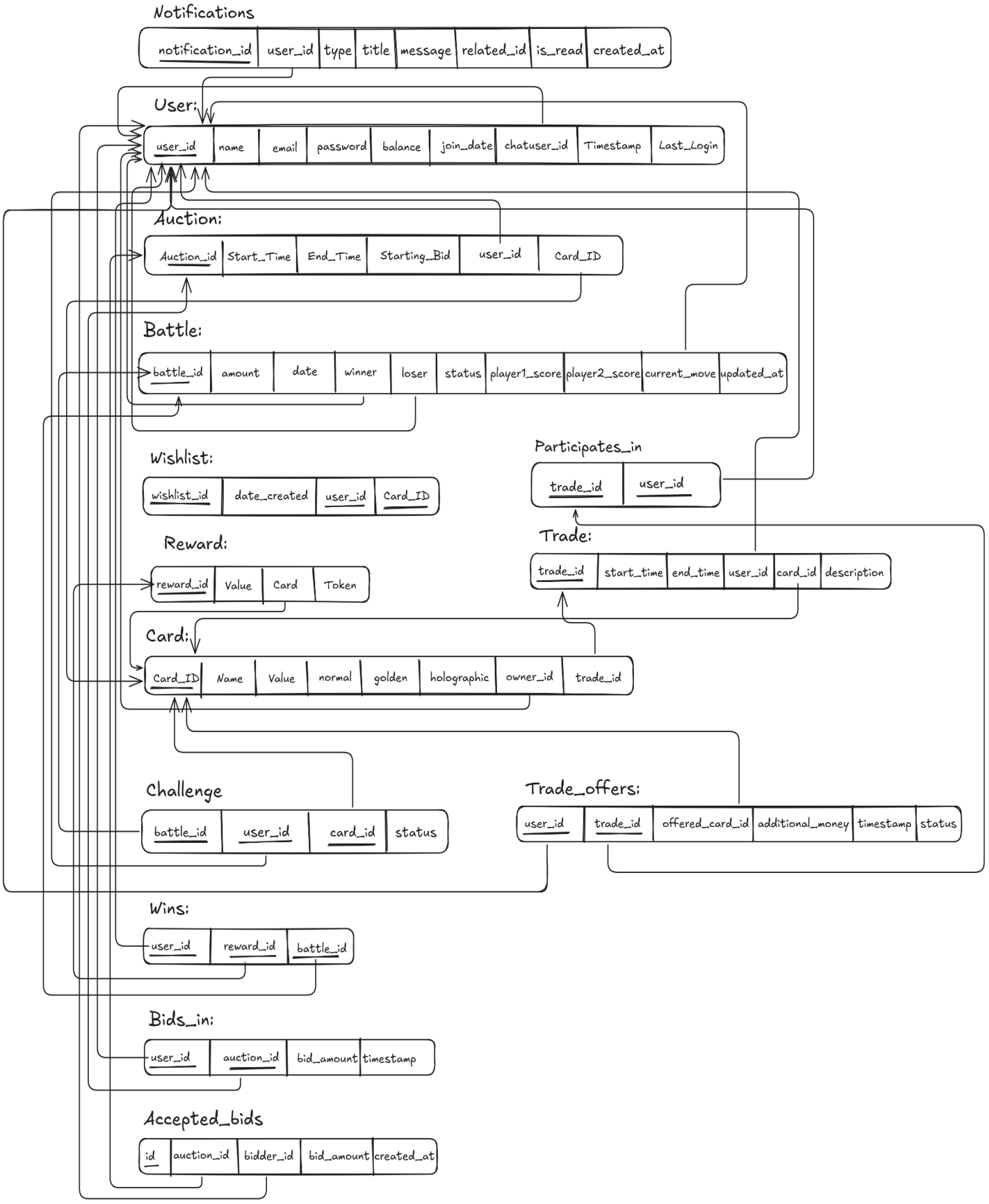
Project Features

ID, Name	Features	
23301293, Zarin Tasnim	Ft 1	Daily Login Reward for user
	Ft 2	Balance Transfer After Auction and Trade
	Ft 3	Adding Cards to user’s collection
	Ft 4	Adding and Removing Card from user’s wishlist
	Ft 5	Auto-matching from battle queue
	Ft 6	Battle System for two users to battle with card, Battle History
	Ft 7	Notification for Auction and Trade and Battle after completion
23301049, Mehruma Mahmud	Ft 1	Private chat system between users
	Ft 2	Create new auctions and trades using the cards from my cards
	Ft 3	Trade cards via in-site balance or other cards
	Ft 4	Auction cards, accept the highest bid
	Ft 5	A separate tab to view all the active personal Auction and trades
	Ft 6	Dynamic Dashboard with quick access to the features
23301105, Sabbir Ahmed Kabbo	Ft 1	
	Ft 2	

ER/EER Diagram



Schema Diagram



Normalization

The schema is already in 3NF. No conversion needed. There exists neither any Partial Dependencies nor any Transitive dependencies in my 3NF relational schema diagram.

Frontend Development

We used **HTML**, **CSS**, **Bootstrap** and **Javascript** for our frontend development of our project for Pokemon Cards Trading System named Gotta Catch'em All. We used a Pokemon game night image from online for our background and maroon red as our theme for the website.

Below, we briefly discuss each of our contributions in frontend development.

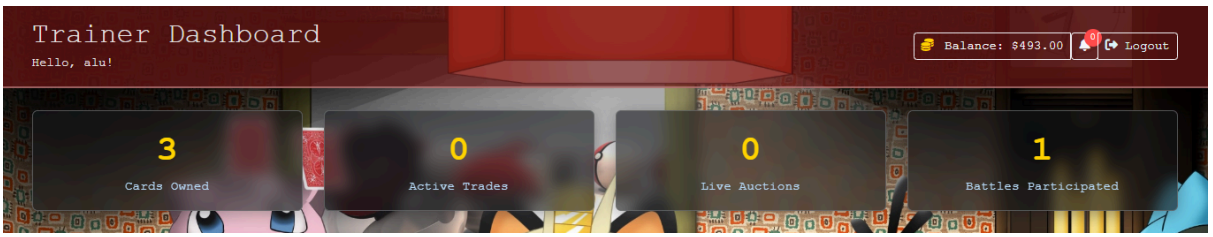
Contribution of ID : 23301293, Name : Zarin Tasnim

Overview: I was responsible for a part of the dashboard, card adding feature, adding card to wishlist feature, battle system, balance transfer system and notification. I have added some of the frontend code views for the features I have done.

For the **Frontend**, I used **HTML**, **CSS** and **Bootstrap** for this project.
Dashboard:

For the dashboard, I added a container class for stats update. It shows the number of cards owned by the user, number of active trades, number of active auctions and the number of battles participated by the user.

Frontend view:



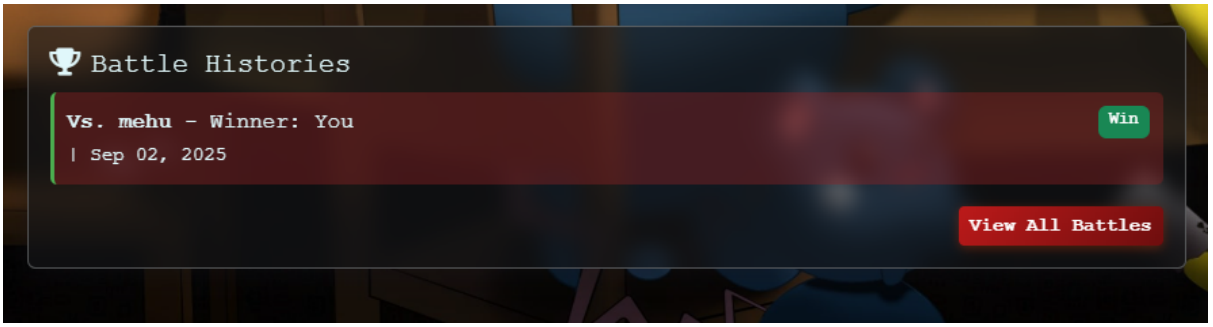
Code Snippet:

```
<div class="container">
  <!-- Stats Overview -->
  <div class="row mb-4">
    <div class="col-md-3">
      <div class="card stat-card">
        <div class="stat-number">{{ cards }}</div>
        <div class="stat-label">Cards Owned</div>
      </div>
    </div>
    <div class="col-md-3">
      <div class="card stat-card">
        <div class="stat-number">{{ trades }}</div>
        <div class="stat-label">Active Trades</div>
      </div>
    </div>
    <div class="col-md-3">
      <div class="card stat-card">
        <div class="stat-number">{{ live_auctions }}</div>
        <div class="stat-label">Live Auctions</div>
      </div>
    </div>
    <div class="col-md-3">
      <div class="card stat-card">
        <div class="stat-number">{{ battles_count }}</div>
        <div class="stat-label">Battles Participated</div>
      </div>
    </div>
  </div>
```

Battle History:

I also added a battle history preview that shows the latest battles fought by the user. I added a button ‘View All Battles’ that redirects users to the Battle page.

Frontend view:



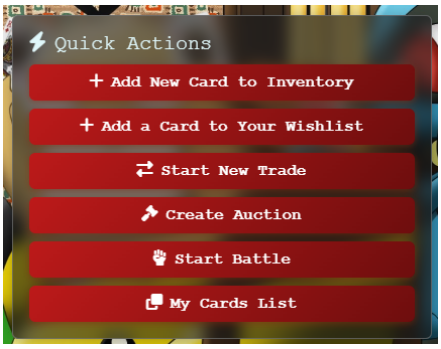
Code Snippet:

```
<!-- Battle Histories -->
<div class="card">
  <div class="card-body">
    <h5 class="card-title">
      <i class="fas fa-trophy me-2"></i>Battle Histories
    </h5>
    {% for battle in battles %}
    <div class="battle-item {% if battle.result == 'Win' %}battle-win{% else %}battle-loss{% endif %}">
      <div class="d-flex justify-content-between">
        <div>
          <strong>Vs. {{ battle.opponent }}</strong> - Winner: {% if
          battle.result == 'Win' %}You{% else %}{{ battle.opponent
          }}{% endif %}
        </div>
        <span class="badge {% if battle.result == 'Win' %}bg-success{% else %}bg-danger{% endif %}">
          {{ battle.result }}
        </span>
      </div>
      <small>
        {% if battle.prize %}Prize: ${{ battle.prize }}{% elif
        battle.stake %}Stake: ${{ battle.stake }}{% endif %} | {{
        battle.date.strftime('%b %d, %Y') }}
      </small>
    </div>
    {% endfor %}

    <div class="text-end mt-3">
      <form action="{{ url_for('battle_history') }}" method="get" class="text-end">
        <button type="submit" class="btn btn-primary btn-sm">
          View All Battles
        </button>
      </form>
    </div>
  </div>
</div>
</div>
```

Quick Actions:

I added buttons ‘My Card List’, ‘Start Battle’, ‘Add New Card to Inventory’ and ‘Add a Card to Your Wishlist’ and they redirect to different pages such as Card page, Battle Page and Wishlist page.

Frontend view:



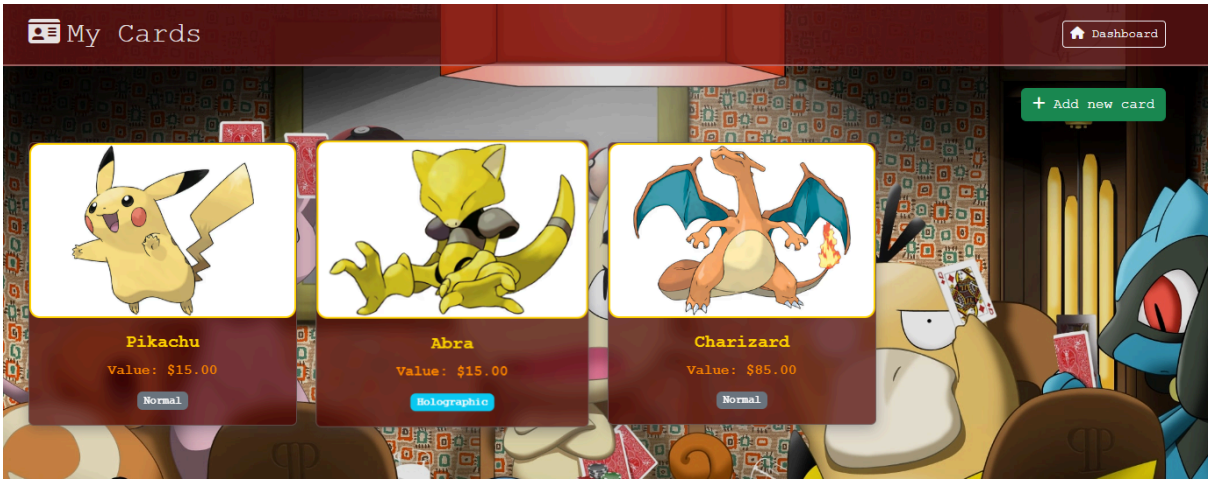
Code Snippet:

