**Juke Testing Document**

Rama Gosula gosula@usc.edu 30393   
Meghan Mehta Mehtam@usc.edu 30393   
Chris Cognetta Ccognett@usc.edu 30393   
Jason Roodman roodman@usc.edu 30393   
Nayha Kamboj nkamboj@usc.edu 26361  
Katie Park Katiepar@usc.edu 30393

Test - Host(jukebox)-Client Connection

* Start Server
  + Steps to Run Test:
    - Run the function to start the server
    - Try to connect to the server
  + Expected Result
    - Different threads can connect to the server
* UpdateThread updates playlist
  + Steps to Run Test:
    - Create a Playlist
    - Create an UpdateThread
    - Populate it with ad-hoc ServerCommands
    - Run the thread
  + Expected Result:
    - The ad hoc ServerCommands are all executed on the playlist associated with the update thread
* UpdateThread gives proper replies
  + Steps to Run Test:
    - Create a Playlist
    - Create an UpdateThread
    - Create three different ad-hoc Threads that access UpdateThreads
    - Call addCommand() multiple times from each UpdateThread, with many different ServerCommands.
  + Expected Result
    - Each ServerCommand receives a reply
    - Each reply corresponds to the ServerCommand that was sent
    - Each reply is given to the correct thread.
    - The playlist is updated by each servercommand correctly
    - Each reply is received in the proper order
* ReceiverThread properly handles commands
  + Steps to Run Test:
    - Create a Playlist
    - Create an updateThread
    - Create a receiver Thread
    - Create mulitple Ad hoc threads that have both the receiver thread’s inputstream and updateThread’s outputstream.
    - Send ServerCommands and ImmediateCommands to the receiver thread via stream
  + Expected Result:
    - Each ServerCommand Receives a Reply
    - ImmediateCommands Receive a Reply
    - Each Reply corresponds to the servercommand sent
    - Each Reply is sent to the proper thread
    - Each Reply is received in the proper order
    - The playlist is updated correctly by each serverCommand
* \*Client’s CommandThread sends commands and receives replies
  + Steps to run test:
    - Connect client and host via socket
    - Create Command Thread
    - Create Playlist
    - Create Update Thread
    - Create Receiver Thread
    - Call AddCommand on updatethread
  + Expected Result:
    - The command is received by the receiver thread
    - The command is either executed by the receiver or passed to the update thread
    - If the command is passed to the update thread it is executed properly
    - The Command Thread receives a proper reply
* Client can send multiple commands
  + Steps to Run Test:
    - \*See Above\*
    - Instead of adding one command, add multiple
  + Expected Result:
    - Each command is executed correctly
    - Each receives a reply
    - The reply fits the command
    - Each receives a reply in the proper order
* Multiple Clients can send commands
  + Steps to Run Test:
    - \*See Above\*
    - Instead of one Command thread, create many
    - Send addCommand() multiple times from each
  + Expected Result:
    - Each command is executed
    - Each command is executed correctly (though maybe not in the exact same order sent)
    - Each command receives the proper reply
    - Each thread receives replies in the order they sent commands

