# ChatApp Java Swing Project Documentation

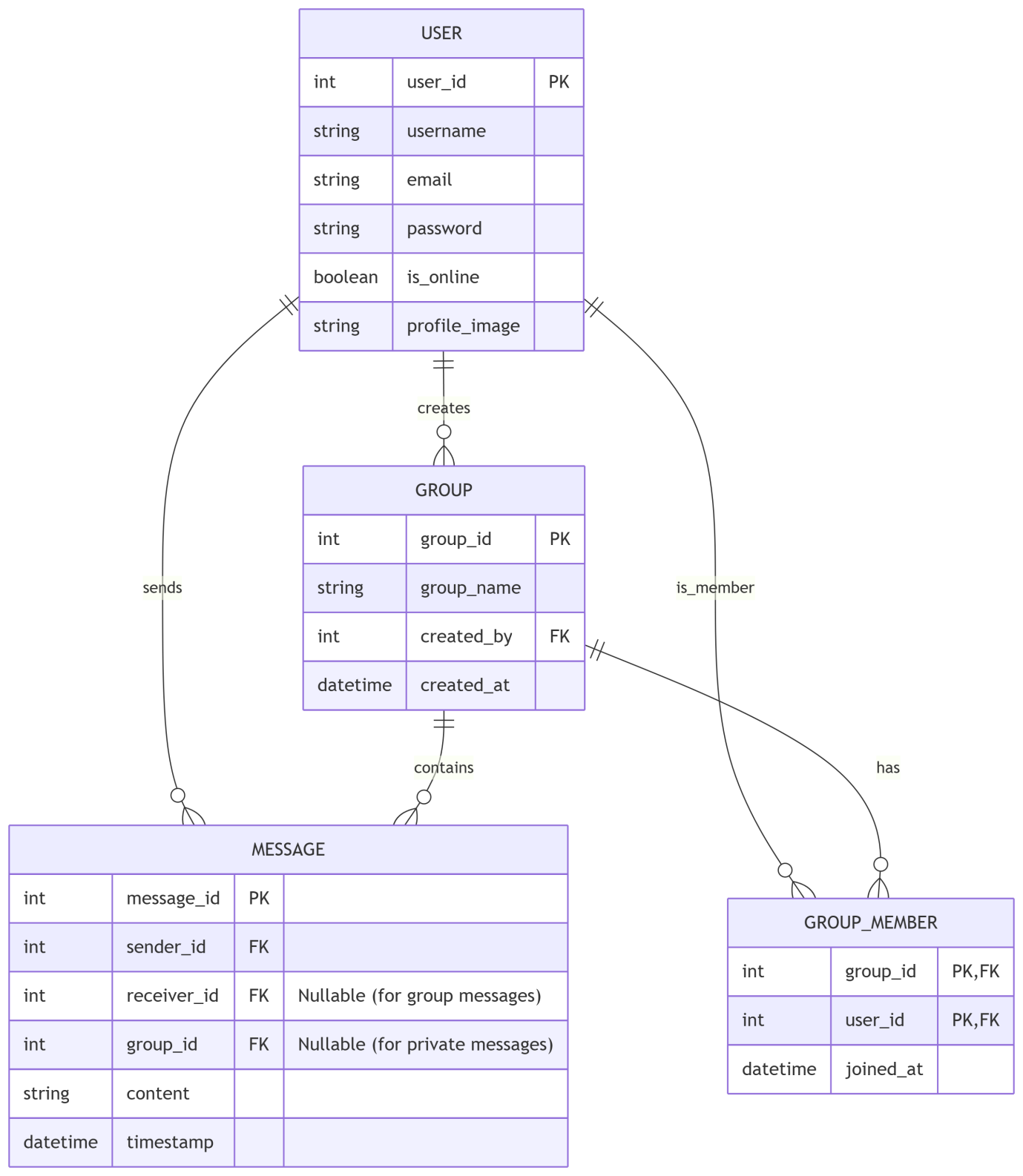
**1. Introduction**

The Real-Time Java Swing Chat Application is a desktop-based chat platform that enables users to communicate instantly with others. Built using Java Swing for the front-end and MySQL for the back-end, this application provides:

* User authentication (login/registration)
* Private messaging between users
* Group chat functionality
* Online status tracking
* Intuitive user interface with dark theme

The application follows a client-server architecture where multiple clients connect to a central server to exchange messages in real-time. The Java Swing GUI provides a responsive and user-friendly interface that works across Windows, macOS, and Linux systems.

