Due Date: Nov 10, 11:59PM EST

/ 40 Marks

Introduction

Create an authentication service that provides a way for your chat server to authenticate users. This authentication service should be reusable. You must use C++.
Assignment can be done in groups up to 3.

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 the 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)**
    1.1. id BIG INT AUTO_INCREMENT;
    1.2. email VARCHAR(255);
    1.3. salt CHAR(64);
    1.4. hashed_password CHAR(64);
    1.5. userId BIGINT;

2. Create a table `user` **(1 mark)**
    2.1. id BIG INT AUTO_INCREMENT;
    2.2. last_login TIMESTAMP;
    2.3. creation_date DATETIME;

ID should be your primary key, add indexes to the appropriate columns. **(1 mark)**

Authentication Service (17 marks):

1. Must use TCP **(1 mark)**

2. Must use lengthprefix header for serialization **(3 marks)**

3. Must be able to create a new account **(8 marks total)**
    3.1. Must use SHA256 as the hash algorithm **(2 marks)**
    3.2. Must use a randomized salt for EACH password **(2 marks)**
    3.3. Must add this account to the MySQL Database **(2 marks)**
    3.4. Must respond with a failure reason on failure. **(1 mark)**
    3.5. Must respond with "success" on success **(1 mark)**

4. Must be able to authenticate an account **(5 marks total)**
    4.1. Must hash the plaintext with SHA256 (**1 mark)**
    4.2. Must compare this hash to the database hash properly **(1 mark)**
    4.3. Must respond with a failure reason on failure **(1 mark)**
    4.4. Must respond with success on success **(1 mark)**
    4.5. Must update the `last_login` column in the `user` table in MySQL **(1 marks)**

Authentication Client (7 marks):

1. Must use TCP (**1 mark)**
2. Must use lengthprefix 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 send a command: REGISTER email password **(1 mark)**
2. If registration was successful, it should tell that client: "Registration successful" **(1 mark)**
3. If registration failed, it should tell that 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. If authentication failed, it should tell that client the reason for the failure **(1 mark)**

Due Date: Nov 10, 11:59PM EST