Audio Stream Project

David S. McDermott, Grace M. Lin
Faculty Advisor: Prof. Nannette D'Imperio
The Pennsylvania State University, Brandywine

Abstract

The objective of this project was to create a desktop application using Java to play MP3 audio files using JavaFX, and an accompanying website with the capability to stream audio over a secure socket layer. The application has the capability to search device folders to find MP3 files and then play them back with user controls for volume as well as pausing, playing, and stopping audio. Users have the capability of adding a device folder to a list of device folders that the application will search for audio in. Perhaps more importantly, users have the ability to add network locations that can serve audio files for the program to play without long term storage on the user's machine. For the server side of the project, a standard LAMP server was constructed using Ubuntu, Apache, MySQL, and PHP. The website renders on a standard web browser, allowing users to register an account and then log in to download the desktop application. The website was secured using the Let's Encrypt plugin for Apache, which forces the webserver to operate on port 443 using HTTPS rather than HTTP on port 80, installs an SSL certificate and configures the webserver to run RSA 2048-bit encryption on all communications. User login information and song metadata was stored in the MySQL database; for security, user passwords are stored as a salted hash using the Blowfish Algorithm in PHP. Overall, a functional website and desktop application were created. The desktop application plays back local and remote MP3 files successfully, and the webserver allows for retrieval of the remote MP3 files by the desktop application.

Notes on the Client

The Java client asynchronously downloads MP3s and then adds them to the song list view. This is accomplished by using an HTTPS connection in a new thread for each song that is downloaded. The songs are stored in a system temporary directory. This makes closing the application more complicated because Mac OS and Windows do not clean up their temporary directories like Linux does, so all files and directories are recursively deleted on the program's exit.

The Blowfish Algorithm

The Blowfish Algorithm was selected to hash passwords before storing them in the MySQL database. Blowfish is a 64-bit algorithm and uses a variable cipher length between 32 and 448 bits. Blowfish is less commonly used than MD5 and SHA, which makes is more secure for password storage, as fewer rainbow tables have been generated. PHP provides a Blowfish implementation that creates salted hashes and outputs a single string with the algorithm used, the salt, and the hash. This allows the PHP hash check to evaluate a comparison of a password that was input using the correct algorithm and salt by parsing the whole hash output.

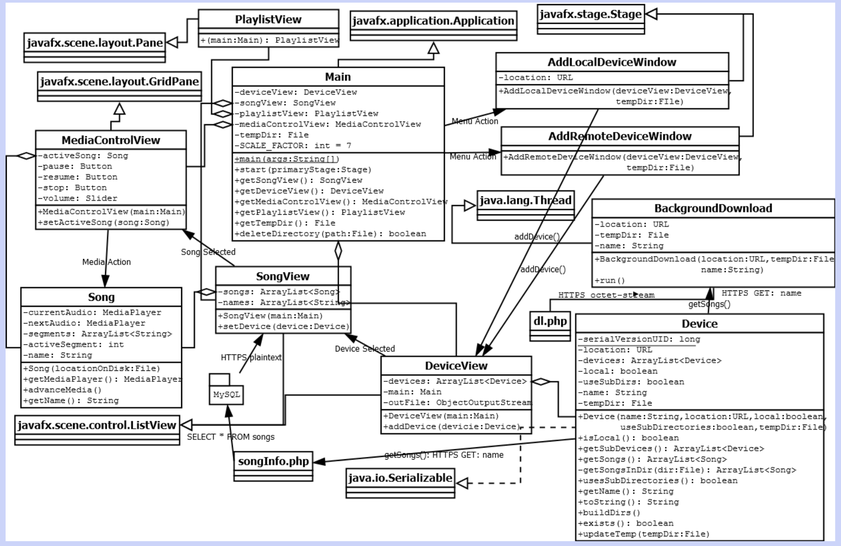

Figure 3: UML Class Diagram for Application

```
<!DOCTYPE html>
<html>

<head><title>Project Stream</title></head>

<?php

include 'header.php';

$name = $_POST["name"];
$email = $_POST["email"];
$password = $_POST["password"];
$dbc = new mysqli("localhost","php","PHP7@localhost","project_stream");
echo "2";
$name = $dbc->real_escape_string($name);
$email = $dbc->real_escape_string($email);
$crypt = password_hash($password, PASSWORD_DEFAULT);
if($dbc->connect_errno){
        echo "<br><br><br>Connection failed: ".$dbc->connect_error;
        exit;
}
echo "3";
//$dbc->beginTransaction();
$dbc->autocommit(true);
echo "4";
$sql = "INSERT INTO login (email, password, name) values('$email','$crypt','$name');";
$dbc->query($sql);
echo "5";
echo $dbc->error;
//$dbc->commit();
echo "6";
$dbc->close();
echo "7";
?>

</html>
```

Figure 1: PHP Register Page

Figure 2: Java Client

HTTPS

HTTPS adds a layer of security over the standard Hyper-Text Transfer Protocol (HTTP) through the use of a Secure Socket Layer (SSL). For this project, Let's Encrypt was used to configure Apache to force HTTPS connections and refuse HTTP. This was achieved by blocking port 80, and forcing connections over port 443. The server communicated over RSA 2048-bit encryption, which protects against a man in the middle attack. RSA encryption is an asymmetric encryption, which means that any data sent from a server to a client is encrypted with a private key and decrypted with a public key, so anyone can read it, but data sent from a client is encrypted using the public key, so only the private key holder can decrypt the data, which protects user data like passwords.

MP3

MPEG-3 Layer audio files are playable using most computer systems that have audio capabilities. Java implements several different ways to play MP3 files, for this application, JavaFX's MediaPlayer class was used. MP3s are comprised of several components. Perhaps the most interesting component is the MP3 frame, which can act as self contained MP3 files and are playable as short audio clips. This means that in theory, MP3 frames can be sent and played in rapid succession to stream MP3 files without necessarily downloading the entire MP3 file. Unfortunately, in reality, splitting MP3s along frames requires a deeper understanding of the MP3 format than was attainable without a license for the official documentation.

Conclusion

The audio streaming application does work to play back audio from both local and remote sources. All network communications were successfully encrypted using SSL and the database layer integrates with PHP in a reasonable manner such that all relevant metadata can be retrieved and communicated to the client. Overall, the application works as intended other than the fact that MP3 files were not parsed out into frames before sending to the client to playback faster.