1. **Entity-Relationship Diagram**



**3. Database Schema**

**User Table**

| Column | Type | Constraints | Description |
| --- | --- | --- | --- |
| user\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique user identifier |
| username | VARCHAR(50) | UNIQUE, NOT NULL | User's display name |
| email | VARCHAR(100) | UNIQUE, NOT NULL | User's email address |
| password | VARCHAR(255) | NOT NULL | Hashed password |
| is\_online | BOOLEAN | DEFAULT FALSE | Online status |
| profile\_image | VARCHAR(255) | DEFAULT NULL | Path to profile image |

**Group Table**

| Column | Type | Constraints | Description |
| --- | --- | --- | --- |
| group\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique group identifier |
| group\_name | VARCHAR(100) | NOT NULL | Group display name |
| created\_by | INT | FOREIGN KEY (user\_id) | User who created the group |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |

**Message Table**

| Column | Type | Constraints | Description |
| --- | --- | --- | --- |
| message\_id | BIGINT | PRIMARY KEY, AUTO\_INCREMENT | Unique message identifier |
| sender\_id | INT | FOREIGN KEY (user\_id) | Message sender |
| receiver\_id | INT | FOREIGN KEY (user\_id) NULL | Private message receiver |
| group\_id | INT | FOREIGN KEY (group\_id) NULL | Group message target |
| content | TEXT | NOT NULL | Message content |
| timestamp | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Message timestamp |

**Group Member Table**

| Column | Type | Constraints | Description |
| --- | --- | --- | --- |
| group\_id | INT | PRIMARY KEY, FOREIGN KEY | Group identifier |
| user\_id | INT | PRIMARY KEY, FOREIGN KEY | Member user identifier |
| joined\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Join timestamp |

**4. Relations with Normalization**

* The database design follows 3rd Normal Form (3NF):

1. **First Normal Form (1NF):**

* All tables have primary keys
* All columns contain atomic values
* No repeating groups

1. **Second Normal Form (2NF):**

* All non-key attributes fully functionally depend on the primary keys
* Group\_Member table has a composite primary key (group\_id + user\_id)
* No partial dependencies

1. **Third Normal Form (3NF):**

* No transitive dependencies
* All non-key attributes are non-transitively dependent on the primary keys
* Example: In User table, profile\_image depends only on user\_id

**5. Relationships:**

* **One-to-Many:**
* User to Message (sender)
* User to Group (creator)
* **Many-to-Many:**
* User to Group (through Group\_Member)
* **One-to-Many:**
* Group to Message

**6. List of Functions**

* **User Management**
* registerUser(): Creates a new user account
* loginUser(): Authenticates user credentials
* logoutUser(): Terminates user session
* updateUserStatus(): Toggles online/offline status
* **Messaging**
* sendPrivateMessage(): Sends message to another user
* sendGroupMessage(): Broadcasts message to a group
* getChatHistory(): Retrieves message history between users
* getGroupChatHistory(): Retrieves group message history
* **Group Management**
* createGroup(): Creates a new chat group
* addUserToGroup(): Adds member to existing group
* getUserGroups(): Lists groups a user belongs to
* getGroupMembers(): Lists members of a group
* **UI Functions**
* showLoginView(): Displays authentication screen
* showChatView(): Main chat interface
* showGroupView(): Group management interface
* showProfileView(): Profile management interface
* updateOnlineStatusUI(): Visual indicator for user status
* switchSidebarView(): Toggles between users/groups lists

**7. Conclusion**

The Java Swing Chat Application provides a robust, real-time messaging solution with an intuitive user interface. It successfully achieves cross-platform compatibility by running on any system with Java, features a responsive dark-themed UI with a clean, modern design, and enables instant message delivery for real-time communication. The application supports both private and group chats, along with user-friendly profile management including customization profile pictures. It demonstrates effective implementation of Java Swing for GUI development, MySQL for data persistence, a client-server architecture, the MVC design pattern, and sound database normalization principles. Looking ahead, future enhancements could include message encryption for enhanced security, file sharing, message read receipts, typing indicators, and even video or voice chat integration. Overall, this project serves as a comprehensive example of building a full-featured desktop application using Java Swing while adhering to best practices in software design, UI development, and real-time communication systems.