CS 580 Client-Server Programming
Fall Semester, 2012
Project Description

San Diego State University -- This page last updated 10/27/12

Video with Questions

Version 1.2

While students remotely videos of lectures they may have questions about the lecture. So we will build a client server system that allows students to do that. When students listen to a video lecture and have a question they need to be able to stop the video and enter the question. The question and the location in the video is sent to the server. For a given video a user needs to be able to see the list of questions for that video. There needs to be a way to select a question and play the video from the point the question was asked (if not a bit before). If someone knows the answer they need to be able submit an answer. When one selects a question they need to be able to see all answers for the question. Of course other students can be asking and answering questions all the time.

Protocol

The general format of a command is:

```
command;key1:value1;key2:value2;…;keyN:valueN;;
```

Key-value pairs can be in any order. The characters ':', ';' and '\' are special. If they occur in a value they must be escaped with a '\' before it is sent. For example if a user's password is 'cat;dog' then it must be converted to 'cat\;dog' before being sent to the server. The server must convert all values sent to it by removing the escape character '\'. This means that if the password is 'cat\dog' it must first be converted to 'cat\\dog' before being sent to the server. The server will strip out the first '\' to convert it back to 'cat\dog'. All values (not keys) returned by the server have the special characters escaped. The client must unescape the values.

In key-value pairs the keys are not case sensitive. That is the key 'id' is the same as the key 'ID'. Values are case sensitive. The 'command' is also not case sensitive. That is the command 'courseList' is the same as 'courselist'

Command sent to the server and responses returned from the server all end in two semicolons ';;'.

Server responses either start with 'ok:' or 'error:'. If 'error:' is returned the value following the colon ':' is a description of the error. What follows the 'ok;' depends on what command is it in response to.

All ids are string representation of a long. The ids increase over time. So given two videos the earlier video will have a lower id.

All commands sent to the server and responses from the server are characters encoded using UTF 8.

Server States

The server has three states: Start, Nonce, Authenticated. Each connection to the server starts in the Start state. Server states are per connection. That is the server can be in the Start state for one connection and the Authenticated state for another connection.

In the Start state only the login and nonce commands are valid. If the login command is successful the server moves to the Authenticated state. If the login command is not successful the server remains in the Start state. If the nonce command is successful the server moves to the Nonce state. If the nonce command is not successful the server remains in the Start state.

In the Nonce state only the safeLogin and logout commands are valid. If the safeLogin command is successful the server moves to the Authenticated state. If the safeLogin command is not successful the server remains in the Nonce state. If the logout command is successful the server returns a response and closes the connection. If the logout command is not successful the server remains in the Nonce state.

In the Authenticated state all commands are valid except: login, safeLogin, nonce. Once in the Authenticated state the server stays in the Authenticated state until the connection is closed.

Errors

The test server will close the connection after three errors on the same connection. When the server detects the third error it send the message 'ok:quit;;' and closes the connection. There are three types of errors: invalid syntax in a command, invalid data (wrong password for example) and internal server errors. It is difficult for the server to recover from invalid command syntax. Once one of those errors occurs it is likely that the next commands sent to the server on the same connection will result in an errors. The server can handle errors caused by invalid data so commands sent after those will be processed correctly. Hopefully internal server errors are very rare.

Whitespace

In the commands sent to the server whitespace can be added before and after a colon ':' and a semicolon ';'.

Commands

Login

Format: login;id:userID;password:userPassword;;

Where userID is the users id, and userPassword is the users password in plain text.

The server either returns 'ok:success' or and error message ('error:error description').

courseList

This is to request a list of the courses the current user is in.

Format: courseList;;

Response: ok:N;name:CourseName1;id:CourseID1;CR
name:CourseName2;id:CourseID2;CR...name:CourseNameN;id:CourseIDN;;

N is the number of courses the user is enrolled in. Each course has a name and an id. This information is sent back using key-value pairs. Courses are separated by CR (carriage return character)

Sample response with 1 course: ok:1;name:CS 101;id:1;;

Sample response with 2 courses: ok:2;name:CS 101;id:1;
name:CS 202;id:2;;

Error response: error:Description

Where Description is a description of the error.

videoList

Request a list of videos for a specific course. Each video has a name, an id, a date, and a url. The date is given in milliseconds since Jan 1, 1970 00:00:00 GMT. The url is the url of the actual video file which is in mp4 format.

Format 1: videoList;course:courseID;;

Format 2: videoList;course:courseID;after:aVideoID;;

Where courseID is the ID of the course you want the videos from. If 'after:aVideoID' is given the list only contains videos with ids that are higher (later) than the id given. 'after' is optional.

Sample: videoList;course:11;;

Sample: videoList;course:11;after:5;;

Response: ok:N;name:VideoName1;id:VideoId1;date:date1;url:url1;CR
name:VideoName2;id:VideoId2;date:date2;url:url2;CR…
name:VideoNameN;id:VideoIdN;date:dateN;url:urlN;;

Where N is the number of videos in the list. Again each video in the list is separated by a carriage return (CR - ascii 13). See sample below for example values.

