**Design Document – bootChat**

1. **Purpose:**

An online text chatting application which allows users to exchange messages with each other. The purpose of this chatting app is to provide an alternative chatting experience to the users from the phone to computer.

1. **Architecture:**

The bootChat application will use the format as that of a layered architecture. The high-level design of the system is as follows:

**Go LanguageServer**

**SQLite database**

**TCP**

**Web Socket**

**JSON**

**TCP**

**Web Socket**

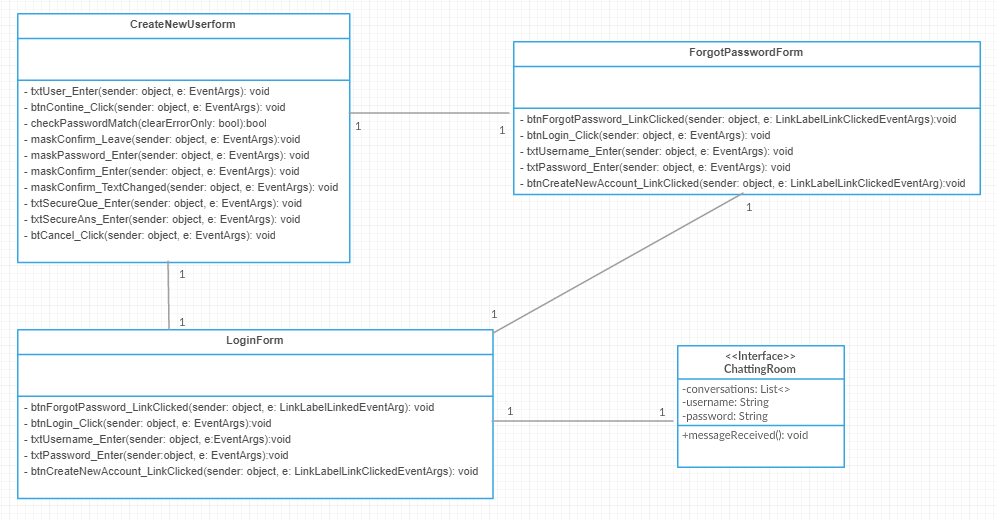
**Client GUI**

All the data transfer between the client and server is done through JSON

**Database management**

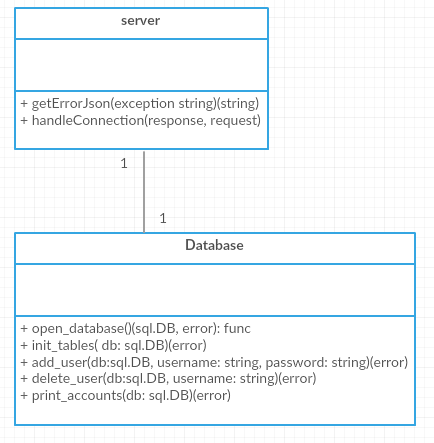
1. **UML Class Diagrams**

**Client-side Class Diagram:**



*Figure 1.1 C# Client-side UML Diagram*

**Server-Side Class Diagram:**



*Figure 1.2 Go Server-side UML Diagram*

1. **Interaction (Use Cases instead of Sequence diagram)**

Use-cases for common scenarios

:)

|  |  |
| --- | --- |
| **Scenario 1** | |
| Description | Every user can have one to one conversation with each other |
| Preconditions | Both users should be on each other’s contact list |
| Use-case | 1. user A register for his account  2. user B register for her account  3. they both add each other on their contact list  4. user A sends message to user B through the chatroom  5. user B receives message from user A and reply to it. |
| Postconditions | The messages exchanged from user A and user B will be stored in the database |

:)

|  |  |
| --- | --- |
| **Scenario 2** | |
| Description | Each user can retrieve messages from the chatting history |
| Preconditions | Both users should be on each other’s contact list, and never delete each other’s message from the interface. |
| Use-case | 1. user A click user B’s message, it will show previous message which had been send couple of hours ago.  2. user A scroll up and it shows previous conversation is loading  3. few seconds later use A see the previous conversation from user B |
| Postconditions | The messages will be retrieved from database as long as user A did not delete message from the database. |

:)

|  |  |
| --- | --- |
| **Scenario 3** | |
| Description | User forgot their password and try to find their password |
| Preconditions | Users already registered their account |
| Use-case | 1. user A tries to log in to his account.  2. He clicks forgot password  3. It takes him to another interface  4. He has to verify his username, security questions and security answers.  5. After he got all those right, he has to create new password, and verify his new password.  6. System shows he changed his password successfully. |
| Postconditions | The new password will be updated in the database associated with his username. |