1. GUI should contain a login system
   1. First screen when loading up the GUI should be a login screen
      1. Should contain an input for the username
      2. Should contain an input for the password
      3. Button to submit data to server
   2. When login data is sent to the server, the server should authenticate the user
      1. Receive JSON data from client and parse into a C# object
      2. Query database with username and extract password hash and salt
      3. Hash and salt received authentication code then compare with stored hash
      4. If they are equal then generate a session token, save it in the database and send the session token to the client
      5. If they aren't equal, then return an error message
   3. GUI should display different items based on response from authentication server
      1. Parse JSON response into C# object
      2. Display error if necessary
         1. Extract error message from object
         2. Create red message box
         3. Set message text to extracted error message
         4. Display red message box on screen
      3. If authentication is successful, take user to main screen
         1. Close current form
         2. Open main screen form

### Objective 2

1. Server should authenticate requests
   1. When a client sends a command to the server, they should include their username and session key
   2. The server should check if these values are correct before performing the command
      1. Parse JSON into C# object
      2. Query database with username and extract session token with the expiration date
      3. Make sure token is valid
         1. Compare expiration date with current time, if the current time is greater than the expiration date then the token is invalid
         2. Compare the token sent with the token in the database, if they are equal then token is valid
      4. If token is valid, then perform command
      5. If token is invalid, then return an error message

### Objective 3

1. Messages between two people should be end-to-end encrypted
   1. When a new chat is created, create a shared secret key using X3DH
      1. Contact the server for a pre-key bundle
      2. Verify the pre-key signature
      3. Generate ephemeral key-pair
      4. Perform 4 Diffie Hellman key exchanges with pre-key bundle
      5. Generate shared key from the 4 key exchanges
      6. Send public ephemeral key, pre key bundle ID and public identity key to recipient
      7. Recipient follows this process in reverse to generate the same shared secret key
   2. When a message is sent, encrypt it using a KDF chain
      1. Generate a new keypair
      2. Perform Diffie Hellman with new private key and recipients signed-pre key (if first message, otherwise use last public key sent by the recipient)
      3. Take the X3DH and new Diffie Hellman values as inputs to the KDF function
      4. Use generated message key and message ID as input to KDF function
      5. Use new generated message key to encrypt the message
      6. Send the public key along with the message and the message ID to the recipient
   3. When a message is received, decrypt it using a KDF chain
      1. Extract public key from message
      2. If the message is the first message sent, perform Diffie Hellman with the received public key and the private signed pre-key, otherwise perform the Diffie Hellman with the received public key and the last generated private key
      3. Use the X3DH key and the new key as input to the KDF chain
      4. Use generated message key and message ID as input to KDF function
      5. Use new generated message key to decrypt the message