Sample response with one Video: ok:1;name:Course
Intro;id:1;date:1346112000000;url:http://www-rohan.sdsu.edu/~whitney/audio/courses/fall12/cs580/cs580_08_28_12.mp4;;

Sample response with no videos: ok:0;;

Sample Error response: error:No such course;;

questionList

Request a list of questions asked for a specific video. Each question has an id, text, a time,a timestamp and a number of answers. The 'text' is the text of the question. It is limited

to 1024 characters. The time is the number of milliseconds from the start of the video that the user had the questions. The timestamp is the date and time that the question was submitted to the server. The timestamp is in milliseconds since Jan 1, 1970 00:00:00 GMT.

Format 1: questionList;video:videoID;;

Format 2: questionList;video:videoID;after:questionID;;

Where videoID is the ID of the video you want the questions from. If 'after: questionID' is given the list only contains questions with ids that are higher (later) than the id given. 'after' is optional.

Sample 1: questionList;video:1;;

Sample 2: questionList;video:1;after:46;;

Sample Response with one question: ok:1;id:47;text:When is the exam?;time:123;timestamp:1349824611927;answers:0;;

Sample Response with three questions: ok:3;id:45;text:a;time:123;timestamp:1349824611927;answers:0;
id:46;text:b;time:123;timestamp:1349828611927;answers:2;
id:47;text:c;time:123;timestamp:1349924611927;answers:9;;

Sample Error Response: error: No such Video;;

questionAdd

Used to add a question. Need to provide the text of the question, the id of the video the question is about and the number of milliseconds from the beginning of the video where the question occurs.

Format: questionAdd;video:videoID;text:QuestionText;time:MillisecondsFromStart;;'

Where videoID is the id of the video the question is being asked about. Question text is the text of the question. MillisecondsFromStart is the milliseconds from the start of the video to the spot the question is being asked.

Sample: questionAdd;video:1;text:cat in hat;time:123;;'

Sample response: ok:success;;

Sample response: error:Time must be valid positive integer;;

answerList

Requests a list of answers to a particular question. Each answer has an id, text and a timestamp. The timestamp is the date and time that the question was submitted to the server. The timestamp is in milliseconds since Jan 1, 1970 00:00:00 GMT. The 'text' is the text of the answer.

Format: answerList;question:QuestionID;;

Format: answerList;question:QuestionID;after:AnswerID;;

Need to provide the id of the question (QuestionID) you want the answers for. There is an optional 'after' key to request answers with id (AnswerID) later than a given answer.

Sample: answerList;question:45;;

Sample: answerList;question:45;after:47;;

Sample Response: ok:3;id:46;text:b;timestamp: 1349824611927;
id:47;text:c;timestamp: 1349828611927;
id:48;text:d;timestamp: 1349924611927;;

As in all list commands the value of ok indicated the number of answers in the list. Each answer contains a set of key-value pairs. The key-value pairs are separated by a carriage return.

answerAdd

Add an answer to a question. Must provide the id of the Question being answered (QuestionID) and the text of the answer (TextOfQuestion).

Format: answerAdd;question:QuestionID;text:TextOfQuestion;;

Sample: answerAdd;question:45;text:I think the exam is soon;;

Sample Response: ok:success;;

logout

Ends the connection with the server.

Sample: logout;;

Sample Response: ok:success;; and server closes the connection.

Alternative Login

The client first sends the nonce command to the server. The server returns a nonce string. The client then computes the MD5 hash of the password + nonce. The client then returns the string representation of the MD5 hash.

nonce

Requests a nonce from the server.

Format: nonce;;

Sample Response:
ok:1098A07717746BE1357D16A8C52DAC2B4B0A4B5B663803C04C84E70C9785662214
308AA517D31A6D9833545989455;;

safeLogin

After receiving a nonce from the server the client response with a safeLogin. The client computes the MD5 hash of password+nonce and converts the resulting byte array into a hex number. The hex number is returned to the server using the safeLogin command along with the user id. If the command is successful the user is logged in.

Format: safelogin;id:UserId;hash:HexOfMD5Hash(password+nonce);;

Sample: safelogin;id:333;hash:F58CB595B0FBBD385317A9294939B36B;;

Sample Response: ok:success;;

Sample Error Response: error:Invalid password;;

Testing

To aid in testing clients the class server has a reset command. When the command is sent to the server all questions and answers sent to the server are removed. Each student has a separate database of questions and answers. When you send the reset command only your database is changed.

Reset

Format: reset;;

Sample Response: ok:success;;

Versions

1.2 Corrected typo in response to VideoList, replaced CourseName and CourseId by Video-Name and VideoId.

1.1 Corrected error in response to courseList.