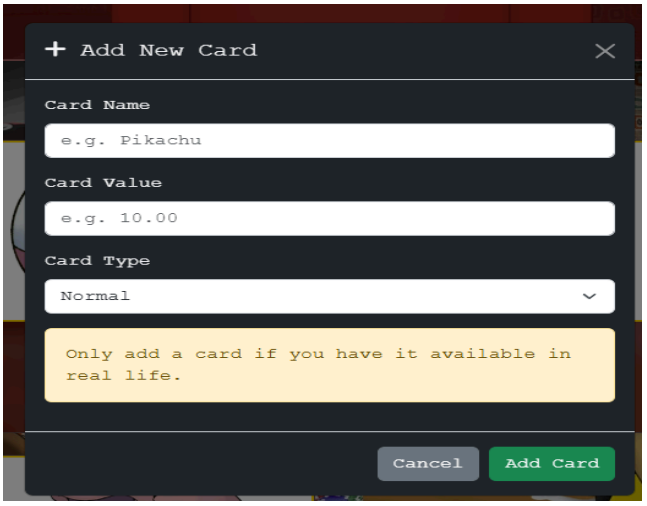
```
<!-- Quick Actions -->
<div class="card">
  <div class="card-body">
    <h5 class="card-title">
      <i class="fas fa-bolt me-2"></i>Quick Actions
    </h5>
    <div class="d-grid gap-2">
      <a href="{{ url_for('my_cards') }}" class="btn btn-primary">
        <i class="fas fa-plus me-2"></i>Add New Card to Inventory
      </a>
      <a href="{{ url_for('wishlist') }}" class="btn btn-primary">
        <i class="fas fa-plus me-2"></i>Add a Card to Your Wishlist
      </a>
      <a href="{{ url_for('market') }}#trades" class="btn btn-primary">
        <i class="fas fa-exchange-alt me-2"></i>Start New Trade
      </a>
      <a href="{{ url_for('market') }}#auctions" class="btn btn-primary">
        <i class="fas fa-gavel me-2"></i>Create Auction
      </a>
      <a href="{{ url_for('battle_history') }}" class="btn btn-primary">
        <i class="fas fa-fist-raised me-2"></i>Start Battle
      </a>
      <a href="{{ url_for('my_cards') }}" class="btn btn-primary">
        <i class="fas fa-clone me-2"></i>My Cards List
      </a>
    </div>
  </div>
</div>
```

Card List and Wishlist:

I made the My Cards page and the Wishlist Page. In my card page, cards owned by the user will appear. In the Wishlist page, cards wishlisted by the users will appear. They both fetch the pictures from an online pokemon website and use the same approach based on the name given by the user while adding. I added buttons for both pages to add cards. ‘Add Card’ button will let the user add Pokemon cards to his collection.

Frontend view:





Code Snippet:

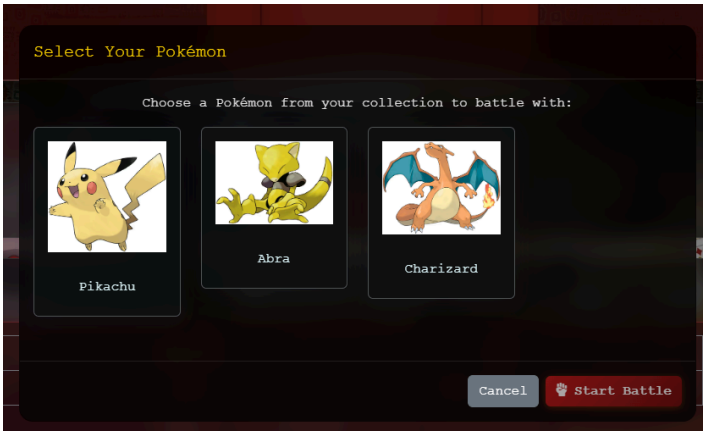
```
<body>
    <div class="modal fade" id="addCardModal" tabindex="-1" aria-labelledby="addCardModalLabel" aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content bg-dark text-light">
                <form method="POST" action="{{ url_for('add_card') }}">
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cancel</button>
                        <button type="submit" class="btn btn-success">Add Card</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <!-- Cards List -->
    <div class="container">
        {% if user_cards %}
        <div class="row">
            {% for card in user_cards %}
            <div class="col-md-3 col-sm-6 mb-4">
                <div class="card h-100 text-center">
                    <div class="pokemon-box">
                        <img src="https://img.pokemondb.net/artwork/avif/{{ card.name|lower }}.avif"
                            alt="{{ card.name|capitalize }}"
                            onerror="this.onerror=null; this.src='https://img.pokemondb.net/artwork/avif/unknown.avif';">
                    </div>
                    <div class="card-body">
                        <h5 class="card-title">{{ card.name|capitalize }}</h5>
                        <p class="card-value mb-1">Value: ${{ "%.2f"|format(card.value) }}</p>

                        {% if card.golden %}
                        <span class="badge bg-warning text-dark">Golden</span>
                        {% elif card.holographic %}
                        <span class="badge bg-info">Holographic</span>
                        {% else %}
                        <span class="badge bg-secondary">Normal</span>
                        {% endif %}
                    </div>
                </div>
            </div>
            {% endfor %}
        </div>
```

Battle Arena:

I made a battle page where users can participate in battle and see the user's battle history. To participate in the battle, the user needs to choose a card from his card list and wait in a queue

for another user to join. When the battle starts, there are buttons for ‘Attack’, ‘Defend’ and ‘Special’ attack.

Frontend view:



Code Snippet:

Battle Queue:

```
    {% elif in_queue %}
    <!-- Queue Status Panel -->
    <div class="card shadow-sm mb-4">
        <div class="card-body text-center">
            <h3 class="card-title mb-3"><i class="fas fa-clock me-2"></i>Waiting for Opponent</h3>
            <div class="queue-status">
                <div class="spinner text-warning me-2"></div>
                <span>You are in the battle queue. Waiting for another player to join...</span>
            </div>
            <form action="{{ url_for('cancel_queue') }}" method="post">
                <button type="submit" class="btn btn-primary mt-3">
                    <i class="fas fa-times me-2"></i>Cancel Queue
                </button>
            </form>
        </div>
    </div>
    {% else %}
    <!-- No Battle Panel -->
    <div class="card shadow-sm mb-4">
        <div class="card-body text-center">
            <h3 class="card-title mb-3"><i class="fas fa-fist-raised me-2"></i>No Active Battle</h3>
            <p class="text-muted">You don't have any ongoing battles at the moment.</p>
            <button type="button" class="btn btn-primary" data-bs-toggle="modal"
                data-bs-target="#pokemonSelectModal">
                <i class="fas fa-plus me-2"></i>Start New Battle
            </button>
        </div>
    </div>
    {% endif %}
```

Battle History:

```
<!-- Battle History Panel -->
{% if battles %}
<div class="card shadow-sm mb-4">
    <div class="card-body">
        <h3 class="card-title text-center mb-4"><i class="fas fa-trophy me-2"></i>Battle History</h3>

        <div class="table-responsive red-blur-table">
            <table class="table table-hover table-bordered align-middle">
                <thead>
                    <tr class="text-center">
                        <th scope="col" style="background: transparent; color: #ffd600;">Battle ID</th>
                        <th scope="col" style="background: transparent; color: #ffd600;">Opponent</th>
                        <th scope="col" style="background: transparent; color: #ffd600;">Date</th>
                        <th scope="col" style="background: transparent; color: #ffd600;">Result</th>
                    </tr>
                </thead>
                <tbody>
                    {% for b in battles %}
                    <tr>
                        <td class="text-center" style="background: transparent; color: #fff;">{{ b.id }}</td>
                        <td class="text-center" style="background: transparent; color: #fff;">{{ b.opponent }}
                        </td>
                        <td class="text-center" style="background: transparent; color: #fff;">{{
                            b.date.strftime('%Y-%m-%d %H:%M') }}</td>
                        <td class="text-center" style="background: transparent;">
                            {% if b.result == 'Win' %}
                            <span class="badge bg-success">Win</span>
                            {% else %}
                            <span class="badge bg-danger">Loss</span>
                            {% endif %}
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>
```

Frontend View:

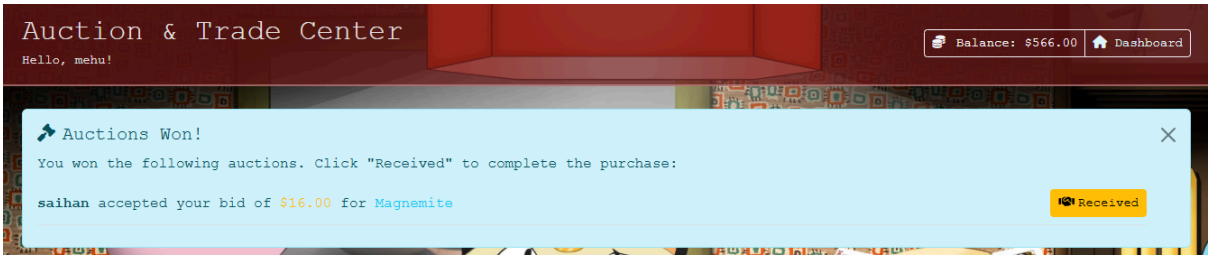
Code Snippet:

```
<div class="container">
    <div class="card shadow-sm mb-4">
        <div class="card-body">
            <div class="row justify-content-center text-center align-items-center">
                <div class="col-md-4 d-flex flex-column align-items-center">
                    <p>Chosen Pokémon: {{ current_battle.opponent_pokemon|capitalize }}</p>
                    <div class="pokemon-box">
                        <img src="https://img.pokemondb.net/artwork/avif/{{current_battle.opponent_pokemon|lower}}.avif"
                            alt="{{ current_battle.opponent_pokemon|capitalize }}"
                            onerror="this.onerror=null; this.src='https://img.pokemondb.net/artwork/avif/unknown.avif';">
                    </div>
                    <p class="mt-2">{{ current_battle.opponent }}</p>
                    <p>Score: {{ current_battle.opponent_score }}</p>
                </div>
            </div>

            <!-- Battle Info -->
            <div class="row mt-4">
                <div class="col-md-8 mx-auto">
                    <div class="turn-indicator">
                        <p class="mb-1"><strong>Current Turn:</strong> {{ current_battle.current_turn }}</p>
                        {% if current_battle.current_turn == session.name %}
                        <p class="mb-0"><strong>Opponent's Last Move:</strong> {{ current_battle.current_move
                            or "N/A" }}</p>
                        {% else %}
                        <p class="mb-0"><strong>Your Last Move:</strong> {{ current_battle.current_move or "N/A"
                            }}</p>
                        {% endif %}
                    </div>
                </div>
            </div>

            <!-- Battle Actions - Only enabled for current player's turn -->
            <div class="row mt-4">
                <div class="col-md-8 mx-auto text-center">
                    <h5>Select Move:</h5>
                    <div class="d-flex justify-content-center gap-3">
                        <!-- Around line where battle actions are defined, update the button disabling logic: -->
                        <button class="btn btn-action" {% if not current_battle.is_users_turn %}disabled{% endif %}
                            onclick="makeMove('attack')">
                            <i class="fas fa-swords me-1"></i>Attack
                        </button>
                        <button class="btn btn-action" {% if not current_battle.is_users_turn %}disabled{% endif %}
                            onclick="makeMove('defend')">
                            <i class="fas fa-shield-alt me-1"></i>Defend
                        </button>
                        <button class="btn btn-action" {% if not current_battle.is_users_turn %}disabled{% endif %}
```

Notification:

I have added a notification pop up when a trade or an auction is completed. If the buyer presses the 'Received' only then the balance transfer will be successful.

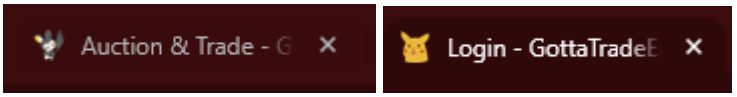
Frontend view:



Code Snippet:

```
{% if accepted_offers %}
<div class="alert alert-success alert-dismissible fade show" role="alert">
  <h5><i class="fas fa-check-circle me-2"></i>Trade Offers Accepted!</h5>
  <p>
    The following trade offers have been accepted. Click "Received" to
    complete the trade:
  </p>
  {% for offer in accepted_offers %}
  <div class="d-flex justify-content-between align-items-center border-bottom pb-2 mb-2">
    <div>
      <strong>{{ offer.trade_owner_name }}</strong> accepted your offer:
      <span class="text-warning">{{ offer.offered_card_name }} + ${{ offer.additional_money
        }}</span>
      for <span class="text-info">{{ offer.requested_card_name }}</span>
    </div>
    <form method="POST" action="{{ url_for('complete_trade') }}" class="d-inline">
      <input type="hidden" name="trade_id" value="{{ offer.trade_id }}" />
      <button type="submit" class="btn btn-warning btn-sm">
        <i class="fas fa-handshake me-1"></i>Received
      </button>
    </form>
  </div>
  {% endfor %}
  <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
</div>
{% endif %}
```

I also added the Tab icon for the login/registration page - Pikachu and all the other pages - Emolga.



