# Exercise 2: Real-time Chat Application

**Time Allocation: 40-50 minutes**

## Objective

Build a simple real-time chat application with WebSocket support where multiple users can send and receive messages instantly.

## Requirements

### Backend (Python with WebSocket support)

1. Use Flask-SocketIO or FastAPI with WebSockets
2. Implement:
   - User connection/disconnection handling
   - Message broadcasting to all connected users
   - Simple username system (no authentication required)
   - Message history (last 50 messages)
3. Message Format:

```python
{
    "username": "string",
    "message": "string",
    "timestamp": "ISO timestamp",
    "id": "unique_id"
}
```

### Frontend

1. Chat interface with:
   - Input field for username
   - Message input area
   - Message display area
   - Online users count
   - Auto-scroll to latest messages
2. Features:
   - Real-time message updates
   - Show "user is typing" indicator

- Timestamp for each message
- Different styling for own vs others' messages

## Evaluation Criteria

- [ ] WebSocket connection established
- [ ] Messages broadcast to all users
- [ ] No page refresh needed
- [ ] Clean and intuitive UI
- [ ] Proper event handling
- [ ] Error handling for disconnections

## Bonus Points

- Private messaging between users
- Emoji support
- Message notifications
- User avatars (using initials)
- Dark/light theme toggle

## Technical Notes

- Can use Socket.IO for easier implementation
- Consider CORS if frontend/backend on different ports
- Handle graceful disconnections