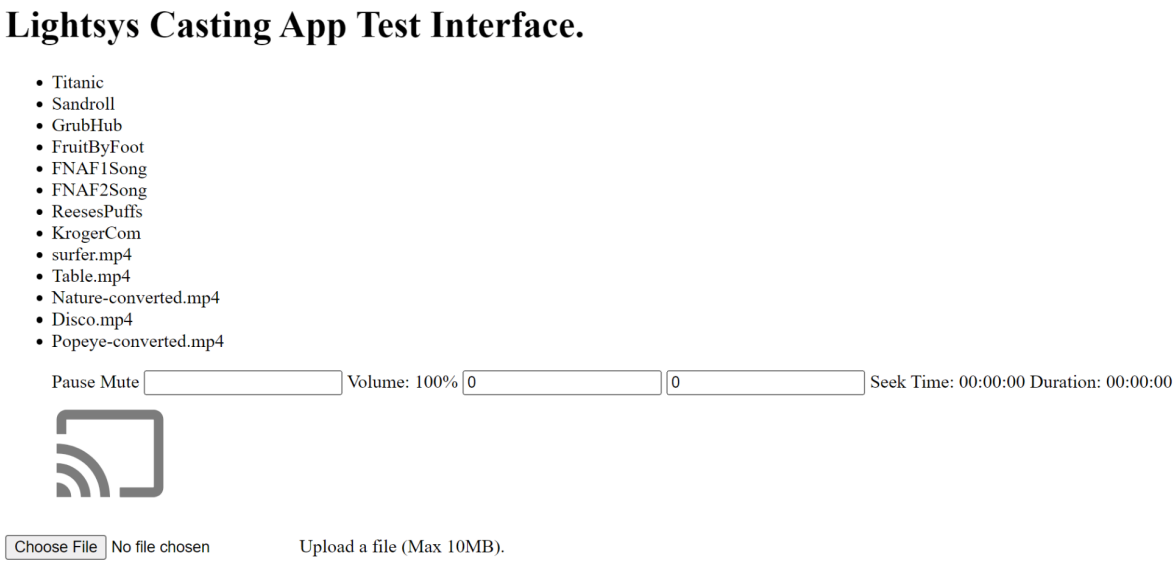

Functionality:

+ Connect to a Google Cast device, by pressing the cast button.
+ Play a custom mp3 or mp4 file by clicking the title of a previously uploaded file.
+ Control the video (play, mute, volume, seek).
+ Upload files to the file database.

Files:

+ index.html: The web app's user interface.
+ css/index.css: Simple styling for the user interface.
+ js/index.js: Loads content from the database using php/getFiles.php and creates links to run them on the media player.
+ js/initCastPlayer.js: Initializes the cast player by setting necessary API parameters.
+ js/initControls.js: initializes the controls for the video by linking callbacks with the API.
+ js/uploadFile.js: adds an event listener for file upload; when this happens, php/uploadFile.php is POSTed to with the uploaded file.
+ php/getFiles.php: Loads content from the MediaItems/fileReg.db database and returns it in JSON.
+ php/uploadFile.php: Uploads the file posted to it to the MediaItems directory on the web server, and creates a corresponding record in MediaItems/fileReg.db.
+ MediaItems/fileReg.db: A sqLite database containing data concerning uploaded files to be loaded into the app.

Google Cast API:

+ This application makes heavy use of the Google Cast API, which takes care of much of the application's control. See docs: https://developers.google.com/cast/docs/

+ The API is initialized by a call to Google's script, which must take place after a callback to window['__onGCastApiAvailable'] is declared (notice in index.html, that js/initCastPlayer.js is called prior to the API load call).
+ The website on which this web app is loaded **must be https** for this to work. It is a requirement of the Google Cast API for the web host to be https.
+ Furthermore, the **browser must be Chrome**, as far as we can tell. Google Cast uniquely supports Chrome.

Interface:

+ Before clicking a link to load a video, the user must begin a cast session using the cast button or the in-browser cast. A video cannot be loaded if there is no cast session active.
+ Once a cast session has been established, a video is loaded by clicking its corresponding name.
+ Play/Pause, Mute/Unmute, and Seek are buttons. Play/Pause and Mute/Unmute can be toggled, while Seek applies the user's input into the two corresponding boxes (the left being minutes, the right being seconds).
+ Volume input is a percentage, from 0% to 100%.
+ File upload supports *about* 8MB. This seems to be a PHP limit, so with different PHP settings one could most likely up this limit to a desired value.
+ Files are uploaded right when they are selected. It's on a "change" event when the file is POSTed to the PHP.
+ After a file upload, the file will be added to the list of playable media on refresh.
+ Uploads are **not sanitized**, as of now. Proper sanitization should be implemented.

MediaItems/fileReg.db:

+ NAME: The name of the media. This is what is added as the link.
+ HTTP: A server link to the content. This should be an http link, not https (as of right now).
+ TYPE: 0 for mp3, 1 for mp4. This changes how the cast API handles the content (as audio or video).

js/index.js:

+ This will send a GET request to php/getFiles.php.
+ Once php/getFiles.php returns (with, hopefully, a JSON array of media files), loadMediaList is called.
+ loadMediaList loops through the loaded media, and creates a link element for each media item, which, on click, loads the data into the google cast API (which initiates a cast of the content).

js/initCastPlayer.js:

+ This contains a callback definition, tied to window['__onGCastApiAvailable']. This is a special callback function which the Google API will call when it is done with setup.
+ The isAvailable flag is true when the API is correctly loaded, or false if there was some error. A common error happens if the host is not on https; the API will refuse to load if the host is on http.

+ At the end of the callback, startControls() in js/initControls.js is called. This requires the API to be set up, so it is called at the end of the callback.

js/initControls.js:

+ This is mostly a class definition, which takes care of the media playback.
+ The constructor loads each component of the player in from the HTML document. These are loaded in by id. It furthermore creates a repeatedly called function, called every 500 ms by default, which updates the UI to reflect the current state of the media (time, volume, etc.)
+ bindCallbacks() sets up the callback functions for the HTML elements.
+ startControls() initializes the controls by creating a Controls instance and binding their callbacks. This needs to be called after API initialization, and in our app is called right at the end of the “intialized” callback (see js/initCastPlayer.js).

js/uploadFile.js:

+ This simply serves as a “bridge” between php/uploadFile.php and index.html.
+ All this file does is sets an event listener on the file input. Once a change is detected, a POST request is sent to php/uploadFile.php as form data to be uploaded to the server.

php/uploadFile.php:

+ First, this file takes information from the uploaded file for placement within the database, such as where it will be uploaded, the filename, and its extension.
+ Then, the file is moved to its final location on the web server using move_uploaded_file.
+ Finally, the information about the file is registered into MediaItems/fileReg.db using the information.
+ **Notice there is no sanitization. This should be done before serious use in anything.**

php/getFiles.php:

+ Primarily gets files from MediaFiles/fileReg.db. Given that there are not many files in use for this short demonstration, a simple *select * from Files* is used.
+ After these are all retrieved, all of the results are looped over and are echoed as JSON. Because the *header* is declared as *application/json*, this response will be interpreted as a JSON result.