Contribution of ID : 23301049, Name : Mehruma Mahmud

Overview: Other than making the login, logout page I have carried out the tasks below:

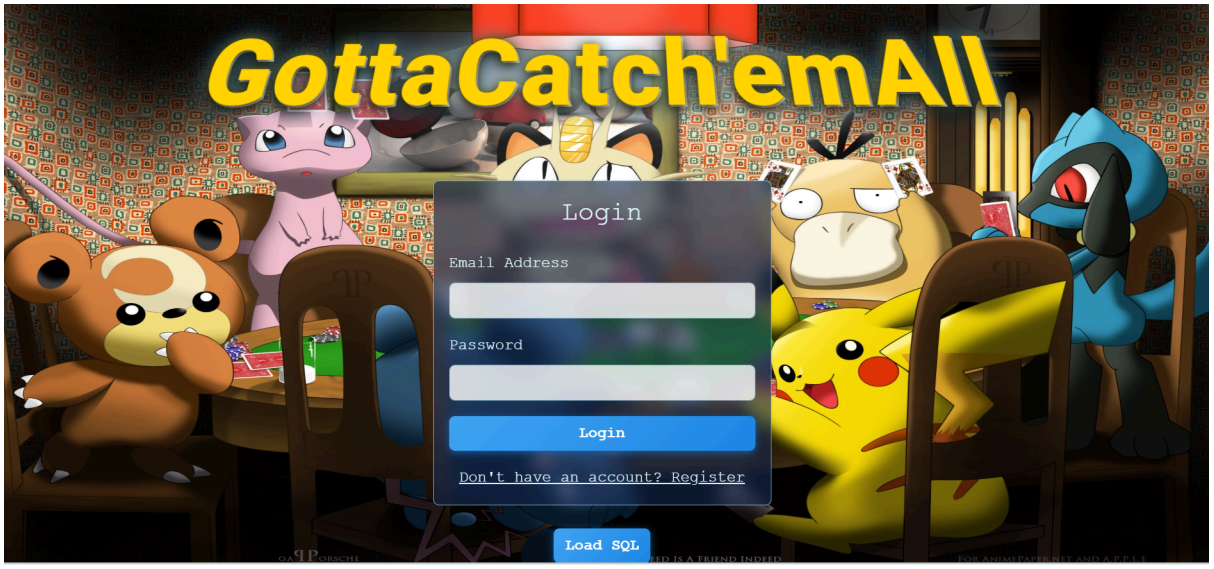
1. Quick access button for trade and auction page
2. Auction and Trade center page
3. Create Auction sidebar

4. Create Trade sidebar
5. Chat sidebar
6. My Items Tab
7. Chat System

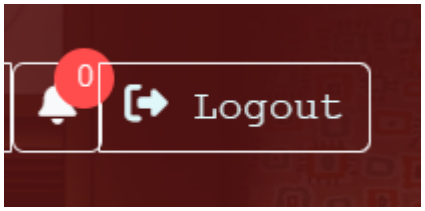
A quick overview of these is provided here:

0. Login and Logout:

Frontend view:



By clicking on the logout button the user gets redirected back to the login page.



P.S: Although I used a picture from google if you paid a little attention it is a small pun. The pokemons in the picture are playing poker which is relevant to the purpose of the website.

Code Snippet:

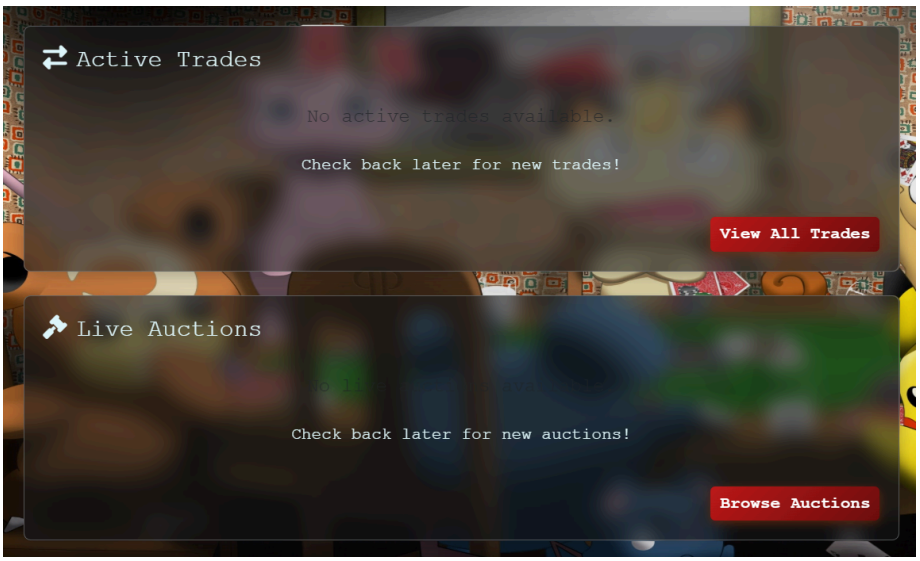
```
<body class="d-flex align-items-center justify-content-center vh-100 flex-column">
    <!-- Straight title -->
    <div class="title-container">
        <div class="title">GottaCatch'emAll</div>
    </div>


    <div class="card shadow-sm " style="width: 22rem;">
        <div class="card-body">
            <h3 class="card-title text-center mb-4">Login</h3>
            <form action="{{ url_for('login') }}" method="POST">
                <div class="mb-3">
                    <label for="email" class="form-label">Email Address</label>
                    <input type="email" class="form-control" id="email" name="email" required>
                </div>
                <div class="mb-3">
                    <label for="password" class="form-label">Password</label>
                    <input type="password" class="form-control" id="password" name="password" required>
                </div>
                <button type="submit" class="btn btn-primary w-100">Login</button>
            </form>
            <div class="text-center mt-3">
                <a href="{{ url_for('register') }}">Don't have an account? Register</a>

            </div>
        </div>
```

```
            </div>
        </div>
    </div>
    <br>
    <form action="{{ url_for('load_sql') }}" method="get">
        <button type="submit" class="btn btn-primary">Load SQL</button>
    </form>
```

1. **Quick access button for trade and auction page:**

The buttons I added on the Live Auction and Active Trades panels redirect the user to the auction and trade center. This has been given here for creating a user friendly environment.

Frontend view:



Code Snippet:

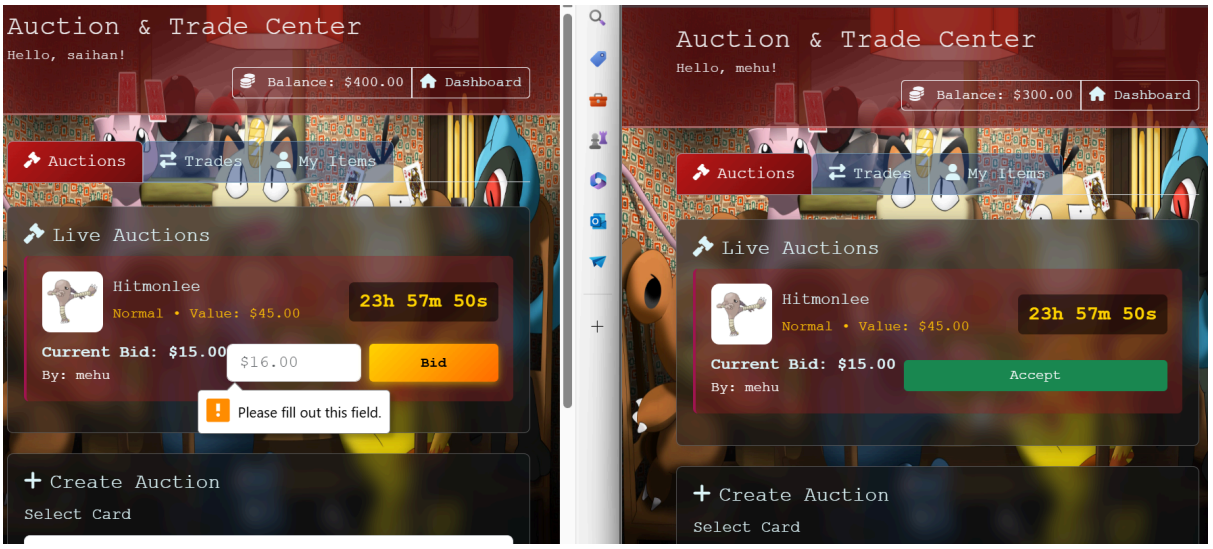
```
    {% endif %}
    <div class="text-end mt-3">
      <a href="{{ url_for('auctions') }}#trades" class="btn btn-primary btn-sm">View All Trades</a>
    </div>
  </div>
</div>

<!-- Live Auctions -->
```

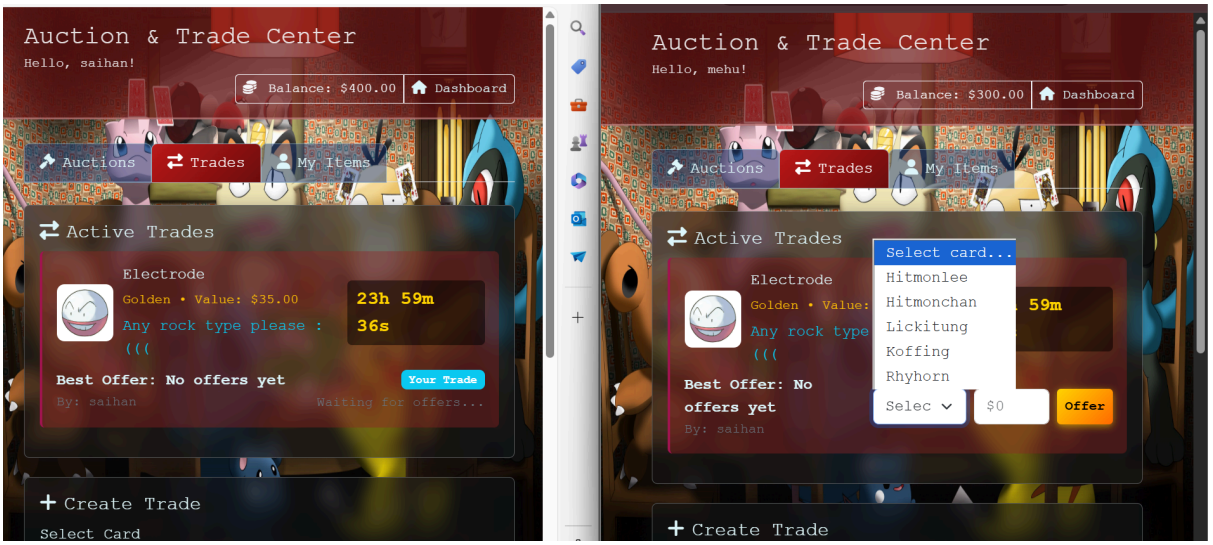
```
      {% endif %}
      <div class="text-end mt-3">
        <a href="{{ url_for('auctions') }}#auctions" class="btn btn-primary btn-sm">Browse Auctions</a>
      </div>
    </div>
  </div>
```

2. Auction and Trade center page:

The Auction & Trade center page gives the user access to 3 tabs where two of those are dedicated to carry out auction and trade based actions.
Auction and Trade provides dynamic use for users.

Frontend view:





Code Snippet:

```
<!-- Navigation Tabs -->
<ul class="nav nav-tabs mb-4" id="marketTabs" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="auctions-tab" data-bs-toggle="tab" data-bs-target="#auctions" type="button"
      role="tab">
      <i class="fas fa-gavel me-2"></i>Auctions
    </button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="trades-tab" data-bs-toggle="tab" data-bs-target="#trades" type="button" role="tab">
      <i class="fas fa-exchange-alt me-2"></i>Trades
    </button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="my-items-tab" data-bs-toggle="tab" data-bs-target="#my-items" type="button"
      role="tab">
      <i class="fas fa-user me-2"></i>My Items
    </button>
  </li>
</ul>
```

```
<!-- Auction Items -->
{% for auction in active_auctions %}
<div class="auction-item">
  <div class="d-flex align-items-center mb-2">
    <img src="https://img.pokemondb.net/artwork/avif/{{ auction.card_name|lower }}.avif"
      class="pokemon-thumbnail" alt="{{ auction.card_name }}"
      onerror="this.onerror=null; this.src='https://img.pokemondb.net/artwork/avif/unknown.avif';" />
    <div class="flex-grow-1">
      <h6 class="mb-1">{{ auction.card_name }}</h6>
      <small class="text-warning">
        {% if auction.golden %}Golden{% elif
        auction.holographic %}Holographic{% else %}Normal{%
        endif %} • Value: ${{ auction.value }}
      </small>
    </div>
    <span class="countdown-timer" data-endtime="{{ auction.end_time.strftime('%Y-%m-%d %H:%M:%S') }}">
      Calculating...
    </span>
  </div>
  <div class="d-flex justify-content-between align-items-center">
    <div>
      <strong>Current Bid: ${{ auction.current_bid or
        auction.starting_bid }}</strong>
```

```
                </button>
            </form>
            {% endif %}
            {% if auction.seller_id == session['user_id'] and auction.current_bid and not
            auction.has_accepted_bid %}
            <form method="POST" action="{{ url_for('accept_bid') }}" class="d-inline ms-2" style="flex: 1;">
                <input type="hidden" name="auction_id" value="{{ auction.auction_id }}" />
                <button type="submit" class="btn btn-success btn-sm w-100">
                    Accept
                </button>
            </form>
            {% endif %}
        </div>
    </div>
    {% else %}
    <div class="text-center py-4">
        <p>No active auctions at the moment.</p>
    </div>
    {% endfor %}
</div>
```

```
<!-- Trade Items -->
{% for trade in active_trades %}
<div class="trade-item">
    <div class="d-flex align-items-center mb-2">
        <img src="https://img.pokemondb.net/artwork/avif/{{ trade.card_name|lower }}.avif"
            class="pokemon-thumbnail" alt="{{ trade.card_name }}"
            onerror="this.onerror=null; this.src='https://img.pokemondb.net/artwork/avif/unknown.avif';" />
        <div class="flex-grow-1">
            <h6 class="mb-1">{{ trade.card_name }}</h6>
            <small class="text-warning">
                {% if trade.golden %}Golden{% elif trade.holographic
                %}Holographic{% else %}Normal{% endif %} • Value: ${{
                trade.value }}
            </small>
            <div class="text-info mt-1">
                {{ trade.description }}
            </div>
        </div>
        <span class="countdown-timer" data-endtime="{{ trade.end_time.strftime('%Y-%m-%d %H:%M:%S') }}">
            Calculating...
        </span>
    </div>
    <div class="d-flex justify-content-between align-items-center">
```

