🛒 Checkout and Chat System Workflow

✅ Frontend Flow (Customer Side)

Step: Proceed to Checkout

- When a customer clicks **"Proceed to Checkout"**, the frontend makes a `POST` request to the `/chat/rooms/` endpoint with the following payload:
    - `customer_id`
    - `farmer_id`
    - `product_id`
    - `quantity`
    - `is_post_checkout = True`
    - `product_image_url` *(optional)*

🧠 Backend Flow

1. Room Creation (`ChatRoomViewSet.create`)

- Backend checks if a chat room already exists between the customer and farmer for this product.
- If **no room exists**, it creates a new `ChatRoom` with:
    - Unique `room_id` (UUID)
    - References to customer, farmer, and product
    - Initial order status: `'new'`
    - `is_new_order = True`

2. Initial Message Creation

Backend automatically creates a default message:

```
Hello! I've just purchased [quantity] of [product].
I'd like to discuss delivery options...
```

-
- If a product image is provided, it is **attached** to the message.
- Sets `has_unread_farmer = True` to notify the farmer.

🔌 WebSocket Connection

- Frontend connects to WebSocket at: `ws/chat/<room_id>/`
- Uses **JWT token** for authentication.
- `ChatConsumer` verifies user access to the room.

💬 Real-time Chat Functionality

- Messages are sent and received via **WebSocket**.
- Each message is saved to the database.
- Chat includes:
  - Read/unread status updates
  - Order status updates (broadcast to both parties)

🚜 Farmer Notification

- Farmer sees the new chat in their **chat list**, including:
    - Blinking indicator (from `is_new_order`)
    - Unread message count
    - Product details and initial message preview

🔄 Order Status Flow

- Farmer can update order status through UI:
    - **New → Active → Completed**
- Each status change is:
    - Saved
    - Broadcast in real-time via WebSocket
    - Displayed in chat history

🧩 Key Components

📦 Models

- `ChatRoom`: Links customer, farmer, product
- `ChatMessage`: Stores messages with read status
- `ChatMessageImage`: Supports multiple image attachments
- `OrderStatus`: Enum for order states

🧮 Views

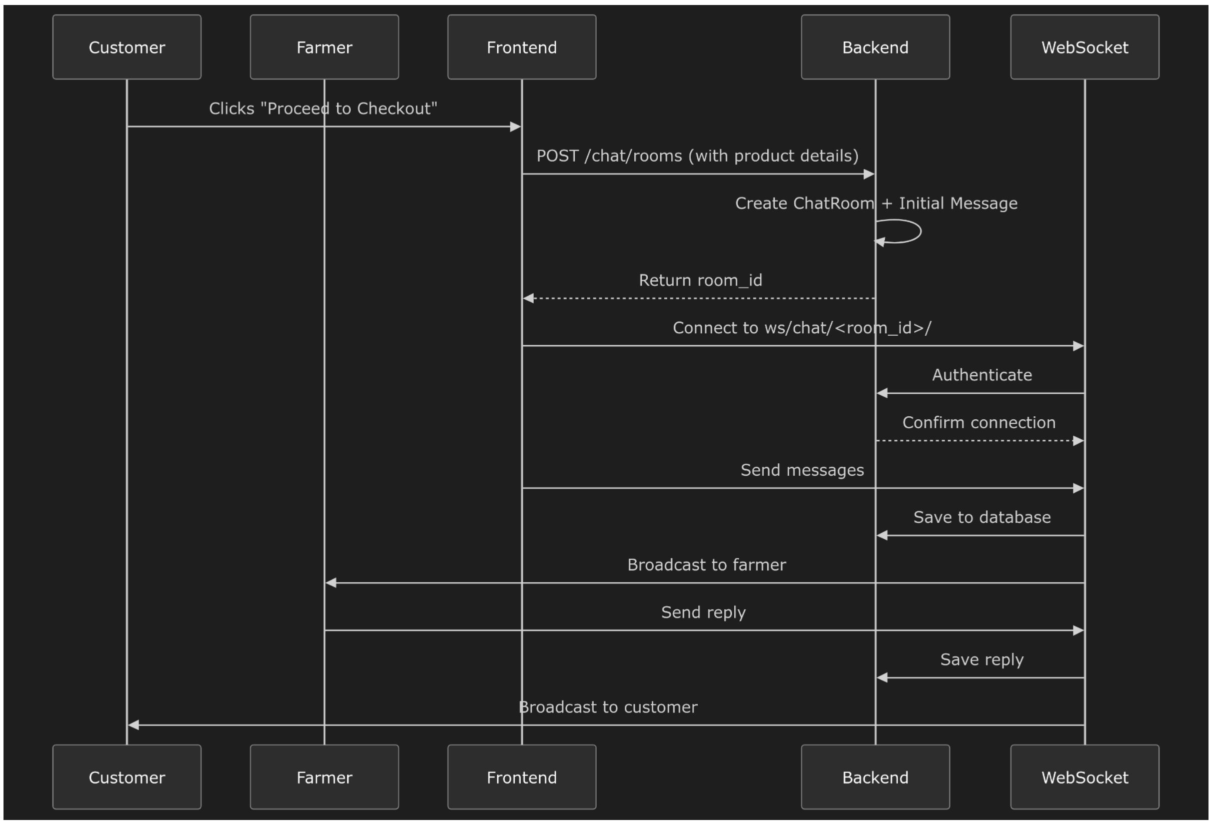
- `ChatRoomViewSet`: Handles room creation and management
- `ChatMessageViewSet`: Manages message CRUD operations

🌐 WebSocket

- `ChatConsumer`: Handles real-time messaging
- Uses **Django Channels** with **Redis backend**
- Supports:
    - Typing indicators
    - Read receipts

🔐 Security

- JWT authentication for:
    - HTTP requests
    - WebSocket connections
- Verifies:
    - Room access
    - User-specific permissions


Customer
Farmer
Frontend
Backend
WebSocket
Clicks "Proceed to Checkout"
POST /chat/rooms (with product details)
Create ChatRoom + Initial Message
Return room_id
Connect to ws/chat/<room_id>/
Authenticate
Confirm connection
Send messages
Save to database
Broadcast to farmer
Send reply
Save reply
Broadcast to customer
Customer
Farmer
Frontend
Backend
WebSocket