Authentication Protocol (7 marks)

1. ~~Create an authentication protocol that uses Google Protocol Buffers​ as its serialization and deserialization method (2 marks)~~
2. ~~Must use the same proto files on server and client (2 marks)~~
3. ~~Must implement a protocol similar to the one below ​(3 marks)~~

Here is an example protocol that you may use as a reference (in pseudo code)

message ​CreateAccountWeb {

long ​requestId;

string ​email;

string ​plaintextPassword;

}

message ​CreateAccountWebSuccess {

long ​requestId;

long ​userId;

}

message ​CreateAccountWebFailure {

long ​requestId;

enum ​reason {

ACCOUNT\_ALREADY\_EXISTS;

INVALID\_PASSWORD;

INTERNAL\_SERVER\_ERROR;

}

}

message ​AuthenticateWeb {

long ​requestId;

string ​email;

string ​plaintextPassword;

}

message ​AuthenticateWebSuccess {

long ​requestId;

long ​userId;

string ​creationDate;

}

message ​AuthenticateWebFailure {

long ​requestId;

enum ​reason {

INVALID\_CREDENTIALS;

INTERNAL\_SERVER\_ERROR;

}

}

Authentication Service Database (3 marks)

1. ~~Create a table ‘web\_auth’ (1 mark)~~
   1. id BIG INT AUTO\_INCREMENT;
   2. email VARCHAR(255);
   3. salt CHAR(64);
   4. hashed\_password CHAR(64);
   5. userId BIGINT;
2. ~~Create a table ‘user’ (1 mark)~~
   1. id BIG INT AUTO\_INCREMENT;
   2. last\_login TIMESTAMP;
   3. creation\_date DATETIME;
3. ~~ID should be your primary key, add indexes to appropriate columns ​(1 mark)~~

Authentication Service (17 marks)

This must be a separate application than your Chat Server

1. ~~Must use TCP ​(1 mark)~~
2. ~~Must use length prefix header for serialization ​(3 marks)~~
3. Must be able to create a new account
   1. Must use SHA256 as the hash algorithm ​(2 marks)
   2. Must use a randomized salt for EACH password ​(2 marks)
   3. Must add this account to the MySQL Database (2 marks)
   4. Must respond with a failure reason on failure **(**1 mark)
   5. Must respond with “success” on success (1 mark)
4. Must be able to authenticate an account
   1. Must hash the plaintext with the SHA256 ​(1 mark)
   2. Must compare this hash to the database hash properly ​(1 mark)
   3. Must respond with a failure reason on failure ​(1 mark)
   4. Must respond with “success” on success (1 mark)
   5. Must update the ‘last\_login’ column in the ‘user’ table in MySQL ​(1 mark)

Authentication Client (7 marks)

This must be the same program as your Chat Server

1. ~~Must use TCP ​(1 mark)~~
2. ~~Must use length prefix header for serialization (3 marks)~~
3. ~~Must connect to the authentication service (1 mark)~~
4. ~~Should be able to create a new account (1 mark)~~
5. ~~Should be able to authenticate a user ​(1 mark)~~

Chat Client​ ​(6 marks)

1. ~~Should be able to send a command: REGISTER email password~~ **~~(~~**~~1 mark)~~
2. ~~If registration was successful, it should tell the client: “Registration Successful” (1 mark)~~
3. ~~If registration failed, it should tell the client the reason for the failure (1 mark)~~
4. ~~Should be able to send a command AUTHENTICATE email password~~ **~~(~~**~~1 mark)~~
5. ~~If authentication was successful, it should say “Authentication successful, account created on [DATE IN DATABASE]” ​(1 mark)~~
6. ~~6. If authentication failed, it should tell the client the reason for the failure ​(1 mark)~~

~~Assignment must be done in Git. Commit messages should be small and sweet (3 marks)~~

Bonuses (10 marks)

Bonuses are only available if all other project requirements are completed properly.

1. Chat client features (e.g. join room, leave room, send message) should not be functional unless the client is authenticated ​(5 marks)
2. Chat client authentication fails if another client is currently connected and authenticated with that user id.​ ​If the user disconnects or logs out, then that user should be able to immediately be allowed to be authenticated from another client ​(5 marks)