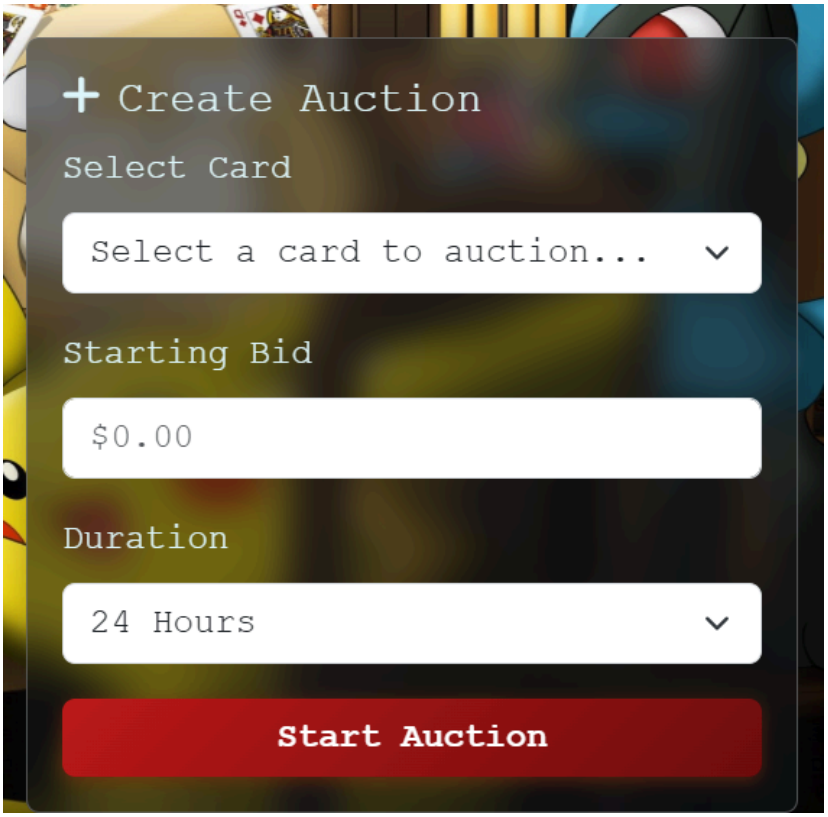
```
                <div>
                    <strong>Best Offer: {{ trade.best_offer or 'No offers yet'
                    }}</strong>
                    <small class="d-block text-muted">By: {{ trade.trader_name }}</small>
                </div>
                {% if trade.trader_id == session.user_id %}
                <!-- User's own trade - show manage options -->
                <div class="text-end">
                    <span class="badge bg-info">Your Trade</span>
                    <br />
                    {% if user_trade_offers.get(trade.trade_id) %}
                    <div class="mt-2">
                        <small class="text-success d-block">{{ user_trade_offers[trade.trade_id]|length }}
                        offer(s) received</small>
                        {% for offer in user_trade_offers[trade.trade_id] %}
                        <div class="border rounded p-2 mb-2 bg-dark">
                            <small class="text-info d-block">
                                <strong>{{ offer.offerer_name }}</strong> offers:
                            </small>
                            <small class="text-warning d-block">
                                {{ offer.offered_card_name }} + ${{
                                offer.additional_money }}
                            </small>
                            {% if offer.status == 'pending' %}
                                offer.additional_money }}
                            </small>
                            {% if offer.status == 'pending' %}
                            <div class="mt-1">
                                <form method="POST" action="{{ url_for('respond_to_trade_offer') }}" class="d-inline">
                                    <input type="hidden" name="trade_id" value="{{ trade.trade_id }}" />
                                    <input type="hidden" name="offerer_id" value="{{ offer.user_id }}" />
                                    <input type="hidden" name="action" value="accept" />
                                    <button type="submit" class="btn btn-success btn-sm">
                                        Accept
                                    </button>
                                </form>
                                <form method="POST" action="{{ url_for('respond_to_trade_offer') }}" class="d-inline">
                                    <input type="hidden" name="trade_id" value="{{ trade.trade_id }}" />
                                    <input type="hidden" name="offerer_id" value="{{ offer.user_id }}" />
                                    <input type="hidden" name="action" value="decline" />
                                    <button type="submit" class="btn btn-outline-danger btn-sm">
                                        Decline
                                    </button>
                                </form>
                            </div>
                            {% elif offer.status == 'accepted' %}
                            <span class="badge bg-success">Accepted - Waiting for confirmation</span>
                            {% elif offer.status == 'declined' %}
```

```
<div class="tab-pane fade" id="trades" role="tabpanel">
    <!-- Other user's trade - show offer form -->
    <form method="POST" action="{{ url_for('place_trade_offer') }}" class="d-flex gap-2">
      <input type="hidden" name="trade_id" value="{{ trade.trade_id }}" />
      <select class="form-select" name="offered_card_id" required style="max-width: 150px">
        <option value="">Select card...</option>
        {% for card in user_cards %}
        <option value="{{ card.card_id }}">
          {{ card.name }}
        </option>
        {% endfor %}
      </select>
      <input type="number" class="form-control" name="additional_money" placeholder="$0" min="0"
        step="0.01" style="max-width: 80px" />
      <button type="submit" class="btn btn-action btn-sm">
        Offer
      </button>
    </form>
    {% endif %}
  </div>
</div>
{% else %}
<div class="text-center py-4">
  <p>No active trades at the moment.</p>
</div>
```

3. Create Auction sidebar

The sidebar helps the user to list a new auction with the cards they own in the system. The starting bid can be set by the user. The time duration for the auction to stay active can also be set from 1 hour to 3 days according to the users' preference.

Frontend view:



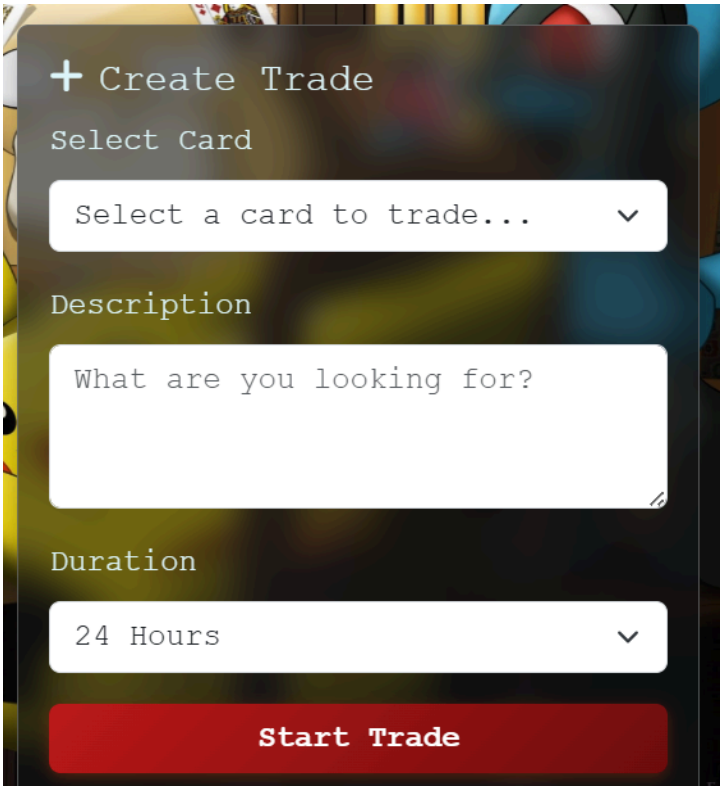
Code Snippet:

```
<form method="POST" action="{{ url_for('create_auction') }}">
  <div class="mb-3">
    <label class="form-label">Select Card</label>
    <select class="form-select" name="card_id" required>
      <option value="">Select a card to auction...</option>
      {% for card in user_cards %}
      <option value="{{ card.card_id }}">
        {{ card.name }} ({% if card.golden %}Golden{% elif
        card.holographic %}Holographic{% else %}Normal{% endif
        %}) - ${{ card.value }}
      </option>
      {% endfor %}
    </select>
  </div>
  <div class="mb-3">
    <label class="form-label">Starting Bid</label>
    <input type="number" class="form-control" name="starting_bid" placeholder="$0.00" min="1"
      step="0.01" required />
  </div>
  <div class="mb-3">
    <label class="form-label">Duration</label>
    <select class="form-select" name="duration">
      <option value="1">1 Hour</option>
```

4. **Create Trade sidebar:**

Just like the Create Auction sidebar, this one helps the user to list a new trade with the cards they own in the system. The user can mention in the message box about what they are expecting from the trade. The time duration for the trade to stay active can also be set from 1 hour to 3 days according to the users' preference.

Frontend view:



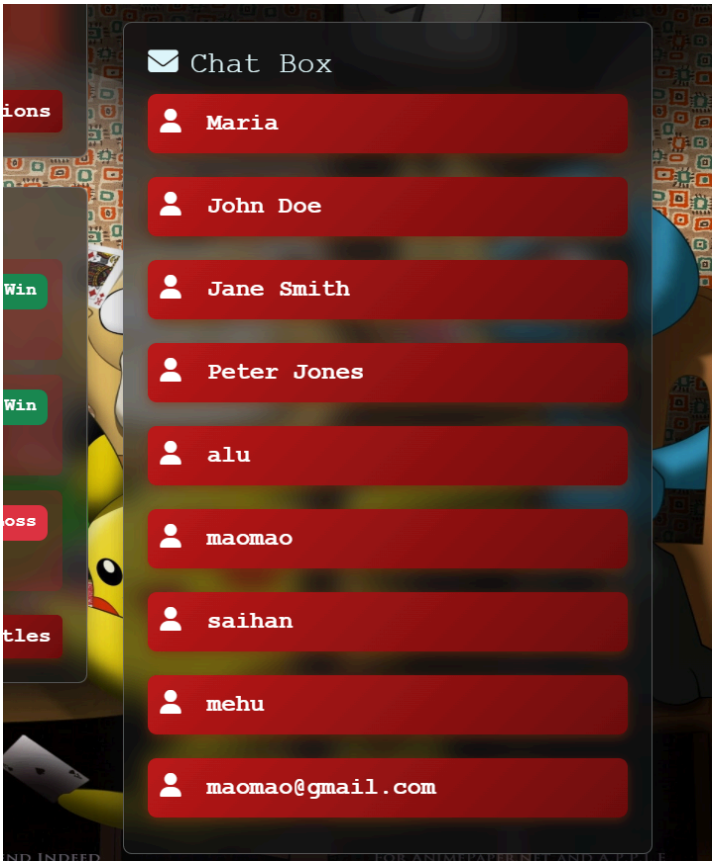
Code Snippet:

```
<h5 class="card-title">
  <i class="fas fa-plus me-2"></i>Create Trade
</h5>
<form method="POST" action="{{ url_for('create_trade') }}">
  <div class="mb-3">
    <label class="form-label">Select Card</label>
    <select class="form-select" name="card_id" required>
      <option value="">Select a card to trade...</option>
      {% for card in user_cards %}
      <option value="{{ card.card_id }}">
        {{ card.name }} ({% if card.golden %}Golden{% elif
        card.holographic %}Holographic{% else %}Normal{% endif
        %}) - ${{ card.value }}
      </option>
      {% endfor %}
    </select>
  </div>
  <div class="mb-3">
    <label class="form-label">Description</label>
    <textarea class="form-control" name="description" placeholder="What are you looking for?" rows="3"
      required></textarea>
  </div>
  <div class="mb-3">
    <label class="form-label">Duration</label>
```

5. Chat sidebar

Frontend view:

By clicking on any of the names shown, the user will be redirected to the chat system where they can chat with the person they wish to chat with (when both parties are present).