Unit Test - Playlist Class

* Add song to playlist
  + Steps to Run Test:
    - Create playlist object
    - Call addSong() on playlist
  + Expected Result:
    - That song is added to the playlist
* Upvote/Downvote Song
  + Steps to Run Test:
    - Create Playlist Object
    - Add 3 different songs to the playlist
    - Upvote the third song, then downvote the second
  + Expected Result:
    - The third song is upvoted, second is downvoted, first is unchanged.
* Add song already in playlist
  + Steps to run test:
    - Create a playlist object
    - Add multiple songs
    - Add one of the songs a second time
  + Expected result:
    - The song that was added twice has an upvote
* Upvote a song not in the playlist
  + Steps to run Test:
    - Create and add songs to playlist object
    - Upvote, but pass in a song not in the playlist
  + Expected result:
    - No change
* Sorting the to-play list
  + Steps to run test:
    - Create playlist and add songs
    - Upvote and downvote songs so that there is a range of positive and negative values. Make sure there are duplicate values, and multiple unvoted songs.
    - Call sortToPlay()
  + Expected Result:
    - The to-play list is sorted in descending order. Any songs with the same number of votes are in the same relative order as before
* Advance Song
  + Steps to Run Test:
    - Create playlist
    - Add songs
    - upvote/downvote different songs
    - Sort playlist
    - Call advanceSong()
  + Expected Result:
    - The song will be appended to ‘played’
    - The current song is the highest song on the to-play list
    - The song is removed from the to-play list
    - The correct song was returned;
* Advance song empty
  + Steps to run test:
    - Create playlist, but don’t add any songs
    - Call advanceSong()
  + Expected Result
    - No change, return null

Test - Login

* Login from existing user with correct username and password
  + Steps to Run Test:
    - Enter a username/password correctly from the database
  + Expected Result:
    - Normal login, with screen transitioning to the Main Menu
* Login from non-existing user with a username/password combo not present in the database
  + Steps to Run Test:
    - Enter an incorrect username/password
  + Expected Results:
    - Error message visible to user, detailing that either password or username is incorrect
    - User is able to re-enter password
* Login with correct username but incorrect password
  + Steps to Run Test:
    - Enter a correct username from the database but incorrect password
  + Expected Results:
    - Error message visible to user, detailing that either password or username is incorrect
    - User is able to re-enter password
* Login with incorrect username but correct password
  + Steps to Run Test:
    - Enter an incorrect username and a correct password from the database
  + Expected Results:
    - Error message visible to user, detailing that either password or username is incorrect
    - User is able to re-enter password
* SQL code entered as part of username/password
  + Steps to Run Test:
    - Enter “Drop Database” or another ‘sneaky’ command
  + Expected Results:
    - Error message visible to user, detailing that either password or username is incorrect
    - User is able to re-enter password

Test - Sign Up

* Sign Up with valid email, username, and password
  + Steps to Run Test:
    - Enter a valid email, username, and password (2x)
  + Expected Results:
    - Confirmation email sent to user
    - User brought to correct “waiting for confirmation” page
* Sign Up with invalid email, but valid username and password
  + Steps to Run Test:
    - Enter an invalid email, valid username, and valid password
  + Expected Results:
    - Error message visible to user, detailing that some part of information was incorrect
    - User is able to re-enter information
* Sign Up with invalid username, but valid email and password
  + Steps to Run Test:
    - Enter an invalid username, valid email, and valid password
  + Expected Results:
    - Error message visible to user, detailing that some part of information was incorrect
    - User is able to re-enter information
* Sign Up with invalid password, but valid email and username
  + Steps to Run Test:
    - Enter an invalid password, valid email, and valid username
  + Expected Results:
    - Error message visible to user, detailing that some part of information was incorrect
    - User is able to re-enter information
* Sign Up with missing information (user doesn’t enter information into all fields)
  + Steps to Run Test:
    - Enter partial information into Sign Up
  + Expected Results:
    - Error message detailing that all information was not provided
    - User is able to re-enter information
* Sign Up with non-matching password and password confirmation fields
  + Steps to Run Test:
    - Enter valid information but differing passwords into the password and password confirmation fields
  + Expected Results:
    - Error message detailing that passwords differed
    - User is able to re-enter information
* SQL code entered as part of new user info
  + Steps to Run Test:
    - Enter “Drop Database” or another ‘sneaky’ command
  + Expected Results:
    - Error message visible to user, detailing that either password or username is incorrect
    - User is able to re-enter information