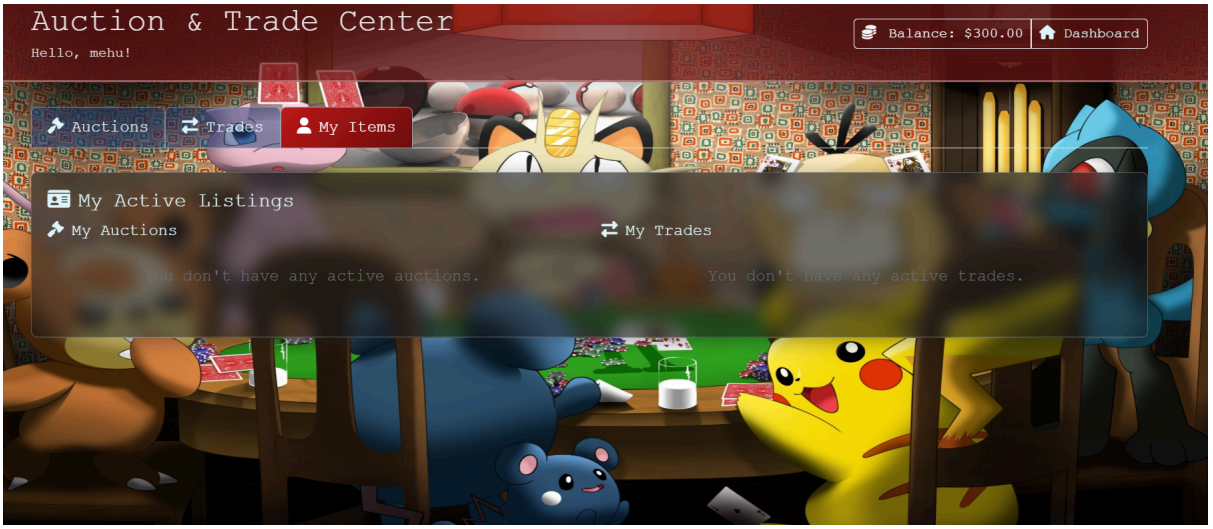
Code Snippet:

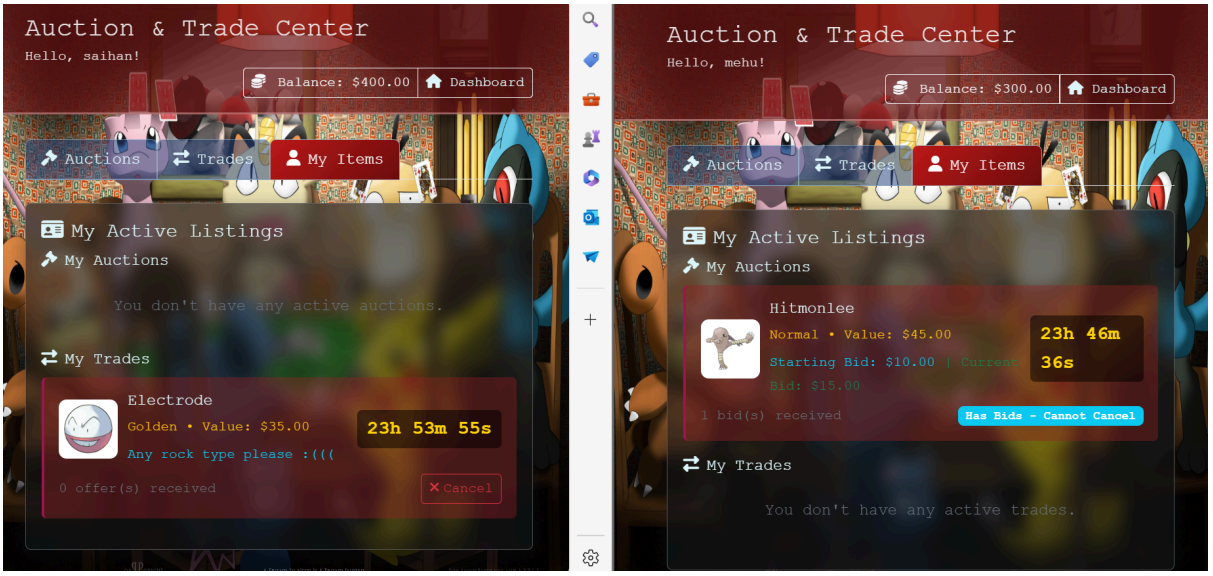
```
</div>
<!-- CHAT Stuffs Kichu ekta -->
<div class="card">
  <div class="card-body">
    <h5 class="card-title">
      <i class="fas fa-envelope me-2"></i>Chat Box
    </h5>
    <div class="d-grid gap-2">
      {% for u in allusers %}
      <form action="{{ url_for('chatbox') }}" method="get" class="mb-2">
        <input type="hidden" name="recipient_id" value="{{ u.user_id }}" />
        <button type="submit" class="btn btn-primary w-100 text-start p-2">
          <div class="card-msg mb-0">
            <i class="fas fa-user me-2"></i> {{ u.name }}
          </div>
        </button>
      </form>
      {% endfor %}
```

6. **My Items tab:**

The users' active listings will show in this tab.

Frontend view:





Code Snippet:

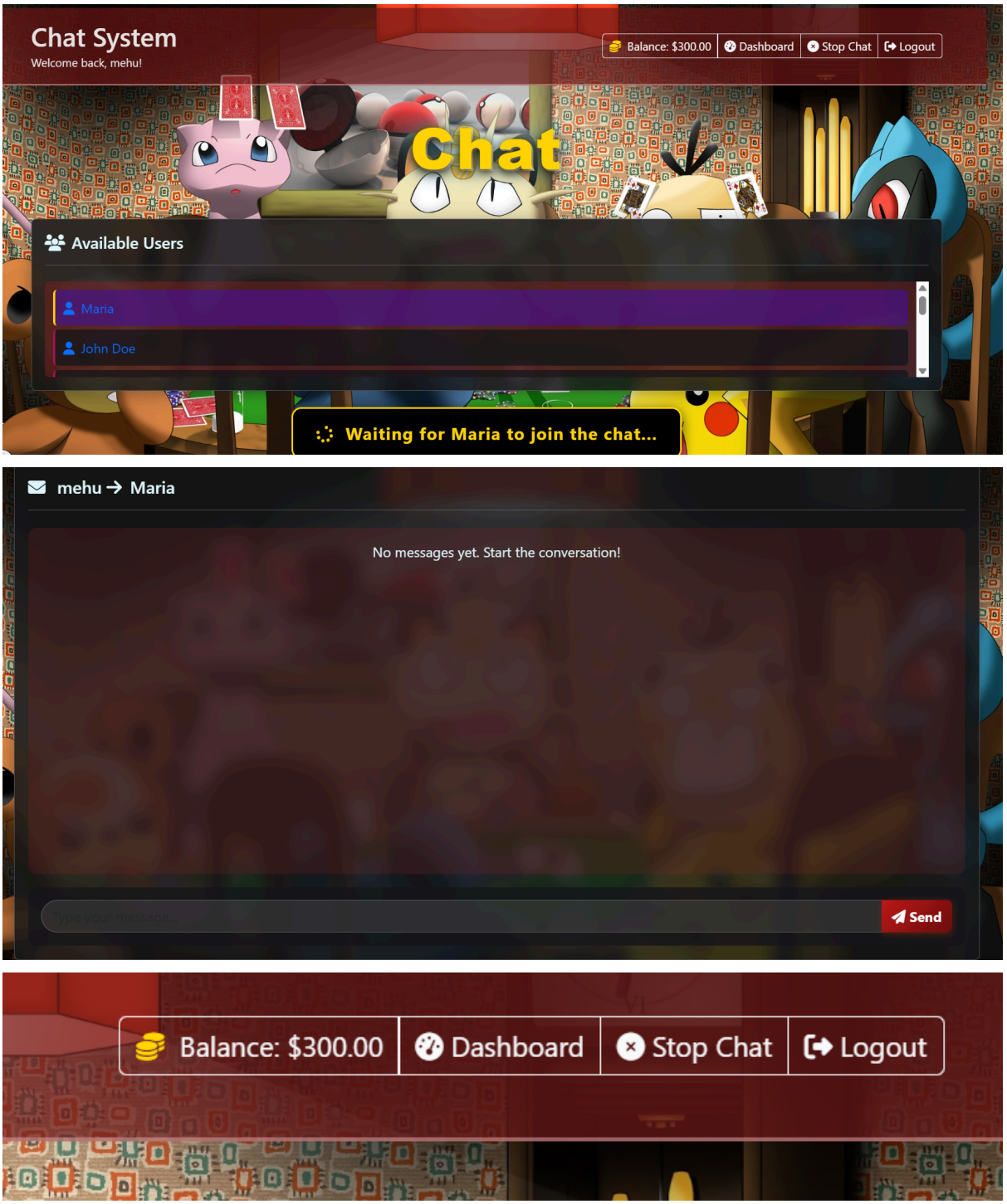
```
<h6><i class="fas fa-gavel me-2"></i>My Auctions</h6>
{% if user_auctions %} {% for auction in user_auctions %}
<div class="auction-item">
  <div class="d-flex align-items-center mb-2">
    <img src="https://img.pokemondb.net/artwork/avif/{{ auction.card_name|lower }}.avif"
      class="pokemon-thumbnail" alt="{{ auction.card_name }}"
      onerror="this.onerror=null; this.src='https://img.pokemondb.net/artwork/avif/unknown.avif';" /
    <div class="flex-grow-1">
      <h6 class="mb-1">{{ auction.card_name }}</h6>
      <small class="text-warning">
        {% if auction.golden %}Golden{% elif
        auction.holographic %}Holographic{% else
        %}Normal{% endif %} • Value: ${{ auction.value }}
      </small>
      <div class="text-info mt-1">
        <small>Starting Bid: ${{ auction.starting_bid
          }}</small>
        {% if auction.current_bid %}
        <small class="text-success">
          | Current Bid: ${{ auction.current_bid }}</small>
        {% endif %}
```

```
                <span class="countdown-timer"
                    data-endtime="{{ auction.end_time.strftime('%Y-%m-%d %H:%M:%S') }}">
                    Calculating...
                </span>
            </div>
            <div class="d-flex justify-content-between align-items-center">
                <small class="text-muted">{{ auction.bid_count }} bid(s) received</small>
                {% if auction.bid_count == 0 %}
                <form method="POST" action="{{ url_for('cancel_auction', auction_id=auction.auction_id) }}"
                    class="d-inline">
                    <button type="submit" class="btn btn-outline-danger btn-sm"
                        onclick="return confirm('Are you sure you want to cancel this auction?')">
                        <i class="fas fa-times me-1"></i>Cancel
                    </button>
                </form>
                {% else %}
                <span class="badge bg-info">Has Bids - Cannot Cancel</span>
                {% endif %}
            </div>
        </div>
        {% endfor %} {% else %}
        <div class="text-center py-3">
            <p class="text-muted">
```

```
    <div class="col-md-6">
        <h6><i class="fas fa-exchange-alt me-2"></i>My Trades</h6>
        {% if user_trades %} {% for trade in user_trades %}
        <div class="trade-item">
            <div class="d-flex align-items-center mb-2">
                <img src="https://img.pokemondb.net/artwork/avif/{{ trade.card_name|lower }}.avif"
                    class="pokemon-thumbnail" alt="{{ trade.card_name }}"
                    onerror="this.onerror=null; this.src='https://img.pokemondb.net/artwork/avif/unknown.avif';" />
                <div class="flex-grow-1">
                    <h6 class="mb-1">{{ trade.card_name }}</h6>
                    <small class="text-warning">
                        {% if trade.golden %}Golden{% elif
                        trade.holographic %}Holographic{% else %}Normal{%
                        endif %} • Value: ${{ trade.value }}
                    </small>
                    <div class="text-info mt-1">
                        <small>{{ trade.description }}</small>
                    </div>
                </div>
                <span class="countdown-timer" data-endtime="{{ trade.end_time.strftime('%Y-%m-%d %H:%M:%S') }}">
                    Calculating...
                </span>
            </div>
        </div>
```

7. **Chat System:**
   In the chat system page the list of users in the site is shown on the available users panel. Below the panel the chatbox can be seen. On the very left of the header there is a “Stop Chat” button. By pressing that a chat session is ended and all the current messages are deleted without logging out.

Frontend view:



Code Snippet:

```
<!-- User List -->
<div class="row">
  <div class="col-12">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">
          <i class="fas fa-users me-2"></i>Available Users
        </h5>
        <div class="chat-sidebar">
          {% for u in all_users %}
          <a href="{{ url_for('chatbox', recipient_id=u.user_id) }}" class="text-decoration-none">
            <div class="chat-user {% if recipient and recipient.user_id == u.user_id %}active{% endif %}">
              <i class="fas fa-user me-2"></i>{{ u.name }} {% if
              u.chatuser_id == user.user_id %}
              <span class="badge bg-success float-end">Online</span>
              {% endif %}
            </div>
          </a>
          {% endfor %}
        </div>
      </div>
    </div>
  </div>
</div>
```

```
<!-- Chat Window -->
<div class="row">
  <div class="col-12">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">
          <i class="fas fa-envelope me-2"></i>
          {% if recipient %} {{ user.name }}
          <i class="fas fa-arrow-right me-1"></i> {{ recipient.name }} {%
          else %} Select a user to start chatting {% endif %}
        </h5>
        <div class="chat-window">
          {% if messages %} {% for message in messages %}
          <div
            class="message {% if message.sender_id == user.user_id %}user-message{% else %}recipient-message{% en
            <div class="message-content">
              {{ message.content }}
              <div class="message-timestamp">
                {{ message.timestamp.strftime('%b %d, %Y %H:%M') }}
              </div>
            </div>
          </div>
          {% endfor %} {% else %}
          <p class="text-center">
            No messages yet. Start the conversation!
          </p>
```

```
          </div>
          {% if recipient %}
          <div class="message-input">
            <form action="{{ url_for('send_message') }}" method="post" id="messageForm">
              <input type="hidden" name="recipient_id" value="{{ recipient.user_id }}" />
              <div class="input-group">
                <input type="text" name="content" class="form-control" placeholder="Type your message..." required
                  maxlength="500" />
                <button type="submit" class="btn btn-primary">
                  <i class="fas fa-paper-plane"></i> Send
                </button>
              </div>
            </form>
          </div>
          {% endif %}
        </div>
```

Contribution of ID : 23301105, Name : Sabbir Ahmed Kabbo

Backend Development

We used **Flask**, **Mysql** and **Xampp** for our backend development of our project for Pokemon Card Trading System. The name of our project is Gotta Catch'em All.

Below, we briefly discuss our contribution in backend development.

Contribution of ID : 23301293, Name : Zarin Tasnim

I am responsible for the Daily Reward feature, Card adding feature for both user's Card list and Wishlist, Battle System, Balance Transfer Integration, Notification feature.

Daily Reward:

I used the time library to find the current login time and compared it with the user's table and if it is the user's first login of the day, 100$ will be added to the user's current balance.

Code Snippet:

```python
# Login route
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        email = request.form['email']
        password = request.form['password']

        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM Users WHERE email = %s AND password = %s', (email, password))
        account = cursor.fetchone()

        if account:
            session['user_id'] = account['user_id']
            session['name'] = account['name']

            # --- Daily reward logic ---
            today = date.today()
            if not account.get('last_login') or account['last_login'] < today:
                daily_reward = 100.00
                new_balance = float(account['balance']) + daily_reward

                # Update balance and last_login in DB
                cursor.execute(
                    'UPDATE Users SET balance = %s, last_login = %s WHERE user_id = %s',
                    (new_balance, today, account['user_id'])
                )
                mysql.connection.commit()
                flash(f'You received {daily_reward} coins as a daily login reward!', 'success')

            flash('Login successful!', 'success')
            return redirect(url_for('dashboard'))
        else:
            flash('Invalid email/password!', 'danger')

    return render_template('login.html')
```

Card adding feature for Card List and Wishlist:

Clicking the add card button for both pages will give an option to put the card name. Then the user input will be added to the Card table in the database with cursor.execute. The query will

for adding card to user's card list will be

```
cursor.execute("""
    INSERT INTO card (card_id, owner_id, name, value, normal, golden, holographic)
    VALUES (%s, %s, %s, %s, %s, %s, %s)
""", (next_card_id, user_id, name, value,
      1 if card_type == 'normal' else 0,
      1 if card_type == 'golden' else 0,
      1 if card_type == 'holographic' else 0))
```

Battle System:

I have used multiple functions for the battle system. One function fetches the battle history from the battle table of the user.

```
# Fetch battle history
cursor.execute("""
    SELECT b.battle_id AS id,
           CASE WHEN b.winner = %s THEN u_loser.name ELSE u_winner.name END AS opponent,
           b.date,
           CASE WHEN b.winner = %s THEN 'Win' ELSE 'Loss' END AS result
    FROM Battle b
    JOIN Users u_winner ON b.winner = u_winner.user_id
    JOIN Users u_loser ON b.loser = u_loser.user_id
    WHERE %s IN (b.winner, b.loser) AND b.status = 'finished'
    ORDER BY b.date DESC
""", (user_id, user_id, user_id))
battles = cursor.fetchall()
```

Another function starts the battle by looking for other users from the challenge table with status 'queued'. If other users are available for the battle then the current user and the first user from the queue will be placed for battle and put into the battle table with status 'ongoing'. The user needs to choose a card from his list to start the battle.

```
# Check if there's someone waiting in the queue
cursor.execute("""
    SELECT c.user_id, u.name, c.card_id, card.name as pokemon_name
    FROM challenge c
    JOIN users u ON c.user_id = u.user_id
    JOIN card ON c.card_id = card.card_id
    WHERE c.status = 'queued' AND c.user_id != %s
    LIMIT 1
""", (user_id,))
waiting_challenge = cursor.fetchone()
```

```
# Create a new battle
cursor.execute("""
    INSERT INTO battle (winner, loser, status, current_turn, current_move, amount)
    VALUES (%s, %s, 'ongoing', %s, 'None', 0.00)
""", (user_id, opponent_id, user_id))
battle_id = cursor.lastrowid

# Insert new challenge for current user
cursor.execute("""
    INSERT INTO challenge (battle_id, user_id, card_id, status)
    VALUES (%s, %s, %s, 'match_found')
""", (battle_id, user_id, card_id))

# Insert new challenge for opponent
cursor.execute("""
    INSERT INTO challenge (battle_id, user_id, card_id, status)
    VALUES (%s, %s, %s, 'match_found')
""", (battle_id, opponent_id, opponent_card_id))
```

Cancel Queue:

Users can cancel the queue before match-making by pressing the cancel queue button. It will remove the user from the challenge table and the battle table.

```
cursor.execute("""
    SELECT battle_id FROM challenge
    WHERE user_id = %s AND status = 'queued'
""", (user_id,))
queue_entry = cursor.fetchone()

if queue_entry:
    battle_id = queue_entry['battle_id']

    # Delete the challenge entry
    cursor.execute("""
        DELETE FROM challenge
        WHERE user_id = %s AND status = 'queued'
    """, (user_id,))

    # Also delete the battle entry if it exists and has status 'queued'
    cursor.execute("""
        DELETE FROM battle
        WHERE battle_id = %s AND status IN ('queued', ' ')
    """, (battle_id,))

    mysql.connection.commit()
```

Battle move:

The user's move is stored and scores are updated after every move. Whoever reaches 100 first wins. The winner gets 200 added to his balance and the loser loses 50 from his balance.

```
# Process move and calculate damage
damage = 0
if move == "attack":
    damage = random.randint(10, 20)
elif move == "special":
    damage = random.randint(15, 30)
# Defend move doesn't do damage
elif move == "defend":
    damage = 5

# Announce winner if anyone's score reaches 100
if (db_battle['player1_score'] + (damage if db_battle['winner'] == user_id else 0)) >= 100:
    winner_id = db_battle['winner']
    loser_id = db_battle['loser']
elif (db_battle['player2_score'] + (damage if db_battle['loser'] == user_id else 0)) >= 100:
    winner_id = db_battle['loser']
    loser_id = db_battle['winner']
else:
    winner_id = None
    loser_id = None
# Update scores in database
# Check opponent's last move
cursor.execute("SELECT current_move FROM battle WHERE battle_id = %s", (battle_id,))
last_move_row = cursor.fetchone()
if last_move_row and last_move_row['current_move'] == "defend":
    damage = 0
if db_battle['winner'] == user_id:
    new_score = db_battle['player1_score'] + damage
    cursor.execute("""
        UPDATE battle SET player1_score = %s, current_move = %s
        WHERE battle_id = %s
    """, (new_score, move, battle_id))
else:
    new_score = db_battle['player2_score'] + damage
    cursor.execute("""
        UPDATE battle SET player2_score = %s, current_move = %s
        WHERE battle_id = %s
    """, (new_score, move, battle_id))

# Switch turns to opponent
cursor.execute("""
    UPDATE battle SET current_turn = %s WHERE battle_id = %s
""", (opponent_id, battle_id))
```

```
cursor.execute("""
    UPDATE users
    SET balance = balance + 200
    WHERE user_id = %s
""", (winner_id,))

cursor.execute("""
    UPDATE users
    SET balance = GREATEST(0, balance - 50)
    WHERE user_id = %s
""", (loser_id,))
```

Balance Transfer Integration:

I added the balance transfer feature. Every time an auction or a trade is complete, and money is involved, the seller receives the money and the buyer balance decreases. When the buyer presses the received button from the notification, it means that he received the card. And then the transfer happens.

```python
# Handle additional money if any
if offer_details['additional_money'] > 0:
    # Deduct from offerer (current user)
    cursor.execute("""
        UPDATE users SET balance = balance - %s WHERE user_id = %s
    """, (offer_details['additional_money'], session['user_id']))

    # Add to trade owner
    cursor.execute("""
        UPDATE users SET balance = balance + %s WHERE user_id = %s
    """, (offer_details['additional_money'], offer_details['user_id']))
```

Notification:

For creating notifications, I used a helper function that inserts into the notifications table. This function is called for trade and auction to get notification after each transaction.

```python
# Notification functions
def create_notification(user_id, notification_type, title, message, related_id=None):
    """Helper function to create notifications"""
    cursor = mysql.connection.cursor()
    cursor.execute("""
        INSERT INTO notifications (user_id, type, title, message, related_id)
        VALUES (%s, %s, %s, %s, %s)
    """, (user_id, notification_type, title, message, related_id))
    mysql.connection.commit()
    cursor.close()
```

Contribution of ID : 23301049, Name : Mehruma Mahmud

Among the backend features, I have implemented the whole auction, trade, and chat system.

Auction System:

At first, on the dashboard in the “Live Auction” panel all the active auctions are shown by fetching them from the database. Then as the user gets redirected to the auction and trade center, they can add new auctions by using the create auction tab. By using that the database updates itself and shows the result on not only the panel on the left but also on My Active Listings. A minimum bid can be set and that is updated on the database as well. The time is set to 24 hours by default for the auction to stay active if not set otherwise.

The highest amount the user from receiver-end can bid is all the money in their balance. After the designated time is over, the query fetches the highest bid along with its user applying DESC and the card and coins are transferred to the respective accounts.

With this, the auction comes to an end and the history of the bid is removed from the database.

```python
@app.route('/create_auction', methods=['POST'])
def create_auction():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    card_id = request.form.get('card_id')
    starting_bid = request.form.get('starting_bid')
    duration_hours = request.form.get('duration', 24)  # Default 24 hours

    try:
        starting_bid = float(starting_bid)
        duration_hours = int(duration_hours)
    except (ValueError, TypeError):
        flash('Invalid input values!', 'danger')
        return redirect(url_for('auctions'))

    if starting_bid <= 0:
        flash('Starting bid must be greater than 0!', 'danger')
        return redirect(url_for('auctions'))

    cursor = mysql.connection.cursor()

    # Verify user owns the card
    cursor.execute('SELECT owner_id FROM card WHERE card_id = %s', (card_id,))
    card = cursor.fetchone()
```

```python
# Verify user owns the card
cursor.execute('SELECT owner_id FROM card WHERE card_id = %s', (card_id,))
card = cursor.fetchone()

if not card or card['owner_id'] != session['user_id']:
    flash('You do not own this card!', 'danger')
    return redirect(url_for('auctions'))

# Check if card is already in an active auction
cursor.execute("""
    SELECT auction_id FROM auction
    WHERE card_id = %s AND end_time > NOW()
""", (card_id,))
existing_auction = cursor.fetchone()

if existing_auction:
    flash('This card is already listed in an active auction!', 'warning')
    return redirect(url_for('auctions'))

# Create the auction
start_time = datetime.now()
end_time = start_time + timedelta(hours=duration_hours)

cursor.execute("""
    INSERT INTO auction (start_time, end_time, starting_bid, user_id, card_id)
    VALUES (%s, %s, %s, %s, %s)
```

```python
@app.route('/market')
def market():
    return redirect(url_for('auctions'))

@app.route('/accept_bid', methods=['POST'])
def accept_bid():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    auction_id = request.form.get('auction_id')

    cursor = mysql.connection.cursor()

    try:
        # 1. Get auction details and verify ownership
        cursor.execute("""
            SELECT a.*, c.owner_id as card_owner_id, c.name as card_name
            FROM auction a
            JOIN card c ON a.card_id = c.card_id
            WHERE a.auction_id = %s AND a.user_id = %s
        """, (auction_id, session['user_id']))
        auction = cursor.fetchone()

        if not auction:
            flash('Auction not found or you are not the owner!', 'danger')
            return redirect(url_for('auctions'))
```

```python
        # 2. Get the highest bid
        cursor.execute("""
            SELECT b.user_id, b.bid_amount, u.name as bidder_name
            FROM bids_in b
            JOIN users u ON b.user_id = u.user_id
            WHERE b.auction_id = %s
            ORDER BY b.bid_amount DESC, b.timestamp DESC
            LIMIT 1
        """, (auction_id,))
        highest_bid = cursor.fetchone()

        if not highest_bid:
            flash('No bids found for this auction!', 'warning')
            return redirect(url_for('auctions'))

        # 3. Store the accepted bid information in a separate table
        # First check if an accepted bid already exists
        cursor.execute("""
            SELECT * FROM accepted_bids WHERE auction_id = %s
        """, (auction_id,))
        existing_accepted_bid = cursor.fetchone()
```

```python
        if existing_accepted_bid:
            # Update existing record
            cursor.execute("""
                UPDATE accepted_bids
                SET bidder_id = %s, bid_amount = %s
                WHERE auction_id = %s
            """, (highest_bid['user_id'], highest_bid['bid_amount'], auction_id))
        else:
            # Create new record
            cursor.execute("""
                INSERT INTO accepted_bids (auction_id, bidder_id, bid_amount)
                VALUES (%s, %s, %s)
            """, (auction_id, highest_bid['user_id'], highest_bid['bid_amount']))

        # 4. Create notification for the winner
        create_notification(
            highest_bid['user_id'],
            'auction_won',
            'You Won the Auction!',
            f"You won the auction for {auction['card_name']} with a bid of ${highest_bid['bid_amount']:.2f}. Click '
            auction_id
        )
```

```python
@app.route('/auctions')
def auctions():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    cursor = mysql.connection.cursor()

    # Fetch active auctions with user and card details
    cursor.execute("""
        SELECT a.auction_id, a.start_time, a.end_time, a.starting_bid,
               u.user_id as seller_id, u.name as seller_name,
               c.card_id, c.name as card_name, c.value, c.normal, c.golden, c.holographic,
               (SELECT MAX(bid_amount) FROM bids_in WHERE auction_id = a.auction_id) as current_bid,
               (SELECT COUNT(*) FROM accepted_bids WHERE auction_id = a.auction_id) as has_accepted_bid
        FROM auction a
        JOIN users u ON a.user_id = u.user_id
        JOIN card c ON a.card_id = c.card_id
        WHERE a.end_time > NOW() AND c.owner_id = u.user_id
        ORDER BY a.end_time ASC
    """)
    active_auctions = cursor.fetchall()

    # Fetch active trades with user and card details
    cursor.execute("""
        SELECT t.trade_id, t.start_time, t.end_time, t.description,
               u.user_id as trader_id, u.name as trader_name,
               c.card_id, c.name as card_name, c.value, c.normal, c.golden, c.holographic,
               (SELECT CONCAT(offered_card.name, ' + $', COALESCE(MAX(to2.additional_money), 0))
```

```python
@app.route('/complete_auction', methods=['POST'])
def complete_auction():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    auction_id = request.form.get('auction_id')

    cursor = mysql.connection.cursor()

    try:
        # 1. Get the accepted bid details from accepted_bids table
        cursor.execute("""
            SELECT ab.*, a.user_id as seller_id, a.card_id, c.name as card_name
            FROM accepted_bids ab
            JOIN auction a ON ab.auction_id = a.auction_id
            JOIN card c ON a.card_id = c.card_id
            WHERE ab.auction_id = %s AND ab.bidder_id = %s
        """, (auction_id, session['user_id']))
        accepted_bid = cursor.fetchone()

        if not accepted_bid:
            flash('Accepted bid not found or you are not the winning bidder!', 'danger')
            return redirect(url_for('auctions'))
```

```
        # 3. Transfer card ownership to the bidder
        cursor.execute("""
            UPDATE card
            SET owner_id = %s
            WHERE card_id = %s
        """, (session['user_id'], accepted_bid['card_id']))

        # 4. End the auction and remove the accepted bid record
        cursor.execute("""
            UPDATE auction
            SET end_time = NOW()
            WHERE auction_id = %s
        """, (auction_id,))

        cursor.execute("""
            DELETE FROM accepted_bids
            WHERE auction_id = %s
        """, (auction_id,))

        mysql.connection.commit()
        cursor.close()

        flash(f'Auction completed! You received {accepted_bid["card_name"]} for ${accepted_bid["bid_amount"]:.2f}',
```

```
@app.route('/place_bid', methods=['POST'])
def place_bid():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    auction_id = request.form.get('auction_id')
    bid_amount = request.form.get('bid_amount')

    try:
        bid_amount = float(bid_amount)
        auction_id = int(auction_id)
    except (ValueError, TypeError):
        flash('Invalid bid amount!', 'danger')
        return redirect(url_for('auctions'))

    cursor = mysql.connection.cursor()

    # Get auction details
    cursor.execute("""
        SELECT a.*, c.name as card_name, u.name as seller_name,
               (SELECT MAX(bid_amount) FROM bids_in WHERE auction_id = a.auction_id) as current_bid
        FROM auction a
        JOIN card c ON a.card_id = c.card_id
        JOIN users u ON a.user_id = u.user_id
        WHERE a.auction_id = %s AND a.end_time > NOW()
    """, (auction_id,))
    auction = cursor.fetchone()
```

```python
    # Determine minimum bid
    current_bid = auction['current_bid'] if auction['current_bid'] else auction['starting_bid']
    min_bid = current_bid + 1 if current_bid else auction['starting_bid']

    # Validate bid
    if bid_amount < min_bid:
        flash(f'Bid must be at least ${min_bid:.2f}!', 'danger')
        return redirect(url_for('auctions'))

    if bid_amount > user_balance:
        flash('You do not have enough balance to place this bid!', 'danger')
        return redirect(url_for('auctions'))

    cursor.execute("""
                SELECT user_id, bid_amount FROM bids_in
                WHERE user_id = %s AND auction_id = %s
            """, (session['user_id'], auction_id))
    existing_bid = cursor.fetchone()
```

```python
# If user has already bid, update instead of insert
    if existing_bid:
        # Calculate the difference between new and old bid
        bid_difference = bid_amount - float(existing_bid['bid_amount'])

        # Check if user has enough balance for the increased bid
        if bid_difference > user_balance:
            flash('You do not have enough balance to increase your bid!', 'danger')
            return redirect(url_for('auctions'))
        else:
            cursor.execute("""
                UPDATE bids_in
                SET bid_amount = %s
                WHERE user_id = %s and auction_id = %s
            """, (bid_amount, existing_bid['user_id'], auction_id))

    else:
    # Place the bid
        cursor.execute("""
            INSERT INTO bids_in (user_id, auction_id, bid_amount)
            VALUES (%s, %s, %s)
        """, (session['user_id'], auction_id, bid_amount))
```