Test - Start Party

* User attempts to Start Party with missing information
  + Steps to Run Test:
    - Enter partial information
  + Expected Results:
    - Error message visible to user detailing that information is
* User attempts to Start Party with correct network connection
  + Steps to Run Test:
    - Click Start Party while connected to the internet
  + Expected Results:
    - Correct transition to Playlist Main Menu
* User attempts to Start Party with incorrect network connection
  + Steps to Run Test:
    - Click Start Party while not connected to the internet
  + Expected Results:
    - Error message detailing that there is no network connection
    - User remains on same screen
* User enters invalid password
  + Steps to Run Test:
    - Enter invalid password (all other information correct)
  + Expected Results:
    - Error message detailing invalid password
    - User is able to re-enter information
* SQL code entered as part of playlist info
  + Steps to Run Test:
    - Enter “Drop Database” or another ‘sneaky’ command
  + Expected Results:
    - Error message visible to user, detailing that either password or username is incorrect
    - User is able to re-enter information

Test - Location

* User cannot see parties on their map
  + Steps to Run Test:
    - Have location turned off and run the location section
  + Expected Results:
    - Pop will display telling user to make sure location is turned on
* User’s location is incorrect and they cannot see the nearby parties
  + Steps to Run Test:
    - Run the location search with low power GPS
  + Expected Results:
    - Nothing will show up on map, and the user can manually enter an address to search
* User can see parties on their map, but not the one they’re looking for
  + Steps to Run Test:
    - Try to join a party via location and the party cannot be found on the map because the host has their location off
  + Expected Results:
    - Suggests to find the party another way. If party is found another way, then suggests the host of the party to turn their location on.

Test - QR Code

* User scan QR code to join a party
  + Steps to Run Test:
    - Have another user generate a party and QR code so it is scannable
  + Expected Results:
    - Once scanned, the user will automatically be brought to the party page
* User scan QR code to join a party but cannot connect
  + Steps to Run Test:
    - Have a user scan a QR code and turn off wifi midway through connecting
  + Expected Results:
    - Pop up will show that explains it could not connect and to double check network settings before re-trying
* User scans QR code, but a party cannot be found
  + Steps to Run Test:
    - A user tries to join a party via the QR code knowing that a current live party doesn’t exist with that QR code (eg. tries an old code)
  + Expected Results:
    - App suggests the user make sure it’s the right code and if that doesn’t work, try the other 2 methods of joining a party.

Test - Playlist Generating

* User wants to generate the current playlist
  + Steps to Run Test:
    - Update the playlist by adding new songs
  + Expected Results:
    - When the refresh button is pressed, the list will be generated sorted by upvotes
* User loses connection
  + Steps to Run Test:
    - Turn off wifi with the playlist open on the phone
  + Expected Results:
    - When the refresh button is pressed, a pop up will show indicating lack of network connectivity

Test - Join Party

* User is able to successfully join a party
  + Steps to run test:
    - Create a playlist
    - create a new user/login
    - Try to join a party/playlist
  + Expected Results:
    - new user is able to join playlist for a party
* User attempts to join a party that no longer exists
  + Steps to run test:
    - User tries to join a party that has just been deleted
  + Expected Results:
    - An error message pops up letting the user know that the party no longer exists
* User attempts to join a party (is unsuccessful)
  + Steps to run test:
    - Create a Party
    - Create an account, user attempts to join a party
  + Expected Results:
    - Error message telling the user that they were unsuccessful in joining the party

Test - My Playlists/Parties

* User is able to access saved playlists
  + steps to run test:
    - Create 2 playlists
    - Save the first playlist but not the second
  + Expected Results:
    - User is able to access the first playlist but not the second
* User has host control of playlist
  + steps to run test:
    - Create a Party
    - Check that the user can add/remove songs in addition to voting
  + Expected Results:
    - host can remove songs whereas guest/non-hosts can only add and vote on songs
* User tries to create a playlist as a guest
  + steps to run test:
    - Go on the app as a guest
    - Try to create a playlist
  + Expected Results:
    - User is not able to create a playlist without a username/password/account