(function) execute: Any

Trade System:

From the dashboard in the “Active Trades” panel all the active trades are shown by fetching them from the database. Then as the user gets redirected to the auction and trade center, they can add new trades through the sidebar which updates the database after the system verifies the card ownership. Time limit can be customized and is constantly updated using query.

From the other end, users can offer trade using their balance or card for an equivalent exchange. After accepting the trade the ownership is automatically transferred and all the changes between the users’ collections are updated accordingly.

```python
@app.route('/create_trade', methods=['POST'])
def create_trade():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    card_id = request.form.get('card_id')
    description = request.form.get('description')
    duration_hours = request.form.get('duration', 24)  # Default 24 hours

    try:
        duration_hours = int(duration_hours)
    except (ValueError, TypeError):
        flash('Invalid duration value!', 'danger')
        return redirect(url_for('auctions'))

    if not description or not description.strip():
        flash('Description is required!', 'danger')
        return redirect(url_for('auctions'))

    cursor = mysql.connection.cursor()
```

```python
    # Verify user owns the card
    cursor.execute('SELECT owner_id FROM card WHERE card_id = %s', (card_id,))
    card = cursor.fetchone()

    if not card or card['owner_id'] != session['user_id']:
        flash('You do not own this card!', 'danger')
        return redirect(url_for('auctions'))

    # Check if card is already in an active trade or auction
    cursor.execute("""
        SELECT trade_id FROM trade
        WHERE card_id = %s AND end_time > NOW()
    """, (card_id,))
    existing_trade = cursor.fetchone()

    cursor.execute("""
        SELECT auction_id FROM auction
        WHERE card_id = %s AND end_time > NOW()
    """, (card_id,))
    existing_auction = cursor.fetchone()

    if existing_trade or existing_auction:
        flash('This card is already listed in an active trade or auction!', 'warning')
        return redirect(url_for('auctions'))
```

```
# Create the trade
start_time = datetime.now()
end_time = start_time + timedelta(hours=duration_hours)

cursor.execute("""
    INSERT INTO trade (start_time, end_time, description, user_id, card_id)
    VALUES (%s, %s, %s, %s, %s)
""", (start_time, end_time, description.strip(), session['user_id'], card_id))

trade_id = cursor.lastrowid
mysql.connection.commit()

# Check if any users have this card in their wishlist and notify them
cursor.execute("""
    SELECT w.user_id, u.name, c.name as card_name
    FROM wishlist w
    JOIN users u ON w.user_id = u.user_id
    JOIN card c ON w.card_id = c.card_id
    WHERE w.card_id = %s AND w.user_id != %s
""", (card_id, session['user_id']))

wishlist_users = cursor.fetchall()
print(f"DEBUG: Found {len(wishlist_users)} users with card {card_id} in their wishlist")

for wishlist_user in wishlist_users:
    print(f"DEBUG: Creating notification for user {wishlist_user['user_id']} ({wishlist_user['name']}) for card
    create_notification(
```

```
@app.route('/place_trade_offer', methods=['POST'])
def place_trade_offer():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    trade_id = request.form.get('trade_id')
    offered_card_id = request.form.get('offered_card_id')
    additional_money = request.form.get('additional_money', 0)

    try:
        additional_money = float(additional_money) if additional_money else 0.0
        trade_id = int(trade_id)
        offered_card_id = int(offered_card_id)
    except (ValueError, TypeError):
        flash('Invalid input values!', 'danger')
        return redirect(url_for('auctions'))

    cursor = mysql.connection.cursor()

    # Get trade details
    cursor.execute("""
        SELECT t.*, c.name as card_name, u.name as trader_name
        FROM trade t
        JOIN card c ON t.card_id = c.card_id
        JOIN users u ON t.user_id = u.user_id
        WHERE t.trade_id = %s AND t.end_time > NOW()
    """, (trade_id,))
```

```
    if not trade:
        flash('Trade not found or has ended!', 'danger')
        return redirect(url_for('auctions'))

    # Check if user is the trade owner
    if trade['user_id'] == session['user_id']:
        flash('You cannot make an offer on your own trade!', 'warning')
        return redirect(url_for('auctions'))

    # Verify user owns the offered card
    cursor.execute('SELECT owner_id FROM card WHERE card_id = %s', (offered_card_id,))
    offered_card = cursor.fetchone()

    if not offered_card or offered_card['owner_id'] != session['user_id']:
        flash('You do not own the offered card!', 'danger')
        return redirect(url_for('auctions'))

    # Get user balance if additional money is offered
    if additional_money > 0:
        cursor.execute('SELECT balance FROM users WHERE user_id = %s', (session['user_id'],))
        user_balance = cursor.fetchone()['balance']

        if additional_money > user_balance:
            flash('You do not have enough balance to offer this amount!', 'danger')
            return redirect(url_for('auctions'))
```

```
    # Check if user already made an offer on this trade
    cursor.execute("""
        SELECT user_id FROM trade_offers
        WHERE trade_id = %s AND user_id = %s
    """, (trade_id, session['user_id']))
    existing_offer = cursor.fetchone()

    if existing_offer:
        # Update existing offer
        cursor.execute("""
            UPDATE trade_offers
            SET offered_card_id = %s, additional_money = %s, timestamp = NOW()
            WHERE trade_id = %s AND user_id = %s
        """, (offered_card_id, additional_money, trade_id, session['user_id']))
        flash('Your offer has been updated!', 'success')
    else:
        # Create new offer
        cursor.execute("""
            INSERT INTO trade_offers (user_id, trade_id, offered_card_id, additional_money)
            VALUES (%s, %s, %s, %s)
        """, (session['user_id'], trade_id, offered_card_id, additional_money))
        flash('Your trade offer has been placed!', 'success')

    mysql.connection.commit()
    cursor.close()

    return redirect(url_for('auctions'))
```

```
# Fetch active trades with user and card details
cursor.execute("""
    SELECT t.trade_id, t.start_time, t.end_time, t.description,
           u.user_id as trader_id, u.name as trader_name,
           c.card_id, c.name as card_name, c.value, c.normal, c.golden, c.holographic,
           (SELECT CONCAT(offered_card.name, ' + $', COALESCE(MAX(to2.additional_money), 0))
            FROM trade_offers to2
            JOIN card offered_card ON to2.offered_card_id = offered_card.card_id
            WHERE to2.trade_id = t.trade_id
            ORDER BY to2.additional_money DESC, to2.timestamp DESC LIMIT 1) as best_offer
    FROM trade t
    JOIN users u ON t.user_id = u.user_id
    JOIN card c ON t.card_id = c.card_id
    WHERE t.end_time > NOW() AND c.owner_id = u.user_id
    ORDER BY t.end_time ASC
""")
active_trades = cursor.fetchall()
```

```
# Fetch trade offers for user's own trades
user_trade_offers = {}
for trade in active_trades:
    if trade['trader_id'] == session['user_id']:
        cursor.execute("""
            SELECT to2.user_id, to2.additional_money, to2.timestamp, to2.status,
                   u.name as offerer_name, c.name as offered_card_name, c.card_id as offered_card_id
            FROM trade_offers to2
            JOIN users u ON to2.user_id = u.user_id
            JOIN card c ON to2.offered_card_id = c.card_id
            WHERE to2.trade_id = %s
            ORDER BY to2.additional_money DESC, to2.timestamp DESC
        """, (trade['trade_id'],))
        user_trade_offers[trade['trade_id']] = cursor.fetchall()

# Fetch user's cards for creating auctions/trades
cursor.execute("""
    SELECT card_id, name, value, normal, golden, holographic
    FROM card
    WHERE owner_id = %s
""", (session['user_id'],))
user_cards = cursor.fetchall()
```

```
# Fetch accepted auctions where current user is the winning bidder (for "Received" button)
cursor.execute("""
    SELECT ab.auction_id, ab.bid_amount,
           c.name as card_name, u.name as seller_name
    FROM accepted_bids ab
    JOIN auction a ON ab.auction_id = a.auction_id
    JOIN card c ON a.card_id = c.card_id
    JOIN users u ON a.user_id = u.user_id
    WHERE ab.bidder_id = %s AND a.end_time > NOW()
""", (session['user_id'],))
accepted_auctions = cursor.fetchall()

# Fetch accepted trades where current user made an offer (for "Received" button)
cursor.execute("""
    SELECT t.trade_id, t.user_id as trade_owner_id, to2.status,
           c1.name as requested_card_name, c2.name as offered_card_name,
           u.name as trade_owner_name, to2.additional_money
    FROM trade_offers to2
    JOIN trade t ON to2.trade_id = t.trade_id
    JOIN card c1 ON t.card_id = c1.card_id
    JOIN card c2 ON to2.offered_card_id = c2.card_id
    JOIN users u ON t.user_id = u.user_id
    WHERE to2.user_id = %s AND to2.status = 'accepted' AND t.end_time > NOW()
""", (session['user_id'],))
accepted_offers = cursor.fetchall()
```

```python
@app.route('/respond_to_trade_offer', methods=['POST'])
def respond_to_trade_offer():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    trade_id = request.form.get('trade_id')
    offerer_id = request.form.get('offerer_id')
    action = request.form.get('action')  # 'accept' or 'decline'

    try:
        trade_id = int(trade_id)
        offerer_id = int(offerer_id)
    except (ValueError, TypeError):
        flash('Invalid input values!', 'danger')
        return redirect(url_for('auctions'))

    if action not in ['accept', 'decline']:
        flash('Invalid action!', 'danger')
        return redirect(url_for('auctions'))

    cursor = mysql.connection.cursor()
```

```python
    # Verify user owns the trade
    cursor.execute("""
        SELECT t.*, c.name as card_name
        FROM trade t
        JOIN card c ON t.card_id = c.card_id
        WHERE t.trade_id = %s AND t.user_id = %s AND t.end_time > NOW()
    """, (trade_id, session['user_id']))
    trade = cursor.fetchone()

    if not trade:
        flash('Trade not found or you do not own this trade!', 'danger')
        return redirect(url_for('auctions'))

    # Update the offer status
    cursor.execute("""
        UPDATE trade_offers
        SET status = %s
        WHERE trade_id = %s AND user_id = %s
    """, (action + 'ed', trade_id, offerer_id))
```

```python
    if action == 'accept':
        # When accepting, decline all other offers for this trade
        cursor.execute("""
            UPDATE trade_offers
            SET status = 'declined'
            WHERE trade_id = %s AND user_id != %s
        """, (trade_id, offerer_id))
        flash('Trade offer accepted! Waiting for the offerer to confirm receipt.', 'success')
    else:
        flash('Trade offer declined.', 'info')

    mysql.connection.commit()
    cursor.close()

    return redirect(url_for('auctions'))
```

main.py > auctions

```python
@app.route('/complete_trade', methods=['POST'])
def complete_trade():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    trade_id = request.form.get('trade_id')

    try:
        trade_id = int(trade_id)
    except (ValueError, TypeError):
        flash('Invalid trade ID!', 'danger')
        return redirect(url_for('auctions'))

    cursor = mysql.connection.cursor()

    # Get the accepted offer details
    cursor.execute("""
        SELECT t.*, to2.*, c1.name as trade_card_name, c2.name as offered_card_name,
               u.name as trade_owner_name
        FROM trade_offers to2
        JOIN trade t ON to2.trade_id = t.trade_id
        JOIN card c1 ON t.card_id = c1.card_id
        JOIN card c2 ON to2.offered_card_id = c2.card_id
        JOIN users u ON t.user_id = u.user_id
        WHERE to2.trade_id = %s AND to2.user_id = %s AND to2.status = 'accepted'
    """, (trade_id, session['user_id']))
    offer_details = cursor.fetchone()

```

```
if not offer_details:
    flash('No accepted offer found for this trade!', 'danger')
    return redirect(url_for('auctions'))

try:
    # Start transaction
    mysql.connection.begin()

    # Transfer cards
    # Give the trade owner's card to the offerer (current user)
    cursor.execute("""
        UPDATE card SET owner_id = %s WHERE card_id = %s
    """, (session['user_id'], offer_details['card_id']))
    cursor.execute("""UPDATE trade SET end_time = NOW() WHERE trade_id = %s""", (trade_id,))

    # Give the offerer's card to the trade owner
    cursor.execute("""
        UPDATE card SET owner_id = %s WHERE card_id = %s
    """, (offer_details['user_id'], offer_details['offered_card_id']))


    # Handle additional money if any
    if offer_details['additional_money'] > 0:
        # Deduct from offerer (current user)
        cursor.execute("""
            UPDATE users SET balance = balance - %s WHERE user_id = %s
        """, (offer_details['additional_money'], session['user_id']))
```

```
    # Handle additional money if any
    if offer_details['additional_money'] > 0:
        # Deduct from offerer (current user)
        cursor.execute("""
            UPDATE users SET balance = balance - %s WHERE user_id = %s
        """, (offer_details['additional_money'], session['user_id']))

        # Add to trade owner
        cursor.execute("""
            UPDATE users SET balance = balance + %s WHERE user_id = %s
        """, (offer_details['additional_money'], offer_details['user_id']))

    # Remove the completed trade and all its offers
    cursor.execute("DELETE FROM trade_offers WHERE trade_id = %s", (trade_id,))
    cursor.execute("DELETE FROM trade WHERE trade_id = %s", (trade_id,))

    mysql.connection.commit()
    flash(f'Trade completed! You received {offer_details["trade_card_name"]} and paid ${offer_details["additiona

except Exception as e:
    mysql.connection.rollback()
    print(f"Error completing trade: {e}")
    flash('Error completing trade. Please try again.', 'danger')

cursor.close()
return redirect(url_for('auctions'))
```

My Items:

Query fetches all the active listings of the user through the user id.

```
    # Fetch user's own active auctions
    cursor.execute("""
        SELECT a.auction_id, a.start_time, a.end_time, a.starting_bid,
               c.card_id, c.name as card_name, c.value, c.normal, c.golden, c.holographic,
               (SELECT MAX(bid_amount) FROM bids_in WHERE auction_id = a.auction_id) as current_bid,
               (SELECT COUNT(*) FROM bids_in WHERE auction_id = a.auction_id) as bid_count
        FROM auction a
        JOIN card c ON a.card_id = c.card_id
        WHERE a.user_id = %s AND a.end_time > NOW()
        ORDER BY a.end_time ASC
    """, (session['user_id'],))
    user_auctions = cursor.fetchall()

    # Fetch user's own active trades
    cursor.execute("""
        SELECT t.trade_id, t.start_time, t.end_time, t.description,
               c.card_id, c.name as card_name, c.value, c.normal, c.golden, c.holographic,
               (SELECT COUNT(*) FROM trade_offers WHERE trade_id = t.trade_id) as offer_count
        FROM trade t
        JOIN card c ON t.card_id = c.card_id
        WHERE t.user_id = %s AND t.end_time > NOW()
        ORDER BY t.end_time ASC
    """, (session['user_id'],))
    user_trades = cursor.fetchall()

    cursor.close()
```

```
def my_trades():
    # Fetch offers for each trade
    trade_offers = {}
    for trade in active_trades:
        cursor.execute("""
            SELECT to2.user_id, to2.additional_money, to2.timestamp,
                   u.name as offerer_name, c.name as offered_card_name
            FROM trade_offers to2
            JOIN users u ON to2.user_id = u.user_id
            JOIN card c ON to2.offered_card_id = c.card_id
            WHERE to2.trade_id = %s
            ORDER BY to2.additional_money DESC, to2.timestamp DESC
        """, (trade['trade_id'],))
        trade_offers[trade['trade_id']] = cursor.fetchall()

    cursor.close()

    return render_template('my_trades.html',
                           active_trades=active_trades,
                           trade_offers=trade_offers)

@app.route('/my_auctions')
def my_auctions():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    cursor = mysql.connection.cursor()
```

```python
@app.route('/my_trades')
def my_trades():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    cursor = mysql.connection.cursor()

    # Fetch user's active trades with offers
    cursor.execute("""
        SELECT t.trade_id, t.start_time, t.end_time, t.description,
               c.name as card_name, c.value,
               COUNT(to2.user_id) as offer_count,
               (SELECT CONCAT(u.name, ': ', offered_card.name, ' + $', COALESCE(to3.additional_money, 0))
                FROM trade_offers to3
                JOIN users u ON to3.user_id = u.user_id
                JOIN card offered_card ON to3.offered_card_id = offered_card.card_id
                WHERE to3.trade_id = t.trade_id
                ORDER BY to3.additional_money DESC, to3.timestamp DESC LIMIT 1) as best_offer
        FROM trade t
        JOIN card c ON t.card_id = c.card_id
        LEFT JOIN trade_offers to2 ON t.trade_id = to2.trade_id
        WHERE t.user_id = %s AND t.end_time > NOW()
        GROUP BY t.trade_id
        ORDER BY t.end_time ASC
    """, (session['user_id'],))
    active_trades = cursor.fetchall()

    # Fetch offers for each trade
```

```python
# Fetch user's active auctions
cursor.execute("""
    SELECT a.*, c.name as card_name, c.value,
           (SELECT MAX(bid_amount) FROM bids_in WHERE auction_id = a.auction_id) as current_bid,
           (SELECT COUNT(*) FROM bids_in WHERE auction_id = a.auction_id) as bid_count
    FROM auction a
    JOIN card c ON a.card_id = c.card_id
    WHERE a.user_id = %s AND a.end_time > NOW()
    ORDER BY a.end_time ASC
""", (session['user_id'],))
active_auctions = cursor.fetchall()

# Fetch user's ended auctions
cursor.execute("""
    SELECT a.*, c.name as card_name, c.value,
           (SELECT MAX(bid_amount) FROM bids_in WHERE auction_id = a.auction_id) as winning_bid,
           (SELECT user_id FROM bids_in WHERE auction_id = a.auction_id
            AND bid_amount = (SELECT MAX(bid_amount) FROM bids_in WHERE auction_id = a.auction_id)) as winner_id
    FROM auction a
    JOIN card c ON a.card_id = c.card_id
    WHERE a.user_id = %s AND a.end_time <= NOW()
    ORDER BY a.end_time DESC
""", (session['user_id'],))
ended_auctions = cursor.fetchall()

cursor.close()
```

Chat System:

The chat system here is not an entity by itself but a recursive relationship. Hence, when both the users are present the chat gets activated and by starting the conversation a temporary space is created to store the data. The messaging happens in real-time. A sent text is stored in the database using sender user's ID or receiver user's ID and timestamp.

```python
# in-memory messages (reset on server restart)
live_chats = defaultdict(list)  # key: frozenset({user_id, recipient_id}), value: list of messages

@app.route('/chatbox', methods=['GET', 'POST'])
def chatbox():
    if 'user_id' not in session:
        flash('Please log in first!', 'danger')
        return redirect(url_for('login'))

    user_id = session['user_id']
    recipient_id = request.args.get('recipient_id', type=int)

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

    # Fetch current user info
    cursor.execute('SELECT * FROM Users WHERE user_id = %s', (user_id,))
    user = cursor.fetchone()

    recipient = None
    messages = []

    print('recipient_id: ', recipient_id)
    print('user_id: ', user_id)
```

```python
    if recipient_id and recipient_id != user_id:  # Prevent self-chat
        # Fetch recipient info
        cursor.execute('SELECT * FROM Users WHERE user_id = %s', (recipient_id,))
        recipient = cursor.fetchone()

        if recipient:
            # Update current user's chatuser_id and timestamp
            cursor.execute(
                'UPDATE Users SET chatuser_id = %s, timestamp = %s WHERE user_id = %s',
                (recipient_id, datetime.now(), user_id)
            )
            mysql.connection.commit()

            # if duijoni present
            if recipient.get('chatuser_id') == user_id:  # Use .get() to avoid KeyError
                # Both are connected tokhon we retrieve messages
                chat_key = frozenset({user_id, recipient_id})
                messages = live_chats.get(chat_key, [])
    else:
        print("Cannot chat with yourself!", "warning")
        return redirect(url_for('dashboard'))
```

```python
    # Fetch all users for sidebar (excluding current user)
    cursor.execute('SELECT user_id, name FROM Users WHERE user_id != %s', (user_id,))
    all_users = cursor.fetchall()

    cursor.close()

    return render_template(
        'chatbox.html',
        user=user,
        recipient=recipient,
        messages=messages,
        all_users=all_users,
        recipient_id=recipient_id
    )
```

```python
@app.route('/send_message', methods=['POST'])
def send_message():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    sender_id = session['user_id']
    recipient_id = request.form.get('recipient_id')
    content = request.form['content'].strip()

    # Validate input
    if not recipient_id or not content:
        flash("Missing recipient or message content!", "warning")
        return redirect(url_for('chatbox'))

    try:
        recipient_id = int(recipient_id)
    except ValueError:
        flash("Invalid recipient ID.", "danger")
        return redirect(url_for('chatbox'))

    # Prevent self-messaging
    if sender_id == recipient_id:
        flash("Cannot message yourself!", "warning")
        return redirect(url_for('chatbox', recipient_id=recipient_id))
```

```python
    # Store message in memory
    chat_key = frozenset({sender_id, recipient_id})

    if chat_key not in live_chats:
        live_chats[chat_key] = []

    live_chats[chat_key].append({
        'sender_id': sender_id,
        'content': content,
        'timestamp': datetime.now()
    })

    return redirect(url_for('chatbox', recipient_id=recipient_id))


@app.route('/stop_chat')
def stop_chat():
    """Clear current user's chat session and go back to dashboard"""
    if 'user_id' not in session:
        return redirect(url_for('login'))

    user_id = session['user_id']
    cursor = mysql.connection.cursor()
    cursor.execute('UPDATE Users SET chatuser_id=NULL WHERE user_id=%s', (user_id,))
    mysql.connection.commit()

    flash("You have left the chat.", "info")
    return redirect(url_for('dashboard'))
```

Contribution of ID : 23301105, Name : Sabbir Ahmed Kabbo

Source Code Repository

Github Repository:

https://github.com/ZarinTasnimNadia/CSE370L-Project-Gotta-Catchem-All

Conclusion

Gotta Catch'em All is a fun Pokemon Card Trading Website for all the Pokemon lovers out there. This project was a really fun experience and we got a lot to learn through making it. We also added a battle system to make it game-like and engaging. In the future, we want to make this website more refined and globally available for all the Pokemon lovers with more features. Wish us luck!!!

References

Not Applicable